



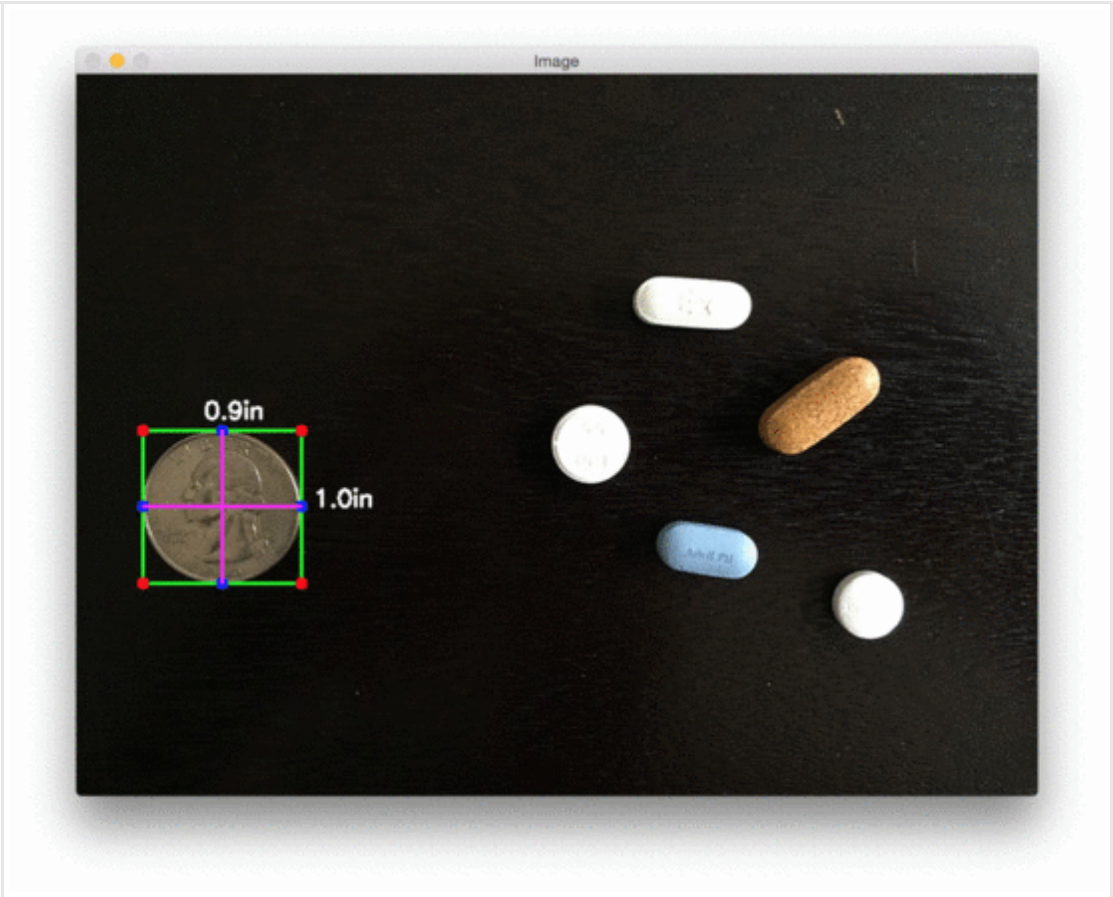
Measuring size of objects in an image with OpenCV

by **Adrian Rosebrock** on March 28, 2016 in **Image Processing, Tutorials**

Like 65

G+1 9

67



Measuring the size of an object (or objects) in an image has been a *heavily requested tutorial* on the PyImageSearch blog for some time now — and it feels *great* to get this post online and share it with you.

Today’s post is the second in a three part series on *measuring the size of objects in an image* and *computing the distances between them*.

[Last week](#), we learned an important technique: how *reliably* order a set of rotated bounding box coordinates in a top-left, top-right, bottom-right, and bottom-left arrangement.

Today we are going to utilize this technique to aid us in computing the size of objects in an image. **Be sure to read the entire post to see how it’s done!**

Looking for the source code to this post?
[Jump right to the downloads section.](#)

Measuring the size of objects in an image with OpenCV

Measuring the size of objects in an image is similar to [computing the distance from our camera to an object](#) — in both cases, we need to define a ratio that measures the number of pixels per a given metric.

I call this the “pixels per metric” ratio, which I have more formally defined in the following section.

The “pixels per metric” ratio

In order to determine the size of an object in an image, we first need to perform a “calibration” (not to be confused with [intrinsic/extrinsic calibration](#)) using a reference object. Our reference object should have two important properties:

- **Property #1:** We should know the *dimensions of this object* (in terms of width or height) in a *measurable unit* (such as millimeters, inches, etc.).
- **Property #2:** We should be able to easily find this reference object in an image, either based on the *placement* of the object (such as the

reference object always being placed in the top-left corner of an image) or via *appearances* (like being a distinctive color or shape, unique and different from all other objects in the image). In either case, our reference should be *uniquely identifiable* in some manner.

In this example, we'll be using the United States quarter as our reference object and throughout all examples, ensure it is always the *left-most* object in our image:



Figure 1: We'll use a United States quarter as our reference object and ensure it is always placed as the left-most object in the image, making it easy for us to extract it by sorting contours based on their location.

By guaranteeing the quarter is the left-most object, we can sort our object contours from left-to-right, grab the quarter (which will always be the first contour in the sorted list), and use it to define our *pixels_per_metric*, which we define as:

$$pixels_per_metric = object_width / know_width$$

A US quarter has a *known_width* of **0.955 inches**. Now, suppose that our *object_width* (measured in pixels) is computed be 150 pixels wide (based on its associated bounding box).

The *pixels_per_metric* is therefore:

$$pixels_per_metric = 150px / 0.955in = 157px$$

Thus implying there are approximately 157 pixels per every 0.955 inches in our image. Using this ratio, we can compute the size of objects in an image.

Measuring the size of objects with computer vision

Now that we understand the “pixels per metric” ratio, we can implement the Python driver script used to measure the size of objects in an image.

Open up a new file, name it `object_size.py` , and insert the following code:

Measuring size of objects in an image with OpenCV	Python
<pre>1 # import the necessary packages 2 from scipy.spatial import distance as dist 3 from imutils import perspective 4 from imutils import contours 5 import numpy as np 6 import argparse 7 import imutils 8 import cv2 9 10 def midpoint(ptA, ptB): 11 return ((ptA[0] + ptB[0]) * 0.5, (ptA[1] + ptB[1]) * 0.5) 12 13 # construct the argument parse and parse the arguments 14 ap = argparse.ArgumentParser() 15 ap.add_argument("-i", "--image", required=True, 16 help="path to the input image") 17 ap.add_argument("-w", "--width", type=float, required=True, 18 help="width of the left-most object in the image (in inches)") 19 args = vars(ap.parse_args())</pre>	<div>Free 21-day crash course on computer vision & image search engines</div>

Lines 2-8 import our required Python packages. We'll be making heavy use of the `imutils` package in this example, so if you don't have it installed, make sure you install it before proceeding:

Measuring size of objects in an image with OpenCV	Shell
<pre>1 \$ pip install imutils</pre>	

Otherwise, if you *do* have `imutils` installed, ensure you have the latest version, which is `0.3.6` at the time of this writing:

Measuring size of objects in an image with OpenCV	Shell
<pre>1 \$ pip install --upgrade imutils</pre>	

Lines 10 and 11 defines a helper method called `midpoint` , which as the name suggests, is used to compute the `midpoint` between two sets of (x, y)-coordinates.

We then parse our command line arguments on **Lines 14-19**. We require two arguments, `--image` , which is the path to our input image containing the objects we want to measure, and `--width` , which is the width (in inches) of our reference object, presumed to be the left-most object in our `--image` .

We can now load our image and preprocess it:

Measuring size of objects in an image with OpenCV	Python
<pre>21 # load the image, convert it to grayscale, and blur it slightly 22 image = cv2.imread(args["image"]) 23 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) 24 gray = cv2.GaussianBlur(gray, (7, 7), 0) 25 26 # perform edge detection, then perform a dilation + erosion to 27 # close gaps in between object edges 28 edged = cv2.Canny(gray, 50, 100) 29 edged = cv2.dilate(edged, None, iterations=1) 30 edged = cv2.erode(edged, None, iterations=1) 31 32 # find contours in the edge map 33 cnts = cv2.findContours(edged.copy(), cv2.RETR_EXTERNAL, 34 cv2.CHAIN_APPROX_SIMPLE) 35 cnts = cnts[0] if imutils.is_cv2() else cnts[1] 36 37 # sort the contours from left-to-right and initialize the 38 # 'pixels per metric' calibration variable 39 (cnts, _) = contours.sort_contours(cnts) 40 pixelsPerMetric = None</pre>	

Lines 22-24 load our image from disk, convert it to grayscale, and then smooth it using a Gaussian filter. We then perform edge detection along with a dilation + erosion to close any gaps in between edges in the edge map (**Lines 28-30**).

Lines 33-35 find contours (i.e., the outlines) that correspond to the objects in our edge map.

These contours are then sorted from left-to-right (allowing us to extract our reference object) on **Line 39**. We also initialize our `pixelsPerMetric` value on **Line 40**.

The next step is to examine each of the contours:

Measuring size of objects in an image with OpenCV	Python
<pre>42 # loop over the contours individually 43 for c in cnts: 44 # if the contour is not sufficiently large, ignore it 45 if cv2.contourArea(c) < 100: 46 continue 47 48 # compute the rotated bounding box of the contour 49 orig = image.copy() 50 box = cv2.minAreaRect(c) 51 box = cv2.cv.BoxPoints(box) if imutils.is_cv2() else cv2.boxPoints(box) 52 box = np.array(box, dtype="int") 53 54 # order the points in the contour such that they appear 55 # in top-left, top-right, bottom-right, and bottom-left 56 # order, then draw the outline of the rotated bounding 57 # box 58 box = perspective.order_points(box) 59 cv2.drawContours(orig, [box.astype("int")], -1, (0, 255, 0), 2) 60 61 # loop over the original points and draw them 62 for (x, y) in box: 63 cv2.circle(orig, (int(x), int(y)), 5, (0, 0, 255), -1)</pre>	

On **Line 43** we start looping over each of the individual contours. If the contour is not sufficiently large, we discard the region, presuming it to be noise left over from the edge detection process (**Lines 45 and 46**).

Provided that the contour region is large enough, we compute the rotated bounding box of the image on **Lines 50-52**, taking special care to use the `cv2.cv.BoxPoints` function for OpenCV 2.4 and the `cv2.boxPoints` method for

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

We then arrange our rotated bounding `box` coordinates in top-left, top-right, bottom-right, and bottom-left order, [as discussed in last week's blog post \(Line 58\)](#).

Lastly, **Lines 59-63** draw the *outline* of the object in *green*, followed by drawing the *vertices* of the bounding box rectangle in as *small, red circles*.

Now that we have our bounding box ordered, we can compute a series of midpoints:

Measuring size of objects in an image with OpenCV	Python
<pre>65 # unpack the ordered bounding box, then compute the midpoint 66 # between the top-left and top-right coordinates, followed by 67 # the midpoint between bottom-left and bottom-right coordinates 68 (tl, tr, br, bl) = box 69 (tltrX, tltrY) = midpoint(tl, tr) 70 (blbrX, blbrY) = midpoint(bl, br) 71 72 # compute the midpoint between the top-left and top-right points, 73 # followed by the midpoint between the top-right and bottom-right 74 (tlblX, tlblY) = midpoint(tl, bl) 75 (trbrX, trbrY) = midpoint(tr, br) 76 77 # draw the midpoints on the image 78 cv2.circle(orig, (int(tltrX), int(tltrY)), 5, (255, 0, 0), -1) 79 cv2.circle(orig, (int(blbrX), int(blbrY)), 5, (255, 0, 0), -1) 80 cv2.circle(orig, (int(tlblX), int(tlblY)), 5, (255, 0, 0), -1) 81 cv2.circle(orig, (int(trbrX), int(trbrY)), 5, (255, 0, 0), -1) 82 83 # draw lines between the midpoints 84 cv2.line(orig, (int(tltrX), int(tltrY)), (int(blbrX), int(blbrY)), 85 (255, 0, 255), 2) 86 cv2.line(orig, (int(tlblX), int(tlblY)), (int(trbrX), int(trbrY)), 87 (255, 0, 255), 2)</pre>	

Lines 68-70 unpacks our ordered bounding box, then computes the midpoint between the top-left and top-right points, followed by the midpoint between the bottom-right points.

We'll also compute the midpoints between the top-left + bottom-left and top-right + bottom-right, respectively (**Lines 74 and 75**).

Lines 78-81 draw the *blue* midpoints on our `image` , followed by connecting the midpoints with *purple* lines.

Next, we need to initialize our `pixelsPerMetric` variable by investigating our reference object:

Measuring size of objects in an image with OpenCV	Python
<pre>89 # compute the Euclidean distance between the midpoints 90 dA = dist.euclidean((tltrX, tltrY), (blbrX, blbrY)) 91 dB = dist.euclidean((tlblX, tlblY), (trbrX, trbrY)) 92 93 # if the pixels per metric has not been initialized, then 94 # compute it as the ratio of pixels to supplied metric 95 # (in this case, inches) 96 if pixelsPerMetric is None: 97 pixelsPerMetric = dB / args["width"]</pre>	

First, we compute the Euclidean distance between the our sets of midpoints (**Lines 90 and 91**). The `dA` variable will contain the *height* distance (in pixels) while `dB` will hold our *width* distance.

We then make a check on **Line 96** to see if our `pixelsPerMetric` variable has been initialized, and if it hasn't, we divide `dB` by our supplied `--width` , thus giving us our (approximate) pixels per inch.

Now that our `pixelsPerMetric` variable has been defined, we can measure the size of objects in an image:

Measuring size of objects in an image with OpenCV	Python
<pre>99 # compute the size of the object 100 dimA = dA / pixelsPerMetric 101 dimB = dB / pixelsPerMetric 102 103 # draw the object sizes on the image 104 cv2.putText(orig, "{:.1f}in".format(dimA), 105 (int(tltrX - 15), int(tltrY - 10)), cv2.FONT_HERSHEY_SIMPLEX, 106 0.65, (255, 255, 255), 2) 107 cv2.putText(orig, "{:.1f}in".format(dimB), 108 (int(trbrX + 10), int(trbrY)), cv2.FONT_HERSHEY_SIMPLEX, 109 0.65, (255, 255, 255), 2) 110 111 # show the output image 112 cv2.imshow("Image", orig) 113 cv2.waitKey(0)</pre>	

Lines 100 and 101 compute the dimensions of the object (*in inches*) by dividing the respective Euclidean distances by the `pixelsPerMetric` value (see the “**Pixels Per Metric**” section above for more information on why this ratio works).

Lines 104-109 draw the dimensions of the object on our `image` , while **Lines 112 and 113** show the image.

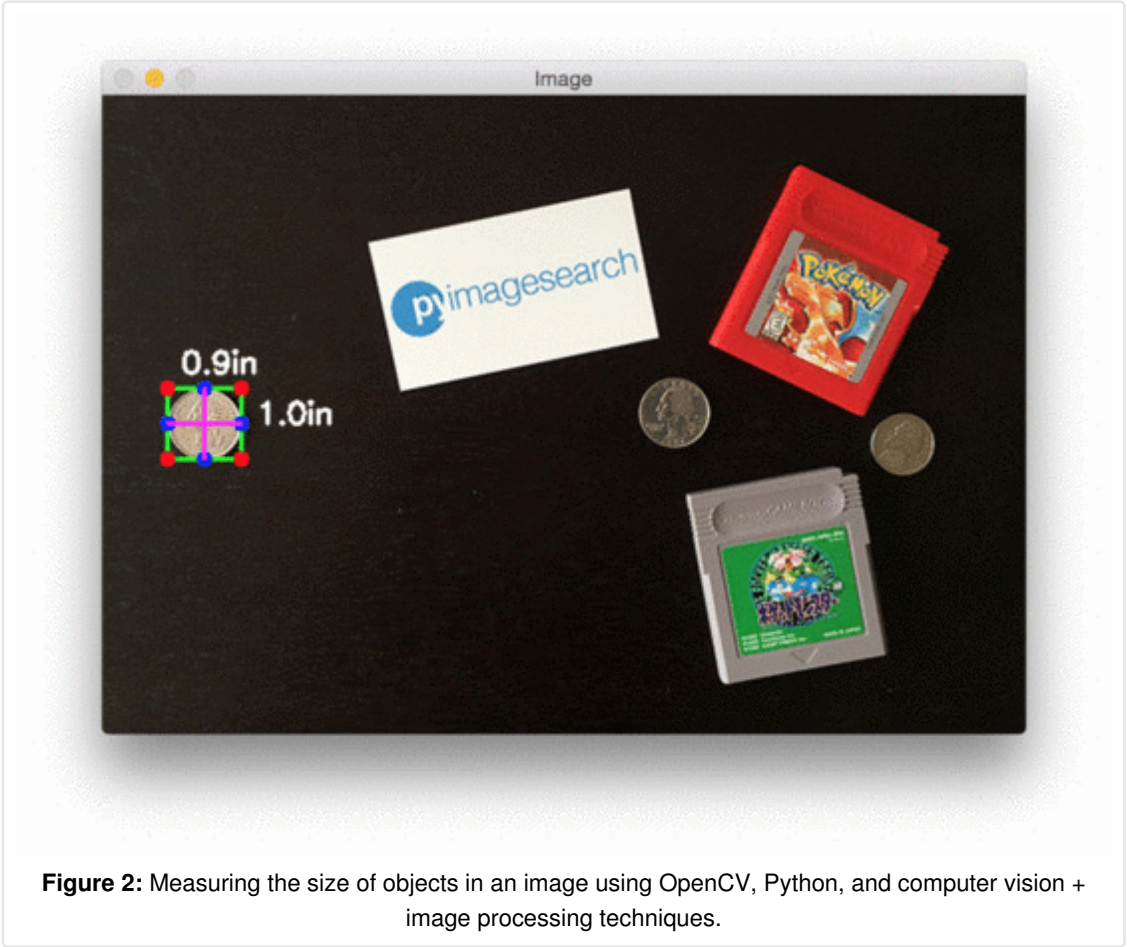
Free 21-day crash course on computer vision & image search engines

Object size measuring results

To test our `object_size.py` script, just issue the following command:

Measuring size of objects in an image with OpenCV	Shell
1 \$ python object_size.py --image images/example_01.png --width 0.955	

Your output should look something like the following:



As you can see, we have successfully computed the size of each object in an our image — our business card is correctly reported as *3.5in x 2in*. Similarly, our nickel is accurately described as *0.8in x 0.8in*.

However, not all our results are *perfect*.

The Game Boy cartridges are reported as having slightly different dimensions (even though they are the same size). The height of both quarters are also off by *0.1in*.

So why is this? How come the object measurements are not 100% accurate?

The reason is two-fold:

1. First, I hastily took this photo with my iPhone. The angle is most certainly *not* a perfect 90-degree angle “looking down” (like a birds-eye-view) at the objects. Without a perfect 90-degree view (or as close to it as possible), the dimensions of the objects can appear distorted.
2. Second, I did not calibrate my iPhone using the intrinsic and extrinsic parameters of the camera. Without determining these parameters, photos can be prone to [radial and tangential lens distortion](#). Performing an extra calibration step to find these parameters can “un-distort” our image and lead to a better object size approximation (but I’ll leave the discussion of distortion correction as a topic of a future blog post).

In the meantime, strive to obtain as close to a 90-degree viewing angle as possible when taking photos of your objects — this will help increase the accuracy of your object size estimation.

That said, let’s look at a second example of measuring object size, this time measuring the dimensions of pills:

Measuring size of objects in an image with OpenCV	Shell
1 \$ python object_size.py --image images/example_02.png --width 0.955	

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

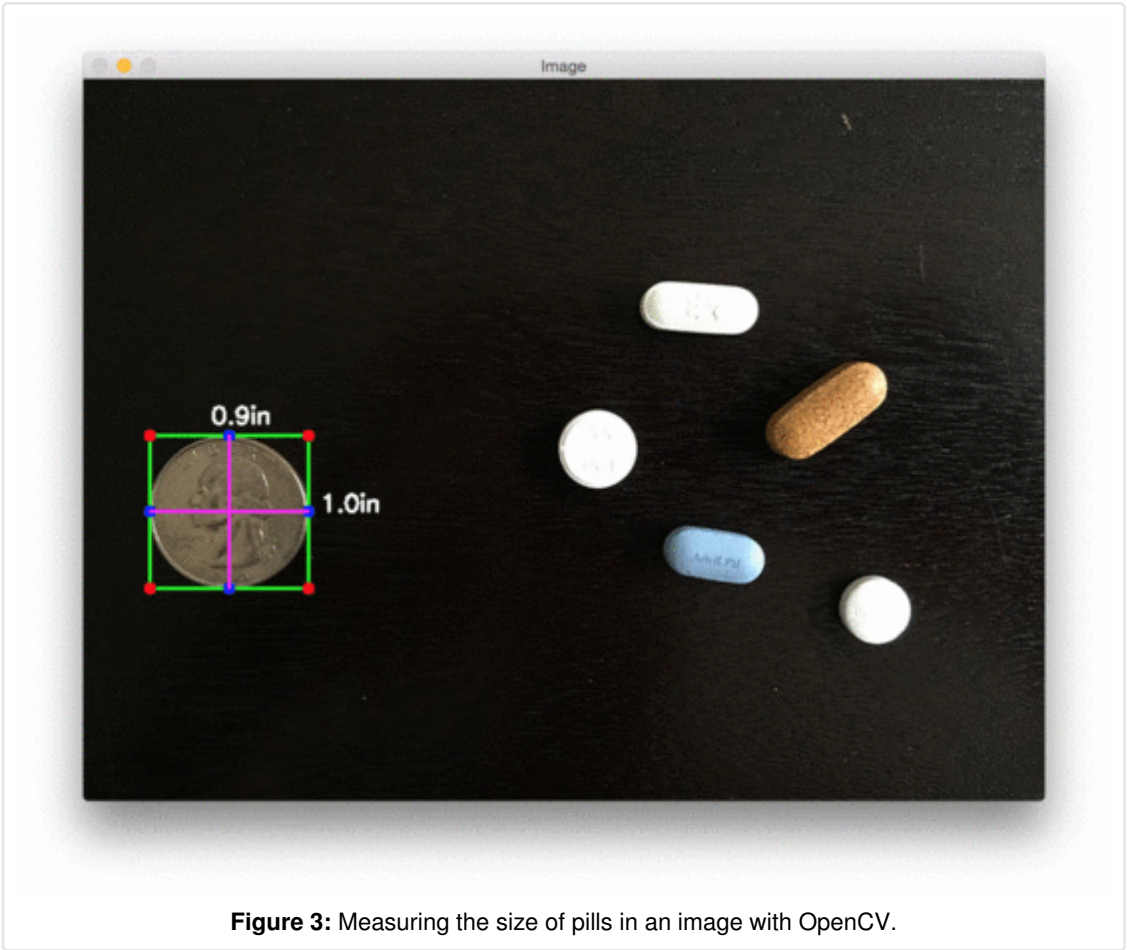


Figure 3: Measuring the size of pills in an image with OpenCV.

Nearly 50% of all 20,000+ prescription pills in the United States are *round* and/or *white*, thus if we can filter pills based on their measurements, we stand a better chance at accurately identification the medication.

Finally, we have a final example, this time using a 3.5in x 2in business card to measure the size of two vinyl EPs and an envelope:

Measuring size of objects in an image with OpenCV	Shell
1 \$ python object_size.py --image images/example_03.png --width 3.5	

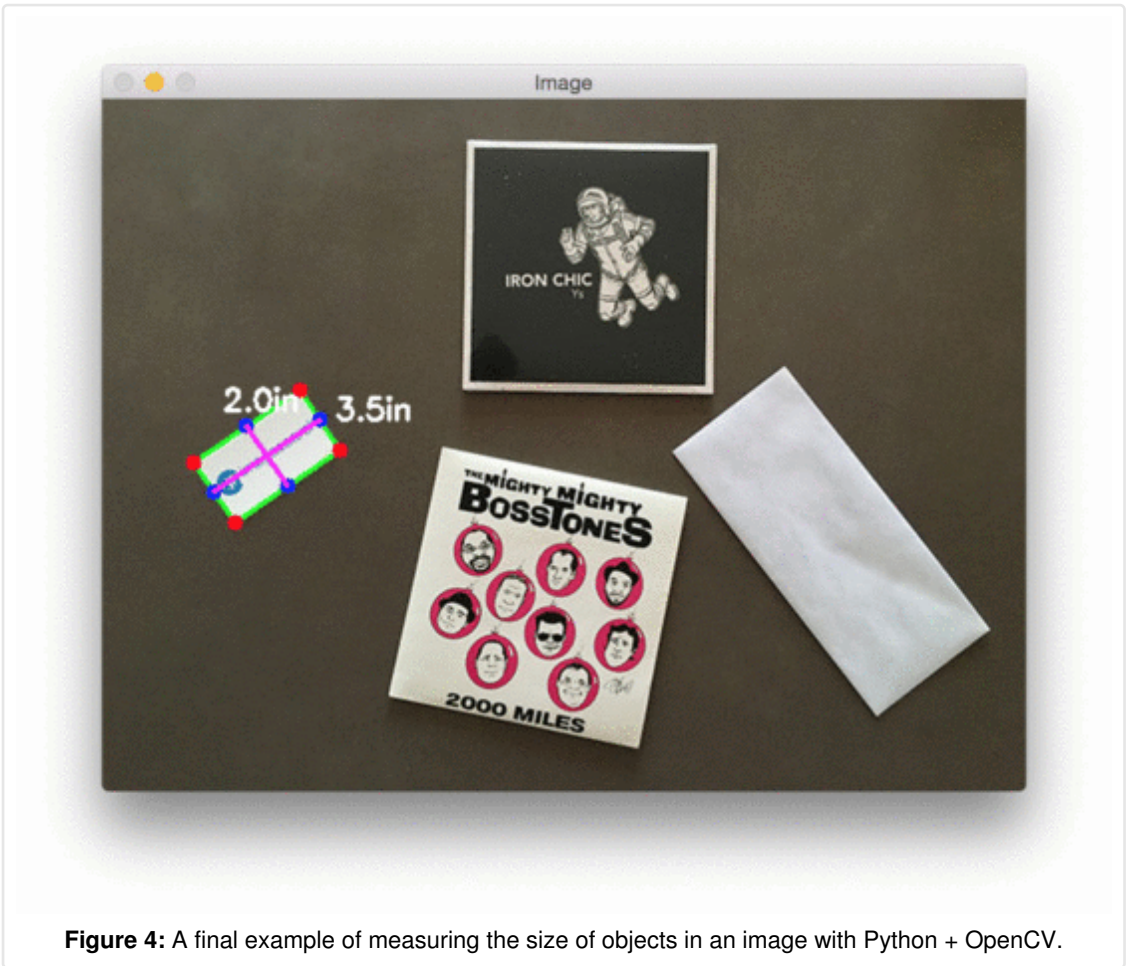


Figure 4: A final example of measuring the size of objects in an image with Python + OpenCV.

Again, the results aren't quite perfect, but this is due to (1) the viewing angle and (2) lens distortion, as mentioned above.

Summary

In this blog post, we learned how to measure the size of objects in an image using Python and OpenCV.

Just like in our tutorial on [measuring the distance from a camera to an object](#), we need to determine our “pixels per metric” ratio, which describes the number of pixels that can “fit” into a given number of inches, millimeters, meters, etc.

To compute this ratio, we need a reference object with two important properties:

- **Property #1:** The reference object should have *known dimensions* (such as width or millimeters, etc.).

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

- **Property #2:** The reference object should be *easy to find*, either in terms of *location* of the object or in its *appearance*.

Provided that both of these properties can be met, you can utilize your reference object to calibrate your *pixels_per_metric* variable, and from there, compute the size of other objects in an image.

In our next blog post, we'll take this example a step further and learn how to compute the distance ***between*** objects in an image.

Ne sure to signup for the PyImageSearch Newsletter using the form below to be notified when the next blog post goes live — *you won't want to miss it!*

Downloads:



If you would like to download the code and images used in this post, please enter your email address in the form below. Not only will you get a .zip of the code, I'll also send you a **FREE 11-page Resource Guide** on Computer Vision and Image Search Engines, including **exclusive techniques** that I don't post on this 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!

Resource Guide (it's totally free).



Enter your email address below to get my **free 11-page Image Search Engine Resource Guide PDF**. Uncover **exclusive techniques** that I don't publish on this blog and start building image search engines of your own!

Your email address

DOWNLOAD THE GUIDE!

🔖 calibration, image processing, object size, pixels per metric, point ordering

< Ordering coordinates clockwise with Python and OpenCV

Measuring distance between objects in an image with OpenCV >

108 Responses to *Measuring size of objects in an image with OpenCV*



Gadget Steve March 28, 2016 at 11:39 am #

REPLY ↩

Nice write-up as always Adrian but you have missed mentioning directly the prerequisite that all of the objects to be measured be co-planar with the reference object even though you did mention that the camera must be at as near as possible 90 degrees to that plane.

To do size and or distance calculations without a reference object you need a) the lens and focus information and b) two or more images of the same scene from different angles so as to use parallax calculations.



Adrian Rosebrock March 28, 2016 at 1:28 pm #

REPLY ↩

Exactly. In a future blog post, I'll detailing how to calibrate your cameras to obtain these parameters and thus undistort the image.



vigneshwer dhinakaran March 28, 2016 at 11:55 pm #

REPLY ↩

Steve, Calibration can be done by taking a top view image and a image of y find a homography matrix by which you wrap perspective the side view image to top

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

will need to find similar points in both the top view reference and input image of different plane, which can be done by finding the sift point followed by a matcher.

You can look up to Adrian's blog on making panoramic images for reference.

Only problem I see is to find the pixels per metric ratio value by giving an width value of the ref image, compensating for the camera distance from the image. Is there any way to solve the camera distance problem in an automated fashion?



Ivan March 29, 2016 at 9:15 am #

REPLY ↩

Maybe 4 square marks in the corners of image cold be useful. The camera is well calibrated if aspect ratio of all marks is the same.



rahulagnihotri25@hotmail.com March 28, 2016 at 11:50 am #

REPLY ↩

Hi,

It is a really helpful article..Thanks for it.

I am currently working on identifying human body and then get the length, width and circumference of the body parts from top to bottom.

I have been able to identify the human body but getting is the measurement is an issue.

Can you please help.

Regards,



Adrian Rosebrock March 28, 2016 at 1:27 pm #

REPLY ↩

So if I understand your question correctly, your goal is to (1) detect the body, (2) extract each body part, and then (3) measure each body part? If so, you should do some research on skeletal models for the body, and perhaps even utilizing a stereo camera to help in the segmentation of each body part.



Ivan March 28, 2016 at 12:29 pm #

REPLY ↩

Wonderful, thank you very much. I think, I can apply this method to measure cells on microscope images. I already have the pixel ratio in EXIF from microscope, so I'll declare it as a constant. Maybe you'll suggest something more optimal for round and ellipsoid objects like cells? I think I should find the smallest and the largest diameter. After this I can find the excentricity.



Adrian Rosebrock March 28, 2016 at 1:26 pm #

REPLY ↩

In that case, you might want to compute the minimum enclosing circle or the minimum enclosing ellipse, then compute your ratios based on that. Also, you might want to look into [ImageJ](#) which is heavily used for microscopic images — I even did some work for the National Cancer Institute developing plugins for ImageJ.



Ivan March 29, 2016 at 1:39 am #

REPLY ↩

Thank you. I use it with CellProfiler, but I often need to develop some new functions. So I use the scipy most of the time for image analysis. Now I want to try the opencv. Thanks again for your blog. Useful tutorials.



Adrian Rosebrock March 29, 2016 at 7:01 am #

REPLY ↩

SciPy is definitely quite useful, but I think you'll find that OpenCV makes it even easier, once you get the hang of it ☐



Ankit March 28, 2016 at 5:53 pm #

REPLY ↩

Great Article Adrian!!

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



Adrian Rosebrock March 28, 2016 at 6:20 pm #

REPLY ↩

Thanks Ankit!



muhammad iqbal March 28, 2016 at 8:52 pm #

REPLY ↩

hi its really helpful and very cleared explained



Clement March 29, 2016 at 9:44 am #

REPLY ↩

Thank you for this post. I'm working on this problem and i found a solution for measuring objects using the focale distance. Obviously this can only work in smartphone api so i've been learning java to perform it and i think i'm not far from a good result. The real problem is that we need a camera with a motorised focale.



Kevin Hainsworth March 29, 2016 at 1:46 pm #

REPLY ↩

This may be considered to be a bit “off the wall” but I have 3 table tennis balls (known size) positioned at the interface between the ceiling and the wall of a room. If I have an image that shows all the balls am I correct in thinking that I can compute the distance between the balls and also the distance from the camera to each of the balls? I appreciate that lens distortion may have some impact but I'm not interested in that here as I only want to obtain an approx. room size. Assuming that the ceiling is level I would like to expand this idea to include adjacent walls and floors i.e. to generate a 3D CAD model of the room.



Adrian Rosebrock March 29, 2016 at 3:36 pm #

REPLY ↩

Yes, you are correct in your thinking. If you can segment the 3 balls from your image and you know their size, you can absolutely use this technique to compute the distance between them (I cover how to do this in next weeks blog post). You'll also have to perform a calibration for the [distance from ball to camera](#), but again, that's easily doable.



Kyle W. April 1, 2016 at 10:16 pm #

REPLY ↩

Nice! Thank you for this post, it is always a pleasure to read through your tutorials and learn something new. Alternatively to this method, if you are interested in calculating the area of an object instead of retrieving its dimensions, there is a handy `contourArea` function in openCV! So once the contours are defined, you can easily calculate the area of each contour enclosure in pixels (1 line of code) and convert to appropriate dimensions via a reference object area. This is especially useful for the blob-like objects I am working with which have poorly defined dimensions.



Adrian Rosebrock April 3, 2016 at 10:31 am #

REPLY ↩

Nice tip Kyle, thanks for sharing!



Marcus W. April 28, 2016 at 3:36 am #

REPLY ↩

Hi. Thats a really nice post. I recently started programming, but i wanted to adapt this programm a bit. How can i change it that way, that my referenceobject is detectet by color?
I hope you can help me 😊

Please excuse my english, i am no native.




Adrian Rosebrock April 28, 2016 at 2:27 pm #

REPLY ↩

To detect an object based on color, I would [start with this blog post](#). I also cover object detection inside [Practical Python and OpenCV](#).

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



Alberto April 28, 2016 at 7:46 am <#>

REPLY ↩

Hi there Adrian, awesome site.

I cannot import imutils and scipy modules.


I think this happens because the comand “sudo pip install imutils” installs it at the route “/usr/local/lib/python2.7/dist-packages” instead of “site-packages” because i can see it exploring the directories but i dont really know if this is correct or no.

I tried to install the modules outside the virtual environment and inside it, previously entering the commands “source ~/.profiles” and “workon cv”.

Iam working in a raspbian jessie fresh install on a raspberry pi 3 and i followed your new openCV 3 installation tutorial without any error.

Please let me know how could i change the imutils/scipy installation directory or what other error i am committing.

Thanks in advance.




Adrian Rosebrock April 28, 2016 at 2:25 pm <#>

REPLY ↩

The reason you are getting this error is because you’re using sudo and pip to install imutils and SciPy into the *system install* of Python. Instead, you should be installing them into the *virtual environment*:


```
1 $ workon cv # or whatever your virtual environment is named
2 $ pip install imutils
3 $ pip install scipy
```



Alberto April 29, 2016 at 4:16 am <#>

REPLY ↩

Thanks for the reply, it seems that the command pip install imutils/SciPy in the cv environment now works after a reboot.




ravi May 25, 2016 at 5:15 am <#>

REPLY ↩

Thank you so much for your great effort..

I have a question, Is it possible to measure the size of an object without the reference object in place.


Is it possible ..?



Adrian Rosebrock May 25, 2016 at 3:21 pm <#>

REPLY ↩

You need to perform some sort of camera calibration. That can be done offline before the script is deployed (provided you’re using the same camera sensors) or you can include the reference object in the image. In either case, the system must be calibrated first.



Ebad Syed May 26, 2016 at 1:44 am <#>


REPLY ↩

I want to extract the largest contour.

I am actually calculating size of a box, however there is another perfect contour inside the box.

So it detects both the squares. And I only want the outside one.

What should I do ?



Adrian Rosebrock May 26, 2016 at 6:20 am <#>

REPLY ↩

You simply need to find the contour with the largest area:

```
c = max(cnts, key=cv2.contourArea)
```

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



EBAD URRAHMAN SYED May 28, 2016 at 11:08 am #

REPLY ↩

thank you so muchhhh !



Juan May 29, 2016 at 8:37 am #

REPLY ↩

Hi. Thanks for this really nice post.
I recently started programming with python, but i wanted to adapt this programm a bit to my situation.
I have a set of images that represent vehicles and I want to classify them in groups based on vehicle height.
How can I do this knowing that I don't have this reference object in place?

Thanks in advance.



Adrian Rosebrock May 29, 2016 at 1:51 pm #

REPLY ↩

At some point, you need to calibrate your camera. You can do this “offline” before you take photos of your vehicles (such as computing the reference object focal length and then removing the reference object). But at some point the camera has to be calibrated.



Peng Fei June 2, 2016 at 9:29 am #

REPLY ↩

Hi Adrian,

Thank you so much for the post! It's really interesting. Any advices for measuring the size of an object in real time?



Adrian Rosebrock June 3, 2016 at 3:07 pm #

REPLY ↩

You can use this same approach to measure the size of objects in real-time. Just wrap the around a method to access a video stream, [such as this one](#).



Themba June 3, 2016 at 8:05 am #

REPLY ↩

Hi Adrian, thanks for the advice and this tutorial. I managed to use this tutorial to calibrate my object tracking project.

Actually, the computer vision bundle helped me to improve my programming skills. And I want to take pyimagesearch gurus course but my problem is that I do not have a good programming background, so it might take me some time.



Adrian Rosebrock June 3, 2016 at 2:57 pm #

REPLY ↩

No problem at all Themba, I'm happy that myself/the PyImageSearch community was able to help on your journey ☐



Themba June 10, 2016 at 8:52 am #

REPLY ↩

Hi Adrian, I have been experimenting with different images and different objects within the image. I noticed that sometimes it does not detect some object, so to make it detect the object I varied the GaussainBlur() values from (7,7) to (11,11). However, still when I do that it does not detect all the object in the image.

So my question is how can I ensure that I detect all of the objects with the image ?



Adrian Rosebrock June 12, 2016 at 9:39 am #

REPLY ↩

As you noted, detecting *all* objects in your image is highly dependent on (1) the image, (2) the contents of the image, and (3) your lighting conditions/environment. It's hard to say which (or even all 3) is contributing to the issue here. All I can say is that you should continue to play with the parameters — but also keep in mind that for certain objects, you won't be able to detect them using basic image processing techniques. Some objects require machine learning and/or *object detection*

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



Piet Visser June 20, 2016 at 2:30 am #

REPLY ↩

Hello Adrian,

First of all my sincere respect for what you're doing.
I'm totally new at this and just started to work on measuring 3D-objects very recently.
My ultimate goal is to be able to get X, Y and Z dimension from a picture taken by a RPi V2-type camera, but the first issue to overcome is how to calibrate the camera.
I think the correct procedure would be to take several pictures from an object with known dimensions on different distances and count pixels to find the ratio formula, correct?
Any other suggestions?



Adrian Rosebrock June 20, 2016 at 5:27 pm #

REPLY ↩

3D is an entirely different world (no pun intended). I honestly don't have much experience in 3D, my main area of expertise focuses on image search engines and image classification (and even lately, deep learning). Essentially, you would need to create a 3D model of your object from 2D images. And at that point, you'll be doing much more complex camera calibration anyway, which would lead to better accuracy.



Bilal Khan June 29, 2016 at 6:37 am #

REPLY ↩

Which IDE is Best to try this code



Adrian Rosebrock June 29, 2016 at 1:59 pm #

REPLY ↩

I personally like either Sublime Text 2 or the [PyCharm](#).



neo September 2, 2016 at 7:11 am #

REPLY ↩

wonderful post ! Explained well
Infact I'm working on a project which detects the dimensions of our dress using camera .
This helps a lot
thanks



Ramanan September 13, 2016 at 10:23 am #

REPLY ↩

I really find your post interesting!
I am doing a project where there is a conveyor on which boxes(cuboid shape) moves There is a 3d camera mounted on top to recognize these boxes and further we need to determine the dimensions and pose of the boxes.

I have been studying and researching a lot on these topics and what algorithms to choose and how to proceed.To be brief i am unable to proceed next.

My task has the following constraints:

1) There should be no database of models created from which one can perform feature matching to detect objects.
The algorithm used for object detection should directly detect the object and provide its pose.

2) The shape of the object is always a cuboid/box shaped. The edges of the box might not be necessarily sharp. It can have rounded edges also. But the object to detected is always box shaped

3) Also my camera would be mounted in an inclined angle(not parallel to the object)

Can i use Opencv for this task. If yes, how do i proceed??

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



Adrian Rosebrock September 13, 2016 at 12:44 pm #

REPLY ↩

Hey Ramanan, this sounds like a pretty neat project, although I don't think it's a great idea to constrain yourself from feature-based approaches. All that said, I personally don't do much work in 3D and stereo vision, so I'm unfortunately not the right person to ask regarding this project. Best of luck!



Eduardo Antonio September 15, 2016 at 7:10 pm #

REPLY ↩

Hi Adrian this post is very helpful

I want to put an "if" to display a "yes", if the object has the size I want

what should I do?

have a nice day ☺



Adrian Rosebrock September 16, 2016 at 8:23 am #

REPLY ↩

After **Line 11** I would do something like:

```
1 if dimA > minSize and dimB > minSize:
2     cv2.putText(orig, "yes", ...)
```

Where the "..." signifies the remaining parameters to the function.



Dj September 16, 2016 at 11:29 am #

REPLY ↩

I def need help on combining several of these projects from pyimagesearch.com. Like skin finder, object detector, measuring, ect. Can you or anyone help me to get a project I'm trying to work on started and working starting ASAP.. Please.... Anyone.



Antonio September 24, 2016 at 11:55 pm #

REPLY ↩

i was wondering if Can i use this code to work together with my webcam?

-Have a nice day!!



Adrian Rosebrock September 27, 2016 at 8:49 am #

REPLY ↩

Sure, absolutely. Simply read a frame from your camera sensor and process the frame as I do in this blog post. [Here is an example of using your webcam](#). You can also find more detailed tutorials inside [Practical Python and OpenCV](#).



Raymart Sison September 28, 2016 at 7:38 am #

REPLY ↩

Can someone please tell me how can i extract those measured value and put it in a txt file?



Adrian Rosebrock September 28, 2016 at 10:33 am #

REPLY ↩

You can use simple file operations to write the measurements to file. I would suggest using [using this tutorial](#) to help you understand the basics of file operations.



Amar October 17, 2016 at 2:05 am #

REPLY ↩

Hey Adrian,

Thank you so much for this great post!

I was having a problem while installing scipy on Rpi 3. I am trying to install it into the virtual e the window "Runing setup.py bdist_wheel for scipy ... /" and hangs there forever.

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

What can I do to install scipy... I would greatly appreciate your help!

Thanks!



Adrian Rosebrock October 17, 2016 at 4:03 pm #

REPLY ↩

It's not "hanging" — it's actually installing. It takes awhile to install SciPy on the Raspberry Pi 3. If I recall correctly, it took approximately 45-60 minutes to install the last time I did it.



Amar October 18, 2016 at 6:47 am #

REPLY ↩

Thanks for the reply, it was eventually installed after waiting for a while!



mahmoud October 27, 2016 at 5:55 am #

REPLY ↩

i want to measure object by millimeters !!



Adrian Rosebrock November 1, 2016 at 9:28 am #

REPLY ↩

That is totally doable with this code. Just change the command line argument to be your known width in millimeters then update the cv2.putText call. The exact same algorithm can still be used.



Larry November 10, 2016 at 7:14 pm #

REPLY ↩

Hi Adrian,

Do you think is possible to get an approximate of the circumference/Shape of an object if we have 2 or 3 pictures from the object?



Adrian Rosebrock November 14, 2016 at 12:18 pm #

REPLY ↩

Yes, once you know the number of pixels per measuring metric you can compute the minimum bounding circle via cv2.minEnclosingCircle. This will give you the radius of the object which you can then use to derive the circumference.



Joey November 27, 2016 at 1:53 am #

REPLY ↩

Hi Adrian! I've been a silent reader of your blogs! Personally, you have helped me a lot especially that I am working on a project now. I just want to know how to do this in real time? Like measuring objects in a video?



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

REPLY ↩

It is totally possible to measure the size of objects in a video stream, just wrap this code in a loop that continuously polls frames from a camera sensor. [This blog post](#) will help you get started. I also provide more examples of working with video streams inside [Practical Python and OpenCV](#).



margarita April 7, 2017 at 11:39 am #

REPLY ↩

Firstly, thank you very much for your guidance! You are amazing! Could you please guide how to make your code work in real time?




Adrian Rosebrock April 8, 2017 at 12:46 pm #

REPLY ↩

I can't provide 1-on-1 guidance or write the code for you, but I do highly

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

how to access video streams. From there you can combine the two tutorials.




Max December 5, 2016 at 1:13 pm #

REPLY ↩

Hello,

Thanks for the nice guide. I have a problem. I Have a camera. I want to take a photo from the camera and mark out an area on it using mouse. Then i want to calculate the area within the marked area. Can you please guide me how do I do it?


Regards



Adrian Rosebrock December 5, 2016 at 1:21 pm #

REPLY ↩

If you want to use mouse events, [please refer to this blog post](#). This post will show you how to use mouse events to capture your click and marking.




Max December 5, 2016 at 1:34 pm #

REPLY ↩

Thanks Adrian for the reply, it was a nice read.

What I need to do is draw a freeform area, not any specific geometric shapes like squares/rectangles/circles, and finally calculate the area within.


Regards
Max



Adrian Rosebrock December 5, 2016 at 1:53 pm #

REPLY ↩

In that case, I would modify the script to maintain a list of (x, y)-coordinates that you click. You can then loop over them and draw them with cv2.line. There is also a polygon drawing function in OpenCV but I can't remember the name of it off the top of my head. You can look it up in the OpenCV docs.




Suraj December 15, 2016 at 5:20 am #

REPLY ↩

after executing the code I'm getting error of image required.. even after mentioning the path and Is there a need to import argparse if yes then plz do tell how to install it.


Regards



Adrian Rosebrock December 18, 2016 at 9:07 am #

REPLY ↩


It sounds like you are having problems with the command line arguments. I would suggest you read [this tutorial](#) before continuing.



Gerdy Hasni December 23, 2016 at 9:31 am #

REPLY ↩


sir, may I ask how to get the angle of things while your program about the measuring of objects? before that, personally I'm very grateful of your code and thanks to you



Adrian Rosebrock December 23, 2016 at 10:47 am #

REPLY ↩

You can use the cv2.minAreaRect function to obtain the angle of a rotated rectangle that encapsulates the object.



GK December 28, 2016 at 4:03 pm #

REPLY ↩

Hi Adrian,

Hope you had a wonderful Christmas!

I have been following your tutorials, and I have learned quite a lot of things from implementir

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

Now I have a question, would it be possible to determine the size of the object with the following conditions:

- 1. Without a reference object
- 2. We know the FOV
- 3. We know the distance – readings from a sensor.

Would you be able to give me some directions?

Thank you,
GK



Adrian Rosebrock December 31, 2016 at 1:28 pm #

REPLY ↩

If you don't have a reference object you would need to calibrate your camera and determine the intrinsic camera parameters first.



GK January 3, 2017 at 1:02 am #

REPLY ↩

Thank you Adrian.
Would you be able to give me some specifics on the parameters?
Here's what I thought I'd do:
I know the distance, real time measurements from a sensor.
I know the F of the camera.
I know the area occupied by the object in the image.
Based on this, following your tutorial on finding the distance between the object and the camera, would it not be possible to find the size of the object?

Note: the object is in the same plane as the entire image. There are no other objects in between or in the background.



Adrian Rosebrock January 4, 2017 at 10:47 am #

REPLY ↩

It's not exactly that simple. I don't have any tutorials on this topic at the moment, but I'll certainly add it to my queue and try to cover it in the future. I can't think of any good existing Python tutorials off the top of my head, but I know [this MATLAB tutorial](#) would be a good start for you to understand the basic principles.



GK January 5, 2017 at 12:53 pm #

Thank you Adrian, for the suggestion. I have one more question.
Would it be possible to measure the size of the object, if I have a reference object with known size and distance, but in separate images?
So, here's what I'm looking at:
I know the size, and the distance to a known object – separate images (following your tutorial, 3 images).
I know the focal length and the actual distance from camera to the target object – from a sensor, I'm able to find the real-time distance.
This looks doable to me, I'm just kind of confused :-/ Would you be able to throw some light on this issue? Thank you ☐



Adrian Rosebrock January 7, 2017 at 9:37 am #

If the images were captured using the same (calibrated) camera sensor, then yes, you could still compute the object size.



Marcos January 9, 2017 at 12:23 pm #

REPLY ↩

Hi!

I have implemented your example and modified it a little to work with live video. Now I'm trying to get measurements on objects' volume using only the webcam. Can you point me in some direction? Thanks!

P.S.: Adrian, your content rocks! Thanks for the effort and congrats on the approaches. Always easy to learn from you.



Adrian Rosebrock January 10, 2017 at 1:10 pm #

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

REPLY ↩



What type of object are you working with? Normally computing the volume would require depth information, in which case I would recommend using a stereo/depth camera instead.



Marcos January 11, 2017 at 11:31 am #

REPLY ↩

Hey, Adrian. Thanks for the attention.

I'm working mainly with boxes and packages. It's for a logistics project. It's a challenge and I'm restrained to webcams, though I can use other commonly available material. Of course I'm expecting not-so-refined measures.

Cheers!



Adrian Rosebrock January 12, 2017 at 8:00 am #

REPLY ↩

Personally, I don't think you're going to get a decent measure of volume without using 2 cameras. If you can setup two webcams you could essentially create a stereo-vision environment by calibrating them and obtaining an approximate depth map.



Ashutosh January 18, 2017 at 12:21 am #

REPLY ↩

I have another thought on the approach. If I crop the image to keep the coin part, and then use the labelling method to find its size in pixels. Finally, using the pixels and the known dimensions I would find the pixels per metric. Would that be a right approach?



Adrian Rosebrock January 18, 2017 at 7:09 am #

REPLY ↩

Yes, if you can manually label the ROI and use the ROI to compute the dimensions this is essentially equivalent to teh calibration step.



ruhi February 1, 2017 at 3:04 am #

REPLY ↩

I have one template image where i have fixed size rectangle which contain some information. Now, i want to search that exact fixed size rectangle from query image then perform cropping. Query image may be in different DPI/resolution format.How can i deal with this issue? Is this article help me? Please suggest some idea for solving it.

Thanks in advance.



Adrian Rosebrock February 1, 2017 at 12:48 pm #

REPLY ↩

If your query image is a different resolution I would suggest always resizing the query to a fixed resolution and then applying the multi-scale template matching technique proposed in [this blog post](#).



Jean February 4, 2017 at 11:20 am #

REPLY ↩

Hi Adrian, I am new to computer vision and as the other commenters said many times I am absolutely grateful with the help you're providing the community.

I noticed your comment regarding the need to aim squarely at the target to obtain a precise measurement. In my case by design my target will be up to 2 degree off. Is there a way to compensate for this?

What I'm trying to do is aim a laser at a ruler placed a few feet out and capture the laser position (e.g. aiming at the 128mm mark). I'm not sure if the better approach is:

1) to write code to detect this laser line and use ocr to interpret where its aiming

or 2) use the technique presented in this post and find a way to offset the 1-2 degree distortion. I know the ruler total length, could I measure distance to the left and right sides of the ruler to derive its angle and then use this info to compensate when measuring the target object length (e.g. from the ruler zero mark to the laser line)?

I'm shooting for 1 or 2 mm accuracy. Thanks in advance for pointing me in the right direction

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



Adrian Rosebrock February 7, 2017 at 9:29 am #

REPLY ↩

The method in this blog post requires only simple camera calibration based on the output image. For your level of accuracy I would suggest instead computing the intrinsic properties of the camera and doing a much more accurate calibration. I don't have a tutorial on this at the moment, but I will try to do one in the future. In the meantime, take a look at [this](#) excellent resource on intrinsic camera properties.



Jackie March 1, 2017 at 5:12 pm #

REPLY ↩

Hi Adrian!

Thank you so much for this post- it's really great!

For some reason, this script doesn't animate for me like yours does. I just see the coin's dimensions and none of the dimensions of the other items. I'm running an Ubuntu VM on Mac.
What would you recommend I do?



Adrian Rosebrock March 2, 2017 at 6:45 am #

REPLY ↩

I'm not sure what you mean by "animate" as I created a GIF file to demonstrate the animation. Click on the active window and press any key to advance the detections. The call to cv2.waitKey pauses execution.



Jackie March 2, 2017 at 1:06 pm #

REPLY ↩

Oh! Thank you so much, Adrian! It all works now.



Ahmed March 19, 2017 at 9:10 pm #

REPLY ↩

Hi Adrian! Thank you so much for this useful post
when i run this program tell me this error

Traceback (most recent call last):
...
from scipy.spatial import distance as dist
ImportError: No module named scipy.spatial

also i installed scipy
i'm waiting for your help



Adrian Rosebrock March 21, 2017 at 7:24 am #

REPLY ↩

It sounds like you might have forgotten to install NumPy and SciPy:

\$ pip install numpy scipy



Rafii Naufal March 29, 2017 at 6:55 pm #

REPLY ↩

hello adrian.. it's work very well in mine...
but if i wanna change the source image to webcam, what steps should i do so it's well work in my raspi 3? i try change the source but it's always failed.. thanks for the knowledges



Adrian Rosebrock March 31, 2017 at 1:56 pm #

REPLY ↩

Please follow this tutorial on [accessing your Raspberry Pi camera](#).



Prateek April 1, 2017 at 6:45 am #

REPLY ↩

Hello Adrian;

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

I want to detect seperate bottles from an image of 5 bottles and process on it that is detect neck and shoulder of bottle and find the centre point of the two and check whether the liquid is filled upto that point. Can you help?



Adrian Rosebrock April 3, 2017 at 2:11 pm #

REPLY ↩

Hey Prateek — I think it would be easier to offer help in this situation if you had an example image of what you’re working with.



E Hamza April 12, 2017 at 9:25 am #

REPLY ↩

hello Adrian, i want to save the measuring sizes and send it to arduino in order to compare it with a data base, is it possible and how ? Can you help?



hamed May 18, 2017 at 1:59 am #

REPLY ↩

My problem is not counting this point is please help me!
Thank you
If these points are not neatly together, and in what way we can resolve this problem.



Ivan May 24, 2017 at 6:25 am #

REPLY ↩

i am try your script but after run. like this ?
usage: object_size.py [-h] -i IMAGE -w WIDTH
object_size.py: error: argument -i/--image is required



Adrian Rosebrock May 25, 2017 at 4:20 am #

REPLY ↩

You need to supply the command line arguments to the script. You can read more about [command line arguments here](#).



shorav May 26, 2017 at 10:24 pm #

REPLY ↩

i am trying to run the code but its throwing the error.. i just want to know that what i have to write in place of -i and --image in order to run the entire code because its throwing an error
Code:
ap.add_argument(“-i”, “–image”,



Adrian Rosebrock May 28, 2017 at 1:03 am #

REPLY ↩

Please read up on [command line arguments](#) before continuing. Knowledge of command line arguments will fix your issue.



Henry June 2, 2017 at 5:16 am #

REPLY ↩

Hi Adrian,

I have a problem py [-h] -i IMAGE -w WIDTH
meas.py: error: the following arguments are required: -i/--image, -w/--width

im using python 3 and opencv3




Adrian Rosebrock June 2, 2017 at 1:58 pm #

REPLY ↩

Please read up on [command line arguments](#). You are forgetting to supply comr


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



Nashwan June 5, 2017 at 5:55 pm <#>

REPLY ↩


Hello Adrian;
Thanks for amazing blog post, thanks in advance for this great totorial..
I want to save the result in one image and result show me measurement for all objects in one image
Plz help me
Regards



Adrian Rosebrock June 6, 2017 at 11:58 am <#>

REPLY ↩

You want to save the resulting image to disk? If so, just use cv2.imwrite.



Nashwan June 6, 2017 at 7:46 pm <#>

REPLY ↩

No, I know this
But I want to draw all the lines for all objects and saved it in a single image..?? How i can do?
Thanks in advance for reply




Jacki June 19, 2017 at 12:11 pm <#>

REPLY ↩

Hi adrean;
I have a question
Why we used dilate and erode algorithm??
What the work of these two algorithms?
If ypu can quide me to these two algorithm

Thanks in advance fo reply
Best regards



Adrian Rosebrock June 20, 2017 at 10:56 am <#>

REPLY ↩

The dilation and erosion helps close any gaps between edges in the edge map.



Jacki June 20, 2017 at 12:09 pm <#>

REPLY ↩

Thank you for reply dear dr. Adrian

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

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

Resource Guide (it's totally free).



Click the button below to get my **free 11-page Image Search Engine Resource Guide PDF**. Uncover **exclusive techniques** that I don't publish on this blog and start building image search engines of your own.

Download for Free!

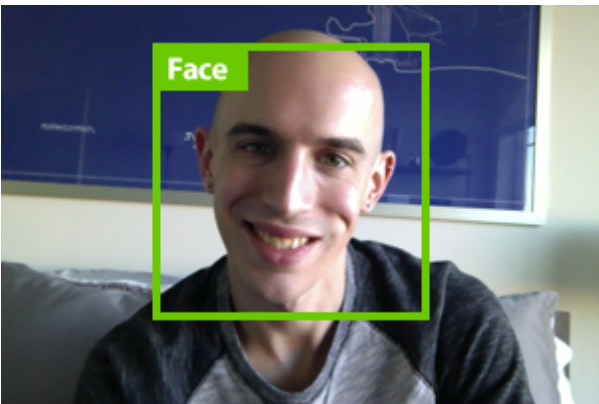
Deep Learning for Computer Vision with Python Book



You're interested in deep learning and computer vision, *but you don't know how to get started*. Let me help. **My new book will teach you all you need to know about deep learning.**

CLICK HERE TO PRE-ORDER MY NEW BOOK

You can detect faces in images & video.



Are you interested in **detecting faces in images & video**? But **tired of Googling for tutorials** that *never work*? Then let me help! I guarantee that my new book will turn you into a **face detection ninja** by the end of this weekend. [Click here to give it a shot yourself.](#)

CLICK HERE TO MASTER FACE DETECTION

PyImageSearch Gurus: NOW ENROLLING!

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


The **PyImageSearch Gurus** course is *now enrolling!* Inside the course you'll learn how to perform:

- Automatic License Plate Recognition (ANPR)
- Deep Learning
- Face Recognition
- *and much more!*

Click the button below to learn more about the course, take a tour, and get 10 (FREE) sample lessons.

TAKE A TOUR & GET 10 (FREE) LESSONS

Hello! I'm Adrian Rosebrock.



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 you in a **single weekend**. I know. It sounds crazy, but it's no joke. My new book is your **guaranteed, quick-start guide** to becoming an OpenCV Ninja. So why not give it a try? [Click here to become a computer vision ninja](#).

CLICK HERE TO BECOME AN OPENCV NINJA

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 OpenCV MAY 25, 2015
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

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

Search

Search...

Find me on **Twitter**, **Facebook**, **Google+**, and **LinkedIn**.
© 2017 PyImageSearch. All Rights Reserved.

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