



Building a Pokedex in Python: Finding the Game Boy Screen (Step 4 of 6)

by **Adrian Rosebrock** on April 21, 2014 in **Building a Pokedex**, **Examples of Image Search Engines**, **Tutorials**

Like 18

G+1 3

2

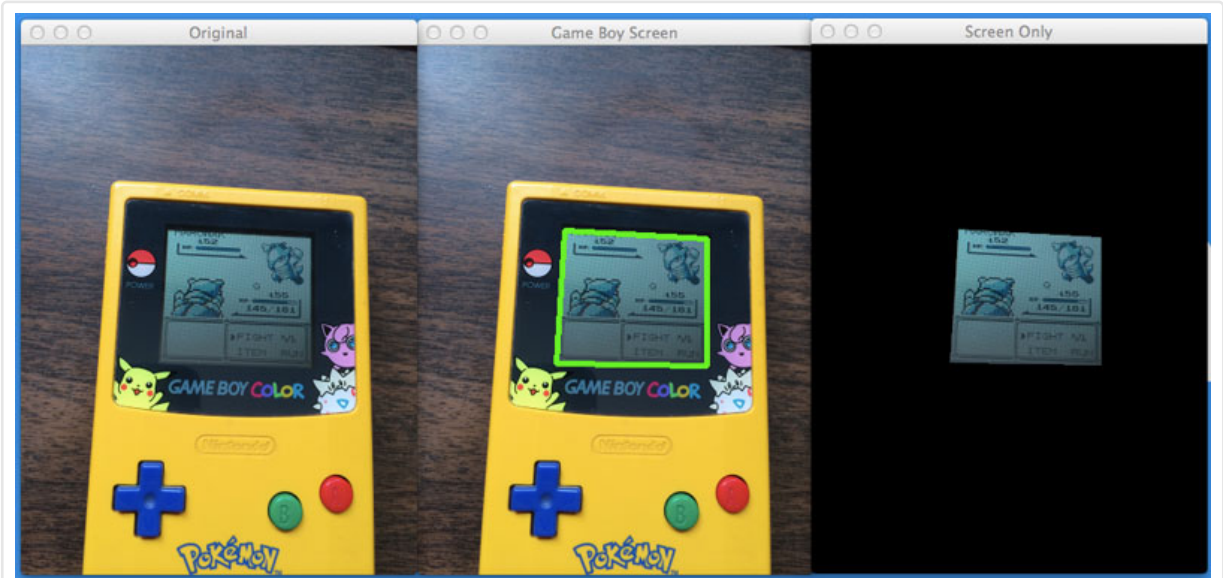


Figure 1: Finding a Game Boy screen in an image using Python and OpenCV.

Quick question.

How does a Pokedex work?

Well, you simply point it a Pokemon, the Pokedex examines its physical characteristics, and the Pokemon is identified instantly.

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

In this case, our smartphone camera is our “Pokedex”. We point our smartphone at our Game Boy, snap a photo of it, and our rival Pokemon is identified (if you don’t believe me, you can see my Pokedex in action by watching [this YouTube clip](#)).

However, there is a lot of information in our image that we don’t need.

We don’t need the shell of the Game Boy case. We don’t need the A, B, up, down, left, right, or start or select buttons. And we certainly don’t care about the background our image was photographed on.

All we care about is the Game Boy screen.

Because once we find that Game Boy screen, we can crop out the Pokemon, and perform the identification.

In this post I’ll show you how to *automatically* find a Game Boy screen in an image using nothing but Python and OpenCV. Specifically, we’ll be using the OpenCV contours functionality and the findContours function in the cv2 package.

Ready?

Here we go.

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

Previous Posts

This post is part of an on-going series of blog posts on how to build a real-life Pokedex using Python, OpenCV, and computer vision and

image processing techniques. If this is the first post in the series that you are reading, definitely take the time to read through it and check it out.

Being able to find a Game Boy screen in an image isn't just cool, it's super practical. I can think of 10-15 different ways to build a small mobile app business using *nothing* but Game Boy screenshots and mobile technology, such as smartphones.

Sound interesting? Don't be shy. [Send me a message and we can chat some more.](#)

Anyway, after you read this post, go back to the previous posts in this series for some added context and information.

- **Step 1:** [Building a Pokedex in Python: Getting Started \(Step 1 of 6\)](#)
- **Step 2:** [Building a Pokedex in Python: Scraping the Pokemon Sprites \(Step 2 of 6\)](#)
- **Step 3:** [Building a Pokedex in Python: Indexing our Sprites using Shape Descriptors \(Step 3 of 6\)](#)

Building a Pokedex in Python: Finding the Game Boy Screen

Before we can *find* the Game Boy screen in an image, we first need an image of a Game Boy:



Figure 2: Our original Game Boy query image. Our goal is to find the screen in this image.

By the way, if you want the raw, original image, be sure to download the source code at the bottom of this post. I've thrown in my FREE 11-page Image Search Engine Resource Guide PDF just to say thanks for downloading the code.

Okay, so now that we have our image, our goal is to find the screen of our Game Boy and highlight it, just as we did in the middle screenshot of Figure 1 at the top of this post.

Fire up your favorite text editor and create a new file named find_screen.py. We're about to get our hands dirty:

Finding the screen of a Game Boy using Python and OpenCV	Python
<pre>1 # import the necessary packages 2 from pyimagesearch import imutils 3 from skimage import exposure 4 import numpy as np 5 import argparse 6 import cv2 7 8 # construct the argument parser and parse the arguments 9 ap = argparse.ArgumentParser() 10 ap.add_argument("-q", "--query", required = True, 11 help = "Path to the query image") 12 args = vars(ap.parse_args())</pre>	

Lines 2-6 just handle importing our packages. We'll make use of skimage, but I won't go over that until the next blog post in the series, so don't worry about that for now. We'll use NumPy like we always do, argparse to parse our command line arguments, and cv2 contains our OpenCV bindings.

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

We only need one command line argument: --query points to the path to where our query image is stored on disk.

If you're wondering about imutils on **Line 2**, refer back to my post on [Basic Image Manipulations in Python and OpenCV](#), where I go over resizing, rotating, and translating. The imutils.py file in the pyimagesearch module simply contains convenience methods to handle basic image processing techniques.

Next up, let's load our query image and start processing the image:

Finding the screen of a Game Boy using Python and OpenCV	Python
<pre>14 # load the query image, compute the ratio of the old height 15 # to the new height, clone it, and resize it 16 image = cv2.imread(args["query"]) 17 ratio = image.shape[0] / 300.0 18 orig = image.copy() 19 image = imutils.resize(image, height = 300) 20 21 # convert the image to grayscale, blur it, and find edges 22 # in the image 23 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) 24 gray = cv2.bilateralFilter(gray, 11, 17, 17) 25 edged = cv2.Canny(gray, 30, 200)</pre>	

On **Line 16** we load our query image off disk. We supplied the path to the query image using the --query command line argument.

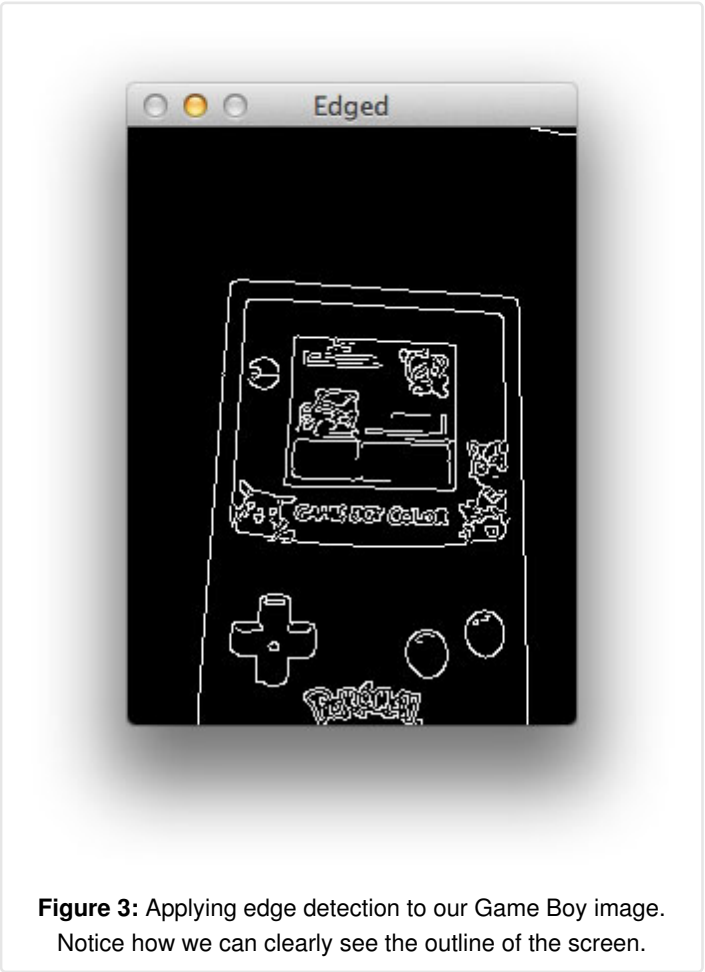
In order to make our processing steps faster, we need to resize the image. The smaller the image is, the faster it is to process. The tradeoff is that if you make your image too small, then you miss out on valuable details in the image.

In this case, we want our new image height to be 300 pixels. On **Line 17** we compute the ratio of the old height to the new height, then we make a clone of the original image on **Line 18**. Finally, **Line 19** handles resizing the image to a height of 300 pixels.

From there, we convert our image to grayscale on **Line 23**. We then blur the image slightly by using the cv2.bilateralFilter function. Bilateral filtering has the nice property of removing noise in the image while still preserving the actual edges. The edges are important since we will need them to find the screen in the Game Boy image.

Finally, we apply Canny edge detection on **Line 25**.

As the name suggests, the Canny edge detector finds edge like regions in our image. Check out the image below to see what I mean:



Clearly we can see that there is a rectangular edge region that corresponds to the screen of our Game Boy. But how do we find it? Let me show you:

Finding the screen of a Game Boy using Python and OpenCV	Python
<pre>27 # find contours in the edged image, keep only the largest 28 # ones, and initialize our screen contour 29 (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE) 30 cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:10] 31 screenCnt = None</pre>	

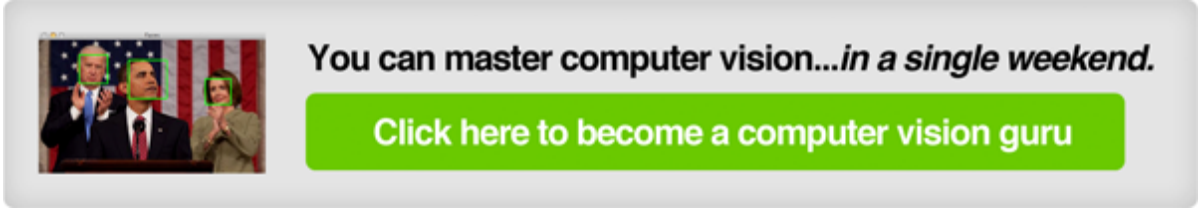
In order to find the Game Boy screen in our edged image, we need to find the *contours* in silhouette of an object — in this case, the outline of the Game Boy screen.

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

To find contours in an image, we need the OpenCV `cv2.findContours` function on **Line 29**. This method requires three parameters. The first is the image we want to find edges in. We pass in our edged image, making sure to clone it first. The `cv2.findContours` method is destructive (meaning it manipulates the image you pass in) so if you plan on using that image again later, be sure to clone it. The second parameter `cv2.RETR_TREE` tells OpenCV to compute the hierarchy (relationship) between contours. We could have also used the `cv2.RETR_LIST` option as well. Finally, we tell OpenCV to compress the contours to save space using `cv2.CV_CHAIN_APPROX_SIMPLE`.

In return, the `cv2.findContours` function gives us a list of contours that it has found.

Now that we have our contours, how are we going to determine which one corresponds to the Game Boy screen?



Well, the first thing we should do is prune down the number of contours we need to process. We know the area of our Game Boy screen is quite large with respect to the rest of the regions in the image. **Line 30** handles sorting our contours, from largest to smallest, by calculating the area of the contour using `cv2.contourArea`. We now have only the 10 largest contours. Finally, we initialize `screenCnt`, the contour that corresponds to our Game Boy screen on **Line 31**.

We are now ready to determine which contour is the Game Boy screen:

Finding the screen of a Game Boy using Python and OpenCV	Python
<pre>33 # loop over our contours 34 for c in cnts: 35 # approximate the contour 36 peri = cv2.arcLength(c, True) 37 approx = cv2.approxPolyDP(c, 0.02 * peri, True) 38 39 # if our approximated contour has four points, then 40 # we can assume that we have found our screen 41 if len(approx) == 4: 42 screenCnt = approx 43 break</pre>	

On **Line 34** we start looping over our 10 largest contours in the query image. Then, we *approximate* the contour using `cv2.arcLength` and `cv2.approxPolyDP`. These methods are used to approximate the polygonal curves of a contour. In order to approximate a contour, you need to supply your level of approximation precision. In this case, we use 2% of the perimeter of the contour. The precision is an important value to consider. If you intend on applying this code to your own projects, you'll likely have to play around with the precision value.

Let's stop and think about the shape of our Game Boy screen.

We know that a Game Boy screen is a rectangle.

And we know that a rectangle has four sides, thus has four vertices.

On **Line 41** we check to see how many points our approximated contour has. If the contour has four points it is (likely) our Game Boy screen. Provided that the contour has four points, we then store our approximated contour on **Line 43**.

The reason why I was able to do this four point check was because I had only a very small number of contours to investigate. I kept only the 10 largest contours and threw the others out. The likelihood of another contour being that large with a square approximation is quite low.

Drawing our screen contours, we can clearly see that we have found the Game Boy screen:

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



Figure 4: We have successfully found our Game Boy screen and highlighted it with a green rectangle.

If you want to draw the contours yourself, just use the following code:

Drawing the Game Boy Screen Contours	Python
<pre>1 cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3) 2 cv2.imshow("Game Boy Screen", image) 3 cv2.waitKey(0)</pre>	

So there you have it, Part 1 of finding the Game Boy screen.

In Step 2 of this post, I'll show you how to perform a perspective transform on the Game Boy screen as if you were "looking down" at your Game Boy from above. Then, we'll crop out the actual Pokemon. Take a look at the screenshot below to see what I mean:

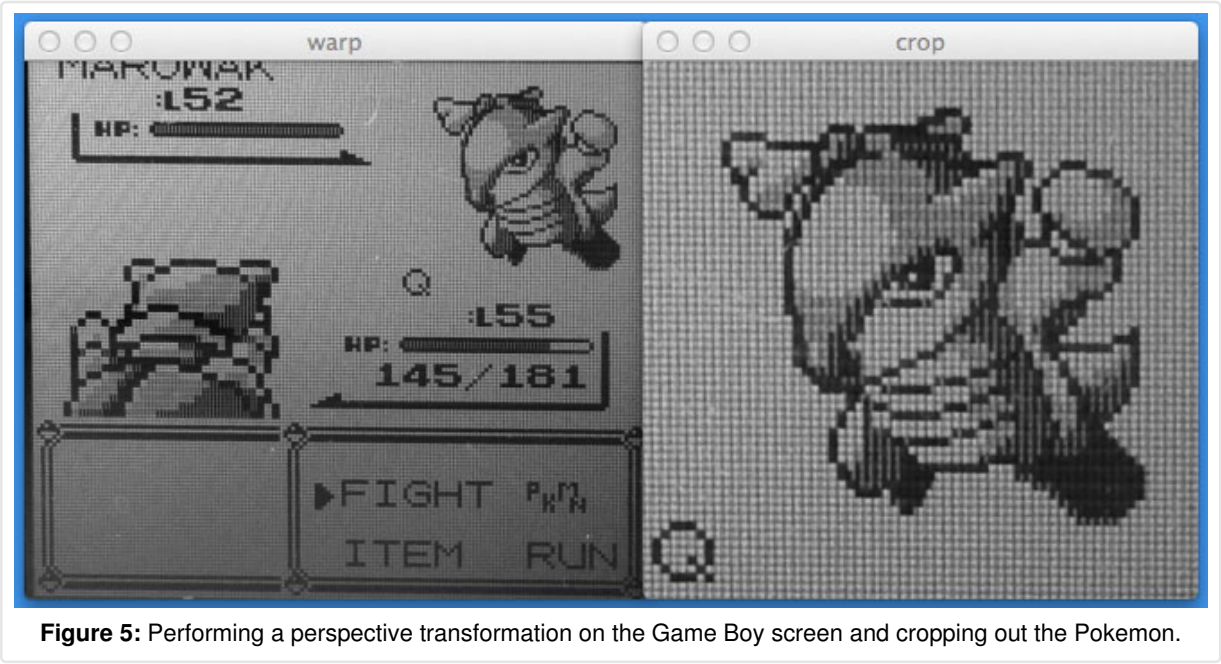


Figure 5: Performing a perspective transformation on the Game Boy screen and cropping out the Pokemon.

Summary

In this post I showed you how to find a Game Boy screen in an image using Python, OpenCV, and computer vision and image processing techniques.

We performed edge detection on our image, found the largest contours in the image using OpenCV and the cv2.findContours function, and approximated them to find their rudimentary shape. The largest contour with four points corresponds to our Game Boy screen.

Being able to find a Game Boy screen in an image isn't just cool, it's super practical. I can think of 10-15 different ways to build a small business using *nothing* but Game Boy screenshots and mobile technology, such as smartphones.

Sound interesting? Don't be shy. [Send me a message and we can chat some more.](#)

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

In the next post, I'll show you how to apply a perspective transformation to our Game Boy screen so that we have a birds-eye-view of the image. From there, we can easily crop out the Pokemon.

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:

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!

DOWNLOAD THE GUIDE!

🔖 [approximate contours](#), [bilateral blurring](#), [blurring](#), [canny](#), [contours](#), [edge detection](#), [find contours](#), [pokedex](#), [pokemon](#)

< Building a Pokedex in Python: Indexing our Sprites using Shape Descriptors (Step 3 of 6)

Building a Pokedex in Python: OpenCV and Perspective Warping (Step 5 of 6) >

72 Responses to *Building a Pokedex in Python: Finding the Game Boy Screen (Step 4 of 6)*



zhahir May 1, 2014 at 5:09 am #

REPLY ↩

Great guidance...thanks.



Meng Xipeng June 28, 2014 at 8:29 pm #

REPLY ↩

You website is awesome, I'm working on commuter vision field. focus on OCR.
I find OpenCV+Python is a quick way to prototype computer vision field.
You shared many great and fun stuff in your website, Thank you for your work.



Adrian Rosebrock June 29, 2014 at 6:40 am #

REPLY ↩

I'm glad you are finding the content useful! ☐



Erkki Nyfors July 1, 2014 at 9:59 pm #

REPLY ↩

how do you suggest to find a license plate from a car picture? ☐

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



Adrian Rosebrock July 2, 2014 at 5:38 am #

REPLY ↩

Sure, we can chat about that. Send me a message and we can talk.



Sasa November 2, 2014 at 4:21 am #

REPLY ↩

I am also intrested in this, maybe you could make some blog post about it.



Adrian Rosebrock November 4, 2014 at 6:54 am #

REPLY ↩

Maybe something along the lines of [this](#)? The code is still rough around the edges, I need to clean it up first.



Ashwin November 10, 2014 at 8:15 pm #

REPLY ↩

I followed your code in the end when you draw the contours on the source image I am not getting the green highlighted region



Adrian Rosebrock November 11, 2014 at 6:59 am #

REPLY ↩

Hi Ashwin, the code below can be used to draw the green highlighted region:

```
cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 3)
cv2.imshow("Game Boy Screen", image)
cv2.waitKey(0)
```



Praveen February 5, 2015 at 12:33 am #

REPLY ↩

Sir, You have used the following module in one of your programs " from skimage import exposure". I am unable to find this skimage module anywhere. Please send me this skimage module



Adrian Rosebrock February 5, 2015 at 6:38 am #

REPLY ↩

You need to install skimage from scikit-image.org first.



Shelly February 12, 2015 at 3:13 pm #

REPLY ↩

Hey, just a heads up that this page seems to be a little broken layout-wise – all the others on your site are working for me. I also couldn't find it using tags... Just wanted to let you know!



Shelly February 12, 2015 at 3:26 pm #

REPLY ↩

Of course, it seems to look fine on the comment-submitted page... And the only tag it's missing is 'pokedex', I think. ☺



Adrian Rosebrock February 12, 2015 at 7:47 pm #

REPLY ↩

Hi Shelly. I've updated the post to include the 'pokedex' tag. Can you send me a screenshot of the formatting that's messed up? It looks normal on my web browser (Chrome on OSX).



Patrick April 14, 2015 at 4:29 pm #

REPLY ↩

Thanks for the guide! This is an awesome post and blog. Very helpful.

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



Adrian Rosebrock April 14, 2015 at 5:27 pm #

REPLY ↩

I'm glad you are enjoying it Patrick! ☐



sai pattnaik May 30, 2016 at 9:22 am #

REPLY ↩

Hello Adrian its a very very helpful post but i have one problem, i need to get the cropped cheques from scanned jpeg in which the background color is white (A4) and the edges of the cheques are also white so its not able to pick the whole cheque rather its picking a portion from the cheque,i think its unable to identify because of the white border and background.. can you help me on this..



Adrian Rosebrock May 31, 2016 at 3:53 pm #

REPLY ↩

Keep in mind that it's easier to write computer vision code for *good* environment conditions that *bad* ones. Simply put — don't use a white background ☺ Select a background that contrasts your cheque color and this will make the project much easier to solve. If you don't, you'll simply make life harder for yourself and may have to resort to utilizing a bit of machine learning to find the cheque borders. This is especially wasteful if you can change your background.



Tham June 9, 2015 at 10:20 pm #

REPLY ↩

Thanks for your interesting post

I have two qusetions about this

1 : What if the four vertices contours is not the rectangle we need?
I noticed shape of the gameboy is rectangle too, if the image contain all of the gameboy but not part of it, the largest rectangle will be the gameboy itself

2 : How could you detect the rotation need to be adjust?
The original image is not 90 degree, I need to rotate the original picture “query_marowak.jpg” to crop the image of pokemon, how could you let the computer knowthe image need to do some rotation?



Adrian Rosebrock June 10, 2015 at 7:08 am #

REPLY ↩

Hey Tham, great questions, thanks for asking.

1. If the four vertices do not correspond to the Game Boy, then indeed, the script will not work. Here we make the assumption that the largest 4-corner region is the Game Boy screen.
2. You can determine the rotation of the bounding box by using the cv2.minAreaRect function. But a better method is to simply apply a perspective transform and obtain the view of the screen, [like we do in this post](#).



Mohammed Ali November 12, 2015 at 1:07 pm #

REPLY ↩

hello my friend
thanks for you
but i have a problem that when i am running the example
on the same arguments on the same image it gives me this error

```
1 (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
2 ValueError: too many values to unpack
```

can you help me please?



Adrian Rosebrock November 13, 2015 at 5:43 am #

REPLY ↩

Hey Mohammed — I've referenced this problem many times on the PyImageSearch blog. Please note that this blog post is intended for *OpenCV 2.4* (which is stated at the top of this post). You are running *OpenCV 3*. The cv2.findContours between the two versions. You can read more about this change [here](#).

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

versions, change the line to:

cnts = cv2.findContours(edged.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)[-2]



Say Hong January 28, 2016 at 4:43 am #

REPLY ↩

the return values are now 3 instead of 2 (as shown on the opencv docs pages).

the return values are image, contours, hierarchy

```
1 image, contours, hierarchy = cv2.findContours(imgsrc, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```



Adrian Rosebrock January 28, 2016 at 3:40 pm #

REPLY ↩

Just to follow up on this comment, for anyone interested in reading how the cv2.findContours method has changed between OpenCV 2.4 (which this tutorial uses) and OpenCV 3, [be sure to read this post](#).



Sachin November 13, 2015 at 10:26 pm #

REPLY ↩

i want to find rectangular shapes in image but if rectangle is incomplete then how do i know which side is missing (i.e top or bottom or right or left?)



Adrian Rosebrock November 14, 2015 at 6:19 am #

REPLY ↩

Hey Sachin — contour approximation is can help with finding shapes, but if you do not have a complete side, then your shape is not a rectangle. In that case, you might want to look into morphological operations such as dilation and closing to help close the gaps between the sides.



Lakshmi January 20, 2016 at 6:50 am #

REPLY ↩

if its not a rectangle, if its a ellipse how can i find the pionts



Adrian Rosebrock January 20, 2016 at 1:43 pm #

REPLY ↩

I demonstrate how to perform circle detection [this post](#). For ellipse detection, take a look at [scikit-image](#).



Lakshmi January 22, 2016 at 12:33 am #

REPLY ↩

thanks a lot.. its very useful to me.. this blog is helpful to learners.. thanks ☐



Manoj February 11, 2016 at 11:53 pm #

REPLY ↩

Hey,i actually want to use like specific points on a part of the image.How can i use contours?



Adrian Rosebrock February 12, 2016 at 3:18 pm #

REPLY ↩

I’m not sure what you mean by “use like specific points on part of an image”. Can you elaborate?



Jaime Lopez February 23, 2016 at 4:04 pm #

REPLY ↩

Hi,

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

I think you have to update your function call cv2.findContours, because it now returns a tuple of 3, this way:

(cnts, _, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)



Adrian Rosebrock February 24, 2016 at 4:37 pm #

REPLY ↩

Hey Jaime, your code is actually incorrect. It should be:

(_, cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

You can read more about how the cv2.findContours function changed in between OpenCV 2.4 and OpenCV 3 [in this blog post](#).



Jaime Lopez April 25, 2016 at 2:45 pm #

REPLY ↩

Hi Adrian,

Do you know how can I find the maximum square (not rotated rectangular) inside a specific contour, giving as parameter the square's centroid? I need all square area fall inside the contour. Any advice?

Thanks in advance, Jaime



Adrian Rosebrock April 26, 2016 at 5:18 pm #

REPLY ↩

You can compute a normal bounding rectangle (not rotated) by using cv2.boundingRect(c) where c is the current contour you are examining. Converting the bounding rectangle to a square can be accomplished by finding the side with the largest length and adjusting all other sides to be equal to that length.



Cat June 22, 2016 at 2:29 pm #

REPLY ↩

Hi Adrian!!!

This isperfect for my project. Do you know how to extract the coordinates of multiple bar codes?

I need to extract 3 bar codes per image. Is there a way to change the command that sorts the contours to do this for me? Thank you!



Adrian Rosebrock June 23, 2016 at 1:16 pm #

REPLY ↩

Hm, I think you might be posting this comment on the wrong blog post? This post doesn't deal with barcodes. In any case, if you want to filter a set of contours (rather than simply grabbing the largest one), simply loop over them and ensure they meet a minimum width, height, or area requirement.



sajad July 28, 2016 at 2:28 pm #

REPLY ↩

thanks



Megha November 15, 2016 at 2:18 am #

REPLY ↩

I have an image that I am fiddling with to detect the outer contours . How can we chat ? I tried using contour sorting to get the largest areas and than use that to get only the outer edges but since my image is complex I am having issues.



Megha November 15, 2016 at 2:20 am #

REPLY ↩

I am also a bit confused with your example because even the outer most edge is a rectangle as well. Why wasnt that detected?

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



Adrian Rosebrock November 16, 2016 at 1:50 pm #

REPLY ↩

The outermost edge of what? The GameBoy? That’s actually not a rectangle — there is no bottom line to the edge.



Akkumaru December 7, 2016 at 2:15 pm #

REPLY ↩

Hi Adrian,

Thank you for making this blog!

I have a peculiar question: is it possible to perform contour detection on one image only to “apply” the detected contours on a different image, given that the dimensions of the two images are identical.

This is because I have an image that contains collages of photographs, and I aim to isolate those photos. Since it is hard to do contour detection on the original image itself, I’m thinking of doing the detection on another image (which is much more simplified), then ‘paste’ the resulting contours on the original to get the original photographs.

Do you think this is possible? Or would you suggest any other approach?

Thank you!



Adrian Rosebrock December 10, 2016 at 7:40 am #

REPLY ↩

You mentioned that the dimensions of the two images are identical — does that mean they line up perfectly? If so, sure. You can absolutely detect contours in one image and then extract ROIs from another image. But you should make sure that the dimensions are indeed correct first.

However, I get the impression that these images may not line up perfectly. In that case, you should apply keypoint detection, local invariant descriptors, and keypoint matching. Inside [Practical Python and OpenCV](#) I demonstrate how to build a computer vision system that can match and recognize book covers in two separate images. The same technique could be used for your photograph collage project as well.



Daniel Senter January 19, 2017 at 6:56 pm #

REPLY ↩

Hello Adrian!

A short time I have been accompanying his work, however, very interesting and motivating. From this, I chose my course completion topic on image recognition in python ... I can already find the image and binarize it, but I need to find the center point of the object that exists in the image, which is a piece that first ls on a treadmill, a webcam takes a photo, identifying whether it is a circle or square, and whether it is red or green, then has an anthropomorphic robot to allocate a part in a predefined place for each feature of the piece.

I would like your help, with tutorial indications that may make my work easier, or even a YouTube video with a similar application. Congratulations again for your work, I look forward to an answer.



Adrian Rosebrock January 20, 2017 at 10:59 am #

REPLY ↩

To find the centroid coordinates of an object in an image, compute its centroid. [This blog post](#) demonstrates computing a centroid. You should also use [this post](#) for help recognizing color and shape.



Lugia February 12, 2017 at 1:59 am #

REPLY ↩

Hi Adrian,

Thanks for sharing this blog. Do you have step5 and step6 of this Pokemon series?



Adrian Rosebrock February 13, 2017 at 1:44 pm #

REPLY ↩

Yes, you can find [step 5 here](#) and [step 6 here](#).



bams March 5, 2017 at 1:21 pm #

REPLY ↩

hello Adrian, could you help me? I have a project for detect license plate number. T

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

background and white text. And I can't detect the plate position using cv2.darwContours. I use mathematical Morphology as a method. And then the second problem is, my program can detect license plate number if the plate white background and black text



Adrian Rosebrock March 6, 2017 at 3:43 pm #

REPLY ↩

Hey Bams — for questions related to ANPR, I would suggest joining the [PyImageSearch Gurus course](#) where I discuss ANPR in detail. I can also provide suggestions on ANPR algorithms there as well.



EMMA March 9, 2017 at 6:16 am #

REPLY ↩

hi Adrian i want to do image segmentation in order to detect human presence after it ,can u help me please



Adrian Rosebrock March 10, 2017 at 3:50 pm #

REPLY ↩

Take a look at my blog post on [person detection](#). That will help you get started.



Rezig April 4, 2017 at 5:40 pm #

REPLY ↩

Hi Adrian,

Thanks for all your amazing tutorials, they are well explained.

I would like to know if it is possible to get the position of a specific rectangle based on its size (for example i want to know the position of the rectangle that has x and y (width and height))

i'll appreciate your help



Adrian Rosebrock April 5, 2017 at 11:53 am #

REPLY ↩

Hey Rezig, thanks for the comment. Do you know the size of the rectangle ahead of time? If so, just loop over all contours that have 4 vertices and then check if the width and height are within your requirements.



Rezig April 25, 2017 at 11:06 am #

REPLY ↩

thanks for your response, actually a figured out a way to do that by using the cv2.moments() or cv2.contourArea(ctr) to calculate the area of all contours and then looping over them to find the concerned rectangle.

But it seams that is not very accurate because the rectangle that i am looking for has width=10 and heigth=100 so normally the area should be 1000 but the cv2.moments and cv2.contourArea(ctr) are giving an area=679 !

i wonder if you have any idea about ?

would your method with contours that have for vertices be more accurate ?

thanks again



Adrian Rosebrock April 25, 2017 at 11:56 am #

REPLY ↩

It's hard to say what the exact error is without looking at your input image. But based on what you're described, I think your rectangle might be rotated? And if so, the bounding box area would be different than the area itself.

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



Rezig April 26, 2017 at 4:42 am #

Hi again,

here is the image i am talking about <http://imgur.com/a/DG0li> (the backgroud is also a part of the image) and i am trying to get the positions of the paddles and the circle in the pong game, for that i am using cv2.houghCircles to detect the circle and cv2.contourArea to detect the paddles (as i said the paddle width and heigth are 10 and 100).

is it the right way to do that ?



Adrian Rosebrock April 28, 2017 at 9:16 am #

It looks like your paddles are “touching” the borders of the image which might be causing the issue. Given that this is a screenshot of a game, I think it would be *much easier* to simply apply [color thresholding](#) to detect the paddles and ball.



Rezig April 28, 2017 at 12:00 pm #

thanks for answer,

yes at first i tried your tutorial of color thresholding and it worked fine. But this solution depends on the computer’s backgroud colors (if it contains the same colors of the game that would make problems). so thats why i prefer to detect the paddles and the pong without refering to their colors !



Adrian Rosebrock May 1, 2017 at 1:52 pm #

My suggestion would be to continue working with thresholding then. After thresholding pad the image via cv2.copyMakeBorder and see if that helps you detect the paddles along the border of the image.



Andy April 28, 2017 at 9:07 am #

REPLY ↩

Hi Adrian,

It is always a nice experience reading your blog. I have a simple problem I cannot get my head around and it might be something you will come around in later blogs.

The issue is quite simple. I have identified shape(s) in a image and drawn the contour(s) using cv2.findContours and identified the coordinates of the relevant x,y points for maintaining the shape. My problems starts when I want to extract these coordinates.

Instead of below. Where one shape will be on multiple lines I would like something I can easily write to a file including metadata and import either again in python or store in a database.

```
[array([[[ 19, 0]],
[[ 19, 82]],
...
[[ 62, 295]],
[[ 62, 0]]], dtype=int32)]
```



Adrian Rosebrock April 28, 2017 at 9:11 am #

REPLY ↩

Hi Andy — so if I understand your question correctly, you would like to save specific contours to disk? If so, you can use pickle:

```
f = open("output.pickle", "wb")
f = write(pickle.dumps(c))
f.close()
```

Where c is a specific contour.



Andy April 28, 2017 at 9:41 am #

REPLY ↩

Thanks Adrian – i knew it was something simple. Have a superb weekend ☺

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



Pranjul Singh June 12, 2017 at 5:07 am #

REPLY ↩

Hey Adrain, I am currently working on a project where I have to measure the size of various body parts like waist, hip, neck etc. from a frontal and lateral image of the person using openCV(I am currently working in python). Since there is an app Mtailor which evaluates the entire thing without any reference object and without taking height as input. But I am looking for taking height as input to make my job easier so that I can use it to calculate pixelpermetric and further use it for calculating size of other body parts. Now the thing is I have to draw contours for which I have thought of apporxPolyDP and then segment each body part approximately by using kollman's distribution of the human body for which I have thought of using Watershed algorithm. You can go through this paper (<https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2012/tr-2012-11.pdf>) to get more details on circumference computation and kollman's distribution. Please guide me whether I am on the right track. I will really be grateful for your precious help. Thank you so much.



Adrian Rosebrock June 13, 2017 at 11:03 am #

REPLY ↩

Hi Pranjul — thanks for the comment; however, I must admit that I don't have any experience working with extracting and measuring the size of body parts. My particular area of expertise falls in image search engines, image classification, and deep learning. This specific question is outside my area of expertise, I'm sorry I couldn't be more help here!



Mark June 14, 2017 at 9:48 am #

REPLY ↩

Hello Adrian, I am attempting to find two rectangles in a video loop, and have found success making this code into a video loop with opencv and python, but I am having trouble figuring out how to have 2 instances, and make the code lock on to the target live, without going off to my shirt or whatever else it seems to find. The point of it is to find the two rectangles, and be able to find the center in between them, which is simple for me to do as i just have to subtract coordinates, my main problem is having two instances of the contour detection. Thanks for the post!



Mark June 14, 2017 at 9:53 am #

REPLY ↩

Also, I'm not too sure if this would be possible, but for the contour to stay on the one specific area? it seems to jump around quite often



Adrian Rosebrock June 16, 2017 at 11:28 am #

REPLY ↩

Hi Mark — do you have any example images of these rectangles you are trying to detect? That would likely help me understand your question better.

Trackbacks/Pingbacks

- Python and OpenCV Example: Warp Perspective and Transform - May 5, 2014
- [...] In my previous blog post, I showed you how to find a Game Boy screen in an image using Python and OpenCV. [...]
- Comparing Shape Descriptors for Similarity using Python and OpenCV - May 30, 2014
- [...] sprites using Zernike moments. We've analyzed query images and found our Game Boy screen using edge detection and contour finding techniques. And we've performed perspective warping and transformations using the cv2.warpPerspective [...]
- Detecting Circles in Images using OpenCV and Hough Circles - PyImageSearch - July 21, 2014
- [...] your blog. I saw your post on detecting rectangles/squares in images, but I was wondering, how do you detect circles in images using [...]
- Target acquired: Finding targets in drone and quadcopter video streams using Python and OpenCV - PyImageSearch - June 7, 2015
- [...] image. We've used in in building a kick-ass mobile document scanner. We've used it to find the Game Boy screen in an image. And we've even used it on a higher level to actually filter shapes from an [...]
- Zero-parameter, automatic Canny edge detection with Python and OpenCV - PyImageSearch - June 14, 2015
- [...] times. We've used it to build a kick-ass mobile document scanner and we've used to find a Game Boy screen in a photo, just two name a couple [...]

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

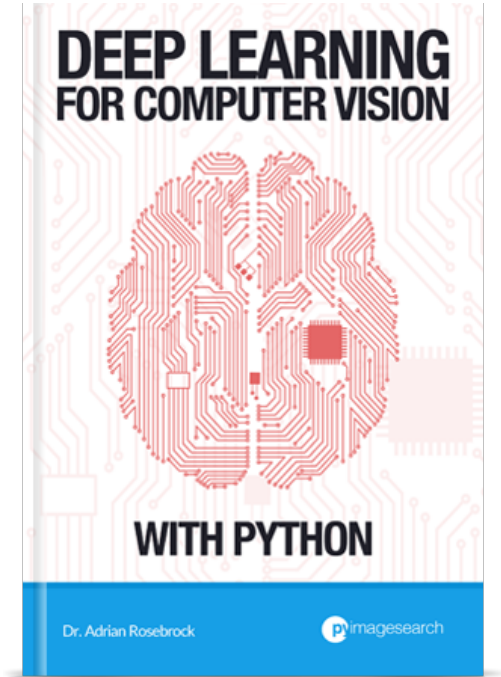
Resource Guide (it's totally free).



Click the button below to get my **free 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!](#)

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

CLICK HERE TO MASTER FACE DETECTION

PylmageSearch Gurus: NOW ENROLLING!


The PylmageSearch 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.

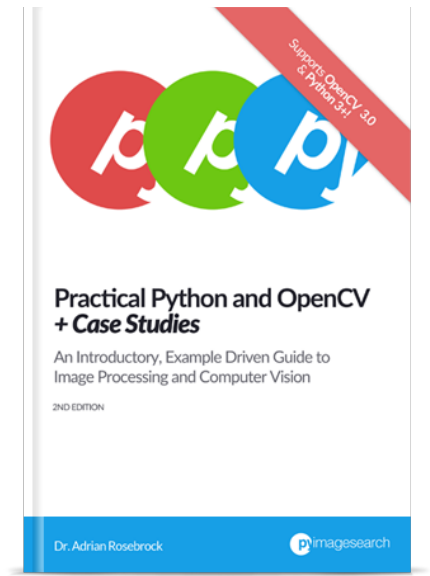
TAKE A TOUR & GET 10 (FREE) LESSONS

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 PylmageSearch RSS Feed and keep up to date with my image search engine tutorials, tips, and tricks

POPULAR

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

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

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