pyimagesearch
be awesome at building image search engines

How to Build a Kick-Ass Mobile Document Scanner in Just 5 Minutes

by **Adrian Rosebrock** on September 1, 2014 in **Image Processing**, **Tutorials**

Remember my friend James from my post on skin detection?

Well, we grabbed a cup of coffee and some groceries yesterday at the local Whole Foods, all while discussing the latest and greatest CVPR papers.

After I checked-out, I whipped out my iPhone and took a picture of the receipt — it's just something that I like to do so I can easily keep track of where my money is going each month (like spending $20 on bacon, for instance).

Anyway, James asked why I wasn't using a document scanning app like Genius Scan or TurboScan to scan the receipt into my phone.

Good question.

I honestly hadn't thought of it.

But then James went too far…

He bet me $100 that I couldn't build a mobile document scanner app of my own.

Well, the joke is on you, James.

I'll be collecting my $100 the next time I see you.

Because honestly, document scanning apps are ***ridiculously easy*** to build.

Genius Scan. TurboScan. Scanner Pro. You name it — the computer vision algorithms running behind the scenes are dead simple to implement.

I bet that after reading this blog post that you could create an app to compete with the big-boy mobile document scanners as well.

You see, scanning a document using your smartphone can be broken down into three simple steps:

- **Step 1:** Detect edges.
- **Step 2:** Use the edges in the image to find the contour (outline) representing the piece of paper being scanned.
- **Step 3:** Apply a perspective transform to obtain the top-down view of the document.

Really. That's it. Only three steps and you're on your way to submitting your own document scanning app to the App Store.

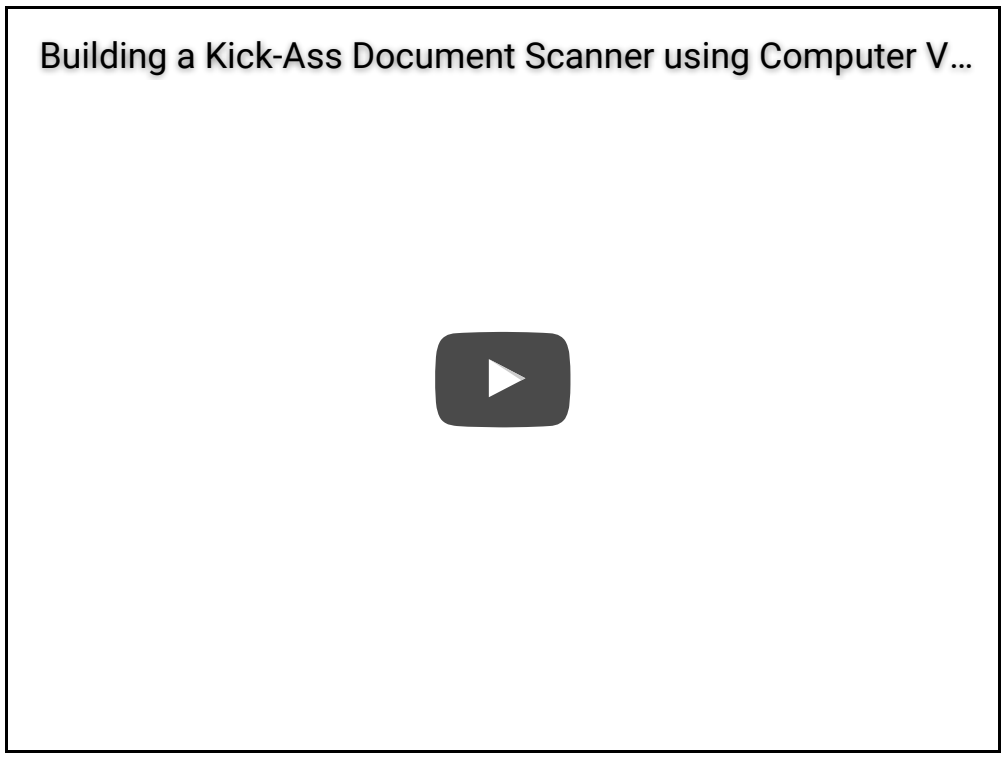
Sound interesting?

Read on. And unlock the secrets to build a mobile scanner app of your own.

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

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

How To Build a Kick-Ass Mobile Document Scanner in Just 5 Minutes



Last week I gave you a special treat — my very own `transform.py` module that I use in all my computer vision and image processing projects. You can read more about this module here.

Whenever you need to perform a 4 point perspective transform, you should be using this module.

And you guessed it, we'll be using it to build our very own document scanner.

So let's get down to business.

Open up your favorite Python IDE, (I like Sublime Text 2), create a new file, name it `scan.py` , and let's get started.

Building a Document Scanner App using Python, OpenCV, and Computer Vision — Python

```
1  # import the necessary packages
2  from pyimagesearch.transform import four_point_transform
3  from pyimagesearch import imutils
4  from skimage.filters import threshold_adaptive
5  import numpy as np
6  import argparse
7  import cv2
8
9  # construct the argument parser and parse the arguments
10 ap = argparse.ArgumentParser()
11 ap.add_argument("-i", "--image", required = True,
12     help = "Path to the image to be scanned")
13 args = vars(ap.parse_args())
```

Lines 2-7 handle importing the necessary Python packages that we'll need.

We'll start by importing our `four_point_transform` function which I discussed last week.

We'll also be using the `imutils` module, which contains convenience functions for resizing, rotating, and cropping images. You can read more about `imutils` in my basic image manipulations post.

Next up, let's import the `threshold_adaptive` function from scikit-image. This function wi

scanned image.

Note: *If you're using the older version* **0.11.x** *of* `scikit-image` *and getting an import error related to the* `filter` *sub-module, you'll want to change the* `threshold_adaptive` *import to:* `from skimage.filter import threshold_adaptive` *(notice how the "s" has been removed from "filters"). In version* **0.12.x** *(and all versions moving forward) of* `scikit-image` *, the* `filter` *sub-module has been re-named to* `filters` *. Consider upgrading your install of* `scikit-image` *if you haven't already.*

Lastly, we'll use NumPy for numerical processing, `argparse` for parsing command line arguments, and `cv2` for our OpenCV bindings.

Lines 10-13 handle parsing our command line arguments. We'll need only a single switch image, `--image` , which is the path to the image that contains the document we want to scan.

Now that we have the path to our image, we can move on to Step 1: Edge Detection.

Step 1: Edge Detection

The first step to building our document scanner app using OpenCV is to perform edge detection. Let's take a look:

Building a Document Scanner App using Python, OpenCV, and Computer Vision — Python

```
15 # load the image and compute the ratio of the old height
16 # to the new height, clone it, and resize it
17 image = cv2.imread(args["image"])
18 ratio = image.shape[0] / 500.0
19 orig = image.copy()
20 image = imutils.resize(image, height = 500)
21
22 # convert the image to grayscale, blur it, and find edges
23 # in the image
24 gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
25 gray = cv2.GaussianBlur(gray, (5, 5), 0)
26 edged = cv2.Canny(gray, 75, 200)
27
28 # show the original image and the edge detected image
29 print "STEP 1: Edge Detection"
30 cv2.imshow("Image", image)
31 cv2.imshow("Edged", edged)
32 cv2.waitKey(0)
33 cv2.destroyAllWindows()
```

First, we load our image off disk on **Line 17**.

In order to speedup image processing, as well as make our edge detection step more accurate, we resize our scanned image to have a height of 500 pixels on **Lines 17-20**.

We also take special care to keep track of the `ratio` of the original height of the image to the new height (**Line 18**) — this will allow us to perform the scan on the *original* image rather than the *resized* image.

From there, we convert the image from RGB to grayscale on **Line 24**, perform Gaussian blurring to remove high frequency noise (aiding in contour detection in Step 2), and perform Canny edge detection on **Line 26**.

The output of Step 1 is then shown on **Lines 29-33**.

Take a look below at the example document:

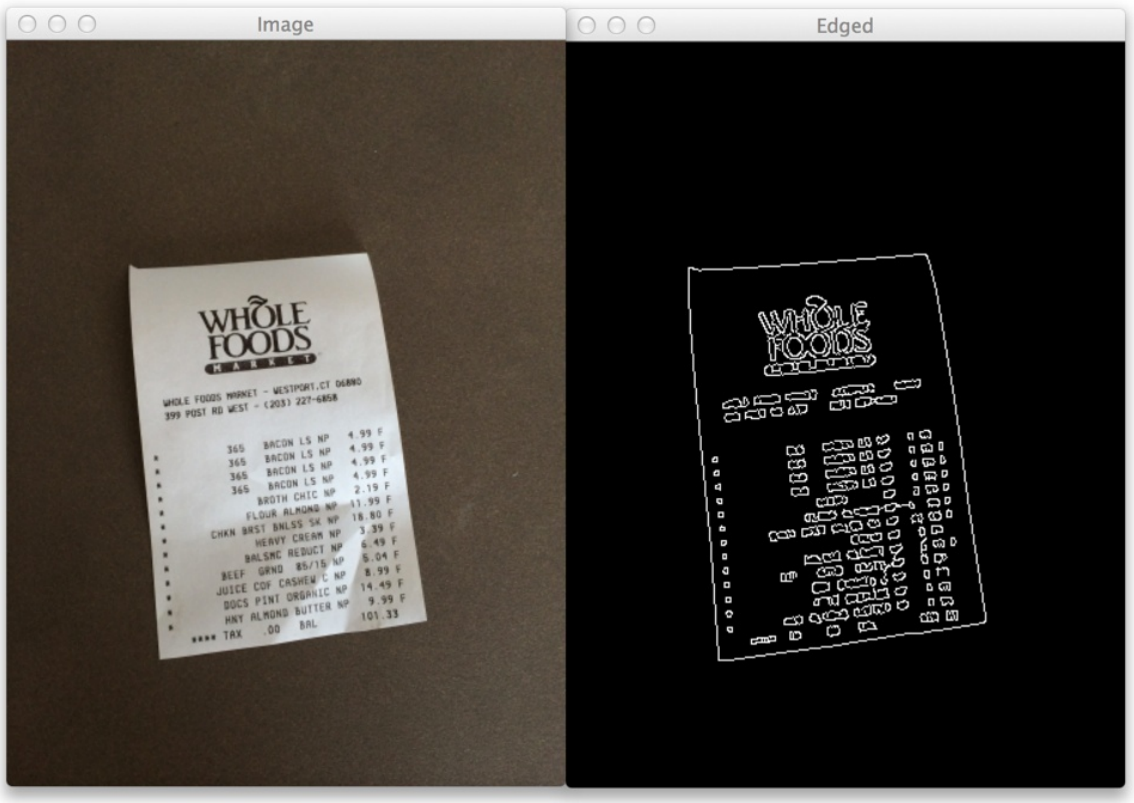


Figure 1: The first step of building a document scanning app. On the *left* we have the original image and on the *right* we have the edges detected in the image.

On the left you can see my receipt from Whole Foods. Notice how the picture is captured at an angle. It is definitely not a 90-degree, top-down view of the page. Furthermore, there is also my desk in the image. Certainly this is not a “scan” of any means. We have our work cut out for us.

However, on the right you can see the image after performing edge detection. We can clearly see the outline of the receipt.

Not a bad start.

Let’s move on to Step 2.

Step 2: Finding Contours

Contour detection doesn’t have to be hard.

In fact, when building a document scanner, you actually have a *serious advantage…*

Take a second to consider what we’re actually building.

A document scanner simply scans in a piece of paper.

A piece of paper is assumed to be a rectangle.

And a rectangle has four edges.

Therefore, we can create a simple heuristic to help us build our document scanner.

The heuristic goes something like this: we’ll assume that the *largest contour* in the image *with exactly four points* is our piece of paper to be scanned.

This is also a reasonably safe assumption — the scanner app simply assumes that the document you want to scan is the main focus of our image. And it’s also safe to assume (or at least should be) that the piece of paper has four edges.

And that’s exactly what the code below does:

Building a Document Scanner App using Python, OpenCV, and Computer Vision — Python

```
35 # find the contours in the edged image, keeping only the
36 # largest ones, and initialize the screen contour
37 (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
38 cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]
39
40 # loop over the contours
41 for c in cnts:
42     # approximate the contour
43     peri = cv2.arcLength(c, True)
44     approx = cv2.approxPolyDP(c, 0.02 * peri, True)
45
46     # if our approximated contour has four points, then we
47     # can assume that we have found our screen
48     if len(approx) == 4:
49         screenCnt = approx
```

```
50          break
51
52  # show the contour (outline) of the piece of paper
53  print "STEP 2: Find contours of paper"
54  cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 2)
55  cv2.imshow("Outline", image)
56  cv2.waitKey(0)
57  cv2.destroyAllWindows()
```

We start off by finding the contours in our edged image on **Line 37**.

A neat performance hack that I like to do is actually sort the contours by area and keep only the largest ones (**Line 38**). This allows us to only examine the largest of the contours, discarding the rest.

We then start looping over the contours on **Line 41** and approximate the number of points on **Line 43 and 44**.

If the approximated contour has four points (**Line 48**), we assume that we have found the document in the image.

And again, this is a fairly safe assumption. The scanner app will assume that (1) the document to be scanned is the main focus of the image and (2) the document is rectangular, and thus will have four distinct edges.

From there, **Lines 53-57** display the contours of the document we went to scan.

And now let's take a look at our example image:

Figure 2: The second step of building a document scanning app is to utilize the edges in the image to find the contours of the piece of paper.

As you can see, we have successfully utilized the edge detected image to find the contour (outline) of the document, illustrated by the green rectangle surrounding my receipt.

Lastly, let's move on to Step 3, which will be a snap using my four_point_transform functi

Step 3: Apply a Perspective Transform & Threshold

The last step in building a mobile document scanner is to take the four points representing the outline of the document and apply a perspective transform to obtain a top-down, “birds eye view” of the image.

Let’s take a look:

Building a Document Scanner App using Python, OpenCV, and Computer Vision — Python

```
59 # apply the four point transform to obtain a top-down
60 # view of the original image
61 warped = four_point_transform(orig, screenCnt.reshape(4, 2) * ratio)
62
63 # convert the warped image to grayscale, then threshold it
64 # to give it that 'black and white' paper effect
65 warped = cv2.cvtColor(warped, cv2.COLOR_BGR2GRAY)
66 warped = threshold_adaptive(warped, 251, offset = 10)
67 warped = warped.astype("uint8") * 255
68
69 # show the original and scanned images
70 print "STEP 3: Apply perspective transform"
71 cv2.imshow("Original", imutils.resize(orig, height = 650))
72 cv2.imshow("Scanned", imutils.resize(warped, height = 650))
73 cv2.waitKey(0)
```

Line 61 performs the warping transformation. In fact, all the heavy lifting is handled by the `four_point_transform` function. Again, you can read more about this function in last week’s post.

We’ll pass two arguments into `four_point_transform` : the first is our original image we loaded off disk (*not* the resized one), and the second argument is the contour representing the document, multiplied by the resized ratio.

So, you may be wondering, why are we multiplying by the resized ratio?

We multiply by the resized ratio because we performed edge detection and found contours on the resized image of *height=500* pixels.

However, we want to perform the scan on the *original* image, ***not*** the *resized* image, thus we multiply the contour points by the resized ratio.

To obtain the black and white feel to the image, we then take the warped image, convert it to grayscale and apply adaptive thresholding on **Lines 65-67**.

Finally, we display our output on **Lines 70-73**.

And speaking of output, take a look at our example document:

Figure 3: Applying step 3 of our document scanner, perspective transform. The original image is on the *left* and the scanned image on the *right*.

On the left we have the original image we loaded off disk. And on the right, we have the scanned image!

Notice how the perspective of the scanned image has changed — we have a top-down, 90-degree view of the image.

And thanks to our adaptive thresholding, we also have a nice, clean black and white feel t

We have successfully built our document scanner!

All in less than 5 minutes and under 75 lines of code (most of which are comments anyway).

More Examples

The receipt example was all well and good.

But will this approach work for normal pieces of paper?

You bet!

I printed out page 22 of *Practical Python and OpenCV*, a book I wrote to give you a guaranteed quick-start guide to learning computer vision:

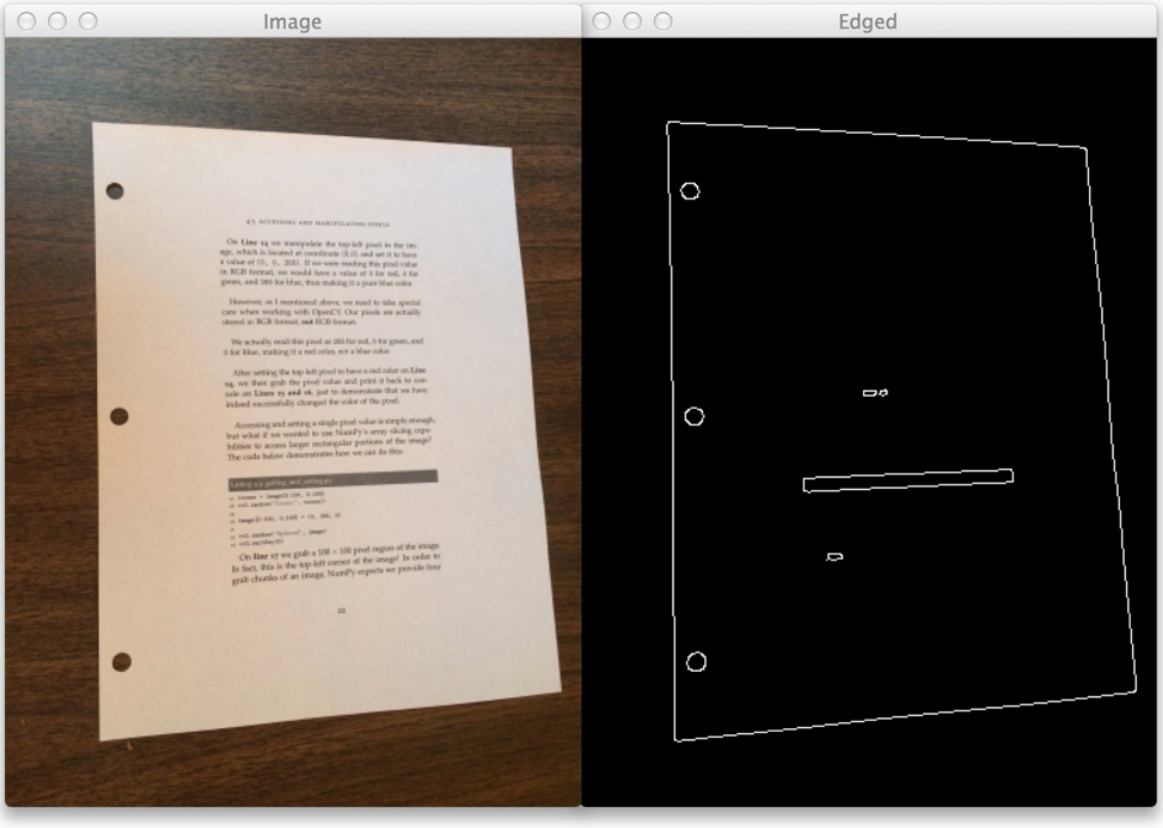


Figure 4: Applying edge detection to scan a document using computer vision.

You can see the original image on the *left* and the edge detected image on the *right*.

Now, let's find the contour of the page:

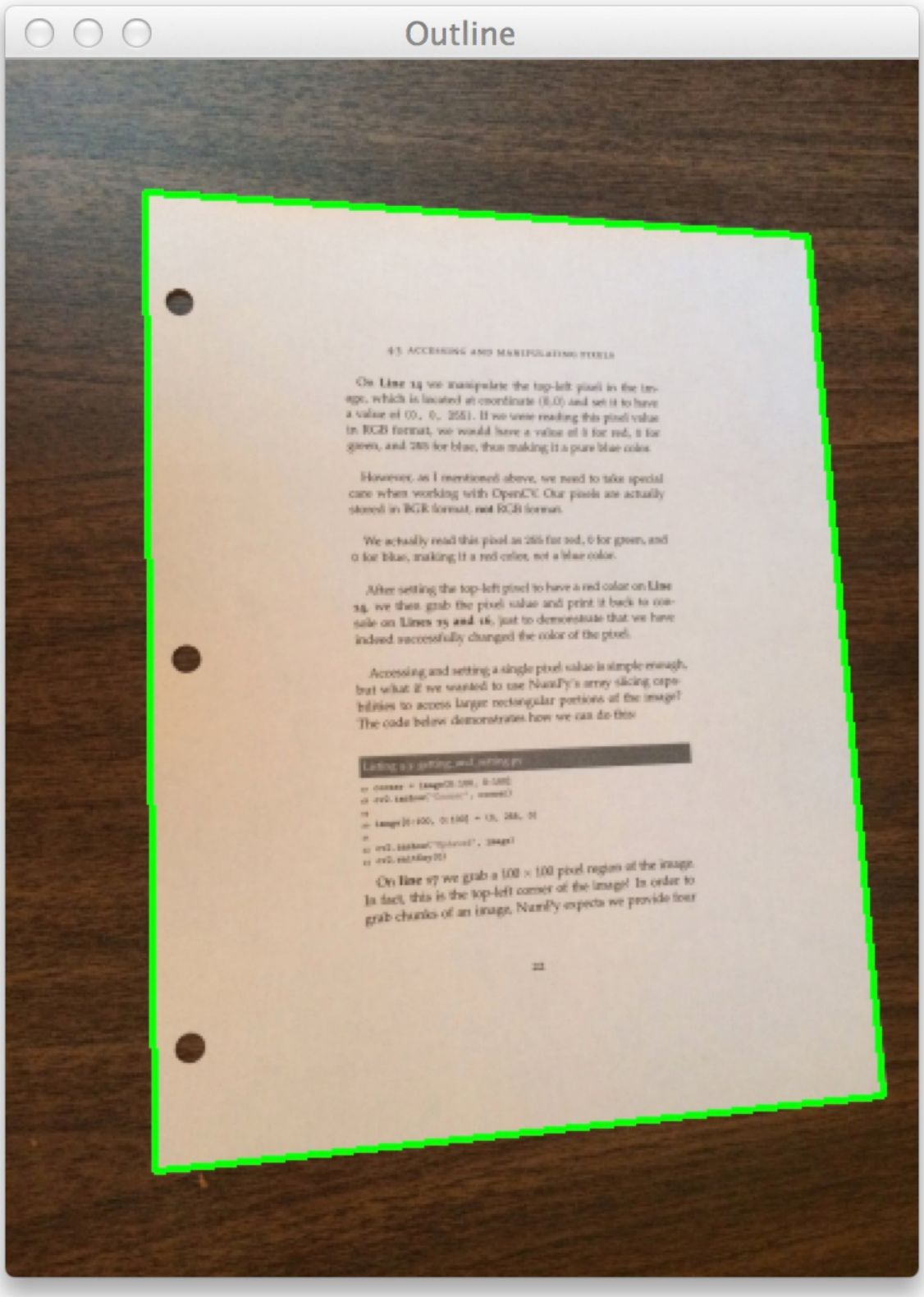


Figure 5: Using the detected images to find the contour and outline of the page to be scanned.

No problem there!

Finally, we'll apply the perspective transform and threshold the image:

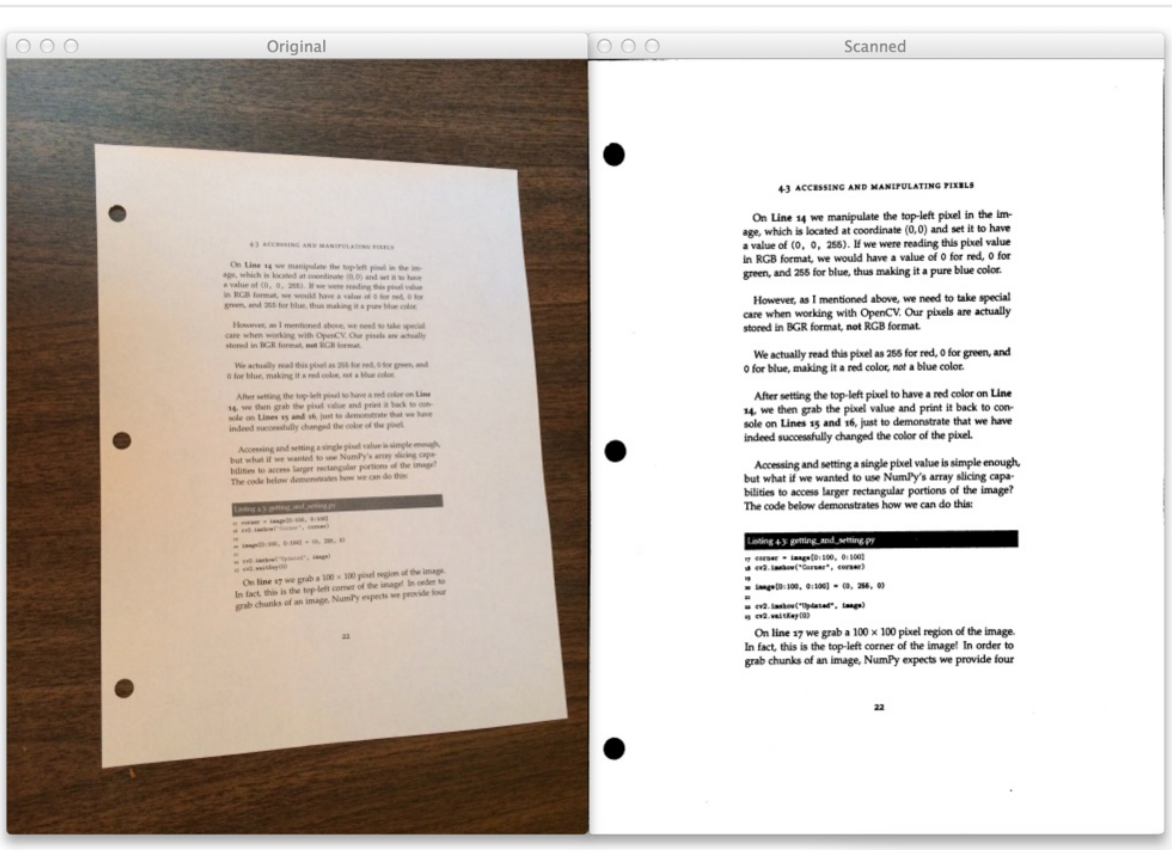


Figure 6: On the *left* we have our original image. And on the *right*, we can see successful!

Another successful scan!

Where to Next?

Now that you have the code to build a mobile document scanner, maybe you want to build an app and submit to the App Store yourself!

In fact, I think you should.

It would be a great learning experience…

Another great “next step” would be to apply OCR to the documents in the image. Not only could you scan the document and generate a PDF, but you would be able to edit the text as well!

Summary

In this blog post I showed you how to build a mobile document scanner using OpenCV in 5 minutes and under 75 lines of Python code.

Document scanning can be broken down into three distinct and simple steps.

The **first step** is to apply edge detection.

The **second step** is to find the contours in the image that represent the document we want to scan.

And the **final step** is to apply a perspective transform to obtain a top-down, 90-degree view of the image, just as if we scanned the document.

Optionally, you can also apply thresholding to obtain a nice, clean black and white feel to the piece of paper.

So there you have it.

A mobile document scanner in 5 minutes.

Excuse me while I call James and collect my money…

Did You Like this Post?

Hey, did you enjoy this post on building a mobile document scanner?

If so, I think you’ll like my book, *Practical Python and OpenCV*.

When you pick up the Case Studies Bundle you’ll learn how to **detect faces in images**, **recognize handwriting**, and **utilize keypoint detection and the SIFT descriptors** to build a system to recognize the book covers!

Sound interesting?

Just click here and pickup a copy.

And in a single weekend you’ll unlock the secrets the computer vision pros use…*and become a pro yourself!*

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).

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

document scanner, **mobile app**, **perspective**, **transform**, **warp**

‹ 4 Point OpenCV getPerspective Transform Example

Thresholding: Simple Image Segmentation using OpenCV ›

226 Responses to *How to Build a Kick-Ass Mobile Document Scanner in Just 5 Minutes*

Aaron Altscher September 2, 2014 at 11:03 am #

REPLY

Very informative and detailed explanation. Everything this blogger publishes is gold!

Adrian Rosebrock September 2, 2014 at 12:16 pm #

REPLY

☐ Thanks, Aaron!

Marx April 24, 2017 at 6:43 am #

REPLY

Hello Adrian. Thanks for your awesome posts! I'm completely blind, and your content has greatly helped me develop a proof of concept prototype in Python for an AI-guided vision system for blind people like me.
Right now, I'm trying to add a function that will instruct a blind user if his or her camera is properly capturing all four points of an object with the largest contours (hence assuming nearest); and
I think I can build on the block of code below to print out statements if the object with largest contours does not have all four points, but: I'm thinking of ways to do this that will allow me to instruct the blind user whether to move the focus of his or her camera to the left, right, up, or down …
– I just need approximations. I think this can be done by getting location coordinates of the missing point/s (out of the 4 supposed points) of the object with the largest contours, and using those values to calculate and print out comprehensible instructions for the end user?
Do you have any suggestions on how this can be done? Thanks in advance! ☐

```
# if our approximated contour has four points, then we
# can assume that we have found our screen
if len(approx) == 4:
>>> screenCnt = approx
>>> break
```

Marx April 24, 2017 at 6:46 am #

REPLY

P.S. This is for providing blind users with guided instructions on how to properly point their camera at a piece of document with text, in order to run the OCR functions of my software…

Adrian Rosebrock April 24, 2017 at 9:31 am #

Hi Marx — this sounds like a wonderful project, thank you for sharing. I think you are on the right track here. Find the largest contour region in the image. If the approximated contour region does not have 4 vertices, tell the end user.

As for determining how the user should move their camera, the angles between the vertices should be (approximately) 90 degrees. Compute the angles between the contours and if the angle is not 90 degrees, you'll know which corner is missing and then be able to give the user instructions on how to move the paper/camera.

Marx April 24, 2017 at 2:50 pm #

Hi Adrian. Thanks for your help! ☐
However, I've been searching and reading up on how to measure the angles of contour vertices for several hours now, but I just can't find something that I can comprehend and use. ☐
I also read up on how to sort contours from left to right and top to bottom, thinking that I'd be able to identify the missing edge and just tell the user to move his or her camera towards that missing edge for now (until I find something that allows me to give the user an approximation on how much to move the camera towards that direction) …
…
Also, I'm thinking if it would be better to implement this for a live camera feed, than for a captured image?
That's mainly because before capturing the image for OCR processing, the user will need to know the right time to capture the image, in real time preferrably …
I'd greatly appreciate your suggestions regarding this matter. Thanks again! ☐

Adrian Rosebrock April 28, 2017 at 9:59 am #

If your goal is to provide real-time feedback to the user, then yes, a live camera feed would be more appropriate. However, this makes the task a bit more challenging due to motion blur. I would instead suggest solving the problem for a single image before you start moving on to real-time video processing.

Skaag Argonius May 11, 2017 at 2:29 pm #

You can eliminate motion blur using a really good camera sensor, and manual control of white balance. This is critical since a lot of sensors come with a controller which by default will try to increase the brightness of the image by capturing multiple frames in succession and adding them together. This process is what creates motion blur so you need to simply disable automatic white balance in the controller, and you'll get clean frames every time. However this also means that in some situations it will be too dark for the sensor to see anything. One way to solve this is to put a large amount of powerful infrared LED lights around or behind the sensor, and remove the infrared filter from the sensor so it becomes sensitive to infrared light. The sensor will not see colors, but for reading text from a page you don't need colors. This way your sensor will see images even in total "darkness" without blinding the non-blind with a potentially strong white light. Reach out to me if you're interested and I will send you information about such a sensor that we use in my company.

Adrian Rosebrock May 15, 2017 at 9:04 am #

Thanks for sharing Skaag!

Bob September 2, 2014 at 4:43 pm #

REPLY

how hard would it be to pan parts of the document so it all fits into one panoramic view?

Adrian Rosebrock September 3, 2014 at 7:25 am #

REPLY

Awesome question, thanks for asking! Substantially harder, but certainly not impossible. Basically, your first step would be to perform "image stitching" to generate the larger panoramic picture. From there, you could essentially use the same techniques used in this post. Would you be interested in a blog post on image stitching?

Lúcio Corrêa November 6, 2014 at 9:58 am #

REPLY

Would it be better to do stiching after perspective transform and threshold or before?

Adrian Rosebrock November 6, 2014 at 2:40 pm #

REPLY

It depends on what your end goal is, but I would do stitching first, then perform the perspective transform.

Jose Luis September 4, 2014 at 7:49 pm #

REPLY

Hello Adrian,

An example of image stitching would be great!!

Thanks for your awesome posts

Can September 10, 2014 at 10:19 am #

REPLY

How can you export to Appstore your applications write with Python?

To my knowledge you cant. You must use c++.

Adrian Rosebrock September 10, 2014 at 11:19 am #

REPLY

Hi Can, you are correct. You must first convert it to iOS, but the algorithm is the same. All you have to do is port the code.

Alex December 8, 2014 at 7:21 pm #

REPLY

Or run the python code on a server and upload the image from your phone.

Adrian Rosebrock December 8, 2014 at 7:41 pm #

REPLY

Exactly. And for applications that don't require real-time processing, I highly recommend doing this. You can update your algorithms on the fly and don't have to worry about users updating their software.

Usama April 21, 2015 at 12:12 pm #

Hi Adrian,

I want to learn how do I run a server so that my application gets process in my computer?

Adrian Rosebrock April 21, 2015 at 1:20 pm #

Hi Usama, I don't have any posts on creating a web API/server for computer vision code yet, but it's in the queue. I should have a blog post out on it within the next few weeks.

---

raphael February 16, 2016 at 2:37 pm #

REPLY

Hi, how can you run a python code on a server? Where can I find a step by step on how to do this? thanks!

Adrian Rosebrock February 16, 2016 at 3:34 pm #

You can see this tutorial on converting an image processing pipeline to an API accessible via a server.

---

bidder February 9, 2016 at 3:40 pm #

REPLY

use kivy (is a python framework to build mobile apps) □

Can September 10, 2014 at 11:19 am #

REPLY

Also why you need scikit-image Open CV already have adaptive threshold?

Adrian Rosebrock September 10, 2014 at 3:13 pm #

REPLY

The scikit-image adaptive threshold function is more powerful than the OpenCV one. It includes more than just Gaussian and mean, it includes support for custom filtering along with median (Although I only only use Gaussian for this example). I also found it substantially easier to use than the OpenCV variant. In general, I just (personally) like the scikit-image version more.

Atanas Minev September 16, 2014 at 10:17 am #

REPLY

Cool (post-OCR) improvement would be to recognize receipts and communicate information with budgeting app ☺

Adrian Rosebrock September 16, 2014 at 10:21 am #

REPLY

Agreed! That would be a really fantastic improvement. And I think (most) budgeting apps provide an API to interface with the app.

Alex October 1, 2014 at 12:08 pm #

REPLY

Hello guys,

Great Job, good instructions, perfect !!!
Do you know, if and how this could be work on Android devices ?

Adrian Rosebrock October 1, 2014 at 12:57 pm #

REPLY

Absolutely, you would just have to look into using the Java OpenCV bindings.

Andrea August 18, 2015 at 7:58 am #

REPLY

This example may be a good starting point: https://github.com/jhansireddy/AndroidScannerDemo

Toni October 15, 2014 at 9:05 am #

REPLY

Very informative. Great!! Thanks.

Matthew Nichols October 15, 2014 at 9:46 am #

REPLY

What applications are you using to code? I don’t recognise the icons.

Cheers,

Great post btw

Adrian Rosebrock October 15, 2014 at 10:09 am #

REPLY

I use Sublime Text 2 most of the time.

qubodup October 15, 2014 at 11:42 am #

REPLY

I’m sorry, what constitutes the “mobile” part of a mobile doc scanner?

To me: it runs on a smartphone that runs Android, iOS, WP8 or Whatever the name of the BB OS is.

Adrian Rosebrock October 16, 2014 at 7:06 am #

REPLY

You are correct, the “mobile” part of a mobile document scanner is an app that runs on a smartphone device that utilizes the smartphones camera to capture documents and “scan” them.

Sasa November 5, 2014 at 9:29 am #

REPLY

I would like to see some OCR on it or on just some simple text or numbers.

Adrian Rosebrock November 5, 2014 at 10:21 am #

REPLY

Hi Sasa, good point. OCR would be a really great extension. If you’re interested in recognizing text, especially handwritten digits, definitely take a look at my Practical Python and OpenCV book. Inside I cover recognizing handwritten digits. Definitely not the same as OCR, but if you’re interested in recognizing text, it’s a pretty cool place to start.

Gaduc December 14, 2014 at 9:32 am #

REPLY

Yeah!
I played with that some time ago in order to scan books.
But I faced to a harder problem: pages of a book are not flat but warped.
I was able to isolate the curve the page made from the flatbed (*).
And then?
There is two transformation to achieve:
– first to convert from a warped surface to a flat one ;
– secondly to convert from a perspective surface to a rectangular one (easy as you did it).
How can we do the first conversion? Bilnear formula?

Another problem I used to face to is to progressively cancel the shade that appears as the distance between the page surface and the camera increases.

(*) There are different methods to achieve that:
– take a shot at 45° ;
– use the shade as an approximate distance from the lens.

I you have idea how to compute that …

db

Manuel October 16, 2016 at 2:42 pm #

REPLY

I would like to know about this too.

SOHAM December 22, 2014 at 11:44 pm #

REPLY

Nice and Clear understanding…

Milo Hyson December 26, 2014 at 2:00 pm #

REPLY

This is a great approach when dealing with small things like a typical receipt. But unless you’re going to take multiple pictures and stitch them together, the resolution will suffer as the item to be scanned gets larger and you have to pull the camera back to get it all into frame. This is where purpose-built document scanners really shine. They can capture a metre-long receipt at full resolution.

mohammad February 10, 2015 at 12:54 am #

REPLY

hi

What IDE do I use Python and Open CV? Please download link

Adrian Rosebrock February 10, 2015 at 6:43 am #

REPLY

Hi Mohammad. I use Sublime Text 2 and PyCharm. Definitely check them out!

mohammad February 11, 2015 at 4:15 am #

REPLY

Thank
You're very cool

Richard March 18, 2015 at 4:57 pm #

REPLY

Hi great post!! I really like your web and your tutorials are the best! Just a doubt… is it better to use a bilateral filter instead of a gaussian one to smooth the image? If a recall right, the bilateral filter preserves better features like edges…
Great post, keep doing this great job! Thanks.

Adrian Rosebrock March 18, 2015 at 6:00 pm #

REPLY

Great observation! A bilateral filter could definitely be used in this situation. However, I choose to use Gaussian smoothing, simply because there were less parameters to tune and get correct. Either way, I think the result would be the same.

Richard March 23, 2015 at 2:34 pm #

REPLY

Thanks!

Jasper April 20, 2015 at 10:36 am #

REPLY

Hi thanks for the post, really helpful! I noticed the contour detection approach isn't working to well when part of the document to capture is offscreen. Any idea's on how to solve this? TIA

Adrian Rosebrock April 20, 2015 at 10:44 am #

REPLY

In general, you'll need the entire document to be in your image, otherwise you won't be able to perform a reliable perspective transform.

Joe Landau June 5, 2015 at 1:43 am #

REPLY

This exercise requires scikit-image, which someone who just installed OpenCV and Python on a new Raspberry Pi 2 would not have. Installing scikit-image in turn seems to require scipy, which I am trying to install (slowly using pip install -U scipy) at this very minute. Perhaps a setup step would help.

Adrian Rosebrock June 5, 2015 at 6:15 am #

REPLY

Good point Joe. How are you liking your Pi 2?

Joe Landau June 5, 2015 at 1:48 pm #

REPLY

So far the Pi 2 is doing well. The installation of scipy took between 1 and 2 hours (I didn't time it) and then scikit-image took only minutes. Using the browser thru VNC displaying 1920 x 1080 is a bit slow, I'll have to work with a smaller screen. I won't know if the Pi 2 is adequate for my application until I get there–if the application works but is slow I will have to go to a faster system, maybe a Tegra.

Adrian Rosebrock June 5, 2015 at 2:09 pm #

REPLY

If you're doing most of your work via terminal, I would suggest using SSH forwarding instead of VNC: `$ ssh -X pi@your_ip_address`. You'll be able to execute your Python scripts via command line and the OpenCV windows will still show up.

Joe Landau June 5, 2015 at 8:39 pm #

REPLY

I withdraw the information about installing scikit-image. I didn't realize that the first try had failed. In fact, it took over an hour.

Alexander June 23, 2015 at 1:49 pm #

REPLY

Hi Adrian,

Thank you so much for the great article and for the rest of your series!

I stumbled with the task of how to correct the document scan of a sheet of paper that has been folded 2 or 4 times?
Could you please take a look at my question here: http://stackoverflow.com/questions/31008791/opencv-transform-shape-with-arbitrary-contour-into-rectangle.

Will appreciate if you could give some direction on how to achieve this.

Adrian Rosebrock June 23, 2015 at 6:33 pm #

REPLY

If the paper has been creased or folded, then you'll want to modify Line 48: `if len(approx) == 4:` and find a range of values that work well for the contour approximation. From there, you'll want to find the top-left, top-right, bottom-right, and bottom-left corners of the approximation region, and then finally apply the perspective transform.

Alexander June 24, 2015 at 4:05 pm #

REPLY

Adrian, thank you for the answer!
But I need to clarify: I'm able to find the corners of a folded\creased paper and perform the proper perspective transform using those four points. In other words, your whole perfect algorithm works fine. What I'm doing is trying to take a step further ☐
The step I'm asking about is how to "straighten" (recover rectangular shape of) the paper with OpenCV? I.e. stretch it so that its edges become touching surrounding rectangle.

J July 15, 2015 at 1:03 am #

REPLY

I'm really not understanding where you put the path to the file that will be scanned. Can you give an example of proper usage of the code on lines 10-13?

Adrian Rosebrock July 15, 2015 at 6:39 am #

REPLY

If you have an image named page.png in an images directory, then your example command would look like this:

```
$ python scan.py --image images/page.jpg
```

Definitely download the code to the post and give it a try!

fariss August 13, 2015 at 12:14 pm #

REPLY

hi,

your source code was very helpful,
what's the functions to do some image processing for the scanned image (contrast, btightness, contrast…) ?, i removed the part that gives the image the 'black and white' paper effect…

REPLY

Adrian Rosebrock August 14, 2015 at 7:23 am #

I would suggest taking a look at the official OpenCV docs to get you started.

REPLY

Bhuvesh August 25, 2015 at 5:22 am #

Great work!!
Sir, can i use this on Raspberry Pi B+ or 2 ?
If Yes, than How!
Please guide me I'm working on some related project.

REPLY

Adrian Rosebrock August 25, 2015 at 6:35 am #

You could certainly use either, but I would suggest going with the Pi 2. From there, you can follow my OpenCV install guide for the Raspberry Pi. Once you have OpenCV installed on your system, you should be able to download and execute the code in this post.

REPLY

Mohd Ali October 8, 2015 at 1:48 am #

I keep getting this error at line 37.

```
(cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

ValueError: too many values to unpack

I've triend this script on 3-4 images but getting same error. I tried to debug, but didn't succeed.

REPLY

Adrian Rosebrock October 8, 2015 at 5:55 am #

I've mentioned the solution to this problem many times on the PyImageSearch blog before. The reason you are getting this error is because you are using OpenCV 3 — this post was written for OpenCV 2.4, well before OpenCV 3 was released. Please see this post for a discussion on how the cv2.findContours return signature changed in OpenCV 3.

REPLY

ashish February 9, 2016 at 1:22 am #

try this….

```
(cnts) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

REPLY

Adrian Rosebrock February 9, 2016 at 3:58 pm #

This is incorrect. For OpenCV 3, it should be:

```
(_, cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

REPLY

parviz October 15, 2015 at 7:56 am #

dude! you are so cool 😀 thanks buddy

REPLY

Adrian Rosebrock October 15, 2015 at 12:28 pm #

REPLY

No problem Parviz ☐

Hacklavya October 15, 2015 at 6:23 pm #

REPLY

I was very happy to see this tutorial, but then I found that you didn't tell howto install OpenCV 2.4.X with Python 2.7

So please tell me link to do that.

or more generally there is no date-wise posts in here, so that I see what is all on your website.

and how can I search a particular post on your website?

Adrian Rosebrock October 16, 2015 at 6:17 am #

REPLY

There is a search bar at the bottom-right corner of the sidebar on every page on the blog. As far explaining how to install OpenCV 2.4 and Python 2.7, I cover that in this post and in my book.

Hacklavya October 25, 2015 at 3:28 am #

REPLY

I tried many help from internet to install scikit image,
but this line:

```
1 from skimage.filter import threshold_adaptive
2 giving following error
3 ImportError: No module named skimage.filter
```

please tell us how to fix it.

Adrian Rosebrock October 25, 2015 at 6:21 am #

REPLY

Please see the official scikit-image install instructions. All you need to do is let pip install it for you:

```
$ pip install -U scikit-image
```

Dirk Josefiak October 13, 2016 at 3:33 pm #

REPLY

thanks a lot for all your work and this tipp also

Cyrus Smith November 4, 2015 at 11:48 am #

REPLY

Hello, Adrian!
I have downloaded your code and tried to launch it on my computer and I failed.
I use PyCharm 4.5, OpenCV300 and Python 2.7.
I think it's not versions thing.
There is a line on main code (scan.py) :

from skimage.filter import threshold_adaptive

But there is no skimage folder in your project…
What should I do?
Thanks.

Adrian Rosebrock November 4, 2015 at 1:20 pm #

REPLY

The error can be resolved by installing scikit-imagee:

```
$ pip install -U scikit-image
```

However, keep in mind that this tutorial is for OpenCV 2.4.X, not OpenCV 3.0, so you'll need to adjust **Line 37** to be:

```
(_, cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

Mike Mehr November 19, 2015 at 5:06 am #

REPLY

Hi Adrian,

I also found that I had to add parens to the print statement arguments on lines 32, 56, and 73 when running with Python 3.5 (they are optional in v2.7).

There is also a warning that the skimage.filter submodule has had a name change to skimage.filters, so I added the 's' on line 7 and now it runs without any errors or warnings. The warning says the old name is deprecated and will be removed in version 0.13. This occurs for both virtual environments/versions of Python.

It seems like these changes might impact some of your other code on the site as well.

Best regards,
Mike

Adrian Rosebrock November 19, 2015 at 6:16 am #

REPLY

Thanks for the update Mike. I'm still trying to figure out a decent solution to handle the OpenCV 2.4 vs. OpenCV 3 issue. As a blog post that will come out this Monday will show, many users are still use OpenCV 2.4. (although that number growing and will continue to grow). But since this is a *teaching* blog, it's not exactly easy to switch the code from one version to another if tow equally sized groups of people are using different versions.

Teddy November 21, 2015 at 11:02 pm #

REPLY

do you have code for building in OCR into the Scanner?

Adrian Rosebrock November 22, 2015 at 7:15 am #

REPLY

Not yet, but that's something I would like to cover in a future blog post.

Gary King November 23, 2015 at 8:42 am #

REPLY

Adrian,

I am a novice Pyphon developer, but was wondering if the scanning and OCR reading could have all been achieved via Javascript/HTML5?. I find it very hard trawling the javascript libraries to find something that will mostly do this. I don't mind glue things together, but would prefer it if most of the hard work were done via some commercial library if possible, and obviously don't mind paying for such a product. Do you know of any libraries that might fit this requirement?.

Adrian Rosebrock November 23, 2015 at 11:28 am #

REPLY

I don't use JavaScript often, but I've heard of Ocard.js being used for OCR with JavaScript. I haven't personally tried it out, but it might be worth looking into.

Mouiche December 3, 2015 at 3:11 am #

REPLY

Dear Adrian and other members
Your job is highly interesting, I have a related project for 3 weeks.
– first, I have to take one scanned page that can be inclined in all directions and do the trans

– Secondly, snap two pages of a book, transform the edge's curves into straight lines and finally have these pages in a rectangle in pdf.
Now, I am trying to use your code as a model but i don't have openCV in my computer. which module or function can i use? I have "skimage" .
Are there other links or documents that can help to solve this problem.
Thank you.

Adrian Rosebrock December 3, 2015 at 6:22 am #

REPLY

Having scikit-image is a good start, but I would really recommend getting OpenCV installed as well. I have tutorials detailing how to get OpenCV installed on your system here. I'm not sure if I understand the second part of your question, but if you're trying to "scan" two pages of a book side-by-side, you can try finding the center spine of the book using edge detection. Once you have it, you can apply perspective transforms like the ones used in this post.

Mouiche December 3, 2015 at 8:38 am #

REPLY

Thank you, I am trying now to install openCV as well.
In my second task I am trying to scan two pages of a book not side-by-side. So I will be asked to manage the horizontal lines that will look like curves and the middle-line between these pages that will not be well seen after scanning, but i need to manage it in this job

Mouiche December 10, 2015 at 5:19 am #

REPLY

Please,

can someone help me on how to proceed for the case of two scanned pages of a book. Two pages are scanned together and i need to do the transformation to have the plat (or flat) rectangular form of these pages.

Adrian Rosebrock December 10, 2015 at 6:48 am #

REPLY

As I suggested over email, the try to detect the "spine" of the book using the Canny edge detector. If you can determine where the boundaries of the book and the top/bottom intersect, you can essentially split the book in two, determine your four points for the transformation, and apply the perspective warp.

Steve Dyson December 14, 2015 at 6:14 pm #

REPLY

Does anyone have something similar built for JS? We are working on a document scanner and would love some help on where to get started.

OCR is not needed – we only need the cropping, alignment, and conversion to b&w.

Thank you!

Steve

Adrian Rosebrock December 15, 2015 at 6:18 am #

REPLY

Computer vision is slowly starting to expand to the browser (since computers are becoming faster), but currently there is no full port of OpenCV to JavaScript, making it hard to build applications like these.

If you want to do something similar to this in JavaScript, I would suggest wrapping the document scanner code into an API, like this tutorial does. From there you can make requests to the API from JavaScript and display your results.

Sainandan December 16, 2015 at 1:27 am #

REPLY

Hey Adrian,

This mini project is really handy in terms of usage and fundamental exposure to IP.

But how do you integrate this python code into a mobile android application ?

Adrian Rosebrock December 16, 2015 at 6:32 am #

REPLY

I would suggest wrapping your Python code using a web framework such as Django or Flask and then calling it like an API. You can find an example of doing this here.

singh December 18, 2015 at 10:46 pm #

REPLY

Hi, scan.py hangs after Step 1. I left it running for more than hour but still didn't finish. I am on Mac OS X El-captain. Using the Opencv 2.49 and Python 2.7. I have all the modules installed. I tried your transform_example.py and that works fine. Am I missing anything here? Do I need to hit a command or something?

Commenting out the following helped. Thanks. (sorry new to Python and OpenCV but loving it so far).
#cv2.waitKey(0)
#cv2.destroyAllWindows()

Adrian Rosebrock December 19, 2015 at 7:45 am #

REPLY

Interesting, I'm not sure why it would take so long to process the image during Step 1. The cv2.waitKey call wouldn't matter, provided that you clicked the window and pressed a button to advance the process. The cv2.waitKey method pauses all execution until a key is pressed.

singh December 18, 2015 at 11:21 pm #

REPLY

How do i save the final image in a new file?

Adrian Rosebrock December 19, 2015 at 7:47 am #

REPLY

You can use the cv2.imwrite method:

```
cv2.imwrite("output.png", warped)
```

And if you're just getting started learning Python + OpenCV, you should definitely take a look at Practical Python and OpenCV. This book can get you up to speed with Python + OpenCV very quickly.

Mukundan January 6, 2016 at 1:26 am #

REPLY

Thanks for this informative article

deepak January 23, 2016 at 5:55 pm #

REPLY

hi Adrian,
really nice tutorial there.I am currently trying to follow it to build an app of my own. I wanted to ask if its possible to generate a crude 3d wireframe model from a photo with probably the users help in correcting the edges. basically take 2-3 photos from a phone and then process it to create a simple 3d wireframe model.

Adrian Rosebrock January 25, 2016 at 4:13 pm #

REPLY

It's certainly possible, but you'll realistically need more than 2-3 photos. I would suggest reading up on 3D Reconstruction.

Dino February 3, 2016 at 4:50 am #

REPLY

Hi Adrian,

In line 37:
(cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE

Does it better to use cv2.RETR_EXTERNAL as second parameter in findContours ?

Adrian Rosebrock February 4, 2016 at 9:20 am #

REPLY

For this project, yes, I would suggest using cv2.RETR_EXTERNAL. I should have mentioned that in the original post ☐ Thanks for noting this!

Linus February 4, 2016 at 4:37 am #

REPLY

Hi, and thanks for a great tutorial.

To the point: I'm having trouble with the part where you sort the contours according to area, around line 38 in the sample code. The problem is that the contours that correspond to big blobs (such as the receipt border) does not at all end near the top of the list. Instead the largest contour is a contour around the WHOLE FOOD title.

I have checked that the contours are indeed sorted in descending order by the value returned by contourArea(), and sifted through all contours to verify that the contour enclosing the receipt indeed is present in the list. However, the area corresponding to that contour is uncannily small (about 12 pixels).

The issue, I believe, is that findContours() here finds the contours of the canny-edges rather than the enclosed objects. However, why this happens to me and none of you is beyond my comprehension. Maybe I have an unknown ascendency to Murphy.

Anyway, does anyone here have any I idea what might be going on?

Also, I'm using opencv 3.1.0 and python 3.4

Adrian Rosebrock February 4, 2016 at 9:10 am #

REPLY

Just to clarify, are you using the same Whole Foods receipt as I am in the blog post? It sounds like there might be an issue segmenting the actual receipt from the image. The *likely* issue is that that the border of the receipt outline is not "complete". Can you take a second look at the edge map of the receipt and ensure that it is all one continuous rectangle?

EDIT: Try changing cv2.RETR_LIST to cv2.RETR_EXTERNAL in cv2.findContours and see if that resolves the issue.

Linus February 5, 2016 at 11:06 am #

REPLY

Yes I use the same image as in the post. Well almost. I used gimp to cut out the part containing the receipt.

Anyhow, the problem was a discontinuity in the edge map, just as you suspected. And a solution was to decrease the size of the Gaussian filter ((3,3) worked for me). Maybe the reason that I get this problem, but not anybody else, is that I use an original image of significantly lower resolution. I suspect your mobile camera does better than 496×669?

Thanks for quick response!

Adrian Rosebrock February 6, 2016 at 9:59 am #

REPLY

Indeed, the cropping in Gimp must have caused some sort of issue. My camera is an iPhone, so the resolutions is very high. I actually reduced the resolution of teh original image for this example. In fact, I wouldn't suggest processing images larger than 640 x 480 pixels if at all possible. It's normally to resize images prior to processing them. The less data there is to process, the faster the algorithm will run. And most computer vision functions expect smaller image sizes for more accurate results (edge detection and finding contours, for example).

Ghassan February 14, 2016 at 9:04 am #

REPLY

if we put image scanned from scanner it will show this error

AttributeError: 'NoneType' object has no attribute 'shape'

how to pass it ?

Adrian Rosebrock February 14, 2016 at 9:48 am #

REPLY

Anytime you see an error related to an image being NoneType, it's 99% of the time due to an image not being loaded from disk properly or read from a stream. Make sure the path you supply to cv2.imread is valid. I demonstrate how to execute the Python script via command line (and pass in the image path argument) in this post.

sagar February 16, 2016 at 2:57 pm #

REPLY

Hi Adrian,

I run your code on Anaconda, Windows. It runs perfectly.
I want to build an mobile android app on android studio, There are many functionalities including document scanning. As user have to get picture of any object, app may be responsible to get result out of it, But this code need some bindings with android code.
How to do this. How to integrate both these together ?

Adrian Rosebrock February 16, 2016 at 3:33 pm #

REPLY

I personally don't use Windows or Visual Studio, nor to I do any coding for the Android environment. That said, you have two options:

1. Convert the code from Python to Java + OpenCV (there are OpenCV bindings available for the Java programming language).
2. Wrap the document scanner code as computer vision API, then upload the image from the Android app to the API, process it, and then return the results.

Reza March 30, 2016 at 4:50 pm #

REPLY

If the second approach (using HTML5/javascript) can be implemented, then the export to mobile phones as a native application (for instance apk for android) would be very easy using CORDOVA.

Adrian Rosebrock March 30, 2016 at 5:13 pm #

REPLY

Which is the exact same approach I took when building both Chic Engine and ID My Pill ☐

I also demonstrate how to wrap a computer vision app as an API and access it via PhoneGap/Cordova inside the PyImageSearch Gurus course

Damien Mac Namara March 14, 2016 at 9:35 pm #

REPLY

Could this be easily modified for video?

Adrian Rosebrock March 15, 2016 at 4:35 pm #

REPLY

Absolutely. You just need to wrap the code in a look that access a video stream. This blog post should be a good starting point.

Damien Mac Namara March 22, 2016 at 7:02 pm #

REPLY

Thanks Adrian for your quick response. I cant recommend you enough.

Adrian Rosebrock March 24, 2016 at 5:23 pm #

REPLY

No problem, happy to help ☐ And thanks for the kind words!

sunchy11 March 16, 2016 at 2:52 am #

REPLY

Hi, Adrain,

I try a lot of different coupons except your sample "whold food "one.

I got the difficulties in finding the four points of the edge.
It looks like the “approx” is not 4 points for some of them.

so the error is ‘screenCnt’ si not defined.

Thank you !

Adrian Rosebrock March 16, 2016 at 8:09 am #

REPLY

If the approximated contour does not have 4 points, then you’ll want to play with the percentage percentage parameter of cv2.approxPolyDP Typical values are in the range of 1-5% of the perimeter. You can try working adjusting this value to help with the contour approximation.

Daniel Bornman April 7, 2016 at 4:06 pm #

REPLY

I’m having issues with getting a clean contour that represents a full piece of paper. My paper contour is represented by two separate tuples in the cnts array. One tuple is for the left and bottom edge, and a distance away is the tuple for the top and right edge. Adjusting parameters within the cnts array is too late to find a all encompassing document contour.

I tried changing the parameter in findContours() as suggested above from cv2.RETR_LIST to cv2.RETR_EXTERNAL but that did not fix the problem.

I took a photo with my iphone of a 8×11 piece of paper with regular type against a plain dark background. I intentionally took it at a slight angle to test the transform function. It appears that the assumption of 4 clean points is failing.

Adrian Rosebrock April 8, 2016 at 12:55 pm #

REPLY

If you’re not getting one contour that represents the *entire* piece of paper, then the issue is likely with the edge map generated by the Canny edge detector. Check the edged image and see if there are any discontinuities along the outline. If so, you’ll want to tune the parameters to cv2.Canny to avoid these problems *or* use a series of dilations to close the gaps in the outline.

Jim October 17, 2016 at 11:30 am #

I got every thing installed, all very smooth, but experiencing the same problem

STEP 2: Find contours of paper
Traceback (most recent call last):
File “scan.py”, line 63, in
cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 2)
NameError: name ‘screenCnt’ is not defined

i tried the auto_canny and still have same error.

Certain problem is with me, but not sure where. Thanks

Adrian Rosebrock October 17, 2016 at 2:01 pm #

If the screenCnt is None it’s because there are no contours in your image that contain 4 vertices. Take a look at your edge map and explore the contours. You might need to tune the value of the contour approximation.

K van de Maan April 9, 2016 at 2:40 pm #

REPLY

thank you

TD April 13, 2016 at 12:57 am #

REPLY

Hi Adrian,

Great article. I am encountering the issue below when following your instruction. Please advi
Note that I am using openCV (3.1), python (3.5.1), numpy (1.11.0) & scikit-image (0.12.3)

….site-packages/skimage/filters/thresholding.py”, line 72, in threshold_adaptive
“block_size {0} is even.”.format(block_size))
ValueError: The kwarg block_size must be odd! Given block_size 250 is even.

Few steps I revised in order to make it worked.
+ parenthesis for the print command: print (“STEP 1: Edge Detection”)

+ skimage.filters —> skimage.filters

+ comment out the following:
# cv2.waitKey(0)
# cv2.destroyAllWindows()

Adrian Rosebrock April 13, 2016 at 6:55 pm #

REPLY

Thanks for sharing TD. It looks like the function signature to threshold_adaptive changed in the latest release of scikit-image. I’ll need to take a closer look at this. I’ll post an update to the blog post when I have resolved the error.

UPDATE: In previous versions of scikit-image (<= 0.11.X) an even block_size was allowed. However, in newer versions of scikit-image (>= 0.12.X), an *odd* block_size is required. I have updated the code in the blog post (along with the code download) to use the correct block_size so everything should once again work out of the box.

Mickey Friedman April 20, 2016 at 12:15 pm #

REPLY

Adrian, this is wonderful!

Unfortunately my code doesn’t run past Step 1…it just stops before “(cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]”

Do you know how I can fix this?

Adrian Rosebrock April 20, 2016 at 5:59 pm #

REPLY

Hey Mickey — please read the previous comments, specifically my reply to “Ashish”. It details a solution to what is (likely) causing your problem with cv2.findContours.

Chuck May 3, 2016 at 10:06 am #

REPLY

Hey man,
you’re amazing!!!!

I’m a German student, and I’m working right now with opencv and Python.
I installed the opencv and python with another post of you.

Now, I wanted to scan a receipt, and searched in Internet. And what did I found? A post you wrote. ☐
thanks a lot!

Adrian Rosebrock May 3, 2016 at 5:44 pm #

REPLY

Awesome! I’m glad I could be of help Chuck! 😀

RANVIR May 25, 2016 at 12:48 am #

REPLY

It is working good with only for the given example but not working in any other image. so please make it dynamic so it can recognize edge of any image ie, in any color any light.

Adrian Rosebrock May 25, 2016 at 3:26 pm #

REPLY

It's hard to ensure guaranteed edges in any color or lighting conditions, but you might want to try the automatic edge detector.

Sid May 27, 2016 at 7:07 am #

REPLY

Hey Adrian,

My current set up is OpenCV 3.1.0 + RPi Model B + Python2.7.9

I've followed your tutorial on installing OpenCV 3 on the Pi. Did that include installing the scikit-learn module ?

I tried running the code the code and got an error : "No module named skimage.filters" on line 7.

What are the changes needed for the code to work on OpenCV 3? Thanks.

Adrian Rosebrock May 27, 2016 at 1:27 pm #

REPLY

You'll need to install scikit-image on your system:

```
1 $ workon cv # to access your virtual environment
2 $ pip install -U scikit-image
```

Sid May 29, 2016 at 8:21 am #

REPLY

Hey Adrian,

I tried the "import Scipy" works out of the cv environment

When I'm in the cv environment it gives the "No module named Scipy" error!!

Is there a way to shift the Scipy folder to the correct path?

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

REPLY

You need to install SciPy into your virtual environment:

```
1 $ workon cv
2 $ pip install scipy
```

gigi June 2, 2016 at 8:14 am #

REPLY

Hey Adrian

Thx for your Kick-Ass-Sample-Code.

One question though:
Line 67: warped = warped.astype("uint8") * 255

I don't really get what's going on here (and why).

Adrian Rosebrock June 3, 2016 at 3:09 pm #

REPLY

After coming out of the threshold_adaptive function, we need to ensure that the image is an 8-bit unsigned integer data type, which is what OpenCV expects.

friedman June 2, 2016 at 5:04 pm #

REPLY

Hi Adrian!

I was wondering if there was a way to adjust the document scanner's sensitivity to edge dete

surfaces. Is there anything i can do about contouring or thresholding?

Adrian Rosebrock June 3, 2016 at 3:03 pm #

REPLY

You can control the edge detection process via the two parameters to the cv2.Canny function. You can read more about these parameters here. But in general, you're going to have a real hard time detecting faint edges against a white surface. Edge detection requires that there be *contrast* between the background and foreground.

uma June 3, 2016 at 1:29 am #

REPLY

hi….
can i get certificate after completion of this course?

Adrian Rosebrock June 3, 2016 at 3:00 pm #

REPLY

I only offer a Certificate of Completion inside the PyImageSearch Gurus course. A Certificate of Completion is not provided for the free OpenCV/Image Search Engine courses.

abggcv June 9, 2016 at 12:07 am #

REPLY

This code did not work on the image of a graph I have:
http://www.dropbox.com/s/buexnnip3z4x4sl/2011-01-17.jpg?dl=0

I ran your code and it did not give me the edges as expected. This code is not generic to be used to scan any kind of document. It would be nice if a generic code or approach can be suggested because that is what the professional scanning apps does.

User should not have to play with parameters like size, gaussian blur

Adrian Rosebrock June 9, 2016 at 5:16 pm #

REPLY

While there are such things as "generic edge detectors", they normally require a bit of machine learning to use. In fact, much of computer vision and machine learning *is* tuning parameters and learning how to tune them properly. Anyway, you might want to give the auto_canny function a try for parameter free edge detection.

Ramon Gajardo June 16, 2016 at 1:40 pm #

REPLY

Please any idea because the line code the scan documents
warped = threshold_adaptive(warped, 255, offset=11)
take too many time like 30 sec.,

Thank you

Adrian Rosebrock June 18, 2016 at 8:20 am #

REPLY

If it's taking a lot time to process your image, then your image is likely too large. Resize your image and make it smaller. The smaller your image is, the less data there is to process, and thus the faster your program will run.

Jon June 20, 2016 at 2:57 pm #

REPLY

Small question, in this line:

cv2.drawContours(image, [screenCnt], -1, (0, 255, 0), 2)

What does the brackets does specifically around screenCnt? Also, is there an other tutorial where the functions, algorithms and arguments are explained or we need to look at the OpenCV documentation?

Adrian Rosebrock June 20, 2016 at 5:21 pm #

REPLY

The brackets simply wrap the contours array as a list, that's all. I would suggest either refer to the OpenCV documentation or go through Practical Python and OpenCV for a detailed explantation of cv2.findContours.

Em June 21, 2016 at 4:24 pm #

REPLY

Hi Adrian,
Beautiful work.
Is there a way to use houghtransform (or some other command) to close open contours? Can you use houghtransform over canny? What would that look like?

sometimes 3 out of four edges of a document come out clearly in pictures, but the fourth is only half detected. Its so close to working perfectly, but i don't know what to do!

-em

Adrian Rosebrock June 23, 2016 at 1:24 pm #

REPLY

If you have open contours, I would suggest using morphological operations to close the gap. A dilation or a closing operation should work well in that case.

John June 22, 2016 at 2:49 pm #

REPLY

Hi Adrian,

I love your blog!

I am having trouble with the image resolution. I want the output of the image resolution to be similar to the image i am inputing.

Also, some images that i input i get "NameError: name 'screenCnt' is not defined". Does that mean the program does not detect 4 edges.

Thank you,

John

Adrian Rosebrock June 23, 2016 at 1:15 pm #

REPLY

Yep, that's correct! I would insert a print statement into your code in the for loop where you loop over the contours to confirm this.

However, a better way to solve this problem would be to keep track of the ratio of the width of *original* to the *resized* image. Perform your edge detection and contour approximation on the resized image. Then, multiply your bounding box region by the ratio — this will give you the coordinates in terms of the original image.

Kalpesh kp August 3, 2016 at 10:38 am #

REPLY

Hello Adrian Rosebrock,

Your blog is superb and like to do something like your demo. I want to develop app in ios which detects objects from video and want to count total number of objects found in Video.

So i plan to divide video in multiple images and then selecting any image and try to identify objects from image. But i am not getting any help with OpenCv much, but while i am looking at your demo. It can help me.

So can you please tell me how can i use your python code in my objective C code.

Thanks in Advance.

Adrian Rosebrock August 4, 2016 at 10:14 am #

REPLY

Your project sounds neat! However, I only provide Python code on this blog post — not Objective-C.

Cristian August 9, 2016 at 4:55 pm #

REPLY

Hi, Adrian!

Can this tutorial still be implemented in an app for android current versions? If yes, how can I get in touch with somebody that does it?

Thanks

Adrian Rosebrock August 10, 2016 at 9:27 am #

REPLY

Yes, you can use this algorithm inside of an Android app, but you would need to port it to Java + OpenCV first. As for as an Android developer, I would suggest using Upwork or Freelancer to find a developer suitable to your needs and budget.

Matthew Montebello August 24, 2016 at 9:41 am #

REPLY

Had to change Line 37 to:

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

with Python3.5.2 and Opencv3

Adrian Rosebrock August 24, 2016 at 12:12 pm #

REPLY

Indeed, this blog post was written well before OpenCV 3 was released. You can read more about the changes to cv2.findContours between OpenCV 2.4 and OpenCV in this post.

John September 2, 2016 at 12:25 pm #

REPLY

How would apply OCR to the processed image?

Adrian Rosebrock September 5, 2016 at 8:10 am #

REPLY

There are many ways to apply OCR to the thresholded image, but to start, you could try Tesseract, the open source solution. I also really like the Google Vision API for OCR as well.

Tahir October 3, 2016 at 2:17 am #

REPLY

Good explanation,can you tell me where i can use this feature mean to say where to use this docment scanner

Adrian Rosebrock October 3, 2016 at 7:11 am #

REPLY

Hey Tahir — can you elaborate more on what you mean by “where to use this document scanner”? I’m not sure I understand what you mean.

Nicky Fandino October 3, 2016 at 3:15 am #

REPLY

Hi, Adrian. It’s been a while since you created this blog. I got a question for you. I want to build this kind of paper scanner myself. Do you think it’s possible to make the scanner detect some sort of QR codes at each corners of the paper ? And then use the qr codes as the corner of the digitized paper instead of using edge detection like yours. I could use some help. Thanks before, btw

Adrian Rosebrock October 3, 2016 at 7:11 am #

REPLY

Absolutely — as long as you can detect the four corners and recognize them pr

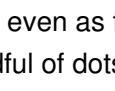
code detection blog posts, but you might want to take a look at the ZBar library. Once yo

post, and then apply the transformation. I would suggest starting with an easier marker than QR codes just to understand the general process.

Nicky Fandino October 3, 2016 at 10:15 pm #

REPLY

Thanks for replying. I'm quite new to image processing, so maybe I need to ask a few questions. I know how to detect a certain shape or a square, but I never try to detect 4 squares. What's the easiest way to do this ? Can you send me a link to your blog that explain this or some other blog maybe ? Also, how can I order them and then apply the transformation ? Is that what line 41 is ?



Adrian Rosebrock October 4, 2016 at 6:54 am #

REPLY

If you're trying to understand how to order coordinates, start here.

From there, read this post on applying a perspective transform.

As for detecting squares, the simplest method is to use contour approximation and examine the number of vertices as this blog post does. I also have an entire blog post dedicated to finding shapes in image.

I hope that helps!

Nicky Fandino October 4, 2016 at 9:18 pm #

Thanks, this really helps !

Oleg Kersh October 10, 2016 at 3:54 pm #

REPLY

Hi Adrian,
Thank you for the very cool article. I am actually trying to port your code to android (using opencv 3.1 and the android bindings) but I have got stuck at tje step of applying the Canny filter.
Although I am using the very same parameters as you are (and also downscale images to 500 rows) the edge detector does not seem to detect horizontal edges of the paper even though there is good contrast and the background is not busy.
It is strange, because vertical and angled edges are picked up nicely.

I have went even as far as lowering the tresholds to 10 and 20 and while it produces tons of false edges (as expected) it does not produce more than a handful of dots from the horizontal or near-horizontal edges.

I suppose i am missing something trivial. I have even tried the opencv android sample app and its canny does pick up edges nicely.

Your help would be really appreciated.

Adrian Rosebrock October 11, 2016 at 12:56 pm #

REPLY

That is quite strange, although I must admit that I do not have any experience working with the Java + OpenCV bindings outside of testing them out by writing a program to load and display an image to my screen. This does seem like a Java specific issue so I would suggest posting on the OpenCV forums.

Hariyama October 14, 2016 at 8:46 pm #

REPLY

Hi, I am a beginner on Python.

I have a question on Line 38:
cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5].

How does the description "[:5]" work?
I understand "sorted" function, but I cannot understand it.

Sorry for bothering you.

Adrian Rosebrock October 15, 2016 at 9:53 am #

REPLY

The [:5] is just an array slice. It simply takes the first 5 elements in the list and c

slicing here.

fabio October 18, 2016 at 9:40 am #

REPLY

Hi

Thanks a lot for the very informative post. Could you elaborate a bit on why you resize the image before the edge detection and why exactly to a height of 500 pixels? Because I tried your technique on a couple of images with and without resizing to 500 pixels. It worked perfectly for the resized images but for the original ones (they were bigger) the edge and contour detection failed horribly. I would probably just need to tune the parameters a bit differently?

Thanks!

Adrian Rosebrock October 20, 2016 at 8:54 am #

REPLY

In computer vision and image processing we rarely process images larger than 600 pixels along the maximum dimension. While high resolution images are appealing to the human eye they are simply too much detail for image processing algorithms. You can actually think of resizing an image as a form of “noise removal”. We discard much of the detail so we can focus our attention on a smaller version of the image that still contains the same contents — but with less “noise”, leading to much better results.

Harry October 27, 2016 at 2:20 pm #

REPLY

Getting error : No module named pyimagesearch.transform.

Please help/

Adrian Rosebrock November 1, 2016 at 9:24 am #

REPLY

You need to download the source code to this blog post using the “Downloads” section of this tutorial. You are likely forgetting to create the pyimagesearch directory and put a __init__.py file inside of it.

jagdish November 7, 2016 at 7:45 am #

REPLY

hey there
which all libraries are you using ?
which version
thanks man

Adrian Rosebrock November 7, 2016 at 2:44 pm #

REPLY

This blog post assumes you are using OpenCV 2.4 and Python 2.7. The code can be easily adapted to work with OpenCV 3 and Python 3 as well.

sushma November 23, 2016 at 11:26 pm #

REPLY

Hi, Can i get same concept (Mobile Document Scanner) on Android…

Adrian Rosebrock November 24, 2016 at 9:36 am #

REPLY

I only cover Python + OpenCV on this blog. You would need to convert it to Java + OpenCV for Android.

Michael December 1, 2016 at 6:18 am #

REPLY

Hi,
Thanks for the awesome and comprehensive tutorial!

I actually want to apply this to some photo of receipts, but unfortunately not all the corner is inside the image (there is even a photo where not even one of the corner is on the image).

Is there a way to use the four_point_transform in this case? If yes, how to do it and if not is there any good way to deskew the image?

Thanks!

Adrian Rosebrock December 1, 2016 at 7:20 am #

REPLY

If you lack the corners you can apply the perspective transform to the entire image, although I really don't recommend this. Otherwise, you can try to deskew the image. I've been meaning to do a tutorial on this, but the gist is that you threshold the image to reveal the text, then computing a rotated bounding box around a text region. The rotated bounding box will give you the angle that you can correct for. Again, I'll try to do a tutorial on this in the future.

Neil December 8, 2016 at 7:05 pm #

REPLY

Hi Adrian

I'm struggling to understand how exactly the number of vertices are approximated in lines 43 and 44. Would you mind explaining this?

Many thanks

Adrian Rosebrock December 10, 2016 at 7:19 am #

REPLY

First, we compute the perimeter of the contour. We then take the perimeter and multiply it by a small percentage. The exact value of the percentage may take some fiddling based on your dataset, but typically values between 0.01-0.05 are common. This percentage controls the actual approximation according to the Ramer-Douglas-Peucker algorithm. The larger epsilon is, the less points included in the actual approximation.

Umair December 21, 2016 at 1:36 pm #

REPLY

Hi,

I am working on a similar project and your tutorials are of great help.

My goal was to detect the total price mentioned in a receipt. How can we achieve that goal so that i can easily detect the price? Any input will be much appreciated.

Adrian Rosebrock December 23, 2016 at 10:58 am #

REPLY

It sounds like you are trying to apply Optical Character Recognition (OCR) which is the process of recognizing text in images. This is a very extensive (and challenging field). To start, I would suggest trying to localize where in the image the total price would be (likely towards the bottom of the receipt). From there you can apply an OCR engine like Tesseract or the Google Vision API.

Arturo December 30, 2016 at 5:25 pm #

REPLY

Thanks Adrian!! The practical uses for Computer Vision techniques are amazing. I like them. Do you have a post or any suggestion on how to load the python code on android mobile cell phones? regards!

Adrian Rosebrock December 31, 2016 at 1:17 pm #

REPLY

Unfortunately, I don't know of a way to use OpenCV + Python on a mobile devices. I would instead suggest creating a API endpoint that can process images and have the mobile app call the API.

Eric January 4, 2017 at 9:53 pm #

REPLY

Hi, it's a great tutorial. May I ask instead of using skiimage adaptive thresholding, is

such as

cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,11,2)

as there is a problem importing skiimage python package to AWS Lambda. If it is possible, it would be great if the parameter in cv2 function could be provided to best suit this mobile scanner application . As I am new to opencv and skiimage. Any suggestions would be appreciated. Thanks in advance.

Adrian Rosebrock January 7, 2017 at 9:42 am #

REPLY

You can certainly use OpenCV's adaptive thresholding algorithm, that's no problem at all.

Jonathan January 9, 2017 at 10:48 am #

REPLY

Hi there, thanks for the great piece of work. I am having trouble to reliably find the countour. Would there be a way to have cv to default to image boundaries or may be bounding rectangle if the countours could not be approximated? Thanks again so much and happy new year

Adrian Rosebrock January 10, 2017 at 1:11 pm #

REPLY

If you are having trouble finding the contour of the document then I would suggest playing with the edge detection parameters. If the script cannot find this contour then there isn't really a way to "default" any coordinates unless you were working in a fixed, controlled document where you know *exactly* where the user was placing the document to be scanned.

Alexander Chebykin January 15, 2017 at 9:35 am #

REPLY

You could just check if value in screenCnt is None, and in case it is, default to the whole image. Also I found that blurring the edged image before approximation makes approximation better.

Alexander Chebykin January 15, 2017 at 9:21 am #

REPLY

Hi. Thanks for the article – it was really helpful.
I went further by adding OCR, and optimizing the code for that matter.
My code and thoughts on the subject can be found here: https://awesomelemon.github.io/2017/01/15/Document-recognition-with-Python-OpenCV-and-Tesseract.html
I hope it will be useful for people who want to make the next step.

Adrian Rosebrock January 15, 2017 at 11:59 am #

REPLY

Thanks for sharing Alexander!

Murthy January 30, 2017 at 11:44 pm #

REPLY

Hi Adrian,
Thanks for this. This was a good beginning to learn OpenCV.
I struggled to install opencv on Mac but then was successful in doing so on a linux box.

I used Python 2.7 and opencv 3.1.0

Line 37 : (cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

in this example gave me error (ValueError: too many values to unpack)

I changed left hand side of line 37 to:

(_, cnts, _) = cv2.findContours(edged.copy(), cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)

and it worked!

Sergi February 8, 2017 at 2:59 pm #

REPLY

Hi Adrian, I followed your tutorial on installing openCV on RPi3, so there it is working into a virtual environment.
Now I find that I can use scipy, skimage and I also tried with sklearn, only outside the virtual environment, inside the v.environment those packages are not found. I have done many times to “pip uninstall” those packages and install them again from the virtual environment, but nothing changes.
Maybe you have some tips since I am completely lost.

Adrian Rosebrock February 10, 2017 at 2:08 pm #

REPLY

Hey Sergi — it sounds like you have installed SciPy, scikit-image, etc. into the global site-packages directory for your system. You mentioned installing them into your virtual environment, but I don’t think the commands worked:

```
$ workon cv # important! access your virtual environment first
$ pip intall numpy
$ pip install scipy
...
```

qkzk March 22, 2017 at 3:03 pm #

REPLY

Hello ! Thank you for the wonderfull work.

I followed your raspberry pi installation guide and managed to install CV 3 on my rpi3 (raspbian jessie with pixel). I went here and downloaded your code.

I’m not able to install scikit-image package in the virtual environment.
I did :

workon cv
pip install -U scikit-image (with sudo too)

first try : failed, a lot of garbage from python but only “memory error”.
second try : No eggs found in /tmp/easy_install-V0e5rR/Cython-0.25.2/egg-dist-tmp-dt9B8k (setup script problem?)

But it keeps failing. I managed to install scikit-image out of the vitual environment with apt-get but, obviously, it doesn’t work in cv…

pip is working in the cv : workon cv, pip install scipy
python
>>> import scipy
doesn’t return an error. It just took 3 hours ☐

—
I’ve read crazy solutions like apt-get download the packages and install them manually but I’d like to find a cleaner workaround.

Any idea ?

Adrian Rosebrock March 23, 2017 at 9:31 am #

REPLY

If you are getting a memory issue, try using the --no-cache-dir when installing via pip:

$ pip install scikit-image --no-cache-dir

This should alleviate the issue.

Vineet March 24, 2017 at 8:18 am #

REPLY

Hello Adrian, can you tell me how to get the transform module? Also, how to set up the module so that I can use it for other programs as well?

Adrian Rosebrock March 25, 2017 at 9:25 am #

REPLY

Use the “Downloads” section of this blog post to download the example code (including the “pyimagesearch” and “transform” mini-modules for the example).

Guy March 25, 2017 at 11:54 am #

REPLY

Hi!
I don't know if you're still active
but anyway
just wanted to thank you
me and my friend are doing a project
and you're code really saves us
without you we were lost

so – making long story short –
THANK YOU VERY VERY MUCH
YOU'RE OUR SAVIOR!!!

Adrian Rosebrock March 28, 2017 at 1:11 pm #

REPLY

Thank you Guy, I'm happy the tutorial helped you out!

Karl Dreyer March 31, 2017 at 9:53 am #

REPLY

Hello, are there any guides or examples how to use OpenCV in a Xamarin Android environment? I'm working on an Android app and need to find a good alternative from Scanbot and other expensive solutions.

Adrian Rosebrock March 31, 2017 at 1:46 pm #

REPLY

Hey Karl — I don't have any experience with OpenCV + Android environments so I'm unfortunately not the right person to ask regarding this question.

Suraj April 9, 2017 at 6:09 pm #

REPLY

Hello Adrian, i just came across this post its very helpful please do you have anything on OCR?
or is it possible to modified this code for this purpose?

Adrian Rosebrock April 12, 2017 at 1:21 pm #

REPLY

I don't have any posts on OCR right, but I will be covering OCR in the future.

Vikram April 14, 2017 at 2:47 pm #

REPLY

Can you suggest some tutorial or any source to make an android or IOS app for mobile scanner and integrate with the mobile camera and convert the clicked image into PDF or jpeg

---

Michiel April 18, 2017 at 5:45 am #

REPLY

Hi Adrian,

Many thanks for uploading your tutorials! I have a problem with installing the pyimagesearch module though. I am using the jupyter notebook, usually with installing packages I use the following code:

!pip install [PACKAGE NAME]

However, this time this gives te following error: Could not find a version that satisfies the requirement pyimagesearch (from versions:)
No matching distribution found for pyimagesearch

Do you have any idea what might cause the problem?

Adrian Rosebrock April 19, 2017 at 12:49 pm #

REPLY

There is no "pyimagesearch" module on PyPI. You need to use the "Downloads

example images, then place the pyimagesearch directory in the same path as your Jupyter Notebook.

Ian May 5, 2017 at 3:14 am #

REPLY

Just wondering … where is the best place to find Android/iOS developer?

vasanth May 7, 2017 at 4:38 am #

REPLY

error: argument -i/–image is required while i run the program

Adrian Rosebrock May 8, 2017 at 12:26 pm #

REPLY

You need to read up on command line arguments.

vasanth May 8, 2017 at 1:08 pm #

REPLY

i’m using windows

vasanth May 8, 2017 at 2:41 pm #

REPLY

sir how to save the scanned image sir

Adrian Rosebrock May 11, 2017 at 9:02 am #

REPLY

I would suggest you use cv2.imwrite to write the image to disk. If you’re just getting started learning the basics of OpenCV and Python, I would absolutely suggest you work through practical Python and OpenCV. This book will help you learn the fundamentals of OpenCV and image processing.

Onkar Pandit May 10, 2017 at 1:08 pm #

REPLY

Hey, great tutorial.Very informative and easy to understand.
However I am unable to figure out how to use it in android studio to build the app.
Please Help

Adrian Rosebrock May 11, 2017 at 8:45 am #

REPLY

Hi Onkar — this blog uses primarily OpenCV and Python. It does not cover Java. I would suggest you read up on Java + OpenCV bindings and convert the code. Alternatively you can build a REST application where the Java app sends the image to be processed on a Python server and the results returned.

Francois May 30, 2017 at 5:41 pm #

REPLY

The imutils module is not included in the code download for Basic Image Manipulations, so I can’t progress part the first part of this tutorial.

Adrian Rosebrock May 31, 2017 at 1:07 pm #

REPLY

Each blog post is independent from the others. You should either use the “Downloads” section of the blog post to download the code or install imutils via pip:

$ pip install imutils

shravankumar June 12, 2017 at 12:25 pm #

REPLY

Hey chief,

I have successfully tried the code on my machine with the provided images, but when I tried with images downloaded from web, unable to find contours around the sheet. What could be done to optimize the code for any image.

Any help would greatly appreciated

Thank you

Adrian Rosebrock June 12, 2017 at 1:03 pm #

REPLY

If there is not enough contrast between the edges of the paper and the background then the paper may not be detected. Furthermore, if paper is noisy/folded/etc. the contour approximation won’t be able to locate the paper region (since it might have more than four vertices). In that case, you would want to train a custom object detector. I’ve also seen work done on using machine learning *specifically* to find edge-regions (even low contrast ones) in documents. I can’t seem to find the paper though. If I do, I’ll be sure to link you to it.

Trackbacks/Pingbacks

Find distance from camera to object using Python and OpenCV - January 19, 2015
[…] More on this methodology can be found in this post on building a kick-ass mobile document […]
Zero-parameter, automatic Canny edge detection with Python and OpenCV - PyImageSearch - April 6, 2015
[…] we’ve used the Canny edge detector a fair amount of 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 […]
Sorting Contours using Python and OpenCV - PyImageSearch - April 20, 2015
[…] We used contours to build a kick-ass mobile document scanner. […]
Target acquired: Finding targets in drone and quadcopter video streams using Python and OpenCV - PyImageSearch - May 4, 2015
[…] way to detect square and rectangular objects in an 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 […]
Accessing the Raspberry Pi Camera with OpenCV and Python - PyImageSearch - May 29, 2015
[…] the dominant colors in an image was (and still is) hugely popular. One of my personal favorites, building a kick-ass mobile document scanner has been the most popular PyImageSearch article for months. And the first (big) tutorial I ever […]
Ordering coordinates clockwise with Python and OpenCV - PyImageSearch - March 21, 2016
[…] little over a year ago, I wrote one my favorite tutorials on the PyImageSearch blog: How to build a kick-ass mobile document scanner in just 5 minutes. Even though this tutorial is over a year old, its still one of the most popular blog posts on […]
Bubble sheet multiple choice scanner and test grader using OMR, Python and OpenCV - PyImageSearch - October 3, 2016
[…] special is that we are going to combine the techniques from many previous blog posts, including building a document scanner, contour sorting, and perspective transforms. Using the knowledge gained from these previous […]

Leave a Reply

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.

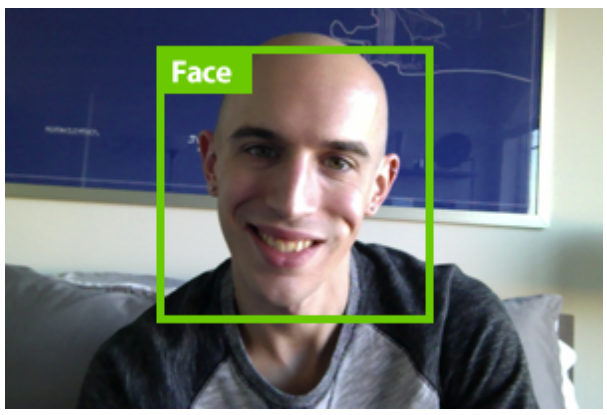
Download for Free!

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.

CLICK HERE TO MASTER FACE DETECTION

PyImageSearch Gurus: NOW ENROLLING!

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

Search

Search...

Find me on **Twitter**, **Facebook**, **Google+**, and **LinkedIn**.

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