

[Open in app](#)

Get unlimited access to all of Medium for less than \$1/week. [Become a member](#)



How to automatically deskew (straighten) a text image using OpenCV



Leo Ertuna · [Follow](#)

Published in [Becoming Human: Artificial Intelligence Magazine](#)

6 min read · Sep 5, 2020

[Listen](#)[Share](#)[More](#)

Leo Ertuna

Full-stack Developer, AI Researcher, Crypto Enthusiast, Petrohead

Today I would like to share with you how to automatically deskew (straightening a rotated image). If you have ever tried to extract text from images — you will know that it's not always a simple task. From camera pictures to scanned documents, there are many cases where the text is rotated. One common step in image pre-processing before feeding the cleaned-up image to an OCR tool.

ng problem
ext
one form or
mandatory

74 Followers

[Follow](#)

As I myself was learning and experimenting with image processing in [OpenCV](#), I found that in the majority of tutorials you just get a copy-pasted code solution, with barely any explanation of the logic behind it. That's just not right. We need to understand the algorithms and how we can combine various image transformations to solve a given problem. Otherwise we won't make any progress as software engineers. So in this tutorial I will try to keep the [code](#) snippets to bare minimum, and concentrate on explaining the ideas that make it work. But don't worry, you can always find the complete code in my GitHub repo by the link at the end of this article.

Deskewing algorithm

Let's start by discussing the general idea of deskewing algorithm. Our main goal will be splitting the rotated image into text blocks, and determining the angle from

them. To give you a detailed break-down of the approach that I'll use:

1. Per usual – convert the image to gray scale.
2. Apply slight blurring to decrease noise in the image.
3. Now our goal is to find areas with text, i.e. text blocks of the image. To make text block detection easier we will invert and maximize the colors of our image, that will be achieved via thresholding. So now text becomes white (exactly 255,255,255 white), and background is black (same deal 0,0,0 black).
4. To find text blocks we need to merge all printed characters of the block. We achieve this via dilation (expansion of white pixels). With a larger kernel on X axis to get rid of all spaces between words, and a smaller kernel on Y axis to blend in lines of one block between each other, but keep larger spaces between text blocks intact.
5. Now a simple contour detection with min area rectangle enclosing our contour will form all the text blocks that we need.
6. There can be various approaches to determine skew angle, but we'll stick to the simple one – take the largest text block and use its angle.

And now switching to python code:

```
# Calculate skew angle of an image
def getSkewAngle(cvImage) -> float:
    # Prep image, copy, convert to gray scale, blur, and threshold
    newImage = cvImage.copy()
    gray = cv2.cvtColor(newImage, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (9, 9), 0)
    thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV +
    cv2.THRESH_OTSU)[1]

    # Apply dilate to merge text into meaningful lines/paragraphs.
    # Use larger kernel on X axis to merge characters into single
    line, cancelling out any spaces.
    # But use smaller kernel on Y axis to separate between different
    blocks of text
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (30, 5))
    dilate = cv2.dilate(thresh, kernel, iterations=5)

    # Find all contours
    contours, hierarchy = cv2.findContours(dilate, cv2.RETR_LIST,
    cv2.CHAIN_APPROX_SIMPLE)
    contours = sorted(contours, key = cv2.contourArea, reverse =
```

True)

```
# Find largest contour and surround in min area box
largestContour = contours[0]
minAreaRect = cv2.minAreaRect(largestContour)

# Determine the angle. Convert it to the value that was
# originally used to obtain skewed image
angle = minAreaRect[-1]
if angle < -45:
    angle = 90 + angle
return -1.0 * angle
```

Trending AI Articles:

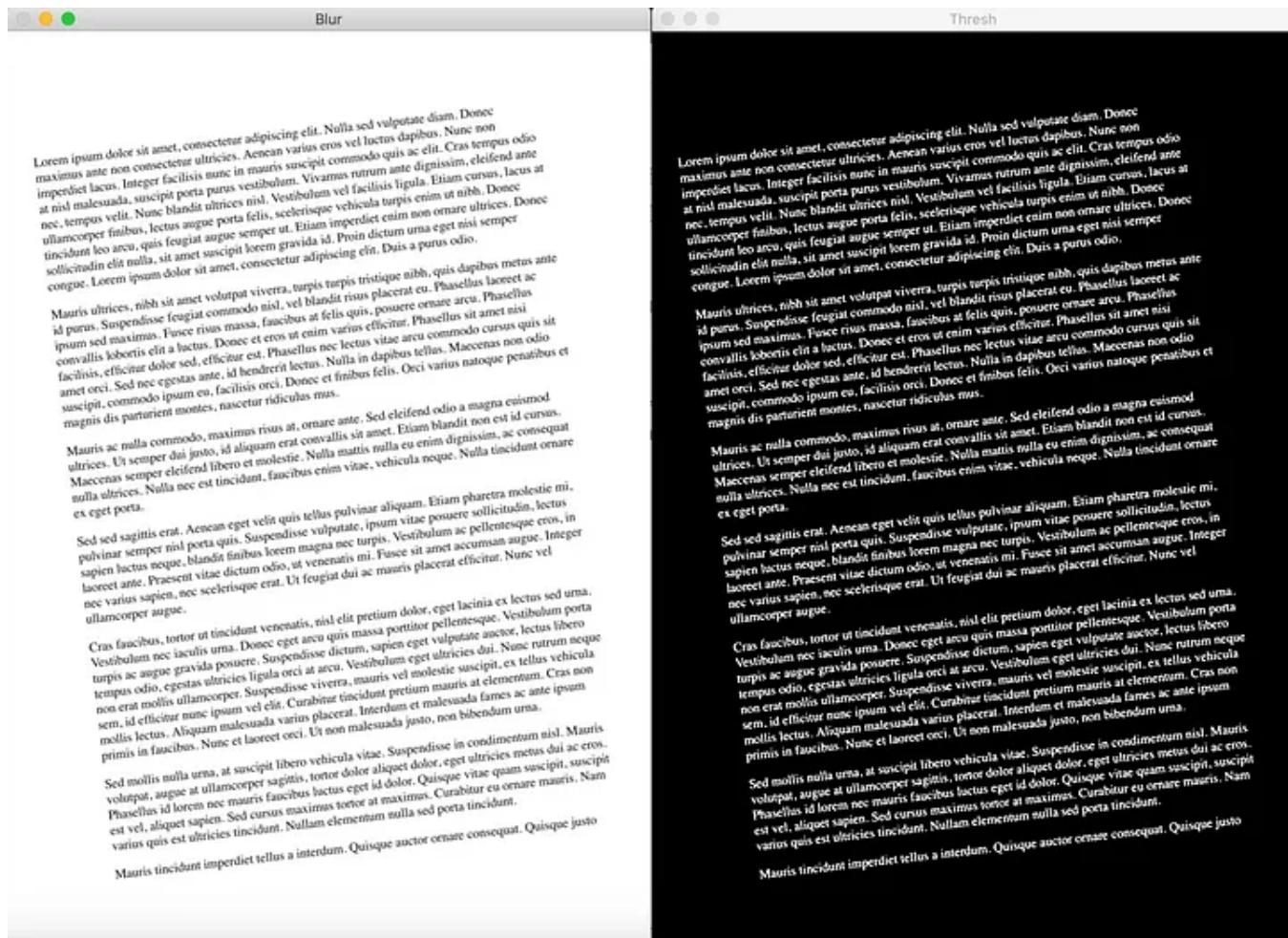
1. Microsoft Azure Machine Learning x Udacity — Lesson 4 Notes
2. Fundamentals of AI, ML and Deep Learning for Product Managers
3. Roadmap to Data Science
4. Work on Artificial Intelligence Projects

After the skew angle is obtained we just need to re-rotate our image:

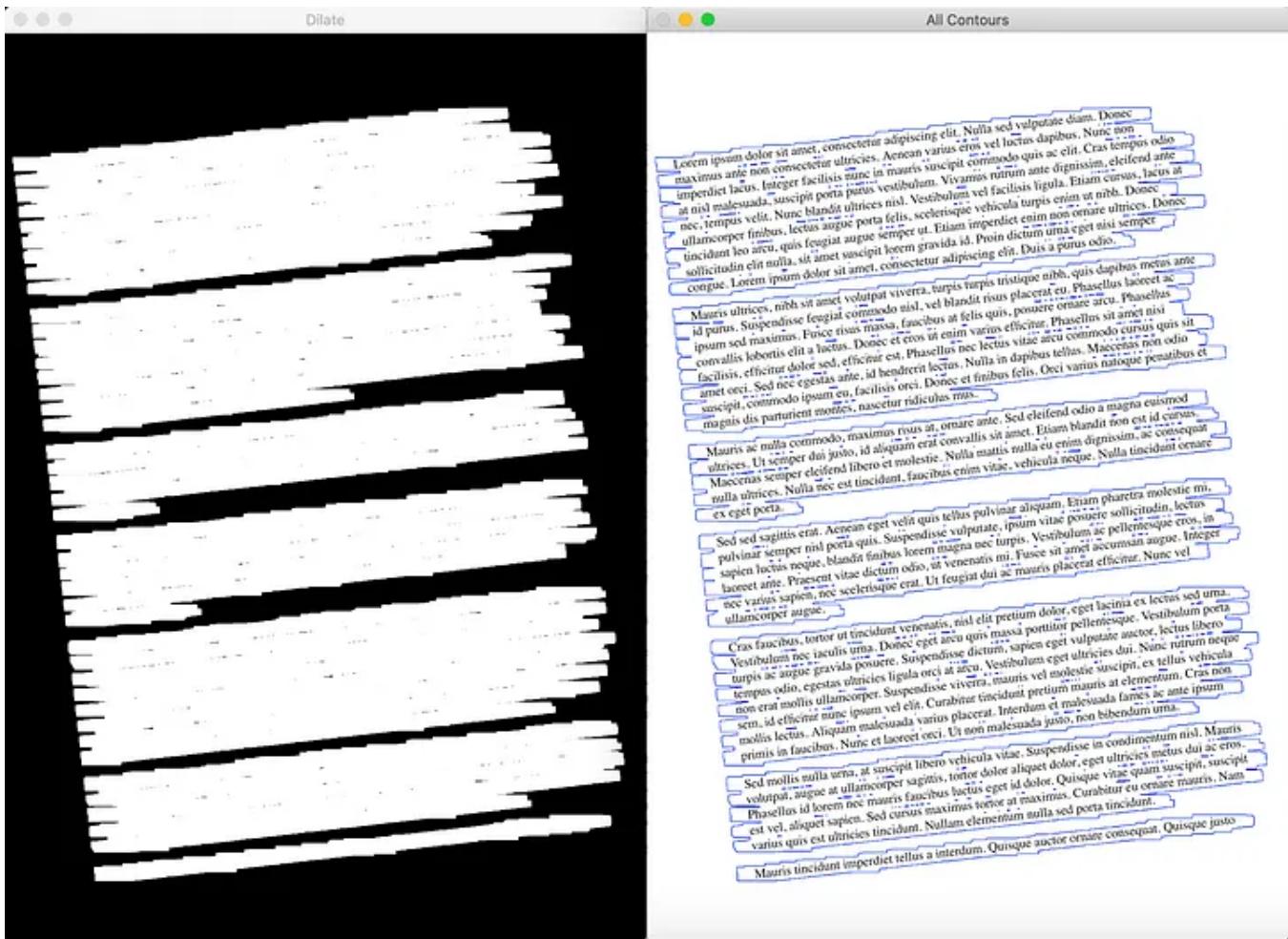
```
# Rotate the image around its center
def rotateImage(cvImage, angle: float):
    newImage = cvImage.copy()
    (h, w) = newImage.shape[:2]
    center = (w // 2, h // 2)
    M = cv2.getRotationMatrix2D(center, angle, 1.0)
    newImage = cv2.warpAffine(newImage, M, (w, h),
    flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE)
    return newImage

# Deskew image
def deskew(cvImage):
    angle = getSkewAngle(cvImage)
    return rotateImage(cvImage, -1.0 * angle)
```

Visualizing the steps



Blur and threshold applied to the image



Dilation and contour detection of text blocks

Largest Contour

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla sed vulputate diam. Donec maximus ante non consectetur ultricies. Aenean varius eros vel luctus dapibus. Nunc non imperdiet lacus. Integer facilisis nunc in mauris suscipit commodo quis ac elit. Cras tempus odio at nisl malesuada, suscipit porta purus vestibulum. Vivamus rutrum ante dignissim, eleifend ante nec, tempus velit. Nunc blandit ultrices nisl. Vestibulum vel facilisis ligula. Etiam cursus, lacus at ullamcorper finibus, lectus augue porta felis, scelerisque vehicula turpis enim ut nibh. Donec tincidunt leo arcu, quis feugiat augue semper ut. Etiam imperdiet enim non ornare ultrices. Donec sollicitudin elit nulla, sit amet suscipit lorem gravida id. Proin dictum urna eget nisi semper congue. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis a purus odio.

Mauris ultrices, nibh sit amet volutpat viverra, turpis turpis tristique nibh, quis dapibus metus ante id purus. Suspendisse feugiat commodo nisl, vel blandit risus placerat eu. Phasellus laoreet ac ipsum sed maximus. Fusce risus massa, faucibus at felis quis, posuere ornare arcu. Phasellus convallis lobortis elit a luctus. Donec et eros ut enim varius efficitur. Phasellus sit amet nisi facilisis, efficitur dolor sed, efficitur est. Phasellus nec lectus vitae arcu commodo cursus quis sit amet orci. Sed nec egestas ante, id hendrerit lectus. Nulla in dapibus tellus. Maecenas non odio suscipit, commodo ipsum eu, facilisis orci. Donec et finibus felis. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.

Mauris ac nulla commodo, maximus risus at, ornare ante. Sed eleifend odio a magna euismod ultrices. Ut semper dui justo, id aliquam erat convallis sit amet. Etiam blandit non est id cursus. Maecenas semper eleifend libero et molestie. Nulla mattis nulla eu enim dignissim, ac consequat nulla ultrices. Nulla nec est tincidunt, faucibus enim vitae, vehicula neque. Nulla tincidunt ornare ex eget porta.

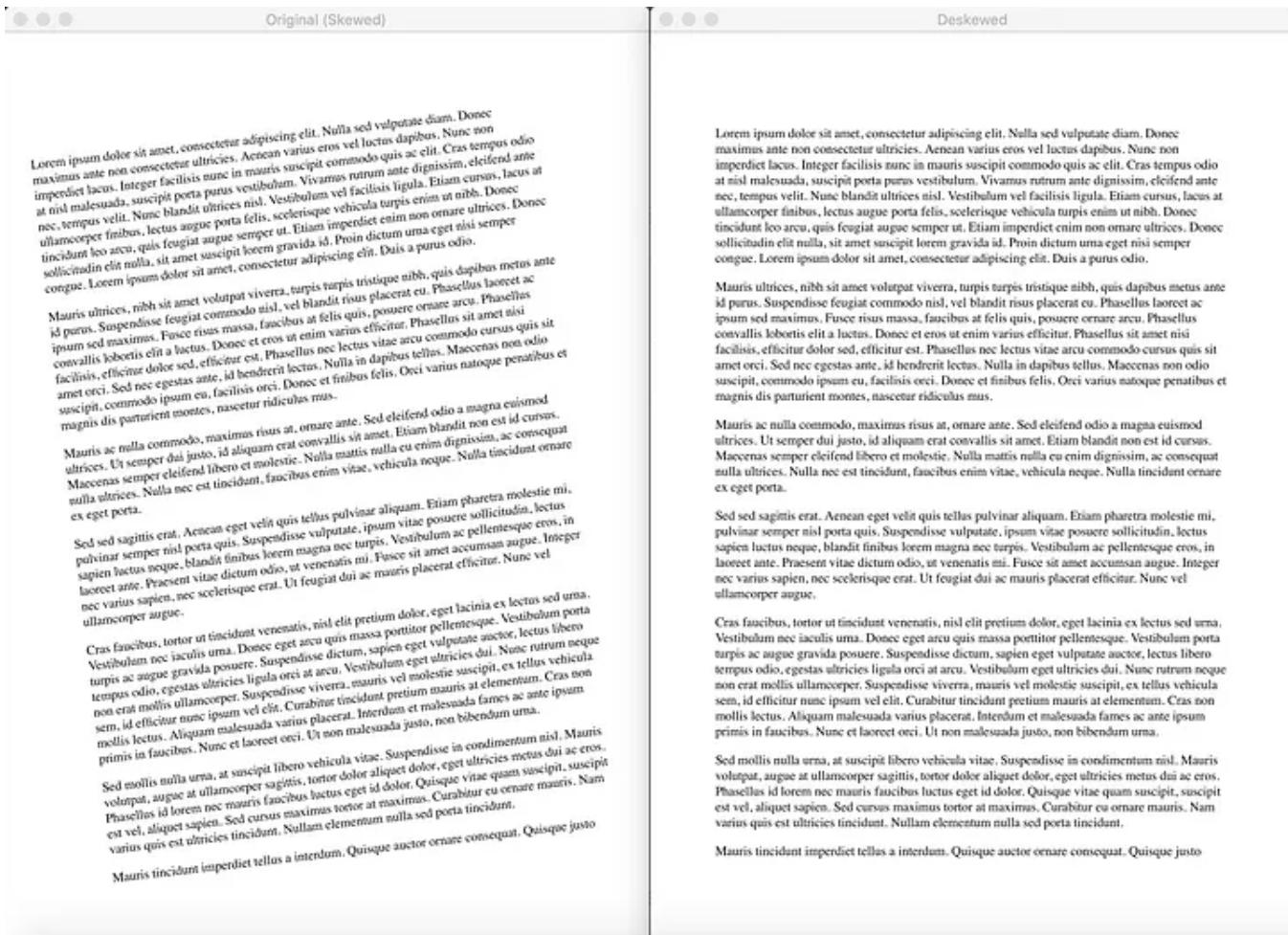
Sed sed sagittis erat. Aenean eget velit quis tellus pulvinar aliquam. Etiam pharetra molestie mi, pulvinar semper nisl porta quis. Suspendisse vulputate, ipsum vitae posuere sollicitudin, lectus sapien luctus neque, blandit finibus lorem magna nec turpis. Vestibulum ac pellentesque eros, in laoreet ante. Praesent vitae dictum odio, ut venenatis mi. Fusce sit amet accumsan augue. Integer nec varius sapien, nec scelerisque erat. Ut feugiat dui ac mauris placerat efficitur. Nunc vel ullamcorper augue.

Cras faucibus, tortor ut tincidunt venenatis, nisl elit pretium dolor, eget lacinia ex lectus sed urna. Vestibulum nec iaculis urna. Donec eget arcu quis massa porttitor pellentesque. Vestibulum porta turpis ac augue gravida posuere. Suspendisse dictum, sapien eget vulputate auctor, lectus libero tempus odio, egestas ultricies ligula orci at arcu. Vestibulum eget ultricies dui. Nunc rutrum neque non erat mollis ullamcorper. Suspendisse viverra, mauris vel molestie suscipit, ex tellus vehicula sem, id efficitur nunc ipsum vel elit. Curabitur tincidunt pretium mauris at elementum. Cras non mollis lectus. Aliquam malesuada varius placerat. Interdum et malesuada fames ac ante ipsum primis in faucibus. Nunc et laoreet orci. Ut non malesuada justo, non bibendum urna.

Sed mollis nulla urna, at suscipit libero vehicula vitae. Suspendisse in condimentum nisl. Mauris volutpat, augue at ullamcorper sagittis, tortor dolor aliquet dolor, eget ultricies metus dui ac eros. Phasellus id lorem nec mauris faucibus luctus eget id dolor. Quisque vitae quam suscipit, suscipit est vel, aliquet sapien. Sed cursus maximus tortor at maximus. Curabitur eu ornare mauris. Nam varius quis est ultricies tincidunt. Nullam elementum nulla sed porta tincidunt.

Mauris tincidunt imperdiet tellus a interdum. Quisque auctor ornare consequat. Quisque justo

Largest text block determined, and wrapped in a min-area rectangle



Original, skewed image (on the left) compared to deskewed result (on the right)

AI Jobs Board

HOME JOBS POST JOB POST RESUME POST A JOB | LOGIN REGISTER

Discover top Jobs in AI, ML, Big Data & More

Don't search for jobs. Find the right fit instead.

Keyword: All Job Category: All Job Type: All Job Location: Search Now

Big Data Jobs

Side note on angle calculation

Your case may require more advanced calculation than just taking the largest block, and there are a few alternative strategies you can start experimenting with.

1 — You can use the average angle of all text blocks:

```
allContourAngles = [cv2.minAreaRect(c)[-1] for c in contours]
angle = sum(allContourAngles) / len(allContourAngles)
```

2 — You can take the angle of the middle block:

```
middleContour = contours[len(contours) // 2]
angle = cv2.minAreaRect(middleContour)[-1]
```

3 — You can try the average angle of largest, smallest and middle blocks.

```
largestContour = contours[0]
middleContour = contours[len(contours) // 2]
smallestContour = contours[-1]
angle = sum([cv2.minAreaRect(largestContour)[-1],
            cv2.minAreaRect(middleContour)[-1], cv2.minAreaRect(smallestContour)[-1]]) / 3
```

That's just some of the alternative ways I can instantly think of. Continue experimenting and find what works best for your case!

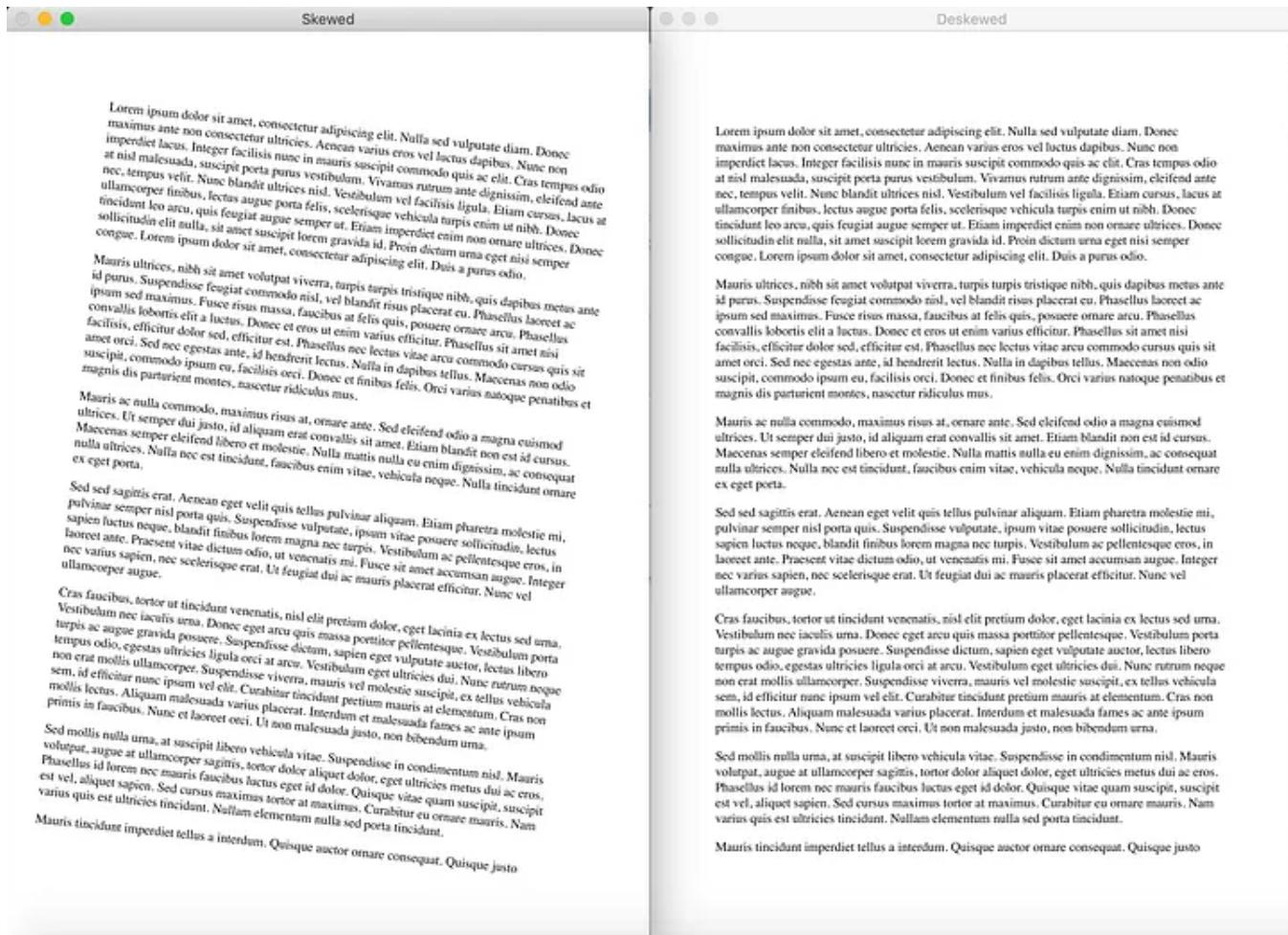
Testing

To test this approach I used a newly generated PDF file, with Lorem Ipsum text in it. The first page of this document was rendered with 300 DPI resolution (the most common setting when working with PDF documents). After that the testing dataset of 20 sample images was generated by taking the original image and randomly rotating it in the range from -10 to +10 degrees. Then I saved the images together with their skew angles. You can find all the code used to generate these sample images in my GitHub repo, I won't go over it in detail here.

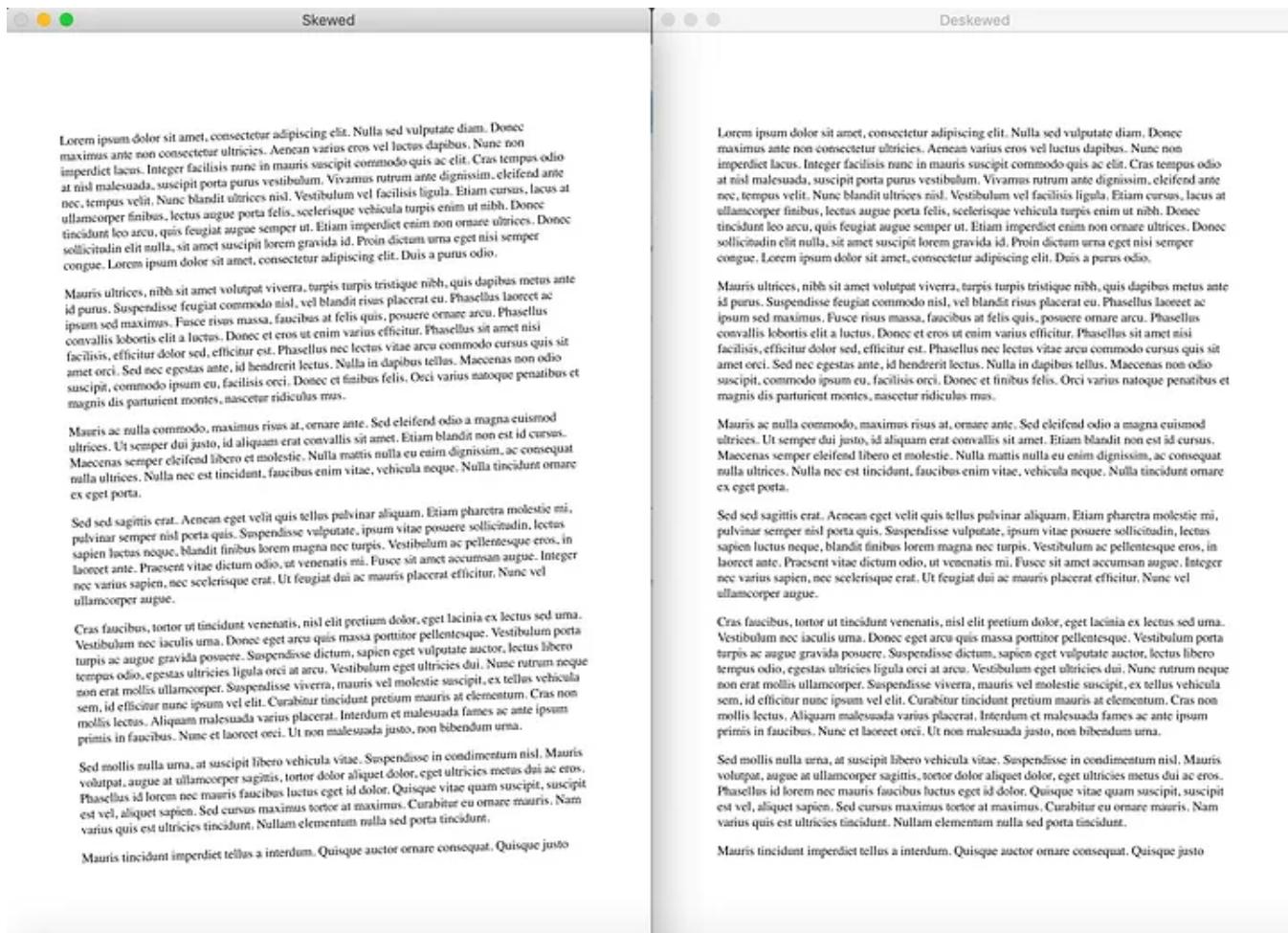
A sample statistics of testing results:

Item #0, with angle=1.77, calculated=1.77, difference=0.0%
Item #1, with angle=-1.2, calculated=-1.19, difference=0.83%
Item #2, with angle=8.92, calculated=8.92, difference=0.0%
Item #3, with angle=8.68, calculated=8.68, difference=0.0%
Item #4, with angle=4.83, calculated=4.82, difference=0.21%
Item #5, with angle=4.41, calculated=4.4, difference=0.23%
Item #6, with angle=-5.93, calculated=-5.91, difference=0.34%
Item #7, with angle=-3.32, calculated=-3.33, difference=0.3%
Item #8, with angle=6.53, calculated=6.54, difference=0.15%
Item #9, with angle=-2.66, calculated=-2.65, difference=0.38%
Item #10, with angle=-2.2, calculated=-2.19, difference=0.45%
Item #11, with angle=-1.42, calculated=-1.4, difference=1.41%
Item #12, with angle=-6.77, calculated=-6.77, difference=0.0%
Item #13, with angle=-9.26, calculated=-9.25, difference=0.11%
Item #14, with angle=4.36, calculated=4.35, difference=0.23%
Item #15, with angle=5.49, calculated=5.48, difference=0.18%
Item #16, with angle=-4.54, calculated=-4.55, difference=0.22%
Item #17, with angle=-2.54, calculated=-2.54, difference=0.0%
Item #18, with angle=4.65, calculated=4.66, difference=0.22%
Item #19, with angle=-4.33, calculated=-4.32, difference=0.23%
Min Error: 0.0%
Max Error: 1.41%
Avg Error: 0.27%

As you can see this approach works quite well, resulting in only minor digressions from the real skew angle. Such errors are no longer noticeable for the human eye and OCR engines.



Test case 1



Test case 2

That's it for today! You can apply the solution I described to most deskewing cases, especially the ones that deal with scanned document processing. But again, every problem is unique, so take this as a starting point and improve upon these basic ideas.

Thank you all for reading this tutorial, I hope you found something useful in it. Good luck out there!

GitHub repo with source code:

JPLeoRX/opencv-text-deskew

[github.com](https://github.com/JPLeoRX/opencv-text-deskew)

In case you'd like to check my other work or contact me:

- [Personal website](#)
- [GitHub](#)
- [PyPI](#)
- [DockerHub](#)
- [Blog](#)
- [LinkedIn \(feel free to connect\)](#)

Don't forget to give us your  !



0

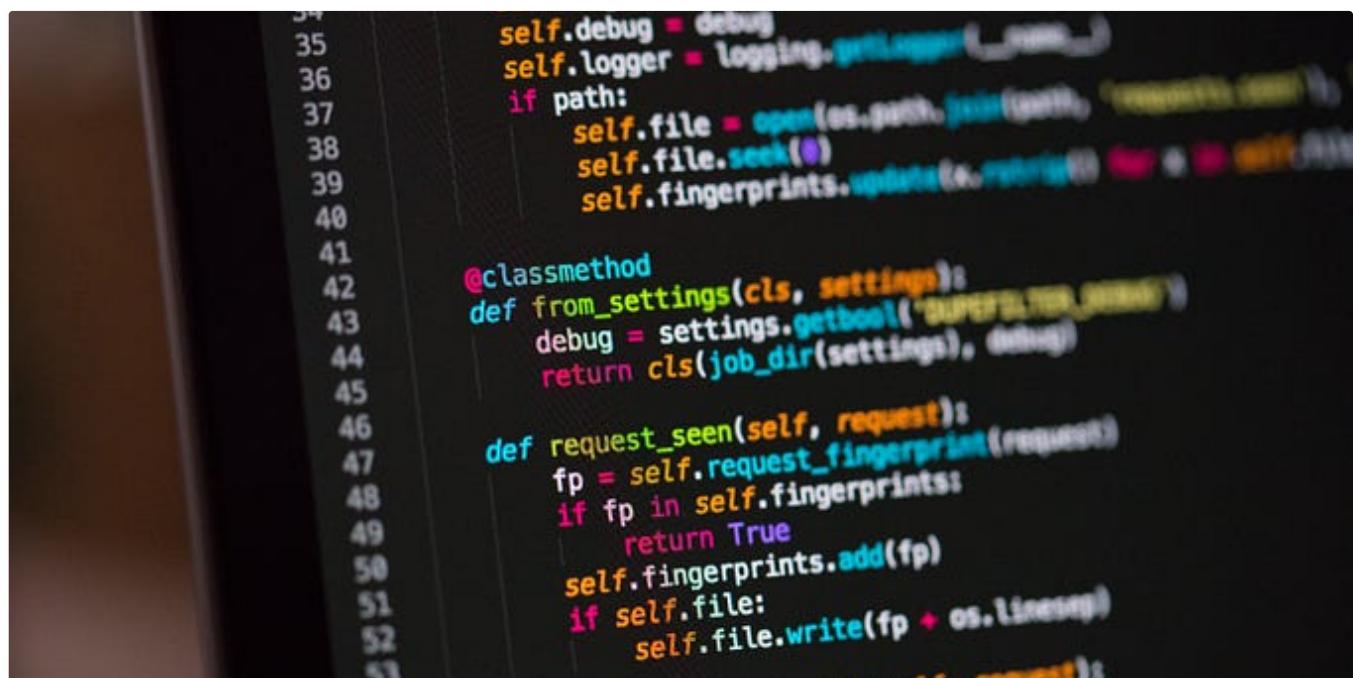
 ...[Python](#)[Opencv](#)[Image Processing](#)[AI](#)[Computer Vision](#)[Follow](#)

Written by Leo Ertuna

74 Followers · Writer for Becoming Human: Artificial Intelligence Magazine

Full-stack Developer, AI Researcher, Crypto Enthusiast, Petrolhead

Recommended from Medium



```
34
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, 'fingerprint.log'), 'a')
39         self.file.seek(0)
40         self.fingerprints = set()
41
42     @classmethod
43     def from_settings(cls, settings):
44         debug = settings.getbool('DEBUG', False)
45         return cls(job_dir(settings), debug)
46
47     def request_seen(self, request):
48         fp = self.request_fingerprint(request)
49         if fp in self.fingerprints:
50             return True
51         self.fingerprints.add(fp)
52         if self.file:
53             self.file.write(fp + os.linesep)
```



Maximilian Strauss in Better Programming

A Practical Guide To Extract Text From Images (OCR) in Python

How to use optical character recognition with three libraries

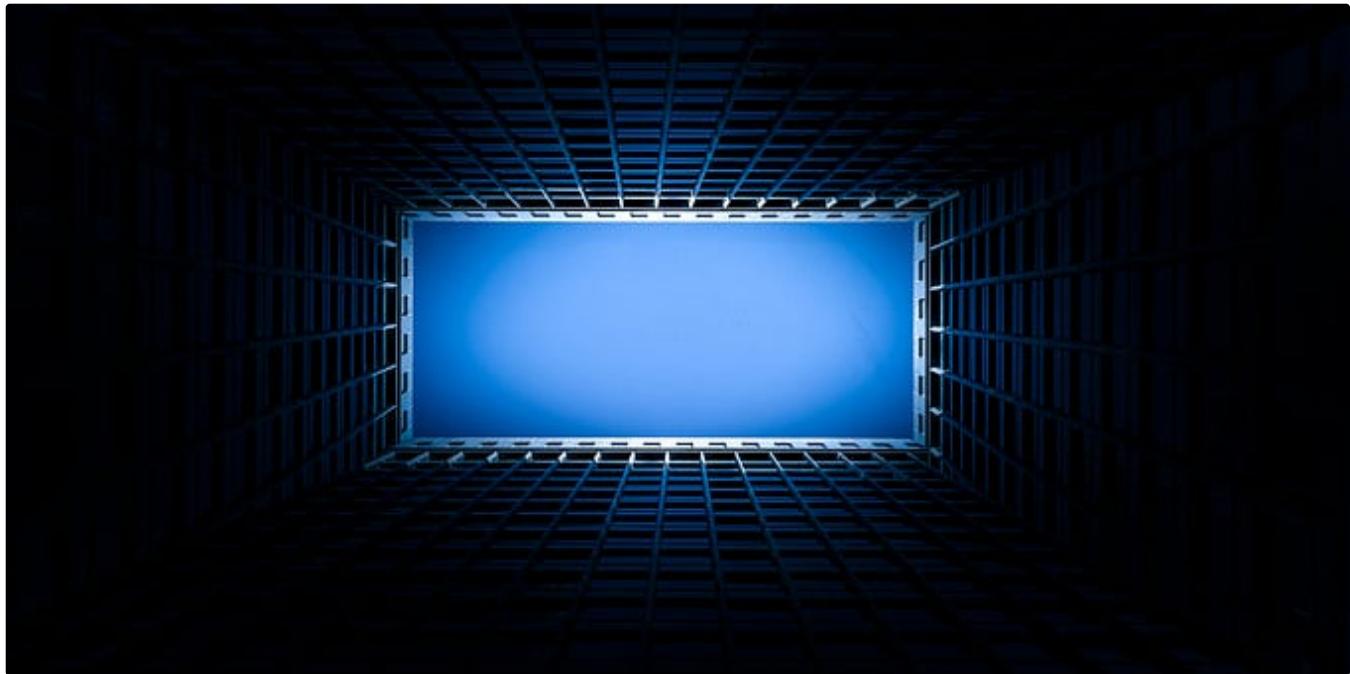
★ · 7 min read · Jan 10



3



...



 Rashida Nasrin Sucky in Towards Data Science

How to Perform Image Segmentation with Thresholding Using OpenCV

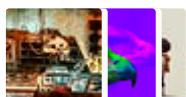
Simple, Otsu, and Adaptive Thresholding Implementation with Examples

◆ · 6 min read · Jan 12



...

Lists



What is ChatGPT?

9 stories · 112 saves



Staff Picks

354 stories · 117 saves



Visual Storytellers Playlist

27 stories · 18 saves



 Rashida Nasrin Sucky in Towards Data Science

Easy Method of Edge Detection in OpenCV Python

Using Canny Edge Detection Efficiently

◆ · 4 min read · Jan 24



...



 Evgenii Munin in Better Programming

Camera Calibration on a Chessboard With Python and OpenCV

Estimate the intrinsic camera matrix and evaluate its precision

◆ · 4 min read · Feb 22



...



 Sagar Shrestha in Level Up Coding

Learn How to Turn your Image into a Cartoon Using Python

This article will cover the various methods of turning a normal image into a cartoon version using Python. We will be using Python...

◆ · 4 min read · Feb 8



...



Dmitrii Eliuseev in Dev Genius

5 Open-Source Deep Learning Tools for Imaging Super-Resolution

Almost everybody has photos or videos made 10–20 years ago. These pictures were made in a time when 1600×1200 or even 640×480 image...

◆ · 10 min read · Oct 14, 2022



4



...

See more recommendations