**9장. 이미지 작업**

거의 매일 기본적으로 이미지로 작업한다. 페이스북 프로필 페이지에 이미지를 업로드하거나 모바일 혹은 웹 애플리케이션을 개발하는 동안 이미지를 조작한다. 풍부한 사용 사례가 있다. 컴퓨터 비전 분야에서 이렇게 많은 발전이 있었기 때문에, 이미징은 중요한 분야가 됐다. 파이썬으로 이미지를 사용하여 작업하는 것은 간단하다.

9장은 다음을 살펴볼 것이다.

-이미지 전환

-리사이징, 자르기 및 썸네일 생성

-복사-붙여넣기 및 워터마킹 이미지

-이미지 차이 및 비교

-얼굴 감지

-비즈니스 프로세스의 이미지 생성

**소개**

전자 세계의 이미지는 0과 1로 구성된 일련의 비트이다. 이미지는 장면이나 문서의 전자 스냅샷이다. 그림 혹은 사진도 디지털화하여 이미지를 만들 수 있다. 이제 이미지를 더 깊이 이해하고 이미지가 어떻게 구성되어 있는지 살펴보자.

모든 이미지는 샘플링되어 픽셀(pixels)이라는 점의 그리드로 표시된다. 이 픽셀은 화면에 표시된 그림의 최소로 관리 가능한 요소를 나타낸다. 이미지에서 사용가능한 픽셀 수가 많을수록, 장치 화면에서는 이미지가 더 정확하게 표시된다.

각 픽셀의 강도(intensity)는 가변적이다. 디지털 세계에서 이미지의 색상은 예를 들면 RGB(red, green, blue) 혹은 CMYK(Cyan, Magenta, Yellow, Black)같은 다른 색상의 세 가지 혹은 네 가지 강도로 표현된다. 컴퓨터 프로그램은 종종 RGBA 형식의 색상을 나타내며, 여기서 A는 알파(혹은 색상의 투명도)를 나타낸다. 각 픽셀은 RGBA 형식의 이진 표현으로 표현되며 시퀀스로 컴퓨터에 의해 저장된다. 그런 다음 컴퓨터는 이 시퀀스를 표시하기 위해 읽고, 경우에 따라 인쇄를 위해 아날로그 버전으로 변환한다. 특정 이미지 속성을 자세히 살펴보자.

**이미지 속성**

몇 가지 이미지 특성을 살펴보자.

-이미지 크기(Image size): 앞서 살펴봤듯이 컴퓨터 이미지는 일련의 0과 1로 저장되며 픽셀(직사각형 점) 단위로 측정된다. 이미지 파일 크기는 포함된 픽셀 수와 저장된 색 정보의 양에 따라 계산된다. 실제로 파일 크기는 컴퓨터 하드 드라이브의 이미지가 차지하는 공간이다.

-비트 심도(Bit depth): 단일 픽셀의 색상을 나타내는 데 사용되는 비트 수이다. 이 개념은 픽셀을 설명하는 데 사용되는 비트 수를 나타내는 픽셀 당 비트 수로 정의할 수 있다. 이미지의 비트 심도가 클수록 저장할 수 있는 색의 수가 증가한다. 1 비트 이미지에는 0과 1의 2개(2 ^ 1)의 색만 저장할 수 있으므로 흰색과 흑백으로 표시된다. 그것과 비교할 때, 8 비트 이미지는 256(2^8) 색상을 저장할 수 있다.

-이미지 해상도(Image resolution): 해상도는 이미지의 픽셀 수를 나타낸다. 해상도는 때때로 이미지의 폭(width)과 높이(height)로 식별된다. 예를 들어 1,920 x 1,024 픽셀 해상도의 이미지는 1,966,080 픽셀을 포함하거나 1.9메가픽셀 이미지이다.

-화질(Image quality): 이미지 저장소에 대한 정보를 기반으로 변경할 수 있다. 이미지를 나타내기 위해 모든 픽셀을 저장할 필요는 없다. 예를 들어 사진에서 파란 색 바다의 연속 블록은 픽셀을 모두 가질 필요가 없으며, 이미지를 압축하여 이미지 품질에 영향을 주지 않으면서 이미지의 디스크 공간을 줄일 수 있다. 디스크 공간의 이러한 감소를 압축(compression)이라 한다. 압축률이 높다는 것은 상세 정보의 눈에 띄는 손실을 의미한다. 오늘날의 세계에서 사용되는 일반적인 압축 유형은 JPG 압축이며, 크기 및 이미지 품질을 떨어뜨린다.

-이미지 형식(Image formats): 이미지는 다른 확장명(확장자?)으로 컴퓨터에 저장된다. BMP 혹은 TIF와 같은 형식은 전혀 압축되지 않는다. 따라서 더 많은 디스크 공간을 차지한다. JPG와 같은 파일은 압축할 수 있으며 압축 수준을 선택할 수도 있다. 따라서 TIF 이미지는 무손실이고 JPG 압축은 손실 압축(lossy compressions)이라고 한다. 손실 압축은 인간이 음영 차이를 차별화하지 못하는 것을 사용한다는 점이 흥미롭다. 다수의 손실 변환은 이미지 품질 저하를 초래하지만 여러 번의 무손실 변환은 이미지 품질을 보존한다. 하지만 일반적으로, 그것은 압축에 있어서 이미지 저하와 크기 간의 교환 조건이다.

그래, 좋은 시작이다. 9장에서는 이미지에 대한 더 많은 개념을 살펴보고 파이썬으로 이미지에서 수행할 수 있는 다양한 작업을 다룬다. 이미지를 조작하는 데 도움이 되는 여러 모듈을 살펴볼 것이다. 9장의 마지막 부분에서는 9장에서 다룬 레시피를 활용한 지식으로 자동화할 수 있는 일반적인 비즈니스 프로세스에 대해서도 다룰 것이다.

9장에서는 다음 파이썬 모듈을 사용할 것이다.

-Pillow(https://pypi.python.org/pypi/Pillow)

-scipy(https://www.scipy.org/)

-opencv(https://pypi.python.org/pypi/opencv-python)

**이미지 전환**

간단한 예제로 이미지 여행을 시작하자. 그러나 시작하기 전에 가상 환경을 구축하자.

1. virtualenv 커맨드를 사용하여 가상 환경을 구축하고 활성화한다.

chetans-MacBookPro:~ Chetan$ cd book/ch10/

chetans-MacBookPro:ch10 Chetan$ virtualenv ch10

New python executable in ch10/bin/python2.7

Also creating executable in ch10/bin/python

Installing setuptools, pip, wheel...done.

chetans-MacBookPro:ch10 Chetan$ source ch10/bin/activate

(ch10)chetans-MacBookPro:ch10 Chetan$

2. 좋아! 이제 9장의 레시피로 작업하기 위한 독립적인 환경을 구성했다. 첫 번째 예제에서는 파이썬의 Pillow 모듈을 사용할 것이다. 레시피를 진행 전에 먼저 이 모듈을 설치하자. Pillow 모듈을 설치하기 위해 가장 좋아하는 python-pip를 사용할 것이다.

(ch10)chetans-MacBookPro:ch10 Chetan$ pip install pillow

You are using pip version 7.1.0, however version 8.1.2 is

available.

You should consider upgrading via the 'pip install --upgrade

pip' command.

Collecting pillow

Downloading Pillow-3.4.2.tar.gz (10.8MB)

100% |████████████████████████████████|

10.8MB 39kB/s

Building wheels for collected packages: pillow

Running setup.py bdist\_wheel for pillow

Installing collected packages: pillow

Successfully installed pillow-3.4.2

이제 환경이 생겼고 Pillow 모듈도 설치됐다. 이제 레시피 준비 작업이 끝났다.

**실행방법**

'실행방법' 절에서는 이미지를 다른 형식으로 변환하는 방법을 다룰 것이다.

1. 우선, 모든 작업을 수행하기 위해 샘플 이미지로 사용할 수 있는 이미지를 다운로드하자. 나는 일몰을 좋아해서 9장의 대부분의 레시피에 일몰 이미지를 사용했다. 이것이 보이는 모습이다. 내 노트북에 저장하고 beach\_sunset.png라고 이름을 지었다.

<그림>

2. 이제 이 이미지를 JPEG 형식으로 변환하는 파이썬 코드를 작성하자. 다음 코드는 우리가 필요로 하는 것을 정확하게 수행한다. convert.py라는 파일로 파이썬 코드를 저장한다.

from PIL import Image

img = Image.open('beach\_sunset.png')

img.save('beach-sunset-conv.jpg','jpeg')

이 프로그램을 파이썬의 convert.py 커맨드로 실행하면 원본 PNG 이미지가 JPG 형식으로 변환되고 asbeach-susnset-conv.jpg로 저장된다.

3. 깔끔하다! 이제 이 이미지에 대해 작업을 한번 더 수행하여 그레이스케일(흑백 형식)로 변환하자. 사람들이 이미지를 흑백 형식으로 변환하여 향수 어린 느낌을 부여하는 것은 일반적이다. 이것은 다음 커맨드 라인을 사용하여 쉽게 수행할 수 있다.

from PIL import Image

img = Image.open('beach\_sunset.png').convert('L')

img.show()

img.save('beach-sunset-gray.png','png')

이제 이 프로그램을 실행하면 다음과 같은 disk-sunset-gray.png라는 이름으로 디스크에 다른 이미지가 생성될 것이다.

<사진>

4. 좋아, 더 나아가 이미지 회전 및 뒤집기와 같은 몇 가지 작업을 수행해보자. 이러한 작업은 재미있는 웹 사이트에서 자주 사용되며 이미지가 있는 곳에서 간단히 활용할 수 있다. 다음 코드는 이미지를 180도 회전시키는 데 도움이 될 것이다.

from PIL import Image

img = Image.open('sunset.jpg')

img.rotate(180).save('sunset180deg.jpg')

기본 이미지로 이 코드를 실행하면 여기에 보이는 것처럼 180도 회전된 이미지, 즉 이미지가 거꾸로 보인다.

<그림>

5. 이미지를 회전시키는 것은 좋지만, 이미지를 뒤집을 수 있다면 정말 재미있을 것이다. PIL은 실망시키지 않으며 이미지를 수평 및 수직으로 뒤집을 수 있는 옵션을 제공한다. 다음 코드는 뒤집기 작업을 수행하는 데 도움이 된다.

from PIL import Image

img = Image.open('sunset.jpg')

img.transpose(Image.FLIP\_LEFT\_RIGHT).save(

'sunset\_horizontal\_flip.png')

img.transpose(Image.FLIP\_TOP\_BOTTOM).save(

'sunset\_vertical\_flip.png')

이제 이 코드를 실행하면 두 개의 이미지가 생성될 것이다. 다음 이미지는 원본 이미지와 동일하지만 수평으로 뒤집는다(마치 이미지의 오른쪽에 거울이 있는 것처럼). 산이 이미지의 오른쪽으로 움직인 방법을 살펴본다.

<그림>

다음 스크린샷은 원본 이미지의 대칭 이미지로, 수직으로 뒤집는다. 산은 여전히 이미지의 왼쪽에 있지만 거꾸로 되어있다. 석양도 이와같다. 일출처럼 보일까?

<그림>

**작동원리**

'작동원리' 절에서는 PNG와 JPEG의 두 가지 이미지 형식을 다룬다. PNG(Portable Network Graphics) 파일은 품질을 저하시키지 않으면서 사진 이미지를 압축하는 비손실 파일이다. 인터넷 그래픽을 위한 훌륭한 파일 형식이다. 여러 배경과 함께 사용할 수 있으며 투명도를 지원한다. 첫 번째 코드 예제인 beach\_sunset.png에 사용된 이미지의 경우, 파일 크기는 550KB이다.

JPEG(Joint Photographic Experts Group)는 손실 압축 기술을 사용하여 이미지를 압축한다. JPG는 이미지 섹션을 픽셀 혹은 타일로 줄여 이미지를 압축한다. JPG 이미지는 설정에 따라 N:1의 비율로 압축될 수 있다. 이미지가 쉽게 압축되고 인터넷 상의 이미지에 액세스하기 위한 인터넷 대역폭을 줄일 수 있기 때문에 JPG는 인터넷의 이미지 표준이 됐다. 변환된 이미지는 파일 크기가 450KB로 PNG 파일보다 거의 20% 더 작다.

이제 파이썬 코드를 살펴보자. Image 클래스는 PIL 모듈에서 가져온다. Image 클래스는 이미지를 디스크에 저장하는 것과 같은 다른 작업과 마찬가지로 이미지의 오픈, 로드 및 변환을 담당한다. 이 예제에서는 open() 메소드를 사용하여 PNG 이미지를 열고 save() 메소드를 사용하여 이미지를 JPEG 형식으로 저장한다.

두 번째 예제에서는 이미지를 흑백 형식으로 변환한다. RGB 및 CMYK 형식과 마찬가지로 흑백을 나타내는 L 형식도 있다. 이미지를 L 형식으로 변환하는 동안, ITU-R luma 형식(L = R \* 299 / 1000 + G \* 587 / 1000 + B \* 114 / 1000)을 사용한다.

파이썬 코드에 관해서도 다시 Image 클래스를 사용하여 파일을 opne() 메소드를 사용하고 convert() 메소드를 사용하여 인수 L로 이미지를 흑백으로 변환한다. 마지막으로 save() 메소드를 사용하여 디스크에 파일을 저장한다. 여기서 파일 형식을 PNG로 유지한다.

세 번째 예제에서는 동일한 Image 클래스와 open() 메소드를 사용하여 img 객체를 가져온다. 이 객체는 회전 각도를 인자로 사용하여 rotate() 메소드를 호출하는 데 사용된다. 이 예제에서는 이미지를 180도 회전하고 마지막으로 save()를 호출하여 회전된 이미지를 sunset180deg.jpg라는 이름으로 디스크에 저장한다.

마지막 예제에서는 PIL 모듈의 transpose() 메소드를 사용하고 Image.FLIP\_LEFT\_RIGHT 및 Image.FLIP\_TOP\_BOTTOM 속성을 사용하여 이미지를 왼쪽-오른쪽 및 위-아래 두 방향 회전한 sunset\_horizontal\_flip.png 및 sunset\_vertical\_flip.png라는 이름으로 저장한다.

**더 알아보기**

Pillow 모듈에는 리사이징, 붙여넣기, 자르기와 같은 이미지에서 보다 복잡한 작업을 수행하는 데 도움이 되는 더 많은 메소드가 있다. 9장의 다음 레시피에서 살펴볼 것이다.

**크기 조정, 자르기 및 썸네일 생성**

이미지 크기 조정 및 자르기와 같은 작업은 이미지의 선택된 부분을 가져오기 위해 매우 일반적이지만 프로그래밍 방식으로 시도할 경우 이러한 작업이 더 이상 지루하지 않을 수 있다. 이러한 작업을 어떻게 수행할 수 있는지 살펴보자.

**준비하기**

이 레시피에서는 Pillow 라이브러리를 사용하여 이미지의 크기를 조정하고 자른다. 이미 Pillow 모듈을 설치했으므로 설치에 대해 걱정할 필요가 없다. 목표에 집중하자.

**실행방법**

1. 먼저 주어진 면적으로 이미지 크기를 조정하는 방법을 살펴보자. 파이썬 파일 resize.py를 생성하고 다음 코드를 붙인다.

from PIL import Image

img = Image.open('sunset.jpg')

resized = img.resize((256,256))

resized.save('sunset-resize.jpg', 'jpeg')

2. 또한 인터넷에서 이미지를 다운로드하고 sunset.jpg라고 이름을 정한다. 이미지는 다음과 같다.

<그림>

3. 이제 python resize.py 커맨드로 파이썬 코드를 실행하고 디스크의 sunset-resize.jpg 이미지를 살펴본다. 이미지의 크기가 조정되고 다음 스크린샷과 유사하게 보인다.

<그림>

이미지는 예상대로 256x256 픽셀이다.

<그림>

4. 프로그래밍에서 종종 필요로하는 또 다른 작업은 이미지에 대한 썸네일을 생성하는 것이다. 썸네일은 원본 이미지의 미리보기로 사용되며 일반적으로 영화 리뷰 웹사이트 혹은 서적 출판 웹사이트에서 사용된다. Pillow 모듈을 사용하여 썸네일을 쉽게 생성할 수 있는지 살펴보자. 파이썬 파일을 생성하고 이 코드를 추가한다.

import os, sys

from PIL import Image

size = 128, 128

infile = "sunset.jpg"

outfile = os.path.splitext(infile)[0] + ".thumbnail.jpg"

if infile != outfile:

try:

im = Image.open(infile)

im.thumbnail(size, Image.BICUBIC)

im.save(outfile, "JPEG")

except IOError:

print "cannot create thumbnail for '%s'" % infile

이제 이 코드를 실행하면 원본 이미지의 썸네일인 sunset.thumbnail.jpg 이미지가 생성될 것이고 아래 스크린샷과 같이 표시될 것이다. 이미지의 크기를 살펴보면 128x128(내 경우는 128x80 픽셀)이 아니다. 이것에 대한 이유를 조금 다룰 것이다.

좋아! 따라서 이미지를 위해 생성된 썸네일을 가지고 프로필 썸네일 혹은 혹은 미리보기 이미지로 웹사이트에서 사용될 수 있다.

<그림>

5. 이 레시피에서 다룰 다른 작업은 이미지 자르기이다. 다음 코드는 우리가 필요로 하는 것을 정확하게 수행한다.

from PIL import Image

img = Image.open('sunset.jpg')

cropImg = img.crop((965, 700, 1265, 960))

cropImg.save('sunset-crop.jpg')

위의 파이썬을 실행하면 디스크에 생성된 원본 일몰 이미지에서 태양을 자른 이미지 sunset-crop.jpg가 표시된다. 이것은 다음과 같다.

<그림>

Pillow를 사용하여 이미지에 대해 여러 작업을 매우 쉽고 직관적으로 수행할 수 있는 방법을 살펴볼 수 있어서 좋았다. 하지만 이러한 작업은 어떻게 작동할까? 사용된 메소드는 무엇인가? 그것들을 살펴보자.

**작동원리**

이 레시피에서는 Pillow의 Image 클래스를 사용하여 이미지의 크기를 조정하고 이미지를 자르고 원본 이미지에서 썸네일을 생성했다.

첫 번째 코드에서는 open() 메소드로 sunset.jpg 이미지를 열었다. 그런 다음 크기 조정된 이미지의 너비와 높이를 나열하는 튜플 인자와 함께 resize() 메소드를 사용했다. 그런 다음 save() 메소드로 파일 이름이 sunset-esize.jpg이고 JPEG 파일 형식인 파일을 디스크에 저장했다.

두 번째 코드에서는 open() 메소드를 사용하여 이미지를 열고 이미지 객체를 가져왔다. 그런 다음 이미지 객체에서 Image 클래스의 thumbnail() 메소드를 사용하여 썸네일을 생성했다. thumbnail() 메소드는 이미지 크기(128 x 128)를 결정하고 BICUBIC 이미지 필터링 메커니즘을 사용한다. 마지막으로 save() 메소드로 이미지를 저장했으며 대상 파일 이름을 sunset.thumbnail.jpg로 설정했다. 썸네일의 크기를 살펴본 결과 정확히 128x128 크기가 아니라는 것을 알았다. 실제로는 128 x 80이다. PIL은 이미지 너비를 128 픽셀로 유지한 다음 높이를 다시 계산하여 이미지의 비율을 유지한다.

그리고 세 번째 예제는 Pillow 모듈에서 Image 클래스의 crop() 메소드로 이미지를 잘랐다. crop() 메소드는 원본 이미지에서 이미지를 조각낼 필요가 있는 네 개의 좌표를 모두 가져온다. 이 예제에서는 원본 이미지를 자르기 위해 좌표를 left = 965, top = 700, right = 1265, bottom = 960으로 지정했으므로 예제에서 보았듯이 얻은 결과는 태양의 이미지이다.

**더 알아보기**

썸네일 생성 예제에서, 더 명확하게 하기위해 이미지에 적용된 필터에 대해 간단히 언급했다. 9장에서 자세히 다루지는 않겠지만 관심이 있다면 http://pillow.readthedocs.io/en/3.0.x/releasenotes/2.7.0.html#default-filter-for-thumbnails에서 자세히 살펴본다.

**복사-붙여넣기 및 워터마킹 이미지**

이 레시피는 디자이너와 마케팅 담당자가 가장 많이 사용하는 작업, 즉 워터마크 이미지를 만드는 작업을 다룰 것이다. 이미지를 서로 복사-붙여넣기하는 재미있는 방법도 다룰 것이다. 계속 살펴보자.

**준비하기**

이 레시피에서는 이미지의 복사-붙여넣기를 위해 Pillow를 계속 사용하지만 워터마킹을 위해 다른 파이썬 모듈인 wand를 사용한다. 따라서 정상적인 연습처럼 코드 작성을 시작하기 전에 먼저 wand 모듈을 설치하자. 가장 좋아하는 도구인 파이썬의 pip로 wand를 설치한다.

(ch10)chetans-MacBookPro:ch10 Chetan$ pip install wand

You are using pip version 7.1.0, however version 8.1.2 is

available.

You should consider upgrading via the 'pip install --upgrade

pip' command.

Collecting wand

Downloading Wand-0.4.3.tar.gz (65kB)

100% |████████████████████████████████|

65kB 101kB/s

Building wheels for collected packages: wand

Running setup.py bdist\_wheel for wand

Stored in directory:

/Users/chetan/Library/Caches/pip/wheels/77/

c2/a3/6cfc4bb3e21c3103df1ce72d7d301b1965657ee6f81cd3738c

Successfully built wand

Installing collected packages: wand

Successfully installed wand-0.4.3

모듈을 이미 설치했는가? 좋아, 이제 시작하자.

**실행방법**

1. 먼저 Pillow로 복사-붙여넣기 작업을 수행하는 방법을 살펴보자. 이전 절에서는 원본 이미지인 sunset.jpg 및 원래 이미지에서 자른 태양 이미지인 sunset-crop.jpg의 두 가지 이미지가 있음을 기억한다. 다음 파이썬 코드에서 이 이미지들을 사용할 것이다.

from PIL import Image

img = Image.open('sunset-crop.jpg')

pasteImg = Image.open('sunset.jpg')

pasteImg.paste(img, (0,0))

pasteImg.save('pasted.jpg')

2. 파일 코드를 copy\_paste.py라는 이름으로 저장하고 파이썬 커맨드로 copy\_paste.py를 사용하여 코드를 실행하자. 코드를 실행하면 다음 스크린샷과 같은 pasted.jpg라는 새 파일이 생성된다.

<그림>

자른 이미지를 복사하여 원본 이미지에 붙이고 붙인 이미지를 pasted.jpg로 저장하는 것이다. 멋지지 않은가?

이제 상업적으로 사용되는 흥미로운 예제를 살펴보자. 이 예제에서는 기존 이미지에 워터마크를 추가하고 다른 이름으로 저장한다. 그러나 파이썬 코드에 들어가기 전에 워터마크 이미지가 어떻게 보이는지 살펴보자.

<그림>

다음 파이썬 코드는 이전의 워터마크를 원래의 sunset.jpg 이미지 파일에 추가하는 데 도움이 된다.

from wand.image import Image

with Image(filename='sunset.jpg') as background:

with Image(filename='watermark.jpg') as watermark:

background.watermark(image=watermark, transparency=0.25,

left=560, top=300)

background.save(filename='result.jpg')

3. 이 코드를 실행하면 프로젝트에서 result.jpg 파일이 생성될 것이다. 다음 스크린샷과 비슷하게 보일것이다. Copyrighted Image 텍스트가 위에 표시된 상태로 이미지가 워터마크되는 방법을 살펴본다.

<그림>

**작동원리**

첫 번째 코드는 PIL 모듈과 Image 클래스를 사용하여 open() 메소드로 자른 이미지와 원본 이미지를 두 파일 즉 img 및 pasteImg의 파일 핸들을 가져왔다.

이름에서 알 수 있듯이 자른 이미지 img를 열고 파일 핸들을 사용하여 pasteImg에 붙인다.

이미지를 붙이기 위해 Pillow의 paste() 모듈을 사용하고 img 파일 핸들을 소스 이미지로 전달했다. 또한 원본 이미지에 자른 이미지를 붙여 넣을 좌표를 전달했다. 좌표를 (0, 0)으로 선택했으므로 자른 이미지는 원본 이미지의 왼쪽 상단 모서리에 붙인다. 마지막으로, 이 이미지를 pasted.jpg로 저장했다.

두 번째 예제에서는 원본 이미지 sunset.jpg와 워터마크 이미지 watermark.jpg를 열고 각각 파일 핸들 background 및 watermark를 생성했다. 그런 다음 wand 모듈의 watermark() 메소드를 사용하여 원본 이미지에 워터마크를 추가했다.

watermark() 메소드는 배경이미지 객체(이 경우 백그라운드, 원래 이미지 객체)에서 작동한다. 이미지를 keyword 인자로 사용하여 워터마크 이미지의 객체를 나타낸다. 워터마크 이미지의 투명도를 설정할 수도 있는데, 여기서 0은 워터마크가 완전히 보이고 1은 보이지 않음을 나타낸다. watermark() 메소드로 얻을 수 있는 또 다른 유용한 점은 원본 이미지에서 워터마크 위치를 선택할 수 있다는 것이다. 이 예제에서는 왼쪽에서 560, 위쪽에서 300이 되도록 좌표를 조절했다.

멋지다. 이 레시피에 나오는 전부다. 9장의 남은 레시피에서 저장하고 있는 것을 살펴보자.

**이미지 차이 및 비교**

여러분은 텍스트 기반 검색을 사용하거나 심지어 구현해야 한다. 하지만 이미지 기반 검색을 수행할 수도 있을까?. 당연히, 구글은 그것을 꽤 잘한다. 어떻게 생각하는가? 직접 구현해야 하는 경우 두 이미지를 비교하는 방법을 잘 알아야한다. 사용 사례에 따라 diff를 얻거나 혹은 두 이미지 간의 차이를 가져올 수도 있다. 이 레시피에서는 두 가지 사용 사례를 다룰 것이다.

-두 이미지의 차이를 얻고 이미지로 그 차이를 저장하는 방법

-두 이미지를 과학적 메소드로 객관적으로 비교하는 방법

**준비하기**

이 레시피는 계속해서 Pillow로 이미지를 비교할 것이다. Image 클래스와 함께 ImageChops 클래스를 사용하여 두 이미지의 차이를 가져온다. 픽셀 단위의 이미지를 비교하기 위해 scipy 모듈을 사용한다.

1. 이미 Pillow 모듈을 설치했으므로 파이썬 pip를 사용하여 scipy 모듈을 설치하자. Mac OS X 머신에서는 scipy 모듈을 설치하기위한 컴파일러가 필요하다. Mac의 brew 커맨드로 Mac에 GCC 컴파일러를 설치할 것이다.

(ch10)chetans-MacBookPro:ch10 Chetan$ brew install gcc

Warning: Building gcc from source:

The bottle needs the Xcode CLT to be installed.

==> Using the sandbox

==> Downloading <https://ftpmirror.gnu.org/gcc/gcc-6.2.0/gcc>-

6.2.0.tar.bz2

Already downloaded: /Users/chetan/Library/Caches/Homebrew/gcc-

6.2.0.tar.bz2

==> Downloading

<https://raw.githubusercontent.com/Homebrew/formulapatches/>

e9e0ee09389a54cc4c8fe1c24ebca3cd765ed0ba/gcc/6.1.0-

jit.patch

Already downloaded: /Users/chetan/Library/Caches/Homebrew/gcc--

patch- 863957f90a934ee8f89707980473769cff47

ca0663c3906992da6afb242fb220.patch

==> Patching

==> Applying 6.1.0-jit.patch

patching file gcc/jit/Make-lang.in

==> ../configure --build=x86\_64-apple-darwin15.5.0 --

prefix=/usr/local/Cellar/gcc/6.2.0 --

libdir=/usr/local/Cellar/gcc/6.2.0/lib/gcc/6 --enablelanguages=

c,c++,objc,obj-c++,fortran

==> make bootstrap

==> make install

==> Caveats

GCC has been built with multilib support. Notably,

OpenMP may not

work:

<https://gcc.gnu.org/bugzilla/show_bug.cgi?id=60670>

If you need OpenMP support you may want to

brew reinstall gcc --without-multilib

==> Summary

/usr/local/Cellar/gcc/6.2.0: 1,436 files, 282.6M, built in 70

minutes 47 seconds

(ch10)chetans-MacBookPro:ch10 Chetan$

2. 이제 GCC가 설치되었으므로, python-pip로 scipy를 설치하자. 시스템에 설치 로그는 다음과 같다.

(ch10)chetans-MacBookPro:ch10 Chetan$ pip install scipy

You are using pip version 7.1.0, however version 8.1.2 is

available.

You should consider upgrading via the 'pip install --upgrade

pip'

command.

Collecting scipy

Using cached scipy-0.18.1.tar.gz

Building wheels for collected packages: scipy

Running setup.py bdist\_wheel for scipy

Stored in directory:

/Users/chetan/Library/Caches/pip/wheels/33/

c4/f5/e00fe242696eba9e5f63cd0f30eaf5780b8c98067eb164707c

Successfully built scipy

Installing collected packages: scipy

Successfully installed scipy-0.18.1

**실행방법**

1. 모듈이 이미 설치됐으므로, 필요에 맞게 모듈을 활용하자. 우선, 두 이미지 간의 차이를 가져와서 그 차이를 이미지 자체로 저장하는 방법을 살펴보자. 다음 코드는 이 작업을 수행한다.

from PIL import Image, ImageChops

def differnce\_images(path\_one, path\_two, diff\_save\_location):

image\_one = Image.open(path\_one)

image\_two = Image.open(path\_two)

diff = ImageChops.difference(image\_one, image\_two)

if diff.getbbox() is None:

print "No difference in the images"

return

else:

print diff.getbbox()

diff.save(diff\_save\_location)

differnce\_images('sunset.jpg','pasted.jpg',

'diff.jpg')

앞의 코드 예제에서는, 원본 이미지인 sunset.jpg와 복사-붙여넣기 이미지인 pasted.jpg의 차이를 계산했다(이전 레시피를 기억하면 pasted.jpg는 원래 일몰 이미지 위에 자른 태양 이미지를 붙여서 얻은 이미지). 차이 이미지가 보이는 방식이다.

<그림>

기본 원본 이미지는 동일하게 유지되므로 잘린 태양 이미지만 어떻게 다른지 살펴본다.좋다! 검은색 영역은 무엇을 나타내는가? ‘작동원리’절에서 설명할 것이다.

2. 이제 객관적인 방식으로 이미지 간의 차이를 계산하자. 이를 위해 scipy 모듈을 사용한다. 다음 코드 예제는 우리가 필요한 것을 도와준다.

from scipy.misc import imread

from scipy.linalg import norm

def compare\_images(img1, img2):

diff = img1 - img2

z\_norm = norm(diff.ravel(), 0)

return z\_norm

img1 = imread("sunset.jpg").astype(float)

img2 = imread("pasted.jpg").astype(float)

z\_norm = compare\_images(img1, img2)

print "Pixel Difference:", z\_norm

위의 파이썬 코드를 실행하면, 두 이미지의 픽셀 차이를 모두 확인할 수 있을 것이다. 예제의 출력은 다음과 같다.

Pixel Difference: 246660.0

**작동원리**

첫 번째 코드에서는 Pillow 라이브러리의 ImageChops 클래스를 사용하여 두 이미지 간의 차이를 계산했다. 평소와 같이, open() 메소드로 두 이미지를 열고 개별 이미지 객체 image\_one 및 image\_two를 얻었다.

이 후 ImageChops 클래스의 difference() 메소드를 사용하고 이 메소드의 인자로 이미지 객체를 전달했다. difference() 메소드는 diff 객체를 반환하며, 기본적으로 두 이미지 간의 차이를 나타내는 객체이다.

마지막으로 차이를 diff.jpg라는 이름으로 디스크에 이미지로 저장했다. 또한 이미지에서 0이 아닌 영역의 경계 상자를 계산하는 diff 객체의 getbbox() 메소드를 사용했다. 여기서 0이 아닌 영역은 sunset.jpg와 pasted.jpg의 차이가 0인 픽셀을 나타낸다.

이제, diff.jpg를 살펴보면 거대한 검은색 영역를 포함한다. 이는 차이가 0인 픽셀이며, 따라서 색상이 검정이다. 동일한 이미지의 경우, getbbox() 메소드는 None을 반환한다.

두 번째 예제에서는 최저 기준(zero norm)을 기초하여 두 이미지를 비교했는데, 이것은 픽셀 수가 0이 아니라는 것을 나타내며, 다시 말해서 두 이미지 간에 얼마나 많은 픽셀이 다른지를 나타낸다. 이미지를 비교하기 위해, 우선 scipy 모듈의 imread() 메소드를 사용하여 두 이미지를 모두 읽는다. 이미지 객체는 img1과 img2이다.

이 후 diff = img1 - img2로 두 이미지 간의 차이를 계산했다. 반환된 이 차이는 scipy의 ndarray 유형이다. 이 차이를 norm() 메소드로 전달하면 이미지 간에 다른 픽셀 수를 반환한다.

**더 알아보기**

이미지를 비교하는 방법은 여러 가지가 있지만, 9장에서 다루지 않는다. 정말로 관심이 있다면 더 중점적으로 독서를 할 것을 추천한다. 하지만 모든 실제적인 목적을 위해서 9장은 충분하다고 생각한다.

**얼굴 추출**

이전 절에서 다양한 이미지 작업을 다뤘다. 이 레시피에서는 더 깊이 파고들어 얼굴 추출(face detection) 같은 고급 작업을 담아보도록 하자.

**준비하기**

이 레시피에서는 파이썬 opencv 모듈을 사용할 것이므로 필요한 모듈을 설치하자.

1. opencv 파이썬 바인딩을 사용하려면 먼저 컴퓨터에 opencv를 설치해야 한다. 내 Mac OS X 머신에서는, 이 방법으로 opencv를 설치하기 위해 brew 유틸리티를 사용한다.

(ch10)chetans-MacBookPro:ch10 Chetan$ brew install

homebrew/science/opencv

==> Tapping homebrew/science

Cloning into '/usr/local/Homebrew/Library/Taps/homebrew/homebrew-

science'...

...

...

==> Summary

/usr/local/Cellar/opencv/2.4.13.1: 277 files, 35.5M

2. 컴퓨터에 opencv를 설치하는 것만으로는 도움이 되지 않는다. 또한 다음 커맨드를 사용하여 cv2.so(.so는 공유 객체 혹은 라이브러리를 나타낸다.)가 가상 환경에 연결되도록 지정해야 한다.

cd ch10/lib/python2.7/site-packages/

ln -s /usr/local/Cellar/opencv/2.4.13.1/lib/python2.7/site-

packages/cv.py

ln -s /usr/local/Cellar/opencv/2.4.13.1/lib/python2.7/site-

packages/cv2.so

좋아! 그래서 지금 이 레시피의 예제에 필요한 opencv를 설치했다.

**실행방법**

1. 가장 좋아하는 편집기로 이동하여 파이썬 파일을 생성하고 face\_detection.py로 저장한다. 이제 다음 코드를 파이썬 파일에 복사한다.

import cv2

face\_cascade = cv2.CascadeClassifier('haarcascade.xml')

original\_image\_path = 'Chetan.jpeg'

image = cv2.imread(original\_image\_path)

faces = face\_cascade.detectMultiScale(

image, scaleFactor=1.1, minNeighbors=3,

minSize=(30, 30), flags=cv2.cv.CV\_HAAR\_SCALE\_IMAGE)

for (x, y, w, h) in faces:

cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.imwrite('chetan\_face.jpg', image)

2. 이제 haarcascade.xml의 이름으로 XML 파일을 생성하고 이 책의 코드 저장소에서 콘텐츠를 복사한다. 예제에서, 내 사진 중 하나인 Chetan.jpeg를 사용하고 있지만, 이 예제에서는 어떤 사진이든 사용할 수 있다. Chetan.jpeg은 다음과 같이 보인다.

<그림>

3. 이제 파이썬 코드를 실행하고 코드가 이미지에서 내 얼굴을 인식할 수 있는지 살펴본다. python face\_detection.py 커맨드로 코드를 실행하고 다음과 같은 Chetan\_face.jpg 이미지를 생성한다. 실제로, 내 얼굴을 감지했다.

<그림>

**작동원리**

레시피에서는 우선 haarcascade.xml 파일로 케스케이드 분류기(cascade classifier) 객체를 생성하기 위해 opencv 모듈을 사용했다. 이 객체를 face\_cascade라고 불렀다.

하르(Haar)의 특징 기반 케스케이드 분류기(cascade classifiers)를 이용한 객체(Object) 감지는 2001년 폴 비올라(Paul Viola)와 미카엘 존스(Michael Jones)에 의해 제안된 효과적인 객체 감지 방법이다. 이것은 케스케이드 함수(cascade function)가 많은 양화 및 음화(positive and negative images)로부터 훈련된 기계 학습 기반 방식이다. 그런 다음 다른 이미지에서 객체를 감지하는 데 사용된다.

하르 특징은 코드 예제에서 사용한 표준 XML 파일에 제공된다. 실제로 원하는 객체를 탐지하기 위해 분류기(classifier)를 훈련시킬 수 있다. 예를 들어, 눈 추출(eye detection)은 다른 분류기를 사용한다.

다음으로, opencv 모듈의 imread() 메소드를 사용하여 원본 기본 이미지인 Chetan.jpeg를 읽고 탐지를 위한 최소 윈도우를 정의했다.

사실 하르 케스케이드 분류기(cascade classifier)는 슬라이딩 윈도우 접근 방식에서 작동하므로 탐지를 위한 최소 윈도우를 필요로 한다. 분류기는 구성된 minNeighbors도 필요하다.

이와 같은 설정은 케스케이드 객체의 detectMultiScale() 메소드에서 구성된다. minSize=(30,30) 및 minNeighbors=3으로 설정했다.

마지막으로, 추출된 이미지를 디스크에 저장했고 원본 이미지는 녹색 사각형을 이미지의 얼굴 추출 표시로 사용한다.

**더 알아보기**

opencv로 얼굴 추출의 매우 간단한 예제를 살펴보고 분류기에 대해 조금 살펴봤다. opencv는 배울 것이 더 많다.

<http://docs.opencv.org/trunk/index.html>는 읽기 흥미로운 리소스에 대한 링크다.

**비즈니스 프로세스의 이미지**

피터(Peter)는 대기업인 MBI 주식회사의 IT 관리자이다. 피터의 회사는 계약 자금 문서, 표준 운영 절차 및 공급망 문서의 대부분을 종이 기반으로 할 수 있을 정도로 충분히 오래 존재해왔다. 피터는 자신의 회사를 종이없는 회사로 만드는 엄청난 책임을 가진다.

즉, 서류 기록을 관리하는 번거로움과 비용을 없앨 책임이 있음을 의미한다. 9장에서는 지금까지 살펴본 이미지관련 지식(그리고 더 많이 배우게 될 것이다)을 통해 피터를 도울 수 있는지 살펴보자.

신중하게 분석한다면 피터는 두 가지 중요한 작업을 수행해야한다.

-종이를 스캔하고 이미지를 전자 형식으로 저장한다.

-쉽게 색인할 수 있도록 이러한 문서의 텍스트 파일을 생성한다.

**준비하기**

이 연습을 위해, 필요한 모듈을 설치하는 것으로 시작하자. 다음 모듈이 필요할 것이다.

-scikit-image (<http://scikit-image.org/>)

-pyimagesearch (<http://www.pyimagesearch.com/>)

-tessaract and pytesseract (<https://pypi.python.org/pypi/pytesseract/>)

모듈 설치를 시작하자.

1. 우선 scikit-image의 설치를 시작하자.

(ch10)chetans-MacBookPro:ch10 Chetan$ pip install scikit-image

You are using pip version 7.1.0, however version 8.1.2 is

available.

You should consider upgrading via the 'pip install --upgrade

pip' command.

Collecting scikit-image

Downloading scikit-image-0.12.3.tar.gz (20.7MB)

100% |████████████████████████████████|

20.7MB 19kB/s

Requirement already satisfied (use --upgrade to upgrade):

six>=1.7.3 in ./lib/python2.7/site-packages (from scikit-image)

Collecting networkx>=1.8 (from scikit-image)

Downloading networkx-1.11-py2.py3-none-any.whl (1.3MB)

100% |████████████████████████████████|

1.3MB 325kB/s

Requirement already satisfied (use --upgrade to upgrade):

pillow>=2.1.0 in ./lib/python2.7/site-packages (from scikitimage)

Collecting dask[array]>=0.5.0 (from scikit-image)

Downloading dask-0.11.1-py2.py3-none-any.whl (375kB)

100% |████████████████████████████████|

376kB 946kB/s

Collecting decorator>=3.4.0 (from networkx>=1.8->scikit-image)

Using cached decorator-4.0.10-py2.py3-none-any.whl

Collecting toolz>=0.7.2 (from dask[array]>=0.5.0->scikit-image)

Using cached toolz-0.8.0.tar.gz

Requirement already satisfied (use --upgrade to upgrade):

numpy in

./lib/python2.7/site-packages (from dask[array]>=0.5.0->scikitimage)

Building wheels for collected packages: scikit-image, toolz

Running setup.py bdist\_wheel for scikit-image

Stored in directory:

/Users/chetan/Library/Caches/pip/wheels/d5/e8/77/925fe026d562a74a0bccf1c7dd

47d00f5f6ab2d395f247e674

Running setup.py bdist\_wheel for toolz

Stored in directory:

/Users/chetan/Library/Caches/pip/wheels/b0/84/bf/7089262387e8ea60bdefb1fdb8

4d2ee99427f6d09c9c7ba37d

Successfully built scikit-image toolz

Installing collected packages: decorator, networkx, toolz,

dask,

scikit-image

Successfully installed dask-0.11.1 decorator-4.0.10 networkx-

1.11

scikit-image-0.12.3 toolz-0.8.0

2. 다음으로 pyimagesearch를 설치하자. 이것은 아드리안 로즈브록(Adrian Rosebrock)이 개발한 훌륭한 라이브러리 모음이다. 아드리안 로즈브록은 자신의 작업을 http://github.com/jrosebr1에서 오픈소스로 공개했다. 사실, 이 코드 레시피에는 pyimagesearch의 스캐너 예제를 활용한다.

3. 마지막으로, tesseract와 pytesseract를 설치하자. OCR(Optical Character Reader) 모듈, tesseract, OCR 모듈을 사용하기 위한 파이썬 모듈인 pytesseract를 설치해야 한다.

(ch10)chetans-MacBookPro:ch10 Chetan$ brew install tesseract

==> Auto-updated Homebrew!

Updated 4 taps (homebrew/core, homebrew/dupes, homebrew/python,

homebrew/science).

..

..

..

==> Installing dependencies for tesseract: leptonica

==> Installing tesseract dependency: leptonica

==> Downloading <https://homebrew.bintray.com/bottles/leptonica>-

1.73.el\_capitan.bottle.tar.gz

#######################################################

################ # 100.0%

==> Pouring leptonica-1.73.el\_capitan.bottle.tar.gz

/usr/local/Cellar/leptonica/1.73: 50 files, 5.4M

==> Installing tesseract

==> Downloading <https://homebrew.bintray.com/bottles/tesseract>-

3.04.01\_2.el\_capitan.bottle.tar.gz

########################################################

############### # 100.0%

==> Pouring tesseract-3.04.01\_2.el\_capitan.bottle.tar.gz

/usr/local/Cellar/tesseract/3.04.01\_2: 76 files, 39M

(ch10)chetans-MacBookPro:ch10 Chetan$ pip install pytesseract

Collecting pytesseract

Downloading pytesseract-0.1.6.tar.gz (149kB)

100% |████████████████████████████████|

151kB 201kB/s

Building wheels for collected packages: pytesseract

Running setup.py bdist\_wheel for pytesseract

Stored in directory:

/Users/chetan/Library/Caches/pip/wheels/f2/27/64/

a8fa99a36b38980aaf8d1d2c87f5dd6b5a0a274b8706e3df36

Successfully built pytesseract

Installing collected packages: pytesseract

Successfully installed pytesseract-0.1.6

그래, 좋아! 이제 '실행방법' 절의 코드를 살펴보자.

**실행방법**

1. 가장 좋아하는 편집기로 이동하여 파이썬 파일을 생성하고 이름을 scanner.py로 지정한다. 피터의 경우, 이미지 형식의 재무 문서에 관한 것이지만, 이 예제를 살펴보면, 나와 함께 편리하게 사용할 수 있는 이미지가 될 것이다. 내 이미지의 모습은 다음과 같다. 앤디 머레이(Andy Murray)에 대한 신문 기사의 그림이고 그것을 디지털화하려고 한다.

<그림>

2. 이제 scanner.py에 다음 코드를 복사하고 python scanner.py 커맨드로 코드를 실행한다.

from pyimagesearch.transform import four\_point\_transform

from pyimagesearch import imutils

from skimage.filters import threshold\_adaptive

import cv2

image = cv2.imread("images/murray.jpg")

ratio = image.shape[0] / 500.0

orig = image.copy()

image = imutils.resize(image, height = 500)

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (5, 5), 0)

edged = cv2.Canny(gray, 75, 200)

cv2.imwrite('scan\_edge.jpg', edged)

(cnts, \_) = cv2.findContours(edged.copy(), cv2.RETR\_LIST,

cv2.CHAIN\_APPROX\_SIMPLE)

cnts = sorted(cnts, key = cv2.contourArea, reverse = True)[:5]

for c in cnts:

peri = cv2.arcLength(c, True)

approx = cv2.approxPolyDP(c, 0.02 \* peri, True)

if len(approx) == 4:

screenCnt = approx

break

cv2.drawContours(image,

[screenCnt], -1, (0, 255, 0), 2)

cv2.imwrite('scan\_contours.jpg', image)

warped = four\_point\_transform(orig, screenCnt.reshape(4, 2) \*

ratio)

warped = cv2.cvtColor(warped, cv2.COLOR\_BGR2GRAY)

warped = threshold\_adaptive(warped, 251, offset = 10)

warped = warped.astype("uint8") \* 255

cv2.imwrite('scanned.jpg', warped)

print "Printing the contents of the image:"

from PIL import Image

img = Image.open("scanned.jpg")

import pytesseract

print(pytesseract.image\_to\_string(img))

파이썬 코드를 실행하면 하드 디스크에 세 개의 이미지가 생성될 것이다. 첫 번째는 edge-detect 이미지이고 제 경우에는 다음과 같이 보인다. scan\_edge.jpg로 저장된다.

<그림>

3. 다음으로, 텍스트가 있는 이미지의 전체 영역을 감지하는 다른 이미지를 얻는다. 이를 윤곽 이미지(contours image)라고하며 scan\_contours.jpg로 생성된다. 텍스트를 사용할 수 있는 이미지의 부분을 강조하는 방법을 살펴보자.

<그림>

4. 마지막으로 앤디 머레이(Andy Murray)의 뉴스 기사를 스캔한 복사본을 얻고, scanned.jpg 파일로 저장된다. 다음 스크린샷에서 뉴스 기사가 얼마나 잘 스캔되었는지 살펴본다.

<그림>

좋아, 처음 시작했을 때 정확히 이루고 싶었던 것이다, 그렇지? 이걸 피터에게 주면 매우 행복해할 것이다. 피터는 비용이 많이 드는 컨설팅 회사 혹은 스캐너를 생각하고 있을 수도 있지만, 우리는 신속하게 할 수 있고, 무료로 할 수 있다.

5. 이것이 충분하지 않은 것처럼, 여기서 피터를 위해 더 많은 것을 했다. 프로그램을 실행하면, 전체 기사의 텍스트를 제공하는 텍스트 출력도 얻을 수 있다. 이 텍스트를 사용하여 피터는 문서를 분류하고 이에 따라 색인을 생성할 수 있다.

Printing the contents of the image:

Vienna: Andy Murray's pm"

suitof the world No. Iranking

was severely tested by Slovak

lefi-hander Martin Klizan in

his Vienna openeron Wednwo

day before the British star

clinched a 6-3, 64 (57), 6-0 win.

The 29-year-old Wimbledon

and Olympic champion,

who is closing in on Novak

, Djokovic's top ranking, took

his season record to 66 wins

and just ninelosses. fl

굉장해! 프로그램의 내부를 '작동 원리' 절에서 살펴보자.

**작동원리**

레시피에서 신문 기사의 사진을 가져오기 시작했다. 이것을 murray.jpg라고 저장했다. 이 이미지를 가져오기 위해 간단한 카메라 폰을 사용했다. 그런 다음 opencv 모듈의 imread() 메소드를 사용하여 이미지를 읽었다.

또한 또한 새로운 높이에 대한 원래 높이의 비율을 계산하고, 원본 이미지를 복사하고 크기를 조정했다. copy() 메소드를 사용하여 이미지를 복사했으며, resize() 메소드를 사용하여 크기를 새로운 높이로 조정했다.

그런 다음 이미지를 cvtColor() 메소드로 그레이스케일(grayscale) 형식으로 변환한 다음 가우시안(Gaussian) 필터(filter)를 적용하여 이미지를 흐리게 표현했다.

Canny() 메소드를 사용하여 텍스트의 가장자리를 감지하기 위해 흐린 이미지(blurred image)를 적용했고, 마지막으로 edge-detect 이미지를 scan\_edge.jpg 파일로 저장했다.

다음으로 findContours() 메소드를 사용하여 이미지의 윤곽(contours)을 찾고 이미지의 결과를 scan\_contours.jpg로 저장했다.

이 후 이미지에 몇 가지 변형을 가했다. 4점 변환(four-point transform)을 사용하면 원본 이미지를 탑-다운 뷰(top-down view)로 볼 수 있다. 이를 위해 four\_point\_transform() 메소드를 사용했다.

또한 이미지를 그레이스케일로 변환한 다음 흑백 용지 스타일로 표현하도록 임계값을 설정했다. cvtColor() 메소드는 이미지를 그레이스케일로 변환하고 threshold\_adaptive() 메소드는 적절한 임계값을 적용한다. 그리고 끝났다. 이미지가 준비되었으며 이미 scanned.jpg 파일로 스캔되어 저장됐다.

그러나 이전 절에서 보았듯이 신문 칼럼의 텍스트도 인쇄했다. 이미지 객체에서 image\_to\_string() 메소드를 사용하여 Pillow의 Image 클래스와 pytessaract 모듈로 스캔된 이미지를 우선 읽음으로써 이를 해결할 수 있다.

좋다, 그래서 종이 기반 문서를 전자 형식으로 변환하는 업무 프로세스를 자동화하고 파일을 회사의 ERP프로세스에 쉽게 적용할 수 있도록 파일에 색인화 기능을 추가했다. 피터는 여러분과 행복하다. 축하해!

**더 알아보기**

스캔한 이미지의 텍스트 데이터를 추출하기 위해 OCR을 살펴봤지만, 지능형 문자 인식(손으로 쓴 텍스트 추출) 및 바코드 인식(다양한 유형의 바코드 인식)과 같은 더 많은 것들을 할 수 있다. 또한 9장에서는 이미지 필터링을 많이 다루지 않았다. 정말로 관심이 있다면, 9장의 범위를 넘어서는 이 주제에 대한 많은 읽을 거리를 얻을 수 있다.