Workshop Notebook 7: AutoCropper with OpenCV

Mandatory Disclosures

1. This is a whirlwind introduction, not exhaustive instruction
2. All images are by courtesy of the University Archives at Texas State University: [http://www.univarchives.txstate.edu](http://www.univarchives.txstate.edu)
3. img_qc_workshop is licensed under the GNU General Public License v3.0, [https://github.com/photosbyjeremy/img_qc_workshop/blob/master/LICENSE](https://github.com/photosbyjeremy/img_qc_workshop/blob/master/LICENSE)
4. Any and all code provided is done so without any warranty or expectation of support by Jeremy Moore, Todd Peters, or Texas State University

```python
In [ ]: # importing
import cv2
import matplotlib.pyplot as plt
import numpy as np
from pathlib import Path
from PIL import Image
import img_qc.img_qc as img_qc
```

```python
In [ ]: # magic that lets us plot directly in the notebook
%matplotlib inline
# parameters for matplotlib to increase our default figure size -- NOTE: figure sizes are in INCHES
plt.rcParams["figure.figsize"] = (12,20) # set as needed for your screen and eyes
# on a high-dpi monitor this will increase the quality of plots on-screen
%config InlineBackend.figure_format = 'retina'
```

```python
In [ ]: image_path = 'data/workshop-7/graduate_catalog_1949/graduate_catalog_1949_0007.tif'
```

```python
In [ ]: # open image
image_original = Image.open(image_path)
# display image
plt.imshow(image_original)
```

```python
In [ ]: # === AutoCrop
# load the image
image = cv2.imread(image_path)
# compute the ratio of the old height to the new height
ratio = image.shape[0] / 500.0
# clone image
original = image.copy()
# resize image
image = img_qc.get_resized_cv_image(image, height=500)
# convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# blur the image
blurred = cv2.GaussianBlur(gray, (5, 5), 0)
# apply Otsu’s automatic thresholding
(T, thresh) = cv2.threshold(blurred, 0, 255,
                         cv2.THRESH_BINARY | cv2.THRESH_OTSU)
# find the contours in the thresholded image keeping the external one
image, contours, hierarchy = cv2.FindContours(thresh.copy(), cv2.RETR_EXTERNAL,
                              cv2.CHAIN_APPROX_SIMPLE)
cnts = contours
```
# sort the contours from left to right
(cnts, bounding_boxes) = img_qc.sort_contours(cnts)
close = image.copy()

# Loop over the contours individually
for (i, c) in enumerate(cnts):
    # if the contour is not sufficiently large, ignore it
    if cv2.contourArea(c) < 700:
        continue

    # compute the rotated bounding box of the contour
    box = cv2.minAreaRect(c)
    box = cv2.boxPoints(box)
    box = np.int0(box)
    cv2.drawContours(clone, [box], 0, (0, 0, 255), 2)

    # re-order the points in tl, tr, br, bl order
    rect = img_qc.order_points(box)

    # clone the image and find the points and angle for minAreaRectangle
    clone = image.copy()
    (x, y), (w, h), theta = cv2.minAreaRect(c)

    # rotate image around center of minAreaRect by theta amount
    if theta < -45:
        theta = 90 + theta
    rotated = img_qc.rotate(clone, theta, (x, y))

    # multiply the rectangle by the original ratio
    rect *= ratio

    # find the points we need to crop the full size original
    tl, tr, br, bl = rect
    startx = max(min(tl[0], bl[0]), 0)
    starty = max(min(tl[1], tr[1]), 0)
    endx = max(tr[0], br[0])
    endy = max(bl[1], br[1])

    # rotate original by theta from minAreaRect
    x *= ratio
    y *= ratio
    rotated = img_qc.rotate(original, theta, center=(x, y))

    # add padding to the image if set as argument
    # NOTE: default is 0
    padding = -50
    pixel_padding = int(padding)
    startx = pixel_padding
    starty = pixel_padding
    # debug: set startx and starty to max of their current value and 0
    startx = max(startx, 0)
    starty = max(starty, 0)
    endx += pixel_padding
    endy += pixel_padding

    # crop the image in memory
    crop = rotated[int(starty):int(endy), int(startx):int(endx)]

    # check for and create a save directory (currently hardcoded, could
    # set as argument)
    head, tail = os.path.split(image_path)
    added_path = "\00_cropped/"
    save_path = head + added_path
    try:
        os.makedirs(save_path)
    except OSERROR:
        if not os.path.isdir(save_path):
            raise

    # split filename and create new output path with (currently hardcoded, could
    # set as argument)
    base, ext = os.path.splitext(tail)
    save_name = base + "_crop" + ext -- took this out so I don't have to rename
    # everything anymore
    save_name = base + ext
    output_path = save_path + save_name
# save our cropped image to disk in the new location
cv2.imwrite(output_path, crop)

In [ ]: # open image
image_autocropped = Image.open(output_path)

# display image
plt.imshow(image_autocropped)