R13922136 周洛君

Computer Vision Homework 1 Report

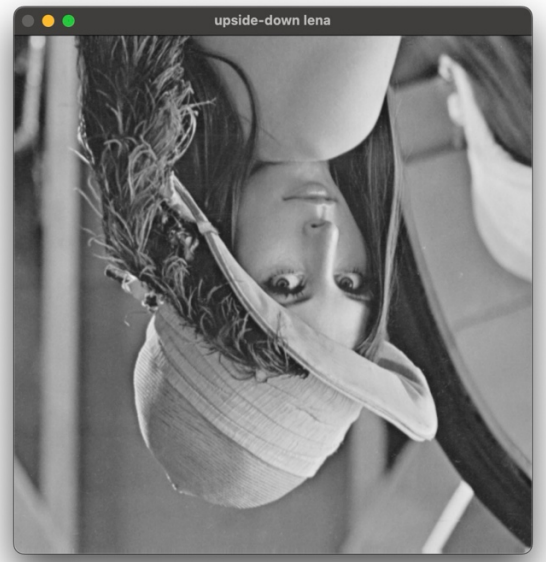
The homework consists of 2 parts, marked by index (a) to (f). Except for (e), all the other requirements were done in the code.

To get the resulting image for all conditions, I modify the original lena.bmp image by first loading it through the “imread()” function in the OpenCV package. After that, the next step is to get the image shape by “.shape”, I saved the value of each dimension in parameters “img_width”, “img_height” and “channels”, corresponding to the value of the 3-tuple result. After adequate modifications (which I would mention later), I use “cv2.imshow()” to present the resulting image, and this line is followed by a line “cv2.waitKey(0)” to keep the window open until an arbitrary key is pressed.

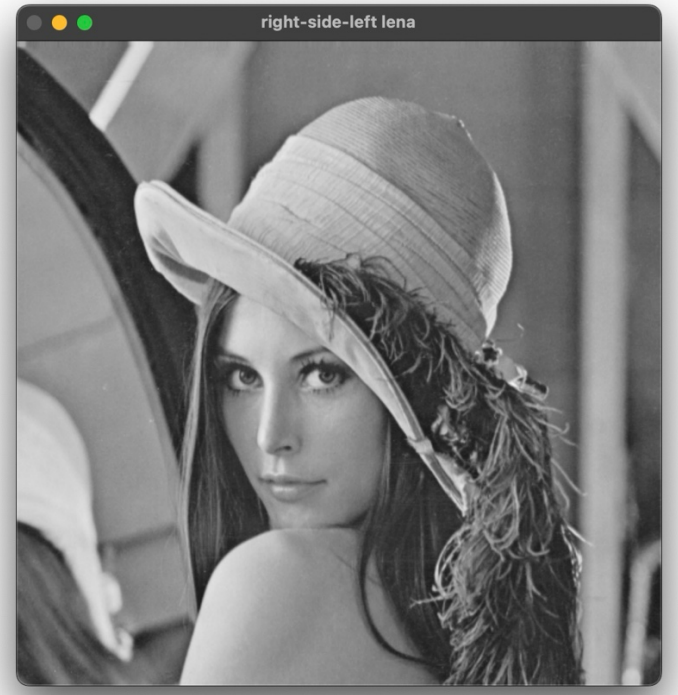
The original picture:

The explanation of each part is as below:

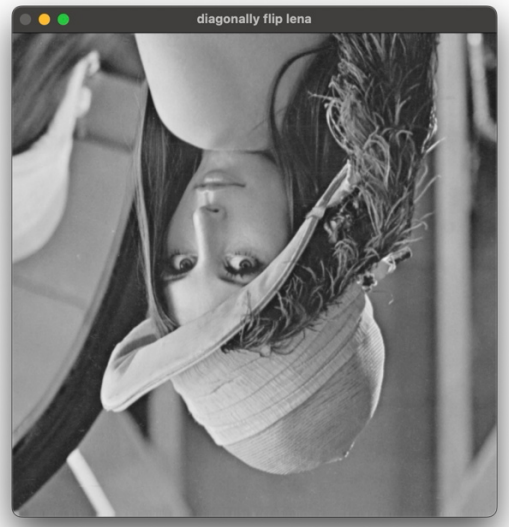
(a) To get the upside down image, I first create a ndarray that is of same shape as the original image. This initialization is done the same in all parts (except for (d)) . After that, I just simply access each row of the original image in an opposite manner, the last row is set to the first row (index 0) of the reverted image, ……, the first row is set to the last row of the reverted image, and the rows between are operated similarly. The reverted image is as follows:



(b) This part is to get the right-side-left image, after the same initialization, the approach is similar to part (a), except that we changed to access each column at a time, the last column of the original image is saved to the first column of the right-side-left image, …… and so on. The resulting image is:



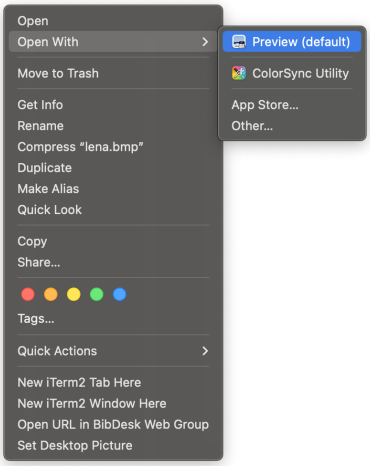
(c) In this part we're doing a diagonally flip, to achieve this goal, we use the result from part (a), each column of the upside down image is saved reversely, that is, we save the last column of the upside down image to the first column of the resulting diagonally



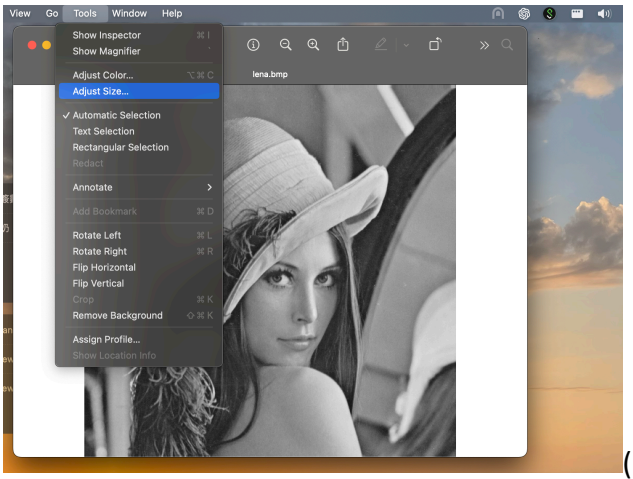
flipped image:

(d) In this part, the initialization is a little bit different, since we're rotating 45 degrees clockwise, the number of rows and columns should be switched in the resulting image, and this is done by setting the dimension values to be "(img_height, img_width, channels)". In this part, we access a pixel but not a column or row at a time, the pixels in the original image are accessed from left to right, from top to bottom, and saved from the rightmost column, from top to bottom:

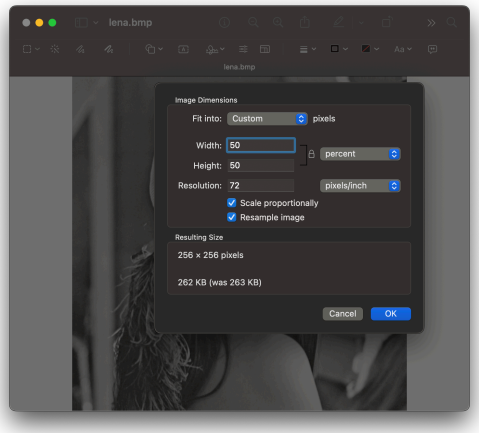
(e) This part we're going to shrink the image in half, and this is not presented by the code. I'll use the default mac application "Preview", we can shrink by using "tools >> adjust size", by setting the unit to be "percent" and the 50 for both width and height, the shrink

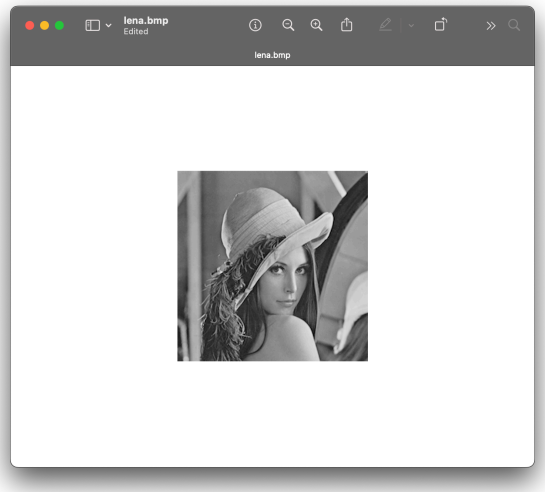


half process will be done.

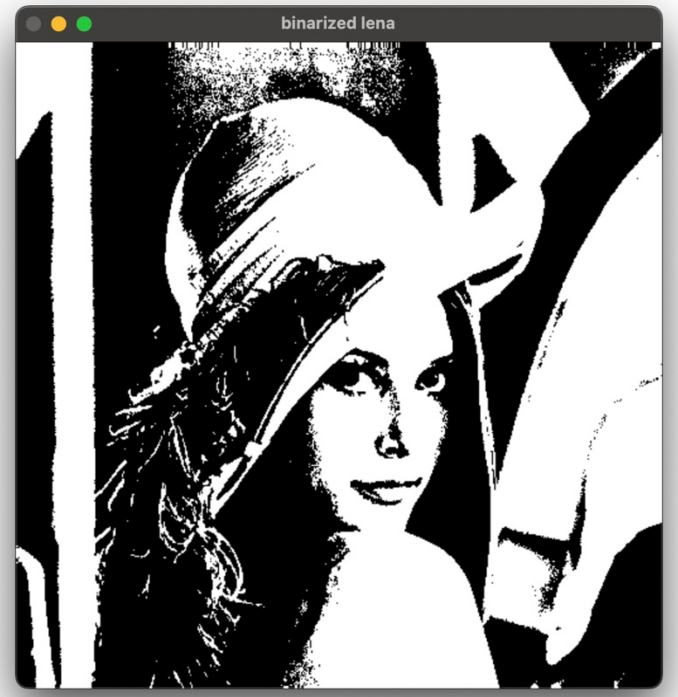


()





(f) The last part is to create the binarized image, we access the pixel from top-left corner to the bottom-right corner as usual, but in this part, we have to access the 3rd dimension, which represents the RGB channels. The value range from 0 to 255, and we set the value of the result image to 255 if the original value is bigger than 128, and set to 0 if the original value is equal or smaller than 128.



This is the rough description of my homework, more details, like some function definitions or explanations about things that are not that intuitive, can be found in the comments of the code.