Computer Programming HW1: Turtle Edge Printer

B07202020
NTUEE/NTUPhys
Hao-Chien Wang

October 6, 2019

The goal of this program is to read an image, find the edge of the image, and print it with the Turtle module. Several images are provided to test. You can also use your own image.

1 Instruction

There are four parameters available for the user in the beginning of the script:

1. `FIGURE_PATH` : set the path to the image.
2. `FILTER_STRENGTH` : Specify the strength of the filter. Higer value will make the filter discard the edges that are not so obvious. And since the total amoun of the line being drawn affect the time needed to print the picture, higher value also results in faster print.
3. `FAST_MODE` : Boolean. the width of the countour is usually larger than one pixel. If true, the *contour of contour* will be printed. Solid contour (made of a lot of lines) will be printed otherwise. Usually increase the speed more than 10 times if enabled.
4. `RESIZE_RATIO` : Resize the image. `New_size = old_size / RESIZE_RATIO`

2 Function description

The description of each function is also written in the comments of the script.

2.1 Modules imported

- PIL: module to read pictures into an array of RGB values
- NUMPY: convenient to process big arrays
- math: required for the `sqrt()` function

- sys: required to clear the last line of input
- turtle: output method

2.2 Functions for Image Processing

- `readImage( filePath, rerize = 1 )`: Read image and transform it into a 3D (height × width × RGB values) numpy array. Receives the path to the picture, ratio to resize (with default value 1), and return a 3D numpy array.
- `brightness_of_elements( imageArrayElement )`: formula to calculate luminance. Receives a 1x3 array consisting the RGB value of the pixel and return the luminance.
- `brightness( imageArray )`: Apply the `brigntness_of_elements` function to every pixel elements.
- `linify( imageArray, filterStrength, fastMode = False )` : use the Prewitt filter to find the edge. Receives the image array ,the filterStrength specifying the strength of the filter and a boolean to decide whether to enable fastMode. The function first apply the filter to each pixel and get `processedArray`. Then, it normalize the value of the array, and set the value above `filterStrength` to 1 and otherwise 0, which is the `finalArray`. Finally, the function results the `finalArray`.
- `getLineFig( filePath, filterStrength, fastMode, resize = 1 )` : encapsulation for functions processing the figure array. Receives the path to the image, filterStrength, fastMode (boolean) and resize ratio, call the functions above, then return the processed image array.

2.3 Functions for Turtle Drawing

- `goto( t, x, y )`: move the turtle without drawing anything. Receives the Turtle object and the target coordinates.
- `draw( figurePath, filterStrength = 1.5, fastMode = False, resize = 1, wait = True )` : encapsulation for drawing methods. Receives the path to the image, `filterStrength` (default value 1.5), fastMode (default value `False`), resize ratio, and a boolean `wait`(default value `True`)specifying whether to wait after finishing the job. It also show the progress in the console. It will keep refreshing the same line, but this feature might not work in every machine.

3 Examples

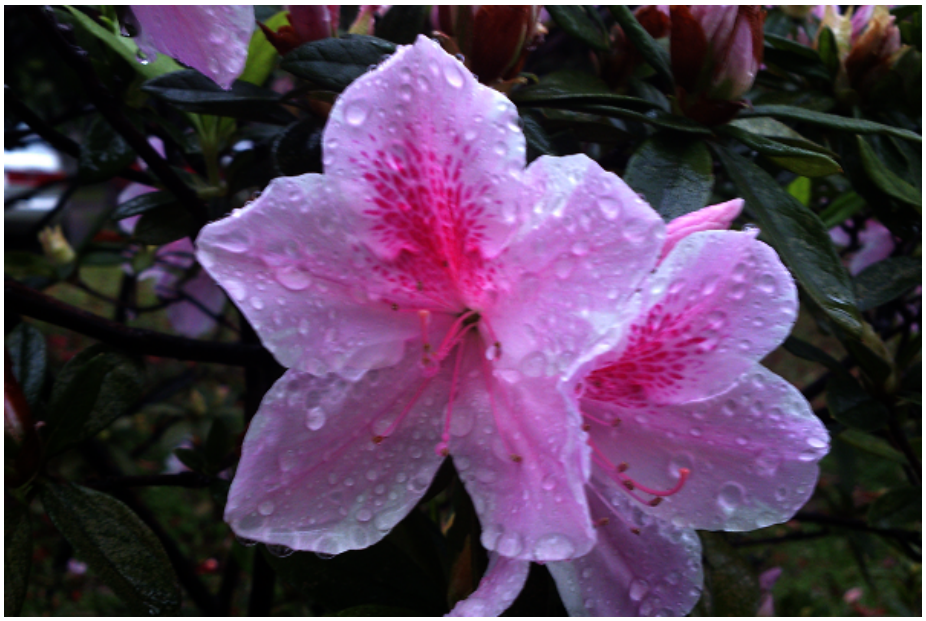

Figure 1: `flower.bmp`: The original image.

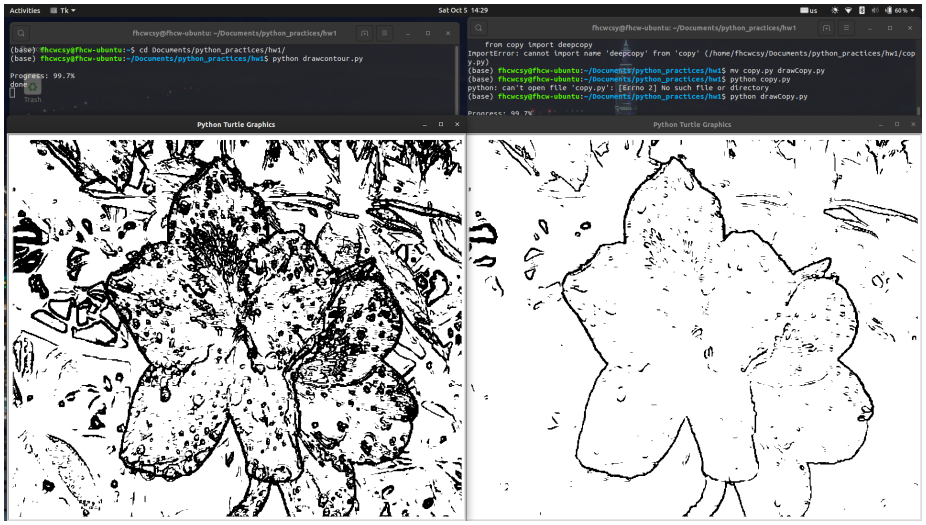

Figure 2: Left: `FASTMODE = False, FILTER_STRENGTH = 0, RESIZE_RATIO = 1`. Right: `FASTMODE = False, FILTER_STRENGTH = 1, RESIZE_RATIO = 1`

Figure 3: `FASTMODE = True`, `FILTER_STRENGTH = 1.5`, `RESIZE_RATIO=1`

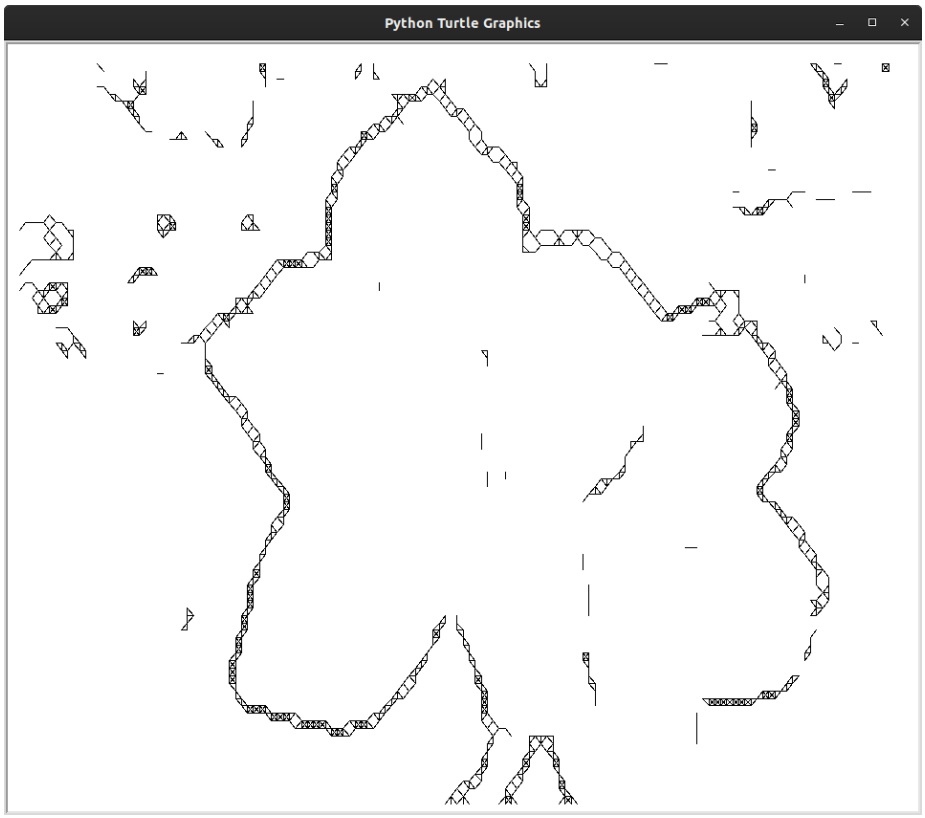

Figure 4: `FASTMODE = True`, `FILTER_STRENGTH = 1.5`, `RESIZE_RATIO=4`

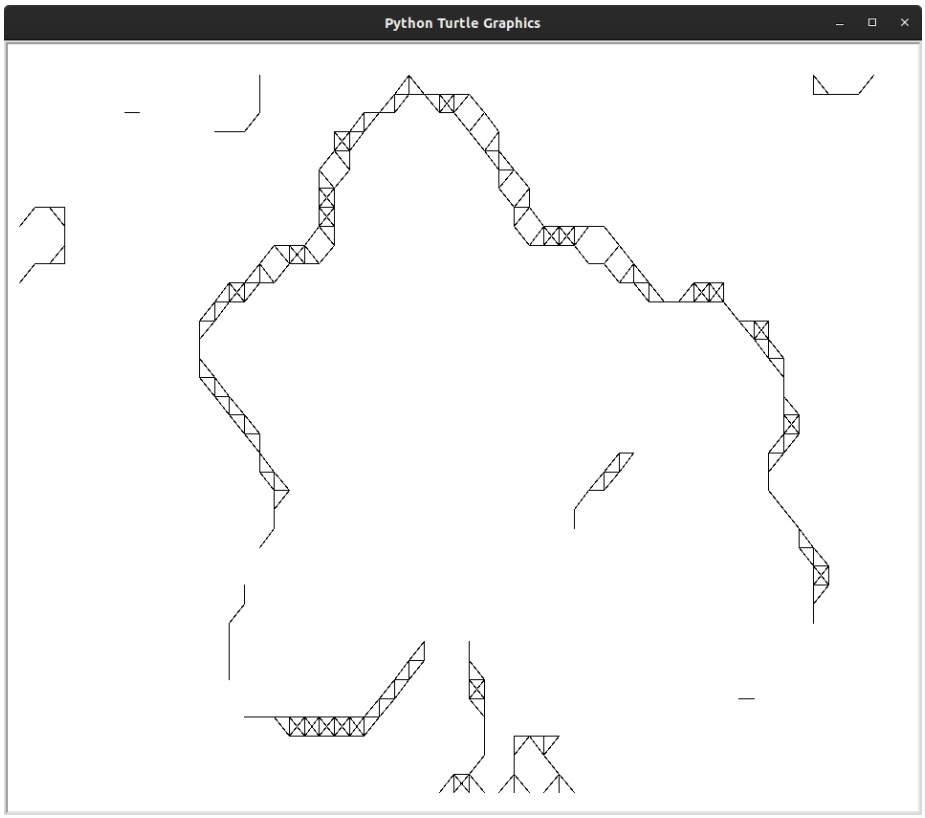

Figure 5: `FASTMODE = True`, `FILTER_STRENGTH = 1.5`, `RESIZE_RATIO=10`