CISC 490: Computer Vision **Instructor:** Sam Vinitsky

P1: Bob's Bad Bridges

Due: Wednesday, September 23 @ 11:59pm

Part 0: Introduction

Brrring, brrrring the phone rings, it's your good friend Robert! As you know, Robert is a vagabond photographer who roams the country taking photos for magazines, newspapers, and blogs. Robert recently got back from a very high profile assignment: photographing the famous covered bridges of Madison County, Iowa.

Unfortunately, Robert got a little... *distracted* while he was there, and so he forgot to check the quality of the photos before he got home. This is a big problem, since the photos didn't come out quite as he expected... but this is where you come in! Robert heard you were taking a course on Computer Vision, and figured you could help him out. Robert has sent over his bad pictures, as well as instructions for how to fix them. Robert's deadline is coming up soon, and he needs the pictures fixed by **Wednesday, September 23rd at 11:59pm**.

Helpful Information

As we saw during Monday's lecture, Python has tools for us to do simple image processing. For this assignment, you must use Python 3 (which you should already have installed from Project 0).

Included in the project folder is a Python file called "`utils.py`" that provides some basic functions to simplify reading, writing, and displaying images. In order for this library to work, you must install the `PIL` and `numpy` libraries. You can do so with Python's built-in `pip` functionality by running the following commands from the command line/terminal:

- `pip3 install Pillow`
- `pip3 install numpy`

Now that you have those libraries installed, you should be able to use `utils.py` from any Python file in the same folder as `utils.py`. Simply add the line "`import utils`" to the beginning of your Python file. The library "`utils.py`" providers the following function:

- `read_image_as_bw(path)`: takes as input a path to an image file, and returns a `numpy` array storing the image.

- `show_image(im_array)`: takes as input a `numpy` array representing an image, and pops up a window showing the image. Useful for debugging. (*Note: Please let me know if you have issues with this function on Windows!!*)
- `save_bw_image(im_array, path)`: takes as input a `numpy` array and a path, and saves the image as a file to that location. You must include the image file name and extension. (i.e. "`sam/pretty_picture.png`"

To use these functions, you must add the prefix "`utils.`" – i.e. `utils.read_image_as_bw(path)`, `utils.show_image(img)`, etc. (For example usage, see lecture notes from Monday, September 14).

Project Requirements

Because Robert is working for a commercial magazine with lots of annoying corporate rules, he has very specific requirements for what libraries you may (and may not) use for this project.

You may use any of the following libraries:

- `math`
- `numpy`
- Any modules in the `numpy` library, including `numpy.random`, or `numpy.whatever`
- `utils`, which is the library that Sam wrote and included in the project folder

You may NOT use any other libraries, including `scipy`. This means so other `import` statements. If there is a library you feel you absolutely cannot do this project without, ask about it on Piazza (but the answer will likely be "no").

Folder Structure

All images for this project are stored in a folder called "images". There are two sub-folders, called "input" and "output". The "input" folder stores all of the image files that Robert needs you to fix. You will need to put the cleaned photos in the "output" folder.

Questions + Answers

This assignment includes various questions, set aside in boxes, and labelled as "Q1", "Q2", etc. Include your answers in a file called "`answers.txt`", or "`answers.pdf`". (Text files are simpler, but if you want to use a word processor for fancier formatting, feel free – just convert the file to a PDF before submitting). Either way, make sure to label your answers with the question numbers...

Part 1: Flipping the Bridge

Somehow, Robert managed to hold his camera **upside-down** while taking a photo of a cool black bridge whose name he forgot to write down. He must have been distracted, or something... Robert needs your help to fix this horrible mistake.

Write a program called `flip_image_upside_down.py` that takes as input an image name (assumed to be in the "input" folder) and outputs that image flipped upside down (the output location should be in the "output" folder, with the same name as the input image).

Here are some examples of how Robert will run your program: (note that not all of these image files are actually in your "input" folder)

- `python3 flip_image_upside_down.py Black.jpg`
- `python3 flip_image_upside_down.py Truck.jpeg`
- `python3 flip_image_upside_down.py Francesca.png`

Once you have the program written, run it with the Black image as input. The flipped image should be written to "images/output".

Hint: The white fence should stay on the right hand side of the image.

Note: Since you are proficient with computers, I'm sure you know a thousand ways to flip an image upside down without writing a complicated Python program. Unfortunately, Robert has specifically requested that you use Python to write a program from scratch. (also, this is about learning to manipulate images, **not** learning how to Google "flip image upside down").

Part 2: Turning the Bridge Back

Robert is a fantastic photographer, but terrible at everything else. He accidentally scanned in his photo of the Cedar bridge *rotated 90 degrees*. Check out the photograph called “`Cedar.jpg`”, stored in the “input” folder. Bridges are usually horizontal, but this one is vertical! That’s gonna be a problem. But that’s where you come in!

Write a program called `rotate_image_counterclockwise.py` that takes as input an image name (assumed to be in the “input” folder) and outputs that image rotated 90 degrees counter-clockwise (the output location should be in the “output” folder, with the same name as the input image).

Here are some examples of how Robert will run your program: (note that not all of these image files are actually in your “input” folder)

- `python3 rotate_image_counterclockwise.py Cedar.jpg`
- `python3 rotate_image_counterclockwise.py OtherBridge.jpeg`
- `python3 rotate_image_counterclockwise.py Sadness.png`

Once you have the program written, run it with the Cedar image as input. The rotated image should be written to “images/output”.

Hint: Again, the white fence should be on the right hand side of the image.

Hint: What should dimensions of the new image be?

Note: Again, do not just use another program to rotate the image, actually write the Python program. Obviously.

Part 3: Noisy Bridges

Oh no! Since he was so distracted during his visit, Robert did not properly store his film. This resulted in several photographs that have been corrupted with *noise*:

- One of the images appears to have speckles all over it, like someone spilled the salt and pepper shakers (“`Hogback.jpg`”)
- One of the images appears to be extremely grainy (“`Roseman.jpg`”)
- One of the images appears to be salt-and-pepper speckled, but also grainy (“`Watson.jpg`”)
- One of the images is extremely messed up, but he’s hoping you can fix it anyways. *This one is extra credit.* (“`Extra.jpg`”)

***Q1**: What kind of noise is affecting the photo of the Hogback bridge? How did you figure that out?*

***Q2**: What kind of noise(s) are affecting the photo of the Roseman bridge? If you can figure this out, how did you do it? If you can’t tell for certain, why not?*

***[EC] Q3**: What kind of noise(s) are affecting the photo of the "`Extra.jpg`" bridge? If you can figure this out, how did you do it? If you can’t tell for certain, why not? [Extra Credit]*

In order to denoise these images, you will implement three of the de-noising algorithms discussed in lecture: mean filter, median filter, and (for *extra credit*), Gaussian blur. You should write the following programs:

- `mean_filter.py`
- `median_filter.py`
- `gaussian_blur.py` [EXTRA CREDIT]

Each program should take *three* inputs: (1) the path to the input image, (2) the desired output path, and (3) the size of the filter to be used (which will always be an odd integer). Here, the filters will all be square, and so the size indicates both the width and the height. For example, a size “3” would give you a 3x3 filter...

Once you have written these programs, use them to denoise the three noisy images. (Hogback, Roseman, and Watson). Robert needs the photo quality to be crisp, so make sure to introduce *as little blur as possible.*

You will need to test each photo with each of the noise removal programs in order to determine which is most effective at removing the most noise while introducing the least blur. You will also need to play around with parameter settings to find the best blur/noise tradeoff for each filter.

Store your best denoised images in the "output" folder `<ORIGINAL_NAME>_clean.jpg`, with `<ORIGINAL_NAME>` replaced with the original file name. There should be a "`Hogback_clean.jpg`", "`Roseman_clean.jpg`", and "`Watson_clean.jpg`" (and for extra credit, "`Extra_clean.jpg`".

Hint: You are allowed to mix and match these filters as you like in order to remove the noise in the more degraded photographs...

Q4*: For the "Hogback" photo...*

- *a) What was the best filter to remove the noise? (while blurring the photo as little as possible)*
- *b) Why do you think that filter worked best?*
- *c) Which "size" caused the least blur, while still removing the noise?*
- *d) For the cleanest image, how much noise was left? How much detail was lost to the blur?*

Q5*: For the "Roseman" photo...*

- *a) What was the best filter to remove most of the noise? (while blurring the photo as little as possible)*
- *b) Why do you think that filter worked best?*
- *c) Which "size" caused the least blur, while still removing the noise?*
- *d) For the cleanest image, how much noise was left? How much detail was lost to the blur?*

Q6*: For the "Watson" photo...*

- *a) Was there a single filter that removed most of the noise?*
- *b) If not, why do you think that is?*
- *c) How did you remove the noise from this image?*
- *d) What "size" values worked best for each filter you used?*
- *e) For the cleanest image, how much noise was left? How much detail was lost to the blur?*

[EC - Q7]*: For the "Extra" photo...*

- *a) Were you able to remove most of the noise from the image? If so, how? If not, why not?*
- *b) For the cleanest image, how much noise was left? How much detail was lost to the blur?*

Part ∞: What to turn in

On Canvas, turn in a zip folder called `<last_name>_p1.zip` (but replace "`<last_name>`" with your last name). In this zip folder, include the following file(s):

- `flip_image_upside_down.py`
- `rotate_image_counterclockwise.py`
- `mean_filter.py`
- `median_filter.py`
- `gaussian_blur.py` [EXTRA CREDIT]
- `answers.txt` or `answers.pdf`

You should also include the "images" folder, which must include an "output" folder with you best cleaned photos: "`Hogback_clean.jpg`", "`Roseman_clean.jpg`", and "`Watson_clean.jpg`" (and for extra credit, "`Extra_clean.jpg`").