

Homework 1 Written Questions

Instructions

- 2 ethical implications questions, which will be expanded on in discussion sections.
- 5 technical questions.
- Include code, images, and equations where appropriate.
- Please make this document anonymous.
- When you are finished, compile this document to a PDF and submit it directly to Gradescope. On upload, **Gradescope will ask you to assign question numbers to your pages**. Making each question end with a page break after your answer is a good way to ease this process.
- This assignment is **fixed length**, and the pages have been assigned for you in Gradescope. As a result, **please do NOT add any new pages**. We will provide ample room for you to answer the questions. If you *really* wish for more space, please add a page *at the end of the document*.
- **We do NOT expect you to fill up each page with your answer**. Some answers will only be a few sentences long, and that is okay.

Ethical Implications

Q1: Hybrid images appear as normal images until the viewer changes distance; they blur the line between authentic and inauthentic. As technology advances, evaluating authenticity becomes increasingly difficult. In 1990, *New York Times* photography critic Andy Grundberg [stated that](#): "In the future, readers of newspapers and magazines will probably view news pictures more as illustrations than as reportage, since they can no longer distinguish between a genuine image and one that has been manipulated."

Do you think we've reached this point and why? What manipulations do you think are permissible for an image to be genuine, if any? Based on this, please list at least three potential solutions (scientific, journalistic, legal, etc.) to address mistrust in photographic media. [6-7 sentences]

A1: Your answer here.

Q2: For your CS1430 final project, you decide to build an [AI dubbing program](#) that can alter an actor's lip movements to make film dubbing more effective. Your project is successful, and creates perfect videos that could depict anyone appearing to say anything.

Please list at least three potential consequences if your project is misused. What are three things that could be done (scientifically, legally, etc.) to try and avoid these problems? [5-6 sentences]

A2: Your answer here.

Technical Questions

Q1: Image convolution, a type of image filtering, is a fundamental image processing tool that you will use repeatedly throughout the course.

- (a) *Explicitly describe* the 3 main components of image convolution: (5-10 sentences)
 - (i) input
 - (ii) transformation (how it happens)
 - (iii) output
- (b) Why is image convolution important in Computer Vision? Which applications does it allow? (5-10 sentences)

A1: Your answer here.

Q2: Correlation and convolution are both filtering operations we can perform to extract information from images.

- (a) What is the difference between convolution and correlation? (3-5 sentences)
- (b) Construct a scenario which produces a different output between both operations. Include the kernel you used and your image results. Use your understanding of convolution and correlation to explain the outputs. (5-10 sentences)

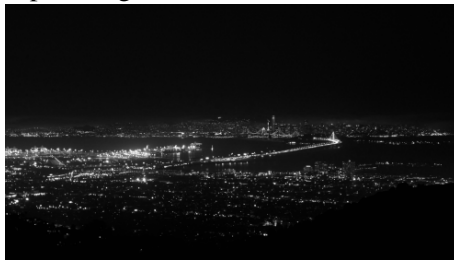
Please use `scipy.ndimage.convolve` and `scipy.ndimage.correlate` to experiment!

A2: Your answer here.

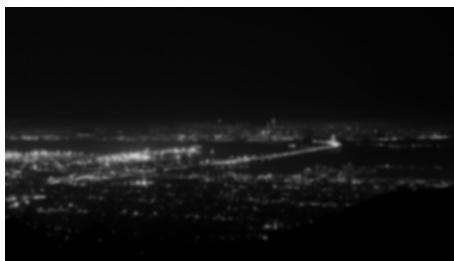
Q3: Please review the 'Thinking in Frequency Part 1' lecture slides on aliasing, and think about how we might anti-alias with a low-pass filter. Related high-pass filters also exist. For (a–c), which kind of filter does the kernel represent? For (d–e), which filter produced the output images?

- (a) $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ ☐ High pass
☐ Low pass
☐ Neither
- (b) $\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$ ☐ High pass
☐ Low pass
☐ Neither
- (c) $\begin{bmatrix} -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{8}{9} & -\frac{1}{9} \\ -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \end{bmatrix}$ ☐ High pass
☐ Low pass
☐ Neither

(d) Input image:

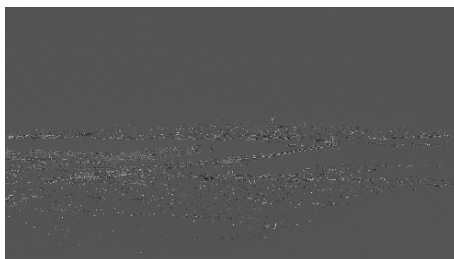


Output image 1:



- ☐ High pass
☐ Low pass

(e) Output image 2:



- ☐ High pass
☐ Low pass

(f) Which of the following statements are true? (Check all that apply).

- ☐ High pass filter kernels will always contain at least one negative number
☐ A Gaussian filter is an example of a low pass filter
☐ A high pass filter is the basis for most smoothing methods
☐ In a high pass filter, the center of the kernel must have the highest value

Q4:

- (a) How does computation time vary with filter sizes from 3×3 to 15×15 (test odd and square sizes, i.e. 3×3 , 5×5 , 7×7 , etc.), and with image sizes from approximately 0.25 to 8 megapixels? To find out fill out the below stencil to graph, for each of the above filter sizes, the correlation between an image size (x-axis) and time to convolve/correlate (y-axis) that image— each filter size should be its own line in the multi-line plot.

The stencil code imports the libraries you will need, but to understand how to use them you must look at the documentation.

- convolve/correlate - [scipy.ndimage.convolve](#) or [scipy.ndimage.correlate](#)
- rescale - [skimage.transform.rescale](#)
- resize - [skimage.transform.resize](#)
- rescale vs resize – an example [here](#)

Add your graph as well as a brief description of what your graph demonstrates, below.

Note A megapixel is 1,048,576 (2^{20}) pixels (1024×1024), or sometimes also 1,000,000 pixels (especially if you manufacture cameras). Megapixels is often shortened to MP or MPix.

Image: [RISDance.jpg](#) (in the .tex directory).

```
import time
import matplotlib.pyplot as plt
from skimage import io, img_as_float32
#use to rescale+resize image
from skimage.transform import rescale, resize
#use to convolve/correlate image
from scipy.ndimage import correlate

#This reads in image and converts to a floating point format
# 1) TODO - replace PATH with the actual path to the
#      downloaded RISDance.jpg image linked above
image = img_as_float32(io.imread('PATH'))

# 2) TODO - change the image size so it starts
#      at 8MPix (calculated as height x width)
#      use one of the imported libraries
original_image =

# 3) TODO - iterate through odd numbers from 3 to 15
#      (inclusive!!) these will represent your filter sizes
#      (3x3, 5x5, 7x7, etc.), for each filter size you will...
for kernel_size in range():

    #because for each loop you are resizing your image, you
    #want to start each loop w/the original image size
    shrinking_image = original_image

    #these lists will hold the values you plot
```



```
image_sizes = [] #x axis
times = [] #y axis

#while image size is bigger than .25MPx
while(shrinking_image.size > 250000):

    # 4) TODO - create your kernel. Your kernel can hold
    # any values, as the kernel values shouldn't
    # affect computation time. Avoid using np.zeros to
    # create your kernel, as this messes up the intended
    # output graph. The size of the kernel
    # must be kernel_size x kernel_size
    kernel =

    #5) TODO - reduce your image size. You can choose by
    # what increments to reduce your image.
    shrinking_image =

    #gets the current time (in seconds)
    start = time.time()

    # 6) TODO - use one of the imported libraries to do
    # your correlation/convolution on the image. You can
    # choose which operation to perform.

    #gets the current time (in seconds)
    end = time.time()

    #7) TODO - figure out what values to append, and
    # append them here
    image_sizes.append()
    times.append()

    #each filter size will be plotted as a separate line, in
    #a multi-line 2-dimensional graph
    plt.plot(image_sizes, times, label=str(kernel.size))

#plot
plt.xlabel('image size (pixels)')
plt.ylabel('operation time (seconds)')
plt.legend(title="filter sizes (pixels)")
plt.show()
```

- (b) Do the results match your expectation given the number of multiply and add operations in convolution? (3-5 sentences)

A4: Your graph and answer here.

Q5: The coding portion of this homework requires the use of the library *numpy*, which provides fast computation with large multi-dimensional arrays and matrices. Here are some small exercises on basic numpy operations to get you started! Write *one* numpy function to complete each of the following tasks.

Note that numpy is usually imported as

```
import numpy as np
```

at the top of the code file. You can then call numpy functions with

```
np.function_name()
```

You are encouraged to test out your answers by creating your own python program, importing numpy and calling and printing the results of your solutions! Some numpy functions you might find useful are [np.squeeze](#), [np.expand_dims](#), [np.clip](#), [np.pad](#), and [np.zeros](#).

- a. Create an array of shape (320,640) filled with zeros.
- b. Given a variable called *img* of shape (1, 1, 320, 640), create a new 2D array variable of shape (320, 640) from *img*, i.e. remove all the 1-sized dimensions.
- c. Conversely, say you have an array variable *img* of shape (320, 640), create a new 2D array variable of shape (1, 320, 640) from *img*, i.e. add a dimension.
- d. Clip the image array, *img*, so all its values lie within the range [-0.5, 0.5].
- e. With an RGB-image array, *img*, of shape (320, 640, 3), retrieve the blue channel of the image while preserving all of *img*'s dimensions and values.
- f. With an RGB-image array, *img*, of shape (320, 640, 3), retrieve the red and blue channels of the image.
- g. Pad an **RGB** (remember, this means the image has 3 channels) image array, *img*, with two zeros on either side of each row (the left and right edges of the image) and 3 zeros values on either side of each column (the tops and bottom edges of the image). Don't add zeros to the front or back (the color channel dimension).

A5: Your answer here. Remember, write *one* numpy function to complete each of the tasks—this includes operators like [] and ..

Feedback? (Optional)

Please help us make the course better. If you have any feedback for this assignment, we'd love to hear it!