CSci 4270 and 6270
Computational Vision,
Fall Semester, 2017-18
Homework 1
Due: Wednesday, September 27, 5 pm

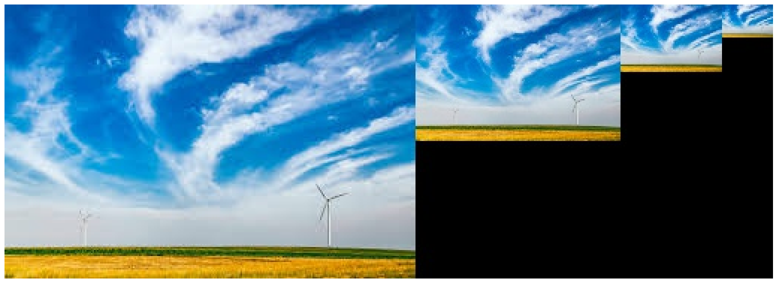
Please refer back to HW 0 for guidelines.

For Problem #2, please submit a separate pdf file giving your answer. You must typeset (not hand-print) the answer. I strongly recommend LaTeX to create nice, clear mathematics. Use the websites OverLeaf or ShareLaTeX to do this. There will be instructions on how to submit the final pdf on the Submitty page.

Problems 3 and 4 both require you to process a set of points represented as a 2d NumPy array. You should be able to do this without any for loops that explicitly iterate over the points, and will only earn full credit if you do. However, the penalty if you find yourself needing to write complete loops will be about 5-10% of the grade on that problem.

Problems

1. (**15 points**) Write a Python program that takes a single image as input and creates a new image that stores the original, a half-sized copy, a quarter-sized copy, etc., side-by-side. Here is an example to help make it clear:



   The original image is all the way to the left. The half-sized image (in each dimension) is the right of the original, aligned at the top, with no extra space between the copies. The quarter-sized image is to the right of the half-sized image, etc. Use integer division to determine actual sizes as you reduce the images. Stop when the inserted image copy would have fewer than 20 rows or fewer than 20 columns. In other words a 19x30 copy would not be inserted but a 30x20 copy would. Use `cv2.resize` to create the copies. All pixels in the output image that are not covered by a copy of the input images should be 0. The command line should be simply

```
p1_pyramid in_img out_img
```

   In addition to writing the image, output text that demonstrates details of what your program is doing. Specifically, for each image copy added to the output image this should be the upper left row and column where the new image is inserted and the shape of the image inserted. At the end, output the final shape of the output image. Closely follow the example provided.

2. (**15 points**) Let $\mathbf{A}$ be an $m \times n$ matrix of real values, with $m \geq n$. What is the relationship between the SVD $\mathbf{A}$ and the eigen decomposition of $\mathbf{A}^\top \mathbf{A}$? Justify your answer. You will

need to know that the eigenvalues of a matrix are unique up to a reordering of eigenvalues and vectors, so you may assume they are provided in any order you wish. By construction, the singular values are non-increasing. (The eigenvectors / singular vectors are unique up to a reordering only if the eigenvalues / singular values are unique.)

3. (**30 points**) Given a set of point coordinates in $\mathbb{R}^2$, compute and output the following:

   (a) The minimum and maximum x and y values.

   (b) The center of mass (average) x and y values.

   (c) The axis along which the data varies the least and the standard deviation (call it s_{min}) of this variation. Note that the standard deviation is the square root of the eigenvalue. Resolve ambiguity in the direction by ensuring that the x component is positive. If this component is 0, make sure the y component is positive.

   (d) The axis along which the data varies the most and the standard deviation (call it s_{max}) of that variation. Resolve the ambiguity here in the same way as the previous problem.

   (e) The closet point form of the best fitting line (through the original data). Ambiguity in the representation should be resolved such that ρ is positive. (You need not check the case of the center of mass being exactly (0,0), which would introduce a further ambiguity.) You should just output the values of ρ and θ, with θ in radians.

   (f) The implicit form of the line. Just output a, b and c. Ensure that $a^2 + b^2 = 1$. Resolve sign ambiguity by ensuring that $c = -\rho$.

   (g) A decision about the shape that best describes the data: a line or an ellipse. This is a line if $s_{\text{min}} < \tau s_{\text{max}}$ and an ellipse otherwise. (Note that τ is an input parameter.)

   All output floating point values must be accurate to three decimal places.

   Finally, output a MatPlotLib plot **saved as an image** containing a scatter plot of the points and of the center of mass. Add to this the best fitting line. Use the functions `scatter`, `plot`, `axes` and `savefig`. Make sure the units both vertically and horizontally are at least roughly the same. I am providing less guidance here on the result than I did on the text output, so make it look good as you see fit. I will not be terribly picky.

   The command line will be

   ```
   python p3_shape.py points.txt tau outfig
   ```

   where `points.txt` is a text file containing the points, `tau` is the value of τ described above and `outfig` in an image file name where you are to write the output figure.

4. (**25 points**) Implement the RANSAC algorithm for fitting a line to a set of points. We will start our discussion with the command line.

   ```
   python p4_ransac.py points.txt samples tau [seed]
   ```

   where `points.txt` is a text file containing the points in exactly the same form as the previous problem, `samples` is a positive integer indicating the number of random pairs of two points to generate, `tau` is the bound on the distance from a point to a line for a point, and `seed` is an optional parameter giving the seed to the random number generator.

   After reading the input, if the seed is provided, your first call to a NumPy function must be

```
  np.random.seed( seed )
```

otherwise, do not call the `seed` function. Doing this will allow us to create consistent output. For each of `samples` iterations of your outer loop you must make the call

```
  sample = np.random.randint( 0, N, 2 )
```

to generate two random indices into the points. If the two indices are equal, skip the rest of the loop iteration (it still counts as one of the samples though). Otherwise, generate the line and run the rest of the inner loop of RANSAC. Each time you get a new best line estimate according to the RANSAC criteria, print out the following values, one per line, with a blank line afterward:

- the sample number (from 0 up to but not including `samples`)
- the indices into the point array (in the order provided by `randint`),
- the values of a, b, c for the line (ensure $c \leq 0$ and $a^2 + b^2 = 1$), and
- the number of inliers.

At the end, output a few final statistics on the best fitting line, in particular output the average distances of the inliers and the outliers from the line. Keep all of your output floating point values accurate to three decimal places.

In the interest of saving you some work, I've not asked you to generate any plots for this assignment, but it would not hurt for you to do so just to show yourself that things are working ok. For similar reasons, no least-squares fit is required at the end — no need to repeat the previous problem.

5. (**Grad Credit: 10 points**) Some digital cameras provide visual warning of pixels that are outside the dynamic range of the image, i.e. they are pure white (255,255,255) or pure black (0,0,0). Because these pixels are on the edge of what the camera can record, it is not known what their "true" intensities are. Write a Python program that mimics this effect. It should take a color image as input and produce as output a new color image where all pure white pixels are converted to pure red (255,0,0) and all pure black pixels (0,0,0) are converted to pure blue (0,0,255).

   The program will be run from the command-line as

   ```
     python p5_highlight.py img_in img_out
   ```

   where `img_in` is the name of the input image file and `img_out` is the name of the output image file. The program should also output exactly two lines of text giving the number of pure white pixels and the number of pure black pixels. As a made up example, this output might be

   ```
   white 164
   black 37
   ```