EECS 442 Computer Vision: Homework 1

Instructions

- This homework is **due at 11:59:59 p.m. on Thursday January 31st, 2019**.
- The submission includes two parts:
  1. **To Gradescope:** a `pdf` file as your write-up, including your answers to all the questions and key choices you made.
     You might like to combine several files to make a submission. Here is an example online link for combining multiple PDF files: https://combinepdf.com/.
  2. **To Canvas:** a `zip` file including all your code for **Part 3**.
     If you are enrolled shortly before the deadline and have no access to Canvas at the deadline, you can send the codes by email. Sending by email is allowed **only** for HW1.
- The write-up must be an electronic version. **No handwriting, including plotting questions.** LaTeX is recommended but not mandatory.

1 Matrix Mathematics Review [15 pts]

(a) (2 pts) $A = \begin{pmatrix} 3 & 0 & 2 \\ 1 & 5 & 4 \end{pmatrix}$, $B = \begin{pmatrix} 7 & 2 \\ 1 & 0 \\ 3 & 9 \end{pmatrix}$. Calculate AB.

(b) (2 pts) Given $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$, the entry in the i-th row and j-th column of A can be denoted as a_{ij}. Similarly, b_{ij} is the i-th row and j-th column of B.
Prove $(AB)^T = B^T A^T$ by comparing every entry of $(AB)^T$ and $B^T A^T$.

(c) (2 pts) We define p-norm of a vector $||\mathbf{x}||_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$ where $p \geq 1$ is a real number and x_i is the i-th entry of $\mathbf{x}$.
$v = [2, 6, 3]^T$. Calculate the corresponding 1-norm and 2-norm.

(d) (2 pts) We define the dot product of two products $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T \mathbf{y}$.
How can the 2-norm of a vector be written as a dot product?

(e) (3 pts) Derive all the eigenvalues and eigenvectors for $\begin{pmatrix} 2 & 0 & 0 \\ -1 & 3 & 3 \\ 6 & -6 & -6 \end{pmatrix}$.

(f) $R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ rotates points in the 2D space counterclockwise through an angle θ about the origin of the Cartesian coordinate system.
Prove: (i) (2 pts) $R(\theta)^{-1} = R(-\theta)$; (ii) (2 pts) $R(\alpha+\beta) = R(\alpha)R(\beta)$.

2 Camera Projection Matrix [25 pts]

In computer vision, a camera matrix or (camera) projection matrix is a 3×4 matrix which describes the mapping of a pinhole camera from 3D points in the world to 2D points in an image.
Let $\mathbf{x} = [x_1, x_2, x_3, 1]^T$ be a representation of a 3D point in *homogeneous coordinates*, and let $\mathbf{y} = [y_1, y_2, 1]^T$ be a representation of the projection of this point in the pinhole camera. Then the following relation holds

$$\mathbf{y} \equiv P\mathbf{x}$$

where P is the camera matrix and the $\equiv$ sign means that the two sides are equal up to a non-zero scalar multiplication, *i.e.* $\mathbf{y} = \lambda P\mathbf{x}$ for some constant λ.

(a) Now we set up a Cartesian coordinate system. In the coordinate, we have three line segments:

- A red line segment, endpoints: $[1.00, 1.00, 1.00]^T$ and $[2.15, 2.15, 2.15]^T$;
- A blue line segment, endpoints: $[-1.00, 3.25, -0.25]^T$ and $[-1.00, 4.00, -1.00]^T$;
- A green line segment, endpoints: $[1.00, 1.00, 0.50]^T$ and $[1.00, 4.00, 2.00]^T$.

Try to solve the following questions:

(i) (10 pts) **Plot** the projection of the three line segments with the given camera projection matrices:
$P_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, P_2 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$.
Hint and instructions:

- You should include two projection results, each of which has projection of the three line segments in red, blue, and green respectively.
- For a point $\mathbf{x} = [x_1, x_2, x_3, 1]^T$ in 3D space, the projection $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} \equiv P\mathbf{x} = \begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix}$.
Then the Cartesian representation is $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} y_1'/y_3' \\ y_2'/y_3' \end{pmatrix}$.
- The projection of a line segment must be a line segment or a point. You can use the projection of the two endpoints to fix the projection of a line segment.

(ii) (5 pts) If you are given the projection derived from P_1, can you recover the three 3D line segments? In other words, are the line segments unique to get the corresponding projection results? Assume that the projection matrix P_1 is always known.
If the answer is yes, give a brief explanation; if no, describe another set of line segments which has the same projection.

(b) Think about a cube with the length of each edge as 2. The cube's center is at the origin of the Cartesian coordinate system and every edge is parallel to one of the three axes of the Cartesian coordinate system,

$i.e.$ each of the vertices is at $[a_1, a_2, a_3]$ where $|a_1| = |a_2| = |a_3| = 1$.

(5 pts) **Plot** the projection with the projection matrix $P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{pmatrix}$.

(5 pts) Does this projection preserve parallelism? In other words – if the 3D lines are parallel in 3D, are they parallel after projection? If yes, explain why. If not, **calculate** the homogeneous representation (in 2D) of two edges that are parallel in 3D but not in 2D. What is the homogeneous coordinate of their vanishing point on the projected plane?

Hints and instructions:

- Assume the cube is transparent. Thus, all of the 8 vertices and 12 edges are involved in the plotting. You can use the solid lines for all the edges no matter whether the edges are behind some face(s).
- Refer to *Benefits of Homogeneous Coords* in Lecture 2 slide to calculate the intersection of lines and the line through two points.

3 Prokudin-Gorskii: Color from grayscale photographs [60 pts]

Overview In computer vision, we use multi-dimensional matrices (tensors) to represent images. An example might be a color image whose dimensions are 1920×1080×3 (height × width × color channels). In this programming problem, we will explore image tensors and match misaligned color channels with one another. You can find a zipfile with everything you need for the assignment on the course website (https://web.eecs.umich.edu/~fouhey/teaching/EECS442_W19/index.html) in the HW1 folder. A basic python file skeleton with some imports has been included in the HW1 folder for you to use (if you so choose).

Background You are tasked with implementing the dream of Russian photographer, Sergei Mikhailovich Prokudin-Gorskii (1863-1944), via a project invented by Russian-American vision researcher, Alexei A. Efros (1975-present)[1]. Sergei was a visionary who believed in a future with color photography (which we now take for granted). During his lifetime, he traveled across the Russian Empire taking photographs through custom color filters at the whim of the czar. To capture color for the photographers of the future, he decided to take three separate black-and-white pictures of every scene, each with a red, green, or blue color filter in front of the camera. His hope was that you, as a student in the future, would come along and produce beautiful color images by combining his 3 separate, filtered images.

Task 1: Combine (10 pts) We will provide you with a folder of Prokudin-Gorskii's black-and-white (grayscale) image composites (`prokudin-gorskii/` in the assignment zip). Each composite (alternatively triple-framed image or triptych) contains three grayscale photos preserved from the early 1900s. The composite looks like a three panel vertical comic strip, with each grayscale photo in the composite positioned vertically above one another. These photos represent images captured with a blue, green, and red filter. Choose a single composite from this folder (your favorite) and write a program in Python that takes the three grayscale panels and simply stacks them across the third color channel dimension to produce a single, colored image. We expect this stacked photo to look wonky and unaligned- fixing this is what you will do in Task 2. Make sure to save your images as RGB instead of BGR and include them in your report.

[1]Credit for this assignment goes to Alexei http://inst.eecs.berkeley.edu/ cs194-26/fa18/

Specifically: Write a function that loads a grayscale tripled-framed image from `prokudin-gorskii/` with something like `skimage.io.imread()`, chops it vertically into thirds, then saves a color image with each third as the correct color channel. Save the output colored image in your report.

Task 2: Alignment (25 pts) As you will have noticed, the photos are misaligned due to inadvertent jostling of the camera between each shot. Your second task is to fix this. You need to search over possible pixel offsets in the range of [-15, 15] to find the best alignment for the different R, G, and B channels. The simplest way is to keep one channel fixed, say R, and align the G and B channels to it by searching over the offset range both horizontally and vertically. Pick the alignment that maximizes a similarity metric (of your choice) between the channels. One such measure is dot product, i.e, R·G. Another is normalized cross-correlation, which is simply the dot product between the L2 normalized R and G vectors. After writing this function, run it on all of the images in `prokudin-gorskii/` and also on `'efros_tableau.jpg'`, so Professor Efros can have his photo restored to color. Include these aligned images and the offsets in your report. For full credit, your report needs to include properly aligned images- find a similarity metric that will accomplish this.

Specifically: Write a function to align the 3 channels of the image produced by Task 1. This function should output the (x,y) offsets required for shifting two of the color channels with respect to third. The third channel might then have a (0,0) offset. Save the newly aligned images from `prokudin-gorskii/` and `'efros_tableau.jpg'` in your report, along with the offsets for each color channel. Write a sentence or two describing what your algorithm is doing.

Task 3: Pyramid (25 pts) For very large offsets (and high resolution images), comparing all the alignments for a broad range of displacements (e.g. [-30, 30]) can be computationally intensive. We will have you implement a recursive version of your algorithm that starts by estimating an image's alignment on a low-resolution version of itself, before refining it on higher resolutions. To implement this, you will build a two-level image pyramid. To do this, you must first scale the triple-frame images down by a factor of 2 (both the width and height should end up halved). Starting with your shrunk, coarse images, execute your alignment from Task 2 over the following range of offsets [-15 15]. Choose the best alignment based on your similarity metric and treat it as the new current alignment. Then in the full resolution images, use this new current alignment as a starting place to again run the alignment from Task 2 in a small range [-15 15]. Run this Pyramid task on the `'seoul_tableau.jpg'` and `'vancouver_tableau.jpg'` images. If your course project goes well, maybe you will see those sights in real life at either ICCV or NeurIPS!

Specifically: Write a function (recursive or not) that shrinks each grayscale image in the triptych to a quarter of its original size (half the width and half the height) and then aligns them with Task 2 in the range of [-15, 15] at the new resolution (this would be equivalent window to [-30, 30] in the full resolution). In the report, write a sentence or two describing what your algorithm is doing. Report the intermediate shift (at the coarse resolution), the next shift at the full resolution, and what the overall total shift was that includes both of these. Save the newly aligned an `'seoul_tableau.jpg'` and `'vancouver_tableau.jpg'` in color in your report. (Hint: if you're struggling, use a different color channel as your anchor!)

Report You must submit a report that includes the offsets, color output images, and description required above. Your description should be such that a reader could implement what you've done after reading your report. Make sure to submit your code separately on Canvas, or email it to Professor Fouhey if you are still on the waitlist.

Libraries You are not to use functions that entirely solve these problems for you. However, libraries like `numpy`, `scikit-image`, `matplotlib`, and `opencv` are allowed and include many useful functions you should use. Specifically, consider using the functions to solve the above tasks:

- `skimage.io.imread()`
- `skimage.io.imsave()`
- `skimage.transform.resize()`
- `numpy.roll()`
- `numpy.pad()`
- `numpy.sum()`
- `numpy.transpose()`
- `numpy.dstack()`

Extra Credit If you implement some interesting alignment techniques, you can earn up to 10 pts extra credit. Try to do even better than the restoration done by the Library of Congress (see: http://www.loc.gov/exhibits/empire/making.html)! To earn extra credit, include a description of your method in your report and improved versions of the composites in `prokudin-gorskii/`. Here are some ideas:

- Automatically crop borders that are not part of the image itself. This involves actively detecting what part of the grayscale image is just an artifact of poor scanning and removing that. (2 pts)
- Automatic contrasting. You can rescale image intensities so that the darkest pixel in the image (on the darkest color channel) is 0 and the brightest pixel is 1 (on its brightest color channel). Try different contrasts and see how it affects perceived image quality. (2 pts)
- Automatic white balance. This involves estimating the illuminant and changing the colors to cancel out this illumination. Step 1 is difficult in general, while step 2 is simple (see the Wikipedia page on Color Balance and section 2.3.2 in the Szeliski book). There exist some simple algorithms for step 1, which don't necessarily work well – assume that the average color or the brightest color is the illuminant and shift those to gray or white. (3 pts)
- Better color maps. There is no reason to assume (as we have) that the red, green, and blue lenses used by Produkin-Gorskii correspond directly to the R, G, and B channels in RGB color space. Try to find a mapping that produces more realistic colors (and perhaps makes the automatic white balancing less necessary). (3 pts)
- Better features. Instead of aligning based on RGB similarity, try using gradients or edges. (3 pts)
- Better transformations. Instead of searching for the best x and y translation, additionally search over small scale changes and rotations. Adding two more dimensions to your search will slow things down, but the same course to fine progression should help alleviate this. (3 pts)

Python Environment We are using Python 3.7 for this course. You can find references for the Python standard library here: https://docs.python.org/3.7/library/index.html. To make your life easier, we **recommend** you to install Anaconda 5.2 for Python 3.7.x (https://www.anaconda.com/download/). This is a Python package manager that includes most of the modules you need for this course.

We will make use of the following packages extensively in this course:

- Numpy (https://docs.scipy.org/doc/numpy-dev/user/quickstart.html)
- OpenCV (https://opencv.org/)
- Pytorch (https://pytorch.org/)
- Matplotlib (http://matplotlib.org/users/pyplot_tutorial.html)

References

https://en.wikipedia.org/wiki/Camera_matrix
https://en.wikipedia.org/wiki/Rotation_matrix
http://inst.eecs.berkeley.edu/~cs194-26/fa18/hw/proj1/
https://sites.google.com/a/umich.edu/eecs442-winter2015/homework/color