**CS 415: Computer Vision I**

# Homework 1

*Prof.: Sathya N. Ravi* Assigned: 09/04/2021, Due: 10/04/2020

Few reminders:

- Homework is due at the end of day on the designated date on Blackboard.
- No homework or project is accepted in mailbox of instructor.
- You may discuss homework with classmates and work in groups of up to **three**. However, you may not share any code, carry out the assignment together, or copy solutions from any other groups. Discussions between groups should be minimal, verbal during lecture or if appropriate, on Campuswire only. The submitted version must be worked out, written, and submitted by your group alone.
- **Important:** Each question (or subpart) should have a **lead** who has finalized the submitted solutions after discussions within group. The lead should be *explicitly* indicated in the submissions. There can be two leads for any question. The submitted solution is assumed to be verified by the other person, and so the grade is assigned equally.
- All solutions must be typeset (I recommend Latex, Markdown, Word etc., please do not use nonstandard formats) with code attached in MATLAB or Python format whichever is appropriate.
- Your final submission will be a zip folder with a PDF file containing solutions for each question including text, sample figures, calculation, analysis, and general writeup. Code and more experiments including figures should be provided in a separate folders.
- Submitting someone else's work (outside of the group) as your own is academic misconduct. Such cheating and plagiarism will be dealt with in accordance with University procedures (see the page on Academic Misconduct at this link).

Answer the following questions as completely as possible. Recall that $d$ dimensional vector spaces (for any finite $d$) over the reals $\mathbb{R}$ can be represented using $\mathbb{R}^d$.

1. Problem 2.10.2 (i) from HZ textbook.
2. Given a circular disk that lies anywhere on a plane parallel to the image plane, what is the shape of the image of the disk?
3. Describe the rank, degrees of freedom, and one more property of the so-called Fundamental matrix $F$.

4. **Setting up Python environment for Vision with Crops.** You will have to implement a basic image transformation using Python as part of this problem. An image $I$ of size $H, W$ is specified by a matrix with $H$ number of rows, and $W$ number of columns. We will focus on the following Crop transformation for the problem:

   (a) Crop: Vanilla cropping corresponds to selecting a region in an image. For a cropping function, given the output size $S$ of the image, the function is supposed to output a random region in the image with the **given** output size $S$. Let $W, H$ be the size of the given image $I$, and $W_c, H_c$ be a random variable representing the size of a random crop $I_c$ (to be outputted).

   We will be implementing a variant of this function in which the region in $I$ to be cropped is specified using two random variables: size/area $A \in (0, 1)$ and the aspect ratio $\alpha \in (0, 1)$ drawn from a uniform distribution with a **given** support (recall that $\sqrt{\alpha} = \sqrt{A}/H = W/\sqrt{A}$). Our function will repeat the following two steps until we get a valid crop:

      i. $W_c$, and $H_c$ are valid if $W_c \leq W$, and $H_c \leq H$. Sample a (valid) random width/height of the crop $W_c, H_c$ (by first drawing a sample of $\alpha$ and $A$).

      ii. If $W_c, H_c$ are valid, then draw the $x-$coordinate $w_1$ of the top left corner pixel uniformly at random between 0 and $W - W_c$. Then, the corresponding $x-$coordinate of top right pixel will be $w_1 + W_c$. Similarly the $y-$coordinate of top left $h_1$ can be drawn from uniformly at random between 0 and $H - H_c$. Then the crop $I_C$ of the given image $I$ is given by $I(w_1 : w_1 + W_c, h_1 : h_1 + H_c)$.

   Once we get a valid crop, we will simply resize the crop $I_C$ to the required size $S$ using **the `resize_image` utility function provided in the zip file**.

   If the random strategy fails for a number of trials $T$, then we will output a fixed sized crop from the center of the given image $I$. Please fill in the code in the class RandomSizedCrop. We recommend using NumPy for cropping the region and using our provided image resize function to resize the region. **Please see additional details in the code provided on Blackboard.**

   (b) Show that for a fixed $\alpha, A$, the crop function described above is a linear transformation of the input image $I$.

   (c) Implement Scale and Rotation transformations in the code.

5. **Understanding Convolutions in MATLAB.** In this problem, you will measure the responses of images to different filters. For each image, you will compute the response of each image pixel to the filters from a "filter bank". Later, you will see how to use these filter outputs to compute image representations.

   (a) Download these images: `cardinal1`, `cardinal2`, `leopard1`, `leopard2`, `panda1`, `panda2`.

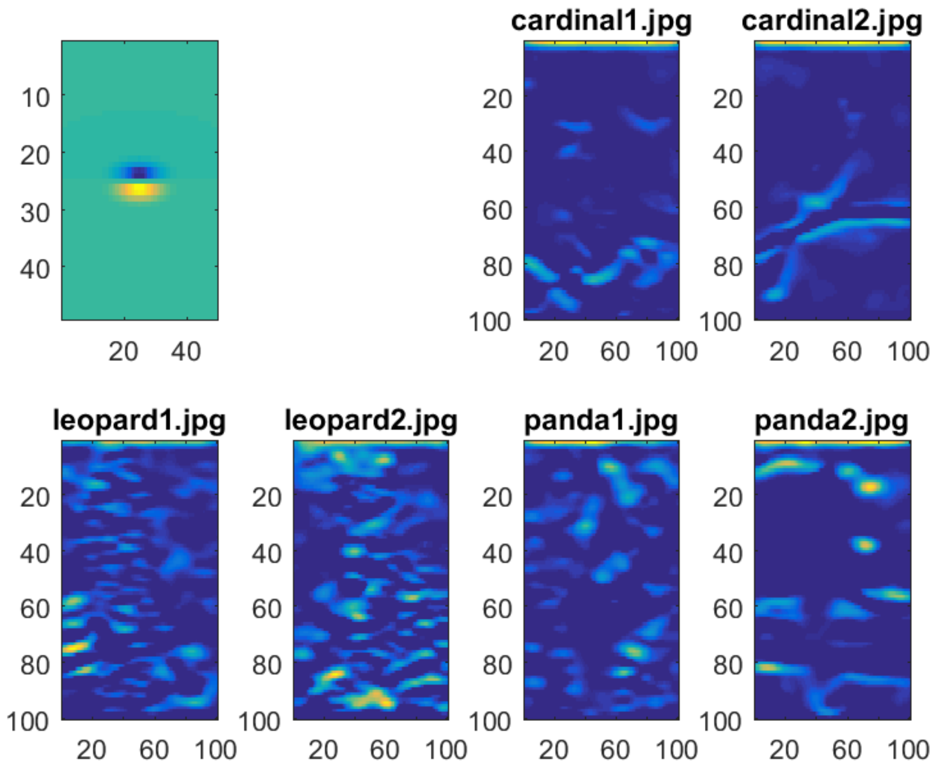

Figure 1: Filter Output Subplot showing responses of filter (shown top left) on various animal images cardinal, leopard, and panda.

(b) Download the script `makeLMfilters` (originally appearing in the Leung-Malik filter bank), and run it (`F = makeLMfilters;`) to get a set of filters. Each filter output `F(:, :, k)` is of size $49 \times 49$, and there are $48$ filters.

(c) In a script `filter_responses.m`, use a cell array to store the list of filenames so you can loop over them (e.g. `filenames{i}`). Read in all images, convert them to grayscale, and resize them to the same square size (e.g. $100 \times 100$), so that the visual map of responses (filtering outputs) can be more comparable across images. Compute the cross-correlation of each image with each of the $48$ filters using `imfilter`.

(d) For each filter, generate a $2 \times 4$ subplot showing the following subplot rows: (1) the filter and a blank subplot, (2) the responses to the cardinal images, (3) the responses to the leopard images, and (4) the responses to the panda images. Refer this link, for how to use the `subplot` function.

(e) Choose and include in your pdf (1) one filter where the responses for the same animal category are similar, while the responses for different animal categories are quite distinct; and (2) one filter where responses of different animals look fairly similar. Eyeballing is good enough for this homework. Name the files `same_animal_similar.png` and `different_animals_similar.png`. See Figure 1 for a sample filter output figure.

**Tips.** You can use

```
saveas(gcf,strcat('responses_to_filter_', num2str(i),'.png'));
```

to save your subplots, so you can later browse through them and pick interesting examples. You can also save images directly without displaying them, by turning the visibility off using `set(0,'DefaultFigureVisible','off');` near the beginning of your script. Make sure to turn it back on after this homework.

6. **Towards Crossdissolving Images.** In this problem, you will create a cross dissolve like effect using image filtering and scaling techniques.
   (a) Download one pair of images: `woman_happy` and `woman_neutral`, or `baby_happy` and `baby_weird`.
   (b) In a script `hybrid_image.m`, read in the first image in the pair as `im1` and the second as `im2`. Convert both images to grayscale, and resize them to the same square size (e.g. $512 \times 512$).
   (c) Apply a Gaussian filter to both, using e.g. `imgaussfilt(im1,10,'FilterSize',31)`. Save the results as `im1_blur`, `im2_blur`.
   (d) Obtain the detail image, by subtracting `im2_blur` from `im2`, and save the result as `im2_detail`.
   (e) Now add `im1_blur` and `im2_detail`, show the image, save it as `'hybrid.png'`, and include it with your submission. You can experiment with scaling it up and down (by dragging the corner of the Matlab figure) to see the "cross-dissolve" (or more informally "hybrid") effect.

**Additional Instructions for Question 5.**

- Install Anaconda. We recommend using Conda to manage your packages.
- The following packages are needed: OpenCV, NumPy, Pillow, Matplotlib, Jupyter, PyTorch. And you will need to figure out how to install them.
- You won't need Cloud Computing or GPU for this assignment.
- Run the notebook using: `jupyter notebook ./code/proj1.ipynb`
- You will need to fill in the missing code in: `./code/student code.py`
- Generate the submission once you've finished the project using: `python zip submission.py`