UNIVERSITY OF CALIFORNIA, MERCED

BERKELEY · DAVIS · IRVINE · LOS ANGELES · MERCED · RIVERSIDE · SAN DIEGO · SAN FRANCISCO

SANTA BARBARA · SANTA CRUZ

ME 021: Engineering Computing
MATLAB® Project

Fall 2023 ME 021 Course Staff: Alex Frias (ASE), Amir Kouravand (GSI), Hansong Lee (GSI)
Hassan Jafari Mosleh (GSI), Ayush Pandey (IOR)

Preliminary Notes

- **You need to choose one project out of four problem statements** Choose a project that is closest to your interests in "engineering computing" out of the four engineering projects posted as PDF on CatCourses (also available in course Box/MATLAB® Project folder). You must denote your choice in the 2nd MATLAB® project milestone, that is, the 7th project milestone on CatCourses.
- Your solution must be exclusively submitted via CatCourses. Email submission will not be accepted. Pay attention to the posted deadline because **the system automatically stops accepting submissions when the deadline passes.** You can upload one or more `.m` or `.mlx` files for your code.
- **To receive credit, you must also demo your project to your TA, either in your lab or during office hours.** You must show your working program, be prepared to explain your code, and answer other questions that the TA asks about MATLAB® programming. You must demo before the end of your last lab of the semester (that is, before Dec. 8, 2023). **You will get a 0 if you do not demo, even if you made a submission.**
- Four milestones will be released to help you make progress towards the project. However, milestones are simply to push you in the right direction. You need to put in extra effort beyond the milestones to complete the project satisfactorily.
- A short (1-2 page) report in PDF format that consists of main parts of the code, the visualizations, and any mathematical equations must be submitted for the final project submission along with the MATLAB® script files.
- The grading rubric for all milestones, demo, and the report submission will be posted on CatCourses and it will be available for the students to read through to prepare their submissions.

Contents

ME 021: Engineering Computing
MATLAB® Project – Image Processing

Ayush Pandey, Hassan Jafari Mosleh, Alex Frias

1 Overview

How does Instagram create all those nice filters on your favorite images!? In this project, you will have a chance to explore those filters and much more with MATLAB®. Images come in all shapes and form. But the underlying math can let us edit all kinds of images, extract valuable information, and even generate new images! MATLAB® makes all of these tasks feasible in a user-friendly manner. In this project, you will explore image properties like brightness, color, contrast, and apply the math of image processing to extract valuable information from images.

2 Preliminaries

2.1 Properties of images

Common ways to characterize the properties of an image include:

1. **Resolution** determines the amount of detail an image holds, often expressed with how many pixels the image has.
2. **Color Depth** indicates the number of bits used to represent the colors in an image. For example, a black and white image has only one bit color depth.
3. **Brightness** refers to the overall lightness or darkness of an image, expressed with absolute value of the pixels.
4. **Contrast** is the difference in luminance or color that makes an object in an image distinguishable from other objects and the background. It is a relative measure.
5. **Noise** can refer to random variations of brightness or color information in images, and often an aspect to be reduced in image processing.

By analyzing these properties, we can enhance, restore, and compress various images. There are many applications of image processing — security, entertainment, and scientific research.

2.2 Mathematical description

Images can be represented using matrices. For an image $\mathbf{M}$:

1. **Brightness adjustment:** The brightness of an image can be adjusted by adding a constant to each pixel value:
$$\mathbf{M}' = \mathbf{M} + b$$

where $\mathbf{M}'$ is the adjusted image and b is a brightness constant.

2. **Contrast adjustment:** The contrast of an image can be modified by scaling the pixel values:

$$\mathbf{M}' = a\mathbf{M}$$

where a is a contrast scaling factor.

3. **Covariance matrix:** The covariance matrix $\mathbf{C}$ of an image can be computed, and the eigenvalues and eigenvectors of $\mathbf{C}$ are found by solving the equation:

$$\mathbf{C}\mathbf{v} = \lambda\mathbf{v}$$

where λ represents the eigenvalues and $\mathbf{v}$ represents the eigenvectors. The eigenvectors corresponding to the largest eigenvalues are the principal components of the image.

4. **Singular Value Decomposition (SVD):** The SVD of an image matrix $\mathbf{M}$ is a the following formula:

$$\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T$$

where $\mathbf{U}$ and $\mathbf{V}$ are two unique matrices, and Σ is a matrix that contains values in the diagonal and all other values are zero. The diagonal values are called singular values.

5. **Filters:** Filters can be applied to images to enhance certain features or remove noise. For example, a Gaussian filter can be used for smoothing:

$$\mathbf{M}' = \mathbf{G} * \mathbf{M}$$

where $*$ denotes the convolution operation and $\mathbf{G}$ is the Gaussian filter matrix. This is a simple operation to run in MATLAB!

3 Computational Tasks

For this project, you need to capture three different kinds of images (an object, a natural landscape, and handwritten text) from your phone and upload them to your computer. Then, you are required to perform the following computational tasks on the images.

3.1 Manipulating images

To enhance your captured images, apply a combination of transformations, including brightness and contrast adjustments. Generate different versions of each image that you have captured and discuss how the image quality varies with these transformations. Document the transformations applied and the rationale behind your optimization strategy to generate the "instagram-worthy" images (this will be your subjective choice).

3.2 Image reconstruction using SVD

An image matrix may look complicated and a collection of a large set of numbers, but not all of those numbers are equally important! Using matrix computations, we can find the most "dominant" set of vectors that can be used to recreate the same image, albeit in a lower quality. You can think of this as an image

compression algorithm.

In this task, you will explore the use of Principal Component Analysis (PCA) to compress an image using MATLAB®. The goal is to understand how matrix computations can reveal the dominant features of an image and how these can be used for compression.

So, what is PCA? Think of PCA as a smart camera filter for your phone's images. It zeroes in on the most important features of your photos, reducing the "data noise" while preserving their essence. By focusing on what matters most, PCA compresses your images efficiently, making them easier to store and share without significant loss in quality.

If your image is a matrix M, then in mathematical terms, PCA is the process of decomposing M into a set of special vectors that represent the directions of maximum variance in your data. These vectors are called the principal components of the image. A matrix transformation method called "Singular Value Decomposition (SVD)" of the matrix M is used to compute these special vectors that give us the principal components of the image. Next, we describe a step-by-step process of how you can compute the SVD for your images.

First, you should convert your RGB image to uint8 and then to a grayscale image (You can use MATLAB's `rgb2gray` function to convert to grayscale). This simplifies the image data and focuses on structural details (if you are feeling up for a challenge, you may continue to work with a color RGB image! Note you will have to calculate SVD three times, one for each color channel).

After this initial setup, you are ready to run the matrix transformation called the Singular Value Decomposition (SVD). MATLAB already has a built-in function for SVD (conveniently called, `svd`). You will not be asked to explain the technical details of how SVD is computed or how it works for the purpose of the ME 021 project. You are free to explore these concepts if you are interested. The SVD function will return three matrices that you can store as U, Σ (called Sigma), and V.

The components or the dominant features in the image are arranged in descending order in the SVD output you obtained in the previous step. The total number of "features" is given by the number of non-zero values in the diagonal of the Σ matrix. Notice that Σ must be a diagonal matrix from your computation above. To compute a reconstruction, you can choose the first few columns of U, the corresponding first few values in Σ and the same number of rows in V. From SVD, you get,

$$M = U\Sigma V^T \tag{1}$$

where V^T is the transpose of the matrix V. The compressed reconstruction can be computed as

$$M^r = U^r \Sigma^r \left(V^r\right)^T \tag{2}$$

where you choose the first r columns of U, the highest r values in Σ and the first r rows in V. Choose a value of r to keep for the image reconstruction and discuss how the number of components affects the quality and size of the reconstructed image. To complete this task, you must display the image and its compressed reconstruction side-by-side.

An approach for image reconstruction in Matlab utilizing SVD matrices can be implemented as follows:

$$\text{reconstructed\_image} = U(:, 1:r)S(1:r, 1:r)V(:, 1:r)^T;$$

where:
U - Gets all the rows but r columns

S - Gets r rows and r columns, making a $r \times r$ submatrix
V^T - Gets all rows and r columns and then transposes it (switching its rows and columns)

The r decides the quality of the compressed image that you can explore.

Note: While implementing the above steps, ensure to document your observations and insights at each stage. Focus on the computational power of indexing multi-dimensional arrays in MATLAB.

3.3 Extracting useful information from images

Temperature Detection from LIF Images: You are given two reference images representing known temperatures and one image with an unknown temperature (see Figure 1. Both of these images are available on the Box folder: the first reference temperature image at 289.6 K, and the second reference temperature image at 310.45 K. You also have the image with unknown temperature values on Box: T_data. Your task is to apply image analysis techniques to the Laser-Induced Fluorescence (LIF) images to estimate the unknown temperature. The analysis technique that you will apply is indexing the matrix at each pixel value and applying algebraic computations to those values. The formula for this computation and a step-by-step description is given in Figure 1 and in this MATLAB script. You might find the imagesc function in MATLAB helpful for image heatmap generation or you could create your own cmap with the `imshow`. Document your methodology and the computational processes used for this estimation.

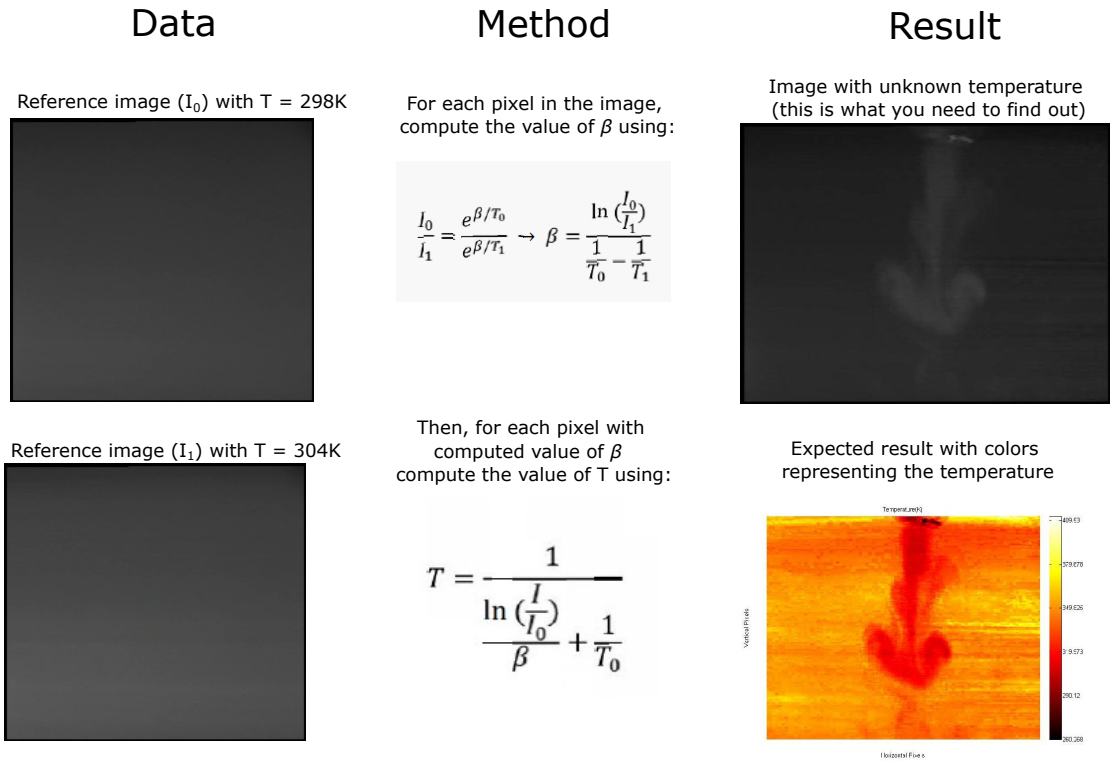

Figure 1: The first column gives two reference images I_0 and I_1. You can use these images to compute the value of β using the formula in the second panel. With the value of β, you can then compute the temperature values in the problem image given in the third column. The goal is to generate the bottom right corner image as a result.

4 Open-ended questions

You may explore one or more of the following open-ended ideas in your project.

1. Can you implement one of the filters that Instagram has using matrix computations?
2. Experiment with different numbers of principal components and observe the effects on the image quality and compression ratio and how this affects the temperature computation.
3. How does one generate new images? Have you heard about Dall-E? Can you create your own matrices to generate a new image that has certain properties?

For the open-ended problems, include justifications for each of your choices, that is, explain your reasoning and thought process. Discuss the potential trade-offs (advantages and disadvantages) and how you address them. Take the open-ended questions as an opportunity to explore data/system design/visualization that interests you the most!