CSE47201: Introduction to Computer Vision

Assignment #1

Due on Wednesday, October 6th, 2021, before the class (10:30 AM)

Objective

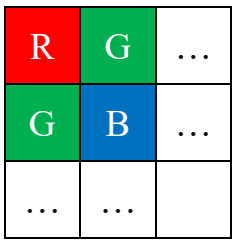
The goal of this assignment is to understand interpolation techniques and image alignment. If you have any issues, feel free to contact the instructor for advice. Please BE creative; feel free to apply what you learned in the class.

Instructions (READ CAREFULLY)

- Complete individual steps and turn in your outputs (see **Task #**) to Blackboard. There are a total of **5 Tasks** in this assignment.
- A single PDF file with all solutions and a discussion of your solution.
- Along with the PDF file you should provide a link for your results and code (e.g., GitHub).
- No extension. Please start your assignment as soon as possible. Do NOT wait until the deadline.

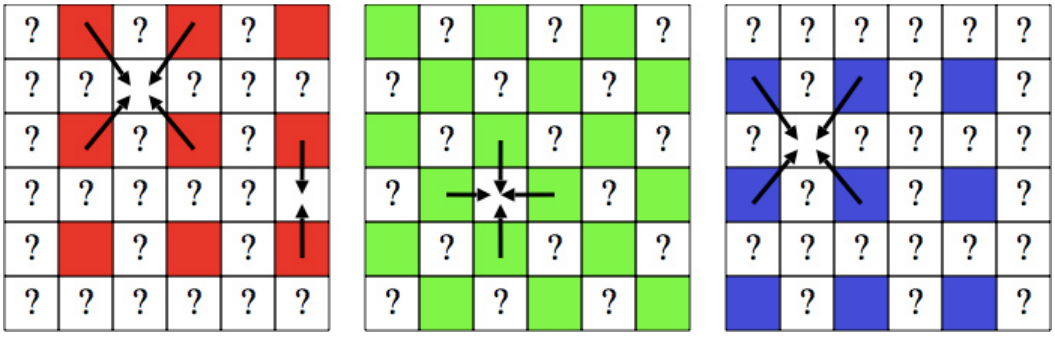
1. Intensity interpolation

The goal of the assignment is to get started with image processing in MATLAB by implementing a very simple demosaicing algorithm. The "mosaic" image (crayons_mosaic.bmp) was created by taking the original color image (crayons.jpg) and keeping only one-color component for each pixel, according to the standard Bayer pattern as shown in the following example:



Task 1: Extract each channel from the mosaic image. Hint: In MATLAB, see point-wise multiplication ".*" and matrix tiling "repmat". Once you read the "mosaic" image into a matrix, entry (1,1) will be the value of the "red" component for pixel (1,1), entry (1,2) will be "green", etc.

Task 2: Implement a very simple linear interpolation approach for demosaicing. For each pixel, fill in the two missing channels by averaging either the four or the two neighboring known channel values:



Avoid using loops! Use MATLAB's imfilter function (filter2 or conv2 work just as well). In addition to your solution, please fill in the following tables to show how you design filters for each channel.

Filter for R

Filter for G

Filter for B

Task 3: The above method, being very simple, does not work perfectly. You can see where it makes mistakes by computing a map of squared differences (summed over the three-color components) between the original and reconstructed color value for each pixel. Compute such a map and display it using the "imshow" or "imagesc" commands (see the documentation for those commands to figure out how to scale the values for good visibility). In addition, compute the average and maximum per-pixel errors for the image. Finally, show a close-up of some patch of the reconstructed image where the artifacts are particularly apparent and explain the cause of these artifacts.

2. Image Alignment

Background: Sergei Mikhailovich Prokudin-Gorskii (1863-1944) was a photographer who, between the years 1909-1915, traveled Russian empire and took thousands of photos of everything he saw. He used an early color technology that involved recording three exposures of every scene onto a glass plate using a red, green, and blue filter. Back then, there was no way to print such photos, and they had to be displayed using a special projector. Prokudin-Gorskii left Russia in 1918. His glass plate negatives survived and were purchased by the Library of Congress in 1948. Today, a digitized version of the Prokudin-Gorskii collection is available online (see Figure 1 for example).

Figure 1. glass plate image alignment: three-color channels (left column) and their reconstructed image (right).

The goal is to learn to work with images in MATLAB by taking the digitized Prokudin-Gorskii glass plate images and automatically producing a color image with as few visual artifacts as possible. In order to do this, you will need to extract the three-color channel images, place them on top of each other, and align them so that they form a single RGB color image.

Task 4: The images in "data" contain all channels in a single (see Figure 1). Your program should divide the image into three equal parts (channels). Note that the filter order from top to bottom is BGR, not RGB!

Task 5: Align two of the channels to the third (you should try different orders of aligning the channels and figure

out which one works the best). For each input image, you will need to display the colorized output and report the (x,y) displacement vector that was used to align the channels. The easiest way to align the parts is to exhaustively search over a window of possible displacements (say [-15,15] pixels), score each one using normalized cross-correlation (NCC, elements in the patch are normalized by their mean and standard deviation like z-score), and take the displacement with the best score – the images to be matched do not actually have the same brightness values (they are different color channels), so NCC is expected to work better than conventional cross-correlation. In MATLAB, you can use "normxcorr2".

Bonus (extra credit): We consider image alignment for high resolution data. The "data_hires" directory contains several high-resolution glass plate scans. You have two options for this task and provide either of them.

(Option 1) Multiscale approach: For these images, exhaustive search over all possible displacements will become prohibitively expensive. To overcome such an issue, implement a faster search procedure such as an image pyramid. An image pyramid represents the image at multiple scales (usually scaled by a factor of 2) and the processing is done sequentially starting from the coarsest scale (smallest image) and going down the pyramid, updating your estimate as you go. It is very easy to implement by adding recursive calls to your original single-scale implementation. Alternatively, if you have other ideas for speeding up alignment of high-resolution images, feel free to implement and test those.

(Option 2) Boundary cropping: Implement and test any additional ideas you may have for improving the quality of the colorized images. For example, the borders of the photograph will have strange colors since the three channels won't exactly align. See if you can devise an automatic way of cropping the border to get rid of the bad stuff. One possible idea is that the information in the good parts of the image generally agrees across the color channels, whereas at borders it does not.