Spring 2020 - CS 477/577 - Introduction to Computer Vision

Assignment Seven

Due: 11:59pm (*) Thursday, March 26.

(*) There is grace until 8am the next morning, as the instructor will not grade assignment before then. However, once the instructor starts grading assignments, no more assignments will be accepted.

Weight: Approximately 5 points

This assignment can be done in groups of 2-3. You should create one PDF for the entire group, but everyone should hand in a copy of the PDF to help me keep track. Make it clear on the first page of your PDF who your group is. Group reports are expected to be higher quality than individual ones.

General instructions

You can use any language you like for this assignment, but unless you feel strongly about it, you might consider continuing with Matlab.

You need to create a PDF document that tells the story of the assignment, copying into it output, code snippets, and images that are displayed when the program runs. Even if the question does not remind you to put the resulting image into the PDF, if it is flagged with ($), you should do so. I should not need to run the program to verify that you attempted the question. See

http://kobus.ca/teaching/assignment-instructions.pdf

for more details about doing a good write-up. While it takes work, it is well worth getting better and more efficient at this.

30% of the assignment grade is reserved for exposition.

Learning goals

- Understand the color constancy problem
- Become fluent with two simple color constancy algorithms
- Additional practice with multiple error measures
- Learn about "oracle" algorithms for understanding performance of algorithms (grads).
- Practice deriving simple minimizers (grads)
- Learn more about optimization (grads)

Assignment specification

This assignment has two parts, one of which is required for both undergrads and grads, and one which is required for graduate students only.

Part A

Background

This gzipped tar file (also on D2L)

http://kobus.ca/teaching/cs477/data/color_constancy_images.tar.gz

contains a directory with image pairs where one of the images is taken under one light and the other image is taken under another light. Hopefully the organization is clear from the file names.

Note that some of these images are very dim because they were imaged so that specularities (if they exist) did not cause too much pixel "clipping". A pixel is clipped if one of the channels would have a value of over 255, and is set to 255. You cannot trust the value of a pixel that is 255, or even ones that are next to 255 ones due to "blooming" effects. However, the images were also taken in such a way to keep noise low. Hence you can scale them up for visualization, and even computation as long as you stay in a floating-point representation.

These images also appear dim because they are linear images (not gamma corrected). This simplifies things because for some questions that we might want to ask of images, linear ones make more sense, and if they were gamma corrected, we would then need to linearize them first. Fortunately for us, this has already been done. If warranted, you are welcome to adjust image ranges or image gamma for images that you put into your report (make a note of it in the caption).

> ***Hint regarding reading the images****. You will likely want to convert the image pixel values to doubles for computation and round() to integers and convert back to uint8 for display. Also, because they are stored as uint8, some pixels might be completely black (0,0,0), which could cause problems when we compute (r,g) error. So, I suggest adding a small number to each value. For consistency of our results, use 0.0001.*

Inspection of the file

`macbeth_syl-50MR16Q.tif`

reveals that the "syl-50MR16Q" light (50MR16Q is just the Sylvania corporation light bulb number) is reasonably close to what this camera expects, as the white patch (bottom left) looks neutral, and the R,G,B values are relatively even (not perfect, but good enough for our purposes). We will thus take the "syl-50MR16Q" as our canonical light, and one interpretation of the color constancy problem is to make images of scenes look as though this light was shining on them (instead of the actual, unknown, light).

In color constancy we often do not care about the absolute scale (magnitude) of our estimate of the illumination (R,G,B), or, similarly the error in the estimated brightness of the image. We will assume that our task is to infer and/or correct the *chromaticity*. To make things easier to grade, report the color of the illumination scaled so that the max value is 250. We will consider two error measures for color constancy, both of which ignore brightness. In addition, you are asked to provide triplets of images: original, corrected, and canonical (target)[1]. This visual comparison should be done using full RGB color, not chromaticity. In more detail:

[1] You should present images in this order (e.g., if you are doing panels of three, then this is the order left to right). This order makes sense because it is input, followed by output, followed by the target (best possible answer). If I were comparing a new algorithm to one or more standard (baseline) ones, I would put the input first, followed by baselines in some natural order (e.g., sophistication or efficacy), followed by the results of the proposed method, again in some natural order (e.g., enabling more features), followed by the target.

- **Angular error of illuminant estimate**. Here, consider the illumination color, and the estimate thereof, as vectors in 3D space, and compute the angle between them in degrees. To get the angle recall that the cosine of the angle between two vectors is the dot product of them, divided by each of their magnitudes (or the dot product of unit vectors). Then you can use `acos()` to get the angle in radians and convert to degrees, or more conveniently use `acosd()`. If you do not divide by their magnitudes, then your argument to `acos()` can be greater than one in absolute value, and then the angle is a complex number (Matlab knows that, even if you do not, so if your answer is a complex number, this is likely the cause).
- **Corrected image**. We can use the illumination estimate to convert an image (*original* image) to a different one which is an estimate of the image as if the scene where illuminated by the canonical light. To do this we use the diagonal transformation computed by the element-wise ratio of the canonical light RGB and the estimated light RGB. This is referred to below as the *corrected image*. If the algorithm is working well, and the diagonal model is OK, then the corrected image will be close to the image of the scene under the canonical light (*target* image), perhaps with a scaling to make the overall brightness similar for visualization. There is one target image for each scene, and they are provided for evaluation.
- **RMS (r,g) mapping error**. Having computed the corrected image, we can compute measures of how close it is to the canonical (target) image. We will focus on the RMS of the (r,g) mapping error. To compute this, you compute the *(r,g)* representation of the corrected image and the target image. Recall that $r=R/(R+G+B)$ and $g=G/(R+G+B)$). When we do this, we must exclude dark pixels. Because these images were taken carefully, we can use a relatively "dark" definition of dark. Lets exclude pixels where $R+G+B$ *i*s less than 10. (If you need to reduce this threshold, let me know by email, and/or in your writeup). The errors are the differences in *r* and *g* between the two images, pixel by pixel. The RMS error is square root of the average squared error over all the *r* and all the *g* (taken together) except where one of the original images was too dark.

Finally, we are ready to begin!

Problems

1) Average some of the pixels fully inside the white patch in macbeth_syl-50MR16Q.tif, and then scale the resulting RGB by a single scale factor so the max is 250 to provide an estimate of the illuminant color (for the Sylvania light, which, as noted above, will be our canonical illuminant) **($)**. (Macbeth is the name of a company that makes things like "color checker" charts for photography and scientific applications, which is now part a company called "X-rite").

> *Tip. One way to do this is to click two diagonally opposite points of a rectangle that fits comfortably within the white patch (the pixels near the edges of the patch are not reliable). Then take the average of all pixels within that rectangle with a few lines of Matlab.*

2) Do the same for macbeth_solux-4100.tif. What is the color of the "solux-4100" light? **($)**

3) What is the angular error between the two light colors found in problems 1) and 2)? **($)**.

4) Now use the diagonal model computed from the illuminant RGB found in problems 1) and 2) to map the second (bluish) image to the one under the canonical light. This is the corrected image as described above, where the "algorithm" is essentially an oracle because you happened to have a white patch in the image, and you (the human) found it. Display three images: The original (bluish) one, the (hopefully) improved (corrected) one, and the canonical one for reference. To reduce confusion about differences between the images being due to brightness instead of chromaticity (color without brightness), scale each of these images by multiplying all the RGB by a single scale factor so that the max value over all the RGB is 250 **($)**.

5) Compute the RMS error in the chromaticity coordinates (r,g) defined above for the pixels between A) the original image and the canonical (i.e., with no color constancy processing), and B) the (hopefully) improved corrected image and the same scene imaged under the canonical illuminant (i.e., using "oracle" color constancy) **($)**.

6) Using the MaxRGB algorithm, estimate the light color in the remaining three solux-4100 images (apples2, ball, and blocks1). Report the angular errors between these three estimates (one for each scene) and the solux-4100 light color you measured using the Macbeth color chart image **($)**.

7) Using the MaxRGB illumination estimate from problem 6), display image triplets: original, corrected image, and canonical (target) image for each of the three scenes (9 images total). All 9 images should be on **a single page**, i.e., there is only one figure. For each scene (row in your table) you follow the same process that you did in problem 4) using the Macbeth color chart scene and the human assisted oracle algorithm **($)**. Report the *(r,g)* RMS error for the mapped images **($)**. Is there good agreement between the general trend of the *(r,g)* RMS error and the angular error computed in problem 6) **($)**?

8) Now repeat the error computations from the previous two questions, using the gray-world method instead of the MaxRGB method and report the two kinds of errors **($, $)**. **[No need to include images in your report for this part, unless you particularly want to]**. Which method is working better on this data **($)**?

Part B (Required for grads only).

> ***Background***. *Often when you are developing algorithms to solve problems, it is a good idea to consider how much improvement is possible. Perhaps the algorithm you are trying to improve is already very good! Such arguments are usually relative to the definition of "good" (e.g., your chosen error measure), and you sometimes need to use a non-ideal definition of "good" due to practical considerations. Regardless, results are often best in the context of baseline (a standard or dumb algorithm), and a target which you do not expect to be able to exceed.*

9) Derive a **formula** for the "best" diagonal map between the (R,G,B) under one light and the (R,G,B) under another light using the combined R, G, and B sum of squared errors in as your definition of "best" (**$**). Based on your derivation alone, is this diagonal map guaranteed to give a better answer using the (r,g) measure than any algorithm you might invent **($)**?

10) Following the same pattern of investigation as in 7), provide before, after, and target images for the correction procedure you just derived **($)**. This is analogous to the single figure with 9 sub-images from question 7). Try to put the figure in context using the note on "background" above **($)**.

11) We often end up optimizing something different than the error we are most interested in because of computational simplicity or speed or pedagogy. One clear example from HW5 is reprojection error versus the error that is minimized using homogenous least squares. A second example might be RMS *(r,g)* error versus the error you minimized in part B. An approximate minimum can provide a good starting point for optimization algorithms that work better with a good starting point. Regardless, this problem is to try to improve upon the answer for B.10 above using an off-the-shelf optimization function. Matlab has quite a few possibilities, such as `fminunc()` to look for nearby local minimum, and `ga()` and `simulannealbnd()` which attempt to find global minimums. Other software platforms provide similar and/or additional options.

Experiment a bit, and report the results of using at least one off-the-shelf[2] optimization method to carry on the quest for finding the "best" diagonal map for each of our three scenes, where "best" is defined by the smallest RMS *(r,g)* mapping error **($)**. Discuss whether you are able to improve upon the results from B.9/B.10 **($)**.

Tip for computational speed. Although it may not be a big deal for this situation, you should internalize the observation that minimizing the sum of squared error is equivalent to minimizing RMS error, and uses less computation.

What to Hand In

As usual, the main deliverable will be PDF document that tells the story of your assignment as described above. Ideally the grader can focus on that document, simply checking that the code exists, and seems up to the task of producing the figures and results in the document. So you need hand in your code as well.

[2] Of course, the optimizing code does not **have** to be off-the-shelf. For example, if you have happened to have written your own for some other project, you might like to try it out on this problem.