CS364/AM792 : Computer Vision 29 July 2019

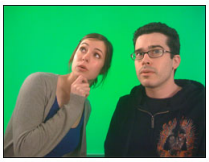
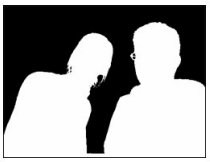
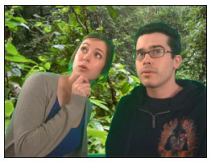
Assignment 1: Image Processing and Feature Matching

Please prepare a single, condensed and neatly edited document for submission. For each problem include a short description of what you did, results, and a brief discussion/interpretation. It is not necessary to include any code in your document, unless a snippet helps to explain your method. Upload a zip-file containing your document and code to SUNLearn (under "Assignment 1") **before 17:00** on 12 August. Also hand in a printed copy of the document, at the latest during class on Tuesday 13 August.

You are free to use any programming language. I recommend Matlab or Python. If you are asked to implement a specific technique, the idea is that you do so from scratch; do not simply use a function from an image processing or computer vision library. Collaboration is restricted to the exchange of a few ideas, and no form of plagiarism will be tolerated. All code, results, and write-up that you submit must be your own work.

---

1. Separate the actors from the green background in the image `greenscreen.jpg` by thresholding the different colour channels in a suitable way. Then place the actors on top of some other (more interesting) background of your own choosing.

---

2. Perform unsharp masking on a greyscale image of your own choosing. You will observe that the output image may contain values outside $[0, 255]$. Why is that? What would be better — to scale all the intensities of the output, or to simply truncate only those values outside the interval?

---

3. Create a function that will take a colour image as input and scale (resize) it by a given factor s. Implement both nearest neighbour (NN) and bilinear (BL) interpolation. Pick a colour image as input and show a handful of results for different s, using both NN and BL interpolation. Show at least one close-up where the difference between NN and BL is clear.

---

4. **(a)** Find, download and/or install any decent image feature detector and matcher (for example SIFT, SURF, FAST, BRIEF or ORB). There are many resources to choose from, including:

- various options in the `feature` module in scikit-image;
- the `extractFeatures` function in MATLAB (requires the Computer Vision System Toolbox);
- the `features2d` and `xfeatures2d` modules in OpenCV;
- Edward Rosten's FAST corners (for Python, MATLAB, C, etc.).

Wrap the chosen modules/functions so that you can easily (1) get a list of feature coordinates and corresponding descriptor vectors from a single image, and (2) match two sets of descriptors from two images. Include a short description of your chosen method, cite your source(s), and explain what you did to get it to perform the two separate tasks above.

(b) Now find and match features in `semper0.jpg` and `semper1.jpg`. Put some effort into picking parameter values that produce good results. Use dots and short line segments over one of the images to visualize features and matches (as discussed in class).

---

p.t.o.

5. Test the robustness of your feature detector and matcher against scale variation, as follows. Select a suitable test image A. Resize this image using your function from problem 3, and call the result B. Now find features in A and B, match them, and measure the *repeatability* of the detection as well as the *accuracy* of the matching*. You should automate the process so that you can generate results for many scale factors (e.g. 0.1 to 3 in steps of 0.1). Plot two graphs (repeatability *vs* scale factor, accuracy *vs* scale factor), show a few typical results (detected features and matches over an image pair), and discuss.

*Repeatability is the percentage of A's features that are detected in B, regardless of how they are matched. Matching accuracy is the percentage of matches between A and B that are in fact correct. It is of course possible to measure these two automatically, without manual inspection, since we know exactly where the features in A are *supposed* to be located in B. Just remember that features cannot be located to extremely high accuracy so, for argument's sake, an error of a pixel or two might be quite acceptable.

Hand in: 12 August 2019