HW3 – submission 16.1.18 23:55

Guide lines

1. You should submit all function and script files written in MATLAB/Python. Your code should be well documented and clear. The code should run from **any** computer and include all path definitions (You should take care of this in the code).
2. Please divide the code by questions.
3. Final report – should include explanations on the implementation and the execution, answers to the questions, results, conclusions and visual results. Do elaborate on all parts of the algorithms/solution. **Please submit a PDF file and not a DOC file.**
4. Please post question regarding this HW on the facebook group (English only): https://www.facebook.com/groups/294298727746314/
5. The grades are highly depended upon the analysis depth of the report.
6. HW could be submitted in pairs.
7. Please include your ID in the report.
8. Eventually submit one compressed file including the code + images PDF.

Good luck!

Question 1 – Laplacian pyramid (50 points):

On this question you will implement a style transfer for headshot portraits, inspired by the paper attached to this exercise (there is no need to read the full paper).

The goal: transfer the style of an example headshot photo onto a new one. This will be done by transferring the local statistics of the example portrait at different scales onto a new one. By that, we could match the properties of the input image (such as the local contrast and the overall lighting direction) to the given example image, while being tolerant to the natural differences between the faces of two different people.

(a) Implement a function that decomposes a gray-level image to its Laplacian pyramid. The function should accept an input image I and the number of pyramid levels n, and should return the pyramid's levels $L_l[I]$. The pyramid level $L_l[I]$ are defined as follows:

$$L_l[I] = \begin{cases} I - I \otimes G(2^2), & l = 1 \\ I \otimes G(2^l) - I \otimes G(2^{l+1}), & 2 \le l < n \\ I \otimes G(2^n), & l = n \end{cases}$$

Where $G(\sigma)$ is a 2D Gaussian kernel with a standard deviation of σ and $\otimes$ is the convolution operator. Note: On this Laplacian pyramid **do not use down sampling** (all the pyramid levels have the same size).

Tips:

- You may first construct the Gaussian pyramid of the image, and then construct the Laplacian pyramid by calculating the differences between the Gaussian pyramid levels.
- You may use the function `fspecial` to create the filter G. Make sure that the matrix representing the filter is about 5 times larger than the filter width σ.
- You may use the function `imfilter` to perform convolution.

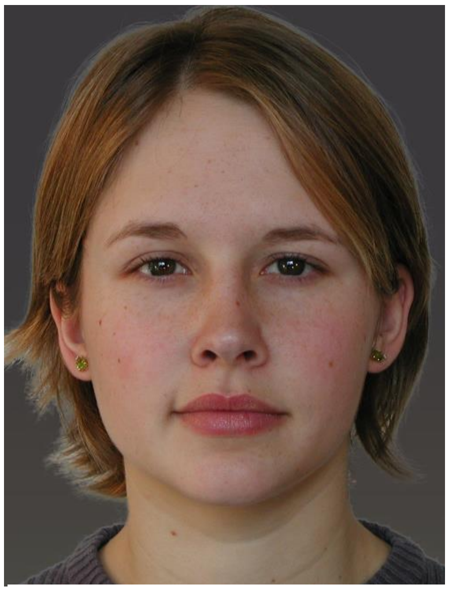
(b) Implement a function that reconstructs an image from its Laplacian pyramid. The function should get the pyramid levels and should return the reconstructed image.

Test: take an image, construct its Laplacian pyramid using the function of section (a) with $n=6$ levels, and reconstruct the image using the function of (b). Is the reconstruction accurate? In your answer, discuss how using down sampling would affect the accuracy of the reconstruction.

In the following sections, you will implement the style transferring process. This is done by multiplying each of the levels of the input image Laplacian pyramid by a gain, which depends on the proportion between the input image pyramid and the example image pyramid. For sections (c)-(f) use the image "data\Inputs\imgs\0004_6.png" as the input image I and "data\Examples\imgs\6.png" as the example image E. Use the images with a dynamic range of [0 1] (*i.e.* use `im2double`).

(c) Background: Before transferring the style we would like to change the background of the input image to be similar to the background of the example images. Each of the example images has a corresponding background image (located at "data\Examples\bgs"). In addition, each input image has a binary mask (located at "data\Inputs\masks"). This mask has values of "0" at pixels correspond to the background and "1" at the face. Implement a function that accepts an input image, its mask and the example background. See example bellow:

Input image with Original Background


Input image with Example Background

(d) Calculate the energy and the Gain: construct the Laplacian pyramid of both the Input image I (with the new background) and the Example image E with n=6 levels. **Construct a separated pyramid for each of the image channels c** (c=R,G,B – three pyramids for each image) using the function that was implemented in section (a) . For each pyramid and for each level l , calculate the local energy S_l according to:

$$S_l[I^c] = \left(L_l[I^c]\right)^2 \otimes G(2^{l+1})$$

Finally, calculate the gain map of each level as:

$$Gain_l^c = \sqrt{\frac{S_l[E^c]}{S_l[I^c]+\varepsilon}}$$

Where $\varepsilon = 10^{-4}$. Clip the Gain of each level to be with maximal value of 2.8 and minimal value of 0.9.

(e) Construct the output image pyramid: The output image O is constructed from a new pyramid:

$$L_l^c[O] = \begin{cases} Gain_l^c \times L_l^c[I] & 1 \le l < n \\ L_n[E^c] & l = n \end{cases}$$

For each of the output image channels (RGB) construct a new pyramids, according to the formalism above. Note: the last pyramid level equals to the last level of the example image.

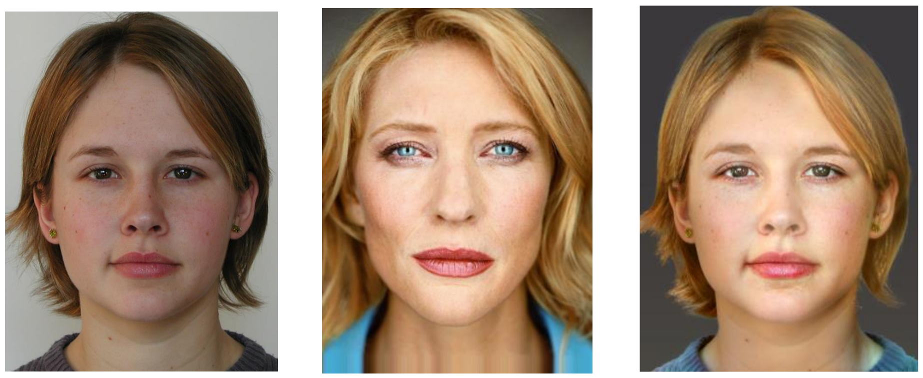
(f) Reconstruct the output image: using the reconstructing function of section (b), reconstruct the RGB channels of the output image from their corresponding pyramids. Fuse the three channels to create an RGB image and present the results. The get better results, replace the background of the output image as in section (c).

(g) Repeat this process for transferring the style of images 16 & 21 to the input image 0004_6.png, and of the images 0,9,10 to the image 0006_001.png. Present all the results.

(h) Run the algorithm on another input or example image which was not given in the data files.

(i) Pyramid Blending: Choose one the previous settings (an input image, style and background), and apply the "Pyramid Blending" algorithm we saw in class. The left-hand side of the output image should be the original input image, and the right-hand side should be the result of step (f). Repeat this experiment for 3 numbers of pyramid levels (e.g. n=1, n=3, n=6), plot the results and explain.

Example for the style transfer result (f):



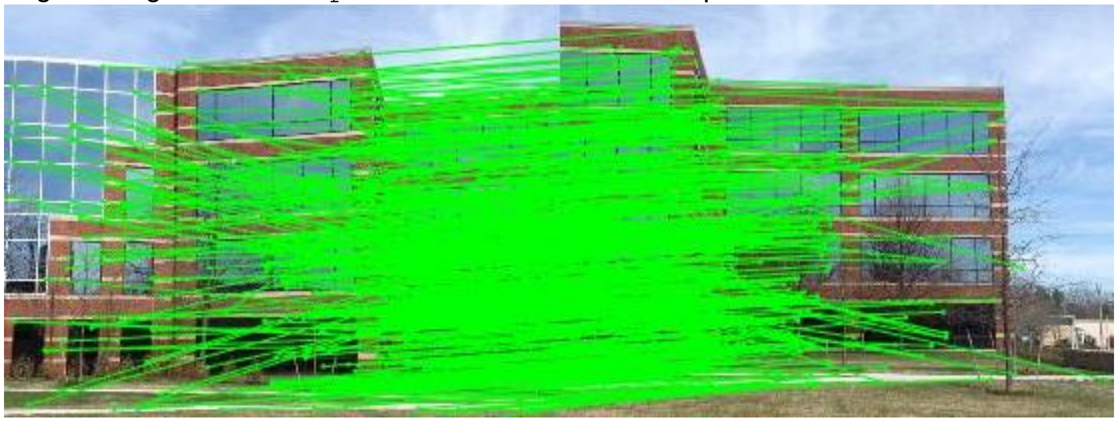
Remark: replicate padding was used on some of the example images, to insure all images have the same size.

Question 2 – Image stitching (50 points):

In this exercise you will implement an image stitching algorithm, based on SIFT descriptors. You are provided with an implementation of SIFT (Created by Andrea Vedaldi UCLA Vision Lab - Department of Computer Science, University of California).
In this exercise, the following Matlab function are **prohibited**:

cp2tform, imtransform, tformarray, tformfwd, tforminv, maketform, imwarp or any other matlab function which calculates a transformation or warping.

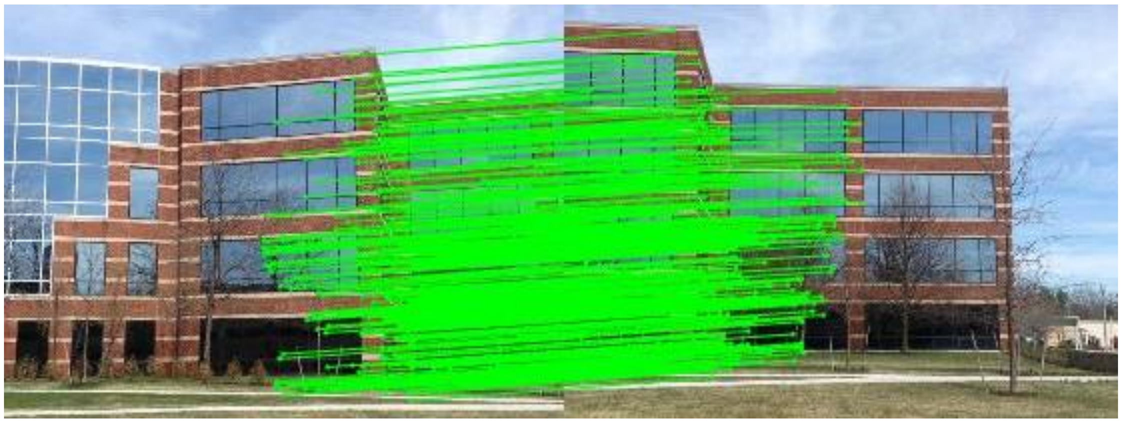
1. Choose a single pair (two consecutive images) for the rest of this work from the folder “building”.
2. For each image (in the chosen pair), convert to greyscale and extract its SIFT features and descriptors, using the function `sift.m`.
3. Find matching key-points: use the function `siftmatch` which gets two sets of SIFT descriptors (corresponding to two images) and return matching key points. Show all matching key-points on the colored images using the function `plotmatches.` For example:



4. Implement a function that gets a set of matching points between two images and calculates the **projective** transformation between them. The transformation should be H_{3x3} homogenous matrix such each key-point in the first image, p, will map to its corresponding key-points in the second image, $\tilde{p}$, according to $p = H\tilde{p}$. How many points are needed?
5. Find all the inliers: Implement a function that finds the transformation according to all the inliers matches, using RANSAC algorithm. At each of the algorithm iteration:
   a. Randomly choose **4** pairs of matching key-points $\{p, \tilde{p}\}$ and calculate the projective transformation according to them, such that for each i $p_i = H\tilde{p}_i$.
   b. Map **all** the coordinates from the first image to the second one. calculates the mapping error.
   c. Save the number of inliers in the current iteration: the number of points that have a mapping error of less than 5.

   Repeat for 1000 iterations. Return the transformation H corresponding to the maximal number of inliers (remember to re-compute it using all the inliers).

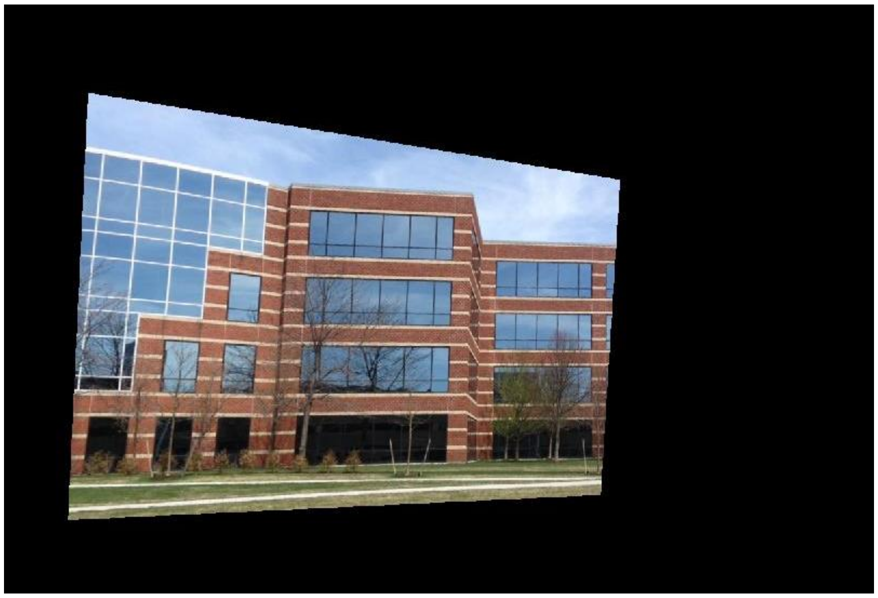
   Show all matching inliers key-points after RANSAC. For example:



useful functions: imshow, hold on/of, plot, mldivide

6. **Image warping:** Implement a function that gets an input image and a projective transformation and returns the projected image. Please note that after the projection there will be coordinates which won't be integers (e.g sub-pixels), therefor you will need to interpolate between neighboring pixels. Project each color channel separately.
7. **Note1:** You will need to pad the input image appropriately so that the transformed image will not get cropped at the edges. The padding doesn't have to be exact, just add enough so that the entire warped image is visible.
   Note2: imwarp() is not allowed. You need to write your own implementation of warping.

Useful functions: interp2



8. **Stitching:** Implement a function that gets two images after alignment and returns a union of the two. The union should be a simple overlay of one image on the other. Leave empty pixels painted black.