HW1 – submission 15.4.19, 23:55

Guide lines

1. Include all your personal details including name, id, and **e-mail address.**
2. You should submit all function and script files written in MATLAB or 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).
3. Please divide the code by questions.
4. 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.**
5. Please post question regarding this HW on the QA forum in Moodle.
6. The grades are highly depended upon the analysis depth of the report.
7. HW can be submitted in pairs.
8. Eventually submit one compressed file including the code + images PDF.

Good luck!

Task 1 –Hough Transform (20 points):

MATLAB provides tools to detect lines in a binary image based on its Hough transform. The purpose of the following exercise is introduce these tools.

Transform

The `hough` computes the Hough transform of a binary image. In the following example, the use of function `hough` is exemplified. Type in the following commands:

```
f=zeros(101,101);
f(1,1)=1;
H=hough(f);
imshow(H,[])
f(101,1)=1;
H=hough(f);
imshow(H,[])
f(1,101)=1;
H=hough(f);
imshow(H,[])
f(101,101)=1;
H=hough(f);
imshow(H,[])
f(51,51)=1;
```

(a) Explain the result of each `imshow` as it appears on the Hough domain.

Let's try a more complex shape now…

(b) Generate a binary image of a square.

(c) Use the canny edge detection to generate an image containing the edges of the square (you may use the function `edge`).

(d) Display the Hough transform of the image, and explain the results.

Voting

The next step is to find line candidates by a voting process.

(e) Lines in the image space are represented as the maximum point of the Hough transform. Explain why.

MATLAB provides us a function named `houghpeaks` to find peaks in the Hough domain. Consult the help file for `houghpeaks` for details.

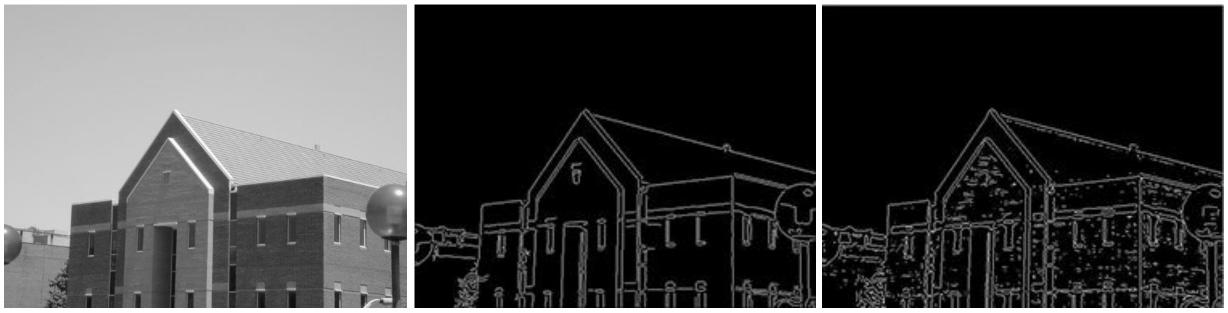
(f) Use the function `houghpeaks` on the Hough transform of the square.

Line Linking

Once a set of candidate peaks has been identified in the Hough domain, it remains to be determined if there are line segments associated with those peaks, as well as start and ending points. The **`houghlines`** function can do this.

(g) Use the function **`houghlines`** and plot the detected lines over the square image.

(h) Reproduce the results for input image **building.jpg**. Try different values of canny edge detector parameters to get the best result.



<u>Task 2 – Laplacian pyramid for style transfer (50 points):</u>

On this question you will implement a style transfer for headshot portraits, inspired by the paper attached to this exercise *"Style Transfer for Headshot Portraits "* (attached).

<u>The goal:</u> 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:

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

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).

<u>Tips:</u>

- 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.

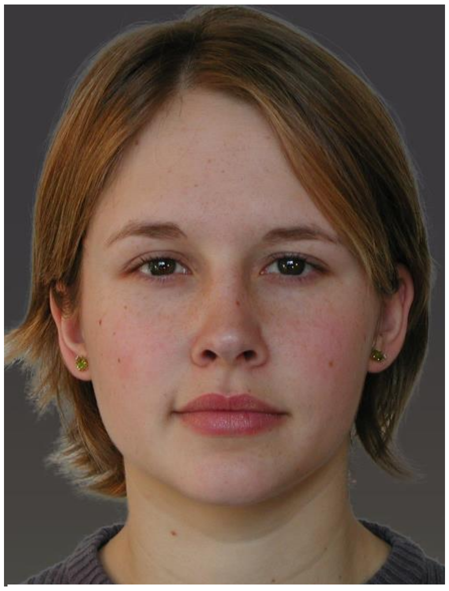
(b) Implement a function that reconstructs an image from its Laplacian pyramid. The function should get the pyramid levels and should return the reconstructed image.

<u>Test:</u> 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 effect 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:

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

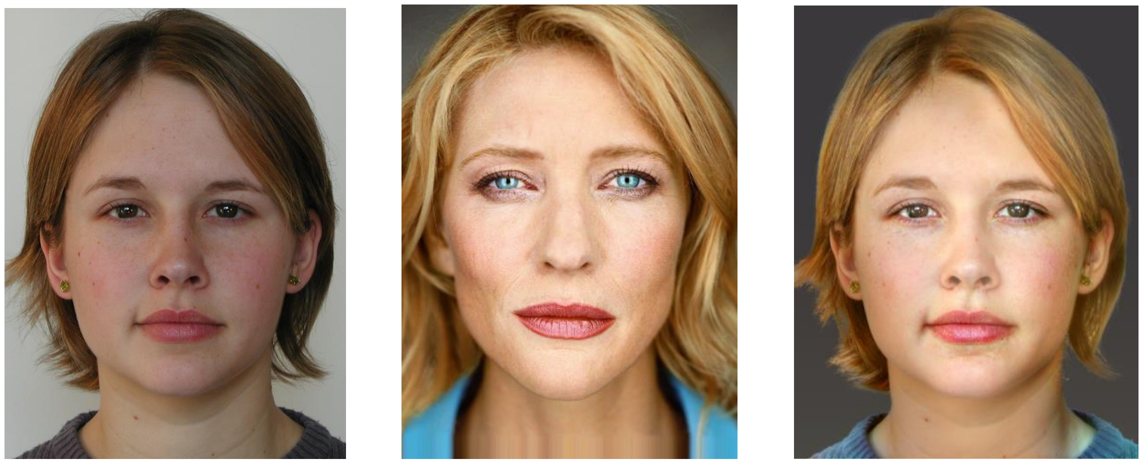
For each of the output image channels (RGB) construct a new pyramids, according to the formalism above. Note: the last pyramid level equals to the last level of the example image.

(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.

Example for the style transfer result:



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