Introduction to Computer Vision

Exercise 4

Due Date: Sunday, Jan. 17, 2018
Submission: in pairs

General Instructions

- The programs should be written in MATLAB. Remember that MATLAB is efficient when the code is vectorized, so use built-in matrix operations and functions instead of loops.

- Write a report that includes a description of your implementation. It ought to include problems you have encountered, their solutions and any assumptions made in your software. In addition, a **serious** discussion of your results must conclude your report. This part ought to explain any problems with your results.
  Any other comments or insights you have gained during your work should find their way to your report. We would like to see how well you understand the subject.

- How and what to submit?
  Please submit your solutions electronically to the following link:

  https://www.dropbox.com/request/zyACKLHqDECubssHzqen

  Using the procedure below:
  1. Create a directory hw4sol-<lastname1>-<lastname2> in which you will work on your solutions. All files mentioned below are assumed to reside in that directory (we will refer to it as the hw4sol directory).
  2. Generate the written parts of your solution in PDF. You can do this any way you want: LATEX, Word or a similar program and convert/export to PDF. Please name this document hw4sol-<lastname1>-<lastname2> .pdf.
  3. Create Matlab code (extension .m), data (.mat), and image (.png, .jpg, etc.) files needed in the hw4sol directory. If file names are not specified explicitly in the assignment, please include a file README with brief description of the files.
  4. When done, create an archive of the entire hw4sol directory (please make sure it contains the directory node itself, not just the files) as .zip or .tar.gz file, and submit it to the link.

Alignment between two images

In this exercise, you should implement the 2D parametric estimation and alignment algorithm that was presented in class. The implementation should be for the case of a global 2D translation (i.e., a global image shift). Note that this is a special case of a homography,

where $H = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix}$

Implement alignment function

Write a function called "align" with prototype:

[shift, warped_im2, abs_diff_im_before, abs_diff_im_after, MSE_before, MSE_after] = align(im1,im2);

Where:

- Inputs:
  - im1 – input image (double grayscale), the reference frame
  - im2 – input image (double grayscale) that will be warped to im1 coordinate system.
- Outputs:
  - Shift - a vector of dimensions 1X2, the 2D shift between the two images (usually a non-integer number). This is the displacement of im2 for aligning it with im1. [Δx,Δy] (Δx horizontal shift, Δy vertical shift)
  - warped_im2 (double) - im2 warped backwards towards im1 according to the computed [Δx,Δy].
  - abs_diff_im_before (double) - An image displaying the absolute differences between the two images before alignment, i.e., of the pixel-wise absolute differences between im1 and im2.
  - abs_diff_im_after (double) - An image displaying the absolute differences between the two images after alignment, i.e., of the pixel-wise absolute differences between im1 and warped_im2.
  - MSE_before - The Mean Square Error over the entire image (i.e., a number per image) for the original images.
  - MSE _ after - MSE over the entire image (i.e., a number per image) for the aligned images in the overlapping areas.

Notes:

- Use a 5-level Gaussian pyramid (original image + 4 extra levels) with parameter a=0.4. Use your code from exercise 2.
  Start the process at the smallest pyramid level and iterate 5 times per level.
  IMPORTANT NOTE: Use the shift from one level as an initialization for the next level. When doing this, remember to scale it accordingly.
- For the derivative calculation use the MATLAB gradient function: [Fx,Fy] = gradient(F)
- After warping the image you will get black pixels at the undefined areas. When computing the shifts you should ignore these areas, i.e. do NOT use those pixels.

- In order to perform the warping between images, you can use the code "shift.m" given.
- You can check the quality of your alignment by viewing the two images after alignment by flickering between them. They should look the same except for areas of object motion.
- You should stick to the function prototype described above, as there will also be an automatic testing of your program.

Apply alignment on image pairs

- Apply your function on the 2 image pairs in the "image pairs" folder.
- For each pair:
    - Save the image after alignment and add to the submission folder.
    - Display in your report:
        - The absolute difference image with the MSE value before alignment in the title
        - The absolute difference image with the MSE value after alignment in the title
        - A graph of the translation calculated in each iteration in each pyramid level (25 in total). Note that the translation value needs to be scaled according to the appropriate pyramid level.

Using alignment for noise reduction

- In this section we provide you with a noisy movie ('input_movie.avi'). The movie is of a static scene and is corrupted with noise. When having multiple repetitions of the same pixel in different frame, we can remove the noise by taking the median of this pixel across all frames .
- In order to do that you need to align all frames to one another. Apply your function "align" on all pairs of frames (frame_12 , frame_i), and replace each pixel with the median of its repetitions across all frames .
- Display frame 12 before and after the noise reduction one next to each other in the following way: figure; imshow([orig_frame_12,clean_frame_12]) and show the result in your report.
- Why do you think you were asked to use median rather than mean for the noise reduction?

Using alignment for removing dynamic parts of the scene

- In this section we provide you with a movie which includes two motions: camera pan and a walking woman ('MICH.avi'). Applying alignment to the entire frame will lock on to the dominant motion (in this case the camera motion). Taking the median of each pixel across all frames would eliminate the walking woman from the entire movie.
- Using the code from the section above, align all the frames to the middle frame and take the median pixel across all frames and plug it back in the corresponding place in **all** the frames

- Backwarp the processed aligned frames to their original coordinate system: this would generate the same video without the walking woman

Note: In order to cope with the large motions in the movie, you should use pyramids with more levels

Good Luck