Introduction to Computer Vision

Exercise 4

Due Date: Sunday, Jan. 15, 2017
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 discussion of your results must be included in 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 understood the subject.
- **How and what to submit?**
  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, MAD_before, MAD_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 1X2 vector [Δx,Δy] containing the computed 2D shift between the two images (Δx horizontal shift, Δy vertical shift - usually non-integer numbers).
  - 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., 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., the pixel-wise absolute differences between im1 and warped_im2.
  - MAD_before - The Mean of Absolute Differences between the two input images (one number, averaged over the entire image).
  - MAD_ after - The Mean of Absolute between the two aligned images (one number, averaged only over the overlapping areas after alignment).

Notes:

- Use a 5-level Gaussian pyramid (original image + 4 extra levels). Use your code from exercise 2 (with the parameter a=0.3).
  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). Note that the gradients on im1 need to be computed only once per pyramid level.
- After warping the image you will get black pixels at the undefined areas (pixels that were beyond the bounding box of im2). 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 **flickering** on the screen between the two images after alignment. They should look aligned, 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 3 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 MAD value before alignment
    - The absolute difference image with the MAD value after alignment
    - A graph of the residual translation calculated in each iteration in each pyramid level (25 in total), expressed in units of the original scale. Note that for this purpose, the translation values need to be scaled up according to the appropriate pyramid level.

Using alignment for noise reduction

- In this section we provide you with a noisy movie of a static scene (corrupted by sensor noise). Having multiple repetitions of the same scene point in different frame allows to remove the sensor noise by taking the median of this scene point across all frames.
- In order to do that you need to align all the frames with respect to one reference frame. Therefore:
  - Apply your function "align" to align all frames in the movie “input_movie” to frame# 15, i.e.: align(frame_15 , frame_i).
    Experiment with 5 iterations per pyramid level to align all the images to Frame 15:
    - Create a back-and-forth movie of the aligned images (1,2,…,20,20,19,…,1), and save it in the submission folder.
    - Flicker between Frame1 and Frame15. Are these two frames perfectly aligned? If not – what do you think is the reason for this?
    - Generate the “median image”, by taking the median of the grayscale values across all the aligned frames for each pixel.
    - View the improvement by **flickering** on the screen between Frame 15 and the median image.
- Display Frame 15 and the median image stacked one above the other in the following way, and add the result in your report:
  figure; imshow([orig_frame_15; clean_frame_15]) .
- Why do you think you were asked to use median rather than mean for the noise reduction? When would you suggest using the mean rather than median for noise reduction?

Note: you can use the functions “VideoReader” and “VideoWriter” to read and write videos. Look at their documentation for more details.

Good luck!