UNIVERSITY OF CALIFORNIA, SANTA BARBARA
Department of Electrical and Computer Engineering

Prof. P. Sen **ECE 178 Digital Image Processing** Fall 2017

Final Project

Due Friday 11:55 pm, December 15

We will be implementing a principled method for image retargeting using the "Patch-Based Bidirectional Similarity" measure. This measure considers two signals S and T to be visually similar if as many as possible patches of S are contained in T, and vice versa. For detailed implementation the students are referred to the publication by Simakov et al. CVPR'08 (`http://www.wisdom.weizmann.ac.il/~vision/VisualSummary.html`). We will be using the objective function as defined in this publication throughout this project.

We will start off by using the public available "PatchMatch" package to implement the retargeting method proposed by of Simakov et al. In the second part we will be implementing the "PatchMatch" algorithm as described in Barnes et al. SIGGGRAPH'09 (`http://gfx.cs.princeton.edu/pubs/Barnes_2009_PAR/index.php`) by ourselves.

1. For the retargeting part, we will implement the algorithm proposed in Simakov et al. CVPR'08. Note that we're going to implement it in a **coarse-to-fine** way, which is different from the fine-to-coarse gradual resizing algorithm (Fig. 4) in the paper. The steps for gradual resizing algorithm are as following:

   (a) **Retargeting at the coarsest scale:** Lets say we have a 194 by 259 image (SimakovFarmer.png) and we want to resize it to 194 by 169 (ratio 1:0.65 = 259:169). To achieve this, first we perform this retargeting operation on a smaller version of the image (coarsest scale). To define the coarsest scale dimension, we predefine the width for the image at coarsest scale to be 30 pixels. This means that the size of the desired retargeting image result at the coarsest scale is 35 by 30 (*ceil*(194*30/169)=35). To perform retargeting at the coarsest scale, first you have to downsample your original image to 35 by 47 (keep the same aspect ratio as the original input image). Then you start the gradual resizing step by doing the patch search and vote. You keep decreasing the image size until you get to 35 by 30. At each time of the resizing process, you fix the height of the image and gradually reduce the width.

   For the patch search and vote process, you can call the function *search_vote_func* provided in the zipped file. It uses the *nnmex* and *votemex* function that can be downloaded from `http://www.cs.princeton.edu/gfx/pubs/Barnes_2009_PAR/patchmatch-2.1.zip`. Note that you can use these two functions from this package at this point, but you have to replace these two functions by your own implementations in part 2 of this project.

   (b) **Gradual resolution refinement:** For this part, we're going to gradually refine the resolution from the coarsest level (35 by 30) to the target resolution (194 by 169). The gradual resolution refinement here keeps the aspect ratio through all the scales. It only improves the image resolution and does not do any retargeting through this process. The retargeting is done only at the coarsest scale. We have define the number of scales in the *FinalProj.m* (numScales = 10), so you have to do the upsampling process by numScales times. For each time of the upsampling

process, you have to upsample the image a little bit by built-in *imresize* , and then refine the image by *search_vote_func* function.

(c) **Final scale refinement:** After you get the output image with size equals to the desired one (194 by 169), you have to call *search_vote_func* again for the final refinement. Once you got all the codes done, you will see the input image (left) and the final retargeting output (right) as following :

2. For this part, you are going to implement functions that can substitute *nnmex* and *votemex* function. You don't have to write it in C/C++. What you have to do is implementing your functions in Matlab and submit all your Matlab codes.

   (a) **Searching NNF step - naive way:** For this part, you have to write your own function to do the searching step and replace the *nnmex* function. For getting the nearest neighbor field (NNF), the naive way is to search through all possible patches in the image and assign the location of the patch which has the lowest L2 distance as the nearest neighbor. You can implement the naive approach first to get familiar with the NNF. The naive approach will take quite long time for the calculation, and we will compare the computation time of naive approach with the accelerated PatchMatch algorithm later.

   (b) **Searching NNF step - PatchMatch:** In order to calculate the NNF efficiently, we are going to implement the the "PatchMatch" algorithm as described in Barnes et al. SIGGGRAPH'09. It's an approximated nearest-neighbor algorithm, which can give us a NNF in a much shorter time compared with naive method. For the details about the algorithm, please refer to the paper.

   (c) **Voting step:** Given the NNFs, we can get the result image by voting process. What the voting process does is actually the same as Eq. 5 in Simakov et al. CVPR'08. Using this formula, both consistency and completeness are taken into consideration. The function you write here is going to replace the *votemex* function.

   When you finish all the codes in this part, you should get results similar to the results you got from using *nnmex* and *votemex*.

So in the end you will have 3 versions of the image retargeting code. The first one, as explained in the first part, uses the *nnmex* and *votemex* functions for search and vote operation. The second one uses your implementation of naive search and your implementation of voting function. The third and final version

uses your implementation of nearest-neighbor search using PatchMatch algorithm and your implementation of voting function.

In addition to the coding part, you have to write a report for your final project. The report should include what are the algorithms you have used, what results did you get, comparison of retargeting results between this method and seam carving method (HW8), and the comparison of computation time between naive approach and PatchMatch algorithm.

Please write your report in SIGGRAPH format. The templates for SIGGRAPH can be found on GS:
Latex: Use *sample-acmtog.tex* to write your report.
Word: Use *WordTemplate.docx* to write your report.

For submitting your code, pack all your files (report and codes) into one compressed file. **DO NOT INCLUDE** the output images and files from other package (e.g. *nnmex*, *votemex*). Last thing, name your zip file in this format: <Perm number>_<First name>_<Last name>. Good luck!