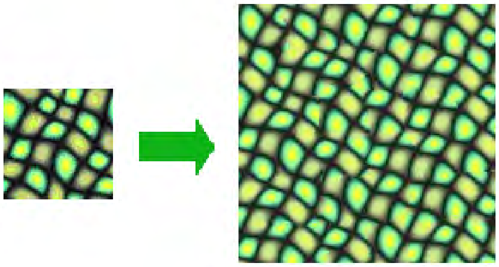
Assignment I : Non-parametric Sampling



There are two parts in this project. In part I, you will be given some texture samples and you need to generate a larger texture image with similar visual appearance to those samples according to the synthesis algorithms we studied in the class. In part II, you will implement a basic version of PatchMatch (which can be used to speed-up texture synthesis) to find similar patches between two images.

Part I. Basic synthesis algorithm:

First, create a large empty image and copy the texture sample to its upper left corner as a starting point for your synthesis program.

Next, iteratively replace blank pixels in the large image with pixels from the sample image. At every iteration, a blank pixel with the largest number of 'known' neighboring pixels is replaced. The pixel selected from texture sample has the closest neighborhood to the blank pixel among all sample pixels. This similarity of neighborhood is evaluated by a Gaussian weighted SSD.

Part II. PatchMatch:

In this part of the project, you are required to find for each patch in image A, a close match in image B. This close match is not exact but approximated by the PatchMatch algorithm. Patches are defined around each pixel in image A and B.

First, for every patch in image A, assign corresponding patch in image B arbitrarily, and store the offsets (u, v) and patch distance D in a 3-channel array. You may use SSD of the color difference to compute D.

Then propagate the offsets in scan order from left to right, top to bottom in odd iteration, and from right to left, bottom to top in even iteration. Replace a patch's offset with its neighbor's if the new offset gives smaller D. For our task, it should suffice to iterate 2 to 5 times.

The random search step is optional. You may refer to sec 3.3.2 in [1] to implement this step if you are keen on improved results.

Visualize the x and y coordinates of the patch correspondence field (plot them as images). Use the pixel color given by (x, y) in image B to reconstruct image A. An example is give below.

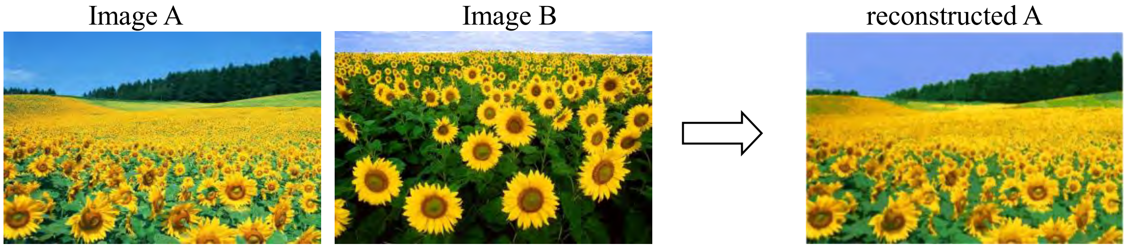


Image A

Image B

reconstructed A

You are required to submit both of your source code (Matlab or C/C++) and report together. Please also submit a **softcopy**. Your report should include a result for each texture sample. The report for this question should be no more than **three** pages. In the report, you are expected to discuss your findings through the experiment. For example, how the neighborhood size affects the synthesis results and speed? What kind of data works best/worst? How the implemented algorithm can be improved in terms of result quality and efficiency (for part I only)?

[1] Barnes, Connelly, et al. "PatchMatch: A randomized correspondence algorithm for structural image editing." *TOG, 2009.*