# Computational Photography - Assignment 3.

March 30, 2008

## 1 Introduction

The objective of this assignment is to implement that texture synthesis paper of Efros and Leung that we covered in class. The paper can be downloaded from the course web-page. Pseudo-code for this algorithm can be found at `http://graphics.cs.cmu.edu/people/efros/research/NPS/alg.html`. Print this out and read through it very carefully. It contains all the details you need to implement the algorithm.

You can perform this assignment in whatever language you prefer. The algorithm is somewhat slow in when implemented in Matlab, hence alternative languages may offer speed benefits. However, it is perfectly possible to do the whole assignment in Matlab, with the code running in reasonable time on the examples provided.

## 2 Objectives

Download the file `assign3.zip` from the course webpage. It contains two small images, `D20.png` and `fill-bread.png`. The objective of this assignment is to implement the texture synthesis algorithm so that you can:

1. Extend `D20.png` by 11 pixels all around. I.e. it is currently 53x49 pixels and you need to add pixels to make it up to 75x71 pixels.

2. Fill in the hole in `fill-bread.png`.

To make things easier (and faster), you can convert the images to grayscale if you prefer. Bonus marks will be awarded for a color version, however.

Please submit filled in versions of the two images along with the source code for your algorithm. Submissions lacking either the images or the code will be penalized.

## 3 Hints'n'tips

Here are a number of hints to help you implement the code in Matlab:

- Use the parameter settings suggested in the pseudo-code, but set `ErrThreshold=0.3`.
- Window sizes in the range 7 – 19 pixels should work well. The smaller the window, the faster the algorithm will run.
- Use the `im2col` function to get all synthesis patches in the image.
- Make sure that no pixels to be filled in are present in any of the synthesis patches.
- Use `fspecial('gaussian',WINDOW_SIZE,SIGMA)` to create `GaussMask`.
- Use `se=strel('square',3);` and `border_mask=imdilate(mask,se)-mask;` to get `PixelList`.
- For speed, try to code the inner loop (`for ii,jj` in pseudo-code) in a single line, i.e. there is no need to loop over all possible patches. You can do this by forming a matrix of size `number_pixels_in_patch` by `number_synthesis_patches` and then replicating the vectorized masks (`ValidMask(:)` and `GaussMask(:)`) `number_synthesis_patches` times.
- Use the `ind2sub` command to map the index of `BestMatch` within the vector of all SSD scores (for all synthesis patches) back to image coordinates. Remember to account for the patch size when doing this.
- To debug the code, it may help to plot the current pixel being filled in and a also rectangle showing the patch chosen on top of the image.

In case of difficulty, first try asking the TA for the class, Denis Kovacs (den.kovacs@gmail.com).