



NUS

National University
of Singapore


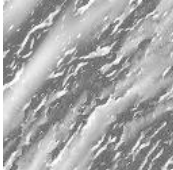



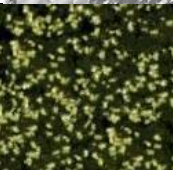
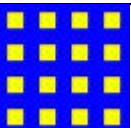
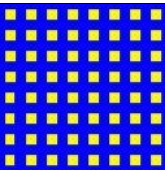


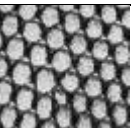
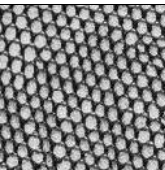
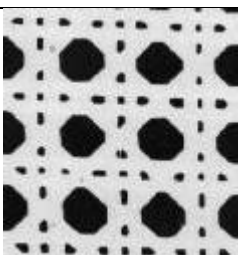
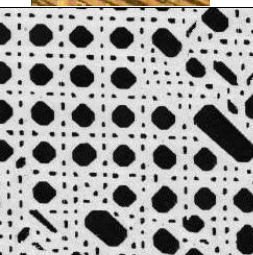

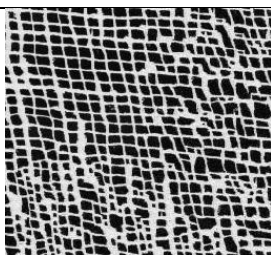
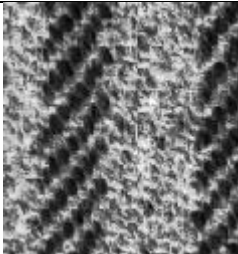
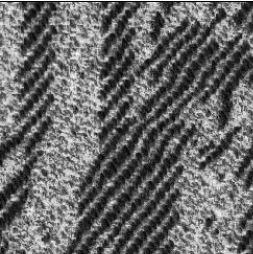



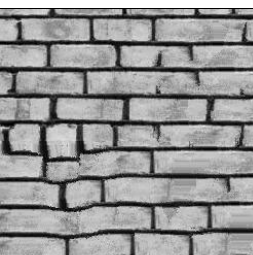
EE4212: Computer Vision

CA2 Assignment 1

Lim Yu Guang

A0172618B

Results

Texture Sample	After Texture Synthesis	Texture Sample	After Texture Synthesis
1. 		2. 	
3. 		4. 	
5. 		6. 	
7. 		8. 	
9. 		10. 	
11. 			

Description of Method

In this assignment, basic synthesis algorithm introduced by Efros and Leung is used to perform the non-parametric texture synthesis. Using this algorithm, a new image grows outward from an initial seed, one pixel at a time, by the texture synthesis process. A Markov random field model is assumed, and the conditional distribution of a pixel given all its neighbours synthesised so far is estimated by querying the sample image and finding all similar neighbourhoods. The algorithm can be summarised in the following pseudo code which will be implemented in MATLAB.

Let `SampleImage` contain the image we are sampling from and let `Image` be the mostly empty image that we want to fill in (if synthesizing from scratch, it should contain a 3-by-3 seed in the center randomly taken from `SampleImage`, for constrained synthesis it should contain all the known pixels). `WindowSize`, the size of the neighborhood window, is the only user-settable parameter. The main portion of the algorithm is presented below:

```
function GrowImage(SampleImage,Image,WindowSize)
while Image not filled do
    progress = 0
    PixelList = GetUnfilledNeighbors(Image)
    foreach Pixel in PixelList do
        Template = GetNeighborhoodWindow(Pixel)
        BestMatches = FindMatches(Template, SampleImage)
        BestMatch = RandomPick(BestMatches)
        if (BestMatch.error < MaxErrThreshold) then
            Pixel.value = BestMatch.value
            progress = 1
        end
    end
    if progress == 0
        then MaxErrThreshold = MaxErrThreshold * 1.1
    end
end
return Image
end
```

Function `GetUnfilledNeighbors()` returns a list of all unfilled pixels that have filled pixels as their neighbors (the image is subtracted from its morphological dilation). The list is randomly permuted and then sorted by decreasing number of filled neighbor pixels. `GetNeighborhoodWindow()` returns a window of size `WindowSize` around a given pixel. `RandomPick()` picks an element randomly from the list. `FindMatches()` is as follows:

```
function FindMatches(Template,SampleImage)
ValidMask = 1s where Template is filled, 0s otherwise
GaussMask = Gaussian2D(WindowSize,Sigma)
TotWeight = sum i,j GaussiMask(i,j)*ValidMask(i,j)
for i,j do
    for ii,jj do
        dist = (Template(ii,jj)-SampleImage(i-ii,j-jj))^2
        SSD(i,j) = SSD(i,j) + dist*ValidMask(ii,jj)*GaussMask(ii,jj)
    end
    SSD(i,j) = SSD(i,j) / TotWeight
end
PixelList = all pixels (i,j) where SSD(i,j) <= min(SSD)*(1+ErrThreshold)
return PixelList
end
```

`Gaussian2D()` generates a two-dimensional Gaussian in a window of given a size centered in the center and with a given standard deviation (in pixels). In our implementation the constant were set as follows: `ErrThreshold = 0.1`, `MaxErrThreshold = 0.3`, `Sigma = WindowSize/6.4`. Pixel values are in the range of 0 to 1.

Discussions

First and foremost, following the pseudo code suggested by Efros and Leung algorithm, I have fixed the following parameters: error threshold = 0.1, max error threshold = 0.3, sigma = window size / 6.4. In order to perform an accurate texture synthesis for different texture sample, the window size needs to be adjusted accordingly and comparable with the size of fundamental components of the texture sample. Furthermore, it can be observed from the experiment that the bigger the window size, the larger the Markov random field, more iterations will be required to synthesise one pixel. This means that longer time will be required to generate the texture synthesised image. For example, in some texture samples, such as texture6.jpg, texture7.jpg and texture11.jpg, the suitable window size is so big that the texture synthesis process becomes extremely slow. In contrast, when the appropriate window size is too small, it will not be able to capture the structure of the texture efficiently. Given a very small window size, the texture synthesis will produce inaccurate results. As such, we face the constraints of quality of the result versus the speed of the texture synthesis process when determining a desired window size (neighborhood size).

Apart from that, the complexity and the size of the sample texture affects the quality of the result and speed of the texture synthesis process. For a more complex texture sample, such as texture11.jpg, a larger window size is required to produce a good result after texture synthesis compared to other texture samples. This is because a larger window size is needed to capture more features or details of the texture sample. However, the speed of the texture synthesis process will be slow because of a large window size used. The size of the texture sample affects the speed of the texture synthesis process considerably. For instance, texture synthesis process for 64x64 texture samples were a lot faster compared to 128x128 texture samples. This is due to a larger area to search when matching neighbourhoods. In addition, the colour of the texture samples will affect the speed of the texture synthesis process. It is observed that synthesising grey texture samples is slower than synthesising coloured texture samples. Perhaps, it is easier to pattern match the neighbourhood of a colored pixel. Therefore, complexity, size and colour of the texture sample will affect the results of the texture synthesis.

Lastly, the implemented basic synthesis algorithm can be improved by synthesising patches (blocks) of pixel instead of pixel by pixel. This will shorten the time taken to find neighbourhoods, and thus, increasing the speed of the texture synthesis process. It will also produce better result depending on the amount of unique features or details in the texture sample. Combined with a proper window size, this enhancement should be able to make the texture synthesis more smooth and real within a short runtime of the algorithm.