Texture Synthesis

Overview

Texture synthesis is the process in which a sample texture is used to generate a larger resolution texture or map the sample texture onto a different image. The technique used in this project is a non-parametric method. This technique grows a new image outward from an initial seed, one pixel at a time. Using the conditional destruction of a pixel given all its neighbors synthesized so far is estimated using a Markov random field model and by querying the sample image and finding all similar neighborhoods. This method aims to preserve as much local structure as possible and seems to produce good results for a wide variety of synthetic and real-world textures.

Algorithm

The algorithm used is the non-parametric texture synthesis technique developed by Efros and Leung. The following is the pseudo-code [1].

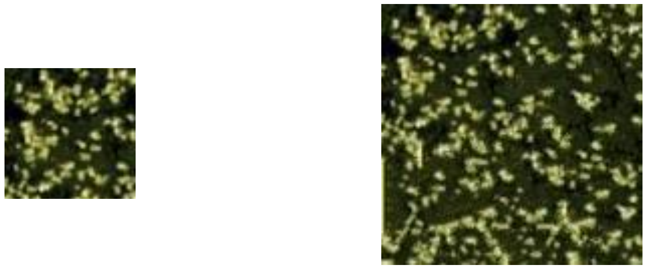
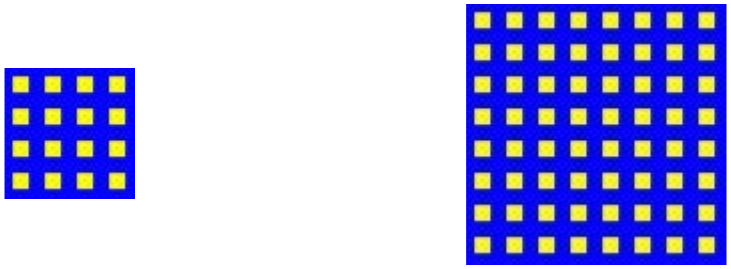
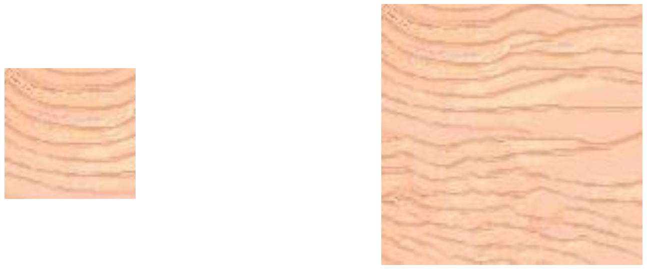
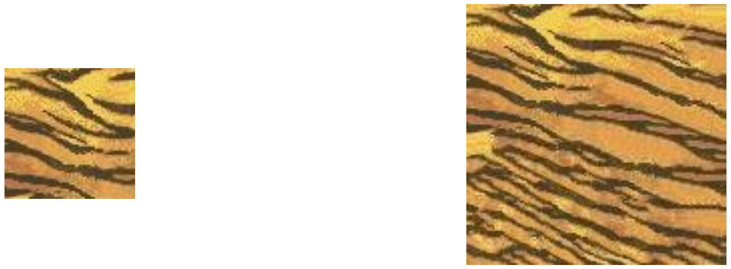
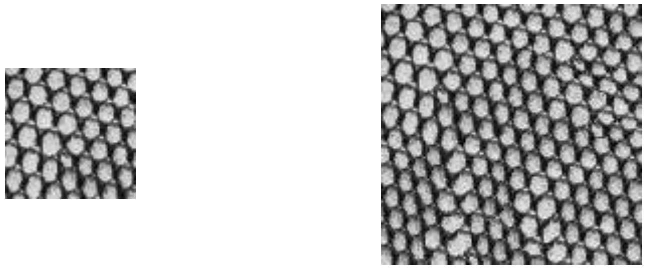
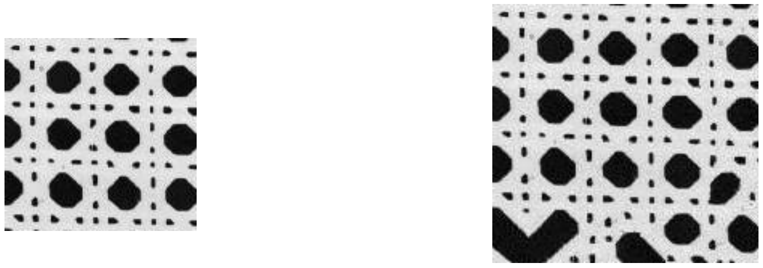
```
function GrowImage(SampleImage,Image,WindowSize)
  while Image not filled do
    progress = 0
    PixelList = GetUnfilledNeighbors(Image)
    foreach Pixel in PixelList do
      Template = GetNeighborhoodWindow(Pixel)
      BestMatches = FindMatches(Template, SampleImage)
      BestMatch = RandomPick(BestMatches)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

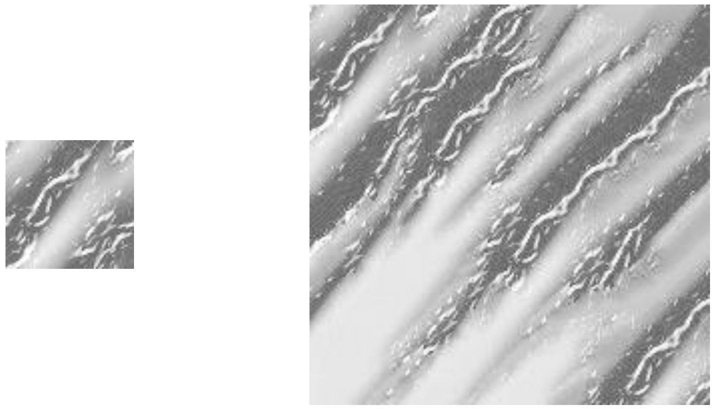
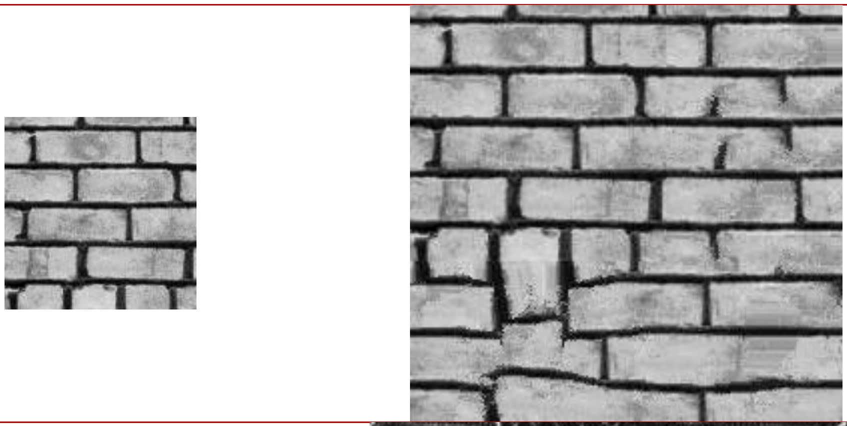
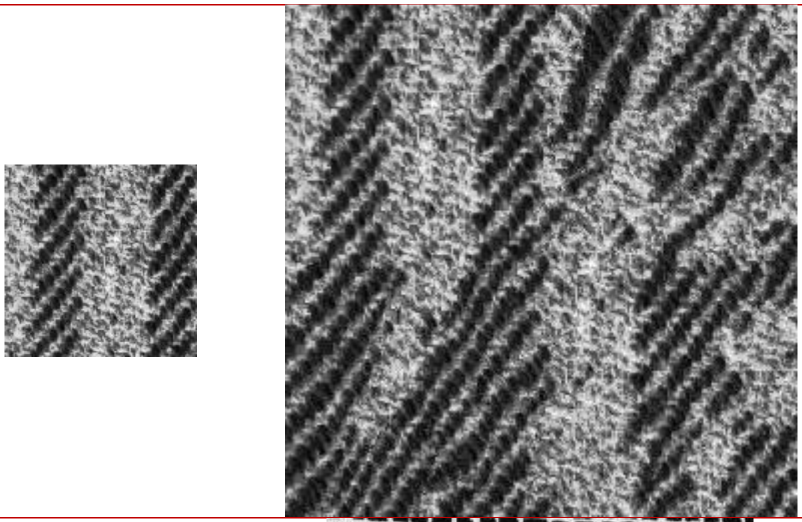
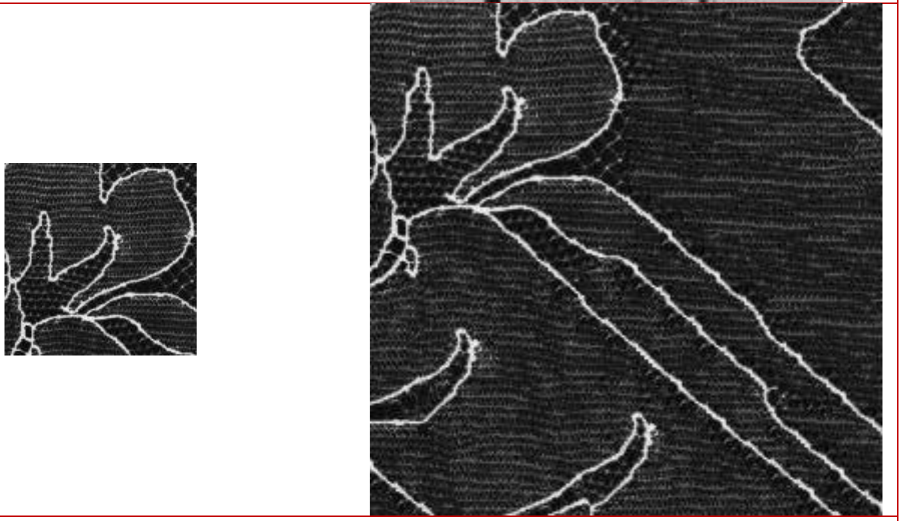
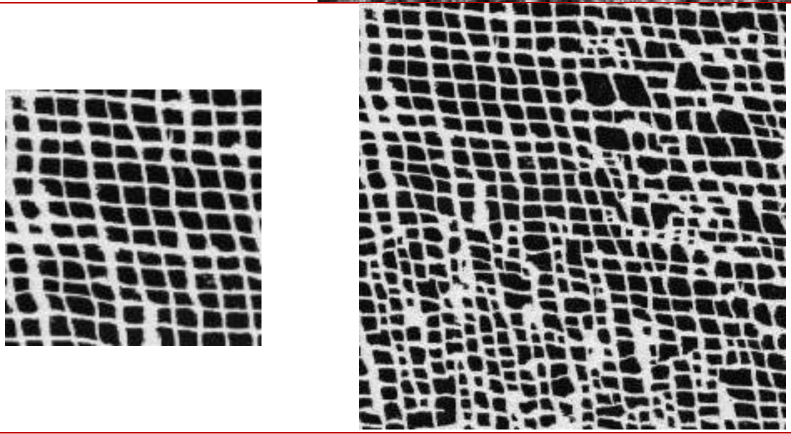
Function GetUnfilledNeighbors() returns a list of all unfilled pixels that have filled pixels as their neighbors (the image is subtracted from its morphological dilation). The list is randomly permuted and then sorted by decreasing number of filled neighbor pixels. GetNeigborhoodWindow() returns a window 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
```

Experimental Results

Results were obtained for the 11 different textures provided using different sampling window sizes.

Discussion & Analysis

Complexity of Textures

The complexity of the textures is a significant limitation. When synthesizing a texture with a constant sampling windows size, the results are not very good. The results can be improved my increasing the sampling window (neighborhood) because this will capture more features of the sample texture. However, this comes with a significant increase in the run time complexity.

Size of Sample Texture

The size of the sample texture greatly affects the speed of the texture synthesis. In the results above, synthesizing from 64x64 texture samples was noticeably faster than synthesizing from 128x128 texture samples. This is because there is a larger area to search when matching neighborhoods.

Size of Neighborhood

The size of the neighborhood or the sampling window size affects both the quality of the result and the runtime of the algorithm. Although a larger sampling window proves to result in textures of higher quality, it comes at the price of a considerably longer runtime.

Grey Scale vs. Color

Synthesizing the texture of grey scale images seems to be slower than synthesizing the texture of color images. This could be because it is easier to pattern match the neighborhood of a colored pixel than a grey scaled pixel.

Verbatim Copies

In texture synthesis, it is often not ideal to produce verbatim copies of the sample texture. Depending on the tolerance parameter, this Efros-Leung algorithm produces verbatim copies.

Run Time Analysis

This algorithm is very time-consuming: generating M pixels from a sample image of N pixels using windows containing K pixels requires O(MNK) operations.

Improvements

Patches vs. Pixels

Perhaps, an improvement for the algorithm would be to synthesize large patches instead of one pixel at a time. This will reduce the time to find neighborhoods and so will increase the run time of the algorithm. It might also produce better results depending on the density of unique features in the sample texture.

Parallelization

The structure of this algorithm is ideal for parallelization as certain pixels can be synthesized independently of each other.

References

[1] A. Efros and T. Leung. Texture synthesis by non-parametric sampling.
http://graphics.cs.cmu.edu/people/efros/research/EfrosLeung.html