EE4212 Computer Vision

Assignment 3: **Non-parametric Texture Synthesis**

Vo Nguyen Khang - U084040W

***Abstract*— Texture synthesis is a method to generate a large texture from a smaller sample. In this assignment, two texture synthesis algorithms (Efros & Leung algorithm [1] and its extension [2]) are implemented and applied on the given texture samples to generate larger texture images with similar visual appearance to those samples.**

I. Basic synthesis algorithm

The basic synthesis algorithm introduced by Efros & Leung [1] is based on the Markov random field (MRF) model. The texture synthesis process grows a new image outward from an initial seed, one pixel at a time. The conditional distribution of a pixel given all its neighbors synthesized so far is estimated by querying the sample image and finding all similar neighborhoods. The algorithm can be summarized as follow:

Algorithm 1 Efros and Leung algorithm

```
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:

Listing 1 `FindMatch()` function

```
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
```

II. Patch based synthesis

Image Quilting [2] is an extended version of the the original Efros & Leung algorithm. From the observation that per-pixel synthesis is very slow, and neighbor pixels are highly correlated; they come up with the idea to make the unit of synthesis as a block of pixels. The MRF property is also applied on this extended version as in the original algorithm, but the extended algorithm is much faster since it synthesizes all pixels in a block at once. The complete quilting algorithm is as follows:

Algorithm 2 Image Quilting

- Pick size of block and size of overlap.
- Synthesize blocks in raster order.
- Search input texture for the block that is most consistent to the synthesized result at overlap regions.
- Compute the error surface between the newly chosen block and the old blocks at the overlap region. Find the minimum cost path along this surface and make that the boundary of the new block. Paste the block onto the texture. Repeat.

The process of finding the minimum error boundary cut between two overlapping blocks on the pixels can easily be done with dynamic programming, as showed in Listing 2. To speed up the block searching process, the approximate nearest neighbor search algorithm can be used.

Listing 2 Finding Min-cut using dynamic programming

1) Calculate the difference at every pixel of the overlapping region.
2) Copy the difference to the cost at the first row.
3) Compute the cost of arriving at all pixels row by row from top to bottom. The cost at $(i, j+1)$ pixel $E_{i,j+1}$ is $E_{i,j+1} = E_{i,j+1} + min\{E_{i-1,j}, E_{i,j}, E_{i+1,j}\}$.
4) Trace back for the optimal seam.

III. Result and discussion

A. Basic synthesis algorithm

In this assignment, the parameters is set as follow: $ErrThreshold = 0.1$, $MaxErrThreshold = 0.3$, $Sigma = WindowSize/6.4$ as in the original paper. The window size needs to be comparable with the size of the basic components of the texture in order to be synthesized correctly. However, the bigger the window size, the larger the MRF, and the more iterations are needed to synthesize one pixel. In some cases, the appropriate window size is so big that the synthesis process becomes unacceptably slow. However, If the window size is too small to capture the structure of the texture, the synthesize algorithm will generate bad results.

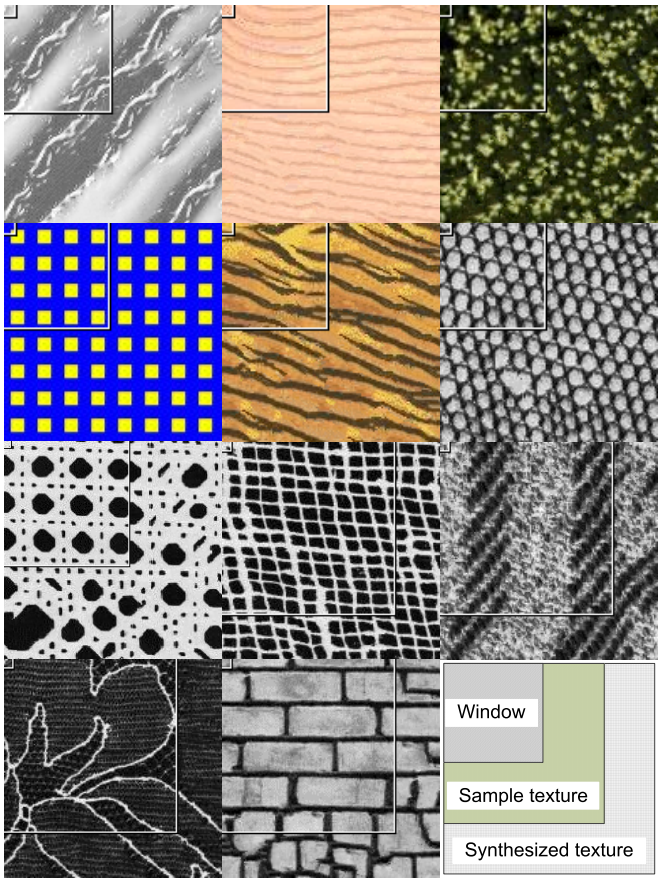

Fig. 1: Basic synthesis results

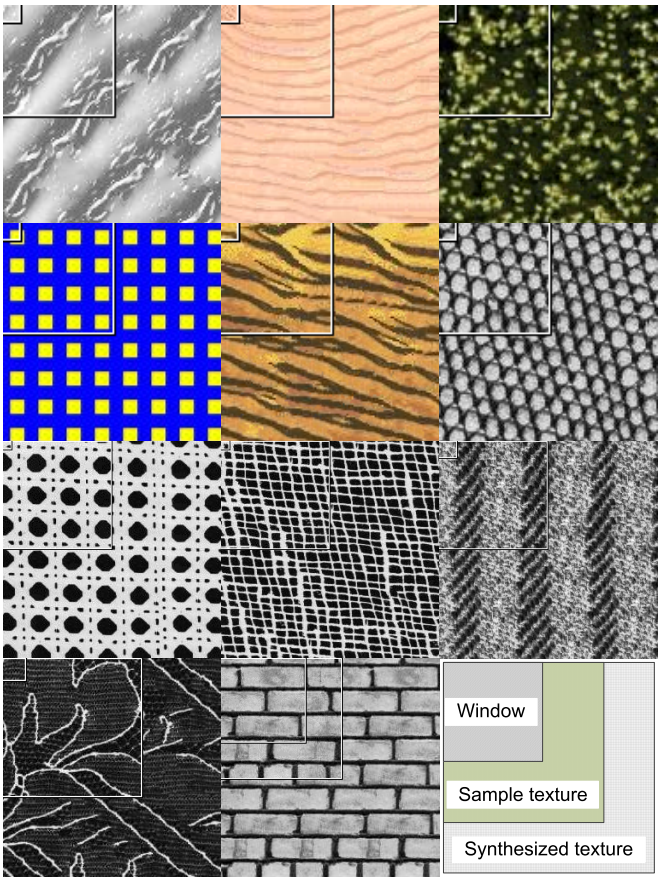

Fig. 2: Patch based synthesis results

B. *Patch based synthesis*

Again, the relative sizes between the block size and the texture's structure is very important. A big block and small overlap region will make the synthesis faster, but in some cases, the quality of the synthesized texture decreases. Unlike the basic synthesis algorithm, if the block size is set too big, the assumption on the correlation between the pixels of the same block is not valid; therefore, setting the block size to big will have the adverse effect on the quality of the synthesized texture. A bigger overlap region makes it easier to find an optimal minimum cut and consequently creates seamless texture, but the big overlap region also decreases the synthesize speed since fewer new pixels will be synthesized in each patch. Comparing the results of the two algorithm, we see that patch based synthesis using image quilting is superior to the basic synthesis in terms of speed and quality.

C. *Suggestions*

Currently, the synthesize speed is slow mostly because the MALTAB code is not fully vectorized. Besides, the approximate nearest neighbor search can be used to reduce the patch searching time. For better synthesis, the algorithm needs to be able to determine the optimal size of the window or the patch.

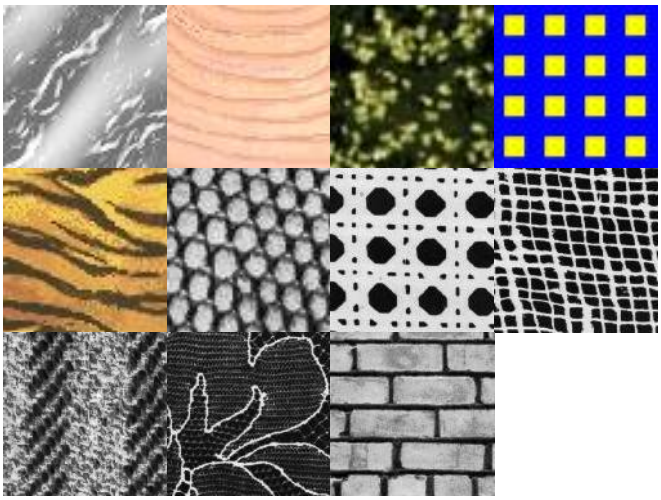

Fig. 3: Sample textures

REFERENCES

[1] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *IEEE International Conference on Computer Vision*, Corfu, Greece, September 1999, pp. 1033–1038.

[2] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," *Proceedings of SIGGRAPH 2001*, pp. 341–346, August 2001.