Computer Vision
Homework 05: Image modeling, Linear Filtering
CS 670, Fall 2019

Name: Kunjal Panchal
Student ID: 32126469
Email: kpanchal@umass.edu

Oct 11, 2019

Contents

1 Problem 1: Window Size in the Texture Synthesis by Efros and Leung

First, we take a look at some basic definitions and concepts pertaining the problem at hand:

1.1 Definitions and Concepts

- MARKOV CHAIN is a mathematical system that experiences transitions from one state to another according to certain probabilistic rules.

  This is used to synthesize text.

  a Markov chain program typically breaks an input text (training text) into a series of words, then by sliding along them in some fixed sized window, storing the first N words as a prefix and then the N + 1 word as a member of a set to choose from randomly for the suffix. [2]

- MARKOV RANDOM FIELD is used in image processing to generate textures as they can be used to generate flexible and stochastic image models.

  In image modeling, the task is to find a suitable intensity distribution of a given image, where suitability depends on the kind of task and MRFs are flexible enough to be used for image and texture synthesis. [3]

- TEXTURE SYNTHESIS is the process of algorithmically constructing a large digital image from a small digital sample image by taking advantage of its structural content. See Figure 1 We want to insert pixel intensities based on existing nearby pixel values.

- EFROS & LEUNG paper [1] a non-parametric method for texture synthesis. The texture synthesis process grows a new image outward from an initial seed, one pixel at a time.

  A Markov Random Field model is assumed, and the conditional distribution of a pixel given all its neighbors synthesized so far is estimated by querying the sample image and finding all similar

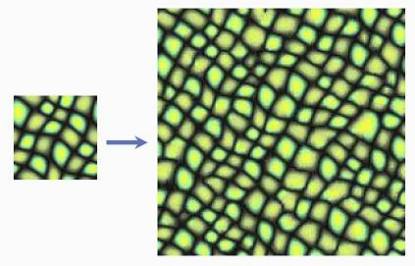

Figure 1: A higher resolution image synthesized with its texture and patterns from a lower resolution texture corpus

neighborhoods.

The method aims at preserving as much local structure as possible and produces good results for a wide variety of synthetic and real-world textures.
See Figure 2

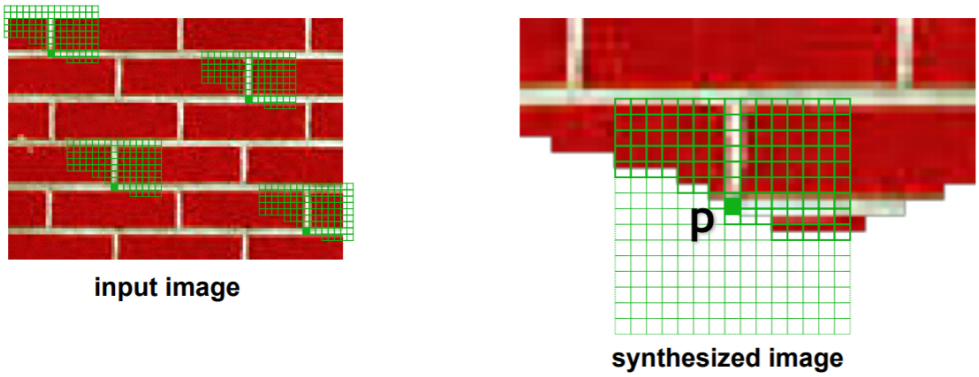

Figure 2: A higher resolution image synthesized with its texture and patterns from a lower resolution texture corpus by markov random field pixel window matching algorithm

1.2 Window Size for Pixel Pattern Matching

In order to guess a new pixel value, we look at all its neighbors in a window of fixed size, the most similar window will decide what the concerned pixel should have as its value.

The only parameter set by the user is the width w of the context window.

" This parameter appears to intuitively correspond to the human perception of randomness for most textures." [1]

In Figure 3, the image corpus has been synthesized four times; each with increasing window size.

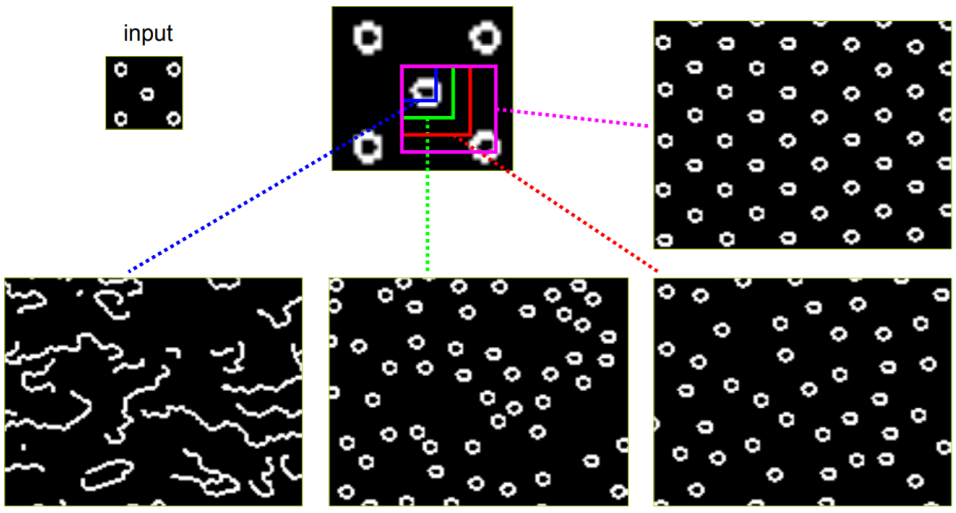

Figure 3: Given a sample image (top left and middle[enhanced]), the algorithm synthesized four new images with neighborhood windows of width 5, 11, 15, and 23 pixels respectively. Perceptually, intuitively the window size corresponds to the degree of randomness in the resulting textures.[1][2]

- BLUE WINDOW: SIZE - 5 PIXEL In this synthesized image the pattern matching window is not big enough to capture the structure of the ring so only the notion of curved segments is preserved.
- GREEN WINDOW: SIZE - 11 PIXEL Here, the window captures the whole ring, but knows nothing of interring distances producing a pattern. Or simply put, the distance between two rings is unknown, thus the rings are synthesized at random places.
- RED WINDOW: SIZE - 15 PIXEL In the third image we see rings getting away from each other as the window is capturing more of the

space where distance between two rings has no other pattern. It still doesn't tell us how far the next ring will be encountered. So, the synthesized image still have rings at random places, but now with a little more distance between them.

- PURPLE WINDOW: SIZE - 23 PIXEL Finally, the inter-ring structure is within the reach of the window as the pattern becomes almost purely structured.

It might also happen that, with a particular window size, the "wrong", "garbage" or "bad" parts of the images are amplified more and more as synthesis goes on.

Some textures to occasionally "slip" into a wrong part of the search space and start growing garbage or get locked onto one place in the sample image and produce verbatim copies of the original. See Figure 4 and 5.

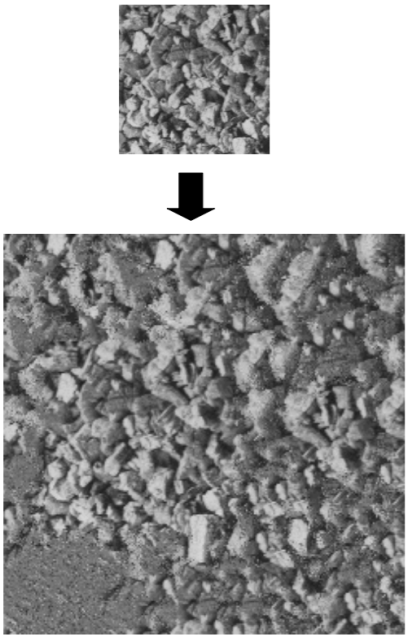

Figure 4: Failure example: Sometimes the growing algorithm "slips" into a wrong part of the search space and starts growing garbage.[1][2]

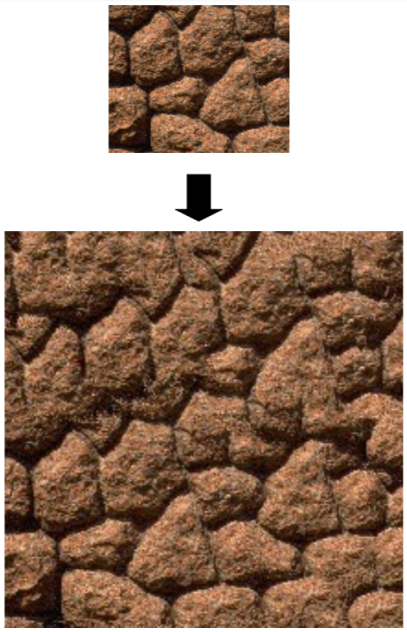

Figure 5: Failure example: Sometimes the growing algorithm gets stuck at a particular place in the sample image and starts verbatim copying.[1][2]

1.3 Conclusion

For the above given reasons (Symmetry of image, Regularity of the patterns, Garbage Amplification, Verbatim Synthesis), we can say that the only parameter of Efros and Leung texture synthesis algorithm, the window size, plays a major role in determining how the final output will look in its symmetry and patterns.

2 Linear Filtering

2.1 (a) Spatial gradient along X axis

We will use Sobel Operator as showed below:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function.

To calculate gradient along X axis, we need to go column-wise. That's why the matrix is in "columns".

The middle column is all 0s as we are taking gradient over X axis. (The difference between two column neighbor is calculated)

The result of this filtering is shown in Figure 6.

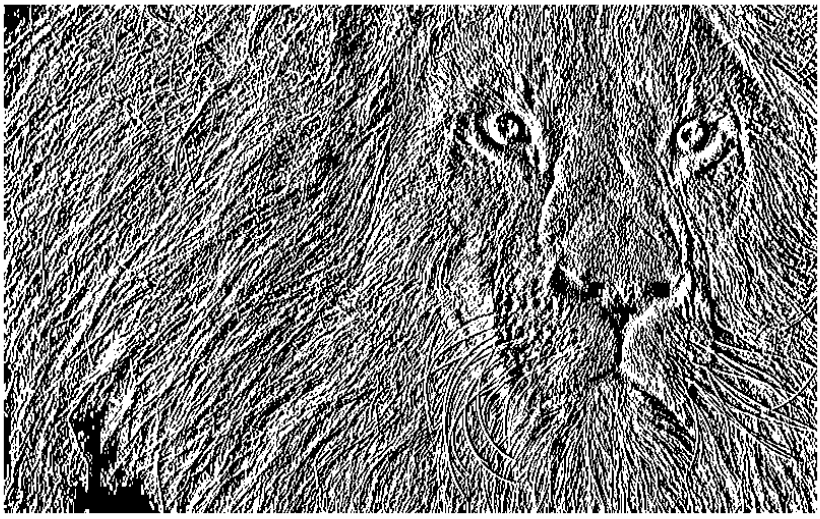

Figure 6: Spatial Gradient along X axis of the lion.png image

2.2 (b) Spatial gradient along Y axis

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

To calculate gradient along Y axis, we need to go row-wise. That's why the matrix is in "rows".

The middle row is all 0s as we are taking gradient over Y axis. (The difference between two row neighbor is calculated)

The result of this filtering is shown in Figure 7.

The most apparent difference between X and Y gradient can be seen

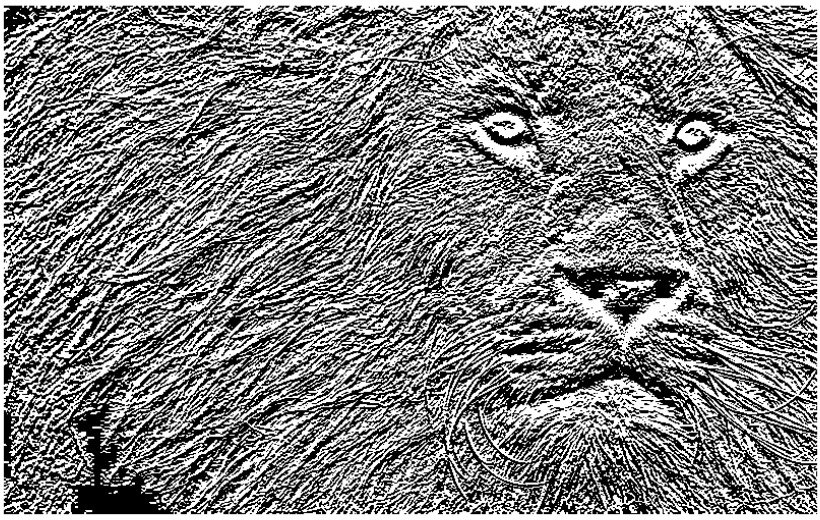

Figure 7: Spatial Gradient along Y axis of the lion.png image

around the eyes of the lion, as the hair are pretty much in criss-cross pattern.

2.3 (c) Average Values at Each Pixel Location

We create a 5 X 5 matrix filter of all ones. And divide it by the sum of values of all elements; which would be 5 X 5 times 1 = 25.

That's how we can get average value of a pixel, considering the values of its neighbors in a 5 X 5 window.

$$\frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The result is shown in Figure 8.

Figure 8: Average Values at Each Pixel Location of the lion.png image

References

[1] Alexei A. Efros and Thomas K. Leung, *Texture Synthesis by Non-parametric Sampling* https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/papers/efros-iccv99.pdf

[2] Subhransu Maji, *Modeling Images* https://www.dropbox.com/s/7a5yp8jpw3s4xi0/lec08_modeling_images.pdf?dl=0

[3] Kindermann and Snell, Ross and Laurie, *Markov Random Fields and their Applications* https://en.wikipedia.org/wiki/Markov_random_field ISBN 978-0-8218-5001-5