ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 3, March 2012

Real Time Image Enhancement Using Texture Synthesis (RETS)

Parth Bhatt, Ankit Shah, Sunil Pathak

Abstract*—Real time Enhancement using texture synthesis combines interpolation, classification and patch based texture synthesis to enhance low resolution imagery. RETS uses a low resolution source image as input and several high resolution sample textures. The output of RETS is a high resolution image with the structure of the source image, but with detail consistent with the high resolution sample textures. Image Interpolation: Interpolation is the primary technique used for image scaling. Image scaling is the process of taking a source image and extending it to create a large image. The primary problem with enlarging images using interpolation is that the large result contains the same amount of discrete data as smaller source image [25]. Two types of interpolation are bilinear and bicubic. Bilinear interpolation uses 2x2 neighbourhoods of data points to calculate pixel color between data points. Bicubic interpolation uses 4x4 neighbourhoods of data points to calculate pixel color between data points [16].Using texture Synthesis solve two problems

- ***It allows to user to specify the detail to be inserted into the output image by providing a representative sample for the system to replicate.***
- ***Applying texture synthesis appropriately will allow us to avoid the unnatural repetition of texture tiles that can occur with standard texture mapping.***

I. INTRODUCTION

High-quality texture can be synthesized in real-time. A key ingredient of the algorithm we propose is a patch-based sampling scheme that uses texture patches of the sample texture as building blocks for texture synthesis.

The advantages of patch-based sampling include

- **Speed:** For synthesizing textures of the same size and comparable (or better) quality, our algorithm is orders of magnitude faster than existing texture synthesis algorithms, including TSVQ-accelerated non-parametric sampling. As a result, high-quality texture synthesis is now a real-time process on a midlevel PC.
- **Quality:** The patch-based sampling algorithm synthesizes high-quality textures for a wide variety of textures ranging from regular to stochastic. Like, that is also a greedy algorithm for non-parametric sampling. However, the patches in sampling scheme implicitly provide constraints for avoiding garbage. For this reason, algorithm continues to synthesize high-quality textures even when and cease to be effective. For natural textures, the results of patch-based sampling look subjectively better.

A. Patch Based Sampling

The patch-based sampling algorithm uses texture patches of the input sample texture I_{in} as the building blocks for constructing the synthesized texture I_{out} In each step, paste a patch B_k of the input sample texture I_{in} into the synthesized texture I_{out}. To avoid mismatching features across patch boundaries, select B_k based on the patches already pasted in I_{out}, $\{B_0 \ldots B_{k-1}\}$. The texture patches are pasted in the order. For simplicity, use square patches of a prescribed size w_B x w_B.

B. Patch Based Sampling Algorithm

i) Randomly choose a w_B x w_B texture patch B_0 from the input sample texture I_{in}. Paste B_0 in the lower left corner of I_{out}. Set k = 1.

ii) Form the set Ψ_B of all texture patches from I_{in} such that for each texture patch of Ψ_B, its boundary zone matches E^o_k ut.

iii) If Ψ_B is empty, set $\Psi_B = \{B_{min}\}$ where $\{B_{min}\}$ is chosen such that its boundary zone is the closest to E^o_k ut.

iv) Randomly select an element from Ψ_B as the k^{th} texture patch B_k. Paste B_k onto the output texture I_{out}. Set k= k+1.

v) Repeat steps (b), (c), and (d) until I_{out} is fully covered.

vi) Perform blending in the boundary zones.

The patch-based sampling algorithm is easy to use and flexible. It can generate tileable textures if so desired. It can be used for constrained synthesis as well. The algorithm has an intuitive randomness parameter. The user can use this parameter to interactively control the randomness of the synthesized texture.

Algorithm combines the strengths of nonparametric sampling and patch-pasting. In fact, both patch-pasting and the pixel-based non-parametric sampling are special cases of the patch based sampling algorithm. The patches in our sampling scheme implicitly provide constraints for avoiding garbage. For this reason, algorithm continues to synthesize high-quality textures even when cease to be effective. For natural textures, the results of patch-based sampling look subjectively better [12].

II. SEGMENTATION

Image segmentation is to cluster pixels into salient image regions, i.e., regions corresponding to individual surfaces, objects, or natural parts of objects. Segmentation could be used for object recognition, occlusion boundary estimation within motion or stereo systems, image compression, image editing, or image database look-up. Image Segmentation is a subset of an expansive field of Computer Vision which deals with the analysis of the spatial content of an image. In particular, it is used to separate regions from the rest of the image, in order to recognize them as objects. It is a method used in the vast field of Artificial Intelligence.

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 3, March 2012

Region Growing is an approach to image segmentation in which neighbouring pixels are examined and added to a region class if no edges are detected. This process is iterated for each boundary pixel in the region. If adjacent regions are found, a region-merging algorithm is used in which weak edges are dissolved and strong edges are left intact. Region Growing offers several advantages over conventional segmentation techniques. Unlike gradient and Laplacian methods, the borders of regions found by region growing are perfectly thin (since we only add pixels to the exterior of our region) and connected. The algorithm is also very stable with respect to noise. Our region will never contain too much of the background, so long as the parameters are defined correctly. Other techniques that produce connected edges, like boundary tracking, are very unstable. Most importantly, membership in a region can be based on multiple criteria. We can take advantage of several image properties, such as low gradient or gray level intensity value, at once.

There are, however, several disadvantages to region growing. First and foremost, it is very expensive computationally. It takes both serious computing power (processing power and memory usage) and a decent amount of time to implement the algorithms efficiently.

A. Texture Segmentation

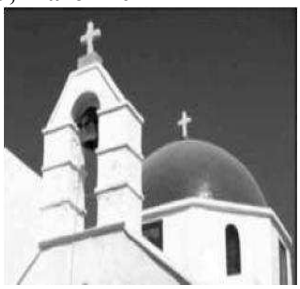
The purpose of texture segmentation is to differentiate textured regions from the rest of the image, which include smooth regions and well-defined edges such as object boundaries. Noticing the fact that textured regions are usually covered with dense edges when performing the edge detection, we use the local edge pixel number as a feature for texture segmentation.

B. Edge Detection

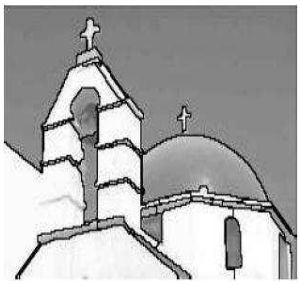
By applying the horizontal and vertical Sobel operators to the luminance channel Y of the image, we can obtain the gradient magnitude and direction for each pixel. Pixels with gradient magnitude larger than a certain threshold T_g are initialized as edge pixels. Since a lot of textures in video images are blurred, in order to obtain a sufficient number of edge pixels in textured regions for the purpose of segmentation, we set a low threshold $T_g = 10$ based on experiments. Raw edge maps usually contain thick edges and are not suitable to be directly used for feature extraction and edge-based interpolation. Therefore we carry out non-maximal suppression to thin the edges. This operation re-classifies an edge pixel to be non-edge if its gradient magnitude is less than that of either of its two neighbours along the gradient direction.

C. Example of Segmentation

Simple Scenes Segmentations of simple gray-level images and color images (RGB images) can provide useful information about the surfaces in the scene.

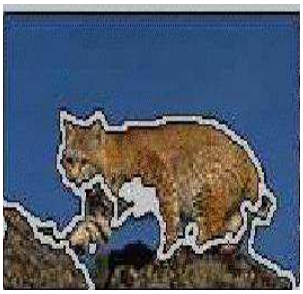


(a) Input Image



(b) Segmented Image

(a) Input Image



(b) Segmented Image

Fig. 1. Examples of Segmentation

III. IMPLEMENTATION

For implementation select one image and then add some noise in that image and taken that image as an input image. Then after removing the noise from that input image and get the output image. Below implementation of the methods for removing the noise from that input image.

A. Fixed Patch Size

Here taken noise patch is fixed size.

PIXEL BASED TEXTURE SYNTHESIS:

A pixel value at a certain location depends only on its immediate neighbourhood. In this approach, for a certain percentage of the selections, use the next column neighbour pixel.

There are two common steps:

a. Searching for the best match for the current output neighbourhood within the sample texture
b. Merging a patch or a pixel with the synthesized output texture.

After these two steps we get an output image which is noiseless.

Patch Based** Texture **Synthesis:

In patch based approach synthesis the result image by stitching together small patches selected from the sample image. In this method synthesis a result image block by block in raster order. Square blocks are used to capture the primary pattern in the sample texture.

There are two common steps:

a. Searching for the best match for the current output neighbourhood within the sample texture
b. Merging a patch or a pixel with the synthesized output texture.

After these two steps we get an output image which is noiseless.

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 3, March 2012

NOISE COMPARISION:

After analyzing both the output images (Pixel based and Patch based) it can be concluded that Patch Based Texture Synthesis is better than Pixel Based Texture Synthesis because in Pixel Based Texture Synthesis the portion of noise is not completely removed so it can be possible that some information will be lost. Patch Based Texture Synthesis gets good effect after removing the noise in image compare to Pixel Based Texture Synthesis.

B. Different size of Patch

Taking different size of matrix n x n as noise (10 x 10, 20 x 20) in image and remove that noise and compare that in which method removal of noise is better.

(a) Input Image **(b) pixel Output** **(c) Patch Output**

Fig. 2. Remove noise

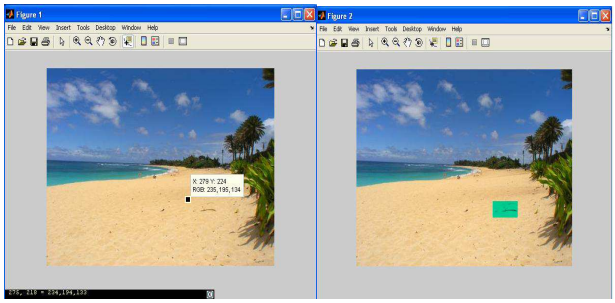
In Pixel Based Texture Synthesis some Information is lost because of different size of patch is taken. Using Patch Based Texture Synthesis information is not lost and gives better output than the Pixel Based Texture Synthesis.

C. Program Complexity

In Pixel Based Texture Synthesis taking One by one pixel and then find it's near neighbour pixel and replaced it. So, program complexity is θ (n^2). In Patch Based Texture Synthesis taking a group of pixels. So not taking one by one pixel so complexity is θ (n^2) – Σ (x x y) where x is a patch size and y is number of patches. So, Patch Based Texture Synthesis complexity is less than Pixel Based Texture Synthesis.

D. Color Image

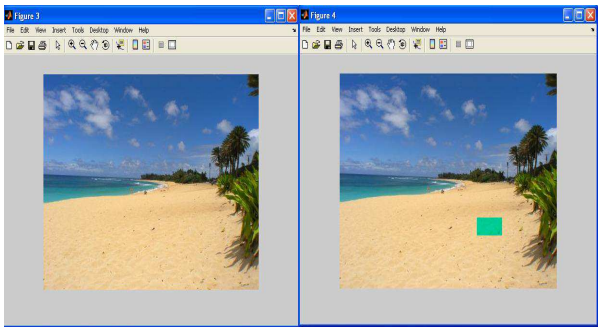
CLEANING IMAGE BY SELECTING AREA:



(a) Input Image **(b) selected area**

Fig. 3. Select Area

We can get area which is to be replaced by selecting four points. By applying Texture on that area we can get the output in which that area is replaced by surrounding pixels to clean that area.In Fig. 4 (a) shown that area is cleaned and in Fig. 4 (b) it is highlighted.



(a) Cleaning Area **(b) Output Image**

Fig. 4.Cleaning Portion

In Fig. 5 first select an area enclosed by arrow and in the output image that area is removed.

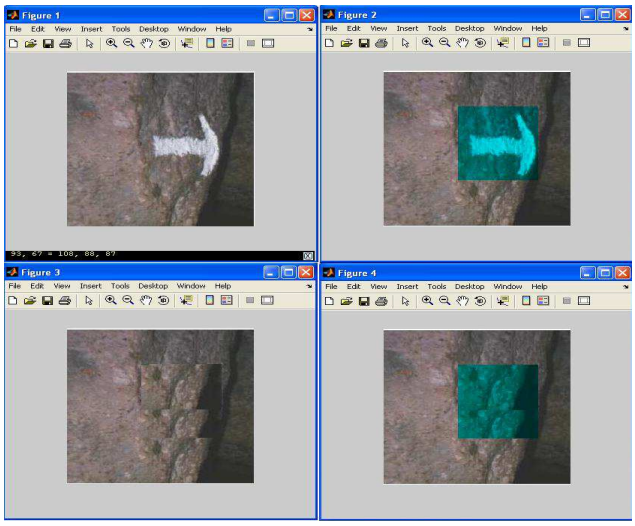


Fig. 5. Example of Cleaning Image

CLEANING IMAGE BY EDGE DETECTION:

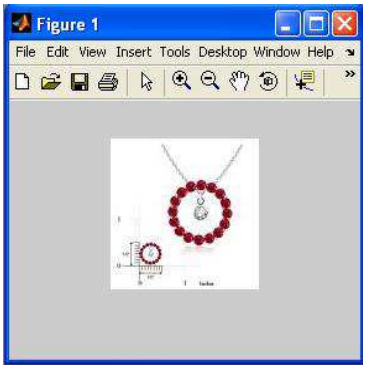


Fig. 6. Input Image

By applying the horizontal and vertical Sobel operators to the luminance channel Y of the image, we can obtain the gradient magnitude and direction for each pixel. Pixels with gradient magnitude larger than a certain threshold are initialized as edge pixels. Since a lot of textures in video images are blurred, in order to obtain a sufficient number of

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 3, March 2012

edge pixels in textured regions for the purpose of segmentation.

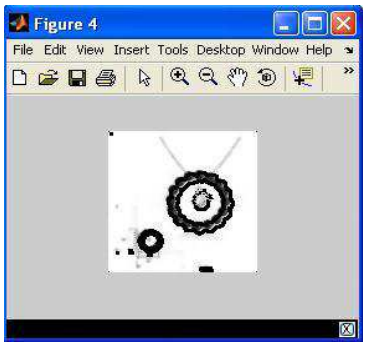


Fig. 7. Edge Detection

After Finding the Edge of all objects choose one pixel of that object which we want to replaced by other texture. So, after choosing the pixels check the surrounding pixel values. When we get change in that surrounding pixel value we have to stop this process and replace selected pixel by surrounding values.

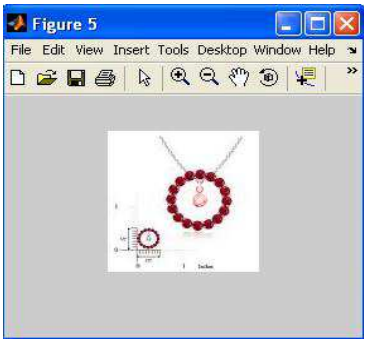


Fig. 8. Output Image

IV. CONCLUSION

The Patch Based Texture Synthesis is finer and better than Pixel Based Texture Synthesis. With the use of Patch Based Texture Synthesis, removal of noise is more efficient than the Pixel Based Texture Synthesis. Some of the information is lost sometimes in Pixel Based Texture Synthesis. Whereas Compared to Pixel Based Texture Synthesis, the loss of information is less in Patch Based Texture Synthesis. There is less difference between Patch Based Texture Synthesis and Pixel Based Texture Synthesis in PSNR (Peak Signal to Noise Ratio) but in case of Program complexity, it is less in Patch Based Texture Synthesis than Pixel Based Texture Synthesis.

Applying Pixel based Texture Synthesis on selected area some information is lost even though it is not an object in color image. In Segmentation we have to find the edges of objects only, so the selecting and cleaning that object give better result compare to selecting an area of the image.

REFERENCES

[1] J. F. Blinn and M. E. Newell. Texture and reaction in computer generated images. Communications of the ACM, (19):542546, 1976.

[2] E. Catmull. A Subdivision Algorithm for Computer Display of Curved Surfaces. Phd thesis, Computer Science Department, University of Utal, Salt Lake City, Utah, 1974.

[3] Martin Szummer and Rosalind W. Picard. Temporal texture modeling. Inter- national Conference on Image Processing, 3:823{826, Sep 1996.

[4] Maneesh Agrawala Andrew C. Beers and Navin Chaddha. Rendering from compressed textures. Proceedings of SIGGRAPH 96, pages 373-378, August 1996.

[5] Alexei Efros and Thomas Leung. Texture synthesis by non-parametric sampling. International Conference on Computer Vision, 2:1033{1038, Sep 1999.

[6] Homan Igehy and Lucas Pereira. Image replacement through texture synthesis. International Conference on Image Processing, 3:186{189, Oct 1997.

[7] Li-Yi Wei. Texture Synthesis By Fixed Neighborhood Searching. PhD thesis, STANFORD UNIVERSITY, November 2001.

[8] Lucas Pereira Homan Igehy. Image replacement through texture synthesis. Computer Science Department, Stanford University.

[9] LI-YI WEI. Deterministic texture analysis and synthesis using tree structure vector quantization. Gates Computer Science Building, Stanford University, a 94309, U.S.A., (386).

[10] Li-Yi Wei Marc Levoy. Fast texture synthesis using tree-structured vector quantization. Stanford University.

[11] Vivek Kwatra Greg Turk4 Li-Yi Wei, Sylvain Lefebvre. State of the art in example-based texture synthesis. The Eurographics Association, 2009.

[12] Yingqing Xu Baining Guo Lin Liang, Ce Liu and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. Technical Report, (MSR-TR-2001-40), March 2001.

[13] David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. SIGGRAPH 95 Conference Proceedings, pages 229-238, Aug 1995.

[14] Jeremy S. De Bonet. Multiresolution sampling procedure for analysis and synthesis of texture images. SIGGRAPH 97 Conference Proceedings, pages 361-368, August 1997.

[15] E. Simoncelli and J. Portilla. Texture characterization via joint statistics of wavelet coefficient magnitudes. Fifth International Conference on Image Processing, 1:62{66, Oct 1998.

[16] Matthew Sorenson. Real time image enhancement using texture synthesis. November 2004.

[17] G. Caenen L. Van Gool A. Zalesny, V. Ferrari. Composite texture synthesis. International Journal of Computer Vision, 2004.

[18] D. Scharstein. Synthesis Algorithm, volume 1583/1999 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 1999.

[19] Hai-Feng Cui Xin Zheng Tong Ruan. An e_cient texture synthesis algorithm based on wt. 6:3472-3477, 2008.

[20] Darwyn Peachey Ken Perlin David S. Ebert, F. Kenton Musgrave and Steven Worley. Texturing and modeling. In A Procedural Approach. Morgan Kaufmann Publishers, 1998.

[21] Ning Zhou Weiming Dong, Ning Zhou. Optimized tile-based texture synthesis. Graphics Interface, Montreal, Canada, 2007.

ISSN: 2277-3754
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 3, March 2012

[22] W. Guo Y. Meng, W.H. Li and Y.L. Liu. Particle swarm optimization method used in pixel-based texture synthesis.

[23] Aaron Bobick Nipun Kwatra Vivek Kwatra, Irfan Essa. Texture optimization for example-based synthesis.

[24] Pizzanu Kanongchaiyosy Jakrapong Narkdej. An efficient parameters estimation method for automatic patch-based texture synthesis.

[25] Ankit Shah, Parth Bhatt, Prof. Kirit J. Modi. Image Enhancement Techniques by Texture Synthesis.

AUTHOR BIOGRAPHY

Parth Bhatt Information Technology Department, Patel College of Science and Technology (PCST), Indore, Madhya Pradesh, India

Ankit Shah Computer Science & Engineering Department, Kautilya Institute of Technology and Engineering (KITE), Jaipur, Rajasthan, India

Sunil Pathak Associate Professor & Head, Department of Computer Science & Engineering, Kautilya Institute of Technology and Engineering (KITE), Jaipur, Rajasthan, India