Constructing texture maps using enhanced Beltrami method

- **Thai Van Nguyen**
- **Tuan Do-Hong**
- **Dung Trung Vo**

Ho Chi Minh city University of Technology, VNU-HCM

(Manuscript Received on June 16th, 2015, Manuscript Revised January 15th, 2016)

ABSTRACT

Image quality enhancement is a crucial requirement in many applications of digital image and video processing. Removing artifacts which are suffered from image compression will lose simultaneously image texture components. This paper combines Beltrami method and the window derivative to construct the texture map in an attempt to preserve image details during filtering artifacts. Texture map enhancement is also proposed. Simulation results show that the texture map is robust to noise and matches to real texture components of image.

Key words: *Standard deviation (STD), Sobel, Beltrami, texture map, window derivative.*

1. INTRODUCTION

Image compression is an inevitable requirement to reduce storage space of mobile devices and channel bandwidth. But compression also reduces quality of the original images. Removing artifacts and still preserving image texture is thus very important. The texture map plays an essential role in order to control the filter's strength. The edge map guided post filters are proposed to enhance image quality in [3], [4], [6], [7]. In these methods, the variance and standard deviation operators are used to construct the edge map [10], [11], [12]. But these operators are sensitive to noise. The authors in [5] use the Sobel operator to classify edge pixels and non–edge pixels. Filtering the artifacts using this classification may blur the image due to leak of texture information. Obviously, constructing the texture map is a challenging problem since it is very difficult to define texture in mathematical terms. In [1], [2], texture feature based on the Beltrami method is used to locate texture in image segmentation.

This paper constructs an enhanced texture map based on the Beltrami method. The texture feature in pixel by pixel accuracy is sensitive to noise. To be more robust and less sensitive to noise, the patch idea is introduced in [2], [8], [9]. However, the texture map using the patch texture feature isn't smooth. The concept of the window derivative is thus introduced in this paper to overcome this issue. The texture feature based on the window derivative produces the texture map with higher accuracy and higher robustness to noise. In this map, there are still many isolated pixels. In order to increase accuracy of the texture map, this paper proposes a novel method to further enhance the texture map quality by removing isolated pixels.

The paper is organized as follows. Section 2 presents the texture map construction methods based on operators. Section 3 proposes the novel method to construct the texture map based on Beltrami method and another method to further enhance the texture map. Simulation results are presented in Section 4. Final, Section 5 gives the conclusions.

2. TEXTURE MAP BASED ON OPERATORS

Texture map is constructed by classifying pixels. Normally, the map quality depends on the classification feature. The pixels in the texture map are generally classified as strong edges, weak edges, strong texture, weak texture and flat areas. Usually, the texture map is estimated based on operators such as standard deviation and edge detector. The threshold values for pixel classification are selected experimentally. Besides, the texture map quality assessment bases on subjective observation. Furthermore, the texture map is used to remove artifacts in compressed images. So, accuracy of the texture map influences image enhancement, which are shown quality metrics such as PSNR, SSIM and visual quality.

2.1 Texure Map Based on Standard Deviation

In every pixel $I(x,y)$, the standard deviation (STD) at this pixel is calculated with a 3x3 window as follows.

$$STD(x,y)=\sqrt{\frac{1}{9}\left(\sum_{m=-1}^{1}\sum_{n=-1}^{1}\left[I(x+m,y+n)-I_{mean}\right]^2\right)} \quad (1)$$

where

$$I_{mean}=\frac{1}{9}\sum_{m=-1}^{1}\sum_{n=-1}^{1}I(x+m,y+n) \qquad (2)$$

The classification is based on the value of $STD(x,y)$, as shown in (3).

$$\text{Pixel type}=\begin{cases}\textit{Strong edge, if } STD(x,y)>35\\ \textit{Weak edge, if }\\ \quad 30<STD(x,y)\le 35\\ \textit{Strong texture, if }\\ \quad 10<STD(x,y)\le 30\\ \textit{Weak texture, if }\\ \quad 5<STD(x,y)\le 10\\ \textit{Flat ,if } STD(x,y)\le 5\end{cases} \qquad (3)$$

2.2 Texture Map Based on the Sobel Operator

The Sobel operator in [10], [11], [12] consists a pair of 3x3 convolution kernels. The kernels along x and y directions are defined in (4) and (5), respectively:

$G_x =$

-1	0	1
-2	0	2
-1	0	1

(4)

$G_y =$

1	2	1
0	0	0
-1	-2	-1

(5)

The gradient magnitude at each pixel is calculated by:

$$|G|=\sqrt{G_x^2+G_y^2} \qquad (6)$$

The classification is based on the value of $|G|$, as show in (7).

$$\text{Pixel type}=\begin{cases}\textit{Strong edge, if } |G|>0.17\\ \textit{Weak edge, if } 0.95<|G|\le 0.17\\ \textit{Strong texture, if } 0.35<|G|\le 0.95\\ \textit{Weak texture, if } 0.15<|G|\le 0.35\\ \textit{Flat, if } |G|\le 0.15\end{cases} \qquad (7)$$

3. PROPOSED TEXTURE MAP ESTIMATION METHOD

3.1 Texture Feature in Beltrami Method

The authors in [1] represent two-dimentional gray level image to three – dimentional Cartesian space, as show in (8).

$$X:(x,y)\rightarrow (X_1 = x, X_2 = y, X_3 = I(x,y)) \quad (8)$$

The texture feature is defined in [2] as follows:

$$F(x,y) = \exp\left(-\frac{\det(g_{xy})}{\sigma^2}\right) \quad (9)$$

where $\sigma^2 > 0$ is a scaling parameter and g_{xy} is defined as in (10).

$$(10) \quad g_{xy} = \begin{pmatrix} 1+\left(\frac{\partial X}{\partial x}\right)^2 & \left(\frac{\partial X}{\partial x}\right)\cdot\left(\frac{\partial X}{\partial y}\right) \\ \left(\frac{\partial X}{\partial x}\right)\cdot\left(\frac{\partial X}{\partial y}\right) & 1+\left(\frac{\partial X}{\partial y}\right)^2 \end{pmatrix}$$

Pixel classificaton based on $F(x,y)$ is then used to construct a texture map as in (11).

$$\text{Pixel type} = \begin{cases} \textit{Strong edge, if } F(x,y) < 10^{-5} \\ \textit{Weak edge, if } 10^{-5} \le F(x,y) < 10^{-3} \\ \textit{Strong texture, if } \\ \quad 10^{-3} \le F(x,y) < 0.35 \\ \textit{Weak texture, if } \\ \quad 0.35 \le F(x,y) < 0.75 \\ \textit{Flat, if } F(x,y) \ge 0.75 \end{cases} \quad (11)$$

$F(x,y)$ based on pixel by pixel is sensitive to noise, so the texture map cannot obtain high accuracy. To be more robust, the authors in [2], [8], [9] propose estimating $F(x,y)$ based on patches. A $P(x,y)$ square patch of size $(\tau+1)\times(\tau+1)$ around *pixel* (x,y) is defined as in (12).

$$P(x,y) = \{I(x+t_x), I(y+t_y)\} \quad (12)$$

$$t_x \in \left[-\frac{\tau}{2}, \frac{\tau}{2}\right] \qquad t_y \in \left[-\frac{\tau}{2}, \frac{\tau}{2}\right] \quad (13)$$

The value of g_{xy} from [2] is derived as in (14).

$$g_{xy} = \begin{pmatrix} 1+(\partial_x P(x,y))^2 & \partial_x P(x,y).\partial_y P(x,y) \\ \partial_x P(x,y).\partial_y P(x,y) & 1+(\partial_y P(x,y))^2 \end{pmatrix} \quad (14)$$

Pixels are then classified as in (15). However, texture map based on patch is not highly accurate since the error of classification is large.

$$\text{Pixel type} = \begin{cases} \textit{Strong edge, if } \quad F(x,y) < 5.10^{-13} \\ \textit{Weak edge, if } \\ \quad 5.10^{-13} \le F(x,y) < 2.10^{-6} \\ \textit{Strong texture, if } \\ \quad 2.10^{-6} \le F(x,y) < 0.12 \\ \textit{Weak texture, if } \\ \quad 0.12 \le F(x,y) < 0.75 \\ \textit{Flat, if } F(x,y) \ge 0.75 \end{cases} \quad (15)$$

3.2 Texture Map Estimation Based on New Feature

In this paper, $F(x,y)$ based on window derivative is introduced to obtain the texture map with higher accuracy. Let $W(x,y)$ window be the size of $(2R+1)\times(2R+1)$ pixels. The value of g_{xy} is defined

as in (16) where the window derivatives are defined in (17) and (18).

$$g_{xy} = \begin{pmatrix} 1+(\partial_x W(x,y))^2 & \partial_x W(x,y)\partial_y W(x,y) \\ \partial_x W(x,y)\partial_y W(x,y) & 1+(\partial_y W(x,y))^2 \end{pmatrix} \quad (16)$$

$$\partial_x W(x,y) = \sqrt{\sum_{m=1}^{2r+1}\sum_{n=1}^{2r+1}[W(m+1,n)-W(m,n)]^2} \quad (17)$$

$$\partial_y W(x,y) = \sqrt{\sum_{m=1}^{2r+1}\sum_{n=1}^{2r+1}[W(m,n+1)-W(m,n)]^2} \quad (18)$$

Texture map based on window derivative is estimated as in (19).

$$\text{Pixel type} = \begin{cases} \textit{Strong edge, if} & F(x,y) < 5.10^{-13} \\ \textit{Weak edge, if} & 5.10^{-13} \le F(x,y) < 2.10^{-6} \\ \textit{Strong texture, if} & 2.10^{-6} \le F(x,y) < 0.12 \\ \textit{Weak texture, if} & 0.12 \le F(x,y) < 0.6 \\ \textit{Flat ,if} & F(x,y) \ge 0.6 \end{cases} \quad (19)$$

(19)

3.3 Texture Map Enhancement

Removing isolated pixels in the texture map is essential. Since textures are geometric structures and noise is not, this paper proposes the method to enhance the texture map quality as shown in Figure 1 by removing isolated noisy texture pixels.

In Figure 1, the input is the texture map with many isolated pixels. A 3x3 window is slided on the texture map. In each window, the algorithm compares the center pixel to its neighbours. If the pixel type of the center pixel is not the majority type of all pixels in the window then the center pixel is replaced by its majority neighbour pixel. If the isolated pixels are all removed the process is finished otherwise it is repeated.

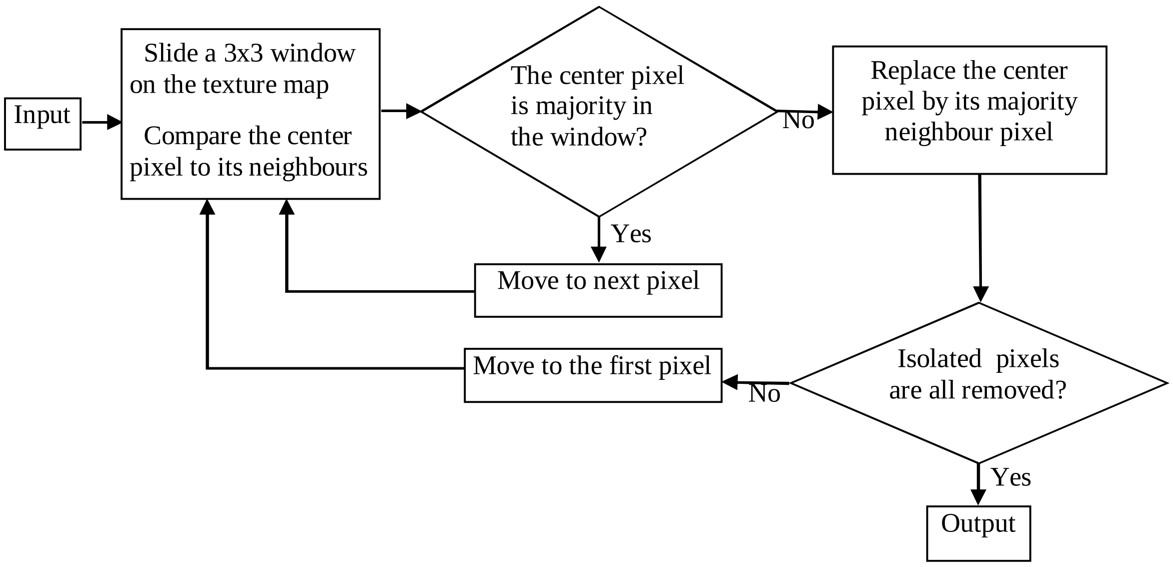

Input
Slide a 3x3 window on the texture map
Compare the center pixel to its neighbours
The center pixel is majority in the window?
No
Replace the center pixel by its majority neighbour pixel
Yes
Move to next pixel
Move to the first pixel
No
Isolated pixels are all removed?
Yes
Output

Figure 1. Flow chart of the texture map enhancement.

4. SIMULATION RESULTS

Many methods on the texture map construction are simulated on a large image data set. Howerver due to space limitation, the only simulation results on the Lena image and the Brick–house image are shown in this paper. The Lena image has a few textures but with various areas. The Brick–house image contains mostly textures with various texture types. The simulation parameters are as follows:

Size of a sliding window: 3x3 pixels
The scaling parameter in $(9): \sigma = 15$
Iteration number: 10

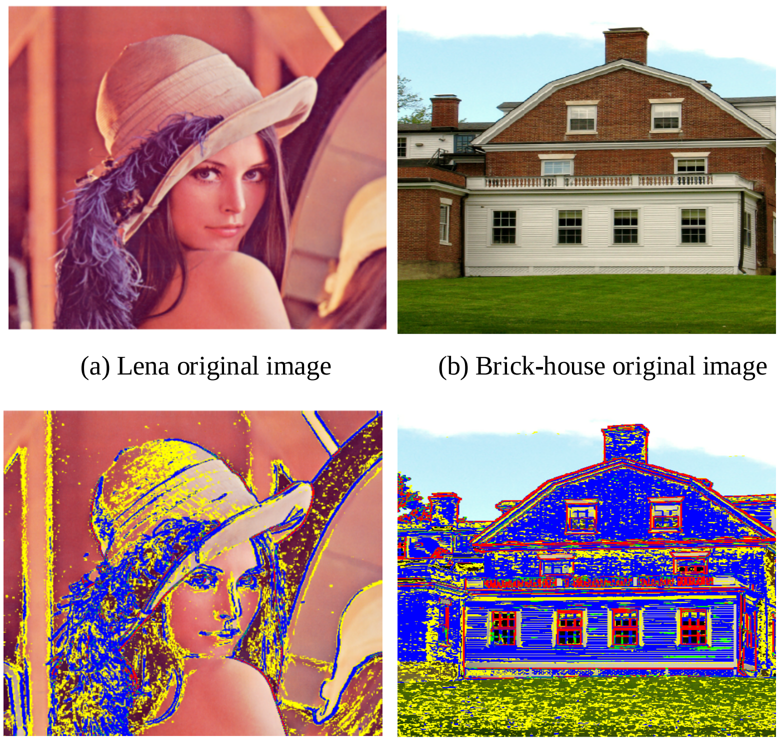
Computer configuration for simulation is as follows:

CPU: Intel(R) Core(TM) i5 2.4GHz
RAM: 4GB
Operating system: Window 7
Simulation software: Matlab 7.10.0 (R2010a)

Simulation results of texture map are shown in Figure 2 to Figure 6. In these figures, the left images are the results for Lena image and the right images are results for Brick–house image.

Colors of the texture map are defined as follows:

Red: strong edge
Green: weak edge
Blue: strong texture
Yellow: weak texture
Others: flat



(a) Lena original image (b) Brick-house original image

(c) Texture map of Figure 2(a) (d) Texture map of Figure 2(b)

Figure 2. Texture map based on STD: (a) and (b) original image, (c) and (d) texture map.

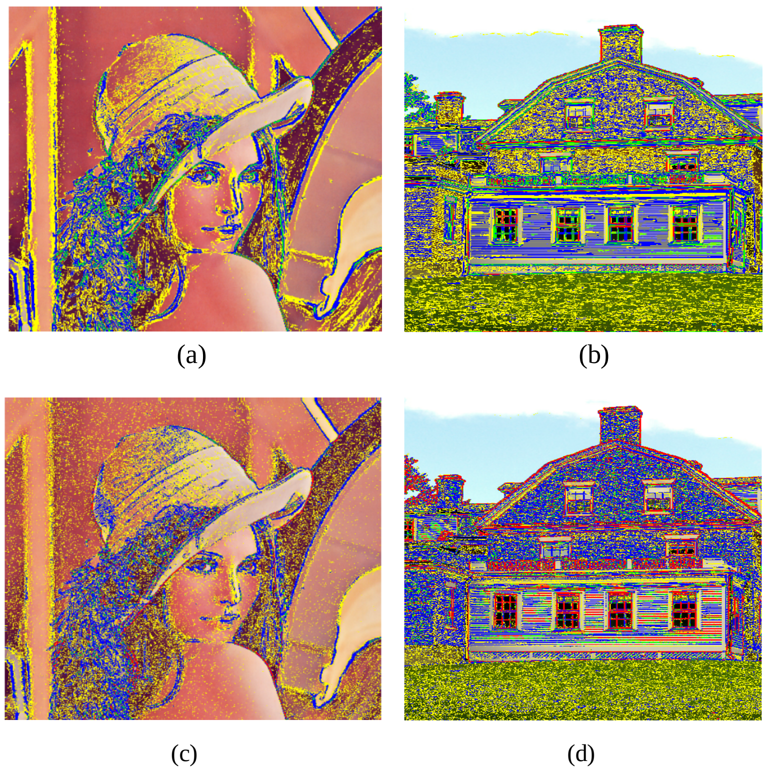


Figure 3. Texture map based on Sobel operator (a and b), and on pixel by pixel Beltrami method (c and d).

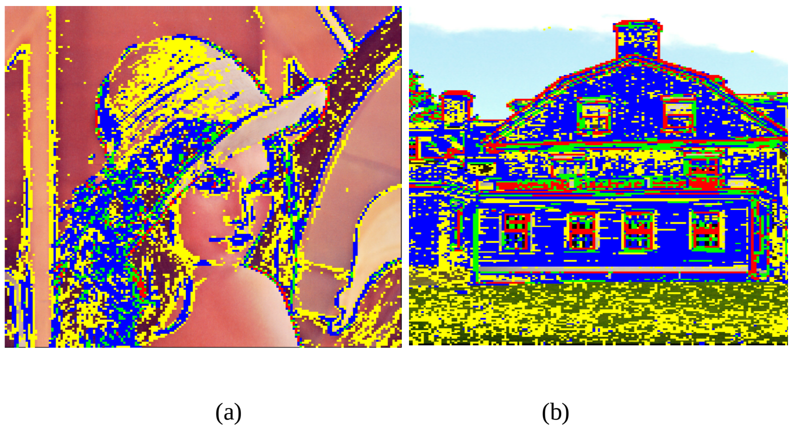


Figure 4. Texture map based on patch Beltrami method.

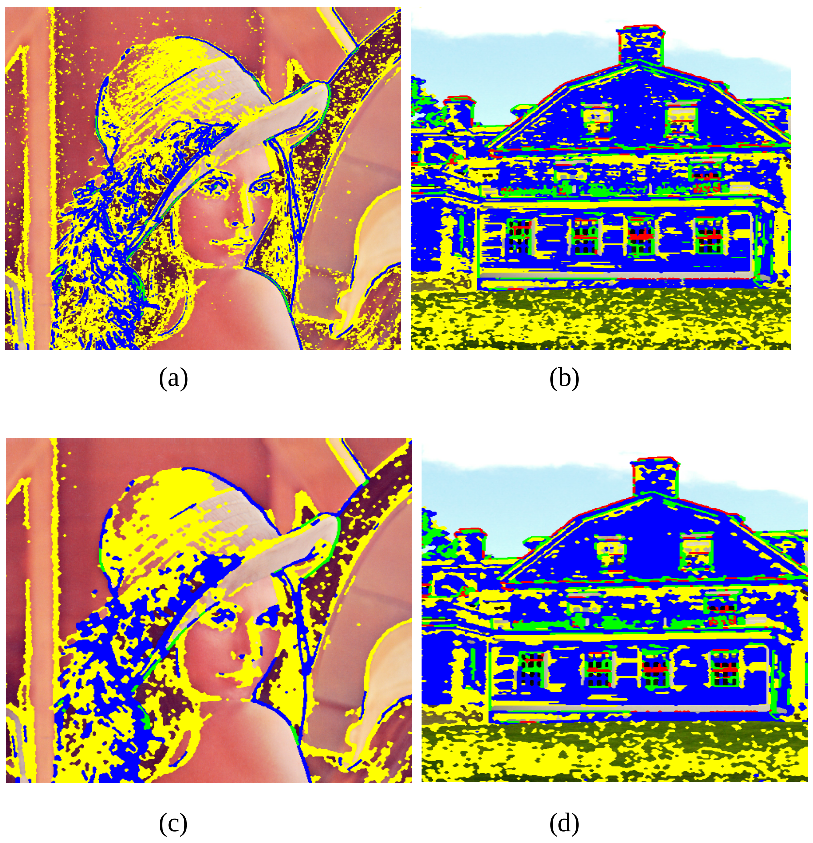


Figure 5. (a) and (b) texture maps based on the window derivative Beltrami method, (c) and (d) texture maps with further enhancement.

Figure 2 and Figure 3 show the texture maps based on STD, Sobel and Beltrami, respectively. These maps detects texture areas of the images. However, there are many isolated pixels in the texture map. The texture map is thus not smooth and is sensitive to noise.

Figure 4 shows the texture maps based on patch. These maps are less sensitive to noise but there are many raw edges because classification error is large. Figure 5 is results of the proposed method. The texture map is more robust and smooth in Figure 6(a) and Figure 6(b). But there are still many isolated pixes in these maps. The further enhanced maps are shown Figure 6(c) and Figure 6(d). The results validate the efficiency of the proposed algorithm. The texture map is smoother and more correspoding to texture areas.

5. CONCLUSIONS

This paper has reviewed the methods of constructing texture maps such as STD method, Sobel operator method and Beltrami method. These methods are sensitive to noise and the texture classification is not highly accurate. The patch Beltrami method produces the texture maps with higher robustness to noise but they are not smooth. This paper proposes a novel texture map estimation based on the window derivative Beltrami method. The texture map constructed by this feature is more accurate than the other methods. However, these maps still have many isolated pixels. Another step is proposed to

further enhance the texture maps. Simulation results on a large image set show that the proposed method introduces the smoother and highly accurate texture maps.

Acknowledgement: *This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number C2015-20-12.*

Thiết lập bản đồ texture dùng phương pháp Beltrami nâng cao

- **Nguyễn Văn Thại**
- **Đỗ Hồng Tuấn**
- **Võ Trung Dũng**

Trường Đại Học Bách Khoa, ĐHQG-HCM

TÓM TẮT

Nâng cao chất lượng ảnh nén là yêu cầu không thể thiếu trong các ứng dụng về xử lý ảnh và video số. Việc lọc bỏ các thành phần suy giảm chất lượng ảnh nén đồng thời sẽ làm mất đi thành phần texture của ảnh.Trong bài báo này, phương pháp Beltrami kết hợp việc tính đạo hàm cửa sổ được sử dụng để thiết lập bản đồ texture với mục đích điều khiển bộ lọc nhằm hạn chế ảnh hưởng đến thành phần chi tiết của ảnh. Một phương pháp nâng cao chất lượng bản đồ texture cũng được đề nghị. Các kết quả mô phỏng cho thấy bản đồ texture bền vững với nhiễu, phù hợp với các thành phần texture thực tế của ảnh.

Từ khóa: *Độ lệch chuẩn, Sobel, Beltrami, bản đồ texture, đạo hàm cửa sổ.*

REFERENCES

[1]. N. Sochen, R. Kimmel, and R. Malladi, *A General Framework for Low Level Vision*, IEEE transactions on image processing, VOL. 7, NO. 3, March 1998

[2]. N. Houhou, J.–P. Thiran and X. Bresson, *Fast Texture Segmentation Based on Semi-Local Region Descriptor and Active Contour*, Global-Science Press, Vol. 2, No. 4, pp. 445-468, November 2009.

[3]. H. Kong, A. Vetro, and H. Sun, *Edge map guided adaptive post-filter for blocking and ringing artifacts removal*, ISCAS, 2004

[4]. H. Kong, Y. Nie, A. Vetro, H. Sun and K. Barner, *Coding artifacts reduction using edge map guided adaptive and fuzzy filtering*, IEEE International Conference on Multimedia and Expo, 2004.

[5]. D. T. Vo, T. Q. Nguyen, S. Yea, A. Vetro , *Adaptive Fuzzy Filtering for Artifact Reduction in Compressed Images and Videos*, IEEE Transactions on Image Processing, Vol. 18, pp. 1057-7149, 2009

[6]. E. Nadernejad, S. Forchhammer, and J.Korhonen, *artifact reduction of compressed images and video combining adaptive fuzzy filtering and directional anisotropic diffusion***,** EUVIP, 2011

[7]. E. Nadernejad, S. Forchhammer, and J.Korhonen, *Adaptive deblocking and deringing of h.264/avc video sequences*, ICASSP, 2013

[8]. A. Efros and Thomas K. Leung, *Texture Synthesis by Non-Parametric Sampling, IEEE* InternationalConference on Computer Vision, 2:10–33,1999

[9]. L. Liang, C. Liu, Y. Q. Xu, B. Guo, and H. Y. Shum, *Real-time texture synthesis by patch-based sampling*, ACM Trans. Graph, 20(3):127–150, 2001

[10].K. Nick, V. Nagesh, L. B Robert, *Design of an image edge detection filter using the Sobel operator*, IEEE journal of solid-state circuits, vol. 23, NO. 2, April 1988

[11].J. Akansha, G. Mukesh, S. N. Tazi, Deepika, *Comparison of Edge Detectors*, International Conference on Medical Imaging, m-Health and Emerging Communication Systems (MedCom), 2014

[12].C. Mala, M. Sridevi, *Parallel algorithms for Edge detection in an Image*, International Conference on Network-Based Information Systems, 2014