GPU-ACCELERATED IMAGE RETEXTURING IN GRADIENT DOMAIN

Ping Li, Hanqiu Sun
Department of Computer Science & Engineering, CUHK, Hong Kong

Jianbing Shen
School of Computer Science & Technology, BIT, Beijing, China

Keywords: Image-based Rendering, Non-local Means Filtering, High Dynamic Range Image.

Abstract: This paper presents the novel GPU-accelated image retexturing approach for both high and low dynamic range images using our newly invented fast NLM filtering. Integrating the fast Maclaurin polynomial kernel filter and the latest GPU-CUDA acceleration, our approach is able to produce real-time high quality retexturing for objects of the interest, while preserving the original shading and similar texture distortion. We apply our revised NLM filtering to the initial depth map to ensure smoothed depth field for retexturing. Our approach using GPU-based fast NLM filtering is designed in parallel, and easy to develop on latest GPUs. Our testing results have shown the efficiency and satisfactory performance using our approach.

1 INTRODUCTION

Image retexturing is an image processing method which takes a single image as input and gives a totally different texture appearance to the objects of interest in the images. The essential process is to replace existing textures in region of interest by new textures, while preserving the original shading and similar texture distortion (Guo et al., 2005). 2D textures are usually applied to represent surface appearances without modelling geometric details. However, manual design of textures is still a tedious task. With the great demand from industry such as movie and game producing, image-based techniques in such application are becoming popular, thus important to retexture objects directly in image space. There are texture editing methods proposed (Fang and Hart, 2006; Tsin et al., 2001), however, few of them can replace the texture with just a single image, and most methods deal with LDR images. Khan et al. (2006) proposed a novel HDR image-based material editing method for making objects' textures transparent and translucent. One limitation of the approach is the slow processing time, due to time-consuming bilateral filtering. It is important to develop efficient approaches for retexturing both HDR and LDR images.

In this paper, we propose the novel image retexturing approach by extracting a region of interest from a single image. Rather than relying on tedious user interactions, we utilize the spectral matting strategy (Levin et al., 2008). Alpha matte automatically identifies the object by minimizing a quadratic energy function, and retexturing results are achieved by transferring new textures to the target object (see Figure 1). In particular, we present revised fast NLM filtering, and develop the approach on GPU-CUDA platform. With simple user interaction, our GPU-accelerated retexturing approach achieved efficient processing, and high quality visual effects similar to the previous work (Fang and Hart, 2004; Khan et al., 2006).

In the remainder of this paper, we first review the related work in Section 2. We present our image retexturing in gradient domain in Section 3, and describe the revised NLM filtering with GPU acceleration in Section 4. Our experimental results are shown in Section 5. Finally, the summary of our work goes to Section 6.

Li P., Sun H. and Shen J. (2010).
GPU-ACCELERATED IMAGE RETEXTURING IN GRADIENT DOMAIN.
In *Proceedings of the International Conference on Imaging Theory and Applications and International Conference on Information Visualization Theory and Applications*, pages 29-34
DOI: 10.5220/0002827400290034

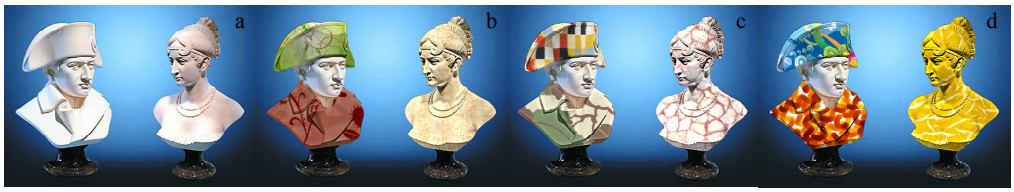

Figure 1: Our image retexturing: (a) the input image, (b-d) image retexturing results generated by us.

2 RELATED WORK

In this section, we mainly review mostly related techniques in image filters, image/video retexturing, and GPU-based processing.

Over years, methods are proposed to remove noise from images. Even algorithms vary, they basically share the same idea making use of nearby information to do averaging. Some filters use spatial information for image blurring such as Gaussian filters (Lindenbaum et al., 1994), anisotropic filters (Perona and Malik, 1990) or the rank vale filters which sort the gray value of the neighbouring pixels (Jäjne, 2005); some carry out the filtering in frequent domain: the Wiener filters (Yaroslavsky, 1985) and Butterworth filters (Gonzalez and Woods, 2008); or by calculus of variations in spatial or frequent domain.

Image retexturing has been fascinating lately. Oh et al. (2001) proposed techniques to change the shape, color and illumination of objects depicted in images. Liu et al. (2004) introduced a user-assisted adjustment on the regular grid of real texture and obtained a bijective mapping between the regular grid of texture and the deformed grid of the surface. However, these methods require elaborate user interactions and only suitable for regular textures. Fang and Hart (2004, 2006) proposed a retexturing technique based on the assumption that lighting satisfies the Lambertian surface, and object's macro structure can be altered. Guo et al. (2008) proposed a novel image/video retexturing preserving the original shading effects. Khan et al. (2006) suggested an image based material editing focusing on changing objects' micro structure. Shen et al. (2007) presented an image-based processing for tone mapping and retexturing of HDR/LDR images. These schemes are usually slow due to time-consuming bilateral filtering. Thus, it's necessary to develop efficient retexturing approach applicable for HDR and LDR images.

Driven by the insatiable market demand for real-time, high-quality 3D graphics, GPU has evolved into a highly parallel, multi-threaded, and many-core processor with great computational horsepower in floating-point calculation. GPU works quite well on floating-point computation while CPU performance ordinarily. Kazhdan and Hoppe (2008) proposed a novel streaming multi-grid GPU solver to solve large linear systems arising from gradient domain image processing. McCann and Pollard (2008) proposed a GPU-based image editing that allows artists to paint in the gradient space with real-time feedback on megapixel images. The reason behind the discrepancy between CPU and GPU is that GPU is specialized for compute-intensive, highly parallel computation and more transistors are devoted to data processing. Here we use GPU-CUDA platform to accelerate our retexturing approach especially the time-consuming filtering process.

3 IMAGE RETEXTURING

Our image retexturing approach is proposed in gradient space using fast NLM filtering for both HDR and LDR images, which has the following main processing steps:

- compute the luminance image in I to get an initial depth map;
- use the revised NLM filter to get smoothed depth map, $I_d^s = FastNLM(I_d)$;
- get the gradient depth map from I_d^s: $\nabla I_g(x,y)$;
- compute the texture coordinate (u_x, u_y) according to the gradient depth map $\nabla I_g(x,y)$;
- derive the new color gradient $\nabla I_g^{new}(x,y)$ from $\nabla I_g(x,y)$ and the input texture gradient ∇T_g;
- reconstruct the final image from $\nabla I_g^{new}(x,y)$ by solving Poisson equation.

3.1 Gradient Depth Recovery

Given the revised NLM filtering, the general retexturing method we adopt is to acquire an image of the object of interest as input, create an alpha

matte to separate the object from the background. By aligning our image manipulation with the object's boundaries, the pixels forming the object are altered to fit the object with new textures.

Computing a depth map from a single image is the classic shape from shading problem. Since it is a severely under-constrained problem (Horn and Brooks, 1989), good solution for arbitrary images do not exist and few general solutions can faithfully compute the depth information. Our method derives a locally consistent depth map from the luminance distribution of an object, where higher luminance values specify the parts of the object closer to the observer. Similar with khan et al. (2006), an initial depth map is computed as:

$$I_d(x,y) = I_{lumi}(x,y)$$

We use our revised NLM filtering to get the smoothed depth map $I_d^s = FastNLM(I_d)$. Then, the filtered depth map I_d^s is used to estimate and find the local gradient depth map. The final recovered gradient depth map $\nabla I_g(x,y)$ is defined in terms of neighbouring depth values, as follows:

$$\nabla I_g(x,y) = (\nabla I_{g_x}(x,y), \nabla I_{g_y}(x,y))$$

The final gradient information is directly used to warp the input textures with the image-based models to achieve image retexturing. Even the depth map we compute may not be precise in the reconstructed map, this don't affect the high-quality retexturing visual effect in our experimental results.

3.2 Retexturing in Gradient Space

Generally, the gradient space $\nabla I_g(x,y)$ is sufficient for estimating the warping of an arbitrary texture, and can be used to retexture the object of interest. We use the lazy snapping (Li et al., 2004) to segment the object in the region of interest. We introduce two linear scale factors v_x, v_y, and use linear combination of gradient value and 2D spatial indices to calculate the texture coordinates (u_x, u_y):

$$\begin{cases} u_x = x + v_x \nabla I_{g_x} \\ u_y = y + v_y \nabla I_{g_x} \end{cases}$$

Considering a pixel (x,y) belonging to the object with a *RGB* color gradient $\nabla I_g(x,y)$, we derive the new color gradient $\nabla I_g^{new}(x,y)$ from its original color gradient $\nabla I_g(x,y)$ and the input texture gradient $\nabla T_g(u_x, u_y)$ using the matting equation as:

$$\nabla I_g^{new}(x,y) = s\nabla I_g(x,y) + (1-s)\nabla T_g(u_x, u_y)$$

Where s is a scalar parameter that linearly interpolates between the original object color gradient and the texture color gradient. With the new retextured gradient $\nabla I_g^{new}(x,y)$, we can get the retextured image by solving a Poisson equation (Perez et al., 2003). Our efficient retexturing approach produces high-quality visual results of image retexturing, also with much less time cost using GPU-CUDA acceleration compared with the previous work.

4 FAST NLM FILTERING

In our approach, we develop the real-time NLM filtering to manipulate gradient space, convert the gradient domain into approximate geometry details for retexturing in image space.

4.1 Revisited Kernel Filter

The weighting functions used in (Choudhury and Tumblin, 2003) are standard Gaussian kernel. Let $x = t/\sigma$, the original Gaussian kernel is:

$$G(x) = e^{-\frac{x^2}{2}}$$

If we take x^2 as input, there are one division, one minus and one exponent operation in the original Gaussian kernel. Exponent operation is very computationally expensive. We need to develop approximation functions (Chatterjee & Milanfar, 2008) that don't involve exponent operation to reduce the cost of operations.

Maclaurin series is a famous continuous function approximation rule, a Taylor expansion about zero. We use Maclaurin series to develop an exponent-free fast Maclaurin kernel as:

$$G^m(x) = \frac{384}{b^2 + 64a}, \quad \begin{cases} a = x^2 + 2 \\ b = a^2 + 12 \end{cases}$$

Taking x^2 as input, the evaluation of $G^m(x)$ in this form consists of three multiplication, one division and three addition operations. Computational power is saved since there are only simple floating-point operations. Maclaurin series is one of the best math-proved approximation rules, thus the new kernel is very close to the original Gaussian kernel, we can surely use our fast Maclaurin kernel filter to appro-

ximate the original Gaussian kernel.

4.2 Simplified NLM Filter

The non-local means filter recently proposed by Buades et al. (2006) is a tremendous improvement for the neighbourhood filters (Smith and Brady, 1997). The NLM filter can be represented as:

$$u(x) = \int w_f(x,y) f(y) dy$$

with the weighting coefficient in the form:

$$w_f(x,y) = \frac{e^{-\frac{d_f^2(x,y)}{h^2}}}{\int e^{-\frac{d_f^2(x,y)}{h^2}} dy}$$

Different from neighbourhood filters, the NLM filter quantifies the similarity of pixels x and y by taking the similarity of whole patches around. The similarity of pixels is evaluated by a distance metric $d_f^2(x,y)$, which contains size r of the compared patches, and a weighting function $g_f(x,y)$ with a parameter h describes how fast the weight decays with increasing dissimilarity of respective patches. Since the similarity measure uses the information of the whole nearby patches instead of single pixel intensity, the NLM filtering is able to remove noise from textured images without destroying the fine structures of the texture itself (see Figure 2).

Using the revisited Maclaurin kernel, we further perform algorithm simplification. Mainly we apply the NLM filter to smooth the gradient domain, the gradient NLM filter at pixel x is defined as:

$$\nabla u(x) = \frac{1}{c(x)} \int G_{\sigma_r}^m(\|y-x\|) G_{\sigma_h}^m(\|d_{\nabla c}(x,y)\|) \nabla f(y) dy$$

$$\begin{cases} c(x) = \int G_{\sigma_r}^m(\|y-x\|) G_{\sigma_h}^m(\|d_{\nabla c}(x,y)\|) dy \\ d_{\nabla c}^2(x,y) = \frac{1}{s(b)} \int_{b(x)} |\nabla f(y-(x-\alpha)) - \nabla f(\alpha)| d\alpha \end{cases}$$

Where $\nabla f(y)$ is the noisy gradient image, $\nabla u(x)$ is result produced by our NLM filter with the parameters σ_h and σ_r, and $G_{\sigma_r}^m(\cdot)$ and $G_{\sigma_h}^m(\cdot)$ are the kernel filters in the form of eighth order Maclaurin function. $c(x)$ is the normalizing coefficient, and $s(b)$ is the area of b. Here, $d_{\nabla c}^2$ represents the normalized sum of absolute gradient differences between blocks around pixel x and y.

4.3 GPU-accelerated Retexturing

Recently the traditional neighbourhood filters can run in real time even without hardware acceleration. But, it is not the case with the NLM filter. We notice that the latest OpenGL or DirectX hardware allows high quality filters for even high resolution images. With the help of latest NVIDIA GPU, we can benefit from using a general purpose programming model CUDA. Features such as shared memory and sync points combined together with flexible thread control allow us to speed up algorithms in parallel.

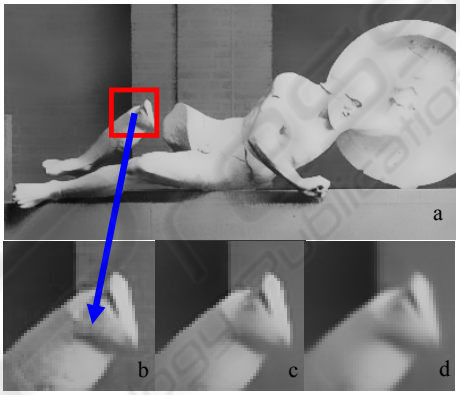

Figure 2: Filtering quality: (a) the initial gradient depth, (b) enlarged detail, (c) KNN filtering result, (d) smoothed result using our revised NLM filtering.

For GPU-CUDA coding, we define $b(x)$ be the spatial neighbourhood of a certain size surrounding pixel x. We consider it as a block of pixels in the size $l \times l$, where $l = 2r+1$, thus x is the center of $b(x)$, the block radius is r. For every pixel n^2 number of weights need to be counted, where $n = 2m+1$; To calculate each weight k^2 number of weights need to be counted, where $k = 2i+1$, texture fetches are performed here to compute the $d_{\nabla c}^2(x,y)$ function. The total n^2k^2 number of texture fetches is needed, so reducing the number of texture fetches highly increase the performance.

To achieve the reduction of texture fetches, we assume that within each block weights do not change. We compute weights for the pixels in center, and use these weights as the convolution coefficients for other pixels within the same block. In this way, we are able to reduce the number of texture fetches to k^2. For the most common value $n = 5$ for the blocks, we can reduce 25 times less texture fetches. The assumption that weights are uniform within the block works well, and most of the revised NLM filter smoothed areas have little visual difference

Figure 3: Top line - LDR image retexturing of the decorated cup using our approach; lower line - HDR image retexturing of the artwork using our approach, in which the left-most is the original image.

from the NLM filter. With the fast Maclaurin kernel filter we develop, the revised NLM filter works much faster than the traditional NLM processing.

5 RETEXTURING RESULTS

We used C with NVIDIA CUDA programming and MATLAB to implement our GPU-accelerated image retexturing. The samples shown in the paper have been tested on an Intel(R) Core(TM)2 Duo CPU 2.3GHz PC with a NVIDIA GeForce 8800 GPU and 2GB RAM. We have compared the revised NLM filtering with bilateral filtering, K-nearest neighbors filtering and the NLM filtering, in which the filters tested are implemented in parallel with GPU acceleration. The resolutions of sample images we tested are in the range from 50×50 to 1600×1600. From the tests, our approach using GPU acceleration can perform more than twice as fast as the NLM filtering. The computational performance statistics for GPU-based filtering is listed in Table 1. Even the GPU KNN filter is somehow faster than our revised NLM filter, it is shown in Figure 2 that the filtering quality of KNN is not satisfied as ours.

Table 1: Timing performance (frame/second) for GPU filters of KNN, bilateral, NLM and our fast NLM filter in relation to image sizes.

GPU Filters	50×50	200×200	800×800	1600×1600
KNN	1372.3	986.3	178.7	49.7
Bilateral	783.6	108.5	32.6	12.6
NLM	651.9	91.6	6.5	1.6
Our fast NLM	1353.8	886.9	132.2	35.7

Figure 4 shows our texture distortion effect of sculpture retexturing in comparison with previous work, where the retextured distortion effect of our result comparable to the synthesis method of Fang and Hart (2004). Both generate good distortion effect conforming to the image underlying geometry. In Guo et al. (2008), user needs to perform considerable mesh stretches to deform new texture conforming to the image geometry. Since we perform image retexturing in gradient space, our approach is able to replace the existing textures in the region of interest, while preserving the original shading and similar texture distortion using the gradient difference info and simple user interaction (i.e. tuning one/two parameters).

More retexturing results are shown in Figure 3 and Figure 5, which showed the impressive visual effects with real-time performance and simple user interaction, including HDR image retexturing (artwork) and LDR image retexturing (decorated cup, fruits, cloth, sculpture) examples. Our approach can be applied to both HDR and LDR images, uniformly. When using HDR display (Hoefflinger, 2007), the visual effects look much better as the high dynamic range to manipulate for the light intensity.

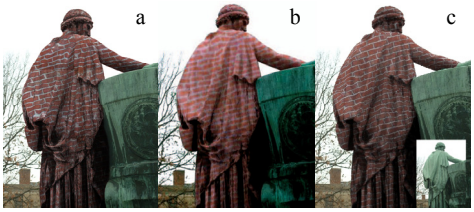

Figure 4: Texture distortion effect: The input image is the inset in (c), (a) is our result, (b) is the result of Guo et al. (2008) and (c) is the result of Fang and Hart (2004).

6 SUMMARY

In this paper, we present the novel GPU-accelerated image retexturing using our revised NLM filtering for both HDR and LDR images. Integrating the fast Maclaurin kernel filter and parallel GPU-CUDA acceleration, our approach is able to produce real-time realistic results of image retexturing with simple user interactions (i.e. tuning one/two parame-

Figure 5: Top line - LDR fruits retexturing using our approach; lower line - LDR cloth retexturing (left) and LDR sculpture retexturing (right) using our approach.

ters). Using the smoothed depth map in gradient space, the reconstructed map provides the retexturing visual qualities. Our experimental results have shown the feasibility and the efficiency of our approach. We will further work on utilizing geometrical properties for retexturing, and extending the image retexturing to video applications with better optimizations.

REFERENCES

Buades, A. and Coll, B. and Morel, J. (2006). A review of image denoising algorithms, with a new one, *Multiscale Modeling and Simulation,* 4(2), 490–530.

Chatterjee, P. and Milanfar, P. (2008). A generalization of non-local means via kernel regression. *Proc. of IS&T Conf. on Computational Imaging VI,* San Jose.

Choudhury, P. and Tumblin, J. (2003). The trilateral filter for high contrast images and meshes. *Eurographics Symposium on Rendering '03*, 186–196.

Fang, H. and Hart, J. (2006). Rototexture: automated tools for texturing raw video. *IEEE Trans. on Visualization and Computer Graphics*, 12(6), 1580–1589.

Fang, H. and Hart, J. (2004). Textureshop: texture synthesis as a photograph editing tool. *International Conference on Computer Graphics and Interactive Techniques*, ACM New York, 354–359.

Gonzalez, R. and Woods, R. (2008). *Digital image processing* (3rd ed.). NJ: Pearson/Prentice-Hall.

Guo, Y. and Wang, J. and Zeng, X. and Xie, Z. and Sun, H. and Peng, Q. (2005). Image and video retexturing. *Computer Animation and Virtual Worlds,* 16, 451-461.

Guo, Y. and Sun, H. and Peng, Q. and Jiang, Z. (2008). Mesh-Guided Optimized Retexturing for Image and Video. *IEEE Transactions on Visualization and Computer Graphics,* 14(2), 426–439.

Hoefflinger, B. (2007). *High-dynamic-range (HDR) vision.* Berlin: Springer.

Horn, B. and Brooks, M. (1989). *Shape from shading.* Mass: MIT press Cambridge.

Jähne, B. (2005). *Digital image processing, concepts algorithms, & scientific app.*. Berlin: Springer-Verlag.

Kazhdan, M. and Hoppe, H. (2008). Streaming Multigrid for Gradient-Domain Operations on Large Images. *Proceedings of ACM SIGGRAPH 2008,* 27(3).

Khan, E. and Reinhard, E. and Fleming, R. and Bulthoff, H. (2006). Image-based material editing. *Proceedings of ACM SIGGRAPH 2006,* 25(3), 654–663.

Levin, A. and Rav-Acha, A. and Lischinski, D. (2008). Spectral matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 30(10), 1699–1712.

Li, Y. and Sun, J. and Tang, C. and Shum, H. (2004). Lazy snapping. *ACM Trans. Graph,* 23(3), 303–308.

Lindenbaum, M. and Fischer, M. and Bruckstein, A. (1994). On Gabor's contribution to image enhancement. *Pattern Recognition,* 27(1), 1-8.

Liu, Y. and Lin, W. and Hays, J. (2004). Near-regular texture analysis and manipulation. *Proceedings of ACM SIGGRAPH 2004*, ACM New York, 368–376.

McCann, J. and Pollard, N. (2008). Real-Time Gradient-Domain Painting. *Proceedings of ACM SIGGRAPH 2008*, 27(3).

Oh, B. and Chen, M. and Dorsey, J. and Durand, F. (2001). Image-based modeling and photo editing. *Proceedings of ACM SIGGRAPH 2001*, ACM New York, 433–442.

Perez, P. and Gangnet, M. and Blake, A. (2003). Poisson image editing, *ACM Trans. Graphics,* 22(3), 313–318.

Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 12(7), 629–639.

Shen, J. and Jin X. and Sun H. (2007). High dynamic range image tonemapping and retexturing using fast trilateral filtering. *The Visual Computer*, 23(9), 641-650.

Smith, S. and Brady, J. (1997). SUSAN - A new approach to low level image processing. *International Journal of Computer Vision,* 23(1), 45–78.

Tsin, Y. and Liu, Y. and Ramesh, V. (2001). Texture replacement in real images. *Proc. IEEE CVPR 2001,* 2, IEEE Computer Society.

Yaroslavsky, L. (1985). *Digital picture processing*. Berlin: Springer-Verlag and New York: Springer-Verlag.