Parallel implementation of image sharpening method using grid warping*

A.D. Gusev, A.V. Nasonov, A.S. Krylov
kryl@cs.msu.ru
Laboratory of Mathematical Method of Image Processing
Faculty of Computational Mathematics and Cybernetics
Lomonosov Moscow State University
Moscow, Russia

The development of image sharpening algorithms is a challenging problem. The paper presents a parallel implementation of grid warping algorithm for the problem of image sharpening. The algorithm shifts pixels toward edge centerlines to make the edges sharper instead of changing pixel values directly. This approach does not amplify noise level and does not add ringing effect.

Keywords: *Image sharpening, Grid warping, CUDA.*

Introduction

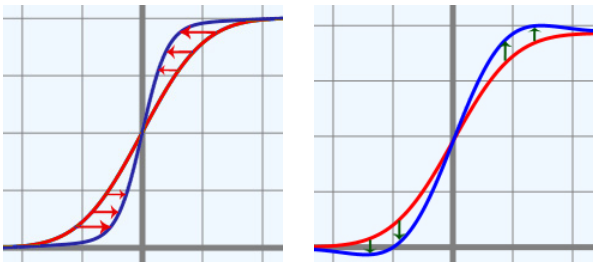
Real images rarely have a high quality. Image blur is typical for real images. It is mainly caused by defocus, camera movement or sensor blur. Image sharpening is a challenging problem. Image deblurring algorithms have to make the image edges and fine textures sharper. At the same time they should not amplify noise, introduce ringing effect or produce other unwanted artifacts.

Typical image sharpening algorithms try to improve the high-frequency information of the image. The simplest deblurring algorithm is unsharp mask method that simply amplifies high-frequency information (see Fig. 1):

$$z_\alpha = \alpha z + (1-\alpha)(z * G_\sigma),$$

where α is the amplification factor, σ is the scale parameter that defines the range of frequencies to be amplified.

Regularization-based methods [1, 2, 3] use a parameter to set a compromise between smooth result with blurry edges and sharp result with artifacts.



(a) (b)

Fig. 1. The idea of edge sharpening by grid warping: (a) Grid warping: pixels are shifted; (b) Typical deblurring approach: pixel values are modified.

The work was supported by Russian Science Foundation grant 14-11-00308.

Grid warping algorithms use another approach to make the image sharper: instead of changing pixel values they transform the pixel grid so that the pixels near the edge move towards the edge centerline [4]. It makes the edge sharper, but does not add noise or ringing effect.

The warping approach for image enhancement was introduced in [5]. The warping of the grid is performed according to the solution of a differential equation that is derived from the warping process constraints. The solution of the equation is used to move the edge neighborhood closer to the edge, and the areas between edges are stretched. The method has several parameters, and the choice of optimal values for the best result is not easy. Due to the global nature of the method the resulting shapes of the edges are sometimes distorted.

In [5] the warping map is computed directly using the values of left and right derivatives. In both methods [5] and [6] the pixel shifts are proportional to the gradient values. It results in oversharpening of already sharp and high contrast edges and insufficient sharpening of blurry and low contrast edges. Both methods also introduce small local changes in the direction of edges and produce aliasing effect due to calculation of horizontal and vertical warping components separately.

The work [4] overcomes the drawbacks of the methods [5] and [6]. It constructs the pixel density map that defines the target pixel density after grid warping. The warping vectors are found from a solution of Poisson equation. The work [7] extends the grid warping algorithm for 3D image sharpening and proposes a direct method for finding warping vectors from the density map.

In this work we propose a parallel implementation of image sharpening using grid warping [4] for GPU.

One dimensional edge sharpening

We describe the pixel shift vectors for one-dimensional edge profile centered at $x = 0$ by the proximity func-

tion $p(x) : p(x) = 1+d'(x)$, where $d(x)$ is the displacement function $d(x)$: $x \to x + d(x)$. The displacement function can be calculated from proximity function using the equation

$$d(x) = \int_{-\infty}^{x} (p(y) - 1)dy \qquad (1)$$

The proximity is the distance between adjacent pixels after image warping. If the proximity function $p(x)$ is less than 1, then the area is densified at the point x (see Fig. 2). If the proximity is greater than 1, then the grid is rarefied. For a non-warped image $p(x) \equiv 1$.

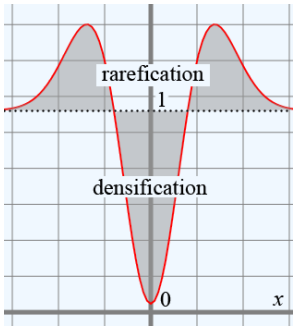


Fig. 2. Example of proximity function for edge sharpening

The proximity function greatly influences the result of the edge warping. On the one hand, the edge slope should become steeper. On the other hand, the area near the edge should not be stretched over some predefined limit to avoid wide gaps between adjacent pixels in the discrete case. The necessary conditions for density and proximity functions are stated in [4].

The work [4] shows the effectiveness of the proximity function constructed as the difference of Gaussian functions

$$p(x) = 1 + \alpha \frac{G_\sigma(x) - G_{k\sigma}(x)}{G_\sigma(0) - G_{k\sigma}(0)}, \qquad (2)$$

where $G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right)$, α is the strength of warping effect, k is the parameter that controls the area of rarefication. We use $\alpha = 1$, $k = 2$. Parameter σ depends on the blur level of superresolution results and is chosen according to superresolution method and scale factor.

Two-dimensional grid warping

In the two-dimensional case the displacement is a vector field $\bar{d}(x, y)$ which is connected to the proximity function by the equation

$$p(x, y) = 1 + \operatorname{div} \bar{d}(x, y).$$

For known $p(x, y)$ the displacement function is obtained from the solution of the equation

$$\bar{d}(x, y) = \nabla u(x, y),$$

$$\begin{cases} \Delta u & = p(x, y) - 1, \\ u(x, y) & = 0 \text{ at image borders.} \end{cases}$$

In order to get the same results as in the 1D case and to keep the edge pixels unwarped, the proximity value should be equal to the 1D proximity function depending on the distance to the edge. However, the distance to the closest edge as an argument of the proximity function is not efficient as it may produce gaps between close edges. Also it blurs edge ends.

We suggest the following method for calculating the proximity function in the two-dimensional case using one-dimensional proximity function $p(x)$:

$$p(x, y) = \frac{\sum_{(x_e,y_e)\in N(x,y)} w(x_e, y_e)p(x_n)}{\sum_{(x_e,y_e)\in N(x,y)} w(x_e, y_e)},$$
$$w(x_e, y_e) = G_\sigma(x_t)|\bar{g}(x_e, y_e)|,$$

where $N(x, y)$ is the set of edge points in the neighborhood of (x, y). The values x_n and x_t are projections of the vector $(x - x_e, y - y_e)$ on the edge gradient vector $\bar{g}(x_e, y_e)$ and on its perpendicular. Edges are detected using Canny edge detection algorithm with zero thresholds [8].

The solution of the equation can be found directly:

$$\bar{d}(x, y) = \frac{\sum_{(x_e,y_e)\in N(x,y)} G_\sigma(x_t)d(x_n)\bar{g}(x_e, y_e)}{\sum_{(x_e,y_e)\in N(x,y)} G_\sigma(x_t)|\bar{g}(x_e, y_e)|} \qquad (3)$$

The calculation of the grid warping vectors (3) is computationally slow. In time critical applications it can be approximated by taking only the nearest edge point in (3).

Finally we perform interpolation on the non-regular pixel grid. The work [4] proposes taking all neighboring pixels (x_k, y_k) of the pixel (x, y) and performing weighted averaging

$$I_R(x, y) = \frac{\sum_k I(x_k, y_k)w(x, y, x_k, y_k)}{\sum_k w(x, y, x_k, y_k)} \qquad (4)$$

where

$$w(x, y, x_k, y_k) = \exp\left(-\frac{(x - x_k)^2 + (y - y_k)^2}{2\sigma_0^2}\right),$$
$$\sigma_0 = 0.3$$

Parallel implementation

Modern GPUs can provide significantly better performance than the CPU but have some architectural limitations that require algorithm adaptation for GPU.

Parallel implementation of the proposed grid warping consists of the following steps:

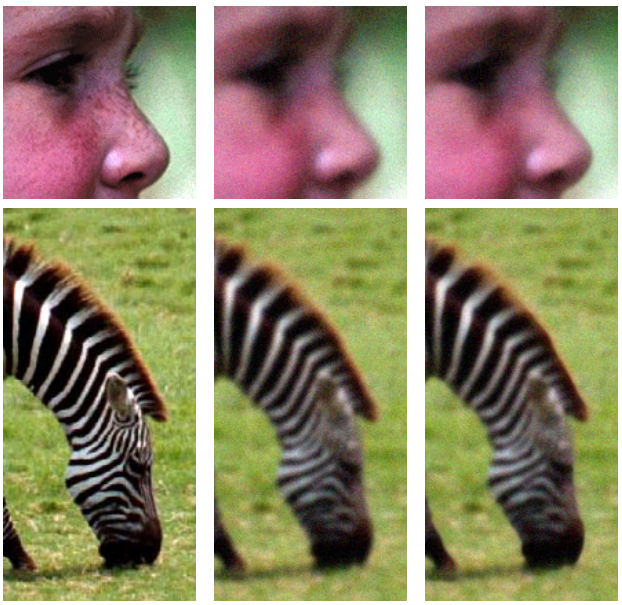
1. Calculation of horizontal and vertical derivatives to find image gradient by convolving the image with 5x5 kernels. Each pixel is calculated independently so this step can be easily parallelized.

2. Perform non-maximum suppression to find the edges. We do not apply hysteresis from Canny edge detector [8] and use only single threshold.

3. Apply the formula (2) pixelwise to find the warping vectors.
4. The interpolation step (4) should be adapted for GPU processing. The problem is to find the neighboring pixels: each pixel is assigned the new coordinates but we need to process the warped pixels. The length of the obtained displacement vectors is limited from (1) and (2) to about $\sim 2\sigma$ so we need to check all the pixels in the neighborhood of the radius 2σ.
In the fast version of the proposed algorithm the step 3 is replaced to the result of Euclidean distance transform [9, 10].

Results

The results of the proposed algorithm are shown in Fig. 3. The reference images are blurred by Gaussian filter with $\sigma = 3$, Gaussian noise with $\sigma = 10$ is added, then image sharpening is applied. It can be seen that the contours become sharper while no noise amplification of ringing effect is observed.
Fig. 4 demonstrates the application of the proposed method to the real medical immunofluorescence image. Grid warping algorithm makes lines thinner.
It takes about 1.2 seconds to process 512×512 using CPU implementation and about 0.3 seconds using GPU implementation with CUDA technology on a computer with Core i7 processor 4 GHz with NVidia GTX 570 video card. The execution time is proportional to the number of pixels.



Reference image Blurred and noisy image Warping result

Fig. 3. Results of the proposed algorithm for synthesized blur and noise

Conclusion

A parallel implementation of grid warping algorithm was developed. It was applied to real and synthetic images to demonstrate the results. The algorithm was implemented for CUDA architecture and showed about 3–5x increase in performance.

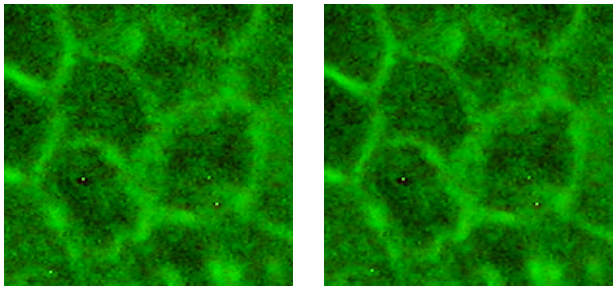


Fig. 4. Results of the proposed algorithm for the real image: the cell boundaries become thinner and sharper

References

[1] *M. Almeida and M. Figueiredo* Parameter estimation for blind and non-blind deblurring using residual whiteness measures // IEEE Transactions on Image Processing, vol. 22, pp. 2751–2763, 2013.

[2] *J. Oliveira, J. M. Bioucas-Dia, and M. Figueiredo* Adaptive total variation image deblurring: A majorization-minimization approach // Signal Processing, vol. 89, pp. 1683–1693, 2009.

[3] *S. D. Babacan, R. Molina, and A. K. Katsaggelos* Variational bayesian blind deconvolution using a total variation prior // IEEE Transactions on Image Processing, vol. 18, pp. 12–26, 2009.

[4] *A. Nasonova, A. Krylov* Deblurred images post-processing by Poisson warping // IEEE Signal Processing Letters, Vol. 22, No. 4, 2015, pp. 417–420.

[5] *N. Arad and C. Gotsman* Enhancement by image-dependent warping // IEEE Transactions on Image Processing, vol. 8, pp. 1063–1074, 1999.

[6] *J. Prades-Nebot et al.* Image enhancement using warping technique // Electronic Letters, vol. 39, pp. 32–33, 2003.

[7] *A. S. Krylov, A. V. Nasonov* 3D image sharpening by grid warping // Lecture Notes in Computer Science (IScIDE2015), Vol. 9242, 2015, pp. 441–450.

[8] *J. Canny* A Computational Approach To Edge Detectio" // IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp. 679–698, 1986.

[9] *Fabbri, Ricardo, et al* 2D Euclidean distance transform algorithms: A comparative survey // ACM Computing Surveys (CSUR), Vol. 40, No. 1, 2008, #2.

[10] *T. T. Cao, K. Tang, A. Mohamed, T. S. Tan* Parallel banding algorithm to compute exact distance transform with the GPU // Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games. 2010, 83–90.