IMPROVED SUPER-RESOLUTION METHOD AND ITS ACCELERATION

Libor Váša, Ivo Hanák, Václav Skala

Department of Computer Science and Engineering, University of West Bohemia
Univerzitní 22, 306 14, Pilsen, Czech Republic
phone: + (420) 377 632 401, fax: + (420) 377 632 402, email: {lvasa|hanak|skala}@kiv.zcu.cz
web: http://herakles.zcu.cz

ABSTRACT

A brief introduction to spatial-domain Super-Resolution methods, i.e. spatial resolution enhancement methods that create one high-resolution image from a series of low-resolution images shifted by a sub-pixel distance, is given. An improvement applicable to some of existing Super-Resolution methods is presented. Principles of digital photography processing techniques are exploited in order to reduce error in the Super-Resolution process. Enhanced registration method applicable to full color images is proposed and results of its hardware implementation are presented.

1. INTRODUCTION

Digital still cameras (DSC) have reached strong position on the field of photographic industry over the past decade. Although parameters of digital equipment constantly improve, there are still areas where analog equipment offers better performance. One of such areas is resolution of gained images.

Using better hardware may provide partial solution to the problem, but there are also several efforts to achieve such goal by software processing of multiple input images shifted by a sub-pixel distance. Such processing is addressed as Super-Resolution (SR).

We will show that SR techniques are capable of improving resolution of images taken by commercially available DSCs. We will show that applying the techniques directly may impair quality of the result due to preprocessing that takes place within the DSCs. We will show how to exploit knowledge about used sensor to improve the results.

We will also show that it is possible to utilize programmable hardware, i.e. GPU, for improving the performance, because SR methods perform image processing operations. As it is common in this area, operations performed on per image element basis are usually simple. However, the fact that these operations are performed on each of the image element means that the whole step utilizing such operation is time extensive.

On the other hand, the GPU is capable to perform simple operation on large amount of data in pipeline manner to perform rendering. It is clearly visible that image processing and GPU computation has a thing in common: they perform rather simple per element operations in large numbers. Therefore it is possible to utilize the GPU for selected parts of our approach that are time extensive in order to improve overall performance of our algorithm.

Most research in this area only considered grayscale images and stated that super-resolving a RGB image is simply a matter of applying some method to each color channel. We will show that our improved method requires a special registration method when color images are reconstructed.

The rest of the paper is organized as follows: necessary introduction to image preprocessing in DSCs is given in section 2, quick introduction to existing spatial-domain SR is given in section 3, our improvement is proposed in section 4 along with its hardware acceleration in section 5. Accuracy and performance experiments are described in section 6, sections 7 gives conclusions and section 8 ideas for future work.

2. IMAGE PREPROCESSING

There is a sophisticated image processing applied to data measured by commonly used image acquisition devices (CCD or CMOS elements) when a computer format image is being created. Most DSCs are using a standard Bayer array of color filters which implies a need for interpolation of measured data to gain a full color image. This step is addressed as demosaicking.

Demosaicking is basically an interpolation process applied to color values measured by Bayer array (see Figure 1) in order to get a RGB triplet in each pixel location. There are many methods to achieve this goal presented in the literature ([2,6,7,10,11,12,14]), because intuitive approaches like linear interpolation produce visible artifacts. Additional preprocessing usually includes white balancing and scaling of values to fit the usual exponential scale used computer image formats.

R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G
G	B	G	B	G	B
R	G	R	G	R	G

Figure 1: Bayer array layout

3. SPATIAL DOMAIN SUPERRESOLUTION

First attempts in the field of super-resolution were performed in frequency domain ([9]) assuming that values

in input images are point samples of the original scene. This assumption fails in case of current DSCs, because the sampling process character is integral. This feature is addressed by spatial domain methods, where the relation between original image and the sampled images is expressed by Equation 1 ([3]).

$$Y_k = \mathbf{D_k}\mathbf{C_k}\mathbf{F_k}X + E_k \quad (1)$$

Where X is a column vector of lexicographically ordered hi-res image
Y_k is a column vector of lexicographically ordered k-th input image
E_k is a column vector of lexicographically ordered additional noise
$\mathbf{F_k}$ is a matrix that represents spatial warp between images (shift)
$\mathbf{C_k}$ is a matrix that represents degradation by camera optics (blur)
$\mathbf{D_k}$ is a matrix that represents integral sampling of the image

Simplest method for spatial domain SR is denoted as Iterative Back-Projection (IBP)[13] and can be described as a search for an image that well predicts given inputs. This is expressed by Equation 2.

$$X^* = \underset{X}{ArgMin}\left(\sum_{k=1}^{N}\left\|\mathbf{D_k}\mathbf{C_k}\mathbf{F_k}X - Y_k\right\|\right) \quad (2)$$

Steepest descent solution of such equation yields ([3,1]) iterative algorithm described by Equation 3, where the β parameter represents a step size of iterative improvement of some initial approximation of the high resolution image.

$$X_{n+1} = X_n - \beta\left[\frac{1}{N}\sum_{k=1}^{N}\left(\mathbf{F_k^T}\mathbf{C_k^T}\mathbf{D_k^T}\left(\mathbf{D_k}\mathbf{C_k}\mathbf{F_k}X_n - Y_k\right)\right)\right] \quad (3)$$

This algorithm is very susceptible to any kind of noise present in input data, which has inspired research on improving robustness of the method. We have implemented two such improved methods, one using pixelwise median instead of averaging the backprojected differences ([15]), and the other using average of the sign function of the differences ([5]).

4. IMPROVED ALGORITHM

Although robustness improving methods do provide good enhancement of the algorithm, it is still very sensitive to any kind of errors in input data.

We have considered a scenario when the user of a DSC wants to improve resolution of the images beyond the limits of the hardware, i.e. the input consists of images gained from the camera with minimal or no preprocessing and we were looking for a way to remove errors from these images.

Demosaicking as necessary part of preprocessing may be source of such error even though today's DSCs utilize advanced techniques. Our idea is to find an algorithm that will not use interpolated data.

Our approach can be viewed as improvement of the basic IBP algorithm by including a binary mask that describes which pixels were measured and which were computed by demosaicking. This is described by Equation 4.

$$X_{n+1} = X_n - \beta\left[Q \otimes \sum_{k=1}^{N}\left(\mathbf{F_k^T}\mathbf{C_k^T}\mathbf{D_k^T}\left(M \otimes \left(\mathbf{D_k}\mathbf{C_k}\mathbf{F_k}X_n - Y_k\right)\right)\right)\right] \quad (4)$$

Where Q is a vector containing number of measurements for each pixel location
M is a vector determining whether a value at corresponding position was measured (value 1) or computed (value 0)
$\otimes$ represents per-element multiplication of vectors

Robustness-enhanced methods may be altered in the same manner, in the case of Zomet method there will appear a median of lower number of values, case of Farsiu algorithm is equal to IBP (average over lower number of pixels).

Although this basically means reducing the input data to one third of its original amount, our testing shows that the results are better than from the non-reduced set due to better accuracy of the reduced input.

The other question that arises is how to perform registration of such images. For testing purposes we have implemented a simplified registration algorithm used in [8,4] that tries to estimate shifts in the X and Y axes by comparing images shifted by various amounts in given interval and choosing the pair of registration parameters (x and y shift) that produces the least difference.

We have tried to exclude interpolated data even from registration. Out first approach was to use the same binary mask as for the SR itself, but this has lead to problems with images of slightly different exposition. For example when registration of a green channel was performed, the slightly lighter images were registered to white squares of a chessboard pattern wrapped over the image, while darker images were registered to positions of black chessboard squares, or other way round. Subsequently, these images only influenced these positions, making lighter pixels even lighter and darker pixels even darker, which in next registration refinement step has lead again to the same effect, only even more visible. In the end a visible chessboard pattern has appeared in the resulting images, and the registration reflected exposition of images more than their exact position. Situation with the red and blue was similar.

Therefore we have decided to implement a simultaneous registration of all RGB channels, where at each position a measured value (R, G or B) is compared to corresponding channel of current hi-res image approximation, and the difference is added to cumulative error measurement. Subsequently, one SR step is performed with the best registration parameters for each color channel separately.

5. HARDWARE REGISTRATION

In order to speedup the whole computation we have utilized GPU for the step of image registration. This step consists basically of comparison of two images in order to gain some statistical information about them. It is clearly visible that this operation of basically a per image element operation. Therefore this step is ideal candidate for GPU implementation.

In order to perform requested operation we chose a programmable element of pixel pipeline, i.e. pixel shader. The selection was based on a fact that pixel shader is

adjusted to be able to process large amount of elements in rather short time. It also allows us to influence every image element simply by rendering a quad over requested area.

There are several versions of pixel shader, differencing in their capabilities. We decided to use version 2.0 because it is a version that is most available today and provides reasonable functionality at the same time. It also usually supports floating-point operations in hardware and data sources even though on some hardware 32 bit IEEE floats are operated with lower accuracy 24 bit arithmetic. However, our experiments show that this lack of accuracy has almost no influence to results when compared to CPU computed values.

In order to be able to perform operations on image we have to transfer data from CPU memory to GPU, i.e. to video memory. For storing of data in video memory, floating-point textures are used. Later during a computation data are sampled by nearest-neighbor sampling and addressed in texture by texture coordinates that are shifted by one half of pixel in order to prevent coordinate rounding issues. All coordinates are linearly interpolated over whole quad implicitly by hardware itself based on coordinates in corners that are computed by CPU.

The operations performed on GPU can be expressed as a simple flow chart shown in Figure 2. The most time consuming part is a reduction step that involves a loop, which cannot be implemented directly in pixel shader program due to flow control limitations of selected pixel shader version.

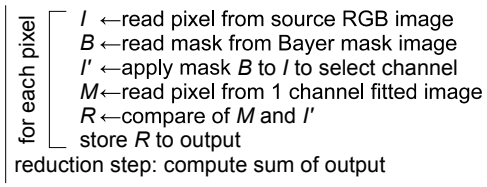

Figure 2: Operations performed on GPU.

In order to bypass this limitation multiple passes have to be performed. In each pass results of previous pass are divided into four quads and these quads are summed into a single one. This is repeated until the area is reduced to a single 1×1 matrix that is read back to CPU and divided by number of processed elements. The reason why the reduction operation is most time consuming is the fact that each step involves switching of render target that causes a pipeline to be flushed.

6. EXPERIMENTAL RESULTS

We have performed series of experiments to test proposed improvement of accuracy and to test performance gain by hardware implementation of registration.

Our first aim was to support the decision to exclude interpolated data from the input set. In our first experiment, we have simulated image acquisition process that takes place in a DSC. We have performed a simulation of integral sampling (basically averaging of certain areas of original image). Values that would not have been measured by Bayer array were removed from resulting image and demosaicking was performed (we have used Cok's constant hue algorithm described in [2], which is aimed to reduce color artifacts on intensity edges).

Subsequently, we have performed series of experiments with Zomet's SR method (which has shown best robustness against errors in input images), showing dependency of the MSE (Mean Squared Error) of the original and reconstructed image according to changing β parameter. Two series of experiments were performed, first utilizing all input data, i.e. original Zomet algorithm, other utilizing only data not influenced by demosaicking, i.e. using proposed improvement. The resulting images are shown in Figure 3 a—f, measured MSE is shown in Figure 4.

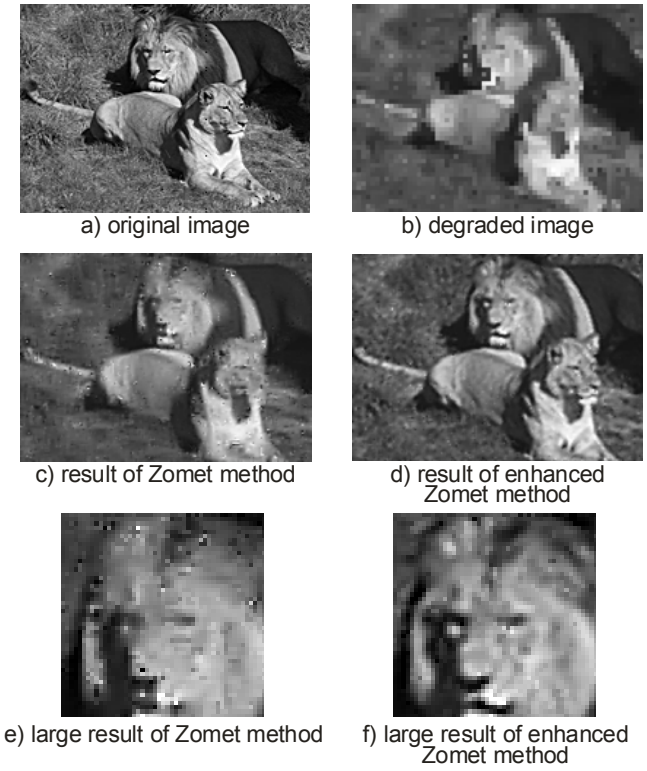

Figure 3: Experimental results

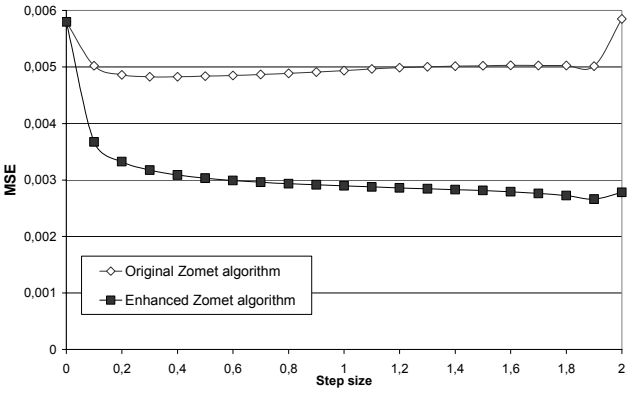

Figure 4: Accuracy results of improved method

One can see that MSE was reduced by 40-50%, similar results were obtained for other input images we have used.

This result along with other experiments shown on our website[1] show that using measured data only justifies reducing the input data set to one third of its original size.
In order to prove whether we have achieved a speedup of computation by utilizing GPU we compared pure software solution with GPU aided one. We used GPU close to current hi-end available on market. We didn't use the top of current GPU because we expected our solution to be used on hardware close to current average rather then top-end. Just to illustrate the power of GPU aided computation we compared times achieved on very slow machine equipped with fast GPU to fast machine equipped with slightly weaker GPU.

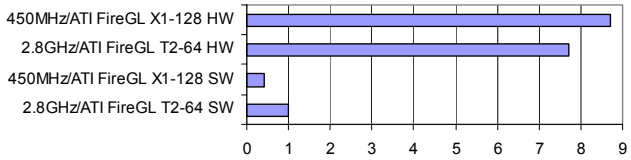

Figure 5: Relative speedup of registration.

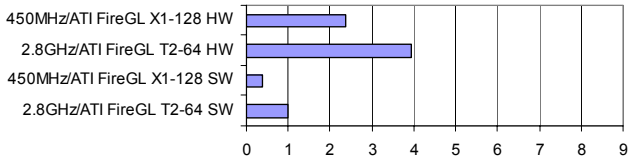

Figure 6: Relative speedup of whole SR process.

As it is clearly visible from Figures 5 and 6, times achieved by GPU aided solution are more than 8 times faster in comparison with non-GPU solution on the same hardware configuration. It is clear from our experiment that even a machine with a slower CPU may provide reasonable performance when aided by GPU.

7. CONCLUSIONS

We have tested SR algorithms suitable for enhancing resolution of images gained by usual DSC. We have explored preprocessing that is performed on the measured data and implemented a simulation of such preprocessing.
We have proposed an enhancement of spatial SR methods that is applicable on all three (IBP, Zomet, Farsiu) implemented SR algorithms, which exploits the knowledge about preprocessing that is performed within the camera. We have tested this enhancement and presented results showing that it may reduce MSE by almost 50%.
We have utilized GPU in order to improve computation times. As we have shown above that we have achieved this goal by decreasing the computation time to 1/8 of pure software solution. We are sure that this decrease will be higher for larger processed images. Due to the fact that we utilize .NET as runtime environment for both pure software and GPU aided solution, the real speedup would be a little bit lower. However, even in such case the gained speedup and the fact that similar GPU as the one we have used are available in a majority of home PCs today surely increases the range of possible use of our GPU aided solution.

[1] http://herakles.zcu.cz/research/superresolution

8. FUTURE WORK

We would like to consider an advanced registration algorithms including rotation estimation of input images. We hope that such improvement will help to bring SR methods closer to practice and enable an easy SR processing of images taken by digital camera held in hand.
Using advanced demosaicking techniques (such as [10]) for the initial approximation may also improve accuracy of our results. Later on we want to increase the actual speedup by utilizing advanced hardware features.

9. ACKNOWLEDGEMENTS

This work was supported by Ministry of Education, Czech Republic: project MSMT 235200005. The authors would like to thank Microsoft and ATI for providing hardware for testing purposes.

REFERENCES

[1] S. Baker, T. Kanade, Limits on Super-Resolution and How to Break Them. *Proc.CVPR00*, 2000
[2] D. R. Cok, *Signal Processing method and apparatus for producing interpolated chrominance values in a sampled color image signal*. U.S. Patent No. 4,642,678 (1987)
[3] M. Elad, A. Feuer, Restoration of a Single Superresolution Image from Several Blurred, Noisy, and Undersampled Measured Images. *IEEE Transactions on Image Processing*, Vol. 6, No. 12, December 1997
[4] M. Elad, Y. Hel-Or, A Fast Super-Resolution Reconstruction Algorithm for Pure Translation Motion and Common Space-Invariant Blur. *IEEE Transactions on Image Processing*, Vol. 10, No. 8, pp. 1187-1193, August 2001.
[5] S. Farsiu, D. Robinson, M. Elad, P. Milanfar, Fast and Robust Super-Resolution. *Proceedings of ICIP* 2003
[6] W. T. Freeman, *Median filter for reconstructing missing color samples*. U.S. Patent No. 4,724,395 (1988)
[7] J. F. Hamilton, J. E. Adams, *Adaptive color plane interpolation in single sensor color electronic camera*. U.S. Patent No. 5,629,734 (1997)
[8] R. C. Hardie, K. J. Barnard, E. E. Armstrong, Joint MAP Registration and High-Resolution Image Estimation Using a Sequence of Undersampled Images. *IEEE Transactions on Image Processing*, Vol. 6, No. 12, December 1997
[9] T. Huang, R. Tsai, Multi-frame image restoration and registration, *Advances in Computer Vision and Image Processing*. volume 1, pages 317-339. JAI Press Inc., 1984
[10] R. Kimmel, Demosaicking: image reconstruction from CCD samples. *Proc. Trans. Image Processing*, vol. 8, pp. 1221-1228, 1999
[11] C. A. Laroche, M. A. Prescott, *Apparatus and method for adaptively interpolating a full color image utilizing chrominance gradients*. U.S. Patent No. 5,373,322 (1994)
[12] A. Lukin, D. Kubasov, An Improved Demosaicking Algorithm. *Proceedings of Graphicon* 2004
[13] S. Peleg, D. Keren, L. Schweitzer, Improving image resolution using subpixel motion. *Pattern Recognition Letter*, vol. 5, pp. 223-226, March 1987
[14] R. Ramanath, W.E. Snyder, G.L. Bilbro, and W.A. Sander, Demosaicking methods for Bayer color arrays. *Journal of Electronic Imaging*, vol. 11, no. 3, pp. 306-315, Jul. 2002.
[15] A. Zomet, A. Rav-Acha, S. Peleg, Robust Super-Resolution, *Proceedings of the Int. Conf. on Computer Vision and Pattern Recognition* (CVPR), vol. 1, pp. 645-650, Dec. 2001