AN IMAGE RESAMPLING METHOD USING COMBINED DIRECTIONAL KERNELS

Andrey Nasonov, Andrey Krylov, Konstantin Chesnakov

Laboratory of Mathematical Methods of Image Processing
Faculty of Computational Mathematics and Cybernetics
Lomonosov Moscow State University
kryl@cs.msu.ru

ABSTRACT

A new edge-directional image resampling method is proposed. The method uses a weighted sum of two adaptive 4x4 interpolation kernels to construct high-resolution pixels. The weights are chosen according to the local gradient features for each pixel. The interpolation kernels are learned using pairs of low- and high-resolution images taken from LIVE image database. The method has low complexity and low memory consumption. It outperforms existing fast edge-directional image interpolation methods.

Index Terms— Fast image resampling, edge-directional interpolation

1. INTRODUCTION

Fast and high quality methods of image upscaling are used in a large area of image processing applications. It is especially important for showing low-resolution content in modern high definition television. Super-resolution algorithms with a magnification factor of 2 are demanded as a base of arbitrary factor magnification methods.

The simplest algorithms are general purpose linear interpolation methods like bilinear, bicubic, and Lanczos interpolation [1]. They are fast but do not perform edge-directed interpolation.

The more complicated edge-directional image interpolation algorithms use prior information about images. The method [2] improves linear interpolation by kernel elongation along edges. This approach produces excellent results at straight edges but fails at image corners and textured areas. NEDI [3] algorithm uses self-similarity of the natural images at different scales. It works well at edges and corners but may corrupt image textures. Edge-guided image interpolation (EGII) [4], Iterative curvature-based interpolation (ICBI) [5] and Directional cubic convolution interpolation (DCCI) [6] algorithms combine the interpolation by two directions at each pixel. The coefficients of the combination are chosen according to the local direction features.

Regularization-based algorithms pose the image upscaling problem as a functional minimization problem with data-fitting and stabilizer terms [7, 8]. The stabilizer holds prior information about the image. Usually used total variation stabilizer smooths the image while keeping edges sharp. The functional minimization problem is computationally very expensive.

Learning-based algorithms use a dictionary containing pairs of low-resolution and corresponding high-resolution image patches. State-of-the-art algorithms use convolutional neural networks [9]. They produce excellent results but are very slow due to a large number of convolutions. Adaptive regression algorithms classify each pixel into a class and apply linear transform to construct a high-resolution patch [10, 11].

In this paper, we present a new low complexity image resampling algorithm based on the ideas of DCCI and regression-based algorithms.

2. THE PROPOSED ALGORITHM

The proposed algorithm is based on DCCI algorithm [6]. The main idea of DCCI is the interpolation procedure that takes a 4×4 pixel block as the input and interpolates the central pixel of the block. The upscaling the a factor of 2 is achieved by applying the interpolation procedure twice: the second time it is applied with 45° rotation (see Figures 1, 2 and 3).

We have made the following improvements to the DCCI algorithm:

1. Adaptive 4x4 kernel is used instead of fixed cubic interpolation using only 4 pixels.

2. An adaptive sharpening is applied before image interpolation to reduce the blur introduced by anti-aliasing filtering during producing the low-resolution image.

Let $u_{i,j}$ be the input low-resolution image, $v_{i,j}$ — the reconstructed high-resolution image.

The method consists of three steps and uses interpolation kernels A, B and C that are obtained from a learning process (see section 2.4).

2.1. First step

To avoid aliasing effect during image downsampling, blur operator is usually applied to the high-resolution image. It

means that there is no direct correspondence between pixels of low- and high-resolution images.

At the first step, we apply adaptive deblurring operator to the low-resolution image to compensate blur introduced during downsampling process:

$$v_{2i,2j} = (u * A)_{i,j}, \tag{1}$$

where A is 5×5 filter.

2.2. Second step

At the second step, pixels with odd coordinates $v_{2i+1,2j+1}$ are interpolated by 4×4 block of surrounding pixels of the low-resolution image (see Fig. 1).

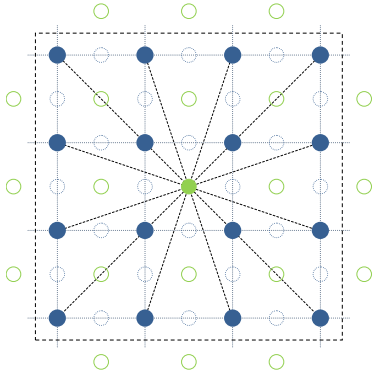


Fig. 1. Interpolation of pixels with odd coordinates at the second step. Blue pixels are pixels of the low-resolution image. The green pixel is the interpolated pixel of the high-resolution image.

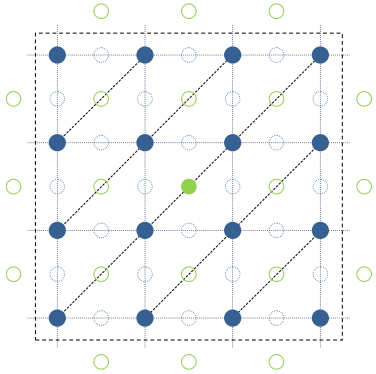


Fig. 2. Calculation of the directional power at the interpolated pixel at the second step as the sum of absolute differences represented by diagonal lines.

First, the directional power values for $45°$ and $135°$ diagonals are calculated. The directional power value is equal to the sum of absolute differences in 3x3 block (see Fig. 2):

$$p_{2i+1,2j+1} = \sum_{n,m=0}^{2} |u_{i+n,j+m} - u_{i+n-1,j+m-1}|,$$

$$q_{2i+1,2j+1} = \sum_{n,m=0}^{2} |u_{i+n-1,j+m} - u_{i+n,j+m-1}|.$$

Next, the directional weight coefficient is calculated:

$$w_{2i+1,2j+1} = \frac{1 + p^k_{2i+1,2j+1}}{2 + p^k_{2i+1,2j+1} + q^k_{2i+1,2j+1}}.$$

Parameter k is chosen experimentally, good results are obtained with $k = 6$ [6].

The weight coefficient is close to either 0 or 1 if the image content is strongly oriented and close to $1/2$ if there is no prevalent direction.

Finally, the high-resolution pixels are interpolated:

$$\begin{aligned} v_{2i+1,2j+1} = {} & w_{2i+1,2j+1}(u * B)_{i,j} + \\ & + (1 - w_{2i+1,2j+1})(u * B')_{i,j}, \end{aligned} \tag{2}$$

where B and B' are 4×4 interpolation kernels for $45°$ and $135°$ respectively.

2.3. Third step

At the third step, the rest of pixels are interpolated. The procedure is similar to the second step but the image is $45°$ rotated (see Fig. 3). Previously interpolated pixels of the high-resolution image calculated at the first two steps are used.

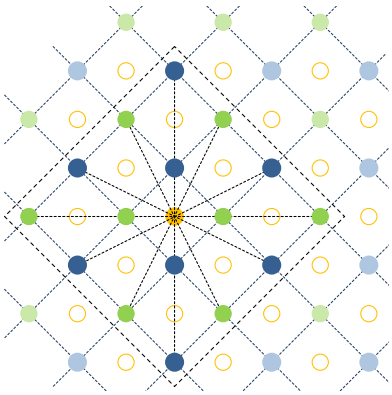


Fig. 3. Interpolation of pixels at the third step.

The direction power values and interpolated pixels are calculated as follows:

$$p_{i,j} = \sum_{n,m=0}^{2} |v_{i+n-m,j+n+m-1} - v_{i+n-m,j+n+m-3}|,$$

$$q_{i,j} = \sum_{n,m=0}^{2} |v_{i+n+m-1,j+n-m} - v_{i+n+m-3,j+n-m}|,$$

$$w_{i,j} = \frac{1 + p_{i,j}^k}{2 + p_{i,j}^k + q_{i,j}^k},$$

$$v_{i,j} = w_{i,j}(u * C)_{i,j} + (1 - w_{i,j})(u * C')_{i,j},$$

where C and C' are 4×4 interpolation kernels for $0°$ and $90°$ respectively.

2.4. Learning interpolation kernels

The interpolation kernels are learned using pairs of corresponding low- and high-resolution images by minimizing the square error sum. Assuming u and v are known images, we find the coefficients A such that

$$\sum_{i,j} \left((u * A)_{i,j} - v_{2i,2j}\right)^2 \rightarrow min. \quad (3)$$

In the case of several training image pairs, we minimize the sum for all the images. This quadratic minimization problem leads to a system of linear equations that is solved trivially. The coefficients of B and C are found in the same way.

To reduce the number of independent coefficients in the interpolation kernels, we restrict the proposed algorithm to be invariant to image rotation, mirroring and transposition:

$$A = \begin{bmatrix} A_1 & A_2 & A_3 & A_2 & A_1 \\ A_2 & A_4 & A_5 & A_4 & A_2 \\ A_3 & A_5 & A_6 & A_5 & A_3 \\ A_2 & A_4 & A_5 & A_4 & A_2 \\ A_1 & A_2 & A_3 & A_2 & A_1 \end{bmatrix},$$

$$B = \begin{bmatrix} B_1 & B_2 & B_3 & B_4 \\ B_2 & B_5 & B_6 & B_3 \\ B_3 & B_6 & B_5 & B_2 \\ B_4 & B_3 & B_2 & B_1 \end{bmatrix}, B' = \begin{bmatrix} B_4 & B_3 & B_2 & B_1 \\ B_3 & B_6 & B_5 & B_2 \\ B_2 & B_5 & B_6 & B_3 \\ B_1 & B_2 & B_3 & B_4 \end{bmatrix},$$

$$C = \begin{bmatrix} & & & C_1 & & & \\ & & C_2 & & C_2 & & \\ & C_3 & & C_5 & & C_3 & \\ C_4 & & C_6 & & C_6 & & C_4 \\ & C_3 & & C_5 & & C_3 & \\ & & C_2 & & C_2 & & \\ & & & C_1 & & & \end{bmatrix},$$

$$C' = \begin{bmatrix} & & & C_4 & & & \\ & & C_3 & & C_3 & & \\ & C_2 & & C_6 & & C_2 & \\ C_1 & & C_5 & & C_5 & & C_1 \\ & C_2 & & C_6 & & C_2 & \\ & & C_3 & & C_3 & & \\ & & & C_4 & & & \end{bmatrix}.$$

In this case, the number of independent coefficients of kernel A is equal to 6 (instead of 25), for pair of kernels B and B' is equal to 6 and for pair of kernels C and C' is equal to 6 too. It means that a total of 18 coefficients should be obtained.

The proposed algorithm was trained using 29 reference images from LIVE database [12, 13] containing photographic images of nature, humans and buildings. The reference images were downsampled by 2 times using Gauss filtering with $\sigma = 0.4$ followed by taking every second pixel.

The obtained coefficients of the kernels are the following:

$$\overline{A} = (-0.00063, -0.00180, 0.02694, 0.01716, -0.15097, 1.45306),$$

$$\overline{B} = (0.03297, -0.04665, -0.04484, -0.05382, 0.12650, 0.58139),$$

$$\overline{C} = (0.04283, -0.05838, -0.04952, -0.03016, 0.25095, 0.45204).$$

3. RESULTS

The proposed algorithm was compared with existing image resampling algorithms of the same computational complexity (image processing time). The main competitors are DCCI [6] and SI [11]. DCCI algorithm uses fixed one-dimensional cubic interpolation instead of adaptive 4x4 kernels. The algorithm SI uses linear mapping from low-resolution patches into 2x2 blocks in the high-resolution image. The authors propose 3 modifications: 3x3 to 2x2 (SI-1), 5x5 to 2x2 (SI-2) and 7x7 to 2x2 (SI-3) mappings. We compare with SI-1 method only because SI-2 and SI-3 are slower than the proposed method. The algorithm SI-1 has been trained using the same training image set as the proposed method.

The results for the factor of 2 are shown in Table 1, Table 2, Fig. 4 and Fig. 5. We have used the same downsampling routine to generate the low-resolution images as for the training stage.

	Bicubic	DCCI [6]	SI-1 [11]	Proposed
Baboon	23.850	23.909	24.179	24.183
Cameraman	25.934	26.315	27.183	26.667
Lena	34.076	34.462	35.041	35.243
Peppers	33.595	33.927	34.676	34.366
Average	29.364	29.653	30.270	30.115

Table 1. Comparison of the proposed method with existing image resampling algorithms (PSNR)

It can be seen that SI-1 has slightly better PSNR and SSIM values for some images but has unwanted 2x2 blocking artifacts at flat areas with gradual intensity change (see Fig. 4 and Fig. 5). This is caused by influence of thresholding in mapping class selection: small changes result in using different mapping class.

	Bicubic	DCCI	SI-1	Proposed
Baboon	0.804	0.805	0.832	0.834
Cameraman	0.906	0.909	0.926	0.919
Lena	0.952	0.953	0.959	0.958
Peppers	0.940	0.940	0.946	0.944
Average	0.900	0.902	0.916	0.914

Table 2. Comparison of the proposed method with existing image resampling algorithms (SSIM)

It takes about 15 ms to perform $512 \times 512 \rightarrow 1024 \times 1024$ upsampling of color image on Intel Core i7 2600K CPU that is about $1.5\times$ slower than DCCI and is equal to SI-1 algorithm. The SRCNN algorithm takes several seconds to process the image.

Although the proposed method has practically the same performance and quality as SI-1, it has an advantage over SI-1 in memory consumption. The proposed method requires only a few bytes of additional memory to store the coefficients of interpolation kernels while SI-1 needs to store hundreds of interpolation kernels. It makes possible the proposed algorithm to be implemented on chip.

The computation complexity is the following. For the input 4×4 grayscale pixel block returning single pixel, it takes 60 additions and subtractions, 18 multiplications, 1 division and 18 logical operations. It can be optimized by reusing the values of directional derivatives when processing adjacent blocks up to 44 additions and subtractions, 18 multiplications, 1 division and 2 logical operations.

The software implementation of the proposed algorithm and our implementations of SI-1 and DCCI are available at the site http://imaging.cs.msu.ru/software.

4. CONCLUSION

A fast edge-directional image resampling algorithm has been proposed. In comparison with existing image resampling algorithms with the same computational complexity, the proposed method shows good objective quality and has low memory consumption. The advantage of the proposed method over DCCI algorithm has been obtained using adaptive 4x4 kernel instead of cubic interpolation and filtering of pixels of the low-resolution image.

The future work will include an adaptation of the proposed algorithm for fast video resampling.

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

5. REFERENCES

[1] Thierry Blu, Philippe Thévenaz, and Michael Unser, "Linear interpolation revitalized," *Image Processing, IEEE Transactions on*, vol. 13, no. 5, pp. 710–719, 2004.

[2] Hiroyuki Takeda, Sina Farsiu, and Peyman Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 349–366, 2007.

[3] X. Li and M.T. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing*, vol. 10, pp. 1521–1527, 2001.

[4] L. Zhang and X.L. Wu, "An edge-guide image interpolation via directional filtering and data fusion," *IEEE Transactions on Image Processing*, vol. 15, pp. 2226–2235, 2006.

[5] A. Giachetti and N. Asuni, "Real time artifact-free image interpolation," *IEEE Transaction on Image Processing*, vol. 20, no. 10, pp. 2760–2768, 2011.

[6] D. Zhou, X. Shen, and W. Dong, "Image zooming using directional cubic convolution interpolation," *IET Image Processing*, vol. 6, no. 6, pp. 627–634, 2012.

[7] Hussein A Aly and Eric Dubois, "Image up-sampling using total-variation regularization with a new observation model," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1647–1659, 2005.

[8] A. S. Krylov, A. S. Lukin, and A. V. Nasonov, "Edge-preserving nonlinear iterative image resampling method," in *Proceedings of International Conference on Image Processing (ICIP'09)*, 2009, pp. 385–388.

[9] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Learning a deep convolutional network for image super-resolution," *Computer Vision–ECCV 2014*, pp. 184–199, 2014.

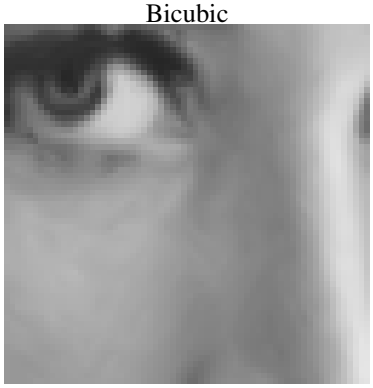
[10] Radu Timofte, Vincent De Smet, and Luc Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," *Computer Vision–ACCV 2014*, pp. 111–126, 2014.

[11] Jae-Seok Choi and Munchurl Kim, "Super-interpolation with edge-orientation based mapping kernels for low complex 2x upscaling," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 469–483, 2015.

[12] Hamid Rahim Sheikh, Muhammad Farooq Sabir, and Alan Conrad Bovik, "A statistical evaluation of recent full reference image quality assessment algorithms," *IEEE Transactions on Image Processing*, vol. 15, no. 11, pp. 3440–3451, 2006.

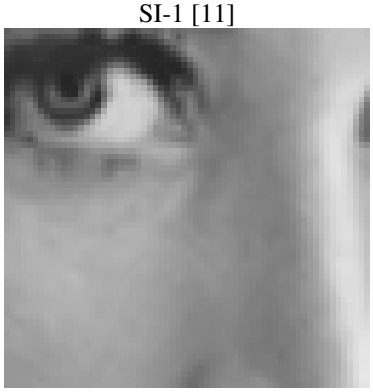
[13] H.R. Sheikh, Z.Wang, L. Cormack, and A.C. Bovik, "Live image quality assessment database release 2," `http://live.ece.utexas.edu/research/quality`.

Bicubic



DCCI [6]

SI-1 [11]



Proposed

Fig. 4. Comparison of the proposed method with existing image resampling algorithms (lena)

Bicubic

DCCI [6]

SI-1 [11]

Proposed

Fig. 5. Comparison of the proposed method with existing image resampling algorithms (cameraman).