A PATTERN-BASED INTER-/EXTRA-POLATION APPROACH FOR IMAGE SCALING[*†]

Jen-Hui Chuang[1], *Horng-Horng Lin*[1], *and Szu-Hui Wu*[1,2 ‡]

[1]Dept. of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan
[2]AU Optronics Corp., Hsinchu Science Park 300, Taiwan

ABSTRACT

A novel approach to simultaneous scaling and enhancing of a digital image is proposed, where efficient code matching is utilized to classify various edge/corner patterns. Adaptive schemes of inter-/extra-polations are adopted to maintain a balance of smoothness and sharpness of the scaled image, with unlimited scaling factors. Unlike many complex methods for image scaling and enhancement, our approach is very simple, making hardware implementation feasible.

Index Terms – Image Scaling, Enhancement, Hardware Design

1. INTRODUCTION

With rapid manufacture growth of large-display, developing low-complexity image magnification and enhancement methods for hardware implementation becomes more and more important. While many works for enhancing scaled image qualities have been proposed in decades, most of them are designed for software implementations, without considering hardware restrictions such as memory limitations and computing power. This motivates us to propose a simple scaling approach that can (i) cope with typical interpolation design, e.g., bi-cubic; (ii) use only simple arithmetic operations and limited memory buffers; (iii) enlarge images up to arbitrary scales; and (ix) maintain sharp object boundaries in scaled image outputs.

Interpolation approaches such as nearest-neighbor, bi-linear and bi-cubic [1] are commonly adopted in hardware design of image scaling. Yet blurring and aliasing effects often come along with these methods. Intuitively, post-processing methods, e.g., hi-boost filtering [2], can be applied to enhance image sharpness. However selecting suitable filter parameters to prevent over-enhancement of noises for various image qualities would be difficult. Rather than post-processing, many approaches focus on solving image scaling and enhancement simultaneously [3-8]. The basic ideas include estimating local edge directions from source image patches, and magnifying boundary regions via directional interpolations. Since edge information is adopted in image scaling, shaper boundaries can be retained. Nevertheless, limited scaling factors [3] and costly edge orientation estimations [4-8] hinder them from becoming hardware solutions. Other methods, like PDE-based scaling [9] and super-resolution [10], also produce sharp scaled images, but the computational complexities are even higher.

For other works on image scaling for hardware design, Kim *et al.* propose an area pixel model to scale images and implement it in an FPGA [11]. However, improvement of image sharpness is less addressed in this low-complexity design. In [12], mesh-based triangulation is applied to model image edges and to control interpolation results. Yet, the need of image triangulation makes it different from our usage of simple regular grid processing.

2. PATTERN-BASED INTER-/EXTRA-POLATION

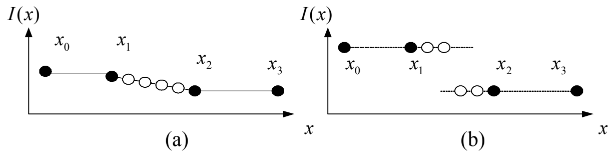


Figure 1. Interpolation and extrapolation for image scaling. The x and y axes denote image positions and intensities respectively. (a) An interpolation case. (b) An extrapolation case.

To illustrate the basic idea of the proposed method, some inter-/extra-polation examples of 1-D signals are given in Fig. 1. In Fig. 1a, since the underlying input signal varies smoothly, pure interpolation is chosen to add sample points between x_1 and x_2. Contrarily, as shown in Fig. 1b, when the intensity difference of x_1 and x_2 is large, a sharp boundary can be preserved by extrapolating new sample points from x_1 and x_2, respectively, toward the midpoint between them.

For 2-D scaling wherein new sample points need to be added in a square whose vertices are a set of 2×2 pixels of a low-resolution image, a similar processing can be applied. To that end, the square is firstly classified as a smooth area or containing predefined object edges/corners. Accordingly, for the former (latter) case, interpolations (extrapolations) are performed to generate sample points that produce a sharp boundary.

[*] This project was partly supported by AU Optronics Corp.
[†] The authors would like to thank Stanly Huang and Brian Hsieh from AU Optronics, and anonymous reviewers, for helpful suggestions.
[‡] Szu-Hui Wu was with National Chiao Tung University. She joined AU Optronics Corp. in March 2007.

2.1. Edge and Corner Patterns

As depicted in Fig. 2, we define edge and corner patterns in a 4×4 window, where the square of the central 2×2 area is the target region for extrapolating new sample points. A 4×4 window size is chosen because it contains more information than 2×2 or 3×3 window sizes for analyzing various directions of edges and corners. Besides, since a 4×4 window is also the basic unit for bi-cubic interpolation, our algorithm can be easily incorporated to the existing interpolation scheme without requiring additional buffers.

When defining edge and corner patterns, it is assumed that intensity values of pixels within a 4×4 window are of two groups. Through analyzing all the two-grouped pixel combinations, seven pattern categories are derived based on the central 2×2 pixels, including: A. horizontal type (Fig. 2a), B. vertical type (not shown for brevity), C. upper-left type (Fig. 2b), D. upper-right type, E. lower-right type, F. lower-left type, and G. cross type (Fig. 2c). And each type consists of edge patterns (e.g., A_1-A_7), corner patterns (e.g., A_8-A_{11}), and a default case (e.g., A_0).

In our definition, an edge direction should be supported by at least two boundary pixels, and a corner should be close to one of the central 2×2 pixels. Separation lines (green arrows in Fig. 2) of the two pixel groups are located in the middle of the estimated edge/corner areas. Their line functions can either be computed and stored in advance, or calculated in real-time. It is worth noticing that each pattern can actually be represented as a 16-bit code. For example, pattern A_4 can be coded as "X11X 0111 0000 X00X." This representation is particularly useful for pattern matching.

2.2. Image Pattern Matching

When scaling an input image, each 2×2 image block is firstly matched to the seven pattern categories by comparing their intensity differences. This first level matching involves only central 2×2 pixels of a 4×4 window. Two pixels, x and y, are said to be of the same group if their intensity difference is within a threshold T_i, i.e., $S(I_x - I_y)$ being true as $|I_x - I_y| \leq T_i$. On the other hand, they are of different groups if the difference is larger than a threshold T_e, i.e., $D(I_x - I_y)$ being true as $|I_x - I_y| \geq T_e$. If $T_i < |I_x - I_y| < T_e$, it is not taken into consideration. Accordingly the decision tree shown in Fig. 3 is adopted to check if a 2×2 block consists of two pixel groups. (For example, the top layer of the tree is interpreted as: *if* $(S(I_1 - I_2))$ {…} *elseif* $(D(I_1 - I_2))$ {…} *else* {*Bi-Cubic*}.)

For a 2×2 block belonging to one of the seven categories, there are two pixel groups inside, say black and white groups. Each group can then be represented by its mean intensity, e.g., μ_b and μ_w. We further match the block to edge, corner or default patterns by examining its intensity distribution in the corresponding 4×4 window. In order to use a code matching scheme, the code bit of each pixel x of the 4×4 window is set to "0", "1", and "X", if $|I_x - \mu_b| \leq T_i$, $|I_x - \mu_w| \leq T_i$, or neither holds respectively. Thus, a 16-bit code can be calculated and used to efficiently find a matched pattern. Note that in code matching, the matching priority is ordered as "edge > corner > default." The entire matching scheme has only two parameters and requires only simple arithmetic operations, making possible hardware implementation simple and efficient.

Once an edge/corner pattern is classified, various extrapolation schemes can be adopted to estimate new sample points from the two color groups in the 4×4 window. In this work, new sample points are simply colored with μ_b and μ_w, respectively. As indicated later in our experiments, such a simple scheme often gives sharp edge boundaries, but may cause certain cartoon effects in some occasions.

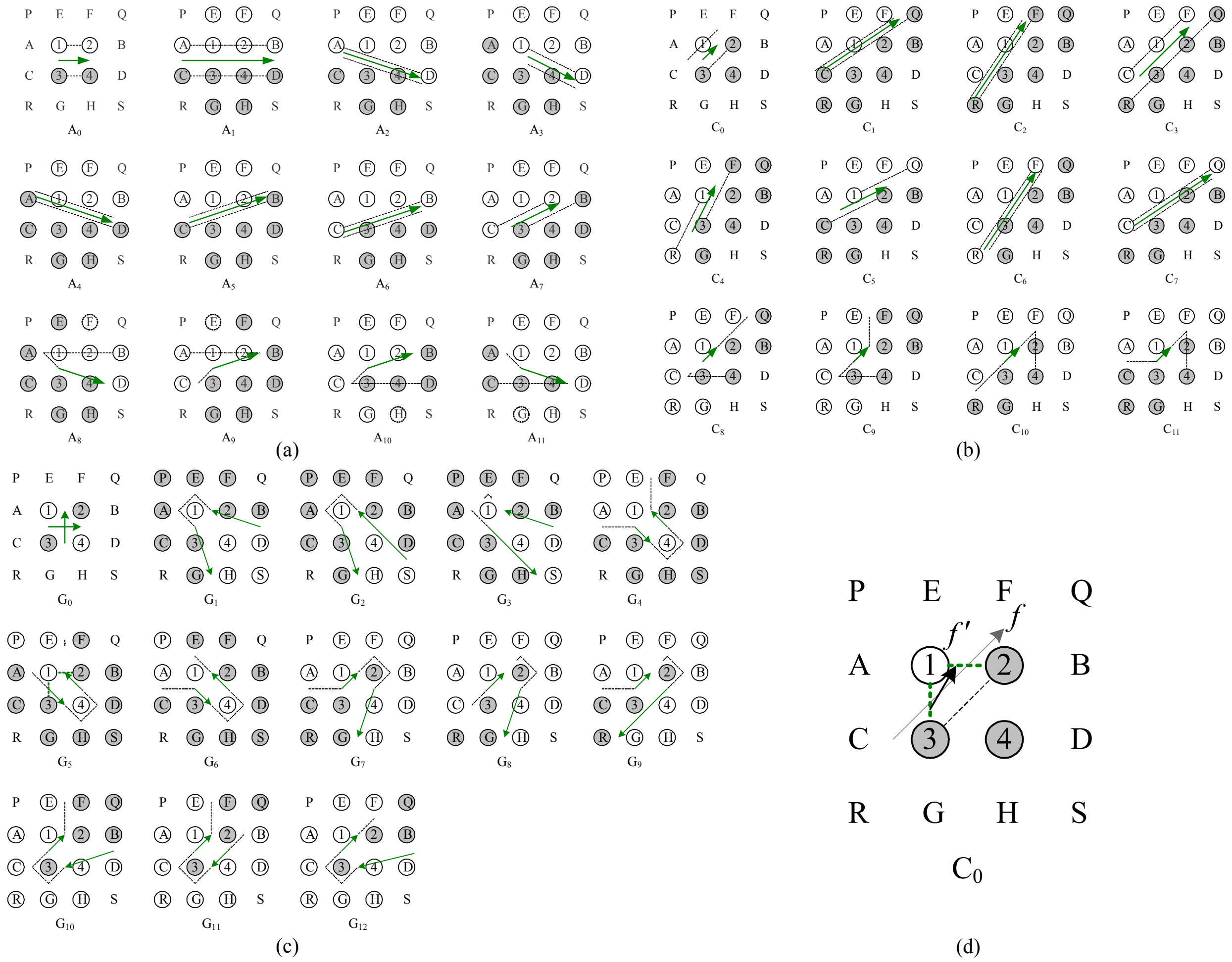
2.3. Boundary Continuity

By extrapolating edge or corner in consecutive image blocks using the finite set of pre-specified separation lines, endpoints of these lines on boarders of neighboring blocks may not coincide, resulting in discontinuities along object boundaries. To overcome such a drawback, a heuristic is proposed to improve boundary continuity by aligning endpoints on block boarders. We use an additional line buffer to record previous endpoints (in DOUBLE precision) on horizontal and vertical boarders of the current 2×2 block (green dotted lines in Fig. 2d). As a separation line is calculated for the current block, these recorded endpoints are used to modify the original line f so that the new line f' will result in better continuity in connecting edges across the boarders of neighboring blocks.

3. EXPERIMENTS AND CONCLUSION

Due to the page limitation, only two experimental results are demonstrated. For the pie drawing experiment in Fig. 4(a), we set $T_i = 0$ and $T_e = 1$ to exam the enlargement effects of edges and corners. In this case, the proposed method produces extremely sharp edge and corner boundaries, compared with nearest-neighbor, bi-cubic and hi-boost post-processing methods. Also, unlike hi-boost post-processing, color consistency of the boundary area is maintained by our approach. The flower image in Fig. 4(b) gives comparisons with bi-cubic, EDI [5], NEDI [6] and the proposed inter-/extra-polation. It is clear that the latter three methods produce sharper images than the first. Among the three enhancement methods, NEDI gives best visual appearance, EDI results in a sharper but slightly noisier image, and the inter-/extra-polation generates an output quality between those from EDI and bi-cubic. Also a slight cartoon effects can be perceived in our result, due to the choice of simple patch extrapolation for enhancing edges. Though the proposed method does not out-perform NEDI and EDI visually, it is more feasible for low-complexity design in hardware implementation.

To sum up, a novel image scaling method for enhancing object boundaries is proposed. An efficient code matching scheme for classifying edge/corner patterns and an effective heuristic for maintaining edge continuity are developed. Thus, efficient hardware implementation of the proposed approach is quite feasible.



P E F Q
A 1 2 B
C 3 4 D
R G H S
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11
(a)
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11
(b)
G0 G1 G2 G3 G4 G5 G6 G7 G8 G9 G10 G11 G12
(c)
f′
f
C0
(d)

Figure 2. Edge and corner patterns. Pixels within a 4×4 window are of two groups, except for the heuristically assigned don't-care pixels (un-circled). (a) Horizontal type. (b) Upper-left type. (c) Cross type. (d) Separation line function estimation.

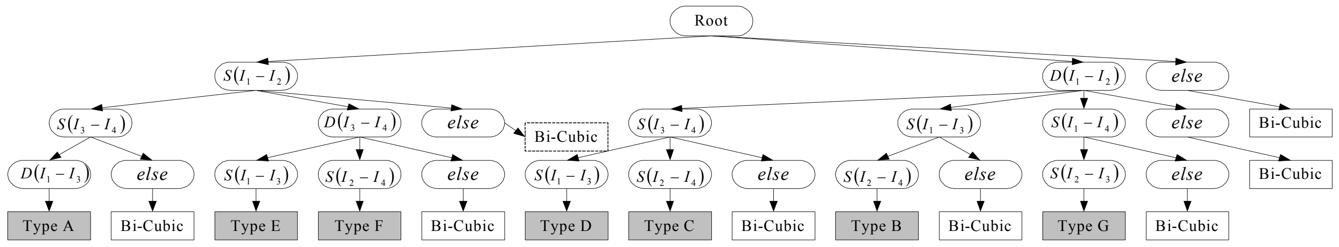


Figure 3. A decision tree for classifying a 2×2 block into seven types.

4. REFERENCES

[1] R.G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans. ASSP*, vol. 29, pp. 1153-1160, 1981.

[2] R.C. Gonzalez and R.E. Woods, *Digital Image Processing*, Prentice Hall, 2nd Ed., 2002.

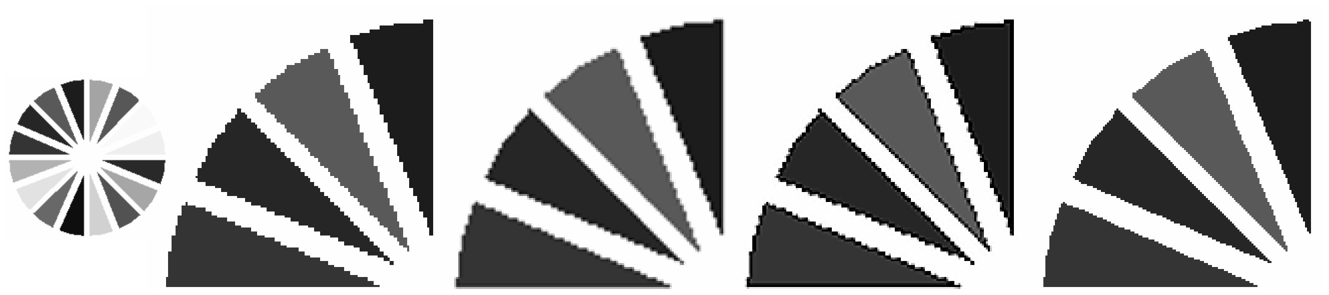
[3] S. Bayrakeri and R. Mersereau, "A New Method for Directional Image Interpolation," *Proc. IEEE ICASSP*, vol. 4, pp. 2383-6, 1995.

[4] K. Jensen and D. Anastassiou, "Subpixel Edge Localization and the Interpolation of Still Images," *IEEE Trans. IP*, vol. 4, no. 3, pp. 285-295, 1995.

[5] J. Allebac and P.W. Wong, "Edge-Directed Interpolation," *Proc. IEEE ICIP*, vol. 3, pp. 707-710, 1996.

[6] X. Li and M.T. Orchard, "New Edge-Directed Interpolation," *IEEE Trans. IP*, vol. 10, no. 10, pp. 1521-1527, 2001.

[7] X. Wu and X. Zhang, "Image Interpolation Using Texture Orientation Map and Kernel Discriminant," *Proc. ICIP*, vol. 1, pp. 49-52, 2005.


(a) Scaling from 180×180 to 300×300. From left to right: source image, nearest-neighbor, bi-cubic, bi-cubic+hi-boost, and inter-/extra-polation.

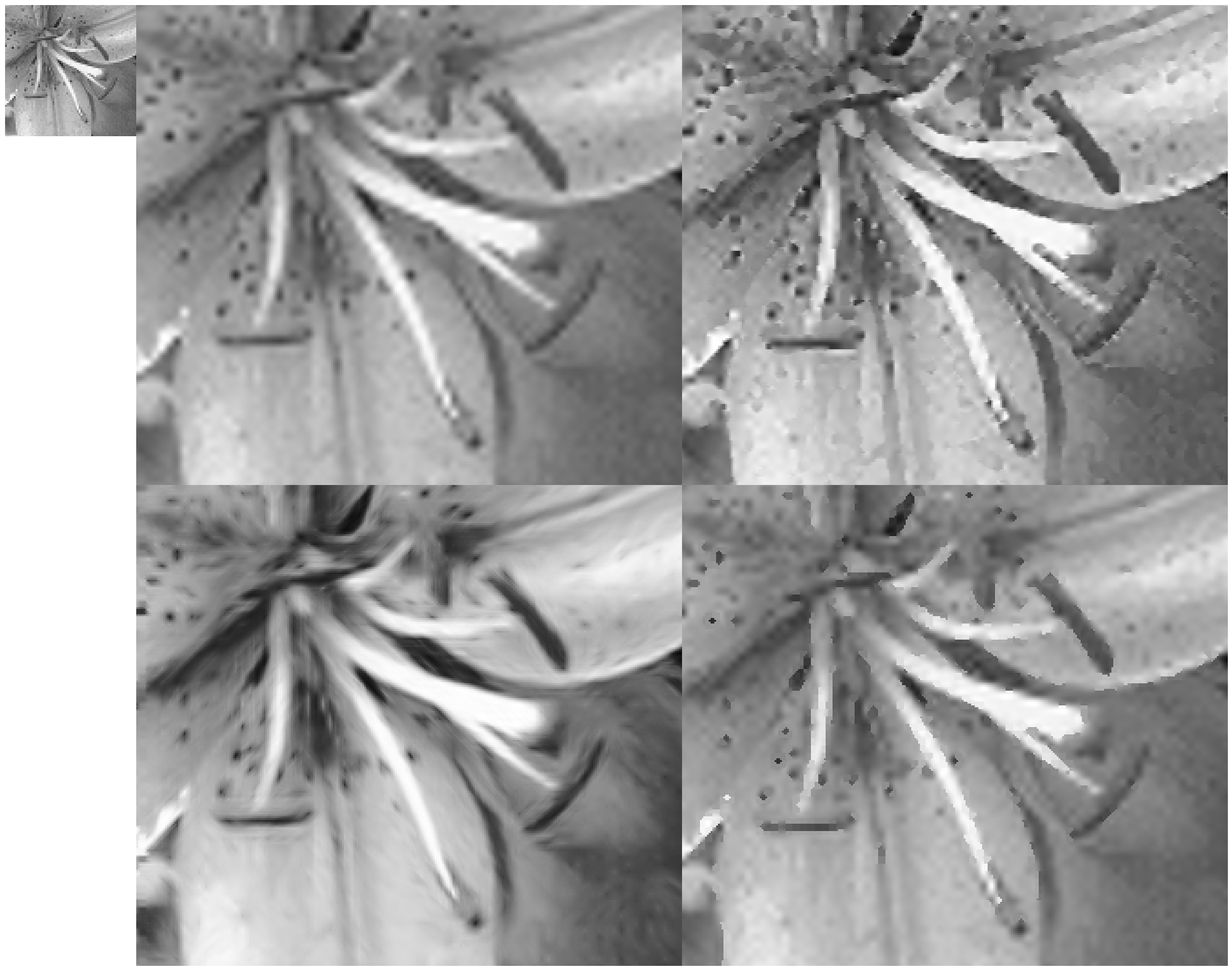

(b) Scaling from 75×75 to 300×300. Top row: source image, bi-cubic, and EDI [5]; bottom row: NEDI[6] and inter-/extra-polation.

Figure 4. Experimental comparisons of different scaling methods. (a) Pie drawing results. (b) Flow image results. (The source image, EDI and NEDI results are downloaded from "http://www.csee.wvu.edu/~xinl/demo/interpolation.html.")

[8] Q. Wang and R. Ward, "A Contour-Preserving Image Interpolation Method," *Proc. ICIP*, vol. 3, pp. 673-676, 2003.

[9] B. Morse and D. Schwartzwald, "Image Magnification Using Level-Set Reconstruction," *Proc. CVPR*, vol. 1, pp. 333-340, 2001.

[10] W.T. Freeman, T.R. Jones, and E.C. Pasztor, "Example-Based Super-Resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, 2002.

[11] C.H. Kim, S.M. Seong, J.A. Lee, and L.S. Kim, "Winscale: An Image-Scaling Algorithm Using an Area Pixel Model," *IEEE Trans. CSVT*, vol. 13, no. 6, pp. 549-553, 2003.

[12] D. Su and P. Willis, "Image Interpolation by Pixel Level Data-Dependent Triangulation," *Computer Graphics Forum*, vol. 23, no. 2, pp. 189-201, 2004.