Square to Hexagonal Lattice Conversion Based on One-Dimensional Interpolation

Xiangguo Li[1], B. Gardiner[2] and S.A. Coleman[2]
[1] College of Information Science and Engineering, Henan University of Technology, Zhengzhou, Henan, 450001 P.R. China
e-mail: xiangguoli@gmail.com
[2] School of Computing and Intelligent Systems, University of Ulster, Magee, BT48 7JL, N. Ireland.
e-mail: b.gardiner@ulster.ac.uk, sa.coleman@ulster.ac.uk

***Abstract*— This paper concerns the square lattice to hexagonal lattice conversion in practical hexagonal image processing, and presents a simplified conversion method that converts the common two-dimensional (2-D) interpolation approach to one-dimensional (1-D) interpolation. This paper is motivated by the sampling interval relationship between the square lattice and the hexagonal lattice, and assumes the 2-D interpolation kernel as separable, then changes the 2-D interpolation into successive 1-D interpolations, and finally reduces to the 1-D interpolation along the horizontal direction only. Compared with the common 2-D interpolation approach, the proposed simplified conversion method is more simple and more computationally efficient, and it is also more suitable for parallel processing. Finally, the experimental results verify the correctness as well as the computational efficiency.**

***Keywords*— Square sampling, hexagonal sampling, lattice conversion, separable filtering**

I. Introduction

The hexagonal lattice is the optimal sampling scheme for circularly band-limited analog images [1]. Compared with the common square lattice, the hexagonal lattice will provide 13.4% fewer samples, which means that the hexagonal sampling and processing will be more efficient in terms of both storage and computational costs; on the other hand, the hexagonal lattice is also superior with respect to its geometric properties, such as equidistant neighbors and uniform connectivity, as shown in Fig. 1. Besides, the hexagonal lattice is common in the structure of biological visual sensors, such as compound eyes of insects [2] and retina of human eyes [3], thus it has attracted the researchers in computer vision since the early years. In recent years, hexagonal image processing research has spread to applications such as edge detection [4], [5], image registration [6], [7], image restoration [8], hexagonal Gabor filtering [9], ultrasound image processing [10], computed tomography reconstruction [11], and adaptive beamforming [12].

However, the practical world predominantly uses the traditional rectangular/square lattice owing to its simplicity, especially the familiar Cartesian coordinate system. As a result, current commercial imaging devices, including imaging sensors (CCD and CMOS) and displays, are almost completely based on the square lattice. Therefore, in order to process the hexagonal images at the present time, the first task is to obtain the hexagonal lattice data by resampling from the original square lattice. Generally, lattice conversion is a two-dimensional (2-D) interpolation problem, and it can be easily understood from the reconstruction and resampling framework. That is, according to sampling theory, if the Nyquist criterion has been fulfilled, the original analog image can be

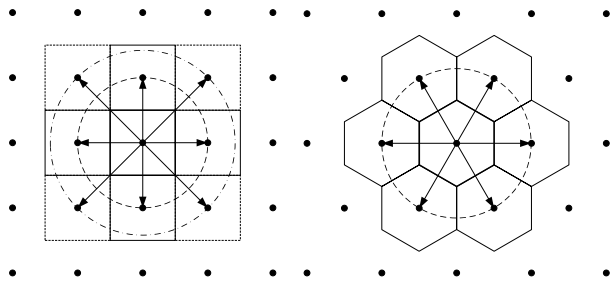


Fig. 1. Illustration of the equidistant neighbors and the uniform connectivity geometric properties between the square lattice (left) and the regular hexagonal lattice (right).

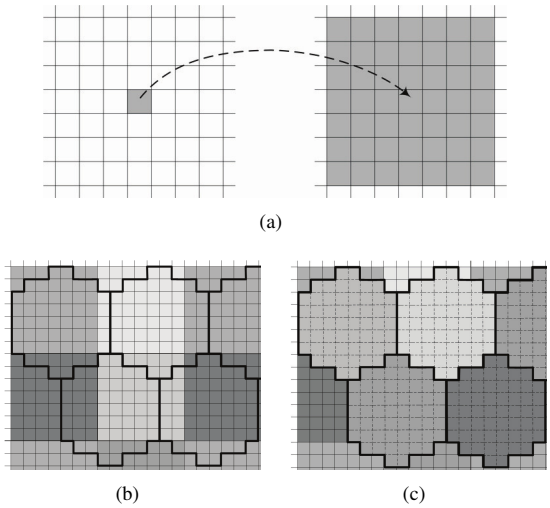


Fig. 2. Illustration of the square to hexagonal lattice conversion by the pixel clustering approach. (a) one square pixel is mapped to a new pixel with 49 sub-pixels (7×7), (b) every 56 sub-pixels are clustered together for the corresponding hexagonal pixel, and (c) the value of each hexagonal pixel is by the average intensity of the sub-pixels within each cluster.

completely reconstructed, and the new sampled data on the expected lattice can be obtained by resampling the reconstructed image with the according lattice [13]. In practice, simple 2-D interpolation kernels are often used, such as nearest-neighbor interpolation [14] and bilinear interpolation [15]. To reduce aliasing artifacts in classical interpolation methods, Ville et al. [16] proposed to use hexagonal spline functions. Later, Condat et al. [17] proposed a reversible conversion method that is similar to the 3-pass shear rotation algorithm. In addition, an intuitive and commonly used approach is based on pixel clustering [18], [5], as illustrated in Fig. 2, in which the original square lattice data are first upsampled to a more denser version and then several square sub-pixels of the intermediate data are clustered together to be mapped into an equivalent hexagonal pixel.

In hexagonal image processing research, it is inevitable that researchers will perform comparisons between an algorithm on the hexagonal lattice and its counterpart on the square lattice; therefore, for better comparison, it's preferred that the source data on both sides are equivalent in the experiments, i.e., high quality lattice conversion is important in such cases. To reach this goal, more accurate but also larger interpolation kernels are generally needed. However, due to the 2-D interpolation nature, the computational costs will be dramatically increased. For this matter, this paper aims to simplify the conversion process and convert the 2-D interpolation problem to 1-D interpolations along the horizontal direction only.

This paper is mainly motivated by the relationship between the square lattice and the hexagonal lattice. If we sample a circularly band-limited analog image using the square lattice and the regular hexagonal lattice separately and compare the interval values of the two sampling lattices in the corresponding directions, we can observe that, if aliasing is avoided exactly, according to the sampling requirements [1], the interval values in one of the two orthogonal directions are different but the interval values in the other direction are the same (for further details, see Sec. II). This hints that the lattice conversions may be implemented only along the direction that has different interval values. In this paper, this direction is the horizontal direction. Indeed, if we treat the 2-D interpolation kernel as a separable one, we can convert the 2-D interpolation into successive 1-D interpolations, and can further reduce the successive 1-D interpolations to 1-D interpolation only along the horizontal direction. Obviously, compared with the original 2-D interpolation approach, the proposed conversion method is more simple and more computationally efficient, and it is also more suitable to utilize the common parallel computation resource. Finally, experiments are performed to verify the correctness and the efficiency.

This paper is organized as follows. Section II summarizes the 2-D sampling theory, including the square lattice sampling and the hexagonal lattice sampling. Section III introduces the proposed conversion method. Section IV provides the experimental results and presents analysis of these results. Finally, a conclusion is presented in Section V.

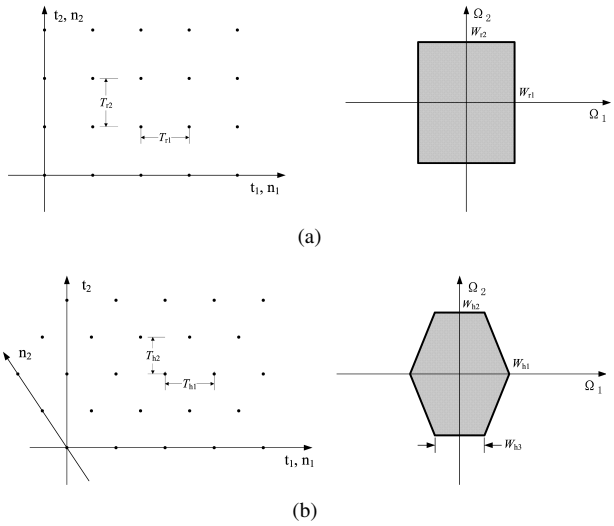


Fig. 3. Illustration of sampling lattices and the corresponding band regions (redrawn from [1]). (a) denotes the rectangular lattice and (b) denotes the hexagonal lattice.

II. 2-D SAMPLING THEORY

2-D sampling theory is the basis for the reconstruction and resampling framework, and it is also the motiviation for the proposed conversion method, thus we briefly introduce the 2-D sampling theory.

Given a 2-D analog image $x_a(t_1, t_2)$, we consider sampling it with the rectangular lattice and the hexagonal lattice, respectively. First, the 2-D rectangular sampling is straightforward to the 1-D sampling and can be expressed as $x(n_1, n_2) = x_a(n_1 T_{r1}, n_2 T_{r2})$, where T_{r1} and T_{r2} are the horizontal and vertical sampling intervals, respectively. If $x_a(t_1, t_2)$ is rectangular band-limited, as shown in Fig. 3(a), the sampling intervals to avoid aliasing must meet:

$$T_{r1} \leq \frac{\pi}{W_{r1}} \text{ and } T_{r2} \leq \frac{\pi}{W_{r2}} \ , \tag{1}$$

where W_{r1} and W_{r2} are the horizontal and vertical bandwidths in radians, respectively.

The hexagonal sampling lattice is illustrated in Fig. 3(b), in which the two sampling directions are skewed but note that the sampling intervals are defined in the two orthogonal directions. In this case, the sampling expression is given by $x(n_1, n_2) = x_a((n_1 - \frac{1}{2}n_2)T_{h1}, n_2 T_{h2})$. If we assume that $x_a(t_1, t_2)$ is band-limited with a hexagonal band region with W_{h1}, W_{h2}, and W_{h3} defined in the figure, the sampling intervals to avoid aliasing must fulfill [1]:

$$T_{h1} \leq \frac{4\pi}{2W_{h1} + W_{h3}} \text{ and } T_{h2} \leq \frac{\pi}{W_{h2}} \ . \tag{2}$$

Then, we consider the special case of the circularly band-limited images that are common in practical imaging systems, and check the sampling requirements of the two sampling lattices in this special case. We define the circular band region

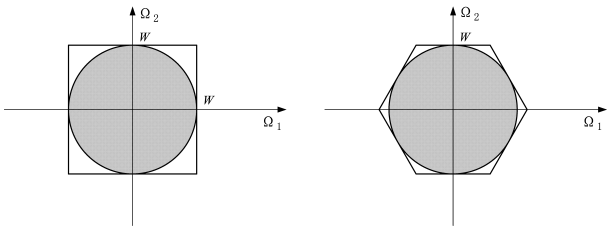


Fig. 4. Illustration of a circular band region embedded in a square region (left) and in a regular hexagonal region (right) (redrawn from [1]).

as $X_a(\Omega_1, \Omega_2) = 0,$ if $\Omega_1^2 + \Omega_2^2 \geq W^2$, where Ω_1 and Ω_2 are the horizontal and vertical frequencies and W is the radius. To avoid aliasing exactly, the band regions in Fig. 3(a) and Fig. 3(b) should be able to hold the circular band region exactly, as shown in Fig. 4. In this special case, the rectangular lattice becomes the square lattice, and the hexagonal lattice becomes a regular one. Accordingly, the sampling requirements of (1) and (2) become the following:

$$T_{r1} = \frac{\pi}{W} \text{ and } T_{r2} = \frac{\pi}{W} \tag{3}$$

for the square lattice and

$$T_{h1} = \frac{2\pi}{\sqrt{3}W} \text{ and } T_{h2} = \frac{\pi}{W} \tag{4}$$

for the regular hexagonal lattice.

III. Proposed method

For the sampling of the circularly band-limited 2-D image, as shown in (3) and (4), it is clear that the vertical intervals in both the square lattice and the regular hexagonal lattice have the same value π/W, but only the horizontal intervals are different for the square lattice and the regular hexagonal lattice. This signifies that it may be possible to perform the square to hexagonal lattice conversion only along the horizontal direction, and it will largely reduce the square to hexagonal lattice conversion. This is the main starting point of the proposed method. In the following, we will prove that, if we assume the 2-D interpolation filter as separable, this simplification is possible and theoretically correct.

To clearly introduce the method, we choose to use the reconstruction and resampling framework. That is, we define the sampling intervals for square lattices as T_{s1} and T_{s2} and the sampling intervals for regular hexagonal lattices as T_{h1} and T_{h2}. Next, we start with the square samples $x_s(n_1, n_2) = x_a(n_1 T_{s1}, n_2 T_{s2})$, reconstruct the analog image $x_r(t_1, t_2)$, and then resample to get the hexagonal samples $x_h(k_1, k_2) = x_r((k_1 - \frac{1}{2}k_2)T_{h1}, k_2 T_{h2})$.

We define the interpolation kernel $h_r(t_1, t_2)$ in the reconstruction as a separable one:

$$h_r(t_1, t_2) = h_{r1}(t_1) h_{r2}(t_2) \ . \tag{5}$$

Then, we can reconstruct the original analog image $x_r(t_1, t_2)$ through the 2-D interpolation:

$$\begin{aligned} & x_r(t_1, t_2) \\ &= \sum_{n_1} \sum_{n_2} x_s(n_1, n_2) h_r(t_1 - n_1 T_{s1}, t_2 - n_2 T_{s2}) \\ &= \sum_{n_1} \sum_{n_2} x_s(n_1, n_2) h_{r1}(t_1 - n_1 T_{s1}) h_{r2}(t_2 - n_2 T_{s2}) \\ &= \sum_{n_1} h_{r1}(t_1 - n_1 T_{s1}) \sum_{n_2} x_s(n_1, n_2) h_{r2}(t_2 - n_2 T_{s2}) \ . \end{aligned} \tag{6}$$

Finally, we resample the reconstructed analog image $x_r(t_1, t_2)$ using the hexagonal lattice:

$$\begin{aligned} x_h(k_1, k_2) &= x_r((k_1 - \frac{1}{2}k_2)T_{h1}, k_2 T_{h2}) \\ &= \sum_{n_1} h_{r1}((k_1 - \frac{1}{2}k_2)T_{h1} - n_1 T_{s1}) \\ &\quad \sum_{n_2} x_s(n_1, n_2) h_{r2}(k_2 T_{h2} - n_2 T_{s2}) \ . \end{aligned} \tag{7}$$

As shown in (3) and (4) before, we have $T_{h2} = T_{s2}$ in this case. Moreover, as a common interpolation function like $Sinc$, $h_{r2}(t_2)$ has an important property that $h_{r2}(0) = 1$ and $h_{r2}(kT_{s2}) = 0,$ when $k \in \mathbb{Z}$ and $k \neq 0$, and this results in $\sum_{n_2} x_s(n_1, n_2) h_{r2}(k_2 T_{h2} - n_2 T_{s2}) = x_s(n_1, k_2)$, i.e., the interpolation along the vertical direction has been eliminated. Then, we can obtain the following result:

$$\begin{aligned} & x_h(k_1, k_2) \\ &= \sum_{n_1} h_{r1}((k_1 - \frac{1}{2}k_2)T_{h1} - n_1 T_{s1}) x_s(n_1, k_2) \ . \end{aligned} \tag{8}$$

It's clear that the interpolation in (8) is indeed along the horizontal direction only, and the 2-D interpolation has been converted to a 1-D interpolation matter. In addition, in the hexagonal lattice array, the odd rows stagger horizontally from the even rows by half the horizontal interval, and this is also clearly shown in (8). Therefore, in the 1-D interpolation processing, we should treat this matter between the odd rows and the even rows.

For each convolution output, its computational costs are generally proportional to the convolution kernel size. Consider a 2-D interpolation kernel with size $N \times N$, the size will be reduced to $2N$ if the separability assumption is adopted, and the effective size can be further reduced to N in the proposed conversion since the vertical direction computation has been eliminated. Obviously, the proposed method simplifies the common 2-D interpolation conversion approach with a factor of N. Therefore, the proposed method is more efficient in both computational cost and the implementation.

IV. Results and Discussion

In this section, we implemented the proposed conversion method and performed experiments for two main goals, one is to verify the correctness of the proposed method, and the other is to demonstrate the computational efficiency against

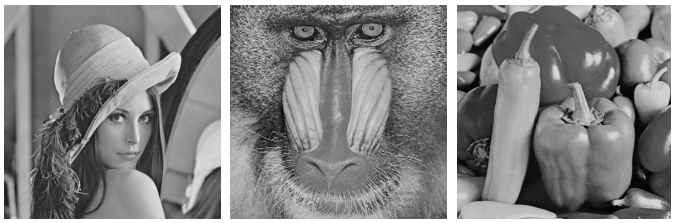


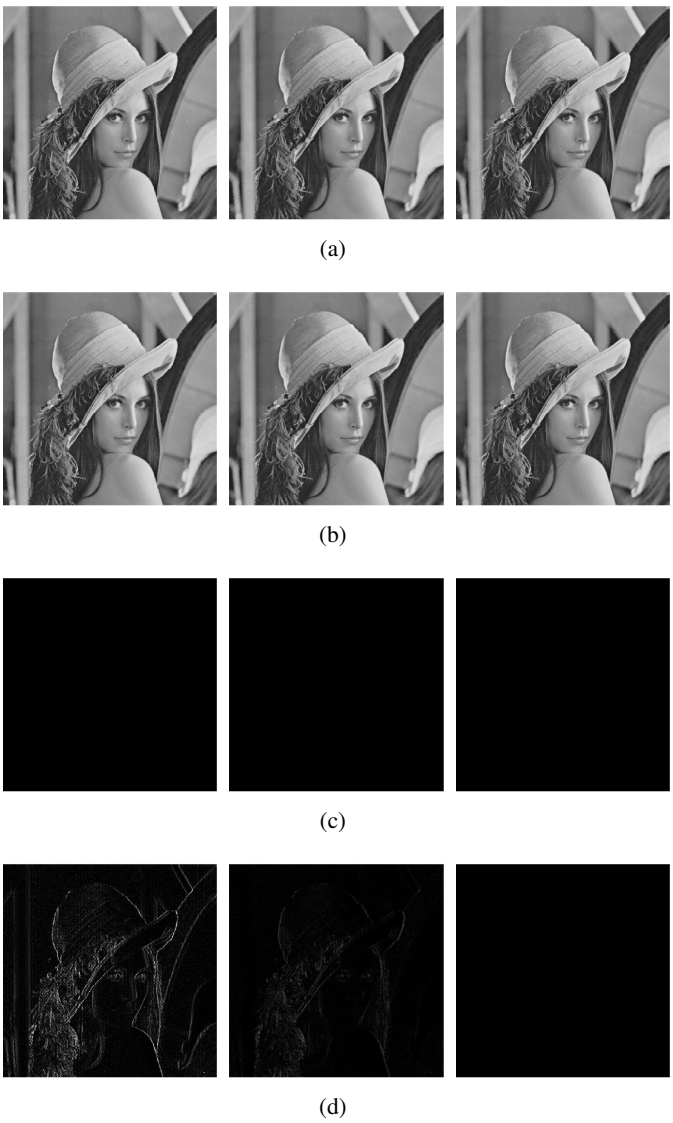
Fig. 5. Test images used to verify the correctness of the proposed method in the experiments: "Lena" (left), "Mandril" (middle), and "Peppers" (right).

the counterpart of the 2-D interpolation approach. Note that the proposed method is not a specific square to hexagonal lattices conversion method but a simplified version of the 2-D interpolation counterpart. In this section, we implemented three common interpolation kernels, i.e., nearest-neighbor, linear, and cubic spline.

Firstly, we performed the conversion on test images. To verify the correctness, we chose to compare the visual effects among the different methods. To do this, we simulated each hexagonal pixel with one hexagonal patch according to the positions of the sampling lattice, and displayed the whole hexagonally sampled data on the simulated hexagonal display. Three test images were used, including "Lena", "Mandril", and "Pepper", as shown in Fig. 5, and the conversion results are shown in (a) and (b) of Fig. 6, Fig. 7, and Fig. 8, respectively.

To verify the correctness, we directly compared between the two implementations with respect to a specific interpolation kernel, for example, the 1-D linear interpolation of the proposed method and the 2-D bilinear interpolation. These results are shown in the part (c) of the three figures, and we can find that difference output is completely dark, indicating the two results are the same and there is no difference. Therefore, these results can verify the correctness of the proposed method, i.e., the simplification of the common 2-D interpolation to 1-D interpolation. Then, we continued to compare the three common interpolation kernels. For each test image, we chose the result of the cubic spline kernel as the ground truth, and computed the differences between each result and the ground truth. The results are presented in part (d) of the three figures. It is clear that the nearest-neighbor kernel performs the worst, and the linear kernel performs better than the nearest-neighbor kernel; these results are consistent with the common conclusion.

Finally, to demonstrate the computational efficiency gained against the 2-D interpolation approach, we performed experiments on images with different sizes, and then compared the run times between the proposed method and the 2-D interpolation counterpart. In these experiments, a total of 16 color images were used, and results listed in Table 1, in which the unit of run times is in second ('s'). Note that the methods were implemented in Matlab, and the experiments were performed on a desktop computer with Intel(R) Core(TM) i7–3770 CPU @ 3.4 GHz and 16 GB RAM.



(a)

(b)

(c)

(d)

Fig. 6. Conversion results and the comparisons on the test image "Lena". (a) are the conversion results of the proposed method, from left to right, nearest-neightbor, linear, and cubic; (b) are the conversion results of the 2-D interpolation approaches, from left to right, nearest-neightbor, bilinear, and bicubic; (c) are the differences between (a) and (b), respectively; (d) are the difference (10 times) of (a) against the bicubic result of (b).

From Table 1, it is observed that the run times of the proposed methods are faster than those of the 2-D interpolation counterparts, particularly with higher resolution images. Furthermore, for the linear kernel with $N = 2$, the proposed method consumes less than 50% of the time when compared with the 2-D bilinear interpolation approach; for the cubic kernel with $N = 4$, the proposed method consumes approximately 27% of the time when compared with the 2-D bicubic interpolation approach. These time saving values are generally

Table 1. Comparisons of the run times (in seconds) between the proposed method and the 2-D interpolation counterpart.

Images	Input sizes	Output sizes	Proposed method			2-D interpolation		
			nearest	linear	cubic	nearest	bilinear	bicubic
1st	678×1024×3	678× 886×3	0.0356	0.0369	0.1675	0.0817	0.0880	0.5312
2nd	667×1000×3	667× 866×3	0.0375	0.0429	0.1480	0.0851	0.0899	0.5246
3rd	1080×1728×3	1080×1496×3	0.0856	0.1159	0.4292	0.2097	0.2753	1.5707
4th	1080×1920×3	1080×1662×3	0.0948	0.1124	0.4647	0.2318	0.2624	1.7377
5th	1200×1600×3	1200×1385×3	0.0848	0.1040	0.4773	0.2240	0.2438	1.5990
6th	1333×2000×3	1333×1732×3	0.1458	0.1606	0.6560	0.3314	0.3611	2.2568
7th	1335×2000×3	1335×1732×3	0.1466	0.1796	0.6724	0.3370	0.3765	2.3085
8th	1366×2048×3	1366×1773×3	0.1693	0.1909	0.7772	0.3748	0.4166	2.5484
9th	1836×2448×3	1836×2120×3	0.2136	0.2716	1.1717	0.5220	0.5933	4.1543
10th	1944×2592×3	1944×2244×3	0.2098	0.2600	1.3257	0.5633	0.6522	4.9262
11th	2048×2560×3	2048×2217×3	0.2776	0.3352	1.5083	0.7338	0.8065	5.0292
12th	2136×3216×3	2136×2785×3	0.3091	0.4456	1.8248	0.7714	0.8622	6.4165
13th	2115×3676×3	2115×3183×3	0.3596	0.4916	2.1329	0.9349	1.0919	7.5219
14th	2896×1944×3	2896×1683×3	0.2426	0.3276	1.4679	0.6422	0.7070	5.3262
15th	2828×4272×3	2828×3699×3	0.5073	0.6973	3.4205	1.4217	1.6410	12.4434
16th	3280×4928×3	3280×4267×3	0.7691	0.9518	4.6304	2.0932	2.3662	16.8526

consistent with the theoretical analysis completed in Section III.

V. Conclusion

In this paper, we have presented a square to hexagonal lattice conversion method, which performs 1-D interpolation along the horizontal direction only. Compared with the original 2-D interpolation approach, the proposed method is more simple and computationally efficient than the existing approaches, and experimental results have verified the correctness as well as the efficiency. In short, the proposed method simplifies the common 2-D interpolation approach to 1-D interpolation only, presenting an approach that would be suitable to utilize the common parallel computation resource. Therefore, the proposed method is valuable for practical hexagonal image processing. Furthermore, the method can be easily modified to perform conversions along the vertical direction, maintaining the same accuracy and computational gain presented in this paper.

References

[1] R. Mersereau, "The processing of hexagonally sampled two-dimensional signals," *Proceedings of the IEEE*, vol. 67, no. 6, pp. 930–949, 1979.

[2] K. J. Vanhoutte, K. F. Michielsen, and D. G. Stavenga, "Analyzing the reflections from single ommatidia in the butterfly compound eye with voronoi diagrams," *Journal of Neuroscience Methods*, vol. 131, no. 1–2, pp. 195–203, 2003.

[3] C. A. Curcio, K. R. Sloan, R. E. Kalina, and A. E. Hendrickson, "Human photoreceptor topography," *The Journal of Comparative Neurology*, vol. 292, no. 4, pp. 497–523, 1990.

[4] K. Mostafa, J. Y. Chiang, and I. Her, "Edge-detection method using binary morphology on hexagonal images," *The Imaging Science Journal*, vol. 63, no. 3, pp. 168–173, 2015.

[5] B. Gardiner, S. Coleman, and B. Scotney, "Multiscale Edge Detection using a Finite Element Framework for Hexagonal Pixel-based Images," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1849 – 1861, 2016.

[6] T. Shima, S. Sugimoto, and M. Okutomi, "Comparison of image alignment on hexagonal and square lattices," in *2010 IEEE International Conference on Image Processing*, 2010, pp. 141–144.

[7] V. Argyriou, "Sub-hexagonal phase correlation for motion estimation." *IEEE transactions on image processing*, vol. 20, no. 1, pp. 110–120, 2011.

[8] E. Linner, R. Strand, and E. Linn, "Comparison of restoration quality on square and hexagonal grids using normalized convolution," in *21st International Conference on Pattern Recognition (ICPR'12)*, 2012, pp. 3046–3049.

[9] S. Veni and K. A. Narayanankutty, "Vision-based hexagonal image processing using Hex-Gabor," *Signal, Image and Video Processing*, vol. 8, no. 2, pp. 317–326, 2014.

[10] S. H. Contreras-Ortiz and M. D. Fox, "Hexagonal filters for ultrasound images," *Journal of Electronic Imaging*, vol. 23, no. 4, p. 043022, 2014.

[11] A. Entezari, M. Nilchian, and M. Unser, "A Box Spline Calculus for the Discretization of Computed Tomography Reconstruction Problems," *IEEE Transactions on Medical Imaging*, vol. 31, no. 8, pp. 1532–1541, aug 2012.

[12] J. F. de Andrade, M. L. R. de Campos, and J. A. Apolinario, "L1-Constrained Normalized LMS Algorithms for Adaptive Beamforming," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6524–6539, dec 2015.

[13] J. Sivaswamy, "Framework for practical hexagonal-image processing," *Journal of Electronic Imaging*, vol. 11, no. 1, pp. 104–114, 2002.

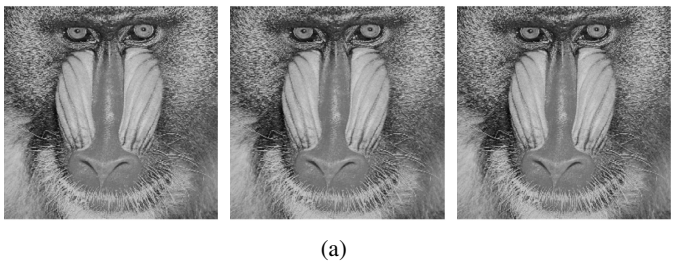
[14] J. Serra and B. Laÿ, "Square to hexagonal lattices conversion," *Signal Processing*, vol. 9, no. 1, pp. 1–13, 1985.

[15] X. He, J. Li, and T. Hintz, "Comparison of Image Conversions Between Square Structure and Hexagonal Structure," in *9th international conference on Advanced concepts for intelligent vision systems*, 2007, pp. 262–273.

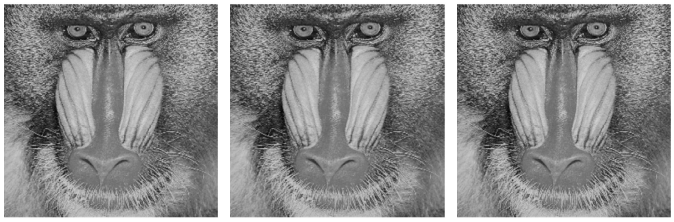
[16] D. Van De Ville, W. Philips, and I. Lemahieu, "Least-squares spline resampling to a hexagonal lattice," *Signal Processing: Image Communication*, vol. 17, no. 5, pp. 393–408, 2002.

[17] L. Condat, D. Van De Ville, and B. Forster-Heinlein, "Reversible, fast, and high-quality grid conversions." *IEEE transactions on image processing*, vol. 17, no. 5, pp. 679–693, 2008.

[18] B. Gardiner, S. Coleman, and B. Scotney, "Comparing Hexagonal Image Resampling Techniques with Respect to Feature Extraction," in *14th International Machine Vision and Image Processing Conference*, 2011.

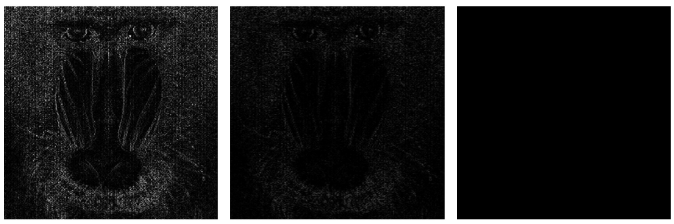


(a)



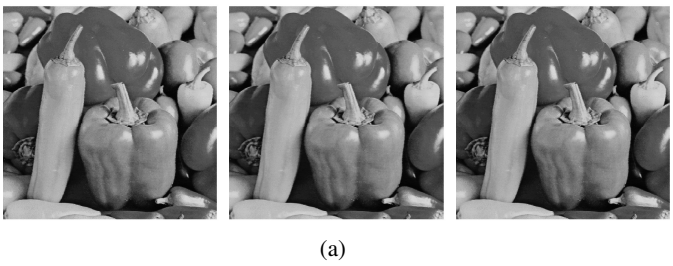
(b)

(c)



(d)

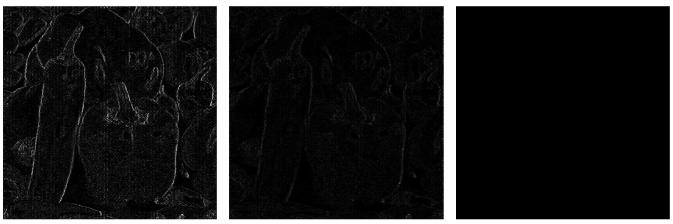
Fig. 7. Conversion results and the comparisons on the test image "Mandril". (a) are the conversion results of the proposed method, from left to right, nearest-neightbor, linear, and cubic; (b) are the conversion results of the 2-D interpolation approaches, from left to right, nearest-neightbor, bilinear, and bicubic; (c) are the differences between (a) and (b), respectively; (d) are the difference (10 times) of (a) against the bicubic result of (b).



(a)

(b)

(c)



(d)

Fig. 8. Conversion results and the comparisons on the test image "Peppers". (a) are the conversion results of the proposed method, from left to right, nearest-neightbor, linear, and cubic; (b) are the conversion results of the 2-D interpolation approaches, from left to right, nearest-neightbor, bilinear, and bicubic; (c) are the differences between (a) and (b), respectively; (d) are the difference (10 times) of (a) against the bicubic result of (b).