Uniformly Partitioning Images on Virtual Hexagonal Structure

Xiangjian He[1,2], Huaqing Wang[1], Namho Hur[2], Wenjing Jia[1], Qiang Wu[1], Jinwoong Kim[2], and Tom Hintz[1]

[1]Department of Computer Systems
Faculty of Information Technology
University of Technology, Sydney
{sean, huwang, wejia, wuq, hintz}@it.uts.edu.au

[2]Broadcasting System Research Group
Digital Broadcasting Research Division
Electronics Telecommunications Research Institute
161 Gajeong-dong, Yuseong-gu, Daejeon, 305-700, Korea
namho@etri.re.kr

Abstract – *Hexagonal structure is different from the traditional square structure for image representation. The geometrical arrangement of pixels on hexagonal structure can be described in terms of a hexagonal grid. Uniformly separating image into seven similar copies with a smaller scale has commonly been used for parallel and accurate image processing on hexagonal structure. However, all the existing hardware for capturing image and for displaying image are produced based on square architecture. It has become a serious problem affecting the advanced research based on hexagonal structure. Furthermore, the current techniques used for uniform separation of images on hexagonal structure do not coincide with the rectangular shape of images. This has been an obstacle in the use of hexagonal structure for image processing. In this paper, we briefly review a newly developed virtual hexagonal structure that is scalable. Based on this virtual structure, algorithms for uniform image separation are presented. The virtual hexagonal structure retains image resolution during the process of image separation, and does not introduce distortion. Furthermore, images can be smoothly and easily transferred between the traditional square structure and the hexagonal structure while the image shape is kept in rectangle.*

Keywords: Hexagonal structure, image scaling, Spiral Architecture, parallel processing, image partitioning

1 Introduction

Computer Vision and Image Processing is a computationally expensive field in which many operations require massive computations, especially when large data sets are involved such as those for stereo image matching and feature extraction. Parallel processing using a cluster consisting of multiple computers is a straightforward method to speed up image processing. Cluster-based parallel computing has the advantages of low cost and high utility. On the other hand, it has the disadvantages of high communication latency and irregular load patterns on the computing nodes [1]. Its performance mainly depends on the amount of and the structure for communications between processing nodes. Therefore, image separation or partitioning for task dividing is critical in a parallel algorithm for image processing.

Many types of image partitioning have been proposed [2] on traditional square image structure such as Row Partition, Column Partition and Block Partition. All of these partition methods do not intend to separate an image into similar copies (or sub-images). Hence, when the sub-images are assigned to various computer nodes for parallel processing, the load-balancing is not guaranteed.

The method for image partitioning presented in [3] is based on a hexagonal image structure, called Spiral Architecture (SA) [4], which is inspired from anatomical considerations of the primate's vision. It partitions the input image uniformly into a number of sub-images as required based

on an operation, called Spiral Multiplication defined on SA. Each sub-image is a near (or similar) copy of the input image with a smaller scale, while all of the sub-images are mutually exclusive and the original image information is all kept in the sub-images. Every sub-image can be processed independently and in parallel by an individual node without data exchange between the nodes. Furthermore, the workload on every node is almost the same as that on any other node because of the uniform partition. Consequently, the computational complexity is greatly reduced and the processing time is significantly shortened. However, our previous work shown in [3] has the following shortcomings. First of all, the shape of input images is hexagon-like and does not coincide with rectangular shape of traditional images. Secondly, the image partition method uses on the Spiral Multiplication that is computationally expensive. Thirdly, images are represented on a mimic Spiral Architecture that introduces a loss on image resolution and creates image distortion when the sub-images have a rotated angle from the original image.

In order to take the advantages and suppress the disadvantages of SA, in this paper, we propose a novel method for uniformly separating image into four similar sub-images based on a newly defined virtual hexagonal image structure [5]. The hexagonal structure can be smoothly converted from the square structure without changing the image shape. In this virtual structure, hexagonal pixels can be easily located through simple computations. It avoids the necessity of using a Spiral Multiplication to compute pixel locations and hence avoids the need to build a large table to record the location information. This virtual structure hardly changes the image resolution and almost does not introduce image distortion.

The rest of this paper is organized as follows. In Section 2, we briefly review the Spiral Architecture. In Section 3, we introduce the construction of a new virtual hexagonal structure. A new technique for uniform image partition is presented in Section 4. Experimental results are demonstrated in Section 5. We conclude in Section 6.

2 Spiral Architecture

On Spiral Architecture, an image is represented as a collection of hexagonal pixels. Each pixel has only six neighbouring pixels with the same distance to it. Each pixel is identified by a number of base 7 called a *spiral address*. The numbered (or addressed) hexagons form the cluster of size 7^n, where n is a positive integer. These hexagons starting from address 0 towards address 7^n tile the plane in a recursive modular manner along a spiral-like curve. As an example, a cluster with size of 7^2 and the corresponding spiral addresses are shown in Figure 1. The image space formed on the Spiral Architecture always has a hexagon-like shape. This shortcoming restricts the applications of image processing on hexagonal structures. Our approach in this paper will overcome this disadvantage.

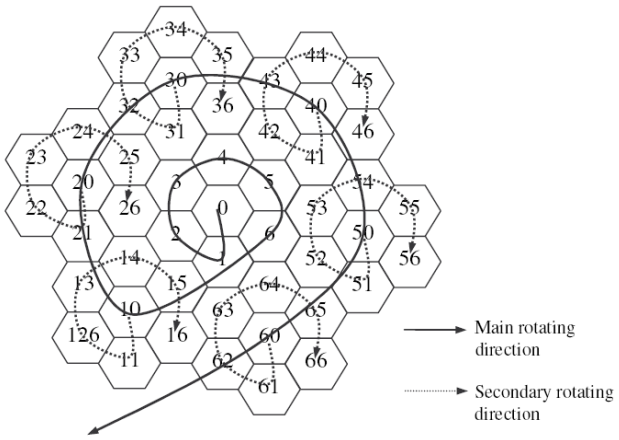

Figure 1. Spiral Architecture with spiral addressing

Two algebraic, useful and important operations have been defined on Spiral Architecture based on spiral addresses. They are *Spiral Addition* and *Spiral Multiplication*. These two operations correspond to two transformations on Spiral Architecture, which are translation and rotation with a scaling. Spiral Multiplication is also often applied to uniformly separate images on Spiral Architecture for parallel processing. In this paper, we perform an algorithm for image partition on a hexagonal structure without using computationally expensive Spiral Multiplication. Our algorithm will maintain the important property that none of image (intensity) information is lost during the separation process.

3 Virtual Hexagonal Structure

In this section, we review the construction of a new virtual hexagonal structure [5].

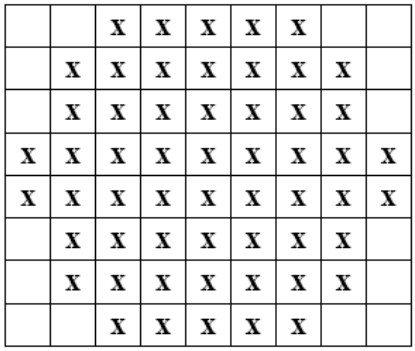

Figure 2. The structure of a single hexagonal pixel

To construct hexagonal pixels, each square pixel is first separated into 7×7 smaller pixels, called sub-pixels. To be simple, the light intensity for each of these sub-pixels is set

to be the same as that of the pixel from which the sub-pixels are separated. Each virtual hexagonal pixel is formed by 56 sub-pixels arranged as shown in Figure 2. To be simple, the light intensity of each constructed hexagonal pixel is computed as the average of the intensities of the 56 sub-pixels forming the hexagonal pixel when necessary.

Figure 3 shows a collection of seven hexagonal pixels constructed with spiral addresses from 0 to 6. From Figure 3, it is easy to see that the hexagonal pixels constructed in this way tile the whole plane without and spaces and overlaps.

From Figure 3, it can be easily computed that the distance from pixel 0 to pixel 1 or pixel 4 is 8. The distance from pixel 0 to pixel 2, pixel 3, pixel 5 or pixel 6 is

$$\sqrt{7^2 + 4^2} = 8.06.$$

which is close to 8. Hence, the feature of equal distance is almost retained and hence this construction hardly introduces image distortion.

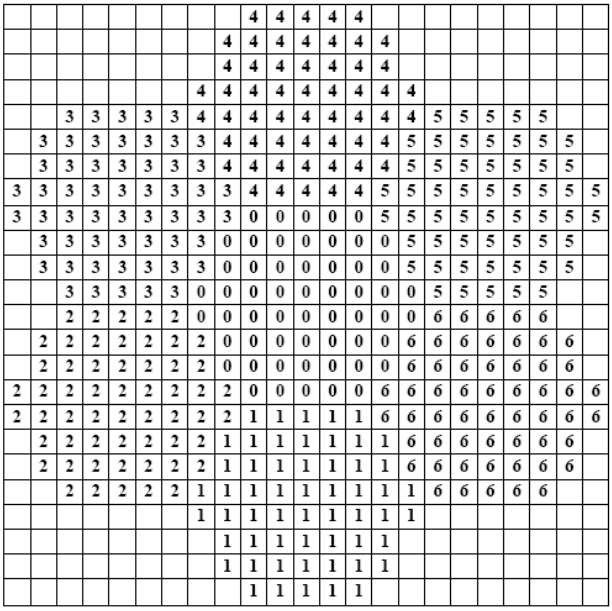

Figure 3. A cluster of seven hexagonal pixels

Through this paper, we assume that original images are represented on a square structure arranged as 16*M* rows and 16*N* columns, where *M* and *N* are two positive integers. This assumption guarantees that sub-images to be obtained in this paper has exactly the same size and are of a complete rectangle shape. $M = 0$ corresponds to the first (top) row and $N = 0$ corresponds to the first (left) column. Let the centre of the virtual hexagonal structure be located at the middle of 8*M*-th row and (8*M* +1)-th row, and at 8*N*-th column. Let us denote the light intensity of the square pixel at row m and column n by $Q(m,n)$, where $0 \le m < 16M$ and $0 \le n < 16N$. Here, we use row 0 to represent the 1st row and column 0 to represent the 1st column and so forth.

Note that there are ($7 \times 16M =$) $112M$ rows and ($7 \times 16N =$) $112N$ columns in the virtual square structure consisting of virtual sub-pixels obtained from the original square pixels. Let us construct the first hexagonal pixel using the 56 sub-pixels with centre located in the middle of rows 56*M*-1 and 56*M* and at column 56*N*-1 of the virtual square structure. This first pixel is called the central hexagonal pixel in the hexagonal structure. It is corresponding to the pixel with spiral address 0 in the Spiral Architecture. After the 56 sub-pixels for the first hexagonal pixel are allocated, all sub-pixels for all hexagonal pixels can be assigned. For our uniform partition algorithms, the assignment of sub-pixels to corresponding hexagonal pixels is not explicitly required. We do not need to compute the intensities for the virtual hexagonal pixels. After image processing for uniform partition, the intensity of each square pixel can be computed as the average of the light intensities of the 7×7 sub-pixels separated from this square pixel.

4 Uniform Image Separation

In this section, we will perform algorithms for separating image represented on the virtual hexagonal structure into four sub-images that look similar and have exact the same image size. The idea is to define rows and columns as images represented on the square structure. Then, each sub-image is a collection of hexagonal pixels from every second row and column. It is not as obvious as in the square structure to find the column and the row of each hexagonal pixel. It is not obvious either to know which pixels form the first row or the first column of a sub-image. We follow the following three steps for the uniform image separation. We first define the rows and columns on the virtual hexagonal structure, and propose an algorithm for computing the row and column of the hexagonal pixel that a given sub-pixel belongs to. Then we present an algorithm for the extraction of the first sub-image. In the third step, algorithms for construction of the second, the third and the fourth sub-images are proposed. After the process for image separation, to be simple, the sub-images represented on hexagonal structure can be mapped back to the square structure using the pseudo code below.

```
for (int m = 0 ; m<16*M ; m++) {
  for (int n=0 ; n<16*N, j++) {
    int k=0 ;
    Q[m][n]=0 ;
    for (i=0 ; i<7 ; i++)
      for (j=0 ; j<7 ; j++) {
        k=k+1 ;
        Q[m][n]=(Q[m][n]+P[7*m+i][7*n+j]) ;
      }
    }
```

```
        Q[m][n]=Q[m][n]/k ;
    }
}
```

Note that, for better display result, a better method for assignments of intensity values of sub-pixels is required through a better interpolation technique. This is, however, beyond the discussion and the objectives of this paper.

4.1 Pixel row and column

Assume a given sub-pixel is at row p and column q. Let R and C represent the number of rows and number of columns needed to move from the central hexagonal pixel to the hexagonal pixel containing the given sub-pixel taking into account the moving direction corresponding to the signs of R and C. Here, pixels on the same column are on the same vertical line. For example, as shown in Figure 4, pixels with addresses 43, 42, 5, 6, 64, 60 and 61 are on the same column with $C = 1$. The row with $R = 0$ consists of the pixels on the horizontal line passing the central pixel and on the columns with even C values, and the pixels on the horizontal line passing the pixel with address 3 and on the columns with odd C values. Other rows are formed in the same way. For example, pixels with addresses 21, 14, 2, 1, 6, 52, 50 and 56 are on the same row with $R = 1$. Figure 5 show rows in a hexagonal structure consisting of 49 hexagons. Following the algorithm proposed in [6], the C and R corresponding to the given sub-pixel can be computed from P_1, P_2, Q_1 and Q_2 defined below.

$$Q_1 = \mathrm{sgn}\{q - 56N + 1\} \max\{\mathrm{int}\, c \mid c \le \frac{|q - 56N + 1|}{7}\},$$

$$Q_2 = (q - 56N + 1) \bmod 7,$$

$$P_1 = \mathrm{sgn}\{p - 56M + 1\} \max\{\mathrm{int}\, c \mid c \le \frac{|p - 56M + 1|}{8}\},$$

$$P_2 = (p - 56M + 1) \bmod 8.$$

4.2 Construct the first sub-image

Let h be an arbitrary given hexagonal pixel on a hexagonal structure. Let us denote the values of C and R corresponding to h by C_h and R_h respectively. Then the first sub-image S_1 is formed by the hexagonal pixels that satisfy the following conditions.

$$S_1 = \{h \mid C_h \bmod 2 = 0, (\frac{C_h}{2} \bmod 2) - (R_h \bmod 2) = 0\}.$$

From the above representation, we can see that any hexagonal pixel must be on the column with even C value; and if the pixel falls onto column with even $C/2$ value then R value for the pixel must be even as well, otherwise R must be odd.

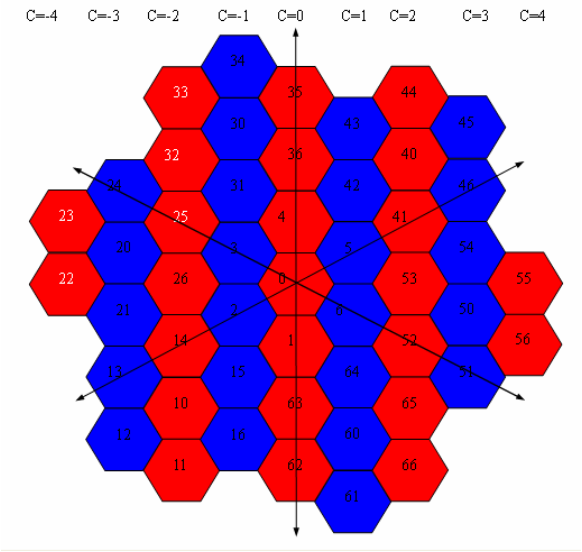

Figure 4. Columns on a hexagonal structure

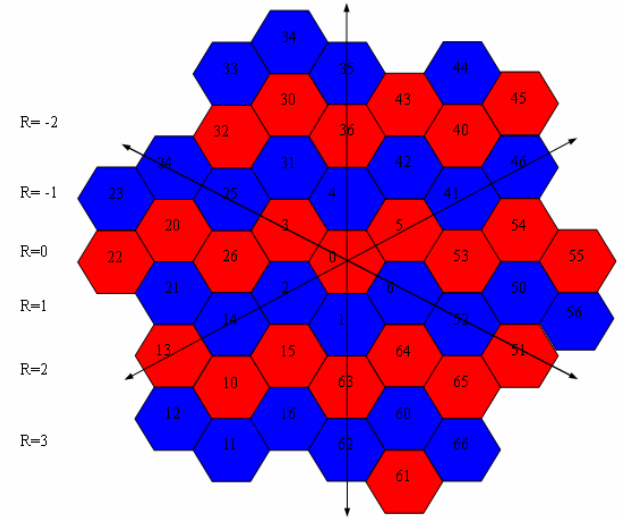

Figure 5. Rows on a hexagonal structure

The arrangement of the hexagonal pixels in S_1 is made as follows. We keep the location of the central pixel (with $C = 0$ and $R = 0$) unchanged, and move any other hexagonal pixel towards the central pixel by its half distance from it. Note that the distance between two adjacent columns is 7 sub-pixels long, and the distance between two adjacent rows is 8 sub-pixels long. Hence, any pixel belonging to S_1 except the central pixel will move by $|C_h|/2*7$ sub-pixels leftwards (or rightwards when C_h is negative) then by $|R_h|/2*8$ upwards (or downwards when R_h is negative).

After the above-mentioned procedure, the first sub-image is formed and it is sitting in the middle of the original image area. In order to fit all four sub-images onto the same image area after the four sub-images are formed, in the next step,

we move S_1 to the left up corner of the original image area such that S_1 will exactly occupy the top 8M rows and left 8N columns of the original rectangular image area. Note that the new centre of S_1 will be located in the middle of rows 28M-1 and 28M and at column 28N-1 of the virtual square structure. Hence, to move the S_1 to the left up corner, we need only move all sub-pixels in S_1 upwards by 28M sub-pixels and leftwards by 28N sub-pixels.

4.3 Construct other sub-images

The second sub-image S_2 consists of all hexagonal pixels that are one hexagonal above or below the pixels in S_1. The third sub-image S_3 consists of all hexagonal pixels that are next to the pixels in S_1 but sitting at the left-bottom side. Similarly, the fourth sub-image S_4 consists of all hexagonal pixels that are next to the pixels in S_1 but sitting at the right-bottom side. Let us now construct sub-images 2 trough 4 one after another in the following.

4.3.1 Construction of S_2

To construct S_2, we first translate all sub-pixels downwards by 8 sub-pixels. By doing so, the first hexagonal pixel (with spiral address 4 as shown in Figure 1) of S_2 that is the one right above the central pixel with spiral address 0 is moved to the central position with C and R both equal to 0. After the translation, the bottom 8 rows in the virtual square structure will be moved out from the original image area. In order not to lose any image information after the translation, we fill the top 8 rows of the virtual square structure by those sub-pixels at the bottom 8 rows while performing the translation. The pseudo code for this movement is shown as follows.

The pseudo code for the translation is described as follows.

```
for (i = 0 ; i<M1 ; i++) {
  for (j = 0 ; j<N1 ; j++) {
    Temp[i][j]=P[i][j] ;
    P[i][j]=255 ;
  }
}
for (i = 0 ; i<M1 ; i++) {
  for (j = 0 ; j<N1 ; j++) {
    if (0<= i <8)
      P[i][j]=P[M1-8+i][j] ;
    else
      P[i][j]=P[i-8][j] ;
  }
}
```

We can now use exactly the same algorithm as shown in Subsection 4.2 to construct S_2 and display it in the middle of the original image area. The next step then is to move S_2 to the right up corner of the original image area such that S_2 will exactly occupy the top 8M rows and right 8N columns of the original rectangular image area.

4.3.2 Construction of S_3

To construct S_3, we first translate all sub-pixels rightwards by 7 sub-pixels, and then upwards by 4 sub-pixels. By doing so, the first hexagonal pixel (with spiral address 2 as shown in Figure 1) of S_3 is moved to the central position. Again, in order not to lose any image information after the translation, the sub-pixels that will be moved out from the image area must be filled into the pixels that will not be translated from any other sub-pixels. The pseudo code for this movement is shown as follows.

```
for (i = 0 ; i<M1 ; i++) {
  for (j = 0 ; j<N1 ; j++) {
    Temp[i][j]=P[i][j] ;
    P[i][j]=255 ;
  }
}
for (i = 0 ; i<M1 ; i++) {
  for (j = 0 ; j<N1 ; j++) {
    if (M1-4<= i <M1) {
      if (j < 7)
        P[i][j]=P[i-M1+4][N1-7+j] ;
      else
        P[i][j]=P[i-M1+4][j-7] ;
    }
    else {
      if (j<7)
        P[i][j]=P[i+4][N1-7+j] ;
      else
        P[i][j]=P[i+4][j-7] ;
    }
  }
}
```

We can now use exactly the same algorithm as shown in Subsection 4.2 to construct S_3 and display it in the middle of the original image area. The next step then is to move S_3 to the left bottom corner of the original image area such that S_3 will exactly occupy the bottom 8M rows and left 8N columns of the original rectangular image area.

4.3.3 Construction of S_4

The construction of S_4 is almost the same as the construction of S_3. We first translate all sub-pixels leftwards by 7 sub-pixels, and then upwards by 4 sub-pixels. By doing so, the first hexagonal pixel (with spiral address 6 as shown in Figure 1) of S_4 is moved to the central position.

We can then use exactly the same algorithm as shown in Subsection 4.2 to construct S_4 and display it in the middle of the original image area. The next step then is to move S_4 to

the right bottom corner of the original image area such that S_4 will exactly occupy the bottom $8M$ rows and right $8N$ columns of the original rectangular image area.

5 Experimental Results

The above algorithms for uniform image separation on the newly developed virtual hexagonal structure are implemented using C++ programming language and tested on a desktop of Pentium IV, 2.8GHz CPU and 480MB memory. Experimental results of the proposed image partitioning algorithms on grey-level images are presented here.

A sample image, called “building” with size of 384*384 is shown in Figure 6. An example of image separation into four sub-images on the virtual hexagonal structure is shown in Figure 7.

Figure 6. A sample image, “building”

In Figure 7, it can be seen clearly that four sub-images have exactly the same size, look identical and are still in the shape of a square. There are no gaps between the sub-images and no overlapping among the sub-images. All information of the original image is maintained during the whole separation process based on the hexagonal structure. The resolution of the whole image represented on the virtual hexagonal image structure and shown in Figure 7 is almost the same as that for Figure 6.

As an illustration, the results of a further partitioning process are displayed in Figure 8 that contained 16 similar sub-images.

Figure 7. The “building” image is uniformly separated into four sub-images of equal size and represented on a virtual hexagonal structure

Figure 8. 16 similar sub-images separated from the “building”

Note that the row and column numbers of the original image do not have to be the same for image separation. Figure 9 shows four sub-image images partitioned from a 512*384 image.

The total time to complete the computation for image separation for an image is less than 1 second. Compared with the method introduced in [7] that takes minutes to complete a rotation, a great improvement has been achieved using this new virtual structure.

Figure 9. Four similar sub-images separated from a 512*384 a car image

6 Conclusions and Discussion

In this paper, we have developed algorithms to uniformly separate an image represented on hexagonal structure into four sub-images. The partitioned sub-images look similar. It is important to know that the arrangement of the pixels in the sub-images does not change their symmetry relationship in the original image. For example, if in the original image a pixel *A* was in one of the 6 neighbouring directions as shown in Figures 4 and 5 corresponding to another pixel *B* belonging to the same sub-image, pixel *A* is in exactly the same direction corresponding to pixel *B* in their sub-image. Another example is that if three pixels *C*, *D* and *E* belong to the same sub-image pixel set and if the distance between *C* and *D* was the same as the distance between *C* and *E* in the original image, then in their sub-image, these two distances are still the same. Furthermore, we do not split any hexagonal pixel during the whole process, and hence all pixel information including the intensity values are maintained. The importance of uniform partition has been demonstrated in many papers addressing parallel image processing on hexagonal image structure. The maintenance of the symmetry property after image separation is critical for image processing such as finding gradient and its magnitude that request neighbouring information for the computations.

One may argue that on the traditional square image structure, we can also separate image uniformly into four parts by simply split every four pixels evenly into four sub-images. However, the sub-images obtained in this way also split the virtual hexagonal pixels and hence no longer hold the features of hexagonal structure. Therefore, those algorithms and applications developed based on hexagonal structure and proved to be more accurate or faster have lost their groundwork to be applied to these sub-images.

In this paper, we have also improved our translation algorithms proposed in [6] by placing the pixels moved out from the image area onto the empty region of the image area. This keeps all image information after image translation. The translation process and the separation process are both reversible. The original image can be fully reconstructed from the separated sub-images using inverse algorithms if needed.

Another contribution in this paper is a proposal to scale down an image to four times smaller images based on the hexagonal structure. Scaling operation is with the operations for image separation.

In our implementation, we implicitly adopt the ideas of two operations defined on Spiral Architecture, namely spiral addition and spiral multiplication, and use them for image separation. However, we do not perform these two operations to avoid a large amount of time requested for the complex computations on the virtual structure. This is very different from any of previous approaches, and has significantly improved the performance in terms of speed and complexity.

Another advantage of using this new approach is that the image shape is consistent with the traditional shape of a rectangle. This make it easier and more flexible for image processing based on hexagonal structure using images captured and displayed on square structure.

As we do not compute the light intensities for virtual hexagons, image resolution is maintained during the separation process, and we save the processing time and memory storage.

As there are simple non-overlapping mappings between the sub-pixels and the square pixels, and the mappings between the sub-pixels and the hexagonal pixels, the results of image processing on the hexagonal structure can be easily mapped back to the square structure for display.

We list other future work related to this paper as follows.

A better approximation of light intensity for sub-pixels is needed to reduce the loss during the image conversion. This is to be done using a more accurate light interpolation technique.

On Spiral Architecture, we have performed image separation into any given number of sub-images. This work

can be extended to the virtual hexagonal structure discussed in this paper.

References

[1] J. M. Squyres, A. Lumsdaine, and R. L. Stevenson, *A Cluster-based Parallel Image Processing Toolkit*, Proceedings of IS&T Conference on Image and Video Processing, San Joes, CA, pp. 228-239, 1995.

[2] C. H. Koelbel, D. B. Loveman, R. S. Schreiber, G. L. S. Jr., and M. E. Zosel, *The High Performance Fortran Handbook*: MIT Press, Cambridge, MA., 1994.

[3] Qiang Wu, Xiangjian He, Tom Hintz and Yuhuang Ye, *Complete Image Partitioning on Spiral Architecture*, in Lecture Notes in Computer Science, Vol. 2745, M.Guo and L.T.Yang (eds.), pp.304-315, 2003, Springer.

[4] P. Sheridan, T. Hintz, and D. Alexander, "Pseudo-invariant Image Transformations on a Hexagonal Lattice," *Image and Vision Computing*, vol. 18, pp. 907-917, 2000.

[5] X. He, T. Hintz, Q. Wu, H. Wang and W. Jia, *A new simulation of Spiral Architecture*, Proc. of International Conference on Image Processing, Computer Vision, and Pattern Recognition, Las Vegas, June 2006, to appear.

[6] X. He, W. Jia, Q. Wu, N. Hur, T. Hintz, H. Wang and J. Kim, *Basic Transformations on Virtual Hexagonal Structure*, International Conference on Computer Graphics, Imaging and Visualization, Sydney, July 2006, to appear.

[7] Q. Wu,, X. He, and T. Hintz, *Virtual Spiral Architecture.* Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, 2004. **1**: p. 399-405.