**CHAPTER 3**

**Theoretical Framework**

This chapter will discuss the basic concepts and framework that will be used In implementing Hexagonal Image Processing such as Resampling, Interpolation, Addressing and Storage, Display of hexagonal Image in a screen and Edge detection.

**3.1 Hexagonal Image Processing using Hexagonal Image Processing Framework**

The current scenario in image processing as what discussed in the previous sections of this study is only applicable for the square-image. To fit the hexagonal-image to the current scene, two ways can be implored. One, is to develop a full hexagonal image processing approach and use various known solutions for image acquisition and visualization. Two, is to develop a mixed approach where one can utilize the advantages of hexagonal images while the remaining parts will be done using square image. The general methodology that will be used in this study for hexagonal image processing will based from Middleton and Sivaswamy’s monograph “Hexagonal Image Processing: A Practical Approach” where they made a Hexagonal Image Processing framework implemented in Python on 2005.

The stated solutions can only be achieved by converting square image to hexagonal and vice versa. But for this study we will use a mixed approach in order to use current image processing techniques for square images.

**3.2 Image Resampling**

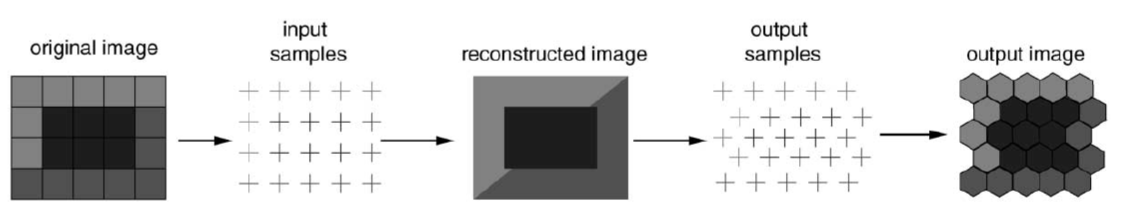
The process on which an image is converted from one lattice to another is called resampling. There are two types of resampling: One is resampling an image from square to hexagonal while the other is from hexagonal to square Image. In this study, we will use both resampling methods for we are using a mixed approach. After the resampling, the resulting hexagonal image could be a True Hexagonal Lattice which will have a regular hexagon shaped pixels or Irregular Hexagonal Lattice which will not be a regular hexagon.

**3.2.1 Square-to-hexagonal-image Conversion**

Let *f* (*x*) represents an image where *x =* (*x1 , x2*) is a spatial variable. To generate a sampled image, *fs*(*x*), an appropriate sampling kernel *h*(*x*) should be used. This process can be written as :

*fs* (*x1, x2*) = , *x*2 *– k*2)

However, in practice the commonly available sampled image is obtained using square lattice, *fss*(*x*) (Wolberg, 1990). From this *fss*(*x*), one can generate a hexagonal image thru resampling and yields a reconstructed image calculated implicitly within the resampling scheme shown below.



**Figure 3.1** Hexagonal sampling

There are problems that can arise in image resampling but the main one is determining the locations of input and output samples relative to the reconstructed image. These are the points in the square and hexagonal lattices, respectively.

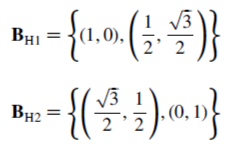
A lattice or commonly known as grid is defined as the set of points which are the centers of the periodic tilings of the plane. Let **B** = {*b1,b2*} be a set of basis vectors for the plane. The set **B** will define a lattice defined by:



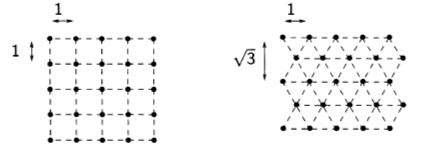
Different basis vector sets will also lead to different lattices that’s why many ways in generating a lattice can be used.



However, in this study we will use two convenient ways in generating a Hexagonal lattice, these are:



Both of the above sets are related by a simple rotation but the latter will be used to generate the desired lattice due to the issue of displaying the resulting data. Also, in examining the relationship between **B**H1 and **B**S it is noted that only *b2* is different which means that horizontal spacing in the two lattices square and hexagonal respectively is the same which is shown below.



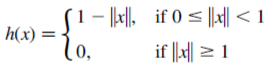
**Figure 3.2** ss

In both **B**H1 and **B**H2 the individual basis vectors are dependent on each other. This will result in redundancy. Take for example, in **B**H1 the second basis vector depends on the first. If an image is re-sampled from an *n* x *n* square-image then the choice of **B**H1 will result in one of the two possibilities:

1. Fixed horizontal lattice spacing
2. Fixed vertical lattice spacing

If the first possibility will occur to give *n* points, it will yield 15% extra vertical lattice points due to its closer packing (due to second basis vector). If a naïve approach will be utilized to display, then this would result to an elongated image. On the other hand, if the second possibility will occur to give *n* points then it will result to an inappropriate aspect ratio. Although this problem can be lessened through taking advantage of the display device’s geometry shown by Her et.al in their study “Resampling on a Pseudohexagonal Grid”. The best option is the First possibility where we will take measure to avoid elongation effect.

The last issue relating to the resampling of an image is the sampling kernel to be used. A very good review about kernel is done by Wolberg on 1990 in his research about “Digital Image Warping” and another review about the choice of kernel for hexagonal lattice is done by Her in the same research mentioned above (Resampling on a Pseudohexagonal Grid” where we observed that the effective kernel for most application is the bi-linear kernel shown below.



Where *x* is as defined above and || || is the standard Euclidean norm.

**3.2.1 Conversion back to square Image**

After the edges are detected, the image will then undergo thru segmentation but to do that we will convert the image back to square-image for this study’s approach is a mixed-hexagonal image processing.

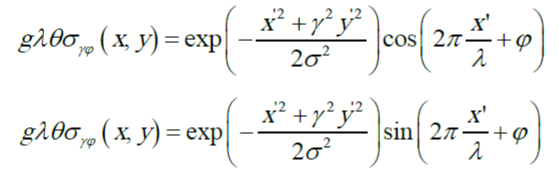
The conversion process will be done through the Hexagonal image processing framework by Middleton for they have provided an algorithm and implementation for converting hexagonal images back to square. The input image is in hexagonal lattice while the output would be in square lattice. This process will require several steps: Given a HIP image, first we need to determine the equivalent size of the target square image for this will make the square and HIP image to be of the same size. Second, the square sampling lattice needs to be defined thru the size information from the first step. Last, the values of the pixel at the square lattice points need to be computed by interpolation, based on the pixel values in the hexagonal image.

**3.3 Interpolation using Gabor filter**

During the conversion from square to hex-image, a considerable amount of image quality loss is common. So, in order to solve this problem and maintain the quality we need to use interpolation techniques for image reconstruction. This process is a routine in every image processing tasks during all transformation that is made on an image. In this study, Gabor filter will be used for image interpolation and will be used on the image not only once because the researcher will convert the image not only once (Jeevan and Krishnakumar, 2016).

Gabor filter (Ji et al., 2004) is the only filter with orientation selectivity that can be expressed as a sum of two separable filters. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function

modulated by a sinusoidal plane wave. The following equations represent the 2D-Gabor function which was proposed by Daugman in 1985. The first equation represents the real part of the function while the second represents the imaginary part.

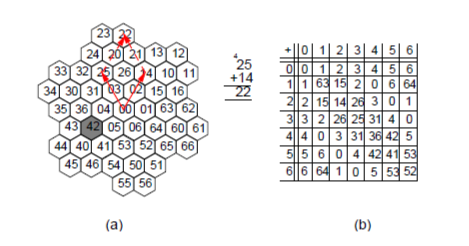


Where ‘x’ and ‘y’ are the position of x and y coordinate of the image. The sigma (σ) of the Gaussian factor which determines the size of the receptive field. The gamma (γ) as the aspect ratio specifies the ellipticity of the Gaussian factor. The value of γ vary in a limited range of 0.23 < γ < 0.92. The lambda(λ) is the wavelength. 1/λ the spatial frequency of the cosine factor.

**3.4 Addressing and Storage**

If a square image is converted to hexagonal image, addressing scheme for the pixels need to be devised. It is rather difficult to address all points from the hexagonal lattice defined from the converted image due to the fact that these points are aligned in two orthogonal directions. There various approaches that are known today such as using Two skewed Axes just like in Rosenfeld, Serra, Staunton, etc. The other is using Three Skewed Axes (Her , 1995 and 1994). Another approach was proposed by Overington in 1992 where the entire Hexagonal array is treated as if it is a rectangular array and Cartesian coordinates are directly employed to address all points. There are other ways in addressing a hexagonal lattice but this study will implore an alternate approach based on the symmetry of the hexagon (Middleton, 2000).

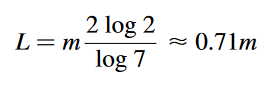
Consider a single hexagon to be a tile at layer 0. Six hexagons can be aggregated in each side by moving anti-clockwise direction. The new structure forms a new layer called layer 1, with the new tile being the super-tile of layer 0. Similarly, this process can be done repeatedly according to any number of layers. An example by Middleton will be found below. The number of hexagons that are contained in a layerL supertile can be computed as 7*L*. Each hexagon can then be numbered uniquely as a sequence of numbers each one giving its position in a given tile. The highlighted hexagon in the figure below is the fourth tile in layer 2 and the second tile at layer 1. Thus, all hexagons in Layer L super-tile can be addressed uniquely by a L-digit base 7 number and this will also encode the location of the hexagon. This indexing scheme implies that all point in the image can be represented by a single coordinate and this is a modified form of the GBT system (Gibson, 1982).



**Figure 3.3** ss

This single-index system for pixel addresses has several advantages. It promotes a full exploitation and manipulation of the symmetry in hexagonal lattice. It also allows the image to be stored using a vector. It makes the processing of the resampled square image more efficient because the Conversion of the Cartesian Coordinate into indices requires only one loop. It is also possible to use a single index for square-sampled images as well. This is done by reordering the rows or columns into a vector and manipulating the pointer into this vector.

Storage requirements for Hexagonal lattice depends on two factors: Resolution and Colour levels that are being used in the image. For a square-image with *M* X *N*  size with 24 bit colour will have a size of 3*MN* bytes. For the hexagonal-image’s required storage space using *L* where total pixel is 7*L* layers would be 3 X 7*L* bytes. For *M = N =* 2*m* , the relationship between *L* and *m* can also be computed as below.



**3.5 Image Processing Operations**

In conducting common image processing operations, one must know the neighbourhood of a pixel. It is the given distance from a central pixel. Therefore, the first hexagonal neighbourhood *N1*contains 6 pixels and the next *N*2 contains 19 pixels. Another neighbourhood *Ng(i)* permitted by the data structure is which is defined as the aggregations of tiles at a given level and appears to be approximately circular in shape.

The balanced ternary arithmetic (Her, 1994) is required in finding the neighbours of a given point in a hexagonal lattice, hence it is analogous to the vector addition. To get *N1* a balanced ternary addition of the numbers 1 to 6 o the central pixel is used. *N*2 and *Ng(i)* can also be found via the same process.

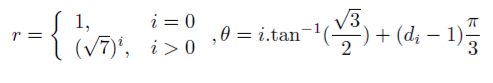
Convolution with a mask is a simple operation that uses neighbourhoods. Compared to the square-image which requires four nested loops, convolution in hexagonal-image requires only two nested loops.

The boundary of the image is easy to compute since hexagonal-image have index less than 7*L* (where L is the number of layers). Another task the derivation of an image pyramid should be performed by starting at the origin and averaging all the pixels in the *Ng(i)* neighbourhoods. Increasing the value of *i* can locate the layers of the pyramid.

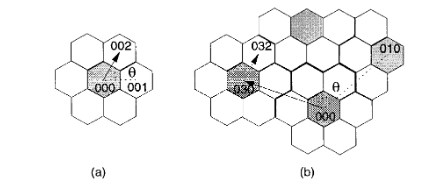
**3.6 Display of Hexagonal-images**

**3.6.1 Coordinate Conversion**

In displaying the hexagonal-image on a screen one must address the issue of conversion of unique index to a screen location. This process is straightforward as what Middleton did in his study “Shape extraction in a hexagonal-image processing framework” and “Edge detection in a hexagonal-image processing framework”, where he utilized polar coordinates. As the index can be written *dn-1…d0,* each number, *d1,* causes a change in the coordinates as shown below:

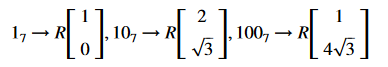


Take the figure below as an example for conversion of 327 to a pair of Cartesian coordinates.

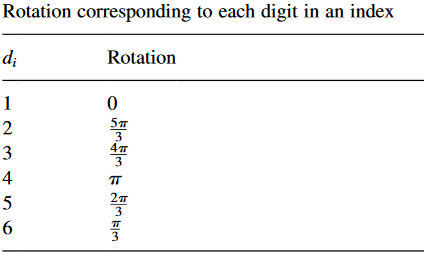


**Figure 3.4** ss

A number in the Balanced Ternary system can be written as *dn-1…d0.* Examination of successive digits starting from *d0* show an increase in radius. This increase can be seen by examining the sequence {17,107,1007,…}. For each point, the Cartesian Coordinates is expressed as a vector as shown below:



R is the inter-pixel spacing. The specific value of each digits are the rotation of this vector about the origin. The angle of rotation is illustrated below.



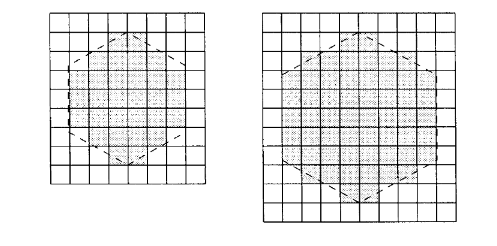
**Table 3.1** dd

A matrix can be formed for these rotations using the standard rotation matrix for Cartesian space. In the case a digit is equal to zero, this digit introduces no offset.

For index 327 from the above illustration, the 2 after suitable rotation and scaling generates the following coordinates (1/2. √3/2), and the 3 generates (-5/2, √3/2). These offsets are then added together to generate the final Cartesian coordinates (-2, √3).

**3.6.2 Constructing a Hyperpixel**

The image will be then displayed on the screen right after the computation of the Cartesian Coordinates. This can be done using the Hyperpixel an aggregation of pixels that when put together looks hexagonal in shape and is more pleasing to the eye as it takes into account the oblique effect in human vision.



**Figure 3.5** ss

There are two possible choices from above figure of a hyperpixel but the larger one is chosen for it represents hexagon’s shape better than the smaller one. A consequence when using the hyperpixel is the lesser screen resolution.

**3.7 Edge Detection and Smoothing of Hex-image**

To prepare the images of the copepods for segmentation an Edge detection filter is to be applied. The basic assumption used in most edge detection techniques is that the edges are characterized by large changes in intensity. Hence, at the location of an edge, the first derivative function should be a maximum or the second derivative should have a zero-crossing (Middleton, 2002). There are many edge detection techniques but Canny edge detector is what we’ll be using in this study.

The said technique is design to be an optimal edge detector under the following criteria: Detection, localization, and single response. It uses Gaussian smoothing and then directional derivatives to estimate the edge directions. It is somewhat a combination of Prewitt and LoG edge detection algorithms.

The operation of the canny first involves derivative computations in the horizontal and vertical directions using 7 x 7 masks. The squared responses are combined to generate a candidate edge map which is thresholded to generate the final edge map.