AIVP ADVANCES IN IMAGE AND VIDEO PROCESSING

SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM

VOLUME 2, ISSUE 2
ISSN: 2054 - 7412

Hexagonal pixel-array for efficient spatial computation for motion-detection pre-processing of visual scenes

Nicoladie D. Tam
Department of Biological Sciences, University of North Texas, Denton, TX 76023, USA
nicoladie.tam@unt.edu

ABSTRACT

Motion-detection, edge-detection, and orientation-detection often require spatial computation of the light intensity difference between neighboring pixel cells. Pre-processing the image at the retinal level can improve the computational efficiency when the parallel processing can be achieved naturally by the spatial arrangement of the pixel-array. Pixel-arrays that require spatial pre-processing computation have different geometric constraints than plain pixel-arrays that do not require such local computation. Our analysis shows that geometric optimization of pixel-arrays can be achieved by using hexagonal arrays over rectilinear arrays. Hexagonal arrays improve the packing density, and reduce the complexity of the spatial computation compared to rectilinear square and octagonal arrays. They also provide geometric symmetry for efficient computation not only at the contiguous neighboring cell level, but also at the higher-order neighboring cell level. The light intensity difference at the higher-order cell level is used to compute the first-order and second-order time-derivatives for velocity and acceleration detections of the visual scene, respectively. Thus, hexagonal arrays increase the computational efficiency by using a symmetric configuration that allows pre-processing of spatial information of the visual scene using hardware implementations that are repeatable in all higher-order neighboring pixels.

Keywords: Hexagonal pixel-array, rectilinear pixels, geometric optimization, computational efficiency, spatial computation, parallel pre-processing.

1. INTRODUCTION

Rectilinear array is commonly used in spatial computation because of the orthogonal nature of the orthogonal axes and the ease of its implementation in the Cartesian coordinate system. Typical examples of rectilinear arrays are implemented in the pixel elements of computer display screens and camera image sensors, where the pixel elements are configured according to the orthogonal *x*- and *y*-axes. Yet, square array does not necessarily provide the most symmetrical geometric configuration for efficient computation in the spatial dimension, when

DOI: 10.14738/aivp.22.153
Publication Date: 4th April 2014
URL: http://dx.doi.org/10.14738/aivp.22.153

computation between the adjacent neighboring cells is required. Efficient spatial computation can be optimized by the spatial relationship and the distance between adjacent cells.

Due to these optimization constraints, rectilinear array may not be an optimal system for spatial computation when the computation requires minimization of distance among neighboring cells in a densely packed area. Since most pixel-arrays in a computer monitor or camera do not require any local computations involving neighboring cells, it is not important to optimize the geometric configuration of the pixel-array in terms of computational efficiency. The choice of rectilinear array or hexagonal array is optimized for the number of colors used in the display pixels. For example, rectilinear array is optimal for 2 or 4 color pixels, such as B/W (black-and-white) pixels, RGBW (red, green, blue, white), and CYGM (cyan, yellow, green, magenta) color pixels, while hexagonal arrays are optimal for 3 color pixels, such as RGB (red, green, blue) color pixels. These choices of the geometric array are not necessarily optimized for spatial computational efficiency, because no local computation is required for displaying the pixels or processing the imagine sensor pixels. But if spatial pre-processing of the visual scene is required, then geometric optimization of the pixel-arrays is important to improve the computational efficiency at the local pixels level. In order to take advantage of the geometric arrangements of the pixel, pre-processing of the visual scene analysis can be done locally at the photo-detector level before transmitting the information to the CPU (central processing unit) for post-processing. The pre-processing can include edge-detection, motion-detection and orientation-detection of the visual scene.

1.1 . Biologically-Inspired Hexagonal Array for Photo-Detectors

Hexagonal array, on the other hand, provides a more compact geometric configuration for local computation that requires a spatial relationship between adjacent cells, such as the retina in the lens eyes and ommatidia [1] in the compound eyes (see Fig. 1). The difference between camera image sensors and retinal cells is that camera image pixels detect light only, but retinal cells not only detect light, but also perform local computation to detect edge, orientation, and motion of the visual scene. The retinal cells pre-process the image signals so that the retina not only detects the intensity of light, but also performs scene analysis for edge-detection, spatial orientation-detection, and motion-detection on the detected image. The pre-processed patterns of spatial and motion orientation information on the visual scene are transmitted from the retina to the brain, in addition to the light intensity information of the image. Therefore, spatial processing of image pattern requires efficient computation at the local pixel level.

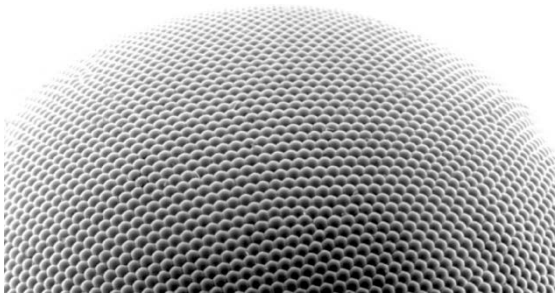

Figure 1: Image of a compound eye showing the hexagonal array of ommatidia (light sensors) of an insect.

Thus, it is important to optimize the geometric pattern of the retinal array of light-sensing pixels to perform efficient local computation of spatial information in relation to the neighboring cells. In this paper, we explore the computational efficiency using hexagonal array compared to rectilinear array. We will compare and contrast the geometric configurations of hexagonal array and rectilinear array in terms of spatial computation so that an optimal geometric design can be determined to improve its computational efficiency.

1.2 Spatial Computation by Local Pixels

Most camera image sensors detect light intensity information and transmit the image intensity information to the central processors for post-processing. In contrast, most retinal photo-detector cells in animals not only detect light intensity, but also process the image locally for edge-detection and motion-detection, before transmitting such orientation-detection information to the brain. Thus, the pre-processing visual image embeds not only a static image, but also a dynamic image that contains both motion and spatial information of the visual scene. Transmitting such orientation-detection dynamic image provides efficiency by compacting the image with both static and dynamic information content of the visual scene.

The first step in image acquisition is the detection of light intensity information by the camera image sensors. In animals, the image is acquired by the photo-detector cells of the retina or the ommatidia of the compound eye. Once the image is acquired, higher-order information concerning the visual scene can be extracted for edge-detection, spatial orientation-detection, and motion-detection [2]. Traditionally, this post-processing of image is performed by the CPUs of the computer instead of locally at the image-detecting array. Yet, the higher-order information can be computed much more efficiently by neighboring pixel cells locally prior to transmitting the image to the CPUs. This is because efficient processing of the local pixel information is highly parallelizable since the neighboring pixels are already geometrically arranged optimally for spatial processing. Instead of post-processing the image pixel-by-pixel sequentially at the computer, parallel pre-processing can be done locally at the image detector array level before transmitting the processed image to the CPU. It is more efficient to process the image by taking advantage of the geometric configurations of the pixels that are already arranged in parallel. This specialized computing architecture of the pixel-array

can lend itself to efficient computation for scene analysis, if the geometric arrangement of the pixels is optimized.

Optimization of the geometric pattern depends highly on the computations required for processing the image. The information contained in each pixel is often related to its neighboring pixels in most real-world visual scenes. For instance, the adjacent pixels are related for detection of a line in a visual scene. If an entire scene is moving in one direction, the adjacent pixels are also related by changing progressively as the scene propagates from one pixel to another. Such changes can be detected by the difference in intensity of adjacent pixels, since the velocity is computed by the distance of the corresponding pixels moved laterally divided by the time of motion. The direction of motion is detected by the direction in which the propagation occurs. Thus, the geometric arrangement of the pixels can provide the direction of motion and/or the orientation of the detected edges automatically prior to post-processing at the CPU.

2. GEOMETRIC OPTIMIZATION FOR SQUARE AND OCTAGONAL PIXEL-ARRAYS

Square array is a good candidate geometric array for computing the local differences in light intensity between neighboring cells. Because the array is orthogonal to each other, the direction (angle) of direct motion-detection is limited to 90^{o}. In a square array, each cell shares 4 contiguous neighbors. Detection of 90^{o}-motion can be detected optimally by the 4 contiguous neighboring cells. If the motion is shifted by a 45^{o} angle, computation between corner cells is required, but these corner cells are non-contiguous (see Fig. 2). Furthermore, the corner cells are not equidistance to the adjacent cells (that share contiguous contact between them).

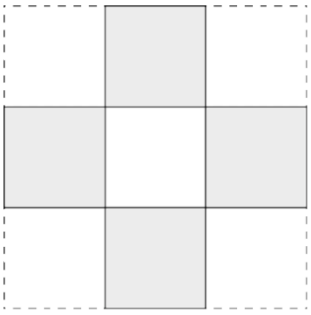

Figure 2: Diagram of a rectilinear square showing the center cell (in white) surrounded by the contagous neighboring cells (in shaded color).

To resolve this corner-cell problem, an interlaced octagon-surround, square-center tile array (see Fig. 3) can be used instead. This square-center arrangement surrounded by octagons will eliminate the non-contiguous corner cells. But this solution introduces an additional dilemma because of the difference in size and shape of the squares and octagons. This means each cell will detect a different amount of photons, which will introduce biases with this asymmetrical geometry, causing erroneous motion-detection. Thus, geometric symmetry of

adjacent cells and equal surface area for all cells are important considerations in spatial computation for proper motion-detection.

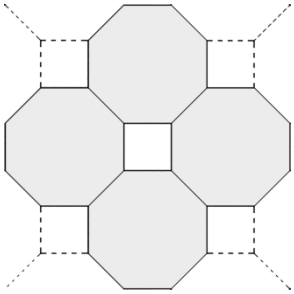

Figure 3: Diagram of a octagon-surround, square-center tile array showing the center cell (in white) surrounded by the contagous neighboring cells (in shaded color).

An alternative is to use a square-surround, octagon-centered arrangement surrounded by alternating squares and octagons (see Fig. 4). This geometric arrangement has all 8 contiguous neighboring cells, which will allow detection of both 45^o and 90^o motions without the corner-cell problem. This arrangement still has the same geometric asymmetry and different surface area as before. In fact, the geometric tile arrangement of Fig. 4 is exactly the same as the tile arrangement of Fig. 3, except that it is rotated by a 45^o angle (by comparing Figs. 3 and 4). Although this configuration resolves the corner-cell problem, it breaks the symmetry of having two classes of cells – those with 8 contiguous neighbors and 4 contiguous neighbors. Thus, it requires two sets of algorithms for computing the local light intensity differences that correspond to the direction of motion angle.

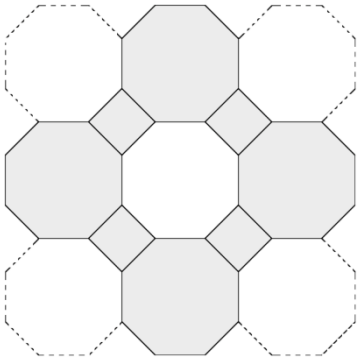

Figure 4: Diagram of a square-surround, octagon-center tile array showing the center cell (in white) surrounded by the contagous neighboring cells (in shaded color).

3. GEOMETRIC OPTIMIZATION FOR HEXAGONAL PIXEL-ARRAY

Hexagonal arrays are often found in compound eyes of insects and invertebrates, forming an array of ommatidia (photo-detectors) for light detection. Such hexagonal arrays (Fig. 1) are more commonly found in animals and plants than any other geometric arrays, such as rectilinear orthogonal arrays This is because motion-detection in compound eyes can be computed based on the local difference of light intensity between adjacent neighboring cells [3-5] of the ommatidia using the process called lateral inhibition [3, 6]. Lateral inhibition is a

contrast-enhancement computation used by an animal's neural circuitry to exaggerate the light intensity differences of the neighboring cells for edge-detection or two-point discrimination. An edge (or a point) is enhanced by suppressing the light intensity of its neighbors using the lateral inhibition spatial circuitry.

In contrast, hexagonal array provides the best candidate for contiguous neighboring-cell computation of local light intensity difference between adjacent cells without the complexity of the square or octagonal rectilinear array discussed above [7]. The first-order computation of light intensity differences can be computed using the contiguous neighboring cells at 60^o angles (see Fig. 5). With this hexagonal configuration, finer angular differences, such as 30^o angles, can be obtained using the second-order adjacent cells. Thus, this provides a means for symmetric computation using one set of computational algorithms to compute the light intensity difference of exactly 6 contiguous neighbors. Each successive higher-order neighbor will compute the light intensity difference with the angular direction rotated by 30^o successively.

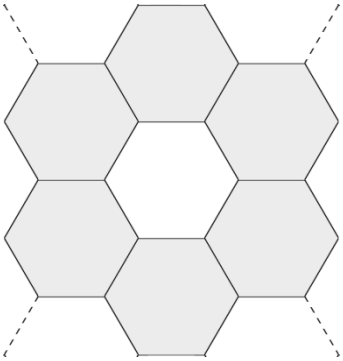

Figure 5: Diagram of a hexagonal array array showing the center cell (in white) surrounded by the contagous neighboring cells (in shaded color).

Edge-detection can be implemented locally at the pixel level by comparing the light intensity gradient of neighboring cells. Lines are detected by similar luminance in adjacent pixels, while edges are detected by sudden change in light intensity along the detected lines. Numerous studies have shown that hexagonal pixel-arrays are more efficient for edge-detection instead of rectilinear arrays by using spatial computation [2, 8-13]. There is a 40% computational efficiency using a hexagonal edge-detection operator [2, 11].

We propose that the use of hexagonal arrays to perform spatial computation can also improve the efficiency of motion-detection, not just edge-detection. The significance of this symmetrical geometry arrangement can be appreciated when the computation of light intensity difference between second-order and higher-order adjacent cells is required, such as the computation needed for finer angular increments, velocity and acceleration computation (second-order and third-order time-derivatives). Velocity of the image is computed by the first-order derivative of the light intensity difference between adjacent cells. Acceleration is computed by the second-order derivative of the light intensity difference between adjacent cells.

Insects (such as flies) often use this motion-detection of shadows to trigger the escape reflex circuitry to flee from predators when the velocity of the shadow exceeds the "critical velocity" threshold. Such escape requires the detection of both motion and direction of the predator's shadow, which can be detected instantaneously at the ommatidia level of the compound eyes using spatial computation. Motion is detected by the velocity and acceleration of the image based on the changes in local light intensity difference between adjacent photo-detector cells. The direction of motion is detected by the propagation of such local changes in these neighboring cells. This spatial computation can be done by a simple circuitry at the photo-detector level of the ommatidia to produce the escape reflex in insects that do not even have the complex circuitry of a brain.

4. GEOMETRIC OPTIMIZATION FOR PACKING DENSITY OF PIXEL CELLS

Hexagonal arrays also provide more optimal packing density than rectilinear square arrays. In order to prove that hexagonal arrays can achieve better packing density than square arrays, let us use circular pixel cells for the array, since a circle is symmetrical in all directions, and a circle will always form contacts with its neighbor, independent of whether the array is rectilinear or hexagonal.

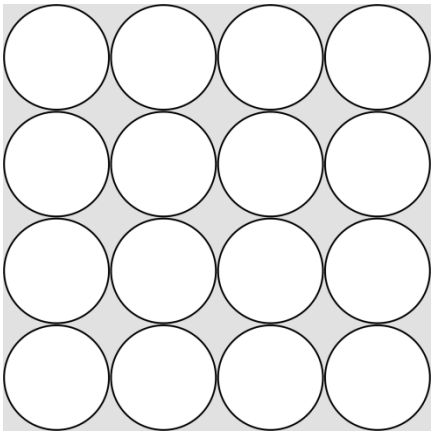

Figure 6: Diagram of a square array of circles to illustrate the packing density of rectilinear square array.

Fig. 6 shows the rectilinear array of circular pixels. The packing density is derived from the percentage of space occupied by the circular pixels compared to the empty space not occupied by the pixel:

$$d_{rect} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{2} = 78.54\% \quad (1)$$

where d_{rect} denotes the package density for the rectilinear array, and r is the radius of the circle.

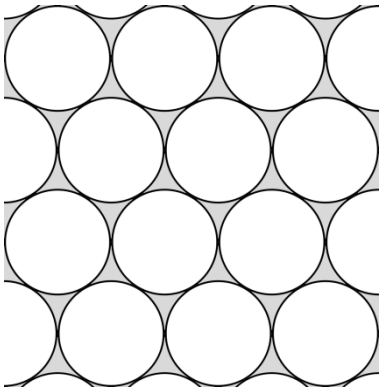

Figure 7: Diagram of a hexogonal array of circles to illustrate the packing density of rectilinear square array.

On the other hand, Fig. 7 shows the hexagonal array of circular pixels. The packing density for hexagonal array is given by:

$$d_{hex} = \frac{7\pi r^2}{3(3r)^2 \sin(\pi/3)} = \frac{7\pi}{27\sin(\pi/3)} = 94.05\% \quad (2)$$

where d_{hex} denotes the package density for the hexagonal array. This shows that hexagonal arrays have a drastic improvement in packing density over rectilinear arrays – 94.1% vs. 78.5%. Optimizing the pixel-array geometrically with hexagonal cells will have a significant improvement on packing pixels in motion-detecting cameras and in the compound eyes that require spatial computation with adjacent neighboring cells.

5. EDGE-DETECTION BY SPATIAL PROCESSING

The edge in a visual scene can be detected by the orientation of a line of sharp change in light intensity. Instead of merely detecting the light intensity of the pixels in the photo-detector array, edge-detection can be processed locally within the pixel-array. The abrupt change in light intensity of the neighboring pixels can be processed spatially within the photo-detector array by taking the light intensity difference of the neighboring pixels.

The orientation of the line can be detected by the similar light intensity in the longitudinal direction of the neighboring pixels and abrupt change in light intensity of the neighboring pixels in the orthogonal direction. Thus, intensity thresholds can be used to detect edges along the longitudinal and orthogonal directions.

Let p be the light intensity detected by the photo-detector. The difference between the light intensity of the neighboring cells, Δp_x, is given by:

$$\Delta p_{x12} = p_{x1} - p_{x2} \quad (3)$$

where p_{x1} and p_{x2} are the adjacent pixels in the x-direction (along the x-axis). Similarly,

$$\Delta p_{y12} = p_{y1} - p_{y2} \quad (4)$$

Δp_x represents the light intensity difference in the *y*-direction. The thresholds for edge-detection in the *x*-direction is given by:

$$\begin{cases} \Delta p_x < \varepsilon \\ \Delta p_y > \theta \end{cases} \quad \text{for } \varepsilon << \theta \tag{5}$$

The edge-detection in any arbitrary direction can be obtained by the rotational transformation of the *x*-axis. Since the geometrical arrangement of the pixel-array lends itself naturally to the orthogonal edge-detection directions for rectilinear pixel-arrays, the principle directions for optimal edge-detection are the horizontal (0^o) and vertical (90^o) directions, if the edges are aligned horizontally or vertically. This can be accomplished by setting the thresholds for the nearest-adjacent neighbors in the hardware implementation for the spatial processing. For diagonally oriented (45^o) edges, the next-adjacent neighbors can be used for setting the threshold.

For hexagonal pixel-arrays, edge-detection can be optimally computed by setting the thresholds for nearest-adjacent neighbors aligned in the 0^o, $60^o$and 120^o directions. Similarly, edges aligned along the 30^o, $90^o$and 150^o directions can be detected by the next-adjacent pixel neighbors. Because the distance between the nearest- and next-adjacent pixels are multiples of each other in the hexagonal array, the hardware implementations for edge-detection spatial computation are symmetrical in all directions (see Fig. 5). In contrast, since the distances between nearest- and next-adjacent neighbors are not multiples of each other for rectilinear pixel-arrays, the complexity of the hardware implementations increases because of the asymmetry in edge-detection for different directions (see Figs. 2-4). Thus, hexagonal pixel-arrays are ideal for edge-detection compared to rectilinear arrays.

6. IMAGE SMOOTHING BY SPATIAL PROCESSING

Since video images are often noisy, the static noise can introduce errors in the edge-detection algorithm when the light intensity exceeds the threshold by accidental noise. To eliminate the noise-effect, spatial smoothing can be done by averaging the neighboring pixels. Instead of computing the difference between neighboring pixels (as in edge-detection), the sum of the neighboring pixels can be used in the smoothing algorithm.

$$p_{x12} = p_{x1} + p_{x2} \tag{6}$$

Thus, the edge-detection can be processed after the image has been smoothed to reduce static noise. In other words, the hardware implementations requires only an adder (and a subtractor, which can be implemented by an adder using 2's complement arithmetic) for each neighboring pixels symmetrically located spatially in the pixel-array.

Note that different thresholds can be used to detect sharp vs. soft edges. Furthermore, since most edges don't necessarily span just the nearest neighboring pixels, multiple higher-

order next-neighbors can be used in both spatial smoothing and edge-detection in combination. The sharpness of the edge can be determined by how many higher-order next-neighbors the leading-edge spans. The length of the edge can be determined by how many higher-order next-neighbors the light intensity remains constant.

7. MOTION-DETECTION BY SPATIAL PROCESSING

Motion-detection can be computed by the rate of change of the local intensity difference of the neighboring cells (photo arrays). An adaptive threshold light intensity difference of the neighboring cells can be used to detect motion direction. Similar to edge-detection, it can detect motions in the 6 different directions at 60^o to each other for a hexagonal array, whereas for rectilinear array, it can only detect motions in 4 different directions at 90^o to each other along the principle axes. Motion-detection can be achieved efficiently using spatial processing algorithm implemented in the hardware.

Since the distance between contagious neighboring cells are known (fixed in the photo-detector array), the rate of change of light intensity can be used to compute the velocity of motion. Let x be the distance traveled in the visual scene, then the velocity of motion, v, is given by:

$$v = \frac{x}{\Delta t} \tag{7}$$

where Δt is the time-increment between each video frame (frame rate). In order to detect motions across different pixels, the corresponding (similar) light intensity would traverse across those pixels. One of the efficient methodologies for detecting motion is to incorporate the edge-detection computed earlier for motion-detection. If an edge is detected progressively along the neighboring pixels, then motion can effectively be detected. The thresholds for sharp edges and soft edges can be adjusted adaptively to accommodate for various visual scenes.

The velocity of the motion can be computed by the propagation of the edge along the neighboring pixels based on the center-to-center distance (Δx) between neighboring pixels. The advantage of hexagonal pixel-array is that the distances between nearest- and next-adjacent neighbors are multiples of each other; the motion-detection circuitry can be symmetrical in all directions, unlike rectilinear array. This symmetry allows for simple hardware implementation for motion-detection without the extra complexity in the asymmetrical nearest- and next-adjacent neighbors for rectilinear array.

8. CONCLUSION

Geometric optimization of pixel-arrays that require local spatial computation can be achieved by using hexagonal arrays, instead of using the traditional rectilinear arrays. Hexagonal arrays improve the packing density, as well as reducing the complexity of spatial computation, compared to rectilinear square or octagonal arrays. Hexagonal arrays also

provide geometric symmetry with the contiguous neighboring cells, as well as higher-order neighboring cells. Hexagonal array increases the computational efficiency by using a symmetric configuration that is repeatable in all higher-order cells. This simplifies the computational hardware implementation for motion-detection, edge-detection, and orientation-detection. The efficiency can also be improved by implementing the spatial computation at the photo-detector level using hexagonal arrays for pre-processing. It will increase the information content of the visual scene transmitted to the CPU (or the brain) by embedding the dynamic information of the pre-processed image that includes motion, edge and orientation-detection, in addition to the static information of a plain motionless image.

9.ACKNOWLEDGEMENT

I greatly appreciate Ms. Krista Smith for the helpful suggestions, and for proofreading the manuscript.

REFERENCES

[1] Blum, M. and T. Labhart, *Photoreceptor visual fields, ommatidial array, and receptor axon projections in the polarisation-sensitive dorsal rim area of the cricket compound eye.* J Comp Physiol A, 2000. 186(2): p. 119-28.

[2] Staunton, R.C., *The design of hexagonal sampling structures for image digitization and their use with local operators.* Image and Vision Computing, 1989. 7(3): p. 162-166.

[3] Coleman, B.D. and G.H. Renninger, *Theory of delayed lateral inhibition in the compound eye of limulus.* Proc Natl Acad Sci U S A, 1974. 71(7): p. 2887-91.

[4] Schultz, A.M. and M.J. Wilcox, *Parallel image segmentation using the L4 network.* Biomed Sci Instrum, 1999. 35: p. 117-21.

[5] Srinivasan, M.V. and G.D. Bernard, *The effect of motion on visual acuity of the compound eye: a theoretical analysis.* Vision Res, 1975. 15(4): p. 515-25.

[6] Laughlin, S., *A simple coding procedure enhances a neuron's information capacity.* Z Naturforsch C, 1981. 36(9-10): p. 910-2.

[7] Tam, D.C., *Computational efficiency circuitry for detecting shadows in hexagonal array of compound eye.* Neurocomputing, 2004. 58–60(0): p. 1073-1078.

[8] Davies, E.R., *Optimising computation of hexagonal differential gradient edge detector.* Electronics Letters, 1991. 27(17): p. 1526 – 1527.

[9] He, X., et al., *Bilateral Edge-detection on a Virtual Hexagonal Structure*, in *Advances in Visual Computing*, G. Bebis, et al., Editors. 2006, Springer Berlin Heidelberg. p. 176-185.

[10] Abu-Baker, S. and R.J. Green. *Detection of edges based on hexagonal pixel formats*. in *Signal Processing, 1996., 3rd International Conference on*. 1996.

[11] Staunton, R.C. *Hexagonal image sampling: A practical proposition*. in *Proc. SPIE 1008, Expert Robots for Industrial Use*. 1989.

[12] Gardiner, B., S. Coleman, and B. Scotney. *Multi-scale Feature Extraction in a Sub-pixel Virtual Hexagonal Environment*. in *Machine Vision and Image Processing Conference, 2008. IMVIP '08. International*. 2008.

[13] Davies, E.R., *Low-level vision requirements.* Electronics & Communication Engineering Journal, 2000. 12(5): p. 197 – 210.