A coordinate system for hexagonal pixels

Wesley E. Snyder[1], Hairong Qi[1], William Sander[2]
[1]Center for Advanced Computing and Communication,
North Carolina State University, Raleigh, NC 27695-7914
[2] U.S. Army Research Office, Research Triangle Park, NC

ABSTRACT

A coordinate system is described which provides a natural means for representing hexagonally-organized pixels. The coordinate system has the unusual property that its basis vectors are not orthogonal. Vector-space properties and operations are described in this coordinate system, and shown to be straightforward computations. Some image processing algorithms, including calculations of image gradients and variable-conductance diffusion, are expressed and analyzed.

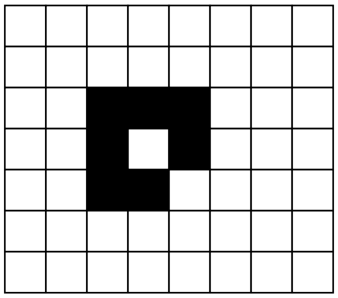
Keywords: hexagonal pixels, h2 coordinate system, orthogonal basis vector, variable conductance diffusion

1. INTRODUCTION

In a number of papers[1, 2], imaging sensors have been described which use hexagonally-organized pixel arrays. Hexagons are the minimum-energy solution when a collection of tangent circles with flexible boundaries are subjected to pressure. The beehive is the best known such naturally occurring organization, but many others occur, including the organization of cones in the human retina.

Traditionally, electronic imaging sensors have been arranged in rectangular arrays, originally because an electron beam needed to be swept in a raster-scan way, and more recently because it is slightly more convenient to arrange charge-coupled devices in a rectangular organization.

Rectangular arrays, however, introduce an ambiguity in attempts to define neighborhoods, and one may define a neighbor of a pixel to be a *four-neighbor* (above, below, left, and right), or an *eight-neighbor* (which includes the neighbors which only touch at the corners). This ambiguity produces a well-known[3, 4] paradox illustrated in Fig. 1. If one chooses an 8-connected definition, the washer-shaped object has a closed boundary. That is, one can travel all the way around the object only passing through immediate neighbors. However, although the boundary is closed, the inside is connected to the outside. A similar paradox occurs if one chooses 4-connectedness. A solution has been proposed which states that one must use 4-connectedness for foreground and 8-connectedness for background (or vice-versa) and the paradox is resolved. Herman[5] follows this philosophy in his careful analysis of connectivity in images, by defining connectivity functions which are different between background and foreground. No literature is known that addresses such issues in multi-color images.



1. An illustration of an object which has a closed boundary but no inside (if one chooses 8-connectedness) or an open boundary, with a distinct inside and outside, if one chooses a 4-connected definition.

An additional disadvantage of the square pixel is that the 8 neighbors around the center pixel are not equidistant, which

Correspondence: W. E. Snyder. Other Author Information: W.E.S.: Email: wes@eos.ncsu.edu, Telephone: 919-515-3015. H.Q.: Email: hqi@eos.ncsu.edu, Telephone: 919-515-3015. W.S.: Email: sander@aro-emh1.army.mil, Telephone: 919-549-0641.

causes the accuracy for the diagonal and off diagonal directions to be reduced in their magnitude.

In this paper, we are motivated to reexamine hexagonal pixels. CCD technology certainly permits development of hexagonally-spaced arrays. Furthermore, ongoing work in silicon retinas[6,7,23] uses this organization. Finally, we see no connectivity paradoxes which occur in hexagonal connectivity analysis: every pixel has exactly six neighbors, foreground, background, or other colors.

Stimets argued in[2] that despite more than three decades of research in computer vision there does not exist today any machine vision system which comes anywhere near the human visual system in its combination of versatility and speed. Foreseeable improvements in processor speed alone will not make up the difference; and he proposed a machine vision system which would incorporate three general features: a hexagonal pixel grid, parallel processing at the local level, and parallel/series procedures with feedback using a pyramidal hierarchy of grids to combine local features and synthesize the overall pattern. Among all these three features, hexagonal pixel grid is fundamental.

From the literature, we find the following reasons for choosing a hexagonal grid:

- It models human visual system more precisely. The transformation from image to cortical representation consists of two stages. The first stage is between the image and the retinal neural image; the second stage is between the retinal neural image and the cortical neural image[8]. The receptors are inhomogeneously distributed on the retina. The largest receptor density occurs in the center of the fovea and decreases radially outwards. The neurons are assumed to be uniformly distributed as a regular hexagonal lattice.
- It benefits hardware design. From a hexagonal grid, it is relatively easy to extract analog data by a set of parallel (vertical) buses[2].
- The distance between a given pixel of interest and its immediate neighbors is the same along any of the six main directions of the hexagonal tessellation, which is a condition that considerably facilitates the implementation of circularly symmetric kernels[2].
- As mentioned earlier, a hexagonal tessellation removes the connectivity ambiguity inherent in rectangular tessellation.
- For radially symmetric objects, hexagonal sampling requires 13% fewer samples than rectangular sampling[9]. The computational load for implementation of digital filters is reduced due to the 12-fold symmetry of the hexagon (that for rectangular is 8-fold).
- Artificial neural networks can be trained with less errors and better recall if hexagonal samples are used[1].
- A hexagonal sampling technique for continuous tone images shows *fewer false contour artifacts, better edge definition*[10].
- In morphological operations for shape analysis, a hexagonal grid offers greater simplicity for nearest neighbor connectivity calculations[11].
- A hexagonal grid performs better in tracking and locating the point sources which ultimately leads to reduced location error and less computational load[9,12].

A good deal of research has been conducted on how to transform between hex domain and square domain. That is, how to conduct the usual operation such as Fourier transform, gradient, diffusion, filter, etc., that are well-defined under Cartesian coordinates, under a hexagonal tessellation, and how to define coordinate systems in hex domain.

Carstens in[13] discussed the issue on how to transform a square pixel representation into a hexagonal pixel representation, requiring that the transform must be invertible. That is, after the operations in hex domain, the inverse transform must be able to transform the results back into square pixels for output.

Waston in[14] presented a hexagonal orthogonal-oriented quadrature pyramid. This pyramid uses a hexagonal lattice for which the basis kernels are orthonormal, local in space, local in frequency, and self-similar. Seven orthonormal basis kernels are use to obtain seven coefficients, which are a complete representation of the neighborhood. The coefficients produced by each distinct type of kernel may be regarded as a subimage.

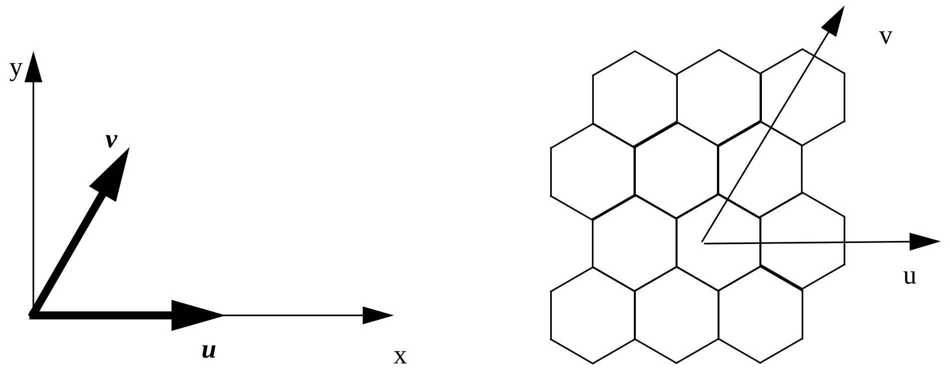
Roessel in[15] proposed Generalized Balanced Ternary (GBT) system as the coordinate system in hex domain.

We now develop a coordinate system to represent the points. This representation should have the following properties:

1. Complete - Be sufficient to represent any point in a 2-space;
2. Unique - For any point in space, there should be only a single pair of numbers, *u*, and *v*, which represent the location of that point;
3. Convertible - Be easily converted to and from Cartesian coordinates; and
4. Efficient - Be a convenient and efficient representation.

We choose and analyze the coordinate system illustrated in Fig. 2. Some aspects of this coordinate system have been primarily investigated. For example, Luczak & Rosenfeld[21] defined a hexagonal distance metric using such a system, and Bell *et. al.*[22] used a complex number formalism to deal with geometric transformation on such a grid. In Fig. 2., each pixel is represented by a horizontal deflection followed by a deflection upward and to the right. These directions are represented by a pair of unit vectors, ***u*** and ***v***, as illustrated in Fig. 2. We refer to this coordinate system as the "h2" system.

Notation: We denote a point in R^2 by $\boldsymbol{p} = u\,\boldsymbol{u} + v\,\boldsymbol{v}$. Where the unbolded character denotes the magnitude in the direction of the unit vector denoted by the bold character. In the case that we discuss two or more points, we will denote different vectors by using subscripts, with the same subscripts on the components, e.g. $\boldsymbol{p}_i = u_i\boldsymbol{u} + v_i\boldsymbol{v}$



2. A coordinate system which is natural to hexagonal tessellation of the plane. The *u* and *v* directions are not orthogonal, Unit vectors ***u*** and ***v*** describe this coordinate system.

We will also use column vector notation for such points:

$$\boldsymbol{p}_i = [u_i, v_i]^T \tag{1}$$

Sometimes, we will be interested in the locations of points in the familiar Cartesian representation, $[x, y]^T$. In this case, we will denote points by subscripts as well. e.g.

$$\boldsymbol{p}_i = [u_i, v_i]^T = [x_i, y_i]^T$$

with corresponding values for *u, v, x,* and *y.*

2. ANALYSIS OF THE h2 SYSTEM

Lemma 1: Any ordered pair, [*u*,*v*], corresponds to exactly one pair [*x*,*y*].

Proof: Using simple trigonometry, and noting that the cosine of 60 degrees is 1/2, it is straightforward to derive that

$$x = u + \frac{v}{2} \text{ and } y = \frac{\sqrt{3}v}{2}. \tag{2}$$

Lemma 2: Any ordered pair of Cartesian coordinates, [*x*,*y*], corresponds to exactly one pair [*u*,*v*].

Proof: By solving Eq. (2) for *u* and *v*, we find

$$u = x - \frac{y}{\sqrt{3}} \text{ and } v = 2\frac{y}{\sqrt{3}}. \tag{3}$$

A set of vectors $\lambda_1, \lambda_2, \ldots\lambda_d$ is said to be a *basis* for the vector space R^d if any vector in R^d can be written as a linear combination of $\lambda_1, \lambda_2, \ldots\lambda_d$. If the λ_i are orthonormal ($\lambda_i\lambda_j = 0$ if $i \neq j$ and $\lambda_i\lambda_i = 1$), this is sufficient to claim that these vectors constitute a basis. However, ***u*** and ***v*** are normalized but they are not orthogonal. To see this, simply represent *u* and *v* in Cartesian coordinates and take their inner product:

$$\boldsymbol{u} = \boldsymbol{x}, \boldsymbol{v} = \frac{1}{2}\boldsymbol{x} + \frac{\sqrt{3}}{2}\boldsymbol{y}, \text{ so } \boldsymbol{u}^T\boldsymbol{v} = [1, 0]\begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} = \frac{1}{2}.$$

Theorem: The vectors ***u*** and ***v*** form a basis for R^2.

Proof: Since *x* and *y* obviously are a basis for R^2, we can write any point ***p*** in R^2 as an ordered pair ***p*** = $[x,y]^T = x\boldsymbol{x} + y\boldsymbol{y}$. But from Lemma 2, we have $\boldsymbol{p} = x\boldsymbol{u} + y\frac{(2\boldsymbol{v} - \boldsymbol{u})}{\sqrt{3}} = \left(x - \frac{y}{\sqrt{3}}\right)\boldsymbol{u} + \frac{2y}{\sqrt{3}}\boldsymbol{v}$. Thus, any point in R^2 may be written as a weighted sum of ***u*** and ***v***. QED

2.1 Properties of the h2 Coordinate System

Following are some observations which are straightforward to derive.

1) Given two vectors, $\boldsymbol{P}_1 = [u_1, v_1]^T$ and $\boldsymbol{P}_2 = [u_2, v_2]^T$, the vector sum $\boldsymbol{P}_1 + \boldsymbol{P}_2$ is $[u_1 + u_2, v_1 + v_2]^T$. (Vector addition works like one expects it to.)

2) The squared (Euclidean) distance between two points P_1 and P_2 is $(u_1 - u_2)^2 + (v_1 - v_2)^2 + (u_1 - u_2)(v_1 - v_2)$.

3) The magnitude of vector $\boldsymbol{P} = [u, v]^T$ is $\sqrt{u^2 + v^2 + uv}$.

4) The projection of one vector $\boldsymbol{P}_1$ onto another, $\boldsymbol{P}_2$, is

$$\frac{2u_1u_2 + u_1v_2 + u_2v_1 + 2v_1v_2}{\sqrt{u_2^2 + v_2^2 + u_2v_2}}. \tag{4}$$

Note that the projection is not the inner product. The case of projection onto the coordinate axes requires special attention. Demonstration of this result provides an interesting insight into the use of this coordinate system, and is therefore presented in tutorial detail:

We define the *Cartesian Inner Product* of two vectors, $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ by $\mathrm{CIP}(\boldsymbol{P}_1, \boldsymbol{P}_2) = x_1x_2 + y_1y_2$ and observe that this is not the same as $u_1u_2 + v_1v_2$. There are two ways to represent CIP ($\boldsymbol{P}_1, \boldsymbol{P}_2$) in terms of *u* and *v*. The first is a direct substitution from Eq. (2),

$$\mathrm{CIP}(\boldsymbol{P}_1, \boldsymbol{P}_2) = \left(u_1 + \frac{v_1}{2}\right)\left(u_2 + \frac{v_2}{2}\right) + \frac{\sqrt{3}v_1}{2} \cdot \frac{\sqrt{3}v_2}{2} = \left(\frac{2u_1 + v_1}{2}\right)\left(\frac{2u_2 + v_2}{2}\right) + \frac{3}{4}v_1v_2 \tag{5}$$

$$= u_1u_2 + \frac{u_1v_2 + u_2v_1}{2} + v_1v_2 \tag{6}$$

Another way to think about CIP (***P*** $_1$, ***P*** $_2$), which will turn out to be useful, is the law of cosines,

$$\|\boldsymbol{p}_1\|^2 + \|\boldsymbol{p}_2\|^2 - \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|^2 = 2\|\boldsymbol{p}_1\|\|\boldsymbol{p}_2\|\cos\theta \tag{7}$$

However, the cosine of the angle between two unit vectors is also the Cartesian inner product:

$$\cos\theta = \frac{\text{CIP}(\boldsymbol{p}_1, \boldsymbol{p}_2)}{\|\boldsymbol{p}_1\|\|\boldsymbol{p}_2\|} \tag{8}$$

$$= \frac{(\|\boldsymbol{p}_1\|^2 + \|\boldsymbol{p}_2\|^2 - \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|^2)}{2\|\boldsymbol{p}_1\|\|\boldsymbol{p}_2\|} \tag{9}$$

Equating the numerators, we have

$$2\text{CIP}(\boldsymbol{p}_1, \boldsymbol{p}_2) = \|\boldsymbol{p}_1\|^2 + \|\boldsymbol{p}_2\|^2 - \|\boldsymbol{p}_1 - \boldsymbol{p}_2\|^2 \tag{10}$$

$$= (u_1^2 + v_1^2 + u_1v_1 + u_2^2 + v_2^2 + u_2v_2 - \|(u_1 - u_2), (v_1 - v_2)\|^2) \tag{11}$$

which simplifies to result in the same result as Eq. (6). Once we have the CIP of the two vectors, it is straightforward to use the fact that the magnitude of the projection of vector $\boldsymbol{p}_1$ onto vector $\boldsymbol{p}_2$ is $\frac{\text{CIP}(\boldsymbol{p}_1, \boldsymbol{p}_2)}{\|\boldsymbol{p}_2\|}$, which results in the result presented in Eq. (4).

5) Given an arbitrary vector $\boldsymbol{P} = [u, v]^T$, its projection in the ***u*** direction is *u+v/2*, and its projection in the ***v*** direction is *v*.

6) The cosine of the angle between two vectors is $\cos\theta = \frac{u_1u_2 + v_1v_2 + \left(\frac{u_1v_2 + v_1u_2}{2}\right)}{\sqrt{u_1^2 + v_1^2 + u_1v_1}\sqrt{u_2^2 + v_2^2 + u_2v_2}}$. The computation of this angle should be compared with the same calculation in *x*-*y*:

$$\cos\theta = \frac{x_1x_2 + y_1y_2}{\sqrt{x_1^2 + y_1^2}\sqrt{x_2^2 + y_2^2}}.$$

Theorem: The equation $v = mu + b$ defines a straight line.

Proof: We substitute Eq. (2) into the slope-intercept form given above, and produce a new equation, in *x* and *y*,

$$y = \frac{\sqrt{3}m}{2+m}x + \frac{\sqrt{3}}{2+m}b.$$

Since *b* is arbitrary, the second term is unconstrained, and we have a straight line with slope $\frac{\sqrt{3}m}{2+m}$ and y-intercept of

$\frac{\sqrt{3}}{2+m}$. QED.

Similarly, one may show

Theorem: For any real *a*, *b*, *c*,

$$z = au + bv + c \tag{12}$$

is the equation of a plane.

Proof:

$$\begin{aligned} z &= ax - \frac{y}{\sqrt{3}} + 2b\frac{y}{\sqrt{3}} + c \\ &= ax + \frac{2b-a}{\sqrt{3}}y + c \end{aligned},$$

which is the equation of a plane with $\frac{\partial z}{\partial x} = a$ and $\frac{\partial z}{\partial y} = \frac{2b-a}{\sqrt{3}}$.

Corollary 1: Two lines represented by the equations $\begin{array}{l} v = m_1u + b_1 \\ v = m_2u + b_2 \end{array}$ are parallel if $m_1 = m_2$, and perpendicular if

$2m_1m_2 = -(2 + m_1 + m_2)$.

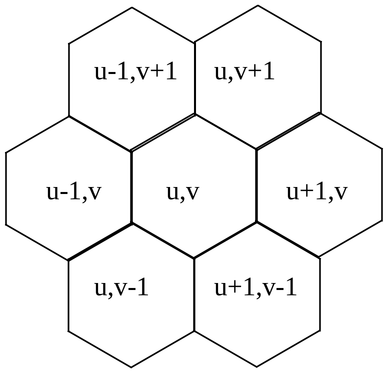
Proof: For the slopes to be equal, we must have $\frac{m_1}{2+m_1} = \frac{m_2}{2+m_2}$ which is only satisfied if $m_1 = m_2$.

For the lines to be perpendicular, we require that the slope of the first line be equal to the negative of the inverse of the slope of the other line, $\frac{m_1}{2+m_1} = -\frac{(2+m_2)}{m_2}$, which simplifies to the result presented. QED

3. IMAGE PROCESSING IN h2 COORDINATES

3.1 Identifying the Neighbors of a Pixel

Given a pixel with coordinates *u*,*v* (assumed integer), the coordinates of the neighbors are illustrated in Fig. 3.



3. The neighborhood of a pixel is absolutely symmetric, with no ambiguity of 4- or 8-neighborhoods.

We observe that no “if” statements are required to determine if the center pixel is on an even- or odd-numbered row. The following loop efficiently accesses all six neighbors of pixel, *u*, *v*:

```
for i = 1 to 6
    nu = u + ou[i]
    nv = v + ov[i]
    value = image[nu][nv]
```

After the arrays *ou* and *ov* have been initialized to *ou* = {-1, -1, 0, 1, 1, 0}, and *ov* = {0, 1, 1, 0, -1, -1}.

We note that this method of accessing the neighbors of a pixel is also useful in rectangular grids for 8-neighbors and is more efficient than the doubly indexed loop:

```
for i = -1 to 1
for j = -1 to 1
    if ((i != 0) or (j !=0))
    value = image[u+i][v+j]
```

It is also interesting to note that *ou* and *ov* are circulations of each other.

3.2 Finding Image Gradients

Rather than approximating orthogonal (*x*-*y*) derivatives[19], we derive partial derivatives directly in the h2 system. To find the gradient of intensity in an image, we will fit a plane to the data in a small neighborhood. This plane will be represented in the form of Eq. (12). We then take partials with respect to *u* and *v* to find the gradients of intensity in those corresponding directions. We choose a neighborhood of 6 points, surrounding a central point, and fit the plane to them. Define the set of data points as $z_{i,}$ (i = 1,...6). Then Eq. (13) represents the error (*E*) in fitting these six points to a plane parameterized by *a*, *b*, and *c*.

$$E^2 = \sum_{i=1}^{6} (z_i - (au_i + bv_i + c))^2 \tag{13}$$

In order to represent *E* in a form that will be easy to differentiate, we will reformulate the argument of the summation using matrix notation. Define vectors $A = [a, b, c]^T$ and $Z = [u, v, 1]^T$. Then *E* may be written using

$$E^2 = \sum_{i=1}^{6} (z_i - A^T Z_i)^2 \tag{14}$$

$$= \sum_{i=1}^{6} (z_i^2 - 2z_iA^TZ_i + A^TZ_iA^TZ_i) \tag{15}$$

First, we observe that $A^TZ = Z^TA$, and rewrite Eq. (15), temporarily dropping the limits on the summation to make the typing easier:

$$= \sum z_i^2 - 2A^T\sum z_iZ_i + A^T(\sum Z_iZ_i^T)A \tag{16}$$

The term in parentheses in equation Eq. (16) is known as the "scatter matrix", the collection of locations of points at which data exists. We denote this matrix with the symbol S. In order to find the value of vector A which minimizes E^2, we take the partial derivative with respect to A:

$$\frac{\partial}{\partial A}E^2 = -2\sum z_iZ_i + 2SA\,. \tag{17}$$

By evaluating S, we find $S = \begin{bmatrix} \sum u_i^2 & \sum u_iv_i & \sum u_i \\ \sum u_iv_i & \sum v_i^2 & \sum v_i \\ \sum u_i & \sum v_i & \sum 1 \end{bmatrix} = \begin{bmatrix} 4 & -2 & 0 \\ -2 & 4 & 0 \\ 0 & 0 & 6 \end{bmatrix}$. (One finds the numerical values by summing the u and v coordinates over each of the pixels in the neighborhood, as illustrated in Fig. 4., assuming the center pixel is at location [0,0].)

Define $\begin{matrix} \sum z_iu_i \equiv \Upsilon_u \\ \sum z_iv_i \equiv \Upsilon_v \end{matrix}$, and setting the partial derivative of Eq. (17) produces a pair of simultaneous equations,

$$\begin{aligned} 4a - 2b &= \Upsilon_u \\ -4a + 8b &= 2\Upsilon_v \end{aligned}, \tag{18}$$

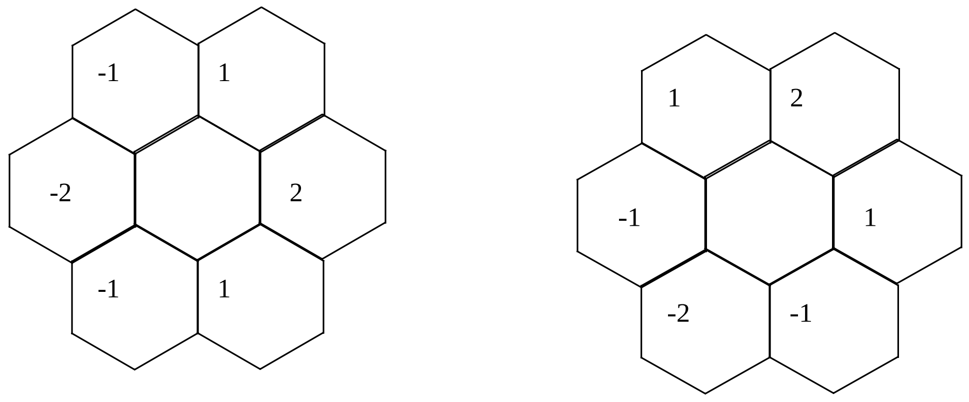
with solution

$$b = \frac{1}{6}(2\Upsilon_v + \Upsilon_u) \tag{19}$$

and similarly,

$$a = \frac{1}{6}(\Upsilon_v + 2\Upsilon_u) \tag{20}$$

Substituting actual values of u and v at each pixel in the six pixel neighborhood, we find that since the gradient in the u direction is a and the gradient in the v direction is b, these numbers can be computed using the convolutions shown below.



4. The convolution kernel used to estimate the gradient of brightness in the ***u*** direction, and in the ***v*** direction

Fig. 5. shows the result of using the identical approach on a rectangular grid, fitting a plane to a 3x3 neighborhood, to produce the convolution kernel

-1	0	1
-1	0	1
-1	0	1

5. The convolution kernel which results from taking the partial derivative with respect to x of the best fitting plane. Note the similarity to the Sobel operator, but the lack of center weighting.

It is interesting to note that the center weighting arises naturally in h2 coordinates, but must be imposed in *x-y* coordinates by making sensitivity arguments, or by pre-blurring with a Gaussian.

3.3 Variable Conductance Diffusion in h2

Variable Conductance Diffusion (VCD)[16,17,20] is a powerful method of image feature extraction in which blurring is allowed to occur except at edges. The term “edge” may be liberally interpreted to mean any image feature which is of interest. For example, Whitaker[18] operates on the gradient of the image rather than the image itself and allows smoothing to occur except where ridges (sharp changes of gradient direction) occur. Such an operation is not a restoration by any means, since most of the information in the original image is lost. It is, however, a very robust way to extract a sort of central axis of an object in a gray scale image.

VCD operates by simulating the diffusion equation

$$\frac{\partial f_i}{\partial t} = \nabla \cdot (c_i \cdot \nabla_i f) \tag{21}$$

where *t* is time. The diffusion equation models the flow of some quantity (the most commonly used example is heat) through a material with conductivity *c*. In the case of VCD, the conductance becomes a function of the spatial coordinates, in this instance parameterized by *i*, and in particular a property of the local image intensities themselves.

To smooth except at edges, we let c_i be small if *i* is an edge pixel, i.e., if a selected image property is locally nonuniform. If c_i is small, in the heat transfer analogy, little heat flows; and in the image, little smoothing occurs. If, on the other hand, c_i is large, then much smoothing is allowed in the vicinity of pixel *i*. VCD then, implements an operation which after repetition produces a nearly piecewise uniform result.

With this understanding, we proceed to analyze the behavior of diffusion algorithms in h2 coordinates. We will begin with the form of the diffusion equation from x-y coordinates, and convert it to h2.

We begin by taking some derivatives of the quantities in Eq. (3),

$$\frac{\partial u}{\partial x} = 1 \qquad \frac{\partial u}{\partial y} = -\frac{1}{\sqrt{3}} \qquad \frac{\partial v}{\partial x} = 0 \qquad \frac{\partial v}{\partial y} = \frac{2}{\sqrt{3}} \tag{22}$$

For the gradient operator of Eq. (21), we use a vector operator definition:

$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$. Let $G_{uv} = \begin{bmatrix} G_u & G_v \end{bmatrix}^T$ be the gradient vector measured with the convolution kernels given above. Using Eq. (2) to project this vector onto the *x*-*y* axis to obtain $G_{xy} = \begin{bmatrix} \left(G_u + \frac{G_v}{2}\right) & \left(\frac{\sqrt{3}}{2} G_v\right) \end{bmatrix}^T$. With this notation, Eq. (21) becomes

$$\frac{\partial f}{\partial t} = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} c\left(G_u + \frac{G_v}{2}\right) \\ c\frac{\sqrt{3}}{2} G_v \end{bmatrix} \tag{23}$$

$$= \frac{\partial}{\partial x}\left(cG_u + \frac{G_v}{2}\right) + \frac{\partial}{\partial y} c\frac{\sqrt{3}}{2} G_v \tag{24}$$

Using the chain rule to represent the derivatives with respect to *x* and *y* in terms of derivatives in *u* and *v*,

$$= \frac{\partial}{\partial u}\left(cG_u + \frac{cG_v}{2}\right)\frac{\partial u}{\partial x} + \frac{\partial}{\partial v}\left(cG_u + \frac{cG_v}{2}\right)\frac{\partial v}{\partial x} + \frac{\partial}{\partial u}\left(\frac{\sqrt{3}}{2} G_v\right)\frac{\partial u}{\partial y} + \frac{\partial}{\partial v}\left(\frac{\sqrt{3}}{2} G_v\right)\frac{\partial v}{\partial y} \tag{25}$$

Now, substitute the values from Eq. (22) and simplify to find

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial u}[cG_u] + \frac{\partial}{\partial v}[cG_v] \tag{26}$$

We have therefore proven the surprising result that

Theorem: The diffusion equation in h2 coordinates is the same as that in *x*-*y* coordinates.

3.4 Storing Data in the h2 Coordinate Representation

It would appear that storing data in the h2 representation is inefficient, as the alignment of the data does not naturally align with a rectangular storage of data in an array in memory. To see this, consider Fig. 6. where the *u*,*v* coordinates are illustrated. The origin of the numbering system has been chosen to ensure that no negative coordinates can occur. With a 7x9 array of pixels such as this one, the numbering goes from *u* = 0 to *u* = 11, and the numbering on v varies from 0 to 7. Observe that there is no pixel [11,2]... [11,6]. Considering memory allocation in a naive way, one would declare a memory array with the horizontal axis, *u*, ranged from 0 to 11, and the *v* from 0 to 6, a total of 84 pixels, in which only 63 pixels are used. In general, for an image with *nr* rows and *nc* columns, such a memory allocation scheme would have a utilization factor of $\frac{\text{nr} \times \text{nc}}{\text{nr} \times \left(\text{nc} + \frac{\text{nr}}{2}\right)}$.

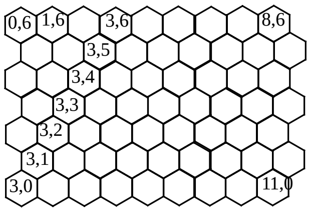
For a square array, this simplifies to a ratio of memory used to memory allocated by 2/3.

However, a simple and efficient memory allocation scheme can be designed to utilize all the memory allocated, and do so even faster than doubly indexed arrays. To allocate an h2 array with *nr* rows and *nc* columns, first, allocate a 1-*d* vector to store the image:

```
pixel image[nr*nc];
```

Then, allocate a vector of pointers to pixels, one to hold the address of the start of each line.

```
pixptr fistpix[nr];
```



6. A numbering scheme for u and v which does not introduce negative indices.

A simple loop initializes each entry of fistpix to contain the address of the first pixel on each line. Then, to access any pixel, say at row *r* and column *c*, construct a pointer with a lookup and an add:

```
ptr = fistpix[r] + c;
```

and access the image using this pointer

```
value = *ptr;
```

We note that this is more efficient than

```
value = image[r][c],
```

as the latter includes a multiply instruction missing from the former scheme. This approach to accessing image data quickly has been used in many applications in *x*-*y* organized images, as well as by the authors of *Numerical Recipes in C*, to access the elements of arrays.

4. CONCLUSIONS

A hexagonal coordinate system (h2) is created and analyzed. We derive several basic properties of this coordinate system and compare them with those in *x*-*y* coordinates. We show that the calculations required to perform most of these operations are straightforward and easily related to their Cartesian equivalents.

Two popular image processing algorithms are formulated in the h2 coordinates: convolution kernel derivation and variable conductance diffusion. It is shown that the use of simple minimum-square-error approach to the derivation of convolution kernels results in symmetric center-weighted operators; also when formulated in terms of *u* and *v*, diffusion equation has exactly the same form as it does when formulated in *x* and *y*.

The computational complexity of image processing algorithms in *u*-*v* coordinates is compared to that in *x*-*y*, and the new algorithms are shown to compare favorably, while avoiding connectivity ambiguities.

REFERENCES

1. H. Longbotham, T. Arnow, “Comparison of rectangular versus hexagonal sensor arrays for learning in neural networks”, Department of Defense grant: DOD 26-0401-2103, 1990.
2. R. Stimets, H. Tang, “Procedures for rapid pattern recognition of three-dimensional objects using parallel processing on a hexagonal pixel grid”, SPIE, v2666, pp2-8.
3. A. Rosenfeld, “Connectivity in Digital Pictures,” *Journal of the ACM*, Vol 17, pp 146-160, 1970.

4. A. Rosenfeld, “Adjacency in Digital Pictures,” *Informat. Contr.* Vol 26, pp 24-33, 1974.
5. G. Herman, “Boundaries in Digital Spaces: Basic Theory”, TR MIPG 186, Medical Image Processing Group, University of Pennsylvania, August 1992.
6. Mead, Carver, *Analog VLSI and Neural systems*, Addison-Wesley, 1989.
7. Delbrueck, Tobias, “Silicon retina with correlation-based, velocity-tuned pixels”, *IEEE Transactions on Neural Networks*, v4, n3, May 1993, pp529-541.
8. D. Eberly, “Hexagonal tessellations in image algebra”, *Image Algebra and Morphological Image Processing*, SPIE, v1350, 1990, pp25-30.
9. R.M. Mersereau, “The processing of hexagonally sampled two-dimensional signals”, *Proceedings of the IEEE*, 67, no. 6, pp930-949, 1979.
10. R.L. Stevenson, G.R. Arce, “Binary display of hexagonally sampled continuous-tone images”, *Journal of the Optical Society of America*, vol. 2, no. 7, pp.1009-1013, July, 1985.
11. M.J.E. Golay, “Hexagonal parallel pattern transformations”, *IEEE Trans. on Computers*, vol. c-18, no. 8, pp733-740, August, 1969.
12. J.A. Cox, “Point-source location using hexagonal detector arrays”, *Opt. Eng*., vol. 26, pp69-74, 1987.
13. B. Carstens, M. Quinn, “Hexagonal domain transform for shape analysis”, *Intelligent Robots and Computer Vision X: Algorithms and Techniques*, SPIE, v1607, 1991, pp197-205.
14. A.B. Watson, A.J. Ahumada, Jr., “A hexagonal orthogonal-oriented pyramid as a model of image representation in visual cortex,” *IEEE Biomedical Engineering*, v.36, no.1, pp97-105, 1989.
15. Jan W. van Roessel, “Conversion of Cartesian coordinates from and to generalized balanced ternary addresses,” *Photogrammetric Engineering and Remote Sensing*, 54, no. 11, pp1565-1570, 1988.
16. K.N. Nordstrom, “Biased anisotropic diffusion: a unified regularization and diffusion approach to edge detection”, *Image and Vision Computing*, Vol 8, no 4, Nov 1990.
17. P. Perona, J. Malik, “Scale-space and Edge Detection using Anisotropic Diffusion”, *IEEE Transactions on PAMI*, 12, 429-439, 1990.
18. R. Whitaker, “Geometry-limited diffusion in the characterization of geometric patches in images”, TR91-039, Dept. of Computer Science, UNC, Chapel Hill, 1991.
19. S. Abu-Bakar, R.J. Green, “Detection of edges based on hexagonal pixel formats”, *Proceedings of ICSP*, 1996, pp1114-1117.
20. W. Snyder, Y. Han, G. Bilbro, R. Whitaker, and S. Pizer, “Image relaxation, restoration, and feature extraction”, *IEEE PAMI*, vol. 16, no. 12, December, 1994.
21. E. Luczak, A. Rosenfeld, “Distance on a hexagonal grid”, IEEE Transactions on Computers, vol. C-25, no. 5, pp532-533, May, 1976.
22. S. Bell, H. Fred, D. Mason, “A digital geometry for hexagonal pixels”, Image and Vision Computing, vol. 7, no. 3, pp194-204, August, 1989.
23. M. Clements, K. Vichienchom, *et. al*. “An implantable neuro-stimulator device for a retinal prothesis”, To Appear on *ISSCC Digest of Technical Papers*, 1999.