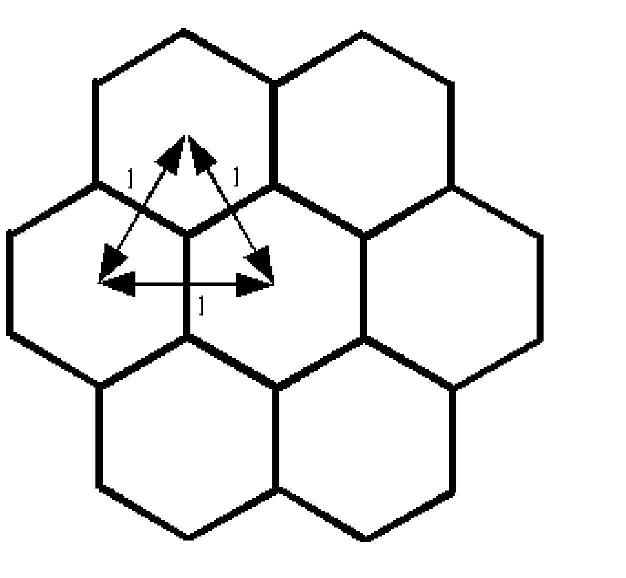
Hexagonal Coordinate Systems

A hexagonal coordinate system is simply a system which rejects the common square lattice upon which most images are mapped and described with in favour of a hexagonal lattice.

Why use a Hexagonal Coordinate System?

There are a number of reasons why hexagon-based descriptions of images are considered useful. One of the major advantages of hexagonal systems lies in the consistent connectivity of its constituent hexagons. In a common rectangular coordinate system, the distance between neighbouring points differs, dependant on whether the neighbour is directly adjacent (in which case distance is 1 unit) or diagonal (where distance is square root of 2 units). Under the hexagonal system, all points are equidistant, at 1 unit. This, along with the simple fact that hexagons are 'rounder' than squares, tend to make the presentation of features in images such as curves more consistent than under normal mappings, which aids in such operations as edge detection. For images rendered under a hexagonal lattice, the constituent points are more densely packed than an equivalent rectangular rendering of roughly the same apparent size – which means more points to perfom visual processing on. Finally, it has been shown that mathematically, many operations of interest to the field of vision can be more successful when dealing with a hexagonal lattice – such as edge detection [1] and shape extraction [2] over images showing complex objects with rounded features.

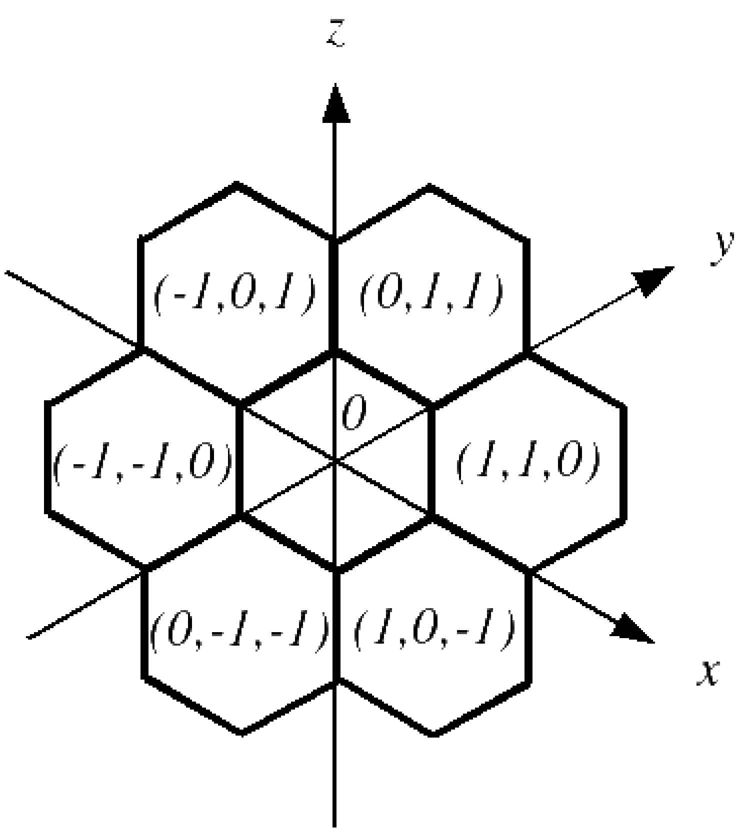


A hexagon, surrounded by its equidistant neighbours.

From the perspective of computer vision, hexagonal coordinate systems hold particular appeal in that they more closely resemble the layout of photo-receptors in the human retina; research has been done that suggests that the simulation of at least some of the capabilities possessed by the human eye and the visual processing areas of the brain can be more easily executed on images that are laid out on a hexagonal lattice, including simulating the human eye's saccades [3] when focusing on aspects of an image.

Practicalities of Using a Hexagonal Coordinate System

There are generally four major considerations that must be pondered upon when using a hexagonal coordinate system [1]:



Example of using 3-element coordinate system to index hexagonal pixels

- *Image Conversion* – Hardware capable of capturing images from the real world directly onto a hexagonal lattice is highly specialist, and so not generally available for use. Therefore, efficient means of converting a standard square-latticed image into a hexagonal one is required before any processing can be performed.
- *Addressing and Storage* – Any manipulations performed on images must be able to index and access individual pixels (in this case hexagons rather than squares), and any image in hexagonal form should be *storable* in hexagonal form (otherwise image conversion would have to be performed every time the image was accessed). Moreover, an indexing system that is simple to follow and makes the arithmetic of certain functions simpler would be very valuable.
- *Image Processing Operations* – In order to make effective use of the hexagonal coordinate system, operations must be designed or be converted that are geared to exploit the strengths of the system, and particularly the strengths of the addressing system used for indexing and storage.
- *Image Display* – As with actually obtaining the image in the first place, display devices in general do not use hexagonal lattices. Therefore the converted image must be returned to a form that can be sent on to an output device (whether a monitor, a printer or some other entity) with the resultant display appearing in natural proportions and scale. The exact nature of this conversion is dependant on the indexing method used. This could be a simple reversion of the original conversion process, or be a more considerable convolution.

Of these four considerations, the first two are examined in a little more depth below; the third requires a separate article for just about every operation devised, and the fourth cannot be elaborated on without going into specifics, except to say that it is the logical reverse of the first consideration.

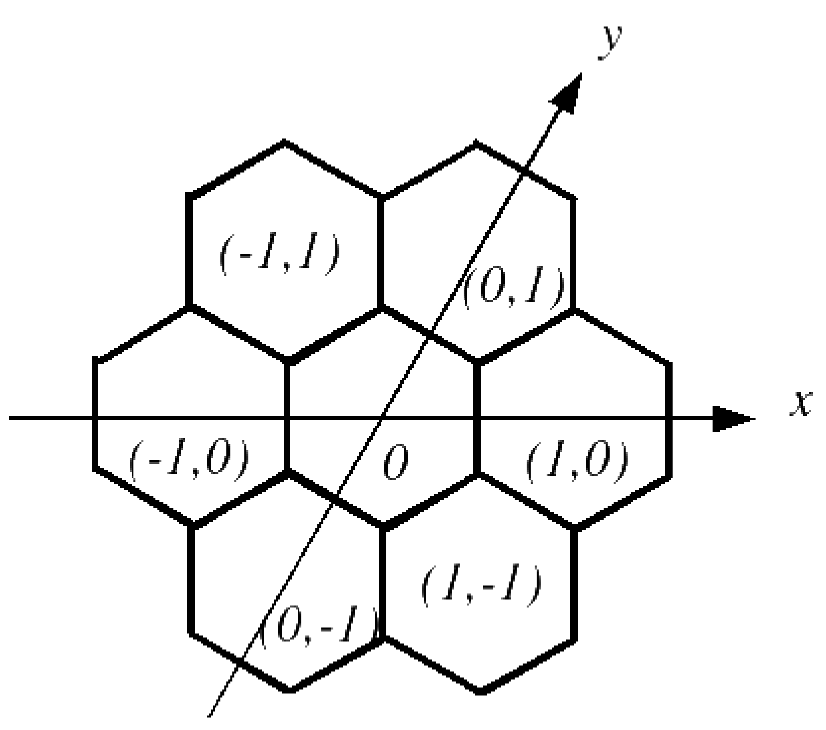
Image Conversion

One method of converting from square lattice to hexagonal lattice is *image resampling*. Hexagonal points are generated by mapping sampled points across an appropriate sampling lattice. As conversions generally rely on maintaining the scale of one of the original cartesian axes, hexagonal lattices are usually denser than square equivalents. This necessitates the extrapolation of additional points. There are many different methods, many of which address this problem, for image resampling, but they are for the most part beyond the scope of this specific article (see [4] for some examples of the issues involved); however a basic sampling lattice can be generated using the basis vectors ($B = \{b_1, b_2\}$) below:

$$B = \{(1,0), (\frac{1}{2}, \frac{\sqrt{3}}{2})\}$$

where the hexagonal lattice generated is the interconnection of the set of points L :

$$L \equiv \{n_1 b_1 + n_2 b_2 : n_i \in \mathbb{Z}, i \equiv 1,2\}$$



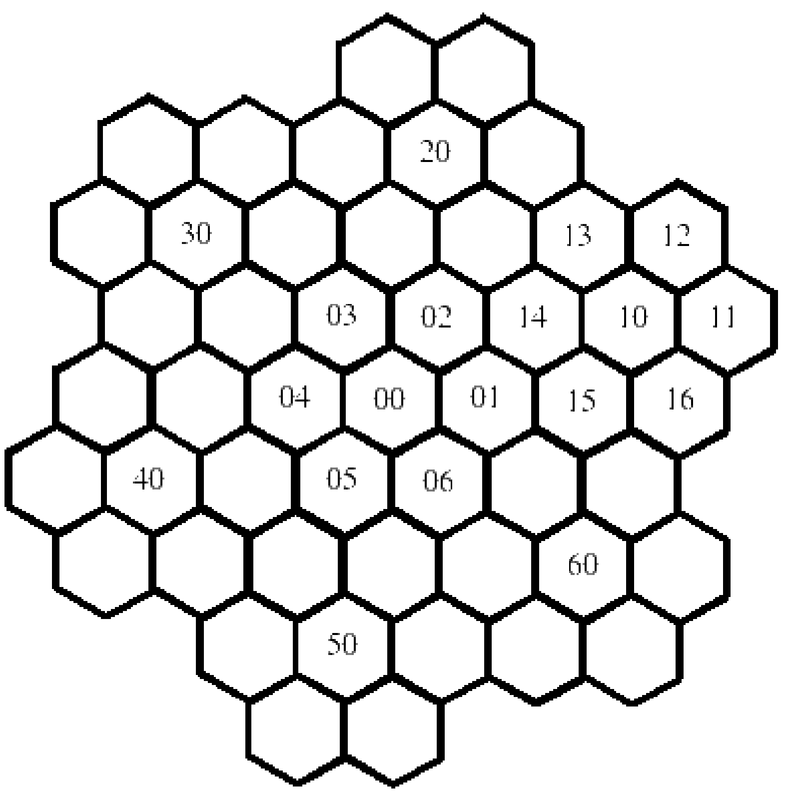
Example of using a skewed-axis coordinate system to index hexagonal pixels

Addressing and Storage

There are numerous ways to store and address a hexagonally-represented image. A simple approach is to use a 3-element coordinate system, where the 3 axes are aligned with the three axes of symmetry present in a hexagon within the lattice. Another method is to used a skewed (x or y) axis.

One particularly interesting system is the layered system used in [1, 2, 3]. In all those papers, a 'tile' in layer 0 is defined by a single hexagon. A layer 1 tile is a hexagon and all 6 of its surrounding neighbours. A layer 2 tile is a layer 1 tile and the size surrounding layer 1 tiles. This continues on for L layers (note that this is not the same L with which a hexagonal lattice was defined). It is clear that there are 7^L hexagons in a layer L tile. It is possible to uniquely index every hexagon in that tile, using an L-digit base 7 number where each digit indicates the specific tile of the layer below in the overarching super-tile, with 0 indicating the centre tile, and 1-6 indicating tiles counted anticlockwise around the centre tile. The diagram to the right demonstrates this. Because indices consist of a single number, an image can be stored as a vector.

This particular method has been shown to be quite useful when certain operations are to be performed on an image so encoded; work has been done by the designers of this method at the University of Auckland on many of things referred top earlier in this article.



A Layer 2 super-tile, with a selection of the hexagonal pixel indices shown

Issues with Hexagonal Coordinate Systems

There are some problems with hexagonal coordinate systems however. One issue is that people are very used to the traditional square lattice. Reasoning in hexes can seem unnatural and therefore a little difficult. While it could be argued that people can become used to it if they have to, it is still the case that they will be naturallly inclined towards reasoning with the traditional cartesian coordinate system by default, with hexagonal systems merely a secondary choice.

The lack of input devices that map onto hexagonal lattices, and the lack of output devices that display as such is also an obstacle. The necessity of converting from squares to hexagons and back again detracts from the usefulness of operating on hexagonal lattices. As such lattices are denser than equivalent square lattices of the same apparent size, unless images are fed in at a deliberately higher resolution than is to be operated on, converted images shall have to extrapolate some pixel locations (which is generally less desirable than having all pixels provided directly from a source). The conversion back to square lattices would collapse some pixel locations into one another, which results in loss of apparent detail (which could result in a lower quality image than the one that was originally fed in).

If one seeks to use hexagonal coordinate systems in their own vision work, then they should first determine whether these problems are outweighed by the inherent advantages of operating with hexagons.

Citations

[1] Lee Middleton, Jayanthi Sivaswamy; Edge detection in a hexagonal-image processing framework; Image and Vision Computing 19 (2201) 1071 – 1081. URL: http://www.ecs.soton.ac.uk/~ljm/archive/a_middleton2001edge.pdf

[2] L Middleton, G Coghill, J Sivaswamy; Shape extraction in a hexagonal-image processing framework. URL: http://www.ecs.soton.ac.uk/~ljm/archive/c_middleton2000shape.pdf

[3] L Middleton, G Coghill, J Sivaswamy; Saccadic exploration using a hexagonal retina. URL: http://www.ecs.soton.ac.uk/~ljm/archive/c_middleton2000sacc.pdf

[4] Dimitri Van De Ville, Rik Van de Walle, Wilfried Philips, Ignace Lemahieu; Image resampling between orthoginal and hexagonal Lattices. URL: http://escher.elis.rug.ac.be/publ/Edocs/DOC/P102_083.pdf