Hexagonal Grid Image Processing Algorithms in Neural Vision and Prediction

NUNO ALBUQUERQUE SOUSA
Faculdade de Engenharia da Universidade do Porto
June 6, 2014

Abstract

In the field of neural prediction of the visual cortex, it is known that 97%[1]of receptive field neurons are described by the response to a 3D Gabor Filter (2D over time). By replacing the stimulus representation - a 2D image with square pixels, with an hexagonal system, some advantages are theoretically possible: better computational speeds, lower memory footprint, and more accurate depiction of the biological phenomena taking place. In this work the framework for handling the algorithms and operations typical in the neural prediction work flow are established, and tested for practicality.

Keywords: Neural Prediction, Hexagonal Image Processing, Visual Cortex, Receptive Field, Honeycomb Scheme, Hex Gabor Filtering.

Introduction

This work is the final report of the UC Investigation Techniques, part of the curricular program of the Doctoral Program in Biomedical Engineering. The purpose of this UC is to allow the graduate students to familiarize and gain competence with a given technique or techniques. The focus of the work presented, a technique which is perceived to be of great potential in the future PhD thesis, is on the application of an hexagonal tessellation instead of a quadratic in the representation of the visual stimulus. The advantages of said change are explored in the Motivations subsection. The broad theme of neural prediction of the visual cortex, is a well established field in itself, and has been the author's theme of thesis for his MSc. The algorithms for classification and prediction are in the author opinion reaching their breaking point in terms of improvement. The focus of the new techniques proposed is the first part of the neural prediction work-flow. The process of neural prediction has usually three stages:

1. Image Analysis and "Feature Extraction"[1]
2. Classification or Regression
3. Non Linear Filtering

It's in the first stage where the substitution from a square grid to a hexagonal one is to occur. Then again the algorithms used in the image analysis are very well established. However the properties of the hexagonal grid may be of some use to gain some ground on this field.

Motivations

The advantages of an hexagonal grid to describe the stimulus data (a movie for instance) over the typical square lattice are:

- The hexagonal pixel is very superior in efficiency, for describing data with a given resolution.
- The neighborhood of an hexagonal pixel grows at a decreased range: the neighborhood of square pixels grows at 8 per growth, while hexagons grow at 6 per growth(see figure 1).
- Smaller Quantetization Error [2] - Again, this is useful when working with downsampled images for computational reasons as is today always[2] the case.
- Greater Angular Resolution [2] - In neural prediction 12 directions are usually used, which can be achieved with a 3 layer hex filter, saving computational resources.
- Connectivity Definition [3] - There are neighborhood variants, of some filters which vary between 4 and 8 pixel connectivity in rectangular grids, this eliminates the problem of choice.
- Equidistance of neighborhood [3] - Better representation of data.
- Higher Symmetry [4] - No foreseeable advantage.
- Most filters are somewhat circular, or at least directional, and given that the neighborhood also grows with some circularity, hexagonal filtering is computationally more efficient.
- Fewer Sampling Points [5] - Computational Efficiency

[1]The features are usually the magnitude of the response of a region of the image to a filter
[2]Today's algorithms are so heavy, and the Gabor Wavelet Processing so time consuming, there is hardly any chance of using original size stimulus files

- The packing of hexagons is tighter than that of squares, resulting in a more approximate simulation of the reality of the retina (see figure 2).

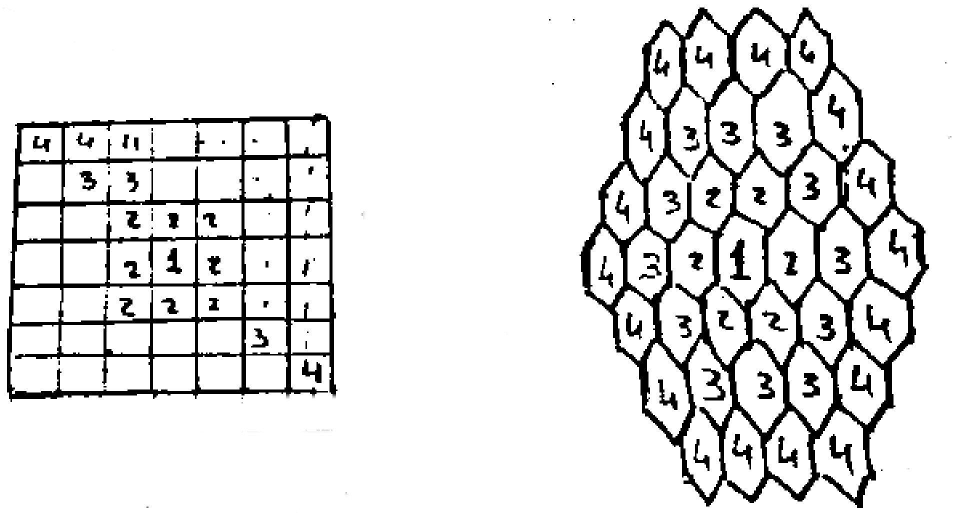

Figure 1: Here shown are the various layers of pixels of a square grid (left) and an hexagonal grid (right). The original pixel constitutes the 1st layer, and 4 layers are shown for each grid. Te hexagonal system grows 6 pixels per layer, the square system grows 8. This is of particular importance for the represented size, as it is a very common size for filters-Consider that the average stimulus size is 16x16, 4x4 is pretty sufficient for a gabor kernel [Original Image]

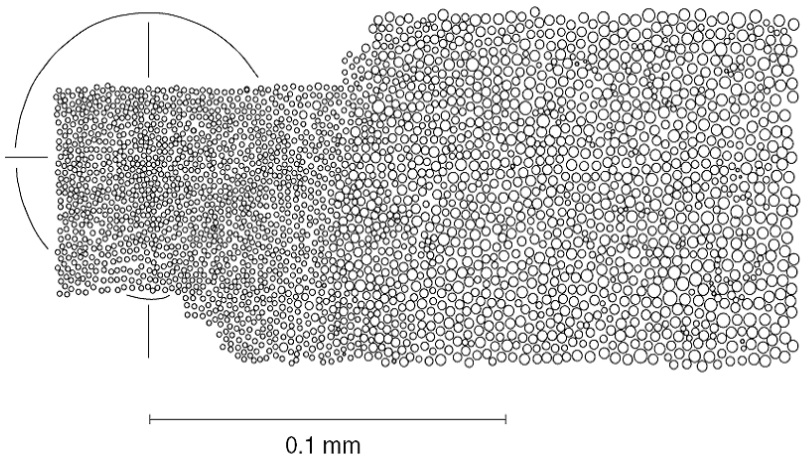

Figure 2: The dispersion of cones and rod cells across the retina. Showing a diminution of concentration as we move away from the fovea. Notice that the receptive fields of the retina exhibit a "packed circles" configuration, which is better approximated by an honeycomb structure than a square grid.[6]

Ultimatly these advantages will translate in the final result being slightly better, either because of the advantages conferred by the hexagonal grid directly, or because of the computational gain in speed, of having more information processed in the same time. Allowing for larger parameter sweeps, and larger filter banks.

The main disadvantage being that there is virtually no established resources for hexagonal grid systems. While matlab code, tools, and toolboxes for standard square image processing are abundant and well established, there isn't a single available code snippet about hexagonal processing, outside of a couple of out of public domain books, (which may contain or not, being they are inaccessible). Even though for now some algorithms have been ported into code, one thing to recall, is that most matrix operations in matlab are very well optimized in assembly code, and are thus more optimized than their high level code hexagonal counterparts. This can be partly evened out by the usage of matrix computation whenever possible, and making use of native matlab instructions when possible, but as we'll see in hexagonal addressing, sometimes the best scenario cannot be achieved with existing tools. A "must" future work would be to implement at least in binary code an efficient convolution tool in Spiral Addressing.

State of the Art

The idea of applying hexagonal grid to the mechanisms of vision processing is not entirely novel: several areas of computer vision are already well familiar with the idea, so there is mathematical background into hexagonal lattices sufficient for our needs. The application of said grids into cognitive neuroscience is novel, however there is a 2012 PhD Thesis which handles a similar area on the topic[7][3]. Applications of hexagonal grid in biomedical engineering include, in a different aspect of image processing the reconstruction of tomography images into a 3D model[8]. The advantages of the hexagon are paragon when solving the complicated equations of image reconstruction.

Perhaps of singular importance would be the application of gabor wavelets in hexagonal grids. The work on this particular subject is existent, and given that the response to gabor filter banks is the current state of the art approach for neural prediction, the elaboration of 3D gabor wavelets hexagonally will be the ultimate goal in this work.

Hexagonal Grid Framework

One of the conclusions that this work must reach is wether or not Hexagonal Grid is best suited for the given applications.

First, let us quantify how much better the hexagon is compared to the square. Assuming that the spacial resolution of an image is the highest distance which is possible to discriminate two separate areas, the square pixel and the haxagon pixel define spatially, the circunference which inscribes them - as seen in figure 3 - we can calculate the efficiency of the two.

The square pixel's resolution is determined by its diagonal. Let us compute it's coverage of the area it represents:

$$\varepsilon_{square} = \frac{A_{pixel}}{A_{circle}} = \frac{\frac{r}{\sqrt{2}}^{2}\pi}{r^2\pi} = \frac{1}{\sqrt{2}} \tag{1}$$

Where A_{pixel} stands for the area covered by the pixel, r, the radius of the inscribing circle

$$A_{pixel} = \frac{r^2 6 sin(\pi/3)}{2} = 3r^2 sin(\pi/3) \tag{2}$$

$$\varepsilon_{hexagon} = \frac{A_{pixel}}{A_{circle}} = \frac{3r^2 sin(\pi/3)}{r^2\pi} = \frac{3 sin(\pi/3)}{\pi} \tag{3}$$

[3]It does elaborate on hexagonal gabor filtering, yet not applied to neural prediction

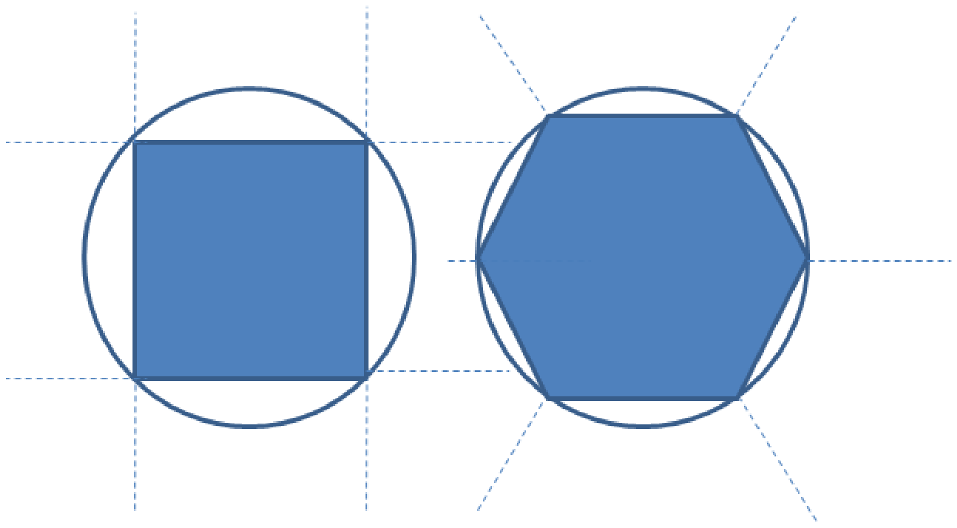

Figure 3: In the figure the inscribing spacial resolution provided by each pixel can be seen. If one imagines the pixel as being part of a lattice as suggested by the doted lines, the zone inside the circle and outside the pixel can be perceived as the waste - the pixel's information if being reflected on other pixels[Original Image].

From which we can establish:

$$\frac{\varepsilon_{hex}}{\varepsilon_{square}} = 1.1695 \tag{4}$$

From 4 we can see that the hexagon contains an extra 16.95% coverage of the defining resolution in comparison with the square.

Curiously, this simple calculation is never presented in the literature, however a spectral assessment by fourier transform was established in [5]. The conclusion the author reaches is that it takes 13.4% less points to have a spectrally similar quality on the image. The method through which this was performed is unavailable to us as we do not possess the full article. Yet it implies DFT calculation. Alternatively we can try and reach this conclusion ourselves:

Given sampling provided by a sampling matrix (directions of sampling):

$$l_{eigenvector} = \begin{bmatrix} E_1 & 0 \\ 0 & E_2 \end{bmatrix} \tag{5}$$

Is in the case of hexagonal sampling (basic trignometry should reach the orientation vectors):

$$l_{eigenvector} = \begin{bmatrix} E_1 & E_1/2 \\ 0 & \sqrt{3}E_1/2 \end{bmatrix} \tag{6}$$

In equation 6 the second column are the x and y projections of the second vector. (Imagine it pointing from the first hexagon onto the top right neighbor for visualization)

From this we can establish the sampling in the spectral domain:

$$L^T l = I \tag{7}$$

Where I is the identity matrix.

Simple algebra returns:

$$L_{hex} = \begin{bmatrix} \frac{2\pi}{E_1} & 0 \\ -\frac{2\pi}{\sqrt{3}E_1} & \frac{4\pi}{\sqrt{3}E_1} \end{bmatrix} \tag{8}$$

Knowing that the maximum frequency those pixels can define is the circle that inscribes them and touches them at the diagonal, we can equate this circle in terms of spectral frequency.

$$\frac{4\pi}{\sqrt{3}E_1} = 2W \tag{9}$$

Bringing the equality back to the spatial domain yelds:

$$l_{quadrado} = \begin{bmatrix} \frac{\pi}{W} & 0 \\ 0 & \frac{\pi}{W} \end{bmatrix} \tag{10}$$

And for the hexagon:

$$l_{hexagono} = \begin{bmatrix} \frac{2\pi}{\sqrt{3}W} & \frac{\pi}{\sqrt{3}W} \\ 0 & \frac{\pi}{W} \end{bmatrix} \tag{11}$$

In this spatial form, we know that the necessary sizes for this spatial representation of the frequency W, therefor, we need only find the ratio between the two magnitudes of the vectors, which may be found by the determinant of their matrices. By computing the ratio between the two, we can present:

$$\frac{|det l_{quadrado}|}{|det l_{hexagono}|} = \frac{\sqrt{3}}{2} = 0.87 \tag{12}$$

There are needed only 87% of the samples in an hexagon grid to discriminate with the same accuracy what the full 100% grid of squares would signify.

The computation based on the area, and the spectral area should be related, though the author is not entirely sure how.

Adressing Systems for Hexagonal Grids

Organization in rectangular pixel arrays are very simple and intuitive, yet, when handling hexagons, there is more than one obvious solution.

In figure 4 we can see one of the simplest methods, which is using the two directing vectors of the hexagonal system as the axis. This adressing scheme is useful for referencing filters, yet no so much for referencing an entire image in hexagonal grid, as either part of the indexes will be of no consequence (as in the oblique axis shown in the figure) or part of the top left corner will have negative values. This is perfectly fine and can be dealt with; however, better schemes exist to reference the whole image: Offset Adressing.

Offset Adressing as seen in figure 5 is pretty much the usual way to reference square bitmaps, with offset even lines (or columns, lines in our preference) which result in an image size directly mapped to relevant coordinates in the Cartesian format.

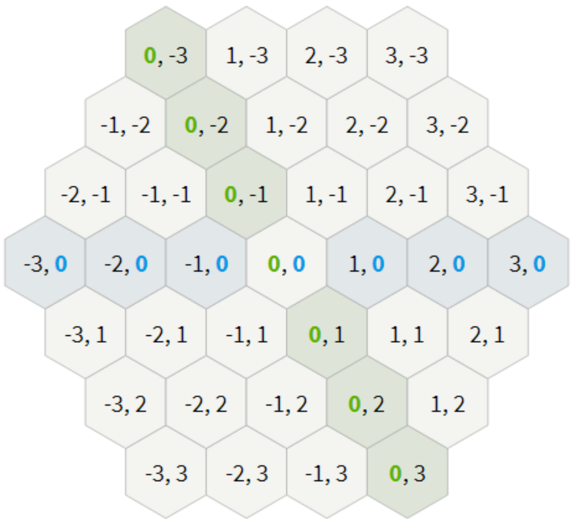

Figure 4: Orthogonal Addressing -This 2D Addressing method is one of the favorites. The two axis form a simple basis on which to refer all pixels. Indexes vary uniformly according to axis direction.[9]

To reference circular like filters, (which are often found -Log, Prewit, Sobel, Top Hat, Bottom Hat, among others) and to apply local operators such as morphological operators or local thresholding there are better ways to reference a pixel to it's neighborhood. A good intuitive system is the 3D Hexagonal Addressing. In this scheme, as seen in figure 6, there are three

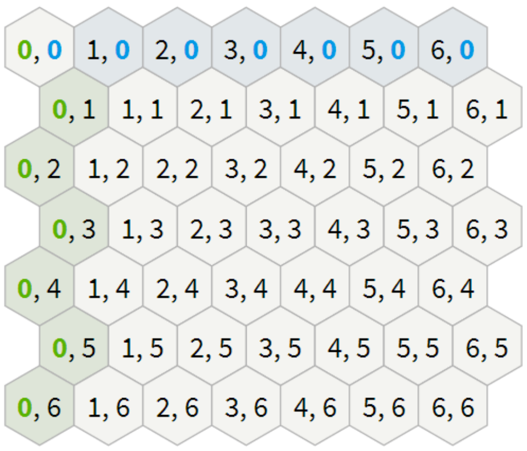

Figure 5: Offset Addressing -This method of addressing results of the translation of the centers of the even lines on a square grid, and is therefore an extremely practical way of containing data.[9]

main axis, each denoting the direction of two sides of the central hexagon. In function of a reference hexagon, we can use this addressing to effortlessly describe filters, morphological operators, nonlinear masks, and general "directions" among pixels.

3D Notation however can arguably waste some indexes, using an excess couple of bytes can be seen as potentially wasteful. Also, if operations sweeping the address space are impossible as each index does not necessarily point to an hexagon: some index values are impossible - try finding 0,2,-1, and the indexes are correlated. With that in mind yet an alternative hexagon addressing scheme exists: Spiral Addressing

There are two types of spiral addressing - the simple one as seen on figure 7, which basically grows as a spiral. This has limited usage apart from a singleton dimension in which to index hexagonal filters and images. And we have the much more versatile Honeycomb Scheme 8. This scheme was based on the hierachical organization of biological systems, and has some interesting properties regarding transformations and arrays. This system has been used by authors [7] to create accelerated gabor wavelet transforms, yet, they used an fpga with a dedicated algorithm for this system. Using matlab this convention is now advantageous at all, because matlab can process the convolution of offset matrices much faster than honeycombs. If a dedicated binary convolution solution was to be created for matlab, then the advantages in rotation could be exploited to increase computational capacity for multiple direction large filter banks, however, at this point that is unfeasible. The algorithm for operating gabor filter banks in Honeycomb

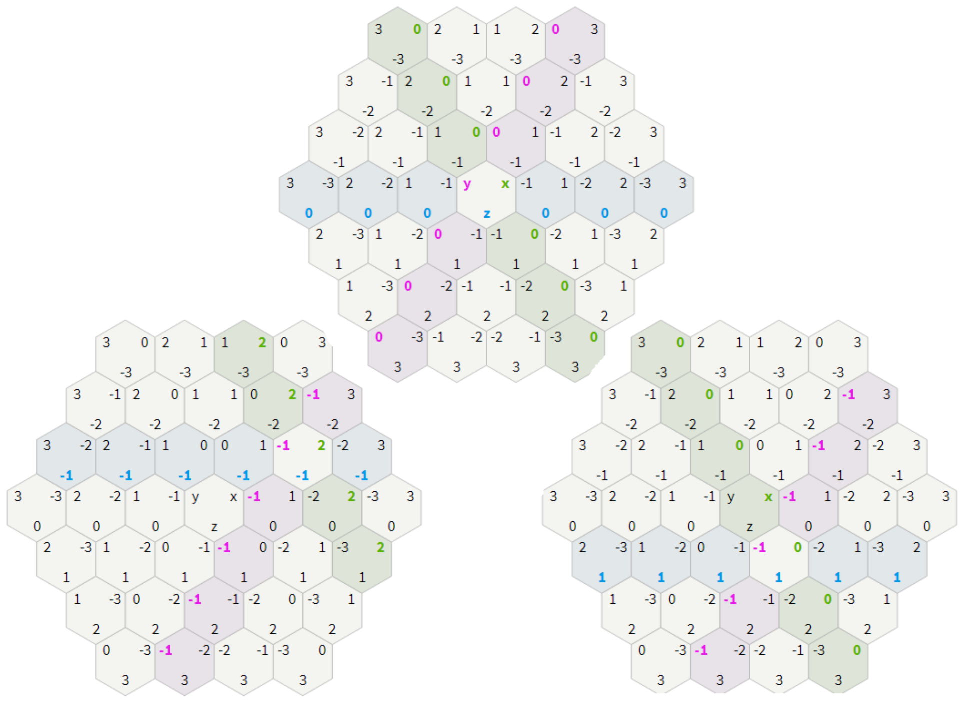

Figure 6: The 3D Addressing method, uses 3 Coordinates to reference each individual hexagon in the lattice. The 3 axis form a 60Âž basis. It might seem redundant to do with 3 address values what can be done with one. But then again square matrices can be addressed by a single index, two indexes is a matter of representation convenience.[9]

scheme is described in that work, and could be of future possible use in this one.

For simplicity, and matlab's delight, offset hexagonal matrices are the choice as the standard format for operations. Yet, to define filters, and neighborhoods, the other formats are more convenient. Thus matlab code converting the coordinates between systems was created, bridging the various systems together.

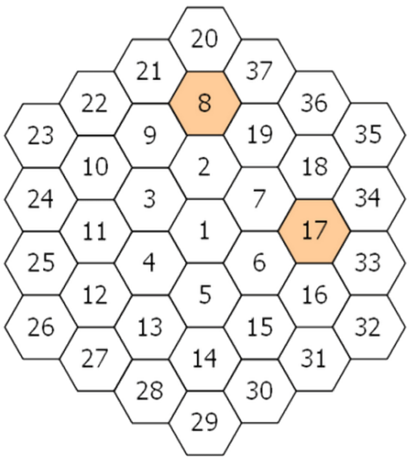

Figure 7: In this scheme each outer layer gets its members filled in, in a circle, with outer layers always having higher numbers. This system is hardly advantageous in any circunstances [9]

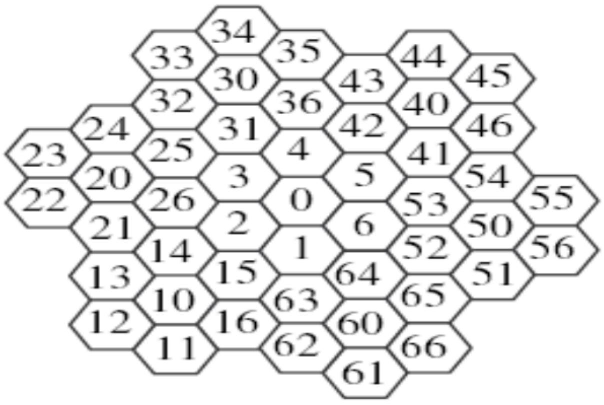

Figure 8: Hexagonal Spiral Honeycomb Scheme uses a base 6 notation to describe the relation of "macro" hexagons to each other. This scheme has great advantages in terms of the computational effort for operations of translation and rotation, being widely used for filtering purposes.[7]

Algorithms

0.1 Sampling into hexagonal grid

Most images and image acquisition systems work with the square grid (there are some useful notable exceptions in radar and nuclear medicine), therefor it was necessary to develop an algorithm to transform an arbitrary image into the hexagonal system. Mathematically we could devise an algebraic method to do this, by taking into account the projections of the hexagons into the squares and symbolically establishing the intersecting areas. This is however not entirely viable, and for simplicity, only rasterization options will be contemplated.

The most reliable method found, consists firstly of an oversampling of the original image, a calculation of the center points for the hexagons, and the average of the pixels in an hexagon mask at the given center points is computed, resulting in the "pixel value" for said hexagon[10]. Figure 9 shows graphically the procedure.

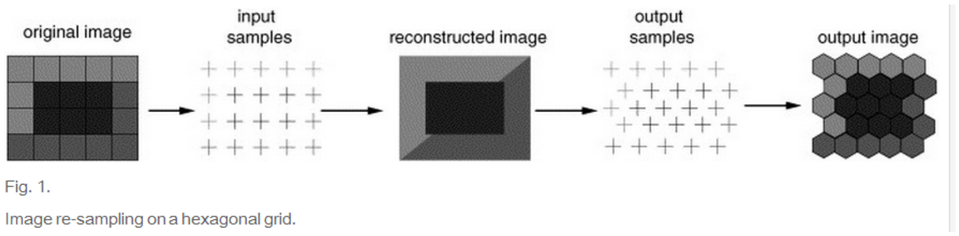

Figure 9: Sample flow of data in Hexagonal Sampling. An image with more resolution is first created, and from said image some points in an hexagonal lattice are either directly sampled, or from the local average/median/maximum/minimum the pixel value is defined. In this work we chose the average, though the median is also pretty good. [7]

0.2 Displaying Hexagonal Grid

It was necessary early on, to allow for the visualization of hexagon lattices in our environment of choice (matlab). Lacking native functions to do that, a sprite system was devised. This system consists of a super-pixel mask, which is applied at each output point, in a higher resolution square grid image (as befits the standard display used in virtually every computer who might run this software). The mask is multiplied by the sample value yielding the expected results.

0.2.1 Hyper-pixel

The Hyper-pixel has the propriety of interpolating an hexagon, as well as fitting with itself in a square matrix. There is no known way of creating hyper pixels or arbitrary size, yet some hyper pixels have been manually designed and are known for hexagonal representation. In figure 10 two proposed examples of hyper-pixel are given.

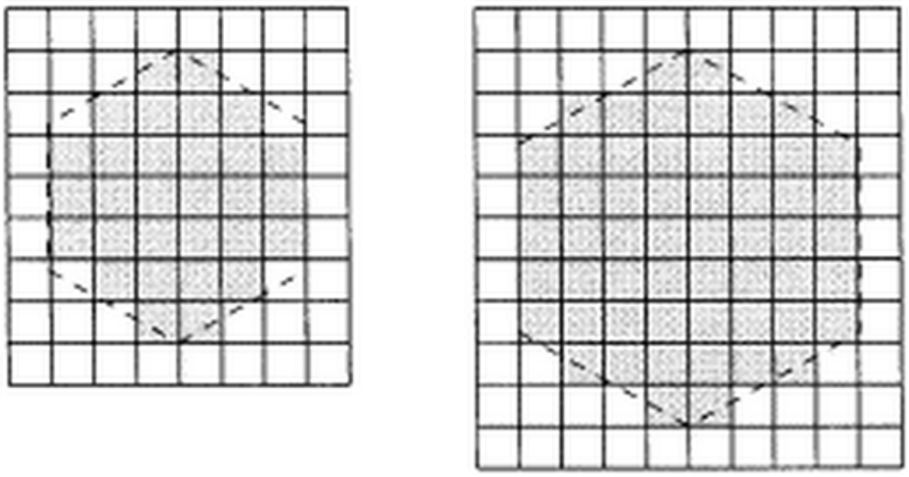

Figure 10: Here are shown two possible candidates for the hyper-pixel, notice that they fit themselves on the edges. The larger hyper pixel was favored in the implementation as the smaller one looked too "circular". The larger one conveys better the image of the hexagon.[6]

The usage of the suggested hyper pixels has an unfortunate drawback, acknowledge by its authors: it compresses the image horizontally slightly, that is because these hyper-pixels are not perfect in terms of size, and their aspect ratio makes them misrepresent the image. This effect is only present in rendering and sampling, all other algorithms and calculations are executed on matrices which aren't affected by the hyper-pixels bias.

The alternative, which is to use a rasterized version of the defined hexagons would imply a much higher computational cost, and the result, results in uneven hexagons which give a poor look to the final result, thus the hyper-pixel was the unchallenged candidate for the hexagonal rendering engine.

0.3 Coordinate Transform

The basis coordinates used are 3D coordinates, orthogonal, and offset. The functions converting each of them in the others were created. The found relations are as follows:

Let the 3D Coordinate system be defined by coordinates x_d,y_d and z_d, the orthogonal system defined by coordinates x_r,y_r, and the offset system defined by x_o and y_o. The following relations were computed and used:

Parameter	Value
Resolution	17x17
λ	1
θ	0
ϕ	0
σ	5
γ	1

Table 1: Parameters for the shown Gabor Filter

$$\begin{cases} x_r = x_d \\ y_r = z_d \end{cases} \tag{13}$$

$$\begin{cases} x_d = x_r \\ z_d = y_r \\ y_d = -x_r - y_r \end{cases} \tag{14}$$

$$\begin{cases} x_f = x_d + (z_d - (z_d mod 2))/2 \\ y_f = z \end{cases} \tag{15}$$

$$\begin{cases} x_d = x_o - (y_o - (y_o mod 2))/2 \\ z_d = y_o \end{cases} \tag{16}$$

$$\begin{cases} x_o = x_r + (y_r - (y_r mod 2))/2 \\ y_o = y_r; \end{cases} \tag{17}$$

0.4 Hexagonal Gabor Filtering

The gabor kernel equation is as follows:

$$\begin{aligned} g(x,y,\lambda,\theta,\phi,\sigma,\gamma) &= e^{-\frac{(xcos\theta+ysin\theta)^2+\gamma^2(-xsin\theta+ycos\theta)^2}{2\sigma^2}} \\ &\times e^{i(2\pi\frac{xcos\theta+ysin\theta}{\lambda}+\phi)} \end{aligned} \tag{18}$$

The preparation of Gabor Filters is the simple discretization of the function g, sampled at given points as provided by a hexagonal grid mesher. A filter with the parameters described in table 1 was constructed with hexagonal sampling.

The convolution of the Gabor filter in it's offset form with an image in offset form is trivial using the build it matlab operator.

Results

The resulting hexagonal renderer was achieved. Some fine tuning might yet be necessary to overcome the image stretching horizontally. Otherwise the results are of the best quality. A random image is shown in figure 11.

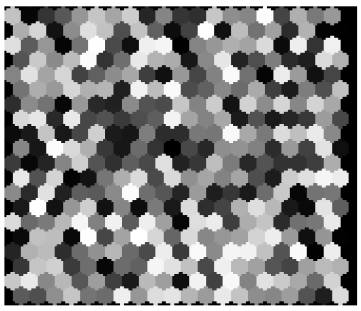

Figure 11: Here is shown a 20x20 random image gray scale. The rendered hyper-pixels are tricky to align, it's not just a matter of applying the mask at sampling points.There need be some considerations regarding the particular hyper-pixel, such as how many top diagonal pixels it has and take that into account.

The implemented hexagonal framework as was described in the earlier sections was used in the stock photo "Lenna" in black and white, fig. 12 - Showcased there are the rendering engine, the hexagonal grid generator, hexagonal sampling, compared to simple down sampling. Earlier the mathematical advantage was established. In this case we have the same amount of pixels in either the quad or the hex grid, and given that it originated from a higher resolution image (512x512), which was oveersampled. to yield 1024x1024. Then 128x128 pixels were used in the sampling, but the hyper pixel size is 8 across, 9 tall, resulting in the image appearing being scaled so as to fit the hexagonal sampling.

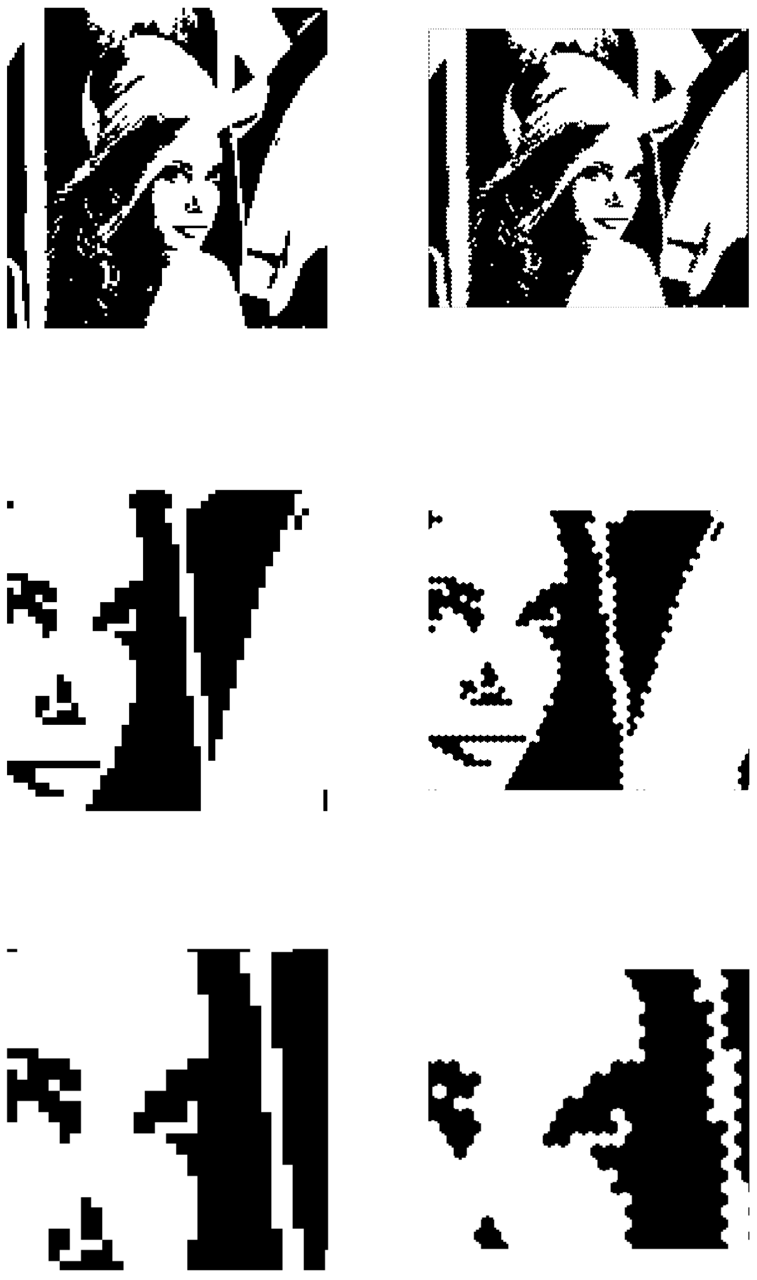

Figure 12: The classical "Lenna" image was down sampled to 128x128 both in the square grid system(left) and the hexagonal grid system (right). The hexagonal renderer implemented skews the image a little, they hold arguably the same information, but hexagonal grid is more insensitive to angular alias.

The desired result for this work was the production of hexagonal gabor filters, as those will be the most tool achieved. A modified version of the gabor filter generator was produced. A filter with the parameters shown in table 1. The resulting hexagonal image filter can be seen in figure 13.

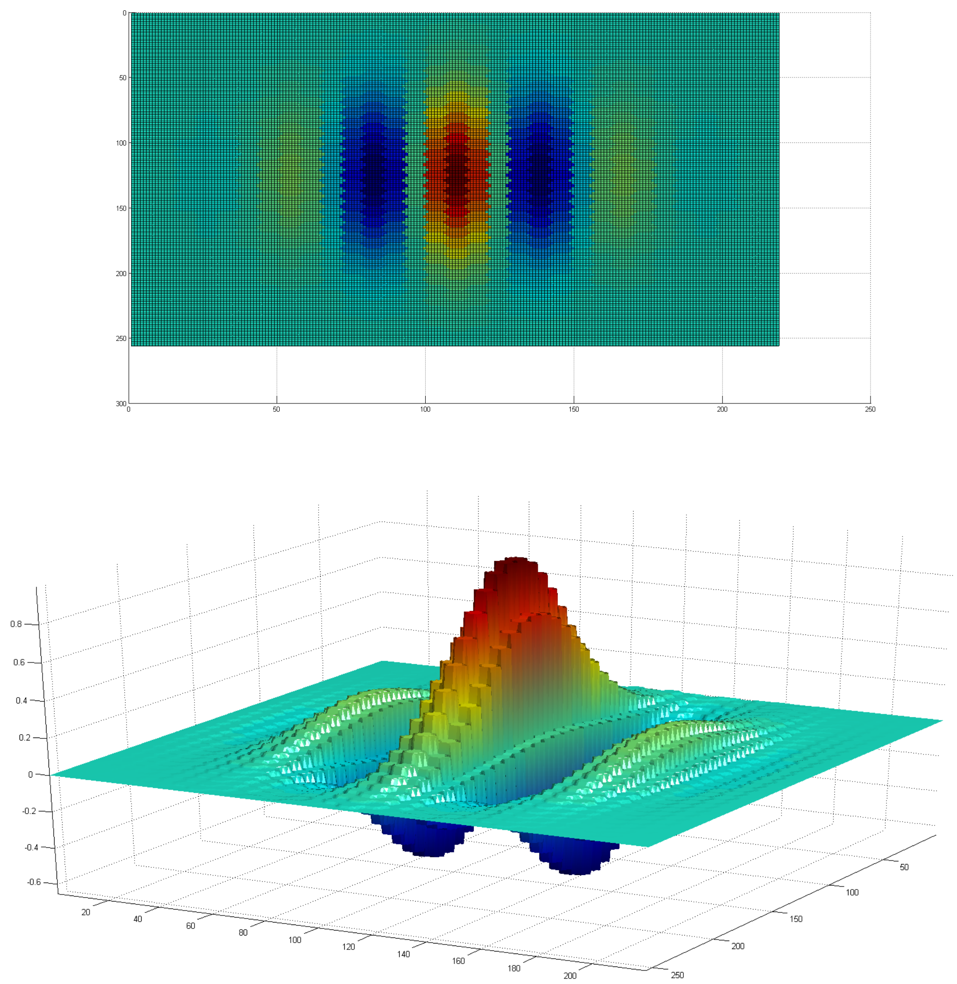

Figure 13: 3D Image, in perspective, with lighting of a gabor filter's magnitude in hexagonal grid. The hexagonal grid was rasterized to 2D Square grid and rendered using matlab's tools, thus the hexagons seen are interpolated by a square grid. The details expected from a gabor filter are present.

These results are pretty standard considering the goals.

Conclusions

The visual inspection of a square grid versus an hexagonal grid show that the latter can convey more information with the same amount of pixels.

As a byproduct of this work some tools have been developed (and for reference there were none available) which enable among other things to operate with hexagonal imaging. A promised future work will be to grind and make more universal said tools, so as to allow the production of a Matlab toolbox. Details on the work to be done can be found in the future work section.

These improvements in processing will hopefully translate in at least better computational performance once incorporated in the main algorithms of neural prediction. The shown quality for lesser pixels cannot be stressed enough, as computational times for 128x128 images are prohibitive, and the computational cost scales with the 3rd power of the size of the image stimulus, so using more filters in less pixels will be advantageous for overall quality.

One simple general conclusion is that hexagonal processing brings many advantages, but at a cost of minutiae details which are normally insufferable. If one is considering hex usage for vision applications outside of the academic domain; the bothersome roadblocks will become prohibitive as one considers one's ideas more interesting that fighting a rather cumbersome system which will provide plenty of out of bounds indexes. Yet it can reward in the end as well.

Future Work

A Natural Development of the referred topics, is to incorporate this stage of processing into the broader picture, that is to say, use the achieved results in the next phases of neural prediction processing. That and other work will be subject of the PhD thesis.

A toolbox with the implemented functions is to be prepared for the community. For such purposes the code must be adapted, for instance, as of now, it doesn't support color, as stimulus in the neural prediction dataset are usually gray scale. The code listing can be found in `http://paginas.fe.up.pt/~bio08062`. It includes hexagonal rendering engine, various coordinate converters, an hexagonal grid sampler, an hexagonal mesh tool (similar to "gridmesh") and an hexagonal 3D Gabor tool.

References

1. P. Dayan and L.F. Abbott. *Theoretical neuroscience*, volume 31. MIT press Cambridge, MA, 2001.
2. Behzad Kamgar-Parsi. Quantization error in hexagonal sensory configurations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(6):665–671, 1992.
3. Xiangjian He and Wenjing Jia. Hexagonal structure for intelligent vision. In *Information and Communication Technologies, 2005. ICICT 2005. First International Conference on*, pages 52–64. IEEE, 2005.
4. Jean Serra. Introduction to mathematical morphology. *Computer vision, graphics, and image processing*, 35(3):283–305, 1986.
5. Russell M Mersereau. The processing of hexagonally sampled two-dimensional signals. *Proceedings of the IEEE*, 67(6):930–949, 1979.
6. Lee Middleton and Jayanthi Sivaswamy. *Hexagonal image processing: A practical approach*. Springer, 2006.
7. Barun Kumar and Er Pooja Kaushik. Vision-based hexagonal image processing based hexagonal image processing based hexagonal image processing.
8. Samuel Matej, Avi Vardi, Gabor T Herman, and Eilat Vardi. Binary tomography using gibbs priors. In *Discrete Tomography*, pages 191–212. Springer, 1999.
9. Red Blob Games. *Hexagonal Grids*, Mar 2013 (accessed June,4 2014). `http://http://www.redblobgames.com/grids/hexagons/`.
10. Richard C Staunton. The processing of hexagonally sampled images. *Adv. Imaging Electron Phys*, 119:191–265, 2001.