Fast Low-level Multi-scale Feature Extraction for Hexagonal Images

S.A., Coleman, B.W., Scotney, B. Gardiner
Ulster University
United Kingdom
Sa.coleman@ulster.ac.uk; bw.scotney@ulster.ac.uk; b.gardiner@ulster.ac.uk

Abstract

Inspired by the human vision system and its capability to process in real-time, an efficient framework for low-level feature extraction on hexagonal pixel-based images is presented. This is achieved by utilising the spiral architecture addressing scheme to simulate eye-tremor along with the convolution of non-overlapping gradient masks. Using sparse spiral convolution and the development of cluster operators, we obtain a set of output image responses "a-trous" that is subsequently collated into a consolidated output response; it is also demonstrated that this framework can be extended to feature extraction at different scales. We show that the proposed framework is considerably faster than using conventional spiral convolution or the use of look-up tables for direct access to hexagonal pixel neighbourhood addresses.

1. Introduction

Inspired by the human vision system and its capability of real-time processing, the possibility of increasing efficiency when performing processing tasks such as feature extraction is considered by combining three characteristics of the human vision system. Firstly, we consider how the visual information is acquired: the fovea, a small region within the retina, which consists of a large number of photoreceptor cones, is accountable for the sharpness obtained during vision capture. These photoreceptors are structured in a hexagonal arrangement [4, 5, 11, 14]. To mimic this characteristic we represent images on a hexagonally structured grid using pixels that are hexagonal in shape. Secondly, ganglion cells do not completely overlap in the central fovea of the retinal photoreceptive fields [9]. Correspondingly, we develop a framework for feature extraction in which image processing operators are convolved with images in a non-overlapping way; this contrasts with conventional operator-image convolution. Thirdly, three types of movement can be found with the human eye: drift, micro-saccades and tremor [12]. As a consequence of the latter, (i.e. eye tremor - an involuntary oscillation of the eye), the human vision system sparsely processes a series of slightly off-set images obtained from these rhythmic movements, rather than processing single static images. Correspondingly we adapt this eye tremor approach to process images by using a set of similarly off-set images, each of which is partially processed by non-overlapping filters.

Recent research has highlighted the many advantages of hexagonal pixel-based images [2, 3, 6, 15, 16, 17]. Sheridan presented an alternative addressing system, known as the Spiral Architecture, that labels each hexagonal pixel with a single co-ordinate address. However, processing images indexed by the Spiral Architecture is not always trivial as the method of retrieving pixel neighbours within a one-dimensional spiral addressing scheme for processing tasks such as operator convolution requires computationally expensive radix-7 addition. We exploit the concept of eye tremor to circumvent this difficulty. Building on our work in [3], we show how using the Spiral Architecture structure, we can develop scalable *cluster operators* that can be applied efficiently to a spirally addressed image. Using the eye tremor concept [11], for each static image, six additional images are generated but with the spiral centre spatially off-set by one pixel. This presents the opportunity to convolve gradient operators efficiently in a non-overlapping fashion with each of the off-set images. For any operator size, we can thus obtain resultant feature maps "*a-trous*" for each off-set image that can be consolidated to form a single complete feature map.

2. Spiral Architecture

We use the spiral addressing scheme in [13] that originates at the centre pixel of the hexagonally structured image (pixel index 0) and progresses outward in a spiral manner using a one-dimensional index. This addressing scheme makes it possible to index each pixel within a hexagonal image using a single co-ordinate value, permitting the efficient storage of the image within a one dimensional vector [10]. Pixels that are neighbours within a 7-pixel layer-1 super-tile within a hexagonal image will form a contiguous set within the one dimensional vector. This is key to achieving fast and efficient processing when applying image processing operators directly to the stored image vector. See [13] for further details.

2.1. Spiral Neighbourhoods

Our approach is based on a specially defined convolution of neighbourhood operators with a set of spatially off-set images. We may define these operators over a range of scales, and we refer to the neighbourhoods over which they are convolved as *cluster neighbourhoods*, as the neighbourhoods may be defined recursively using clusters of neighbourhoods at a smaller scale. Within an overall hexagonal lattice structure we may consider hierarchically ordered clusters with six-fold rotational symmetry. Each lowest order (layer-0) cluster consists of a single pixel. Each pixel has six immediate neighbours: a pixel and its immediate neighbours form a layer-1 cluster (containing seven pixels). Similarly we may generate layer-2 clusters (of 49 pixels) where each layer-2 cluster comprises a layer-1 cluster and its six immediately neighbouring layer-1 clusters. We may generate higher order clusters in a recursive fashion, in which a layer-*n* cluster is generated by combining a layer-(*n-1)* cluster with its 6 neighbouring layer(*n-1)* clusters. For example, a single layer-2 cluster, comprising seven layer-1 clusters (each centred at a pixel labelled 0) is presented in Figure 2.

3. Cluster Operators

We use the term *cluster operators* for operators that are applied to cluster neighbourhoods in the spiral architecture. In recent work by Coleman et al., [3, 7, 8] it has been shown how adaptive hexagonal operators can be created using a finite element framework constructed of either derivative operators

aligned orthogonally (ie, aligned along the *x*- and *y*- axis), or tri-directional operators, each separated by 60°, aligned along the *x*-, *y*- and *z*- hexagonal axes. In this paper, our operators are aligned along the *x*- and *y*- Cartesian axes and are constructed by placing a node at the centre of each pixel centre and connecting these using a virtual lattice of equilateral triangles. We associate a piecewise linear basis function ϕ_t with each node *t*, where $\phi_w = 1$ at node *w=t* and $\phi_w = 0$ at all other nodes $w \neq t$. Therefore each ϕ_w is a "tent-shaped" function with support constrained to a small neighbourhood of six triangular elements centred on node *w*. A hexagonal image is represented by a function $I = \sum_{q \in A} I(q)\phi_q$, where *A* denotes the set of all nodal addresses and the image intensity value of the each pixel centred at node *q* is denoted by $\{I(q)\}$.

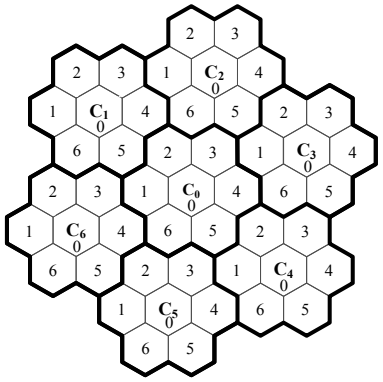
Operators for feature detection, more specifically edge detection, are often derived using first derivative approximations. Similarly, over cluster neighbourhoods we present a weak form of the first directional derivative $I u/I b$ [$\underline{b} \times \underline{u}$. By considering cluster neighbourhoods of increasing size (corresponding to layer-λ, $\lambda = 1,...,L$) we may generate operators at increasing scale. Applied to a pixel location with index *w*, the derivative is approximated over a λ-neighbourhood $N_\lambda(w)$ by multiplying a test function $v \in H^1$ with the derivative term; the result is then integrated over $N_\lambda(w)$. The neighbourhood size refers to the corresponding layer-λ cluster, e.g. seven hexagonal pixels are included in a $N_1(w)$ neighbourhood cluster, and 49 hexagonal pixels in a $N_2(w)$ neighbourhood cluster. Hence, we can obtain a layer-λ directional first derivative $D_\lambda(w)$ at each node *w* as

$$D_\lambda(w) = \int_{N_\lambda(w)} \underline{d} \cdot \nabla I \psi_w^\lambda d\Omega$$

where $\underline{b}$ 8 (cos⁄,sin⁄) represents the unit direction vector. Thus we can say that for any choice of layer $\lambda = 1,2,3,...$ etc., each test function $v \in H^1$ has its support constrained over the neighbourhood $N_\lambda(w)$, and

$$D_\lambda(w) = \sum_{q \in A} \left(I(q) \int_{N_\lambda(w)} \underline{d} \cdot \nabla \phi_q \psi_w^\lambda d\Omega \right) = \sum_{q \in N_\lambda(w)} H_\lambda(q) \times I(q)$$

where H_λ is a hexagonally structured operator equivalent in size to the hexagonal λ-neighbourhood. In this instance, a Gaussian function was selected for the layer-λ neighbourhood test function $v \in H^1$, implemented so that at least 95% of the function's cross section covers $N_\lambda(w)$. For implementation purposes, as shown in Figure 1, a layer-2 operator can be stored as a sequence of 7-point layer-1 operators in rows of a two-dimensional array indexed by the layer-1 cluster neighbourhoods that comprise the corresponding layer-2 neighbourhood cluster. Efficient implementation using such a storage structure for layer λ cluster operators in general is discussed in Section 4.



Cluster	Cluster Index 0	1	2	3	4	5	6
C_0	0	-0.011	-0.005	0.005	0.011	0.005	-0.005
C_1	-0.008	-0.001	-0.004	-0.003	-0.007	-0.012	-0.020
C_2	0.002	-0.007	-0.004	0.004	0.012	0.007	0
C_3	0.011	0.012	0.013	0.005	0.009	0.016	0.014
C_4	0.008	0.007	0.012	0.020	0.010	0.004	0.003
C_5	-0.002	-0.012	-0.007	0	0.007	0.004	-0.004
C_6	-0.011	-0.009	-0.016	-0.014	-0.012	-0.013	-0.005

Fig. 1. Storage structure for a larger scale convolution operator (x-derivative in this example)

4. Bio-inspired framework for Fast Processing

Adapting the characteristics of the human vision system discussed in Section 1, a bio-inspired framework for fast processing has been developed that has a number of elements: firstly, the concept of eye tremor is simulated by marginally off-setting images taken from a selected scene; secondly the paradigm of sparse processing of a scene is implemented using a sparse spiral convolution that enables interaction of a λ -neighbourhood cluster operator to be limited to an easily defined subset of λ -neighbourhood clusters; thirdly, an efficient approach for acquiring the addresses of neighbouring hexagonal pixels within the spiral architecture for λ -neighbourhood clusters is developed for convolution of cluster derivative operators.

4.1. Eye Tremor Simulation

Consider a particular scene represented by a hexagonal pixel-based image I_0. Six additional images, $I_j, j = 1,...,6$, are also generated for that scene, with each image I_j off-set by one pixel in the direction of one of the hexagonal axes. The result is six images spatially off-set by one pixel from the origin of the "base" image I_0, producing the "eye tremor" processing framework. The spiral address 0 in each image $I_j, j = 1,...,6$ corresponds to the original hexagonal pixel with address *j* in I_0.

4.2. Non-overlapping Spiral Convolution

To convolve a hexagonally structured derivative operator H_λ of "size" λ with a given image I_0, H_λ is sparsely applied to each image $I_j, j = 0,...,6$, and the resultant vectors combined to achieve the consolidated output.

The operator H_λ is applied only to pixel locations with a spiral address 0 (*mod* 7), i.e. sparse convolution. This is illustrated in Figure 2, where pixel positions are labelled

indicating, for each image $I_j, j = 0,...,6$, which pixel location equates to 0(*mod* 7). This labelling of each pixel in image I_0 can then be used to define "base-7-zero" convolution $S_\lambda = H_\lambda \otimes_{7_0} I$ of the operator H_λ with image I by:

$$S_\lambda(w_0) = \sum_{w \in N_\lambda(w_0)} H_\lambda(w) \times I(w)$$

$$\forall w_0 \in \{w | w = 0(\mathrm{mod}7)\}$$

and $N_\lambda(w)$ denotes the neighbourhood cluster (layer-λ) placed at the pixel with address *w*. Therefore, the simulation of eye tremor in relation to a particular scene represented by a base image I_0, can be implemented by applying "base-7-zero" convolution ($\otimes_{7_0}$) of the derivative operator H_λ to each image $I_j, j = 0,...,6$, resulting in seven output responses:

$$S_\lambda^j = H_\lambda \otimes_{7_0} I_j, j = 0,...,6$$

which when consolidated provide the overall output

$$E_\lambda = H_\lambda \otimes I_0.$$

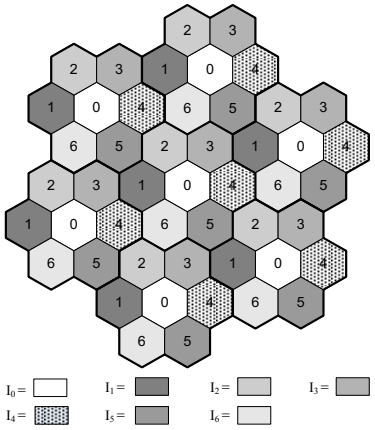


Fig. 2. Position of pixels in the "base" image I_0 with equivalent value 0 (mod 7) addresses for images $I_j, j = 0,...,6$.

4.3. Address Identification of Cluster Neighbourhood

For any layer-λ cluster operator, when implementing "base-7-zero" convolution as defined in Section 4.2 it is required to identify the spiral addresses of the pixels in each of the layer-λ cluster neighbourhoods that are centred at pixels with spiral address 0 (*mod* 7). In the case of $\lambda = 1$, i.e., a simple 7-point hexagonal operator, this task is particularly straightforward: if the centre pixel of the cluster neighbourhood has spiral architecture address w_0 ($w_0 = 0, \mathrm{mod}7$), then the 6 adjacent neighbours are addressed in a clockwise direction, giving an ordered set of addresses as $\{w_0 + j\}_{j=0}^6$.

These addresses are contiguous in the vector used to store the image data. Hence, when $\lambda = 1$, "base-7-zero" convolution may be implemented efficiently in the spiral architecture by simply identifying segments of length 7 of the one-dimensional image storage vector and using point-by-point multiplication with the 7 values of the cluster operator stored in a vector in the same clockwise order (i.e., in the order corresponding to the addresses $\{j\}_{j=0}^6$).

For larger operators, when $\lambda > 1$, in order to obtain a set of spiral architecture addresses to define the layer-λ cluster neighbourhood $N_\lambda(w_0)$, we follow a recursive process based on the cluster neighbourhood hierarchy. This process identifies an ordered set containing the spiral architecture addresses of the centres of the layer-1 cluster neighbourhoods that comprise $N_\lambda(w_0)$. As the address of each such centre contains the value 0 *mod* 7, the layer-λ operator may be applied efficiently as a set of layer-1 operators to the corresponding set of layer-1 cluster neighbourhoods, in each case using the simple procedure described above for the case $\lambda = 1$. For example, as illustrated in Figure 1, a layer-2 cluster operator may be implemented efficiently as an ordered set of layer-1 operators, each of which comprises 7 values that are convolved with 7 contiguous image values in the one-dimensional storage vector.

Application of a layer-λ operator requires identification of an ordered set of addresses of the centres of the layer-1 cluster neighbourhoods contained in $N_\lambda(w_0)$: there are $\lambda - 1$ recursive steps to be carried out, with 7^α centres of layer-($\lambda - \alpha$) cluster neighbourhoods identified at *step* $\alpha = 1, \ldots, \lambda - 1$. Starting with $c_0 = w_0$, for each of the $7^{\alpha-1}$ layer-($\lambda - \alpha - 1$) centres, $c_{\alpha-1}$, identified at step $\alpha - 1$, 7 layer-($\lambda - \alpha$) centres are identified at step α as:

$$c_\alpha = c_{\alpha-1} + (10)^{\lambda-\alpha} i_\alpha, \; i_\alpha = 0,...,6$$

Thus, after $\lambda - 1$ recursive steps, we generate $7^{\lambda-1}$ layer-1 centres that are indexed as

$$index(c_{\lambda-1}) = \sum_{\alpha=1}^{\lambda-1} (10)^{\lambda-\alpha-1} i_\alpha$$

Considering the centre of each layer-1 cluster neighbourhood, $c_{\lambda-1}$, the addresses for the pixels in that cluster neighbourhood are represented by

$$\{c_{\lambda-1} + j\}_{j=0}^6$$

In Figure 3, we show pseudo-code for the implementation of the cluster neighbourhood address identification as described above, i.e., the code generates an ordered set, $\{c[j]\}$, of the spiral architecture addresses that comprise $N_\lambda(w_0)$. Based on this approach, in the number of spiral additions required to obtain a set of spiral addresses in a layer-λ cluster neighbourhood $N_\lambda(w_0)$ is significantly reduced compared with the number of spiral additions needed to obtain a set of spiral addresses for a λ-cluster neighbourhood $N_\lambda(w)$ with $w \neq 0, \mathrm{mod}7$. (For $w \neq 0, \mathrm{mod}7$, a complete set of 7^λ spiral additions is necessary).

Thus, as demonstrated by the run-times presented in Table 1, our proposed methodology of using sparse "base-7-zero" for operator convolution is considerably more efficient than the standard spiral convolution approach. An alternative approach would be to pre-compute and store all of the cluster neighbourhood addresses in a look-up table (LUT). However, due to the computational burden associated with extensive recall of LUT values, even this approach is less computationally efficient than

our proposed algorithm.

```
int c[0] =  w0; //centre
int m=0;
for (α=1;  α ≤ λ−1;  α++)
    {
     neighbour(c[0],  α );
    }

neighbour(int c, int a)
    {
     for (i=1, i<=6, i++)
        {
                m=m++;
                c[m] = c +  10^a i
                a= a-1;
                if (a>0)
               {
                  neighbour(c[m],a);
                }
                a= a+1;
        }
    }
```

Fig. 3. Algorithm for address identification

5. Performance Evaluation

To determine the efficiency of the proposed eye tremor approach, performance evaluation was completed with run-times measured for convolving an operator using the eye-tremor approach, and these then compared with run-times measured for convolving an operator with a spirally addressed image using a standard approach where the neighbouring addresses for each pixel are obtained using two different methods: (i) where pixel neighbourhoods are found using time consuming hexagonal arithmetic; and (ii) a LUT is generated that stores the neighbourhood addresses prior to operator convolution. (The time taken to generate the LUT is 0.4017s in the case of $\lambda=1$ and 4.2172s in the case of λ=2). Evaluation consisted of applying both the λ=1 and λ=2 cluster operators to the *Lena* image of size 400*x*400 which has been converted from square to hexagonal pixels using the method in [13].

Table 1. Performance of convolving 7-point operator

Convolution Approach	**Run-time** ($\lambda=1$)	**Run-time** ($\lambda=2$)
Proposed "eye tremor"	0.0079s	0.1399s
Standard	3.7621s	24.5104s
Standard using LUT	0.0711s	0.2191s

The results in Table 1 (averaged over 100 runs using a workstation with a 2.99Ghz Pentium D processor and 3.50Gb of RAM) demonstrate that the implementation speed using standard spiral convolution can be enhanced by using a LUT; but more importantly the results show that our proposed eye-tremor approach is orders of magnitude faster than standard spiral convolution, and significantly faster even than use of a LUT: nine times faster for a λ=1 cluster operator, and twice as fast for a λ=2 cluster operator. The overhead of storing the LUT should also be considered, which increases substantially for larger λ values. The run-times in Table 1 should be considered as relative, as no attempt has been made to optimise implementation.

6. Conclusion

Motivated by the human vision system, a biologically inspired approach to efficient feature extraction has been presented, which consists of a hexagonal pixel-based framework that permits the implementation of eye tremor and non-overlap of operator convolution. We have shown that by producing feature maps "*a-trous"* (at various scales), and storing these in one-dimensional vectors, a single complete feature map can be generated significantly faster than producing the equivalent feature map via standard convolution or with the aid of a pre-stored neighbourhood LUT.

As a focus for future work, the proposed eye tremor approach will be investigated for fast video processing. Each frame in a sequence can be slightly off-set from the previous frame in a cyclic pattern. Hence (possibly overlapping) sub-sequences of seven frames may be processed at high speed to generate a "continuous" time-variant feature map.

References

[1] Birdsong, James B., and Nicholas I. Rummelt. "The hexagonal fast fourier transform." 2016 IEEE International Conference on Image Processing (ICIP), 2016.

[2] Coleman, S.A., Scotney, B.W., Gardiner, B., "Tri-directional Gradient Operators for Hexagonal Image Processing", J. of Visual Communication and Image Representation, 38 . pp. 614-626, 2016.

[3] Coleman, SA, Scotney, BW and Gardiner, B (2011) *Biologically Motivated Feature Extraction.* In: 16th ICIAP, Ravenna, Italy. Springer. Vol 6978/2 10 pp.

[4] Curcio, C.A., et al., "Human Photoreceptor Topography", J. of Comparative Neurology, Vol. 292, pp. 497-523, 1990.

[5] Gardiner, B., Coleman, S.A., Scotney, B.W., "Biologically Motivated Feature Extraction Using the Spiral Architecture" IEEE ICIP Brussels, 2011.

[6] Gardiner, Bryan, Coleman, Sonya and Bryan, Scotney, "Multiscale Edge Detection using a Finite Element Framework for Hexagonal Pixel-based Images", IEEE Trans on Image Processing, 25 (4). pp. 1849-1861, 2016.

[7] Hirsch, J. and Miller, W.H., "Does Cone Positional Disorder Limit Resolution?", J. Optical Soc. of America A: Optics, Image Sci, and Vision, Vol.4, pp. 1481-1492, 1987.

[8] Middleton, L. and Sivaswamy, J., "Hexagonal Image Processing; A Practical Approach", Springer 2005.

[9] Roka, A., et al., "Edge Detection Model Based on Involuntary Eye Movements of the Eye-Retina System" *Acta Polytechnica Hungarica*, 4(1), pp31-46, 2007

[10] Sheridan, P., Spiral Architecture for Machine Vision, Ph.D. Thesis, University of Technology, Sydney, 1996.

[11] Shima, T., Saito, S., & Nakajima, M., "Design and Evaluation of More Accurate Gradient Operators on Hexagonal Lattices", *IEEE Trans PAMI*, 32 (6), pp. 961-973, 2010.

[12] Wu, Q., He, X., Hintz, T., "Virtual Spiral Architecture," Int Conference on Parallel and Distributed Processing Techniques and Applications, pp. 339-405, 2004

[13] Wuthrich, C.A, Stucki, P. "An Algorithmic Comparison Between Square-and Hexagonal-based Grid", Graphical Models and Image Processing, Vol. 53, pp. 324-339, 1999.