Fast, Multi-Scale Image Processing on a Square Spiral Framework

John Fegan
Ulster University – Magee
Northland Road
Derry/Londonderry BT48 7JL
028 9036 6305, +44
fegan-j2@ulster.ac.uk

Sonya Coleman
Ulster University – Magee
Northland Road
Derry/Londonderry BT48 7JL
028 7167 5030, +44
sa.coleman@ulster.ac.uk

Dermott Kerr
Ulster University – Magee
Northland Road
Derry/Londonderry BT48 7JL
028 7167 5330, +44
d.kerr@ulster.ac.uk

Bryan Scotney
Ulster University
Cromore Road
Coleraine BT52 1SA
028 7012 4648, +44
bw.scotney@ulster.ac.uk

ABSTRACT

Efficient processing of digital images is a key consideration in many machine vision tasks. Traditional image processing approaches often struggle to meet this demand, particularly at the initial low-level of processing image pixels. To overcome this, we propose a spiral based processing approach which takes inspiration from the asymmetric lattice of interlocking cells found in the human visual system. Here we demonstrate the efficiency of the proposed spiral approach for multi-scale feature extraction. This is complemented by a biologically inspired image acquisition process which is used to capture nine image frames at different spatial locations. The results demonstrate that the biologically inspired spiral approach offers a faster alternative to corresponding traditional image processing approaches.

CCS Concepts

• Digital Image Processing→Spiral Image Processing→Square Spiral Image Processing→Biologically Inspired Image Processing→Multi-Scale Feature Extraction

Keywords

Fast Image Processing; Square Spiral Address Scheme; Eye Tremor; Multi-Scale Feature Extraction; Edge Detection.

1. INTRODUCTION

Efficient processing of digital images is a key consideration in many machine vision tasks where computer hardware must operate on a stream of images under strict time constraints.

Traditional Image Processing (TIP) techniques often struggle to meet this demand and researchers believe that image processing performance could be improved by incorporating characteristics found in biological vision systems. For example, in the TIP framework, an image is captured on a rectangular lattice and the pixels are stored according to a two-dimensional (2D) address scheme. In contrast, the Human Visual System (HVS) senses stimuli through an asymmetric lattice of interlocking hexagonal cells [1]. This observation has inspired a framework where an image is captured on a hexagonal lattice and the pixels are stored according to a one-dimensional (1D) spiral address scheme [2]. Research has shown that this hexagonal framework can perform image processing much faster than the TIP framework [3]. Despite this benefit, the runtime performance of the hexagonal framework is undermined by a lack of hardware that can capture hexagonal pixel-based images and the subsequent cost needed to map a traditional image to a hexagonal image. To circumvent these issues, an analogous square spiral (squiral) framework has been proposed for use with standard image hardware [4]. In previous research, Squiral Image Processing (SIP) was primarily conducted using neighborhood operators that overlay an image at a single 3 × 3 scale. Building on the research in [5] we present an efficient approach to multi-scale feature extraction using two alternative methods: the first approach involves using a Look Up Table (LUT) to find the addresses of each pixel's surrounding neighbors; the second approach utilizes a *Coarse*, non-overlapping processing strategy where only a fraction of the pixels are processed and enlarged to produce a feature map.

2. TRADITIONAL IMAGE PROCESSING FRAMEWORK

There are two common ways to store an image in linear computer memory. The first way is to map the consecutive pixels of each row or column to a contiguous array [6]; however, the contiguous property is not preserved in two directions. By extension, an index derived from a single parameter is insufficient to navigate the array in two directions. In this case, two parameters are needed: a vertical coordinate *y* and a horizontal coordinate *x*.

Depending on the storage order, an index is derived as width × y + x or height × x + y. Subsequently, a dynamic array navigation algorithm will require multiplication at each row and addition at each column, or vice versa. The second way to store an image in linear computer memory is to treat each row or column as its own array. As illustrated in Figure 1 this layout requires a separate (Iliffe) array to locate each row or column [7]. Consequently, two indices are needed to locate a pixel: one to locate a row or column and another to locate a pixel relative to the start of that row or column. This avoids the heavy computation of the previous data structure as each pixel is indexed in two directions. Despite this advantage, the benefits of this arrangement are outweighed by broken continuity caused by the intermittent fetching of row or column locations from secondary memory space. The two data structures discussed offer simple ways to store an image and index its pixels, but they fail to provide a contiguous sequence for each pixel and its surrounding neighbors. For example, in order to operate on a 3 × 3 neighborhood centered on a pixel $I_{x,y}$, both rows and columns must be traversed. While pixels either side are within close proximity to $I_{x,y}$, the other neighbors above and below are located further from $I_{x,y}$ in the storage structure. Although this is not too concerning in small images, in large images the distance between a pixel and its neighbors can be substantial which can hinder performance [6]. The SIP framework alleviates this problem by storing the image in a 1D array structure where the pixels are within proximity to their nearest neighbors.

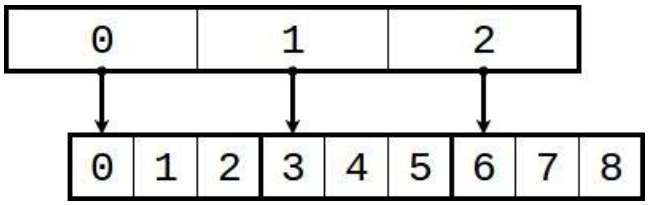

Figure 1. Iliffe Storage for a 3 × 3 Matrix

3. SQUARE SPIRAL IMAGE PROCESSING FRAMEWORK

As illustrated in Figure 2 the SIP framework employs a sampling scheme that partitions an image into subsets known as layers. The first pixel, layer 0, is located at the center of an image. Thereafter the pixel at layer 0 and its eight nearest neighbors are sampled in an outward spiral to form layer 1. Each subsequent layer encompasses eight clusters that surround the previous layer where each cluster contains the same number of pixels as the previous layer and the pixels are indexed in the same sequence as the previous layer. In accordance with this scheme the image is mapped to a contiguous array that perpetrates a base 9 index scheme. Consequently, each pixel that is indexed 0 mod 9 is stored adjacent to its eight nearest neighbors in the 1D array. This simplifies and facilitates fast spatial processing of these pixels as they, and importantly their nearest neighbors, can be traversed sequentially. However, the pixels which are not indexed 0 mod 9 cannot be processed this way as they are not stored adjacent to all of their nearest neighbors. This poses a difficulty when applying a neighborhood operator to *every* pixel in an image in order to obtain a complete output. However, this is readily overcome via a novel procedure based on involuntary eye movements called tremors as discussed in Section 3.1.

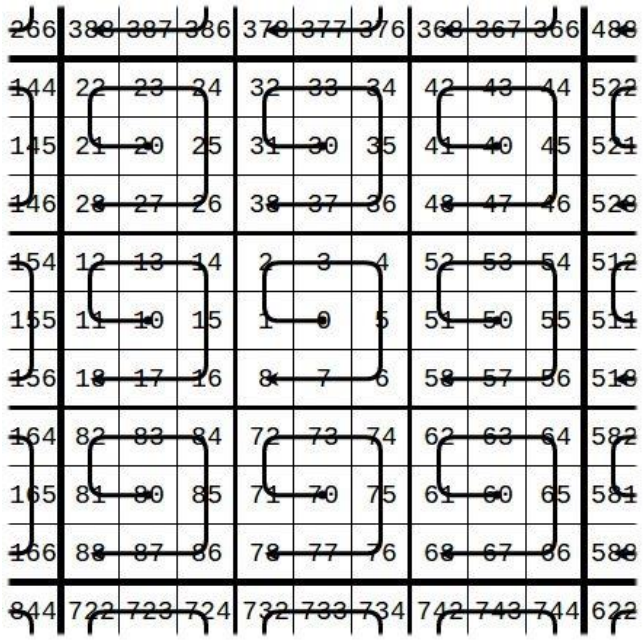

Figure 2. Square Spiral Sampling Scheme

3.1 Eye Tremor

In humans, involuntary eye movements called tremors are thought to cause spatial displacement of visual stimuli which the HVS can then sparsely interpret to derive meaning [1]. In a similar way, the behavior of eye tremor can be simulated in the SIP framework by capturing several image frames at different spatial locations. In accordance with the SIP framework, nine image frames are captured: the first frame is captured from the center of the sampling lattice; and the eight succeeding frames are captured from the immediate points surrounding the center. This procedure complements the squiral sampling scheme and maps each 2D point to a 0 mod 9 location in an array. Although this is accomplished using nine frames instead of one, each frame is only sparsely processed (only one-ninth of the standard processing overhead is required), and then the nine frames are combined to provide a complete feature map. The ability to process neighborhoods larger than 3 × 3 pixels is more challenging because they are partially fragmented. For example, a 9 × 9 neighborhood centered on a pixel I_{10} is distributed across the sub-clusters [I_{10}, I_{18}], [I_{150}, I_{158}], [I_{140}, I_{148}], [I_{20}, I_{38}], [I_{70}, I_{88}], and [I_{160}, I_{168}]. In Section 3.2 we present an efficient solution to overcome this problem and achieve fast multi-scale feature extraction.

3.2 Multi-Scale Neighborhood Processing

As mentioned in Section 1, an approach to multi-scale feature extraction using SIP was previously presented in [5]. However, the shortcoming of this approach is that as the scale of a neighborhood operator increases more frames need to be sampled. For example, to process an image with a 9 × 9 neighborhood operator, a total of 81 frames are required. For this reason, the solution in [5] is of limited use and not considered viable for machine vision applications. Instead we propose two alternative approaches. The first approach involves using a LUT to store a set of neighborhood indices for each pixel. To reiterate, the SIP framework stores pixels beside their eight nearest neighbors in a contiguous sequence.

By extension, when eye tremor is used each initial pixel will have eight contiguous neighbors across nine frames. Therefore, it is only necessary to store one-ninth of a pixel's neighborhood indices in a LUT as illustrated in Figure 3 Then, when an index is retrieved, eight additional neighbors can be traversed via sequential navigation. Despite this benefit, the performance of using a LUT is compromised by the need to retrieve an index from a separate location in memory. For example, in order to retrieve a pixel from an image I, the pixel's index i must be fetched from a LUT L using index j such that $i = L_j$. To alleviate this cost the LUT is used to assign an alias to each neighbor at I_j . This procedure incurs a minor one-time cost (~0.103s given a 2187 × 2187 image) but it allows the neighbor at I_j to be accessed via its alias for the remaining lifetime of the program. The second approach on squiral based, multi-scale feature extraction is to adjust the eye tremor procedure so that the shift between each frame is one-third of the considered neighborhood operator's length. For example, if a 9 × 9 neighborhood operator is to be applied, each frame is captured at a different cardinal or intermediate location, three pixels from the initial center of the sampling lattice. This will center each frame on a different 0 mod 9^{λ} location, where λ denotes the layer of the neighborhood operator minus one. Thus, each pixel with an index 0 mod 9^{λ} is stored adjacent to 9^{λ} - 1 neighbors. As a consequence, only a fraction of pixels can be processed contiguously with their neighbors. Thereafter the equidistant outputs can be consolidated or enlarged by 9^{λ} - 1 pixels to obtain a *Coarse* feature map. This is referred to as non-overlapping neighborhood processing because a neighborhood operator does not overlay any pixel more than once per frame.

0	10	20	30	40	50	60	70	80
10	150	140	20	30	0	70	80	160
20	140	260	380	370	30	0	10	150
30	20	380	370	360	40	50	0	10
40	30	370	360	480	520	510	50	0
50	0	30	40	520	510	580	60	70
60	70	0	50	510	580	620	740	730
70	80	10	0	50	60	740	730	720
80	160	50	10	0	70	730	720	840

Figure 3. Eye Tremor Look Up Table

4. IMPLEMENTATION

To implement convolution on a squiral image, a neighborhood operator must be stored in computer memory as an array of nine element clusters according to the SIP sampling scheme. The first algorithm based on a LUT aliasing approach proceeds as follows: loop through the eye tremor alias array; loop through each cluster in the neighborhood operator; loop through each element in the cluster; and apply each cluster element to a corresponding pixel. Note that in this description (and in the implementation assessed in this paper) a third loop is used to navigate the clusters in a neighborhood operator. Incidentally, it is possible to manually unroll this loop since each cluster is a definitive nine elements long. In the second, non-overlapping algorithm each eye tremor image is navigated in 9^{λ} pixel jumps; for each pixel visited apply the first element of the neighborhood operator; loop through 9^{λ} - 1 adjacent pixels; and convolve with the subsequent neighborhood operator values. A computational analysis of the above algorithms is provided in Table 1, Table 2, and Table 3. The results indicate that the SIP algorithm using the LUT aliasing approach is more expensive than the non-overlapping SIP algorithm, but it is still a respectable improvement over the TIP convolution algorithm**.**

5. PERFORMANCE EVALUATION

In this section, the efficiency of multi-scale SIP based feature extraction is demonstrated using two neighborhood operators: a 3 × 3 Laplace operator; and a 9 × 9 Laplace of Gaussian (LoG) operator. In TIP both operators are applied according to standard convolution theory. In SIP the Laplace operator is applied using the non-overlapping eye tremor approach described in Section 3. The SIP LoG operator is applied using the two methods discussed in Section 3.2: the first method being a *Fine* application using a LUT and pixel aliasing; and the second method being a *Coarse* application using non-overlapping convolution.

5.1 Visual Performance

For visual evaluation, feature maps for the standard test image "peppers" are provided in Figure 4. Feature maps *a* and *b* were obtained using TIP where *a* is the map obtained from the Laplace operator and *b* is the map obtained from the LoG operator. Feature maps *c*, *d*, and *e* were obtained using SIP where *c* is the map obtained from the Laplace operator, *d* is the map obtained from a *Fine* application of the LoG operator, and *e* is the map obtained from a *Coarse* application of the LoG operator. Incidentally, the native resolution of the "peppers" image is 512 × 512 pixels but here it was padded to 729 × 729 (layer 6) pixels to accommodate the sizing criterion of the SIP framework. For visual clarity the outputs were normalized and thresholded. The final results were slightly cropped to discard border anomalies. As expected, TIP maps *a* and *b* are identical to corresponding SIP maps *c* and *d* and have been numerically verified as such. SIP map *e* appears quite similar to maps *b* and *d*. It can also be argued that map *e* is visually much more meaningful than maps *a* and *c*.

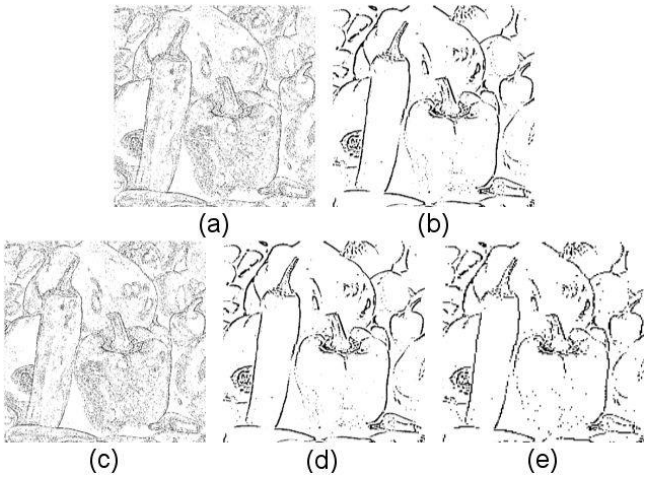

Figure 4. Feature Maps

5.2 Computational Performance

Table 1 provides a computational analysis for a TIP based multi-scale convolution approach, while Table 2 and Table 3 provide a computational analysis for the two SIP based multi-scale convolution approaches. In addition, an example count is provided based on a 3 × 3 image and a 3 × 3 neighborhood operator. In the assessment, a square region with a length *Rl* and an area *Ra* is considered for processing. Likewise, a square neighborhood operator with a length *Kl* and an area *Ka* is used. In the case of squiral convolution, the attribute *Ew* denotes the size of a single eye tremor frame while the attribute *Kh* denotes the number of nine element clusters in a squiral neighborhood operator. The results of the computational analysis demonstrate that the two SIP algorithms require much less computation than the TIP algorithm.

Table 1. Analysis of TIP Convolution

Operation	Assessment	Count
Assignments	1+3*Rl*+3*Rl* 2+3*Rl* 2*Kl*+2*Rl* 2*Kl* 2	280
Comparisons	4+*Rl*+*Rl* 2+*Rl* 2*Kl*+*Rl* 2*Kl*	124
Subtractions	*Rl*+*Rl* 2	12
Additions	*Rl*+*Rl* 2+2*Rl* 2*Kl*+3*Rl* 2*Kl* 2	309
Multiplications	*Rl* 2*Kl* 2	81

Table 2. Analysis of SIP Convolution (*Fine*)

Operation	Assessment	Count
Assignments	1 + 2*Ra* + 3*RaKh* + 2*RaKh*	208
Comparisons	3 + *Ra* + *RaKh* + 9*RaKh*	102
Additions	*Ew* + 2*RaKh* + 27*RaKh*	270
Multiplications	*RaKh* + 9*RaKh*	81

Table 3. Analysis of SIP Convolution (*Coarse*)

Operation	Assessment	Count
Assignments	1 + 3*Ra* + 2*EwKa*	172
Comparisons	2 + *Ra* + *EwKa*	83
Additions	*Ra* + 3*RaKa*	225
Multiplications	*RaKa*	81

5.3 Runtime Performance

Table 4 shows the runtimes for the TIP and SIP convolution approaches discussed at the beginning of this section, with image sizes noted in the leftmost column. The approaches were implemented in C++ 14 and compiled using GNU g++ version 5.4.0 with default optimization. The runtimes were measured in seconds (s) using the CPU clock and are given as the average of one thousand executions. The experimentation system included an Intel Core i7-4790 CPU @ 3.60GHz × 8, 16GB RAM and Ubuntu 16.04 LTS 64-bit (Linux).

Table 4. Feature Detection Runtimes

	TIP		SIP		
	Laplace	**LoG**	**Laplace**	**LoG (*Fine*)**	**LoG (*Coarse*)**
2432	0.002s	0.016s	0.001s	0.012s	0.001s
7292	0.017s	0.139s	0.011a	0.108s	0.012s
21872	0.158s	1.249s	0.098s	0.991s	0.102s

The runtime results show that SIP with eye tremor is faster than TIP for neighborhood operator applications under identical circumstances. Taken individually, an application of the Laplace operator was approximately 40% faster using SIP compared to TIP; an application of the LoG operator was approximately 25% faster using SIP compared to TIP; and a *Coarse* application of the LoG operator using SIP was approximately 1200% faster than the LoG application using TIP. Interestingly, a *Coarse* application of the LoG operator is significantly faster than a *Fine* application. Furthermore, the visual results in Figure 4 show only a slight degradation between the *Fine* and *Coarse* feature maps which is not considered significant. Based on this, it is hypothesized that *Coarse* neighborhood operations could be highly beneficial in machine vision and robotics applications and will be the focus of further work

6. CONCLUSION

The research presented in this paper has demonstrated that the SIP framework can offer a faster alternative to the TIP framework. Moreover, we have demonstrated two alternative ways to navigate neighborhoods at different scales using an eye tremor inspired approach. We assert that these alternatives are more desirable than the squiral based, multi-scale processing method used in the current literature and provide evidence that a spiral framework is advantageous for real time video processing. Therefore, future work will initially focus on real-time video processing using the SIP framework in conjunction with eye tremor. This will lead on to interest point detection for the SIP framework.

7. REFERENCES

[1] Róka, A., Csapó, Á., Reskó, B., Baranyi, P., "Edge Detection Model Based on Involuntary Eye Movements of the Eye-Retina System", Acta Polytechnia Hungarica, vol. 4, no. 1, pp. 31 - 46, 2007.

[2] Middleton, L., Sivaswamy, J., "Hexagonal Image Processing - A Practical Approach", Springer, 2005.

[3] Scotney, B., Coleman S., Gardiner, B., "Biologically Motivated Feature Extraction Using the Spiral Architecture", International Conference on Image Processing, pp. 221-224, 2011.

[4] Jing, M., Scotney, B., Coleman S., McGinnity, M., "A Novel Spiral Addressing Scheme for Rectangular Images", International Conference on Machine Vision Applications, pp. 102 - 105, 2015.

[5] Jing, M., Coleman, S., Scotney, B., McGinnity M., "Multiscale "Squiral" (Square-Spiral) Image Processing", Irish Machine Vision and Image Processing Conference Proceedings, pp. 33 – 40, 2015.

[6] Knuth, D., "The Art of Computer Programming", 3 ed., vol. 1, Addison Wesley, 1997.

[7] Iliffe, J., "The Use of the Genie System in Numerical Calculations", Annual Review in Automatic Programming, vol 2., pp. 1 – 28, 1961.

[8] Fegan, J., Coleman, S., Kerr, D., Scotney B., "Fast Corner Detection Using a Spiral Architecture", Irish Machine Vision and Image Processing Conference Proceedings, pp. 51 - 57, 2016.