Fast Corner Detection Using a Spiral Architecture

J. Fegan,[1] S.A., Coleman,[1] D. Kerr,[1] B.W., Scotney[2]

[1]*School of Computing and Intelligent Systems,*
[2]*School of Computing and Information Engineering,*
Ulster University, Northern Ireland

Abstract

Fast image processing is a key element in achieving real-time image and video analysis. Here, a novel framework based on a spiral architecture is used to facilitate fast image processing, in particular, fast corner detection. Unlike a conventional image addressing scheme where the picture elements are indexed using two-dimensional Cartesian coordinates, a spiral addressing scheme enables the image to be stored and indexed as a one-dimensional vector. Image processing is hastened through the combined use of the one-dimensional structure and a lookup table. The performance is evaluated by the application of a corner detector based on the Harris corner detection algorithm. The results demonstrate the efficiency of the proposed approach compared with a typical two-dimensional implementation.

Keywords: Fast Image Processing, Spiral Architecture, Corner Detection, Lookup Table, Eye Tremor

1 Introduction

Since its inception, digital image processing has largely leaned upon the intuitive notion that two-dimensional (2D) visual data can be sampled as a matrix of picture elements (pixels) using a rectangular lattice of sensors. This has resulted in a Cartesian coordinate system where each pixel is referenced by an index in the horizontal and vertical directions. This approach has worked well for tasks where the time taken to process an image is not a primary concern. However, research has shown that, compared with a one-dimensional (1D) approach, operating on a matrix requires additional computation to locate the pixels in two directions [1]–[5]. This has implications for activities such as video processing where the system in question is expected to operate on a stream of consecutive image frames under strict time restraints. Subsequently there has been a growing interest among researchers to explore alternative image representations. Strategies such as Hexagonal Image Processing (HIP) have demonstrated computational performance improvements resulting in a subsequent runtime advantage over a conventional 2D approach [1], [2]. Despite this fact, existing image capture and processing hardware is predominately based on a rectangular architecture and this has limited the research and practical applications of hexagonal imaging methods [3]–[6]. In response to these concerns a new framework has been proposed, one that attempts to reconcile the prevalent rectangular framework with the strengths of the HIP framework. This new approach uses a newly developed sampling method based on a square spiral (squiral) address scheme that is similar to the hexagonal address scheme of the HIP framework [6]. In this paper, corner detection is used to evaluate the performance and effectiveness of this approach.

2 Squiral Image Processing Framework

Previous research has shown that the Squiral Image Processing (SIP) framework is capable of delivering fast results [3]–[5]. This is partly attributed to the SIP address scheme, which avoids using 2D Cartesian coordinates in favour of a 1D index sequence. More precisely, an image in the SIP framework is sized according to a template called a layer. The first layer (0) is a single pixel located at the centre of the image. The next layer (1) encompasses layer 0 and its 8 surrounding pixels. Thereafter, each subsequent layer encompasses its preceding layer and 8 regions of the same size that surround it. The 9 element makeup of each layer promotes a recursive base 9 address scheme. In this scheme the indexing begins at the centre element of each layer and continues outwards in a clockwise spiral (Figure 1). Ultimately each layer is denoted by the position of a digit in a pixel's index and an element is denoted by the value of that digit. In accordance with this scheme the image is unravelled onto a vector. This means that only a single loop is required to traverse the image for subsequent processing. Despite this benefit, the vectorised nature of a SIP image makes it difficult to locate the neighbours of pixels that are not the centre of a squiral. There are currently two solutions that solve this problem: create and reference a table that stores each pixels Cartesian neighbour locations as SIP indices [1], [2]; find each pixels neighbours by shifting the pixels in various directions, an approach based on a biological process of involuntary eye movements called tremors [7].

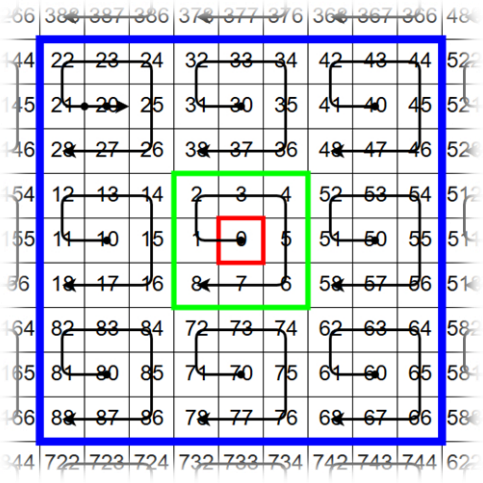

Figure 1: SIP Address Scheme

2.1 SIP Neighbour Lookup Table

	1	2	3	4	5	6	7	8
1	15	14	2	3	0	7	8	16
2	14	26	38	37	3	0	1	15
3	2	38	37	36	4	5	0	1
4	3	37	36	48	52	51	5	0
5	0	3	4	52	51	58	6	7
6	7	0	5	51	58	62	74	73
7	8	1	0	5	6	74	73	72
8	16	15	1	0	7	73	72	84

Figure 2: Neighbour LUT

The first of the two aforementioned solutions relays on a pre-calculated lookup table (LUT) to find the locations of each pixel's Cartesian neighbours in a SIP vector. In this situation each pixel has its own record of neighbour indices in the LUT. Accordingly, a pixel's index is used to access a record and find its neighbour pixels. For example, the LUT in Figure 2 demonstrates that the 8 immediate neighbours of pixel 1 can be found at positions 15, 14, 2, 3, 0, 7, 8, and 16 in the SIP vector. It should be noted that creating the LUT is a one-time procedure and it can be saved and reloaded as required. Incidentally, a single LUT can be used with differently sized SIP images. A notable benefit of this solution is that neighbourhood operations require only two loops. This is opposed to four loops that are common in a typical 2D approach [3], [4]. In the case of neighbourhood navigation, the two neighbourhood loops that are normally used are replaced with one loop that is used to fetch a neighbour index from the LUT. In the case of convolution, the filter in question will need to be vectorised in accordance with the SIP address scheme. This permits the use of a single loop where the index of each filter element is used to retrieve a neighbour index from the LUT. In both cases the use of the LUT presents another advantage by avoiding computation that is otherwise needed to locate a pixel's neighbours.

2.2 SIP Eye Tremor Image Processing

As previously noted, the biological behaviour of eye tremor can be mimicked to find a pixel's Cartesian neighbours in a SIP vector. In this solution the image is sampled once initially before it is offset and resampled several times [1]–[5]. More specifically, in the SIP framework, the initial 2D image is treated as a base which is sampled according to the SIP address scheme. After this the image is offset so that the next element in the SIP address sequence is centred on the rectangular lattice. The offset image is sampled and the process is repeated several times, (Figure 3.1). The output of this procedure is a matrix composed of several SIP vectors [3]–[5]. The diagonal symmetry of this matrix (Figure 3.2.) means that it can be navigated in two different ways: the first way is to loop through each pixel in the base image and refer to the vertically adjacent neighbours in the offset images; the second way is to sparsely process a fraction of the pixels in each image with reference to their horizontally adjacent neighbours. Previous research on edge detection has indicated that this approach is faster than 2D edge detection and SIP edge detection using a neighbour LUT [1], [2]. For this reason, it was the first strategy adopted in the development of a SIP corner detector.

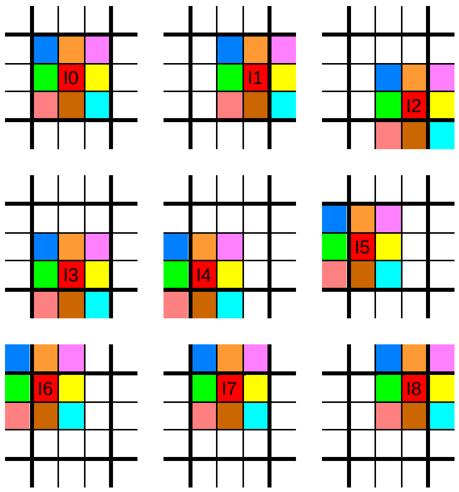

Figure 3.1: Eye Tremor Sampling Scheme

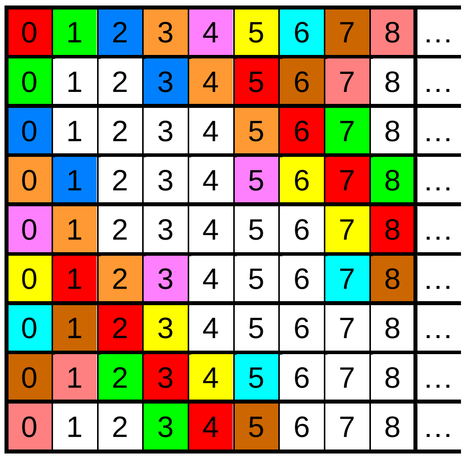

Figure 3.2: Eye Tremor Image

3 SIP Corner Detection

The goal of this research was to investigate the performance of the SIP framework for the application of corner detection. To this end the Harris corner detector was used as a base in the development of the corner detector presented here. As part of this corner detection procedure, the image gradient is computed in two directions. A common way to do this is to apply a gradient filter by convolution [8]. In most cases two filters are used: the first filter computes the gradient in the horizontal direction; the second filter computes the gradient in the vertical direction. Once the gradients are computed they are manipulated and smoothed. This smoothing is typically achieved through another convolution with a Gaussian smoothing filter [8]. However, this presents a problem for the eye tremor approach because a convolution with the gradient filter generates an output where most of the gradient pixels are not adjacent to their Cartesian gradient neighbours (Figure 4). Therefore, the gradient neighbour pixels must be located using base 9 arithmetic, or the eye tremor gradient image must be combined to form a complete 2D gradient image and sampled using the eye tremor scheme. To avoid the computational cost of these solutions, in the work presented here a neighbour LUT was used.

0	1	2	3	4	5	6	7	8	10	11	12	13	14	15	16	…
1	15	14	2	3	0	7	8	16	11	155	154	12	13	10	17	…
2	14	26	38	37	3	0	1	15	12	154	146	28	27	13	10	…
3	2	38	37	36	4	5	0	1	13	12	28	27	26	14	15	…
4	3	37	36	48	52	51	5	0	14	13	27	26	38	2	1	…
5	0	3	4	52	51	58	6	7	15	10	13	14	2	1	8	…
6	7	0	5	51	58	62	74	73	16	17	10	15	1	8	72	…
7	8	1	0	5	6	74	73	72	17	18	11	10	15	16	84	…
8	16	15	1	0	7	73	72	84	18	156	155	11	10	17	83	…

0	1	2	3	4	5	6	7	8	10	…
1	15	14	2	3	0	7	8	16	11	…
2	14	26	38	37	3	0	1	15	12	…
3	2	38	37	36	4	5	0	1	13	…
4	3	37	36	48	52	51	5	0	14	…
5	0	3	4	52	51	58	6	7	15	…
6	7	0	5	51	58	62	74	73	16	…
7	8	1	0	5	6	74	73	72	17	…
8	16	15	1	0	7	73	72	84	18	…

Figure 4: Post Neighbourhood Processing Problem

After the convolution with the smoothing filter, the outputs are used in an auto-correlation function to calculate a corner score for each pixel [8]. After this a threshold is applied to the corner scores to reveal prominent corner pixels. Optimal corner points can then be selected by suppressing all non-maximum corner pixels within a specified neighbourhood region. In the previous steps of the Harris algorithm an 8 neighbour LUT was sufficient to process the image because the operations only concerned a pixel's 8 immediate neighbours. However, to perform effective non-maximum suppression (NMS), it is usually necessary to evaluate a larger neighbourhood. It was thought, at first, that this neighbourhood could be navigated using the 8 neighbour LUT, a pixel's own neighbour record and the neighbour records of other pixels. In practice, problems are caused by the order of the pixels in the SIP vector. For example, in Figure 5a the 9x9 (layer 2) neighbourhood of pixel 10 is found using its own neighbour record and the neighbour records of the 8 corresponding pixels that surround it. However, it is shown in Figure 5b that the 8 corresponding pixels that succeed pixel 10 in the SIP vector are not the pixels that surround it in the 2D plane. This means that computation is needed to find these corresponding pixels and their neighbour records. In this paper this problem was overcome by using a larger, 728 neighbour LUT.

276	268	267	266	388	387	386	378	377	376	368
134	142	143	144	22	23	24	32	33	34	42
135	141	140	145	21	20	25	31	30	35	41
136	148	147	146	28	27	26	38	37	36	48
104	152	153	154	12	13	14	2	3	4	52
105	151	150	155	11	10	15	1	0	5	51
106	158	157	156	18	17	16	8	7	6	58
174	162	163	164	82	83	84	72	73	74	62
175	161	160	165	81	80	85	71	70	75	61
176	168	167	166	88	87	86	78	77	76	68
834	842	843	844	722	723	724	732	733	734	742

(a)

286	278	277	276	268	267	266	388	387	386	378	377	376	368	367	366	488
124	132	133	134	142	143	144	22	23	24	32	33	34	42	43	44	522
125	131	130	135	141	140	145	21	20	25	31	30	35	41	40	45	521
126	138	137	136	148	147	146	28	27	26	38	37	36	48	47	46	528
114	102	103	104	152	153	154	12	13	14	2	3	4	52	53	54	512
115	101	100	105	151	150	155	11	10	15	1	0	5	51	50	55	511
116	108	107	106	158	157	156	18	17	16	8	7	6	58	57	56	518
184	172	173	174	162	163	164	82	83	84	72	73	74	62	63	64	582
185	171	170	175	161	160	165	81	80	85	71	70	75	61	60	65	581
186	178	177	176	168	167	166	88	87	86	78	77	76	68	67	66	588
824	832	833	834	842	843	844	722	723	724	732	733	734	742	743	744	622

(b)

Figure 5: NMS Problem

4 Performance Evaluation

In preparation for testing, several steps were taken to ensure the outcomes of this research would be reliable and fair. The corner detector developed for the SIP framework was adapted from a corresponding 2D counterpart. This was done to minimise syntactical differences that could affect an implementation's runtime performance. A 243x243 pixel (layer 5) image was used as a test case in all the experiments presented here. The Sobel filter was selected to compute the image gradient, and a 3x3 pixel (layer 1) Gaussian filter with a standard deviation of 1 was used for smoothing. A threshold of 70,000 was applied to the Harris response matrix, and a 27x27 pixel (layer 3) neighbourhood region was used for NMS. The convolution was non-separable and three modes were used for all neighbourhood operations: ***Ignore*** pixels with any neighbour indices outside the image bounds; ***Discard*** neighbour indices outside the image bounds; ***Wrap*** neighbour indices that are outside an image border so that they assume an index at the opposite border. For the first two modes a neighbourhood LUT with out of bound neighbour indices was used. For the last mode, a LUT with wrapped index values was used. The corner maps in Figure 5 were produced using the configuration outlined above.

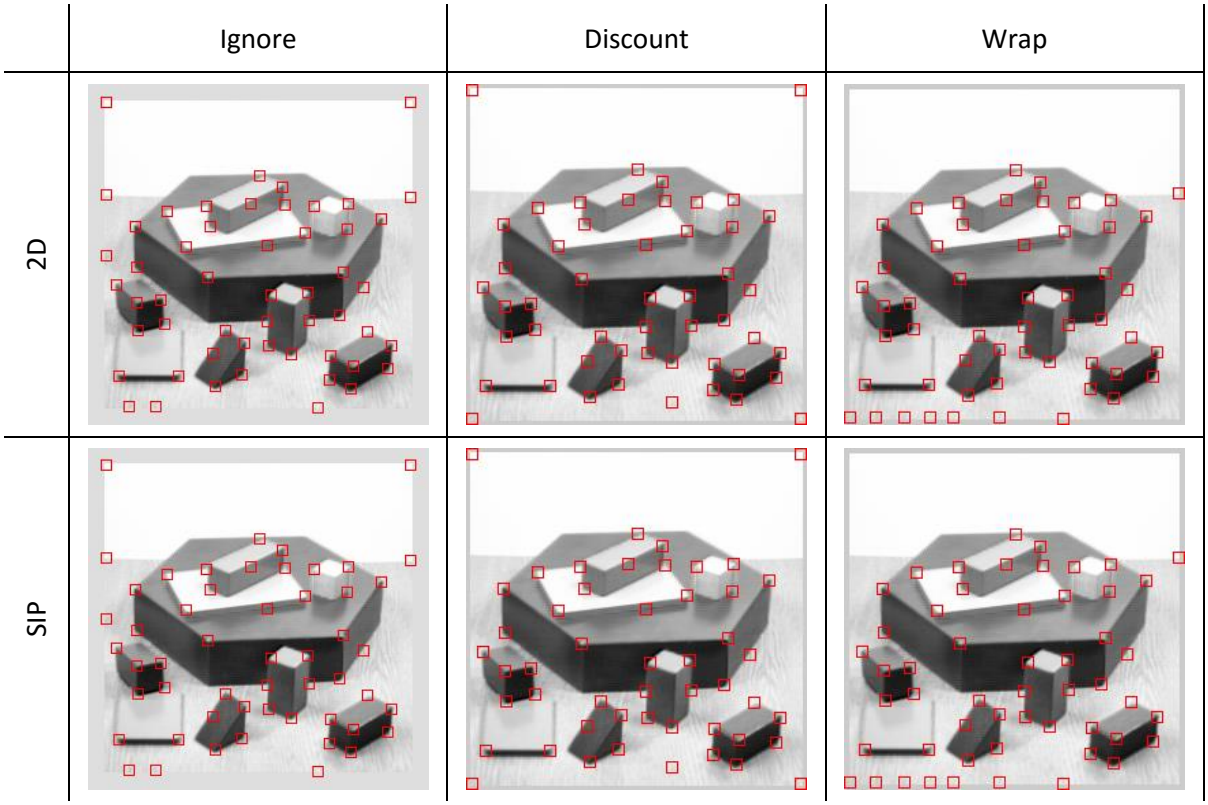

Figure 6: Corner Maps

Note that a few corners have been detected around the image borders; this is a typical response for neighbourhood operations at these regions. Regardless, any issues that these anomalies present can usually be overcome by ignoring or cropping them. As a note for future research, the question is open as to how border pixels in a 1D SIP vector will be handled. For now though, it can be seen that the corner maps of both frameworks are identical. To add further credence to this, the results have been verified as numeric equals in Matlab. This, like the findings in previous SIP research asserts that the SIP framework is capable of delivering outputs that are identical to those produced by a 2D framework [3], [4].

4.1 Runtime Evaluation

The tables below show the times it took to perform SIP conversion, as well as corner detection, in both frameworks with the three modes of neighbourhood operation. For these experiments the runtime assessments were conducted on an Intel Core i7 4790 CPU with 16GB of RAM. The times are based on the average wall-clock times over 1000 runs and were measured using the Matlab functions, *tic* and *toc*. These are the functions recommended by the official Matlab documentation for measuring time reliably [9]. Table 1 shows the runtimes of the conversions to and from a layer 5 SIP image. Table 2 shows the runtimes for the corner detection procedure as measured from the first convolution with the Sobel filter and ending with NMS. The runtime costs for loading the image, setting up filters and displaying the corner maps are not accounted for because they have no bearing on the corner detection process.

2D -> SIP	0.0000095s
SIP -> 2D	0.0006358s

Table 1: Layer 5 Framework Conversion Times

	Runtimes	
	2D	SIP
Ignore	0.3851772s	0.2128920s
Discard	0.5171593s	0.2280309s
Wrap	0.5433419s	0.1678514 s

Table 2: Corner Detection Runtime Results

For the 2D framework with mode Ignore, a 13 pixel border was added to the image to expand it to 269x269 pixels. This was done to account for the 13 neighbour pixels that would extend outside the image during layer 3 NMS. For the same reason, a layer 6 border was added to the SIP image. In both cases, convolution and NMS were restricted to the central layer 5 region. This was done to permit an inbounds convolution on the same data set used by the other modes. Likewise, it keeps the operating conditions across both frameworks as similar as possible. In the case of Discard, a simple boundary check was used to remove neighbour indices that were out of bounds: the 2D framework required four checks to ensure that a neighbour index was within the four image borders; the SIP framework required only one check to ensure that a neighbour index was less than the upper bounds of the SIP vector.

The results show that, compared to a 2D framework, corner detection can be performed much faster on the SIP framework if it is used in conjunction with a neighbourhood LUT. This is observed even if the framework conversion times (Table 1) are summed with the SIP corner detection runtimes. In agreement with previous research it is believed that this performance gain is due, in part, to the noted characteristics of SIP. Namely, that less loops are needed to navigate a vectorised SIP image [3], [4]. The other part of this performance gain is due to the neighbourhood LUT which avoids runtime costs that are normally needed to calculate a pixel's neighbour indices. This is especially notable for neighbourhood operations that use a Wrap mode where the neighbour indices would normally undergo additional calculations to find their circular value.

5 Conclusion

It has been shown that the SIP framework is capable of detecting corners in a way that is equal to a 2D approach, but with significant improvements in algorithmic run-times for non-separable operators. Furthermore, it has been demonstrated that a framework based on a spiral scheme and utilising a neighbourhood LUT is capable of producing fast results. The issues raised in this paper highlight the need to further investigate the eye tremor approach and overcome the current implementation issues and potentially speed up this process further. It could also be useful, in terms of memory allocation, to investigate other methods for addressing SIP neighbourhoods using an 8 neighbour LUT. Future research will extend on the work presented here by developing SIP based Interest Point detectors for applications on video data and ultimately high speed robotic vision challenges.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement No. 607691, SLANDAIL (Security System for Language and Image Analysis). This work was completed under a PhD studentship supported by the Department of Education and Learning (DEL).

The materials presented and views expressed here are the responsibility of the author(s) only. The EU Commission takes no responsibility for any use made of the information set out.

References

[1] B. Scotney, S. Coleman, and B. Gardiner, "Biologically Motivated Feature Extraction Using the Spiral Architecture," in *International Conference on Image Processing*, 2011, pp. 221 – 224.

[2] S. Coleman, S. Bryan, and G. Bryan, "A Biologically Inspired Approach for Fast Image Processing," in *International Conference on Machine Vision Applications*, 2013, pp. 129 – 132.

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

[4] M. Jing, S. Coleman, B. Scotney, and M. McGinnity, "Multiscale 'Squiral' (Square-Spiral) Image Processing," in *Irish Machine Vision and Image Processing (IMVIP)*, 2015.

[5] M. Jing, S. Coleman, and B. Scotney, "Biologically Motivated Spiral Architecture for Fast Video Processing," in *International Conference on Image Processing*, 2015, pp. 2040 – 2044.

[6] L. Middleton and J. Sivaswamy, *Hexagonal Image Processing: A Practical Approach*, vol. 224, no. 4. 2005.

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

[8] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Procedings of the Alvey Vision Conference*, 1988, pp. 147–151.

[9] Mathworks, "Measure Performance of Your Program," *R2016a Documentation*, 2016. [Online]. Available: https://uk.mathworks.com/help/matlab/matlab_prog/measure-performance-of-your-program.html?requestedDomain=www.mathworks.com. [Accessed: 16-Mar-2016].