saLib – A TOOLBOX AND VISUALISATION TOOL FOR IMAGE PROCESSING ON SPIRAL ARCHITECTURE

Stefan Bobe and Gerald Schaefer
School of Computing and Informatics
Nottingham Trent University, Nottingham, UK
e-mail: Gerald.Schaefer@ntu.ac.uk

KEYWORDS

Spiral architecture, hexagonal pixel, image processing toolbox, Matlab.

ABSTRACT

The spiral architecture (SA) represents an alternative model of image representation which uses hexagonal rather than square picture elements. This data structure is not only closely related to biological vision systems but also offers many advantages compared to the normal rectangular representation.
This paper presents saLib, a toolbox and visualisation tool for image processing purposes on the spiral architecture under Matlab. saLib provides basic functionality such as storing, translating and rotating hexagon-based images, as well as methods to display them. Additionally, a visualisation tool is provided which can be used for the convenient operation on images or single addresses and the presentation of the results. saLib is available to fellow researchers for download.

1. INTRODUCTION

The common representation structure for digital images is that of a rectangular grid of square-shaped picture elements. While this has advantages it also has certain drawbacks such as that every pixel has to be identified by its row and column (i.e. two) co-ordinates or that pixels adjacent to one central pixel have different distances to the centre of the central pixel (the diagonal elements are further away than the horizontal or vertical adjacent pixels). To find a better representation, the idea of a hexagonal lattice has been introduced (Sheridan, 1996). In this model the basic picture element has the form of a hexagon. Adjacent hexagons all have the same distance from each other. Furthermore a special addressing algorithm, the so-called spiral architecture (SA), addresses every hexagon with only one co-ordinate and allows the use of a special algebra. This algebra can be used to operate on the address values independent from the actual co-ordinates of their position.
Based on this new data structure several other applications have been developed, providing basic tools for image processing like translation or rotation of image content or the transformation of the rectangular representation to the spiral architecture.
In this paper we present saLib, a Matlab toolbox which provides access to various image processing functionality based on the spiral architecture. SA-based data structures are available as are functions for operations such as image translation and rotation. A graphical user interface is also provided as a visualisation tool to demonstrate the functionality provided.
The rest of the paper is organised as follows: Section 2 provides an overview of the theory behind spiral architecture image processing. Section 3 describes the library part of saLib while Section 4 covers the visualisation tool. Section 5 concludes the paper.

2. THEORY OF SPIRAL ARCHITECTURE

Sheridan (1996) introduced the spiral architecture (SA) as a concept for machine vision on a hexagonal lattice which not only provides a model closer to biological vision systems but also has other advantages over the conventional square pixel-based image representation. The basic picture element in SA has the form of a hexagon. It follows then that each element has six direct neighbouring cells (one on each side) and that these are exactly the same distance away (in contrast to the 8-neighbourhood of a square pixel). While the basic concept of hexagonal pixels was not new, Sheridan's work introduced a special addressing algorithm which identifies each hexagon of the structure with a unique one-dimensional address in base 7. The addressing algorithm, along with special addition and multiplication algorithms, provides the spiral architecture with the algebraic feature of a Euclidean ring. Several other algorithms based on this new data structure have been introduced since.

2.1. Spiral Counting

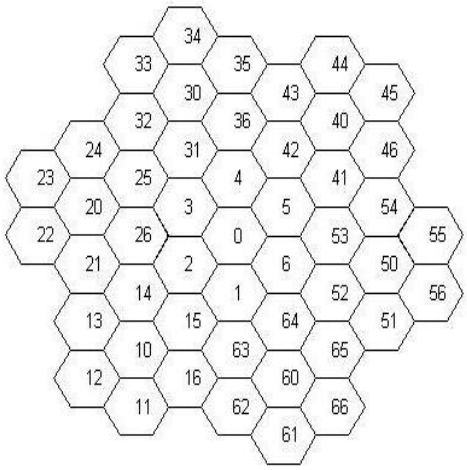
The addresses in SA are in base 7 and arranged in a spiralling way (see Figure 1 for the first 49 elements on the SA). Spiral counting (Sheridan, 1996) can be used to walk along the addresses in spiral architecture consecutively. The algorithm starts at the hexagon with address 0 and goes from this address to the address 1, called the ‘key’ as it defines the rest of the algorithm. Addresses 2 to 6 are arranged in 60° steps clockwise in the same distance as address 1 around the starting address. The next address, 10, is then found by going from 6 to 1 and twice the distance further in the same direction. Addresses 20 to 60 are again arranged in 60° steps clockwise around address 0 with the same distance as 10. At each of these addresses, addresses 11 to 16, 21 to 26 etc are arranged in the same distances

Proceedings 19th European Conference on Modelling and Simulation
Yuri Merkuryev, Richard Zobel, Eugène Kerckhoffs © ECMS, 2005
ISBN 1-84233-112-4 (Set) / ISBN 1-84233-113-2 (CD)

and directions as 1 to 6 around 0. Address 100 can be found in the same way as 10, this time using addresses 60 and 10 instead of 6 and 1, etc.
It was shown (Sheridan, 1996) that spiral counting can be generalised by either a variable starting address or a variable key. These two cases define spiral addition and spiral multiplication respectively.

2.2 Spiral Addition

Spiral addition (Sheridan and Hintz, 1999) of two spiral addresses *x* and *y* is equivalent to spiral counting in the key of 1 starting at address *x* for *y* addresses. Sheridan also suggested a 'carry rule' instead of spiral counting, which allows algebraic operations on the addresses independent from their co-ordinate representation (and is also computational more efficient). For the addition, the two addresses are decomposed to their digits and each pair of digits is added separately, using a special table for spiral addition (Sheridan, Hintz and Alexander, 2000). For results with a 2-digit length the carry rule spiral adds the higher decimal power to the next higher decimal power of one of the values.
A special form of spiral addition is the modulus of it, which uses the normal modulus function with a decimal power as modulus.
The use of modulus spiral addition on the whole spiral architecture results in a translation operation which shifts the contents of the image. Hexagons that would be shifted out of the normal range are wrapped around the whole spiral structure.



Figures 1: The first 49 elements on the Spiral Architecture

2.3 Spiral Multiplication

Spiral multiplication (Sheridan and Hintz, 1999) uses a variable key for spiral counting with the constant starting address 0. Spiral multiplication of *x* and *y* is equivalent to spiral counting of *y* addresses in the key of *x*. Again the two values can be decomposed to their digits, but in this case every digit of the first value is spiral multiplied with every digit of the second value. The spiral multiplication uses again a special multiplication table for each pair of digits (Sheridan, Hintz and Alexander, 2000). Decimal powers are multiplied in normal fashion. Afterwards the results for each digit of the first address are spiral added.
A special modulus function is also defined to reduce the result of spiral multiplication. If the address is not a multiple of 10 then the normal modulus is used. In case the first address is a multiple of 10 the result after spiral multiplication p is reduced by [(*p*+ (*p*/modulus)) mod modulus].
Modulus spiral multiplication can also be applied to the whole spiral structure and leads for a multiplication of address 1 to 6 to a rotation of the image by a multiple of 60°. For other addresses the modulus spiral multiplication produces several rotated copies of the original image or a new distribution of the old hexagons throughout the structure.

2.4. Virtual Spiral Architecture

(Wu, He and Hintz, 2004) introduced an algorithm to translate the rectangular to the hexagonal image structure allowing conventional images to be converted to SA images. The two structures are overlaid and the pixel values of each hexagonal pixel evaluated by interpolating between the pixel values of the underlying square cells. The interpolation is performed by splitting the original pixels into several smaller elements with the same intensity. The values and the number of the underlying points determine the value of each hexagon.

2.5. Log Space Based Transformations

Certain special addresses in the spiral architecture can be used to describe every other address (Sheridan, Hintz and Alexander, 2000). This can be accomplished by repeatedly applying modulus spiral multiplication and modulus spiral addition respectively with the same address. This repeated application cycles through the structure. The number of repeated operations to reach an address depends on the size of the structure and is used as a new address to describe the spiral address. This new address space was termed 'log space'. Repeated modulus spiral addition starts at address 0 and cycles through the whole structure whereas repeated modulus spiral multiplication only cycles through the addresses that are not a multiple of 10. Therefore the addresses that are multiples of 10 have to be split into their order of magnitude and the multiple of it. Not every address can be used to build these log spaces for every size of the structure however certain addresses such as address 1 for spiral addition and address 62 for spiral multiplication can be found. Application of log space transformations results in a vast gain of improvement in terms of computational complexity. Using log space operations such as image rotation can

be performed in a fraction of the time required by conventional methods.

2.6. Rotation Without Scaling

Spiral multiplication on the whole image structure has the restriction that only the spiral multiplication with addresses 1 to 6 results in a rotation (by a multiple of 60°) without a scaling effect on the image. Wu, He and Hintz (2002) developed a special method which allows a rotation by any address or angle without scaling. For this purpose, the rotation angle is split into its multiple of 60° and the remainder. The multiple of 60° is used for spiral multiplication. The rotation by the fraction is then reached by rotating the centre of each hexagon of the whole structure by this angle around address 0. After this rotation each centre of a hexagon gets a new point and for each of these points the closest hexagon has to be found. This so called 'pull back' can map several points to one address whereas other hexagons do not get assigned. Applying this operation to the output image results in the hexagon of the input image.

3. saLib LIBRARY

This paper presents saLib, a Matlab toolbox and visualisation tool for image processing on the spiral architecture. Matlab (Mathworks, 2005) is a matrix-based mathematical programming language and widely used in fields such as engineering and the computer sciences including image processing and machine vision. saLib is intended as a toolbox to be used for image processing and machine vision research based on SA as well as a tool to visualise certain SA operations that can be used for educational purposes.

Table 1: Pre-defined data structures in saLib

Data structure	Description
sa_ADDTABLE	Matrix used for spiral addition .
sa_MULTTABLE	Matrix used for spiral multiplication.
sa_HEXADDR	Position of the hexagons (e.g. for the display with sa_imview()).
sa_LOGADD	Matrices used for log space transformation to shift an image.
sa_LOGMULT	Matrices used for spiral multiplying an image by log space transformation.
sa_OCTAVEPOLS	Contains polar co-ordinates of the first hexagon of each octave.

saLib includes the functionality of all the operations covered in Section 2 as well as special algorithms developed for the understanding of the spiral architecture and the presentation of the images under Matlab. A full list of all functions provided and all data structures of saLib is provided in Tables 1 to 3.
The algorithms that are included in the toolbox can be divided into two groups: those that operate on spiral addresses and those that are applied to the whole spiral structure.
The remainder of this section explains the functionality of the library part of saLib while Section 4 describes the visualisation tool coming with saLib. saLib is available to fellow researchers for download from http://vision.doc.ntu.ac.uk/.

3.1 Displaying the Spiral Architecture

The sa_imview() function is used to display an image on the spiral architecture in order to be able to visualise the outcome of image processing operations on spiral images. Image can be either greyscale or full (RGB) colour images and all image data types provided in Matlab are supported. An example of an image displayed using sa_imview() is shown in Figure 2.

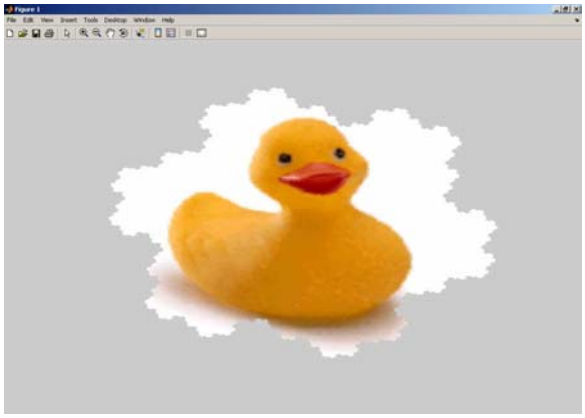


Figure 2: Display of an Image on Spiral Architecture

3.2. Converting between Rectangular and Spiral Images

Since at the moment there is little hardware support for imaging based on hexagonal pixels and as virtually all images in existence are based on the rectangular square-pixel stucture functionality for converting rectangular images to spiral equivalents and back is needed. In saLib the sa_rect2spiral() and sa_spiral2rect() functions provide this functionality of converting from a rectangular image to its spiral counterpart and backwards respectively. For converting square-pixel based images to hexagon-based ones the virtual spiral architecture method described in Section 2.4 is used, i.e. the rectangular and the spiral structure are overlaid with each other, square pixels further divided and then averaged to provide the hexagonal pixel values. In order to obtain an appropriate size of the resulting hexagonal image the size parameter n of the SA is chosen so that the original rectangular images fits into its 7^n hexagonal counterpart. Those hexagons outside the area of the rectangular image are filled with black.

The inverse operation sa_spiral2rect() uses the same method to evaluate the pixel values of the resulting rectangular image. Minimum and maximum co-ordinates on the spiral architecture are used to determine the number of rows and columns for the rectangular image. Pixels outside the hexagonal structure are set to white.
Figures 3 shows a sample image (Figure 3a) that is being transformed to a hexagonal representation (Figure 3b) and then back to a rectangular one (Figure 3c).

Figure 3a: Original rectangular image

Figure 3b: Image from 3a represented in spiral architecture

Figure 3c: Image from 3b transferred back to rectangular image

3.3. Operations on Spiral Addresses

Some of the functions in saLib operate directly on spiral addresses and it is these functions that the methods that operate on whole images rely on. sa_spiralcount() implements spiral counting as discussed in Section 2.1. sa_getval(), sa_nhood() and sa_nhoodval() can be used to return pixel values of certain addresses and their neighbouring hexagons. sa_add() and sa_multiply() provide methods for spiral addition and multiplication whereas sa_modadd() and sa_modmultiply() encapsulate modulus addition and multiplication. Finally, the functions sa_hex2cart() and sa_cart2hex() allow conversion between spiral addresses and Cartesian co-ordinates.

For further details on these functions the reader is referred to Table 2.

Table 2: Address-based functions in saLib

Function	Description
sa_spiralcount()	**spiral counting** Performs spiral counting for a given number of addresses starting at a given starting address and returns the resulting address.
sa_getval()	**get pixel values** Returns the values of a set of given addresses.
sa_nhood()	**neighbourhood addresses** Returns addresses of six surrounding hexagons for a given address.
sa_nhoodval()	**neighbourhood values** Returns pixel values of six surrounding hexagons for a given address.
sa_add()	**spiral addition** Performs spiral addition (see Section 2.2) of two given addresses using sa_ADDTABLE and returns the result.
sa_multiply()	**spiral multiplication** Performs spiral multiplication (see Section 2.3) of two given addresses using sa_MULTTABLE and returns the result.
sa_modadd()	**modulus spiral addition** Performs modulus spiral addition (see Section 2.2) of two given addresses and returns the result.
sa_modmultiply()	**modulus spiral multiplication** Performs modulus spiral multiplication (see Section 2.3) of two given addresses and returns the result.
sa_hex2cart()	**Cartesian co-ordinates from spiral address** Converts a given spiral address to Cartesian co-ordinates.
sa_cart2hex()	**spiral address from Cartesian co-ordinates** Converts a given set of Cartesian co-ordinates to their corresponding spiral address. 1, 2 or 3 addresses are returned, depending on whether the given point is within a hexagon, on the edge, or in the cusp.

3.4. Operations on Spiral Images

Based on the functions discussed above several operations can be performed on complete spiral images. sa_imadd() performs spiral addition on a spiral image which results in a shifting of the image contents. sa_immultiply() allows spiral multiplication on an image thus resulting in a rotation and scaling of the original image. Image rotation without simultaneous scaling can be achieved using the sa_walkjump() and sa_walking() functions. Functionality for log space operations as discussed in Section 2.5 is also provided; images can be translated and rotated/scaled using the sa_imlogadd() and sa_imlogmultiply() functions.
More detailed information on all the functions operating on spiral images is given in Table 3.

Table 3 Image-based functions in saLib

sa_rect2spiral()	**rectangular image to spiral architecture** Converts a rectangular image to its representation in spiral architecture using the virtual SA algorithm (see Section 2.4).
sa_spiral2rect()	**spiral architecture to rectangular image** Converts an image on spiral architecture to its rectangular representation.
sa_iminfo()	**show image info** Returns class, size, and range of a given spiral image.
sa_imview()	**show spiral image** Displays a given spiral image on screen.
sa_imadd()	**spiral addition on image** Performs spiral addition on whole spiral image resulting in a translation of the image content (see Section 2.2).
sa_immultiply()	**spiral multiplication on image** Performs spiral multiplication on whole spiral image resulting in a rotation and scaling of the image content (see Section 2.3).
sa_spiral2alog()	**log space for spiral addition** Converts the spiral architecture to log space for spiral addition (see Section 2.5).
sa_spiral2mlog()	**log space for spiral multiplication** Converts the spiral architecture to log space for spiral multiplication (see Section 2.5).
sa_imlogadd()	**spiral addition on image using log space** Performs spiral addition on a whole spiral image using a log space from sa_spiral2alog(). Works faster than sa_imadd().
sa_imlogmultiply()	**spiral multiplication on image using log space** Performs spiral multiplication on a whole spiral image using a log space obtained from sa_spiral2mlog(). Works faster than sa_immultiply().
sa_walkjump()	**rotation of spiral image without scaling** Allows rotation of a spiral image without scaling according to Section 2.6.
sa_walking()	**rotation of spiral image without scaling** Performs rotation of a spiral image without scaling. Works similar to sa_walkjump() but does not produce gaps in the resulting output image.

4. saLib VISUALISATION TOOL

The toolbox also provides a graphical user interface which allows the user to visualise the results of the functions provided by the library. A screen shot of the visualisation tool is given in Figure 4. The tool is divided into two sections. The top section offers operations that are applied on a complete spiral image whereas the bottom section provides operations on addresses and co-ordinates of their centre.

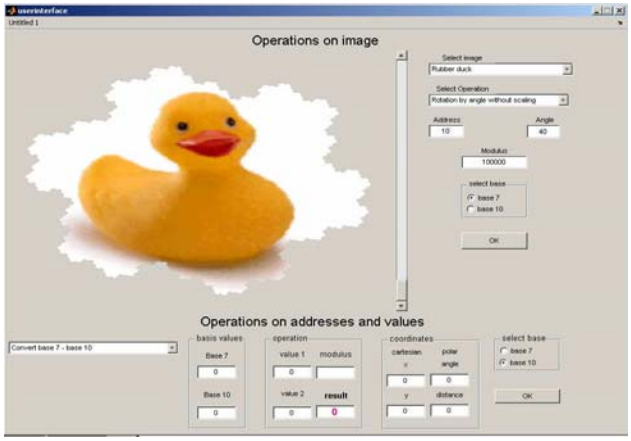


Figure 4: saLib visualisation tool

As indicated, the bottom part of the visualisation tool provides an interface for the functions discussed in Section 3.3. Users have the possibility to enter spiral addresses or co-ordinates and perform the following operations:

- Evaluation of Cartesian and polar co-ordinates for a given address.
- Definition of the nearest hexagon or hexagons for Cartesian co-ordinates.
- Spiral addition of two spiral addresses.
- Spiral multiplication of two spiral addresses.
- Modulus spiral addition of two addresses.

- Modulus spiral multiplication of two addresses.
- Conversion of base 7 to base 10 values and vice versa.

All spiral addresses and modulus can be entered either as base 7 or base 10 values. Results are given in numerical form and displayed. This part of the user interface is indented as an education tool for understanding the basic operations on the spiral architecture.

The top part of the tool allows the visualisation of the effect of spiral operations (discussed in Section 3.4) on whole images.

Operations that can be executed are:

- Shifting of an image by spiral addition (with or without log space transformation).
- Rotation and scaling of an image through spiral multiplication.
- Rotation without scaling of an image, either by a spiral address or by a specific angle, using the sa_walkjump() and sa_walking() functions.

Images of a sample session of the saLib visualisation tool are given in Figures 5a to 5c. Figure 5a shows the result of spiral additon on the Duck image shown in Figure 4. Figure 5b displays the outcome of spiral multiplication of the image resulting in a rotation and simultaneaous scaling of the image. It can be oberserved that 7 downscaled and rotated instances of the original image are generated. Finally, Figure 5c gives an example of rotation without scaling.

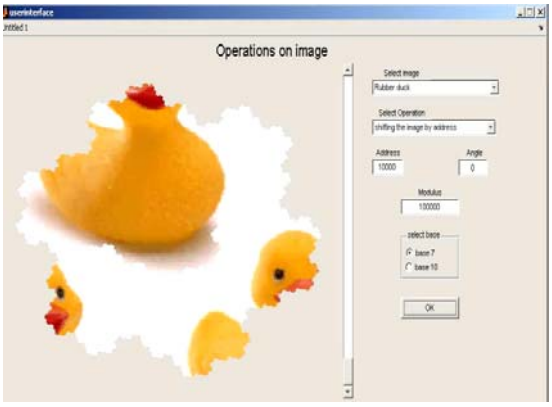


Figure 5a: Duck image after spiral addition of address 10^4

5. CONCLUSIONS

The spiral architecture offers geometrical as well as algebraic features which makes it a powerful new model for representing and processing images. We have introduced saLib, a Matlab toolbox for image processing on the spiral architecture. saLib offers a library of functions for converting and displaying spiral images as well as performing common operations such as translation and rotation of the image content. In addition, saLib provides a graphical tool which allows the visualisation of the functionality provided and which can hence be also employed as an educational application. saLib is available to fellow researchers from http://vision.doc.ntu.ac.uk/.

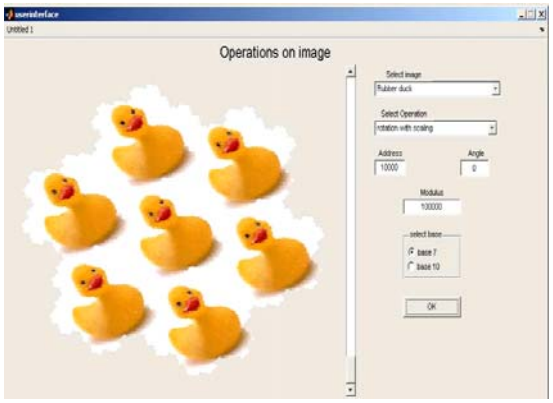


Figure 5b: Duck image spiral multiplied by address 10^5

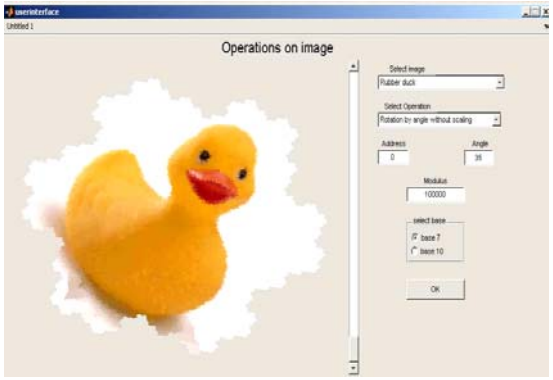


Figure 5c: Duck image rotated (without scaling) by 35°

REFERENCES

Mathworks, 2005 Website: http://www.mathsworks.com.

Sheridan, P., 1996. "Spiral Architecture for Machine Vision", *Ph.D. thesis*, University of Technology, Sydney, Australia.

Sheridan, P. and Hintz, T., 1999. "Primitive Image Transformations on a Hexagonal Lattice", *Technical Report*, Charles Stuart University, Bathurst, Australia.

Sheridan, P., Hintz, T. and Alexander, D., 2000. "Pseudo-invariant image transformations on a hexagonal lattice", *Image and Vision Computing*, Vol. 18, pp. 907 – 917.

Wu, Q., He, X. and Hintz, T., 2004. "Virtual Spiral Architecture", *Proc. Int. Conference on Parallel and Distributed Processing Techniques and Applications*, Vol. 1, pp 399 – 405

Wu, Q. He, X. Hintz, T., 2002. "Image Rotation without scaling on Spiral Architecture", *Proc. Int. Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pp. 515 – 520.