The 1.5D Sieve Algorithm

Clément Fredembach and Graham Finlayson

University of East Anglia

Abstract

The sieve is a morphological scale-space operator that filters an input signal by removing intensity extrema at a specific scale. In images, this processing can be carried out along a path -the 1D sieve- or over a connected graph -the 2D sieve. The 2D version of the sieve generally performs better; it is however much more complex to implement.

In this paper we present the 1.5D sieve, a Hamiltonian path-based version of the sieve algorithm that behaves "in between" the 1D or 2D sieve algorithms, depending on the number of paths used. Experiments show that its robustness to the presence of noise and its performance in texture classification are similar to the original 2D sieve formulation, while being much faster and simpler to implement.

1 Introduction

The sieve algorithm uses morphological filters to remove signal extrema at different scales. The filters used are successive *max* and *min* operators that are applied on windows of different sizes; the size of the window corresponds to the sieving scale. The sieve output is causal: edges vanish as the scale increases and no new image structures, such as edges or extrema, are introduced [4].

Morphological scale-space operators have been successfully applied to various areas of image processing, such as image denoising [13,23], segmentation [6,12], and texture classification [20,21]. In the latter case, while the sieve outperformed other texture classification algorithms, the optimal version to use (1D or 2D) depends on the type of texture analysed. Additionally, sieves have also shown to be very resilient to noise [17].

Email address: clement.fredembach@epfl.ch graham@cmp.uea.ac.uk (Clément Fredembach and Graham Finlayson).

This paper presents the 1.5D sieve, a path-based algorithm that behaves in between the 1D and 2D sieves. The 1.5D sieve is strictly equivalent to the 1D sieve when a single path is used, and is equivalent to the 2D sieve when a large number of Hamiltonian paths are used. The 1.5D sieve, being a path-based approach as opposed to an connected-area approach, is much less complex than the 2D sieve, both from a mathematical and implementation perspective.

We illustrate the performance of our algorithm by replicating the experiment of Harvey et al. on the sieve's robustness to noise [17], and Southam and Harvey's experiment on texture classification [19]. In both cases, we show that the 1.5D sieve results are comparable to the more complex 2D algorithm.

This article is organised as follows: Section 2 reviews the concepts of 1D and 2D sieves, and introduces Hamiltonian paths. In Section 3, we describe our 1.5D sieve algorithm. Section 4 deals with the theoretical equivalence of the 1.5D sieve to the 1D and 2D algorithms. Results on noise robustness and texture classification are presented in Section 5. Section 6 concludes the paper, while Appendix A illustrates the random Hamiltonian path generation algorithm.

2 Background

The 1D sieve output Y of a signal X at a scale S is calculated in two passes. In each pass, minima and maxima of length $\leq S$ are detected and removed. The passes are:

$$Y_{\text{tmp}} = \min_{f}(\max_{b}(X)) \tag{1}$$

$$Y = \max_{f}(\min_{b}(Y_{\text{tmp}})) \tag{2}$$

where f and b define a respectively forward or backward centred window of size $S + 1$. The succession of equations (1) and (2) is also known as an $\mathcal{M}$ sieve. The sieve can also be computed by removing maxima first, i.e., reversing the order of eqs. (1) and (2). This operation is also known as an $\mathcal{N}$ sieve [2]. An illustration of the ($\mathcal{M}$) 1D sieve algorithm is shown in Fig. 1.

The 2D sieve works on the same principle, but uses a more complex graph connectivity-based approach where, for a 2D signal, extrema are detected and removed if their connected *area* is $\leq S$. This process involves complicated "bookkeeping" because extrema can have complicated shapes (e.g., a "$\mathcal{Q}$"-shaped region with 100 pixels is an extrema when $S = 100$) and the connectivity of the region itself, as well as all its neighbours, has to be kept at all times. While efficient data structures can be used, creating and linking them can be time consuming [4]. The 2D algorithm is illustrated in Fig. 2.

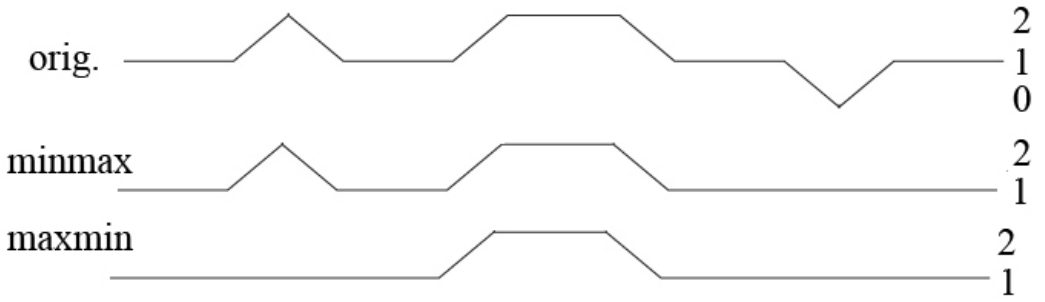

Fig. 1. *A 1D signal (first row) and the result of the sieve algorithm at scale one after each of the passes described in equations (1) and (2)*

Fig. 2. *A synthetic image (left) and the output of the 2D sieve algorithm after the minmax (middle) and maxmin (right) steps at a scale equivalent to the area of the black region.*

Complexity-wise, the 1D sieve is $O(N)$ where N is the number of pixels in the image. The 2D sieve's complexity is higher than $O(N)$, although no precise figure has been given by the authors [3]. The 1.5D sieve has a complexity of $O(MN)$, where M is the number of paths used (typically 20-30, and thus $M << N$).

Other scale-space approaches propose using a tree structure, rather than a planar connected graph, such as Salembier and Garrido Max tree [18] or Bangham et al. sieve trees [6,5,7]. Their complexity varies between $O(GN)$ for the max-tree (where G is the number of gray-levels in the image) to $O(N^2 logN)$ for other approaches [14]. For a detailed review of these algorithms and comparative performance, we refer the reader to [14].

2D sieves frequently use graphs to represent images, a process illustrated in Fig. 3. On this graph representation, Hamiltonian paths are defined as: "a path in a graph such that all vertices are visited once and once only." [11]. Examples of Hamiltonian paths are illustrated in Fig. 3(b) and (c). These paths are used to process an image in a one-dimensional manner; indeed, raster-type paths (a sub-class of Hamiltonian paths) are the ones generally used by the 1D sieve algorithm when applied to images.

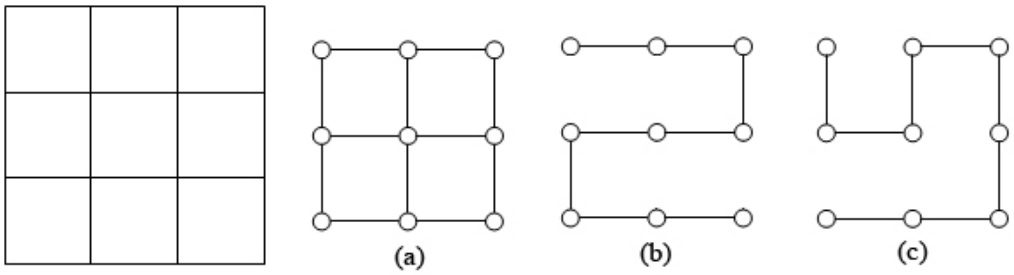

Fig. 3. *Schema of a 3x3 image and corresponding graphs.* ***(a):*** *The graph associated with the image.* ***(b):*** *A raster path over the graph. This type of Hamiltonian path is used in the original 1D sieve.* ***(c):*** *A more general Hamiltonian path over the graph.*

3 The 1.5D Sieve

While the 1D sieve is simple, it is of limited use in image processing. The 2D sieve, on the other hand, is more versatile but requires complex data structures that have to be created and maintained. We propose that using multiple random Hamiltonian paths in a conservative framework can approximate the behaviour of the 2D sieve while retaining the simplicity of the 1D algorithm.

The 1.5D sieve algorithm proceeds as follows: let X be the input image and let us compute M random Hamiltonian paths: $p_1, \ldots, p_M$ over X using the method of [10] (see Appendix A for more details). We define $p_i(X)$ as the "vectorisation" of X along the path p_i (this property is illustrated in Fig. 4).

The 1.5D sieve algorithm is defined as:

$$p_i(y^i_{\text{tmp}}) = \min_f(\max_b(p_i(X))), \quad i = 1, \ldots, N \tag{3}$$

$$Y_{tmp} = \min_{i \in N}(y^i_{tmp}) \tag{4}$$

$$p_i(y^i) = \max_f(\min_b(p_i(Y_{\text{tmp}}))), \quad i = 1, \ldots, N \tag{5}$$

$$Y = \max_{i \in N}(y^i) \tag{6}$$

Where y^i_{tmp} and y^i are the images resulting from sieving along the path p_i (illustrated in Fig. 4).

If $i = 1$ (i.e., there is only a single path), the 1.5D algorithm is exactly equivalent to the 1D sieve: eqs. (3) and (5) are equal to eqs. (1) and (2), while eqs. (4) and (6) have no effect.

However when $i > 1$, eqs. (4) and (6) act as a "conservative check". The algorithm will consider that a given region is an extremum only if *all* the M paths agree. This behaviour is illustrated in Fig. 4, where two paths consider a region to be an extremum at a certain scale while a third path does not. This conservative behaviour is in accordance with the sieve property of causality: no new extrema must be created [2].

Note that our 1.5D sieve algorithm is readily parallelisable; indeed, the steps described by eqs. (3) and (5) can be carried out independently for each of the M paths, their output being combined at the end. This is not the case with the 2D sieve

To create the random Hamiltonian paths we use the method presented in [10], an overview of which is given in Appendix A. This method was chosen because its complexity is linear with respect to the number of pixels in the image and, importantly, because it enables many different random paths over a single image. In contrast, other methods either find a single path per image [1,9] or, like the Christofides algorithm, have a $O(N^2)$ complexity [8].

4 Equivalence of the 1.5D to the 1D and 2D sieves

As seen in the previous section, when a single path is used, the 1.5D formulation is strictly identical to the 1D sieve. The equivalence of the 1.5D to the 2D sieve is achieved if every region of an image admits a path that visits all its pixels sequentially, and, if said paths can be found.

Let I be an image and I_2 be the expanded (by a factor of 2) version of I. Let R be a region of I; it was shown in [10] that R_2, the expanded version of R admits an Hamiltonian circuit. Let us now define $\overline{R}$ to be the complement of R in I. Since $\overline{R}$ is itself a region of I, it follows that $\overline{R_2}$, its expanded equivalent, also admits a Hamiltonian circuit. A schema illustrating these variables is shown in Fig. 5.

Moreover, in 4-connected grid graphs, two disjoint, adjacent Hamiltonian circuits can be merged into a single, encompassing one [8]. It follows that there exist at least one Hamiltonian circuit over I_2 such that the whole of R_2 will be explored sequentially. By using that circuit as a path in the 1.5D sieve algorithm, we ensure that the exact scale of R_2 will be found.

The considered algorithm to generate Hamiltonian paths uses a spanning tree construction; it is able to generate as many different paths over I_2 as there are different spanning trees in I (for more details see [10] and Appendix A). Thus, generating all possible paths on I_2 ensures the exact scale of every region is

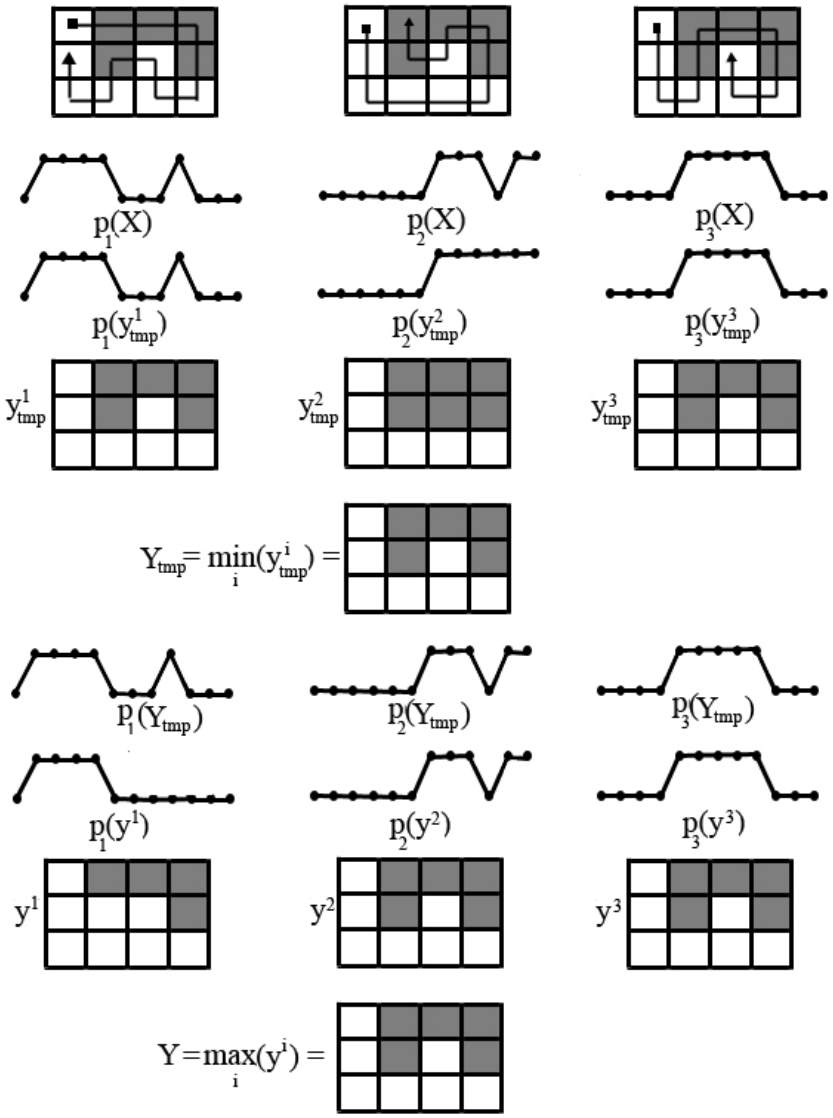

Fig. 4. *An illustration of the 1.5D algorithm using 3 different paths. The paths are impressed over a binary image (gray=1, white=0); the square represents the starting point of the path, the triangle the end. We observe that when paths disagree the route of conservatism is chosen. This is to ensure that no additional image structures will be created, in accordance with the sieve decomposition theorem.*

found. In this limiting case, the 1.5D sieve is therefore equivalent to the 2D algorithm.

This is a theoretical equivalence only, because the number of possible paths, M_{max}, is commensurable. In [22] it has been shown that for a grid graph comprising N vertices (i.e., an image with N pixels), there were $e^{1.16N}$ possible spanning trees, so $M_{max} = e^{1.16N}$, which considering the usual size of images

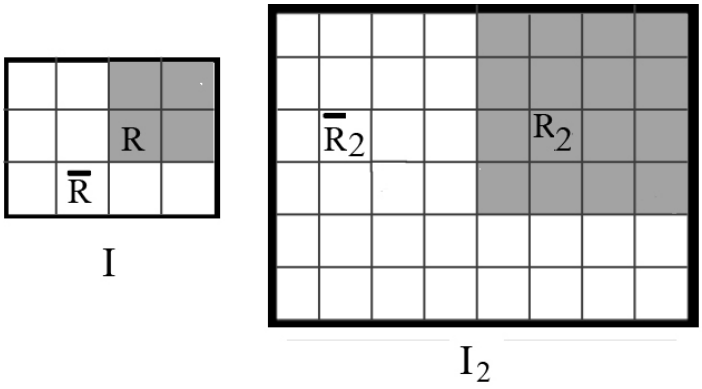

Fig. 5. *A* 3×4 *image* I *and its* 6×8 *expanded version* I_2*. A sample region* R *of* I *and its corresponding expanded version* R_2 *are shown, as well as their respective complements* $\overline{R}$ *and* $\overline{R_2}$*.*

is impossible to compute.

In reality, a number of paths equal to the number of an image's regions N_R is sufficient to correctly assess the scale of every region, but this number can still be quite large and its computation expensive. In practice, we will use M paths where:

$$1 < M < N_R << N << P_{max} \tag{7}$$

Our algorithm will therefore behave in between the 1D and 2D sieves.

5 Experiments

In this section, we evaluate the performance of the 1.5D sieve algorithm. Fig. 6 shows the output of the 1D, 2D and 1.5D sieves at different scales and for different numbers of paths.

5.1 Robustness to Noise

The first experiment is to assess the 1.5D sieve's robustness to noise. To do so, we use the framework of Harvey et al. [17] where the variance of scale estimation is computed in the presence of Gaussian and Impulse noise.

Sample images from this test are shown in Fig. 7. The baseline image consists of a grayscale disc cast on a uniform background. The diameter of the disc is 50 pixels and its amplitude is 144. The size of the background target is 100×100 pixels and its amplitude is 112. This image is then corrupted by

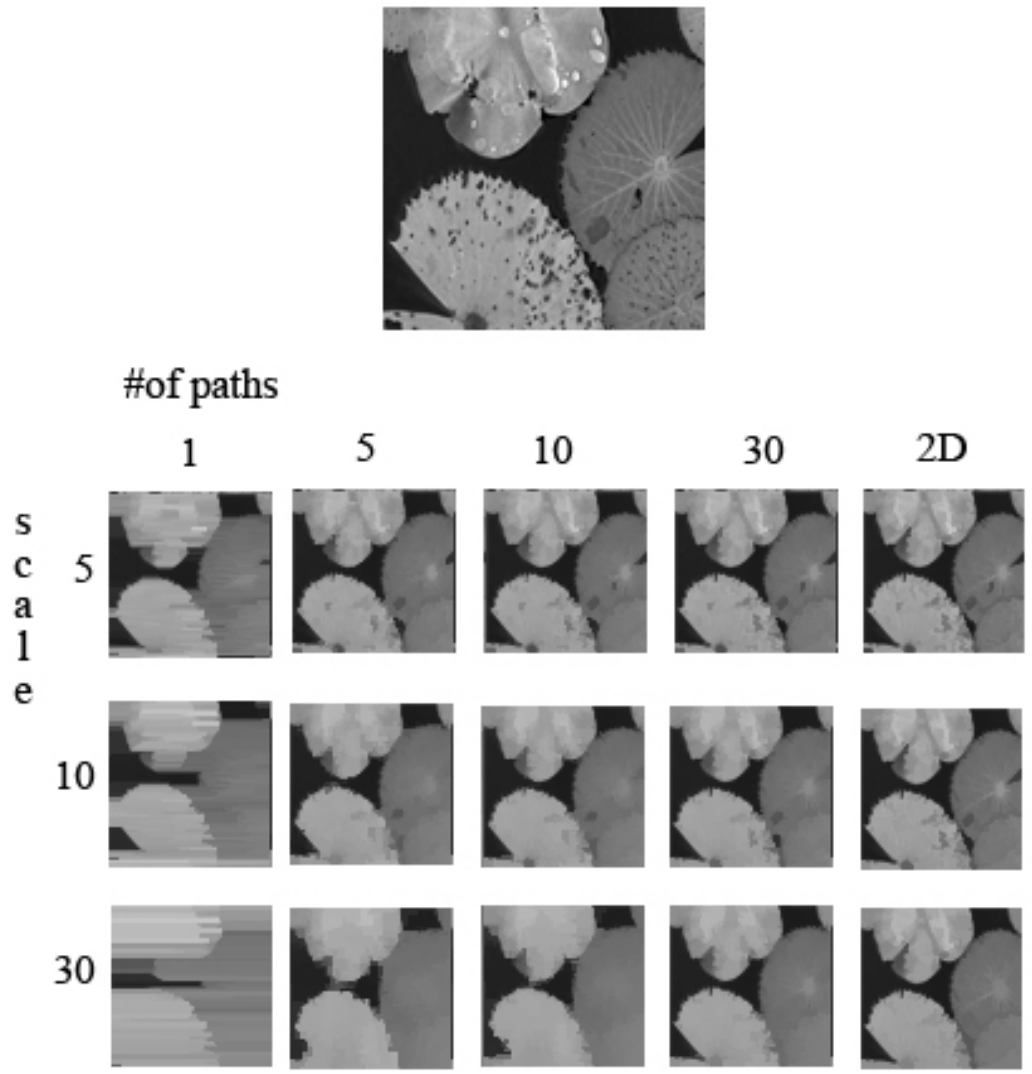

Fig. 6. *The results of sieving an image with the 1D (first column), 1.5D (with 5, 10 and 30 paths) and 2D (last column) sieve at various scales. The size of the image is* 64×64*; for the 1.5D sieve, we see that the more paths used, the closer the results are to the 2D algorithm.*

either uncorrelated Gaussian noise ($\mu = 0,\ \sigma = 24$) or, alternatively, Impulse noise, where pixels are replaced with a random value in the range [0,255] and a noise density of 0.2.

We then sieve the image at all possible scales; the scale at which the largest area is detected is assumed to be the scale of the disc (which is true in the noiseless case). The disc centre is then calculated as the centre of mass of the detected area. Repeating this experiment over 150 instances of noise, we calculate the standard deviations of the estimated circle position (x and y) and scale: σ_x, σ_y and σ_s. The smaller this standard deviation, the more robust the algorithm is in the presence of noise.

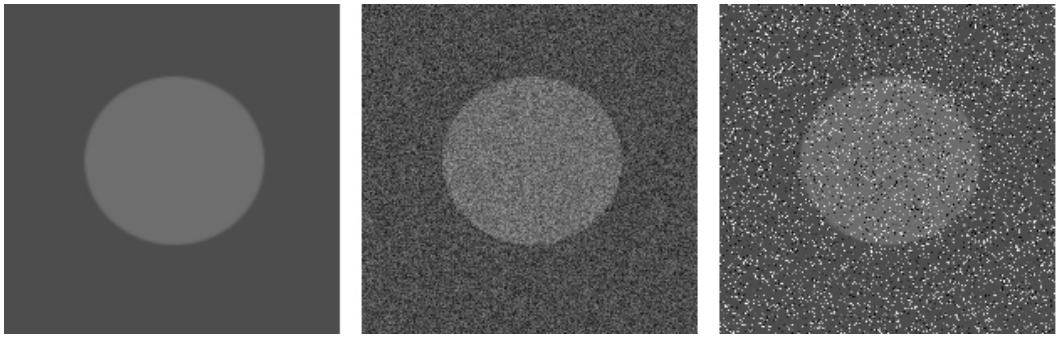

Fig. 7. *The target image used in the robustness experiment (left); corrupted with Gaussian noise (middle) and Impulsive noise (right)*

	Gaussian Noise				Impulsive Noise			
	2D	$1.5D_5$	$1.5D_{10}$	$1.5D_{30}$	2D	$1.5D_5$	$1.5D_{10}$	$1.5D_{30}$
σ_x	0.280	9.3	5.32	0.270	0.0425	9.12	4.76	0.046
σ_y	0.243	9.28	5.4	0.22	0.0416	9.32	4.68	0.0472
σ_s	55	235	102	48	3.91	202	47	4.01

Table 1
Standard deviations of scale and position estimates in the presence of Gaussian and Impulsive noise for the 2D sieve and the 1.5D sieve using 5, 10, and 30 paths.

Table 1 shows the results for both the 2D and the 1.5D (using 5, 10 and 30 paths) sieves. The difference between the two algorithms diminishes as more paths are used. With 30 paths the 1.5D sieve is almost equivalent to the 2D algorithm.

5.2 Texture Classification

The second experiment compares the performance of the 1.5D with the 1D and 2D sieves for texture classification. A comprehensive review of state of the art algorithms carried out in [19] showed that, for the Outex_TC_00000 and Outex_TC_00010 test suites [15,16], the best performing algorithms were the 1D and 2D sieve, respectively. The test suite TC_00000 is rotationally invariant and uses leave-out half cross validation (in all the Outex test suites the training and testing sets are provided). The TC_00010 training set consists of 480 textures from 24 classes, imaged at a fixed orientation of 0^o. The testing set comprises 3840 images of the same 24 classes but the textures are rotated by 5^o, 10^o, 15^o, 30^o, 45^o, 60^o, 75^o and 90^o. Consequently, one path will not suffice to discriminate between rotated textures. For this test, the 1D sieve feature vector is actually composed of six different raster-type paths oriented at every 15^o [19]. Sample textures are shown in Fig. 8.

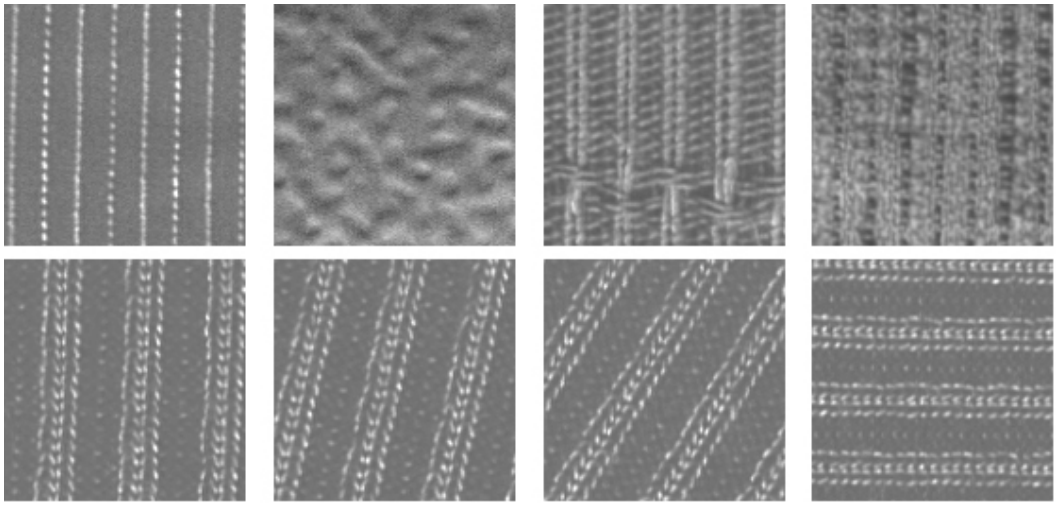

Fig. 8. *Sample textures from the Outex TC_00000 suite (first row) and a texture under various orientations in the Outex TC_00010 suite (second row).*

The texture analysis is carried out using scale granularity. For each algorithm: 1D, 1.5D and 2D sieve, the texture images are sieved at scales 1, 2, 5, 13 and 30. From these six images (the original one and the five sieved images), we generate granularity images by taking the absolute difference of two consecutive scales: original and scale one, scale one and scale two, scale two and scale five, etc. The first three moments (mean, standard deviation, and skewness) of each granularity image are then calculated, thus resulting in a 15-dimensional feature vector. In the 1D case, since there are six independent paths, the feature vector will have 90 entries.

A feature vector is created for each image in the training set, then, each test image's feature vector is calculated and its Euclidian distance to the training data is used to determine which class the test image belongs to. A more detailed explanation of this procedure can be found in [17].

In the 1.5D case, since the textures are strongly oriented, we use "masks" to guide the paths. Specifically, we create gradient images that will be the input of the random Hamiltonian paths algorithm, i.e., the gradients of the masks will be the labels of the graph's edges prior to computing the minimum spanning tree (see Appendix A for details). As a result, the paths will loosely adopt the masks' orientation. In this experiment, we will use 24 paths based on 12 rotations of the masks at regular angles (we create 2 paths per angle). Some orientations of the masks are shown in Fig. 9.

The 1.5D sieve algorithm therefore yields a 15-dimensional feature vector, the same size as the 2D sieve, that is used to classify textures. The Outex sets have defined training and testing procedures that we follow. The results, Table 2, illustrate that the 1.5D sieve algorithm performs "in between" the 1D and 2D methods and displays a generally good performance. Indeed, the

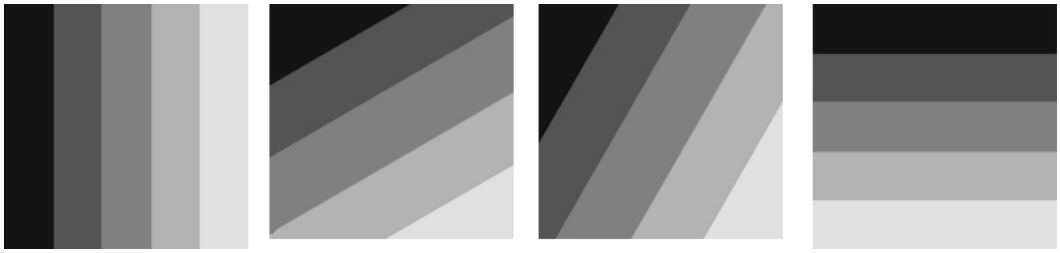

Fig. 9. *Various orientations of masks used in the creation of random Hamiltonian paths.*

TC_00	Success Rate	vector length	TC_10	Success Rate	vector length
1D	**0.998**	90	1D	0.718	90
1.5D	0.99	15	1.5D	0.902	15
2D	0.95	15	2D	**0.943**	15

Table 2
Recognition rates for both the rotationally invariant and variant Outex sets with 1D, 1.5D (using 24 paths of various orientation) and 2D sieves.

1.5D sieve's average performance over both texture sets is almost identical to the 2D algorithm and much higher than the 1D sieve.

6 Conclusion

We have shown that multiple random Hamiltonian paths could be used in the sieve framework. Using multiple paths and adopting a conservative approach, a 1.5D sieve algorithm (named as such because it behaves as either the 1D or 2D sieve in its limiting case) was developed.

Advantages of our algorithm are that it only takes about 20 lines of Matlab code to implement: the 2D sieve algorithm is considerably more complex to code. Also, the 1.5D sieve algorithm is readily parallelisable and has a linear (with the number of pixels) complexity.

Results using relatively few paths show that it can be as robust as the 2D sieve while performing comparably in the task of texture classification.

Appendix A

In this appendix we give an overview of the random Hamiltonian path algorithm. For a detailed analysis and a proof that the algorithm indeed always

yield a Hamiltonian path, we refer the reader to [10].

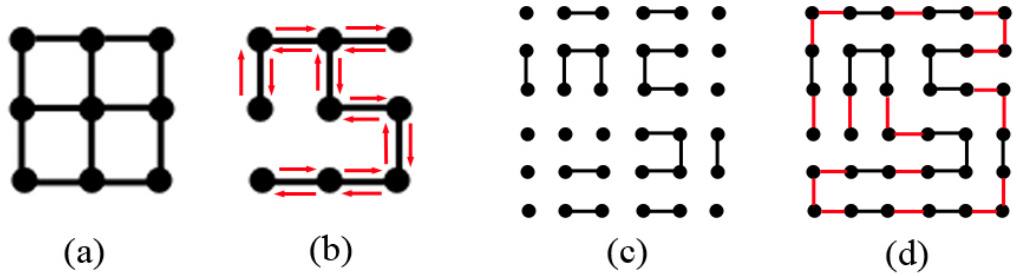

Fig. 10. ***(a):*** *the considered graph G.* ***(b):*** *a spanning tree T obtained by weighting G randomly and an illustration of its "walk around" (arrows).* ***(c):*** *the expanded version of T where the connectivity between the groups of nodes is kept.* ***(d):*** *the Hamiltonian path, obtained by "walking" in the same manner than on T. Note the shape similarity between the Hamiltonian path and the minimum spanning tree.*

Let us consider an image I and its graph representation G. The first step is to construct the minimum spanning tree of G: T. The construction of T is important since the shape of the Hamiltonian path depends on the shape of T. In general, we want this shape to be random; this randomness allows us to derive the 1.5D equivalence to the 2D algorithm shown in section 4.

To ensure randomness, we weigh the edges of G with random weights prior to computing T. In the texture classification experiment, we used a more "structured" approach and used the masks's gradients as weights. The root of the spanning tree is always chosen at random, we have found it to be of negligible influence in our experiments.

To find the Hamiltonian path that corresponds to T, we first expand T and then complete the missing edges by "walking around" the tree. An illustration of the whole procedure is shown in Fig. 10, while more details about the algorithm and its linear complexity can be found in [10].

References

[1] G.S. Adhar. Optimal parallel algorithms for cut vertices, bridges, and hamiltonian path in bounded interval tolerance graphs. In *Proc. of the Eighth International Conference on Parallel and Distributed Systems. (ICPADS)*, pages 91–98, 2001.

[2] J.A. Bangham, P. Chardaire, C.J. Pye, and P.D. Ling. Multiscale nonlinear decomposition: The sieve decomposition theorem. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18:529–539, 1996.

[3] J.A. Bangham and R. Harvey. Avoiding windows and building trees: graph methods and morphology in vision. British Machine Vision Association Meeting

(www.bmva.ac.uk), 2002.

[4] J.A. Bangham, R. Harvey, P.D. Ling, and R.V. Aldridge. Morphological scale-space preserving transforms in many dimensions. *Journal of Electronic Imaging*, 5:283–299, 1996.

[5] J.A. Bangham, J. Hidalgo, and R. Harvey. Robust morphological scale-space trees. In *Proceedings Noblesse Workshop on Non-Linear Model-Based Image Analysis (NMBIA)*, pages 133–139, 1998.

[6] J.A. Bangham, J. Hidalgo, R. Harvey, and G. Cawley. The segmentation of images via scale-space trees. In *Proc. of the British Machine Vision Conference*, pages 33–43, 1998.

[7] J.A. Bangham, K. Moravec, R. Harvey, and M. Fisher. Scale-space trees and applications as filters, for stereo vision and image retrieval. In *Proc. of the British Machine Vision Conference*, pages 113–122, 1999.

[8] B. Bollobas. *Graph Theory*. Springer Verlag, 1979.

[9] S.D. Chen, H. Shen, and R. Topor. An efficient algorithm for constructing hamiltonian paths in meshes. *Parallel Computing*, 28:1293–1305, 2002.

[10] C. Fredembach and G.D. Finlayson. Hamiltonian path based shadow removal. In *Proc. of the 16th British Machine Vision Conference (BMVC)*, pages 502–511, 2005.

[11] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[12] D. Gimenez and A.N. Evans. Colour morphological scale spaces for image segmentation. In *Proc. of the British Machine Vision Conference*, pages 909–918, 2005.

[13] D. Gimenez and A.N. Evans. Colour morphological scale-spaces from positional colour sieve. In *Proc. of the IEEE DICTA*, 2005.

[14] A. Meijster and H.F. Wilkinson. A comparison of algorithms for connected set openings and closings. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24:484–494, 2002.

[15] T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen, and S. Huovinen. Outex- new framework for empirical evaluation of texture analysis algorithms. In *Proceedings of the 16th International Conference on Pattern Recognition*, pages 701–706, 2002.

[16] Outex. http://www.outex.oulu.fi.

[17] R.Harvey, A.Bosson, and J.A.Bangham. The robustness of some scale-spaces. In *Proc. of the British Machine Vision Conference*, pages 11–20, 1997.

[18] P. Salembier and L. Garrido. Binary partition tree as an efficient representation for image processing, segmentation and information retrieval. *IEEE Trans. on Image Processing*, 9:561–576, 2000.

[19] P. Southam. Texture analysis with the sieve. Ph.D. Thesis, University of East Anglia, Norwich, U.K., 2006.

[20] P. Southam and R. Harvey. Compact rotation-invariant texture classification. In *Proc. of the International Conference on Image Processing*, pages 3033–3036, 2004.

[21] P. Southam and R. Harvey. Towards texture classification in real scenes. In *Proc. of the British Machine Vision Conference*, pages 240–250, 2005.

[22] F.Y. Wu. Number of spanning trees on a lattice,. *J. Phys. A: Math. Gen.*, 10:L113–L115, 1977.

[23] N. Young and A.N. Evans. Image noise reduction using attribute morphology filters. In *Proc. 6th IEEE-EURASIP workshop on non-linear signal and image processing*, 2003.