Constrained Pseudo-Brownian Motion for Image Enhancement

Roberto Montagna and Graham D. Finalyson; University of East Anglia; Norwich, UK

Abstract

The retinex theory of colour vision simulates certain features of the human visual system and is used as a method for image enhancement. Since its introduction in 1971 many flavours of retinex have been devised, which implement slight variations of the original concept. For example, Marini and Rizzi developed a retinex algorithm based on Brownian motion paths. However, while their approach delivers interesting results, it has a high computational complexity. We propose an efficient algorithm that generates pseudo-Brownian paths with a very important constraint: we can guarantee a lower bound to the number of visits to each pixel, as well as its average. We then present a retinex implementation that exploits the paths generated with this algorithm. In order to keep the number of visits per pixel low, we take a multi-scale approach: our path-based computation is performed at different scales of the input image and each result is averaged with the result obtained from larger scales. In doing so, we in effect combine the Brownian motion approach with the scale-space method of the McCann99 algorithm.

In this paper we describe the details of our path generator, and we show some images processed with our retinex implementation compared with those obtained with the McCann99 retinex algorithm. The results show that, in general, the Brownian motion approach requires a smaller number of pixel comparisons per scale to achieve similar output images.

Introduction

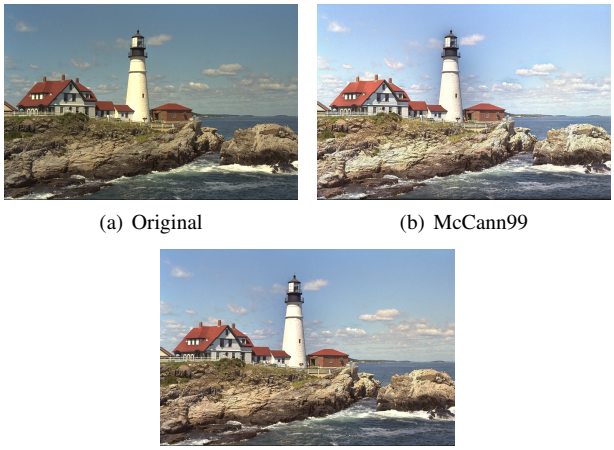
The retinex theory of colour vision, first introduced by Land and McCann [15], was devised in an attempt to model certain features of human colour vision. With the advent of digital imaging, it found several applications. Image enhancement and restoration [1, 21] is its most typical use, but it proved valuable as a tone-mapping operator for high dynamic range images [12, 20, 22]. Broadly, we can divide retinex algorithms in two classes: scale-space [1, 8, 19] and path-based [15, 17]. While in this paper we are not concerned with the details of how retinex works, we can say that the main idea of scale-space algorithms is to start from a thumbnail of an image. The result of the processing on that is then brought up to a larger scale, where it is averaged with the result computed at that scale. This process is repeated up to the full-scale image. This kind of algorithm is very efficient and therefore it has been adopted by HP for their digital still cameras [22]. Path-based retinex algorithms, instead, generate several paths for each pixel, and the final pixel value is estimated by averaging the results of the computations along each path. For example, Marini and Rizzi [17] propose to perform this computation using Brownian motion paths. In figure 1 we show an example of a picture processed with the McCann99 multi-scale retinex [19] (using the reference implementation by Funt et al. [11]), and processed with a Brownian path-based retinex.

Brownian motion is a time-dependent stochastic process B_t modelled after the observation of the movements of particles suspended in a fluid. A process B_t has two properties [7]:

1. with probability 1 (i.e., "almost certainly"), the process is continuous;
2. at each time step, the spatial displacement is random with normal distribution. More precisely, given any two time instants $s \geq t$, the increments $B_s - B_t$ are independent with normal distribution having mean 0 and standard deviation $s-t$.

Although Brownian motion is a continuous-time process, from this second property one can imagine it represented by a path that takes a random step in a random direction at discrete time instants. Figure 2 shows an example with 1000 steps of time increment 1, where in both dimensions the length of each increment is a normally distributed random value with mean 0 and standard deviation 1.

Brownian motion has strong links with the human colour vision. Zeki [23] found that the centroid cells in the region V4 of the visual cortex are distributed in a manner that strongly resembles Brownian motion. This fact was the motivation for Marini and Rizzi [17] to propose an implementation of the retinex image enhancement algorithm based on Brownian motion. The drawback of their approach is the computational complexity for generating the paths: they state that the time for processing a 640×480 image is about ten minutes, which is unacceptable. Although one has to take into account that since the publication of Marini and Rizzi's work the speed of computers has greatly increased, it is not yet feasible to embed such an algorithm in a hand-held device, such as a camera or a mobile phone.



(a) Original (b) McCann99

(c) Random path

Figure 1. *Image from the Kodak dataset [14] processed with a scale-space retinex (McCann99) and with our random path-based approach.*

©2010 Society for Imaging Science and Technology

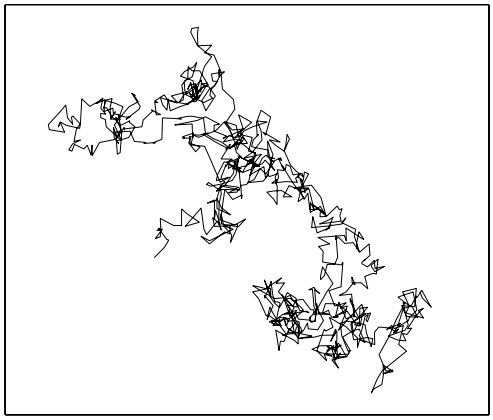


Figure 2. *An example of Brownian motion in two dimensions.*

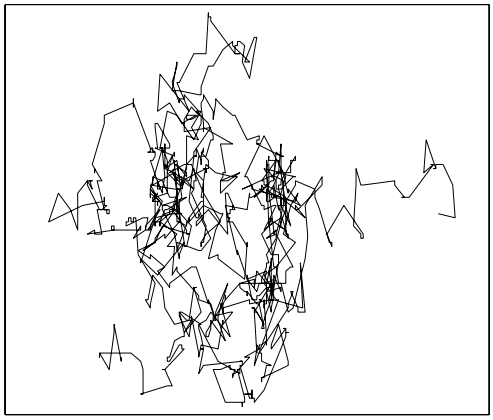


Figure 3. *A thousand steps of a path generated with our algorithm.*

In this paper, starting from the work of Fredembach and Finlayson [9, 10] on random visits to the pixels of an image, we propose an algorithm that generates pseudo-Brownian paths efficiently and guarantees the minimum and the average number of times each pixel is visited (figure 3). These features are very useful for a path-based retinex, but our approach has a drawback. While its time complexity is low ($O(N \log N)$ on an image with N pixels), it requires a large amount of memory to run and it is therefore impractical to perform too many visits for each pixel, as would be desirable. Thus, we devised another way to achieve a large number of visits per pixel. With the scale-space approach in mind, we apply the path-based computation on each scale so that we can limit the number of visits per pixel. Furthermore, experiments show that to obtain the same result as the McCann99 retinex [19], our approach requires on average a smaller number of pixel comparisons per scale.

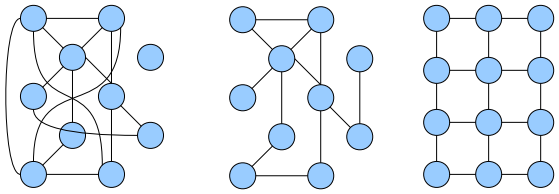


Figure 4. *A generic graph (left), a planar connected graph (centre) and a grid graph (right).*

Hamiltonian paths on the pixels of an image

Visiting all the pixels of an image exactly once is a special case of the problem known as Hamiltonian path, or traveller salesman problem (TSP). While in general TSP is known to belong to the *NP*-complete class [4] (i.e., it is computationally hard to solve), if the input is a grid graph (figure 4) there are algorithms that belong to the class *P*, i.e., they can run in polynomial time [3, 5, 9, 10]. The main idea behind these methods lies in the construction of a minimum spanning tree (MST) or a random spanning tree (RST) over the pixels of a subsampled version of an image. More precisely, a graph is constructed having as vertices blocks of 2×2 pixels and edges connecting each block of pixels to its neighbouring blocks (figure 5(b)). Then, a spanning tree is computed (figure 5(c)) in $O(N \log N)$ operations, for example using Prim's algorithm [4], where N is the number of vertices in the graph. Finally, on the tree a pre-order visit is performed (i.e., starting from the root each vertex is visited before its children), which returns the sequence of visited vertices of the tree. This sequence contains each vertex more than once but, because vertices are composed of four pixels, it is possible to obtain a path that visits each pixel exactly once (figure 5(d)). If the visited spanning tree is random, then this path is a random Hamiltonian path.

Random walk from Prim's algorithm

The above procedure may at first glance appear complicated. If we just generated a random walk we would "almost certainly" visit every pixel [16] and, if the walk is long enough, visit every pixel the same number of times. However, we wish to guarantee we visit every pixel a certain lower bound of times, and a random walk cannot do that. One possibility would be to generate many random Hamiltonian paths. Unfortunately, a random Hamiltonian path does not have a very "random" structure, or at least it is not Brownian. The algorithm we propose here generates a pseudo-random path over the pixels of an image (or, more generally, over the vertices of a grid graph) and is similar in spirit to the TSP algorithms described above. With our method we can set a lower bound k to the number of time the path will visit each pixel; moreover, we can prove that on average the path will visit each pixel $m = 2k$ times.

Since we want to visit each pixel multiple times, instead of a spanning tree as in the above methods, we consider what we may call a "spanning multigraph" (figure 6). A multigraph is a graph where the set of edges is a multiset, that is, the same element can appear multiple times within the set. In other words, a multigraph admits more than one edge, also known as "parallel edges", be-

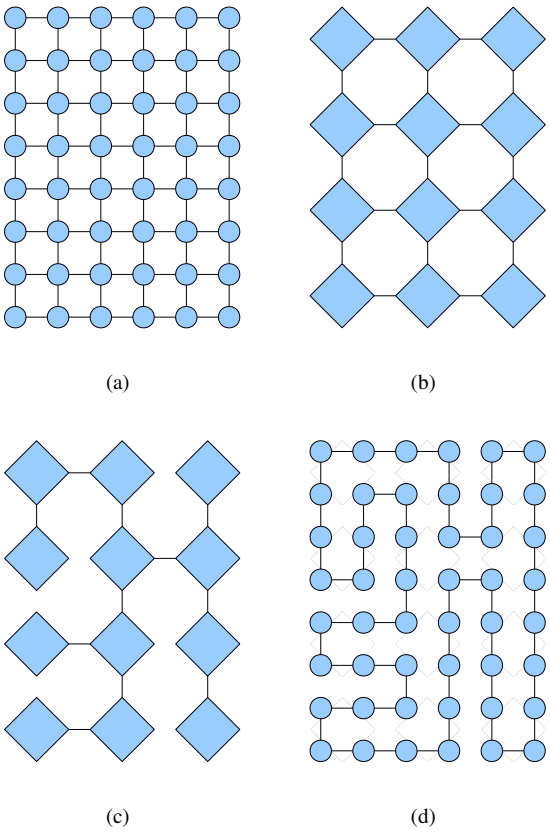


Figure 5. *The original grid graph (a) is subsampled (b). Then, a spanning tree is computed (c), and after upsampling to the original resolution it is possible to adjust the edges to obtain a Hamiltonian path (d).*

tween any pair of vertices [13]. An element that appears z times in a multiset is said to have multiplicity z. Given a connected graph $G = (V,E)$ (i.e., a graph that admits a path between any two of its vertices), our algorithm generates a random multigraph whose visit returns a random path on the original graph. We modified Prim's algorithm for MST [4] so that it considers each edge k times, rather than just one. In short, Prim's algorithm "grows" a tree starting from a vertex of the graph, and adding one edge to the tree at each iteration, so that the new edge will connect the tree to a vertex that does not already belong to it. Thus, once an edge is in the tree, it cannot be considered again because the two connected vertices both belong to the tree. What we do is somewhat similar but we use a different constraint: at each iteration, we add an edge to the multigraph, but we allow to consider a new vertex k times. More in detail, our algorithm works as follows. We start with the multisets $S = \{r\}$ (r is a vertex of G, i.e., $r \in V$), $Q = \emptyset$ and $T = \emptyset$.

1. *Repeat*
2. $Q = Q \cup \{(u,v)\}$, where (u,v) are all the edges so that $u \in S$ and $v \in V \setminus S$ or $v \in S$ with multiplicity less than k (this step requires $\mathrm{O}(\log N)$ operations).
3. Select randomly an edge $(x,y) \in Q$ so that $y \notin S$ or $y \in S$ with multiplicity less than k; then $T = T \cup \{(x,y)\}$ and $S = S \cup \{y\}$. If in Q there is no edge (x,y) satisfying the requirements for y, we discard all the content of Q (this step takes $\mathrm{O}(\log N)$ operations). (In practice, every time we extract an edge from Q, we check whether it is valid or not, and if not, we discard it and extract another one.)
4. *Until* $Q = \emptyset$ (iterate $\mathrm{O}(N)$ times).

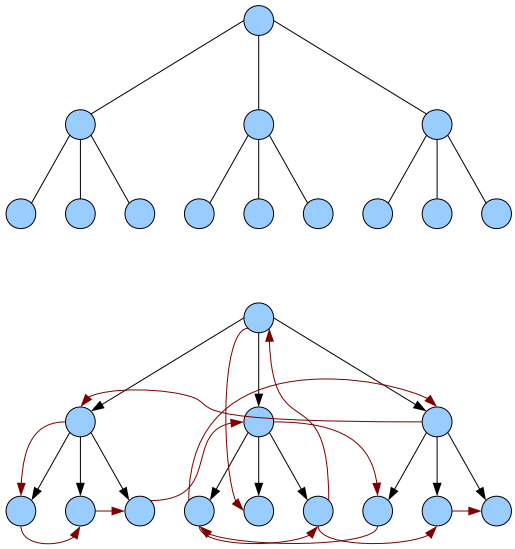


Figure 6. *A ternary tree (top), possibly obtained from a MST algorithm, and a "spanning multigraph" as it might be generated by our algorithm (bottom).*

From the number of operations performed at each of these steps, we can determine the computational complexity to be $\mathrm{O}(N \log N)$. When $k = 1$ this algorithm is equivalent to Prim's algorithm for MST, modified in order to generate RST (in Prim's algorithm for MST, step 3 always selects the edge with the minimum weight). With $k > 1$, the resulting graph $G' = (V,T)$ is a multigraph. Although the original graph is not directed, it is easier to consider each edge (u,v) as oriented in the way it is inserted in the set Q. This leads to consistency with trees, whose edges always go from "parent" to "children", therefore conceptually we can imagine G' as a ternary tree where the nodes are repeated several times. More precisely, each node is repeated exactly k times because of the following:

Lemma 1. *Each vertex in the multigraph G' has exactly k entering edges, apart from the initial vertex r that has exactly $k-1$ entering edges.*

This is what our algorithm does in step 3: it forces the number of edges entering each vertex to be k. From this property we can compute the total number of edges in G': if the initial graph G has N nodes, we have $|T| = (N-1)k + k - 1 = Nk - 1$ edges. A pre-order visit of this tree yields the path with the desired properties and crosses each edge twice, once on its way from root to leave and once on its way back. Therefore, the following holds:

Lemma 2. *The final length of the path in an image of N pixels is $2kN - 1$.*

©2010 Society for Imaging Science and Technology

As a consequence of this property, it is straightforward to see that on average the path visits each pixel $2k$ times when N is large.

From random walks to Brownian motion

The path generated with the above procedure shares features with the random walk on a lattice. For example, we can measure that on average our path takes any of the four directions with probability about 0.25 (although, if an image is rectangular, there is a slight bias towards the longer axis). At every time step, a random walk moves of one pixel in one of the four possible directions (unless it hits a border of the image). However, as we mentioned before, one of the features of the Brownian motion is that the displacement is random. In each dimension, it follows a normal distribution with mean 0 and standard deviation equal to the considered time step [7].

Adapting our random path generator to Brownian motion does not cause much computational overhead and the computational complexity stays the same. Unlike TSP algorithms, we do not require the graph to be a grid – it is enough if it is connected. Therefore we can slightly modify the geometry of the initial graph, allowing normally distributed "jumps" of random length, generated with Marsaglia and Tsang's ziggurat algorithm [18]. In more detail, the initial graph G that we consider in our algorithm above preserves the neighbourhood structure of the image (i.e., it is shaped as a grid graph, as in figure 4). Here, to the grid graph we add some edges, which connect pixels that are not neighbours in the image. Lemma 1 still holds, and so does lemma 2. The visit to the obtained multigraph delivers a path whose displacements are more similar to those of the Brownian motion. We have the control over the statistic of the path: assuming that the time step is 1, we can set it so that its displacements have a normal distribution of mean 0 and standard deviation 1. However, we found that we obtain better results if we force the "jumps" to be slightly longer, i.e., still random and normally distributed but with a larger standard deviation (as for example in figure 3).

Multi-scale Brownian motion retinex

As we mentioned before, our data structure is efficient in terms of time complexity but it requires a large amount of memory. We found that with $k > 100$ our algorithm is impractical to run: one can quickly calculate that with $k = 100$, the length of the path will be 200 times the number of pixels of an image. In order to keep the number of visits for each pixel low, we adopted a multi-scale approach. The algorithm we propose applies a path-based retinex to the pixels of the image at each scale, following the path generated with the above procedure. Retinex performs ratios and products of pixel intensities, but because they are computationally expensive, one can transform them into differences and sums by taking the logarithm of the image. In practice, along a path of pixels $x_1, \ldots, x_n$, we estimate the new value (or "new product") $\mathrm{NP}(x_i)$ of the pixel x_i as

$$\log \mathrm{IP}(x_i) = \log \mathrm{NP}(x_{i-1}) + \log x_i - \log x_{i-1} \tag{1}$$

$$\log \mathrm{NP}(x_i) = \frac{1}{2}\big(\log \mathrm{OP}(x_i) + \min\{\log \mathrm{IP}(x_i), 0\}\big). \tag{2}$$

The first line of this formula represents the ratio and product operation of retinex and delivers the "intermediate product" $\mathrm{IP}(\cdot)$. The second line introduces two operations. First of all, the reset step, represented by the minimum that clips the value of $\mathrm{IP}(\cdot)$ to 1 (or equivalently, its logarithm to 0). Second, the average as described by Frankle and McCann [8]: every pixel of the image is visited multiple times, and every time a new $\mathrm{NP}(x_i)$ is estimated by averaging $\mathrm{IP}(x_i)$ with the "old product" $\mathrm{OP}(x_i)$, i.e., the $\mathrm{NP}(x_i)$ that was computed during the previous visit. Finally, equations (1) and (2) represent a generic iteration of retinex, which requires a first step. Namely, (1) assumes that $\log \mathrm{NP}(x_1) = 0$, that is to say, $\mathrm{NP}(x_1)$ is white. In (2) the value of $\mathrm{OP}(x_i)$ is initialised to the result of the smaller scale: our implementation is multi-scale, therefore it operates on a thumbnail of the image, and the result is then brought up to a larger scale, and so forth until the full-size image is reached. In the smallest version of the image, $\log \mathrm{OP}(\cdot)$ is initialised to 0.

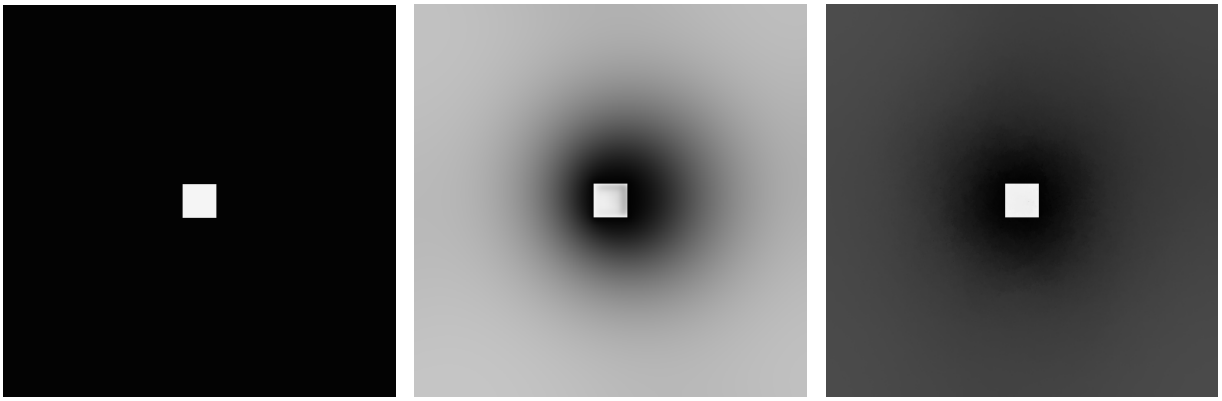
(a) McCann99 (b) Random path

Figure 7. *Rendering of a HDR radiance map (courtesy of Paul Debevec [6]). To obtain this result, the McCann99 algorithm performs 32 comparisons per pixel per scale, our implementation only 16 on average.*

Results

In this section we compare the results of our algorithm with those of the McCann99 retinex [19], using the reference implementation by Funt et al. [11]. In McCann99, each iteration on each scale performs eight pixel comparisons (because it involves a pixel and its eight neighbours) therefore we will count the number of comparisons rather than the number of iterations. We show four examples of images processed first with the McCann99 algorithm, and then with ours. We would like to stress at this point that the images we show are not necessarily of good quality. In fact, they are by choice images that have proven to be problematic for retinex, because our purpose is to draw attention to the differences between the behaviour of the two algorithms.

The first test image is displayed in figure 1, with a photo taken from the Kodak dataset [14]. The McCann99 (32 comparisons per pixel per scale) causes a dark halo to appear around the lighthouse, and smears the narrow frame at the bottom of the picture, while our approach (on average 32 comparisons per pixel per scale) delivers an almost artefact-free result. In figure 7 we show the rendering of a radiance map, from [6]. Both the McCann99 algorithm and our random path retinex introduce halos in the output image. The two images look quite similar, but Mc-



(a) Original (b) McCann99 (c) Random path

Figure 8. *This image, taken from [11], is typically prone to artefacts when processed with retinex algorithms. With a similar number of comparisons per pixel per scale (32 with McCann99, and 32 on average with our approach), our method delivers a much more plausible resulting image.*

(a) Original (b) McCann99 (c) Random path

Figure 9. *Both retinex algorithms converge to the original image. This result is obtained with a multi-scale iteration scheme, that is, the smaller the scale, the more iterations are performed. With 9 scales, the McCann99 algorithm compares each pixel* $256 \cdot 2^s$ *times with its neighbours, with* s *the number of the scale (*$s = 1$ *is the full image). Our algorithm obtains this result with* $16 \cdot 2^s$ *comparisons only, again with 9 scales.*

Cann99 requires 32 comparisons per pixel per scale, while our implementation only 16 on average. Figure 8, taken from [11], shows a dark halo around the white square in both images processed with retinex. However, the McCann99 halo has a clear bias towards the top-right of the image, because of the order the pixels are compared. Moreover the halo is also "reflected" inside the white square. Our path-based approach produces a halo as well, but its shape is less well defined than in the McCann99 output. Furthermore, there is no halo inside the white square and the background is closer to the original black colour. Finally, figure 9 shows an example of convergence as intended by Brainard and Wandell [2]: our algorithm requires fewer iterations (one sixteenth) of those required by McCann99 to converge to the original image.

Algorithm speed comparison for the image in figure 7.

Algorithm	Complexity	Time for 64 visits
McCann99	$O(N)$	12.6 s
Proposed approach	$O(N \log N)$	59.9 s
Marini and Rizzi	$O(k\chi N)$	n.a.

In terms of speed, with the same number of comparisons per pixel McCann99 has the advantage. This algorithm has linear complexity on the number of pixels and is therefore very efficient, as we show in the table above for a 512×768 image. On the other hand our random path approach requires fewer comparisons per pixel to obtain results similar to those of the McCann99 algorithm, and this can give some advantage to our method. We would like to stress here that our implementation (in Matlab and C) is still a prototype and with proper tuning it will be possible to achieve significant speed-ups. As for Marini and Rizzi's retinex [17], we could not obtain a direct comparison because the authors did not specify precisely how they generated their paths. However, the complexity of their method depends mainly on the parameter χ, which tunes the length of the paths, and k, the number of paths. The authors write that k tends to remain small, but do not give any detail about χ.

Conclusions and future work

In this paper we presented an efficient retinex algorithm that combines the advantages of Brownian random paths to those of the multi-scale approach. In the comparison with those of the McCann99 algorithm [11, 19], our results show less evident artefacts with the same number of ratios per pixel (figures 1 and 8), and similar output images with a smaller number of ratios per pixels (figures 7 and 9). From this, we can conclude that the convergence of our algorithm, as intended by Brainard and Wandell [2],

©2010 Society for Imaging Science and Technology

is faster. Furthermore, our main contribution lies in the efficient random path generator, an algorithm that requires only $O(N \log N)$ operations and generates a pseudo-Brownian motion that visits each pixel at least a fixed number of times.

In the future, we are planning to draw a comparison between the paths generated with our algorithm, random walks and the Brownian motion, to gain a better understanding of the links between our pseudo-random path generator and these stochastic processes.

Acknowledgments

This work is supported by the UK EPSRC grant number EP/E012248/1.

References

[1] Kobus Barnard and Brian V. Funt. Analysis and improvement of multi-scale retinex. In *Proceedings of Fifth Color Imaging Conference*, pages 221–226. IS&T and SID, November 1997.

[2] David H. Brainard and Brian A. Wandell. Analysis of the retinex theory of color vision. *J. Opt. Soc. Am.*, 3(10):1651–1661, 1986.

[3] Shao Dong Chen, Hong Shen, and Rodney Topor. An efficient algorithm for constructing Hamiltonian paths in meshes. *Parallel Comput.*, 28(9):1293–1305, 2002.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to algorithms*. The MIT Press, second edition, 2001.

[5] Revital Dafner, Daniel Cohen-Or, and Yossi Matias. Context-based space filling curves. *Comput. Graph. Forum*, 19(3):209–218, 2000.

[6] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 369–378. ACM Press/Addison-Wesley Publishing Co., 1997.

[7] Richard Durret. *Brownian motion and martingales in analysis*. Wadsworth Advanced Books & Software, 1984.

[8] Jonathan A. Frankle and John J. McCann. Method and apparatus for lightness imaging. U.S. Patent No. 4,384,336, 1983.

[9] Clément Fredembach and Graham D. Finlayson. Hamiltonian path-based shadow removal. In *Proceedings of 16th British Machine Vision Conference*, volume 2, pages 502–511. BMVA, 2005.

[10] Clément Fredembach and Graham D. Finlayson. The 1.5D sieve algorithm. *Pattern Recogn. Lett.*, 29(5):629–636, 2008.

[11] Brian V. Funt, Florian Ciurea, and John J. McCann. Retinex in MATLAB^{TM}. *J. Electron. Imaging*, 13(1):48–57, 2004.

[12] Davide Gadia, Alessandro Rizzi, and Daniele Marini. Tuning Retinex for HDR images visualization. In *Proceedings of Second European Conference on Color in Graphics, Imaging and Vision*. IS&T, 2004.

[13] Ronald Gould. *Graph theory*. The Benjamin/Cummins Publishing Company Inc., 1988.

[14] Kodak. Kodak Lossless True Color Image Suite. Webpage, 2004. URL `http://r0k.us/graphics/kodak/`.

[15] Edwin H. Land and John J. McCann. Lightness and retinex theory. *J. Opt. Soc. Am.*, 61(1):1–11, 1971.

[16] Gregory F. Lawler and Lester N. Coyle. *Lectures on Contemporary Probability*. Student Mathematical Library, 2000.

[17] Daniele Marini and Alessandro Rizzi. A computational approach to color adaptation effects. *Image Vision Comput.*, 18(13):1005–1014, 2000.

[18] George Marsaglia and Wai Wan Tsang. The Ziggurat Method for Generating Random Variables. *J. Stat. Softw.*, 5(8):1–7, 2000.

[19] John J. McCann. Lessons learned from Mondrians applied to real images and color gamuts. In *Proceedings of Senventh Color Imaging Conference*, pages 1–8. IS&T and SID, 1999.

[20] Laurence Meylan and Sabine Süsstrunk. High dynamic range image rendering with a retinex-based adaptive filter. *IEEE T. Image Process.*, 15(6):2820–2830, 2006.

[21] Zia-ur Rahman, Daniel J. Jobson, and Glenn A. Woodell. Retinex processing for automatic image enhancement. *J. Electron. Imaging*, 13(1):100–110, 2004.

[22] Robert Sobol. Improving the Retinex algorithm for rendering wide dynamic range photographs. *J. Electron. Imaging*, 13(1):65–74, 2004.

[23] Semir Zeki. *A vision of the brain*. Blackwell Science Ltd, 1993.

Author Biography

Roberto Montagna received his BSc in computer science in 2004 and his MSc in intelligent and multimedia systems in 2007 from the University of Verona (Italy). Since 2007 he has been a PhD student at the University of East Anglia (Norwich, UK) under the supervision of Prof. Graham D. Finlayson.

Graham D. Finlayson obtained his BSc in Computer Science from the University of Strathclyde (Glasgow, Scotland) in 1989. He then pursued his graduate education at Simon Fraser University (Vancouver, Canada) where he was awarded his MSc and PhD degrees in 1992 and 1995 respectively. From August 1995 until September 1997, Dr. Finlayson was a Lecturer in Computer Science at the University of York (York, UK) and from October 1997 until August 1999 he was a Reader in Colour Imaging at the Colour & Imaging institute, University of Derby (Derby, UK). In September 1999 he was appointed a Professor in the School of Computing Sciences, University of East Anglia (Norwich, UK).