Path-based computations in colour image processing

Roberto Montagna

A thesis submitted for the Degree of
Doctor of Philosophy

University of East Anglia
School of Computing Sciences

March 2011

Abstract

Path-based computations have been brought into image processing with the purpose of reducing the two-dimensional structure of images into one dimension. There may be different reasons to do that, for example often one-dimensional algorithms are much simpler and more efficient than their two-dimensional counterparts. Also, in some cases, two-dimensional algorithms might not exist at all.

In this thesis we examine the role of various curves in different aspects of image processing. In colour-based image retrieval, we show that Lissajous curves generate a set of optimal interpolation points that we use for the approximation of chromaticity histograms. Extensive tests show that the method we propose matches and in some cases improves the retrieval results of existing techniques based on the DCT and PCA.

We then take into account two aspects of image enhancement: the appearance of colour images and the generalised conversion from colour to greyscale. A classic algorithm for image enhancement is retinex. Originally devised as a model for human colour vision 40 years ago, it still plays a prominent role, finding applications for example in the tone mapping of high dynamic range images. Retinex was initially proposed as a path-based algorithm, and later it was shown to work very well with Brownian paths. However, the implementation was computationally prohibitive because it had to generate several paths per pixel. In this context, we propose an efficient pseudo-Brownian path generator with very useful constraints: it guarantees the minimum and the average number of times each pixel is visited, which is very important for image enhance-

ment. We develop our approach generalising the concept of minimum spanning tree of a graph.

In the colour to greyscale problem, we consider the colour gradient yielded by the structure tensor. This formulation is valid regardless of the number of channels in the input image, therefore it can be used for multispectral images as well. In previous work, it has been shown that the reintegration of this gradient produces a greyscale image that contains contrast information from all the channels. However, this proved to be a hard problem because the colour tensor gradient is, in general, non-integrable. In this thesis we provide a simple way of reducing the integrability error, and we explore the possibility of a retinex-based reintegration of the structure tensor gradient by means of our path-based and other retinex implementations. The resulting greyscale images show details that are lost with other, more traditional transformations. We also perform a preference test with human observers, who tend to prefer our method over other colour to greyscale transformations.

Acknowledgements

Firstly, my thanks go to my supervisor Prof. Graham Finlayson. I still do not know exactly why he chose me out of the various candidates he had for this Ph.D. position. Yet he did, so I can safely say that without him I would not have written this thesis. Of course that is not all – I must thank him for his constant support, encouragement and advice, all of which have left a distinct footprint on the finished product.

Another person whom I want to thank for giving me guidance is Dave Connah. I received a lot of help from him throughout my Ph.D. in the form of tea breaks, discussions and the proofreading of countless papers (well, not countless, but it must have been terribly boring anyway).

Thanks also to Barry Theobald, for the not infrequent tips on dodging UEA bureaucracy, or at least on squeezing through its mesh. He also went through the painstaking job of reading this thesis from top to bottom, helping me to make it what it is now.

In the academic world I also would like to thank my Ph.D. examiners, Prof. Arnold Smeulders and Dr. Mark Fisher, for their final touch that gave this thesis its current shape.

Next in the long list of people I need to thank are my parents. They brought me up, taught me the importance of studying, they encouraged my interest in computers despite not understanding it, and last but not least they supported me when I decided to do a Ph.D. abroad. Also, when I was growing up they sent me to study English abroad, which in hindsight turned out to be a winning decision.

I was warned how hard it is to write the acknowledgements. If they are exhaustive they are too long and you risk forgetting someone. If you keep them short, you leave out a lot of people. Pitching it right is by no means an easy task so I hope I?ll be forgiven if I forget someone. I have to thank my friends from the Colour Research Lab for making days at work more entertaining: Aurora, Dome, Javier, Michał, Mike, Stuart; Mary and Rowena who are not strictly speaking part of the lab but I feel I should count them in; my friends Daniela and Carlo for sharing most of my non-working life; my girlfriend Charlotte for bearing with me while I was writing up (and for bearing with me in general); and my old friend Kristina for sending me all the journal papers that were not available from UEA.

Last but not least, I have to thank two groups of friends for which the list of names would be overly long. My friends in Italy, whom I see only when I return on holiday, and my friends from salsa, whom I see when I go out and account for a great part of my social life.

Contents

List of Figures

List of Tables

Publications

The following are publications by the author related to this work:

- R. Montagna and G. D. Finlayson. Histogram compression and image retrieval through Padua points interpolation. In *Proceedings of IS&T's Fourth European Conference on Color in Graphics, Imaging and Vision*, pages 362–366, Terrassa, Spain, June 2008.

- G. D. Finlayson and R. Montagna. Optimal interpolation and ℓ^p norm minimisation in colour indexing. In *Proceedings of IS&T and SID's Sixteenth Color Imaging Conference*, pages 274–278, Portland, Oregon (USA), November 2008.

- R. Montagna and G. D. Finlayson. Reducing integrability artefacts for data fusion through colour space manipulation. In *Proceedings of IEEE Color and Reflectance in Imaging and Computer Vision Workshop*, pages 1955–1961, Kyoto, Japan, October 2009.

- G. D. Finlayson, R. Montagna and D. Connah. A unified approach to colour2grey and image enhancement through gradient field integration. In *Proceedings of IS&T and SID's Seventeenth Color Imaging Conference*, Albuquerque, New Mexico (USA), November 2009.

- R. Montagna and G. D. Finlayson. Constrained pseudo-Brownian motion for image enhancement. In *Proceedings of IS&T and SID's Eighteenth Color and Imaging Conference*, San Antonio, Texas (USA), November 2010.

- R. Montagna and G. D. Finlayson. Padua point interpolation and ℓ^p-norm minimisation in colour-based image indexing and retrieval. Accepted for publication by *IET Image Processing*, 2011.

- R. Montagna and G. D. Finlayson. Constrained pseudo-Brownian motion and its application to image enhancement. Accepted for publication by *Journal of the Optical Society of America*, 2010.

Glossary

DCT	discrete cosine tranform
DFT	discrete Fourier tranform
HI	histogram intersection
HSV	hue, saturation, value
HT	hybrid transform
NIR	near-infrared
PCA	principal component analysis
PDE	partial differential equation
RGB	red, green, blue
SVD	singular value decomposition
SW	Socolinsky and Wolff's method
SWD	Socolinsky and Wolff's method on desaturated images
SWL	Socolinsky and Wolff's method on log-images

Chapter 1

Introduction

Digital cameras have radically changed people's approach to photography but this is not necessarily related to an improvement in quality. In fact, there is no comprehensive study on whether film or digital photography provides better image quality, and probably there is no need for one. Nevertheless, Fairchild (2010) proposed a case-study in which two film cameras and two digital cameras are compared. This work is far from being conclusive, but the author stresses that the real strength of digital systems is their flexibility. Memory cards can easily store hundreds or even thousands of photos and are small enough to fit a few dozen in your pocket. Professional and amateur photographers can work relatively easily in remote areas of the world, their main limitation being the life of the batteries. This is a far cry from Frank Hurley's efforts in 1917, when amazingly he managed to save 120 glass plates from the disaster of Shackleton's expedition to Antarctica (State Library of New South Wales, 2008, 2009) – from the point of view of numbers alone, nowadays this would not be nearly as impressive.

Certainly such flexibility is handy, so it is natural to ask if this comes at a cost. At the moment of writing, a quick internet search provides the following prices: a point-and-shoot compact digital camera (of reasonable quality) with a 12.1-megapixel sensor costs £82.20, and a 8-gigabyte memory card costs £11.25. So, we can safely say that for less

than £100 you can buy all the equipment you need for consumer digital photography. We should not forget to mention phone cameras. Virtually all mobile phones on the market today come with an embedded digital camera. In most cases the image quality is nowhere near that of a £100 camera, however it is good enough for most people's needs.

Flexibility in this type of photography also comes from the ease in which digital images can be manipulated, and this does not require a digital camera. A photo can be taken on a film and then digitised using a scanner. However, the availability of reasonably priced digital cameras has certainly helped to spread digital manipulation techniques to a wider public, and "photoshopping" is certainly simpler and cheaper than film editing. This offers a lot of possibilities for correcting poor quality photos, however in some cases it is desirable to automate certain tasks. This is certainly true for computer vision, but not necessarily only in that field. Sometimes, a user would want a photo to "look good" without any further effort or specification. The problems we are trying to address belong to this category, and in this chapter we will provide a brief introduction to them.

1.1 Areas of research addressed in this thesis

In the central part of this thesis, we will take into account three different problems and will show how we approach them in a path-based fashion. First of all, the problem of colour indexing and retrieval will be considered. Clearly, one of the big advantages of digital cameras is the number of photos that can be taken – currently in the order of hundreds for a cheap memory card. Within a few months of regular use, it is easy to have accumulated thousands of digital photographs, making it hard to locate a specific one, unless they have been painstakingly tagged and described from the start. It is therefore desirable for users to have access to a system that simplifies this task. Such a system

could, for example, retrieve all photos similar to a given sample image and, in chapter 3, we will consider one aspect of this problem – colour. While colour in itself is not enough to retrieve images accurately, it is a very useful cue since fast algorithms exist that can narrow down the search for more elaborate and slower algorithms, which also consider other cues. In this context, colour is often represented as a colour or chromaticity histogram (Swain and Ballard, 1991). These are normally stored along with images so that when a new image comes, its histogram can be compared against those in the database. Further work has then been carried out to reduce the size of these histograms. In this thesis we propose to sample two-dimensional histograms along a Lissajous path with a technique called Padua point interpolation. We propose this technique as it has certain optimality properties that we thought useful for image indexing. We will show that this method competes with state-of-the-art statistical methods based on the DCT and PCA.

The second problem we will consider with a path-based approach is image enhancement, namely retinex algorithms. Based on a model of human colour vision developed 40 years ago, retinex algorithms have found many applications in image processing, particularly in restoration (for example, removing haze) and enhancement (for example, showing details that are underexposed or overexposed in the input image). Retinex was originally proposed as an operator based on piecewise linear paths, but here we will focus on a later implementation that uses Brownian motion (Marini and Rizzi, 2000). It was shown that this type of path was particularly useful with retinex, however the implementation was computationally expensive. The reason for this is that random paths cannot be trusted to visit all the pixels of an image, so the authors proposed to generate several paths per pixel. In chapter 4 we overcome this problem with an efficient algorithm that generates pseudo-random walks and pseudo-Brownian paths, with two important constraints: one is on the minimum number of times every pixel is visited, the other on the average number of times every pixel is visited. We employ this algo-

rithm in a multi-scale retinex implementation, which proves to be more efficient than the previous Brownian motion retinex and, at the same time, provides better results than traditional multi-scale approaches.

Finally, in chapter 5 we consider the third problem, image fusion, and we will show that retinex can play a prominent role in this field as well. Here, the purpose of image fusion is to represent a multichannel image (for example, a three-channel colour image or a multispectral image obtained with remote sensing techniques) with a single summary greyscale. The general approach to this problem is to perform some weighted average of the channels, in which the weights can be fixed or somehow determined from the content of the image itself. In the case of colour images, another typical solution is to convert the image into some perceptual colour space, for example CIE-Lab (e.g., Hunt, 1998), and then take the luminance channel as the solution. Both approaches have the drawback of not representing details that only depend on chromatic contrast or multichannel contrast. Multichannel contrast does not necessarily mean chromatic since it can come from non-visible light (from now on, when we use the word "chromatic" in this chapter we will implicitly mean "multichannel" as well, unless stated otherwise). In other words, since chromatic information is lost in the transformation, the relative detail is not present in the output image. A possible solution to this problem is to measure somehow the chromatic differences and remap them as grey-level differences in the output image. In chapter 5 we consider the colour tensor gradient (Di Zenzo, 1986), a formulation for determining a single gradient field that contains contrast information from all the channels. From this gradient, it is possible to reconstruct a greyscale image by means of reintegration. This is obtained by setting up a partial differential equation (PDE). Unfortunately, solving this equation is not a straightforward step because the gradient field is, in general, non-integrable. It is possible to find least-squares approximate solutions with different methods, however this can result in very obvious visual artefacts despite the mathematical soundness of the approach. Our first contribution

here is that we devised a simple and straightforward way to reduce the integrability error of the gradient. A simple desaturation of the colour content, suitably redefined in the case of multichannel images, yields a more integrable colour tensor gradient. Our second contribution lies on the reintegration step. The relation between retinex and PDEs has been known for years, thus PDE techniques have been used to solve retinex. Our approach is to use retinex for the solution of the PDE arising from the data fusion problem. The main advantage of using retinex is its robust perceptual foundations, which are reflected in the enhancement of the output images. In chapter 6 we describe our psychophysical experiment in which observers had to evaluate the appearance of images obtained with our method in comparison with those obtained with existing techniques.

1.2 Why path-based?

In chapter 2 we will show that in the image processing literature several examples of path-based algorithms exist. In the different problems we are trying to address we are also taking this approach, so its advantages still need to be clarified. Images, by their nature, are defined on a two-dimensional domain, meaning algorithms that operate on them need to take this into account. The added complexity of a two-dimensional algorithm can vary. In some cases, this is limited and everything works seamlessly – for example, the fast Fourier transform can be simply performed separately in each dimension (e.g., Frigo and Johnson, 2005). In other cases, however, the added complexity is substantial and two-dimensional algorithms are slower and more complicated than their one-dimensional version, as in the case of the sieve algorithm (e.g., Southam, 2006). Moreover, in some cases two-dimensional algorithms may not even exist. For example, in image compression, entropy encoders expect a stream of bytes as input, and need somehow to transform the two-dimensional structure of the original data into a one-dimensional structure. This is exactly the main advantage and purpose of path-based

algorithms: they can view the structure of an image as one-dimensional. However, it is far from trivial to see what type of path should be used to traverse all the pixels. Some paths are very easy to compute but are less beneficial than other, more complex ones. In chapter 2 we will analyse different kinds of paths and see that their different properties make each of them useful for different problems.

1.3 Outline of the thesis

In chapter 2 we review various types of paths and their properties, show some of their applications and also introduce the retinex theory of colour vision. In chapter 3 we present our method for colour-based image indexing and retrieval, based on Lissajous curves, then in chapter 4 we propose our pseudo-random path generator and apply it to retinex. In chapter 5 we propose our data fusion method and in chapter 6 we evaluate this method by means of a psychophysical experiment. Finally, we conclude in chapter 7.

Chapter 2

Background

In this thesis we are proposing the use of path-based algorithms for various applications in image processing. This preliminary chapter reviews different types of paths that can be employed in this field and present some of their applications. Moreover, we will summarise some features of the retinex theory, since it has an important role in our research.

2.1 Space-filling curves

The concept of a space-filling curve was devised to prove the existence of mappings from the unit interval to the unit square (Butz, 1968). The first formulation was given by Peano (1890), although the most famous and perhaps most used was introduced one year later by Hilbert (1891). While Peano produced an explicit formulation of his curve, Hilbert's construction was mostly geometrical and substantially the same as that shown in figure 2.1.

Peano and Hilbert curves are fractals (Mandelbrot, 1983). The recursive formulation that we see in figure 2.1 can be repeated indefinitely, meaning that if we "zoom to" a Hilbert curve in the real unit square we keep seeing a Hilbert curve. In image

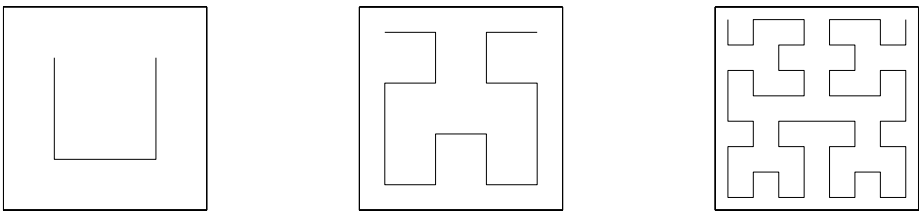


Figure 2.1: First three iterations of a Hilbert curve.

processing, space-filling curves have the property of prioritising neighbouring pixels in the visit, before proceeding with visiting pixels that are far away, thus exploiting the two-dimensional locality. This is a great advantage over raster paths, which sweep an image only horizontally or vertically, depending on the implementation, and thus exploit locality only in that direction.

2.2 Random paths

Random paths are instances of more general stochastic processes that, for the purposes of image processing, visit the pixels of an image in a random order. We consider two types of random paths: Brownian motion and the random walk. Under some simplifying assumptions, the two constructions are equivalent. Due to its simpler formulation, we will start with the random walk.

2.2.1 Random walk

The simplest form of random walk is on a straight line. In this case, at each step the "walker" moves to the right with a certain probability p and to the left with a probability $q = 1 - p$. Here we will assume that each step has unitary length and that $p = q = 0.5$; thus, we can imagine the walker throwing a coin at each step and moving to the right if the result is a head or to the left if it is a tail. The first property of the random walk

is the expected distance from its initial position after n steps. We define Δ_j to be the displacement at step j (i.e., $\Delta_j = \pm 1$) and $S_n = \sum_{j=1}^{n} \Delta_j$ the distance from the origin after n steps. Its expected value is $\mathbf{E}[S_n] = 0$, because $\mathbf{E}[\Delta_j] = 0$. It is not easy to find $\mathbf{E}[|S_n|]$, but we can instead obtain $\mathbf{E}[S_n^2]$. We report here the brief calculation provided by Lawler and Coyle (2000).

$$\begin{aligned} \mathbf{E}[S_n^2] &= \mathbf{E}\left[\left(\sum_{j=1}^{n} \Delta_j\right)^2\right] \\ &= \sum_{j=1}^{n} \mathbf{E}[\Delta_j^2] + \sum_{j \neq k} \mathbf{E}[\Delta_j \Delta_k] \end{aligned} \tag{2.1}$$

Since each step is independent from the previous $\mathbf{E}[\Delta_j \Delta_k] = \mathbf{E}[\Delta_j]\mathbf{E}[\Delta_k]$. We also have $\Delta_j^2 = 1$ and $\mathbf{E}[\Delta_j] = 0$, thus

$$\mathbf{E}[S_n^2] = n \tag{2.2}$$

assuming steps of unitary length. From this we can conclude, at least informally, that the expected distance from the origin after n steps is proportional to $\sqrt{n}$.

The second property of the random walk that will be useful for the purpose of this thesis is a formula for the probability of the walker being at distance d from its initial position after n steps. For simplicity, let us focus on the case where the walker ends d steps to the right, following the formulation by Rudnick and Gaspari (2004). In the coin-tossing example, this is equivalent to asking what is the probability of obtaining h heads and $n - h$ tails in any order, where

$$d = h - (n - h) = 2h - n. \tag{2.3}$$

Since n and d are known, we can write h as

$$h = \frac{1}{2}(d + n). \tag{2.4}$$

The probability of having n throws with h heads is $\binom{n}{h}$ and the probability of any n-step walk is $(1/2)^n$. Therefore we have

$$P(\#\text{heads} = h) = \frac{1}{2^n}\binom{n}{h}. \tag{2.5}$$

This allows us to formulate the probability $P(n, d)$ of the walker being at a distance d after n steps (that is, we are considering a generic distance d, not d steps to the right)

$$P(n, d) = 2P(\#\text{heads} = h) = \frac{1}{2^{n-1}}\binom{n}{\frac{d+n}{2}}. \tag{2.6}$$

Because we are dealing with images, we are interested in the random walk on a lattice (which corresponds to the integer domain $\mathbb{Z} \times \mathbb{Z}$), where the walker can move to the right, to the left, up and down. We will restrict our discussion to the case where each direction has the same probability $p = 0.25$ and we can show that the property of equation (2.2) still holds (in fact, it holds for the random walk in any number of dimensions, see Lawler and Coyle, 2000). To prove that, we have to consider the Manhattan distance, or city-block distance, i.e., given two points $A = (a_x, a_y) \in \mathbb{Z}^2$ and $B = (b_x, b_y) \in \mathbb{Z}^2$,

$$D(A, B) = |a_x - b_x| + |a_y - b_y|. \tag{2.7}$$

In the two-dimensional case, each displacement has still expected value $\mathbf{E}[\Delta_j] = 0$, and because at each time step the walker moves only in one direction, $\Delta_j^2 = 1$. Time independence still holds as well, so equations (2.1) and (2.2) are valid also in this case. The problem with the random walk on the lattice is that, to the best of our knowledge, in the literature there is no formulation such as that of equation (2.6) to compute the

probability $P(n, d)$ of the walker being at distance d after n steps. In chapter 4 we present the way we devised to calculate this measure.

2.2.2 Brownian motion

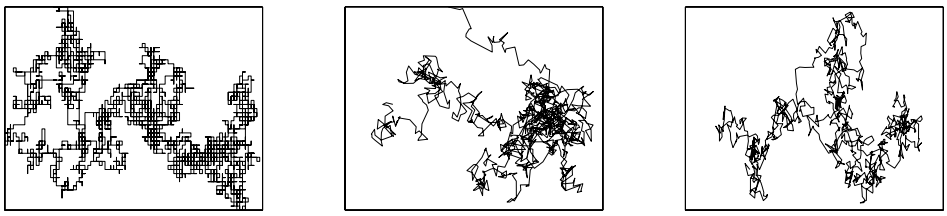
Brownian motion is the apparently random movement of particles suspended in a fluid. Initially only described through empirical observation, it found several applications after its mathematical formalisation as a Wiener process, for example the solution of PDEs (Durret, 1984) and the modelling of diffusion processes (Freedman, 1971). Intuitively, Brownian motion on a plane can be seen as a particle that moves in a random direction with random displacement at each time step, although this comparison is not precise as Brownian motion is a continuous process. A k-dimensional Brownian motion is a process B_t, $t \in [0, +\infty) \subset \mathbb{R}$, having values in $\mathbb{R}^k$, having the following properties:

1. if $t_0 < t_1 < \ldots < t_n$ then $B(t_0), B(t_1) - B(t_0), \ldots, B(t_n) - B(t_{n-1})$ are independent.

2. If $s, t \geq 0$, the increment $B(t+s) - B(t)$ has normal distribution with zero mean and variance s.

3. B_t is almost certainly (i.e., with probability 1) continuous.

In other words, this means that if we sample this process at discrete instants, for each dimension the increments are independent and normally distributed. The fact that it is continuous means that we can sample it as densely as we like, and its increments will still be random. Put another way, Brownian motion is a fractal (e.g., Falconer, 2003), thus we could "zoom in" indefinitely and still recognise a Brownian motion. In figure 2.2 we show the path followed by a particle whose trajectory is determined by Brownian motion.



(a) *(b)* *(c)*

Figure 2.2: Two simulations of Brownian motion in two dimensions compared with a random walk. Brownian motion, starting from a given origin, at each time step changes its position of a random value in the x and in the y direction, thus leaving the trail we see in (b) and (c). The random walk (a) instead moves on a grid, thus is changes its position always of a unitary value ($+1$ or -1), picking a random direction (x or y) at each time step.

2.3 Hamiltonian paths

The Hamiltonian path of a graph is a cycle that visits every vertex of the graph exactly once. It is similar to the travelling salesman problem, which takes a weighted graph as input and requires that the Hamiltonian cycle has minimum cost. Both problems, in their general formulation, are known to be NP-complete. This means in practice that their computational complexity is exponential on the size of the input graph (this is not the exact definition of NP-completeness; further details can be found in many textbooks on algorithms, e.g., Cormen et al., 2001). However, if the input graph is a planar grid graph, at least for the Hamiltonian path problem, there are algorithms that run in polynomial time (Chen et al., 2002; Dafner et al., 2000; Fredembach and Finlayson, 2005, 2008). An example of a planar grid graph is shown in figure 2.3a. It is quite straightforward to see how an image can be seen as a planar grid graph: nodes correspond to pixels, and edges correspond to a neighbouring relation between pixels. 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, as we show in figure 2.3. 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 2.3b). Then, a spanning tree is computed (figure 2.3c), for example using Prim's algorithm (Cormen et al., 2001), which requires $\mathrm{O}(N \log N)$ operations on a graph having N vertices (although there exist algorithms that run in expected linear time, see Karger et al., 1995). 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 2.3d). If the visited spanning tree is random, then this path is a random Hamiltonian path. The complexity of this algorithm is potentially $\mathrm{O}(N)$ expected time. The downsampling and upsampling operations are linear, and so is the visit to the spanning tree. The bottleneck here is the MST but, as we said, it is possible to find one in expected linear time.

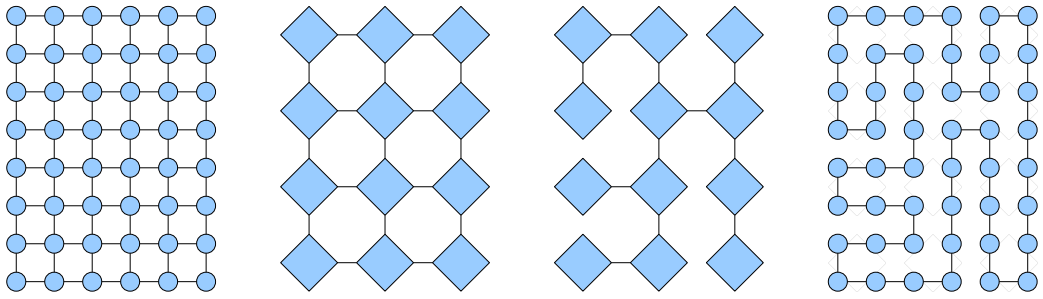


Figure 2.3: Hamiltonian path on the pixels of an image represented as a graph. The original grid graph (a) is subsampled (b). Then, a spanning tree is computed (c), and after upsampling to the original resolution it is possible to adjust the edges in order to obtain a Hamiltonian path (d).

2.4 Applications of path-based computations

2.4.1 Image compression

Perhaps the first operation ever to benefit from a path-based approach has been image compression. Entropy encoders, such as the LZW algorithm (Welch, 1984; Ziv and Lempel, 1978), expect a stream of data as input, which means that the two-dimensional structure of an image has to be lost and the pixels treated in sequence. Initially, this was not considered a problem. For example, the GIF file format uses a scan-line that traverses the image from left to right and from top to bottom (CompuServe Inc., 1990). However, this method does not take into account that neighbouring pixels tend to be more similar than pixels far away from each other. Ignoring this fact reduces the efficiency of the compression of entropy encoders, therefore it was proposed to change the order of the visit to increase the compression ratio. Ansari and Fineberg (1992) proposed to use Peano-Hilbert curves, such as that in figure 2.1, to traverse the pixels in a way that privileges locality, and showed that this is an improvement over the scan-line method. A further improvement has been introduced by Dafner et al. (2000). In their work, they show that finding a Hamiltonian path over the pixels of an image, using the method described in section 2.3, could potentially improve the compression ratio even further. They use information from the gradient of the image to determine the weights of the subsampled version of the graph (i.e., that in figure 2.3b). The minimum spanning tree algorithm will select only the edges of minimum weight, therefore in the final Hamiltonian path the difference between subsequent pixels will be minimal, easing the work of the entropy encoder. Unfortunately, this technique has some overhead. While Peano-Hilbert curves have a predefined formulation, the Hamiltonian path has to be somehow stored together with the compressed image in order to reconstruct the position of the pixels. Dafner et al. show how to encode the curve using only 0.75 bits per pixel prior to compression, but this overhead is enough to make it less effective than

Peano-Hilbert curves.

2.4.2 Sieve operator

Hamiltonian paths find application in different aspects of image processing. For example, they have been used in the implementation of a sieve filter. The sieve is a scale-space hierarchical operator that "simplifies" an image by removing extrema and merging neighbouring pixels into flat regions (Bangham et al., 1996a,b; Southam, 2006). There are one and two-dimensional sieve algorithms and while the former are more efficient, the latter are more useful in image processing. Fredembach and Finlayson (2008) introduced another algorithm based on random Hamiltonian paths. In short, the sieving is performed along m of such paths. When $m = 1$ this algorithm is equivalent to the one dimensional sieve. However, the authors prove that if one generates all the possible different paths over the image, their algorithm is equivalent to the two-dimensional sieve. This theoretical result is supported by experimental data, which shows that with a reasonable number of random paths (the authors indicate $m = 24$ as good choice for their experiments) the behaviour of the algorithm is close to that of the two-dimensional sieve (Fredembach, 2006; Fredembach and Finlayson, 2008).

2.4.3 Shadow removal

Shadow removal has also seen a successful application of Hamiltonian paths. Put simply, the principle behind shadow removal lies in the observation that a shadow edge is due only to a change in the illumination, under the assumption that the shadow is projected onto a uniform surface. If the position of the shadow edge is known, one can remove this edge from the gradient (in practice simply setting the corresponding gradient to zero), whose reintegration should recover a shadow-free image (see Finlayson et al., 2002, 2006). This, however, is problematic when the shadow edge coincides with

a material edge because one cannot simply set the gradient to zero. If one does so, the resulting image will present visible artefacts due to reintegration error. To overcome this problem, Fredembach and Finlayson (2005) proposed a path-based reintegration. If the shadow edge is completely removed from the image (i.e., not simply set to zero, but ignored by means of a mask), one can think of a path that enters and exits the shadow area only in one location. The probability of that location being on a material edge is low, and if several random Hamiltonian paths are considered, each connecting the shadow area to the rest of the image through a different location, the average of the reintegration over the different paths should yield an artefact-free image. Once the shadow-free image is obtained, the edge is recovered with an inpainting algorithm (see Fredembach, 2006; Fredembach and Finlayson, 2005 and references within). The path-based reintegration has two major advantages in this application. Two dimensional reintegration algorithms are more computationally expensive and require boundary conditions to be set. The path-based algorithm instead has linear complexity (although there is some multiplicative factor that depends on the number of paths to be used) and no boundary condition problems, since the one dimensional integral is well defined up to a constant that in this case can be easily recovered.

2.4.4 Image enhancement

Paths have played a quite important part in image enhancing, especially in relation to the retinex algorithm (an explanation of the principles behind this algorithm will be provided in section 2.5). Several different flavours of retinex have been developed: multi-scale (Frankle and McCann, 1983; McCann, 1999), random spray (Provenzi et al., 2007) and PDE-based (Morel et al., 2009, 2010) are some examples. However, in the original publication (Land and McCann, 1971), retinex was based on random piecewise linear paths. While a detailed specification of those paths was not provided, Brainard and Wandell (1986) argued that they can be seen as random walk. Under the same

assumption, Morel et al. (2009, 2010) set up a framework where they could prove the equivalence between retinex and PDEs. Other than this use, the advantage of the random walk over other methods is limited. Given the nonlinearity of the retinex computations, the order in which the pixels are visited influences the output image. Funt et al. (2000, 2004) showed that the point spread functions for the multi-scale algorithms have a clear bias that depends on the visit order. In the random walk this does not happen, since the path is expected to diffuse equally in all directions.

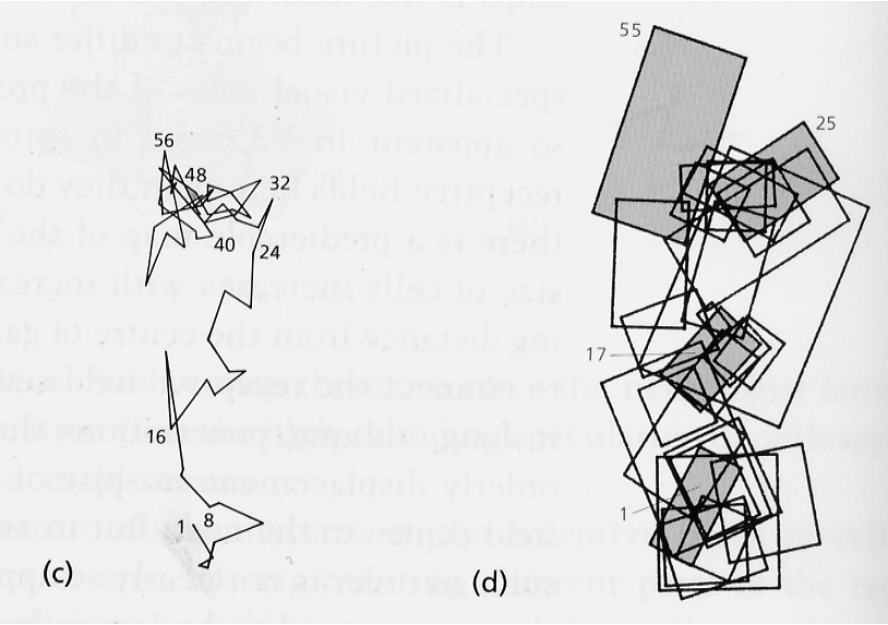


Figure 2.4: Distribution of receptors in the area V4 of the human visual cortex and their receptive fields (from Zeki, 1993, pp. 151–155).

While the benefits brought to retinex by the random walk may be limited, Marini and Rizzi (2000) showed that Brownian motion has instead some substantial advantages, at least from a theoretical point of view. They observed that the distribution of certain receptors in the area V4 of our visual cortex, described by Zeki (1993), resembles Brownian motion (see figure 2.4). Since retinex models human colour vision, Marini and Rizzi proposed an implementation based on Brownian paths and showed

that it had advantages over previous work. On the other hand, they had to admit that their algorithm is computationally expensive, because it has to generate several Brownian paths for each pixel. This is almost an obligatory choice: a single Brownian path would eventually visit all the pixels of an image, but in practice this would take long time.

2.5 Retinex algorithms

The retinex theory of colour vision, first introduced by Land (1964), was devised in an attempt to model certain features of human colour vision. Its aim was to reproduce colour constancy, that is, the capability of human vision of discriminating object colours independently from the illumination colour. Retinex algorithms achieve that by separating the effect of the illumination from the reflectance of surfaces. However, Land's experiments on perception were carried out in a "flat world" made of coloured patches, and this introduces some restrictions to retinex. Namely, there are two assumptions for it to work properly. First, the MaxRGB hypothesis has to hold, i.e., there has to be a surface having white reflectance in the scene, so that the colour of the light is given by the maximum value of each of the (R, G, B) channels (Land, 1977). Second, there cannot be specular highlights, which would be incorrectly interpreted as a white surface. These constraints are less severe than it might seem, thus retinex found several applications once digital imaging became more widespread. Image enhancement is one of its typical uses (e.g., Barnard and Funt, 1997; Rahman et al., 2004), which is also related to the recovering of photographs captured in poor visibility conditions (e.g., Jobson et al., 2003; Woodell et al., 2005). More recently, it proved valuable as a tone-mapping operator for high dynamic range images (e.g., Gadia et al., 2004; Meylan and Süsstrunk, 2006; Sobol, 2004).

Retinex algorithms estimate the lightness trying to separate illumination from re-

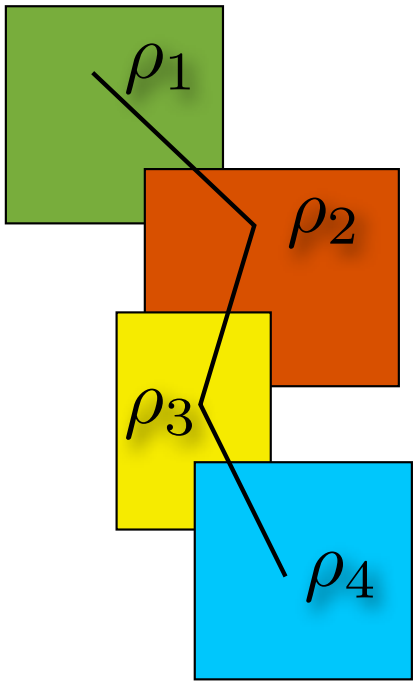


Figure 2.5: An application of path-based retinex to coloured patches. For a given estimate $i'(\rho_1)$ of the first patch, the estimate of the second patch is $i'(\rho_2) = i'(\rho_1) \cdot i(\rho_2)/i(\rho_1)$. Similarly, $i'(\rho_3) = i'(\rho_2) \cdot i(\rho_3)/i(\rho_2)$ and $i'(\rho_4) = i'(\rho_3) \cdot i(\rho_4)/i(\rho_3)$.

flectance, because Land and McCann (1971) suggest that the perception of colour is mainly due to the latter, rather than the former. In their paper, they consider a scene consisting of several coloured patches, which they call a Mondrian, and piece-wise linear curves across different patches. Thus, we can approximate pixel intensities as $i(\rho) = r(\rho)l(\rho)$ along a path ρ across the image, where $r(\rho)$ represents the reflectance and $l(\rho)$ the illumination. Land and McCann assume that across an image the reflectance varies sharply at the border between objects, whereas the illumination changes smoothly. Thus, to estimate the appearance $i'(\rho_n)$ at the nth pixel of a path ρ, they pro-

pose a sequence of ratios and products as follows:

$$i'(\rho_n) = i(\rho_1) \cdot \delta\left(\frac{i(\rho_2)}{i(\rho_1)}\right) \cdot \delta\left(\frac{i(\rho_3)}{i(\rho_2)}\right) \cdot \ldots \cdot \delta\left(\frac{i(\rho_n)}{i(\rho_{n-1})}\right) \approx \frac{r(\rho_n)}{r(\rho_1)}, \tag{2.8}$$

where δ represents a threshold operator over each ratio $i(\rho_k)/i(\rho_{k-1})$. To clarify this, let us consider figure 2.5 as an example. Retinex computes the ratio $i(\rho_2)/i(\rho_1)$, and will assign to the estimate $i'(\rho_2)$ for the patch ρ_2 the value $i'(\rho_2) = i'(\rho_1) \cdot i(\rho_2)/i(\rho_1)$. Similarly, $i'(\rho_3) = i'(\rho_2) \cdot i(\rho_3)/i(\rho_2)$, and so on. The estimate of the first patch is generally $i'(\rho_1) = 1$ and later we will explain why. The threshold δ arises from the assumption that illumination varies slowly, whereas reflectance varies abruptly. In other words, for each k we have $l(\rho_k) \approx l(\rho_{k-1})$, therefore whenever $i(\rho_k)/i(\rho_{k-1})$ is close enough to 1 we assume that the only the illumination contributes to the ratio, thus δ forces its value to 1, that is

$$\delta(x) = \begin{cases} 1 \text{ if } |x-1| \leq \tau \\ x \text{ if } |x-1| > \tau. \end{cases} \tag{2.9}$$

In practice, such a sequence of products and divisions is computationally very expensive (especially in the years when this method was devised), hence it is wiser to consider the logarithm of the image in order to transform products into sums and divisions into differences, so if $I(\rho) = \log i(\rho)$, and defining a threshold operator δ', equation (2.8) becomes

$$\begin{aligned} I'(\rho_n) = I(\rho_1) + \delta'\big(I(\rho_2) - I(\rho_1)\big) + \\ + \delta'\big(I(\rho_3) - I(\rho_2)\big) + \ldots + \delta'\big(I(\rho_n) - I(\rho_{n-1})\big). \end{aligned} \tag{2.10}$$

Here, the threshold becomes

$$\delta'(x) = \begin{cases} 0 \text{ if } |x| \leq \tau' \\ 1 \text{ if } |x| > \tau'. \end{cases} \tag{2.11}$$

This whole chain of operations would be performed several times for each pixel of an image; the final estimate would be given by the average of the estimates obtained from all the paths.

Over the years, several variations of the original retinex formulation have been proposed. A feature that has had a fundamental role is the reset step (Land and McCann, 1971). To estimate more accurately the reflectance of a certain area, this area has to be compared to a very bright area within the scene. That is to say, as previously mentioned, the pixel in position ρ_1 of each path is assumed to have reflectance 1. The reset was introduced because this assumption may not hold true, therefore it can happen that the sequence of ratios comes across a brighter area than its starting point. If this is the case, the overall product computed to there will have value greater than 1 (or, in logarithm, the overall sum will be greater than 0), thus a reset is triggered and the sequence of ratios restarts assuming reflectance 1 in the current position. This can be represented, for the pixel in position ρ_m, as

$$\begin{aligned} t(\rho_m) &= I'(\rho_{m-1}) + \delta'\big(I(\rho_m) - I(\rho_{m-1})\big) \\ I'(\rho_m) &= \min\{t(\rho_m), 0\}, \end{aligned} \tag{2.12}$$

where with $t(\,\cdot\,)$ we represent a temporary estimate. Another important modification was introduced by Frankle and McCann (1983) to the average step. In the early retinex description by Land and McCann (and also in more recent formulations, e.g., Marini and Rizzi, 2000), the average is taken only when the estimate from all the paths has been obtained. Frankle and McCann instead update the average every time a new estimate

has been computed, that is, as from equation (2.12),

$$I'_k(\rho_m) = \frac{1}{2}\left(I'_{k-1}(\rho_m) + \min\{t(\rho_m), 0\}\right), \tag{2.13}$$

where the k indicates the kth estimate to the pixel in position ρ_m. Reset and average are very important for many versions of retinex, whereas the threshold operation has been often dropped (e.g., Frankle and McCann, 1983; Funt et al., 2004; McCann, 1999; Provenzi et al., 2007), and received attention again only recently (Morel et al., 2010).

So far we have referred to path based retinex algorithms, however in the literature there are several examples of two-dimensional retinex. For example, Frankle and McCann (1983) consider differences between pixels at different and increasingly small distances, separating the horizontal and vertical directions. McCann (1999), using a slightly different approach, considers at each pixel the ratios with the eight neighbouring pixels, and the operation is repeated in several scales of the image. In the limit (infinite steps), it has been shown that Frankle and McCann's algorithm returns the original image scaled by a constant (see Brainard and Wandell, 1986). Instead, for a smaller number of steps per scale, it tends to enhance the detail in the image.

2.6 Conclusion

In this chapter we reviewed different types of paths that are used in image processing. In this field, all these types of paths are in practice space-filling curves. Peano-Hilbert curves are space-filling by definition, while the random walk and Brownian motion will eventually traverse all the pixels of an image when the number of steps is large enough. Hamiltonian paths can be used as context-based space filling curves, if they are based on spanning trees that follow minimum pixel differences. We also presented various applications of these paths, which are several and span different branches of image processing. Finally, we introduced the retinex algorithm, which will have a fundamental

role in chapters 4 and 5.

In the next chapter we will describe our first contribution in this thesis. Its topic, image indexing, has not been covered in this chapter because, to the best of our knowledge, path-based computations have not been applied to it. In short, image indexing consists in providing an image signature that reflects somehow the image content, so that similar signatures corresponds to similar images. In the next chapter we will show how we do this with an algorithm based on Lissajous paths.

Chapter 3

Lissajous curves in image indexing

Lissajous curves are perhaps better known because of their characteristic appearance rather than their applications. Various institutions are using a Lissajous curve as their logo, for example the MIT Lincoln Laboratory (2010), and devices have been built that display these curves purely for aesthetic purposes (Scarpelli, 1978). Mathematically they are defined as a family of parametric curves having the following equations

$$\begin{aligned} x(t) &= A\sin(\omega_x t + \delta_x) \\ y(t) &= B\sin(\omega_y t + \delta_y). \end{aligned} \tag{3.1}$$

In fact, definitions vary slightly in form but not in substance: the use of cosines is sometimes preferred (e.g., Moriguchi et al., 2000). In the formulation to which we refer, when the ratio ω_x/ω_y is rational the curve is closed (Cundy and Rollett, 1961), as shown for example in figure 3.1. Lissajous curves have practical applications in astrophysics, where they are used to describe certain types of orbits, also adopted in space missions (e.g., European Space Agency, 2009; Lo et al., 1998). Additionally, in knot theory, three-dimensional Lissajous curves are used to describe certain types of knots (Bogle et al., 1994).

To the best of our knowledge, the applications of Lissajous curves in image pro-

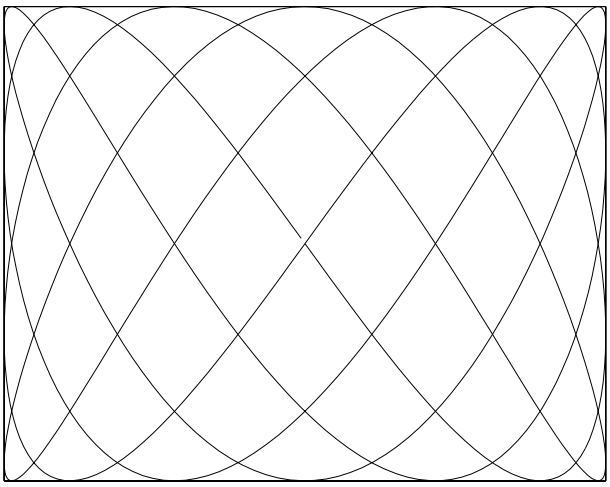


Figure 3.1: A Lissajous curve with parameters $A = B = 1$, $\omega_x = 4$, $\omega_y = 7$, $\delta_x = \delta_y = 0$.

cessing are limited. In our literature survey, we found that Moriguchi et al. (2000) proposed a method for reconstructing magnetic resonance images by sampling data along a Lissajous trajectory at equidistant values of t (figure 3.2). Then, they interpolate the samples to a Cartesian grid of equidistant points in the x and y axes. While their work is very different to ours in its scope, we will show that this underlying idea is not entirely dissimilar to an important tool we employ in this chapter, namely the Padua point interpolation method.

In this chapter we look at the problem of image indexing and retrieval. Several cues can be used in this task, and colour has proven to be a powerful one, especially in the form of colour and chromaticity histograms. Although in complex scenes these features might not provide high accuracy on their own, very fast algorithms can use them in order to narrow down the search for other more accurate, yet slower, algorithms that utilise other cues. Here, for chromaticity histograms we effectively sample the two-dimensional distribution at the self-intersection point of a specially chosen Lissajous

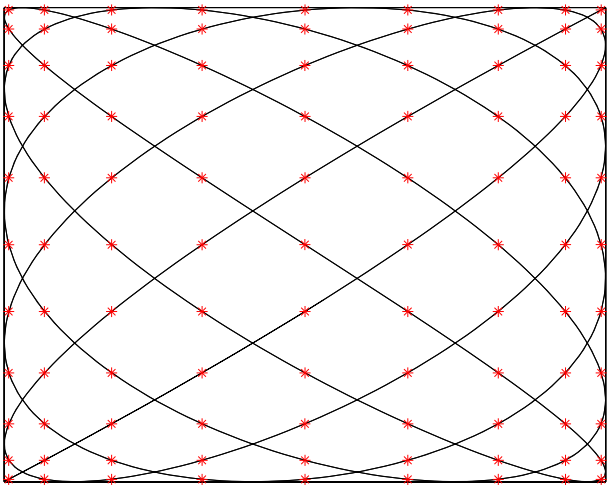


Figure 3.2: Sampling of a Lissajous curve according to the method described in the paper by Moriguchi et al. (2000).

curve. The advantage of doing this is that sampling at these "Padua points" (so called because of the institution where their importance was discovered) is that they have certain optimality properties. Specifically, if we reconstruct a function from its values at the Padua points then the maximum deviation from the original function is bounded. This seemed to us a powerful criterion for representing distributions. From the Padua point theory we represent a chromaticity distribution with a parsimonious signature for image indexing. We test the method for indexing different datasets and compare its performance with that of histogram intersection (Swain and Ballard, 1991). We also consider two other dimensionality reduction methods based on the discrete cosine transform and principal component analysis, which in previous work showed excellent results in image indexing and retrieval tasks (Berens et al., 2000). Our experiments show that the Padua points deliver a compact representation that supports a modest increase of the indexing performance compared with the other two methods.

3.1 The Padua points

Polynomial interpolation is a powerful approximation tool, and probably one of the oldest known ways to fit a certain set of data with a function. However, when implemented naïvely, the error in the polynomial approximation may be unacceptable: the so-called Runge phenomenon causes heavy oscillations that grow with the interpolation degree. To limit the approximation error, in one variable one can interpolate using the Chebyshev nodes (e.g., Mathews and Fink, 1999; Press et al., 2007) – because of the position of these nodes in the domain $[-1, 1]$, the approximating polynomial is quasi-optimal in the sense that its error differs up to a logarithmic factor from the error of the optimal approximating polynomial in a ℓ^∞-norm sense (known as minimax polynomial). This difference is measured by the Lebesgue constant, which grows with the interpolation degree. It is obviously desirable that this growth be slow, and Brutman (1997) proved it to have a lower bound of $\mathrm{O}(\log n)$ in the case of interpolation of one variable, and of $\mathrm{O}(\log^2 n)$ in two variables, where n is the interpolation degree. To prove that an interpolation method is quasi-optimal, one has to compute its Lebesgue constant. If a method is non-optimal, its Lebesgue constant will have a faster growth than the aforementioned lower bounds. The advantage of the quasi-optimal interpolation is simplicity, as it is computationally demanding to calculate the minimax polynomial.

If in one variable the Chebyshev nodes are bound to be optimal in terms of the growth of the Lebesgue constant, for interpolation in two variables only recently Caliari et al. (2005) have introduced a set of points that satisfy this property. These points are called Padua points, after the Italian university where they were first discovered. To date, they are the only known points that guarantee unisolvence (i.e., their interpolating polynomial is unique) and have a proven minimal growth of the Lebesgue constant of $\mathrm{O}(\log^2 n)$ for interpolation in two variables (Bos et al., 2006, 2007).

The Padua points are defined in the square $[-1, 1] \times [-1, 1] \subset \mathbb{R}^2$ (but they can be easily mapped to any rectangular domain) as the union of two grids of Chebyshev

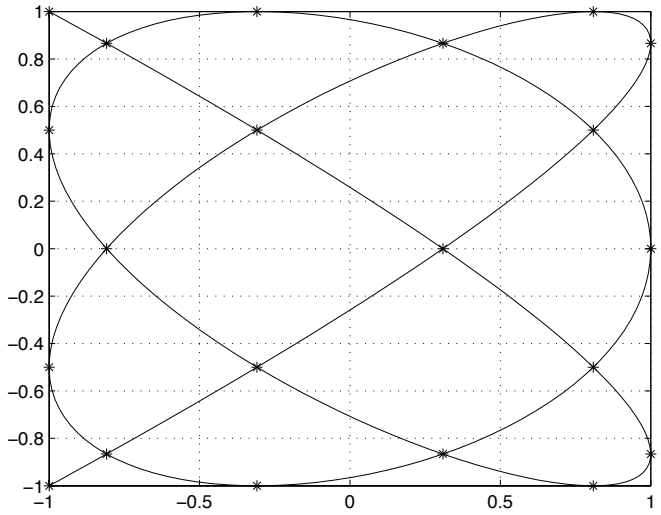


Figure 3.3: Padua points of the first family of interpolation degree $n = 5$ and their generating Lissajous curve as in equation (3.3).

points. Given the interpolation degree n, we have $N = \frac{1}{2}(n+1)(n+2)$ points defined as

$$\begin{aligned} \text{Pad}_n &= \{\boldsymbol{\xi} = (\xi_1, \xi_2)\} \\ &= \left\{ \gamma\left(\frac{k\pi}{n(n+1)}\right),\ k = 0, \ldots, n(n+1) \right\}, \end{aligned} \tag{3.2}$$

where $\gamma(t)$ is their generating curve (see Bos et al., 2006). In practice, the Padua points are obtained sampling a Lissajous curve at equidistant values of t. This is substantially the same principle as that described by Moriguchi et al. (2000), although their sampling was different and thus their interpolation scheme could not be optimal. There are four families of Padua points, obtained with subsequent 90 degree rotations of the following Lissajous curve:

$$\gamma(t) = (-\cos((n+1)t), -\cos(nt)) \quad t \in [0, \pi]. \tag{3.3}$$

Interestingly, the samplings of this curve (or its rotations) at the Padua points fall in very particular positions. Two of them lie on consecutive vertices of the square, $2n-1$ lie on the edges of the square, and the remaining lie on the self-intersections of the curve (see figure 3.3).

Given a continuous function $f\colon [-1,1]^2 \to \mathbb{R}$, the interpolation polynomial $\mathcal{P}_n$ of degree n obtained with the Padua point method has the following formulation:

$$\mathcal{P}_n f(\mathbf{x}) = \sum_{k=0}^{n} \sum_{j=0}^{k} c_{j,k-j} \hat{T}_j(x_1) \hat{T}_{k-j}(x_2) \approx f(\mathbf{x}), \tag{3.4}$$

where $\hat{T}_i(\,\cdot\,)$ are the normalised Chebyshev polynomials of the first kind of degree i, that is,

$$\begin{aligned} \hat{T}_0(x) &= T_0(x) = 1 \\ \hat{T}_i(x) &= \sqrt{2} T_i(x) = \sqrt{2}\cos(i \arccos(x)). \end{aligned} \tag{3.5}$$

Provided that $\boldsymbol{\xi} = (\xi_1, \xi_2)$ are the Padua points for interpolation degree n, we would like to have a simple formula to obtain the coefficients $c_{j,k-j}$. This can be done as follows:

$$\begin{aligned} b_{j,k-j} &= \sum_{\boldsymbol{\xi} \in \mathrm{Pad}_n} f(\boldsymbol{\xi}) w_{\boldsymbol{\xi}} \hat{T}_j(\xi_1) \hat{T}_{k-j}(\xi_2), \quad 0 \le j \le k \le n \\ c_{j,k-j} &= b_{j,k-j}, \quad c_{n,0} = \frac{1}{2} b_{n,0}, \end{aligned} \tag{3.6}$$

where $f(\boldsymbol{\xi})$ is the value at each Padua point of the function we are going to approximate and $w_{\boldsymbol{\xi}}$ are the weights corresponding to each Padua point, defined as

$$w_{\boldsymbol{\xi}} = \frac{1}{n(n+1)} \cdot \begin{cases} \frac{1}{2} & \text{if } \boldsymbol{\xi} \text{ is a vertex point} \\ 1 & \text{if } \boldsymbol{\xi} \text{ is an edge point} \\ 2 & \text{if } \boldsymbol{\xi} \text{ is an interior point.} \end{cases} \tag{3.7}$$

In practice, once we have determined the interpolation degree and thus the Padua points, the only thing we need to know to perform the interpolation is the value $f(\boldsymbol{\xi})$. In other words, the "inputs" for the construction of an interpolating polynomial are only its degree and the value of the original function at each Padua point. The algorithm for computing the coefficients $c_{j,k-j}$ requires $\mathrm{O}(n^3)$ steps, where n is the interpolation degree (Caliari et al., 2008). A faster algorithm based on the FFT is possible in theory, having complexity $\mathrm{O}(n^2 \log n)$, however in practice it is faster only for very large n (Caliari et al., 2011; Montagna, 2007).

Finally, we would like to remark on what the minimal growth of the Lebesgue constant implies. As above, it provides the theoretical growth of the approximation error. Bos et al. (2006) proved that given a function $f\colon [-1,1]^2 \subset \mathbb{R}^2 \to \mathbb{R}$ and its Padua point approximation $\mathcal{P}_n f$,

$$E_n(f) = \|f - \mathcal{P}_n f\|_\infty \leq c(f;p) n^{-p} \log^2 n, \tag{3.8}$$

where p is the continuity of the function (that is, $f \in C^p([-1,1]^2)$) and $c(f;p)$ is a constant related both to the function f and to its continuity p. In a more compact notation, we can write the approximation as

$$E_n(f) \in \mathrm{O}(n^{-p} \log^2 n). \tag{3.9}$$

Put another way, for a given function or distribution, and for a certain interpolation degree, we understand a priori how well we can fit the data. Indeed, since n is small, we know a priori the fit will be good.

There are two features of the Padua points which are especially suitable for compression purposes. First of all, once we know the degree n of the interpolation we have a formula to determine the points. This means that there is no need to store their coordinates. Second, the interpolation formula for the Padua points can be written in the

bivariate Chebyshev polynomials orthonormal basis. This means that we can, without ambiguity, identify an interpolating polynomial from its coefficients. Together with unisolvence, this feature can be seen as a valuable tool for the identification of a given function.

3.2 Histogram approximation and comparison

The Padua points are defined in a square in $\mathbb{R}^2$, as above, but there is no known equivalent in higher dimension domains. For this reason we can only represent two-dimensional histograms. In our experiments we approximated chromaticity histograms from the RGB colour space, with axes (r, g) defined as

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B} \tag{3.10}$$

and hue-saturation histograms, with axes (H, S), from the well-known HSV colour space (e.g., Gonzalez et al., 2004). This provides a smooth function, represented by the interpolating polynomial of the Padua points, that approximates the density of the pixels projected onto the (r, g) or onto the (H, S) plane.

The main problem with approximating a histogram with a polynomial is described concisely in equation (3.9): the smoother the original function (i.e., the larger p), the lower the approximation error. The original function here is a histogram that is intrinsically not smooth, since it is a discrete representation. Our approach to address this issue is not to build a smooth density function that approximates a histogram, but rather to build the density function directly. Since all that we need to know to obtain the interpolating polynomial are the values of some function at the coordinates of the Padua points, we weight each pixel with a Gaussian distribution. More precisely, we perform the following steps:

1. Project each pixel of the image on to the domain (r, g) or (H, S).

2. Compute the coordinates of the Padua points in the same domain (which is $[0, 1] \times [0, 1] \subset \mathbb{R}^2$).

3. Build a bivariate Gaussian distribution centred on a Padua point $\boldsymbol{\xi} = (\xi_1, \xi_2)$ (i.e., with mean equivalent to the coordinates of $\boldsymbol{\xi}$).

4. Compute the weights of all pixels according to the distance from the considered Padua point using the Gaussian function constructed in the previous step, then assign their sum to $f(\boldsymbol{\xi})$ (i.e., $f(\boldsymbol{\xi})$ measures the density of pixels around each Padua point).

5. Repeat the procedure from step 3 for each Padua point.

In this way we get the values of $f(\boldsymbol{\xi})$ for each Padua point, which is sufficient for us to compute the coefficients $c_{j,k-j}$ of equation (3.6), and store them into a characteristic vector $\mathbf{C}$. This vector can be regarded as an image signature, or rather as a signature of its chromaticity distribution. Moreover, because the Padua points are unisolvent, we obtain one and only one interpolating polynomial and therefore a unique signature $\mathbf{C}$. An example of what a chromaticity distribution approximated in this way looks like is shown in figure 3.4 on the next page.

In this framework, the choice of the Gaussian plays an important role too. If it has a large standard deviation then all pixels will contribute significantly to the distribution at all Padua points. Conversely, if the standard deviation is too small then we may not completely measure datasets (in the worst case all chromaticities could be unique). We chose the standard deviation of our Gaussian empirically. After running several preliminary tests on Swain and Ballard's dataset (see section 3.3.1), we found that a standard deviation of 0.006 has the best retrieval performance in both (r, g) and (H, S) colour spaces (chromaticities lie in the unit square, so 6 standard deviations, i.e., the effective

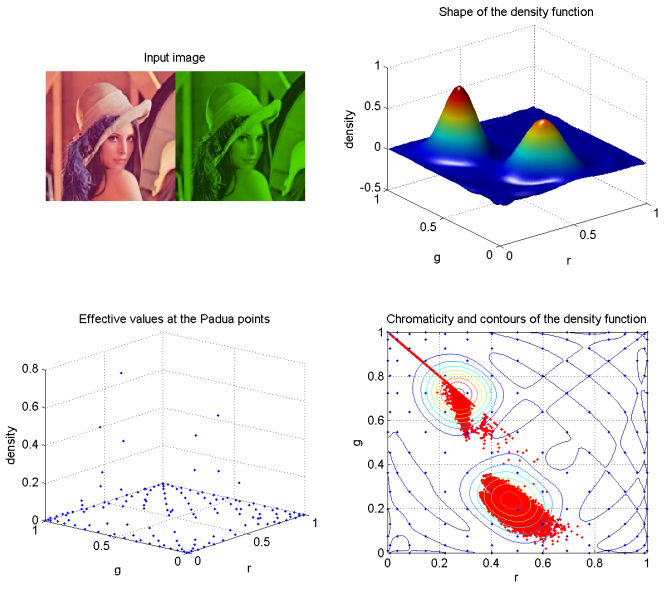


Figure 3.4: An example of a chromaticity distribution function. We chose the top-left image in order to show more clearly how the Padua points adapt to the shape of the distribution. The top-right plot is the function generated by the Padua points and the bottom-left one shows the values at each of the Padua points. Finally the bottom-right plot is the chromaticity distribution on the plane, combined with a contour plot of the Padua point function and the positions of the Padua points themselves.

support of a Gaussian, are about 3.5% of the extent of the domain). However, we found also that on different datasets other standard deviations may perform better. For this reason, we ran further preliminary tests on subsets of each dataset, and the values found in this way (reported in section 3.5) were then used in the final experiments.

Here we outline the steps required to match an unknown image in an archive of M model images.

1. Compute the characteristic vectors of the model images $\mathbf{C}_1, \ldots, \mathbf{C}_M$ (in a real scenario we may assume that they are already stored with the images).

2. Compute the characteristic vector of the unknown image $\mathbf{C}_u$.

3. Sort the model images according to their distance $d(u, i) = \|\mathbf{C}_u - \mathbf{C}_i\|_2$ from the

unknown one.

The image that is the "closest" to the unknown one in terms of the 2-norm distance will be the best match.

3.2.1 Implementation issues

The Padua point interpolation used in this work is based on the API provided in the package XuPad2D (Caliari et al., 2007). With these functions, it is straightforward to generate the Padua points, and then to obtain the coefficients of the interpolating polynomial. Once we have those, it is also very easy to evaluate the polynomial on any set of points. Generating the Padua points is a very quick task. The bottleneck in our method is actually evaluating the Gaussian functions (one for each Padua point) at each pixel projected to the two dimensional space. Naïvely, if we have N Padua points and an image of M pixels, we need to perform $\mathrm{O}(MN)$ operations to compute this sum. Fortunately, Greengard and Strain (1991) developed a quicker algorithm, known as the Fast Gauss Transform, that requires only $\mathrm{O}(M + N)$ operations. The only drawback is that it computes an approximation to the "exact" solution, however the error introduced is marginal and can, for the purpose of this work, be ignored. Only after this weighted sum has been calculated it is possible to obtain the coefficients of the interpolating polynomial, which are then used for the matching.

3.3 Image retrieval experiments

In order to test the efficacy of our retrieval system, we performed an experiment that consisted of creating an archive of model images, and then querying the archive with unknown images. The retrieval task consisted of sorting the model images according to the distance from the unknown image. In general, the datasets we used contained several models corresponding to any given unknown image.

To evaluate the performance of our method we used *precision* and *recall* (e.g., Müller et al., 2001). Given the set of documents T (in our case a set of images) retrieved by a query and the set of relevant documents R in a collection, precision and recall are defined as

$$\begin{aligned} \text{precision} &= \frac{|T \cap R|}{|T|} \\ \text{recall} &= \frac{|T \cap R|}{|R|}, \end{aligned} \tag{3.11}$$

where with $|\cdot|$ we denote the number of elements in a set. These two values are to be considered in relation to each other because they are not meaningful if taken separately. For example, it is trivial to have 100% recall by retrieving all the documents of the collection. Müller et al. (2001) suggest various ways to use precision and recall as performance measures for content-based image retrieval systems, among them:

- $P(0.5)$, precision at 50% recall (i.e., precision after retrieving half of the relevant documents) and
- P_{N_r}, precision after N_r documents are retrieved, where N_r is the number of relevant documents in the collection (this value is equal to the recall after N_r documents are retrieved).

In agreement with these directions, after some preliminary tests we decided to adopt the following measures:

- $P(0.5)$, as above;
- $P(0.75)$, precision at 75% recall;
- $P(1)$, precision at 100% recall (i.e., precision after retrieving all relevant documents);
- P_{N_r}, as above.

Finally, we also wanted to measure the significance of the results of our experiments. For this purpose, we adopted the signed-rank test (e.g., Devore and Peck, 1986) to compare our method to the best performing of the others. The test statistic will say whether the difference in performance is significant or not. We explain this method with a toy example, in table 3.1, in which we compare two indexing methods called X and Y. Here, as previously explained, the two methods will sort all the M images contained in the archive according to the distance from a given unknown image. We assume we are querying the archive with six unknown images, and that we measure the P_{N_r} for each query. With the signed-rank test, for each unknown image we consider the difference between the two methods, and we test the null hypothesis $H_0 \colon \mu_d = 0$ that the mean difference is zero. In the table, we show the P_{N_r} for each query for both methods. After computing the difference in the ranking, the differences have to be sorted by

Unknown images	I_1	I_2	I_3	I_4	I_5	I_6
P_{N_r} for X	0.2	1	0.2	0.8	0.15	0.7
P_{N_r} for Y	0.5	0.4	0.4	0.9	0.3	0.3
$X - Y$	-0.3	0.6	-0.2	0.1	-0.15	0.4
Signed-rank Z	-4	6	-3	1	-2	5

Table 3.1: Example of the signed-rank test. In the second and third row we show how each of the five relevant images were ranked by methods X and Y. The difference between the two is used to compute the signed-rank.

absolute value. In our example this yields $0.1, 0.15, 0.2, 0.3, 0.4, 0.6$, therefore the ranks for $X - Y$ are $4, 6, 3, 1, 2, 5$, to which we have to apply the sign of the corresponding difference $X - Y$ to obtain the signed-ranks as in the table. Finally, we compute the signed-rank sum

$$S = \sum_{i=1}^{6} Z_i = 3. \tag{3.12}$$

Values of $|S|$ are tabulated for different numbers of samples (in our case six). With $S = 3$, according to the tables in Devore and Peck (1986), we cannot reject null

hypothesis that the mean $\mu_d = 0$. When the number of observations is large, the signed-rank sum is approximated with a normal distribution having standard deviation $\sigma = \sqrt{n(n+1)(2n+1)/6}$ (with n number of samples), thus the value $z = S/\sigma$ is compared against the critical values of the standard normal distribution. In the actual experiments we will not only use the P_{N_r} measure, but also all the others previously proposed.

3.3.1 Swain and Ballard's dataset

The first dataset we used is that from Swain and Ballard (1991). The dataset consists of 90 pictures of different objects on a black background. Of these, 63 are model pictures and 27 are unknown images. Each of the unknowns corresponds to one of the models. Compared to the models, the unknown objects may be translated, rotated, scaled, occluded and deformed. Because of the structure of this dataset, it is not possible to use precision and recall as metrics for evaluating the retrieval results of tested methods: each query has only one exact match, therefore, as seen in previous use (e.g., Berens et al., 2000; Swain and Ballard, 1991), the most suitable metric in this case is the average match percentile. For this reason, and for its small size, we used this dataset only for preliminary tests.

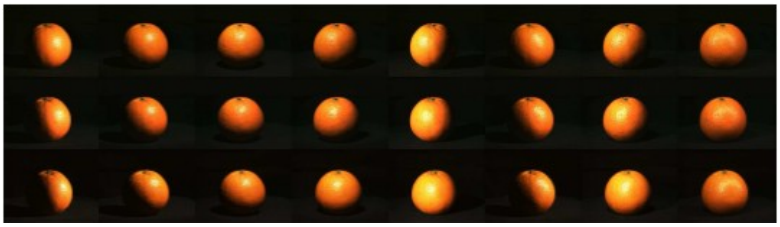


Figure 3.5: An example of the viewing conditions of an object belonging to the ALOI dataset (Geusebroek et al., 2005).

3.3.2 The Amsterdam Library of Object Images

Since our method projects the image onto the chromaticity plane, it should not theoretically be affected by differences in light intensity, therefore we chose a dataset where it is possible to test this capability. The Amsterdam Library of Object Images (ALOI) has exactly this feature. This and the large size of the dataset persuaded us to choose it for testing our method. The library displays a thousand objects, each in several configurations: in particular, we were interested in the images captured under different illumination directions. Geusebroek et al. (2005) described the set where the pictures were taken. The object was surrounded by five lights, and the camera could be set-up in one of three different geometries: straight in front of the object, turned at 15 to the object and turned at 30 degrees to the object. In addition the object could also be turned with respect to the camera, so that the camera movement could be compensated. In some cases the object was photographed with only one of the five lights turned on, but two pictures were also taken where two lights (two on the right side and two on the left side) were simultaneously on, and finally one where all the lights were on. For each of these light conditions, a picture was taken from all three positions of the camera. This resulted in 24 different illumination conditions for each object, as in figure 3.5. From the 24 versions of each object, we selected the one with the most uniform illumination as the query: we had therefore a set of a thousand query images, and all the remaining 23 000 images were treated as models.

3.3.3 The Columbia Object Image Library

Another smaller dataset is the Columbia Object Image Library (COIL-100, see Nene et al., 1996), which is widely used in pattern recognition. It features 7200 pictures of a hundred objects (see figure 3.6), which are each photographed from 72 different angles. Their shape and size can be different from different viewing angles, but histograms are

Figure 3.6: The 100 objects photographed in the COIL-100 dataset (Nene et al., 1996).

not sensitive to scaling, and we expect the Padua point representation to share this property. However, the colour distribution may also vary across different images: objects may change their colour content depending on the point of view.

3.3.4 The MPEG-7 dataset

Finally, for the sake of completeness we ran our tests on the MPEG-7 dataset (International Organisation for Standardisation, 2004). This dataset has a different structure from those described above. It features a total of 5466 images, consisting of photos and sequences of video frames. Arguably, it is much more difficult than the other datasets described here for a colour based only retrieval task. In the other datasets, the matches consist of objects per se; here the matches are real scenes slightly different from each

other. Its query set is provided and is composed by 50 ground-truth sets, each containing a query image and some (from 2 to 32) images highly similar to the query, all of them belonging to the database, for a total of 50 query images and 386 ground-truth images (each query image is included in its own ground-truth set, but in our case it will always score as best match). This is the only dataset we use where not all the images are involved in the queries.

3.4 Methods tested in our experiments

Histogram intersection The histogram intersection (HI), introduced by Swain and Ballard (1991), features a direct comparison between colour or chromaticity histograms. Given two histograms H and H' containing each m bins, their intersection is

$$I(H, H') = \frac{\sum_{j=1}^{m} \min(H_j, H'_j)}{\sum_{j=1}^{m} H_j}. \tag{3.13}$$

The larger this value, the more similar the two histograms are (if one visualises the two histograms, this measure is equivalent to their overlapping area). We used this method with chromaticity histograms of different sizes: 32×32, 16×16 and 8×8.

Discrete cosine transform The discrete cosine transform (DCT) is widely used as a method for dimensionality reduction (among its other applications, it is the core of the JPEG image compression), and already employed in colour indexing. In order to maintain consistency with the previous work of Berens et al. (2000), we took the DCT of a 16×16 chromaticity histogram, and we performed the indexing using the 66 lower frequencies.

Hybrid transform The hybrid transform (HT) is a method introduced by Berens et al. (2000) based on the DCT and on PCA (principal component analysis, also known as

the Karhunen-Loêve transform or KLT). In this case, we applied the DCT to a 32×32 chromaticity histogram and we selected the lower 136 frequencies, where we applied PCA. After that, we used the 66 larger principal components for the indexing. PCA is a very powerful statistical tool that is optimal in a least-squares sense as a method for dimensionality reduction. However, PCA needs to be trained for each dataset. Histogram intersection, the Padua point method and the DCT perform the indexing directly on their output, i.e., if we want to add some images to our database we can simply compute their indexing vectors and compare them to any query image. In the HT instead, we found the best score when the training and the indexing test were performed on the same dataset. Here we were also interested in finding a worst case, that is, training on one dataset and indexing another (e.g., training on the ALOI and experiment on the MPEG-7). Finally, we considered also an average case, where PCA was trained on all the datasets.

Padua point method While we have already discussed widely how the Padua point method (PP) works, we have not yet specified the amount of compression we used. We compressed the histogram firstly with a polynomial of degree $n = 30$ (i.e., storing 496 coefficients), then with a polynomial of degree $n = 10$ (66 coefficients). We will abbreviate these two methods with PP 30 and PP 10, respectively. Furthermore, in analogy to the HT, where 66 principal components out of 136 are returned, the indexing vector of the Padua points also could be truncated. We propose to consider PP 30 (496 coefficients), but in this case performing the indexing with the first 66 coefficients only, which correspond to the polynomials of lower degree (later we will refer to this as PP 30-10). In fact, in this case the indexing is effectively performed on polynomials of degree $n = 10$, because we are discarding the coefficients of the higher degree polynomials; on the other hand, we are taking advantage of the finer sampling in the domain, which is performed with 496 points instead of 66. It is important to stress that in practice this

lower degree polynomial still minimises the ℓ^∞ norm of the error, because of the properties of the Chebyshev polynomials (e.g., Press et al., 2007, pp. 233–239). The only difference is that with the PP 10 and PP 30 methods the polynomial is an interpolation of the function, while with PP 30-10 the polynomial is only an approximation.

3.5 Indexing results

In this section we describe the results obtained from each of the tests we conducted. In each of the tables we enumerate the methods, the number of coefficients stored and the results for each of the metrics described in section 3.3. We divide each table in two groups: methods that use a large signature and methods that use a small signature. For both groups, the best scores for each metric are in bold face.

	Type	Coefficients	$P(0.5)$	$P(0.75)$	$P(1)$	P_{N_r}
Large	HI	1024	36.26%	19.93%	6.77%	39.73%
	PP 30	496	**37.79%**	**21.73%**	**7.71%**	**39.40%**
	HI	256	27.12%	14.88%	4.10%	31.27%
Small	HT best	66	28.27%	15.43%	4.94%	32.86%
	HT avg	66	28.25%	15.42%	4.93%	32.85%
	HT worst	66	28.05%	15.20%	4.88%	32.66%
	PP 30-10	66	**34.06%**	**20.45%**	**7.66%**	**35.99%**
	PP 10	66	27.98%	15.93%	4.94%	30.70%
	DCT	66	22.03%	12.10%	3.65%	26.77%
	HI	64	13.98%	7.11%	2.00%	18.65%
PP 30-10 vs. HT best	z value		8.13	10.62	13.45	5.62
	p-value		< 0.001	< 0.001	< 0.001	< 0.001

Table 3.2: Comparison of the results of histogram intersection, Padua points ($\sigma^2 = 0.001$ for $n = 30$ and $\sigma^2 = 0.0061$ for $n = 10$), DCT and HT (trained on the MPEG-7 dataset for the worst case) in the (r, g) space on the ALOI dataset. The signed-rank test between PP 30-10 and HT best shows that the difference of performance is significant in all the measures.

Table 3.2 summarises the test results on the ALOI dataset using the (r, g) chromaticity space. In this experiment the Padua points score extremely well, beating the retrieval

performance of the histogram intersection on 1024 bins when storing roughly half of the coefficients. In terms of dimensionality reduction we obtained the best results with the PP 30-10 method. Storing just 66 coefficients, it performs much better than the other considered dimensionality reduction methods, and it almost reaches the performance of the histogram intersection that stores 1024 coefficients. The improvement over the best results obtained with the HT is statistically significant with $p < 0.001$ for all the measures. In this dataset the Padua points score particularly well, considering that even the PP 10 method scores roughly the same as the HT and the histogram intersection with 256 coefficients.

	Type	Coefficients	$P(0.5)$	$P(0.75)$	$P(1)$	P_{N_r}
Large	HI	1024	**36.48%**	**20.09%**	**6.62%**	**40.26%**
	PP 30	496	26.68%	13.63%	4.07%	32.85%
	HI	256	30.54%	17.02%	5.43%	36.01%
Small	HT best	66	21.83%	11.24%	3.31%	28.97%
	HT avg	66	21.10%	10.82%	3.15%	28.56%
	HT worst	66	20.06%	10.22%	2.99%	28.01%
	PP 30-10	66	**26.14%**	**13.05%**	**3.85%**	**32.08%**
	PP 10	66	20.98%	10.47%	3.07%	28.48%
	DCT	66	23.36%	11.95%	3.78%	29.88%
	HI	64	25.71%	13.10%	3.59%	31.50%
PP 30-10 vs. HT best	z value		11.66	12.95	12.34	9.57
	p-value		< 0.001	< 0.001	< 0.001	< 0.001

Table 3.3: Comparison of the results of histogram intersection, Padua points ($\sigma^2 = 0.001$ for $n = 30$ and $\sigma^2 = 0.0061$ for $n = 10$), DCT and HT (trained on the MPEG-7 dataset for the worst case) in the (H, S) space on the ALOI dataset. The signed-rank test between PP 30-10 and HT best shows that the difference of performance is significant in all the measures.

In table 3.3 we show the results of the same experiment run on the (H, S) coordinates. Once again, the Padua points perform better than the other dimensionality reduction methods. The improvement of PP 30-10 over the HT is statistically significant with $p < 0.001$ for all the measures. However, if we compare these results with those in table 3.2, we notice that while the performance of the histogram intersection and of

the DCT improves, all other methods work substantially better in the (r, g) space.

	Type	Coefficients	$P(0.5)$	$P(0.75)$	$P(1)$	P_{N_r}
Large	HI	1024	81.22%	**61.59%**	**49.14%**	**74.54%**
	PP 30	496	**81.41%**	59.01%	46.94%	72.70%
	HI	256	80.27%	60.44%	47.69%	73.42%
Small	HT best	66	**78.75%**	55.63%	**44.76%**	**70.62%**
	HT avg	66	**78.75%**	55.57%	44.70%	70.56%
	HT worst	66	77.98%	54.67%	43.86%	70.01%
	PP 30-10	66	78.27%	**55.77%**	42.04%	69.73%
	PP 10	66	73.91%	50.20%	37.53%	65.69%
	DCT	66	75.57%	53.49%	42.05%	68.66%
	HI	64	66.25%	46.20%	37.19%	61.35%
PP 30-10 vs. HT best	z value		0.84	0.31	2.05	1.18
	p-value		> 0.2	> 0.2	< 0.05	> 0.2

Table 3.4: Comparison of the results of histogram intersection, Padua points ($\sigma_x = 0.0004, \sigma_y = 0.0009$ for $n = 30$ and $\sigma^2 = 0.0061$ for $n = 10$), DCT and HT (trained on the ALOI dataset for the worst case) in the (r, g) space on the COIL-100 dataset. There is no statistical significance in the difference between PP 30-10 and HT best, except for the measure $P(1)$, which is in favour of the latter.

In table 3.4 and table 3.5 on the following page we show the results of the same experiment run on the COIL-100 dataset. The histogram intersection still obtains the best performance in both colour spaces, at the cost of a larger amount of data to be stored; however, in the (H, S) domain it scores very well even with only 64 bins (significantly better than PP 30-10, $z = 33.9$). As for the dimensionality reduction, in this case the HT achieves the best results, even in its worst case (in fact, in (r, g) the difference between the best and the worst case is almost negligible). The Padua points instead perform consistently better than the DCT, with some differences between the two colour spaces. The results obtained from the COIL-100 dataset confirm that the Padua points perform better in the (r, g) domain than in (H, S), as the tests on the ALOI dataset showed.

The tests on the MPEG-7 dataset, in table 3.6 and table 3.7, see once again the histogram intersection giving the best results, with the drawback of storing larger amounts

	Type	Coefficients	$P(0.5)$	$P(0.75)$	$P(1)$	P_{N_r}
Large	HI	1024	**84.85%**	**65.06%**	**53.53%**	**77.52%**
	PP 30	496	64.05%	47.38%	38.60%	63.72%
	HI	256	80.47%	61.35%	49.63%	74.77%
Small	HT best	66	72.83%	49.55%	39.85%	68.17%
	HT avg	66	71.14%	47.86%	38.97%	66.93%
	HT worst	66	68.22%	45.14%	38.19%	65.00%
	PP 30-10	66	64.00%	47.33%	38.47%	63.59%
	PP 10	66	62.85%	46.63%	39.10%	63.58%
	DCT	66	55.37%	43.61%	37.68%	61.08%
	HI	64	**76.11%**	**58.44%**	**46.74%**	**71.46%**
PP 30-10 vs. HI	z value		5.54	6.65	5.91	6.27
	p-value		< 0.001	< 0.001	< 0.001	< 0.001

Table 3.5: Comparison of the results of histogram intersection, Padua points ($\sigma^2 = 0.0098$ for $n = 30$ and $\sigma^2 = 0.0061$ for $n = 10$), DCT and HT (trained on the MPEG-7 dataset for the worst case) in the (H, S) space on the COIL-100 dataset. The signed-rank test between PP 30-10 and HI shows that the difference of performance is significant in all the measures.

of data. Once again the HT obtains the highest scores among the dimensionality reduction methods that store 66 coefficients, although its results are not significantly better than those of PP 30-10. Remarkably, there is almost no difference at all between the best and the average case of the HT. Moreover, in (H, S) the worst case scores better than the average case. As for the Padua points, while in the (r, g) domain their results are comparable to those of the HT, in (H, S) they are substantially worse. In fact, that is the case where the Padua points perform the worst, and the only one where their result is overtaken by that of the DCT. Note that in table 3.7 we computed the significance of the difference between the Padua points and the HT best, since for each metric a different method has the best score.

	Type	Coefficients	$P(0.5)$	$P(0.75)$	$P(1)$	P_{N_r}
Large	HI	1024	**81.78%**	**70.13%**	**44.51%**	**70.76%**
	PP 30	496	77.77%	54.70%	37.21%	63.45%
	HI	256	72.03%	48.90%	30.51%	61.98%
Small	HT best	66	**73.84%**	48.98%	**34.44%**	**61.76%**
	HT avg	66	**73.84%**	48.98%	34.42%	**61.76%**
	HT worst	66	73.74%	48.55%	34.16%	60.64%
	PP 30-10	66	71.68%	**50.35%**	31.41%	61.36%
	PP 10	66	58.30%	39.69%	21.54%	54.01%
	DCT	66	71.09%	44.12%	29.04%	58.13%
	HI	64	53.97%	38.52%	22.07%	52.09%
PP 30-10 vs. HT best	z value		0.77	0.06	1.10	0.01
	p-value		> 0.2	> 0.2	> 0.2	> 0.2

Table 3.6: Comparison of the results of histogram intersection, Padua points ($\sigma^2 = 0.0012$ for $n = 30$ and $\sigma^2 = 0.0061$ for $n = 10$), DCT and HT (trained on the COIL-100 dataset for the worst case) in the (r, g) space on the MPEG-7 dataset. The signed-rank test between PP 30-10 and HT best shows that there is no statistical significance in the difference between the two methods.

3.6 Minimisation of ℓ^p norms

The Padua point approach minimises the maximum deviation from the original data, or the ℓ^∞ norm of the error. In this section we try to understand if there is a relation between the type of norm that we minimise and the image indexing performance.

Verifying whether a certain norm minimisation is enough for obtaining a good indexing performance can be done by considering the least-squares approximation, which minimises the ℓ^2 norm of the residuals and can be generalised to minimise any ℓ^p norm. Although the retrieval task is similarly based on the coefficients of the approximating polynomial, we do not perform in this case the Gaussian smoothing and we build the approximation directly from the histogram, since we are not limited by the position of the interpolation points in the domain. The technique is almost the same as the one described in section 3.2.

	Type	Coefficients	$P(0.5)$	$P(0.75)$	$P(1)$	P_{N_r}
Large	HI	1024	**88.34%**	**79.27%**	**57.56%**	**77.59%**
	PP 30	496	70.98%	51.79%	31.39%	62.73%
	HI	256	83.38%	74.18%	50.83%	74.03%
Small	HT best	66	76.99%	63.54%	**45.74%**	69.86%
	HT avg	66	74.82%	63.84%	40.98%	**70.28%**
	HT worst	66	78.10%	63.62%	43.32%	68.10%
	PP 30-10	66	69.04%	45.69%	28.45%	60.26%
	PP 10	66	66.27%	41.33%	25.48%	57.31%
	DCT	66	**79.45%**	60.03%	42.41%	69.98%
	HI	64	79.31%	**66.72%**	40.45%	69.20%
PP 30-10 vs. HT best	z value		1.63	3.69	3.62	2.81
	p-value		< 0.2	< 0.001	< 0.001	< 0.01

Table 3.7: Comparison of the results of histogram intersection, Padua points ($\sigma^2 = 0.0006$ for $n = 30$ and $\sigma^2 = 0.0061$ for $n = 10$), DCT and HT (trained on the ALOI dataset for the worst case) in the (H, S) space on the MPEG-7 dataset. The signed-rank test between PP 30-10 and HT best shows that different levels of statistical significance depending on the measured considered.

1. Project each pixel of the image on to the domain (r, g).

2. Build the chromaticity histogram, represented by the vector h.

3. Build the matrix A with the Chebyshev polynomial basis of proper degree, as required by the least-squares method.

4. Obtain the vector $\mathbf{x}$ that minimises the ℓ^p norm of the residuals

$$\|\varepsilon\|_p = \|A\mathbf{x} - h\|_p, \tag{3.14}$$

5. Use the vector $\mathbf{x}$ for the indexing, as described for the vector $\mathbf{C}$ in section 3.2.

In step 1 we mention only the (r, g) space because, as we have seen in section 3.5, the Padua points achieve better and more consistent performance when approximating the

chromaticity histogram. Therefore, from this point onwards all the experiments will be carried out in the (r, g) space.

When $p = 2$, the norm minimisation is a simple least-squares approximation, however for a generic p the implementation is not as straightforward. For the ℓ^∞ norm, we set up a linear programming problem (see, e.g., Cheney and Kincaid, 2004), for the ℓ^p norm, with $1 < p < \infty$ we implemented the Iterative Re-weighted Least Squares (IRLS) algorithm (e.g., Moon and Stirling, 2000).

Finally, the Gaussian smoothing might also influence the results. To test this, we again build a histogram, but the height of the bins is not just the number of projected pixels within a certain interval: in this case, similarly to the Padua point method, we use a Gaussian to weight the pixels according to their distance from the bin (as described in section 3.2). After obtaining a smoothed histogram, we can get the coefficients of an approximating polynomial through the least-squares method and then compare the retrieval performance to the one obtained without Gaussian smoothing.

3.6.1 Experiments and results

The same experimental procedure described in section 3.5 can be used to answer the questions raised in the previous section: we adapted it in order to approximate the histogram with the least-squares minimisation, the ℓ^p norm minimisation (IRLS), the ℓ^∞ norm minimisation (linear programming), and finally we tested the Gaussian smoothing on the least-squares approximation. The degree of the approximating polynomial was always the same in all the methods, and it has been set to $n = 10$, as in the PP 10 method described in section 3.4. Here we performed our experiments on the COIL and ALOI datasets. Nevertheless, none of the methods proposed reaches results comparable to those of the Padua points. The minimisation of different norms of the residuals shows some decline of the performance when p gets larger, although this decline is not strictly monotonic, with some oscillations and some difference between the datasets.

As we can see in figure 3.7 and figure 3.8 on the next page, both $P(0.5)$ and P_{N_r} in the ALOI dataset decrease almost monotonically; however, this is not true for the COIL dataset (figure 3.9 and figure 3.10), where both measures show some oscillations while decreasing, and then grow again for the ℓ^∞ norm. Since for $p = 2$, that is with the least-squares, we have the best results, we investigated whether for $p = 1$ the results might be even better. For this purpose, as suggested by Schroeder et al. (1991), we used the IRLS method with $p = 1.001$, because in general the solution of the minimisation of the ℓ^1 norm is not unique. Still, even in this case the results are worse than those obtained with the least-squares.

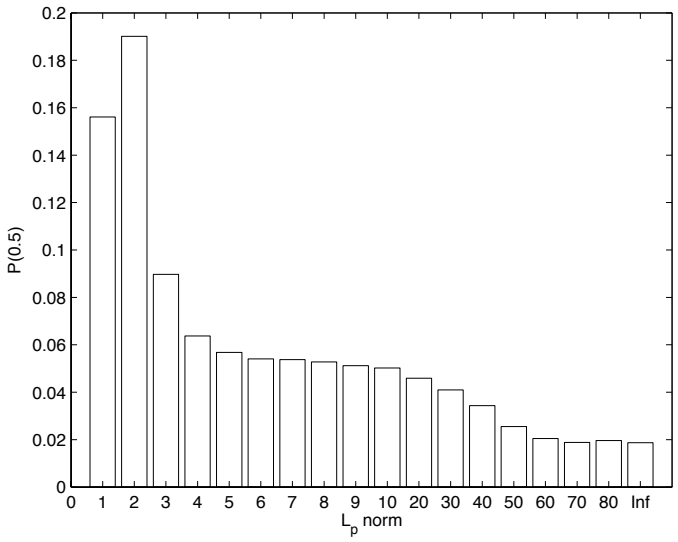


Figure 3.7: Precision at 0.5 recall on the ALOI dataset with different p norms (on the x axis).

Even if the least-squares approach has the best performance out of all the other ℓ^p norms, we can see in table 3.8 that the Padua points still do better. Therefore, we attempted to match the Padua point results by applying the Gaussian smoothing to the least-squares, without a significant change in their retrieval performance.

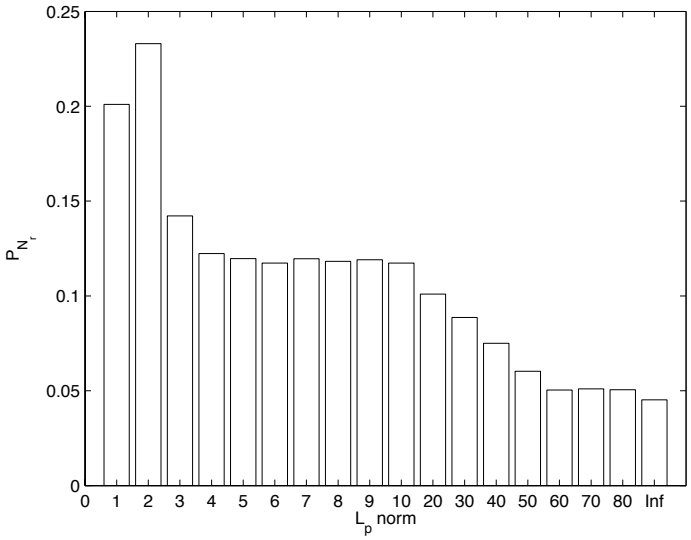


Figure 3.8: Precision after N_r documents retrieved on the ALOI dataset with different p norms (on the x axis).

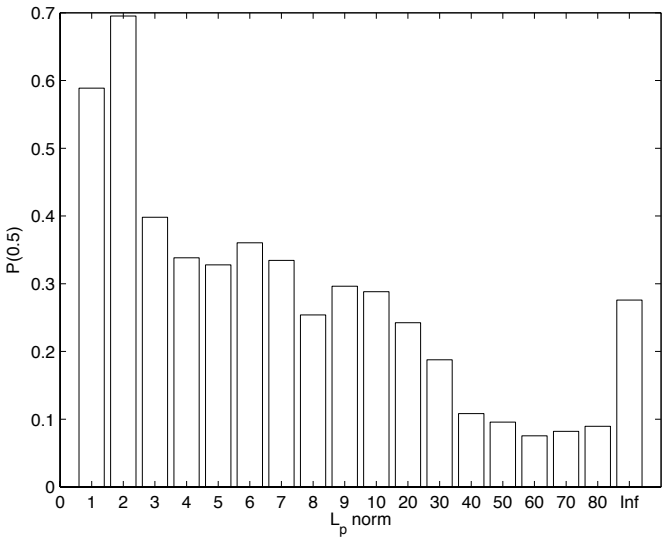


Figure 3.9: Precision at 0.5 recall on the COIL dataset with different p norms (on the x axis).

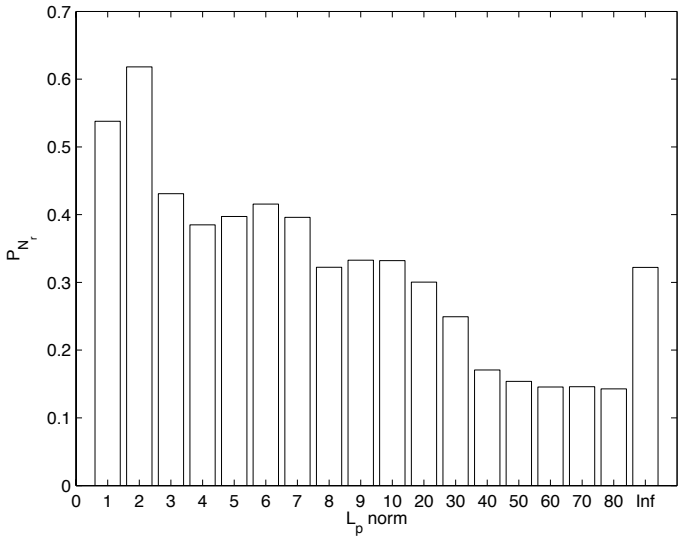


Figure 3.10: Precision after N_r documents retrieved on the COIL dataset with different p norms (on the x axis).

	Padua points		Least-squares	
	$P(0.5)$	P_{N_r}	$P(0.5)$	P_{N_r}
COIL	73.91%	65.69%	69.53%	61.82%
ALOI	27.98%	30.70%	19.06%	23.30%

Table 3.8: Comparison of the retrieval results of the least-squares approximation and the Padua point method.

3.7 Conclusion

In this chapter we have presented an image indexing and retrieval method based on the Padua point interpolation. Our method performs well, especially when working in the chromaticity space. Further, it achieves good performance without any training and obtains similar results to those of the "trained" HT based on PCA. We notice that in the ALOI dataset the retrieval results of the Padua point method are best-in-class. In the COIL and MPEG-7 datasets the results are very close to those of the HT, with a slight advantage for one of the methods depending on how we measure the performance. On

the other hand, we consistently outperform the DCT on a similar number of coefficients. This is significant, as our method requires no training and so is general. In contrast, it is possible that the best HT basis chosen by a dataset may not scale to other image sets.

Because the Padua point interpolation, in some sense, bounds the maximum deviation from the original function, we tried to approximate histograms by minimising different ℓ^p norms of the residuals, including ℓ^∞. However, we found that the least-squares approximation was the one with the best retrieval results, indicating that the ℓ^∞ norm minimisation is not, on its own, the feature of the Padua points that facilitates a good performance. Moreover, the retrieval results of the least-squares were fairly poor, if compared to those of the Padua points. A major contribution is given by the Padua points themselves, because they are responsible for the near-optimality of this kind of interpolation, in the same way as the Chebyshev nodes are near-optimal in one dimension. Without Padua points it is very hard, if not impossible, to obtain the coefficients of a nearly-optimal approximating polynomial.

We would like at this point to remind the reader that our work was not about parameter tuning, so it is possible that further tweaking on our method would obtain best results on all datasets. Instead, we explored the possibility of introducing a new kind of polynomial interpolation in image indexing, and in this sense we think that the method we proposed delivers very interesting results.

Chapter 4

Pseudo-random walk and pseudo-Brownian path generator

Brownian motion (whose properties were described in section 2.2.2 on page 11) was initially observed in particles floating on a fluid. However, it also presents itself in nature in other contexts. For our purposes, we focus on its relation to human vision. It is known that the involuntary micro-saccades that the human eye almost constantly performs follow a Brownian motion scheme (Engbert and Kliegl, 2004). Moreover, this is true also when the human eye follows a moving target. If the target moves quickly and often changes direction, our eye tries to anticipate its position with predictive saccades that follow a Brownian motion distribution (Shelhamer, 2005). Finally, the distribution of centroid cells in the region V4 of our visual cortex (see Zeki, 1993, pp. 151–155) strongly resembles a Brownian motion trail, as noted in (Marini and Rizzi, 2000) (see figure 2.4 on page 17). These observations suggest an important relation between this random process and human vision. Brownian motion has therefore been applied, with some success, in image processing applications where modelling human vision is important, for example the retinex algorithm by Marini and Rizzi (2000). Their approach is to carry out the retinex computations along Brownian paths, and this has very in-

teresting results. Unfortunately, the computational complexity for generating the paths makes it highly impractical to use. It would be desirable to overcome this limit, since retinex has several applications in digital imaging (see section 2.5 on page 18).

Here we consider that Brownian motion proved to be useful for image processing tasks, but has some practical disadvantages. First, a Brownian path across the pixels of an image could be relatively fast and easy to implement, but it cannot guarantee anything about how the pixels are visited: some pixels may be visited more often than others, some may be skipped completely. Second, Marini and Rizzi's approach of generating several Brownian paths per pixel has a high computational complexity: 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 their work the speed of computers has greatly increased, so has the resolution of images. For these reasons, starting from previous work on random Hamiltonian paths (Fredembach and Finlayson, 2005, 2008) we develop an algorithm that generates a random walk on the pixels of an image with a constraint: we guarantee the minimum and the average number of times that each pixel is visited. A further and relatively simple development of this algorithm transforms it in such a way that the paths it generates are pseudo-Brownian, yet retaining the same constraint on the number of visits per pixel. We are also interested in verifying how similar these pseudo-random paths are to the actual random processes, and we do so by comparing the properties of random walk and Brownian motion we described in chapter 2 with the empirical measurements on the pseudo-random paths we generate.

This chapter is structured as follows. In section 4.1 we briefly recall the properties of random walk and Brownian motion that we described in detail in chapter 2. We also specify two types of retinex to which we will refer later, Marini and Rizzi's Brownian motion retinex and McCann's multi-scale retinex. In section 4.2 we detail our algorithm for generating paths and in section 4.3 we show how to extend it to obtain pseudo-

Brownian motion. We also analyse the similarity of these paths to the actual stochastic processes. In section 4.4 we describe the application of our algorithm to retinex, and in section 4.5 we provide some examples that show the advantages of our approach to retinex over previous methods.

4.1 Background

In chapter 2 we described various properties of random walk and Brownian motion and now we want to use them as a "benchmark" that verifies whether or not the paths generated with our approach are Brownian. Unfortunately, we can already tell that this is not the case. The reason for this is that our algorithm needs to have a memory of its previous states, thus it violates the Markov property, i.e., it cannot be time independent. However, we can try to determine whether the paths our algorithm generates resemble the actual random processes.

We are interested in two properties of the random walk. First, the expected distance from the origin after n steps. We showed that this value is proportional to $\sqrt{n}$ regardless of the number of dimension of the random walk. Second, the probability $P(n, d)$ of the walker being at a distance d from the origin after n steps. We showed how to obtain a formula for the one-dimensional case, which is

$$P(n, d) = \frac{1}{2^{n-1}} \binom{n}{\frac{d+n}{2}}. \tag{2.6}$$

However, to the best of our knowledge, in the literature there is no formulation equivalent to that in equation (2.6) for the two-dimensional random walk. Therefore, we took a computational approach and devised a simple and efficient algorithm to calculate $P(n, d)$ for the two-dimensional random walk. We discarded the obvious brute-force approach of computing all possible paths of length n and counting those that end at distance d from the origin, since that becomes quickly impractical as n grows (in Matlab

$n = 10$ takes already too long). We start instead with a simple observation on the random walk. The probability $w^n_{i,j}$ that at the nth step the walker is in position i, j depends on the probabilities that at step $n-1$ the walker is in any of the neighbouring positions, times the probability of a step to i, j:

$$w^n_{i,j} = \frac{1}{4}w^{n-1}_{i-1,j} + \frac{1}{4}w^{n-1}_{i+1,j} + \frac{1}{4}w^{n-1}_{i,j-1} + \frac{1}{4}w^{n-1}_{i,j+1}. \tag{4.1}$$

In this equation one can recognise a discrete convolution by a mask

$$M = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{bmatrix}. \tag{4.2}$$

This can be seen more clearly in the following example. Let us represent a subset of a lattice with a 5×5 matrix, whose coefficients represent the probability of the walker being in that position at that time step. Let us call W^0 the matrix representing the initial step $t = 0$, and let its indices i, j range from -2 to 2, assuming that the initial position of the walker is the central coefficient $w^0_{0,0}$.

$$W^0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad W^1 = \begin{bmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{bmatrix}, \quad W^2 = \begin{bmatrix} 0 & 0 & \frac{1}{16} & 0 & 0 \\ 0 & \frac{1}{8} & 0 & \frac{1}{8} & 0 \\ \frac{1}{16} & 0 & \frac{1}{4} & 0 & \frac{1}{16} \\ 0 & \frac{1}{8} & 0 & \frac{1}{8} & 0 \\ 0 & 0 & \frac{1}{16} & 0 & 0 \end{bmatrix}. \tag{4.3}$$

In W^0 the walker is at the origin with probability $p = 1$, as it has not taken any step yet. It is clear what happens in W^1: the walker takes one step, and thus the probability of its position at $t = 1$ spreads to the four neighbours of its position at $t = 0$. Looking at W^1, one can see that it looks like W^0 after a convolution with the mask M above;

similarly, W^2 is W^1 after a convolution with M. Thus, in the representation we defined, the convolution with M is exactly one step of the random walk, and the probability distribution of the position of the walker at the time step $t = n$ is given by the recurrence $W^n = W^{n-1} * M$, with a starting point W^0 as in (4.3). The probability $P(n, d)$ is therefore given by $\sum_{h,k} w^n_{h,k}$, with h, k representing all the positions at city-block distance d from the initial position.

As for the properties of the Brownian motion, described in section 2.2.2 on page 11, we can immediately state that two out of three of them cannot apply to our paths: they are neither continuous nor time independent. The third property, the normality of the increments in each dimension, can be tested in our paths, therefore it can be used to compare them to simulated Brownian paths.

4.1.1 Marini and Rizzi's retinex

In this section we want to delineate briefly the Brownian motion retinex proposed by Marini and Rizzi (2000), because it plays an important role in the development of our work. It is a path-based retinex, where for each pixel a reflectance is estimated using several paths. Along the kth path, an estimate $I'_k(\rho_n)$ is computed in the logarithm of the image similarly to equation (2.10) on page 20. So, if the path crosses the pixels $\rho_1, \ldots, \rho_n$,

$$I'_k(\rho_n) = \sum_{i=1}^{n-1} \delta'(I(\rho_{i+1}) - I(\rho_i)), \tag{4.4}$$

with δ' threshold operator

$$\delta' = \begin{cases} 1 \text{ if } |I(\rho_{i+1}) - I(\rho_i)| > \tau \\ 0 \text{ otherwise.} \end{cases} \tag{4.5}$$

The final estimate is given by the average of the estimates of all the p paths:

$$I'(\rho_n) = \frac{1}{p} \sum_{k=1}^{p} I'_k(\rho_n). \tag{4.6}$$

A reset operation is also inserted in equation (4.4), thus the effective length of the kth path depends only on where it encounters the pixel with the highest intensity.

The novelty of this retinex implementation is, clearly, the type of paths chosen. For each pixel several paths are generated, originating from a random point and ending to the pixel itself. Two parameters control the number of paths per pixel p and the length of each path. These two features influence the complexity of the algorithm, which is $\mathrm{O}(p\chi N)$, where χ depends on the length of the paths. While Marini and Rizzi say that p is normally small, they do not provide any detail about χ. However, we can infer that its value might be large, since the authors themselves admit that their algorithm is computationally inefficient.

4.1.2 The McCann99 retinex

In this section we focus on the McCann99 retinex (McCann, 1999) and its implementation (Funt et al., 2000, 2004), since we use it as reference to test our results in section 4.5. This version of retinex does not use paths. Rather, it compares each pixel with its eight neighbours, updating the estimate after each comparison. In the following formulas, the letters O, I and N in superscript stand for old value, intermediate value and new value respectively, whereas the lack of superscripts indicates the original pixel value (note that we are again considering the logarithm of an image). Let us assume we are considering the pixel of coordinates i, j.

$$I^{\mathrm{I}}_{h,k} = I^{\mathrm{O}}_{h,k} + I_{i,j} - I_{h,k}, \tag{4.7}$$

where h, k iterate through the neighbours of i, j. I^{I} is computed over the whole image, then clipped to 0 (reset step), and finally averaged with the old estimate, that is,

$$I^{\mathrm{N}} = \frac{1}{2}\left(\min\{I^{\mathrm{I}}, 0\} + I^{\mathrm{O}}\right). \tag{4.8}$$

The value I^{N} computed here will become I^{O} in the next iteration. These two equations compose one iteration of McCann99 retinex. A peculiarity of this algorithm is the multi-scale approach. In other words, the original image is shrunk and the computations are performed over different scales. The algorithm starts from a thumbnail of the image (assuming $I^{\mathrm{O}} = 0$ everywhere). After iterating a given number of times, the last I^{N} is upscaled and becomes the initial I^{O} for the larger scale, and so on until the full-size image.

The complexity of this algorithm is linear, although it depends also on a user-provided parameter m that sets the number of iterations. Let us consider an image having N pixels, and start from the computations on its full-scale version. Equation (4.7) takes a constant time, and is repeated for all the eight neighbours of every given pixel, thus the total number of operations is about $8N$. Also equation (4.8) takes a constant time for each pixel. Both operations are iterated m times, thus we can approximate the running time with $\mathrm{O}(mN)$. This is valid for each scale, which has a fraction $q < 1$ of the pixels compared to its immediately larger scale. Therefore the total number of operations is

$$\mathrm{O}\left(\sum_{s=1}^{\infty} q^s mN\right) = \mathrm{O}\left(mN \sum_{s=1}^{\infty} q^s\right) = \mathrm{O}(mN), \tag{4.9}$$

assuming that we are infinitely scaling down the image. The infinite sum of q^s is a geometric series that converges to a constant value, thus it can be removed from the $\mathrm{O}(\,\cdot\,)$ notation (here we are considering an infinite number of scales only to show that this does not influence the computational complexity of the algorithm, however in reality one has clearly to scale down the image a finite number of times). The number m of it-

erations is normally low, we can safely say that normally $m < 30$, thus we can consider it constant and write the complexity of the McCann99 algorithm as $\mathrm{O}(N)$.

4.2 Random walk from Prim's algorithm

In section 2.3 on page 12 we described an algorithm to solve the TSP (travelling salesman problem) over grid graphs. The merit of this algorithm is mainly its efficiency, since in general this problem is NP-complete. This might appear a rather complicated approach to random paths, so one may wonder why we do not simply simulate a random walk. It is a known result that up to two dimensions an infinite random walk visits the whole $\mathbb{Z}^2$ lattice with probability 1 (Rudnick and Gaspari, 2004). Intuitively we can say that if the walk is long enough it will visit every pixel of an image the same number of times. However, this is impractical: the random walk we can generate has necessarily a limited length, thus some pixels may be visited significantly more often than others and some may not be visited at all. Our purpose here is to avoid such unbalance: we wish to guarantee we visit every pixel a certain lower bound of times, but also limit the final length of the path. One possibility would be to generate many random Hamiltonian paths. Unfortunately, a random Hamiltonian path does not have a very "random" structure. At the beginning it will be quite "random" but then it will have fewer and fewer pixels to choose from. Moreover, we expect it not to abide to the properties described in section 2.2.1 on page 8. 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 in section 2.3 on page 12. With our method we can set a lower bound k to the number of times that the path will visit each pixel; moreover, we can prove that on average the path will visit each pixel $m = 2k$ times.

We start with an intuitive consideration. What happens if in the TSP algorithm

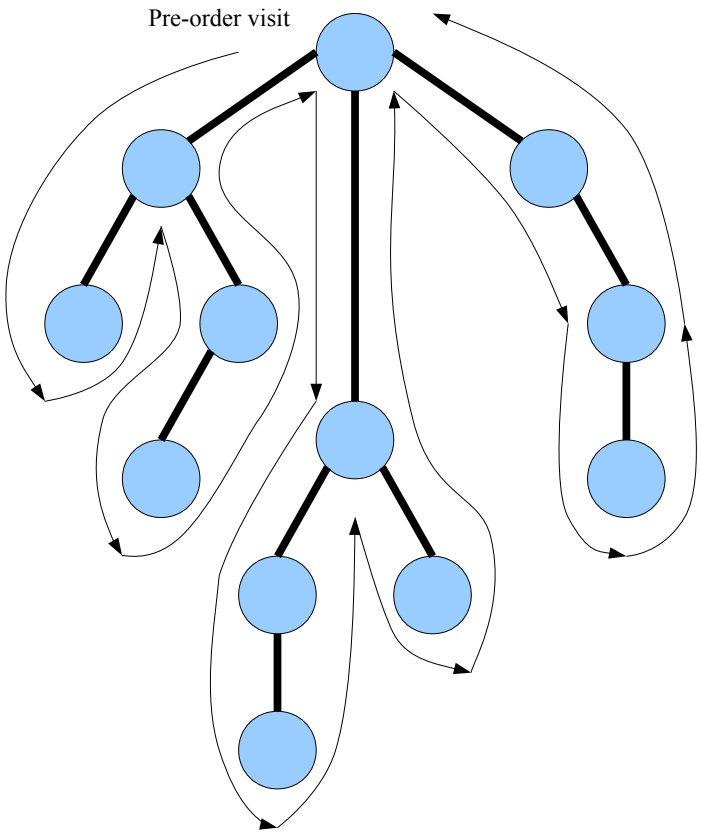


Figure 4.1: Visit to a tree. From this figure it is possible to see that the visit "touches" each node $h + 1$ times, where h is the number of children of that node.

described in chapter 2 we visit the spanning tree of an image directly, without any subsampling and upsampling step as in figure 2.3c and figure 2.3d on page 13? A pre-order visit of the tree would go through each pixel more than once. Specifically, if a node of that tree has h children, the visit will cross the corresponding pixel $h + 1$ times (figure 4.1). 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". 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", between any pair of vertices (Gould, 1988). An element x that appears z times in a multiset X is said to have multiplicity z. This is often indicated with the notation $\mathbf{1}_X(x) = z$, called multiplicity function. 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 (e.g., Cormen et al., 2001) 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. Here is how our algorithm works (an example of its application is shown in figure 4.2 and figure 4.3). We start with the multisets $S = \{r\}$ (r is a vertex of G, i.e., $r \in V$, and is chosen arbitrarily), $Q = \emptyset$ 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 $\mathbf{1}_S(v) < k$ (this step requires $\mathrm{O}(\log N)$ operations). If the graph allows loops, i.e., edges whose start and end points are the same vertex, one might want to add the further constraint that $u \neq v$.

3. Select randomly an edge $(x, y) \in Q$ so that $y \notin S$ or $y \in S$ with $\mathbf{1}_S(y) < k$; then $T = T \cup \{(x, y)\}$, $S = S \cup \{y\}$ and $Q = Q \setminus \{(x, y)\}$ (this step takes $\mathrm{O}(\log N)$ operations). If in Q there is no edge (x, y) satisfying the requirements for y, we discard all the content of Q.

4. *Until* $Q = \emptyset$ (iterate $\mathrm{O}(N)$ times).

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

Proof. In step 3 our algorithm forces the number of edges entering each vertex to be k. Let us consider a vertex x: once the algorithm reaches x, it will add to the set Q the edges to all its neighbours. This means that once the neighbours are visited as well, further edges leading to x will be added to Q. This repeats until x has k entering edges. The root r has $k - 1$ entering edges because it starts already in the multiset S. From this property we can compute the total number of edges in G': if the initial graph G has N vertices, G' has $|T| = (N - 1)k + k - 1 = Nk - 1$ edges. □

Starting from the root, we can at this point perform a pre-order visit to the G'. A pre-order visit on a tree considers each node before its offspring (Cormen et al., 2001), and the sequence it yields is the path we seek. As a consequence of lemma 4.1, the following properties hold:

Property 4.1. *The final length of the path in an image of N pixels is $2kN - 1$.*

Proof. A pre-order visit of the tree obtained with our algorithm yields the path with the desired properties, and crosses each edge twice, once on its way from root to leaves

and once on its way back. This means that $2kN - 2$ edges appear in the path, and thus $2kN - 1$ vertices. □

Property 4.2. *On average the path visits each pixel $2k$ times when N is large.*

Proof. Property 4.1 says that the path is $2kN-1$ pixels long. Thus, $(2kN-1)/N \approx 2k$ with N large. □

Property 4.3. *The path visits each pixel at least k times.*

Proof. From lemma 4.1 we know that, except for the root, every vertex of the graph has k entering edges. This implies that the visit has to enter each pixel at least k times. In fact, it visits exactly k times only vertices that do not have any outgoing edge and thus are crossed only on the way from root to leaves. Finally, even if the root has only $k - 1$ entering edges, by definition it has outgoing edges and therefore must be visited more than $k - 1$ times. □

At this point we would like to provide evidence of how the paths generated with the algorithm described above maintain the same statistical properties as the random walk. In figure 4.4a we compare the average distance from the origin (or drift from the origin) after 10^4 of our paths with that of random Hamiltonian paths. As reference, we show also the expected drift, as from equation (2.2). Note that the expected drift of the random walk is actually proportional to the square root of the number of steps (Lawler and Coyle, 2000), and that is exactly what figure 4.4a suggests about our pseudo-random paths. Random Hamiltonian paths instead show some oscillations, probably due to the strong constraint of visiting each pixel exactly once. In figure 4.4a our paths is set to $k = 16$: if we set a smaller number, say $k = 1$, the curve of its drift would resemble that of the random Hamiltonian path. In figure 4.4b we show the drift of some regular paths, namely a raster and a Hilbert curve. We show a smaller number of steps (1000) in order to give a better idea of the regular structure of these two curves. After testing

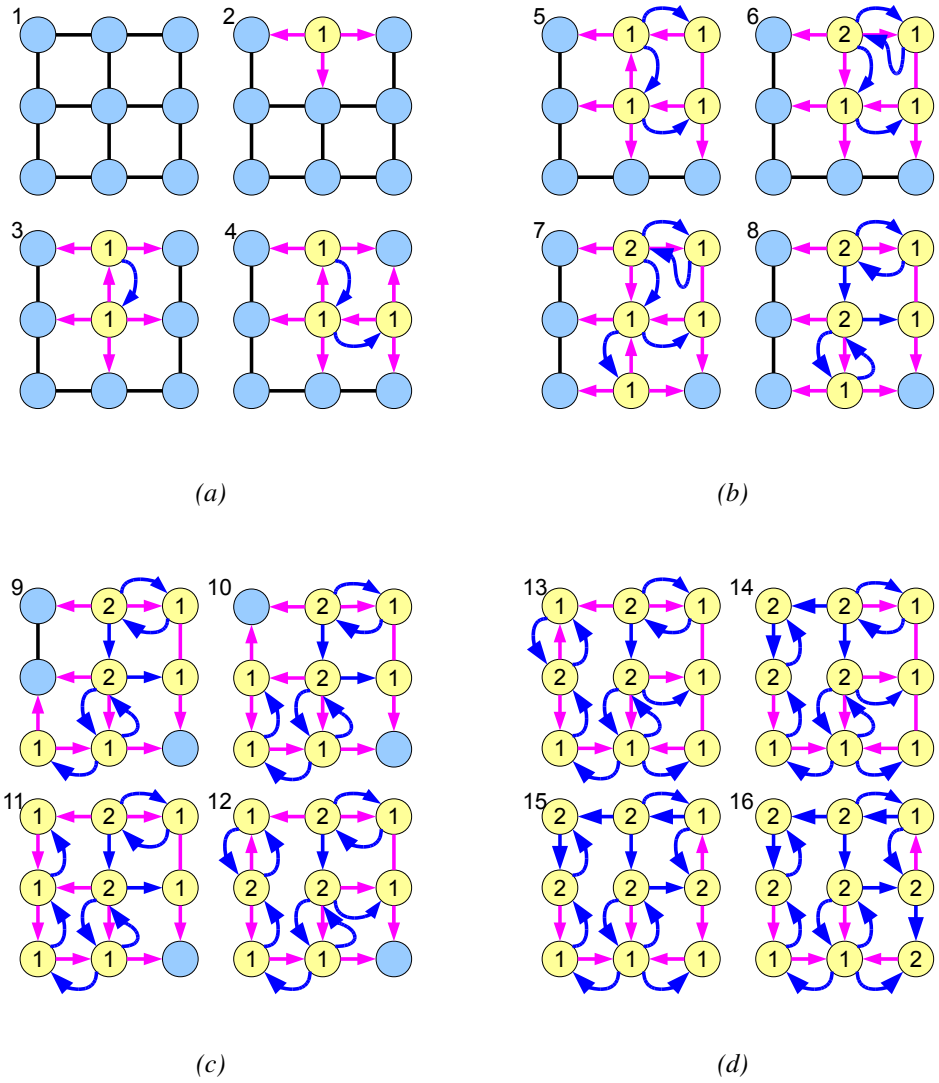
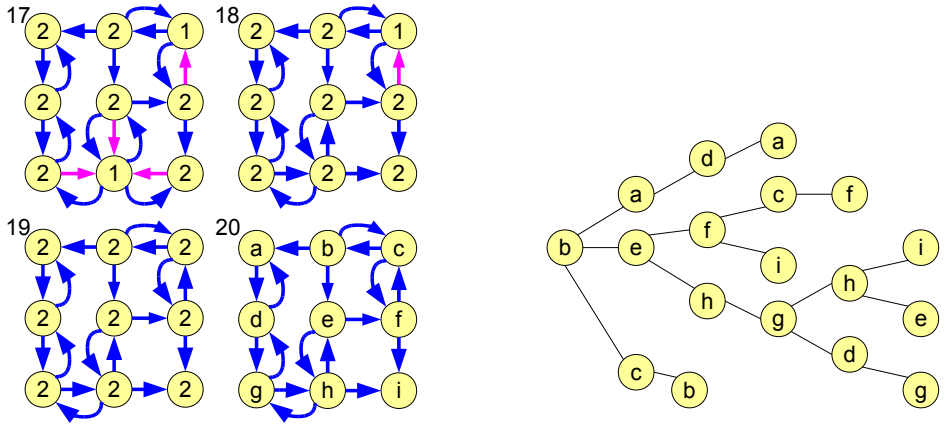


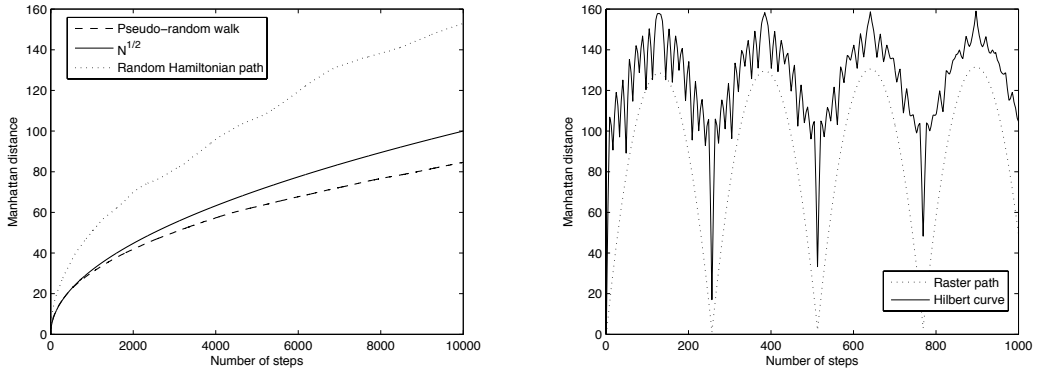
Figure 4.2: Application of our algorithm to a 3×3 grid graph, with parameter $k = 2$. Yellow vertices are in S with multiplicity given by the inscribed number, blue edges are in T and magenta edges are in Q. Starting from (a), in step 2 our algorithm randomly selects a node that is going to be the root r of the tree. Thus, all the edges that connect the vertices in S with the others go into Q (or become magenta). In step 3, it picks randomly an edge from Q, adds the edge to T (or it paints it blue) and adds new edges to Q. The algorithm continues this way, always picking a random edge from Q and adding it to T. Note in (b), step 6, that the first vertex reaches multiplicity 2 in S, and all its edges become outgoing only. In (c), step 12, we see that a direct connection between two vertices disappears, since both have already multiplicity 2.



(a) *(b)*

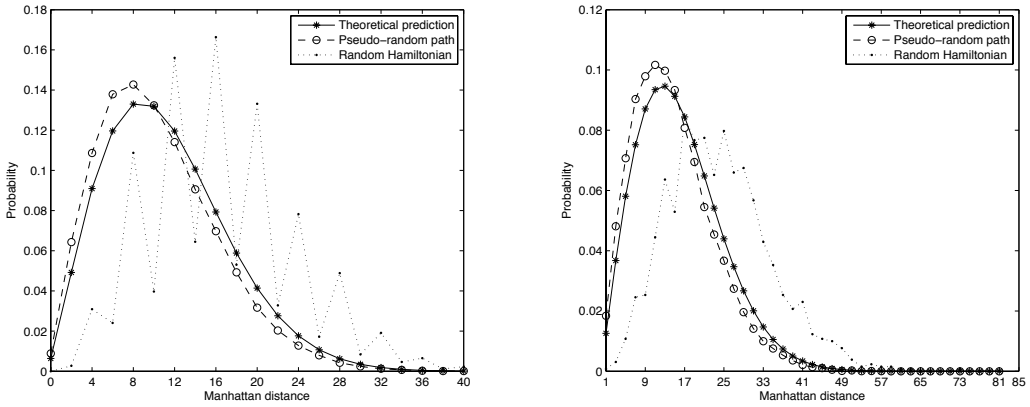
Figure 4.3: In (a) the algorithm keeps proceeding until all vertices are in S twice and thus Q is empty. In step 20 we are labelling the vertices with letters from a to i, so that we can see how a spanning multigraph is built. For the sake of clarity, we unroll this multigraph and show it as a tree with repeated vertices in (b). Even though vertices are repeated, step 20 in (a) and this tree have the same number of edges. Also, in the tree there is no need to stress the directionality of the edges, since this comes naturally in the parent-child node relation. To this tree we apply a visit such as that in figure 4.1 to obtain the final path.

the average drift of our paths, we want to see what happens to the probability $P(n, d)$ of being at a distance d after n steps. As it is possible to see in equation (4.3), for any fixed $d < n$, the value of $P(n, d)$ oscillates between 0 and a positive number as n grows, since it is impossible for the walker to be at an odd distance from the origin after an even number of steps. The same holds for the empirical $\tilde{P}(n, d)$. The plots in figure 4.5 are actually samplings at the non-zero probability distances, and are shown as continuous curves exclusively for the sake of clarity. We can see that even if $\tilde{P}(n, d)$ measured on our paths is slightly shifted from $P(n, d)$, the shape of the two curves is almost identical. On the contrary, $\tilde{P}(n, d)$ of random Hamiltonian paths oscillates heavily and deviates substantially from $P(n, d)$. Once again, we took this measurement with $k = 16$, and we expect that small values of k will result in measurements similar to those of random Hamiltonian paths.



(a) *(b)*

Figure 4.4: Average drifts from the origin of different type of paths. Our random path in (a) visits each pixel on average 32 times (i.e., $k = 16$). The drifts of raster paths and Hilbert curves (b) are very different from those of random walks. The raster paths we considered scan the image horizontally, changing direction every time the boundary is reached (this way, each single step is only one-pixel long).

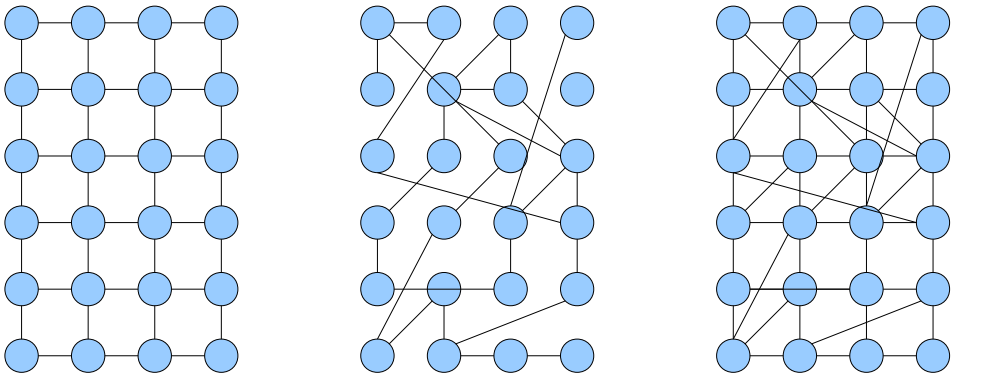


(a) $n = 100$, $d = 0, \ldots, 40$ *(b)* $n = 201$, $d = 1, \ldots, 81$

Figure 4.5: Two instances of the probability $P(n, d)$ and the measured $\tilde{P}(n, d)$, with even (a) and odd (b) numbers of steps and distances, shown as continuous curves for clarity. We set $k = 16$ for our random path.

4.3 From random walk to Brownian motion

We have shown that the algorithm we described above generates paths that share features with the random walk on a lattice. However, as we explained in section 2.2.2 on page 11, one of the features of the Brownian motion is that the displacement is random. In each dimension, it follows a normal distribution with zero mean and standard deviation equal to the considered time step. Our random path generator adapts quite naturally to Brownian motion, without substantial computational overhead and with the same $\mathrm{O}(N \log N)$ complexity. Unlike TSP algorithms, we do not require the graph to be a grid: it is enough if it is connected. The grid structure is necessary for meeting the constraint of visiting each pixel exactly once, but when we relax this constraint, the grid is no longer necessary. 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 (Marsaglia and Tsang, 2000). More in detail, for the grid graph (figure 4.6a) we substitute random edges (figure 4.6b), which connect pixels that are not neighbours in the image. This way, we obtain a random multigraph whose visit delivers a pseudo-Brownian path. Its displacements are more similar to those of the Brownian motion, as we can see in the normal probability plot in figure 4.7a. Unfortunately, in this case lemma 4.1 and its corollaries do not hold any more. To obtain the Brownian motion, we are discarding the grid structure of the image and the random edges we introduce may or may not include all pixels the right number of times, since the new graph might not be connected, that is, some vertices may not be reachable from other vertices of the graph (in fact, this event seems to be very rare. From our tests it would seem that for $k \geq 8$ in general the paths still respect the constraint we give). In order to guarantee the validity of lemma 4.1 and its corollaries, we have to keep the original grid structure of the graph, and subsequently add the random edges, such as in figure 4.6c. This way, we know that the graph is connected and that is enough for our algorithm to meet all the constraints we need. On the other hand,

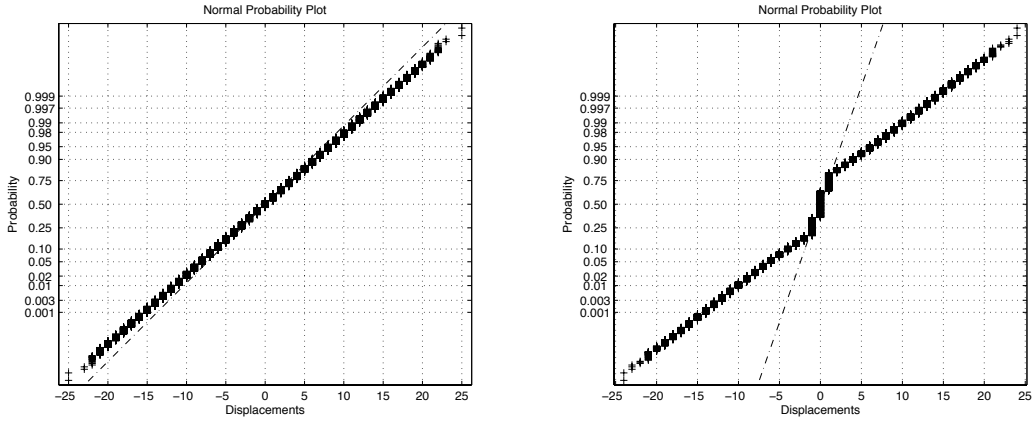


(a) Grid graph *(b) Graph with random edges* *(c) Combination of (a) and (b)*

Figure 4.6: Grid graphs (a) are a natural way of representing images: each vertex corresponds to a pixel, and edges connect pixels that are neighbours. Our algorithm on such a graph generates a pseudo-random walk. To obtain pseudo-Brownian motion, we discard the grid structure and generate random edges (b) having normally distributed length. This approach does not guarantee that the graph is connected, i.e., there may exist two vertices that are not connected by a path. In order to guarantee this constraint, we superimpose (a) with (b) and we obtain (c).

in this case we lose the normality of the displacements, as we can see in figure 4.7b. Specifically, that normal probability plot shows that the data have longer tails than a normal distribution having the same standard deviation. However, we can explain this by looking at figure 4.8b, which shows an interesting fact. Here, rather than estimating the standard deviation from the data, we use the same as in the random number generator. This way, the tails match perfectly and the "error" between the histogram and the normal distribution occurs only where the displacements due to the grid structure are, that is in -1, 0 and 1. In practice, this is as if we were superimposing a random walk to the pseudo-Brownian motion. In our retinex implementation, which we explain in the next section, we will use our path generator on grid graphs with added random edges as in figure 4.6c. These extra edges are obtained by generating a normally distributed random displacement (with $\mu = 0$ and $\sigma^2 = 5$) for the x and one for the y direction

using the ziggurat algorithm.

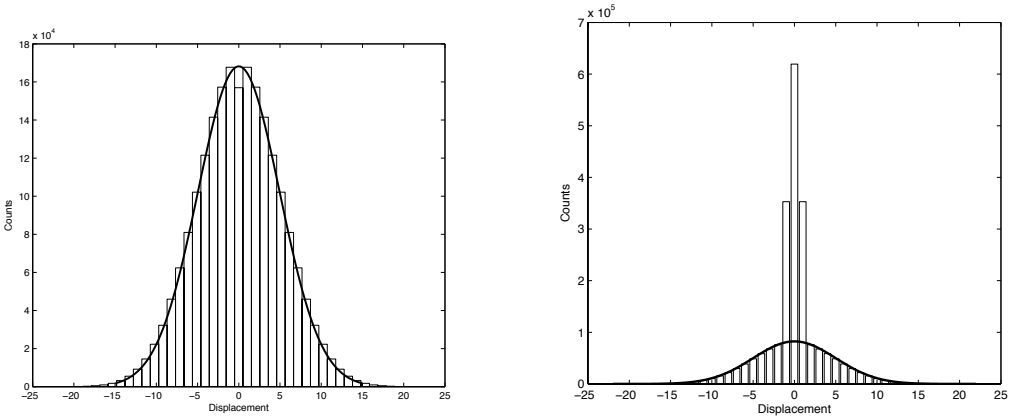


(a) Displacements without grid structure. *(b) Displacements with grid structure.*

Figure 4.7: Normal probability plots of the displacements (a) without and (b) with grid structure in the graph. The dashed line represents a normal distribution, and the black crosses the measured displacements. The deviation from the line in (b) shows that the tails of the measured data are longer than those of a normal distribution with the same mean and standard deviation as the data. These plots are only for the displacements in the y direction, as the plots for x are almost identical.

4.4 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. Although the theoretical space required is $\mathrm{O}(N)$, 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 as the one described in section 4.1.2. 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. This way, we keep the memory usage low, and at the same time the complexity does not increase. In practice, on a path ρ of pixels



(a) Displacements without grid structure. *(b) Displacements with grid structure.*

Figure 4.8: Normal distribution overlapped to the displacement histogram. If we do not force the grid structure in the graph (a), the histogram and the normal distribution match almost perfectly, as shown already in figure 4.7a. If we force the grid structure instead (b), the histogram has a peak on steps of length -1, 0 and 1, as we would expect. The normal distribution in (b) has the same standard deviation as the random steps introduced by the algorithm. These plots are only for the displacements in the y direction, as the plots for x are almost identical.

$\rho_1, \ldots, \rho_n$, using the same notation as in section 4.1.2, we have

$$I^{\mathrm{I}}(\rho_{i+1}) = I^{\mathrm{N}}(\rho_i) + I(\rho_{i+1}) - I(\rho_i) \tag{4.10}$$

$$I^{\mathrm{N}}(\rho_{i+1}) = \frac{1}{2}\left(I^{\mathrm{I}}(\rho_{i+1}) + I^{\mathrm{O}}(\rho_{i+1})\right). \tag{4.11}$$

Equations (4.10) and (4.11) represent a generic iteration of retinex, after which $I^{\mathrm{N}}(\rho_i)$ becomes $I^{\mathrm{O}}(\rho_i)$. In the first iteration, (4.10) assumes that $I^{\mathrm{I}}(\rho_1) = 0$, that is to say the pixel is white. In (4.11) the value of $I^{\mathrm{O}}(\rho_{i+1})$ 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, $I^{\mathrm{O}}(\,\cdot\,)$ is initialised to 0, just like in the McCann99 algorithm.

Let us now detail the computational complexity of our retinex implementation. Let

s be the number of scales we are dividing a certain input image. Each scale has $1/4$ of the pixels of its larger version. If n is the number of pixels of a certain scale m, then retinex is doing $\mathrm{O}(n)$ pixel comparisons and our path generator requires $\mathrm{O}(n \log n)$. Since $n = N \times 1/4^m$ (where m is the current scale), omitting the $\mathrm{O}(\cdot)$ notation for a moment we can write the overall complexity as

$$\sum_{m=0}^{s} \frac{1}{4^m} N \log \left(\frac{1}{4^m} N \right) + \sum_{m=0}^{s} \frac{1}{4^m} N, \tag{4.12}$$

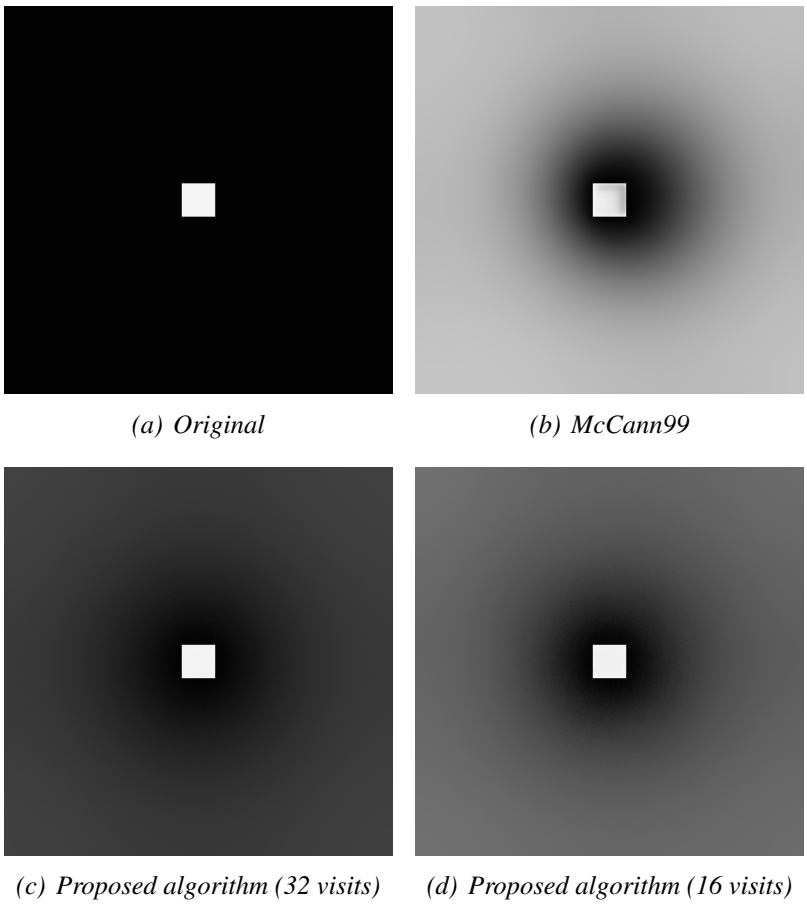
where the first sum is number of operations required by the path generation, and the second sum the number of operations required by retinex. We can expand this in

$$\begin{aligned} &= N \sum_{m=0}^{s} \frac{1}{4^m} (\log N - m \log 4) + \sum_{m=0}^{s} \frac{1}{4^m} N = \\ &= N \log N \sum_{m=0}^{s} \frac{1}{4^m} - N \left(\log 4 \sum_{m=0}^{s} \frac{m}{4^m} - \sum_{m=0}^{s} \frac{1}{4^m} \right) = CN \log N - C'N \end{aligned} \tag{4.13}$$

for some positive constants C and C'. The complexity of our retinex implementation is therefore $\mathrm{O}(N \log N)$.

4.5 Results

An exhaustive comparison of our retinex with other implementations would be beyond the scope of our work. Moreover, in the work by Marini and Rizzi (2000) a number of comparisons and arguments are presented, which all support the goodness of the Brownian path approach. For these reasons, we limit our direct comparison to the McCann99 retinex (McCann, 1999) and its reference implementation in Matlab (Funt et al., 2004). First of all we would like to consider the computational complexity. The McCann99 algorithm has been embedded in digital cameras because of its efficiency, since its memory requirements are minimal and its time complexity is $\mathrm{O}(N)$ on the number



(a) Original *(b) McCann99*

(c) Proposed algorithm (32 visits) *(d) Proposed algorithm (16 visits)*

Figure 4.9: Image from Funt et al. (2004). With a similar number of comparisons per pixel per scale, 32 with McCann99 (b), and 32 on average with our approach (c), our method delivers a much more plausible resulting image. With an average of 16 visits per pixel (d), we can see that the halo artefact around the white square is not biased in any particular direction, whereas even with 32 visits per pixel in McCann99 the halo is clearly biased towards the top-right corner of the image (b).

of pixels (in table 4.1 we compare the complexity of different retinex implementations). In terms of actual time, this makes some difference. For example, in order to process a 512×768 RGB image with an average of 64 comparisons per pixel McCann99 requires 12.6 s, while our algorithm requires 54.7 s (McCann99 is implemented in Matlab, our algorithm in Matlab and C). Both implementations can be further optimised, so these times should be considered in proportion to each other rather than absolute.

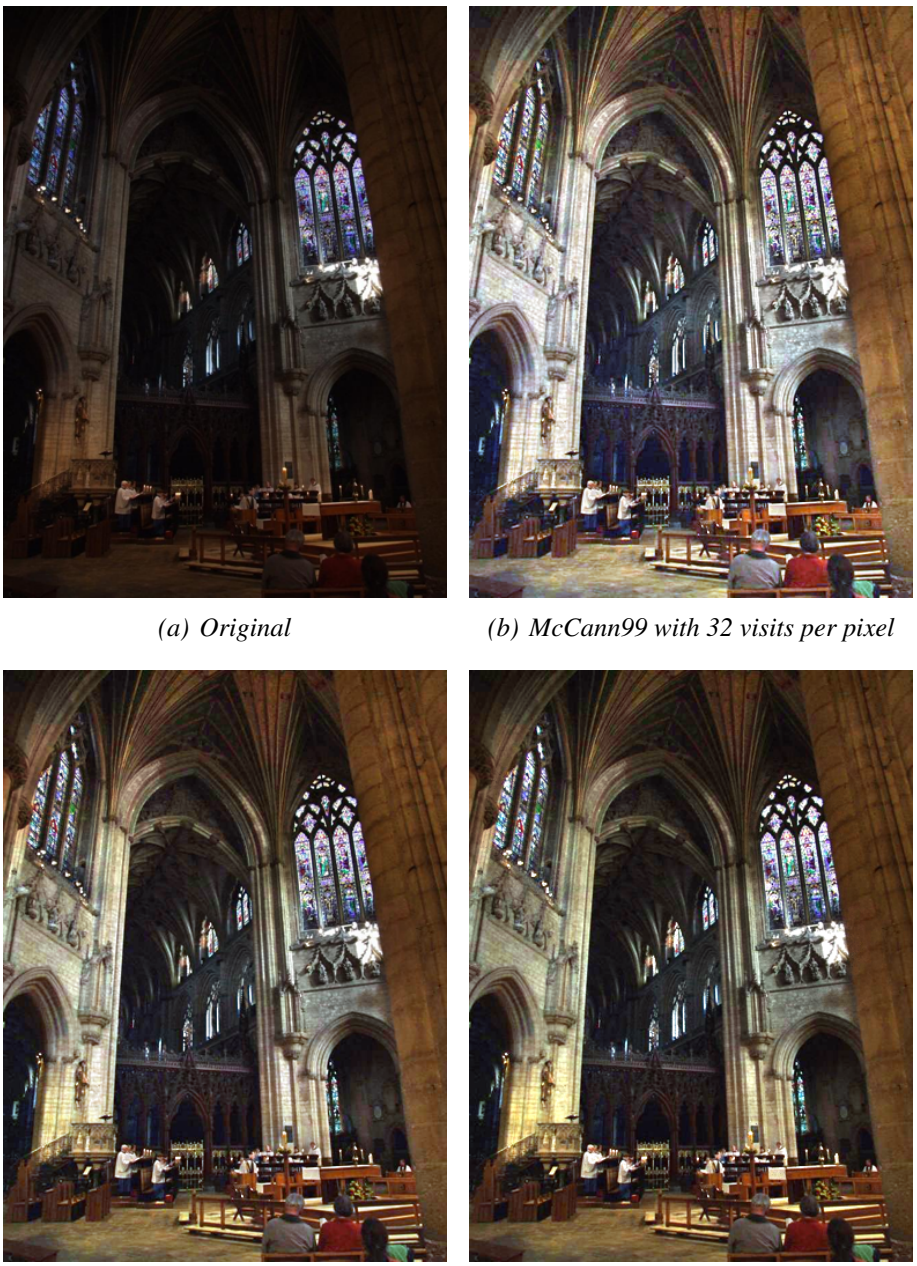
(a) Original *(b) Original, detail*

(c) McCann99 *(d) McCann99, detail*

(e) Proposed algorithm *(f) Proposed algorithm, detail*

Figure 4.10: Image from the Kodak dataset (Kodak, 2004), processed with four McCann99 iterations (32 visits per pixel) and $k = 16$ with our approach (32 visits per pixel on average). Notice the thin frame in (b), which smears into the photo in (d) while remaining "solid" in (f)

Now we will present some images processed with the two algorithms. In McCann99, each iteration involves eight pixel comparisons, therefore we will count those rather than the number of iterations; as for our algorithm, we will consider the average



(a) Original *(b) McCann99 with 32 visits per pixel*

(c) McCann99 with 128 visits per pixel *(d) Proposed algorithm*

Figure 4.11: Photo of the interior of Ely cathedral (England), provided by the authors. Here it was processed with McCann99 retinex with (b) 32 and (c) 128 visits per pixel, and then compared with 32 visits per pixel of our approach (d).

number of visits per pixel. The images we will 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. Let us consider first of all the image in figure 4.9a, from Funt et al. (2004). As we can see, figure 4.9b, processed with McCann99, shows a halo around the white square in the centre of the image, and a dark halo inside the square. Both halos have a rather clear bias towards the top-right corner of the image. The image processed with our algorithm instead shows a halo only outside the white square, evenly spread in all directions due to the randomness of the pseudo-Brownian path. We can see this in both figures 4.9c and 4.9d, but especially in the latter, due to the smaller number of visits per pixel. Moreover, figure 4.9c shows that with a similar number of iterations our algorithm delivers a more plausible dark background, as opposed to the light grey resulting from the McCann99 algorithm. In figure 4.10 we show a photo from the Kodak dataset of true colour images (we show the complete dataset in appendix A). The original has a good quality, and we can see that McCann99 introduces some dark halo around the lighthouse; moreover, it smears the narrow frame at the bottom of the image, as we can see in figure 4.10b, 4.10d and 4.10f. The same artefacts are not present with our Brownian-based approach. In figure 4.11 no artefacts are evident in any of the processed images, however McCann99 has to iterate substantially longer to achieve the same result as our approach. These three images suggest what the one in figure 4.12 shows more clearly. It is known that retinex converges to the original image when iterating a large number of times (Brainard and Wandell, 1986), and to do this our method requires fewer comparisons than the McCann99. Specifically, to obtain the image in figure 4.12, it required one sixteenth of the comparisons for each scale (16×2^s versus 256×2^s, with $s = 1, \ldots, 9$ the number of the scale). We repeated the same test on all the images of the Kodak dataset and we found that the SSD (sum of squared differences) between the original and the processed images is consistently higher for

(a) Original *(b) McCann99* *(c) Proposed algorithm*

Figure 4.12: Image from the Kodak dataset (Kodak, 2004), processed with a multi-scale iteration scheme, i.e., the smaller the scale, the more iterations are performed. With nine scales, the McCann99 algorithm compares each pixel 256×2^s times with its neighbours, with s the number of the scale ($s = 1$ is the full image). Our algorithm obtains this result with 16×2^s comparisons only.

the McCann99 algorithm (20 times out of 24 images), showing that the convergence rate of this algorithm is lower than that of our approach. In appendix A we show the whole Kodak dataset, however we will not display the images processed as described here (i.e., with a very large number of comparisons per pixel) as they look practically identical to the originals.

Algorithm	**Complexity**
Proposed retinex	$\mathrm{O}(N \log N)$
McCann (1999)	$\mathrm{O}(N)$
Frankle and McCann (1983)	$\mathrm{O}(N)$
Marini and Rizzi (2000)	$\mathrm{O}(p\chi N)^a$
Random spray (Provenzi et al., 2007)	$\mathrm{O}(nkN)^b$
PDE retinex (Morel et al., 2010)	$\mathrm{O}(N \log N)$

[a] p is the number of Brownian paths per pixel, while χ depends on the length of the paths.
[b] n is the number of sprays per pixel, and k the number of comparisons per spray.

Table 4.1: Computational complexities of retinex implementations.

4.6 Conclusion

In this chapter we have described an efficient retinex algorithm that combines the advantages of Brownian motion and those of the multi-scale approach. The convergence of this algorithm, as defined by Brainard and Wandell (1986), is faster than that of the McCann99 retinex. This is mainly due to our pseudo-random path generator, which we use to simulate the Brownian motion. This algorithm generates paths that guarantee a minimum and average number of visits per pixel. Despite these strong constraints, they maintain certain statistical features of the random walk and, to some extent, of the Brownian motion as well. Furthermore, the computational complexity of our algorithm is $\mathrm{O}(N \log N)$ on the number of pixels visited. One drawback of our approach is the amount of memory required, which is $\mathrm{O}(N)$. Even though further work may improve this, it is unlikely that the linear lower bound can be overcome, because the spanning tree has to be stored in order to be visited. From the point of view of the computational complexity, some improvements might be introduced by considering another algorithm for the MST problem, for example that introduced by Karger et al. (1995). This algorithm runs in expected linear time and, in our problem, its worst-case running time would be $\mathrm{O}(N \log N)$.

Chapter 5

Paths in gradient-domain data fusion

Gradient based operations are a powerful tool widely used in image processing and computer vision, from edge detection (e.g., Gonzalez et al., 2004) to salient point extraction (e.g., Jain et al., 2001). However, because the gradient as such is defined only for scalar functions, in many cases the input image is converted into greyscale before any further operation. While this can be acceptable in some circumstances, it can cause loss of detail in the presence of chromatic differences. With these issues in mind, Di Zenzo (1986) introduced the colour tensor. Informally, Di Zenzo's insight was that the ideas of "magnitude of difference" and "difference in a given direction" were ideas which held in colour as well as greyscale. Di Zenzo's consideration resulted in the so-called colour structure tensor, a 2×2 symmetric positive semidefinite matrix summarising pixel-wise all the gradients in all the image channels in the x and y directions and the correlations between these gradients. Importantly, this structure tensor can be decomposed, via an eigenvector decomposition, into a scalar x and y derivative field. These derivatives best approximate, in a least-squares sense, the derivatives of the multichannel image. Di Zenzo's tensor has been used to extend the detection of luminance-based only features to the colour domain (Cumani, 1991; van de Weijer et al., 2006).

Another application of Di Zenzo's tensor, and the focus of this work, is image fu-

sion. In the image fusion problem a multichannel image (e.g., a three-channel colour image or a 200-channel satellite image) is represented by a single summary greyscale. An elegant solution to the image fusion problem is to use Di Zenzo's tensor to calculate x and y derivatives per pixel that best approximate the multichannel derivatives. Then Socolinsky and Wolff (2002) observed that one can reintegrate this gradient and obtain a visualisation of the chromatic contrast of an image. Moreover, since the colour tensor is not limited to working with three channels only, one can obtain greyscale visualisations of multispectral images of any number of channels, finding applications in, for example, the displaying of remote sensing data.

This gradient-based approach is a significant improvement over previous methods that attempt to find "good" weights for computing the average of channels (e.g., Raol, 2010). These methods can be substantially seen as projections of the image range onto an achromatic line, with the drawback that any variation that is coplanar to the projection will not be visible in the output image. Moreover, in satellite imaging it is often necessary to include a preprocessing step that corrects the effect of atmospheric scattering. A typical technique for doing this consists only of the subtraction of a constant value from each channel (dark-object subtraction, Chavez, 1988), which is not needed when working with gradients. It is to be noted that the colour tensor is not the only way of finding a multispectral gradient. For example, for colour-specific applications Gooch et al. (2005) (and in a similar manner Rasche et al., 2005) suggest to use the colour distance in the CIE-Lab space. However, as mentioned above, the colour tensor is defined for any number of channels, so it can be directly applied to multispectral images. Moreover, its definition is consistent with that of gradient, that is, the colour tensor gradient of a scalar image (i.e., greyscale) is equivalent to the gradient of the image itself.

Unfortunately, the Socolinsky and Wolff method does not always work. There are two problems. Firstly, the colour tensor decomposition into x and y derivatives does not in fact return the sign of the gradient vector. In Socolinsky and Wolff's method

the sign is set to equal the sign of the brightness channel (when this sign is not defined it is set arbitrarily). Secondly, and more seriously, the derived gradient field is not integrable. This is not surprising. When we differentiate an N-pixel image we have N gradients in the x direction and N in the y direction: twice as much data we started with. Changing any of these derivatives can result in a non-integrable gradient field, i.e., the circumstance where the modified gradient field does not correspond to any real image. Almost always when we derive a gradient via the Di Zenzo structure tensor we end up with a non-integrable gradient field.

To get around this problem Socolinsky and Wolff use an approximate reintegration, so that their greyscale output has a gradient that is the closest possible to the colour tensor gradient in a least-squares sense. This approach has the drawback that the least-squares approximated reintegration can have visible artefacts such as unnatural contrast and smeared edges (e.g., figure 5.2).

With our work we wish to improve Socolinsky and Wolff's method by reintegrating results of Di Zenzo's gradient fields but without the artefacts. We begin with the simple observation that Di Zenzo's gradient field of a greyscale image is exactly integrable (since, trivially, it is the same as the greyscale gradient). What happens then if we mix in some proportion a greyscale image (e.g., the brightness image, or some other derived greyscale) to each of the image channels? Intuitively, we expect two things: first (and usefully) the Di Zenzo gradient field to become more integrable and second (less usefully) for the reintegrated image to look more like the chosen greyscale.

Mixing grey in is, of course, another way of saying we desaturate the image. Remarkably, we show that only small degrees of desaturation can result in a much more integrable gradient field. This is especially true when we take a logarithm of the image, calculate the Di Zenzo gradient field and reintegrate. We do this because, perceptually, log-differences are much more meaningful measures of contrast. The move to log-space has a second benefit: there exists a well known image enhancement algorithm, called

retinex, which we use to reintegrate a gradient field. Retinex has the property that, while reintegrating, it simultaneously increases the contrast of the reintegrated image. That is, using retinex we can "put back in" any contrast we might have lost by desaturating the image. For reintegration, the obvious choice here would be to use the path-based retinex that we introduced in chapter 4, however this approach does not prove to be the best choice. The problem is that Brownian paths connect pixels that are not neighbours, making it difficult to apply this path-based retinex to gradients. For this reason, we found that it is more practical to use directly a method from the literature.

Reintegration has been approached in various ways since it arises in a number of situations. Fattal et al. (2002) for example propose to manipulate the gradients of radiance images to compress their high dynamic ranges and then reintegrate these. For the reintegration they suggest to use a multigrid solver (e.g., Press et al., 2007). Agrawal et al. (2005, 2006) instead consider different problems that yield non-integrable fields, for example surface reconstruction, and suggest a graph-based method to improve the integrability of a gradient. In data fusion, Piella (2009) encounters a problem similar to the one we are approaching now. In her paper, she proposes an iterative reintegration method and to reduce the error she constrains the solution to a "legitimate" (i.e., error-free) greyscale image. Here we introduce desaturation to improve the integrability of the gradient, thus rather than an error-free reintegration we need a contrast enhancing one.

This chapter is organised as follows. In section 5.1 we provide the definition of the colour tensor and we discuss the problem of its integrability. In section 5.2 we propose our desaturation-based method for reducing artefacts, we extend the concept of saturation beyond colour to multispectral images, and we show how this influences the colour tensor. In section 5.3 we place our reintegration method in logarithmic space, and in section 5.4 we analyse the properties of the colour tensor in logarithmic space. Finally, we discuss how to measure integrability error and, at the end of the chapter, we

present some results of our data fusion method.

5.1 Background

5.1.1 The colour tensor

Di Zenzo (1986), in his paper, represented a multichannel image as a function $f\colon \mathbb{R}^2 \to \mathbb{R}^3$ and then considered the Jacobian matrix

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} \end{bmatrix}, \tag{5.1}$$

where $f_i(x, y)$ represents the ith colour channel. As J comprises the gradient information relative to all the channels of the image, given a vector $\mathbf{u} = [\cos\theta, \sin\theta]^T$ of unitary length, the vector $J\mathbf{u}$ represents the colour change in direction $\mathbf{u}$. Di Zenzo then proceeded with maximising $\|J\mathbf{u}\|_2$, i.e., finding the direction of maximum change of f and the relative magnitude. He performed explicitly this maximisation by writing the function $\|J\mathbf{u}\|_2^2 = \mathbf{u}^T J^T J\mathbf{u} = F(\theta)$,

$$F(\theta) = g_{11}^2 \cos^2\theta + 2g_{12}\cos\theta\sin\theta + g_{22}^2\sin^2\theta, \tag{5.2}$$

where g_{ij} are the coefficients of the symmetric 2×2 matrix $J^T J$ such that $g_{12} = g_{21}$. He then computed the zeros of $dF/d\theta$, finding that the extrema occur in correspondence of

$$\theta = \arctan\left(\frac{2g_{12}}{g_{11} - g_{22}}\right) \pm \frac{\pi}{2}. \tag{5.3}$$

Socolinsky and Wolff (2002) introduced the same concept (calling it structure tensor), but extended it to generic multispectral images, represented, similarly as before,

by a function $f\colon \Omega \subset \mathbb{R}^2 \to \mathbb{R}^n$. Further, they solved the maximisation problem in a different, and arguably more elegant way: because $\|J\mathbf{u}\|_2^2 = \mathbf{u}^T J^T J\mathbf{u}$ is a quadratic form, its maximum value is equivalent to the largest eigenvalue of the 2×2 matrix $J^T J$ in the direction given by the corresponding eigenvector (e.g., Horn and Johnson, 1985). Formally, they solve

$$(J^T J - \lambda \mathbf{I})\mathbf{u} = \mathbf{0} \tag{5.4}$$

for some scalar λ and some vector $\mathbf{u}$. Since $J^T J$ is a 2×2 matrix, its characteristic polynomial

$$\det(J^T J - \lambda \mathbf{I}) = 0 \tag{5.5}$$

is quadratic, thus the computation of the eigenvalues is straightforward. Say that we have

$$J^T J = \begin{bmatrix} g_{11} & g_{12} \\ g_{12} & g_{22} \end{bmatrix} \tag{5.6}$$

($J^T J$ is symmetric by construction). Then, the two eigenvalues are

$$\lambda^{\pm}(J^T J) = \frac{1}{2}\left(g_{11} + g_{22} \pm \sqrt{(g_{11} - g_{22})^2 + 4g_{12}^2}\right), \tag{5.7}$$

where with $\lambda^+(\,\cdot\,)$ and $\lambda^-(\,\cdot\,)$ we denote the maximum and minimum eigenvalue of a matrix respectively. Now, substituting either of the eigenvalues for λ in equation (5.4) yields the corresponding eigenvector. The system has infinite solutions, because the length of the eigenvector is unspecified, thus one can choose any vector $\mathbf{u}_1 \neq \mathbf{0}$ from the space of the solutions and then normalise it. For example, it can be convenient to choose

$$\mathbf{u}_1(\lambda) = \begin{bmatrix} g_{12} \\ \lambda - g_{11} \end{bmatrix} \tag{5.8}$$

and thus the eigenvector corresponding to the maximum eigenvalue will be

$$\mathbf{v}^{+}(J^{T}J) = \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}. \tag{5.9}$$

Now the colour tensor gradient on a certain pixel would be given by

$$T_x = \sqrt{\lambda^{+}} v_x^{+}(J^{T}J), \quad T_y = \sqrt{\lambda^{+}} v_y^{+}(J^{T}J). \tag{5.10}$$

The problem with this gradient is that the sign is not defined. The eigenvalues of $J^{T}J$ are always non-negative, and the eigenvectors can be chosen arbitrarily from the infinite solutions of (5.4). A possible workaround to this issue is to use the sign of some greyscale version of the image as sign for T. Other solutions are explored by Drew et al. (2009).

5.1.2 Colour tensor of a single-channel image

The property we show here follows directly from the definition of the colour tensor and its formulation as an eigenvalue problem, and has already been pointed out by Socolinsky and Wolff (2002). If we compute the colour tensor of a single-channel image, the gradient defined this way will correspond to the definition of gradient of a scalar function (except for the sign). If the image is $G\colon \mathbb{R}^2 \to \mathbb{R}$, its Jacobian will be equal to the gradient $J = \nabla G$, therefore

$$J^{T}J = \begin{bmatrix} \left(\frac{\partial G}{\partial x}\right)^2 & \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} \\ \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} & \left(\frac{\partial G}{\partial y}\right)^2 \end{bmatrix}. \tag{5.11}$$

In this case, $J^{T}J$ has rank 1 and determinant 0, because J has only one row. Therefore, one of the eigenvalues of $J^{T}J$ is zero. Knowing that $\operatorname{tr}(J^{T}J) = \sum_i \lambda_i$, we do not need the characteristic polynomial to find $\lambda^{+}(J^{T}J) = (\partial G/\partial x)^2 + (\partial G/\partial y)^2$ and

$\lambda^-(J^TJ) = 0$. We can immediately see that $(\lambda^+(J^TJ))^{1/2} = \|\nabla G\|_2$. Using formula (5.8) one can readily see that the colour tensor yields ∇G, albeit the sign remains undefined.

Note that whenever the columns of J are linearly dependent (in this case they are, because they are scalars), $\mathrm{rank}(J^TJ) = 1$ and thus the colour tensor has only one non-zero eigenvalue.

5.1.3 Integrability issues

The main problem with a gradient field $T = (T_x, T_y)$ yielded by the colour tensor is its integrability. In general, T is non-integrable, therefore its visualisation is not straightforward. Socolinsky and Wolff suggest to solve the minimisation problem

$$\arg\min_L \|\nabla L - T\|_2, \tag{5.12}$$

that is, finding the greyscale image L whose gradient is the closest to T in a least-squares sense. To do that, they set up a Poisson equation, by calculating the divergence of the gradient T:

$$\nabla^2 L = \frac{\partial T_x}{\partial x} + \frac{\partial T_y}{\partial y}. \tag{5.13}$$

A solution to this equation that satisfies (5.12) can be found by using an iterative relaxation method (Socolinsky and Wolff, 2002), or deconvolving by a Laplacian filter using Fourier methods (Borenstein, 1999).

In figure 5.1 we show an example of non-integrability of the colour tensor gradient field. That specific configuration of colours yields a gradient $T = \{\Delta x_1 = \sqrt{2}, \Delta x_2 = \sqrt{2}, \Delta y_1 = \sqrt{2}, \Delta y_2 = 0\}$. Reintegrating T is equivalent to finding four scalar values for the pixels a, b, c, d, minus an integration constant k. Thus, assuming that the pixel

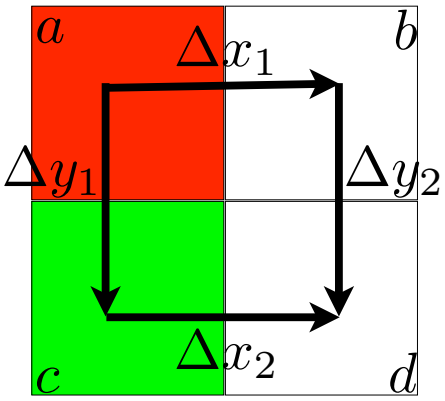


Figure 5.1: Four pixels of RGB values $a = [1, 0, 0]$, $c = [0, 1, 0]$, $b = d = [1, 1, 1]$ and their colour tensor gradient $T = \{\Delta x_1 = \sqrt{2}, \Delta x_2 = \sqrt{2}, \Delta y_1 = \sqrt{2}, \Delta y_2 = 0\}$.

value for a is k, we would have the following:

$$\begin{cases} a = k \\ b = d = a + \sqrt{2} \\ c = a + \sqrt{2} \\ d = b = c + \sqrt{2} = a + 2\sqrt{2}. \end{cases} \tag{5.14}$$

The second and fourth relations are not compatible. One might argue that, since the colour tensor does not provide a sign for the gradient, it could be possible to change the signs in T so that the inconsistencies in (5.14) disappear. However it is not so, as one can see by trying all the 2^3 possible sign assignments to T. While this example may seem artificial, similar situations are not rare when computing a colour tensor gradient field over real images. Notice the abnormal contrast between the hats and the sky in figure 5.2. This results from non-integrability and we can see in figure 5.2c that there is correspondence between visible artefacts and curl.

Given a generic vector field $V = (V_x, V_y)$, V is integrable if and only if its curl is

(a) Input image *(b) Socolinsky and Wolff* *(c) Curl map*

Figure 5.2: An example of image (from the dataset by Kodak, 2004) whose colour tensor gradient field is non-integrable: the Fourier integration yields an output image with a dark halo by the second hat from the left. In (c) we visualise the curl of the colour tensor gradient (darker is smaller, lighter is larger). We notice that the most visible artefacts appear where the curl is large for many pixels.

zero, i.e.,

$$\mathrm{curl}(V) = \frac{\partial V_x}{\partial y} - \frac{\partial V_y}{\partial x} = 0. \tag{5.15}$$

V can be defined as a combination of an integrable component V_0 and a non-integrable one V_ε

$$V = V_0 + V_\varepsilon, \tag{5.16}$$

thus from (5.15) follows that

$$\mathrm{curl}(V) = \mathrm{curl}(V_0) + \mathrm{curl}(V_\varepsilon) = \mathrm{curl}(V_\varepsilon). \tag{5.17}$$

From this last equation we can see that $\mathrm{curl}(V)$ carries some information about the integrability of V, since it is entirely dependent on V_ε. In the example shown in figure 5.1, we have $|\,\mathrm{curl}(T)| = \sqrt{2}$. In all our tests we consider only the absolute value of the curl, since its sign does not add any information about integrability.

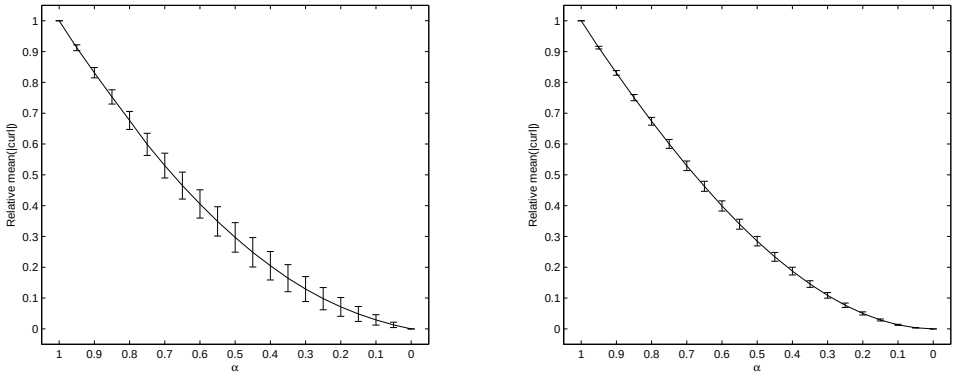
5.2 Reducing integrability artefacts

Observing figure 5.2 we are given a clue on how to proceed. The integrability artefact occurs at an edge which is highly coloured (where the strength of the edge does not just depend on brightness). Perhaps if we could reduce the magnitude of the gradients near this edge then integrability would improve and we would recover a better greyscale reproduction.

To investigate this idea we define the saturation of an image by a scalar α, so that $\alpha = 0$ for a greyscale image and $\alpha = 1$ for the colour original. For this experiment α controls the saturation channel of the HSV (hue, saturation and value, e.g., Gonzalez et al., 2004) image representation. If an (R, G, B) pixel maps to (h, s, v) then we diminish saturation by 50% resulting in $(h, 0.5s, v)$ and inverting the HSV transform results in an RGB image which we input to the Socolinsky and Wolff method. In figure 5.3a we show the mean curl changing with α with standard deviation bars for the 24 Kodak reference images (Kodak, 2004). Notice how the curve decreases most quickly near $\alpha = 1$, that is, we obtain the quickest improvement in integrability by a small change in our original colour image.

It turns out, for the purposes of our analysis, that the HSV colour space is not the easiest to deal with, therefore we provide a different definition of saturation that is more suitable for us. Given a multispectral n-channel image $I = \{C_1, C_2, \ldots, C_n\}$, we will call its *brightness* the value

$$l = \frac{C_1 + C_2 + \ldots + C_n}{n}. \tag{5.18}$$



(a) Desaturation in HSV space *(b) Desaturation in RGB space*

Figure 5.3: Mean curl versus decreasing saturation. Desaturation is performed as in equation (5.20) in (b) and in the HSV colour space in (a). The curves represent the relative mean curl decrease (i.e., normalised by its maximum value), averaged over all the images of the Kodak dataset (Kodak, 2004). The error bars represent the standard deviation for each level of desaturation.

The saturation vector then is

$$S = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \end{bmatrix} = \begin{bmatrix} C_1 - l \\ C_2 - l \\ \vdots \\ C_n - l \end{bmatrix}. \tag{5.19}$$

We will denote with L a vector of length n having all its elements equal to l. Thus, from (5.19) we have $I = L + S$, and in order to adjust the saturation we use again the factor α so that we obtain a new image

$$I' = L + \alpha S. \tag{5.20}$$

We repeat the desaturation experiment for the 24 Kodak images with this new definition of saturation. The results are shown in figure 5.3b. One very useful property of this

formulation is the orthogonality between L and S. Using the definition of S as in (5.19), the only non-trivial value of l that solves the equation $L \cdot S = 0$ is that in equation (5.18), that is, the average of the channels. As a consequence of this property, and because the Jacobian matrix of a desaturated image I' is

$$J = J_L + \alpha J_S, \tag{5.21}$$

(with J_L and J_S Jacobian matrices of the L and S component of I, respectively) the colour tensor of (5.20) is

$$\begin{aligned} J^T J &= (J_L + \alpha J_S)^T (J_L + \alpha J_S) \\ &= J_L^T J_L + \alpha^2 J_S^T J_S + \alpha J_L^T J_S + \alpha J_S^T J_L \\ &= J_L^T J_L + \alpha^2 J_S^T J_S, \end{aligned} \tag{5.22}$$

where the cross products $J_L^T J_S$ and $J_S^T J_L$ are null because the columns of J_L and J_S are orthogonal. In this framework, we can prove some interesting properties of the colour tensor. First, we can show that its determinant shrinks with the desaturation. Second, we can show that if $\det(J^T J) = 0$ both in the image and in its 180-degree rotated version, then the curl of the gradient field is zero as well. These two properties may lead us to think that there is a direct relation between desaturation and curl. Unfortunately, we found that this is not true, and we will provide a counterexample that proves this. Let us begin with proving the following

Lemma 5.1. *The relation between the desaturation factor α and the determinant of the colour tensor is given by*

$$\det(J^T J) = \lambda^+(J_L^T J_L)\alpha^2 d + \alpha^4 \det(J_S^T J_S), \tag{5.23}$$

where L and S are the components of the image as in equation (5.20)*, J_L and J_S their*

respective Jacobian matrices and d *a non-negative real number.*

Proof. Let us consider a matrix C that diagonalises $J_L^T J_L$. C will be orthogonal because $J_L^T J_L$ is real and symmetric, therefore $J_L^T J_L = C^T D_L C$. For simplicity, from now on we will denote with $\lambda_L = \lambda^+(J_L^T J_L)$ the only non-zero eigenvalue of $J_L^T J_L$. From this,

$$\begin{aligned} \det(J^T J) &= \det(J_L^T J_L + \alpha^2 J_S^T J_S) = \\ &= \det(C^T D_L C + \alpha^2 C^T R_S C), \end{aligned} \tag{5.24}$$

where R_S is a generic matrix such that $J_S^T J_S = C^T R_s C$. Since C is orthogonal, $\det(C) = 1$, and because $\det(AB) = \det(A)\det(B)$ if A and B are square matrices,

$$\begin{aligned} \det(J^T J) &= \det(C^T [D_L + \alpha^2 R_S] C) = \\ &= \det(D_L + \alpha^2 R_S) = \\ &= \det\left(\begin{bmatrix} \lambda_L & 0 \\ 0 & 0 \end{bmatrix} + \alpha^2 \begin{bmatrix} a & b \\ c & d \end{bmatrix} \right) = \\ &= \det\left(\begin{bmatrix} \lambda_L + \alpha^2 a & \alpha^2 b \\ \alpha^2 c & \alpha^2 d \end{bmatrix} \right) = \\ &= (\lambda_L + \alpha^2 a)\alpha^2 d - \alpha^4 bc = \\ &= \lambda_L \alpha^2 d + \alpha^4 \det(R_S) = \\ &= \lambda_L \alpha^2 d + \alpha^4 \det(J_S^T J_S). \end{aligned} \tag{5.25}$$

In section 5.1.2 we showed that for a single-channel image G, the maximum and non-zero eigenvalue λ^+ of its colour tensor is equal to $\|\nabla G\|^2$. Here we are considering L that has $n > 1$ channels equal to l, as defined in equation (5.18). In this case, the colour tensor will be n times the colour tensor of l, therefore $\lambda_L = n\lambda_l = n\|\nabla l\|^2$. □

Before showing that whenever $\det(J^T J) = 0$ the curl of the colour tensor gradient

is zero as well, we will have to prove another lemma.

Lemma 5.2. *Given the colour tensor $J^T J$ of an image, if $\det(J^T J) = 0$, the length of the x and y components of the colour tensor gradient is equal to the distance between the colours in the colour space.*

Proof. We know that when $\det(J^T J) = 0$ the columns of J are linearly dependent, i.e., because J has two columns the vectors they represent must be parallel. Let us denote with J_x the first column of J, relative to the direction x, and with J_y the column relative to the direction y. If they are parallel, we can write the colour tensor as

$$J^T J = \begin{bmatrix} \|J_x\|^2 & \|J_x\|\|J_y\| \\ \|J_y\|\|J_x\| & \|J_y\|^2 \end{bmatrix}. \tag{5.26}$$

Since this matrix has only one non-zero eigenvalue, we have that $\lambda^+ = \|J_x\|^2 + \|J_y\|^2$. According to equation (5.8), the corresponding eigenvector prior to normalisation will be

$$\mathbf{v}^+ = \begin{bmatrix} \|J_x\|\|J_y\| \\ \|J_y\|^2 \end{bmatrix}, \tag{5.27}$$

thus, after some simplifications, the gradient yielded by the colour tensor is

$$\sqrt{\lambda^+}\frac{\mathbf{v}^+}{\|\mathbf{v}^+\|} = \begin{bmatrix} \|J_x\| \\ \|J_y\| \end{bmatrix}. \tag{5.28}$$

We can summarise this property saying that if $\det(J^T J) = 0$, the x and y gradients are equivalent to the lengths of the respective colour differences in the colour space. □

Lemma 5.3. *Given the colour tensor $J^T J$ of an image, if $\det(J^T J) = 0$ everywhere in the image and in its 180 degree rotated version, then the curl of the colour tensor gradient is null.*

Proof. Let us consider figure 5.4a. Assuming we calculate derivatives in the forward direction we can calculate Δx_1 and Δy_1 from the tensor at pixel a, having Jacobian J_a. Calculating the derivatives in the reverse direction we obtain Δx_2 and Δy_2 from the tensor at pixel d, having Jacobian J_d. This is equivalent to computing the colour tensor of an image, and then the colour tensor of the same image rotated by 180 degrees. Now, if the determinant of both colour tensors is null, the following happens. Let $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{d}$ be the points in the colour space that correspond to the four pixels in the image plane in figure 5.4a. The two columns of J_a correspond to $\mathbf{b}-\mathbf{a}$ and $\mathbf{c}-\mathbf{a}$ respectively, and are parallel because $J^T J_a$ has null determinant. Thus, $\mathbf{b}$, $\mathbf{c}$ and $\mathbf{a}$ lie on the same line. The two columns of J_d correspond to $\mathbf{d}-\mathbf{c}$ and $\mathbf{d}-\mathbf{b}$ respectively, and are parallel as well. Thus, $\mathbf{d}$ has to lie in the same line as the other three points. Following lemma 5.2 we can write the curl as

$$\begin{aligned} &\Delta x_1 - \Delta x_2 - \Delta y_1 + \Delta y_2 = \\ =&\|\mathbf{b}-\mathbf{a}\| - \|\mathbf{d}-\mathbf{c}\| - \|\mathbf{c}-\mathbf{a}\| + \|\mathbf{d}-\mathbf{b}\|, \end{aligned} \tag{5.29}$$

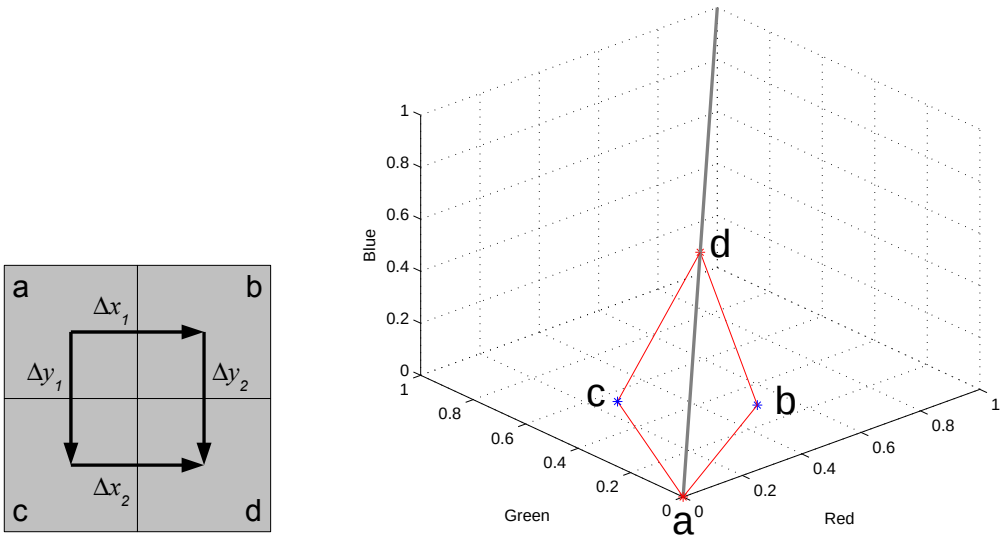
because the lengths of the colour tensor components correspond to the colour distances in the colour space. However, we noted before that the four points lie on the same line, which we can consider a reference axis so that the points can be regarded as scalar values. In this case, the second part of equation (5.29) becomes

$$(\mathbf{b}-\mathbf{a}) - (\mathbf{d}-\mathbf{c}) - (\mathbf{c}-\mathbf{a}) + (\mathbf{d}-\mathbf{b}) = 0, \tag{5.30}$$

because on a line norms are equivalent to simple differences. □

These lemmas provide a useful insight into the properties of the colour tensor. They also lead us to hope that we can find a direct relation between shrinking determinant and shrinking curl. Unfortunately this is not possible, because even if we have evidence that the mean of the curl over an image shrinks with shrinking saturation, this is not true

pixel-wise. In other words, for some colour combinations the curl computed over the colour tensor gradient increases when the saturation decreases. This is a consequence of the fact that, while it is true that $\det(J^T J) = 0$ implies zero curl, the opposite is false, i.e., zero curl does not imply zero determinant or, equally, non-zero determinant does not imply non-zero curl.



(a) *(b)*

Figure 5.4: Configuration of four pixels on the image plane (a) and their relation in the RGB colour space (b).

Let us consider in more detail the issue of curl and how it relates to saturation. Specifically, we intend to show that there are special situations where a reduction in saturation can increase the curl. Let us first make a simple image with 4 colour vectors corresponding to the pixels a, b, c and d in figure 5.4a. For the purposes of this exercise, let the corresponding RGBs be $a = [0, 0, 0]$, $b = [0.25, 0, 0.25]$, $c = [0, 0.25, 0.25]$, $d = [0.5, 0.5, 0.5]$. Again, as we described in the proof of lemma 5.3, let us use forward differences to obtain the colour tensor for a, thus finding Δx_1 and Δy_1, and backward

differences to obtain the colour tensor for d, finding Δx_2 and Δy_2. For a, the tensor is

$$J^T J_a = \begin{bmatrix} 0.125 & 0.0625 \\ 0.0625 & 0.125 \end{bmatrix}, \tag{5.31}$$

with principal eigenvector (orientation) $\mathbf{v}_a^+ = [0.7071, 0.7071]^T$ and eigenvalue $\lambda_a^+ = 0.1875$ (thus, gradient magnitude $(\lambda_a^+)^{1/2} = 0.4330$). For d, the tensor is

$$J^T J_d = \begin{bmatrix} 0.375 & 0.3125 \\ 0.3125 & 0.375 \end{bmatrix}, \tag{5.32}$$

with eigenvector $\mathbf{v}_d^+ = [0.7071, 0.7071]^T$, eigenvalue $\lambda_d^+ = 0.6875$ and gradient magnitude $(\lambda_d^+)^{1/2} = 0.8292$. Choosing the sign of all vectors being positive we have a gradient field that is integrable, because the curl $\Delta x_2 - \Delta x_1 - \Delta y_2 + \Delta y_1$ is by construction equal to zero.

Geometrically, we visualise the colour vectors in figure 5.4b. We have a diamond shape of RGB values, centred on the greyscale axis. We can rotate the diamond by a rigid body transform and be sure that the gradient field derived from the corresponding tensors will have 0 curl. Given a rotation matrix A, it is known to be orthogonal, i.e., $A^T = A^{-1}$ (see, e.g., Press et al., 2007, chap. 21). Therefore, if we try to rotate the Jacobian matrix J, the colour tensor does not change because $(AJ)^T AJ = J^T A^T AJ = J^T J$. Now, suppose we rotate our colour vectors by a matrix

$$A_{r,g,b} = \begin{bmatrix} 0.84 & -0.22 & 0.5 \\ 0.47 & 0.77 & -0.43 \\ -0.29 & 0.60 & 0.75 \end{bmatrix} \tag{5.33}$$

corresponding to a rotation of $\pi/6$ around the red axis, $\pi/6$ around the green axis and $\pi/12$ around the blue axis. The colour tensors for a and d will not change. Now we

desaturate, $\alpha = 0.6$, these tensors and as before calculate the tensors at a and d. Here the tensors are equal to

$$J^T J'_a = \begin{bmatrix} 0.096 & 0.07 \\ 0.07 & 0.090 \end{bmatrix} \tag{5.34}$$

and

$$J^T J'_d = \begin{bmatrix} 0.361 & 0.331 \\ 0.331 & 0.347 \end{bmatrix} \tag{5.35}$$

The corresponding derived gradient vectors are then:

$$\sqrt{\lambda_a^+}\mathbf{v}_a^+ = \begin{bmatrix} -0.2923 \\ -0.2791 \end{bmatrix}, \ \sqrt{\lambda_d^+}\mathbf{v}_d^+ = \begin{bmatrix} -0.5915 \\ -0.5791 \end{bmatrix}. \tag{5.36}$$

There is no assignment of sign to the above vectors which will result in a zero curl, although in this specific example, with the right signs, it will be quite small ($\mathrm{curl} \approx 10^{-3}$). Note that even in the worst case, eventually the curl has to become zero with $\alpha = 0$. Therefore, in the previous example, $\alpha = 0.1$ will yield a smaller curl than $\alpha = 0.6$.

In summary, a diamond arrangement of colours centred on the greyscale axis results in an integrable field for a 4 pixel region. Applying a rigid body transform moves the diamond off the greyscale axis. When we desaturate the resulting vectors, they do not maintain the diamond shape and instead decouple off the plane. In this circumstance a zero curl, integrable field, becomes non-integrable. Of course this counterexample took some effort to construct. For real natural images desaturation, empirically, results in a more integrable derived gradient field. In figure 5.5 we show that in the worst cases the curl grows in about 3% of the pixels.

Let us now look at how this empirical result impacts on real images calculated from real tensor derived gradient fields. In figure 5.6 we show two colour images and their greyscale equivalents calculated according to Socolinsky and Wolff (2002). Notice the

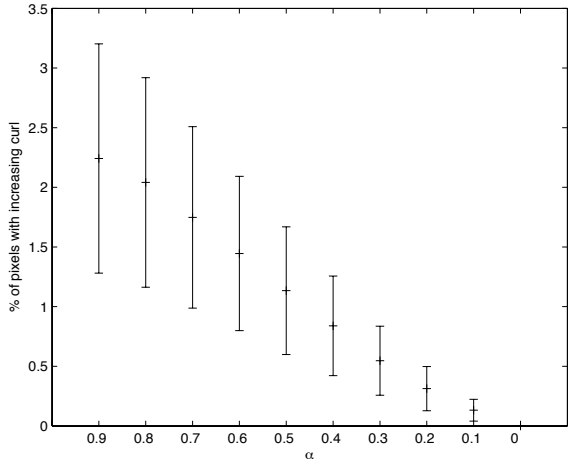
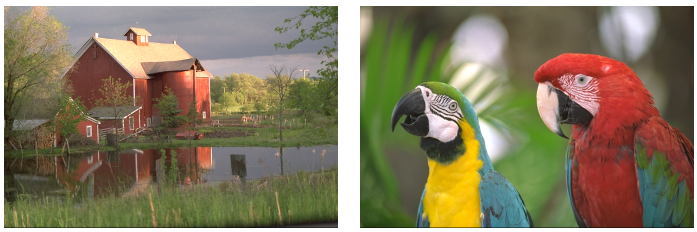


Figure 5.5: Percent of pixels where the curl increases with desaturation, averaged over the Kodak dataset (Kodak, 2004). Error bars indicate in-dataset standard deviations. Note that all images have the same number of pixels.

highly visible integrability artefact in figure 5.6c. In the third row of the figure we show the same colour images where saturation is reduced to 50%. The corresponding Socolinsky and Wolff fused images are shown in the fourth row. Notice that, visually, these images have almost no artefacts. However, viewed critically, it does appear that the Socolinsky and Wolff outputs for the original colour images do capture the original colour aspect more faithfully. We cannot see this in figure 5.6c because the artefact stretches the range of the image well beyond $[0, 1]$ and thus the linear global mapping "washes out" the rest of the image. However, we can prove a lemma that shows that the intensity of the colour tensor gradient shrinks with the desaturation.

Lemma 5.4. *The eigenvalues of $J^T J$ shrink with the desaturation factor α.*

Proof. Let us start the proof with a little remark on the norm of $J\mathbf{u}$. For any non-zero



(a) Original

(b) Original

(c) Socolinsky and Wolff

(d) Socolinsky and Wolff

(e) $\alpha = 0.5$

(f) $\alpha = 0.5$

(g) Socolinsky and Wolff on desaturated image

(h) Socolinsky and Wolff on desaturated image

Figure 5.6: Images from the Kodak dataset (Kodak, 2004) processed with Socolinsky and Wolff's technique before and after desaturation. Note the heavy integrability artefact in (c), completely removed in (g) by means of the desaturation. However, the image in (d) is artefact-free and the desaturation, in this case, only reduces the contrast (h). This phenomenon is not observed in (c) and (g) because the artefact is well out of the range $[0, 1]$. Thus, the linear global mapping makes the rest of the image look "washed out".

vector $\mathbf{u}$ we have that

$$\begin{aligned}\|J\mathbf{u}\|_2^2 = \mathbf{u}^T J^T J\mathbf{u} = \mathbf{u}^T (J_L^T J_L + \alpha^2 J_S^T J_S)\mathbf{u} = \\ = \mathbf{u}^T J_L^T J_L \mathbf{u} + \alpha^2 \mathbf{u}^T J_S^T J_S \mathbf{u} = \\ = \|J_L\mathbf{u}\|_2^2 + \alpha^2 \|J_S\mathbf{u}\|_2^2.\end{aligned} \tag{5.37}$$

In other words, the norm of $J\mathbf{u}$ depends monotonically on α. Let us now consider two values $0 \leq \alpha_1 < \alpha_2 \leq 1$ and their respective Jacobian matrices

$$\begin{aligned}J_1 = J_L + \alpha_1 J_S \\ J_2 = J_L + \alpha_2 J_S.\end{aligned} \tag{5.38}$$

Let $\lambda^+(J_1^T J_1)$ be the maximum eigenvalue of $J_1^T J_1$, and $\mathbf{v}_1^+$ its corresponding eigenvector. Similarly, let $\lambda^+(J_2^T J_2)$ and $\mathbf{v}_2^+$ be the maximum eigenvalue of $J_2^T J_2$ and its corresponding eigenvector. From the properties of the quadratic forms, we have

$$\begin{aligned}\lambda^+(J_1^T J_1) = \mathbf{v}_1^{+T} J_1^T J_1 \mathbf{v}_1^+ = \\ = \|J_1\mathbf{v}_1^+\|_2^2 \leq \|J_2\mathbf{v}_1^+\|_2^2 \leq \|J_2\mathbf{v}_2^+\|_2^2 = \\ = \mathbf{v}_2^{+T} J_2^T J_2 \mathbf{v}_2^+ = \lambda^+(J_2^T J_2).\end{aligned} \tag{5.39}$$

Here, the first inequality follows from equation (5.37) and the fact that $\alpha_1 < \alpha_2$. The second inequality instead is due to the fact that $\mathbf{v}_1^+$ is not, in general, an eigenvector of $J_2^T J_2$, and therefore it does not maximise the quadratic form $\mathbf{x}^T J_2^T J_2 \mathbf{x}$. These relations prove that there is a monotonic relation between α and the maximum eigenvalue of $J^T J$. With the same argument, one can prove that the same holds for the minimum

eigenvalue:

$$\begin{aligned}\lambda^-(J_2^T J_2) &= \mathbf{v}_2^{-T} J_2^T J_2 \mathbf{v}_2^- = \\ &= \|J_2\mathbf{v}_2^-\|_2^2 \geq \|J_1\mathbf{v}_2^-\|_2^2 \geq \|J_1\mathbf{v}_1^-\|_2^2 = \\ &= \mathbf{v}_1^{-T} J_1^T J_1 \mathbf{v}_1^- = \lambda^-(J_1^T J_1),\end{aligned} \tag{5.40}$$

where $\mathbf{v}_1^-$ and $\mathbf{v}_2^-$ are the eigenvectors corresponding to the minimum eigenvalues of $J_1^T J_1$ and $J_2^T J_2$ respectively. □

This result may be somewhat unsurprising, but it shows beyond doubt that the colour contrast encoded in the greyscale derived after desaturation is less. Addressing this issue is the key concern of the next section.

5.3 Contrast enhancing reintegration

In the previous section we showed that desaturation reduces the contrast information carried by the colour tensor gradient. We also proved in lemma 5.4 that this is bound to happen because the magnitude of the gradient shrinks with the desaturation. This fact urges us to find some way of reinforcing the contrast. One could think of applying some post-processing step on the output image, but there would be no obvious choice on what to do. For example, on a somewhat similar situation, Smith et al. (2008) apply a Laplacian pyramid in order to strengthen the edge contrast, but other methods may work equally well. In our case it is more convenient to take a step backward and consider that what we are trying to do is to visualise a gradient field. As stated in section 5.1.3 this is a reintegration problem, therefore one might wonder whether it is possible to introduce some contrast enhancement already at this stage. We find an answer to this question in the relation between PDEs and retinex, which has been known since the work of Horn (1974) and explored again by Borenstein (1999). More recently, Morel et al. (2009,

2010) proved that the original retinex by Land and McCann (1971) is equivalent to a Poisson equation if the chosen paths are symmetric random walks. In all these cases it is shown how PDEs can solve retinex if certain constraints are met. Here, we do the opposite: we have a PDE problem to solve, and we use retinex to do that. We already described retinex in chapter 2, thus here we will only focus on how retinex can be used for reintegration. Let us start from equation (2.10), which we report here:

$$I'(\rho_n) = I(\rho_1) + \delta'\big(I(\rho_2) - I(\rho_1)\big) + \\ + \delta'\big(I(\rho_3) - I(\rho_2)\big) + \ldots + \delta'\big(I(\rho_n) - I(\rho_{n-1})\big). \quad (2.10)$$

Here, we are estimating the colour of the nth pixel of a path ρ. Now, if we ignore the thresholding operator δ', we have a one-dimensional reintegration along this path, provided that the differences represent a discrete derivative operator. If we substitute these differences with the gradient obtained from the colour tensor, the retinex computations will be able to perform the reintegration. In chapter 4 we introduced our path-based retinex algorithm, thus a natural choice would be to apply it to this problem. However, the pseudo-Brownian motion retinex that we proposed does not quite work for this purpose, and it is easier and preferable to use the pseudo-random walk. The former does some "leaps" in its paths, because one of the features of the Brownian motion is the random increment for each step. This is a problem in reintegration, because we would need to compute gradients between non-neighbouring pixels. That is why it is wiser to use the random walk approach, since it always moves in steps of one pixel and therefore uses the information from the colour tensor gradient directly. Therefore, once we have computed the colour tensor gradient $T = (T_x, T_y)$ from an image, we put it in equation (2.10) and we obtain

$$I'(\rho_n) = I(\rho_1) + T(\rho_2, \rho_1) + T(\rho_3, \rho_2) + \ldots + T(\rho_n, \rho_{n-1}), \quad (5.41)$$

where $T(\rho_{i+1}, \rho_i)$ indicates the colour tensor gradient between the two pixels $I(\rho_{i+1})$ and $I(\rho_i)$. Note that we remove the thresholding step, since that would affect the integrability of the gradient, possibly cancelling the benefits of desaturation. The reset step instead is useful because it provides a natural way to choose the integration constant. Let us remind ourselves that we are assuming $I(\rho_1) = 0$, and that the cumulative sum is clipped to zero every time it goes above that value (reset step). In a multi-scale context, one can simply compute the colour tensor gradient and carry out the reintegration for each scale.

Using the random-walk based approach has some drawbacks. The strength of our retinex algorithm, introduced in chapter 4, lies in the pseudo-Brownian paths, which in this case we have to give up because, for the reasons explained above, they do not lend themselves to solve reintegration. Thus, we are limiting our path-generating algorithm to pseudo-random walks. While this approach seems to work generally well, it is possible to see the effects of the random walk on the output if we observe the bottom of the image in figure 5.7. The original has a thin grey frame along the bottom edge (see figure 5.2a on page 88), and the random walk smears it giving what looks like a "diffusion" effect. This issue could be probably removed by increasing the number of visits per pixel, however that would not be practical to achieve.

To solve reintegration, we chose another approach. Knowing that the McCann99 algorithm gives similar problems of smearing to the random walk approach (see section 4.5 on page 72), we based our algorithm on Frankle and McCann's retinex (Frankle and McCann, 1983). Its computational complexity is $\mathrm{O}(N)$ on the number of pixels, which means that our prototype Matlab implementation transforms a 1-megapixel image into greyscale in about 12 seconds on a 2.1 GHz dual core Intel Core Duo processor (this includes the computation of the colour tensor gradient).

Figure 5.7: Example of artefacts produced by the random-walk reintegration. The image looks generally good, however there is a "diffusion" smearing on the narrow frame at the bottom.

5.3.1 Frankle and McCann's retinex

Here we consider Frankle and McCann (1983) retinex and its reference implementation by Funt, Ciurea, and McCann (2000, 2004). We use the same notation as in section 4.1.2 on page 58, where we described the McCann99 algorithm: the letters O, I and N in superscript stand for old value, intermediate value and new value respectively, and no superscript indicates the original image value. In some way, Frankle and McCann's algorithm is very similar to the McCann99, but the multi-scale approach is taken less explicitly. Frankle and McCann define a shift value s that varies during the computation. So, if we are considering a pixel i, j, along the rows we have

$$\begin{aligned} I^{\mathrm{I}}_{i,j} &= I^{\mathrm{O}}_{i+s,j} + I_{i,j} - I_{i+s,j}, \\ I^{\mathrm{N}} &= \frac{1}{2}(\min\{I^{\mathrm{I}}, 0\} + I^{\mathrm{O}}), \end{aligned} \tag{5.42}$$

where the second line is an average updated at each iteration (this was one of the novelties introduced by this retinex algorithm), and the $\min$ operation is the reset step. Similarly, along the columns we have

$$\begin{aligned} I^{\mathrm{I}}_{i,j} &= I^{\mathrm{O}}_{i,j+s} + I_{i,j} - I_{i,j+s}, \\ I^{\mathrm{N}} &= \frac{1}{2}(\min\{I^{\mathrm{I}}, 0\} + I^{\mathrm{O}}). \end{aligned} \tag{5.43}$$

If the image size is $w \times h$, the subscripts will range in $1 \ldots w$ for the columns and in $1 \ldots h$ for the rows, i.e., for example j will stop when $j+s = w$ or when $j+s = 1$ (this second case occurs when s is negative). When no subscript is indicated, the operation is performed on all pixels. Both equations (5.42) and (5.43) are computed in each iteration, and a user-defined parameter k sets the number of iterations performed for each s. The shift value s is initialised to the second largest power of 2 that is smaller than both w and h (i.e., $s = 2^{\lfloor \log_2(\min\{w,h\}) \rfloor - 1}$), then updated to $s := -s/2$ until $|s| < 1$. The change of sign is required to avoid the biasing of the comparisons towards one direction only.

As we already mentioned above, on a N-pixel image the complexity of this algorithm is $\mathrm{O}(N)$, although it depends also on the number of iterations k, which is normally small. Let us assume, without loss of generality, that $N = (2^m + 1) \times (2^m + 1)$ and that s remains always positive. For simplicity, let us initialise $s = 1$ and proceed "backwards", doubling its value every time instead of halving it. With $s = 1$, we perform $k(2^m \times (2^m + 1))$ comparisons for the rows and just as many for the columns, plus N averages. With $s = 2$ the number of comparisons halves, i.e., we have $k(2^{m-1} \times (2^m + 1))$ for the rows and the same for the columns, again plus N averages. From these two cases

we can infer the general formula

$$\begin{aligned}\sum_{i=0}^{m} \frac{2k(2^m \times (2^m+1))}{2^i} + \mathrm{O}(N) = 2k(2^m \times (2^m+1)) \sum_{i=0}^{m} \frac{1}{2^i} + \mathrm{O}(N) = \\ = 2k\,\mathrm{O}(N) \sum_{i=0}^{m} \frac{1}{2^i} + \mathrm{O}(N) = \mathrm{O}(kN),\end{aligned} \tag{5.44}$$

that becomes $\mathrm{O}(N)$ when k is small.

Finally, let us see in detail how we can use Frankle and McCann's retinex to reintegrate a colour tensor T. It should be clear from equation (5.41) that, in general, substituting the colour tensor gradient for the difference step in the retinex computation will reintegrate the gradient. However, in Frankle and McCann the comparisons are performed between pixels that are far away from each other, and there is no explicit downsampling of the image so that one could compute a colour tensor gradient for each scale. On the other hand, the comparisons are carried out first row-wise and then column-wise, so one can simply consider the colour tensor of the full-size image and sum its rows or its columns so that the final difference corresponds to the correct shift s. Thus, the first line of equation (5.42) becomes

$$I^{\mathrm{I}}_{i,j} = I^{\mathrm{O}}_{i+s,j} + \sum_{\iota=i}^{i+s} T_{y(\iota,j)} \tag{5.45}$$

and the first line of equation (5.43) becomes

$$I^{\mathrm{I}}_{i,j} = I^{\mathrm{O}}_{i,j+s} + \sum_{\iota=j}^{j+s} T_{x(i,\iota)}. \tag{5.46}$$

In practice, the one-dimensional reintegration described for path-based retinex exists also here, with the difference that it is performed separately in the two directions x and y.

5.4 Colour tensor in logarithmic space

In the previous section we described a retinex-based integration method, but we did so without taking into account that retinex requires us to work with the logarithm of an image. In general, it has long been known that logarithms have certain advantages over the linear representation of images (Stockham, 1972). Moreover, percentage differences are perceptually more important than absolute differences (Weber's law, e.g., Wandell, 1995). However, we cannot assume that this would not affect the colour tensor and it is therefore important to consider explicitly how logarithms influence it and what happens to the integrability of its gradient. We represent the colour tensor of a log-image considering the chain rule of derivatives:

$$\frac{d}{dx}\log f(x) = \frac{1}{f(x)}\frac{df(x)}{dx}. \tag{5.47}$$

If we denote with **J** the Jacobian matrix of a log-image, we have

$$\mathbf{J} = \begin{bmatrix} \frac{1}{f_1}\frac{\partial f_1}{\partial x} & \frac{1}{f_1}\frac{\partial f_1}{\partial y} \\ \frac{1}{f_2}\frac{\partial f_2}{\partial x} & \frac{1}{f_2}\frac{\partial f_2}{\partial y} \\ \frac{1}{f_3}\frac{\partial f_3}{\partial x} & \frac{1}{f_3}\frac{\partial f_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{1}{f_1} & 0 & 0 \\ 0 & \frac{1}{f_2} & 0 \\ 0 & 0 & \frac{1}{f_3} \end{bmatrix} \times \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} \end{bmatrix}. \tag{5.48}$$

Let us call Λ the diagonal matrix in the above equation. The colour tensor of the log-image will then be

$$\mathbf{J}^T\mathbf{J} = J^T\Lambda^T\Lambda J = J^T\Lambda^2 J, \tag{5.49}$$

where J is the Jacobian of equation (5.1). This formulation is inherently more complicated than the colour tensor in linear space of equation (5.22), and therefore it is not straightforward to see what happens when desaturating the input image. Let us keep in mind that the desaturation is performed in linear space as described in section 5.2, thus Λ depends on the saturation factor α. For this reason we can resort to empirical

measures. The first result we find is that, unlike for the colour tensor in linear space, the determinant of the colour tensor of a log-image does not in general shrink with the desaturation factor α. This property still holds on average, as we show in figure 5.8, which represents the mean determinant computed over the images of the Kodak dataset. Introducing some approximation, we can formalise a case in which the determinant

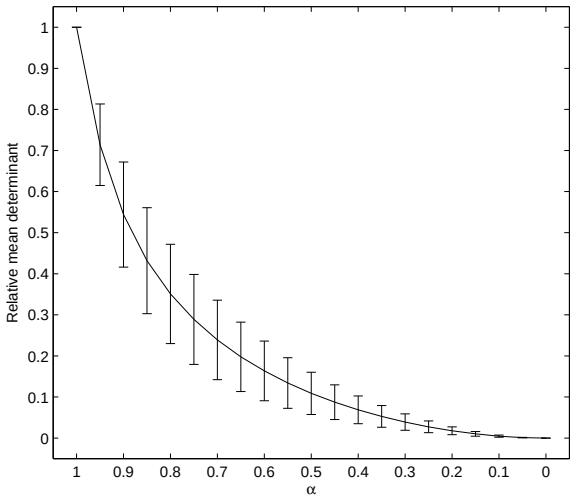


Figure 5.8: Mean determinant in log space versus decreasing saturation. The curve represents the relative mean determinant decrease, averaged over all the images of the Kodak dataset.

shrinks. Namely, considering the Taylor expansion of $\ln(x)$ around 1 with $x \in (0, 1]$, we can show that for very bright colours the determinant shrinks.

$$\begin{aligned} \ln x &= -\sum_{k=1}^{\infty} (-1)^k \frac{(x-1)^k}{k} = \\ &= (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \cdots . \end{aligned} \tag{5.50}$$

When $x \to 1$ this expansion can be approximated as follows:

$$\ln x = (x-1) + \mathrm{O}\left((x-1)^2\right) \approx x - 1, \tag{5.51}$$

because the terms with power $k > 1$ tend to zero much more quickly than the first term, which has power $k = 1$. Now if we substitute x with $l + \alpha S_i$, where S_i is the saturation of the ith channel, and we use the backward differences in order to approximate the derivatives, the Jacobian matrix will be exactly the same as the one that we obtain from the original linear image, and so will be the colour tensor.

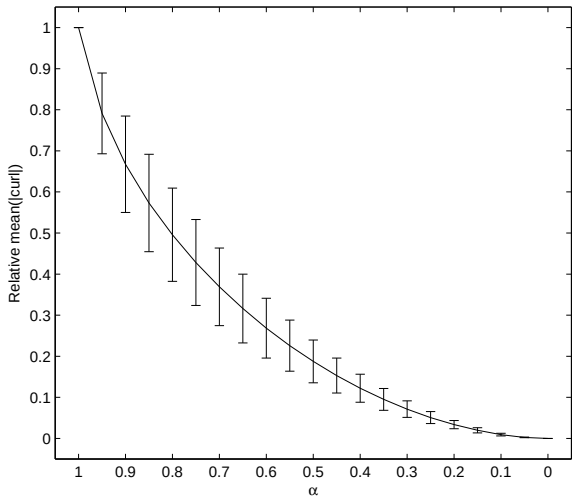


Figure 5.9: Mean curl in log space versus decreasing saturation. This curve is similar to that in figure 5.3, but the gradient is computed on the logarithm of the image.

In a similar fashion to what we did in section 5.2, we display what happens to the mean curl of a log-image in figure 5.9. The desaturation is performed exactly as previously, before taking the logarithm, while the gradient is computed directly on the log-image. The plot shows the relative mean curl averaged over the Kodak dataset (Kodak, 2004), and the error bars represent the standard deviation.

5.5 Determinant as a measure for integrability?

Throughout this chapter we exploited the direct relation between curl and integrability: when the curl of a gradient is zero, the integrability error is zero as well. Because of

this, when a gradient field is non-integrable it is natural to choose the absolute value of its curl as a measure of the integrability error. Thus, when the gradient is obtained from an image, we will average the absolute value of the curl over the whole gradient field. However, we ask: when the gradient field is obtained from the colour tensor, can we have another measure of integrability? We argue that the determinant of the colour tensor matrix is suitable for this purpose.

Let us $Z = J^T J$ be a colour tensor matrix as defined in section 5.1. We have proven in lemma 5.3 that, under certain conditions, $\det(Z) = 0$ implies $\mathrm{curl}(T) = 0$, where $T = (T_x, T_y)$ is the gradient field obtained from the colour tensor Z. The condition for this to happen is that the determinant has to be zero both on the image and on its 180-degree rotated version. Now, we can say that in general when Z is the colour tensor of a n-channel image with $n > 1$, then we have $\mathrm{rank}(Z) = 2$. Let us consider the problem of finding a matrix having $\mathrm{rank} = 1$ that bests approximates Z. To define the "best approximation" we use the Frobenius norm (e.g., Horn and Johnson, 1985), which can be defined as

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n} \lambda_i^2}, \tag{5.52}$$

where A is a $n \times n$ matrix and λ_i are the eigenvalues of A (this is not the general definition of the Frobenius norm; for simplicity we only consider the case in which A is square and has real eigenvalues, similarly to the colour tensor matrix). The problem we are trying to solve is then finding a matrix $\tilde{Z}$, with $\mathrm{rank}(\tilde{Z}) = 1$, which minimises the error function $E(X) = \|Z - X\|_{\mathrm{F}}$. From the literature (e.g., Moon and Stirling, 2000), we can find this matrix $\tilde{Z}$ as follows.

$$Z = V^T D V = V^T \begin{bmatrix} \lambda^+ & 0 \\ 0 & \lambda^- \end{bmatrix} V \approx V^T \begin{bmatrix} \lambda^+ & 0 \\ 0 & 0 \end{bmatrix} V = \tilde{Z}, \tag{5.53}$$

where $V^T D V$ is the eigenvalue decomposition of Z. This is essentially what we are

doing when we compute the colour tensor gradient T. We argue that when $\mathrm{rank}(Z) = 2$, T is in general non-integrable by construction, because we have to introduce this approximation of Z with $\tilde{Z}$. We know that when $\det(Z) = 0$ (of course excluding the trivial case in which $Z = \mathbf{0}$), then $\mathrm{rank}(Z) = 1$, and we know that in this case the gradient it yields is integrable. But in this case, we do not need to introduce any approximation because, since $\mathrm{rank}(Z) = 1$, $\tilde{Z} = Z$. So we can say that the very presence of the second non-zero eigenvalue in Z indicates that some approximation is needed, and we can say that the approximation error is given by λ^- itself. We know, from lemma 5.4, that λ^- is bound to decrease with the saturation, and in lemma 5.1 we provide a measure of how, as a consequence of this, the determinant shrinks as well. We can therefore say that the determinant measures the approximation that we are introducing in the construction of the gradient field and thus, albeit indirectly, provides a measure for the integrability of the gradient field itself.

5.6 Results

In this section we want to show some examples of images processed with our algorithm. In figure 5.10 we consider again the images proposed in figure 5.6, where we showed the effects of desaturation. Here we show their appearance using retinex reintegration. We see that, compared with Socolinsky and Wolff's algorithm, the greyscale images we provide are richer in contrast, thanks to the enhancing qualities of retinex. Another example is that in figure 5.11, from the McGill database (Olmos and Kingdom, 2004). This image is particularly problematic because its colour tensor gradient field is highly non-integrable, as we can see in figure 5.11b. Decreasing saturation helps to some extent, but we can see a gradient in the sky background in figure 5.11d that was not present in the original. Retinex reintegration with desaturation (figure 5.11f) improves the result, which looks however somewhat noisy in the sky background. This

(a) *(b)*

Figure 5.10: Images from figure 5.6 on page 99 converted into greyscale using retinex reintegration and desaturation $\alpha = 0.5$.

noise is present in the original, although it is visible only on a careful inspection. In figure 5.12 on page 114 we show what happens to an image whose colour tensor gradient does not cause integrability artefacts. We can see in figure 5.12c that Socolinsky and Wolff's method works perfectly fine. However, the image produced with our method (figure 5.12d) is generally richer in contrast. Note that when using a traditional colour-to-greyscale, in figure 5.12b, some of the flowers are completely hidden in the background. In figure 5.13 on page 115 we show our first example of n-to-1 channel transformation on an image provided by Fredembach and Süsstrunk (2008a,b). Images 5.13a and 5.13b can be incorporated into a single four-channel image. We can see in figure 5.13d that Socolinsky and Wolff's method works substantially better than a simple average of the channels, as in figure 5.13c. However, our retinex reintegration, in figure 5.13f, substantially improves the contrast of the greyscale output, although without desaturation it provides a useless image, in figure 5.13e. Finally, we conclude in figure 5.14 on page 116 with another example of n-to-1 channel fusion. We show a seven-channel image from the Landsat project (U.S. Geological Survey, 2011). The image is composed of three visible light channels, two near-infrared channels, one mid-infrared channel and a thermal channel (U.S. Geological Survey, 2010). We present

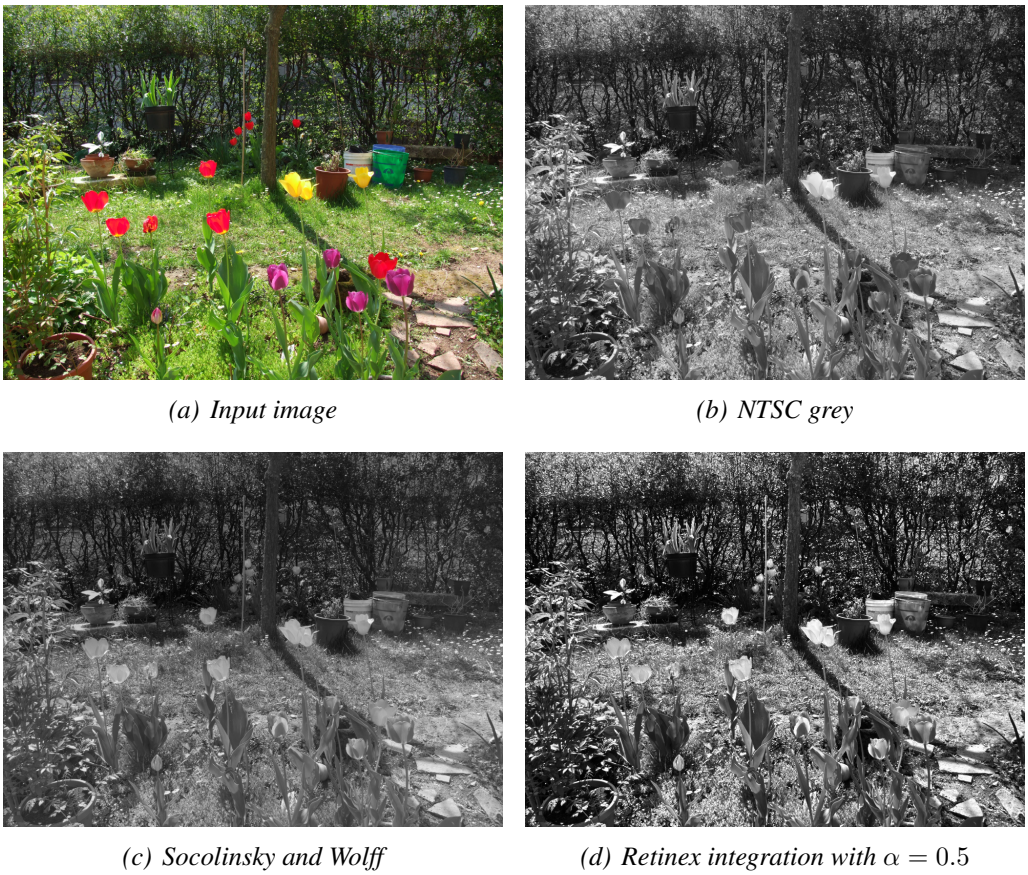
(a) Input image *(b) Socolinsky and Wolff* *(c) Colour image with* $\alpha = 0.5$

(d) Socolinsky and Wolff with $\alpha = 0.5$ *(e) NTSC grey* *(f) Retinex integration with* $\alpha = 0.5$

Figure 5.11: Image (a) from the McGill database (Olmos and Kingdom, 2004) and integration of its colour tensor before (b) and after (d) desaturation. The mean curl of the colour tensor gradient of (a) is 0.0234, while that of (c) is 0.0086. In the last row, we have the retinex integration (f) and a NTSC greyscale (e), obtained with Matlab's *rgb2gray* command.

three different ways of combining the channels. First, in figure 5.14h we show the result of a method that uses PCA to determine the weights for averaging the channels (Raol, 2010) (prior to that we also applied the dark-object subtraction, see Chavez, 1988). In figure 5.14i we show the result of Socolinsky and Wolff's method, and finally in figure 5.14j we show the image produced with our method, in which many details appear sharper than in figure 5.14h and 5.14i.

These examples give only a qualitative idea of the performance of our method. For this reason, we carried out a preference experiment to understand whether or not human observers "like" our method. We describe this experiment thoroughly in the next chapter.



(a) Input image *(b) NTSC grey*

(c) Socolinsky and Wolff *(d) Retinex integration with $\alpha = 0.5$*

Figure 5.12: Image provided by the authors (a) transformed in greyscale with Matlab's *rgb2gray* command (b), with Socolinsky and Wolff's method (c) and with the proposed retinex reintegration (d). Observe that in (b) the red flowers are almost indistinguishable from the grass background, while they are again visible in (c). In (d) they are also slightly darker than the yellow flowers.

(a) Colour image

(b) Near-infrared image

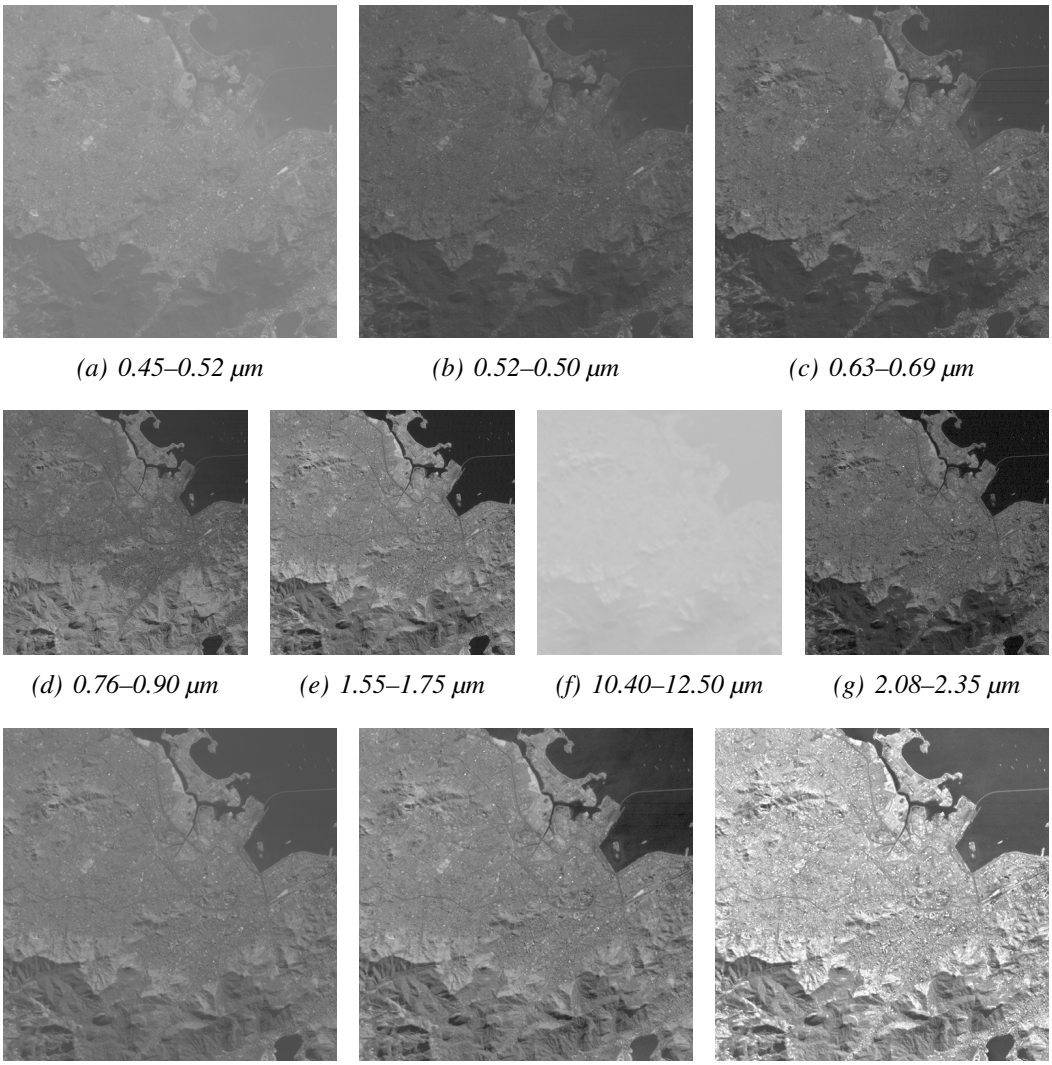
(c) Average image

(d) Socolinsky and Wolff

(e) Retinex reintegration

(f) Retinex with $\alpha = 0.5$

Figure 5.13: A colour (a) and a near-infrared image (b) (Fredembach and Süsstrunk, 2008a) can be seen as a single four-channel image. We rendered it in greyscale as mean of the four channels (c), with Socolinsky and Wolff's algorithm (d), and finally with our retinex integration without (e) and with (f) desaturation. Note that (e) presents heavy artefacts that make it unacceptable for any use.



(a) 0.45–0.52 µm *(b) 0.52–0.50 µm* *(c) 0.63–0.69 µm*

(d) 0.76–0.90 µm *(e) 1.55–1.75 µm* *(f) 10.40–12.50 µm* *(g) 2.08–2.35 µm*

(h) PCA fusion *(i) Socolinsky and Wolff* *(j) Proposed method*

Figure 5.14: Fusion of a seven-channel satellite image (U.S. Geological Survey, 2011) composed by three visible channels (a), (b), (c), two near-infrared channels (d), (e), one thermal (f) and one mid-infrared (g) channel. Our method here, in (j), uses $\alpha = 0.5$ and the ℓ^1 norm of the channels to determine the sign of the colour tensor gradient.

5.7 Conclusion

In this chapter we presented and discussed a method for image fusion, which combines spectral information of multiple channels into a single greyscale image. Our contribution is manifold. First of all, we showed that the integrability error of the colour tensor gradient is reduced by desaturation of the input image. Second, we showed that retinex can be effectively used as a reintegration method.

To achieve these results we analysed some properties of the colour tensor. We proved that reducing the saturation of an image always affects the magnitude of the eigenvalues of the $J^T J$ colour tensor matrix, as well as its determinant. While this seems to point to the direction of proving a reduction of the integrability error, measured by the curl, this is unfortunately not true. There are a minority of ill cases in which reducing the saturation causes the curl to grow. However, in general this does not happen and if we measure the average curl of the colour tensor gradient over a whole image, we find that it shrinks with the saturation.

At this point we would like to remark on the retinex-based reintegration. Our attempt to apply to this problem the implementation of retinex we proposed in chapter 4 has been, at least in part, not too successful. While the idea works in principle, it seems to have some boundary condition issues: we showed in figure 5.7 on page 104 that the pseudo-random walk causes some diffusion artefacts around the edges of an image. This issue may be solved by "moving" the boundary, that is, replicating some pixels at the edge and therefore increasing the size of the input image. However, this is certainly not a proper solution, as we do not know a priori how large the artefact will be, and replicating the whole image would be too computationally expensive. Note that this problem occurs only because we are trying to reintegrate a non-integrable gradient. Our retinex algorithm does not cause the same artefacts when not applied to the reintegration problem. We resorted to use Frankle and McCann's algorithm mainly because it does not present any problem at the boundaries (or, if it does, it is not visible). This

solution has also the advantage of having a lower computational complexity.

Finally, we would like to stress that the relation between PDEs and retinex has been known for some time, however to the best of our knowledge this is the first time that retinex is used to solve a PDE and not vice versa. This has multiple advantages over other methods. First, in the case we use Frankle and McCann's implementation, its computational complexity is linear on the number of pixels of an image, while other methods tend to be slower. Second, having been designed as a model for our vision, retinex has strong perceptual foundations, thus we expect that images obtained with our method will be preferred to other images. In the next chapter we will describe a preference experiment that we designed in order to determine whether or not this is the case.

Chapter 6

Psychophysical experiment

In chapter 5 we introduced a data fusion method that relies on retinex to reconstruct a greyscale image from a gradient field. Retinex, which has a central role in this thesis (see the brief introduction in section 2.5 on page 18), has robust perceptual foundations, since it was devised as a model for human vision. Therefore, we expect that our data fusion method will be successful in the sense that it will produce images pleasant to look at. Clearly, there is the need to test this hypothesis. To do this, we set up a preference experiment where observers judged the colour to greyscale method that produces the images they "preferred most". This chapter is structured in three parts. In section 6.1 we will provide some background on preference experiments, introducing Thurstone's law of comparative judgement. In section 6.2 we will detail the set up of our experiment and its results and finally, in section 6.3, we will discuss its outcome.

6.1 Background

Thurstone's law (Thurstone, 1927) was introduced with the purpose of constructing scales of psychophysical stimuli. Intuitively, we can say that for obtaining a total ordering of stimuli, it is sufficient to compare them pair by pair, constructing a "tourna-

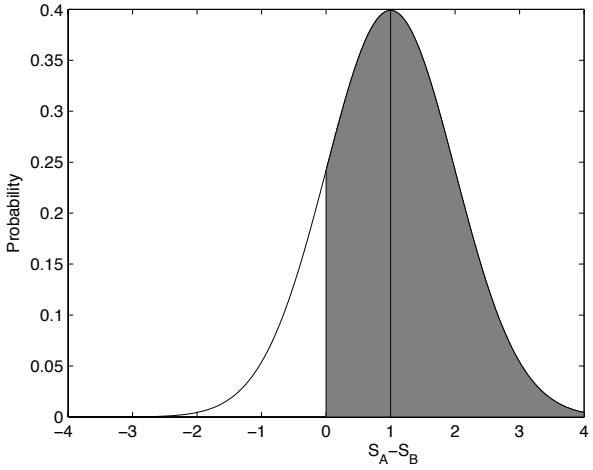


Figure 6.1: The shaded area represents the probability $P(S_A > S_B)$ that the response to a stimulus S_A is greater than that to a stimulus S_B.

ment" in which the stimulus that produces a greater response "wins" the comparison. Thurstone's idea is that discriminating between two stimuli causing responses S_A and S_B ("scale values") can be modelled as a normally distributed random variable. This distribution represents $S_A - S_B$ over a number of observations, and its mean should be a good estimate of its actual value. However, observers are never asked to quantify the difference between stimuli. Rather, they only judge which one is greater, and from this information, with some assumptions, one can estimate the scale values of the stimuli. For example, let us say that S_A causes a response greater than that of S_B. The proportion of times that S_A is judged greater than S_B will determine a normal distribution such as that in figure 6.1, where the area of the shaded region is the probability $P(S_A > S_B)$, or equivalently $P(S_A - S_B > 0)$ (we are using the same notation as Engeldrum, 2000). Thurstone's law has several cases, but here we will consider only case V, since it is the one we used for our experiment as well as the most used in general. Case V introduces a simplification in the standard deviation of the probability distribution of $S_A - S_B$, which is assumed to be $\sqrt{2}$. The value of $P(S_A > S_B)$ is measured experimentally and

corresponds to the normal cumulative distribution function

$$H(S_A - S_B) = \frac{1}{2\sqrt{\pi}} \int_0^{+\infty} \exp\left(-\frac{1}{2}\left(\frac{t - (S_A - S_B)}{\sqrt{2}}\right)\right) dt, \tag{6.1}$$

having mean $S_A - S_B$ and standard deviation $\sqrt{2}$. Then, we can find the scale value difference $S_A - S_B$ by inverting $H(\,\cdot\,)$. That is, assuming that the data will determine the value of

$$H(S_A - S_B) = P(S_A > S_B) \tag{6.2}$$

we have

$$S_A - S_B = H^{-1}(H(S_A - S_B)) = H^{-1}(P(S_A > S_B)). \tag{6.3}$$

The case we have just discussed involves only the comparison between two stimuli. If we have a number of stimuli, we can write the probability of each comparison in a matrix. Say we have three stimuli, we will have

$$\mathbf{P} = \begin{bmatrix} P(S_1 > S_1) & P(S_1 > S_2) & P(S_1 > S_3) \\ P(S_2 > S_1) & P(S_2 > S_2) & P(S_2 > S_3) \\ P(S_3 > S_1) & P(S_3 > S_2) & P(S_3 > S_3) \end{bmatrix}, \tag{6.4}$$

where clearly the diagonal is all zeros. To obtain $\mathbf{P}$, one creates a score matrix C, where each coefficient $c_{i,j}$ is the count of how many times S_i is judged greater than S_j. $\mathbf{P}$ is then obtained by normalising the scores by the total number of observations. Just as in equation (6.3) we can compute the scale value differences

$$H^{-1}(\mathbf{P}) = \mathbf{S} = \begin{bmatrix} S_1 - S_1 & S_1 - S_2 & S_1 - S_3 \\ S_2 - S_1 & S_2 - S_2 & S_2 - S_3 \\ S_3 - S_1 & S_3 - S_2 & S_3 - S_3 \end{bmatrix}. \tag{6.5}$$

Finally, we can easily obtain the scale values if we assume that their mean $\bar{S} = 0$. We

can observe that the sum of the first row of $\mathbf{S}$ divided by the number of samples will be

$$\frac{1}{3}\sum_{i=1}^{3}(S_1 - S_i) = S_1 - \bar{S}. \tag{6.6}$$

Clearly, assuming $\bar{S} = 0$, we obtain directly the scale value S_1. The procedure is exactly the same for each row. Finally, we can compute the 95% confidence interval for the estimated scale values, whose formula for normal distributions is

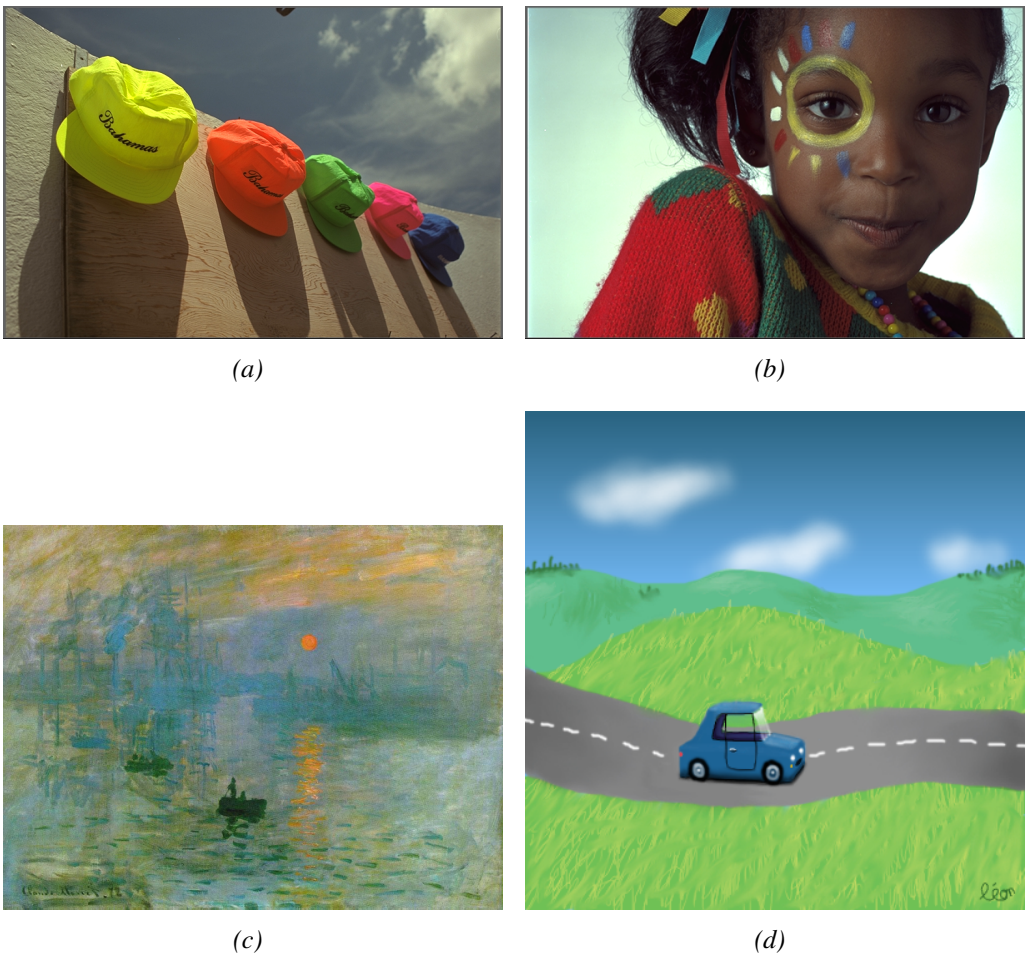
$$S_i - 1.96\frac{\sigma}{\sqrt{n}} \leq \mathbf{E}[S_i] \leq S_i + 1.96\frac{\sigma}{\sqrt{n}} \tag{6.7}$$

(e.g., Devore and Peck, 1986) with n number of observations and $\sigma = \sqrt{2}$, keeping in mind that we are assuming zero mean. For our experiment we refer to the code in the "Colour Engineering Toolbox" by Green and MacDonald (2002), which implements the procedure described in Engeldrum's book with a few modifications (mainly inspired by the Bradley-Terry model, Bradley and Terry, 1952) that handle the cases when $\mathbf{P}$ has also non-diagonal zero coefficients.

6.2 The experiment

The preference experiment is similar to that designed by Connah et al. (2007), whose purpose was to determine which colour to greyscale transformation produces the most pleasing images. Here we are interested in understanding how the reintegration of the colour tensor gradient influences the user preferences, thus we compared only those methods that have been described in chapter 5. That is:

- direct least-squares integration of the colour tensor field (which later we will abbreviate as SW, as in Socolinsky and Wolff's method);
- least-squares integration of the colour tensor field of desaturated images (SWD,



(a) *(b)*

(c) *(d)*

Figure 6.2: Samples of the dataset used in our preference experiment: (a) “hats”, (b) “girl”, (c) “sunrise”, and (d) “voiture”.

i.e., Socolinsky and Wolff’s method on desaturated images);

- least-squares integration of the colour tensor of log-images (SWL, i.e., Socolinsky and Wolff’s method on log-images);
- proposed approach (abbreviated RET as in retinex integration).

To these, we added the comparison with a luminance image (LUM). We assumed that the colour space for the images used in our test is sRGB, thus the luminance channel

A_{lum} is given by

$$A_{\text{lum}} = 0.2126 \times R + 0.7152 \times G + 0.0722 \times B \tag{6.8}$$

after removing the gamma correction (Caldwell et al., 2008). Figure 6.2 shows some of the images in our dataset. They include artwork (Monet's "Impression Sunrise"[1] and "voiture" image[2]) and photographic pictures ("hats" and "girl" from Kodak, 2004). Two more photos were used ("poppies", depicting a field of poppies, and "parrot", portraying a colourful parrot perched on a branch on a background of green vegetation) but we were unable to determine whether anyone owns their copyright and thus they are not shown in this thesis. In our experiment, we tried to avoid using images that present heavy artefacts that make them look absolutely implausible with at least one method (e.g., figure 5.11 on page 113 looks reasonable only with our method and with the NTSC greyscale). While artefacts are certainly important factors to determine whether a method is good or not, we were interested in seeing what people tend to prefer when the images they are shown do not look obviously "wrong".

We evaluated the five methods enumerated above by means of a balanced pairwise comparison, where each greyscale image is compared to all the other greyscale versions obtained from the same colour original. The number of comparisons for each colour image is $N(N-1)/2$ and in this case $N = 5$ gives 10 comparisons for each of the six images, for a total of 60 choices to be performed by the subjects taking part in the experiment. For each of the six images in our dataset five greyscale images were generated, using the same parameters for what concerns the sign of the colour tensor gradient (we used the gradient obtained using ℓ^∞ norm of the channels as reference for the sign) and for the desaturation when needed ($\alpha = 0.5$ in the hsv space). The only exception was the sign for the SWL method. In this case, instead of the ℓ^∞ norm

[1] `http://ibiblio.org/wm/paint/auth/monet/first/impression`
[2] Courtesy of Leon Bli, `http://leonbli.com`.

we used as reference for the sign the average of the channels, i.e., ℓ^1 norm, otherwise images were rendered with obvious integration artefacts. Finally, we applied a linear mapping to shrink the range of images obtained with the SW and SWD methods to the range $[0, 1]$; for SWL instead we applied a logarithmic mapping so that the range of the output log-image would match the range of the input log-image. Each pair of greyscale methods was compared once for each of the images, so each subject had 60 choices to make. The order in which the images and the greyscale methods were presented was random. For each comparison the colour image was shown in the centre of the screen, and two different greyscale versions were displayed side by side of the original. The subjects had to consider the three images and choose which of the greyscale images they preferred. The instruction given were:

> "Consider the colour image and the two greyscale versions. Which greyscale would you prefer as a reproduction of the original?"

The subjects were left to decide themselves the criteria they should apply to make their choice. The experiment was performed by 30 volunteer subjects. All of them declared to have normal colour vision and had normal or corrected to normal acuity. The subjects sat 80 cm away from the screen, and had no constraint on the time available to perform their choices.

The experiment produced a raw score matrix, whose normalisation would produce a matrix $\mathbf{P}$ such as that of equation (6.4). For example, the score matrix C for the "hats" image is in table 6.1 (except for the last column, where we simply report the sum of the rows). To interpret the absolute scores obtained we employ Thurstone's law of comparative judgement, case V, as described in section 6.1. This analysis produces a normalised score and a 95% confidence interval for each of the algorithms. Positive scores mean that the method is generally preferred, while negative scores indicate the opposite. Following the methodology of Connah et al. (2007), by means of Kendall's coefficient of agreement u we can evaluate the agreement among the observers. First of

	LUM	RET	SW	SWD	SWL	Total
LUM	0	11	28	28	28	95
RET	19	0	26	24	30	99
SW	2	4	0	2	24	32
SWD	2	6	28	0	29	65
SWL	2	0	6	1	0	9

Table 6.1: Raw score matrix for the "hats" image.

all, we find the total number of agreements between pairs:

$$\Sigma = \frac{1}{2} \sum_{i \neq j} c_{i,j}(c_{i,j} - 1), \tag{6.9}$$

where $c_{i,j}$ are elements of the score matrix C. Kendall's coefficient of agreement (Kendall and Babington-Smith, 1940) is defined as

$$u = \frac{8\Sigma}{N(N-1)T(T-1)} - 1 \tag{6.10}$$

where N is the number of methods tested (in our case 5) and T is the number of observations, in our case how many people took part in the experiment, i.e., 30. When $u = 1$, the observers have been unanimous, whereas when $u = -1/(T-1)$ (with T even) or $u = -1/T$ (with T odd), the observers have substantially replied randomly (for our experiment, $u = -1/(T-1) = -0.034$). Finally, we can test the null hypothesis that there is no agreement among the subjects by means of the χ^2 test statistics for Kendall's coefficient (e.g., Banterle et al., 2009)

$$\chi^2 = \frac{N(N-1)(1 + u(T-1))}{2} \tag{6.11}$$

with $N(N-1)/2 = 10$ degrees of freedom.

The results are reported in table 6.2. The coefficient u is relatively high for all the images, as the χ^2 test shows being significant at the 0.001 confidence level in all the

images. Some observers, after the experiment, pointed to the "poppies" image as the hardest to evaluate. However, even in this case, where the u coefficient and the χ^2 value are smaller than for other images, the agreement among the observers is strong.

Image	u	χ^2	Confidence
Girl	0.7715	233.73	$p < 0.001$
Hats	0.6074	186.13	$p < 0.001$
Parrot	0.4487	140.13	$p < 0.001$
Poppies	0.1577	55.83	$p < 0.001$
Sunrise	0.2717	88.80	$p < 0.001$
Voiture	0.2492	82.27	$p < 0.001$

Table 6.2: Results of our experiment for each of the six images employed.

In figure 6.3 we show the scale values obtained by applying Thurstone's law. We see that our method (RET) scores significantly higher, indicating that it has been generally preferred over other methods. The SW method seems to score substantially lower than SWD and even LUM. The reason for this seems to be in the choice of the integration constant. The range of the output of the SW method is wider than that of the SWD, therefore the naïve linear compression we applied slightly "washes out" some of the images. A simple histogram equalisation could be employed to reduce this effect, but in that case the whole experiment would be compromised by the introduction of this further factor.

6.3 Conclusion

In this chapter we evaluated the results of the method for image fusion introduced in chapter 5 by means of a balanced paired comparison test. Our method applies retinex to the problem of reintegrating a gradient field. Because of the perceptual foundations of retinex, we expected our method (RET) to generate images that human observers would prefer over other methods. Namely, we compared it with the sRGB luminance chan-

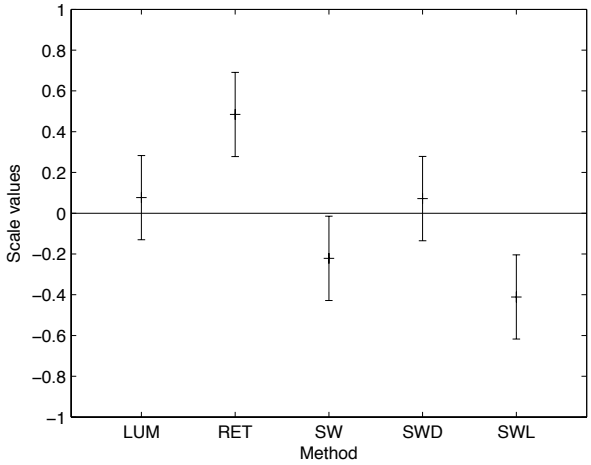


Figure 6.3: Outcome of applying Thurstone's law to preference judgements on our dataset. The x axis shows the five methods tested, while the y axis represents the estimated scale value for each method. Note that the values sum up to zero, thus positive values suggest that the method was preferred and negative values suggest the opposite. The error bars represent 95% confidence interval.

nel (LUM), Socolinsky and Wolff's original method (SW) and two of its variations (on desaturated images, SWD, and on logarithm of images, SWL). Our psychophysical experiment shows that its output is generally judged more pleasant than all other methods. Note that the advantage of SWD over SW is obvious on images where SW generates artefacts. However, as we already pointed out, the advantage in our experiment might be caused by the linear scaling on the image range. On the other hand, RET does not have this problem, since its log-range is clipped to zero directly by retinex, thus the final image range is automatically in $[0, 1]$. The linear mapping might be also the reason why LUM performed so well compared to SW and SWD. Probably it causes a general reduction of contrast, especially in SW, which diminishes the positive effect of the extra detail. LUM, instead, preserves all the luminance contrast of the input. Apart from this, the final outcome of our experiment is just as we expected it, since observers reliably chose our method over the others. Less agreement in the "voiture" and in the "poppies" images might be due to the fact that at least two of the methods generate a very similar

outputs that may confuse observers.

Chapter 7

Conclusion

In this thesis we have investigated the application of path-based computations to three problems in image processing: colour-based image retrieval, image enhancement and data fusion. However, this is not the limit of our contribution. In the process of carrying out this work, we not only applied existing algorithms to these problems, rather we also developed our own and stated various theoretical properties, as summarised in this chapter.

7.1 Contributions

In chapter 3 we applied Lissajous paths to the problem of colour-based image indexing and retrieval. A specific sampling of a certain Lissajous curve gives the Padua points, which are optimal for interpolation in two variables. We then used this tool to approximate chromaticity histograms for the task of colour indexing. The Padua point interpolation yields a linear combination of Chebyshev polynomials, whose coefficients we used for the indexing. Given that the Padua points have a fixed position in their domain, we devised a way of sampling histograms by using some Gaussian weighting. Keeping in mind that our work was intended as explorative, we are more than satisfied

with its performance. Extensive experiments showed not only that our method easily improves the results of the DCT, but also that it matches those of the hybrid transform based on the DCT and PCA.

In chapter 4 we applied pseudo-Brownian motion to image enhancement. Here our main contribution is clearly the algorithm that generates pseudo-random paths. We provided a formal definition based on graphs and proved the following properties:

1. Given an integer number k, the path will visit every pixel of the image at least k times.

2. On average, the path will visit every pixel $2k$ times.

Moreover, we also analysed the statistical properties of the paths it yields to compare them with the properties of true random walk and Brownian motion. For the random walk, we also devised a computational method to determine the probability $P(n, d)$ of the walker being at a distance n from the origin after d steps. An explicit formulation for this value is known in one dimension but, to the best of our knowledge, there is no equivalent in multiple dimensions. We found the following similarities:

1. When used to generate a pseudo-random walk, the value $\tilde{P}(n, d)$ measured from the path obtained with our algorithm and the theoretical value $P(n, d)$ of the random walk are very close.

2. The measured average distance from the origin follows the theoretical prediction of $\sqrt{n}$.

3. In pseudo-Brownian paths, the displacements at each step are random and have similar distribution to those of the real Brownian motion.

Finally, we proposed a retinex implementation based on our path generator and showed that it is efficient and its results are better than those of traditional multi-scale methods.

In chapter 5 we proposed a method for data fusion based on the colour tensor. Our contribution here is manifold. First of all, the colour tensor yields a single gradient field that comprises the contrast information from all the channels of an image, regardless of number. This gradient is, in general, non-integrable. We found that reducing the saturation (colour saturation or its multichannel extension that we defined) in the input image is a simple way to improve the integrability of the colour tensor gradient, and we provided the supporting empirical data. However, we could not prove this property. Yet, in our attempt at doing so we proved various properties of the effect of desaturation on the colour tensor.

1. Both eigenvalues of the colour tensor shrink as the saturation decreases.

2. The determinant of the colour tensor shrinks as the saturation decreases. This clearly follows on from point 1, since the determinant is equal to the product of the eigenvalues. However, in this case we could provide an explicit formula for the determinant as a function of the saturation factor α.

3. When the determinant of the colour tensor is zero both in the image and its 180-degree rotated version, the gradient is integrable.

During this process, we also brought a counterexample to the basic property we wanted to prove. We found that in certain rare cases, desaturation increases the integrability error of the gradient. Even though this happens very infrequently, it means that, in general, it is not possible to prove that desaturation always decreases the integrability error.

The second major contribution of chapter 5 is in the reintegration step. Once the colour tensor gradient is computed, it can be visualised by means of reintegration, obtained by solving a PDE. In the literature, PDEs have often been applied to retinex. Here we did the opposite, showing that retinex can be used to solve a PDE with impressive results. The combination of integrability improvement and retinex reintegration results

in a powerful data fusion method, whose performance was tested on human observers. In chapter 6 we provided the details of our psychophysical experiment, which showed the validity of our method.

7.2 Future work

Our contributions are not definitive at all and there is space for further investigation. Our Lissajous-based image retrieval method works very well, however we noticed that the standard deviation σ of the Gaussian weighting affects its performance heavily. We believe that more research would be necessary to understand this relation, so that in the future we will be able to provide an optimal choice of σ.

Our random-path generator has been applied to image enhancement. Probably, the next step it needs is an efficient implementation. Ours is at prototype level and certainly could be substantially improved. Apart from this, it would be interesting to find other applications as well. An example could be the sieve algorithm, which already proved to work well with random Hamiltonian paths (Fredembach and Finlayson, 2008). Additionally, since our algorithm has a general definition based on graphs, it might have applications in fields yet to be considered.

Finally, in chapter 5 we analysed several properties of the colour tensor. In our approach to reduce the integrability error, we did not consider the sign of the colour tensor gradient, which is intrinsically undefined. This is a known open problem that various authors have tried to address (e.g., Drew et al., 2009), so far without a final answer. The way we approach this is to apply to the colour tensor gradient the sign from an integrable gradient, e.g., the gradient from the average of the channels, or from the ℓ^2-norm or ℓ^∞-norm of the channels. However, all these are suboptimal choices because they can work perfectly in some images and very poorly in others, making it impractical to stick to any of them rigidly. Even though this issue is not directly

dependent on our own work, it fits perfectly with what we think is the main problem of our method. Ideally, we would like our method to be fully automatic: the user provides an image and a greyscale rendering is returned. For now, the sign of the gradient is only the first of various choices a user would have to make, the others being the type of desaturation (HSV or the one we defined from RGB), the level of desaturation and the number of retinex iterations for the reintegration.

Appendix A

The Kodak dataset

In this appendix we show all the images contained in the Kodak dataset (Kodak, 2004), which was mentioned several times throughout this thesis. We show them in their original form, processed with the McCann99 retinex algorithm (see Funt et al., 2004; McCann, 1999), which we detail in section 4.1.2 on page 58, and with the pseudo-Brownian path retinex that we propose in chapter 4. Both algorithms divide each image in the same number of scales and perform on average 32 comparisons per pixel.

(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

(j) *(k)* *(l)*

Figure A.1: Original images in the first column; processed with the McCann99 retinex in the second column; processed with our pseudo-Brownian motion retinex in the third column. Observe the colour shift (towards green) introduced by the McCann99 in (e). The same colour shift is not visible in the image produced by our algorithm (f).

(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

(j) *(k)* *(l)*

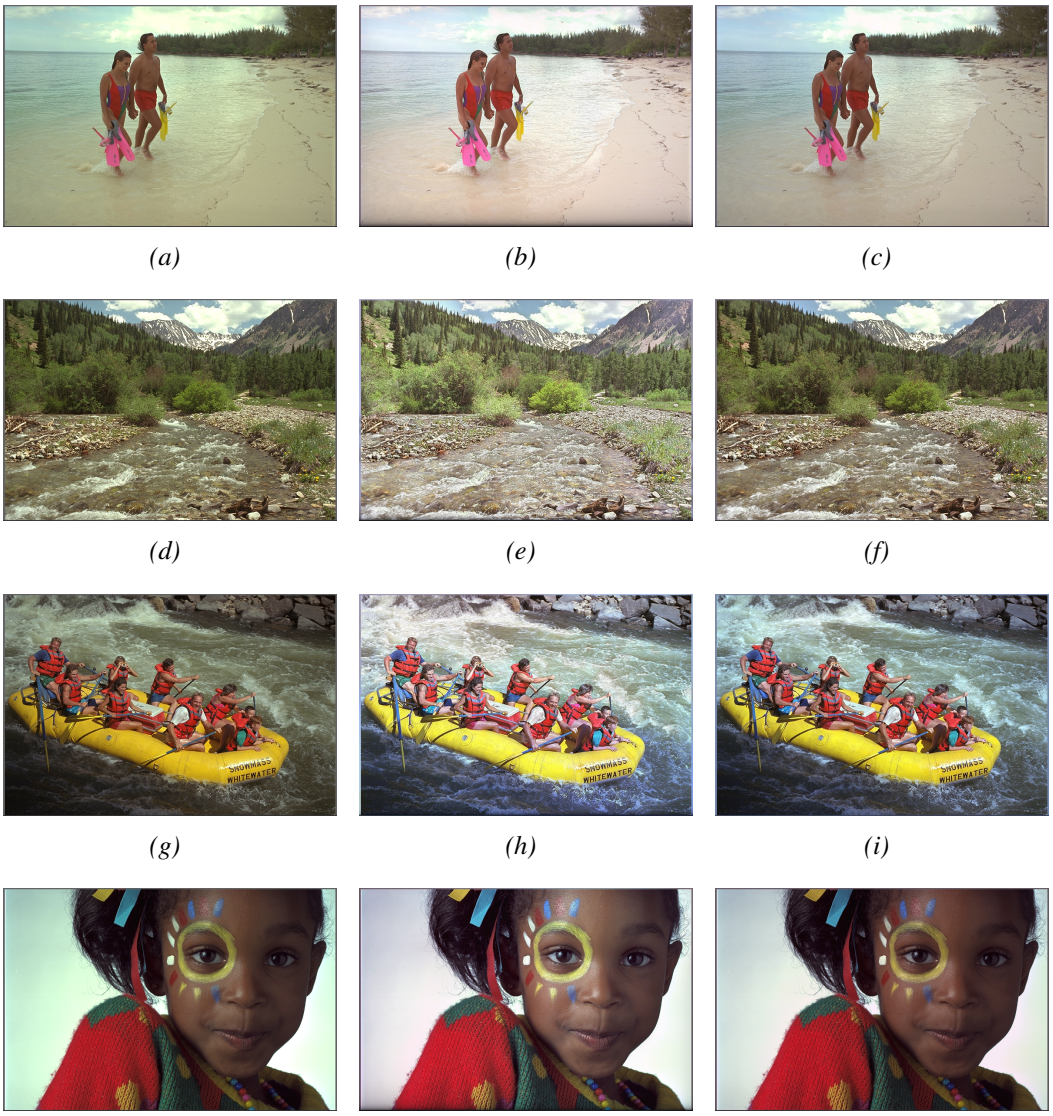
Figure A.2: Original images in the first column; processed with the McCann99 retinex in the second column; processed with our pseudo-Brownian motion retinex in the third column. Note in (e) and (h) a dark smearing at the bottom of the image, caused by the presence of a thin dark frame at the boundary of the image. This artefact is not present in the corresponding images (f) and (i) generated with our algorithm.

(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

Figure A.3: Original images in the first column; processed with the McCann99 retinex in the second column; processed with our pseudo-Brownian motion retinex in the third column. Like in figure A.2, the thin dark frame at the boundary of images (b) and (e) (this time on their right) is smeared, while our retinex implementation does not cause this artefact in (c) and (f).



(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

(j) *(k)* *(l)*

Figure A.4: Original images in the first column; processed with the McCann99 retinex in the second column; processed with our pseudo-Brownian motion retinex in the third column.

(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

Figure A.5: Original images in the first column; processed with the McCann99 retinex in the second column; processed with our pseudo-Brownian motion retinex in the third column. Note in (b) and (c) that both retinex implementations remove the yellowish colour cast that is present in the original image (a).

(a) *(b)* *(c)*

(d) *(e)* *(f)*

(g) *(h)* *(i)*

(j) *(k)* *(l)*

Figure A.6: Original images in the first column; processed with the McCann99 retinex in the second column; processed with our pseudo-Brownian motion retinex in the third column.

(a) *(b)* *(c)*

(d) *(e)* *(f)*

Figure A.7: Original images in the first column; processed with the McCann99 retinex in the second column; processed with our pseudo-Brownian motion retinex in the third column.

References

Agrawal, A., Chellappa, R., and Raskar, R., 2005, "An algebraic approach to surface reconstruction from gradient fields," in *Proceedings of Tenth IEEE International Conference on Computer Vision (ICCV)*, vol. 1, pp. 174–181.

Agrawal, A., Raskar, R., and Chellappa, R., 2006, "Edge suppression by gradient field transformation using cross-projection tensors," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2301–2308, IEEE Computer Society.

Ansari, A. and Fineberg, A., 1992, "Image data ordering and compression using Peano scan and LOT," *IEEE T. Consum. Electr.*, vol. 38, no. 3, pp. 436–445.

Bangham, J. A., Chardaire, P., Pye, C. J., and Ling, P. D., 1996a, "Multiscale nonlinear decomposition: the sieve decomposition theorem," *IEEE T. Pattern Anal.*, vol. 18, no. 5, pp. 529–539.

Bangham, J. A., Harvey, R. W., Ling, P. D., and Aldridge, R. V., 1996b, "Morphological scale-space preserving transforms in many dimensions," *J. Electron. Imaging*, vol. 5, no. 3, pp. 283–299.

Banterle, F., Ledda, P., Debattista, K., Bloj, M., Artusi, A., and Chalmers, A., 2009, "A psychophysical comparison of inverse tone mapping techniques," *Comput. Graph. Forum*, vol. 28, no. 1, pp. 13–25.

Barnard, K. and Funt, B. V., 1997, "Analysis and improvement of multi-scale retinex," in *Proceedings of Fifth Color Imaging Conference*, pp. 221–226, IS&T and SID.

Berens, J., Finlayson, G. D., and Qiu, G., 2000, "Image indexing using compressed colour histograms," *IEE Proc.—Vis. Image Signal Process.*, vol. 147, no. 4, pp. 349–354.

Bogle, M. G. V., Hearst, J. E., Jones, V. F. R., and Stoilov, L., 1994, "Lissajous knots," *J. Knot Theor. Ramif.*, vol. 3, no. 2, pp. 121–140.

Borenstein, E., 1999, "Lightness and brightness computations by Retinex-like algorithms," Master's thesis, Feinberg Graduate School, Weizmann Institute of Science, Rehovot, Israel.

Bos, L., Caliari, M., De Marchi, S., Vianello, M., and Xu, Y., 2006, "Bivariate Lagrange interpolation at the Padua points: the generating curve approach," *J. Approx. Theory*, vol. 143, no. 1, pp. 15–25.

Bos, L., De Marchi, S., Vianello, M., and Xu, Y., 2007, "Bivariate Lagrange interpolation at the Padua points: the ideal theory approach," *Numer. Math.*, vol. 108, no. 1, pp. 43–57.

Bradley, R. A. and Terry, M. E., 1952, "Rank analysis of incomplete block designs, I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3–4, pp. 324–345.

Brainard, D. H. and Wandell, B. A., 1986, "Analysis of the retinex theory of color vision," *J. Opt. Soc. Am.*, vol. 3, no. 10, pp. 1651–1661.

Brutman, L., 1997, "Lebesgue functions for polynomial interpolation — a survey," *Ann. Numer. Math.*, vol. 4, pp. 111–128.

Butz, A. R., 1968, "Space filling curves and mathematical programming," *Information and Control*, vol. 12, no. 4, pp. 314–330.

Caldwell, B., Cooper, M., Guarino Reid, L., and Vanderheiden, G., 2008, "Web Content Accessibility Guidelines (WCAG) 2.0," `http://www.w3.org/TR/WCAG20/Overview.html`, accessed 12 October 2010.

Caliari, M., De Marchi, S., Montagna, R., and Vianello, M., 2007, "XuPad2D," `http://www.math.unipd.it/~marcov/CAAsoft.html`, accessed 23 October 2010.

Caliari, M., De Marchi, S., Sommariva, A., and Vianello, M., 2011, "Padua2DM: fast interpolation and cubature at the Padua points in Matlab/Octave," *Numer. Algorithms*, vol. 56, no. 1, pp. 45–60.

Caliari, M., De Marchi, S., and Vianello, M., 2005, "Bivariate polynomial interpolation on the square at new nodal sets," *Appl. Math. Comput.*, vol. 165, no. 2, pp. 261–274.

Caliari, M., De Marchi, S., and Vianello, M., 2008, "Algorithm 886: Padua2D – Lagrange interpolation at Padua points on bivariate domains," *ACM T. Math. Software*, vol. 35, no. 3, pp. 1–11.

Chavez, P. S., Jr., 1988, "An improved dark-object subtraction technique for atmospheric scattering correction of multispectral data," *Remote Sens. Environ.*, vol. 24, no. 3, pp. 459–479.

Chen, S. D., Shen, H., and Topor, R., 2002, "An efficient algorithm for constructing Hamiltonian paths in meshes," *Parallel Comput.*, vol. 28, no. 9, pp. 1293–1305.

Cheney, W. and Kincaid, D., 2004, *Numerical Mathematics and Computing*, chap. 17, pp. 691–719, Thomson Brook/Cole, 5th ed.

CompuServe Inc., 1990, "Graphic Interchange Format," `http://www.w3.org/Graphics/GIF/spec-gif89a.txt`, accessed 19 November 2010.

Connah, D., Finlayson, G. D., and Bloj, M., 2007, "Seeing beyond luminance: a psychophysical comparison of techniques for converting colour images to greyscale," in *Proceedings of Fifteenth Color Imaging Conference*, pp. 336–340, IS&T and SID.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., 2001, *Introduction to algorithms*, The MIT Press, 2nd ed.

Cumani, A., 1991, "Edge detection in multispectral images," *CVGIP: Graphical Models and Image Processing*, vol. 53, no. 1, pp. 40–51.

Cundy, H. M. and Rollett, A. P., 1961, *Mathematical models*, Oxford University Press, 2nd ed.

Dafner, R., Cohen-Or, D., and Matias, Y., 2000, "Context-based space filling curves," *Comput. Graph. Forum*, vol. 19, no. 3, pp. 209–218.

Devore, J. and Peck, R., 1986, *Statistics, the exploration and analysis of data*, West Publishing Company.

Di Zenzo, S., 1986, "A note on the gradient of a multi-image," *Comput. Vision Graph.*, vol. 33, no. 1, pp. 116–125.

Drew, M. S., Connah, D., Finlayson, G. D., and Bloj, M., 2009, "Improved colour to greyscale via integrability correction," in *Proceedings of Electronic Imaging*, IS&T and SPIE.

Durret, R., 1984, *Brownian motion and martingales in analysis*, Wadsworth Advanced Books & Software.

Engbert, R. and Kliegl, R., 2004, "Microsaccades keep the eyes' balance during fixation," *Psychological Science*, vol. 15, no. 6, pp. 431–436.

Engeldrum, P. G., 2000, *Psychometric scaling – A toolkit for imaging systems development*, Imcotek Press, Winchester, MA, USA.

European Space Agency, 2009, "Herschel: orbit/navigation," `http://sci.esa.int/science-e/www/object/index.cfm?fobjectid=34699`, accessed 17 December 2010.

Fairchild, M. D., 2010, "Still photography throwdown: silver halide vs. silicon," in *Proceedings of 18th Color and Imaging Conference*, pp. 154–159, IS&T and SID.

Falconer, K., 2003, *Fractal geometry – Mathematical foundations and applications*, John Wiley & Sons Inc., 2nd ed.

Fattal, R., Lischinski, D., and Werman, M., 2002, "Gradient Domain High Dynamic Range Compression," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 249–256.

Finlayson, G. D., Hordley, S. D., and Drew, M. S., 2002, "Removing shadows from images using retinex," in *Proceedings of Tenth Color Imaging Conference*, pp. 73–79, IS&T and SID.

Finlayson, G. D., Hordley, S. D., Lu, C., and Drew, M. S., 2006, "On the removal of shadows from images," *IEEE T. Pattern Anal.*, vol. 28, no. 1, pp. 59–68.

Frankle, J. A. and McCann, J. J., 1983, "Method and apparatus for lightness imaging," U.S. Patent No. 4,384,336.

Fredembach, C., 2006, "Estimating, detecting and removing illumination in images," Ph.D. thesis, School of Computing Sciences, University of East Anglia, Norwich, UK.

Fredembach, C. and Finlayson, G. D., 2005, "Hamiltonian path-based shadow removal," in *Proceedings of 16th British Machine Vision Conference*, vol. 2, pp. 502–511, BMVA.

Fredembach, C. and Finlayson, G. D., 2008, "The 1.5D sieve algorithm," *Pattern Recogn. Lett.*, vol. 29, no. 5, pp. 629–636.

Fredembach, C. and Süsstrunk, S., 2008a, "Colouring the near-infrared," in *Proceedings of Sixteenth Color Imaging Conference*, pp. 176–182, IS&T and SID.

Fredembach, C. and Süsstrunk, S., 2008b, "Colouring the near-infrared: supplementary material," `http://ivrg.epfl.ch/supplementary_material/FS_CIC08/index.html`, accessed 11 January 2011.

Freedman, D., 1971, *Brownian motion and diffusion*, Holden-Day.

Frigo, M. and Johnson, S. G., 2005, "The design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216–231.

Funt, B. V., Ciurea, F., and McCann, J. J., 2000, "Retinex in Matlab," in *Proc. 8th Color Imaging Conference*, pp. 112–121, IS&T and SID.

Funt, B. V., Ciurea, F., and McCann, J. J., 2004, "Retinex in MATLAB™," *J. Electron. Imaging*, vol. 13, no. 1, pp. 48–57.

Gadia, D., Rizzi, A., and Marini, D., 2004, "Tuning Retinex for HDR images visualization," in *Proceedings of Second European Conference on Color in Graphics, Imaging and Vision*, pp. 326–331, IS&T.

Geusebroek, J. M., Burghouts, G. J., and Smeulders, A. W. M., 2005, "The Amsterdam Library of Object Images," *Int. J. Comput. Vision*, vol. 61, no. 1, pp. 103–112.

Gonzalez, R. C., Woods, R. E., and Eddins, S. L., 2004, *Digital image processing using Matlab*, Pearson Prentice Hall.

Gooch, A. A., Olsen, S. C., Tumblin, J., and Gooch, B., 2005, "Color2gray: Salience-preserving color removal," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 634–639.

Gould, R., 1988, *Graph theory*, The Benjamin/Cummins Publishing Company Inc.

Green, P. and MacDonald, L., eds., 2002, *Colour Engineering: Achieving Device Independent Colour*, John Wiley & Sons Inc.

Greengard, L. and Strain, J., 1991, "The Fast Gauss Transform," *SIAM J. Sci. Stat. Comp.*, vol. 12, no. 1, pp. 79–94.

Hilbert, D., 1891, "Über die stetige Abbildung einer Line auf ein Flächenstück," *Math. Ann.*, vol. 38, no. 3, pp. 459–460.

Horn, B. K. P., 1974, "Determining lightness from an image," *Comput. Vision Graph.*, vol. 3, pp. 277–299.

Horn, R. A. and Johnson, C. R., 1985, *Matrix analysis*, Cambridge University Press.

Hunt, R. W. G., 1998, *Measuring colour*, Fountain Press, Kingston-upon-Thames, England, 3rd ed.

International Organisation for Standardisation, 2004, "MPEG-7 Overview," `http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm`, accessed 29 November 2010.

Jain, R., Kasturi, R., and Schunck, B. G., 2001, *Machine vision*, McGraw-Hill, Inc.

Jobson, D. J., Rahman, Z., and Woodell, G. A., 2003, "Feature visibility limits in the nonlinear enhancement of turbid images," in *Proceedings of SPIE*, vol. 5108.

Karger, D. R., Klein, P. N., and Tarjan, R. E., 1995, "A randomized linear-time algorithm to find minimum spanning trees," *J. ACM*, vol. 42, no. 2, pp. 321–328.

Kendall, M. G. and Babington-Smith, B., 1940, "On the method of paired comparisons," *Biometrika*, vol. 31, no. 3–4, pp. 324–345.

Kodak, 2004, "Kodak Lossless True Color Image Suite," `http://r0k.us/graphics/kodak/`, accessed 23 October 2010.

Land, E. H., 1964, "The retinex," *Am. Scientist*, vol. 52, pp. 247–264.

Land, E. H., 1977, "The retinex theory of color vision," *Sci. Am.*, vol. 237, no. 6, pp. 108–128.

Land, E. H. and McCann, J. J., 1971, "Lightness and retinex theory," *J. Opt. Soc. Am.*, vol. 61, no. 1, pp. 1–11.

Lawler, G. F. and Coyle, L. N., 2000, *Lectures on Contemporary Probability*, Student Mathematical Library.

Lo, M. W., Williams, B., Bollman, W. E., Han, D., Corwin, R. A., Hong, P. E., Howell, K. C., Barden, B., and Wilson, R., 1998, "Genesis mission design," in *AAS/AAIA Astrodynamics Specialist Conference*, Boston, MA (USA).

Mandelbrot, B. B., 1983, *The fractal geometry of nature*, W. H. Freeman and Co.

Marini, D. and Rizzi, A., 2000, "A computational approach to color adaptation effects," *Image Vision Comput.*, vol. 18, no. 13, pp. 1005–1014.

Marsaglia, G. and Tsang, W. W., 2000, "The Ziggurat Method for Generating Random Variables," *J. Stat. Softw.*, vol. 5, no. 8, pp. 1–7.

Mathews, J. H. and Fink, K. D., 1999, *Numerical Methods Using Matlab*, chap. 4, pp. 236–237, Prentice Hall.

McCann, J. J., 1999, "Lessons learned from Mondrians applied to real images and color gamuts," in *Proceedings of Senventh Color Imaging Conference*, pp. 1–8, IS&T and SID.

Meylan, L. and Süsstrunk, S., 2006, "High dynamic range image rendering with a retinex-based adaptive filter," *IEEE T. Image Process.*, vol. 15, no. 6, pp. 2820–2830.

MIT Lincoln Laboratory, 2010, "Lincoln Laboratory Logo," `http://www.ll.mit.edu/about/History/logo.html`, accessed 26 November 2010.

Montagna, R., 2007, "Hyperinterpolation at Xu points and interpolation at Padua points in the square: computational aspects," Master's thesis, Dipartimento di Informatica, Università degli Studi di Verona, Italy.

Moon, T. K. and Stirling, W. C., 2000, *Mathematical methods and algorithms for signal processing*, chap. 3, pp. 183–185, Prentice Hall.

Morel, J. M., Petro, A. B., and Sbert, C., 2009, "Fast implementation of color constancy algorithms," in *Proceedings of Electronic Imaging*, IS&T and SPIE.

Morel, J. M., Petro, A. B., and Sbert, C., 2010, "A PDE formalization of retinex theory," *IEEE T. Image Process.*, vol. 19, no. 11, pp. 2825–2837.

Moriguchi, H., Wendt, M., and Duerk, J. L., 2000, "Applying the uniform resampling (URS) algorithm to a Lissajous trajectory: fast image reconstruction with optimal gridding," *Magn. Reson. Med.*, vol. 44, no. 5, pp. 766–781.

Müller, H., Müller, W., Squire, D. M., Marchand-Maillet, S., and Pun, T., 2001, "Performance evaluation in content-based image retrieval: overview and proposals," *Pattern Recogn. Lett.*, vol. 22, pp. 593–601.

Nene, S. A., Najar, S. K., and Murase, H., 1996, "Columbia Object Image Library (COIL-100)," Tech. rep., Columbia University, New York (USA).

Olmos, A. and Kingdom, F. A. A., 2004, "McGill Calibrated Colour Image Database," `http://tabby.vision.mcgill.ca`, accessed 16 December 2010.

Peano, G., 1890, "Sur une courbe, qui remplit toute une aire plane," *Math. Ann.*, vol. 36, no. 1, pp. 157–160.

Piella, G., 2009, "Image fusion for enhanced visualization: a variational approach," *Int. J. Comput. Vision*, vol. 83, no. 1, pp. 1–11.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P., 2007, *Numerical Recipes*, Cambridge University Press, 3rd ed.

Provenzi, E., Fierro, M., Rizzi, A., De Carli, L., Gadia, D., and Marini, D., 2007, "Random Spray Retinex: a new retinex implementation to investigate the local properties of the model," *IEEE T. Image Process.*, vol. 16, no. 1, pp. 162–171.

Rahman, Z., Jobson, D. J., and Woodell, G. A., 2004, "Retinex processing for automatic image enhancement," *J. Electron. Imaging*, vol. 13, no. 1, pp. 100–110.

Raol, J. R., 2010, *Multi-Sensor Data Fusion with MATLAB*, chap. 10, CRC Press.

Rasche, K., Geist, R., and Westall, J., 2005, "Detail preserving reproduction of color images for monochromats and dichromats," *IEEE Comput. Graph. Appl.*, vol. 25, no. 3, pp. 22–30.

Rudnick, J. and Gaspari, G., 2004, *Elements of the Random Walk*, Cambridge University Press.

Scarpelli, J. B., 1978, "Apparatus for video display of audio frequency patterns," U.S. Patent No. 4,176,375.

Schroeder, J., Yarlagadda, R., and Hershey, J., 1991, "L_p normed minimization with applications to linear predictive modeling for sinusoidal frequency estimation," *Signal Process.*, vol. 24, no. 2, pp. 193–216.

Shelhamer, M., 2005, "Sequence of predictive saccades are correlated over a span of $\sim$2 s and produce a fractal time series," *J. Neurophysiol.*, vol. 93, no. 4, pp. 2002–2011.

Smith, K., Landes, P. E., Thollot, J., and Myszkowski, K., 2008, "Apparent Greyscale: A Simple and Fast Conversion to Perceptually Accurate Images and Video," *Comput. Graph. Forum*, vol. 27, no. 2, pp. 193–200.

Sobol, R., 2004, "Improving the Retinex algorithm for rendering wide dynamic range photographs," *J. Electron. Imaging*, vol. 13, no. 1, pp. 65–74.

Socolinsky, D. A. and Wolff, L. B., 2002, "Multispectral image visualization through first order fusion," *IEEE T. Image Process.*, vol. 11, no. 8, pp. 923–931.

Southam, P., 2006, "Texture analysis with the sieve," Ph.D. thesis, School of Computing Sciences, University of East Anglia, Norwich, UK.

State Library of New South Wales, 2008, "Hurley's Antarctica," `http://www.sl.nsw.gov.au/discover_collections/natural_world/antarctica/hurley/index.html`, accessed 26 January 2011.

State Library of New South Wales, 2009, "Frank Hurley colour Paget plates of Ernest Shackleton's 'Endurance' expedition to Antarctica 1915," `http://www.flickr.com/photos/statelibraryofnsw/sets/72157618020442474/`, accessed 26 January 2011.

Stockham, T. G., 1972, "Image processing in the context of a visual model," *Proc. IEEE*, vol. 60, no. 7, pp. 828–842.

Swain, M. J. and Ballard, D. H., 1991, "Color indexing," *Int. J. Comput. Vision*, vol. 7, no. 1, pp. 11–32.

Thurstone, L. L., 1927, "A law of comparative judgement," *Psychol. Rev.*, vol. 34, no. 4, pp. 273–286.

U.S. Geological Survey, 2010, "Landsat Satellite and Sensor Information," `http://landsat.usgs.gov/Satellite_and_Sensor_Information.php`, accessed 26 May 2011.

U.S. Geological Survey, 2011, "Landsat missions," `http://landsat.usgs.gov`, accessed 26 May 2011.

Van de Weijer, J., Gevers, T., and Smeulders, A. W. M., 2006, "Robust photometric invariant features from the color tensor," *IEEE T. Image Process.*, vol. 15, no. 1, pp. 118–127.

Wandell, B. A., 1995, *Foundations of Vision*, Sinauer Associates, Inc.

Welch, T. A., 1984, "A technique for high-performance data compression," *Computer*, vol. 17, no. 6, pp. 8–19.

Woodell, G. A., Jobson, D. J., Rahman, Z., and Hines, G., 2005, "Enhancement of imagery in poor visibility conditions," in *Proceedings of SPIE*, vol. 5778.

Zeki, S., 1993, *A vision of the brain*, Blackwell Science Ltd.

Ziv, J. and Lempel, A., 1978, "Compression of individual sequences via variable-rate coding," *IEEE T. Inform. Theory*, vol. 24, no. 5, pp. 530–536.