Colour and Texture Image Analysis in a Local Binary Pattern Framework

by

Seth Winston Stewart Nixon

Submitted for the degree of Doctor of Philosophy

UNIVERSITY OF EAST ANGLIA

School of Computing Sciences

September 2019

Abstract

In this Thesis we use colour and Local Binary Pattern based texture analysis for image classification and reconstruction [redacted] In complementary work we offer a new texture description called the Sudoku transform, an extension of the Local Binary Pattern. Our new method when used to classify members of benchmark datasets shows a performance increment over traditional methods including the Local Binary Pattern. Finally we consider the invertibility of texture descriptions and show how with our new method - Quadratic Reconstruction - that a highly accurate image can be recovered purely from its textural information.

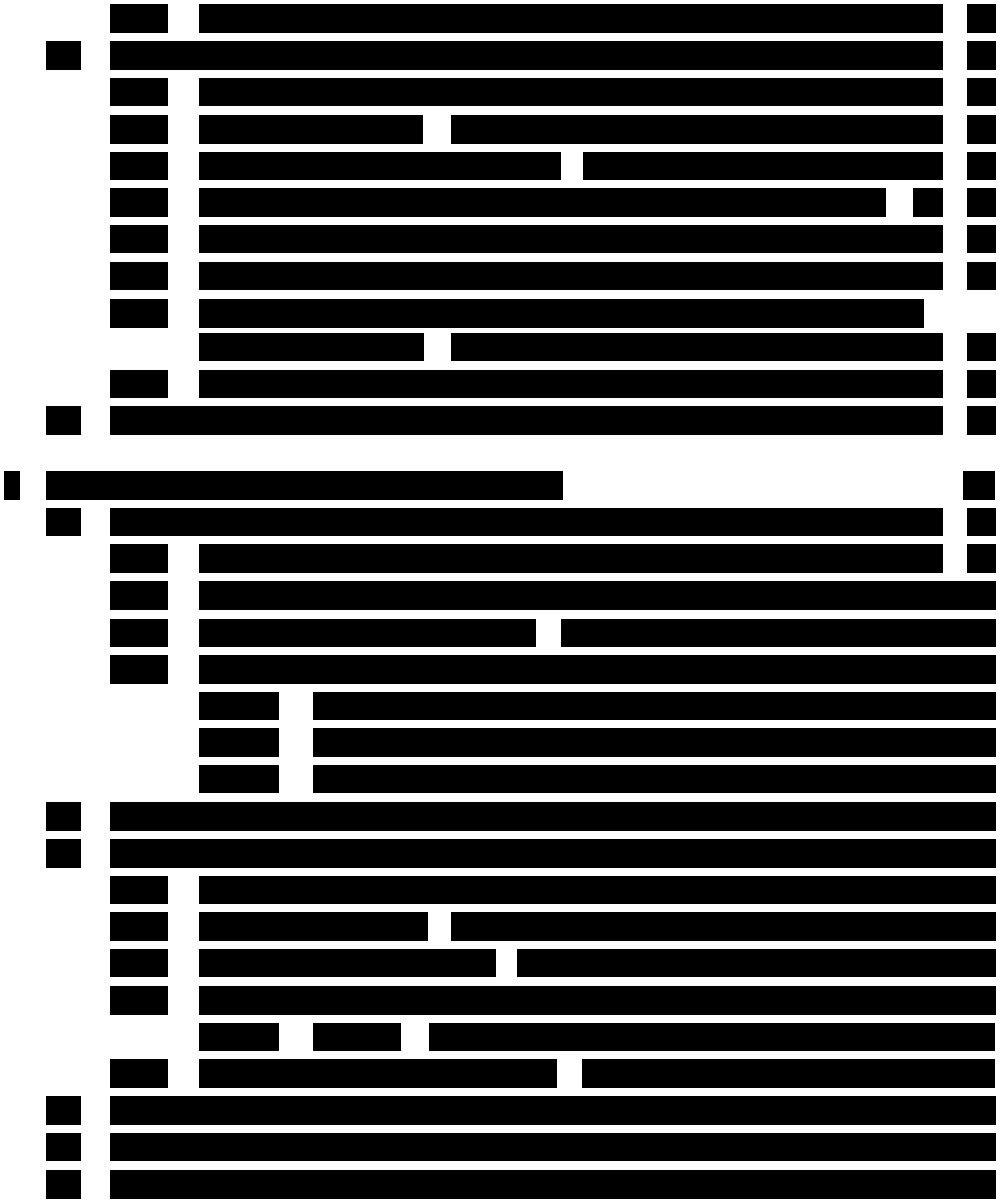
Acknowledgements

I would like to sincerely thank my supervisor Professor Graham Finlayson for all of his hard work and help. Especially I would like to thank him for his unwavering belief in me throughout.

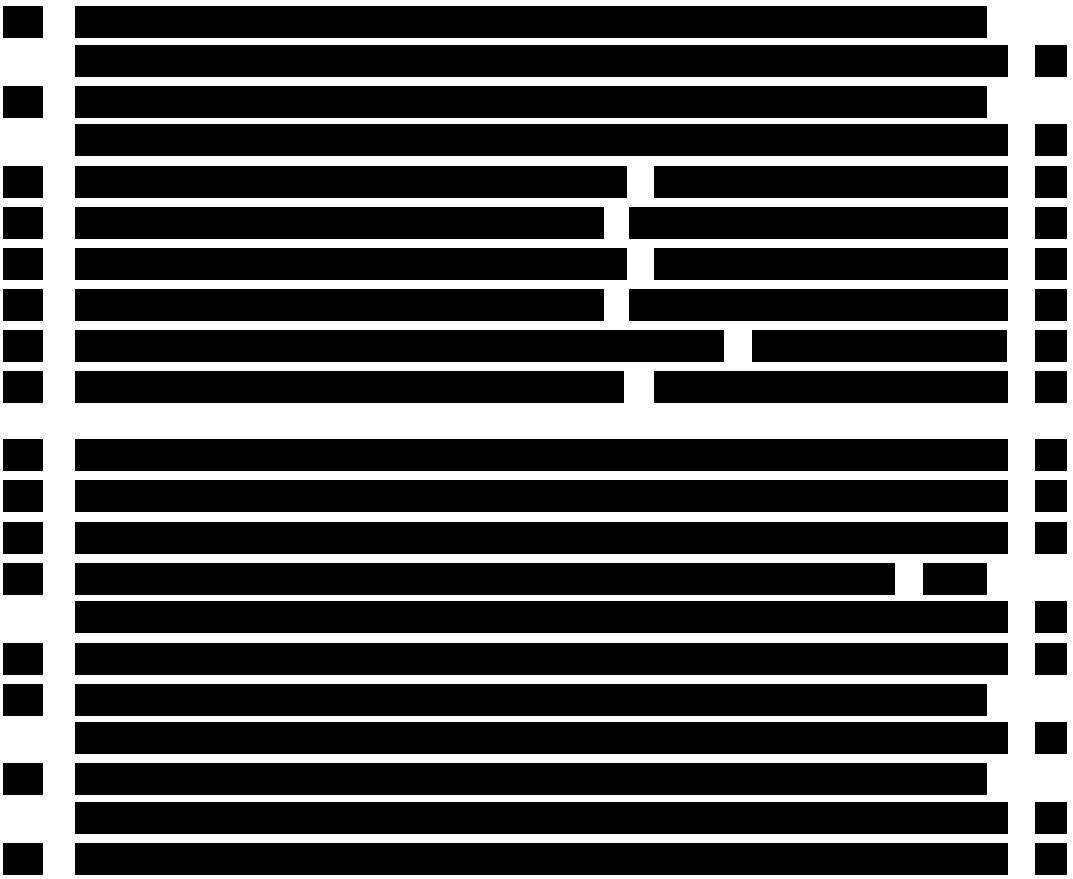
[redacted]

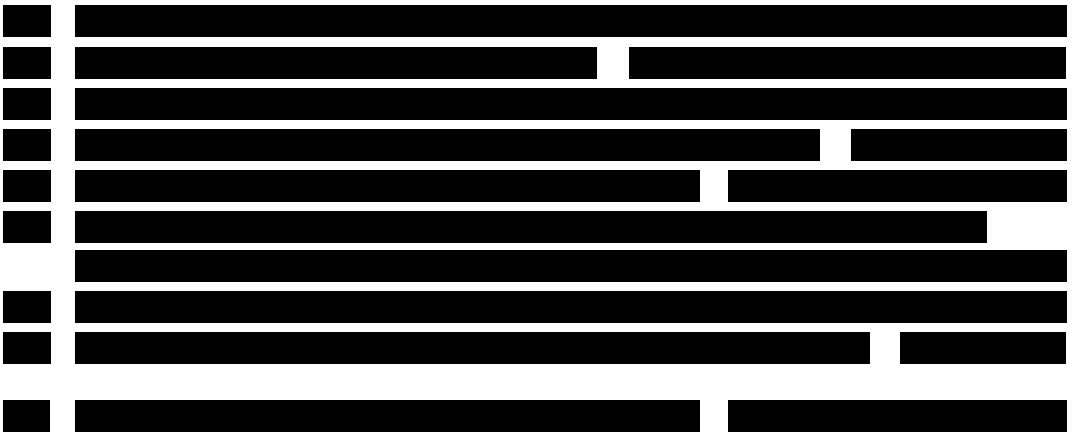
Finally I would like to thank my family for their eternal support, especially throughout this degree.

Contents



List of Figures

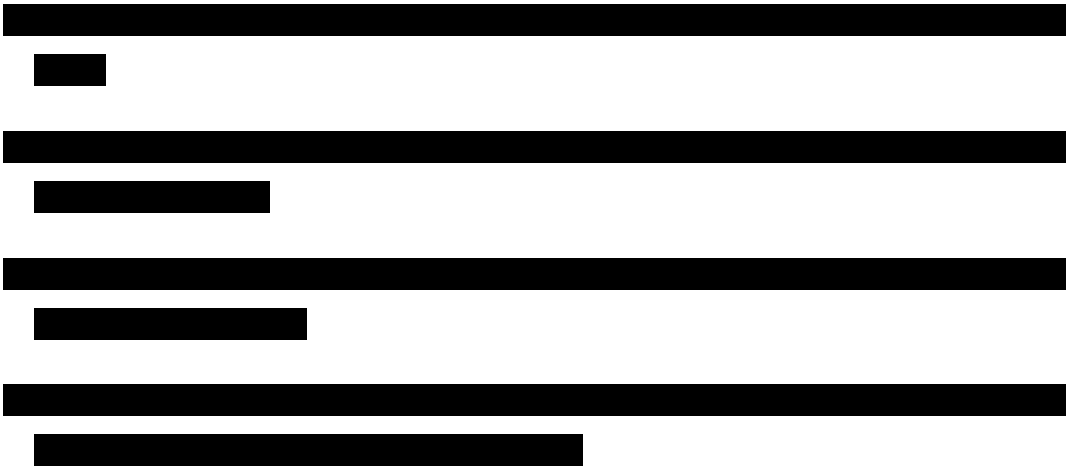




List of Tables

Publications

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



5 Finlayson, Graham, and Seth Nixon. "Sudoku texture classification." *Electronic Imaging*, 2016.

6 Finlayson, Graham, and Seth Nixon "Reconstructing a colour image from its texture" *Color and Imaging Conference*, Accepted for oral presentation in Nov 2018.

7 Nixon, Seth, and Graham Finlayson "The Sudoku texture representation", In preparation, *Image and Vision Computing*

Chapter 1

Context and contributions

1.1 Image analysis

This thesis focuses on image analysis using methods from two primary areas. In Section 1.1.1 we will introduce the area of texture analysis. In Section 1.1.2 we will introduce colour in the context of image classification.

1.1.1 Texture analysis

Texture is the feel or appearance of a surface. It can be considered heuristically using terms such as "rough" or "smooth". It can also be considered visually as the pattern(s) or lack thereof that a surface exhibits. Texture is widely considered to be a feature of a region, not a point [2]. The interaction between different elements in that area is what forms the full appearance or the texture. In digital imaging these elements become pixels, singularly or in groups, and analyses of these has proven useful in a wide range of areas including fingerprint recognition [3], facial expression recognition [4] and image segmentation [65]. In this thesis we consider texture as a feature, using it primarily for classification.

Texture analysis hinges on the assumption that it can be quantified and/or understood using computational methods. It has a long history and many techniques have been used including Gabor filters [54], Local Binary Patterns (LBP) [5] and Basic Image Features (BIFs) [6]. There are two primary schools of thought as to how texture is best quantified. "Global" methods consider texture as a property of an image as a whole. "Local" methods suggest that texture is a feature of local areas of an image. Sampling strategy is also a key. "Dense" sampling methods consider textural information at every point in the image. "Sparse" sampling methods only consider especially *salient* regions in an image. Densely computed histograms of local features such as LBP have evidenced benchmark performance in many areas such as face recognition [7]. The Scale Invariant Feature Transform (SIFT) is a further local method [8] which samples the image Sparsely, forming local descriptions of the selected points. It is used in areas such as fingerprint recognition [9] and medical imaging [10]. [redacted]

A significant problem in the area of texture is that no standard description of it exists and no single method works best in all conditions. Methods have been proposed which attempt to organise how we classify texture. A major contribution to this was [11], where the authors proposed that texture classification can be split into four categories: statistical properties, mathematical models, geometric methods and signal processing methods. This work was then extended by [12] where they define a taxonomy of texture which also considers colour. One particular description of a set of texture classification methods is the Histograms of Equivalent Patterns (HEP) [13]. This defines a framework which encompasses many benchmark methods such as LBP. They make the distinction that all methods which are instances of HEP partition the feature space. This is based on image patches by applying a pre-defined function on the intensities of that patch. The texture contributions of this thesis all fit within the HEP framework.

1.1.2 Colour analysis

Image colour is a significantly better understood problem than texture. From a physics point of view a colour is formed from the surface reflection of singular or multiple wavelengths of light from the visible spectrum. From a computational standpoint given a colour space (e.g. HSV, L*a*b or RGB) and an illuminant (e.g. D50, D65) a colour can be defined precisely using a set of 3 values. Pertinent to this thesis is image classification using colour and as a method this has been widely applied.

1.3 Constraints

In writing this thesis we do not attempt either to survey the whole area - it is too vast - or even to claim that we are developing the worlds best classification algorithm, or the most comprehensive and robust image feature or description.

[redacted]

1.4 Contributions

The contributions of this work arise from the industrial focus and from academic insight. We show how feature descriptions and classical machine learning techniques can be used to provide robust systems for product inspection. We then

extend the basis of the techniques in description and in reconstruction. The main contributions are then:

[redacted]

3 We develop a **novel texture histogram** for texture analysis within the Local Binary Pattern framework. This new method increases the textural information encoded while improving classification capability with higher invariant attribute analysis.

4 We propose a **new reconstruction approach** to show the completeness of the image descriptors. We show that by using quadratic programming it is possible to reconstruct an image using its local texture codes.

The minor contributions are:

[redacted]

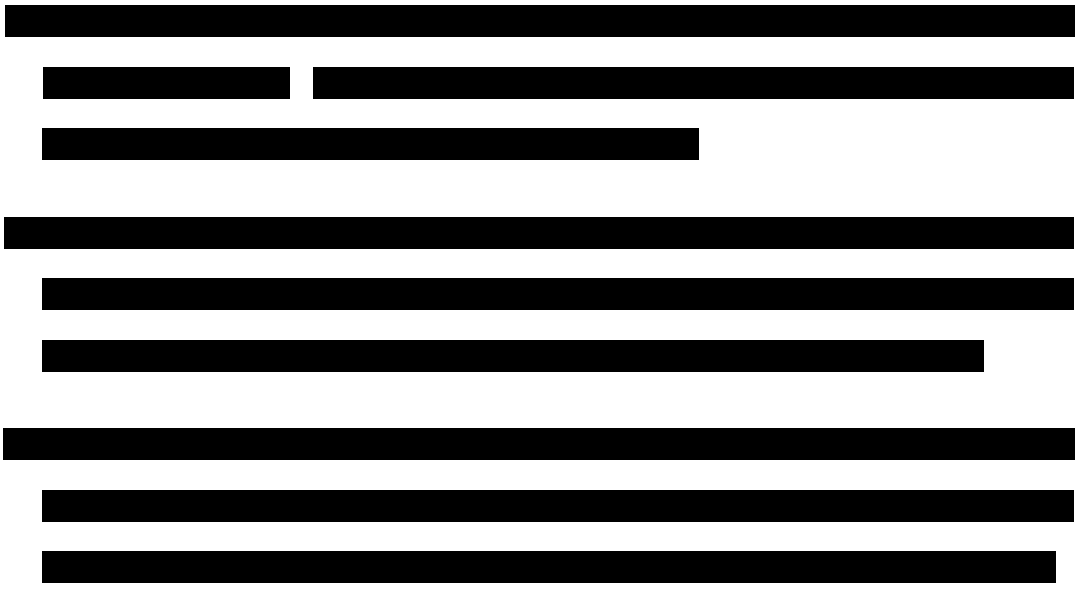
1.5 Thesis structure

We describe our contributions first, concentrating on the industrial material before progressing to the new work on texture and image reconstruction. We survey relevant background literature in texture and colour analysis in Chapter 2. We then move in Chapter 3 where we detail our industrial partner in full, the data provided and our remit from them. Next in Chapter 4 [redacted] This is followed in Chapter 5 [redacted] Chapter 6 describes our new work in texture analysis via a novel Local Binary Pattern framework which we term "Sudoku". We then show in Chapter 7 how is possible to reconstruct images from their texture features using quadratic programming. This shows that the descriptions are unique and reversible, supporting the performance capabilities in texture image analysis.

1.6 Publications

1.6.1 Internal Reports

[redacted]



1.6.2 Conference papers

E Finlayson, Graham, and Seth Nixon. "Sudoku texture classification." *Electronic Imaging*, 2016: This forms the basis of Chapter 5 and is associated with Contribution 3.

F Finlayson, Graham, and Seth Nixon "Reconstructing a colour image from its texture" *Color and Imaging Conference*, Accepted for oral presentation in November 2018: The paper shows how images can be reconstructed from their texture descriptions, as described in Chapter 6, using the basis of Chapter 5.

1.6.3 Journal papers

G Nixon, Seth, and Graham Finlayson "Understanding image reconstruction using Sudoku and LBP", In preparation, *Image and Vision Computing*

Do to the commercial sensitivity of the industrial areas of this thesis some of the publications resulting from this thesis remain only as internal reports.

Chapter 2

Background

This Thesis focuses primarily on digital image classification: determining the contents of an image using computational techniques. We focus on a two stage process, firstly processing images into some distinct description and secondly using classifiers to distinguish between our descriptions.

Specifically we are interested in image texture classification using histograms. To begin this Chapter we will discuss a standard method for image classification in Sections 2.1 and 2.2. Secondly in Section 2.3 we will focus in on how texture can be used to discriminate between different types of images. Thirdly in Section 2.4 we will overview the spectrum of histogram based texture classification methods. Next in Sections 2.5 to 2.7 we delve into various methods for feature extraction based on and around the Local Binary Pattern (LBP), a focus of our work. Finally we will overview a method for visualising our results in Section 2.8. Sections 2.9 and 2.10 review some remaining literature relevant to our work in Chapters 6 and 7.

2.1 Image classification with feature vectors

A question often asked in computer vision is: "How similar are two images?". The simplest method would be to simply take the difference between the two images, if the two images are identical the difference will be 0. This approach is rarely used however as it has many problems. For example, an image compared with a copy of itself which is shifted 1 pixel to the right would have a comparatively large difference despite the images being very nearly identical. Also, the raw pixel values often contain a large amount of redundant data.

In vision the most common method for comparing images is to compute feature vectors from the images and then compare these instead. These feature vectors are a numeric description of the contents of the image, formed based on a mathematical foundation or an observation. Consider the images "kobi.png" and "football.jpg" from the MATLAB default package and a feature vector for each in Figure 2.1.

The feature vectors in Figure 2.1 are the intensity histograms of the images. They are simply counts of the number of times intensity i occurs in each image $0 \leq i \leq 255$. We can now measure how similar these two images are by comparing these feature vectors. Feature vector comparison can be performed using a distance metric, a classifier or other methods and some of these are detailed in the next Section.

2.2 Classification

Ultimately in classification we boil everything down to a single number. This can be anything from a 1 or a 0, indicating yes or no, to a percentage, indicating a measure of similarity.

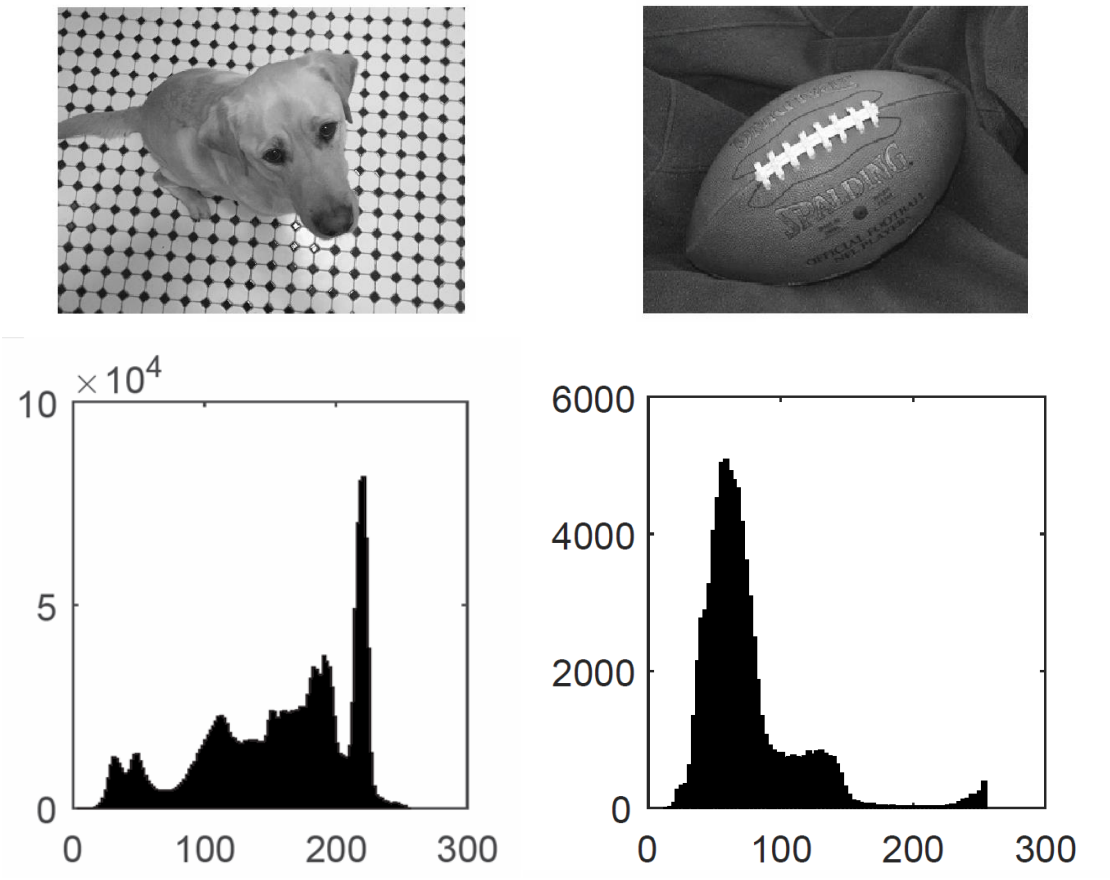


FIGURE 2.1: Two images and their associated intensity histograms.

2.2.1 Distance metrics

One of the most common ways for comparing two feature vectors is to use a distance metric. These take the two vectors and perform some calculations between the two to form a single number indicating difference or similarity. Many have been proposed in the literature and usually fall into two groups: the first group are the general mathematical constructs, the second is specifically designed calculations to target an application or hand crafted feature vector.

Let us consider two vectors $\vec{x}$ and $\vec{y}$ of equal length with values $x_1...x_n$, $y_1...y_n$ where x_i denotes the ith value in the vector $\vec{x}$ $(1 \leq i \leq n)$. One of the most commonly used distance metrics is the Euclidean distance, also known as the L2 distance (the L2 norm of the difference between two vectors). The Euclidean distance considers $\vec{x}$ and $\vec{y}$ as vectors in Cartesian space and states that the difference between $\vec{x}$ and

$\vec{y}$ is the distance between them. Euclidean distance is equal to the square root of the sum of the squared differences between the two vectors, a lower value indicates a more similar pair of vectors:

$$ED = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{2.1}$$

Histogram intersection is another distance metric specific to histogram comparison. It defines similarity between $\vec{x}$ and $\vec{y}$ as the amount each bin overlaps. This is expressed as:

$$HI = \sum_{i=1}^{n} min(x_i, y_i) \tag{2.2}$$

Unlike Euclidean distance Histogram Intersection is a measure of similarity, with a higher value indicating a better match. A pre-requisite of this method is that both histograms are normalised. It can also be shown that Histogram Intersection is equivalent to the L1 distance (the L1 norm of the difference vector), otherwise known as the Manhattan distance [17] when the histograms sum to 1 (the histograms are normalised).

$$L1 = \sum_{i=1}^{n} |(x_i - y_i)| \tag{2.3}$$

There are many other distance metrics used in a variety of applications including Kullback-Liebler Divergence [5], Chi-squared distance [18] or the Hamming distance [19].

2.2.2 k-Nearest Neighbour classification

One of the most commonly used classifiers for a set of feature vectors is the K Nearest Neighbor (kNN) classifier. If we consider a simple problem with n classes of samples: the premise is that a sample from class 1 will be most similar to other samples from class 1, class 2 will match to class 2 and so on up to class n. Choice of distance metric is paramount as the difference measured between vectors drives classification [20].

There are many different schemes for implementing a kNN classifier. One example would be to consider a database as a whole. If we have a set of classes of samples. To classify a single sample we calculate the distance between it and every other sample in the dataset. We take the minimum k distances and assign the sample the modal class within those k results. Once repeated for every sample in the dataset the percentage accuracy of the pass is the number of times the solution is correct divided by the total number of samples. See Figure 2.2 for an example.

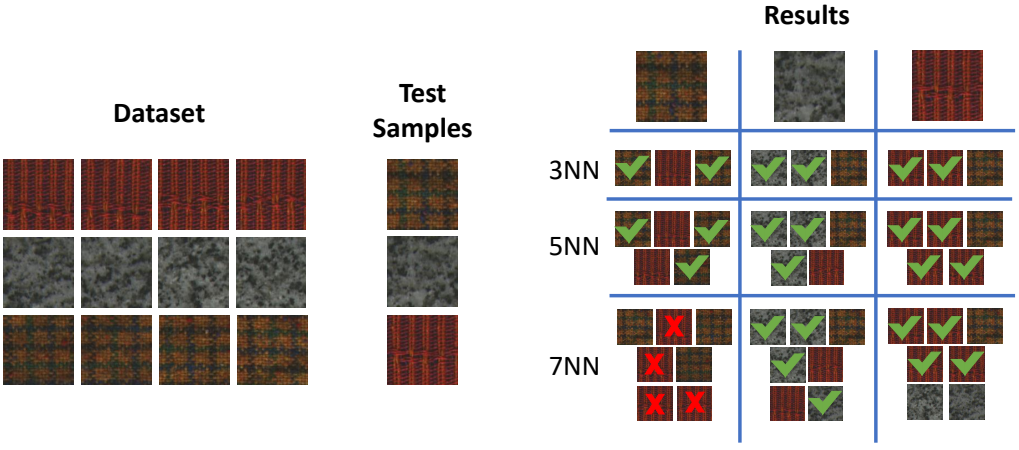


FIGURE 2.2: Example of KNN classification, Left we have a 3 class training dataset, centrally we have 3 query images and right we have KNN classification results with 3, 5 and 7 nearest neighbors.

KNN classifiers do have drawbacks the primary issue being that of complexity. With a standard KNN classifier computational time scales in $\mathcal{O}(n)$ for a single

sample where n is the total number of samples in the dataset. This means that for a full experiment (classifying every image in the dataset) computational time scales in $\mathcal{O}(n^2)$. The number of comparisons required to totally evaluate a dataset is expressed as:

$$n * (n - 1) = n^2 - n$$

as each sample must be compared with every sample except itself.

KNN classifiers have been used in a wide range of applications. Examples include the work of [21] where they detect land mines from radar. In [22] they use a nearest neighbor classifier as part of their process for handwritten digit recognition. Finally we use a KNN classifier to evaluate our new texture representation in Chapter 6.

2.2.3 The Naive Bayes classifier

Supervised machine learning algorithms form decisions based on a set of predefined knowledge. The Naive Bayes classifier is a classic example based on Bayes theorem [23]. This is defined as:

$$P(A|B) = \frac{P(A)P(B|A)}{P(B)} \tag{2.4}$$

Where $P(A|B)$ is the probability of event A given that event B has occurred, $P(B|A)$ is the Probability of B given event A has occurred, $P(A)$ is the probability of A occuring in isolation and $P(B)$ is the probability of event B occuring in isolation. This can be rephrased into a classification problem if A is replaced with Class C and B with feature vector x.

$$P(C|x) = \frac{P(C)P(x|C)}{P(x)} \tag{2.5}$$

Remembering x is a feature vector and C is a class label, equation 2.5 gives the probability of a sample x having class label C. $P(x|C)$ is known as the *prior* probability, $P(C)$ is known as *class* probability and $P(C|x)$ is known as *posterior* probability.

Let us set up a demonstrative example. We have 6 books from 2 authors, 4 of which are written by David and 2 are written by Sophie. Both authors use the words *"Adventure, Explored, Friend"* in their books, however they use them varying amounts. David uses them with probability [0.1,0.6,0.3] respectively and Sophie with probability [0.4,0.2.0.4]. We have two class labels: C_D for Davids books, and C_S for Sophie's. We have $P(C_D)$ which is $\frac{4}{6}$ (4 books of the 6 are Davids) which is known as the prior probability for the class C_D. $P(C_S)$ is $\frac{2}{6}$. We also have the probability of the 3 words occurring given they were written by a specific author. $P(\text{“}Adventure''|C_D) = 0.1$, $P(\text{“}Adventure''|C_S) = 0.4$ etc.

Now, a scrap of paper appears with the phrase *"The adventure went well, our hero and his friend explored the forest. Their next adventure would be.."*. A Naive Bayes classifier can be employed to determine the most likely author of this phrase. In this case, the words in the scrap are the unknown feature vector x and we are trying to determine which class label C_D or C_S is most likely. Firstly, let us calculate for David.

$$P(C_D|x) = \frac{P(C_D)(P(x|C_D)}{(P_x)} \tag{2.6}$$

We have two unknowns: the posterior probability we are trying to obtain ($P(C_D|x)$ and the prior probability of the feature vector given that David is presumed the author ($P(x|C_D)$). $P(x|C_D)$ is a vector of probabilities, each referring to a word

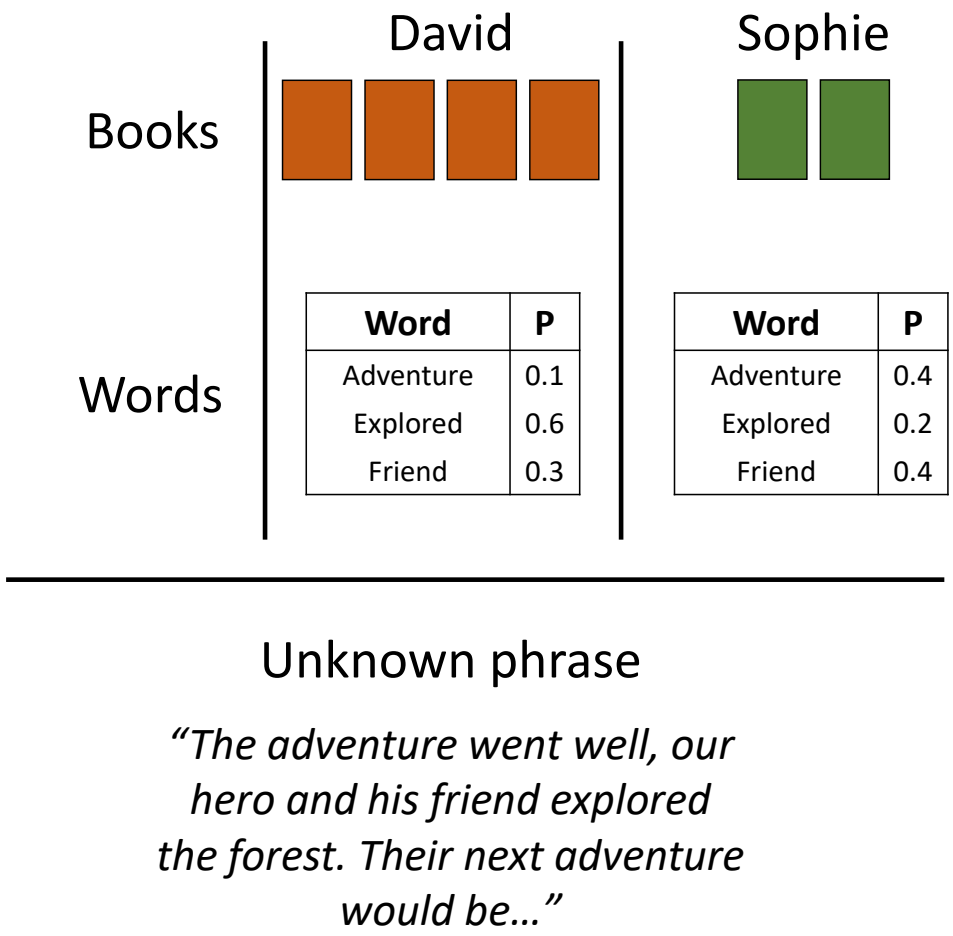


FIGURE 2.3: The book problem.

in the phrase and the likelihood that David wrote it. To compute a vector of prior probabilities into a scalar we exploit the identity:

$$P(A, B) = P(A)P(B) \tag{2.7}$$

Where $P(A, B)$ means the probability of A and B. In this case we can write $P(x|C_D)$ as the product of each element of the feature vector:

$$P(x|C_D) = P(x_1|C_D)P(x_2|C_D)...P(x_n|C_D) = \prod_{i=1}^{n} P(x_i|C_D) \tag{2.8}$$

Also, $P(x)$ (the denominator in the Bayes equation, also known as the *evidence*) is constant independent of class so it can be removed: leading to the final equation for determining class probability:

$$P(C_D|x) = (P(C_D) \prod_{i=1}^{n} P(x_i|C_D) \tag{2.9}$$

Our unknown x has 15 variables (words in the phrase) and we have Davids 3 words and their prior probabilities: $P(\text{“}Adventure''|C_D) = 0.1$, $P(\text{“}Explored''|C_D) = 0.6$ and $P(\text{“}Friend''|C_D) = 0.3$. There are words in the phrase which do not appear in Davids vocabulary, in this case we will ignore them (In practice the zero probabilities must be accounted for. Implementations of a Naive Bayes classifier will usually implement a mechanism to account for zero probabilities. One example is Laplace Smoothing [24]). Our final computation for David is then:

$$P(C_D|x) = \frac{4}{6} * 0.1 * 0.6 * 0.3 * 0.1 = 0.0012 \tag{2.10}$$

And for Sophie is:

$$P(C_S) = \frac{2}{6} * 0.4 * 0.2 * 0.4 * 0.4 = 0.0043 \tag{2.11}$$

The resulting values can then be normalized to form probabilities:

$$P(C_D) = \frac{0.0012}{0.0043 + 0.0012} = 0.218 \tag{2.12}$$

$$P(C_S) = \frac{0.0043}{0.0043 + 0.0012} = 0.782 \tag{2.13}$$

Despite Sophie only having two books she is by far the most likely author of the phrase with a posterior probability of 0.782 compared to Davids 0.218.

The primary assumption of the Naive Bayes classifier is that features are independent: each feature has no bearing on any of the other features (hence "Naive"). In the majority of cases this assumption is untrue, for example in our problem it is plausible that if you write about "Adventure" you are also likely to use the word "Explore". This would suggest that "Adventure" and "Explore" as features are dependent on each other to some degree. However in practice the classifier still performs very well.

Naive Bayes has also been extended: in our above example we are using discrete data, the *Gaussian* Naive Bayes offers a solution for continuous distributions [25]. *Multinomial* Naive Bayes offers multiclass support [26]. Others include the *Normal* and *Lognormal* [25]. In application Naive Bayes classifiers has shown good performance in a wide range of areas, examples include image classification [27] and text classification [28]. [redacted]

In this Section we have discussed how an image may be classified using an arbitrary feature vector. In the next Section we present an overview of the methods for forming feature vectors based on textural information.

2.3 Texture classification - an overview

Image classification using texture is a problem of representational efficiency: how we can capture the maximal amount of discriminative texture information from an image. And then how we use this information to form a description of the images texture. The question then becomes: what is image texture? In reality this question is answered by how capture and description of image texture is approached, the validity of the answer is then the effectiveness of said approach. This Section is ordered as follows. Firstly we shall give a broad overview of the texture field, some methods fall under the category of *Histogram based texture classification* and will instead be discussed in Section 2.4. Secondly we shall give

a more comprehensive view of two key methods, the Gray Level Co-occurrence matrix and the Gabor Filter. Feature vector classification is a two part process - first forming the feature vector and secondly classifying it. Classifier choice plays a large part in the performance of a method, however here we are interested in just the feature vectors. Thus we shall avoid discussing how these methods perform where possible. Texture is a very large field, many surveys have been undertaken to capture the field [29] [30] [31] [13]. Here we simply aim to discuss the general considerations and then the more salient approaches proposed in the literature.

2.3.1 Invariance

The process of developing a texture feature has two primary aspects. Firstly, how textural information is captured. Secondly, the invariance of the method to changes in imaging setup. Invariance is desirable as while a texture may remain the same, the image may change and this may affect the textural description a method produces. The transforms that a method may wish to be invariant to begin with general imaging artifacts such as noise or blur. More specific attributes such as rotation, scaling, shifting and other affine transformations or illumination changes can then be considered. For example if we consider the images in Figure 2.4.

As human beings we can intuitively tell that all four pictures have a *wood* texture despite each of the transformations. Ideally, computational texture algorithms should be able to be able to do the same. Depending on the application of a texture method invariance to one or all possible transforms is desirable, however some invariances are more complex than others. Rotation is simple in consideration as a textured image and its rotated version have the same texture. Invariance to this is in some cases a property of the method, e.g. global statistics from images are naturally rotationally invariant. Otherwise it is usually obtained by some rotational transform of the features themselves "aligning" them in some way [5][1].

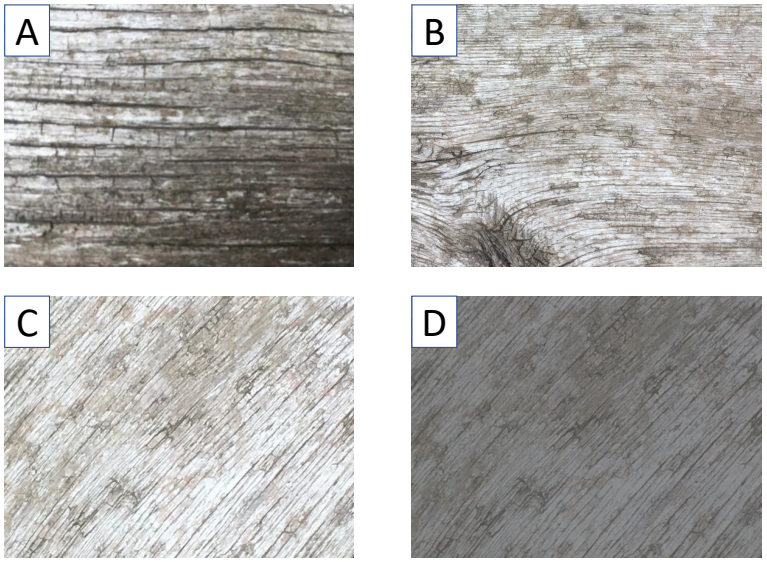


FIGURE 2.4: Examples of transforms in images. A is the original, B is a scaled version of A, C is scaled and rotated and D is a contrast and brightness reduced version of C.

Scale is a more complex issue as different texture elements can occur at different scales. For instance a Brick wall may be classified using the layered brick pattern or the micro-structure of a brick. Image scale is also an arbitrary continuous space, unlike rotation which is a *bounded* continuous space: there are only 360 degrees of rotation whereas there are no well defined limits to scale. Also, image resolution becomes a factor as fine detail textural structure may not be present in a coarsely scaled image. Scale invariance is a challenge to achieve. Most methods utilise a multi-scale approach where features captured at multiple scales form the description [6]. Other methods are defined in a scale-selective manner where the scale is a parameter [32][5]. Other methods attempt to achieve true scale invariance through use of adaptive scale selection where the scale(s) of the texture is approximated from an image [33]. In the next Section we shall give an overview of the general field of texture based image analysis. The methods detailed attempt to address one or more of the described issues.

2.3.2 Approaches to texture classification

Some authors consider texture as a statistical property of an image. First order statistics such as the mean or the variance of an image do have the benefit of rotational and translational invariance, however they lack discriminative power. Thus, second order statistics of images have been a focus of research. One of the earliest methods is that of Haralick and Shanmugan [32] where they present texture as the co-occurrence of gray-levels in an image at some distance and angle. Matrices calculated at varying distances and angles are computed and statistics of these matrices provide an index for recognition. A later extension was the Gray Level Difference Matrix, where co-occurrence of differing pixel values was measured instead [34]. Another statistical approach is that of [35] where they use a range of multivariate analysis techniques such as the Single Valued Decomposition (SVD), autocorrelation and autocovariance functions on raw images to extract texture features.

The Fourier transform is an immensely well regarded technique in both mathematics and computer science. Its transformation of information from the spatial domain to the frequency domain has heralded numerous advances across many disciplines. The primary products of the 2D Fourier Transform are the magnitude and phase spectra of an image. When computed over for a texture image, usually using a Discrete Fourier Transform (DFT), the distribution of the magnitude spectrum describes the coarseness and directionality of a texture. Two example textured images with their Fourier power spectra are shown in Figure 2.5, note how the spectrum is more concentrated for the coarse texture.

In [36] they use coefficients computed from the magnitude spectrum as texture features. The Fourier transform is also key in many filtering applications.

Filtering approaches localise the image in scale and orientation to extract features. Laws Operators [37] are widely considered to be the first filtering method in texture. These texture measures are calculated from the responses of a set of

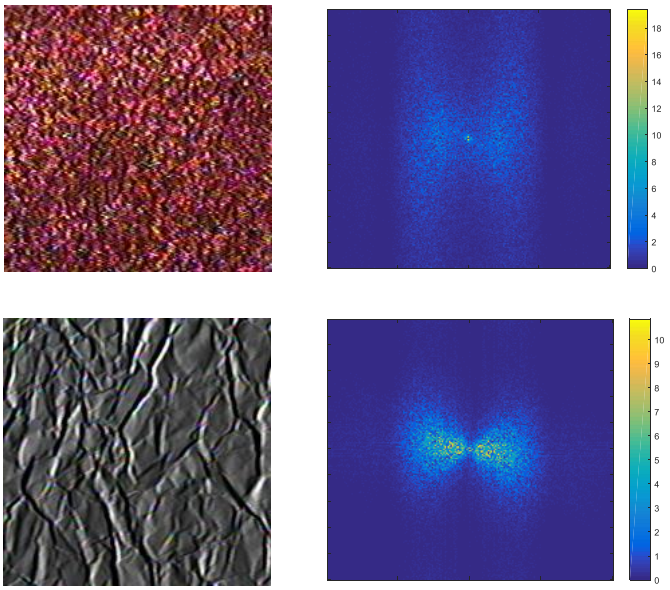


FIGURE 2.5: Two textured images from the CUReT database and heatmaps of their Fourier power spectra with the DC coefficients set to 0. Images were converted to grayscale before having the Fourier transform applied.

separable linear filters. Separable filters are those that exist in 1 dimension in x and y. The product of two separable filters produces a 2D filter. A large selection of 1D filters for extracting level, edge, spot, wave and ripple structures are defined; the multiplication of any two of these producing a distinct 2D filter. Examples are shown in Figure 2.6.

```
L5    = [  1   4   6   4   1 ]
E5    = [ -1  -2   0   2   1 ]
S5    = [ -1   0   2   0  -1 ]
W5    = [ -1   2   0  -2   1 ]
R5    = [  1  -4   6  -4   1 ]

L5/E5 = [ -1  -2   0   2   1
          -4  -8   0   8   4
          -6 -12   0  12   6
          -4  -8   0   4   8
          -1  -2   0   2   1 ]
```

FIGURE 2.6: The Level 5, Edge 5, Spot 5, Wave 5 and Ripple 5 1D filters. And the 2D Level5/Edge5 filter.

Images are convolved with a set of these filters replacing each pixel with a set of measures describing its textural "energy". A large contingent of modern filtering

methods for texture use Gabor filters [38] [39], a type of filter designed to approximate the visual system in mammals. Statistics taken from a careful selection of these has shown discriminative capability as a feature vector. This approach is reviewed in more detail in Section 2.3.4.

Probabilistic structures are also prevalent in texture classification. Markov Random Fields (MRFs) are one example that model random processes and image textures satisfy the necessary conditions to be calculated by this model. In the work of Chellappa et al. [40] they make the assumption that a texture can be generated by a *Gaussian* MRF. They first show that an textured images pixels satisfy the *Markov Property*. Then, for a given textured image, they calculate a least squares estimate of the parameters of the model that would generate the texture. These estimated parameters form a feature vector. Autoregressive models are an instance of MRFs. The Simultaneous Autoregressive Model (SAR) of texture is a member of a class of models *"specifically designed to model spatially autocorrelated data based on neighborhood relationships"*[41]. In [42] the authors present a multi-resolution and rotational invariant SAR. Again, estimated parameters of the model which would generate a texture provide a feature vector.

Wavelet transforms offer a decomposition of an image into oriented sub-bands of information at a certain scale. In imaging they are similar to the Fourier transform in that they transform an image from the spatial domain to the spatial-frequency domain. However where Fourier models an image as a sum of infinite sinusoids, wavelet transforms model an image as a sum of finite wavelets via scaling, rotation and translation. Usually designed to be over-complete they also offer excellent reconstructive capability. Figure 2.7 shows an example wavelet decomposition.

A huge variety of different wavelets are available and some have seen use in texture. In [43] they use the channels of a Discrete Wavelet Transform to form a feature. In [44] they form feature vectors using the Dual Tree Complex Wavelet Transform (DTCWT) to extract frequency and directional information from an image. [45]

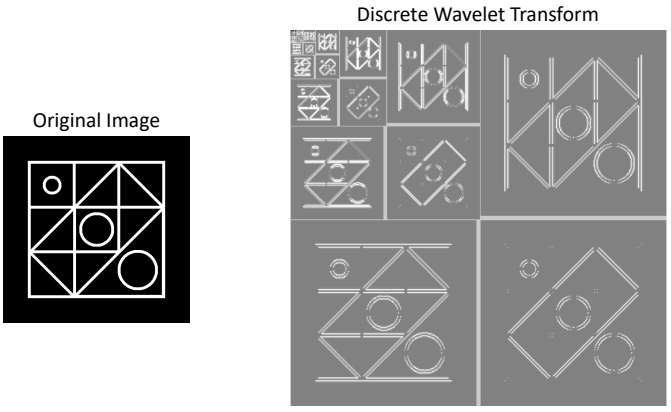


FIGURE 2.7: Example wavelet decomposition using three orientations: 0 deg, 45 deg and 90 deg and six scales.

offers another method based on the DTCWT. They show that the variance and entropy calculated from each of the oriented sub-bands of information at each scale offer feature vectors which are discriminative and robust to noise [45].

A perfect fractal is a shape which, among other things, if equally subdivided and re-scaled produces itself. Shapes which exhibit this property have an integer fractal dimension, shapes which do not have a fractal dimension somewhere been two integers, examples are shown in Figure 2.8. In imaging the fractal dimension is considered a measure of complexity or roughness and can be used as a feature for describing texture.

In [46] they calculate the fractal dimension of textures and complement it with a set of measures known as *lacunarity* to form feature vectors. Other methods exploit the fractal dimension using a fractional Browninan motion (fBm) model. The fBm model is characterised by the single "Hurst" parameter which is a measure of the roughness of an image. In [47] they use a generalised set of Hurst parameters as features.

Sparse sampling techniques presume that only certain areas of an image contain important textural information. In [48] they present a method based on local affine

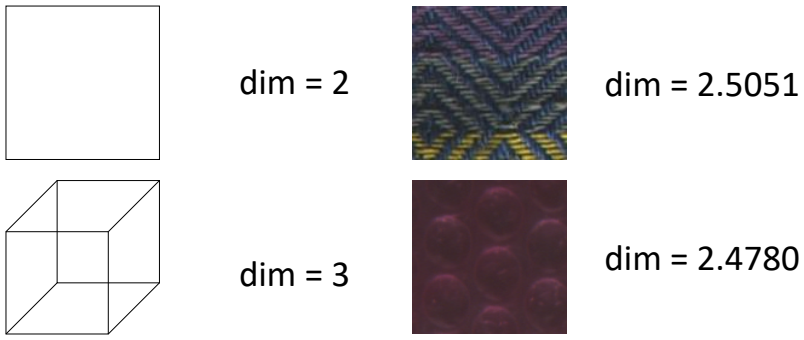


FIGURE 2.8: Example shapes and their fractal dimensions. On the left: a perfect square and a perfect cube. On the right: two examples from the Outex_TC_00013 dataset.

regions which are invariant to local transforms. They use Harris and Laplacian detectors to extract feature points from an image. Descriptors for these points come in two forms. Firstly the Intensity Normalised Spin Image based on Spin Images [49]. Secondly the RIFT (Rotationally Invariant Feature Transform) descriptor based on SIFT [8]. These descriptors are clustered to form signatures, which can be compared using a metric such as the Earth Movers Distance.

Learning approaches use a data driven approach and machine learning techniques to classify texture images. Deep learning techniques such as that of [50][51] are known for exceptional performance albeit with a rather obfuscated mechanism. In [50] they use a modification of the traditional Convolutional Neural Network to extract highly dimensional feature vectors from images. They show performance equivalent or better to current state of the art deep learning techniques. As part of the contribution of Hayman et al. [52] they offer a learning approach using Support Vector Machines evidencing the importance of scale in texture classification algorithms.

As evidenced above, texture is an immense field of research. There is a huge literature available with a near-infinity of methods and variations. This is in part due to its nebulous nature. In the remainder of this Section we will cover in more

detail two of the more salient developments over the history of this area. Firstly in Section 2.3.3 we shall discuss Gray Level Co-occurrence matrices. Secondly we cover Gabor filters in Section 2.3.4.

2.3.3 Gray Level Co-occurrence Matrices

The work of Haralick et al. [32] [2] is one of the earliest methods of texture description. Their statistical approach characterises an image from its simultaneous occurrence of intensities in scale and orientation.

Consider a 8-bit grayscale image, there are 256 different intensities so the co-occurrence matrix will have 256 rows (i) and 256 columns (j). Each cell in the 256×256 matrix corresponds to the number of times intensity i and intensity j occur simultaneously at distance d and angle θ. In [32] the authors recommend producing matrices for the angles 0 deg, 45 deg, 90 deg and 135 deg with distance at 1 or 2. See Figure 2.9.

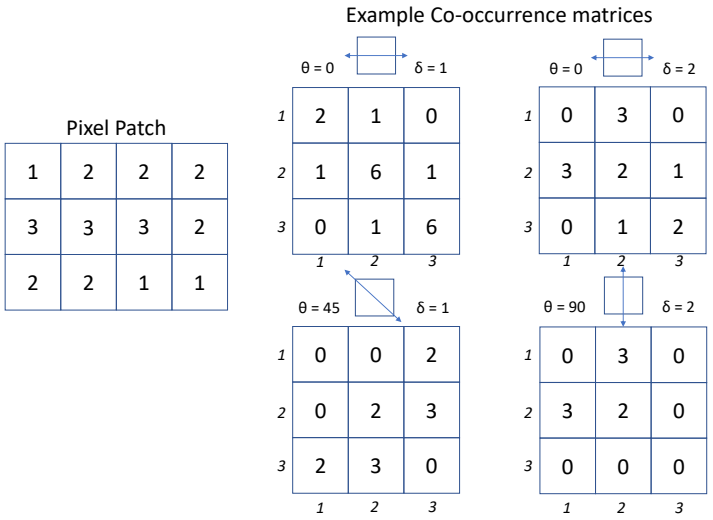


FIGURE 2.9: A pixel window with three gray-levels and corresponding example co-occurrence matrices with varying θ and d

The angular variation is in place to ensure that all orientations are captured by the matrix. It is worth noting that this only achieves orientation-selectivity, not rotational invariance. An image and its counterpart rotated clockwise 45 deg will have

corresponding GLCMs at a 45 deg offset. However this method does correspond effectively for unrotated textures. Also, later we shall see that this principle of orientation-selectivity remains employed in more recent methods using filters. The distance parameter δ is analogous to modern methods for capturing scale. By increasing the value of δ we will capture texture information at coarser scales. With appropriately selected values for θ and δ the assumption is that the co-occurrence matrices contain all of the textural information of the image. The texture features for an image are then a set of statistics calculated over the matrices. This method could equally be considered as a histogram as the GLCM's are a count of a specific set of features, however in practice they are rarely used in their raw form.

2.3.4 Gabor filters

Filtering approaches are common in all areas of image processing. Examples include Gaussian and mean filters which, among other things, are used for image smoothing, differential filters are used for edge detection and band pass filters remove certain frequency bands from images. In texture analysis Gabor filters are the most common technique. Gabor filters are linear filters which extract oriented local frequency information.

A 2-D Gabor filter is defined as a sinusoidal wave modulated by a Gaussian envelope [39]. The *response* of a Gabor filter at a point *(x,y)* is given by.

$$\psi(x,y) = \frac{F^2}{\pi\gamma\eta} e^{-F^2[(\frac{x'}{\gamma})^2+(\frac{y'}{\eta})^2]} e^{i2\pi F x'} \tag{2.14}$$

where

$$x' = xcos\theta + ysin\theta \tag{2.15}$$

$$y' = -xcos\theta + ysin\theta \tag{2.16}$$

Within this formulation there are four parameters. F is the central frequency of the filter, this can be considered analogous to scale, and θ is the orientation. γ and η are smoothing parameters in the x and y directions in an image. An evaluation of these parameters can be found in [53].

In a texture classification task, to obtain the maximum amount of information, a set of filters is defined at various scales and orientations. Common practice is to choose a set of scales which reflect the texture components of the image in question, and a set of angles which ensure you have all 360 deg of rotation covered. See Figure 2.10 for an example filter-bank.

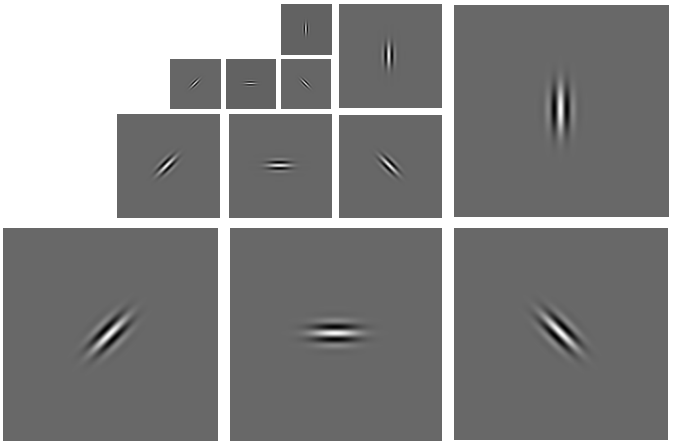


FIGURE 2.10: Visualisation of an example Gabor filterbank with 3 scales and 4 orientations (0 deg, 45 deg, 90 deg and 135 deg).

Linear convolution is the process where a linear filter is applied to an image. Every pixel in the image is replaced with a linear sum, the weightings defined by the filter, of its neighbors. What results from convolution of an image with Gabor filters is a set of Gabor filter responses. An example is shown in Figure 2.11.

Note how the filters extract different textural information at different scales. In the fine scale responses the dogs fur is highlighted, whereas in the coarser scales the spotted pattern of the dogs coat is brought to the fore.

The Fourier transform offers key insight into filtering. The convolution of a filter and an image in the spatial domain is equivalent to the multiplication of the

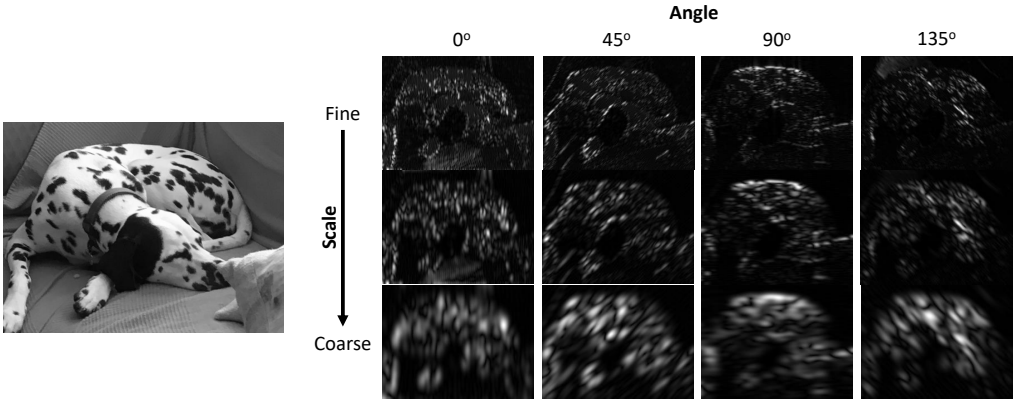


FIGURE 2.11: Image filtered by the Gabor filter bank in Figure 2.10.

frequency domain of the image and the frequency domain of the filter. The filtering method described in this Section also has a frequency domain implementation, intuitively this makes sense as the filter itself extracts the frequency information from the image. Thus Gabor filters can be seen as a Fourier description with localized capability. Also, the Fourier transform has real and imaginary coefficients and consequently the magnitude and phase spectra. These components have used individually in a Gabor texture description [39] .

The most common approach for define a feature vector from a set of Gabor Filter responses is to calculate a set of statistics. In [54] and [55] their feature vector is the set of means μ and standard deviations σ of the magnitude of the filter responses for an image. Other metrics computed from Gabor filters include the Gabor Energy Feature, the Complex Moment and the Grating Cell Operator Feature [39].

2.3.5 Conclusions

In this Section we have discussed a selection of the approaches for capturing an images texture. Each of the methods above has shown discriminative ability. We have also discussed invariance: its importance and also its complexity. All texture classification work undertaken in this thesis begins with the Local Binary Pattern, a histogram based texture feature. In the next Section we shall overview the

universe of histogram based texture classification. Then in Section 2.5 we shall comprehensively review the Local Binary Pattern.

2.4 Histogram based texture classification

Histograms are the count of a discrete space of variables. If we have a bag of fruit with 3 apples, 4 oranges and 1 banana, the histogram of the contents of the bag looks as in Figure 2.12.

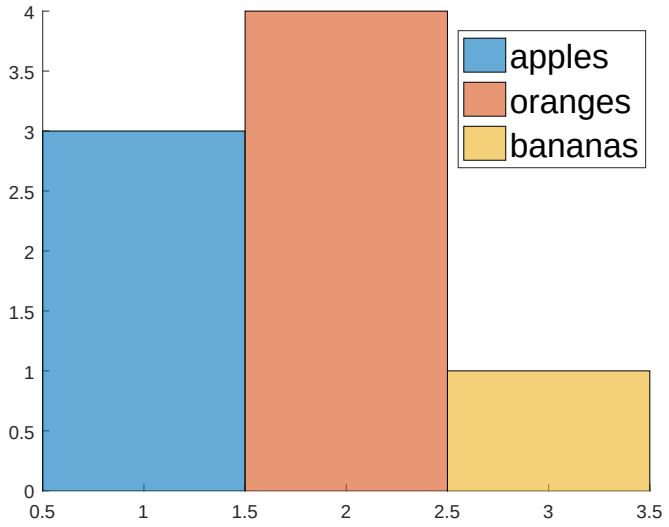


FIGURE 2.12: Histogram of 3 apples, 4 oranges and 1 banana.

In imaging histograms are a sub-set of feature vectors. Where feature vectors are any set of numbers describing an image, histograms are the distribution of a discrete vocabulary of features within an image. It should be noted that signature based methods also use discrete vocabularies of features, however those methods tend to perform clustering or some other processing to form a final description and use specialized measures for comparison. In this Section we focus only on histograms and first discuss the primary challenges in developing a texture histogram. Secondly we offer an overview of methods presented in the literature.

The primary goal of histogram based texture methods is to define a feature space, then partition the space to form a set of discrete "words". There are two main ways develop this vocabulary. One is to develop a set of features relative to the data you are trying to classify. This informs a *data-driven* partition of the space. Examples include [56] [57] [1]. The second method is to define a *general* set of features. This general set exists separately to any data and is usually defined mathematically, referring to a set of structures or some other observation on the composition of texture. Examples of these include Local Binary Patterns [5] and Basic Image Features [6].

The difference between these two methods is that of *generality*. The methods which generate a general set of features tend to perform well across a large swathe of textured images. They also benefit from computational efficiency increases as they require no pre-training step. The methods which inform a data-driven partition will often offer the specific textural information which their dataset requires. However the vocabulary they generate, if taken out of context, may offer little to no descriptive ability on other data. Also, depending on how the data is used to form the vocabulary, bias may be introduced into the representation [58]. All methods covered in Section 2.4.1 will fall into one of the two brackets.

2.4.1 Approaches to histogram based texture classification

Julesz [59] presented the concept of *Textons* as a set of fundamental image structures. In the work of Leung and Malik [56] the Texton was formalized as a cluster centre in filter response space. They show that a dictionary of these Textons calculated over an image or set of images offers discriminative capability. In [57] they extend this to a 3D Texton. They filter 20 different views of the same textured material to capture surface variations. They then perform K-means clustering over the set of responses to generate a dictionary of Textons. An unknown stack of 20 images has its pixels labelled with the closest Texton representation to the local

jet of filter responses. These labels are then histogrammed to form the feature vector which can then be compared with a set of model histograms. An issue with this method is that of high dimensionality, the clustering has to be performed over 960 dimensions. The Textons are also not rotationally invariant, and in practice operating on multiple views of the same texture is cumbersome. Cula and Dana [60] offered a solution with significantly reduced dimensionality. Schmid [61] offers a rotationally invariant Texton algorithm with some scale selectivity.

The work of Varma and Zisserman [1] is perhaps the best known Texton method applied to a single view of a texture (as opposed to the multiple view definition of [57] [60]). They offer a rotationally invariant Texton computed using their MR4 and MR8 filterbanks. These filterbanks achieve rotational invariance by combining the maximum responses of sets of oriented filters. The MR4 filterbank is comprised of six edge, six bar, one Gaussian and one Laplacian of Gaussian resulting in 14 filters and 4 responses. This filterbank is shown in Figure 2.13. The MR8 filterbank is an augmented MR4, where the edge and bar filters also have three scales, resulting in 38 filters with 8 responses.

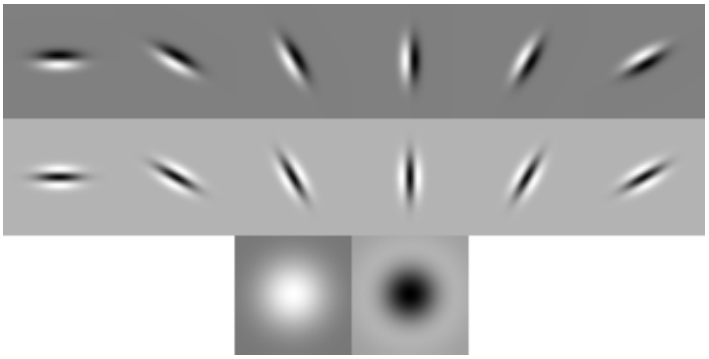


FIGURE 2.13: MR4 filterbank [1]. Code courtesy of *https://www.robots.ox.ac.uk/ vgg/research/texclass/filters.html*

Their maximum response methodology significantly reduces the dimensionality of the clustering to 4 dimensions with VZMR4 and 8 with VZMR8. The VZMR8 method is often used as a benchmark technique for comparison of newer methods.

Later in [62] they offered a further method known as VZ-Joint. Instead of using local jets of filter responses as the basis, they instead cluster the joint distribution of local intensities to form their Textons. Varma and Garg [63] use this principle further and combine it with fractal geometry. Textons are formed by clustering a set of local fractal dimension and fractal length features computed densely across an image.

One early texture histogram was the Texture Spectrum (TS) [64]. In local 3x3 neighborhoods the central pixel is compared to its neighbours. If a neighbour is less than the central pixel it is assigned a 0, equal to the centre a 1 and greater than the central point a 2. These labels are then vectorised to form the feature for that pixel, the distribution of these features is then the histogram. By using relative comparisons the features are invariant to monotonic illumination functions applied to the image, however there is no consideration of scale or rotation. Local Binary Patterns (LBP) [5] are perhaps the most well known texture descriptors and build on the TS. They offer a compact, multi-scale and rotationally invariant texture description. They remove the equality from TS resulting in a binary string composed of greater than or less than relationships per pixel. Neighborhoods are sampled circularly around the centre, with points not landing on pixel centres approximated, and are defined for an arbitrary radius and number of points. A comparison of the TS and LBP features is shown in Figure 2.14.

After some processing and grouping of the binary strings what results is an effective histogram which has been used in many applications [65][7]. It also has a huge range of extensions to target various applications [66][67][68]. This method is detailed comprehensively in Section 2.5.

Spectral histograms are histograms of the marginal distributions of a filterbank. The marginal distribution of an entity is the set of probabilities of occurrence of a discrete set of properties of the entity. Thus this method can be considered a histogram of histograms. In [69] they use four filters: the intensity filter, gradient

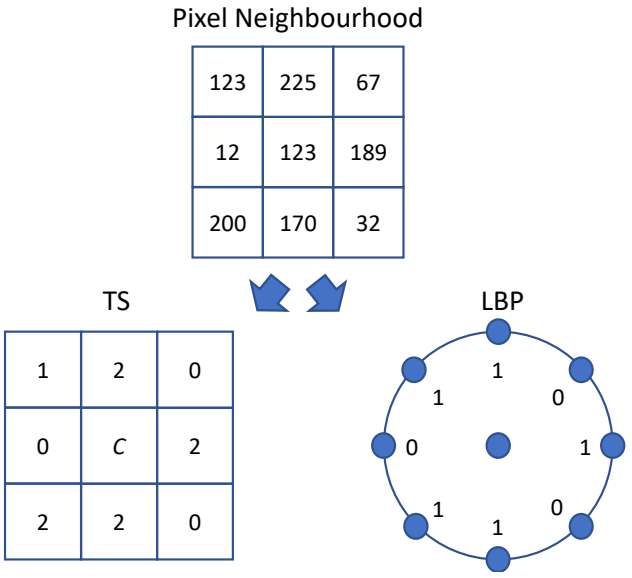


FIGURE 2.14: Comparsion of the TS and LBP features on a 3x3 neighborhood. LBP is parametrised with 8 points and radius 1, corner pixels are interpolated.

filters, Laplacian of Gaussian filters and Gabor filters at, where applicable, varying orientation and scale, the marginal distributions of which are estimated using a windowing function. They select the marginal distributions which comprise the final histogram based on a greedy algorithm which maximises *classification gain*, i.e. they optimise the set of filters to obtain maximum performance.

The Fourier transform has also seen use in forming texture histograms. Local Phase Quantisation (LPQ) was proposed by Ojansivu and Janne [70]. They offer a blur insensitive texture histogram using local phase. They compute DFTs densely across an image and take 4 low-frequency coefficients from the phase component of each, decorellate them via a whitening transform and then quantise them. The histogram of the quantised coefficients shows significant classification accuracy increases on blurred images compared with LBP.

Basic Image Features (BIFS) [6] offer a compact dictionary of fundamental image features. The BIFs themselves are their mathematically defined set of seven qualitatively different image structures based on image symmetries. An image is convolved with six Gaussian derivative filters, each pixel in the original image is

labelled with the BIF most similar to its 6-vector of filter responses. The authors find that, for texture classification, only 5 of the filters and 6 of the BIFs are useful for description. They also find that a multiscale representation is key. Their histogram of co-incidences of BIFs (denoted BIF columns) across four scales shows strong performance. They then extend the multiscale notion to multiple histograms with different base scales. Classifying these with a kNN classifier using the Bhattacharya distance (a weighted average metric for multiple histograms) they observed best in class performance on the UIUCTex and KTH-Tips datasets on publication.

2.4.2 Conclusions

Histogram based texture classification has clear discriminative ability. In this Section we have offered a broad swathe of the methods available. The texture classification work in this thesis is focused primarily on the Local Binary Pattern (LBP). In the next Section we will give a detailed review of LBP and its extensions.

2.5 Local Binary Patterns

Local Binary Patterns (LBP) provide a mechanism for quantifying structure in local neighborhoods and across an entire image. LBP is one of the most, if not *the* most, well known histogram based texture classification methods available. It pervades many applications of computer vision such as texture analysis [71], texture segmentation [65], face recognition [7] and medical imaging [72].

A single LBP is formed from two components: a central pixel C and its neighbourhood of pixels $p_1, p_2 ... p_n$. The neighborhood p is the set of pixels at distance r away from the centre. Each pixel in this neighbourhood is compared to C, if its intensity is greater or equal it is assigned a 1, otherwise a 0, see Figure 2.15.

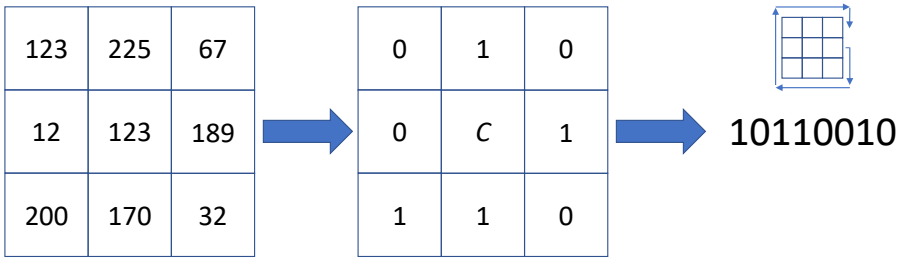


FIGURE 2.15: Transformation of a local neighbourhood into its LBP glyph.

This relational approach to encoding pixels gives the LBP complete invariance to any monotonic changes in the grayscale of an image. Traditionally, these values are then read off on a per-pixel basis to form a set of binary strings for an image. These strings then form a histogram which can be used as an index for classification [5]. This method is denoted $LBP_{r,p}$ where p is the number of pixels in the neighbourhood, and r is the distance from the central pixel at which the members of p are sampled. This method can be expressed with the equation

$$LBP_{r,p} = \sum_{i=0}^{P-1} s(C - p_i)2^i \tag{2.17}$$

Where P is the size of the neighbourhood p and the function $s(x)$ is defined as:

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{2.18}$$

2.5.1 Rotational invariance

A common problem in many areas of computer vision is that an image and its rotational transform usually have differing histograms, feature vectors or descriptions, regardless of having the same content. This differing description can be useful, for example detecting motion across a video sequence, however as we are trying to find corresponding images rotational invariance is desirable. This is achieved for

LBP by rotating each pattern, with circular wraparound, to its maximum value. Consider the neighbourhoods in Figure 2.16.

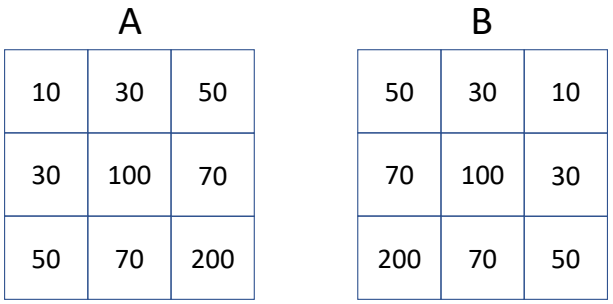


FIGURE 2.16: Neighbourhood A and its 90 deg rotation B.

We are interested in structure. If we consider a triangle and then rotate it an arbitrary number of degrees it still remains a triangle. This holds for local pixel structure as well. Neighbourhood B in Figure 2.16 is clearly the 90 deg clockwise rotation of neighbourhood A. We can consider neighbourhoods A and B to be equal from a structural point of view. If we then move to A and B's LBP counterparts in Figure 2.17.

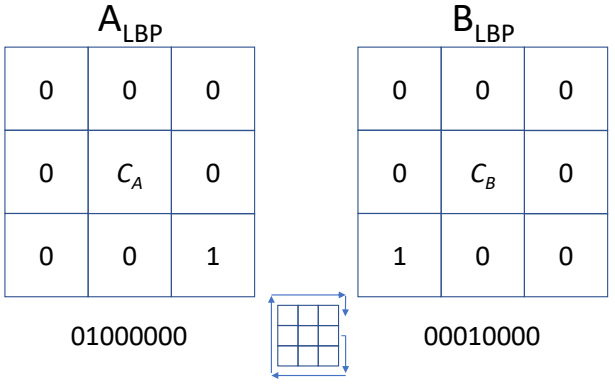


FIGURE 2.17: LBP counterparts of A and B in Figure 2.16. LBP codes beneath.

These are read off from a pixel at a fixed position. In this instance, and all our work with LBP, we will read from the right-central pixel. Neighbourhood A has LBP code 01000000 and B has LBP code 00010000. Structurally these patterns are identical, thus we would like them to match. We can achieve this by rotating,

or bit-shifting, the pattern elements with circular wraparound until we obtain the maximum or minimum value of the pattern (maximum or minimum is an arbitrary choice, as long as it is consistent). In the case of Figure 2.17 the value of both patterns is maximally 10000000 and minimally 00000001. This, in effect, rotates each neighbourhood 45 deg (for an 8-bit pattern) per bit shift such that they all align according to their LBP representation. In all our work with LBP we choose to shift to the maximum. This extension changes the notation and is now expressed as $LBP^{ri}_{r,p}$. r and p are as before (see Section 2.5) and the superscript ri denotes rotationally invariant.

2.5.2 Border handling

An issue in many image processing tasks is how the border (the *edge*) pixels of an image are handled. For LBP it is not possible to code the edge pixels correctly: firstly there are less than 8 pixels in each neighbourhood, and secondly the pixels being coded are not centred in a neighbourhood. There are multiple common solutions for addressing this including:

- Reflection - A reflection of the image is added along the borders.
- Repetition - The images edge pixels are repeated along the borders.
- Removal - Edge pixels are ignored.
- Interpolation - A "best guess" pixel or set of pixels is added at each location along the border based on calculations from pixels near the border of the image.
- Wrap-around - The pixels from the opposite edge of the image are wrapped around. This assumes the image is infinitely tileable.

Each of the above methods has benefits and issues. For instance removal of edge pixels has the benefit that all LBP codes are produced from the image itself with

no synthetic data added. However a proportion of the information is lost from those edge pixels. Wrap-around allows you to use the both the edge pixels and original data from the image, however the assumption of infinite tile-ability is usually incorrect.

2.5.3 Uniformity

Uniformity considers the number of changes from 0 to 1 in a single pattern. For instance the pattern 01010101 has a uniformity value of 8 and 11111111 has uniformity 0. Patterns with uniformity 0 or 2 describe low level structural features such as edges, dark spots or line ends [5], see Figure 2.18.

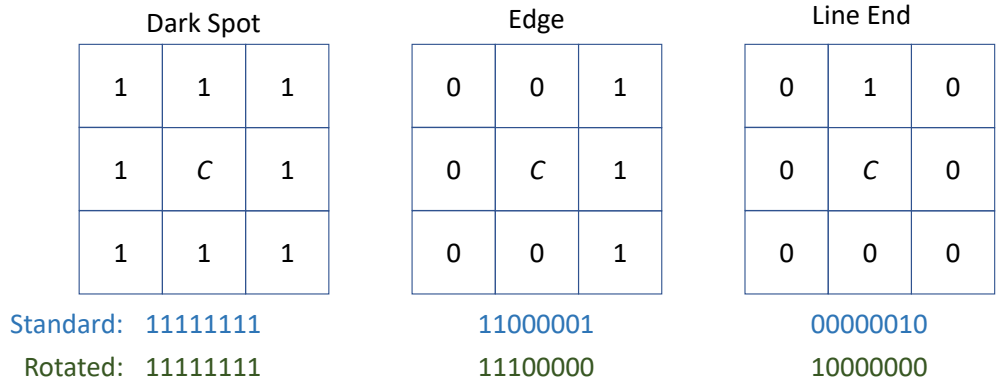


FIGURE 2.18: Examples of uniform patterns.

Through analysis of pattern occurrence it can be shown that up to 95% of patterns in an image have uniformity 2 or less and the remainder can be considered as insignificant or noisy [5]. When forming histograms patterns with uniformity > 2 can be grouped into a single bin without discriminative loss. When combined with rotational invariance this condenses the number of patterns from 2^p to $p + 2$ where p is the number of neighbouring pixels being sampled. Uniformity 2 when combined with rotational invariance, see Section 2.5.1, LBP is denoted $LBP^{riu2}_{r,p}$ where r, p and ri are as before and $u2$ denotes Uniformity 2 [73].

2.5.4 Circular sampling

The basic formulation of LBP is based on a max *distance* from the centre. If we use a distance of 1 then we are talking about the 8 direct pixel neighbours in a 3×3 neighbourhood. In a 5×5 neighbourhood we would use the 16 pixels at distance 2 away from the centre.

A more recent, and more effective, formulation of LBP uses pixels a max *radius* away from the centre forming a circular sampling pattern [5]. See Figure 2.19.

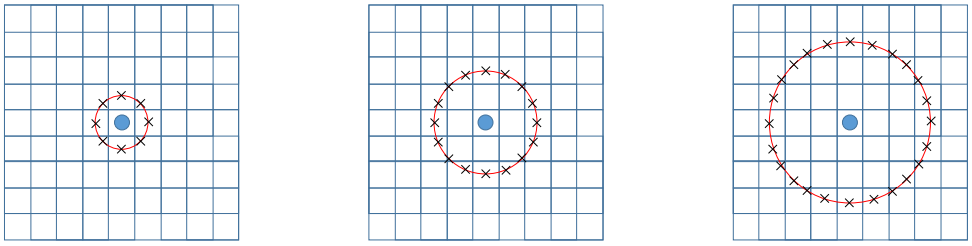


FIGURE 2.19: Circular Sampling neighbourhoods. From left to right: $LBP_{1,8}$, $LBP_{2,16}$ and $LBP_{3,24}$.

A gray-scale images pixel grid is a discrete Cartesian grid with only the integer values. Each pixel is indexed with coordinates centering the pixel on location *(x,y)*. With reference to Figure 2.19 a large proportion ($\geq 50\%$) of sampling points in a circular sampling pattern do not fall on pixel centres. To allow these sampling points to be correctly captured we must adjust our perception of an image. Now we must consider an image to be a continuous Cartesian grid where pixel centres fall on the integer values. This allows us to interpolate a quartet of pixels, based on a location, to sample an estimated pixel exactly centred on a sampling point. Figure 2.20 shows an example of pixel interpolation.

Figure 2.20 shows the process of interpolating a pixel not located on a pixel centre. It is simply a weighted, by the sampling points location, sum of the 4 nearest neighbours located within a local coordinate system. The weightings will also always sum to 1. As x and y are always in the range *0,1* the value of the interpolated pixel must fall between the smallest and largest values in the neighbourhood.

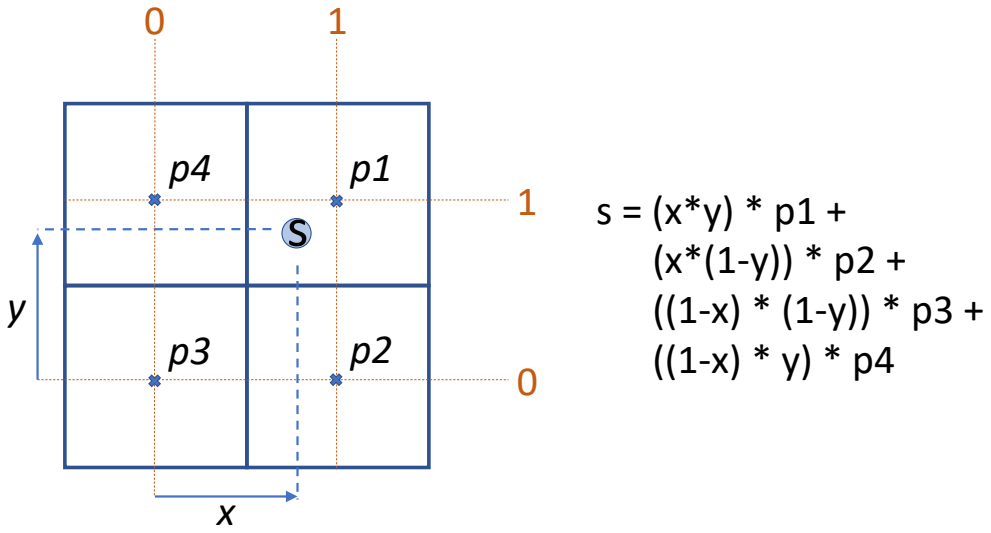


FIGURE 2.20: Example of interpolating a pixel within a quadrant. Coordinates are shown in red, S is the pixel we are sampling and x and y are the coordinates of S within the quadrant of pixels.

Finally the entire process of classifying images using LBP histograms is detailed in Figure 2.21.

Figure 2.21, in summary, begins with a single image, moves on to a single neighborhood within that image and then how this is interpolated to form a circular sampling pattern as in 2.5.4. This is then processed into its LBP code. The whole image is formed into LBP codes which are all then rotated to their maximum value as in Section 2.5.4. The non uniform patterns are then grouped (assigned their own value) as in Section 2.5.3 leaving us with our final representation denoted $LBP^{riu2}_{1,8}$. The values in the image are then counted to form a histogram. Finally we show a simple example of classification where histograms of three different classes of texture image are clustered on an *(x,y)* plane. In the next Section we discuss some of the drawbacks of LBP.

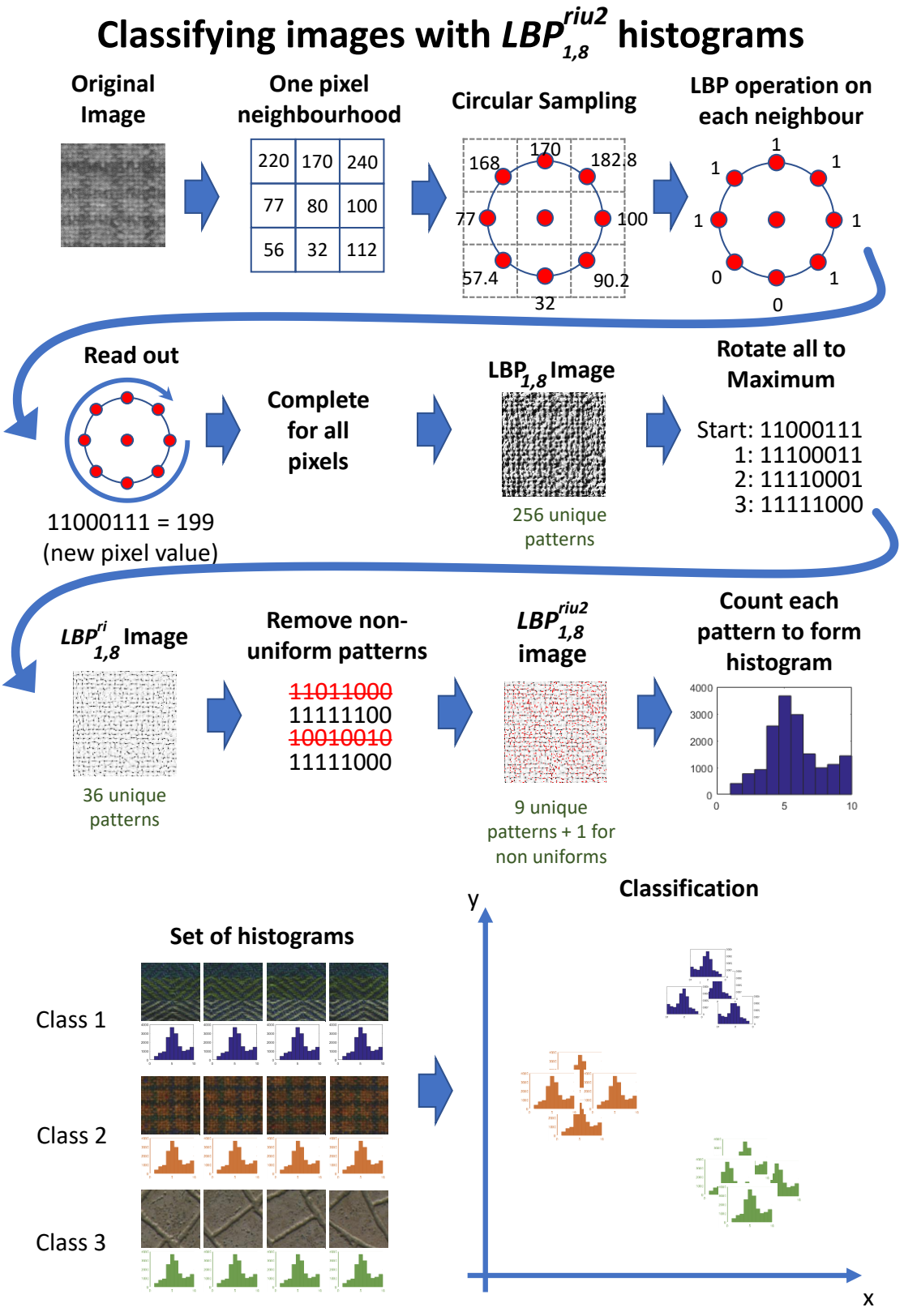


FIGURE 2.21: Classification of images using LBP histograms

2.5.5 Drawbacks

While the LBP method has been shown to have discriminative power when applied as a description it has some faults. Firstly it is very noise sensitive. For instance consider an image with random valued noise (a common problem caused by poor transmission, faulty camera hardware or image coding errors [74]) where certain pixels in an image are assigned random values. For an $LBP^{riu2}_{1,8}$ process each of these incorrect pixels will appear in 9 different LBP codes potentially changing them. As noise density increases (or signal to noise ratio (SNR) decreases) so does the number of corrupted codes, hence the final histogram becomes more and more distorted. Figure 2.22 shows an example of this.

From Figure 2.22 we can see that with a barely visible amount of noise the LBP histogram is slightly affected. Once the noise density increases the histogram quickly becomes unrecognisable.

Another concern is that of magnitude representation. One of the tenets of LBP is that the structure of a neighbourhood, or the sign of a difference, is far more important than the magnitude of the values contained therein. This may be correct however the magnitude information may still be significant. Consider the neighbourhoods in Figure 2.23.

The pertinent feature of Figure 2.23 is that neighbourhood A has near uniform structure whereas B is more random, with a significantly higher variation in intensity. The issue is that, regardless of their vastly different structures, they form identical LBP codes. While the examples in Figure 2.23 are extreme cases the reader can convince themselves that this issue can cause a loss of information and in some cases a very weak description of a pixel neighbourhood. There have been solutions proposed which aim to solve this problem. Examples include Completed LBP detailed in Section 2.6.4 and Improved LBP in Section 2.6.3. In Chapter 6 we propose a method which explicitly encodes relative magnitude.

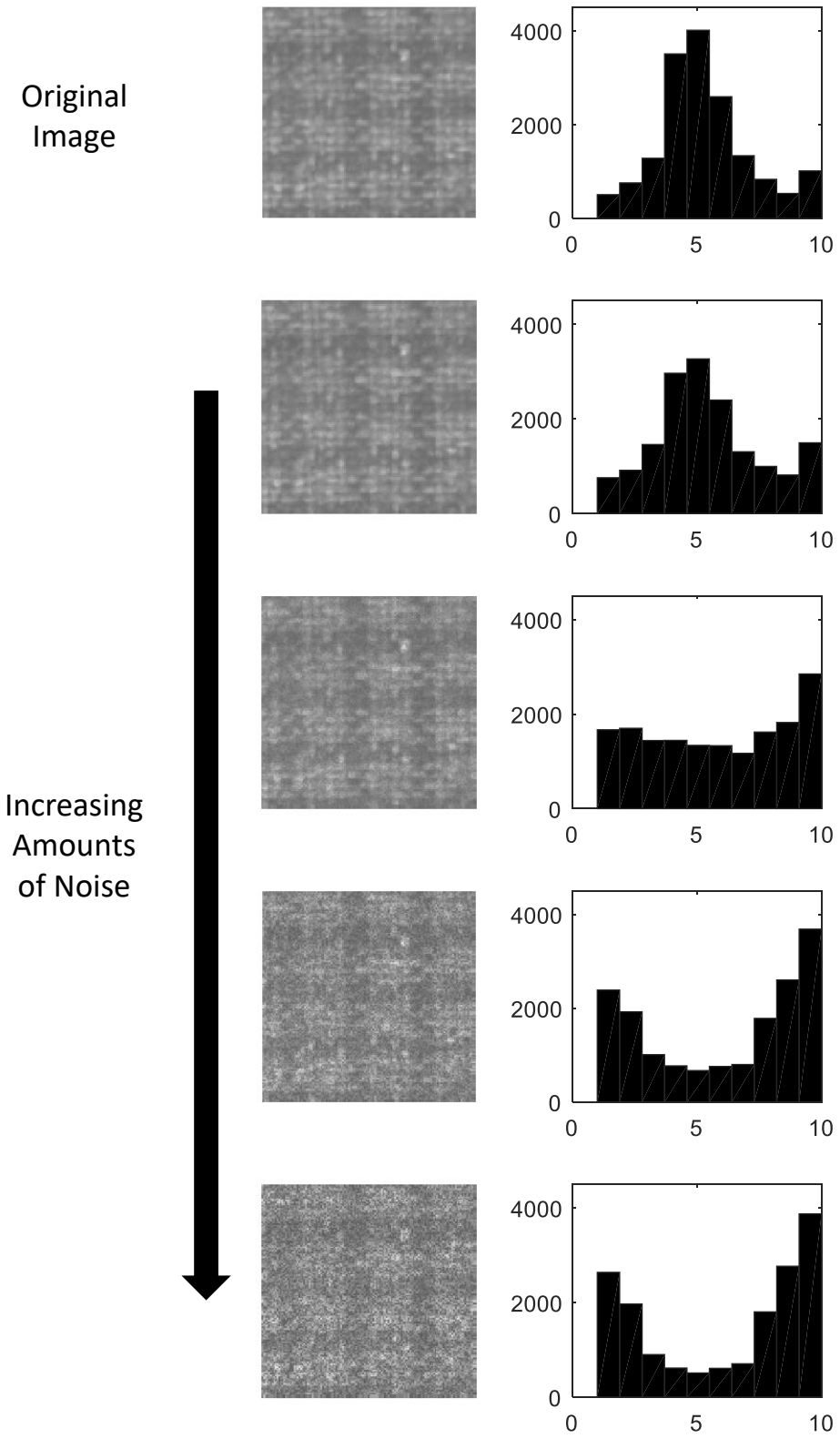


FIGURE 2.22: Examples of an image corrupted by increasing levels of gaussian white noise and the associated LBP histograms.

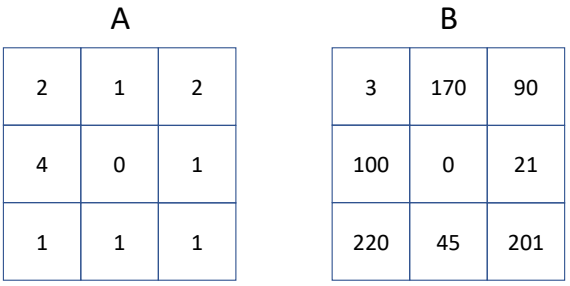


FIGURE 2.23: Examples of pixel neighbourhoods. Left with near uniform structure, right with a more random structure.

2.6 Variations and extensions to LBP

As our contribution, described in Chapter 6 can be considered a variant of LBP we will now briefly review some of the commonly referenced extensions and variations of LBP.

2.6.1 Contrast

The contrast of a pattern $VAR_{r,p}$ is an extension which encodes strength of an LBP[66]. $VAR_{r,p}$ is defined as

$$VAR_{r,p} = \frac{1}{N}\sum_{i=1}^{N}(x_n - \mu)^2 \tag{2.19}$$

Where x are the neighbouring pixels and μ is the mean gray level of the neighborhood. $LBP^{riu2}_{1,8}$ histograms are one dimensional. Bin i refers to the number of times LBP i occurred in an image. When incorporating $VAR_{r,p}$ they use a two dimensional histogram where now bin *(i,j)* is the count of LBP i and VAR j occurring on the same pixel. These histograms are denoted $LBP_{r,p}/VAR_{r,p}$.

$VAR_{r,p}$, by definition, is naturally rotationally invariant. Also, as this is calculated relevant to a local mean it retains some grey-scale invariance. The contrast value is quantised to 8 levels to avoid sparseness in the final histograms [66].

2.6.2 Local Ternary Patterns

While invariant to monotonic changes in the grey-scale of an image, LBP is sensitive to noise and non-monotonic intensity functions. It also cannot distinguish uniform regions. This is due to the single comparison with a central pixel.

The Local Ternary Patterns (LTP) operator proposed in [67] offers a noise resistant extension. This is achieved by adding a second *tolerance* threshold. The tolerance threshold changes the patterns from a binary two-valued LBP string to a three valued ternary string. In these codes a pixel is assigned 0 if it is within the tolerance threshold of the central pixel, -1 if less than and 1 if greater. The advantage being that it allows the pattern to be robust to noise and distinguish near uniform and uniform neighbourhoods with an appropriately selected tolerance. However this does remove some of the invariance to grayscale changes [67]. This process is usually separated into a positive and negative part, where a pattern is generated with a positive and negative LBP function on a neighbourhood.

$$LTP_{pos} = \sum_{i=1}^{N} p_i \geq (p_c + t) \tag{2.20}$$

$$LTP_{neg} = \sum_{i=1}^{N} p_i \leq (p_c - t) \tag{2.21}$$

Where t is the tolerance threshold. The process is visualised in Figure 2.24, note how the values within the threshold range are assigned 0 in both patterns.

The positive and negative patterns, across and entire image, can be used to form positive and negative histograms which can then be concatenated or expressed jointly in a 2D histogram [67].

When combined with a grid based classification framework for face recognition their experiments showed modest improvements over their LBP counterparts [67].

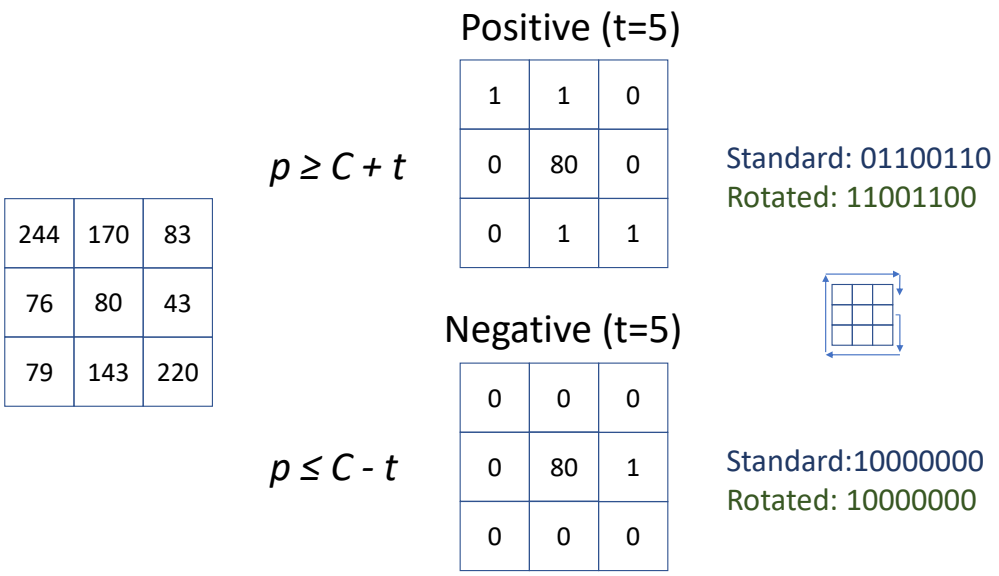


FIGURE 2.24: A neighbourhood processed into its positive and negative Local Ternary Patterns.

LTP has also shown promise in areas such as content based image retrieval [75] and medical imaging [76].

This method has been even further extended into methods such as the Extended Local Ternary Pattern [77] or the Relaxed Local Ternary Pattern [78].

2.6.3 Improved LBP

The observation made by the Improved LBP (ILBP) is that the original LBP encoding does not represent the local structure as well as is possible. To address this they instead threshold the neighbourhood by a weighted local mean denoted m. This is expressed as:

$$m = \frac{1}{n+1}(\sum_{i=0}^{p-1}(p_i + C)) \tag{2.22}$$

with ILBP expressed as:

$$ILBP_{r,p} = \sum_{i=0}^{n-1} s(p_i - m)2^i + s(C - m)2^n \tag{2.23}$$

$$s(x) = \begin{cases} 1 & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{2.24}$$

Where, as in equation 2.17, n is the size of the neighbourhood of pixels, $p_1 ... p_n$ are the pixels in the neighbourhood and C is the central pixel.

This retains the invariance to illumination of the canonical LBP while improving the structural representation of each pattern. It does however increase complexity as they must also encode the central pixel. Thus ILBP has twice the patterns of standard LBP [68].

This is applied to face detection using the CMU-MIT and Yaleb datasets. They use a Multivariate Gaussian Distribution with a Bayesian decision rule. They show detection rates of up to 90% on CMU-MIT and find that the false rejects from YaleB are mostly images with large illumination distortion (i.e. non monotonic functions), such that the textural information is no longer present [68].

2.6.4 Completed LBP

There are certain questions surrounding LBP. How does the simple code confer so much of the discriminative information? How much information is lost in the transform? In [79] they attempt to answer these questions by offering a "generalised complete" form of the LBP known as Complete LBP (CLBP).

Their process involves decomposing each neighbourhood into its centre and Local Difference Sign Magnitude Transform (LDSMT). This effects a three level representation of a local neighbourhood: firstly the Central (CLBP_C) which encodes

the central pixel. Secondly the Sign (CLBP_S) which encodes the sign of the difference between the central pixel and its neighbours (this part is identical to traditional LBP). Finally the Magnitude (CLBP_M) which encodes the magnitude of the difference between the central pixel and its neighbours. See Figure 2.25

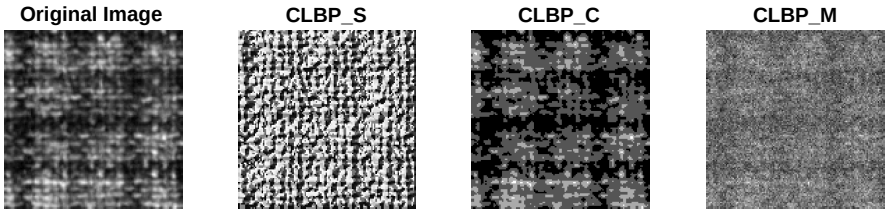


FIGURE 2.25: An image converted into its $CLBP\_S$, $CLBP\_C$ and $CLBP\_M$ counterparts. ($CLBP\_C$ is quantised to 16 levels, $CLBP\_M$ is approximated for visualisation)

With the LDSMT applied to every local neighbourhood in an image three primary transformations are produced: CLBP_C, CLBP_S and CLBP_M. In [79] these are combined in multiple different manners. Primarily histograms are formed jointly between two of the methods. For instance CLBP_S/M is a two dimensional joint histogram of instances of CLBP_S and CLBP_M. They found that with a 3D joint histogram CLBP_S/M/C stronger classification performance was obtained over traditional LBP. They also found that the Sign of the transform held the majority of the discriminative information. However contrary to traditional LBP they find the magnitude is also useful [79].

This representation has the traditional benefits of LBP and more. CLBP_S is identical to traditional LBP so retains grey-scale and rotational invariance. By the nature of the transform the complete feature (all three CLBP components, with no quantisation on CLBP_C (Figure 2.25 is quantised)) is also perfectly invertible (if CLBP_C is not quantised), and so can be considered in a dual sense.

2.6.5 Dominance LBP

Dominance LBP (DLBP) changes the way in which patterns are considered for the final histogram [80]. Instead of choosing the Uniform, or most likely, patterns to form the histogram; Dominance LBP considers the top $n\%$ most occurring patterns across a dataset and use these as the patterns which are histogrammed. Again, all other patterns are grouped into a bin appended to the end. The premise is that the which appear the most should provide a more accurate index for the data concerned [80], an example of an images dominant patterns is shown in 2.26.

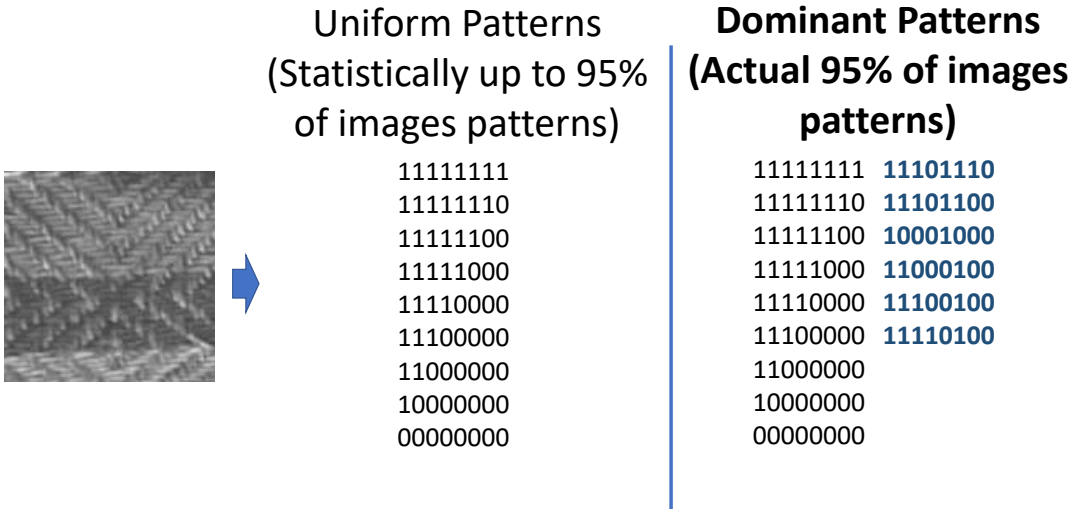


FIGURE 2.26: Example of uniform patterns vs dominant patterns on an image.

A common facet of DLBP, present in Figure 2.26 is that all Uniformity 2 patterns are still present. Also note how the supplemental dominant patterns are all of Uniformity 4.

In [80] the authors augment these features with responses from circularly symmetric Gabor filters. They show good results classifying members of the Outex, Brodatz and CUReT texture databases, particularly under sub-optimal conditions such as random rotation and noise.

2.6.6 Strength LBP

In [72] they present a novel formulation of LBP known as "Strength" or "Scale Adaptive" LBP. Where traditional LBP considers patterns at a single radius across an entire dataset, "Strength" LBP considers patterns at multiple radii per pixel with the same number of points. Each of these patterns has its strength calculated as in Equation 2.25:

$$Str = \sum_{i=1}^{p} |p_i - p_c| \tag{2.25}$$

Where p_c is the central pixel and p_i are the neighbours. The pattern with the highest strength is the one selected to represent that pixel. When applied to classification of images of skin defects both benign and malignant they show strong results [72].

2.6.7 Center-Symmetric LBP

Much of the advantage of LBP, and its extensions, is the invariance to any monotonically increasing change in the image intensities. This invariance is in effect a property of coding pixels relative to each other since it is these relations which are independent of typical intensity transforms).

Center-Symmetric LBP (CS-LBP) was presented as a region of interest descriptor based on the Scale Invariant Feature Transform [8] and LBP [81]. It uses binary thresholding to form a local histogram of an interest region. We refer the reader to [82] for more information on interest region detection.

The CS-LBP operator uses comparisons between diametrically opposed pixels in a neighbourhood to form its patterns. This has the characteristic of generating a 4 bit code and as such there are only 2^4 patterns. Each region of interest image is

separated into a cartesian grid of cells, presented in the paper as 3×3 or 4×4. Each cell generates a histogram based on the responses of a CS-LBP operator on the pixels inside that cell. The feature for an image comprises of a concatenation of these histograms after normalisation and weighting steps.

After parameterisation of their operator, namely the radius of the neighbourhood and the number of pixel comparisons, they show object classification results on the Pascal Visual Object database. They show CS-LBP out-performs SIFT in a large number of cases, while significantly decreasing the computation time of an experiment [81].

2.7 Colour analysis

When humans look at images the colours present form a significant part of our recognition process [83]. Colour histograms are one of the computer vision analogues to this process. These have shown good performance in areas such as content based image retrieval [84], human face detection [85] and medical imaging [86].

A colour histogram is a representation of the distribution of intensities in an image. For the simplest form we shall first consider an 8 bit grayscale image A where each pixel is represented by a single integer between 0 and 255. If we were to form an intensity histogram of this image it would be one dimensional and have 256 bins where bin i $(1 \leq i \leq 256)$ contains the number of times intensity i appeared in A. If we now consider a new colour image B in the RGB colour space each pixel now has three values each representing the amount of Red, Green and Blue said pixel is comprised of. The histogram of this image is now three dimensional with each dimension of length 256. In this case bin *(i,j,k)* corresponds to the number of times intensity i in R, j in G and k in B occur on the same pixel.

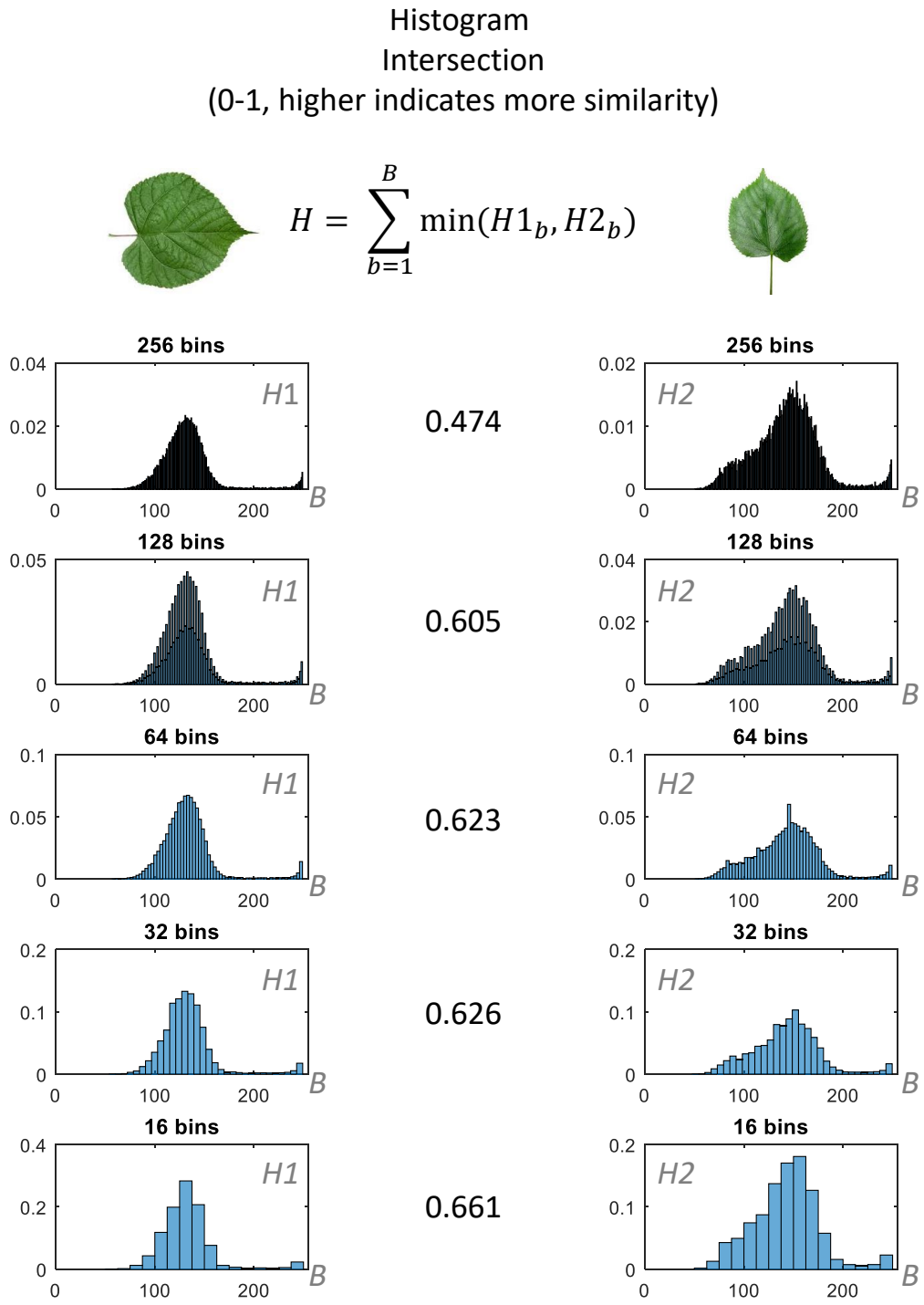


FIGURE 2.27: Examples of histogram quantisation on images of Leaves

The above definition allows for a histogram where each intensity, or intensity RGB triple, indexes to a single bin. While this is a complete representation it may not always be the best course of action. Perceptually intensities with close values have a very similar appearance. For example consider the two RGB triples *(0,0,1)* and *(0,0,0)*, they both represent almost exactly the same colour however index to a different bin. This can be considered a sensitivity, one which is especially prone to noise. This can be alleviated by quantisation of the colour space. If we now consider a 64 bin colour histogram which still represents all possible intensities: [1,2,3,4], [5,6,7,8] are aggregated as single bins. This, in effect, says that intensities close to each other are similar enough to be considered the same. In reality this can be useful, consider as an example leaves. In Figure 2.27 we show two visually similar leaves. We histogram the green channel of these leaves and find that as we increase quantisation (increase the number of intensities indexed to each bin) we also increase histogram similarity.

Quantisation does not completely solve the issue. In the example of a 64 bin histogram intensities 5 and 6 are grouped in the same bin, however 5 and 4 are in a different bin despite having the same difference. One method for alleviating this is histogram smoothing. If a quantised histogram is smoothed by an appropriately chosen function (e.g. Gaussian, cosine) before bins are aggregated. This has the effect of allowing intensities to "bleed" into other neighbouring bins [17].

[redacted]

[redacted]

- [redacted]
- [redacted]
- [redacted]
- [redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

2.9 Texture synthesis

Another angle to consider texture analysis from is: taking images and devolving them into some form to better understand them. The inverse problem is another area of interest: given some form of understanding can you create an image. This is where the topic of image synthesis arises. Specific to this thesis is that of image synthesis from texture - given some understanding of the texture of an image, can we: synthesize plausible examples of it, recover the original purely from the textural information or synthesize information in the image which has been lost (noise removal, hole filling). We will now present a sample of the methods presented on this problem.

2.9.1 Exemplar based texture synthesis

Efros and Leung [90] present an exemplar based (that is, based on known examples) synthesis algorithm. The method is based on local image patches, with unknown pixels being assigned values based on probabilities determined by patches of original image. The texture of the image is modeled as an MRF, the assumption being that a pixels intensity given that of its neighbours is independent from the remainder of the image.

Given an unknown pixel and a window around it (the width of the window being a parameter) the objective is to choose the most similar patch from the rest of the image and assign the value of the central pixel of that patch to our unknown. To achieve this they construct the set of all “similar” patches (similarity is determined

by the Sum of Squared Distances weighted by a Gaussian Kernel). The histogram of the central pixel values of this set forms an approximation of the conditional probability density function (pdf) of our unknown pixel. The most likely pixel value is then chosen as shown in Figure 2.31.

An issue in application, for example image hole filling, is that most pixels which require synthesizing have members of their window which are also unknown. This is accounted for by only measuring patch similarity on the known pixels of the window around them, and normalising the error in the conditional pdf based on the number of known pixels [90].

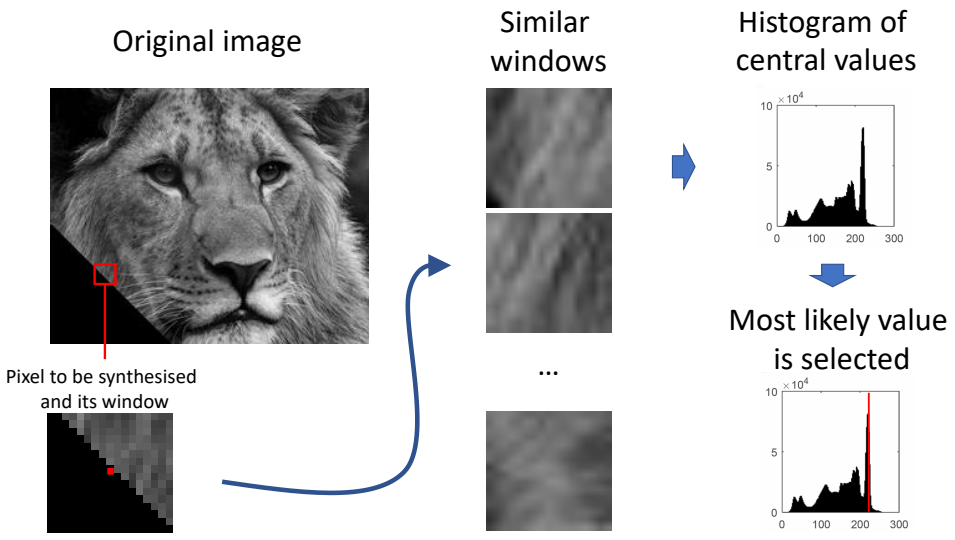


FIGURE 2.31: A pixel being synthesized by the Efros and Leung algorithm.

Efros and Leung show good results in image hole filling, synthesizing plausible large textures from a small sample and extrapolating images into larger versions.

Diffusion is another method for image synthesis based on following contours (specifically *isophotes* or lines of constant intensity) [91]. While good at producing large scale linear structure diffusion unfortunately fails to effectively reproduce fine detail texture, instead producing a blurred effect [92] [93]. While the technique may fail in our specific application, the principles are still a powerful tool. In the work of Criminisi et Al. [93] they use these principles to extend the work Efros et Al.

Their method uses isophotes to inform the order in which pixels are synthesized allowing them to more effectively capture larger scale linear image structures. Their method, when applied to removal of foreground objects in digital images, showed good results compared to the current state of the art at publication. Bugeau and Bertalmio [92] go further by fully combining exemplar based texture synthesis and diffusion in their work. They form two images, firstly a structure image using the diffusion method of [94]. Secondly a texture image using a modified version of the algorithm in [93] and combine the two. They show promising results on a number of images.

2.9.2 Reconstruction using Complex Wavelet Coefficients

Portilla and Simoncelli [95] offer a technique using the statistics from a complex wavelet transform. They use a steerable pyramid decomposition [96] augmented with quadrature pair filters to extract oriented frequency information from an image at a number of scales. The statistics of these responses are used to form a set of statistical constraints which define their model of texture.

They synthesize textures using their method termed "synthesis by analysis". In simple terms they begin with the model of the texture they wish to synthesize containing all of the statistical constraints, and a Gaussian white noise image. They then build the model of the Gaussian white noise image and iteratively force the model to satisfy each of the statistical constraints which "warps" the white noise image into a statistical counterpart of the original. The quality of the reconstruction is then a measure of the quality of their model. Firstly in their results on the images which would not be considered texture (a face, a bullseye and a crowd) they capture some of the local structure yet the image as a whole is somewhat scrambled rendering it unrecognisable. Secondly on natural texture images their

results are effective yet some distortion is still present. Finally they show exceptional synthesis of highly random high frequency textures such as animal fur and wood grain.

2.9.3 Reconstruction from Feature Points

Lillholm et al. [97] present an image modeling and reconstruction framework based on statistics. Initially their work presents the concept of an image *metamerism* class. Colour *metamerism*, the original use of the term, refers to how varying colour spectra can appear identical subject to changes in viewing conditions. In this case we are discussing *spatial* metamerism: for images to be of the same class the same spatial and corresponding *measurements* must be obtained. They show that given a set of local statistical measurements, minimising a norm given these they can synthesize the "simplest" member of a metameric class [97].

Further, and more pertinent to this project, they apply their minimisation subject to local measurements algorithm to reconstructing images from feature points. Namely they extract "blob" [98] and "edge" [99] points in scale space from images. As only a sample of all points in an image is required for reconstruction, method of point selection is key. Thus point selection is based on scale-normalised feature strength (a measure of the information in a particular feature point). They experiment with minimising the $L2$, $\|\nabla L1\|$ (L1 gradient magnitude) and $\|\nabla L2\|$ (L2 gradient magnitude) norms to synthesize images. They show that number of points chosen is key. Blob points capture most of the stochastic complexity of the images rather quickly, however at large numbers of selected points edge features produce a much more accurate reconstruction.

2.9.4 Texture synthesis conclusions

We have offered a selection of the methods currently available in the area of image synthesis from texture. The method we offer in Chapter 7 operates on spatially located features similarly to the work of Efros et al., whereas the method in Section 2.9.2 operates on global measures. The features we and Efros et al. reconstruct from are localised in the image space, that is we know *where* they are: the algorithm then ascertains *what* they are. In the case of [95] and the method begins with forming a global statistical model of an image. This model is a description of the properties of the images texture: there is no notion of what a pixel is until an image synthesis algorithm is applied. This approach produces images which are perceptually similar or statistically identical to a prior example, however they are entirely synthetic.

2.10 The Rank Transform and extensions

Image correspondence is the problem of detecting corresponding entities across two or more images of the same scene. The Rank transform is a method proposed in this area. This method and its extensions, while not texture analysis algorithms, share similarities with our work in Chapter 6. In the succeeding Sections we review the relevant literature.

2.10.1 The Rank Transform

The work of Zabih and Woodfill [100] presents their novel image transform for detecting corresponding pixels between scenes. They offer a two part approach: firstly the *Census* transform which assigns to a pixel a bit string denoting which neighbouring pixels which are less than it. Secondly the *Rank* transform which

assigns to a pixel its rank in some square window around it of diameter d. See Figure 2.32.

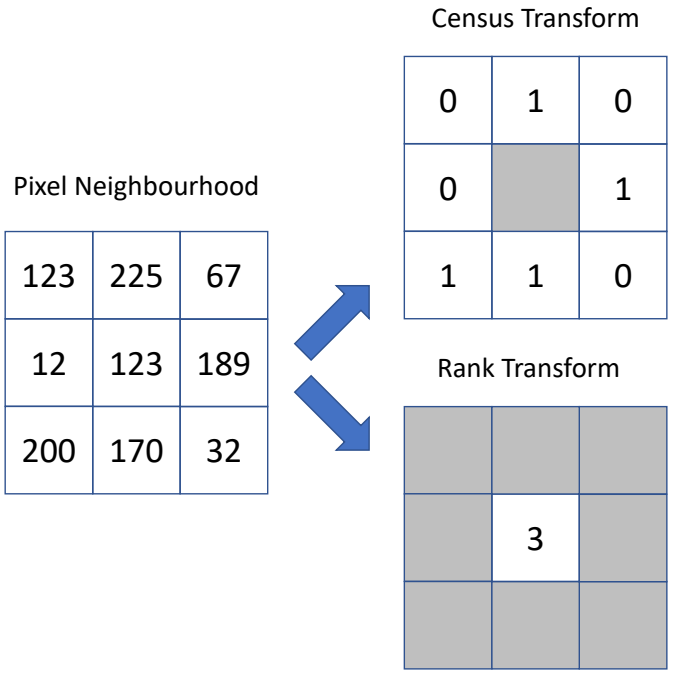


FIGURE 2.32: The Rank and Census transforms of a single pixel neighbourhood.

Similarly to LBP both the Rank and the Census transforms are non-parametric as such are invariant to monotonic changes in gray-scale. With the pixels of two images converted into their rank and census transforms, correspondence can then be measured. This is measured differently for each of the two transforms. To compare two Census transformed pixels, the hamming distance between the bit strings is used. For the Rank transform L1 correlation is used between two rank transformed images. In their experiments the authors show lower error rates than some previous art methods.

The Census transform does bare resemblance to LBP; however it differs in some key points. Firstly the bit strings in the Census transform are read off from an arbitrary, consistent across an image and within an experiment, point. LBP requires a specific concatenation so as to obtain rotational invariance, a property

not exhibited by the census transform. Secondly only individual strings are compared, there is no consideration of forming an index for a whole image such as the histogram in LBP. The census strings are instead used individually and as spatial information for the Rank transform [101].

2.10.2 The Complete Rank Transform

Presented in [101] the Complete Rank Transform (CRT) is based on the observation that the Rank transform wastes a large amount of information. It extends the Rank and Census transforms of Zabih and Woodfill [100] to capture a maximum amount of local information. Instead of assigning a pixel its rank in a local neighbourhood they assign it a signature comprised of the complete ranking of its local neighbourhood. See Figure 2.33.

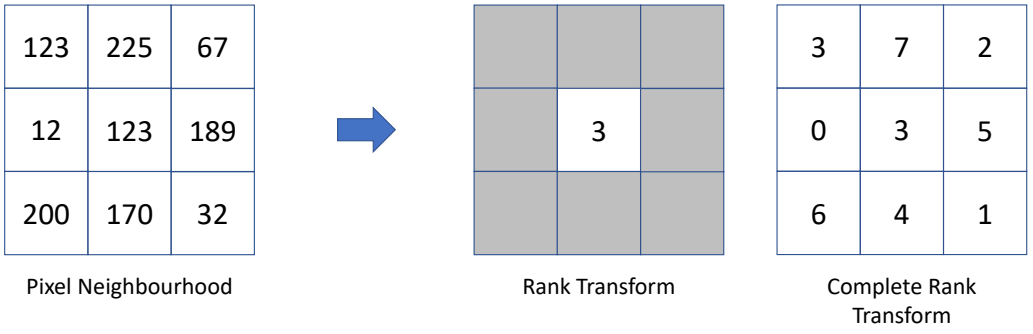


FIGURE 2.33: Comparative example of the Rank and Complete Rank transforms in a 3×3 neighbourhood.

This description contains the maximum amount of comparative information between all pixels in a neighbourhood. They also present a Complete Census Transform (CCT), however it occurs that the CRT is an extrapolated version of the CCT and therefore contains identical information.

This descriptor is, again, designed for Optical Flow/Image Correspondence algorithms (following entities or pixels through a series of images). Their experiments show improved results over the original Rank and Census transforms, with little to no computational cost increase.

The CRT bares resemblance to our work in Chapter 6. We suggest that in the same way the Census transform differs from LBP, our Sudoku transform differs from the CRT. This is detailed more completely in Section 6.1.4

2.11 Overall conclusions

There has been much interest in LBP since its inception. A recent study in 2017 has compared performance of 32 different LBP approaches [102]. Here we intend to capitalize on this basis since it has the properties we seek [redacted] Further, it will lead to our new approach for describing texture based on LBP. We have also described techniques which appear suited to this application, but we have yet to exploit. While we note that CLBP is perfectly invertible we have not yet considered this in the context of other methods: this will be explored in Chapter 7. While representation has been the focus of most studies until now there been very few approaches to reconstruction as we shall find later. [redacted]

Chapter 3

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

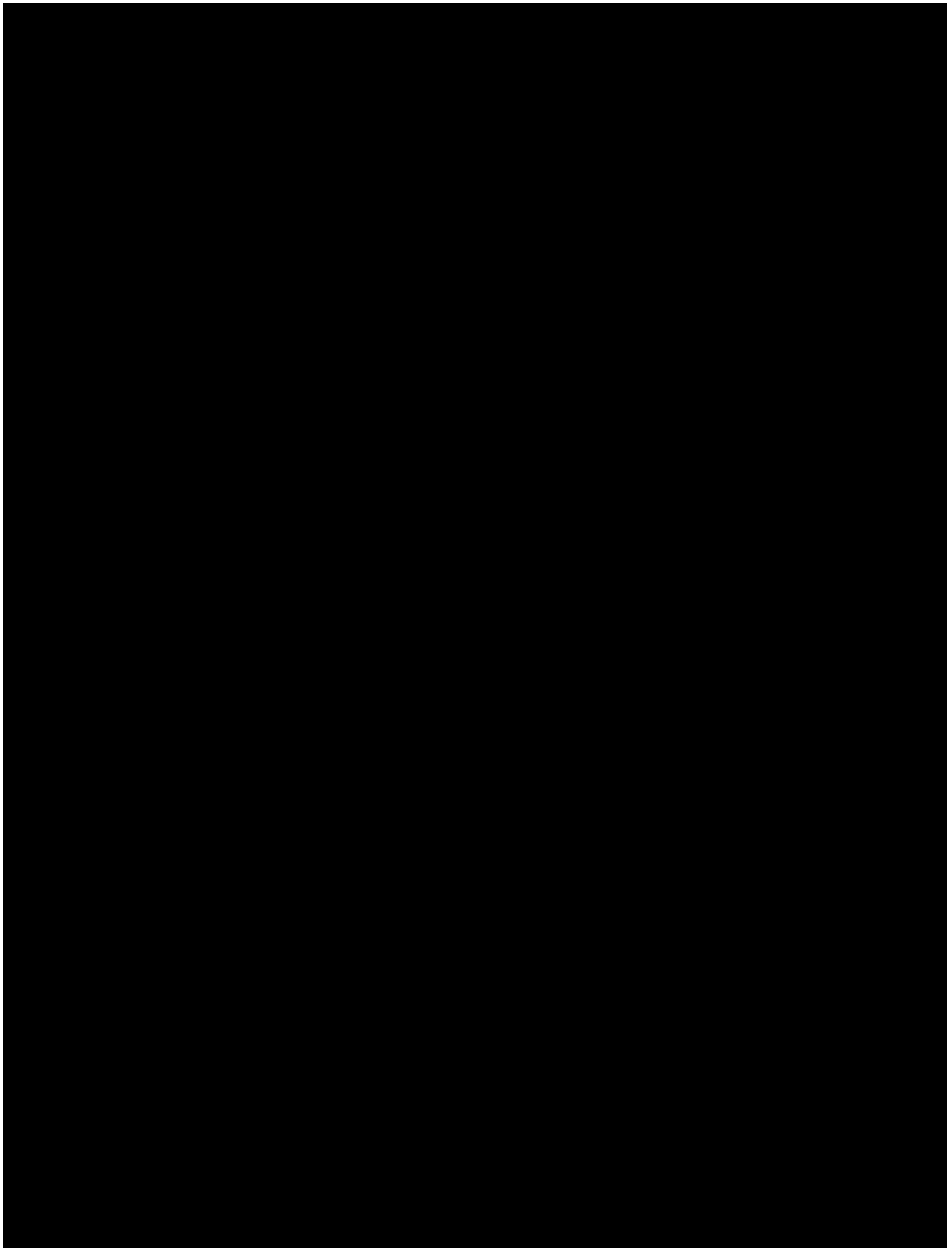
[redacted]

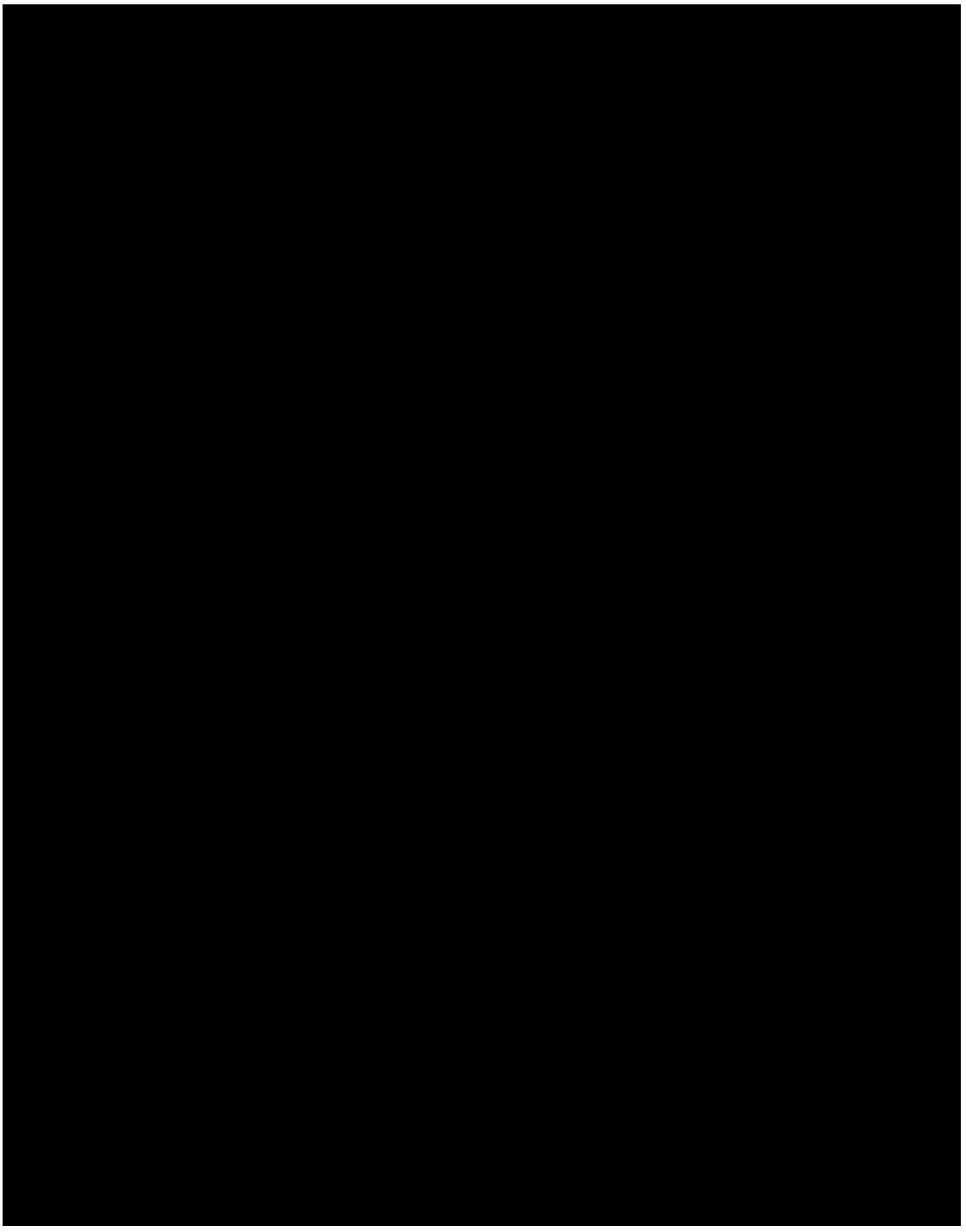
[redacted]

[redacted]

[redacted]

[redacted]





[redacted]

Chapter 4

[redacted]

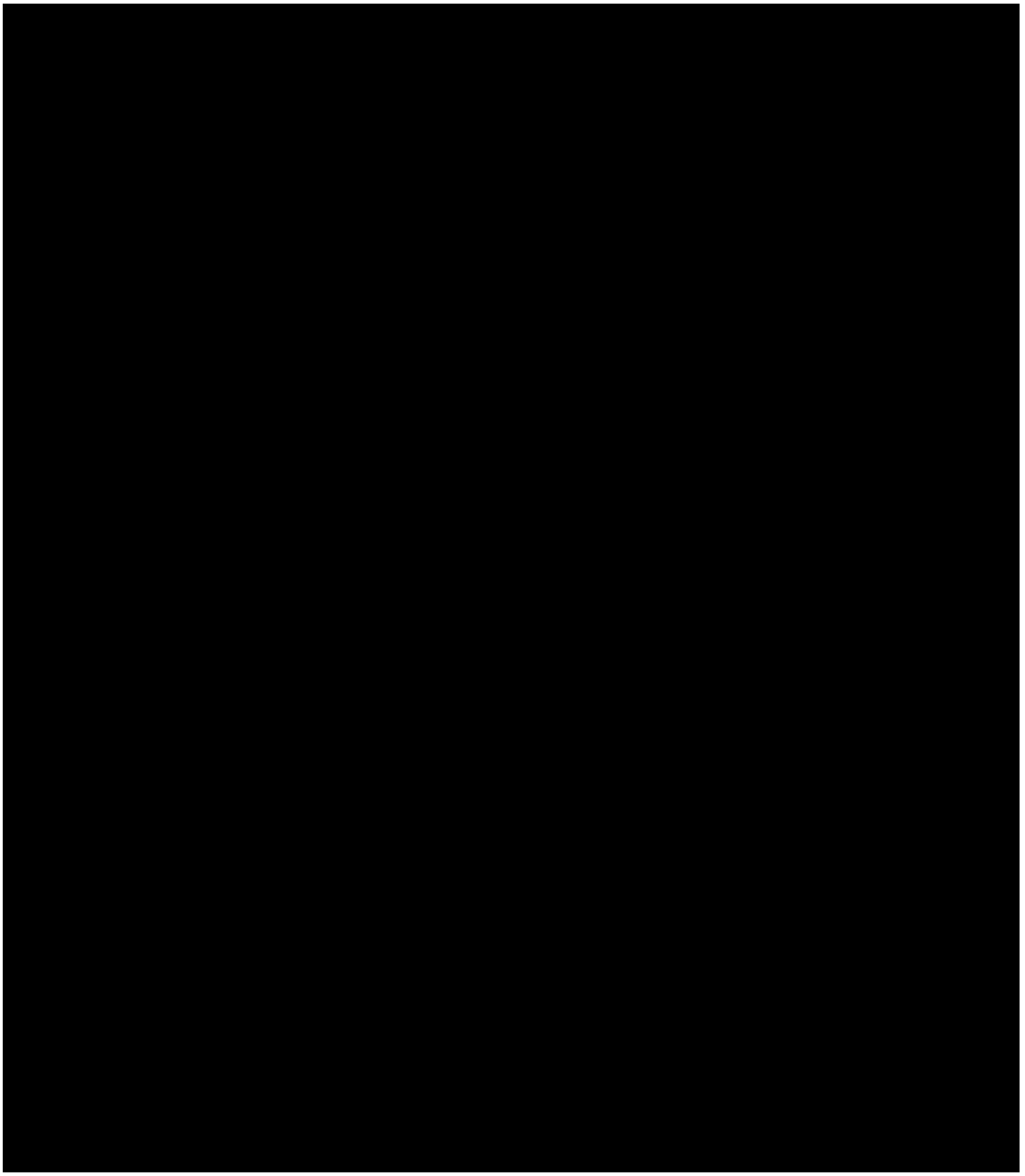
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

[redacted] [redacted]

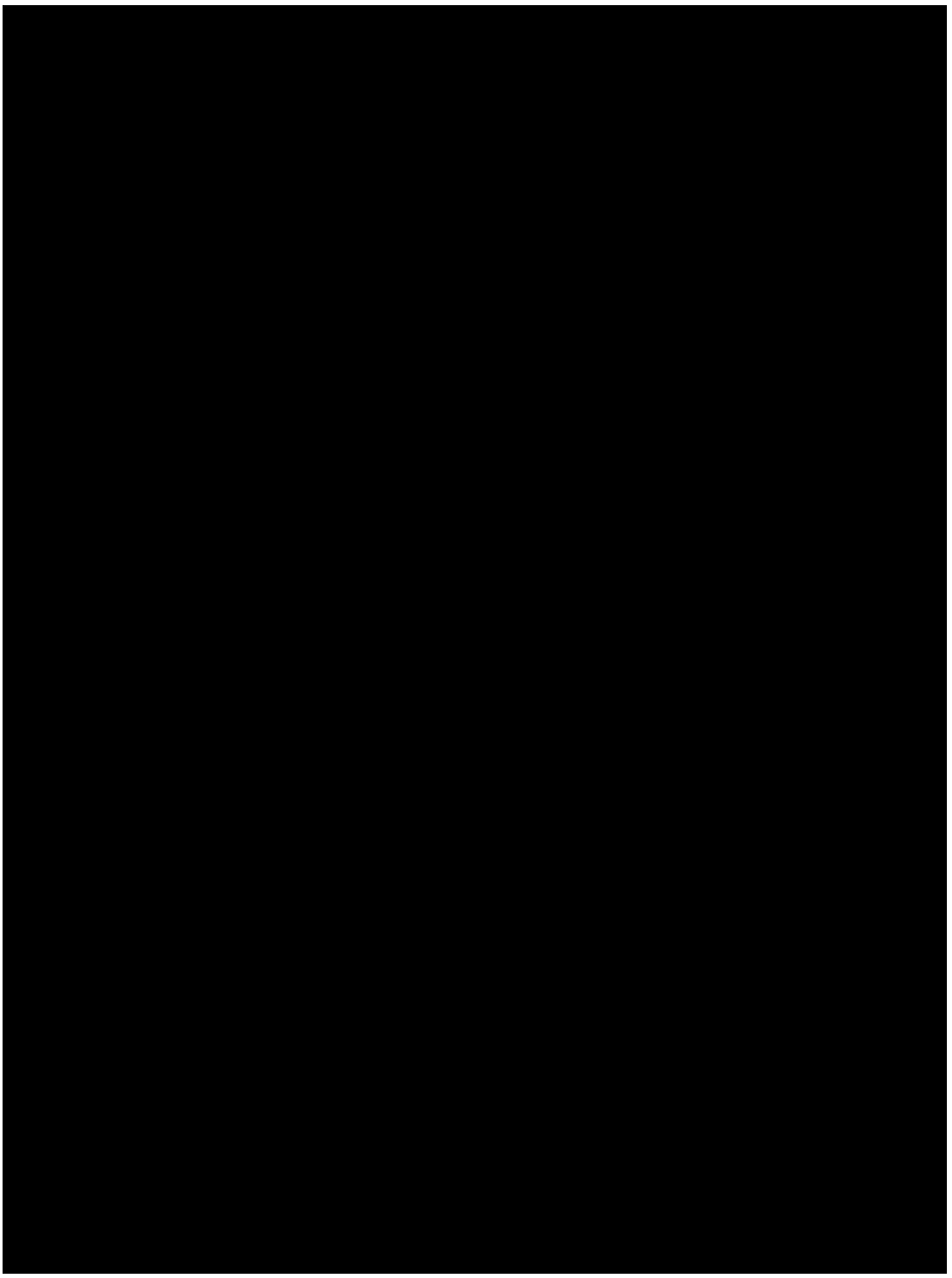
[redacted]

[redacted]

[redacted]

[redacted]

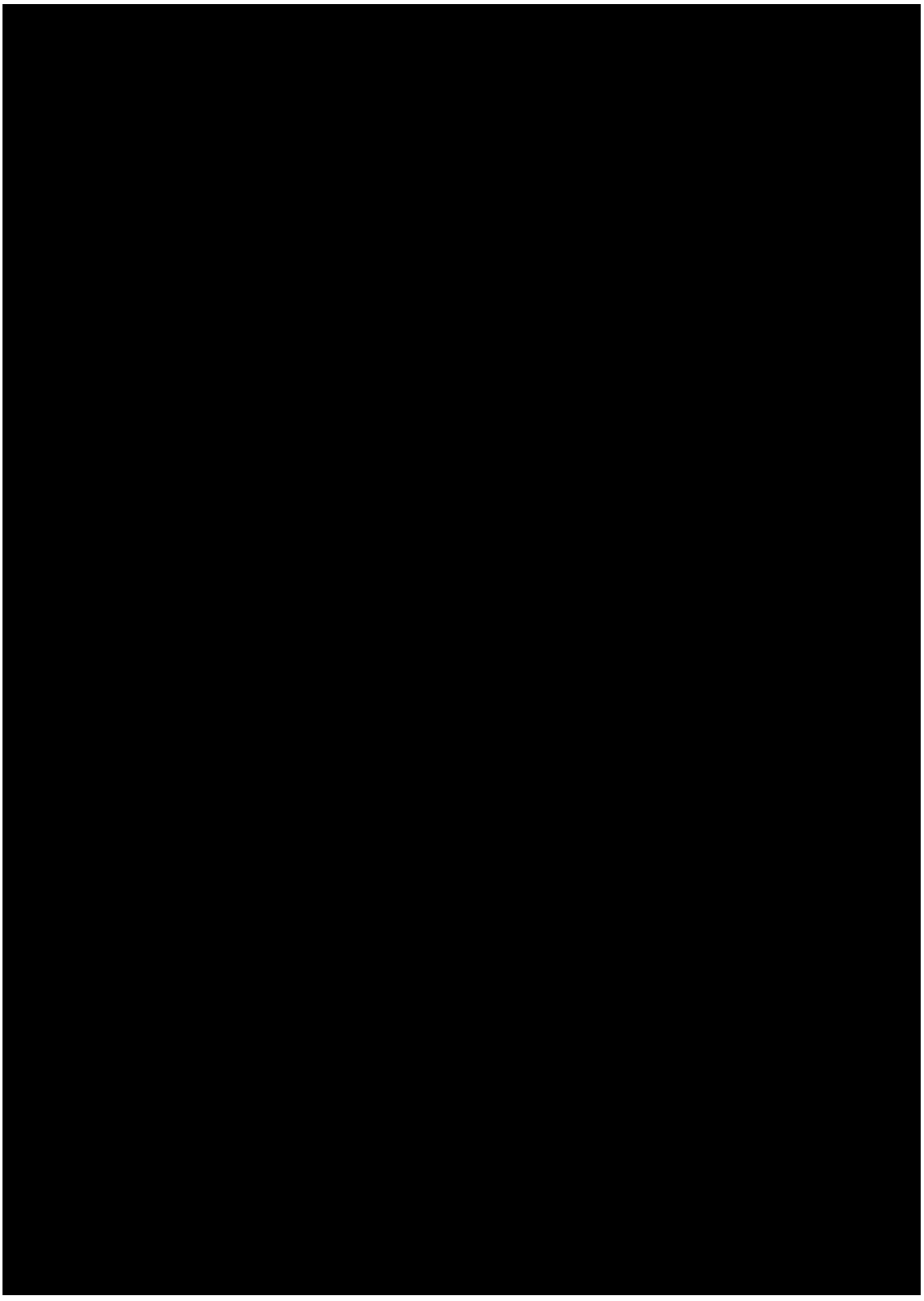
[redacted]



[redacted]

[redacted]

[redacted]



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted] [redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

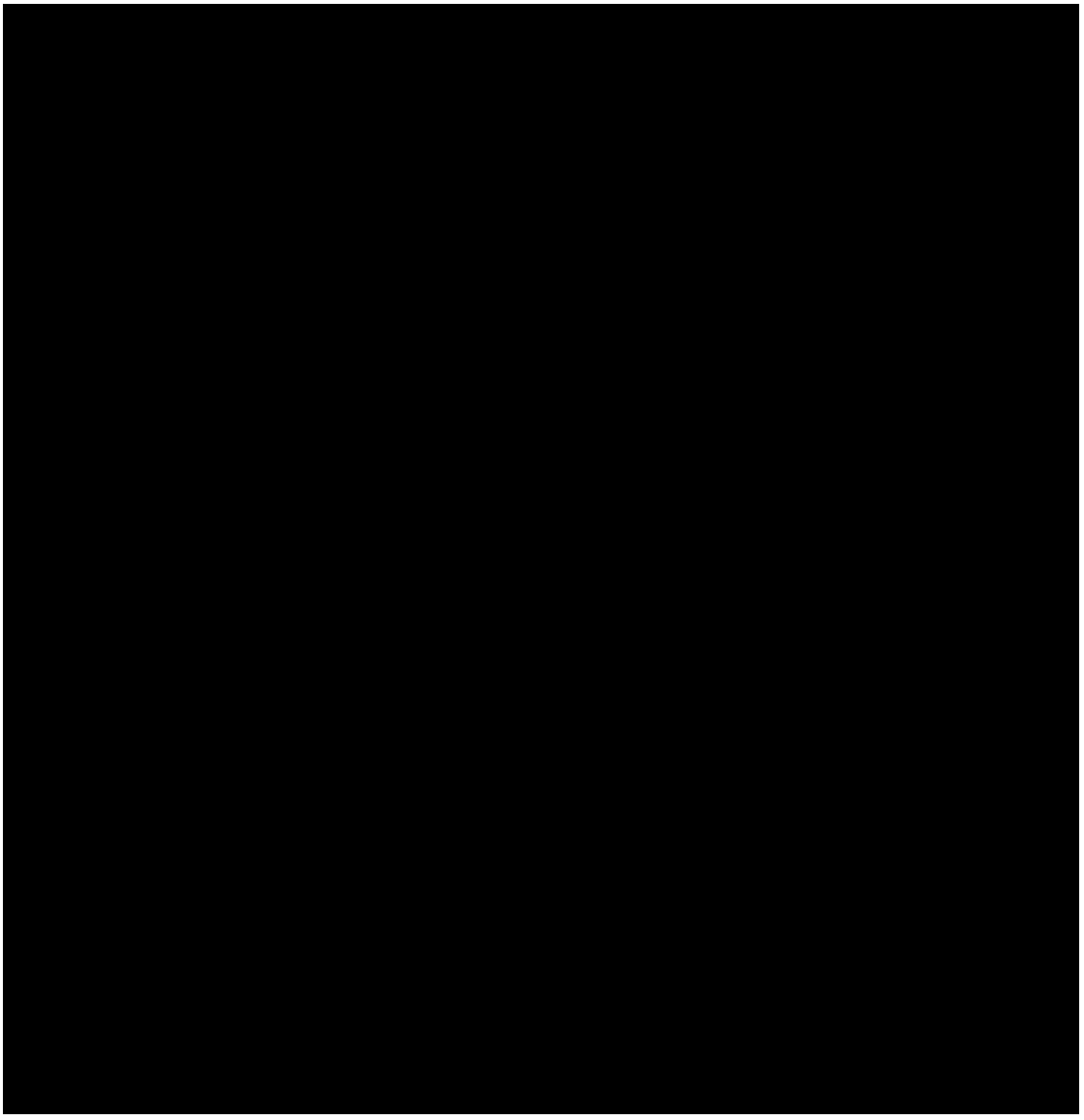
[redacted]

[redacted]

[redacted]

[redacted]

[redacted]



[redacted]

[redacted]

Chapter 5

[redacted] contributions on classification

[redacted]

[redacted]

In this Chapter we will evaluate a classifier called the Support Vector Machine or SVM. The classic SVM treats all feature vectors as vectors in Euclidean space. In Section 5.1 we will discuss the mechanics and component parts of the SVM from both a mathematical and real world perspective. Secondly we will discuss our experiments and the results obtained. Finally we will evaluate our work and form our conclusions.

We are aware that there are available packages such as *libsvm* to investigate use of this method. [redacted]

5.1 The Support Vector Machine

The SVM is a way of separating two sets of points in space. This is achieved by forming a separating hyperplane between the two sets which has the *maximum symmetric margin*. This maximises both classes distance to the decision boundary while having the additional benefit of optimising directionality of the decision boundary within the space [16].

5.1.1 Primal form

Let us define a set of points x with two classes positive and negative (for this case positive and negative are linearly separable) and a vector w normal to the separating hyperplane and offset b such that

$$\vec{w} \cdot x + b = 0 \tag{5.1}$$

For all x which lie on the hyperplane. For all x in the positive class:

$$\vec{w} \cdot x + b \geq 1 \tag{5.2}$$

and for all x in the negative class:

$$\vec{w} \cdot x + b \leq -1 \tag{5.3}$$

These two equations enforce the *margin*. Now we will assign a class label y to each class which has the value *-1* for negative and *1* for positive. If we then multiply Equations 5.2 and 5.3 by y we find we can enforce the margin while constraining w using just one equation

$$y(\vec{w} \cdot x + b) \geq 1 \tag{5.4}$$

The width of the margin is defined as:

$$\frac{2}{\|\vec{w}\|} \tag{5.5}$$

As this is the element we wish to maximise, we now have all of the pieces we require to can wrap this as a quadratic programming minimisation as follows

$$min \frac{1}{2}\|\vec{w}\|^2 \quad s.t. \quad y(\vec{w} \cdot x + b) \geq 1 \quad \forall x \tag{5.6}$$

A 2 dimensional primal SVM is shown in Figure 5.1.

In the example shown in Figure 5.1 the positive and negative classes are perfectly linearly separable. $\vec{w}$ is the normal to the hyperplane and b is the y intercept. This is the primal form of the SVM. Equation 5.6 can be solved for any linearly

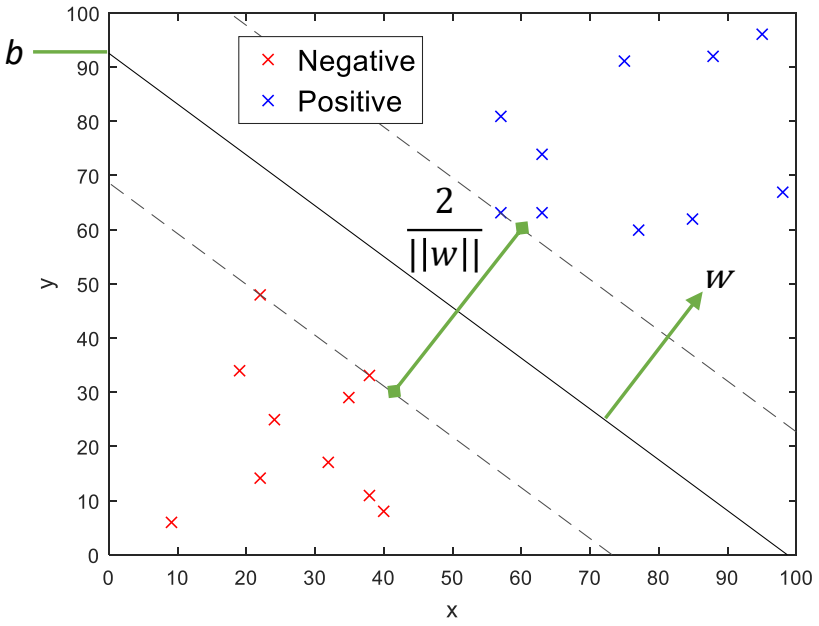


FIGURE 5.1: 2-Dimensional example of a primal SVM.

separable datasets by mapping it to a quadratic programming algorithm [16, 109]. This will calculate an explicit vector w and offset b over the input vectors. These can then be used as in Equation 5.1 with a test sample x and the sign of the result will determine which class x belongs to. The classifier can be biased towards either class by varying the threshold above or below *0*.

5.1.2 Dual form

While the primal form is very useful from an intuitive standpoint. It becomes extremely inefficient when the dimensionality of the input vectors becomes large or if we encounter a set of data which are not linearly separable. To solve this we use the dual form

$$\begin{aligned} find \quad \alpha_1 ... \alpha_n \quad s.t. \quad max(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i \cdot x_j) \\ \sum_{i=1}^{n} y_i \alpha_i = 0 \quad \alpha_i \geq 0 \quad \forall \alpha \end{aligned} \tag{5.7}$$

Equation 5.7 is exactly equal to Equation 5.6, however the shape of the solution is different. Instead of finding an explicit weight vector w Equation 5.7 finds a set of weights α which when applied to the input vectors form an implicit separating hyperplane. Note that the majority of the α will be *0* as they are far from the hyperplane and vectors with $\alpha > 0$ are known as the Support Vectors. This optimisation is entirely dependent on dot products between vectors instead of the raw vectors themselves which allows us freedom with the size of our input vectors.

The quadratic programming mapping for this problem is calculated over the dot products between all vectors in the training set. The output is a set of weights α. As the alphas in combination with the input vectors form an implicit separating hyperplane the classification function is as follows for a query sample x_q

$$\sum_{i=1}^{n} y_i \alpha_i x_i \cdot x_q + b \tag{5.8}$$

Where the sign of the result of Equation 5.8 determines the class. Specifically for the linear case, the hyperplane normal w and bias b can be calculated explicitly as in Equation 5.6 using the following.

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i \tag{5.9}$$

$$b = y_k - w \cdot x_k \tag{5.10}$$

Equation 5.10 only holds for x_k with $\alpha > 0$. From Equations 5.8, 5.9 and 5.10, for a linear SVM, a query can be performed with a single dot product as w can be calculated offline as in Equation 5.11.

$$\sum_{i=1}^{n} y_i \alpha_i x_i \cdot x_q + b = w \cdot x_q + b \tag{5.11}$$

5.1.3 Non linearly separable data

A further extension of the SVM is known as "Slack" or "Regularisation". Regularisation is a common practice in many optimisation and machine learning algorithms. This is effected by bounding the output of the optimisation such that none of the input data become too highly valued. Alternatively it can be viewed as a method for incorporating more of the training data into the final solution. This helps avoid overfitting and allows us to have intentional misclassification. To implement this in the SVM formulation Equation 5.7 becomes

$$find \quad \alpha_1...\alpha_n \quad \epsilon_1...\epsilon_n \quad s.t. \quad max(\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i \cdot x_j) \\ \sum_{i=1}^{n} y_i\alpha_i = 0 \quad \lambda \geq \alpha_i \geq 0 \quad \forall\alpha \tag{5.12}$$

Where $\epsilon_1...\epsilon_n$ are the slack variables, abstracted within the machinery of the algorithm, assigned to each vector by the solver and λ (also known as C) is the regularisation parameter. This has the effect of bounding the size of the $\alpha_i...\alpha_n$ (See Section 2.3) by λ. As only a maximum λ of each dot product can be used in the final solution, a larger proportion of the input data must be used to form the separating hyperplane. As a result the margin becomes softer. For an example see Figure 5.2.

The blue and red points are different classes with the circles being the support vectors. Note when $\lambda = 0.0001$ the classifier is deliberately misclassifying a point to allow for a more general solution. A few observations can be made with regards to this. Firstly using more of the input data in the solution implies a more general solution, as such a lower lambda should be desirable. Secondly as Lambda becomes smaller the number of non-zero alphas increases and consequently testing time

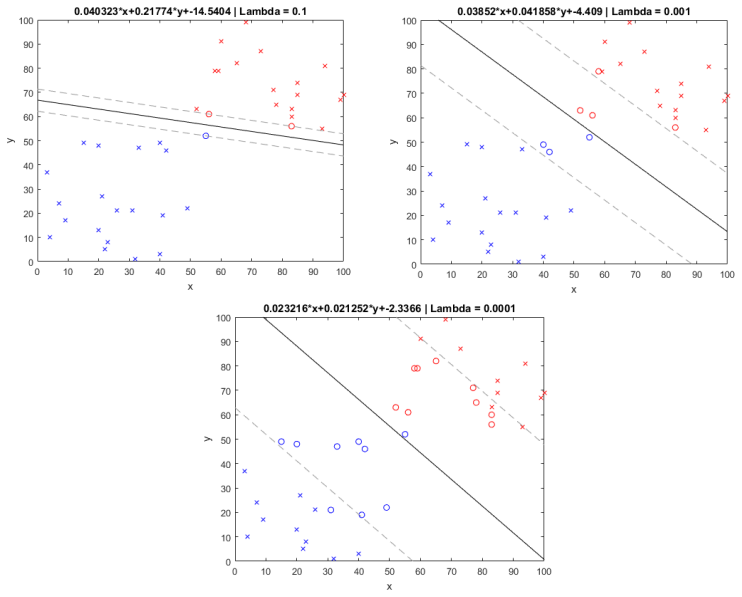


FIGURE 5.2: Examples of soft margin classification using an SVM.

increases, accordingly a higher lambda is desirable when time to compute is at a premium.

5.1.4 Forming non-linear decision boundaries

Kernels were proposed initially with SVMs as a small extension which could prove useful in a small number of circumstances [110]. Later it was discovered that kernels were extremely powerful and now considered to be a fundamental part of the SVM.

Kernels are a method for mapping your vectors from their current feature space (In the case of linear SVMs this is Euclidean) into some other feature space known as the Embedded Space. Some examples include Polynomial kernels which map features into a higher dimensional Euclidean space, or the Gaussian Radial Basis Function (RBF) which maps features into an exponential space.

An issue with this is that with large data sets or large feature sizes the complexity of the mapping can become extremely high. In the case of the RBF it is nearly

impossible to explicitly map your features. To overcome this we use a Kernel Function as in Equation 5.13.

$$K(u, v) = \phi(u) \cdot \phi(v) \tag{5.13}$$

A kernel function maps two vectors $(\vec{u}, \vec{v})$ to their inner product in another space ϕ. This allows us complete freedom in the design of our kernel as we never need to explicitly map our features into the other space, we only need to understand how our features relate to each other within said space.

One example is the L1 kernel in Equation 5.14 .

$$K(\vec{u}, \vec{v}) = 1 - \sum_{i=1}^{n} |\vec{u_i} - \vec{v_i}| \tag{5.14}$$

Where $\vec{u}$ and $\vec{v}$ are the two vectors and n is the length of each vector. This kernel function has the effect of implicitly mapping our vectors into a space where their dot product is equal to the L1 distance between the two vectors. As stated in Section 5.1.2 the optimisation is only dependant on the dot products between vectors, this definition removes the need to consider the actual space, only the relationship between each pair of vectors. When it comes to implementing a linear SVM, which can also be considered an SVM using a linear Kernel, we pass the quadratic programming function the set of dot products between all vectors in the set. So to use the histogram intersection kernel we simply pass the function the set of L1 distances between all vectors instead. To use this Equation 5.7 becomes

$$find \quad \alpha_1 ... \alpha_n \quad s.t. \quad max(\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)) \tag{5.15}$$

with classification function

$$\sum_{i=1}^{n} y_i \alpha_i K(x_i, x_j) + b \tag{5.16}$$

5.1.4.1 An intuitive example of kernels

Say we have 3 classes of data and we want to represent them with a single feature, as in Figure 5.3.

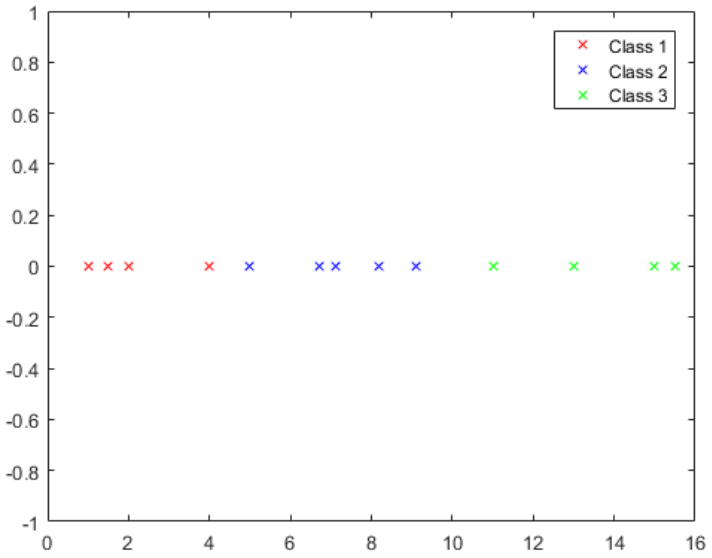


FIGURE 5.3: Three classes of data.

In this example the data are not entirely linearly separable, see Figure 5.4.

The red line separates all of class 1 from the rest and the green line separates all of class 3 from the rest, however there is no possible straight line which will separate class 2. To completely separate these classes we can use a polynomial kernel. To do this we project our features onto a quadratic curve, see Figure 5.5.

Now we can separate all 3 classes completely using single lines as in Figure 5.6.

It is worth noting that as we are remaining in Euclidean space the original lines of separation still exist. As such this kernel can never create a worse separation between classes.

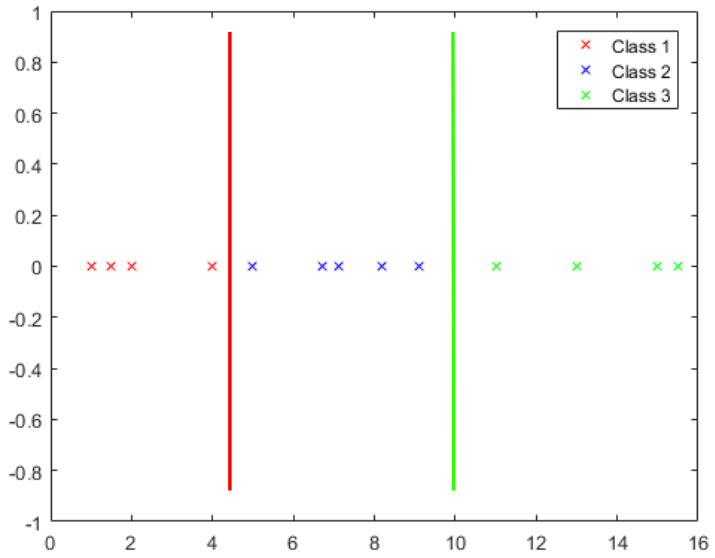


FIGURE 5.4: The separations between the three classes of data.

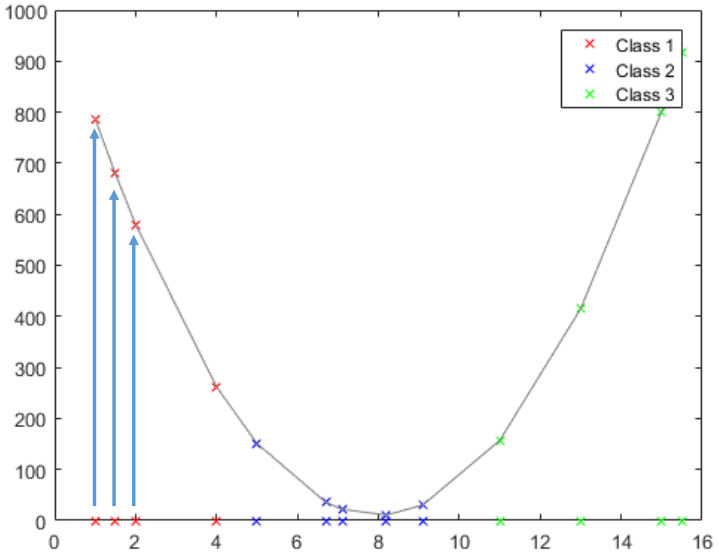


FIGURE 5.5: Projection onto a quadratic curve.

Now with this intuition of spatial projection we can revert our features back to their space in Figure 5.3. As long as we have a kernel function which tells us how our features relate in the embedded space in Figure 5.6 we never need to actually perform the projection.

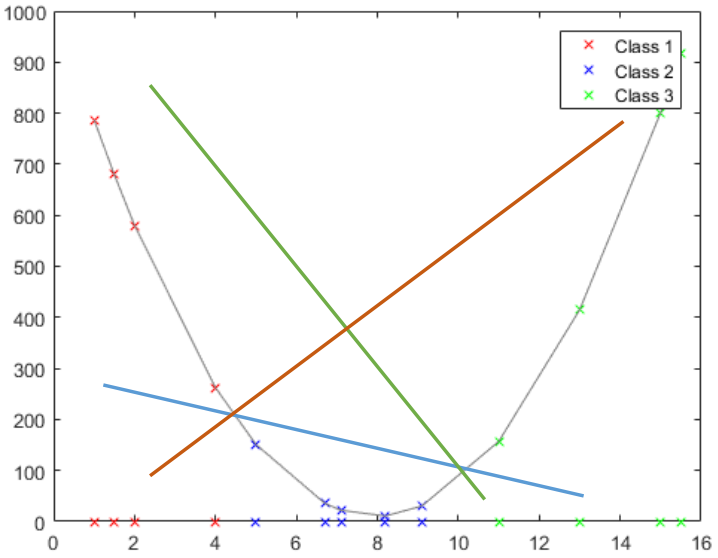


FIGURE 5.6: New separations on the quadratic curve.

5.1.4.2 Additional tuning parameters

One way of examining kernels is to treat them as a mapping. For a linear SVM we map our feature vectors into their complete set of dot products. The separating hyperplane is then calculated using a weighted sum of these. For a Radial Basis Function (RBF) kernel, we map our feature vectors in the following manner

$$K(u, v) = e^{\frac{||u-v||^2}{\sigma}} \tag{5.17}$$

This mapping, and many others, employs the additional tuning parameter σ. This allows us to scale the output of the kernel.

5.1.4.3 List of kernel functions used

The Kernels we used in the experiments were chosen based on two criterion. Firstly we chose the Linear, Polynomial and Radial Basis Function (RBF) kernel, as they are widely regarded as canonical kernels and any evaluation of the SVM would be incomplete without them. [redacted]

[redacted] we chose the L1, L1 RBF and Histogram Intersection RBF. [redacted]

Linear

$$K(u, v) = u \cdot v \tag{5.18}$$

Polynomial [111].

$$K(u, v) = (1 - u \cdot v)^2 \tag{5.19}$$

Radial Basis Function [112].

$$K(u, v) = e^{\frac{||u-v||^2}{\sigma}} \tag{5.20}$$

L1

$$K(u, v) = 1 - \sum_{i=1}^{n}(|u_i - v_i|) \tag{5.21}$$

Histogram Intersection Radial Basis Function

$$K(u, v) = e^{\frac{\sum_{i=1}^{n}(min(u_i, v_i))}{\sigma}} \tag{5.22}$$

Laplacian Radial Basis Function [113].

$$K(u, v) = e^{\frac{\sum_{i=1}^{n}(|u_i - v_i|)}{\sigma}} \tag{5.23}$$

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted] [redacted]

[redacted]

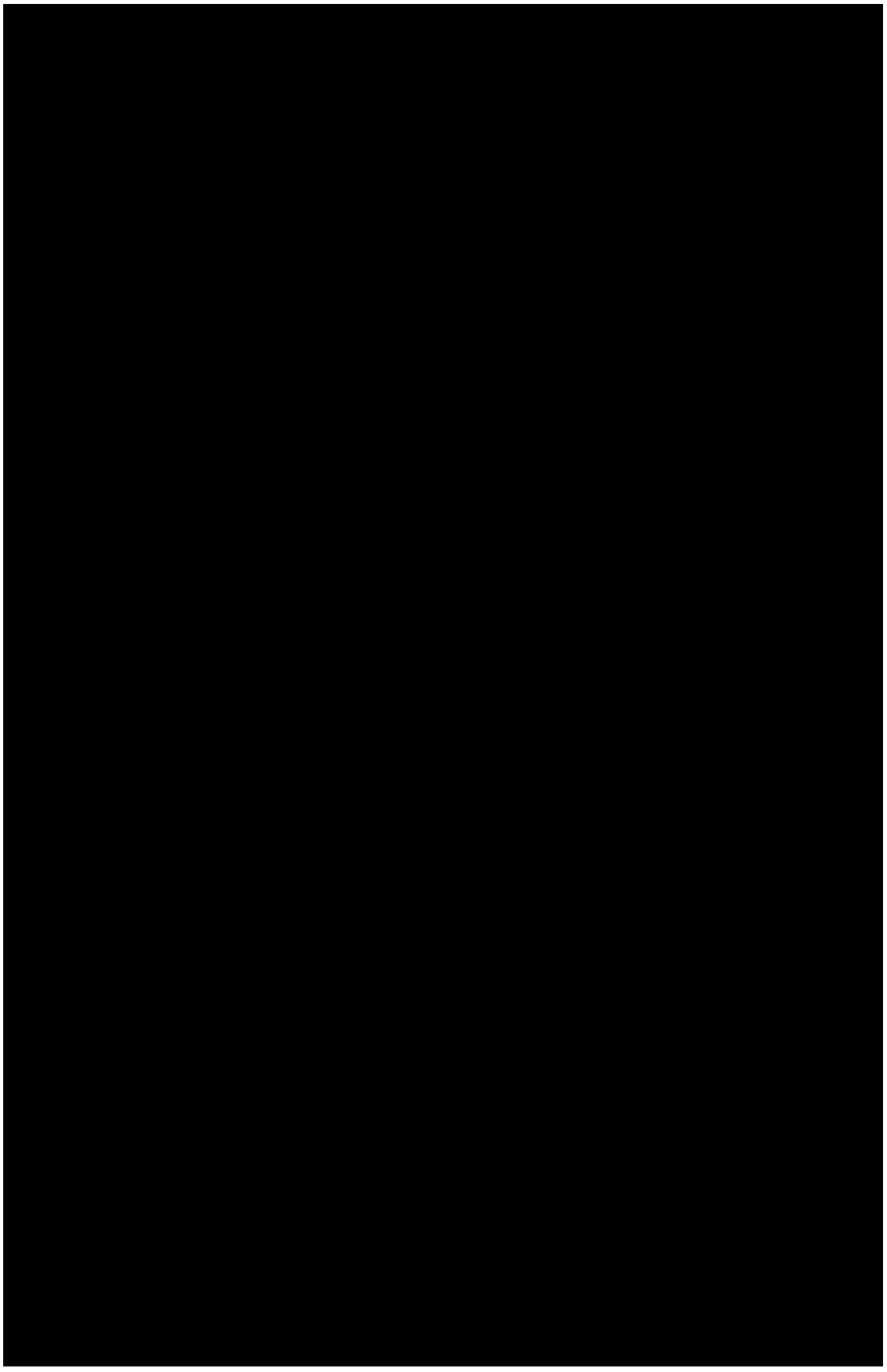
[redacted]

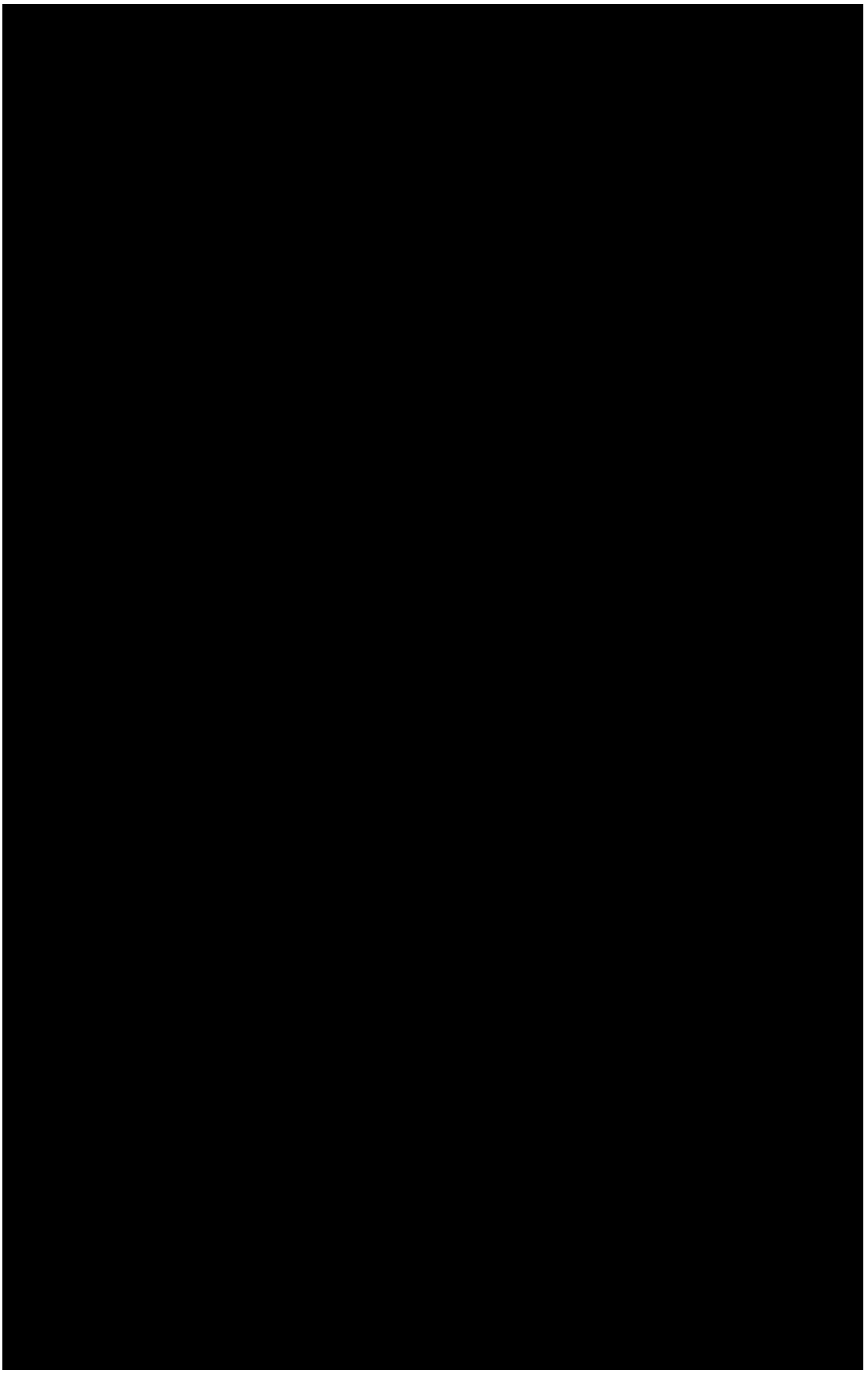
[redacted]

[redacted]

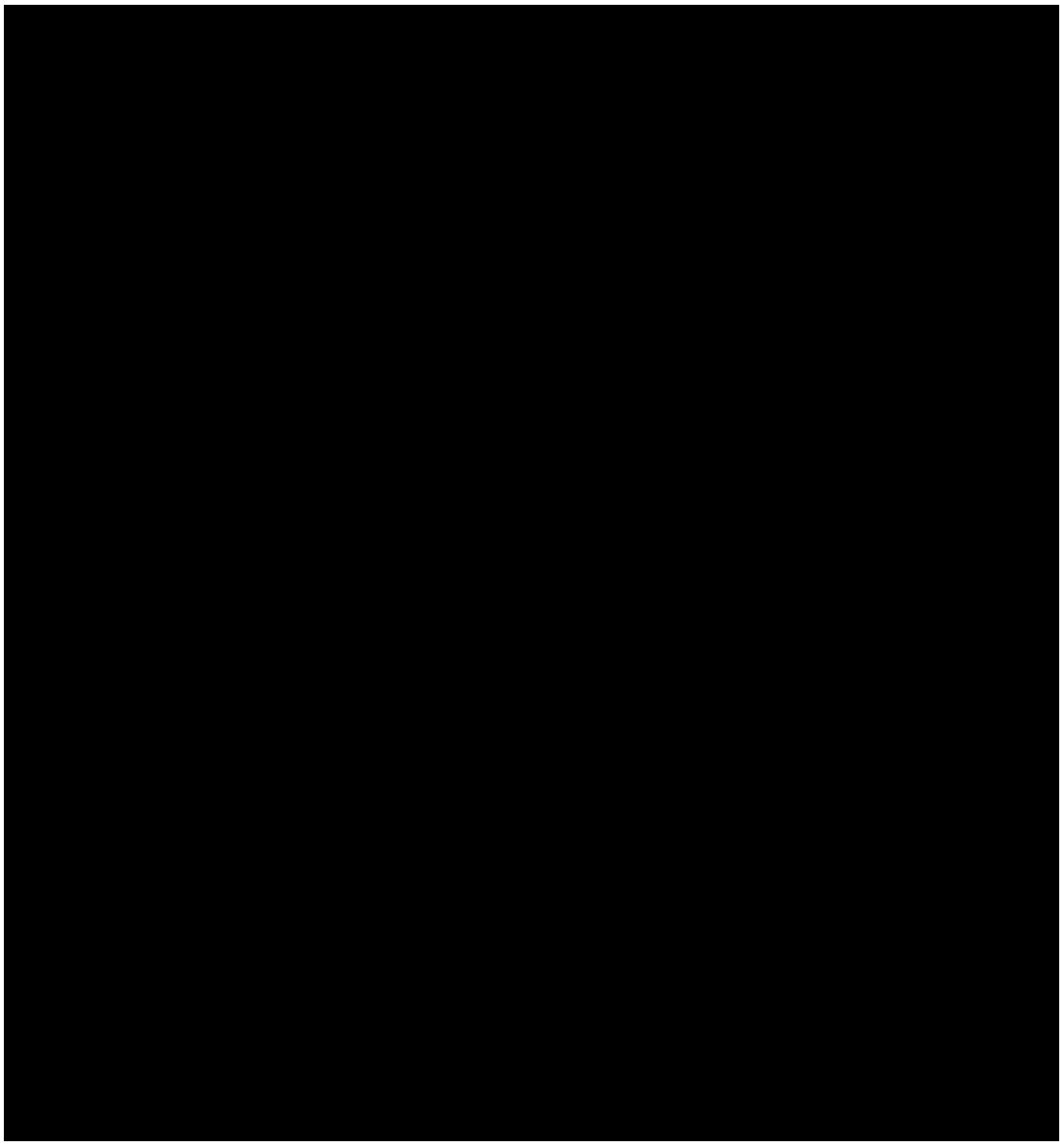
[redacted]

[redacted]





[redacted]



[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

Chapter 6

Sudoku texture classification

The Local Binary Pattern (LBP) is a benchmark method described in Section 2.5. In this Chapter we propose a new method named the Sudoku transform. Our new method encodes a local neighbourhood using a total rank-ordering, this forms a more robust description of a neighbourhood while retaining the benefits of LBP. When combined with a equailised grouping scheme to form histograms we show stronger performance than LBP on a number of benchmark databases.

Parts of this work were published in conference publication [E]: "Sudoku Texture Classification". This Chapter forms contribution [3]: "Developing a novel texture histogram for image analysis in the Local Binary Pattern framework"

An LBP is a binary string which describes a neighbourhood of pixels. It is formed by comparing a central pixel with its neighbours, if the neighbour is greater than the centre it is assigned 1, if less a 0. It was observed in Section 2.5 that LBP only crudely encodes magnitude information. In Section 2.6.4 Completed LBP was described which additionally encodes magnitude. They found that this magnitude supported better performance. We contend that **relative** magnitude is important. Our new representation describes how a local area is organised in terms of magnitude while retaining the primary benefits of LBP.

The rest of this Chapter is organised as follows. In Section 6.1 we propose the method for forming our patterns using a complete local neighbourhood ranking and the subsequent feature vector, henceforth termed the Sudoku transform,. We then go on to discuss our results.

6.1 Method

A Sudoku grid is a form of puzzle, popularised in the Japanese newspaper *The Monthly Nikolist*[114] and is found in most newspapers in the UK. A picture of a Sudoku puzzle is shown in Figure 6.1.

5	3	2	9	8	6	7	4	1
4	8	7	2	1	5	3	6	9
6	9	1	4	3	7	5	8	2
3	2	5	1	7	4	8	9	6
7	6	4	3	9	8	1	2	5
8	1	9	5	6	2	4	3	7
1	5	6	8	2	3	9	7	4
9	7	8	6	4	1	2	5	3
2	4	3	7	5	9	6	1	8

FIGURE 6.1: A completed Sudoku puzzle. the numbers in black are placed by the puzzle designer and the numbers in red must be deduced

Figure 6.1 shows a completed puzzle. The primary characteristic of a Sudoku puzzle is that the numbers 1 to 9 must appear only once in each column, row and the nine 3×3 cells. The specific rule regarding 3×3 cells formed the inspiration for the method we shall now describe.

We form a pattern by comparing all pixels in a neighbourhood simultaneously, forming a rank-ordering, shown in Figure 6.2.

We read this clockwise as a string starting at the right-middle pixel and appending the central pixel's rank to the end of the string. This can be expressed as

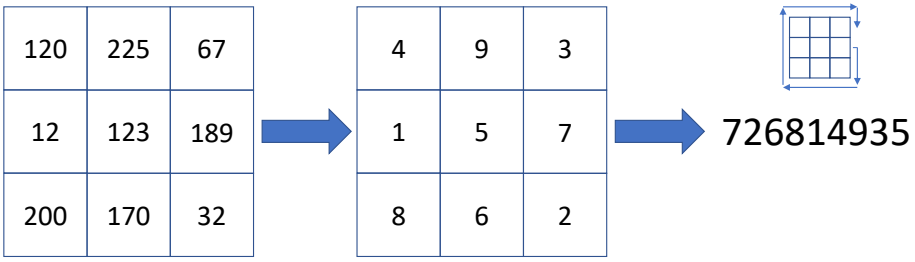


FIGURE 6.2: Transformation of pixel intensities from a 3×3 image patch to rank values.

$$S(P) = \sum_{i=1}^{n} RANK(p_i) * 10^{i-1} \tag{6.1}$$

where the *RANK* operator assigns the rank to pixel p in neighbourhood P. In analogy to LBP this number is then "bit" shifted with circular wrap around to its minimum value to achieve rotational invariance. The pattern derived from Figure 6.2 would be 726814935 and then shifted to 149357268. This process is applied to every pixel in an image to form our "Sudoku" image.

A key feature of LBP and also our Sudoku representation is that it is invariant to all typical photometric distortions that can occur when the capture conditions change (e.g. scaling, offsets or any non-linear increasing functions applied to the image). The Sudoku rank also has the advantage that it is a full rank order and is not based on the binary relation of a central pixel and its neighbours. As all our Sudoku patterns begin with a 1 there are only 8 numbers in each pattern which can change. As each pattern contains each number from 1 to 9 only once this means the number of rotationally invariant Sudoku patterns is $8! = 40320$. We plausibly have more information because the number of rotationally invariant LBPs is 36. Our hypothesis is that the Sudoku feature which compares all pixels with each other will capture yet more salient information and so support more accurate texture recognition. This representation also retains they key benefit of simplicity.

Currently the method is only defined for 3×3 neighbourhood. A generalisation of this method could lead to an $n\times n$ neighbourhood where n is the number of pixels in the neighbourhood. This would require a base n^2 numeric system. This would lead to the ability, like LBP, to define Sudoku patterns with variable radii and sampling size.

6.1.1 Uniformity

LBP uniformity is described earlier in Section 2.5.3. The analyses in [5] of uniformity in LBP patterns is directly applicable to our method as we are still using relational encoding and we know how our ranks vary around our central pixel. In our Sudoku method we apply LBP *uniformity 2* rule to our Sudoku features. We do this by only considering patterns which vary above or below the central pixels rank twice or less. All other patterns are placed in a bin appended to the end of the histogram. Examples of a non-uniform and a uniform Sudoku pattern are shown in Figure 6.3.

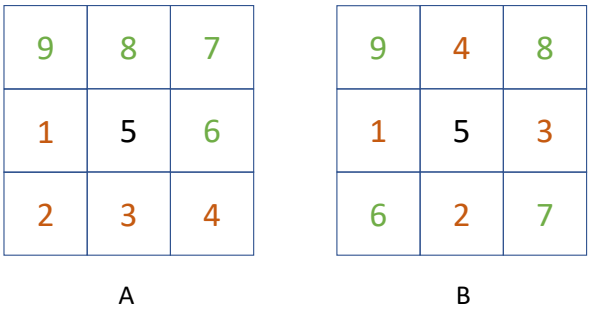


FIGURE 6.3: Two Sudoku patterns. Left is uniform and right is non-uniform.

In Figure 6.3 pattern A is uniform and pattern B is not. The numbers in Green are greater than the central rank and the numbers in orange are less. The neighbouring pixels in B oscillate between greater and less than as you read around the pattern resulting in a uniformity value of 8 - that is 8 changes above or below the central pixels rank. This pattern would be grouped in the final bin of a histogram.

Neighbourhood A has only two transitions resulting in a uniformity of two. This pattern would be accepted.

In LBP uniformity significantly compresses the histogram which can be considered an advantage. Unfortunately this is not the case for Sudoku. As the central rank of a neighbourhood can be 1 or 9 all possible Sudoku patterns are also potentially uniform. With regards to this it may be that there is a deeper meaning to *Uniformity* with respect to this representation. However this is not within the scope of this thesis.

6.1.2 Equality

In our representation we also consider equality. Currently if two pixels are equivalent then one is arbitrarily assigned a higher rank based on its position in the neighbourhood. We address this by allowing two pixels to have the same rank. See Figure 6.4.

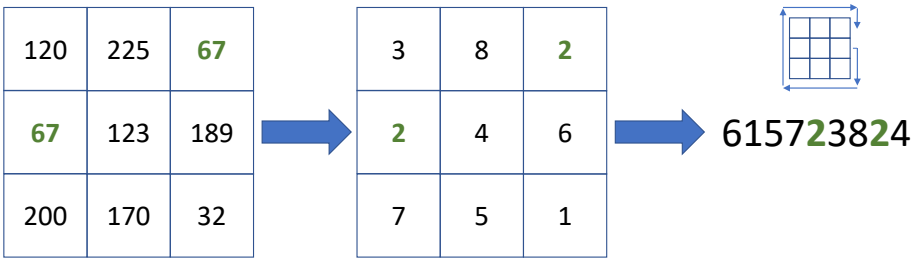


FIGURE 6.4: Transformation of a neighbourhood into its Sudoku glyph with equality.

It is worth considering the increased complexity by adding this step. Without equality there are $9! = 362880$ possible glyphs or $8! = 40320$ with rotation. The combinatorics become significantly more complex once we add equality. For example a 9 pixel pattern with two repetitions, such as the in Figure 6.4, would be of the form 157238246. Now if we consider all possible patterns with 2 repetitions we can calculate this using the combinatorics as follows.

$$7! * \binom{9}{2} \tag{6.2}$$

There are 7! different ways in which the unique values can be organised multiplied by the number of ways it is possible to choose 2 numbers from 9. We must consider all possible ways in which patterns can have repetitions, calculate the number of possible patterns for each permutation and then take the sum all of the above. We present this value as 871030 without rotation.

6.1.3 Histogram formation

We choose to begin the Sudoku pattern with *1*. This means all our initial features are large integers. In order to remove any bias in the representation (from how we make the nine digit number) we map the calculated Sudoku integer non-linearly to the interval [0,1]. To accomplish this we calculate an increasing function $f()$ with a resulting n values. These n values when used as quantisation levels on the Sudoku integers of a whole dataset effect that the histogram of that dataset is uniform. The histogram of quantised, according to the previously calculated function, Sudoku ranks for a given image will not be uniform and is used as the index for texture recognition. A visualisation is shown in Figure 6.5.

6.1.4 Relation to the Complete Rank Transform

Before we discuss our experiments we must discuss the Complete Rank Transform (CRT) [101], a method with obvious similarities to our Sudoku Transform.

The CRT is an extension a previous method known as the Rank Transform of [100]. The authors present two features in their work known the Rank Transform and the Census transform (for a full description see Section 2.10.1). The Census transform

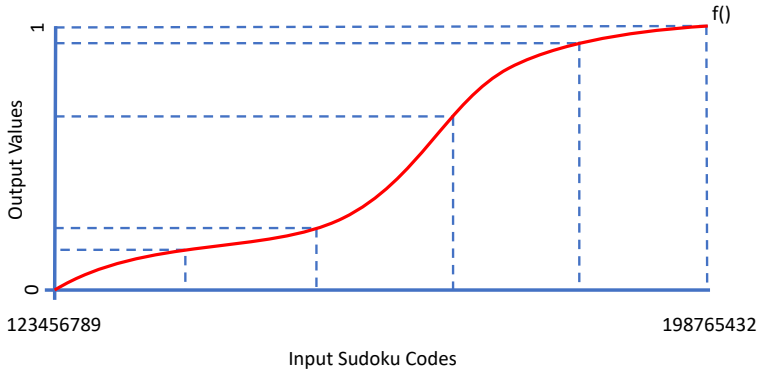


FIGURE 6.5: Visualisation of a histogram equalisation function with 5 levels.

bares a strong visual resemblance to the LBP however it is widely accepted that the two methods are distinct [115][116][117]. The reasons are as follows:

- The Census transform concatenates the bits in an arbitrary manner where LBP uses a specific choice to obtain rotational invariance. Rotational invariance is a property not exhibited by the Census transform.
- There is no "Uniformity" equivalent step in the Census transform, all bit strings are considered equal in terms of descriptive ability.
- The Census transform does not form a histogram like LBP. Instead strings are compared individually by the Hamming distance [19].

For the same reasons, and more, that the Census transform is distinct from LBP, the Sudoku transform is distinct from the CRT. Specifically:

- As with the Census transform the CRT concatenates the bits in an arbitrary manner where our Sudoku method uses a specific choice.
- Our Sudoku patterns are mapped through a histogram equalisation function to remove bias where the CRT ranks are used as signatures per pixel.

- The CRT does not produce a global index for an image (like the Sudoku histograms), rather the individual signatures are input into an Optical Flow algorithm to perform matching.

One final difference is that the Census and Complete Rank Transforms would require manifestly different implementations to be effective in texture classification. Just as the Sudoku transform and LBP could not work, without serious modification, in an Optical Flow application. As a result we shall not be comparing the methods further.

6.2 Classification

For our classification experiments we will compare our Sudoku method, with and without equality, with LBP and Local Ternary Patterns: a method which encodes greater than, less than and *close* to the central pixel. This method is fully described in Section 2.6.2.

We classify our histograms using a K Nearest Neighbour (KNN) classifier [20] using the χ^2 distance as our comparison metric:

$$\chi^2 = \sum_{i=1}^{b} \frac{(x_i - y_i)^2}{x_i + y_i} \tag{6.3}$$

which measures the dissimilarity between two histograms x and y with number of bins b.

We calculate the distance between a test sample and every training sample. We rank the training samples in increasing order of their distance to the test sample and take the first K results (the K most similar images). The test sample is assigned the modal class within those K results. In our experiments we arbitrarily set $K = 3$ as we are interested in comparative performance. In the case of a 3-way

tie we take the first (closest) images class as our result, this is equivalent to using K = 1. $K = [5, 7, 9]$ were tested and little difference was found between them.

To perform our experiments we use a two-fold classification protocol. With the images in each class sorted alphabetically by file name; firstly we use the odd indexed images for training and even indexed for testing. We then swap the training and the testing data for the 2nd pass. The final result, detailed in Table 6.1, is the mean percentage accuracy of the two passes. A full workflow depicting our entire experimental process is shown in Figure 6.6.

6.2.1 Data

We perform our experiments on members of the Curet [118], Outex_TC_00013 (hereafter named as Outex) [119] and Vistex Datasets [120], examples of which can be seen in Figure 6.7.

For all images which are colour we first convert them to grey-scale using the function $0.2989 * R + 0.5870 * G + 0.1140 * B$ where R, G and B denote the Red Green and Blue channels respectively.

6.2.2 Sensitivity testing

The number of bins in an LBP_{riu2} histogram is absolute, there can be no more or no less as each pattern indexes to a bin. In Sudoku histograms we group patterns into bins according to a histogram equalisation function. This allows us a vary the quantisation level on a per experiment basis. This can be considered a sensitivity of the representation and as such must be evaluated.

The results in Section 6.4 are generated by classifying the images with a variable number of bins and selecting a representative result. Specifically the function we use is $round(2^i)$ where $i = 2 : 0.1 : 10$. This calculates 73 unique bin values on an

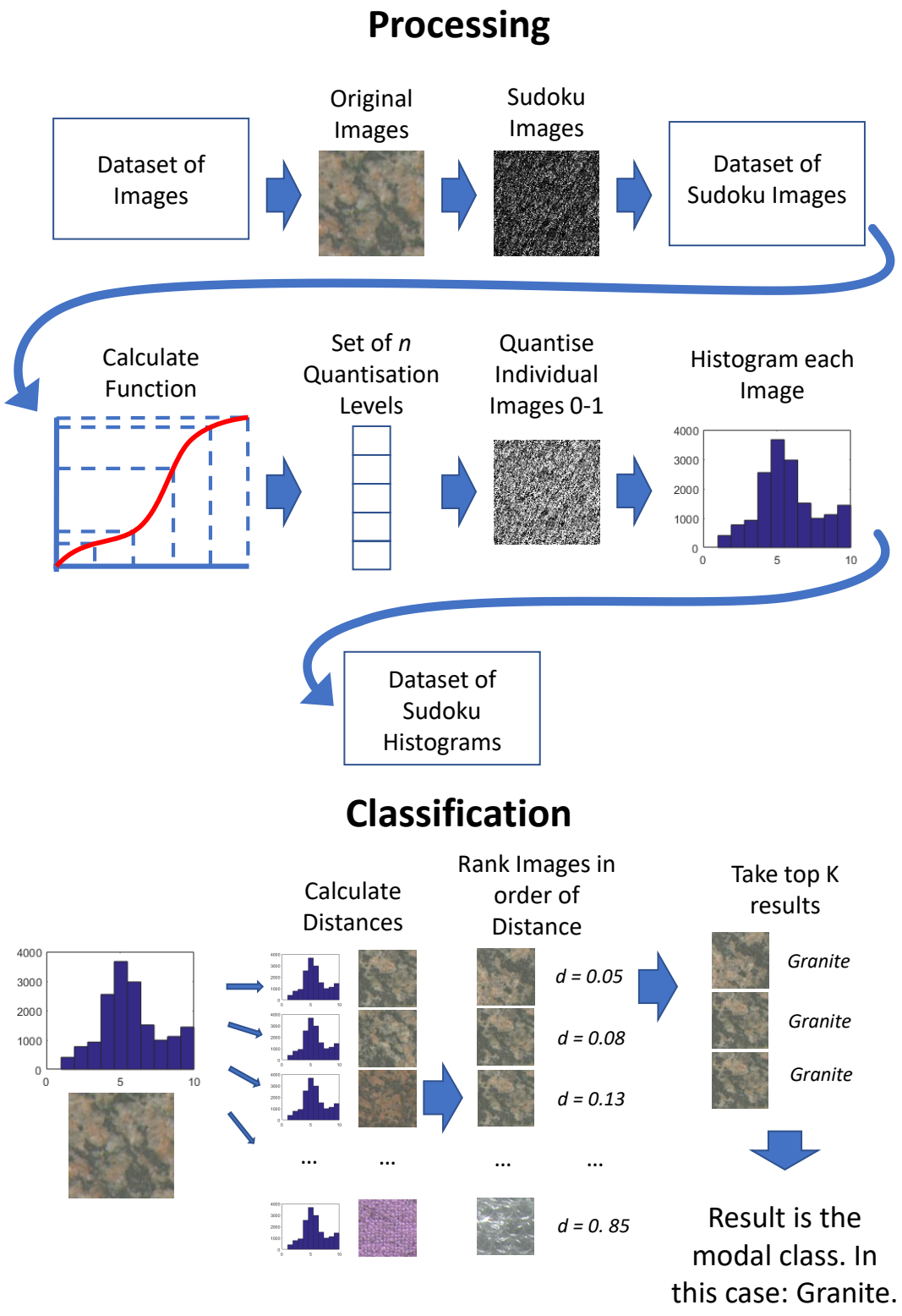


FIGURE 6.6: The full workflow of our experimental process.

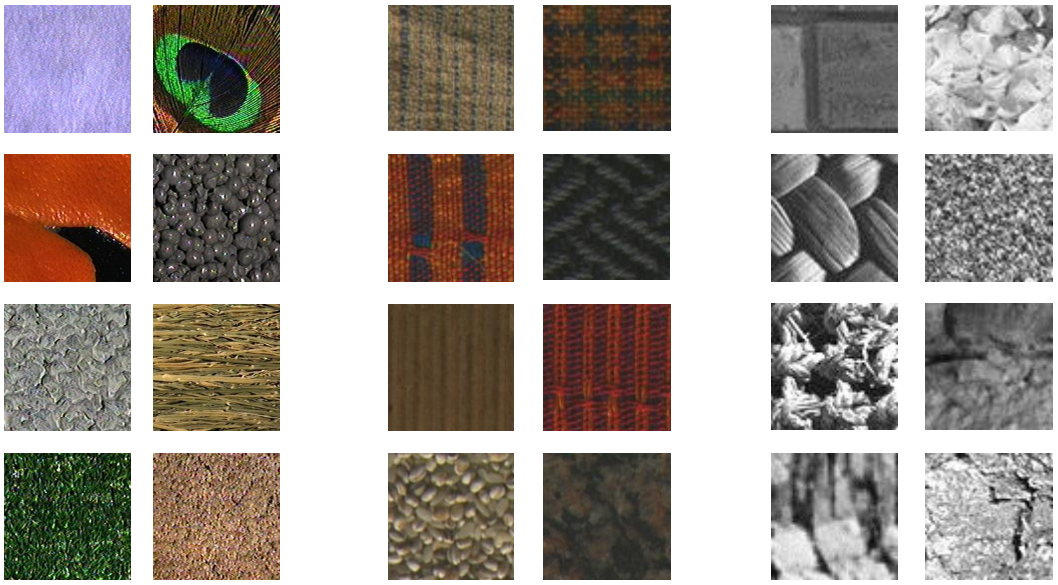


FIGURE 6.7: Examples of the threes datasets, from left to right: Curet, Outex, Vistex.

exponential scale between 2 and 1024. See Figure 6.8 for the full range of results on the Outex and Vistex datasets.

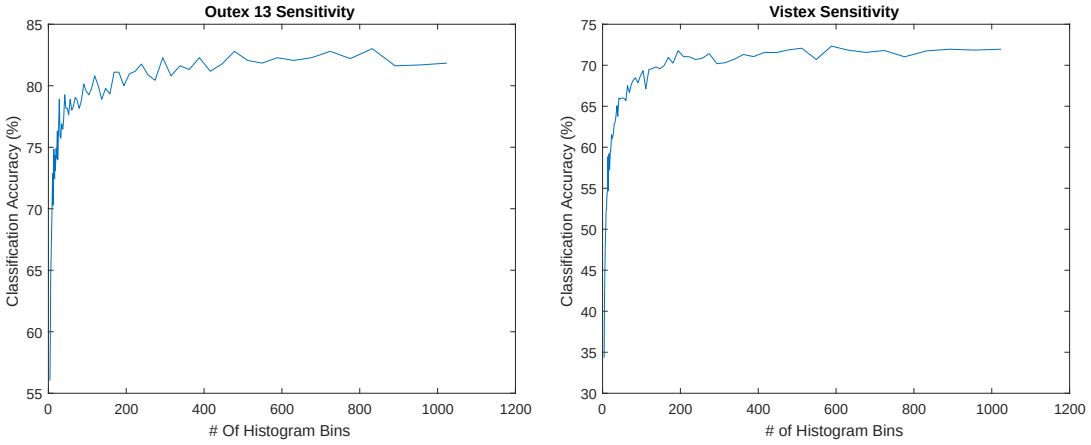


FIGURE 6.8: Bin density vs classification accuracy on Outex and Vistex.

We find that the sensitivity of the method varies on a pseudo-logarithmic scale which flattens at approximately 250 bins. This suggests that there is a certain amount of granularity required for a Sudoku histogram to accurately capture an images patterns, however once this is reached adding more bins only adds sparseness to the representation.

6.3 Experimental parameters

All methods use an 8 point 1 radius pattern. The per method parametrisation is as follows

- Sudoku - We empirically choose a quantisation level based on best performance as described in Section 6.2.2.. This results on histograms with three different lengths (one per dataset).
- Sudoku with Equality (SudokuE) - Parametrisation remains the same as with the above Sudoku patterns however we now incorporate equality into the local codes (See Section 6.1.2)
- LBP - We use a rotationally invariant uniformity 2 pattern; resulting in histograms with 10 bins.
- LTP - We use the same parameters as LBP. The additional threshold is $\pm$ 5. The ternary patterns are calculated in their positive and negative parts and concatenated to form a 20 bin histogram.

6.4 Results

Table 6.1 shows our results. We compare our Sudoku method with traditional LBP and LTP.

	Curet	Outex	Vistex
Sudoku	89.1	82.3	71.8
SudokuE	**91.3**	**84.8**	77.3
LBP	78.6	67.8	70.2
LTP	84.4	72.3	**80.3**

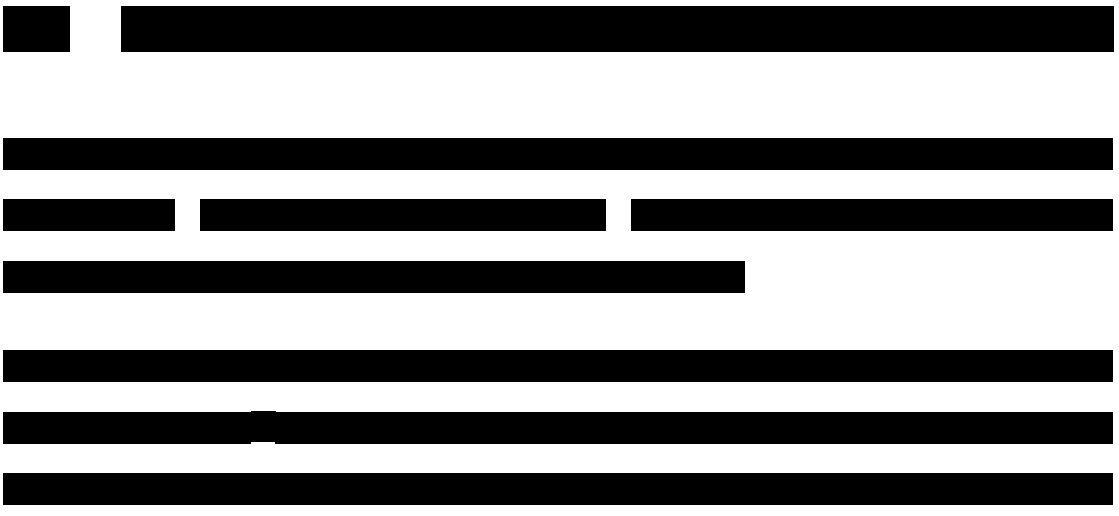
TABLE 6.1: Table of mean percentage accuracies for the 4 methods described over the 3 databases.

	Curet	Outex	Vistex
Sudoku	362	294	194
SudokuE	676	256	388

TABLE 6.2: Table of bin quantities used for the two Sudoku methods over the 3 databases.

We observe a performance increase over both traditional LBP and LTP when using either form of Sudoku description. SudokuE also produces a mild but consistent performance increase over Sudoku. On Vistex our method performs worse than LTP, but not LBP.

These performance increments do come at a cost of significantly increased feature vector length. This increase varies from 19.4 to 67.6 times that of LBP. While a concern this is not a problem with regards to this Chapter: as the primary thesis of the method is that a rank-ordering obtains more salient information from a neighbourhood than simple comparative measures. An LBP histogram has a finite length. There are only 256 possible patterns in a 3x3 neighbourhood and it has been shown that 9 of these are sufficiently discriminative [5]. Our representation increases the volume of potential patterns and a larger histogram, so as to represent the encoded information more completely, is expected.



6.6 Conclusions

We have presented a new feature based on LBP. Our feature considers the relationships between all pixels in a local neighbourhood and forms a histogram based on the distribution of said features. Over the three datasets our experiments show a mean increase in classification accuracy of **12.3%** over traditional LBP. Relative to the more complex LTP we show a mean performance increase of **4.8%**, noting that we observe a **3%** deficit on the Vistex dataset.

We now move to examine invertibility of the description. We present a new method for recovering images from their Sudoku and LBP patterns.

Chapter 7

Image Recovery From Texture

Methods for image texture analysis usually take an image and convert it into some textural representation. In this Chapter we present a new method for inverting this: taking a textural representation of an image and recovering the original. We first extend a method from the prior art and then present our new method. Our new method offers a significantly more accurate recovery.

Parts of this work were published in conference publication [F]: "Recovering a Colour Image from its Texture". This Chapter forms contribution [4]: "Proposing a new reconstruction approach to show the completeness of the image descriptors".

7.1 Introduction

Texture features can be considered as methods for encoding an image: taking pixel intensities or filter responses and forming them into a description which can be used to solve problems including recognition and matching. In this Chapter we are considering the inverse problem: given a textural representation of an image, how well can we recover the original.

The usual goal of a texture representation is to enable recognition however the structural aspect of texture is also important. We might ask, for example, how easy it is to synthesize a texture from an exemplar. An important work here is that of Efros et al. where given a seed image or image patch they "grow" a texture which is visually similar [90]. Particularly relevant to this chapter we might ask how much of the original image can we recover from the texture representation.

This last question is interesting. Image processing is full of examples of dual representations. For example it is well known that an image and its Fourier transform are bijectively related. And, of course, some image processing tasks are better performed in one representation than the other (e.g. fast convolution is carried out in Fourier space) [121]. We propose that a measure of texture "as a representation" is the extent to which it "encodes" the original image.

Image reconstruction from texture has many applications. An example is Cryptography: plausibly an image's texture features could be transmitted and then decoded elsewhere off-line to recover the image. Another example would be analysis of a texture representation. The reconstruction would form a visualization of the knowledge encoded therein.

Local Binary Patterns (LBP) are one of the most successful and commonly used texture representations. In their simplest form an LBP feature is a binary string which encodes whether the brightness at an eccentric pixel is more or less than (1 or 0) than the centre. These 1s and 0s are read out to form a binary number which can be read as an integer. Typically these numbers are grouped (e.g. histogrammed) to form a feature vector for indexing. The key strengths of LBP are its invariance to monotonic changes in image brightness (by scaling, gamma functions or tone mapping) and its robustness to rotation [5, 73].

There are many extensions to LBP. One such is the Sudoku representation. Rather than encoding relationships to a central pixel pixels are ranked in a 3x3 neighborhood (using numbers 1 through 9). By construction the Sudoku grid has the same

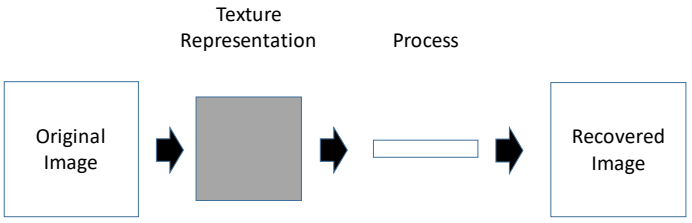


FIGURE 7.1: The image recovery pipeline.

monotonic invariance as LBP but is a richer feature set [122].

Both LBP and Sudoku deliberately remove greater or lesser amounts of magnitude information. This can be considered as removing unimportant intensity representation in favor of a more structural description. The question we ask is: given the per-channel textural information for an image; how well can we recover the color image? We present results for two methods: one previous art known as the Minimum Contrast (MC) algorithm [123] and our own proposed method called Quadratic Reconstruction (QR). We show that essentially the LBP or Sudoku texture encoding (at a pixel) specifies the intensity relationship that pixels in a proximal region need to satisfy. Over the whole image the effect of these local relations propagate. We demonstrate how - using an optimization technique called Quadratic Programming - we can recover the minimum norm image that satisfies the constraints. Compared to the prior art MC method, our new QR method provides a much better recovery. Further, we show Sudoku QR is much improved compared with is LBP variant.

The rest of this chapter is organized as follows: First we shall detail the MC and QR algorithms. Secondly experiments are presented.

7.2 Image recovery

While the methods detailed above go as far as to form feature vectors, we would like to take a step back and consider just the pattern in 3x3 neighbourhoods. That is we use the set of comparisons performed by the method while retaining the pixel locations. This is important as once the patterns are formed into strings and/or rotated the positioning of each pixel becomes near-arbitrary making recovery a significantly more complex task. The following two methods, Minimum Contrast and Quadratic Reconstruction, use these as the basis for their reconstructions. The recovery pipeline is shown in Figure 7.1.

7.2.1 The Minimum Contrast Algorithm

The minimum contrast algorithm was proposed in [123] as a way of inverting the Local Binary Pattern. The primary goal was two-fold, firstly to provide a recognizable reconstruction of an input image, and secondly to show what information was lost in the LBP conversion.

To reconstruct an image using this method first consider the 4×3 patch in Figure 7.2

1	33	85	197
242	69	124	12
145	21	83	211

FIGURE 7.2: A patch of pixels.

In this 4×3 patch there are two 3×3 neighbourhoods centred around rows, columns (2,2) and (2,3), seen in Figure 7.3.

1	33	85
242	69	124
145	21	83

33	85	197
69	124	12
21	83	211

FIGURE 7.3: The two neighbourhoods of 7.2.

We can then transform these neighbourhoods into their LBP counterparts as in Figure 7.4.

0	0	1
1	$C_{2,2}$	1
1	0	1

A

0	0	1
0	$C_{2,3}$	0
0	0	1

B

FIGURE 7.4: LBP transformations of Figure 7.3.

Considering only the LBP transformations: if we first examine A we can see that the top left pixel is less than the centre, as defined by the 0. This means that the central pixel of A must be at least 1 pixel intensity greater than the top left. We can also see that the bottom right pixel is greater than the centre, as defined by the 1. This means that the bottom right of A is at least 1 pixel intensity greater than the centre; and transitively at least 2 pixel intensities greater than the top left. If we then move to examine B we can see that the central pixel of B is greater

than the bottom central pixel, which corresponds to the bottom right pixel of A, and that the bottom right pixel of B is greater than the centre. From this we can deduce that the bottom right of B is at least 5 pixel intensities greater than the top left of A, or in other words the **minimum contrast** between the two pixels is 5. Using this series of greater than relationships we can express this as a path, see Figure 7.5.

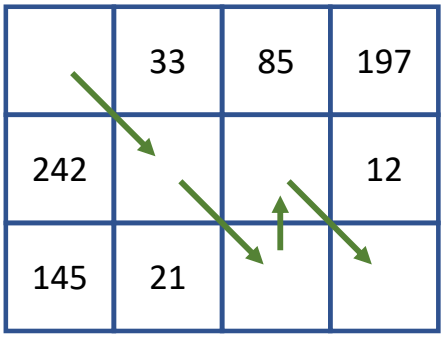


FIGURE 7.5: One greater than path through Figure 7.2.

It is worth noting that this is not the longest path available which reaches the bottom right. However this is the longest *explicit* path defined by A and B. The actual longest path would become explicit if the neighbourhood around row, column (1,1) were part of this example.

To expand this process we must start at every local minimum in an image and recursively generate every possible greater than path through said image. In each pixel location we store the length of the longest path to that pixel. The path length is defined as the pixel brightness in the reconstructed texture image. The final result is an image in which the local binary patterns match exactly to the original, More details of the implementation of this method can be found in [123].

We extend this process to use the Sudoku feature. In a Sudoku pattern we have the rank ordering of a neighbourhood expressed as the numbers 1 to 9. This means that if one pixel is ranked 9 and another pixel is ranked 2 the minimum intensity

difference between those two pixels is 7. To incorporate this into the minimum contrast algorithm when we explore a greater than relationship we increment the path length by the absolute rank difference between the two pixels. For an example of both LBP and Sudoku minimum contrast reconstruction see Figure 7.6.

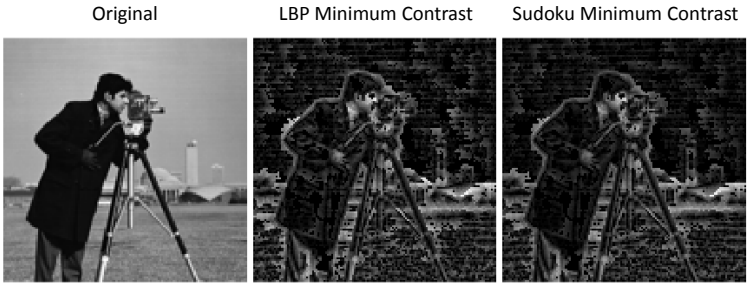


FIGURE 7.6: Examples of LBP and Sudoku minimum contrast reconstruction on MATLABs "Cameraman.tif".

7.2.2 Quadratic Reconstruction

Our proposed method forms the problem in terms of quadratic programming. If we consider an image to be a vector $\vec{x}$ with the pixels as variables $x_1...x_n$. For LBP we can formulate a set of linear constraints such that each pixel x_i is constrained to be at least 1 greater or lesser than its neighbours (while enforcing positivity).

The difference of 1 is drawing attention to the fact that we have an ordinal relationship e.g. that the central pixel is larger than one neighbour. For a neighbourhood $x_1...x_n$ with central pixel x_c the weakest way we can interpret this circumstance (assuming an image is encoded using integers) is that $x_p - x_c \leq 1$ where x_p is the larger pixel.

For a single 3x3 region and its LBP coding we have 8 of these kinds of relations. And, of course we know all pixels $x_1...x_n \geq 0$. See Figure 7.7 as an example of how we turn a pattern in to a set of inequality relations.

Now we must consider how to recover an image given these inequalities. Our key insight is to employ a method called Quadratic Programming (QP) [109]. Per

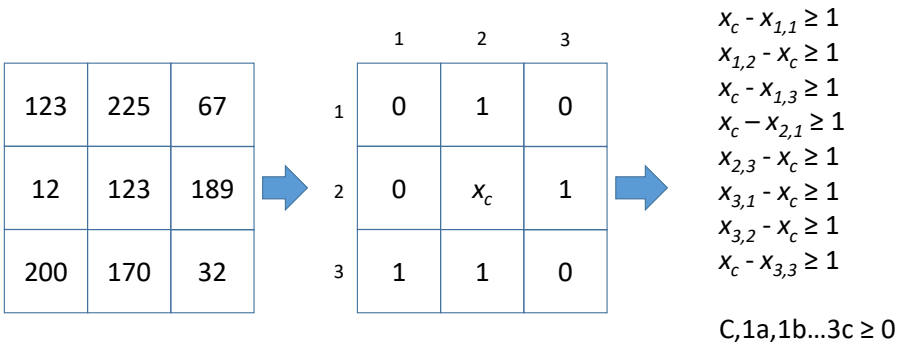


FIGURE 7.7: Transformation of an LBP into its associated constraints.

pixel we have 8 inequality relations where each pixel has a value between 0 and 255 inclusive. Let x_i denote the ith pixel in an image $\vec{x}$ (understanding that the pixel is in a a 2D grid, but for our purposes it is useful to think of there being n pixels in an image and x_i is the ith one). Now for neighbouring pixels we have the linear inequalities as per Figure 7.7. But, thinking of the image as a vector of pixels these inequalities now refer to the ith and jth pixels where (i and j will be far apart).

We construct a large matrix A such that in each row we encode a single inequality relationship. So, in terms of our example the kth row at position i of A could have 1 and at position j a 0 if $x_i > x_j$. We have a corresponding vector B (which is a vector of 1s). Now we can write

$$A\vec{x} \geq \vec{B} \tag{7.1}$$

In the above equation we are explicitly writing in matrix form all the inequalities that arise from every LBP encoding. Now we minimize:

$$min||\vec{x}||^2 s.t. A\vec{x} \geq \vec{1} \tag{7.2}$$

We solve for the above using QP. QP is guaranteed to find the global optimum solution.

For an LBP transformation there are 8 comparisons between a central pixel and its neighbours. As such there are 8 constraints defined per pixel. Each pixel appears in 9 different neighbourhoods so the transitive relationships between neighbourhoods in an image will naturally be preserved.

Intuitively we can apply the same methodology to the Sudoku pattern. But, now there will be more constraints. Indeed in a 3×3 neighbourhood the total number of binary comparisons possible is $\binom{9}{2} = 36$. Forming these as above gives us an exact representation of the Sudoku encoding and solving with QP remains the same. Pixel equality, encoded by Sudoku, is incorporated by constraining equal pixels to have a negligible difference $\approx 10^{-}6$ instead of 1.

In Figure 7.8 we show examples of Quadratic Reconstruction (QR) using LBP and Sudoku information on MATLAB's "cameraman" image.

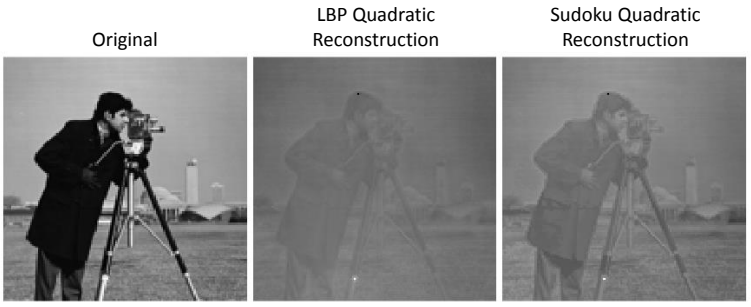


FIGURE 7.8: Examples of quadratic reconstruction on "Cameraman.tif".

7.2.2.1 Tone curve mapping

As a further processing stage in our QR pipeline we perform tone mapping on the calculated images using Isotonic Regression. This finds a least squares fit y to a vector x based on a known quantity x'. This is subject to the constraints $y_i \leq y_{i+1}$ [124]. This enforced monotonicity retains the illumination invariant properties of

LBP and Sudoku. By finding a least squares fit between the original image and the new image we bring the QR pixel intensities more in line with that of the original image. For examples see Figure 7.9.

FIGURE 7.9: Example of Isotonic Regression on a Sudoku Quadratic Reconstruction of MATLABs "Cameraman.tif".

Of course, in the texture representation we do not know the original image so cannot really carry out Isotonic Regression. But, plausibly a tone curve could be stored along with texture features which could be deployed if a reconstruction was necessary.

7.3 Experiments

For all of our experiments we employ the pipeline detailed in Figure 7.1. To compare our reconstructions with the original image we use the Structural Similarty index (SSIM). This is a perceptual measure based on image degradation [125].

We reconstruct images from four databases. Outex_00013 [119], Vistex [120], Curet [118] and Ponce [48]. We use a small, consistently chosen, selection of images from these datasets as follows:

- Outex_TC_00013: We use the first and second images from each class resulting in a database of 136 128x128 RGB images.

- Vistex: We use the first and second images from each class resulting in a database of 116 64x64 grey-scale images.
- Curet: Each class in Curet has an illumination gradient. The first numbered images are dark and the highest numbered images are bright. We choose the 47th and 48th images from each set (the middle two). Due to processing time constraints we resize the images to 0.5 of their original resulting in a database of 122 100x100 images.
- Ponce: We use te first and second images from each class resulting in a database of 50 images. Due to memory constraints we resize the images to 0.25 of their original. Resulting in 160x120 grey-scale images.

We reconstruct four datasets in six different ways:

- LBP Minimum Contrast (LMC)
- Sudoku Minimum Contrast (SMC)
- LBP Quadratic Reconstruction (LQR)
- Sudoku Quadratic Reconstruction (SQR)
- LBP Quadratic Reconstruction with Isotonic Regression (LQRI)
- Sudoku Quadratic Reconstruction with Isotonic Regression (SQRI)

For all images which are colour we first convert them to grey-scale using the function $0.2989 * R + 0.5870 * G + 0.1140 * B$ where R, G and B denote the Red Green and Blue channels respectively.

	LMC	SMC	LQR	LQRI	SQR	SQRI
Outex13	0.26	0.26	0.65	0.69	0.77	**0.96**
Vistex	0.39	0.51	0.33	0.70	0.55	**0.91**
Curet	0.44	0.45	0.35	0.64	0.61	**0.93**
Ponce	0.50	0.53	0.30	0.69	0.56	**0.89**

TABLE 7.1: SSIM results for the 6 methods over the 4 datasets.

7.3.1 Results

SQRI clearly performs best out of all of the six methods. The reconstructions are very close to their original images on all four datasets: especially Outex_TC_00013 where we have an average difference of 0.04 between the reconstructions and the originals. Figure 7.10 shows one example from each dataset, note how the SQRI reconstructions, especially on Curet and Outex, are perceptually almost identical to the originals.

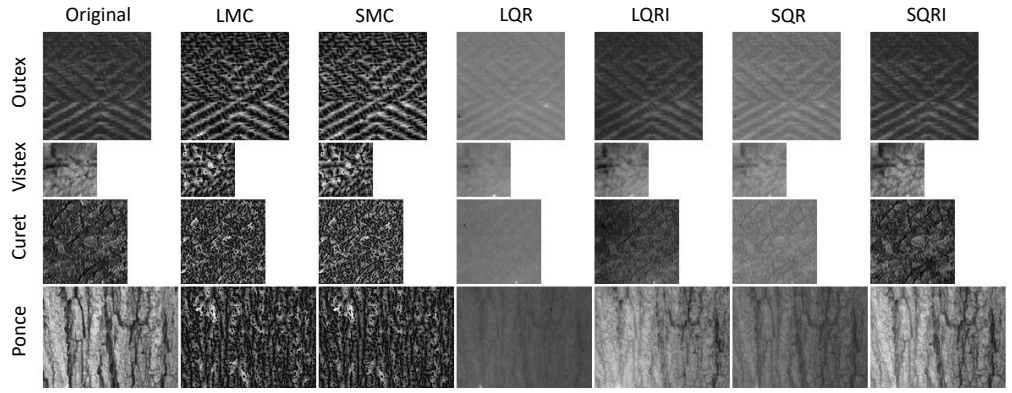


FIGURE 7.10: Examples of grey-scale reconstructions from Outex, Vistex, Curet and Ponce.

Sudoku provides a more significant performance boost in QR (≥ 0.12) than in MC (≤ 0.03). This is likely due to the shortcoming of Sudoku in MC detailed in Section 7.2.1, that is that the reconstructed codes only match closely, not exactly. This is due to how edges are computed: the maximum rank on an edge pixel can only be six, and on a corner can only be four. This leads to a loss of information across the entire boundary of the image which then propagates into all other

neighbourhoods. It might be that the MC reconstructions could be improved in implementation however this is not within the scope of this Thesis.

Using Isotonic Regression to tone map our images significantly increases performance across all four datasets. In some cases it appears to be a vital part of the process: LQR is outperformed by MC on Vistex, Curet and Ponce while SQR is only marginally better on Vistex and Ponce. LQRI and SQRI however provide a significant performance boost.

7.4 Reconstruction in colour

Our grey-scale results are very promising. An interesting question would be how well the methods translate to RGB images. The pipeline we use requires some modifications, these are shown in Figure 7.11.

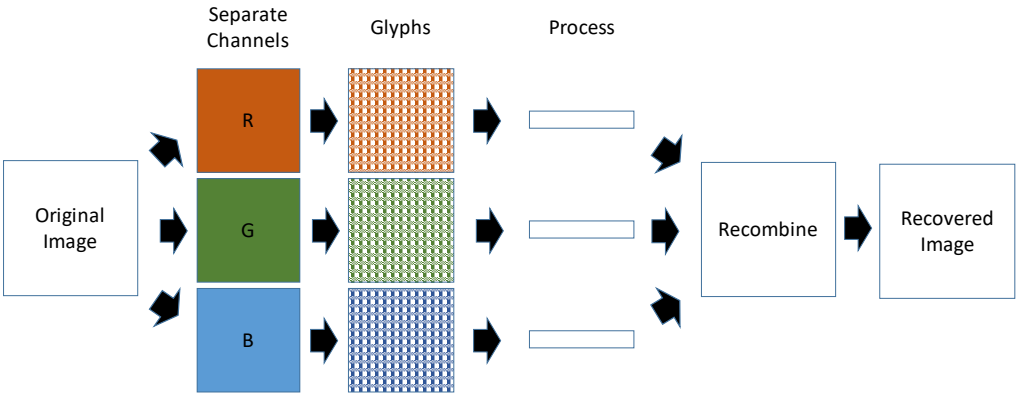


FIGURE 7.11: The color image recovery pipeline.

We choose 3 images from the MATLAB default package: kobi.png, football.jpg and onion.png. We also use 3 images from the Outex_TC_00013 texture dataset: 000087.bmp, 000366.bmp and 000397.bmp. Table 7.2 shows our results. Figure 7.12 shows the images corresponding to Table 1.

As expected SQRI is once again our strongest performing reconstruction. Sudoku features also provide stronger reconstruction in QR but not MC. This is likely not

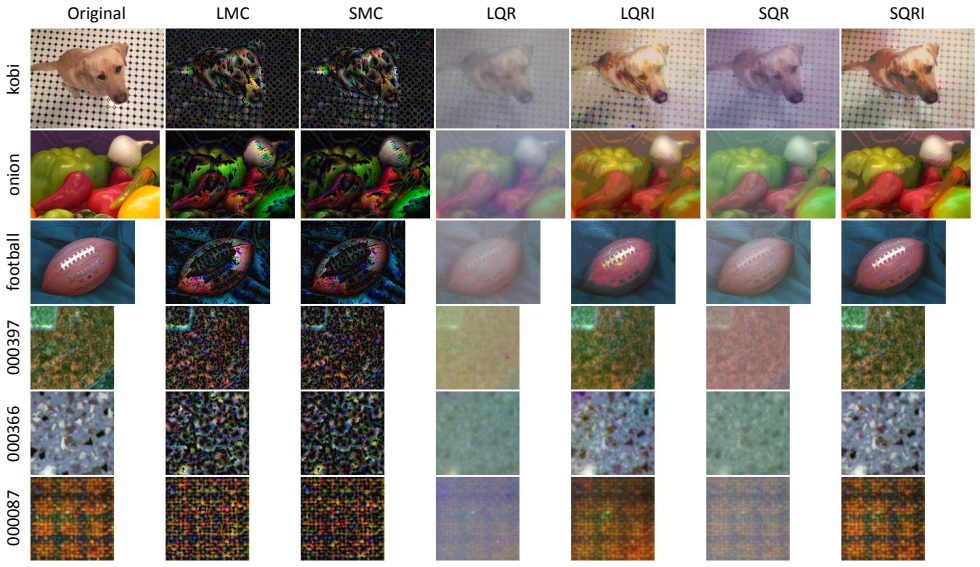


FIGURE 7.12: All resultant images from our experiments.

	LMC	SMC	LQR	LQRI	SQR	SQRI
kobi	0.20	0.20	0.29	0.53	0.46	**0.64**
football	0.30	0.28	0.53	0.59	0.75	**0.82**
onion	0.37	0.35	0.50	0.56	0.67	**0.71**
000397	0.32	0.30	0.64	0.89	0.80	**0.98**
000366	0.28	0.27	0.75	0.89	0.90	**0.96**
000087	0.40	0.40	0.49	0.70	0.63	**0.97**

TABLE 7.2: Table 1: SSIM results for the 6 methods over the 6 images.

significant as our earlier results on grey-scale show that over larger sample sizes SMC very slightly outperforms LMC on single channels.

QR performs significantly better on the texture images than on the regular images. This is due to the fact that the texture images selected contain fine detail, or high frequency, patterns. As we only sample neighbouring pixels it is expected that the high frequency information propagates more efficiently than low frequency. A possible solution for regular images would be to combine multiple different scales in the solution.

7.5 Evaluation

We have presented a novel use of quadratic programming applied to image recovery. We have shown that our QR method provides a statistically more similar image using a perceptual metric. Subjectively we also believe the SQRI images to be more visually similar.

LMC and SMC have provided poor results. It might be that their reconstructions could be improved in implementation as the earlier results on grey-scale suggest the translation to color should be routine but this does not seem to be the case.

QR does have drawbacks - it is currently very computationally expensive. There are methods of circumventing this such as sparser sampling of the input image and this will be explored in the future. Quadratic Programming itself is an expensive process and this is the primary source of slowdown.

Perceptual comparison is also interesting. Isotonic Regression provides a mathematically more similar image in all cases. However for some images, for example kobi and football, we believe that the reconstruction without isotonic regression is more perceptually similar. A more complete set of results would include perceptual testing, this will be considered in the future.

7.6 Conclusions

In conclusion we have shown that given an image's textural information it is possible to obtain a recognisable image. We have also shown that with our novel quadratic programming method computed in the RGB channels we can recover a highly accurate colour image. Our experiments on grayscale datasets show our method (QR with Isotonic regression) offers an SSIM increase of up to **0.7** over MC. However in some cases the increased knowledge of the Sudoku representation is required for QR to out perform MC. Specifically on the grayscale Vistex,

Curet and Ponce datasets LQR offers a performance loss of **0.18**, **0.1** and **0.23** SSIM respectively over LMC, where SQR offers an increment of **0.22**, **0.16** and **0.03** over SMC. This provides further evidence of the greater knowledge encoded by the Sudoku representation from the consistently more accurate reconstructions this description provides. Use of an Isotonic regression tone map increases performance across the board, in some case severely. For instance QR with a tone curve on the Vistex dataset gives an SSIM value of **0.70** vs **0.33** without. Using a tone curve with SQR gives an average SSIM increase of **0.3** across the four datasets.

This contribution is only the second in this new area. As such this new ground has a huge potential for future research. The possibility of investigating the accuracy of a description through its invertibility allows for study of the encoding and representational efficiency. Clearly there remains much future work possible in this area. We now move to our overall conclusions and proposed future work.

Chapter 8

Conclusions and future work

8.1 Overall conclusions

This thesis has concentrated on the understanding and deployment of LBP type texture and colour analysis [redacted] Outside the industrial application of this work, we have also developed a new texture representation and investigated its use in classification and image recovery. We have also proposed and evaluated a new method in the field of image recovery for inverting a texture descriptor to, as far as is possible, the original image (from which the original texture was derived).

[redacted]

[redacted]

Our new technique for texture description, termed "Sudoku", encodes relative intensity variation in a local neighbourhood and then forms a feature vector based on histogram equalisation. Our experiments show an average classification accuracy increase of *12.3%* against LBP and **5.5%** compared with LTP consistent with the improved knowledge of the representation.

We have considered the invertibility of local LBP type texture features, including Sudoku. Using our Sudoku feature we have consistently reconstructed an image more accurately than when using an LBP. Further to this we have presented and evaluated a novel method using quadratic programming which improves reconstruction significantly when compared to the previous art. With our method up to **0.97** SSIM between a reconstructed texture image and the original was observed. On "natural" images we reconstructed an image with an average SSIM of **0.72** compared to the average of **0.29** for the previous art. This new field gives opportunity for much future development.

[redacted]In Chapter 6 we presented a new and novel extension to LBP termed Sudoku. In this chapter we carried out evaluations on standard datasets and showed that for a simple histogram of features type classification system the Sudoku representation advances the state of the art. [redacted] In chapter 7 we consider the problem of calculating full colour images from LBP type texture descriptions.

We show that for LBP and Sudoku descriptors it is possible using our new technique based on Quadratic Programming to recover images from texture with a high degree of accuracy and also significantly better the prior art.

8.2 Future work

This research has taken place during the revolution on deep learning. Deep learning has made many changes to the approaches used for automated image analysis. The techniques used within this thesis have yet to exploit the advantages offered and this is a prudent avenue of future research in this area. It is worth noting that current research on deep learning investigates features that result in classification capability.

[redacted]

With regards to our Sudoku method for classification we believe there is much room for improvement (despite the performance benefits). First and foremost we believe that considering the local rankings as an integer is a crude way of exploiting the representation, even while histogram equalisation does account for this somewhat. We would like a more complete description of our local rankings. The concept of *uniformity* is also interesting, and whether or not it has a deeper significance when the number of local comparisons increases. A full analysis of all pattern occurrences across multiple large datasets would determine whether or not any observations can be made. Finally the possibility of clustering Sudoku ranks in an image to form a set of Textons, similarly to VZ-Joint in [62], may bear some notice.

Image recovery from texture is a very new area and as such there is a huge scope for future work. Firstly we would like to compare other texture descriptors within the framework. We believe that with CLBP (see Section 2.6.4) we should theoretically be able to obtain a perfect reconstruction. QP is the only optimisation technique which we have considered, we would expect a Deep Learning technique, should one be developed, to provide a significant performance increase in the future.

Finally through image recovery from texture we have shown that certain Texture Features can be considered in a *dual sense*. That is, we can go from an image to its texture and back to the image: there are two spaces with a mapping from one to the other and back again. This suggests that the fundamental transforms of LBP and Sudoku can be considered as texture *spaces*. The Fourier transform [121] is a dual transform and some tasks are better placed in one of the two spaces. While the Fourier transform has the property of perfect reconstruction, and QR from texture does not (yet), this concept is still applicable. These texture *spaces* are

something worth exploring as they could open avenues for further understanding of texture and images in general.

Appendix A

[redacted]

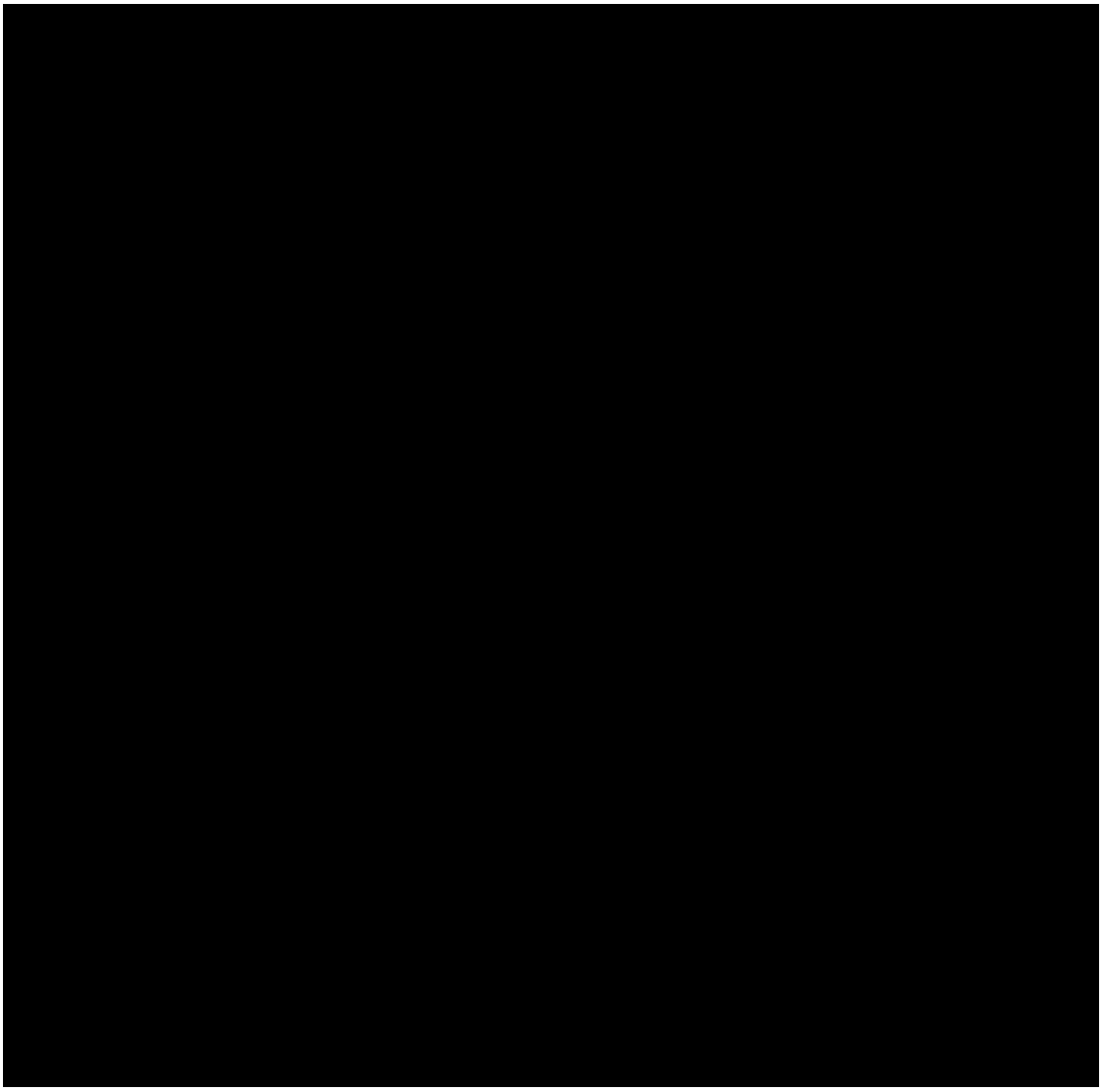
[redacted]

[redacted]

[redacted]

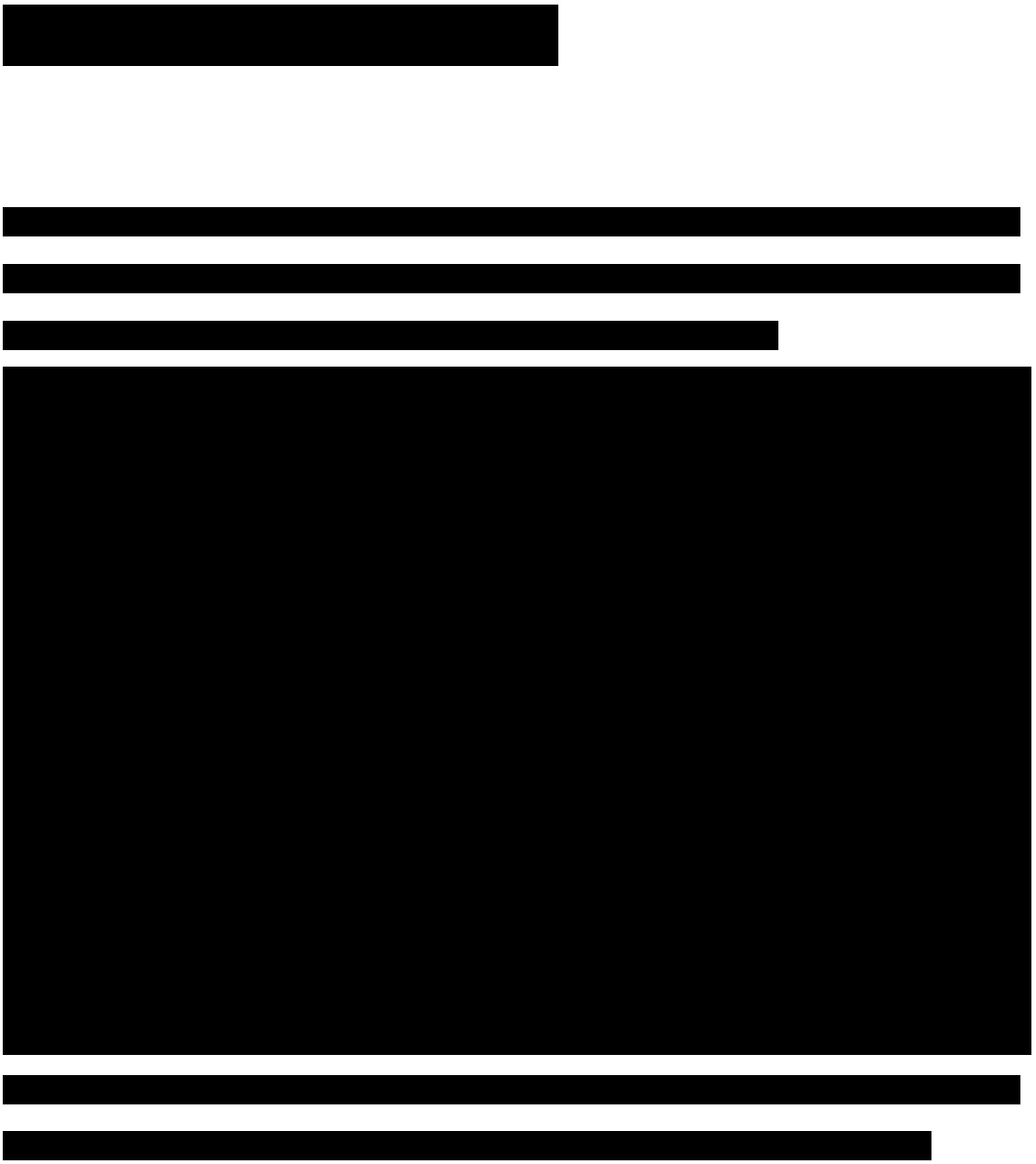
[redacted]

[redacted]



[redacted]

Appendix B



[redacted]

[redacted]

Bibliography

[1] M Varma and A Zisserman. Classifying images of materials: Achieving viewpoint and illumination independence. In *European Conference on Computer Vision*, pages 255–271. Springer, 2002.

[2] RM Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[3] AK Jain, A Ross, and S Prabhakar. Fingerprint matching using minutiae and texture features. In *Image Processing, 2001. Proceedings. 2001 International Conference on*, volume 3, pages 282–285. IEEE, 2001.

[4] C Shan, S Gong, and PW McOwan. Robust facial expression recognition using local binary patterns. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–370. IEEE, 2005.

[5] T Ojala, M Pietikäinen, and T Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on pattern analysis and machine intelligence*, 24(7):971–987, 2002.

[6] M Crosier and LD Griffin. Using basic image features for texture classification. *International journal of computer vision*, 88(3):447–460, 2010.

[7] T Ahonen, A Hadid, and M Pietikäinen. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041, 2006.

[8] DG Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[9] Y Xu, S Huang, H Ji, and C Fermüller. Scale-space texture description on sift-like textons. *Computer Vision and Image Understanding*, 116(9): 999–1013, 2012.

[10] GT Flitton, TP Breckon, and NM Bouallagu. Object recognition using 3d sift in complex ct volumes. In *BMVC*, pages 1–12, 2010.

[11] M Tuceryan and AK Jain. Texture analysis. *Handbook of pattern recognition and computer vision*, 2:235–276, 1993.

[12] F Bianconi, RW Harvey, P Southam, and A Fernández. Theoretical and experimental comparison of different approaches for color texture classification. *Journal of Electronic Imaging*, 20(4):043006, 2011.

[13] A Fernández, MX Álvarez, and F Bianconi. Texture description through histograms of equivalent patterns. *Journal of mathematical imaging and vision*, 45(1):76–102, 2013.

[14] M Pagola, R Ortiz, I Irigoyen, H Bustince, et al. New method to assess barley nitrogen nutrition status based on image colour analysis: comparison with spad-502. *Computers and electronics in agriculture*, 65(2):213–218, 2009.

[15] AJ Perez, F Lopez, JV Benlloch, and S Christensen. Colour and shape analysis techniques for weed detection in cereal fields. *Computers and electronics in agriculture*, 25(3):197–212, 2000.

[16] SR Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14(1):5–16, 1998.

[17] MJ Swain and DH Ballard. Color indexing. *International journal of computer vision*, 7(1):11–32, 1991.

[18] M Kokare, BN Chatterji, and PK Biswas. Comparison of similarity metrics for texture image retrieval. In *TENCON 2003. Conference on Convergent Technologies for the Asia-Pacific Region*, volume 2, pages 571–575. IEEE, 2003.

[19] RO Duda and PE Hart. Pattern classification and scene analysis. *A Wiley-Interscience Publication, New York: Wiley*, 1973.

[20] P Cunningham and SJ Delany. k-nearest neighbour classifiers. *Multiple Classifier Systems*, 34:1–17, 2007.

[21] H Frigui and P Gader. Detection and discrimination of land mines in ground-penetrating radar based on edge histogram descriptors and a possibilistic k-nearest neighbor classifier. *IEEE Transactions on Fuzzy Systems*, 17(1): 185–199, 2009.

[22] H Yan. Handwritten digit recognition using an optimized nearest neighbor classifier. *Pattern Recognition Letters*, 15(2):207–211, 1994.

[23] A Tsymbal, S Puuronen, and DW Patterson. Ensemble feature selection with the simple bayesian classification. *Information fusion*, 4(2):87–100, 2003.

[24] F Peng and D Schuurmans. Combining naive bayes and n-gram language models for text classification. In *European Conference on Information Retrieval*, pages 335–350. Springer, 2003.

[25] GH John and P Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[26] JD Rennie, L Shih, Jaime Teevan, and DR Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 616–623, 2003.

[27] B Gorte and A Stein. Bayesian classification and class area estimation of satellite images using stratification. *IEEE Transactions on Geoscience and Remote Sensing*, 36(3):803–812, 1998.

[28] AM Kibriya, E Frank, B Pfahringer, and G Holmes. Multinomial naive bayes for text categorization revisited. In *Australasian Joint Conference on Artificial Intelligence*, pages 488–499. Springer, 2004.

[29] T Randen and JH Husøy. Filtering for texture classification: A comparative study. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4): 291–310, 1999.

[30] L Nanni, A Lumini, and S Brahnam. Survey on lbp based texture descriptors for image classification. *Expert Systems with Applications*, 39(3):3634–3641, 2012.

[31] H Wechsler. Texture analysisa survey. *Signal Processing*, 2(3):271–282, 1980.

[32] RM Haralick, K Shanmugam, et al. Textural features for image classification. *IEEE Transactions on systems, man, and cybernetics*, (6):610–621, 1973.

[33] T Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

[34] RW Conners and CA Harlow. A theoretical comparison of texture algorithms. *IEEE transactions on pattern analysis and machine intelligence*, (3):204–222, 1980.

[35] K Kvaal, JP Wold, UG Indahl, P Baardseth, and T Næs. Multivariate feature extraction from textural images of bread. *Chemometrics and intelligent laboratory systems*, 42(1-2):141–158, 1998.

[36] JS Weszka, CR Dyer, and A Rosenfeld. A comparative study of texture measures for terrain classification. *IEEE transactions on Systems, Man, and Cybernetics*, (4):269–285, 1976.

[37] Kenneth I Laws. Rapid texture identification. In *Image processing for missile guidance*, volume 238, pages 376–381. International Society for Optics and Photonics, 1980.

[38] I Fogel and D Sagi. Gabor filters as texture discriminator. *Biological cybernetics*, 61(2):103–113, 1989.

[39] SE Grigorescu, N Petkov, and P Kruizinga. Comparison of texture features based on gabor filters. *IEEE Transactions on Image processing*, 11(10): 1160–1167, 2002.

[40] R Chellappa and S Chatterjee. Classification of textures using gaussian markov random fields. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 33(4):959–963, 1985.

[41] JM Ver Hoef, EE Peterson, MB Hooten, EM Hanks, and M Fortin. Spatial autoregressive models for statistical inference from ecological data. *Ecological Monographs*, 88(1):36–59, 2018.

[42] J Mao and AK Jain. Texture classification and segmentation using multiresolution simultaneous autoregressive models. *Pattern recognition*, 25(2): 173–188, 1992.

[43] M Unser. Texture classification and segmentation using wavelet frames. *IEEE Transactions on image processing*, 4(11):1549–1560, 1995.

[44] S Hatipoglu, SK Mitra, and N Kingsbury. Texture classification using dual-tree complex wavelet transform. 1999.

[45] T Celik and T Tjahjadi. Multiscale texture classification using dual-tree complex wavelet transform. *Pattern Recognition Letters*, 30(3):331–339, 2009.

[46] JM Keller, S Chen, and RM Crownover. Texture description and segmentation through fractal geometry. *Computer Vision, Graphics, and image processing*, 45(2):150–166, 1989.

[47] LM Kaplan. Extended fractal analysis for texture classification and segmentation. *IEEE Transactions on Image Processing*, 8(11):1572–1585, 1999.

[48] S Lazebnik, C Schmid, and J Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1265–1278, 2005.

[49] AE Johnson and M Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence*, 21(5):433–449, 1999.

[50] M Cimpoi, S Maji, and A Vedaldi. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3828–3836, 2015.

[51] M Anthimopoulos, S Christodoulidis, L Ebner, A Christe, and S Mougiakakou. Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. *IEEE transactions on medical imaging*, 35(5):1207–1216, 2016.

[52] E Hayman, B Caputo, M Fritz, and J Eklundh. On the significance of real-world conditions for material classification. In *European conference on computer vision*, pages 253–266. Springer, 2004.

[53] F Bianconi and A Fernández. Evaluation of the effects of gabor filter parameters on texture classification. *Pattern recognition*, 40(12):3325–3335, 2007.

[54] BS Manjunath and W Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on pattern analysis and machine intelligence*, 18(8):837–842, 1996.

[55] CBR Ng, G Lu, and D Zhang. Performance study of gabor filters and rotation invariant gabor filters. In *11th International Multimedia Modelling Conference*, pages 158–162. IEEE, 2005.

[56] J Malik, S Belongie, J Shi, and T Leung. Textons, contours and regions: Cue integration in image segmentation. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 918–925. IEEE, 1999.

[57] T Leung and J Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43(1):29–44, 2001.

[58] F Jurie and B Triggs. Creating efficient codebooks for visual recognition. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 604–610. IEEE, 2005.

[59] B Julesz. Textons, the elements of texture perception, and their interactions. *Nature*, 290(5802):91, 1981.

[60] OG Cula and KJ Dana. Recognition methods for 3d textured surfaces. In *Human Vision and Electronic Imaging VI*, volume 4299, pages 209–220. International Society for Optics and Photonics, 2001.

[61] C Schmid. Constructing models for content-based image retrieval. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 2, pages II–II. IEEE, 2001.

[62] M Varma and A Zisserman. Texture classification: Are filter banks necessary? In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–691. IEEE, 2003.

[63] M Varma and R Garg. Locally invariant fractal features for statistical texture classification. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE, 2007.

[64] D He and L Wang. Texture unit, texture spectrum, and texture analysis. *IEEE transactions on Geoscience and Remote Sensing*, 28(4):509–512, 1990.

[65] T Ojala and M Pietikäinen. Unsupervised texture segmentation using feature distributions. *Pattern recognition*, 32(3):477–486, 1999.

[66] Z Guo, L Zhang, and D Zhang. Rotation invariant texture classification using lbp variance (lbpv) with global matching. *Pattern recognition*, 43(3): 706–719, 2010.

[67] X Tan and B Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE transactions on image processing*, 19(6):1635–1650, 2010.

[68] H Jin, Q Liu, H Lu, and X Tong. Face detection using improved lbp under bayesian framework. In *Image and Graphics (ICIG'04), Third International Conference on*, pages 306–309. IEEE, 2004.

[69] X Liu and D Wang. Texture classification using spectral histograms. *IEEE transactions on image processing*, 12(6):661–670, 2003.

[70] V Ojansivu and J Heikkilä. Blur insensitive texture classification using local phase quantization. In *International conference on image and signal processing*, pages 236–243. Springer, 2008.

[71] T Mäenpää and M Pietikäinen. Texture analysis with local binary patterns. In *Handbook of pattern recognition and computer vision*, pages 197–216. World Scientific, 2005.

[72] F Riaz, A Hassan, MY Javed, and MT Coimbra. Detecting melanoma in dermoscopy images using scale adaptive local binary patterns. In *Engineering*

in Medicine and Biology Society (EMBC), 2014 36th Annual International Conference of the IEEE, pages 6758–6761. IEEE, 2014.

[73] S Brahnam, LC Jain, L Nanni, A Lumini, et al. *Local binary patterns: new variants and applications*. Springer, 2014.

[74] RH Chan, C Ho, and M Nikolova. Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization. *IEEE Transactions on image processing*, 14(10):1479–1485, 2005.

[75] P Srivastava, NT Binh, and A Khare. Content-based image retrieval using moments of local ternary pattern. *Mobile Networks and Applications*, 19(5): 618–625, 2014.

[76] C Muramatsu, T Hara, T Endo, and H Fujita. Breast mass classification on mammograms using radial local ternary patterns. *Computers in biology and medicine*, 72:43–53, 2016.

[77] WH Liao. Region description using extended local ternary patterns. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1003–1006. IEEE, 2010.

[78] J Ren, X Jiang, and J Yuan. Relaxed local ternary pattern for face recognition. In *ICIP*, pages 3680–3684, 2013.

[79] Z Guo, L Zhang, and D Zhang. A completed modeling of local binary pattern operator for texture classification. *IEEE Transactions on Image Processing*, 19(6):1657–1663, 2010.

[80] S Liao, MWK Law, and ACS Chung. Dominant local binary patterns for texture classification. *IEEE transactions on image processing*, 18(5):1107–1118, 2009.

[81] M Heikkilä, M Pietikäinen, and C Schmid. Description of interest regions with center-symmetric local binary patterns. In *ICVGIP*, volume 6, pages 58–69. Springer, 2006.

[82] K Mikolajczyk, T Tuytelaars, C Schmid, A Zisserman, J Matas, F Schaffalitzky, T Kadir, and L Van Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1-2):43–72, 2005.

[83] I Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115, 1987.

[84] L Kotoulas and I Andreadis. Colour histogram content-based image retrieval and hardware implementation. *IEE Proceedings-Circuits, Devices and Systems*, 150(5):387–393, 2003.

[85] J Fang and G Qiu. A colour histogram based approach to human face detection. 2003.

[86] MW Mackiewicz, M Fisher, and C Jamieson. Bleeding detection in wireless capsule endoscopy using adaptive colour histogram model and support vector classification. In *Medical Imaging 2008: Image Processing*, volume 6914, page 69140R. International Society for Optics and Photonics, 2008.

[87] RM Henkelman, I Kay, and MJ Bronskill. Receiver operator characteristic (roc) analysis without truth. *Medical Decision Making*, 10(1):24–29, 1990.

[88] JT Townsend. Theoretical analysis of an alphabetic confusion matrix. *Perception & Psychophysics*, 9(1):40–50, 1971.

[89] D Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of mathematical psychology*, 12(4):387–415, 1975.

[90] AA Efros and TK Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[91] J Weickert. *Anisotropic diffusion in image processing*, volume 1. Teubner Stuttgart, 1998.

[92] A Bugeau and M Bertalmio. Combining texture synthesis and diffusion for image inpainting. In *VISAPP 2009-Proceedings of the Fourth International Conference on Computer Vision Theory and Applications*, pages 26–33, 2009.

[93] A Criminisi, P Pérez, and K Toyama. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on image processing*, 13(9):1200–1212, 2004.

[94] D Tschumperlé. Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's. *International Journal of Computer Vision*, 68 (1):65–82, 2006.

[95] J Portilla and EP Simoncelli. Texture modeling and synthesis using joint statistics of complex wavelet coefficients. In *IEEE workshop on statistical and computational theories of vision*, 1999.

[96] WT. Freeman and EH Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9):891–906, 1991.

[97] M Lillholm, M Nielsen, and LD Griffin. Feature-based image analysis. *International Journal of Computer Vision*, 52(2-3):73–95, 2003.

[98] T Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1993.

[99] T Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 465–470. IEEE, 1996.

[100] R Zabih and J Woodfill. Non-parametric local transforms for computing visual correspondence. In *European conference on computer vision*, pages 151–158. Springer, 1994.

[101] O Demetz, D Hafner, and J Weickert. The complete rank transform: A tool for accurate and morphologically invariant matching of structures. In *BMVC*, 2013.

[102] L Liu, P Fieguth, Y Guo, X Wang, and M Pietikäinen. Local binary features for texture classification: taxonomy and experimental study. *Pattern Recognition*, 62:135–160, 2017.

[103] I Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM New York, 2001.

[104] J Kontinen, J Röning, and RM MacKie. Texture features in the classification of melanocytic lesions. In *International Conference on Image Analysis and Processing*, pages 453–460. Springer, 1997.

[105] SM Pizer, PE Amburn, JD Austin, R Cromartie, et al. Adaptive histogram equalization and its variations. *Computer vision, graphics, and image processing*, 39(3):355–368, 1987.

[106] T Mäenpää and M Pietikäinen. Classification with color and texture: jointly or separately? *Pattern recognition*, 37(8):1629–1640, 2004.

[107] T Caelli and D Reye. On the classification of image regions by colour, texture and shape. *Pattern recognition*, 26(4):461–470, 1993.

[108] DL Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. *AMS Math Challenges Lecture*, 1:32, 2000.

[109] M Frank and P Wolfe. An algorithm for quadratic programming. *Naval Research Logistics (NRL)*, 3(1-2):95–110, 1956.

[110] BE Boser, IM Guyon, and VN Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[111] K Muller, S Mika, G Ratsch, K Tsuda, and B Scholkopf. An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks*, 12(2):181–201, 2001.

[112] MD Buhmann. Radial basis functions. *Acta numerica*, 9:1–38, 2000.

[113] O Chapelle, P Haffner, and VN Vapnik. Support vector machines for histogram-based image classification. *IEEE transactions on Neural Networks*, 10(5):1055–1064, 1999.

[114] E Pegg Jr. Ed pegg jr.s math games: Sudoku variations, 2005.

[115] M Rodrigues, M Kormann, and P Tomek. A comparative analysis of binary patterns with discrete cosine transform for gender classification. 2014.

[116] R Spangenberg, T Langner, and R Rojas. Weighted semi-global matching and center-symmetric census transform for robust driver assistance. In *International Conference on Computer Analysis of Images and Patterns*, pages 34–41. Springer, 2013.

[117] T Chakraborti and A Chatterjee. A novel binary adaptive weight gsa based feature selection for face recognition using local gradient patterns, modified census transform, and local binary patterns. *Engineering Applications of Artificial Intelligence*, 33:80–90, 2014.

[118] KJ Dana, B Van Ginneken, SK Nayar, and JJ Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG)*, 18 (1):1–34, 1999.

[119] T Ojala, T Mäenpää, M Pietikäinen, J Viertola, et al. Outex-new framework for empirical evaluation of texture analysis algorithms. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 701–706. IEEE, 2002.

[120] *Vision texture database*. Michigan Institute of Technology, http://www.vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html, 2016.

[121] RN Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

[122] G Finlayson and S Nixon. Sudoku texture classification. *Electronic Imaging*, 2016(14):1–5, 2016.

[123] BM Waller, MS Nixon, and JN Carter. Image reconstruction from local binary patterns. In *2013 International Conference on Signal-Image Technology & Internet-Based Systems*, pages 118–123. IEEE, 2013.

[124] RE Barlow. Statistical inference under order restrictions; the theory and application of isotonic regression. Technical report, 1972.

[125] Z Wang, AC Bovik, HR S, and EP Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[126] T Song, H Li, F Meng, Q Wu, and J Cai. Letrist: locally encoded transform feature histogram for rotation-invariant texture classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 2017.

[127] A Takemura, A Shimizu, and K Hamamoto. Discrimination of breast tumors in ultrasonic images using an ensemble classifier based on the adaboost algorithm with feature selection. *IEEE Transactions on Medical Imaging*, 29(3):598–609, 2010.

[128] H Permuter, J Francos, and IH Jermyn. Gaussian mixture models of texture and colour for image database retrieval. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 3, pages III–569. IEEE, 2003.

[129] N Dalal and B Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.