JAIST Repository
https://dspace.jaist.ac.jp/

Title	Robust Content-based Image Hash Functions Using Nested Lattice Codes
Author(s)	Nguyen Xuan, Thanh
Citation	
Issue Date	2016-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/13630
Rights	
Description	Supervisor:Assoc. Prof. Brian Michael Kurkoski, 情報科学研究科, 修士

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

Robust Content-based Image Hash Functions Using Nested Lattice Codes

By THANH XUAN NGUYEN

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Brian Michael Kurkoski

March, 2016

Robust Content-based Image Hash Functions Using Nested Lattice Codes

By THANH XUAN NGUYEN (1310208)

A thesis submitted to
School of Information Science,
Japan Advanced Institute of Science and Technology,
in partial fulfillment of the requirements
for the degree of
Master of Information Science
Graduate Program in Information Science

Written under the direction of
Associate Professor Brian Michael Kurkoski

and approved by
Professor Tadashi Matsumoto
Professor Masashi Unoki

February, 2016 (Submitted)

Copyright © 2016 by THANH XUAN NGUYEN

Abstract

This contribution uses nested lattice codes to improve fundamental hash functions for particular image retrieval systems and a promising class of similarity search applications. Our proposed content-based image hash function takes advantages of SURF for feature extraction and lattices for quantizing feature vectors to hash values. The goal is to develop a lattice-based hashing scheme such that there is a proportional relationship between Euclidean distance and metric distances (Hamming distance or, as in this thesis, weighted Hamming distance and first difference distance) to increase the hash function's robustness. As a major result, our proposed two-dimensional nested lattice code reduces the normalized mean square error (NMSE) by 20% compared to two-dimensional Gray code.

In terms of similarity search, it has been established as an essential paradigm for a variety of applications, including information retrieval, data mining, multimedia database searching and machine learning. The similarity search problem is to find the object (e.g., image, sound, video, file) most similar to a given object in a set of objects, which are usually represented by a collection of real number feature vectors in Euclidean space. The most simple solution for comparison is sample-by-sample which is computationally slow. Our motivation is comparing data more efficiently by developing a lattice-based hashing scheme.

In terms of a particular image hashing system, content-based hash functions are widely used to index and protect data from distortion attacks that steal or alter data illegally. This method generates the hash value from the image features, then it is suitable for multimedia files indexing which should be able to tolerate some minor modifications. In addition, the content-based approach can be applied to image retrieval systems, such as multimedia database indexing which is simulated in the experiment section of this thesis.

A framework for the content-based hashing system includes feature extraction and quantization. First, the input signal is pre-processed to real feature vectors using signal processing techniques such as singular value decomposition (SVD), speeded up robust features (SURF), scale invariant feature transform(SIFT), Fourier transform and other signal processing operations. Then real feature vectors are converted to binary hash value using codes such as Gray code and Reed-Muller code or, as in this thesis, lattice code. In this research, we concentrate on improving the quantization step by using SURF and nested lattice codes.

Lattices are efficient structures for various geometric, coding and quantization problems. Lattice code has several advantages compared to a Gray code which is

widely used in quantization step. While a Gray code requires a scalar quantizer, lattices employ vector quantization. It is well-known that vector quantizers have lower quantization error than scalar quantizers; therefore, a lattice code is more suitable for quantization.

In summary, this thesis first proposed a weighted Hamming distance and first difference distance as new metrics versus Euclidean distance. The experiment result shows that our proposed metrics are better than traditional Hamming distance in terms of reflecting the similarity between vectors. Second, the Gray code is replaced by a lattice code, a nested lattice indexing scheme is proposed, and multi-dimensional nested lattices experiment. In results so far, the combination of two-dimensional nested lattice and first difference distance reduces the normalized mean square error (NMSE) by 20% compared to two-dimensional Gray code. Finally, to construct a complete content-based image hash function, we used SURF to extract feature vectors in the feature extraction step; then, using nested lattice code for quantization step. This image hash function takes the advantages of both SURF(a robust content-based feature extraction against distortions) and nested lattice (an efficient quantizing scheme).

Keywords: Nested lattice codes, nested lattice indexing, content-based, image hash functions.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Associate Professor Brian Michael Kurkoski for the continuous support of not only my academic research, but also my life in Japan Advanced Institute of Science and Technology (JAIST). He instructed me throughout my study with his knowledge, guidance, and encouragement. Additionally, he granted me an opportunity to conduct research on a collaborative project, which was supported by JSPS Kakenhi Grant Number 26289119.

I also would like to convey my special thanks to Dr. Ha Thanh Le, who gave me valuable advice in the first year of my master program in Vietnam. My sincere thanks also go to Professor Tadashi Matsumoto and Associate Professor Masashi Unoki as committee members of my midterm and final defenses, who sent instructive questions and helpful comments to clarify and improve my research.

Last but not least, I owe my loving thanks to my family and my friends. Without their encouragement and understanding, it would have been impossible for me to accomplish the master program.

Thank you.

Contents

Abstract **i**

Acknowledgements **iii**

1 Introduction **1**
1.1 Research motivation 1
1.2 Research approaches and objectives 2
1.3 A review of image hashing 3
1.4 Contributions 3
1.5 Thesis outline 4

2 Preliminaries **5**
2.1 Lattice code 5
2.1.1 Lattice definition 5
2.1.2 Voronoi region and Voronoi code 6
2.1.3 Coset of lattice 6
2.1.4 Lattice quantizer 7
2.2 Nested Lattice 7
2.3 Gray code and Gray indexing scheme 9
2.3.1 Gray code 9
2.3.2 Gray indexing scheme 10
2.4 Lattice quantizing algorithms 10
2.4.1 Tie and untie functions 10
2.4.2 D_n quantizer 11
2.4.3 Coset lattice quantizer 12
2.4.4 E_8 quantizer 12
2.4.5 A_n quantizer 13
2.4.6 A_2 quantizer 14
2.5 Image feature extraction 16
2.5.1 Interesting point detector 16
2.5.2 Speeded up robust features (SURF) descriptor 19

3 Proposed Algorithm **23**
3.1 Nested lattice indexing 23
3.2 Metrics 24

3.2.1 Euclidean distance . . . 25
3.2.2 Hamming distance . . . 25
3.2.3 Weighted Hamming distance . . . 25
3.2.4 First difference distance . . . 26
3.3 Indexing scheme evaluation based on normalized mean squared error . 26
3.4 The best choice for shift vector **a** . . . 27

4 Experimental results **29**
4.1 Two-dimensional nested lattice indexing schemes A_2 . . . 30
4.2 Nested lattice E_8 indexing schemes simulation . . . 33
4.3 Feature extraction results using SURF . . . 35
4.4 Simulation of Image database . . . 37

5 Conclusion **40**
5.1 New metric distances . . . 40
5.2 Nested lattice indexing scheme . . . 40
5.3 Content-based image hash function . . . 41
5.4 Future work . . . 42

References **44**

Publications **47**

List of Figures

1.1 A framework for hashing system. 2

2.1 One-dimensional lattice. 5
2.2 Voronoi region $V_{4\Lambda}(\mathbf{a})$. 6
2.3 The self-similar hexagonal lattice with nesting factor $k = 3$. 8
2.4 A nested lattice with nesting ratio $r = 3$. 8
2.5 The steps of the reflect and prefix method to generate Gray sequences. 9
2.6 Hexagonal lattice quantizer. 14
2.7 Image response for Gaussian derivatives convolution. By K. Grauman, B. Leibe [29]. 17
2.8 Original input house image. 18
2.9 Detected interesting points using Hessian matrix on house image. . . 18
2.10 Scale pyramid with four down-scale levels. 19
2.11 Scale pyramid with four down-scale levels applied to house image. . . 19
2.12 Choosing dominant orientation using orientation sliding window of size $\frac{\pi}{3}$. 20
2.13 Choosing and dividing square region based on assigned orientation. . 21
2.14 Some demonstrations of the basis SURF descriptor vectors for subregions. 21

3.1 Voronoi region $V_{2\Lambda}(\mathbf{0})$. 28
3.2 Voronoi region $V_{2\Lambda}(\mathbf{a})$, where $\mathbf{a} = 2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2})$. 28

4.1 Case (a): The Hamming distance versus the Euclidean distances of Gray indexing. 31
4.2 Case (a): The weighted Hamming distance versus the Euclidean distances of nested A_2 lattice indexing. 31
4.3 Case (a): The first difference distance versus the Euclidean distances of nested A_2 lattice indexing. 32
4.4 Case (b): The variation of A_2's NMSE as a function of noise variance. 32
4.5 Case (a): The weighted Hamming distance versus the Euclidean distances of nested E_8 lattice indexing. 33
4.6 Case (a): The first difference distance versus the Euclidean distances of nested E_8 lattice indexing. 34
4.7 Case (b): The variation of E_8's NMSE as a function of noise variance. 34
4.8 Surf features of original grayscale cameraman image. 35

4.9 Surf features of compressed cameraman: JPEG 5%. 36
4.10 Surf features of rotated cameraman image: 30°. 36
4.11 Surf features of noised cameraman image. 37
4.12 Image database simulation process. 37
4.13 Cameraman image under some Stirmark Benchmark attacks. 38
4.14 Example queries results. 39

List of Tables

4.1 Case (a): Nested A_2 lattice and Gray indexing scheme simulation information. 30
4.2 Case (a): Nested E_8 lattice indexing schemes results. 33
4.3 Stirmark Benchmark attacks detail. 39
4.4 Database structure, table design. 39

Chapter 1

Introduction

1.1 Research motivation

Similarity search is a common problem in a large variety of applications such as information retrieval, data mining, multimedia database searching and machine learning [1] [2] [3] [4]. A search involves comparing process in a database of objects (e.g., image, sound, video, file). The most simple solution for comparison is sample-by-sample which is computationally slow. Practically, search applications require a response as quickly as possible to user queries. The major motivation of this thesis is comparing data more efficiently by developing a lattice-based hashing scheme.

In terms of particular image hashing system, digital images nowadays are increasingly transmitted over the Internet and between mobile devices such as smartphones. It is easy to make an unauthorized copy and manipulate the content by using widely available image processing software. Therefore, image hash functions are used in image indexing and authentication techniques to index and protect data from distortion attacks that steal or alter data illegally. Cryptographic and content-based hash functions are two primary data hashing techniques. Traditionally, data integrity issues are addressed by cryptographic hash functions which are key-dependent and bit sensitive. This method is usually applied to text message and file authentication which requires all message bits to be unchanged [5] [6] [7]. In contrast, content-based hash functions generate the hash value from the image features. This method is suitable for multimedia files indexing which should be able to tolerate some minor modifications. In addition, the content-based approach can be applied to image retrieval systems [8] [9] [10], such as multimedia database indexing which is simulated in the experiment section of this thesis. Retrieval applications, such as online image search engines, require a response as quickly as possible to user queries. While sample-by-sample image comparison is computationally slow, robust content-based hash functions can compare numerous files in multimedia databases efficiently.

In short, developing an efficient hashing scheme has a promising class of similarity search applications. Specifically, this research focuses on image hashing systems by applying lattice codes.

1.2 Research approaches and objectives

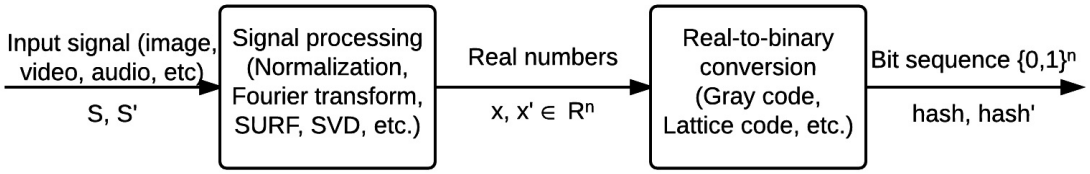

Figure 1.1: A framework for hashing system.

A framework for the content-based hashing system is represented in Figure 1.1. Consider an original input signal $\mathbf{s}$ and its modified signal $\mathbf{s}'$. First, input signal $\mathbf{s}$ is pre-processed to real feature vectors $\mathbf{x}$ using signal processing techniques such as singular value decomposition (SVD) [11], speeded up robust features (SURF) [12], scale invariant feature transform(SIFT) [13], Fourier transform [14] and other signal processing operations. Then real feature vectors are converted to binary hash value $hash$ using codes such as Gray code and Reed-Muller code or, as in this thesis, lattice code. Similarly, modified signal $\mathbf{s}'$ is processed to feature vectors $\mathbf{x}'$, then is converted to hash value $hash'$. In this research, we concentrate on improving the real-to-binary conversion by using SURF and lattice codes.

Euclidean distance is widely known as a good measure of the similarity between features $\mathbf{x}$ and $\mathbf{x}'$ [15]. We let the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}'$ be $d_E = ||\mathbf{x}' - \mathbf{x}||$, and let the metric distance between $hash$ and $hash'$ be $d_M = MetricDistance(hash, hash')$ where d_M represents an arbitrary metric on hash values. By robustness, the greater the difference between two features, the greater the difference of their hash values that is desired [16]. If features $\mathbf{x}$ and $\mathbf{x}'$ are similar, then hash values $hash$ and $hash'$ should also be similar. If $\mathbf{x}$ and $\mathbf{x}'$ are very different, then $hash$ and $hash'$ should also be different. Then, we expect d_E to be proportional to d_M, for a good hash scheme. Using the Euclidean distance between feature vectors allows us to study hashing schemes without considering particular signal processing techniques.

A lattice is a code over an n-dimensional real space, and it has several advantages compared to a Gray code which is widely used in coding step. While a Gray code requires a scalar quantizer, lattices employ vector quantization. It is well-known that vector quantizers have lower quantization error than scalar quantizers [17] [18]; therefore, a lattice code is more suitable for quantization. The goal of this research is to find a hash-value encoding scheme such that the metric distance between hash values is proportional to their Euclidean distance. However, it is impossible to achieve a purely linear relationship, so our objective is to minimize the error between original Euclidean distance and predicted Euclidean distance based on metric distances.

To summarize, we first use the relationship between distances (Euclidean distance versus Hamming distance, weighted Hamming distance and first difference distance) to select the most appropriate code and coding scheme for quantization

step. Afterward, we use SURF technique in the feature extraction step to construct a complete image hashing function. This hashing function finally applies to a multimedia database experiment.

1.3 A review of image hashing

In fact, many hashing schemes have been proposed, but finding a proportional relationship as mentioned above remains a challenge.

In 2000, Venkatesan and Ramarathnam [16] used randomized signal processing strategies and message authentication code (MAC) from cryptography for a non-reversible compression of images into random binary strings. As a result, it minimizes the probability that two hash values may collide and is robust against image changes due to compression, geometric distortions, and other attacks.

From another perspective, in 2011, Parrao et al. [19] used image normalization and SVD as the first signal processing stage, then applied Gray code to obtain the hash binary sequence in image hash functions. According to their paper, the robustness of the hash functions was increased against rotation, scaling, and JPEG attacks.

Faloutsos in 1988 [20], Swaminathan et al. in 2006 [21], Zhu et al. in 2010 [22] also used Gray code as the discrete-binary conversion stage of image hashing to improve clustering of similar records. Those studies show that Gray code is better than natural binary code in binary quantization stage. In 2012, Yuenan et al.[23] proposed hash functions based on random Gabor filtering and dithered lattice vector quantization (LVQ). Basically, a four-dimensional lattice is used for quantization; and then a Gray code normalizes codewords at the end. Their approach can be considered as using Gabor filtering and a combination of a Gray code and a lattice.

1.4 Contributions

This research has three major contributions, including proposing new metric distances, an indexing scheme and a complete content-based hash function for image indexing system.

First, we proposed a weighted Hamming distance and first difference distance as new metrics versus Euclidean distance. The experiment result shows that our proposed metrics are better than traditional Hamming distance in terms of reflecting the similarity between vectors.

Second, the Gray code is replaced by a lattice code, a nested lattice indexing scheme is proposed, and multi-dimensional nested lattices experiment. This research takes advantage of lattices for quantizing feature vectors to hash values. In the quantization step, nested lattices tend to keep a proportional relationship between Euclidean distance and mentioned metric distances (Hamming distance or, as in this thesis, weighted Hamming distance and first difference distance) to increase the hash function's robustness. However, it is impossible to achieve a purely

linear relationship, so our objective is to minimize the mean squared error of linear predictor function from metric distance to Euclidean distance among images using lattice codes. In results so far, the combination of two-dimensional nested lattice and first difference distance reduces the normalized mean square error (NMSE) by 20% compared to two-dimensional Gray code.

Finally, to construct a complete image hash function, we used SURF to extract feature vectors in the first step; then, using nested lattice code for quantization step. This image hash function takes the advantages of both SURF(a robust content-based feature extraction against distortions) and nested lattice(an efficient quantizing scheme).

1.5 Thesis outline

The outline of the remainder of this thesis is as follows. Chapter 2 gives the background of lattice codes, Gray codes, SURF feature extraction, Hamming distance, weighted Hamming distance, first difference distance and Euclidean distance metrics. This chapter also introduces decoding algorithms for some well-known lattices. Chapter 3 explains the proposed nested lattice coding method, evaluation method and how to choose the optimized coset vector $\mathbf{a}$ of nested lattice code. Chapter 4 shows simulation results and performance comparison of Gray code and nested lattice code. Chapter 4 also shows the simulation of a complete image hash function applied to multimedia database indexing. Chapter 5 is the conclusion and future work.

Chapter 2

Preliminaries

2.1 Lattice code

2.1.1 Lattice definition

Lattices are effective structures for various geometric, coding and quantization problems. A lattice Λ is a linear additive subgroup of $\mathbb{R}^n$. In n dimensions, a lattice point $\mathbf{x} \in \Lambda$ is an integral, linear combination of the basis vectors:

$$\mathbf{x} = G \cdot \mathbf{b} = \sum_{i=1}^{n} \mathbf{g_i} b_i, \tag{2.1}$$

where
$\mathbf{b} \in \mathbb{Z}^n$ is a vector of integers;
$G = \begin{bmatrix} \mathbf{g_1} & \mathbf{g_2} & \cdots & \mathbf{g_n} \end{bmatrix}$ is an n-by-n generator matrix;
and $\mathbf{g_i}$ are n-dimensional basis column vectors, for $i \in \{1, 2 \ldots n\}$.

The corresponding fundamental volume of lattice Λ is $V(\Lambda) = \det(\Lambda) = |\det(G)|$. A lattice Λ with expansion factor k forms itself a lattice. We define $k\Lambda$ as a nested lattice with factor k. More detail about nested lattice and its characteristics can be found in sub-section 3.1.

Some well-known lattices are A_2, D_4, E_8 [17]. The simplest lattice is the integers $\mathbb{Z}$ with generator matrix $G = [1]$, as shown in Figure 2.1. In one dimension, all lattices only differ by scale, so they are considered as equivalent.

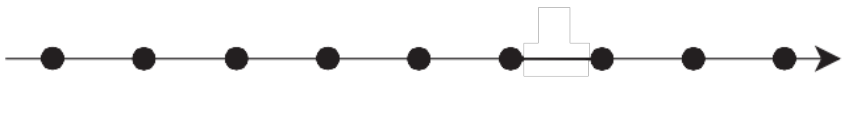

Figure 2.1: One-dimensional lattice.

2.1.2 Voronoi region and Voronoi code

The Voronoi region [24] [25] or fundamental cell $V(\mathbf{x})$ consists of all points of $\mathbb{R}^n$ which are at least as close to $\mathbf{x}$ as to any other lattice point, given by:

$$V(\mathbf{x}) = \{\mathbf{z} \in \mathbb{R}^n : ||\mathbf{z} - \mathbf{x}||^2 < ||\mathbf{z} - \mathbf{y}||^2, \text{ for all } \mathbf{y} \in \Lambda, \mathbf{y} \neq \mathbf{x}\}. \tag{2.2}$$

Let $V_{r\Lambda}(\mathbf{a})$, integer r, vector $\mathbf{a} \in \mathbb{R}^n$, denote the Voronoi region for $r\Lambda$, shifted by vector $\mathbf{a}$. A Voronoi code $C_{r\Lambda}(\mathbf{a})$ consists of every lattice Λ point which is placed inside the Voronoi region $V_{r\Lambda}(\mathbf{a})$:

$$C_{r\Lambda}(\mathbf{a}) = \Lambda \cap V_{r\Lambda}(\mathbf{a}). \tag{2.3}$$

Figure 2.2 depicts 16 lattice points inside a solid line Voronoi region $V_{4\Lambda}(\mathbf{a})$ which is enlarged 4 times from $V_{r\Lambda}(\mathbf{0})$, then translated by vector $\mathbf{a}$.

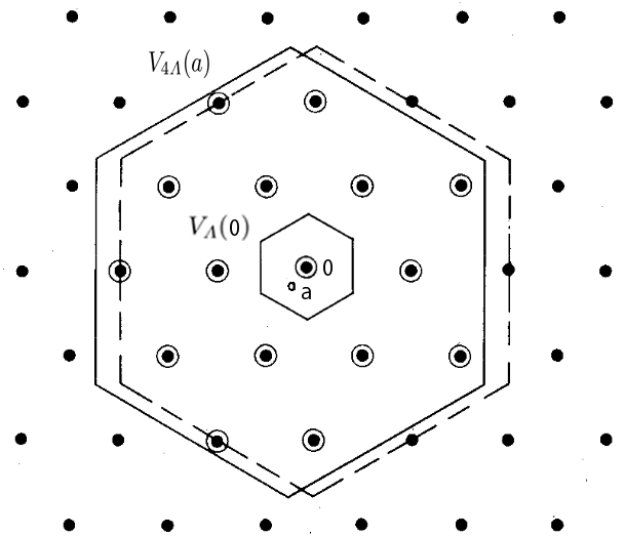

Figure 2.2: Voronoi region $V_{4\Lambda}(\mathbf{a})$.

2.1.3 Coset of lattice

A coset of a lattice can be understood as a shifted version of the lattice, which is shifted by vector $\mathbf{a}$, or formally:

$$\Lambda_{\mathbf{a}} = \mathbf{a} + \Lambda = \{\mathbf{a} + \mathbf{x} : \mathbf{x} \in \Lambda\}. \tag{2.4}$$

The coset itself is not a lattice, but their difference vectors between every pairs of points form a lattice. The union of $\Lambda_{\mathbf{a}}$ over all $\mathbf{a}$ obviously fill the whole space of $\mathbb{R}^n$. Lattices can also be represented by union of a group of cosets. In some

cases, lattice quantization algorithms lead to their cosets quantization algorithms. It means, instead of quantizing a vector to a lattice, we can quantize the vector to the lattice cosets. More detail about decoding algorithms can be found at Section 2.4.

2.1.4 Lattice quantizer

A lattice quantizer maps an n-dimensional input vector $\mathbf{y} = (y_1, y_2, \ldots, y_n)$ to a lattice point $\mathbf{x}^* \in \Lambda$ closest to $\mathbf{y}$, or more formally,

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\Lambda} ||\mathbf{y} - \mathbf{x}||^2, \tag{2.5}$$

where $||\cdot||^2$ denotes squared Euclidean distance. And, a quantization error vector $\mathbf{e}$ is defined as:

$$\mathbf{e} = \mathbf{y} - \mathbf{x}^*. \tag{2.6}$$

The decoding algorithms for many well-known lattices, such as A_n, D_n, E_n are introduced in Section 2.4. In this research, we implemented A_2 and E_8 and applied to our image hashing functions.

2.2 Nested Lattice

Let Λ_1, Λ_2 be two $n-$dimensional lattices with two generator matrixes G_1, G_2 respectively.

$G_1 = \begin{bmatrix} \mathbf{g_{1,1}} & \mathbf{g_{1,2}} & \cdots & \mathbf{g_{1,n}} \end{bmatrix}$ is an n-by-n generator matrix of Λ_1;

$G_2 = \begin{bmatrix} \mathbf{g_{2,1}} & \mathbf{g_{2,2}} & \cdots & \mathbf{g_{2,n}} \end{bmatrix}$ is an n-by-n generator matrix of Λ_2.

According to nested lattice definition in [28], two n-dimensional lattices (Λ_1, Λ_2) are called *nested*, Λ_1 is $fine$ lattice and Λ_2 is $coarse$ lattice, if

$$\Lambda_2 \subset \Lambda_1. \tag{2.7}$$

It means that each basis vector of G_2 must be a combination of all basis vectors of G_1, or formally,

$$G_2 = G_1 \times J, \tag{2.8}$$

$$\mathbf{g}_{2,i} = \sum_{k=1}^{n} j_{k,i}\mathbf{g}_{1,k}. \tag{2.9}$$

where J is n-by-n nesting matrix.

An characteristic of nested lattice is *nesting ratio* r, this number represents the relation between fine lattice and coarse lattice. In detail, consider a pair (Λ_1, Λ_2), where Λ_1 is fine lattice, Λ_2 is coarse lattice, the *nesting ratio* r is:

$$r(\Lambda_1, \Lambda_2) = \sqrt[n]{\frac{V(\Lambda_2)}{V(\Lambda_1)}} = \sqrt[n]{det(J)}. \tag{2.10}$$

In this research, we consider self-similar nested lattices, a special case of nested lattices that satisfy nesting matrix $J = kI$, where I is the identity matrix and nesting factor k is an integer. In short, $\Lambda_2 = k \times \Lambda_1$ with nesting factor k. In this thesis, we call nesting factor k to distinguish self-similar nested lattice from nesting matrix J in general nested lattice cases. Figure 2.3 shows an example of the self-similar hexagonal lattice with nesting factor $k = 3$. Figure 2.4 depicts a two-dimensional nested lattice with nesting ratio $r = 3$, it means the volume of the coarse lattice is nine time bigger than the volume of fine lattice.

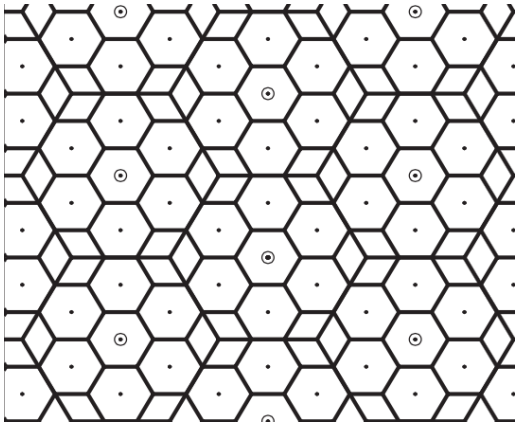

Figure 2.3: The self-similar hexagonal lattice with nesting factor $k = 3$.

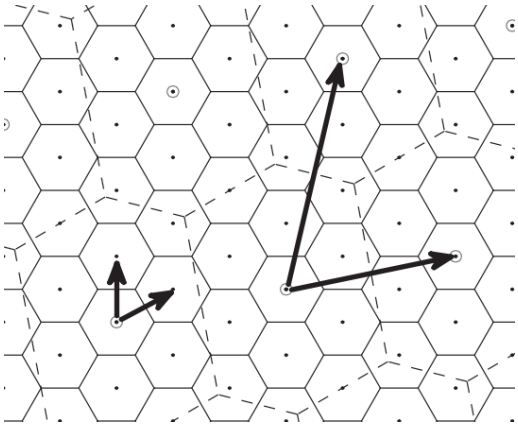

Figure 2.4: A nested lattice with nesting ratio $r = 3$.

2.3 Gray code and Gray indexing scheme

2.3.1 Gray code

As literature review discussion in Section 1.3 above, Gray code is much better than natural binary code and widely used in discrete-to-binary conversion stage of current image hashing. In this research, we used Gray code as a coding scheme to compare with our proposed nested lattice code schemes.

A Gray code is a binary code where two consecutive codewords differ by only one bit. Gray codes are widely used to reduce the number of bit errors in digital communication systems [26]. Gray codes are also known as reflected binary code or single distance code. This section introduces a recursive construction which encodes binary sequences to Gray codes and a straightforward binary-to-gray conversion algorithm which is implemented in this research. Notice that, n^{th} level Gray code G_n represents a class of n bits Gray code, it mean n^{th} level Gray code has 2^n codewords. For instance, G_1 has 2 codewords, G_2 has 4, and G_3 has 8 codewords.

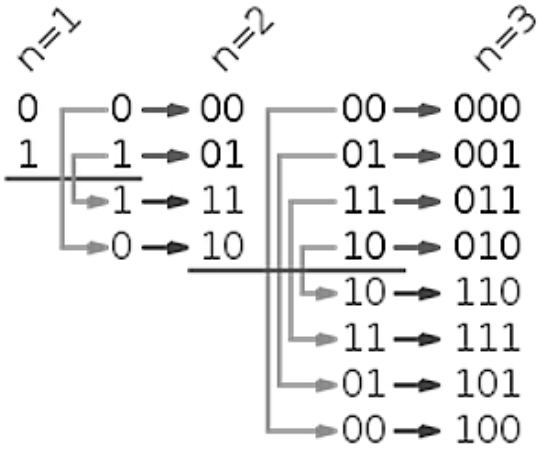

Figure 2.5: The steps of the reflect and prefix method to generate Gray sequences.

Figure 2.5 depicts the algorithm to recursively generate Gray codes from binary sequences for two next levels from level one. In detail, let consider the first level of binary Gray code $G_1 = (0, 1)$ and finding the second level of Gray code G_2. First, copy two codewords as old components from first level of Gray, then reflect two copied codewords as new components. We currently have four temporary codewords $\{0(old), 1(old), 1(new), 0(new)\}$. Second, add prefix 0 to old components, add prefix 1 to new components, we have the second level of Gray code $G_2 = \{00, 01, 11, 10\}$. Similarly, we can generate the $(n + 1)^{th}$ level of Gray code from n^{th} of Gray code with $n \in \mathbb{N}$.

In terms of direct conversion between Gray code and natural binary code, consider a binary sequence $\mathbf{x} = \{d_1, d_2, \ldots, d_{n-1}, d_n\}$ and finding the Gray code of $\mathbf{x}$. To convert $\mathbf{x}$ to Gray code, start from the last bit (the right side) d_n. If the d_{n-1} is 0 copy $g_n = d_n$; otherwise replace $g_n = 1 - d_n$. Then proceed to the next component, $d_{n-1}, \ldots, d_1$ until hit the first component d_1. For the first component d_1, g_1 will

be assigned 0. The sequence $g_1, g_2, \ldots, g_n$ is the corresponding Gray code of input binary sequence $\mathbf{x}$. More formally,

$$g_i = \begin{cases} 0 & \text{if } i = 1 \\ 1 - d_i & \text{if } i \neq 1; d_{i-1} = 1 \\ d_i & \text{if } i \neq 1; d_{i-1} = 0 \end{cases} \text{, for } i = 1, \ldots, n. \tag{2.11}$$

2.3.2 Gray indexing scheme

In this sub-section, we introduce $m \times n$ bits Gray code to index an n-dimensional real input vector $(x_1, \ldots, x_n)$ for $x_i \in [0, 2^m], i = \{1, \ldots, n\}$, uniformly distributed. First, input vector $(x_1, \ldots, x_n)$ is quantized to closest integer vector by rounding to the nearest integer component:

$$(y_1, \ldots, y_n) = Q_{Integer}(x_1, \ldots, x_n). \tag{2.12}$$

Second, translate integer vector $(y_1, \ldots, y_n)$ into binary sequences $(b_1, \ldots, b_n)$. Then, apply binary-to-gray conversion algorithm mentioned above to convert binary sequences b_i length m to Gray sequences c_i length m. Finally, those n sequences of Gray code were concatenated to form an unique binary hash sequence.

$$GrayHash = c_1, \ldots, c_n. \tag{2.13}$$

2.4 Lattice quantizing algorithms

Efficient quantizing algorithms of lattices (A_n, E_n) were introduced by John Conway [17] [30]. In this thesis, we use those techniques to quantize image's feature vectors into lattice points. This section summarizes quantizing algorithms for some well-known lattices, such as A_2 and E_8. In short, quantizing algorithms find the closest lattice points of an input vector.

2.4.1 Tie and untie functions

For later use, *tie* and *untie* functions are defined. Basically, tie and untie functions are used to calculate closest and second closest integer points to a given point, respectively. Consider an $n-$dimensional vector $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, let tie function $f(\mathbf{x})$ round $\mathbf{x}$ to closest integer vector.

$$f(\mathbf{x}) = (f(x_1), f(x_2), \ldots, f(x_n)). \tag{2.14}$$

In contrast, the untie function $g(\mathbf{x})$ rounds vector $\mathbf{x}$ in a semi-wrong way. All elements of $\mathbf{x}$ are rounded to the closest integer except element that furthest from an integer. Let $w(x)$ be the wrong round function, and then the untie function is defined as:

$$g(\mathbf{x}) = (f(x_1), f(x_2), \ldots, w(x_k), \ldots, f(x_n)). \tag{2.15}$$

where x_k is the element of the vector $\mathbf{x}$ with the furthest distance from an integer.

In detail, round function $f(x)$ and wrong round function $w(x)$ are formally defined as bellow:

$$x = 0, \qquad then f(x) = 0, w(x) = 1; \tag{2.16}$$

$$0 < m \leq x \leq m + \frac{1}{2}, \qquad then f(x) = m, w(x) = m + 1; \tag{2.17}$$

$$0 < m + \frac{1}{2} \leq x \leq m + 1, \qquad then f(x) = m + 1, w(x) = m; \tag{2.18}$$

$$-m - \frac{1}{2} \leq x \leq -m < 0, \qquad then f(x) = -m, w(x) = -m - 1; \tag{2.19}$$

$$-m - 1 \leq x \leq -m - \frac{1}{2} < 0, \qquad then f(x) = -m - 1, w(x) = -m. \tag{2.20}$$

where $x \in \mathbb{R}$, $m \in \mathbb{Z}$.

2.4.2 D_n quantizer

$D_n (n \geq 2)$ lattices just include $n-$dimensional integer vectors that have even sums. The process of finding the closest D_n lattice point to $\mathbf{x} = (x_1, x_2, ..., x_n)$ can be applied the idea of the tie and untie functions.

As we mentioned above, $f(\mathbf{x})$ is the closest integer vector to $\mathbf{x}$ with smallest squared Euclidean distance, $g(\mathbf{x})$ is the second closest integer vector to $\mathbf{x}$. According to the definition of the tie and untie functions, vector $f(\mathbf{x})$ and $g(\mathbf{x})$ differ by exactly one element x_k. Therefore, the difference between the sum of elements of $f(\mathbf{x})$ and the sum of elements of $g(\mathbf{x})$ equals the difference between $f(x_k)$ and $g(x_k)$ and equals one. More formally,

$$\begin{aligned} &\sum f(x) - \sum g(x) \\ &= (f(x_1), f(x_2), \ldots, f(x_k), \ldots, f(x_n)) \\ &- (f(x_1), f(x_2), \ldots, w(x_k), \ldots, f(x_n)) \\ &= f(x_k) - w(x_k) \\ &= 1. \end{aligned} \tag{2.21}$$

As a result, $\sum f(x)$ and $\sum g(x)$ are two consecutive integer numbers, one is even, the other is odd. Therefore, either $f(x)$ or $g(x)$ is a D_n lattice point that satisfies two conditions: Closest to vector $\mathbf{x}$ and has an even sum of components.

2.4.3 Coset lattice quantizer

The basic idea of coset lattice quantizer is finding the closest point to a lattice by finding the closest points to their cosets. Suppose that a lattice Λ has d cosets, and then lattice Λ can be represented as a union of d cosets:

$$\Lambda = \bigcup_{0}^{d-1} (\mathbf{r_i} + \Lambda). \tag{2.22}$$

Given a vector $\mathbf{x}$ in $n-$dimensional space, $h(\mathbf{x})$ is the closest point of Λ to $\mathbf{x}$. As a result, the nearest point of $\mathbf{r} + \Lambda$ to $\mathbf{x}$ is $y = h(\mathbf{x} - \mathbf{r}) + \mathbf{r}$. So far, finding the closest point to a coset is possible. Instead of finding closest lattice point Λ to $\mathbf{x}$, we find closest point $\mathbf{y_i} = h(\mathbf{x} - \mathbf{r}) + \mathbf{r}$ of all coset $\mathbf{r_i} + \Lambda$ to $\mathbf{x}$. Then comparing the distances from $\mathbf{y_i}$ to $\mathbf{x}$ to select the closest point.

2.4.4 E_8 quantizer

In this sub-section, we introduce how to decode E_8 lattice with the idea of dual lattice quantizer above. E_8 can be represented by the union of two D_8 cosets.

$$E_8 = D_8 \cup (D_8 + \frac{\mathbf{1}}{\mathbf{2}}), \tag{2.23}$$

where

$$\frac{\mathbf{1}}{\mathbf{2}} = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}).$$

Applying the dual lattice quantizer idea, we can find closest lattice point to an eight-dimensional vector by finding closest D_8 cosets point and its coset $D_8 + \frac{\mathbf{1}}{\mathbf{2}}$. D_8 decoding process is explained in D_n quantizer Section above.
Given vector $\mathbf{x}$:

$$\mathbf{x} = (x_1, x_2, \ldots, x_8) \in \mathbb{R}^8. \tag{2.24}$$

First, find the closest D_8 lattice point to $\mathbf{x}$ by calculating $f(\mathbf{x})$ and $g(\mathbf{x})$, and then select the vector which has an even sum of coordinates, named $\mathbf{y_0}$.
Second, find the closest $D_8 + \frac{\mathbf{1}}{\mathbf{2}}$ lattice point to $\mathbf{x}$. Similarly, compute $f(\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}})$ and $g(\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}})$, then choose the even sum vector, named $\mathbf{y_1}$.
Finally, calculate and compare the Euclidean distance between $\mathbf{y_0}$ and $\mathbf{y_1}$ to $\mathbf{x}$ and choose the vector which has smaller distance.

For example, to find the closest point of E_8 to

$$\mathbf{x} = (0.1, 0.2, 0.2, 1.4, 1.7, -0.8, -0.7, 1.8).$$

First, we compute:

$$f(\mathbf{x}) = (0, 0, 0, 1, 2, -1, -1, 2), sum = 3, odd.$$

As we can see, the fourth element of $f(\mathbf{x})$ is the worst round with biggest error $|1 - 1.4| = 0.4$. Then 4^{th} element will be selected to round in wrong way in $g(\mathbf{x})$.

$$g(\mathbf{x}) = (0, 0, 0, 2, 2, -1, -1, 2), sum = 4, even.$$

and take $\mathbf{y_0} = g(\mathbf{x}) = (0, 0, 0, 2, 2, -1, -1, 2)$.
We similarly compute $f(\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}})$ and $f(\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}})$:

$$\begin{aligned}
\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}} &= (-0.4, -0.3, -0.3, 0.9, 1.2, -1.3, -1.2, 1.3), \\
f(\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}) &= (0, 0, 0, 1, 1, -1, -1, 1), sum = 1, odd, \\
g(\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}) &= (-1, 0, 0, 1, 1, -1, -1, 1), sum = 0, even.
\end{aligned}$$

so

$$\begin{aligned}
\mathbf{y_1} &= g(\mathbf{x} - \frac{\mathbf{1}}{\mathbf{2}}) + \frac{1}{2} \\
&= (-0.5, 0.5, 0.5, 1.5, 1.5, -0.5, -0.5, 1.5).
\end{aligned}$$

Finally,

$$||\mathbf{x} - \mathbf{y_0}||^2 = 0.71, ||\mathbf{x} - \mathbf{y_1}||^2 = 0.81.$$

In conclusion, $\mathbf{y_0} = g(\mathbf{x}) = (0, 0, 0, 2, 2, -1, -1, 2)$ is the closest point of E_8 to vector $\mathbf{x} = (0.1, 0.2, 0.2, 1.4, 1.7, -0.8, -0.7, 1.8)$.

2.4.5 A_n quantizer

This section introduces how to find closest $A_n (n \geq 1)$ lattice point to an arbitrary point. Consider vector $\mathbf{x} \in \mathbb{R}^{n+1}$, quantize vector $\mathbf{x}$ to closest A_n lattice point.
First, compute the sum of all elements of $\mathbf{x}$ and replace $\mathbf{x}$ by:

$$s = \sum_{i=0}^{n} x_i, \tag{2.25}$$

$$\mathbf{x}' = \mathbf{x} - \frac{s}{n+1}(1, 1, \ldots, 1). \tag{2.26}$$

Then compute $f(\mathbf{x}') = (f(x'_0), \ldots, f(x'_n))$ and sum of $f(\mathbf{x}')$'s elements $\Delta = \sum_{i=0}^{n} f(x'_i)$. Afterward, compute $\delta(x'_i)$, then sort x'_i in order of $\delta(x'_i)$ increasingly.

$$\delta(x'_i) = x'_i - f(x'_i). \tag{2.27}$$

$$-\frac{1}{2} \leq \delta(x'_{i_0}) \leq \delta(x'_{i_1}) \ldots \leq \delta(x'_{i_n}) \leq \frac{1}{2}. \tag{2.28}$$

Finally,
If $\Delta = 0$, $f(\mathbf{x}')$ is the closest point of A_n to $\mathbf{x}$.
If $\Delta > 0$, the closest point of A_n to $\mathbf{x}$ is obtained by subtracting 1 from the components $f(x'_{i_0}), \ldots, f(x'_{i_{\Delta-1}})$.
If $\Delta < 0$, the closest point is obtained by adding 1 to the components $f(x'_{i_n}), f(x'_{i_{n-1}}), \ldots, f(x'_{i_{n-\Delta+1}})$.

2.4.6 A_2 quantizer

A_2 lattice or hexagonal lattice quantizer can use the same technique as A_n above. However, in this sub-section, we introduce another algorithm, named shifting A_2 quantizer which was implemented in this research. The basic idea of shifting A_2 is considering original hexagonal lattice as a union of a lattice and its coset.

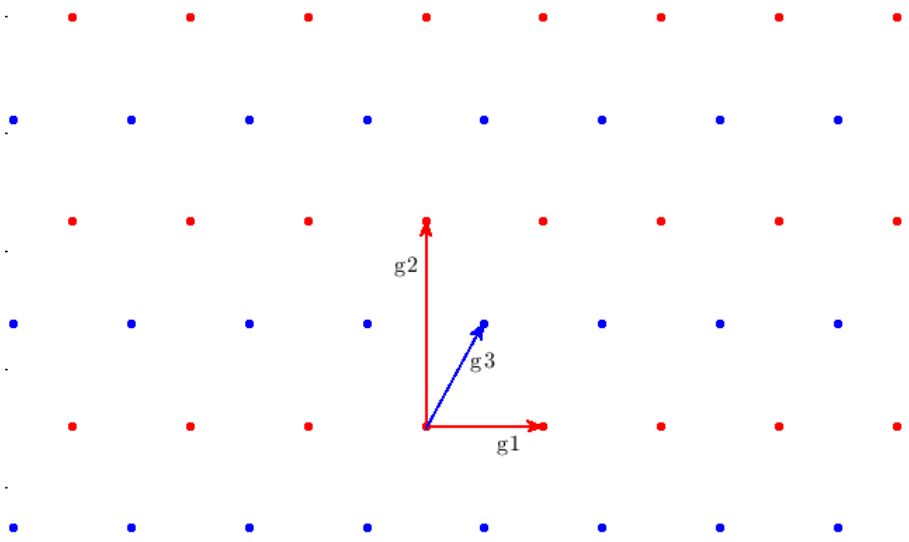

Figure 2.6: Hexagonal lattice quantizer.

Visually, Figure 2.6 shows an A_2 hexagonal lattice which has generator matrix G and consists of both red dots and blue dots. As we can see, set of red dots forms a two-dimensional lattice, called A_2^{red}, has generator matrix G^{red}. Besides, set of blue

dots forms a two-dimensional coset lattice, called A_2^{blue}, has generator matrix G^{blue}. The generator matrixes are defined as:

$$\mathbf{g_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{g_2} = \begin{bmatrix} 0 \\ \sqrt{3} \end{bmatrix}, \mathbf{g_3} = \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}, \tag{2.29}$$

$$G = \begin{bmatrix} \mathbf{g_1} & \mathbf{g_3} \end{bmatrix}, \tag{2.30}$$

$$G^{red} = \begin{bmatrix} \mathbf{g_1} & \mathbf{g_2} \end{bmatrix}, \tag{2.31}$$

$$G^{blue} = G^{red} + \mathbf{g_3} = \begin{bmatrix} \mathbf{g_1} + \mathbf{g_3} & \mathbf{g_2} + \mathbf{g_3} \end{bmatrix}. \tag{2.32}$$

As we can see, hexagonal lattice A_2 is equal to the union of the red lattice and the blue coset lattice: $A_2 = A_2^{red} \cup A_2^{blue}$. So, quantizing an arbitrary point to hexagonal A_2 lattice means finding the closest point to A_2^{red} and A_2^{blue}. This algorithm firstly consider quantizing a point to A_2^{red}, then using shift technique to consider the closest point to coset A_2^{blue}.

In terms of A_2^{red} lattice, two basis vectors $\mathbf{g_1}$ and $\mathbf{g_2}$ are orthogonal, so the integer formation of A_2^{red} forms two-dimensional integers, or other word, $\mathbb{Z}^2$. A_2^{red} is firstly transformed into integer format as bellow:

$$\mathbf{x} = \mathbf{b}G^{red} \Rightarrow \mathbf{b} = (G^{red})^{-1}\mathbf{x}. \tag{2.33}$$

Now, finding closest integer vector to $\mathbf{b}$ means applying tie-function to vector $\mathbf{b}$, so $f(\mathbf{b})$ is the closest $\mathbb{Z}^2$ vector to $\mathbf{b}$. Then, inverse transform $f(\mathbf{b})$ to A_2^{red} by multiplying corresponding generator matrix.

$$y^{red} = f(\mathbf{b})G^{red}. \tag{2.34}$$

In terms of A_2^{blue}, this coset lattice is A_2^{red} shifted by vector $\mathbf{g_2}$. Then, the closest vector of A_2^{blue} to $\mathbf{x}$ is calculated as:

$$y^{blue} = (f(\mathbf{b} - \frac{\mathbf{1}}{\mathbf{2}}) + \frac{\mathbf{1}}{\mathbf{2}})G^{red}. \tag{2.35}$$

Finally, comparing the Euclidean distance between y^{red} and y^{blue} to $\mathbf{x}$ to determine the closest point with smaller distance.

2.5 Image feature extraction

As we mentioned above in sub-section 1.2, a typical hashing framework includes two main processes: Feature extraction and Quantization. In general, the input signal can be files, images, videos, etc. Depending on input characteristics, each type of input signal requires their appropriate feature extraction techniques.

In the scope of this research, we only consider the image as the input signal. This sub-section introduces SURF, which has been applied to images in our research. That technique was proved that outperforms state-of-the-art both in speed and accuracy in 2008 [12]. A feature extraction technique normally includes a detector and a descriptor. The detector is responsible for selecting interesting points (or key points) that can represent image's characteristics. Then, descriptor describes the detected interesting points by numbers that contain as much information as possible with low complexity and keeping sufficiently distinctive.

2.5.1 Interesting point detector

Many detectors have been already proposed, but, keep in mind that the purpose is extracting interesting points from the input image. A suitable detector can select the same interesting points as much as possible under image distortions, such as rotation, scaling, noise, or any other kind of attacks. It means that the selected interesting points should not change even when the image has been distorted. That leads interesting points regularly be corners, edges and strong texture that usually keep stable under attacks.

SURF uses Hessian detector that relies on Hessian matrix [12]. Concretely, consider a pixel $\mathbf{a} = (x, y)$ in a $2D$ image I, the $\mathbf{a}$'s Hessian matrix and its determinant are defined as:

$$H(\mathbf{a}, s) = \begin{bmatrix} I_{xx}(\mathbf{a}, s) & I_{xy}(\mathbf{a}, s) \\ I_{xy}(\mathbf{a}, s) & I_{yy}(\mathbf{a}, s) \end{bmatrix}, \tag{2.36}$$

$$det(H(\mathbf{a}, s)) = \begin{vmatrix} I_{xx}(\mathbf{a}, s) & I_{xy}(\mathbf{a}, s) \\ I_{xy}(\mathbf{a}, s) & I_{yy}(\mathbf{a}, s) \end{vmatrix} = I_{xx}(\mathbf{a}, s)I_{yy}(\mathbf{a}, s) - I_{xy}^2(\mathbf{a}, s). \tag{2.37}$$

where
s is the scale level of the current image. $I_{xx}(\mathbf{a}, s)$ is the convolution of Gaussian second order derivative $\frac{\partial^2}{\partial x^2}$ on x with I at pixel $\mathbf{a}$. Similarly, $I_{yy}(\mathbf{a}, s)$ is the convolution of Gaussian second order derivative $\frac{\partial^2}{\partial y^2}$ on y with I at pixel $\mathbf{a}$.$I_{xy}(\mathbf{a}, s)$ is the convolution of Gaussian second order derivative $\frac{\partial^2}{\partial x \partial y}$ on both x and y with I at $\mathbf{a}$.

The convolution between I and Gaussian derivative obtains derivative responses in vertical and horizontal directions all over the image in pyramid scale. The scale

s in scale pyramid guarantee that detector selects best group of interesting points against scaling attack. Then, searching for strong derivative responses by comparing determinant values of Hessian matrixes.

Figure 2.7 visualizes the image's response with Gaussian derivative. In this experiment, input image simply has a white background, some polygons, lines and dots. Obviously, the interesting points should be the polygon's corners, head and tail of lines, and highlighted dots because they are difficult to disappear by scaling, rotation or other kinds of attacks. In detail, the top-left image represents I_{xx} that responses to Gaussian second order derivative $\frac{\partial^2}{\partial x^2}$. As we can see, I_{xx} highlights vertical textures. Similarly, I_{yy} (bottom-right image) and $I_{x,y}$ (bottom-left) emphasize the horizontal textures and other directions. In short, each element of Hessian matrix attempts to capture the texture in determining directions.

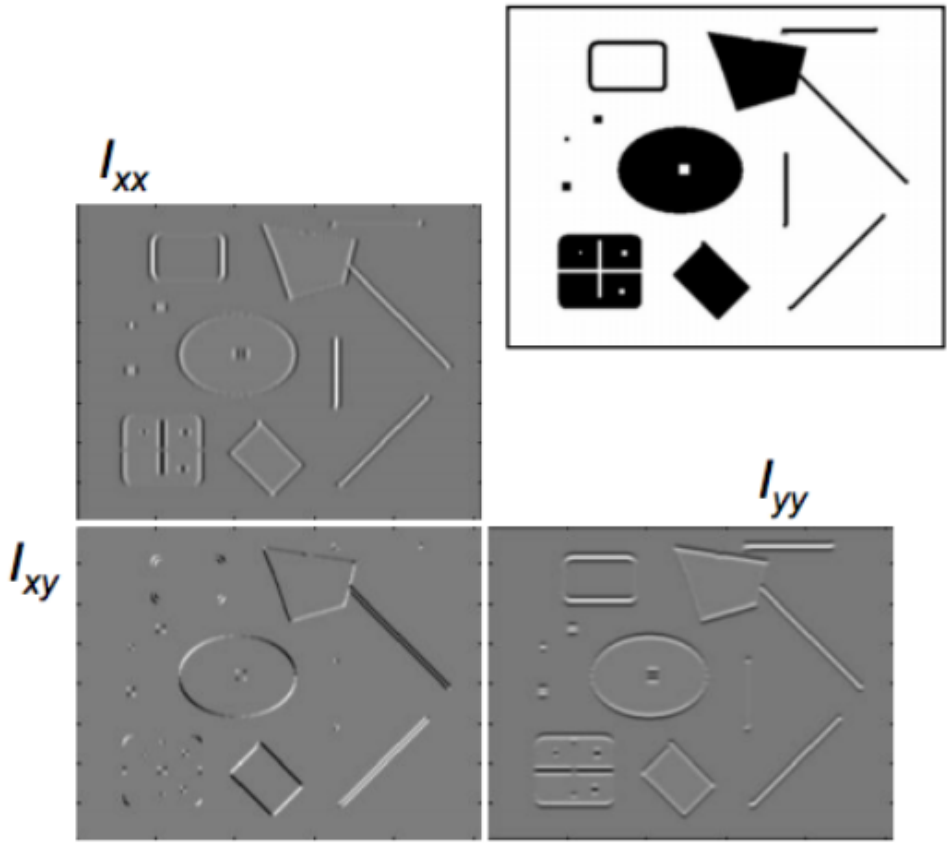

Figure 2.7: Image response for Gaussian derivatives convolution. By K. Grauman, B. Leibe [29].

Figure 2.8 and Figure 2.9 show detected interesting points using Hessian matrix on a test image with four levels down-scale pyramid (As shown in Figure 2.10 and Figure 2.11). The green circles represent the detected interesting points on the house

image. As we can see, almost all detected interesting points located in corners and strong texture regions.

Figure 2.8: Original input house image.

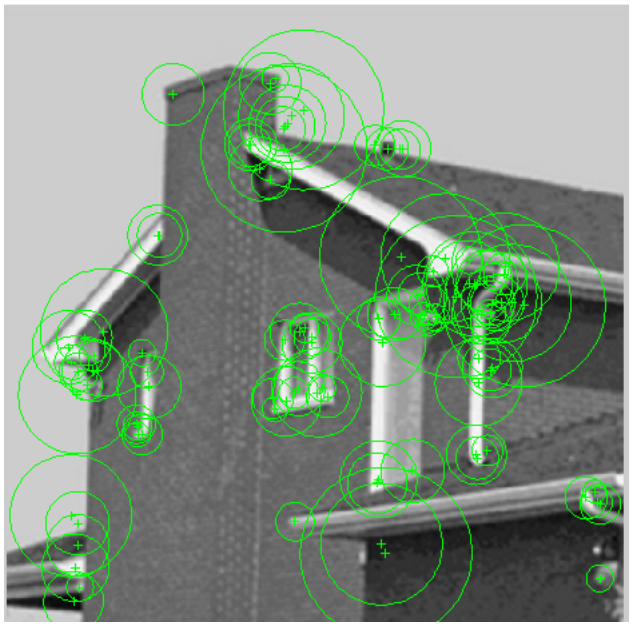

Figure 2.9: Detected interesting points using Hessian matrix on house image.

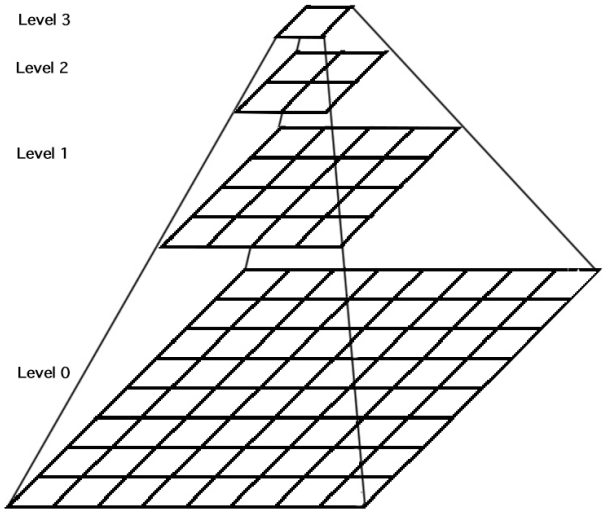

Figure 2.10: Scale pyramid with four down-scale levels.

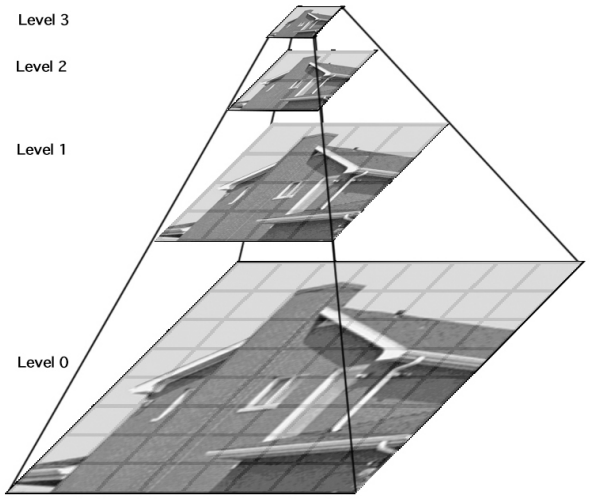

Figure 2.11: Scale pyramid with four down-scale levels applied to house image.

2.5.2 Speeded up robust features (SURF) descriptor

In general, descriptors are used to describe detected interesting points of images. SURF descriptor obtains image characteristics from local regions centralize at the interesting point based on orientation. Consider an interesting point, SURF descrip-

tor firstly assigns an orientation to the current interesting point. Then, a descriptor based on Haar wavelet filters is used to gather information around the interesting point in the selected orientation.

Interesting point orientation assignment

This step attempts to assign a direction to interesting points, so a descriptor can relatively describe the key points. As a result, the SURF descriptor can be invariant to image rotation.

Concretely, a circular region centralizes at the key point of radius $6s$ is considered, where s is the scale level of selected key point, we named it *interesting region*. Then, Haar wavelet filter is applied to interesting region in both vertical and horizontal directions. After that, Haar wavelet responses are weighted by a Gaussian function with center at key point and standard deviation $\sigma = 2s$. The output of Haar wavelet filtering and Gaussian weighting are two-dimensional vectors corresponding to points in the interesting region. Figure 2.12 visualizes how to choose dominant orientation apparently. An orientation sliding window of size $\frac{\pi}{3}$ is applied over the interesting region circle to calculate sum of all output vectors located inside current window. The longer sum vector, the more sensitive the interesting region response to Haar wavelet filter. Finally, the dominant orientation is the one that has the longest sum vector.

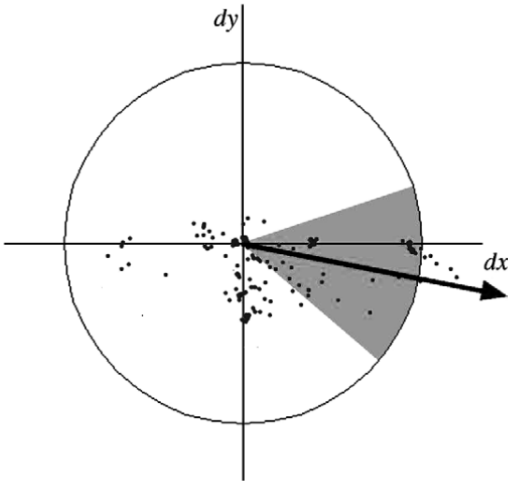

Figure 2.12: Choosing dominant orientation using orientation sliding window of size $\frac{\pi}{3}$.

Local descriptor based on Haar wavelet filter

As a result of orientation assignment, this step firstly chooses a square region of size $20s$, centralizes at key point and orients on selected direction. Afterward, the square region is divided into 4×4 sub-square regions of 5×5 samples, as shown

in Figure 2.13. For each sub-region, we apply the Haar wavelet filter and calculate the responses. However, the SURF descriptor does not use the responses directly, a four-dimensional vector **Haar** based on Haar wavelet responses is extracted.

$$\mathbf{Haar} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|). \tag{2.38}$$

where d_x is the horizontal Haar wavelet response (filter size $2s$); $\sum d_x$ is the sum of all d_x responses within one sub-region; d_y is the vertical Haar wavelet response (filter size $2s$); $\sum d_y$ is the sum of all d_y responses with in one sub-region.

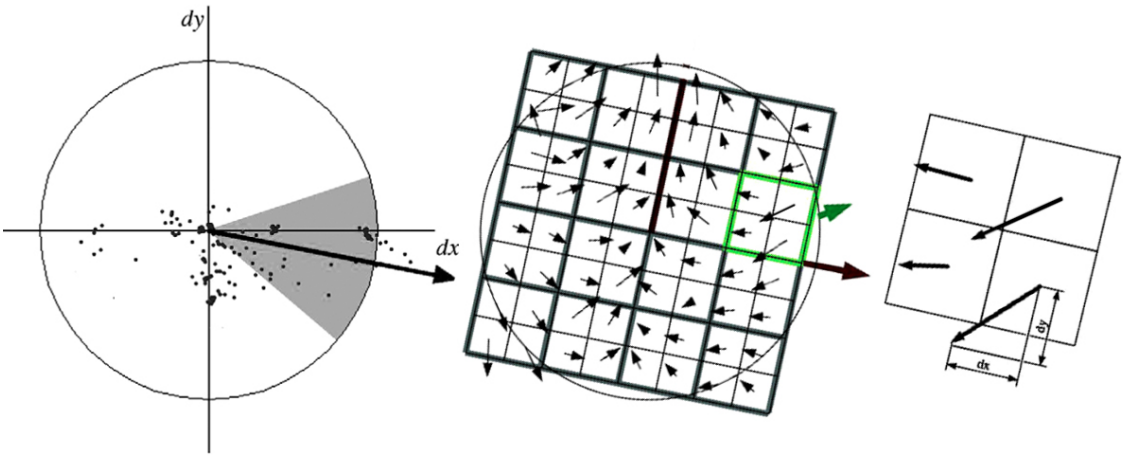

Figure 2.13: Choosing and dividing square region based on assigned orientation.

Figure 2.14: Some demonstrations of the basis SURF descriptor vectors for sub-regions.

Figures 2.14 shows the *Haar* vectors of three basic sub-region image patterns. As we can see, three patterns are obviously distinctive with *Haar* vectors. As a result,

Haar vector is also powerful to distinguish the combination of three basic patterns above.

Every sub-region is represented by a four-dimensional vector **Haar**, a square region have 4×4 sub-region, so an interesting point is represented by a $64-$dimensional vector. In short, a SURF descriptor vector is a $64-$dimensional vector, which is synthesized from Haar wavelet responses of a square image region.

Chapter 3

Proposed Algorithm

A hash scheme maps real numbers to bits. A good hash scheme will preserve distance as well as possible. That is, the Euclidean distance between two points in the real space should be proportional to the metric distance between the hash values of those two points. Our indexing scheme quantizes points into nested lattice points in multiple levels. If two points are far from each other, they tend to be quantized to different lattice points in high levels. In contrast, if two points are close together in Euclidean space, they should be quantized to same lattice points in high levels, and quantized to different lattice points only at low levels. To preserve distance, the higher the level, the higher the weight the bits groups should be assigned. On the other hand, consider from the highest level of nested lattice to the lowest level, the index of the first different position also represents the difference between two bit sequences. That is the idea for applying weighted Hamming distance in order to assign weights exponentially to every group of bits of codewords.

3.1 Nested lattice indexing

We indexed n-dimensional real points using m levels nested lattice in n-dimensional space. Consider an n-dimensional real input datapoint $\mathbf{x} \in [0, 2^{m-1}]^n$, uniformly distributed. We define lattices Λ_{2^i} by an n-by-n generator matrix $G_{\Lambda_{2^i}}$:

$$G_{\Lambda_{2^i}} = 2^i G_\Lambda = 2^i \begin{bmatrix} \mathbf{g_1} & \mathbf{g_2} & \cdots & \mathbf{g_n} \end{bmatrix}, \text{ for } i \in \{0, 1 \ldots (m-1)\}. \tag{3.1}$$

We also define a shift vector $\mathbf{a}$ as below.

$$\mathbf{a} = [a_1, \ldots, a_n]. \tag{3.2}$$

The best choice of shift vector $\mathbf{a}$ is explained in detail in Section 3.4.

An algorithm for finding a hash value $LatticeHash$ from a real input vector $\mathbf{x}$ is given in three main steps.

Step 1: We shift the real input $\mathbf{x}$ by vector $2^i\mathbf{a}$ at the corresponding level 2^i or more formally,

$$\mathbf{y}^{(i)} = \mathbf{x} - 2^i\mathbf{a}, \text{ for } i \in \{0, 1 \ldots (m-1)\}. \tag{3.3}$$

Step 2: Datapoint $\mathbf{y}^{(i)}$ is quantized to lattice point $\mathbf{z}^{(i)}$ by Λ_{2^i}. In this step, the corresponding lattice quantizers of Λ_{2^i} are implemented.

$$\mathbf{z}^i = Q_{\Lambda_{2^i}}(\mathbf{y}^{(i)}), \text{ for } i \in \{0, 1 \dots (m-1)\}. \tag{3.4}$$

Next, we calculate the vector $b^{(i)}$, which is the integer representation of the lattice point $\mathbf{z}^{(i)}$.

$$\mathbf{b}^{(i)} = G^{-1}_{\Lambda_{2^i}} \mathbf{z}^{(i)}, \text{ for } i \in \{0, 1 \dots (m-1)\}. \tag{3.5}$$

After that, the vector $\mathbf{b}^i$ is indexed inside Voronoi region $V_{2\Lambda_{2^i}}(\mathbf{0})$ which is generated by magnifying current fundamental region two times. In other words, we indexed the coset representatives of quotient group $\Lambda_{2^{i+1}}/\Lambda_{2^i}$. For instance, in two-dimensional lattice A_2, codewords in one level includes $(0,0), (0,1), (1,0), (1,1)$. Particularly,

$$index^{(i)} = \mathbf{b}^{(i)} \bmod 2, \text{ for } i \in \{0, 1 \dots (m-1)\}. \tag{3.6}$$

Step 3: Finally, m n-bit $index_i$ binary sequences corresponding to m levels nested lattice $2^0, 2^1, \dots, 2^{m-1}$ are concatenated into a binary hash sequence:

$$LatticeHash = index^{(1)}, index^{(2)}, \dots, index^{(m)}. \tag{3.7}$$

In this research, we applied the nested indexing scheme to hexagonal lattice A_2 and E_8 lattice. Those lattices have generator matrixes as below:

$$G_{A_2} = \begin{bmatrix} 1 & 1/2 \\ 0 & \sqrt{3}/2 \end{bmatrix}, \text{and} \tag{3.8}$$

$$G_{E_8} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 \end{bmatrix}. \tag{3.9}$$

3.2 Metrics

This paper uses Hamming distance, weighted Hamming distance and first difference distance as metrics versus Euclidean distance to evaluate the normalized mean squared errors. Based on those errors, we can compare and choose the most efficient code and metric distance for image hashing.

3.2.1 Euclidean distance

Euclidean distance or Euclidean metric is the distance between two points in Euclidean space. It is fundamental and widely known as a good measure of the similarity between samples. In the case of comparison only, to reduce the computational complexity, squared Euclidean distance is used more frequently than standard Euclidean distance. Squared Euclidean distance is simply a squared version of Euclidean to remove the square root. For n-dimensional space, Euclidean distance and squared Euclidean distance from $\mathbf{x}$ to $\mathbf{y}$ are defined as:

$$\mathrm{d_E} = ||\mathbf{y} - \mathbf{x}|| = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}, \tag{3.10}$$

$$\mathrm{d_E^2} = ||\mathbf{y} - \mathbf{x}||^2 = \sum_{i=1}^{n}(y_i - x_i)^2. \tag{3.11}$$

3.2.2 Hamming distance

Hamming distance is the number of positions that must be changed to transform one sequence to another [27]. The Hamming distance $d_H(\mathbf{x}, \mathbf{y})$ between two n-dimensional vectors of equal length $\mathbf{x}$ and $\mathbf{y}$ is the number of positions where they differ:

$$d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} d_H(x_i, y_i), \tag{3.12}$$

where

$$d_H(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \neq y_i \end{cases}, \text{ for } i = 1, \ldots, n. \tag{3.13}$$

We are working with binary vectors, and the Hamming distance simplifies to the XOR of two vectors.

3.2.3 Weighted Hamming distance

The purpose of weighted Hamming distance is to preserve the distance between vectors. The higher the level of the nested lattice, the higher the weight the bits groups will be assigned. We propose a weighted Hamming distance measurement which assigns weights exponentially to every group of bits of sequences of n-dimensional and m levels:

$$d_{WH}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m \cdot n} w_i d_H(x_i, y_i), \tag{3.14}$$

where

$$w_i = 2^{\lfloor i/n \rfloor}, \text{ for } i = 1, \ldots, m \cdot n.$$

3.2.4 First difference distance

In this paper, we also propose the concept of first difference distance which reflects the similarity between bit sequences. Two bit sequences will be compared from the last element to the first element, the index of the first different element will be marked as first difference distance. In n-dimensional space, consider two binary sequences:

$$\begin{aligned} \mathbf{x} &= \{(x_1, \ldots, x_n)^{(1)}, \ldots, (x_1, \ldots, x_n)^{(n)}\}, \\ \mathbf{y} &= \{(y_1, \ldots, y_n)^{(1)}, \ldots, (y_1, \ldots, y_n)^{(n)}\}. \end{aligned} \tag{3.15}$$

First difference distance is the index of the first group of n bits that they are different. More formally,
For $i \in [0, n], i \in \mathbb{Z}$,

$$d_{FD}(\mathbf{x}, \mathbf{y}) = i \Leftrightarrow \begin{cases} (x_1, \ldots, x_n)^{(n-i)} \neq (y_1, \ldots, y_n)^{(n-i)} \\ (x_1, \ldots, x_n)^{(n-j)} = (y_1, \ldots, y_n)^{(n-j)} \end{cases} \tag{3.16}$$

$$\forall j \in [0, (i-1)], j \in \mathbb{Z}. \tag{3.17}$$

For later use of first difference distance in the lattice indexing scheme, the last group of bits and the first group of bits correspond to the highest level of lattice and the lowest level of lattice vectors. In terms of the idea, the first difference distance counts the number of similar lattice codeword from highest level to lowest level, then saves the index; therefore the higher value of first difference distance, the greater dissimilarity between two vectors.

3.3 Indexing scheme evaluation based on normalized mean squared error

In this section, we consider two input distribution cases and introduce normalized mean squared error (NMSE) as a robustness measurement.

For the hashing system input, we consider two distributions for the original feature vectors $\mathbf{x}$ and the modified vectors $\mathbf{x}'$. In both cases, original vectors $\mathbf{x}$ are uniformly distributed, but modified vectors $\mathbf{x}'$ are different.
Case (a): $\mathbf{x}$ and $\mathbf{x}'$ are both uniformly distributed. Then we apply lattice indexing scheme and Gray indexing scheme and compare their performance.
Case (b): $\mathbf{x}$ is uniform and $\mathbf{x}'$ is obtained by adding Gaussian noise to $\mathbf{x}$, as:

$\mathbf{x}' = \mathbf{x} + \mathbf{N}(0, \sigma^2)$. In this case, we vary the Gaussian variance, then analyze how the noise variance affects the indexing scheme's performance.

NMSE is used to compare the robustness of indexing schemes. After generating $\mathbf{x}$ and $\mathbf{x}'$ according to the two mentioned cases, input vectors are decoded to binary sequences. Then, the Euclidean distance d_E and other metric distances (d_H or, as in this paper, d_{WH} and d_{FD}) between every pair $(\mathbf{x}, \mathbf{x}')$ are computed. Recall the target is calculating d_E from some metric distances d_M, so we use least mean squared error technique to fit d_E and d_M by a linear predictor function $d'_E = \alpha d_M + \beta$, where α and β are coefficients with least mean squared error. Then, we define the NMSE between estimated d'_E and sample's d_E for n-dimensional space and N pairs of d'_E and d_E as:

$$NMSE = \frac{1}{n} MSE = \frac{1}{nN} \sum_{i=1}^{N} (d'_{Ei} - d_{Ei})^2. \tag{3.18}$$

NMSE is dimensionless. The smaller the NMSE, the better the linearity the indexing scheme can achieve and the more robust the indexing scheme is.

3.4 The best choice for shift vector a

There are infinite choices for the shift vector $\mathbf{a}$, this sub-section explains how to choose the optimized vector. Firstly, we introduce the concept of inefficiently indexed regions (IIR) which consist of distinct points in Euclidean space with zero Hamming distances between indexed codewords. A set consists of k regions $R_1, \ldots, R_k$ are IIR if and only if there exist pairs of $(\mathbf{x}, \mathbf{x}')$ such that:

$$\mathbf{x} \in R_i, \mathbf{x}' \in R_j, d_E(\mathbf{x}, \mathbf{x}') > 0, d_{WH}(\mathbf{x}, \mathbf{x}') = \mathbf{0}, \text{ for } i, j \in \{1, \ldots, k\}, i \neq j. \tag{3.19}$$

For instance, as shown in Figure 3.1, A and A' are relatively IIR together, similarly with B, B' and C, C'. All pairs which have one element from A (hash value 01), the other from A' (also hash value 01) are indexed to the same codeword with Hamming distance equal zero, but they have a large Euclidean distance. In short, IIRs increase the MSE.

Consider a two-dimensional lattice Λ and a Voronoi region $V_{2\Lambda}(\mathbf{a})$. According to the proposed rules in Section 3.1, all points in $V_{2\Lambda}(\mathbf{a})$ are indexed as shown in Figure 3.1 and Figure 3.2 where $\mathbf{a} = \mathbf{0}$ and $\mathbf{a} = 2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2})$ respectively. Comparing these two cases, while 75% of $V_{2\Lambda}(\mathbf{0})$ area is IIR, $V_{2\Lambda}(2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2}))$ has only 25%. The objective is to estimate the Euclidean distance between two points from Hamming distance between those two points with MSE as small as possible. We believe that minimizing MSE is equivalent to minimizing the area of IIRs by choosing vector a. As we can see, when moving vector $\mathbf{a}$, $V_{2\Lambda}(2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2}))$ is the best choice to achieve the minimum percentage of ineffective regions. In this research, for two-dimensional space, we used $V_{2\Lambda}(2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2}))$. The centroid of $V_{2\Lambda}(2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2}))$ is the lattice's deep hole [17], which is the point of the plane furthest from the lattice.

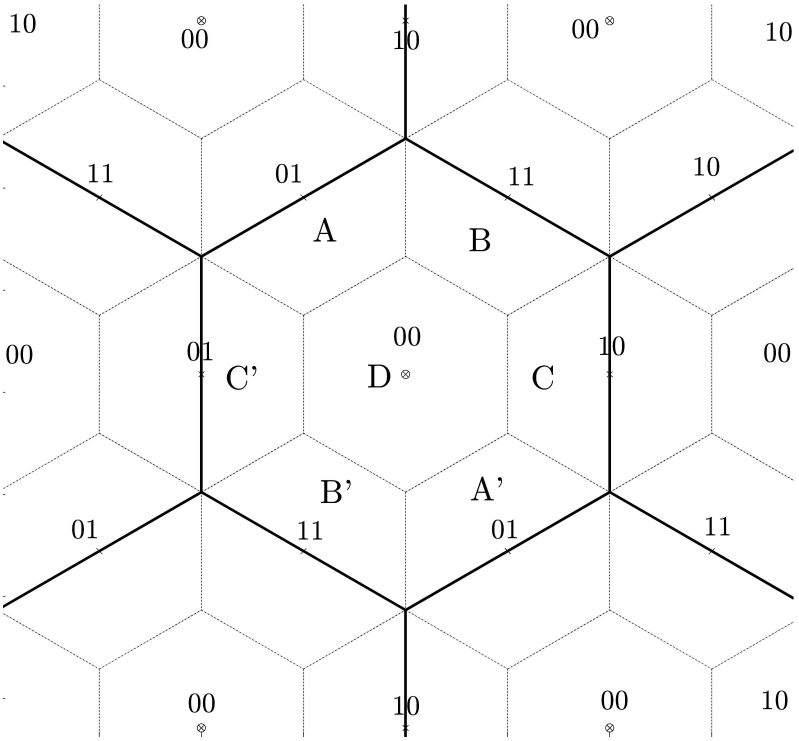

Figure 3.1: Voronoi region $V_{2\Lambda}(\mathbf{0})$.

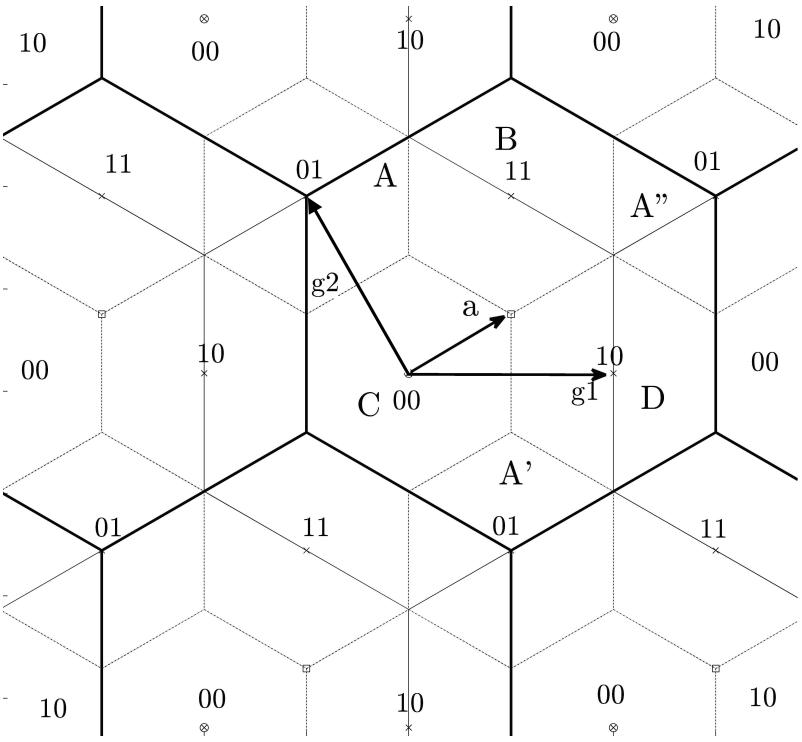

Figure 3.2: Voronoi region $V_{2\Lambda}(\mathbf{a})$, where $\mathbf{a} = 2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2})$.

Chapter 4

Experimental results

This chapter describes our four simulations using quantization coding schemes, feature extraction, and a complete image indexing database application.

In the two first simulations, we implemented two-dimensional nested lattice code A_2, eight-dimensional nested lattice E_8, and Gray code. In particular, nested A_2 used coset vector $\mathbf{a} = 2/3(\mathbf{g_1})+1/3(\mathbf{g_2})$, as explained in Section 3.4. In order to compare to Gray code, we assumed two cases of input data: Case (a) has both original vectors and modified vectors are uniformly distributed; Case (b) has uniformly distributed original vectors and additional Gaussian noise vectors as modified vectors. In addition, we used weighted Hamming distance and first difference distance as new metric distances versus traditional Hamming distance. So far, the results pointed out that the combination of two-dimensional nested lattice A_2 with coset vector $\mathbf{a}$ and first difference distance is better than eight-dimensional lattice, and reduces the normalized mean square error (NMSE) by 20% compared to two-dimensional Gray code.

In terms of feature extraction, we used SURF to detect interesting points from grayscale images, then obtain feature vectors. In order to indicate the robustness of SURF, we applied SURF to both original images and distorted images, such as rotation, JPEG compression, noise attack. The simulation shows that SURF interesting points and feature vectors are perceptually stable under attacks. Key points were mainly located at strong texture areas, edges. In other words, SURF extracts feature vectors based on the content of the images. This research does not improve SURF, we only show its efficiency under distortion, and then take the advantage by applying SURF to our hashing system.

The last simulation is constructing an image indexing database. This database application used a complete nested lattice image hashing function. As we mentioned above, we took the advantage of SURF as a feature extraction, then apply two-dimensional nested lattice A_2 as a quantizer. This is an application of a particular content-based image hashing function, and we firmly believe that nested lattices can become a coding-based scheme for a class of similarity search application.

4.1 Two-dimensional nested lattice indexing schemes A_2

A two-dimensional nested lattice and a Gray code indexing scheme were implemented and simulations were ran with the two input distribution cases (Case (a) and Case (b)) which are described in Section 3.3. For fair comparison, we used: dimension $n = 2$, fundamental volume equal to one ($V(\Lambda) = 1$), level $m = 7$ means 14 bits per hash value, 10^4 two-dimensional real input data points (or vectors) $\mathbf{x} \in [0, 2^{m-1}]^2$ uniformly distributed in Case (a) and 10^4 additional two-dimensional Gaussian noise vectors $\mathbf{N}(0, \sigma^2)$ in Case (b). Particularly, these two indexing schemes are based on two corresponding quantizers, therefore quantization error depends on the lattice (and is slightly better for the hexagonal lattice). The density of points relates the possible number of hash values to the quantization error, and that is why we fairly compared lattice and Gray quantizer with fundamental volume equal to one.

When the input vectors $\mathbf{x}$ and $\mathbf{x}'$ are uniformly distributed, as in Case (a) in Section 3.3, the (d_E, d_H) for every sample pairs in the dataset, along with its linear predictor function for Gray indexing are shown in Figure 4.1. Similarly, the (d_E, d_{WH}) and (d_E,d_{FD}) for every possible pairs, along with their linear predictor functions for nested lattice indexing are shown in Figure 4.2 and Figure 4.3, respectively. Table 4.1 depicts the NMSE and other details about this comparison simulation. The smaller the NMSE, the better the linearity the indexing scheme can achieve.

Table 4.1: Case (a): Nested A_2 lattice and Gray indexing scheme simulation information.

Indexing scheme	**Metric distance**	**NMSE**
Gray code	d_E vs. d_H	113.7115
Lattice code A_2	d_E vs. d_{WH}	106.8708
Lattice code A_2	d_E vs. d_{FD}	91.7450

When the input $\mathbf{x}$ is uniformly distributed and $\mathbf{x}'$ is Gaussian distributed, as in Case (b) in Section 3.3, we adjust noise intensity by changing Gaussian variance σ^2. The (d_E, d_H), (d_E,d_{FD}) between original vectors $\mathbf{x}$ and noisy vectors $\mathbf{x}'$ are computed and fitted by linear predictor functions. Figure 4.4 represents the variation of NMSE values of linear predictor functions as a function of noise variance σ^2 for two-dimensional Gray indexing and lattice indexing.

We observe that the nested lattice indexing generally has better performance than Gray code indexing. In Case (a), the combination of nested lattice indexing and first difference distance reduces approximately 20% NMSE compared to Gray indexing. In Case (b), on average, nested lattice indexing has smaller NMSE than Gray indexing.

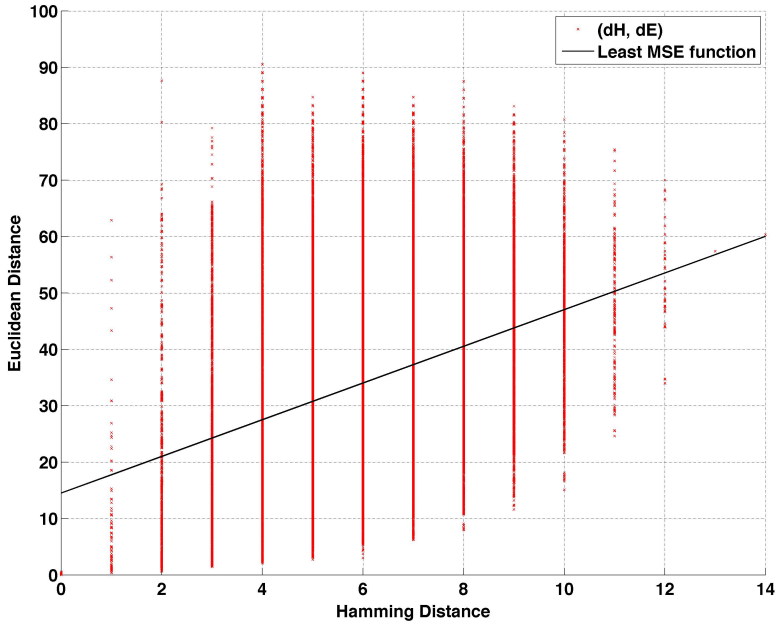

Figure 4.1: Case (a): The Hamming distance versus the Euclidean distances of Gray indexing.

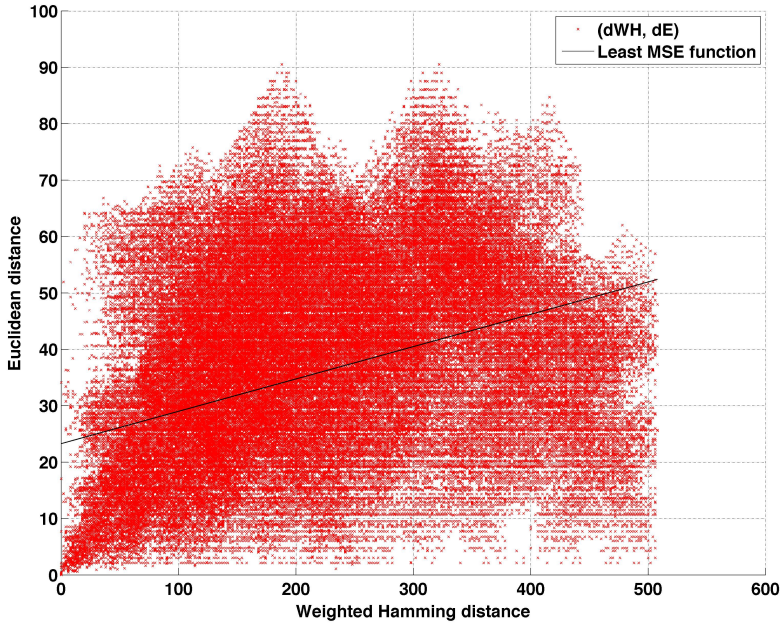

Figure 4.2: Case (a): The weighted Hamming distance versus the Euclidean distances of nested A_2 lattice indexing.

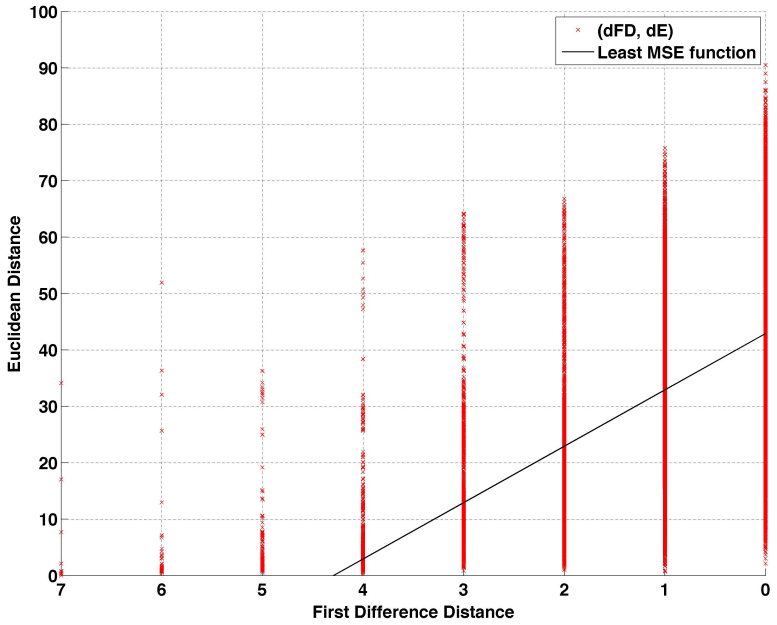

Figure 4.3: Case (a): The first difference distance versus the Euclidean distances of nested A_2 lattice indexing.

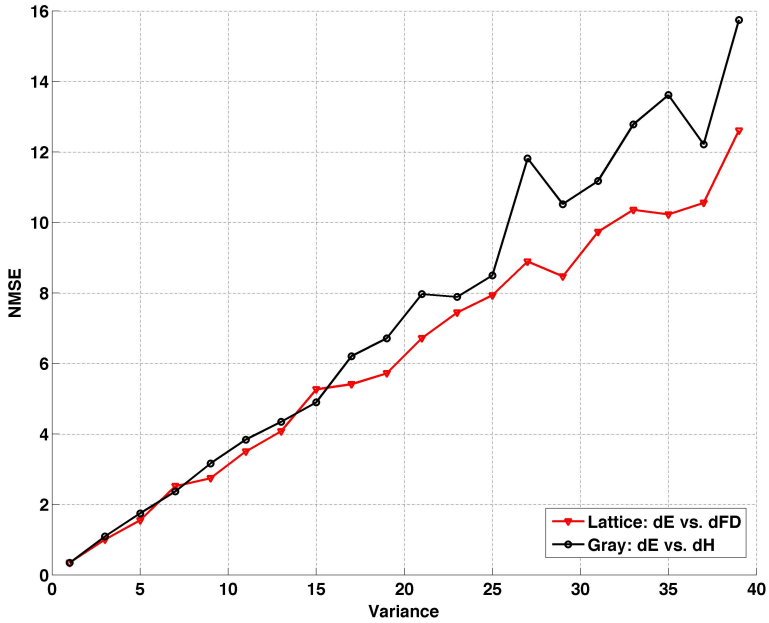

Figure 4.4: Case (b): The variation of A_2's NMSE as a function of noise variance.

4.2 Nested lattice E_8 indexing schemes simulation

This section shows the simulation of eight-dimensional nested lattice E_8, called $SIM2$. The process similar to two-dimensional nested lattice A_2, called $SIM1$. Consistently, we still experimented two case studies and using NMSE as the evaluation index, explained in Section 3.3. Two differences between $SIM1$ and $SIM2$: First, whereas $SIM1$ input was two-dimensional vectors, $SIM2$ input was eight-dimensional vectors. However, the volume of lattice's fundamental regions are both equal one, it guarantees the fair comparison. Second, $SIM1$ used nested A_2 indexing scheme, while $SIM2$ used nested E_8 indexing scheme. Since NMSE is dimensionless, which is normalized by dimension, then the comparison is fair.

Figure 4.5, 4.6, 4.7, and table 4.2 show the visualization and numerical results. In conclusion, the combination of nested A_2 and first difference distance still is the best choice for quantization stage.

Table 4.2: Case (a): Nested E_8 lattice indexing schemes results.

Indexing scheme	**Metric distance**	**NMSE**
Lattice code E_8	d_E vs. d_{WH}	105.1206
Lattice code E_8	d_E vs. d_{FD}	110.4302

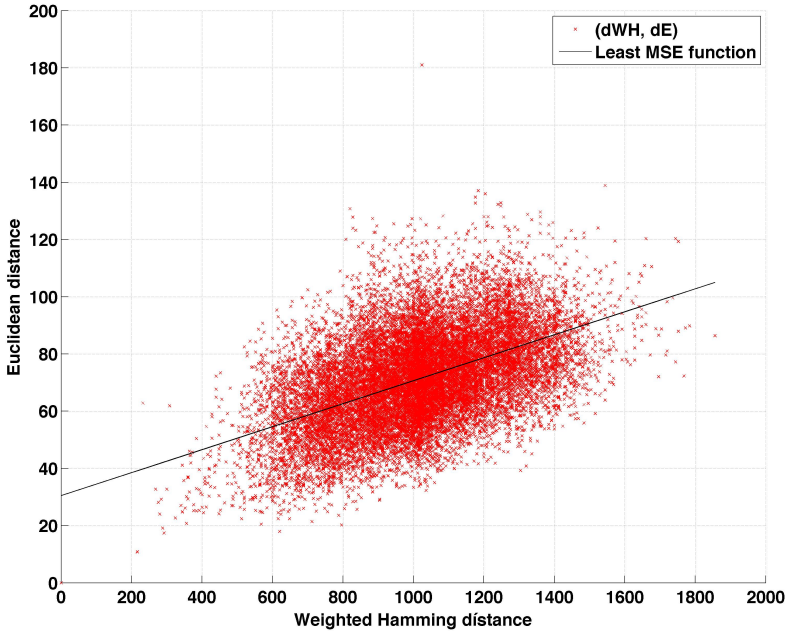

Figure 4.5: Case (a): The weighted Hamming distance versus the Euclidean distances of nested E_8 lattice indexing.

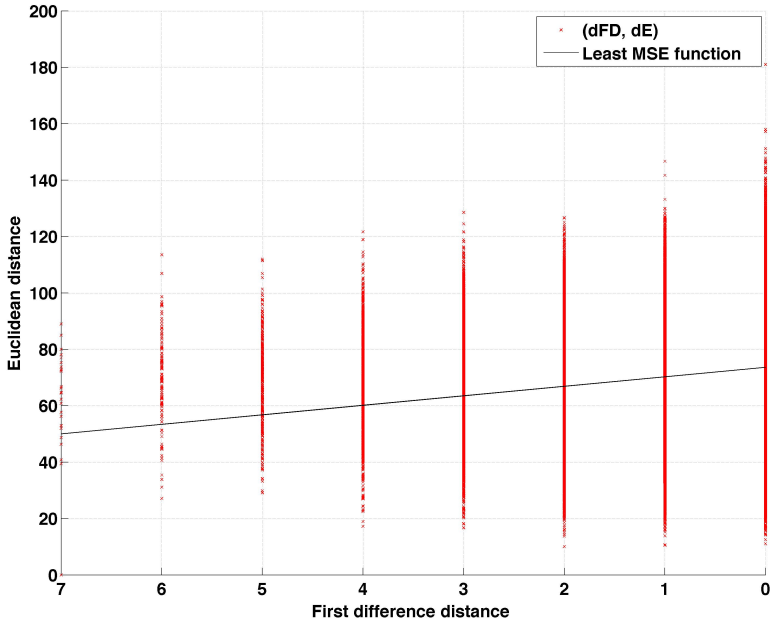

Figure 4.6: Case (a): The first difference distance versus the Euclidean distances of nested E_8 lattice indexing.

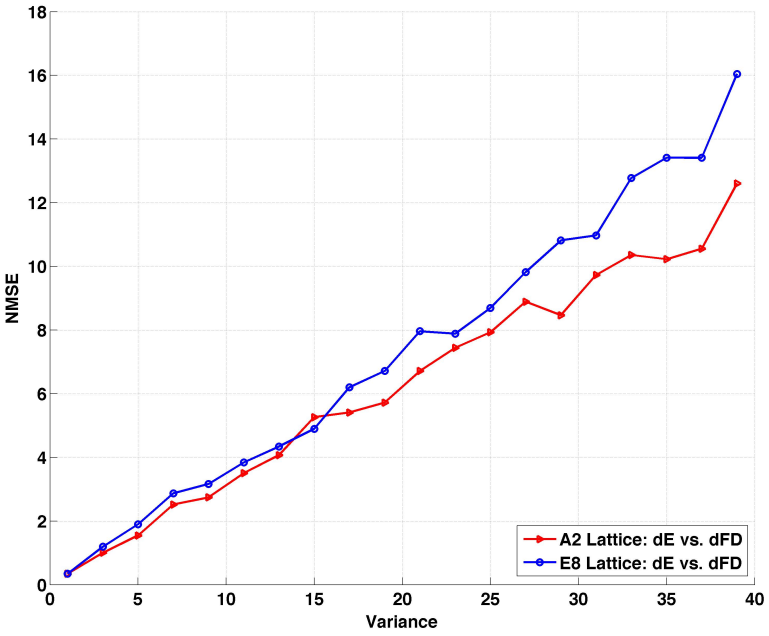

Figure 4.7: Case (b): The variation of E_8's NMSE as a function of noise variance.

4.3 Feature extraction results using SURF

As discussed previously, we used SURF in the feature extraction step of our hash function. This section shows the robustness of SURF under some attacks, such as rotation, noise, and JPEG compression.

For visualization, we used a standard test picture named *cameraman*, grayscale, resolution 512×512. Figure 4.8 depicts the original cameraman image with the green circles represent detected interesting points. Figure 4.9, 4.10, 4.11 indicate distorted cameraman images and their identified interesting points respectively. As we can see, over all the images, the greens points, which visualize interesting points are almost similar, and locate on edges and strong texture areas. That shows the robustness of SURF under image distortions; it means perceptually similar images have similar SURF feature vectors. Our image hash function takes advantage of SURF to tolerate small image modifications and attacks.

Figure 4.8: Surf features of original grayscale cameraman image.

Figure 4.9: Surf features of compressed cameraman: JPEG 5%.

Figure 4.10: Surf features of rotated cameraman image: 30°.

Figure 4.11: Surf features of noised cameraman image.

4.4 Simulation of Image database

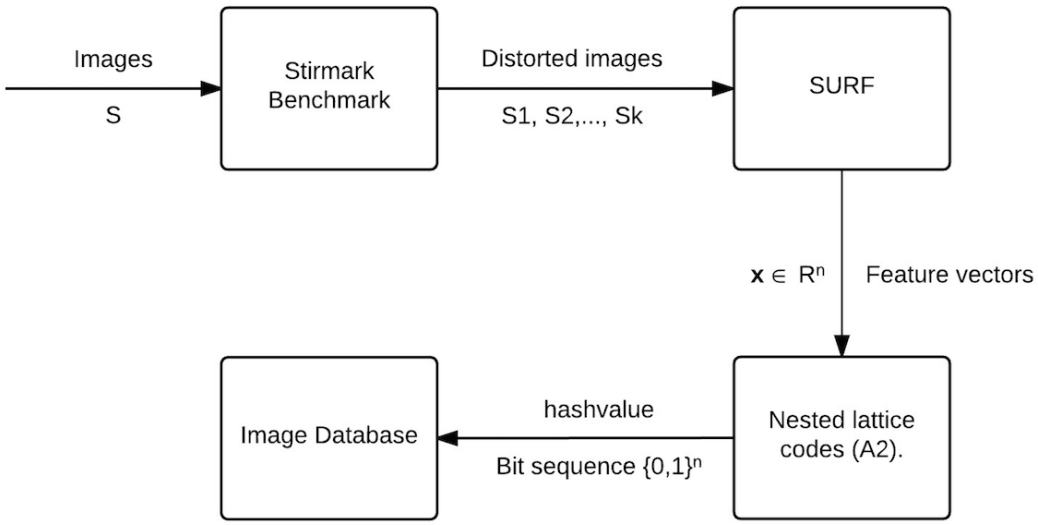

Figure 4.12: Image database simulation process.

We designed a simple image database system to apply proposed content-based image hash function. This simulation consists of four main steps, as shown in Figure 4.12: prepare input images dataset; apply SURF feature vector extraction; apply

nested lattice A_2 code; store images with their hash sequences and do some queries. This experiment used following tools:

- Matlab R2014a tools and Matlab programming language.
- Postgresql 9.3 [31]: An open source database.
- Postgresql JDBC driver [32]: Link between Postgresql and Matlab.
- Stirmark Benchmark 4.0 [33]: A tool which is used to distort image database.

In terms of database structure design, our database simply has only one table named *images* to store all necessary information. Table 4.4 describes the meaning of each column in our database.

In terms of the image dataset, we firstly choose some standard test images, such as cameraman, lena, and mandrill. Afterward, we use Stirmark Benchmark tool to attack every input image, then collect distorted images. Figure 4.13 shows some images under Stirmark Benchmark attacks. The attacks detail are described in Table 4.3. In the end, we have a dataset of 520 distorted images which is generated from ten original images.

Next, we process distorted image dataset with the content-base image hash function using nested lattice code. As shown in Figure 4.12, our hash function uses SURF extraction in the first stage to select 50 64-dimensional feature vectors from each image. Then two-dimensional A_2 nested lattice quantize each $64-$dimensional SURF vectors to 320-bit sequence, the final hash value has length of 16000 bit, or 2 kB. Images and their hash values are stored in Postgresql database.

In order to show the advantage of this database system, we run some queries. In detail, suppose that we have an image S, and then we want to search the most similar image in the database. **Input:** An image S; **Ouput:** A list of the most similar images in database. This is a search process: Firstly, image S is hashed into hash sequence H, then compare H with hash values of existing images in the database which is column *hash_a2*. In terms of comparing first difference distance, a list of similar hash values with input H is selected. The result is a list of images corresponds to the list of selected hash values. Some queries input and results are plotted, as shown in Figure 4.14.

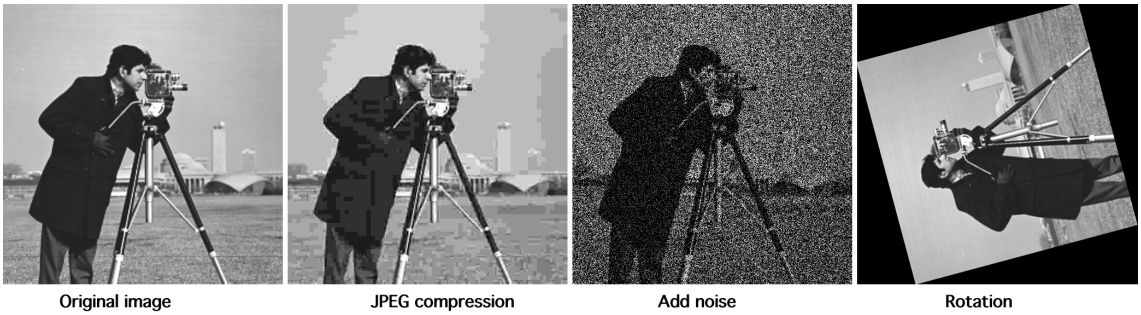

Figure 4.13: Cameraman image under some Stirmark Benchmark attacks.

Table 4.3: Stirmark Benchmark attacks detail.

Attack	**Attack description**	**Number of distorted images**
Rotation	Rotate from 0 to 180 degree	12
JPEG compression	Compression ratio 5% to 100%	20
Noise	Noise from 0% to 100%	20

Table 4.4: Database structure, table design.

Column	**Data type**	**Description**
id	serial	The index of the record, we used this column as primary key. Type *serial* mean auto increment integer.
name	text	The name of the image
path	text	The path to image on disk. We do not save the whole image blob of bits in the database, we save them on disk and their path. It makes the database lightweight.
hash_a2	text	The hash value, which is result of nested lattice A_2 quantizer.

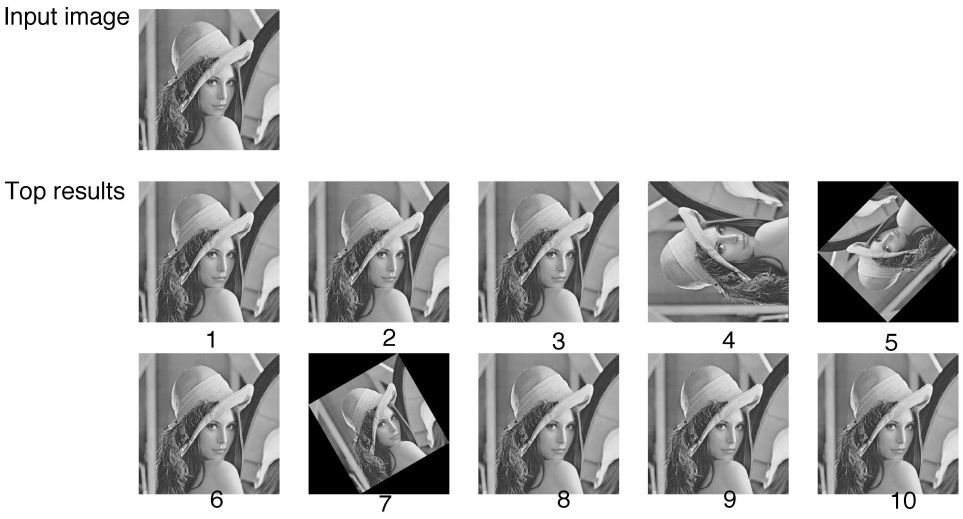

Figure 4.14: Example queries results.

Chapter 5

Conclusion

In summary, the main contributions of this research include:

- Efficient metric distances: weighted Hamming distance and first difference distance.
- Nested lattice indexing scheme with coset vector **a**.
- Build a complete content-based hash function for image indexing system using SURF as feature extraction and nested lattice as a quantizer.

5.1 New metric distances

We proposed the weighted Hamming distance and the first difference distance as new metrics versus Euclidean distance. The purpose of both weighted Hamming distance or first difference distance is to preserve the distance between vectors. Weighted Hamming distance assigns weights exponentially to every group of bits of sequences. The higher the level of nested lattice, the higher the weight the bits groups will be assigned. In terms of first difference distance, this metric counts the number of similar lattice codeword from highest level to lowest level, then saves the index; therefore the higher value of first difference distance, the greater dissimilarity between two vectors. The experiment result shows that our proposed metrics are better than traditional Hamming distance in terms of reflecting the similarity between vectors.

5.2 Nested lattice indexing scheme

A lattice code replaced the Gray code, a nested lattice indexing scheme is proposed, and multi-dimensional nested lattices experiment. As discussed, Gray code is much better than natural binary code and widely used in discrete-to-binary conversion

stage of current image hashing. This research used Gray code as a base coding scheme to compare with our proposed nested lattice code schemes. Precisely, we examined the combination of proposed metrics distance (includes weighted Hamming distance and first difference distance) and nested lattice code versus the existing technique of Gray code and Hamming distance.

The nested lattice indexing scheme takes advantage of lattices for quantizing feature vectors to hash values. In the quantization step, nested lattices tend to keep a proportional relationship between Euclidean distance and mentioned metric distances (Hamming distance or, as in this paper, weighted Hamming distance and first difference distance) to increase the hash function's robustness. To compare among indexing schemes, we compared their normalized mean square error. As experimental results so far, the combination of two-dimensional nested lattice and first difference distance reduces the normalized mean square error (NMSE) by 20% compared to two-dimensional Gray code.

In addition, coset lattices are considered. There are infinite choices for the shift vector or coset vector $\mathbf{a}$, we explained how to choose the optimized $\mathbf{a}$ in Section 3.4. So far, we only have the best choice of $\mathbf{a} = 2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2})$ for hexagonal lattice A_2 since higher dimensional lattices are difficult to visualize. For future work, we plan to look for the optimized coset vectors for higher dimensional lattices.

In short, developing an efficient indexing scheme has a promising class of similarity search applications which usually deal with comparing feature vectors. Notably, this research focuses on image hashing systems by applying proposed nested lattice codes.

5.3 Content-based image hash function

We constructed a complete image hash function and experimented with multimedia database indexing. Our image hash function used SURF to extract feature vectors from images; then, using nested lattice code for quantization step. This image hash function takes the advantages of SURF, which is a robust content-based feature extraction against distortions such as rotation, scaling or compression. On the other hand, the hash function also takes the advantages of proposed nested lattice which reflects the similarity between feature vectors by the similarity between codewords.

In our multimedia database simulation, images are stored with their corresponding hash sequences. The hash sequences (or hash values) are generated by applying proposed hash function (the combination of SURF and two-dimensional nested lattice code), so those hash sequences contain the SURF features which represent the image's characteristics. For retrieving image from this database system, instead of image comparing sample-by-sample, we can compare their hash sequences using first difference distance metric, then infer the similarity between images.

5.4 Future work

As future work, we intend to investigate on higher dimensional lattices with the expectation of a better relationship between metric distance and Euclidean distance. We plan to optimize the coset vector $\mathbf{a}$ for higher dimensional lattices even though they are difficult to visualize, especially for E_8. On the other hand, multimedia database indexing is just a simple application of nested lattice code, we would like to pursue the class of similarity applications. Since many promising applications in machine learning, information retrieval, and other areas have to work with feature vector, so we believe that nested lattice indexing scheme become a based coding scheme in quantization stage.

This thesis was prepared according to the curriculum for the Collaborative Education Program organized by Japan Advanced Institute of Science and Technology and University of Engineering and Technology, Vietnam National University, Hanoi.

References

[1] Faloutsos, Christos, and Douglas W. Oard. "A survey of information retrieval and filtering methods." (1998).

[2] Faloutsos, Christos, et al. "Efficient and effective querying by image content." Journal of intelligent information systems 3.3-4 (1994): 231-262.

[3] Cost, Scott, and Steven Salzberg. "A weighted nearest neighbor algorithm for learning with symbolic features." Machine learning 10.1 (1993): 57-78.

[4] Hastie, Trevor, and Rolbert Tibshirani. "Discriminant adaptive nearest neighbor classification." Pattern Analysis and Machine Intelligence, IEEE Transactions on 18.6 (1996): 607-616.

[5] Menezes, Alfred J., Paul C. Van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. CRC press, 1996.

[6] Preneel, Bart. "Cryptographic hash functions." European Transactions on Telecommunications 5.4 (1994): 431-448.

[7] Bellare, Mihir, Ran Canetti, and Hugo Krawczyk. "Keying hash functions for message authentication." Advances in Cryptology?CRYPTO?96. Springer Berlin Heidelberg, 1996.

[8] Huang, Chung-Lin, and Dai-Hwa Huang. "A content-based image retrieval system." Image and Vision Computing 16.3 (1998): 149-163.

[9] Gudivada, Venkat N., and Vijay V. Raghavan. "Content based image retrieval systems." Computer 28.9 (1995): 18-22.

[10] Abdel-Mottaleb, Mohammed S., and Santhana Krishnamachari. "Image retrieval system." U.S. Patent No. 6,754,675. 22 Jun. 2004.

[11] Andrews, Harry C., and Claude L. Patterson III. "Singular value decomposition (SVD) image coding." Communications, IEEE Transactions on 24.4 (1976): 425-432.

[12] Bay, Herbert, et al. "Speeded-up robust features (SURF)." Computer vision and image understanding 110.3 (2008): 346-359.

[13] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[14] Ozaktas, Haldun M., Zeev Zalevsky, and M. Alper Kutay. The fractional Fourier transform. Wiley, Chichester, 2001.

[15] Agrawal, Rakesh, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. Springer Berlin Heidelberg, 1993.

[16] Venkatesan, Ramarathnam, et al. "Robust image hashing."Image Processing, 2000. Proceedings. 2000 International Conference. Vol. 3. IEEE, 2000.

[17] J. H. Conway and N. J. A. Sloane, Sphere Packings, Lattices and Groups, New York, NY, Springer-Verlag, 3rd ed., 1999.

[18] Gray, Robert M. "Vector quantization." ASSP Magazine, IEEE 1.2 (1984): 4-29.

[19] Ricardo Antonio Parrao Hernandez, Mariko Nakano Miyatake, and Brian M. Kurkoski. "Robust image hashing using image normalization and SVD decomposition." MWSCAS, 2011 IEEE 54th Midwest Symposium on. IEEE, 2011.

[20] Faloutsos, Christos. "Gray codes for partial match and range queries." Software Engineering, IEEE Transactions on 14.10 (1988): 1381-1393.

[21] Swaminathan, Ashwin, Yinian Mao, and Min Wu. "Robust and secure image hashing." Information Forensics and Security, IEEE Transactions on 1.2 (2006): 215-230.

[22] Zhu Guopu, et al. "Fragility analysis of adaptive quantization-based image hashing." Information Forensics and Security, IEEE Transactions on 5.1 (2010): 133-147.

[23] Li, Yuenan, et al. "Robust image hashing based on random Gabor filtering and dithered lattice vector quantization." Image Processing, IEEE Transactions, 2012.

[24] John H. Conway, N. J. A. Sloane, "A Fast Encoding Method for Lattice Codes and Quantizers", IEEE *Trans. Inf. Theory* IT-29, 820-824, 1983.

[25] Conway, J. H., and N. J. A. Sloane. "On the Voronoi regions of certain lattices." SIAM Journal on Algebraic Discrete Methods 5.3 (1984): 294-305.

[26] Gray, Frank. "Pulse code communication." U.S. Patent. 17 Mar. 1953.

[27] Hamming, Richard W. "Error detecting and error correcting codes." Bell System technical journal 29.2 (1950): 147-160.

[28] Zamir, Ram. Lattice Coding for Signals and Networks: A Structured Coding Approach to Quantization, Modulation, and Multiuser Information Theory. Cambridge University Press, 2014.

[29] Beaudet, Paul R. "Rotationally invariant image operators." International Joint Conference on Pattern Recognition. Vol. 579. 1978.

[30] Convay, J. H., and N. J. A. Sloane. "Fast quantizing and decoding algorithms for lattice quantizers." IEEE Trans Inform Theory 28.2 (1982): 227-232.

[31] http://www.postgresql.org/.

[32] https://jdbc.postgresql.org/.

[33] http://www.petitcolas.net/watermarking/stirmark/.

Publications

Major research

[1] **Thanh Xuan Nguyen**, Ricardo Antonio Parrao Hernandez, Brian Michael Kurkoski, "Robust Content- Based Image Hash Functions Using Nested Lattice Codes", in Proc. of International Workshop on Digital-forensics and Watermarking (IWDW), Japan, Oct. 2015.

[2] **Thanh Xuan Nguyen**, Ricardo Antonio Parrao Hernandez, Brian Michael Kurkoski, "Nested Lattice Hashing Scheme for Similarity Search Applications", in Proc. of IEICE 2015, pp. 19-24, Error Correcting Codes (ECC), Japan, Sep. 2015.

[3] **Thanh Xuan Nguyen**, Ricardo Antonio Parrao Hernandez, Brian M. Kurkoski, "Image hashing using lattice codes", Workshop on Lattice Coding and Start Code, University of Electro-Communications (UEC), Japan, Jun. 2015.

Minor research

[4] Do, K. P., Nguyen, B. T., **Nguyen, X. T.**, Bui, Q. H., Tran, N. L., Nguyen, T. N. T., Vuong, V. Q., Nguyen, H. L. and Le, T. H., 2015. Spatial Interpolation and Assimilation Methods for Satellite and Ground Meteorological Data in Vietnam. Journal of Information Processing Systems. 11, 4, 556–572.

[5] **Nguyen*, X. T.**, Nguyen*, B. T., Do*, K. P., Bui*, Q. H., Nguyen*, T. N. T., Vuong**, V. Q. and Le*, T. H., 2015. "Spatial Interpolation of Meteorological Variables in Vietnam using the Kriging Method", Journal of Information Processing Systems. 11, 1, 134–147.