Robust Content-Based Image Hash Functions Using Nested Lattice Codes

Thanh Xuan Nguyen(✉), Ricardo A. Parrao Hernandez, and Brian M. Kurkoski

Japan Advanced Institute of Science and Technology, Nomi, Japan
{thanhnx,ricardo.parrao,kurkoski}@jaist.ac.jp

Abstract. This contribution improves content-based hash functions for image retrieval systems using nested lattice codes. Lattice codes are used to quantize image feature vectors to final hash values. The goal is to develop a nested lattice indexing scheme such that there is a proportional relationship between Euclidean distance and some metric distances (Hamming distance or, as in this paper, weighted Hamming distance and first difference distance) in order to increase the hash function's robustness. The proposed two-dimensional nested lattice code reduces the normalized mean squared error (NMSE) by 20 % compared to two-dimensional Gray code.

Keywords: Nested lattice codes · Nested lattice indexing · Image hash functions · Content-based

1 Introduction

Today, digital images 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 softwares. Therefore, image hash functions are used as an image authentication technique to protect data from distortion attacks that steal or alter data illegally. Cryptographic and content-based hash functions are two major data hashing techniques. Traditionally, data integrity issues are addressed by cryptographic hash functions, which are key-dependent and bit sensitive. This technique is usually applied to text message and file authentication, which requires all message bits to be unchanged [1]. In contrast, a content-based hash functions generates the hash value from the image features. This method is more suitable for multimedia files, which should be able to tolerate some minor modifications. In addition, the content-based approach can be applied to image retrieval systems [2]. 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.

B.M. Kurkoski—This work was supported by JSPS Kakenhi Grant Number 26289119.

Y.-Q. Shi et al. (Eds.): IWDW 2015, LNCS 9569, pp. 406–417, 2016.
DOI: 10.1007/978-3-319-31960-5_33

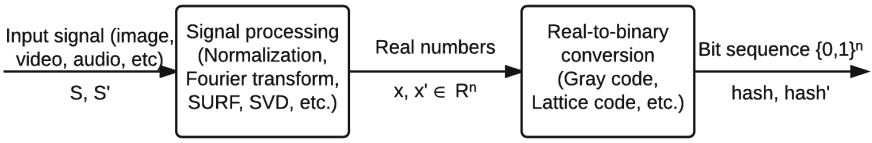

Fig. 1. A framework for hashing system.

A framework for hashing system is represented in Fig. 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 vector decomposition (SVD), speeded up robust features (SURF), Fourier transform 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 paper, 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.

Euclidean distance is widely known as a good measure of the similarity between features $\mathbf{x}$ and $\mathbf{x}'$ [3]. 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 [4]. 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 be also 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 specific signal processing schemes.

In fact, many hashing schemes have been proposed, but finding a proportional relationship remains a challenge. In 2000, Venkatesan and Ramarathnam [4] 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. [5] used image normalization and SVD as the first signal processing stage, then apply Gray code to obtain the binary hash sequence in image hash functions. According to his paper, the robustness of the hash functions was increased against rotation, scaling and JPEG attacks. Faloutsos in 1988 [6], Zhu et al. in 2010 [7] also used Gray code as the discrete-binary conversion stage of image hashing to improve clustering of similar records. In 2012, Yuenan et al. proposed hash functions based on random Gabor filtering and dithered lattice vector quantization (LVQ). A four-dimensional lattice is used for quantization, but codewords are normalized by a Gray code at the end. Basically, their approach can be considered as using Gabor filtering and a combination of a Gray code and a lattice.

In this paper, the Gray code is replaced by a lattice code. We also propose weighted Hamming distance and first difference distance as new metric distances. A lattice is a code over an n-dimensional real space and it has several advantages compared to Gray code. While Gray code requires a scalar quantizer, lattices employ vector quantization. It is well-known that vector quantizers have lower quantization error than scalar quantizers [9,10], 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 mean squared error of linear predictor function from metric distance to Euclidean distance among images using lattice codes.

The outline of the remainder of this paper is as follows. Section 2 gives background of lattice codes, Gray codes, Hamming distance, weighted Hamming distance, first difference distance and Euclidean distance metrics. Section 3 gives the proposed lattice coding method, evaluation method and the best choice of nested lattice code. Section 4 shows simulation results and performance comparison of Gray code and nested lattice code. Section 5 is the conclusion and future work.

2 Background

2.1 Lattices and Nested Lattice Codes

A lattice Λ is a linear additive subgroup of $\mathbb{R}^n$. Lattices form effective structures for various geometric, coding and quantization problems. Some well-known lattices are A_2, D_4, E_8 [9]. In n dimensions, a lattice point $\mathbf{x} \in \Lambda$ is an integral, linear combination of the basis vectors:

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

where $\mathbf{b} \in \mathbb{Z}^n$ is a vector of integers, $G = \left[\mathbf{g_1}\ \mathbf{g_2}\ \ldots\ \mathbf{g_n}\right]$ 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 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.

2.2 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,

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

where $||\cdot||^2$ denotes squared Euclidean distance. And, we define quantization error vector $\mathbf{e}$:

$$\mathbf{e} = \mathbf{y} - \mathbf{x}^*. \tag{3}$$

2.3 Voronoi Region

The Voronoi region [12] $V(\mathbf{x})$ 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:

$$V(\mathbf{x}) = \{\mathbf{z} \in \mathbb{R}^n : ||\mathbf{z} - \mathbf{x}||^2 < ||\mathbf{z} - \mathbf{y}||^2, \text{ for all } \mathbf{y} \in \Lambda, \mathbf{y} \neq \mathbf{x}\}. \tag{4}$$

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 $V_{r\Lambda}(\mathbf{a})$:

$$C_{r\Lambda}(\mathbf{a}) = \Lambda \cap V_{r\Lambda}(\mathbf{a}). \tag{5}$$

Figure 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 $\mathbf{a}$.

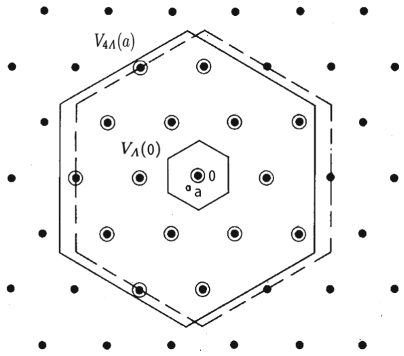

Fig. 2. Voronoi region $V_{4\Lambda}(\mathbf{a})$.

2.4 Gray Code and Gray Indexing

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 [13]. Gray codes are also known as reflected binary code. This section describes a recursive construction which encode binary sequences to Gray codes. Figure 3 depicts the algorithm to recursively generate Gray codes from binary sequences for two next levels from level one.

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

$$(y_1, \ldots, y_n) = Q_{Integer}(x_1, \ldots, x_n). \tag{6}$$

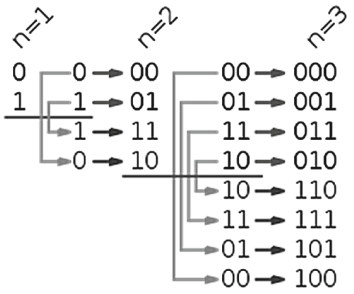

Fig. 3. The steps of the reflect and prefix method to generate Gray sequences.

Then, integer vector $(y_1, \ldots, y_n)$ is translated into binary sequences $(b_1, \ldots, b_n)$, then level m Gray code encode $(b_1, \ldots, b_n)$ to binary Gray sequence $(c_1, \ldots, c_n)$. Finally, those n sequences of Gray code were concatenated to an unique binary hash sequence.

$$GrayHash = c_1, \ldots, c_n. \tag{7}$$

2.5 Metrics

This paper uses Hamming distance, weighted Hamming distance, first difference distance and Euclidean distance as metrics to evaluate the normalized mean squared error.

Hamming distance is the number of positions that must be changed to transform one string to another [14]. The Hamming distance $d_H(\mathbf{x}, \mathbf{y})$ between two n-dimensional bit vectors $\mathbf{x}$ and $\mathbf{y}$ is the number of positions where they differ:

$$d_H(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} d_H(x_i, y_i), \tag{8}$$

where

$$d_H(x_i, y_i) = \begin{cases} 0 & \text{if } x_i = y_i \\ 1 & \text{if } x_i \neq y_i \end{cases}, \text{ for } i = 1, \ldots, n. \tag{9}$$

We propose a weighted Hamming distance measure which assigns weights exponentially to every group of bits of sequences of n-dimensional and m levels:

$$d_{WH}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{m \cdot n} w_i d_H(x_i, y_i), \tag{10}$$

where

$$w_i = 2^{\lfloor i/n \rfloor}, \text{ for } i = 1, \ldots, m \cdot n. \tag{11}$$

The purpose of weighted Hamming distance is explained in Sect. 3.

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: $\mathbf{x} = \{(x_1, \ldots, x_n)^{(1)}, \ldots, (x_1, \ldots, x_n)^{(n)}\}$ and $\mathbf{y} = \{(y_1, \ldots, y_n)^{(1)}, \ldots, (y_1, \ldots, y_n)^{(n)}\}$, first difference distance is the index of the first group of n bits, that they are different. More formally,
For $i \in [0, n], i \in \mathbb{Z}$,

$$d_{FD}(\mathbf{x}, \mathbf{y}) = i \Leftrightarrow \begin{cases} (x_1, \ldots, x_n)^{(n-i)} \neq (y_1, \ldots, y_n)^{(n-i)} \\ (x_1, \ldots, x_n)^{(n-j)} = (y_1, \ldots, y_n)^{(n-j)} \end{cases} \tag{12}$$

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

Euclidean distance is the distance between two points in Euclidean space. For n-dimensional space, Euclidean distance from $\mathbf{x}$ to $\mathbf{y}$ is defined as:

$$\mathrm{d_E} = ||\mathbf{y} - \mathbf{x}|| = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}. \tag{13}$$

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 a lattice Λ_{2^i} by a n-by-n generator matrix $G_{\Lambda_{2^i}}$:

$$G_{\Lambda_{2^i}} = 2^i G_\Lambda = 2^i \left[\mathbf{g_1}\ \mathbf{g_2}\ \ldots\ \mathbf{g_n}\right], \text{ for } i \in \{0, 1 \ldots (m-1)\}. \tag{14}$$

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

$$\mathbf{a} = [a_1, \ldots, a_n]. \tag{15}$$

The best choice of shift vector $\mathbf{a}$ is explained in detail in Sect. 3.3.

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,

$$\mathbf{y}^{(i)} = \mathbf{x} - 2^i\mathbf{a}, \text{ for } i \in \{0, 1 \ldots (m-1)\}. \tag{16}$$

Step 2: Datapoint $\mathbf{y}^{(i)}$ is quantized to lattice point $\mathbf{z}^{(i)}$ by Λ_{2^i}:

$$\mathbf{z}^i = Q_{\Lambda_{2^i}}(\mathbf{y}^{(i)}), \text{ for } i \in \{0, 1 \ldots (m-1)\}. \tag{17}$$

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

$$\mathbf{b}^{(i)} = G^{-1}_{\Lambda_{2^i}}\mathbf{z}^{(i)}, \text{ for } i \in \{0, 1 \ldots (m-1)\}. \tag{18}$$

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, the coset includes $(0,0), (0,1), (1,0), (1,1)$. Particularly,

$$index^{(i)} = \mathbf{b}^{(i)} \mod 2, \text{ for } i \in \{0, 1 \ldots (m-1)\}. \tag{19}$$

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

$$LatticeHash = index^{(1)}, index^{(2)}, \ldots, index^{(m)}. \tag{20}$$

3.2 Evaluation Method 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 measure.

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 change the Gaussian variance, then analyzing how the noise variance affects the indexing scheme's performance.

NMSE is used to compare the robustness between 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 pairs $(\mathbf{x}, \mathbf{x}')$ 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:

$$NMSE = \frac{1}{n} MSE = \frac{1}{nN} \sum_{i=1}^{N} ||d'_{Ei} - d_{Ei}||. \tag{21}$$

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

3.3 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 one. 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:

$$\mathbf{x} \in R_i, \mathbf{x}' \in R_j, d_E(\mathbf{x}, \mathbf{x}') > 0, d_{WH}(\mathbf{x}, \mathbf{x}') = \mathbf{0}, \text{ for } i, j \in \{1, \ldots, k\}, i \neq j. \tag{22}$$

For instance, as shown in Fig. 4, 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 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 Sect. 3.1, all points in $V_{2\Lambda}(\mathbf{a})$ are indexed as shown in Figs. 4 and 5 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 as low MSE 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 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 [9], which is the point of the plane furthest from the lattice.

4 Simulation of Two-Dimensional Indexing Schemes

A two-dimensional nested lattice and a Gray indexing scheme were implemented and simulations were ran with the two input distribution cases

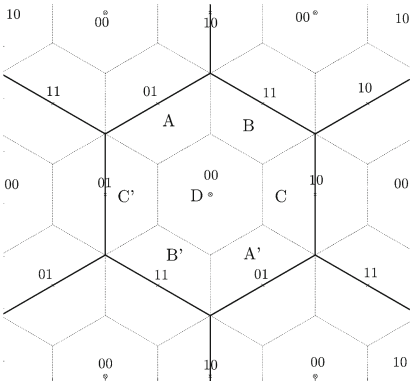

Fig. 4. Voronoi region $V_{2\Lambda}(\mathbf{0})$.

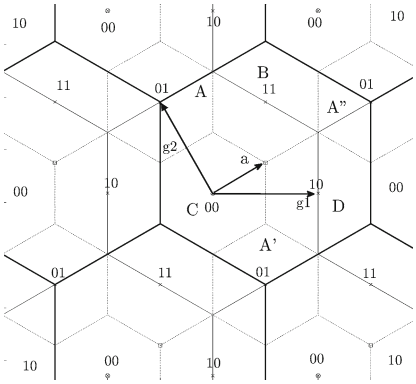

Fig. 5. Voronoi region $V_{2\Lambda}(\mathbf{a})$, where $\mathbf{a} = 2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2})$.

Table 1. Case (a): Nested lattice and Gray indexing scheme simulation information.

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

which are described in Sect. 3.2. 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 Sect. 3.2, the (d_E, d_H) for every sample pairs in the dataset, along with its linear predictor function for Gray indexing are shown in Fig. 6. 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 Figs. 7 and 8, respectively. Table 1 depicts the NMSE and other details about this comparison simulation. The smaller the NMSE, the better the linearity the indexing scheme can achieve.

When the input $\mathbf{x}$ is uniformly distributed and $\mathbf{x}'$ is Gaussian distributed, as in Case (b) in Sect. 3.2, 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 9 represents the

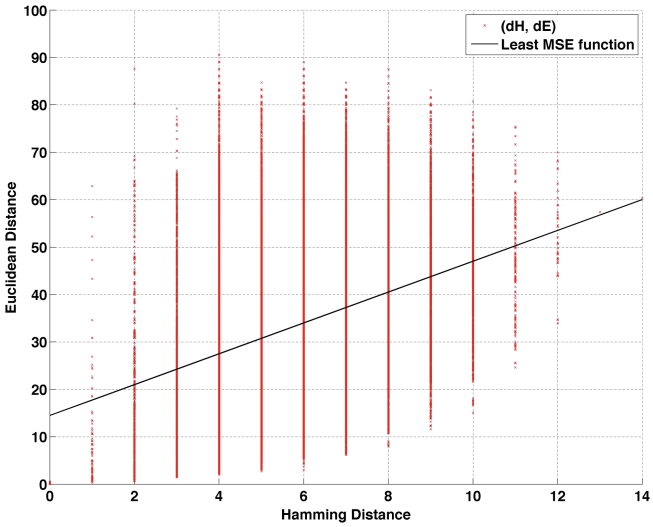

Fig. 6. Case (a): The Hamming distance versus the Euclidean distances of Gray indexing.

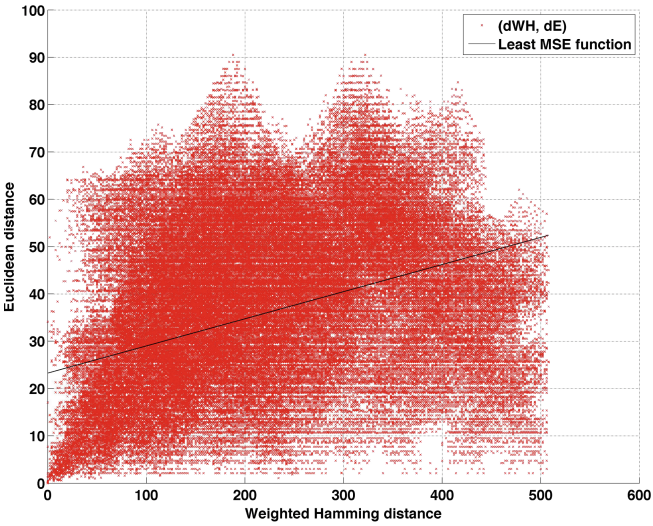

Fig. 7. Case (a): The weighted Hamming distance versus the Euclidean distances of nested lattice indexing.

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, nested lattice indexing generally have better performance than Gray indexing. In Case (a), the combination of nested lattice indexing and

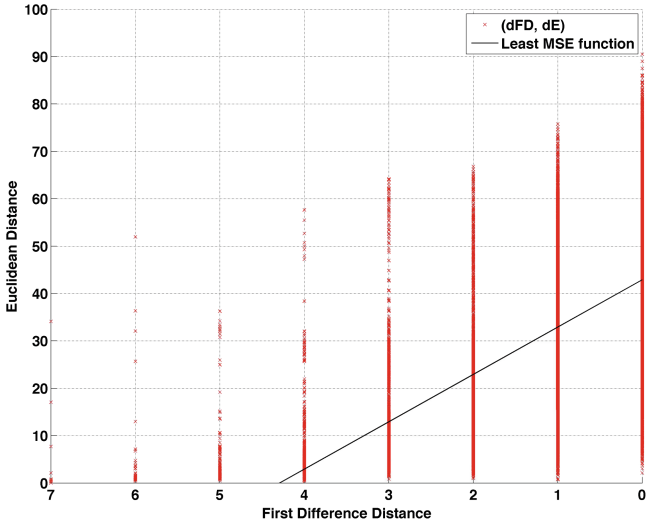

Fig. 8. Case (a): The first difference distance versus the Euclidean distances of nested lattice indexing.

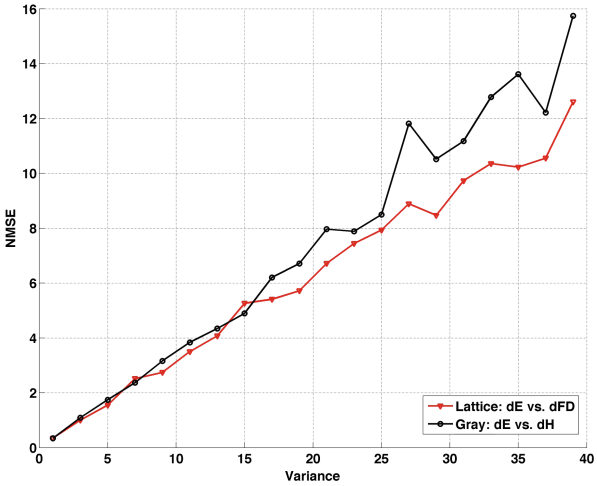

Fig. 9. Case (b): The variation of NMSE as a function of noise variance.

first difference distance reduces approximately 20 % NMSE compared to Gray indexing. In Case (b), on average, nested lattice indexing have smaller NMSE than Gray indexing.

5 Conclusion

In this paper, we developed a nested lattice indexing scheme that is suitable for content-based hash functions to increase the robustness. In addition, weighted Hamming distance, first difference distance and a coset lattice with shift vector $\mathbf{a} = 2/3(\mathbf{g_1}) + 1/3(\mathbf{g_2})$ were proposed to efficiently reflect the Euclidean distance. As a result, the NMSE of nested lattice indexing scheme was reduced 20 % compared to Gray indexing scheme. As future work, we will apply higher dimensional lattices with the expectation of a better relationship between metric distance and Euclidean distance. To demonstrate the effectiveness of lattice-based hashing, we plan to apply it to image's speeded up robust features (SURF) [11] to quantize and index SURF-based feature vectors to final hash values.

References

1. Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
2. Huang, C.-L., Huang, D.-H.: A content-based image retrieval system. Image Vis. Comput. **16**(3), 149–163 (1998)
3. Agrawal, R., Faloutsos, C., Swami, A.: Efficient similarity search in sequence databases. In: Lomet, D.B. (ed.) FODO 1993. LNCS, vol. 730, pp. 69–84. Springer, Heidelberg (1993)
4. Venkatesan, R., et al.: Robust image hashing. In: Proceedings, 2000 International Conference on Image Processing, vol. 3. IEEE (2000)
5. Hernandez, R.A.P., Miyatake, M.N., Kurkoski, B.M.: Robust image hashing using image normalization and SVD decomposition. In: 2011 IEEE 54th Midwest Symposium on MWSCAS. IEEE (2011)
6. Faloutsos, C.: Gray codes for partial match and range queries. IEEE Trans. Softw. Eng. **14**(10), 1381–1393 (1988)
7. Guopu, Z., et al.: Fragility analysis of adaptive quantization-based image hashing. IEEE Trans. Inf. Forensics Secur. **5**(1), 133–147 (2010)
8. Li, Y., et al.: Robust image hashing based on random Gabor filtering and dithered lattice vector quantization. IEEE Trans. Image Process. **21**, 1963–1980 (2012)
9. Conway, J.H., Sloane, N.J.A.: Sphere Packings, Lattices and Groups, 3rd edn. Springer, New York (1999)
10. Gray, R.M.: Vector quantization. IEEE ASSP Mag. **1**(2), 4–29 (1984)
11. Bay, H., Tuytelaars, T., Van Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006, Part I. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
12. Conway, J.H., Sloane, N.J.A.: A fast encoding method for lattice codes and quantizers. IEEE Trans. Inf. Theory **IT–29**, 820–824 (1983)
13. Gray, F.: Pulse code communication. U.S. Patent, 17 March 1953
14. Hamming, R.W.: Error detecting and error correcting codes. Bell Syst. Tech. J. **29**(2), 147–160 (1950)