A New Feature Coding Scheme for Video-Content Matching Tasks

A NEW FEATURE CODING SCHEME FOR VIDEO-CONTENT MATCHING TASKS

BY

YINGCHAN QIAO, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2017) McMaster University
(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	A New Feature Coding Scheme for Video-Content Matching Tasks
AUTHOR:	Yingchan Qiao B.Sc., (Electric Infroamtion Engineering) Beihang University, Beijing, China
SUPERVISOR:	Dr. Jun Chen
NUMBER OF PAGES:	x, 59

To my family and friends

Abstract

This thesis present a new feature coding scheme for video-content matching tasks. The purpose of this feature coding scheme is to compress features under a strict bitrate budget. Features contain two parts of information: the descriptors and the feature locations. We propose a variable level scalar quantizer for descriptors and a variable block size location coding scheme for feature locations. For descriptor coding, the SIFT descriptors are transformed using Karhunen-Loéve Transform (KLT). This K-L transformation matrix is trained using the descriptors extracted from the 25K-MIRFLICKR image dataset. The quantization of descriptors is applied after descriptor transformation. Our proposed descriptor quantizer allocates different bitrates to the elements in the transformed descriptor according to the sequence order. We establish the correlation between the descriptor quantizer distortion and the video matching performance, given a strict bitrate budget. Our feature location coding scheme is built upon the location histogram coding method. Instead of using uniform block size, we use different sizes of blocks to quantize different areas of a video frame. We have achieved nearly 50% reduction in the bitrate allocated for location information compared to the bitrate allocated by the coding schemes that use uniform block size. With this location coding scheme, we achieve almost the same video matching performance as that of the uniform block size coding. By combining the descriptor and location coding schemes, experimental results have shown that the overall feature coding scheme achieves excellent video matching performance.

Acknowledgements

I would like to express my deep gratitude to Professor Jun Chen, my research supervisors, for his patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to thank our administrative assistant of ECE department, McMaser university, Cheryl, for her advice and assistance in keeping my progress on schedule. My grateful thanks are also extended to Mr. Muhammad Airabeiah for his help in discussing the analysis procedure and revising the structure of my thesis, to Ms Ting Yin, who helped me utilizes the descriptor selection method and the video matching measurements.

Finally, I wish to thank my parents for their support and encouragement throughout my study.

Notation and abbreviations

SIFT:	Scale Invariant Feature Transform
SURF:	Speeded Up Robust Features
CHOG:	Compressed Histogram of Gradients
PCA:	Principal Component Analysis
LSH:	Locally Sensitive Hashing
KLT:	Karhunen-Loéve Transform
GOP:	Group Of Pictures
GCC:	Geometric Consistency Check
TPR:	True Positive Rate
FPR:	False Positive Rate

Contents

Abstract **iv**

Acknowledgements **vi**

Notation and abbreviations **vii**

1 Introduction **1**

1.1 Prior work in feature compression 4

1.2 Overview and contributions 7

2 Coding of Feature Descriptors **11**

2.1 Overview of Scale Invariant Feature Transform (SIFT) 11

2.2 Descriptor Persistency 14

2.3 Statistics of SIFT 15

2.3.1 The Karhunen-Loéve Transform 15

2.3.2 Empirical distributions of the transformed elements 16

2.4 Descriptor Coding Scheme 17

2.4.1 Descriptor Quantizer Design 19

2.4.2 Encoding of Descriptors 24

2.4.3 Decoding of Descriptors 25

3 Feature Location Coding **26**

3.1 Location Histogram Coding Overview 27

3.2 Feature Location Coding Design 29

3.2.1 Variable Block-sized Location Histogram Coding 30

4 Experimental Results **36**

4.1 Experiment and Data Set Description 36

4.2 Descriptor Coding Scheme Evaluation 38

4.2.1 The Relation Between Performance and The Number of Descriptors 39

4.2.2 Design of The Feature Descriptor Encoder 39

4.2.3 Performance of The Feature Descriptor Coding Using Different Number of Descriptors 45

4.3 The Evaluation of the Location Coding Scheme 46

4.3.1 The Threshold for Block Subdivision 47

4.3.2 Performance of Variable Block-size Location Coding 48

4.4 Performance of Overall Coding Scheme 52

5 Conclusion **54**

List of Figures

1.1 Video matching System Diagram. 8

1.2 Local feature compression scheme. 9

2.1 An extreme is detected by comparing with 26 neighbors in 3×3 image regions at current and adjacent scales. 13

2.2 Eigenvalues of Transformed Descriptor covariance matrix. 17

2.3 Distribution Histograms of sample bins. 18

2.4 The rate-distortion of the sample bins. 23

3.1 Context Template.The darker block is not considered as neighbors. For each block, nearby previously coded blocks are used as context. . . . 27

3.2 The clockwise circular Scanning method. 28

3.3 The examples of the persistent descriptor distributions using different numbers of descriptors. 30

3.4 An example of location encoding using variable block-size. 32

3.5 The workflow of a location histogram decoder 34

4.1 The TPR corresponding to the number of elements. 40

4.2 The TPR corresponding to the number of elements using 80 descriptors. 41

4.3 The TPR corresponding to the number of elements using 150 descriptors. 42

4.4 The TPR corresponding to the number of elements using 200 descriptors. 43

Chapter 1

Introduction

For the past few decades, digital media has been playing important role in modern life. This could be seen in the rapidly growing volume of multimedia content (visual content) on the Internet, most of which is uploaded from smart phones and tablets. Moreover, lots of computer vision applications have been receiving more interest recently due to the advances in computational capabilities and machine learning technologies. Object recognition [1], online image search and retrieval [2], and object tracking [3] are some good examples of those applications.

In the most of visual applications, local image features are widely used. They are designed to represent different characteristics of appearance or shapes in an image [4]. A local feature is composed of two parts: the location of an interest point and a descriptor. An interest point is located by an invariant region detector while the descriptor provides a concise representation of the image patch around each interest point. Since there are changes in illumination, scale and any other transformations in videos and images, visual features are expected to be robust and distinctive. Some efforts have been made on finding invariant local features. **(author?)** [5] have developed a scale invariant feature transform descriptors (SIFT) in 20th century. Of the many proposed features, it is probably the most commonly used in multiple visual research applications. Others like Speeded Up Robust Features (SURF) by **(author?)** [6] and Gradient Location and Orientation Histogram descriptor (GHOG) by **(author?)** [4] are also well-known in the computer vision community.

Many multimedia-based applications require transmission of visual data through networks with limited bandwidth. This imposes serious restriction on the transmission bitrate, which, in turn, demands efficient data compression techniques. Hence, the traditional choice is to compress the visual data before communicating them to the receiver, where the data is analyzed, called the " *Compress then Analyze*" (*CtA*) paradigm. In recent years, *CtA* scenarios is considered out of date. One of its disadvantages is that the visual content after transmission is a lossy representation. This usually compromises the receiver's ability to carry out the analysis task accurately[7].

Meanwhile, an alternative paradigm *Analyze then Compress* (*AtC*) has been gaining more attention. It builds on the fact that some visual applications work perfectly based on local features, disregarding the pixel to pixel content. The steps of *AtC* are performed by first extracting local features immediately. Then, they are compressed for efficient transmission. The compression of visual features becomes the key concern to successfully deploy *AtC* scheme since the network poses strict constraints on the available bandwidth. Several researches tackled this problem, making efforts to propose the efficient feature coding methods [8], [9], [10].

In this context, the Compact Descriptors for Visual Search (CDVS) standard is a great breakthrough proposed by the Moving Picture Experts Group (MPEG).

Generally, it builds a framework for descriptor extraction and performance evaluation, which helps to develop efficient and interoperable visual search application on image level. Extending this to videos, MPEG has been planning to standardize technologies for visual content matching in purpose, called Compact Descriptors for Video Analysis (CDVA). CDVA aims to design a compact representation of video features. A video can be viewed as multiple single frames. Analysis of videos requires a set of local descriptors representing each frame. As a result, a huge volume of data is produced, and, therefore, compression becomes necessary. Hence, CDVA places greater emphasis on smaller descriptor sizes and shorter runtimes.

There are three main advantages of feature compression [10] for video analysis tasks:

1. Reduced transmission latency: given available bandwidth of the network, it is infeasible to return matching results in a few minutes, since the huge volume of transmission data is needed for a video .

2. Reduced storage space: the local features extracted from a video need a large number of bits to present. Compression make it efficiently to store as much as video data in a limited storage space of database.

3. Reduced computational complexity: a matching application requires the comparison of thousands of descriptors in the query video to a comparable or greater number of descriptors extracted from the reference video. It would be too computationally demanding to keep the full information without compressing them.

As a starting point, MPEG dedicated the CDVA standard to basic visual analysis tasks, like search and retrieval. The video retrieval task is enabled by using the video matching application to find the best match among all candidate videos in the database. Performing video matching needs a pair of two videos. One is defined as query video and the other is the reference video.

The way of matching videos using the CDVA framework is to use CDVS features in each frame of a video. Most of the work in CDVA focuses on the technologies for efficient extraction of compact video features. However, video cannot be treated as a large set of single continuous pictures. The most remarkable differences between images and videos, temporal correlation and motion consistency, have not been fully utilized so far.

Therefore, the core idea in this work is an improved coding scheme for video features. It uses the property of feature persistentcy [11]. The proposed feature coding scheme can be used in many video analysis tasks. Our experimental results rely on the SIFT descriptors, which are commonly used in visual applications. However, this work is not constrained to this specific kind of descriptors. We mainly focus on video pairwise matching, but it can be easily extended to video retrieval applications. We show that our coding scheme provides a reliable matching performance with low data rate.

1.1 Prior work in feature compression

Local descriptors are usually high dimensional vectors. For instance, a SIFT descriptor is a 128-dimensional vector. Meanwhile, the corresponding feature location information takes 20 bits per descriptor to represent the special information of the horizontal and vertical coordinates (x,y) in each image. When extracting more features to enhance matching accuracy, the data for storing uncompressed location information grows fast.

Several methods are proposed to reduce the dimensionality of descriptors and to form Low-bit-rate descriptors. Most of designs use SIFT descriptors as benchmark. Descriptor compression first started with PCA-SIFT [12] descriptors. Ke and Sukthankar used 39×39 patch of pixel to compute horizontal and vertical gradient instead of 41×41 employed in SIFT. Then they implemented Principal Component Analysis (PCA) to convert SIFT descriptor to a 20-dimensional descriptor. PCA-SIFT descriptor beats SIFT on shorter computational runtime and higher matching accuracy. The author of [12], they learned that the patches around keypoints have the same scale properties. This allows the PCA-SIFT descriptors to represent the patch correctly with a small number of dimensions. **(author?)** [13] proposed a linear discriminant embedding system to build a linear projection used on non-linear transformed version of input features. Their method provided 5–10× dimensionality reduction relative to a original SIFT descriptor, while reaching a comparable image matching performance. **(author?)** [8] first observed that using distributed source coding could reduce bitrate when matching visual correspondence. Therefore, they proposed an approach of using quantized random projection on SIFT and further construct a compact descriptor hash. Hamming distance of binary hashes is adopted when computing the similarities between features. In their paper,they proved that using descriptor hashes to determine visual correspondence is more rate efficient and less computational consuming. This ramdom projection method for finding matching features is more rate-friendly than other prior methods. By studying the entropy characteristics of SURF descriptors and using an information theoretic approach, Takacs developed an entropy encoder for SURF in[1] that reduces the size of a descriptor 7 times without affecting matching performance. **(author?)** [14] introduced machine learning into representing image information. They found that Locally Sensitive Hashing (LSH) produces binary descriptors that work fine when used in a number of computer vision applications. When limited to a small bitrate, LSH performed poorly. In their method, they generated binary descriptors from a new learning approach: Restricted Boltzmann Machines (RBM). Most recently, a brand new descriptor was created by Chandrasekar in [15], called Compressed Histogram of Gradients (CHOG). They exploited the gradient statistic properties obtained around interest keypoints in canonical patches and considered a histogram of gradient as a descriptor. This unique design of histogram allowed them to use more efficient distance measurements in terms of matching or retrieval research. Note that building from gradient statistic model enables encoding descriptors more efficiently by taking advantage of the information theoretical method. They finally chose a lossless encoding scheme as Huffman tree algorithm and achieved a comparable matching porformance as with uncompressed descriptors.

Another interesting approach introduced by Chandrasekhar [9] is a lossy compression scheme based on transformed features. They first applied Karhunen-Loéve Transform (KLT) on each individual feature descriptor in order to decorrelate its elements and quantize them using a uniform scalar quantizer. The bitrate is reduced by using an entropy encoder. They have also studied the trade-off between rate and distortion. At the decoder, inversing the transformation is applied to decoded feature descriptors. By comparing to the original SIFT descriptor, they achieve a 16× compression rate with negligible matching error introduced.

On the other hand, compressing the information of feature locations has not gained much attention compered to descriptor compression. Location coding is prompted by the process of CDVS. The most remarkable achievements are coming from the research teams of Stanford University, Peking University and Huawei [16]. To summarize, all proposed methods utilize the fact that features are concentrated around interesting objects in images rather than being randomly scattered. CDVS adopted location histogram coding scheme introduced by **(author?)** [17], which lays a special uniform-sized grid on the top of the image and generates a location histogram map to represent location information. For compressing the histogram map they introduced a context-based arithmetic coder ,which encodes each block of the histogram with conditional probabilities given a specific map pattern. Later, they designed an improved encoding method built upon their previous work, called sum-based context coding technologies [18]. By using this improved location histogram coding method, they overcame the drawback that the reference context table grows exponentially fast as the context consists of greater number of neighborhood points stages. While sum-based context is suitable for larger range of map patterns, it is time consuming to collecting training data to form the context table.

1.2 Overview and contributions

Fig 1.1 depicts a block diagram of our video feature coding scheme, and Fig 1.2 shows a more detailed block diagram for the compression process. As shown in Fig 1.1, video feature coding scheme is composed of two parts: the encoder and the decoder. Query features are extracted and compressed at the encoder side and transmitted to the decoder. This work uses the concept of *descriptor persistentcy* [11], which utilizes

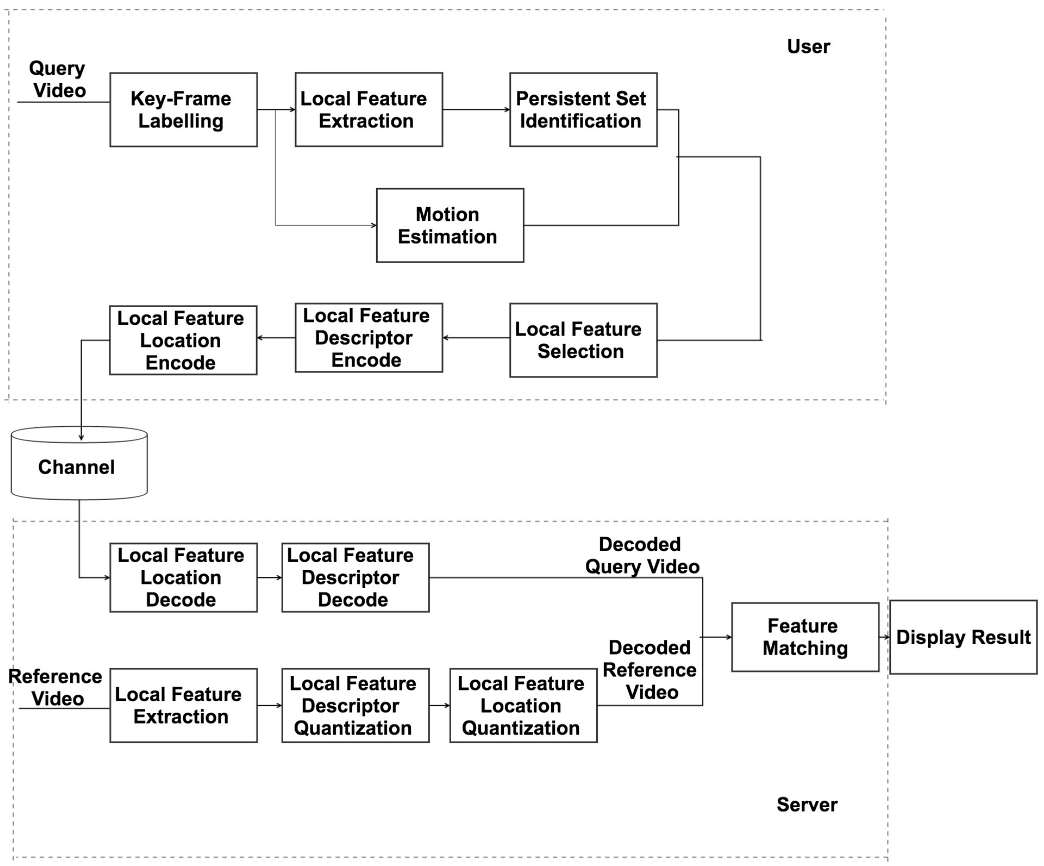


Figure 1.1: Video matching System Diagram.

temporal correlation and motion consistency (more details will be given in Ch.2). We first segment videos into groups, called Groups Of Pictures(GOP). Each GOP contains several consecutive frames. Then, we extract SIFT descriptors and select a subset of the persistent descriptors in GOPs. The main purpose of feature coding scheme is to compress them efficiently.

In this work, we mainly focus on video feature compression, which is presented in Fig.1.2. The proposed feature coding scheme has two parts: the encoder and decoder. Firstly, we introduce a descriptor compression scheme based on the Karhunen-Loéve

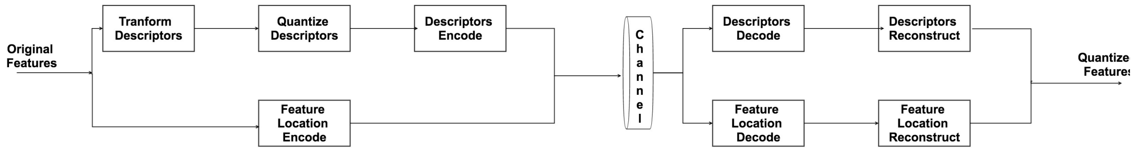


Figure 1.2: Local feature compression scheme.

Transformation (KLT). The encoder generates a lossy version of the descriptors. This is achieved by reducing the dimensionality after descriptor transformation. Then, we apply a scalar quantizer to each element of the transformed descriptor. This scalar quantizer uses the empirical distribution estimated for every element of the transformed descriptor using a training dataset (25K-MIRFLICKR database [19]). We use the water-filling algorithm to allocate the rate to every element. The quantization level varies in order to suit the different bitrates assigned to each descriptor.

Now we turn the attention to coding the locations of the extracted features. The importance of designing location coding is that if we use floating number to represent location coordinates (x, y), it would produce a comparable size to that of the compressed descriptors [17]. Our coding method builds upon the location histogram coding scheme adopted in CDVS standard. The innovation in our location coding is that we use variable block-size grid to build the histogram instead of the uniform size grid used in CDVS. This choice was inspired by the observation on the distribution of the persistent descriptors in a frame. The scheme is designed to first use a large block size grid. Within this grid, for the block with many features, we use smaller size blocks to finer quantize the locations of those features. To compress the location information, we follow the same encoding method as in CDVS, a context-based arithmetic encoder is used for the histogram map and an ordinary arithmetic encoder is used for the histogram counts.

The video matching part happens at the decoder side, where reference videos are quantized using our proposed feature coding method. The reference and query videos are declared to be a matching pair if half of the GOPs find matches in the reference video. The entire matching scheme can be described as follows. Firstly, we match the persistent set of each GOP in the query with the features extracted from every frame in the reference. For every matching operation, we introduce ratio test and Geometric consistency check (GCC) to determine the number of matching pairs of descriptors [11]. The features from the query GOPs and the reference frame are examined using the ratio test. It produces a set of probable matching descriptor pairs with some mismatches. Then this set of matching descriptor pairs is refined using GCC to obtain a more accurate set of matching descriptor pairs. We use the number of the matching pairs to indicate how similar the query and the reference videos are [11].

The rest of this thesis is organized according to the architecture in Fig 1.2. In Chap 2, and Chap 3, we present our descriptor and location coding schemes respectively. The organization of those two chapters is similar. First, we start by presenting the background and a brief introduction of the techniques implemented. Then, we move to describing our coding methods. In Chap 4, the experimental results of descriptor coding and feature location coding are presented separately. Moreover, we combine these two parts as a whole feature compression method and use it into the video matching scheme in [11]. The matching performance evaluation of the feature coding scheme is shown in the end of this chapter.

Chapter 2

Coding of Feature Descriptors

This chapter introduces a variable level scalar quantizer for each element of the SIFT descriptor. Our aim is to code each feature individually. Given a bitrate budgets, the descriptor coding scheme has the flexibility to choose the quantization levels according to the number of descriptors used to represent a frame in videos. Since the descriptors extracted in a frame are distinct, it is better if we can develop an encoding scheme for every descriptor itself. Independently code each feature has its own apparent advantage. It leads to the efficient quantization of each descriptor without any pre-knowledge from other neighbor descriptors by using the intrinsic feature statistics.

2.1 Overview of Scale Invariant Feature Transform (SIFT)

SIFT[5] starts with detecting interested keypoints. In order to identify the points that their locations and scale are reliable under different views of a same object or scene, SIFT constructs a Gaussian scale space $L(x, y, \sigma)$, consisting of continuous possible scales. It is generated from the convolution of the original input image $I(x, y)$ with Gaussian functions with valuable widths $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.1}$$

To detect constant keypoints in scale space, SIFT computes scale-space extrema in the Difference-of-Gaussian (DoG) function $D(x, y, \sigma)$, which is obtained by subtracting a smaller scaled Gaussian convolved image from a k scaled one: $D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$. Each octave of scale-space is divided into an integer number s, every octave has $s + 3$ images and forms $s + 2$ DoG images. Next octave is processed by resampling Gaussian-convolved images and taking every second pixels in each row and column. The extrema in DoG pyramid are defined as keypoints. An extremum is usually defined as the maximum or the minimum point compared to its 26 neighbor pixels, 8 in current scale and 9 in the scale above and below accordingly, shown in Fig.2.1. Then the localization accuracy is improved by applying a second order Taylor series expansion, which gives the final location:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}}, \tag{2.2}$$

where D and its derivatives are calculated at this point and $\mathbf{x}$ is represented as $\mathbf{x} = (x, y, \sigma)^T$. To eliminate weak extrema corresponding to edges, the function value at an extremum is used. If its value is below a fixed threshold, this point is excluded.

After detecting keypoints from normalized input patch, the gradient location and the orientation need to be assigned. The orientation histogram contains 36 bins,

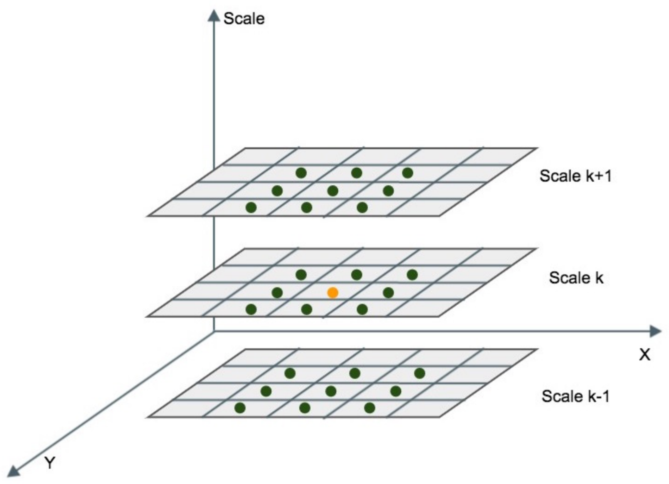


Figure 2.1: An extreme is detected by comparing with 26 neighbors in 3×3 image regions at current and adjacent scales.

covering 360 degrees of orientation. It is formed by the gradient orientations of this point within a 4×4 location grid region. Both gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are precomputed using pixel differences. The contribution to the location and orientation of each sample is weighted by the gradient magnitude and a Gaussian-weighted circular window. The highest peak in histogram is considered the dominant direction of the gradients. A keypoint is created with this dominant direction. All steps described above produce keypoints characterized by their location, orientation and scale.

A keypoint descriptor is computed based on the gradient magnitude and orientation of a collection of pixels within a 4×4 image region around this keypoint location that is as distinctive as possible. The orientation of every pixel within this region is quantized into 8 angles and the gradient magnitude of each histogram entry is the summation of each pixel's magnitudes whose orientation is quantized to this direction. The descriptor is formed as a vector consisting of the values of all entries in an orientation histogram. The length of each SIFT descriptor is $4 \times 4 \times 8 = 128$. Finally the feature vector is normalized to unit length to be invariant to illumination.

2.2 Descriptor Persistency

In [11], the concept of descriptor persistency have been introduced. A descriptor is usually called persistent if it exists in several consecutive frames (GOP). The persistent set is a collection of persistent descriptors in a GOP. The most straightforward method to find that persistent set of a GOP requires matching all descriptors extracted fron all frames of the GOP, which is infeasible due to the computational complexity of this matching process. Hence, [11] has introduced a method to approximate the persistent set. This approximation of a persistent set is obtained using two nonadjacent frames in a GOP, which are referred to as keyframes. The step of Keyframe labeling in Fig 1.1 defines a fixed number of consecutive K frames as a GOP. The keyframes are chosen as the first and last frames in a GOP. The first frame in a GOP further is viewed as the base frame while the last frame is the reference frame. We use ratio test and Geometric Consistency Check (GCC) to find the matching descriptor pairs in the baseframe and the reference frame. Additionally, the motion estimation approach is used to rank the matching descriptors found after ratio test and GCC. This ranking allows the encoder to control the size of the output persistent set. This is the main purpose of the *Persistent Set Identification* step.

2.3 Statistics of SIFT

As introduced in Sec.1.1, many efforts focus on dimensionality reduction, since the analyzing tasks with high dimensional descriptors is extremely computational demanding [12], [20], [9]. KLT is used widely to reduce descriptor dimension with limited performance error introduced[21].

2.3.1 The Karhunen-Loéve Transform

The objective of the KLT is to find a lower-dimensional representation that accounts for the variance of the elements in the descriptors. The measure of importance of these elements is given by the eigenvalues of the covariance matrix across the image space. The low dimensionality can be achieved by dropping some elements with relatively small eigenvalues.

We adopt the KLT in our descriptor coding scheme with two strong reasons. One important consideration is that the mean squared error (MSE) is the minimum compared to any other dimensional truncation using linear and unitary transformation[21], since only the elements with the eigenvalue smaller than a given threshold are discarded. In other words, K-L transformation gives the best representation of the original feature descriptors in a shorter length version while the lowest loss is achieved. Another advantage is taken from the fact that KLT is easy to implement. KLT is feature dependent, and obtaining the covariance matrix from a collection of an exhaustive set of training data is computation consuming. However, once the computation of KLT is done for a large training set, the covariance matrix of the data source is known both at encoder and decoder side.

We view each SIFT descriptor as a 1×128 vector. Every element in a feature vector represents a single dimension in SIFT. Given a large set of training descriptors $(\mathbf{d}^1, \mathbf{d}^2, \mathbf{d}^3, \ldots, \mathbf{d}^N)$, we compute the mean value and the covariance matrix as:

$$\bar{\mathbf{d}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{d}^n, \qquad \mathbf{C}_d = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{d}^n - \bar{\mathbf{d}}) * (\mathbf{d}^n - \bar{\mathbf{d}})^T, \tag{2.3}$$

where $\bar{\mathbf{d}}$ is a vector of 1×128, $\mathbf{C}_d$ is a symmetric matrix of 128×128. all eigenvalues of $\mathbf{C}_d$ are nonnegative since it is a covariance matrix. The orthogonal eigenvectors of $\mathbf{C}_d$ forms the basis function of features, while the eigenvalues, $\lambda_1, \lambda_2, \ldots, \lambda_k$, shows the average distribution of each element in feature vectors. By multiplying original SIFT descriptor with the eigenvectors, all dimensions in the SIFT descriptors are transformed and decorrelated. We further compute the covariance matrix $\mathbf{C}_{d\_tran}$ to the transformed descriptor. The eigenvalues of $\mathbf{C}_{d\_tran}$ are shown in Fig 2.2. As shown in Fig. 2.2, all eigenvalues of $\mathbf{C}_{d\_tran}$ are in descending order, which reveals the fact that elements at the head of a K-L transformed feature vector have more importance than those in the tail. Moreover, since the eigenvalues decrease sharply, most of the energy is concentrated in the first k coefficients. Only using the first k coefficients of transformed descriptors helps us to achieve the purpose of dimensionality truncation. Since the eigenvalue drops sharply as the number of elements increases, the maximum number of elements in the transformed descriptor used to design the descriptor quantizer is 40.

2.3.2 Empirical distributions of the transformed elements

Since KLT is applied to all descriptors in the training set, we collect all values of each element in the transformed descriptors to build the empirical distribution histogram for them, as shown in Fig.2.3.

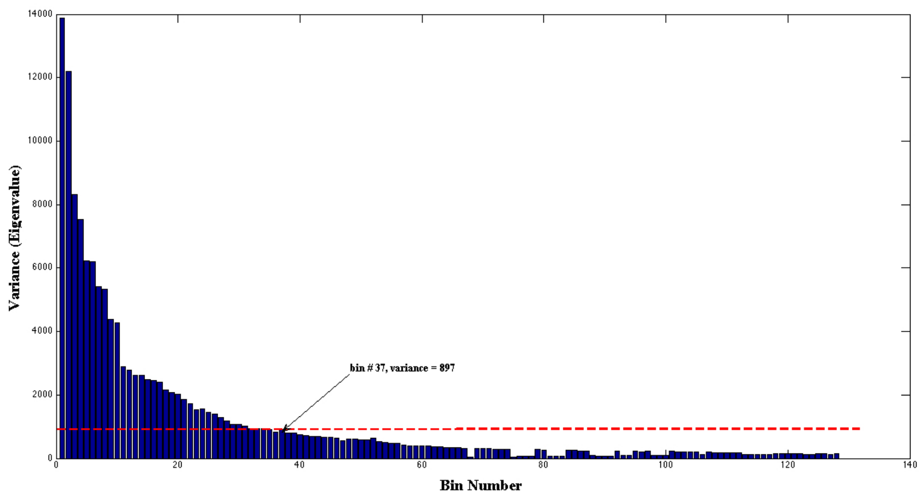
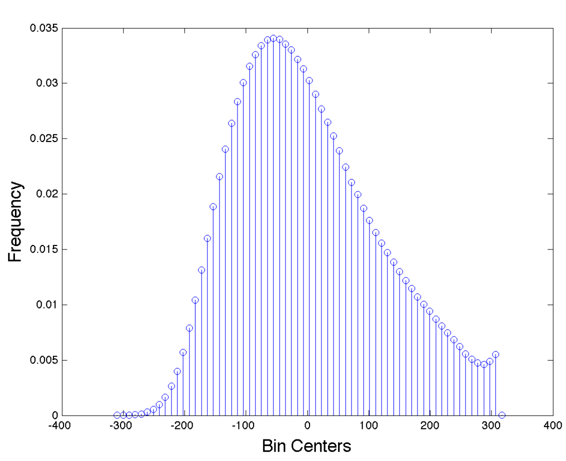


Figure 2.2: Eigenvalues of Transformed Descriptor covariance matrix.

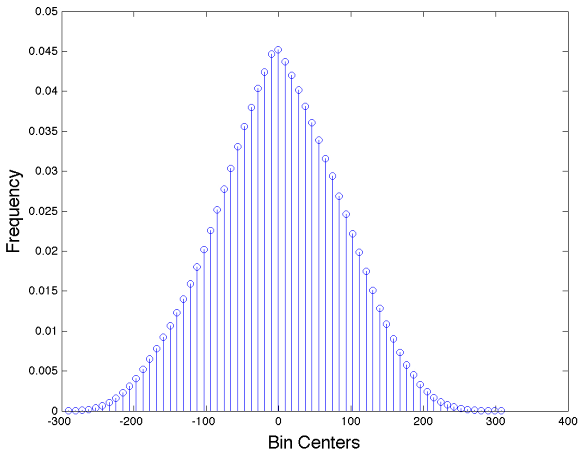
We use the mean of the interval to represent the interval in $x - axis$ while the $y - axis$ shows the frequency of the element value located in this histogram interval. As shown in Fig 2.3, the distributions of elements in transformed descriptors are Gaussian-liked. Taking advantage of this finding, we introduce the theory of Optimal Water-Filling to design a variable level quantizer for the transformed feature vectors instead of a uniform scalar quantization that describes in [9].

2.4 Descriptor Coding Scheme

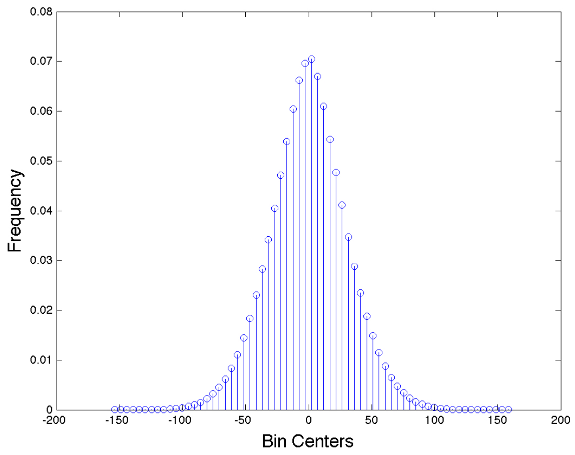
We build a descriptor coding method for K-L transformed descriptors. KLT not only can be used for dimensionality reduction, but also enable us to treat each dimension individually. By performing KLT, the elements of the SIFT descriptor become decorrelated. The goal of our coding is to compress elements according to their importance in the transformed descriptor.



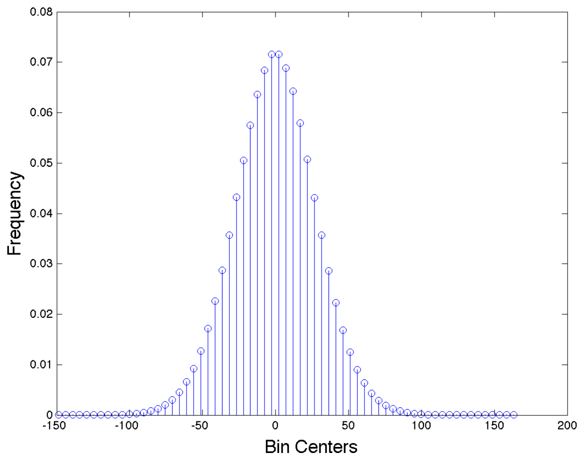
(a) 1st element Distribution



(b) 4th element Distribution



(c) 35th element Distribution



(d) 40th element Distribution

Figure 2.3: Distribution Histograms of sample bins.

2.4.1 Descriptor Quantizer Design

Exploit the Algorithm of Optimal Water-Filling

The Water-Filling algorithm is widely applied to solve the problem that the total rate is shared with multiple users, and each of users has different noise conditions. This total rate constraint is placed among all users [22]. In the proposed feature descriptor coding scheme, our descriptor quantizer assigns different bitrate to represent different elements in a transformed descriptor. The loss of accuracy is an analogue to the noise of each user while the total volume of bitrate is an analogue to the rate constraints in the optimal water filling problem. This ensures the descriptor coding scheme can achieve the minimum loss of matching accuracy with the lowest bitrate.

For the first k elements, where most of energy compacts, a finer quantizer helps to obtain a better approximation to the real value in an element of transformed descriptor. The objective of developing descriptor quantizer is to set different quantization levels according to the element sequence. More levels are needed for the elements in the head. As the element number increases, less levels are needed. For each element, we apply a scalar quantization. The descriptor encoder transmits a sequence of interval indexes to the decoder. For the index of n_{th} element, e_n, needs $\log_2(e_n)$ bits to represent. Additionally, any choice of quantization levels L_n used to quantize this element in $S = \{2^{r-1}+1, \ldots, 2^r - 1, 2^r\}$ requires r bits to represent e_n. Therefore, we set the number of quantization levels to be 2^r given the number of bit r. Given the number of elements used for quantization k, the number of descriptors M, and the number of quantization levels for each element, this descriptor design can be modeled as the following problem.

$$\begin{aligned} \min \quad & D = \sum_{l=1}^{L} \sum_{b_n \in \alpha_l} (b_n - \hat{e_n})^2 p(b_n), \\ \text{s.t.} \quad & \sum_{d=1}^{M} \sum_{n=1}^{k} \log_2 L_n \leq R \end{aligned} \tag{2.4}$$

where b_n is the value of n_{th} element in a descriptor; $p(b_n)$ is the probability of n_{th} element takes the value of b_n; $\hat{e_n}$ is the reconstruction value which is used to construct the quantized version of descriptors; R is the bit budget for the transmission of all descriptors.

The Distortion of the Descriptor Quantizer

Quantizing an element of a transformed descriptors means that the empirical distribution histogram is segmented into several quantization intervals, each of which is delimited with thresholds and has a representative referred to as the reconstruction value. The calculation of thresholds and reconstruction values of intervals is based on a large set of training descriptors.

We use Lloyd-Max Algorithm [23] to compute thresholds and representation values. From Fig 2.3, we obtain the distribution $y-axis$ and mean value $x-axis$ of each element. The design of Lloyd-Max's scalar quantizer is to find its decision thresholds and the reconstruction values for every intervals given the number of quantization levels for every interval. Since the empirical distributions we estimated are discrete, the maximum number of intervals for each distribution is limited to 32 intervals. The stopping criterion for this algorithm is based on the change in distortion between two iterations. If this change (difference) is below a certain level, the algorithm is terminated.

Let an L-level scalar quantizer $Q(b_n)$ be characterized in terms of a set of L real-valued quantization points for element b_n, *i.e.*, reconstruction values $\{y_1, y_2, y_3, \ldots, y_L\}$ and decision thresholds $\{t_0, t_1, t_2, \ldots, t_L\}$. In our case, t_0, t_L represent respectively the minimum and maximum shown in distribution histogram, according to different situations of bins. Decision thresholds are used to define the intervals of the quantizer. We have an ordering of the thresholds, that is,$t_0 < t_1 < t_2 < \cdots < t_{L-1} < t_L$. Thus, intervals are given by:

$$\alpha_m = (t_{m-1}, t_m], m = 1, 2, \ldots, L \tag{2.5}$$

Intervals of $\{\alpha_2, \ldots, \alpha_{L-1}\}$ are called the inner intervals, while α_0, α_L are referred to as the outer intervals. An element is quantized to the reconstruction value y_m if its value falls in the quantization interval α_m . Hence, an L-level scalar quantizer is defined as a functional mapping of an input bin value b_n onto an output representation:

$$Q(b_n) = y_m, \quad b_n \in \alpha_m. \tag{2.6}$$

The quality of a quantizer is measured by the distortion of the output reconstruction value comparing to the original. The most common and convenient measurement of distortion between the input value and reproduction is the average mean-squared error. We consider a quantizer to be optimal if there is no L-level scalar quantizer that can provide lower distortion. The total distortion is defined as:

$$D = \sum_{m=1}^{L} \sum_{b_n \in \alpha_m} (b_n - y_m)^2 p(b_n), \tag{2.7}$$

where $p(b_n)$ is discrete normalized distribution histogram with $\sum_{b_n} p(b_n) = 1$. Minimizing the distortion D is to take derivatives of Eq. 2.7 with y_m , α_m, leading to those conditions for the optimal quantizers:

$$t_m = \frac{y_m + y_{m+1}}{2} \tag{2.8}$$

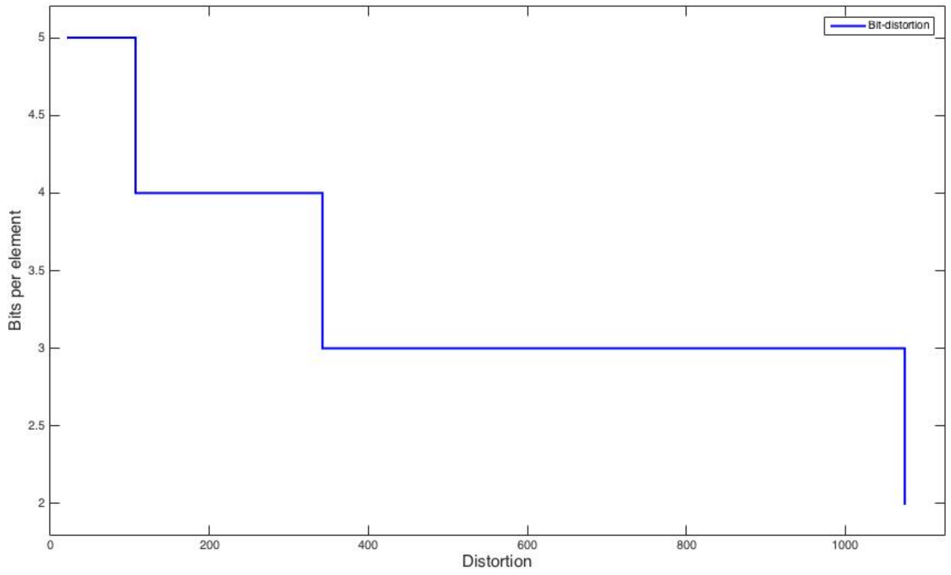
$$y_m = \frac{\sum_{t_{m-1}}^{t_m} p(b_n) b_n}{\sum_{t_{m-1}}^{t_m} p(b_n)} \tag{2.9}$$

The way of implementing the L-M algorithm is by first generating an initial set of intervals that uniformly partition the distribution histogram into L intervals. And then, using Eq. 2.8 and Eq. 2.9 alternatively until the distortion convergence is obtained. Once the scalar quantizer is determined for every elements, the thresholds and reconstruction values are stored for use at encoder and decoder.

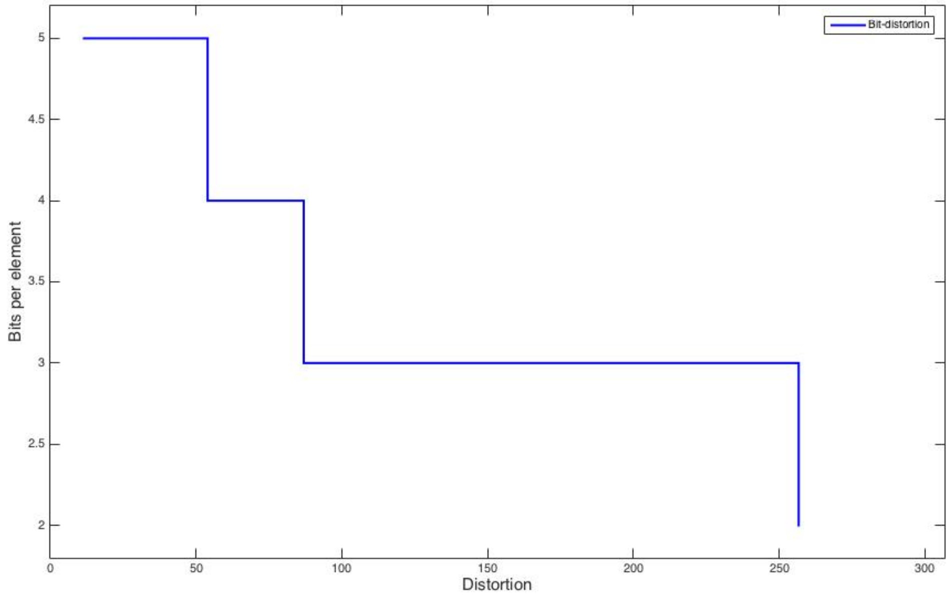
The Relation Between Bitrate and distortion

In this section, we discuss the relation between the bitrate and the distortion of the descriptor quantizer. In Sec.2.4.1, we mention that the maximum number of intervals is 32 so that the maximum number of bits used for each index is 5. Hence, we consider four options to the number of levels used to quantize the elements. They are 32, 16, 8, and 4. We compute the distortion for each setting of the quantization level using Eq.2.7 for every empirical distribution of elements. We generate the figures presenting the number of bins and corresponding distortion.

Shown in Fig.2.4 are the relation between the rate and distortion of the example bins. We can see that the number of bits used for different elements is different



(a) 1st bin



(b) 20th bin

Figure 2.4: The rate-distortion of the sample bins.

given a specific distortion. The more bitrates are allocated to the elements in the head while the fewer are allocated for the elements in the tail. We aim to design different descriptor quantizer for different number of descriptors using the same bitrate budgets. Given a fixed number of descriptors M, we compute the minimal distortions using different number of elements while maintaining the bitrate to be lower than the bitrate constraint using the generalized Water-Filling idea in Sec.2.4.1. For different number of element used in the descriptor quantizer k, the number of quantization levels for each element is determined by the minimum distortion to this k elements and the bitrate-distortion figures as shown in Fig.2.4.

Then, several descriptor quantizers are designed according to different number of elements while the number of descriptors is M. To decide the descriptor encoding scheme of using M descriptors of frames to perform video matching tasks, we apply these descriptor quantizers into the tasks of video matching and compare their matching performance. The descriptor quantizer for this number of descriptors M is determined to be the one that provide us the best matching performance.

2.4.2 Encoding of Descriptors

The encoder applies scalar quantization only to the first k elements of the transformed descriptors. The encoder takes each element individually and finds the corresponding interval. Then, it generates the sequence of interval indexes and passes them to the decoder.

2.4.3 Decoding of Descriptors

Since the descriptor quantizer is known both at the encoder and decoder side, once it receives an index of the quantization intervals, it maps these indexes to real-valued reconstructions to form the quantized descriptors.

Chapter 3

Feature Location Coding

In this chapter, we propose a scheme of compressing the location information to reduce the bitrate required to transmit feature locations. Features are usually extracted from corners or edges in an image, and their location information is always represented in floating numbers. Besides, the accurate representation of feature locations is related to the performance of matching applications, since they are used in GCC step. Hence, the location information of a large set of descriptors usually takes use of a huge amount of data to transmit. A natural approach to lossily coding the feature locations is to quantize the locations and entropy code the quantized data. In [20], a lossy encoding feature locations method, called the histogram location coding is adopted in the CDVS standard. It first covers the image uniformly with non-overlapping blocks and converts the feature location into the histogram map. To compress the data used for transmitting the histogram map, the context-based arithmetic coder is used.

In our feature location coding scheme, we make a improvement on the location coding method used in CDVS standard. Since we only use a small set of features to represent the keyframe in videos, we can prove that some large areas of the frame

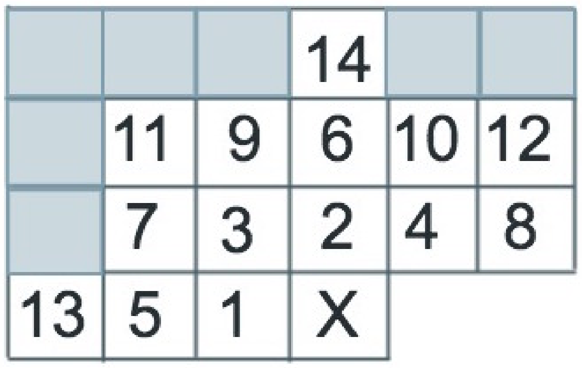


Figure 3.1: Context Template.The darker block is not considered as neighbors. For each block, nearby previously coded blocks are used as context.

do not contain any descriptor. The improvement is achieved by using a large size of blocks to quantize the " somehow useless " parts of frame while using the smaller size of blocks for the areas where lots of features are gathered.

3.1 Location Histogram Coding Overview

Given an input patch, SIFT extraction is applied first to detect the interest points. Then, this image would be overlaid with the non-overlapping blocks. A squared block of size 3×3 is typically used. The feature location is represented by a histogram map and the histogram counts. Each histogram block represents the block used to cover the image. The histogram map is a binary map that indicates whether a histogram block is empty or non-empty. They use '1' as a flag for the non-empty block and '0 ' otherwise. The location histogram counts shows the number of features within the non-empty blocks. Since features are re-ordered by their location information, the order of descriptors is changed correspondingly.

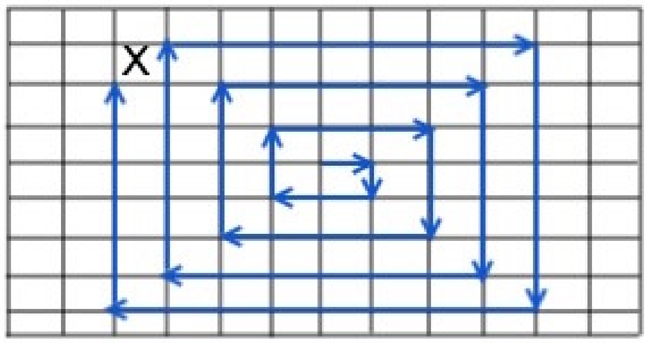


Figure 3.2: The clockwise circular Scanning method.

Building the histogram map requires to scan the blocks using the predetermined scan order. The scanning method used in CDVS is a clockwise circular scanning since features are dense in the central area of an image as in Fig 3.2. The histogram map is encoded by the context-based arithmetic coder. It uses the states of the 14 neighbor blocks within a template to encode this current scanned block. The template used is shown as Fig 3.1. The neighbors are set based on the order of the Euclidian distance to the coded block. However, the indexes of neighbors are not strictly defined as in Fig 3.1, because some neighbors have the same distance to the target coding block, *i.e.*, neighbors pairs with the index of 1 and 2 and another pair with indexed of 3 and 4 to the current block X.

Let (x_m, y_m) be the m^{th} neighbor blocks in the histogram map, the context for the current block is defined as :

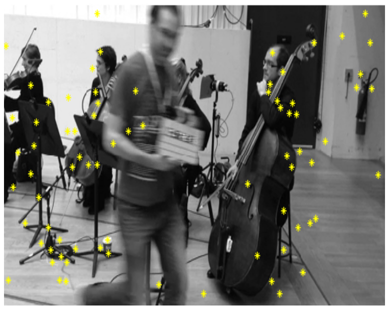
$$p_c = \sum_{m=1}^{14} 2^{m-1} \times I(m) \tag{3.1}$$

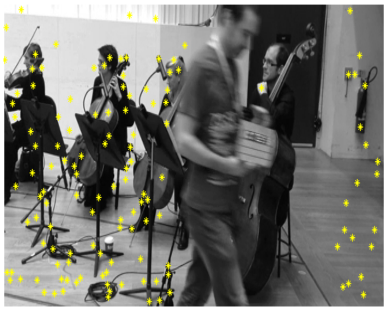
where $I(m)$ indicates the state of m^{th} neighbor block, taking a value of 0 or 1. For each symbol in the histogram map, we collect the states of its neighbors and compute p_c given the neighbor template. The context-based arithmetic coding uses the conditional probability $P(t|p_c)$ to encode symbol t. To encode the histogram counts, a 64 symbol, single model, static arithmetic coder is used. In [9], they learn the effect of the template size on the coding performance. They get the conclusion that using the larger template improves the coding performance, but the gain is negligible when the size of it increases over 16.

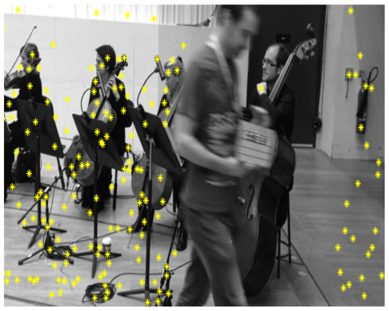
At the decoder, the histogram location decoder reconstructs the states of the histogram blocks. To obtain the reconstructed location of the features that are in the same non-empty block, the decoder quantizes them all to the coordinates of the center points. The number of features in this non-empty block is represented in the histogram counts.

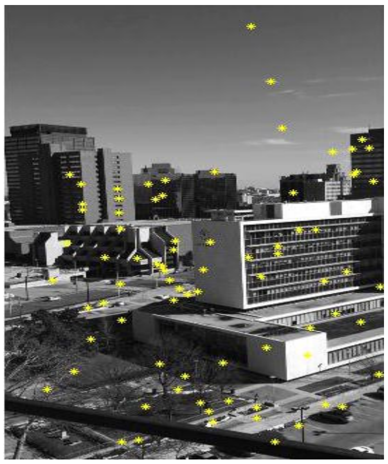
3.2 Feature Location Coding Design

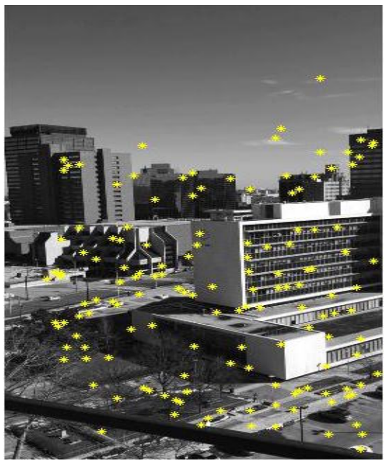
Our proposed method is built upon the location histogram coding scheme described in Sec 3.1, called variable block-sized location coding. It uses different sizes of blocks to encode the location of features for different areas of a keyframe. As shown in Fig 3.3 are the examples of the subset of persistent descriptors within a GOP. The numbers of persistent descriptors in each frame are 80,150, and 200 accordingly. Our advantage is taken from the insight that most of the area in the keyframe does not have the features that exist within a GOP. If the same histogram location coding method in Sec 3.1 is performed for this small set of features, most of the histogram blocks are shown as empty. This is a waste that if we transmit many 0s through network, since only the information of the non-empty histogram blocks is useful in


(a) Case 1: 80 descriptors


(b) Case 1:150 descriptors


(c) Case 1:200 descriptors


(d) Case 2: 80 descriptors


(e) Case 2:150 descriptors

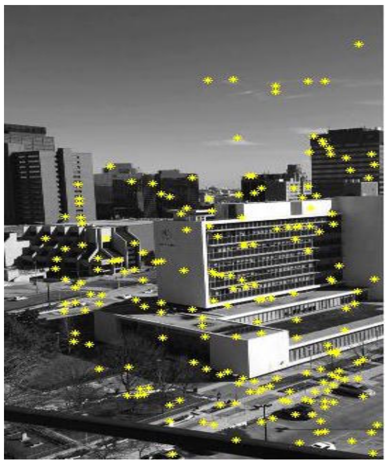

(f) Case 2:200 descriptors

Figure 3.3: The examples of the persistent descriptor distributions using different numbers of descriptors.

video matching.

3.2.1 Variable Block-sized Location Histogram Coding

Encoder

We apply the raster scan order as start from top left to the bottom right of a frame. We first use the large size of blocks to partition the keyframes. At the same time, we record the number of descriptors in every non-empty block as the same described in Sec.3.1. However, as for the case when several features are gathered in one large block, only using the same coordinates to represent the location of them causes the mismatched feature increased when using GCC in video matching. Hence, we provide the method to subdivide the large block into smaller blocks. In this way, those features that are gathering in one large block obtain a better quantization to their locations. We introduce the subdivision threshold $Thres_{sub}$ to determine when a large block needs to be subdivided, that is, when the counts value of it exceeds $Thres_{sub}$. The states of the smaller histogram blocks are recorded right behind the state to their parent block. The scanning method used for these smaller blocks is the raster scan as well. The example of how to use variable sizes of blocks to build the histogram map is shown in Fig 3.4.

At this time, one problem needs to be solved, in which the size of the histogram map is no longer a predetermined parameter. For the case of coding with uniform blocks, the columns and rows of the histogram map are determined by the size of the frame:

$$R_m = \frac{W_f}{W_b}, \quad C_m = \frac{H_f}{H_b} \tag{3.2}$$

where R_m and C_m is the number of rows and columns of the histogram map respectively, W_f and H_f indicates the width and height of a frame, and W_b and H_b is for blocks correspondingly. The subdivision operation increases the size of histogram map. The number of small blocks for finer quantizing a larger block is computed as:

$$N_s = \frac{W_b \times H_b}{W_{sb} \times H_{sb}} \tag{3.3}$$

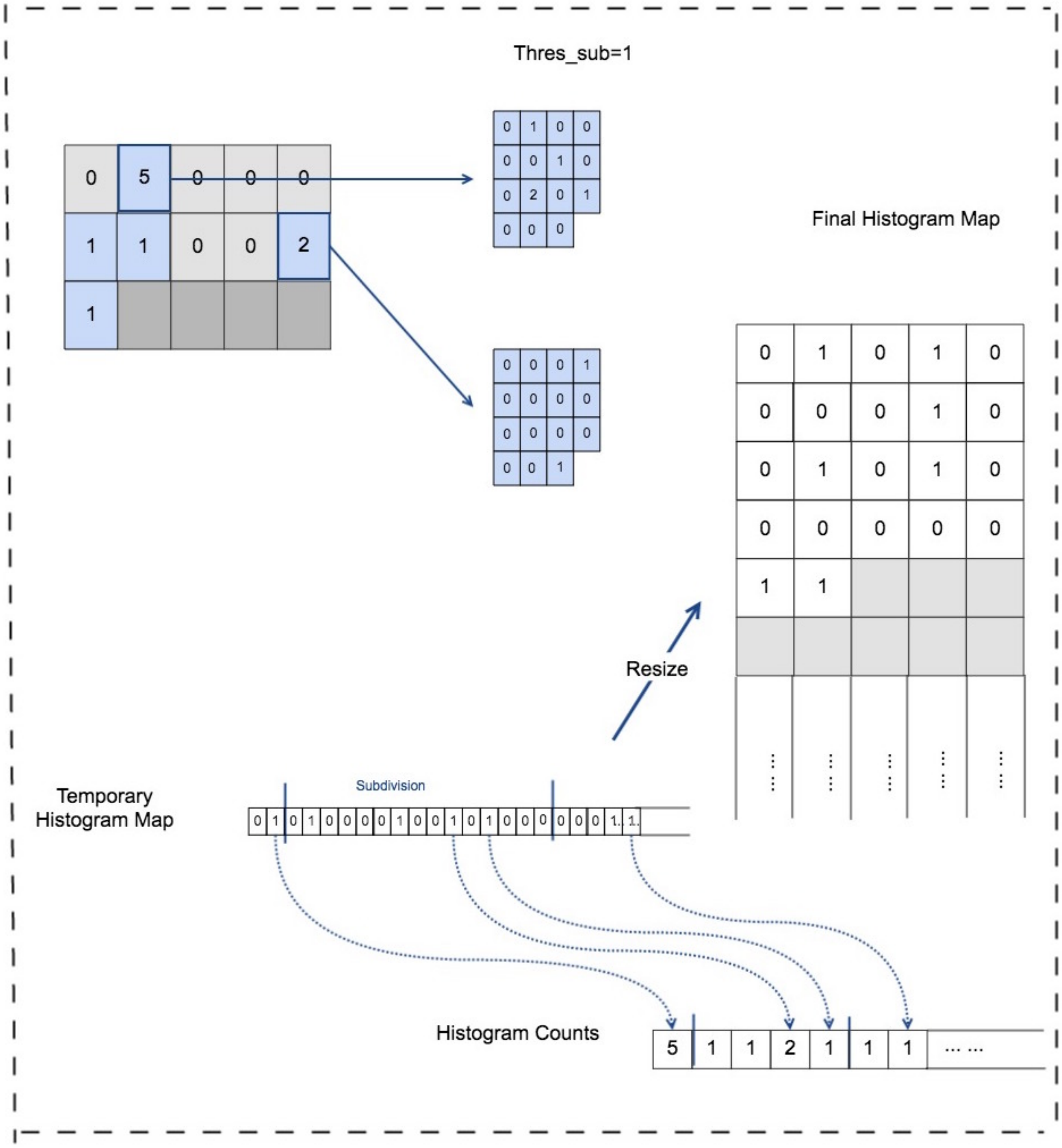


Figure 3.4: An example of location encoding using variable block-size.

W_{sb} and H_{sb} is the notations for the width and height of a subblock. The subdivision operation increases the size of the histogram map. To maintain the neighbor template for coding the histogram blocks using the context-based arithmetic coding, we keep the number of the column same as C_m while the number of the histogram rows are determined by:

$$R_m^{'} = \frac{S_m}{C_m} \tag{3.4}$$

S_m is the total volume of 0s and 1s to store the states for both large and small blocks. The states of the small blocks only needs to use $N_s - 1$ symbols to describe since the state of the last one can be predicted by the previously coded small blocks. The histogram map of the variable block-sized location coding scheme is fed into the context-based arithmetic coder. At the same time, the histogram count records the number of features whenever encoder scans to a non-empty block.

Decoder

The histogram map is first decoded by the context-based arithmetic decoder. Then, the decoder continues to reconstruct the grids laid on the image with noticing the numbers from the histogram counts. When the decoder sees a 1 in the histogram map, it would learn the feature number within this block by referring to the histogram counts. The subdivision scheme and its threshold is known both at the encoder and the decoder. Hence, the number of small blocks to better represent the descriptor location within this large block, N_s, is known at decoder as well. Once the decoder reads the end of following $N_s - 1$ symbols, it would automatically consider the next symbol as describing the state of the next large block. The general working flow is described as Fig 3.5. To finish the location quantization, the location of all descriptors

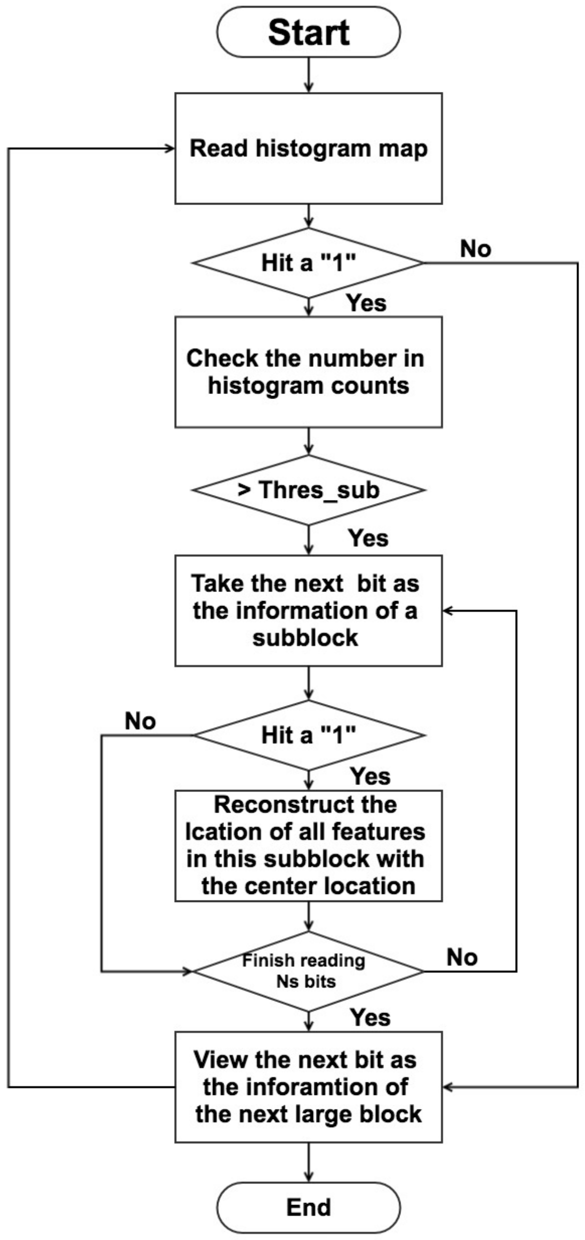

Start
Read histogram map
Hit a "1"
No
Yes
Check the number in histogram counts
> Thres_sub
Yes
Take the next bit as the information of a subblock
No
Hit a "1"
Yes
Reconstruct the lcation of all features in this subblock with the center location
Finish reading Ns bits
No
Yes
View the next bit as the inforamtion of the next large block
End

Figure 3.5: The workflow of a location histogram decoder

in the same large block are mapped to the central point.

Chapter 4

Experimental Results

In this chapter, we study the performance of the proposed feature coding scheme. In section 4.1, we give a brief overview of the dataset and the experiment setup. In section 4.2, we present different feature descriptor quantizers according to different number of descriptor with different lengths. We also study their effects on the video matching process. In section 4.3, we evaluate the feature location coding scheme, and we study the performance of different block size combinations. In the last section, we present the overall evaluation of the entire feature coding scheme which combines the best feature descriptor and location quantization schemes.

4.1 Experiment and Data Set Description

In this experiment, we use $4.6KBps/sec$ as the bitrate budget for the feature coding scheme, which is the lowest bitrate adopted in other CDVA analysis tasks. Since compressed descriptors usually demand more bitrate than that of their compressed locations, we allocate the bitrate budgets as follows,: $4.4KBps/sec$ for descriptors and

$0.2KBps/sec$ for location information. The evaluation of our feature coding scheme is performed with the MPEG-CDVA dataset, which is prepared for CDVA standardization [24]. It consists of more than 10 thousand of videos with different contents, *e.g.*, buildings, people, birds, books, etc. For the limited computational resources we have, the experiments are conducted on a small subset of randomly selected 360 videos from several categories: people, buildings, books and birds. To further lessen the computational burden, all videos are resized to the same frame resolution of 640×360. For the feature descriptor coding scheme, an image training dataset, 25K-MIRFLICKR database [19], is used to estimate the covariance matrix of the SIFT descriptor elements, which is used to find the K-L transformation matrix. This dataset contain more than 25000 images that cover almost all categories of objects, such as people, animals, buildings, etc. The average number of features extracted from each image is more than a thousand. After obtaining the transformation matrix, it is used to decorrelated the elements of all those extracted descriptors. Then, as described in Chap 2, empirical distributions are estimated for the elements of the transformed version of the SIFT descriptor. For the feature location coding scheme, to compute the bitrate used to compress feature location information, the context-based arithmetic encoder from JBIG standard is used. This JBIG coder works with the context template size of 10.

There are two performance measurements defined in CDVA-Evaluation Framework [25], namely True Positive Rate (TPR) and False Positive Rate (FPR). They have been used to assess the pairwise-matching performance. The two rates are given by:

$$\begin{aligned} TPR &= \frac{TP}{TP+FN}, \\ FPR &= \frac{FP}{TP+FN}, \end{aligned} \tag{4.1}$$

where TP is the number of true positives, FN is the number of false negatives, FP is the number of false positives, and TN is the number of true negatives. Each of the above two rates has to be evaluated using a list of pairs of videos. The first list is of matching videos, i.e., pairs from the same category, while the second is for non-matching pairs of videos, i.e., cross category pairs. The evaluation of the matching performance is based on The TPR for a FPR that is less than or equal to 1%.

4.2 Descriptor Coding Scheme Evaluation

The design of the variable level scalar quantizer needs to take in consideration three variables. These are the number of descriptors, the number of elements in a transformed descriptor (Length of the descriptor), and the number of quantization levels for each element. The problem seems to be complicated if we optimize the descriptor quantizer for the three variables simultaneously. Hence, we design the descriptor quantizer for 80, 150 and 200 descriptors. For each number of descriptors, we design the descriptor quantization using different descriptor length. The descriptor coding scheme is determined to be the quantizer that provides the best matching performance among all quantizers with different length of the descriptors while the bitrate is under the descriptor bitrate budget.

The different size of the persistent set is controlled by the local feature selection in Fig.1.1. Unlike the selection method described in [11], we control the number of descriptors by randomly selecting the persistent descriptors within the persistent set of a GOP. The maximum number of elements used for the descriptor quantizer is 40, since we see that most of energy is contained in the first 40 elements in Fig.2.2. Different quantizers are designed to provide the lowest distortion in Eq.2.4 according to different number of elements. The maximum number of levels used to quantize the elements in the transformed descriptor is 32. We cannot partition the empirical distribution of the elements more than 32 intervals.

4.2.1 The Relation Between Performance and The Number of Descriptors

In this section we provide the performance of applying the finest quantization to descriptors using different number of descriptors. We show that the effects of using more descriptors to represent a video frame on the matching performance.

Shown in Fig 4.1 is the matching performance of different number of descriptors using different number of quantized elements. We can see that the performance with the finest quantization of the elements increases as the number of descriptors increases. This is expected, since the more descriptors are used to represent a frame in videos, the better description of this frame is obtained. We find that more accurate representation of the frames in the video improves the video pairwise matching performance.

4.2.2 Design of The Feature Descriptor Encoder

Given the data budget, the finest quantization cannot be used for a large set of descriptors, since its bitrate would exceed the limitation. Hence, we consider designing

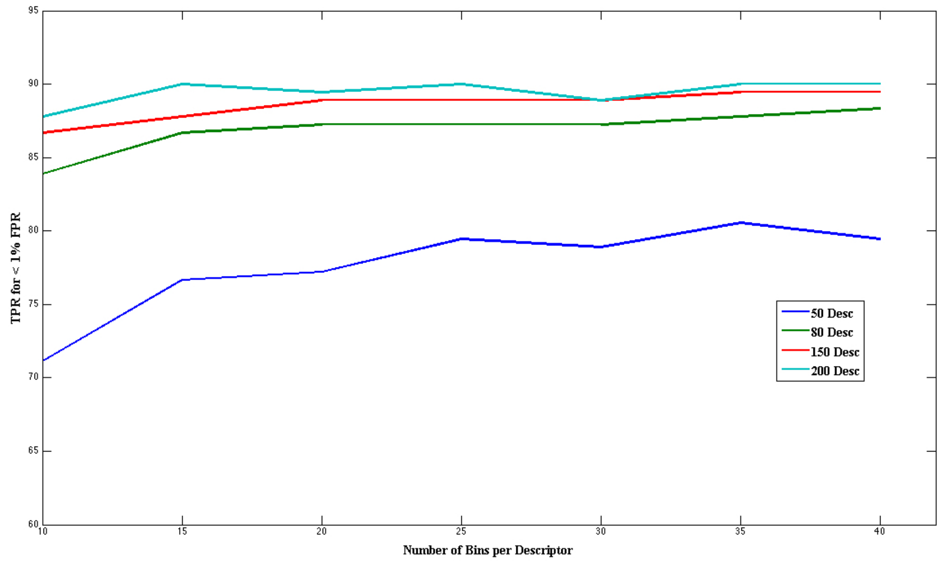
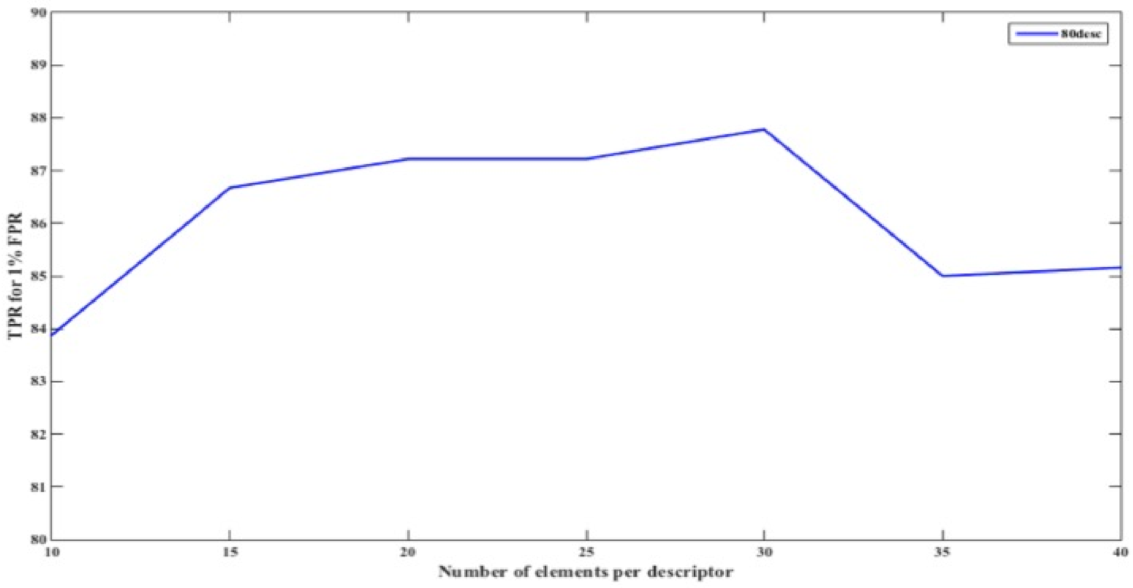
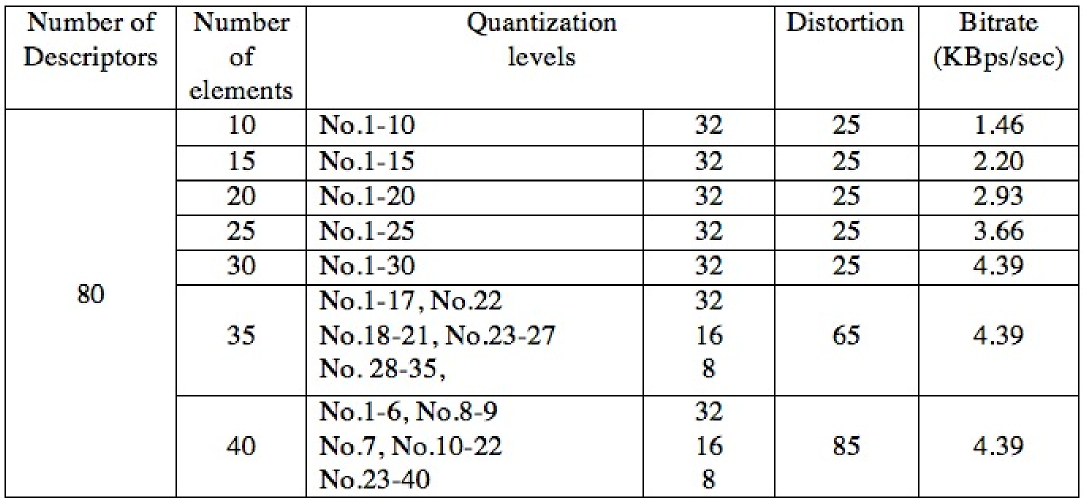


Figure 4.1: The TPR corresponding to the number of elements.

a variable level descriptor quantizer.

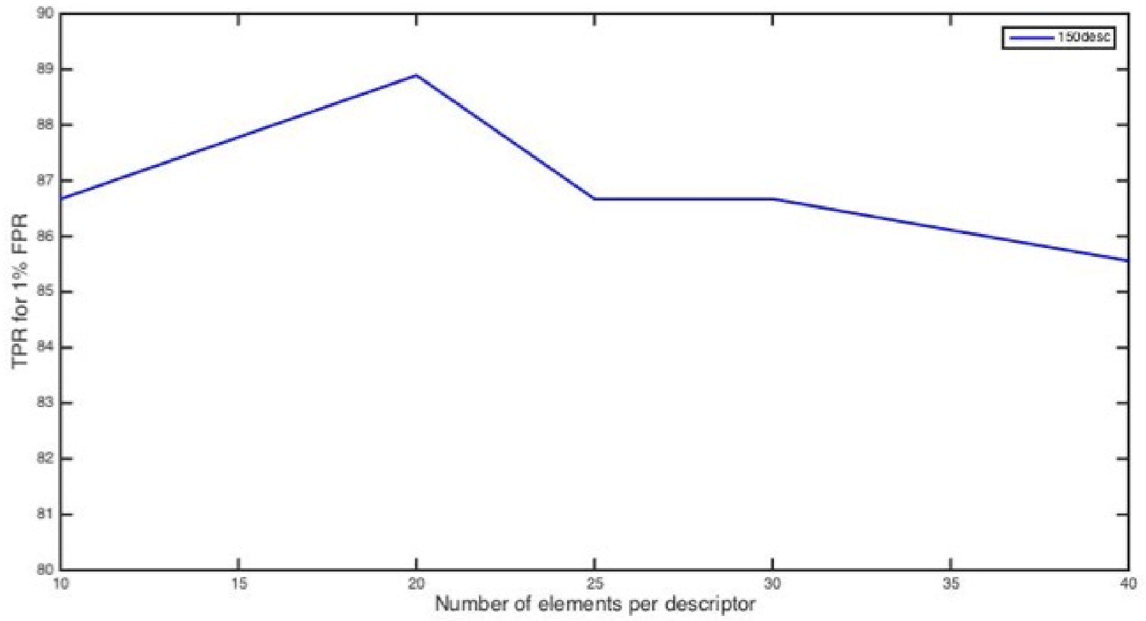
Shown in Fig 4.2, Fig 4.3, and Fig 4.4 are the descriptor quantizer matching performance of 80, 150, and 200 descriptors using different length of descriptors. Each of them are designed with the minimum distortion according to different number of elements. To illustrate, we take the performance of 150 descriptors in Fig 4.3 as an example. We see that the matching performance increases when the bitrate used increases, which means that the more bitrate is used to represent a frame, the higher matching performance can be obtained. Moreover, when the bitrate reaches the budgets, the higher distortion results in the poorer performance. In other words, the variable level descriptor quantizer with the minimal distortion always guarantees the best video matching performance, as long as the bitrate used to represent the feature descriptors reaches the data budget. However, the quantizers with bitrates smaller than the bitrate budget usually have lower distortions, yet they cannot achieve good

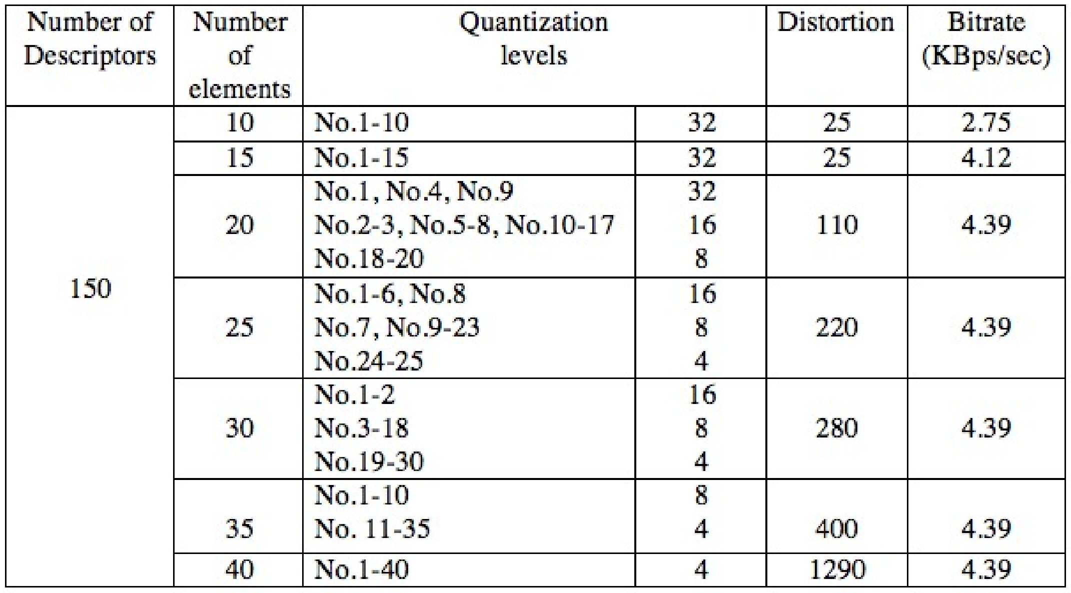




Number of Descriptors	Number of elements	Quantization levels		Distortion	Bitrate (KBps/sec)
80	10	No.1-10	32	25	1.46
	15	No.1-15	32	25	2.20
	20	No.1-20	32	25	2.93
	25	No.1-25	32	25	3.66
	30	No.1-30	32	25	4.39
	35	No.1-17, No.22 No.18-21, No.23-27 No. 28-35,	32 16 8	65	4.39
	40	No.1-6, No.8-9 No.7, No.10-22 No.23-40	32 16 8	85	4.39

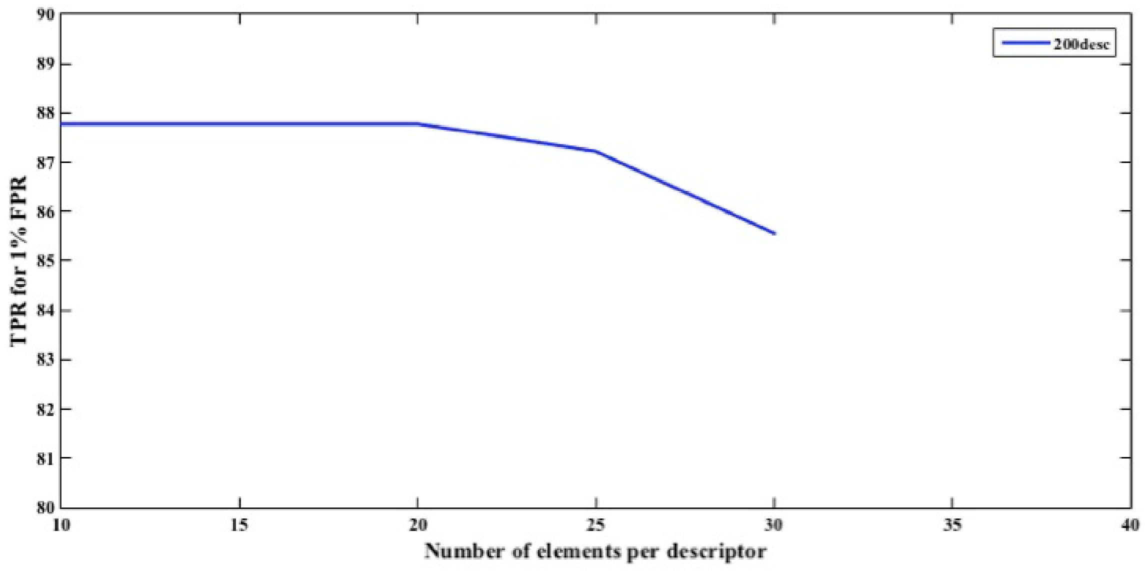
Figure 4.2: The TPR corresponding to the number of elements using 80 descriptors.

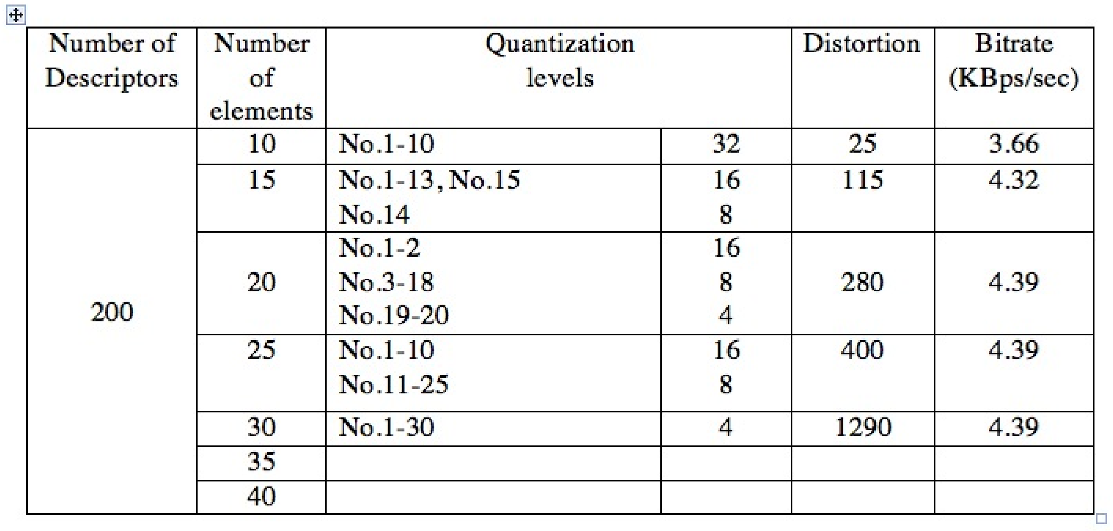




Number of Descriptors	Number of elements	Quantization levels		Distortion	Bitrate (KBps/sec)
150	10	No.1-10	32	25	2.75
	15	No.1-15	32	25	4.12
	20	No.1, No.4, No.9 No.2-3, No.5-8, No.10-17 No.18-20	32 16 8	110	4.39
	25	No.1-6, No.8 No.7, No.9-23 No.24-25	16 8 4	220	4.39
	30	No.1-2 No.3-18 No.19-30	16 8 4	280	4.39
	35	No.1-10 No. 11-35	8 4	400	4.39
	40	No.1-40	4	1290	4.39

Figure 4.3: The TPR corresponding to the number of elements using 150 descriptors.





Number of Descriptors	Number of elements	Quantization levels		Distortion	Bitrate (KBps/sec)
200	10	No.1-10	32	25	3.66
	15	No.1-13, No.15	16	115	4.32
		No.14	8		
	20	No.1-2	16	280	4.39
		No.3-18	8		
		No.19-20	4		
	25	No.1-10	16	400	4.39
		No.11-25	8		
	30	No.1-30	4	1290	4.39
	35				
	40				

Figure 4.4: The TPR corresponding to the number of elements using 200 descriptors.

performance since they do not fully use the bitrate budget (see the bitrate of the descriptor quantizers for 10 or 15 elements in Fig.4.3). The performance of these descriptor quantizers cannot be better than the performance of the quantizer which uses the maximum bitrate permitted with the lowest distortion possible. To this end, the variable level descriptor quantization for different number of descriptors is found to be the one with the lowest distortion and the maximal bitrate used, shown in Table4.1.

Number of Descriptors	Number of elements	Bitrate	Distortion	Quantizer	
80	30	4.39	25	No.1 - 30	32 level
150	20	4.39	110	No.1, 4, 9	32 level
				No.2 - 3, 5 - 8, 10 - 17	16 level
				No.18 - 20	8 level
200	20	4.39	280	No.1 - 2	16 level
				No.3 - 18	8 level
				No.19 - 20	4 level

Table 4.1: The variable level descriptor quantizer designed for different number of descriptors

However, the discussion above does not hold completely to the case of 200 descriptors shown in Fig.4.4. We can see that the matching performance of using 10 and 15 elements is the same while the bitrate used for 10 elements does not fully meet the target bitrate. By comparing the quantizers for 10 and 15 elements, the quantizer for 15 elements is much coarser than that for 10 elements, since design of the quantizer is restricted by the bitrate limitation. It indicates that it is important to quantize the first 10 elements finner than the following elements.

4.2.3 Performance of The Feature Descriptor Coding Using Different Number of Descriptors

In this section, we discuss the proper number of descriptor used to represent a video frame given a bitrate budget. We compare the matching performance for different numbers of descriptors using the corresponding descriptor quantizers shown in Table 4.1.

Shown in Table 4.2 is the matching performance of different number of descriptors. All three descriptor quantizers use the same bitrate to represent the descriptors. The difference is the distortion. The quantizer for 80 descriptors has the lowest distortion, 25, while the distortion for 200 descriptors is the highest, 280. However, the performance of these two descriptor quantizers is not as good as that of 150 descriptors, which means the elements in the transformed descriptor do not need to be quantized as precise as possible. Instead, the video matching task tolerates a relatively less accurate representation of the feature descriptors as long as more descriptors are added. However, if the distortion is continuously increased, this higher distortion results in a poorer performance.

Number of Descriptors	TPR
80	87.78%
150	88.89%
200	87.78%

Table 4.2: The matching performance using different number of descriptors

4.3 The Evaluation of the Location Coding Scheme

The objective of using variable block-sized location scheme is to keep as much useful location information as possible while reduce its bitrate. As we know, the more persistent descriptors are selected in the keyframe, the higher is the number of descriptors falling in the same block. Besides, in [17], they show that using a large size of blocks introduces higher error rates in the image matching than using small size of blocks. Since the location of all features that are in one block is quantized to the same value, the inaccurate approximation of feature locations occurs if we use a large size blocks. Moreover, it also has a negative impact on matching performance because the GCC technology relies heavily on the reconstructed feature locations. Hence, in our location coding scheme, we consider how to set the subdivision threshold properly. In this way, we quantize the area where the features are dense precisely while only lose some location information in areas with sparse features. On the other hand, as there are several sizes of the blocks, we can create different combinations of block sizes. Therefore, we evaluate the coding performance for different block combinations. By setting a proper subdivision threshold and the the choice of the block combination, we show that our location coding scheme can provide a reliable matching performance with a low bitrate needed.

The Proper Subdivision Threshold and The Block Size Combination

According to [17], there are four sizes of squared blocks, $32 \times 32, 16 \times 16, 8 \times 8, 4 \times 4$. Blocks with sizes of 32 and 16 are considered as large blocks while blocks with sizes 4 and 8 are used as small blocks. Hence, we come up with four combinations as well, shown in Table 4.3. As in the example of 16 *sub* 4, we use the blocks of size 16×16

as the large blocks while use the blocks of size 4×4 to subdivide a large block to quantize the feature location more accurately.

Block combination
16 *sub* 4
16 *sub* 8
32 *sub* 4
32 *sub* 8

Table 4.3: The four combinations of different block sizes

4.3.1 The Threshold for Block Subdivision

Setting a proper subdivision threshold is the first consideration to design an efficient variable block-size location coding scheme. We first compute the average number of descriptors in the non-empty large blocks. The mean value represents the most common status in all non-empty histogram blocks. The greater amount of descriptors than the average in one block means the location of those descriptors requires finer quantization. In Table 4.4, we show the average number of descriptors of the two large histogram blocks using the following equation:

$$M_d = \frac{N_d}{S_m^1}, \tag{4.2}$$

where N_d is the number of features in a frame, and S_m^1 is the number of non-empty blocks. As using different operating points, the subdivision threshold for each combination is determined by the video matching performance using the nearest two integers of the average. On the other hand, setting an improper threshold may causes

Number of Descriptors	32×32	16×16
80	1.424	1.141
150	1.788	1.223
200	2.271	1.341

Table 4.4: The average number of descriptors in one non-empty block for different modes

Number of	Block	Bitrates (Bytes/GOP)		TPR	
descriptors	combination	Thres=1	Thres=2	Thres=1	Thres=2
150	16 *sub* 4	66.85	63.17	86.67%	86.67%

Table 4.5: The performance of varying the set of thresholds

higher bitrate without gaining much in matching. Shown in Table 4.5 is a sample of the matching performance and bitrates of the location coding scheme using two thresholds with a combination of 16 *sub* 4. We can see that transmitting the feature locations using higher bitrates cannot guarantee the improvement in video matching performance. We pick the thresholds that produce a comparable matching TPR while using fewer bits. As in the case of Table 4.5, the subdivision threshold is set to be 2.

We also learn the subdivision thresholds for all the blocks combinations using different number of descriptors, which are shown in Table 4.6:

4.3.2 Performance of Variable Block-size Location Coding

To show how much bitrate can be saved by using our location coding scheme, we compare the variable block-sized location coding scheme with the histogram location coding method using the uniform block sizes. To be able to compare these two

Numbers of Descriptors	Block combinations	Subdivision Threshold
80	16 *sub* 4	1
	16 *sub* 8	1
	32 *sub* 4	2
	32 *sub* 8	1
150	16 *sub* 4	2
	16 *sub* 8	2
	32 *sub* 4	1
	32 *sub* 8	1
200	16 *sub* 4	2
	16 *sub* 8	2
	32 *sub* 4	2
	32 *sub* 8	2

Table 4.6: The subdivision thresholds for different number of descriptors

coding schemes, we use the same persistent descriptor selection method in Fig. 1.1 and measure the bitrate and video matching performance for both cases.

Table 4.7 is the coding performance for different block sizes using different number of descriptors. We use the case of 80 descriptors as the example to illustrate the advantages of our location coding scheme. We can see that the matching performance of the variable block-size location coding scheme using every block combination is better or equivalent to the highest TPR using CDVS location coding method. For the variable block-size location coding scheme, the best performance is produced using the block combination 16 *sub* 8. Accordingly, the highest TPR using uniform block size location coding method is produced with the block size of 8 pixels. However, the bitrates used in our coding scheme are significantly less than those used in the method of uniform block size. When both two methods reach the best matching performance, the bitrate used in our location coding scheme is reduced in 39.1%. Meanwhile, comparing the uniform block size location coding scheme using the block size of 4 with our location coding scheme using the block combination 32 *sub* 4, the bitrate is decreased by 56.8% decreased while maintaining the same video matching performance. As for the cases of 150 and 200 descriptors, we can also achieve a comparable matching performance while the bitrate used is much lower.

In the end of the feature location coding evaluation, we set the block combination for each operating point. Since we put a strict bitrate limitation to the entire feature coding scheme, we favor the quantization scheme that generates the lowest bitrate but with comparable performance to that of the other schemes. We have found that the block combination of 32 *sub* 8 can achieve a similar matching performance to other combinations while the bitrate used is much lower than the others. Thus, we

Number of descriptors	Block size	Bitrate (Bytes/GOP)	TPR
80	Uniform 4	102.63	84.17%
	Uniform 8	103.03	83.83%
	16 *sub* 4	62.84	83.16%
	16 *sub* 8	62.75	84.17%
	32 *sub* 4	44.23	83.83%
	32 *sub* 8	44.34	83.16%
150	Uniform 4	104.68	87.22%
	Uniform 8	104.76	86.67%
	16 *sub* 4	63.98	86.67%
	16 *sub* 8	63.83	86.67%
	32 *sub* 4	61.45	87.22%
	32 *sub* 8	51.70	86.67%
200	Uniform 4	105.85	87.88%
	Uniform 8	105.69	87.88%
	16 *sub* 4	65.18	87.22%
	16 *sub* 8	64.29	86.67%
	32 *sub* 4	61.25	86.67%
	32 *sub* 8	54.06	86.67%

Table 4.7: The performance for using different location schemes with different block sizes

suggest to use the block combination of 32 *sub* 8 in the variable block-sized location coding scheme, but the subdivision threshold is changed when operating on different number of descriptors, shown in Table 4.7.

4.4 Performance of Overall Coding Scheme

To evaluate the overall feature coding scheme, we combine the best descriptor and location information quantizers, discussed in Sec.4.2 and 4.3. The matching performance using different number of descriptors is shown in Table 4.8. The matching performance of the overall feature coding scheme using 150 descriptors is the best, which implies that it is unnecessary to use as much features as possible to represent a video frame. For the case of 80 descriptors, we apply the finest quantizer to descriptors. As more descriptors are added to represent a frame in the case of 150, the matching performance using a relatively coarser descriptor quantizer is better. However, if the number of descriptor is continuously increasing while the quantization accuracy is sacrificed, this results in the degradation of the matching performance.

Number of Descriptors	Bitrate (KBps/sec)	TPR
80	4.51	84.44%
150	4.54	86.11%
200	4.54	85.56%

Table 4.8: The performance of the feature coding scheme using different number of descriptors

By comparing the matching performance in Table 4.2 and Table 4.8, we see the effect of introducing location quantization to the feature coding system. The matching performance after location quantization decreases compared to that without location quantization. The effect of location quantization on the matching performance depends on the number of descriptor generated. The matching performance is dropped by 3.34% and 2.78% using 80 and 150 descriptors respectively compared to the matching performance of only applying descriptor quantization, which means feature location information is more important and requires a fine quantization when using a small set of descriptors to match videos. As the number of descriptors increases, a relatively poor representation of feature locations does not affect much to the video matching task.

Chapter 5

Conclusion

In this thesis, we present a new feature coding scheme for videos. For descriptors, we design a variable level scalar quantizer using empirical distributions of K-L transformed SIFT descriptors. We have achieved the low bitrate by only quantizing the first few elements which contain most of energy. We have learned that a good representation of a frame improves the matching performance. In particular, for a given bitrate budget, variable quantization of the elements in the transformed descriptors results in a low distortion, which, in turn, leads to a better video pairwise matching performance. We have shown that for a target bitrate, there is a nonlinear relation between the matching performance and the number of descriptor used to represent a frame; after a specific number of descriptors, the quantization becomes relatively coarse as more descriptors are added, causing the performance to degrade. For the feature locations, we design a variable block size location coding scheme. Compared to the coding scheme using uniform block sizes, we have reduced the bitrate for coding feature location information nearly by half with a negligible video matching error rate introduced. We have combined the best descriptor and feature location coding schemes to form the overall system, which has achieved excellent video pairwise matching performance given a target bitrate. However, to suit our feature coding scheme to several bitrate budgets, more experiments must be done.

Bibliography

[1] G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pp. 427–434, ACM, 2008.

[2] D. Ritendra, J. Li, and W. J. Z, "Content-based image retrieval: approaches and trends of the new age," in *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pp. 253–262, ACM, 2005.

[3] G. Takacs, V. Chandrasekhar, B. Girod, and R. Grzeszczuk, "Feature tracking for mobile augmented reality using video coder motion vectors," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 1–4, IEEE Computer Society, 2007.

[4] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1615–1630, Oct. 2005.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[6] C. Cai, P. Wang, and Y. h. Liang, "Fast image stitching based on improved surf," pp. 411–416, May. 2016.

[7] Z. Alaitz and P. Xavier, "Impact of lossy compression on mapping crop areas from remote sensing," *International journal of remote sensing*, vol. 34, no. 8, pp. 2796–2813, 2013.

[8] C. Yeo, P. Ahammad, and K. Ramchandran, "Rate-efficient visual correspondences using random projections," in *15th IEEE International Conference on Image Processing*, pp. 217–220, Oct. 2008.

[9] C. Vijay, T. Gabriel, C. David, T. Sam, S. Jatinder, and G. Bernd, "Transform coding of image feature descriptors," in *IST/SPIE Electronic Imaging*, International Society for Optics and Photonics, 2009.

[10] M. Johnson, "Generalized descriptor compression for storage and matching.," 2010.

[11] M. Alrabeiah, T. Yin, Y. Qiao, L. Li, J. Chen, D. He, and Y. Wang, "Persistent descriptors: A new descriptor compression approach for video analysis," Feb. 2017.

[12] Y. Ke and R. Sukthankar, "Pca-sift: a more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–506–II–513 Vol.2, Jun. 2004.

[13] G. Hua, M. Brown, and S. Winder, "Discriminant embedding for local image descriptors," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, Oct. 2007.

[14] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, Jun. 2008.

[15] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod, "Chog: Compressed histogram of gradients a low bit-rate feature descriptor," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2504–2511, Jun. 2009.

[16] Z. Wang, S. Tsai, G. Cordara, L.-Y. Duan, R. G. Tiejun Huang, W. Gao, and B. Girod, "Cdvs core experiement 3: Standford/peking/huawei contribution," Jul. 2012.

[17] T. S. C. David, T. Gabriel, C. Vijay, S. J. P, and G. Bernd, "Location coding for mobile image retrieval," in *Proceedings of the 5th International ICST Mobile Multimedia Communications Conference*, Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2009.

[18] T. S. S, C. David, T. Gabriel, C. Vijay, M. Mina, G. Radek, and G. Bernd, "Improved coding for image feature location information," in *SPIE Optical Engineering+ Applications*, International Society for Optics and Photonics, 2012.

[19] M. J. Huiskes and M. S. Lew, "The mir flickr retrieval evaluation." in MIR '08:

Proceedings of the ACM International Conference on Multimedia In- formation Retrieval, 2008. New York.

[20] L. Y. Duan, V. Chandrasekhar, J. Chen, J. Lin, Z. Wang, T. Huang, B. Girod, and W. Gao, "Overview of the mpeg-cdvs standard," *IEEE Transactions on Image Processing*, vol. 25, pp. 179–194, Jan. 2016.

[21] S. Suvorova and J. Schroeder, "Automated target recognition using the karhunen–loeve transform with invariance," *Digital Signal Processing*, vol. 12, no. 2-3, pp. 295–306, 2002.

[22] M. Shinwari, A. Youssef, and W. Hamouda, "A water-filling based scheduling algorithm for the smart grid," *IEEE Transactions on Smart Grid*, vol. 3, pp. 710–719, Jun. 2012.

[23] P. Scheunders, "A genetic lloyd-max image quantization algorithm," *Pattern Recognition Letters*, vol. 17, no. 5, pp. 547–556, 1996.

[24] I. JTCI/SC29/WG11/N15339, "Cfp for compact descriptors for video analysis search and retrieval," 2015.

[25] I. JTCI/SC29/WG11/N15338, "Evaluation framework for compact descriptors for video analysis- search and retrieval," 2015.