Technische Universität München
Lehrstuhl für Medientechnik

Feature-preserving image and video compression

Jianshu Chao, M. Sc.

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Die Dissertation wurde am 12.11.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 22.02.2016 angenommen.

To my beloved parents and Munich

Abstract

Nowadays, mobile devices equipped with cameras are ubiquitous and wireless communication networks are widely accessible. Captured images or videos can be transmitted to a server from anywhere at any time. Owing to limited network resources, e.g., when many consumers compete for the network resources or the wireless link speed is low, images or videos have to be compressed at low bitrates for transmission. Conventional image and video compression standards assume that humans are the information receivers. Therefore, they maximize the quality of the compressed content by exploiting known properties of the human visual perception system. However, lately, in an increasing number of applications, such compressed images or videos are fed into computer vision algorithms for analysis at a server, such as mobile visual search, object retrieval and tracking, and surveillance. This motivates us to control the encoding process in such a way that important features are preserved rather than optimizing it for the human visual system. This thesis presents several works on preserving the most important features when compressing images or videos at relatively low bitrates in a standard-compatible manner.

The feature quality is affected significantly by compression artifacts at low bitrates. To design approaches to preserve more features, the properties of typical image features are investigated, and it is found that 1) large-scale features are more robust than small-scale features, 2) keypoint detection is more important than descriptor calculation, and 3) high frequencies of an image are less important for detecting scale-space-based features. Considering these feature properties, various schemes are proposed in this thesis. First, still images are considered and encoded using an image compression standard, e.g., JPEG. Several feature-preservation approaches are proposed based on the aforementioned feature properties: a) deblocking filters are applied to low-quality images, b) a larger number of bits is assigned to small-scale features using a novel rate-distortion model, and c) a novel JPEG quantization table is designed for scale-space-based features. The results show that the proposed approaches lead to higher matching performance when compared with the default JPEG image compression standard. Then, the approaches for images are extended to video compression. Similarly, rate-distortion models are developed to encode videos using the H.264/AVC standard. In addition, in prac-

tical video communications, compressed videos are required to meet the target constraints of the communication channels for transmission. Therefore, a novel rate control framework is proposed for feature preservation in H.264/AVC video compression. The proposed framework achieves the target bitrates, while preserving more features compared with the default rate control scheme. Furthermore, given that keypoint detection is more critical, encoding the original keypoints from uncompressed images or video frames is proposed to improve feature descriptor quality. The experiments are performed based on the H.265/HEVC standard, and the improved feature-matching performance and the retrieval performance justify the keypoint encoding idea.

The proposed feature-preserving image and video compression schemes offer standard compatible image and video bitstreams, which hence can be decoded by standard decoders.

Kurzfassung

Mit Kameras ausgestattete Mobilgeräte sind heute allgegenwärtig und drahtlose Kommunikationsnetze allgemein zugänglich. Die mit den Geräten aufgenommenen Bilder oder Videos können an einen Server von überall zu jeder Zeit übertragen werden. Aufgrund der begrenzten Netzwerkressourcen, zum Beispiel wenn viele Verbraucher um die Netzwerkressourcen konkurrieren oder die drahtlose Verbindungsgeschwindigkeit niedrig ist, müssen Bilder oder Videos für niedrige Bitraten zur Übertragung komprimiert werden. Herkömmliche Bild- und Videokompressionsstandards nehmen an, dass Menschen die Informationsempfänger sind. Daher wird die Qualität der komprimierten Inhalte unter Ausnutzung bekannter Eigenschaften des menschlichen visuellen Wahrnehmungssystems optimiert. Jedoch werden die komprimierten Bilder oder Videos in letzter Zeit auch häufig einem Computer Vision Algorithmus zur Analyse auf einem Server, wie beispielsweise für mobile visuelle Suche, Objekterkennung und -verfolgung zugeführt. Das veranlasst uns, den Encodierungsprozess so zu kontrollieren, dass für die Computer Vision wichtige Merkmale erhalten bleiben statt den Inhalt für das menschliche visuelle System zu optimieren. Diese Arbeit präsentiert mehrere Ansätze für die Erhaltung der wichtigsten Merkmale, wenn Bilder oder Videos bei relativ niedrigen Bitraten in einer standard-kompatiblen Weise komprimiert werden.

Die Merkmalsqualität wird bei niedrigen Bitraten signifikant durch Kompressionsartefakte beeinflusst. Um Ansätze zur Erhaltung von mehr Merkmalen zu entwerfen, wurden die Eigenschaften der typischen Bildmerkmale untersucht. Es wurde festgestellt, dass 1) großflächige Merkmale robuster sind als kleine, 2) Merkmalsdetektionwichtiger ist als Deskriptorberechnung, und 3) hohe Frequenzen eines Bildes zur Erfassung Scale-Space-basierter Merkmale weniger wichtig sind. Unter Berücksichtigung dieser Merkmalseigenschaften werden in dieser Arbeit verschiedene Ansätze für die merkmalserhaltende Bild- und Videokompression vorgeschlagen. Zunächst werden Standbilder unter Verwendung eines Bildkompressionsstandards, beispielsweise JPEG, codiert. Basierend auf den vorgestellten Eigenschaften werden verschiedene Herangehensweisen vorgeschlagen: a) Deblocking-Filter werden auf niederqualitative Bilder angewandt, b) kleinflächige Merkmale werden mit einem neuen Raten-Verzerrungs-Modell mehr Bits zugewiesen, und c) eine neue JPEG-Quantisierungstabelle für Scale-Space-basierte Merkmale wurde entworfen. Die Ergebnisse zeigen, dass die vorgeschlagenen Ansätze verglichen mit dem unveränderten JPEG-Standard zu besseren Matching-Ergebnissen führen. Anschließend werden die Ansätze für die Bilder auf Video-Kompression erweitert. Auf ähnliche Weise werden Raten-Verzerrungs-Modelle, um Videos mit dem H.264/AVC-Standard zu kodieren, entwickelt. Darüber hinaus sind in der realen Video-Kommunikation komprimierte Videos erforderlich, um die eingeschränkten Ziele der Kommunikationskanäle für die Übertragung einzuhalten. Daher wird eine neue Ratenkontrolle für die Merkmalserhaltung in H.264/AVC Video-Kompression vorgestellt. Die vorgeschlagene Ratenkontrolle erreicht die Zielbitraten unter Beibehaltung von mehr Merkmalen im Vergleich zur Standard-Ratenkontrolle. Des Weiteren, da die Merkmalspunktdetektion empfindlicher als die Deskriptorberechnung ist, wird vorgeschlagen, die ursprünglichen Merkmalspunkte von unkomprimierten Bildern oder Videobildern als Seiteninformation zu übertragen, um die Qualität der Merkmalsbeschreibung zu verbessern. Die Versuche wurden auf der Grundlage des H.265/HEVC Standard durchgeführt, wobei das verbesserte Merkmals-Matching und die Wiedergewinnungsleistung diese Merkmalspunktcodierungsidee rechtfertigen.

Die vorgeschlagenen merkmalserhaltenden Bild- und Videokompressionsverfahren erzeugen standardkompatible Bild- und Video-Bitströme, die durch Standard-Decoder decodiert werden können.

Acknowledgements

I would like to give my sincere appreciation to people who have contributed and assisted my research at the Chair of Media Technology and to those who have helped, supported, or encouraged me during these years.

First of all, I must give great thanks to my supervisor, Professor Eckehard Steinbach, for giving me an opportunity to come to Germany and assigning an exciting topic to work on. His kind personality, guidance, ideas, and advice have helped me reach the completion of this thesis. During these years, besides the technical knowledge that I have acquired, I have also learned many important skills for research and my future career, e.g., working independently, getting along with other people, analyzing challenges, finding effective solutions, and presenting my work. These skills that I learned from my supervisor and other people give me much confidence to face future work and life.

Also, I would like to thank Professor Matteo Cesana from Politecnico di Milano for serving as my second examiner and Professor Bernhard Seeber for heading the examination committee.

I am especially grateful to the China Scholarship Council for financially supporting me for four years. Thanks should go to the administrative staff in China and Germany for arranging many things for me.

I would like to thank Professor Lexing Xie of the Research School of Computer Science at the Australian National University. She hosted me warmly as a visiting researcher and made my stay enjoyable. She pointed out a new direction, following which I found other important solutions to my research problem. Indeed, I had a great experience in Australia.

My sincere appreciation should definitely go to the previous and current colleagues and the administrative staff at the Chair of Media Technology. I would like to thank Hu Chen, Robert Huitl, Anas Al-Nuaimi, Georg Schroth, and Damien Schroeder for helping or collaborating with me in my research; Florian Schweiger, Dominik van Opdenbosch, Dmytro Bobkov for correcting my papers; the colleagues that I have had great fortune to interact with: Yang Peng, Fan Zhang, Xiao Xu, Rahul Chaudhari, Burak Cizmeci, Prof. Liang Zhou, Ali El Essaili, Christoph Bachhuber, Jingyi Xu, and all other colleagues at our chair; and the visiting colleagues that I was lucky enough to meet. In particular, I would like to express my sincere thanks to Dr. Martin Maier, Ingrid Jamrath, Gabriele Kohl, and Simon Krapf for their administrative help and support.

Last but not least, I owe my parents a debt of gratitude for their love. They have been very supportive over the years, although they are far away from me. So, I would like to dedicate this thesis to my beloved parents.

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Description	Definition
AKM	Approximate K-Means	page 63
AP	Average Precision	page 22
API	Application Programming Interface	page 2
AVC	Advanced Video Coding	page 4
BoF	Bag-of-Feature	page 21
BRIEF	Binary Robust Independent Elementary Features	page 13
BRISK	Binary Robust Invariant Scalable Keypoints	page 8
CBIR	Content-Based Image Retrieval	page 2
CDVS	Compact Descriptors for Visual Search	page 15
DCT	Discrete Cosine Transform	page 32
DMOS	Differential Mean Opinion Score	page 81
DoG	Difference-of-Gaussian	page 9
DPCM	Differential Pulse Code Modulation	page 49
DWT	Discrete Wavelet Transform	page 121
FAST	Features from Accelerated Segment Test	page 8
FREAK	Fast REtinA Keypoint	page 13
FV	Fisher Vector	page 21
GLOH	Gradient Location and Orientation Histogram	page 13
GOP	Group of Pictures	page 53
GPS	Global Positioning System	page 1
GMM	Gaussian Mixture Model	page 21
GNSS	Global Navigation Satellite System	page 2
HEVC	High Efficiency Video Coding	page 4
HRD	Hypothetical Reference Decoder	page 71
HVS	Human Visual System	page 3
JPEG	Joint Photographic Experts Group	page 4
KAZE	A feature operates completely in a nonlinear scale space	page 8
LIOP	Local Intensity Order Pattern	page 13
LoG	Laplacian-of-Gaussian	page 9

Abbreviation	Description	Definition
LTE	Long-Term Evolution	page 2
MAD	Mean Average Difference	page 71
MAR	Mobile Augmented Reality	page 18
mAP	mean Average Precision	page 22
MB	Macroblock	page 31
MFD	Medial Feature Detector	page 8
MROGH	A rotationally invariant descriptor	page 13
MSER	Maximally Stable Extremal Regions	page 8
MVS	Mobile Visual Search	page 1
NN	Nearest Neighbor	page 19
NNDR	Nearest Neighbor Distance Ratio	page 19
PAO	Precision at One	page 110
PCA-SIFT	Principal Components Analysis based SIFT	page 13
PSNR	Peak Signal-to-Noise Ratio	page 17
QP	Quantization Parameter	page 43
R-D	Rate-Distortion	page 4
RDO	Rate-Distortion Optimization	page 59
RANSAC	RANdom SAmple Consensus	page 20
RLC	Run Length Coding	page 49
SAMVIQ	Subjective Assessment of Multimedia Video Quality	page 81
SDK	Software Development Kit	page 2
SIFT	Scale Invariant Feature Transform	page 2
SSIM	Structure SIMilarity	page 17
STAR	Based on Center Surrounded Extrema detector	page 8
SURF	Speeded Up Rapid Features	page 2
VLAD	Vector of Locally Aggregated Descriptors	page 21
VW	Visual Word	page 21
WαSH	Weighted α-SHapes	page 8
Wi-Fi	Wireless Fidelity	page 2

Chapter 1

Introduction

The field of computer vision was established in the early 1970s [Sze11], and since then many important applications have been studied. For instance, in medical image processing, researchers use computer vision algorithms to detect malignant changes or classify tissues from medical images. Surveillance systems monitor the environment and warn the user about relevant changes in it. Object recognition systems detect or recognize objects of interest. Augmented reality applications enhance a user's perception of reality by integrating virtual objects or information into real-world contexts. Autonomous vehicles use buildings or landmarks to produce maps of the surroundings. In fact, the applications that rely on computer vision algorithms are too numerous to list here.

1.1 Computer vision meets mobile communication

Computer vision algorithms can be run on a powerful computer to perform specific tasks. However, nowadays images and videos are increasingly being captured using mobile devices and transmitted to a server via a communication network for analysis. In this client-server architecture, distributed and cooperative visual processing is required. We call this concept *computer vision meets mobile communication*. Typically, a mobile client sends processed data to a server, and the server executes computer vision algorithms to extract desired information. For example, a video surveillance system records an image or a video and sends a compressed version of it to a server for analysis to find suspicious persons or events. When a vehicle is moving in an area with poor GPS signal, a camera-based driver assistance system in the vehicle can send an image or a video to the cloud to get an estimation of its location. Many emerging applications are based on computer vision meets mobile communication. Specifically, a topic called mobile visual search (MVS) is being studied widely. An MVS system compares an image or video captured by a mobile device against a database of images or videos on a server to find relevant information. The server typically uses a content-based image retrieval (CBIR) engine to retrieve the desired information and sends it back to the mobile device. Various applications employ MVS. For instance, one can capture a snapshot of a movie poster to watch the movie trailer and read user comments in order to decide whether to buy a ticket. When walking in front of a famous landmark, one can take a picture of it and be presented with a website detailing its history. When encountering an interesting product (e.g., a book, clothes, or a bottle of wine) on the go or in a shopping mall, one can take a picture of it and be presented with prices from different suppliers. MVS technology narrows the gap between the physical world and digital experience by pushing additional information to mobile devices. A few MVS-based products are available commercially, such as Google Goggles [Goo11], Metaio [Met15], Mobile Acuity [Acu15], TinEye [Tin15], CamFind [Cam15], Wikitude [Wik15], SenseTime [Sen15], findSimilar [Cor15], Slyce [Sly15], and Layar [Lay15]. In addition, a few companies provide developer tools or platforms such as Google Glass Development Kit [Goo14], Metaio SDK [Met15], SenseTime SDK and cloud APIs [Sen15], Qualcomm Vuforia [Qua15], and Wikitude SDK [Wik15].

Recent advances have been driven by several factors. First, imaging systems are becoming increasingly cheaper. Hence, mobile devices equipped with cameras of high resolution are ubiquitous. Second, high-speed communication networks (e.g., LTE-Advanced and Wi-Fi 802.11n) are widely available. Therefore, we can share captured images or videos from anywhere at any time. Third, the processing power of mobile devices has increased, allowing for complex image or video processing. Finally, a variety of sensors, such as GNSS receiver, digital compass, and gyroscope can be combined with an imaging sensor to perform complex tasks.

1.2 Technical challenges

In the computer vision meets mobile communication scenario, many technical challenges are required to be addressed, such as feature detection, retrieval engine, and data transmission. The detection of salient image features is a fundamental step in many computer vision applications, including MVS. Although various feature detection and description algorithms are proposed in the literature, such as scale-invariant feature transform (SIFT) [Low04] and speeded-up robust features (SURF) [BTG06], we cannot search everything using these features. MVS technology is generally efficient when searching for books, DVDs, logos, landmarks, wines, artworks, and so on. Hence, feature extraction is still a hot research topic. Regarding retrieval engine, owing to billions of images and videos provided by the Internet, especially social networks such as Facebook, Flickr, and Instagram, indexing and searching are extremely complex and inefficient processes. Thus, the development of a large-scale CBIR engine for visual search is another hot topic. The ever-expanding amount of multimedia data on the Internet has fostered research on effective, efficient, and robust algorithms to organize, index, and retrieve images or videos. Nevertheless, a full discussion of these two efforts is beyond the scope of this thesis.

In this thesis, we consider the data transmission challenge and assume that captured images or videos are compressed for transmission. Typically, images or videos are transmitted over a communication network and analyzed at a server. With adequate network resources, transmitted images or videos can be compressed with high quality. However, when the wireless link is weak or many customers compete for network resources, images or videos are compressed to low bitrates. The MVS system imposes requirements of latency, processing speed, and network bandwidth because these factors significantly affect user experience. Users desire a low latency and robust recognition performance.

Typically, an image or video encoder assumes that a human is the output information sink. Hence, the encoder minimizes the perceivable distortion for a given bit budget. In other words, lossy compression schemes underlying these standards, e.g., JPEG, are optimized for and adapted to the human visual system (HVS). However, as described above, in an increasing number of emerging application scenarios, the information sink is not a person but a computer algorithm. In this case, adaptation to the HVS is no longer necessary, and the encoder should control the encoding process such that the important and relevant features of an image are preserved after compression. In fact, typical low-bitrate compression artifacts (e.g., blockiness) confuse feature-based image analysis systems. Therefore, this thesis addresses the challenge of preserving features when compressing images or videos at low bitrates. Using feature-preserving image or video compression, the subsequent matching or retrieval performance can be improved. In this thesis, we use MVS, including the related features, compression schemes, and retrieval engines, to illustrate the proposed feature-preserving image and video compression approaches.

1.3 Contributions and organization of this thesis

This thesis presents various approaches to feature preservation for compressed images and videos. Images and videos encoded using the proposed approaches are standard-compatible and can be decoded by any standard decoder.

Chapter 2 presents the background and related work. This chapter discusses the state-of-the-art features, feature-related compression approaches in the literature, image and video compression standards, and the evaluation frameworks used in our experiments.

Chapter 3 describes the feature properties in the presence of compression. First, experiments are performed to investigate the impact of compression on the feature quality. The results show that image compression at low bitrates adversely affects feature-matching performance. Hence, in Section 4.1, first, different detector-descriptor combinations are tested, and it turns out that the combination of the Hessian-Affine detector and the SIFT descriptor is the most robust feature in the presence of compression. Second, the sensitivity of features at different scales under compression are tested. It is found that in the presence of compression artifacts, large-scale features are more robust compared with small-scale features. Based on this observation, approaches are proposed for image compression in Section 4.2 and for video compression in Chapter 5. Third, the sensitivity of keypoints and descriptors is evaluated. It is found that keypoint detection is easily affected by compression artifacts, and in contrast, descriptor calculation is more robust. Thus, approaches to encode the original keypoints for improved matching performance are discussed in Chapter 6. Fourth, owing to the fact that keypoint detection is more important, a theoretical analysis for various keypoint detectors is performed. We find that high frequencies of an image are less important in the keypoint detection process for scale-space-based detectors. Hence, in Section 4.3, a novel quantization table for JPEG encoding to improve feature detection performance is proposed. The aforementioned observations of feature properties in the presence of compression at low bitrates motivate our proposed approaches. Table 1.1 summarizes the observations and the corresponding chapters of this thesis. Nevertheless, we have organized the proposed approaches based on the different encoding standards used in the experiments. Chapters 4, 5, and 6 present several approaches using JPEG, H.264/AVC, and H.265/HEVC, respectively.

Table 1.1: Key observations of low-bitrate images and corresponding approaches.

Experiments	Observations	Chapters
Detector-Descriptor combinations	Hessian-Affine + SIFT is the most robust feature	Section 4.1
Sensitivity of features at different scales	Large-scale features are more robust	Section 4.2, Chapter 5
Sensitivity of keypoints and descriptors	Keypoint detection is more important	Chapter 6
Sensitivity of detectors to image frequencies	High frequencies are less important for scale-space-based detectors	Section 4.3

In Chapter 4, the scenario that still images are captured by a mobile device and encoded using the JPEG standard is considered. Three different ideas to improve the feature-matching performance or image-retrieval performance are proposed: 1) post-processing of a low-quality JPEG-encoded image at a server. 2) novel rate-distortion (R-D) optimization of a JPEG-encoded image in a mobile device. 3) novel JPEG quantization table for encoding images in a mobile device.

In Chapter 5, the scenario that videos are encoded using an H.264/AVC video encoder is considered. The first contribution is to perform R-D optimization for an H.264/AVC-encoded video in a mobile device, an extension of the second approach for images reported in Chapter 4. Second, by considering bandwidth requirements, a novel rate control framework is proposed, which preserves a relatively greater number of features at the same bitrates.

In Chapter 6, based on the observation that the negative impact of compression can be reduced using the original keypoints, we propose the encoding and transmission of the original keypoints as side information to the server and the extraction of only descriptors from the compressed image or video at the server. Specifically, intra- and inter-frame keypoint selection and encoding are proposed for images and videos, respectively.

Chapter 7 presents the conclusions and discusses the future directions of feature-preserving image and video compression research.

Parts of this thesis have been published in [CS11, CS12, CCS13, CANSS13, CHSS15, CSX15, CS16].

Chapter 2

Background and related work

This chapter presents relevant background and related work on feature-related compression research. First, Section 2.1 introduces state-of-the-art features proposed in the literature. In Section 2.2, previous studies on feature-related compression are reviewed, and the approaches are categorized into three classes for comparison. Section 2.3 briefly reviews widely used image and video compression standards because the proposed approaches generate image or video bitstreams compatible with these standards. The evaluation frameworks used in our experiments are discussed in Section 2.4. This chapter provides readers with an overview of the state-of-the-art work and the underlying elements necessary to understand the research details presented throughout this thesis.

2.1 State-of-the-art features

The extraction of features from images or videos is fundamental to many computer vision algorithms. The extracted features are frequently compared with features in a database to identify correspondences. Typically, in feature extraction, keypoints (also called interest points or salient points) are detected first, and then local descriptors (also called feature descriptors, visual descriptors, or feature vectors) are calculated from the image patch located around the keypoints.

2.1.1 Keypoints

Edges, corners, or blobs are all “interesting” parts of an image, and the algorithms used to find these parts are called detectors. Different detectors detect different types of interest points, e.g., SIFT detects corners and blobs while Harris-Affine [MS04] detects edges and corners. Ideally, feature extraction processes identify features that are shift-invariant, scale-invariant, rotation-invariant, illumination-invariant, and so on.

In the detection process, first, the above-mentioned interesting points should be detected repeatedly in different images containing the same object under different imaging conditions. Keypoint locations in an image are used to indicate interesting points, and the designed algorithm should identify shift-invariant keypoints to find similar images. Second, owing to the distance from which an object is observed with an eye or a camera, the object appears to have a different size compared to its physical size in the real world. *Scale-space* theory [Lin09, Lin12] detailed the framework for multiscale image representation derived from physics and biological vision in the computer vision community. Thus, the notion of a keypoint scale is essential for describing the keypoint size of an image. Third, rotation-invariance can be achieved by assigning a maximum gradient calculated from the pixels around the keypoint. Therefore, images captured from different angles can be identified correctly. These aforementioned properties are the main properties of keypoints in most feature detection algorithms. In addition, other properties are implemented in many detection algorithms. Real-time detection is required in various applications, for example, features from accelerated segment test (FAST) [RD06] and binary robust invariant scalable keypoints (BRISK) [LCS11] detectors were designed for this purpose. Illumination-invariance is also implemented using image gradients; for example, SIFT and SURF features use gradient magnitude normalization to this end. Affine-invariance deals with viewpoint changes of the same object. Mikolajczyk et al. [MTS$^+$05] discussed and compared several affine region detectors.

In the literature, various feature detectors have been proposed, and the most popular and important ones are evaluated in our experiments. The Hessian-Affine, Harris-Affine [MS04], and maximally stable extremal regions (MSER) [MCUP02] detectors outperformed other detectors in most of the experiments conducted by Mikolajczyk [MTS$^+$05]. Two other widely-used detectors, SIFT and SURF, are also included in our experiments. The STAR detector (based on center surrounded extrema) [AKB08] uses bi-level filters to approximate the Laplacian, which outperforms other detectors and can be computed in real time. The BRISK detector, which is based on FAST [RD06], is also compared in our experiments because various detectors derived from FAST are widely used in many applications owing to their short extraction time. We also include the recently proposed medial feature detector (MFD) [AR11] and weighted α-shapes (WαSH) [VRA12] detector in our evaluation. The KAZE [ABD12] detector, which detects features in a nonlinear scale-space outperformed other classic detectors [ABD12] and is hence included in our evaluation.

Figure 2.1 groups these detectors into non-affine-invariant and affine-invariant types. The plot on the left side shows five non-affine-invariant keypoint detectors, while that on the right presents five affine-invariant keypoint detectors. We can see that the non-affine-invariant detectors yield circular keypoints, while the affine-invariant detectors produce elliptical key-

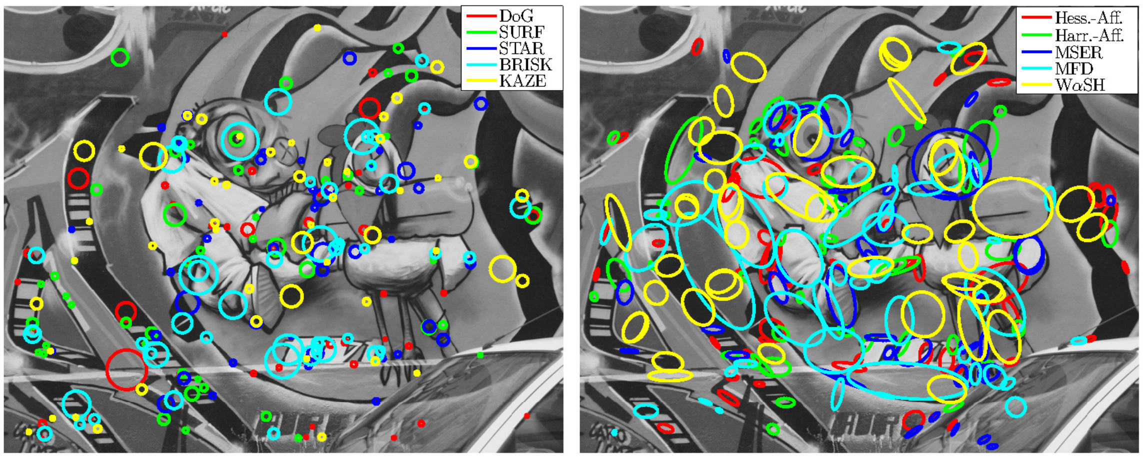

Figure 2.1: Non-affine-invariant detectors and affine-invariant detectors.

points. Owing to the different detection algorithms used in each detector, they detect different parts of the image. Figure 2.1 shows only a few randomly selected keypoints.

2.1.1.1 Difference-of-Gaussian detector

Each detector uses its own scheme to localize keypoints in an image. The difference-of-Gaussian (DoG) detector used in SIFT feature extraction is a classic example. A few critical points of this detector are explained in detail to facilitate better understanding of the typical detection process and of our proposed approaches that follow.

Difference-of-Gaussian images: As pointed out in scale-space theory [Lin09, Lin12], owing to the absence of prior knowledge about relevant scales of image data, it is reasonable to represent an image using multiple scales. Previous studies have singled out the Gaussian kernel as a canonical approach for building multi-scale images. Thus, first, the *scale-space representation*, which is a family of Gaussian-smoothed images at different scales, needs to be constructed. Typically, given an initial image signal I, its scale-space representation L is computed by convolving with the Gaussian kernel [Lin98, Lin09]:

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

This leads to

$$L(x, y; \sigma) = G(x, y; \sigma) * I(x, y) \tag{2.2}$$

The standard deviation σ is varied to obtain multiple Gaussian-smoothed images.

In addition, it has been shown that applying the Laplacian-of-Gaussian (LoG) to an image is critical for detecting blobs. Based on the Gaussian-smoothed scale-space representation, *scale-normalized Laplacian* with the factor σ^2 is vital for automatic detection of blobs from an image. Here, the scale-normalized Laplacian image is expressed as follows:

$$\nabla^2_{norm} L = \sigma^2 \nabla^2 L \tag{2.3}$$

where ∇^2 is the Laplacian operation on the scale-space representation L. Note that according to the basic associative property, the application of the Laplacian operator to the scale-space representation is equivalent to the filtering of the original image with the LoG operator.

$$\begin{aligned}\nabla^2 L &= \nabla^2(G(x,y;\sigma) * I(x,y)) \\ &= (\nabla^2 G(x,y;\sigma)) * I(x,y) \\ &= LoG * I(x,y)\end{aligned} \tag{2.4}$$

Furthermore, it has been shown that the scale-space representation satisfies the diffusion equation [Lin94, Low04].

$$\frac{\partial L}{\partial \sigma} = \sigma \nabla^2 L \tag{2.5}$$

The differential equation given above can be solved by approximating it with a difference equation according to the finite difference method. Thus, as long as k is close to 1, $\sigma\nabla^2 L$ can be expressed as follows.

$$\sigma\nabla^2 L = \frac{\partial L}{\partial \sigma} \approx \frac{L(x,y;k\sigma) - L(x,y;\sigma)}{k\sigma - \sigma} \tag{2.6}$$

and therefore,

$$L(x,y;k\sigma) - L(x,y;\sigma) \approx (k-1)\sigma^2\nabla^2 L \tag{2.7}$$

Given that k is a constant applied to σ for smoothing the images, it does not influence keypoint localization. As can be seen in Equation (2.7), the subtraction of two adjacent scale-space images approximates the scale-normalized Laplacian (Equation 2.3); thus, blobs can be detected within these images. According to the distributive property, the above-mentioned term can be expressed as follows:

$$\begin{aligned}D(x,y;\sigma) &= L(x,y;k\sigma) - L(x,y;\sigma) \\ &= G(x,y;k\sigma) * I(x,y) - G(x,y;\sigma) * I(x,y) \\ &= (G(x,y;k\sigma) - G(x,y;\sigma)) * I(x,y) \\ &= DoG * I(x,y)\end{aligned} \tag{2.8}$$

where $G(x,y;k\sigma) - G(x,y;\sigma)$ is called the DoG function [Low04] and $D(x,y;\sigma)$ represents the DoG scale-space.

Figure 2.2 shows the construction of DoG images. The initial image is smoothed using a Gaussian kernel with a base scale σ_0. Discrete scale representation is feasible for simplifying the continuous scale-space; thus, the smoothing factor increases logarithmically, i.e., $\sigma_{n+1} = k\sigma_n = k^{n+1}\sigma_0$. As a result, the construction of the scale-space representation is discrete but is efficient and sufficient, as described by Lowe [Low04]. To efficiently process images, Lowe [Low04] downsampled a Gaussian image by selecting every second horizontal pixel and every second vertical pixel, producing an image whose scale has twice that of the previous

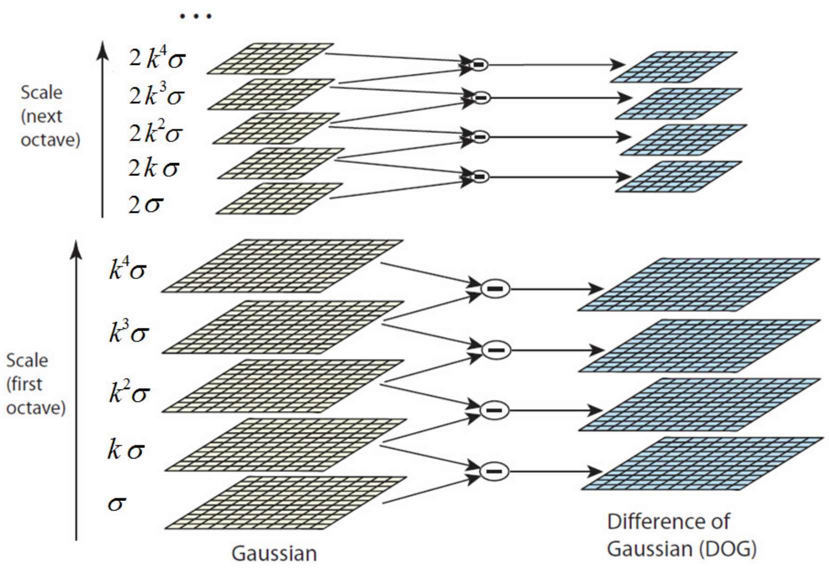

Figure 2.2: The construction of DoG images ($N = 2$, $k = 2^{1/2}$) [Low04] ©2004 Springer.

image. In the scale-space, the smoothed images that have the same resolution are within the same *octave*.

Scale-space extrema detection: Keypoints are detected as local extrema in DoG images across the spatial pixel domain and across various scales in the scale-space. Therefore, each point is compared to eight spatial neighbors, nine neighbors in the lower scale, and nine neighbors in the higher scale, as shown in Figure 2.3.

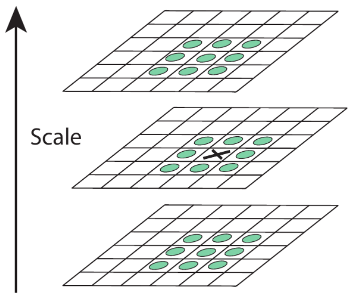

Figure 2.3: Extrema detection of DoG images [Low04] ©2004 Springer.

Note that the lowest and highest DoG images in one octave are excluded when searching for local extrema. Therefore, if one divides the scales in one octave into N intervals, $N + 2$ DoG images (right stack in Figure 2.2) and $N + 3$ Gaussian images (left stack in Figure 2.2) are required. Then, the scales of DoG images are completely covered for automatic scale detection. The scale in the DoG scale-space can be expressed in a different form as follows:

$$\sigma(o, i) = \sigma_0 2^{o+i/N} \tag{2.9}$$

where o is the *octave index* and i is the *scale index* in the o-th octave. Typically, the interval N is set to 3; thus, the constant $k = 2^{1/N} = 2^{1/3}$.

Accurate keypoint localization: The extrema (keypoint candidates) are found in the three-dimensional DoG scale-space, and these extrema are localized on the integer grid and at discrete scales. To find more accurate extrema in the DoG scale-space, the DoG detector selects ± 1 neighboring sample points in each dimension (spatial and scale domains) and locates the sub-pixel and sub-scale keypoints by fitting a three-dimensional quadratic function to the DoG scale-space. To calculate quadratic coefficients, the derivatives are approximated using pixel differences obtained from neighboring sample points.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{2.10}$$

where $\mathbf{x} = (x, y; \sigma)^T$ is the scale-space coordinate and $D(\mathbf{x})$ approximates the DoG function. The DoG detector takes the derivative of Equation (2.10) with respect to $\mathbf{x}$ and equates it to zero. Then, the sub-pixel and sub-scale extrema can be calculated as follows:

$$\hat{\mathbf{x}} = -\frac{\partial^2 D}{\partial \mathbf{x}^2}^{-1} \frac{\partial D}{\partial \mathbf{x}} \tag{2.11}$$

It has been shown that the refined sub-pixel and sub-scale extrema improve the matching performance and feature stability.

Weak extrema removal: After detecting many extrema, the detector removes the weak extrema with low contrast and the extrema from edge responses.

First, the value of the Taylor expansion function $D(\mathbf{x})$ in Equation (2.10) is calculated at point $\hat{\mathbf{x}}$. The detector substitutes $\hat{\mathbf{x}}$ into Equation (2.10) to obtain the extrema $D(\hat{\mathbf{x}})$.

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}} \tag{2.12}$$

This value can be used to remove the extrema with low contrast. Typically, the $D(\hat{\mathbf{x}})$ value of an extremum should be greater than 0.03; otherwise, the extremum is discarded.

Second, it has been shown that the DoG function yields high responses along edges, but these are unstable extrema. Based on the theory that principal curvatures along the edge and across the edge have significantly different values, the extrema with asymmetric principal curvature values are discarded to eliminate edge responses. The detector calculates derivatives for extrema using the difference of neighboring sample points and forms a 2×2 Hessian matrix as follows:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.13}$$

Then, it preserves only those extrema for which

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{2.14}$$

where r is typically set to 10. Other extrema are determined as unstable keypoints.

Orientation assignment: To achieve rotation invariance, the detector computes the most prominent orientation from the image patch around each detected keypoint and then assigns the orientation to the keypoint. To this end, the gradient orientations and magnitudes from the image patch are accumulated into a 36-bin histogram. The peak of this histogram indicates the dominant orientation for the keypoint. Then, regardless of image rotation, the following descriptor calculation is performed relative to the orientation.

In summary, given an input image I, a detected SIFT keypoint can be expressed as follows:

$$k_i = \{l_i, \sigma_i, \theta_i\} \tag{2.15}$$

where l_i (a vector of x_i, y_i) is the location of keypoint i in the image, σ_i is the scale, and θ_i is the orientation of the keypoint. Then, the set of keypoints can be expressed as $K = \{k_1, k_2, ..., k_N\}$, where N is the number of detected keypoints.

2.1.2 Descriptors

After detecting keypoints, local descriptors are used to describe the information around the detected keypoints. Many descriptor calculation algorithms have been proposed in the literature. Mikolajczyk et al. [MS05] presented a comprehensive evaluation of classic descriptors for image matching. The authors in [DAP11, MM12] compared descriptors proposed more recently. If the local descriptors of different image patches are highly distinctive, one can efficiently find different images of the same object by comparing these sparse feature descriptors. In our experiments, SIFT, PCA-SIFT [KS04], shape context [BMP02], gradient location and orientation histogram (GLOH) [MTS+05] and SURF are used. In addition, recently proposed rotationally invariant descriptors (called MROGH) [FWH12] and local intensity order pattern (LIOP) [WFW11] seem to be strong competitors. The two descriptors performed better than classic descriptors in the respective studies. Hence, they are used in a few of our experiments. These descriptors output feature vectors whose values are typically mapped onto the range [0, 255]. Besides these descriptors, binary descriptors such as binary robust independent elementary features (BRIEF) [CLSF10], BRISK, and fast retina keypoint (FREAK) [AOV12], have been proposed in the literature. These binary features are generally faster to compute and require less storage. They are not evaluated in our experiments, but a few of the proposed ideas can be also applied to them. Next, we present the SIFT descriptor as a typical descriptor example, and details pertaining to the other aforementioned descriptors can be referred to in the literature.

2.1.2.1 SIFT descriptor

The SIFT descriptor describes a local image patch in a set of gradient orientation histograms. For one keypoint, SIFT computes the gradient in a 16×16 window around it. To reduce the influences of gradients that are relatively far away from the center, gradient magnitudes in the window are weighted using a Gaussian function. The window is subdivided into a 4×4 array. In each element of the array, the weighted gradient magnitudes are accumulated into eight orientation bins. Therefore, the final feature vector (i.e., the total orientation histogram) has a length of 4×4×8 = 128. In addition, several normalization schemes are applied to remove illumination effects. Figure 2.4 shows the extraction process of a SIFT feature vector.

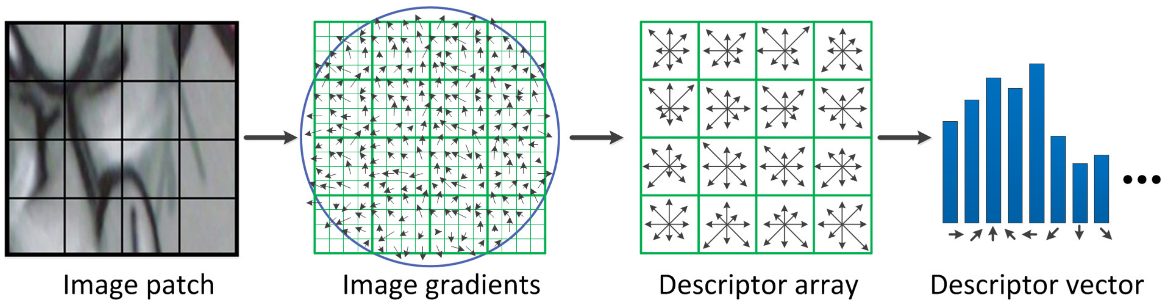

Figure 2.4: Computation of SIFT descriptor based on image patch [Low04].

As a result, a descriptor obtained using a keypoint detected in image I is expressed as follows:

$$d_i = \Psi(k_i|I) \tag{2.16}$$

where Ψ implies the descriptor extraction on I using keypoint k_i. The set of descriptors is expressed as $D = \{d_1, d_2, ..., d_N\}$.

2.2 Feature-related compression

In mobile visual search, most feature-related processing is typically performed at a server. Note that feature extraction can be performed by either the client or server. For client-side feature extraction, compressed features are uploaded to the server. The feature compression operation should lead to small distortion relative to the uncompressed features. For server-side feature extraction, images or videos are compressed and transmitted to the server. In this case, it is important to minimize the impact of compression on feature extraction at the server. Ideally, features extracted from compressed images or videos should be identical or at least very similar to those extracted from the uncompressed version. Both client- and server-based approaches fall into the emerging area of feature-related compression approaches.

Girod et al. [GCGR11] introduced three MVS architectures: 1) mobile device transmits compressed images, 2) mobile device extracts descriptors and transmits compressed descriptors, and 3) mobile device stores a dataset in its cache and transmits only compressed descriptors when the object of interest is not available in the cached dataset. By considering feature-related compression, we categorize previous studies into three classes. The first class directly compresses features. The second class compresses canonical image patches and transmits or stores these patches for further processing. Third is the standard image/video-compression-based approaches.

2.2.1 Feature compression

Chandrasekhar et al. [CTC$^+$09] proposed a framework called CHoG for compressing feature descriptors. Then, they presented a survey on SIFT compression schemes [CMT$^+$10]. For videos, Makar et al. [MCT$^+$14] proposed encoding of SIFT descriptors based on inter-frame encoded patches, and Baroffio et al. [BCR$^+$14] proposed intra- and inter-frame modes for encoding SIFT features extracted from video sequences. The proposed coding schemes outperformed direct transmission of images or videos with respect to their evaluation metrics. In addition, Redondi et al. [RBCT13, RBA$^+$13, BAC$^+$14] compressed features and transmitted them to the server side; this class is called *analyze-then-compress* paradigm. The compact descriptors for visual search (CDVS) [MPE] standard is aimed at standardizing technologies for encoding features at low bitrates. In CDVS, the elements of a compact descriptor are calculated from the difference between the elements of the original descriptor. They are assigned priorities specified in the standard. Then, the elements with low priorities are removed to reduce the dimension of the compact descriptor. Thereafter, the real-valued elements are quantized into three values (-1, 0, 1) to further reduce the data [ISO14]. In general, this class provides the highest compression ratio among the three classes. However, images or videos are not available at the server, and only one specific feature type (e.g., SIFT) is available.

2.2.2 Patch compression

Makar et al. [MCC$^+$09] proposed compressing relevant local patches after feature extraction. In a follow-up work [MLCG12], they proposed a method for designing a gradient-preserving quantization matrix to encode the extracted image patches, and in another follow-up work [MTC$^+$13], they proposed an efficient inter-frame predictive coding approach for canonical patches extracted from video sequences. The patch-encoding approach does not output a watchable video, but the patches can be used to extract other types of descriptors, which is advantageous compared to the above-mentioned feature compression scheme.

2.2.3 Feature-preserving image and video compression

The authors of [RBCT13] have dubbed the standard image-compression-based architecture the *compress-then-analyze (CTA)* paradigm. Furthermore, a few approaches involve modifying standard image and video compression algorithms such that the features extracted from the compressed images or videos are as similar as possible to the features extracted from the uncompressed images or videos. These approaches (also belonging to the third class) are referred to as *feature-preserving image and video compression* [CS11, CS12, CCS13, CSX15]. To this end, the authors of [DLC$^+$12] and [CCS13] optimized the JPEG quantization table, whereas in [CS11, CS12], the rate allocation strategy is modified to preserve the most important features. Recently, [CSX15] has proposed encoding the SIFT keypoints and transmitting them as side information along with the compressed image. Similarly, [BCR$^+$15] proposed transmitting the encoded BRISK keypoint locations, scales, and differential BRISK descriptors along with the image to a server. In both approaches, the keypoints are sent as side information for improved feature extraction from compressed images. Compared with the first two classes, the advantages of feature-preserving image and video compression are that the decoded images or videos can also be viewed and stored for future use and that other types of features can later be extracted. For solutions providing standard-compatible images or videos, standard decoders can be used to decode the images or videos. In this thesis, we focus on **feature-preserving image and video compression** and present our work in detail.

Note that in all studies mentioned above, the objective is to achieve low-bitrate data transmission for applications such as mobile visual search, video surveillance, and visual localization.

2.3 Image and video compression standards

Compression aims to reduce the number of bits needed to represent the original image or video information. Typically, there are two compression schemes: *lossless* and *lossy*. The former scheme compresses images or videos without losing any information, and it can reconstruct the original data completely. It is used specifically for medical images, professional photography or digital cinematography. The latter scheme introduces acceptable distortion to achieve a significantly large compression ratio. This scheme is commonly used for media streaming via the Internet or videoconferencing. In MVS applications, we must compress data in a lossy manner owing to limited communication network resources. To this end, for images, JPEG and JPEG 2000 are the two major standards, and for videos, H.264/AVC is a widely deployed compression standard in many consumer electronics. Its successor H.265/HEVC offers superior compression performance and is gaining traction in terms of usage. To evaluate the approaches proposed in this thesis, experiments are conducted using the popular JPEG standard for image compression and the H.264/AVC and H.265/HEVC standards for video compression.

Quality evaluation of compressed images or videos is a basic but challenging problem.

Many techniques and metrics for image or video quality assessment have been researched and developed over the past few decades. Such quality evaluation can be classified into two types based on whether human viewers are involved in the process: *subjective* and *objective*. In cases where the HVS is the information sink for compressed images or videos, it is natural to develop a quality evaluation metric by standard procedures involving human observers, i.e., subjective quality evaluation. In the literature [IR07, IT96], various metrics such as viewing conditions, test materials, observer selection, assessment methods have been described for a subjective assessment of video quality. The subjective viewers' ratings reliably indicate video quality but depend heavily on viewers' personal conditions and experience. Moreover, the evaluation procedure is laborious and time-consuming. However, researchers have traditionally used peak signal-to-noise ratio (PSNR) to compute image or video quality in objective quality evaluation. The definition is simple, and the computation is instantaneous. Hence, this method is widely accepted and still dominant in quality evaluation. Nevertheless, it neglects the HVS and cannot match human perception of image or video quality. To address this issue, a few advanced objective metrics are proposed to approximate the results of subjective quality evaluation by human observers. For example, the structure Similarity (SSIM) index [WBSS04] was introduced by assuming that human visual perception is highly adapted for receiving structural information from an image. It is shown that SSIM yields better evaluation performance with different datasets. Many other advanced objective evaluation metrics have been proposed, and the researchers [LK11, CSRK11, SSBC10, Cha13] reviewed, classified, discussed, and compared numerous image and video evaluation methods proposed in the literature.

In this thesis, the aforementioned standards are used to encode images and videos, respectively. As discussed above, image or video encoding algorithms normally take a human-centric approach and maximize the subjective/objective quality of the compressed content by exploiting known limitations of the human visual perception system. In computer vision meets mobile communication scenarios, image or video content is processed by computer algorithms as opposed to humans. In such scenarios, lossy compression schemes need not maximize visual quality. The core ideas of this thesis are to control the encoding process so that the most important and relevant features are preserved satisfactorily for low-bitrate encoding. The subsequently applied image or video analysis approaches (e.g., detection, matching, or tracking) can then achieve better performance. Note that our ideas can also be applied to other compression standards or future standards.

2.4 Evaluation frameworks

Feature evaluation is typically performed using either a frame- or system-level evaluation framework. The former approach is normally based on only a few images, can compare many features in a short time, and is application independent [Ved12]. However, it cannot reflect the actual performance for specific applications. *Repeatability* and *matching score* [MTS+05] are the most important criteria in this context for evaluating detectors and descriptors, respectively. System-level feature evaluation, in contrast, employs many images and is typically performed for just one specific feature or one specific application, e.g., MVS, video surveillance, or object classification. This evaluation requires a large dataset, specific performance criterion, and long test time. In this thesis, we aim to preserve greater numbers of features of images and video sequences at the frame-level. However, these features should be used in a real system. Whether the improvement of feature quality can translate into a noticeable improvement in a real system should also be verified. Therefore, to evaluate feature quality and the final performance in a real system, both evaluation frameworks in our experiments are used.

2.4.1 Frame-level evaluation

Many datasets are available for frame-level evaluation. Among these datasets, the Oxford Affine Covariant Features dataset [MTS+05] has been used extensively in the literature. This dataset consists of eight image sets: *graf*, *wall*, *boat*, *bark*, *bikes*, *trees*, *ubc*, and *leuven*. In addition, we employ the Stanford mobile augmented reality (MAR) dataset [MTC+13], which consists of 23 videos (each containing a static single object) and 23 corresponding reference images. These two datasets are used in our experiments, the details of which are explained in the following chapters.

Typically, features are extracted from an image and compared with those extracted from a reference image. This process is called *pairwise feature matching*. Ideally, feature locations and feature descriptors before and after image or video compression would be identical if preserved perfectly. This is, however, impossible in the case of lossy compression. In many computer vision applications, two features are considered to be identical as long as their "distance" in the descriptor space is less than a certain threshold. Corresponding features should refer to the same point in the scene before and after some form of image processing.

In the literature, various detector and descriptor evaluation methods have been proposed. Schmid et al. [SMB00] defined the *ϵ-repeatability* of point pairs, which is widely used as a detector evaluation criterion. They examined the repeatability of location errors ϵ ranging from 0.5 to 5 pixels. Mikolajczyk et al. [MTS+05] introduced the *overlap error*, which is defined as the error of region-to-region ellipses from a pair of affine region detectors. The number of feature pairs with less than 40% overlap error is called the **number of correspondences**. The **repeatability score** is calculated as the ratio of the number of correspondences to the smaller number of features in the pair of images. In [MTS+05], detectors with different shapes were represented using elliptical regions to facilitate their meaningful comparison. Several image conditions were examined, including viewpoint change, zoom plus rotation, image blur, JPEG compression, and light change. In our experiments, a few variants of the criteria in [MTS$^+$05] are used for simplicity. According to the measure, the overlap error of two feature regions should be sufficiently small so that they can be matched using a robust descriptor. In a few of our experiments, only the uncompressed and compressed version of a given image are compared. The homography between an uncompressed image and its compressed version is hence an identity matrix. Therefore, the overlap error formula can be simplified as

$$\epsilon = 1 - \frac{R_a \cap R_b}{R_a \cup R_b} \tag{2.17}$$

where R_a and R_b represent the regions of features from the uncompressed and compressed images. $R_a \cap R_b$ denotes the intersection of the two regions, and $R_a \cup R_b$ denotes their union. For example, the detectors of SIFT and SURF features produce circular keypoint regions. Therefore, only the difference of locations and the scale of the original feature need to be checked. Hence, a feature is defined to be repeatable, if: (1) the scale of the new feature in the SIFT scale space is within a factor of $\sqrt{2}$ of the original scale, and (2) the location is within σ pixels of the original feature where σ is the scale of the feature in the original image.

For descriptor evaluation, there are three strategies in general. First, two descriptors (D, D_a) are treated as a matched pair if the distance between them is below a threshold ($||D - D_a|| < t$), which is called *threshold-based matching* [MS05]. This strategy shows the true distance of two features in the descriptor space. The second one is called nearest neighbor (NN) matching strategy; it compares one feature to numerous features and finds the nearest one as its match [CLSF10, LCS11]. To remove a few incorrectly matched features, the third strategy involves finding the nearest feature (D_a) and the second nearest feature (D_b) and applying thresholding to the distance ratio between them. Hence, the query feature (D) and the nearest feature (D_a) are matched if $||D - D_a||/||D - D_b|| < t$ (e.g., 0.8) [MS05, Low04]. This strategy is similar to the previous one (NN) and is known as the nearest neighbor distance ratio (NNDR) strategy.

Two cases are considered when counting the **number of matches** in the following experiments. 1) When comparing features from a compressed image and an uncompressed image where the homography is an identity matrix, a correct match is found if a feature is repeatable and its descriptor is the nearest neighbor in the descriptor space (e.g., NNDR strategy). This case is shown in Figure 2.5. The **matching score** (ranging from 0% to 100%) is computed as the ratio of the number of matched features in the compressed image to the number of original features in the uncompressed image.

$$matching\ score = \frac{\#correct\ matches}{\#original\ features} \tag{2.18}$$

Note that the term matching score was originally used for comparing two different images. It was computed as the ratio of the number of correct matches to the smaller number of detected

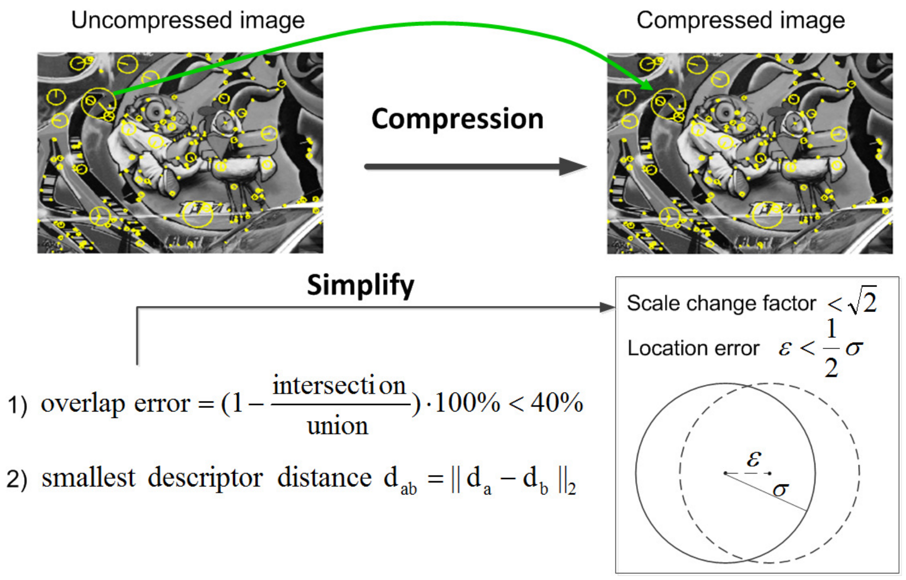
Uncompressed image
Compressed image
Compression
Simplify
Scale change factor $< \sqrt{2}$
Location error $\varepsilon < \frac{1}{2}\sigma$
1) overlap error $= (1 - \frac{\text{intersection}}{\text{union}}) \cdot 100\% < 40\%$
2) smallest descriptor distance $d_{ab} = \| d_a - d_b \|_2$
ε
σ

Figure 2.5: Criterion for matches between uncompressed and compressed images.

features in one pair of images in [MTS+05]. In this thesis, it is used as the percentage of preserved features with respect to original features.
2) When comparing a compressed image with a reference image without knowing the homography, we must first obtain tentative matches by comparing the descriptors of the two images in the descriptor space using a descriptor comparison strategy (e.g., NNDR). Then, random sample consensus (RANSAC) [FB81] is used to remove incorrectly matched features assuming affine transformation between the reference and compressed images. This case is shown in Figure 2.6. The number of matches after RANSAC can be used as the performance evaluation criterion.

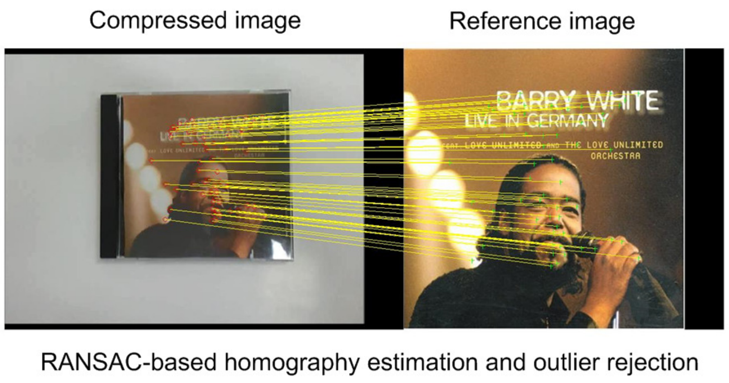

Figure 2.6: The criterion for matches between a reference image and a compressed image.

To summarize, the repeatability score, number of correspondences, matching score, and number of matches are typically used as the pairwise feature-matching performance evaluation criteria. Note that in this thesis, the number of matches refers to the number of preserved features. A few variants are used, and the corresponding settings of each experiment will be explained in detail.

2.4.2 System-level evaluation

System-level evaluation is performed using a real system, e.g., object detection, text recognition, image retrieval, or video surveillance systems. Each application has its own evaluation framework and specific criterion. In this thesis, the CBIR engine used in MVS is explained as an example.

Many test image datasets such as the Oxford Buildings dataset [PCI$^+$07], the Holiday dataset [JDS10], ImageCLEF [Ima] are available for image retrieval evaluation. In this thesis, we mainly use the Oxford Buildings dataset, which is widely used for evaluation in the literature. It consists of 5062 database images and a query set of 55 images obtained from 11 distinct locations in the city (5 query images for each location). Each query image is associated with several label files that identify the images in the database that are relevant to the query image.

Typically, each image provides hundreds or thousands of features. Therefore, frame-level matching in a CBIR engine for a large-scale dataset is infeasible. In the literature, many schemes address the issue of image retrieval on a large scale. These schemes typically produce a vector of each image using its local descriptors. Notably, three image representation schemes are widely used: bag-of-features (BoF) / bag-of-words [SZ03], Fisher vector (FV) [PSM10], and vector of locally aggregated descriptors (VLAD) [JDSP10]. The BoF scheme quantizes the feature descriptor space into so-called visual words (VWs) by k-means clustering. Each descriptor is assigned to its closest centroid (namely VW). The BoF vector of an image is then formed by accumulating the VWs of all descriptors in this image. The histogram is then normalized, and it forms a k-dimensional vector. The similarity of a pair of images can be compared using L_1 or L_2 distance of two BoF vectors. The FV scheme uses Gaussian mixture models (GMMs) to model the descriptor distribution instead of the VWs. The local descriptors of a training dataset are clustered to generate GMMs, which approximate the descriptor distribution in the descriptor space. A GMM is represented by a set of parameters such as mixture weight, mean value, and covariance matrix. Then, both first- and second-order differences between the GMMs and local descriptors are calculated and concatenated to form a vector. Then, this vector is power and L2 normalized [PSM10] to finally generate an FV representation of an image. The VLAD scheme can be treated as a simplified version of the FV scheme. First, instead of the GMMs used in FV, VLAD uses k-means to generate the VWs. Second, VLAD uses only the first-order difference to produce an FV vector. The details of each scheme can be referred to in previously mentioned studies. Instead of quantizing or modeling descriptors as used in the aforementioned schemes, some researchers continue to use original descriptors directly in image-retrieval engines for small-scale datasets because the precision of the retrieval system can be improved using the descriptors directly. For example, Jégou et al. [JDS11] exploited descriptor distances for a precise image search application. They first exploited the information generated by distances from top k-nearest neighbors (k-NN). They compared the distances of k-1 first neighbors with that of the k-th nearest neighbor and proposed a query-adaptive criterion to improve voting quality. Second, they exploited distances associated with reciprocal nearest neighbors. The algorithm checked a pair of descriptors to determine whether they are in each other's k-NN list. These two ideas refine the similarity measure among descriptors, thus improving retrieval performance. This is called the *descriptor-distance-based* scheme in this thesis.

Regarding the performance evaluation metrics, *precision* and *recall* are typically used in the CBIR community. Precision is the ratio of the number of relevant images retrieved to the number of retrieved images. Recall is the ratio of the number of relevant images retrieved to the number of total relevant images in the database [MRS08]. These two metrics can be explained as follows.

$$precision = \frac{\#\ \{relevant\ images\ retrieved\}}{\#\{retrieved\ images\}} \tag{2.19}$$

$$recall = \frac{\#\ \{relevant\ images\ retrieved\}}{\#\{relevant\ images\}} \tag{2.20}$$

The CBIR engine typically returns a list of ranked images. However, the number of retrieved images can be selected differently. By tuning the number of retrieved images, precision can be plotted as a function of recall (i.e., *precision-recall curve*). A typical way to evaluate performance is to look at how precision and recall change as the number of retrieved images changes. In an ideal retrieval system, the precision is 1 for all recall values. Rather than comparing curves of different retrieval results, sometimes, it is helpful to use a single number to measure performance. In the image-retrieval community, the average precision (AP) score is used when considering the two metrics. It is calculated as the area under the precision-recall curve for a query image [PCI+07]. Averaging the AP scores of all query images, the **mean average precision** (mAP) score as an overall metric of the retrieval system is obtained. Note that the number of retrieved images can be set to a fixed number (e.g., 100) or as a certain percentage of the overall database (e.g., 5%).

As mentioned previously, features and the CBIR engine are not in the scope of our research. We only need a retrieval system to validate our proposed feature-preserving approaches. Hence, we mainly use the Oxford Buildings dataset, BoF-based retrieval engine [PCI+07], descriptor-distance-based retrieval engine [JDS11], and the mAP score to evaluate the performance of feature-preserving image and video compression in our experiments.

Chapter 3

Effects of low-bitrate compression on image features

Our goal is to preserve the most important features when compressing images or videos at relatively low bitrates in a standard-compatible manner. In this chapter, first, the impact on feature quality is studied using different image compression standards. Then, based on the observations in our experiments, the sensitivity of features at different scales is discussed, and the sensitivity of keypoints and descriptors is also presented. In addition, a theoretical analysis is performed to study the sensitivity of detectors to image frequencies. The observations made in this chapter motivate our proposed approaches in Chapters 4, 5, and 6.

3.1 Impact of compression on feature quality

Features from images encoded with JPEG, H.264/AVC Intra, and H.265/HEVC Intra are extracted and then matched to features extracted from a set of uncompressed reference images. The number of matches as a function of the bitrate of the compressed images is plotted, and the results achieved with the different standards are compared. The results show that feature quality is significantly affected by compression artifacts at low bitrates.

The Stanford MAR dataset [MTC$^+$13] is used in this study. As introduced before, this dataset comprises 23 videos and 23 corresponding reference images. Each consists of 100 frames, recorded with a resolution of 640 $\times$ 480 at 30 fps. Similar to [MTC$^+$13], we use eight video sequences (*OpenCV*, *Wang Book*, *Barry White*, *Janet Jackson*, *Monsters Inc*, *Titanic*, *Glade*, and *Polish*) for pairwise feature-matching evaluation. SIFT features are extracted using the VLFeat [VLF] implementation. Similar to [MTC$^+$13], the top 200 features of each frame are selected according to the CDVS Test Model. For evaluating the matched descriptors, the NNDR strategy and the number of matches between a compressed image and a reference image as described in Section 2.4.1 are used.

Images from the eight test video sequences are encoded using JPEG with quality values of 4, 8, 12, and 16 (MATLAB, grayscale); JM reference software (version 18.4) [HHIb] with QP values of 38, 42, 46, and 50; and HEVC Test Model (version 16.0) [HHIa] with QP values of 38, 42, 46, and 50. Then, SIFT features are extracted from the compressed images and compared with the SIFT features extracted from the corresponding reference images. The number of matched features after applying RANSAC and the bitrates of the encoded images are averaged for all test images.

3.1.1 Feature quality comparison for different compression standards

The solid red, green, and blue curves in Figure 3.1 show the feature-matching performance (see the number of matches described in Section 2.4.1) for JPEG-, H.264/AVC Intra-, and H.265/HEVC Intra-encoded images. Owing to the superior rate-distortion (R-D) quality of H.265/HEVC, the H.265/HEVC Intra-encoded images lead to considerably better performance than the images encoded with JPEG and H.264/AVC Intra in terms of the number of matches at a given bitrate. Moreover, there is strong dependency on the available bitrate for a given compression standard. The greater the number of bits, the higher is the number of matches the pairwise matching yields. Here, we find that the feature quality is significantly affected by compression at low bitrates. Therefore, in Section 4.1, several detector-descriptor combinations are used to investigate the feature that is the most robust in the presence of compression. In addition, several filtering approaches are applied to improve the retrieval performance.

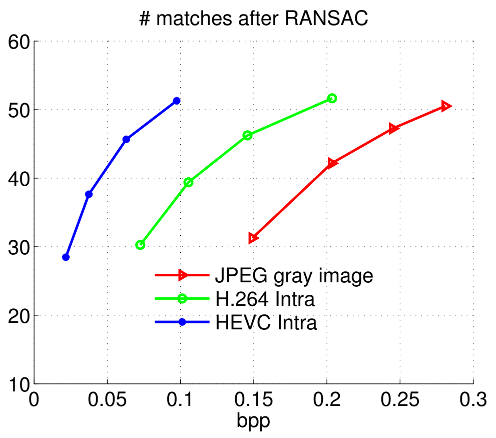

Figure 3.1: SIFT feature matching performance for different compression standards.

3.2 Sensitivity of features at different scales

This section presents the sensitivity of features detected at different scales in the presence of compression. To better distribute the bit budget for images, we need to investigate feature properties to design a better way than traditional image compression targeted for humans. Each SIFT feature has a scale, and the features of different scales can overlap, i.e., one image block can be relevant for several features. This is shown in Figure 3.2. Hence, simple block-by-block bit allocation will not be optimal. Therefore, one main challenge we address is allocating the bit budget such that as many important features as possible are preserved after compression.

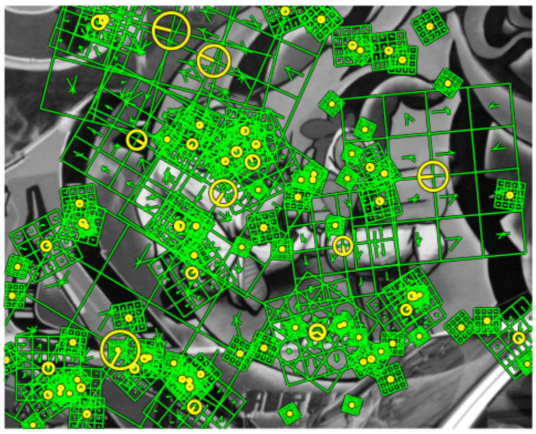

Figure 3.2: SIFT features on sample image.

In this experiment, test images from the Oxford Affine Covariant Features dataset are used. For these images, we aim to preserve the 200 strongest features. The strongest features are those that lead to the largest detector response during detection. The images are compressed using the JPEG codec from [Gro]. Here, we observe the sensitivity of features at different scales in the presence of compression. To evaluate the features in different octaves, the matching score (see Equation (2.18)) calculated between the uncompressed and compressed image pairs is used as the criterion. An observation about feature scales and the matching score is performed. The curves in Figure 3.3 show the matching scores separately for the SIFT feature in each octave for images compressed with JPEG quality level 50. Note that in SIFT feature detection, the input image can be linearly interpolated to form a larger image, which is used to build the DoG images in the “-1”-th octave [Low04]. This process increases the number of stable keypoints, but is optional. In this experiment, the keypoints in the “-1”-th octave are included, as shown in Figure 3.3. The seven different curves are from the first images of the *graf*, *bikes*, *cars*, *bark*, *trees*, *ubc*, and *wall* sets, respectively. Only a few octaves contain SIFT features. From the curves, we conclude that the lower the octave, the more vulnerable are the features. That is because features in lower octaves have smaller scales, and they are easily influenced by JPEG compression. In comparison, large-scale features are more robust to compression artifacts. Large-scale features contain greater amounts of information, and thus they have higher discriminative power [MTS+05]. To preserve the strongest features across all scales, we should allocate a higher bit budget to the error-prone small-scale features. In Chapter 5, similar experiments for videos are presented showing that this property also applies in the presence of video compression as well. Therefore, in Section 4.2 and Chapter 5, we present different approaches to feature-preserving image and video encoding using this property.

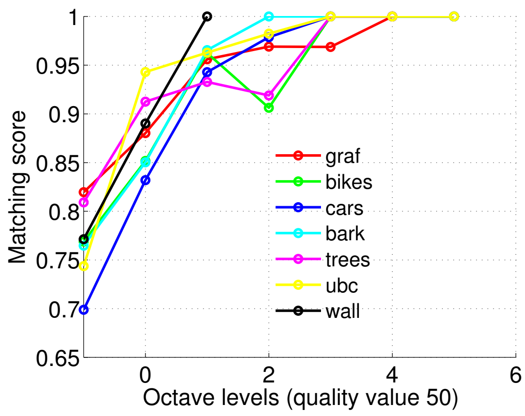

Figure 3.3: Matching scores in each octave

3.3 Sensitivity of keypoints and descriptors

In this section, the sensitivity of keypoints and descriptors under compression is investigated. The dataset, criterion, and experimental settings are the same as those in Section 3.1. As described in Section 2.1, given an input image I, the detected keypoints are expressed as k_i (Equation (2.15)), and the descriptors obtained using the keypoints detected in image I are expressed as d_i (Equation (2.16)). Similarly, the descriptors extracted from the compressed image $\overline{I}$ can be expressed as follows:

$$\overline{d_i} = \Psi(\overline{k_i}|\overline{I}) \tag{3.1}$$

where $\overline{k_i}$ represents the keypoints detected in $\overline{I}$. A simple experiment is performed to show that keypoint detection is easily affected by compression artifacts, and that in contrast, the descriptor is more robust. To demonstrate this, the descriptors of the compressed images were calculated using the keypoints extracted from the original (uncompressed) images. This can be expressed as follows:

$$d'_i = \Psi(k_i|\overline{I}) \tag{3.2}$$

This means that features d_i' have the same keypoints k_i as in the uncompressed case. This allows us to ignore inaccurate keypoint detection from the compressed image and exclusively evaluate robustness of the descriptor in the presence of compression artifacts. In this experiment, this idea is applied to the images encoded with H.265/HEVC Intra. Note that this idea can be applied to other image encoding schemes or standards as well. As the original keypoints k_i are idealistic and cannot be obtained from the compressed image $\overline{I}$, the matching result obtained using H.265/HEVC Intra and the original keypoints is shown as a dashed line (upper bound) in Figure 3.4. The results indicate that if the original keypoints are preserved, the descriptor vectors can be matched well despite strong compression. This observation motivates us to explicitly encode keypoints and send them as side information along with the compressed image. A similar observation for videos and the proposed keypoint encoding approaches in detail are presented in Chapter 6.

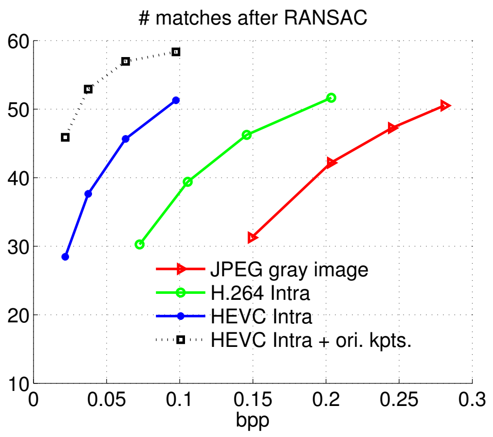

Figure 3.4: SIFT feature-matching performance for different standards. The dashed line denotes the result obtained for descriptors calculated on the H.265/HEVC Intra-encoded images using the original keypoints from uncompressed images.

3.4 Sensitivity of detectors to image frequencies

As presented in Section 3.3, the feature detection process is more sensitive at low bitrates. Therefore, in this section, we theoretically investigate the sensitivity of various detectors to image frequencies. Mikolajczyk et al. [MTS$^+$05] compared several popular detectors and showed that no detector was superior to all the other detectors under all image conditions. The experiments revealed that the Hessian-Affine, Harris-Affine, and MSER detectors seem to be the top three for most types of image changes. In the case where only JPEG compression artifacts are considered, the Hessian-Laplacian and Harris-Laplacian [MS01, Lin98] detectors instead of the Hessian-Affine and Harris-Affine is examined. In addition, two widely used detectors, DoG (SIFT detector) and SURF, are compared. The watershed segmentation-based MSER detector, which does not perform scale-space construction, is included in the discussion as well.

3.4.1 Scale-space derivatives of different detectors

As shown in Section 2.1.1.1, usually, scale-space-based detectors proceed as follows: 1) computation of multi-scale images, 2) local extrema detection, 3) location and scale refinement, and 4) weak keypoint removal. First, the operators used for scale selection of scale-space-based detectors are compared. Inspired by biological vision, the scale-space theory has been studied extensively by the computer vision community in the context of feature detection [You87, Lin98, Lin09, Lin12]. The scale-space representation is a family of smoothed images at different scales. As introduced in Section 2.1.1.1, given an image signal I, its scale-space representation L (Equation (2.2)) is computed by convolving the image I with the Gaussian kernel $G(x, y; \sigma)$ (Equation (2.1)).

The standard deviation σ is varied to obtain multi-scale images. Based on the concept of Gaussian-smoothed scale-space representation, different detectors use different convolution kernels. The operation is performed by convolving image signal I with Gaussian derivatives to detect corners, edges, and blobs. The generated multi-scale image family is called *scale-space derivatives* [Lin09]. Then, various types of linear or nonlinear combinations of scale-space derivatives are used to detect keypoints. The different detectors considered in this section perform this as follows.

Laplacian of Gaussian (Hessian-Laplacian and Harris-Laplacian)

$$\nabla^2_{norm} L = \sigma^2 (L_{xx} + L_{yy}) \tag{3.3}$$

Approximate determinant of Hessian (SURF)

$$\begin{aligned} det(\mathcal{H}_{approx}) &= \widetilde{L_{xx}}\widetilde{L_{yy}} - (0.9 \cdot \widetilde{L_{xy}})^2 \\ &\approx L_{xx}L_{yy} - (L_{xy})^2 \end{aligned} \tag{3.4}$$

Difference-of-Gaussian (DoG)

$$\begin{aligned} DoG &= L(x, y; k\sigma) - L(x, y; \sigma) \\ &= (G(x, y; k\sigma) - G(x, y; \sigma)) * I(x, y) \\ &\approx (k-1)\nabla^2_{norm} L \end{aligned} \tag{3.5}$$

where

$$L_{xx} = \frac{\partial^2 G(x, y; \sigma)}{\partial x^2} * I(x, y) \tag{3.6}$$

$$L_{yy} = \frac{\partial^2 G(x, y; \sigma)}{\partial y^2} * I(x, y) \tag{3.7}$$

$$L_{xy} = \frac{\partial^2 G(x, y; \sigma)}{\partial x \partial y} * I(x, y) \tag{3.8}$$

and $\widetilde{L_{xx}}$, $\widetilde{L_{yy}}$, $\widetilde{L_{xy}}$ are box-filtered approximations thereof [BTG06]. In brief, the scale-space derivatives can be represented by the common formula [Lin09]:

$$L_{x^\alpha y^\beta}(x, y; \sigma) = (\partial_{x^\alpha y^\beta} G(x, y; \sigma)) * I(x, y) \tag{3.9}$$

where α and β vary from 0 to 2, i.e., α=2 and β=0 for L_{xx}; α=1 and β=1 for L_{xy}. Equivalently, owing to the associative property, the scale-space derivative can also be computed by directly differentiating the scale-space representation L.

$$\begin{aligned} L_{x^\alpha y^\beta}(x, y; \sigma) &= \partial_{x^\alpha y^\beta}(G(x, y; \sigma) * f(x, y)) \\ &= \partial_{x^\alpha y^\beta} L(x, y; \sigma) \end{aligned} \tag{3.10}$$

In contrast to these scale-space-based interest point detectors, MSER detects keypoints without any smoothing process. It finds stable and connected intensity regions using a watershed segmentation algorithm. During this process, fine and large structures of various sizes are detected automatically without the generation of multi-scale images.

3.4.2 Low-pass Gaussian filter

As shown in the above formulae, with the exception of MSER, the detectors use different combinations of scale-space derivatives. For one specific scale, the image can be convolved with a band-pass filter, e.g., the Laplacian of Gaussian. During detection, however, multiple scales need to be calculated. Regardless of the combination of scale-space derivatives used by the detectors, all detectors first smooth the images using a low-pass Gaussian filter, according to Equation (3.10). When σ increases, the cut-off frequency of the filter shifts to a low frequency in the frequency domain. The cut-off frequency is defined by the standard deviation σ_0 of the initial scale layer. Frequencies higher than the cut-off frequency are useless for feature detection.

According to this observation, a novel quantization table for the widely used JPEG compression standard is designed to improve feature detection performance, as described in Section 4.3. The results justify that high frequencies of an image are less important for scale-space-based detectors.

3.5 Summary

In this chapter, three key observations obtained by appropriately designed experiments and a theoretical analysis are presented. The first observation is that feature quality is significantly affected by compression artifacts at low bitrates. The H.265/HEVC Intra-encoded images show much better performance than the H.264/AVC Intra- and JPEG-encoded images. Then, the sensitivity of features detected at different scales is tested, and the experimental results lead to the second observation that large-scale features are more robust than small-scale features. In addition, a further experiment shows that if the original keypoints were given by an oracle, the corresponding extracted descriptors are much more robust. This leads to the third observation that the detection process is more essential in feature extraction. Finally, several detectors are analyzed theoretically. It is found that for the scale-space-based detectors, high image frequencies are less important because they are filtered by the Gaussian kernel in the first step of detection. Based on these observations, different approaches for feature-preserving image and video compression are presented in the following chapters.

Chapter 4

Feature-preserving image compression

This chapter presents the contributions to feature-preserving image compression. We consider the scenario that a mobile device captures a still image, and encodes and transmits it to a server. The JPEG standard is widely used and continues to dominate the consumer electronics world and the World Wide Web, owing to its good compression ratio and computational simplicity. Thus, it is used as an example compression standard. Note that a few of the proposed ideas can also be applied to other image compression standards. Section 4.1 details the results of a study, which investigates the impact of JPEG compression on the performance of an image retrieval system for different feature detector-descriptor combinations. The results show that among all tested detectors, the Hessian-Affine detector is the most robust performer in the presence of strong JPEG compression. Additionally, to improve the retrieval performance, the JPEG-encoded query images are post-processed using different deblocking filters. The retrieval gains of the different detector-descriptor pairs are compared. The results show that for some features, the retrieval performance benefits from two of the deblocking approaches at low bitrates. Section 4.2 presents our work on image encoding in the mobile device. Because the information sink is not a person but a computer algorithm, the image encoder in the mobile device should control the encoding process such that the important and relevant features of the image are preserved after compression. Since large-scale and small-scale features have different sensitivities to compression artifacts as observed in Section 3.2, the image macroblocks (MBs) are categorized into several groups based on the feature scales. Then, a novel rate-distortion model based on the SIFT feature-matching score is proposed. The dependency between the quantization table in the JPEG file and the common Lagrange multiplier is obtained from a training image database. Thereafter, for a given image quality, this relationship to perform R-D optimization for each group is exploited. The results show that the proposed algorithm achieves better feature preservation when compared to standard JPEG encoding. In Section 4.3, according to the previous theoretical analysis of scale-space-based detectors, a novel quantization table based on the impact of scale-space processing on the discrete cosine transform (DCT) basis functions is proposed. The proposed table employs large quantizers for high frequencies, and the experimental results show that the novel quantization table outperforms the JPEG default quantization table in terms of feature repeatability, number of correspondences, matching score, and number of correct matches. Figure 4.1 shows a typical mobile visual search architecture as well as the relevant sections in the diagrams.

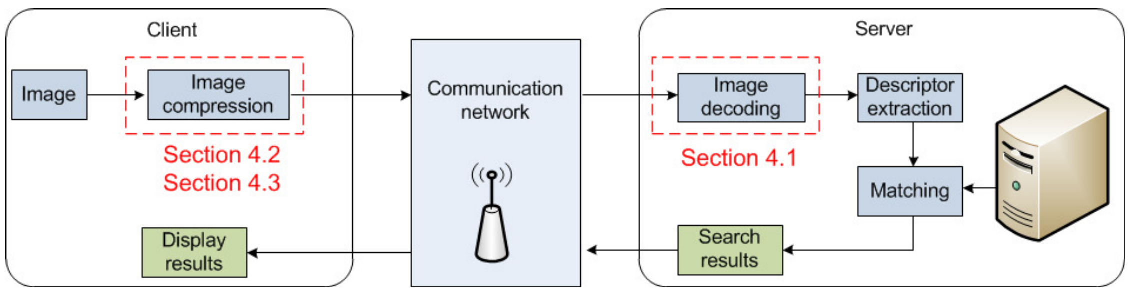

Figure 4.1: Architecture of image encoding and transmission for mobile visual search.

4.1 Image post-processing

In this section, we describe the effects of low-bitrate JPEG compression on the performance of an image retrieval system. For several features proposed in the literature, the impact of JPEG compression on feature-matching performance has already been evaluated [MS05, LCS11, FWH12, WFW11]. The evaluations were based on the criteria and the dataset provided by Mikolajczyk et al. [MTS$^+$05], focusing on the repeatability or the matching score between uncompressed and compressed image pairs.

The extent of influence of the compression artifacts on the performance of a real image retrieval system is addressed for the first time in this work. This study differentiates itself from other works by involving many currently available detectors and descriptors, as opposed to limiting the study to a few of the most established features. Moreover, we opt for a system-level comparison, thus making sure to study the impact of image compression on the overall performance of a retrieval system. In addition, possible improvements in retrieval are investigated by applying different deblocking filters to JPEG-encoded query images. Another important feature of this study is its easy reproducibility because a publicly available benchmarking framework is used, namely, the VLBenchmarks [LGV12]. This retrieval benchmark closely follows the work on the descriptor-distance-based retrieval scheme [JDS11] introduced in Section 2.4.2.

4.1.1 Detector and descriptor combinations

The detectors mentioned in Section 2.1.1, namely, Hessian-Affine, Harris-Affine, MSER, DoG, SURF, STAR, BRISK, MFD, WαSH, and KAZE, are included in this evaluation. As descriptors, SIFT, PCA-SIFT, shape context, GLOH, SURF, MROGH, and LIOP are used. The VLBenchmarks deals with real-valued descriptors. Hence, binary descriptors (e.g., BRISK) are not included in our experiments.

4.1.1.1 Experimental settings

In the following experiments, the Oxford Buildings dataset is used. It should be noted that the images are already compressed with JPEG, although at high quality. The features evaluated in our work are extracted from the Y component, so all images are first transformed to the YCbCr domain. In our experiments, these gray scale (Y component) images are referred to as *original* images. The number of nearest neighbors for each query descriptor during retrieval is set to 100, meaning that the nearest 100 descriptors for a given query descriptor are returned. The parameters *OkImagesNum*, *JunkImagesNum*, and *BadImagesNum* are set to *inf*; thus, all images in the Oxford Buildings dataset are included in our experiments. In VLBenchmarks, *queryNum* is set to 8 by default to obtain the precision-recall curve. The benchmark determines the mAP score, which is obtained by averaging the mean precision under the precision-recall curve over all query images.

4.1.1.2 Implementation and parameters for the features

Typically, there exist several implementations of detectors and descriptors. We find that the original implementations by the authors typically perform better than 3rd party implementations, and hence use the original implementations whenever available. In addition, the number of features influences the retrieval performance. In general, the greater number of features detected, the better are the results produced by the retrieval system. In our experiments, feature-extraction speed is not considered. The parameters of each detector are tuned to generate approximately 500 features per image for the 55 query images. Table 4.1 lists the noteworthy parameters for each detector and the corresponding source of implementation used in our work. Note that the DoG parameters of the authors' implementation can not be tuned; thus, we use *VlFeatSift* in our experiments.

The executables of SIFT, PCA-SIFT, shape context, and GLOH descriptors are provided by [MTS$^+$05]. The MROGH and LIOP descriptor calculations are done using the executables provided by the authors. These implementations use a descriptor region that is three times larger than the detector region. The enlarged elliptical descriptor region is mapped into a circular region to obtain affine-invariance. The patch is then scaled into a patch measuring 41 pixels in length, and the feature descriptor is calculated. Therefore, these descriptors are

Table 4.1: Selected parameters of the studied detectors.

Detectors	Parameters	Values	Source
Hessian-Affine	threshold	640	Authors [MTS+05]
Harris-Affine	threshold	2200	
MSER	es	1	
	mm	8	
DoG	PeakThresh	4.6	VLBenchmarks [LGV12]
	firstoctave	0	
SURF	threshold	56000	Authors [BTG06]
STAR	responseThreshold	26	OpenCV 2.4.3 [Ope]
BRISK	threshold	76	Authors [BTG06]
MFD	ff	2	Authors [AR11]
WαSH	t	0	Authors [VRA12]
Kaze	dthreshold	0.00175	Authors [ABD12]

comparable. In summary, in this experiment we evaluate 10 different feature detectors and combine them with 6 different descriptors, leading to a total of 10 × 6 detector-descriptor combinations to be evaluated. In addition, the SIFT (*VlFeatSift*), SURF (original implementation), and Kaze (original implementation) features, which detect keypoints and produce feature vectors at the same time, are compared.

4.1.1.3 Results of different detector-descriptor combinations

The mAP scores of the original query images are summarized in Table 4.2. The left part in the table shows the names of the detectors (column 1), the average numbers of features of the database images (column 2), and the average numbers of features of the query images (column 3). Although the parameters of the detectors are tuned to produce approximately 500 features for each of the 55 query images, the number of descriptors in the database images varies significantly. It should be noted that the WαSH detector can not detect a large number of features, even when the parameter t is set to 0. From Hessian-Affine to WαSH in the upper half of the table, the detectors are affine-invariant, and from DoG to Kaze in the lower half, they extract circular regions as keypoints. The columns "SIFT - LIOP" present the mAP scores of each detector-descriptor combination. For the affine-invariant features, the descriptors obtained affine invariance by mapping the regions into circular ones. Generally speaking, the affine-invariant detectors perform better than their non-affine-invariant counterparts. Among them, MSER performs the best, followed by MFD in our experiments. The authors of the MFD, WαSH, and Kaze detectors showed that their detectors outperformed DoG, SURF, Hessian-Affine, and STAR. However, the results obtained with our dataset suggest that MSER is the best performer. There are several reasons for the different results obtained in our experiments. First, we find that different implementations of one specific detector produce remarkably different results, e.g., *VlFeatMser* in VLBenchmarks performs significantly worse than the author's implementation we use here. This phenomenon is also true when using the OpenCV implementations of SIFT and SURF. This is one important reason for the different rankings of the detectors in our results. Second, the parameters used in our experiments are different than those used in other works. In addition, the image retrieval framework and its setup are different.

As for the descriptors, in [FWH12, WFW11] and the evaluation work of [MM12], MROGH and LIOP were the best performers. However, those evaluation studies were based on the dataset of Mikolajczyk [MTS$^+$05]. Compared to this dataset, the Oxford Buildings dataset used here contains more realistic image changes such as lighting change, perspective change, scale change, and camera photographic change. Under such severe conditions, the detected locations, scales, orientations of keypoints, and pixel intensities vary to a large extent. Thus, the segmentation techniques in MROGH and LIOP are affected strongly, leading to poor performance. Surprisingly, the classic SIFT descriptor stands the test of time and is superior to all other descriptors. A comparison of the SIFT, SURF, and Kaze features is summarized in Table 4.3. *SIFT** means the descriptors, too, are extracted using *VlFeatSift*. Compared to the SIFT descriptor extraction of [MS05] in Table 4.2, its performance increases from 0.556 to 0.592 because the radius of the region used for computing the descriptor is six times larger than the detected scale, as opposed to three times in [MS05]. The Kaze feature does not show superior results in our experiments.

Table 4.2: mAP scores (%) of original query images using various detector-descriptor combinations

Detectors	avg. #db	avg. #qry	SIFT	PCA-SIFT	GLOH	Sha.-con.	MROGH	LIOP
Hess.-Aff.	926.7	496.1	**61.6**	53.8	59.8	56.1	48.2	46.4
Harr.-Aff.	952.6	495.5	**56.6**	48.7	56.4	51.8	47.4	46.1
MSER	775.7	461.9	**69.4**	57.5	66.1	64.5	50.7	51.5
MFD	1002.2	486.6	**63.9**	44.1	58.0	59.3	41.6	43.6
WαSH	454.5	216.4	**53.0**	39.5	46.3	48.1	31.5	37.1
DoG	872.9	500.2	**55.6**	49.0	52.9	47.1	53.9	40.8
SURF	851.2	507.6	**60.6**	53.4	59.0	54.0	53.1	46.1
STAR	783.1	513.7	**56.6**	49.6	54.8	50.2	51.6	42.4
BRISK	1006.4	506.7	**52.8**	41.2	49.9	48.1	41.2	36.2
Kaze	915.6	509.8	**53.4**	46.8	52.3	47.1	49.3	39.3

Table 4.3: mAP scores (%) of original query images using SIFT, SURF, and Kaze features

Detectors	avg. #db	avg. #qry	SIFT*	SURF	Kaze
DoG	872.9	500.2	**59.2**		
SURF	851.2	507.6		57.6	
Kaze	915.6	509.8			52.7

4.1.2 Strongest feature under compression

To examine the effects of JPEG compression on the retrieval results, we use the software from [Gro] to compress the 55 query images with quality values of 4, 8, 12, 16, and 20. The mAP scores of each detector-descriptor combination is computed by averaging the results of the 55 JPEG-encoded query images encoded at the same quality level.

First, we are interested in the robustness against strong JPEG compression. We focus on the SIFT descriptor because it yields the best image-retrieval performance for the original query images. In addition, we consider shape context and MROGH for comparison. From Table 4.4, it can be seen that the Hessian-Affine detector is quite robust. The mAP score of the Hessian-Affine detector drops from 0.616 to 0.547, which is the highest when compared to the other detectors for quality value 4. By contrast, the mAP score of the MSER detector is only 0.413, indicating that it is strongly affected by the compression artifacts. This is because blocking artifacts generate many unwanted keypoints during the execution of the watershed segmentation algorithm for images with strong JPEG encoding. For quality values 12 and higher, the MSER detector shows its superior performance again.

Table 4.4: mAP scores (%) of query images compressed at different qualities

Detectors	SIFT	SC	MROGH	SIFT	SC	MROGH	SIFT	SC	MROGH
	Quality 4			Quality 8			Quality 12		
Hess.-Aff.	**54.7**	48.3	40.1	**60.3**	54.2	46.3	61.2	55.8	46.9
Harr.-Aff.	48.9	43.3	38.2	54.7	49.4	43.7	55.9	50.4	45.5
MSER	41.3	34.2	25.6	57.6	49.9	39.2	**63.3**	55.8	44.1
MFD	47.0	42.3	28.0	56.7	52.0	34.3	59.3	54.7	37.5
WαSH	45.2	40.9	25.4	50.8	45.8	29.5	50.0	45.6	30.2
DoG	29.7	23.9	33.3	40.6	34.7	45.9	47.1	39.7	50.0
SURF	47.4	41.8	40.3	56.4	49.6	50.0	57.7	51.6	51.1
STAR	38.9	32.0	36.3	48.2	41.7	45.2	51.9	43.1	48.5
BRISK	41.5	37.9	29.4	46.5	42.1	37.0	49.4	45.0	39.2
Kaze	37.7	32.7	36.3	47.4	41.9	46.2	50.8	44.7	48.4
	Quality 16			Quality 20					
Hess.-Aff.	60.9	55.4	47.3	61.3	55.8	47.7			
Harr.-Aff.	55.5	50.3	45.8	56.9	52.0	46.3			
MSER	**65.5**	58.8	46.1	**67.6**	61.4	47.7			
MFD	61.0	57.0	38.9	61.2	57.6	39.3			
WαSH	52.4	48.1	31.0	52.5	48.6	31.4			
DoG	49.1	41.0	51.4	49.8	42.2	51.4			
SURF	59.6	52.3	52.9	59.7	53.2	52.3			
STAR	53.1	46.3	50.2	54.1	46.9	50.8			
BRISK	50.2	45.8	39.7	51.0	46.6	40.2			
Kaze	52.0	45.0	48.4	52.3	45.2	49.3			

4.1.3 Applying deblocking filters

The post-processing approaches for removing blocking artifacts can be divided into three broad categories: spatial domain, DCT frequency domain, and hybrid filtering methods. Pham et al. [PvV05] proposed a hybrid filtering method, which involves a soft blend of edge preservation and low-pass filtering of JPEG-encoded images. In our experiment, this deblocking filter is called *DIP* approach. Foi et al. [FKE07] proposed a pointwise shape-adaptive DCT (*SA-DCT* approach) for filtering blocking artifacts, which is also used in our experiments. Nosratinia et al. [Nos01] averaged shifted versions of already compressed images to reduce coding artifacts, which is named as *REAPP* approach. The default settings of all these deblocking filters are used in our experiments. Figure 4.2 shows an example of the zoomed-in query image *all_souls_000013*. These deblocking approaches target the improvement of visual quality, but it has not been studied whether they have a positive impact on the performance of a feature-based image retrieval system.

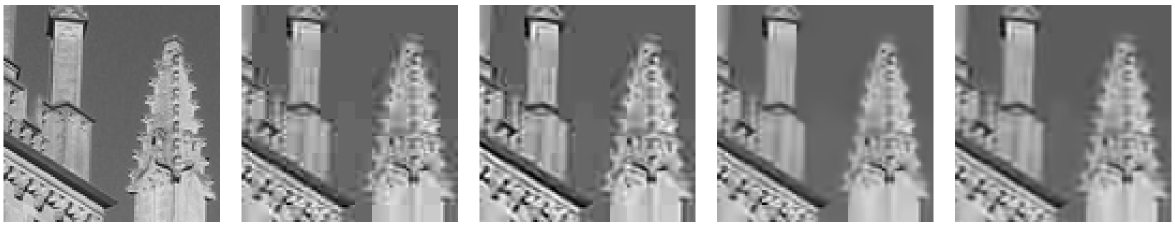

Figure 4.2: From left to right: Original image, JPEG-encoded image with quality value 12 (PSNR = 28.4 dB), DIP-deblocked image (PSNR = 27.8 dB), SA-DCT-deblocked image (PSNR = 29.2 dB), and REAPP-deblocked image (PSNR = 28.9 dB).

Thus, subsequently, the three different deblocking approaches are applied to the 55 query images, and the corresponding mAP score is determined. First, PSNR and SSIM are calculated to examine whether the deblocking approaches work and which one performs the best in terms of image quality. Figure 4.3 shows that the SA-DCT approach performs the best, followed by the REAPP approach for all quality values in terms of both PSNR and SSIM. The DIP approach performs worse than JPEG at high bitrates in terms of PSNR, but for all quality values, it is better in terms of SSIM.

The effects of the three deblocking approaches are compared. In Table 4.5, the white, green, and yellow backgrounds represent for the three deblocking approaches the same, increased, and decreased mAP scores compared to the normal JPEG-encoded images in Table 4.4, respectively. For MSER, REAPP increases the performance at quality values of 4, 8, 12, and 16, with mAP gain increasing with the increasing of the compression rate. The performance of the MFD and WαSH detectors is also improved by the deblocking approaches SA-DCT and REAPP at the lower quality values 4 and 8. However, the deblocking approach

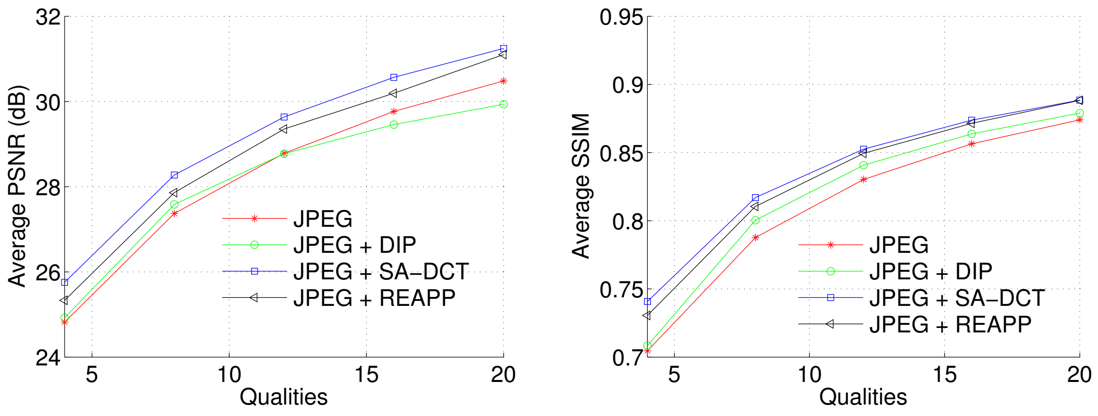

Figure 4.3: Comparison of PSNR and SSIM for different deblocking filters

DIP degrades the performance in most cases. For the other detectors, the performance decreases after deblocking. The results for quality value 20 are already close to the results of the original query images. Thus, the mAP scores of normal JPEG images, DIP-, SA-DCT- and REAPP-deblocked images show no big differences. In addition, the shape context and MROGH descriptors are compared beside the SIFT descriptor. For the low quality values, both SA-DCT and REAPP have a positive impact on the MROGH descriptor in most cases. This can be explained by the fact that SA-DCT and REAPP remove many strange pixels caused by compression artifacts in patches, as can be seen in Figure 4.2. Thereafter, the subsequent pixel intensity-based processes of sorting and segmentation in the MROGH algorithm are more accurate. The SA-DCT and REAPP approaches positively influence the MSER, MFD, and WαSH detectors for low-bitrate query images as well.

Table 4.5: mAP scores (%) of query images after deblocking (the white, green, and yellow backgrounds represent the same, increased, and decreased mAP scores compared to the normal JPEG-encoded images in Table 4.4)

Detectors	SIFT			Shape context			MROGH		
Quality 4	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP
Hess.-Aff.	50.4	53.4	54.0	44.4	46.9	48.1	38.9	41.3	40.5
Harr.-Aff.	43.7	48.3	48.7	38.9	42.8	43.7	36.3	38.9	38.8
MSER	39.3	42.0	42.8	31.6	37.3	37.3	31.4	33.1	30.9
MFD	46.5	49.9	50.6	40.6	44.4	45.7	29.5	30.0	28.9
WαSH	45.1	46.4	45.7	39.9	40.7	41.0	27.1	25.9	25.9
DoG	29.5	27.5	24.8	23.9	24.4	22.5	32.8	36.0	33.6
SURF	45.3	48.2	47.7	39.4	41.4	41.3	40.6	43.9	42.4
STAR	34.6	35.7	34.4	28.6	28.9	27.6	35.0	38.8	38.0
BRISK	36.7	38.5	39.1	31.8	33.9	33.9	27.0	28.6	27.3
Kaze	36.5	36.6	37.0	32.7	32.1	31.7	36.7	40.1	38.3
Quality 8	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP
Hess.-Aff.	58.5	59.2	59.5	52.7	52.7	53.4	44.8	46.3	45.9
Harr.-Aff.	52.6	53.0	54.0	47.3	48.0	48.4	42.5	44.5	43.9
MSER	57.7	58.1	59.1	49.4	50.6	52.0	41.2	42.0	41.4
MFD	56.9	56.7	57.6	51.6	52.0	53.3	35.1	35.1	35.0
WαSH	50.1	50.9	51.2	45.6	45.8	46.6	30.3	30.0	30.4
DoG	40.0	37.9	38.3	33.8	33.2	34.0	45.6	46.7	45.2
SURF	54.3	54.4	56.3	47.4	48.0	48.5	49.0	51.1	50.0
STAR	45.3	46.2	45.7	37.6	39.3	38.9	45.1	46.8	45.7
BRISK	45.5	44.4	45.1	41.3	40.2	40.9	36.2	36.4	34.7
Kaze	46.2	46.4	47.8	40.9	39.7	41.9	45.8	47.1	46.6
Quality 12	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP
Hess.-Aff.	61.5	60.5	61.2	55.2	54.8	55.6	46.2	46.9	46.5
Harr.-Aff.	55.6	55.3	55.8	49.6	50.1	50.7	45.0	45.7	45.3
MSER	63.0	62.8	63.9	55.4	55.4	56.4	44.5	46.0	45.3
MFD	57.9	58.4	59.2	54.1	53.5	55.4	36.5	37.9	37.8
WαSH	51.7	51.1	51.2	47.0	46.4	46.4	31.5	32.2	30.5
DoG	45.7	44.4	44.4	38.1	38.1	38.3	49.1	50.0	49.8
SURF	56.6	56.8	57.6	49.8	50.2	51.1	50.2	52.3	51.4
STAR	50.3	49.6	49.8	42.4	42.5	42.2	48.4	49.1	48.9
BRISK	49.2	48.7	49.1	45.2	43.0	43.8	39.1	37.9	37.7
Kaze	50.1	49.4	50.3	43.2	43.3	44.0	47.3	48.2	47.5
Quality 16	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP
Hess.-Aff.	60.9	60.5	60.5	55.1	54.9	54.7	47.1	47.5	47.2
Harr.-Aff.	55.9	55.3	55.9	49.9	49.9	50.1	45.4	46.2	45.9
MSER	64.9	65.5	65.9	57.3	58.1	58.6	44.6	46.0	47.0
MFD	59.4	60.1	60.2	55.8	55.8	56.0	37.9	38.5	39.1
WαSH	50.7	51.6	52.0	46.4	46.5	47.2	30.3	31.4	31.3
DoG	48.9	46.9	47.5	40.1	39.0	39.8	50.6	51.0	51.1
SURF	58.6	58.3	59.6	50.8	51.5	51.9	51.7	52.6	52.5
STAR	51.9	50.7	51.7	44.4	43.9	44.7	49.2	50.6	50.3
BRISK	50.1	49.0	48.9	47.0	44.2	44.0	39.2	39.2	38.6
Kaze	51.5	50.9	51.9	44.6	43.5	44.7	48.0	48.7	48.7
Quality 20	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP	DIP	SA-DCT	REAPP
Hess.-Aff.	62.1	61.0	61.5	55.9	55.2	56.1	47.2	47.7	47.5
Harr.-Aff.	56.6	56.6	56.6	50.6	51.3	51.5	45.3	46.2	46.0
MSER	65.7	66.7	66.8	58.9	60.7	60.9	45.7	47.7	47.7
MFD	60.5	60.1	61.0	56.5	57.1	57.2	38.3	39.3	39.4
WαSH	52.2	52.7	51.7	47.2	48.3	47.4	30.7	32.8	32.1
DoG	49.4	48.0	49.2	41.8	41.1	41.7	51.3	51.4	51.5
SURF	58.8	58.5	59.2	51.5	51.7	52.5	51.8	52.5	52.4
STAR	53.3	52.9	53.6	45.4	45.9	46.2	50.0	50.5	51.0
BRISK	50.3	50.4	49.4	47.1	45.9	45.0	40.2	40.1	39.1
Kaze	52.0	51.2	52.2	45.4	44.3	44.7	48.1	49.3	49.1

4.2 Novel R-D optimization of JPEG-encoded image

Although Hessian-Affine + SIFT is the most robust feature under compression, the classic SIFT feature (DoG detector and SIFT descriptor) is mainly used as an example feature in the following experiments. In this section, we present the work pertaining to the design and validation of a rate-distortion optimization strategy for image compression. The human perception-oriented distortion measure used traditionally is replaced by a distortion measure that helps us preserve SIFT features. To this end, we modify encoder control; specifically, the quantization and bit allocation problems are addressed. We use the available bit budget such that feature detection algorithms applied to uncompressed and compressed images produce as many identical SIFT features as possible.

The work in this section is motivated by the observation that the sensitivity is different for features with different scales, as mentioned in Section 3.2. Under the experimental settings as in Section 3.2, the VLFeat software [VLF] is used to extract SIFT features. The images are compressed using the JPEG codec from [Gro] with variable quantization according to the JPEG standard extension syntax [IT96]. The test images are from [MTS+05], and we aim to preserve the 200 strongest features from these images. The matching score between the uncompressed and compressed image pairs is used as the performance metric.

4.2.1 Bit allocation for blocks without relevant features

In the process of feature extraction, the SIFT descriptor is computed from a 4×4 array of histograms with 8 orientations. The size of the array in the image is mσ, where σ is the scale of the feature and m = 3.0. Hence, the SIFT descriptor is calculated from a square with an edge length of 12σ pixels. We start our evaluation with a simple test: The blocks within 6σ radius of a feature are compressed using normal compression quality, while other parts that contain no features are compressed at much lower quality. These areas, which are irrelevant for feature extraction, could even be deleted.

To validate our idea, we compress all relevant MBs at the same quality level (e.g., 50) and the irrelevant MBs using worst quality. For the sake of compatibility with the JPEG extension (variable quantization [IT96]), blocks that do not contribute to the strongest features are compressed using non-linear Q_SCALE and SCALE_CODE = 31. Note that if any other compression standard is to be used, we could similarly use the strongest compression for the irrelevant MBs. Figure 4.4 shows the matching score of traditional JPEG and the simple bit allocation method described above (from now on referred to as “approach 1” in this section) as a function of the bitrate. The matching scores are not influenced as the bitrates decrease significantly. For example, the bitrate of the normally compressed image *graf* is 0.9817 bpp (middle point, quality level 50), and the corresponding matching score is 96.5%. In comparison, the bitrate of the image compressed using the simple feature-preserving approach is 0.7413 bpp, and the corresponding matching score is 96.5%. Of course, this is expected because SIFT features could even be extracted and matched individually. Therefore, we can allocate a greater number of bits to areas containing SIFT features, and other blocks use the worst quality. This represents our first step in bit allocation.

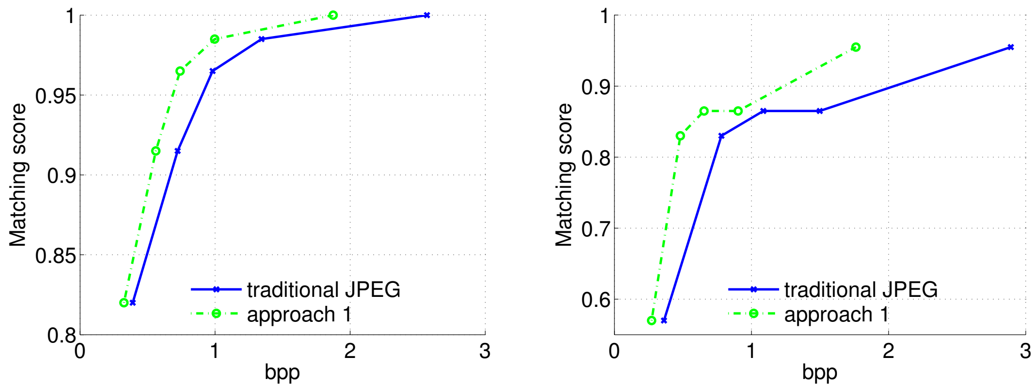

Figure 4.4: Matching scores as a function of bitrate. Blue: traditional JPEG images; Green: approach 1 for bit allocation (Upper: test image *graf*; Lower: test image *bark*)

4.2.2 Categorization of blocks containing features

Based on the observation on the sensitivity of features at different scales in Section 3.2, blocks can be categorized according to feature octaves as follows:

Step 1. Calculate SIFT features in each octave separately.

Step 2. Find the blocks containing features in the first octave; tag these blocks as group 1.

Step 3. Find the relevant blocks in the next octave. If the blocks have not been tagged before, tag them as a new group.

Step 4. Repeat step 3 until the last octave level is reached.

In our experiment, for the *graf* and *bark* test images, 4 and 5 octaves contain features, respectively. Consequently, 4 and 5 groups are obtained. The reasons for this categorization are as follows. First, the matching score cannot be increased by changing the quantizer of a single block. We have to simultaneously adjust the quantizers of all blocks relevant to one feature together. Second, small-scale features are more vulnerable to compression, and they should be treated preferentially. Third, small-scale features require higher quality after optimization of quantization, and the relevant blocks' information will be well retained. The possible overlapping parts with large-scale features would not influence the successive allocation process for large-scale features.

4.2.3 Rate-Distortion Optimization

To distribute the bit budget in an even better manner among the blocks that contain SIFT features or parts thereof, an improved rate-distortion optimization strategy can be applied. Instead of simple block-by-block bit allocation, group-by-group bit allocation is performed considering the sensitivity of features at different scales.

4.2.3.1 Rate-Distortion optimization theory [Ort96]

Ortega [Ort96] presented a scheme for optimal bit allocation under multiple rate constraints. If one aims to encode N input blocks using a given set ($\mathcal{Q}$) of admissible quantizers, for a given block i, the selection of quantizer j generates a rate r_{ij} while incurring a distortion cost d_{ij}. Here, the goal is to find a suitable quantizer $x(i)$ for each block i such that the total distortion is minimized for a given rate constraint R_{budget}. Here, quantizer allocation is represented as the vector $\mathbf{x} \in \mathcal{X} = \mathcal{Q}^N$ [Ort96]. Then, the optimal solution to budget-constrained bit allocation is finding an $\mathbf{x}_\mathbf{u}^*$ such that

$$\mathbf{x_u}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} d_{ix(i)} \tag{4.1}$$

subject to

$$\sum_{i=1}^{N} r_{ix(i)} \leq R_{budget} \tag{4.2}$$

This bit allocation problem can be solved using a Lagrangian optimization approach. The Lagrangian cost function associated with a solution $\mathbf{x}$ is defined by

$$J_i(\lambda, \mathbf{x}) = d_{ix(i)} + \lambda r_{ix(i)} \tag{4.3}$$

where λ, a non-negative real value, is the Lagrangian multiplier. Given a certain value λ, one can find the solution to the following function:

$$\mathbf{x}_\lambda{}^* = \arg\min_{\mathbf{x}} \sum_{i=1}^{N} J_i(\lambda, \mathbf{x}) \tag{4.4}$$

where the solution $\mathbf{x}_\lambda{}^*$ is also the solution to Equation (4.1). The rate constraint R_{budget} can then be expressed as follows.

$$R_{budget} = R^*(\lambda) = \sum_{i=1}^{N} r_{ix^*_\lambda(i)} \tag{4.5}$$

The selection of a quantizer for each block i can be performed independently. As pointed out by [Ort96], if the selected quantizers correspond to the same rate-distortion trade-off for each block, the optimal solution for a given budget is obtained. The trade-off is determined by λ. Minimization of Equation (4.3) corresponds to selection of an operating point that first touches the line of absolute slope determined by the value λ. This idea is shown in Figure 4.5. The author called this optimal solution a "constant slope" solution [Ort96]. Then, the problem of the budget-constrained bit allocation is running various λ values and finding an appropriate λ such that $R^*(\lambda)$ meets the budget requirement. In the following experiment, the specific values of rate constraints are not considered, and we only use several different λ values to show the improvement brought about by the proposed approach.

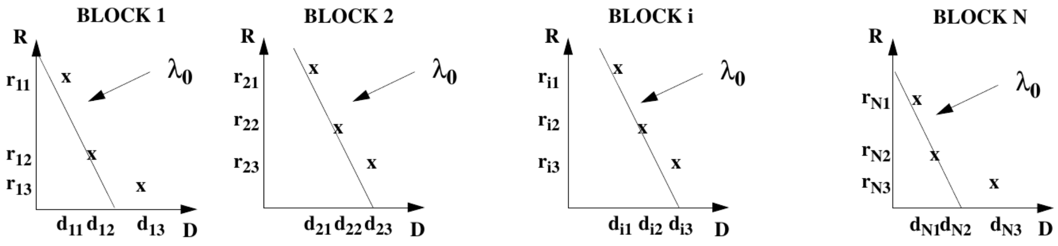
BLOCK 1
BLOCK 2
BLOCK i
BLOCK N
R
D
λ_0
r_{11} r_{12} r_{13}
d_{11} d_{12} d_{13}
r_{21} r_{22} r_{23}
d_{21} d_{22} d_{23}
r_{i1} r_{i2} r_{i3}
d_{i1} d_{i2} d_{i3}
r_{N1} r_{N2} r_{N3}
d_{N1} d_{N2} d_{N3}

Figure 4.5: Example of Lagrangian-based bit allocation. The constant slope λ_0 yields the optimal rate-distortion performance for the given blocks [Ort96]. ©1996 IEEE.

4.2.3.2 Novel distortion metric

The JPEG codecs [Gro] provide a selection of image qualities (Q) from 1 to 100 which correspond to 100 values used to scale the default quantization table. The final quantization table in the JPEG file is generated according to the selected scaling factor. The approximate relationship between the recommended scales (S) and image qualities in [Gro] is as follows:

$$S = \begin{cases} 50/Q & \text{if } Q < 50, \\ 2 - 0.02 * Q & \text{if } Q \geq 50. \end{cases} \tag{4.6}$$

According to the JPEG standard extension [IT96], only one scale S is applied to the default quantization table to generate the final quantization table used for encoding an image. To obtain variable quantization, each MB is assigned a SCALE_CODE (SC), which indicates the scaling applied to the final quantization table. Note that only the quantizers for AC coefficients are determined by the SCALE_CODE, and the quantizers for DC coefficients are taken from the quantization table. Consequently, the two variables can be tuned during image encoding: the scale S that is determined by Q and the SCALE_CODE for each MB. This case is similar to traditional video encoding. The video coder is required to adjust several mutually dependent parameters simultaneously, e.g., the quantization parameter (QP) and the different modes of each MB. Sullivan et al. [SW98] found an approximate relationship between the λ_{MODE} and the QP value. The optimum mode can then be obtained from the rate-distortion curve of different modes using λ_{MODE}. Note that the variable to be selected is the mode of each MB rather than the quantizer of each block shown in Section 4.2.3.1, and λ_{MODE} is constant across different MBs. This leads to extensive studies on the use of Lagrangian techniques for controlling hybrid video coders. In our experiment, we use *non-linear Q_SCALE* [IT96], where the MB has the same quantizers as the quantization table if its SCALE_CODE (*SC*) equals 12. Similar to the case in [SW98], a quantization control strategy is designed as follows.

We assume that the matching score is directly related with the DCT coefficient quantizer and each group is independent. Next, we propose a novel distortion measure and design an encoding mode and a quantization control strategy. If the compressed image has the exact same set of features (i.e., no distortion of features), the matching score should be 1; else, the score should be smaller. We treat (1 - matching score) as our distortion metric. Blocks with SIFT features are compressed using the following rate-distortion model.

$$D_{ms} = 1 - matching\ score \tag{4.7}$$

$$J = D_{ms} + \lambda_{SC} R = \sum_{i=1}^{N} D_i + \lambda_{SC} (\sum_{i=1}^{N} R_i) \tag{4.8}$$

where N is the number of groups, D_i is the distortion of the i-th group, and R_i is the rate of the i-th group. Since the groups are assumed to be independent, the total distortion is the sum of the individual distortions J_i.

$$J = \sum_{i=1}^{N} J_i = \sum_{i=1}^{N} (D_i + \lambda_{SC} R_i) \tag{4.9}$$

Equation (4.8) is optimized simply by minimizing the cost function separately for each group using a common Lagrange multiplier λ_{SC}.

4.2.3.3 Lagrange Multiplier

First, a training database is used to determine the relationship between the common λ_{SC} and the image qualities following the method described in [SW98]. The uncompressed image database is taken from [SS04], where the first 1200 color images are encoded at all quality levels. The matching score for each quality and the bitrate are recorded. Note that the matching score is discrete because it is calculated from the ratio of correct matches to the number of original features of an image pair. For a fixed λ_{SC} value, the Lagrangian cost function in Equation (4.8) is minimized to obtain the optimum quality Q of an image. Figure 4.6 shows the relative frequency of the optimum qualities Q for the Lagrange multiplier λ_{SC} varied over nine values: 2, 1.5, 1, 0.5, 0.3, 0.15, 0.1, 0.05, and 0.01. From the chart, we can see that for high and low qualities there is a close relationship with λ_{SC}. The middle λ_{SC} values span a large range of qualities. In our experiment, by contrast, when we optimize the image, for a given quality level, the matching score does not change significantly even when λ_{SC} varies considerably within a certain range. In other words, the image qualities seem not very sensitive to λ_{SC} within a specific range. Hence, it is sufficient to obtain an approximate relationship between the quality levels and λ_{SC}.

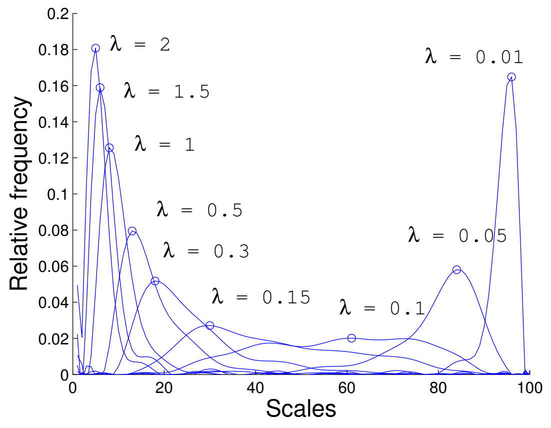

Figure 4.6: Relative frequency of chosen optimum qualities vs. scales

Next, the JPEG qualities are mapped to real scales (S), which are used to multiply the standard quantization table. Each tested λ_{SC} value corresponds to a unique quantization scale S having the highest relative frequency. These data points are used to fit a curve. The fitted curve, shown in Figure 4.7, is an approximation of the relationship between the scale and the Lagrange multiplier:

$$\lambda_{SC} \approx 0.012 \cdot S^2 + 0.08 \cdot S \tag{4.10}$$

For small scales, λ_{SC} tends to be linear with S; for large scales, λ_{SC} tends to be quadratic with S. A default quantization table is included in the JPEG file according to the input quality level. Before performing R-D optimization using Equation (4.8), we use Equation (4.10) to calculate the λ_{SC} for the given quality level.

Then, SC is selected from 9 to 15 to confine the bit budget. Hence, individual J_i are minimized as follows:

$$SC_i^* = \arg\min_{SC} J_i, \;\; SC \in \{12, 12 \pm 1, 12 \pm 2, 12 \pm 3\} \tag{4.11}$$

The variable quantization is performed by tuning the SCALE_CODE so that the optimum SC_i for each group is obtained, thus minimizing the Lagrangian cost function. R-D optimization is first performed for group 1, which contains small-scale features, by adjusting SC_1 in Equation (4.11). The quantizers of the first group are fixed after finding the optimum SC_1. Then, R-D optimization is repeated for group 2. This continues until the last group.

4.2.4 Experimental results

Figure 4.8 shows the matching results of the two test images *graf* and *bark*. The blue curves denote the traditional JPEG images; green curves denote images after bit allocation with

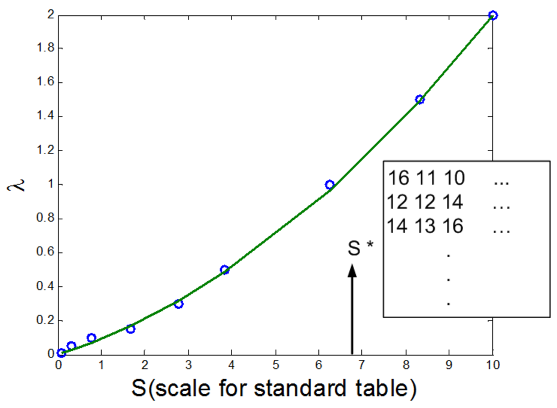

Figure 4.7: Lagrange multiplier vs. scale for standard table

approach 1; red curves denote images after R-D optimized bit allocation. Obviously, using the proposed R-D optimization, better performance is achieved compared to the traditional JPEG compression. We can also see an improvement compared to approach 1 introduced in Section 4.2.1. Similar improvements are observed for other images from [MTS$^+$05]. From the results, it is found that if the percentage of blocks without relevant features is high, approach 1 achieves most of the gains, while approach 2 leads to slightly incremental improvement over the results of approach 1. This is because the bitrate decreases substantially when many MBs employ the worst quality, and only a few MBs need tuning when performing R-D optimization. By contrast, approach 2 achieves larger gains than approach 1 when most of the blocks contain features. Hence, the two approaches finally achieve improved SIFT feature preservation performance compared to traditional JPEG encoding.

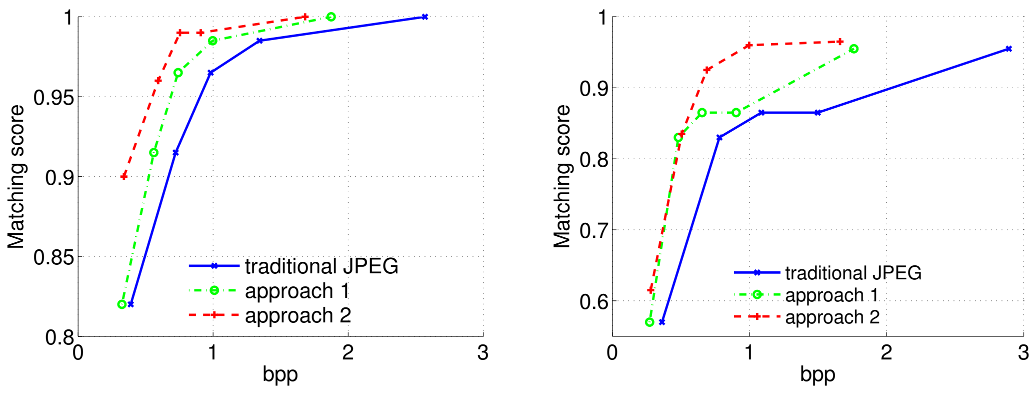

Figure 4.8: Results of our feature-preserving bit allocation for two different test images. (Upper: test image *graf*; Lower: test image *bark*)

4.3 Novel JPEG quantization table

In the previous section, the macroblocks in images are compressed using variable quantization, which is supported by the JPEG standard extension syntax [IT96]. The extension syntax is, however, not widely supported, and the encoding process is computationally expensive. Therefore, a simpler method is presented below.

JPEG compression is based on a default quantization table. However, it should be noted that the standard does not mandate a specific quantization table. Thus, alternative tables can be defined by users. A few works have attempted to design novel quantization tables targeting different applications. Change et al. [CWL99] proposed a quantization table based on the HVS, and the scheme based on this table outperformed baseline JPEG and many other schemes in terms of rate-distortion performance. An optimized quantization table scheme for individual images was proposed in [Wat93]. In [JKAA06] and [KS09], the recognition performances of iris and face recognition systems were improved by compressing images using novel quantization tables. Makar et al. [MLCG12] proposed encoding canonical patches with a gradient-preserving quantization matrix, which achieved lower gradient distortion and better descriptor matching performance. Duan et al. [DLC$^+$12] proposed a novel distortion measure and employed an evolutionary algorithm to develop a better quantization table for feature extraction.

In this work, our goal is to design a novel quantization table for improving the feature detection performance of scale-space-based detectors mentioned in Section 3.4, namely, Hessian-Laplacian, Harris-Laplacian, SURF, and DoG. The JPEG-encoded images in the proposed approach are completely standard-compatible, and they can be decoded by any baseline JPEG decoder. The computational complexity of both the encoder and decoder is the same as the default JPEG encoding. Additionally, the detection performance is improved.

4.3.1 Evaluation settings

As presented in Section 3.4, for scale-space-based detectors, e.g., Hessian-Laplacian, Harris-Laplacian, DoG, and SURF, the high frequencies are less important during detection. Thus, these detectors are used to perform the experiments. The criteria of repeatability score, number of correspondences, matching score, and number of matches, as well as the test dataset provided by the authors in [MTS$^+$05], are used. The open-source software VLBenchmarks [LGV12] employs this evaluation framework, hence, it is used in our experiments. The executable of the Hessian-Laplacian and Harris-Laplacian detectors is provided by the authors of [MTS$^+$05]. Table 4.6 lists a few noteworthy parameters of each detector and the source of the implementation used in this work. The parameter *PeakThresh* 7.65 results in the same threshold value 0.03 as in the classic paper [Low04], and the parameter *FirstOctave* is set to 0, yielding less features in order to be comparable to other features. Other parameters are set by default in the corresponding source codes. For calculating the matching score and the number of correct matches, the descriptors of the detected regions using the Hessian-Laplacian, Harris-Laplacian, and MSER detectors are computed using the SIFT descriptor implementation provided by [MTS+05]. The DoG detector in VLBenchmarks has its own SIFT descriptor implementation, and the SURF implementation yields both keypoints and corresponding descriptors.

Table 4.6: Selected parameters of studied detectors.

Detectors	Parameters	Values	Source
Hessian-Laplacian	threshold	500	Oxford [MTS+05]
Harris-Laplacian	threshold	1000	Oxford [MTS+05]
MSER	es	1	Oxford [MTS+05]
DoG	PeakThresh FirstOctave	7.65 0	VLBenchmarks [LGV12]
SURF	threshold	1000	ETH [BTG06]

4.3.2 Weights of DCT basis

As shown in the theoretical analysis in Section 3.4, all scale-space-based detectors first smooth the images using a low-pass Gaussian filter. The standard deviation σ_0 of the initial scale layer determines the cut-off frequency. The frequencies beyond this frequency are useless for feature detection. Here, the JPEG standard is used for the evaluation. We evaluate the impact of the Gaussian filter on the discrete cosine transform (DCT) basis functions for the sake of designing a novel quantization table. DCT transforms an image block (typically 8 × 8 pixels) into the frequency domain, and the DCT coefficients correspond to different spectral sub-bands. In the spatial domain, an image block can be represented by a linear combination of the DCT basis images multiplied by the DCT coefficients [Kha03].

$$f = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C_{(u,v)} F_{(u,v)} \tag{4.12}$$

where $C_{(u,v)}$ are the coefficients and $F_{(u,v)}$ are the DCT basis functions. Thus,

$$\begin{aligned} G_\sigma * f &= G_\sigma * (\sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C_{(u,v)} F_{(u,v)}) \\ &= \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C_{(u,v)} (G_\sigma * F_{(u,v)}) \end{aligned} \tag{4.13}$$

The energy of $G_\sigma * F_{(u,v)}$ is used as the importance score of the coefficient $C_{(u,v)}$. The energy of $A_{M\times M} = G_\sigma * F_{(u,v)}$ is computed as

$$\mathcal{E}(A) = \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} A_{ij}^2 \tag{4.14}$$

When the energy of $G_\sigma * F_{(u,v)}$ is zero, it means that this DCT basis function has no contribution after smoothing. Thus, it can be quantized using a coarse quantizer. Contrarily, large values imply high importance of this DCT basis function, which should be quantized using a fine quantizer.

Each detector uses a similar initial scale σ_0 to convolve the original image with different operators, e.g., $\sigma_0 = 1.2$ for Hessian-Laplacian, Harris-Laplacian, and SURF, and $\sigma_0 = 1.5199$ for DoG in [LGV12]. Therefore, we design the quantizers $Q_{(u,v)}$ according to $G_{1.2}$. The energy matrix of each coefficient is shown in Table 4.7.

Table 4.7: Energies from $G_{1.2} * F_{(u,v)}$.

0.6984	0.4652	0.2654	0.1050	0.0297	0.0065	0.0013	0.0002
0.4652	0.3098	0.1768	0.0700	0.0198	0.0043	0.0008	0.0001
0.2654	0.1768	0.1008	0.0399	0.0113	0.0025	0.0005	0.0001
0.1050	0.0700	0.0399	0.0158	0.0045	0.0010	0.0002	0.0000
0.0297	0.0198	0.0113	0.0045	0.0013	0.0003	0.0001	0.0000
0.0065	0.0043	0.0025	0.0010	0.0003	0.0001	0.0000	0.0000
0.0013	0.0008	0.0005	0.0002	0.0001	0.0000	0.0000	0.0000
0.0002	0.0001	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000

4.3.3 Novel quantization table

Then, the quantizers in our proposed quantization table are designed as follows.

$$Q_{(u,v)} = min\{round[s \cdot \frac{1}{\mathcal{E}(G_\sigma * F_{(u,v)})}],\ 255\} \tag{4.15}$$

where $\mathcal{E}$ is the energy calculation function. s is a scalar value used to ensure that the first AC quantizer $Q_{(0,1)}$ is equal to the one in the default table (value 11) because standard JPEG applies differential pulse code modulation (DPCM) coding for DC coefficients and run length coding (RLC) for AC coefficients. Owing to the use of the same first AC quantizer, the resulting image has a similar bitrate compared to the scenario in which the default quantization table is used. In addition, all values in the quantization table are limited to 255. The corresponding quantization table is presented in Table 4.8. From the table, it can be seen that 1) the quantization table is symmetric, 2) it is suitable for standard zig-zag scan and RLC, and 3) many of the high frequencies seem to have little relevance for feature detection. Next, we report the results of our experiments, which show that the proposed quantization table improves feature detection performance compared to the default JPEG quantziation table.

Table 4.8: Proposed quantization table.

7	11	19	49	172	255	255	255
11	17	29	73	255	255	255	255
19	29	51	128	255	255	255	255
49	73	128	255	255	255	255	255
172	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255

4.3.4 Comparison results

The work in [DLC+12] is related to our work, so we compare the performance of their proposed quantization table. Figure 4.9 shows the performance comparison among the default quantization table, our novel quantization table, and the quantization table proposed in [DLC+12]. Figure 4.9 shows that our proposed quantization table performs the best. The performance of all the scale-space-based detectors is improved with our novel quantization table. For example, with the scale-space-based detectors, the number of correct matches is improved by 5% to 12.5% at a bitrate of 0.35 bpp. The MSER detector, however, has performance similar to that of the default quantization table, which is expected. Because the MSER detector does not comprise a Gaussian smoothing process and compression artifacts generate many spurious features, its performance can not be improved by our approach.

Our quantization table is designed for the base $\sigma_0 = 1.2$, but this value can probably be different in other detectors. For example, if a smaller σ_0 is applied in scale selection, the cut-off frequency will be higher, and many high frequencies can be helpful for improving detection. If a larger σ_0 is used, the quantizers will have a greater number of large values. Therefore, we then use $\sigma_0 = 0.8$, 1.2, and 1.6 to generate different quantization tables and compared detector repeatability scores of the generated tables. Figure 4.10 shows that the table obtained for $\sigma_0 = 0.8$ performs the worst, while the repeatability scores of $\sigma_0 = 1.2$ and 1.6 are similar for the tested scale-space-based detectors. This is because the table generated for $\sigma_0 = 0.8$ leads to an increased number of bits owing to its finer quantizers for high frequencies. By contrast, the table for $\sigma_0 = 1.6$ leads to lower bitrate. However, the σ_0 is actually not the smallest scale of the detected features, e.g., DoG and SURF detect features starting from the second layer in the scale-space. It is also found that the results are worse if we use very high values of σ_0, which lead to the elimination of a lot of image information. The detection process is complex and is devised to be robust to errors, and the standard JPEG encoding syntax targets the best possible rate-distortion performance. Many aspects affect the rate-repeatability curves; thus, the proposed quantization table is not necessarily the best for one specific detector.

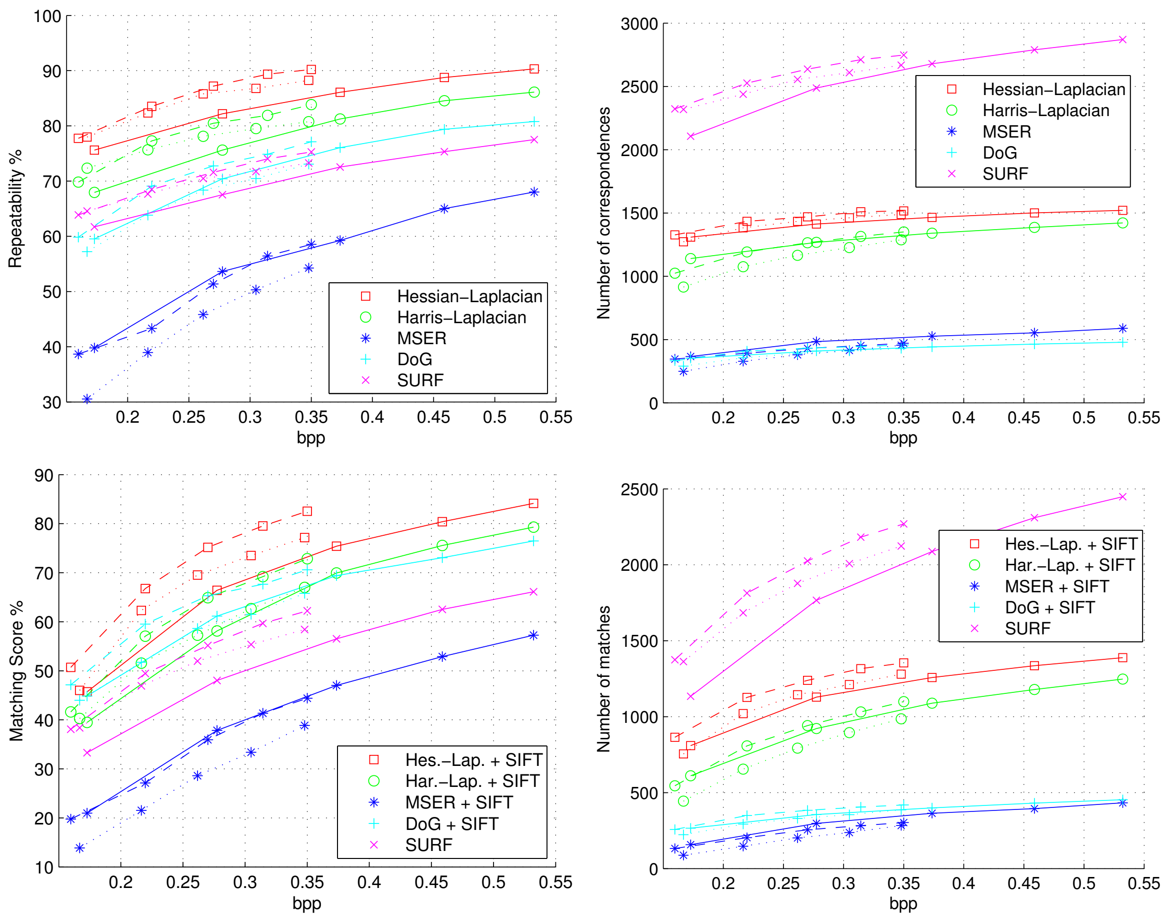

Figure 4.9: Comparison of default quantization table (solid curves), proposed quantization table (dashed curves) and quantization table of [DLC+12] (dotted curves).

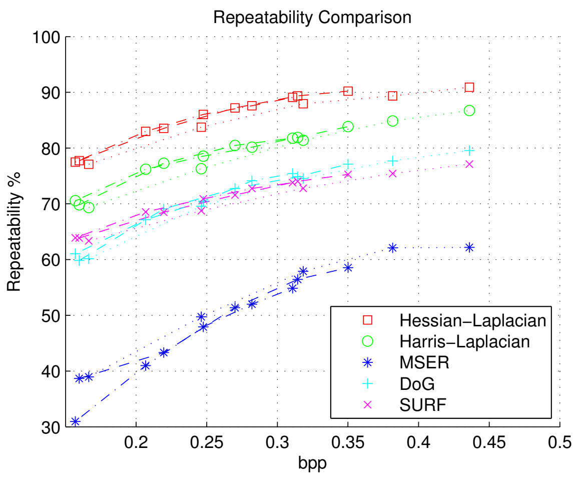

Figure 4.10: Repeatability comparison of proposed quantization tables for σ_0=0.8 (dotted curves), σ_0=1.2 (dashed curves), and σ_0=1.6 (dash-dotted curves).

4.4 Summary

In the first section of this chapter, the image-retrieval performance of different combinations of detectors and descriptors is evaluated using VLBenchmarks. It is found that the SIFT descriptor combined with the MSER detector outperform even recently proposed feature types. In a second evaluation, we quantify the effect of JPEG compression on the retrieval performance and find that the combination of Hessian-Affine and SIFT is the most robust feature under severe compression. Finally, deblocking filters are applied to the compressed query images for removal of compression artifacts, and the change in the retrieval performances of a wide range of detector-descriptor combinations is measured. It is found that although the deblocked images are improved in terms of visual quality, the retrieval performance improves only in the case of high compression and mostly when the MSER detector is used, irrespective of the descriptor, or when the MROGH descriptor is used, irrespective of the detector. The results also show that the retrieval performance typically suffers when a deblocking filter is used on high-quality JPEG queries. Moreover, only the SA-DCT and REAPP deblocking approaches can yield retrieval performance improvements. In the second section, a strategy for preserving SIFT features during JPEG image compression is proposed and validated. Specifically, two bit allocation methods are proposed. The results show that the proposed algorithms successfully preserve the strongest SIFT features, even at low bitrates. The idea underlying the proposed approaches can be applied to other compression schemes (e.g., JPEG 2000) or other features (e.g., SURF). In the third section, according to the property of scale-space-based feature detectors, a novel technique for designing a JPEG quantization table is proposed. The JPEG images with the proposed quantization table have higher repeatability scores, higher number of correspondences, higher matching scores, and higher number of matches compared to the JPEG images with the default quantization table. As expected, the MSER detector does not experience any improvement when applied to JPEG images with our novel quantization table. These proposed approaches for images, corresponding to feature-preserving image compression, improve the feature-matching performance or the retrieval performance.

Chapter 5

Feature-preserving video compression

This chapter presents our works on feature-preserving video compression. In scenarios such as video surveillance, object retrieval and tracking, and action recognition, video content is processed by computer algorithms rather than a person. Hence, the lossy video compression schemes used therein need not maximize visual quality. Similar to the effect of image compression on feature quality, the performance of typical video analysis algorithms is strongly influenced by encoding artifacts at low bitrates. In Section 5.1, to achieve reasonable feature-matching performance even for low bitrate videos, we propose allocating the bit budget during compression such that important features are preserved. Specifically, two bit allocation approaches to preserve the strongest SIFT features of H.264/AVC-encoded videos are proposed. In both approaches, we first categorize the MBs in a Group of Pictures (GOP) into several groups based on the scale property of the SIFT features presented in Section 3.2. In our first approach, a novel R-D model based on the matching score is applied to allocate the bit budget to these groups. In our second approach, to reduce computational complexity, the detector characteristics of correctly matched pairs are analyzed, and an R-D optimization method based on repeatability is proposed. Both approaches achieve better feature preservation when compared to standard video encoding, which is optimized for maximum picture quality. In Section 5.2, we consider the transmission of compressed videos via a communication network in practical video communication. A rate control strategy to preserve the most important features is required considering network capacity constraints. This section presents a novel rate control framework for H.264/AVC-based video coding. The presented strategy enhances the preservation of features such as SIFT or SURF compared to the default rate control algorithm in the JM reference software. First, a novel variant of the matching score for feature preservation based on the BoF concept is proposed. Then, the matching scores are collected as a function of the quantization parameters and analyzed for different feature types. Based on this analysis, MBs are categorized into different groups before encoding. Our rate control algorithm assigns different QPs to each group based on its importance in feature extraction. The experimental results show that the proposed rate control algorithm achieves the desired target bitrate and preserves a greater number of features compared to videos encoded using the default rate control. The proposed approach not only improves feature preservation but also leads to a noticeable performance improvement in a real image retrieval system. In this chapter, the videos used in the proposed approaches are completely standard-compatible and can be decoded by any H.264/AVC decoder. Figure 5.1 shows the architecture of video encoding and transmission. The challenges we address in this chapter are highlighted in red in the diagram.

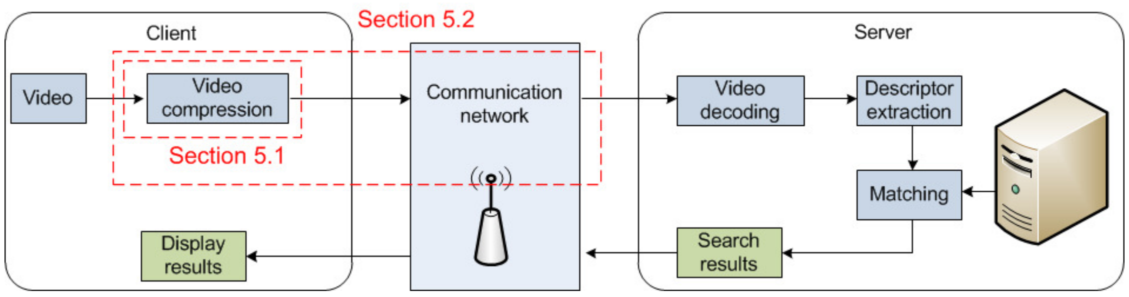

Figure 5.1: Architecture of video encoding and transmission

5.1 Novel R-D optimization for H.264/AVC-encoded videos

The H.264/AVC video compression standard [JVT03] is widely deployed in consumer electronics and multimedia communications applications. Our goal here is to preserve the SIFT features in H.264/AVC-encoded videos. Similar to the frame-level evaluation introduced in Section 2.4, a variant of this scheme is used in the following experiments. We extract the features in the original image and compare them with about 300K features extracted from a reference image database plus the features in the compressed image, examining whether the matched descriptors are in the compressed image. As introduced in the frame-level evaluation framework, it is better to reject false matches by comparing the nearest neighbor to the second nearest neighbor in the descriptor space (i.e., NNDR matching strategy). In our experiments, NNDR is set to 0.8, as proposed in [Low04], and FLANN [ML09] is used to determine the nearest neighbors in the feature database. If the NNDR is greater than 0.8, the feature preservation is assumed to have failed. This process is shown in Figure 5.2. Similarly, the "correct matches" are the corresponding matched pairs in the uncompressed and compressed images that satisfy the above-mentioned NNDR criterion. In our experiments, VLFeat [VLF] (version 0.9.13) is used for SIFT feature extraction. Video compression is performed using the H.264/AVC reference software JM [HHIb] (version 17.2).

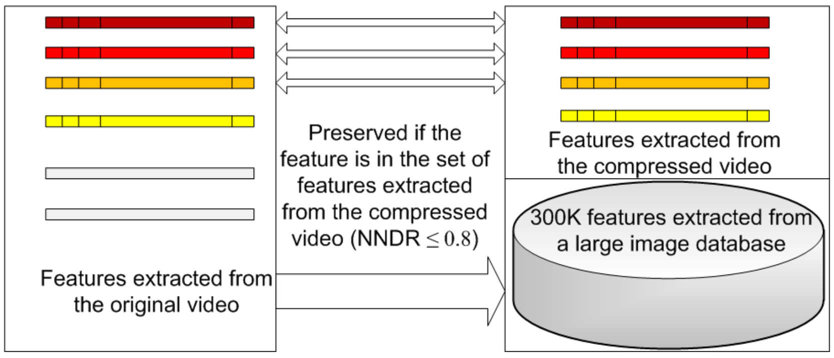
Features extracted from the original video
Preserved if the feature is in the set of features extracted from the compressed video (NNDR ≤ 0.8)
Features extracted from the compressed video
300K features extracted from a large image database

Figure 5.2: Feature preservation criterion

5.1.1 Matching score of video sequences

The matching score (ranging from 0% to 100%) is used to show the performance of different encoding approaches. It is defined as the ratio of the number of preserved (matched) features in a compressed video to the number of original features in the uncompressed video.

$$matching\ score = \frac{\sum N_p^i}{\sum N_o^i} \tag{5.1}$$

where N_p^i is the number of preserved features in the i-th compressed frame and N_o^i is the number of original features in the i-th uncompressed frame. In the following experiments, traditional image quality metrics (e.g., PSNR or SSIM) are not used; instead, the matching score in Equation (5.1) is used because we aim to optimize the number of preserved features rather than the visual quality. By preserving a greater number of features, the computer algorithms processing the compressed video frames will perform better.

5.1.2 Descriptor-based bit allocation

Our main goal is to design and validate R-D optimization strategies for SIFT feature-preserving video compression. We modify encoder control using a novel distortion measure to determine which QP to choose for each MB to preserve the most important features. The available bit budget is used such that uncompressed and compressed videos produce as many matching features as possible.

5.1.2.1 Bit allocation for blocks without relevant features

A SIFT feature covers only a local image area and the descriptor is calculated within a 4×4 array of histograms with eight orientations. The video quality of the MBs that do not contribute to the important features has no effect on the matching score. Similar to our previous work on images in Section 4.2, these MBs are compressed using the worst QP of 51 in conformance with the H.264/AVC standard.

5.1.2.2 Categorization of blocks containing features

Next, an R-D optimized QP decision is applied to the MBs containing features. The SIFT features in one frame typically extend across several MBs and overlap with each other. The traditional block-by-block R-D optimization is hence unsuitable. Different MBs have varying importance for feature preservation. In our previous experiment in Section 3.2, the sensitivity of the features detected at different scales to compression artifacts is determined. In general, there are fewer features at higher scales, while large-scale features have higher matching scores. As explained before, this is because features at lower scales occupy smaller regions, so they are more easily influenced by compression artifacts. Large-scale features cover large image areas and contain more information; thus, they have higher discriminative power, which makes it easier to match them [MTS+05]. Following the approach proposed in Section 4.2, we categorize frame MBs into several groups based on the octave values of the features and allocate higher bit budgets to the error-prone small-scale features. Our MB categorization method is as follows:

```
Step 1. Calculate SIFT features in each octave separately for all frames.
Step 2. Find the MBs containing features in the first octave and tag these MBs as group 1.
Step 3. Find the relevant MBs in the next octave. If these MBs have not been tagged before, tag them as a new group.
Step 4. Repeat step 3 until the last octave level is reached.
Step 5. Perform R-D optimization from group 1 to the last group.
```

At low bitrates, many MBs of P-frames are coded using the skip mode in H.264/AVC. In this mode, MBs are copied from the previous frame and no residual signal is transmitted. Hence, we can not tune the QP of the current MB in the current frame. Therefore, when calculating the matching score using the proposed feature-preserving R-D optimization scheme, an entire GOP is treated as a unit, i.e., one group includes MBs from across all frames in a GOP in Step 2.

5.1.2.3 Rate-Distortion Optimization

Important features from the original video and the corresponding matched features in the H.264/AVC-encoded video are identified. The matching score of a GOP is the ratio of the total number of important features to the number of matches in the compressed frames. Then, similar to the formulae in Section 4.2.3.2, the R-D optimization model is defined as follows:

$$D_{ms} = 1 - matching\ score(GOP) \tag{5.2}$$

$$J = D_{ms} + \lambda_{QP} R \tag{5.3}$$

where D_{ms} is the distortion metric used in R-D optimization for relevant MBs with SIFT features. A distortion value of 0 implies all features of one GOP are preserved after compression, while a value of 1 implies no feature is preserved. Similar to Section 4.2.3.2, it is assumed that each group is independent and the total distortion is the sum of individual distortions. Equation (5.3) is optimized simply by discretely minimizing the cost function of each group (i.e., J_i).

$$J = \sum_{i=1}^{N} J_i = \sum_{i=1}^{N} (D_i + \lambda_{QP} R_i) \tag{5.4}$$

where N is the number of groups, D_i is the distortion of the i-th group, and R_i is the rate of the i-th group. J_i is minimized separately using a common Lagrange multiplier λ_{QP}. Unlike our previous work on images in Section 4.2 or the work of Sullivan et al. [SW98], we only need to decide the QP values of each group because the QP values of each group yield different bitrates. This is similar to the case presented in the R-D optimization theory described in Section 4.2.3.1, in which only the quantizers generate different rates. The video is encoded with all QPs to determine the R-D curves and the optimal solution – the "constant slope" solution – is used. For a fixed λ_{QP}, the optimal QP of each group can be chosen on the convex hull of the R-D points. Hence, the minimization of the individual J_i can be written as follows:

$$QP_i^* = \arg\min_{QP} J_i, \;\; QP \in \{0, 1, 2, ..., 51\} \tag{5.5}$$

As Step 5 above shows, we first perform R-D optimized QP selection for group 1 because of the high vulnerability of the small-scale features in said group. The QP of the MBs in the first group is then fixed after finding the minimum J_i. Thereafter, the R-D optimization process is repeated for group 2. This continues until the last group.

5.1.3 Detector-based bit allocation

The approach described above is computationally expensive because the descriptors need to be calculated and compared with a descriptor database containing 300K features. As introduced in Section 2.4.1, the repeatability score is calculated as the metric for the detected keypoints. Lowe [Low04] presented that a feature is defined to be repeatable if the scales, locations, and orientations of a pair of features are similar within certain tolerances.

Ideally, we need to set the parameters such that the repeatable features are identical to those obtained via correctly matched descriptors according to our previous descriptor matching strategy. From [Low04], we can see that all repeatability curves are very close to the percentage of descriptors correctly matched to a large database, indicating that the repeatability criterion is closely related to the matching score. This also justifies the observation in Section 3.3 that keypoint detection is more important. Similar to [Low04], a feature in the original image is defined to be repeatable, if (1) the scale of a feature in the compressed frame is within a factor of $\sqrt{2}$ of the original scale; (2) the feature location is within σ pixels of the original feature, where σ is the scale of the feature in the original image; and (3) the orientations of two features are required to be within 15° (optional). However, the criterion is examined carefully instead of using these parameters directly in our work. If the parameters of repeatability are too loose, then a greater number of false matches would be included. By contrast, true matches would be expelled if the parameters are too strict.

5.1.3.1 Statistical observations

To determine the relationship between detector characteristics and correctly matched descriptors, we gather statistics about them. In this experiment, the first images of the *graf*, *bikes*, *cars*, *bark*, *trees*, *ubc*, and *wall* sets [MTS+05] are converted to YUV files and subsequently compressed as I frames. The SIFT parameter *peakthresh* is tuned to generate 500-700 features in the original images. Each descriptor is compared to the 300K feature database as well as the descriptors in the compressed images using a QP of 0-51. In total, there are nearly 160K correctly matched pairs. The Euclidean distances of the locations of matched pairs are calculated

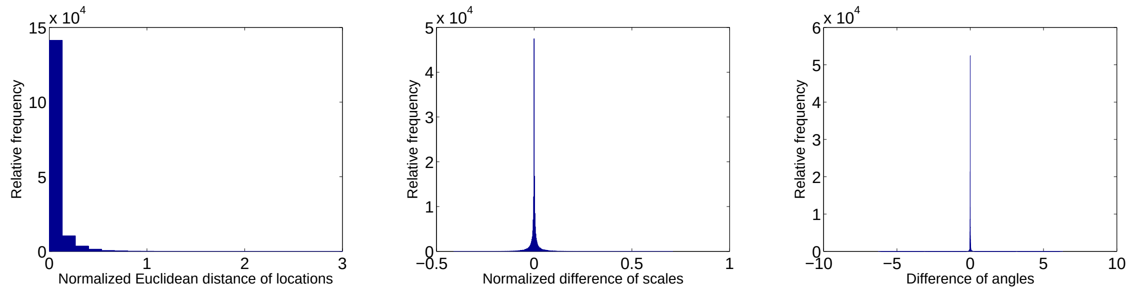

Figure 5.3: Relative frequencies of locations, scales, and angles.

and then normalized using the scales of the original features, i.e., $\sqrt{(x-xc)^2+(y-yc)^2}/s$, where (x,y) is the original location, (xc,yc) is the new location in the compressed image, and s is the feature scale. Figure 5.3 (left) shows the relationship between the normalized Euclidean distances and their distribution for the aforementioned 160K pairs. In our experiment, if $\sqrt{(x-xc)^2+(y-yc)^2}/s < 1$, the new feature is highly probable to be a correct match (the location requirement). Figure 5.3 (middle) presents the distribution of the normalized difference of scales, i.e. $|s-sc|/s$, where s is the original scale and sc is the new scale in the compressed image. The new scale should be very close to the original scale, and in our experiments, $|s-sc|/s < 0.25$ is required (the scale requirement). Figure 5.3 (right) shows the difference of angles of the correctly matched pairs. From the distribution we can see that the variance of the angle difference is smaller than the distribution of locations or scales because the descriptors are easily affected by the angle errors. There are a few possible correct matches at around 2π, e.g., if an original angle θ is 0.001 and the corresponding angle θc is 6.281, then $\theta - \theta c = -6.28$. Hence, $|\theta - \theta c| < 0.2094$ *or* $||\theta - \theta c| - 6.2832| < 0.2094$ is required (the angle requirement, 0.2094 is 12°). In summary, if a detected feature in the original image has a corresponding detected feature in the compressed image that satisfies the above three requirements, the original feature is considered repeatable. Accordingly, the repeatability in this work is defined as follows:

$$repeatability = \frac{\#repeatable\ features}{\#original\ features} \tag{5.6}$$

5.1.3.2 Rate-Distortion optimization

The R-D optimization process is similar to that in Section 5.1.2.3, except the following R-D model based on repeatability is used.

$$D_{re} = 1 - repeatability(GOP) \tag{5.7}$$

$$J = D_{re} + \lambda_{QP} R \tag{5.8}$$

where D_{re} is the distortion metric used in detector-based R-D optimization for relevant MBs with SIFT features. The distortion is 0 when the detectors in one GOP are all repeatable, and it is 1 when no detectors are repeatable after compression. In this approach, there is no need to calculate the descriptors and compare them with the 300K descriptor database. The flow diagram of the two approaches is shown in Figure 5.4.

5.1.4 Results

In our experiments, videos are encoded using the Baseline Profile with RDO enabled using high complexity mode. The GOP size is set to 12, and its structure is IPPP$\cdots$. Only the first frame is encoded as an I frame, and one previous frame is used as a reference frame for P frames. The *peakthresh* in VLFeat is set to 12 for image intensities within [0, 255] to extract the strongest features, and all other parameters are set to default values. The CIF format video *Tempete* [Uni] is used as the test video (100 frames are encoded and the frame rate is 30 fps) because it contains a specific object and the camera zooms out, producing different scales of the scene. We optimize video encoding according to the matching score-based and the repeatability-based schemes, respectively. The QPs are set to 30, 35, 40, 45, and 50 for the normally H.264/AVC-encoded video. The experimental λ_{QP} is varied as 2, 1, 0.5, 0.3, and 0.1 to generate suitable bitrates. Figure 5.5 shows the final matching scores and the repeatability values of normally H.264/AVC-encoded videos and our approaches as functions of bitrate. It can be seen that both proposed approaches improve SIFT feature preservation.

Finally, to compare the three strategies, the total number of matches for all frames in one H.264/AVC-encoded video is counted to show the number of features that can be preserved.

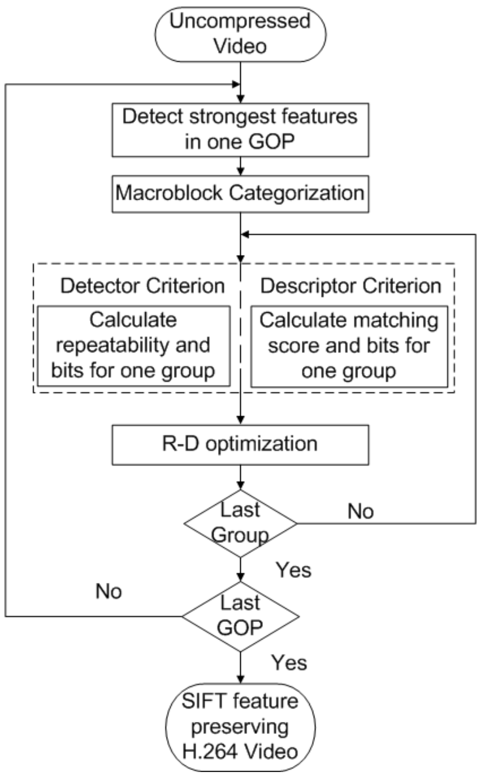
Uncompressed Video
Detect strongest features in one GOP
Macroblock Categorization
Detector Criterion
Calculate repeatability and bits for one group
Descriptor Criterion
Calculate matching score and bits for one group
R-D optimization
Last Group
No
Yes
Last GOP
No
Yes
SIFT feature preserving H.264 Video

Figure 5.4: Flow diagram of two proposed R-D optimization approaches.

Figure 5.6 shows that the two proposed approaches increase the number of matching features significantly compared to the normally H.264/AVC-encoded video. These numbers also indicate that the repeatability employed in our optimization process is reasonable because it is closely connected to the descriptor matching performance. In the videos with feature preservation, the HVS is not considered. Consequently, there are some visual inconsistencies owing to large QP changes among MBs. We will discuss the work that takes the human observer into account in the following section.

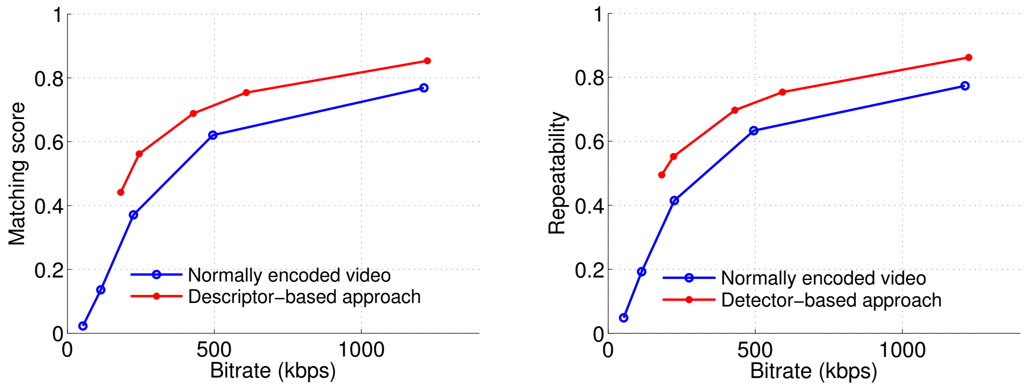

Figure 5.5: Matching scores as a function of bitrate (left). Repeatability as a function of bitrate (right).

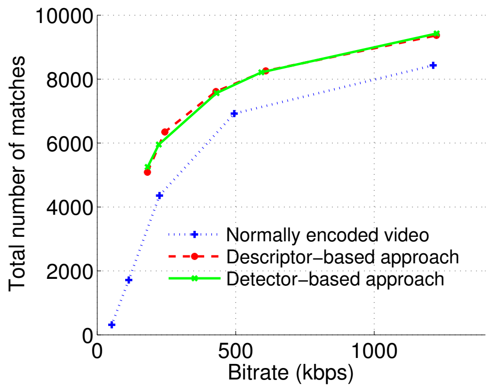

Figure 5.6: Total number of matches as a function of bitrate.

5.2 Novel rate control framework

In practical video communications, the compressed videos are typically transmitted over capacity-constrained communication channels. In order to meet the target constraints on bitrate and buffer size, rate control strategies have been intensively studied in the context of video coding standards, such as [VSW99] for the MPEG-4 standard, and JVT-G012 [LPL$^+$03] and JVT-W042 [LT07] for the H.264/AVC standard. Besides the accuracy of the achieved bitrate, many strategies have been proposed to improve the visual quality for a given target rate. For example, the authors [HLL$^+$12] proposed a region-based rate control scheme where the MBs with similar characteristics were treated as a basic unit, and an optimization model was proposed to achieve better subjective and objective quality. Ou et al. [OHC11] proposed an SSIM-based rate control algorithm, which assumed that the structural similarity (SSIM) index [WBSS04] correlates better with the perceived video quality than MSE, and achieved up to 32% bitrate reduction compared to the JM reference software [HHIb] at the same quality. In [ABCK06] and [CHC$^+$10], the authors detected regions of interest and assigned the available bits to the regions according to their importance. Zhang et al. [ZNC09] proposed a two-pass rate control scheme for high definition videos. Compared to other rate control schemes, their scheme [ZNC09] achieved approximately a constant PSNR for all frames. All the aforementioned schemes are based on the assumption that videos are eventually perceived by humans, and hence, their strategies are to improve the perceptual quality while meeting the target bitrate.

In the aforementioned scenarios (e.g. driver assistance services, surveillance, object retrieval, video search, etc.), however, the compressed videos are processed by computer vision algorithms like feature extractors rather than by humans. In this case, it is more important to preserve the information required to extract the same features than to optimize for visual quality. Therefore, the rate control algorithm should aim at preserving the strongest features at a given bitrate. In this work, a rate control algorithm is developed for SIFT and SURF feature preservation (see Figure 5.7). Other features will also benefit from our proposed rate control framework.

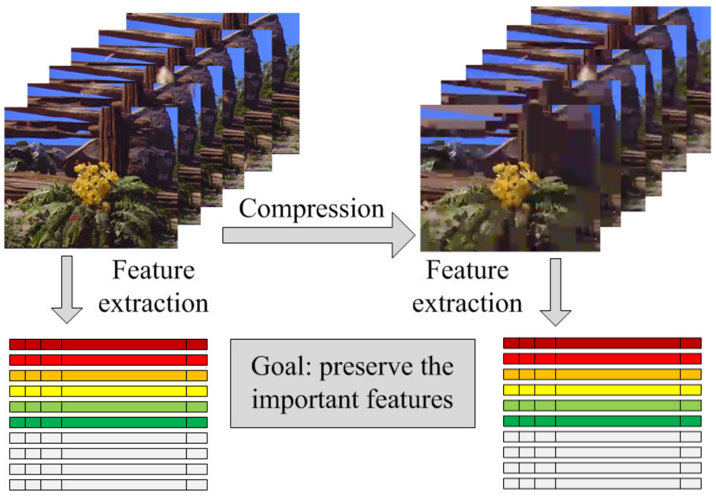

Figure 5.7: The extraction of features on an uncompressed and a compressed video. Ideally, the feature extraction algorithm should return the same set of important features (position and descriptor) after compression. Colored feature vectors are the strongest ones which are required to be preserved.

In the previous section, we propose categorizing MBs into different groups and controlling the QPs for each group such that the most important and relevant features are preserved even at low bitrates. The core idea is to calculate the rate-matching score curves and to use a Lagrangian cost function to obtain different QPs for different groups. The previously proposed approaches, however, need to quantize the video using various QPs to get the rate-matching score curves for each group. This is computationally complex. In addition, the rate control is not considered in the previous work. In this work, a novel criterion is proposed for evaluating feature preservation performance based on the BoF concept, and an improved MB categorization approach for each frame is proposed based on the feature characteristics. Then, the matching scores are collected as a function of the QPs, and a heuristic QP assignment approach for different groups is used for I frames. In addition, the work in this section presents a novel rate control algorithm, tests different numbers of features to be preserved, further considers the human observer, and conducts many experiments showing the performance of our proposed feature-preserving rate control framework.

5.2.1 BoF-based evaluation criterion

In this work, another variant of feature preservation criterion is proposed. Section 2.4.2 discusses a state-of-the-art retrieval system which uses the BoF-based approach. It quantizes the feature descriptors into so-called VWs by k-means clustering. The features are represented by their corresponding VWs, and the image is represented by a BoFs vector. With this approach, millions of images can be compared. Then, geometric verification is applied to remove false matches and improve the retrieval performance because the quantized VWs do not assure the exact same image patch pairs. There are several ways to measure the spatial consistency of correspondences in order to remove the outliers. The popular robust estimation techniques RANSAC [FB81] or the Hough transform [Low04] can be applied to the query and database images to estimate the geometric transform. However, these techniques are computationally expensive. Fast and weak geometric verification methods have also been widely studied, e.g. [SZ03], [JDS08], [CWM04].

5.2.1.1 Definition of feature preservation in this work

It is found that matched feature pairs, according to the NNDR strategy in Section 2.4.1, do not necessarily result in the same VWs. This means that these previous metrics are not strict enough to indicate whether a feature is preserved correctly after image modification. To address this issue, the BoFs approach is integrated into the definition of feature preservation. In this work, the quantizer is based on the approximate k-means (AKM) algorithm [PCI+07], which uses a flat vocabulary where descriptors are assigned to k-means centroids via randomized kd-trees. If a feature extracted from the compressed frame is quantized into the same VW as a feature in the uncompressed frame, the feature in the compressed frame is preserved with a high probability. The authors [PCI+07] found that one million VWs yielded the best performance and many follow-up studies used one million VWs in the literature. For this reason, one million VWs are trained and used to determine whether a feature is preserved. The number of trained VWs can actually be altered in specific applications. In addition, different geometric verification approaches are also proposed to improve the precision of image retrieval systems. Therefore, the feature locations should also be checked in our experiments. The detectors of SIFT and SURF features produce circular keypoint regions, therefore, only the difference of locations and the scale of the original feature need to be checked. Regarding the spatial consistency, a simpler location check is used in this work. Figure 5.8 indicates our definition of feature preservation and the relationship between the position error and the feature scale. Here, one feature in the uncompressed image is defined to be preserved correctly in our experiments if: (1) one feature in the compressed frame is quantized into the same visual word; (2) the location of this feature is within $\sigma/2$ pixels of the original location in the uncompressed frame, where σ is the scale (SIFT) or the approximate scale (SURF) of the original feature in the scale space. Also, the matching score introduced in Equation (5.1) is used in the following experiments.

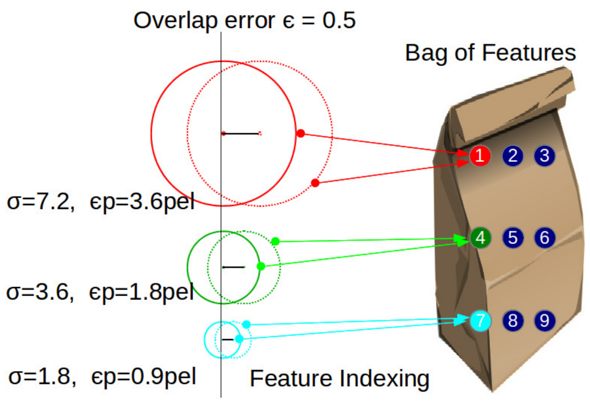

Figure 5.8: Solid circles represent features from the uncompressed image; dotted circles refer to features from the compressed image. Circles with the same color correspond to the same visual word. The overlap error in all three examples is 50%. The corresponding position error ϵ_p is a function of the scale σ.

5.2.2 Statistical feature analysis

This section first presents the dataset used to train the VWs and the video sequences to be tested in our experiments. The characteristics of the features are carefully investigated and a new MB categorization is detailed. A heuristic bit allocation approach is tested to show the performance improvement for our MB categorization scheme. In our proposed rate control algorithm, a heuristic QP assignment method is used for different MB groups in I frames, and the QPs for different groups in the subsequent P frames are calculated using a quadratic model to achieve the target bitrate.

5.2.2.1 Dataset and experimental settings

First, one million VWs are trained using the Oxford Buildings dataset [PCI$^+$07] which consists of 5062 database images. The feature extraction program is from the OpenCV library (version 2.4.3) [Ope]. The detectors and descriptors of SIFT and SURF features are examined in our experiments, and some of the settings are listed in Table 5.1.

Table 5.1: The settings for SIFT and SURF in OpenCV library

SIFT	SURF
contrastThreshold: 0.03	extended: 0
edgeThreshold: 10	hessianThreshold: 1000
nOctaveLayers: 3	nOctaveLayers: 2
nOctaves: 4	nOctaves: 4
sigma: 1.6	upright: 0

Video sequences containing rich features are used in the experiments. Eight *CIF* format video sequences [Uni] (*bus, coastguard, container, flower, foreman, mobile, tempete*, and *wa-*

terfall) and four *720p* format video sequences [TUM] (*720p50_parkrun_ter, 720p50_mobcal_ter, 720p50_shields_ter*, and *720p5994_stockholm_ter*) are used. Note that in this 720p video database the first two or four frames of the videos are grey frames and they are removed before coding. All videos are encoded using the baseline profile in the JM reference software (version 18.4) [HHIb].

5.2.2.2 Feature analysis

In many practical cases, only a small number of good features is required to perform image retrieval. In the experiments of this work, the strongest features are those which lead to the strongest detector responses during feature extraction. Figure 5.9 shows typical detector responses of SIFT and SURF features in an image. In order to find an appropriate bit allocation for the MBs, we first make observations of SIFT and SURF features.

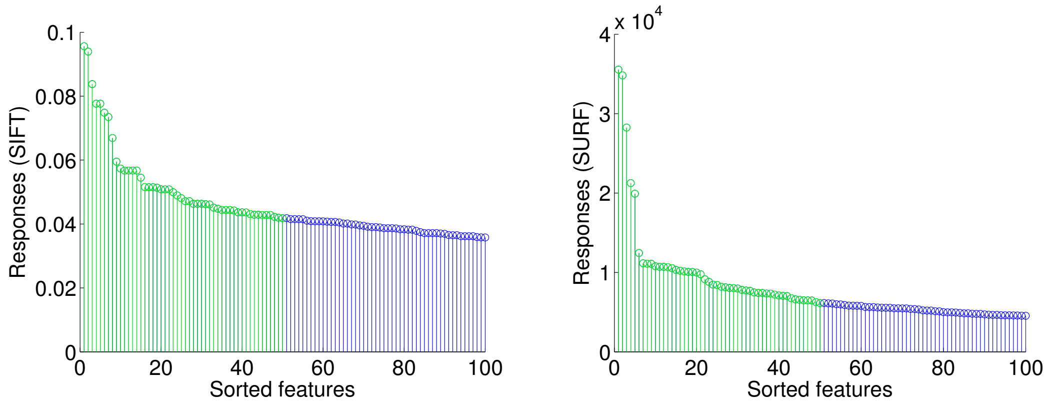

Figure 5.9: Detector responses for the top 100 SIFT and SURF features. The top 50 features are marked in green. Test image: the first frame of *bus* sequence.

5.2.2.3 Number of features to be preserved

Figure 5.10 shows the feature regions of the top 100 and top 800 SIFT features for the same image. If only a small number of features is to be preserved, some of the MBs become irrelevant for the feature extraction process, as none of the feature regions overlap with these blocks (see left plot). If many features are to be extracted (see right plot), typically the whole image is covered by the features. In this case, the degrees of freedom in bit allocation while preserving sufficient features are significantly reduced. Here, we select a certain number of features for each frame. In the following experiments, the top 1000 features in each frame are treated as the strongest features which need to be preserved for the 720p format videos, and the top 100 features in each frame for the CIF format videos.

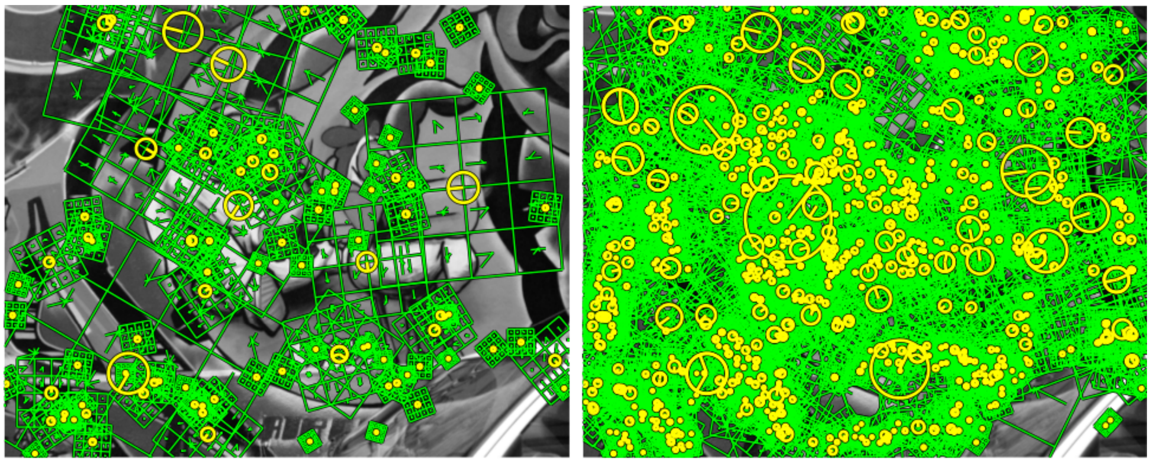
Figure 5.10: Visualization of the strongest 100 and 800 SIFT features on the same image.

5.2.2.4 Features in each octave

In SIFT feature detection, the image is filtered by a Gaussian kernel and subsampled, recursively. Keypoints are detected by searching for the extrema among their neighbors in the image as well as along the adjacent scales. SURF pursues a similar idea except that it uses various box filters and an integral image representation to compute the responses. Similar to the work in Section 3.2, the robustness of the features in each octave is tested for different video compression factors in the following.

In general, I frames in an encoded video need the highest number of bits, and the following P frames are strongly affected by the quality of the previous I frame. Therefore, I frames are critical for feature extraction. In our first experiment, for the sake of collecting more features of the high octaves, the first 50 frames of the above video sequences are encoded as I frames using the JM reference software, and the feature matching scores (Equation (5.1)) for different octaves are calculated. Figure 5.11 shows the matching scores as a function of the QP value for features in different octaves and the average number of features within each octave. From the left part of Figure 5.11, we can see that the features in the 1st octave are most vulnerable because they have the smallest regions. The larger scale features (detected at higher scales) have larger regions and hence more pixel information, thus showing higher matching scores. This observation agrees with that in Section 3.2. Second, the number of features in the first octave is dominant for both SIFT and SURF features. There are only few features in the 6th octave for SIFT, thus the matching score is not monotonically decreasing in the upper left plot. Third, at a certain level of matching score (e.g. 0.4 or 0.6), the corresponding QPs for the features in adjacent octaves have approximately $\Delta QP = 5$ and $\Delta QP = 6$ for SURF and SIFT, respectively. Similar observations are found for video sequences in CIF format as well.

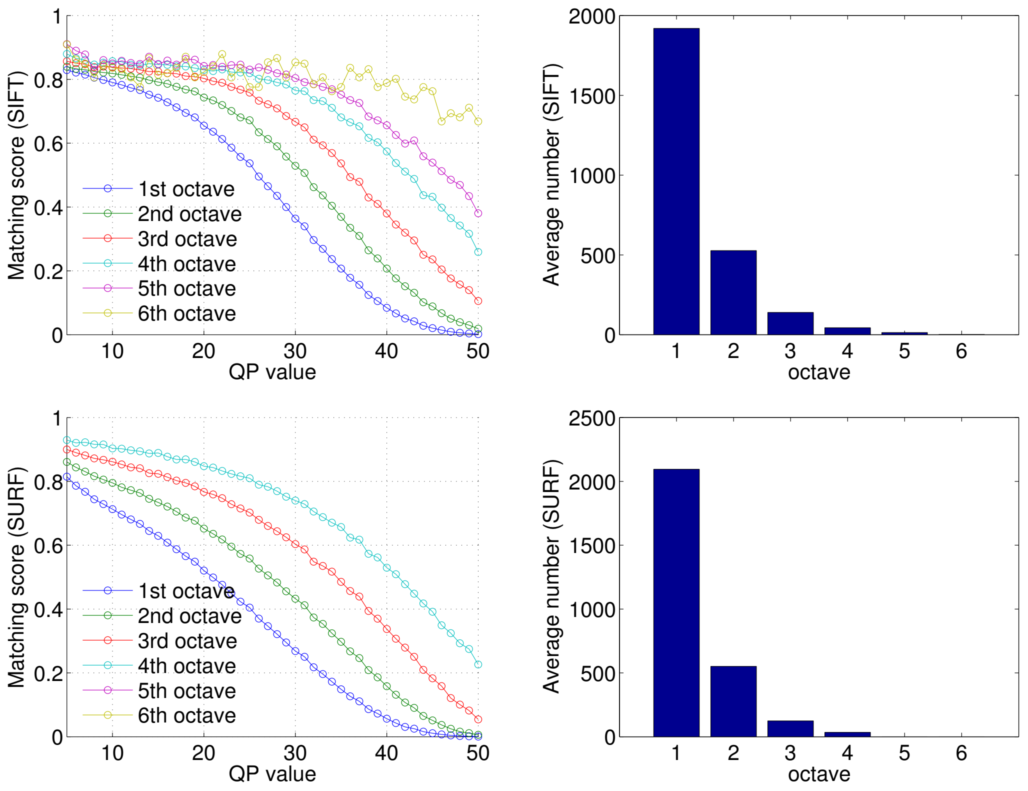

Figure 5.11: SIFT and SURF feature matching scores as a function of the QP value (left) and average number of features detected using the settings in Table 5.1 for different octaves (right) for uncompressed 720p video sequences.

5.2.2.5 Feature regions

As we introduced before, the feature extraction process consists of two steps: keypoint detection and descriptor calculation. Figure 5.10 illustrates that the detector area is much smaller than the descriptor area. The regions inside the yellow circles are the effective detector areas, while the green squares are the regions where the descriptor vectors are calculated. During the calculation of the feature descriptors, a Gaussian weighting is applied to the magnitude of the gradient value. Therefore, the pixels near to the keypoint are more important than other pixels in the calculation of feature vectors. A simple experiment is performed to justify this, and the results are shown in Figure 5.12. Note that this experiment is similar to the experiment to test the sensitivity of keypoints and descriptors for images in Section 3.3. The *prime SIFT*, *SIFT*, *prime SURF*, and *SURF* features are all extracted from the first frame of the video sequence *720p50_mobcal_ter*, which is encoded with different QPs. The difference between the normal features and the "prime" features in Figure 5.12 is that the *prime SIFT* or *prime SURF* features directly use detected keypoints from the uncompressed frame and the descriptors are calculated based on these keypoints on the compressed image. This means that the prime features have exactly the same keypoints as the ones in the uncompressed frame. Similarly, this serves to ignore the inaccurate keypoint detection from the compressed frame and to exclusively evaluate the descriptor robustness to video compression artifacts. As shown in the figure, as long as the keypoints are correctly extracted, the feature matching performance is much better, especially for SIFT features. This means that if the detected keypoints are carefully preserved, the descriptor vectors can be well preserved even for strong compression. Therefore, we can compress the keypoint region and other region in one feature using different qualities.

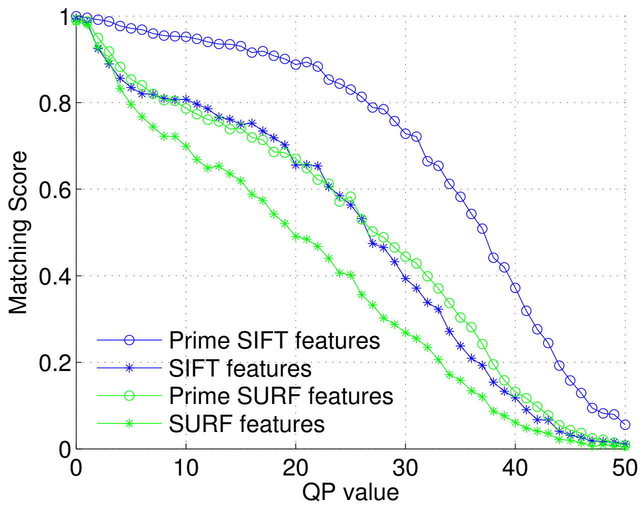

Figure 5.12: The prime SIFT or prime SURF features use the feature keypoints obtained from the uncompressed first frame in video sequence *720p50_mobcal_ter* and then compute the descriptor vectors on the compressed first frame encoded with different QP values.

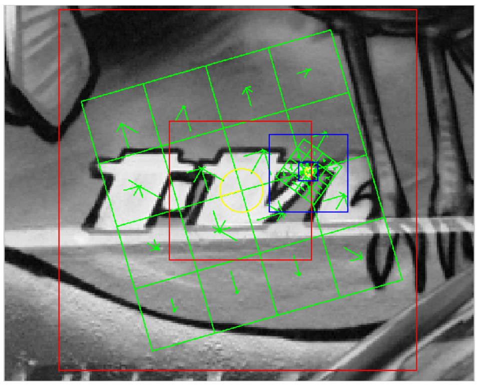

Figure 5.13: Two features (yellow circles: keypoints; green grids: regions for calculating feature vectors) for MB categorization. The two red boxes indicate the core part and the outer part of one large feature; the two blue boxes indicate the core part and the outer part of a small feature. Note that the final group index for MBs within the blue boxes is determined by the small feature, rather than the large feature.

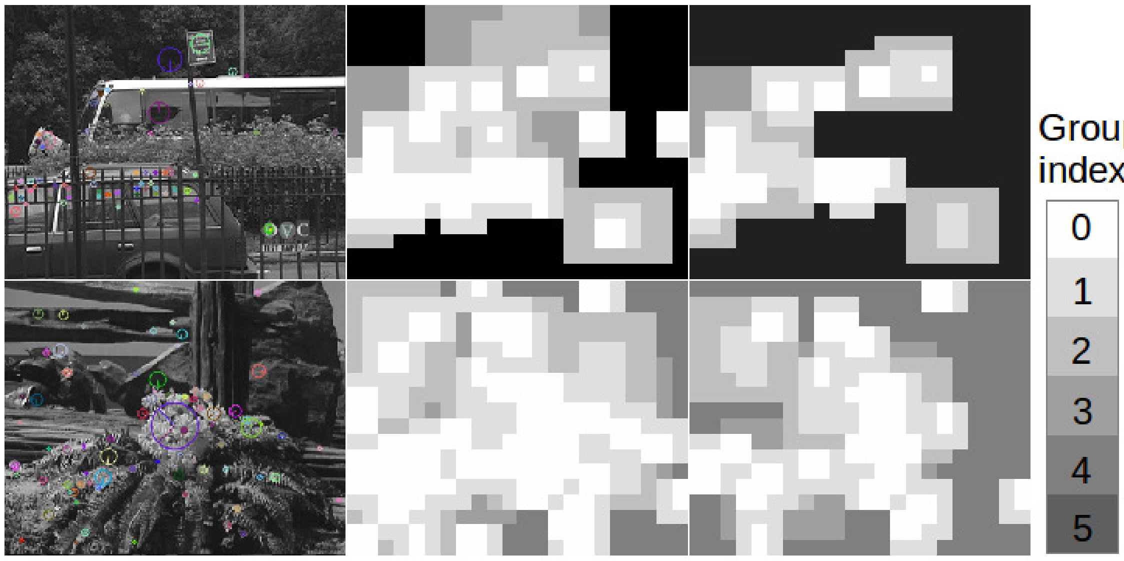

Figure 5.14: Two examples for MB categorization. Left: one frame with the top 100 keypoints. Middle: corresponding MB categorization for the top 100 features. Right: corresponding MB categorization for the top 50 features. White represents the first group, increasing gray values represent higher-indexed groups, and black represents MB's without features.

5.2.2.6 MB categorization

Based on the first observation that features in different octaves have different matching performance under compression (Section 5.2.2.4), we categorize the features within an octave into the same group. If a MB is shared by different features, the minimum octave index is selected as the final group index. However, from Figure 5.10 we can see that the whole image can be covered by a few features. Therefore, according to the second observation (Section 5.2.2.5), the region of one feature can be further divided into two different parts: the inner (core) part which is close to the keypoint center and the outer part extending until the descriptor border. In our experiments, the core part is three times the scale σ. According to the descriptor calculation process, the borders for SIFT and SURF descriptors are 7.071 times and 10.5 times the scale σ, respectively. Figure 5.13 is a magnified part of Figure 5.10 showing two features with their core and outer parts. In this categorization approach, the MBs containing the core part have the same group index i as the keypoint's octave index while the MBs containing the outer part have a group index $i + 1$. The MBs with lower indices are compressed at higher quality due to their importance for feature extraction. Unlike the prime features in Figure 5.12, the keypoints can not be detected exactly at the same pixel position as the ones detected in the uncompressed image, so the outer part is assigned only a slightly larger group index $i + 1$ to ensure that the QP difference is not too large within one feature. It should be noted that the final group index of one MB is the smallest index of all overlapping features, and the MBs which are irrelevant for features have the largest index.

Figure 5.14 shows two examples of the final MB categorization. First, the top 100 SIFT features from both frames are extracted. Then, the MB categories for the top 100 and top 50 features are compared. The upper row in Figure 5.14 illustrates that for this test image, many MBs can be compressed strongly because they have no relevance for feature extraction. By contrast, the lower row shows that for the second test image all MBs are relevant. Second, from the middle and right columns in Figure 5.14, we can see that the number of features to be preserved also significantly affects the MB categorization. As we discussed in Section 5.2.2.3, if a large number of features needs to be preserved, all MBs can be covered by small-indexed groups reducing the degrees of freedom in bit allocation. We will show the effect of the number of required features to be preserved in Section 5.2.4.2.

In summary, the first step of our approach is to extract features from each frame and categorize the MBs into different groups. This can be done in a pre-processing step and is less time-consuming compared to the encoding process in our experiments.

5.2.2.7 QP assignment for different MB groups

In Section 5.2.2.4, the QPs required for each octave at a certain matching score level are shown. For the sake of simplicity, we assume in the following explanation that each group ideally corresponds to each individual octave. Then, in order to achieve a certain matching score, the MBs in the first group are assigned an initial QP value. The subsequent QPs in further groups can be obtained from the QP of the first group in order to achieve the same matching score by using larger QPs. In the default JM reference encoding process, a fixed QP_0 is assigned to all MBs. By contrast, in our bit allocation process, the QP for each group is assigned as $QP_0 + g * 5$ for SIFT and $QP_0 + g * 6$ for SURF, respectively, where g is the group index for the MB (g starts from 0 in our experiments). The MBs without features have QP value 51.

Now that we have designed the MB categorization and the QP assignment for each group, experiments are performed to evaluate the effectiveness of our approach. The first 150 frames of the video *720p50_mobcal_ter* are encoded, and the top 1000 features of each frame are aimed to be preserved. The frame rate is 50 fps, the GOP size is 50, and its structure is IPPP$\cdots$. Only the first frame is encoded as an I frame. First we use fixed QPs from 22 to 36 to encode the videos, and the matching scores are calculated for the encoded videos. In our approach, QP values from 18 to 34 are selected for the first group and larger QP values are selected for the next groups as described before. All other settings are set to default. Figure 5.15 shows the obtained improvement for both SIFT and SURF features. As can be seen from the figure, assigning different QP values for different groups is a feasible way to improve the rate-matching score performance.

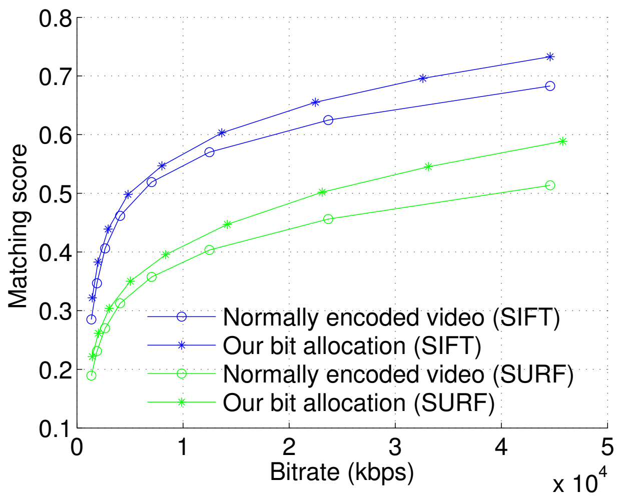

Figure 5.15: The matching scores of SIFT and SURF features for the default H.264/AVC-encoded videos and our QP assignment approach. The top 1000 features of each frame for the video *720p50_mobcal_ter* are targeted to be preserved.

5.2.3 Rate control

With an appropriate rate control scheme, the videos are encoded to meet the target rate. This section presents our proposed rate control approach that preserves more features compared to the default rate control scheme. Our scheme is conceptually based on [LSW05, LGP$^+$06], and the implementation is based on the H.264 JM reference software [HHIb]. It also complies with the hypothetical reference decoder (HRD). Only GOP level rate control and picture level rate control are needed. Because the process is similar to [LSW05, LGP$^+$06], only the important modifications will be detailed in the following, especially highlighting the differences.

5.2.3.1 MAD and QP models

The JM reference software adopts the rate control algorithm proposed in [LGP$^+$06]. The work uses a linear MAD prediction model to solve the chicken-and-egg dilemma of the QP decision in H.264/AVC. In our work, the linear model is adapted to each MB group. Therefore, the linear model is modified as:

$$\tilde{\sigma}_g(j) = a_{1,g} \times \sigma_g(j-1) + a_{2,g} \tag{5.9}$$

where $\tilde{\sigma}_g(j)$ is the predicted MAD of MB group g in the current frame j and $\sigma_g(j-1)$ is the actual MAD of the MB group g in the previous frame $j-1$. $a_{1,g}$ and $a_{2,g}$ are two coefficients for group g, and the MAD is predicted independently for each group in our rate control approach. Similar to [LGP$^+$06], the values of $a_{1,g}$ and $a_{2,g}$ for each group are initially set to 1 and 0, respectively. After encoding each frame, the values are updated accordingly. While the research presented in [LGP$^+$06] includes the concept of basic unit and the formulae looked similar to our modified one, the basic unit in [LGP+06] is fixed across frames and should have the same number of MBs within one frame. In our model, the concept of groups does not correspond to the concept of basic unit. The shape of the groups is not fixed because the locations and scales of detected features change across frames, with the result that the number of MBs is different for each group. As different QP values are used for different groups, we make the assumption that the MAD can be predicted by the previous encoded MAD of the same group. It should be noted that in the original JM reference software, the predicted MAD and actual MAD are normalized by the number of MBs in a frame. In our rate control process, $\tilde{\sigma}_g(j)$ and $\sigma_g(j-1)$ are the MADs normalized by the number of MBs in group g:

$$\tilde{\sigma}_g(j) = \frac{\sum_{m=0}^{N_g(j)} MAD_{g,m}(j)}{N_g(j)} \tag{5.10}$$

where $N_g(j)$ is the number of MB's in group g in frame j and $MAD_{g,m}(j)$ is the MAD of the m^{th} MB in group g.

The quadratic model to calculate the QP is also modified as follows:

$$T_g(j) = c_{1,g} \times \frac{\tilde{\sigma}_g(j)}{Q_{step,g}(j)} + c_{2,g} \times \frac{\tilde{\sigma}_g(j)}{Q^2_{step,g}(j)} \tag{5.11}$$

where $T_g(j)$ is the average number of texture bits for group g in the current frame j. $c_{1,g}$ and $c_{2,g}$ are the two coefficients of the quadratic model for group g, and $Q_{step,g}$ is the quantization step for group g. In this way, we can calculate an individual QP for each group.

5.2.3.2 GOP level rate control

The bit allocation at the GOP level and for each frame is similar to the one in [LSW05] and the JM reference software. Again, the main modifications and differences are highlighted here. The GOP level rate control takes the number of remaining bits, the occupancy of the virtual buffer, and the channel conditions into account. In our work, we assume a constant bitrate channel, and B frames are not used.

Unlike the default rate control scheme in the JM reference software, each group has a different initial quantization parameter. In our approach, the QP of the first group (group index g starts from 0) for the IDR frame and the first P frame ($QP_{1,0}(1)$) is set as follows:

$$QP_{1,0}(1) = \begin{cases} 32 & bpp \leq l1 \\ 22 & l1 < bpp \leq l2 \\ 18 & l2 < bpp \leq l3 \\ 8 & bpp > l3 \end{cases} \tag{5.12}$$

where $bpp = \frac{Bit\ rate}{Frame\ rate \times N_{pixel}}$. In $QP_{1,0}(1)$, the first subscript refers to the GOP index, the second subscript refers to group index, and the last number 1 means the I frame in each GOP.

$l1$, $l2$, $l3$ are calculated according to the width of the video (see JM reference software). The reason for tuning the QPs is that MBs belonging to other groups can be encoded with a larger QP. In order to get a similar number of bits as the default rate control scheme yields, the QP for the first group is adjusted slightly smaller. Then the QPs for the next groups are assigned based on the QP for the first group:

$$QP_{1,g}(1) = QP_{1,0}(1) + g * \Delta QP \tag{5.13}$$

where according to our previous observation, $\Delta QP = 5$ and $\Delta QP = 6$ for SURF and SIFT, respectively.

From the second GOP on, the initialization parameter $QP_{i,0}(1)$ for the first group ($g = 0$) is calculated based on the previous QPs for this group:

$$\begin{aligned} QP_{i,0}(1) = max\Big\{&QP_{i-1,0}(1) - 2, min\Big\{QP_{i-1,0}(1) + 2, \\ &\frac{SumPQP(i-1,0)}{N_p(i-1)} - min\Big\{2, \frac{N_{i-1}}{15}\Big\}\Big\}\Big\} \end{aligned} \tag{5.14}$$

where $SumPQP(i-1,0)$ is the sum of the average frame quantization parameters in group 0 ($g = 0$) for all P frames in the $(i-1)^{th}$ GOP. $N_p(i-1)$ is the total number of P frames and N_{i-1} is the total number of frames in the $(i-1)^{th}$ GOP. Similarly, the QPs for the next groups in the I frame in the i^{th} GOP are assigned based on the QP for the first group:

$$QP_{i,g}(1) = QP_{i,0}(1) + g * \Delta QP \tag{5.15}$$

All other settings of our approach are the same as in the JM reference software.

5.2.3.3 Frame level rate control

The target number of bits allocated to the current frame is determined based on the target buffer level, the frame rate, the available channel rate, the actual buffer occupancy, the remaining bit budget, and boundary bits which must conform with the HRD requirements. The details are described in [LSW05] and the JM reference software [HHIb].

Once the target bitrate $T(j)$ for the current frame j is available, the bit allocation problem for each group needs to be addressed. The number of bits assigned for each group should be related to the different importance for feature extraction. However, we can not simply assign different QPs to each group for a P frame to achieve the target number of bits. We assume that each group in the P frames has similar properties as the corresponding group in the I frame. Thus, each group needs a similar proportion of bits in the P frames as the group in the I frame. Here, we propose a weighting scheme based on the information from the I frame in each GOP. The importance weight for group g in the current GOP is calculated as follows:

$$W_g = \frac{normH_g^I + normT_g^I}{\sum\limits_{g=0}^{N_g}\Big(normH_g^I + normT_g^I\Big)} \tag{5.16}$$

where $normH_g^I$ and $normT_g^I$ represent the normalized header bits and the average texture bits for group g in the I frame. In the following P frames of the current GOP, these weights are used when assigning bits for each group:

$$\tilde{T}_g(j) = T(j) * \frac{W_g * N_g(j)}{\sum\limits_{g=0}^{N_g(j)} \Big(W_g * N_g(j) \Big)} \tag{5.17}$$

Therefore, the $T_g(j)$ in the quadratic model of Equation (5.11) is calculated as:

$$T_g(j) = \frac{\tilde{T}_g(j)}{N_g(j)} - normH_g(j-1) \tag{5.18}$$

where the $normH_g(j-1)$ is the normalized header bits of the previous P frame. Note that $\tilde{T}_g(j)$ is normalized by $N_g(j)$ in the above equation, since the $\tilde{\sigma}_g(j)$ is also normalized in Equation (5.11). Similar to the JM software, the target number of bits for each group is then bounded by

$$\begin{aligned} T_g(j) &= max \left\{ T_g(j), \frac{R}{4f} \times \frac{W_g * N_g(j)}{\sum\limits_{g=0}^{N_g(j)} \Big(W_g * N_g(j) \Big)} \Big/ N_g(j) \right\} \\ &= max \left\{ T_g(j), \frac{R}{4f} \times \frac{W_g}{\sum\limits_{g=0}^{N_g(j)} \Big(W_g * N_g(j) \Big)} \right\} \end{aligned} \tag{5.19}$$

where R is the bitrate and f is the frame rate. Thereafter, the corresponding quantization parameter $QP_g(j)$ for group g can be calculated correctly by using the target bit $T_g(j)$ and the coefficients in Equation (5.11). To address the inherent inaccuracy of the models and to ensure good video quality, the following constraints are applied:

a) Constraint from the QP of the previous frame

The quality of the quantized picture should not change abruptly, so the QP for each group is constrained by the QP of the previous frame:

$$QP_{i,g}(j) = max\Big\{ QP_{i,g}(j-1) - 4, min\Big\{ QP_{i,g}(j-1) + 4, QP_{i,g}(j) \Big\} \Big\} \tag{5.20}$$

b) Constraint for each group within the current frame

The MAD prediction model and the quadratic model are inevitably not accurate, especially when the number of MBs in higher groups is quite small. Generally, the first group has the largest number of MBs, and its QP can be calculated more accurately using the proposed models. Thus, the predicted QP for the current group is further bounded by the previous group from the second group on. The minimum and maximum allowed QP differences between the current and previous groups are 3 and 8, which are heuristically set according to the ΔQP for adjacent groups:

$$QP_{i,g}(j) = max\Big\{QP_{i,g-1}(j) + 8, min\Big\{QP_{i,g-1}(j) + 3, QP_{i,g}(j)\Big\}\Big\},\ g > 0 \tag{5.21}$$

c) Constraint due to encoded bits error from two previous frames

The rate control approach is not part of the standard, and the default rate control algorithm implemented in the JM reference software is not optimal. It is found that while the default rate control algorithm can achieve the target bitrate for most videos, the remaining bits are often exhausted halfway in one GOP for some of the videos. As a result, the final bitrate is not accurate, and the video quality is significantly affected. Because successive P frames have similar properties, their target number of bits should be close. If the calculated QP for the previous frame is inappropriate, the actual encoded number of bits will be quite different from the target number of bits. We therefore use a novel scheme to adjust the estimated QP before encoding the current frame:

$$QP_{i,g}(j) = \begin{cases} max\Big\{QP_{i,g}(j-1) + 1, QP_{i,g}(j)\Big\},\ if\ b(j-1) > 1.2 * T(j-1) \\ \hfill AND\ b(j-2) > 1.2 * T(j-2) \\ min\Big\{QP_{i,g}(j-1) - 1, QP_{i,g}(j)\Big\},\ if\ b(j-1) < 0.8 * T(j-1) \\ \hfill AND\ b(j-2) < 0.8 * T(j-2) \end{cases} \tag{5.22}$$

where $b(j-1)$ and $b(j-2)$ are the actually generated bits; $T(j-1)$ and $T(j-2)$ are the target bits for the $(j-1)^{th}$ and $(j-2)^{th}$ frames. It is found that using the information of two previous encoded frames and the empirically selected values (0.8, 1.2) are appropriate to adjust the current QP estimation. The conditions for the over-use and under-use of target bits are both considered in the above equation, although in most cases the models for high bitrate estimation are accurate enough.

Afterwards, the MBs are encoded by the final $QP_{i,g}$ calculated for each group g, and the virtual buffer is updated according to the real number of encoded bits and the channel rate. The linear MAD prediction model and the quadratic R-D model are also updated for each group.

It should be noted that in the above process, we calculate $T_g(j)$ and the corresponding $QP_{i,g}(j)$ for each group. However, we can simplify this process. When the $QP_{i,0}(j)$ of first group in the P frames is computed, the $QP_{i,g}(j)$ of other groups can be assigned directly based on $QP_{i,0}(j)$ as follows:

$$QP_{i,g}(j) = QP_{i,0}(j) + g * \Delta QP \tag{5.23}$$

This is similar to Equation (5.13) and (5.15), which are done for I frames. In this case, Equations (5.17), (5.18), and (5.19) will be computed only for the first group, and the constraint in Equation (5.21) will not be necessary. This simplification does not take the number of bits for other groups into account, however, it would be still feasible to achieve the target bitrate because higher groups in P frames normally consume smaller number of bits. In the following, we will only show the results of the previous approach.

5.2.4 Pairwise matching results

In order to evaluate the performance of feature preservation for different approaches, extensive experiments are performed on the test video sequences. First, the videos are encoded using various fixed QP values (i.e. without rate control). Then the target bitrates for the default rate control in the JM reference software and our proposed rate control are set according to these fixed QP encoded videos. Afterwards, the matching scores of the fixed QP approach, the default rate control approach, and our proposed approach are compared.

In further experiments, we alter the number of features to be preserved. The results show that our rate control approach always achieves better matching scores compared to the default rate control approach.

5.2.4.1 Main results

Based on the JM reference software, some of the important experimental settings are listed in the following:

1) Baseline profile, LevelIDC is 40 for CIF format videos and 50 for 720p format videos

2) Fixed QP values of 24, 27, 30, 33, and 36

3) Frame rate 30 fps and 50 fps for the CIF and 720p format videos, respectively

4) First 150 frames and 250 frames for the CIF and 720p format videos are encoded, respectively

5) GOP size is 50 frames and only the first frame is encoded as I frame

6) The number of reference frames is 5

7) RDO is enabled

All other settings use the default values. Our goal is to preserve 100 SIFT/SURF features for the CIF format videos and 1000 SIFT/SURF features for the 720p format videos. The matching scores are calculated and compared for videos encoded by the fixed QP approach, the default rate control approach, and our proposed rate control approach. Table 5.2 and Table 5.3 show the results for the CIF format and the 720p format videos. From the tables, we can see that our rate control approach achieves similar target bitrates as the default rate control approach. In addition, our approach achieves better feature preservation performance compared to the default approach, which yields even worse matching scores compared to the fixed QP approach.

Table 5.2: Bitrate and matching score comparison for CIF format videos

Features			SIFT features				SURF features			
Video	Bit rate (kbps)		Bit rate	Matching score (%)			Bit rate	Matching score (%)		
sequences (CIF)	Fixed QP (Target)	Default (Rate control)	Ours	Fixed QP	Default	Ours	Ours	Fixed QP	Default	Ours
Bus	2317.39	2317.20	2316.51	68.53	68.61(+0.07)	**72.34**(+3.80)	2316.23	55.66	55.38(-0.28)	**60.46**(+4.80)
	1557.35	1557.82	1556.66	61.61	61.04(-0.56)	**65.87**(+4.26)	1555.98	48.63	48.37(-0.27)	**52.54**(+3.91)
	998.82	999.89	996.57	52.90	52.88(-0.02)	**56.96**(+4.06)	998.40	40.84	40.65(-0.19)	**44.05**(+3.21)
	635.86	635.91	635.80	44.32	43.40(-0.93)	**47.05**(+2.73)	634.56	33.21	33.04(-0.17)	**35.77**(+2.56)
	394.95	395.40	394.23	34.32	34.15(-0.17)	**37.52**(+3.20)	394.33	25.72	25.51(-0.21)	**27.52**(+1.80)
Coastguard	2531.17	2531.04	2530.46	61.70	61.95(+0.25)	**69.55**(+7.85)	2530.89	49.80	49.65(-0.15)	**61.57**(+11.77)
	1696.92	1696.91	1696.05	53.67	53.82(+0.15)	**63.14**(+9.46)	1696.16	42.21	42.91(+0.70)	**54.61**(+12.39)
	1034.77	1034.27	1034.24	43.78	45.08(+1.30)	**54.42**(+10.64)	1035.58	34.00	34.93(+0.93)	**45.00**(+11.00)
	586.83	588.26	586.16	33.98	34.72(+0.73)	**43.14**(+9.15)	587.13	26.03	26.53(+0.49)	**35.69**(+9.66)
	298.29	300.01	297.89	24.86	25.03(+0.17)	**30.51**(+5.64)	298.17	18.07	19.16(+1.09)	**26.52**(+8.45)
Container	482.31	482.34	481.33	66.15	62.13(-4.02)	**66.97**(+0.82)	481.96	53.22	51.92(-1.30)	**58.61**(+5.39)
	263.76	263.77	263.78	58.29	55.79(-2.50)	**59.68**(+1.39)	264.27	46.21	44.91(-1.29)	**49.77**(+3.57)
	145.33	145.69	145.78	49.94	47.67(-2.26)	**51.89**(+1.96)	145.41	41.05	39.42(-1.63)	**42.47**(+1.42)
	87.64	88.38	87.86	40.69	40.55(-0.13)	**42.94**(+2.26)	87.87	31.85	32.27(+0.42)	**35.55**(+3.70)
	56.63	57.48	56.95	31.98	33.37(+1.39)	**37.63**(+5.64)	56.98	24.22	24.11(-0.11)	**30.45**(+6.23)
Flower	2919.45	2920.87	2918.62	68.19	68.53(+0.35)	**71.49**(+3.30)	2921.76	45.54	45.79(+0.25)	**49.35**(+3.81)
	2054.79	2057.94	2055.56	61.71	61.04(-0.67)	**64.28**(+2.57)	2057.03	38.91	38.23(-0.68)	**40.73**(+1.82)
	1357.47	1354.74	1357.18	52.52	52.56(+0.04)	**55.77**(+3.25)	1359.72	31.29	31.23(-0.06)	**33.14**(+1.85)
	846.95	845.55	846.49	42.79	42.99(+0.20)	**45.08**(+2.29)	846.89	24.09	25.03(+0.94)	**26.35**(+2.26)
	497.05	496.25	495.83	33.08	33.34(+0.26)	**35.60**(+2.52)	495.51	18.73	18.84(+0.11)	**19.65**(+0.93)
Foreman	790.31	789.62	788.23	59.44	59.93(+0.48)	**60.91**(+1.47)	789.73	50.57	50.29(-0.29)	**52.99**(+2.41)
	493.71	492.43	493.70	52.54	51.97(-0.57)	**53.22**(+0.68)	493.92	42.24	41.43(-0.81)	**44.01**(+1.77)
	315.87	316.19	315.48	43.50	43.08(-0.42)	**45.04**(+1.55)	315.55	34.47	33.73(-0.75)	**36.39**(+1.92)
	212.87	215.96	213.29	35.05	36.18(+1.13)	**36.93**(+1.88)	213.00	26.72	26.67(-0.05)	**28.30**(+1.58)
	145.66	150.19	146.07	26.91	**28.62**(+1.72)	28.06(+1.15)	145.45	18.83	19.75(+0.91)	**20.31**(+1.47)
Mobile	3615.86	3615.76	3610.24	74.30	74.33(+0.04)	**79.21**(+4.92)	3610.13	58.05	57.80(-0.25)	**63.14**(+5.09)
	2438.22	2439.83	2434.91	68.25	68.53(+0.28)	**73.82**(+5.57)	2437.18	51.27	51.10(-0.17)	**56.73**(+5.46)
	1489.76	1490.00	1486.60	60.34	60.00(-0.35)	**66.06**(+5.71)	1490.31	43.20	43.71(+0.51)	**48.19**(+4.99)
	835.99	834.78	835.49	51.46	50.96(-0.50)	**56.62**(+5.16)	834.75	36.63	36.75(+0.12)	**40.12**(+3.49)
	460.51	460.98	459.76	41.71	43.63(+1.92)	**46.77**(+5.06)	459.71	30.20	30.73(+0.53)	**33.30**(+3.10)
Tempete	2469.80	2470.97	2468.77	67.89	67.74(-0.16)	**71.66**(+3.77)	2469.61	50.05	50.42(+0.37)	**53.65**(+3.60)
	1586.59	1587.62	1585.37	60.37	60.87(+0.50)	**64.68**(+4.31)	1586.42	42.56	43.13(+0.57)	**46.57**(+4.01)
	929.35	929.97	928.21	51.46	51.32(-0.14)	**55.49**(+4.04)	930.69	35.21	35.77(+0.56)	**38.34**(+3.13)
	518.86	520.24	519.24	42.16	41.21(-0.95)	**45.85**(+3.68)	519.09	27.87	28.54(+0.67)	**31.07**(+3.20)
	294.26	295.05	294.37	32.36	33.96(+1.60)	**35.87**(+3.51)	294.82	21.11	22.18(+1.07)	**24.27**(+3.16)
Waterfall	766.64	767.71	765.56	55.84	55.43(-0.41)	**58.41**(+2.57)	765.11	39.06	39.43(+0.37)	**41.57**(+2.51)
	436.75	437.17	435.96	47.61	46.92(-0.69)	**48.82**(+1.21)	436.66	31.78	31.23(-0.55)	**34.46**(+2.68)
	261.26	262.21	261.28	38.38	37.74(-0.64)	**41.04**(+2.66)	262.28	25.26	25.01(-0.25)	**28.00**(+2.74)
	157.86	160.05	157.88	29.12	28.02(-1.10)	**32.22**(+3.10)	158.67	18.63	18.78(+0.15)	**22.03**(+3.41)
	98.14	99.29	98.45	20.65	20.07(-0.58)	**22.45**(+1.80)	98.70	12.08	12.93(+0.85)	**15.08**(+3.00)

Table 5.3: Bitrate and matching score comparison for 720p format videos

Features			SIFT features				SURF features			
Video	Bit rate (mbps)		Bit rate	Matching score (%)			Bit rate	Matching score (%)		
sequences (720p)	Fixed QP (Target)	Default (Rate control)	Ours	Fixed QP	Default	Ours	Ours	Fixed QP	Default	Ours
Mobcal	23.68	23.70	23.67	62.48	61.68(-0.80)	**65.47**(+2.98)	23.67	45.63	45.09(-0.53)	**49.01**(+3.38)
	9.29	9.29	9.28	54.50	53.73(-0.77)	**56.11**(+1.61)	9.28	38.10	37.88(-0.23)	**40.17**(+2.06)
	4.08	4.08	4.08	46.17	46.17(+0.00)	**48.31**(+2.14)	4.09	31.26	31.40(+0.14)	**33.14**(+1.88)
	2.21	2.23	2.21	37.83	38.24(+0.41)	**41.57**(+3.74)	2.21	25.22	25.12(-0.11)	**27.98**(+2.76)
	1.36	1.38	1.37	28.50	28.61(+0.12)	**33.91**(+5.42)	1.36	18.94	18.16(-0.78)	**22.07**(+3.13)
Parkrun	71.39	71.40	71.34	71.02	71.13(+0.11)	**77.16**(+6.14)	71.35	51.00	50.88(-0.13)	**57.35**(+6.34)
	43.94	43.93	43.92	64.06	63.66(-0.40)	**69.05**(+4.98)	43.93	43.66	43.46(-0.21)	**48.04**(+4.38)
	26.08	26.09	26.06	55.04	54.90(-0.15)	**59.08**(+4.04)	26.09	35.53	35.60(+0.07)	**38.67**(+3.14)
	16.02	16.02	16.00	45.71	46.19(+0.48)	**49.63**(+3.91)	16.01	28.25	28.58(+0.33)	**30.42**(+2.17)
	9.62	9.62	9.61	35.71	36.79(+1.08)	**39.42**(+3.71)	9.62	20.98	21.90(+0.92)	**23.18**(+2.20)
Shields	22.53	22.54	22.51	65.28	64.43(-0.85)	**69.29**(+4.01)	22.55	50.98	50.52(-0.46)	**54.79**(+3.81)
	8.28	8.29	8.29	55.93	55.32(-0.61)	**58.81**(+2.88)	8.30	42.53	42.10(-0.43)	**45.08**(+2.55)
	3.92	3.93	3.92	46.94	47.04(+0.10)	**50.60**(+3.67)	3.92	35.17	35.28(+0.11)	**38.16**(+2.99)
	2.18	2.20	2.19	38.53	39.79(+1.27)	**43.10**(+4.57)	2.19	28.64	29.67(+1.03)	**32.45**(+3.81)
	1.31	1.34	1.31	29.11	29.88(+0.77)	**34.03**(+4.91)	1.31	21.43	22.03(+0.61)	**24.98**(+3.55)
Stockholm	29.49	29.51	29.48	64.61	64.51(-0.10)	**70.14**(+5.53)	29.47	49.08	48.82(-0.25)	**54.72**(+5.65)
	8.16	8.16	8.17	55.12	54.52(-0.60)	**57.53**(+2.41)	8.16	39.62	39.18(-0.44)	**42.22**(+2.60)
	2.65	2.66	2.66	45.86	46.12(+0.26)	**48.19**(+2.33)	2.65	31.70	31.66(-0.04)	**33.94**(+2.24)
	1.42	1.44	1.43	37.55	37.53(-0.02)	**41.13**(+3.57)	1.43	24.79	24.82(+0.03)	**28.31**(+3.52)
	0.88	0.93	0.88	28.20	27.87(-0.33)	**32.77**(+4.57)	0.88	17.59	17.82(+0.24)	**21.86**(+4.27)

5.2.4.2 The number of features to be preserved

Previously we only showed the results of preserving 100 features for the CIF format videos and 1000 features for the 720p format videos. This section presents the results of other target numbers, showing that our algorithm always has a positive influence on the rate-matching score performance. We try to preserve the 500, 1000, 1500, and 2000 strongest features for 720p format videos in our following experiments, respectively. Figure 5.16 draws the rate-matching score curves for SIFT and SURF features for the video *720p50_mobcal_ter*. Other videos have similar curves. It can be seen from Figure 5.16 that our algorithm improves the preservation performance also for other numbers of features. It should be noted that the gap between the default rate control approach and ours gets smaller when the number of features to be preserved is higher. With increasing number of features, our curve approaches the default rate control's curve, because more MBs are covered by features and only a few MBs belonged to higher groups. As a result, fewer MBs can be encoded using larger QP values, and the bitrate can not be reduced. An extreme example is that all MBs are covered by the first group, and our approach would produce the same results as the default rate control. In conclusion, the number of features to be preserved can be tuned as needed in some scenarios, and our approach always has a non-negative impact on the preservation performance.

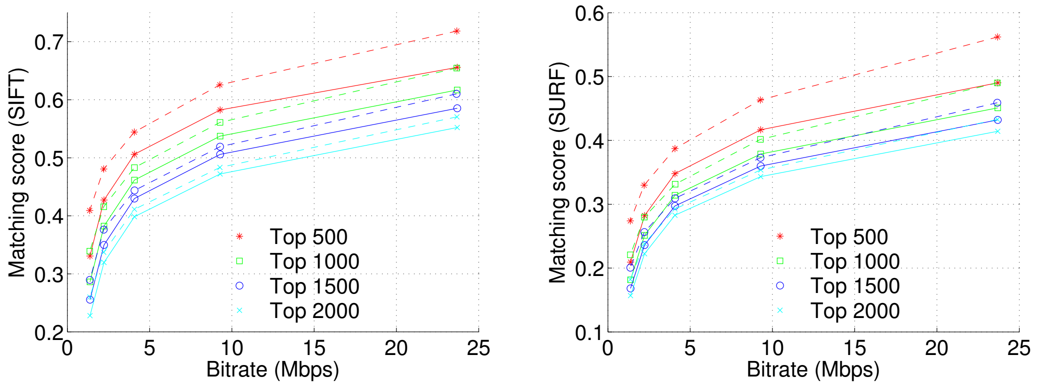

Figure 5.16: The top 500, 1000, 1500 and 2000 features to be preserved for the video *720p50_mobcal_ter*. The solid curves and dashed curves represent the results of the default rate control approach and our proposed one, respectively.

5.2.5 Human observer

The videos encoded by our proposed approach, however, did not take the human observer into consideration. To this end, many additional QP constraints are finally applied in order to ensure good visual quality. The resulting videos are more comfortable for human eyes while still achieving better feature preservation performance. Consequently, a trade-off between visual quality for humans and feature preservation performance is achieved.

In the previous approach, the MBs irrelevant for feature extraction are assigned a minimum number of bits because they are compressed by QP value 51. If many of the MBs are not covered by features, the videos encoded by our approach will be quite unacceptable for a human observer. In some scenarios, the videos are possibly consumed first by computer vision algorithms, and then checked by humans. Therefore, we need to take the visual quality into account in this case. Most of the videos encoded by our proposed approach actually provide acceptable visual quality. However, our algorithm can be easily tuned to take the human observer into account. The important adjustments and noteworthy points are listed here.

a) The MBs which are not relevant to feature extraction should not be compressed by QP value 51. They are treated as the last group in our previous approach.

b) Equation (5.13) and Equation (5.15) are changed as follows:

$$QP_{i,g}(1) = QP_{i,0}(1) + g * 3,\ i \geq 1 \tag{5.24}$$

We constrain the allowable QP difference between two adjacent groups to 3. Furthermore, the maximum allowable QP difference for the entire frame is set to 10. Thus, the QP for high-indexed groups is constrained by that of the first group as follows:

$$QP_{1,g}(1) = min\{QP_{1,0}(1) + 10, QP_{1,g}(1)\},\ g > 0 \tag{5.25}$$

c) Equation (5.21) is modified as follows:

$$QP_{i,g}(j) = max\{QP_{i,g-1}(j) + 5, min\{QP_{i,g-1}(j) + 2, QP_{i,g}(j)\}\},\ g > 0 \tag{5.26}$$

which makes the QP variation smaller compared to our previous approach. The $QP_{i,g}(j)$ should also be constrained to make sure that the maximum QP difference for this frame is 10.

$$QP_{i,g}(j) = min\{QP_{i,0}(j) + 10, QP_{i,g}(j)\},\ g > 0 \tag{5.27}$$

d) In the default rate control, the QP variation between two consecutive frames is set to 4. In our previous approach and the approach considering human observer, the allowed QP difference is also 4 for each group.

Table 5.4 and Table 5.5 present the comparison of matching scores for the CIF and 720p format videos. The approach additionally considering the human observer is named *Ours+HO*. The target bitrates are set according to the videos encoded by QP values 24, 30, and 36. Compared to the default rate control, our previous approach generally achieves the most significant improvement in terms of matching score. *Ours+HO* approach also improves the feature preservation performance. It should be noted that for some videos the *Ours+HO* approach yields the best performance. This can be explained as follows. First, our previous approach (*Ours*) does not assure the optimal performance because the QP assignment for I

Table 5.4: Matching score comparison for CIF format videos

Features	SIFT features				SURF features			
Video	Bit rate (kbps)	Matching Score (%)			Bit rate (kbps)	Matching Score (%)		
sequences	Ours+HO	Default	Ours	Ours+HO	Ours+HO	Default	Ours	Ours+HO
Bus	2314.50	68.61	**72.34**(+3.73)	71.93(+3.32)	2315.27	55.38	**60.46**(+5.08)	58.83(+3.45)
	997.19	52.88	**56.96**(+4.08)	55.06(+2.18)	996.37	40.65	**44.05**(+3.40)	43.09(+2.44)
	394.89	34.15	**37.52**(+3.37)	36.39(+2.24)	394.64	25.51	**27.52**(+2.01)	26.77(+1.26)
Coastguard	2530.89	61.95	**69.55**(+7.60)	68.21(+6.26)	2529.62	49.65	**61.57**(+11.92)	57.39(+7.73)
	1033.70	45.08	**54.42**(+9.34)	53.17(+8.09)	1034.20	34.93	**45.00**(+10.07)	41.75(+6.83)
	297.70	25.03	**30.51**(+5.48)	29.41(+4.38)	298.75	19.16	**26.52**(+7.36)	24.24(+5.08)
Container	482.77	62.13	66.97(+4.84)	**67.34**(+5.21)	482.44	51.92	58.61(+6.69)	**58.76**(+6.84)
	145.46	47.67	**51.89**(+4.22)	50.40(+2.73)	145.33	39.42	**42.47**(+3.05)	42.33(+2.91)
	57.01	33.37	**37.63**(+4.25)	37.18(+3.80)	56.89	24.11	**30.45**(+6.34)	28.71(+4.60)
Flower	2916.61	68.53	**71.49**(+2.96)	71.22(+2.69)	2919.24	45.79	**49.35**(+3.56)	48.06(+2.27)
	1356.41	52.56	**55.77**(+3.21)	55.01(+2.44)	1356.80	31.23	**33.14**(+1.91)	32.88(+1.65)
	496.17	33.34	**35.60**(+2.26)	35.26(+1.92)	495.74	18.84	19.65(+0.81)	**20.07**(+1.23)
Foreman	789.71	59.93	**60.91**(+0.99)	60.63(+0.71)	789.72	50.29	**52.99**(+2.70)	52.39(+2.11)
	315.05	43.08	**45.04**(+1.97)	44.70(+1.62)	314.95	33.73	**36.39**(+2.67)	35.52(+1.79)
	146.04	**28.62**	28.06(-0.57)	28.16(-0.46)	145.72	19.75	20.31(+0.56)	**20.95**(+1.21)
Mobile	3609.80	74.33	**79.21**(+4.88)	78.16(+3.83)	3616.85	57.80	**63.14**(+5.34)	61.94(+4.14)
	1492.08	60.00	**66.06**(+6.06)	64.58(+4.58)	1488.37	43.71	**48.19**(+4.47)	47.29(+3.57)
	459.76	43.63	**46.77**(+3.14)	46.32(+2.69)	459.56	30.73	**33.30**(+2.57)	32.33(+1.60)
Tempete	2465.50	67.74	**71.66**(+3.92)	71.11(+3.37)	2469.60	50.42	**53.65**(+3.23)	53.07(+2.65)
	927.96	51.32	**55.49**(+4.18)	54.39(+3.07)	928.80	35.77	**38.34**(+2.57)	37.57(+1.80)
	294.02	33.96	**35.87**(+1.91)	35.76(+1.80)	294.82	22.18	24.27(+2.09)	**24.55**(+2.37)
Waterfall	764.52	55.43	**58.41**(+2.98)	58.02(+2.59)	765.09	39.43	**41.57**(+2.15)	41.03(+1.61)
	260.84	37.74	**41.04**(+3.30)	**41.04**(+3.30)	260.64	25.01	**28.00**(+2.99)	27.45(+2.43)
	98.38	20.07	**22.45**(+2.38)	21.96(+1.88)	98.57	12.93	**15.08**(+2.15)	14.77(+1.84)

Table 5.5: Matching score comparison for 720p format videos

Features	SIFT features				SURF features			
Video	Bit rate (mbps)	Matching Score (%)			Bit rate (kbps)	Matching Score (%)		
sequences	Ours+HO	Default	Ours	Ours+HO	Ours+HO	Default	Ours	Ours+HO
Mobcal	23.67	61.68	**65.47**(+3.78)	65.02(+3.34)	23.67	45.09	**49.01**(+3.92)	48.31(+3.22)
	4.08	46.17	**48.31**(+2.14)	**48.31**(+2.14)	4.08	31.40	**33.14**(+1.74)	32.95(+1.55)
	1.37	28.61	**33.91**(+5.30)	33.57(+4.96)	1.37	18.16	22.07(+3.92)	**22.41**(+4.25)
Parkrun	71.32	71.13	**77.16**(+6.03)	76.01(+4.88)	71.33	50.88	**57.35**(+6.47)	55.61(+4.73)
	26.07	54.90	**59.08**(+4.19)	58.55(+3.66)	26.04	35.60	**38.67**(+3.07)	37.92(+2.32)
	9.62	36.79	**39.42**(+2.63)	39.36(+2.56)	9.63	21.90	**23.18**(+1.28)	22.82(+0.92)
Shields	22.53	64.43	**69.29**(+4.87)	68.85(+4.43)	22.53	50.52	**54.79**(+4.27)	53.76(+3.24)
	3.92	47.04	**50.60**(+3.56)	50.49(+3.45)	3.92	35.28	**38.16**(+2.88)	37.44(+2.16)
	1.31	29.88	**34.03**(+4.15)	33.81(+3.93)	1.31	22.03	24.98(+2.94)	**24.99**(+2.95)
Stockholm	29.47	64.51	**70.14**(+5.63)	69.42(+4.92)	29.48	48.82	**54.72**(+5.90)	53.71(+4.88)
	2.65	46.12	48.19(+2.07)	**48.25**(+2.13)	2.65	31.66	**33.94**(+2.28)	33.79(+2.13)
	0.88	27.87	32.77(+4.90)	**32.83**(+4.96)	0.88	17.82	**21.86**(+4.03)	21.85(+4.02)

frames is based on a heuristic observation. Many of the neighboring MBs are encoded by a large QP difference which results in a large gradient difference. Second, it is not always true for different videos that $\Delta QP = 5$ and $\Delta QP = 6$ for SURF and SIFT, respectively. For example, the lower left plot in Figure 5.11 shows that the ΔQP for SURF is smaller than 6 when the QP value for the first octave is larger than 35. Therefore, a smaller ΔQP should be better. The higher matching scores of the *Ours+HO* approach for the low quality videos *flower*, *foreman*, *tempete*, *mobcal*, and *shields* (the third row for each video in Table 5.4 and Table 5.5) justify this observation. Anyway, the $\Delta QP = 5$ and $\Delta QP = 6$ for SURF and SIFT are heuristic and can be adjusted at lower qualities for some videos.

5.2.5.1 Resulting frames

Figure 5.17 shows the first frames from the eight H.264/AVC-encoded videos from the *Ours+HO* approach with target bitrates set according to the videos encoded by a fixed QP value 30. Obviously, the frames are still enjoyable for human eyes. Because the feature extractors detect blobs or corners, the complex regions of frames have more small scale features, while the flat regions have less features or tend to have no features at all. In our algorithm, these complex regions are encoded with high quality, and human eyes also focus more on these regions. In our experiments, some of the resulting videos are actually even better than the videos encoded by the default rate control algorithm, especially due to the QP constraint for the over-use of target bits.

5.2.5.2 Subjective test

In order to assess how the proposed rate control method impacts the video perception, we perform a subjective test using the SAMVIQ method [IR07], which is specifically designed for assessing multimedia applications. A five-second-long sequence is encoded with the default rate control and with the *Ours+HO* approach (bitrate is set according to the videos encoded by a fixed QP value 36), and then presented to the test subjects. Besides the two evaluated rate control approaches, the test subjects also rate a reference sequence (best possible quality), a hidden reference, and a medium quality sequence. The quality anchors are recommended to stabilize the test results [IR07]. Test subjects rate the videos on a continuous scale from 0 to 100. After the screening procedure described in [IR07], the data of 16 test subjects is verified to be valid. For each sequence, the average rating over the 16 test subjects is taken and a differential quality score - differential mean opinion score (DMOS) [IT08] - is then computed by:

$$DMOS = R(sequence) - R(hiddenreference) + 100 \tag{5.28}$$

where R is the average rating over the 16 test subjects. The DMOS value is used in our data analysis as the subjective quality rating. Figure 5.18 shows the DMOS values for the 8 tested videos as well as the standard deviation of the ratings. Due to the large QP difference in one frame, the *Ours+HO* approach in general shows as expected an inferior visual quality compared to the default one. For the test videos 3, 5, and 8, our DMOS values are close or even surpass the default ones.

5.2.6 Retrieval performance

The previous experiments compare the number of preserved features at frame level for different approaches. Generally the more features are preserved, the better the performance the retrieval system should have. In order to test the ultimate performance for different

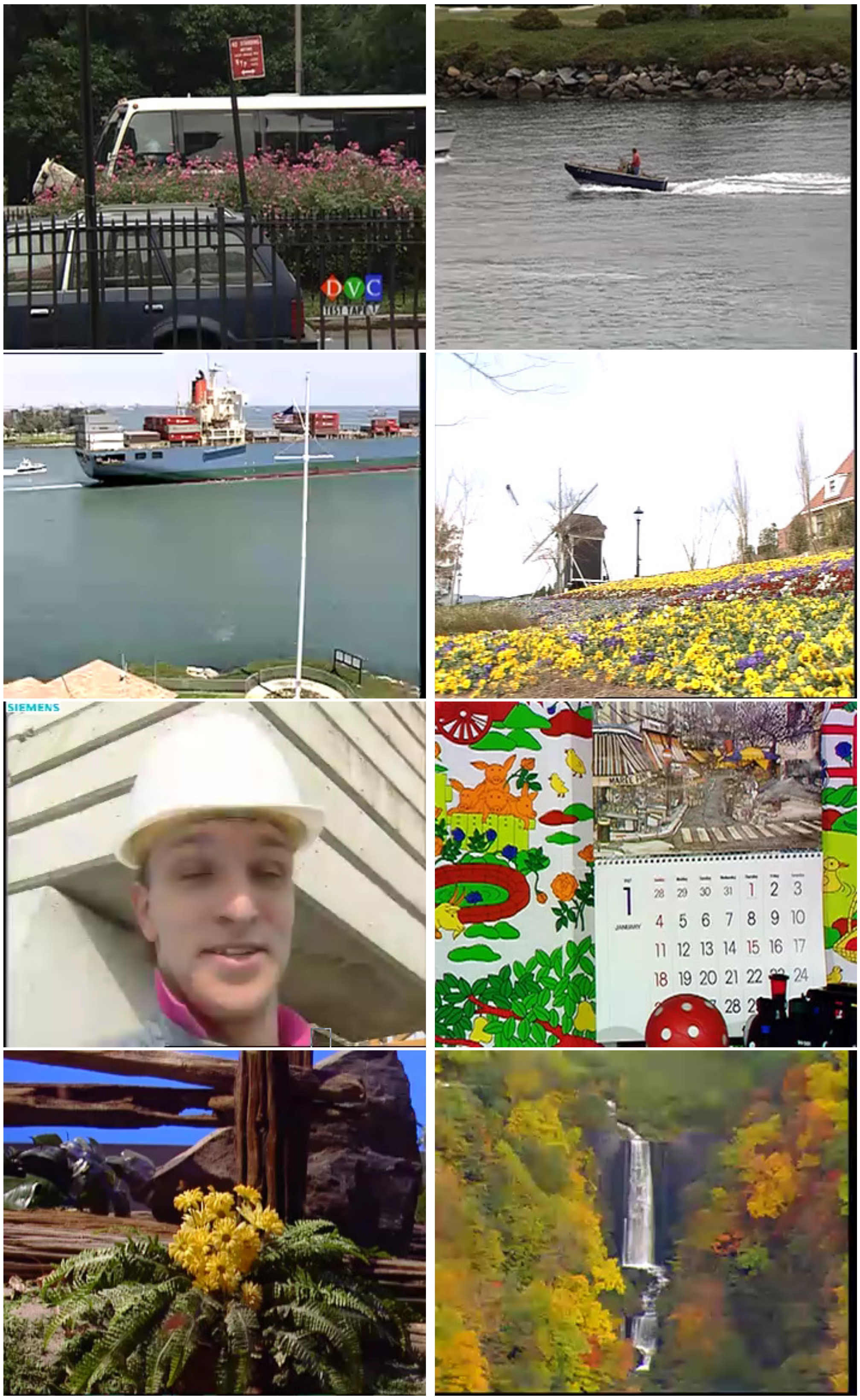

Figure 5.17: First frames from the H.264/AVC-encoded video sequences in CIF format.

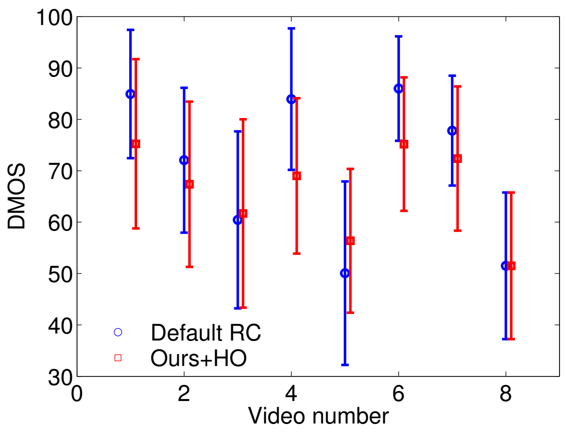

Figure 5.18: DMOS comparison between the videos encoded with the default rate control and with the proposed *Ours+HO* approach.

approaches, the Stanford MAR dataset [MTC+13] and the image retrieval system [PCI+07] (except the fast spatial matching part) are used because the video search engine [SZ03] is actually similar to an image retrieval system. The 23 reference images in the Stanford MAR dataset and the Oxford Buildings dataset are combined as a joint database consisting of 5086 images. The number of returned images for each query image is 5% of the overall database.

Similar to [MTC+13], the strongest 200 SIFT features are targeted to be preserved, and the videos are first encoded using fixed QP values 35, 39, 43, and 47. Other encoding parameters are: IPPP··· structure, IntraPeriod = 50 frames, FrameRate = 30. Then, the target bitrates of *Ours* and *Ours+HO* rate control approaches are set according to these fixed QP encoded videos. The frames of 23 encoded videos are used as the query images to retrieve the relevant reference images in the combined database. Thus, 2300 query frames in total are used to calculate the mAP score. Table 5.6 shows the mAP scores for the different approaches. The mAP score is 0.8398 for the uncompressed videos. The mAP scores for QP value 35 already approach the one for the uncompressed frames in our experiment. The mAP scores drop for larger QP values because the feature detection process is significantly influenced by the blocking artifacts. Our proposed rate control approach improves the retrieval performance for all the query video frames. The gains increase as the target rate becomes smaller.

Table 5.6: mean Average Precision for SIFT features

	mean Average Precision (mAP)			
QP	Fixed QP	Ours	Ours+HO	Uncompressed
35	0.8070	0.8087(+0.0017)	0.8168(+0.0098)	0.8398
39	0.7380	0.7639(+0.0259)	0.7543(+0.0163)	
43	0.6537	0.6836(+0.0299)	0.6779(+0.0242)	
47	0.4709	0.5068(+0.0359)	0.5007(+0.0298)	

5.3 Summary

In the first section, two R-D optimization strategies for preservation of SIFT features in H.264/AVC-encoded videos are proposed. In the first approach, a matching score-based R-D optimization process is applied. In the second approach, a repeatability-based method for bit allocation is proposed. The results showed that both our proposed approaches achieve better matching performance compared to normally H.264/AVC-encoded video. Notably, both approaches proposed in this work are computationally complex for real-time encoding. Therefore, in the second section, a heuristic QP assignment is used for each group according to their importance in feature extraction. Based on this, a novel rate control approach is proposed to preserve the important local image features. The results show that our approach preserves a greater number of features, while achieving the target bitrate. In further experiments, different numbers of features are targeted to be preserved. The results show that our approach still achieves better performance. Then, to make the videos more suitable for human observers, we adjust our approach and find a trade-off between matching score and visual quality. The results show that the encoded videos are more pleasant to humans, while the matching scores are improved. Finally, it is shown that the improved feature preservation also translated into an improved mAP score in a real retrieval system. The entire framework can be applied to features other than SIFT and SURF. The videos encoded using our approaches are fully standard-compatible, and any H.264/AVC decoder can be used to decode them.

Chapter 6

Keypoint encoding and transmission as side information

Image or video compression has been shown to have an adverse effect on feature-matching performance. The negative impact of compression can be reduced by using the keypoints extracted from uncompressed video for descriptor calculation from the compressed image, as shown in Section 3.3. Based on this observation, in Section 6.1, we discuss an approach to efficiently encode the locations, scales, and orientations of the keypoints extracted from an uncompressed image. Furthermore, a new approach is proposed for selecting relevant yet fragile keypoints as side information for the image, thus further reducing the volume of data. In a further experiment in Section 6.2, it is shown that the observation also applies to video sequences. Based on this observation, we propose to provide these keypoints to the server as side information and to extract only the descriptors from the compressed video. First, four different frame types for keypoint encoding are proposed to address different types of changes in video content. These frame types represent a new scene, the same scene, a slowly changing scene, or a rapidly moving scene and are determined by comparing features between successive video frames. Then, Intra, Skip and Inter modes of encoding the keypoints for different frame types are proposed. For example, keypoints for new scenes are encoded using the Intra mode, and keypoints for unchanged scenes are skipped. As a result, the bitrate of the side information related to keypoint encoding is significantly reduced. Finally, pairwise matching and image retrieval experiments are conducted to evaluate the performance of the proposed approach using the Stanford MAR dataset and 720p format videos. The results show that the proposed approach offers significantly improved feature matching and image retrieval performance at a given bitrate. The keypoint encoding idea is shown in Figure 6.1.

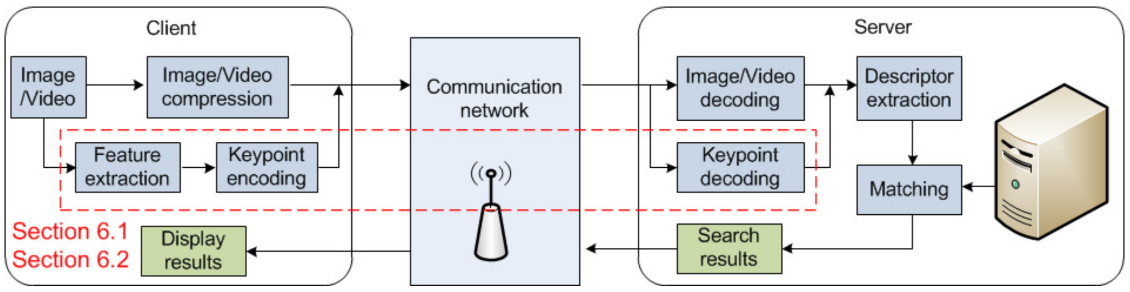

Figure 6.1: Architecture of video encoding and transmission

6.1 Keypoint encoding for images at low bitrates

In Section 3.3, we observe that keypoint detection is easily affected by compression artifacts, while descriptor calculation is more robust. Therefore, in this section, we investigate whether explicit signaling of keypoint information (location, scale, and orientation) improves the quality of the features extracted from compressed images. In the literature, there are a few related works on keypoint encoding. Tsai et al. [TCT$^+$12] and CDVS [MPE] proposed approaches to encode the locations of keypoints using sum-based context-based arithmetic coding. Yue et al. [YSYW13, YSWY12] proposed encoding of the locations, scales, and orientations of large-scale features. They encoded a downsampled image with a fixed QP, and only a few keypoints with large scales were preserved because the small-scale features were of no use in their scenario. Then, the downsampled image and the encoded keypoints were transmitted to the server. In this work, we are interested in feature preservation performance for features of different scales at low bitrates. We build on these works and propose an approach that combines explicit keypoint encoding with standard image compression. The keypoint information is transmitted as side information along with the compressed image. Under the same experimental settings as in Section 3.1, the experiments are performed with JPEG, H.264/AVC Intra, and H.265/HEVC Intra, and present the results as plots that show the total bitrate versus the number of successfully matched features.

6.1.1 Keypoint quantization and encoding

Given that the location, scale, and orientation of keypoints are represented using floating-point numbers, we need to quantize them for efficient transmission. Different quantization methods for location, scale, and orientation are applied. First, the experimentally determined statistics for keypoint quantization are presented. From these results, we determine the quantization coarseness for which matching results are obtained similar to those of unquantized keypoints.

6.1.1.1 Keypoint quantization

For locations, Tsai et al. [TCT+12] used a spatial grid over the original image with step sizes of 4, 6, and 8 pixels. In another work [YSYW13], locations are quantized into integer values, meaning they were quantized on a grid with a step size of 1 pixel as follows:

$$\tilde{l_i} = f \cdot round(\frac{l_i}{f}) \tag{6.1}$$

where f is the quantization factor. In our experiments, we determine the quantization factor to be used such that the quantized locations does not significantly affect the number of feature matches. f is varied from 1 to 5.

In SIFT, the scale σ_i can be represented as follows:

$$\sigma_i = \sigma_0 2^{(o+s/3)} + \Delta\sigma \tag{6.2}$$

where σ_0 is a base scale offset (i.e., 2.0159), o is an octave ranging from 0 to x depending on the size of the image (x is less than 4 for our dataset), and s is the integral scale in the range [0, 2]. Thus, 3 and 2 bits are sufficient to represent o and s, respectively. $\Delta\sigma$ is an offset calculated to increase the accuracy of the scale estimates in SIFT keypoint detection. This process fits a quadratic polynomial with the values of the detected scale space extremum ($\sigma_0 2^{(o+s/3)}$) to localize the more accurate scales with a resolution higher than the scale-sampling density (see Section 2.1.1.1). Thus, the value $\Delta\sigma$ is related to the scale space extremum ($\sigma_0 2^{(o+s/3)}$). We calculate the difference between the detected scale σ_i and its corresponding scale space extremum ($\sigma_0 2^{(o+s/3)}$). Then, the difference is normalized as follows:

$$\Delta\sigma_n = (\sigma_i - \sigma_0 2^{(o+s/3)})/\sigma_0 2^{(o+s/3)} \tag{6.3}$$

Figure 6.2 shows the magnitudes and the distribution of $\Delta\sigma_n$ for the features extracted from all eight video sequences. From the magnitudes (left plot), we can see that the values of the normalized differences are mainly within [-0.2, 0.2]. The distribution on the right in Figure 6.2 has an approximately Gaussian form. The normalized difference is encoded using the Lloyd-Max quantization algorithm. The codebook and partition are trained using these statistics. We assign 3, 2, 1, and 0 bits for $\Delta\sigma_n$, where 0 bits means the $\sigma_0 2^{(o+s/3)}$ is used without adding any offset. Thus, the scales (o, s, and $\Delta\sigma_n$) are assigned 8, 7, 6, and 5 bits, respectively, for testing.

Similar to location quantization, orientations are quantized as follows.

$$E(\theta_i) = round((\frac{\theta_i}{2\pi} + 0.75) \times (2^t - 1)) \tag{6.4}$$

$$\tilde{\theta_i} = \Big(\frac{E(\theta_i)}{2^t - 1} - 0.75\Big) \cdot 2\pi \tag{6.5}$$

where 0.75 is an offset that ensures the values ($\frac{\theta_i}{2\pi} + 0.75$) are within [0, 1) in the VLFeat SIFT implementation. Then, the index $E(\theta_i)$ can be represented by values within [0, 2^t) and

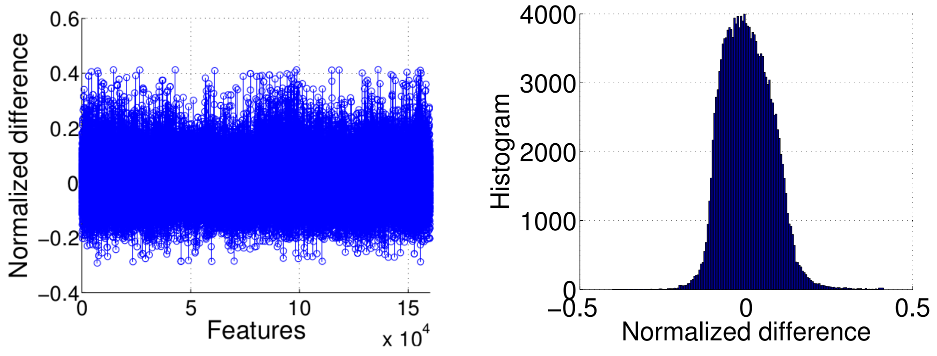

Figure 6.2: Magnitude and distribution of $\Delta\sigma_n$ for 160,000 features extracted from eight test video sequences.

can be encoded by fixed-length coding with t bits. In our test, t is set to 7, 6, 5, 4, and 3, which quantize the orientations into 128, 64, 32, 16, and 8 levels, respectively.

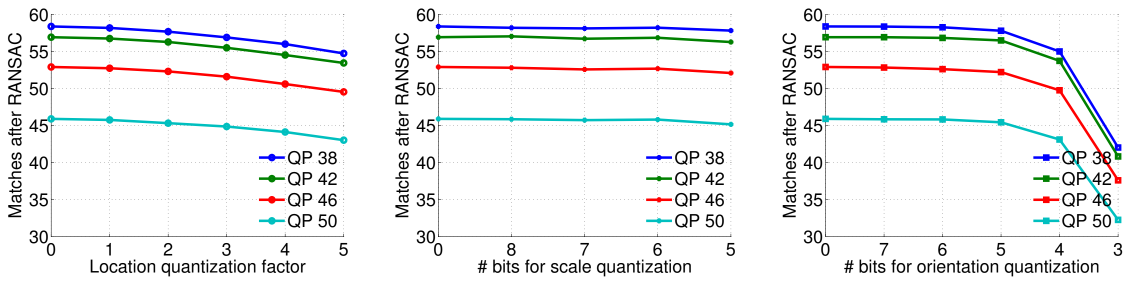

Figure 6.3: Effects of keypoint quantization. The value 0 on the horizontal axis indicates the idealistic uncompressed keypoints.

Figure 6.3 shows the effects of keypoint quantization on the matching performance. To be able to individually evaluate the impact of quantization on one quantity (e.g., location), the other quantities (e.g., scale and orientation) are represented using the original floating-point values. The matching performance for uncompressed keypoints is shown at the very left of each plot (value 0 on the horizontal axis). In the left plot, it can be seen that the matching performance starts dropping at a quantization factor of 2 for locations. In the middle plot, we can see that scale quantization does not affect the matching performance, even when using only 6 bits. In the right plot, the matching performance drops when encoding with 5 bits for orientations.

Our goal is to select reasonable quantization levels such that the matching performance is not affected, and the number of bits required is small. Thus, we choose the location quantization factor 1 (see left plot), 6 bits for scale (see middle plot), and 6 bits for orientation (see right plot) in the following experiments. As a result, the quantized keypoints yield feature-matching performance similar to that with the ideal uncompressed keypoints. In total, the encoding of scale and orientation for one keypoint requires 12 bits.

6.1.1.2 Context model for location coding

As explained in the previous section, the original locations are quantized into integer values (quantization factor 1). To encode the quantized locations, the same approach as in CDVS [MPE] is used, i.e., sum-based context-based arithmetic coding. The number of bits for location encoding depends on the distribution and number of keypoints to be encoded. In sum-based context-based arithmetic coding, first, the context model must be trained. Similar to [TCT$^+$12] and CDVS [MPE], the Inria Holidays Dataset* and the Caltech Building Dataset† are used for training. Note that the joint dataset comprises 1741 images. Because the location quantization factor is set to 1, the block width is correspondingly set to 1. Unlike the fixed number of features to be encoded in the CDVS Test Model, a different number of keypoints (K^3, see Table 6.1) for each image needs to be encoded in the following experiments. Thus, two sets are tested to determine the context range (i.e., 200 features and 100 features) for encoding these locations. For simplicity, we use the same test dataset and encode the quantized locations again after obtaining the context model. The context range of 1 to 60 is selected from, and the average number of bits used for encoding the locations is recorded. For both sets, the curves flatten beyond the context range of 45 with a minimum at 49. Thus, the context range of 49 is selected, and the quantized locations are encoded using the correspondingly trained model.

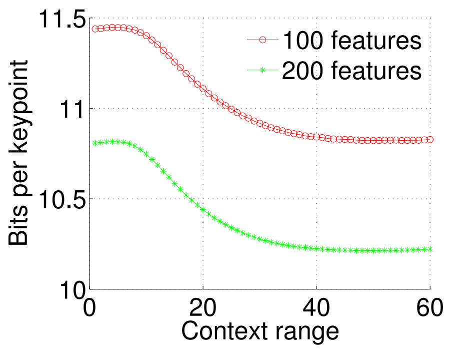

Figure 6.4: Average number of bits required for keypoint location encoding as a function of context range.

6.1.1.3 Keypoint set

The previously observed quantization levels for locations, scales, and orientations are used. The set of all original keypoints is expressed as $K = \{k_1, k_2, ..., k_N\}$, and the set of all original descriptors is $D = \{d_1, d_2, ..., d_N\}$, where N is the number of keypoints (i.e., 200). Similarly, the set of all quantized keypoints is expressed as $K^1 = \{k_1^1, k_2^1, ..., k_N^1\}$, and the extracted

*http://lear.inrialpes.fr/people/jegou/data.php

†http://vision.caltech.edu/malaa/datasets/caltech-buildings/

descriptor set is $D^1 = \{d_1^1, d_2^1, ..., d_N^1\}$. In the next section, we will show that a few keypoints in the set K^1 can be removed before transmitting them to the server, leading to a further rate reduction for the side information.

6.1.2 Keypoint selection and transmission

To further reduce the bitrate required for the side information, two steps are proposed to reduce the number of keypoints to be encoded.

First, spurious keypoints are removed from K^1. The definition of spurious keypoints is that the descriptors extracted around those keypoints from the compressed image $\overline{I}$ cannot be matched to the descriptors extracted from the original (uncompressed) image I. The remaining keypoints are expressed as follows:

$$K^2 = \left\{k_i^1 \in K^1 | M(d_i^1, d_i) = 1\right\} \tag{6.6}$$

where $M(\cdot,\cdot) = 1$ denotes the matched pairs of d_i^1 and d_i (see Equation (2.16)). Please note that the matching process checks two aspects for a pair of features from d_i^1 and d_i. First, the descriptors are required to be matched according to the NNDR strategy. Second, the pair should be located within 1.5 pixels. If a pair meets these two requirements, it is considered as a matching pair.

In the previous step, only the keypoints derived from the original image I are considered. However, application of the detector algorithm to the compressed image $\overline{I}$ generates some keypoints as well. A few of them are good enough to produce correct descriptors relative to the descriptors of the uncompressed image, especially when the compressed image $\overline{I}$ is of good quality. Thus, we propose removing the duplicated keypoints in K^2 relative to the keypoints $\overline{K}$ (see Equation (3.1)) extracted from the compressed image $\overline{I}$. The useful keypoints K^c of the compressed image is expressed as follows:

$$K^c = \{\overline{k_i} \in \overline{K} | M(\overline{d_i}, d_i) = 1\} \tag{6.7}$$

The remaining keypoints are expressed as:

$$K^3 = \{K^2 \setminus K^c\},\ thus,\ K^3 \subseteq K^2 \tag{6.8}$$

The information C about whether a keypoint from the compressed image is useful is encoded by binary arithmetic coding, and the resulting bitrate is added to the total keypoints rate. Note that after removing the duplicated keypoints, the remaining keypoints K^3 and C are encoded and transmitted as side information.

6.1.3 Keypoints at server

At the server, the keypoints K^3 are decoded, and the useful keypoints K^c are selected from the keypoints extracted from the compressed image using the information C. These two sets of keypoints are then combined, and the descriptors are calculated around these keypoints.

$$K^* = \{K^3 \cup K^c\},\ D^* = \Psi(K^*|\overline{I}) \tag{6.9}$$

where K^* is the final set of keypoints for which the descriptor set D^* is extracted from the compressed image $\overline{I}$.

6.1.4 Experimental results

Table 6.1 shows the average number of keypoints in the sets K^1, K^2, and K^3. As described above, K^2 comprises the keypoints remaining after removing the spurious ones relative to K^1. For large QP values, K^2 is small owing to the presence of strong compression artifacts, which lead to the generation of many spurious features. K^3 comprises the keypoints remaining after dropping the duplicated ones relative to the keypoints obtained from the compressed image. For small QP values, the detector extracts many useful keypoints from the compressed image. Thus, there remain only a few keypoints K^3 to be encoded. According to Equations (6.7) and (6.8), at the server side, the useful keypoints K^c can be obtained from the keypoints $\overline{K}$ of the compressed image and the information C. The keypoints K^3 and K^c are combined, and the number of keypoints K^* is identical to the number of keypoints in K^2.

Table 6.1: Average number of keypoints per frame

QP	38	42	46	50
K^1	200	200	200	200
K^2	192.6	189.0	178.8	155.2
K^3	90.0	111.9	130.1	131.4
K^*	192.6	189.0	178.8	155.2

Figure 6.5 shows the original keypoints K in red and the combined keypoints K^* in green. Most of them overlap, indicating that our proposed keypoints are almost the same as the original keypoints. However, a few original keypoints have no counterparts. The descriptors calculated from these keypoints are spurious, so they are discarded before transmission to save bits.

The solid black line in Figure 6.6 shows the final matching performance of our proposed approach as a function of the total bitrate used for the compressed image plus the side information. It is seen that the number of matches is improved significantly. At an average of approximately 42 matches per image, the proposed approach achieves 6.5× bitrate reduction compared to JPEG, 3.9× bitrate reduction compared with H.264/AVC Intra, and 1.6× bitrate reduction compared with H.265/HEVC Intra. Note that there is a trade-off between the matching performance and the quantization levels for locations, scales, and orientations. We only present the results obtained using the selected quantization levels, which seem to be reasonable for improving the matching performance.

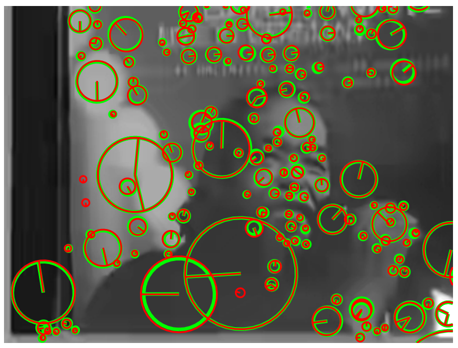
Figure 6.5: Original keypoints K are in red and the combined keypoints K^* at the server are shown in green.

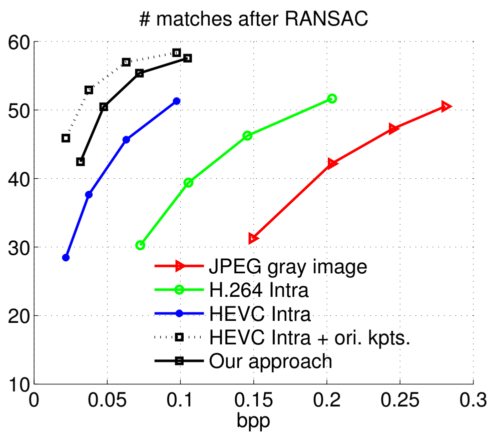

Figure 6.6: Comparison of matching performance of different approaches. The proposed approach is represented by using the solid black line. The bitrate includes the encoded keypoints and the H.265/HEVC Intra-encoded image.

6.2 Keypoint encoding for video at low bitrates

Few studies have addressed the compression of features extracted from videos. In [BCR$^+$14, BAC$^+$14], the authors proposed intra- and inter-frame coding modes for SIFT descriptors and binary local descriptors, respectively, to encode the descriptors extracted from a video sequence. The authors of [MTC$^+$13] proposed inter-frame predictive coding techniques for image patches and keypoint locations, and they subsequently proposed an *inter-descriptor coding* scheme [MCT$^+$14] to encode the descriptors extracted from such patches. In these approaches, descriptors/patches are extracted from videos and compressed for transmission; however, the videos themselves are not stored or sent to a server. By contrast, in this work, compressed videos are transmitted via a communication network to a server to perform feature extraction. The advantages previously discussed with respect to images also apply to videos. Based on our previous study regarding images in Section 6.1, we propose in this work a predictive keypoint encoding approach to encode the original keypoints extracted from uncompressed videos. In the proposed approach, the compressed keypoints are sent as side information along with the compressed videos at low bitrates. At the server, the decoded keypoints (locations, scales, and orientations) are used to extract feature descriptors from the videos. The proposed approach is illustrated in Fig. 6.1. The proposed framework is fundamentally different from those reported in other studies [BCR+14, MCT+14], which encode and transmit the descriptors. By contrast, in the proposed approach, only the keypoints are encoded and transmitted along with the compressed video. To evaluate the proposed approach, experiments are conducted using the widely used H.264/AVC and H.265/HEVC standards for video encoding. The results are presented as plots that show the number of successfully matched features versus the bitrate. In addition, we show the percentages of images that are successfully retrieved using various approaches via a content-based image retrieval engine.

6.2.1 Impact of video compression on feature quality

In this section, the impact of video compression on the features extracted from compressed videos is investigated. For this purpose, we first extract SIFT features from videos encoded using H.264/AVC and H.265/HEVC and compare these features with the features extracted from a set of uncompressed reference images. Then, we plot the number of matches as a function of bitrate and compare the results for the different video compression standards. Similar to Section 3.1, the Stanford MAR dataset, the JM reference software, the HEVC Test Model, the VLFeat SIFT implementation, the NNDR strategy, and the number of matches after RANSAC are used in this work.

6.2.1.1 Feature-matching performance for different video compression standards

In the previous work designed for images, the video sequences in the Stanford MAR dataset are treated as individual images and encoded using H.264/AVC Intra and H.265/HEVC Intra. In this work, the eight test video sequences are encoded as normal videos. In order to be able to compare our results with the patch-based encoding approaches in [MTC+13, MCT+14], we use the same parameters as in [MTC+13, MCT+14] for H.264/AVC. Additionally, we use more QP values to produce high-quality videos. The settings are as follows: the IPPP$\cdots$ structure, IntraPeriod = 50 frames, $\text{QP}_{Pframes} = \{26, 30, 34, 38, 42, 46, 50\}$, and $\text{QP}_{Iframes} = \text{QP}_{Pframes}$ -3. The parameters used for H.265/HEVC encoding are the same as those in the example provided in the HEVC version 16.0 manual [HHIa]. The GOP structure is IBBBPBBBP$\cdots$, and QP = $\{22, 26, 30, 34, 38, 42, 46, 50\}$. In the following experiments, SIFT features are extracted from the compressed frames and compared with the SIFT features extracted from the corresponding uncompressed reference images. The number of matching features after RANSAC is applied and the bitrates for the encoded videos are averaged over all test frames. The solid red and green curves in Fig. 6.7 represent the feature-matching performance for H.264/AVC- and H.265/HEVC-encoded videos as a function of bitrate. The H.265/HEVC-encoded videos exhibit much better performance than the videos encoded with H.264/AVC in terms of the number of feature matches at a given bitrate.

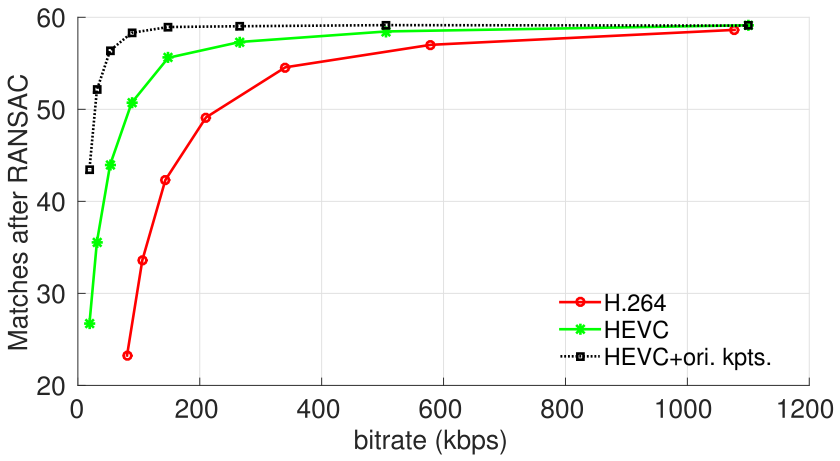

Figure 6.7: SIFT-feature-matching performance for different video compression schemes. The dotted line shows the results obtained for descriptors calculated from the H.265/HEVC-encoded videos using the original keypoints extracted from the uncompressed video frames.

Similar to our previous study on keypoint encoding for improved feature extraction from compressed images in Section 6.1, a simple experiment is performed to demonstrate that the results of keypoint detection can be easily affected by video compression artifacts and that descriptors are more robust. To this end, we calculate the descriptors from the compressed video frames using the keypoints extracted from the original (uncompressed) frames. This idea can also be expressed by the formulae in Section 3.3. Because the original keypoints k_i cannot be obtained from the compressed frame $\overline{I}$, we present the matching results obtained based on H.265/HEVC-encoded videos as a dotted line (upper bound) in Fig. 6.7. For videos of higher quality, the gap between the *HEVC* approach and the *HEVC+ori. kpts.* approach in Fig. 6.7 becomes increasingly smaller. However, high-quality videos also require a much higher bitrate and can be transmitted only if the necessary network resources are available. To avoid excessive bandwidth requirements for applications such as mobile visual search or video surveillance, we must compress the data to a low bitrate for transmission. This is the motivation for the current research on feature compression algorithms in the literature as well as the MPEG CDVS standard. Thus, our goal is not to provide high-quality video for human observers. Instead, we are interested in videos encoded at low bitrates for communication networks of limited capacity. The results indicate that if the original keypoints are preserved, then the feature-matching performance can be improved, especially for strong compression (i.e., low-bitrate encoding). This observation motivates us to encode the keypoints and send them as side information along with such videos compressed to low bitrates. In the following experiments, we will use QP = {38, 42, 46, 50} for H.264/AVC and H.265/HEVC encoding.

6.2.2 Core ideas

In our previous study in Section 6.1, we presented a keypoint encoding approach for still images. Applying this approach directly to individual frames in a video sequence would significantly increase the bitrate, as will be discussed in Section 6.2.4. To address this issue, similar to the conventional inter-frame prediction scheme in video coding, we propose several keypoint prediction approaches that significantly reduce the number of keypoints to be encoded and thus the bitrate required for the side information.

6.2.2.1 Frame types for keypoint encoding

The locations, scales, and orientations of keypoints detected in consecutive frames are related. If the keypoints are detected independently for each frame, then some of the keypoints may disappear or reappear across the frames as a result of the feature detection process. However, a keypoint that has disappeared from the previous frame may still yield a useful descriptor for the current frame. To address this issue, the authors of [MTC$^+$13] proposed a *temporally coherent keypoint detector* in which the detected patches are propagated to consecutive video frames. Unlike their approach, we extract keypoints and use them to predict the keypoints (locations, scales, and orientations) in consecutive video frames because, in our case, the video itself is also available. Here, we introduce four different types of frames for keypoint encoding. These frame types are illustrated in Fig. 6.8. Two of the four frame types are similar to those proposed in [MTC$^+$13]: *Detection frames* (*D-frames*) and *Skip frames* (*S-frames*). For D-frames, the keypoints are extracted using a conventional feature detection process. For S-frames, all keypoints are estimated using the keypoints from the previous frame, and the direct encoding of the keypoints is skipped. Therefore, S-frames enable a significant reduction in the amount of side information, as discussed in the following section. After a certain number of frames, it might no longer be possible to estimate the keypoints of the current frame using the previously identified keypoints because, for example, the object of interest is leaving the field of view or the estimated keypoints do not yield effective descriptors calculated from the current video frame. To address this case, we add a third type of video frame, termed an *Update frame* (*U-frame*), to update the keypoints. In contrast to D-frames, U-frames combine both conventionally detected keypoints and forward-estimated keypoints because a few of the estimated keypoints are still sufficient for calculating the descriptors. However, we find that if the scene is moving quickly, then even U-frames will generate a large number of bits. To resolve this issue, we add a fourth type of frame, called a *Null frame* (*N-frame*), for which keypoint encoding and transmission are switched off. For N-frames, no side information is transmitted, and the features are extracted exclusively from the compressed frames. The method of determining the frame types and the keypoint encoding processes for the different frame types are explained in the following sections.

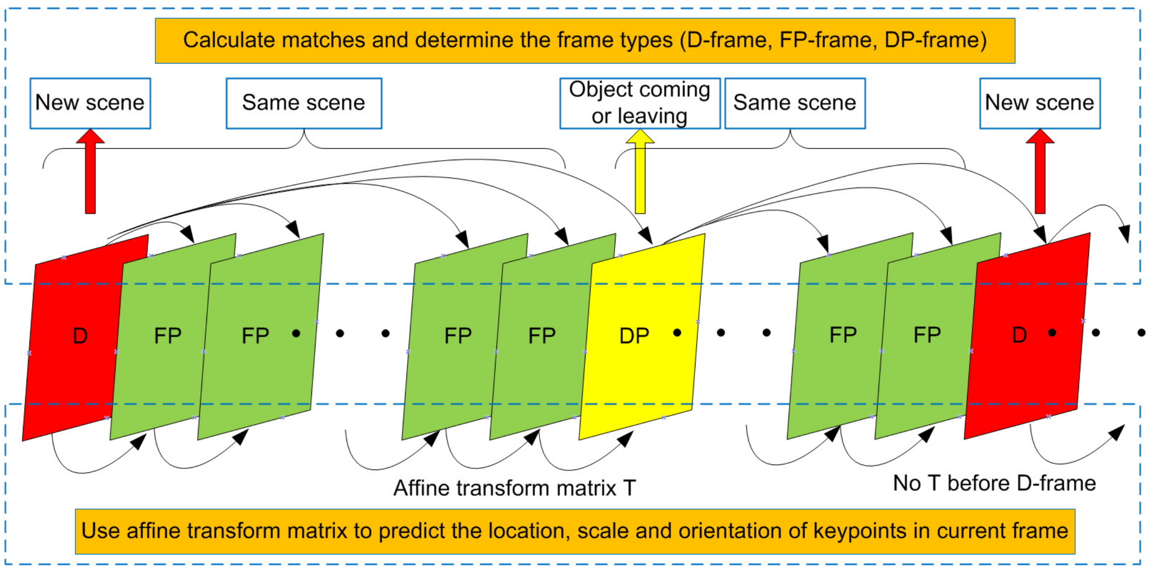
Calculate matches and determine the frame types (D-frame, FP-frame, DP-frame)
New scene
Same scene
Object coming or leaving
Same scene
New scene
D
FP
FP
FP
FP
DP
FP
FP
D
Affine transform matrix T
No T before D-frame
Use affine transform matrix to predict the location, scale and orientation of keypoints in current frame

Figure 6.8: Examples of the four frame types for keypoint encoding.

6.2.2.2 Keypoint encoder

As previously described, we predict the keypoints for S- and U-frames from the keypoints in the previous frame, as illustrated in Fig. 6.9. Note that the keypoint decoder follows the reverse process. Below, we define the terminology that will be used throughout this paper for clarity.

• *Estimate* means to estimate the locations, scales and orientations of the keypoints in the current frame using keypoints from previous frames.

• *Update* means to estimate the keypoints and also use differential keypoint encoding to update the keypoints.

• *Predict* means to either estimate or update the keypoints. Note that although estimate and predict are synonyms, they are used for different purposes in this paper.

• In *Intra mode*, the keypoints are extracted by conventional means from the current frame without reference to keypoints from previous frames and are encoded using a process similar to that presented in Section 6.1.

• In *Skip mode*, the keypoints are estimated and then stored in the buffer. Thus, differential keypoint encoding and transmission are skipped.

• In *Inter mode*, the keypoints are estimated and the differential encoder is used to update their locations, scales and orientations. Afterward, they are stored in the buffer.

Note that only the Intra mode is used to encode the keypoints for D-frames; only the Skip mode is used for S-frames; all three modes (the Intra, Skip, and Inter modes) are used for U-frames; and no keypoint encoding is performed for N-frames. We will explain the components of the diagram in Fig. 6.9 in detail in the following sections.

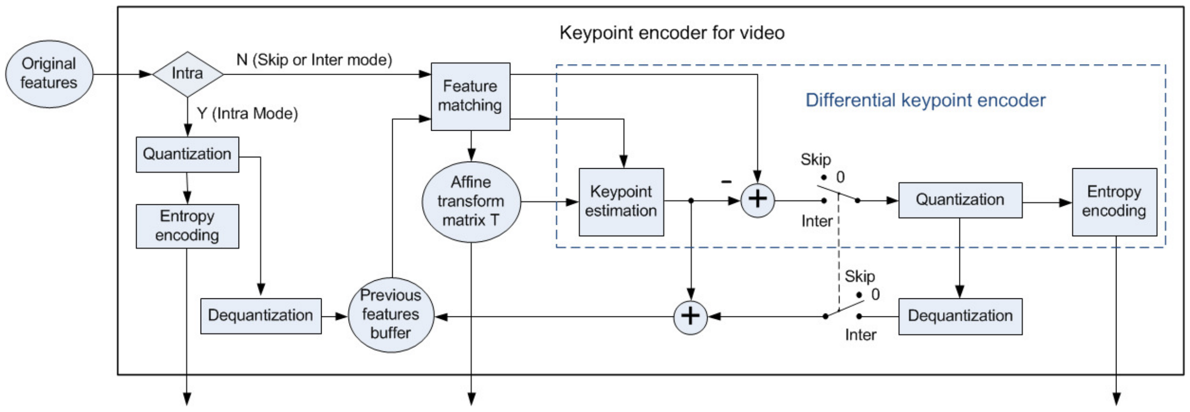

Figure 6.9: Keypoint encoding for video.

6.2.3 Keypoint prediction

When the camera or the object in a video is moving, the locations, scales and orientations of the detected keypoints are also gradually changing. Fig. 6.10 shows the keypoints detected in frame 1 and frame 20 of the video sequence *Barry White*. We can see that the keypoints are still closely related. The red squares indicate a pair of related keypoints. However, the location, scale and orientation of the keypoint have changed; thus, we must predict the keypoints for the current frame using the previous keypoints.

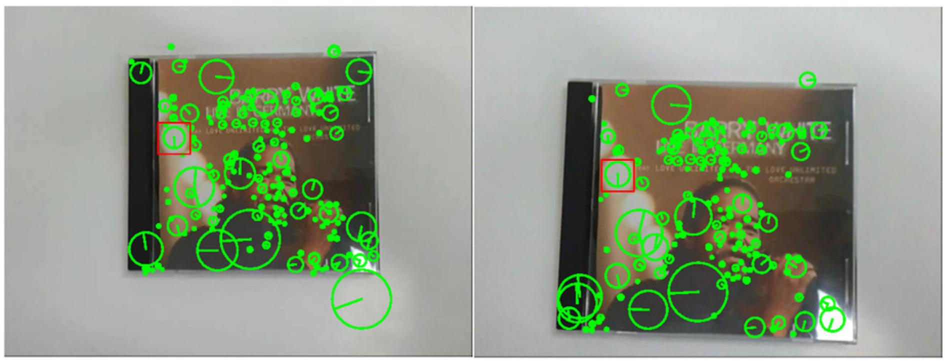

Figure 6.10: Keypoints from video frame 1 and video frame 20 in the example video *Barry White*.

6.2.3.1 Feature matching and affine transformation

As shown in Fig. 6.8, we determine the frame types for keypoint encoding by matching the features of the current frame to the features of the previous frame. To simplify the discussion, let us first consider only D-frames and S-frames to describe our proposed framework. As introduced in the previous section, the keypoints in D-frames are detected without reference to previous frames (Intra mode). We estimate the keypoints for the subsequent S-frames using all keypoints from the previous frame. The keypoint estimation process for S-frames is represented by *Skip* in Fig. 6.9. The number of frames between consecutive D-frames is the *detection interval* Δ, which is set to a fixed number (e.g., 5 or 20) in our first experiment.

Two video frames have a relationship that can be locally described by a geometric transformation, e.g., an affine transformation or a perspective transformation. In our experiments, we assume that the relationship between two consecutive frames can be described by an affine transformation and transmit the corresponding affine transform matrix. The locations, scales, and orientations of the keypoints in the current frame can then be estimated using this transform and the keypoints from the previous frame. The features from the previous frame are stored in the *Previous features buffer*. The block *Feature matching* is used to calculate the affine transform matrix. In the literature, several affine transform matrices were determined in [WSG05] for subregions of a video frame with the goal of improving the conventional rate-distortion performance in video coding. We can also use several affine transform matrices in the case that the video contains multiple objects of interest that are moving independently. In the Stanford MAR dataset, all of the test videos contain objects of interest that are moving in the same direction. Thus, similar to [MCT$^+$14], we use only one common transform matrix in this paper. However, multiple matrices can be used for complex video scenes to improve performance. Next, we will explain the blocks labeled *Affine transform matrix T* and *Keypoint estimation* in Fig. 6.9 in detail.

6.2.3.2 Location, scale, and orientation estimation

After obtaining the affine transform matrix, we estimate the keypoints in the current frame using the keypoints from the previous frame. The estimated location can be easily calculated as follows:

$$\begin{aligned}\begin{bmatrix}\hat{x}\\ \hat{y}\\ 1\end{bmatrix} &= T\cdot\begin{bmatrix}x\\ y\\ 1\end{bmatrix}\\ &= \begin{bmatrix}a & b & t_x\\ c & d & t_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ 1\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & t_x\\ 0 & 1 & t_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}a & b & 0\\ c & d & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\ y\\ 1\end{bmatrix}\end{aligned} \tag{6.10}$$

where (x, y) is the keypoint location in frame $f - 1$ and $(\hat{x}, \hat{y})$ is the estimated location in frame f. Equation (6.10) represents the affine transformation in homogeneous coordinates. Here, T is the affine transform matrix, and we first decompose the transform matrix into two component matrices as shown in Equation (6.10). The second matrix can be further decomposed as follows:

$$\begin{aligned}A &= \begin{bmatrix}a & b\\ c & d\end{bmatrix}\\ &= \begin{bmatrix}r_1 & 0\\ 0 & r_2\end{bmatrix}\begin{bmatrix}1 & 0\\ q & 1\end{bmatrix}\begin{bmatrix}cos(\phi) & sin(\phi)\\ -sin(\phi) & cos(\phi)\end{bmatrix}\end{aligned} \tag{6.11}$$

where the matrix on the left represents the scaling, the middle matrix represents the shearing, and the matrix on the right represents the rotation. Note that this decomposition is not unique. Here, r_1 and r_2 are the scaling factors in two directions, q is the shearing factor, and ϕ is the clockwise rotation angle. By solving the four linear equations represented by Equation (6.11), we obtain the following:

$$\begin{aligned}r_1 &= \sqrt{a^2 + b^2}\\ r_2 &= \frac{ad - bc}{\sqrt{a^2 + b^2}}\\ q &= \frac{ac + bd}{ad - bc}\\ \phi &= atan(b, a)\end{aligned} \tag{6.12}$$

These formulae are graphically illustrated in Fig. 6.11. After the affine transformation, the circular keypoint in the first frame has taken on an elliptic shape in the second frame. Its area is determined by the scaling factors r_1 and r_2 and the shearing factor q in Equation (6.11). From Fig. 6.11, we can see that the shearing and rotation transformations do not change the area, whereas r_1 and r_2 do affect the area. Because the keypoint of a SIFT feature has a circular shape, we assume that the elliptical area of an estimated keypoint should be the same

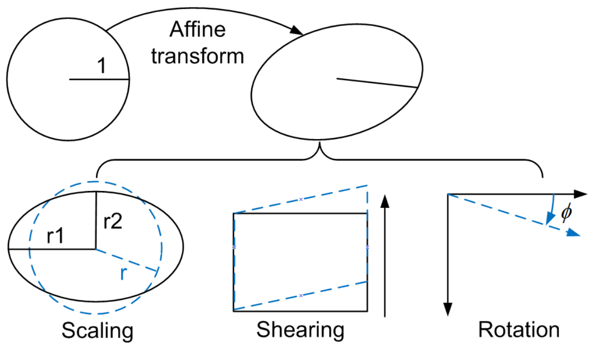

Figure 6.11: Scaling, shearing, and rotation transformations derived from matrix A.

as the area of the corresponding keypoint (the dashed blue circle with radius r) in the current frame that would be detected using the conventional detection process. From the formulae for calculating the areas of a circle and an ellipse, the following scaling factor is obtained:

$$\begin{aligned} \pi r^2 &= \pi r_1 r_2 \\ r &= \sqrt{r_1 r_2} \\ s &= \frac{r}{1} = r \end{aligned} \tag{6.13}$$

To summarize, the location $\hat{l}^f$ (a vector of $\hat{x}$ and $\hat{y}$) of a possible keypoint $\hat{k}^f$ in the current frame f can be estimated by applying T in Equation (6.10) to the location l^{f-1} (a vector of x and y) of a keypoint k^{f-1} in the previous frame $f-1$. This operation is expressed as follows:

$$\begin{bmatrix} \hat{l}^f \\ 1 \end{bmatrix} = T \cdot \begin{bmatrix} l^{f-1} \\ 1 \end{bmatrix} \tag{6.14}$$

The scale of the keypoint is estimated by multiplying the scaling factor s to the scale σ as follows:

$$\hat{\sigma}^f = s \cdot \sigma^{f-1} = \sqrt{r_1 r_2} \cdot \sigma^{f-1} \tag{6.15}$$

Furthermore, the orientation is estimated by rotating the orientation θ^f as follows:

$$\hat{\theta}^f = \theta^{f-1} - \phi \tag{6.16}$$

Thus, the block labeled *Keypoint estimation* in Fig. 6.9 can be detailed as shown in Fig. 6.12.

We perform an initial experiment to justify our method of estimating the location, scale, and orientation of a keypoint. In this experiment, we use the unquantized keypoints from the D-frames and the unquantized affine transform matrix T. The descriptors are calculated from the uncompressed video frames using the estimated keypoints in the S-frames to demonstrate the accuracy of the estimation. As described in Section 3.1, eight videos, each containing a single static object, are used; the SIFT descriptors from the video frames are compared

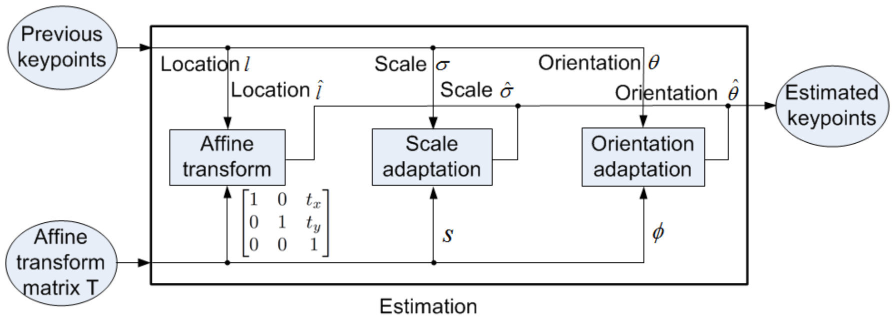

Figure 6.12: The location, scale and orientation estimation.

Table 6.2: Evaluation of location, scale, and orientation estimation for keypoints.

Adaptation	Avg. # matches	
	Δ=5	Δ=20
loc. only (Equation (6.14))	58.33	49.20
loc.+sc. (Equation (6.14) + Equation (6.15))	58.55	50.98
loc.+orient. (Equation (6.14) + Equation (6.16))	58.96	57.64
loc.+sc.+orient. (Equation (6.14) to Equation (6.16))	59.19	59.13
Independent detection	59.13	

with those extracted from the corresponding reference images, and the average numbers of matching features are recorded.

We perform this experiment to verify the performance of the blocks labeled *Affine transform*, *Scale adaptation*, and *Orientation adaptation* in the diagram shown in Fig. 6.12. In Table 6.2, *loc. only* refers to the case in which only the keypoint locations are estimated using Equation (6.14), whereas the scales and orientations are simply estimated as the scales and orientations of the previous keypoints, i.e., $\hat{\sigma}^f = \sigma^{f-1}$ and $\hat{\theta}^f = \theta^{f-1}$. The notation *loc. + sc.* refers to the case in which the scales are also modified using a scaling factor, as in Equation (6.15). The notation *loc. + sc. + orient.* indicates the case in which the locations, scales, and orientations are all appropriately transformed. In addition, the results of conventional keypoint detection and descriptor calculation for each separate frame are provided for comparison (*Independent detection*). In this case, all frames are D-frames, and the keypoints are detected independently for each frame. Table 6.2 shows that the matching performance is improved by applying Equation (6.15) and (6.16), indicating that our estimation method is effective. Note that for a detection interval of Δ=5, *loc. only* also achieves a high number of matches, and the improvement achieved for *loc. + sc. + orient.* seems quite small because of the small detection interval and because the test videos each contain only a single object. However, when the detection interval is large (Δ=20) or the video content is rapidly changing, *loc. only* and *loc. + sc.* can easily fail to produce correct descriptors. Accordingly, we can see that *loc. + sc. + orient.* achieves much better performance in the third column of Table 6.2.

In addition, two possible reasons that the keypoints estimated using Equations (6.14) to (6.16) slightly outperform the independently detected keypoints are that the RANSAC computation yields inconsistent numbers of matching features and that keypoints may therefore disappear or reappear in consecutive frames as a result of the feature detection and feature selection processes. The estimated keypoints, which are absent in conventional feature detection, are still sufficient to extract effective descriptors. As a result, the average number of matching features obtained from the estimated keypoints could be slightly higher than that obtained using the independent detection method. In the following experiments, we use the estimated keypoints (location, scale, and orientation) to extract descriptors.

6.2.3.3 Affine transform matrix quantization

The affine transform matrix contains real-valued numbers; therefore, we must perform lossy encoding for this matrix. The affine transform parameters are quantized via scalar quantization in [WSG05]. The authors of [MCT+14] encode the affine transform matrix T using differential keypoint location coding. They uniformly quantize the parameters a, b, c, d, t_x, and t_y in Equation (6.10) using 7 bits, resulting in 42 bits in total for the matrix T. By contrast, we propose to quantize the parameters r_1, r_2, q, and ϕ in Equation (6.12). The associated keypoints in consecutive frames have similar scales and small differences in orientation. We find that the parameters r_1 and r_2 lie within the range [0.9, 1.1], the parameter q remains within the range [-0.05, 0.05], and ϕ is within the range [-0.15, 0.15]. This is because the affine transform matrix T is calculated between two consecutive frames. To fairly compare our method with the quantization method proposed in [MCT+14], we also assign 42 bits to the matrix T. For r_1 and r_2, we use 6-bit quantizers, and for q, we assign 7 bits. We observed in our experiment that the quantization of ϕ strongly affects the matching performance; therefore, 9 bits are assigned to this parameter. For t_x and t_y, we assign the same number of bits as in [MCT+14] (namely, 7) to enable the exclusive comparison of the quantization methods for the matrix A in Equation (6.11). Subsequently, to improve the matching performance, we increase the allotted space for encoding the matrix T to 48 bits, i.e., 7, 7, 7, 9, 9, and 9 bits for r_1, r_2, q, ϕ, t_x, and t_y, respectively, using uniform quantization. The quantized $\overline{T}$ can be obtained from the quantized parameters in Equation (6.11). We use a detection interval of Δ=5 in our experiments for illustration, and the results are shown in Table 6.3. As can be seen, we achieve slightly better results than the method in [MCT+14]. Our result for 48 bits approaches the performance achieved using the unquantized T. In the following experiments, we use the quantized $\overline{T}$ in place of the uncompressed T, using 48 bits to encode the affine transform matrix.

Table 6.3: Comparison of different quantization methods for the affine transform matrix T

Quantization	Avg. # matches
Uncompressed T	59.19 (Table 6.2, Δ=5)
Method in [MCT$^+$14] (42 bits)	59.00
Our method (42 bits)	59.06
Our method (48 bits)	59.13

6.2.3.4 Adaptive detection interval

In the previous section, we presented experiments performed using a fixed detection interval Δ. However, a fixed detection interval will not be adequate when objects are entering or leaving a scene because the descriptors extracted from certain of the estimated keypoints will become spurious. Additionally, when the object of interest does not change across a large number of frames, a fixed detection interval may result in a new set of conventionally detected keypoints being sent sooner than is necessary. To address these issues, we propose to use an adaptive detection interval. Specifically, we will insert a D-frame or a U-frame when the keypoints from the previous D- or U-frame are insufficient for feature extraction in the current frame.

Adding D-frames or U-frames adaptively

The proposed process for determining the frame type is as follows. For a new frame, we first extract the descriptors for the estimated keypoints and compare them with the features from the previous D- or U-frame.

• If the affine transform matrix cannot be calculated or is incorrect, this indicates that a new scene has begun because the estimated keypoints cannot produce correct descriptors. Therefore, we specify a D-frame for keypoint encoding.

• If the number of matches is greater than a certain threshold (e.g., $\epsilon = 80\%$) with respect to the number of features in the previous D- or U-frame, then most of the estimated keypoints can be considered effective for the current frame. Therefore, an S-frame is specified. Note that the threshold ϵ significantly affects the determination of S-frames and the bitrate of the side information. For the test dataset, the selected value is reasonable; however, it can be adjusted for other datasets or applications.

• If several of the keypoints are still valid, although the object of interest is leaving the current scene, another object is entering the scene, or many keypoints deviate from the actual keypoints after a large number of frames, then we designate the current frame as a U-frame.

The reason that we compare the features of the current frame with the features of the previous D- or U-frame is that the initial keypoints propagated to the intervening frames originate from this previous D- or U-frame. After determining the frame type, we calculate the affine transform matrix T between the previous frame and the current frame if the frame is designated as an S- or U-frame. Note that we do not use the affine transform matrix between the current frame and the previous D- or U-frame because the compression of this matrix requires a higher bitrate. An individual T for each S- or U-frame is then quantized and transmitted as side information. Transmitting T is not required for D-frames.

Comparison of adaptive and fixed detection intervals

We compare the adaptive-interval and fixed-interval schemes. In this experiment, the previous settings are used: a fixed $\Delta = 5$, quantization of the affine transform matrix using 48 bits, and conventional feature detection for D- or U-frames. Note that we still use the original (unquantized) keypoints for the D- and U-frames because the objective here is to evaluate the performance of our adaptive detection interval scheme for uncompressed video frames. In addition, we add an *independent detection* scheme to detect features for each frame individually in a conventional manner. From Fig. 6.13, we can see that the performance of the fixed-interval scheme is similar to that of the independent detection scheme because the interval Δ is quite short. The green curve represents the performance of the adaptive-interval scheme, and the red dots represent the D- and U-frames. Interestingly, the keypoints from the D-frames strongly affect the descriptor calculation and matching in the consecutive S-frames. As a result, the adaptive-interval scheme sometimes outperforms the independent detection scheme (e.g., at frame 70), but it is inferior, for example, at frame 80.

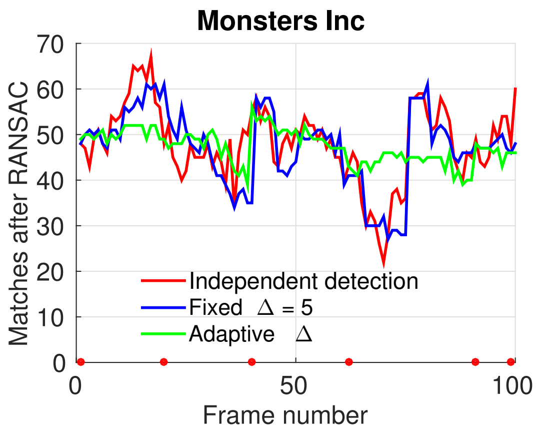

Figure 6.13: Matching performance comparison among independent detection, a fixed detection interval ($\Delta = 5$), and an adaptive detection interval (example video: *Monsters Inc.*). The red dots indicate the frames at which conventional detection is applied in the adaptive-interval scheme.

Table 6.4 shows the average number of matches and the average number of D- or U-frames over all videos. The fixed-interval scheme achieves similar matching performance to that of the independent detection scheme. In the dataset we are using, each video contains only one object and no scene changes. Therefore, the performance of the fixed-interval scheme does not suffer. The performance of the adaptive-interval scheme is somewhat reduced because of the existence of more S-frames; however, the number of D- and U-frames is significantly reduced. This will result in a significant reduction in the amount of the side information, as shown in Section 6.2.4.3. Moreover, because of the manner in which we insert D- or U-frames, our adaptive-detection-interval scheme is still suitable for a rapidly changing scene. We will discuss experiments using videos containing scene changes in Section 6.2.5.2.

Table 6.4: The average numbers of matches and of D- and U-frames over all videos.

Schemes	Avg. # matches	Avg. # D/U-frame
Indep. detect.	59.13 (Table 6.2)	100
Fixed interval (Δ=5)	59.13 (Table 6.3)	20
Adap. interval ($\epsilon = 80\%$)	57.95	4.63

6.2.3.5 Switching off keypoint encoding and transmission

When the camera or one or more objects are moving quickly, the number of features matched in the next frame could fall below the threshold $\epsilon = 80\%$. In this case, we need to designate a U-frame. Thus, the bitrate of the keypoints will be significantly increased by the necessity of specifying many U-frames for rapidly moving scenes. This situation can be mitigated by reducing the threshold ϵ; however, because this threshold is used to indicate the percentage of well-estimated keypoints, its value should not be set too low. To reduce the bitrate of the side information, as shown in Fig. 6.8, we add N-frames for quick scene changes. This is motivated by the observation that the video frames for a fast-moving scene are less relevant to the computer vision algorithm run at the server.

The method for determining an N-frame is as follows. Once we have a D- or U-frame, we assume that a new scene or an updated scene is coming. However, if the scene is too short, we may need to add many D- or U-frames, which would introduce a significant increase in bitrate. Therefore, we need to check the length of the scene corresponding to the current D- or U-frame. If the subsequent N_s frames are not S-frames, then the current D- or U-frame is changed to an N-frame. This indicates that the scene corresponding to the current frame does not remain stable across N_s+1 frames. We skip the keypoint encoding for this frame, and the next frame is assumed to be a new D-frame. This process is then performed again for the new D-frame. We switch off the keypoint encoding and transmission for such N-frames, and at the server, the features for these frames are directly extracted from the decoded frames. We will discuss the N-frames selected in a retrieval experiment using *Multiple Objects* video sequences in Section 6.2.5.2.

6.2.4 Keypoint encoding and transmission

In the previous sections, we presented several experiments to justify the feasibility of using predicted keypoints, quantizing the affine transform matrix, and using an adaptive detection interval. In addition, we proposed switching off the keypoint encoding and transmission when the scene is moving quickly to reduce the number of D- or U-frames. As a result, the number of keypoints that must be encoded is significantly reduced. Next, we will describe the keypoint encoding approaches used for the different types of frames. First, we use 2 bits to indicate the frame type for each frame. For D-frames, similar to the approach used in our previous study of keypoint encoding for still images, we quantize the keypoint locations into integer values and use sum-based context-based arithmetic coding to encode them. We use 12 bits for scale and orientation encoding. In our experiments, this keypoint encoding method is referred to as the *Intra mode*. For S-frames, we use the estimated keypoints directly and send only the quantized affine transform matrix, which requires 48 bits. This procedure is denoted by *Skip mode* in Fig. 6.9. For U-frames, we use three modes to encode the keypoints: the *Intra*, *Skip* and *Inter* modes. The Inter mode is identical to the differential keypoint encoding mode shown in Fig. 6.9. If this mode is selected, then the differences between the scales, locations, and orientations of a matched pair are encoded and transmitted. Note that for an N-frame, the encoding and transmission of the keypoints and the affine transform matrix T are skipped.

6.2.4.1 Intra mode keypoint encoding

Keypoints are encoded using the previously observed quantization levels for locations, scales, and orientations (see Section 6.1). If all encoded keypoints $\tilde{k}_i$ are sent as side information, the extracted descriptors are expressed as follows:

$$\tilde{d}_i = \Psi(\tilde{k}_i|\overline{I}),\ with\ \tilde{k}_i = Dec(Enc(k_i)) \tag{6.17}$$

where $Enc(\cdot)$ and $Dec(\cdot)$ denote the encoding and decoding, respectively, of the keypoints.

6.2.4.2 Keypoint encoding for U-frames

For U-frames, we use three modes to encode the keypoints: the *Intra*, *Skip* and *Inter* modes. The Inter mode is the differential keypoint encoding mode shown in Fig. 6.9. The differences between the scales, locations and orientations of a matched pair are encoded and transmitted. These steps are described as follows.

Step 1. Find the matches between two consecutive frames using the NNDR matching strategy. We denote the matched keypoints in the previous frame by $\mathbf{K^a} = \{k_1^a, ..., k_i^a, ..., k_N^a\}$ and the matched keypoints in the current frame by $\mathbf{K^b} = \{k_1^b, ..., k_i^b, ..., k_N^b\}$.

Step 2. Use the quantized affine transform matrix to estimate the locations (6.14), scales (6.15), and orientations (6.16) of the keypoints $\hat{\mathbf{K}}^{\mathbf{b}}$ in the current frame from the keypoints $\mathbf{K^a}$ in the previous frame. We denote the keypoint estimation by $\mathbf{K^a} \mapsto \hat{\mathbf{K}}^{\mathbf{b}} = \{\hat{k}_1^b, ..., \hat{k}_i^b, ..., \hat{k}_N^b\}$.

Step 3. Calculate the differences to obtain $\Delta l_i = l_i^b - \hat{l}_i^b$, $\Delta\sigma_i = \sigma_i^b - \hat{\sigma}_i^b$ and $\Delta\theta_i = \theta_i^b - \hat{\theta}_i^b$ for each pair $\{k_i^b, \hat{k}_i^b\}$ of $\{\mathbf{K^b}, \hat{\mathbf{K}}^{\mathbf{b}}\}$.

Step 4. Remove incorrect matches if the differences are too large, i.e., $\Delta l_i > 16$, $abs(\Delta\sigma_i)/\hat{\sigma}_i^b > 0.3$, or $abs(Q(\Delta\theta_i)) > 4$, where $Q(\cdot)$ denotes the index difference derived from (6.5). As a result, correct matches have the following properties: Δl_i lies on the interval [-16, 16]; the index of $\Delta\sigma_i/\hat{\sigma}_i^b$ lies on the interval [0, 5], where it is quantized into five levels; and $Q(\Delta\theta_i)$ lies on the interval [-4, 4].

Step 5: a) The Skip mode is used if the matched keypoints satisfy the following three conditions: 1) $\Delta l_i <= 1$; 2) the index of $\Delta\sigma_i/\hat{\sigma}_i^b$ is equal to 2, which means that the scale has not changed; and 3) $Q(\Delta\theta_i) = 0$.

b) The Inter mode is used for the differential coding of any other correctly matched keypoints. The differential values are encoded via arithmetic coding.

c) The Intra mode is employed for non-matched features and incorrectly matched features, which are treated as features corresponding to new scene content. These keypoints are encoded in the same manner as the keypoints in D-frames.

Note that the bitrate for keypoint encoding is determined by the threshold ϵ that is used to determine the frame types and by the quantization factors that are used for each quantity in the Intra and Inter modes.

6.2.4.3 Bitrate comparison

We encode the keypoints using three different schemes and compare the resulting bitrates. First, all frames are treated as D-frames, which means that all features are independently extracted using the decoded keypoints. Second, only the first frame is considered to be a D-frame, with the following frames being U-frames. Third, S-frames and an adaptive detection interval are added. We compare these schemes to demonstrate the potential bitrate reduction offered by the proposed approach. Fig. 6.14 shows the results for the three different schemes. The introduction of S-frames and an adaptive detection interval leads to a significant bitrate reduction (by a factor of 18 compared with D-frames only), but the matching performance is not significantly affected.

Fig. 6.15 shows the number of keypoint bits for each frame of the video sequence *Wang Book*. The adaptive detection interval reduces the number of D- or U-frames by exploiting the keypoints that remain coherent across frames. For an S-frame, we use 48 bits for the affine transform matrix and 2 bits to indicate the frame type. For U-frames, many keypoints are

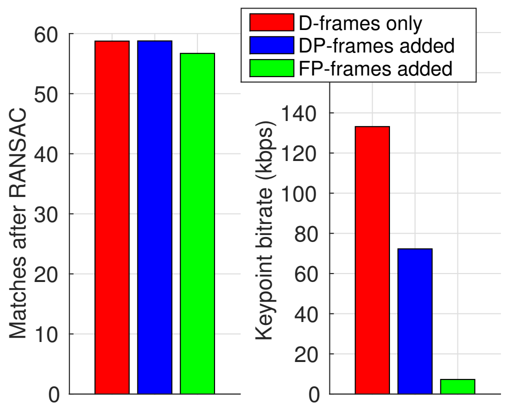

Figure 6.14: Left: matching performance comparison for the *D-frames only* scheme, the *U-frames added* scheme and the *S-frames added* scheme over all videos. Right: bitrate of the side information.

encoded in the Intra or Inter mode, therefore requiring a large bitrate. However, incorrect keypoints or deviated keypoints are corrected by adding U-frames.

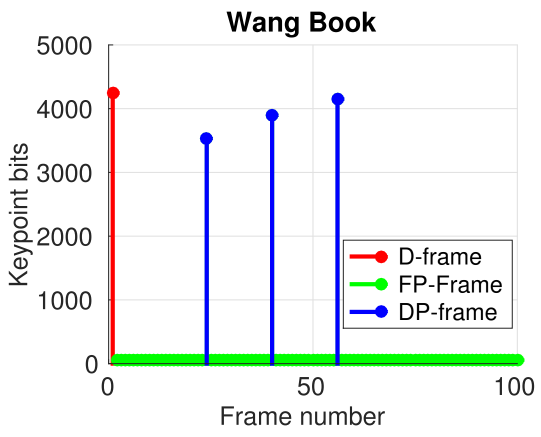

Figure 6.15: Number of keypoint bits used for each frame (example video: *Wang Book*).

6.2.5 Experimental results

In the previous section, we described how to encode the keypoints for the different types of frames. Thus far, the encoded keypoints have been used to extract SIFT descriptors from the original video frames. Here, we add the encoded keypoints as side information for compressed videos that are encoded with different QP values. The pairwise matching and retrieval performances are compared with those for standard H.265/HEVC-encoded videos.

6.2.5.1 Pairwise matching results for videos with different QP values

First, we perform pairwise matching for the eight videos presented in Section 3.1. The blue line in Fig. 6.16 shows the final matching performance of the proposed approach as a function of the total bitrate used for the compressed video plus the keypoint side information. The number of matches for a given bitrate budget is significantly improved. In general, a 5$\times$ bitrate reduction is achieved compared with conventional H.264/AVC encoding. This result is better than those for the patch-encoding approaches presented in [MTC$^+$13] (2.5$\times$) and [MCT$^+$14] (4$\times$). Note that in the cited studies, the bitrate reduction was also calculated in comparison with the performance of H.264/AVC encoding; therefore, the comparison is quantitatively fair. Unlike these patch-encoding approaches, we provide standard-compatible videos in addition to the locations, scales, and orientations of the keypoints for a geometric consistency check. The keypoints of a correctly matched image pair should have correlated locations, scales and orientations. Thus, the keypoint information is valuable for eliminating outliers and increasing precision.

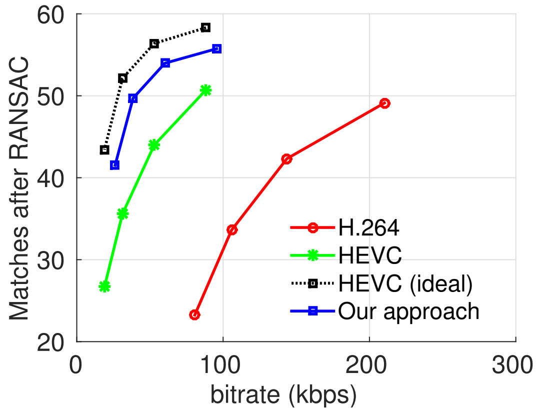

Figure 6.16: Matching performance comparison for various approaches (the blue line represents the proposed approach). The bitrate includes both the encoded keypoints and the H.265/HEVC-encoded videos.

6.2.5.2 Retrieval results

In our previous experiments, we compared the number of preserved features using pairwise matching. Notably, in content-based image retrieval systems, performance typically improves with an increasing number of preserved features. To evaluate the retrieval performance of our proposed scheme, we use video sequences in the Stanford MAR dataset that show multiple objects. Each *Multiple Objects* video consists of 200 frames and contains three different objects of interest. The first two *Multiple Objects* videos are used in our retrieval experiments. Similar to the retrieval experiment in Section 5.2.6, excluding the fast spatial matching component, we use a previously proposed image retrieval system [PCI+07]. The MIRFLICKR-25000 [HTL10] database and the 23 reference images from the Stanford MAR dataset are used as the training dataset. Similar to [MTC+13], we extract up to 300 SIFT descriptors for each image in the database and train one million visual words from these descriptors. For the test frames, we extract 200 SIFT features and pass them to the retrieval engine. After obtaining a shortlist of candidate matching images from the retrieval system, we run RANSAC on the top 100 images in the shortlist to reorder these retrieved images for improved precision. We run the retrieval for each frame of the *Multiple Objects* videos. As noted in a previous study [MCT+14], this operation is redundant because the retrieval results for consecutive frames are closely related. However, the objective of this experiment is to examine the performance of different approaches in a scenario wherein an object of interest is leaving or entering the scene. Note that in this experiment, the threshold ϵ is set to 80% (Section 6.2.3.4), N-frames are used, and N_s (Section 6.2.3.5) is set to 4 for these video sequences. We encode the videos using H.265/HEVC. Note that we use only a QP value of 46 (Section 6.2.1.1). Three approaches are compared, i.e., feature extraction from the uncompressed video frames, feature extraction from the compressed video frames, and feature extraction using the encoded keypoints.

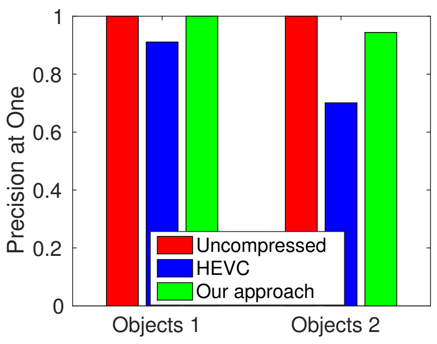

Figure 6.17: PAO values for two *Multiple objects* video sequences.

Precision at One

Similar to [MCT+14], we first plot the Precision at One (PAO). The PAO is defined as the ratio between the number of correctly retrieved images in the top position and the total number of frames used for retrieval. Note that not all 200 frames are used to calculate the PAO. Only the frames whose locations are tagged in the ground-truth files are used. Fig. 6.17 shows the PAO values for the three tested approaches for the *Multiple Object* videos. The proposed approach based on encoded keypoints yields a significant improvement in terms of the PAO. The bitrates of the two encoded video sequences are 37.08 kbps and 23.44 kbps, and the bitrates for the encoded keypoints are 7.55 kbps and 9.11 kbps, respectively. As seen from the figure, adding the encoded keypoints as side information significantly improves the retrieval performance. In our experiment, we find that the proposed approach offers superior performance compared with videos encoded using a smaller QP value at the same bitrate.

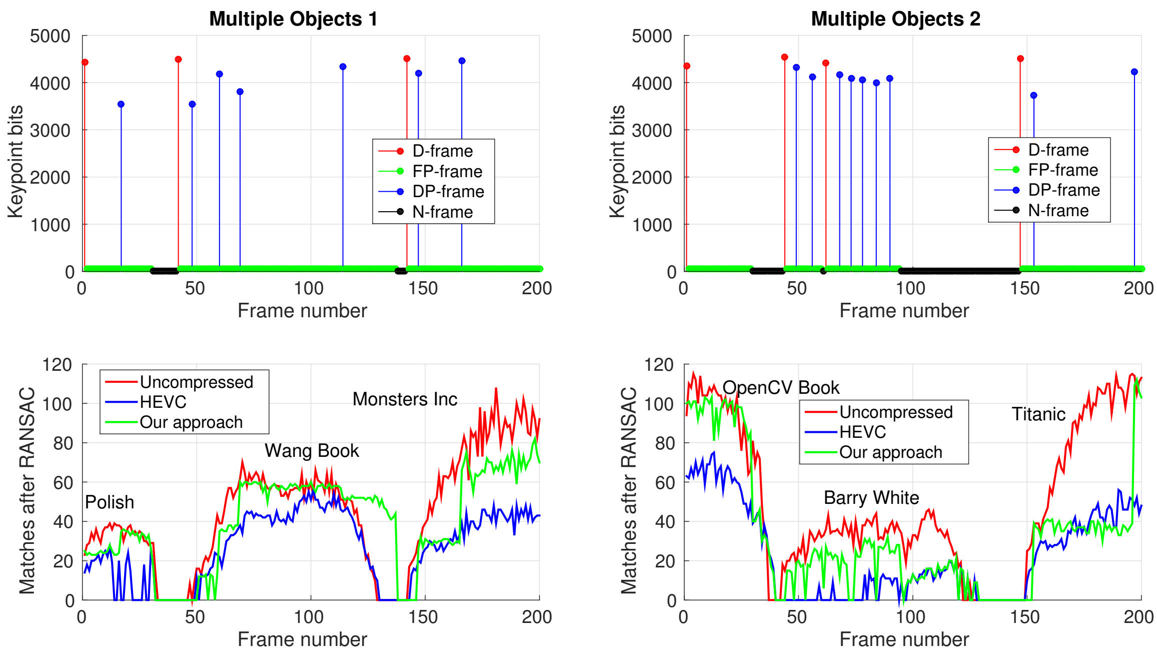

Figure 6.18: The upper plots show the keypoint bits for individual video frames. The lower plots show the pairwise matching performance between the video frame and the top retrieved image.

Number of matches in top retrieved images

The upper plots of Fig. 6.18 show the frame types and numbers of bits for individual video frames. The red, green, blue, and black dots represent the keypoint bits for D-, S-, U-, and N-frames, respectively. The keypoint bits for D- and U-frames vary for different video frames, whereas each S-frame requires 50 bits and each N-frame requires 2 bits, thereby resulting in a large bitrate reduction. The lower plots of Fig. 6.18 show the number of matches achieved using the pairwise matching scheme only when the system retrieves the correct image in the top position. A value of zero indicates that the retrieved image is not in the top position or that no matching image is identified for the video frame. In general, the proposed approach yields an increased number of matches. In the lower right plot, because of the significant amount of glare on the *Barry White* CD cover, many spurious features are extracted across consecutive video frames. These keypoints cannot be propagated to consecutive video frames. Therefore, many frames are selected as N-frames, for which keypoint encoding and transmission are skipped (e.g., video frames 95 to 125), resulting in impaired matching performance for *Barry White*. The results reported in [MCT$^+$14] show similar performance impairments. In addition, the glare on the CD cover causes the selection of a greater number of D- or U-frames (e.g., video frames 49 to 90) when encoding the keypoints. There is a drop between frame 155 and frame 195 compared with the uncompressed video, which occurs because the frames during this portion of the video are all of the S-frame type. In Fig. 6.13, the number of matches in the current S-frame is closely related to the number of matches in the previous D- or U-frame. From the drop observed in the figure, we can see that the same is also true here. The performance in this portion of the video could be improved by adding a U-frame (i.e., tuning the parameters).

The top detected images for the three approaches differ for certain video frames. For example, as shown in the lower left plot in Fig. 6.18, the proposed approach detects *Wang Book* correctly from frame 130 to frame 137, whereas the other two approaches detect no relevant images. By contrast, the other two approaches detect *Monsters Inc.* from frame 144 to frame 146, but the proposed approach fails to detect any relevant image. Fig. 6.19 presents two example frames to illustrate the results. Note that these frames are not included in the calculation of the PAO because the transition sections of the videos are not included in the ground-truth files in the Stanford MAR dataset.

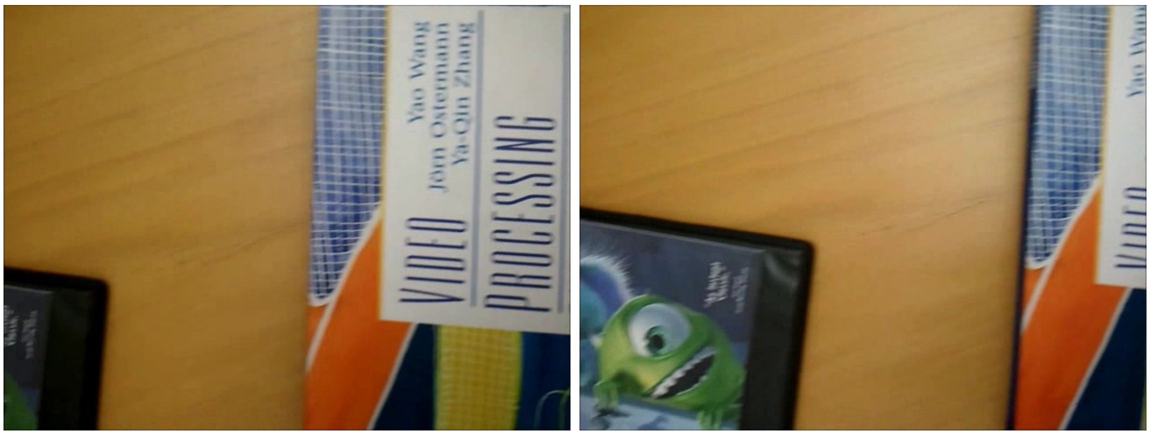

Figure 6.19: Transition frames from *Wang Book* to *Monsters Inc.*. Left: frame 135, where the proposed approach correctly detects *Wang Book* and the other two approaches fail. Right: frame 145, where the proposed approach fails to detect the relevant image and the other two approaches successfully detect *Monsters Inc.*.

Retrieval results for 720p video sequences

The Stanford MAR dataset used in the previous experiments is quite small and simple. Therefore, in this retrieval experiment, we use two 720p format video sequences [TUM] that contain rich features to evaluate our proposed approach: *720p50_mobcal_ter* and *720p5994_stockholm_ter*. Note that four frames of each video are gray frames, which are removed before coding. We displayed the first frame of each video on a monitor and acquired an image of it using a mobile phone. The resulting images therefore deviate considerably from the original as a result of noise, perspective transformation, illumination changes, and so on. The processed images, shown in Fig. 6.20, are used as reference images and integrated into the database used in the previous retrieval experiment to form the training database.

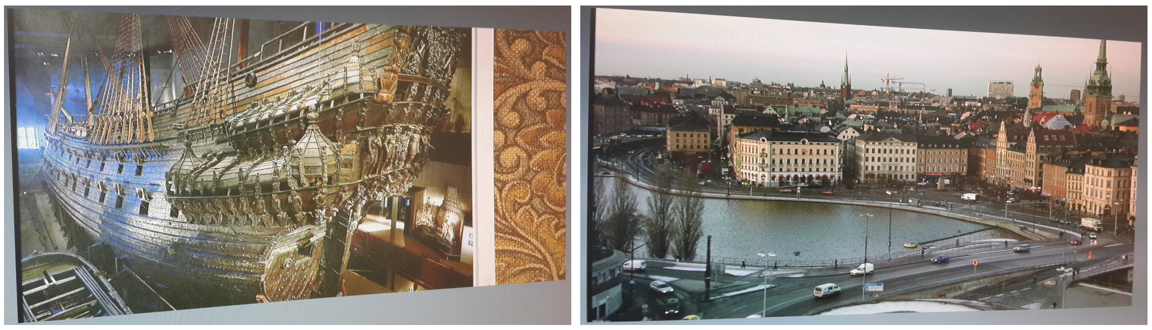

Figure 6.20: Reference images derived from the first frames of each 720p video.

As in the previous experiment, the 720p format videos are encoded using H.265/HEVC with a QP value of 46. The keypoints are extracted from the uncompressed video and encoded as side information. Because of the relatively large size of 720p videos, we extract 300 SIFT features from each frame and pass them to the retrieval engine used previously. The other settings are the same as those used in the previous retrieval experiment. The bitrates of the two encoded video sequences are 64.50 kbps and 41.74 kbps, and the bitrates of the encoded keypoints are 9.20 kbps and 6.16 kbps, respectively. We plot the number of matches between the correctly retrieved image (i.e., the reference image is in the top position) and the video frames for the uncompressed videos, the H.265/HEVC-encoded videos and the videos encoded using our proposed approach in Fig. 6.21 (only the results for the first 250 frames are shown). It is apparent that using the encoded keypoints significantly improves the retrieval performance. Note that the content of the first frame gradually disappears in the following frames, and therefore, the number of matches should decrease through consecutive frames. However, in Fig. 6.21, the number of matches obtained using our scheme increases significantly. As noted before, the number of matches in subsequent S-frames strongly relies on the number of matches in the previous D-frame, as shown in the figure. In addition, we can see that additional frames can be correctly retrieved with the addition of side information, e.g., frames 175 to 200 in the plot on the lower left and frames 95 to 139 in the plot on the lower right. Compared with the results of the previous retrieval experiment, it is more clearly demonstrated here that the predicted keypoints yield correct descriptors calculated based on these transition frames. It should be noted that the matches identified by our proposed scheme significantly outnumber the matches for the uncompressed video. This can be explained as follows. Only 300 features are extracted from each 720p video frame; therefore, they are highly sparse. Then, a few even stronger features are detected from the coming scene, which are identified as the top 300 features. The previous scene is still present in the video content; thus, the predicted keypoints can yield more valid descriptors than the scheme using the uncompressed video because the first frame is the reference image. Considering the resolution of 720p videos, the scenario depicted in Fig. 6.21 can rapidly change if more features are detected, i.e., if the number of features is sufficient or if they are sufficiently distributed across the image. Note that no N-frames are specified among the first 250 frames. Fig. 6.22 shows two video frames for which only our approach returned a correct match in the top position (see the lower plots in Fig. 6.21). We can see that the contents of the query and reference images (see Fig. 6.20) overlap to a large extent. Note that the subsequent video frames still contain many features that match with the relevant reference image; however, the relevant reference frame is not returned in the top position. Therefore, the numbers of matches are not shown in Fig. 6.21.

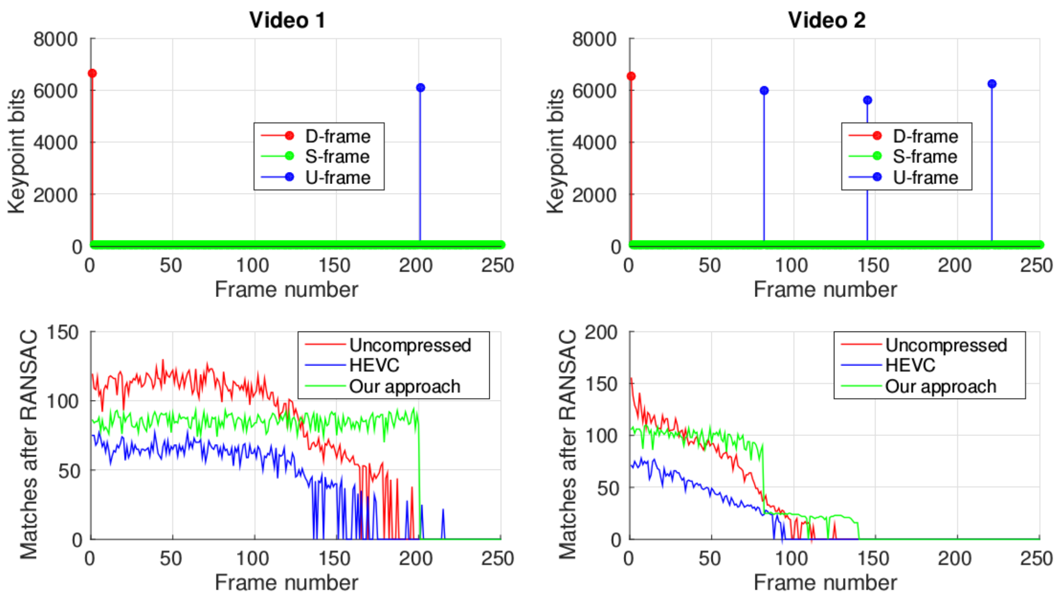

Figure 6.21: The upper and lower plots show the keypoint bits for individual video frames and the pairwise matching performance between the video frame and the top retrieved image, respectively.

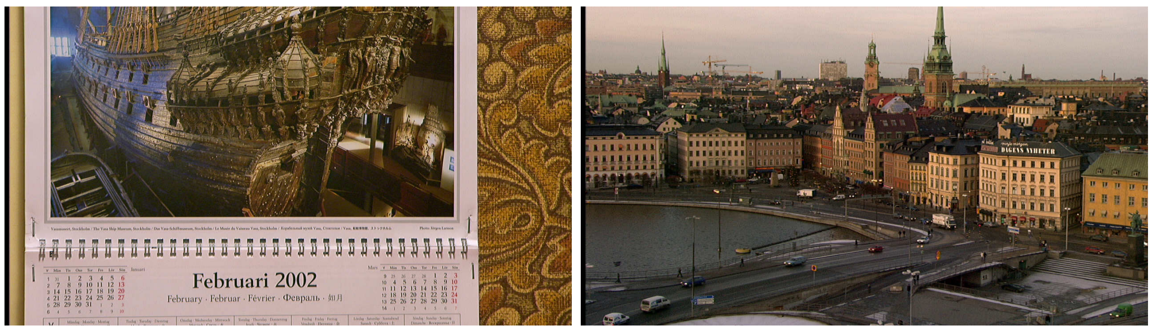

Figure 6.22: Examples of retrieved transition frames. Left: frame 200 of *720p50_mobcal_ter*. Right: frame 139 of *720p5994_stockholm_ter*

6.2.6 Discussion

6.2.6.1 Bitrate reduction for keypoints

We obtained our results using heuristically selected parameters based on the statistics for keypoint encoding. In the Intra mode procedure presented in Section 6.2.4.1, we quantize the locations using a factor of 1 and use 12 bits to encode the scales and orientations. Note that we can modify these parameters to achieve a larger bitrate reduction for D- and U-frames. In the Inter mode procedure presented in Section 6.2.4.2, the differential values of locations, scales, and orientations can be quantized using larger quantizers to further reduce the bitrate for U-frames. The threshold ϵ (Section 6.2.3.4) determines whether the current frame is designated as a U-frame. Therefore, this parameter strongly affects the length of a series of S-frames. A lower ϵ value results in a larger number of S-frames for the current D- or U-frame, which yields a lower bitrate for keypoint encoding. Note that this value should not be too small because overly small values will affect the matching performance. The value of N_s (Section 6.2.3.5) is used to check the number of S-frames associated with the current D- or U-frame. If the length of the window is too short, then the current D- or U-frame is changed to an N-frame to eliminate the encoding and transmission of keypoints in rapidly moving scenes. Furthermore, in our experiments, it is determined that the acceptable N_s value ranges from 3 to 12, depending on the frame rate. In the previous section, we propose removing spurious keypoints and duplicated keypoints (see Section 6.1) to further lower the number of keypoints to be sent to the server. Note that a spurious keypoint for the current frame could be a useful keypoint for subsequent frames; therefore, we do not directly apply this approach to videos. However, removing duplicated keypoints with respect to the keypoints extracted from the compressed video frame is still feasible. To summarize, there is always a trade-off between the matching performance and the bitrate for keypoint encoding. Note that we present only the results obtained using the parameters selected in the previous sections, which appear reasonably effective for improving the matching performance.

6.2.6.2 Reusing previous descriptors

For S-frames or in the Skip mode for U-frames, the SIFT descriptors are calculated again at the server based on the estimated keypoints from the compressed frames in our experiments. The purpose of this recalculation is to verify the correctness of the proposed keypoint estimation approach. However, the descriptor calculation process can be skipped by directly using the previous descriptors at the server, i.e., the descriptors for the S-frames and some descriptors for the U-frames can be reused from the descriptors calculated for previous frames. Thus, the computation time at the server can be reduced. In our experiments, from the pairwise matching results shown in Fig. 6.16 and the retrieval results shown in Fig. 6.18, it can be seen that the extraction of descriptors from predicted keypoints remains effective.

6.2.6.3 Uploading frames and encoded keypoints

In certain applications, when a mobile device captures a video, it is unnecessary to upload all video frames. To address this scenario, [MCT+14] discusses two schemes, both including a *Retrieval State* and a *Pair-wise State*. In the *Retrieval State*, the descriptors extracted from the frame are used to retrieve a relevant image. In the *Pair-wise State*, the descriptors from the current frame are compared with the previously retrieved image using the pairwise matching approach. The authors suggest that better bitrate reduction can be achieved using *On-Device Tracking* [MCT+14]. In our proposed system, we can also perform a process that is similar to *On-Device Tracking* [MCT+14]. When a new object is detected, the frame is HEVC Intra-encoded, and the keypoints are encoded using the Intra mode and sent as side information with the encoded frame. Note that this is similar to our previously proposed approach for images in Section 6.1, which achieves improved performance at low bitrates. The server then performs image retrieval and sends the relevant image back to the mobile device. Then, the descriptors from the following frames are compared with the retrieved image. If no new object is found, then there is no need for data transmission between the mobile device and the server.

6.2.6.4 Computational complexity

It should be noted that the processes of keypoint detection, feature extraction and matching, and keypoint encoding are all based on uncompressed video frames. Therefore, in practice, we can run these processes in parallel with the actual video compression, as shown in Fig. 6.1, to speed up the processing. In addition, to reduce computational complexity, we can replace the descriptor extraction and descriptor matching processes with simpler detectors, descriptors, and matching procedures, e.g., we can heuristically determine which keypoints can be used in the next frame. This keypoint encoding process can be optimized to achieve a reduced computation time compared with the H.265/HEVC encoding time. Moreover, because the keypoint decoding is separate from the decoding of the video, the encoded bitstreams need not be transmitted simultaneously with the video. In certain applications, they can be transmitted later when the communication channel is not busy to improve the matching performance.

6.3 Summary

Although image or video compression artifacts adversely affect the feature-matching performance, the situation can be improved significantly by using the original SIFT keypoints. In the first section, a novel approach for feature-preserving image compression is proposed, where the encoded keypoints are signaled as side information along with the compressed image. Our results show that the proposed approach significantly improves the matching performance for low-bitrate images. In the second section, the idea is applied to video compression, and four different types of frames are introduced for keypoint encoding based on considerations regarding different behaviors in consecutive video frames. Then, we propose methods of predicting keypoints, quantizing the affine transform matrix, adopting an adaptive detection interval, and switching off the keypoint encoding when the scene is moving quickly to reduce the keypoint bitrate. The Intra, Inter and Skip modes of encoding the keypoints are described. Finally, pairwise matching and image retrieval are performed. The results show that the proposed approach achieves improved performance in feature matching at a given rate. Similarly, other types of keypoints (e.g., SURF, MSER, or FAST) can also be encoded for improved feature extraction using the proposed framework. In addition, when more features (e.g., 500 features) must be transmitted, the increase in bitrate incurred by our proposed scheme is much smaller than that of other schemes.

Chapter 7

Conclusions and outlook

7.1 Conclusions

It is shown that the compression artifacts at low bitrates significantly affect feature quality. The combination of the Hessian-Affine detector and the SIFT descriptor is the most robust one under severe compression. Then, several experiments were performed to investigate the properties of features for the sake of designing feature-preserving image and video compression. We examined the sensitivities of features at different scales, keypoints and descriptors, and various detectors to image frequencies. These observations motivated the various schemes proposed in this thesis.

In the scenario that still images are captured using a mobile device and encoded using an image compression standard, e.g., JPEG standard, we first considered the case that the images were already encoded and transmitted to the server. The results show that at the server, the application of deblocking filters to the images can improve the retrieval performance for some features. Then, feature-preserving bit allocation was proposed during image encoding process in the mobile device. Given that small-scale features are more vulnerable, the MBs were categorized into various groups, and greater numbers of bits were assigned to the small-scale features using a novel R-D model. In addition, a novel JPEG quantization table was designed for scale-space-based features considering that the high frequencies are unimportant. The results show that the proposed approaches lead to higher matching performance compared to the default JPEG standard.

The third part of this thesis extends the above-mentioned approaches for images to video compression. Similarly, R-D models were designed to encode videos using the H.264/AVC standard. As a result, greater numbers of features were obtained compared to normally H.264/AVC-encoded videos. In practical video communications, compressed videos should meet the constraints of communication channels for transmission. Therefore, a novel rate control framework was proposed for feature preservation in H.264/AVC video compression. Similarly, the MBs were categorized into different groups based on their importance in feature extraction. The GOP level and the frame level rate allocation schemes were designed, and different quantization parameters were assigned to each group. The experimental results show that our framework achieves the target bitrate, while preserving a greater number of features.

Then, based on the property that keypoint detection is more vulnerable under compression, we proposed encoding the original keypoints of uncompressed images or video frames and sending them along with the compressed images or videos. Consequently, the quality of the feature descriptors calculated at the server using the transmitted keypoints is better. For images, the approach of intra-frame keypoint selection and encoding was proposed. For videos, four different frame types were introduced for keypoint encoding based on the type of the video content, and the Intra, Skip, and Inter modes were used to encode the keypoints of each frame type. This can be treated as an approach of inter-frame keypoint selection and encoding. Therefore, for both images and videos, only a subset of keypoints was required to be transmitted. The improved feature-matching performance and retrieval performance show the effectiveness of this keypoint encoding idea.

The images and videos used in our feature-preserving image and video compression schemes are standard-compatible and can be decoded using any standard decoder. This is advantageous because the images or videos can be watched or stored for future use, flexible feature types can be extracted from them, and their locations, orientations, and scales can be used for geometric verification. Note that the ideas proposed herein can be applied to other features.

7.2 Outlook

The techniques proposed in this thesis address a wide range of questions pertaining to feature-preserving image and video compression. Nevertheless, there remain many research topics to be studied in this field.

Although various approaches have been proposed in this thesis, thus far, we have not examined the ultimate improvement achievable by combining some of these approaches. For example, the keypoint encoding and transmission idea can be combined with the approach of selecting different QP values for different groups to possibly further improve the rate-matching performance.

The images and videos were compressed at low bitrates assuming limited channel bandwidth. However, as network resources grow, the images or videos can be encoded at high quality. Consequently, the feature quality would probably not be affected significantly. For example, the authors in [KO05] coined in this context the term *critical video quality* to determine the smallest video bitrate that would not affect the accuracy of face detection and face tracking algorithms. When reaching the critical video quality, e.g., the proposed keypoint encoding idea would probably be not necessary. Therefore, in practice, one should investigate the critical image or video quality for a specific application and determine whether the proposed approach should be applied at high bitrates.

Regarding the compression standard, JPEG 2000 has not been exploited for feature preservation. JPEG 2000 uses the discrete wavelet transform (DWT) to process a pixel domain image. Specifically, it offers several techniques for region-of-interest coding that are intrinsically suitable for feature-preserving bit allocation. It would be quite interesting to design a scheme that maximizes the feature quality given a certain rate.

In our experiments, only spatial features were investigated. In addition, many spatio-temporal detectors and descriptors have been proposed in the literature. The popular and commonly used features are, for example, 3D-SIFT [SAS07], extended SURF [WTG08], dense trajectories [WKSL11], HOG/HOF [LMSR08], motion boundary histograms [WKSL11], and so on. The extent to which the compression artifacts affect these features should be examined, and the feature-preserving techniques should be designed carefully considering the feature properties. We believe that a few of the proposed approaches would also be applied to these spatio-temporal features.

Last but not least, in this thesis, we presented our works and performed the experiments using MVS as an example application. In other applications, such as human detection and tracking, face recognition and tracking, one probably needs to modify the proposed feature-preserving image and video compression schemes.

Bibliography

Publications by the author

[CANSS13] J. Chao, A. Al-Nuaimi, G. Schroth, and E. Steinbach. Performance Comparison of Various Feature Detector-Descriptor Combinations for Content-Based Image Retrieval with JPEG-encoded Query Images. In *Proc. IEEE International Workshop on Multimedia Signal Processing (MMSP)*, October 2013. [cited at p. 5]

[CCS13] J. Chao, H. Chen, and E. Steinbach. On the Design of a Novel JPEG Quantization Table for Improved Feature Detection Performance. In *Proc. IEEE International Conference on Image Processing (ICIP)*, September 2013. [cited at p. 5, 16]

[CHSS15] J. Chao, R. Huitl, E. Steinbach, and D. Schroeder. A Novel Rate Control Framework for SIFT/SURF Feature Preservation in H.264/AVC Video Compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(6):958–972, 2015. [cited at p. 5]

[CS11] J. Chao and E. Steinbach. Preserving SIFT Features in JPEG-encoded Images. *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 301–304, September 2011. [cited at p. 5, 16]

[CS12] J. Chao and E. Steinbach. SIFT Feature-Preserving Bit Allocation for H.264/AVC Video Compression. In *Proc. IEEE International Conference on Image Processing (ICIP)*, September-October 2012. [cited at p. 5, 16]

[CS16] J. Chao and E. Steinbach. Keypoint Encoding for Improved Feature Extraction from Compressed Video at Low Bitrates. *IEEE Transactions on Multimedia*, 18(1):25–39, January 2016. [cited at p. 5]

[CSX15] J. Chao, E. Steinbach, and L. Xie. Keypoint Encoding and Transmission for Improved Feature Extraction from Compressed Images. In *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, 2015. [cited at p. 5, 16]

General publications

[ABCK06] D. Agrafiotis, D. R. Bull, C. N. Canagarajah, and N. Kamnoonwatana. Multiple Priority Region of Interest Coding with H.264. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 53–56, 2006. [cited at p. 61]

[ABD12] P. F. Alcantarilla, A. Bartoli, and A. J. Davison. KAZE Features. In *Proc. European Conference on Computer Vision (ECCV)*, pages 214–227. Springer, 2012. [cited at p. 8, 34]

[Acu15] Mobile Acuity. `http://www.mobileacuity.com/`, 2015. [cited at p. 2]

[AKB08] M. Agrawal, K. Konolige, and M. R. Blas. CenSurE: Center Surround Extremas for Real-time Feature Detection and Matching. In *Proc. European Conference on Computer Vision (ECCV)*, pages 102–115. Springer, 2008. [cited at p. 8]

[AOV12] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 510–517, June 2012. [cited at p. 13]

[AR11] Y. Avrithis and K. Rapantzikos. The Medial Feature Detector: Stable Regions from Image Boundaries. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 1724–1731, 2011. [cited at p. 8, 34]

[BAC$^+$14] L. Baroffio, J. Ascenso, M. Cesana, A. Redondi, and M. Tagliasacchi. Coding Binary Local Features Extracted from Video Sequences. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 2794–2798, Oct 2014. [cited at p. 15, 92]

[BCR$^+$14] L. Baroffio, M. Cesana, A Redondi, M. Tagliasacchi, and S. Tubaro. Coding Visual Features Extracted From Video Sequences. *IEEE Transactions on Image Processing*, 23(5):2262–2276, May 2014. [cited at p. 15, 92, 93]

[BCR$^+$15] L. Baroffio, M. Cesana, A. Redondi, M. Tagliasacchi, and S. Tubaro. Hybrid Coding of Visual Content and Local Image Features. `http://arxiv.org/abs/1502.07828`, February 2015. [cited at p. 16]

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002. [cited at p. 13]

[BTG06] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded Up Robust Features. In *Proc. European Conference on Computer Vision (ECCV)*, pages 404–417, Graz, Austria, May 2006. Springer. [cited at p. 2, 29, 34, 48]

[Cam15] CamFind. `http://camfindapp.com/`, 2015. [cited at p. 2]

[Cha13] D. M. Chandler. Seven Challenges in Image Quality Assessment: Past, Present, and Future Research. *ISRN Signal Processing*, 2013, 2013. [cited at p. 17]

[CHC$^+$10] J.-C. Chiang, C.-S. Hsieh, G. Chang, F.-D. Jou, and W.-N. Lie. Region-of-Interest Based Rate Control Scheme with Flexible Quality on Demand. In *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, pages 238–242, 2010. [cited at p. 61]

[CLSF10] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *Proc. European Conference on Computer Vision (ECCV)*, pages 778–792, Crete, Greece, October 2010. Springer. [cited at p. 13, 19]

[CMT$^+$10] V. Chandrasekhar, M. Makar, G. Takacs, D. Chen, S. S. Tsai, N. Cheung, R. Grzeszczuk, Y. Reznik, and B. Girod. Survey of SIFT Compression Schemes. In *Proc. International Workshop on Mobile Multimedia Processing (WMMP)*, Istanbul, Turkey, August 2010. [cited at p. 15]

[Cor15] Cortexica. findsimilar. `http://www.cortexica.com/`, 2015. [cited at p. 2]

[CSRK11] S. Chikkerur, V. Sundaram, M. Reisslein, and L. J. Karam. Objective Video Quality Assessment Methods: A Classification, Review, and Performance Comparison. *IEEE Transactions on Broadcasting*, 57(2):165–182, 2011. [cited at p. 17]

[CTC$^+$09] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, R. Grzeszczuk, and B. Girod. CHoG: Compressed Histogram of Gradients. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, Miami, Florida, USA, June 2009. [cited at p. 15]

[CWL99] L. Chang, C. Wang, and S. Lee. Designing JPEG Quantization Tables Based on Human Visual System. In *Proc. IEEE International Conference on Image Processing (ICIP)*, pages 376–380, 1999. [cited at p. 47]

[CWM04] O. Chum, T. Werner, and J. Matas. Epipolar Geometry Estimation Via RANSAC Benefits From the Oriented Epipolar Constraint. In *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, Cambridge, U.K., August 2004. [cited at p. 63]

[DAP11] A. L. Dahl, H. Aanæs, and K. S. Pedersen. Finding the Best Feature Detector-Descriptor Combination. In *Proc. IEEE International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 318–325, 2011. [cited at p. 13]

[DLC$^+$12] L. Duan, X. Liu, J. Chen, T. Huang, and W. Gao. Optimizing JPEG Quantization Table for Low Bit Rate Mobile Visual Search. In *Proc. IEEE Visual Communications and Image Processing (VCIP)*, pages 1–6, San Diego, CA, USA, November 2012. [cited at p. vi, 16, 47, 50, 51]

[FB81] M. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981. [cited at p. 20, 63]

[FKE07] A. Foi, V. Katkovnik, and K. Egiazarian. Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images. *IEEE Transactions on Image Processing*, pages 1395–1411, May 2007. [cited at p. 37]

[FWH12] B. Fan, F. Wu, and Z. Hu. Rotationally Invariant Descriptors Using Intensity Order Pooling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):2031–2045, October 2012. [cited at p. 13, 32, 35]

[GCGR11] B. Girod, V. Chandrasekhar, R. Grzeszczuk, and Y. A. Reznik. Mobile Visual Search: Architectures, Technologies, and the Emerging MPEG Standard. *IEEE MultiMedia*, 18(3):86–94, 2011. [cited at p. 14]

[Goo11] Google. Goggles. `http://www.google.com/mobile/goggles/`, 2011. [cited at p. 2]

[Goo14] Google. Glass. `http://www.google.com/glass/start/`, 2014. [cited at p. 2]

[Gro] Independent JPEG Group. http://www.ijg.org/. [cited at p. 25, 36, 40, 43]

[HHIa] HHI. HEVC software. `http://hevc.hhi.fraunhofer.de/`. [cited at p. 24, 93]

[HHIb] HHI. JM reference software. `http://iphome.hhi.de/suehring/tml/`. [cited at p. 24, 54, 61, 65, 71, 73]

[HLL$^+$12] H.-M. Hu, B. Li, W. Lin, W. Li, and M.-T. Sun. Region-Based Rate Control for H.264/AVC for Low Bit-Rate Applications. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(11):1564–1576, 2012. [cited at p. 61]

[HTL10] Mark J. Huiskes, Bart Thomee, and Michael S. Lew. New Trends and Ideas in Visual Concept Detection: the MIR Flickr Retrieval Evaluation Initiative. In *Multimedia Information Retrieval*, pages 527–536. ACM, 2010. [cited at p. 110]

[Ima] ImageCLEF. `http://imageclef.org/datasets`. [cited at p. 21]

[IR07] ITU-R. Rec. BT.1788 Methodology for the Subjective Assessment of Video Quality in Multimedia Applications, 2007. [cited at p. 17, 81]

[ISO14] ISO/IEC. Test Model 10: Compact Descriptors for Visual Search, April 2014. [cited at p. 15]

[IT96] ITU-T. Rec. T.84 Information Technology - Digital Compression and Coding of Continuous-Tone Still Images: Extensions. *Unknown Journal*, 1996. [cited at p. 17, 40, 43, 44, 47]

[IT08] ITU-T. Rec. P.910 Subjective video quality assessment methods for multimedia applications, 2008. [cited at p. 81]

[JDS08] H. Jégou, M. Douze, and C. Schmid. Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search. In *Proc. European Conference on Computer Vision (ECCV)*, pages 304–317, Marseille, France, October 2008. Springer. [cited at p. 63]

[JDS10] H. Jégou, M. Douze, and C. Schmid. Improving Bag-of-Features for Large Scale Image Search. *International Journal on Computer Vision*, 87(3):316–336, February 2010. [cited at p. 21]

[JDS11] H. Jégou, M. Douze, and C. Schmid. Exploiting Descriptor Distances for Precise Image Search, 2011. Technical Report 7656, INRIA. [cited at p. 22, 32]

[JDSP10] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating Local Descriptors into a Compact Image Representation. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 3304–3311, 2010. [cited at p. 21]

[JKAA06] G. Jeong, C. Kim, H. Ahn, and B. Ahn. JPEG Quantization Table Design for Face Images and Its Application to Face Recognition. *Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E89-A(11):2990–2993, November 2006. [cited at p. 47]

[JVT03] JVT. Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T rec. H.264-ISO/IEC 14496-10 AVC), March 2003. [cited at p. 54]

[Kha03] S. Khayam. The Discrete Cosine Transform (DCT): Theory and Application, 2003. Tutorial, Department of Electrical & Computing Engineering, Michigan State University. [cited at p. 48]

[KO05] P. Korshunov and W. T. Ooi. Critical Video Quality for Distributed Automated Video Surveillance. In *ACM Multimedia*, pages 151–160, 2005. [cited at p. 120]

[KS04] Y. Ke and R. Sukthankar. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 506–513, October 2004. [cited at p. 13]

[KS09] M. Konrad and H. Stögner. Evolutionary Optimization of JPEG Quantization Tables for Compressing Iris Polar Images in Iris Recognition Systems. In *Proc. IEEE International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 534–539, September 2009. [cited at p. 47]

[Lay15] Layar. `https://www.layar.com/`, 2015. [cited at p. 2]

[LCS11] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary Robust Invariant Scalable Keypoints. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November 2011. [cited at p. 8, 19, 32]

[LGP$^+$06] Z. G. Li, W. Gao, F. Pan, S. W. Ma, K. P. Lim, G. N. Feng, X. Lin, S. Rahardja, H. Q. Lu, and Y. Lu. Adaptive Rate Control for H.264. *Journal of Visual Communication and Image Representation*, 17(2):376–406, April 2006. [cited at p. 71, 72]

[LGV12] K. Lenc, V. Gulshan, and A. Vedaldi. Vlbenchmarks. `http://www.vlfeat.org/benchmarks/`, 2012. [cited at p. 32, 34, 47, 48, 49]

[Lin94] T. Lindeberg. Scale-Space Theory: A Basic Tool for Analysing Structures at Different Scales. *Journal of Applied Statistics*, 21(2):224–270, 1994. [cited at p. 10]

[Lin98] T. Lindeberg. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 30(2):79–116, 1998. [cited at p. 9, 27, 28]

[Lin09] T. Lindeberg. Scale-Space. *Encyclopedia of Computer Science and Engineering*, 4:2495–2504, 2009. [cited at p. 8, 9, 28, 29]

[Lin12] T. Lindeberg. Scale Selection Properties of Generalized Scale-Space Intererest Point Detectors. *Journal of Mathematical Imaging and Vision*, 4, September 2012. [cited at p. 8, 9, 28]

[LK11] W. Lin and C.-C. J. Kuo. Perceptual Visual Quality Metrics: A Survey. *Journal of Visual Communication and Image Representation*, 22(4):297–312, 2011. [cited at p. 17]

[LMSR08] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning Realistic Human Actions from Movies. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, 2008. [cited at p. 121]

[Low04] D.G. Lowe. Distinctive Image Feature from Scale-Invariant Keypoints. *International Journal on Computer Vision*, 60(2):91–110, November 2004. [cited at p. v, 2, 10, 11, 14, 19, 25, 47, 54, 57, 63]

[LPL$^+$03] Z. G. Li, F. Pan, K. P. Lim, G. Feng, X. Lin, and S. Rahardja. Adaptive Basic Unit Layer Rate Control for JVT. Document JVT-G012-rl, March 2003. [cited at p. 61]

[LSW05] K. P. Lim, G. Sullivan, and T. Wiegand. Text Description of Joint Model Reference Encoding Methods and Decoding Concealment Methods. Document JVT-O079, April 2005. [cited at p. 71, 72, 73]

[LT07] A. Leontaris and A. M. Tourapis. Rate Control Reorganization in the JM Reference Software. Document JVT-W042, April 2007. [cited at p. 61]

[MCC$^+$09] M. Makar, C.-L. Chang, D. Chen, S. Tsai, and B. Girod. Compression of Image Patches for Local Feature Extraction. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 821–824, Taipei, Taiwan, April 2009. [cited at p. 15]

[MCT$^+$14] M. Makar, V. Chandrasekhar, S. S. Tsai, D. Chen, and B. Girod. Interframe Coding of Feature Descriptors for Mobile Augmented Reality. *IEEE Transactions on Image Processing*, 23(8):3352–3367, August 2014. [cited at p. 15, 92, 93, 98, 102, 103, 109, 110, 111, 116]

[MCUP02] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *Proc. British Machine Vision Conference (BMVC)*, pages 384–396, Cardiff, UK, September 2002. [cited at p. 8]

[Met15] Metaio. `http://www.metaio.com/home/`, 2015. [cited at p. 2]

[ML09] M. Muja and D. G. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Application (VISSAPP)*, pages 331–340, 2009. [cited at p. 54]

[MLCG12] M. Makar, H. Lakshman, V. Chandrasekhar, and B. Girod. Gradient Preserving Quantization. In *Proc. IEEE International Conference on Image Processing (ICIP)*, Orlando, Florida, USA, September-October 2012. [cited at p. 15, 47]

[MM12] O. Miksik and K. Mikolajczyk. Evaluation of Local Detectors and Descriptors for Fast Feature Matching. In *Proc. IEEE International Conference on Pattern Recognition (ICPR)*, Tsukuba Science City, Japan, November 2012. [cited at p. 13, 35]

[MPE] MPEG. Compact Descriptors for Visual Search. `http://mpeg.chiariglione.org/standards/mpeg-7/compact-descriptors-visual-search`. [cited at p. 15, 86, 89]

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. [cited at p. 22]

[MS01] K. Mikolajczyk and C. Schmid. Indexing Based on Scale Invariant Interest Points. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 525–531, Vancouver, Canada, 2001. [cited at p. 27]

[MS04] K. Mikolajczyk and C. Schmid. Scale & Affine Invariant Interest Point Detectors. *International Journal on Computer Vision*, 60(1):63–86, 2004. [cited at p. 7, 8]

[MS05] K. Mikolajczyk and C. Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005. [cited at p. 13, 19, 32, 35]

[MTC$^+$13] M. Makar, S. S. Tsai, V. Chandrasekhar, D. Chen, and B. Girod. Interframe Coding of Canonical Patches for Low Bit-Rate Mobile Augmented Reality. *International Journal of Semantic Computing*, 7(1):5–24, 2013. [cited at p. 15, 18, 23, 83, 92, 93, 95, 109, 110]

[MTS$^+$05] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65(1-2):43–72, November 2005. [cited at p. 8, 13, 18, 19, 20, 26, 27, 32, 33, 34, 35, 40, 46, 47, 48, 56, 58]

[Nos01] A. Nosratinia. Enhancement of JPEG-Compressed Images by Re-Application of JPEG. *Journal of VLSI Signal Processing*, 27:69–79, February 2001. [cited at p. 37]

[OHC11] T. S. Ou, Y. H. Huang, and H. H. Chen. SSIM-Based Perceptual Rate Control for Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(5):682–691, May 2011. [cited at p. 61]

[Ope] OpenCV. `http://opencv.org/`. [cited at p. 34, 64]

[Ort96] A. Ortega. Optimal Bit Allocation Under Multiple Rate Constraints. In *Proc. IEEE Data Compression Conference (DCC)*, pages 349–358, 1996. [cited at p. v, 42, 43]

[PCI$^+$07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Minneapolis, US, June 2007. [cited at p. 21, 22, 63, 64, 83, 110]

[PSM10] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher Kernel for Large-Scale Image Classification. In *Proc. European Conference on Computer Vision (ECCV)*, pages 143–156. Springer, 2010. [cited at p. 21]

[PvV05] T.Q. Pham and L. van Vliet. Blocking Artifacts Removal by a Hybrid Filter Method. In *Proc. Annual Conference of the Advanced School for Computing and Imaging*, pages 372–377. TU Delft, 2005. [cited at p. 37]

[Qua15] Qualcomm. Vuforia. `https://developer.vuforia.com/`, 2015. [cited at p. 2]

[RBA$^+$13] A Redondi, L. Baroffio, J. Ascenso, M. Cesano, and M. Tagliasacchi. Rate-Accuracy Optimization of Binary Descriptors. In *IEEE International Conference on Image Processing (ICIP)*, pages 2910–2914, September 2013. [cited at p. 15]

[RBCT13] A Redondi, L. Baroffio, M. Cesana, and M. Tagliasacchi. Compress-Then-Analyze vs. Analyze-Then-Compress: Two Paradigms for Image Analysis in Visual Sensor Networks. In *IEEE International Workshop on Multimedia Signal Processing (MMSP)*, pages 278–282, September 2013. [cited at p. 15]

[RD06] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *Proc. European Conference on Computer Vision (ECCV)*, pages 430–443, Graz, Austria, May 2006. Springer. [cited at p. 8]

[SAS07] P. Scovanner, S. Ali, and M. Shah. A 3-Dimensional SIFT Descriptor and Its Application to Action Recognition. In *Proc. ACM Multimedia*, pages 357–360, 2007. [cited at p. 121]

[Sen15] SenseTime. http://www.sensetime.com/en, 2015. [cited at p. 2]

[Sly15] Slyce. http://slyce.it/, 2015. [cited at p. 2]

[SMB00] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of Interest Point Detectors. *International Journal of Computer Vision*, 37(2), June 2000. [cited at p. 18]

[SS04] G. Schaefer and M. Stich. UCID - An Uncompressed Colour Image Database. In *Proc. SPIE Storage and Retrieval Methods and Applications for Multimedia*, pages 472–480, 2004. [cited at p. 44]

[SSBC10] K. Seshadrinathan, R. Soundararajan, A. C. Bovik, and L. K. Cormack. Study of Subjective and Objective Quality Assessment of Video. *IEEE Transactions on Image Processing*, 19(6):1427–1441, 2010. [cited at p. 17]

[SW98] G.J. Sullivan and T. Wiegand. Rate-Distortion Optimization for Video Compression. *IEEE Signal Processing Magazine*, 15(6):74–90, 1998. [cited at p. 43, 44, 57]

[SZ03] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 1470–1477, Nice, France, October 2003. [cited at p. 21, 63, 83]

[Sze11] R. Szeliski. *Computer Vision: Algorithms and Applications.* Springer, 2011. [cited at p. 1]

[TCT$^+$12] S. S. Tsai, D. Chen, G. Takacs, V. Chandrasekhar, M. Makar, R. Grzeszczuk, and B. Girod. Improved Coding for Image Feature Location Information. In *Proc. SPIE Applications of Digital Image Processing*, volume 8499, 2012. [cited at p. 86, 87, 89]

[Tin15] TinEye. http://www.tineye.com/, 2015. [cited at p. 2]

[TUM] TUM. 720p format video sequences. ftp://ftp.ldv.ei.tum.de/videolab/public/SVT_Test_Set/720p/. [cited at p. 65, 112]

[Uni] Arizona State University. http://trace.eas.asu.edu/yuv/. [cited at p. 59, 64]

[Ved12] A. Vedaldi. Modern Features: Advances, Applications, and Software. https://sites.google.com/site/eccv12features/slides, 2012. [cited at p. 18]

[VLF] VLFeat. http://www.vlfeat.org/. [cited at p. 23, 40, 54]

[VRA12] C. Varytimidis, K. Rapantzikos, and Y. Avrithis. WαSH: Weighted α-Shapes for Local Feature Detection. In *Proc. European Conference on Computer Vision (ECCV)*, pages 788–801. Springer, 2012. [cited at p. 8, 34]

[VSW99] A. Vetro, H. Sun, and Y. Wang. MPEG-4 Rate Control for Multiple Video Objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):186–199, February 1999. [cited at p. 61]

[Wat93] A. B. Watson. DCT Quantization Matrices Visually Optimized for Individual Images. In *Proc. SPIE Human Vision, Visual Processing, and Digital Display*, pages 286–291, 1993. [cited at p. 47]

[WBSS04] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, April 2004. [cited at p. 17, 61]

[WFW11] Z. Wang, B. Fan, and F. Wu. Local Intensity Order Pattern for Feature Description. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 603–610, Barcelona, Spain, November 2011. [cited at p. 13, 32, 35]

[Wik15] Wikitude. `https://www.wikitude.com/`, 2015. [cited at p. 2]

[WKSL11] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176, 2011. [cited at p. 121]

[WSG05] T. Wiegand, E. Steinbach, and B. Girod. Affine Multipicture Motion-Compensated Prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):197–209, Feb 2005. [cited at p. 98, 102]

[WTG08] Geert Willems, Tinne Tuytelaars, and Luc Gool. An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector. In *Proc. European Conference on Computer Vision (ECCV)*, pages 650–663. Springer, 2008. [cited at p. 121]

[You87] R. Young. The Gaussian Derivative Model for Spatial Vision: I. Retinal Mechanisms. *Spatial Vision*, 2:273–293, 1987. [cited at p. 28]

[YSWY12] H. Yue, X. Sun, F. Wu, and J. Yang. SIFT-Based Image Compression. In *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, pages 473–478, 2012. [cited at p. 86]

[YSYW13] H. Yue, X. Sun, J. Yang, and F. Wu. Cloud-Based Image Coding for Mobile Devices - Toward Thousands to One Compression. *IEEE Transactions on Multimedia*, 15(4):845–857, 2013. [cited at p. 86, 87]

[ZNC09] D. Zhang, K. N. Ngan, and Z. Chen. A Two-Pass Rate Control Algorithm for H.264/AVC High Definition Video Coding. *Signal Processing: Image Communication*, 24(5):357–367, 2009. [cited at p. 61]