Pattern Matching in Compressed Texts and Images
Pattern Matching in Compressed Texts and Images

Don Adjeroh
West Virginia University
USA
don@csee.wvu.edu

Tim Bell
University of Canterbury
New Zealand
tim.bell@canterbury.ac.nz

Amar Mukherjee
University of Central Florida
USA
amar@eecs.ecf.edu

now
the essence of knowledge
Boston – Delft
Editorial Scope

Foundations and Trends® in Signal Processing will publish survey and tutorial articles on the foundations, algorithms, methods, and applications of signal processing including the following topics:

- Adaptive signal processing
- Audio signal processing
- Biological and biomedical signal processing
- Complexity in signal processing
- Digital and multirate signal processing
- Distributed and network signal processing
- Image and video processing
- Linear and nonlinear filtering
- Multidimensional signal processing
- Multimodal signal processing
- Multiresolution signal processing
- Nonlinear signal processing
- Randomized algorithms in signal processing
- Sensor and multiple source signal processing, source separation
- Signal decompositions, subband and transform methods, sparse representations
- Signal processing for communications
- Signal processing for security and forensic analysis, biometric signal processing
- Signal quantization, sampling, analog-to-digital conversion, coding and compression
- Signal reconstruction, digital-to-analog conversion, enhancement, decoding and inverse problems
- Speech/audio/image/video compression
- Speech and spoken language processing
- Statistical/machine learning
- Statistical signal processing
  - classification and detection
  - estimation and regression
  - tree-structured methods

Information for Librarians
Foundations and Trends® in Signal Processing, 2012, Volume 6, 4 issues. ISSN paper version 1932-8346. ISSN online version 1932-8354. Also available as a combined paper and online subscription.
Pattern Matching in Compressed Texts and Images

Don Adjeroh¹,* , Tim Bell²,† and Amar Mukherjee³,‡

¹ Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506-6109, USA, don@csee.wvu.edu
² Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, tim.bell@canterbury.ac.nz
³ School of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL 32816, USA, amar@eecs.ucf.edu

Abstract

This review provides a survey of techniques for pattern matching in compressed text and images. Normally compressed data needs to be decompressed before it is processed, but if the compression has been done in the right way, it is often possible to search the data without having to decompress it, or at least only partially decompress it. The problem can be divided into lossless and lossy compression methods, and then in each of these cases the pattern matching can be either exact

*This work was partially supported by grants from the US National Science Foundation IIS-0228370 and IIS-0312484.
†Supported by a grant from the Hitachinaka Techno Center Inc, Japan.
‡This work has been partially supported by grants from the US National Science Foundation IIS-9977336 and IIS-0207819.
or inexact. Much work has been reported in the literature on techniques for all of these cases, including algorithms that are suitable for pattern matching for various compression methods, and compression methods designed specifically for pattern matching. This work is surveyed in this review. The review also exposes the important relationship between pattern matching and compression, and proposes some performance measures for compressed pattern matching algorithms. Ideas and directions for future work are also described.

*Keywords:* Compressed pattern matching; text compression; image compression; performance measures; searching.
## Contents

1 Introduction 1
1.1 Data Compression Methods 4
1.2 Compressed Pattern Matching 6
1.3 Compressed Domain versus Transform Domain Analysis 7
1.4 Organization 8

2 Search Strategies 11
2.1 The Pattern Matching Problem and its Variants 11
2.2 Search Strategies for Text 15
2.3 Search Strategies for Images 21

3 Relationship Between Searching and Compression 25
3.1 Pattern Matching for Compression 25
3.2 Compression for Pattern Matching 28

4 Searching Compressed Data: Performance Measurement 31
4.1 Performance Measures for Compression Algorithms 31
4.2 Performance Measures for Pattern Matching Algorithms 34
4.3 Performance Measures for Compressed Pattern Matching 35
5 Searching Compressed Data: Text 39

5.1 Dictionary Methods 40
5.2 Searching on LZ-encoded Text 43
5.3 Grammar-based Compression 52
5.4 Run-length Encoding 60
5.5 Statistical (symbolwise) Compression 61
5.6 Text Compression by Sorted-contexts 67
5.7 Variable-to-fixed Length Codes 69
5.8 Others 73

6 Searching Compressed Data: Images 77

6.1 Searching Lossless Compressed Images 79
6.2 Searching Compressed Document Images 89
6.3 Searching on Lossy Compressed Images 92

7 Directions for Further Research 109

7.1 New Compressed Pattern Matching Algorithms 109
7.2 New Search-aware Compression Algorithms 112
7.3 Compressed Image Descriptors 113
7.4 Performance Measures 114
7.5 Integration and Adaptation 114
7.6 Hardware Implementation 116
7.7 New Applications 116

8 Conclusion 119

Acknowledgments 121

References 123
Given a text sequence (the database) and a pattern sequence (the query), the pattern matching problem is to search in the text to determine all the locations (if any) where the pattern occurs. Searching for a pattern is an important activity that is performed on a daily basis, with significant applications in a diverse range of areas, including signal processing, computer vision, robotics, pattern recognition, data mining, computational biology, and bioinformatics, to name a few. Today computers are increasingly being used to process text, digitized images, digital video, and various other types of data. However, representing the data can require large amounts of storage space, and operations on the data, such as searching for a pattern, are often time consuming. Even the amount of image or textual data typically available to the ordinary user has witnessed a tremendous growth due to a number of factors, such as improvements in storage and communication technologies (for example, the web, electronic mail, smartphones, general wireless mobile devices), widespread deployment of digital libraries, improved document processing techniques, and the availability of different types of sensors, including cameras and scanners. Apart from
the problem of sheer size, the huge amounts of data involved also pose problems for efficient search and retrieval of the required information from the stored data.

Since the digitized data are usually stored using compression techniques, and because of the problem of efficiency (in terms of storage space, computational time, and power consumption), the trend now is to keep the compressed data in its compressed form for as much time as possible. This means that operations such as search and analysis on the data (be it text or images) is ideally performed directly on the compressed representation, without decompression, or at least, with minimal decompression. Intuitively, compared to working on the original uncompressed data, operating directly on the compressed data will require the manipulation of less data, and hence should be more efficient. This also avoids the often time-consuming process of decompression, and the problem of temporary storage space that may be required to keep the decompressed data. The need to search data directly in its compressed form has been recognized by international compression standards such as MPEG-4, MPEG-7 [31] and H.264 [37], where part of the requirement is the ability to search for objects directly in the compressed video.

This review surveys techniques that solve the two basic problems of efficiency (in storage and computation) at the same time. That is, the digitized image or text is stored and searched in a compressed format. Without addressing both problems together, compression and searching work against each other, since a simple system would have to decompress a file before searching it, thus slowing down the pattern matching process. However, there is a strong relationship between compression and pattern matching, and this can be exploited to enable both tasks to be performed efficiently at the same time.

In fact, pattern matching can be regarded as the basis of compression. For example, a dictionary compression system might identify the occurrences of an English word in a text, and replaces these with a reference to the word in a lexicon. The main task of the compression system is thus to identify patterns (in this example, words), which are then represented using a compact code. The identified pattern need not be an exact match every time (for example, in lossy compression).
Thus, the strong connection between searching and compression can be traced way back to the early days of lossy compression of signals. For instance, Shannon proposed block source coding with a fidelity criterion (essentially the basis for vector quantization), whereby the encoder uses a reproduction codebook (akin to the dictionary above), and searches in the codebook for the codeword with a minimum distortion to the input vector, and transmits the index for the codeword. If the type of pattern used for compression is the same as the type being used during a later search of the text, then the compression system can be exploited directly to perform a fast search. In the example of the dictionary system, if users wish to search the compressed text for words, then they could look up the word in the lexicon, which would immediately establish whether a search will be successful. If the word is found, then its code could be determined, and the compressed text searched for the code. This will considerably reduce the amount of data to be searched, and the search will be matching whole words rather than a character at a time. In one sense, much of the searching has already been performed off-line at the time of compression.

The potential savings are significant. Text can typically be compressed to less than one-third of its original size, and images are routinely compressed to a tenth or even a hundredth of the size of the raw data. These factors indicate that there is considerable potential to speed up searching, and indeed, systems exist that are able to achieve much of this potential saving. For instance, compressed domain indexing and retrieval is the preferred approach to multimedia information management, where orders of magnitude speedup have been recorded over operations on uncompressed data.

There are various surveys on the general problem of content-based image and video retrieval. None has focused on the restricted problem of compressed domain image retrieval. Similarly, pattern matching and compression were considered in the survey paper. However, compressed pattern matching was not considered, and various techniques have been developed since then. More importantly, we try to compare and contrast methods that have been proposed for compression, and for compressed pattern matching in text and images. Thus we consider compressed pattern matching for both lossy...
compression (used for images) and for lossless compression (used for text and images).

1.1 Data Compression Methods

We begin with a brief introduction to data compression methods. More complete introductions to the topic of compression are available [50, 91], and the reader is referred to textbooks for more details [3, 51, 264, 294, 298]. There is also an IEEE conference series on data compression from 1991 to the present — see http://www.cs.brandeis.edu/~dcc/.

Compression methods are generally classed as *lossless* or *lossy*. Lossless methods enable the original data to be recovered exactly, without any error. Lossless compression, sometimes called *text compression*, is typically used for text, and to a lesser extent on images, such as in medical imaging where exact reconstruction of the original image is important. In contrast, lossy methods allow some deterioration in the original data, and are generally applied in situations where the data have been digitized from an analog source (such as images, video and audio). Usually the level of deterioration is near-imperceptible, yet considerable compression improvement can be achieved because the system is not storing unnecessary detail. Many lossy methods include a lossless method as a sub-component. For example, an image or video compression system might transform the image to a frequency domain, quantize some of the frequency domain coefficients (which is a lossy step), and then encode the coefficients using a lossless method.

Figure 1.1 shows a general model of what happens in data compression. The data transformation stage transforms the input data into a form that will make it easier to compress, for instance by exposing the redundancies or repetitions in the data. Some transformation schemes convert the input data or signal into the frequency domain, with the aim of packing most of the energy in the signal into only a few transform coefficients. The specific transformation performed depends on the type of input data. For text, the transformation could be a simple cyclic permutation of the text sequence (for example, the Burrows–Wheeler transform, or BWT [3, 70]); for images, we could have spatial prediction where the original image is essentially represented as a sequence.
1.1 Data Compression Methods

Fig. 1.1 General model for data compression. The triangle markers show the points where pattern matching on the transformed or compressed data can take place.

of prediction errors; for audio signals we could have temporal prediction on the signals; and for video, spatio-temporal prediction using motion vectors can also be viewed as a form of transformation. Standard linear transforms that pack the energy into a few coefficients, such as the Fourier transform or the wavelet transform, are quite common in image, video and audio compression. The particular transformation applied will have an impact on the compression performance, and on the ability to search the compressed data.

The quantization stage is used to reduce the number of distinct values in a signal to a much smaller number. Input data or a signal may contain a large number of distinct values. For analog signals (without digitization), the signal is continuous and the digitization process must quantize it into a set of distinct values, so a digital video has already had some quantization applied. Transforming the digitized signal puts the representation into a domain where the effect of further quantization will be less perceptible. Quantization of the transformed signal reduces the representation to a smaller set of distinct values, each of which can be represented using fewer bits, thus requiring reduced data storage. Quantization however leads to reduced accuracy in the data representation. In fact, the quantization stage represents the major source of compression, but also the major source of data loss (error) in the reconstructed data. In the case of a channel with limited bandwidth, the data rate available will determine the level of quantization needed, and in this case the quantization can be seen as maintaining the best possible fidelity for the capacity available. In some cases for images, video, or audio, it may be possible to keep the loss of accuracy to a level where
Introduction

It isn’t observable by a human, given the limitations of human perception. Thus, quantization is common in such applications. Quantization could be performed on either scalar or on vector values, leading to the respective notions of scalar quantization and vector quantization. The encoding stage (also called the coding stage) codes the data to remove further redundancies, often based on the probability distribution of symbols in the data, or symbols in the quantized data for lossy compression.

Decompression involves performing the reverse operations of decoding, de-quantization and inverse transformation. The operations before and after the quantization stage are generally reversible and hence do not introduce any loss or artefacts in the compression. (Here, we ignore errors due to limited arithmetic precision, such as round-off errors during the transformation stage). Quantization, however, is not reversible and thus introduces some error in the compression process. In effect, from the viewpoint of compression models, the major difference between lossless and lossy compression is the quantization stage: lossless compression does not involve any quantization. The quantization stage also accounts for the huge compression ratios often achievable in lossy data compression.

1.2 Compressed Pattern Matching

In traditional pattern matching one is given a text and a pattern, and the problem is to determine whether the pattern occurs in the text. The search could be exact, whereby the pattern matches a substring of the text with no errors, or could be inexact or approximate, whereby some mismatches or errors could be allowed in the match. The result could be simply a binary decision on whether the pattern occurs in the text, a count of the number of occurrences, or possibly a listing of all the locations of occurrences (if any). Variations of the pattern matching problem have been studied, including multiple pattern matching, parameterized pattern matching, and multi-dimensional pattern matching. Compressed pattern matching involves one or more of the pattern matching variants, with the constraint that either the text, the pattern, or both are in compressed form [15] [133]. In general,
1.3 Compressed Domain versus Transform Domain Analysis

exact pattern matching is a natural fit for lossless compression, though inexact pattern matching can also be performed on lossless compressed data. On the other hand, given their nature, one can only hope for approximate matches for lossy compressed data. We identify two major categories for compressed pattern matching problems. **Fully compressed pattern matching** is when the text and the pattern are both compressed, and matching involves no form of decompression. Let $\mathcal{C}$ be a compression scheme. Then, given $T_c = \mathcal{C}(T)$, the compressed version of the text $T$, and $P_c = \mathcal{C}(P)$, the compressed form of the pattern $P$, the problem of fully compressed pattern matching is to determine all the positions in $T$, where the pattern $P$ occurred, without first decompressing $T_c$ and $P_c$. The more general term **compressed pattern matching** is used for the following less restricted problem: Given $T_c = \mathcal{C}(T)$, and the (uncompressed) pattern $P$, determine all the positions in $T$, where the pattern $P$ occurred, without first decompressing $T_c$. Most efficient pattern matching algorithms perform some type of preprocessing (either on the prefixes or the suffixes) of the pattern, $P$. Thus fully compressed pattern matching is made more difficult, since the usual preprocessing on the pattern is no longer easy to achieve without decompressing $P$. For text, various algorithms have been proposed for both variants of the compressed pattern matching problem. For compressed-domain image retrieval, the usual assumption is that the query image is not compressed. However, some methods on 2D image pattern matching also consider fully-compressed pattern matching.

1.3 Compressed Domain versus Transform Domain Analysis

Our primary focus in this review is on compressed pattern matching for text and images. Compressed pattern matching is one activity under the general area of compressed domain analysis. In image and video analysis, the terms “compressed domain” and “transform domain” are often used interchangeably. Here we distinguish between the two. With compressed domain analysis, the required analysis is performed on the compressed data with minimal or no decoding, and before the stage of inverse quantization. For transform domain analysis, the required analysis is performed on the transform coefficients, typically
Introduction

Fig. 1.2 Transform domain versus compressed domain analysis. Whether operating in the transform domain or compressed domain depends on the point(s) in the compression pipeline where the analysis is performed. The type(s) of analyses are characterized as follows: transform domain: points (1), (2), (4), and (5); compressed domain: point (3); compressed domain with partial decoding: points (4) and (5).

after the transformation or quantization stages in the compression pipeline, but before the final encoding stage. Here the objective of the transformation could be simply for efficient analysis, and not necessarily for compression or data storage. This is typical in some signal analysis or image analysis applications. If the data is already compressed, transform domain analysis could be performed after decoding the compressed stream, but before the inverse transformation stage. Figure 1.2 uses the compression pipeline of Figure 1.1 to explain the difference between compressed domain and transform domain analysis. For images or video, compressed (or transform) domain analysis could include general operations, such as image enhancement, noise removal, shape, color or texture-based image retrieval, etc. Recent surveys on general compressed or transform domain analysis of images and video are provided in [254, 343]. From the description above, most video and image retrieval and search operations, especially those for lossy compression, are performed in the transform domain (mainly after the stages of transformation, or quantization in a few cases, but generally before the encoding stage). Some text pattern matching methods operate in the transform domain (with partial decoding), while others operate in the compressed domain, that is, after final encoding, such as using Huffman codes.

1.4 Organization

In this review, we survey compression methods for text and images, especially identifying the search techniques that they use, and how they
could be exploited for searching the compressed data later. First, we consider search strategies and pattern matching methods for uncompressed data, to set the scene for more sophisticated systems. Next we explore the interesting relationship between searching and data compression. This is followed by a discussion on performance measurement for compressed pattern matching. The next two sections survey techniques that have been developed for searching compressed data, which is sometimes called *compressed-domain pattern matching*. The first section looks at methods for lossless data compression (as used for text). The next section then surveys methods that have been proposed for pattern matching on compressed images. This is presented in two parts. The first part considers methods for pattern matching on lossless compressed images, which often borrow a lot from methods for searching compressed text. The second part surveys methods where the image has been compressed using lossy methods. The review concludes with a speculation on the likely directions of future work in the area.

Given the breadth of the topics involved, we will focus only on compressed pattern matching for text and images. Video and audio will be mentioned at times, but without much detail. Also, we will only briefly describe other types of signal processing activities (different from pattern matching) that are often performed in the compressed domain. For lossless compression (for text and images), we focus mainly on exact pattern matching. For lossy compression, we focus mainly on compressed domain or transform domain retrieval. Throughout this work, we assume that data from analog sources, such as images, audio and video, have already been digitized. Thus we ignore the potential data loss due to the digitization process.


References


References


References


References


References

References


References


Full text available at: http://dx.doi.org/10.1561/2000000038
References


References

References


References


References


Full text available at: http://dx.doi.org/10.1561/2000000038
References


References

143


References


References


References


References


