Theoretical Computer Science 638 (2016) 52–62

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Compressed matching for feature vectors ☆

Shmuel T. Klein [a], Dana Shapira [b,*]

[a] Department of Computer Science, Bar Ilan University, Ramat Gan 52900, Israel
[b] Department of Computer Science, Ariel University, Ariel 40700, Israel

ARTICLE INFO

Article history:
Received 27 March 2015
Received in revised form 6 December 2015
Accepted 15 December 2015
Available online 21 December 2015

Keywords:
Data compression
Feature vectors
SIFT
Fibonacci code

ABSTRACT

The problem of compressing a large collection of feature vectors is investigated, so that object identification can be processed on the compressed form of the features. The idea is to perform matching of a query image against an image database, using directly the compressed form of the descriptor vectors, without decompression. Specifically, we concentrate on the Scale Invariant Feature Transform (SIFT), a known object detection method, as well as on Dense SIFT and PHOW features, that contain, for each image, about 300 times as many vectors as the original SIFT. Given two feature vectors, we suggest achieving our goal by compressing them using a lossless encoding by means of a Fibonacci code, for which the pairwise matching can be done directly on the compressed files. In our experiments, this approach improves the processing time and incurs only a small loss in compression efficiency relative to standard compressors requiring a decoding phase.

1. Introduction

One of the topics on which Amihood Amir has done pioneering work is known as *Compressed Matching*. This is an extension of the classical Pattern Matching paradigm, in which the match has to be performed in the compressed domain, without first decompressing. Given a pattern P, a text T and complementing encoding and decoding functions $\mathcal{E}$ and $\mathcal{D}$, the Compressed Matching problem is to locate P in the compressed text $\mathcal{E}(T)$. While the traditional approach searches for the pattern in the decompressed text, i.e., searching for P in $\mathcal{D}(\mathcal{E}(T))$, compressed matching calls for rather compressing the pattern too, and looking for $\mathcal{E}(P)$ in $\mathcal{E}(T)$, with the necessary adaptations. This has first been mentioned in Amir's paper with Benson dealing with two-dimensional run-length coding at the Data Compression Conference in 1992 [1,2], and has since then triggered a myriad of related investigations. As a tribute to Amir, we present here yet another application of the compressed matching idea, this time to compressed feature vectors used in Image Processing.

The tremendous storage requirements and ever increasing resolutions of digital images, necessitate automated analysis and compression tools for information processing and extraction. A main challenge is detecting patterns even if they were rotated or scaled, working directly on the compressed form of the image. In a more general setting, a collection of images could be given, and the subset of those including at least one object, which is a rotated or scaled copy of the original object, is sought. An example for the former could be an aerial photograph of a city in which a certain building is to be located,

☆ This is an extended version of papers that have been presented at the Information Mining and Management Conference (IMMM 2014) and the Prague Stringology Conference (PSC'14) in 2014, and appeared in their Proceedings.

* Corresponding author.
E-mail addresses: tomi@cs.biu.ac.il (S.T. Klein), shapird@ariel.ac.il (D. Shapira).

http://dx.doi.org/10.1016/j.tcs.2015.12.021
0304-3975/

an example for the more general case could be a set of pictures of faces of potential suspects, which have to be matched against some known identifying feature, like a nose or an eyebrow.

There are several methods for transforming an image into a set of feature vectors for the purpose of object detection, such as SIFT (Scale Invariant Feature Transform) by Lowe [29], GLOH (Gradient Location and Orientation Histogram) [31], and SURF (Speed-up-Robust Features) [5], to mention only a few.

SIFT is a high probability object detection and identification method, which is done by matching the query image against a large database of local image features. Lowe's object recognition method transforms an image into a set of feature vectors, each of which is invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion. Feature descriptor vectors are computed for the extracted key points of objects from a set of reference images, which are then stored in a database. An object in a new image is identified after matching its features against this database using the Euclidean L_2 distance.

The matching process consists in a first stage of comparing each feature vector of the query image with each feature vector in the database. In a second stage, the best matching candidates are filtered out, and a clustering process is applied. Finally, each cluster passes a further more detailed model verification [29]. While the first stage could be done in a single sequential scan of the database, the latter stages require the possibility of direct access to the individual feature vectors.

The main idea of SIFT is to carefully choose a subset of the features so that this reduced set will be representative of the original image and will be processed instead. Obviously, there are applications in which working on a dense set of features, rather than the sparse subsets mentioned above, is much better, since a larger set of local image descriptors provides more information than the corresponding descriptors evaluated only at selected interest points. In the case of object category or scene classification, experimental evaluations show that better classification results are often obtained by computing the so-called Dense SIFT descriptors (or DSIFT for short) as opposed to SIFT on a sparse set of interest points [7]. The dense sets may contain about 300 times more vectors than the sparse sets.

Query feature compression can contribute to faster retrieval, for example, when the query data is transmitted over a network, as in the case when mobile visual applications are used for identifying products in comparison shopping. Moreover, since the memory space on the mobile device is restricted, working directly on the compressed form of the data is sometimes required. A device with restricted memory is also an example showing that one still may need the space saving implications of compression, even though the time savings are often emphasized.

In this paper we suggest to apply metric preserving compression methods on the features of an image so that they can be processed in their compressed form. There are two ways to interpret the expected gains: on the one hand, one may consider the reduction of the required space; on the other hand, assuming that the space to be used is fixed in advance, compression allows the storage of more vectors, so that, instead of choosing a representative set of interest points, possibly reducing the object detection accuracy, one can increase the number of key points that can be processed using the same amount of memory storage.

2. Related work

A feature descriptor encoder is presented in Chandrasekhar et al. [15]. They transfer the compressed features over the network and *decompress* them once data is received for further pairwise image matching. Chen et al. [16] perform tree-based retrieval, using a scalable vocabulary tree. Since the tree histogram suffices for accurate classification, the histogram is transmitted instead of individual feature descriptors. Also Chandrasekhar et al. [13] encode a set of feature descriptors jointly and use tree-based retrieval when the order in which data is transmitted does not matter, as in our case. Several other SIFT feature vector compressors were proposed, and we refer the reader to [12] for a comprehensive survey. DiLillo et al. [19,20] applied compression-based tools (dimensionality reduction, vector quantization, and coding) to provide object recognition as a preprocessing step. These are *lossy* techniques, in which a part of the data cannot be recovered. We propose a special encoding, which is a *lossless* alternative to the above, and is not only compact in its representation, but can also be processed directly *without* any decompression.

Fig. 1 visually represents our approach as opposed to the traditional one of feature based object detectors and previous research regarding feature descriptors compression. The client uses any feature detector for extracting key points from the image, and computes the relevant vectors. These features are then sent along a network to the server, where pairwise pattern matching is applied against the stored database, as shown in Fig. 1(a). Fig. 1(b) depicts the scenario assumed in previous research that deals with compressed feature descriptors: compression is applied to the vectors before transmission, and decompression is performed once the descriptors are received on the server's side. Unlike traditional work, the current suggestion omits the decompression stage, and performs pairwise matching directly on the compressed data, as shown in Fig. 1(c). Similar work, using quantization, has been suggested by Chandrasekhar et al. [14]. We do not apply quantization, and rather use a lossless encoding.

We thus wish to perform the matching against the query image in the compressed form of the feature descriptor vectors so that the metric is retained, i.e., vectors are close in the original distance (e.g., Euclidean distance based on nearest neighbors according to the Best-Bin-First-Search algorithm in SIFT) if and only if they are close in their compressed counterparts. This can be done either by using the same metric but requiring that the compression should not affect the metric, or by changing the distance so that the number of false matches and true mismatches does not increase under this new distance. In the present work, we stick to the first alternative and do not change the L_2 metric used in SIFT.