A Method to Detect Repeated Unknown Patterns in an Image

Paulo J.S.G. Ferreira and Armando J. Pinho(✉)

IEETA/DETI, Universidade de Aveiro, Aveiro, Portugal
{pjf,ap}@ua.pt

Abstract. Consider a natural image that has been manipulated by copying, transforming and pasting back fragments of the image itself. Our goal is to detect such manipulations in the absence of any knowledge about the content of the repeated fragments or the transformations to which they might have been subject. The problem is non-trivial even in the absence of any transformations. For example, copy/paste of a textured fragment of a background can be difficult to detect even by visual inspection. Our approach to the problem is a two-step procedure. The first step consists in extracting features from the image. The second step explores the connection between image compression and complexity: a finite-context model is used to build a complexity map of the image features. Patterns that reappear, even in a somewhat modified form, are encoded with fewer bits, a fact that renders the detection of the repeated regions possible.

Keywords: Tampering detection · Finite-context models · Kolmogorov complexity · SIFT

1 Introduction

Finding repetitions of exact or approximate *unknown* patterns in images can be a difficult problem even for a human observer. This paper addresses the problem and proposes an approach that combines feature extraction with information-theoretic analysis.

Consider a natural image that has been manipulated by copying, transforming and pasting fragments of the image itself. Our goal is to detect such manipulations in the absence of any knowledge about the content of the repeated fragments, or the transformations that they might have undergone.

We stress the fact that the repetitions are *unknown*. There is an important difference between known repetitions and unknown ones. Given one pattern (i.e. a fragment of an image), it is easy to find matching patterns. However, if the

Funded in part by National Funds through FCT - Foundation for Science and Technology, in the context of the project PEst-OE/EEI/UI0127/2014.

A. Campilho and M. Kamel (Eds.): ICIAR 2014, Part I, LNCS 8814, pp. 12–19, 2014.
DOI: 10.1007/978-3-319-11758-4_2

nature of the repeating pattern is totally unknown, the problem becomes much harder (even for exact or almost exact repeats).

The first step in our approach consists in extracting as many relevant features from the image as possible, bearing in mind one crucial condition: the features must be invariant to any transformations that the manipulations might have introduced. For example, if the transformations include rescaling, then the feature extraction step should yield scale-invariant features.

In the examples the feature extraction step is performed using SIFT, a well known method that provides invariance with respect to a number of transformations (see [1], also [2]). However, our approach is not tied to SIFT and the feature extraction step could in principle be performed using other appropriate techniques — either SIFT variants or refinements or totally distinct approaches. The feature extraction step is followed by an analysis step which uses information-theoretic tools. The goal of this second step is to determine the complexity of the features using a class of compression algorithms able to approximate the Kolmogorov complexity of the target. The key insight is that the Kolmogorov complexity of a repeated pattern is essentially that of the pattern itself. Repetitions or quasi-repetitions are associated with low-complexity data regions. To turn this idea into practice we use finite-context models that can capture, in a compact form, the most relevant content in the set of features.

In the absence of any transformations the problem is simpler but still non-trivial. The feature extraction step can then be omitted, and a complexity map of the image itself is known to lead to interesting and useful results [3,4].

2 The Method

2.1 Feature Extraction and Quantization

As stated, the features can be extracted using SIFT or other methods, provided that they are invariant to the transformations used during the image manipulations. We assume that each feature extracted from the image is associated with an image coordinate. For example, in SIFT there are at least four natural parameters: the coordinates, the scale σ and the angle θ. It is also assumed that each feature is a vector with fixed dimension (dimension 128 for SIFT feature vectors). At the end of this step one has a collection of features or vectors, associated with a specific image coordinate. The ith feature will be denoted by $f_i(x, y)$, where (x, y) denotes the associated image coordinate pair. The feature vectors are stored as columns in a data matrix, in no particular order.

It is advisable to go through the set of features and discard those of limited usefulness. This depends on the features and on their parameters. For example, if one is interested in detecting *small* repeated unknown fragments, SIFT features associated with smaller scales σ are probably more relevant than those associated with larger scales. We found that SIFT features with σ greater than $5 - 6$ have a negligible impact on the results. The elements of SIFT feature vectors are integers in the range 0-255, but in general they may be real numbers. Even in

the case of integers, one could wish to reduce the number of amplitude levels, for reasons that will become clear later.

We subject the elements of all non-discarded feature vectors to scalar quantization. Our implementations have used Lloyd-Max quantization or the well known K-means method. The main parameter is the number of clusters, which determines the number of distinct intensities, the initialization policy (typically random) and the overclustering factor (typically 1.0, meaning that no overclustering is performed). In the case of K-means, we used the implementation in the `mlpack` [5] C++ library.

2.2 Sorting

The quantization step yields a collection of quantized feature vectors, stored as columns in a data matrix in no particular order. Before the complexity analysis stage we need to convert the vectors into a single bitstream. First, we sort the feature vectors. Then, we create the bitstream by vectorizing the sorted vectors, i.e. by stacking the columns of the sorted data matrix. Sorting impacts the performance significantly. We tried the following approaches:

1. Sort the features $f_i(x, y)$, $i = 1, 2, \ldots N$ by considering their coordinates (x, y) on the image, in row-major order.
2. Idem, but use column-major order.
3. Sort the features $f_i(x, y)$, $i = 1, 2, \ldots N$ by the Euclidean (L^2) distance to the origin of (x, y).
4. Idem, using a zig-zag scan (essentially, the L^1 distance to the origin of (x, y)).
5. Order the feature vectors by L^2 norm.
6. Order the feature vectors by L^1 norm.
7. Order the feature vectors by L^∞ norm.
8. Cluster the features by proximity and sort them by cluster. Within the same cluster, scan row-by-row.

Clustering the features by proximity leads to good results, since it tends to keep any features that are close to each other on the image close to each other on the sorted data matrix.

Consider an image that includes three features A, B and C, one at position (x, y), another at $(x+1, y)$ and a third at $(x, y+1)$. Row-major or column-major scans of the image are unlikely to preserve the proximity between these three features. At least one of the three features is likely to end up in a distant column of the sorted data matrix. By contrast, when clustering by distance,A, B and C will be assigned to the same cluster with overwhelming probability. Sorting by cluster creates a data matrix in which features close to each other on the image tend to be assigned to columns that are also close to each other on the matrix. The advantages of this will become clear later.

2.3 Finite-Context Encoding

The input to the finite-context encoders is the bitstream obtained by stacking the columns of the sorted, quantized data matrix. A finite-context model provides

Fig. 1. Left: the original image. Center: the manipulated image. Right: The manipulated image, with the edited regions outlined.

an information measure of the number of bits required to represent the current symbol, conditioned on the accumulated knowledge of all past symbols. We use this information to build a complexity profile of the bitstream, in which the bitrate at one point of the bitstream indicates how complex it is. Since any given point in the bitstream can be mapped to a feature vector and hence to an image point, the complexity profile can be related to the complexity of the image itself.

The data are scanned symbol by symbol. When a pattern is found for the first time, the encoder assigns to it a certain complexity, i.e. number of bits needed to represent it. When the pattern is seen again, the number of bits needed will be smaller. The complexity assigned to a pattern therefore depends on the order by which the stream is scanned, a fact that could mask the first occurrences of some patterns.

To remove this dependency, we scan the sorted data twice, once in the forward direction and once in the backward direction. The two complexity profiles obtained are then combined to produce the final complexity profile, $p_i = \min(d_i, b_i)$, where d_i and b_i denote the profiles in the forward and backward directions. This prevents the masking of the first occurrence of a repeating pattern (in forward scans) or the masking of its last occurrence (in backward scans).

To encode the data we use a set of competing finite-context models as described in [6]. The probability estimates provided by each model are averaged using weights updated by a recursive procedure. We used models of depth 3, 5, 8, 10 and 15. As for the parameter α, we took $\alpha = 1$ for the lower order models and $\alpha = 0.05$ for the models of order 10 and 15. A description of all the parameters and their roles can be found in [6].

3 Why Finite-Context Models

The Kolmogorov complexity [7–12] of A is denoted by $K(A)$ and represents the size of the smallest program that produces A and stops. $K(A)$ is not computable, and so it has to be approximated by a computable measure, such as

Lempel-Ziv based complexity measures [13], linguistic complexity measures [14] or compression-based complexity measures [15], which provide approximations and hence upper bounds on the Kolmogorov complexity.

The bitstream produced by a lossless compression algorithm, together with the appropriate decoder, enables the reconstruction of the corresponding original data. Thus, the number of bits required for representing the decoder and the bitstream can be viewed as an estimate of the Kolmogorov complexity of the data. Lossless compression methods thus provide approximations to the Kolmogorov complexity, with better compression algorithms yielding tighter bounds.

Kolmogorov theory can be used to measure object similarity. Li et al. proposed a similarity metric [16] based on an information distance [17], defined as the length of the shortest binary program that transforms A and B into each other, and a practical analog based on standard compressors, called the normalized compression distance [16]. These ideas have been successfully applied in astronomy, genomics, handwritten digits languages, literature, music and virology [18], but are less used in images for one reason. According to Li et al. [16], a compression method needs to be "normal" in order to be used as a normalized compression distance. This means that compressing the concatenation of A with itself should generate essentially the same number of bits as compressing A alone [18]. To satisfy this requirement, the compression algorithm needs to accumulate knowledge about the data as the compression proceeds. It has to collect statistics, i.e., it has to create an internal model of the data.

The Lempel-Ziv algorithms create internal data models and are among the most often used compression algorithms in compression-based complexity applications, including those reported in the imaging field [19–21]. Unfortunately, although they are quite effective for 1D data, they do not perform as well in the case of images or multi-dimensional data. State-of-the-art image compressors, such as JPEG2000 or JPEG-LS, perform better but are not normal. They decorrelate the data using either a transformation or a predictive method, and assume an *a priori* data model that remains essentially static during compression. The decorrelating step destroys most of the data dependencies, leaving to the entropy coding stage the mere task of encoding symbols from an (assumed) independent source. As a result, they cannot be used for conditional complexity estimation. These obstacles lead us to propose compression algorithms based on finite-context models that are both normal and adequate to images [22–25].

The fact that finite-context models are normal and show good performance on images makes them useful to build compression-based image complexity measures. We used them to find unknown (non-transformed) repeated patterns in images [3,4] and again in the present, more challenging application.

4 Results and Discussion

Fig. 1 shows the original image and the manipulated image, formed by copying, translating and rotating a fragment of the image and pasting it back. The regions are not immediately obvious under visual inspection. For convenience, they are outlined in the image on the right.

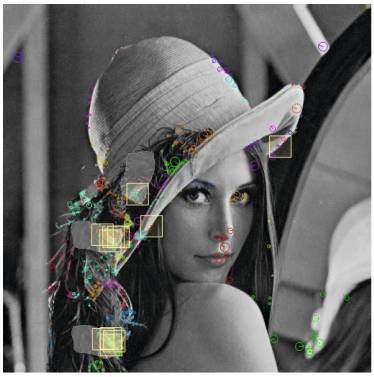

Fig. 2. Detected regions (squares) and SIFT features (circles). The manipulated regions are outlined for convenience only; the input to the algorithm was the image shown in Fig. 1 (center).

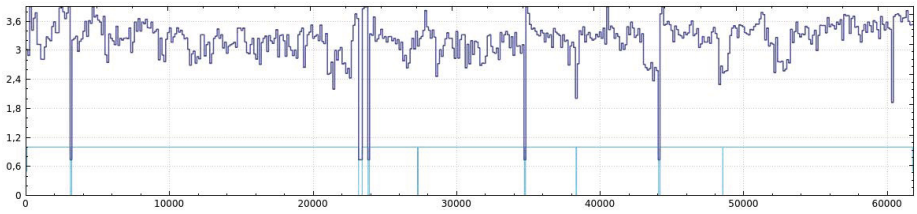

Fig. 3. The median of the complexity profile. The curve below it measures the balance between high and low complexity values in a running histogram of the profile (see text).

Fig. 2 shows the position of the SIFT features as circles (with $\sigma \leq 7$) and the low-complexity regions detected (the squares). In addition to the manipulated regions, the algorithm also marked certain other image regions as similar among themselves. This is unavoidable, since natural images may well contain such regions. Any reasonable algorithm designed to detect unknown repetitions or quasi-repetitions will very likely encounter and report both artificial repetitions, the result of introduced manipulations, and natural repetitions.

Fig. 3 shows the median value of the complexity profile, computed over segments of 128 symbols. This corresponds to the median value of the complexity over each feature (in the case of SIFT, over 128 symbols). The curve on the bottom is obtained by comparing the number of times that in each feature vector the complexity profile assumes small values with the number of times that it assumes large values. By small and large we mean, respectively, the bottom 4 and the top 4 bins in a 10-bin running histogram of the profile.

The quantization is necessary to reduce the number of intensity levels and therefore the alphabet size in the context models. We found that values of about

20 levels are adequate, but found different values (as small as 15) also useful. The sorting step is an important one. The optimal sorting strategy depends on the image and the location of the repetitions, but clustering the features by proximity appears to be the best general strategy. The reason is that it tends to keep features that are close to each other on the image close to each other on the sorted data matrix.

To fully appreciate the impact of this, consider an exact repeat of a region containing n features. Assume that the repeated part also contains n features, and that they are similar to the original ones, as one would expect. Sorting by proximity tends to keep the features of the first region together on the data matrix, forming a set of columns adjacent to each other. The same applies for the repetition. The bitstream will therefore present two identical segments, of size $128n$ symbols each. Other sorting strategies may lead to scattered identical pairs of segments of 128 symbols each, which are harder to detect.

Concerning the context models, we found that it is important to combine a number of models of different lengths, and it is important to use a smaller value of α (typically 0.05) for deeper models (say, above 10). We have obtained good results with 3, 4 and 5 models of depths between 3 and 15. It is important to implement the deeper models using e.g. hash tables, since a context array would require a prohibitive amount of memory.

The computational requirements depend mainly on the number of features and the dimension of each feature vector. An image of average complexity with several hundred features can be processed in a few seconds (this includes the time necessary to extract the features and quantize the data). In the example given there were 483 features and the total computation time did not exceed a couple of seconds, with the algorithm running on a laptop computer.

A limitation of the approach is that it is oblivious to tampering in regions for which SIFT returns no features. This is not a limitation of SIFT, but of the approach itself. Given any other feature extraction algorithm and a region S of the image, if there are no features $f_i(x, y)$ with $(x, y) \in S$, we will not be able to detect repetitions of subsets of S. This becomes more serious as the size of the edited regions decreases, since the probability of a feature lying on a region naturally decreases with its size. In general, however, SIFT seems appropriate and produces a sufficiently rich feature set, for all but the smallest edits. Despite the limitations, we feel that our approach provides an interesting solution to the challenging problem of detecting unknown repeats in images.

References

1. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision **60**(2), 91–110 (2004)
2. Otero, I.R., Delbracio, M.: The anatomy of the SIFT method. Image Processing On Line (2012), http://demo.ipo.im/demo/82
3. Pinho, A.J., Ferreira, P.J.S.G.: Finding unknown repeated patterns in images. In: EUSIPCO 2011, Barcelona, Spain, pp. 584–588 (2011)

4. Pratas, D., Pinho, A.J.: On the detection of unknown locally repeating patterns in images. In: Campilho, A., Kamel, M. (eds.) ICIAR 2012, Part I. LNCS, vol. 7324, pp. 158–165. Springer, Heidelberg (2012)
5. Curtin, R.R., Cline, J.R., Slagle, N.P., March, W.B., Ram, P., Mehta, N.A., Gray, A.G.: MLPACK: A scalable C++ machine learning library. J. of Machine Learning Research **14**, 801–805 (2013)
6. Pinho, A.J., Pratas, D., Ferreira, P.J.S.G.: Bacteria DNA sequence compression using a mixture of finite-context models. In: IEEE SSP 2011, Nice, France, pp. 125–128 (2011)
7. Solomonoff, R.J.: A formal theory of inductive inference. Part I. Information and Control **7**(1), 1–22 (1964)
8. Solomonoff, R.J.: A formal theory of inductive inference. Part II. Information and Control **7**(2), 224–254 (1964)
9. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Problems of Information Transmission **1**(1), 1–7 (1965)
10. Chaitin, G.J.: On the length of programs for computing finite binary sequences. Journal of the ACM **13**, 547–569 (1966)
11. Wallace, C.S., Boulton, D.M.: An information measure for classification. The Computer Journal **11**(2), 185–194 (1968)
12. Rissanen, J.: Modeling by shortest data description. Automatica **14**, 465–471 (1978)
13. Lempel, A., Ziv, J.: On the complexity of finite sequences. IEEE Trans. on Inf. Theory **22**(1), 75–81 (1976)
14. Gordon, G.: Multi-dimensional linguistic complexity. Journal of Biomolecular Structure & Dynamics **20**(6), 747–750 (2003)
15. Dix, T.I., Powell, D.R., Allison, L., Bernal, J., Jaeger, S., Stern, L.: Comparative analysis of long DNA sequences by per element information content using different contexts. BMC Bioinformatics **8**(Suppl. 2), S10 (2007)
16. Li, M., Chen, X., Li, X., Ma, B., Vitányi, P.M.B.: The similarity metric. IEEE Trans. on Inf. Theory **50**(12), 3250–3264 (2004)
17. Bennett, C.H., Gács, P., Li, M., Vitányi, P.M.B., Zurek, W.H.: Information distance. IEEE Trans. on Inf. Theory **44**(4), 1407–1423 (1998)
18. Cilibrasi, R., Vitányi, P.M.B.: Clustering by compression. IEEE Trans. on Inf. Theory **51**(4), 1523–1545 (2005)
19. Tran, N.: The normalized compression distance and image distinguishability. In: Human Vision and Electronic Imaging XII - Proc. of SPIE, p. 64921D (January 2007)
20. Mallet, A., Gueguen, L., Datcu, M.: Complexity based image artifact detection. In: DCC 2008, Snowbird, Utah, p. 534 (2008)
21. Gondra, I., Heisterkamp, D.R.: Content-based image retrieval with the normalized information distance. Computer Vision and Image Understanding **111**, 219–228 (2008)
22. Pinho, A.J., Neves, A.J.R.: Lossy-to-lossless compression of images based on binary tree decomposition. In: IEEE ICIP 2006, Atlanta, GA, pp. 2257–2260 (2006)
23. Pinho, A.J., Neves, A.J.R.: L-infinity progressive image compression. In: PCS 2007, Lisbon, Portugal (2007)
24. Pinho, A.J., Neves, A.J.R.: Progressive lossless compression of medical images. In: IEEE ICASSP 2009, Taipei, Taiwan (2009)
25. Neves, A.J.R., Pinho, A.J.: Lossless compression of microarray images using image-dependent finite-context models. IEEE Trans. on Medical Imaging **28**(2), 194–201 (2009)

http://www.springer.com/978-3-319-11757-7

Image Analysis and Recognition
11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014, Proceedings, Part I
Campilho, A.; Kamel, M. (Eds.)
2014, XXIII, 518 p. 214 illus., Softcover
ISBN: 978-3-319-11757-7