COPY-MOVE DETECTION ALGORITHM EFFICIENCY INCREASE USING BINARY SPACE PARTITIONING TREES

A. Kuznetsov[1,2], E. Myasnikov[1]

[1]Samara National Research University, Samara, Russia
[2] Image Processing Systems Institute - Branch of the Federal Scientific Research Centre "Crystallography and Photonics" of Russian Academy of Sciences, Samara, Russia

Abstract. Duplicates embedding is one of the most frequently used type of digital image change. The copy-move procedure includes copying an image fragment from one part and pasting it to another part of the same image. Moreover, this fragment can be changed using some transformations, like affine or contrast enhancement. Existing copy-move detection methods consist of two main steps: feature calculation in overlapping processing windows and the nearest feature search in Euclidean space using lexicographic sorting or kd-tree search. In this paper we suggest to use binary space partitioning trees on the second step of the detection algorithm – vp-tree (vantage-point tree). We present the search speed comparison using vp-tree and kd-tree. The results show advantage of the proposed solution over kd-tree use.

Keywords: duplicate, copy-move, forgery, kd-tree, vp-tree, binary space partitioning tree

Citation: Kuznetsov A, Myasnikov E. Copy-move detection algorithm efficiency increase using binary space partitioning trees. CEUR Workshop Proceedings, 2016; 1638: 373-378. DOI: 10.18287/1613-0073-2016-1638-373-378

1 Introduction

In the digital age it is impossible to imagine a more popular method of presenting and transmitting information than digital images and video. They are used in all spheres of life, and are applied in many fields of science and technology. Such a huge amount of data cannot be missed by users with malicious intent whose aim is to provide forgery data to the end user. Depending on the image content, it can be used in the compromising form and cause serious political and economic consequences if the data changes will not be detected. That is why in today's world the urgent task is to develop methods and algorithms for artificial digital image changes detection. The urgency of the problem is also due to the dynamic growth of the number of software products for digital imaging. This software does not require special skills or training.

Among all the existing image forgery creation methods one of the most frequently used is to copy and paste an image fragment. Such attacks are called copy-move attacks, and the copied fragment – a duplicate. Between copying and pasting different transformations can be applied to the duplicate: affine (scale, rotation), brightness (contrast enhancement, additive noise), and others. The widespread use of this particular attack is caused by its implementation simplicity.

Nowadays one can find a large number of papers on development of distorted and plain copy-move detection algorithms [1-5]: the common step used in them is the development of some features that are invariant to duplicates distortions that appear during the transformation step. These features are usually calculated in an overlapping processing window mode [2]. Previously there were developed plain copy-move detection algorithms, which were based on the hash values calculation in a sliding window and their frequencies calculation using a hash table [4-5]. The proposed solution has shown high efficiency at a low computational complexity.

In this paper, we assume the operation of feature calculation is solved and focus on the study of the nearest features search procedure. The paper is structured as follows. The first part describes the general scheme of copy-move detection algorithms. The second part is devoted to the binary space partitioning trees description. The third part contains the results of conducted experiments on the computational complexity of the proposed method using a variety of sample volumes.

2 General copy-move detection algorithm scheme

A large number of works devoted to the development of copy-move detection algorithms exists nowadays [1-3]. Most of them adhere the general scheme shown in Fig. 1.

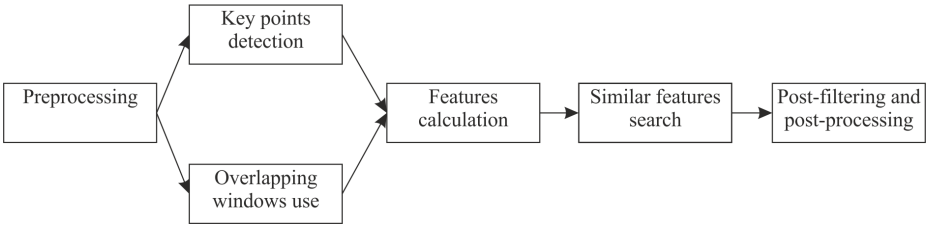

Fig. 1. General scheme of a copy-move detection algorithm

The first stage of the algorithm includes preprocessing of the analyzed image. During this step, noise filtering or image channels combining operation is applied. This step is followed by two alternative procedures. The first is to calculate the key points of the image (for example, in the work [3]), the second characterizes a partitioning scheme of the image into blocks to apply a processing window further. The third step of the algorithm is dedicated to the calculation of features, based on the key points or calculated using pixel values of the processing window according to the previous step. The next stage is to find the nearest features, which are then considered as potential

duplicates. This procedure is the most resource-intensive in the whole scheme of image analysis, therefore, the processing time of the whole algorithm depends on its choice. The final stage of the algorithm is post-filtering and post-processing. This step reduces the number of false detected duplicates and missed duplicates, which obviously increases the accuracy of the copy-move detection algorithm.

Most studies on copy-move detection algorithms development [1-3] pay a lot of attention to the research of new features invariant to various distortions. These features lead to detection accuracy improvement. The task of computational complexity reduce is not considered by most of the authors. The obvious solution to the nearest feature vectors search is their pairwise comparison. This approach is computationally inefficient and has complexity $O(N^2)$, which does not allow to use it for large data analysis, for example, in remote sensing data analysis tasks. To reduce the computational complexity, the great majority of authors use two main methods: lexicographical sorting and kd-tree search.

The lexicographic sorting means that all the feature vector are placed in the feature matrix in a row-wise way, and then the sorting of rows is applied starting with the first feature. As a result, the nearest feature vectors will be located on the adjacent rows of the matrix. An example of this approach is demonstrated in the paper [6].

In most of the studies [7-8] kd-tree [9] is used for the nearest feature vectors search. It is used in conjunction with the k nearest neighbors search algorithm, which allows to generate a list of k nearest vectors to every feature vector in Euclidean space. In comparison with the lexicographic sorting this approach leads to better search results and, consequently, to the detection accuracy increase.

In this paper, we propose to use an alternative binary space partitioning tree – vp-tree [10]. It is assumed that the use of vp-tree will lead to a better analysis time performance in comparison with kd-tree. This assumption is based on the results of previous studies in which vp-tree showed better results in search tasks [10], as well as reducing the size of multi-dimensional data [11].

3 Binary space partitioning trees used in the study

Kd-tree [9] is balanced binary space partitioning tree, the construction of which for a set of vectors is as follows. One axis among all coordinate axes is selected that will be used in partitioning of the set of input vectors. Selection of the coordinate axis can be carried out in a cyclical order (first coordinate then second coordinate, etc.) or based on the maximum variation of the vectors in the set along a selected axis (the latter option was used in this paper).

After selecting a coordinate axis, having subset of vectors we choose vector with a median value along the coordinate axis selected for partitioning. The chosen vector is associated with the root of the tree, and all the other vectors are divided into two subsets. In the first subset we place vectors with values of the corresponding coordinate below the median value. In the second subset we place vectors with values of the corresponding coordinate above the median value. So formed subsets are associated with the left and the right subtrees of the root, respectively. After this, the above pro-

cess is repeated recursively for the left and the right subtrees. Fig. 2a shows an example of kd-tree construction in two-dimensional space.

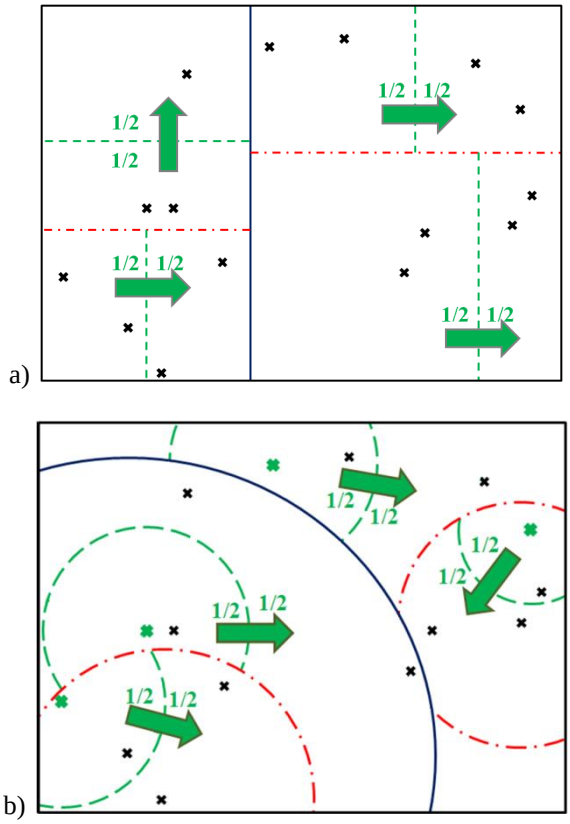

Fig. 2. Binary space partitioning trees in two-dimensional space: (a) kd-tree, (b) vp-tree. Different levels of decomposition are shown with different lines: solid line (dark blue) – first level; dash-dot line (red) – second level; dash line (green) – third level

Vp-tree [10] is also a balanced binary space partitioning tree. Its construction for a set of vectors is as follows. We choose one vector from a set of vectors (vantage point), which becomes the root of the tree. Then all the other vectors are divided into two subsets, so that the first subset contains vectors located by a distance not exceeding a value of R from the vantage point, and the second subset contains vectors located by a distance greater then R from the vantage point. The threshold value R is selected so that the number of elements in the first and the second subsets differed by no more than one. Formed in this way subsets are associated with the left and the right subtrees of the root, respectively. Described process is repeated recursively for the left and the right subtrees. Fig. 2b shows an example of vp-tree construction in two-dimensional space.

Trees built using the algorithms described above can be used to find the nearest neighbor using the branch and bound algorithm as described in [9, 10]. It should be noted that the considered structures perform decomposition in essentially different ways: with hyperplanes orthogonal to the coordinate axes in the kd-trees and with hyperspheres with centers at the vantage points in the vp-trees.

4 Experiments

For experimental study, we used a standard PC (Intel Core i5-3470 3.2. GHz, 4 Gb RAM). The algorithms were implemented in C ++ using Microsoft Visual Studio 2013 development environment. The objects of the study are feature sets of different size containing from 5,000 to 15,000 vectors.
In the experiment, for each feature vector we search the nearest vector among the remaining vectors of the sample. Upon completion of the algorithm, a list containing pairs of indices of closest vectors is generated, and the run time of the search algorithm is evaluated in seconds. Comparison of the search time is conducted for the three solutions: elementwise comparison of feature vectors (brute force), kd-tree which is the most commonly used structure in the duplicate detection problem, and vp-tree which is the binary space partitioning tree proposed to be used for the duplicate detection in images in this paper. Results of the experiments shown in Table 1, show that the proposed approach performs search for several seconds faster than the kd-tree, and significantly faster than the brute force approach with increasing sample size.

Table 1. Comparison of the nearest feature vector search time (sec.) for different sample sizes from 5,000 to 15,000

	5000	7500	10000	12500	15000
brute force	9,6	22,7	39,0	62,0	90,0
kd-tree	5,7	13,4	24,5	36,3	49,0
vp-tree	4,6	11,0	22,1	34,6	47,2

5 Conclusion

In this study, an alternative approach using vp-trees was proposed to search for similar feature vectors in the problem of duplicate detection in digital images. The proposed solution shown best nearest vector search time in comparison with the most frequently used solution based on kd-trees. It is planned to conduct a study on the effectiveness of different vantage point selection algorithms to solve the problem of duplicate detection in digital images.

Acknowledgements

This work was supported by the Russian Foundation for Basic Research (RFBR) grants № 16-37-00056 мол_а and № 16-37-00202 мол_а.

References

1. Christlein V, Riess C, Jordan J, Angelopoulou E. An evaluation of popular copy-move forgery detection approaches. IEEE Trans. Inf. Forensic Secur, 2012; **7**(6): 1841–1854.
2. Popescu A, Farid H. Exposing digital forgeries by detecting duplicated image regions. URL: http://www.ists.dartmouth.edu/library/102.pdf
3. Fridrich J, Soukal D, Lukas J. Detection of copy-move forgery in digital images. URL: http://www.ws.binghamton.edu/fridrich/Research/copymove.pdf
4. Glumov N, Kuznetsov A, Myasnikov V. The algorithm for copy-move detection on digital images. Computer Optics, 2013; 37(3): 360–367.
5. Vladimirovich KA, Valerievich MV. A fast plain copy-move detection algorithm based on structural pattern and 2D Rabin-Karp rolling hash. In: Campilho A, Kamel M. (eds.) ICIAR 2014, Springer, Heidelberg, 2014, Part I. LNCS, 8814: 461–468.
6. Bayram S, Sencar H, Memon H. An efficient and robust method for detecting copy-move forgery. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, 2009: 1053–1056.
7. Huang H, Guo W, Zhang Y. Detection of copy-move forgery in digital images using SIFT algorithm. In: Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008; 2: 272–276.
8. Ju S, Zhou J, He K. An Authentication Method for Copy Areas of Images. International Conference on Image and Graphics, 2007: 303–306.
9. Bentley JL. Multidimensional binary search trees used for associative searching. Communications of the ACM, 1975; 18(9): 509.
10. Yianilos: Data structures and algorithms for nearest neighbor search in general metric spaces. Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, 1993: 311–321.
11. Myasnikov EV. Evaluation of Space Partitioning Data Structures for Nonlinear Mapping. 23rd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2015 – Full Papers Proceedings, 2015: 109-118.