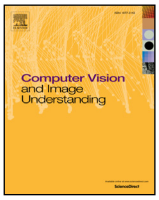
Computer Vision and Image Understanding 154 (2017) 35–47

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu



Pair-wisely optimized clustering tree for feature indexing

The-Anh Pham

Hong Duc University, Thanh Hoa city, Vietnam

ARTICLE INFO

Article history:
Received 9 February 2016
Revised 25 May 2016
Accepted 29 July 2016
Available online 1 August 2016

Keywords:
Feature indexing
Fast nearest neighbour search
Clustering tree
Hierarchical Voronoi diagram

ABSTRACT

This paper presents a new approach for indexing real feature vectors in high dimensional space. The proposed approach is developed based on Pair-wisely Optimized Clustering tree (POC-tree) that exploits the benefit of hierarchical clustering and Voronoi decomposition. The POC-tree maximizes the separation space of every pair of clusters at each level of decomposition, making a compact representation of the underlying data. Searching in the POC-tree is efficiently driven by the bandwidth search strategy. A single POC-tree can be used to create effective index of data for both exact and approximate nearest neighbour search. We also present a new method to combine multiple weak POC-trees for boosting the search performance for specific datasets in very high dimensional space. Extensive experiments have been conducted to evaluate the proposed approach in which it outperforms the state-of-the-art methods for all the datasets used in our experiments.

1. Introduction

Fast feature matching is a crucial need to support time-critical image processing systems. Although the processing time can be lessened by reducing the feature dimensionality, this process may not give a significant improvement in search performance. Moreover, in many computer vision applications, the feature dimensionality must be sufficiently high to make the descriptors highly distinctive for differentiating thousands of objects. This requirement raises an expensive computation cost for the subsequent steps of matching the feature vectors and object retrieval/recognition. Hence, a promising solution for fast matching of real feature vectors would be the incorporation of an efficient indexing scheme. Basically, an indexing algorithm is formulated as the task of reorganizing the data in such a way that it is able to produce swiftly the answers for the queries of exact nearest neighbour (ENN) search and/or approximate nearest neighbour (ANN) search.

Although many indexing algorithms have been introduced in the literature, we are aware of a few methods that are shown to perform well in practice. Such methods have been known as randomized KD-trees (Muja and Lowe, 2009), hierarchical K-means tree (Muja and Lowe, 2009) and randomized clustering trees (Muja and Lowe, 2012). The rationale to make these methods being effective for fast searching is two-fold: the way of building the hierarchical index tree(s) and the way of traversing the tree(s). For the former aspect, the use of randomized trees to account for the multifold domain of the data has indeed given a significant improvement in search performance. Coupling with the construction of randomized trees, priority search strategy, which explores the nodes in the increasing order of their distance to the query, has shown quite good results for the later aspect of tree traversing.

In this work, we introduce a new approach for indexing real feature vectors. The proposed approach exploits the advantages of hierarchical clustering and Voronoi decomposition. It creates a compact representation for the cluster volumes by pair-wisely maximizing the separation space of every pair of clusters at each level of decomposition. The resulting index tree is thus so-called pair-wisely optimized clustering tree (POC-tree). Searching in the POC-tree is efficiently driven by a bandwidth search strategy which is efficient to prune unnecessary search space while searching for nearest neighbors (Pham et al., 2015). The POC-tree can be used alone to create effective index of data for both ENN and ANN search tasks. It is also possible to create and combine multiple weak POC-trees for boosting the search performance for vector data in very high dimensional space. We compare the POC-tree(s) indexing algorithm with the aforementioned methods and show the superiority of the proposed approach for all the datasets used in our experiments.

The proposed indexing method has some close relation with other methods. First, it is closely connected to the hierarchical K-means trees introduced in Muja and Lowe (2009); 2012); Nister and Stewenius (2006) but is different from them with a compact representation of the underlying data. Second, the proposed bandwidth search procedure is somewhat similar to the priority search strategy (Beis and Lowe, 1997; Muja and Lowe, 2014) but avoids maintaining a priority queue and hence is more

E-mail address: phamtheanh@hdu.edu.vn

http://dx.doi.org/10.1016/j.cviu.2016.07.011
1077-3142/© 2016 Elsevier Inc. All rights reserved.

efficient. Finally, the use of multiple weak POC-trees for improved performance is somehow related to multiple randomized KD-trees (Silpa-Anan and Hartley, 2008). Unlike them, each weak POC-tree is not randomly created but is characterized by a parameter of branching factor.

The rest of this paper has been organized as follows. Section 2 reviews the main approaches in the literature for feature indexing. Section 3 is devoted to the presentation of the proposed approach. Experiments and discussions are provided in Section 4. We conclude the paper and discuss about our further work in Section 5.

2. Related work

A large number of methods have been proposed in the literature to deal with the problem of fast nearest neighbor search. In this work, we restrict our discussion to the most representative methods devoted to work in the vector space. These methods may be grouped into three main classes: space partitioning, clustering and hashing.

2.1. Space-partitioning-based methods

As stated before, we shall mention in this section the most representative tree-based techniques that have been particularly designed to work efficiently in the vector space. We therefore avoid the discussion of other tree structures that can generally work in both metric space and vector space such as metric trees and spill trees (Liu et al., 2004; Navarro, 2002).

The main spirit of the space-partitioning-based methods is to hierarchically divide the underlying data space into sub-spaces in such a way that the cardinality of every sub-space at the same level is roughly equivalent. The partitioning process is repeated until the cardinality of every new sub-space at the highest level is small enough. A tree-based structure is often employed to represent the resulting sub-spaces at all levels of decomposition. Doing so, the obtained tree has a nice property of balance which is important for efficient searching. Different techniques have been presented to achieve such a balance decomposition. The most popular technique, KD-tree (Friedman et al., 1977), uses a hyperplane perpendicular to one of the coordinate axes for splitting data without considering the intrinsic distribution of the underlying data. Multiple rotated KD-trees (NKD-trees Silpa-Anan and Hartley, 2008), are constructed by pre-rotating the original data by different angles. In a similar fashion, multiple randomized KD-trees, (RKD-trees Silpa-Anan and Hartley, 2008), are constructed in which each tree is built by selecting randomly, at each node, a split axis from few dimensions having the highest variance.

There are also growing interests in applying principal component analysis (PCA) technique to perform space partitioning. The principal axis tree (PA-tree) (McNames, 2001) partitions the data via the narrowest direction by applying principal component analysis at each level of decomposition. The LM-tree (Pham et al., 2013) suggests using the two highest variance axes in the PCA space for data partitioning. Experiment results carried out in these works have shown the potential performance of PCA-based trees. A recent evaluation (Verma et al., 2009) of different tree-based structures also reported that principal direction (PD) tree performs consistently better than many other tree structures for ANN search.

While most of previous attempts suggest using only one coordinate axis to perform space splitting at each sub-level of the tree, a recent work (Pham et al., 2013) proposes to use the two highest variance axes in the PCA space for data partitioning. The combination of the two axes has shown promising results for various datasets. Interestingly, the same spirit of using multiple coordinate axes for data partitioning has also been exploited in Wang et al. (2014b). In this work, the authors proposed combining many coordinate axes in a linear fashion with trinary-valued weights to create a so-called trinary-projection (TP) tree. In addition, they also adopted the maximum variance criterion for ensuring compact space partitioning and exploited the benefit of using multiple randomized TP trees for improving search performance.

Once the hierarchical indexing trees are created, fast proximity search for nearest neighbors is often driven by using a branch-and-bound technique (Friedman et al., 1977; McNames, 2001; Pham et al., 2013). At each intermediate node u of the search path, a lower distance bound is computed from u to the query Q. If the lower bound exceeds the distance from Q to the nearest point found so far, it is unnecessary to explore u and we can proceed with other nodes. A noticeable improvement for driving the search process is known as priority search (Beis and Lowe, 1997; Muja and Lowe, 2014). The basic idea is to explore the nodes in the increasing order of their distances to the query. Doing so, a priority queue is created to maintain the branches which are closer to the query at every level of the search path. The search process terminates early once a pre-determined number of data points have been visited. Although the priority search strategy was shown to give a significant improvement in search performance, the process of maintaining a priority queue during the online search is rather expensive and can degrade the search performance for specific datasets.

2.2. Clustering-based methods

The clustering methods iteratively divide the data into sub-clusters by using a clustering method such as K-means (Muja and Lowe, 2009), K-medoids (Muja and Lowe, 2012) and agglomerative clustering method (Leibe et al., 2006). The decomposition process continues until the sub-clusters are small enough. Searching in the hierarchical tree is proceeded by traversing down the tree until a leaf node is reached. At each intermediate level, the node with the center closest to the query is chosen to be next explored. Multiple randomized clustering trees have also been exploited for efficiently indexing of the data (Muja and Lowe, 2012). In addition, priority search strategy has been successfully incorporated in these works to give a dramatic improvement in search speedup (Muja and Lowe, 2009; 2012; 2014).

Much of effort in ANN search has been paid to product quantization (PQ) techniques that can be considered as a hybird approach of space-partitioning and clustering. The potential advantage of a product quantization method is the capability of building the index for very large-scale and high-dimensional datasets. The underlying idea is to decompose the original data space into distinct sub-spaces and then to create a separate codebook for the data contained in each sub-space (Jegou et al., 2011a). The inverted file technique is often incorporated to make the search process more effective. Since the successful demonstration of the PQ technique for ANN search, many attempts (Babenko and Lempitsky, 2014; Ge et al., 2014; Kalantidis and Avrithis, 2014; Norouzi and Fleet, 2013; Zhang et al., 2015) have been investigated to optimize the quantization technique by different ways. The optimized product quantization (OPQ) method (Ge et al., 2014) suggests optimizing the space decomposition process so as to be better fitting to the underlying data. This work, however, is less-efficient to the cases that underlying data exhibit a multi-model distribution (Kalantidis and Avrithis, 2014). To address this mater, local optimization of space decomposition was presented in Kalantidis and Avrithis (2014). Nevertheless, learning locally an optimized PQ using a parametric fashion requires some prior knowledge about the underlying distribution (*e.g.,* typically a Gaussian distribution). In practice, such a prior knowledge is not always available.