IMPROVED EARTH OBSERVATION DATA RETRIEVAL THROUGH HASHING ALGORITHMS

A.-C.Grivei[1], C.Văduva[1],and M.Datcu[12],

[1]University Politehnica of Bucharest (UPB), Romania
[2]Remote Sensing Technology Institute, German Aerospace Center (DLR), Germany

ABSTRACT

Throughout the years, a wide range of satellite mission enabled the creation of a huge amount of Earth Observation (EO) data carrying complex information, whose exploitation is left behind due to the lack of handling capabilities. Computational resources are hardly keeping up with content analysis and information retrieval. In order to increase the search speed through data warehouses for knowledge discovery, new indexing methods are required to handle both the size and the informational complexity of EO data. Feature extraction algorithms are able to describe the image content, yet, they require a very complex database. In this paper, we propose a methodology that combines feature extraction, hashing methods and optimized indexing to convert the images characteristics into hash codes in an effort to speed up the search process. For our experiments, we run our procedure on a data-set composed of several Sentinel-2 acquisitions form across Europe and we assess the query times.

Index Terms— image hashing, Earth Observation, database queries, content based image retrieval

1. INTRODUCTION

Data provided by the multitude of EO satellite missions, such as ones coordinated by USGS (United States Geological Survey) or ESA (European Space Agency), sums up to vast quantities. The wide variety of sensors produces data with variable spatial and spectral resolutions, and use different acquisition methods (passive, active). Finding an efficient method to access all the data can be cumbersome. In order to store and index the data in such a manner in which the contained information can be easily and rapidly retrieved, any proposed solution must be highly scalable as the active EO missions provide increasing amounts of data.

In order to optimize the retrieval time for the data, hashing algorithms can be employed to map the high-dimensional feature vectors to a compact binary hash code which can be structured into an index table. Two kernel-based nonlinear hashing methods for remote sensing are introduced in [1]. First one (KULSH, kernel-based unsupervised locality-sensitive hashing) uses unlabeled images to define hash functions while the second (KSLSH, kernel-based supervised hashing) relies on semantics extracted from annotated images. They use the BOVW (Bag of Visual Words) method obtained from 100.000 random SIFT (scale-invariant feature transform) descriptors. The hashes are searched through a linear scan and both retrieval time and storage costs are improved.

The iterative quantization (ITQ) algorithm is introduced in [2] as a solution to create similarity-preserving binary codes for efficient similarity search in large-scale image collections. The algorithm uses multi-class spectral clustering and use the CIFAR data set [3] containing 32x32 pixels multimedia images.

A product quantization method for approximate nearest neighbor search is presented in [4]. Combined with an inverted file system provides high efficiency. The scalability of the approach was tested on SIFT and GIST descriptors and validated on a two billion vectors dataset.

PRH (Partial Randomness Hashing) [5] proposes random projections to map the images to a lower Hamming space in a data-independent manner; then a transformation weight matrix is learned based on the training data set of 28x28 pixels SAR (synthetic aperture radar) images. Finally the Hamming ranking search outperforms: mean average precision (MAP), precision of the top K returned examples and precision-recall curves.

In this paper, we present a fast content based image retrieval (CBIR) method for huge volumes of EO data. The workflow is based on a combination of feature extraction, image hashing and database indexing. Through this method the size and complexity of the database used to store EO can be reduced. Also the average query time is just under 0.4 s for over 20 million stored samples (binary codes). The paper is structured as follows. Section two presents the tested EO data set (Sentinel-2). The proposed method is detailed in Section three, while the overall results are offered in Section four. The conclusions regarding the proposed workflow are presented in the last Section.

A. C. Grivei e-mail: alex.grivei@ymail.com
M. Datcu e-mail: mihai.datcu@dlr.de.

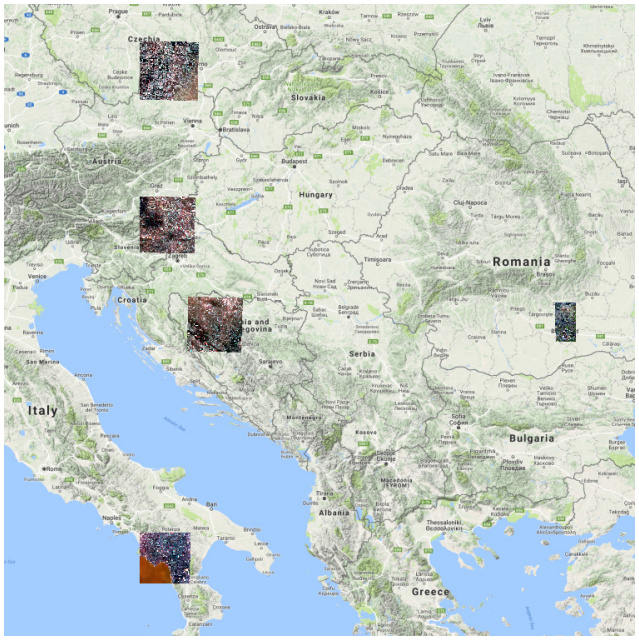
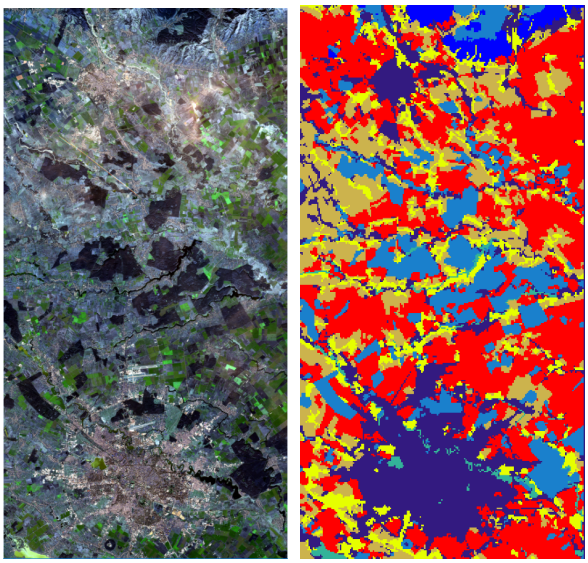


Fig. 1: Test data set composed of four $109.8km \times 109.8km$ Sentinel-2 images which cover portions of South Italy, North-West Bosnia, Est Slovenia, South-Est Austria, Central-South Czech Republic and a $40km \times 78km$ image from the Southern part of Romania spanning between the cities Bucharest and Ploiești.



(a) (b)

Fig. 2: a) Training data set: the smallest image of our data set, the Sentinel-2 image over Bucharest, Ploiești and the geographical area between the two cities. b) Reference map containing 7 classes (manual annotation): High Density Population, Forest, Water, Fragmented Agriculture, Low Density Population, Agriculture, Mountain.

2. USED DATASET

In order to test our method, we built a data set containing five Sentinel-2 images [6] from different countries across Europe, as seen in Fig. 1. We chose to do so in order to have a high diversity of land cover classes and to make it easier to extrapolate the resulted dataset to the entire surface of the European continent. The Sentinel-2 satellites have multiple sensors on board and acquire data over a wide spectral range and at three spatial resolutions 10 m, 20 m and 60 m, each acquisition covering approx. 12.000 sq km. In order to determine the maximum amount of spatial information and to define and extract as many semantic classes as possible we kept the four 10 m bands (B2, B3, B4, and B8). As such, the data set is composed of 4 images (sized of $109.8km \times 109.8km$) acquired over South Italy, North-West Bosnia, Est Slovenia, South-Est Austria, Central-South Czech Republic and a smaller image (sized of $40km \times 78km$) which covers the Southern part Romania spanning between the cities Bucharest and Ploiești. We used all the images to build our encoder, but only used the image over Romania for our tests. The image was manually annotated as seen in Fig. 2. We considered a set of seven semantic classes, as one can usually distinguish in medium resolution EO acquisitions: High Density Population, Forest, Water, Fragmented Agriculture, Low Density Population, Agriculture, and Mountain.

For our initial tests we divided the five images in 5×5 pixel (or $50m \times 50m$) patches across all four bands. From each patch we applied different feature extraction algorithms which are presented in Section three. Considering the size of each image, 1.248.000 patches are generated for our training image while for each of the other four larger images, a number of 4.822.416 patches are generated resulting in a total of 20.537.664 patches. These were all ingested in the database used for the tests.

3. USED METHOD

The overall goal of our method is to reduce the dimensionality of the extracted feature vectors and improve the retrieval times from huge databases. We try to accomplish this through a combination of feature extraction algorithms, hash code generation and optimal indexing methods for the obtained hashes. The overall work flow of our method is presented in Fig.3 and contains two main phases.

The first phase starts with the extraction of fixed sized patches from the entire dataset. From each patch feature vectors are obtained. These are used to build an encoder for the generation of binary hash codes. The resulted hashes are used to build a binary index tree which further improves the query times, by reducing the number of tested hash codes.

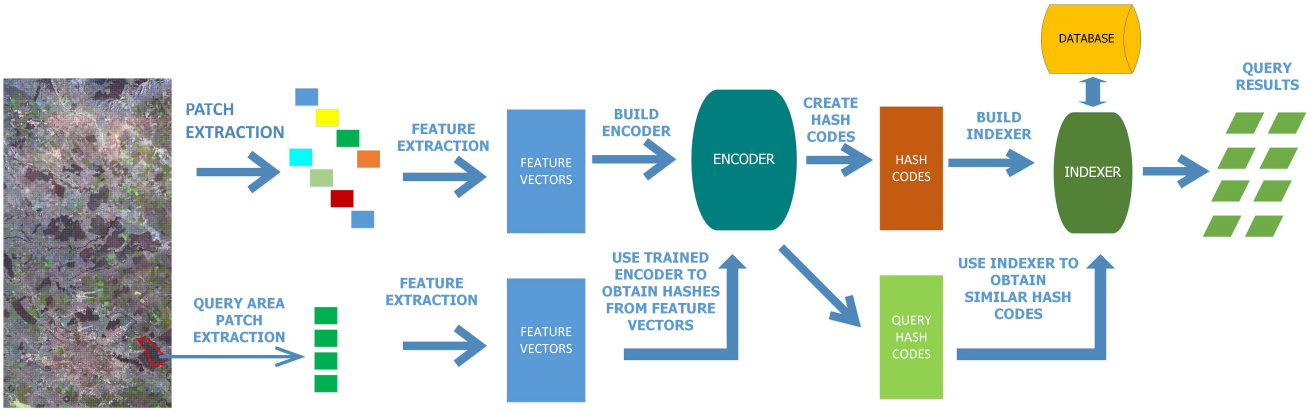


Fig. 3: Method flow chart. The top data flow represents the creation of the encoder and goes from left to right. It starts with the extraction of application defined patch sizes, followed by feature extraction algorithms. The hashing encoder uses the obtained feature vectors to build an optimal transformation from feature vectors to hash codes. Based on the generated hashed an indexing system is created in order to improve data extraction, and data is added to the database. The bottom data flow is used during the CBIR phase. From left to right, the zone of interest is selected and decomposed in an optimal number of patches with the same size as the ones used in the previous phase. Feature vectors are extracted, hashes are obtained using the previously created encoder, and the hash codes are used for the database interrogations.

The second stage of the method envisages a CBIR approach starting with the selection of an area of interest. The zone is divided in the patches which are fully contained in the selected area. The encoder resulted in the previous stage is used to generate appropriate hash codes. These in turn are used to query the database and obtain the resulted patches.

For our initial test we used a feature vector represented by a combination of spectral indices for water (MNDWI - Modified Normalized Difference Water Index), vegetation (NDVI - Normalized Difference Vegetation Index) and build up (NDBI - Normalized Difference Build-Up Index). The spectral indexes are computed using their specific methods for each pixel in a patch. The size of the resulted feature vector, for a patch of $5 \times 5 pixels$, is 75 (25 MNDWI values + 25 NDVI values + 25 NDBI values).

For the hash encoder we used the PQI (Product Quantization Indexer) algorithm [7] which was provided by the HDIdx Python library [8]. The purpose of the algorithm is to reduce the size of a feature vector through quantization, which is a destructive process. It reduces the size of a feature vector from 75 double (64 bit) values to just 64 bits. The compact binary codes resulted from the compressed original feature vectors are indexed in the database using a binary tree. In order to improve the query results return time we perform approximate nearest neighbor (NN) searches instead of exact NN searches.

4. EXPERIMENTS AND RESULTS

In order to verify our method, we used the dataset presented in Section two. The entire data set was used to train the encoder with a wide range of possible semantic classes. The image over Bucharest has been used in the performance analysis phase. All of the algorithms were verified using an Intel i7 3.6GHz quad core, eight thread machine with 32GB of RAM. This enabled us to use the entire data set and load all the feature vectors in memory.

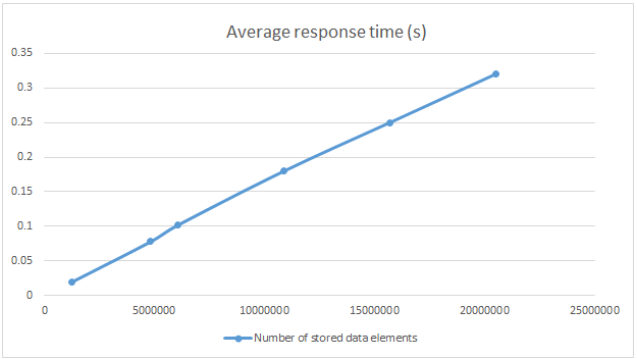


Fig. 4: Average response time for a query with a database of different sizes.

In order to verify the scalability of the method we made a set of experiments with variable number of ingested hashes and we computed the returned accuracies for each of the seven annotated classes. The average response time needed to return the first 100 elements from the database for a query is given in Fig. 4. As it can be observed, the method has a linear dependency from 1.2 million to 20 million samples in the database. The algorithm was run using Python which can have some speed impediments and the presented times can be

Table 1: Average accuracy per class

Queries	Response time (s)	Average Accuracy	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
1	0.018957125	52.57142857	60	100	40	20	20	100	20
5	0.090606	55.77142857	56.8	86	94.4	33.2	23.2	87.2	9.6
10	0.186791625	60.8	68	96.2	77.6	45.6	57.2	56.4	24.6
20	0.356107625	55.94285714	62.8	95.6	64.9	41.9	35.8	76.7	13.9
40	0.70321075	57.29444444	57.1	95.6	66.35	35.55	29.45	60.2	18.7
80	1.405936	56.49642857	59.85	95.65	82.75	43.025	31.925	66.625	15.65

further improved by rewriting the algorithm in C or cuda.

We also analyzed the average returned accuracy for each of the annotated classes. The overall results are presented in Table 1. We tested the framework on parallel queries ranging from 1 to 80. The average accuracy is around 55% but this is due to the fact that some of the classes performed very poorly in comparison to the rest. The best performing class was the Forest class with an average accuracy of 94% while the worst was the Mountain class with an average accuracy of 17%.

The overall results can be further improved by increasing the size of the patch in order to offer more information for the feature extraction algorithms. The purpose is to extend the experiments to different patch sizes, feature extraction algorithms, hashing algorithms and indexing methods. Although, the small patch size of 5×5 pixels has generated a large amount of data elements for our tests, it is clearly not suited to distinguish between the annotated semantic classes.

5. CONCLUSIONS

Storing feature vectors in databases can be cumbersome and searches based on distance computations can be very slow. Hashing promises to be a good method to improve the retrieval time for content based queries form databases. Combined and rightfully adjusted, they can provide enhanced performances in terms of speed and relevance for the target classes.

The processing overhead is increased in the ingestion phase when each added element needs to have its hash code computed and added to the database. In turn, the database search times are reduced in the query phase due to the simplified computations needed.

The quality of the hash encoder is proportional to the size of the training dataset. Considering the total amount of data generated by the Sentinel-2 mission and the overall variety of classes, the training process can be an issue due to the fact that the training data must be as exhaustive as possible.

6. REFERENCES

[1] B. Demir and L. Bruzzone, "Hashing-based scalable remote sensing image search and retrieval in large archives," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 892–904, 2016.

[2] A. Gordo Y. Gong, S. Lazebnik and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2916–2929, 2013.

[3] A. Krizhevsky, "Learning multiple layers of features from tiny images," *technical report, Univ. of Toronto*, 2009.

[4] M. Douze H. Jegou and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 117–128, 2011.

[5] P. Li and P. Ren, "Partial randomness hashing for large-scale remote sensing image retrieval," *IEEE Geoscience and Remote Sensing Letters*, pp. 464–468, 2017.

[6] "Sentinel-2 product specifications," *https://sentinel.esa.int/documents/2479 04/685211/Sentinel-2-Products-Specification-Document*.

[7] M. Douze H. Jegou and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 117–128, 2011.

[8] "Hdidx library," *https://pypi.python.org/pypi/hdidx/*.