RapidEarth
A Search-by-Classification Engine for Large-Scale Geospatial Imagery
Lülf, Christian; Martins, Denis Mayr Lima; Salles, Marcos Antonio Vaz; Zhou, Yongluan; Gieseke, Fabian

Published in:
31st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ACM SIGSPATIAL GIS 2023

DOI:
10.1145/3589132.3625601

Publication date:
2023

Document version
Peer reviewed version

Document license:
CC BY-NC-ND

Citation for published version (APA):
RapidEarth: A Search-by-Classification Engine for Large-Scale Geospatial Imagery (Demo Paper)

Christian Lülf  
University of Münster  
Münster, Germany  
christian.luelf@uni-muenster.de

Denis Mayr Lima Martins  
University of Münster  
Münster, Germany  
denis.martins@uni-muenster.de

Marcos Antonio Vaz Salles  
Independent Researcher  
Portugal  
msalles@acm.org

Yongluan Zhou  
University of Copenhagen  
Copenhagen, Denmark  
zhou@di.ku.dk

Fabian Gieseke  
University of Münster  
Münster, Germany  
fabian.gieseke@uni-muenster.de

ABSTRACT
Data exploration and analysis in various domains often necessitate the search for specific objects in massive databases. A common search strategy, often known as search-by-classification, resorts to training machine learning models on small sets of positive and negative samples and to performing inference on the entire database to discover additional objects of interest. While such an approach often yields very good results in terms of classification performance, the entire database usually needs to be scanned, a process that can easily take several hours even for medium-sized data catalogs. In this work, we present RapidEarth, a geospatial search-by-classification engine that allows analysts to rapidly search for interesting objects in very large data collections of satellite imagery in a matter of seconds, without the need to scan the entire data catalog. RapidEarth embodies a co-design of multidimensional indexing structures and decision branches, a recently proposed variant of classical decision trees. These decision branches allow RapidEarth to transform the inference phase into a set of range queries, which can be efficiently processed by leveraging the aforementioned multidimensional indexing structures. The main contribution of this work is a geospatial search engine that implements these technical findings.

CCS CONCEPTS
• Information systems → Search engine indexing. • Computing methodologies → Classification and regression trees.

KEYWORDS
search engine, decision trees, classification, index structures

1 INTRODUCTION
The tasks of geospatial imagery analysis and exploration are confronted with an ever-increasing challenge of managing and mining vast amounts of data. This surge in data volume is notably attributed to the advancement in satellite technology and the frequency of image capture [6]. Given this growth in data, there is a pressing need for tools that empower users to navigate through these massive amounts of data quickly.

In response to this need, a very recent and promising approach has emerged that comprises a variant of decision trees called decision branches, bundling together machine learning inference with database index structures [8]. This method enables the fast identification of objects of interest within large data catalogs in a matter of mere seconds. However, it remains unclear how such an approach can be integrated into a seamless geospatial search experience. Ideally, departing from only a few labeled images as a reference point, users should be able to readily locate similar objects in satellite imagery—for instance, identifying areas of forest degradation within the Amazon.

In this paper, we present RapidEarth, a system that leverages decision branches to construct a fast and effective geospatial search engine for satellite imagery. RapidEarth is the first-ever prototype to integrate decision branches in an end-to-end application, demonstrating their applicability in a large-scale setting. RapidEarth features a user-friendly graphical web interface where users can effortlessly navigate across aerial imagery and interactively define their search queries by pointing and clicking on a map.

So far, previous geospatial search engines on image similarity have traditionally depended on k-nearest neighbor approaches, which return the k closest images relative to the chosen input image [1, 4, 7, 9]. Despite their fast query time (i.e., time to produce results once a user query is issued), query results may lack comprehensiveness due to inherent limitations of the underlying methods: The returned images are restricted to the top k neighbors, and the query intent can be defined by only a single item. By contrast, using decision branches, we can transform a search task into a binary classification problem, which allows us to train a machine learning model using multiple objects of interest (e.g., class label 1) and even include dissimilar objects that should be excluded from the query results (i.e., class label 0), enabling more precise delineation between the objects of interest and the rest. Thereby, we expect query results to be more precise and complete in comparison to previous...
search engines. By coupling the search model with index structures, we ensure that the query response time remains competitive to k-nearest-neighbor-based search solutions.

Contributions. Key contributions of this work are:

(i) We present a prototype of our RapidEarth search engine that implements a search-by-classification approach with decision branches for an aerial data set (see Figure 3). The prototype is publicly available at https://web.rapid.earth.
(ii) Through a set of demonstration scenarios, we illustrate how RapidEarth can be used to locate target objects, such as solar panels, in rich satellite imagery. The selected demonstration scenarios will be the basis for discussions with the attendees regarding the underlying decision branches technique and its implementation in RapidEarth.
(iii) We also provide the source code that allows other researchers to (re)use the search engine for their own data and use cases. The code repository is publicly available at https://github.com/decisionbranches/rapidearth.

The rest of the paper is structured as follows. Section 2 describes the workflow of our search engine. Section 3 elaborates on the data set used for our prototype. We delve into the system architecture of RapidEarth in Section 4. Subsequently, Section 5 outlines the application of RapidEarth through a practical demonstration including the search of solar panels over the country of Denmark. The paper ends with a short summary in Section 6.

2 APPROACH

The workflow of processing a query with our search engine is sketched in Figure 1. The workflow is separated into two phases (1) Offline Preprocessing and (2) Query Processing. In the offline phase, executed prior to the search engine’s launch, we pre-built the index structures needed to speed up the user query. Here, we first extract meaningful features from the data (in this case satellite imagery data) that capture some characteristics of the underlying data structures (e.g., colors, detected structures like streets, houses, etc.). How we extracted the features is described in Section 3 in more detail. Given the extracted features of all data, we start building the index structures which are based on k-d trees [2].

The query processing phase represents the actual procedure through which user queries are handled within our search engine. Given an incoming user query, defined by positive and negative instances, we gather the corresponding feature vectors for the data and train our decision branches model to separate these two classes from each other. The positive classes in the feature space are represented by multidimensional boxes (characterized by orthogonal boundaries along the axes) and can therefore be translated into range queries (in other words: SQL queries, in which predicates within the WHERE clause represent model decisions; see Figure 2). Our pre-built index structures enable efficient processing of these range queries, returning the target objects from the database during the inference phase. An essential characteristic of our decision branch classifier is its “index-awareness.” This implies that the multidimensional boxes generated by the decision branches correspond to the same feature subsets covered by one of our pre-built index structures such that all queries can be answered by only using the index structures.

3 DATA SET AND DATA PREPROCESSING

In our prototype, we showcase our search-by-classification approach on an aerial data set from Denmark of the year 2018 in a resolution of 12.5 cm per pixel. We split the data set into patches of the size 400x400 pixels, with a step size of 200 pixels. In total, this yielded 90,429,772 patches that can be selected for various search tasks. In order to translate the image patches into meaningful feature vectors that can be used to train our decision branches, we used Vision Transformers (ViT) [5] as feature extractor. We trained a ViT-T model using the self-supervised learning framework DINO [3] on a random subset of 400,000 patches of our aerial data set of Denmark for 100 epochs on one GPU RTX 3090.2 By employing self-supervised training, the ViT model learns intrinsic data characteristics without being directed toward a specific classification task. This strategy ensures that the resulting features are versatile and applicable to an array of search tasks, rather than being confined to predefined classes. After training the ViT, we extracted 384 features per patch from the final layer of the feature extraction model which in total consumed around 130 GB of storage for the entire data set.

4 SYSTEM OVERVIEW

The architecture of our search engine comprises three major components as shown in Figure 4. These are:

---

1While various alternative index structures are available for efficiently handling range queries, we have chosen to implement k-d trees in our prototype due to its straightforward implementation.

2The feature quality and thereby also the quality of the query results can be even further improved by increasing the size of the training set as well as the model size.
Before the search engine can be started, some preprocessing steps need to be taken: (a) The features for the image data need to be extracted (see Section 3); (b) The index structures required for the range query processing must be built; (c) The lookup table mapping the image patches to their geolocations needs to be setup. Afterward, the search engine can be deployed and queries can be processed. In Figure 4, the overall workflow of how a query is processed in our search engine is demonstrated. Once the user selects some positive and negative patches to create a user query and a decision branch model, the query information is sent to the search application (Step 1). In the search application, the chosen model is trained with the labeled patch features, deriving range queries from the trained model (Step 2). These range queries are processed using the pre-built index structures (Step 3). The resulting object IDs and their geolocations, along with certain query statistics such as the number of returned objects and query time, are then sent back to the web application (Step 4). Now, the web application starts to visualize the identified query rectangles on the web interface while concurrently requesting the corresponding image patches from the data application (Step 5). These image patches for the discovered objects are fetched from the image database (Step 6) and asynchronously loaded and displayed in the web interface sidebar (Step 7).

### 4.1 Search Models

Our current prototype contains five different classification models for processing the user queries, namely Decision Branches\(^5\), Decision Branches Ensemble\(^6\) comprising 25 individual models, Decision Tree, Random Forest based on 25 decision trees and a Nearest Neighbor baseline, where only the 1,000 nearest neighbors are returned. Note that only the decision branch models and the nearest neighbor baseline can leverage the pre-built index structures. The traditional scan-based approaches decision tree and random forest are included to highlight the difference in query response time of the utilized “index-aware” models in contrast to scan-based approaches. The

---

\(^3\)https://leafletjs.com/

\(^4\)https://fastapi.tiangolo.com

\(^5\)Termed DBranch\[B\] in [8].

\(^6\)Termed DBEns\[B\] in [8].
nearest neighbor baseline is included for the sake of completeness. Although this model is based on a minor subset of the total 384 features, leading to a potential compromise on result accuracy, it does offer fast query response times. This efficiency is due to its utilization of the underlying k-d tree index structures.

4.2 User Interface

The web-based user interface of RapidEarth shown in Figure 3 and includes a configuration panel (left) and a map panel (right). These components work cohesively to ensure a seamless user experience.

Configuration Panel. RapidEarth includes a multitude of settings where users can: ❶ select the searcher; ❷ start the search for selected objects; ❸ reset the search to focus on a new class of objects; ❹ enable or disable the area selection feature, which enables users to select entire areas that to be labeled as negative (e.g., users could label entire forests to exclude them from the search); ❼ configure the number of negative samples that are added to the query; ❽ import and export user queries. Furthermore, the interface shows the images of the query results as well as the query statistics (query time and number of found objects).

Map Panel. Transitioning to the right, we have an interactive web map that allows users to delve into geospatial and label data crucial for training the specified search models selected on the configuration panel. When zoomed to a specific level, users are able to select positive and negative instances that compose the training set—with a left mouse click marking positive instances (red rectangle) and a right click designating negative instances (blue rectangle). Note that the current position of the cursor is represented by an orange rectangle. Once a sufficient number of patches have been labeled, a simple click on the “Start Search” button (❹) triggers the search. The search results are color coded to further assist the user, with green rectangles representing patches identified during the search but present in the training set, and yellow rectangles representing newly discovered ones.

5 DEMONSTRATION

In our demonstration, the audience can (1) interact with RapidEarth to query objects over the country of Denmark, and (2) refine posed queries based on the search results. During the initial search, participants can query, e.g., solar panels. The search results are mapped for visual representation and individual patches are loaded onto the sidebar for closer examination (see Figure 5). These patches in the sidebar are arranged in descending order of the model’s confidence. A higher position in the sidebar corresponds to more frequent appearances of the patches within the boxes of the decision branches.

One of the salient features of our search engine lies in its capacity for refinement. This empowers users to iteratively fine-tune their query based on the search outcomes. If the initial results aren’t satisfactory, visitors will be invited to refine their queries by tweaking the positive and negative instances until the desired outcome is achieved. Unlike scan-based alternatives, which necessitate a full re-scan for each new query, our search engine deftly delivers new results within a span of seconds, optimizing user experience.

6 SUMMARY

We introduce RapidEarth as a tool to aid remote sensing and earth observation analysts in rapidly finding objects of interest in very large satellite imagery data sets. A short video showcasing demonstration scenarios is available at https://youtu.be/jwS96I1qhU8 and the source code for RapidEarth can be found at https://github.com/decisionbranches/rapidearth. We encourage readers to adapt RapidEarth for their own data sets and use cases.

ACKNOWLEDGMENTS

Fabian Gieseke acknowledges support from the Independent Research Fund Denmark (grant number 9131-00110B) and from the German Federal Ministry of Education and Research (AI4Forest project; grant number 01IS23025).

REFERENCES


*Underlying caching mechanisms of the OS can even further accelerate the adapted queries by caching already loaded parts of the index.