Google Landmark Retrieval Challenge

Finding the same landmarks of a query image in a large dataset

Orest Rehusevych & Andriy Pankiv

Faculty of Applied Sciences at Ukrainian Catholic University

Abstract

In this report, we overview different approaches for finding the same landmarks from a large dataset that corresponds to a given query image. We apply several traditional and non-traditional models to our Google Landmark Retrieval dataset and make a comparison between them based on speed, performance rate, and simplicity. As we had an established amount of time for project implementation, our main goal was to make our approach perform as fast as possible due to computing capability constraints.

Importance of the problem

Image retrieval is a fundamental problem in computer vision: given a query image, can you find similar images in a large database? This is especially important for query images containing landmarks, which accounts for a large portion of what people like to photograph.

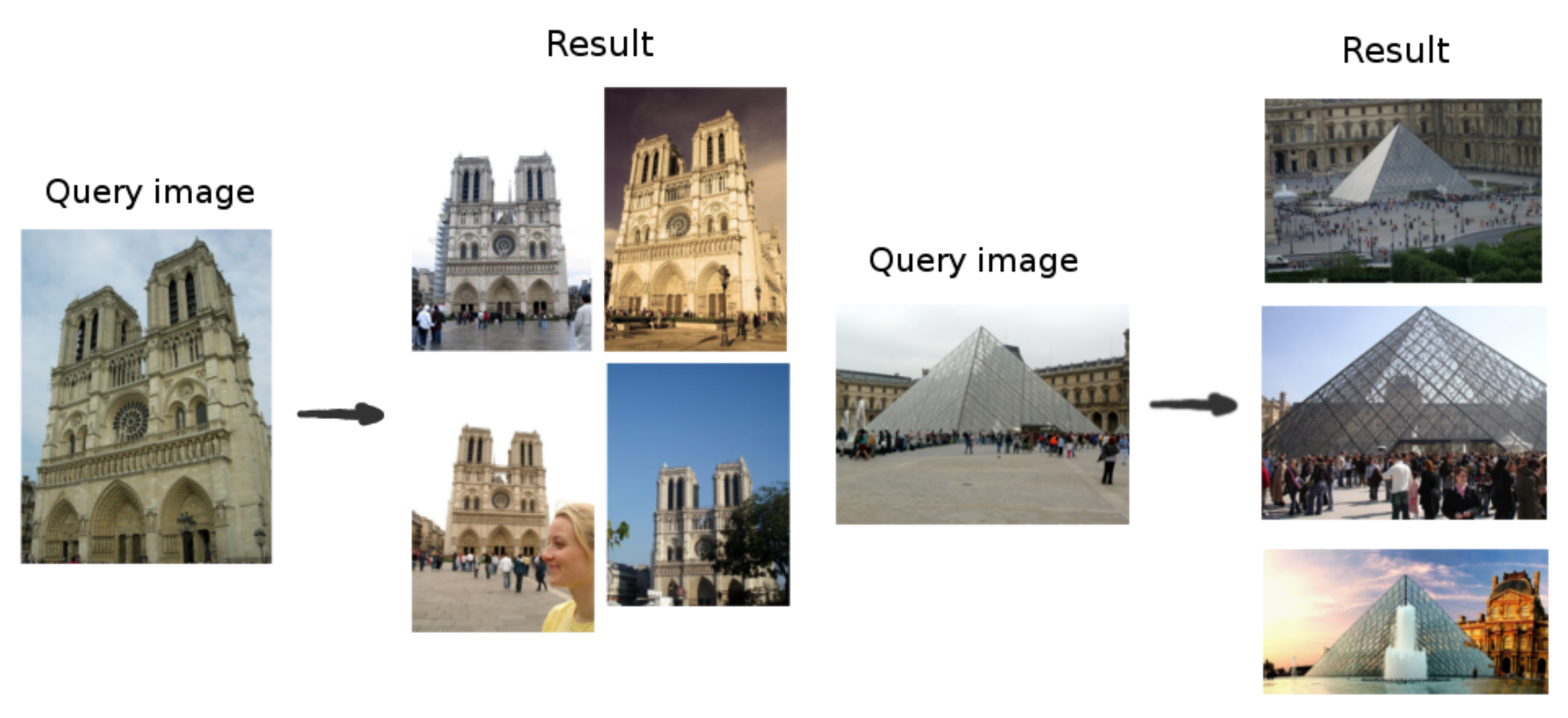

Figure 1: Example of finding all the landmarks from the dataset that correspond to the query image.

Moreover, nowadays humanity produce more and more images and it becomes much more complicated to find something needed in this huge amount of photos. So we try hard to find new ways to represent image content for easier and faster search, so that customer would spent less time on usual image search activities.

Datasets

During the process of project development we used 3 datasets: 2 for testing purposes and 1 main for final submission and checking how our models and approaches would work with really big data. Our datasets:

- **Paris6K**
  The Paris Dataset consists of 6412 images collected from Flickr by searching for particular Paris landmarks. There are 12 unique landmarks over which an object retrieval system can be evaluated.
- **Oxford5K**
  The Oxford Buildings Dataset consists of 5062 images collected from Flickr by searching for particular Oxford landmarks. There are 11 unique landmarks over which an object retrieval system can be evaluated.
- **Google Landmark Retrieval Dataset**
  - train set - 1 116 904 images. (359 GB)
  - test set - 117 703 images. (40 GB)

  The dataset we use is the largest worldwide dataset for image retrieval research, comprising more than a million images of 15K unique landmarks

Modeling

Partition Min-hashing

This approach is simple enough in understanding and implementation, but at the same time is highly effective talking about big data. The algorithm can be divided into 3 steps:

- **Image preparation**
  We convert an image into greyscale to have only one channel instead of three, by doing this we improve the speed of the algorithm. After that, we resize the image to 256x256 to easily divide them into squares, also better speed as we have fewer pixels. Last part of image preparing is cropping it into squares - partitions.

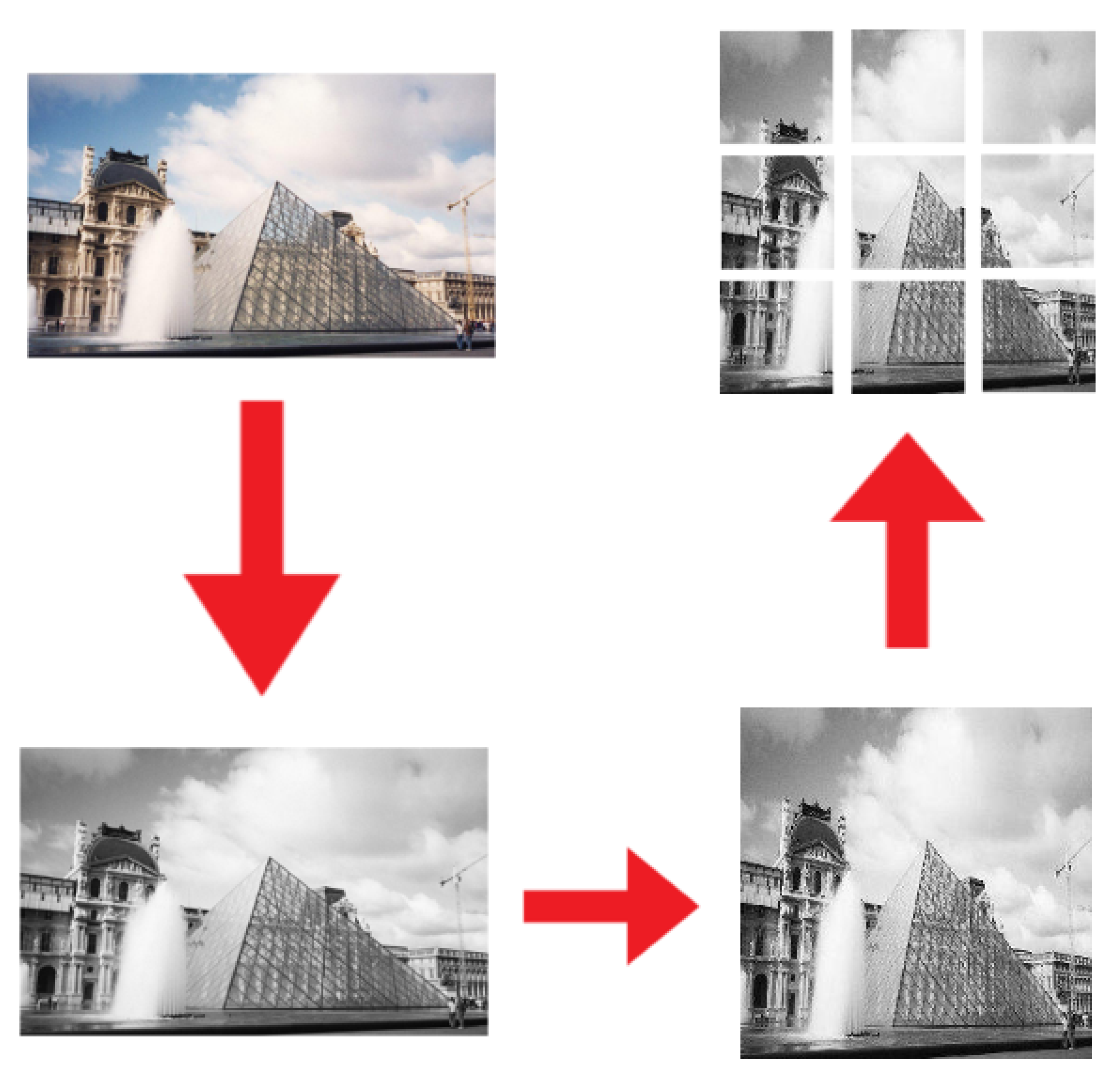

Figure 2: Process of image preprocessing: Converting image to greyscale (1 color channel) => Resizing image to 256x256 => Cropping image to 9 equal squares - partitions

- **Hashing and calculating the true hash**
  The main part of the algorithm. We take M different hash functions. Take first hash function and calculate a hash for N partitions and calculate a true hash, repeat for all hash functions, where the true hash is the minimum hash value from all partition of specific hash function. The resulting array would be the set of length M containing true hashes.
- **Calculating the similarity between the hashes**
  This part of the algorithm includes calculating the similarity between hashes using any of the next four following similarity metrics.

Evaluation

- **Hamming distance**

$$\Delta(x,y) := \sum_{x_i \neq y_i} 1, i = 1, \ldots, n.$$

- **Our own similarity metric SummOne**, which we considered the right choice for our algorithm.
  Here is how it works. Calculate the hamming distance of every pair with the next logic : IF 1 - diff (h11, h21) $\geq$ t = 1 else 0, where t is a threshold established for particular hash function. After comparison of every pair, we get a binary array of the length M. Distance would be the sum of ones in the binary array.
- **Normalized hamming distance**

$$\Delta_n(x,y) := \frac{1}{n} \sum_{x_i \neq y_i} 1, i = 1, \ldots, n.$$

- **Equality Percentage (EP)**

$$EP := 100 \cdot \Delta_n.$$

Comparison

We have made a comparison for all our datasets: Paris6K, Oxford5K and Google Landmark Retrieval Dataset on how much time it takes to compare 1 to ALL images in a dataset with different number of independent hash functions.

Dataset	3 Hashes	6 Hashes	9 Hashes	12 Hashes
Paris6k	0.605790	0.834278	0.996818	1.170207
Oxford5k	0.394306	0.561861	0.731656	0.901909
GLRD[1]	100.0097	139.6139	172.8474	213.2116

Table 1: Comparison of time cost of finding the similarity of hashes between 1 vs ALL images in each dataset with different number of independent hash-functions (time in sec).

Conclusions

In this poster we briefly explained how Partition Min-hashing algorithm works, explained why it can be applied to our problem and made a comparison of its usage with different parameters on various datasets.

References

[1] Filip Radenovic Jiri Matas Anastasiya Mishchuk, Dmytro Mishkin. Working hard to know your neighbors margins: Local descriptor learning loss.

[2] Rodiney Elias Maral Joo Augusto da Silva Jnior and Marcos Aurlio Batista. Image retrieval: Importance and applications.

[3] Qifa Ke Michael Isard and David C. Lee. Partition min-hash for partial duplicate image discovery.

[4] M. H. Jakubowski R. Venkatesan, S.-M. Koon and P. Moulin. Robust image hashing.

[5] Adrian Rosebrock. Image hashing with opencv and python.

[6] Guoqing Wang and Jun Wang. Sift based vein recognition models: Analysis and improvement.

[7] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions.