Circuits and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://cas.tudelft.nl/

CAS-2022-00

M.Sc. Thesis

Re-ranking for Improved Image Query-Based Search

Qi Zhang

Abstract

In image search, an algorithm tries to identify images in a database that are similar to a query image. Image search has numerous applications. For example, image search can help historians find images of a historical building from a large image database of buildings worldwide. Feature extraction and nearest neighbors methods are standard steps in the image search task. Current state-of-the-art image search engines extract features from images through CNNs. Next, they find top-n images closest to the query image by nearest neighbor search (NN search). After NN search, the engine outputs results. However, it is still hard for current image search technologies to identify images under complicated situations, such as illumination, viewpoints, and rotation changes. These changes will reduce the accuracy of the engine. Therefore, how to improve the accuracy of the image search engine is still a challenging task.

Researchers want to design re-ranking methods for image search to improve accuracy. According to previous research, there are mainly two types of re-ranking: global feature-based re-ranking and local feature-based re-ranking. Furthermore, both types of re-ranking methods could improve the accuracy of the image search engine. However, previous re-ranking methods are not fast or accurate enough. We implement novel global and local feature-based re-ranking methods to improve the accuracy of image search significantly. We test our re-ranking methods on popular image search databases. Experiments show that our re-ranking methods improve accuracy while slightly increasing computation time.

Re-ranking for Improved Image Query-Based Search

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Qi Zhang
born in Beijing, China

This work was performed in:

Circuits and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"Re-ranking for Improved Image Query-Based Search"** by **Qi Zhang** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 12 August 2022

Chairman: _____
prof.dr.ir. Justin Dauwels

Advisor: _____
prof.dr.ir. Justin Dauwels

Committee Members: _____
prof.dr. David Tax

_____

Abstract

In image search, an algorithm tries to identify images in a database that are similar to a query image. Image search has numerous applications. For example, image search can help historians find images of a historical building from a large image database of buildings worldwide. Feature extraction and nearest neighbors methods are standard steps in the image search task. Current state-of-the-art image search engines extract features from images through CNNs. Next, they find top-n images closest to the query image by nearest neighbor search (NN search). After NN search, the engine outputs results. However, it is still hard for current image search technologies to identify images under complicated situations, such as illumination, viewpoints, and rotation changes. These changes will reduce the accuracy of the engine. Therefore, how to improve the accuracy of the image search engine is still a challenging task.

Researchers want to design re-ranking methods for image search to improve accuracy. According to previous research, there are mainly two types of re-ranking: global feature-based re-ranking and local feature-based re-ranking. Furthermore, both types of re-ranking methods could improve the accuracy of the image search engine. However, previous re-ranking methods are not fast or accurate enough. We implement novel global and local feature-based re-ranking methods to improve the accuracy of image search significantly. We test our re-ranking methods on popular image search databases. Experiments show that our re-ranking methods improve accuracy while slightly increasing computation time.

Acknowledgments

Studying far from my hometown is not easy for me. But it was also an exciting journey because I could see and learn about the world from a new perspective. This process is sweet and sour at the same time. Adventure comes and goes, but in the end, only good things remain in my heart.

I would like to thank Dr.ir. Justin Dauwels for giving me this thesis work opportunity and his guidance throughout this work. Without his help, this work could not have been finished. His suggestions have not only improved this work but also encouraged me to achieve a higher level.

I would also like to express my gratitude to my teammates, Yuanyuan and Yanan. We have worked as the best team to solve problems and supported each other mentally. Not only we are teammates, but also we are friends in life.

I am lucky to study in the CAS group where I obtain a lot of help. I would like to thank Yanbo Wang and Cristian Meo for their support throughout this work. They always share inspiring ideas and encourage me to be better.

Finally, I am indebted to my family. I could not make my dream come true without their support. They always believe me and encourage me to keep going on my way.

Qi Zhang
Delft, The Netherlands
12 August 2022

Contents

List of Figures

List of Tables

1 Introduction

We illustrate the structure of image search engines in Figure 1.1. With recent advances in the field of computer vision, there are a lot of outstanding research studies [13–18] that aim to enhance the performance of image search engines. There is an image database and a query image. Image search engines can find similar images to the query image from a database. For example, if a user inputs an image into the engine, the engine will search an existing gallery database and output images similar to the input image to the user. This process mainly involves feature extraction [13, 14, 17, 19–21] and nearest neighbor search (NN search) technology [22]. Feature extraction refers to extract features from an input image (query) and database images (gallery) by hand-crafted [2,3] or CNN [13–17,23] methods. The extracted features are global, meaning a single vector represents an image. Then, NN search, such as k-nearest neighbors (KNN) and approximate nearest neighbors (ANN), is implemented to calculate the distance between the features of the query image and gallery images. Also, NN search sorts the distance to find the gallery images most similar to the query image. Feature extraction and NN search are necessary steps in image search. As this process is based on the query image, this engine is a query-based search engine.

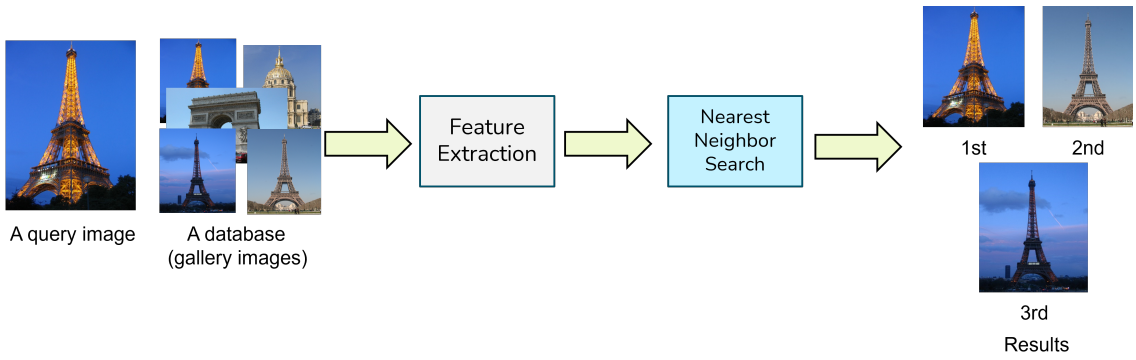

Figure 1.1: The off-the-shelf structure of image search engines. There are two main steps: feature extraction and NN search.

However, due to the complicated situations in gallery images and feature structures, the accuracy of this pipeline is limited. The off-the-shelf image search technologies neglect the unique information of each image while focusing on detecting the primary information of images. For example, in Figure 1.2 and Figure 1.3, there are two images of the Eiffel tower in different illumination situations. Although the main object in these two images is the same, it is hard for off-the-shelf image search technologies to determine that they are similar. Recently, researchers have implemented re-ranking to solve the accuracy dilemma to combat this problem. Re-ranking is a post-processing technology in image search, which re-ranks the preliminary results (candidate images)

from the NN search to obtain higher accuracy. Unfortunately, the current re-ranking methods are not ideal because:

1. Although state-of-the-art re-ranking methods are accurate for complex images, they are too computationally demanding to be feasible for a wide variety of image search pipelines.
2. Although some re-ranking methods are fast, they lack analysis of critical information in images, which leads to a little improvement in accuracy.

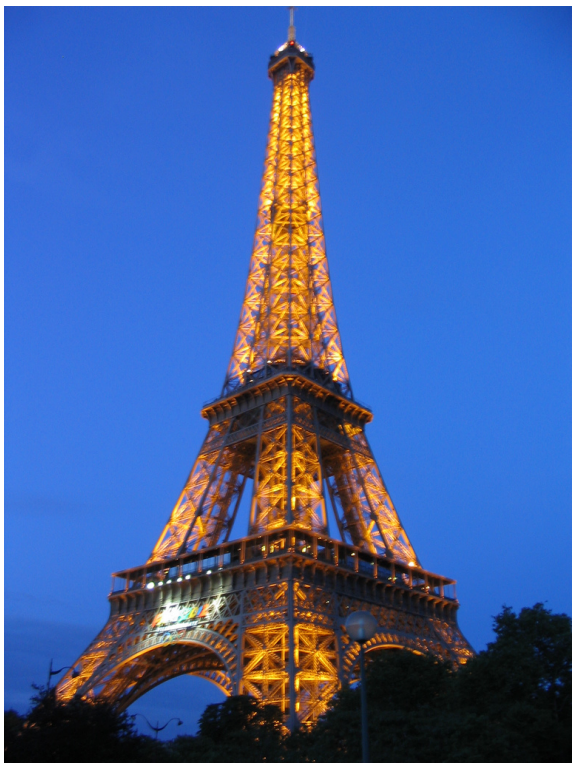

Figure 1.2: Image A (illuminated Eiffel tower) from RParis6k [1].

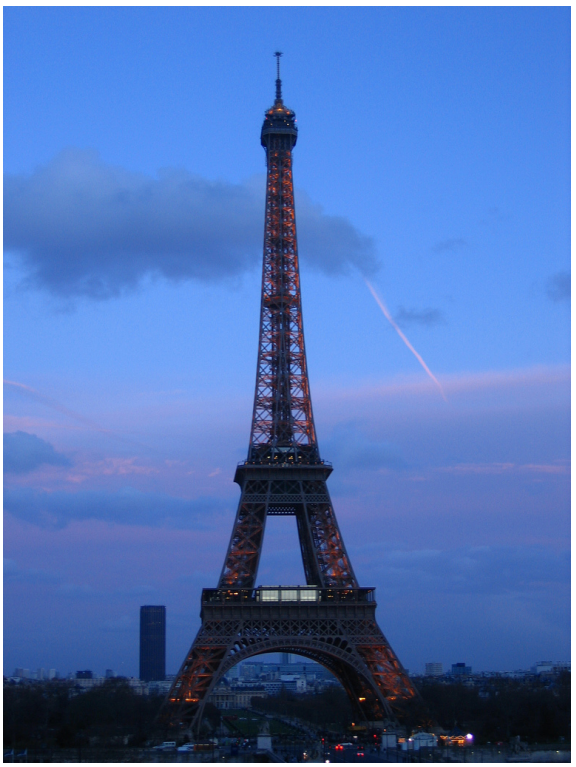

Figure 1.3: Image B (dim Eiffel tower) from RParis6k [1].

Therefore, we attempt to implement re-ranking methods to improve the accuracy of image search engines with only a slight increase in computation time. We illustrate the structure of our image search pipeline in Figure 1.4.

1.1 Problem Description

With recent advances in feature extraction techniques, active researches aim to improve the accuracy of local or global feature-based re-ranking. The local feature means we can divide an image into small local parts, extract the critical information of these parts, and combine information into a feature matrix. Therefore, in local feature-based re-ranking, we use a feature matrix to represent an image and implement feature

Figure 1.4: The pipeline of our image search engine. There are three main steps: feature extraction, NN search and re-ranking.

matching. Specifically, researchers have focused on extracting local features to analyze the geometric similarity between two images, calculate the similarity scores, and re-rank gallery images by the scores. We illustrate the pipeline of local feature-based re-ranking in Figure 1.5.

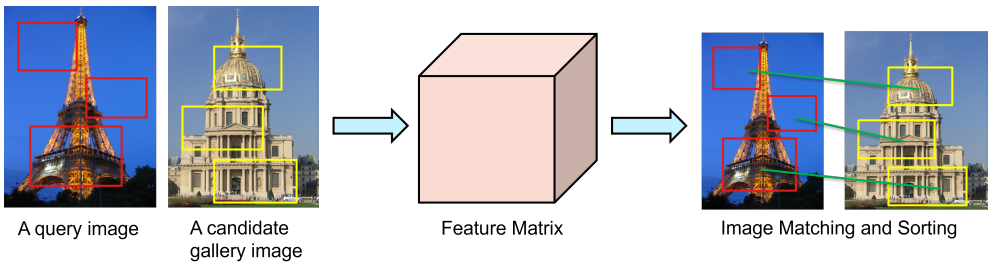

Figure 1.5: The pipeline of local feature-based re-ranking. We extract local features to match images.

By contrast, the global feature encodes an image into a single feature vector. Global feature-based re-ranking methods aim to enhance the feature vector to achieve robust performance. Furthermore, researchers augment global features from the previous feature extraction step and re-rank many gallery images without extracting new features. Global feature-based re-ranking methods are usually much faster than local feature-based re-ranking because global features are lighter than local features. We illustrate the pipeline of global feature-based re-ranking in Figure 1.6.

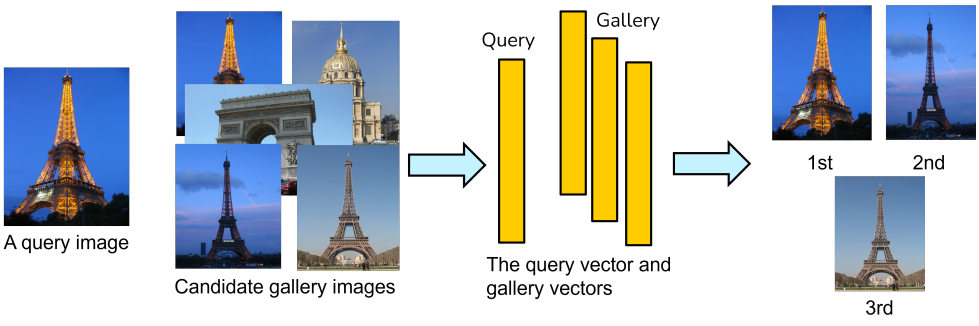

Figure 1.6: The pipeline of global feature-based re-ranking. We enhance global features from the feature extraction step to re-rank images.

However, both methods are far from ideal. On the one hand, local feature-based re-ranking is relatively slow because local features are multidimensional matrices that are too computationally demanding to process large databases. In this way, the feature extraction latency and database memory requirements would be extremely high [17,21], which leads our engine to be impossible to obtain results fast. On the other hand, although the global feature-based re-ranking is faster, it is not robust to complicated situations in images, e.g., illumination, viewpoint, and rotation changes [13, 17, 23]. The global feature of a query image can only contain some information and is not representative enough. We illustrate the advantages and disadvantages of local and global feature-based re-ranking in Table 1.1.

Metric	Local feature-based re-ranking	Global feature-based re-ranking
Extracting new features	Yes	No
Using features from the feature extraction step	No	Yes
Feature type	Matrix	Vector
Speed	Slow	Fast
Accuracy	High	Medium
Extra memory requirement	Yes	No
Pixel-to-pixel analysis	Yes	No

Table 1.1: The advantages and disadvantages of local and global feature-based re-ranking.

The detailed analysis of Table 1.1 is as follows:

1. **Features:** Local feature-based re-ranking extracts local feature matrices from candidate images instead of global features from the feature extraction step. By contrast, global feature-based re-ranking uses and enhances global feature vectors from the feature extraction step [13, 17, 23, 24].

2. **Speed:** First and foremost, the local feature matrix contains more information from an image and is heavier than the global feature vector. Secondly, local feature-based re-ranking requires extracting new local features, while global feature-based re-ranking uses existing global features. For these two reasons, global feature-based re-ranking is faster than local feature-based re-ranking [17, 24].

3. **Accuracy:** Local feature-based re-ranking is robust to complicated situations in images because the local feature matrix contains most information from an image. By contrast, global feature-based re-ranking is not robust to complex conditions because the global feature vector may lack important information from an image [13, 17, 23].

4. **Extra memory requirement:** The local feature matrix requires extra memory because the local feature matrix contains much information [5, 9, 17, 21]. We need to store the information in memory. By contrast, the global vector is light, and we do not need to use extra memory to store it.

5. **Pixel-to-pixel analysis:** Pixel-to-pixel analysis refers to comparing a pair of images by pixel, which is from the local feature matrix. Therefore, only local feature-based re-ranking can provide pixel-to-pixel analysis. We show an example of the pixel-to-pixel analysis of a pair of images in Figure 1.7. The original images are from ROxford5k [1].

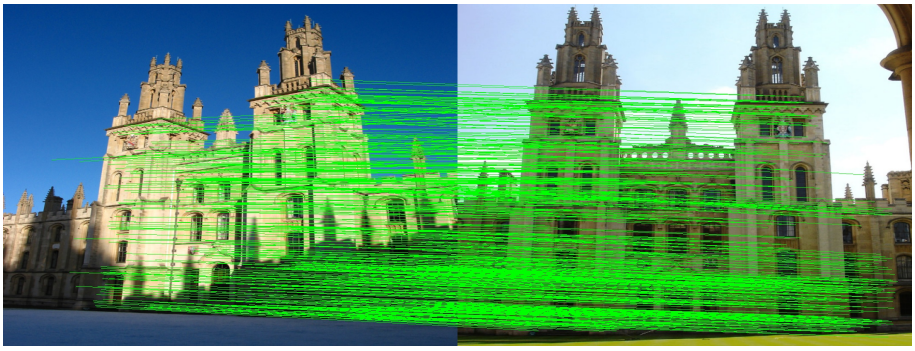

Figure 1.7: Pixel-to-pixel analysis. Green lines show similarities between two images.

The core requirements of our re-ranking methods are as follows:

1. **Speed:** The re-ranking should be fast such that it can be done in a fraction of a second.

2. **Accuracy:** The re-ranking methods of this work should be more accurate than off-the-shelf re-ranking methods.

Therefore, both off-the-shelf local and global feature-based re-ranking methods can not meet the requirements.

1.2 Objectives

According to the previous analysis, the main objectives of this work are as follows:

1. **Analysis:** To systematically review state-of-the-art re-ranking methods.

2. **Speed:** To develop a local or global feature-based re-ranking method that can meet the speed requirement based on off-the-shelf approaches.

3. **Accuracy:** To develop a local or global feature-based re-ranking method to improve the accuracy of image search engines.

1.3 Contributions and Innovations

The main contributions of this work are as follows:

1. An accurate and fast local feature-based re-ranking method, called SIFT-AffNet-HardNet-AdaLAM (SAHA), which is based on SIFT [2, 3], AffNet [7], HardNet [8] and AdaLAM [25]. We use AffNet, HardNet, and AdaLAM to improve the accuracy of SIFT. Furthermore, we combine bubble sort with SAHA to achieve accurate sorting. Finally, we use an offline/online design to accelerate SAHA. Therefore, SAHA is more accurate and faster than SIFT.

2. An accurate and memory-friendly local feature-based re-ranking method based on an existing image matching technology, called Local Feature Matching with Transformers (LoFTR) [9]. LoFTR is an existing local feature-based image matching method. We have applied it for a new problem, image search. Because LoFTR can not sort matching results, we use bubble sort after the LoFTR module to achieve accurate sorting.

3. A global feature-based re-ranking method, Query and Gallery Enhancement (QGE), which is based on query expansion [26, 27] and diffusion [28, 29]. First and foremost, we introduce a weight parameter w_i and L2 normalization in the original query expansion to improve the accuracy. The weight and L2 normalization make the new query feature more informative than the original query feature. Furthermore, we use a fixed number of iterations and remove the spatial verification step in query expansion to improve the speed. Finally, query expansion can not improve the accuracy of searching. Diffusion is a method to search features accurately with only a slight increase in computation time. Researchers have not combined these methods before. Therefore, we use these two approaches to design QGE. QGE is more accurate than the original query expansion method with only a small increase in computation time. Therefore QGE meets all requirements of our task.

1.4 Outline

We show the outline of this report in Figure 1.8.

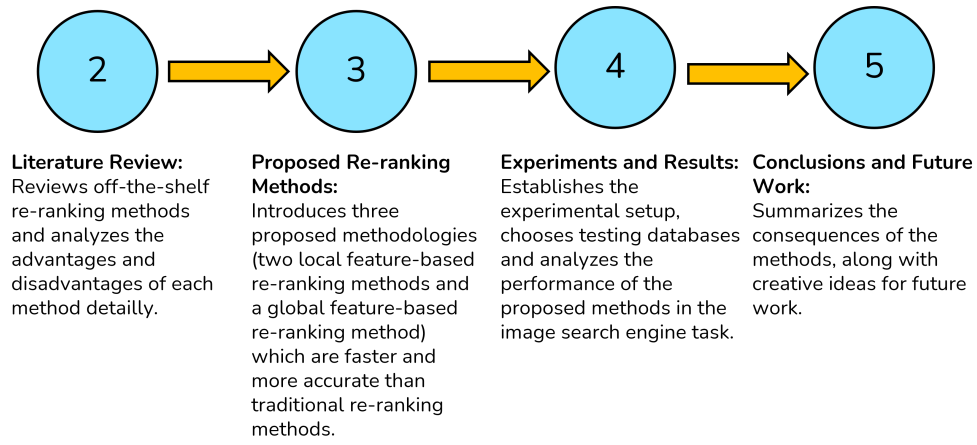

Figure 1.8: The outline of this report.

2 Literature Review

Previous re-ranking methods for image search include local or global feature-based re-ranking methods. Also, researchers refer to local feature-based re-ranking methods as geometric re-ranking or spatial verification methods [13, 17, 30, 31], and the global feature-based re-ranking methods as non-geometric re-ranking methods [31].

This chapter will review local and global feature-based re-ranking methods proposed in the literature. Scale Invariant Feature Transform (SIFT) is the most representative local feature-based re-ranking method, and query expansion is the most representative global feature-based re-ranking method. We will review these methods in this section.

2.1 Local Feature Re-ranking (SIFT)

The most representative method of local feature-based re-ranking is Scale Invariant Feature Transform (SIFT) [2, 3]. We will review SIFT in detail in this subsection.

Local feature-based re-ranking methods utilize local features from images to analyze the similarity between images pair by pair [2, 3, 17]. Specifically, there is a query image and related candidate images after preliminary ranking. In local feature-based re-ranking, a query image and one of the candidate images are analyzed to obtain matches between this pair of images. Furthermore, local feature-based re-ranking will process other candidate images similarly to obtain their matches with the query image. Instead of using the global features from the feature extraction step, local feature-based re-ranking extracts new local feature matrices. A local feature matrix contains keypoints and descriptors of an image. On the one hand, keypoints detect the position information of important points in an image. On the other hand, descriptors describe the local content information from these important points based on the position information. Local feature-based re-ranking is robust to multi-object, illumination, viewpoint, and rotation changes in images [2, 3, 5, 32] because keypoints and descriptors contain most key information of an image. Moreover, local feature-based re-ranking can provide a pixel-to-pixel comparison picture between a pair of images, which can intuitively show the similarity of a pair of images and assist users in analyzing these two images. We illustrate an example of the pixel-to-pixel comparison picture of a pair of images in Figure 2.1.

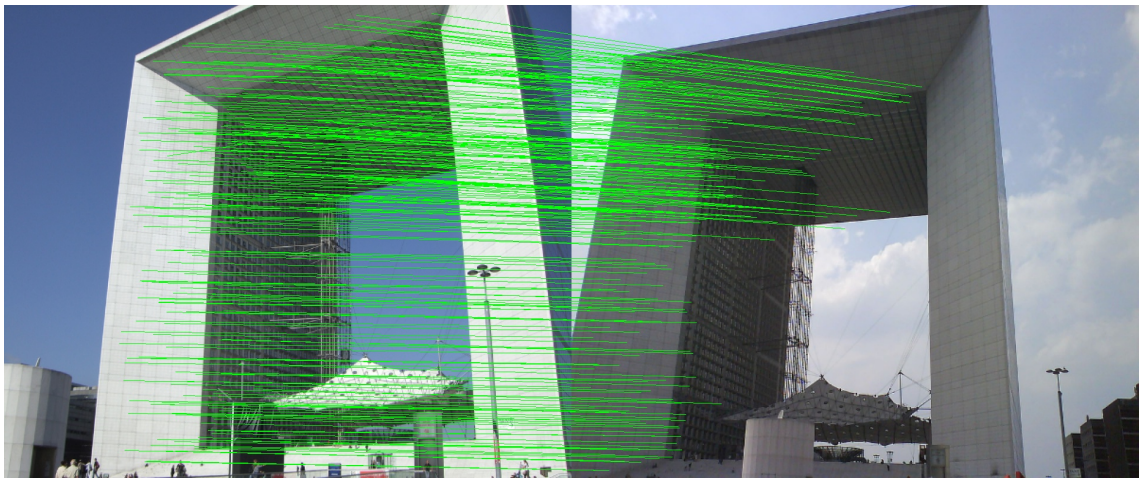

Figure 2.1: The pixel-to-pixel comparison picture of a pair of images. Green lines show similarities between two images.

The original images are from RParis6k [1].

In Figure 2.1, the small image on the left side is the query image, and the other small image is a candidate image. According to the pixel-to-pixel comparison picture provided by local feature-based re-ranking, the more the matches, the more similar the two images are.

There are two essential steps in local feature-based re-ranking: 1) extracting local feature matrices (keypoints and descriptors) of images and 2) matching these keypoints and descriptors of images pair by pair. In this subsection, we review related works about these two steps.

2.1.1 Handcrafted Local Feature Extraction

There are two ways to implement local feature extraction: handcrafted local feature extraction and deep learning local feature extraction. Firstly, we will review the handcrafted local feature extraction.

Handcrafted local feature extraction extracts local feature matrices based on classical algorithms. The most fundamental algorithm is Scale Invariant Feature Transform (SIFT) [2, 3, 20, 21], which is the foundation of many other handcrafted methods [19, 33–37]. These handcrafted methods have been proved to be very accurate, and many researchers use them as re-ranking methods or benchmarks in the image search system [16, 17, 23, 38–41].

SIFT is a representative handcrafted method that has been widely applied and achieved outstanding results in object recognition [2, 3], image matching [2, 3, 19–21, 33, 34, 36], and image search [13, 17, 19–21, 23]. Therefore, SIFT is regarded as the representative of handcrafted methods. We will focus on SIFT.

There are four main stages in SIFT: 1) scale-invariant feature detection, 2) local feature matching, 3) spatial verification by linear least squares, and 4) outlier detection.

First and foremost, we apply the scale-invariant feature detection to images, which is also the most important step in SIFT. In this step, SIFT extracts keypoints to precisely

detect the location of points with important information and extract descriptors from keypoints. Keypoints are defined as the maxima and minima value of the difference of Gaussian (DoG) scale space [2,3,42], as an approximation of the Laplacian of Gaussian (LoG) [3,37]. Then we implement the Gaussian kernel to calculate the convolution of a two-dimensional Gaussian function with an image to obtain keypoints, because its derivatives are smoothing and optimal for scale-space analysis [2,3,35]. Since the 2-dimension Gaussian function is separable, the convolution of the 2-dimension Gaussian function is obtained by the following Equation 2.1 [3]:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}, \tag{2.1}$$

where x refers to the position of each point in an image and σ refers to the scale parameter for extracting features from an image. To finish the 2-dimension Gaussian calculation, the first convolution with $\sigma = \sqrt{2}$ to extract keypoints K_1 is performed. Then, a second convolution on the extracted keypoints K_1 to extract new keypoints K_2 with $\sigma = \sqrt{2}$ is performed. Continuous iterations can obtain an image extraction structure similar to a pyramid. Furthermore, we can implement the Hessian matrix to strengthen the Gaussian function in speed and accuracy [35]. Moreover, we can replace the gradient orientations in DoG with Haar wavelet responses in the $x-$ and $y-$directions [35] to achieve better performance. Figure 2.2 shows the structure of DoG.

After obtaining keypoints by DoG, we can transform an image into a local descriptor matrix in which every single vector is invariant to illumination, viewpoint, and rotation situations and robust to local geometric distortion. Except for keypoints and descriptors from SIFT, we can introduce the FAST keypoints and BRIEF descriptors to achieve a faster performance [36] in the handcrafted local feature extraction.

Secondly, local feature matching is crucial in SIFT and many SIFT-like algorithms. The nearest neighbor search [22,43] is performed to obtain the most similar candidate descriptors with the descriptors of the query image in minimum Euclidean distance [2,3,19,22,35] in SIFT and SIFT-like algorithms. Also, we can use different distances with unit Euclidean norm to improve the matching [19].

In the third step, we implement spatial verification by linear least squares to enhance the quality of matching results. All matching scores from keypoints and descriptors are subject to a verification procedure in which we use a linear least squares solution for the parameters of the affine transformation relating each pair of image points $[u\ v]^T$ and a pair of the corresponding model points $[x\ y]^T$. The affine transformation of the model points $[x\ y]^T$ to the image points $[u\ v]^T$ is elaborated as Equation 2.2 [3]:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \tag{2.2}$$

where the model translation is $[t_x\ t_y]^T$. The affine rotation, scale and stretch parameters are represented by m_1, m_2, m_3 and m_4 respectively [2,3]. To obtain the image transformation parameters, the equation above can be derived into Equation 2.3 [3]:

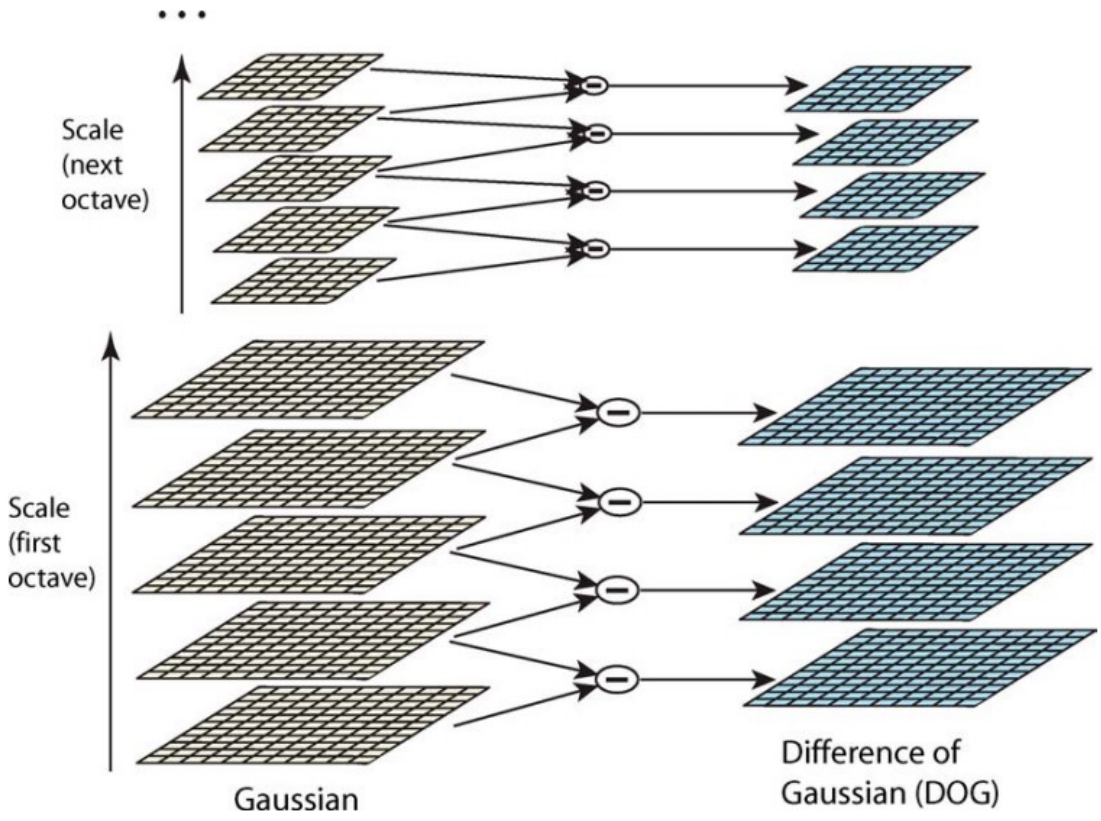

Figure 2.2: The structure of Difference of Gaussian (DoG) [2].

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ \dots & & & & & \\ \dots & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ . \\ . \end{bmatrix}. \tag{2.3}$$

Therefore, we can obtain the relationship between a pair of image points and a pair of model points. Furthermore, we give the linear solution of each point in an image in Equation 2.4 [3]:

$$\hat{x} = (A^T A)^{-1} A^T b, \tag{2.4}$$

where $\hat{x}$ is an unknown n-dimensional parameter vector, A is a known m-by-n matrix and b is a known m-dimensional measurement vector [2]. Equation 2.3 minimizes the sum of the squares of the distances from the projected image points to the corresponding model points, which verifies the relationship of the model locations in an image. We show the results of models and points verification in an image in Figure 2.3.

The final step is outlier detection. Based on spatial verification, we can remove outliers by checking for the relationship between model locations and image points.

Figure 2.3: Results of models and points verification in an image [3].

Each match must agree within 15 degrees orientation, $\sqrt{2}$ change in scale, and 0.2 times the maximum model size. If the remaining points are fewer than 30% after discarding outliers, we regard the match as failure [3].

Most handcrafted algorithms are SIFT and SIFT-like algorithms and have shown many advantages as follows:

1. **Accuracy:** SIFT and SIFT-like algorithms exhibit high accuracy and repeatability in image search and image matching task [2, 3, 19–21, 34–37]. Handcrafted algorithms show outstanding robustness to the illumination, viewpoint and rotation changes in an image.

2. **Applicability:** Unlike learned algorithms, handcrafted algorithms do not require trained neural networks that are computationally demanding, ensuring handcrafted algorithms are friendly to different hardware and fast.

2.1.2 Deep Learning Local Feature Extraction

Since deep learning models are powerful in processing images, researchers have tried implementing deep learning models to simulate SIFT. Therefore, researchers use deep

learning models to extract features.

Deep learning local feature extraction uses the convolutional neural network (CNN) as the backbone, and performs feature extraction on an image by trained parameters [44–49]. Because of the excellent performance in feature extraction, CNN is rapidly developed into a popular feature extraction approach comparable to handcrafted approaches [20, 21, 50].

Learned local feature extraction builds hierarchical multi-layer CNN networks with various filtering sizes to extract local features in an image. The CNN backbone is one of the most important component in learned local feature extraction [45, 48, 49, 51]. CNN can effectively reduce the dimensionality of images to represent a large amount of information into a small amount of information and retain core information from an image. Furthermore, the information extracted by CNN can provide robustness against complicated situations such as multi-object, illumination, viewpoint, and rotation changes. Therefore, CNN has been widely used in image matching [52–56] and image search [13, 16–18, 57] and become the highlighted representative approach in deep learning feature extraction [21, 45, 48, 49].

There are four main parts in CNN [45, 48, 49, 51], as shown in Figure 2.4.

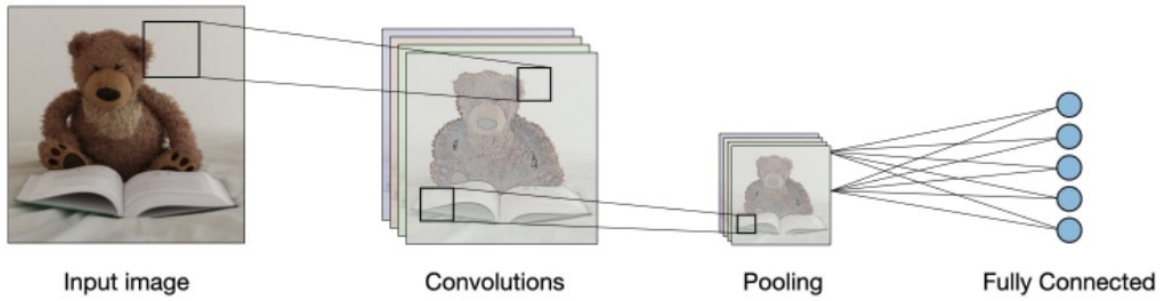

Figure 2.4: The structure of CNN [4].

1. **The convolutional layer:** First and foremost, the convolutional layer is responsible for extracting features from an image. Generally, each convolutional layer contains a special-size convolution kernel. An image is passed through these convolutional layers and transformed into higher-dimensional data. [49], as shown in Figure 2.5.

2. **The Rectified Linear Units (ReLU) layer:** Next, the ReLU layer enhances the non-linear characteristics of the decision function and the entire neural network, making the data more robust, highly refined, and highly repeatable. Then, the pooling layer divides the image into several rectangular regions and outputs the maximum value for each local region [49].

3. **The pooling layer:** Furthermore, the pooling layer reduces the number of parameters and solves problems of overfitting and over sensitivity [48, 49].

4. **The fully connected layer:** Finally, the fully connected layer stitches the data

from multiple convolutional and pooling layers to output a single-pass feature vector [16] or multi-dimensional matrix [45, 48, 49].

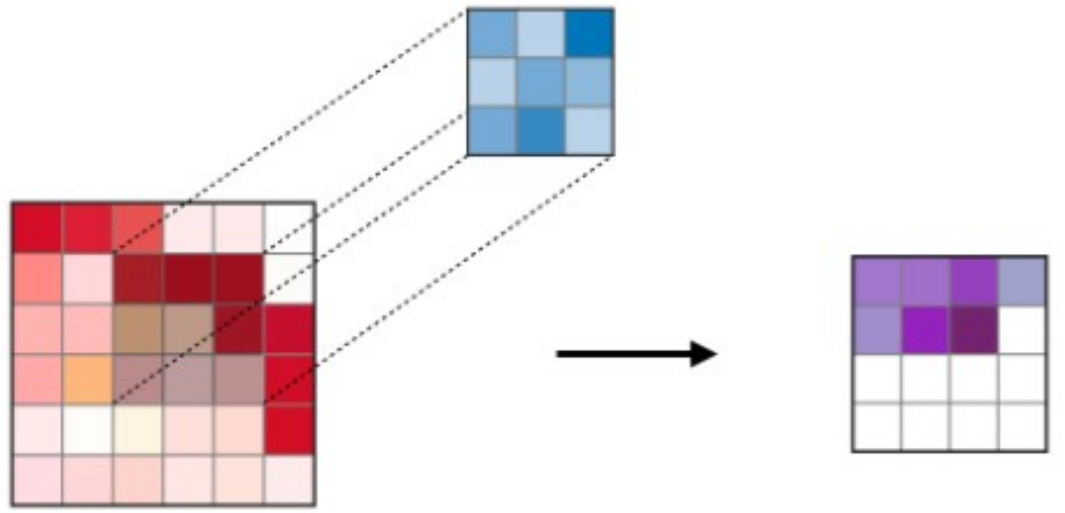

Figure 2.5: The convolutional layer [4].

The trained CNN backbone can perform feature extraction on an image through this pipeline. There are two important components in the deep learning local feature extraction: feature detection and description [5, 58, 59]. As handcrafted methods, CNN-based local feature extraction methods follow a traditional detect-then-describe pipeline. In this pipeline, extraction methods detect the coordinates of important points containing key information by the CNN backbone A and then extract the descriptors of these points according to these coordinates by another CNN backbone B. Therefore researchers use two CNN backbones A and B to detect keypoints for feature detection and describe descriptors for feature description in an image respectively and achieve the accuracy that surpassed handcrafted methods [5, 58, 60, 61]. However, the design of different CNN backbones reduces the extraction coherence and decreases the accuracy. Researchers have jointly implemented extracting keypoints and descriptors to solve this problem to improve the accuracy of learned featured [5, 59, 62]. To implement a single CNN backbone to extract keypoints and descriptors simultaneously, researchers use a detect-and-describe structure to design the CNN backbone and achieve higher accuracy [5]. Figure 2.6 shows the difference between the detect-then-describe and the detect-and-describe structure.

Furthermore, the detect-and-describe structure becomes the popular approach because it achieves outstanding performance compared to the traditional detect-then-describe structure [5, 32, 58, 59, 62–64].

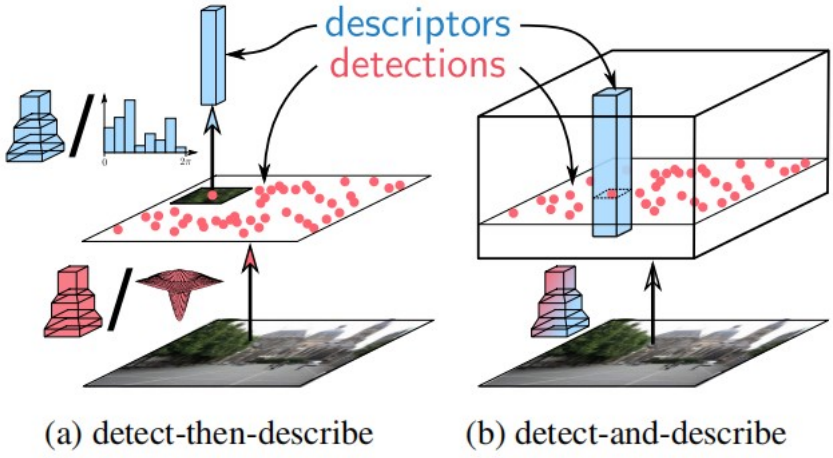

Figure 2.6: The detect-then-describe structure and the detect-and-describe structure [5].

2.1.3 Local Feature Matching

After obtaining the keypoints and descriptors of an image by SIFT, we need to implement local feature matching to analyze the similarity of keypoints and descriptors from different images. There are two main steps in local feature matching: matching and outliers filtering.

Matching is critical to obtaining the matching lines between two images based on the obtained keypoints and descriptors. The more matching lines mean the higher similarity between the two images. One of the mainstream matching methods is brute-force (BF) matching [65–67]. Researchers combine the BF matching with SIFT to achieve re-ranking. The BF matching calculates the similarity between each descriptor of an image A and each descriptor of an image B by the Euclidean distance. Generally, we implement the L2 normalization to calculate the similarity in the BF matching. The L2 normalization is shown in Equation 2.5 [68]:

$$\|d_A - d_B\|_{L_2} = \sqrt{\sum (d_A - d_B)^2}, \tag{2.5}$$

where d_I refers to descriptors of image I. The BF matching is straightforward and can provide accurate matching lines in simple image situations [66, 69]. However, the speed of the BF matching is relatively low because it needs to calculate all potential matches in two images exhaustively [66, 69].

We can also implement learned matching methods to achieve accurate feature matching. SuperGlue [6] is a representative method of the learned matching. In SuperGlue, an attentional Graph Neural Network is implemented to aggregate keypoints and descriptors from an image I into new $d_{I'}$, where d refers to descriptors. By SuperGlue, $d_{A'}$ and $d_{B'}$ can be obtained from two images A and B respectively. Then, we implement a back-end optimizer to calculate the matching lines between the two images based on d_A and d_B. Figure 2.7 shows the structure of SuperGlue.

According to experiments in the paper [6], SuperGlue shows high accuracy and is

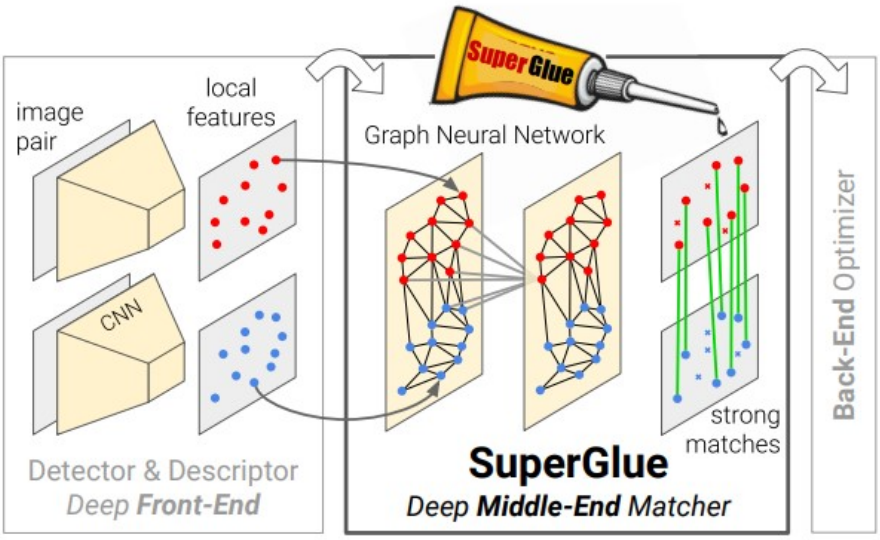

Figure 2.7: The structure of SuperGlue [6]. Red points and blue points are detectors and descriptors respectively. SuperGlue combines different local features together to obtain strong matches.

robust to complicated situations in an image. However, SuperGlue can only achieve the best results when the keypoints and descriptors are from SuperPoint [6, 58], a feature extraction method. Therefore, SuperGlue is not applicable for every local feature extraction method.

2.1.4 All-In-One Local Feature Matching

According to the above analysis, we can conclude that the mainstream local feature-based re-ranking includes two main and separate steps: feature extraction and feature matching. And there are many secondary steps in each step. Although this structure enables local feature re-ranking more flexibly to modify each part individually, the structure lacks coherence. Each secondary step A requires the output data from the previous secondary step B, and the output of A is the input data of the next secondary step C. Not only this design reduces the accuracy [9, 62, 70, 71], but also it is difficult for researchers to optimize the whole structure.

Therefore, researchers have developed the all-in-one local feature matching method. An all-in-one local feature matching method should realize local feature extraction and feature matching by a single module. Therefore it is more integrated. Some researchers try to reduce secondary steps in feature extraction and matching to design the structure [62]. Some researchers implement transformers for local feature extraction to improve the integration of structure [24, 70]. According to the results from these works, the all-in-one local feature matching structure improves the accuracy of feature extraction and matching, reduces complexity, and improves speed [24, 62, 70].

2.1.5 The Whole Process of Local Feature-based Re-ranking

First and foremost, there is a query image and many candidate images from the preliminary ranking. Then, we can implement SIFT to extract features from these images. Furthermore, we can use feature matching to match the query image with all candidate images based on the extracted features. After the local feature extraction and feature matching, we can obtain the number of matches of each pair of images. A large number of matches implies a strong similarity between the candidate and query images. We finish the local feature-based re-ranking after all candidate images are ranked.

We will use SIFT as a local feature-based re-ranking benchmark in experiments.

2.2 Global Feature Re-ranking (Query Expansion)

The most popular global feature-based re-ranking method is query expansion [26, 27]. We briefly review query expansion in detail in this subsection.

The core idea of local feature-based re-ranking is to calculate the number of matching lines between a query image and all candidate images and re-rank candidate images. Local feature-based re-ranking implements local feature extraction to re-extract the local features of the query image and candidate images instead of using the global features obtained in the feature extraction step of the image search pipeline.

On the contrary, the global feature-based re-ranking strengthens the global features extracted in the feature extraction step to re-rank. Through the feature extraction step in the image search pipeline, we can obtain the feature vector of the query image. This feature vector is a global feature that does not contain all critical information about multi-object, illumination, viewpoint, and rotation changes in the image, which drastically reduces the accuracy of the image search pipeline. Therefore, researchers try to enhance the feature of the query image and re-rank images.

Intuitively, if an image A is similar to an image B, and B is similar to an image C, then A is similar to C even if features from A and C are not explicitly similar. In Figure 2.8, there are three partially overlapping images A, B and C. We can consider image A and image C the same scene, even though their global features are unlikely to match. If features of image A, image B, and image C are combined to build a new image feature f by query expansion, this new image feature f can contain more information in this scene, and thus improves the accuracy to search related images about this scene. Furthermore, query expansion can also benefit the new feature to represent information of gradually different illumination, viewpoint, and rotation conditions. Therefore, the new feature is more representative.

Specifically, we use the features of candidate images G_i obtained by the NN search step in the image search pipeline to re-build the query feature Q. The new query feature $Q^{'}$ is the average value of the candidate features G_i [26]:

$$Q^{'} = \frac{1}{N+1}(\sum_{i=0}^{N} G_i + Q), \tag{2.6}$$

where $Q^{'}$ refers to the new query feature, G_i refers to the i-th candidate feature, and N refers to the number of candidate features. The new query feature absorbs information

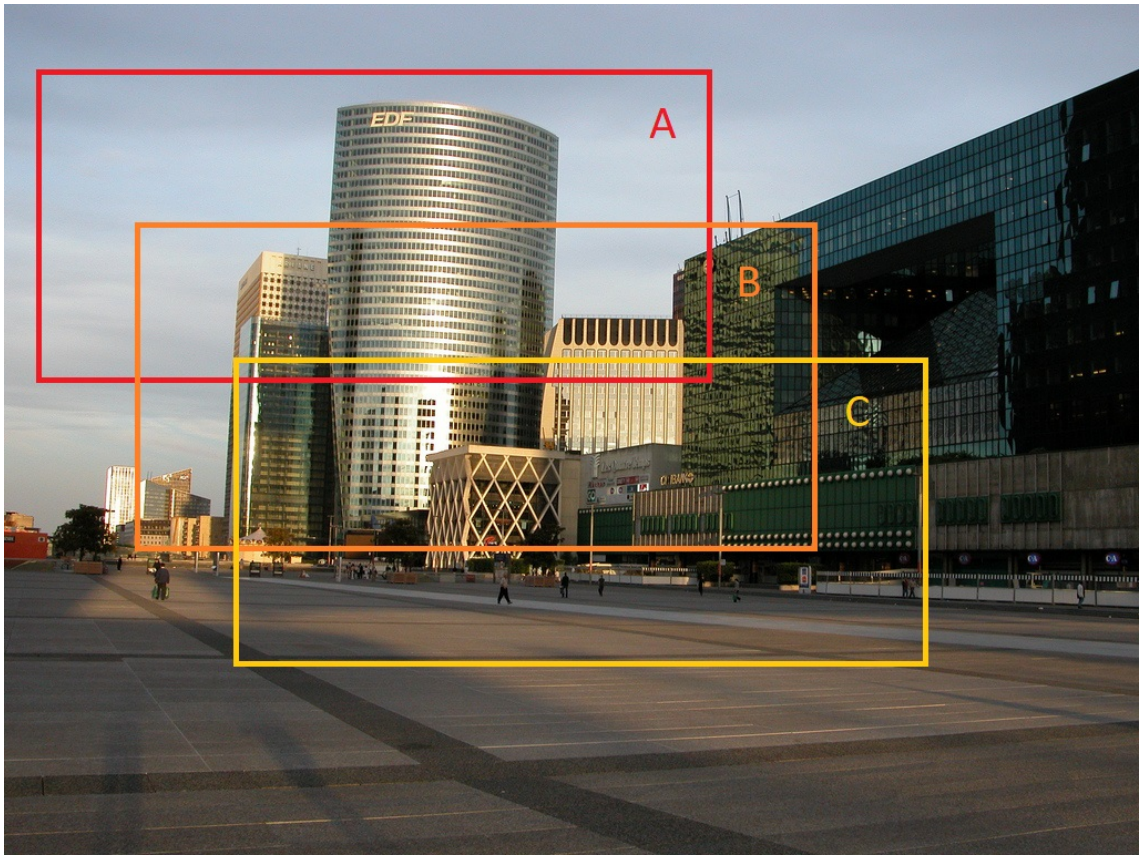

Figure 2.8: The example of query feature expansion. Image A, B, and C are parts of an image. Although they are from an image, their features are different. Query expansion enables a query feature to represent the three images.

in candidate features, enabling the new query feature to obtain more information than the original query feature and be robust to complicated situations. Then the new query feature is used to re-search candidate features to find the most related images. Furthermore, to filter out information from the negative features, we can use a linear Support Vector Machine (SVM) [19]. The global feature-based re-ranking methods based on query expansion have achieved outstanding performance [26, 72]. This step improves the accuracy by filtering out irrelevant features. Query expansion also implements spatial verification and unfixed iterations to improve accuracy [26, 27]. Query expansion uses spatial verification to verify results. Then query expansion repeats this process until it finds more than 30 verified images [26].

A wide range of image search methods implement query feature expansion as the global feature-based re-ranking [14,15,31,57,72–74]. Compared with local feature-based re-ranking methods, query feature expansion shows two significant advantages:

1. Query feature expansion uses the global features from the previous feature extraction step in the image search pipeline instead of extracting new features, reducing computing costs and improving the re-ranking speed.

2. Instead of using computationally demanding matching algorithms, query feature expansion absorbs information from candidate images to improve the robustness of the query feature, which dramatically simplifies the computation.

Therefore, query feature expansion is very suitable for processing large databases.

However, there is a significant disadvantage of query feature expansion. Although the new query feature is more robust to complex conditions in an image than the original query feature, it is not robust because it only absorbs information from a few candidate images [26, 27]. The new query feature only collects information about itself and some main objects in candidate features and can not comprehensively summarize all critical information, such as multi-object, illumination, viewpoint, and rotation [5, 21, 26, 27]. Therefore, the result of query feature expansion is not outstanding when the aforementioned situations are very different in gallery images.

We will use query expansion as a global feature-based re-ranking benchmark in experiments.

3 Proposed Re-ranking Methods

This section introduces two local feature-based re-ranking methods with different advantages: SIFT-AffNet-HardNet-AdaLAM (SAHA) and Local Feature Matching with Transformers (LoFTR), which can improve the accuracy of image search engines and are faster than traditional local feature-based re-ranking methods. We implement powerful local feature enhancement methods, an offline/online design, and transformer to build these two methods. Furthermore, we use a powerful global feature-based re-ranking method, called Query and Gallery Enhancement (QGE), which is very fast and can provide higher accuracy than local feature-based re-ranking methods. We illustrate the three methods in Figure 3.1.

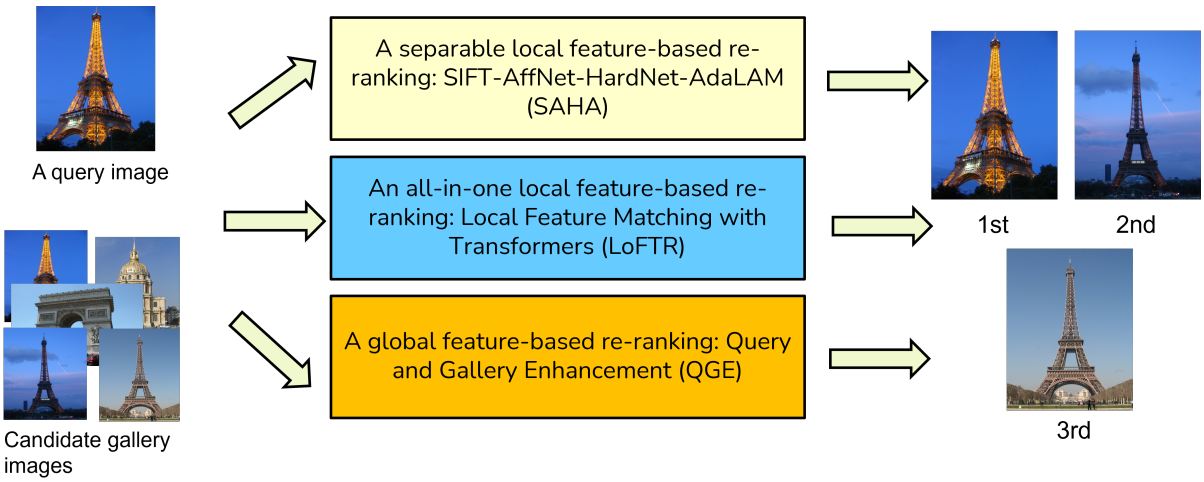

Figure 3.1: Three solutions. SAHA and LoFTR are local feature-based re-ranking methods. QGE is a global feature-based re-ranking method.

There are two essential parts in local feature-based re-ranking: feature extraction and matching. SAHA is a separable local feature-based re-ranking method, which means SAHA combines different modules to extract and match features. By contrast, LoFTR is an all-in-one local feature-based re-ranking method, which means LoFTR uses a single module for feature extraction and matching. And QGE is a global feature-based re-ranking method, which means it uses the global features from the feature extraction step in image search engines to implement re-ranking instead of extracting new local features.

3.1 SIFT-AffNet-HardNet-AdaLAM and the Offline/Online Design

In local feature-based re-ranking structures, many methods rely on SIFT as the feature detection and description methods to obtain keypoints and descriptors, respectively. We can process every image by SIFT to extract keypoints and descriptors. However, there is only a tiny improvement in SIFT, or SIFT-like algorithms, which leads to these algorithms can not describe images precisely as images are more and more complicated. Nevertheless, the keypoint detection provided by SIFT or SIFT-like algorithms is still valuable. Therefore, it is urgent to improve SIFT to perform more accurate feature descriptions in images. And we can implement feature enhancement methods on SIFT. We illustrate the structures of popular local feature-based re-ranking methods in Figure 3.2.

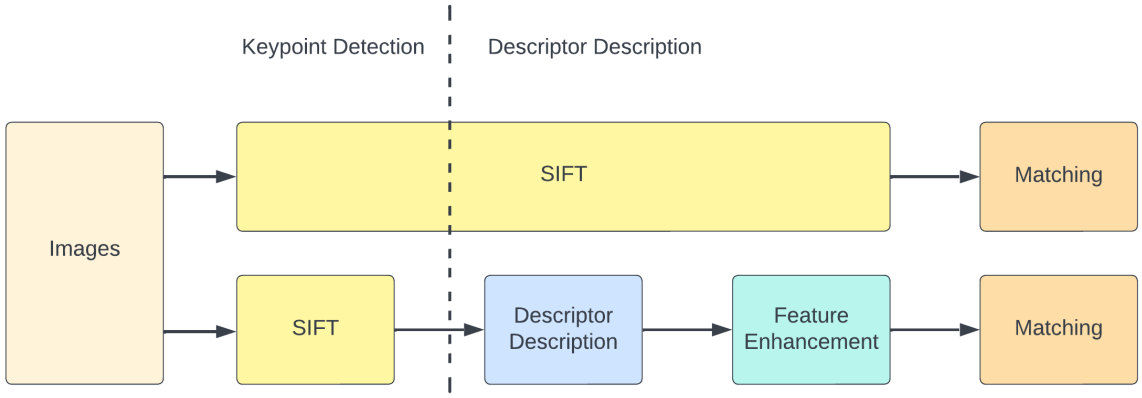

Figure 3.2: The difference between traditional SIFT-based local feature-based re-ranking and enhanced SIFT-based local feature re-ranking. The pipeline at the top shows the traditional SIFT-based feature extraction, and the pipeline at the bottom shows the enhanced SIFT-based feature extraction.

Therefore, we postulate that we can significantly improve local feature-based re-ranking by maintaining the keypoint detection of SIFT and replacing the description step of SIFT with other algorithms. Moreover, we can replace traditional matching methods, such as BF, with state-of-the-art matching methods to filter out outliers and match images better. Therefore, we aim to build a local feature-based re-ranking structure based on SIFT, called SIFT-AffNet-HardNet-AdaLAM (SAHA) [2, 3, 7, 8, 25] to achieve more accurate performance.

3.1.1 Feature Detection

As shown in Figure 3.2, SIFT is still a wise choice for keypoint detection. In Section 2, we have elaborated on how SIFT extracts keypoints and descriptors from an image. Therefore, this section will focus on how to enhance SIFT.

In the proposed local feature-based re-ranking structure, SIFT is the first step to extract preliminary keypoints from images, as shown in the Figure 3.3:

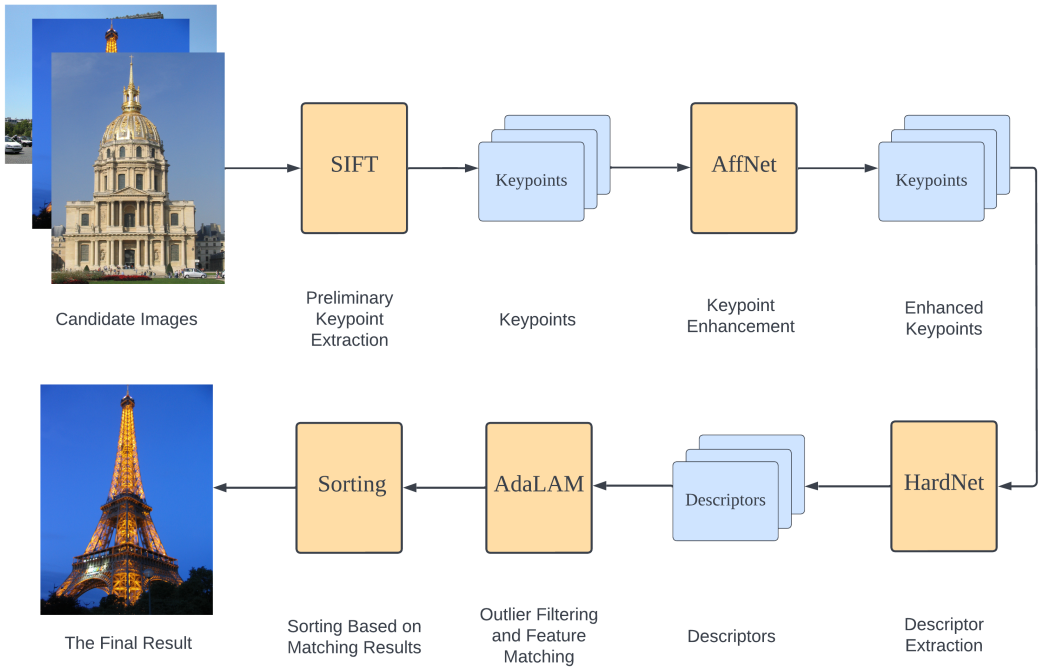

Figure 3.3: The structure of the SIFT-AffNet-HardNet-AdaLAM (SAHA) re-ranking. We use AffNet, HardNet and AdaLAM to enhance SIFT and use bubble sort to re-rank results.

3.1.2 Feature Enhancement

Feature enhancement refers to improving keypoints to be robust to illumination, viewpoint, and rotation changes in images. We introduce a powerful feature enhancement method, AffNet, that focuses on the affine shape and orientation parameters with CNN and hard negative-constant loss to extract the affine shape estimators in images to improve keypoints [7]. This subsection will introduce AffNet.

3.1.2.1 The Structure of AffNet

AffNet is a CNN-based method, and there are three critical components in AffNet: 1) a convolutional layer with a fixed size filter, 2) a padding layer, and 3) a Batch Normalization Layer with the activation function (ReLU), as shown in the Figure 3.4.

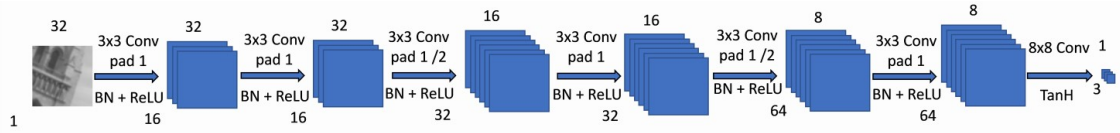

Figure 3.4: The structure of the AffNet [7].

The inputs of AffNet are patches P of an image and the related keypoints extracted by SIFT. The convolutional layer and the padding layer extract precise information from images and keypoints. Furthermore, the Batch Normalization layer keeps the learning rates and reduces the strong dependence on initialization. Finally, the ReLU function introduces non-linear complexities to the model. And there is a TanH activation function to maintain results smoother without the Batch Normalization layer or the padding layer in the last step.

And there are two critical theoretical components in AffNet: affine shape parametrization and loss function design.

3.1.2.2 Affine Shape Parametrization

The affine-covariance property of local features allows features to be robust to multi-object, illumination, viewpoint, and rotation changes. AffNet extracts the affine-covariance information from an image and implements the affine transformation matrix A to present the information [7]:

$$A = \begin{pmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{pmatrix}, \tag{3.1}$$

where μ_{ij} is the autocorrelation parameter of different keypoints i and j. The affine transformation is a geometric transformation that preserves lines and parallelism to map the geometric information of an image into an affine matrix [75, 76]. We often implement the affine matrix for local feature description and enhancement and adapt it to independent of any image resolution [77]. For an image A, the related affine matrix A contains much scale information. To obtain the residual shape information for improving the robustness of the features [75], we need to decompose the affine matrix A [7]:

$$A = \lambda R(\alpha) A^{'} = \det A \begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} \mu_{11}^{'} & 0 \\ \mu_{21}^{'} & \mu_{22}^{'} \end{pmatrix}, \tag{3.2}$$

where $\lambda = \det A$ refers to the scale parameter, $\mu_{ij}^{'}$ refers to the affine shape parameter of keypoints i and j, $R(\alpha)$ refers to the orientation matrix and $A^{'}$ refers to the residual affine shape matrix with the affine parameter $\mu_{ij}^{'}$ and $\det A^{'} = 1$ [7]. Then the residual affine shape matrix $A^{'}$ of the affine shape parametrization of decriptors can be obtained by Equation 3.3 [7]:

$$A^{'} = I + A^{''} = \begin{pmatrix} \mu_{11}^{'} & 0 \\ \mu_{21}^{'} & \mu_{22}^{'} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} \mu_{11}^{''} & 0 \\ \mu_{21}^{''} & \mu_{22}^{''} \end{pmatrix}, \tag{3.3}$$

where $A^{''}$ is the second-order residual shape matrix and $\mu_{ij}^{''}$ is the second-order affine parameter.

Finally, we obtain the affine shape parametrization of descriptors.

3.1.2.3 Loss Function

Next, we will use the obtained affine parameter in the training process.

First and foremost, AffNet proposes a loss function called hard negative-constant loss (HardNegC) [7]. HardNedC is an improved loss function of the hard negative triplet margin loss function. In the negative triplet margin loss function, there is an anchored descriptor (a correct descriptor), a positive descriptor (similar to the anchored descriptor) and a negative descriptor (an incorrect descriptor) used to calculate the distance between these descriptors. However, in HardNedC, the distance between the anchored descriptor and the closest negative descriptor is treated as constant. Therefore, the loss function is shown in Equation 3.4 and the distance is shown in Equation 3.5 [7]:

$$L = \frac{1}{n} \sum_{i=1,n} \max(0, 1 + d(s_i, \dot{s_i}) - d(s_i, s_N)), \tag{3.4}$$

$$d(s_i, s_N) = \min(\min_{j \neq i} d(s_i, \dot{s_j}), \min_{j \neq i} d(s_j, \dot{s_i})), \tag{3.5}$$

where n is the number of descriptors, s_i is the descriptor i, $d(s_i, \dot{s_i})$ is the distance between the two descriptors and $d(s_i, N)$ is a distance between the descriptor i and the hardest negative descriptor (the most negative descriptor) s_N in the mini-batch [7]. Therefore, the partial derivative of the loss function L equals 0 [7]:

$$\frac{\partial L}{\partial s_N} = 0. \tag{3.6}$$

If we apply the affine transformation to every patch in an image, we can combine the affine transformation matrix with the loss function [7]:

$$A(\theta | (P, \dot{P})) = arg\ \min_{\theta} L(s, \dot{s}), \tag{3.7}$$

where θ is the affine transformation model parameter and $(P, \dot{P})$ is a matching pair of two patches of the input image.

3.1.3 Feature Description

Descriptors can be extracted and enhanced after SIFT-based keypoints are extracted from an image and enhanced by AffNet. The CNN architecture has been proved to boost the performance of image description [5, 21, 60, 61]. We implement a descriptor description method based on CNN, called HardNet, that extracts descriptors in a compact way [8]. HardNet aims to mimic the SIFT-based description method by a CNN structure to achieve higher performance. Therefore, we can implement HardNet as the descriptor description method in the enhanced SIFT-based feature extraction.

3.1.3.1 The Structure of HardNet

We implement HardNet on a traditional CNN structure. In each convolutional layer, we use padding with zeros to preserve the spatial information around keypoints, and a convolutional operation to reduce the spatial size. Furthermore, we use the Batch Normalization layer and ReLU non-linearity in each convolutional layer. Dropout regularization with 0.1 dropout rate is implemented before the last convolution layer to resize the data. However, the final convolutional layer is different from other layers. The final convolutional layer has a Batch Normalization layer and an L2 normalization layer. The output of the HardNet is L2 normalized to produce 128-D descriptors with unit length. The Figure 3.5 explains that the structure of HardNet [8].

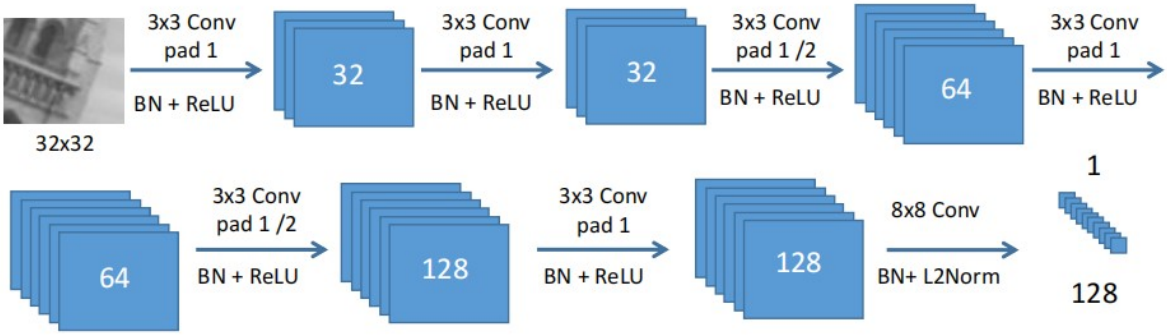

Figure 3.5: The structure of HardNet [8].

3.1.3.2 Loss Function

The learning objective of HardNet is to simulate the SIFT-based description. Therefore, HardNet proposes a sampling method for this process, as shown in Figure 3.6.

Specifically, a batch $X = (A_i, P_i)_{i=1..n}$ of matching local patches is created, where A is the anchored (target) patch and P is the positive (true) patch from the previous AffNet step [7,8]. Furthermore, the patches A and P are passed through the network for training, as shown in Figure 3.5. Descriptors a_i and p_j of A and P are extracted from patches by the network respectively. Then these descriptors are used for calculating the distance matrix D of every descriptor in Figure 3.6. The distance matrix D is an L2 pairwise distance matrix [8]:

$$D = \mathrm{c} * \mathrm{dist}(a, p), \tag{3.8}$$

$$d(a_i, p_j) = \sqrt{2 - 2a_i p_j}, i = 1...n, j = 1...n. \tag{3.9}$$

Next, HardNet needs to search the closest non-matching descriptors of each matching pair a_i and p_i to obtain more information for training. In Figure 3.6, $p_{j_{min}}$ is the closest non-matching descriptor to a_i [8]:

$$j_{min} = \arg \min_{j=1...n, j \neq i} d(a_i, p_j). \tag{3.10}$$

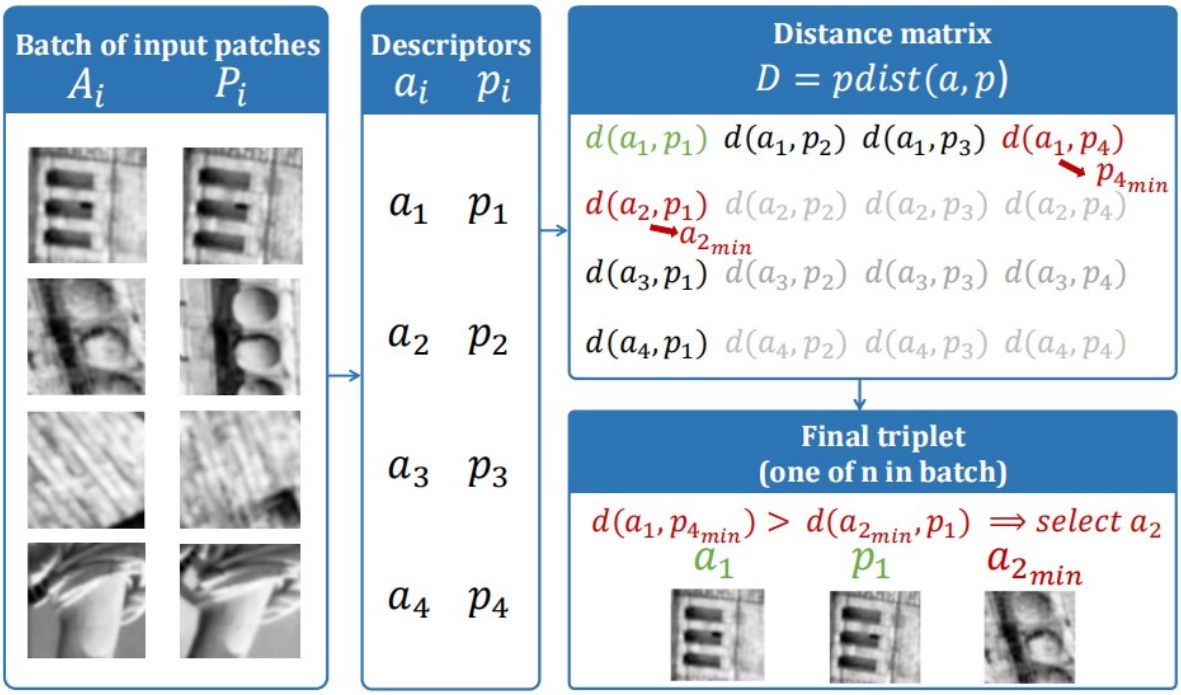

Figure 3.6: The sampling of HardNet [8]. We use patches to extract descriptors. Then we can obtain the distance matrix of descriptors. Finally, we can obtain a triplet.

And $a_{k_{min}}$ is the closest non-matching descriptor to p_i [8]:

$$k_{min} = \arg \min_{k=1..n, k \neq i} d(a_k, p_i). \tag{3.11}$$

Finally, a triplet information $(a_i, p_i, p_{j_{min}})$ is introduced from each quadruplet of descriptors $(a_i, p_i, p_{j_{min}}, a_{k_{min}})$. And the another triplet information $(p_i, a_i, a_{k_{min}})$ is introduced in the same way. The purpose of calculating the triplet information $(a_i, p_i, p_{j_{min}})$ and $(p_i, a_i, a_{k_{min}})$ is to obtain the minimal distance of $(a_i, p_{j_{min}})$ and $(a_{k_{min}}, p_i)$ respectively [8].

Moreover, we can obtain the triplet margin loss in Equation 3.12 [8]:

$$L = \frac{1}{n} \sum_{i=1,n} \max\left(0, 1 + d(a_i, p_i) - \min(d(a_i, p_{j_{min}}), d(a_{k_{min}}, p_i))\right). \tag{3.12}$$

Finally, we have built the structure and training process of HardNet. We implement HardNet as the feature description step.

3.1.4 Feature Matching

Local feature filtering and matching are critical to the local feature-based re-ranking method. Although the feature extraction and feature enhancement algorithms have improved the keypoints and descriptors, there are still many outliers in the matches.

The outliers cause wrong matches in a pair of two images that will mislead the sorting result.

For example, there is a query image A and two candidate images B and C. A and B are not similar. However, there are many outliers in descriptors of A and B, which erroneously results in a high number of matches of A and B. A and C are very similar. But their number of matches is not as high as that of A and B. Thus, this misleads image search engines to determine that B is more similar to A than C, and rank B before C.

We have introduced popular off-the-shelf matching methods, and there are many limitations in those methods, such as the applicability, accuracy, and speed. Therefore, we implement a more balanced feature filtering and matching method, Adaptive Locally-Affine Matching (AdaLAM), to perfectly solve the feature matching problem. We will introduce three main components in AdaLAM: 1) core assumptions, 2) seed point selection, and 3) feature selection, filtering, and validation.

3.1.4.1 Core Assumptions

Based on the keypoints k_1 and k_2 and the descriptors d_1 and d_2 of images I_1 and I_2 respectively, we can compute the putative nearest neighbor matches m of I_1 and I_2, where nearest neighbors are computed from the keypoint pair of k_1 and k_2 and the descriptor pair of d_1 and d_2 [22, 25]. Therefore keypoints and descriptors are important for accurate matching. Although there are AffNet for enhancing keypoints and HardNet for precise descriptor extracting, there are still many incorrect matches due to limitations in the outlier filtering. Therefore, outlier filtering is necessary for correct matching. AdaLAM aims to provide a subset $m_0 \subseteq m$ that contains as many correct inliers as possible. AdaLAM implements outlier filtering on orientation and scale to prune wrong matches and limits the final matches M to a fixed-size subset of preliminary nearest neighbor matching results of keypoints and descriptors.

The matches show similarities between the two images. To better approximate the matches, we implement an affine transformation A again. This affine transformation enables keypoints and descriptors robust to multi-object, illumination, viewpoint, and rotation changes. However, this assumption can be weakened by the following situations:

1. The main objects in an image are not always planar, which leads to non-linear deviation in the affine transformation.

2. The keypoints may not be near each other, increasing the distortion of the affine transformation.

3. The matches may not represent the projection of three-dimensional objects in images caused by rotation and viewpoints changes.

Therefore, AdaLAM improves the outlier filtering and feature matching based on these three situations (assumptions).

3.1.4.2 Seed Points Selection

As affine transforms A represents local transformations around keypoints k_i, called hypotheses seed points, we can use the nearest neighbor correspondences to guide the nearest neighbor search for correct candidate keypoints. Moreover, AdaLAM uses the ratio test between the first two nearest neighbors [2, 3]. By the ratio test, we can find the seed point with the highest score within neighbors. Furthermore, we use the distance D between the seed point and its neighbors in feature selection, filtering, and validation.

3.1.4.3 Feature Selection and Filtering

We need to obtain a threshold T to find the correct descriptors. The threshold T aims to filter descriptors. If the distance between the candidate descriptor and the anchored descriptor is greater than the threshold T, this descriptor is considered an outlier and vice versa. AdaLAM sets a similarity score c to describe the distance. The larger the distance, the lower the score. Therefore, the score c is the core of feature selection and filtering.

To obtain c of every descriptor, AdaLAM sets $S_i = (x_1^{Si}, x_2^{Si})$ represent the anchored descriptor correspondence. In AdaLAM, the affine transformation is shown in Equation 3.13 and 3.14 [25]:

$$\alpha^{S_i} = \alpha_1^{S_i} - \alpha_2^{S_i}, \tag{3.13}$$

$$\theta^{S_i} = \frac{\theta_1^{S_i}}{\theta_2^{S_i}}, \tag{3.14}$$

where α^{S_i} refers to the orientation component of the anchored descriptor i and θ^{S_i} is the scale component of the anchored descriptor i.

For the orientation component and the scale component, there are thresholds T_α and T_θ respectively. These two thresholds measure the difference between candidate descriptors and the anchored descriptor S_i. The preconditions of the correspondence are shown in Equation 3.15, 3.16, 3.17, and 3.18 [25]:

$$\|x_1^{S_i} - x_1\| \leq \lambda D_1, \tag{3.15}$$

$$\|x_2^{S_i} - x_2\| \leq \lambda D_2, \tag{3.16}$$

$$|\alpha^{S_i} - \alpha^p| \leq t_\alpha, \tag{3.17}$$

$$|\ln(\frac{\theta^{S_i}}{\theta^p})| \leq t_\theta, \tag{3.18}$$

where D_1 and D_2 are the distance between the anchored descriptor, and λ is a hyper-parameter that regulates the overlap between inclusion neighborhoods [25].

Then, the correspondence (k_1, k_2) of keypoints k_1 and k_2 can be obtained and are shown in Equation 3.19, 3.20 and 3.21 [25]:

$$(k_1, k_2) = ((x_1, d_1, \theta_1, \alpha_1), (x_2, d_2, \theta_2, \alpha_2)), \tag{3.19}$$

$$\alpha^p = \alpha_1 - \alpha_2, \tag{3.20}$$

$$\theta^p = \frac{\theta_1}{\theta_2}. \tag{3.21}$$

According to geometric verification [78], an affine transform hypothesis A_i^j can be set. A_i is an affine transformation and j is the number of iterations. As the filtering threshold of geometric verification is related to the number of outliers, AdaLAM calculates each hypothesis based on multiple thresholds $t_1 ... t_n$ to find out the most applicable threshold in most situations [25]. Therefore, the residuals R with respect to A_i^j, the corresponding inlier set P and the hypothesis scoring function c for a correspondence (x_1, x_2) can be computed [25]:

$$R(A_i^j, x_1, x_2) = \|A_i^j x_1 - x_2\|, \tag{3.22}$$

$$P_i^j(t_k) = \left\{ (x_1, x_2) \in N_i \middle| R(A_i^j, x_1, x_2) \leq t_k \sqrt{|\det(A_i^j)|} \right\}, \tag{3.23}$$

$$c = \begin{cases} 0 & \text{if } \det(A_i^j) \geq t_\theta^2 \ \forall \ \det(A_i^j) \leq \frac{1}{t_\theta^2} \\ |P_i^j(t_k)| & \text{otherwise} \end{cases}, \tag{3.24}$$

where $|P_i^j(t_k)|$ is the number of inliers in the set P. So the score c is maximized by the affine transformation for each seed point S_i and for each threshold $t_k \in t_1 ... t_n$. And the highest scoring inlier set $P_i(t_k)$ can be obtained by the least squares solution A_{i,t_k} [25].

As we found the inlier set, we can accomplish the outlier filtering. Outliers in the descriptors of all images are filtered out. We consider descriptors of the top-k images close to descriptors of the query image in the Euclidean distance to be the top-k similar images.

3.1.5 Sorting

Finally, we obtain the number of matches between the query image and all candidate images. Then we implement the bubble sort [79] to sort these candidate images. In this task, there is a sequence S to save the results of the numbers of matches of all pairs. Each number is an element in the series S.

Bubble sort repeatedly walks through all elements (the numbers of matches of all pairs of the query image and candidate images) in the sequence, comparing two elements and swapping them if they are in the wrong order. This process is repeated through the series until no more exchanges are needed. The "bubble sort" comes from the fact that the more minor elements are slowly "floated" to the top of the sequence by swapping. We illustrate the pseudo-code in Algorithm 1.

Algorithm 1 Bubble Sort

Sort First Two Elements:
First and foremost, bubble sort compares adjacent elements m_1 and m_2 in the sequence S, where m_i is the number of matches between the query image and the candidate image i. If the first element m_1 is larger than the second element m_2, bubble sort swaps them.
Sort Each Pair:
Furthermore, bubble sort implements the same comparison for each pair of adjacent elements, from the first pair at the beginning to the last pair at the end.
repeat
Sort All Elements:
Bubble sort repeats the above steps for all elements except the last one.
until if all elements are sorted

The advantages of bubble sort are:

1. Bubble sort is stable, which means the results are always the same when two pairs have the same number of matches in the sequence. However, some algorithms will offer different results in this case, which means these algorithms are not stable.

2. The space complexity of bubble sort is very low, only $O(1)$ [80].

The disadvantage of bubble sort is that the time complexity of bubble sort is high, which is $O(n^2)$ [80].

After bubble sort, the higher the number of matches between a candidate image and the query image, the higher the ranking of this candidate image. Finally, the SAHA finished re-ranking.

3.1.6 Offline/Online Design

We design an offline/online structure for SAHA to speed up SAHA. We illustrate the structure of SAHA with the offline/online design in Figure 3.7 and 3.8.

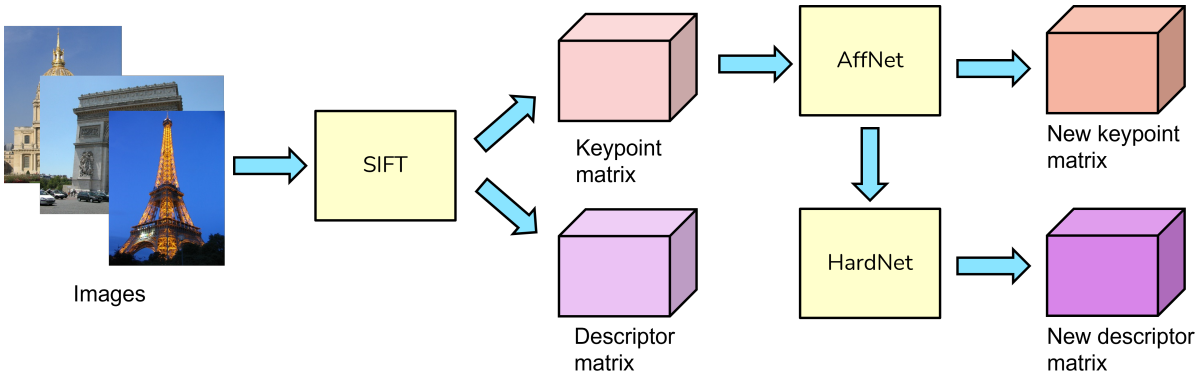

Figure 3.7: The offline step of SAHA. We use AffNet and HardNet to enhance SIFT. Then we save features.

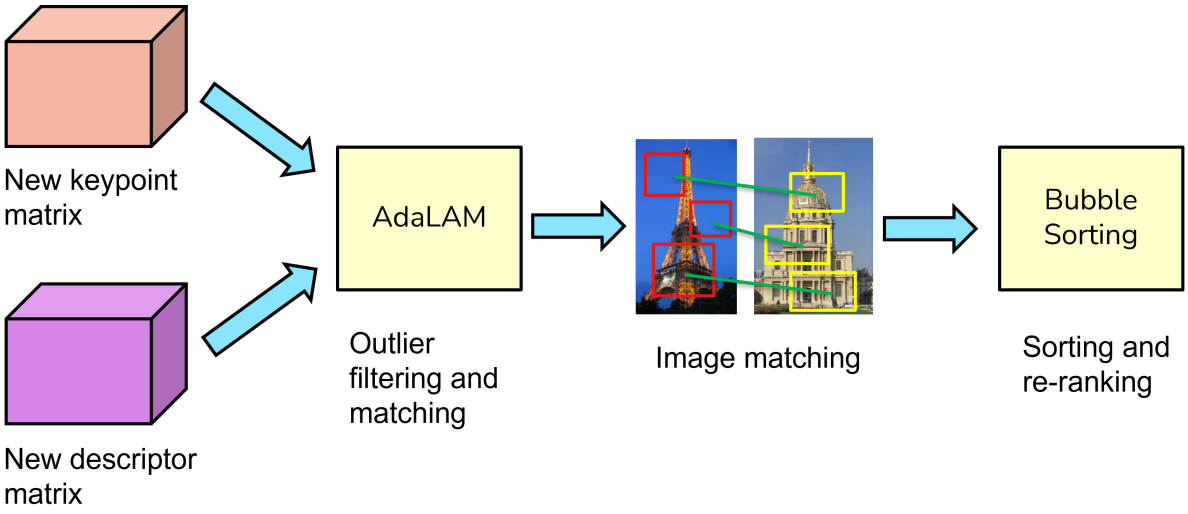

Figure 3.8: The online step of SAHA. We use AdaLAM to match features and implement bubble sort to re-rank results.

In the offline step, keypoints and descriptors of all images are extracted by SIFT, AffNet, and HardNet and saved in the memory. Then all keypoints and descriptors are exported from the memory and analyzed by AdaLAM in the online step. We implement AdaLAM to filter outliers, match features, and execute bubble sort to complete the ranking. Since image search engines only need to implement AdaLAM and bubble sort in the online step, which significantly speeds up the speed of SAHA. Furthermore, we can use this strategy for SAHA and other local feature-based re-ranking methods that adopt the separable design.

The advantages of SAHA with the offline/online design are as follows:

1. **Accuracy:** SAHA is more accurate than traditional SIFT-based re-ranking methods by enhancing features with AffNet and HardNet and enhancing outlier filtering and matching with AdaLAM.
2. **Speed:** SAHA is faster than traditional SIFT-based re-ranking methods by the offline/online design.

However, this method also has disadvantages: SAHA needs to store the feature information of all images from the offline stage, which leads to a significant increase in memory. Therefore, the offline/online design is a way to trade memory for speed.

3.1.7 Contributions

Our contributions in SAHA are as follows:

1. We use AffNet, HardNet and AdaLAM to improve feature extraction and matching of SIFT. Therefore, SAHA is more accurate than SIFT.
2. SAHA can not sort matching results. We use bubble sort after SAHA to achieve accurate sorting.

3. We use the offline/online design to accelerate SAHA.

3.2 LoFTR

If a user needs to use local feature-based re-ranking with a memory limitation, an all-in-one local feature-based re-ranking method, called Local Feature Matching with Transformers (LoFTR), is more suitable than SAHA. LoFTR is an existing image matching technology for analyzing the similarity of two images, including local feature extraction, outlier filtering, and matching. We have applied it for a new task, image search.

We implement LoFTR as an all-in-one feature processing module in the local feature re-ranking. Although LoFTR is slower than SAHA, it requires lower memory space. We illustrate the pipeline of the LoFTR re-ranking in Figure 3.9.

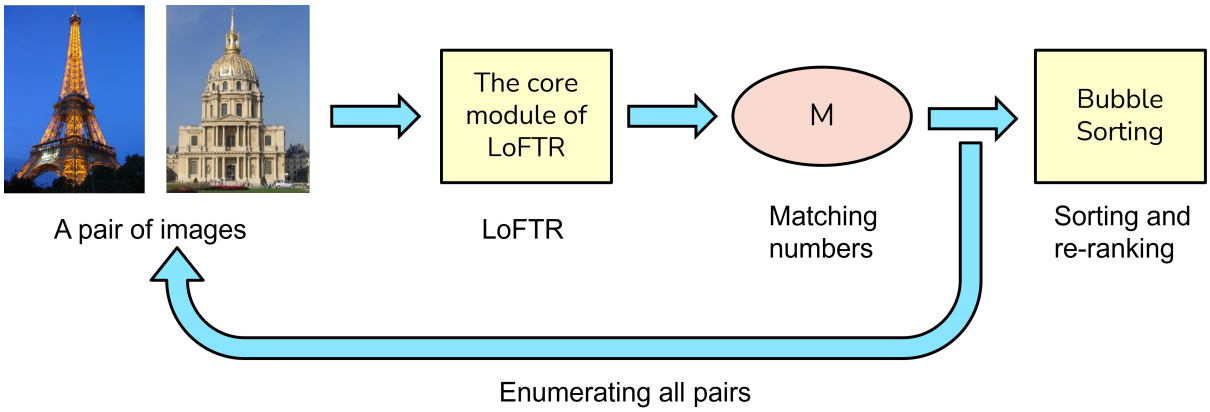

Figure 3.9: The pipeline of the LoFTR re-ranking. We use LoFTR to obtain matching numbers of images. Then we implement bubble sort to re-rank results.

LoFTR uses an all-in-one structure to perform feature extraction and feature matching of a pair of images. We illustrate the all-in-one structure of LoFTR in Figure 3.10. We will illustrate how to use LoFTR to implement an accurate all-in-one local feature-based re-ranking method to complete the following steps:

1. Local Feature Extraction,
2. Local Feature Transformer Module,
3. Coarse-level Matching Module,
4. Coarse-to-Fine Module,
5. Supervision,
6. Sorting.

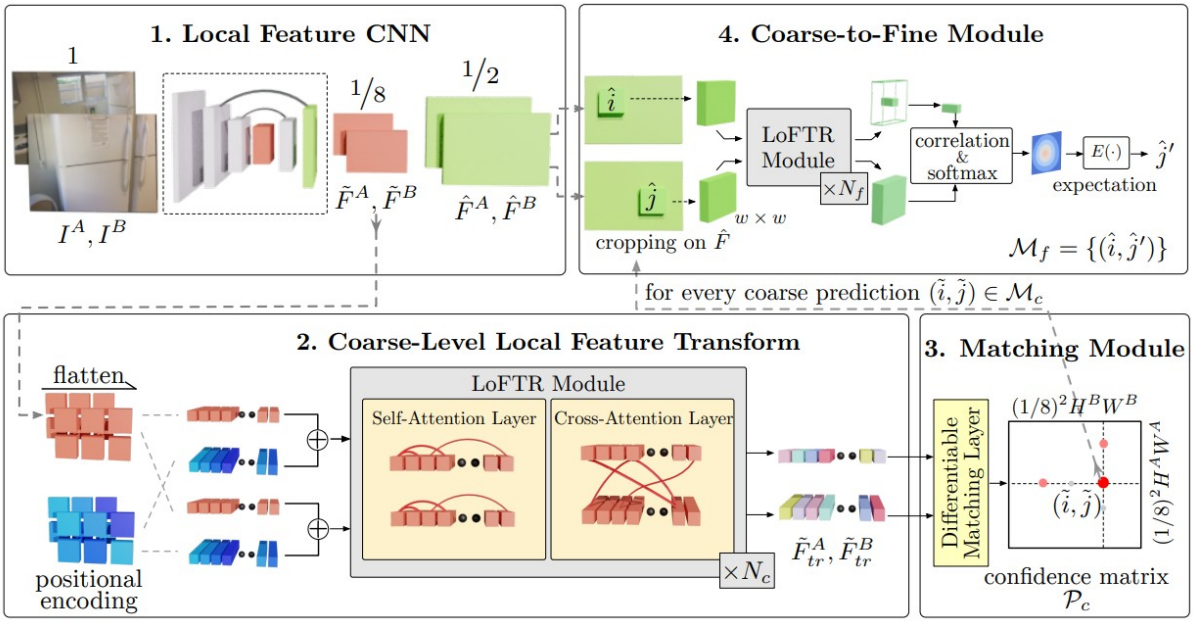

Figure 3.10: The structure of LoFTR [9]. There are four steps: local feature CNN, coarse-level local feature transform, matching module, and coarse-to-fine module.

3.2.1 Local Feature Extraction

LoFTR implements feature extraction without feature detection to improve the feature matching speed. Also, the query image and a candidate image must be input into LoFTR simultaneously to compare the two images more accurately.

As shown in Figure 3.10, LoFTR uses CNN as the backbone to extract the features of the two input images I_1, I_2. Different from traditional methods, LoFTR only extracts descriptors in images. LoFTR uses CNN to crop the input images into different sizes of images: $I_1^{\frac{1}{8}}$ and $I_2^{\frac{1}{8}}$ with the one-eighth size of the original image, and $I_1^{\frac{1}{2}}$ and $I_2^{\frac{1}{2}}$ with the one-half size of the original image. Then these new images are extracted for the corresponding one-eighth features $F_1^{\frac{1}{8}}$ and $F_2^{\frac{1}{8}}$ and the one-half features $F_1^{\frac{1}{2}}$ and $F_2^{\frac{1}{2}}$.

3.2.2 Local Feature Transformer Module

Then, LoFTR uses a transformer [81] to coarsely process the one-eighth features $F_1^{\frac{1}{8}}$ and $F_2^{\frac{1}{8}}$ to obtain the heaviest features. The encoding layer converts features into query vectors, key vectors, and value vectors [81]. LoFTR not only uses the encoding layer to extract three vectors but also uses the positional encoding method to give each element unique position information in the sinusoidal format. By adding the position encoding to $F^A(\frac{1}{8})$ and $F^B(\frac{1}{8})$, these features will become position-dependent to facilitate feature matching. After that, these features are processed by self-attention and cross-attention layers [9, 81]. After N_c times operations, we can obtain the feature points $\tilde{F}^A_{tr}$ and $\tilde{F}^B_{tr}$ after rough processing.

3.2.3 Coarse-level Matching Module

The next step is the coarse-level matching with a dual-softmax operator. LoFTR uses a score matrix S between the transformed features to calculate the probability of soft mutual nearest neighbor matching between $\tilde{F}_{tr}^{A}$ and $\tilde{F}_{tr}^{B}$. Therefore, the score matrix S and the matching probability P_c are shown in Equation 3.25 and 3.26 [9]:

$$S(i,j) = \frac{1}{\tau} \left\langle \tilde{F}_{tr}^{A}(i), \tilde{F}_{tr}^{B}(j) \right\rangle, \tag{3.25}$$

$$P_c(i,j) = \text{softmax}(S(i,x))_j * \text{softmax}(S(y,j))_i, \tag{3.26}$$

where x and y are elements in the one-eighth and one-half dimensions respectively. Based on the confidence matrix P_c, LoFTR implements the mutual nearest neighbor (MNN) criteria to filter coarse outliers and uses final matches higher than a threshold of θ_c. Equation 3.30 shows the matching [9]:

$$M_c = \left\{ (\tilde{i},\tilde{j}) \mid \forall (\tilde{i},\tilde{j}) \in \text{MNN}(P_c), P_c(\tilde{i},\tilde{j}) \geq \theta_c \right\}. \tag{3.27}$$

3.2.4 Coarse-to-Fine Module

Next, by the coarse-to-fine step, we process these matches and the features $F^{\frac{1}{2}}$. Every match (i,j) is located and then cropped into two sets of local windows of size $w * w$ at fine-level feature maps. A LoFTR module transforms the cropped feature maps by N_f times to obtain two transformed local feature maps F^A and F^B. Then, we use correlation and softmax to compute these matches and the expectation of features $F^{\frac{1}{2}}$. Furthermore, LoFTR correlates F^A with F^B to obtain a heatmap with the matching probability of each feature. By computing expectation over the probability distribution, the final fine-level matches M_f are obtained [9].

3.2.5 Supervision

LoFTR implements two loss functions to supervise the training process.

The first loss function is the coarse-level function [9]:

$$L_c = -\frac{1}{|M_c^{gt}|} \sum_{(i,j)\in M_c^{gt}} \log P_c(\tilde{i},\tilde{j}), \tag{3.28}$$

where M_c^{gt} is the ground-truth coarse matches for calculating the mutual nearest neighbors.

The second loss function is the fine-level function [9]:

$$L_f = \frac{1}{|M_f|} \sum_{(\hat{i},\hat{j}')\in M_f} \frac{\|\hat{j}' - \hat{j}'_{gt}\|_2}{\sigma^2(\hat{i})}, \tag{3.29}$$

where $\sigma^2(\hat{i})$ is the total variance of the corresponding heatmap.

Finally, the total loss function is [9]:

$$L = L_c + L_f. \tag{3.30}$$

3.2.6 Sorting

LoFTR analyzes two images simultaneously: one of which is the query image and the other is a candidate image. Therefore, LoFTR can only calculate the similarity between the query image and all candidate images one by one and then obtain the numbers of matches of all pairs. After getting the numbers of matches of all pairs, we use the same bubble sort as SAHA to obtain the re-ranking result.

The advantages of LoFTR-based re-ranking are as follows:

1. **Accuracy:** Through a unified feature extraction and matching module, LoFTR-based re-ranking has higher coherence, which improves its accuracy.

2. **Memory:** Compared to SAHA, there is no high memory requirement in LoFTR.

However, there is a disadvantage of LoFTR: it is relatively slow because of its heavy structure. The transformer, the differentiable matching layer, the correlation and softmax layer, and cropping operations in the LoFTR module make it computationally demanding. Therefore, LoFTR-based re-ranking is not suitable for large databases.

According to the previous analysis, both SAHA and LoFTR are unsuitable for large databases because of the memory or computational costs. Therefore, we must provide a fast, light, and accurate re-ranking method.

3.2.7 Contributions

Our contributions in LoFTR are as follows:

1. LoFTR is an existing local feature-based image matching method to analyze a pair of images. We implement it in image search. Its accuracy is higher than previous local feature-based re-ranking methods.

2. LoFTR can not sort matching results. We use bubble sort after the LoFTR module to achieve accurate sorting.

3.3 Query and Gallery Enhancement

We use a powerful global feature-based re-ranking method, called Query and Gallery Enhancement (QGE), which is much faster and lighter than the local feature-based re-ranking methods, SAHA and LoFTR, and can obtain higher accuracy. We design QGE for image search engines in large databases.

QGE is based on query expansion [26, 27] and gallery diffusion [28, 29], absorbs the advantages of these methods and corrects their shortcomings. QGE uses the query expansion method to make the obtained query features more representative and expand gallery features to overcome the robustness problem in the previous global feature-based re-ranking methods.

Instead of extracting new features, QGE uses the global features obtained in the previous feature extraction step for re-ranking. There are two essential steps in QGE: Query Enhancement and Gallery Enhancement. This section will introduce these two parts.

3.3.1 Query Enhancement

We implement query expansion [26, 27] to enhance a query feature. Query expansion refers to expanding the query feature to make it more representative by absorbing information from a few candidate features.

In the previous steps of image search engines, we extract the feature of the query image. However, this feature can only represent the main object, illumination, viewpoint, and rotation information in the query image. If the primary object information in a candidate image is similar to that in the query image while the illumination, viewpoint, and rotation situations are different, NN search could decide that this candidate image is different from the query image. Therefore, we need to make the query feature contain more information, making it more robust to multi-object, illumination, viewpoint, and rotation situations.

The way to make the query feature contain more background information is using information from candidate features to enhance the query feature. First and foremost, after feature extraction and NN search, the feature Q of the query image and the features $G_1...G_i...G_n$ of the top-n candidate images are obtained, where i refers to the index of candidate images. We select the top-k candidate features to expand the query feature and build a new query feature $Q^{'}$.

Specifically, the new query feature $Q^{'}$ is constructed by weighted averaging the original query feature Q and top-k candidate features $G_1...G_k$:

$$Q^{'} = \frac{1}{\|Q + \sum_{i=1}^{k} w_i G_i\|_{L_2}}(Q + \sum_{i=1}^{k} w_i G_i), \tag{3.31}$$

$$w_i = (\frac{1}{i})^{\alpha}, \tag{3.32}$$

where $Q^{'}$ refers to the new query feature after normalization, Q refers to the original query feature, G_i refers to the i-th candidate feature, w_i refers to the weighted parameter for the candidate features, and α refers to a parameter for resizing the weight. According to diffusion [28, 29], increasing the weight of candidate features enables the new query feature to absorb more information. Also, we implement the L2 normalization to ensure the norm of query and candidate features to be consistent. This step combines the original query feature and candidate features together well while query expansion only uses arithmetic mean to build a new query feature.

Moreover, we implement a recursive expansion to improve the robustness of $Q^{'}$. Specifically, we implement the weighted averaging on Q with iterations. After the first iteration, a new query feature $Q^{'}$ is obtained and used as the input in the second iteration. We can obtain new candidate features $G_i^{'}$ by NN search for the new query feature. In order to further improve the accuracy, we use $Q^{'}$ as an independent variable to obtain $Q^{''}$:

$$Q^{''} = \frac{1}{\|Q^{'} + \sum_{i=1}^{k} w_i G_i^{'}\|_{L_2}} (Q^{'} + \sum_{i=1}^{k} w_i G_i^{'}), \tag{3.33}$$

$$w_i = (\frac{1}{i})^{\alpha}, \tag{3.34}$$

where $Q^{''}$ refers to the new query feature after the second iteration, $Q^{'}$ refers to the query feature after the first iteration, and $G_i^{'}$ refers to the i-th candidate feature for the query feature $Q^{'}$. Iterations can enhance the robustness of the query feature and expand the structure of candidate features to enable them to detect more similar gallery features [28, 29, 31]. Therefore, we can detect more candidate features similar to the query feature in the gallery features. This process can expand the candidate features and enhance gallery features. Query expansion uses unfixed iterations until it finds enough correct results. This process is slow because query expansion needs to enumerate many results. We implement a fixed number of iterations to achieve a balance of speed and accuracy. By the iteration process, we can find more and more correct gallery features to enhance the gallery.

After the iteration process, the final query feature Q^F combines the characteristics of the original query feature Q and the candidate features G, which has higher compatibility and can represent more information. Figure 3.11 shows the initial state before query enhancement.

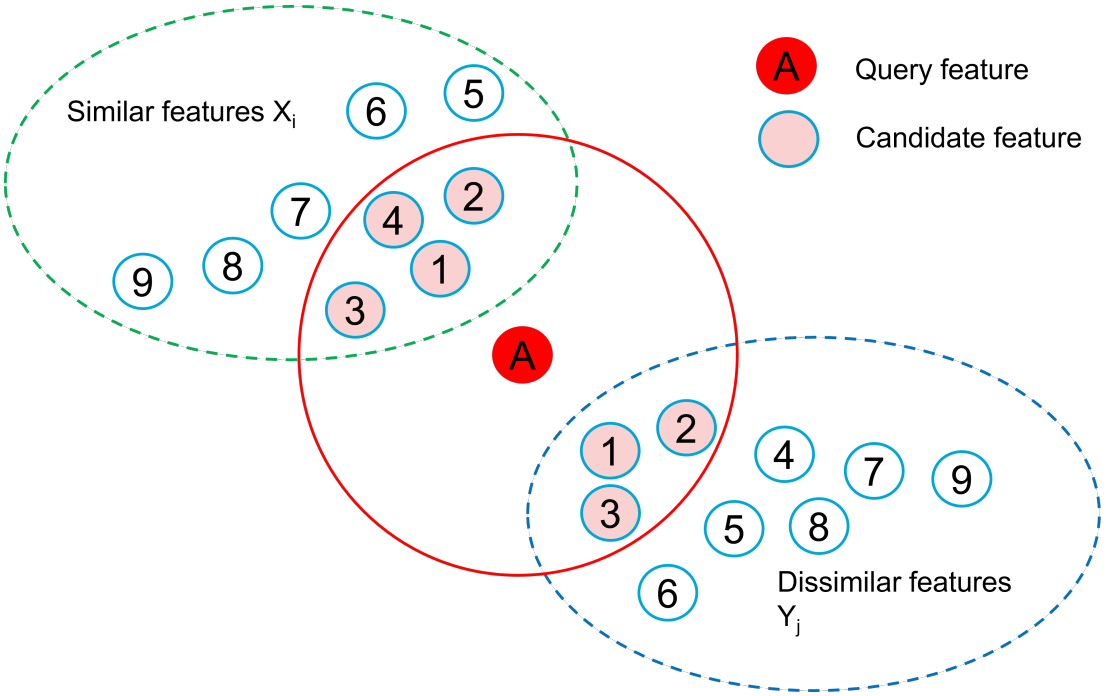

Figure 3.11: The preliminary search. There are similar and dissimilar features in candidate features. Preliminary results (candidate features) are in the red circle.

There is a query feature A (the small red point), features X_i similar to A, and features Y_j dissimilar to A. We extract features from corresponding images. The

features X_i similar to A are in the green circle, and the features Y_j dissimilar to A are in the blue circle. The main objects of X_i are the same as the main object of A, and the main objects of Y_j differ from the main object of A. If a point is close to the Euclidean distance of the query feature, it is likely to be considered similar to the query feature by image search engines. Theoretically, all X_i should be regarded as similar to A by image search engines since these features have the same primary object. However, since each feature contains different information of illumination, viewpoint, and rotation, some X_i are far from A, and some Y_j are close to A. In both X_i and Y_j, features that are considered candidate features are filled in pink and are in the big red circle. Therefore, there are features that are dissimilar to A in candidate features. To solve this problem, we implement query enhancement, as shown in Figure 3.12.

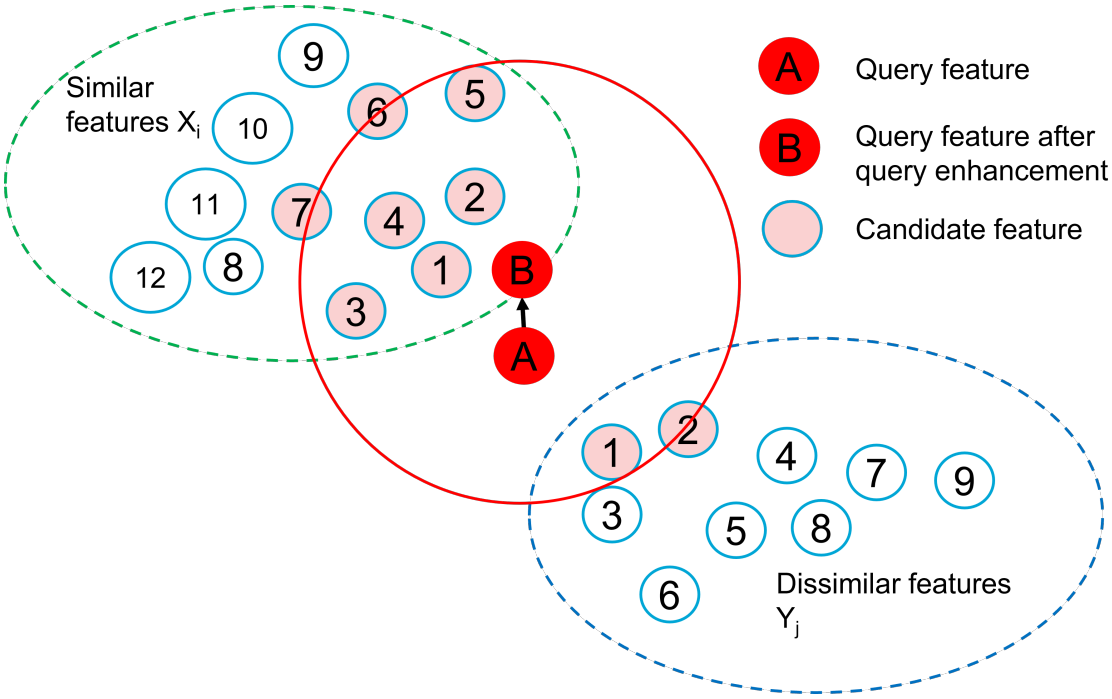

Figure 3.12: Query enhancement. The new query feature B is closer to similar features than the original query feature A.

Through query enhancement, a new query feature B is built based on the query feature A and the candidate features. The new query feature B is closer to similar features X_i because there are more X_i than Y_j in candidate features. As B is closer to similar features, more X_i are considered candidate features and detected as potential candidate features. Moreover, less Y_j are considered candidate features. Therefore, the results after query enhancement are more accurate and the gallery features are initially enhanced.

3.3.2 Gallery Enhancement

After enhancing the query feature, we continue enhancing the gallery features. The gallery features are features of all database images. We implement a graph construction

method, diffusion [28,29], to expand the relationship between every gallery feature and further enhance the gallery.

There are six main steps in gallery enhancement:

1. Graph construction,
2. Random walk,
3. Decomposition,
4. Offline construction,
5. Truncation,
6. Online search.

First and foremost, we set the database as $G = \{G_1...G_i...G_n\}$, where each G_i is the i-th gallery feature of the image I. And there is a final query feature $Q^{'}$ from the previous step, query enhancement.

We define the full feature set in graph construction as $E = \{Q^{'}, G_1...G_i...G_n\}$, and we denote i-th element in E as E_i. We continuously use the affine matrix A to calculate the correlations between every element in E, and the element in A is [29]:

$$a_{ij} = \begin{cases} s(E_i, E_j), & i \neq j \\ 0 & \text{otherwise} \end{cases}, \tag{3.35}$$

where $\forall i, j \in \{1, ..., n+1\}$, and s is the Euclidean similarity. Then we define the degree matrix D to be a diagonal matrix to symmetrically decompose A into the stochastic matrix S [29]:

$$S = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}, \tag{3.36}$$

where S is a variant of the typical transition matrix $D^{-1}A$ with the same eigenvalues and eigenvectors [29], and each diagonal element d_{ii} in D is $\sum_{j=1}^{n+1} a_{ij}$. The stochastic matrix S describes a graph of the query feature and gallery features.

Secondly, we implement a random walk to reach a convergence state to calculate adjacent dependencies (similarities) in the graph [28, 29]. The initial state starts from the query feature, walks randomly along the path in the previously established graph, and continuously updates its state. The random walk aims to re-analyze the correlation of each point in the graph, enhancing the correlation of the gallery features with the query feature. The initial state of all features is $F^0 = [{F_Q^0}^T, {F_G^0}^T]^T$. We illustrate the iteration process of the random walk in Equation 3.37 [29]:

$$F^{t+1} = \alpha S F^t + (1-\alpha) F^0, \ \alpha \in (0,1), \tag{3.37}$$

where α is the probability of the random walk. Finally, the stable state is in a closed-form solution [29]:

$$F^* = (1-\alpha)(I - \alpha S)^{-1} F^0, \ \alpha \in (0,1). \tag{3.38}$$

After convergence, the final state F^* contains the similarities of gallery features to the query feature.

Decomposition aims to extract re-ranking information about gallery features by the matrix S and the final state F^* [29]:

$$S = \begin{bmatrix} S_{QQ} & S_{QG} \\ S_{GQ} & S_{GG} \end{bmatrix}, \tag{3.39}$$

where S_{QQ} refers to the autocorrelation of the query feature, S_{QG} refers to the correlation of the query feature and the gallery features, S_{GQ} denotes the correlation of the gallery features and the query feature, S_{GG} denotes the autocorrelation of the gallery features. Then the decomposed solution is [29]:

$$F_G^* = (1-\alpha)(I - \alpha S_{GG})^{-1} S_{GQ} F_Q^0, \ \alpha \in (0,1). \tag{3.40}$$

And the clean form of Equation 3.40 is [29]:

$$F_G^* = L_\alpha^{-1} y, \tag{3.41}$$

where L_α refers to $I - \alpha S_{GG}$, y refers to $S_{GQ} F_Q^0$, and $1 - \alpha$ is ignored [29].

To speed up the method, we implement the offline construction step after decomposition. We implement an offline design [29] to compute each column $\{c_{i_1} ... c_{i_h}\}$ of L_α^{-1} and builds non-zero entries $\{v_1 ... v_h\}$ of y and save them. Then, the linear combination is obtained [29]:

$$F_G^* = \sum_j v_i c_{i_j}. \tag{3.42}$$

Truncation is the next step. In truncation, we downsize the original gallery to speed up the process. There is a truncation parameter k. We build the new gallery by extracting top-k elements from the original gallery.

After the truncation step, the online search step initializes the gallery enhancement by searching the query feature and implementing linear combination [29]:

$$F_G^* = L_\alpha^{-1} y. \tag{3.43}$$

Finally, we can obtain the most similar candidate features and accomplish the query and gallery enhancement.

The process of gallery enhancement is shown in Figure 3.13.

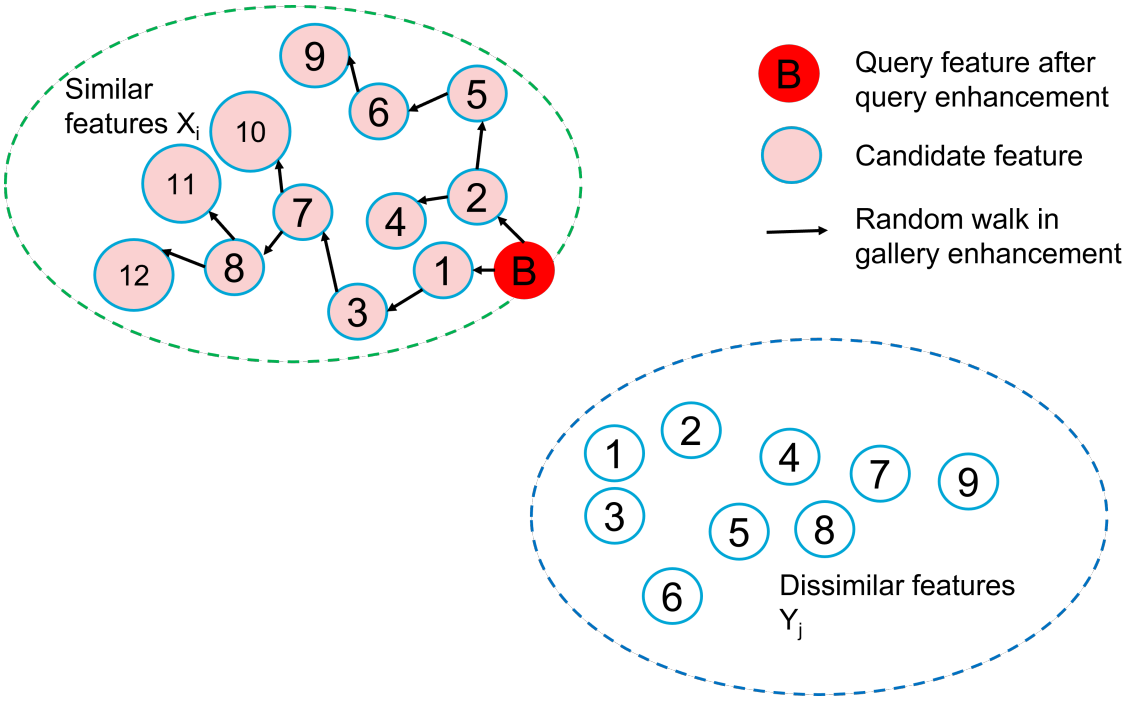

Figure 3.13: Gallery enhancement. Our method enables the new query feature B to find out many candidate features from similar features.

Instead of implementing the Euclidean distance to find candidate features, we implement the random walk in the gallery features. The random walk builds a graph with similar features from the query feature, and the closest candidate features X_i. Based on this graph, we can find most similar features as candidate features, and a few different features are considered candidate features. Therefore, QGE enables image search engines to be more accurate.

3.3.3 Contributions

Our contributions in QGE are as follows:

1. Query expansion uses arithmetic mean to build new query features, which does not consider the weight of candidate features in building a new query feature. To build an accurate query feature, we introduce a weight parameter w_i in candidate features. A large weight parameter implies a strong similarity between candidate and query features. Furthermore, we implement the L2 normalization to ensure the norm of feature vectors is consistent. Therefore, the weight enables the new query feature to be more informative than the original query feature.

2. We improved the iterative process. Query expansion implements iterations until we can obtain enough correct results. This process is slow because we have to enumerate many candidate features. Therefore we use a fixed number of iterations. This design can achieve a better balance of speed and accuracy. Furthermore, this design enables us to enhance the gallery.

3. Query expansion employs spatial verification to increase accuracy further. We studied this step and found that it can only increase the accuracy a little with high time costs. Therefore, we remove this step to improve the speed of the method.

4. Diffusion is a method to improve the accuracy of re-ranking with only a slight increase in computation time. It is based on the random walk and is a very efficient method. Researchers haven't combined query expansion and diffusion before. We find that these two methods can complement each other well: our improved query expansion can enhance the query feature and gallery features, while diffusion can enhance the search process. So we combine the two approaches and design QGE. QGE is more accurate than query expansion and diffusion and is very fast.

4 Experiments and Results

In this section, we test the three methodologies on different databases to evaluate their performance. We use the server platform of the CAS group.

4.1 Experiment Setup

4.1.1 Datasets

We use several databases in experiments, including Revisited Oxford5k [1,10], Revisited Paris6k [1,11], Oxford5k [10], Paris6k [11], Revisited Oxford (with 1 Million Distractors) [1,10], Revisited Paris (with 1 Million Distractors) [1,11] and Google Landmark Version 2 [12].

Oxford5k is a set of images (5,062) with 1024∗768 resolution comprising 11 different landmark buildings in the Oxford5k database. And there are 55 query images for the ground truth evaluation. For each landmark, the Oxford5k database provides five different query regions. Revisited Oxford5k (ROxford5k) is an improved database based on Oxford5k [10]. The example images from Oxford5k are shown in Figure 4.1.

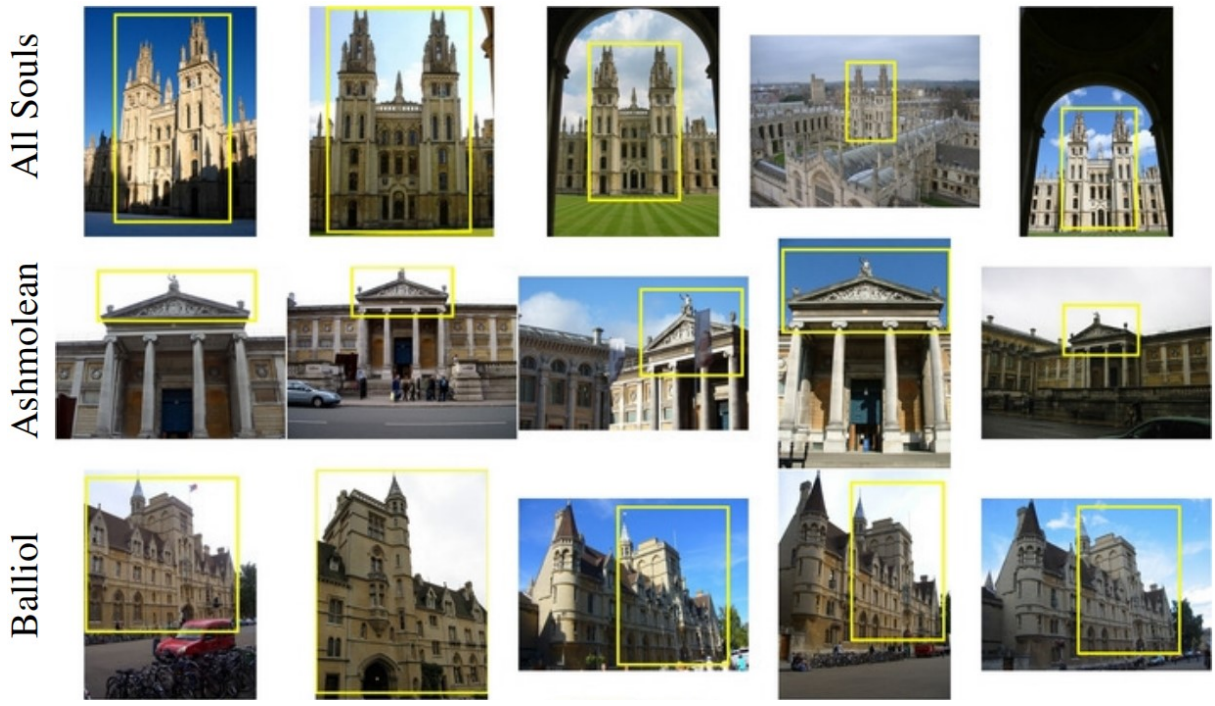

Figure 4.1: Example images of Oxford5k [10].

In Figure 4.1, the main objects are in the yellow boxes.

There are four kinds of images in the Oxford5k [10]:

1. Good (Easy): a nice, clear image of the main object and is easy to identify;
2. OK (Hard): more than 25% of the object is visible and is hard to identify;
3. Junk (Unclear): less than 25% of the object is visible, or there is a very high level of occlusion or distortion. Therefore it is unclear;
4. Absent (Unclear): the object is not present. Therefore it is unclear.

Paris6k is a set of images (6,300) with $1024 * 768$ resolution about famous buildings in Paris. And there are 55 query images for the ground truth evaluation. There are also four kinds of images (Good, OK, Junk and Absent) as Oxford5k. Revisited Paris6k (RParis6k) is an improved database based on Paris6k [11]. We show the example images from Paris6k in Figure 4.1.

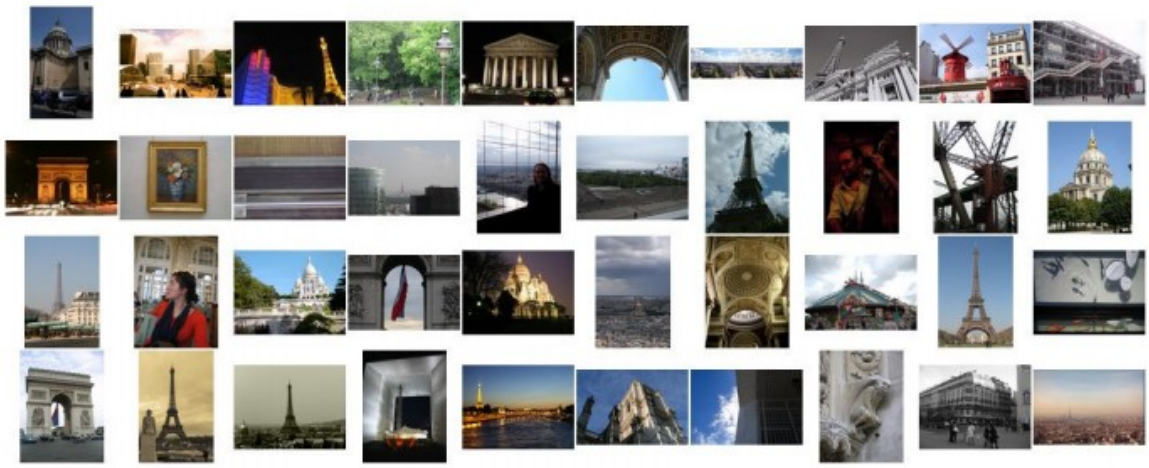

Figure 4.2: Example images of Paris6k [11].

ROxford5k and RParis6k fix annotation errors in Oxford5k and Paris6k and increases the number of query images from 55 to 70, respectively. Therefore, we choose ROxford5k and RParis6k as the primary databases and use Oxford5k and Paris6k as supplementary databases.

In ROxford5k and RParis6k, there are three kinds of difficulty settings:

1. Easy: Good images are treated as positive, while OK, Junk, and Absent are ignored;
2. Medium: Good and OK images are treated as positive, while Junk and Absent are ignored;
3. Hard: OK images are treated as positive, while Good, Junk, and Absent are ignored.

We test three methods under all these three difficulty criteria.

There are another 1 million (1M) distractors for ROxford5k and RParis6k, which increase the difficulty for accuracy and speed [1]. We usually use 1M distractors to test the performance of image search pipelines in extreme cases. Furthermore, 1M distractors require extremely high-level hardware. Therefore only a few existing methods have tested ROxford5k and RParis6k with 1M distractors [13, 15–17, 23]. We will also take the 1M distractors challenge.

The Google Landmarks Dataset Version 2 (GLD V2) is a larger dataset than ROxford5k and RParis6k, containing 5 million images of buildings or natural landmarks worldwide. And there are Training, Testing, and Validation sub-databases in GLD V2 for different purposes. In this work, we use the testing dataset of GLD V2 with $1,129$ query images and $761,757$ gallery images for testing the re-ranking method under extreme situations. Example images from GLD V2 is shown in Figure 4.3 and 4.4:

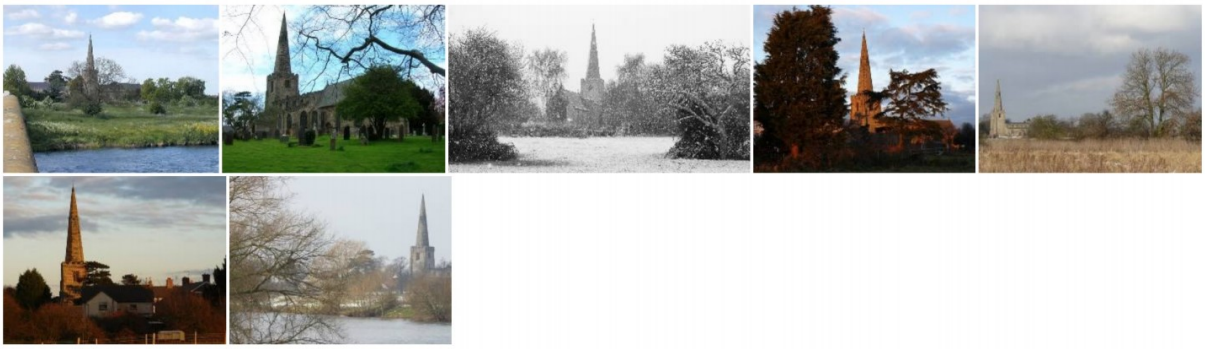

Figure 4.3: Example images of GLD V2 [12].

Figure 4.4: Example images of GLD V2 [12].

4.1.2 Evaluation Metrics

We evaluate the accuracy and speed of re-ranking methods in experiments. For accuracy, we use mean average precision (mAP) [1, 10, 11]. Mean average precision is based on precision and recall, which are shown in Equation 4.1 and 4.2 [82]:

$$p = \frac{\text{correct images} \cap \text{retrieved images}}{\text{retrieved images}}, \tag{4.1}$$

$$r = \frac{\text{correct images} \cap \text{retrieved images}}{\text{correct images}}, \tag{4.2}$$

where p refers to precision and r refers to recall. Therefore, we can obtain the Average Precision by Equation 4.3 [82]:

$$\text{AP@}n = \frac{1}{\text{GTP}} \sum_{k}^{n} \text{P@}k * \text{rel@}k, \tag{4.3}$$

where GTP refers to the total number of ground truth positives, n refers to the top n images we want, P@k refers to the precision@k and rel@k refers to a relevance function [82]. Finally, mAP can be calculated by Equation 4.4 [82, 83]:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i. \tag{4.4}$$

We use time (second) for testing the time cost for speed. The less the time means the faster the speed.

4.1.3 Benchmarks

We implement two famous traditional re-ranking methods as benchmarks: SIFT (a local feature-based re-ranking method) [19] and k-reciprocal (a global feature-based re-ranking method) [72].

1. SIFT is a local feature-based re-ranking method. Many image search engines use SIFT for re-ranking [13, 16, 17, 19, 23]. Therefore, we use SIFT as a local feature-based benchmark.

2. Query expansion is a popular global feature-based re-ranking method. Many image search engines implement query expansion for re-ranking [13–15, 17, 23]. Furthermore, k-reciprocal is a re-ranking method based on query expansion [72], and it is more accurate than the original query expansion method [72]. Therefore, we use k-reciprocal as a global feature-based benchmark.

3. In experiments, we use ResNet101 with GeM pooling as feature extraction. ResNet101 with GeM pooling is popular in image search as feature extraction because of its high accuracy [14–17,23]. First and foremost, we implement ResNet101 with GeM pooling to extract global features from images. Next, we use nearest

neighbor search to obtain preliminary results. Finally, we test our re-ranking methods. We show the pipeline in experiments in Figure 4.5.

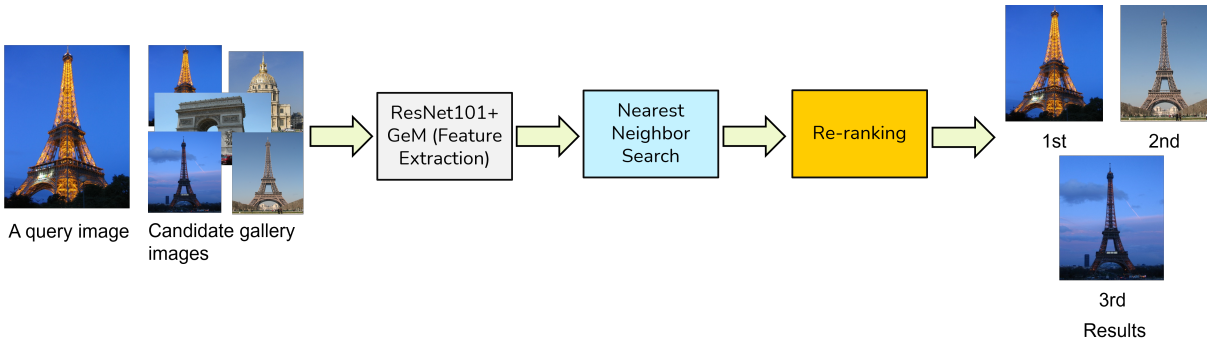

Figure 4.5: The pipeline in experiments. We implement ResNet101 with GeM pooling as feature extraction and use NN search to obtain preliminary results. After NN search, we test our re-ranking methods.

4.2 Experiments

4.2.1 Single-Pair Image Experiments with Local Features

At first, we compare the single-pair image matching performance of SAHA and LoFTR with the traditional local feature-based comparison method, SIFT [19], to demonstrate the accuracy of SAHA and LoFTR on a single pair of images. There are two images (A and B) with the same primary object, the Eiffel tower, as shown in Figure 4.6 and 4.7.

Figure 4.6: Image A (the Eiffel tower at daytime) [11].

Figure 4.7: Image B (the Eiffel tower at night) [11].

Then we implement the traditional SIFT-based algorithm to analyze the similarity between these two images, as shown in Figure 4.8.

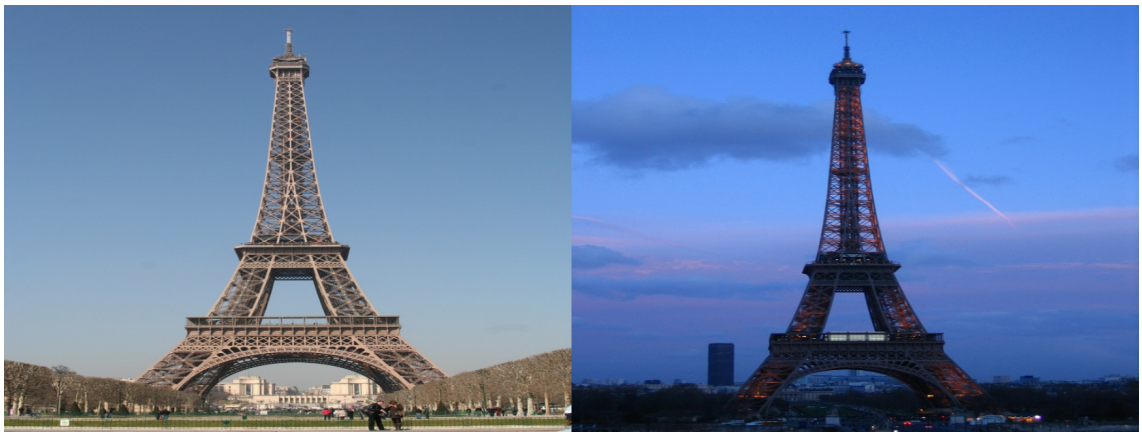

Figure 4.8: The result of the traditional SIFT-based method.

As shown in Figure 4.8, there is no match between the two images. Although the main objects in both images are the same, different background situations, such as illumination, decrease the sensitivity of the traditional SIFT-based algorithm and lead to poor performance.

Next, we implement SAHA and LoFTR to analyze the similarity between these two images, as shown in Figure 4.9 and 4.10.

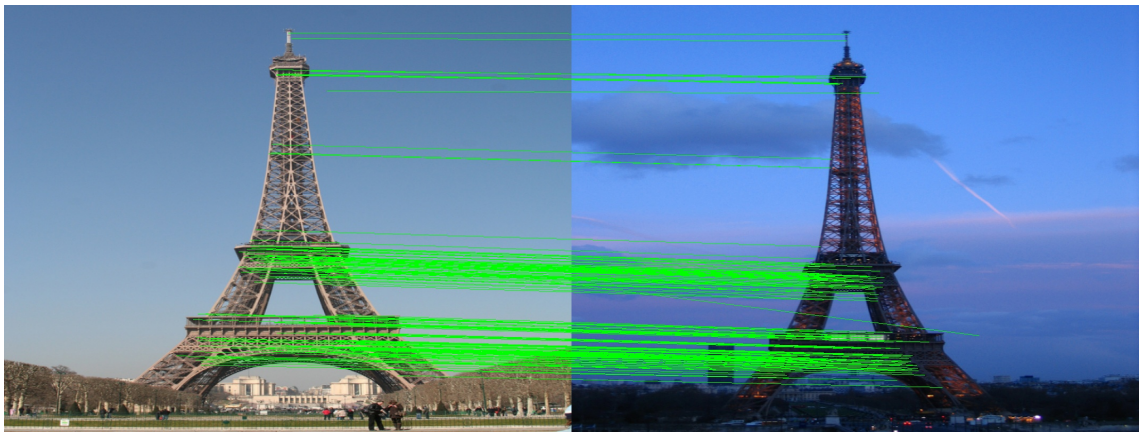

Figure 4.9: The result of SAHA.

The results of SAHA and LoFTR are much better than the result of the traditional SIFT-based algorithm. There are more accurate and dense matches from SAHA and LoFTR, which enables image search engines to consider these two images are similar as the more matches, the higher the similarity. Furthermore, SAHA and LoFTR show robustness to complicated situations in images, such as illumination. According to experiments, SAHA and LoFTR outperform the traditional local feature-based re-ranking method, SIFT.

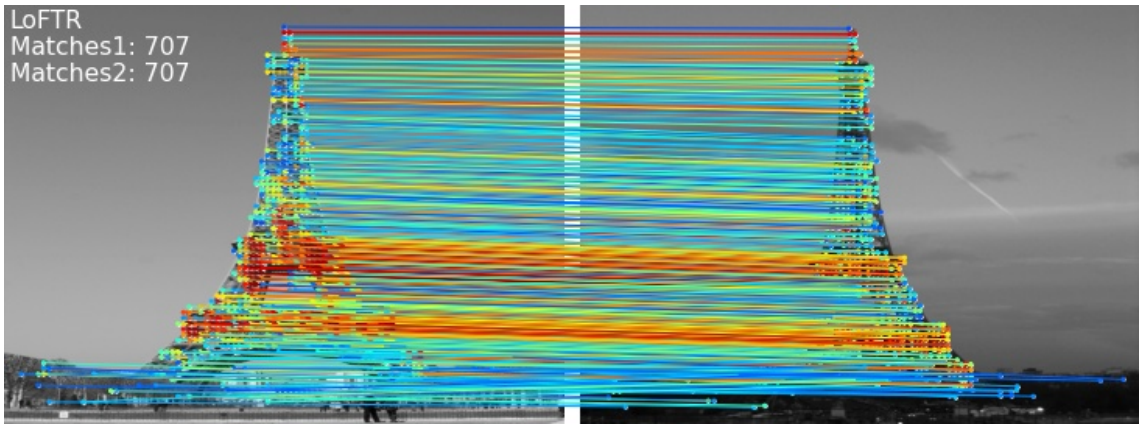

Figure 4.10: The result of LoFTR.

4.2.2 Experiments on ROxford5k and RParis6k

After the single-pair image experiments, we test three methods, SAHA, LoFTR, and QGE, on ROxford5k and RParis6k, the main databases in this report. And we implement a local feature-based re-ranking method, SIFT [19], and a global feature-based re-ranking method, k-reciprocal [72], as benchmarks. K-reciprocal is a famous re-ranking method by enhancing the query feature [72]. It is based on query expansion [26, 27]. In experiments, we use ResNet101 with GeM pooling [14] as feature extraction and L2 nearest neighbor search [22, 43]. There are two metrics: mAP and Time (second). We show the results in Table 4.1.

Methods	ROxford5k				RParis6k			
	mAP			Time (second)	mAP			Time (second)
	Easy	Medium	Hard		Easy	Medium	Hard	
ResNet101-GeM	82.24	63.46	37.18	0.1136	92.29	77.60	55.62	0.0337
ResNet101-GeM + SAHA	84.59	65.24	40.43	0.8986	93.04	77.76	55.80	0.9597
ResNet101-GeM + LoFTR	**84.76**	65.68	40.26	2.2845	93.59	77.79	55.82	4.6716
ResNet101-GeM + QGE	84.65	**69.07**	44.26	**0.1155**	**94.13**	**87.93**	**77.23**	**0.0355**
ResNet101-GeM + SIFT*	78.23	61.50	36.63	10.2977	92.26	77.73	55.76	5.5654
ResNet101-GeM + k-reciprocal*	83.14	68.45	**48.29**	1.2270	91.73	77.84	56.25	1.3406

Table 4.1: The results of SAHA, LoFTR and QGE on ROxford5k and RParis6k.

In Table 4.1, $*$ refers to benchmark methods, SIFT refers to the traditional local feature-based re-ranking method, SIFT re-ranking, and k-reciprocal refers to the widespread global feature-based re-ranking method, k-reciprocal re-ranking. The best results are in bold. Compared to the preliminary ranking, SAHA and LoFTR improve the accuracy with medium time costs. As local feature-based re-ranking, SAHA and LoFTR outperform SIFT in precision and speed. The ROxford5k and RParis6k require 4.3 GB memory in total. In this experiment, SAHA requires extra 20 GB memory while LoFTR does not require additional memory. Although SAHA and LoFTR

show good accuracy and medium speed, QGE is even better. Compared to SAHA, LoFTR, SIFT, and k-reciprocal, QGE shows excellent accuracy and speed in both databases. Especially in RParis6k, QGE exhibits a huge accuracy advantage, while its time cost is almost negligible. Therefore, QGE outperforms traditional local and global feature-based re-ranking methods in accuracy and speed. Also, SAHA and LoFTR are relatively slow on ROxford5k and RParis6k, which are medium databases. According to the speed requirement, the re-ranking method should finish the process in one second. Therefore, SAHA and LoFTR are not suitable for large databases. By contrast, QGE can meet the speed requirement with higher accuracy. Local features use many memory and storage space, making it impossible to test local feature-based re-ranking methods on these datasets. Moreover, QGE requires no extra memory because global features are much lighter than local features. Therefore we recommend QGE for large datasets. Thus, we continue to test QGE on other databases without SAHA and LoFTR.

4.2.3 Experiments on Other Databases

Furthermore, we test QGE on five other databases: Oxford5k, Paris6k, ROfxord5k+1 million distractors (ROxford5k+1M), RParis6k+1 million distractors (RParis6k+1M) and Google Landmark Version 2 (GLD V2) to analyze the extreme performance of QGE. Oxford5k [10] and Paris6k [11] are medium databases that were popular before ROxford5k and RParis6k [1]. ROxford5k+1M [1], RParis6k+1M [1] and GLD V2 [12] are large databases that are only used for extreme test in image search engines [13, 16, 17, 23]. As ROxford5k and RParis6k, there are three modes in ROxford5k+1M and RParis6k+1M: Easy, Medium, and Hard, with the same meaning. Also, we test the influence of the number of candidate features on QGE. In experiments on these databases, we use ResNet101 with GeM pooling [14] as feature extraction and approximate nearest neighbor search [84–86]. We show the results of medium databases in Table 4.2.

Methods	Oxford5k		Paris6k	
	mAP	Time (second)	mAP	Time (second)
ResNet101-GeM	88.74	0.0394	94.52	0.0389
ResNet101-GeM + QGE (top-3)	91.02	**0.0434**	**97.21**	**0.0442**
ResNet101-GeM + QGE (top-10)	90.76	0.0475	**97.21**	0.0477
ResNet101-GeM + QGE (top-30)	90.91	0.0438	97.18	0.0453
ResNet101-GeM + QGE (top-50)	**91.67**	0.0455	97.19	0.0474
ResNet101-GeM + QGE (top-100)	84.84	0.0458	97.17	0.0470

Table 4.2: The results of QGE on Oxford5k and Paris6k.

In Table 4.2, top-n refers to n candidate features used for QGE. The best results are in bold. Although the preliminary ranking achieves very high accuracy, QGE goes further and improves the accuracy with small-time costs. We should consider an important point: improving n can not always obtain higher accuracy because not all

top candidate features represent the correct images. As there are interfering factors in images, such as multi-object, illumination, viewpoint, and rotation, some top candidate features (false positive) represent inappropriate images. And the higher the n, the more wrong candidate features appear. QGE absorbs wrong information from incorrect candidate features to build the new query feature, which leads to decreased accuracy. Therefore, an appropriate n enables image search engines to achieve high accuracy in medium databases.

The results of large databases are shown in Table 4.3.

Methods	ROxford5k+1M				RParis6k+1M				GLD V2	
	mAP			Time (second)	mAP			Time (second)	mAP	Time (second)
	Easy	Medium	Hard		Easy	Medium	Hard			
ResNet101-GeM	57.64	40.07	19.11	0.0187	57.60	40.01	21.00	0.0109	12.88	0.0727
ResNet101-GeM + QGE (top-3)	**74.03**	54.83	**30.45**	0.2052	86.47	66.48	43.17	**0.1964**	**17.08**	0.1664
ResNet101-GeM + QGE (top-10)	73.02	**55.31**	30.25	0.2078	**87.38**	68.51	45.33	0.1974	16.48	**0.1644**
ResNet101-GeM + QGE (top-30)	65.65	47.75	23.13	**0.2038**	86.48	69.49	46.63	0.1965	14.43	0.1683
ResNet101-GeM + QGE (top-50)	64.22	45.90	20.41	0.2112	86.22	**69.88**	**47.44**	0.2004	12.88	0.1696
ResNet101-GeM + QGE (top-100)	62.72	43.30	16.66	0.2151	84.65	69.00	46.67	0.2007	10.62	0.1738

Table 4.3: The results of QGE on ROxford5k+1M, Paris6k+1M and GLD V2.

In Table 4.3, top-n refers to n candidate features used for QGE. The best results are in bold. Compared to preliminary ranking, QGE improves accuracy in large datasets with only a small increase in computation time. Especially, QGE shows outstanding performance in ROxford+1M and RParis+1M. As the results in Oxford5k and Paris6k, results in these large databases show that improving n can not always obtain higher accuracy in these databases for the same reason. However, the results are different in RParis6k+1M. In RParis6k+1M, the high n leads to high accuracy because there are more correct images in candidate features (images) in RParis6k+1M. Therefore, increasing n enables QGE to absorb correct information from candidate features and improve accuracy. Furthermore, the higher the n, the higher the time costs. Therefore, an appropriate n enables image search engines to achieve high accuracy in large databases.

In summary, QGE exhibits outstanding performance in both accuracy and speed and meets all requirements of this task.

4.3 Discussion

According to experiments, SAHA, LoFTR, and QGE show outstanding results, and the detailed analysis of the results is as follows:

1. Compared to the preliminary ranking, SAHA and LoFTR can provide higher accuracy on ROxford5 and RParis6k. Although SAHA and LoFTR are faster than the traditional local feature-based re-ranking method, they are much slower than global feature-based re-ranking methods. And SAHA requires extra 20 GB memory to implement the offline/online structure. The local feature is too computa-

tionally demanding to extract and match, making it unsuitable for large databases and unable to meet the speed requirement of this task.

2. Compared to the traditional local feature-based re-ranking method, SIFT, SAHA, and LoFTR show higher accuracy and require less processing time. The popular local feature-based re-ranking method, SIFT, can not provide high accuracy for our pipeline because our preliminary ranking is very accurate, which makes it hard for SIFT to improve the accuracy. According to the results, the offline/online design improves the speed of SAHA to make it much faster than SIFT.

   Furthermore, AffNet, HardNet, and AdaLAM improve the accuracy of SIFT. Moreover, LoFTR provides high accuracy, showing that the all-in-one structure can improve the local feature-based re-ranking. Therefore, SAHA and LoFTR improve the local feature-based re-ranking in speed and accuracy, respectively. Finally, SAHA and LoFTR could be accurate image matching methods.

3. According to Table 4.1, QGE shows the best performance in both accuracy and speed. Furthermore, QGE outshines the popular global feature-based re-ranking method, k-reciprocal, in accuracy because QGE enhances the query feature and gallery features while k-reciprocal only enhances the query feature.

4. To test the performance of QGE on other datasets, we implement QGE on Oxford5k, Paris6k, ROxford5k+1M, RParis6k+1M, and GLD V2. QGE shows excellent performance with only a small increase in computation time. Therefore, QGE is suitable for large datasets because it only needs little time to finish re-ranking and significantly improve accuracy.

Overall, we improve the traditional local and global feature-based re-ranking methods and use an efficient re-ranking method, QGE, which is suitable for large databases.

5 Conclusions and Future Work

5.1 Conclusions

After analyzing the results, we can conclude that the three methodologies (SAHA, LoFTR, and QGE) outshine off-the-shelf methods, and QGE satisfies all objectives of the thesis mentioned in Section 1.2. Furthermore, we summarize the conclusions of this work in the following statements:

1. **To improve the accuracy of local feature-based re-ranking methods:** We implement two ways: 1) using AffNet, HardNet, and AdaLAM to obtain more robust features and matching; 2) using an all-in-one module (LoFTR) to extract and match features. We provide SAHA and LoFTR based on these two ways. According to the experiment results, SAHA and LoFTR outperform the traditional local feature-based re-ranking method, SIFT, and improve the accuracy of preliminary ranking.

2. **To reduce the time cost of local feature-based re-ranking methods:** We implement an offline/online design for SAHA. The offline/online design is suitable for separable methods, like SAHA, and makes it faster than SIFT. However, SAHA and LoFTR can not provide fast enough speed because local features are too heavy to extract, calculate and match.

3. **To improve the accuracy of global feature-based re-ranking methods:** It is necessary to design a global feature-based re-ranking method because local feature-based re-ranking methods are not fast enough to meet the speed requirement. Global feature-based re-ranking is much faster than local feature-based re-ranking because global features are much lighter than local features. Therefore, we implement QGE. Traditional global feature-based re-ranking methods focus on improving query features. QGE improves query features and gallery features at the same time, which enables it to be more accurate than traditional global feature-based re-ranking methods.

4. **To decide the best solution:** According to experiments, QGE exhibits the best performance in both accuracy and speed. Furthermore, QGE is suitable for large databases. Therefore QGE is the best solution for this task.

5.2 Future Work

A few of the possible directions to explore as an extension to this work are:

1. The local feature-based re-ranking method SAHA shows excellent results in matching a pair of images. We could implement SAHA in the image matching task to test its performance.

2. We use non-learned methods to design QGE. However, neural networks have shown outstanding performance in computer vision [5, 7–9, 13, 16, 17, 23]. Therefore, a CNN-based query and gallery enhancement method could achieve different outcomes.

3. We enable global feature-based re-ranking to be more accurate in image search engines. We can implement it in other search tasks, such as text and video search engines.

4. In our image search engine, re-ranking is unique and separable. However, a unifying design could improve the coherence of the pipeline [13, 17, 23]. The re-ranking step can be combined with feature extraction and NN search to obtain more accurate or faster results.

Bibliography

[1] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, "Revisiting oxford and paris: Large-scale image retrieval benchmarking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 1573–1405, 2004.

[3] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[4] A. Amidi and S. Amidi, "Convolutional neural networks." https://stanford-edu.tudelft.idm.oclc.org/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

[5] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler, "D2-net: A trainable cnn for joint description and detection of local features," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[6] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[7] D. Mishkin, F. Radenovic, and J. Matas, "Repeatability is not enough: Learning affine regions via discriminability," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[8] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas, "Working hard to know your neighbor's margins: Local descriptor learning loss," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[9] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, "Loftr: Detector-free local feature matching with transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8922–8931, June 2021.

[10] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[11] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Lost in quantization: Improving particular object retrieval in large scale image databases," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.

[12] T. Weyand, A. Araujo, B. Cao, and J. Sim, "Google landmarks dataset v2 - a large-scale benchmark for instance-level recognition and retrieval," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[13] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," 2016.

[14] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1655–1668, 2019.

[15] J. Revaud, J. Almazan, R. S. Rezende, and C. R. d. Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[16] T. Ng, V. Balntas, Y. Tian, and K. Mikolajczyk, "Solar: Second-order loss and attention for image retrieval," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 253–270, Springer International Publishing, 2020.

[17] B. Cao, A. Araujo, and J. Sim, "Unifying deep local and global features for image search," 2020.

[18] H. Jun, B. Ko, Y. Kim, I. Kim, and J. Kim, "Combination of multiple global descriptors for image retrieval," 2019.

[19] R. Arandjelović and A. Zisserman, "Three things everyone should know to improve object retrieval," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2911–2918, 2012.

[20] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative evaluation of hand-crafted and learned local features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[21] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1224–1244, 2018.

[22] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[23] M. Yang, D. He, M. Fan, B. Shi, X. Xue, F. Li, E. Ding, and J. Huang, "Dolg: Single-stage image retrieval with deep orthogonal fusion of local and global features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11772–11781, October 2021.

[24] F. Tan, J. Yuan, and V. Ordonez, "Instance-level image retrieval using reranking transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12105–12115, October 2021.

[25] L. Cavalli, V. Larsson, M. R. Oswald, T. Sattler, and M. Pollefeys, "Handcrafted outlier detection revisited," in *European Conference on Computer Vision*, 2020.

[26] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman, "Total recall: Automatic query expansion with a generative feature model for object retrieval," in *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

[27] O. Chum, A. Mikulík, M. Perdoch, and J. Matas, "Total recall ii: Query expansion revisited," in *CVPR 2011*, pp. 889–896, 2011.

[28] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, and O. Chum, "Efficient diffusion on region manifolds: Recovering small objects with compact cnn representations," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[29] F. Yang, R. Hinami, Y. Matsui, S. Ly, and S. Satoh, "Efficient image retrieval via decoupling diffusion into online and offline processing," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9087–9094, Jul. 2019.

[30] M. Teichmann, A. Araujo, M. Zhu, and J. Sim, "Detect-to-retrieve: Efficient regional aggregation for image search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[31] C. Masone and B. Caputo, "A survey on deep visual place recognition," *IEEE Access*, vol. 9, pp. 19516–19547, 2021.

[32] U. Singh Parihar, A. Gujarathi, K. Mehta, S. Tourani, S. Garg, M. Milford, and K. M. Krishna, "Rord: Rotation-robust descriptors and orthographic views for local feature matching," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1593–1600, 2021.

[33] Y. Ke and R. Sukthankar, "Pca-sift: a more distinctive representation for local image descriptors," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 2, pp. II–II, 2004.

[34] A. Abdel-Hakim and A. Farag, "Csift: A sift descriptor with color invariant characteristics," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, pp. 1978–1983, 2006.

[35] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 404–417, Springer Berlin Heidelberg, 2006.

[36] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

[37] F. Dellinger, J. Delon, Y. Gousseau, J. Michel, and F. Tupin, "Sar-sift: A sift-like algorithm for sar images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 1, pp. 453–466, 2015.

[38] M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronin, and C. Schmid, "Local convolutional features with unsupervised training for image retrieval," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[39] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[40] C. Celik and H. S. Bilge, "Content based image retrieval with sparse representations and local feature descriptors : A comparative study," *Pattern Recognition*, vol. 68, pp. 1–13, 2017.

[41] L. Zheng, Y. Yang, and Q. Tian, "Sift meets cnn: A decade survey of instance retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1224–1244, 2018.

[42] Y. Lv, G. Jiang, M. Yu, H. Xu, F. Shao, and S. Liu, "Difference of gaussian statistical features based blind image quality assessment: A deep learning approach," in *2015 IEEE International Conference on Image Processing (ICIP)*, pp. 2344–2348, 2015.

[43] L. Cayton, "Fast nearest neighbor retrieval for bregman divergences," in *ICML 08: Proceedings of the 25th international conference on Machine learning*, pp. 112–119, 2008.

[44] K. Y. Shin Matsuo, "Cnn-based style vector for style image retrieval," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, June 2016.

[45] V. A. Sindagi and V. M. Patel, "A survey of recent advances in cnn-based single image crowd counting and density estimation," 2018.

[46] O. Seddati, S. Dupont, S. Mahmoudi, and M. Parian, "Towards good practices for image retrieval based on cnn features," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2017.

[47] N. Vo, L. Jiang, C. Sun, K. Murphy, L.-J. Li, L. Fei-Fei, and J. Hays, "Composing text and image for image retrieval - an empirical odyssey," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[48] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," 2020.

[49] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," 2021.

[50] D. L. T. H. Y. T. Ke Yan, Yaowei Wang, "Cnn vs. sift for image retrieval: Alternative or complementary?," in *Proceedings of the 24th ACM international conference on Multimedia*, October 2016.

[51] Wikipedia, "Convolutional neural network." https://en.wikipedia.org/wiki/Convolutional_neural_network.

[52] L. Ma, Z. Lu, L. Shang, and H. Li, "Multimodal convolutional neural networks for matching image and sentence," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[53] I. Melekhov, J. Kannala, and E. Rahtu, "Siamese network features for image matching," in *2016 23rd International Conference on Pattern Recognition (ICPR)*, pp. 378–383, 2016.

[54] C. Leng, H. Zhang, B. Li, G. Cai, Z. Pei, and L. He, "Local feature descriptor for image matching: A survey," *IEEE Access*, vol. 7, pp. 6424–6434, 2019.

[55] J. Ma, X. Jiang, A. Fan, J. Jiang, and J. Yan, "Image matching from handcrafted to deep features: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1573–1405, 2021.

[56] Z. Zhou, Q. M. J. Wu, S. Wan, W. Sun, and X. Sun, "Integrating sift and cnn feature matching for partial-duplicate image detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 593–604, 2020.

[57] F. Radenovic, G. Tolias, and O. Chum, "CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples," *CoRR*, vol. abs/1604.02426, 2016.

[58] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superpoint: Self-supervised interest point detection and description," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[59] J. Revaud, P. Weinzaepfel, C. De Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger, "R2d2: Repeatable and reliable detector and descriptor," 2019.

[60] Y. Ono, E. Trulls, P. Fua, and K. M. Yi, "Lf-net: Learning local features from images," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[61] A. Barroso-Laguna, E. Riba, D. Ponsa, and K. Mikolajczyk, "Key.net: Keypoint detection by handcrafted and learned cnn filters," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[62] U. Efe, K. G. Ince, and A. Alatan, "Dfm: A performance baseline for deep feature matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4284–4293, June 2021.

[63] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[64] Z. Luo, L. Zhou, X. Bai, H. Chen, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan, "Aslfeat: Learning local features of accurate shape and localization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[65] E. Bostanci, N. Kanwal, B. Bostanci, and M. S. Guzel, "A fuzzy brute force matching method for binary image features," 2017.

[66] A. Jakubović and J. Velagić, "Image feature matching and object detection using brute-force matchers," in *2018 International Symposium ELMAR*, pp. 83–86, 2018.

[67] OpenCV, "Feature matching." https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html#:~:text=Brute%2DForce%20matcher%20is%20simple,the%20BFMatcher%20object%20using%20cv.

[68] Wikipedia, "Norm (mathematics)." https://en.wikipedia.org/wiki/Norm_(mathematics).

[69] Wikipedia, "Brute-force search." https://en.wikipedia.org/wiki/Brute-force_search.

[70] W. Jiang, E. Trulls, J. Hosang, A. Tagliasacchi, and K. M. Yi, "Cotr: Correspondence transformer for matching across images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6207–6217, October 2021.

[71] S. Tang, J. Zhang, S. Zhu, and P. Tan, "Quadtree attention for vision transformers," *CoRR*, vol. abs/2201.02767, 2022.

[72] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[73] B. Ko, M. Shin, G. Gu, H. Jun, T. K. Lee, and Y. Kim, "A benchmark on tricks for large-scale image retrieval," *CoRR*, vol. abs/1907.11854, 2019.

[74] S. Pang, J. Ma, J. Zhu, J. Xue, and Q. Tian, "Improving object retrieval quality by integration of similarity propagation and query expansion," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 760–770, 2019.

[75] P. Dong and N. Galatsanos, "Affine transformation resistant watermarking based on image normalization," in *Proceedings. International Conference on Image Processing*, vol. 3, pp. 489–492 vol.3, 2002.

[76] Wikipedia, "Affine transformation." https://en.wikipedia.org/wiki/Affine_transformation.

[77] K. Mikolajczyk and C. Schmid, "Scale affine invariant interest point detectors," 2004.

[78] R. C. B. Martin A. Fischler, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," 1981.

[79] O. Astrachan, "Bubble sort: an archaeological algorithmic analysis," 2003.

[80] Wikipedia, "Bubble sort." https://en.wikipedia.org/wiki/Bubble_sort.

[81] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[82] R. J. Tan, "Breaking down mean average precision (map)." https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52.

[83] Wikipedia, "Evaluation measures (information retrieval)." https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Mean_average_precision.

[84] K. Hajebi, Y. Abbasi-Yadkori, H. Shahbazi, and H. Zhang, "Fast approximate nearest-neighbor search with k-nearest neighbor graph," 2011.

[85] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[86] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, "Approximate nearest neighbor search on high dimensional data — experiments, analyses, and improvement," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1475–1488, 2020.