Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Diploma Thesis

DETECTION OF OBJECTS OF INTEREST IN LARGE DYNAMIC SETS OF DATA

Bc. Aleš Franěk

Supervisor: prof. Ing. Jiří Matas, Ph.D.

Study Programme: Open Informatics
Specialisation: Artificial Intelligence

Prague, 2014

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Computer Science and Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Aleš Franěk**

Study programme: Open Informatics
Specialisation: Artificial Intelligence

Title of Diploma Thesis: **Detection of Objects of Interest in Large Dynamic Sets of Data**

Guidelines:

1. Review the state-of-the-art in fast detection of objects of interest from an unknown view in rapidly changing large collections of images.
2. Discuss motivation applications and formulate requirements on the detector with respect to missed detections, false alarms and the time of detection.
3. Propose a method of fast detection of objects of interest from an unknown view including the process of building a suitable representation of objects of interest.
4. Evaluate the performance of the method on real data.

Bibliography/Sources:

Forsyth, D.A. and Ponce, J., ``Computer Vision: A Modern Approach,'' 2nd edition 2011.
Tinne Tuytelaars, Krystian Mikolajczyk: Local Invariant Feature Detectors: A Survey. Foundations and Trends in Computer Graphics and Vision 3(3): 177-280 (2007).

Diploma Thesis Supervisor: prof.Ing. Jiří Matas, Ph.D.

Valid until the end of the summer semester of academic year 2014/2015

L.S.

doc. Ing. Filip Železný, Ph.D.
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, March 3, 2014

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací

V Praze dne ....

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my advisor professor Ing. Jiří Matas, Ph.D. for his support, enthusiasm, patience and tremendous guidance and feedback. I would also like to thank professor Yi-Leh Wu from the National Taiwan University Science of Technology for his engagement during my stay abroad. Michal Illich and the company Wikidi have also my appreciation for providing me with an outstanding working environment. Last but certainly not least, my special thanks goes to my family, friends and Zuzana Vaňková for their continuous support and encouragement during my whole studies.

Abstract [EN]

Social media have become a part of peoples' lives. People use it to express their feelings and their preferences. It is important for marketers to monitor sentiment around their brands. While the automatic textual analysis is widely used, some interpretation of shared images is still at its beginning. Social media are a specific domain because the amount of shared data huge and they require unique solutions.

We propose a new end-to-end method for image retrieval for logo recognition. We put emphasis on near perfect precision and short query time. In order to satisfy these requirements, we use state-of-the-art feature based methods. We use ORB detector, FREAK descriptor, Multi-probe LSH matching algorithm and RANSAC for verification. We also introduce our own improvements to the process like fast non-maxima suppression, mutual keypoint verification among the training images or tests which allow RANSAC to decline wrong hypotheses before computing their support.

We have implemented functional program which was evaluated on two datasets. FlickrLogos-32 is the first one and it is a standard dataset for logo recognition. It consist of 32 logotype classes downloaded from photo sharing service Flickr. The second one is our dataset of 5 million images downloaded from Twitter. One of our contributions is making this dataset and providing it for the future research.

Our algorithm works with 100% precision and 47% recall for the Flickr dataset and with 99% precision and 18% recall for the Twitter dataset.

Abstrakt [CS]

Sociální media se stala nedílnou součástí životů lidí. Lidé je používají, aby vyjádřili svoje pocitiy a názory. Pro merketéry je důležité, aby zde sledovali povědomí o své značce. Zatímco analýza přirozeného jazyka je zde hojně používaná, automatická interpretace sdílených obrázků se teprve ujímá. Sociální média jsou díky svému obrovskému objemu dat specifická oblast, která vyžaduje nová unikátní řešení.

Navrhujeme novou kompletní metodu pro vyhledávání obrazu a rozpoznávání log. Kladli jsme důraz na vysokou přesnost algoritmu a na jeho rychlost. Abychom toho dosáhli, použili jsme nejmodernější metody. Použili jsme ORB detektor, FREAK deskriptor, vyhledávací algoritmus Multi-probe LSH a RANSAC pro finální verifikaci. Mimo to jsme také představili vlastní vylepšení vyhledávacího procesu – rychlou metodu pro lepší distribuci bodů zájmu v obrázku, vzájemná verifikace a filtrace bodů zájmu mezi trénovacími daty nebo série testů, které odhallí některé špatné hypotézy prostorové transformace aniž by se musela počítat jejich plná podpora.

Algoritmus jsme úspěšně naimplementovali a vyhodnotili ho na dvou datasetech. První se nazývá FlickrLogos-32 a je to standardní dataset pro rozpoznávání log. Skládá se z 32 tříd log stažených ze služby Flickr. Druhý dataset obsahuje 5 milionů obrázů ze sociální sítě Twitter. Jeden z našich přínosů je právě vytvoření tohoto datasetu a poskytnutí ho k dalšímu výzkumu.

Náš algoritmus dokázal na datech z Flickru nalézt 47% skutečných log bez falešné detekce. Pro data z Twitteru našel 18% vyskytujících se log s 1% nesprávně označených obrázků.

Table of Contents

1 Introduction 1
2 State of the Art 5
2.1 Feature Detectors 6
2.1.1 Harris descriptor 6
2.1.2 Difference of Gaussians 9
2.1.3 SUSAN 9
2.1.4 FAST 9
2.1.5 ORB 10
2.2 Feature Descriptors 11
2.2.1 SIFT 11
2.2.2 BRISK 12
2.2.3 FREAK 14
2.3 Feature based image retrieval 16
2.3.1 Bag of Words 16
2.3.2 Multi-Probe Locality Sensitive Hashing 18
2.3.3 Bundle Min-Hashing 20
2.4 Geometrical Verification 23
2.4.1 Weak Geometrical Constraints 23
2.4.2 Hough Transform 23
2.4.3 RANSAC 24
3 Method 27
3.1 Requirements 27
3.2 Image Description 28
3.2.1 Feature Detection 28
3.2.2 Spatial Non-Maxima Suppression 28
3.2.3 Feature Description 30
3.3 Preprocessing of Training Images 32
3.4 Feature matching 36
3.4.1 Geometrical verification 38
3.5 Implementation 40

4 Datasets 41
4.1 Flickr Dataset 41
4.2 Twitter Dataset 42
4.3 Oxford Dataset 43
5 Experiments 44
6 Conclusion 51
7 References 53
A. Examples of results 59
B. Content of the attached DVD 61

1 Introduction

Social media are becoming more and more significant. In 2014, two thirds of the global internet population are actively using some kind of social network [1]. Facebook[1] alone, the world's biggest social network, has 1.28 billion monthly active users [2].

The amount of data transferred and stored by the social media is growing rapidly. While Facebook stores 250 billion photos and another 350 million photos are uploaded every day [3], users of the photo messaging application Snapchat[2] send up to 700 million photos per day [4]. Instagram[3] is a smaller, yet still widely popular photo-sharing social network recently acquired by Facebook. Its users post 70 million photos daily [5]. All these images are also often linked to an extensive amount of metadata about users' personal information, networking and settings.

Access to all this data creates a huge opportunity for the marketers. It has been estimated that social media advertising revenue will reach $15 billion by the year 2018 [6]. The marketers are able to target their campaigns to the right audience as well as to effectively connect with their customers. It allows them to run thousands of versions of their campaigns, each tailored for a very specific demographic group, and to adjust them as they get immediate feedback. That is why the comprehensive analysis of the data from social media is so crucial.

While analysis of users' networking and behaviour and of the textual part of the users' posts are exploited by many social media management tools, an automatic interpretation of images is still not widely integrated, which is the motivation for development of methods proposed in this thesis. A highly desired service in this context is certainly to detect products and logos of the companies so they would be able to measure the effects of their campaigns as well as to track the sentiment around their brands.

There has been a recent increase in startup companies which aim to detect and monitor brands' logos in images on social networks. Gaze Metrix [7], Phashtag [8], LTU Technologies [9] or $3.6 million funded [10] Ditto Labs [11]to name a few. Unfortunately, none of these companies gives out any technical information about its solution or performance statistics.

We are interested in developing a method which would be able to monitor objects of interest in images in order to measure users' engagement on social networks. We have chosen Twitter[4] as our social network of choice in order to evaluate the performance of our method since images from Facebook and Snapchat are hard or even impossible to

[1] http://www.facebook.com

[2] http://www.snapchat.com

[3] http://www.instagram.com/

[4] http://www.twitter.com

obtain due to the privacy policies of these services. The exact number of images shared on Twitter is not public. Twitter only states that 500 million tweets are sent daily [12] which enables us to make a rough estimate of tens of millions of shared images per day. However, not all the images must be processed. At least one quarter of the posts on Twitter are duplicates [13], so called retweets, and this ratio is even much higher for posts with photos [14]. Also, some users might not have a sufficient audience or they might not come from monitored geographical regions.

This determines the requirements on the system. While there are no time constraints on the offline stage, the recognition itself must be done on the fly since the data stream is constant. Even with described pre-filtering of irrelevant images taken into account, the system must be able to process millions images per day which equals to dozens of images per second. The time of the recognition also must not be worse than linearly dependent on the number of detected classes.

Due to the enormous number of queries and the estimate that the *a priori* probability that an image contains the desired class is between 10^{-5} and 10^{-4}, another requirement is that the false positive rate (number of false positives divided by number of all negatives) should be close to zero, otherwise the false positives would heavily outnumber the true positives among the results. Since the actual *a priori* probability is unknown and thus it cannot be simulated in the experiments, the requirement for this work is to have zero false positive rate, i.e. to have the perfect precision. This setting certainly decreases the recall however perfect recall is not essential for this application since we are rather interested in overall changes in the statistics.

This thesis proposes an end-to-end solution for the object detection based on the state-of-the-art methods. To meet the requirements mentioned above, it uses FAST corner detector [15, 16] with rotation estimation proposed in [17], binary FREAK descriptor [18], multi-probe locality sensitive hashing [19] for matching the features and RANSAC for the geometrical verification. It also introduces a cascade of various tests and filters which reject most of the unpromising images before they advance to further parts of the algorithm.

The experiments will be done on two datasets. The first dataset contains 2240 labelled images of 32 classes and 6000 general images, all downloaded from the photo sharing service Flickr[5], provided by the Multimedia Computing and Computer Vision Lab, Augsburg University[6]. The second one consists of 5 million images which were attached to Twitter posts which contained some of the selected keywords.

The rest of the thesis is structured as follows. State-of-the-art methods for fast object recognition and image retrieval are reviewed in Chapter 2. In Chapter 3, suitability and performance of these methods is discussed with respect to the requirements. A new end-to-end algorithm for detection of objects of interest in large dynamic sets of data is proposed. Used datasets are described in Chapter 4. Performance of the algorithm is

[5] http://www.flickr.com

[6] http://www.multimedia-computing.de/flickrlogos/

evaluated by a series of experiments on real data in Chapter 5. Suitability of the proposed algorithm for real life applications is summarized in Chapter 6 as well as further improvements are suggested.

2 State of the Art

In this chapter, we review currently used methods for image retrieval and their suitability for the application of object detection in the social media domain will be discussed.

There are several approaches how to solve an image retrieval task. Detection and description of local features is almost always the substance. Feature based methods are widely used in many computer vision applications. It would be generally a hard task to look for correspondences in images by trying to naively align them and then compare the differences in pixel intensities. The viewpoint may change, there might be different illumination conditions, different levels of noise in the image and/or occlusions may appear. The best practise is to find local highly descriptive regions (features), describe them in some manner which is invariant to slight changes caused by geometrical transformation, illumination changes, noise, etc., and look for correspondences in some space of these features.

Even though there are certainly computer vision applications where the colour information is essential, most of the algorithms, where the feature matching is involved, use the grayscale images even if the 3-channel colour images are available. The extra information about the colour does not usually compensate for the substantial slowdown of processing three channels instead of one. The grayscale pixel intensities are obtained by simply averaging out the colour channels.

Comparing or efficient storing of local features directly would not be possible for larger databases, therefore some kind of abstraction must be used.

Jégou et al. [20] developed a way to aggregate local descriptors into a compact image representation called VLAD. A single fixed-size vector represents the whole image. It is obtained by assigning d-bit-long descriptors of local features to previously computed k centroids. Summing the descriptors per each centroid results in k vectors which together create $k * d$-bits-long image decsriptor. Search and indexing are reduced to distance approximation problem. This method works well even for databases of tens or hundereds of million of images, however it is rather designed to search for simillar scenes. The objects of interes we search for can be found in very different scenes with different scales and orientations. Therefore VLAD and other approaches which generalize an image to a single descriptor cannot be used.

Bag of words is another widely used algorithm for the image retrieval. It quantizes feature descriptors of an image to so called visual words. The image is then represented by a sparse vector of occurances of these visual words. The algorithm is decribed in more detail in Chapter 2.3.1, however it was outperformed by bundle min-hashing for the purposes of logo detection.

To the best of our knowledge, Romberg and Lienhart [21] has achieved the best results in logo recognition so far. They use bundle min-hashing to obtain a shortlist of candidate matching images from the database. Bundle min-hashing is further described in Chapter 2.3.3.

Based on the state-of-the-art method of bundle min-hashing, we have decided to use feature based methods and image indexing to retrieve a shortlist of possible candidate solutions from the database. We have also chosen to use spatial verification to re-rank the shortlist. Unlike the bundle min-hashing, we focus on very fast keypoint detection and description as well as the overall retrieval time.

2.1 Feature Detectors

Representing an image by a set of local features allows algorithms to be robust to occlusions. Local futures are detected as regions of pixels in the image rather than only single pixels/points which would not be very descriptive. Local features, interest points, keypoints, distinguished regions or patches are different terms which appear in the literature but usually denote the same thing.

Förstner [22] formulated basic requirements which a good feature detector should fulfil:

- *Distinctness* – The points should be distinct from their neighbourhood.
- *Invariance* – The selection and the positions of the points should be invariant to geometric transformations and illumination changes
- *Stability* – The selection should be robust to noise
- *Seldomness* – Elements of repetitive patterns should not be selected or at least get lower weight
- *Interpretability* – The selection principle should be interpretable in some sense, e.g. looking for edges, corners or blobs

Commonly used modern detectors meet these requirement up to some level. There might be a trade-off between robustness and speed of the algorithm but there are usually even more aspects which must be taken into account with respect to the application. Some detectors can provide sub-pixel precision or estimate rotation or affine transformation. Some detectors are more suitable for stereo vision, some perform better at object tracking. And some are even three orders of magnitude faster than others.

There is simply not one single detector which would outperform the other ones and defining requirements for the descriptor must be a part of the design of a recognition system.

2.1.1 Harris descriptor

Moravec [15] proposed a measure for local change E in direction (u, v)

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2, \tag{2.1}$$

where $w(x,y)$ is a window function and $I(x,y)$ is the intensity of the pixel in the y-th row and the x-th column of the image I. Then, a point is considered as a corner if the change E is large in all directions.

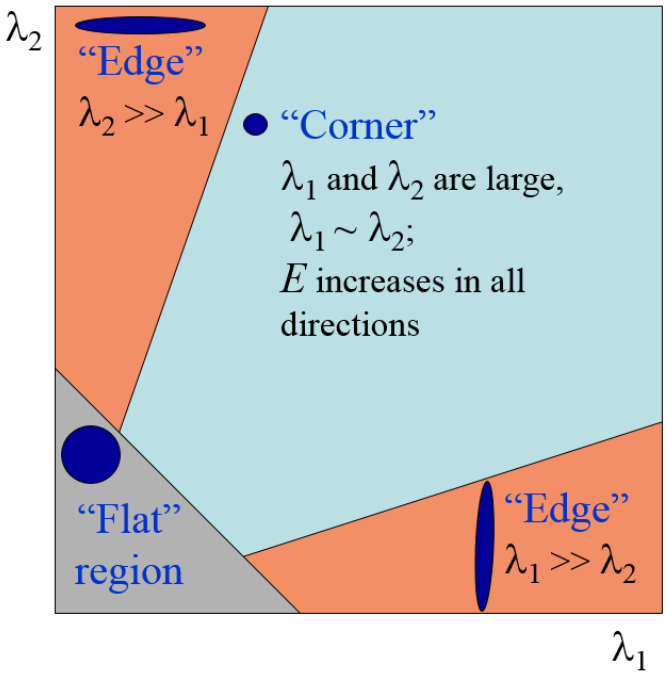


Figure 2.1 The space of eigenvalues of the matrix M and interpretation of their values. [16]

Harris and Stephens [23] used 2nd order Taylor expansion to approximate this equation and reformulated the change measure to

$$E(u,v) = [u,v]M\begin{bmatrix}u\\v\end{bmatrix} \tag{2.2}$$

where M is the structure tensor

$$M = \sum_{x,y} w(x,y)\begin{bmatrix}I_x^2 & I_xI_y\\ I_xI_y & I_y^2\end{bmatrix} \tag{2.3}$$

where I_x is the image derivative in the x direction.

They then observed that they can use the eigenvalues λ_1, λ_2 of the matrix M to do the analysis of the intensity change in the shifting window. The eigenvalues define the directions and magnitude of the fastest and the slowest changes.

If the eigenvalues are both small, it means E is almost constant in all directions which corresponds to a flat region. If only one of the eigenvalues is large, it corresponds to an edge. And if the eigenvalues are both large, E increases in all directions which corresponds to a corner (see Figure 2.1).

The exact computation of the eigenvalues would be computationally too expensive. However, they realized that

$$\det(M) = \lambda_1 \lambda_2 \tag{2.4}$$

and

$$trace(M) = \lambda_1 + \lambda_2 \tag{2.5}$$

so they introduced a measure of corner response

$$R = \det(M) - \kappa(trace(M))^2 \tag{2.6}$$

where κ is an empirical constant 0.04. The large values of R correspond to the large eigenvalues of M. Corners are the local maxima of this function.

Harris detector is invariant to rotation and intensity shift since only the derivatives are used, however it is not invariant to the changes in intensity scale and image scale. The scale invariance was solved by Mikolajczyk and Schmid [24] who developed the Harris-Laplace detector as well as the Harris-Affine detector, an affine invariant detector.

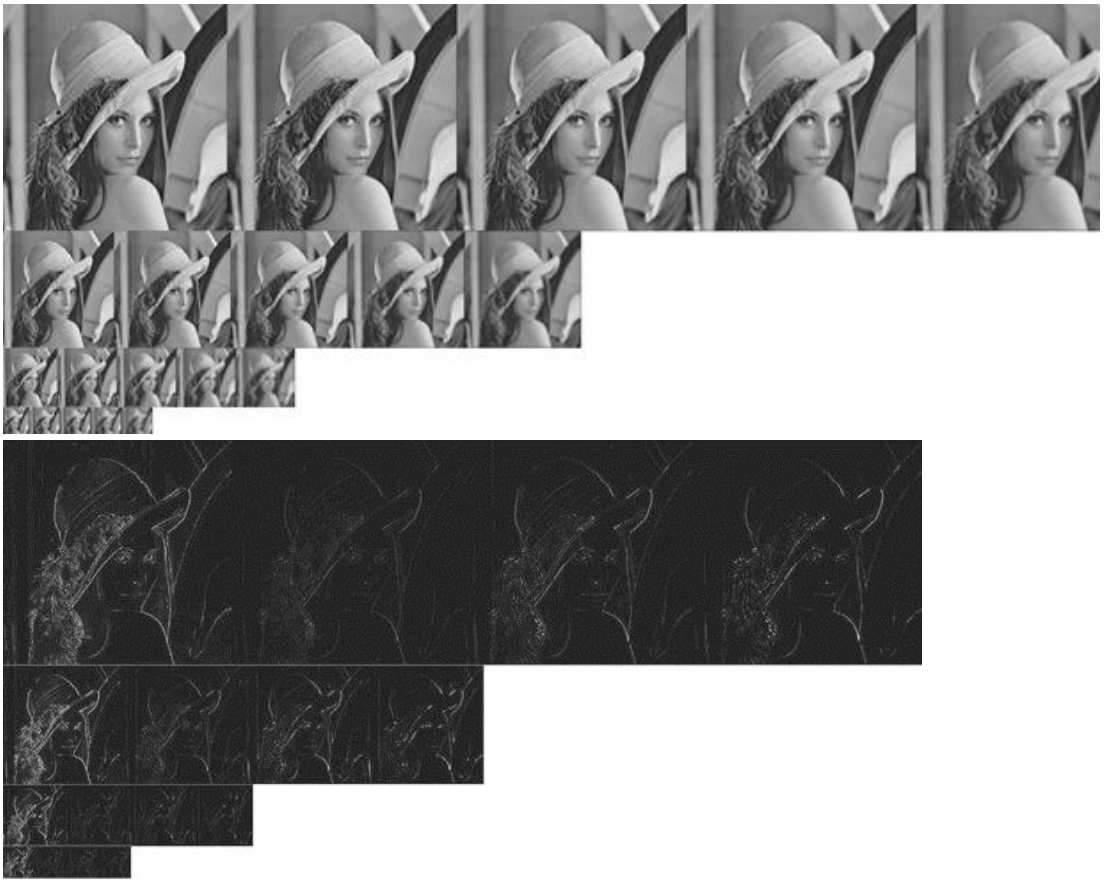


Figure 2.2 Images of the scale space pyramid for DoG. Images blurred with different Gaussian kernels (up) and their differences (down). [52]

2.1.2 Difference of Gaussians

The Difference of Gaussians (DoG) [25] is another widely used way to find local features. It approximates the Laplacian with the difference of images smoothed by different Gaussian convolution masks.

The image is smoothed by the Gaussian kernels of different sizes. This is done for every octave of the multi-scale pyramid. The blob and edge responses are obtained by subtracting the blurred images within the same octave (see Figure 2.2). Then, eigenvalues of the Hessian matrices of the local maxima are evaluated and only the strongest ones over space and scales are kept. These points correspond to the corners in the image.

2.1.3 SUSAN

Smith and Brady [26] introduced a corner detector which uses a morphological approach rather than computing the local gradients. SUSAN stands for Smallest Univalue Segment Assimilating Nucleus. It looks at the circular neighbourhood of each pixel and compares its intensity values to the centre pixel, the nucleus. If the difference between the intensity value of a neighbouring pixel and the intensity value of the nucleus is smaller than a given threshold, it belongs to so called USAN. The idea is that USAN will fill up most of the neighbourhood in homogenous and non-descriptive areas. And vice versa, neighbourhoods of corners should not contain a lot of pixels which belong to USAN. The local features are then the local minima of the USAN measure.

Larger weights were assigned to the pixels closer to the nucleus to make the algorithm more robust. Also, a series of rules is introduced to eliminate the less promising points before computing the whole USAN.

2.1.4 FAST

Rosten and Drummond [15, 16] improved the SUSAN detector by not comparing the centre pixel with all the pixels in the neighbourhood but only with pixels on a fixed circle around the centre. Bresenham's circle[7] of radius 3 with 16 pixels on the perimeter was used. A region can be a corner feature only if there is at least n contiguous pixels which are all either brighter or all darker by the threshold t than the centre pixel.

This procedure does not compute any information about the corner response of the region and FAST (Features from accelerated segment test) actually returns much more keypoints than other descriptors. To be able to perform the non-maxima suppression, authors introduce a fast score function V given by

$$V = \max\left(\sum_{x \in S_{bright}} |I(p \to x) - I(p)| - t\,,\ \sum_{x \in S_{dark}} |I(p \to x) - I(p)| - t\right) \qquad (2.7)$$

[7] Bresenham's circle algorithm is a method in computer graphics of how to select pixels in order to draw a circle in a pixel raster

with

$$S_{bright} = \{x | I(p \to x) \geq I(p) + t\} \tag{2.8}$$

$$S_{dark} = \{x | I(p \to x) \leq I(p) - t\} \tag{2.9}$$

where $x \in \{1..16\}$ corresponds to the location on the circle around the centre pixel p (see Figure 2.3), so $p \to x$ denotes a pixel at that relative position to p and $I(p \to x)$ is the intensity value of that pixel.

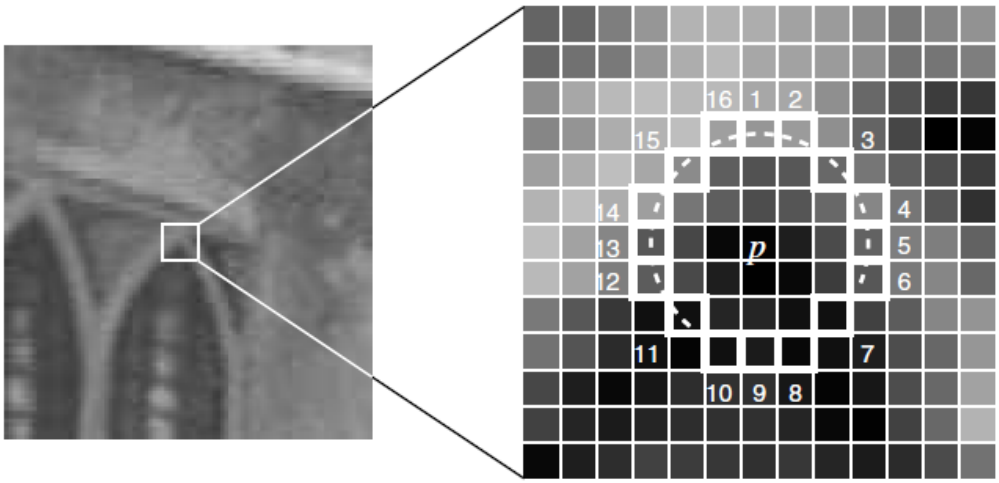


Figure 2.3 Corner point detected by FAST. Only the 16 highlighted pixels are involved in the detection. The dashed line indicates an arc which passes through 12 contiguous pixels which are all brighter than the centre point p by more than threshold t. [10]

There is no need for evaluating all the pixels on the perimeter. In fact, some regions can be already rejected after first two comparisons. A big decision tree was trained by the ID3 algorithm [27] to determine in which order the pixels should be checked. This results in an optimization where only 3.8 pixels need to be checked on average. The FAST detector is up to 30 times faster than DoG according to [28].

FAST is certainly one of the fastest known detectors however it is not scale invariant which makes it inapplicable for some more complex applications.

2.1.5 ORB

ORB (Oriented FAST and Rotated BRIEF) [17] is a detector and descriptor algorithm which improved and combined two know solutions, FAST and BRIEF [29].

FAST is not scale invariant so a multi-scale Gaussian pyramid was used to achieve this property. ORB computes more time consuming Harris score instead of the fast score function used by FAST. However, since the score is calculated only for the preselected corner points, it is still orders of magnitude faster than the Harris detector.

FAST also does not compute the orientation of features since it was originally developed for tracking in videos where this information was not necessary. ORB uses a

simple way to determine the orientation proposed in [30]. It defines the moments of a patch as

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{2.10}$$

and the intensity centroid

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right). \tag{2.11}$$

Orientation of a patch is then the direction of vector $\overrightarrow{PC}$ going from centre of the patch P to the intensity centroid C. It can be computed as

$$\theta = \text{atan2}(m_{01}, m_{10}) \tag{2.12}$$

where *atan*2 is a quadrant-aware version of arctan. The original paper [30] distinguishes orientation for patches with bright or dark background, but the authors of ORB use the same measure for all patches since it is consistent regardless of the corner type.

The orientation measure may become unstable when $\left|\overrightarrow{PC}\right|$ is too small, however the authors claim it rarely happens for FAST corners. They even showed that using the intensity centroid performed much better for noisy images than using the histogram of gradient directions, another popular way to determine patch orientation, used for example by SIFT [31, 32].

2.2 Feature Descriptors

Comparing pixel intensities within the patches directly would be time inefficient and would not be robust to noise and spatial and illumination transformations. The best practise is to obtain invariant descriptors, vectors of either real numbers or bits, and compare the keypoints in a space of these descriptors.

Descriptors use various image properties such as histograms of gradients, intensity differences, analysis of the Fourier power spectrum and so on.

2.2.1 SIFT

David Lowe [31, 32] presented a complete algorithm for detection, description, indexing, matching and verification of the local features. His solution became widely popular and his articles are one of the most cited works in the computer vision community.

Several new ideas were introduced. The features were detected as the local extrema of the differences of Gaussians in the scale-space pyramid. Sub-pixel and sub-scale precision was achieved by interpolating the neighbourhood of a pixel over all three dimensions in the scale space. The orientations of the keypoints were determined by the modes of gradient orientation within the local region.

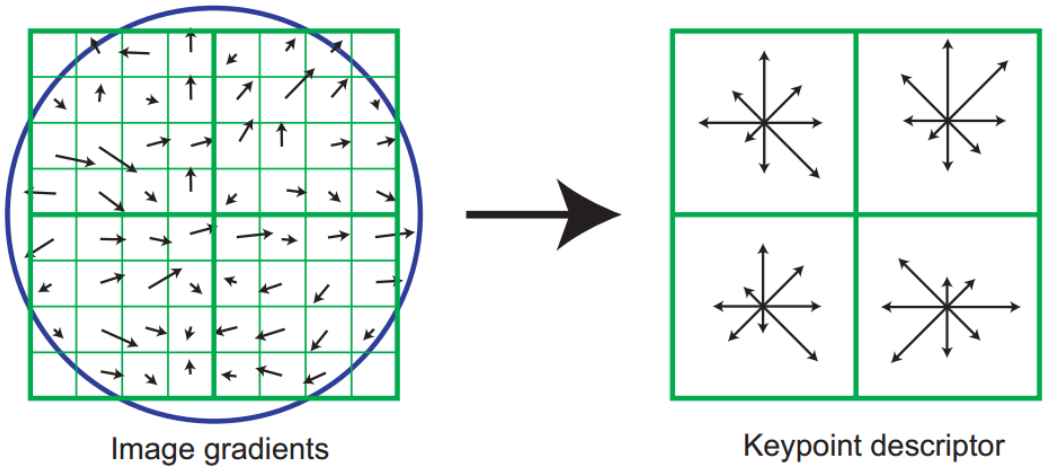


Figure 2.4 SIFT descriptor. Gradient magnitudes and orientations are first computed for each point in the patch (left) and then weighted by a Gaussian window, indicated by the blue circle. These values are then accumulated into orientation histograms for each 4x4 subregion (right). The arrows indicate the sums of the gradient magnitudes of particular orientations. This figure is only a simplified scheme because it shows 8x8 lattice, whereas SIFT descriptor is constructed from 16x16 lattice. [25]

Then, the region was divided into 4x4 lattice and the gradients were quantized with into 8 histogram bins within each cell. 128-element-long SIFT descriptor was constructed from these histograms.

Lowe also proposed indexing of the features by a modified k-d tree using the best-bin-first search method. To match features between two images, the closest and the second closest pairs were retrieved for each feature. If the ratio between the Euclidean distances of these pairs was larger than a certain threshold, the closest pair was considered as a good match. Further spatial verification of tentative matches was done by the Hough transform.

SIFT descriptor was later improved by SURF [33] which is one order of magnitude faster. Both SIFT and SURF are non-commercial and a licence must be obtained for any commercial applications.

2.2.2 BRISK

BRISK (Binary Robust Invariant Scalable Keypoints) [34] represents one of the binary descriptors. To obtain the keypoints, it uses FAST detector across the scale space. Afterwards it performs non-maxima suppression according to the score s which is defined as a maximum threshold t for which an image point remains a FAST corner. Furthermore, it refines positions and scales of keypoints by interpolating positions of corresponding

keypoints across the scale octaves. It achieves sub-pixel and sub-scale precision by this refinement.

BRISK uses pairs of pixels from a sampling pattern for the description. There are $N = 60$ points evenly distributed on concentric circles around the centre pixel (see Figure 2.5). Pixel intensities are smoothed with Gaussian blur proportionally to the separation of points on each concentric circle, so the outer points are smoothed with the largest kernel. Notice there is no overlap between the regions thus there are no aliasing effects.

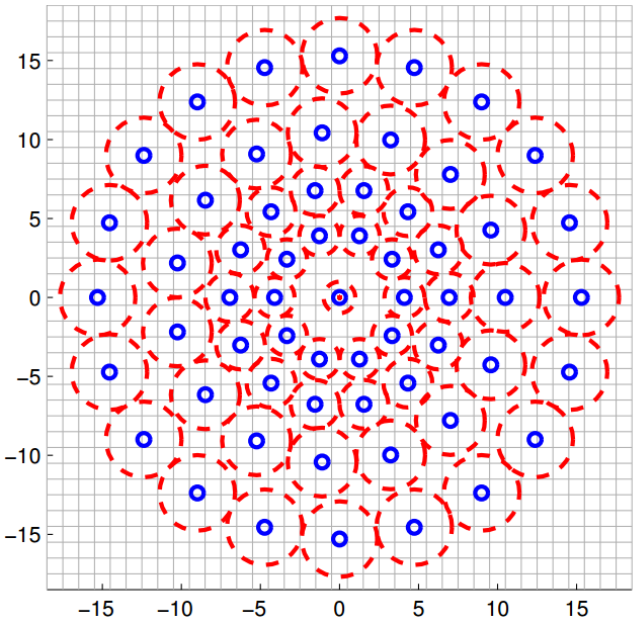


Figure 2.5 BRISK sampling pattern. Blue circles depict the sampling locations while red dashed circles correspond to the standard deviations of Gaussian kernels. [27]

The size of a set of all possible pairs of such points is $N(N-1)/2 = 1770$. The authors divide this set into three subsets according to the distances of these pairs. Set $\mathcal{S}$ of the short-distance pairings, set $\mathcal{L}$ of the long-distance pairings and the rest belongs to the set of unused pairings which are neither long nor short.

Keypoint orientation is computed as a weighted sum of gradients of all pairs from the set $\mathcal{L}$. The keypoint is rotated accordingly and the descriptor is constructed by comparing the intensities of paired points from the set $\mathcal{S}$. $\mathcal{S}$ contains 512 pairs and each pair corresponds to one bit which results in a 512-bit-long feature vector.

The advantage of bit-string feature vectors is that the distances between them can be computed really fast. The Hamming distance is obtained using bitwise XOR followed by adding up 512 bits afterwards. This procedure may be even speeded up thanks to the modern architectures of processors and their instruction sets.

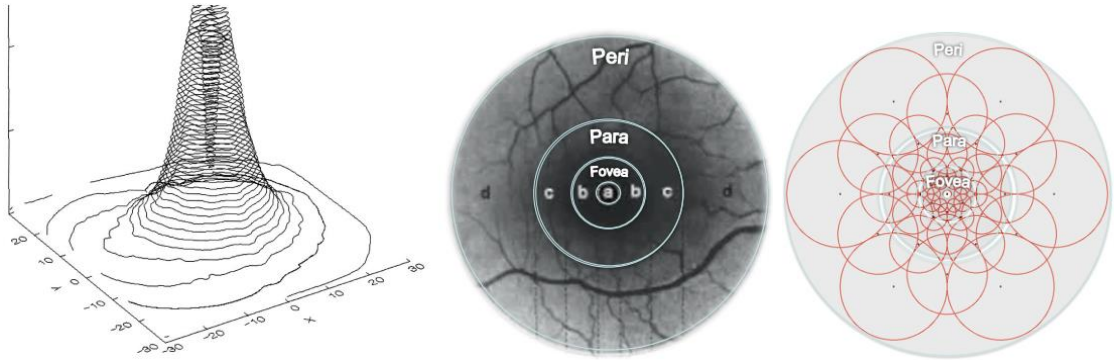


Figure 2.6 Human retina and FREAK sampling pattern. Density of ganglion cells over the retina (left) clustered into four areas: (a) foveola, (b) fovea, (c) parafoveal and (d) parifoveal (middle) and sampling pattern used by FREAK mimicking these areas (right). Red circles correspond to the sizes of Gaussian kernels used for smoothing. [13]

2.2.3 FREAK

FREAK (Fast Retina Keypoint) [18] is a binary descriptor whose sampling pattern was inspired by the human visual system, more specifically by the density of ganglion cells over the retina (see Figure 2.6). Alahi et al. studied biological pathways leading to the action potentials in the nervous system. They emulated this system by a series of binary tests over the pairs of pixel regions.

Unlike the BRISK sampling pattern, sampling points are the densest in the middle of the patch and the density drops exponentially towards the edges. Also, sampled regions overlap each other. The intensity values are smoothed by Gaussian kernels of sizes proportional to their local density.

The way to estimate the orientation of keypoints is very similar to BRISK, however FREAK uses only 45 preselected pairs (see Figure 2.7) opposed to the few hundreds pairs of BRISK. Let $\mathcal{O}$ be the set of these 45 pairs. The orientation θ is then computed as

$$\theta = \frac{1}{M} \sum_{(p_i,p_j)\epsilon\mathcal{O}} \left(I(p_i) - I(p_j)\right) \frac{p_i - p_j}{\|p_i - p_j\|} \tag{2.13}$$

where p is a vector of the spatial coordinates of a centre of a particular receptive field.

The construction of a descriptor is again very similar to BRISK. Let D be the descriptor vector, $\mathcal{P}$ the set of pairs of receptive fields and N its size. Then

$$D = \sum_{0 \le a \le N} 2^a T(p_i^a, p_j^a) \tag{2.14}$$

where

$$T(p_i^a, p_j^a) = \begin{cases} 1 & \text{if } I(p_i^a) > I(p_j^a) \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

and where (p_i^a, p_j^a) is the a-th pair from the set $\mathcal{P}$.

The pairs were not selected according to their spatial distance between the receptive fields because this way some of the pairs may be correlated and thus not sufficiently descriptive. A better way to select the most descriptive pairs is to use training data, compute $T(p_i, p_j)$ for all pairs, select a pair with the highest variance and iteratively add pairs which have both high variance and low correlation with the already selected points. The authors used a dataset of fifty thousand keypoints to select 512 pairs.

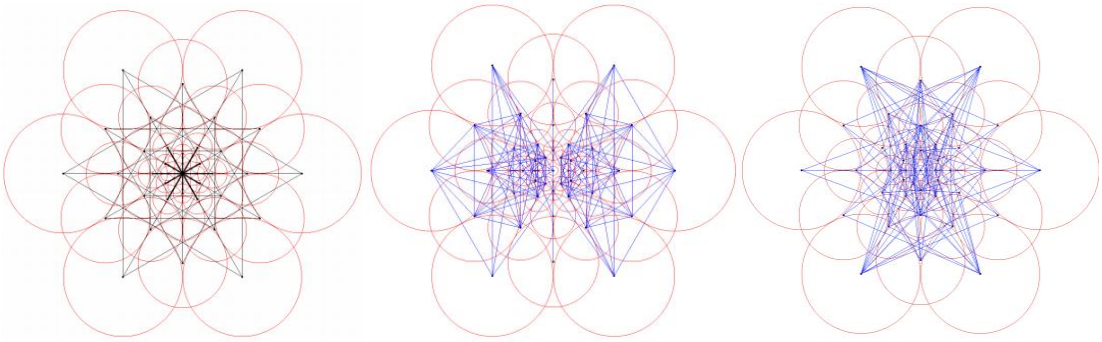


Figure 2.7 FREAK pattern pairs. Orientation pattern pairs (left), 128 pattern pairs of the coarsest cascade (middle) and 128 pattern pairs of the finest cascade (right). [13]

Interestingly, there is a structure in the selected pairs. A coarse-to-fine ordering was automatically achieved. Bits are divided into 16-byte-long cascades which might be used for matching consecutively. The authors observed that 90% of candidate matches were discarded with the first 16 bytes of the descriptor. Remaining tentative correspondences may be further refined with the finer cascades. The choice of 16 bytes per cascade is there to match the hardware requirements as the authors note that the time to compare 1 or 16 bytes is almost equivalent with Single Instruction and Multiple Data (SIMD) instructions on Intel processors since operations are performed in parallel.

Comparison of properties of mentioned binary descriptors is shown in Table 2.1.

	Sampling pattern	Orientation	Sampling pairs
BRISK	Concentric circles with more points on outer rings.	Comparing gradients of long pairs.	Using only short pairs.
FREAK	Overlapping concentric circles with more points on inner rings.	Comparing gradients of preselected 45 pairs.	Learned pairs.

Table 2.1 Comparison of BRISK and FREAK descriptors. [53]

2.3 Feature based image retrieval

Euclidean distance for float feature vectors and Hamming distance for bit feature vectors are the usual measures of distance for descriptors. However, the time complexity of finding the nearest neighbours of n d-element-long descriptors in a database of m points is $\mathcal{O}(nmd)$, which is too slow even for mid-sized databases. Luckily, there are other faster ways to obtain tentative correspondences.

2.3.1 Bag of Words

Bag of words (BoW) is a simplifying model for matching documents commonly used in natural language processing. It represents a document as a multiset (bag) of words disregarding grammar or word order. The multisets may be also understood as histograms of words. The comparisons of the documents are then done as weighted intersections of these multisets. There are several different approaches to weight the elements to obtain the best results.

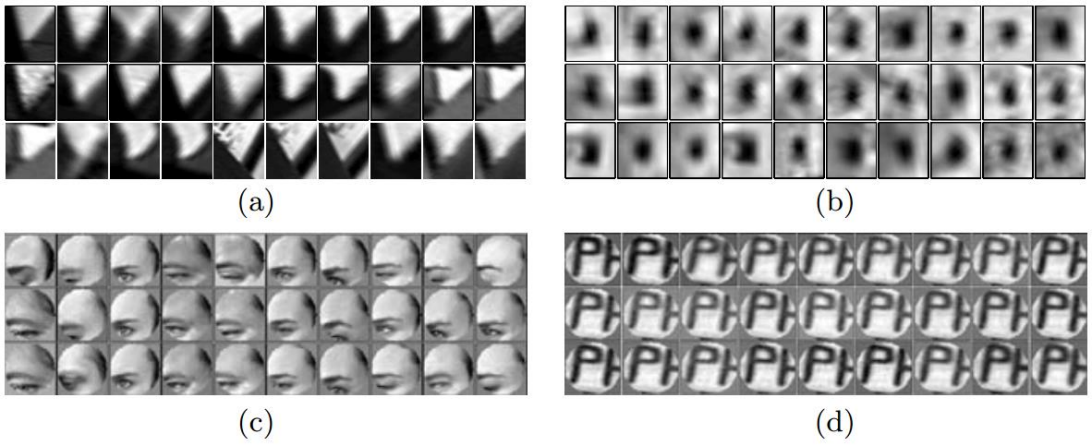


Figure 2.8 Samples of normalized regions which belong to the same visual words. [28]

Sivic and Zisserman [35] used analogous approach in the image retrieval domain. The application was to query with an image of an object and find all video frames

containing this object. Each frame was used like a single document and the descriptors of the image analogically corresponded to the words in the document.

First of all, descriptors from the training set were quantized into the visual words which is analogous to stemming[8]. The descriptors were clustered into k groups using the K-means algorithm. Each cluster centre represented one visual word and every descriptor was assigned to its nearest cluster centre. Examples of local features which belong to the same cluster centre can be seen in Figure 2.8.

Then, an inverted file was constructed. Inverted file is a structure which holds a list of images in which each visual word appeared together with a number of its occurrences. This structure can be easily represented as a sparse matrix where the rows are the indices of visual words and the columns correspond to the images in the database. Elements of this matrix are sums of occurrences of the visual words in the particular images.

At the query time, the descriptors are obtained, assigned to the visual words and a sparse vector is constructed. The similarity measure is defined as normalized scalar product (cosine of angle) of the sparse vectors

$$\mathrm{score}(v_q, v_d) = \frac{v_q^\intercal v_d}{\sqrt{v_q^\intercal v_q}\sqrt{v_d^\intercal v_d}}. \tag{2.16}$$

This can be obtained by simply multiplying the sparse vector and the inverted file matrix under the assumption that both are already pre-normalized.

Analogously to the textual case, some words are more descriptive and some hold no information at all. That is why a stop list was used and the most 5% and the least 5% occurred words were eliminated. Furthermore, *tf-idf* weights were used. *Tf* (term frequency) says how often a term occurs in a document. It is supposed to decrease weights of long documents. It is defined as

$$\mathrm{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{2.17}$$

where $n_{i,j}$ is frequency of a term t_i in a document d_j. *Idf* (inverse document frequency) says how often a word occurs in the whole database and it upweights the less frequent and thus more descriptive terms. It is defined as

$$\mathrm{idf}_i = \log \frac{|D|}{|\{j : t_i \epsilon d_j\}|} \tag{2.18}$$

where $|D|$ is size of the database, i.e. number of documents.

[8] Stemming is a process of reducing inflected words to their stems or root forms used in linguistic morphology.

Images with the best score are selected as candidates. The advantage of this approach is that the tentative correspondences of keypoints are already specified according to the visual words and no further matching is needed. Some of the disadvantages are that the images with repetitive structures get substantially higher scores and that assigning of descriptors to visual words may take long time for large vocabularies. The problem of repetitive structures was worked out among others by [36] and more efficient ways how to assign the visual words were proposed in [37, 38].

2.3.2 Multi-Probe Locality Sensitive Hashing

Locality sensitive hashing (LSH) is a method for probabilistic dimensionality reduction also often used for approximate nearest neighbour search in high-dimensional data. Approximate nearest neighbour means that the retrieved point is not guaranteed to be the actual nearest one, however the speedup over linear search is often enormous and the results are usually sufficient for real-world applications. The key idea of locality sensitive hashing first introduced in [39] is that hashes of close object will more likely collide than hashes of objects which are far from each other. So unlike the cryptographic hash functions, collisions of hashes of close objects are here a desired property. More formally, let S be the domain of objects and D the distance measure. Locality sensitive hash family $\mathcal{H} = \{h : S \to U\}$ is (d_1, d_2, p_1, p_2)-sensitive for D if for any $p, q \in S$

$$\text{if } D(p,q) < d_1, \text{then } P[h(p) = h(q)] \geq p_1 \tag{2.19}$$

$$\text{if } D(p,q) > d_2, \text{then } P[h(p) = h(q)] \leq p_2 \tag{2.20}$$

where $d_1 < d_2$ and $p_1 > p_2$. There are different LSH families for different distance measures. Performance of a hashing method is highly dependent in the quality of the hashing functions they use. Families for Jaccard similarity, Hamming distance and l_1 and l_2 norms were shown in [39]. Here, each hash function is defined as

$$h_{a,b}(q) = \left\lfloor \frac{a \cdot q + b}{W} \right\rfloor \tag{2.21}$$

where a is a d-dimensional random vector and b is a real number uniformly chosen from the range $[0, W]$.

We can define a family of m-bit hash functions $\mathcal{G} = \{h : S \to U^m\}$ where $g(p) = [h_1(p), h_2(p), \dots, h_m(p)]$ and where each h_i is randomly chosen from $\mathcal{H}$. Probability of collision $P[g(p) = g(q)]$ decreased to ${p_2}^m$ for far objects. However it also decreased to ${p_1}^m$ for close objects. That is the reason why there are used l hash functions $g_1, \dots, g_l \in \mathcal{G}$ to construct indexing data structure. Each hash table is created using one function g_i.

A retrieval scheme is straightforward then. For query object q construct hash vectors $g_1(q), g_2(q), \dots, g_l(q)$, find colliding objects in appropriate hash tables and corresponding hash buckets, compute distances of these candidates and find the closest k ones among them.

This approach certainly decreases the number of distances which need to be computed, however it still requires up to hundreds of hash tables to cover most of the neighbouring data points [19]. The size of hash tables is proportional to the size of the database and memory requirements may exceed the main memory size which causes substantial slowdown. This issue was partially solved by the entropy-based LSH [40]. Suppose the distance of a query q and its nearest neighbour p is R_p. Every hash bucket has a certain probability it contains p so it would make sense to probe the hash buckets with the highest probabilities. However, obtaining this probabilities is cumbersome. Entropy-based LSH rather creates random points $q_1', \ldots, q_i'$ at distance R_p, computes their hashes and adds points retrieved by them to the candidate list. The drawbacks of these approach are that much more hash values need to be computed and there is high probability that the hashes will collide so a lot of the computation is wasteful. Also, the number of newly generated points $q_1', \ldots, q_i'$ and the distance R_p need to be estimated and these values have significant influence on the quality of the results. Entropy-based LSH is 2 to 3 times more space efficient then basic LSH but the query time may increase up to two times [19].

Multi-probe LSH [19] was designed to decrease both the memory demands and the query time. The idea is that if an object p is close to the query q but it is not hashed into the same hash bucket, the hash is likely to be similar, i.e. the distance $D_H[(g(p), g(q))]$ will be small. In order to probe buckets with the highest success probability multi-probe LSH constructs a *hash perturbation vector* $\Delta = (\delta_1, \ldots \delta_m)$. The perturbation vector extends the selection of hash buckets in a following manner: when a hash bucket $g(q) = [h_1(q), h_2(q), \ldots, h_m(q)]$ is probed, the hash bucket $g(q) + \Delta$ is probed as well (see Figure 2.9).

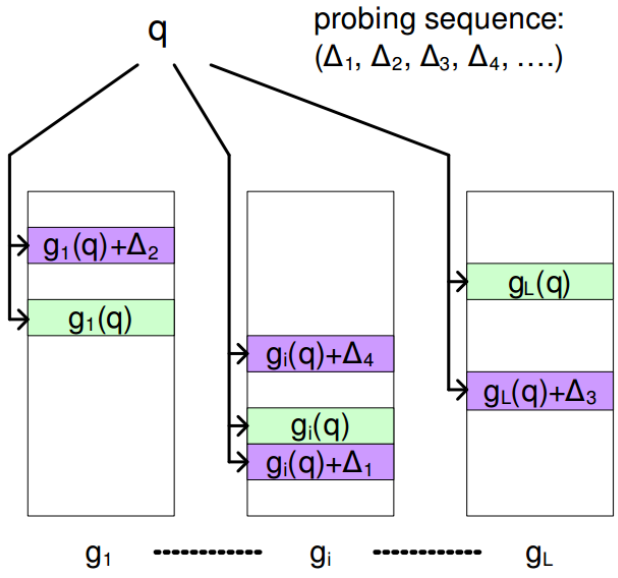


Figure 2.9 Multi-probe LSH search. Green buckets are probed by basic LSH. Multi-probe LSH uses perturbation vectors to probe even the neighbouring buckets (violet). [14]

To probe only the closest neighbouring hash buckets, the elements of perturbation vector Δ are limited in a range $\delta_i \in \{-1,0,1\}$. Since the perturbation vectors are added directly to the hash values, there is no need to compute the point perturbations and their hash values like it is done by the entropy-based LSH. Also, the perturbation vectors map to unique hash values so no hash bucket is probed more than once.

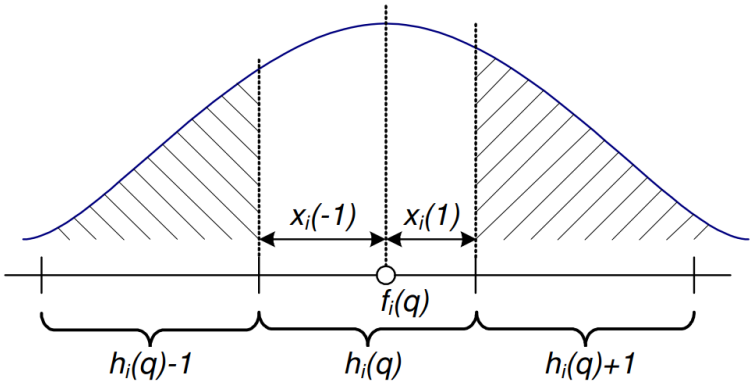


Figure 2.10 Probability of falling into the neighbouring slots for the nearest neighbours of point q. [14]

An n-step perturbation vector Δ has exactly n non-zero elements. The total number of possible n-step buckets is $l \times \binom{m}{n} \times 2^n$. Probing all of these buckets would be impractical. Also, not all these buckets are equally good. Multi-probe LSH uses the information about the position of $h_i(q)$ within the slot of width W to determine the buckets with the highest success probability. Let $x_i(\delta_i)$ be the distance from a point $f_i(q) = a_i \cdot q + b_i$ to the border of the slot of $h_i(q)$ in the δ_i direction (see Figure 2.10). The most promising perturbation vectors are then selected according to the score function

$$\text{score}(\Delta) = \sum_{i=1}^{m} x_i(\delta_i)^2. \tag{2.22}$$

Multi-probe LSH reduces the number of hash tables by a factor of 14 to 18 over the basic LSH and by a factor of 5 to 8 over the entropy-based LSH. It also performs 10 times less probes than entropy-based LSH which makes it more time efficient.

Multi-probe LSH is a general method for approximate nearest neighbour search and so it can be used for the image retrieval task by finding correspondent features from the database for every feature in the query image. A shortlist of possible results can be constructed according to these correspondences. While this might be sufficient for some applications, more complex ones will require further verification of the correspondences and some kind of re-ranking.

2.3.3 Bundle Min-Hashing

Bundle min-hashing is a method for image retrieval proposed by Romberg and Lienhart [21]. It detects SIFT features and it describes neighbourhood of each keypoint by a feature

bundle b of its neighbouring keypoints on similar scales (see Figure 2.11). The features are quantized to visual words just like for the bag of words. Features within the bundles are indexed by min-hashing.

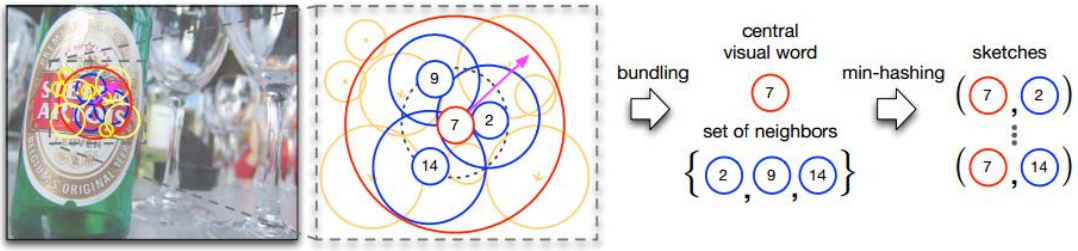


Figure 2.11 Bundle min-hashing. The neighbourhood of a local feature (red) is described by its neighbouring features which are on the similar scales (blue). These features are quantized to visual words and the neighbours are min-hashed to form the sketches. [20]

Min-hashing is a locality-sensitive hashing technique used for approximate similarity search of sparse sets. Two images may be represented by sets of their visual words I_1 and I_2. Min-hashing approximates the Jaccard similarity which measures overlap of such sets as

$$J(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|}. \tag{2.23}$$

Given set of n visual words $I = \{v_1, \dots, v_n\}$ and a hash function h which deterministically maps each visual word to a random value from a uniform distribution, the min-hash function mh is defined as

$$mh(I) = \underset{v_i \in I}{argmin}\, h(v_i). \tag{2.24}$$

The probability that a min-hash function has the same value for two sets is the same as their overlap. More formally

$$P\big(mh(I_1) = mh(I_2)\big) = J(I_1, I_2) = \frac{|I_1 \cap I_2|}{|I_1 \cup I_2|}. \tag{2.25}$$

Many different min-hash functions must be used since only one single min-hash function would not be descriptive enough. Using more min-hash function results in more false positives. Min-hashes are often coupled to k s-tuples called sketches to improve precision of the retrieval. Sketches collide only if all their min-hashes agree. Probability of a successful retrieval is then

$$P\{retrieval\} = 1 - (1 - J(I_1, I_2)^s)^k. \tag{2.26}$$

Bundle min-hashing uses different sketches. Let $b(x_i)$ be the feature bundle of a feature x_i. Formally

$$b(x_i) = \{x_j | x_j \in N(x_i)\} \tag{2.27}$$

where $N(x_i)$ is a set of neighbouring features. A sketch for a given min-hashing function is

$$\left(v_i, mh\left(W\left(b(x_i)\right)\right)\right) \tag{2.28}$$

where v_i is the visual word label of feature x_i and W is a function which maps the set of features $b(x_i)$ to a set of appropriate visual word labels.

Collisions of such sketches determine the set of possible corresponding images. Authors observed very small response ratio compared to the bag of words method, e.g. the precision of retrieval is much higher and mostly relevant correspondences are retrieved. This reduces the time needed for re-ranking of the tentative correspondences. Authors furthermore propose a method for fast re-ranking 1P-WGC-RANSAC which uses retrieved correspondences of sketches to estimate the spatial transformations between images. Each correspondence has its associated scale and orientation which are used to determine a similarity transformation. All of these transformations are then evaluated. Their spatial consistency with the rest of the correspondences determines their support. Features which are not consistent with the most supported hypothesis are filtered out. Retrieved images are re-ranked by the number of verified correspondences at the end.

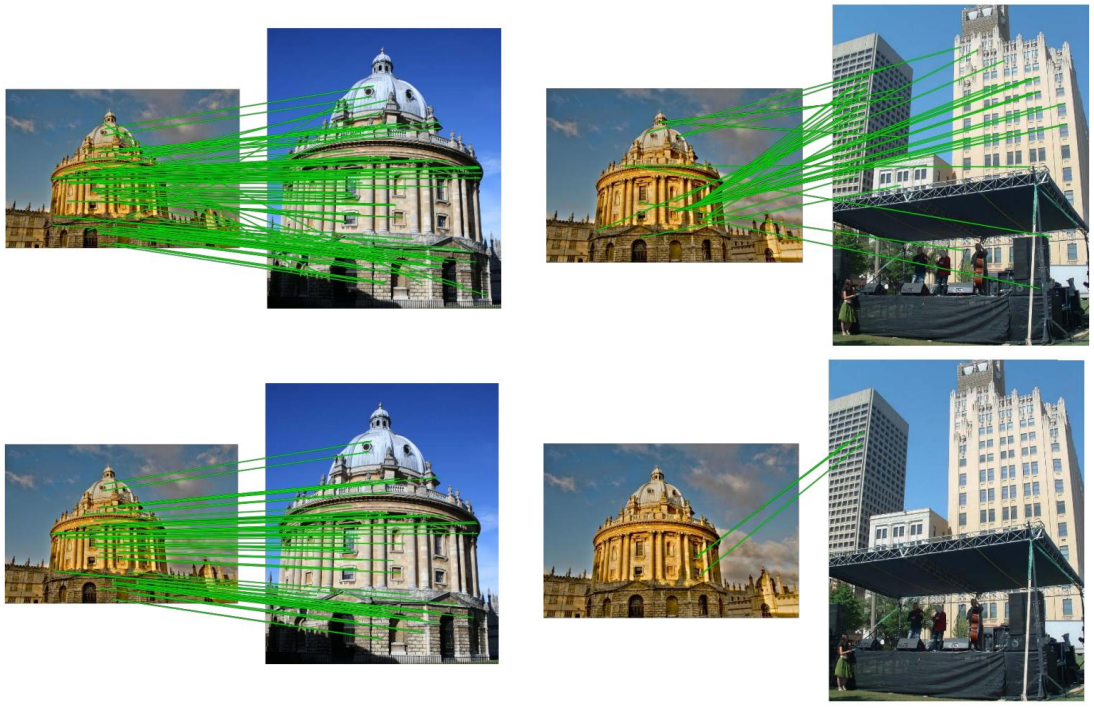


Figure 2.12 Geometrical verification. Upper row: There might be a lot of false matches among the tentative correspondences (green); Lower row: Geometrical verification filters out inconsistent outliers. [57]

2.4 Geometrical Verification

Nearest neighbour algorithms may return a lot of tentative correspondences and even though distances of descriptors may be small and keypoints match locally, they don't necessarily have to correspond to the same object in the scene. So images with a large number of tentative correspondences don't necessarily have to depict the same objects (see Figure 2.12). Therefore some sort of post-verification is often used to filter out inconsistent matches.

2.4.1 Weak Geometrical Constraints

Semilocal constraints were proposed in [41] and used in [35]. Spatial consistency of a match of keypoints is defined rather loosely by requiring that at least n out of the k nearest features in the query image must correspond to the k nearest features in the retrieved image. The best value for n was empirically set to be a half of k. In order to increase the recognition rate even more, a geometric constraint is added. It requires the mutual angles of the neighbouring features to be locally consistent (see Figure 2.13).

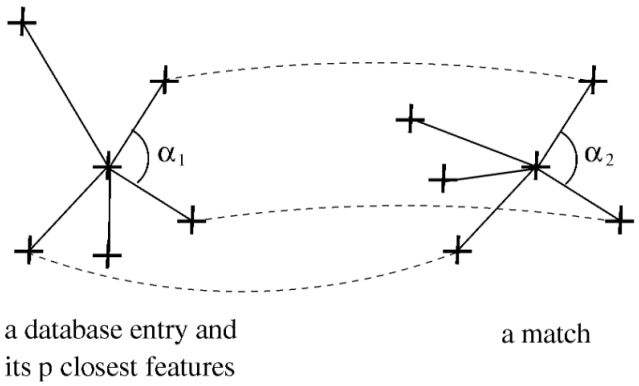


Figure 2.13 Semilocal geometrical verifications. In order to be verified, neighbouring features of a match must correspond as well as their mutual angles. [34]

2.4.2 Hough Transform

Generalized Hough transform [42, 43] is a method in computer vision which uses voting procedure to find the most probable model on data. Each candidate point votes for all models which it is consistent with. These votes are accumulated in quantized multidimensional array. Local maxima correspond to the most probable model parameters (see Figure 2.14).

Hough transform as widely used technique as it is applicable on many different problems and can handle high percentage of outliers. Its disadvantages may be possible problems with quantization and that is impractical for high-dimensional models.

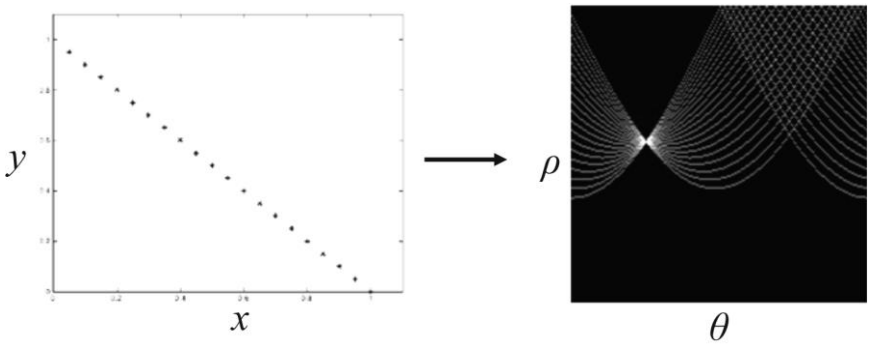


Figure 2.14 Hough transform. Tested data points (left) and their votes (white) for models in polar coordinates (right). The coordinates of the most supported model are clearly distinguishable. [59]

2.4.3 RANSAC

Random sample consensus (RANSAC) [44] is a popular probabilistic and iterative algorithm for robust model estimation. It randomly selects a minimal sample s from a set of points $\mathcal{S}$ in order to create a hypothesis. Then it calculates an error function for each data point in $\mathcal{S}$, selects data which support the current hypothesis and computes score of the hypothesis (see Figure 2.15). This procedure is repeated until stopping conditions are met. The final outcome is the hypothesis which achieved the best score.

Probability that all points in the sample are inliers is

$$\mathrm{P}_{good} = \frac{\binom{I}{m}}{\binom{N}{m}} = \prod_{j=0}^{m-1} \frac{I-j}{N-j} \tag{2.29}$$

where N is size of the set $\mathcal{S}$, I is number of inliers and m is size of the minimal sample. If we want to have confidence c that at least one of the samples contains only inliers, then

$$\left(1 - \mathrm{P}_{good}\right)^k \leq 1 - c \tag{2.30}$$

where k is number of trials. This means that at least

$$k \geq \frac{\log(1-c)}{\log\left(1 - P_{good}\right)} \tag{2.31}$$

trials must be performed to achieve confidence c that the true set of inliers was found.

RANSAC can handle even small fraction of inliers, although the number of necessary trials grows polynomially with this fraction and exponentially with the size of the minimal sample m. RANSAC is a very general robust estimation method, yet there exist a lot of algorithms which improve some of its imperfections, e.g. Locally Optimized RANSAC [45], DEGENSAC [46], PROSAC [47] or WaldSAC [48].

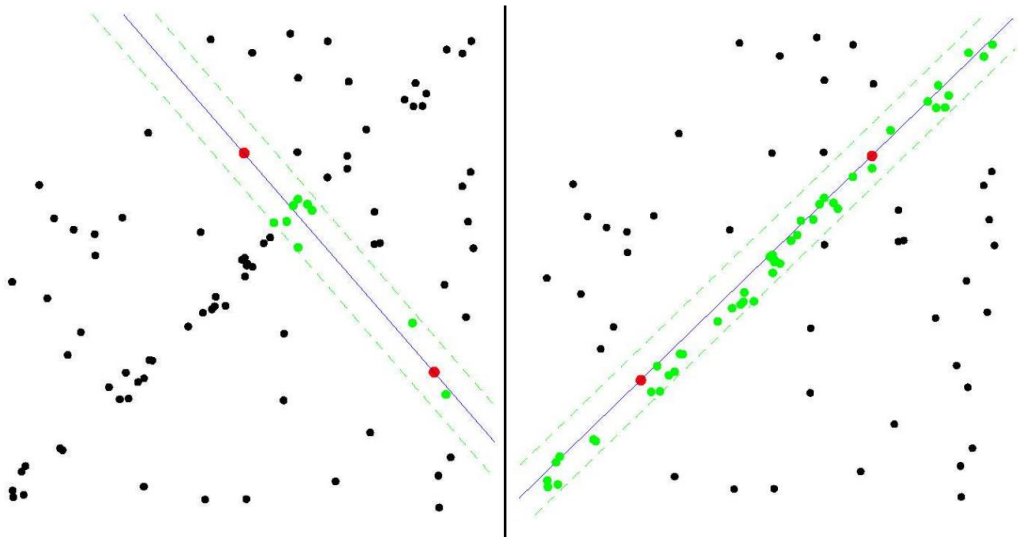


Figure 2.15 RANSAC hypotheses. Two different hypotheses on the same data (black). Hypothesis (blue) constructed from two sample points (red) and points which support this hypothesis (green). [58]

3 Method

First, we will review the requirements on the system. To satisfy the specification and requirements we have to make a choice of detector, descriptor, matching algorithm and a method of final verification. We will first study state-of-the-art for components and then propose their composition.

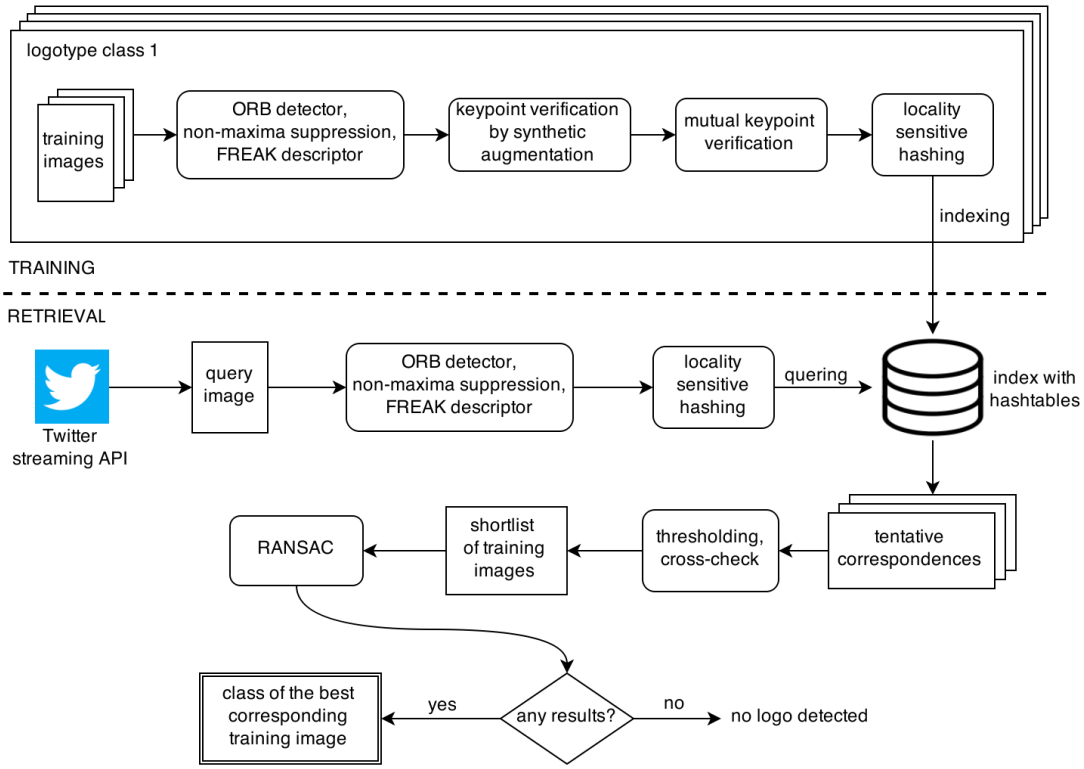


Figure 3.1 Image retrieval for logo recognition. This flowchart shows the training as well as the retrieval part of the algorithm. The rectangles represent data structures while the ones with the rounded edges represent functions and processes. Individual parts of the algorithm are further described in this chapter.

3.1 Requirements

Hundreds of millions images are shared on social media every day, however most of them are not publicly accessible. We focus only on Twitter for which we estimated tens of millions of daily images. We found out that at least two thirds of these images are duplicates. There is also vast amount of spam and some images do not need to be checked because some users do not have sufficient audience or come from unrelated geographical region and so on. The final estimate are low millions of images to be checked per day, i.e. dozens of images per second.

The number of queries is enormous and images which contain sought logotype occur very rarely. We estimated the *a priori* probability that an image contain desired logotype between 10^{-5} and 10^{-4}. Therefore near perfect precision is required otherwise false

positives would outnumber true positives in the results. On the other hand, perfect recall is not necessary since we are interested in observing changes in trends during campaigns rather than aiming to retrieve all presented images with tracked logos.

There are no computational requirements on the offline stage of the algorithm.

3.2 Image Description

3.2.1 Feature Detection

Detector part of ORB algorithm was chosen for detecting the keypoints. According to the measurements in Table 5.2, it is the fastest detector which provides scale invariance and orientation estimation. It detects FAST corners across the Gaussian scale space and it uses Harris response function to select the most stable features. The selection is performed within each octave independently. Number of retrieved keypoints is proportional to the size of the image in the particular level of the Gaussian pyramid. Let N_t be the maximum number of desired keypoints, L the number of levels of the Gaussian pyramid and s_f its scaling factor. Then, the number of keypoints in the l-th level is

$$n_l = N_t \frac{1 - \frac{1}{s_f}}{1 - \left(\frac{1}{s_f}\right)^L} \left(\frac{1}{s_f}\right)^l. \tag{3.1}$$

ORB implementation in OpenCV allows to set any factor for the first level of the Gaussian pyramid, i.e. some of the images in the pyramid may be even larger than the original image.

FAST keypoints are detected after finding an arc of pixels on a circular neighbourhood which are either all brighter or all darker than the centre pixel by some threshold. This procedure already divides the keypoints to two sets – either dark corners on a bright background or dark corners on a dark background. Even though this is not often mentioned in the literature, it semantically makes sense to treat these two sets of keypoints separately. This mainly saves time during the matching phase when distances of keypoints of different types are not computed as it is unlikely these two keypoints correspond to the same object in the scene anyway.

3.2.2 Spatial Non-Maxima Suppression

Common feature detection phenomenon is that keypoints with the highest corner responses may be found primarily in one or few regions in the picture, e.g. trees or richly textured areas. This certainly decreases the descriptiveness of the keypoint set and more uniform distribution is desired. Adaptive non-maxima suppression which solves this problem was proposed in [49]. Keypoints are sorted in descending order according to their corner responses and they are iteratively added to a set of better distributed keypoints.

At the time when a keypoint is added to this set, remaining surrounding keypoints within a certain radius are checked. If their corner response is smaller by some factor than the corner response of the currently added keypoint, they are removed from the list of candidate keypoints and they cannot be added to the final set anymore.

This is a functional technique, however it requires some computation since k-NN search needs to be performed in order to find surrounding keypoints. Any unnecessary computation is undesirable, so slightly different approach is proposed. Rather than scanning the neighbourhood of each keypoint, a grid over the image is constructed. Keypoints are again added one by one according to their corner responses. There is a limit k_c of maximum number of keypoints in every grid cell. When this limit is reached, no more keypoints can be added to the particular cell. Since the sizes of the query images may be diametrically different, we suggest to specify the maximum number of retrieved keypoints for the whole image N_t with respect to the size of the picture, i.e.

$$N_t = N_{Mpx} \cdot n_r \cdot n_c \cdot 10^{-6} \tag{3.2}$$

where N_{Mpx} is the number of desired keypoint by one megapixel, n_r is the number of rows of the image and n_c is the number of its columns.

Figure 3.2 Non-maxima suppression. No keypoint (red) covers the HP logo after using standard ORB detector (left). Using spatial non-maxima suppression distributes the keypoints more uniformly (right).

We want the grids cells to be squarish and to have the same size. To achieve both of these properties, the size of the cells is $c_r \times c_c$ pixels where

$$c_r = \frac{n_r}{\left\lceil \sqrt{\frac{N_{Mpx}}{k_c} \cdot \frac{n_r}{n_c}} \right\rceil} \tag{3.3}$$

$$c_c = \frac{n_c}{\left\lceil \sqrt{\frac{N_{Mpx}}{k_c} \cdot \frac{n_c}{n_r}} \right\rceil}. \tag{3.4}$$

Figure 3.2 shows how spatial non-maxima suppression helps to distribute the keypoints more uniformly and to find keypoints on logos with lower contrast.

3.2.3 Feature Description

FREAK descriptor algorithm was used to obtain the feature vectors. It performed very well in the tests (see Table 5.1) and it also provides orientation estimation. As it is described in [18], 512 bits it the FREAK descriptor are somewhat ordered from coarse to fine. So it is possible to shorten the length of the descriptor in order to speed up the matching process when looking for the fine correspondences is not necessary. Shortened feature vectors of lengths 256 and 384 bits along with the full length vectors of 512 bits were used in the experiments. Matching time is proportional to the lengths of the descriptors and shorter descriptors results in faster matching. However, shortening of the descriptors also leads to slightly worse recall. Three-quarter 384-bit-long descriptors were finally selected as they performed the best trade-off between speed and recall.

3.2.3.1 Colour Description

Since logotypes usually have their distinctive colours, we attempted to add this information to the descriptor. RGB intensities were not used directly due to their susceptibility to illumination shifts. Patches were first converted to the HSV colour space. Pixel values of the hue channel were weighted by their saturation and quantized to a histogram h_c of length N. See Figure 3.3 for examples of shapes of hue histograms for different patches. The new colour feature vector F_c was constructed as

$$F_c = \sum_{0 \le a < N} 2^a T[h_c(a)] \tag{3.5}$$

where

$$T[h_c(a)] = \begin{cases} 1 & \text{if } h_c(a) \ge th_c \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

where th_c is empirically set threshold. Colour feature vectors F_c were then appended to the original FREAK descriptors.

Even though additional colour information in the descriptors helped to detect some logos which were not retrieved before, it had the opposite effect on many others. Furthermore, some logos appear in a broad variety of colours (typically clothing companies but even some more) so each version need to be represented in the database. After all, using the colour information did not improved the overall accuracy so it was not worth the additional computation and it was not used.

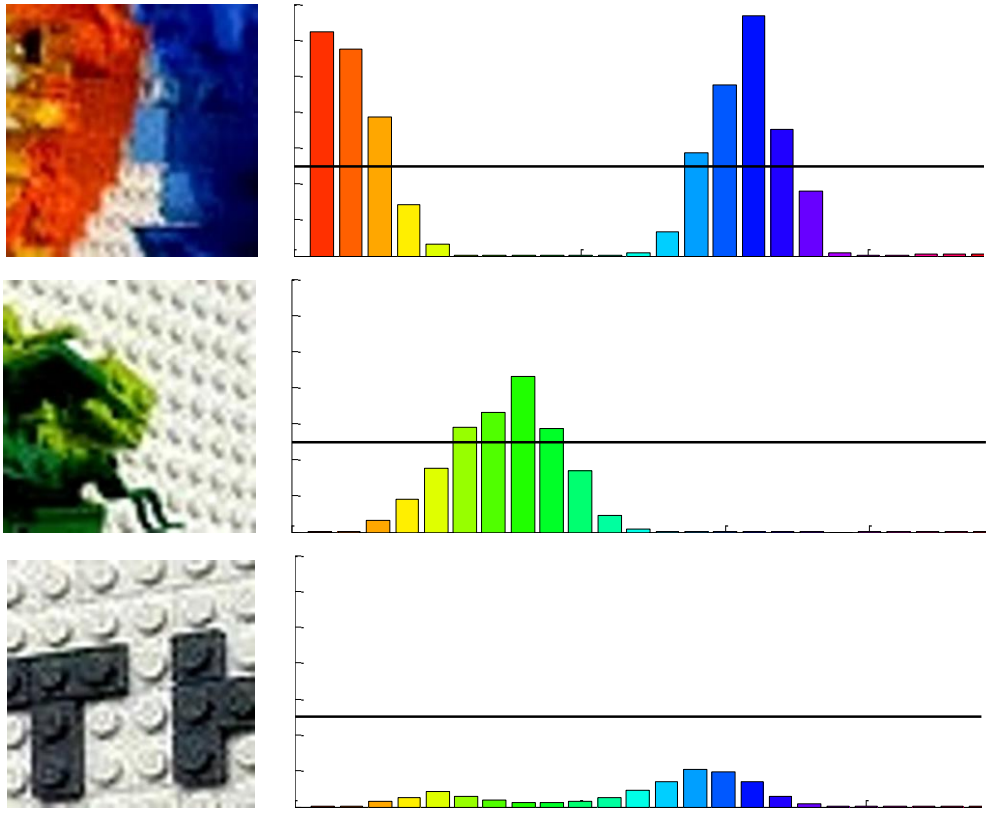


Figure 3.3 Weighted hue histograms. Different patches are shown on the left and their hue histograms are shown on the right. Black lines show the threshold th_c. Note that patch without any dominant colour has subsequently empty colour feature vector F_c (bottom).

3.3 Preprocessing of Training Images

There is no time constraint on the offline training stage of the algorithm, so more precise and thus more time consuming operations may be used to obtain the best quality features. To filter out unstable keypoints with low repeatability from the training set, an algorithm using synthetic augmentation of training images and mutual keypoint verifications is proposed. It is partially inspired by the synthetic database augmentation used in [21].

Algorithm 3.1 Synthetic image augmentation for keypoint filtering

Input: Image I, set of projection matrices $\mathcal{P}$.
Output: Set of transformed images and set of sets of their keypoints.
projectedImages $\leftarrow \{I\}$
for each P **in** $\mathcal{P}$
    $I_p \leftarrow$ TRANSFORM-IMAGE(I,P)
    projectedImages $\leftarrow$ APPEND(I_p)
end for
projectedKeypointsSet $\leftarrow \emptyset$
for each *image* **in** *projectedImages*
    keypoints $\leftarrow$ FIND-KEYPOINTS(*image*)
    projectedKeypointsSet $\leftarrow$ APPEND(*keypoints*)
end for
filteredKeypointsSet $\leftarrow \emptyset$
for each *keypoints* **in** *projectedKeypointsSet*
    filteredKeypoints $\leftarrow \emptyset$
    $P \leftarrow$ GET-TRANSFORM(*keypoints*)
    for each k **in** *keypoints*
        $k_p \leftarrow$ INVERSE-TRANSFORM(k,P)
        $z \leftarrow 0$
        for each *differentKeypoints* **in** *projectedKeypointsSet* \ *keypoints*
            $Q \leftarrow$ GET-TRANSFORM(*differentKeypoints*)
            for each h **in** *differentKeypoints*
                $h_q \leftarrow$ INVERSE-TRANSFORM(h,Q)
                if DISTANCE(k_p,h_q) $< th_f$ **and** SAME-SCALE(k_p,h_q)
                    $z \leftarrow z + 1$
                end if
            end for
        end for
        if $z \geq k_f$
            filteredKeypoints $\leftarrow$ APPEND(k)
        end if
    end for
    filteredKeypointsSet $\leftarrow$ APPEND(*filteredKeypoints*)
end for
return *projectedImages*, *filteredKeypointsSet*

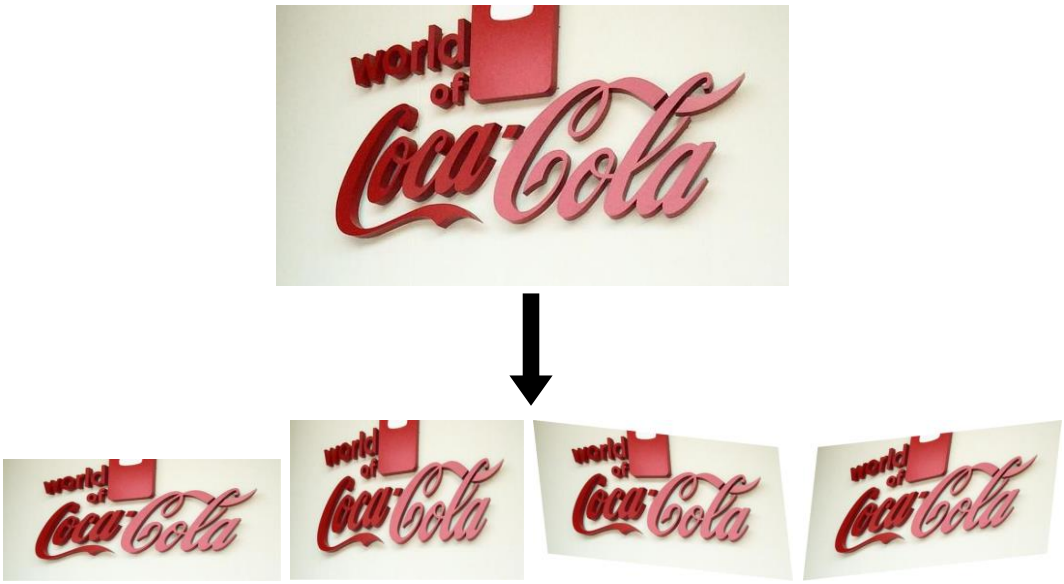


Figure 3.4 Synthetic augmentation. The training image is shrinked and skewed in both x- and y-direction.

Four different trans formations were used to augment the training images. Effects of these transformations can be seen in Figure 3.4.

Let α by the scaling factor. The transformation matrices are $S_x(\alpha)$, $S_y(\alpha)$, $K_x(\alpha)$ and $K_y(\alpha)$, where $S_x(\alpha)$ shrinks the image in x-direction by factor α and $K_x(\alpha)$ skews the image along the x-axis and rotates it appropriately. Elements of the transformations matrices look as follows:

$$\begin{aligned} S_x(\alpha) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \sqrt{\alpha} \end{bmatrix}, \quad K_x(\alpha) = \begin{bmatrix} (\alpha+1)^2 & \alpha^2-1 & 0 \\ \alpha^2-1 & (\alpha+1)^2 & 0 \\ 0 & 0 & 4\alpha \end{bmatrix}, \\ S_y(\alpha) &= \begin{bmatrix} \alpha & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \sqrt{\alpha} \end{bmatrix}, \quad K_y(\alpha) = \begin{bmatrix} (\alpha+1)^2 & 1-\alpha^2 & 0 \\ 1-\alpha^2 & (\alpha+1)^2 & 0 \\ 0 & 0 & 4\alpha \end{bmatrix}. \end{aligned} \tag{3.7}$$

The design of the transformation matrices ensures preservation of area of the images and thus also the area of the local features. Therefore the local features within the same octave can be directly compared without any further conversion of their scales. See Figure 3.5 for visualisation of shapes of warped local features.

The algorithm obtains new images by projecting the original image using the transformation matrices. ORB keypoints are located in each image independently. Each keypoint of each image is compared with all the other keypoints of the other images which were found within the same octave. The keypoint is preserved only if there is at least k_f images which contain spatially corresponding keypoints, i.e. the distance between the tested keypoint and some keypoint from another image is lower than threshold th_f. Of course, to measure distances between keypoints from different images, the coordinates

must be projected to the same basis using the inverses of the corresponding transformation matrices. For more precise description see Algorithm 3.1.

Threshold th_f was set to 1 pixel to be sure only truly corresponding local features are compared. The effect of choosing different k_f can be seen in Figure 5.2. $k_f = 1$ showed to be adequate to filter out most of the undesired keypoints and simultaneously maintain the sufficient number of the filtered ones. Unlike the synthetic database augmentation in [21], new synthetic images serve only for the filtering of the keypoints and they are not add to the training set. Adding these similar images to the database resulted in slowing down the retrieval without any noticeable increase of recall.

The previous algorithm certainly filters out keypoints with low repeatability from the training images, however it may still preserve keypoints which do not belong to the local features of the logo, e.g. because of noise or some unrelated texture which is also

Algorithm 3.2 Keypoint filtering using multiple images of the same logotype

Input: Set of sets of keypoints $\mathcal{K}$ of images of the same logotype.
Output: Set of sets of filtered keypoints $\mathcal{K}_f$.
$occurrences \leftarrow \emptyset$
for each *keypointSet* **in** $\mathcal{K}$
    $occurrences \leftarrow$ append vector of zeros of size of *keypointSet*
end for
for $i \leftarrow 1$ **to** $\mathcal{K}$
    for $j \leftarrow i{+}1$ **to** $\mathcal{K}$
        $matches \leftarrow$ FIND-CLOSEST-MATCHES($\mathcal{K}(\boldsymbol{i})$, $\mathcal{K}(\boldsymbol{j})$)
        $H \leftarrow$ RANSAC($matches$)
        $inliers \leftarrow$ FIND-INLIERS($matches$,H)
        for each $\{keypointId_i$,$keypointsId_j\}$ **in** $inliers$
            $occurrences[i][keypointId_i] \leftarrow occurrences[i][keypointId_j] + 1$
            $occurrences[j][keypointId_j] \leftarrow occurrences[j][keypointId_j] + 1$
        end for
    end for
end for
$\mathcal{K}_f \leftarrow \emptyset$
for $i \leftarrow 1$ **to** $|\mathcal{K}|$
    $filteredKeypointSet \leftarrow \emptyset$
    $occurrenceVector = occurrences(i)$
    $keypointSet = \mathcal{K}(i)$
    for $j \leftarrow 1$ **to** $|keypointSet|$
        if $occurrenceVector(j) \geq th_{in}$
            $filteredKeypointSet \leftarrow$ APPEND($keypointSet(j)$)
        end if
    end for
    $\mathcal{K}_f \leftarrow$ APPEND($filteredKeypointSet$)
end for
return $\mathcal{K}_f$

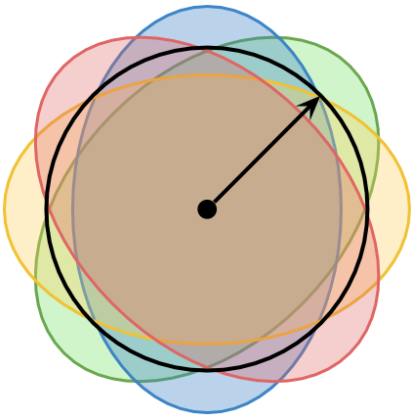


Figure 3.5 Transformation of local features. Area of the original local feature is shown in black. Shapes of new local features obtained by the synthetic augmentation are shown in colours. Each colour correspond to one projection. Note that the local features are warped however the area remains consistent.

presented in the image. Another way to filter out keypoints is to use other images of the same logotype. The method described in Algorithm 3.2 uses linear-search algorithm to find correspondences of keypoints with the closest descriptors in a pair of images and then a homography is found using RANSAC. This is done for every pair of images and the keypoints which did not figure as inliers at least th_{in} times are removed (see Figure 3.6).

The purpose of this step is not to find precise transformations between the training images. The purpose is to filter out unrelated local features and the transformations serve only to verify the spatial correspondences. Therefore, the constraints on RANSAC inliers may be rather loose. Threshold th_{RANSAC} was set to 8 pixels, i.e. a match was considered as an inlier if the spatial difference between a keypoint from the first image and a projected keypoint from the second image was smaller than 8 pixels.

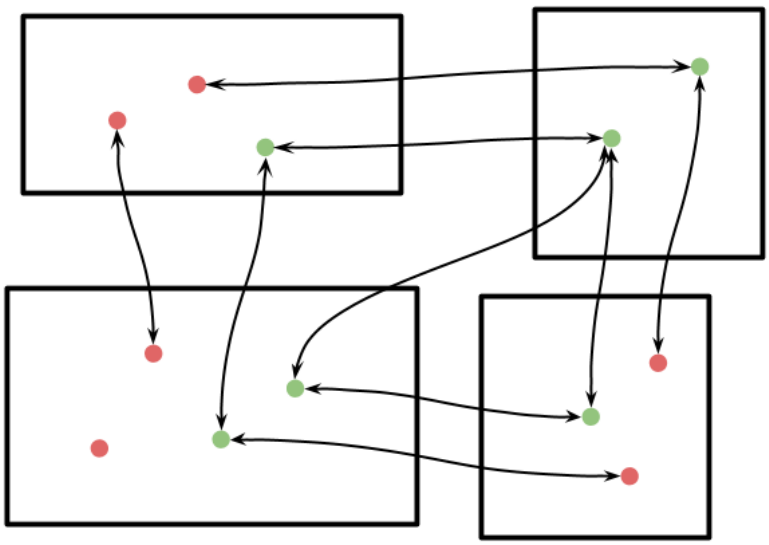


Figure 3.6 Mutual verification of keypoints. Black arrows represent geometrically verified correspondences of keypoints. With $th_{in} = 2$, red keypoints will be filtered out and only green ones will stay. Notice that a keypoint may be filtered out whereas its corresponding keypoint remains.

3.4 Feature matching

Fast retrieval of the nearest neighbours of keypoints of query images is the crucial part of the application. Linear search which would find the exact nearest neighbours is not applicable due to its inefficiency. Much more effective approximate nearest neighbour methods are sufficient in this case. Multi-probe locality sensitive hashing was chosen for the nearest neighbour retrieval since it has decent performance and works well for the Hamming distance measure unlike some other nearest neighbour algorithms. Recall/speedup trade-off of the multi-probe LSH is shown in Figure 5.3. Multi-probe LSH with 15 hashing tables, 29-bit-long hashes and 1-step perturbation vectors is used for all further measurements as it was the fastest configuration with 60% recall for 5-NN and with 50% recall for 10-NN.

Let $\mathcal{Q}$ be a set of query images, $\mathcal{B}$ a set tested brands and $\mathcal{R}$ a set of training images.

$$\mathcal{R} = \left\{ r_j^b \mid b \in \mathcal{B}, 1 \leq j \leq n_b \right\} \tag{3.8}$$

where r_j^b is the j-th image out of n_b images of a brand b. Keypoints of image r_j^b are denoted k_j^b. Analogically, keypoints of a query image $q \in \mathcal{Q}$ are denoted k_q.

The retrieval Algorithm 3.3 works as follows. First of all, LSH hash values are computed for all training keypoints. Then, for each keypoint from k_q of query image q are retrieved its n_{kNN} nearest neighbours. These newly obtained matches correspond to different training images, so each match is assigned to its appropriate image.

So far, there was no constrain on the Hamming distance between the matched keypoints. The larger the distance is, the less likely the pair of keypoints correspond to the same object in the scene (see Figure 3.9). That is why all matches which have the Hamming distance larger than a threshold th_{Hamm} are filtered out.

There may still be an enormous number of matches between some pairs of images although most of them are most likely false positives. In theory, RANSAC, which we use for geometrical verification, can handle even a large portion of outliers, however at the cost of performing more trials and thus substantially slowing down the computation. Also, there is often a case that several keypoints from one image correspond to a single keypoint in the second image. This is obviously wrong since only one of these matches can be true positive. Moreover, geometrical verification by estimating spatial transformation between images may fail as the projection of the first image to the second one degenerates to a single point. Filtering according to the ratio between the closest and the second closest match used by Lowe [31, 32] is not sufficiently applicable for the Hamming distance. We rather use mutual nearestness of the keypoints, another widely used method to obtain consistent matches. A match stays in the set of tentative correspondences only if the keypoint from the first image is the closest one to the keypoint from the second image and vice versa. We found out this condition was too restrictive for some pairs of images with smaller number of correspondences, therefore we loosened this condition by allowing

even the second closest matches to be satisfactory. An important part of this step is that no additional keypoint distances are computed, i.e. only the distances computed during the multi-probe LSH retrieval are used.

Even though RANSAC is generally considered as a fast method of geometrical verification, it would be overwhelming to perform it for every training image. Only the k_{best} most promising images are selected. A natural way to determine the best candidates is by the number of their tentative correspondences. However, images with a large number of keypoints have higher probability of finding incorrect random matches which may easily overweight true matches of other images with not so many keypoints. Weighting the number of correspondences by the number image keypoints led to the opposite

Algorithm 3.3 Brand recognition

Input: Set of training image keypoints $\mathcal{R}$ and query image keypoints q.
Output: Detected brand b.
Offline stage:
    TRAIN-LSH($\mathcal{R}$)
Online stage:
    knnMatches $\leftarrow$ FIND-KNN-MATCHES-BY-LSH(q,n_{kNN})
    $\{m_1^{b_1}, \ldots, m_{n_{b_1}}^{b_1}, m_1^{b_2}, \ldots, m_{n_{b_m}}^{b_m}\} \leftarrow$ ASSIGN-MATCHES-TO-TRAIN-IMAGES(*knnMatches*)
    scores $\leftarrow \emptyset$
    for $m_j^{b_i}$ **in** $\{m_1^{b_1}, \ldots, m_{n_{b_1}}^{b_1}, m_1^{b_2}, \ldots, m_{n_{b_m}}^{b_m}\}$
        $m_j^{b_i} \leftarrow$ KEEP-CLOSE-MATCHES($m_j^{b_i}$,th_{Hamm})
        $m_j^{b_i} \leftarrow$ CROSS-CHECK($m_j^{b_i}$,q)
        $s_j^{b_i} \leftarrow \left|m_j^{b_i}\right| / \sqrt{\left|r_j^{b_i}\right|}$
        scores $\leftarrow$ APPEND($s_j^{b_i}$)
    end for
    bestInliers $\leftarrow \emptyset$
    for $s_j^{b_i}$ **in** FIND-BEST-SCORES(scores,k_{best})
        H $\leftarrow$ RANSAC($m_j^{b_i}$)
        inliers $\leftarrow$ FIND-INLIERS($m_j^{b_i}$,H)
        if $|inliers| > th_{succ}$
            return b_i
        else if $|inliers| > |bestInliers|$
            $b \leftarrow b_i$
            bestInliers $\leftarrow$ *inliers*
        end if
    end for
    if *bestInliers* $> th_{min}$
        return b
    else
        return *none*
    end if

problem, i.e. images with less keypoints were favoured. This happened because the differences among the numbers of keypoints of training images were up to two orders of magnitude from tens to thousands of keypoints. This means that some images needed up to hundred less tentative correspondences to overtake the others. Finally, a score obtained by dividing the number of tentative correspondences by the square root of the number of keypoints appeared to be the best solution which favoured neither the images with a lot of keypoints nor the images with not so many of them.

A short list of length k_{best} is created. Geometrical verification is done on the most promising image first. If the number of its inliers is larger than th_{succ}, its brand is immediately returned as a successful result of the retrieval. If not, the verification is successively performed on the rest of the shortlist to the one with the lowest score. If none of the candidates exceeded th_{succ}, the result is the brand of an image with the largest number of inliers which is bigger th_{min}. If no candidate reached th_{min}, no logo is detected.

Figure 3.7 Deformed projection. Training image (left), query image (middle left), warped training image according to the transformation found without checking of the diagonals (middle right) and warped training image according to the transformation found after the diagonal checking (right).

3.4.1 Geometrical verification

As it was said earlier, RANSAC is used to perform the geometrical verification. Images vary in scales, rotations and viewpoints, however logos remain more or less planar (see Figure 4.2). Homography estimation is an appropriate way to determine geometrical transformation in this case. Homography is a projective transformation which needs a sample of 4 pairs of points to be unambiguously defined. The projection of a point with coordinates $[x_i, y_i]$ is computed as

$$\lambda_i \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{3.9}$$

where $[x_i', y_i']$ are the projected coordinates, H is 3×3 real matrix with rank equal to 3 and λ_i is a scaling factor.

OpenCV version of homography estimation sometimes returned very odd results. Moreover, it obligatorily performs refinement of the projection using the Levenberg-Marquardt method [50] which slows down the computation. So slightly modified version of RANSAC was newly implemented. The candidate hypothesis is still determined by 4 randomly selected pairs of keypoints. However, before projecting the keypoints, computing their distances from their expected positions and counting up the inliers, two simple tests are performed. Only the corner points of the training image are projected by the homography H. Projection of the training image to the query image is expected to be close to the rectangular shape and not too heavily deformed (see Figure 3.7). The level of deformation is simply computed as a ratio of the diagonals. More formally, let A, B, C and D be the corner points of the training image and A', B', C' and D' be their projections according to the homography H. H is admitted only if

$$\frac{1}{th_{diag}} \leq \frac{\|A'C'\|}{\|B'D'\|} \leq th_{diag}. \tag{3.10}$$

Threshold th_{diag} was empirically set to 1,5.

Figure 3.8 Chirality. Training image (left), query image (middle left), warped training image according to the transformation found by OpenCV (middle right) and warped training image according to the transformation found after the chirality checking (right).

There is also another phenomenon known as chirality. It relates to the reconstruction of positions of cameras in epipolar geometry. There are situations when the equations for the fundamental matrix are satisfied but some points in the 3D space may be situated behind the camera while others are in front of it which obviously does not correspond to the real scene. As a result, projected image is cut at infinity (see Figure 3.8). The diagonal test as it is computed in the previous step does not detect this case because positions of the corner points may look fine. The solution of this problem is straightforward – scaling factors $\lambda_A, \lambda_B, \lambda_C$ and $\lambda_D$must all have the same sign which means they are either all in front of the camera or all behind it. The case when all projections are behind the camera is actually feasible here. It only means that the camera is oriented to the opposite direction and since we do not need neither the camera position nor its orientation, this case may be admitted here.

Keypoints projections are computed only if both of these conditions are satisfied. Because deformed and improbable homographies are skipped right away, the

computational time of RANSAC speeded up twice. A match m_i consisting of training point p_i^t and query point p_i^q is considered as an inlier if

$$\left\| p_i^q, \frac{H(p_i^t)}{\lambda_i} \right\| < th_{in} \tag{3.11}$$

where th_{in} is a threshold. The value of th_{in} affects the number of inlier points (see Figure 3.9) and thus the speed of the algorithm (see Equations 2.23 and 2.25). We can allow to have rather big th_{in} since the purpose of this step is to filter out spatially inconsistent matches and not to find the exact transformation between images unlike some other computer vision applications where the exact estimate is crucial.

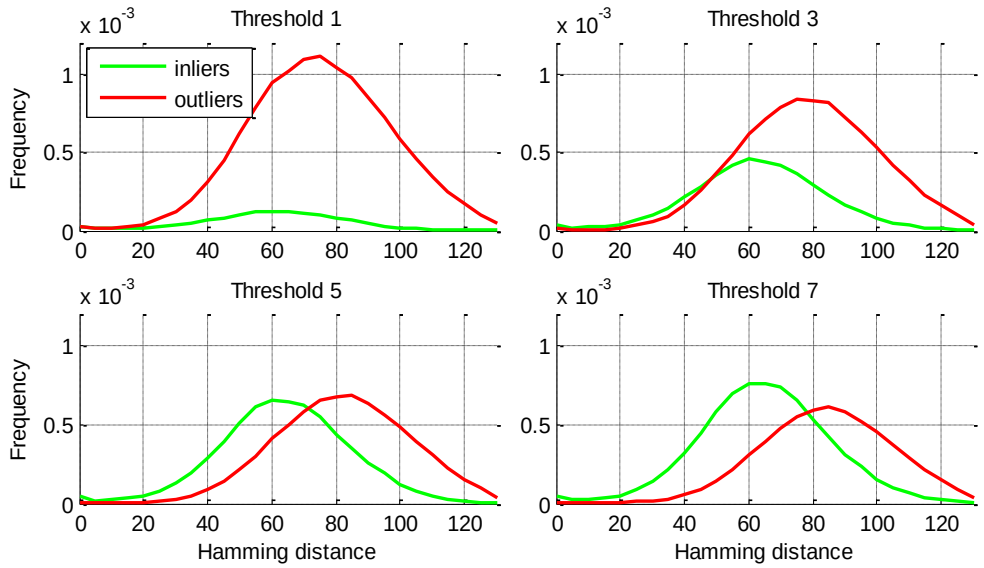


Figure 3.9 Histograms of Hamming distances of inlier and outlier points after using different RANSAC thresholds th_{in}. Outliers heavily outnumber inliers if th_{in} is too small and the constraints are too tight (top row). The smaller the Hamming distance is, the more likely a match is an inlier (bottom row). These plots also give an insight into how big reasonable values for th_{Hamm} should be.

3.5 Implementation

The application was implemented using the language C++ and open source libraries OpenCV 2.4.9 [51] and Boost 1.46.1 [52]. Multi-probe LSH part of the library was imported from Fast Library for Approximate Nearest Neighbours (FLANN) [53]. Additional scripts for data manipulation were implemented in the language Python 2.7. All source codes are available on the attached DVD.

4 Datasets

4.1 Flickr Dataset

FlickrLogos-32 [54] is a standard dataset for multi-class logo detection/recognition used in [21, 55]. It is not publically accessible but it is available after requesting the authors. It contains 70 instances for each of 32 logotypes as well as 6000 general non-logo images. All of these images were downloaded from the photo sharing service Flickr. The classes are Adidas, Aldi, Apple, Becks, BMW, Carlsberg, Chimay, Coca-Cola, Corona, DHL, Erdinger, Esso, Fedex, Ferrari, Ford, Foster's, Google, Guinness, Heineken, HP, Milka, Nvidia, Paulaner, Pepsi, Ritter Sport, Shell, Singha, Starbucks, Stella Artois, Texaco, Tsingtao and UPS. The non-logo images were downloaded from Flickr with the queries "building", "nature", "people" and "friends". Visual summary of different logotypes can be seen in Figure 4.1. Some images may contain several instances of the same logo. Images were manually annotated, so the locations of logos are exactly known (see Figure 4.2). The average size of an image is 0.71 MPx.

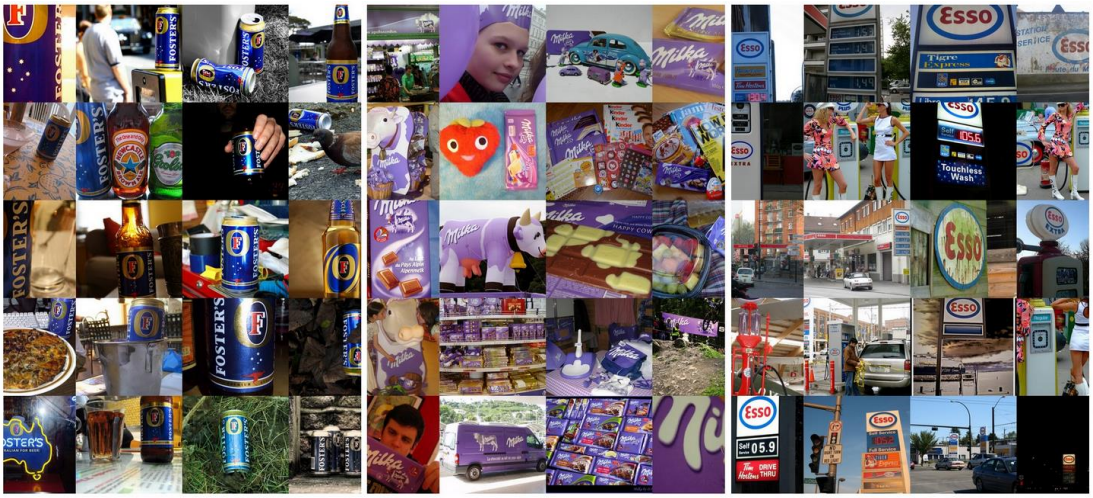


Figure 4.1 Flickr dataset. Examples of images of classes Foster, Milka and Esso.

Authors divided the dataset into three subsets. Training set contains 10 images of each logo. Validation and test set each contain 30 images of each logo and 3000 non-logo images.

We will denote the training set $\mathcal{R}_F$. 300 random non-logo pictures were randomly selected from the validation set. Union of these non-logo pictures and the set $\mathcal{R}_F$ is be denoted $\mathcal{Q}_F$. $\mathcal{R}_F$ is used as a training set and $\mathcal{Q}_F$ as a test set in all the experiments unless otherwise stated. Naturally, if a query $q \in \mathcal{Q}_F$ was also presented in $\mathcal{R}_F$, identical image $r \in \mathcal{R}_F$ was rejected from that particular search. Only the annotated areas were used for detection of training keypoints.

Figure 4.2 Pixel-level annotations. Original image (top left) and a mask of locations of logos (top right). Only keypoints which lie in the coloured areas are used for training (bottom).

4.2 Twitter Dataset

A huge set of tweets was kindly provided by the Czech company Wikidi[9]. These tweets were obtained by using the Twitter streaming API [56]. Thirty two streams, one for each class name from the Flickr dataset, ran from July 31st to September 24th. The tweets were obtained in the JSON text format and it contained information about the tweets, users' profiles and their settings. This is a lot of data for each tweet and only small portion is relevant for us. That is why the relevant attributes were saved to a tab-separated values (TSV) files. They contain the date, time, text and URL of tweets, username, corresponding keyword, followers count, the image URL and an indication whether the tweet was a retweet. These TSV files can be found on the attached DVD, only retweet entries had to be filtered out in order to fit the capacity of the disc.

The average image size is 0.48 MPx. Naturally, these images are not further labelled or annotated in any way.

[9] http://www.wikidi.cz/

4.3 Oxford Dataset

Oxford dataset is a standard dataset used by Mikolajczyk [57] and many others. It contains 8 scenes and 6 images for each of these scenes. The images have different zoom, level of blurriness, rotation, viewpoint, illumination conditions and JPEG compression so feature repeatability may be tested under different conditions. See Figure 4.3 for examples of this dataset.

Figure 4.3 Oxford dataset. Examples of scenes from the Oxford dataset are shown here.

5 Experiments

If not otherwise stated, all experiments were implemented using OpenCV 2.4.9 API on a machine with Linux Mint 17, Intel Core i7-4500U 1.8GHz, integrated Intel HD Graphics 4400, 8GB DDR3 RAM and SSD hard drive.

		Oxford dataset	Flickr dataset	Other literature
Morphological detectors	**FAST-9** (s.s.)	0.018	0.010	0.005 [11]
	FAST-9 (s.s., NMS)	0.023	0.013	0.006 [11]
	FAST-9 (m.s.)	0.097	0.064	0.005 [11]
	FAST-9 (m.s., NMS)	0.108	0.069	0.006 [11]
	ORB	0.109	0.081	0.042 [12]
	BRISK	0.187	0.148	0.034 [27]
	SUSAN			0.080 [10] 0.034 [11]
Gradient detectors	**DoG**			0.208 [11] 0.781 [26]
	SIFT	1.673	1.259	1.367 [42] 3.146 [27]
	SURF	2.128	1.781	0.234 [26] 0.211 [27]
	Harris			0.125 [10] 0.109 [11]
	Hessian			1.758 [42]
Affine detectors	**MSER**	0.893	0.732	1.289 [43]
	Hessian Affine			5.332 [43]
	Harris Affine			2.793 [43] 70.313 [42]
	Harris Affine Region			23.438 [42]

Table 5.2 Detector times. All times are normalized to s/MPx. Abbreviation s.s. denotes single scale, m.s. multi scale and NMS non-maxima suppression. Image dataset and Flickr dataset are further described in Chapter 4. Processing times cited from other sources may vary due to the different implementation, settings and/or processing power. Affine detectors are shown only for completeness even though they were never considered for this application due to their insufficient speed.

		Oxford dataset	Flickr dataset	Other literature
Binary descriptors	**ORB**	0.083	0.082	
	BRISK	0.027	0.026	0.031 [13] 0.021 [27]
	FREAK	0.088	0.024	0.018 [13]
Gradient detectors	**SIFT**	0.425	0.550	2.5 [13] 5.286 [27]
	SURF	0.798	0.826	1.4 [13] 0.389 [27]

Table 5.1 Descriptor times. All times are normalized to ms/keypoint. Image dataset and Flickr dataset are further described in Chapter 4 and as in the Table 5.2, processing times cited from other sources may vary due to the different implementation, settings and/or processing power.

First of all we tested the performance of available detectors (see Table 5.2) and descriptors (see Table 5.1). We selected ORB as our detector since it is the fastest one with scale and rotation invariance. FREAK was our choice of descriptor since it is also the fastest one and showed better repeatability than the other ones in [18].

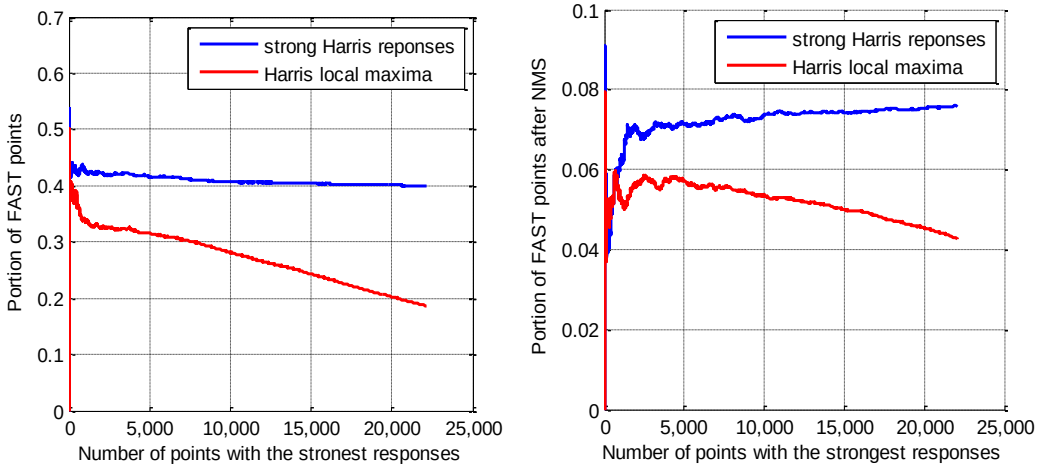


Figure 5.1 Relation of Harris and FAST detectors. Plot on the left shows how many keypoints with the strong Harris response satisfy the condition for FAST corners. Plot on the right shows how many of the Harris keypoints correspond to the FAST keypoints after the FAST non-maxima suppression. Both test were performed on images from the entire Oxford dataset.

ORB detector looks for FAST corners in the scale space and it uses Harris response to choose only the most stable keypoints. ORB detector is faster than Harris detector since more time consuming Harris response is computed only on the pre-filtered subset determined by FAST keypoints. FAST keypoints thus might be a good way to reduce computational time of Harris detector. Figure 5.1 shows how many of Harris keypoints also satisfy the condition for FAST corner. Interestingly, Harris local maxima satisfy this condition less frequently then regions with strong Harris response in general. This may be understood that FAST corners have wider regions of cornerness than Harris corners. Figure 5.1 also shows how much Harris response only little correlate with FAST response (Equation 2.7) which is used for non-maxima suppression of FAST keypoints.

Synthetic augmentation and verification of keypoints of the training images proved to remain only the keypoints with high repeatability. See Figure 5.2 for histograms of how many keypoints remained after using different parameters k_f. Interestingly, there were some images which lost four fifths of their keypoints even with $k_f = 1$. This setting, $k_f = 1$, was also used for the rest of the experiments since bigger values decreased the number of keypoints too drastically.

Multi-probe LSH is one of the key elements of the algorithm. The number of hash tables, the length of hash keys and the multi-probe level affect the time and recall of the retrieval. Figure 5.3 also shows that recall decreases for further matches. Multi-probe LSH with 15 hashing tables, 29-bit-long hashes and 1-step perturbation vectors is used for all

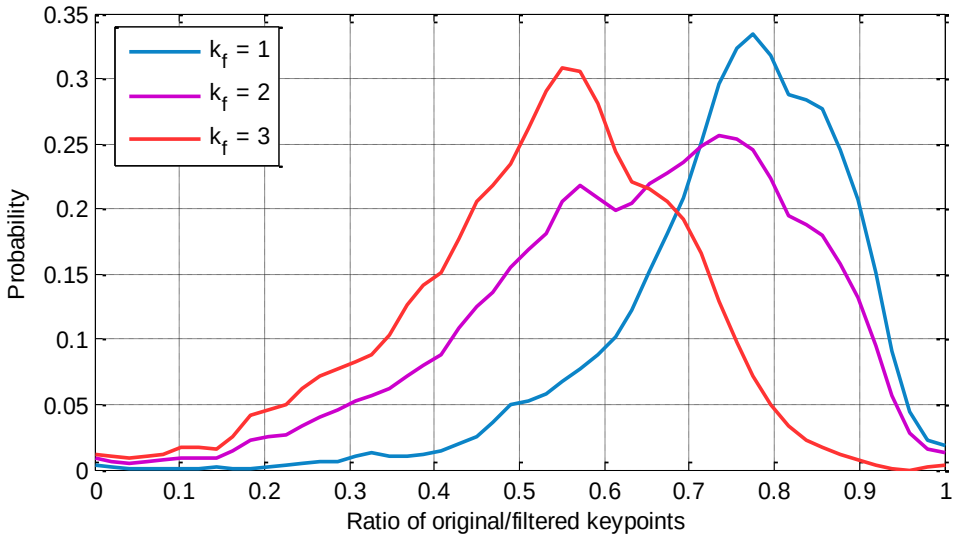


Figure 5.2 Histogram of what portion of keypoints remained after the synthetic augmentation. The synthetic augmentation verification with different k_f was used for all training images from the Flickr dataset (see Chapter 4.1). This chart shows three histograms of how many keypoints remained. The bigger k_f is, the more noticeable the filtering is.

further measurements as it was the fastest configuration with 60% recall for 5-NN and with 50% recall for 10-NN.

There have been introduced some improvements to speedup the retrieval process and also relatively many parameters. Different configurations were tested on the Flickr dataset and evaluated. While some of the configurations perform similarly, some perform distinctly worse at both retrieval time and recall. Performance of the best ones can be seen in Figure 5.4. There is visible trade-off between the computational time and recall. It depends on the particular application requirements which property is more preferred.

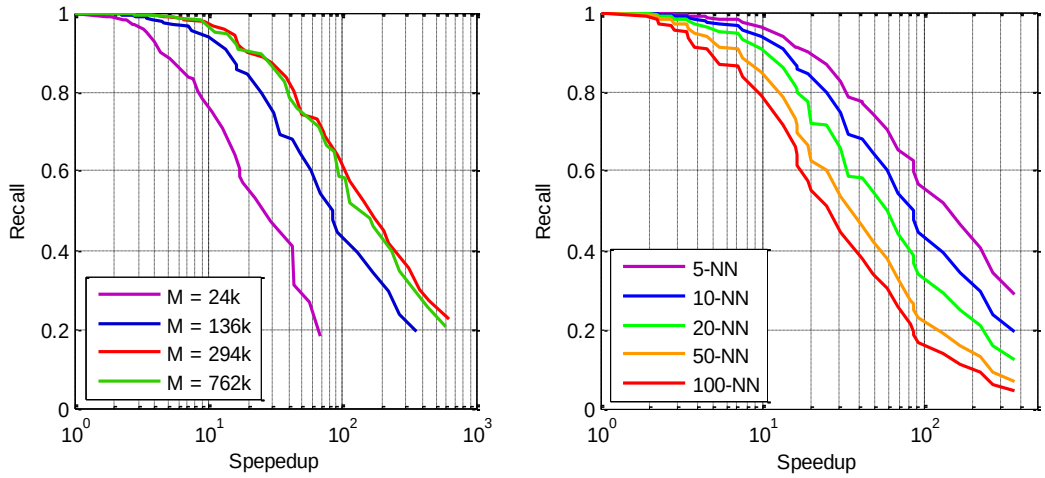


Figure 5.3 Multi-probe LSH performance. Multi-probe LSH proves to be orders of magnitude faster than the linear search, especially when size of the database M increases (left). It performs the best for the closest matches and the recall decreases when trying to retrieve further matches (right).

There is a sharp decrease of recall once first false positives appear. That makes it clear where to set the threshold th_{min} in order to separate the true and false positives. Its optimal value is different for each configuration though.

Total retrieval time needed for the test set $\mathcal{Q}$ varied from 194 to 439 seconds which is equivalent to 315 to 708 ms per image. Recall that these values are for a single thread computation and a laptop with low-power optimized processor. Retrieval time of the same tests but on a more powerful computer with Intel Xeon E5-2620 2.10 GHz CPU was exactly twice faster. See Figure 5.6 for a visualization of the time consumption of the particular steps of the algorithm.

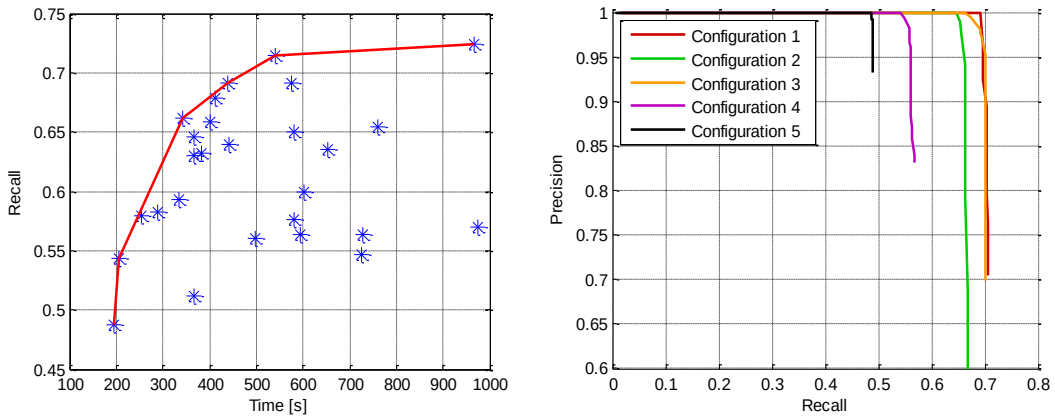


Figure 5.4 Performance on the Flickr dataset. Recall/time trade-off of different configurations (blue) is visualized on the left. The x-axis shows the total retrieval time for the set $\mathcal{Q}$. The red line connects the configurations which outperformed the others. On the right are shown precision/recall curves of such configurations. See Table 5.3 for further description of the configurations.

	Recall	**Time**			
		Keypoint acquisition	k-NN	Cross-check	RANSAC
Configuration 1	69.1	110.3	235.0	9.7	84.9
Configuration 2	66.2	109.1	155.9	8.1	70.3
Configuration 3	57.9	106.9	78.0	8.3	61.3
Configuration 4	54.3	104.3	71.2	2.3	28.3
Configuration 5	48.7	104.4	72.5	1.0	16.6

Table 5.3 Performance of different configurations. Configuration 1 uses all steps described in the Chapter 3 except any pre-filtering of training keypoints. It uses 64-byte long descriptors. Configuration 2 also does not do any keypoint pre-filtering and it uses 48-byte-long descriptor and strict stable matching when only the nearest neighbours are allowed (not the second nearest). Configuration 3 is the same as Configuration 1, but it only uses 32-byte-long descriptor. Configuration 4 is the same as Configuration 3, but it pre-filters training keypoints by estimation of the mutual transformations. And finally, Configuration 5 is the same as Configuration 1 except it performs both types of training keypoint pre-filtering.

The recall in Table 5.3 goes from 48.7% to 69.1%, but remember that we use different test set than the authors of bundle min-hashing. When we used Configuration 2 and the same datasets as in [21], we obtained 100% precision and 47.3% recall.

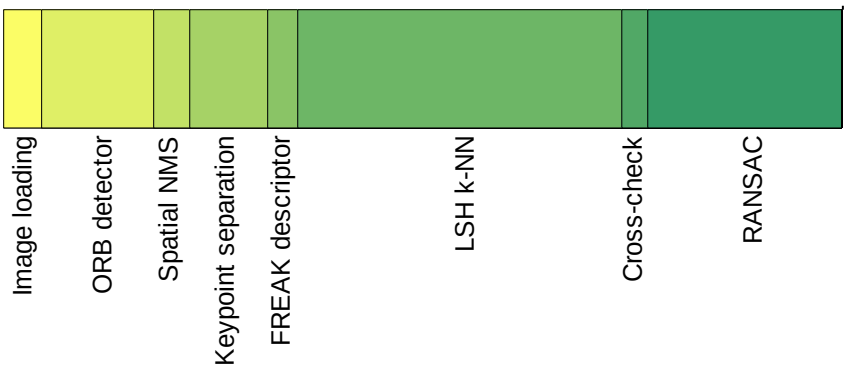


Figure 5.6 Distribution of time consumption. Configuration 2 was used. Multi-probe LSH and RANSAC are the two most time consumptive steps. Keypoint separation to dark and bright keypoints is also non-negligible even though it is only division of one set to two. It is because OpenCV does not provide polarity of keypoints and re-construction of Gaussian pyramid is needed for their estimation. This would be most likely corrected in the future work by changing the source code of OpenCV.

Even though overall recall is satisfactory for all the configurations, it is important to break down the results according to the individual logo classes. As can be seen in Figure 5.5, values of recall may differ a lot for different classes. Upon closer inspection we found out that some of the logotypes have worse retrieval rate because they have less local features. Insufficient number of local features makes matching less robust to the viewpoint and illumination changes among the images. This is partially caused by using

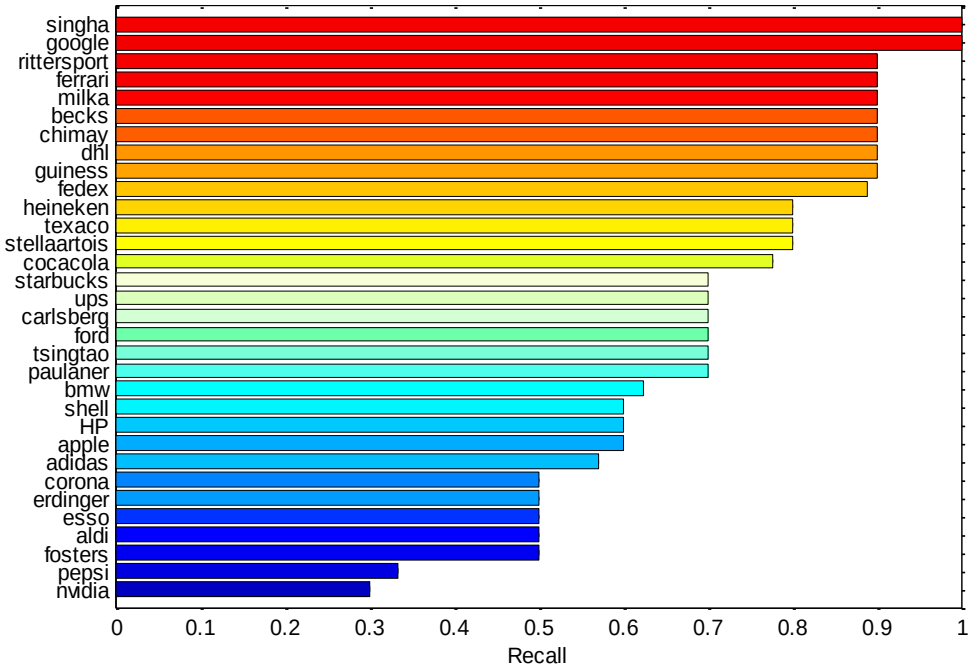


Figure 5.5 Recall of different brands. The variance of recall is large depending on the presented logo. Success rate of the retrieval is dependent on the number of keypoints. Classes which often have less keypoints are more prone to the imperfections in images.

the corner detector even for rounded logotypes such as Apple, Pepsi or BMW. Some logos appear in several different variations (Adidas, Nvidia) which also makes them harder to match.

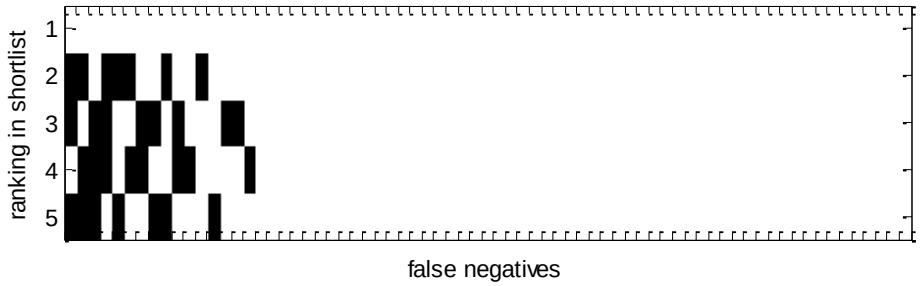


Figure 5.7 Analysis of false negatives. Each of the 71 columns corresponds to one false negative query image from the Flickr dataset. The rows indicate classes of first five images from the final shortlist after the spatial verification. The black fields indicate where query and training classes were the same.

We also analysed members of shortlists for all false negative cases to get a better idea what happens during re-ranking of the shortlist. As it can be seen in Figure 5.7, only small portion of ten to fifteen rejected images actually had chance to be retrieved. The other ones did not have any member of the true class in their shortlist so they could not be correctly match. This means that lower recall is not fault in spatial verification.

Twitter dataset was used for the final experiment (Figure 5.8). Images in this dataset are smaller than the ones in the Flickr dataset and logos usually occupy smaller area of the picture. That is why the result is noticeably worse. However, even recall about 20% is sufficient for making an analysis of trends in data.

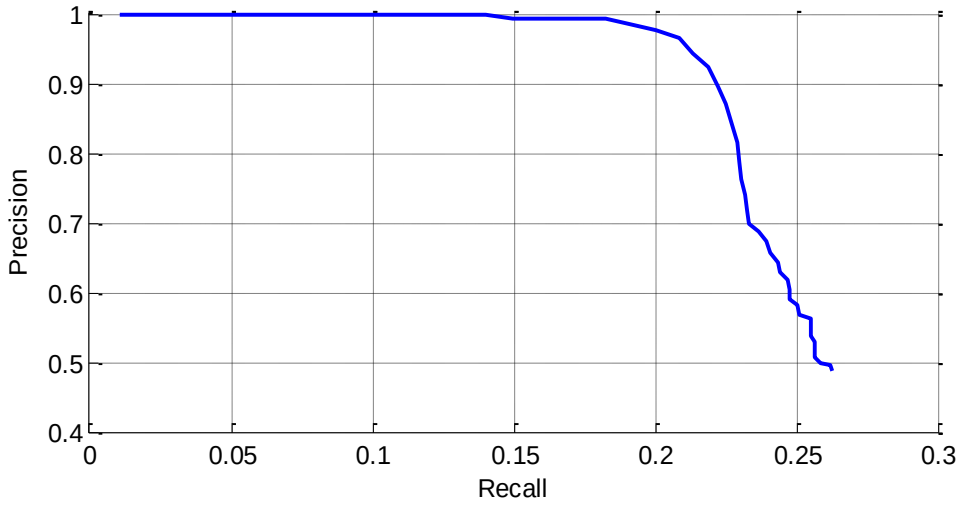


Figure 5.8 Performance on the Twitter dataset. Precision/recall curve (blue) was obtained using Configuration 2. As for the Flickr dataset, the curve steeply falls after the first false positives appear.

6 Conclusion

Rapid development of social media analysis is a recent trend. While natural language processing and analysis of interactions among users are widely used and developed fields, understanding and interpretation of shared images in the social media domain are still at their beginning. Due to the huge volume and variety of the images, it is certainly a challenging task which requires novel approaches.

A new end-to-end solution for the fast detection of objects of interest from an unknown view in rapidly changing large collections of data was proposed based on the reviewed state-of-the-art methods. In addition to the used state-of-the-art methods for detection, description and matching, we introduces several improvements – fast non-maxima suppression of keypoints, pre-filtering of training keypoints by synthetic augmentation and by mutual verification of images within the same class and improved RANSAC which counts support only for the relevant hypotheses.

The system is supposed to process millions of images daily, so a great emphasis was put on its speed. The retrieval time varies according to resolution and complexity of the image. The average retrieval time was less than one second for the Flickr dataset on a low-power processor. Using more powerful server processor exactly doubled the speed. Images downloaded from Twitter are typically smaller and less complex. That is why the retrieval time of images from Twitter about twice faster. This means that a single server machine with multiple processing cores would be sufficient for Twitter monitoring if multithread support was implemented.

We are able to achieve both high precision and satisfactory retrieval time but our recall 47% on the test set from the Flickr dataset is noticeably worse than 83% achieved by bundle min-hashing. Differences are in every step of the process so it is hard to tell which part causes this decrease of recall. The bundle min-hashing paper does not mention the overall retrieval time so it is possible that its retrieval time would not be sufficient for the given requirements.

Our other contribution is that we obtained a sample of 5 million tweets with images from Twitter. It take up 400 GB of storage nevertheless we provide text files with URL of the images on the attached DVD so it can be used for future research. We achieved 99% precision and 18.3% recall which should still be sufficient for monitoring of trends how much people share images with given logos.

We are aware of limitations of the algorithm. Rounded or very simple logos are not suitable for our approach because of the lack of detected keypoints. Repetitive structures within a logo may results in incorrect matching of the keypoints. We also did not take care of special cases when there are multiple logos in one image.

For the future work, we would suggest modifying the source code of OpenCV for efficient calculation of polarity of FAST keypoints and parallelization of the algorithm which has not been implemented yet and which would be necessary for the real life application. Further research can be done on the approximate nearest neighbour

algorithm as well as on the use of colour which was not beneficial for us. And finally, detection of multiple classes in one picture would be a useful feature for the marketers.

7 References

[1] S. KEMP, "Social, Digital & Mobile Worldwide in 2014," We Are Social, 9 January 2014. [Online]. Available: http://wearesocial.net/blog/2014/01/social-digital-mobile-worldwide-2014/. [Accessed 9 July 2014].

[2] "Company Info," Facebook, 31 March 2014. [Online]. Available: http://newsroom.fb.com/company-info/. [Accessed 9 July 2014].

[3] "A Focus on Efficiency," Internet.org, 16 September 2013. [Online]. Available: http://internet.org/efficiencypaper. [Accessed 9 July 2014].

[4] E. HAMBURGER, "Real talk: the new Snapchat brilliantly mixes video and texting," Teh Verge, 1 May 2014. [Online]. Available: http://www.theverge.com/2014/5/1/5670260/real-talk-the-new-snapchat-makes-texting-fun-again-video-calls. [Accessed 9 July 2014].

[5] "Press Page - Instagram," Instagram, [Online]. Available: http://instagram.com/press/. [Accessed 16 July 2014].

[6] E. PACHECO and R. UDOWITZ, "U.S. Social Media Advertising Revenues to Reach $15B by 2018," BIA/Kelsey, 5 May 2014. [Online]. Available: http://www.biakelsey.com/Company/Press-Releases/140515-U.S.-Social-Media-Advertising-Revenues-to-Reach-$15B-by-2018.asp. [Accessed 9 July 2014].

[7] "gazeMetrix.com | Locate and manage your brand in social photos," Gaze Metrix, [Online]. Available: https://www.gazemetrix.com/. [Accessed 16 July 2014].

[8] "Phashtag," Phashtag, [Online]. Available: http://phashtag.com/Logos. [Accessed 16 July 2014].

[9] "Content Tracking « LTU – The Image Recognition API," LTU Technologies, [Online]. Available: http://www.ltutech.com/solutions/visual-content-tracking/. [Accessed 16 July 2014].

[10] "Ditto | AngelList," Angel.co, [Online]. Available: https://angel.co/ditto. [Accessed 16 July 2014].

[11] "Ditto Labs, Inc.," Ditto Labs, Inc., [Online]. Available: http://ditto.us.com/. [Accessed 16 July 2014].

[12] "About Twitter, Inc.," Twitter, [Online]. Available: https://about.twitter.com/company. [Accessed 9 July 2014].

[13] Y. LIU, C. KLIMAN-SILVER and A. MISLOVE, "The tweets they are a-changin': Evolution of Twitter users and behavior," in *Proceedings of the 8th International AAAI Conference on Weblogs and Social Media (ICWSM'14)*, Ann Arbor, MI, 2014.

[14] B. B. COOPER, "How Twitter's Expanded Images Increase Clicks, Retweets and Favorites [New Data]," Buffer, 13 Novemer 2013. [Online]. Available: http://blog.bufferapp.com/the-power-of-twitters-new-expanded-images-and-how-to-make-the-most-of-it. [Accessed 9 July 2014].

[15] . E. ROSTEN and T. DRUMMOND, "Machine learning for high speed corner detection," in *European Conference on Computer Vision*, vol. 1, 2006, pp. 430-443.

[16] E. ROSTEN, R. PORTER and T. DRUMMOND, "Faster and better: a machine learning approach to corner detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 32, pp. 105-119, 2010.

[17] E. RUBLEE, V. RABAUD, K. KONOLIGE and G. BRADSKI, "ORB: An efficient alternative to SIFT or SURF," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2011, pp. 2564-2571.

[18] A. ALAHI, R. ORTIZ and P. VANDERGHEYNST, "FREAK: Fast Retina Keypoint," in *IEEE Conference on Computer Vision and Pattern Recognition*, Providence , Rhode Island: IEEE , 2012.

[19] Q. LV, W. JOSEPHSON, Z. WANG, C. Moses and K. LI, "Multi-Probe LSH: Efficient Indexing for High-Dimensional Similarity Search," in *Proceedings of the 33rd International Conference on Very Large Data Bases*, Vienna, VLDB Endowment, 2007, pp. 950-961.

[20] H. JÉGOU, M. DOUZE, C. SCHMID and P. PÉREZ, "Aggregating local descriptors into a compact image representation," in *IEEE Conference on Computer Vision & Pattern Recognition*, 2010, pp. 3304-3311.

[21] S. ROMBERG and R. LIENHART, "Bundle Min-hashing for Logo Recognition," in *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, New York, NY, USA, ACM, 2013, pp. 113-120.

[22] W. FÖRSTNER, "A feature based correspondence algorithm for image matching," *International Archives of Photogrammetry and Remote Sensing,* vol. 3, pp. 150-166, 19 August 1986.

[23] C. HARRIS and M. STEPHENS, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147-151.

[24] K. MIKOLAJCZYK and C. SCHMID, "Indexing based on scale invariant interest points," in *Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, 2001, pp. 525-531.

[25] T. LINDERBERG, "Scale-space theory: A basic tool for analysing structures at different scales," *Journal of Applied Statistics,* vol. 21, pp. 225-270, 1994.

[26] S. M. SMITH and J. M. BRADY, "SUSAN—A New Approach to Low Level Image Processing," *Int. J. Comput. Vision,* vol. 23, pp. 45-78, May 1997.

[27] J. R. QUINLAN, "Induction of Decision Trees," *Machine Learning,* vol. 1, pp. 81-106, 1986.

[28] T. TUYTELAARS and K. MIKOLAJCZYK, "Local Invariant Feature Detectors: A Survey," *Found. Trends. Comput. Graph. Vis.,* vol. 3, pp. 177-280, January 2008.

[29] M. CALONDER, V. LEPETIT, C. STRECHA and P. FUA, "BRIEF: Binary Robust Independent Elementary Features," in *11th European Conference on Computer Vision (ECCV)*, Heraklion, 2010.

[30] P. ROSIN, "Measuring Corner Properties," *Computer Vision and Image Understanding,* vol. 73, pp. 291-307, February 1999.

[31] D. G. LOWE, "Object recognition from local scale-invariant features," *The Proceedings of the Seventh IEEE International Conference on Computer Vision,* vol. 2, pp. 1150-1157, September 1999.

[32] D. G. LOWE, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vision,* vol. 60, pp. 91-110, November 2004.

[33] H. BAY, A. ESS, T. TUYTELAARS and L. VAN GOOL, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding (CVIU),* vol. 110, no. 3, pp. 346--359, 2008.

[34] S. LEUTENEGGER, M. CHLI and R. SIEGWART, "BRISK: Binary Robust Invariant Scalable Keypoints," *IEEE International Conference on Computer Vision,* pp. 2548-2555, 2011.

[35] J. SIVIC and A. ZISSERMAN, "Video Google: Efficient Visual Search of Videos," in *Toward Category-Level Object Recognition*, vol. 4170, J. Ponce, M. Hebert, C. Schmid and A. Zisserman, Eds., Springer Berlin Heidelberg, 2006, pp. 127-144.

[36] A. TORII, J. SIVIC, T. PAJDLA and M. OKUTOMI, "Visual Place Recognition with Repetitive Structures," in *CVPR*, 2013.

[37] D. NISTÉR and H. STWEÉNIUS, "Scalable Recognition with a Vocabulary Tree," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2006, pp. 2161-2168.

[38] G. SCHINDLER, M. BROWN and R. SZELISKI, "City-Scale Location Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, Minneapolis, MN, IEEE, 2007, pp. 1 - 7.

[39] I. PIOTR and M. RAJEEV, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, Dallas, Texas, USA, ACM, 1998, pp. 604-613.

[40] R. PANIGRAHY, "Entropy based nearest neighbor search," in *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, Miami,Florida, USA, 2006, pp. 1186-1195.

[41] C. SCHMID and R. MOHR, "Local Grayvalue Invariants for Image Retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 19, pp. 530-535, May 1997.

[42] R. DUDA and P. HART, "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Commun. ACM,* vol. 15, no. 1, pp. 11-15, 1972.

[43] D. BALLARD, "Generalizing the Hough transform to detect arbitrary shapes," in *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 1987, pp. 714-725.

[44] M. A. FISCHLER and R. C. BOLLES, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM,* vol. 24, no. 6, pp. 381-395, 1981.

[45] O. CHUM, J. MATAS and J. KITTLER, "Locally optimized RANSAC," in *Pattern Recognition*, 2003, pp. 236-243.

[46] O. CHUM, T. WERNER and J. MATAS, "Two-View Geometry Estimation Unaffected by a Dominant Plane," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 772-779.

[47] O. CHUM, MATAS and Jiří, "Matching with PROSAC - progressive sample consensus," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 220-226.

[48] O. CHUM and J. MATAS, "Optimal Randomized RANSAC," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, no. 8, pp. 1472-1482, June 2008.

[49] M. BROWN, R. SZELISKI and S. WINDER, "Multi-Image Matching Using Multi-Scale Oriented Patches," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, Washington, DC, USA, IEEE Computer Society, 2005, pp. 510-517.

[50] K. LEVENBERG, "A Method for the Solution of Certain Non-Linear Problems in Least," *The Quarterly of Applied Mathematics,* vol. 2, pp. 164-168, 1944.

[51] "OpenCV," Itseez, 2014. [Online]. Available: http://opencv.org/. [Accessed 9 July 2014].

[52] "Boost C++ Libraries," [Online]. Available: http://www.boost.org/. [Accessed 16 July 2014].

[53] "FLANN - Fast Library for Approximate Nearest Neighbors," The University of British Columbia, [Online]. Available: http://www.cs.ubc.ca/research/flann/. [Accessed 16 July 2014].

[54] "FlickrLogos-32 Dataset," Augsburg University, [Online]. Available: http://www.multimedia-computing.de/flickrlogos/. [Accessed 16 July 2014].

[55] S. ROMBERG, R. LIENHART and L. G. PUEYO, "Scalable Logo Recognition in Real-World Images," in *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, New York, NY, USA, ACM, 2011, pp. 25-25.

[56] "The Streaming APIs | Twitter Developers," Twitter, Inc., [Online]. Available: https://dev.twitter.com/streaming/overview. [Accessed 16 July 2014].

[57] K. MIKOLAJCZYK, "Krystian Mikolajczyk Homepage," 2004. [Online]. Available: https://lear.inrialpes.fr/people/mikolajczyk/. [Accessed 28 August 2014].

[58] J. ŠLERKA, "#boston aneb Cesta tam a zase zpátky," YouTube, 6 May 2013. [Online]. Available: http://youtu.be/6lXJA5vQx9c. [Accessed 9 July 2014].

[59] H. MORAVEC, "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," in *tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University & doctoral dissertation, Stanford University*, 1980.

[60] S. LEWIS, "One Second on the Internet," [Online]. Available: http://onesecond.designly.com/. [Accessed 9 July 2014].

[61] KRIKORIAN, Raffi, "New Tweets per second record, and how!," Twitter, 16 August 2013. [Online]. Available: https://blog.twitter.com/2013/new-tweets-per-second-record-and-how. [Accessed 9 July 2014].

[62] "Cambridge Michaelmas Term – Week 0," 12 October 2009. [Online]. Available: http://technofutures.wordpress.com/2008/10/12/cambridge-michaelmas-term-week-0/. [Accessed 28 August 2014].

[63] G. LEVI, "Tutorial on Binary Descriptors – part 1," 26 August 2013. [Online]. Available: http://gilscvblog.wordpress.com/2013/08/26/tutorial-on-binary-descriptors-part-1/. [Accessed 28 August 2014].

[64] M. MUJA and L. David, "FLANN - Fast Library for Approximate Nearest Neighbors," [Online]. Available: http://www.cs.ubc.ca/research/flann/. [Accessed 28 August 2014].

[65] K. MIKOLAJCZYK and C. SCHMID, "Scale & Affine Invariant Interest Point Detectors," *International Journal of Computer Vision* , vol. 60, no. 1, pp. 63-86, 1 October 2004.

[66] K. MIKOLAJCZYK and e. al., "A Comparison of Affine Region Detectors," *International Journal of Computer Vision,* vol. 65, no. 1-2, pp. 43-72, November 2005.

[67] O. CHUM and J. MATAS, "Large Scale Image Retrieval," 8 March 2010. [Online]. Available: https://cw.felk.cvut.cz/wiki/_media/courses/a4m33mpv/2010.03.08_large-cale-image-retrieval.pdf. [Accessed 28 August 2014].

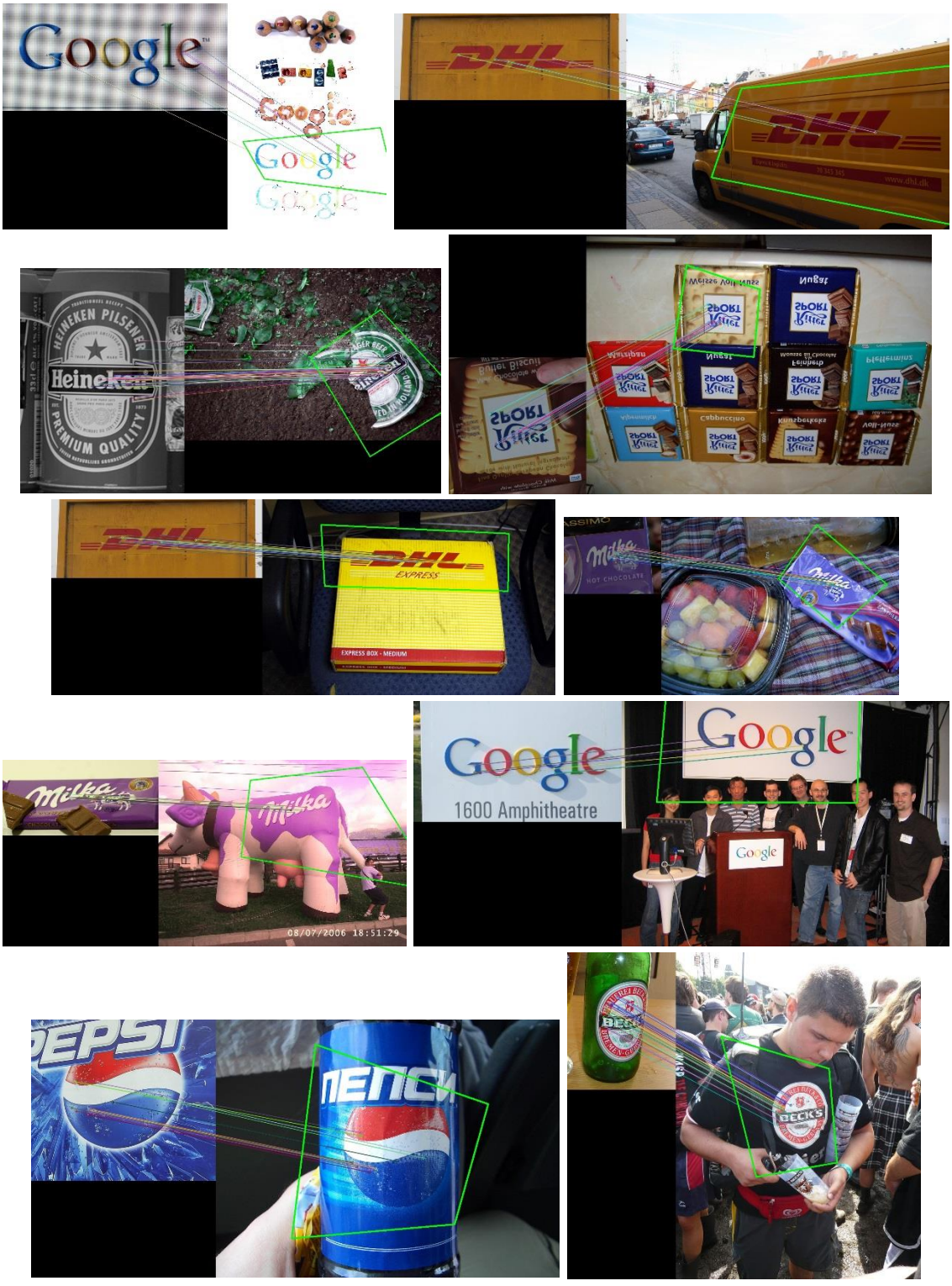
[68] J. MATAS and O. CHUM, "State-of-the-art RANSAC," 2011. [Online]. Available: http://www.imgfsr.com/CVPR2011/Tutorial6/RANSAC_CVPR2011.pdf. [Accessed 28 August 2014].

[69] J. MATAS, "Hough Transform," 10 May 2010. [Online]. Available: https://cw.felk.cvut.cz/wiki/_media/courses/a4m33mpv/2010.05.10_hough-transform.pdf. [Accessed 28 August 2014].

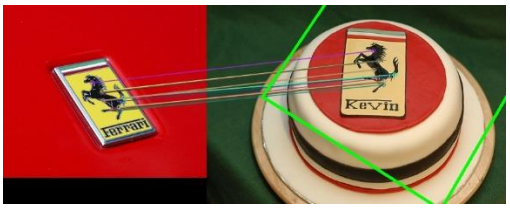
[70] D. FROLOVA and D. SIMAKOV, "Matching with Invariant Features," Weizmann Inst. of Science, Computer Vision Lab, March 2004. [Online]. Available: http://www.wisdom.weizmann.ac.il/~daryaf/InvariantFeatures.ppt. [Accessed 16 July 2014].

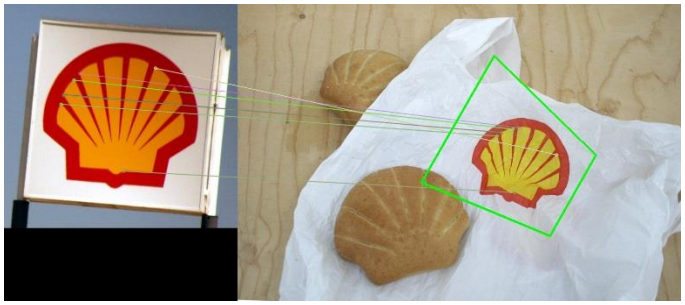
A. Examples of results

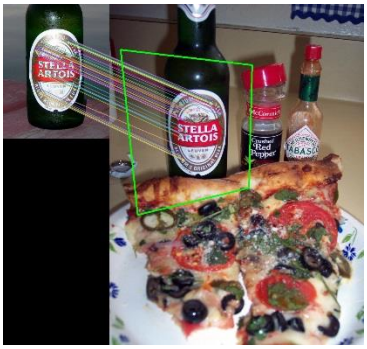
Images on the right are the query images from the Flickr dataset. On the left are the best matching images from the database. The green boxes show the spatial transformation of the training image and the multi-coloured lines show spatially verified correspondences.




INTERNATIONAL
HOUSE OF MUSIC


Ferrari
Kevin




STELLA ARTOIS


STELLA ARTOIS


PREMIUM LAGER BEER
STELLA ARTOIS
LEUVEN
BELGIUM'S ORIGINAL BEER
STELLA ARTOIS


FedEx
DHL
EXPRESS
FedEx
Federal Express


adidas
ORIGINALS
adidas
STAR WARS


GUINNESS
GUINNESS

B. Content of the attached DVD

/

- src/ # the source code of the algorithm
- thesis/ # this thesis in pdf file
- twitter dataset/ # dataset of 5M tweets
  - data/ # tsv files with the tweets
  - src/ # scripts for downloading the images
- utils/ # additional scripts
  - annotations creator/ # tool for marking the positions of logos
  - pics2tweets/ # tool which matches image results back to the coresponding tweets