Czech Technical University in Prague

Faculty of Electrical Engineering

Doctoral Thesis

Month and year:
February 2017

Name of candidate:
Petr Gronát

Place Recognition by Per-Location Classifiers

Doctoral Thesis

Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Cybernetics

Petr Gronát

Prague, January 2017

Supervisor: Tomáš Pajdla
Supervisor-specialist: Josef Sivic

Study Program No.: P–2612–Electrotechnics and Informatics
Branch No.: 3902V035–Artificial Intelligence and Biocybernetics

“To Maruška, Rufka and Filip”

Petr Gronát Doctoral Thesis
February 2017

Place Recognition by Per-Location Classifiers

Abstract

Place recognition is formulated as a task of finding the location where the query image was captured. This is an important task that has many practical applications in robotics, autonomous driving, augmented reality, 3D reconstruction or systems that organize imagery in geographically structured manner. Place recognition is typically done by finding a reference image in a large structured geo-referenced database.

In this work, we first address the problem of building a geo-referenced dataset for place recognition. We describe a framework for building the dataset from the street-side imagery of the Google Street View that provides panoramic views from positions along many streets, cities and rural areas worldwide. Besides of downloading the panoramic views and ability to transform them into a set of perspective images, the framework is capable of getting underlying scene depth information.

Second, we aim at localizing a query photograph by finding other images depicting the same place in a large geotagged image database. This is a challenging task due to changes in viewpoint, imaging conditions and the large size of the image database. The contribution of this work is two-fold; (i) we cast the place recognition problem as a classification task and use the available geotags to train a classifier for each location in the database in a similar manner to per-exemplar SVMs in object recognition, and (ii) as only a few positive training examples are available for each location, we propose two methods to calibrate all the per-location SVM classifiers without the need for additional positive training data. The first method relies on p-values from statistical hypothesis testing and uses only the available negative training data. The second method performs an affine calibration by appropriately normalizing the learned classifier hyperplane and does not need any additional labeled training data. We test the proposed place recognition method with the bag-of-visual-words and Fisher vector image representations suitable for large scale indexing.

Experiments are performed on three datasets: $25,000$ and $55,000$ geotagged street view images of Pittsburgh, and the 24/7 Tokyo benchmark containing $76,000$ images with varying illumination conditions. The results show improved place recognition accuracy of the learned image representation over direct matching of raw image descriptors.

Abstrakt

Pojem rozpoznávání místa je formulován jako úloha nalezení místa, kde byl pořízen dotazovaný obraz. Tato významná úloha má praktické aplikace v robotice, autonomním řízení, rozšířené realitě, 3D rekonstrukci či systémech, které oraganizují obrazová data geograficky strukturovaným způsobem. Rozpoznávání místa se obvykle provádí nalezením referenčního obrazu ve velké strukturované georeferenční databázi.

Tato práce se nejprve zabývá tvorbou georeferenční databáze pro rozpoznávání místa. Popisuje způsob stavby databáze z Google Street View snímků, které poskytují panoramatické pohledy zachycené v mnoha ulicích, městech a venkovských oblastech po celém světě. Kromě stahování panoramat je popsán způsob generovaní perspektivních snímků a získávání hloubkových map zachycené scény.

Dále tato práce cílí na lokalizaci dotazovaného obrazu hledáním dalších obrázků v georeferenční databázi zachycujících stejné místo. Jedná se o nelehkou úlohu, kde je třeba se vypořádát se změnami polohy kamery, světelnými podmínkami a velikostí databáze. Přínos této práce je dvojí. (i) formulace problému rozpoznávání místa jako úlohy klasifikační a za použití geotagů natrénování klasifikátorů pro každou lokaci v databázi podobně jako per-exemplar SVM v rozpoznávání objektů. (ii) protože pro každou lokaci je dustupných pouze několik pozitivních trénovacích příkladů, byly navrhnuty dvě kalibrační metody pro per-exemplar SVM, které nepotřebují pozitivní trénovací data. První metoda je založená na p-values a používá pouze nagativní trénovací data. Druhá metoda je založena na afinní kalibraci pomocí příslušné normalizace normálového vektoru naučené nadroviny. Navrhovaná metoda rozpoznávání místa je testována na bag-of-words a Fisher vector obrazových reprezentacích vhodných pro indexování velkých databází.

Experimenty jsou provedeny na třech datasetech: geotagované obrázky Google Street View z města Pittsburgh o velikostech 25000 a 55000 snímků a datasetu 24/7 Tokyo, který obsahuje 76000 obrázků s výraznými rozdíly ve světelných podmínkách. Výsledky vykazují výrazně lepší přesnost rozpoznání místa za použití reprezentací na základě naučených klasifikátorů.

This thesis is submitted to the Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Petr Gronát

Acknowledgements

The real spirit of achieving a goal is through the way of excellence and perpetual discipline. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various personalities.

I owe my deepest gratitude to my outstanding Ph.D. advisors, Josef Sivic (INRIA) and Tomáš Pajdla (CMP), for their patience, support and keeping my Ph.D. on a good track. They always created a friendly atmosphere, shared their visions and kept me focused. Also, I would like to thank my collaborator Relja Arandjelovic for sharing his thoughts, coffee, and drinks with me. I am deeply grateful to Alyosha Efros who gave me an opportunity to briefly work with his group at UC Berkeley during my internship.

Special thanks belong to Andrej Mikulik (CMP) who motivated me a lot at my Ph.D. journey with his optimism. Thanks to him, I discovered that nearly anything is possible to achieve with passion, patience, hard work and having great people around. Many thanks belong to my colleagues at INRIA who I had a pleasure to meet and to work with, namely Toby Dylan Hocking, Visesh Charii, Maxime Oquab, Vadim Kantorov, Vincent Delaitre, Mathiew Aubry, Guillaume Obozinski, Augustin Lefevre, Ivan Laptev, Anastasios Giovanidis, Guillaume Seguin, and other great colleagues in both INRIA and CMP. Finally, I wish to thank my wife, my kid and my family for immense support and patience.

Contents

List of Figures

List of Tables

1 Introduction

"All things are difficult before they are easy."
— Dr. Thomas Fuller

The Internet contains a vast collection of imagery that grows every moment. In 2013 the social network Facebook claimed [31] that every single minute users upload about 208,300 new images, for Instagram, this number was about 27,800. Thus, at the time of writing this thesis, there are about 340,000,000 new images uploaded per day. Beside of images, Youtube users upload more than 100 hours of video every minute. A considerable portion of these images and videos does not contain a geo-location information. Hence, a substantial amount of attention has been paid to developing techniques to pinpoint the location of the images or videos, commonly referred to as visual place recognition or geo-localization.

1.1 Objective

The goal of the thesis is twofold; We aim at building a geo-referenced place recognition dataset from imagery available on the Internet, and, second, we wish to develop a new method for place recognition in an outdoor urban environment.

While several place recognition datasets have been released in the past, these datasets often suffer from noisy ground truth labels, do not cover entire city uniformly, are biased towards landmarks, or suffer from the pure image quality. To cope with these drawbacks it is necessary to build a dataset that covers street side imagery uniformly across the urban environment, a dataset that has reasonably accurate geotags, image resolution and the image quality. The dataset should cover the same scene captured at a different time.

Using the dataset, the goal of the place recognition is to localize an unknown query image by finding similar images from the database depicting the same place and retrieving a shortlist of these images. The location of the query image is then performed by assigning a geotag of the most similar image from the shortlist. The images within the shortlist can be re-ranked by a more expensive method such as geometric verification. Hence the primary challenge of the place recognition lies in building a small shortlist containing at least one database image capturing the same place as the query image. Notice that we do not aim at camera pose estimation. To design a method amenable of building the shortlist, one has to design a method that is able to deal with the camera viewpoint changes, illumination changes, confusing objects (the objects such as traffic signs, zebra crossings or road marks appearing at the multiple places).

The objective of this thesis is to create a system that builds a geo-referenced dataset from data available on the Internet and to design a system capable of localizing an unknown image in a large urban area of the scale of a city. First, We develop a method that can fetch high-resolution Google StreetView panoramas with associated metadata including depth-maps and automatically

generate databases for place recognition. Finally, the thesis develops a place recognition method that exploits the structured manner of the database, casts the place recognition problem as a classification task and learns a calibrated per-location classifier for each location in the database. This is illustrated in figure 1.2 where the heat map layer indicates distribution of the calibrated classification score for the given query image shown in blue. The four top-ranked database images are shown in green (correct location) and red (incorrect location).

1.2 Motivation

The image geo-location plays an important role in several fundamental tasks in computer vision, robotics, augmented reality, navigation and photogrammetry, the ancestor of computer vision. Knowing where the image or video was captured is essential in the systems that organize and analyze imagery by its geographical location. For instance, Geographic Information System (e.g. QGIS, ArcGIS) registers aerial and satellite imagery with the maps, robot navigation systems perform ego-localization based on image content captured by its camera sensors. Regarding the navigation, an accurate indoor localization is often performed by analyzing an image or a short video sequence captured by a user. One of the most popular online photo repositories Panoramio organizes its content by geo-location. Panoramio [23] is a geolocation-oriented photo sharing mashup that and presents its content to a user in geographically structured manner. It can be accessed as a layer in the Google Earth [22]. It allows Google Earth users to learn more about a given area by viewing the photos that other users have taken at that place. Geo-localizing the photographs can also be used for city reconstruction.

A substantial amount of attention has been paid to developing techniques to pinpoint the location of the images or videos. At the beginning of the century, most systems targeted at satellite and aerial imagery where geo-referenced datasets consist of images captured from an overhead view. Query images were mostly captured by aircraft or satellites hence methods developed in that end in view predominantly stand on registration of a planar scene.

The availability of visual data has shifted dramatically in the last decade due to emerging new data storage technologies, distributed systems, mobile devices and social networks. A considerable portion of the visual data is formed by street view imagery. Big companies put an effort in collecting large geo-referenced databases. For instance, the Google Street View, provides panoramic views from positions along many streets, cities and rural areas worldwide. It is worth noting that geo-referenced datasets play a fundamental role in emerging techniques for autonomous driving. Private companies such as HERE, Zoox, Uber and Tesla Motors are currently collecting their own street side imagery and combining it with data from other sensors such as inertial measurement unit, LiDAR, GPS or GLONASS. Petabytes of data serve as a fundamental source of information in designing systems amenable to localize and navigate an autonomous vehicle in complex urban environments.

1.3 Challenges

How can we recognize the same street-corner in the entire city or on the scale of the entire country despite the fact it can be captured in different illuminations or change its appearance over time? The fundamental scientific question is also what is the appropriate representation of a place that is rich enough to distinguish similarly looking places yet compact to, represent entire cities or

Figure 1.1: **Challenges in place recognition.** The place recognition system must achieve robustness to the large variability in a scene such as time of the day, weather changes, seasonal effects, repetitive elements such as traffic signs, pedestrians, and cars, occlusions or changes in the camera viewpoint. Some of these challenges are illustrated above. The three images capture the same place in San Francisco captured at a different time of the year.

countries.

The goal of place recognition is to find a reference database image that depicts the same place (scene) as is captured in the unknown query image. The query image is then assigned a geo-location of the reference image, as illustrated in figure 1.2. One strategy to apply is image matching or instance retrieval based methods, which are, in nature, similar to the place recognition regarding the goal and challenges to deal with.

For instance, the query and database images may depict the same 3D object (e.g. building) from a different camera viewpoint, under different illumination conditions (day, evening, sunny, cloudy) or the object can be partially occluded by nonstatic objects such as cars or pedestrians. The structured geo-referenced database may contain from few thousands to a couple of millions of images. Driven by prominent advances in the large-scale image retrieval of the Internet imagery, the image matching based approach was adopted to the place recognition.

While place recognition and image retrieval are similar regarding the goals and difficulties, it differs from the place recognition in several angles. In image matching, the goal is to find many images similar to the query image. However in the place recognition, rather than retrieving many similar images, the goal is to find a reference image (or a handful of reference images) that depicts exactly the same location shown in the query image. While image retrieval databases are typically unstructured collections of images, place recognition databases are usually structured: images have geotags, are localized on a map and depict a consistent 3D world. Knowing the structure of the database can be leveraged, and leads to significant improvements in both speed and accuracy of the place recognition.

A major challenge for visual place recognition system is to achieve robustness to the large variability in scene appearance that can be observed in the real world. Such changes (induced by the time of day, weather or seasonal effects as well as human activity) are a ubiquitous challenge for all place recognition systems. Another challenging problem to deal with are the confusing or repetitive elements that are present in urban environments (traffic signs, road marks, cars, windows, vegetation). The system should be able to recognize the same place captured from different camera viewpoints, under very different illumination conditions and, very often, captured in a different season. Some of these challenges are illustrated in figure 1.1.

1.4 Contributions

This thesis has three main contributions listed below.

First, we propose a method for building geo-referenced place recognition datasets from images available on the Internet and offer it to the community of researchers in the form of software package [25] named `streetget`. The proposed method (chapter 3) is capable of downloading recent and historical spherical panoramas, and its associated meta-data including approximate depth-maps. The spherical panoramas can be used to generate perspective cameras with known location and intrinsic parameters. In our recent work of Aranjelovic et al. [2], it was shown that using the historical spherical panoramas was crucial to train the convolutional neural network for place recognition as, without using it, the network does not generalize well.

Second, equipped with the geo-referenced dataset, we cast the place recognition problem as a classification task where we use available geotags as a weak form of supervision to train a classifier for each location in the database (chapter 4). These classifiers are subsequently used for ranking the database images at query time.

Finally, as only a few positive training examples are available for each location, we propose two methods to calibrate all the per-location SVM classifiers without the need for additional positive training data. The first method (section 4.2) relies on p-values from statistical hypothesis testing. The second method (section 4.3) performs an affine calibration by appropriately normalizing the learned decision hyperplane. We also describe a memory efficient classifier representation for the sparse bag-of-visual-word vectors (section 4.4) and experimentally demonstrate benefits of the proposed approach (chapter 5).

1.5 Thesis summary

1.5.1 Building geo-referenced datasets

We developed a method for building geo-referenced datasets for place recognition from images available on the Internet. The algorithm collects imagery available on Google Street View. The proposed method is capable of collecting both recent and historical panoramas and can collect depth information along with additional metadata. Subsequently, we build the geo-referenced database of perspective images (55k) from the collected data.

Google Street View provides panoramic 360-degree views from designated roads throughout its coverage area. Google Maps API v3 provides a way to embed the Street View service into a custom website using the JavaScript. The possibilities are, however, quite restricted. The user can only display the most recent panorama (the panoramas are being updated periodically), it is not possible to view the whole panorama, the maximum resolution can not be accessed, a depth information is also not accessible, and only approximate location of the panorama is available.

To deal with the above-mentioned drawbacks, we exploit the communication between the web browser and the Google Street View server and discover a structure of HTTP requests necessary for getting the raw Street View data. These HTTP requests are then used by the algorithm to fetch raw data and to build a geo-referenced dataset from the desired area.

Using the method described in this thesis, we are able to collect full resolution (90 Mpx) spherical panoramas, historical panoramas, accurate GPS coordinates, inertial measurement unit data and

Figure 1.2: **Place recognition by per-location classifiers.** At each place on the map we train per-location classifier. The classifier learns relative importance of the image features for recognizing certain location. Given the unknown query image (blue frame), the similarity with the database images is measured by the calibrated classification score. The heat map layer indicates distribution of the calibrated classification score for the given query image. The four top-ranked database images are shown with its respective location in the map. Notice that one retrieved image is correct (green frame) and depicts the same place while other three top-ranked are incorrect (red frame).

a scene depth representation from post-processed LiDAR point cloud.

We will describe how the Street View data are internally organized as a large directed graph, how to build this graph to get the raw Street View data and, finally, how to use these data to generate a geo-referenced dataset for place recognition.

1.5.2 Per-location classifiers

So far, we have discussed how the geo-referenced database can be built. Armed with the geo-referenced database, we developed a method which is able to localize a query photograph by finding database reference images depicting the same place. We cast the place recognition problem as a classification task, and for each database image, in turn, we train a classifier. The classifier learns relative importance of the image features for recognizing the location. At the query time, a similarity with the database images is evaluated using the classifiers. This is illustrated in figure 1.2 where the heat map layer indicates distribution of the calibrated classification score for a given query image across the designated area.

The local features such as Scale Invariant Feature Transform (SIFT) are heavily used in the computer vision literature. In this work, we use the SIFT features for their befitting properties such as view and scale invariance and robustness to partial occlusion. We aggregate the features

into L2 normalized bag-of-visual-words (BOW) vector which is used as an image representation.

The image retrieval can be done by extracting the SIFT features from the query image, computing its BOW vector and, finally, measuring the similarity with the query by performing a dot product with each database BOW. Besides of the BOW representation, other popular aggregation methods for image representation recently used in the computer vision literature are Vector of Locally Aggregated Descriptors (VLAD) [35] and Fisher Vectors (FV) [36]. In this thesis, we will also utilize FV image representation.

One particular challenge we address here is the problem of confusing features. Man-made structures in the urban area are often similar to each other and appear the same at different geographical locations, cars, pedestrians and vegetation are not relevant for recognizing the place. For instance, consider the Starbucks coffee logo that looks the same at multiple locations in the city, a stop traffic sign or a zebra crossing sign that looks the same across the country, cars buses and pedestrians that typically move, hence can not be a strong indicator for place recognition. Some of these features are, in general, non-informative, but notice that in a particular context at a certain location can be very discriminative. For instance, there are few dozens Starbucks coffee in the city, thousands of zebra crossings and hundreds of stop traffic signs, but there might exist only one place where the Starbucks coffee is situated next to the stop traffic sign and the zebra crossing.

Following this intuition, we aim at learning relative importance of the features in the database image. This is achieved by learning an exemplar support vector machine (e-SVM) classifier for each database image in turn. The learned weight vector will give a higher weight to the discriminative features related to the learned location (the positive training data) and will down weight features that are non-discriminative with respect to the far-away locations (the negative training data).

Since these classifiers are learned independently of each other, their output scores are not directly comparable. Typically, for each trained classifier, this problem can be solved by fitting a sigmoid function (logistic regression) to the distribution of the output scores for negative and positive data. The sigmoid function is then used as a calibration function for the classifier. The output of the classifier can be then treated as a probability score and can be directly compared with other calibrated score.

However, this requires bellyful amount of both positive and negative training data or a held out set devoted only for calibration. This is a problem in place recognition task since we typically have one (or a handful) positive training example for each location and this example is used for training. However, there is plentiful negative examples, all the images that are sufficiently far away (say 200m) from the positive training example can be treated as negative data, because they can not capture the same scene. We take advantage of this fact and propose a new method for calibrating the classifiers from the only negative data.

To summarize, rather than learning the importance of image features globally, we learn its relative importance locally for each database image in turn. This is achieved by learning per-exemplar classifiers. Because each classifier is trained independently, the calibration of the classifier score appears to be a critical issue. Because of the lack of positive training data, the traditional calibration methods can not be used. Therefore we propose calibration procedure that utilizes only the negative data.

We will show how to train the per-location classifiers on top of the bag-of-word features and Fisher vectors image representations, how to calibrate the classifiers from the negative data and how to store them efficiently. Finally, we will demonstrate its performance on three place recognition datasets.

1.6 Thesis outline

In chapter 2 we give an overview of the state-of-the-art related to our work. Namely, we focus on image features and image representations, existing place recognition methods, discriminative learning, support vector machines (SVM) and linear discriminant analysis (LDA). In chapter 3, we present a method for building geo-referenced databases for place recognition. We show how to get full resolution panoramas of the street side panoramas and depth representation from the Google Streetview and how to generate perspective images with depth information from this data. In chapter 4, we introduce per-location classifiers for place recognition and propose two calibration methods. Finally, in chapter 5 we present experimental results of the proposed methods on three datasets obtained by the framework proposed in chapter 3.

1.7 Publications related to the thesis

Journal papers

Gronat, P., Obozinski, G., Sivic, J., Pajdla, T. (2016) Learning and calibrating per-location classifiers for visual place recognition. *International Journal of Computer Vision (IJCV), Springer USA, 10.1007/s11263-015-0878-x* [50%]

Conference papers

Gronat, P., Obozinski, G., Sivic, J., Pajdla, T. (2013) Learning and calibrating per-location classifiers for visual place recognition, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p.907-914* [50%]

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J. (2016) NetVLAD: CNN architecture for weakly supervised place recognition, R. Arandjelovic, P. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* [15%]

Workshop papers and techreports

Gronat, P., Sivic, J., Pajdla, T. (2012) Learning local distance functions for place recognition. *In Proceedings of the 17th Computer Vision Winter Workshop, CVWW2012.* [50%]

Gronat, P., Havlena, M., Sivic, J., Pajdla, T. (2011) Building streetview datasets for place recognition and city reconstruction. *Tech. Rep. CTU-CMP-2011-16, Czech Tech University.* [50%]

Software

Gronat, P. (2015) `streetget`, *A small package for fetching Google StreetView data with easy to use CLI and Python API.* http://www.di.ens.fr/willow/research/streetget [100%]

2 Related work

"My definition of an expert in any field is a person who knows enough about what's really going on to be scared."

— P.J. Plauger

The task of geo-localizing a given input query photograph has recently received considerable attention. There is a lot of applications where the organization of the geographical data is used. The output of the geo-localization system can be a coarse geo-localization on the level of continents and cities [16, 30, 37] or a name of the depicted landmark [40]. In this work, we focus on visually recognizing the "same place" by finding an image in the geo-referenced database that depicts the same building facade or street-corner as shown in the query [10, 13, 39, 56, 66, 69].

2.1 Local Features and Image Representations

Over the years, researchers have proposed a plethora of different visual features spanning a wide spectrum, from very local to full-image representations. The Scale Invariant Feature Transform (SIFT) descriptor [43] is popular in image retrieval and large scale place recognition. The SIFT is a local feature descriptor invariant to translations, rotations and scaling transformations in the image domain and robust to moderate perspective transformations and illumination changes. The descriptor generally describes the spatial distribution of pixel intensity gradients in a patch. A similar approach is adopted by [6] who present the Speeded Up Robust Features (SURF).

Histogram of Oriented Gradients (HOG), also relies on describing gradients in the image but is mostly used in object detection as a representation of an entire object or its part. Dalal et al. [14] introduce Histograms of Oriented Gradient (HOG) features for pedestrian detection. In this work they study the question of feature sets for robust visual object recognition, adopting linear SVM based human detection as a test case. Authors show experimentally that grids of features significantly outperform existing feature sets for human detection. Finally, it is shown that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. In this work, authors also introduced new dataset for human recognition.

Sivic et al. [61] propose a method of object retrieval which searches for and localizes all the occurrences of an object in a video, given a query image of the object. Local viewpoint invariant SIFT features are quantized using a pre-trained visual word vocabulary and aggregated into a term frequency-inverse document frequency (tf-idf) vector. In computer vision and text retrieval community, this approach is also known as bag-of-words (BOW). The temporal continuity of the video within a shot is used to track the regions in order to reject unstable regions and reduce the effects of noise in the descriptors. A benefit of the tf-idf representation is that it can be stored in the inverted file structure which yield in an immediate ranking of the video key frames/imags

containing the object.

It is worth noting that this is the first effective and scalable approach for image retrieval, still very popular. A lot of research has been done on top of this work [47, 46, 67, 33, 50]. For instance, Philbin et al. [47] address problem of building a vocabulary for large-scale image retrieval and introduce a quantization method based on randomized trees and propose an efficient spatial verification stage to re-rank the results returned from our bag-of-words model. Nister et al. [46] present recognition scheme that efficiently scales to a large number of objects. The local region descriptors are hierarchically quantized in a vocabulary tree. The vocabulary tree allows a larger and more discriminatory vocabulary to be used efficiently. Authors demonstrate that the proposed scheme leads to a significant improvement in retrieval quality. The most significant property of the scheme is that the tree directly defines the quantization, hence, the quantization and the indexing are therefore fully integrated.

Turcot and Lowe [67] takes an additional step to bag-of-words (BOW) method to reduce the memory footprint by selecting only a small subset of the training features. After training the BOW vocabulary, each database image is treated as a query image. Then, $m-$best scoring images are geometrically verified via RANSAC for affine transform and survived features are further eliminated by estimating the epipolar geometry. In addition to the feature pruning, authors propose query augmentation by building a graph of geometrically adjacent images. Using this graph, the image can be augmented by adding pruned features from adjacent images.

Motivated by that the Fisher kernels have been introduced to combine the benefits of generative and discriminative approaches, Jegou et al. [33] propose to apply the Fisher kernel framework to the image categorization. In the Fisher kernel framework generative models can process data of variable length and discriminative methods can have flexible criteria. The underlying generative model for the Fisher kernel is a visual vocabulary represented as a Gaussian mixture model which approximates the distribution of low-level features in images. Authors showed that Fischer kernel can be viewed as an extension of the bag-of-words (BOW) approach and demonstrate superior performance on two datasets.

In another work, Jegou et al. [36] aim at large-scale image search and addresses search accuracy, efficiency, and memory usage. They present and evaluate different ways of aggregating local image descriptors into a vector and show that the Fisher kernel achieves better performance than the reference BOW approach for any given vector dimension. Finally, they jointly optimize dimensionality reduction and indexing in order to achieve accurate, yet compact image representation.

To reduce the memory footprint of the image representation, Jegou et al. [34] introduce product quantization for approximate nearest neighbor search. Descriptor space is decomposed into a Cartesian product of low dimensional subspaces where each subspace is quantized separately. A vector is then represented by a short code composed of its subspace quantization indices. Finally, the Euclidean distance between two vectors can be efficiently estimated from their codes. Authors present extensive search accuracy even on image representation encoded into few dozens of bits.

Mid-level visual representations aim to capture information at the level of complexity higher than typical "visual words", but lower than full-blown semantic objects. Singh et al. [60] aim at discovering a set of discriminative patches which can serve as a fully unsupervised mid-level visual representation. The learned mid-level patches should be representative (occur frequently in the visual world) yet discriminative (must be different enough from the rest of visual world). This is achieved by discriminative clustering where each iteration consists of k-means clustering random patches, learning the SVM classifier for each cluster as positive training data and, finally, updating the cluster with the new examples discovered by utilizing the SVM on the validation set.

The method exhibits consistent semantic entities over the clusters. Given a weakly-labeled image collection, Doerch et al. [15] propose a method that discovers visually-coherent patch clusters that are maximally discriminative with respect to the labels and demonstrating state-of-the-art results for scene classification.

Similar to other work in large scale place recognition [13, 39, 56, 66] and image retrieval [46, 47, 61, 36], we describe each image by a set of local invariant features [6, 43] that are encoded and aggregated into a fixed-length single vector descriptor for each image. In particular, in this work we consider the sparse tf-idf weighted bag-of-visual-words representation [61, 47] and the compact Fisher vector descriptors [36].

The resulting vectors are then normalized to have unit L_2 norm and the similarity between the query and a database vector is measured by their dot product. This representation has some desirable properties such as robustness to background clutter and partial occlusion. Efficient retrieval can then achieved using inverted file indexing [34].

2.2 Place recognition

Visual place recognition [13, 39, 56] is a challenging task as the query and database images may depict the same 3D structure (e.g. a building) from a different camera viewpoint, under different illumination, or the building can be partially occluded. In addition, the geotagged database may be very large. For example, we estimate that Google Street View of France alone contains more than 60 million panoramic images. It is, however, an important problem as automatic, accurate and fast visual place recognition would have many practical applications in robotics, augmented reality or navigation.

The *visual place recognition* problem is typically treated as large-scale instance-level retrieval [13, 10, 39, 56, 66, 69], where images are represented using local invariant features [43] encoded and aggregated into the bag-of-visual-words [12, 61] or Fisher vector [36] representations.

While in image retrieval databases are typically unstructured collections of images, place recognition databases are usually structured: images have geotags, are localized on a map and depict a consistent 3D world. Knowing the structure of the database can lead to significant improvements in both speed and accuracy of place recognition. Examples include: (i) the image database can be further augmented by 3D point clouds [38, 41, 32], automatically reconstructed by large-scale structure from motion (SfM) [1, 38, 58], which enables accurate prediction of query image camera position [42, 52, 70, 53]. (ii) constructing an image graph [8, 48, 67], where images are nodes and edges connect close-by images on the map [65], or (iii) using the geotagged data as a form of supervision to select local features that characterize a certain location [39, 56] or re-rank retrieved images [69].

2.2.1 Building explicit 3D reconstruction.

Agarwal et al. [1] present a system that can match and reconstruct 3D scenes from extremely large collections of photographs. The system uses a collection of novel parallel distributed matching and reconstruction algorithms that minimize serialization bottlenecks. Authors show that it is possible to reconstruct cities consisting of 150K images in less than a day on a cluster. Such a 3D point cloud can be further used for city scale place recognition. Klinger et al. [38] describe a structure-from-motion framework that handles generalized cameras, such as moving rolling-shutter cameras,

and works at an unprecedented scale of billions of images covering millions of linear kilometers of roads by exploiting a good relative pose prior along vehicle paths.

Irschara et al. [32] address efficient view registration with respect to a given 3D reconstruction. Authors present a fast location recognition technique based on structure from motion point clouds. Vocabulary tree-based indexing of features directly returns relevant fragments of 3D models. Finally, they propose a compressed 3D scene representation which improves recognition rates while simultaneously reducing the computation time and the memory consumption. The paper demonstrates the approach by matching hand-held outdoor videos to known 3D urban models, and by registering images from online photo collections to the corresponding landmarks.

Li et al. [42] address the problem of place recognition by estimating a full 6-DOF-plus-intrinsic camera pose with respect to a large geo-referenced 3D point cloud. The main contribution is a scalable method for accurately recovering 3D camera pose from a single photograph taken at an unknown location. Authors claim that proposed 2D-to-3D matching approach to image localization is advantageous compared with image retrieval approaches because the pose estimate provides a powerful geometric constraint for validating a hypothesized location of an image, thereby improving recall and precision.

Their approach combines geometric constraints and image based recognition and proposes to compute a representative set of 3D point fragments that cover a 3D scene from arbitrary viewpoints and utilize a vocabulary tree data structure for fast feature indexing. A subsequent matching approach and geometric verification directly delivers the pose of the query image.

Sattelr et al. [52] propose a pipeline for determining the pose of a query image relative to a 3D point cloud reconstruction. They propose a search method based on both 2D-to-3D and 3D-to-2D search to establish matches between image features and scene points needed for pose estimation. A unified formulation of search in both directions allows exploiting the distinct advantages of both strategies. In [53] Sattler et al. propose a localization approach based on prioritization scheme of feature matches that allows to significantly accelerate 2D-to-3D matching. Because the scheme itself suffers from the feature quantization artifacts, authors propose to recover the matches lost due to the quantization by a 3D-to-2D search. Finally, it is shown how to exploit co-visibility information from the reconstruction process and to use it to improve the efficiency of the localization pipeline.

In [70], Zeisl et al. propose to shift the task of finding correct image correspondences from the matching stage to the pose estimation step. Instead of using first nearest neighbors and retaining matches that are likely to be inliers, authors simplify the matching problem and consider one-to-many correspondences, which however results in a large number of matches with a very small inlier ratio. Then an extensive spatial verification is performed early on in the pose estimation procedure. Authors introduce a voting-based spatial verification process that exploits a known gravity direction and an approximate knowledge of the camera height using a setup.

Li et al. [41] use a 3D point cloud built from a large photo collection to localize given query photograph. Based on the knowledge about features visibility in the 3D model authors devise an adaptive, prioritized algorithm for matching a representative set of SIFT features covering a large scene to a query image for efficient localization. Their approach is based on considering features in the scene database, and matching them to query image features, as opposed to more conventional methods that match image features to visual words or database features.

2.2.2 Constructing an image graph.

Turcot and Lowe [67] authors propose query augmentation by building a graph of geometrically adjacent images. Using this graph the image can be augmented by adding features from adjacent images.

Cao et al. [8] exploit a structure of a geo-referenced database and build a graph of images based on visual connectivity. Then they identify subgraphs and learn a local distance function for each. Given the query image, each database image is ranked according to the learned local distance functions in order to place the image in the right part of the graph. Torii et al. [65] formulate place recognition problem as a regression on an image graph with geotagged images as nodes and edges connecting close by images. Given a query image, a similarity between the query and pairs of database images is computed using edges of the graph and considering linear combinations of their feature vectors. Finally, the query location can be predicted by interpolating locations of matched images in the graph without estimation of multi-view geometry.

2.2.3 Using the geotagged data as a form of supervision.

One of the successful data-driven place recognition approaches were presented by Hays and Efros [30]. They target on recognition on the scale of the entire planet. Authors propose a simple algorithm for estimating a distribution over geographic locations from a single image using a purely data-driven scene matching approach. Given the query image, a set k-nearest neighbors in the multi-feature space forms an estimate of a geographic location - a probability graph over the entire globe. This is achieved by leveraging a large geo-referenced dataset of the size of 6 million images. Zamir et al. [69] propose a feature matching based scheme. They quantize SIFT features into a learned vocabulary tree. At the query time, each feature is pruned based on proposed pruning strategy and survived features are quantized. Quantized features votes for location on a map using the geotags, weak votes are removed by measuring the Kurtosis of the vote distribution. Finally, Vote distribution is smoothed by a Gaussian and the query image is localized by the identifying the database image corresponding to the highest peak of the voting distribution.

Knopp et al. [39] propose a method to avoid features leading to confusion in place recognition using geotags attached to database images as a form of supervision. One of the key problems in place recognition is the presence of objects such as trees or road markings, which frequently occur in the database and hence cause significant confusion between different places. Authors develop a method for automatic detection of image-specific and spatially-localized groups of confusing features.

Torii at al. [66] deal with repeated structures that are notoriously hard for establishing correspondences using multi-view geometry or for place recognition since repeated structures violate the feature independence assumed in the bag-of-visual-words representation. Authors describe a representation of repeated structures suitable for scalable retrieval and geometric verification which yield an important distinguishing feature for many places.

Sattler et al. [51] aim at re-ranking the retrieved images within the shortlist. Standard re-ranking technique utilizing spatial verification via RANSAC [29] implicitly assume that the number of inliers found by spatial verification can be used to distinguish between a related and an unrelated database photo with high precision. Authors show that this assumption does not hold for large datasets due to the appearance of geometric bursts, for instance, sets of visual elements appearing in similar geometric configurations in unrelated database photos, and propose algorithms for detecting and

handling geometric bursts by exploiting the geo-tags obtained from GPS or SfM.

Schindler et al. [56] look at the problem of place recognition in a large image dataset using a vocabulary tree. In this work, authors demonstrate that traditional feature matching approaches do not work very well as the size of the dataset increases and show retrieval performance can be significantly improved by carefully selecting the vocabulary.

Chen et al. [10] use a large geo-referenced database of street-level images and use facade-aligned and viewpoint aligned representations to localize the query photograph. Authors present their own geo-referenced dataset as well as a set of query images from mobile devices with the ground truth labels. However, it is worth noting that their ground truth is very noisy and contains many false positive images. This makes it difficult to use this dataset for accurate evaluation of place recognition. Gronat et al. [26] proposed method for building geo-referenced datasets for place recognition and city reconstruction. Described pipeline is amenable to downloading Google Street view panoramas with the corresponding metadata.

Kalogerakis et al. [37] aim on estimating a geographic location of sequences of time-stamped photographs. A prior distribution over travel describes the likelihood of traveling from one location to another during a given time interval. An image likelihood for each location is defined by matching a test photograph against the training database. Inferring location for images in a test sequence is then performed using the Forward-Backward algorithm. Utilizing temporal constraints allows the method to geolocate images without recognizable landmarks.

■■■

In contrast to the related work summarized above, in this thesis, we investigate learning a discriminative place-specific image representation. A similar idea has been recently explored in [8] who learn a graph-based discriminative representation for landmark image collections where typically many images are available for each landmark. In this thesis, we focus on street-level images such as Google Street View, which have greater coverage, but typically only one or a small number of images are available for each place. To address this issue, we learn a discriminative re-weighting of the descriptor unique to each image in the database using per-exemplar support vector machine [44].

2.3 Support Vector Machines

In two decades, Support Vector Machine (SVM) became an indivisible component of computer vision techniques. Below, we give a brief overview of the most relevant publications related to the thesis.

2.3.1 Support Vector Machines for Place Recognition

Li et al. [40] aim at image classification using multiclass SVM. They exploit geotags of a large collection of photos from Flickr to identify peaks in the density distribution. These peaks correspond to frequently photographed landmarks. They learn models for these landmarks with a multiclass

SVM, using vector-quantized interest point descriptors as features. Authors conclude that in some cases image features alone yield comparable classification accuracy to using text tags as well as to the performance of human observers.

Zadrozny et al. [68] show how to obtain accurate probability estimates for multiclass SVM problems by combining calibrated binary probability estimates. Authors also propose a method for obtaining calibrated two-class probability estimates that can be applied to any classifier that produces a ranking of examples.

Doersch et al. [16] seek to automatically find visual elements, e.g. windows, balconies, and street signs, that are most distinctive for a certain geo-spatial area, for example, the city of Paris. They use discriminative clustering approach that is able to take into account a weak geographic supervision, in order to identify the visual elements distinguishing architectural elements of different places. Authors show that these elements are visually interpretable and perceptually geo-informative.

2.3.2 Per-exemplar support vector machines

The exemplar support vector machine (e-SVM) has been used in a number of visual recognition tasks including category-level recognition [44], cross-domain retrieval [59], scene parsing [62] or as an initialization for more complex discriminative clustering models [16, 60]. The main idea is to train a linear support vector machine (SVM) classifier from a single positive example and a large number of negatives. The intuition is that the resulting weight vector will give a higher weight to the discriminative dimensions of the positive training data point and will downweight dimensions that are non-discriminative with respect to the negative training data.

Malisiewicz et al. [44] propose a method object detection based on training a separate linear SVM classifier for every exemplar in the training set. Each the exemplar-SVMs is defined by a single positive instance and potentially millions of negatives examples. While each detector is quite specific to its exemplar, it is observed that an ensemble of such exemplar-SVMs offers surprisingly good generalization.

The objective of Shrivastava et al. [59] is to find visually similar images such as photos taken over different seasons or lighting conditions, paintings, hand-drawn sketches, etc. They propose a method that estimates the relative importance of different features in a query image based on the notion of “data-driven uniqueness". To learn the weight vector, they utilize per-exemplar SVMs similarly to Malisiewicz et al. [44] except that the negative training data are not guaranteed to contain negative examples only. In practice, this does not seem to hurt the SVM, suggesting that this approach is yet another application where the SVM formalism can be successfully applied. The exemplar support vector machine is learned at query time where the weight vector is used as a new query image representation. However, this requires training a new classifier afresh for each query that is computationally demanding.

An interesting alternative to linear exemplar-SVM is presented in work of Gharbi et al. [19]. They re-interpret the exemplar-SVM between a single point and set of negative examples as the computation of the tangent to the manifold of images at the query. They show that in high-dimensional space of the image features all points lie at the periphery, and they are usually separable from the rest of the set. The set of all images in the feature space is approximated by a Gaussian fitted by computing the covariance matrix. The computation of the tangent at a query point is performed by multiplication by the inverse of the covariance matrix. This approach results in a speedup of the image retrieval tasks and is shown that is equivalent to feature space whitening. This approach related to [28] and shares many similarities with Linear Discriminant Analysis briefly

discussed in the section below.

Tighe et al. [62] utilize exemplar SVM for labeling each pixel in an image with its semantic category, achieving a broad coverage across hundreds of object categories. The system combines region-level features with per-exemplar SVM sliding window detectors.

In this work, similar to [44] who learn per-exemplar object category representation, we learn per-exemplar classifiers for each place in the database offline. A key advantage is that each per-exemplar classifier is trained independently and hence the learning can be heavily parallelized. The per-exemplar training brings, however, also a significant drawback. As each classifier is trained independently a careful calibration of the resulting classifier scores is required [44].

2.3.3 Calibrating classifier scores

Several calibration approaches have been proposed in the literature (see [18] and references therein for a review). The most known consists of fitting a logistic regression to the output of the SVM [49].

Platt et al. [49] present calibration of the SVM classifier using the sigmoid function. The output of the classifier is turned into the posterior probability to enable post-processing. Instead of creating probabilities directly by training a kernel classifier with a logit link function and regularizing maximum likelihood score, which produces non-sparse kernel machines, authors first train the SVM and then train the parameters of an additional sigmoid function to map the SVM outputs into probabilities. The SVM and sigmoid yields probabilities of comparable quality to the regularized maximum likelihood kernel method, while still retaining the sparseness of the SVM.

This approach, however, has a major drawback as it imposes a parametric form (the logistic a.k.a. sigmoid function) of the likelihood ratio of the two classes, which typically leads to biased estimates of the calibrated scores. Another important calibration method is the isotonic regression [68], which allows for a non-parametric estimate of the output probability. Unfortunately, in our setup, the fact that we have only a single positive example (or only very few of them which are almost identical, and which are all used for training) essentially prevents us from using any of these methods. To address these issues, we develop two classifier calibration methods that do not need additional labeled positive examples.

Related to propsed calibration methods is also the recent work of Scheirer et al. [55] who develop a classifier calibration method for face attribute similarity search. Scheirer et al. [55] aim at classifying multiple visual attributes utilizing a large dataset of face images. A raw attribute score is obtained by training the SVM classifier. To compare the raw scores, authors propose a calibration procedure that fits a Weibull distribution to the distribution of scores. The calibration turns a binary SVM to a probabilistic decision. Their method (discussed in more detail in section 4.2) also does not require labeled positive examples but, in contrast to us, uses a parametric model (the Weibull distribution) for the scores of negative examples.

2.4 Linear discriminant analysis and whitening

Our work is also related to linear discriminative transformations of feature space that have shown good performance in object recognition [19, 28] and 2D-3D alignment [5, 4]. Aubry et al. [5] present a technique that can reliably align arbitrary 2D depictions of an architectural site, including drawings, paintings, and historical photographs, with a 3D model of the site. They develop a new

compact representation of complex 3D scenes. The 3D model of the scene is represented by a small set of discriminative visual elements that are automatically learned from rendered views. The set of visual elements, as well as the weights of individual features for each element, are learned in a discriminative fashion. The authors present closed form solution which is tightly related to LDA.

Aubry et al. [4] aim on object category detection of "chair" as a type of 2D to 3D alignment problem. They utilize the large quantities of 3D CAD models that have been made publicly available on-line and propose an exemplar-based 3D category representation. Authors learn view-dependent mid-level visual elements utilizing approach similar to LDA and propose affine calibration from negative examples. Their system is able to align 3D models with 2D objects in images in complex scenes containing chairs.

While conceptually the idea of finding a discriminative projection of the original feature space is similar to our work, the main difference is in the used loss function. While we use hinge loss [57] to train the new discriminative representation of each place, [5, 19, 28] use the Euclidean loss. The advantage of using the Euclidean loss is that the discriminative projection can be computed in closed form. Hariharan et al. [28] point out that training SVM over HOG features can become expensive as the number of classes increases. In such cases, instead of SVM, they propose to utilize LDA models. Estimated covariance matrices capture properties of the natural images, and whitening HOG features with these covariances removes naturally occurring correlations between the HOG features. Authors demonstrate that whitened features are considerably better than the original HOG features and demonstrate its performance experimentally.

Using the Euclidean loss, the resulting projection is tightly related to Linear Discriminant Analysis and whitening the feature space [5, 19, 28]. Such whitened representations have shown promise for image retrieval [33] or matching HOG [14] descriptors [15], however, we have found they do not perform well (chapter 5) for place recognition.

3 Building datasets for place recognition

"Amazing things happen when you pull individual pieces of information together into larger linked datasets: meaning emerges, as you produce facts from figures."

— Ben Goldacre

Google Maps combined with Street View images can serve as a powerful tool for place recognition or city reconstruction tasks. In this chapter, we present a way how to build geotagged datasets of perspective views from Google Maps. Given the GPS coordinates and the parameters of the area of interest, the algorithm can build a graph of panorama locations, download corresponding panoramas and depth maps, and generate perspective views.

Each panorama on Google Maps Street View has associated metadata from which the GPS location and the direction of the view can be extracted. The metadata also contains information about the neighboring panoramas, hence a list of panoramas covering a certain area can be built. The metadata also contains links to the temporal neighbors which are the panoramas of the same place captured at a different time. Temporal data turned to be a very valuable source of data.

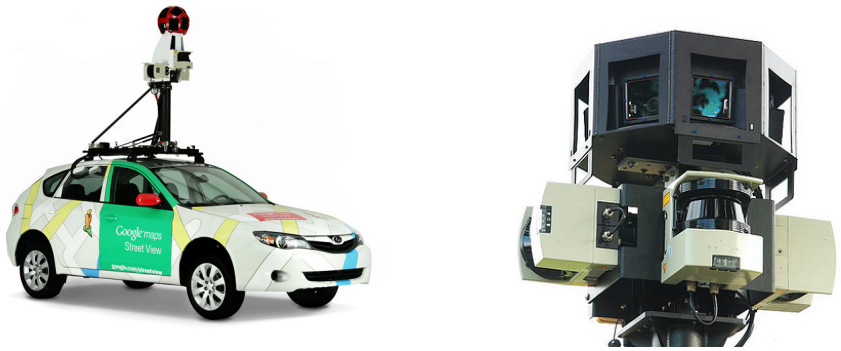
Finally, each downloaded panorama is cut into a set of overlapping perspective views and stored while the camera GPS location, yaw, and pitch, date are coded in the filename of the perspective view and the associated metadata file. Such geotagged database can be subsequently used for place recognition and structure from motion 3D reconstruction.

The Google Maps API v3 provides convenient access to some of the Google Street View data. However, it has some limitations in terms of accessibility to the data and number of requests per hour. Hence, the proposed algorithm does not use the Google Street View API but is build on HTTP request communication between the Street View client and the Google Maps server.

In this chapter, we provide an abstraction of how the Street View data are represented at the Google Maps server, what particular HTTP requests to use to fetch the Street View data and how to decode the obtained data. Then we describe how to build a spherical panorama out of this data, how to retrieve underlying depth information, and finally, how to generate perspective views from the spherical panoramas.

3.1 A big picture

The Street View data is mostly captured by cars equipped with various kind of sensors attached to the data capturing equipment(see Figure 3.1). Some sensors important for the Street View are spherical ladybug camera, LiDAR, inertial measurement unit (IMU) and accurate GPS. Other sensors such as WiFi and 3G/4G signal are involved only in the Goole Maps services helping the localization when used on a cell phone. It is worth noting that since 2015 the cars began carrying



(a) (b)

Figure 3.1: **Google Street View capturing equipment.** (a) A Google car with capturing equipment. (b) Detail of the capturing equipment. The Street View capturing device contains a spherical camera system, LiDAR, inertial measurement unit (IMU), accurate GPS, WiFi monitor and pollutant sensors.

sensors to detect pollutants such as nitrogen dioxide, ozone, and particulate matter.

The Street View data is captured approximately every $15m$ on the road. At each panorama location, the device captures a panoramic image, a GPS data, and IMU data are along with the LiDAR point cloud. The spherical panorama is then stored at different resolutions called zoom levels, and for efficiency, each panoramic image is broken into few dozens of image tiles that can be stitched back together. The details are given in the section below.

The LiDAR data are post-processed, the scene depth is approximated as a set of 3D planes. The planes are fitted to the LiDAR point cloud via RANSAC. The parameters of the planes can be retrieved from the Google server as metadata. The details will be given later in this chapter.

The metadata also contains a GPS location, IMU data such as a yaw w.r.t. the absolute North, hence the car heading in the world, and the gravity vector, is just to name a few. Most importantly, the metadata contains links to the spatially adjacent panorama locations. For instance, at the panorama location at a street crossing the adjacent panoramas can be backward, forward left and right. This is illustrated in the Figure 3.2 where the Street View is depicted as a large graph. Each of the neighbor panorama locations contains its own list of spatial neighbors and so on.

Finally, each panorama location contains links to the temporal neighbors. The temporal neighbor is a panorama location that is captured approximatelly at the same GPS position but is captured at a different time, typically a different year or month. This is illustrated in Figure 3.2. The temporal links can be retrieved from metadata, details will be given later in this chapter.

To conclude, the street view can be viewed as a large directed graph where a node represents the data (spherical panorama, IMU, depth representation e.t.c.) and edges represent the adjacency (see Figure 3.2) of the panorama locations. There are two types of adjacency, the spatial adjacency, and the temporal adjacency.

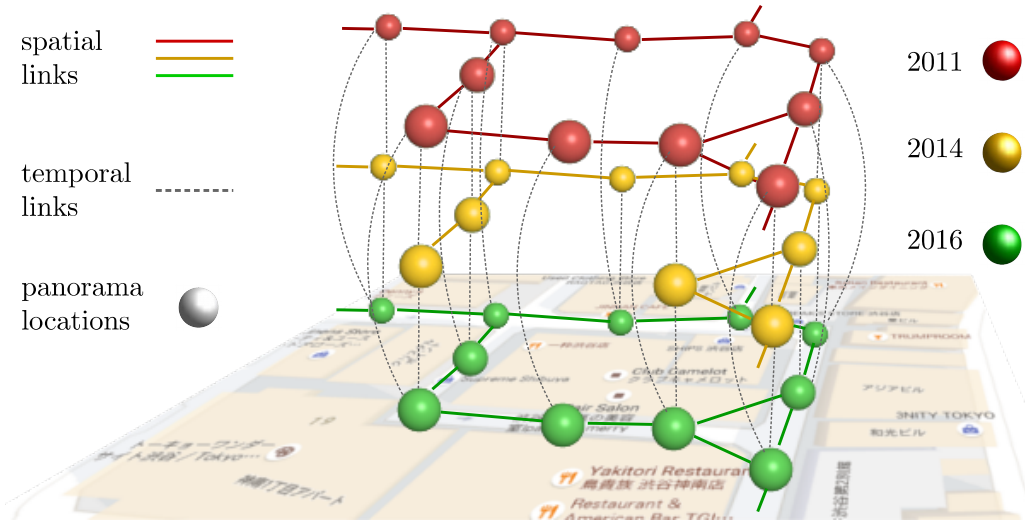


Figure 3.2: **Google Street View as a graph.** Each node in a graph represents a panorama location, green. There are three layers of nodes, green, yellow and red. Each layer corresponds to a set of panorama locations captured in a different year. Solid edges represent spatial adjacency while dashed edges represent temporal adjacency. Notice that panorama locations captured at different year lie approximately at the same location (they have very similar GPS coordinates), however, for some years a panorama location may not be sampled (see the yellow layer), the node is missing. Notice that in general the graph is directed edges, however for brevity, we depict the graph as undirected.

3.2 Algorithm overview

As mentioned in the previous section, the Street View can be viewed as a large graph. Thus, given a starting node, and an area of interest, for instance, a GPS rectangle, the goal is to (i) build the graph and (ii) for each node download the spherical panorama and metadata. We use the breadth-first search (BFS) algorithm and rather than first building the graph and then fetching the data we fetch the data while exploring the graph. The algorithm is sketched below.

Algorithm 1 Crawler

```
1:  procedure CRAWLER(id_start, validArea())
2:      V ← empty set                                  ▷ visited panoramas
3:      Q ← empty queue                                ▷ queued panoramas
4:      Enqueue(Q, id_start)
5:      while not Empty(Q) do
6:          id ← Dequeue(Q)
7:          if Has(V, id) then
8:              continue
9:          Add(V, id)
10:         gps ← getGPS(id)
11:         if not validArea(gps) then
12:             continue
13:         saveData(id)                               ▷ save panorama and metadata
14:         for id_n in getNeighbours(id) do
15:             if not Has(V, id_neighbour) then
16:                 Enqueue(Q, id_neighbour)
17:         for id_n in getTemporalNeighbours(id) do
18:             if not Has(V, id_neighbour) then
19:                 Enqueue(Q, id_neighbour)
        return
```

A speed of a download is limited by the Internet connection and latency between HTTP requests. While there is nothing to do about the former problem, the overall latency can be amortized by parallelizing the BFS. Fetching one panorama location requires from a few dozens to hundreds of HTTP requests each of which has some associated latency. Hence, while waiting for the response of the first HTTP request, another request can be send and so on. Hence, the while-loop in Algorithm 1 can be parallelized by (i) using multiple threads and thread-safe implementation of the queue data structure and (ii) by parallelizing the function `saveData` which has to fetch many panorama image tiles as described in the section below. Detailed implementation can be found in the documentation of the `streetget` package [25].

3.3 Getting initial panorama hash ID

Each panorama location, a node in the graph (see Figure 3.2), is identified by a unique hash id. Hence, the adjacency list of neighbor panorama locations is represented by a list of unique hash

Feature name:	Location in JSON object:
Panorama hash ID	`data['Location']['panoId']`
Panorama latitude	`data['Location']['lat']`
Panorama longitude	`data['Location']['lng']`
Panorama heading	`data['Projection']['pano_yaw_deg']`
Panorama date	`data['Data']['image_date']`
Panorama width	`data['Data']['image_width']`
Panorama height	`data['Data']['image_width']`
Tile width	`data['Data']['tile_width']`
Tile height	`data['Data']['tile_height']`
Zoom levels	`data['Location']['zoomLevels']`
Depth binary string	`data['model']['depth_map']`

Table 3.1: **An important part of the panorama metadata.** The JSON metadata are parsed to the Python `data` object.

identifiers. The hash id is required to fetch the panorama image and the metadata. Hence, the algorithm must be supplied by a panorama id of the initial location.

To retrieve the panorama hash id a geocode HTTP request must be sent. The request is supplied by initial GPS coordinate and a search radius in meters. If a panorama is present at the initial location within the specified radius, a server's response is metadata in a JSON format. This metadata contains, among the other information, a hash id of the found panorama. The detail of the HTTP geocoding request is given in the Appendix A. The relevant components of the JSON metadata, including the panorama location hash id, are discussed next.

3.4 Metadata

There are two important HTTP requests, each retrieving different portion a panorama location metadata as a serialized object. The first portion of the metadata contains information about the panorama location such as a panorama hash id, GPS location, panorama heading w.r.t. the absolute North, available zoom levels, a size of the image tiles, and a depth information encoded in a binary string, is just to name a few. The second portion of the metadata contains temporal links to adjacent panorama locations. We first discuss where to find important data about the panorama location and then, second, we discuss how to retrieve temporal links contained in the second metadata object.

The HTTP request for the first portion of the metadata can be found in Appendix A. Response to the request is a serialized object in JSON format. For brevity, let assume that JSON string has been deserialized in Python programming language into the object `data`. The data structure is fairly complex, Table 3.1 highlights how to navigate to the most important piece of information inside the `data` object. A conversion to other programming languages such as R, C++ or Matlab is straightforward.

A detail of the HTTP request for the second portion of the metadata can be found in Appendix A. A response to the request is a serialized javascript object which does not exactly follow the

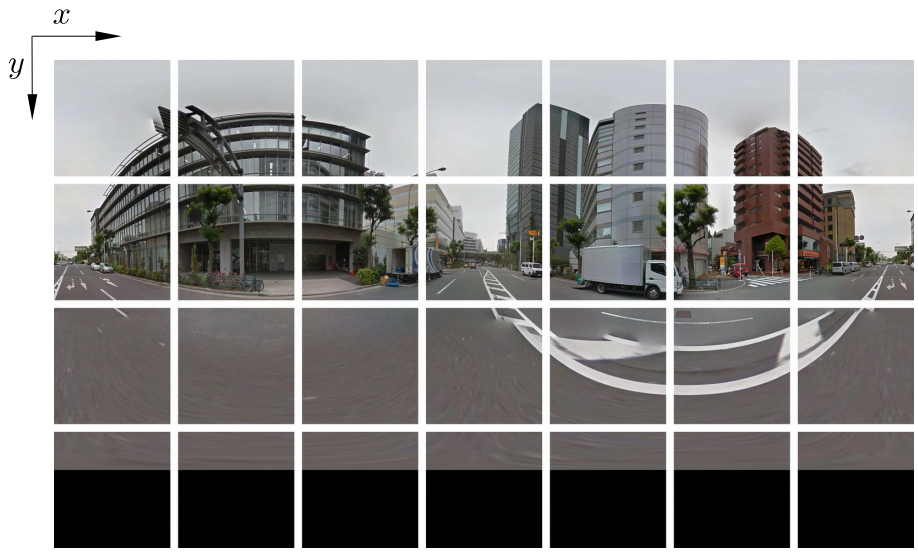


Figure 3.3: **Panorama tile mosaic at zoom level 3 before stitching and cropping.** Notice that panorama is wrapped around. After stitching the tiles, the panorama must be cropped to the appropriate size.

JSON format, but can be converted to JSON (see A for details).

performing the following steps; (i) remove the first line and (ii) replace each `','` with `'null,'` if the comma `','` does not have a value on its left size. For example, the string `[3,4,1, , , [ , , ,[6]]]` would be converted to `[3,4,1,null,null,[null,null,null,[6]]]`.

After applying these steps, the string can be parsed as JSON. For brevity, let assume that JSON string has been deserialized in Python programming language into object `data`. A conversion to other programming languages such as R, C++ or Matlab is straightforward. The available dates of temporal neighbors can be found in the list at `data[1][0][5][1][8]` and corresponding hash id's can be than found in the list `data[1][0][5][1][3][0]` when iterated backwards. The extraction of the information requires more work and the details are given in Appendix A.

3.5 Stitching a spherical panorama

The 360° x 180° panorama in the equirectangular projection model is stored on Google server at different zoom levels. The panorama is saved as a few dozens of image tiles, 1 to 299, depending on the zoom level, that can be downloaded and stitched together to form an equirectangular image. This is illustrated in Figure 3.3. Each tile can be fetched by sending an HTTP request. The request is supplied by panorama hash id, the *zoom_level* and x, y coordinate of a particular tile (see Appendix A for details). At the time of writing this thesis, each panorama tile has a size of 512 x 512 pixels and can be retrieved from metadata (see Table 3.1).

The number of tiles and resolution of the spherical panorama at different zoom levels was determined experimentally, and results are shown in table 3.2. Notice that panorama dimensions are typically not multiples of the tile size, hence the stitched panorama must be cropped. This is

Zoom level	Panorama size	Number of tiles
5	13312 x 6656	26 x 13
4	6656 x 3328	13 x 7
3	3329 x 1664	7 x 4
2	1665 x 832	4 x 2
1	833 x 416	2 x 1
0	417 x 208	1 x 1

Table 3.2: **Panorama resolution and the number of image tiles.** At the time of writing this work, for all zoom levels, the image tile size was 512x512. Notice that panorama resolution is not multiple of 512 hence, after stitching, the panorama must be cropped to the appropriate size.

illustrated in Figure 3.4.

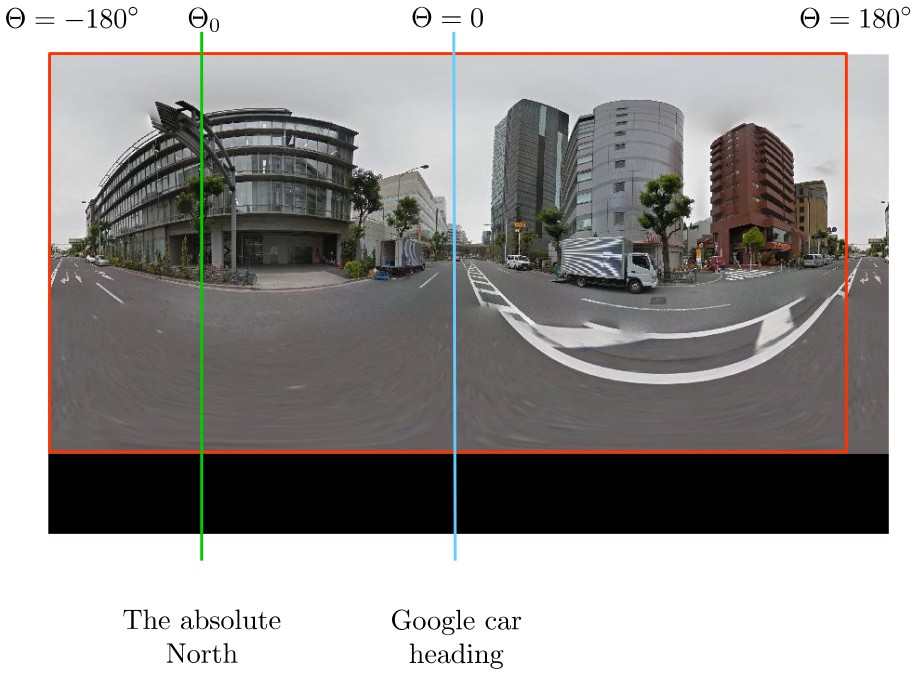


Figure 3.4: **Panorama crop and absolute North direction.** After stitching the image tiles, the panorama must be cropped to the appropriate size (red). Notice that stitched panorama wraps around. A center of the panorama matches the Google car heading (blue), the angle Θ. A direction of the absolute North (green), the angle Θ_0, is different for each panorama and can be found in the metadata.

3.6 Cutting perspective images

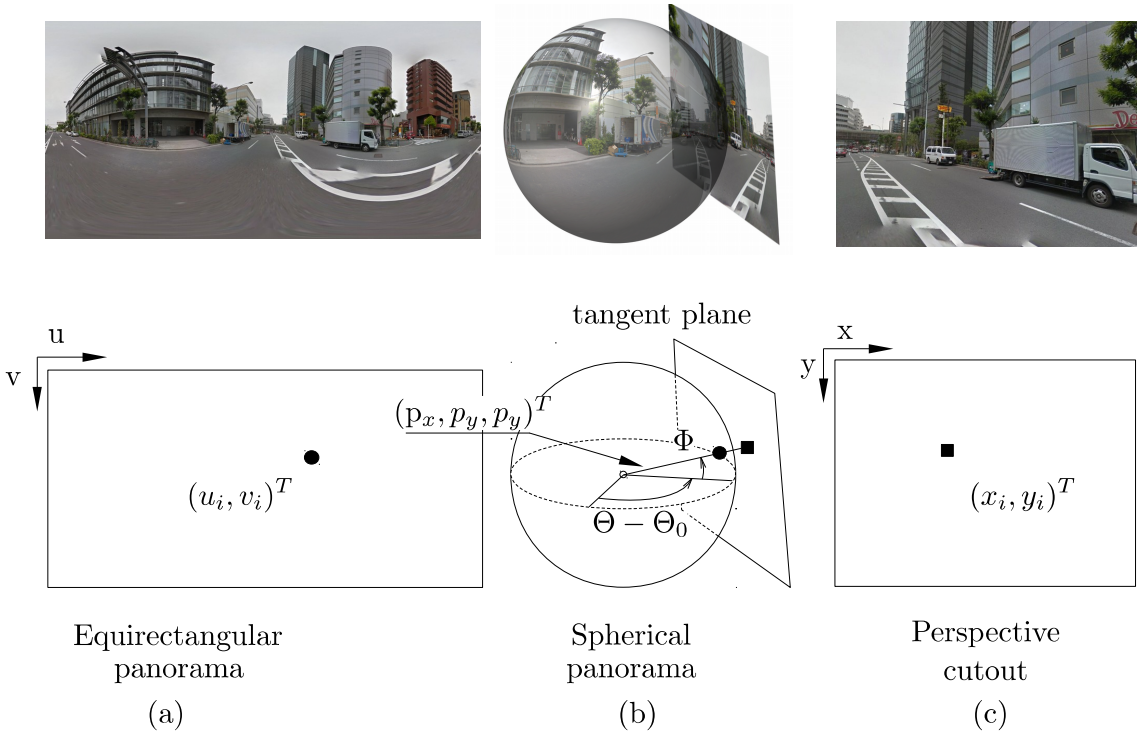


Figure 3.5: **From panorama to perspective image.** Equirectangular image (left) is projected onto a surface of the unit sphere and is then projected to a tangent plane (center). Resulting perspective image is shown on the right.

Perspective views are cut out from the downloaded panoramic images. A given panorama, the equirectangular image, can be mapped onto a surface of a unit sphere using the transformation from image points to unit vectors of their rays which can be formulated as follows. For the equirectangular image having the dimensions I_h and I_w, a point $\mathbf{u} = (u_i, u_j)^T$ in the image coordinates is transformed into a unit vector $\mathbf{p} = (p_x, p_y, p_z)^T$ in spherical coordinates such that:

$$p_x = \cos\Phi \sin(\Theta - \Theta_0), \quad p_y = \sin\Phi, \quad p_z = \cos\Phi \sin(\Theta - \Theta_0) \tag{3.1}$$

where angles Θ and Φ are computed as follows

$$\Theta - \Theta_0 = \left(u_i - \frac{I_w}{2}\right)\frac{2\pi}{I_w} \tag{3.2}$$

$$\Phi = \left(v_j - \frac{I_h}{2}\right)\frac{\pi}{I_h} \tag{3.3}$$

To generate the perspective image, we project a surface of the unit sphere to the tangent plane of a perspective image as follows. We (i) compute the unit vector $\mathbf{p}$ for each pixel of the tangent

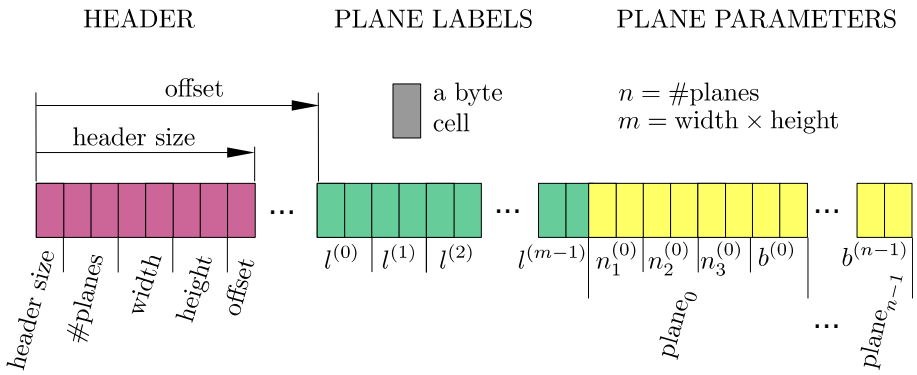


Figure 3.6: **Internal structure of the depth binary data string.** The string consists of three parts, a header, plane labels and plane parameters. The header contains information about its length in bytes and offset. Then it contains a size of a label matrix *width* and *height* and, finally, a number of planes. Another part contains *width* x *height* plane labels as unsigned short integers. The last part contains plane parameters, the normal vector $\mathbf{n} = (n_1, n_2, n_3)^T$ and bias b all represented as floats.

plane, (ii) convert it to spherical coordinates and finally (iii) convert the spherical coordinates to the Cartesian coordinates of the equirectangular panorama and sample the corresponding pixel value. Typically, we generate perspective images of the size 1024 x 768 pixels with a horizontal field of view (HFOV) of 90° by projecting the surface of the unit sphere to its tangent planes in 12 different directions per 360°. Technically, we perform a bilinear interpolation in the source equirectangular image coordinates.

Angle $\Theta = 0°$ corresponds to the center of the stitched panorama, the Google capturing device heading, while Θ_0 indicates a direction of the absolute North as illustrated in the image 3.4. Relevant angles Θ_0 for generating cutouts are can be found in corresponding metadata (see item 'Panorama heading' Table 3.1). Figure 3.9 at the end of this chapter shows an example of perspective cutouts.

3.7 Depth representation

A depth of a scene is captured by a LiDAR and is initially represented as a point cloud. Then, the depth representation is simplified such that the scene is approximated by a set of 3D planes. It is a reasonable representation as the Street View mostly captures the urban areas where the scene objects are planar (building facades, pavements, traffic signs, etc.) or can be approximated by a set of planes.

The scene depth representation is contained inside the first portion of the metadata file (see Table 3.1). The depth representation is stored in a binary string. Its structure is depicted in Figure 3.6

The binary string can be divided into three parts; a header, plane labels, and plane parameters. The first byte in the header represents a length of the header in bytes. The header contains three unsigned short integers that represent a number of 3D planes and a plane label matrix *width* and *heigth*. The entire binary string is represented in little endian notation.

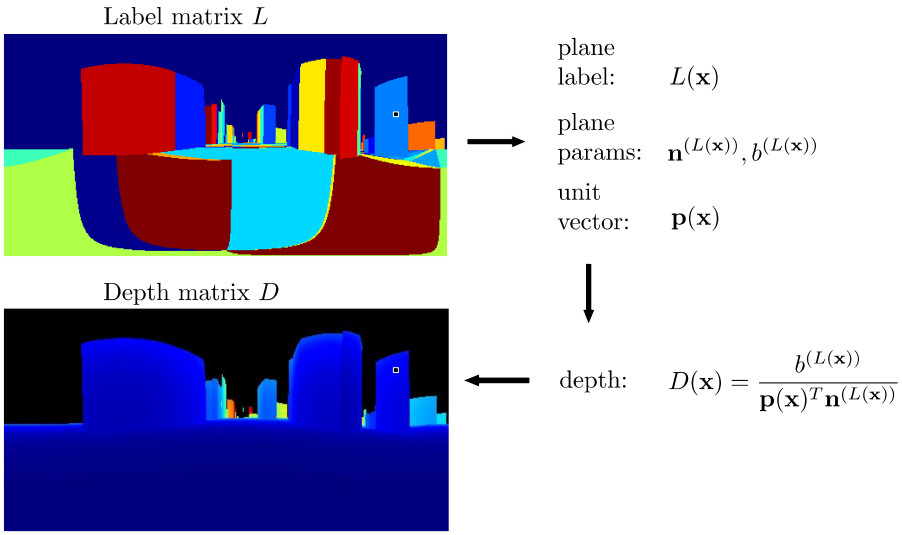


Figure 3.7: **From label matrix to the depth map.** To compute a depth at position $\mathbf{x}$, here illustrated as a small black pixel with the white edge, of the label matrix L (top), we get label id $L(x)$ and retrieve the respective plane parameters $\mathbf{n}$ and b from the metadata. Similarly to perspective image cutout, we compute unit vector $\mathbf{p}$ and, finally, compute the depth as an intersection of the ray defined by $\mathbf{p}$ with the 3D plane.

After the header follow *witdh* x *height* unsigned short integers that represent a label matrix. Last part of the binary string contains the number of planes x 4 signed floats that represent normal vector $\mathbf{v}$ of a plane and its distance from origin b. Finally, it is worth noting that the binary string in metadata is compressed. Details of the binary string unpacking and decoding are given in Appendix A.

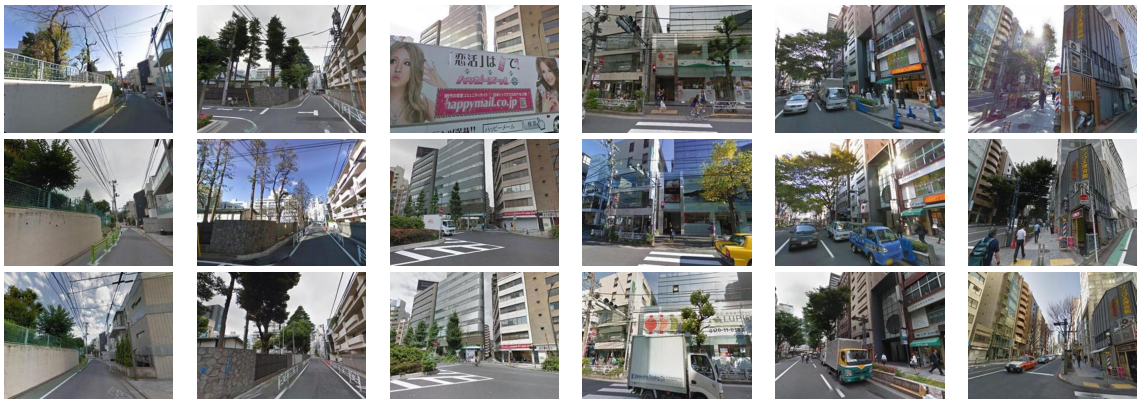
Finally, the label matrix L can be composed by stacking the label values into a matrix of the size *width* x *depth*. Each entry of the label matrix L can be converted to the depth as depicted on figure 3.7. Each label at coordinate $\mathbf{x} = (x, y)^T$ has assigned a unit vector $\mathbf{p}$, see Equation 3.1, and associated plane parameters, the normal vector $\mathbf{n}$ and bias b. The corresponding depth is then computed as an intersection of the ray defined by unit vector $\mathbf{p}$ and the plane defined by $\mathbf{n}$ and b as follows

$$d = \frac{b}{\mathbf{p}^T \mathbf{n}} \tag{3.4}$$

The result is a depth matrix that can be thought as an equirectangular depth map. A procedure to generate perspective depth maps is be described next.

3.8 Perspective depth map

A generation of the perspective depth map is a combination of the perspective cut out and the depth map generation described above. A generation of the pixel-aligned perspective depth map consists of two steps; (i) computation of perspective cutout from the label matrix and (ii) computation of depth for each pixel of the perspective label matrix, hence the intersection of rays with respective planes. The first step is essentially identical to cutting the perspective images except that instead of RGB channels there is only one channel and instead of linear interpolation, the nearest neighbor interpolation is performed since the pixel values correspond to the integer plane label. Perspective depth maps are illustrated in Figure 3.10 at the end of this chapter.



(a) (b) (c) (d) (e) (f)

Figure 3.8: **Examples of the Google Street View temporal imagery.** Each column shows perspective images generated from panoramas from nearby locations, taken at different times. The goal of this work is to learn from this imagery an image representation that: has a degree of invariance to changes in viewpoint and illumination (a-f); has tolerance to partial occlusions (c-f); suppresses confusing visual information such as clouds (a,c), vehicles (c-f) and people (c-f); and chooses to either ignore vegetation or learn a season-invariant vegetation representation (a-f).

3.9 Importance of the temporal data

Regarding the place recognition task, the temporal data has recently been used in [2] where authors propose a new Convolutional Neural Network architecture for place recognition. Authors implement a layer that that mimics behavior of the Vector of Locally Aggregated Descriptors (VLAD) which is a popular descriptor pooling method for both instance level retrieval [36] and image classification [20].

In the work, the dataset is built by the method proposed in this chapter. Examples of collected temporal imagery are shown in figure 3.8. The figure captures several places captured at different times. Notice slight changes in the camera viewpoint, changes illumination conditions, nonstatic objects such as cars, pedestrians, billboards and other occluders and the vegetation that makes place recognition task challenging. It has been shown that the temporal data is crucial for good

place recognition accuracy as, without it, the neural network does not generalize well, and learns, for example, that recognizing cars is important for place recognition, as the same parked cars appear in all images of a place. It was shown that using the temporal data, it is possible to train representation that is invariant to such confusing features.

3.10 Conclusion

We have described the internal representation of the Google Street View and its analogy to a directional graph, and how this graph can be fetched. We have shown how to get a panorama metadata, panorama images and stitch them together, where to find and how to interpret the most relevant information about the panorama locations and its neighbors. Finally, we have shown how to generate panorama images, depth maps and how to generate perspective images and perspective depth maps.

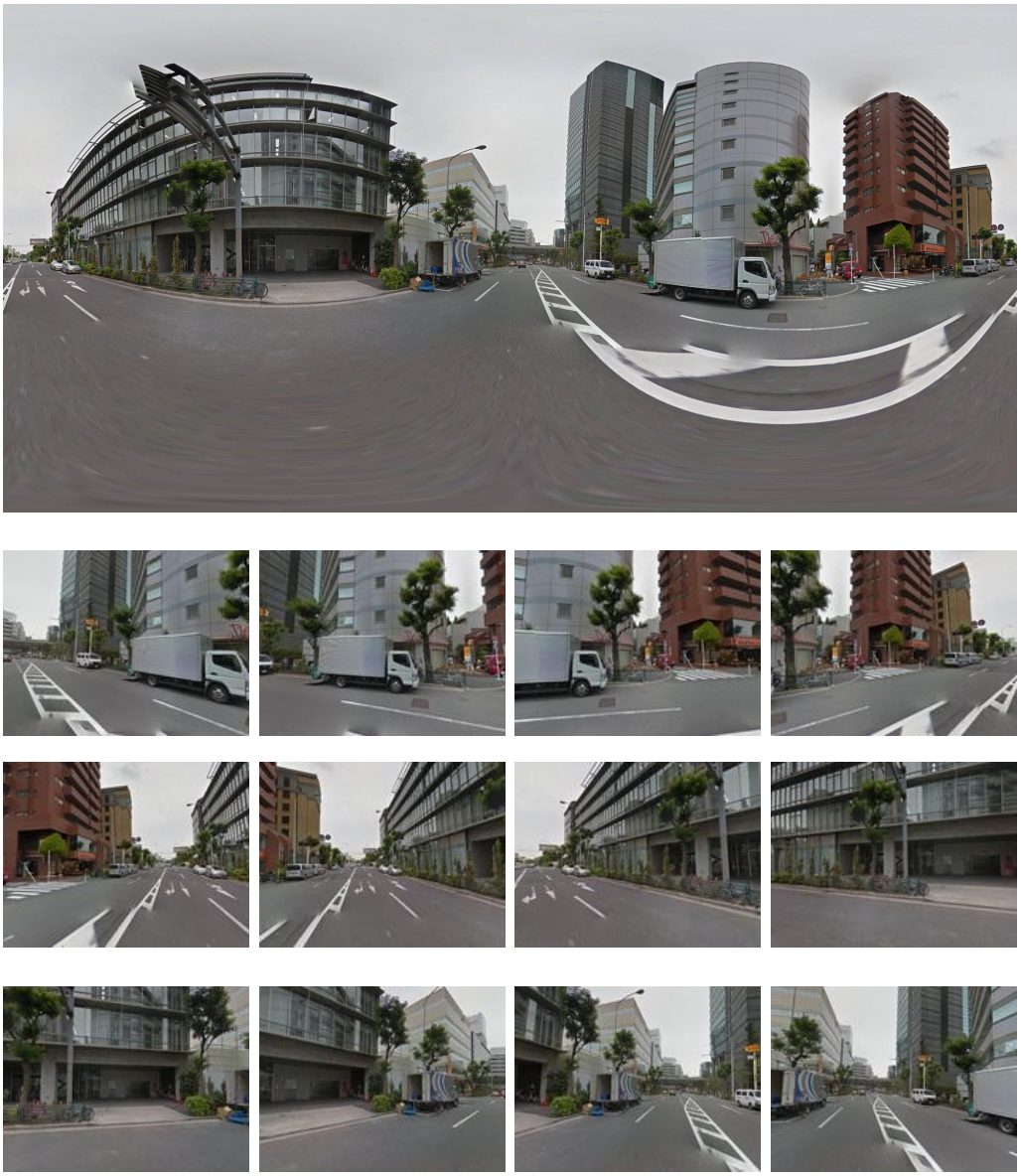


Figure 3.9: **Panorama perspective cutouts.** Twelve perspective cutouts per 360° each having 90° horizontal field of view and pitch 4°.

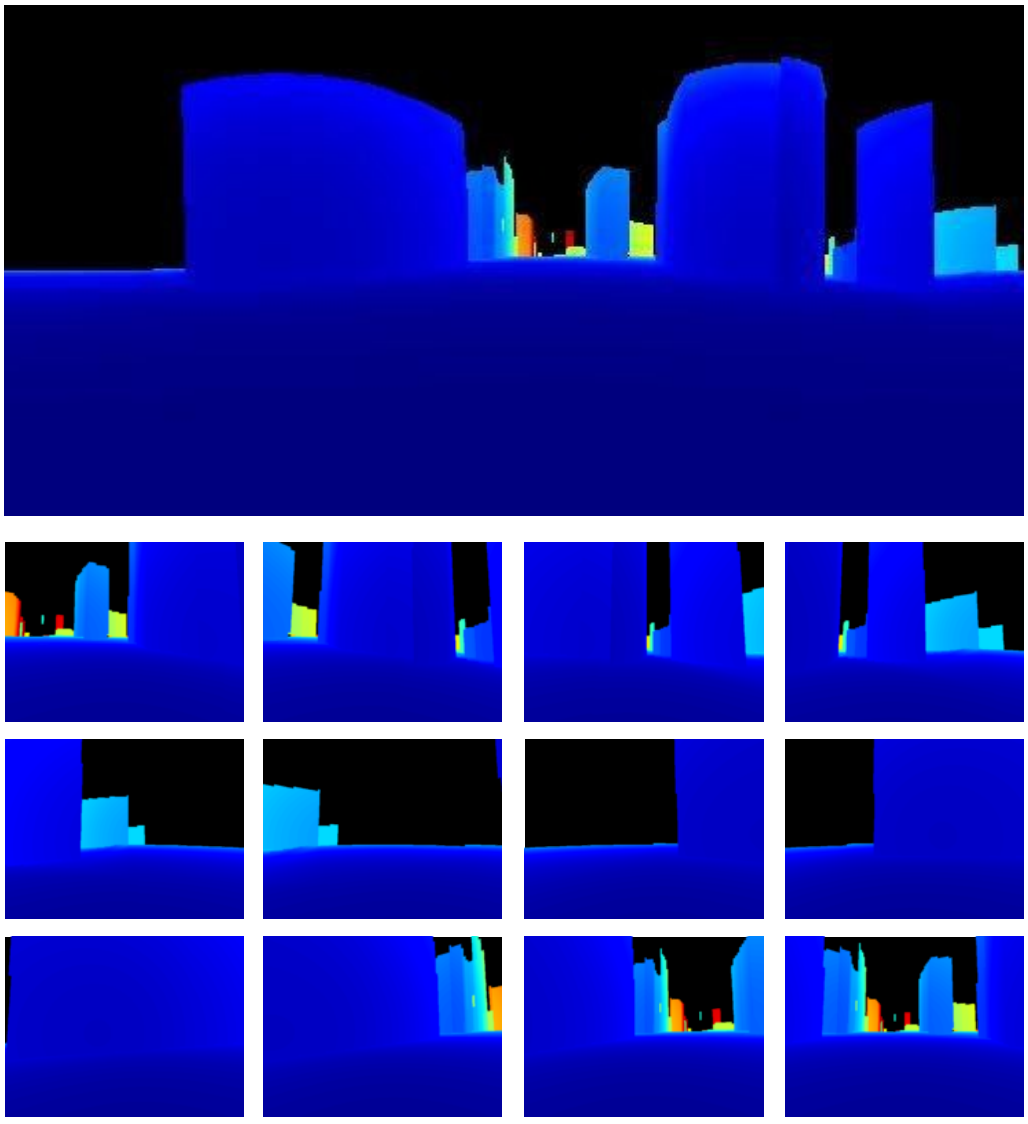


Figure 3.10: **Depth map perspective cutouts.** Equirectangular depth map (top) and twelve perspective cutouts (bottom) per 360° each having 90° horizontal field of view and pitch 4°. All images were generated from the label matrix and plane parameters.

4 Per-location classifiers

"The more original a discovery, the more obvious it seems afterwards."

— Arthur Koestler

In the previous chapters we explained recent challenges in place recognition [13, 39, 56, 66], reviewed different approaches [32, 41, 42, 8, 48, 67, 65, 39, 56, 69] and described how to build a datasets for place recognition (Chapter 3). Here, we formulate place recognition as a classification task, and train per-location classifier for each location stored in the database utilizing linear per-exemplar support vector machine (e-SVM). Finally, we show that each learned per-location classifier must be calibrated and we propose two calibration methods, one is well suited for bag-of-words image representations while the latter one works well with Fisher vectors.

4.1 Per-location classifiers for place recognition

We are given an image descriptor $\mathbf{x}_j$, one for each database image j. This representation can be a sparse tf-idf weighted bag-of-visual-words vector [61] or a dense compact descriptor such as the Fisher vector (FV) [36]. The goal is to learn a score f_j for each database image j, so that, at test time, given the descriptor $\mathbf{q}$ of the query image, we can either retrieve the correct target image as the image j^* with the highest score

$$j^* = \arg\max_j f_j(\mathbf{q}) \tag{4.1}$$

or use these scores to rank candidate images and use geometric verification to identify the correct location in an n-best list. Instead of approaching the problem directly as a large multiclass classification problem, we tackle the problem by learning a per-exemplar linear SVM classifier [44] for each database image j. Similar to [39], we use the available geotags to construct the negative set $\mathcal{N}_j$ for each image j. The negative set is constructed so as to concentrate difficult negative examples, i.e. from images that are far away from the location of image j and similar to the target image as measured by the dot product between their feature vectors. The details of the construction procedure will be given in section 5.1. The positive set $\mathcal{P}_j$ is represented by e single positive example, which is $\mathbf{x}_j$ itself. Each SVM classifier produces a score s_j which is a priori not comparable with the score of the other classifiers. A calibration of these scores will therefore be key to convert them to comparable scores f_j. This calibration problem is more difficult than usual given that we only have a single positive example and will be addressed in section 4.2.

4.1.1 Learning per-location SVM classifiers

Each linear SVM classifier generates a score s_j of the form

$$s_j(\mathbf{q}) = \mathbf{q}^T \mathbf{w}_j + b_j \tag{4.2}$$

where $\mathbf{w}_j$ is a weight vector re-weighting contributions of individual visual words and b_j is the bias specific for image j. Given the training sets $\mathcal{P}_j$ and $\mathcal{N}_j$, the aim is to find a vector $\mathbf{w}_j$ and bias b_j such that the score difference between $\mathbf{x_j}$ and the closest neighbor from its negative set $\mathcal{N}_j$ is *maximized*. Learning the weight vector $\mathbf{w}_j$ and bias b_j is formulated as a minimization of the convex objective

$$\begin{aligned}\Omega(\mathbf{w}_j, b_j) = ||\mathbf{w}_j||^2 + C_1 \sum_{\mathbf{x} \in \mathcal{P}_j} h(\mathbf{w}_j^T \mathbf{x} + b_j) \\ + C_2 \sum_{\mathbf{x} \in \mathcal{N}_j} h(-\mathbf{w}_j^T \mathbf{x} - b_j),\end{aligned} \tag{4.3}$$

where the first term is the regularizer, the second term is the loss on the positive training data weighted by scalar parameter C_1, and the third term is the loss on the negative training data weighted by scalar parameter C_2. This is a standard SVM [7] formulation (4.3), also used in exemplar-SVM [44]. In our case h is the squared hinge loss, which we found to work better in our setting than the standard hinge-loss. Parameters $\mathbf{w}_j$ and b_j are learned separately for each database image j in turn.

4.1.2 The need for calibrating classifier scores

Since the classification scores s_j are learned independently for each location j, they cannot be directly used for place recognition as in eq. (4.1). As illustrated in figure 4.1, for a given query $\mathbf{q}$, a classifier from an incorrect location (b) can have a higher score (eq. (4.2)) than the classifier from the target location (a). Indeed, the SVM score is a signed distance from the discriminating hyperplane and is a priori not comparable between different classifiers. This issue is addressed by calibrating scores of the learned classifiers. The goal of the calibration is to convert the output of each classifier into a probability (or in general a “universal" score), which can be meaningfully compared across classifiers. In the following two sections we develop two classifier calibration methods that do not need additional labelled positive examples.

4.2 Non-parametric calibration of the SVM-scores from negative examples only

In this section we describe a classifier calibration method that exploits the availability of large amounts of negative data, i.e. images from other far away locations in the database. In particular, the method estimates the significance of the score of a test example compared to the typical score of the (plentifully available) negative examples. Intuitively, we will use a large dataset of negative examples to calibrate the individual classifiers so that they *reject the same number of negative examples* at each level of the calibrated score. We will expand this idea in detail using the concepts from hypothesis testing.

4.2.1 Calibration *via* significance levels

In the following, we view the problem of deciding whether a query image matches a given location based on the corresponding SVM score as a hypothesis testing problem. In particular, we appeal to ideas from the traditional frequentist hypothesis testing framework also known as Neyman-Pearson (NP) framework (see e.g. [9], chap. 8).

We define the null hypothesis as $H_0 = \{\text{the image is a random image}\}$ and the alternative as $H_1 = \{\text{the image matches the particular location}\}$. The NP framework focuses on the case where the distribution of the data under H_0 is well known, whereas the distribution under H_1 is not accessible or too complicated to model, which matches perfectly our setting.

In the NP framework, the *significance level* of a score is measured by the p-value or equivalently by the value of the cumulative density function (cdf) of the distribution of the negatives at a given score value. The cdf is the function F_0 defined by $F_0(s) = \mathbb{P}(S_0 \leq s)$, where S_0 is a random variable corresponding to the scores of negative data (see figure 4.2 for an illustration of the relation between the cdf and the density of the function). The cdf (or the corresponding p-value[1]) is naturally estimated by the empirical cumulative density function $\hat{F}_0$, which is computed as:

$$\hat{F}_0(s) = \frac{1}{N_c} \sum_{n=1}^{N_c} 1_{\{s_n \leq s\}}, \tag{4.4}$$

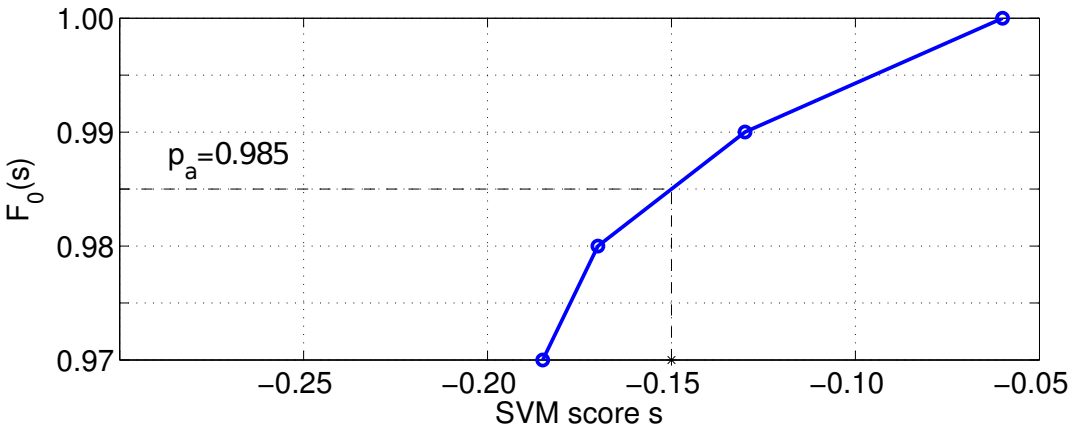
where $(s_n)_{1 \leq n \leq N_c}$ are the SVM scores associated with N_c negative examples used for calibration. Note that no positive examples are involved in the construction of the cumulative density function. $\hat{F}_0(s)$ is the fraction of the negative examples used for calibration (ideally held out negative examples) that have a score below a given value s. Computing $\hat{F}_0$ exactly would require to store all the SVM scores for all the calibration data for all classifiers, so in practice, we only keep a fraction of the larger scores. We also interpolate the empirical cdf between consecutive datapoints so that instead of being a staircase function it is a continuous piecewise linear function such as illustrated in figure 4.1. Given a query, we first compute its SVM score s_q and then compute the calibrated probability $f(q) = \hat{F}_0(s_q)$. We obtain a similar calibrated probability $f_j(q)$ for each of the SVMs associated with each of the target locations, which can now be ranked.

Two other examples of score calibration functions are shown in figure 5.8 in section 5.2. Note that while figure 4.1 illustrates only few points on the cdf, the two plots in figure 5.8 show a complete cdf that contains on the order of $25k$ data points. Note also that the two cumulative density functions in figure 5.8 are similar but not identical.

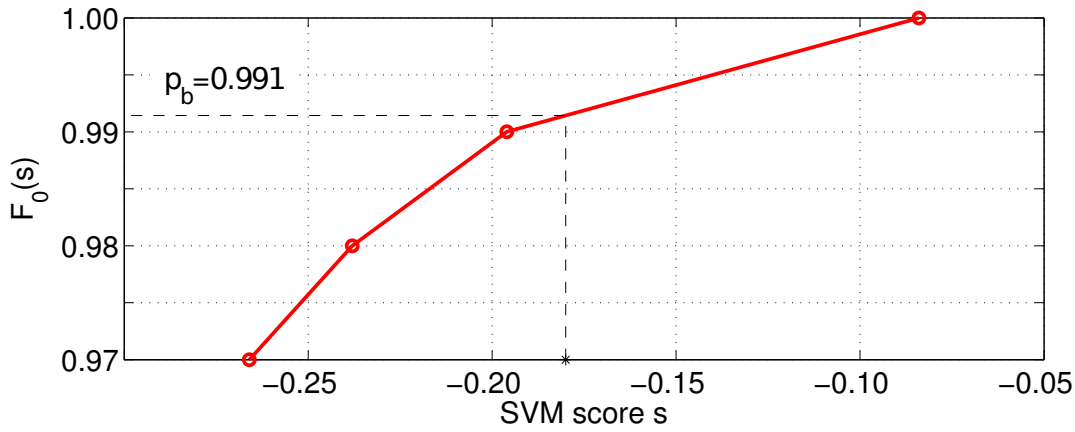
4.2.2 Summary of the calibration procedure

For each trained place-specific classifier s_j we construct the empirical cumulative density function (4.4) of scores of the negative examples and keep only its top K values. This can be done offline and the procedure is summarized in Algorithm 2. At query time, given a query image descriptor $\mathbf{q}$, we compute the uncalibrated classification score $s_j(\mathbf{q})$ and then use the stored cdf values to compute the calibrated score $f_j(\mathbf{q})$. This procedure is performed for each database image j and is summarized in Algorithm 3. Finally, the best candidate database image is selected

[1] The notion most commonly used in statistics is in fact the p-value. The p-value associated to a score is the quantity $\alpha(s)$ defined by $\alpha(s) = 1 - F_0(s)$; so the more significant the score is, the closer to 1 the cdf value is, and the closer to 0 the p-value is. To keep the presentation simple, we avoid the formulation in terms of p-values and we only talk of the probabilistic calibrated values obtained from the cdf F_0.

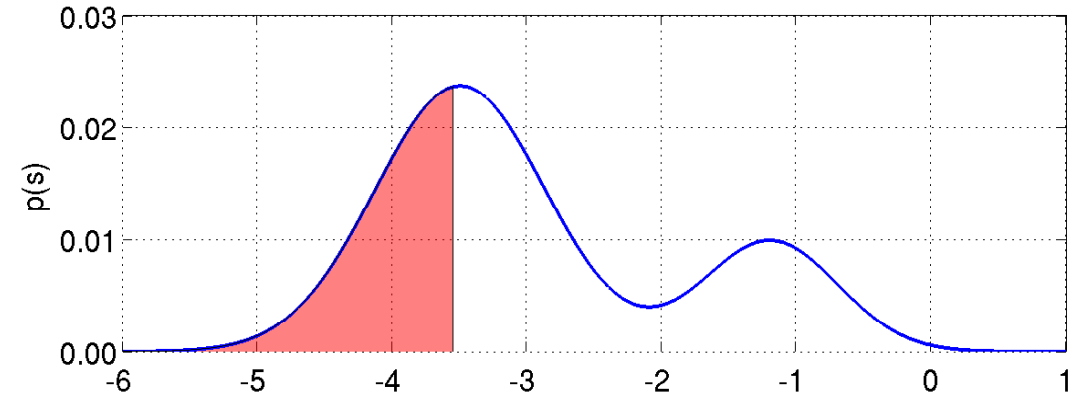


(a)

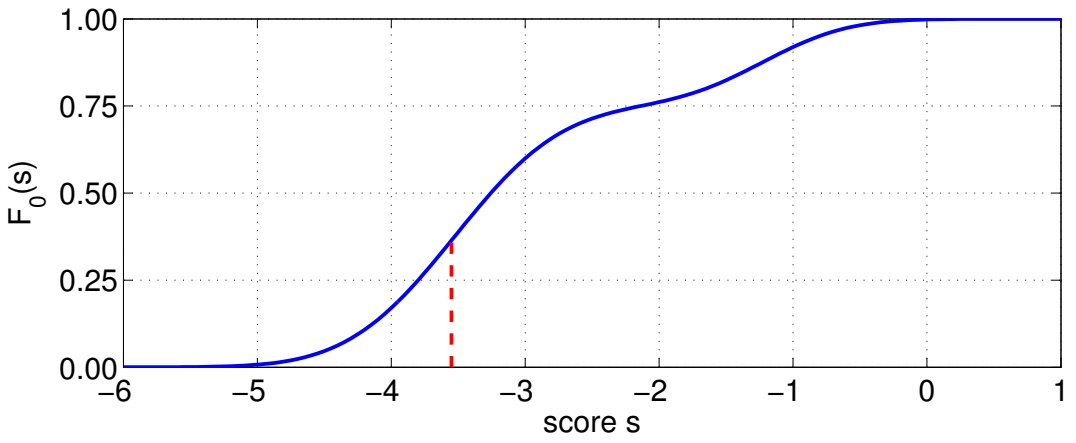


(b)

Figure 4.1: **An illustration of the proposed normalization of SVM scores for database images.** In each plot, the x-axis shows the raw SVM score. The y-axis shows the calibrated output. For the given query, the raw SVM score of image (b) is lower than for image (a), but the calibrated score of image (b) is higher than for image (a).



(a)



(b)

Figure 4.2: **Cumulative density function.** Illustration of the relation between (a) the probability density of the random variable S_0 modeling the scores of the negative examples and (b) the corresponding cumulative density function $F_0(s) = \mathbb{P}(S_0 \leq s)$.

by equation (4.1). Alternatively, candidate database images can be also ranked according to the calibrated score.

Algorithm 2 P-value calibration: offline stage

Input: $\mathcal{X}$ ... column wise matrix of image descriptors
$\mathbf{w}_j$, b_j ... learned SVM weights and biases
Output: $\hat{F}_{0j}$ ... calibration functions

procedure P-VALUE CALIBRATION
  $N \leftarrow$ database size
  $\mathcal{X} \leftarrow$ descriptor matrix of negative examples
  for $\forall j \in 1 \dots N$ **do**
    $N_c \leftarrow$ number of negative examples
    $\mathbf{w} \leftarrow$ learned SVM weight for image j
    $b \leftarrow$ learned SVM bias for image j
    $\sigma \leftarrow \mathbf{w}^T \mathcal{X} + b$
    Compute the cdf:
    $\mathbf{s_j} \leftarrow$ sorted σ in descending order
    $\hat{F}_{0j} \leftarrow [N_c \dots 0]/N_c$

4.2.3 Discussion

It should be noted that basing the calibration only on the negative data has the advantage that we privilege precision over recall, which is justified given the imbalance of the available training data (many more negatives than positives). Indeed, since we are learning with a single positive example, intuitively, we cannot guarantee that the learned partition of the space will generalize well to other positives, whose scores in the test set can potentially drop significantly. By contrast, since we are learning from a comparatively large number of negative examples, we can trust the fact that new negative examples will stay in the half-space containing the negative training set, so that their scores are very unlikely to be large. Our method is therefore based on the fact that we can measure reliably how surprising a high score would be if it was the score of a negative example. This exactly means that we can control false positives (type I error) reasonably well but not false negatives (type II error or equivalently the power of our test/classifier), exactly as in the Neyman-Pearson framework.

An additional reason for not relying on positive examples for the calibration in our case is that (even if we had sufficiently many of them) the positive examples that we collect using location and geometric verification from the geotagged database typically have illumination conditions that are extremely similar to each other and not representative of the distribution of test positives which can have very different illuminations. This is because of the controlled nature of the capturing process of geotagged street-level imagery (e.g. Google Street View) used for experiments in this work. Close-by images are typically captured at a similar time (e.g. on the same day) and under similar imaging conditions.

Scheirer et al. [55] propose a method, which is related to ours, and calibrate SVM scores by computing the corresponding cdf value of a Weibull distribution fitted to the top negative scores. The main difficulty is that the Weibull model should be fitted only to the tail of the distribution

Algorithm 3 P-value calibration: online stage

Input: $\mathbf{q}$ ... query image descriptor
$\mathbf{w}_j$, b_j ... learned SVM weights and biases
$\hat{F}_{0j}$ ... learned calibration function

Output: $f_j(\mathbf{q})$ ... calibrated score

procedure CALIBRATING SCORES
  $\mathbf{q} \leftarrow$ query image descriptor
  $N \leftarrow$ database size
  for $\forall j \in 1 \dots N$ **do** // for each database image
    $\mathbf{w} \leftarrow$ learned SVM weight for image j
    $b \leftarrow$ learned SVM bias for image j
    $\hat{F}_0 \leftarrow \hat{F}_{0j}$ // Empirical cdf
    $\mathbf{s} \leftarrow \mathbf{s_j}$ // Corresponding sorted scores
    $s_q \leftarrow \mathbf{q}^T\mathbf{w} + b$ // compute uncalibrated classifier score
    Find n such that $s_n \leq s_q < s_{n+1}$
    Compute the interpolated empirical cdf value:
    $\hat{F}_0(s_q) \approx \hat{F}_0(s_n) + \frac{s_q - s_n}{s_{n+1} - s_n}(\hat{F}_0(s_{n+1}) - \hat{F}_0(s_n))$.
    $f_j(\mathbf{q}) = \hat{F}_0(s_q)$ // output the calibrated score

of the negatives, which is in general difficult to identify. As a heuristic, Scheirer et al. propose to fit the Weibul model to false positives (i.e. the negative samples classified incorrectly as positives). But in our case, most of the exemplar SVMs that we are training have zero false positives in a held out set, which precludes the application of their method.

Finally, we should remark that we are not doing here calibration in the same sense of the word as the calibration based on logistic regression (or isotonic regression), since logistic regression estimates the probability of making a correct prediction by assigning a new data to class 1, while we are estimating how unlikely it would be for a negative example to have such a high score. The calibration with either methods yields “universal" scores in the sense that they are comparable from one SVM to another, but the calibrated values obtained from logistic regression are not comparable to the values obtained from our approach.

4.3 Affine calibration by normalizing the classification hyperplane

The non-parametric calibration method described in the previous section has two computational disadvantages, which make it hard to scale-up to very large datasets. First, the method requires storing the non-parametric model of the calibration function for each learned classifier. This has memory complexity of $O(NK)$, where N is the number of images (classifiers) in the database and K the number of stored elements of the non-parametric model. For typical values of $K = 1000$ and $N = 1\text{M}$ this would require additional 4GB of memory, comparable to the size of the inverted index itself. Second, computing the cumulative density function requires applying all N learned classifiers to the entire set of negative examples, which has also size N. As a result computing the cdf has complexity $O(N^2)$, which becomes quickly infeasible already for datasets with N larger than $100,000$.

To address these issues we first describe an affine calibration model that calibrates the classifier score with a simple linear function defined by only two parameters: its slope and offset, greatly reducing the required storage. Second, we show that the parameters of the affine calibration function can be obtained by normalizing the learned classification hyper-plane without applying the classifiers on the negative data and thus bringing down the computational complexity to $O(N)$. As a result, computing and storing the calibration functions becomes feasible for very large datasets with 1M images.

4.3.1 Affine calibration model

Using the affine calibration model we transform the uncalibrated score $s_j(\mathbf{q})$ of query $\mathbf{q}$ with a linear function

$$f_j(\mathbf{q}) = \alpha_j s_j(\mathbf{q}) + \beta_j, \tag{4.5}$$

where $f_j(\mathbf{q})$ is the output calibrated score, and α_j and β_j are scalar calibration parameters specific to each classifier j. In this work we use linear classifiers, hence substituting for $s_j(\mathbf{q})$ the linear classifier from (4.2) results also in a linear calibrated classifier

$$f_j(\mathbf{q}) = \widetilde{\mathbf{w}}_j^T \mathbf{q} + \widetilde{b}_j, \tag{4.6}$$

where $\widetilde{\mathbf{w}}_j = \alpha_j \mathbf{w}_j$ and $\widetilde{b}_j = \alpha_j b_j + \beta_j$. Note that the calibrated classifier (4.6) has the same form as the original classifier (4.2) and hence this representation does not require any additional storage compared to storing the original classifier. The question remains how to set the parameters α_j and β_j of the calibration function (4.5), which is discussed next.

4.3.2 Calibration by normalization

Parameters of the affine calibration function (4.5) could be learned from negative training data in a similar manner to, for example, [4]. We have tried to estimate the parameters in a similar manner by fitting a line to the tail of the cdf, however this procedure did not yield satisfactory results. In addition, as discussed above, in our case this requires running all N classifiers on all N images, which is prohibitive for large datasets. Instead, we have found that a good calibration can be obtained by normalizing the learned hyperplane $\mathbf{w}$. In particular, we set

$$\alpha_j = \frac{1}{||\mathbf{w}_j||}, \tag{4.7}$$

$$\beta_j = -b_j \alpha_j, \tag{4.8}$$

where $\mathbf{w}_j$ and b_j are the parameters of the learned SVM hyper-plane for location j and $||\mathbf{w}||$ is the L_2 norm of $\mathbf{w}$. Given this choice of α_j and β_j the calibrated classification score (4.6) reduces to

$$f_j(\mathbf{q}) = \frac{1}{||\mathbf{w}_j||} \mathbf{w}_j^T \mathbf{q} = \widetilde{\mathbf{w}}_j^T \mathbf{q}. \tag{4.9}$$

The intuition is that when $\mathbf{q}$ is L_2 normalized, equation (4.9) is equivalent to computing the normalized dot-product between vectors $\mathbf{q}$ and $\mathbf{w}$. This was found to work well in image retrieval [61] or matching whitened HOG descriptors [15]. In this work we investigate whether this intuition about descriptor matching can be used as a form of calibration for the learned place-specific classifier. Note that this form of calibration by normalization is scalable to very large datasets as it (i)

requires only $O(N)$ computations offline to pre-compute the calibration parameters for each of the N learned classifiers (equations (4.7) and (4.8)), and (ii) does not need any additional storage or computation at query time as the calibration parameters can be included in the classifier (4.6). In Appendix we examine the per-exemplar SVM cost and give an additional intuition why calibration by re-normalization works.

4.4 Memory efficient classifier representation

We learn a linear discriminative classifier with weight vector $\mathbf{w}_j$ and bias b_j for each image j in the database. These classifier parameters become the new representation for each image. In this section we discuss how the classifier parameters can be stored in a memory efficient manner that is amenable for indexing. The goal is to apply all the learned classifiers to the query descriptor $\mathbf{q}$

$$\mathbf{s} = \mathbf{q}^T \mathcal{W} + \mathbf{b}, \tag{4.10}$$

where $\mathcal{W}$ is $d \times N$ matrix storing all the learned $\mathbf{w}_j$ classifiers as columns, $\mathbf{b}$ is a $1 \times N$ vector storing all the learned bias values b_j, $\mathbf{q}$ is the input query descriptor, $\mathbf{s}$ is a $1 \times N$ vector of output scores for all classifiers in the database, N is the number of images in the database and d is the dimensionality of the image representation. As discussed in detail in section 5.1 we investigate two different image representations: (i) the compact Fisher vectors [36] and (ii) the bag-of-visual-word vectors [61]. The learned classifiers for these two image descriptors have different statistics and require different methods for storing and indexing. Next, we discuss the classifier representations for the two types of image representations.

Fisher vectors: The Fisher vector descriptors are not sparse, but have a relatively low-dimension $d \in \{128, 512, 2048\}$ hence it is possible to store directly the (non-sparse) matrix $\mathcal{W}$ containing the learned classifier parameters $\mathbf{w}$. In this work we exhaustively compute the classifier scores for all images in the database (given by equation (4.10)) using efficient (but exact) matrix-vector multiplication routines. However, this computation can be further sped-up using product quantization indexing as described in [34].

Bag-of-visual-words: In the bag-of-visual-words representation, each image is represented by a high dimensional vector $\mathbf{x}$, where the dimensionality d is typically $100,000$, but the vector is very sparse with only about 2,000 non-zero entries. The learned $\mathbf{w}_j$ are of the same (high) dimension d but are *not sparse*. As a result, directly storing the learned classifiers becomes quickly infeasible. To illustrate this, consider a database of $N = 1,000,000$ images. Storing the original descriptors with about 2,000 non-zero entries for each image would take around 8GB. However, directly storing the learned non-sparse $100,000 \times 1,000,000$ matrix $\mathcal{W}$ would require 400GB of memory. To address this issue we have developed an alternative indexing structure taking advantage of the dual form of the linear classifier as a sparse linear combination of a small number of support vectors [57]. The key observation is that the number of support vectors k is significantly lower then dimensionality d of the original image descriptor. In the following we omit index j for clarity. In detail, we represent each $\mathbf{w}$ by its corresponding coefficients α_i of the linear combination of the support vectors (individual image descriptors) $\mathbf{x}_i$ such that

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i = \mathcal{X} \cdot \alpha, \tag{4.11}$$

where α_i, the elements of vector α, are coefficients of the linear combination of the training data points $\mathbf{x}_i$ and the matrix $\mathcal{X}$ contains (as columns) descriptors of the entire database. Note that the vector α is sparse and the number of non-zero elements depends on the number of support vectors k.

As a result, matrix $\mathcal{W}$ containing all learned classifier weights can be expressed in the dual form as

$$\mathcal{W} = \mathcal{X}\mathcal{A}, \tag{4.12}$$

where $\mathcal{X}$ is the (sparse) matrix of the bag-of-visual-words image descriptors and $\mathcal{A}$ is the (sparse) matrix of α coefficients, where each column corresponds to vector α from (4.11). Instead of storing all (non-sparse) weight vectors $\mathcal{W}$, which has memory complexity $O(dN)$ where d (= 100,000) is the dimensionality of the image representation and N is the size of the database, we store two sparse matrices $\mathcal{X}$ and $\mathcal{A}$, which has memory complexity $O(mN + kN)$ where m (=2,000) is the number on non-zero elements in the original bag-of-visual-word descriptors, and k is the typical number of support vectors. In our case k is about the size of the training data which is around 500. As a result, the storage requirements are significantly reduced. For example, for a database of 1M images the dual representation requires only about 10 GB of storage compared to 400GB for directly storing classifiers $\mathcal{W}$. Note that sparsity can be imposed directly on the learned classifiers $\mathbf{w}$ by appropriate regularization [57]. However, we found this approach did not yield competitive results in terms of accuracy.

4.5 Intuition why calibration by normalization works

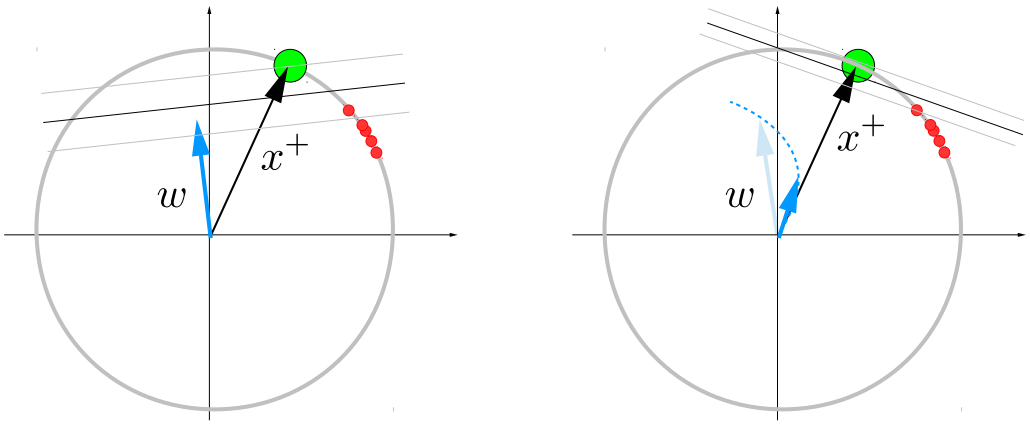
In section 5.2 we show that the simple calibration by normalization often results in surprisingly good place recognition performance without the need for any additional positive or negative calibration data. In this appendix, we give a possible explanation why this simple calibration works. We focus on the case of a single positive training example, i.e. when training set $\mathcal{P} = \mathbf{x}^+$, which is the typical case for place recognition where only one positive example is available for each place. The analysis holds also for the case of multiple expanded positive examples as in our case the positive examples are coming from the same database of Street View images, and hence have very similar statistics (illumination, capturing conditions, the same camera, etc.).

In particular, we first analyze the SVM objective and show that the learned hyperplane $\mathbf{w}$ can be interpreted as a new descriptor $\mathbf{x}^*$ that replaces the original positive example $\mathbf{x}^+$ and is re-weighted to increase its separation from the negative data. Second, we show that when $\mathbf{x}^*$ is normalized, i.e. $\mathbf{x}^* = \frac{\mathbf{w}}{||\mathbf{w}||}$, the dot-product $\mathbf{q}^T\mathbf{x}^*$ corresponds to measuring the cosine of the angle between the (normalized) query descriptor $\mathbf{q}$ and the new descriptor $\mathbf{x}^*$, which was found to work well in the literature for descriptor matching, as discussed in section 4.3.2. The two steps are given next.

4.5.1 Analysis of per-exemplar SVM objective

For a single positive example $\mathcal{P} = \mathbf{x}^+$, the per-exemplar SVM objective (4.3) can be written as

$$\begin{aligned}\Omega(\mathbf{w}, b) = ||\mathbf{w}||^2 + C_1 \cdot h(\mathbf{w}^T\mathbf{x}^+ + b) \\ + C_2 \sum_{\mathbf{x}\in\mathcal{N}} h(-\mathbf{w}^T\mathbf{x} - b).\end{aligned} \tag{4.13}$$



(a) $C_2 > 0$ (b) $C_2 \to 0$

Figure 4.3: **An illustration of the effect of decreasing parameter C_2 in the exemplar support vector machine objective.** The positive exemplar $\mathbf{x}^+$ is shown in green. The negative data points are shown in red. All training data is L2 normalized to lie on a hyper-sphere. (a) For $C_2 > 0$, the normal $\mathbf{w}$ of the optimal hyper-plane moves away from the direction given by the positive example $\mathbf{x}^+$ in a manner that reduces the loss on the negative data. (b) As the parameter C_2 decreases the learned $\mathbf{w}$ becomes parallel to the positive training example $\mathbf{x}^+$ and its magnitude $||\mathbf{w}||$ goes to 0.

In the following, we analyze the objective (4.13) and provide intuition why *re-normalized* weight vector $\mathbf{w}$ can be interpreted as a new descriptor. In particular, we show first that when the weight C_2 of the negative data in objective (4.13) goes to zero the learned normalized $\widetilde{\mathbf{w}}$ is identical to the original positive training data point $\mathbf{x}^+$. Second, when $C_2 > 0$, the learned vector $\widetilde{\mathbf{w}}$ moves away from the positive vector $\mathbf{x}^+$ to increase its separation from the negative data. The two cases are detailed next.

Case I: $C_2 \to 0$. The goal is to show that when the weight C_2 of the negative data in objective (4.13) goes towards zero, the resulting hyperplane vector $\mathbf{w}$ is parallel with the vector of positive training descriptor $\mathbf{x}^+$. When $\mathbf{w}$ is normalized to have unit L2 norm the two vectors are identical. First, let us decompose $\mathbf{w}$ into parallel and orthogonal part with respect to the positive training data point $\mathbf{x}^+$, i.e. $w = \mathbf{w}^\perp + \mathbf{w}^{||}$, where $(\mathbf{w}^\perp)^T\mathbf{x}^+ = 0$. Next, we observe that when the weight of the negative data diminishes ($C_2 \to 0$), any non-zero component $\mathbf{w}^\perp$ will increase the value of the objective. As a result, for $C_2 \to 0$ the objective is minimized by $\mathbf{w}^{||}$, i.e. the optimal $\mathbf{w}$ is parallel with $\mathbf{x}^+$.

In detail, for $\mathbf{w} = \mathbf{w}^\perp + \mathbf{w}^{||}$, the objective (4.3) can be written as

$$||\mathbf{w}^\perp + \mathbf{w}^{||}||^2 + C_1 \cdot h\left((\mathbf{w}^\perp + \mathbf{w}^{||})^T\mathbf{x}^+ + b\right) + C_2 \sum_{\mathbf{x}\in\mathcal{N}} h\left(-(\mathbf{w}^\perp + \mathbf{w}^{||})^T\mathbf{x} - b\right). \tag{4.14}$$

Note that the orthogonal part $\mathbf{w}^\perp$ does not change the value of the second term in (4.14) because

$(\mathbf{w}^{\perp} + \mathbf{w}^{||})^T\mathbf{x}^+ = (\mathbf{w}^{||})^T\mathbf{x}^+$, and hence (4.14) reduces to

$$||\mathbf{w}^{\perp} + \mathbf{w}^{||}||^2 + C_1 \cdot h\left(\mathbf{w}^{||T}\mathbf{x}^+ + b_j\right) + C_2 \sum_{\mathbf{x}\in\mathcal{N}} h\left(-(\mathbf{w}^{\perp} + \mathbf{w}^{||})^T\mathbf{x} - b\right). \tag{4.15}$$

In the limit case as $C_2 \to 0$ any non-zero component $\mathbf{w}^{\perp}$ will increase the value of the objective (4.15). This can be seen by noting that the third term vanishes when $C_2 \to 0$ and hence the objective is dominated by the first two terms. Further, the second term in (4.15) is independent of $\mathbf{w}^{\perp}$. Finally, the first term will always increase for any non-zero value of $w^{\perp}$ as $||\mathbf{w}^{\perp}+\mathbf{w}^{||}||^2 \geq ||\mathbf{w}^{||}||$ for any $\mathbf{w}^{\perp} \neq 0$.

As a result, in the limit case when $C_2 \to 0$ the optimal $\mathbf{w}$ is parallel with $\mathbf{x}^+$. Note also, that when C_2 is exactly equal to zero, $C_2 = 0$, the optimal $\mathbf{w}$ vanishes, i.e. the objective (4.15) is minimized by trivial solution $||\mathbf{w}|| = 0$ and $b = -1$. The effect of decreasing the parameter C_2 is illustrated in figure 4.3.

Case II: $C_2 > 0$**.** When the weight C_2 of the negative data in the objective (4.15) increases the direction of the optimal w will be different from $\mathbf{w}^{||}$ and will change to take into account the loss on the negative data points. Explicitly writing the hinge-loss $h(x) = \max(1-x, 0)$ in the last term of (4.15), we see that $\mathbf{w}$ will move in the direction that reduces $\sum_{\mathbf{x}\in\mathcal{N}} \max\left(1 + \mathbf{w}^T x + b, 0\right)$, i.e. that reduces the dot product $\mathbf{w}^T\mathbf{x}$ on the negative examples that are active (support vectors).

4.5.2 The need for normalization

Above we have shown that the learned hyperplane $\mathbf{w}$ moves away from the positive example $\mathbf{x}^+$ in a manner that reduces the loss on the negative data. The aim is to use this learned vector $\mathbf{w}$ as a new descriptor $\mathbf{x}^*$ replacing the original positive example $\mathbf{x}^+$. However, we wish to measure the cosine of the angle between the the new descriptor $\mathbf{x}^*$and the query image $\mathbf{q}$. This is equivalent to the normalized dot product, hence the vector $\mathbf{w}$ needs to be normalized.

5 Experiments

"There's a fine line between fishing and just standing on the shore like an idiot."

— Steven Wright

THUS far, in chapter 3 we discussed how to build geo-referenced datasets for place recognition. In chapter 4 we hereafter proposed per-location classifiers for place recognition and showed that calibration of the trained classifiers is critical. It was shown how to train the classifiers utilizing exemplar-SVM, and two calibration methods have been proposed, one suitable for BOW image representation, the latter for Fisher vectors image representation.

In this chapter, we present experiments on three geo-referenced datasets collected by the earlier version of the `streetget` package presented in chpater 3. First, we describe datasets and implementation details of performed experiments. Second, we discuss the results, show how proposed method performs over the baselines and provide some intuition about the trained classifiers by visualizing learned weights. Finally, we discuss improvement and failure cases, and scalability of the method.

5.1 Experimental setup and implementation details

In this section we first describe the geo-referenced datasets used for the experiments. Then we outline local feature descriptors and the two types of used image descriptors, namely the bag-of-visual-word (BOW) and the Fisher vector (FV), and finally give implementation details of the classifier and calibration function learning procedure.

5.1.1 Image datasets

Experiments are performed on datasets collected by early version of the `streetget` package described in chapter 3. There are two versions of the Pittsburgh dataset [27] that differ in size, and challenging Tokyo 24/7 dataset [64]. The datasets and its query sets are described next.

Pittsburgh dataset. The first dataset contains Google Street View panoramas downloaded from the Internet covering an area of 1.3×1.2 km^2 of the city of Pittsburgh (U.S.). Similar to [10], we generate for each panorama 12 overlapping perspective views corresponding to two different elevation angles 4° and 28° to capture both the street-level scene and the building façades. This results in a total of 24 perspective views each with 90° FOV and resolution of 960×720 pixels. In this manner, we generate two versions of this dataset. The first version is generated in the same manner as [65], it contains the panoramas within the perimeter of the $50m$ from the query images

Figure 5.1: **Example query images from the Pittsburgh dataset.** The first row shows a sample of the query images from the Pittsburgh Google Street View research dataset [21]. The second row contains corresponding ground truth database images from the database. Notice changes in the camera viewpoint, illumination conditions, occlusion and change of the urban environment over time.

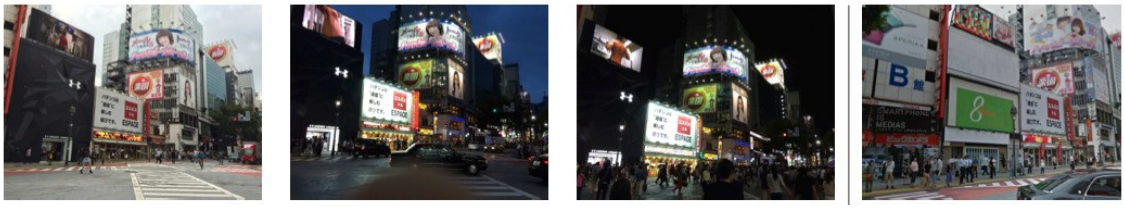
(described below) resulting in the dataset of the size $25k$. The latter includes all panoramas in the area and contains $55k$ images.

As a query set with known ground truth GPS positions, we use 8999 panoramas from the Google Street View research dataset [21], which cover approximately the same area, but were captured at a different time, and typically depict the same places from different viewpoints and under different illumination conditions. We generate a test query set such that we first select a panorama at random, and second, we generate a perspective image with a random orientation and random elevation pitch. This way we synthesize 4,000 query test images. Both the query and database images are available upon request at [24]. Examples of the query images and database images are shown in figure 5.1.

24/7 Tokyo dataset. The 24/7 Tokyo dataset [64] contains Google Street View panoramas downloaded from the Internet covering an area of $1.6 \times 1.6\,\mathrm{km}^2$ of the city of Tokyo. The dataset contains 76k perspective views. The query set contains 315 query images from 105 distinct locations captured by different types of camera phones. This query dataset is very challenging as each location is captured at three different times: during a day, at sunset and during night. The dataset is available upon request at [63]. Examples of the query images and database images are shown in figure 5.2.

5.1.2 Image descriptors

We perform experiments with two types of image descriptors: the sparse high-dimensional bag-of-visual-word vectors [61] and the compact (not-sparse) Fisher vectors [36]. Details of each are given



Query (a) Query (b) Query (c) Database image (d)

Figure 5.2: **Example query images from the 24/7 Tokyo dataset.** Each place in the query set is captured at different times of day: (a) daytime, (b) sunset, and (c) night. For comparison, the database street-view image at a close-by position is shown in (d). Note the major changes in appearance (illumination changes in the scene) between the database image (d) and the query images (a,b,c). (Courtesy of Akihiko Torii.)

next.

Bag-of-visual-word representation. We extract SURF descriptors [6] for each image and learn a vocabulary of 100k visual words by approximate k-means clustering [47] from a subset of features from 5,000 randomly selected database images. Then, a tf-idf weighted vector [61] is computed for each image by assigning each descriptor to the nearest cluster center. Finally, all database vectors are normalized to have unit L_2 norm.

Fisher vectors. Following [36] we project the extracted 128-dimensional rootSIFT [3] descriptors to 64 dimensions using PCA. Default parameters have been used to extract the SIFT descriptors. The projection matrix is learned on a set of descriptors from 5,000 randomly selected database images. This has also the effect of decorrelating the rootSIFT descriptor. The 64-dimensional descriptors are then aggregated into Fisher vectors using a Gaussian mixture model with $N = 256$ components, which results in a $2 \times 256 \times 64 = 32,768$-dimensional descriptor for each image. The Gaussian mixture model is learned from descriptors extracted from 5,000 randomly sampled database images. The high-dimensional Fisher vector descriptors are then projected down to dimension using PCA learned from all available images in the database. The resulting low-dimensional Fisher vectors are then normalized to have unit L2-norm, which we found to be important in practice.

5.1.3 Parameters of per-location classifier learning

To learn the exemplar support vector machine for each database image j, the positive and negative training data are constructed as follows. The *negative training set* $\mathcal{N}_j$ is obtained by: (i) finding the set of images with geographical distance greater than 200 m; (ii) sorting the images by decreasing value of similarity to image j measured by the dot product between their respective descriptors (BOW of FV); (iii) taking the top $N = 500$ ranked images as the negative set. This is illustrated in figure 5.3. In other words, the negative training data consists of the hard negative images, i.e. those that are similar to image j but are far away from its geographical position, hence, cannot have the same visual content. The *positive training set* $\mathcal{P}_j$ consist of the descriptor $\mathbf{x}_j$ of the target image j.

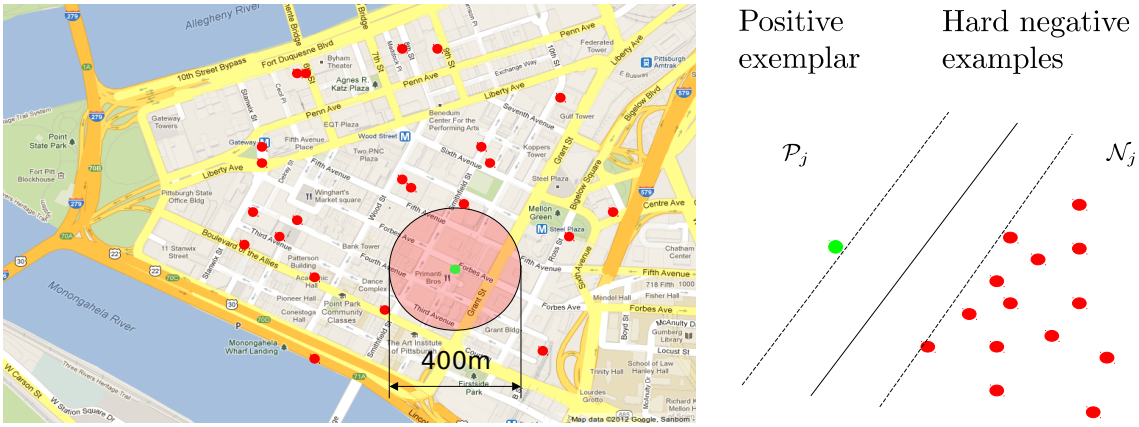


Figure 5.3: **Per-location classifier training data.** (left) The positive training set $\mathcal{P}_j$ consist of the descriptor of the target image j. The negative training set $\mathcal{N}_j$ consist of hard negative examples that are geographically further than 200m from the target image j. (right) An illustration of learned exemplar-SVM hyperplane with margin.

We found that for the bag-of-visual-words representation it was useful to further expand [11] positive training set by close by images that view the same scene structures. These images can be identified by geometric verification [47] as follows. We first build a graph where each image in the database represents a node and an edge represents a spatial adjacency in the world. An edge is present if the positions of the two images are within $50m$ of each other. Then, we score each edge by the number of geometrically verified matches [47]. Finally, we remove edges with the score below a threshold of $t_m = 40$ matches. It is worth noting that the resulting graph contains many isolated nodes. This typically indicates that the viewpoint change between two adjacent panoramas is large. For each image in the database, we include between zero and five extra positive examples that are directly connected in the graph.

For the support vector machine classifier (SVM) training we use the `libsvm` [17] library. The same C_1 and C_2 parameters are used for all per-exemplar classifiers, but find the optimal value of the parameters for each image representation by a cross-validation evaluating performance on a held out query set.

For the calibration by re-normalization, we L_2 normalize the learned $\mathbf{w}_j$ using equation (4.9) and use this normalized vector as the new image descriptor $\mathbf{x}'_j$ for image j. At query time we compute the descriptor $\mathbf{q}$ of the query image and measure its similarity score to the learned descriptors $\mathbf{x}'_j$ for each database image by equation (4.1).

For the p-value calibration, we take the learned classifier for each database image j and compute its SVM score for all other database images to construct its empirical cumulative density function (4.4). We keep only the top 1,000 values that, in turn, represent the calibration function. At query time, given the query descriptor $\mathbf{q}$, we compute the SVM score (4.2) for each database image j, and compute its calibrated SVM score f_j (4.4).

5.2 Results

We evaluate the proposed per-location classifier learning approach on two different image descriptors: the bag-of-visual-words model (section 5.2.1) and Fisher vectors (section 5.2.2). We also compare the recognition accuracy of the two learned representations relative to their compactness measured by their memory footprint (section 5.2.3). Finally, we compare results to linear discriminant analysis (LDA) and whitening baselines (section 5.2.4), outline the main failure modes (section 5.2.5) and discuss the scalability of our method (section 5.2.6). Since the ground truth GPS position of each query image is available, for each method we measure performance using the percentage of correctly recognized queries (Recall) similarly to, e.g., [10, 39, 54]. We deem the query as correctly localized if at least one of the top K retrieved database images is within 20 meters from the ground truth position of the query.

5.2.1 Bag-of-visual-words model

Results for the bag-of-visual-words image representation are shown in table 5.1. Learning per-location classifiers with either calibration method (*p-val* and *w-norm*) clearly improves over the standard bag-of-visual-words baseline (BOW) that does not perform any learning. In addition, both calibration methods significantly improve over the learned SVM classifiers without any calibration (BOW SVM no calib) underscoring the importance of calibration for the independently learned per-location classifiers. In table 5.1, we also compare performance to our implementation of the confuser suppression approach (Conf. supp.) of [39] that, in each database image, detects and removes features that frequently appear at other far-away locations (using parameters $t = 3.5$ and $w = 70$). The results show an improvement by our method, especially at recall@1.

Inspecting the detailed plots in figure 5.4 we further note that the *p-val* calibration performs slightly better than the *w-norm* calibration for shorter top K shortlists but this effect is reversed for larger K. This could be attributed to the fact that the *p-val* calibration uses the negative data to control false positive errors, but has less control over false negatives, as discussed in section 4.2.3.

In figure 5.8 we visualize the learned SVM weights on BOW for *p-val*. We visualize the contribution of each feature to the SVM score for the corresponding query image. Red circles represent features with negative weights while green circles correspond to features with positive weights. The area of each circle is proportional to the contribution of the corresponding feature to the SVM score. For instance for the left figure notice that the correctly localized queries (c) contain more green colored features than queries from other places (b) and (a). Query (b) gets a high score because the building has orange and white stripes similar to the sun-blinds of the bakery, which are features that also have large positive weights in the query image (c) of the correct place. In the top row, we visualize the calibration of raw SVM score for three different queries. The calibration function of the target image j is shown in the blue and the corresponding SVM scores of the three queries are denoted by red circles. Notice that both images (b) and (c) have high calibrated score even their respective SVM score was different.

In figures 5.9 - 5.11 at the end of this section we show examples of query images correctly and incorrectly localized by our and the baseline methods. Finally, examples of correctly and incorrectly localized queries are shown in figure 5.16 at he end of this chapter.

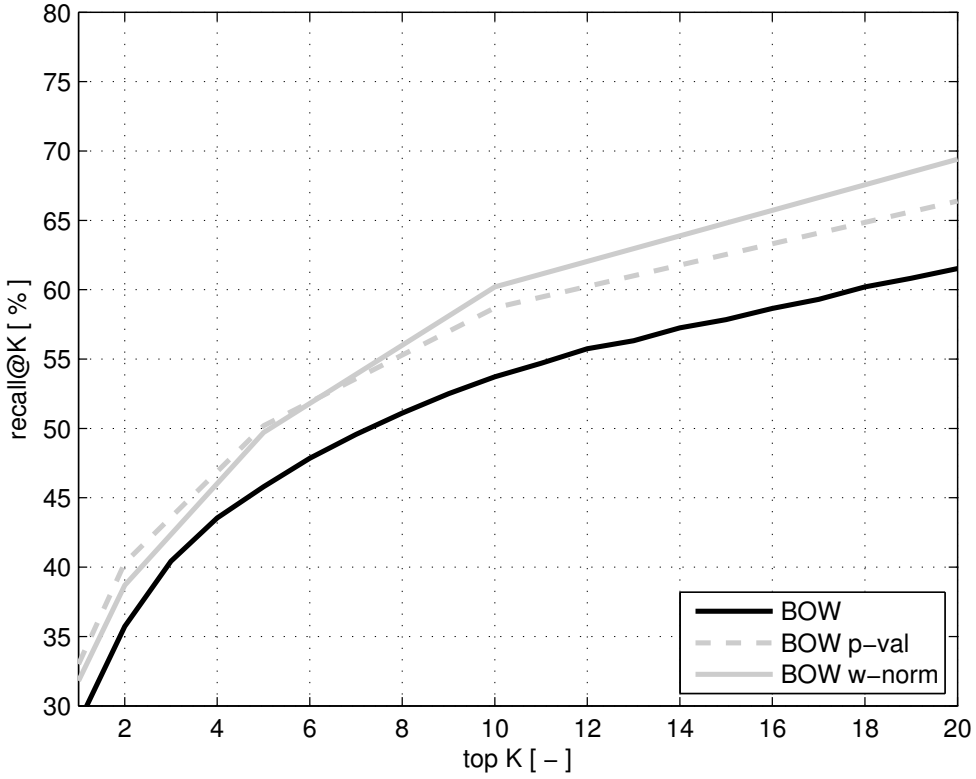


Figure 5.4: **Evaluation of the learned bag-of-visual-words representation on the Pittsburgh 25k [27] dataset.** The graph shows the fraction of correctly recognized queries (recall@K, y-axis) vs. the number of top K retrieved database images for the raw bag-of-visual-words baseline (BOW) and the learned representation with two different calibration methods (p-val and w-norm).

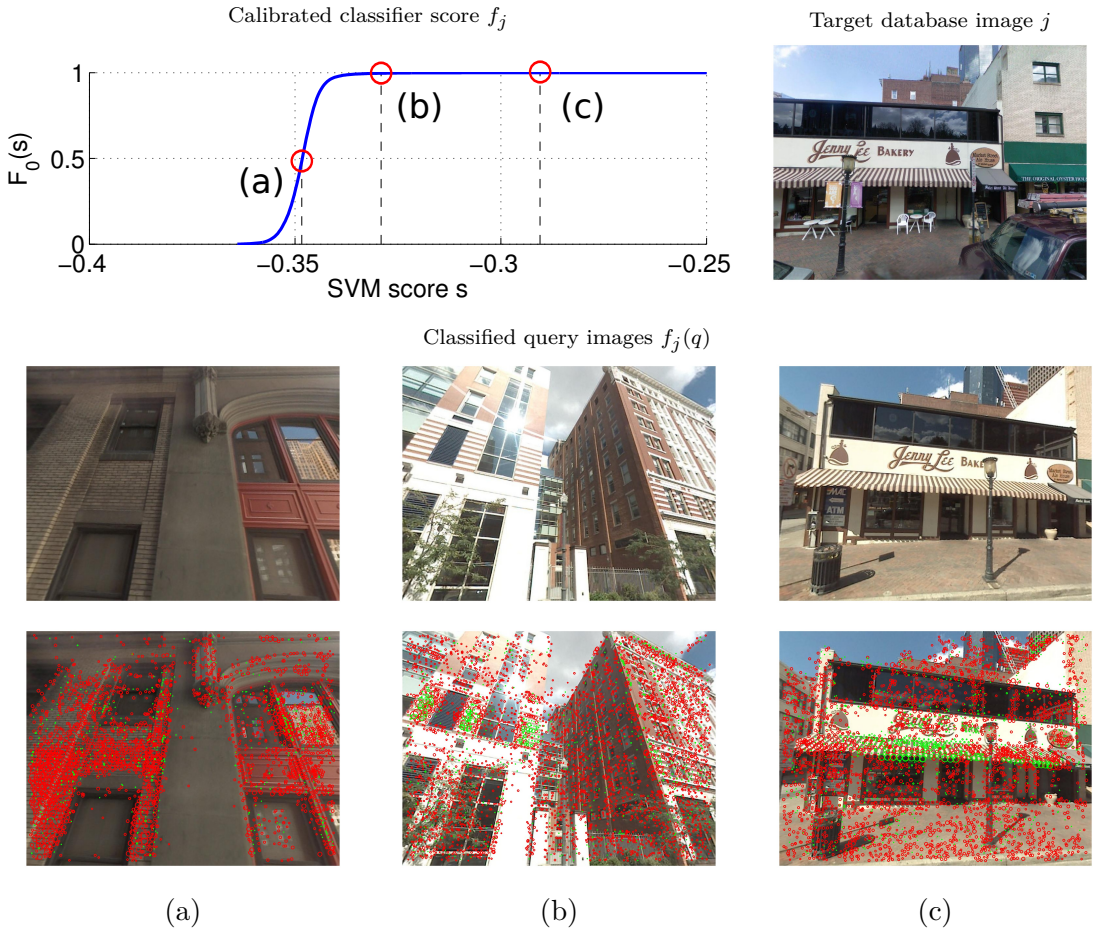


Figure 5.5: **A visualization of learned feature weights for two database images. In each panel:** *first row:* (Right) Target database image j. (Left) Cumulative density function (or calibrated score) learned for the SVM scores of the corresponding classifier f_j; three query images displayed on the *second row* are represented by their SVM scores and cdf values $F_0(s)$, denoted (a)-(c) on the graph. *Third row:* A visualization of the contribution of each feature to the SVM score for the corresponding query image. Red circles represent features with negative weights while green circles correspond to features with positive weights. The area of each circle is proportional to the contribution of the corresponding feature to the SVM score. Notice that the correctly localized queries (c) contain more green colored features than queries from other places (b) and (a). Please also note that the calibration *cdf*s in the left and right panel are similar but not identical.
Query (b) gets a high score because the building has orange and white stripes similar to the sun-blinds of the bakery, which are features that also have large positive weights in the query image (c) of the correct place.

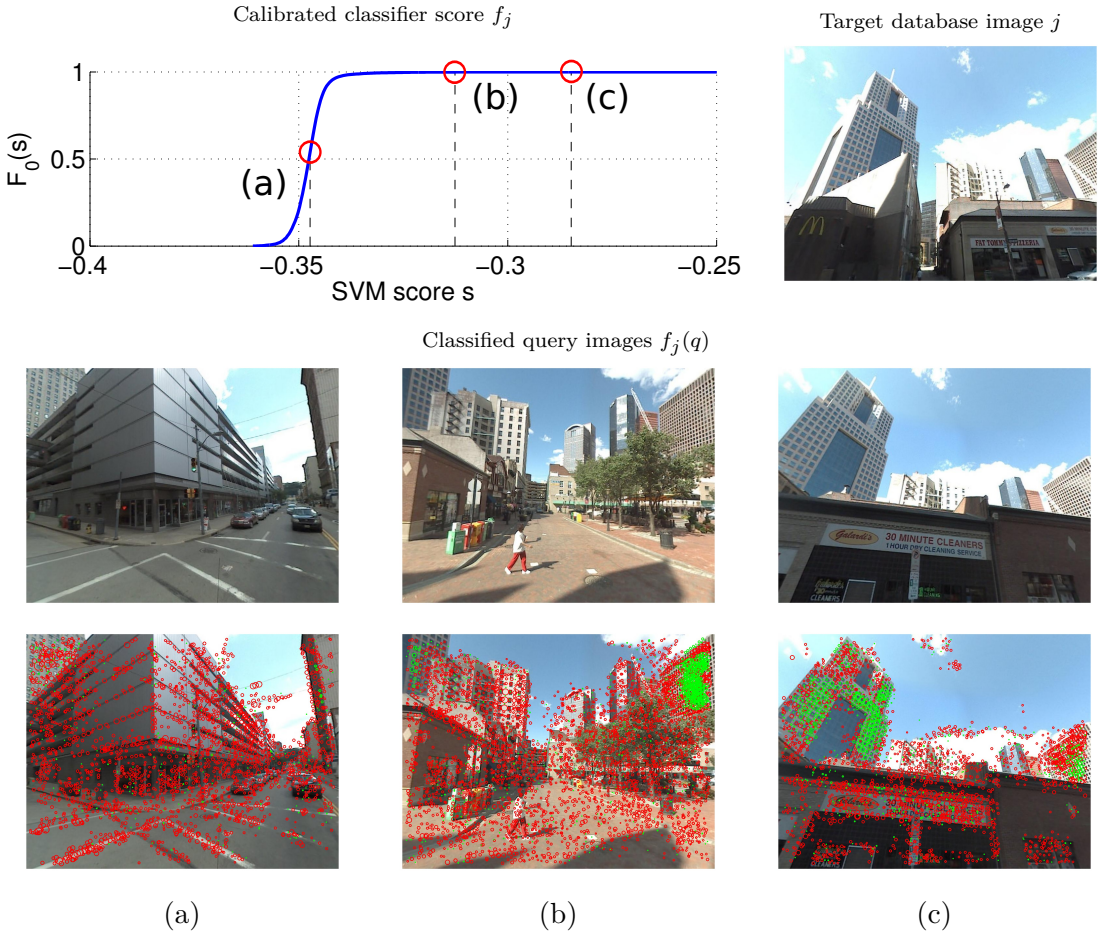


Figure 5.6: **A visualization of learned feature weights for two database images. In each panel:** *first row:* (Right) Target database image j. (Left) Cumulative density function (or calibrated score) learned for the SVM scores of the corresponding classifier f_j; three query images displayed on the *second row* are represented by their SVM scores and cdf values $F_0(s)$, denoted (a)-(c) on the graph. *Third row:* A visualization of the contribution of each feature to the SVM score for the corresponding query image. Red circles represent features with negative weights while green circles correspond to features with positive weights. The area of each circle is proportional to the contribution of the corresponding feature to the SVM score. Notice that the correctly localized queries (c) contain more green colored features than queries from other places (b) and (a). Please also note that the calibration *cdf*s in the left and right panel are similar but not identical.
Query (b) is in fact also an image of the same location with a portion of the left skyscraper in the target image detected in the upper left corner and the side of the rightmost building in the target image detected in the top right corner. Both are clearly detected by the method as indicated by a large quantity of green circles in the corresponding regions.

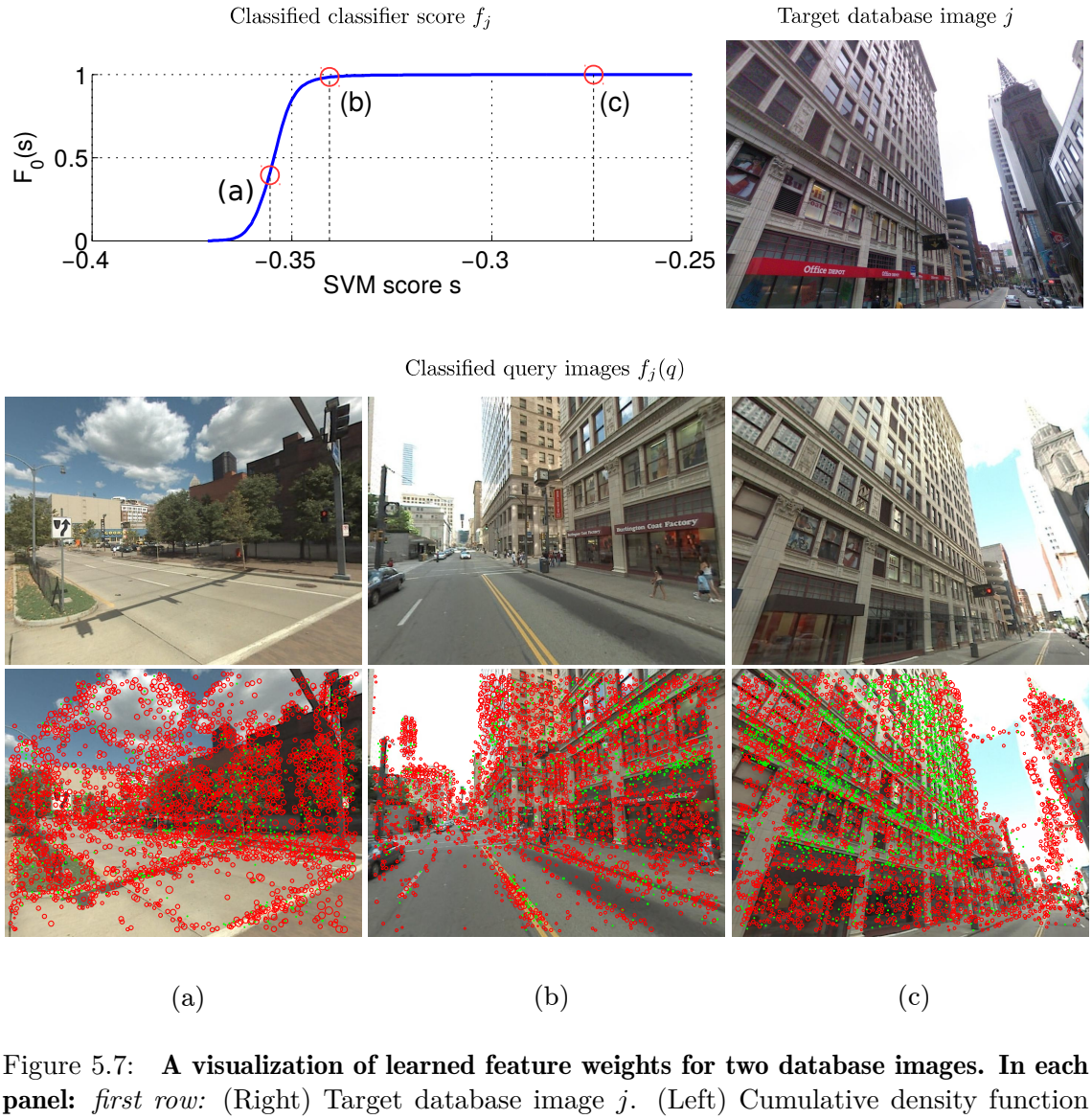


Figure 5.7: **A visualization of learned feature weights for two database images. In each panel:** *first row:* (Right) Target database image j. (Left) Cumulative density function (or calibrated score) learned for the SVM scores of the corresponding classifier f_j; three query images displayed on the *second row* are represented by their SVM scores and cdf values $F_0(s)$, denoted (a)-(c) on the graph. *Third row:* A visualization of the contribution of each feature to the SVM score for the corresponding query image. Red circles represent features with negative weights while green circles correspond to features with positive weights. The area of each circle is proportional to the contribution of the corresponding feature to the SVM score. Notice that the correctly localized queries (c) contain more green colored features than queries from other places (b) and (a). Please also note that the calibration *cdf*s in the left and right panel are similar but not identical.

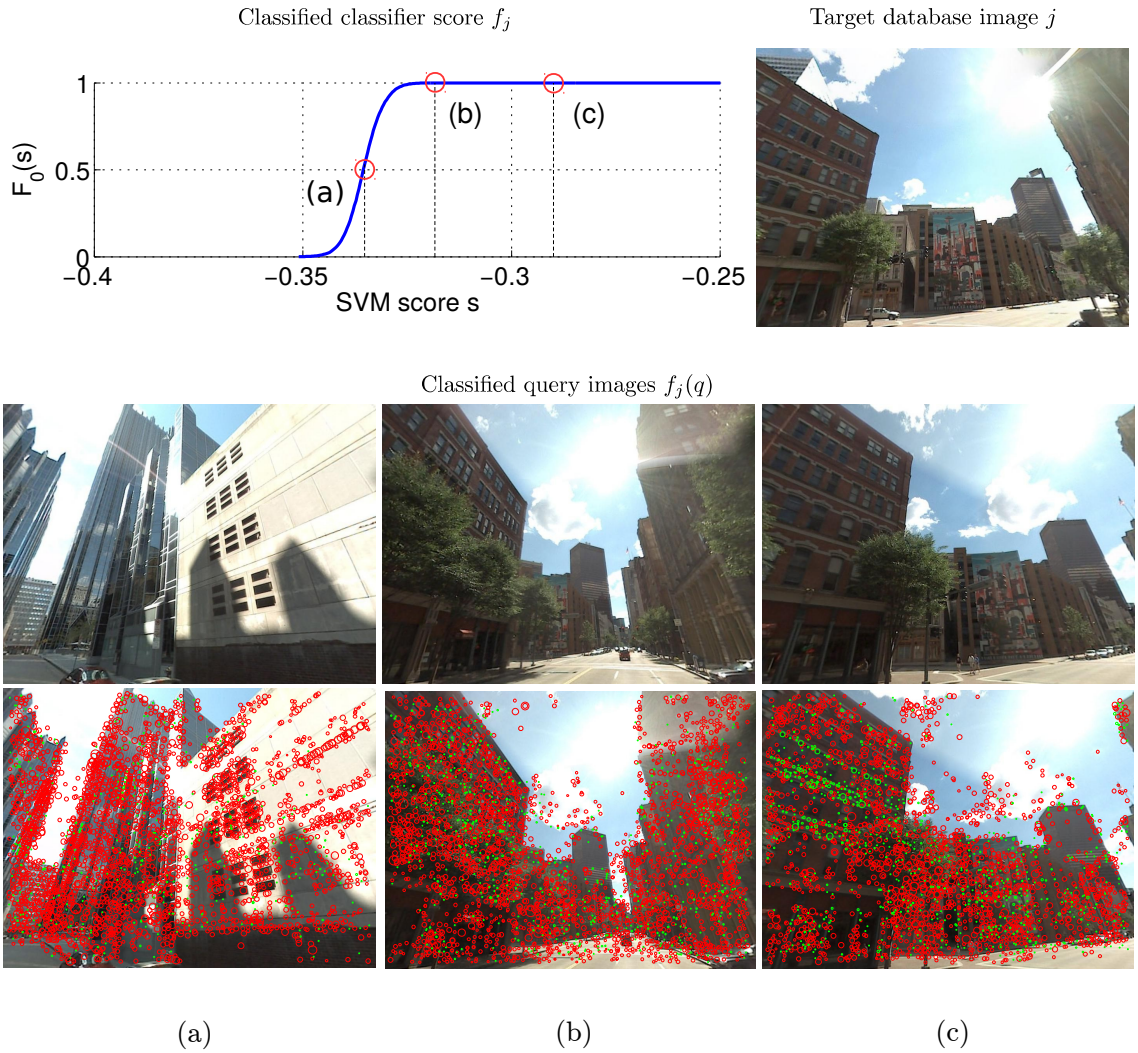


Figure 5.8: **A visualization of learned feature weights for two database images. In each panel:** *first row:* (Right) Target database image j. (Left) Cumulative density function (or calibrated score) learned for the SVM scores of the corresponding classifier f_j; three query images displayed on the *second row* are represented by their SVM scores and cdf values $F_0(s)$, denoted (a)-(c) on the graph. *Third row:* A visualization of the contribution of each feature to the SVM score for the corresponding query image. Red circles represent features with negative weights while green circles correspond to features with positive weights. The area of each circle is proportional to the contribution of the corresponding feature to the SVM score. Notice that the correctly localized queries (c) contain more green colored features than queries from other places (b) and (a). Please also note that the calibration *cdf*s in the left and right panel are similar but not identical. It is worth noting that query (b) is in fact an image of the target location but seen from further away and from a different angle.

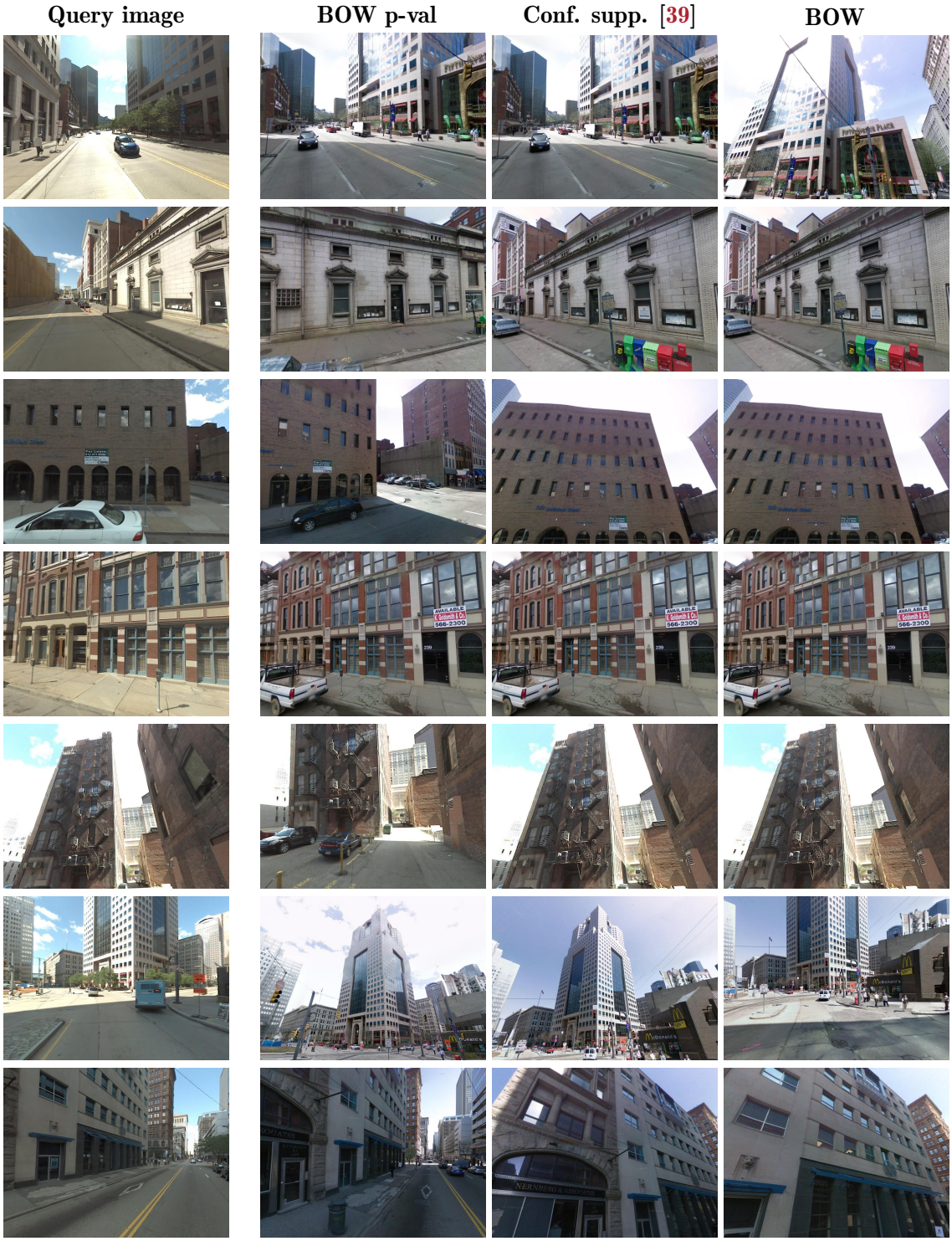


Figure 5.9: **Examples of query images correctly localized by all methods.** (a) query image. (b) top-ranked image retrieved by the per-location classifiers (proposed method). (c) top-ranked image retrieved by the baseline confuser suppression method. (d) top-ranked image retrieved by the baseline bag-of-visual-words method.

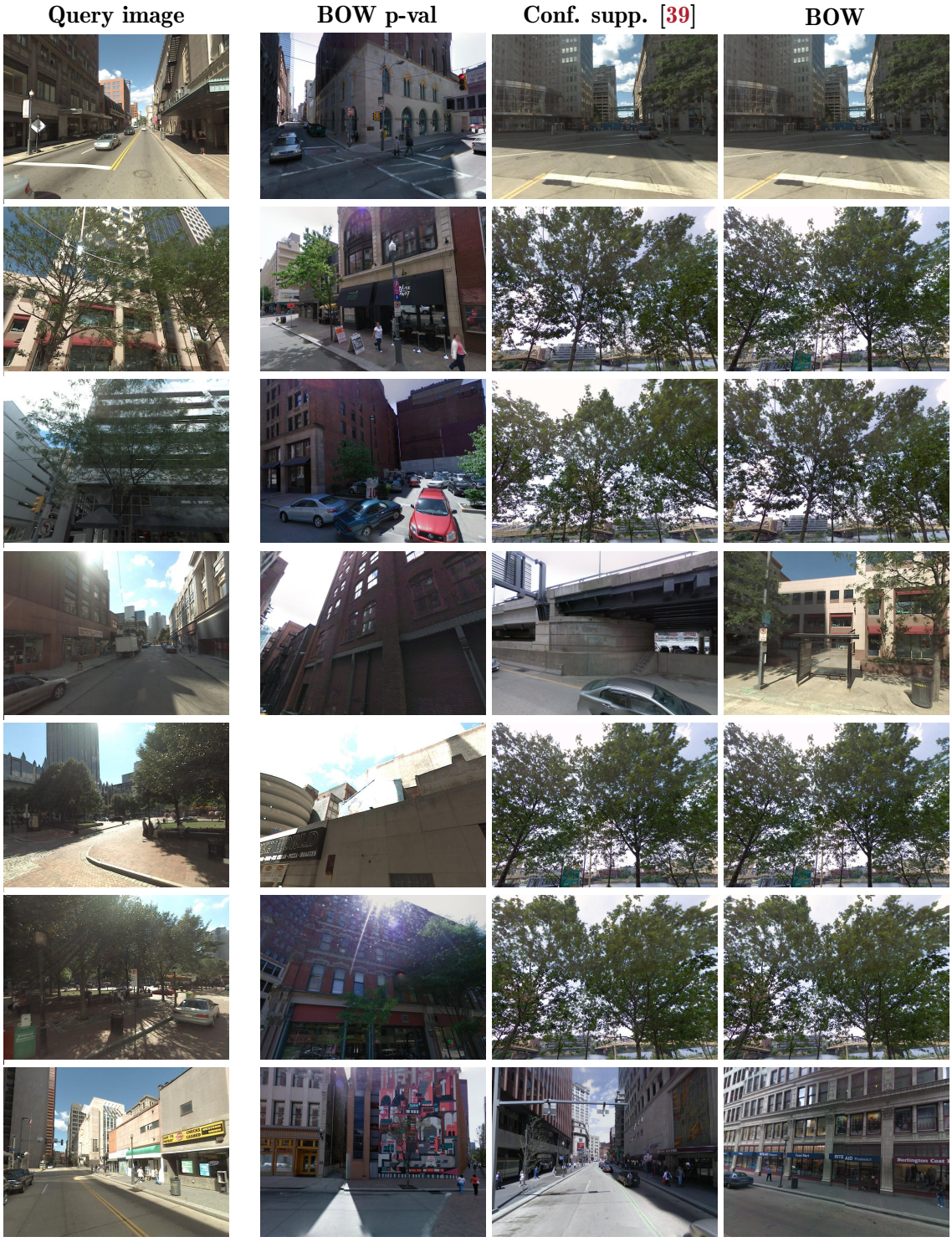


Figure 5.10: **Examples of query images incorrectly localized by all methods.** (a) query image. (b) top-ranked but incorrect image retrieved by the per-location classifiers (proposed method). (c) top-ranked but incorrect image retrieved by the baseline confuser suppression method. (d) top-ranked but incorrect image retrieved by the baseline bag-of-visual-words method. Occlusions by trees often present significant challenge for tested visual place recognition methods.

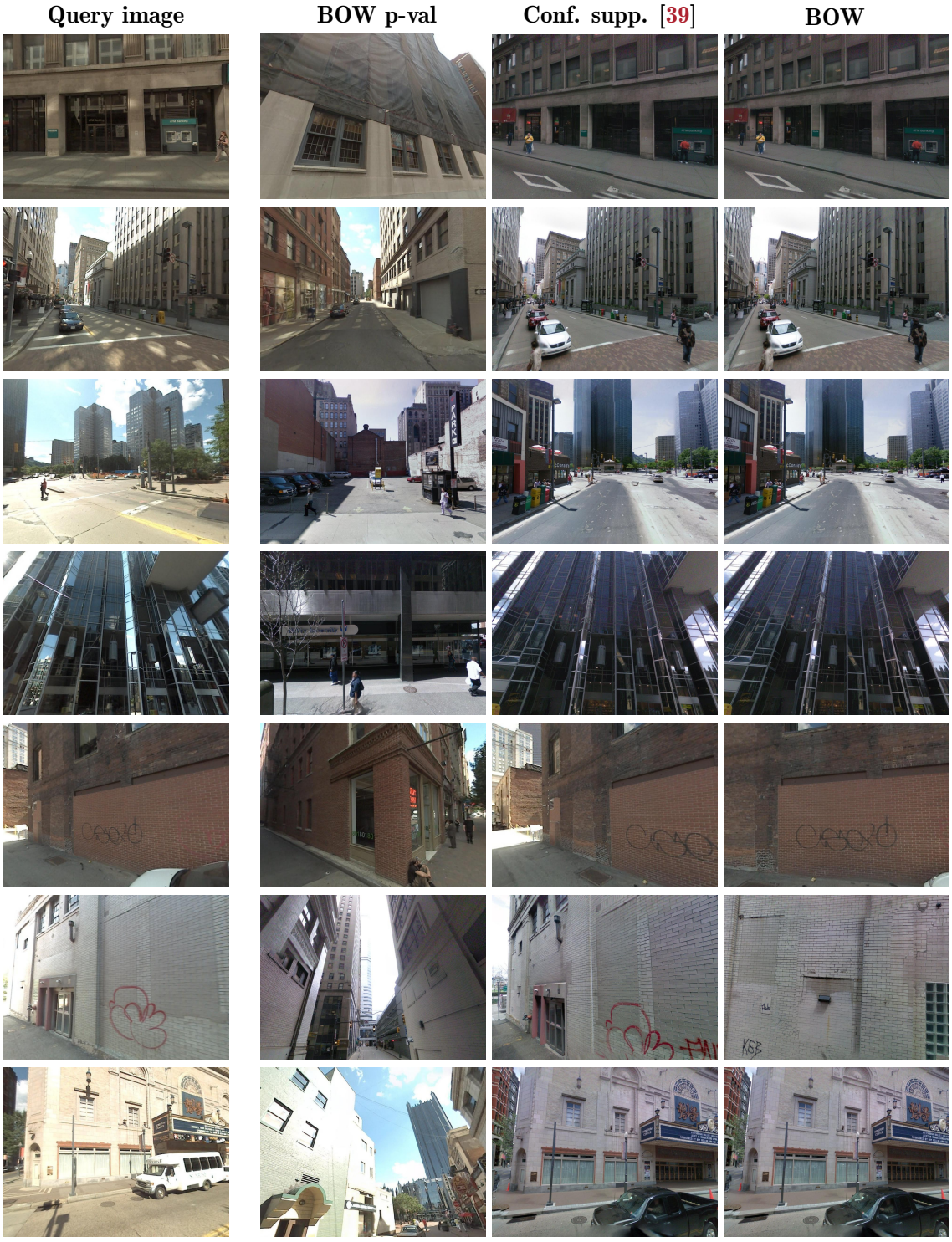


Figure 5.11: **Examples of query images incorrectly localized by our method but correctly localized by the baselines.** (a) query image. (b) top-ranked but incorrect image retrieved by the per-location classifiers (proposed method). (c) top-ranked image retrieved by the baseline confuser suppression method. (d) top-ranked image retrieved by the baseline bag-of-visual words method. The proposed method is sometimes confused by high-scoring similar repeated texture patterns on facades.

recall@K [%]	1	2	5	10	20
Method:	25k Pittsburgh				
BOW SVM no calib.	6.4	8.1	13.5	17.5	20.5
BOW	28.7	35.7	45.8	53.7	61.5
BOW Conf. supp [39]	29.6	37.3	48.9	59.3	69.2
BOW w-norm	31.8	38.7	49.7	**60.2**	**69.4**
BOW p-val	**33.0**	**40.3**	**50.2**	58.7	66.4

Table 5.1: **Evaluation of the learned bag-of-visual-words representation on the Pittsburgh 25k dataset.** The table shows the fraction of correctly recognized queries (recall@K) for the different values of $K \in \{1, 2, 5, 10, 20\}$ retrieved database images. The learned representations (BOW w-norm and BOW p-val) outperform the raw bag-of-visual-words baseline (BOW) as well as the learned representation without calibration (BOW SVM no calib).

5.2.2 Fisher vectors

Results of the proposed per-location learning method for the Fisher vector image representation for different dimensions are shown in table 5.2 and figure 5.12. Similar to bag-of-visual-words, the learned representation (w-norm) significantly improves the place recognition performance over the baseline Fisher vector (FV) matching without learning. The improvements are consistent across different lengths of shortlist K and for the different dimensionality of the Fisher vector representation. We report results only for the w-norm calibration as we found that the p-val calibration did not perform well for the learned Fisher vector classifiers (top 1 recall of 25.3% compared to baseline performance of 33.6% for dimension 128).

When examining the results we have observed that for bag-of-visual-words the cdf estimated on the database well represents the scores of (unseen) negative query images at test time. However, this is not the case for Fisher vectors where estimated cdf on the database does not represent well the scores of negative query images at test time. The scores of (unseen) negative query images often fall outside of the estimated cdf or at the very tail that is only sparsely sampled. As a result, the estimated query image p-values for Fisher vectors are often over-confident and incorrect.

Notice that the proposed per-location learning method consistently improves performance over the raw Fisher vector descriptors on the larger Pittsburgh 55k dataset and the challenging 24/7 Tokyo dataset (76k images). Examples of correctly and incorrectly localized queries are shown in figures 5.17 and 5.16 at the end of this chapter. Next, we compare the performance of the two learned representations relative to their memory footprints.

5.2.3 Analysis of recognition accuracy vs. compactness

Here we analyze the recognition accuracy of the learned representations vs. their compactness measured by their memory footprint on the Pittsburgh 25k image dataset. Ideally, we wish to learn a more compact representation, which still improves the recognition accuracy. However, usually, there is a trade-off between the discriminative power of the representation and its size, where having a more compact representation reduces the recognition accuracy [36]. We observe a similar behavior but our learned representation results in a higher recognition accuracy for a given

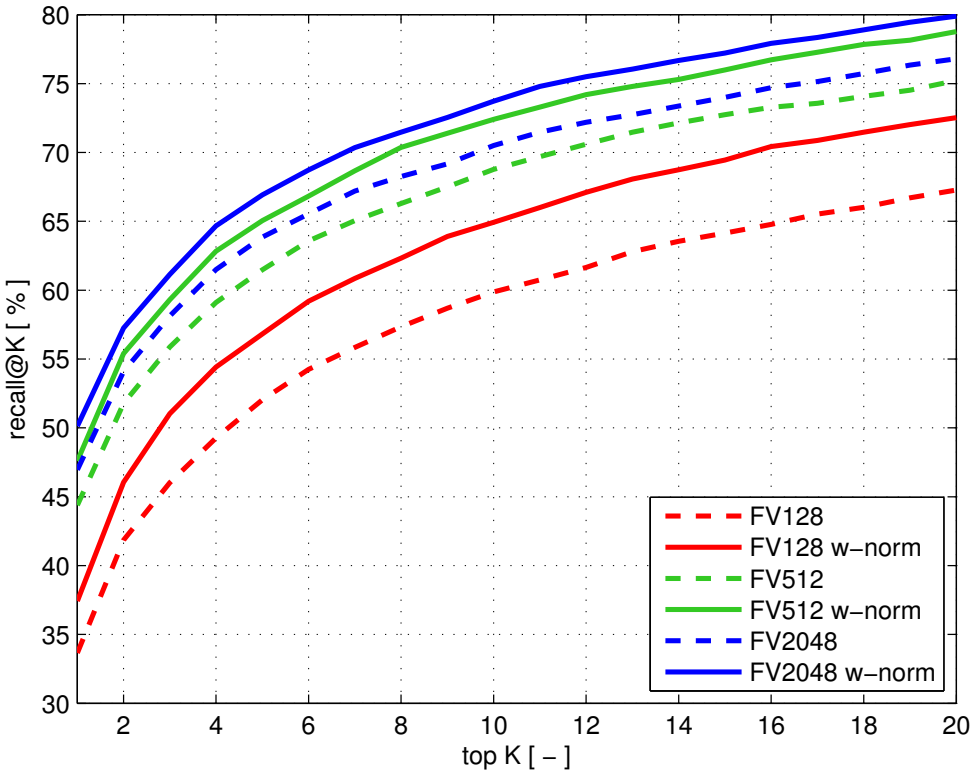


Figure 5.12: **Evaluation of the learned Fisher vector representation on the Pittsburgh 25k [27] dataset.** The graph shows the fraction of correctly recognized queries (recall@K, y-axis) vs. the number of top K retrieved database images for the raw Fisher vector baseline (FV) for different dimensions compared to the learned representation (*w-norm*). Note the consistent improvements over all lengths of shortlist K for all dimensions.

size, or alternatively, significantly reduces the size of the representation for a given accuracy. The results are summarized in figure 5.13. The figure shows the recognition performance (y-axis) for the different dimensionality of the Fisher vector representation, which corresponds to different memory footprints (x-axis). For example, for $d = 128$ the memory footprint is about 24 MB, whereas for $d = 2048$ the memory footprint is about 384 MB. Note that the x-axis is in log-scale. The bag-of-visual-words representation has a fixed dimensionality (and fixed memory footprint) and hence each bag-of-visual-words method is shown only as a single point on the graph. For Fisher vectors, the results demonstrate that for a given level of accuracy (y-axis) the proposed method learns a more compact (lower-dimensional) representation (x-axis). For example, our learned 128-dimensional descriptor (memory footprint of 24 MB) achieves a similar accuracy (around 65%) as the 256-dimensional raw Fisher descriptor (memory footprint of 51MB, interpolated from figure 5.13). This corresponds to 50% memory savings for the same level of recognition performance. Note that similar to [36], we observe a decrease in performance at high-dimensions for both the FV baseline and our method. The results also demonstrate the benefits of using the compact FV descriptors compared to the bag-of-visual-words baseline achieving significantly better recognition accuracy

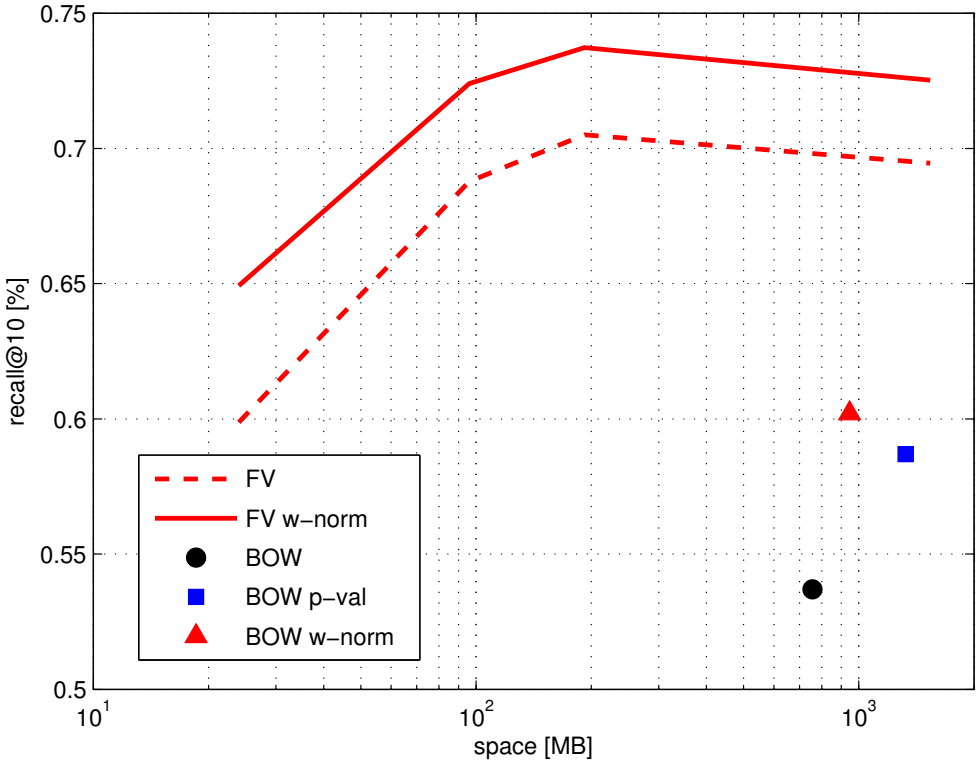


Figure 5.13: **The recognition performance vs. the memory requirements for the Pittsburgh 25k dataset.** The fraction of correctly localized queries at the top 10 retrieved images (y-axis) vs. the memory footprint (x-axis) for the different representations. For Fisher vectors, the learned descriptor (FV *w-renorm*) clearly outperforms the raw Fisher vector descriptor (FV) for all dimensions corresponding to different memory footprints (x-axis). Learnt per-location representations for the bag-of-visual-words model (BOW p-val and BOW w-norm) also improve performance over the raw bag-of-visual-words (BOW). However, the Fisher vectors provide much better recognition performance for the same memory footprint.

for a similar memory footprint.

5.2.4 Comparison to linear discriminant analysis (LDA) and whitening baselines

We have compared our method to the linear discriminant analysis (LDA) [5, 28, 19] and whitening [33] baselines. Results are reported on the Pittsburgh $25k$ dataset. The LDA baseline finds a discriminative linear projection of the feature space by minimizing a Euclidean loss rather than hinge loss used in our work. In detail, following [5] we have used all available database to learn the covariance matrix and used calibrated LDA score (see [5] eq. 11) to obtain a classifier for each database image. We have applied the LDA method on the 128-dimensional Fisher vector descriptor but have obtained significantly worse performance (31.9% for recall@1) than our method (recall@1 of 38.3%). We believe the better performance of our method can be attributed to (i) the use of hinge-loss and (ii) training using the top scoring hard negative examples that are specific to each place. Next, we compare results to PCA compression followed by whitening as suggested in [33]. For bag-of-visual-words, we follow [33] and compare performance to PCA whitening to a target dimension of 4096. We have observed performance drop compared to the raw bag-of-visual-words baseline (28.7% to 26.1% for recall@1). We hypothesize this could be attributed to the large dictionary size used in our work (100k), whereas [33] report improved results for single dictionary whitening only for dictionaries of up to 32k visual words. Finally, we have also applied PCA whitening on Fisher vector descriptors of dimensions 128, 512 and 2048, but have not observed significant improvements over the baseline raw descriptors. In fact, for the highest dimension (2048) we have observed a performance drop (49.6% to 41.3%), which could be attributed to amplification of low-energy noise as also reported in [33].

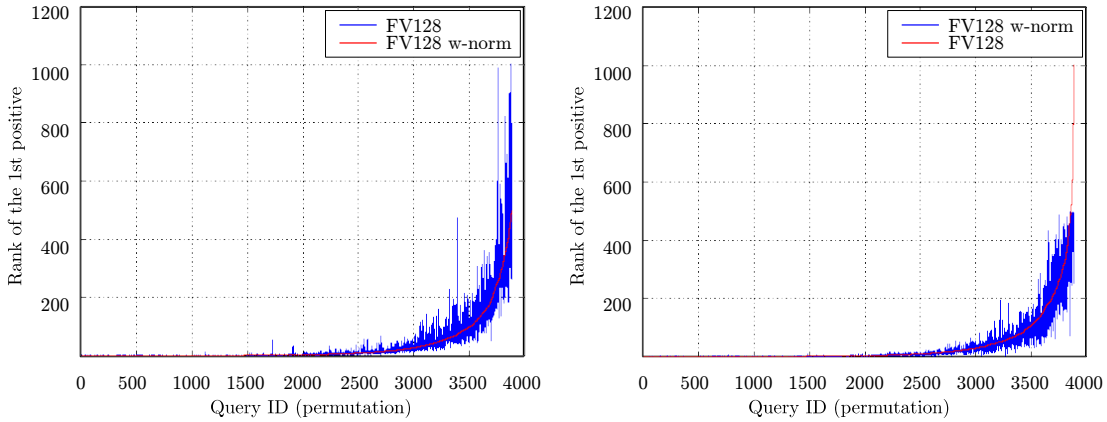
5.2.5 Analysis of improvements and failure cases

We have examined the improvement and failures of the w-norm method w.r.t. the Fisher vector baseline on the Pittsburgh 25k dataset. We analyzed the cases for which the w-norm method improves the rank of the first ground truth image compared to the baseline and for which the rank of the first ground truth image is made worse. For brevity, in the following text, we use “positive" image to denote the first ranked ground truth image for a given query.

In detail, we consider a list of tuples, each tuple corresponds to a query image and contains two elements. The first element is a rank of the first positive image retrieved by the raw FV128 while the second element is a rank retrieved by the FV128 w-norm method. Tuples in the list are then sorted w.r.t. the FV128 w-norm rank.

A visualization of the sorted list is provided in figure 5.14 (a). The red curve represents the rank of the FV128 w-norm while the corresponding rank given by the raw FV128 is depicted in blue. For instance, the blue peak at the query ID value 3395 means that for that particular query image the first positive image was originally found at the rank of 481 but the rank was improved by FV128 w-norm method to 64. The similar plot is constructed by sorting the list w.r.t. the raw FV128 and the result is shown in figure 5.14 (b). In this case, the red curve indicates rank obtained by the raw FV128 while a rank obtained by the FV128 w-norm is depicted in blue.

The plots indicate that the FV128 w-norm not only improves the results for small shortlists but improves the results for difficult queries that are typically ranked very low. Notice that for difficult queries, some images are ranked close to 1000 when using the raw FV128, while none is ranked above 500 when using the FV128 w-norm method. A few examples of the difficult queries along



(a) Ours (red) vs. baseline (blue)

(b) Baseline (red) vs. ours (blue)

Figure 5.14: **Analysis of the change in ranking.** Each data point in the plot shows a rank of the best positive image for given a query obtained by two different methods shown in red (the baseline method) or blue (the alternative method). The baseline ranks (red) are sorted in the ascending order. Each blue bar shows a difference in ranking obtained by the alternative method. The blue being above the red curve means that the ranking got worse and vice versa. (left) The FV128 w-norm method has been used as a baseline. (right) The raw FV128 method has been used as a baseline. Notice the trend of improving the ranking when FV 128 w-norm is used.

with retrieved positive images are shown in figure 5.18.

We further analyze how the method affects the ranking in smaller shortlists, which is not obvious from figure 5.14. The scatter plot in figure 5.15 shows how the ranks are improved or made worse by the FV128 w-norm method. Each point represents a rank of the best scoring positive image for a given query image. The plot suggests that FV128 w-norm attracts images into the smaller shortlists but sometimes the best scoring positives are pushed out.

For example, considering a shortlist of the size 20 we want to identify when: (i) an image with the rank of 20 is attracted into the shortlist (improvement), and (ii) an image with the rank of ≤ 20 is pushed out of the shortlist using our method (aggravation).

In figure 5.15, we observe that in many cases a low-ranked true positive image by the baseline (ranked $30-70$) is attracted into the shortlist of size 20 by the w-norm method, resulting in an improvement. Note that in many other cases our method improves ranking but here we only count the cases for which the baseline method does not have any true positives in the top 20 shortlist. On the other hand, in 39 cases our method makes the results worse and removes a correct image from the top 20 shortlist but typically only to top 40.

Finally, we observe that aggravation typically occurs on hard examples where the baseline performance is already bad. When visually inspecting the failure cases we observed that our method typically fails on queries containing a big portion of the sky clouds or vegetation, narrow streets or tunnels and sometimes retrieves images capturing the same building from a different viewpoint or a larger distance. Examples of difficult queries along with the first correct image are shown in figure 5.18 at the end of this chapter.

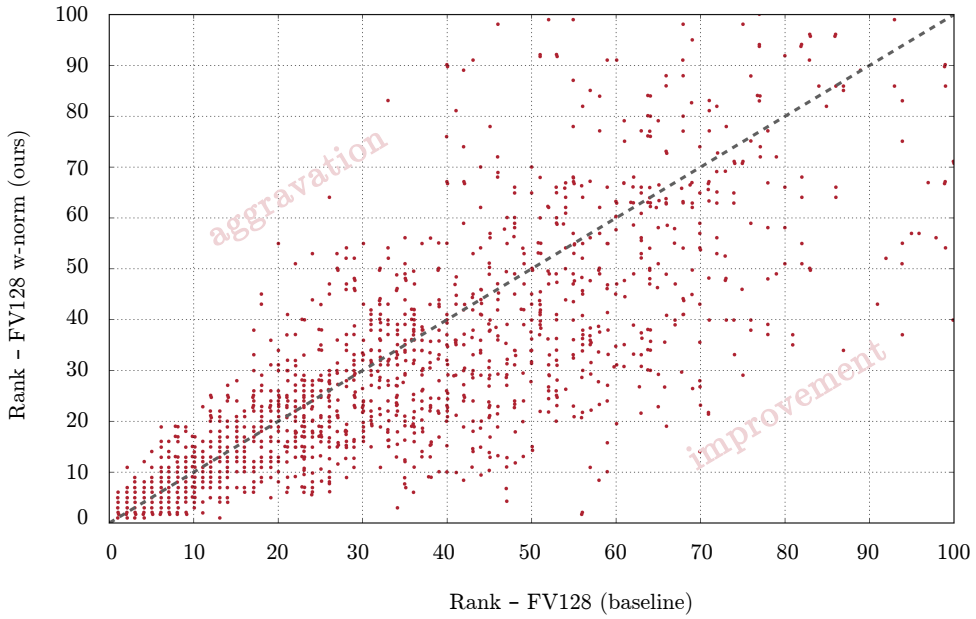


Figure 5.15: **Rank scatter plot.** Each data point in the scatter plot corresponds to two different rankings of the best scoring positive image. The rank obtained by the raw FV128 is shown on abscissa while the rank obtained by the FV128 w-norm method is shown on the ordinate. If a point appears below the diagonal line, it is better ranked by the proposed method than the baseline.

5.2.6 Scalability

In the offline stage, our method collects hard negative examples for each location in the database, which are consequently used to train exemplar SVM classifiers. As only a constant number of examples (1-5 positives and 500 negatives) is used to train each per-location classifier the overall complexity of training is linear, $O(N)$, i.e. we need to train one classifier (with constant training time) for each of N images in the database. The bottleneck of the offline stage is collecting the negative examples that is quadratic $O(N^2)$ in the database size. In other words, for each of N database images, we need to find the top 500 most similar negatives among all N database images. However, we believe that even finding negatives can be scaled-up to very large datasets with standard compression techniques such as product quantization (PQ) [34] combined with sub-linear approximate nearest neighbor search [45].

At query time our method needs to compute the calibrated e-SVM score (equation (4.2)) of the query for each image in the database. In the case of w-norm method, the calibration weights can be included in the classifier weight matrix, as discussed in section 4.4. For the p-val calibration method, each e-SVM score must be calibrated using K stored values of the non-parametric CDF model. This requires a search for the two closest values and subsequent interpolation, which yields complexity of $O(N \log K)$. Since K is only a constant both the w-norm and p-val methods have a linear time complexity (in the size of the database) at query time but with different constants. However, in practice, the constant in the p-val method can be quite large. The actual running time per query is $340ms$ for the bag-of-visual-words representation with p-val calibration and $3ms$ for the FV128 descriptor with w-norm calibration. Both timings are on the 25k Pittsburgh dataset on a desktop with CPU Intel Xenon E5 using a single thread. Hence in practice, the p-val method may be scalable only to medium size datasets. For the w-norm method, the query time can be further sped up using sub-linear approximate nearest neighbor search [45] on compressed descriptors [34], making the method scalable to very large datasets.

recall@K [%]	1	2	5	10	20
Method / Dataset:	25k Pittsburgh				
FV128	33.6	41.8	52.0	59.8	67.7
FV128 w-norm	**38.3**	**47.5**	**57.7**	**65.8**	**72.7**
FV512	44.3	51.7	61.4	68.7	75.2
FV512 w-norm	**47.6**	**55.4**	**65.1**	**72.4**	**78.8**
FV2048	46.9	54.1	63.8	70.5	76.8
FV2048 w-norm	**50.2**	**57.3**	**67.0**	**73.8**	**78.0**
FV16384	45.3	54.1	63.8	69.4	75.3
FV16384 w-norm	**49.3**	**56.0**	**65.9**	**72.5**	**76.8**
	55k Pittsburgh				
FV128	10.9	14.1	20.2	26.4	33.2
FV128 w-norm	**13.5**	**17.7**	**25.0**	**31.8**	**39.0**
FV512	17.3	21.1	28.4	34.2	40.3
FV512 w-norm	**19.8**	**25.1**	**32.7**	**38.7**	**46.0**
FV2048	19.2	23.5	29.9	35.2	41.9
FV2048 w-norm	**20.8**	**25.9**	**33.1**	**38.7**	**45.9**
	24/7 Tokyo				
FV128	14.2	20.0	27.9	34.2	41.5
FV128 w-norm	**16.9**	**22.0**	**29.6**	**37.2**	**44.8**
FV512	35.2	40.3	43.8	48.2	57.1
FV512 w-norm	**36.1**	**42.0**	**46.8**	**52.8**	**61.4**
FV2048	37.4	42.5	48.5	53.9	58.7
FV2048 w-norm	**42.9**	**46.7**	**52.8**	**58.8**	**66.7**
FV4096	42.9	46.3	54.0	59.0	64.8
FV4096 w-norm	**44.3**	**47.1**	**54.7**	**61.1**	**66.5**

Table 5.2: **Evaluation of the learned Fisher vector representation on the Pittsburgh [27] and 24/7 Tokyo [64] datasets.** The table shows the fraction of correctly recognized queries (recall@K) for the different values of $K \in \{1, 2, 5, 10, 20\}$ retrieved database images. The learned Fisher vector representation (FV *w-norm*) consistently improves over the standard Fisher vector matching baseline (FV) for all target dimensions.

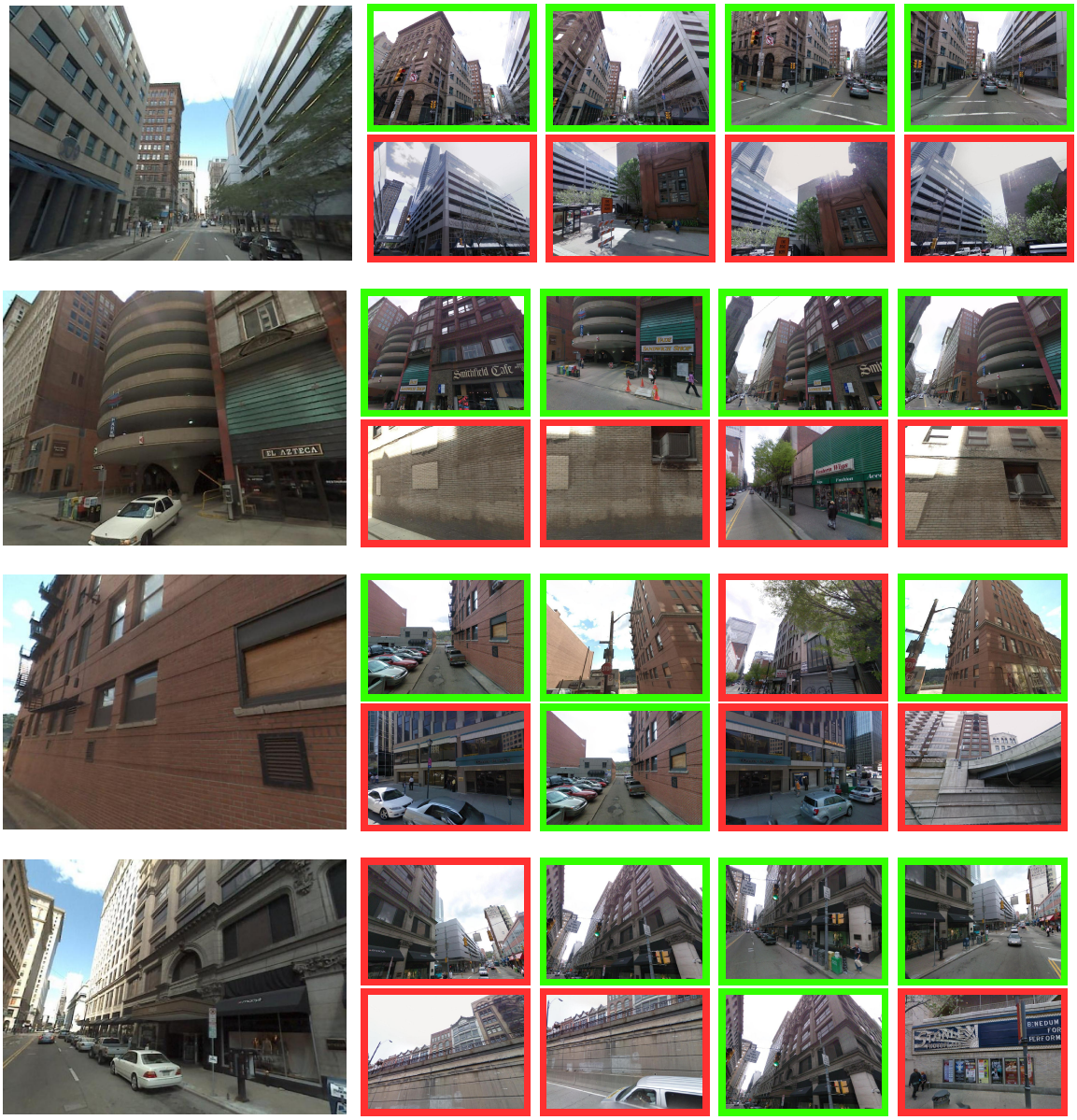


Figure 5.16: **Examples of correctly and incorrectly localized queries for the learned bag-of-visual-words representation.** Each example shows a query image (left) together with correct (green) and incorrect (red) matches from the database obtained by learned bag-of-visual-words representation *p-val* method (top) and the standard bag-of-visual-words baseline (bottom). Note that the proposed method is able to recognize the place depicted in the query image despite changes in viewpoint, illumination and partial occlusion by other objects (trees, lamps) and buildings. Note also that bag-of-visual-words baseline is often confused by repeating patterns on facades and walls.

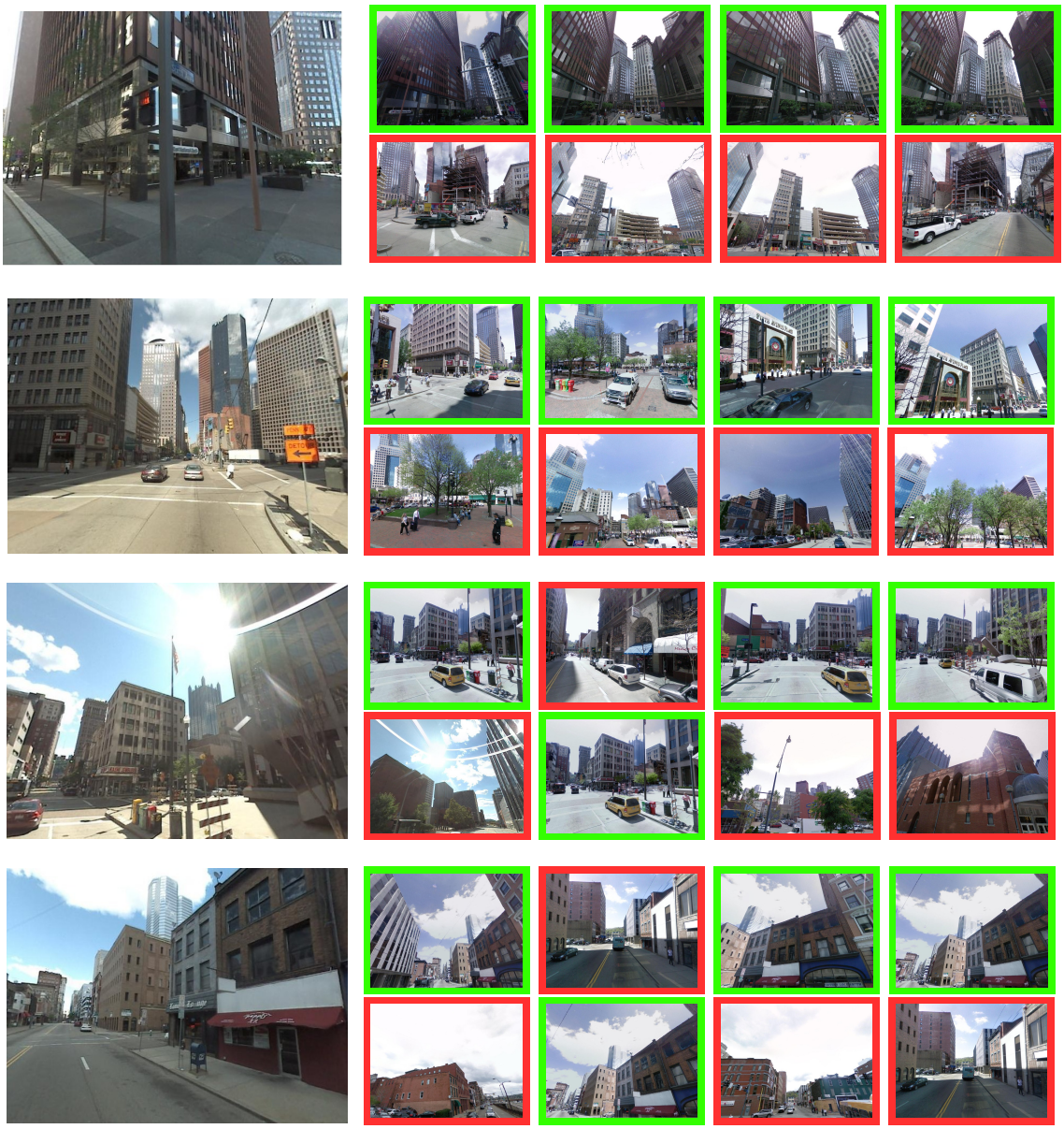


Figure 5.17: **Examples of correctly and incorrectly localized queries for the learned Fisher vector representation.** Each example shows a query image (left) together with correct (green) and incorrect (red) matches from the database obtained by the learned Fisher vector representation *w-norm* method (top) and the standard Fisher vector baseline (bottom) for dimension 128. Note that the proposed method is able to recognize the place depicted in the query image despite changes in viewpoint, illumination and partial occlusion by other objects (trees, lamps) and buildings. Note that the baseline methods often finds images depicting the same buildings but in a distance whereas our learned representation often finds a closer view better matching the content of the query.

Figure 5.18: **Failure cases on difficult queries.** In each column, we show a difficult query image (left) and the first correct image obtained by the baseline method FV128 (top-right) and the proposed method FV128 w-norm (bottom-right) along with its rank. Here we show examples, where proposed method performs worse than the baseline. Regarding the difficult queries, we observed that our method typically fails on queries containing a big portion of the sky clouds or vegetation, narrow streets or tunnels and sometimes retrieves images capturing the same building from a different viewpoint or a larger distance.

6 Discussion

"There are no such things as applied sciences, only applications of science."

— Louis Pasteur

In this thesis we proposed a method for building place recognition datasets and releaseed it to public in the form of a software package [25]. We proposed a new approach for the place recognition. In particular, we cast the place recognition problem as a classification task, we show that subsequent calibration of the classification score is crucial and propose two calibration strategies. The major contributions and future work are summarized below.

6.1 Contributions

We developed an algorithm capable of building geo-referenced datasets for, but not limited to, place recognition. We released this package as open source project called `streetget` [25]. We have discussed the internal representation of the Google StreetView panoramas, how the data and metadata are stored, how to fetch them and, finally, how to use the data to generate a place recognition dataset. The method is also capable of getting the historical panoramas and the scene depth maps.

We have shown that place recognition can be cast as a classification problem and have used geotags as a readily-available supervision to train an ensemble of classifiers, so called per-location classifiers, one for each location in the place recognition database. As only a few positive examples are available for each location, we have developed two procedures to calibrate the output of each classifier without the need for additional positive training data. We have shown that learning the per-location representations improves the place recognition performance over the raw bag-of-visual-words and Fisher vector matching baselines. The developed calibration methods are not specific to place recognition and can be useful for other per-exemplar classification tasks, where only a small number of positive examples are available [44].

6.2 Future work

Being able to generate a large geo-referenced temporal dataset with panoramas capturing the same scene at different times of the year/decade, yet being able to retrieve the scene depth opens new research opportunities not limited to place recognition.

Using the temporal data, we would like to train a convolutional neural network (CNN) to predict which parts of a scene within the image are correlated with the arrow of are likely to look differently in the future. Examples may include a crack on the wall that grows in the time, flaking off the

facade is only getting worse, a coffee shop that changes owner and changes its appearance (e.g. turns into the Starbuck's coffee).

Another opportunity is to exploit the scene depth information. Despite the fact that the scene is approximated by a set of planes, this representation can be used as a weak form of supervision for a CNN that could rectify the input image, learn the approximate camera projection matrix or help with the semantic segmentation of an input image.

Acknowledgements

This work was supported by the MSR-INRIA laboratory, the EIT-ICT labs, Google, the ERC project LEAP and the EC project PRoViDE FP7-SPACE-2012-312377, LADIO EU-H2020 No. 731970. Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL or the U.S. Government.

Bibliography

[1] Agarwal, S., Snavely, N., Simon, I., Seitz, S., and Szeliski, R. Building Rome in a day. In *ICCV* (2009), pp. 72–79. 11

[2] Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. NetVLAD: CNN architecture for weakly supervised place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (2016). 4, 29

[3] Arandjelović, R., and Zisserman, A. Three things everyone should know to improve object retrieval. In *IEEE PAMI* (2012). 47

[4] Aubry, M., Maturana, D., Efros, A., Russell, B., and Sivic, J. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR* (2014). 16, 17, 40

[5] Aubry, M., Russell, B., and Sivic, J. Painting-to-3D model alignment via discriminative visual elements. *ACM Transactions on Graphics* (2014). 16, 17, 61

[6] Bay, H., Tuytelaars, T., and Van Gool, L. SURF: Speeded up robust features. In *ECCV* (2006). 9, 11, 47

[7] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. 34

[8] Cao, S., and Snavely, N. Graph-based discriminative learning for location recognition. In *CVPR* (2013), IEEE, pp. 700–707. 11, 13, 14, 33

[9] Casella, G., and Berger, R. *Statistical inference*. Duxbury Press, 2001. 35

[10] Chen, D., Baatz, G., Köser, Tsai, S., Vedantham, R., Pylvanainen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., Girod, B., and Grzeszczuk, R. City-scale landmark identification on mobile devices. In *CVPR* (2011). 9, 11, 14, 45, 49

[11] Chum, O., Philbin, J., Sivic, J., Isard, M., and Zisserman, A. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV* (2007). 48

[12] Csurka, G., Bray, C., Dance, C., and Fan, L. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV* (2004), pp. 1–22. 11

[13] Cummins, M., and Newman, P. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems* (Seattle, USA, June 2009). 9, 11, 33

[14] Dalal, N., and Triggs, B. Histogram of oriented gradients for human detection. In *CVPR* (2005). 9, 17

[15] Doersch, C., Gupta, A., and Efros, A. A. Mid-level visual element discovery as discriminative mode seeking. In *NIPS* (2013). 11, 17, 40

[16] Doersch, C., Singh, S., Gupta, A., Sivic, J., and Efros, A. A. What makes paris look like paris? *SIGGRAPH 31*, 4 (2012). 9, 15

[17] Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A library for large linear classification. *J. Machine Learning Research 9* (2008), 1871–1874. 48

[18] Gebel, M., and Weihs, C. Calibrating classifier scores into probabilities. *Advances in Data Analysis* (2007), 141–148. 16

[19] Gharbi, M., Malisiewicz, T., Paris, S., and Durand, F. A Gaussian approximation of feature space for fast image similarity. Tech. rep., MIT, 2012. 15, 16, 17, 61

[20] Gong, Y., Wang, L., Guo, R., and Lazebnik, S. *Multi-scale Orderless Pooling of Deep Convolutional Activation Features*. Springer International Publishing, Cham, 2014, pp. 392–407. 29

[21] Google. Icmla 2011 streetview recognition challenge, 2011. xix, 46

[22] Google. Google earth. *https://www.google.com/earth/*, Jul 2016. 2

[23] Google. Panoramio - photos of the world. *http://www.panoramio.com*, 2016. 2

[24] Gronat, P. Project webpage: Learning and calibrating per-location classifiers for visual place recognition, 2013. 46

[25] Gronat, P. `streetget` - a small python package for building place recognition datasets. *http://www.di.ens.fr/willow/research/streetget/*, 2015. 4, 22, 69, 79

[26] Gronat, P., Havlena, M., Šivic, J., and Pajdla, T. Building streetview datasets for place recognition and city reconstruction. Research Report CTU–CMP–2011–01, Center for Machine Perception, K13133 FEE Czech Technical University, Prague, Czech Republic, January 2011. 14

[27] Gronat, P., Obozinski, G., Sivic, J., and Pajdla, T. Learning and calibrating per-location classifiers for visual place recognition. In *CVPR* (2013). xix, xxi, xxiii, 45, 50, 59, 65

[28] Hariharan, B., Malik, J., and Ramanan, D. Discriminative decorrelation for clustering and classification. In *ECCV* (2012). 15, 16, 17, 61

[29] Hartley, R. I., and Zisserman, A. *Multiple View Geometry in Computer Vision*, second ed. Cambridge University Press, ISBN: 0521540518, 2004. 13

[30] Hays, J., and Efros, A. A. im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)* (2008). 9, 13

[31] Horaczek, S. How many photos are uploaded to the internet every minute?, 2014. 1

[32] Irschara, A., Zach, C., Frahm, J.-M., and Bischof, H. From structure-from-motion point clouds to fast location recognition. In *CVPR* (2009). 11, 12, 33

[33] Jégou, H., and Chum, O. Negative evidences and co-occurences in image retrieval: The benefit of pca and whitening. In *ECCV* (2012), pp. 774–787. 10, 17, 61

[34] Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 33*, 1 (2011), 117–128. 10, 11, 41, 64

[35] Jégou, H., Douze, M., Schmid, C., and Pérez, P. Aggregating local descriptors into a compact image representation. In *CVPR 2010 - 23rd IEEE Conference on Computer Vision & Pattern Recognition* (San Francisco, United States, 2010), IEEE Computer Society, pp. 3304–3311. 6

[36] Jégou, H., Perronnin, F., Douze, M., Sánchez, J., Pérez, P., and Schmid, C. Aggregating local image descriptors into compact codes. *IEEE PAMI 34* (2012), 1704–1716. 6, 10, 11, 29, 33, 41, 46, 47, 58, 59

[37] Kalogerakis, E., Vesselova, O., Hays, J., Efros, A., and Hertzmann, A. Image sequence geolocation with human travel priors. In *IEEE 12th International Conference on Computer Vision (ICCV)* (2009), pp. 253–260. 9, 14

[38] Klingner, B., Martin, D., and Roseborough, J. Street view motion-from-structure-from-motion. In *ICCV* (2013). 11

[39] Knopp, J., Sivic, J., and Pajdla, T. Avoidng confusing features in place recognition. In *ECCV* (2010). 9, 11, 13, 33, 49, 55, 56, 57, 58

[40] Li, Y., Crandall, D., and Huttenlocher, D. Landmark classification in large-scale image collections. In *ICCV* (2009). 9, 14

[41] Li, Y., Snavely, N., and Huttenlocher, D. Location recognition using prioritized feature matching. In *ECCV* (2010). 11, 12, 33

[42] Li, Y., Snavely, N., Huttenlocher, D., and Fua, P. Worldwide pose estimation using 3d point clouds. In *ECCV* (2012). 11, 12, 33

[43] Lowe, D. Distinctive image features from scale-invariant keypoints. *IJCV 60*, 2 (2004), 91–110. 9, 11

[44] Malisiewicz, T., Gupta, A., and Efros, A. A. Ensemble of exemplar-svms for object detection and beyond. In *ICCV* (2011). 14, 15, 16, 33, 34, 69

[45] Muja, M., and Lowe, D. G. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 36* (2014). 64

[46] Nister, D., and Stewenius, H. Scalable recognition with a vocabulary tree. In *CVPR* (2006). 10, 11

[47] Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. Object retrieval with large vocabularies and fast spatial matching. In *CVPR* (2007). 10, 11, 47, 48

[48] Philbin, J., Sivic, J., and Zisserman, A. Geometric latent dirichlet allocation on a matching graph for large-scale image datasets. *IJCV* (2010). 11, 33

[49] Platt, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers* (1999). 16

[50] Radenović, F., Tolias, G., and Chum, O. *CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples.* Springer International Publishing, Cham, 2016, pp. 3–20. 10

[51] Sattler, T., Havlena, M., Schindler, K., and Pollefeys, M. Large-scale location recognition and the geometric burstiness problem. In *CVPR* (2016). 13

[52] Sattler, T., Leibe, B., and Kobbelt, L. Improving image-based localization by active correspondence search. In *ECCV* (2012). 11, 12

[53] Sattler, T., Leibe, B., and Kobbelt, L. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016). 11, 12

[54] Sattler, T., Weyand, T., Leibe, B., and Kobbelt, L. Image retrieval for image-based localization revisited. In *Proc. BMVC* (2012). 49

[55] Scheirer, W., Kumar, N., Belhumeur, P. N., and Boult, T. E. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *CVPR* (2012). 16, 38

[56] Schindler, G., Brown, M., and Szeliski, R. City-scale location recognition. In *CVPR* (2007). 9, 11, 14, 33

[57] Scholkopf, B., and Smola, A. *Learning with kernels*. MIT press, Cambridge, 2002. 17, 41, 42

[58] Schönberger, J. L., Zheng, E., Pollefeys, M., and Frahm, J.-M. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)* (2016). 11

[59] Shrivastava, A., Malisiewicz, T., Gupta, A., and Efros, A. A. Data-driven visual similarity for cross-domain image matching. In *SIGGRAPH ASIA* (2011). 15

[60] Singh, S., Gupta, A., and Efros, A. A. Unsupervised discovery of mid-level discriminative patches. In *ECCV* (2012). 10, 15

[61] Sivic, J., and Zisserman, A. Video Google: A text retrieval approach to object matching in videos. In *ICCV* (2003). 9, 11, 33, 40, 41, 46, 47

[62] Tighe, J., and Lazebnik, S. Finding things: Image parsing with regions and per-exemplar detectors. In *CVPR* (2013). 15, 16

[63] Torii, A. Project webpage: 24/7 place recognition by view synthesis. 46

[64] Torii, A., Arandjelović, R., Sivic, J., Okutomi, M., and Pajdla, T. 24/7 place recognition by view synthesis. In *CVPR* (2015). xxiii, 45, 46, 65

[65] Torii, A., Sivic, J., and Pajdla, T. Visual localization by linear combination of image descriptors. In *ICCV Workshops* (2011), IEEE, pp. 102–109. 11, 13, 33, 45

[66] Torii, A., Sivic, J., Pajdla, T., and Okutomi, M. Visual place recognition with repetitive structures. In *CVPR* (2013). 9, 11, 13, 33

[67] Turcot, P., and Lowe, D. Better matching with fewer features: The selection of useful features in large database recognition problem. In *WS-LAVD, ICCV* (2009). 10, 11, 13, 33

[68] Zadrozny, B., and Elkan, C. Transforming classifier scores into accurate multiclass probability estimates. In *ACM SIGKDD* (2002). 15, 16

[69] Zamir, A., and Shah, M. Accurate image localization based on google maps street view. In *ECCV* (2010). 9, 11, 13, 33

[70] Zeisl, B., Sattler, T., and Pollefeys, M. Camera pose voting for large-scale image-based localization. In *The IEEE International Conference on Computer Vision (ICCV)* (December 2015). 11, 12

APPENDIX

A - HTTP requests

"Computer Science is no more about computers than astronomy is about telescopes."

— EW Dijkstra

Below we show several important HTTP GET requests for fetching the panorama image and its associated metadata. For brevity alle data structures are are considered as Python objects.

A.1 GPS to hash identifier id

This is a geocoding request. Supplied with the GPS coordinate and search radius it finds any panorama within the radius. A response to the request is metadata in JSON format. A hash identifier can be further retrieved from the metadata.

Base URL

```
https://geo0.ggpht.com/cbk
```

Query parameters

```
{'authuser': '0',
 'cb_client': 'maps_sv.tactile',
 'hl': 'en',
 'll': '50.000000,14.410000',
 'output': 'json',
 'radius': 15}
```

An example of full URL string

```
https://geo0.ggpht.com/cbk?authuser=0&cb_client=maps_sv.tactile&ra...
```

where ellipses denote rest of the string.

A.2 Metadata - the first portion

The first portion of the metadata contains information about the panorama location such as a panorama hash id, GPS location, panorama heading w.r.t. the absolute North, available zoom levels, a size of the image tiles, and a depth information encoded in a binary string. Supplied with `panoid` hash identifier a response to the HTTP request is metadata in JSON format. Important portion of the metadata is highlighted in Chaprer 3.

Base URL:

```
https://cbks1.google.com/cbk
```

Query parameters:

```
{'cb_client': 'apiv3',
 'dm': 1,
 'dmz': 0,
 'hl': 'en-US',
 'oe': 'utf-8',
 'output': 'json',
 'panoid': 'KzDzUS3ub-yrzbOLNomavw',              # <- panorama hash ID
 'pm': 0,
 'pmz': 0,
 'v': 4}
```

An example of full URL string

```
https://cbks1.google.com/cbk?dmz=0&dm=1&panoid=KzDzUS3ub-yrzbOLNomavw&h...
```

where ellipses denote rest of the string.

A.3 Metadata - temporal links

The second portion of the metadata contains temporal links to adjacent panorama locations. Supplied with hash identifier a response to the HTTP request is metadata in in serialized object. The response to the request is a serialized javascript object which does not exactly follow the JSON format, but can be converted to JSON performing the following steps; (i) remove the first line and (ii) replace each `',` with `'null,` if the comma `',` does not have a value on its left size. For example, the string `[3,4,1, , , [ , , ,[6]]]` would be converted to `[3,4,1,null,null,[null,null,null,[6]]]`.

Location of the temporal links within this data structure is discussed in Chaprer 3.

Base URL:

```
https://www.google.fr/maps/photometa/v1
```

Query parameters:

```
{'authuser': 0,
 'hl': 'en',
 'output': 'json',
 'pb': '!1m1!1smaps_sv.tactile!2m2!1sen!2sfr!3m3!1m2!1e2!2s' \
     'KzDzUS3ub-yrzbOLNomavw' \                  # <- panorama hash ID
     '!4m17!1e1!1e2!1e3!1e4!1e5!1e6!1e8!4m1' \
     '!1i48!5m1!1e1!5m1!1e2!6m1!1e1!6m1!1e2' \
}
```

An example of full URL string

```
https://www.google.fr/maps/photometa/v1?pb=%211m1%211smaps_sv.tactile...
```

A.4 Extraction of the temporal links

The code snipped below assumes that JSON was deserialized into a Python `data` object.

```
def extract_temporal_links(data):
    aux = data[1][0][5][1]  # an interesting part of the metadata

    # Get timestamps of available time machine panoramas
    tstamps = []                      # time stamps
    for x in aux[8]:
        tstamps.append(tuple(x[1]))  # (year, month)

    # Get corresponding panoID hashes
    pano_ids = [''] * len(tstamps)         # an empty string list alloc
    for j in range(1, len(tstamps) + 1):
        pano_ids[-j] = aux[3][0][-j][0][1] # pano_id hash string

    # List of tuples (pano_id, tstamp)
    return zip(pano_ids, tstamps)
```

For coherence note that in rare cases the time stamps are not available, in this case the metadata must be handled in a different way. More details can me found in the `streetget`package [25].

A.5 Getting panorama tiles

The HTTP request for panorama tile is supplied with the zoom level and location x, y of the tile within the panorama (see Figure 3.3 in Chapre 3). A response to the request is a jpg image.

Base URL:

```
url ='https://geo2.ggpht.com/cbk'
```

Query parameters:

```
{
    'output':   'tile',
    'zoom':     zoom,
    'x':        x,
    'y':        y,
    'panoid':   'KzDzUS3ub-yrzbOLNomavw'        # <- panorama hash ID
}
```

An example of full URL string

```
https://geo2.ggpht.com/cbk?x=25&y=2&output=tile&panoid=KzDzUS3ub-yrz...
```

A.6 Unpacking the depth binary sting

```
def getDepthData(data):
    encoded = data['model']['depth_map']

    # Decode
    encoded += '=' * (len(encoded) % 4)
    encoded = encoded.replace('-', '+').replace('_', '/')
    data = encoded.decode('base64').decode('zip')       # base64 encoded

    # Read header
    hsize = ord(data[0])                # header size in bytes
    fmt = Struct('< x 3H B')            # little endian, padding byte, 3x unsigned short int
    n_planes, width, height, offset = fmt.unpack(data[:hsize])

    # Read plane labels
    n = width * height
    fmt = Struct('%dB' % n)
    lbls = fmt.unpack(data[offset:offset+fmt.size])
    offset += fmt.size

    # Read planes
    fmt = Struct('< 4f')                # little endian, 4 signed floats
    planes = []
    for i in xrange(n_planes):
        unpacked = fmt.unpack(data[offset:offset+fmt.size])
        planes.append((unpacked[:3], unpacked[3]))
        offset += fmt.size

    return (width, height), lbls, planes
```

B - Author's Publications

"In science one tries to tell people, in such a way as to be understood by everyone, something that no one ever knew before. But in poetry, it's the exact opposite."

— Paul Dirac

B.1 Publications related to the thesis

B.1.1 Publications in impacted journals

Gronat, P., Obozinski, G., Sivic, J., Pajdla, T. (2016) Learning and calibrating per-location classifiers for visual place recognition. *International Journal of Computer Vision (IJCV), Springer USA, 10.1007/s11263-015-0878-x* [25-25-25-25]

B.1.2 Other publications - Conference papers

Gronat, P., Obozinski, G., Sivic, J., Pajdla, T. (2013) Learning and calibrating per-location classifiers for visual place recognition, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), p.907-914* [25-25-25-25]

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J. (2016) NetVLAD: CNN architecture for weakly supervised place recognition, R. Arandjelovic, P. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* [20-20-20-20-20]

B.1.3 Other publications - Workshop papers and techreports

Gronat, P., Sivic, J., Pajdla, T. (2012) Learning local distance functions for place recognition. *In Proceedings of the 17th Computer Vision Winter Workshop, CVWW2012.* [33-33-33]

Gronat, P., Havlena, M., Sivic, J., Pajdla, T. (2011) Building streetview datasets for place recognition and city reconstruction. *Tech. Rep. CTU-CMP-2011-16, Czech Tech University.* [25-25-25-25]

B.1.4 Other publications - Software

Gronat, P. (2015) `streetget`, *A small package for fetching Google StreetView data with easy to use CLI and Python API.* http://www.di.ens.fr/willow/research/streetget [100]

B.2 Other publications

B.2.1 Papers

Gronat P. (2010) Affine Correlation Method for Direct Vorticity, Skew and Displacement Estimation, *Topical Problems of Fluid Mechanics, Prague, Czech Republic.* [100]

Tarasov, A.S., Wang, C.N., Shen, C.P., Chang, J.M., Gronat, P., Lizana, P.C.A., Wagner, O.I. (2009) Flexural rigidity and persistence length of neuronal processes as a novel characteristic for describing neuropathological phenotypes. *International Workshop on Nonlinear Dynamics in Biological Systems and Soft-matter Biophysics, Academica Sinica, Taipei, Taiwan, R.O.C.* [14-14-14-14-14-14-14]

Gronat, P., Chen, C.F., (2008) Influence of Geometrical Properties and Stiffness of Foundation on Stress Behavior of Supported Plate and its Application in MEMS Sensor Parts. *Materials Symposia,Taichung, Taiwan, R.O.C.* [50-50]

B.2.2 Books

Gronat, P. (2011) Cylindrical Bending of a Plate on an Elastic Foundation. *Chapter in the book "Beams and Frames On Elastic Foundation 3", pages C376-C393, Technical University of Ostrava, ISBN 978-80-248-2257-0, Czech Republic.* [100]

Gronat P. (2009) 性基座板曲分析及三影像理. *Master Thesis, Chung Hua University, Taiwan R.O.C.* [100]

C - Citations of Author's work

Gronat, P., Obozinski, G., Sivic, J., Pajdla, T. (2016) Learning and calibrating per-location classifiers for visual place recognition, *International Journal for Computer Vision (IJCV), p.319-336* [cited by 41] [1]

Arandjelovic, R., & Zisserman, A. (2014, November). DisLocation: Scalable descriptor distinctiveness for location recognition. In Asian Conference on Computer Vision (pp. 188-204). Springer International Publishing.

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., &Sivic, J. (2016). NetVLAD: CNN architecture for weakly supervised place recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5297-5307).

Donoser, M., & Schmalstieg, D. (2014). Discriminative feature-to-point matching in image-based localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 516-523).

Lu, G., Yan, Y., Ren, L., Saponaro, P., Sebe, N., & Kambhamettu, C. (2016). Where am i in the dark: Exploring active transfer learning on the use of indoor localization based on thermal imaging. Neurocomputing, 173, 83-92.

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2015). NetVLAD: CNN architecture for weakly supervised place recognition. arXiv preprint arXiv:1511.07247.

Weyand, T., Kostrikov, I., & Philbin, J. (2016). Planet-photo geolocation with convolutional neural networks. arXiv preprint arXiv:1602.05314.

Tang, K., Paluri, M., Fei-Fei, L., Fergus, R., & Bourdev, L. (2015). Improving image classification with location context. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1008-1016).

Weyand, T., & Leibe, B. (2015). Visual landmark recognition from internet photo collections: A large-scale evaluation. Computer Vision and Image Understanding, 135, 1-15.

Le Barz, C., Thome, N., Cord, M., Herbin, S., & Sanfourche, M. (2014, September). Global robot ego-localization combining image retrieval and hmm-based filtering. In 6th Workshop on Planning, Perception and Navigation for Intelligent Vehicles (pp. 6-p).

Gopalan, R. (2015). Hierarchical sparse coding with geometric prior for visual geo-location. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2432-2439).

Li, X., Larson, M., & Hanjalic, A. (2015). Global-scale location prediction for social images using geo-visual ranking. IEEE Transactions on Multimedia, 17(5), 674-686.

Modolo, D., Vezhnevets, A., Russakovsky, O., & Ferrari, V. (2015, June). Joint calibration of

[1] Citations according to Google Scholar, 28.2.2017.

ensemble of exemplar svms. In 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 3955-3963). IEEE.

Fernando, B., Tommasi, T., & Tuytelaars, T. (2015). Location recognition over large time lags. Computer Vision and Image Understanding, 139, 21-28.

Torii, A., Dong, Y., Okutomi, M., Sivic, J., & Pajdla, T. (2014). Efficient Localization of Panoramic Images Using Tiled Image Descriptors. Information and Media Technologies, 9(3), 351-355.

Jin Kim, H., Dunn, E., & Frahm, J. M. (2015). Predicting Good Features for Image Geo-Localization Using Per-Bundle VLAD. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1170-1178).

Aubry, M., Russell, B. C., & Sivic, J. (2015). Visual geo-localization of non-photographic depictions via 2D-3D alignment. Visual Analysis and Geolocalization of Large-Scale Imagery.

Mousavian, A., Košecká, J., & Lien, J. M. (2015, May). Semantically guided location recognition for outdoors scenes. In 2015 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4882-4889). IEEE.

Le Barz, C., Thome, N., Cord, M., Herbin, S., & Sanfourche, M. (2015, September). Exemplar based metric learning for robust visual localization. In Image Processing (ICIP), 2015 IEEE International Conference on (pp. 4342-4346). IEEE.

Le Barz, C., Thome, N., Cord, M., Herbin, S., & Sanfourche, M. (2015). Absolute geo-localization thanks to Hidden Markov Model and exemplar-based metric learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 9-17).

Girdhar, R., Fouhey, D. F., Kitani, K. M., Gupta, A., & Hebert, M. (2016, March). Cutting through the clutter: Task-relevant features for image matching. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1-9). IEEE.

Said, S. H., Boujelbane, I., & Zaharia, T. (2014, September). Recognition of urban buildings with spatial consistency and a small-sized vocabulary tree. In 2014 IEEE Fourth International Conference on Consumer Electronics Berlin (ICCE-Berlin) (pp. 350-354). IEEE.

Charles, J., Pfister, T., Magee, D., Hogg, D., & Zisserman, A. (2015). Personalizing Human Video Pose Estimation. arXiv preprint arXiv:1511.06676.

Sivic, J. (2014). Visual search and recognition of objects, scenes and people (Doctoral dissertation, Ecole Normale Supérieure de Paris-ENS Paris).

Gopalan, R. (2015). Hierarchical Sparse Coding With Geometric Prior For Visual Place Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

Becattini, F., Seidenari, L., & Del Bimbo, A. (2016, June). Indexing Ensembles of Exemplar-SVMs with rejecting taxonomies. In Content-Based Multimedia Indexing (CBMI), 2016 14th International Workshop on (pp. 1-6). IEEE.

Sinha, S. N., Hedau, V., Zitnick, C. L., & Szeliski, R. (2016). A Memory Efficient Discriminative Approach for Location-Aided Recognition. In Large-Scale Visual Geo-Localization (pp. 279-298). Springer International Publishing.

Mousavian, A., & Kosecka, J. (2016). Semantic Image Based Geolocation Given a Map. arXiv preprint arXiv:1609.00278.

Liu, P., Yang, P., Wang, C., Huang, K., & Tan, T. (2016). A Semi-Supervised Method for

Surveillance-Based Visual Location Recognition.

Wang, J., Cheng, Y., & Feris, R. S. (2016). Walk and Learn: Facial Attribute Representation Learning from Egocentric Video and Contextual Data. arXiv preprint arXiv:1604.06433.

Choi, J., Larson, M., Li, X., Friedland, G., & Hanjalic, A. (2016). Where to be wary: The impact of widespread photo-taking and image enhancement practices on users' geo-privacy. arXiv preprint arXiv:1603.01335.

Kim, H. J. (2014). Place Recognition Using Reliable Features.

Mousavian, A., & Košecka, J. Semantically Aware Bag-of-Words for Localization.

Tapu, R., Mocanu, B., & Zaharia, T. (2016). A computer vision-based perception system for visually impaired. Multimedia Tools and Applications, 1-37.

Sattler, T., Havlena, M., Schindler, K., & Pollefeys, M., Large-Scale Location Recognition and the Geometric Burstiness Problem.

Li, X., Larson, M. A., & Hanjalic, A. (2016). Geo-distinctive Visual Element Matching for Location Estimation of Images. arXiv preprint arXiv:1601.07884.

Modolo, D., Vezhnevets, A., & Ferrari, V. Context Forest for Object Class Detection.

Kumar, D. (2016). Deep Learning Based Place Recognition for Challenging Environments.

Zisserman, A., Arandjelović, R., & Church, C. Advancing Large Scale Object Retrieval.

Zemene, E., Tariku, Y., Idrees, H., Prati, A., Pelillo, M., & Shah, M. (2017). Large-scale Image Geo-Localization Using Dominant Sets. arXiv preprint arXiv:1702.01238.

Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S., & Cremers, D. (2016). Image-based Localization with Spatial LSTMs. arXiv preprint arXiv:1611.07890.

Zamir, A. R., Hakeem, A., Van Gool, L., Shah, M., & Szeliski, R. (2016). Introduction to Large-Scale Visual Geo-localization. In Large-Scale Visual Geo-Localization (pp. 1-18). Springer International Publishing.

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., Sivic, J. (2016) NetVLAD: CNN architecture for weakly supervised place recognition, R. Arandjelovic, P. ***IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*** **[cited by 48]** [2]

Lin, T. Y., RoyChowdhury, A., & Maji, S. (2015). Bilinear cnn models for fine-grained visual recognition. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1449-1457).

Tolias, G., Sicre, R.,& Jégou, H. (2015). Particular object retrieval with integral max-pooling of CNN activations. arXiv preprint arXiv:1511.05879.

Razavian, A. S., Sullivan, J., Carlsson, S.,& Maki, A. (2016). [Paper] Visual Instance Retrieval with Deep Convolutional Networks. ITE Transactions on Media Technology and Applications, 4(3), 251-258.

Arroyo, R., Alcantarilla, P. F., Bergasa, L. M.,& Romera, E. (2016). Fusion and binarization of CNN features for robust topological localization across seasons. In IEEE/RSJ IROS.

Zheng, L., Yang, Y.,& Tian, Q. (2016). SIFT Meets CNN: A Decade Survey of Instance Retrieval. arXiv preprint arXiv:1608.01807.

Valigi, N. How can Deep Learning help Robotics and SLAM.

Durand, T., Thome, N.,& Cord, M. WELDON: Weakly Supervised Learning of Deep Convolutional Neural Networks.

Gordo, A., Almazan, J., Revaud, J.,& Larlus, D. (2016). End-to-end Learning of Deep Visual Representations for Image Retrieval. arXiv preprint arXiv:1610.07940.

Mohedano, E., Salvador, A., McGuinness, K., Marques, F., O'Connor, N. E.,& Giro-i-Nieto, X. (2016). Bags of Local Convolutional Features for Scalable Instance Search. arXiv preprint arXiv:1604.04653.

Schönberger, J. L., Price, T., Sattler, T., Frahm, J. M.,& Pollefeys, M. A Vote-and-Verify Strategy for Fast Spatial Verification in Image Retrieval.

Alzu'bi, A., Amira, A.,& Ramzan, N. (2016, August). Compact Root Bilinear CNNs for Content-Based Image Retrieval. In Image, Vision and Computing (ICIVC), International Conference on (pp. 41-45). IEEE.

Ustinova, E.,& Lempitsky, V. (2016). Learning Deep Embeddings with Histogram Loss. arXiv preprint arXiv:1611.00822.

Zhou, Q., Wang, C., Liu, P., Li, Q., Wang, Y.,& Chen, S. (2016). Distribution Entropy Boosted VLAD for Image Retrieval. Entropy, 18(8), 311.

Amato, G., Falchi, F., Gennaro, C.,& Vadicamo, L. (2016, October). Deep Permutations: Deep Convolutional Neural Networks and Permutation-Based Indexing. In International Conference on Similarity Search and Applications (pp. 93-106). Springer International Publishing.

Tang, P., Wang, X., Shi, B., Bai, X., Liu, W.,& Tu, Z. (2016). Deep FisherNet for Object Classification. arXiv preprint arXiv:1608.00182.

Lan, Z. (2016). Towards Usable Multimedia Event Detection from Web Videos (Doctoral dissertation, Disney Research).

[2] Citations according to Google Scholar, 28.2.2017.

De Nadai, M., Vieriu, R. L., Zen, G., Dragicevic, S., Naik, N., Caraviello, M., Hidalgo, C. A.,Sebe, N.,& Lepri, B. (2016, October). Are Safer Looking Neighborhoods More Lively?: A Multimodal Investigation into Urban Life. In Proceedings of the 2016 ACM on Multimedia Conference (pp. 1127-1135). ACM.

Markuš, N., Pandžić, I. S.,& Ahlberg, J. (2016). Learning Local Descriptors by Optimizing the Keypoint-Correspondence Criterion. arXiv preprint arXiv:1603.09095.

Verbeek, J., Machine learning solutions to visual recognition problems.

Bilen, C., Zepeda, J.,& Pérez, P. The CNN News Footage Datasets: Enabling Supervision in Image Retrieval.

Radenović, F., Tolias, G.,& Chum, O. (2016). CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples. arXiv preprint arXiv:1604.02426.

Weyand, T., Kostrikov, I., $ Philbin, J. (2016, October). Planet-photo geolocation with convolutional neural networks. In European Conference on Computer Vision (pp. 37-55). Springer International Publishing.

Gordo, A., Almazán, J., Revaud, J., $ Larlus, D. (2016, October). Deep image retrieval: Learning global representations for image search. In European Conference on Computer Vision (pp. 241-257). Springer International Publishing.

Razavian, A. S., Sullivan, J., Carlsson, S., $ Maki, A. (2016). Visual Instance Retrieval with Deep Convolutional Networks. ITE Transactions on Media Technology and Applications, 4(3), 251-258.

Lev, G., Sadeh, G., Klein, B., $ Wolf, L. (2016, October). RNN fisher vectors for action recognition and image annotation. In European Conference on Computer Vision (pp. 833-850). Springer International Publishing.

Zhao, Y., Wang, L., Gao, Z., Comor, I., Zhang, W., $ Zhou, L. (2016, November). Semi-Supervised Weight Learning for the Spatial Search Method in ConvNet-Based Image Retrieval. In Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on (pp. 1-8). IEEE.

Chadha, A., $ Andreopoulos, Y. (2016). Voronoi-based compact image descriptors: Efficient Region-of-Interest retrieval with VLAD and deep-learning-based descriptors. arXiv preprint arXiv:1611.08906.

Muñoz, J.A., Li, L.T., Dourado, Í.C., Nogueira, K., Fadel, S.G., Penatti, O.A., Almeida, J., Pereira, L.A., Calumby, R.T., dos Santos, J.A. and Torres, R.D.S., RECOD@ Placing Task of MediaEval 2016: A Ranking Fusion Approach for Geographic-Location Prediction of Multimedia Objects.

Brachmann, E., Krull, A., Nowozin, S., Shotton, J., Michel, F., Gumhold, S., $ Rother, C. (2016). DSAC-Differentiable RANSAC for Camera Localization. arXiv preprint arXiv:1611.05705.

Chen, J., Liang, J., Lu, H., Yu, S. I., $ Hauptmann, A. (2016). Videos from the 2013 Boston Marathon: An Event Reconstruction Dataset for Synchronization and Localization.

Leveau, V. (2016). Spatially Consistent Nearest Neighbor Representations for Fine-Grained Classification (Doctoral dissertation, Université Montpellier). Chicago

Zhong, Z., Su, S., Cao, D., Li, S., $ Lv, Z. (2016). Detecting ground control points via convolutional neural network for stereo matching. Multimedia Tools and Applications, 1-16.

Diba, A., Sharma, V., $ Van Gool, L. (2016). Deep Temporal Linear Encoding Networks. arXiv

preprint arXiv:1611.06678.

Cascianelli, S., Costante, G., Bellocchio, E., Valigi, P., Fravolini, M. L., $ Ciarfuglia, T. A. (2016, September). A robust semi-semantic approach for visual localization in urban environment. In Smart Cities Conference (ISC2), 2016 IEEE International (pp. 1-6). IEEE.

Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S., $ Cremers, D. (2016). Image-based Localization with Spatial LSTMs. arXiv preprint arXiv:1611.07890.

Tai, L., $ Liu, M. (2016). Deep-learning in Mobile Robotics-from Perception to Control Systems: A Survey on Why and Why not. arXiv preprint arXiv:1612.07139.

Gordo, A., Almazán, J., Revaud, J., $ Larlus, D. Learning global representations for image retrieval.

Zhang, H., Xue, J., $ Dana, K. (2016). Deep TEN: Texture Encoding Network. arXiv preprint arXiv:1612.02844. Chicago

Stylianou, A. (2016). Indoor Scene Localization to Fight Sex Trafficking in Hotels.

Zhang, G., Zeng, Z., Zhang, S., Zhang, Y., $ Wu, W. (2017). SIFT Matching with CNN Evidences for Particular Object Retrieval. Neurocomputing. Chicago

Lee, K., Lee, S., Jung, W. J., $ Kim, K. T. (2017). Fast and Accurate Visual Place Recognition Using Street-View Images. ETRI Journal, 39(1), 97-107.

Sharif Razavian, A. (2017). Convolutional Network Representation for Visual Recognition (Doctoral dissertation, KTH Royal Institute of Technology).

Ong, E. J., Husain, S., $ Bober, M. (2017). Siamese Network of Deep Fisher-Vector Descriptors for Image Retrieval. arXiv preprint arXiv:1702.00338. Chicago

Li, Y., Xu, Y., Wang, J., Miao, Z., $ Zhang, Y. (2017). MS-RMAC: Multi-Scale Regional Maximum Activation of Convolutions for Image Retrieval. IEEE Signal Processing Letters.

Chen, Z., Jacobson, A., Sunderhauf, N., Upcroft, B., Liu, L., Shen, C., ... $ Milford, M. (2017). Deep Learning Features at Scale for Visual Place Recognition. arXiv preprint arXiv:1701.05105. Chicago

Gronat, P., Havlena, M., Sivic, J., Pajdla, T. (2011) Building streetview datasets for place recognition and city reconstruction. ***Tech. Rep. CTU-CMP-2011-16, Czech Tech University.*** **[cited by 8]** [3]

Doersch, C., Singh, S., Gupta, A., Sivic, J., & Efros, A. (2012). What makes paris look like paris?. ACM Transactions on Graphics, 31(4).

Fang, Q., Sang, J., & Xu, C. (2013, October). Giant: Geo-informative attributes for location recognition and exploration. In Proceedings of the 21st ACM international conference on Multimedia (pp. 13-22). ACM.

Fang, Q., Sang, J., & Xu, C. (2014). Discovering geo-informative attributes for location recognition and exploration. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 11(1s), 19.

Doersch, C., Singh, S., Gupta, A., Sivic, J., & Efros, A. A. (2015). What makes Paris look like Paris?. Communications of the ACM, 58(12), 103-110.

Verstockt, S., Gerke, M., & Kerle, N. (2015). Geolocalization of Crowdsourced Images for 3-D Modeling of City Points of Interest. IEEE Geoscience and Remote Sensing Letters, 12(8), 1670-1674.

Bansal, A., & Singh, K. K. Storytelling Patches: Predicting Tourist Spots in a City.

Doersch, C. Data Analysis Project: What Makes Paris Look like Paris?.

董, 居秋彦, & 奥富正敏. (2013). BoF の分割表を用いた画像索による自己位置方位推定. 情理学会研究告 (コンピュタビジョンとイメジメディア (CVIM)), 1-6.

Shapiro, A. (2017). Street-level: Google Street View's abstraction by datafication. New Media & Society, 1461444816687293.

[3] Citations according to Google Scholar, 28.2.2017.

D - Beaux sites d'escalade de France

"The time you enjoy wasting is not wasted time."
— Bertrand Russell

A majority of my Ph.D. I spent in Paris, which became my home for last four years. During my studies, sometimes, it was very helpful to switch the brain off, leave my 24" screen behind and relax a bit. Paris is a beautiful city with dynamic live and lot of options for leisure. However, one thing is tough to find when living in Paris, the outdoor rock climbing. During the years, I found several rock climbing spots that are reachable within a couple of hours, some of them even reachable by a train.

In this appendix, I would like to share these spots with whoever wants to clear his head in nature while playing the chess with the rock. Finally, I would like to acknowledge my rope, quickdraws, harness and my climbing partners. Despite many dramatical falls we always survived with no injuries. Thank you!

Spots

Viaduc des Fauvettes (48.678423, 2.152275)

The Viaduct des Fauvettes is a disused railway bridge damaged during the WWII and renovated in 70's. It is located on the communes of Gometz-le-Châtel and Bures-sur-Yvette, in Essonne. The bridge is used as a training site for rock climbing and speleology. The spot can be reached from the center of Paris by RER and walk within about forty minutes. The routes are very well secured, mostly the 50m rope and eight quickdraws are needed, however, the longest route requires a 70m long rope.

Saint-Vaast-les-Mello (49.268537, 2.401343)

Close to the Saint-Vaast-les-Mello, there is a lovely abandoned limestone mine which is being used as a rock climbing site for a few decades. Despite being close to Paris, it is not very busy. It is even possible to camp there. The are routes up to 12 meters and all well secured. Few routes are tough and are difficult catch the fall when clipping the second quickdraw.

Les Andelys (49.252093, 1.383858)

This lovely city with a castle on the bank of the river Seine can be tracked back in the history up to the 6th century. West from the city at the river bank, between Les Andelys and Val-Saint-Martin

there is a huge white rock massive with more than one hundred secured routes. When visiting this area, it is worth mentioning few smaller spots lying around; Le Thuit, La Roquette, and Conelles. The last spot may be closed for few months during the year because of nesting of kites.

It is worth noting that the rock is sometimes fragile and you may expect unexpected falls or falling rocks, hence wearing a helmet is a good idea.

Saffres (47.371920, 4.581067)

Saffres is one of best spots close to Dijon or Auxerre, however it takes about two and half hours to get there by a car from Paris. Close to the climbing spot, there is a free camping place with tap water, toilets, and a fireplace. The routes are reasonably secured, the rock is solid and falls only on wet spots covered by vegetation. There are two other climbing spots in this region, Surgy and Clamecy, the topo can sometimes be found at the outdoor shop. When visiting this area, do not miss an opportunity to visit Vézelay Abbey monastery, one of the four major routes through France for pilgrims going to Santiago de Compostela in Galicia, in the north-western corner of Spain.