Visual Discovery of Landmarks and their Details in Large-Scale Image Collections

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der
RWTH Aachen University zur Erlangung des akademischen Grades
eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Tobias Weyand
aus Siershahn, Deutschland

Berichter: Prof. Dr. Bastian Leibe
Prof. Dr. Ondrej Chum

Tag der mündlichen Prüfung: 30.09.2016

Selected Topics in Computer Vision

herausgegeben von
Prof. Dr. Bastian Leibe
Lehr- und Forschungsgebiet Informatik 8
(Computer Vision)
RWTH Aachen University

Band 2

Tobias Weyand

Visual Discovery of Landmarks and their Details in Large-Scale Image Collections

Shaker Verlag
Aachen 2016

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de.

Zugl.: D 82 (Diss. RWTH Aachen University, 2016)

Printed in Germany.

ISBN 978-3-8440-4882-7
ISSN 2198-3372

Shaker Verlag GmbH • P.O. BOX 101818 • D-52018 Aachen
Phone: 0049/2407/9596-0 • Telefax: 0049/2407/9596-9
Internet: www.shaker.de • e-mail: info@shaker.de

Abstract

With their rapid growth in recent years, Internet photo collections have become an invaluable repository of visual data. In particular, they provide detailed coverage of the world's landmark buildings, monuments, sculptures, and paintings. This wealth of visual information can be used to construct landmark recognition engines that can automatically tag a photo of a landmark with its name and location. Landmark recognition engines rely on clustering algorithms that are able to group several millions of images by the buildings or objects they depict.

This grouping problem is very challenging since the massive amount of Internet images requires efficient and highly parallel algorithms, and the appearance variability of buildings caused by viewpoint, weather and lighting changes requires robust image similarity measures. Most importantly, it is critical to define a clustering criterion that results in meaningful object clusters. The *Iconoid Shift* algorithm we present in this thesis uses a very intuitive definition: It represents each object by an iconic image, or *Iconoid*, which is the image that has the highest overlap with all other images of the object. The object cluster is then the set of all images that have a certain minimum overlap with the Iconoid. We find Iconoids by performing mode search using a novel distance measure based on image overlap that is more robust to viewpoint and lighting changes than traditional image distance measures. We propose efficient parallel algorithms for performing this mode search. In contrast to most previous algorithms that produced a hard clustering, Iconoid Shift produces an overlapping clustering and thus elegantly handles images showing multiple nearby landmarks by assigning them to multiple clusters.

The increasing density of Internet photo collections allows us to go a step further and to even discover sub-structures of buildings such as doors, spires, or facade details. To this end, we present the *Hierarchical Iconoid Shift* algorithm that, instead of a flat clustering, produces a hierarchy of clusters, where each cluster represents a building sub-structure. This algorithm is based on a novel hierarchical variant of Medoid Shift

that tracks the evolution of modes through scale space by continuously increasing the size of its kernel window.

But which objects can a landmark recognition engine built by automatically mining Internet photo collections recognize? And how to construct such a system such that it is efficient and achieves high recognition performance? To answer these questions, we perform a large-scale evaluation of the different components of a landmark recognition system, analyzing how different choices of components and parameters affect performance for different object categories such as buildings, paintings or sculptures.

As a final contribution, we consider a practical problem of the image retrieval methods that our algorithms are based on: a large fraction of the photos in Internet photo collections has visible watermarks, timestamps, or frames embedded in the image content. These artifacts often cause false-positive image matches. We present a simple but highly efficient and effective method to detect such matches and thus prevent errors in landmark discovery and recognition.

Zusammenfassung

Durch ihr rapides Wachstum in den letzten Jahren sind Foto-Sharing-Webseiten zu einer sehr wertvollen Quelle visueller Daten geworden. Sie bieten eine Fülle von Fotos der Denkmäler, Monumente, Skulpturen und Gemälde der Welt. Aus diesem Reichtum visueller Informationen lassen sich Landmark Recognition Engines konstruieren, die ein Foto von einem Denkmal automatisch mit seinem Namen und Standort versehen können. Landmark Recognition Engines bauen auf Clustering Algorithmen auf, die Millionen von Fotos nach den abgebildeten Gebäuden oder Objekten gruppieren.

Dieses Gruppierungsproblem ist sehr komplex, da die massive Menge an Internetfotos effiziente und hochgradig parallele Algorithmen erfordert. Zudem sorgen verschiedene Blickwinkel, Tageszeiten und Wetterbedingungen für starke Veränderungen im Aussehen der Gebäude, weshalb robuste Ähnlichkeitsmaße für Bilder benötigt werden. Letztlich ist es wichtig, ein Clustering-Kriterium zu definieren, das sinnvolle Objektcluster ergibt. Der *Iconoid Shift* Algorithmus, den wir in dieser Dissertation präsentieren, verwendet eine sehr intuitive Definition von Clustern: Er repräsentiert jedes Objekt durch ein ikonisches Bild, oder *Iconoid*. Der Iconoid eines Objekts ist das Bild, welches die größte Gesamtüberlappung mit allen anderen Bildern dieses Objekts hat. Das Cluster eines Iconoid ist die Menge aller Bilder, die eine gewisse Mindestüberlappung mit dem Iconoid haben. Wir finden Iconoids durch Mode Search, unter Verwendung eines neuen Ähnlichkeitsmaßes, welches auf der Bildüberlappung basiert und daher robuster bezüglich Veränderungen in Blickwinkel und Beleuchtung ist als traditionelle Bildähnlichkeitsmaße. Wir schlagen effiziente parallele Algorithmen für diese Mode Search vor. Im Gegensatz zu den meisten vorherigen Algorithmen zum Clustern von Bildern, welche ein hartes Clustering produzieren, erzeugt Iconoid Shift ein überlappendes Clustering und kann daher elegant mit Bildern umgehen, die mehrere benachbarte Denkmäler zeigen, indem diese Bilder mehreren Clustern zugeordnet werden.

Die steigende Dichte an Fotos auf Foto-Sharing-Webseiten erlaubt es uns, noch einen Schritt weiter zu gehen und sogar Sub-Strukturen von Gebäuden, wie Türen, Türme oder Fassadendetails aufzufinden. Hierzu präsentieren wir den *Hierarchical Iconoid Shift* Algorithmus, der statt eines flachen Clustering eine Hierarchie von Clustern produziert, in der jedes Cluster ein Gebäude oder eine Sub-Struktur repräsentiert. Dieser Algorithmus basiert auf einer neuen Variante von Medoid Shift, die die Evolution von Modi im Scale Space verfolgt während sie kontinuierlich die Größe des Kernel-Fensters erhöht.

Aber welche Arten von Objekten kann eine ausschließlich durch Mining von Foto-Sharing-Webseiten erstellte Landmark Recognition Engine erkennen? Und wie konstruiert man eine solche Engine, sodass sie möglichst effizient ist und gleichzeitig eine gute Erkennungsrate erreicht? Um diese Fragen zu beantworten, führen wir eine großangelegte Evaluation einer Landmark Recognition Engine durch und zeigen auf, wie die Wahl ihrer Komponenten und Parameter die Erkennungsrate verschiedener Arten von Objekten, wie Gebäuden, Gemälden oder Skulpturen beeinflusst.

Unser letzter Beitrag bezieht sich auf ein praktisches Problem der Image Retrieval Methoden, auf denen unsere Algorithmen basieren: Ein großer Anteil von Internetfotos hat sichtbare Wasserzeichen, Datums- und Uhrzeitinformationen oder Rahmen, die in den Bildinhalt eingebettet sind. Diese Artefakte verursachen oft falsch-positive Bildmatches. Wir präsentieren eine einfache, aber hocheffektive und effiziente Methode, um solche Matches zu detektieren und dadurch Fehler beim Auffinden und Erkennen von Denkmälern zu verhindern.

Acknowledgments

This thesis would not have been possible without the invaluable support I received from many important people.

I would like to express my gratitude to my advisor Bastian Leibe for giving me the opportunity to work at his group, for providing his guidance, insight and ideas and for helping me grow my skills as a researcher and teacher. I would also like to thank Ondra Chum for agreeing to be the second examiner for my thesis.

My time at the Computer Vision group at RWTH Aachen would not have been half as fun without my amazing colleagues Alex Hermans, Aljosa Osep, Claudia Prast, Dennis Mitzel, Dora Kontogianni, Esther Horbert, Francis Engelmann, Georgios Floros, Ishrat Badami, Lucas Beyer, Michael Kramp, Patrick Sudowe, Stefan Breuers, Umer Rafi and Wolfgang Mehner as well as my friends and collaborators from the Computer Graphics Group Arne Schmitz, Robert Menzel, Sven Middelberg and Torsten Sattler. I would like to thank my excellent students Siyu Tang, Christian Kalla, Jan Hosang, Archana Kumari, Lukas Nack and Chih-Yun Tsai!

Dennis, thank you for being the most amazing office mate, and for always keeping me informed on the latest offers on mydealz. Georgios, you are THE BEST! Patrick, thanks for being an awesome partner in our group's sysadmin team. I will never forget the spontaneous fixing sessions we had when Urquell had crashed once again. Aljosa, much thanks for being such good office mate and climbing partner! I enjoyed the many discussions we had, whether they were about machine learning or about whether gcc should stop complaining so much. Dora, I enjoyed working with you during my final year. I am sure to have passed the torch of landmark recognition into safe hands! Ishrat, I will keep fond memories of all the interesting conversations over Samosas at Akl and the fun electronics tinkering sessions in the lab. Torsten, thank you for the many many productive and insightful discussions both in the "City Reconstruction" meeting and over lunch at our favorite restaurant Lara. I would like to thank Dog, our office stuffed dog, for his always entertaining pranks and for even showing up at my goodbye party

sporting sunglasses and a hat! I probably wouldn't be where I am without Thomas Deselaers, my long-time Hiwi advisor and diploma thesis advisor. I would like to thank him for piquing my interest in computer vision and machine learning and for all his guidance and help throughout the years.

I would also like to thank the RWTH HPC cluster team, for providing easy access to computational resources for the whole university, and for their friendly support even when my jobs were misbehaving and slowing down the jobs of other users.

Another big thank you goes to my proofreaders Aljosa, Dora, Ishrat, Michael and Torsten and to everyone who helped with the ground truth annotations for the landmark recognition study in Ch. 6.

I would like to thank Google for supporting my research with a Google faculty research award. In particular, I would like to thank Hartwig Adam, for giving me the opportunity to do two internships at the Google Computer Vision Team in LA, as well as my former intern hosts and now colleagues Marco Andreetto and Yuan Li.

A very heartfelt thank you goes to my wife Sammi for her support during the final stages of my thesis! I would like to thank my uncle Christoph for the countless hours of playing with construction kits, steam machines and electronics and sparking my excitement for science and technology. Finally, I would like to thank my parents, without whom none of this would have been possible, for allowing me to play with my computer way too much when I was a child and for enabling me to pursue a career in computer science.

Contents

1 Introduction

1.1 Motivation

The Internet has become a source of an unfathomable and ever-growing amount of visual information. In particular, photo sharing websites like Flickr have extensive imagery of famous buildings and historic monuments, but also a surprisingly dense coverage of smaller tourist attractions, like individual paintings or statues, and even street art. New photos and are being uploaded every second at a steadily accelerating pace. For example, on an average day in 2013, 1.6M publicly available photos were posted on Flickr[1], 55M photos were posted on Instagram[2], 214M photos were uploaded to Google Plus[3], and 350M photos were posted on Facebook[4]. This growth is mostly due to the explosive spread of smartphones in recent years. In 2013 alone, almost 1B smartphones were sold[5]. Consequently, billions of people now carry Internet-connected cameras in their pockets that they use to capture and share visual impressions of the world around them.

Computer Vision research has begun to leverage this vast amount of visual data, e.g. to automatically learn to recognize objects or to construct 3D models of the landmark buildings and monuments of the world. This research has enabled applications such as automatic photo annotation, visual recognition and search, and interactive tour guides. A prerequisite for these applications is the ability to automatically discover objects in large and unstructured collections of photos. This is the task we address in this thesis. Several approaches for unsupervised object discovery in large photo collections have already been proposed in recent years. These approaches view the task as a clustering problem and aim to divide the image collections into groups of photos of individual buildings or objects. Despite the considerable progress in this line of work, several open

[1] http://flickr.com/photos/franckmichel/6855169886/
[2] http://instagram.com/press/ (Oct. 14 2013)
[3] http://googleblog.blogspot.de/2013/10/google-hangouts-and-photos-save-some.html
[4] http://internet.org/efficiencypaper
[5] https://www.gartner.com/newsroom/id/2665715

problems remain. For example, it is not clear how an object cluster should be defined and what photos it should contain. The grouping criteria used in previous approaches did not have a notion of an object and thus these approaches tend to under-segment or over-segment the image collection. Moreover, most of the algorithms used in landmark clustering so far produce a *hard* clustering, meaning each image can only belong to one cluster. Therefore, if an image shows two landmarks, it is not clear which cluster to assign it to. Furthermore, when a landmark has interesting sub-structures such as architectural details, they cannot form their own clusters, since their photos will be contained in the cluster of the landmark. For example, the gates of Notre Dame cathedral in Paris are interesting objects on their own, and they in turn contain several statues that should also be viewed as individual objects. It therefore seems that a *hierarchical*, *soft* clustering would be more suitable than a *flat*, *hard* one. The coverage of the world's landmarks on Internet photo collections is dense enough to enable such a detailed clustering, but this potential has so far been ignored.

In this thesis, we set out to address these problems: We propose an intuitive definition of object clusters where each object is represented by an iconic image, or *Iconoid*, and each cluster simply contains all images that overlap with the Iconoid by more than a certain amount. This definition naturally allows for overlapping clusters and can thus also handle close-by landmarks and even sub-structures elegantly. An Iconoid is an image that has maximum mutual *overlap* with all other images of the same object, i.e., it is a view that is most favored by photographers. We view the task of object discovery as a *mode estimation* problem and define a distance measure between images based on their overlap. Iconoids are modes w.r.t. this distance measure and represent views that are most favored by photographers.

We present an object discovery algorithm called *Iconoid Shift* that efficiently solves this mode estimation problem by propagating image overlaps through locally explored matching graphs. Building on this, we present a hierarchical image clustering algorithm, called *Hierarchical Iconoid Shift*, that performs mode search on all scales and constructs dendrograms that represent the detail hierarchies of landmarks. These algorithms represent the main contribution of this thesis.

As an additional contribution, we shift our attention to a particular application of the resulting clustering, namely landmark recognition. Using a dataset of 500k photos of Paris, we first investigate what objects can be discovered by a landmark discovery algorithm. We then analyze each component of a start-of-the-art landmark recognition engine and point out the challenges that arise at each stage of the pipeline. Moreover, we evaluate how successfully objects of different categories can be discovered in the photo collection and recognized in a new query photo.

We believe that this thesis both advances the start-of-the-art in object discovery in Internet photo collections as well as increases our understanding of what these approaches can achieve.

1.2 Contributions

This thesis makes the following contributions:

- We present an algorithm called *Iconoid Shift* that casts the task of landmark discovery as a *mode search* problem. Based on Medoid Shift (Sheikh et al., 2007), Iconoid Shift searches for iconic views, called *Iconoids*, that maximize the *mutual overlap* with other images of the same object. We present efficient algorithms for performing this mode search in local matching graphs and a parallelization scheme that allows applying the algorithm to web-scale photo collections.

- Taking inspiration from scale space theory (Witkin, 1984), we present a hierarchical clustering algorithm called *Hierarchical Medoid Shift* (HMS) that tracks the evolution of density maxima across different scales. The result is a set of dendrograms that describe the density structure of the dataset. In contrast to previous approaches, HMS increases the scale quasi-continuously and is thus completely *parameter-free*.

- Applying HMS to landmark discovery, we present the *Hierarchical Iconoid Shift* (HIS) algorithm that is able to find interesting objects at all scales. HIS discovers architectural details such as gates or spires as well as the landmarks they belong to and organizes them in a cluster hierarchy. We present algorithms for performing this clustering efficiently and in a highly distributed manner.

- We perform a detailed evaluation of landmark recognition based on Internet photo collections. Our analysis shows how many and what kinds of objects can be discovered in Internet photo collections and how different choices for the components of such a system affect the final recognition performance. In particular, we consider different object categories like paintings, building details or museum artifacts and point out the respective challenges that exist for them.

- We address the problem of false-positive image matches caused by watermarks, timestamps and frames (WTFs) often found in Internet photo collections. We develop an efficient detector for such matches based on the spatial distribution of matching image regions and show that it is effective in preventing clusters of different objects from being merged due to false-positive matches.

- We have collected a large-scale dataset of 500k photos of Paris from Flickr and Panoramio and made it publicly available[6]. Moreover, we have published ground truths for the tasks of landmark clustering and landmark recognition that we created by investing significant manual annotation effort. These ground truths will enable other researchers to evaluate their methods in a large-scale setting and compare them to ours.

[6] http://www.vision.rwth-aachen.de/data/paris500k

1.3 Structure of the Thesis

This thesis is structured as follows:

In **Chapter 2**, we begin by discussing previous work. We first look at different definitions of iconic images from both cognitive psychology and computer vision and then review different approaches for finding iconic images for entire object categories as well as for specific objects. Then, we consider previous work in object clustering and compare it to our work. Finally, we discuss different approaches to landmark recognition based on image retrieval, classification and pose estimation, respectively.

Chapter 3 first introduces the basic techniques that our work is based on. We review image matching with local features, image retrieval using bags-of-visual-words and inverted files, spatial verification using RANSAC and the construction of matching graphs. We then introduce the Paris 500k dataset that we use throughout the thesis. As a first experiment, we compare two landmark discovery algorithms based on spectral clustering (Philbin and Zisserman, 2008) and min-hash (Chum and Matas, 2010), respectively, which brings to light the challenges of landmark discovery we address with the algorithms presented in the following chapters. This comparison was originally presented in Weyand et al. (2010).

In **Chapter 4**, we present the *Iconoid Shift* algorithm for automatic landmark building discovery in large, unstructured image collections. Casting the task as a *mode estimation* problem, Iconoid Shift searches for views that have *maximal mutual homography overlap* with images in their neighborhood. These modes, called *Iconoids*, are the images that show the respective building or object from the most central, iconic view. Because Iconoid Shift is based on Medoid Shift (Sheikh et al., 2007), it inherits its intuitive parameters and its well-studied properties such as guaranteed convergence. Because our algorithm only performs a local exploration of the matching graph, it is more efficient than approaches that require the construction of the matching graph of the whole dataset in advance and can easily be parallelized to run on hundreds of machines simultaneously. Therefore, the approach is applicable for large-scale analysis of photo collections, which we demonstrate on the Paris 500k dataset as well as a dataset of 459k images of Barcelona. This chapter is based on research originally presented in Weyand and Leibe (2011).

Building on this, **Chapter 5** introduces the *Hierarchical Iconoid Shift* (HIS) algorithm. While previous landmark discovery algorithms were mainly focused on buildings, HIS also discovers *details* like portals, statues or windows. Instead of a *hard*, *flat* clustering, HIS produces a *soft*, *hierarchical* clustering that represents each building as a dendrogram of overlapping clusters where each cluster corresponds to a building part or detail. As the basis for HIS, we introduce the *Hierarchical Medoid Shift* (HMS) algorithm that is based on Medoid Shift. Taking inspiration from Scale Space theory (Witkin, 1984), HMS tracks the evolution of density maxima using mode search while continuously increasing the kernel scale. The result is a set of dendrograms describing the density structure of

the input data. HMS is completely parameter-free, has the same complexity as Medoid Shift and is easy to parallelize. HIS inherits the desirable properties of HMS and applies it to clustering Internet photo collections. We evaluate HIS on 800k images of 34 landmarks and show that it can extract an often surprising amount of detail and structure that can be applied, e.g., to provide a mobile user with more detailed information on a landmark or even to extend the landmark's Wikipedia article. The chapter is based on research originally presented in Weyand and Leibe (2013).

In **Chapter 6**, we then consider the application of a clustering returned by a landmark discovery algorithm for landmark recognition. The task here is to identify photographed buildings or objects in query photos and to provide the user with relevant information on them. The process of building such a landmark recognition system typically consists of three steps: (i) clustering large amounts of images by the objects they depict; (ii) determining object names from user-provided tags; and (iii) building a robust, compact, and efficient recognition index. Based on an Iconoid Shift clustering of Paris 500k, we analyze each component of this pipeline in order to answer the following questions: How many and what kinds of objects can be discovered automatically? How to best use the resulting image clusters to recognize the object in a query? How can the object be efficiently represented in memory for recognition? How reliably can semantic information be extracted? And finally: What are the limiting factors in the resulting pipeline from query to semantics? We evaluate how different choices of methods and parameters for the individual pipeline steps affect overall system performance and examine their effects for different query categories such as buildings, paintings or sculptures. The chapter is based on research originally published in Weyand and Leibe (2015).

Finally, in **Chapter 7**, we address a problem frequently encountered in computer vision approaches based on Internet photos. An increasing number of photos in Internet photo collections comes with watermarks, timestamps, or frames (in the following called *WTFs*) embedded in the image content. In image retrieval and matching such *WTFs* often cause false-positive matches, which in turn harm image clustering approaches by causing clusters of different buildings to be joined into one. This affects applications like landmark recognition or large-scale structure-from-motion, which rely on clean building clusters. We present a simple, but highly effective detector for such false-positive matches. Given a matching image pair with an estimated homography, we first determine photoconsistent regions in both images. Exploiting the fact that *WTFs* typically appear near the border, we build a spatial histogram of the consistent regions and apply a classifier to decide whether the match is due to a WTF. This approach is general enough to recognize a large variety of watermarks, timestamps, and frames and is efficient enough to be applied in large-scale image clustering and retrieval settings. We collected a challenging dataset of WTFs found in Internet photo collections and demonstrate the efficiency and effectiveness of our detector on this dataset as well as in actual image clustering applications. This chapter is based on research we originally presented in Weyand et al. (2015).

Note: *The thesis is based on the technical contributions of my respective first author publications (Weyand and Leibe, 2011, 2013, 2015; Weyand et al., 2010, 2015). Several images and text passages of this thesis are taken from these articles. However, additional content has been added in order to provide deeper insights into the approaches.*

2 State of the Art

The task we set out to solve in this thesis is finding interesting objects in large-scale image collections. Ideally, each object should be represented by a cluster of all photos showing the object as well as an iconic image. In this section, we put our work in context and discuss previous work on iconic images and image clustering. We first compare different works from cognitive psychology that have studied which views humans perceive as iconic. Then, we take a look at how computer vision research has approached the problem of finding iconic images automatically. While early work has been focused on finding iconic views of single, isolated objects, later work has harnessed the large amount of photos available online to find iconic images of object categories as well as specific objects like landmark buildings. We then discuss related work on object clustering, focusing on the underlying clustering algorithms. Finally, we discuss related work in landmark recognition, which is a common application of object clusterings. Approaches to this task can be divided into three categories: Image retrieval, classification and pose estimation.

2.1 Iconic Images

"What makes an image or view of an object *iconic* or *canonical*?" This question has been the subject of research both in psychology and computer vision. In the following, we take a look at ideas from both fields and discuss how they relate to another.

2.1.1 Cognitive Psychology

In their seminal work, Palmer et al. (1981) analyzed which views people prefer when photographing an object, which views they see when imagining an object, which views they most easily recognize an object from and which views they subjectively prefer. The participants of the study consistently chose the same views for each of these four tasks. Palmer et al. then formulated two hypotheses for what makes a view canonical:

The *frequency* hypothesis states that the view that an object is seen from most often is the canonical one, and the *maximal information* hypothesis states that the view that reveals the most information on the structure of the object is the canonical one. In contrast with Palmer et al. (1981), Blanz et al. (1999) found that *maximal information* views are preferred when photographing the object, while the mental images people have of objects are usually from straight frontal or side views. The experiments of Mezuman and Weiss (2012) showed that the views that Palmer et al. (1981) identified as canonical for a set of objects were also the views from which these objects were most frequently depicted in Internet photos, which implies that the *frequency* hypothesis holds for Internet photos. However, Mezuman and Weiss (2012) also found that Palmer et al.'s *frequency* and *maximal information* hypotheses do not generally hold for all object categories. Weinshall and Werman (1997) propose two criteria for canonical images: *likelihood*, which is the probability of seeing a certain view of an object, and *stability*, meaning how little an object changes in appearance when changing the viewpoint. For example, a frontal view of a building is more stable than a corner view. They proved that if viewing directions are uniformly distributed, *likelihood* and *stability* are actually identical. This, however, is not the case for internet photo collections.

The definition of iconic images in our work also follows the *frequency* definition of Palmer et al. (1981): our Iconoid Shift algorithm selects the views that have the highest total *overlap* with all other views of the same object. Because of the photographer bias in Internet photo collections (Mezuman and Weiss, 2012), these views form modes in the view distribution, which we find using mode search.

2.1.2 Computer Vision

Early work on finding iconic images in computer vision has focused on finding canonical views for single, isolated objects. Denton et al. (2004) propose an approach for selecting a small set of canonical views of a 3D object by finding views that are most similar to the views around them, effectively maximizing the *stability* criterion of Weinshall and Werman (1997). In contrast to previous definitions, Hall and Owen (2005) argue that the front, side and top views of an object should be considered canonical, and that these are the *least likely* views. They propose an algorithm that selects three orthogonal views that explicitly minimizes the *likelihood* criterion.

More recently, several approaches have used large amounts of photos from the Internet to find representative images for an object category (Berg and Berg, 2009; Jing and Baluja, 2008; Jing et al., 2007; Raguram and Lazebnik, 2008) or iconic images for specific objects like landmark buildings (Avrithis et al., 2010; Crandall et al., 2009; Kennedy and Naaman, 2008; Raguram et al., 2011; Simon et al., 2007; Yang et al., 2011). In the following we will consider these approaches in detail.

Iconic Images of Object Categories. Several papers have addressed the task of generating a set of iconic images that can serve as a visual *summary* for an object category.

An application of this is summarizing the results of an image search engine into a diverse set of photos. Jing et al. (2007) retrieve photos of an object category from Google images, build a *matching graph* by pairwise SIFT matching, and find the image with the highest *valence* in each connected component. The result is the image that matches the most other images, which can be viewed as an approximation of the *frequency* criterion (Palmer et al. (1981)). Berg and Berg (2009) collect large numbers of images for different categories and first apply heuristic filters to select images that fulfill two criteria: (i) The object should clearly stand out from the background. (ii) Photos should be dissimilar from photos of other categories. They then cluster the remaining images with k-Medoids using a distance measure based on the match score of local geometric blur features and select the resulting cluster centers as iconic images. Raguram and Lazebnik (2008) cluster the GIST descriptors (Olivia and Torralba, 2001) of the input Flickr photos using k-Means to first create groups of similar images. To incorporate semantic information, they intersect this clustering with a PLSA clustering based on the image tags to obtain a joint visual-semantic clustering. Unlike most other approaches, they select the iconic images for each cluster only using a quality function based on color distribution, blur, etc. and not by their similarity with other photos. Motivated by their finding that canonical views are the views from which objects are most frequently photographed, Mezuman and Weiss (2012) propose an algorithm to extract canonical views for object categories by finding modes in the distribution of their GIST descriptors. Finally, Jing and Baluja (2008) find representative images by applying PageRank (Page et al., 1999) to the image matching graph to find well-connected images.

Iconic Images of Specific Objects. Approaches for finding iconic images for *specific* objects are typically based on tourist photos from Internet photo collections such as Flickr. Raguram et al. (2011) first cluster the GIST descriptors of the images with k-means, then build a matching graph from the images closest to the cluster centers by performing SIFT matching followed by estimating epipolar geometries. The images that have the highest number of inliers with their adjacent images in the graph are selected as iconic images. This is similar to the *valence* criterion (Jing et al., 2007), but here the incident edges of an image are weighted by the number of inliers. Simon et al. (2007) formulate an objective function that explicitly incorporates *likelihood* (Palmer et al. (1981)) and *orthogonality* (no two canonical views may be too similar). Here, likelihood is formulated as the number of 3D points an image shares with other images in a structure-from-motion reconstruction, which is related to both the weighted valence criterion from Raguram et al. (2011) as well as the *overlap* measure we use in this work. Yang et al. (2011) apply PageRank on a matching graph constructed by SIFT matching similar to Jing and Baluja (2008) to find representative images of landmarks. Kennedy and Naaman (2008) first cluster the input images based on global descriptors using k-Means and then determine iconic images for each cluster based on (i) their dissimilarity to images from other clusters (similar to Berg and Berg (2009)), (ii) their closeness to

the cluster center and (iii) the valence criterion based on a matching graph built by a pairwise matching of SIFT features. Crandall et al. (2009) build the full matching graph, weight each edge by the number of homography inliers, segment it using spectral clustering (Ng et al. (2001)) and select the image with the highest weighted valence in each cluster. Cao and Snavely (2013) also build the matching graph of the input image collection and then select a set of iconic images by computing its *dominating set*, i.e., a minimal set of images such that each image in the dataset is adjacent to at least one image in this set. This strategy selects images that match as many other images as possible. Avrithis et al. (2010) use Kernel Vector Quantization (Tipping and Schölkopf, 2001), which determines a minimal subset of photos such that each photo in the original set has at least a certain minimum number of inliers with at least one photo in the subset. Both the approaches by Cao and Snavely (2013) and Avrithis et al. (2010) therefore seek a set of iconic images that *covers* the dataset. In Chapter 6.6, we use these two approaches for summarizing landmark clusters and find that they yield comparable results.

2.1.3 Discussion

In summary, most approaches for finding iconic images from Internet photo collections optimize criteria related to *frequency* (Palmer et al. (1981)), sometimes combined with additional criteria like orthogonality or aesthetic quality. A popular way of implementing this definition is the *valence* criterion (Crandall et al. (2009); Frahm et al. (2010); Jing et al. (2007); Kennedy and Naaman (2008); Philbin et al. (2011); Raguram et al. (2011); Zheng et al. (2009)). We argue that the valence criterion is not optimal, because an image that has many local feature matches with other images is not necessarily the best view, since SIFT features are designed to be viewpoint invariant. Thus, even if an image does not show a frontal, centered view of an object, it can still have a high number of matches. Another problem of the valence criterion is that it prefers textured image regions since they have more SIFT features than uniform surfaces. This gives views revealing more texture or detail a higher likelihood of being selected as the iconic view. Finally, the valence criterion only considers views directly connected to the current one in the matching graph. However, even images that are not directly adjacent might have similar views, e.g., if drastic lighting changes prevent the formation of a direct match. Our approach propagates image overlap through the matching graph, ensuring that all overlapping images are discovered. Moreover, our approach is not biased towards textured regions because our distance measure is a function of image overlap, not of the number of inliers.

2.2 Object Clustering

The task of object clustering is to group images into clusters corresponding to individual buildings or objects. Applications of this line of research include landmark recognition (Avrithis et al. (2010); Gammeter et al. (2009, 2010); Quack et al. (2008); Raguram et al. (2011); Yang et al. (2011); Zheng et al. (2009), search result summarization (Berg and Berg (2009); Jing and Baluja (2008); Jing et al. (2007); Kennedy and Naaman (2008); Raguram and Lazebnik (2008)), scene summarization (Epshtein et al. (2007); Simon et al. (2007)), object mining (Chum and Matas (2010); Chum et al. (2009); Crandall et al. (2009); Philbin and Zisserman (2008); Philbin et al. (2011)), 3D reconstruction (Agarwal et al. (2009); Frahm et al. (2010); Raguram et al. (2011); Snavely et al. (2006, 2008a)), and building interactive tour guides (Papadopoulos et al. (2010); Snavely et al. (2006, 2008a)).

To summarize web image collections, Kennedy and Naaman (2008); Raguram and Lazebnik (2008) simply cluster global image descriptors using k-means. Berg and Berg (2009) use k-medoids and a similarity on geometric blur features. Most approaches first build a matching graph of the image collection by performing a pairwise SIFT matching. Subsequently, they use various graph clustering algorithms to partition the matching graph into clusters that should ideally correspond to individual landmarks or objects. While early approaches produced a *hard* clustering, where each image only belongs to one cluster, newer approaches have shifted to producing a *soft* clustering, where an image can belong to several overlapping clusters. *Hard* clustering algorithms used for image clustering include Hierarchical Agglomerative Clustering (Gammeter et al. (2009, 2010); Quack et al. (2008); Zheng et al. (2009)), Spectral Clustering (Crandall et al. (2009); Philbin and Zisserman (2008)), Connected Components Analysis (Agarwal et al. (2009); Frahm et al. (2010); Raguram et al. (2011); Snavely et al. (2008a)), and SCAN (Papadopoulos et al. (2010)).

Object clustering methods that yield an *overlapping* clustering include Query Expansion (Chum and Matas (2010); Chum et al. (2009)), Kernel Vector Quantization (Avrithis et al. (2010)), Probabilistic Latent Semantic Analysis (Simon and Seitz (2008)), Latent Dirichlet Allocation (Philbin et al. (2011)), and Medoid Shift (Weyand and Leibe (2011, 2013)).

For many of these approaches, the definition of a cluster is not well-aligned with the definition of an object. For example, Philbin and Zisserman (2008) use spectral clustering, which produces an *over-segmentation* that breaks down single objects into multiple clusters. Therefore, an additional merging step is required which again merges clusters corresponding to the same object into one. On the other hand, connected components based methods (Agarwal et al., 2009; Frahm et al., 2010; Raguram et al., 2011; Snavely et al., 2008a) tend to *under-segment* the image collection, causing multiple objects to be inside the same cluster. In our proposed algorithm, clusters have an intuitive definition: At the center of each cluster is an iconic image of a building or

object, and all images that overlap with the iconic image by at least a certain fixed amount belong to the cluster of this iconic image.

All of the algorithms discussed so far produce a *flat* clustering, i.e., an unordered set of clusters. There has been little work that produces a *hierarchical* image clustering. Epshtein et al. (2007) use Mean Shift (Comaniciu and Meer (2002)) in a top-down fashion to create a hierarchical structure of the different views of a scene. However, their approach only operates in 2D on a top-down view of the scene, which limits its granularity.

Several landmark clustering papers use geotags to limit the computational effort of building the matching graph and performing the clustering by pre-grouping images based on their location (Avrithis et al. (2010); Crandall et al. (2009); Gammeter et al. (2009, 2010); Kennedy and Naaman (2008); Quack et al. (2008); Zheng et al. (2009)). This is often necessary to make visual clustering feasible on large-scale image collections. For example, single-link hierarchical agglomerative clustering (Manning et al. (2008)) has quadratic complexity in the number of input images, so it can only be applied to smaller groups of geographically close images (Quack et al. (2008); Zheng et al. (2009)). However, this step limits the applicability of these approaches to images where geotags are available. Our approach is inherently parallelizable and scalable to very large image collections, making geographic pre-clustering unnecessary.

2.2.1 Discussion

In summary, most previous work on image clustering has produced a *hard*, *flat* clustering. We argue that instead, a *soft*, *hierarchical* clustering is more suitable to the task of object discovery in large-scale image collections. A soft clustering allows images to belong to multiple objects at once, which is necessary for the case of nearby landmarks. A hierarchical clustering can better represent objects with sub-structures. For example, if a building has interesting details on its facade or in its interior, then both the whole building and the details should form clusters, and the detail clusters should be child nodes of the building cluster.

2.3 Landmark Recognition

A clustering of images into landmark buildings can be used for recognizing landmarks in a given query image. There are three different approaches for this in the literature: Image Retrieval, Classification and Pose Estimation. In the following, we will discuss previous work for each of them and discuss their advantages and shortcomings.

2.3.1 Image Matching and Retrieval

To recognize the landmark in a query image, several approaches (Gammeter et al., 2009; Philbin et al., 2007; Quack et al., 2008; Sivic and Zisserman, 2003; Zheng et al., 2009) implement a *best match* strategy where the query is matched against a database consisting of the union of all landmark clusters. The landmark corresponding to the cluster containing the best matching image is returned. Quack et al. (2008) perform an expensive pairwise local feature matching against the database. To reduce the effort, they ask the user to draw a geographic bounding box on a map and only match against photos from this region. Zheng et al. (2009) use k-D trees to speed up local feature matching. Gammeter et al. (2009) retrieve images using inverted indexing and bags-of-visual-words (BoVWs, Nistér and Stewénius (2006); Philbin et al. (2007); Sivic and Zisserman (2003)). Li et al. (2008) only want to decide *whether* the query image contains a specific landmark. Given a dataset of photos of one landmark, they perform image retrieval using either k-NN search in GIST feature space or vocabulary tree matching (Nistér and Stewénius, 2006) using SIFT features. Then, they apply a threshold to the retrieval score to decide if the query contains the object. Both Avrithis et al. (2010) and Johns and Yang (2011) compress the images in a cluster into a joint BoVW representation and perform inverted file retrieval to find the best matching *scene models* for a query image.

2.3.2 Classification

An alternative approach is to view the task as a classification problem where each landmark is a class. Gronat et al. (2013) learn exemplar SVMs based on the BoVWs of the visual features of the database images. Li et al. (2009) learn a multi-class SVM and additionally use the BoWs (bags-of-words) of the textual tags of the images as features. Bergamo et al. (2013) use a similar approach, but perform classification using 1-vs-all SVMs. Instead of using approximate k-Means (Philbin et al., 2007) for feature quantization, they reconstruct the landmarks using structure-from-motion and train random forests on the descriptors of each structure-from-motion feature track. These random forests are then used for quantizing descriptors.

2.3.3 Pose Estimation

The goal of pose estimation is to determine the camera location and orientation for a given query image. This reduces the problem of landmark recognition to checking if the camera is pointed at a landmark. Several works solve this task by matching the query against street level imagery such as Google Street View panoramas using local feature based image retrieval (Baatz et al. (2010); Chen et al. (2011); Johns and Yang (2014); Knopp et al. (2010); Schindler et al. (2007); Torii et al. (2011)). Other approaches are based on 3D point clouds created by applying structure-from-motion on Internet photo

collections or manually collected photos (Li et al. (2010, 2012); Sattler et al. (2009, 2011)). Since image retrieval methods cannot be applied to 3D points, these approaches directly match the query descriptors against the descriptors of the image features that the 3D points were reconstructed from. After a set of 2D-3D correspondences has been established, the camera pose is determined by solving the perspective-n-point (PnP) problem (Hartley and Zisserman (2004)). Since the descriptor matching problem becomes computationally expensive when matching against very large 3D models, hybrid methods (Cao and Snavely (2013); Irschara et al. (2009); Sattler et al. (2012)) have been proposed that first perform efficient image retrieval using inverted files and then solve the PnP problem based on the relatively small set of 3D points associated with the 2D features of the retrieved images. Similarly, Hao et al. (2012) match SIFT features from the query image against structure-from-motion reconstructions of landmarks. However, instead of point-to-point correspondences their approach tries to find matching *3D visual phrases*, i.e., triplets of close-by 3D points with a known geometric configuration.

2.3.4 Discussion

We have discussed three approaches for landmark recognition: retrieval, classification and pose estimation.

An advantage of *classification* over both pose estimation and retrieval is that it requires fewer resources at runtime because it only needs the model parameters, which usually fit into RAM. In contrast, classification and pose estimation methods need the SIFT descriptors of the database. Since they are too large to be stored in RAM they typically need to be loaded from disk for every query, which slows down recognition. A major disadvantage of discriminative models is that they need to be re-trained every time new images and landmarks are added. Moreover, a classifier will assign *every* image a landmark label regardless of whether it contains a landmark or not.

When using *pose estimation* for landmark recognition, it is also possible to return their precise camera position and orientation, which can be useful in applications like augmented reality tour guides. However, pose estimation relies on either regularly sampled images (e.g. Google Street View panoramas), which are not available everywhere, or structure-from-motion reconstructions, which cannot always be computed robustly. Moreover, even if the camera is pointed at the landmark, it might still not be visible due to occlusion or bad weather and lighting conditions.

While *image retrieval* based landmark recognition has higher storage requirements than classification based methods, it has the desirable property that the index is easy to extend, because new images can simply be added to it incrementally. Bag-of-visual-words based retrieval methods have made it possible to scale image retrieval indexes to millions of images. This may be the reason why most landmark recognition approaches are based on image retrieval. We take a closer look at these approaches in our evaluation of landmark recognition in Ch. 6.

3 Foundations

In this chapter, we introduce the basic techniques underlying the approaches we present in the following chapters. A fundamental requirement for clustering photos is the ability to tell whether two images show the same object. We will refer to this task as *matching*. Our approaches make use of local feature based matching which we introduce in Section 3.1. In order to retrieve images matching a given query image, we use the popular visual words based image retrieval pipeline that we explain in Section 3.2. In Section 3.3, we introduce the Paris 500k dataset that we will use to evaluate our algorithms throughout this thesis. Finally, in Section 3.4, we perform a case study based on two existing landmark discovery algorithms, which motivates the approaches we will introduce in the following chapters.

3.1 Matching with Local Features

Even for humans, determining whether two images show the same object can be a very complex task. However, since our target application is clustering tourist photos by their depicted buildings, we can make certain simplifying assumptions. Firstly, we can assume that the images show the same specific object *instance*. This makes our task significantly easier than recognizing object *classes*, which are subject to a much wider variety of appearance changes. Secondly, we can assume *rigid*, *non-articulated* objects that do not change their shape. However, since we are dealing with tourist photos taken at different times and from different vantage points, we need matching techniques robust w.r.t. variations in lighting and perspective, as well as partial occlusions.

The popular *local feature* based matching pipeline (Grauman and Leibe, 2011; Lowe, 2004; Tuytelaars and Mikolajczyk, 2008) therefore seems best suited to our needs. Given a pair of images, the basic idea is to establish a set of *correspondences* between them. A correspondence is a pair of image regions that lie on the same part of the depicted object in both images . If an image pair has a sufficient number of spatially coherent correspondences, we consider it a match.

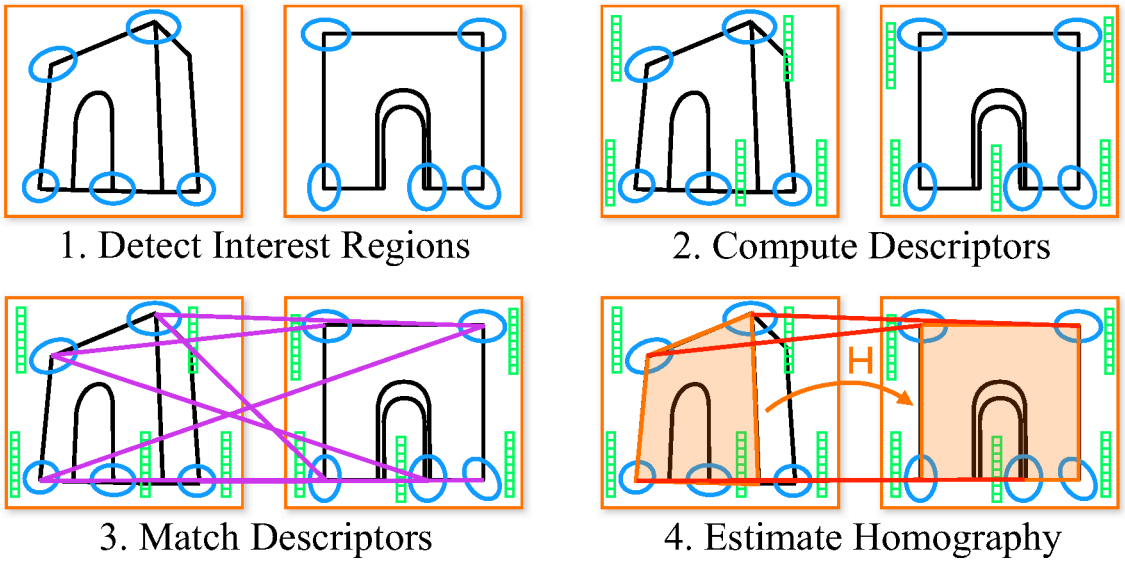


Figure 3.1: *Local feature based matching pipeline.*

The local feature based matching pipeline consists of the following steps (Fig. 3.1):

1. We detect a set of *interest regions* located at salient points in the images (Sec. 3.1.1). An important requirement for these regions is *repeatability*, i.e., they should be detected on the same part of the object in both images.

2. We compute a *descriptor* for each interest region (Sec. 3.1.2), i.e., a vector that captures the region's appearance.

3. By matching the interest region descriptors in both images, we establish a set of putative correspondences that have similar descriptors but are not necessarily spatially coherent.

4. Based on these correspondences, we estimate a geometric transformation, e.g., a homography, between the images and only keep those correspondences that are consistent with this transformation (Sec. 3.1.3). We will refer to these correspondences as *inliers*. If an image pair has enough inliers, we assume that the images show the same object and call the pair a *match*.

We will now look at each of these steps in more detail. In this discussion, we focus on the methods we use in our implementation and refer the reader to the literature for other methods and further details (Lowe, 2004; Mikolajczyk and Schmid, 2004, 2005; Szeliski, 2011; Tuytelaars and Mikolajczyk, 2008).

3.1.1 Interest Region Detection

The first step in the matching pipeline is to detect *repeatable* interest regions in the input images. Typical choices for this include the Harris detector that responds to corners (Harris and Stephens, 1988), the Hessian (Beaudet, 1978) and Difference-of-Gaussians (DoG) (Lowe, 2004) detectors that respond to blobs, and the MSER detector (Maximally stable extremal regions, Matas et al. (2002)) that returns the most stable regions in a watershed segmentation of the image. The Harris and Hessian detectors only return *interest points*, i.e., 2D image locations, while the DoG and MSER detectors return *interest regions*, i.e., connected groups of pixels that correspond to some object part. (However, as will discuss later, the Harris and Hessian detectors can be extended to also return interest regions.)

In order to be repeatable, an interest region detector must be *covariant* w.r.t. certain image transformations. An operator F is called covariant w.r.t. a transformation T on an image I if the operator and the transformation commute, i.e., $T(F(I)) = F(T(I))$, or $T \circ F = F \circ T$. For example, if an interest region detector is rotation-covariant, first rotating an image and then extracting interest regions yields the same result as first extracting interest regions and then applying the rotation to them. Additionally, a detector should be *invariant* w.r.t. changes in brightness and contrast, i.e., it should yield the same result regardless of such changes.

Most interest region detectors are fully translation and rotation covariant, and are invariant to lighting changes to a limited degree. Many detectors are also *scale-covariant* (e.g. DoG), and some (e.g. MSER) are even *affine-covariant*, i.e., covariant w.r.t. translation, rotation, anisotropic scaling and shearing. The latter two provide a certain level of covariance w.r.t. perspective distortions, which is desired when matching images taken from different camera angles. Mikolajczyk and Schmid (2004) combined the Harris and Hessian interest point detectors with a scale selection algorithm (Lindeberg, 1998) and an affine normalization algorithm (Lindeberg and Garding, 1997) such that they return scale-covariant and affine-covariant interest regions, respectively.

We refer the reader to Tuytelaars and Mikolajczyk (2008) for an excellent survey on interest region detectors. In the following, we will introduce the Hessian-Affine detector (Beaudet, 1978; Mikolajczyk and Schmid, 2004), which we used for our experiments[1].

Hessian Interest Points. The basic workflow of the Harris and Hessian detectors is to first compute a response function of the input image and then to identify the local maxima of this function using non-maximum suppression. The resulting image coordinates are returned as interest points. While the Harris detector responds to corners, the Hessian detector responds to blob-like structures. Given an input image I, the first step of computing the Hessian response function is to compute the second image derivatives.

[1]We used the implementation by K. Mikolajczyk available from http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html. In the meantime, M. Perdoch released an implementation that is faster, open source, and delivers higher performance in matching benchmarks. It is available from https://github.com/perdoch/hesaff.

The derivative images are then convolved with a Gaussian kernel of bandwidth σ such that the value of each pixel is a weighted average of a Gaussian window around it. From these blurred second derivatives I_{xx}, I_{xy} and I_{yy}, the Hessian matrix at the pixel position (x, y) is computed as

$$H(x, y, \sigma) = \begin{bmatrix} I_{xx}(x, y, \sigma) & I_{xy}(x, y, \sigma) \\ I_{xy}(x, y, \sigma) & I_{yy}(x, y, \sigma) \end{bmatrix}. \tag{3.1}$$

If we imagine the image intensity as a heightmap on the x/y plane, the Hessian matrix describes the curvature of the resulting surface at a given point. The Hessian response function $R_H(x, y)$ is the determinant of the Hessian matrix multiplied by a scale normalization factor σ^2 that is required for comparability across scales

$$R_H(x, y) = \sigma^2 det(H(x, y, \sigma)) = \sigma^2 \left(I_{xx}(x, y, \sigma) I_{yy}(x, y, \sigma) - I_{xy}^2(x, y, \sigma) \right). \tag{3.2}$$

Laplacian Scale Selection. In order to compute a characteristic scale for each detected interest point, Mikolajczyk and Schmid (2004) propose the following procedure. First, the *image pyramid* of the input image is computed by repeatedly downscaling the image. The Hessian detector is then run on each pyramid level to yield a set of interest points for each scale level. Since the pyramid scales are discretized, the scales of these interest points are only approximate. To find the exact scales, the scale selection algorithm by Lindeberg (1998) is used. Starting with an interest point position and an initial scale, the algorithm computes a scale signature function over a range of scales and selects the scale s at which this signature function reaches its extremum. A typical choice of scale signature function is the Laplacian-of-Gaussian, which, like the Hessian determinant, has a high response for blob-like structures. The resulting interest regions are referred to as *Hessian-Laplace* interest regions.

Affine Region Estimation. Starting at the circular regions returned by the Laplacian scale selection, Mikolajczyk and Schmid (2004) apply the affine normalization algorithm by Lindeberg and Garding (1997) to obtain affine-covariant interest regions. The affine shape of the interest region is estimated using the *second moment matrix*, often also called the auto-correlation matrix. In contrast to the Hessian matrix that consists of blurred *second* image derivatives, the second moment matrix consists of *first* image derivatives blurred with a Gaussian kernel of scale σ_D. The response is again blurred by a Gaussian kernel with scale σ_I and finally weighted by a scale normalization factor σ_D^2

$$M(x, y, \sigma_D, \sigma_I) = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(x, y, \sigma_D) & I_x(x, y, \sigma_D) I_y(x, y, \sigma_D) \\ I_x(x, y, \sigma_D) I_y(x, y, \sigma_D) & I_y^2(x, y, \sigma_D) \end{bmatrix}. \tag{3.3}$$

Here, $g(\sigma_I)$ is a Gaussian kernel with scale σ_I. The eigenvectors and eigenvalues of this matrix have a geometric interpretation. The eigenvectors represent the directions of the largest and smallest change in intensity, respectively, and the eigenvalues represent the amount of change. This matrix can be visualized as an ellipse whose *semi-major* and

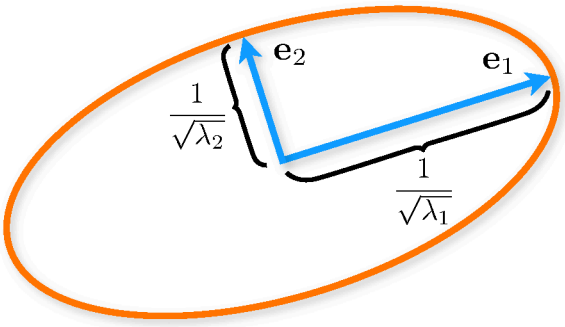


Figure 3.2: *Ellipse representation of the second moment matrix and eigenvectors* $\mathbf{e}_1$ *and* $\mathbf{e}_2$ *and eigenvalues* λ_1 *and* λ_2.

semi-minor axes (i.e. its directions of longest and shortest diameter, respectively) are given by the eigenvectors of the second moment matrix (Fig. 3.2).

The affine normalization algorithm alternates between the following steps:

1. Estimate the affine shape of the image region using the second moment matrix.
2. Warp the region such that its affine shape becomes a circle.
3. Correct the location and scale of the interest region by again performing Hessian interest point detection and Laplacian scale selection in the warped region.

After the eigenvalues of the second moment matrix become equal, the algorithm has converged and terminates. (See Lindeberg and Garding (1997) for an analysis of its convergence conditions.) The resulting *Hessian-Affine* interest regions are ellipses defined by five parameters: The (x, y) position of the interest point and the three parameters of the (symmetric) 2x2 affine matrix that define the shape of the ellipse.

3.1.2 Interest Region Description and Matching

An interest region descriptor is a fixed-size vector that captures the appearance of an image region. A descriptor should be *distinctive*, so that interest regions in two images can be matched by comparing their descriptors. Moreover, like an interest region detector, a descriptor should be *invariant* to brightness and contrast changes.

Before the descriptor is computed, the image patch of its interest region is extracted and normalized to a standard shape (usually a square) by applying the inverse rotation and affine transformation of the interest region. Because of this normalization, the descriptor does not need to have a high level of invariance w.r.t. image transformations. In fact, too much invariance would negatively affect its distinctiveness (Tuytelaars and

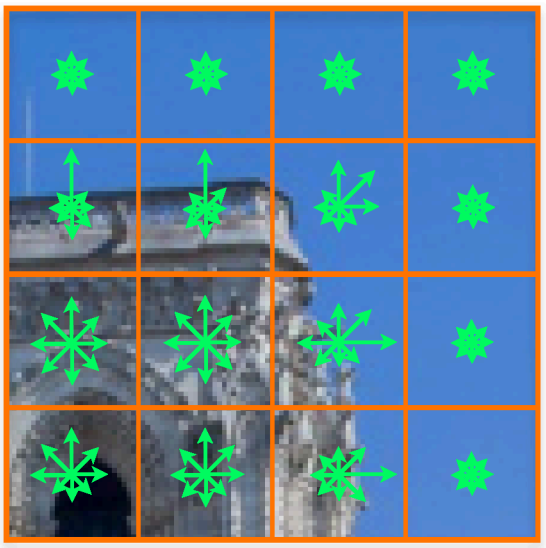


Figure 3.3: *The SIFT descriptor consists of 4x4 histograms of oriented gradients that have 8 orientation bins each.*

Mikolajczyk, 2008, Sec. 1.4.1). However, some invariance to these factors is helpful to make up for slight errors in the interest region location and shape (Lowe, 2004, Sec. 6).

There is a wide variety of interest region descriptors (Mikolajczyk and Schmid (2005) provide a survey). The Scale-Invariant Feature Transform descriptor (SIFT, Lowe (2004)) is arguably the most popular descriptor to date[2], and several variations of it have been proposed, e.g., SURF (Bay et al., 2008), GLOH (Mikolajczyk and Schmid, 2005) or DAISY (Tola et al., 2010).

We now briefly describe the SIFT descriptor (Fig. 3.3) which we use in our experiments[3]. First, a 4x4 grid is overlaid over the input image patch. Inside each grid cell, gradient directions and magnitudes are computed at 4x4 sampling positions and accumulated into a gradient histogram with 8 directional bins. Gradients are weighted by a Gaussian window such that gradients farther from the patch center have less influence on the descriptor. Each gradient sample contributes its gradient magnitude to the 4 grid cells closest to its own position, where contributions are weighted by the distance of the sample position to the center of each grid cell. In each of these four gradient histograms, the sample contributes to the two orientation bins closest to its own orientation. In total, each gradient sample contributes to $2x2x2 = 8$ histogram bins (2 spatial dimensions and one orientation dimension) and contributions are weighted using tri-linear interpolation. Therefore, a sample contributes its value to the same histogram bin even if the descriptor

[2]According to Google Scholar, SIFT (Lowe, 2004) has been cited 25,591 times as of August 2014.

[3]We used the implementation by K. Mikolajczyk available from http://www.robots.ox.ac.uk/~vgg/research/affine/descriptors.html.

is shifted by up to two grid cells or rotated by up to 90 degrees. The orientation histograms of all cells are concatenated, yielding a 128-dimensional vector. This descriptor is invariant w.r.t. brightness changes since it is based on image gradients. To improve invariance w.r.t. contrast, the descriptor is normalized to unit length w.r.t. the L^2 norm. Additionally, to improve invariance to non-linear brightness changes, the dimensions of the feature vector are clipped at a value of 0.2 and the resulting vector is re-normalized. This step reduces the influence of very strong gradients on descriptor distances. Finally, each dimension is multiplied by 256 and represented by an unsigned byte. The size of a SIFT descriptor is therefore 128 bytes.

Given two images and their SIFT descriptors extracted from Hessian-Affine interest regions, we now want to find correspondences between their interest regions. For this, each SIFT descriptor in the first image is assigned its nearest neighbor in the second image. Distances between SIFT descriptors can be efficiently computed by their scalar product, which is a linear function of their squared Euclidean distance, since SIFT descriptors have unit length[4]. To prevent false-positive correspondences due to repetitive structures in the images, Lowe (2004) propose to discard correspondences that do not pass the *second nearest neighbor* test. The idea of this test is to find both the nearest and second nearest neighbor for an interest point descriptor and to compute the ratio of their distances d_1/d_2. If this ratio exceeds a threshold (usually 0.8), the distances are too similar, meaning that we cannot confidently decide whether the first or second nearest neighbor would be the correct match. Therefore, we discard correspondences whose distance ratio is above a certain threshold. Using the remaining correspondences, we can now compute a spatial transformation between the input images. If this step succeeds and a sufficient number of correspondences are consistent with the resulting transformation, we assume that the same object is present in both images.

3.1.3 Transformation Estimation

We now want to estimate a transformation that maps the coordinates of each pixel in the first image to its corresponding location in the second image. The choice of transformation depends on the object geometry and how the camera moves relative to the object. We are mainly interested in matching photos of buildings taken from different perspectives. Assuming that we can approximate buildings as mostly flat surfaces, we choose *homographies* as our transformation model. A homography is a mapping between two perspective projections of a planar object that preserves straight lines, but not parallel lines. It is represented by a 3×3 matrix with 8 degrees of freedom: scaling, translation, shearing and perspective foreshortening along the x and y axes, respectively.

[4] $\|x - y\|^2 = \|x\|^2 - 2x^\top y + \|y\|^2 = 2 - 2x^\top y$

To apply a homography H to a pixel position $(x, y, 1)^\top$ given in homogeneous coordinates, the position is multiplied by the homography and subsequently dehomogenized, which yields the projected position $(x_p, y_p, 1)^\top$.

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \mathrm{H} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad , \quad \begin{pmatrix} x_p \\ y_p \\ 1 \end{pmatrix} = \begin{pmatrix} x'/z' \\ y'/z' \\ 1 \end{pmatrix} \tag{3.4}$$

Since a homography has 8 degrees of freedom, it can be estimated from 4 or more correspondences of 2D points. Inserting all correspondences into the above equation yields a linear system that can be solved in the least squares sense using singular value decomposition (SVD). This method is called the Direct Linear Transform (DLT) algorithm. We refer the reader to Hartley and Zisserman (2004) for a detailed discussion of different methods for estimating homographies and other image transformations.

Estimating the homography between a pair of images using the putative correspondences between their interest regions is a chicken-and-egg problem. The correct homography can only be estimated from a set of correct correspondences. However, we do not know which correspondences are correct unless we already know the correct homography. Simply using all correspondences and solving the resulting over-constrained least squares problem is not likely to work, since we can expect many correspondences to be wrong, especially if the object of interest only takes up a small fraction of the images.

A solution to this problem is the Random Sample Consensus (RANSAC, Fischler and Bolles (1981)) algorithm. RANSAC is a randomized algorithm for estimating a transformation in the presence of a large number of *outliers*, i.e., correspondences that are not consistent with the transformation. In each iteration, RANSAC draws a minimal random sample of correspondences (4 in the case of a homography), estimates the transformation from them and then checks how many correspondences are inliers. If the sample contains outliers, the transformation will be wrong and thus have a small number of inliers. A homography estimated from only correct correspondences, however, is expected to have a large number of inliers. After a certain number of iterations, the transformation with the most inliers is therefore chosen as the correct one. The final transformation is re-estimated from all the inliers of the best homography using least squares.

To determine whether a correspondence is an inlier or an outlier w.r.t. a homography, we check how close the projection of the interest point in the first image is to its corresponding interest point in the second image. This can be done, for example, using the *symmetric transfer error* which measures the error of projection in both directions. Let $\mathrm{t}(\mathrm{H}, \mathbf{x})$ be a function that projects a point $\mathbf{x}$ using a homography H as in Equation (3.4). Given a correspondence, i.e., a pair of interest point positions $\mathbf{x_1}$ in image 1 and $\mathbf{x_2}$ in image 2 and a homography H that projects from image 1 to image 2, the symmetric transfer error is

$$e(\mathbf{x_1}, \mathbf{x_2}, \mathrm{H}) = \|\mathrm{t}(\mathrm{H}, \mathbf{x_1}) - \mathbf{x_2}\|^2 + \|\mathrm{t}(\mathrm{H}^{-1}, \mathbf{x_2}) - \mathbf{x_1}\|^2. \tag{3.5}$$

Correspondences whose symmetric transfer error is below a certain threshold are considered inliers. An image pair that has a sufficiently large number of inliers is considered a *match*.

Many RANSAC variants have been proposed in the literature (see Raguram et al. (2008) for a comparison). We use Spatially Consistent Random Sample Consensus (SCRAMSAC, Sattler et al. (2009)), which runs a spatial consistency pre-filter similar to the one used in Sivic and Zisserman (2009) before applying RANSAC. In informal experiments we found that this provides a speedup over regular RANSAC without having a negative effect on the results.

We now have all the components required to determine whether two images show the same object. However, if we want to know which images from a large dataset match a given query image, we would need to perform this expensive matching procedure for every image. In the following, we will therefore introduce an efficient image retrieval pipeline based on local features that makes it possible to find the set of images matching a query in very large datasets in sub-linear time w.r.t. the number of images in the dataset.

3.2 Image Retrieval using Inverted Files

We now introduce an image retrieval framework originally proposed by Sivic and Zisserman (2003) that makes efficient text retrieval techniques applicable to image retrieval. First, we explain the original text retrieval methods and then discuss how the same methods can be applied to images by converting local features into *visual words*.

3.2.1 The Vector Space Model for Text Retrieval

Given a very large corpus D of text documents, e.g., the world wide web, the task of a search engine is to quickly retrieve documents that match a given query string. A simple (Boolean) approach for this would be to sort the documents by the number of words they have in common with the query. To do this, we define a finite, fixed-size *vocabulary* V, i.e., a sorted list of all possible words, and represent a document as a *bag of words*, i.e., a histogram of word frequencies. In the Boolean vector space model, this histogram is binary, i.e., entry w in the bag of words is 1 if word w occurs in the document and 0 otherwise. We create such a bag of words histogram for each document in our corpus and stack them to form a *term-document matrix* $\mathrm{M} \in \{0,1\}^{|D|\times|V|}$. Entry $\mathrm{M}(d,w)$ in this matrix is 1 if word w occurs in document d. Given a query document, we compute its bag of words vector and multiply it with the term-document matrix. The result is a vector whose dth element is the dot product of the query and document d, which is the number of common words that document has with the query. Thus, we can now sort the documents by their number of common words with the query and return the resulting ranking.

Since the term-document matrix is very sparse and would we wasteful to store in memory, a more compact but equivalent representation is used that is called the *inverted index*, or *inverted file*. For each word, the inverted index simply lists the documents that this word appears in. Each such list is called a *posting list*. To compute the number of common words of the query with each document, the posting list of each word occurring in the query is traversed. For each document in the posting list, its number of common words with the query is incremented by one. This can also be viewed as a *voting* process, where each word from the query casts votes for the documents it appears in. The inverted index not only saves memory, but also makes retrieval much more efficient. Instead of all words in the dictionary, we now only need to consider the words that actually occur in the query. Moreover, the computation time no longer depends on the number of documents in the corpus, but only on the number of documents that have words in common with the query. Retrieval therefore has *sub-linear* computational complexity in the number of documents.

This Boolean retrieval model, while instructional, has at least three shortcomings: Firstly, common words like "the" will have the same importance as more informative words like "rhinoceros". Secondly, words receive equal weight regardless of how often they occur in a document. For example, we should expect that if a document contains the word "cat" 50 times and the word "dog" one time, it is probably about cats and thus much more relevant to the query "cat" than to the query "dog". Thirdly, different document lengths are not accounted for. Longer documents that contain more words will have a higher likelihood of accidentally having a word in common with the query than shorter documents. Thus, long documents have an unfair advantage in retrieval.

These problems are addressed by the tf-idf weighting scheme. In this scheme, the weight of a term is increased if it occurs frequently in a document, i.e., if it has a high *term frequency* (tf). Likewise, the weight of a term is decreased if it occurs in many documents in the corpus, i.e., if it has a high *inverse document frequency* (idf). Let df_w denote the number of documents that the term w occurs in. Then, a term w in a document d will get the weight $\mathrm{tf}_{wd} \cdot \mathrm{idf}_w$, where tf_{wd} is the frequency of term w in document d and

$$\mathrm{idf}_w = \log\left(\frac{|D|}{\mathrm{df}_w}\right). \tag{3.6}$$

The idf of a term is therefore lower if it occurs more frequently in the corpus and becomes 0 if the term occurs in every document.

To incorporate tf-idf weighting into the bag of words model, each term w in the bag of words histogram of a document d is weighted by $\mathrm{tf}_{wd} \cdot \mathrm{idf}_w$. Given two tf-idf vectors $\mathbf{v_a}$ and $\mathbf{v_b}$, their similarity is computed as

$$\mathrm{sim}(\mathbf{v_a}, \mathbf{v_b}) = \frac{\mathbf{v_a}^\top \mathbf{v_b}}{\|\mathbf{v_a}\|\|\mathbf{v_b}\|} = \frac{\sum_w \mathrm{tf}_{wa} \cdot \mathrm{tf}_{wb} \cdot \mathrm{idf}_w^2}{\sqrt{\sum_w \mathrm{tf}_{wa}^2} \cdot \sqrt{\sum_w \mathrm{tf}_{wb}^2}}. \tag{3.7}$$

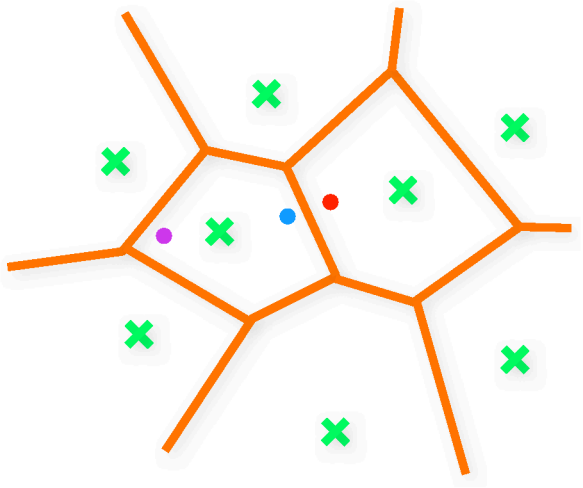


Figure 3.4: *Visual words are the Voronoi cells formed by the cluster centers making up the visual vocabulary (green). The colored dots illustrate two types of quantization errors: The blue and red dots are very close in descriptor space, but are assigned different visual words, which can lead to* false-negative matches. *The purple and blue dots are far apart in descriptor space, but are still assigned to the same visual word which can lead to* false-positive matches.

The normalization by the norms of the tf-idf vectors accounts for different document lengths. This similarity is called the *cosine similarity*, because the dot product of two unit vectors is the cosine of the angle between them. The tf-idf weighting scheme can efficiently be incorporated into inverted file retrieval by storing the tfs in each posting list entry. The idf of each vocabulary term and the norm of the tf-idf vector of each document can be pre-computed for efficiency. The cosine similarities of each document with the query can then be accumulated efficiently during retrieval.

We refer the reader to Manning et al. (2008) for a more detailed introduction to text retrieval methods.

3.2.2 Image Retrieval with Bags of Visual Words

The basic idea that enables the application of text retrieval techniques to images is to regard the local features of an image as *visual words*. However, since each SIFT descriptor is a 128 byte vector, the dictionary would contain 256^{128} words and thus it would be highly unlikely for a pair of images to even have one visual word in common. Therefore, it would be desirable to quantize the descriptor space into a more manageable number of visual words (which is usually chosen to be between 10^3 and 10^7).

Descriptor Quantization.

Because SIFT features are highly non-uniformly distributed, a regular lattice over feature space (Tuytelaars and Schmid, 2007) would have mostly empty bins and thus the vast majority of words in the dictionary would never occur. Instead, Sivic and Zisserman (2003) quantize feature space adaptively to the data using *vector quantization*. For this, a representative sample of the SIFT descriptors of all images is clustered using k-means (Duda et al., 2000). The resulting cluster centers are referred to as *visual words* and the set of all cluster centers is called the *visual vocabulary*. A SIFT descriptor is now represented by the index of its closest visual word in the visual vocabulary. This way, a 128-dimensional descriptor becomes an integer between 1 and $|V|$, where $|V|$ is the size of the vocabulary. In the following, we will also refer to this index as *visual word*. Two SIFT descriptors are assigned the same visual word if they are closest to the same cluster center. The set of all SIFT descriptors with the same visual word forms a Voronoi cell in the 128-dimensional descriptor space (Fig. 3.4). Intuitively, a visual word should capture the appearance of a certain part of an object and its Voronoi cell should span its variability in appearance. This quantization makes descriptor matching much more efficient, because we only have to compare two integers instead of computing the dot product of two 128-dimensional vectors.

Quantization Errors. A drawback of this matching procedure is that it is much less precise than exact descriptor comparison due to quantization errors. For example, two descriptors that are very close in feature space might still end up in different Voronoi cells and thus not form a match. Likewise, two descriptors in the same Voronoi cell are not necessarily close in feature space, as illustrated in Figure 3.4. Depending on the size of the vocabulary, there will be more *false-positive* or *false-negative* feature matches. In a too coarse vocabulary, the Voronoi cells can be so large that SIFT descriptors of different object parts will match. In a too fine vocabulary, slightly different SIFT descriptors of the same object part might end up in different cells. To allow for a more fine-grained matching without sacrificing the benefits of visual word matching, there are approaches to softly assign a SIFT feature to multiple visual words (Philbin et al., 2008) or to introduce finer levels of quantization that allow computing approximate descriptor distances of descriptors assigned to visual words (Jégou et al., 2008, 2011).

Bags of Visual Words. An image is now converted to a *bag of visual words* by assigning each of its SIFT descriptors a visual word and building a $|V|$-dimensional histogram of their frequencies. The resulting representation is analogous to the bag of words representation from text retrieval and now enables us to perform image retrieval using inverted files. tf-idf weighting is applicable here as well and has an intuitive interpretation. We can consider visual words as parts of objects, like the wheel of a car or a corner of a window. If an image contains a certain part more frequently, this part is likely more important and should thus receive a higher weight. Likewise, if a part appears in many images, it likely carries less information than a rarely occurring part and should receive a lower weight.

Approximate Nearest Neighbor Search. Assigning a SIFT feature to a visual word requires finding its closest cluster center in the visual vocabulary. In a naive implementation, this would require computing $|V|$ scalar products, which would be very computationally expensive for typical visual vocabulary sizes (we use 1M visual words in our experiments). Therefore, we would like to speed up the search using efficient data structures. Unfortunately, exact nearest neighbor search techniques like k-d trees (Bentley, 1975) would not bring a speed gain over linear search in 128 dimensions due to the curse of dimensionality. However, we can use approximate nearest neighbor search methods (Arya and Mount, 1992; Silpa-Anan and Hartley, 2008) that are more efficient than linear search, but are not guaranteed to find the correct nearest neighbor every time. We use an approach based on a forest of randomized k-d trees as proposed by Philbin et al. (2007). The basic idea is to build a set of k-d tree search indices for the visual vocabulary, where the construction of each k-d tree is randomized.

A k-d tree is a binary space partitioning tree. Each node of the tree is a hyperplane that splits its space into two half-spaces. A k-d tree is constructed from a set of data points (cluster centers in our case) in a top-down fashion. At each node, the dimension with the highest variance is chosen as the splitting dimension, i.e., the dimension perpendicular to the splitting hyperplane. The position of the splitting plane along this dimension is chosen to be the median of the projections of all data points onto this dimension. Therefore, each node has an associated data point that lies inside its hyperplane.

Given a query data point, we search its nearest neighbor by descending the tree. At each node, we choose the half-space containing the query point. During the descent, we keep track of the closest data point to the query we have encountered so far. We also store all nodes along our path whose splitting planes are closer to the query than the current closest point in a priority queue. This is necessary, because the nearest neighbor might be on the other side of one of these hyperplanes. Each time we encounter a point closer to the query point than the currently closest neighbor, we prune the priority queue by removing all points farther from the query than the new closest point. After arriving at a leaf, we process the nodes in the priority queue by descending the tree from each of them, updating the current closest neighbor and priority queue along the way.

In a *randomized* k-d tree, the splitting dimension is chosen randomly with a probability proportional to the variance across each dimension and the splitting position is chosen as a random data point close to the median. When searching the nearest neighbor of a query SIFT descriptor using a forest of k-d trees, we descend each tree as described above and insert the alternate nodes to visit in a *global* priority queue. Each entry in this priority queue stores the id of a tree and the id of a node within this tree. We also keep track of the global nearest neighbor node over all trees. After having descended each tree and built up the priority queue, we process only a *fixed number* of nodes from the priority queue, as described above.

Because we do not process all the nodes in the priority queue, we might not find the exact nearest neighbor to the query. We can increase the probability of finding the correct nearest neighbor by either increasing the number of backtracking steps, which

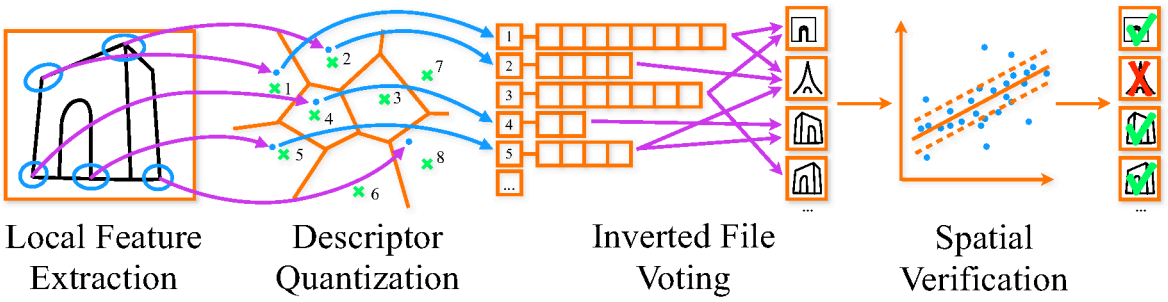


Figure 3.5: *Visual Word based image retrieval pipeline.*

affects computation time, or the number of trees, which mainly affects memory use. These two parameters make it possible to find a tradeoff between accuracy and speed that fits the given computational resources. We use approximate nearest neighbor search for both k-means clustering and the computation of the bags of visual words.

Visual Words Based Image Retrieval Pipeline. We now have all the components required to implement an efficient image retrieval engine. To build the retrieval index for an image corpus, we perform the following steps.

1. Detect Hessian-Affine interest regions and extract SIFT descriptors from all images.
2. Draw a uniform sample of the SIFT descriptors from the database and cluster them using approximate k-means to create a visual vocabulary.
3. Assign the SIFT descriptors of all images to visual words using approximate nearest neighbor search to generate a bag of visual words for each image.
4. Build an inverted file index from the bags of visual words and pre-compute the idf of each visual word as well as the norm of the tf-idf vector of each image.

Given a new query image, we then retrieve matching images using the following procedure (Fig. 3.5):

1. Detect Hessian-Affine interest regions and extract SIFT features from the query.
2. Assign the SIFT descriptors of the query to visual words using approximate nearest neighbor search to generate a bag of visual words for it.
3. Query the inverted index with the bag of visual words and rank the results by the cosine similarity of their tf-idf vectors to the tf-idf vector of the query.
4. For the top-k ranked images, perform spatial verification by matching SIFT features to establish correspondences and estimating a homography using RANSAC. If an image has more than a certain number of homography inliers with the query, it is considered a *match*.

3.2.3 Matching Graphs

We would now like to cluster images using the local feature based matching techniques introduced above. A popular approach for this is to build a *matching graph* (e.g., Avrithis et al. (2010); Chum and Matas (2010); Quack et al. (2008); Raguram et al. (2011); Zheng et al. (2009)). In this graph, each image is a node, and two images are connected by an edge if they match. Sometimes, edges are also weighted, e.g., by the number of inliers. Given an image collection, a matching graph is built by creating an inverted index and querying it with each image. The query image is then connected to all matching images. The resulting graph now allows us to analyze the connectivity structure of the input image set and makes it possible to apply existing graph clustering algorithms like connected components analysis, hierarchical agglomerative clustering (Duda et al., 2000) or spectral clustering (Ng et al., 2001).

3.3 The Paris 500k Dataset

We now introduce our Paris 500k dataset that we use for many experiments throughout this thesis. Paris 500k is a realistic large-scale dataset for evaluating landmark discovery algorithms. Besides that, it allows for analyzing what objects can actually be discovered in internet photo collections (see Ch. 6). Landmark discovery algorithms aim to find often-photographed objects in large amounts of tourist photos from Internet image collections like Flickr, Picasa, or Panoramio. However, most datasets that have been used to evaluate these algorithms so far have not been realistic (we discuss different landmark datasets in Section 6.2.1). For example, the Oxford 105k dataset (Philbin et al., 2007) consists of 5k images of different landmark buildings in Oxford and 100k random distractor images. The Oxford part of the dataset was created by retrieving images of the Oxford landmarks using keyword searches on Flickr. Despite the large number of distractor images, the task of discovering landmarks in the dataset is very easy, because it contains only 11 landmarks, all of them building-scale, meaning there is a large number of photos of each of them. In a realistic setting, there are hundreds to thousands of popular objects in a city, some of which as obscure as a painting in a museum or a graffiti in the street. To evaluate object discovery in a challenging real-world setting, we created the Paris 500k dataset.

The dataset consists of 500k images of Paris from Flickr[5] and Panoramio[6]. To avoid a bias towards certain landmarks, we do not use keyword searches, but instead query for a geographic region covering the inner city of Paris. Figure 3.6 shows the distribution of the collected photos. The dataset is publicly available on the web[7]. We created two

[5] http://www.flickr.com
[6] http://www.panoramio.com
[7] http://www.vision.rwth-aachen.de/data/paris500k

Figure 3.6: *Geographical distribution of the photos in the* Paris 500k *dataset.*

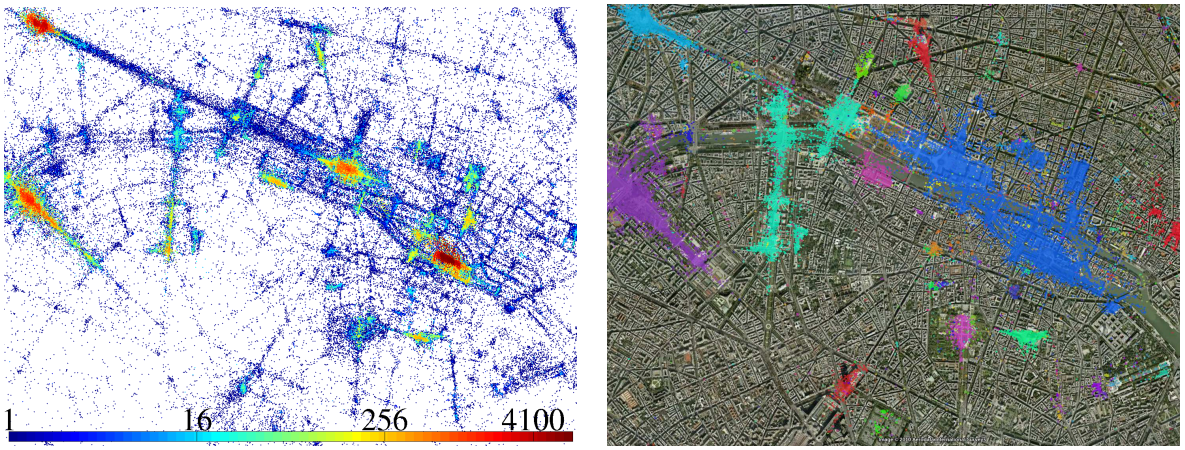
ground truths for it, one for landmark clustering that we will introduce in the following section and one for landmark recognition that we introduce in Chapter 6.

3.3.1 Dataset Statistics

We compute the matching graph of the dataset using the inverted file image retrieval pipeline introduced above. To limit the memory requirements, we apply geo-spatial binning to only mach geographically close images, similar to Quack et al. (2008). For this, we use four overlapping grids with 200x200m cells. The grids are shifted by 100m in latitudinal and longitudinal direction. Each image is inserted into four overlapping cells and the matching graph of each cell is constructed using image retrieval. The resulting matching graphs are merged into one global matching graph. While this reduces computational requirements for matching and prevents false-positive matches to some degree, we found that geo-spatial binning also isolates images with inaccurate geotags, since they will end up in the wrong cell and not be matched against other images showing

	Paris 500k	Oxford 5k
# Features	1,564,381,034	16,334,970
# Nodes	501,356	5,052
# Edges	11,356,090	11,957
avg. valence	45.3	4.7
max valence	4,100	83

Table 3.1: *Statistics of* Paris 500k *and* Oxford 5k *and their matching graphs.*



(a) (b)

Figure 3.7: *(a) Density of the matching graph. Color encodes node valence (log scale). (b) Connected components of the matching graph containing at least 20 images.*

the same object. We consider an image pair a match if it has 15 or more inliers w.r.t. its estimated homography. The matching graph is available online along with the dataset.

Statistics of the matching graph of Paris 500k are shown in Table 3.1 and compared to the matching graph of Oxford 5k, which is the landmarks part of the Oxford 105k dataset. The average valence of the Paris dataset is an order of magnitude higher than the average valence of the Oxford dataset due to the extreme density of tourist photos at the most popular places. The photo with the highest valence (4,100) is a frontal shot of the facade of Notre Dame. Figure 3.7a visualizes the density of the matching graph of Paris and Figure 3.7b shows its connected components. The largest connected component (blue, 58,652 images) spans an area ranging from Notre Dame to the Louvre. This shows that connected components can give a good initial grouping (Philbin and Zisserman, 2008), but further segmentation is required for a building-level clustering. The largest connected component on the Oxford dataset is All Souls College (406 images). Table 3.2 gives statistics of the connected component sizes. For both datasets, more than half of the images did not have a single match. This shows the

	CCs	images
total	303,522	501,356
$= 1$	277,490	277,490
≥ 2	26,032	223,866
≥ 20	397	150,367
≥ 100	63	138,122
≥ 500	19	129,961

(a) Paris 500k

	CCs	images
total	3,297	5,052
$= 1$	2,917	2,917
≥ 2	380	2,135
≥ 20	11	929
≥ 100	2	518
≥ 500	0	0

(b) Oxford 5k

Table 3.2: *Statistics of the connected components. The middle column gives the number of connected components with a particular size, and the right column gives the total number of images in these components. Connected components of size 1 are images for which no match was found.*

large amount of "junk" photos present on Internet photo collections. As also observed by Gammeter et al. (2010), the connected component sizes are power-law distributed.

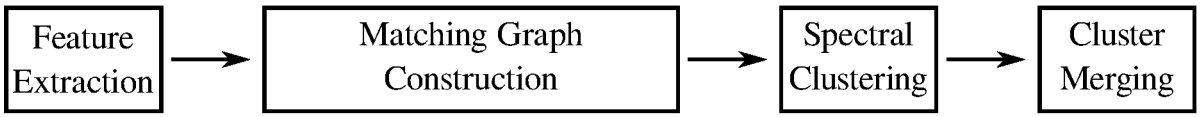
3.4 Comparison of two Landmark Discovery Algorithms

For an initial experiment, we compare two landmark discovery algorithms on the Paris 500k dataset. The first is the method by Philbin and Zisserman (2008) that builds the matching graph of the whole dataset and applies spectral clustering (Ng et al., 2001). The second method by Chum et al. (2009) finds a set of *seed* images using Geometric min-hash, a hashing scheme for bags of visual words, and uses them as starting points to grow clusters by query expansion. Figure 3.8 shows a schematic overview of both pipelines. We analyze the strengths and weaknesses of both approaches which motivates the methods we present in the following chapters.

This work was performed in collaboration with my student Jan Hosang and was published at the ECCV'10 RMLE workshop Weyand et al. (2010). Jan contributed with several ideas and by implementing the spectral clustering approach as well as tools for ground truth annotation, visualization and computation of statistics. Moreover, he performed several of the experiments described in this section.

3.4.1 Spectral Clustering

Philbin and Zisserman (2008) first compute the matching graph of the entire dataset as we described in Section 3.2.3. Because they observed that the connected components of the matching graph tend to contain several landmarks (cf. Fig. 3.7b), they then break each connected component down into smaller clusters using spectral clustering (Ng et al., 2001). Since spectral clustering requires the user to specify the desired number of clusters



(a) *Spectral clustering (Philbin and Zisserman, 2008).*

(b) *Min-hash (Chum et al., 2009).*

Figure 3.8: *The two different landmark discovery pipelines we compare. The dashed boxes denote our spectral clustering add-on for improving cluster purity.*

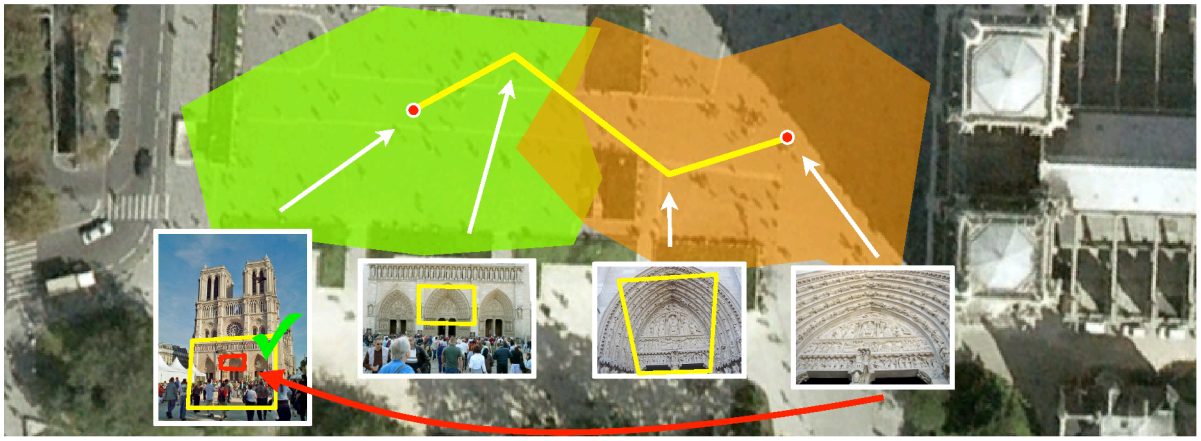


Figure 3.9: *The approach for merging spectral clusters of the same landmark proposed by Philbin and Zisserman (2008). The green and orange clusters correspond to close-up and full views of the building facade, respectively. The method determines whether the clusters show the same scene by projecting the image boundaries of their iconic images (red dots) along the shortest path in the matching graph.*

in advance, Philbin and Zisserman (2008) find the optimal number of clusters for each connected component by choosing the number of clusters that maximizes the Newman Q measure (Newman and Girvan, 2004). They report that the resulting clusters have high purity, but that they over-segment each landmark into several smaller clusters. To again join clusters of the same landmark, they propose the following procedure (Fig. 3.9):

1. For each cluster, choose the image with the highest valence in the matching graph as its *canonical image*.
2. For each pair of clusters, find the shortest path between their canonical images in the matching graph.

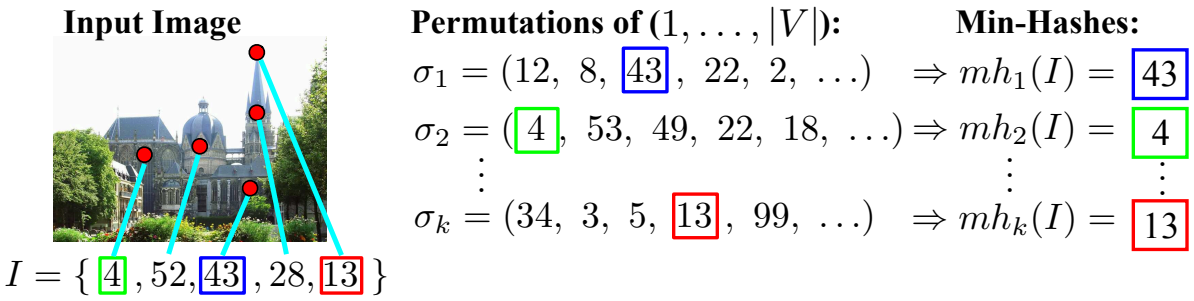


Figure 3.10: *An example min-hash computation. The input image and its bag of visual words are shown on the left. Given a random permutation (middle), a min-hash is the first visual word in the image that occurs in the permutation.*

3. Estimate the overlap of the canonical images by projecting the image boundaries of one of them into the other using the known homographies between the image pairs along the shortest path.

4. If the canonical images overlap by more than a certain amount, merge their clusters.

3.4.2 Min-Hash

While the spectral clustering based method first builds the whole matching graph and then divides it into clusters, the min-hash (Chum and Matas, 2010) and Geometric min-hash (Chum et al., 2009) methods start at seed images found by min-hashing and grow clusters using query expansion (Chum et al., 2007b). This method performs only a local exploration of the matching graph and is therefore expected to be faster than the spectral clustering method that requires the construction of the full matching graph. An overview of the pipeline is given in Figure 3.8b. The stages in dashed boxes are an extension that we propose later in this section.

Min-hash and Geometric min-hash. Min-hash (Broder, 1997) is a technique from text retrieval used by (Chum and Matas, 2010; Chum et al., 2008, 2009) for efficiently discovering pairs of similar images in large image collections. The special property of min-hash is that the probability of an image pair being discovered increases with its similarity. This makes min-hash suitable as a near-duplicate image detector (Chum et al., 2007a). Chum and Matas (2010) use it to discover *seeds* for image clustering.

A min-hash is a pseudo-random number generated from the visual words of an image. Let V be a visual vocabulary. Given a random permutation of the numbers $\{1, \ldots, |V|\}$, the min-hash of an image is the first of the image's visual words occurring in the permutation. Figure 3.10 demonstrates min-hash on a toy example. The probability of two images having the same min-hash equals the Jaccard similarity coefficient (intersection

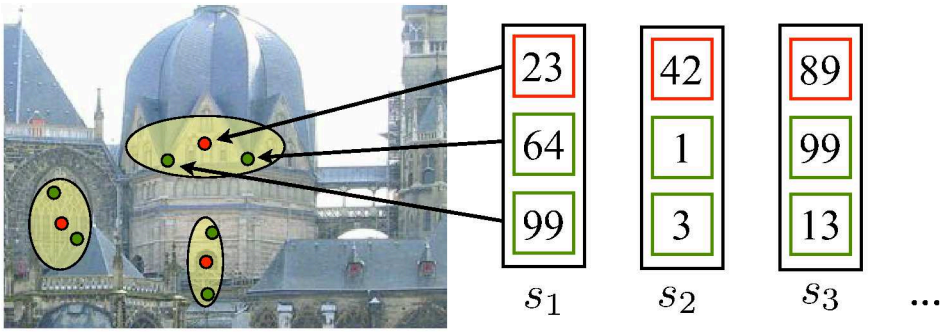


Figure 3.11: *Geometric min-hash example. The min-hashes in each sketch are restricted to come from the same affine neighborhood.*

over union) of their bags of visual words (Chum et al., 2007a). To decrease the number of random collisions, several min-hashes are summarized into s-tuples called *sketches* ($s = 3, \ldots, 5$). Each image is represented by k sketches ($k = 512$ in Chum and Matas (2010)). An image pair only *collides* if the images have at least one identical sketch.

To efficiently find min-hash collisions, hash-tables are created that store for each min-hash the list of images with this hash. This is comparable to an inverted index (Sec. 3.2), but sparser. This hashing procedure enables constant-time collision detection (Chum and Matas, 2010), however at much lower recall than with a full inverted index.

In Geometric min-hash (Chum et al., 2009), sketches are created from features in a spatial neighborhood. This is done by selecting the first min-hash in a sketch randomly and then restricting the search for the remaining min-hashes to the affine-covariant interest region around the first feature (Fig. 3.11). To ensure that the first min-hash has a unique position in the image, only the visual words that occur *once* in the bag of visual words are considered when computing the first min-hash. In Geometric min-hash, a sketch collision means that the colliding images not only have the colliding visual words in common, but also that the corresponding features come from the same image region. Because of this more distinctive definition of a sketch, Chum et al. (2009) report an increase in precision and recall over standard min-hash even with the sketch size reduced to $s = 2$ and the number of min-hashes per image reduced to $k = 60$. In our experiments, we use Geometric min-hash.

Cluster growing. Chum and Matas (2010) use the images with colliding min-hashes as cluster *seeds* and grow clusters by applying *query expansion* (Chum et al., 2007b). For each cluster seed image discovered by min-hash, they perform image retrieval (Sec. 3.2), average the bags of visual words of the spatially verified results with the bag of visual words of the query and use the result as a new query. This process is performed recursively for each of the retrieved images until no new images are found.

In our implementation of query expansion, we exploit the already computed matching graph of the dataset. We do not perform averaging of bags of words, but simply issue

recursive queries with all spatially verified results. By performing this process recursively, we effectively explore the connected component of the matching graph that the initial query image belongs to. This method is more prone to drift than the method from Chum et al. (2007b), but much more efficient.

Extension: Spectral clustering. Because the cluster growing process explores the connected components of the matching graph, multiple landmarks can potentially end up in the same cluster (see Fig. 3.7b). We therefore combine the above method with the method of Philbin and Zisserman (2008) by performing two additional steps (Fig. 3.8b). We segment the grown connected components with spectral clustering and subsequently merge them using the scheme described in Section 3.4.1. With this hybrid approach we hope to combine the advantages of the two approaches. We do not need to pre-compute the full matching graph, but avoid ending up with clusters that span multiple buildings.

3.4.3 Experiments

To evaluate both landmark discovery approaches on the Paris 500k dataset, we created a clustering ground truth that we introduce in the following. We compare the approaches based on mean cluster purity and mean cluster recall as well as runtime.

Ground Truth. To establish a ground truth clustering, we first over-segmented the matching graph using spectral clustering on the connected components (Sec. 3.4.1). As reported by Philbin and Zisserman (2008), the resulting clusters had a high purity with only a negligibly low number of outliers. We then asked human raters to perform two tasks on the largest clusters returned by spectral clustering. To remove the remaining outliers, we showed each cluster to a human rater who was asked to click on all the images that visibly do not belong to the cluster. To join over-segmented clusters, we showed human raters the canonical images for each pair of clusters and asked them whether they show the same building from the same view. The resulting ground truth consists of 79 clusters covering 94k images. It is publicly available along with the dataset.

Evaluation Measures. Using this ground truth, we evaluate landmark clustering results based on *mean cluster purity* and *mean cluster recall*, which are analogous to precision and recall. In the following, a *clustering* is a set of clusters, and a cluster is a set of images. Let G denote the ground truth clustering, and C a clustering produced by a landmark discovery algorithm, and let N_C and N_G denote the total number of images *covered* by C and G, respectively. An image is covered by a clustering if it is in at least one of the clusters of the clustering. Then, mean cluster purity P and mean cluster recall R are defined as

$$P = \frac{1}{N_C} \sum_{c \in C} \max_{g \in G} \{|c \cap g|\}, \tag{3.8}$$

$$R = \frac{1}{N_G} \sum_{g \in G} \max_{c \in C} \{|c \cap g|\}. \tag{3.9}$$

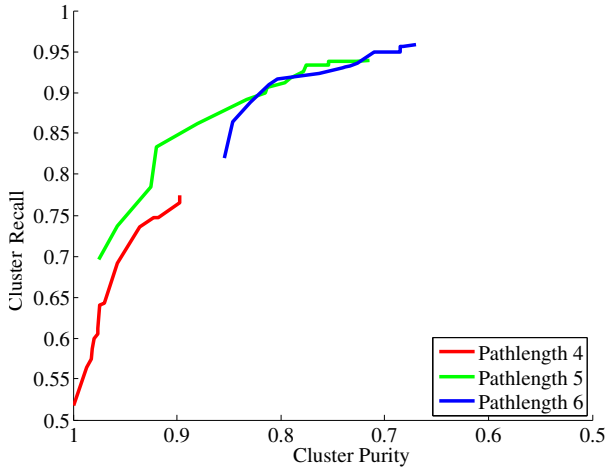


Figure 3.12: *Performance of the spectral clustering method (Sec. 3.4.1). Along each line, the overlap threshold of cluster representatives is varied from 0% to 100%.*

Note that these evaluation measures require *non-overlapping* clusterings, i.e., clusterings in which clusters are pairwise disjoint. While the algorithms we present in the following chapters produce overlapping clusterings, the two algorithms we analyze here each produce non-overlapping clusterings.

Spectral Clustering. For each connected component, we perform a spectral clustering (Sec. 3.4.1). This results in a total of 3,881 clusters. We found that the homography-based merging step requires some tuning to produce the desired results. Because it simply transforms the image boundary coordinates using the product of all homographies along a path, it does not consider that each homography is only valid in its support region. This causes projection errors that accumulate along the path. Limiting the path length is an effective way to restrict this effect. This means that two clusters are not merged if the shortest path between their canonical images exceeds a certain length. Furthermore, we only merge two clusters if the overlap between their representative images is higher than a certain lower bound. Figure 3.12 shows the effect of both parameters on cluster purity and recall. A higher path length threshold leads to a loss in cluster purity, since clusters are incorrectly joined. Too short paths cause us to miss cluster pairs that should be joined, resulting in low recall. The best tradeoff is a path length of 5. Similarly, a too high overlap threshold will cut off paths between valid matches, while a too low threshold will allow paths between barely overlapping images. Even for a suitable choice of these thresholds, the merging step can still perform false merges in some problematic cases typical for internet photo collections: For example, timestamps, watermarks or frames embedded in the photos can create false-positive edges in the matching graph that serve as "tunnels" between unrelated landmarks. We present a method for automatically detecting such cases in Chapter 7.

Geometric min-hash. We now evaluate the performance of the min-hash pipeline

k	# seeds	cluster purity	cluster recall
1	784	50.2%	93.2%
5	4,437	51.7%	96.5%
10	8,753	51.8%	96.8%
30	20,453	51.9%	97.1%
120	51,855	51.9%	97.1%

Table 3.3: *Clustering performance of the min-hash pipeline for different numbers of sketches k. The sketch size is set to $s = 2$.*

(Sec. 3.4.2). The parameters of min-hash seed generation are the sketch size s and the number of sketches k. More sketches will yield more collisions and thus more seeds. Larger sketch sizes make the algorithm more selective, causing it to return only very similar images, which significantly decreases the number of seeds. Since we found that a sketch size of $s = 3$ already yields too few useful seeds, we follow Chum et al. (2009) and set $s = 2$ in our experiments.

Since the probability that two images cause a min-hash collision is proportional to the intersection over union of their bags of visual words, most of the min-hash collisions are duplicates and near-duplicates, i.e., very similar images with only slight differences in color or contrast. We observed that the vast majority of near-duplicate images does not show landmark buildings or other objects of interests, but are often portraits, photos of animals or other non-landmark objects. Chum and Matas (2010) therefore manually removed duplicates for their experiments. We filter out near-duplicates by removing all min-hash collisions whose tf-idf cosine distance (Sec. 3.2.1) is above a certain threshold (0.3) and use the remaining images as seeds from which we grow clusters.

Table 3.3 shows the cluster purity and cluster recall for different settings of k. Cluster recall is very high even when using only a single sketch, suggesting that images forming min-hash collisions tend to depict landmark buildings. However, the cluster purity is only between 50% and 52%, since many connected components explored by our simple version of query expansion cover multiple landmarks.

To examine more closely how suitable min-hash seeds are for landmark discovery, we compare it against random seeding. Figure 3.13 (left) shows the distribution of the min-hash seeds for $k = 60$ and $s = 2$ (31,946 images), and Figure 3.13 (right) shows the distribution of the same number of randomly selected images. The random images are much more scattered over the city while the images selected by min-hash concentrate around landmarks, which is the desired behavior for a seed selection algorithm. For a quantitative comparison, we use the following procedure: For each number of seeds k, we draw as many random images from the dataset as there are min-hash collisions. We draw a set of random images 10 times for each value of k and give the average results for the 10 sets of images. Table 3.4 compares the results of the cluster growing process for min-hash and random seeds. The number of discovered clusters is roughly proportional

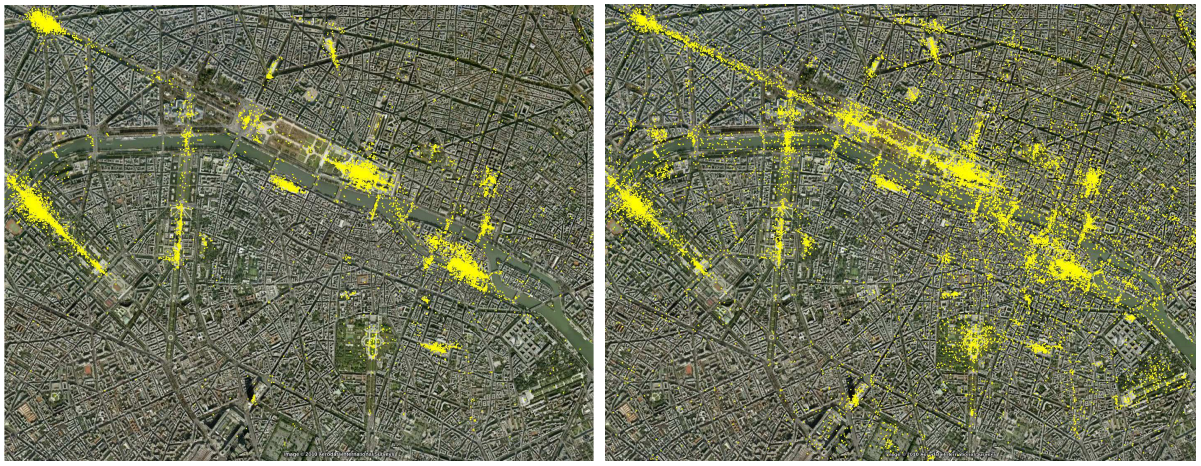


Figure 3.13: *Distribution of min-hash seeds for $k = 60$ and $s = 2$ (left) compared to the distribution of the same number of randomly drawn seed images (right).*

		min-hash		random (avg. of 10)	
k	# seeds	# CCs	avg. CC size	# CCs	avg. CC size
1	784	58	2,158.2	590.0	222.6
2	1,883	102	1,268.9	1,407.5	97.6
3	2,570	141	941.1	1,887.9	74.5
5	4,437	220	620.5	3,236.1	44.9
10	8,753	360	389.3	6,292.2	24.5
30	20,453	915	161.6	14,463.7	11.8
120	51,855	3,022	52.8	35,607.5	5.7

Table 3.4: *Results of cluster growing starting from min-hash seeds and random images, respectively. The sketch size is $s = 2$.*

to the number of seeds. Comparing min-hash to a random selection of seed images shows that roughly ten times the number of connected components are found, but their average size is roughly ten times smaller. This shows that randomly selected images more likely belong to small connected components than images selected using min-hash, confirming that min-hash collisions are more likely to occur in larger clusters (Chum and Matas, 2010).

Hybrid Method. Because the clusters discovered using query expansion become too large and thus cover multiple landmarks, we apply spectral clustering and homography-based cluster merging on top of the min-hash pipeline. Figure 3.14 shows a comparison of this hybrid approach with spectral clustering. The cluster recall of the hybrid method increases with a growing number of sketches and almost reaches the recall of spectral

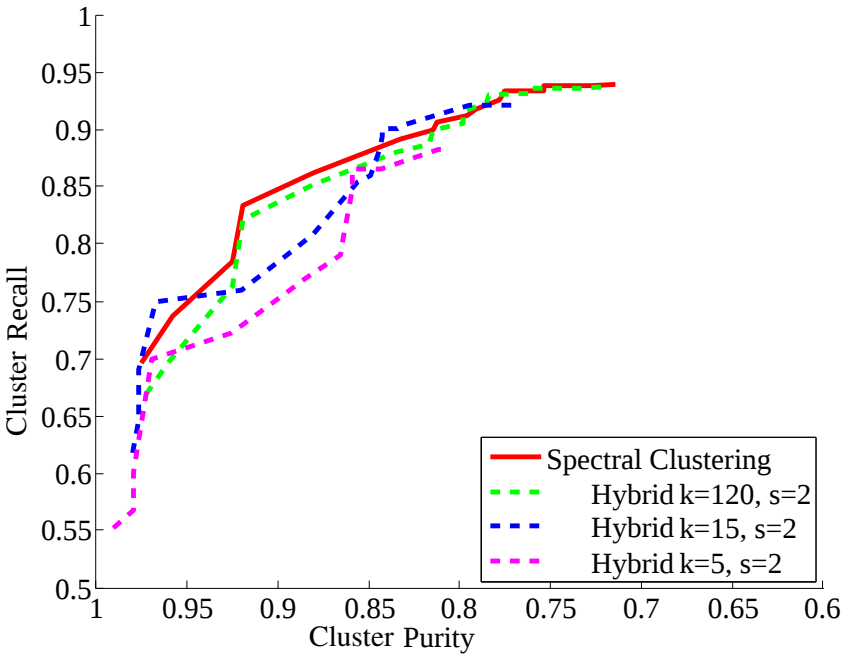


Figure 3.14: *Cluster recall and purity for spectral clustering (Sec. 3.4.1) and the hybrid method that combines min-hash (Sec. 3.4.2) with spectral clustering and subsequent cluster merging. Along each line, the overlap threshold is varied. The path length threshold is set to 5.*

clustering for $s = 120$ sketches. For comparison, min-hash reaches at most a cluster purity of 51.9% (Tab. 3.3).

Runtime Comparison. Since the hybrid method produces comparable results to the spectral clustering method, we now analyze how much time is saved by only performing a local exploration of the matching graph. We will not cover feature extraction time, because this step is necessary for both approaches. The timings are based on a C++ implementation of image retrieval and matching, a C++/Matlab implementation of min-hash, and a Matlab implementation of spectral clustering and homography-based cluster merging. Computations were performed on a computing cluster with Intel processors running between 2GHz and 3GHz.

The computation time of the spectral clustering pipeline (Fig. 3.15) includes pairwise matching (47 days), spectral clustering (14 days) and cluster merging (12 hours). The total computation time is 61.5 days. The computation time of the min-hash pipeline is influenced by the sketch count k. This parameter directly affects the time for computing the min-hashes and it indirectly affects the cluster growing time through the number of discovered clusters. For a choice of $k = 5$, the total runtime is 16.5 days, and for $k = 120$, the runtime is 30 days. Depending on the parameter settings, min-hash is thus two to four times faster.

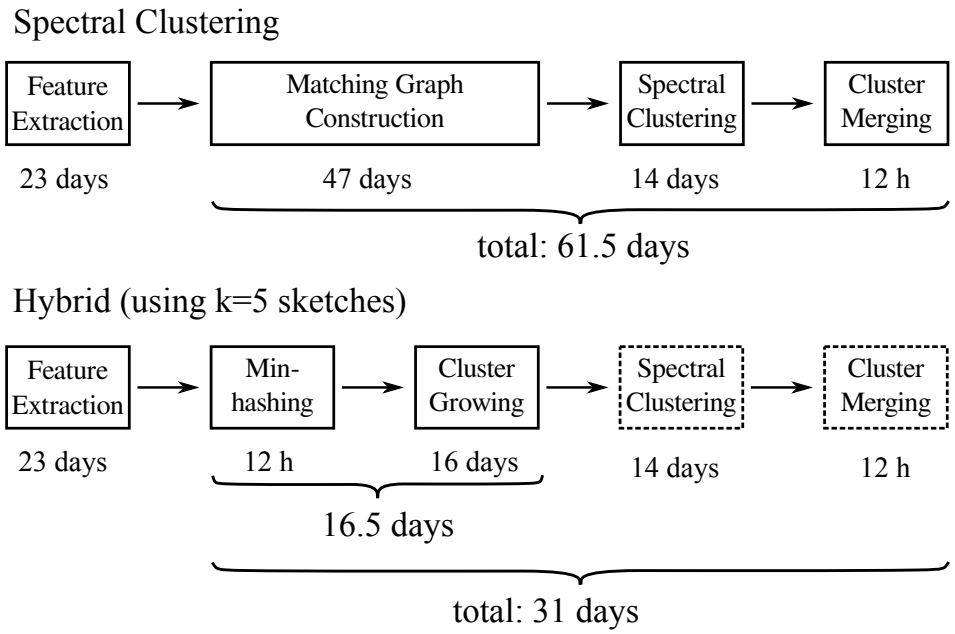


Figure 3.15: *Runtime comparison of the spectral clustering and hybrid methods.*

Figure 3.15 compares the runtime of the spectral clustering method with our hybrid method. The additional steps of spectral clustering and cluster merging (dashed boxes) add a total of 14.5 days to the runtime of the min-hash pipeline. Spectral clustering takes almost the same time in both pipelines, since its runtime is highly dominated by the largest connected components, which are discovered by both approaches. When using $k = 5$ sketches, the total runtime of the hybrid method is about half the runtime of the spectral clustering method, and it is still 17 days faster when using $k = 120$ sketches. Considering that this method has only slightly lower clustering performance (Fig. 3.14), it represents a better tradeoff than the original spectral clustering method.

3.4.4 Discussion

As an initial experiment, we have evaluated two landmark discovery approaches: The spectral clustering based approach by Philbin and Zisserman (2008) and the min-hash based approach by Chum et al. (2009). We found that the spectral clustering approach can produce high quality clusters, but also has high runtime requirements since it needs to compute the full matching graph. Moreover, its heuristic cluster merging step requires some tuning to deliver acceptable results. The min-hash pipeline (Chum and Matas, 2010) performs cluster growing from a set of seed images, and therefore does not need to compute the full matching graph, making it faster than the spectral clustering method. However, without appropriate control of their growth, the explored clusters tend to cover multiple landmarks, resulting in decreased cluster purity compared to the spectral

clustering method. We therefore used a hybrid approach that breaks down the clusters grown by the min-hash method using spectral clustering. This method combines the advantages of both approaches. It is faster than the full spectral clustering pipeline since it only explores the matching graph locally, but can deliver clustering results of comparable quality. This analysis has brought to light some of the challenges that current landmark recognition pipelines are still facing.

Cluster Definition. The hardest challenge seems to be the cluster definition itself. The connected components of the matching graph are an *under-segmentation* of the dataset, because they often contain multiple nearby landmarks. Spectral clustering *over-segments* the matching graph and requires a heuristic merging step to join clusters of the same building. Ideally, we would like a clustering in which each cluster contains all the images of one landmark. However, since the image matching pipeline we presented in this chapter does not have a concept of objects, it is not clear when to stop the exploration process.

Entry Point Selection. Given a seed image of a landmark, we can find other images of this landmark using image retrieval (Sec. 3.2.2). However, it is not clear how to select suitable seed images. While our analysis has confirmed the claim of Chum and Matas (2010) that min-hash based seeds tend to lie on landmark buildings with a higher probability than random images, they are still far from optimal. The majority of min-hash seeds are (near) duplicates, which are usually "junk" images, and even the seeds that show landmarks are not always well-suited for an exploration of the building. For example, an image taken at night or from a very oblique angle might not match most other images of the same building, because local feature based matching (Sec. 3.1) only has limited invariance w.r.t. changes in lighting and perspective. Therefore, it has so far remained an open question how to find suitable images as starting points for exploring the images of a landmark.

Computational Cost. Finally, although inverted indices (Sec. 3.2) enable image retrieval in sub-linear time, runtime is an important issue, especially when considering the rapid growth of the number of photos on the Internet. Even with the computational power available to companies like Google or Facebook, the construction of the full matching graph will soon become infeasible. A local exploration like the one performed by the min-hash method (Sec. 3.4.2) would be able to handle the growing number of images better, since it focuses on the photos that actually show landmarks.

In the following chapter, we present a landmark discovery algorithm that addresses these challenges. We provide answers to the questions of how to define a cluster and how to automatically find suitable starting points for landmark exploration and propose an algorithm that implements these ideas at low computational cost, since it only requires a local exploration of the matching graph.

4

Iconoid Shift: Landmark Discovery by Mode Search

In this chapter, we present the Iconoid Shift algorithm for discovering landmark buildings in Internet photo collections. In particular, we want to address the following problems that still exist in current landmark discovery approaches:

1. *High computational complexity.* Many landmark discovery approaches (e.g. Avrithis et al. (2010); Philbin and Zisserman (2008); Quack et al. (2008); Raguram et al. (2011); Zheng et al. (2009)) require the construction of a matching graph (Sec. 3.2.3) for the whole dataset. This can become expensive as the number of images grows very large. Moreover, a large part of the matching effort is spent on "junk" photos, like pictures of animals or food, that will not form meaningful clusters. The computational complexity of the clustering algorithms that are applied on this matching graph is often very high as well. For example, single-link clustering (Manning et al., 2008) has quadratic complexity in the number of images. Several approaches (Avrithis et al., 2010; Crandall et al., 2009; Gammeter et al., 2009, 2010; Kennedy and Naaman, 2008; Quack et al., 2008; Zheng et al., 2009) therefore pre-group the input images based on their geotags and run their clustering algorithms on each of the resulting groups. This, however, limits them to images with geotags, and can also introduce new errors, since geotags are often inaccurate.

2. *Unsuitable definitions of clusters.* Cluster definitions used in previous work have no notion of objects and therefore tend to *under-segment* (Agarwal et al., 2009; Frahm et al., 2010; Raguram et al., 2011; Snavely et al., 2008a) or *over-segment* (Philbin and Zisserman, 2008) objects (cf. Sec. 2.2 for a more detailed discussion).

   Moreover, most approaches produce a *hard clustering* (e.g. Agarwal et al. (2009); Crandall et al. (2009); Philbin and Zisserman (2008); Quack et al. (2008); Zheng et al. (2009)) where each image can only belong to one cluster. Therefore, if an image shows multiple landmarks, it will be arbitrarily assigned to only one of them. Furthermore, when using a hard clustering, building details, like the windows and

gates on, e.g., the west facade of Notre Dame, cannot form their own clusters since their photos would be assigned to the whole building.

Finally, in most approaches, *every* image will be assigned to a cluster. However, when mining images from the web, the majority of photos will be "junk" photos that do not show an interesting building or object. Such photos should rather be filtered out and not be included in the clustering result.

3. *Unintuitive parameters.* Many approaches are hard to use since their parameters often have very unintuitive interpretations, or ask the user to make almost impossible predictions about the dataset. For example, spectral clustering (Ng et al., 2001) used in Philbin and Zisserman (2008) requires the user to specify the number of clusters, which is not possible since the number of objects in the image collection is unknown in advance. Single-link clustering requires the user to specify a threshold for the linkage between two clusters, i.e., the minimum similarity any two images in the clusters need to have in order to join the clusters. Here, image similarity is often defined as the tf-idf score or the number of inliers of an image pair. Setting this threshold is not trivial. Because a single link between two clusters suffices to merge them, one would choose this threshold rather conservatively to avoid false positive matches from joining clusters. However, a too high threshold might cause images of the same object not to be included in the cluster. Moreover, the number of inliers may vary depending on image size and content (we discuss this in more detail in Section 4.1). Finally, the optimal setting might vary depending on which pair of clusters is considered for joining.

In this chapter, we present an algorithm, called *Iconoid Shift*, that addresses the above problems:

1. It is *fast*, because (i) it does not require the construction of the full matching graph in advance. Instead, it uses *local exploration* to only build the required parts of the matching graph on demand. (ii) The computational complexity of the underlying clustering algorithm does not depend on the number of images in the dataset, since it operates only on local image neighborhoods. Finally, it can be implemented in a distributed fashion to run on hundreds of machines in parallel.

2. It has an *intuitive definition of clusters.* Each landmark is represented by an iconic image, or *Iconoid*, and each cluster simply contains all images that overlap with the Iconoid by more than a certain amount. This definition naturally allows for overlapping clusters and can thus also handle close-by landmarks and even substructures elegantly.

3. It has two *intuitive parameters.* The *kernel bandwidth* simply defines the minimum overlap an image needs to have with an Iconoid to belong to its cluster. The *number of seeds* defines the tradeoff between computation time and number of discovered objects.

Figure 4.1: *A typical sample of the photos of a landmark from an Internet photo collection. Which of them should be the iconic image?*

Since our clusters are defined by iconic images, which image of a building should we choose as the iconic image? We illustrate our definition of iconic images on an example shown in Figure 4.1. An iconic image should show the *whole building*, and the building should be in the *center*, shown from a *frontal viewing angle*, and *fill the entire image*. Hence, in the above example, the middle image would become the Iconoid, and all other images would be in its cluster, since they have an overlap with the Iconoid.

Now that we have defined iconic images, how do we find them automatically? For this, we make use of the distribution of the photos in Internet photo collections. On a city scale, the density of photos is largest around touristic landmarks (Fig. 3.6). On a landmark scale, we observe that the density of photos is highest around iconic views that fit the above definition (Fig. 4.2), while most other photos of a landmark will overlap with its iconic view. Therefore, we define the Iconoid as the image with the

(a) *Pantheon* (b) *Palais du Luxembourg*

Figure 4.2: *Distribution of tourist photos of two landmarks in Paris obtained using structure-from-motion. Each camera position is represented by a black pyramid. The density of tourist photos is highest at central views of the whole building. (Figure courtesy of Torsten Sattler)*

maximum mutual overlap with all other images of the same object. To measure image overlap, previous distance measures based on tf-idf (Manning et al., 2008), the number of inliers, or the distance of global descriptors like GIST (Olivia and Torralba, 2001) are not suitable, since they do not have a direct geometric interpretation. We therefore propose a new distance measure, the *homography overlap distance* that directly measures the spatial overlap of images based on the homography support region. This allows us to find iconic images as density *modes* w.r.t. this distance. Since they are the most frequent views, our definition of iconic views is related to the *frequency* hypothesis of Palmer et al. (1981) which defines iconic views as the views from which an object is most frequently seen. (We relate our definition of iconic views to previous definitions from cognitive psychology in Section 2.1.) However, in contrast to Palmer et al. (1981), we did not choose this definition of iconic views because of aesthetics or the amount of information that the view reveals, but because it allows for a meaningful and intuitively clear definition of clusters.

Since our iconic images are density *modes* w.r.t. the homography overlap distance, we can use *mode search* methods to find them. Specifically, our algorithm is based on Medoid Shift (Sheikh et al., 2007), a variant of Mean Shift (Comaniciu and Meer, 2002) that is applicable to arbitrary metric spaces. Given a seed image, Iconoid Shift explores the images within the kernel support radius by building a local matching graph using recursive image retrieval. It then shifts the kernel center to the image that has maximum overlap with the other images under the kernel. This procedure is repeated until convergence. The intuitive interpretation of this algorithm is as follows: We al-

ternately explore images showing the same object as the current view and shift to the image that shows the depicted structure from the most central (and therefore iconic) viewpoint among their neighboring images. Our algorithm inherits the properties of the well-understood Medoid Shift (Sheikh et al., 2007) and Mean Shift (Comaniciu and Meer, 2002) algorithms, including their guaranteed convergence, intuitive parameters and trivial parallelization.

In summary, this chapter makes the following contributions:

- We present an algorithm that uses mode search to perform landmark discovery in large-scale image collections.
- For this, we introduce a distance measure for images based on their overlap.
- We present efficient algorithms to propagate the overlap between images in the local matching graph, which is faster and more robust to view changes than using direct local feature matching.
- We show how to parallelize our algorithm to many machines.
- We evaluate our algorithm and show that it is capable of discovering popular buildings and objects in two city-scale datasets.

The remainder of this chapter is structured as follows: In the following section, we review the grouping criteria and clustering algorithms used by other landmark discovery algorithms. In Section 4.2, we lay out the foundations by explaining the Mean Shift and Medoid Shift algorithms. We then introduce our overlap-based image distance in Section 4.3. Having all required tools in place, we present the Iconoid Shift algorithm in Section 4.4 and efficient methods for propagating homography overlaps in the matching graph in Section 4.5. We present our evaluation results in Section 4.6 and finally conclude the chapter in Section 4.7.

This chapter is based on our paper (Weyand and Leibe, 2011) presented at ICCV 2011.

4.1 Related Work

Since Iconoid Shift introduces a new grouping criterion and a new clustering algorithm for landmark discovery, we first discuss the grouping criteria and clustering algorithms employed in previous work. We already discussed previous methods for finding iconic images in Section 2.1 and methods for image clustering and object discovery in Section 2.2.

4.1.1 Grouping Criteria

Previous object discovery approaches have applied different grouping criteria to cluster images. Crandall et al. (2009) group photos by the proximity of their geotags by simply

running Medoid Shift on their GPS coordinates. They determine the most frequently photographed objects by photo tags and descriptions, but ignore visual similarity for their clustering.

Some approaches group photos by their global similarity. Li et al. (2008); Raguram and Lazebnik (2008) apply k-means clustering on GIST descriptors (Olivia and Torralba, 2001), and Kennedy and Naaman (2008) apply k-means on Gabor textures and color moment features. While this achieves a basic image grouping by content, clustering by global appearance typically produces clusters of highly similar images. Moreover, an image match based on global descriptors has no geometric meaning and is no guarantee that the images actually show the same object. Finally, global descriptors only have a limited degree of robustness w.r.t. occlusions or lighting changes.

Berg and Berg (2009) compute an image distance based on the average matching score of local geometric blur descriptors (Berg and Malik, 2001) and perform k-medoids clustering (Kaufman and Rousseeuw, 1990). While local features are more invariant w.r.t. image transformations than global features, a descriptor matching without spatial verification cannot guarantee geometric consistency.

The most popular grouping criterion is the number of inlier feature correspondences, w.r.t. a geometric transformation such as a homography (Agarwal et al., 2009; Avrithis et al., 2010; Cao and Snavely, 2013; Frahm et al., 2010; Gammeter et al., 2009, 2010; Philbin and Zisserman, 2008; Quack et al., 2008; Raguram et al., 2011; Snavely et al., 2008a; Zheng et al., 2009). Since clustering approaches that operate in Euclidean space such as k-means are not applicable here, these approaches build matching graphs (Sec. 3.2.3) and apply different graph clustering algorithms (see below). While the number of inliers can allow more direct conclusions about the geometric relationship between images, it also depends on the number of interest points extracted from the input images, which varies depending on image resolution and the amount of texture in the scene. For example, two images might only share a small region, but still have many inliers if this region is highly textured. In contrast, two images can have a large area in common although they do not have many inliers if only few interest points are detected on the object they depict. Irschara et al. (2009) observed that unevenly distributed inliers tend to cover a smaller fraction of the image than evenly distributed inliers and account for this by weighting the number of inliers with the estimated fraction of the image covered by them. However, even this weighted count will vary with the amount of texture in the images. Another problem of using the number of inliers as a grouping criterion is that it depends on the invariance of the underlying feature matching pipeline. While most interest point detectors and descriptor offer some level of invariance w.r.t. perspective and lighting changes, matching can break down if the images are taken at different times of day or from too different vantage points (Mikolajczyk and Schmid (2004) perform an analysis of the perspective invariance of different interest point detectors). This would cause their amount of shared content to be underestimated. Another consequence of this is that the number of inliers is not *transitive*. Although the image pairs (A, B)

and (B, C) are matches, A and C do not necessarily match too if the amount of change between them is too high.

Simon et al. (2007) therefore even take a step further and first reconstruct the scene using structure-from-motion and then define image similarity based on the number of common 3D points that two views are seeing. This has the advantage of being a transitive definition, because two images have shared 3D points even if they do not match. Even if a 3D point was not triangulated from the local features of an image, they can still infer from the camera position and orientation that this point is visible in an image. This method is therefore not limited by the invariance of the feature matching process.

Our homography overlap distance measure directly estimates the overlap between images. It is therefore independent of the number of inliers, has a direct geometric interpretation and allows for a grouping on the *object* level. Moreover, it can even be inferred for image pairs that do not have a direct match as long as they are connected via a path in the matching graph.

4.1.2 Clustering Algorithms

A large number of different clustering algorithms have been used for object discovery. Kennedy and Naaman (2008); Raguram and Lazebnik (2008) use *k-means*, which is only applicable in Euclidean spaces and thus not suitable for most similarity measures that are based on local features. Berg and Berg (2009) use *k-medoids* (Kaufman and Rousseeuw, 1990), which restricts cluster centers to lie on data points. This makes it applicable to non-Euclidean spaces and has the advantage that the cluster centers are images and not arbitrary points in descriptor space. Berg and Berg (2009) therefore use the medoids as iconic images. However, both k-means and k-medoids require the user to specify the desired number of clusters in advance, which is not possible in most object discovery applications.

Several approaches from the large scale structure-from-motion literature (Agarwal et al., 2009; Frahm et al., 2010; Raguram et al., 2011; Snavely et al., 2008a) build a matching graph by estimating pairwise epipolar geometries and creating an edge between all image pairs that have a sufficient number of inliers. They then run structure-from-motion on each *connected component* of this graph. As we discussed in Section 3.4, the connected components of the matching graph typically cover multiple buildings. This is desired in the case of structure-from-motion approaches which try to reconstruct entire city areas. For landmark recognition applications, however, we are more interested in a clustering on the scale of individual buildings or objects.

Several papers aimed at landmark recognition (Gammeter et al., 2009, 2010; Quack et al., 2008; Zheng et al., 2009) use *Single-Link Clustering* (Manning et al., 2008). This enables clustering at a finer scale, but requires the choice of a linkage threshold, which is far from trivial (see above). Furthermore, single-link clustering has quadratic computational complexity in the number of input images, making it unsuitable for larger image collections unless a pre-grouping based on geotags is performed. Crandall et al.

(2009); Philbin and Zisserman (2008) apply *Spectral Clustering* (Ng et al., 2001) on the matching graph. We analyzed this algorithm in Section 3.4 and could confirm the observation of Philbin and Zisserman (2008) that it tends to produce an over-segmentation of the dataset, making it necessary to join clusters of the same object in a post-processing step. Moreover, its high computational complexity makes it unsuitable for large-scale image collections and, like k-means, it requires choosing the desired number of clusters in advance.

The above algorithms all produce a *hard* clustering, where each image is assigned to exactly one cluster. Therefore, these algorithms cannot handle images showing multiple landmarks or represent buildings with sub-structures. Since they assign *every* image to a cluster, they also do not discard junk images. We now look at object discovery approaches that produce an *overlapping* clustering.

The object discovery algorithm of Chum and Matas (2010); Chum et al. (2009), which we analyzed in detail in Section 3.4, first applies min-hashing to generate a set of seed images. It then uses recursive *Query Expansion* (Chum et al., 2007b) to discover object clusters. While the resulting clusters can be overlapping, query expansion is prone to object drift, which can still cause clusters to contain multiple objects. Avrithis et al. (2010) use *Kernel Vector Quantization* (Tipping and Schölkopf, 2001) for visual clustering, which selects a minimal subset of the input images such that each image has a certain minimum number of inliers with the cluster center. Its definition therefore allows for an image to be in multiple clusters. However, Kernel Vector Quantization does not select density maxima, but an arbitrary subset of images whose kernels cover the dataset. The clusters are therefore not necessarily meaningful objects. Moreover, by its definition, it does not allow images to be discarded, so junk images will still form clusters.

To the best of our knowledge, ours is the first object discovery approach that uses *Medoid Shift*, which has several desirable properties for this application. (i) It converges to local density maxima, which correspond to popular views. (ii) It is able to handle images showing multiple landmarks as well as sub-structures since it produces an overlapping clustering. (iii) Images that are outside of the radius of any Iconoid are considered "junk" and are not included in the clustering result, and (iv) it is efficient and easily parallelizable, because it only operates on local neighborhoods.

4.2 Mode Search

To lay the foundations for Iconoid Shift, we first give a brief review of mode search based on Mean Shift (Cheng, 1995) and Medoid Shift (Sheikh et al., 2007) and introduce the homography overlap distance in Section 4.3. We then introduce the Iconoid Shift algorithm in Section 4.4.

4.2.1 Mean Shift

In mode search, we seek to find the density maxima, or *modes*, in a dataset. Additionally, we want to associate each point in the dataset with one of the modes to create a clustering. The density modes of a given dataset $\{\mathbf{x}_i\}$, are the the maxima of its kernel density estimate $f(\mathbf{x})$:

$$f(\mathbf{x}) = c \sum_i \Phi(\mathrm{d}(\mathbf{x}, \mathbf{x}_i)). \tag{4.1}$$

Here, $\Phi(\cdot)$ is a kernel, e.g. a Gaussian, $\mathrm{d}(\cdot)$ is a distance function, e.g. the Euclidean distance, and c is a normalization constant, such that $f(\mathbf{x})$ integrates to 1. We can think of this function as a mountainous terrain where $f(\mathbf{x})$ gives the height at each point $\mathbf{x}$. Starting at any point, we can find a maximum simply by walking uphill, i.e., by performing *gradient ascent*. The maxima of the kernel density are the density modes of the data.

In their influential work, Fukunaga and Hostetler (1975) proposed an algorithm called *Mean Shift* that finds the density maxima by iteratively shifting a kernel window to the weighted mean of the points within it. That is, given the current point $\mathbf{y}_k$, we shift to the new point $\mathbf{y}_{k+1}$ given by:

$$\mathbf{y}_{k+1} = \frac{\sum_i \mathbf{x}_i \varphi(\mathrm{d}(\mathbf{y}_k, \mathbf{x}_i))}{\sum_i \varphi(\mathrm{d}(\mathbf{y}_k, \mathbf{x}_i))} \tag{4.2}$$

Cheng (1995) showed that Mean Shift is *equivalent* to gradient ascent in the kernel density if the kernel $\varphi(\cdot)$ is the negative derivative of the kernel $\Phi(\cdot)$. The kernel $\Phi(\cdot)$ is then referred to as the *shadow* of $\varphi(\cdot)$

$$\varphi(\mathbf{x}) = -\Phi'(\mathbf{x}). \tag{4.3}$$

To cluster a dataset with Mean Shift, the algorithm is initialized once with each data point and the update rule (Eq. (4.2)) is repeated until convergence at a mode. Then, each data point is associated with the mode it converged to. The resulting clusters are therefore sets of points that converge to the same mode. If the goal is not a complete clustering of the dataset, but only to find the modes, the algorithm can be initialized with only a smaller set of *seed* data points. This has the risk of missing modes, but brings significant savings in computation time.

In computer vision, Mean Shift has become a popular algorithm with applications in image segmentation (Comaniciu and Meer, 2002) and tracking (Comaniciu et al., 2000). It is often preferred over k-means since it does not require the user to specify the desired number of clusters in advance and because it is not limited to convex clusters.

4.2.2 Medoid Shift

Mean Shift, despite its success, has two problems that limit its applicability: (i) It can only be used in Euclidean spaces, and (ii) the resulting modes do not necessarily lie on

data points and therefore do not always have a direct interpretation. An example for the latter is clustering of global image descriptors, where only the data points themselves correspond to images, but other points in space do not have a meaningful interpretation.

The Medoid Shift algorithm by Sheikh et al. (2007) solves these problems. They observed that the Mean Shift minimization (Eq. (4.2)) is equivalent to shifting the kernel center to the point with lowest sum of weighted distances under the kernel:

$$\mathbf{y}_{k+1} = \arg\min_{\mathbf{y}} \left\{ \sum_i \mathrm{d}(\mathbf{y}, \mathbf{x}_i) \varphi(\mathrm{d}(\mathbf{y}_k, \mathbf{x}_i)) \right\}. \tag{4.4}$$

The only formal difference in Medoid Shift is that the kernel center must always lie on a data point:

$$\mathbf{y}_{k+1} = \arg\min_{\mathbf{y} \in \{\mathbf{x}_i\}} \left\{ \sum_i \mathrm{d}(\mathbf{y}, \mathbf{x}_i) \varphi(\mathrm{d}(\mathbf{y}_k, \mathbf{x}_i)) \right\}. \tag{4.5}$$

This small change generalizes Mean Shift from Euclidean spaces to general metric spaces (e.g. fully connected graphs). The Medoid Shift algorithm only requires a distance metric to be defined between each pair of data points. It therefore suffices to have a distance $\mathrm{d}(\cdot, \cdot)$ that fulfills:

- Symmetry: $\mathrm{d}(x, y) = \mathrm{d}(y, x)$
- Identity of indiscernibles: $\mathrm{d}(x, x) = 0$
- Triangle inequality: $\mathrm{d}(x, z) \leq \mathrm{d}(x, y) + \mathrm{d}(y, z)$

The price for this generality is a higher computational complexity. While Mean Shift only requires linear time to compute the weighted mean of the data points under the kernel window, Medoid Shift requires quadratic time, since it needs to compute the weighted sum of distances from each point under the kernel window to each other point.

4.3 Homography Overlap Distance

Iconoid Shift applies Medoid Shift to find iconic views in large community photo collections such as Flickr or Panoramio. These views, called *Iconoids*, are modes w.r.t. the *homography overlap distance*, which we define in the following. Here, x_{ij} is the bounding box around the inlier features in image j and x_{ji} is the bounding box around the inlier features in image i (see Fig. 4.4). Since the modes we are searching for are images having maximal overlap with their neighborhood, we need a distance measure that rewards similar views while penalizing view changes like panning and zooming. To determine the *overlap region* between two images i and j, we estimate a homography H_{ji} (Sec. 3.1.3) that maps from image i to j. In order for the overlap regions to be consistent with

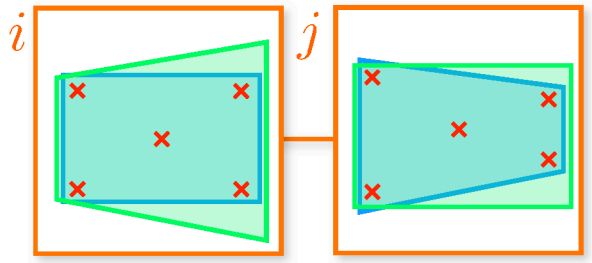


Figure 4.3: *Two possible definitions of the overlap regions. Blue: Axis-aligned bounding box in image i, projected into image j. Green: Axis-aligned bounding box in image j, projected into image i.*

the homography, we cannot simply define both overlap regions as axis-aligned bounding boxes, because the overlap regions would not be aligned when applying the homography. We therefore define only one overlap region as an axis-aligned bounding box and the other as its image w.r.t. the homography. As shown in Figure 4.3, we have two choices for this. We choose the pair of boxes that encloses the set of inliers better, i.e., the pair whose sum of areas is smaller (the blue boxes in Fig. 4.3). This makes the overlap regions consistent with the homography,.

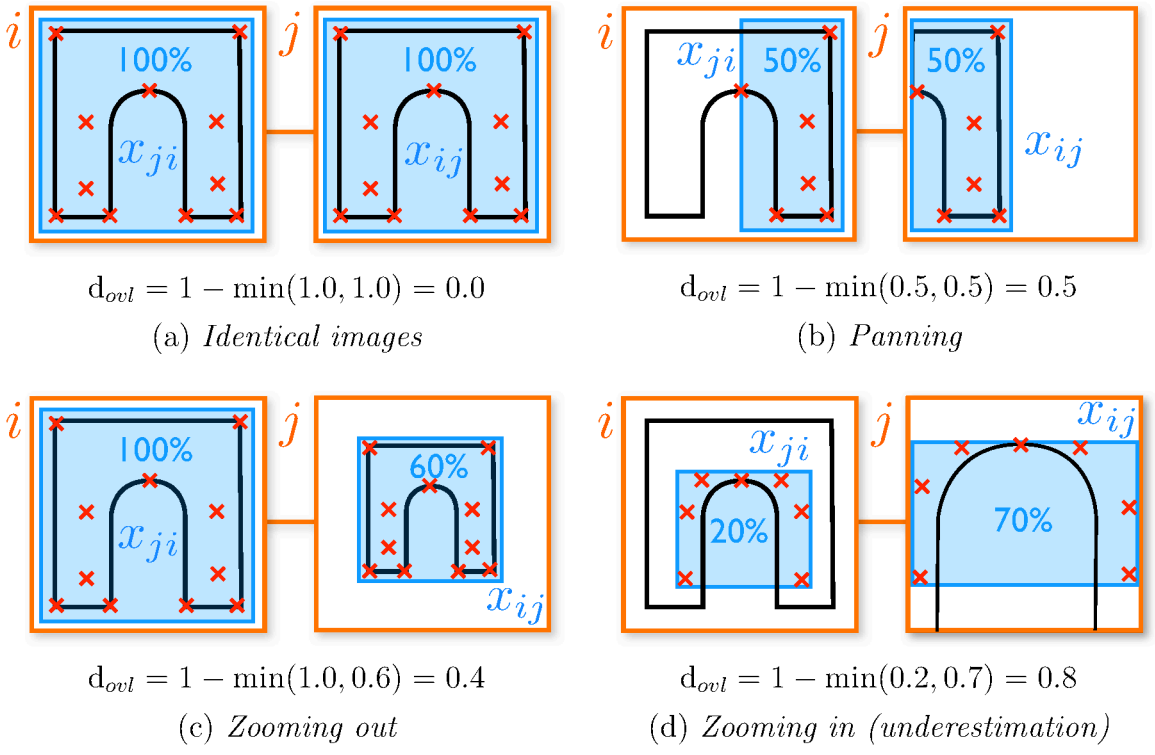
We then compute the relative size of the overlap regions in both images and define the *homography overlap distance* as one minus the minimum of these relative sizes:

$$\mathrm{d}_{ovl}(i,j) = 1 - \min\left\{\frac{||x_{ji}||}{||R_i||}, \frac{||x_{ij}||}{||R_j||}\right\}. \tag{4.6}$$

Here, $||R_i||$ and $||R_j||$ denote the area of image i and j, respectively.

4.3.1 Properties

Some examples of computing the homography overlap distance are shown in Figure 4.4. If the images are identical (Fig. 4.4a) their overlap distance is 0, since both inlier bounding boxes fill the whole images. If we pan the view (Fig. 4.4b), the size of the overlap region decreases equally in both images and d_{ovl} increases. In the case of zooming out (Fig. 4.4c) or zooming in (Fig. 4.4d), the relative size of the smaller overlap region determines the value of d_{ovl}. As illustrated in Figure 4.4d, this method can sometimes underestimate the overlap due to homogeneous image regions where no interest points are present. We take these small errors into account in our experiment and leave more exact methods for estimating the overlap, e.g. based on photoconsistency or structure-from-motion, for future work.



$d_{ovl} = 1 - \min(1.0, 1.0) = 0.0$

(a) *Identical images*

$d_{ovl} = 1 - \min(0.5, 0.5) = 0.5$

(b) *Panning*

$d_{ovl} = 1 - \min(1.0, 0.6) = 0.4$

(c) *Zooming out*

$d_{ovl} = 1 - \min(0.2, 0.7) = 0.8$

(d) *Zooming in (underestimation)*

Figure 4.4: *Estimation of the homography overlap distance. Red crosses denote inlier features. Blue boxes denote the overlap regions.*

4.3.2 Transitive Homography Overlap Distance

To determine local modes, Medoid Shift requires the pairwise distances of all images within the kernel radius. However, computing these pairwise distances by direct feature matching (as done, e.g., in Quack et al. (2008)) is very costly. Instead, we compute the *minimum spanning tree* of the images within the kernel radius and infer the distances of all image pairs using their connecting path in the tree.

A simple approach to compute the overlap of two images would be to multiply all homographies along their connecting path in the minimum spanning tree, computing the set of inliers of the images w.r.t. this homography, and determining the overlap region based on them. However, our experiments (Sec. 4.6.4) show that this method is not robust due to the limited invariance of the interest point detectors and descriptors and thus strongly underestimates the overlap. Furthermore, this method is computationally expensive, because it needs to determine the homography inliers for every image pair.

In contrast, the method we present in the following is independent of feature matches and avoids the costly step of determining the homography inliers. Therefore, it is able to bridge much larger variations in both viewing angle and lighting, because it directly transforms the overlap region itself.

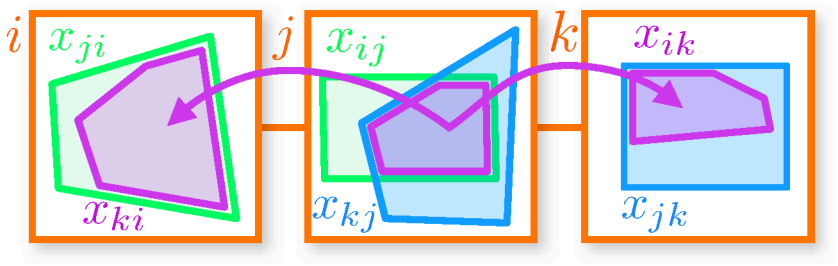


Figure 4.5: *Transitive homography overlap computation. The intersection of x_{ij} and x_{kj} is projected into both i and k and the homography overlap distance (Eq. (4.6)) is computed using x_{ki} and x_{ik}.*

This scheme is inspired by the homography-based cluster merging scheme by Philbin and Zisserman (2008) that we described in Section 3.4.1. However, our propagation scheme avoids errors in overlap estimation by not projecting the whole image region, but instead restricting the projected region to only the homography support.

In the following, assume there exists a direct correspondence between the image pairs (i, j) and (j, k) and our goal is to infer the homography overlap distance of (i, k). As a simple example, consider Figure 4.5. We estimate the overlap regions x_{ki} and x_{ik} (purple regions in images i and k) by intersecting x_{ij} with x_{kj} (the green and blue regions in image j), and projecting the intersected region into images i and k using the known homographies. The homography overlap distance can then be computed as in Eq. (4.6) without explicitly matching i and k. Likewise, we can easily compute the homography from i to k as

$$H_{ki} = H_{kj} H_{ji}. \tag{4.7}$$

We denote the projection of an overlap region by

$$x_{ji} = H_{ij} x_{ij}. \tag{4.8}$$

In this notation, the whole area of the overlap region is projected by the homography. To define this scheme formally, let $x \cap y$ be the intersection between two regions x and y. We then define the overlap region of image i in image k as

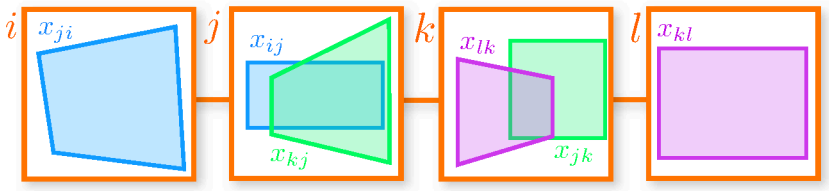
$$x_{ik} := H_{kj}(x_{ij} \cap x_{kj}). \tag{4.9}$$

Now, x_{ki} can be computed analogously:

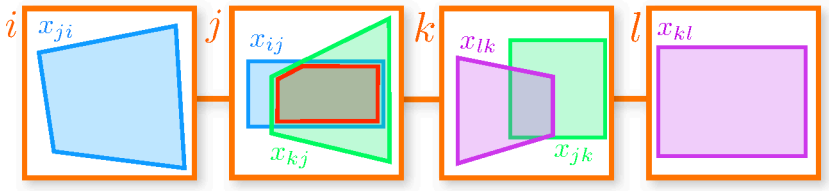
$$x_{ki} = H_{ij}(x_{ij} \cap x_{kj}) \tag{4.10}$$

By applying this scheme recursively, we can propagate overlap regions along paths. As an example, consider a path of four images (i, j, k, l) with correspondences only between adjacent images (Fig. 4.6). We can compute x_{il} using Eq. (4.9) twice:

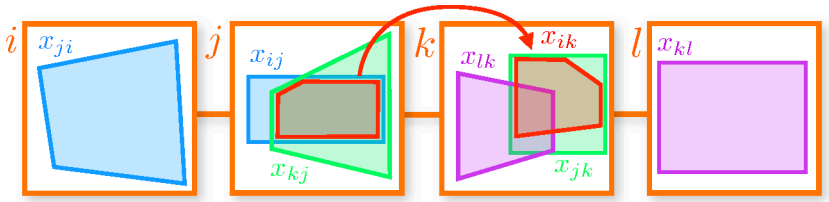
$$x_{il} = H_{lk}(H_{kj}(x_{ij} \cap x_{kj}) \cap x_{lk}) \tag{4.11}$$



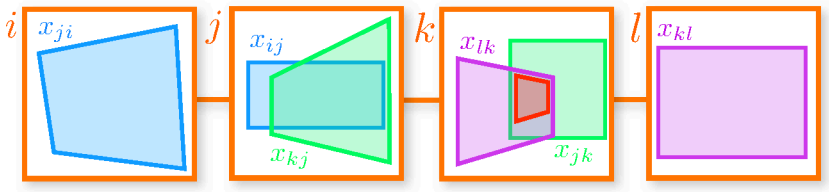
(a) *Input: A sequence of four images.*



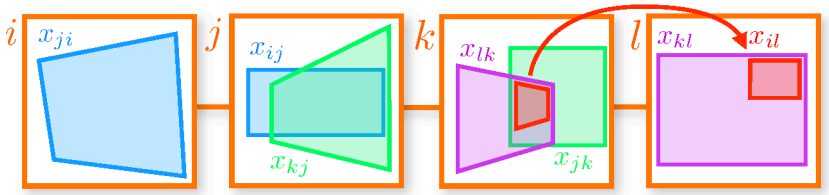
(b) x_{ij} *and* x_{kj} *are intersected.*



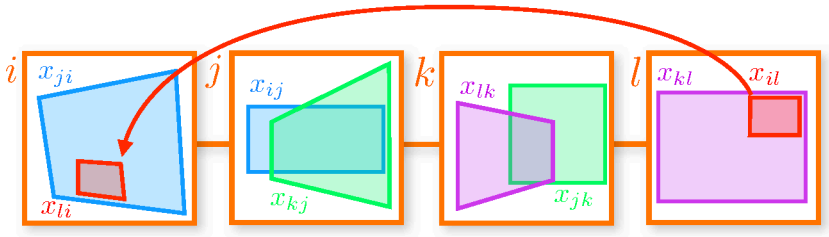
(c) *The intersection is projected into* k *to yield* x_{ik}.



(d) x_{ik} *is intersected with* x_{lk}.



(e) *The intersection is projected into* l, *yielding* x_{il}.



(f) x_{il} *is projected back into* i, *yielding* x_{li}.

Figure 4.6: *Homography overlap propagation along a path.*

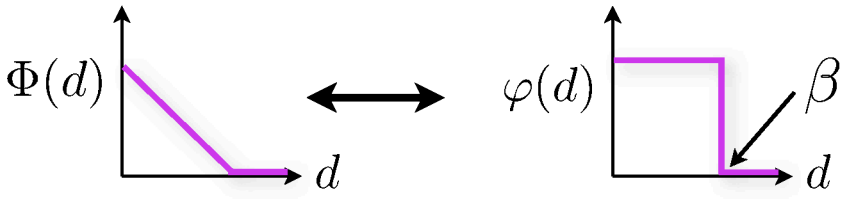


Figure 4.7: *Profiles of our used kernel pair. Left: Shadow kernel (hinge function), right: Medoid Shift kernel (step function).*

By alternating intersection and projection in this way, we can determine the transitive homography overlap distance between any pair of images that are connected through a path of pairwise correspondences. In the algorithm that we will present in Section 4.5.2, we compute the overlaps of all image pairs within the kernel radius by propagating their overlap through their connecting path in a minimum spanning tree. By computing homography overlap distances using this propagation scheme, we can guarantee that the triangle inequality is fulfilled. In practice, violations only occur in cases where the polygon intersection algorithm fails due to degenerated polygons. Those cases are so rare that they do not have any effect on the algorithm's convergence and can easily be filtered out.

4.3.3 Hinge Kernel

Having introduced our distance function on images, we define the *shadow* (Sec. 4.2.1) kernel $\Phi(\cdot)$ as a *hinge function* that is 0 for all distances above a threshold β (Fig. 4.7). The kernel $\varphi(d) = -\Phi'(d)$ then becomes a *step function* that cuts off all distances greater than β:

$$\Phi(d) = \left(1 - \frac{d}{\beta}\right) \text{ if } d < \beta \text{ , 0 otherwise} \tag{4.12}$$

$$\varphi(d) = \frac{1}{\beta} \text{ if } d < \beta \text{ , 0 otherwise} \tag{4.13}$$

4.4 Iconoid Shift

Now, we have all the components that are needed to define Iconoid Shift. It is based on Medoid Shift, but makes some extensions to incorporate image retrieval and overlap distance computation.

The basic principle of the Iconoid Shift algorithm is visualized in Figure 4.8. Starting from an initial seed image, the *root*, we construct a minimum spanning tree of its *support set*, i.e., all images within its kernel window, by locally exploring its neighborhood using recursive image retrieval. Matches are verified by computing their homography overlap distance (Eq. (4.6)) with the root node. For the children of the root, this is done directly

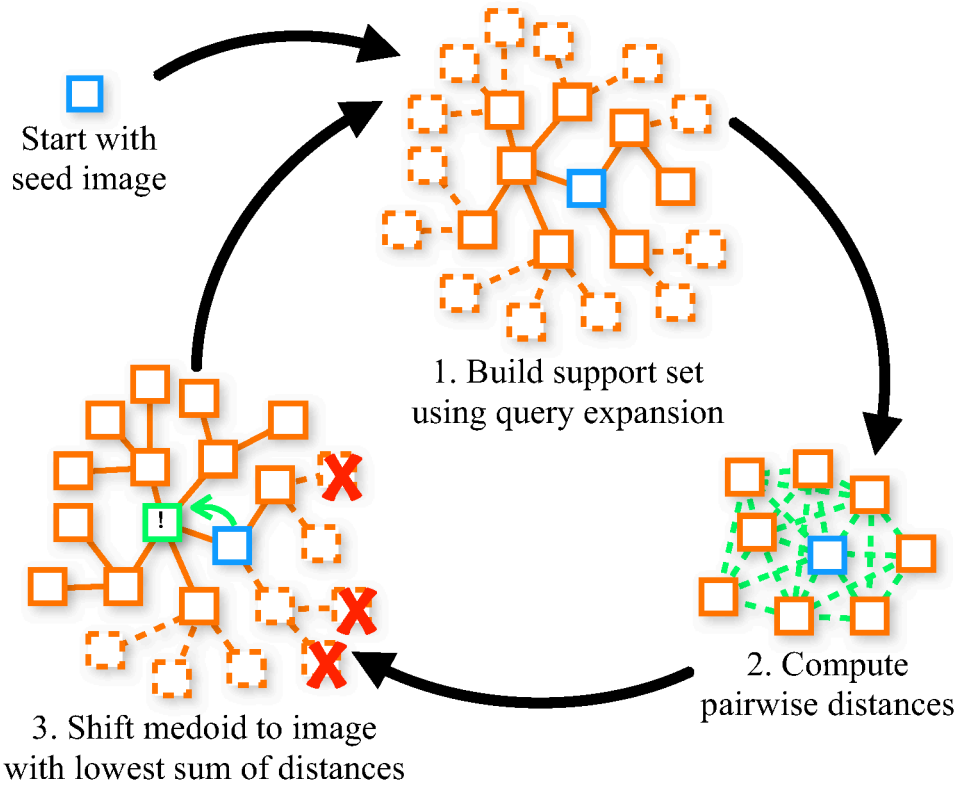


Figure 4.8: *Steps of the Iconoid Shift algorithm. Dashed boxes denote images outside the current kernel window.*

(Fig. 4.3). For nodes farther away, the overlap is propagated transitively (Fig. 4.6). In the second step, we compute the pairwise distances between all images in the graph. This can be performed very efficiently (cf. Sec. 4.5) by again exploiting the transitive definition of the homography overlap distance (Fig. 4.6). Finally, we determine the image with the smallest sum of weighted overlap distances (Eq. (4.5)) and iterate this procedure using this image as the new root. This is repeated until a convergence point, the *Iconoid*, is reached. This mode search is performed for a previously selected group of *seed* images and the set of resulting Iconoids and their support sets are returned. The full algorithm is shown in Alg. 1. It can easily be parallelized by distributing the mode search for different seeds to multiple threads or compute nodes.

The algorithm's steps have an intuitive interpretation: Starting with a seed image, we explore the set of images overlapping with it and compute their pairwise overlaps. Then, we compute the weighted sum of overlap distances for each image (Eq. (4.5)). This sum is lowest for the image that has the highest overlap with all other images. This image is the most iconic view of the object under the current kernel window. The search converges at the Iconoid, which is the most popular view of the object in the

Algorithm 1 *Iconoid Shift.*

```
// Input: A collection C of tourist photos.
// Output: The set of P Iconoids and their support sets.
S ← draw a set of seed images from C.
for s ∈ S do
  y_0 ← ∅, y_1 ← s, k ← 1
  while y_{k-1} ≠ y_k do
    Build minimum spanning tree T starting from y_k. (Sec. 4.5.1)
    Complete missing edges in T by propagating overlaps. (Sec. 4.5.2)
    y_{k+1} ← the image in T minimizing Eq. (4.5).
    k ← k + 1
  end while
  If y_k is a new Iconoid, add (y_k, T) to P.
end for
```

local neighborhood of the initial seed image. The Iconoid tends to be the most frontal and centered view on an object (see Fig. 4.10).

In contrast to Mean Shift or Medoid Shift, our approach produces an *overlapping clustering*, since we define the clusters as the Iconoid support sets, i.e., all images whose overlap with the Iconoid is larger than $1-\beta$. The cluster definition of the original Mean Shift would not make sense in this case, since the seed image might not have an overlap with its Iconoid. Unlike, e.g., Agarwal et al. (2009); Philbin and Zisserman (2008) we do not compute the full matching graph but only the local neighborhoods of the images along the kernel trajectory, which reduces computation time.

4.5 Efficient Implementation

We now introduce efficient algorithms for both the exploration and the distance computation steps of Iconoid Shift. The proposed exploration procedure builds a minimum spanning tree of images overlapping with a central image, which enables an efficient implementation of the pairwise distance computation by homography overlap propagation (HOP). In particular, by interleaving the distance computation and Medoid Shift minimization steps, the memory requirements of our algorithm are linear in the number of images within the kernel window.

4.5.1 Local Exploration and Minimum Spanning Tree Construction

The exploration procedure works by querying an image retrieval system (Sec. 3.2) with the root node r to obtain potentially matching images. Each match i is verified by computing the homography overlap distance with the root node $\mathrm{d}_{ovl}(i, r)$. If this distance is within the kernel radius, i.e., $\varphi(\mathrm{d}_{ovl}(i, r)) > 0$, the match is accepted and added to the

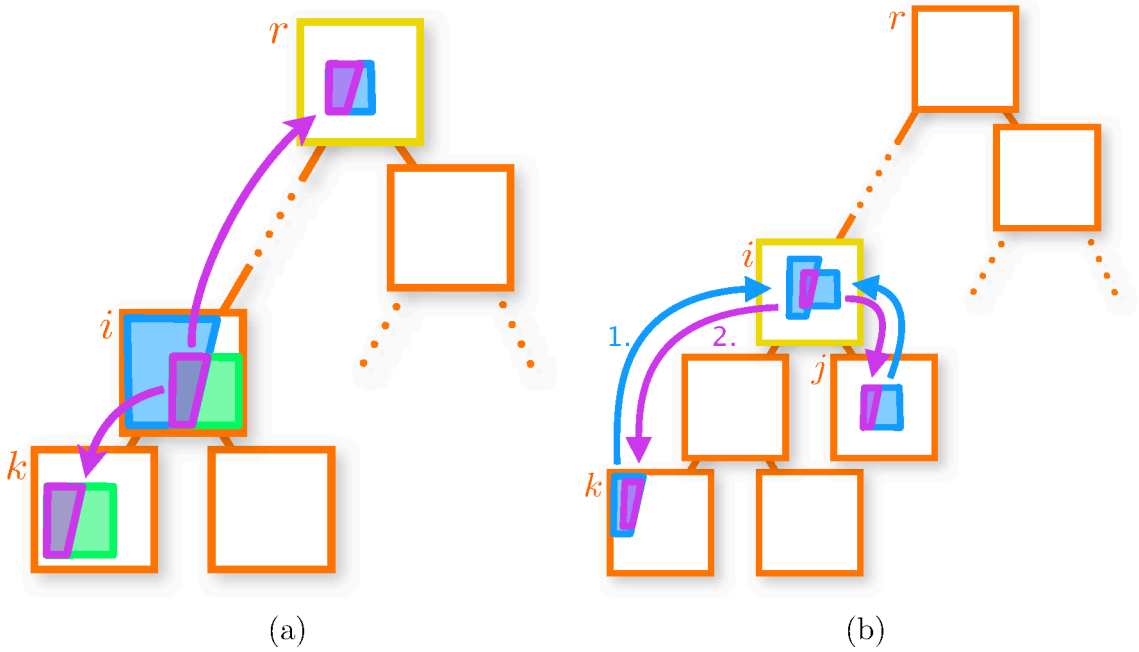


Figure 4.9: *Homography overlap propagation. (a) Computing the overlap of the root r with a new node k. (b) Distances between nodes in different sub-trees are propagated via their lowest common parent i.*

graph. This procedure is executed recursively, building up a local matching graph in a breadth-first manner (Fig. 4.8, step 1). In order to efficiently compute the homography overlap distances with the root node, each node i stores its overlap region with the root x_{ri} and its homography with the root H_{ir}. After a new match k has been retrieved for an inner node i in the graph, its overlap with the root is computed as follows (Fig. 4.9a): First, the overlap of k and i (green) is computed and intersected with the known overlap region of the image i with the root (blue). The intersected region (purple) is then projected into k using the homography that has been estimated by the image retrieval engine and into the root using the known homography H_{ir} that is stored in i. Finally, the overlap distance $\mathrm{d}_{ovl}(i, r)$ is computed using Eq. (4.6). This way, only $O(N)$ propagation steps have to be performed, where N is the number of images within the kernel radius. After the distances of all nodes from the root are known, we can pre-compute the kernel weights $\varphi(\mathrm{d}_{ovl}(r, j))$ for each node j that we need for the following step. Finally, we compute the minimum spanning tree of the local matching graph, using overlap distances as edge weights.

4.5.2 Homography Overlap Propagation (HOP)

Having constructed the minimum spanning tree, we now compute the distances between all pairs of nodes (Fig. 4.8, step 2) and determine the medoid using Eq. (4.5). A naive implementation of this step would require runtime and storage cost in $O(N^2)$ which quickly becomes infeasible for large image sets. Instead, we propose an efficient divide-and-conquer algorithm that requires only linear space.

Our algorithm exploits the transitive homography overlap distance (Sec. 4.3.2) to propagate overlaps in the minimum spanning tree. The central idea is that for two images in different subtrees, propagation always goes through the lowest common parent. The overlap of this parent with all images under it can be pre-computed and re-used for all pairs of images below it. In the homography overlap propagation algorithm, we iterate through the graph in a breadth-first fashion, making each node the lowest common parent once.

For each lowest common parent i, we proceed in two steps: First, we propagate the homography overlap of i to all nodes j in its sub-tree in the same way as in the exploration stage (Fig. 4.9a). Second, we compute the distances between all pairs of nodes (j, k) that are connected via i, that is all nodes in *different sub-trees* of i. This step is shown in Figure 4.9b. The overlaps of k with i and j with i (blue regions in image i) were already computed in the first step. We intersect them to obtain the purple region that we project into both k and j using the homographies we computed during the first step. Then, we compute the distance $\mathrm{d}(j, k)$ using Eq. (4.6). The summation of weighted distances (Eq. (4.5)) is performed on the fly after each distance computation using the weights we computed in the local exploration stage. Finally, we return the node with the minimum weighted sum of distances. The full algorithm is given in Alg. 2.

This algorithm has $O(N)$ memory complexity in the number of nodes N, because we directly accumulate the kernel-weighted sums of distances (Eq. (4.4)) instead of storing all $N^2/2$ pairwise distances and computing the weighted sums in a separate step. The time complexity of this algorithm is $O(N^2)$. Each overlap propagation step enables us to compute the homography overlap distance between two nodes. There are N propagation targets and N lowest common parents overall. Thus, $O(N^2)$ top-down propagation steps are performed. The number of pairwise distance calculations in different sub-trees is also $O(N^2)$, because one distance calculation is performed for each pair of nodes where no node is an ancestor of the other.

4.5.3 Parallel Tree Traversal

To cluster a set of data points, Medoid Shift (Sheikh et al., 2007) first computes the successor for each input data point by performing the minimization from Equation (4.5). This produces a directed graph with one outgoing edge per data point. Then, by simply following edges in this graph, the algorithm determines the modes of the dataset.

Algorithm 2 *Homography Overlap Propagation (HOP).*

// Input: A minimum spanning tree T with root r.
// Output: The medoid m.
var $\{D_i\}$ *// Sums of weighted distances for all images i*
$D_i \leftarrow 0$ for all images i in T
for all images i in T traversed breadth-first starting at r **do**
  // Step 1: Propagate root overlap *(Fig. 4.9a)*
  for all images j under i traversed breadth-first **do**
    Compute H_{ij}, x_{ij}, x_{ji} by recursive propagation (Eq. (4.9))
    $D_j \leftarrow D_j + \mathrm{d}_{ovl}(i,j)\varphi(\mathrm{d}_{ovl}(r,i))$ (Eq. (4.5))
    $D_i \leftarrow D_i + \mathrm{d}_{ovl}(i,j)\varphi(\mathrm{d}_{ovl}(r,j))$ (Eq. (4.5))
  end for
  // Step 2: Compute pairwise distances (Fig. 4.9b)
  for all image pairs (m,n) in different subtrees of i **do**
    Compute $\mathbf{x}_{nm}$ and $\mathbf{x}_{mn}$ by intersecting $\mathbf{x}_{ni}$ and $\mathbf{x}_{mi}$. (Fig. 4.9b)
    $D_m \leftarrow D_m + \mathrm{d}_{ovl}(m,n)\varphi(\mathrm{d}_{ovl}(r,n))$ (Eq. (4.5))
    $D_n \leftarrow D_n + \mathrm{d}_{ovl}(m,n)\varphi(\mathrm{d}_{ovl}(r,m))$ (Eq. (4.5))
  end for
end for
$m \leftarrow \arg\min_i\{D_i\}$

The parallel implementation of Iconoid Shift follows the same principle. However, because we only initialize the algorithm with a subset of the input images as seeds, we do not need to exhaustively compute the successor of each image. Instead, we initialize each Iconoid Shift worker with a seed image and compute its successor by performing an iteration of the Iconoid Shift algorithm. We then check if the successor has already been visited by another job. If not, we perform the next iteration with this image. If it has been visited already by another worker, we know that our current worker will follow the same path from now on and therefore initialize the current worker with another seed. We proceed in this way until the set of seeds is empty.

As illustrated in Figure 4.11, each run of Iconoid Shift follows a branch in a tree whose leaves are the seeds and whose roots are the Iconoids. Each worker follows a branch until it finds a node that has already been visited. This way, the tree is built up in parallel and each transition is only performed once.

We now present two speedups to increase the efficiency of Iconoid Shift further.

4.5.4 Memorizing Overlaps

We can avoid repeating retrieval and propagation steps in different iterations by re-using previous results. For this, we memorize the *corona* images, i.e., the images outside the support set that are adjacent to images within the support set (dashed boxes in Figure 4.8). After shifting the medoid, we re-build the minimum spanning tree re-using

previously determined homographies and overlap regions. Since some of the corona images will now enter the support set, we can use their known overlaps and homographies. Images that are no longer inside the support set but are still adjacent to an image inside the support set now become corona images. Finally, corona images no longer adjacent to an image inside the support set are removed. This way, the nodes known from the previous iteration can be processed at much lower cost, because we do not need to perform image retrieval and propagation of homographies and overlap regions. However, this method requires to store pairwise overlaps in memory, increasing the memory usage from linear to quadratic in the number of images under the kernel. In our implementation, we therefore did not make use of this idea, but we use it in Hierarchical Iconoid Shift (Ch. 5).

4.5.5 Basin of Attraction

A common speedup used in Mean Shift is to associate the *basin of attraction*, i.e., the points within a narrow radius around the mode, with the mode directly instead of performing an extra mode search for them. This can save computation time because a mode search from a point very close to a mode will likely converge to the same mode. This speedup can be used in Iconoid Shift by removing the basin-of-attraction images of each Iconoid from the seed set each time a new Iconoid is found.

4.6 Experimental Results

We now present our experimental results. We show that Iconoid Shift actually converges to views that suit our definition of iconic images and we compare them to the views selected by the valence criterion. By running it on the Paris 500k dataset (Sec. 3.3) and a dataset of 459k images from Barcelona, we demonstrate that Iconoid Shift can automatically discover meaningful object clusters in a fully unsupervised way in a large collection of photos.

4.6.1 Does Iconoid Shift actually select iconic views?

Our first question is whether our definition of Iconoids fits the intuitive concept of an iconic image. Figure 4.10 shows four typical runs of Iconoid Shift starting with views of landmark buildings taken at oblique angles or large distances (left column). Each run took three iterations to converge to an Iconoid (right column), which typically shows a frontal, centered and full view. Starting with a given view of a landmark building, Iconoid Shift tends to tilt, zoom and orbit around the object until it reaches a view that is favored by human photographers. As an interesting side effect, it automatically selects whether a portrait or landscape format photo fits the object better, because a photo that is completely filled by the object has higher mutual overlap with its neighbors.

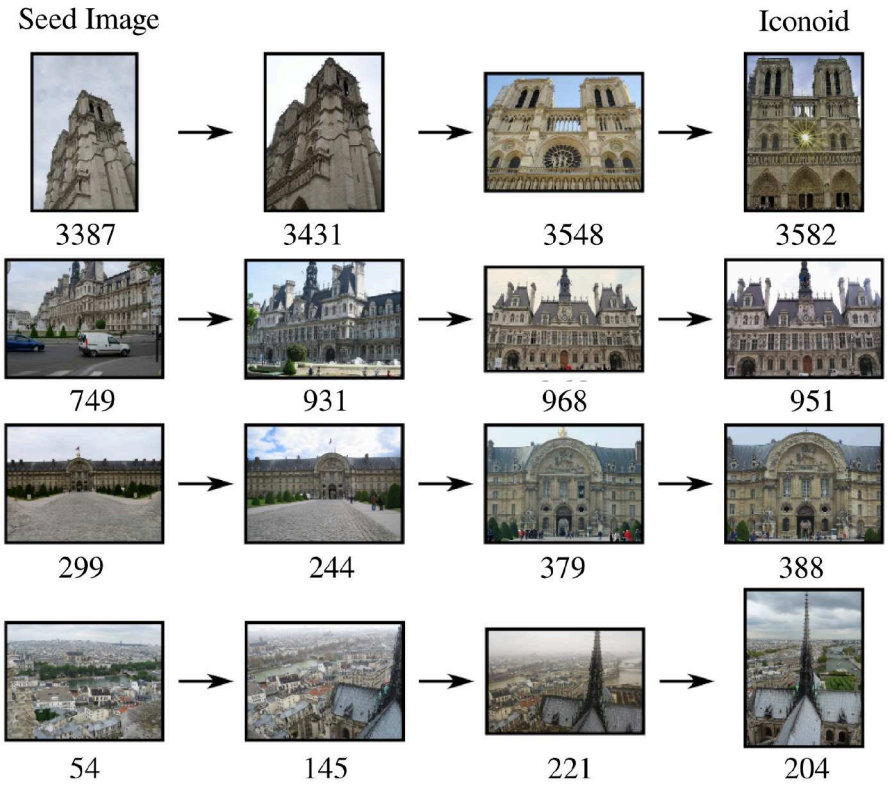
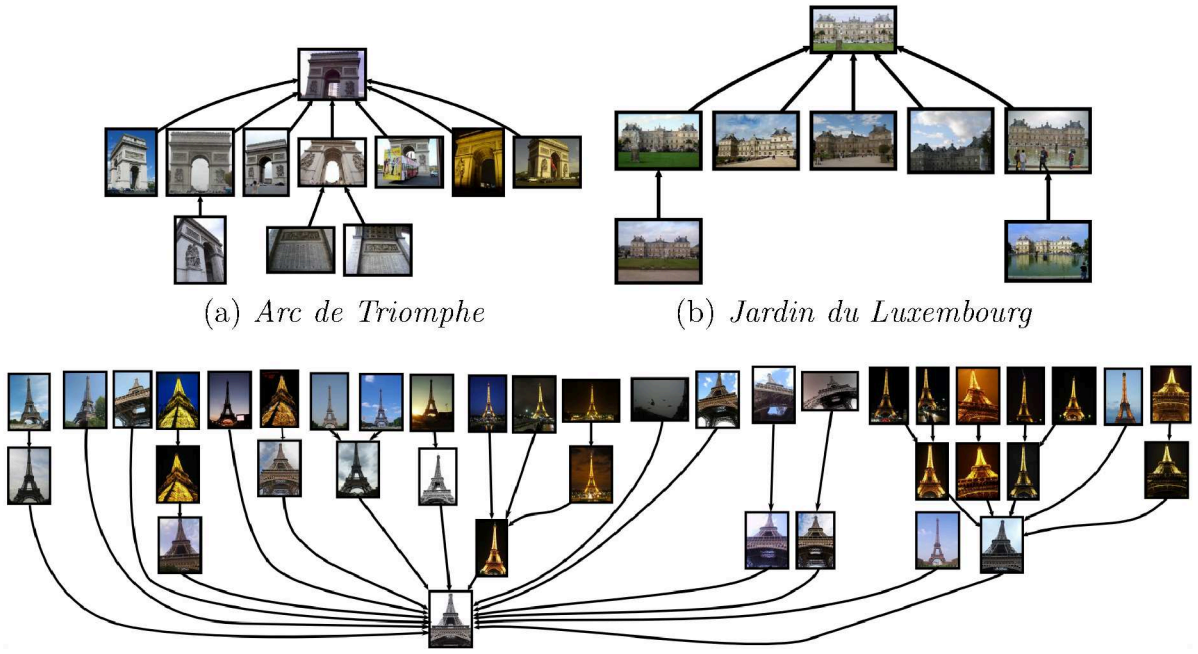


Figure 4.10: *Examples of Iconoid Shift sequences with the support set sizes at each step.*

The support set size (given below the images) is often higher for more "iconic" views, because the more typical a view is the more images overlap with it. However, since we optimize the mutual overlap and not the number of images in the support set, this number does not increase consistently.

Figure 4.11 shows a number of Iconoid Shift runs starting from different seed views (leaves) that each converge in the same Iconoid (root). Each path from a leaf to the root is a convergence path of Iconoid Shift.

For a comparison of the iconic images selected by Iconoid Shift with the images humans select to depict landmark buildings, we use the images in the Wikipedia article of Barcelona. Figure 4.12 shows that the views selected by our algorithm are generally very similar to those selected by the Wikipedia authors, which suggests that our maximum mutual overlap criterion corresponds to the human notion of iconic views. However, a noticeable systematic difference is that the weather is always better in the images from Wikipedia, which makes them more aesthetically pleasing. This could motivate a future extension of the algorithm that also takes into account features related to aesthetics, such as the color distribution.



(a) *Arc de Triomphe* (b) *Jardin du Luxembourg*

(c) *Eiffel Tower*

Figure 4.11: *Iconoid Shift runs converging in the same Iconoid.*

4.6.2 How does Iconoid Shift compare to the valence criterion?

A very popular approach for finding iconic images is the *valence criterion* (Sec. 2.1.2), which selects the images with the highest (weighted) valence in the matching graph (Crandall et al., 2009; Frahm et al., 2010; Jing et al., 2007; Kennedy and Naaman, 2008; Philbin et al., 2011; Raguram et al., 2011; Zheng et al., 2009). We compare our work with two approaches: Crandall et al. (2009) build the full matching graph, weigh each edge by the number of homography inliers, segment it using spectral clustering, and select the image with the highest *weighted* valence in each cluster. Philbin and Zisserman (2008) additionally merge the spectral clusters showing the same building by trying to propagate a homography between their images with maximum *unweighted* valence (Sec. 3.4). A qualitative comparison is given in Figure 4.13. In general, Iconoid Shift tends to select more central views than the weighted and unweighted valence criteria, which are based only on feature similarity and do not have a direct geometric interpretation. The numbers show the neighborhood sizes w.r.t. the respective neighborhood criterion (membership in the support set vs. adjacency in the matching graph). We define neighborhood using the geometric overlap that is propagated independently of feature matches and thus discover more images of the same object than direct feature-based matching.

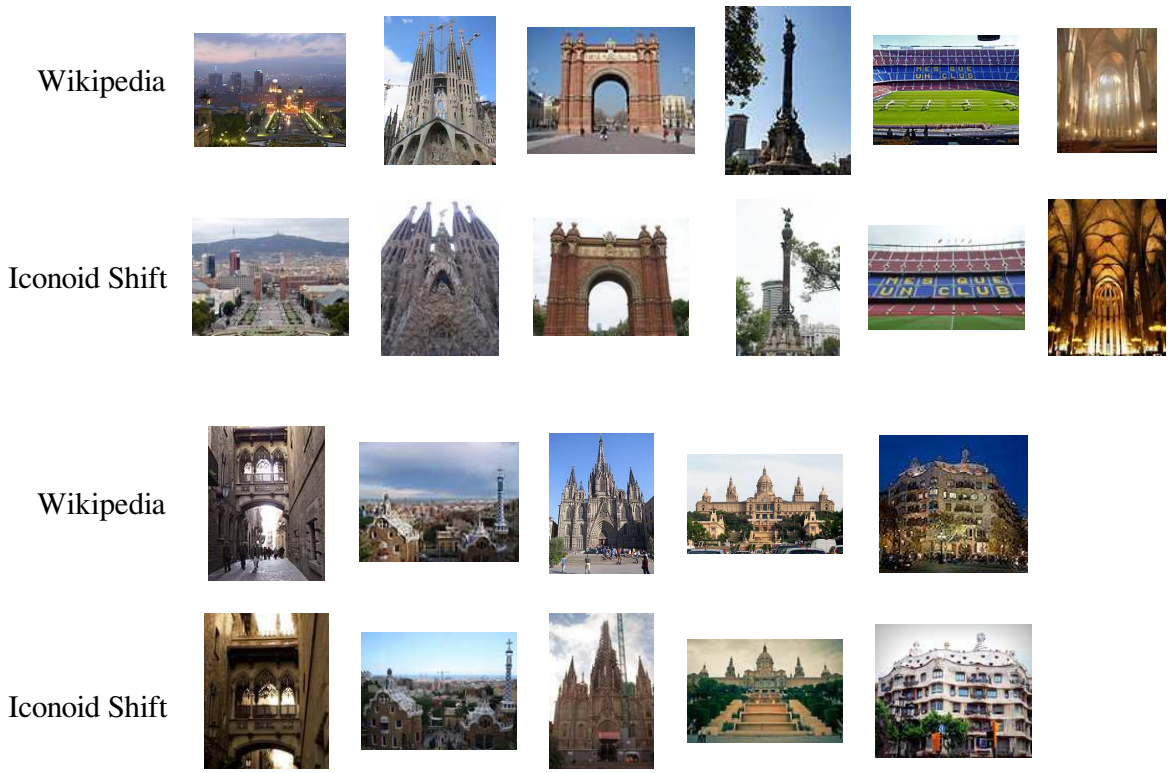


Figure 4.12: *Comparison of images used to depict popular landmarks of Barcelona from the Wikipedia article of Barcelona and the iconic images selected for these landmarks by Iconoid Shift.*

4.6.3 How are the Minimum Spanning Trees structured?

Iconoid Shift returns both the Iconoid and its minimum spanning tree (Fig. 4.14). Since this tree was constructed by recursive image retrieval, its branch structure reveals the structure of the Iconoid's neighborhood. For example, branches may contain specific views of the object or depict the object in certain lighting conditions. This has interesting applications such as navigating the scene of an Iconoid by following paths in the tree. Figure 4.15 shows a path where the perspective gradually moves from below the Eiffel Tower to a complete view of it.

4.6.4 Is the simple propagation scheme sufficient?

In order to verify that the transitive overlap propagation scheme (Fig. 4.5) is necessary to fully explore an Iconoid's neighborhood, we compare it to the simple scheme (Sec. 4.3.2) that multiplies homographies along the path and determines the inlying feature matches. We use a smaller dataset of 100k images of Paris and initialize Iconoid Shift with a set of 25 seed images generated by Geometric min-Hash. The simple scheme discovered 17

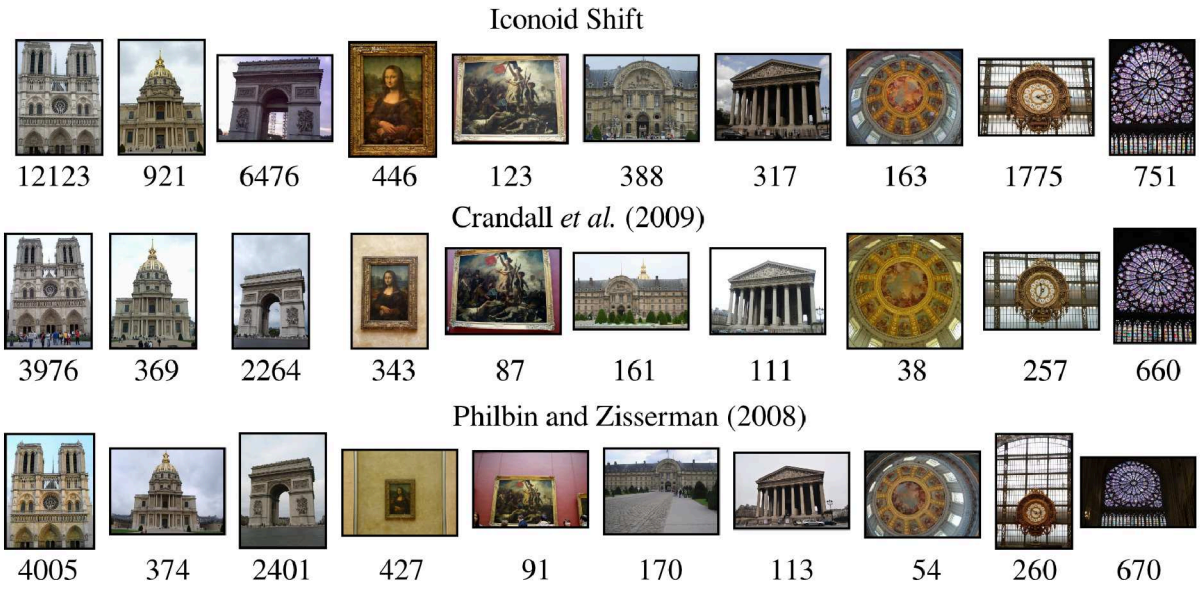


Figure 4.13: *Comparison of Iconoids (top) and the spectral clustering approaches of Crandall et al. (2009) and Philbin and Zisserman (2008) which select the images with maximum (weighted) valence. Numbers denote the number of images associated with the iconic by membership of the support set (top) and adjacency in the matching graph (middle and bottom).*

clusters with an average size of 137.9, while the transitive scheme discovered 16 clusters with an average size of 230.8. Visual inspection showed that in general, the images discovered by the simple method cover a lower variety of viewpoints, because it relies on direct feature matches and thus on the invariance of the interest point descriptor and detector, while the transitive scheme propagates the overlap region independently of direct feature matches. Furthermore, computation using the simple scheme took 24x longer than the transitive scheme, because direct overlap computations are more costly and require local features to be loaded from disk. We can therefore conclude that the transitive scheme is preferable both in terms of results and computation time.

4.6.5 Large-Scale Results

In order to show that Iconoid Shift can automatically identify the tourist hotspots of an entire city, we apply it to the full 500k images of Paris as well as to a dataset of 469k images from the inner city of Barcelona. Since we found that Geometric min-hash (Chum et al., 2007b) yields good starting points for growing clusters (Sec. 3.4), we use it to generate the set of seed images. We generate the seeds using min-Hash sketches of size 2. From the resulting set of colliding images, we remove duplicates by applying a tf-idf threshold and filter out multiple seeds of the same object by building a matching graph of the min-Hash seeds using image retrieval (Sec. 3.2.2), identifying its connected

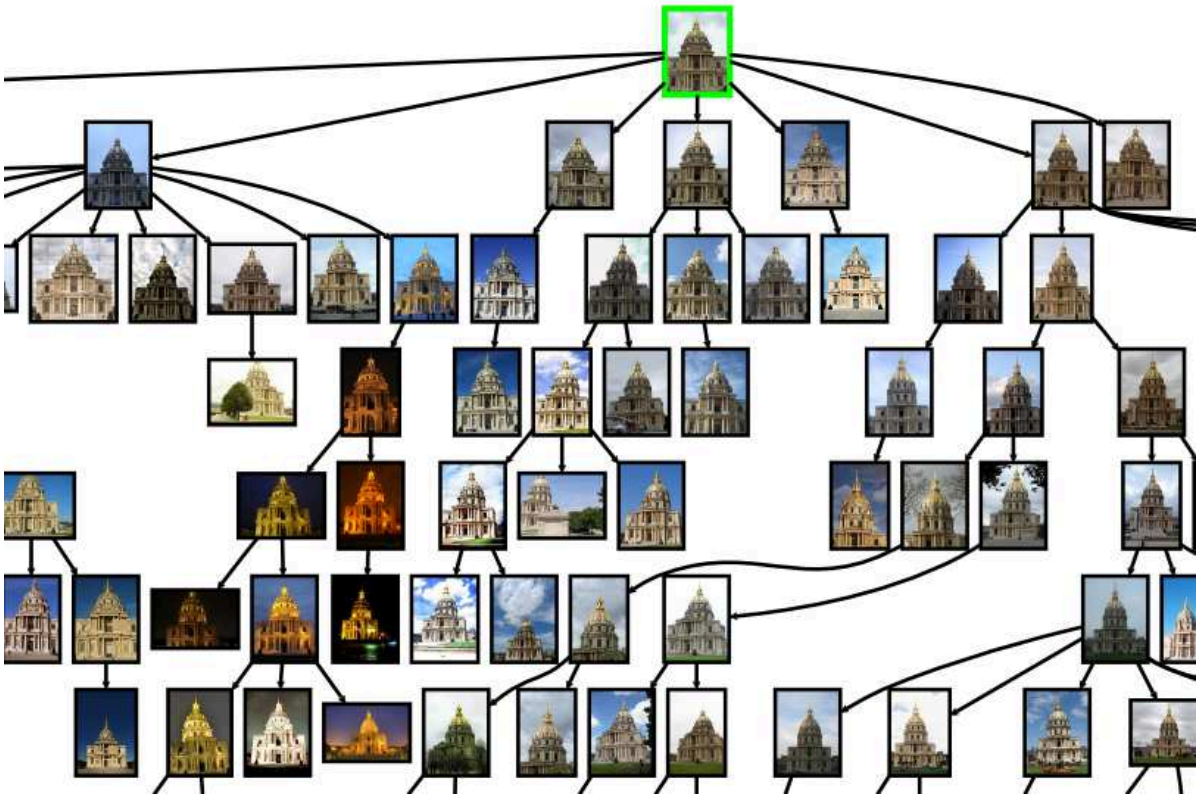


Figure 4.14: *Part of the minimum spanning tree of Les Invalides (921 images) with its Iconoid (green border) at the root.*

components and choosing one representative for each connected component. Since we only build such components for the seed set, this step is inexpensive. This reduced set of images is used to seed Iconoid Shift. We use the hinge kernel (Eq. (4.13)) and set $\beta = 0.9$, such that all images in the support set of an Iconoid need to have at least 10% overlap with it.

Statistics of the two experiments are shown in Table 4.1. By visual inspection we did not find any false positives in the support sets except for those caused by watermarks, timestamps, and frames that some users have added to their photos. As we show in Chapter 7, these artificial overlays can effectively be detected and the resulting matches can be discarded. On the Barcelona dataset, it was necessary to use a much larger number of Geometric min-hash sketches to obtain a number of seeds comparable to Paris. Interestingly, although they have similar numbers of Iconoids, the number of images covered by the Barcelona clustering is less than a third of the images covered by the clustering of Paris. This is because the most popular landmarks of Paris form larger clusters than the landmarks of Barcelona, as can be seen when comparing the number of Iconoids with 100 or more images as well as the mean and median cluster sizes.

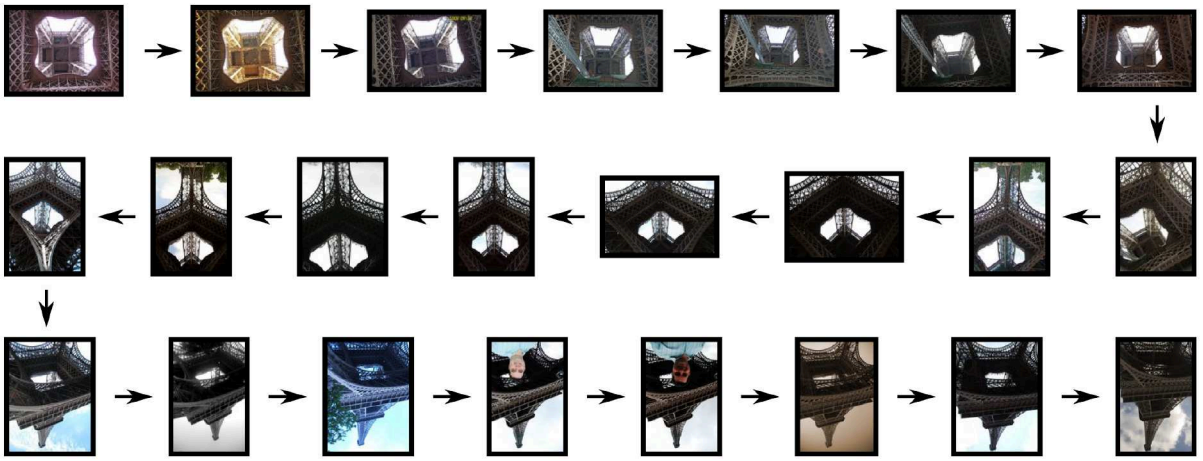


Figure 4.15: *Path in the local shortest path tree of the Eiffel Tower.*

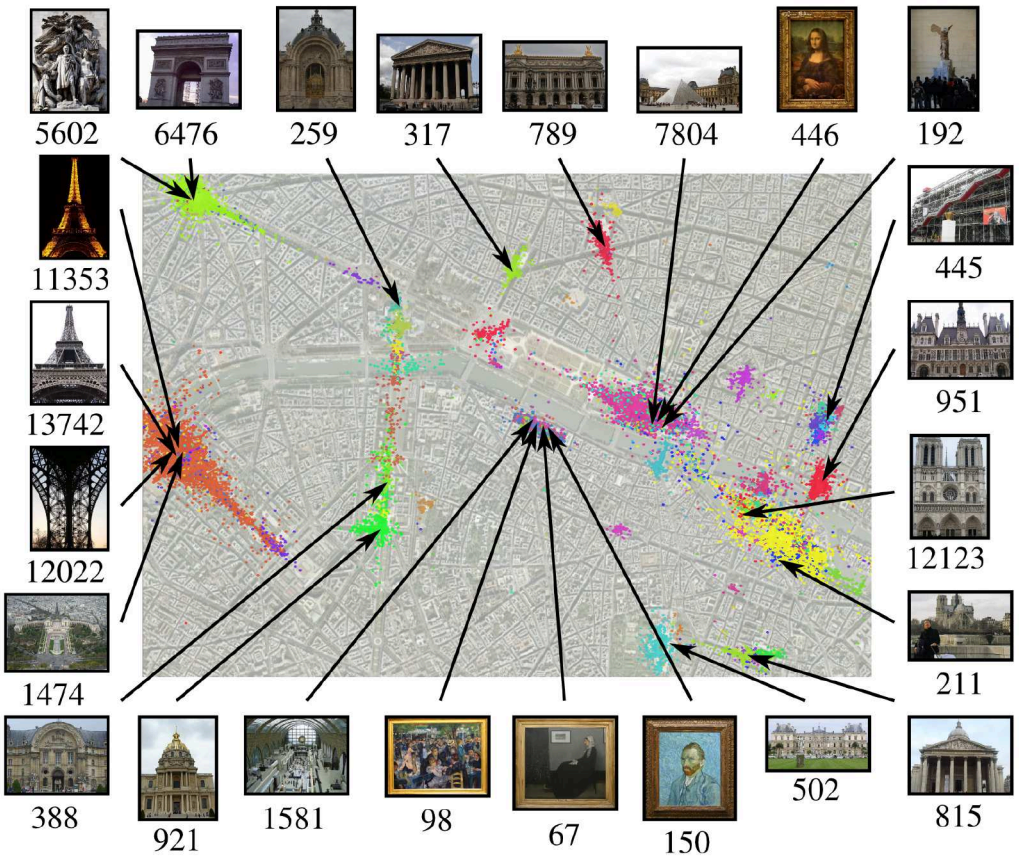


Figure 4.16: *Map of Iconoids and their cluster sizes discovered automatically by our algorithm in the* Paris 500k *dataset.*

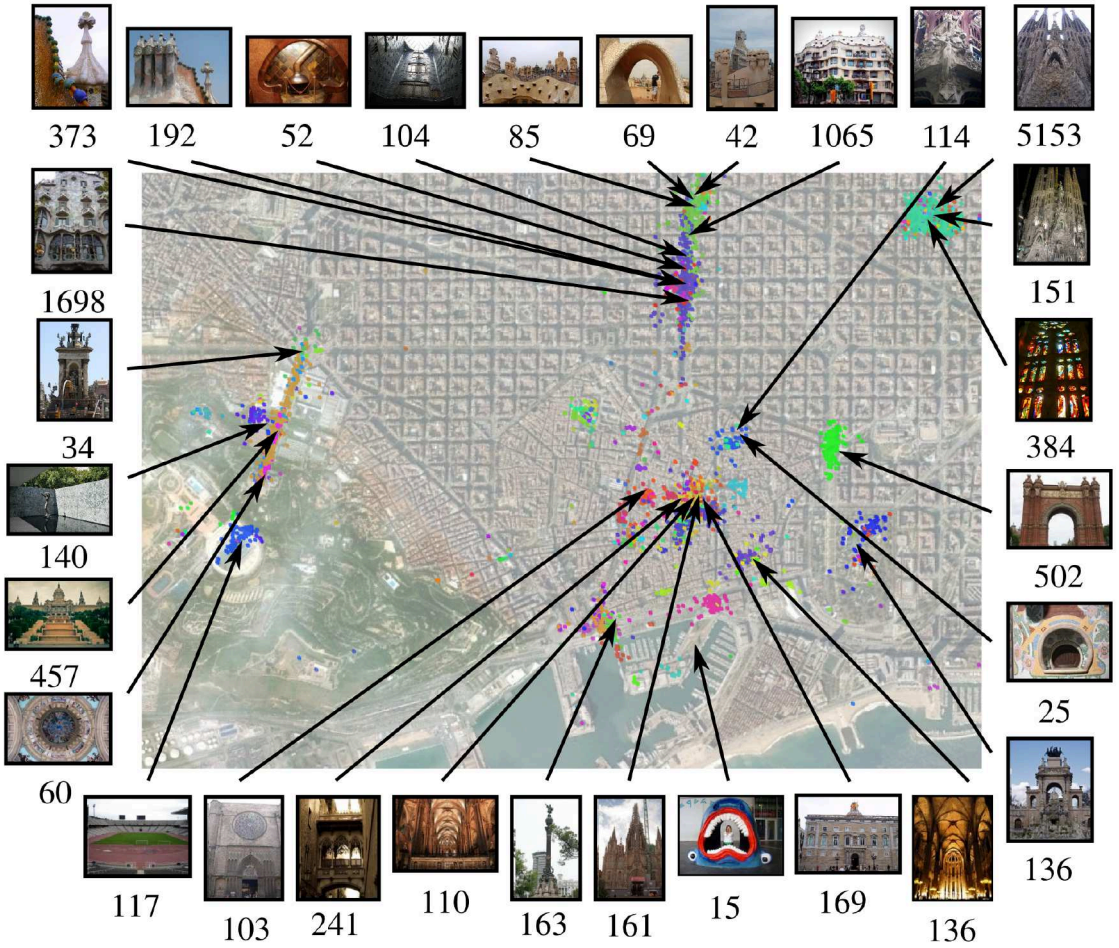


Figure 4.17: *Map of Iconoids and their cluster sizes discovered automatically by our algorithm in the Barcelona dataset.*

Figure 4.16 and Figure 4.17 show maps of the discovered Iconoid support sets and some example Iconoids of varying types and support set sizes. The top landmarks of Paris are the Eiffel Tower (7 Iconoids covering 16,342 images), Notre Dame (4 Iconoids covering 13,369 images) and the Arc the Triomphe (4 Iconoids covering 7,764 images). The top landmarks of Barcelona are Sagrada Familia (3 Iconoids, 5,584 images), Park Güell (2 Iconoids, 1,851 images) and Casa Battlo (1 Iconoids, 1,698 images). We perform a more detailed analysis of the clusters produced by Iconoid Shift and the objects it discovers in Chapter 6.

	Paris	Barcelona
Dataset size	501,356	458,980
min-hash sketches	5	120
Colliding images	10,487	17,213
Seeds	477	521
Iconoids	359	471
Iconoids ≥ 10	210	250
Iconoids ≥ 100	101	43
Images Covered	76,787	23,009
Mean Cluster Size	626.8	70.3
Median Cluster Size	18	10

Table 4.1: *Statistics of large-scale runs of Iconoid Shift on Paris and Barcelona. "Iconoids $\geq$ 10" is the number of Iconoids having 10 or more images in their support set. "Images Covered" is the total number of images in at least one Iconoid support set.*

4.6.6 Runtime analysis

The large-scale clustering of Paris 500k finished within a couple of hours on a small computing cluster with 10 nodes. On a single PC, the whole process would have taken 64h3m. Of these, 51h21m were required for image retrieval (inverted file queries and RANSAC verification). The remaining 11h42m are mostly due to bounding box transformations and clipping.

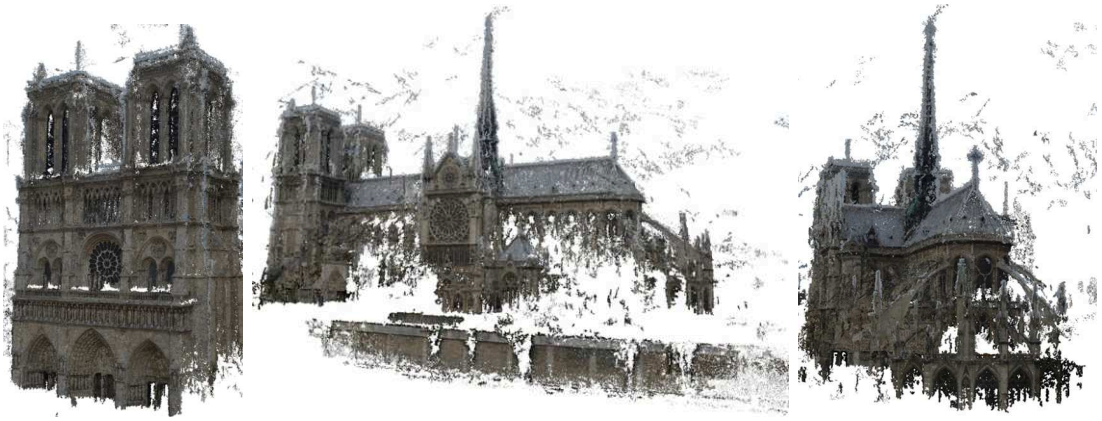
Note that the process can be sped up further by employing the basin of attraction speedup (Sec. 4.5.5). Finally, since we used a Matlab/Mex implementation, we expect further speedups from a pure C/C++ implementation.

4.6.7 Structure from Motion Reconstruction

The clusters produced by Iconoid Shift can be used to reconstruct buildings in 3D using structure-from-motion (Fig. 4.18). Because landmark buildings typically have several popular photo taking spots, Iconoid Shift often discovers multiple Iconoids of the same building or object. Since they overlap, clusters showing the same building can easily be identified (Fig. 4.19) and their union can be used as input for a structure-from-motion reconstruction.

Since structure-from-motion also reconstructs the camera positions and orientations of the input images, we can visualize the Iconoid positions in 3D to get a spatial impression of the favorite picture taking spots of a landmark (Fig. 4.20).

(a) *Arc de Triomphe*



(b) *Notre Dame*

Figure 4.18: *3D reconstructions from Iconoid support sets created using Bundler (Snavely et al., 2006) and PMVS (Furukawa and Ponce, 2009).*

4.7 Conclusion

In this chapter, we have presented the Iconoid Shift algorithm for discovering landmark buildings and other frequently photographed objects in large-scale image collections. We defined a distance measure for images based on their overlap that can be efficiently propagated through a graph. This propagation scheme also makes the distance measure more invariant to changes in viewing angle and lighting conditions than distances defined using a feature-based matching. Iconoid Shift performs mode search in the overlap space, where the modes, called Iconoids, correspond to the most popular views of a building. Clusters are formed by images within a certain distance radius to the mode. Intuitively, clusters are simply all images that have a certain minimum overlap with their Iconoid. Iconoid Shift is based is the well-studied Mean Shift (Fukunaga and Hostetler, 1975)

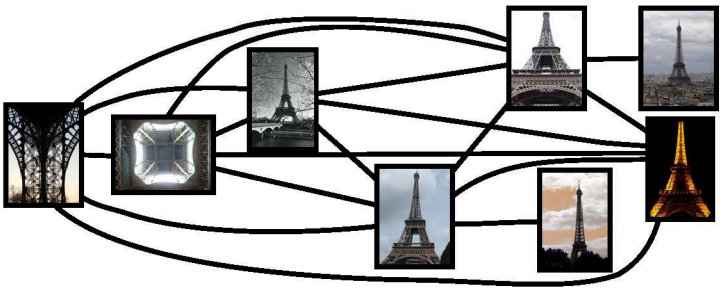


Figure 4.19: *Overlapping clusters of the Eiffel Tower. Each cluster is represented by its Iconoid. Two clusters are linked if they overlap.*

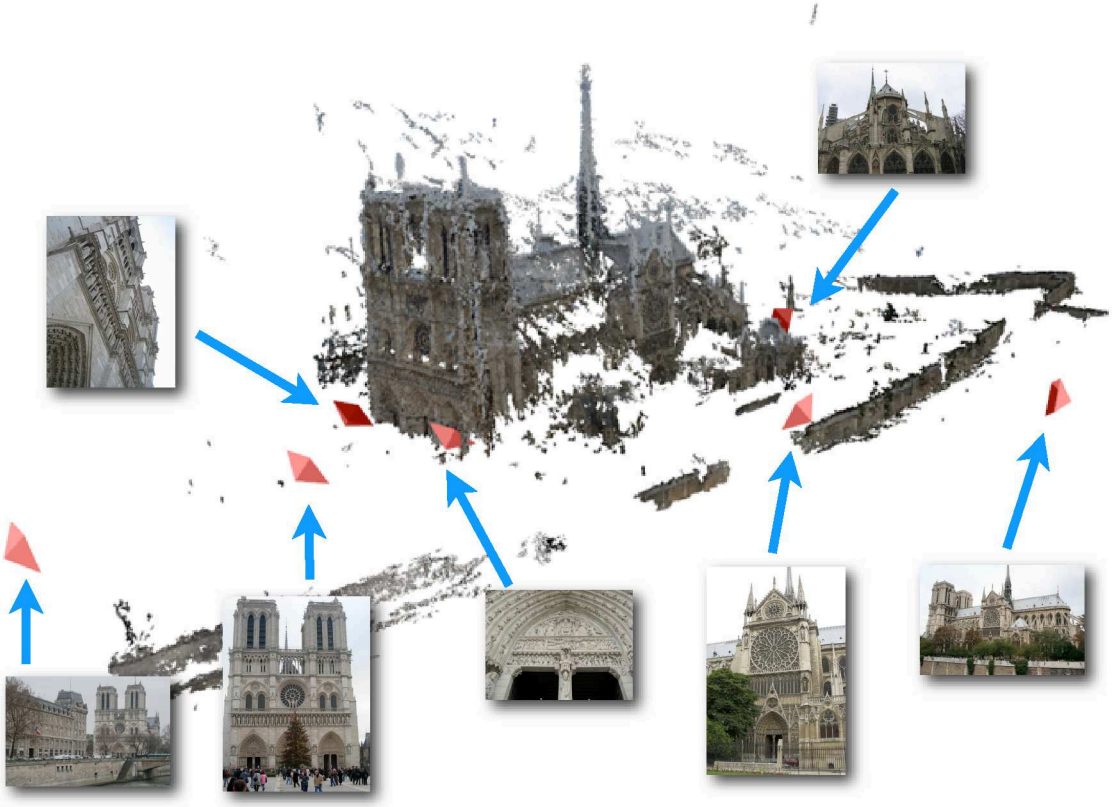


Figure 4.20: *3D reconstruction of the Iconioid camera positions for Notre Dame Cathedral. (Figure courtesy of Torsten Sattler)*

and Medoid Shift (Sheikh et al., 2007) algorithms and inherits their properties such as their intuitive interpretation of kernels and their proven convergence.

Given a seed image, Iconoid Shift performs a local exploration of the images within its kernel support, computes the pairwise overlaps of all image pairs and shifts the kernel window to the image with the highest overlap with all other images. We introduced an

efficient algorithm for performing the pairwise overlap computation through propagation. This algorithm is much faster than computing all overlaps based on homography inliers and has only linear memory complexity. Since Iconoid Shift instances that are initialized with different seeds are completely independent from one another, Iconoid Shift can trivially be applied in parallel on separate compute nodes, which makes it applicable to very large image sets.

Our experiments have shown that Iconoid Shift indeed converges to frontal and centered views, that are often a better depiction of the object than the views chosen by the popular valence criterion. We have applied Iconoid Shift on a large-scale dataset of 500k photos of Paris and have shown that it discovers a large number of popular objects, including all of the major landmarks of Paris, but also smaller objects such as statues or paintings. We give a detailed analysis of what objects Iconoid Shift discovers in Chapter 6 and show how the resulting clustering can be applied in a landmark recognition system.

We can think of Iconoid Shift as a self-correcting query expansion process. In their min-hash based clustering algorithm, Chum and Matas (2010) propose to use query expansion for exploring clusters. However, this approach tends to explore large components typically covering several buildings, as we found in Section 3.4. Instead, Iconoid Shift explores only a local neighborhood in which all images have a certain overlap with the Iconoid. The subsequent mode shift can be seen as a correction of the entry point in a direction that captures the depicted landmark better and that also typically has a larger neighborhood. This not only leads to higher quality clusters, but also saves the computational effort of exploring entire connected components of the matching graph.

A problem that Iconoid Shift inherits from Medoid Shift is the use of a fixed kernel bandwidth. This is an oversimplification of the problem because on realistic datasets different modes will have different scales. When choosing a too large bandwidth, we might miss smaller modes, since they are simply "blurred out" in the resulting kernel density. When choosing a too small bandwidth, the kernel density will have too many local maxima, simply caused by the uneven data distribution. We have used a relatively large kernel bandwidth for our experiments that was suitable for finding building-scale objects, but this likely caused us to miss many smaller objects. In the following chapter, we introduce a variant of Medoid Shift that is able to find clusters at all scales. We apply this algorithm in the Iconoid Shift framework we introduced in this chapter and show that it is capable of also discovering smaller objects like building details.

While our method of estimating the overlap between two images using bounding boxes around the inlier features is efficient, it is also prone to overestimating or underestimating the true overlap between the images due to low-textured areas, as we have shown in Figure 4.4d. To estimate the overlap more precisely, future directions could be using the convex hull around the inliers instead of bounding boxes, or using high photoconsistency regions or the commonly seen 3d points of a structure-from-motion reconstruction instead of homography inliers.

Another problem we observed in our experiments was caused by watermarks, timestamps or frames that have been added to the photos by the users that uploaded them. These artificial overlays can cause false-positive image matches that can cause unrelated objects to be joined into the same cluster. In Chapter 7, we therefore introduce a detector for such overlays that can be used to filter out such false-positive matches before performing the clustering.

5

Hierarchical Iconoid Shift: Discovering Details by Hierarchical Mode Search

A common problem of many landmark discovery algorithms, including Iconoid Shift, is that the user has to specify the granularity of the desired clustering in advance. For example, single-link clustering (Manning et al., 2008), which is used in, e.g., Gammeter et al. (2009); Quack et al. (2008); Zheng et al. (2009), requires the user to specify a linkage threshold. Kernel Vector Quantization (Tipping and Schölkopf, 2001), used in Avrithis et al. (2010), and Medoid Shift (Sheikh et al., 2007), which Iconoid Shift is based on, require the user to specify a kernel bandwidth. Not only is this choice far from trivial, but different objects may require different clustering granularities to be discovered. Because in most landmark discovery engines the granularity is optimized for building-level objects, they often miss many of the smaller, less popular objects.

Building details are a particularly challenging case, since they form smaller clusters within a larger building cluster. Figure 5.1 illustrates the problem on a toy example. The large cluster, Notre Dame Cathedral, has two sub-clusters corresponding to facade details. When applying Iconoid Shift with a too high kernel bandwidth (Fig. 5.1a), only the building itself will form a cluster, since the large kernel will "blur out" the details. Therefore, they will not form local maxima in the estimated kernel density distribution and will thus not be discovered by mode search. When using a too small bandwidth (Fig. 5.1b), the building and its details will each form clusters, but the building cluster will not contain all images of the building, because only images with a very high overlap with the Iconoid will be in its cluster. Moreover, it is not clear how clusters are related and there is no notion of the scale or importance of a cluster. Ideally, we would like a clustering that automatically picks the correct bandwidth for each mode and that produces a hierarchy describing which clusters are sub-structures of other clusters (Fig. 5.1c).

The dilemma of choosing the correct bandwidth does not only affect landmark discovery, but is a general problem of Mean Shift and Medoid Shift, because the scale of

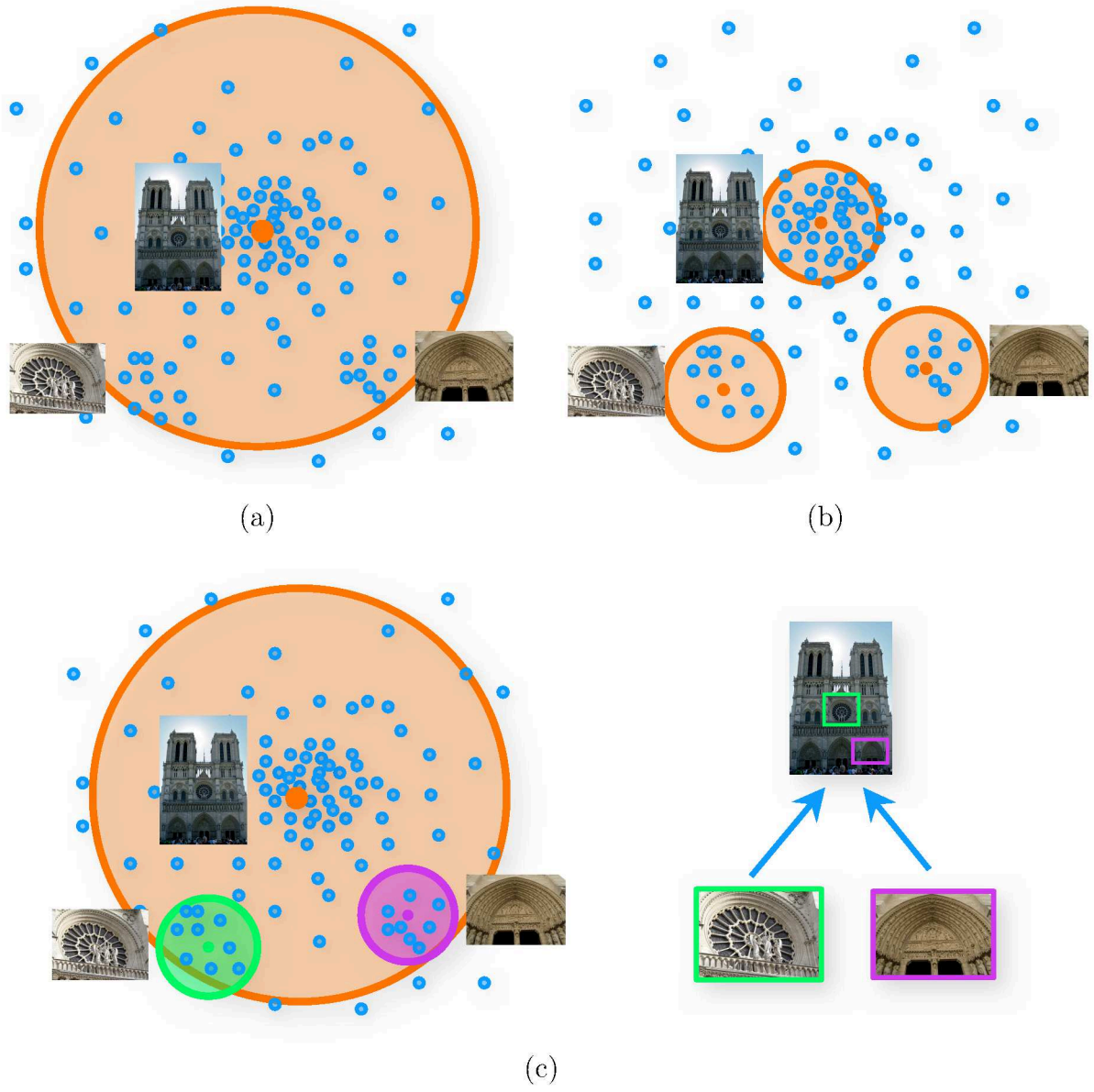


Figure 5.1: *The bandwidth dilemma. Blue dots represent images and orange circles represent the kernel support radii. (a) A clustering performed with a too high bandwidth will miss details. (b) A too low bandwidth will yield too small clusters for larger modes. (c) An ideal clustering would adapt the bandwidth to the data distribution and output clusters as a hierarchy.*

clusters is not always known in advance and can vary across the dataset. Consequently, hierarchical variants of Mean Shift have been proposed (Chakravarthy and Ghosh, 1996; Leung et al., 2000). These approaches run Medoid Shift several times, each time increas-

ing the kernel bandwidth by a fixed amount and using the modes resulting from each run as the seeds of the next run. If two seeds converge to the same mode, their clusters are merged. This way, a hierarchy of clusters is constructed. This approach, however, replaces the problem of how to choose the bandwidth with the problem of how to choose the bandwidth *increment*. If the bandwidth is incremented in too fine steps, there will be almost no change between subsequent iterations resulting in a waste of computing power. If the bandwidth increment is chosen too large, this approach might miss merge events and skip over the bandwidth that is suitable to discover certain clusters.

In this chapter, we introduce a hierarchical variant of Medoid Shift that does not require the choice of a fixed bandwidth increment, but instead increases the bandwidth in a *quasi-continuous* way, meaning the result is the same as if the bandwidth were increased continuously. Taking inspiration from Scale Space Filtering (Witkin, 1984), Hierarchical Medoid Shift follows the evolution of density maxima as the kernel bandwidth increases and builds a dendrogram from their merging behavior. This dendrogram serves as a hierarchical description of the input dataset. Each branch corresponds to a cluster and the bandwidth at which a branch merges into another branch determines the kernel bandwidth of its cluster. The algorithm is completely *parameter-free*, has the same complexity as Medoid Shift and is just as easy to parallelize.

We apply Hierarchical Medoid Shift to landmark clustering using the homography overlap distance introduced in Chapter 4. We will present an efficient propagation scheme to incrementally compute pairwise image overlaps as the kernel bandwidth increases and a parallelization scheme to perform this hierarchical clustering in a highly distributed manner. The resulting algorithm, called *Hierarchical Iconoid Shift* builds cluster hierarchies that describe landmarks and their details more accurately than the flat clustering that previous landmark clustering algorithms produced. Moreover, it has the same computational complexity and scalability as Iconoid Shift.

We evaluate this algorithm on a large-scale dataset of 802k Flickr photos of 36 historic landmarks and show that it produces detailed dendrograms that reflect the structure of the scene. Using Wikipedia as a source for landmark details, we show that our algorithm discovers many more details than a flat clustering at a fixed bandwidth.

The resulting landmark dendrograms can serve many interesting applications. Firstly, since Hierarchical Iconoid Shift discovers full views as well as details, it can be used to improve the number of objects a landmark recognition system can recognize. Since the relative positions of the parts w.r.t. the full building are known, it can also be used for building interactive tour guides or to provide a user of a landmark recognition system with much more details than current systems.

This chapter makes the following contributions:

- A scale-continuous *Hierarchical Medoid Shift* algorithm that has the same complexity as regular Medoid Shift, but, instead of a flat clustering, produces dendrograms reflecting the density structure across all scales. The algorithm is parameter-free and can easily be implemented in a distributed manner to scale to a large number of machines.
- We present a hierarchical landmark clustering algorithm called *Hierarchical Iconoid Shift*, based on the Iconoid Shift framework (Ch. 4), that applies Hierarchical Medoid Shift to landmark clustering in Internet photo collections.
- We propose to apply the resulting hierarchical landmark clustering to automatically creating summaries of touristic landmarks that could be used in mobile recognition apps. Moreover, we show how the hierarchies can be used to extend Wikipedia articles of landmark buildings with new details.

The remainder of this chapter is structured as follows: In the following section, we present related work. We then introduce the Hierarchical Medoid Shift algorithm in Section 5.2 and apply it to landmark clustering in the Hierarchical Iconoid Shift algorithm in Section 5.3. We show the results of applying this algorithm on a large-scale landmark dataset in Section 5.4.1, compare the results to Iconoid Shift and outline potential applications. Finally, we conclude the chapter and discuss future directions in Section 5.5.

5.1 Related Work

In landmark clustering, there are only few approaches with similar goals as Hierarchical Iconoid Shift, which we will discuss in the following. After that, we consider previous work in hierarchical clustering and hierarchical mode search in particular.

5.1.1 Landmark Clustering

Crandall et al. (2009) cluster photos from Internet photo collections by geo-location both at the city level and at the landmark level. For this, they perform geographical bucketing with different bucket sizes for the two scales and sub-sample the photos such that each bucket contains at most one photo from each user. Then, they find density modes by running Medoid Shift on the sub-sampled photos. The result is a multi-scale clustering, but not a hierarchical one, since the approach by Crandall et al. does not establish connections between clusters at different scales. In contrast to their work, we perform clustering based on photo content, not metadata, and we are interested in a much finer clustering granularity, relating buildings and details as opposed to cities and landmarks.

Epshtein et al. (2007) share our goal of creating a hierarchical summary of a scene. They first perform a structure-from-motion reconstruction and then create a 2D top-down view of the scene called the "geo-relevance image" in which the intensity of each pixel corresponds to the *relevance* of its scene position. This image is created by accumulating 2D projections of the view frustums of all the cameras in the scene such that the value of each pixel is the number of cameras whose view frustums overlap at its position. Then, the geo-relevance image is segmented hierarchically by applying Mean Shift in a top-down fashion. The first run of Mean Shift performs a coarse clustering, and recursive runs are performed that further segment each of the resulting clusters using a smaller kernel bandwidth. As for our approach, the result of their method is a tree of clusters describing the scene. However, since their approach operates on a 2D top-down view of the scene and uses discrete bandwidth increments for clustering, the detail granularity it can discover is limited. Furthermore, view frustums do not solely determine what a photo is actually showing, and two view frustums can intersect even though their images have no overlap which might cause some regions to be falsely considered relevant. Our approach directly uses image overlap as its grouping criterion and performs a scale-continuous hierarchical clustering.

Like Hierarchical Iconoid Shift, the "3D Wikipedia" system by Russell et al. (2013) produces a fine-grained description of a landmark building an its details. To achieve this goal, they perform web searches for all noun phrases in the Wikipedia article of the landmark and localize the retrieved images in its structure-from-motion reconstruction using pose estimation. The result is a browseable 3D model of the scene with bounding boxes around prominent details that have hyperlinks to the paragraph in Wikipedia that describes them. This approach however relies on Wikipedia, which in our experience often provides only an incomplete list of details of a landmark. Also, this approach does not produce a hierarchy, but only a set of links between an article and a 3D model.

While we take a bottom-up approach to detail discovery, Mikulik et al. (2014) discover details in a top-down fashion. Starting with the full view of a facade, their Hierarchical Query Expansion approach mines increasingly detailed images of the building.

5.1.2 Hierarchical Clustering

One of the most popular hierarchical clustering algorithms is Hierarchical Agglomerative Clustering (Manning et al., 2008) that iteratively merges the two closest clusters and thus builds a dendrogram with each data point as a leaf and a cluster containing the whole dataset as the root. The landmark clustering approaches that use this method (Gammeter et al., 2009, 2010; Quack et al., 2008; Zheng et al., 2009) ignore the resulting hierarchy and instead simply use the clustering that results after stopping at a predetermined cluster similarity threshold. Apart from its high computational complexity (cubic in general, quadratic when using the single-linkage criterion), a disadvantage of Hierarchical Agglomerative Clustering is that clusters do not have an intuitive interpretation in the context of object discovery. For example, the clusters of Single-Link

clustering are actually the connected components of the matching graph when removing all edges whose weight is below the linkage threshold. As we have shown in Section 3.4, connected components are not necessarily limited to single objects, but in practice often cover entire city areas. The reason for this is that two objects will already be in the same cluster if at least one image exists that shows them together. Therefore, clusters do not necessarily have a single "subject". In contrast, when using a *density-based* method like Iconoid Shift, each cluster has a mode that serves as a representative for it and defines its subject. Several approaches for performing a *density-based hierarchical* clustering have been proposed: Paris and Durand (2007) create a hierarchical image segmentation by first performing Mean Shift clustering using a *single fixed scale*, and then successively merging the resulting clusters using a criterion derived from Morse theory. Yuan et al. (2012) propose a hierarchical agglomerative version of Mean Shift that works by iteratively summarizing points that are assumed to converge to the same mode. Both algorithms only operate in Euclidean space and, despite being hierarchical, only operate on one fixed kernel bandwidth.

The key inspiration for our work is the *Scale Space Filtering* algorithm by Witkin (1984). It builds the *Scale Space* of a 1D signal by blurring it with a continuously growing Gaussian, then tracks inflection points and builds a dendrogram from their merging behavior. Leung et al. (2000) apply this idea to 2D images by tracking density maxima with a procedure similar to Mean Shift while increasing the kernel bandwidth in discrete steps. Given a set of bandwidth steps $\beta_1, \ldots, \beta_N$, their approach iteratively performs mode search, using the modes of the run at β_i as seeds for the run at β_{i+1}. Chakravarthy and Ghosh (1996) propose a similar approach that estimates the density using a Radial Basis Function network. Their approach is applicable to higher-dimensional Euclidean spaces, but also still requires fixed bandwidth steps. Later works (DeMenthon, 2002; Vatturi and Wong, 2009) applied this approach of alternately performing mode search and incrementing the bandwidth to video segmentation and category detection, respectively. However, all of these works are not applicable to non-Euclidean metric spaces, which severely limits the image distance measures that can be used to apply them for object clustering. Moreover, they all require the choice of fixed bandwidth steps. However, choosing these bandwidth steps is far from trivial. If the intervals between bandwidths are chosen too coarse, some cluster merging events might be missed. If the intervals are chosen too fine, many unnecessary mode searches will be performed, increasing the computational effort unnecessarily. In fact, different parts of the dataset might require different bandwidth increments.

5.1.3 Discussion

In contrast to the above approaches, Hierarchical Medoid Shift increases the bandwidth in a quasi-continuous way. It chooses the bandwidth increment dynamically and locally based on the data instead of forcing the user to decide on a fixed increment. Finally, because it relies on Medoid Shift, it works in arbitrary metric spaces.

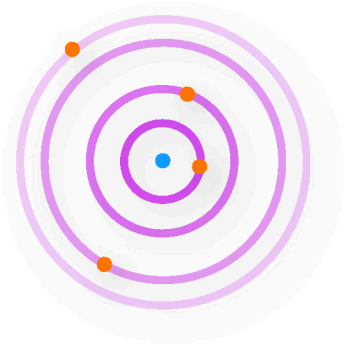


Figure 5.2: *The bandwidths at which data points enter the growing kernel window are their distances to the medoid.*

5.2 Hierarchical Medoid Shift

We now introduce the *Hierarchical Medoid Shift* (HMS) algorithm that is based on Medoid Shift, but produces a hierarchical clustering by continuously growing the kernel window and tracking the merging behavior of density maxima. Like Medoid Shift, HMS is a general clustering algorithm that can likely be useful for many different clustering tasks. In this chapter, we focus on its application to clustering Internet photo collections and leave other applications as future work.

5.2.1 Quasi-continuous Kernel Growth

In the following, we will assume that the kernel profile is a *step function* (Eq. (4.13)), which is a common choice in practical applications of Mean Shift and Medoid Shift. This allows us to make the following observations:

1. Since the kernel profile is piecewise constant, the weights of the points under the kernel will not change when its bandwidth is increased. Therefore, the weighted distance sums in the Medoid Shift minimization (Eq. (4.5)) will *only* change when a new point enters the kernel window.

2. Because the kernel has finite support and the dataset is finite, there is only a finite number of bandwidth steps at which a new data point enters the kernel window (Fig. 5.2). These steps are the distances of the points in the dataset to the medoid.

These observations enable *continuously* growing the kernel by examining only a *finite* number of bandwidth steps. Because the medoid cannot change unless a new data point enters the kernel window, the resulting clustering is the *same* as if the kernel bandwidth were grown continuously.

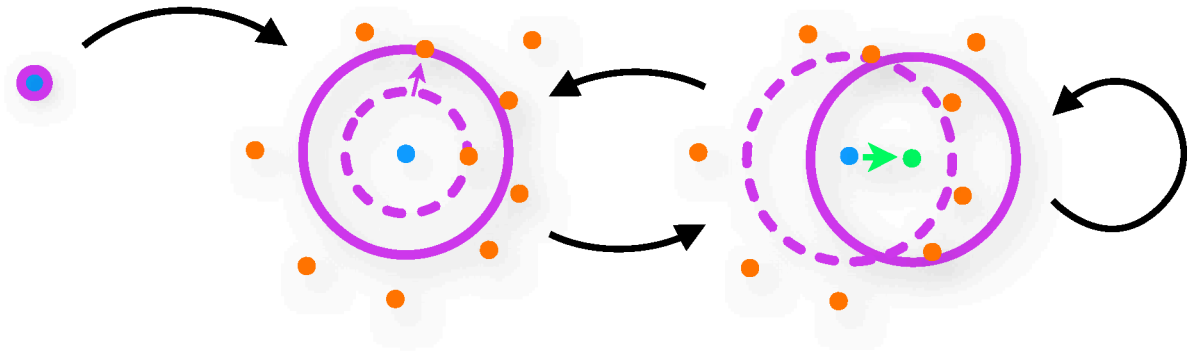


Figure 5.3: *Hierarchical Medoid Shift. After being initialized with a seed (left), HMS iterates between two steps: Increasing the kernel bandwidth until a new point enters the kernel window (middle), and seeking the medoid using Medoid Shift (right).*

Algorithm 3 *Hierarchical Medoid Shift.*

// Input: Point set $\{\mathbf{x}\}$*, seed point* $\mathbf{y}_0$
// Output: Sequence of modes and bandwidths $S = ((\beta_k, \mathbf{y}_k))$
$k := 0$, $\beta_0 := 0$, $S = ((\beta_0, \mathbf{y}_0))$
while $\exists \mathbf{x} : \mathrm{d}(\mathbf{x}, \mathbf{y}_k) > \beta_k$ **do**
  $\beta_{k+1} := \min_{\mathbf{x}:\mathrm{d}(\mathbf{x},\mathbf{y}_k)>\beta_k} \{\mathrm{d}(\mathbf{x}, \mathbf{y}_k)\}$ *// Grow kernel*
  $k := k + 1$
  // Perform Medoid Shift
  $\hat{\mathbf{y}} := \emptyset$
  while $\hat{\mathbf{y}} \neq \mathbf{y}_k$ **do**
    $\hat{\mathbf{y}} := \mathbf{y}_k$
    $\mathbf{y}_k = \arg\min_{\mathbf{y}\in\{\mathbf{x}\}} \{\sum_{\mathbf{x}} \mathrm{d}(\mathbf{y}, \mathbf{x}) \varphi_{\beta_k}(\mathrm{d}(\mathbf{y}_k, \mathbf{x}))\}$ (Eq. (4.5))
  end while
  Append $(\beta_k, \mathbf{y}_k)$ to S
end while

5.2.2 The Hierarchical Medoid Shift Algorithm

We now formulate the Hierarchical Medoid Shift algorithm that grows the kernel bandwidth in a quasi-continuous way (Fig. 5.3): We start from a seed point at kernel bandwidth $\beta = 0$ and build a priority queue of its nearest neighbors, ordered by their distances to the seed. These distances define the discrete steps in which the kernel window grows. In each step, we pop an element from the priority queue, increase the kernel bandwidth to its distance from the seed and check if the current kernel center is still the medoid. If not, we find the new medoid by performing Medoid Shift iterations (Eq. (4.5)) until convergence and re-build the priority queue. Then, we continue with the next growing operation. We repeat this procedure until the queue is empty, i.e., all data points are inside the kernel window. The detailed algorithm is given in Alg. 3.

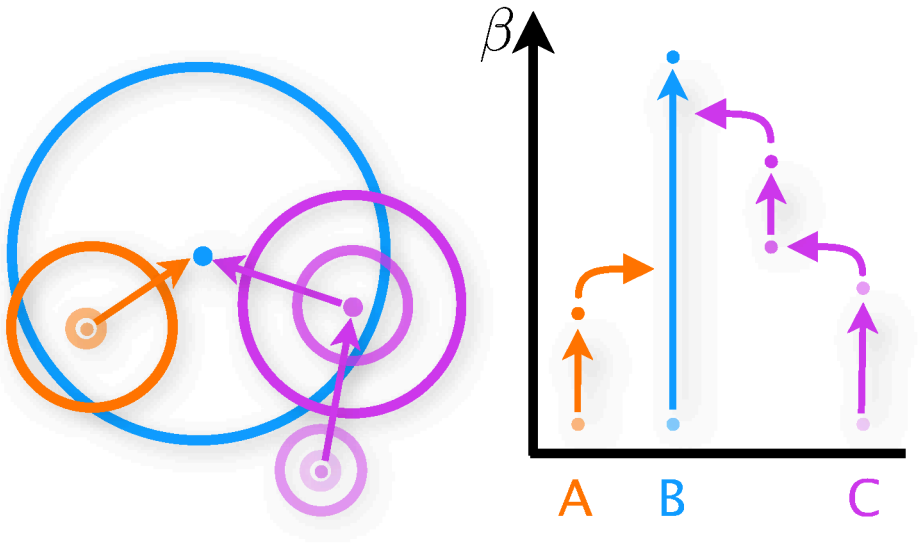


Figure 5.4: *Relationship between clusters and dendrograms in Hierarchical Medoid Shift.*

Every time a point enters the kernel, we compute its distances to all points under the kernel support. Thus, in total, the number of distance computations performed is quadratic in the number of points under the final kernel window. The computational complexity is therefore the same as for standard Medoid Shift. However, in contrast to Medoid Shift, HMS is completely parameter-free.

The algorithm is initialized once with every data point. Similar to the evolution of local maxima in scale space (Witkin, 1984), medoids corresponding to small maxima will merge to form larger maxima (Fig. 5.4). The resulting convergence sequences therefore form a dendrogram of the density structure of the dataset at all scales. A horizontal slice through this dendrogram yields the set of medoids at a particular scale.

5.2.3 Two Clustering Definitions

In Mean Shift (Cheng, 1995) and Medoid Shift (Sheikh et al., 2007), a cluster is the set of all points that converge in the same mode. This results in a *hard* clustering, since every point is assigned to one cluster. For Iconoid Shift (Ch. 4), we used a different cluster definition that is more suitable to our application. Here, a cluster consists of all points under the kernel support of a mode. This definition yields an *overlapping* clustering. For Hierarchical Medoid Shift, both hard and overlapping cluster definitions are possible as well. In both definitions, each point forms its own cluster at the lowest scale. If we use the *hard* cluster definition, then every merging of two branches in the dendrogram will simply cause the clusters of the two merging branches to be joined. The resulting cluster hierarchy will therefore be *nested*, since clusters at lower bandwidths are always subsets of clusters at higher bandwidths. In the *overlapping* cluster definition, the resulting

cluster hierarchy is *non-nested*, since kernel windows can shift when the bandwidth is increased. When the kernel window shifts, points that have been within the kernel radius of a mode at a lower bandwidth might not be in its radius at a higher bandwidth. In this definition, at each kernel bandwidth, the set of modes defines an *overlapping* clustering. In summary, we have the choice between a *hard*, *nested* clustering and an *overlapping*, *non-nested* clustering. Like in Iconoid Shift, we use the overlapping cluster definition for Hierarchical Iconoid Shift, but there might be other applications of HMS where the hard clustering definition is more suitable.

5.2.4 Parallelization

HMS can be parallelized to run on multiple machines by initializing each worker with one seed. We then run HMS independently on each worker. We can imagine each run of the algorithm as tracing out a branch of the dendrogram from bottom to top, where we move upwards each time we increase the kernel bandwidth β and we move horizontally each time we shift (Fig. 5.4). We keep track of the dendrogram branches in a central data structure. There is no need to keep the bandwidths of concurrent runs synchronized. We only need to check for collisions with existing branches before growing or shifting operations. As soon as the current branch collides with another branch, we know that it will follow the same path. Therefore, we stop its corresponding worker and re-initialize it with a new seed.

5.2.5 Computational Complexity

We now show that both Medoid Shift and HMS have a *worst-case* complexity of $O(N^3)$ in the dataset size N. In the following, we assume the pairwise distances of all data points are already known. If not, they can be precomputed in $O(N^2)$ time, which would not dominate the runtime of the algorithm.

Medoid Shift. In practice, Medoid Shift is not run by initializing it with a seed and performing iterations until it converges at a Medoid. Instead, its implementation exploits the fact that the algorithm can only have a finite number of *states* because the kernel always has to be centered on a data point. Medoid Shift therefore exhaustively computes for each point in the dataset the next point to shift to. (In Mean Shift (Cheng, 1995), it would not be possible to exhaustively pre-compute the shifts, because the kernel center can lie at any position in space.) The next point is found by the Medoid Shift minimization (Eq. (4.5)). The result of this is a directed graph in which each point has an outgoing edge to the next point, except for the medoids, which are the convergence points of the algorithm. The minimization requires at most $O(N^2)$ operations and is done for each of the N data points. Hence, the shift computation step requires at most $O(N^3)$ operations for the whole dataset. Then, each point is associated with a mode by following its path in the directed graph until reaching a mode. This step takes at most

$O(N^2)$ operations. Because the computation time of the algorithm is dominated by the shift computation, its overall complexity is $O(N^3)$.

Hierarchical Medoid Shift. Although it produces a hierarchical clustering instead of a flat one, HMS also has a worst-case complexity of $O(N^3)$. We show that HMS visits at most $O(N^2)$ states and that the worst-case computation time spent in each state is in $O(N)$. For the latter, we introduce an incremental scheme that allows HMS to perform the Medoid Shift minimization (Eq. (4.5)) based on memorized partial results from lower bandwidths.

Given the current data point $\mathbf{y}_k$ and current bandwidth β_n, HMS computes a set of *weighted distance sums* $S^k_{\mathbf{y},n}$ for each point $\mathbf{y}$ under the kernel window of the current data point:

$$S^k_{\mathbf{y},n} = \sum_i \mathrm{d}(\mathbf{y}, \mathbf{x}_i)\varphi(\mathrm{d}(\mathbf{y}_k, \mathbf{x}_i)). \tag{5.1}$$

Instead of computing this sum explicitly in each step, we compute it incrementally each time the kernel bandwidth is increased. When we pop the next closest point $\mathbf{x}_m$ from the priority queue, we increase the kernel bandwidth to the distance of this new point from the current data point $\mathbf{y}_k$. Then, we update the distance sum $S^k_{\mathbf{y},n}$ of each data point $\mathbf{y}$ under the kernel window of $\mathbf{y}_k$ by adding to it the distance $\mathrm{d}(\mathbf{y}, \mathbf{x}_m)$ between $\mathbf{y}$ and the new point $\mathbf{x}_m$, which is weighted by the kernel window:

$$S^k_{\mathbf{y},n} := S^k_{\mathbf{y},n-1} + \mathrm{d}(\mathbf{y}, \mathbf{x}_m)\varphi(\mathrm{d}(\mathbf{y}_k, \mathbf{x}_m)). \tag{5.2}$$

We then determine the new medoid $\mathbf{y}_{k+1}$ by finding the point with the lowest distance sum:

$$\mathbf{y}_{k+1} = \arg\min_{\mathbf{y}} \left\{ S^k_{\mathbf{y},n} \right\}. \tag{5.3}$$

This is equivalent to the original Medoid Shift minimization (Eq. (4.5)), but only requires us to perform $O(N)$ additions at each bandwidth step.

Since there are N data points, and for each data point, there are N possible bandwidths (given by the distances of the current point to all other points), there is a total of $O(N^2)$ possible states of HMS. A toy example for the set of HMS states is shown in Fig. 5.5. When running HMS in practice, only a subset of these states is actually visited, because smaller modes merge into larger modes as the bandwidth increases.

Having shown that there are $O(N^2)$ possible states and that incremental computation of distance sums requires $O(N)$ steps in each state, we still need to show that no more than $O(N)$ computations are performed in each state. We assume that HMS is run linearly, i.e., that it is initialized with one seed at a time. As soon as we either reach the maximum kernel bandwidth or merge into an already processed state, we initialize it with the next seed. In any state, we decide whether to *shift* or *grow* by checking if the current medoid still has the lowest distance sum (Eq. (5.3)). This check has $O(N)$ complexity. If we decide to *grow*, we add one point to the kernel window and update the distance sums as above, which has $O(N)$ complexity, as explained above. If we decide to *shift*, there are two possible cases.

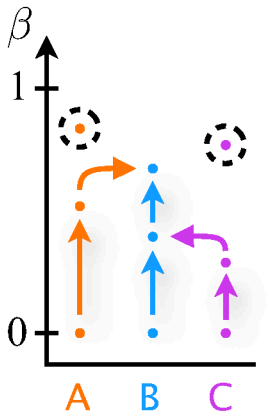


Figure 5.5: *The set of states of HMS for a dataset of 3 points. Each point has three states (denoted by the colored dots above it): One at $\beta = 0$ and two at the distances of the other two data points. Arrows indicate state transitions. The states marked by dashed circles are never visited, because A and C merge into B before reaching them.*

1. We transition to a state that has already been visited in a previous run. In this case, we do not need to do anything else in this state and can continue with the next seed.

2. We transition to a state that has not yet been visited. To compute its weighted distance sums incrementally, we would need the weighted distance sums at the next lower bandwidth for the current point. However, this bandwidth might not yet have been visited either. We therefore compute the weighted distance sums incrementally, starting at the lowest bandwidth for the current data point for which weighted distance sums exist, or at $\beta = 0$ if the point has not yet been visited at all. This, however, would require $O(N^2)$ operations, and we only want to spend $O(N)$ operations per state. But since we memorize the weighted distance sums we compute for each state along the way, we do not need to perform these computation steps in future runs. So, we effectively only change the *order* in which we compute the weighted distance sums. For example, if we need to perform the incremental computation for k bandwidths below the current one, we do not need to perform this computation in k future states, meaning only the incremental computation from the next lower bandwidth to the current state needs to be counted in this step. Therefore, shifting operations effectively require $O(N)$ computation time.

In summary, since HMS visits at most $O(N^2)$ states and in each state it performs at most $O(N)$ operations, the worst-case complexity of applying HMS on a dataset of N points is therefore $O(N^3)$.

5.3 Hierarchical Iconoid Shift

We now describe the Hierarchical Iconoid Shift (HIS) algorithm that applies Hierarchical Medoid Shift to produce a hierarchical clustering of large amounts of photos from Internet photo collections. For this, we present efficient schemes for growing the kernel bandwidth and shifting the mode that use the HOP algorithm (Sec. 4.5.2) to propagate overlaps to new images, re-using previously computed overlaps where possible.

Recall that the *corona* (Sec. 4.5.4) is the set of images whose overlap distance to the medoid is greater than the kernel bandwidth β, but that match at least one image from the support set (i.e. the dashed images in Figure 5.6). The overlaps of the corona images with the medoid define the discrete scale steps for growing the kernel. Upon being discovered by querying an image retrieval system (Sec. 3.2.2), the corona images are inserted into a priority queue and prioritized by their overlap with the medoid. HIS maintains a minimum spanning tree (MST) of all images in the corona and support set. New images are added to this tree incrementally using the linear time algorithm by Chin and Houck (1978).

Figure 5.6 shows the steps of the HIS algorithm. We start with a single seed image in the support set and an initial kernel bandwidth $\beta = 0$. Then, we query an image retrieval engine with the seed and add its matching images to the corona. We compute the overlaps of the new corona images with the seed and add the images with non-empty overlap to the priority queue. At the start of each iteration, we first take the top image from the priority queue and increase the kernel bandwidth to the overlap distance of this image from the medoid. Then, we compute the image's pairwise distances to all images in the support set using HOP (cf. Sec. 4.5.2) in the MST. Next, we complete the corona by retrieving images matching the newly added image, computing overlaps with the medoid using HOP, and inserting them into the priority queue. Finally, we check whether the medoid has shifted by performing the Medoid Shift minimization (Eq. (4.5)). If not, we directly continue with the next growing step. Otherwise, we perform Iconoid Shift iterations until convergence. After converging to a mode, we continue growing the kernel. The HIS algorithm is finished when the corona is empty, i.e., all images overlapping with the medoid are in the support set.

In the Iconoid Shift iterations, we re-use previously computed overlaps as described in Section 4.5.4 to save computation time (Fig. 5.6, left). The support set update is done in three steps: 1. We move support set images that are no longer within the kernel radius to the corona. 2. We remove corona images with no match in the support set. 3. While there are images in the corona whose overlap distance to the medoid is smaller than β, we add them to the support set, compute their overlaps with all support set images and retrieve new corona images as above.

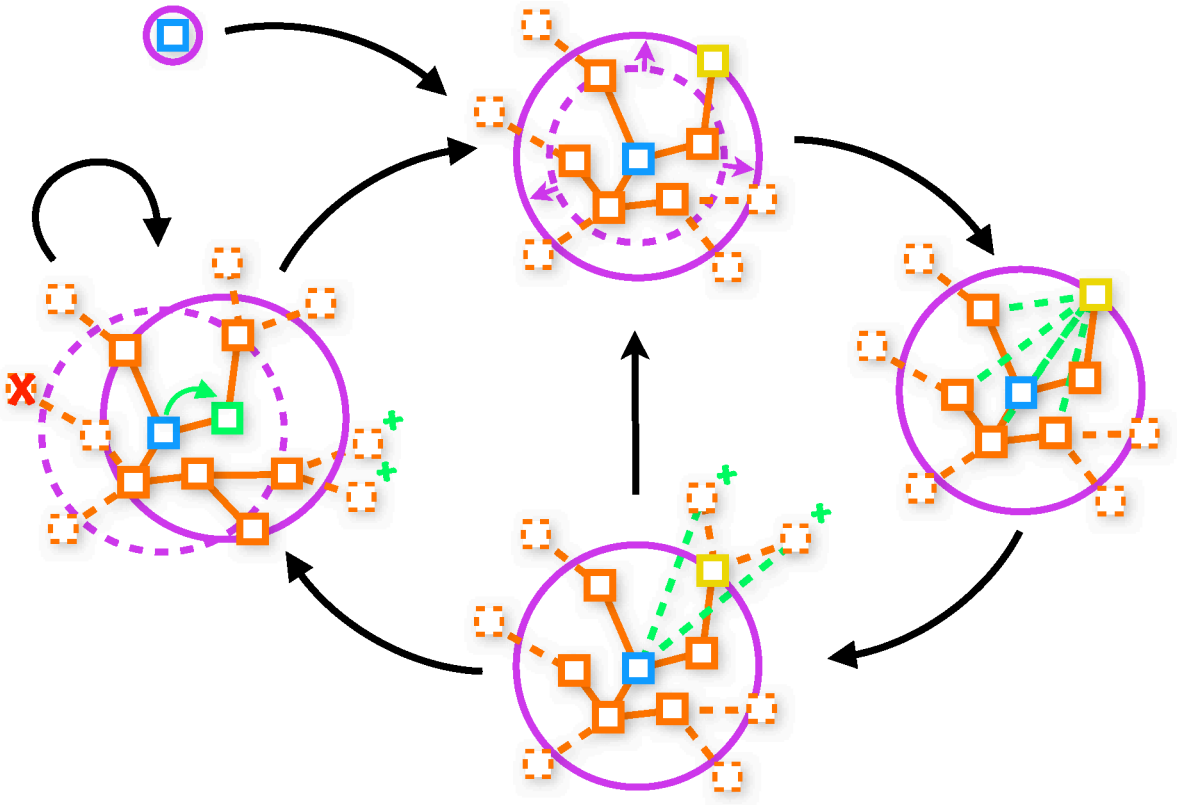


Figure 5.6: *Hierarchical Iconoid Shift. (top) The corona image closest to the medoid (yellow) is added to the support set. The kernel bandwidth is expanded to the distance of the new image to the medoid. (right) The pairwise distances between the new image and all other support set images are computed using HOP. (bottom) Images matching the new image are retrieved, inserted into the corona and added to the priority queue. (left) The medoid is shifted until convergence at a mode and the MST and priority queue are updated accordingly. If the mode did not change after adding the new image, this step is skipped.*

5.3.1 Computational Requirements

HIS can easily be parallelized in the same way as HMS (Fig. 5.4). Since adding an image to a support set has linear complexity in the number of images currently in the support set, the total complexity of growing a branch is quadratic in the number of images in the final support set, i.e., the number of images overlapping with the Iconoid. Since we can stop processing a branch if a worker merges into an existing branch (Sec. 5.2.4), the overall runtime of HIS depends on the branching factor of the dendrogram, and thus on the density distribution in the dataset. We observed that typically, there are a few prominent Iconoids that grow for a large range of bandwidths that most other branches merge into after a few growing steps. Like Iconoid Shift, Hierarchical Iconoid

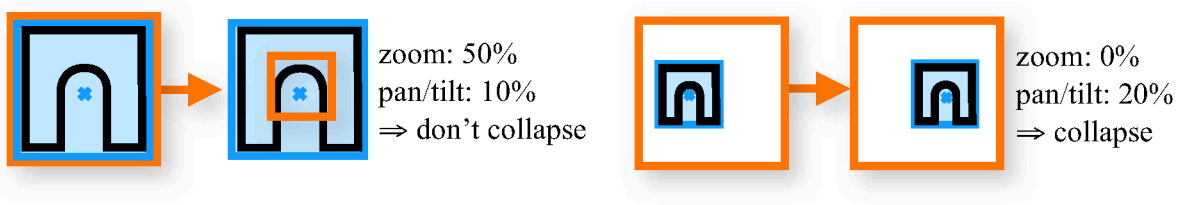


Figure 5.7: *Camera motion estimation based on the area and centroid of the overlap region (blue).*

Shift can be implemented with linear memory complexity in the number of images under the final support set. However, in our implementation, we decided to re-use previously computed overlaps (see above), which yields a performance improvement, but requires storing pairwise overlaps, and thus leads to quadratic memory complexity.

5.3.2 Dendrogram Simplification

We found that, in practice, the dendrograms produced by HIS often contain redundant seeming branches, e.g., branches that show only a slightly different view from the branch they merge into. This redundancy is mainly caused by two factors:

1. Each input image initially forms its own cluster. These clusters are the leaves of the dendrograms. The clusters gradually merge as the bandwidth is increased, but on the lower levels of the dendrogram, merging clusters are very similar in appearance.

2. If a landmark is photographed under different lighting conditions, the same view can have multiple modes, e.g., for day and night.

Since we are interested in creating compact hierarchical scene descriptions, we perform two measures to simplify the dendrograms that the algorithm produces.

1. We remove clusters with less than three images from the dendrograms. We choose three as the threshold since at least three images are required such that a cluster can have a unique Iconoid.

2. Since redundant dendrogram branches usually show very little change in perspective, we use a simple but effective scheme to simplify them. We descend each dendrogram from the root in a breadth-first fashion and estimate the camera motion for each outgoing edge of the current node. If the camera motion is too small, we collapse the edge, i.e., we remove the child and attach its children to the current node. We estimate the camera motion using a scheme similar to the one proposed by Ladikos et al. (2010): Since each child is in the support set of its parent, we can compute their overlap region using HOP. We then use the change in relative size

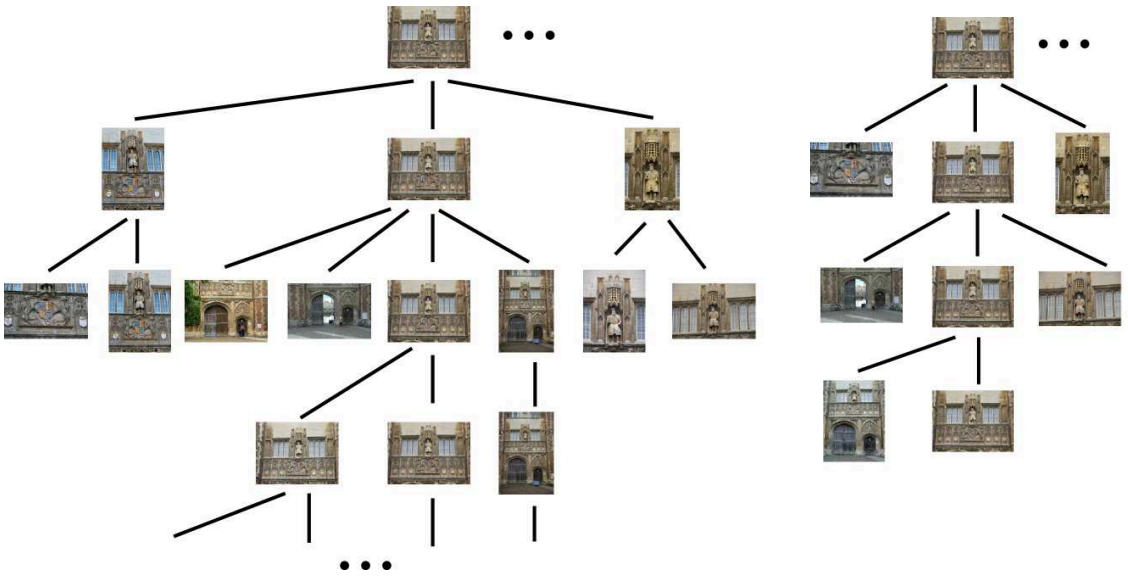


Figure 5.8: *Part of a dendrogram from* Trinity College *before (left) and after (right) simplification.*

of the overlap region as an estimate of the zoom and the relative movement of the centroid of the overlap region to estimate the amount of panning or tilt (Fig. 5.7). An example result of this simplification is shown in Figure 5.8. We collapse an edge if the size of the overlap region changes by less than 50% and its center shifts by less than 33% of the image size.

5.4 Experiments

We now perform experiments with Hierarchical Iconoid Shift on a large-scale dataset of 36 historic landmarks. We quantitatively and qualitatively analyze the dendrograms our algorithm produces for these landmarks. Moreover, we compare the resulting clustering to the clustering produced by Iconoid Shift in terms of the amount of discovered details. For this, we rely on Wikipedia authors to provide a list of details of each building and compare the number of details discovered by each algorithm. Finally, we discuss two potential applications: Extending Wikipedia articles and automatically building landmark summaries.

5.4.1 Dataset

To test the ability of our algorithm to discover details of landmarks, we collected a dataset of 36 landmarks that have a large number of details depicted on their Wikipedia pages and downloaded Flickr images in a geographic bounding box around them. The

dataset is divided into a total of 802,129 images. In two cases (*Westminster Palace / Westminster Abbey* and *Piazza San Marco / Basilica San Marco*), we combined two close-by landmarks. Thus, the resulting dataset has a total of 34 image sets.

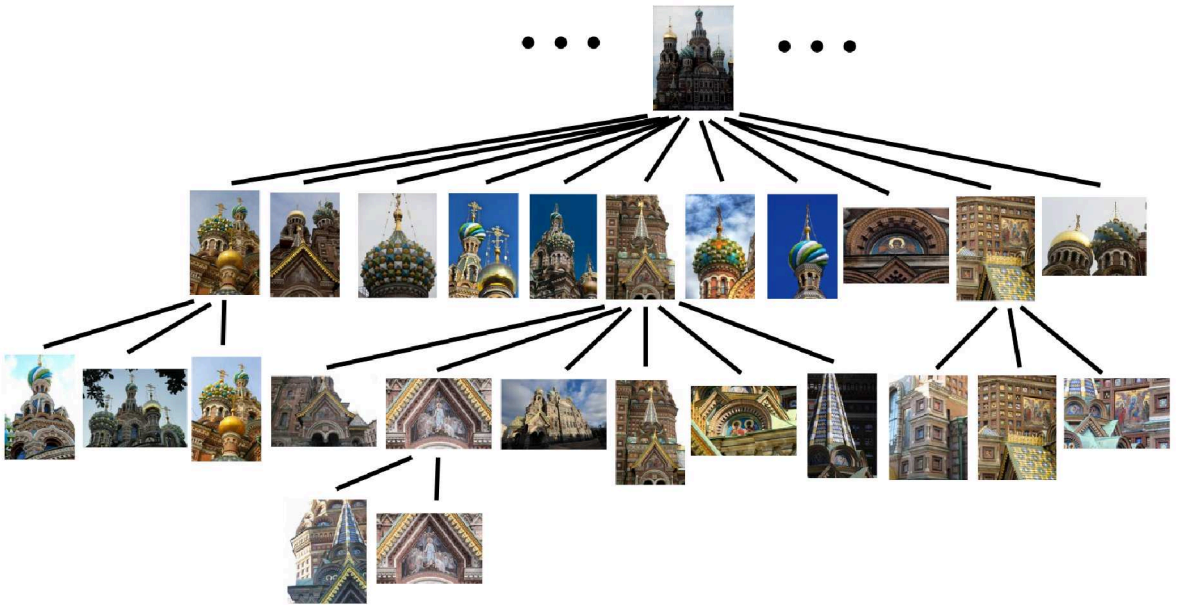
5.4.2 Results

Dendrogram Structure. We apply HIS on the 34 image sets by seeding it once with each image and run the algorithm in parallel on a computing cluster (Sec. 5.2.4). We then perform dendrogram simplification (Sec. 5.3.2), which reduces dendrogram size by 55.6% on average. HIS typically produces one dendrogram for each facade or view of a landmark and several smaller ones covering individual isolated objects. Figure 5.9 and Figure 5.10 show some typical examples of dendrograms produced by HIS. Some example paths from leaf to root are shown in Figure 5.11. It can be seen that the details discovered at lower bandwidth levels generally merge into more global structures when moving further up the dendrogram and thus to higher kernel bandwidths β. However, when increasing the bandwidth, sometimes the algorithm also shifts to more popular structures close to the current Iconoid (e.g. rows 3 and 4 of Fig. 5.11). Iconoids higher up in the hierarchy are therefore not necessarily show superordinate structures of their children, but might also be views where the camera has panned or tilted from a less popular view to a more popular one.

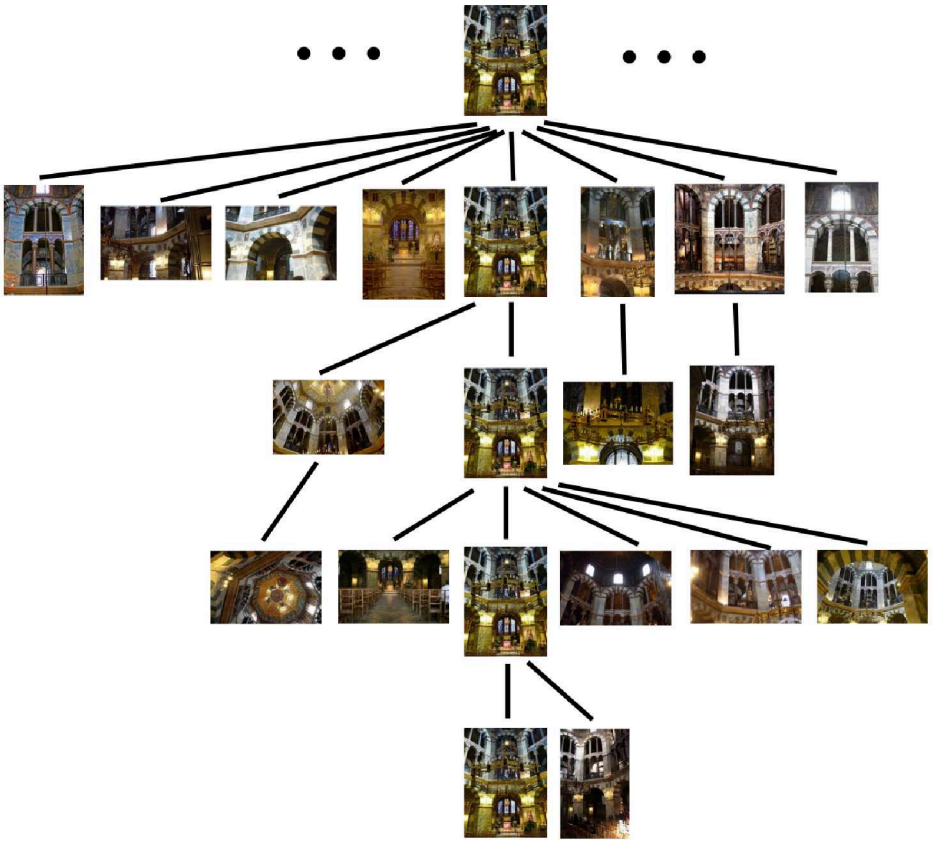
Dendrogram Statistics. Statistics of all dendrograms of the dataset are given in Table 5.1 and Table 5.2. The mean size of the dendrograms of a landmark can be an indication of how fragmented a scene is. For example, the mean dendrogram size of the *Louvre* and *Museée d'Orsay* is low, because every individual artwork in these museums will form its own (very shallow) dendrogram. The *Angkor Wat* and *Wat Phra Kaew* temples are spread over a larger geographical area and are therefore fragmented into several scenes that each form smaller dendrograms. The largest dendrograms are formed by landmarks that consist of a single, complex structure such as the *Pantheon*, the *Church on Spilt Blood* and the *Trevi Fountain*.

Singletons are single Iconoids whose dendrograms were either collapsed into a single node by the dendrogram simplification step or that contain only 3 images. They are typically objects with no superordinate structures, e.g., paintings in a museum, which explains the high number of singletons in the Louvre. Another cause of singletons are "junk" photos, which were either wrongly geotagged as being close to the landmark, or simply do not depict the landmark.

Figure 5.12 (left) shows the development of the number of clusters as a function of β. As β increases, HIS converges to larger scale structures, causing more and more dendrogram branches to merge, which in turn decreases the number of clusters. The steep drop in the beginning is caused by the quick formation of Iconoids as well as the dropout of images that do not have a match.

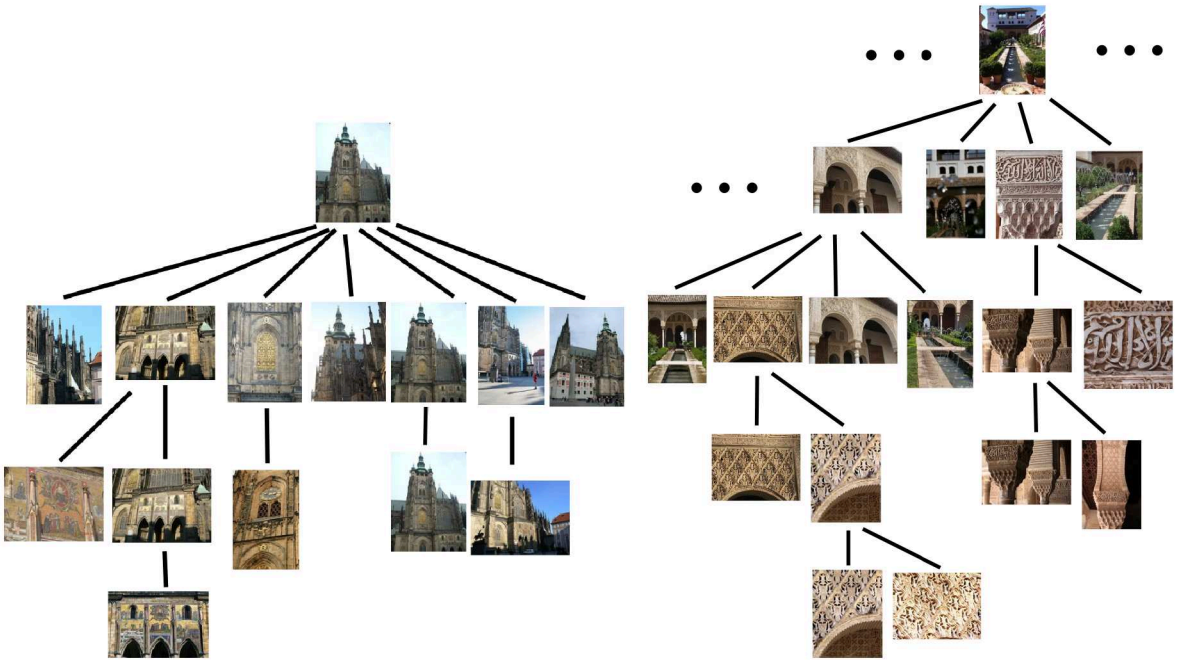


(a) *Church on Spilt Blood*

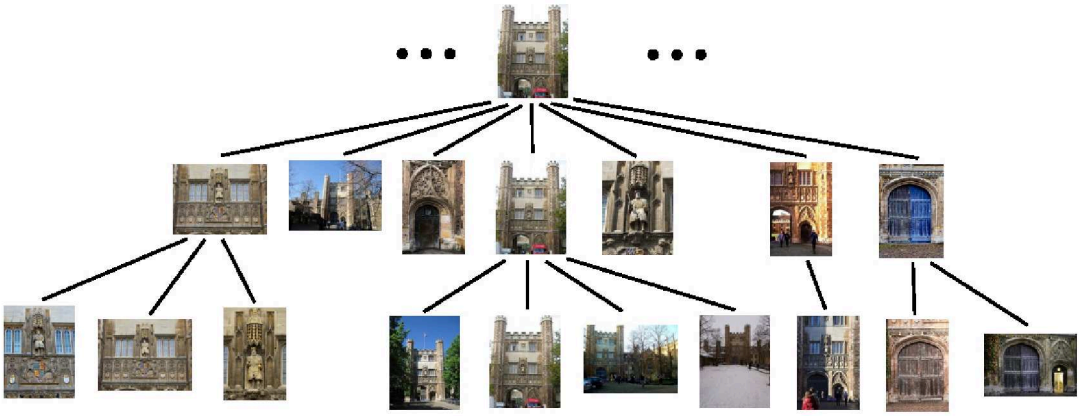


(b) *Aachen Cathedral*

Figure 5.9: *Example dendrograms of detail hierarchies automatically discovered by Hierarchical Iconoid Shift (part 1).*



(a) *Old Town Square Prague* (b) *Alhambra*



(c) *The Great Gate of Trinity College*

Figure 5.10: *Example dendrograms of cluster structures automatically discovered by Hierarchical Iconoid Shift (part 2).*

How many details does HIS find? We would now like to know how many of the details actually present in a scene are automatically discovered by our algorithm. To answer this question, we rely on Wikipedia authors to provide a list of the relevant details of a building. For each landmark, we downloaded the Wikipedia article in the language version having most details (This is usually the language of the country where the landmark is located). For the two joint image sets *Westminster Palace and Westminster*

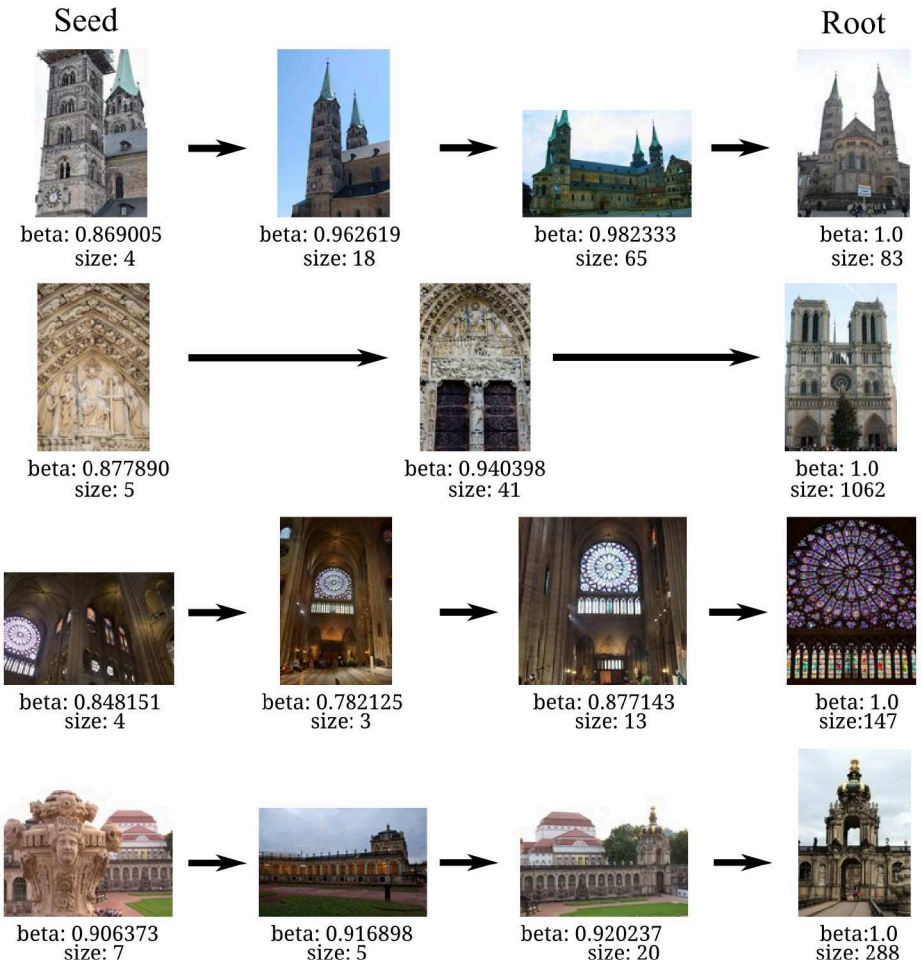


Figure 5.11: *Example convergence paths from seed (left) to root (right). Captions show cluster bandwidth and size.*

Abbey and *Piazza San Marco and Basilica San Marco*, we downloaded both Wikipedia articles. We then extracted a textual label for each detail based on its HTML alt-tag or image caption. Since only a part of the photos in the Wikipedia article of a landmark depict details, we manually removed non-detail photos like floor plans or portraits of people associated with the landmark. We then matched the Wikipedia details against our discovered details using visual word based image retrieval (Sec. 3.2.2). A Wikipedia image matches an Iconoid if they are related by a homography with at least 15 inliers and their homography overlap distance is less than 0.9. To ensure high recall for this matching, we additionally consider matches between Wikipedia details and images directly adjacent to the Iconoid in its matching graph. For these matches, we compute the overlap between the Wikipedia image and the Iconoid using HOP (Sec. 4.5.2). If an Iconoid had multiple Wikipedia matches, we keep the one with the lowest overlap distance.

We note that there are two caveats with this approach: Firstly, we cannot expect an exhaustive list of details from Wikipedia, and secondly, SIFT-based matching and spatial verification using homographies is not suitable for all details. For example, many of the details of *Milan Cathedral* and *Notre Dame* are statues, which were discovered by HIS, but that could not successfully be matched, because they are weakly textured and highly non-planar. Here, specialized matching techniques for weakly-textured objects would be necessary (Arandjelovic and Zisserman, 2011). Therefore, the numbers we provide should be considered a pessimistic measure of the recall of detail discovery.

The results of this experiment are shown in the bottom two rows of Table 5.1 and Table 5.2. *Wikipedia Details* is the number of details depicted in each article and *WP* $\rightarrow$ *Iconoid* is the number of Wikipedia images with at least one matching Iconoid. Generally, the number of details discovered depends on the number of images available for the landmark, since a dense enough coverage of the landmark is needed to identify density modes. HIS therefore finds a large fraction of the details of *Piazza & Basilica San Marco* and *Notre Dame*, but almost none for *Nidaros Cathedral*, which has only 581 images. Overall, about half of the Wikipedia details have a matching Iconoid.

Which details does HIS miss? As we noted above, the number of matched Wikipedia details is a pessimistic measure of the number of details discovered by our algorithm. Nevertheless, there were several cases where our algorithm missed details that are present on Wikipedia. We identified three main reasons for this (see Fig. 5.13):

- Some details, like the ones depicted in Figure 5.13a, are historically relevant and therefore mentioned in Wikipedia, but not visually striking enough to attract photographers. Without a tourist guide or guidebook, people visiting a landmark might even be unaware of those details. Therefore, there are not enough photos of them to form a mode. Such details simply violate our assumption that details are frequently photographed objects and thus our algorithm cannot discover them by design.

- Other details, like the ones depicted in Figure 5.13b, are too small to be photographed in isolation. For example, the bottom right detail is a part of the ceiling mural in the Sistine Chapel. With a standard camera, like a cheap consumer camera or smartphone, it is impossible to photograph this part in isolation. This would require either a telephoto lens, or cropping the detail out afterward. For such details, only the superordinate structure will form a cluster. Moreover, details like ornaments (left) that are found all over a building are unlikely to form a cluster since every photographer will focus on a different part and the photos of it do not overlap.

- Finally, some building details, like the ones depicted in Figure 5.13c, are hard to access, e.g., because they can only be seen when taking a (possibly paid) guided tour. Other objects, like the church bell, might not even be accessible to the public at all. Finally, in many touristic attractions, like churches, photography is simply

Landmark	Aachen Cathedral	Alhambra	Angkor Wat	Arc de Triomphe	Bamberg Cathedral	Branden-burg Gate	Chichen Itza	Church on Spilt Blood	Cologne Cathedral
Images	2,077	55,532	26,485	33,128	964	34,684	9,176	3,588	21,104
Dendrograms	40	562	445	176	22	290	124	21	210
Mean Dend. Size	7.28	11.22	6.88	12.77	6.86	7.93	8.56	17.33	11.44
Singletons	47	563	409	198	16	391	122	20	264
Iconoids	277	5,801	2,850	2,137	128	2,321	1,003	335	2,267
Images Covered	1,192	32,783	13,641	23,731	552	14,221	5,422	2,459	12,900
Wikipedia Details	22	12	5	23	52	9	19	7	18
WP → Iconoid	10	9	2	14	10	4	7	6	9

Landmark	Old Town Sq. Prague	Pantheon	Piazza& Basilica San Marco	Plaza de Espana	Reichstag	Sagrada Familia	Sistine Chapel	St Pauls Cathedral	St Peters Basilica
Images	28,349	17,832	40,654	7,689	22,064	17,249	31,337	30,345	32,882
Dendrograms	242	105	331	64	181	176	312	200	304
Mean Dend. Size	10.83	20.41	12.71	8.48	10.59	13.52	11.53	14.32	12.31
Singletons	283	95	408	96	224	162	242	268	228
Iconoids	2,485	1,677	4,009	546	1,866	2,173	3,245	2,735	3,361
Images Covered	19,399	13,960	22,780	2,522	11,013	12,554	25,606	20,367	26,509
Wikipedia Details	9	12	36	7	11	37	41	10	52
WP → Iconoid	7	9	21	3	4	22	14	5	31
Iconoid Shift									
Iconoids			411			645	387		
Images Covered			16,468			10,569	21,038		
WP → Iconoid			13			8	2		

Table 5.1: *Results of HIS on 34 landmarks (part 1).* Images Covered *is the total number of images in Iconoid support sets.* Singletons *are individual Iconoids that are not part of a dendrogram.*

forbidden, which severely limits the amount of photos of them that are available in Internet photo collections.

The missed details revealed some interesting hidden assumptions that are not only present in our algorithm, but also in other visual object discovery approaches. In order to be discovered by visual clustering, an object must be *visually striking*, *accessible to photographers*, and it must be *possible to photograph it in isolation*.

How does HIS compare to Iconoid Shift? We compare HIS to Iconoid Shift (Ch. 4) on a subset of 7 landmarks (lower block of Tab. 5.1 and Tab. 5.2). The Iconoid Shift clustering covers almost the same number of images, but has a much coarser granularity, resulting in significantly less Iconoids. The reason for this is that IS performs mode

Landmark	Edinburgh Castle	Florence Cathedral	Hagia Sophia	Louvre	Milan Cathedral	Museée d'Orsay	Nidaros Cathedral	Notre Dame
Images	27,465	37,035	24,823	69,928	41,192	17,879	581	35,709
Dendrograms	287	296	241	884	341	312	11	266
Mean Dend. Size	11.73	12.82	11.40	8.09	13.75	6.25	4.27	13.42
Singletons	332	289	196	1,046	381	398	11	266
Iconoids	3,128	3,456	2,510	6,974	4,378	1,970	48	3,257
Images Covered	17,618	28,444	16,911	36,742	25,591	11,775	218	24,841
Wikipedia Details	18	29	26	256	60	36	17	90
WP → Iconoid	12	18	17	99	22	25	1	54

Landmark	St Stephens Cathedral	St Vitus Cathedral	Taj Mahal	Trevi Fountain	Trinity College	Wat Phra Kaew	Westminster Palace & Abbey	Zwinger
Images	15,399	21,392	11,070	13,502	3,815	29,453	31,010	5,737
Dendrograms	199	215	67	49	72	406	270	59
Mean Dend. Size	9.78	12.20	14.87	15.41	5.33	8.88	12.54	13.71
Singletons	221	130	106	70	75	372	293	49
Iconoids	1,805	2,324	972	739	373	3,313	3,200	726
Images Covered	8,894	16,569	8,112	10,773	1,621	17,762	22,732	3,814
Wikipedia Details	46	15	37	8	11	13	36	17
WP → Iconoid	19	10	8	6	5	6	16	8
Iconoid Shift								
Iconoids	606		279		161			218
Images Covered	7,307		6,869		1,286			3,143
WP → Iconoid	14		4		3			6

Table 5.2: *Results of HIS on 34 landmarks (part 2).*

search only on a single fixed scale ($\beta = 0.9$ in our experiment), while HIS finds modes on all scales. Comparing the number of discovered Wikipedia details, it can be seen that especially for landmarks like *Sagrada Familia* or the *Sistine Chapel* that have many close-by details, IS finds significantly less details. On the other hand, for landmarks like *Zwinger* where the objects on Wikipedia are mostly non-hierarchical, e.g. several distinct buildings, the advantage of HIS is smaller. Figure 5.14 compares objects discovered by both algorithms (left) to objects only discovered by HIS.

How to name the discovered details? We found that the Wikipedia image captions can serve as very accurate labels for the Iconoids. As we will show in Chapter 6, high-quality labels for smaller, less popular objects, cannot be reliably found by examining frequently occurring terms in photo titles and tags (Quack et al., 2008; Simon et al., 2007;

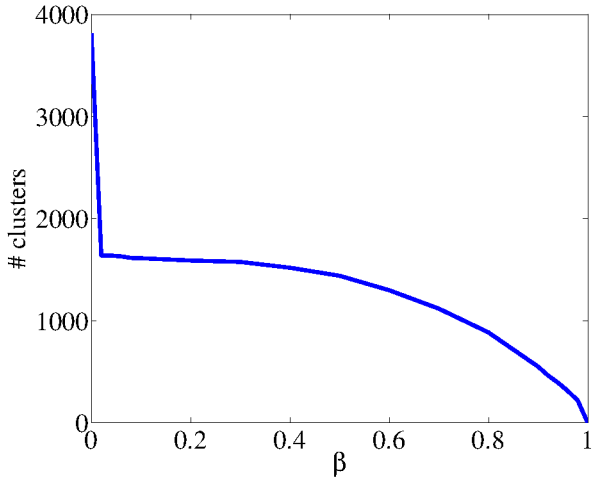


Figure 5.12: *Development of the number of clusters as the kernel bandwidth β increases (for* Trinity College*). At $\beta = 0$, the number of clusters equals the number of images in the dataset. At $\beta = 1$, the number of clusters equals the number of dendrograms, because only the roots of the dendrograms survive until a kernel bandwidth of 1.*

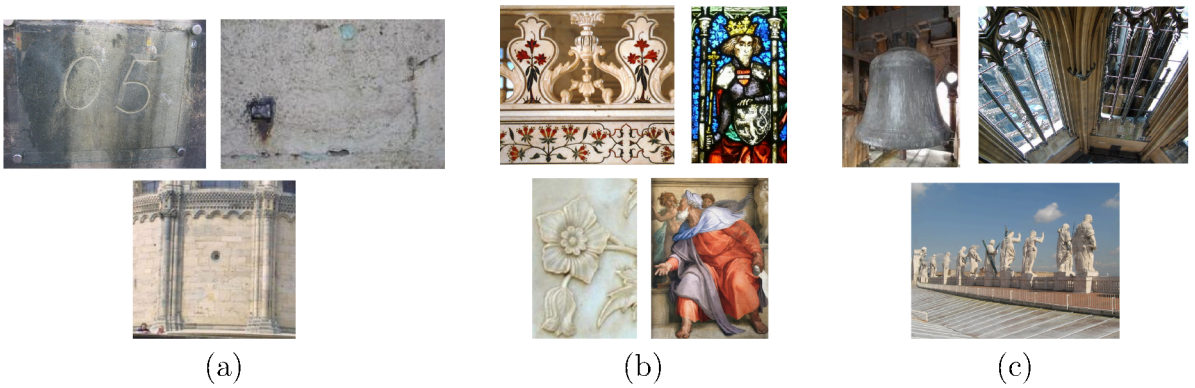


Figure 5.13: *Building details that are mentioned on Wikipedia, but not discovered by HIS. (a) Details that are historically relevant, but not visually striking. (b) Details that are hard to photograph in isolation without a telephoto lens or cropping them out. Top left and bottom left: ornaments. Top right: part of a larger stained glass window. Bottom right: small part of the ceiling mural in the Sistine Chapel. (c) Details that are hard to access or not always open to the public.*

Zheng et al., 2009), since photographers often do not (correctly) label landmark details. Some examples of Iconoids with matching Wikipedia details are shown in Figure 5.15.

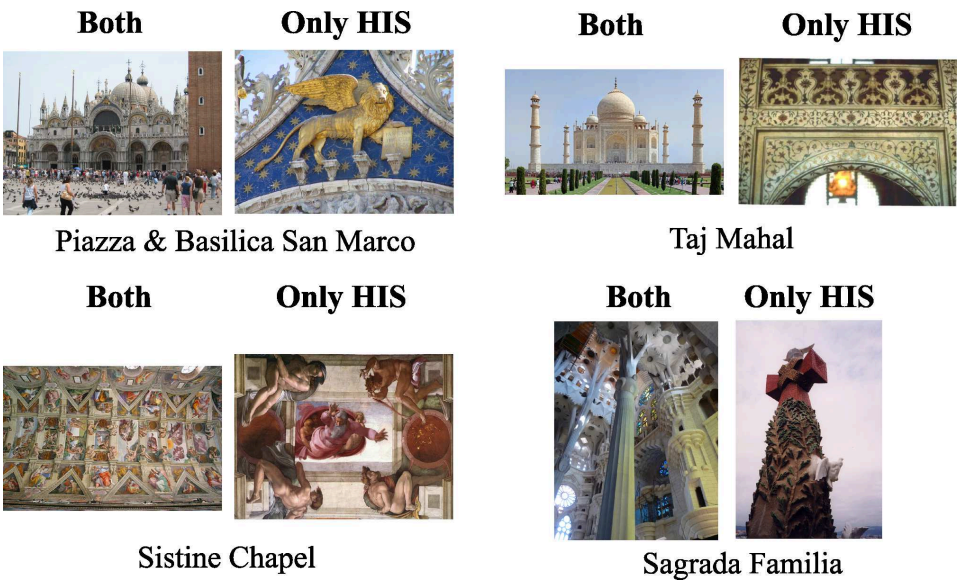


Figure 5.14: *Objects discovered by both HIS and IS (left) vs. objects discovered only by HIS (right). While both algorithms find larger-scale structures, only HIS also finds small-scale details.*

5.4.3 Applications

We now outline two potential applications for the cluster hierarchies produced by HIS.

Extending Wikipedia. While HIS did not discover some of the details present on Wikipedia (see above), many of the details it discovered were not mentioned in the articles. A potential application of HIS would therefore be to suggest new details to be added to Wikipedia. Our prototype implementation of this idea uses the section structure of Wikipedia articles to build a *Wikipedia graph* for each landmark. In this tree, the landmark is the root node, its children are the top-level sections, the children of each section are its sub-sections, and so on. Each image is inserted as a leaf under its respective (sub-)section node. We then match the Wikipedia images against the discovered Iconoids as described above to create links between the Wikipedia graphs and the HIS dendrograms. By exploiting these links and the estimated camera motion between dendrogram nodes (Sec. 5.3.2), it is sometimes possible to propose the article section where a new detail should be added.

As an example, Figure 5.16 shows the main gate of *Bamberg Cathedral* that has two figure groups on the left and right side of the entrance. In the HIS dendrogram, both of them were correctly discovered as children of the entrance itself, but only the left one is mentioned in Wikipedia and is therefore the only child of the gate in the Wikipedia graph. By matching Iconoids to Wikipedia details, we established links between the gate nodes (yellow) and the left figure group nodes (orange). We know that the right figure group is a part of the gate since the gate is its parent in the dendrogram and the

Figure 5.15: *Example Iconoids (left) with matched Wikipedia images and labels (right).*

camera motion associated with the edge is "zoom out". We can therefore propose to add this node to the article section describing the main gate.

Landmark Summaries. Another application is depicted in Figure 5.17 and Figure 5.18. Using the discovered dendrograms describing the scene structure, as well as the image clusters associated with each node, we can build visualizations of landmark building facades that are far more useful to the user of a mobile visual search application than the name of the main landmark alone. The images in the Iconoid clusters can serve two purposes here. Firstly, they help localize the details on the facade using either homography overlap propagation or structure-from-motion. Secondly, as in Gammeter et al. (2010), the clusters provide additional images for each detail showing it under different lighting conditions and viewing angles and can thus be used for "offline query expansion" to make recognition of these details more robust compared to matching against the Wikipedia images alone.

In contrast to "3D Wikipedia" (Russell et al., 2013), which produces similar summaries by associating bounding boxes in the structure-from-motion reconstruction of a landmark with noun phrases from its Wikipedia article, our approach does not rely on Wikipedia articles, but only on the distribution of tourist photos from Internet photo collections.

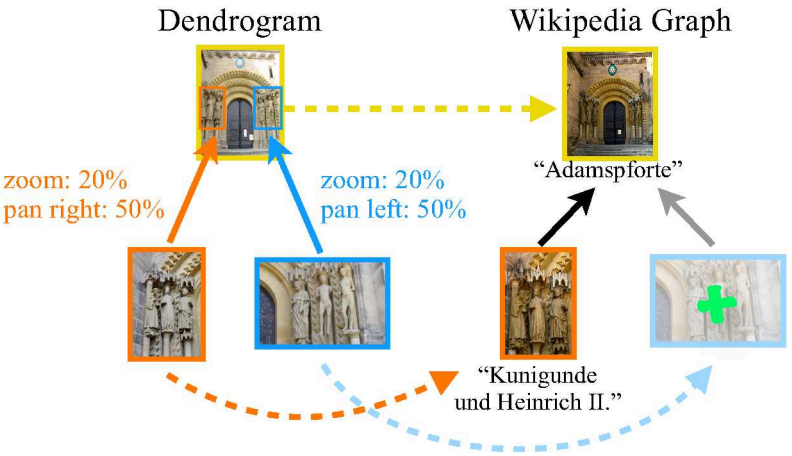


Figure 5.16: *Finding new details to add to Wikipedia. The right child of the node showing the gate is not present on Wikipedia, but we can propose a place to insert it into the article based on its parent and sibling in the dendrogram and the estimated camera motion between them.*

It therefore also finds details not mentioned on Wikipedia and works on landmarks for which no Wikipedia article exists.

5.5 Conclusion

In this chapter, we have presented the *Hierarchical Iconoid Shift* (HIS) algorithm that automatically discovers details of landmark buildings in large-scale collections of tourist photos from the Internet. HIS is based on the *Hierarchical Medoid Shift* (HMS) algorithm, a new variant of Medoid Shift (Sheikh et al., 2007) that, instead of a flat clustering at a single scale, produces a *dendrogram of clusters*. Similar to the Scale Space Filtering algorithm (Witkin, 1984), HMS follows the evolution of density maxima as the kernel bandwidth increases and builds a dendrogram from their merging behavior. However, in contrast to previously proposed hierarchical variants of Mean Shift, HMS increases the bandwidth in a *quasi-continuous* way, which eliminates the need to choose a bandwidth increment and makes parallel execution more efficient, since different threads can work on different bandwidth levels simultaneously. HMS is completely parameter-free and has the same computational complexity as Medoid Shift. We have presented a scheme to parallelize HMS to many machines that makes it applicable to large-scale datasets.

Hierarchical Iconoid Shift applies HMS to the task of clustering large-scale image collections. HIS produces a hierarchical clustering of the photos of a landmark where the lower nodes correspond to individual details and the root is a full view of the building. Given a seed image, HIS iteratively adds new images to the kernel window and performs mode search to find the new Iconoid. We have presented a linear-time algorithm for

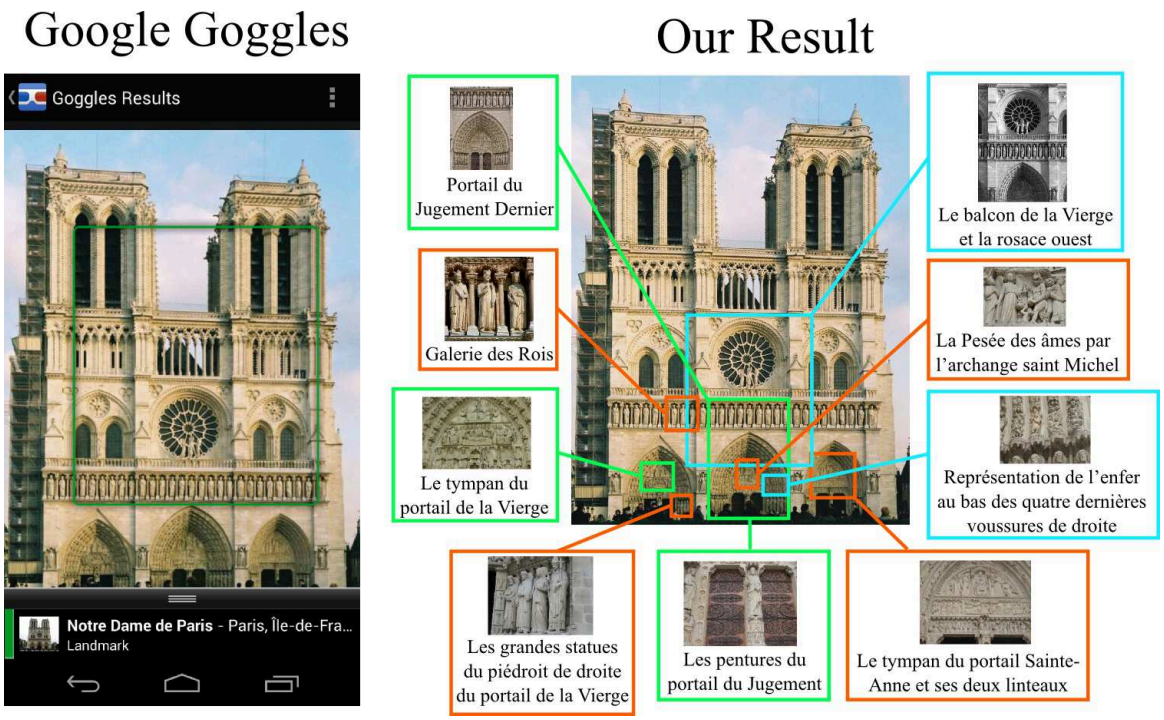


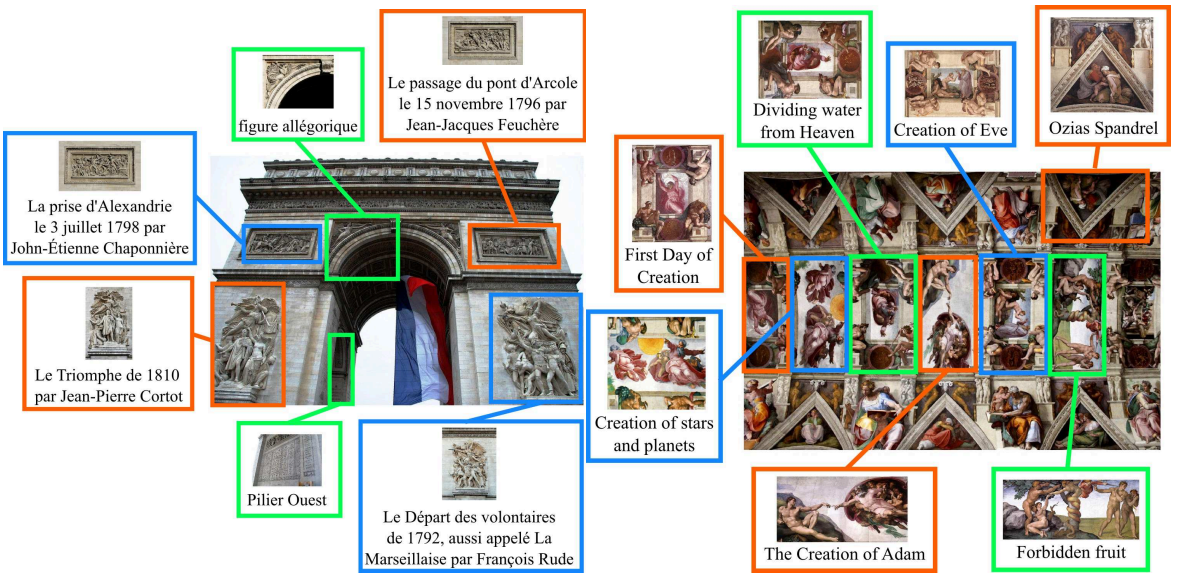
Figure 5.17: *While Google Goggles only recognizes the entire building, the details discovered by our algorithm could be used to provide the user with more useful information.*

performing the growing operation and a scheme for shifting the medoid that makes use of previously computed overlaps to save computation time.

We have demonstrated our algorithm on a set of 36 landmarks with a large amount of detail and shown that it discovers many of the details depicted on Wikipedia and outperforms Iconoid Shift (Ch. 4) in terms of the number of discovered details. The scene hierarchies produced by HIS could be useful for a large range of applications including landmark description and visual recognition of detail structures.

A problem of the dendrograms produced by Hierarchical Iconoid Shift is redundancy, which we addressed by eliminating nodes whose perspective is too similar to their parent (Sec. 5.3.2). However, even after this simplification not all dendrogram nodes are meaningful objects. A still open question is therefore if it is possible to automatically rate the relevance of a node. For example, Leung et al. (2000) propose several measures such as *lifetime*, *compactness*, *isolation* and *outlierness* to characterize the clusters produced by their hierarchical mode search algorithm that might be applicable in our case.

Another future direction for both IS and HIS would be to extend them to enable *incremental* clustering. Since the amount of tourist photos on the Internet is increasing rapidly, restarting the clustering process from scratch every time new images become available may soon become prohibitive. A scheme to update an existing IS or HIS clus-



(a) *Arc de Triomphe* (b) *Sistine Chapel*

Figure 5.18: *Details of the Arc de Triomphe and the Sistine Chapel mural discovered by Hierarchical Iconoid Shift. Captions were automatically obtained from Wikipedia.*

tering with new photos that re-uses previously computed overlaps could save significant amounts of computation time.

Ultimately, Hierarchical Iconoid Shift could be used to produce *knowledge graphs* of landmark buildings that represent the knowledge of a building in a structured way. Such a graph could be used in a system like Wolfram Alpha[1] to answer questions like *"What is the name of the leftmost portal of Notre Dame?"* or *"How many spires does the Church on Spilt blood have?"*. This graph could be constructed by combining the *semantic* information encoded in the Wikipedia articles (What are the details called? What object categories do they belong to?) with the *spatial* information in the HIS dendrograms (Where are the details located relative to each other? Which details are parts of which other details?) While such a graph could be very useful for many applications, we believe that the methods required for semantically parsing articles on landmarks have to go far beyond simply parsing the document structure. Here, methods from the field of natural language processing will be necessary to understand the article on a deeper level and build a detailed graph of the mentioned entities and their relationships.

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

6

Evaluation of Landmark Recognition based on Internet Photo Collections

In the previous chapters, we have presented algorithms for clustering large sets of photos from Internet photo collections to discover landmarks, their details, and other objects of interest. There are several applications for the resulting clusters, including landmark recognition, 3D reconstruction or interactive tourist guides. In this chapter, we focus on *landmark recognition*, i.e., recognizing landmarks and other objects in a new query photo. Applications of landmark recognition include *photo auto-annotation* and *mobile visual search*. A photo auto-annotation system recognizes objects in a user's photo albums and labels them accordingly, saving the user the effort of manually labeling them. A mobile visual search system provides a user with information on an object that they took a picture of with their smartphone.

Constructing the database of objects underlying a landmark recognition system automatically from Internet photo collections has several advantages: Firstly, objects are discovered in an unsupervised, fully automatic way, making it unnecessary to manually create a list of objects and collecting photos for each of them. Secondly, the level of detail of the object representation is automatically adapted to the demand. The most popular objects will be represented by the most photos in the database, increasing their chance of successful recognition, while only little memory is used on less popular objects. Thirdly, in the case of photo auto-annotation, the database is built from the data it is meant to be applied to, namely photos from Internet photo collections. Therefore, the resulting set of objects is likely to be much better adapted to the queries a photo auto-annotation or visual search system might receive than a hand-collected set of objects. The approach to construct landmark recognition systems from Internet photo collections has gained popularity in the research community (Avrithis et al., 2010; Gammeter et al., 2009; Kalantidis et al., 2011; Quack et al., 2008; Weyand and Leibe, 2011) and is also being used in applications such as Google Goggles (Zheng et al., 2009).

The first step of constructing a landmark recognition system from Internet photos is to discover interesting objects in the image collection using a landmark clustering

algorithm. Each discovered object is represented as a cluster of photos. Then, a name or description for each of these objects is determined, typically by examining user-provided titles and tags. After this, a compact retrieval index for efficient recognition is built from the photos of the discovered objects. To identify the object in a query image, matching images from the database are retrieved. Based on the matches, the system determines which object is present in the query.

In this chapter, we evaluate this whole process of constructing landmark recognition systems from Internet photos and provide answers to the following questions:

- How many and what kinds of objects are present in Internet photo collections and what is the difficulty of discovering different landmark types (Sec. 6.4)?
- How to decide which landmark was recognized given a list of retrieved images (Sec. 6.5)?
- How to efficiently represent the discovered objects in memory for recognition (Sec. 6.6)?
- Are the user-provided tags reliable enough for determining accurate object names (Sec. 6.7)?
- Given the entire retrieval, recognition and semantic labeling pipeline, what are the factors effectively limiting the recognition of different object categories (Sec. 6.8)?

We evaluate landmark recognition in a realistic, large-scale setting using the Paris 500k dataset (Sec. 3.3). Using Iconoid Shift (Ch. 4) as an analysis tool we determine how many and what kinds of objects are present in the dataset and use the resulting clustering as the basis for landmark recognition. To evaluate the quality of landmark recognition, we collected a query set of almost 3,000 Flickr images from Paris and created an exhaustive ground truth for the relevance of each of the discovered landmarks with respect to each of the 3,000 queries. To give a detailed performance analysis for different types of objects, we introduce a taxonomy for the objects landmark recognition systems are able to recognize. Throughout our evaluation, we report both summary performances over the entire database and detailed findings for different object categories that show how their recognition is affected by the different stages of the system.

This chapter is based on our paper " Visual Landmark Recognition from Internet Photo Collections: A Large-Scale Evaluation" (Weyand and Leibe, 2015).

6.1 Engine Architecture

The architecture of a typical landmark recognition engine such as the ones presented by Avrithis et al. (2010); Gammeter et al. (2010); Quack et al. (2008); Zheng et al. (2009) is shown in Figure 6.1. Large amounts of tourist photos are clustered, resulting in a set

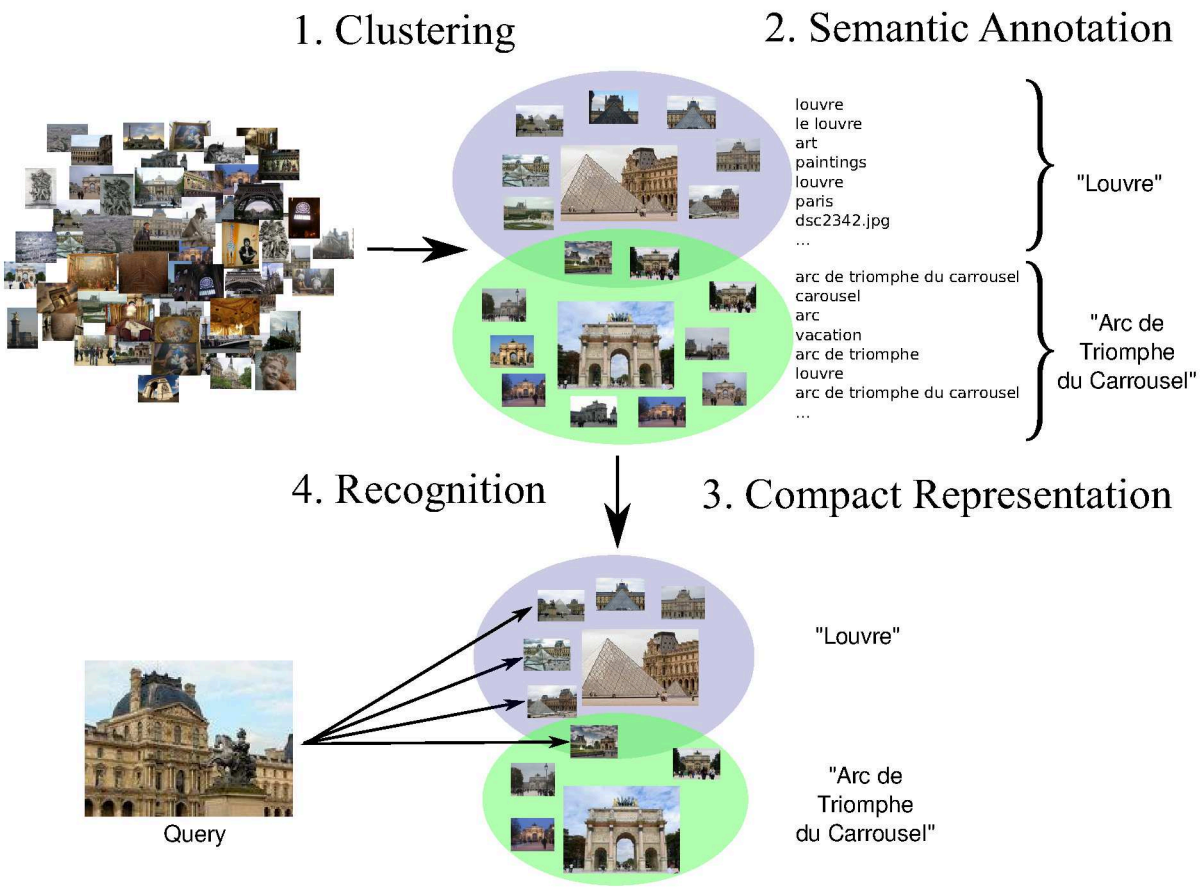


Figure 6.1: *The architecture of a landmark recognition system. 1.* Objects *are discovered by visually clustering touristic photos (Sec. 6.4). We call the photos in an object cluster its* representatives. *2.* Semantic *annotations are mined from user-provided tags (Sec. 6.7). 3. The search index is made more compact by eliminating redundancy (Sec. 6.6). 4. The object in a query photo is recognized by retrieving similar photos and exploiting the knowledge of their cluster memberships (Sec. 6.5).*

of *objects*. By *object*, we denote a cluster of images that show the same entity. We will refer to the images in each object cluster as its *representatives*. Since the clusters may overlap, a representative can belong to multiple objects. Each object is then associated with *semantics* (typically its name), e.g., by mining frequently used image tags. The set of representatives for each cluster is often decimated by eliminating redundant images in order to save memory and computation time. To recognize the object in a query image, a visual search index (Sec. 3.2.2) containing all *representatives* is queried, producing a ranked list of matches. Based on this list, *objects* are ranked w.r.t. their relevance to the query and the corresponding *semantics* are returned.

In this chapter, we evaluate different choices for the components of this framework and demonstrate how they affect the system's overall performance. Section 6.4 considers the

stage of determining a set of objects by clustering images from internet photo collections and shows how many objects from which categories can be discovered. Given a ranking of the representatives for a query, Section 6.5 analyzes different schemes for determining the object shown in the query image. In Section 6.6, we consider different ways of speeding up search and reducing memory requirements by removing redundant representatives. Finally, in Section 6.7 we analyze the stage of semantic annotation based on frequent tags and perform an end-to-end analysis of the performance of the whole pipeline from query to semantics.

6.2 Related Work

We now give an overview how the individual parts of the pipeline introduced above have been approached in previous work. For a general discussion of previous work in landmark recognition, we refer the reader to Section 2.3.

6.2.1 Datasets

We created our own query set and ground truth for the experiments in this chapter, because the available benchmarks did not support such an evaluation. Most datasets only cover very few, mostly building-scale, landmarks (e.g., European Cities 1M (Avrithis et al., 2010), Statue of Liberty, Notre Dame and San Marco (Li et al., 2008), Oxford Buildings (Philbin et al., 2007), Paris Buildings (Philbin et al., 2008)). Another problem is that their ground truths are designed for other tasks. Image retrieval datasets (e.g. Oxford Buildings (Philbin et al., 2007), Paris Buildings (Philbin et al., 2008), INRIA Holidays (Jégou et al., 2008)) are not suitable for our evaluation, because we want to evaluate object recognition, i.e., recognizing the object(s) in a query image, and not image retrieval, i.e., retrieving images similar to a query from a database. Image-based localization datasets (Aachen (Sattler et al., 2012), Vienna (Irschara et al., 2009), Dubrovnik and Rome (Li et al., 2010)) evaluate how accurately the camera pose of the query image can be estimated. While this is more related to our problem, our goal differs from pose estimation, because camera pose does not necessarily determine what object the camera is really seeing (we discuss this difference in more detail in Sec. 2.3.3). The San Francisco (Baatz et al., 2010) and Landmarks 1K (Li et al., 2012) datasets are closest to our requirements, but both of them focus on large, building-level landmarks while we are explicitly interested in also evaluating the recognition of smaller, non-building objects.

6.2.2 Landmark Discovery

Landmark Recognition Engines are typically based on a visual search index from objects discovered in Internet photo collections (Avrithis et al., 2010; Gammeter et al., 2010;

Quack et al., 2008; Zheng et al., 2009). Since we already discussed previous work in landmark discovery in previous chapters, we refer the reader to the following sections: We gave an overview of the underlying clustering approaches in Section 2.2 and analyzed them w.r.t. their grouping criteria and clustering algorithms in Section 4.1. Moreover, we evaluated the approaches by Chum and Matas (2010) and Philbin and Zisserman (2008) in Section 3.4.

6.2.3 Eliminating Redundancy

Several methods have been proposed to reduce the size of the visual search index. An obvious method is to apply standard compression techniques (Jégou et al., 2009b), which reduces memory consumption at the cost of computational efficiency. Instead, we are interested in *eliminating redundancy* already before index construction.

Several works have addressed this problem at the *image level*, i.e., by removing redundant images from the index. Li et al. (2008) summarize the input image collection in a set of *iconic* images by applying k-means clustering (Duda et al., 2000) based in GIST descriptors (Olivia and Torralba, 2001), and use only these images to represent a landmark in retrieval. (Gammeter et al., 2010) identify sets of very similar images using complete-link hierarchical agglomerative clustering and replace them by just one image. This step yields a slight compression of the index without loss in performance. Instead of performing clustering, Yang et al. (2011) only determine a set of canonical views by applying PageRank (Page et al., 1999) on the matching graph of the image collection. They then discard all other views and match the query only against the canonical views.

Other works have addressed the problem at the *feature level*. Turcot and Lowe (2009) perform a full pairwise matching of the images in the dataset and remove all features that are not at least once inliers w.r.t. a homography. They report a significant reduction of the number of features while maintaining similar retrieval performance. Avrithis et al. (2010) and Johns and Yang (2011) combine the images in a cluster into a joint bag-of-visual-words representation. Avrithis et al. (2010) use Kernel Vector Quantization (Tipping and Schölkopf, 2001) to cluster redundant features and keep only the cluster centers. While this method only yields a slight compression, the aggregation of features into a *scene map* brings significant improvements in recognition performance. (Johns and Yang, 2011) perform structure-from-motion and summarize features that are part of the same feature track. (Gammeter et al., 2009) estimate bounding boxes around the landmark in each image in a cluster and remove every visual word from the index that never occurs inside a bounding box. This is reported to yield an index size reduction of about a third with decreasing precision.

There is also work in *pose estimation* that aims to eliminate redundancy in the dataset. In their hybrid 2D-3D pose estimation approach, Irschara et al. (2009) generate a set of synthetic views by projecting the points of a structure-from-motion reconstruction onto a set of virtual cameras placed at regular intervals in the scene. They then decimate the set of synthetic views using a greedy set cover approach that finds a minimal subset

of views such that each view in the subset has at least 150 3D points in common with an original view. Cao and Snavely (2014) use a similar criterion, but instead of views, they decimate the set of points in an SfM point cloud used for localization. Instead of set cover, they use a probabilistic variant of the k-cover algorithm.

6.2.4 Semantic Annotation

The most common approach to perform *semantic annotation* of the discovered landmark clusters is by statistical analysis of user-provided image tags, titles and descriptions. In order to remove uninformative tags like "vacation", Quack et al. (2008) first apply a stoplist and then perform frequent itemset analysis (Agrawal et al., 1993) to generate candidate names. These names are verified by querying Wikipedia and matching images from retrieved articles against the landmark cluster. Zheng et al. (2009) also apply a stoplist and then simply use the most frequent n-gram in the cluster. Crandall et al. (2009) deal with uninformative tags in a more general way by dividing the number of occurrences of a tag in a cluster by its total number of occurrences in the dataset. Simon et al. (2007) additionally account for tags that are only used by individual users by computing a conditional probability for a cluster given a tag, marginalizing out the users.

Unfortunately, a much larger problem, also observed by Simon et al. (2007), exists for the task of semantic assignment that is much harder to fix: For most clusters accurate tags are simply not available. In our analysis (Sec. 6.7), we will show for which clusters these methods will still result in accurate descriptions and point out the sources of this problem.

6.3 Evaluation Setup

6.3.1 Dataset

Our evaluation is based on the Paris500k dataset (Sec. 3.3). In contrast to many other datasets (Sec. 6.2.1) the images were retrieved using a geographic bounding box query rather than keyword queries to ensure an unbiased distribution of touristic photos.

6.3.2 Query set, Categories and Evaluation

To collect realistic queries for the task of automatic annotation of photos uploaded to a photo sharing website, we downloaded 10k images from the same geographic region as Paris 500k from Flickr and ensured that they were no (near) duplicates of any image in the original dataset. Since we consider the task of *specific object recognition*, not *object categorization*, we filtered out unsuitable queries like food, pets, plants or cars. To only include objects that have a *chance* of being recognized based on the Paris 500k

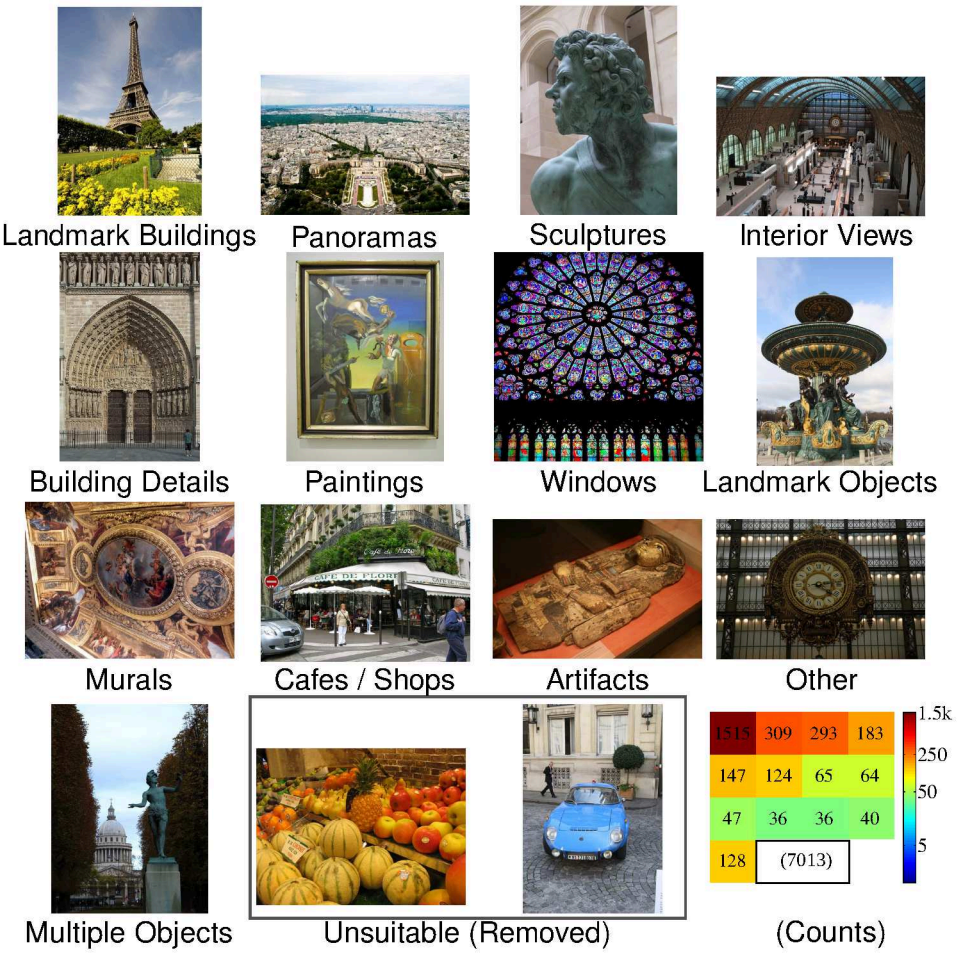


Figure 6.2: *Our set of query categories and the number of query images in each category (bottom right).*

dataset, we also exclude queries that do not match *any* image in Paris 500k, leaving 2,987 queries. We manually grouped the queries into the categories of Figure 6.2 in order to enable an analysis of the recognition performance for each query type. We summarize non-building objects such as bridges, fountains or columns under the *Landmark Objects* category. The *Artifacts* category contains historic objects such as sarcophagi or ancient tools. Objects that do not fit into any other category were categorized as *Other*. Note that there is a large variance in the number of query images for each category (given on the bottom right of Fig. 6.2). The average scores over all query images we provide in this paper therefore have a bias towards the larger categories. This effect is desired, since we want the query distribution to be representative of a real application in a photo auto-annotation system. In addition to this, however, we will also provide a detailed analysis for all 13 categories, focusing on four categories representative of different use cases, namely *Landmark Buildings*, *Paintings*, *Building Details* and *Windows*.

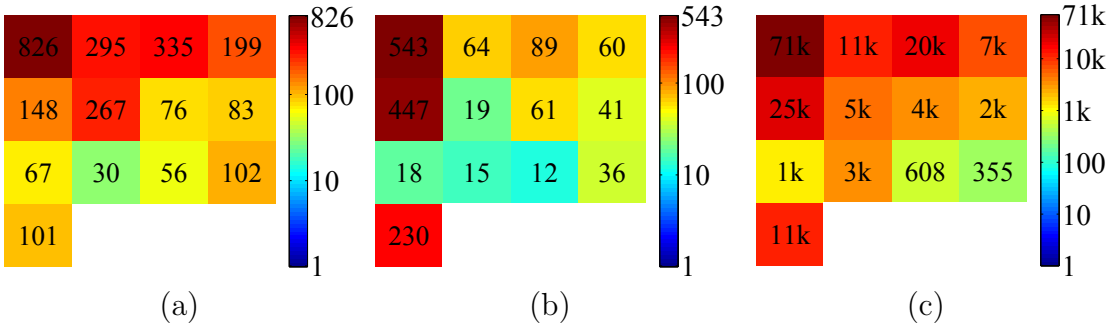


Figure 6.3: *(a) Number of Iconoid clusters in each category. (b) Average cluster size. (c) Total number of images in clusters. Categories are in the same order as in Fig. 6.2.*

6.3.3 Scoring

We would like to evaluate the performance of a landmark recognition system in a realistic scenario. In a photo auto-annotation application, the system should assign a user's photos reliable labels without supervision. A mobile visual search app like Google Goggles can also give the user a small selection of objects and let them pick the correct one. Therefore, we consider only the top-3 objects returned by the system.

For annotating recognition results, we showed the query and the iconic image of the recognized object to raters and asked them to rate the object's relevance to the query as "good" if it is the exact object in the query image, "ok", if it is somewhat relevant to the query, and "bad" if it is irrelevant. An object should be rated as "ok", e.g., if the query image shows a whole building, but the match only shows a detail of that building, or vice versa. In case the query is a detail of a building and the recognized object is a different detail of the same building, the match should be rated as "bad". If the query shows multiple landmarks, and the object is one of them, the match should still be rated as "good".

Based on this rating, we define four scores: *good-1* is the fraction of queries with a "good" top-1 match; *ok-1* is the fraction of queries with an "ok" or "good" top-1 match; *good-3* is the fraction of queries with a "good" match in the top-3; and *ok-3* is the fraction of queries with an "ok" or "good" match in the top-3.

6.3.4 Baseline Recognition Performance

For an estimate of the difficulty of the different query categories, we perform image retrieval (Sec.3.2.2) against the full Paris 500k dataset and manually rate the relevance of the top-3 images for each query according to the above scheme (Tab. 6.1). Note that these results only show the relevance of *retrieved images*, not *recognized objects*, but can serve as upper bounds for the recognition performance for each category. In total, the top-1 match was "good" for 92.74% and at least "ok" for 96.42% of the queries. Since

Category	%good-1	%ok-1	%good-3	%ok-3
Landmark Buildings	94.32	98.22	97.62	98.42
Panoramas	87.70	95.15	91.59	95.79
Sculptures	92.15	95.56	94.20	96.25
Interior Views	85.25	89.07	89.07	92.35
Building Details	87.76	91.84	89.80	91.84
Paintings	97.58	98.39	98.39	98.39
Windows	95.38	95.38	95.38	95.38
Landmark Objects	93.75	96.88	96.88	96.88
Murals	100.00	100.00	100.00	100.00
Cafes / Shops	80.56	80.56	83.33	83.33
Artifacts	91.67	91.67	94.44	94.44
Other	92.50	97.50	97.50	97.50
Multiple Objects	98.44	98.44	98.44	98.44
Total	92.74	96.42	95.58	96.92

Table 6.1: *Performance of plain image retrieval using the full dataset.*

images that did not have a match in the database are not used in the query set, the remaining 3.58% had only false-positive matches in the top-3.

6.4 Landmark Object Discovery

The first step of building a landmark recognition system is to cluster the image collection into *objects* (Sec. 6.1). A guiding question for our evaluation is: What object types can be discovered by such a clustering? As we motivated in Section 6.2.2, we choose Iconoid Shift (Ch. 4) as our analysis tool to answer this question, since it produces a set of overlapping clusters, which can represent "overlapping" objects, e.g., both the entire facade of Notre Dame and individual statue groups on it. In addition, it has intuitive parameters for controlling the granularity and number of discovered clusters by varying the kernel bandwidth β and the number of seeds, respectively.

6.4.1 Clusters Discovered per Category

To analyze what objects can be discovered by visual clustering, we run Iconoid Shift on the Paris 500k dataset. We choose a kernel bandwidth of $\beta = 0.9$, meaning that an image needs to have at least 10% overlap with an Iconoid to belong to its cluster. We perform several runs of the algorithm using different numbers of seed images selected randomly from Paris 500k in order to analyze the tradeoff between runtime and the number of objects discovered.

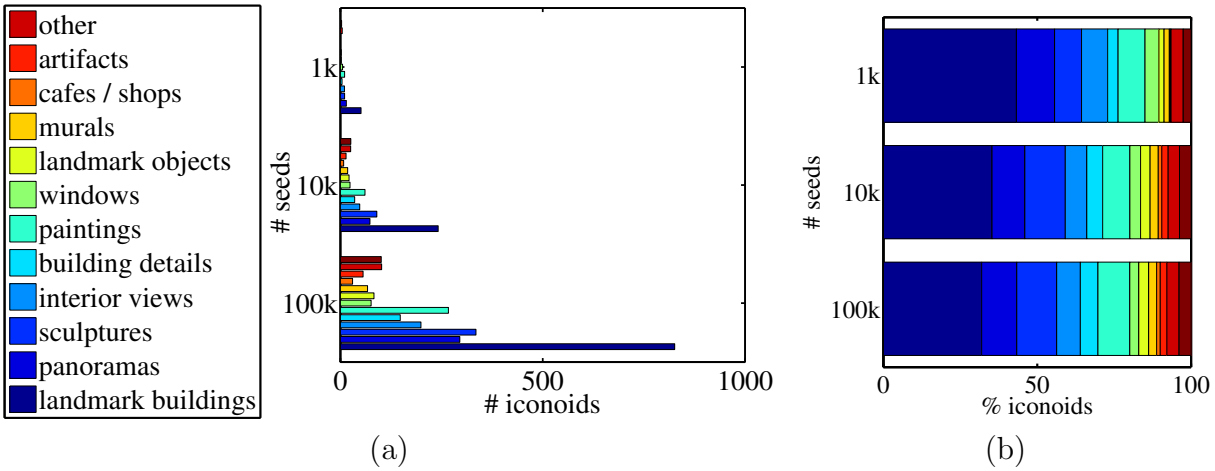


Figure 6.4: *Distribution of categories for different numbers of seeds counting only clusters of size 5 or larger. (a) absolute (b) relative.*

To examine what kinds of objects the algorithm finds, we categorize all resulting Iconoid clusters of at least size 5 using the scheme from Figure 6.2. Figure 6.3 shows the number of discovered clusters for each category, their average size and the number of images covered by clusters. *Landmark Buildings* are the largest category with 826 clusters covering 71k images. The average cluster size of *Building Detail* is surprisingly large, because some clusters include many photos of the full facades due to our low choice of overlap threshold for Iconoid Shift. *Painting* clusters are small on average, while *Windows* have fewer but larger clusters.

Figure 6.4 shows the effect of the number of seeds on the number of clusters discovered per category. When using only 1k or 10k seeds, the category distribution remains relatively constant. The share of *Landmark Building* clusters decreases with increasing number of seeds (Fig. 6.4b), since more of the smaller objects such as *Paintings* or *Sculptures* are discovered.

6.4.2 Distribution of Cluster Sizes and Performance Gap

The effect of the number of seeds on the distribution of cluster sizes is shown in Figure 6.5a. As reported by Gammeter et al. (2010), the cluster sizes are power law distributed. We can observe that the distribution shifts towards smaller clusters when more seeds are used. Figure 6.5b shows that the number of large clusters flattens out more quickly than the number of small clusters when increasing the number of seeds. This is because large landmark clusters are found first, but more seeds help find more obscure places and objects. When using 100k seeds, 12,776 Iconoids are returned, but only 3,088 of them contain 5 or more images.

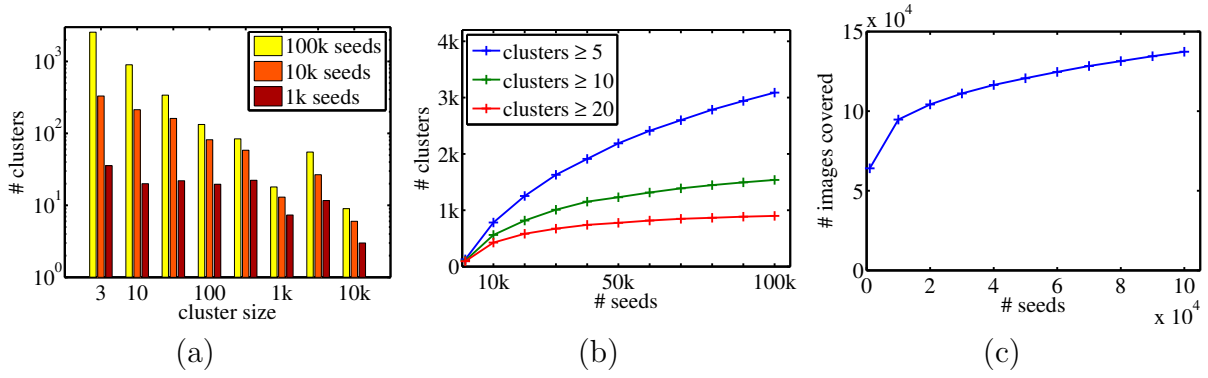


Figure 6.5: *(a) Cluster size distribution for different numbers of seeds (Note that both axes are logarithmic.) (b) Comparison of growth rates for different cluster sizes. (c) Total number of images covered by the clustering.*

Figure 6.5c shows the total number of database images covered for different numbers of seeds. Using 100k seeds, a total of 137,291 images (27.4% of the dataset) are covered by the clustering. The remaining images are either irrelevant or missed by the clustering. To estimate how many and which objects the clustering missed, we compare the retrieval performance using this reduced set of images to the full set (cf. Tab. 6.1). The result (Fig. 6.6) shows in which categories performance is lost and gives an upper bound on what a landmark recognition system can achieve based on this clustering. While for some categories, such as *Landmark Buildings* or *Windows*, the loss in performance is small, other categories like *Paintings*, *Landmark Objects* or *Cafes* show a strong decrease, because their clusters are small and thus more likely to be missed by the seeding process. This effect is the main cause of the total performance gap of 17.05% between the full database and the cluster images.

6.4.3 Discussion

Since often-photographed objects are discovered first, seed-based clustering can be computationally much more efficient than computing the whole matching graph (Sec. 3.4). However, to sufficiently cover seldom photographed objects such as museum exhibits, a large number of seeds is necessary. The coverage of such objects could also be increased by seeding strategies that avoid the bias to large clusters. Small object discovery approaches (Chum et al., 2009; Letessier et al., 2012) or approaches that crawl tourist guide websites (Zheng et al., 2009) might also help cover these objects better and close the above performance gap further. Whatever strategy is chosen, the results in Figure 6.6 show that such additional steps are necessary if *Landmark Objects*, *Cafes / Shops* or *Artifacts* shall be recognized. For most of the following experiments, we choose to use 100k seeds to ensure good coverage of details and small objects.

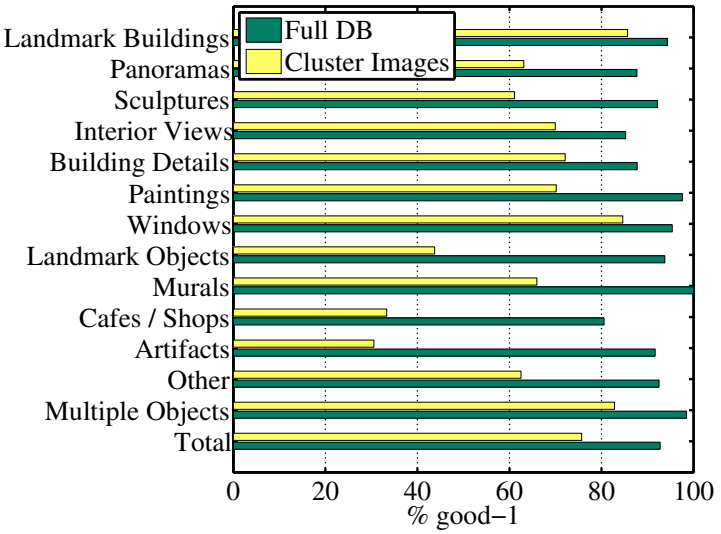


Figure 6.6: good-1 *retrieval performance for the full database vs. only the images discovered by Iconoid Shift using 100k seeds.*

6.5 Landmark Object Recognition

By clustering a large collection of tourist photos, we have discovered numerous interesting objects and set of representative images for each of them. To recognize a new object in a query image, a landmark recognition system performs retrieval in the set of discovered object representatives (Fig. 6.1). The open question here is: Given a ranking of representatives, how to rank the objects they belong to by their relevance to the query? To this end, we compare five object scoring methods and evaluate their respective tradeoffs of performance vs. database size and their suitability for different object categories.

6.5.1 Ground Truth Generation

For this evaluation, we introduce a new ground truth containing relevance ratings (Sec. 6.3.3) of the Iconoids discovered with 100k seeds w.r.t. the query set. An exhaustive relevance annotation of the 12,776 Iconoids for each of the 2,987 query images would require about 883 person-days of human work, assuming 2s of annotation effort per query-Iconoid pair. Therefore, we took two measures to reduce the amount of manual labor. (1) We summarized queries showing exactly the same view into 2,042 groups, since the same Iconoids are relevant for them. For this, we computed a pairwise matching of the queries and manually inspected each pair of matching images, discarding all pairs that do not show exactly the same view. We then constructed a matching graph from the verified edges and computed its connected components. During annotation, each group was represented by one image, and annotations for it were transferred to

all other members of the group. (2) We automatically rated an Iconoid as irrelevant for a group of queries if *none* of the Iconoid's representatives were spatially verified *at least once* when querying an image retrieval system with *each* query in the group. Since the landmark with the largest number of images in the Paris 500k dataset is the Eiffel Tower with about 20k images, any query can have at most 20k relevant images. To leave some room for ranking errors, we performed spatial verification for the top-30k retrieved images ranked w.r.t. their tf-idf scores. The 26.8k remaining pairs of query groups and Iconoids were manually annotated according to the rating scheme introduced in Section 6.3.3. Annotations were performed by 28 people over a period of 8 weeks. Each pair was shown to 3 people who were asked to rate it as *good*, *ok* or *bad* according to the scoring scheme we introduced in Section 6.3.3, and the final annotation was decided by majority voting. In the 1.8k cases where all three annotations were inconsistent, the image pair was passed to a fourth annotation for a definite annotation[1].

6.5.2 Methods

We evaluate five object scoring methods:

- **Center** represents objects only by the cluster center (i.e., Iconoid) of each object and discards all other representatives, as done in Yang et al. (2011). The object ranking is then simply the same as the representative ranking.
- **Size** returns all objects with at least one matching representative, and scores them by their cluster size, i.e., by the number of times they were photographed.
- **Voting** lets each matching representative cast a vote for each object it belongs to (note that clusters can overlap). Objects are then ranked by their number of votes.
- **Best Match** returns the object with the highest scoring representative, as done in, e.g., Gammeter et al. (2009, 2010); Quack et al. (2008); Zheng et al. (2009). A difference in our case is that we are using a soft clustering, so a representative can belong to multiple objects. In this case, we return the object with the largest cluster size. This method can therefore also be viewed as a variant of the *Size* method that only uses the best matching representative.
- **Overlap** uses *Homography Overlap Propagation* (Sec. 4.5.2) to compute the overlap of the query with each Iconoid. The method first computes the overlap region of the query with the matching representative and then propagates this region into the Iconoid via the shortest path in the Iconoid cluster's matching graph. This is done using homography overlap propagation. If a query matches multiple representatives of the same Iconoid, the overlap is computed for each of them and

[1]The ground truth, including including the query images, the Iconoid clusters, the query-Iconoid relevance annotations, and the query category annotations is available at http://www.vision.rwth-aachen.de/data/paris500k/paris-dataset.

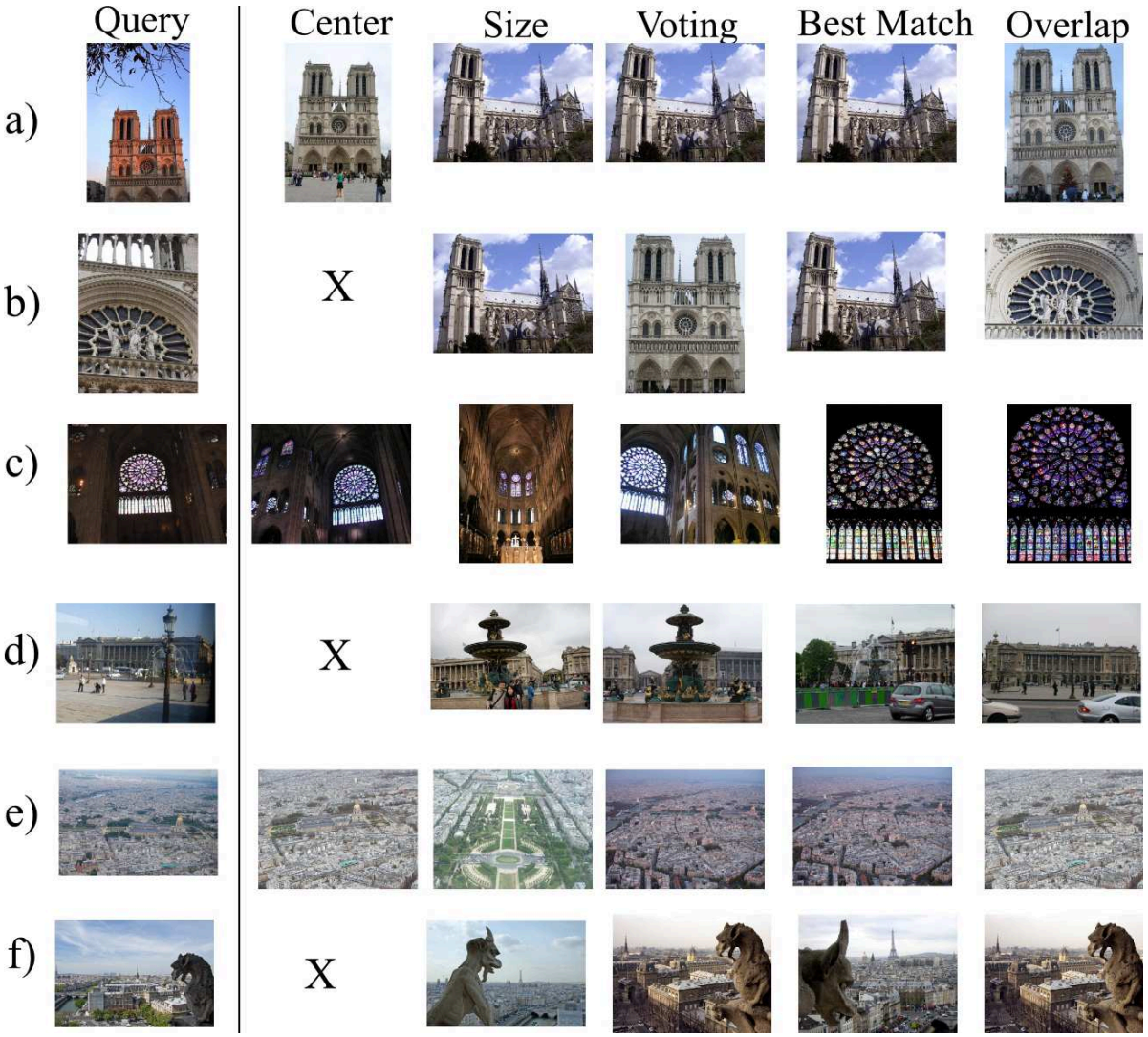


Figure 6.7: *Top-scoring objects for different object scoring methods. "X" means that no objects were recognized.*

the largest overlap is used. Objects are then ranked in decreasing order of their Iconoids' overlaps with the query.

We now first compare these methods using the objects discovered with 100k Iconoid Shift seeds and then show the effect of the number of seeds on their performance.

6.5.3 Results

We evaluated the five approaches based on the ground truth introduced in Section 6.5.1. To estimate the error introduced by the image retrieval pre-filter we used to reduce

	% good-1	% ok-1	% good-3	% ok-3
Centers	39.60	45.56	42.99	46.03
Size	57.11	73.32	66.42	76.26
Voting	59.42	**76.00**	69.80	**77.64**
Best Match	60.40	75.39	67.26	77.10
Overlap	**63.71**	75.93	**71.78**	77.13

Table 6.2: *Performance of different object scoring methods.*

annotation effort (Sec. 6.5.1), we performed a small control experiment. We manually rated the relevance of the top-3 Iconoids retrieved for each query using the *Voting* method (Sec. 6.5.2) and found that 0.7% of the Iconoids rated "good" or "ok" were filtered out. We believe this small false-negative rate is still acceptable, since the simplified annotation procedure significantly reduced the amount of manual labor.

The recognition performance of the different methods is compared in Table 6.2. Figure 6.7 shows the top scoring objects for typical queries. *Center* finds images closely resembling the query, but often fails to find *any* matching objects, because the cluster centers are not sufficient to recognize all objects under different viewing conditions due to the limited invariance of the matching process. However, since it only requires one image per object, it is by far the fastest and most memory efficient method. Because *Size* chooses the largest cluster, it often finds a viewpoint more popular than the query (popular landmarks are often represented by multiple clusters from different viewpoints). Sometimes this effect is desired since the largest cluster usually depicts the object best, but it can also cause drift (Fig. 6.7b-f), i.e., instead of the query object, a nearby object is recognized. *Voting* finds the clusters with the most matching representatives. This makes it less prone to drift (Fig. 6.7b,e,f) and causes it to achieve higher performance than *Size*. Despite being simpler than *Size*, *Best Match* outperforms it. The reason is that *Best Match* considers only the closest matching representative to the query, making it less prone to drift than *Size* that also looks at farther away matches (Fig. 6.7c,e). *Overlap* has the best *good-1* performance, because it computes the actual overlap of the query image with each cluster's iconic image and selects the iconic whose view is closest to the query (Fig. 6.7a-f).

The *good-1* performances by query type are shown in Figure 6.8. *Size* compares well to the other methods on *Landmark Buildings* and *Cafes / Shops*, where the largest cluster is often the correct one (e.g., the full view of a facade). On *Paintings*, all methods including *Center* have similar performance, because *Paintings* are flat objects and are usually photographed under the same viewing conditions. Since this makes painting retrieval very easy, multiple representatives do not bring an advantage, explaining the relatively good performance of *Center*. *Size* performs worse than *Voting* and *Best Match* on *Panoramas* (Fig. 6.7e), because it tends to drift to more popular nearby views, moving the query object out of the field of view. The same effect occurs on *Building Details*

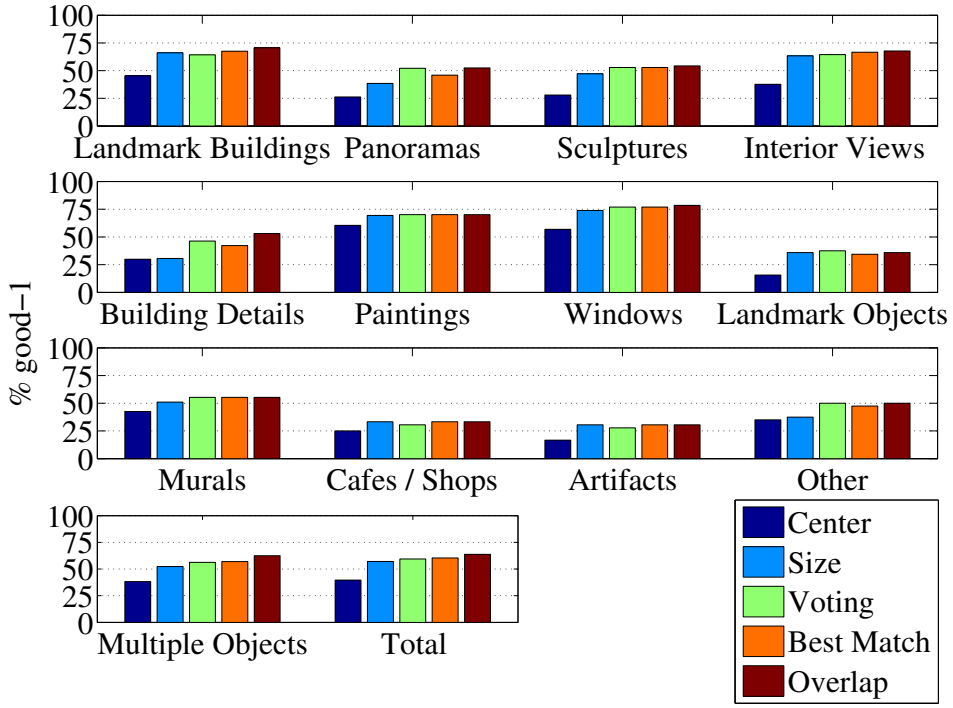


Figure 6.8: *Performance of object scoring methods by query type.*

(Fig. 6.7b), where *Size* tends to return views of the whole building instead. *Overlap* has a particular advantage on classes where other methods tend to drift (*Panoramas*, *Building Details*, *Multiple Objects*), since it usually finds the iconic image that best matches the photographed part (Fig. 6.7a-f). *Center* works relatively well for *Windows* and *Murals*, because, like *Paintings*, they are only photographed from a limited range of viewing angles, making them easy to match.

6.5.4 Effect of the Number of Seeds

We now analyze the effect of the number of object clusters (which depends on the number of seeds) on the performance of the five methods and compare object retrieval performance by category. As Figure 6.9a shows, the performance of the methods does not differ much for 1k and 10k seeds, except that *Center* consistently performs worst for the reasons explained above. The differences become slightly more pronounced at 100k seeds, where the density of objects is very high and methods that are less prone to drift to nearby objects gain an advantage. Conversely, this shows that simpler methods are sufficient if the object density is low. Figure 6.9b shows the performance for different query types when using the *Voting* method. *Sculptures*, *Paintings*, *Windows* and *Murals* show the steepest improvement since they require more seeds to be sufficiently covered by the clustering, while *Landmark Buildings* can already be recognized when using a smaller

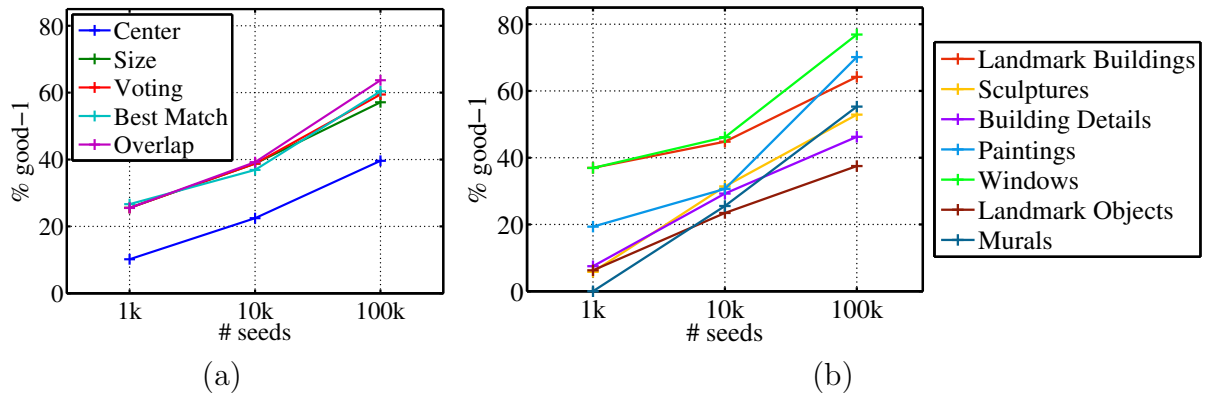


Figure 6.9: *(a) good-1 performance of object scoring methods for different numbers of Iconoid Shift seeds. (b) good-1 performance by query type using the* Voting *method.*

number of seeds since they form large clusters that are discovered early. Surprisingly, *Windows* have the highest recognition rate overall. The reasons for this are that (i) they are easy to recognize since they are flat, highly textured objects, and (ii) they get discovered already with few seeds, since *Window* clusters are almost three times the size of *Painting* clusters (Fig. 6.3b).

6.5.5 Discussion

The ideal choice of method varies by application: *Center* provides high performance for flat objects at low computational and memory cost. *Voting* has high accuracy across all categories and its speed makes it applicable, e.g., for mobile visual search. The popular *Best Match* method has similar performance and efficiency, but is outperformed by *Overlap*, which has the highest performance overall but also the highest computational cost. *Overlap* is therefore better suited for offline applications requiring high accuracy, e.g., photo auto-annotation.

Even when looking at the best performing method, *Overlap*, there is still a difference of 11.98 percent points between the *good-1* performance of object retrieval (63.71%) and the *good-1* performance of 75.69% of plain image retrieval (Fig. 6.6). The cause for this *clustering gap* could be either a too coarse clustering or imprecise object ranking. If we consider that the difference between *good-1* and *good-3* for *Overlap* (8.07%) is larger than the gap of 3.91 percent points between the *good-3* performance of *Overlap* and the *good-1* performance of plain image retrieval, it becomes apparent that the main part of the clustering gap is due to the object ranking. Hence, there is still room for improved object ranking methods to close this gap.

Figure 6.9a shows diminishing returns in performance when the number of clusters increases (note the logarithmic x-axis), since the most popular queries are covered first and the long tail of queries requires exponentially more effort.

The performance gap between *Center* and other methods shows that the representatives are necessary for ensuring invariance to different viewing conditions. However, it is desirable to reduce the set of representatives, since it determines the memory use and speed of the retrieval index. This is examined more closely in the following section.

6.6 Efficient Representations for Retrieval

Since the discovered landmark representatives are highly redundant, subsampling them can save memory and computation time. The goal here is to reduce the set of representatives in a way that still preserves as much visual variability as possible in order to ensure good retrieval performance. The methods we present in the following work by summarizing groups of similar images, as in e.g., Avrithis et al. (2010); Gammeter et al. (2010), which can be done efficiently by exploiting the similarity information that is already available from the *matching graph* constructed during clustering (Sec. 3.2.3). We evaluate four approaches and compare them against a random baseline. We use the number of homography inliers as edge weights for the matching graph (as done in, e.g., Avrithis et al. (2010); Gammeter et al. (2010); Quack et al. (2008)).

6.6.1 Methods

We compare the following five methods for reducing redundancy in the sets of representatives:

- **Complete-Link** (Gammeter et al., 2010) performs hierarchical agglomerative clustering and replaces each complete link component containing at least 3 images by its image with the most neighbors.
- **Kernel Vector Quantization (KVQ)** (Tipping and Schölkopf, 2001) is a clustering method that selects a minimum number of points such that each point in the dataset is within radius r of at least one selected point. It is used by Avrithis et al. (2010) to reduce the set of features in a cluster after projecting features from all images into a single iconic image, called *Scene Map*. Applying the same idea on the image level, we use it to find a minimal subset of representative images such that each image in a cluster has a given minimum matching score with at least one image in the subset.
- **Dominating Set** chooses a subsample such that each representative is adjacent to at least one image in the original cluster. This subset is found by solving the corresponding set cover problem using the greedy set cover algorithm (Johnson, 1974).

- **Fine Iconoids** performs a second, finer Iconoid Shift clustering at bandwidth $\beta = 0.7$ that covers the image collection at a very fine granularity. The representatives for each (coarse) cluster are then chosen to be all the fine Iconoids in it. This is similar to the approach of Raguram et al. (2011) that represents objects by a set of iconic images found by clustering GIST descriptors.
- **Random** is the baseline method that simply draws a random subsample of the representative images.

6.6.2 Results

We now compare the tradeoffs between the number of representatives and recognition performance of these methods using the *Voting* method (Sec. 6.5.2) for object scoring (Fig. 6.10 and Fig. 6.11). The number of representatives that the *Complete-Link*, *KVQ* and *Dominating Set* methods return can be controlled by first deleting edges below a certain edge weight threshold to make the matching graph sparser and then running the algorithm. We generated 3 sets of representatives with each method by applying thresholds of 15, 30 and 50 inliers.

As reported in Gammeter et al. (2010), *Complete Link* only slightly reduces the representative set while maintaining high recognition performance. *Dominating Set* and *KVQ* yield comparable results since they optimize similar criteria. They represent a good tradeoff, allowing for a reduction to about 40% while still achieving a 60.9% *good-1* performance (at threshold 50). An interesting result is that *Fine Iconoids* performs well on *Paintings* and *Building Details*, but lower than *Random* on *Landmark Buildings*. Visual inspection showed that Iconoids do not cover the more obscure viewing conditions necessary for robust recognition, since the algorithm is designed to converge to popular views. It therefore performs higher on categories with a limited range of possible viewing conditions.

6.6.3 Discussion

In this evaluation, we focused on reduction methods that work on the *image level*. We have shown that in particular *KVQ* and *Dominating Set* methods can achieve high compression at only a small loss in precision. Some recent approaches also perform this reduction on the feature level, usually combined with *offline query expansion*, i.e., projecting features into matching images, which is reported to even improve precision over baseline retrieval (Avrithis et al., 2010; Turcot and Lowe, 2009). An evaluation of these methods would be an interesting task for future work.

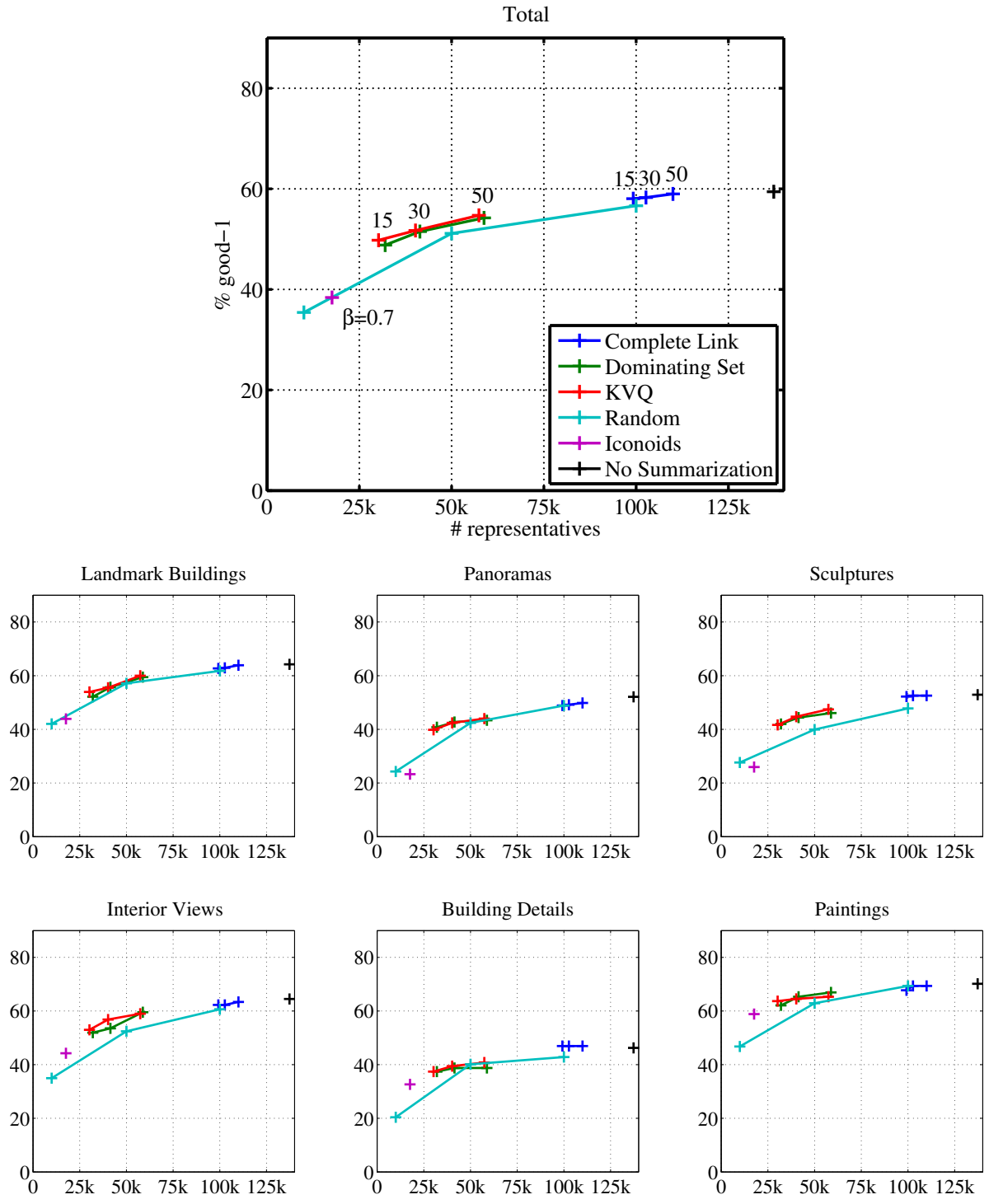


Figure 6.10: *Performance / index size tradeoff of cluster summarization methods using edge weight thresholds of 15, 30 and 50. (Part 1)*

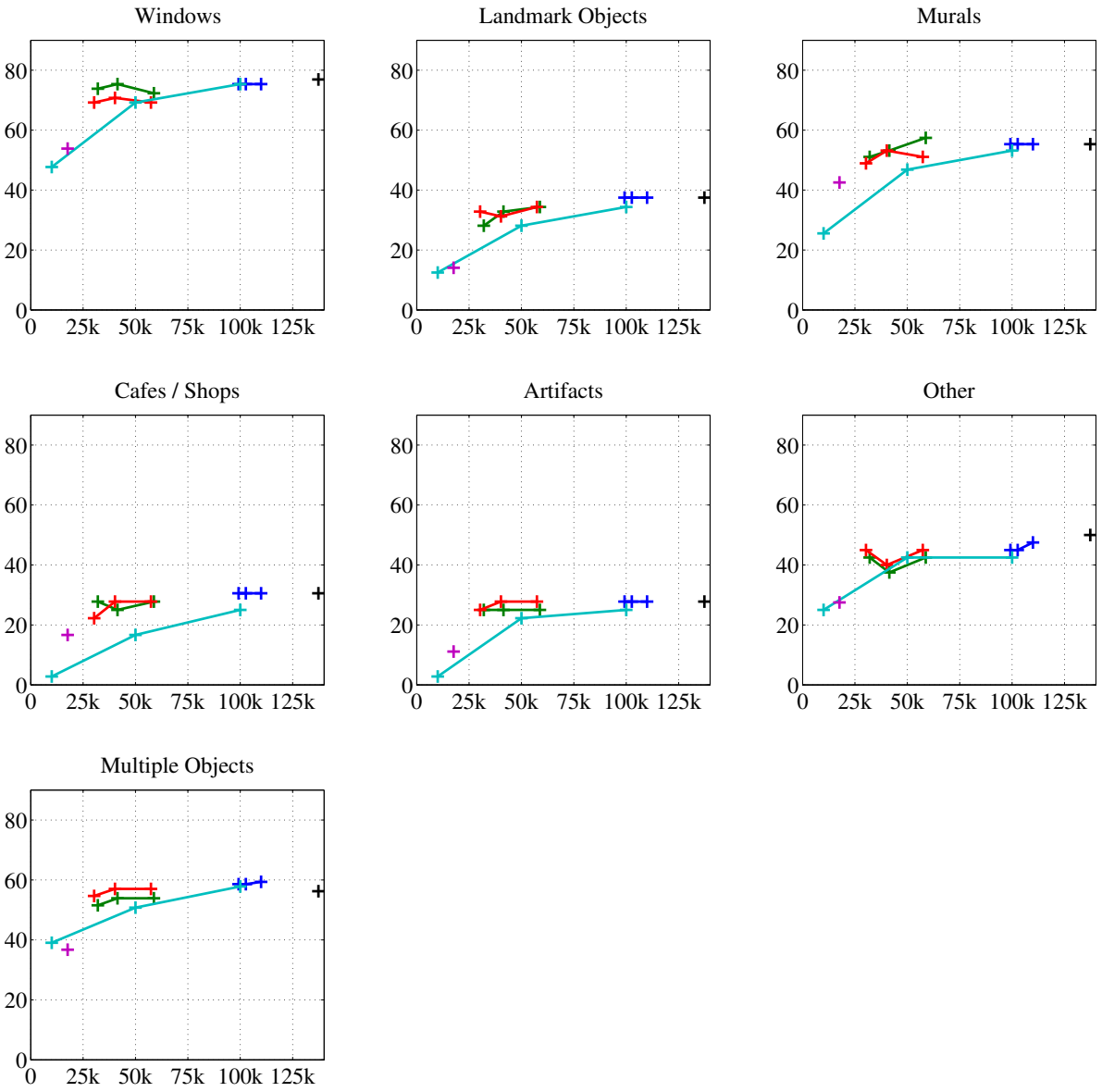


Figure 6.11: *Performance / index size tradeoff of cluster summarization methods using edge weight thresholds of 15, 30 and 50. (Part 2)*

6.7 Interfacing Images with Semantics

To find suitable descriptions for the discovered objects, and thus to enable linking them with information on the web, the usual method is to perform statistical analysis of the tags and titles that users provided for the photos in a cluster (Crandall et al., 2009; Quack et al., 2008; Simon et al., 2007; Zheng et al., 2009). Quack et al. (2008) mine *frequent itemsets* (Agrawal et al., 1993) in all tags of a cluster to generate candidate names. The top-15 candidates are then used to query Wikipedia, and the retrieved articles are verified by matching the images occurring in them against the images in the cluster. In a small informal experiment we found that frequent itemsets returns many noisy and non-descriptive names like "vacation", "photo", "canon", or "europe", which need to be filtered by a comprehensive stoplist. Furthermore, tags that are frequently used by the same user like "summer vacation 2008" can be ranked higher than correct but less frequent terms.

The method of Simon et al. (2007) is specifically designed to handle both of these problems. It probabilistically computes a score $score(c, t)$ for each pair of tag t and cluster c. This score is based on the conditional probability of cluster c given tag t, resulting in tags that mainly occur in cluster c. By marginalizing over the users, tags that are frequent in the cluster, but used by only few users are ranked low. We use the method of Simon et al. (2007) in our evaluation since we found that it yields more reliable tags than the method of Quack et al. (2008). We analyze for which objects we can reliably find semantics and examine the performance gap between object retrieval and semantic annotation.

6.7.1 Data Preparation

We first need to define a set of tags for each image based on the metadata provided by the respective photo sharing website. The Paris 500k dataset consists of images from Flickr and Panoramio. Images on Flickr have both tags and a title, so we treated the title as an additional tag. Since images on Panoramio have no tags, we used the titles as their sole tag. We preprocessed image tags by applying a very small stoplist containing terms such as "Paris" and "France" and removing filenames like "DSC002342.JPG".

6.7.2 Tag Quality Annotation

In order to analyze tag quality, we manually rated the quality of the top-3 tags for each object cluster containing 6 or more images. On average, annotation of a single object-tag pair took about 30-60 seconds, since often a web search was necessary to verify the correctness of a tag. This annotation was performed by six people who annotated a total of 2,536 objects. The annotators were asked to rate each image-tag pair as "good" if the tag accurately describes what is visible in the iconic image (e.g., the full name of a building, the title and painter of a painting) and as "ok" if it provides at least some

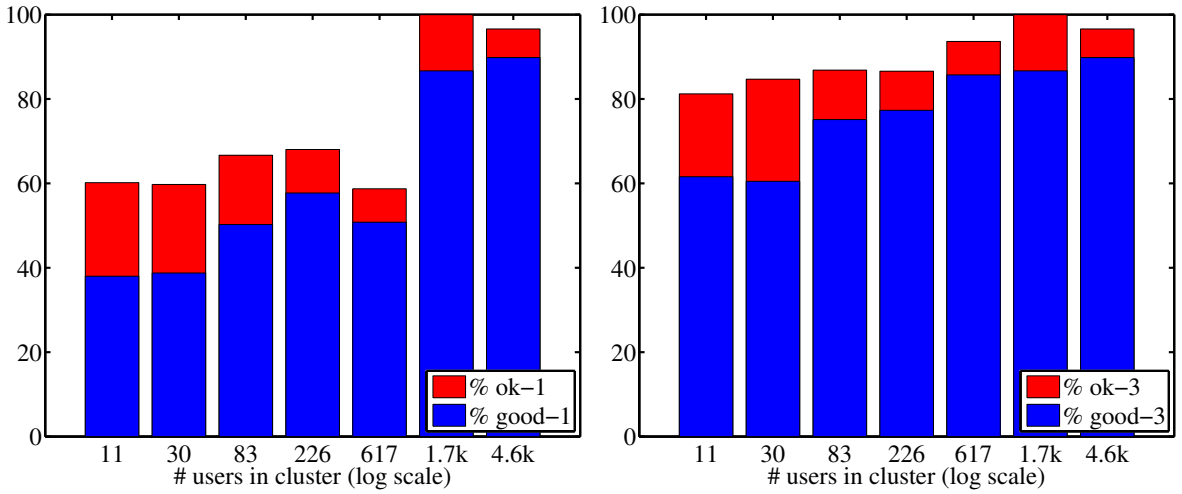


Figure 6.12: *Tag quality as a function of the number of individual users in the cluster. Left: top-1 tag, right: top-3 tags*

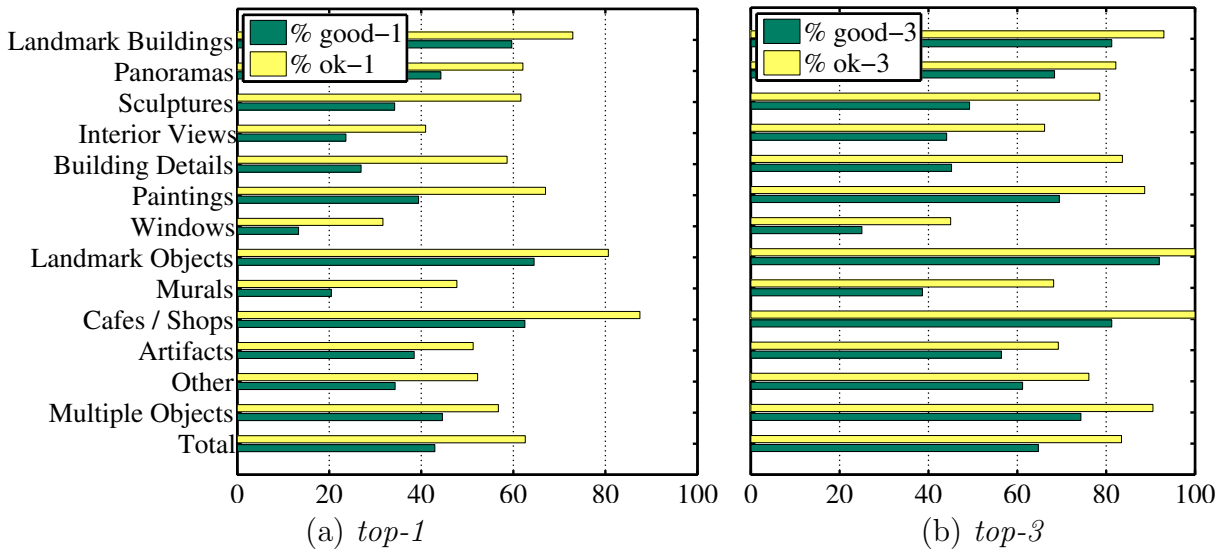


Figure 6.13: *Tag quality of different object categories for Iconoid clusters of size 6 and higher.*

helpful information, such as the creator of a sculpture, but not its name, or if the tag is accurate, but contains noise terms like "me in front of Notre Dame". This annotation allows us to re-use the evaluation measures we used for object retrieval in Sec. 6.5.

museedorsay	musee d' orsay	musée-d'orsay	museè d'orsay	museo d'orsay
muse´e d'orsay	museed'orsay	muséed'orsay	musee d'orsee	museo orsay
musee d´orsay	muséedorsay	musÉe d'orsay	musee d'orsey	museu de orsay
musee d´ orsay	musée dorsay	musèe d orsay	museed'orsey	museu d´orsay
musee d`orsay	musée d´orsay	musèe d'orsay	musee orsay	museu d'orsay
musee d orsay	musée d`orsay	musÈe d'orsay	musée orsay	museu d`orsey
musee d 'orsay	musée d'orsay	museé d'orsay	museo orsay	museumdorsay
musee d'orsay	musée d’orsay	museéd'orsay	museo d´orsay	museum d´orsay

Figure 6.14: *Different spellings of "Musée d'Orsay" encountered in the dataset.*

6.7.3 Tag Mining

We first analyze the influence of the number of users contributing photos to a cluster on the reliability of automatic semantic annotation. The method of Simon et al. (2007) outputs a ranking of potential names for each cluster, allowing us to examine the accuracy of the top-1 and the top-3 tags. Figure 6.12 shows that tag reliability clearly increases with more users. In particular, over 80% of clusters with over 1k users have a *good* top-1 tag. With increasing user count, descriptions also become more precise, since the fraction of *ok* tags decreases. Figure 6.13 shows the tag quality for different categories. The tags determined for *Landmark Buildings*, *Landmark Objects* and *Cafes / Shops* are most reliable, since their names are typically well-known. *Cafes / Shops* are particularly easy to tag since their name is usually directly visible. *Murals*, *Windows*, *Sculptures* and *Building Details* are usually lacking proper annotations since photographers often do not know their names and only label them with generic tags. For example, *Building Details* are often tagged with the name of the entire building. This causes the large difference in the *good-1* and *ok-1* scores of these categories.

6.7.4 Discussion

While for most large clusters suitable semantics can be found, for small and medium sized clusters the difference in ok-1 and good-1 scores (Fig. 6.12) suggests that better tag ranking could greatly help the recognition of less popular objects. Significant improvements can likely be made by increasing robustness w.r.t. different languages, spelling errors and tag noise. For example, Figure 6.14 shows a selection of different spellings of "Musée d'Orsay" from our dataset that would all be treated as separate tags by current methods. Mining Wikipedia (as done in Quack et al. (2008)) or tourist guide websites (as done in Zheng et al. (2009)), or performing specialized per-category metadata mining (as done in Arandjelović and Zisserman (2012a) for sculptures) might also help in naming the less popular objects.

6.8 End-to-End Performance

In Section 6.5, we analyzed different methods for assigning *objects* to queries and measured accuracy on a visual level. In this section, we perform this analysis on a *semantic* level, based on the labels assigned in the previous section.

6.8.1 Setup

To evaluate the system from end to end, we cluster the Paris 500k dataset with *Iconoid Shift* using 100k seeds and mine semantics for the clusters using the method of Simon et al. (2007). We use the *Voting* object scoring method (Sec. 6.5) to rank objects w.r.t. a query. Subsampling (Sec. 6.6) is not used. We then rate the relevance of the top-scoring tag of the top-scoring object for each query as either *good*, *ok* or *bad* (Sec. 6.7.2).

6.8.2 Results

Figure 6.15 shows the results of semantic annotation and compares it to the plain object recognition performance (yellow bars in Fig. 6.8). *Landmark Buildings* have the highest performance, because their large cluster size enables robust recognition and semantic assignment. The reason semantic assignment has even higher performance than plain object recognition is that semantic assignment sometimes corrects errors of object recognition: It often happens that the query image shows the whole building, but the matching object is a detail of the building, e.g., a door of Notre Dame or a leg of the Eiffel Tower. This occurs, e.g., because the detail is most prominent in the query due to perspective, or because large parts of the building are occluded, but the detail is still visible. However, photos of building details are often labeled with the name of the whole building, since photographers do not know, e.g., the name of a particular door of Notre Dame. Therefore, the query is correctly assigned the name of the whole building even though the recognized object was a detail.

For categories with smaller clusters, there are different bottlenecks: *Artifacts* and *Landmark Objects* are hard to recognize, because they are compact objects and form only small clusters. However, they typically have high quality tags (Fig. 6.13). In other cases, a relevant object can reliably be retrieved, but low tag quality prevents successful semantic annotation. This problem is most prominent for *Windows* and *Murals* (Fig. 6.15). They are flat and photographed under a limited range of conditions, making them easy to recognize, but they suffer from low quality semantics, since information on them is not easily available. In total, 51.8% of the queries could be assigned good semantics, while for 59.4% a good object was retrieved; 61.9% of queries were assigned ok tags, while for 76.0% an ok object was retrieved. In the following, we summarize the causes of this gap and discuss possible solutions.

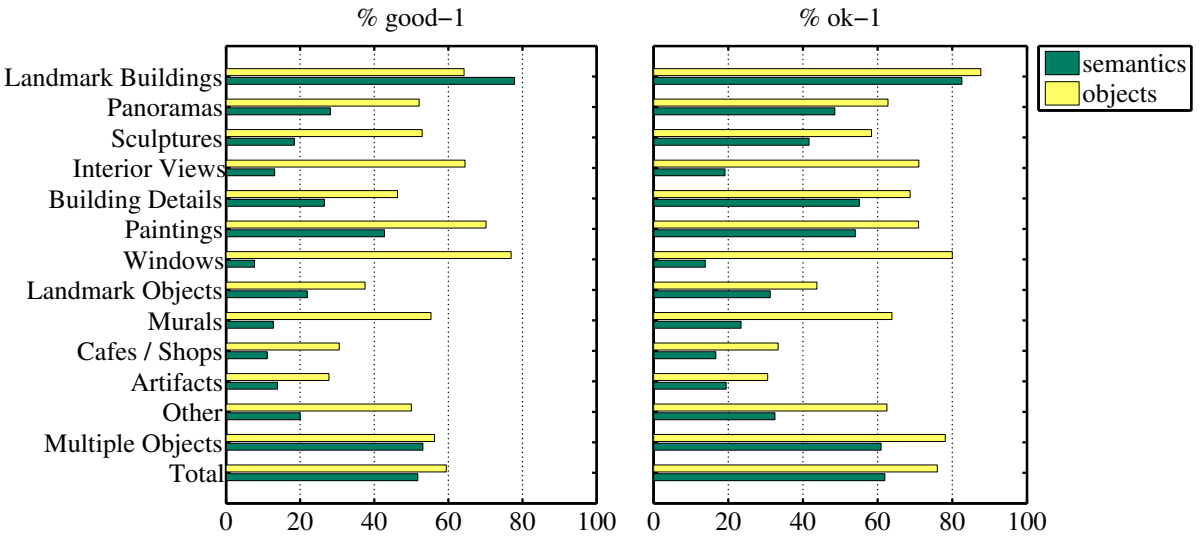


Figure 6.15: *Quality of* good *(left) and* ok *(right) semantics assigned to queries, compared to the quality of the objects retrieved.*

6.9 Discussion and Conclusion

We now sum up the findings of our evaluation by answering the questions posed in the introduction and discuss the areas where improvements can still be made.

How many and what kinds of objects are present in Internet photo collections and what is the difficulty of discovering objects of different object categories?

The question how many objects there are cannot be answered based directly on the number of clusters, because there can be multiple clusters of the same object showing different views, and clusters of non-objects, e.g., party photos, pictures of animals and food, or photo bursts. We performed an annotation experiment (Sec. 6.4.1) and labeled the 3,088 clusters containing five or more images discovered in the Paris500k dataset. 2,585 (83.7%) clusters were labeled as objects and 503 (16.3%) were labeled as non-objects (Note that this ratio will shift more strongly towards non-objects when also considering clusters containing less than 5 images.) Approaches for detecting and removing such non-object clusters could be a direction for future work, since they unnecessarily increase the size of the retrieval index and increase the chance of recognition errors. The distribution of object categories (Fig. 6.3a) shows that, not surprisingly, *Landmark Buildings* form the largest category, followed by *Sculptures*, *Panoramas* and *Paintings*. Seed-based object discovery algorithms (Chum and Matas, 2010; Chum et al., 2009; Weyand and Leibe, 2011) find the most photographed objects first, because when

drawing a random image from an Internet photo collection, the likelihood of drawing an often photographed object is higher. Therefore, much more effort is required to also discover objects in the long tail of the size distribution. This could be addressed in future work, e.g., by seeding methods that avoid the bias to large clusters, or methods that explicitly mine for small objects (Chum et al., 2009; Letessier et al., 2012).

How to decide which landmark was recognized given a list of retrieved images?

We analyzed five methods for this task (Sec. 6.5). Our experiments clearly showed that having a set of representative images for each cluster is necessary to recognize it under difficult conditions such as extreme viewpoint changes, occlusion, lighting changes, blur, etc. (first two rows of Fig. 6.16). This can be viewed as a form of offline query expansion. However, like query expansion, this method is also prone to *drift* (Fig. 6.17, top row), which can cause confusion between nearby objects. While the often-used *Best Match* method (Gammeter et al., 2009, 2010; Quack et al., 2008; Zheng et al., 2009) avoids drift better than some other methods, we found that by using a method that explicitly maximizes the *Homography Overlap* between the query and the object's iconic image, even higher precision can be achieved. However, the ranking gap (*good-1* vs. *good-3* performance) remains relatively large, suggesting that there is potential for more accurate object ranking methods.

How to efficiently represent the discovered objects in memory for recognition?

We analyzed four *image level* techniques for eliminating redundancy in the database (Sec. 6.6). Our analysis revealed that it is important to keep representatives showing obscure views and extreme lighting conditions. Therefore, representing the object by a set of popular views (as done in our *Fine Iconoids* method or in Raguram et al. (2011)) did not perform better than random subsampling. We proposed two methods that achieve an acceptable tradeoff between database size and recognition performance, but observed that there is still potential for better methods. A comparative analysis of methods that eliminate redundancy at the *feature level* (Avrithis et al., 2010; Gammeter et al., 2009; Turcot and Lowe, 2009) would also be an interesting future direction.

Are the user-provided tags reliable enough for determining accurate object names?

The largest room for improvement of the overall performance of landmark recognition is in semantic annotation. We determined two reasons for the performance loss at this step (Sec. 6.7): (1) User-provided tags come in different languages, have spelling errors

Figure 6.16: *Examples of successful recognition and annotation.*

and contain noise terms (Fig. 6.14). Methods robust to these factors could provide more reliable semantics even for small clusters. (2) Often, insufficient information is available to photographers, causing non-descriptive tags (Fig. 6.17, bottom right). This might be addressed by crawling relevant encyclopedia (Quack et al., 2008) or tourist guide articles (Zheng et al., 2009) or using image search engines (Arandjelović and Zisserman, 2012a). However, sometimes, the presence of accurate tags in small clusters can also lead to surprisingly accurate results (Fig. 6.16, rows 3 and 4).

What are the factors effectively limiting the recognition of different landmark types?

Our end-to-end analysis (Sec. 6.8) showed that the factors that limit recognition performance are quite varied and strongly depend on the object category. Some objects like *Windows* or *Murals* are easy to recognize *visually*, but often lack accurate tags, which

prevents semantic assignment. This could be addressed using the methods mentioned above. Other objects such as *Artifacts* or *Landmark Objects* do have accurate tags, but are harder to recognize due to their spatial structure and small cluster size. Their recognition could be improved by mining more photos of them from the web (Gammeter et al., 2010). Finally, improvements to image retrieval and matching, e.g., improved feature representation (Arandjelović and Zisserman, 2012b), improved feature quantization (Jégou et al., 2008, 2011), or ranking methods robust to problems like repeated patterns (Jégou et al., 2009a) (Fig. 6.17, bottom left) or watermarks, timestamps and frames (Ch. 7)), will directly benefit both landmark clustering and recognition.

6.9.1 Limitations

While our evaluation has brought to light several opportunities for progress, its scope could still be broadened in future work. Due to our choice of dataset our taxonomy of queries is certainly biased towards the landmarks of Paris. A larger dataset from several cities would increase the generality of the evaluation. Our query set was collected from Internet photo collections and is therefore representative for the task of photo auto-annotation. While this bias only affects the score average and not the per-category scores, a second query set for the task of mobile visual search would make it possible to identify problems specific for that task. The set of methods we analyzed was carefully chosen, but an analysis of other approaches (e.g., for clustering or semantic annotation) may bring further insights into how the component choices affect overall performance.

6.9.2 Conclusion

In this chapter, we have evaluated the automatic construction of visual landmark recognition engines from Internet image collections. We used our large-scale Paris 500k dataset of 500k photos from Paris, collected a set of 3k typical query images, and created a ground truth for evaluating large-scale photo auto-annotation. For each component of the pipeline, we evaluated how different methods and parameters affect overall performance as well as the performance for individual query categories. We proposed several novel methods for various sub-tasks, some of which outperform literature approaches. In our analysis, we have identified areas where such a system performs well, as well as areas where improvement is still possible.

Figure 6.17: *Examples of failure modes. Top row: Drift due to dominance of nearby objects. Bottom left: Failed retrieval caused by repeating patterns. Bottom right: Too generic description.*

7

Detecting Watermarks, Timestamps and Frames

As a last contribution of this thesis, we now consider a problem that we have encountered several times throughout our work and for which no satisfactory solutions exist so far. We found that the main cause of false-positive image matches we encountered were different types of artificial overlays that users have added to their photos (see Fig. 7.1). For example, many amateur and professional photographers add visible *watermarks* in the shape of logos or signatures to their photos for copyright reasons. Another problem are *frames*, which are becoming increasingly common because of smartphone camera apps like Instagram or Snapseed that allow the user to add photo frames or worn-out border effects to their photos to simulate a vintage look. Moreover, despite the availability of date and time information in EXIF tags, many users set their cameras to embed a visible *timestamp* in the image's pixel data itself.

These watermarks, timestamps and frames, or *WTFs* for short, particularly affect local feature based image retrieval and matching methods (Sec. 3.1), because a small geometrically consistent region is sufficient to form a match. If, for example, the same frame is added to two images, it will likely cause the same set of interest regions to be detected in both images. These interest regions are probably going to have very similar SIFT descriptors which will form geometrically consistent correspondences and thus cause the images to match. This was also reported by Müller et al. (2005) who observed that logos and frames in X-ray images are a frequent cause of false-positive matches in medical image retrieval. In Iconoid Shift (Ch. 4), frames are particularly problematic, because they will cause the overlap, which is estimated using a bounding box around the inliers (Fig. 4.4), to be almost 100%.

Furthermore, WTFs affect image clustering and landmark mining approaches (Sec. 2.2) based on matching graphs (Sec. 3.2.3). A WTF can cause two images to share an edge, even though they show different objects, and this false-positive edge can cause unrelated clusters to be joined into one (Fig. 7.9). Such clusters would not be suitable as a basis for Landmark Recognition systems (Sec. 2.3), since they rely on pure clusters that each

Figure 7.1: *Examples of Watermarks (left), Timestamps (middle), Frames (right). Top: Original images, Bottom: WTFs in higher resolution.*

contain just one object or building. If this is not the case due to WTFs randomly linking unrelated images, a recognition system will confuse the joined buildings.

Moreover, WTFs can form non-object clusters. For example, Chum et al. (2009) report that timestamps form *pseudo*-clusters when performing small object discovery with Geometric min-hash (Sec. 3.4.2) on the Oxford 105k dataset (Philbin et al., 2007). But WTFs practically affect all vision datasets crawled from the web, such as Oxford 5k (Philbin et al., 2007) and Paris Buildings (Philbin et al., 2008), ImageNet (Deng et al., 2009), the Dubrovnik and Rome datasets (Li et al., 2010), European Cities 1M Avrithis et al. (2010), and Paris 500k (Sec. 3.3), to name but a few. Moreover, companies like Google, Facebook or Yahoo (Flickr) rely on massive corpora of Internet images to build their vision-based services.

Finally, WTFs can disturb large-scale structure-from-motion engines based on Internet photos (Agarwal et al., 2009; Crandall et al., 2011; Frahm et al., 2010; Furukawa et al., 2012; Snavely et al., 2006, 2008a,b). If WTFs are present in the input photos, reconstruction will either fail completely or produce incorrect reconstructions where unrelated buildings are connected by photos containing WTFs.

We present an approach to detect whether an image match was caused by a WTF. This method can be applied to filter image retrieval results and to prevent false-positive matches when building a matching graph for image clustering. Given an image pair and its estimated homography, our approach computes a map of highly similar image regions and builds a spatial histogram from it. This histogram serves as input for a binary classifier that decides whether the match was caused by a WTF. This approach is general enough to detect watermarks, timestamps, and frames alike, while being fast enough to be integrated into an existing retrieval or clustering system without introducing too much computational overhead.

While several approaches exist for detecting and reading timestamps in photos (Chen and Zhang, 2003; Fumin et al., 2004; Li, 2006; Shahab et al., 2011), most approaches for detecting visible watermarks are targeted at videos (Kuo et al., 2008; Meisinger et al., 2005; Yan et al., 2005). Because they make strong assumptions about the appearance and position of the timestamps or watermarks, they would fail when applied to the variety of WTFs present in Internet photos. So far, there is no unified approach for detecting watermarks, timestamps, and frames. Metadata, like user-ids or GPS, can

potentially help identify WTF matches, but this data is not always available and often unreliable. We compare our method against a GPS-based approach in Section 7.3.

This chapter makes the following contributions:

- We present a simple and efficient method for detecting whether an image match was caused by watermarks, timestamps or frames.
- For training and evaluation of this detector, we collected a dataset of 3.6k real-world WTF and 33k non-WTF image pairs mined from Internet photos of Paris.
- We perform a detailed evaluation showing the effect of different parameters of the method.
- We apply our detector in a clustering setting and show that it can fix several problems caused by WTFs such as falsely joined clusters, clusters containing unrelated images and pseudo-clusters formed by WTFs.
- We made the source code of the method as well as the dataset publicly available at http://www.vision.rwth-aachen.de/projects/fixing-wtfs.

The chapter is structured as follows. First, we review previous work for detecting watermarks, timestamps or frames in images and videos. In Section 7.2, we present our WTF detection pipeline. We introduce our dataset and evaluate our approach in Section 7.3 and apply it in a clustering setting on the Paris 500k and Oxford 105k datasets. We conclude the chapter in Section 7.4.

This chapter is based on our paper Weyand et al. (2015) presented at WACV 2015. The research for the paper was done in collaboration with my student Chih-Yun Tsai. Chih-Yun contributed to this work with several ideas as well as by performing the implementation work and experiments.

7.1 Related Work

Most of the work on detecting and reading *timestamps* in photos has focused on scanned analog photos and thus on dot font and 7-segment timestamps exposed onto the analog film. While Chen and Zhang (2003) and Fumin et al. (2004) perform template matching with manually created digit templates, Li (2006) uses Self-Generating Neural Networks (SGNNs) to model digit appearance. In contrast to Chen and Zhang (2003) who make no assumptions on timestamp position, Fumin et al. (2004) and Li (2006) limit their search to only the four image corners, while Shahab et al. (2011) learn location priors from a training set. Unlike these works, our goal is not to segment or read timestamps, but merely to detect their presence.

Müller et al. (2005) observed that false positives in radiological image retrieval are often caused by text, logos and frames in the X-ray images. Because in this application, the overlays are separated from the content and the background is always black,

connected components analysis and thresholding suffice to remove the WTFs. While the goal and motivation of Müller et al. (2005) are very similar to ours, general Internet photos require more elaborate methods since content and overlays have a much higher variability in appearance.

A related line of research is to detect text and timestamps in videos. Sato et al. (1998) and Yin et al. (2002) exploit the fact that the background behind the text constantly changes and apply a running temporal minimum over the image intensity to isolate timestamps. This is not applicable in our setting where only two images with an overlay are given that might not have as much background variation. Another approach that only works on videos was used by Li et al. (2006) and Yu et al. (2013). In both their cases, the timestamp includes seconds, which enables searching for parts of the image that change every second. Li et al. (2006) do not even use OCR, but infer the passed time based on the changing pattern of the clock digits.

Meisinger et al. (2005) and Yan et al. (2005) address the problem of removing TV station logos from videos. Both use a simple frame differencing approach, which is however only effective with dynamic backgrounds and non-transparent logos. Therefore, Yan et al. (2005) additionally use a Bayesian classifier with a location prior that favors the image corners.

Zhang et al. (2010) aim to detect objects that have been pasted onto images. Like ours, their method works on pairs of images related by a homography. Assuming that the estimated homography describes the transformation of the background object, while the pasted object does not obey the homography, they warp the images onto each other and subtract them. Image regions with high pixel-wise difference are assumed to be the pasted object. In contrast to our work, their method assumes that the image pair shows the *same subject* and an additional pasted object. WTF matches, on the other hand, have *different* subjects and the homography describes the transformation of the WTF itself. Therefore, in our case, WTFs are the most *similar* image regions.

In summary, most previous work has focused on either detecting timestamps in photos or watermarks (station logos) in videos, but these approaches are too specialized to their application domains and thus not applicable to our problem. Moreover, to the best of our knowledge, no one has addressed the problem of detecting frames in photos, and we are not aware of any previous paper that addresses the detection of watermarks, timestamps, and frames with a single approach.

7.2 Method

The design goals of our detector were that it should be *general* enough for all kinds of watermarks, timestamps and frames, it should have *high accuracy* since a single missed WTF can already cause clusters of unrelated objects to be merged (Fig. 7.9), and it should be *fast* so it can be applied in large-scale settings. Our detector is meant to be used as a post-processing step in an image retrieval system. Given an image *match*, i.e.,

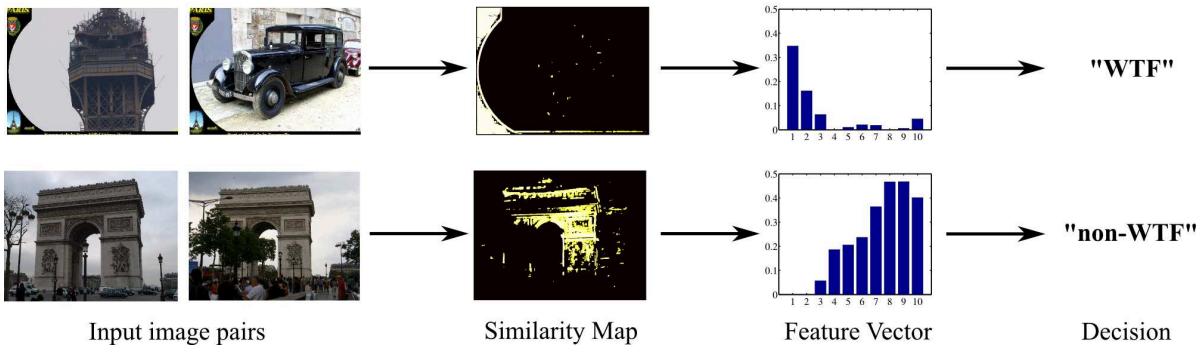


Figure 7.2: *Workflow of the WTF detector. Top: WTF match, bottom: non-WTF match.*

a pair of images related by a homography, it decides whether it is caused by a WTF. An alternative would be to detect WTFs in every *single* image. However, we would like to make the decision *per match* instead of *per image* because an image that has a WTF can still have valid matches.

Although WTFs come in many shapes and sizes (Fig. 7.1), we can make certain assumptions about them. Watermarks and frames always have the exact *same appearance*, since they are simply templates pasted onto the image. Timestamps are slightly more challenging since the background is usually still visible behind them and their appearance differs depending on the time. All WTFs are typically *close to the border* of the image, but they may have varying positions and scales in each of the two matching images. For example, a professional photographer might always place her logo such that it does not occlude the subject.

Based on the above assumptions, we now propose a method to detect WTF matches. Our detector takes as input a *match*, i.e., a pair of images and a homography relating them. Since the detector plugs into a visual word based image retrieval pipeline (Sec. 3.2.2) directly after the spatial verification stage, the homography as well as its inliers will be directly available at no extra cost. Our detector then outputs a binary decision (WTF or not) and two *similarity maps*, i.e., segmentations of the input images into WTF and non-WTF pixels.

The basic steps of our detector are as follows (Fig. 7.2):

1. Detect highly similar image regions.

2. Compute a feature vector consisting of a spatial histogram of the similar regions and some summary statistics.

3. Classify the match as WTF or non-WTF.

In this section, we propose different choices for each of these three steps and evaluate them in Section 7.3.

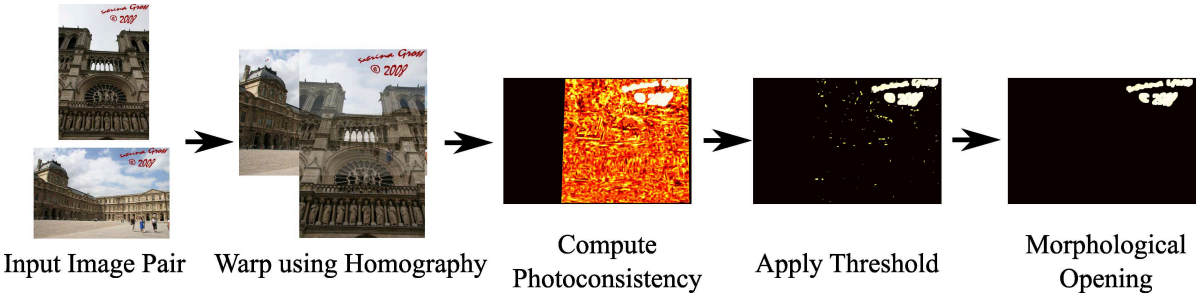


Figure 7.3: *Similarity map computation based on photoconsistency.*

7.2.1 Detecting similar image regions

In the first step, we compute an image similarity map that we later extract features from. We propose two approaches for this: one based on *photoconsistency* and one based on homography *inliers*. In the following, we refer to the images in a match as image 1 and image 2.

Photoconsistency. The process of computing similarity maps based on *photoconsistency* is illustrated in Figure 7.3. Because WTFs can appear at different positions in the image, we first warp image 2 onto image 1 using the homography between them. We then compute pixel-wise photoconsistency scores based on normalized cross-correlation (NCC) by sliding a window of radius r over both images. At each position, we serialize the pixel luminances in both windows into the vectors f and g and compute their NCC as

$$f_0 = f - \bar{f}, \quad g_0 = g - \bar{g}, \quad NCC(f,g) = \langle f_0/\|f_0\|, g_0/\|g_0\| \rangle \,, \tag{7.1}$$

where $\bar{f}$ and $\bar{g}$ denote the mean of f and g, respectively, $\langle ., . \rangle$ denotes the dot product and $\|.\|$ denotes the L_2 norm. Black image frames often cause both f and g to be 0. Since we want to detect this case, we still want them to have a high photoconsistency score and define $NCC(0,0) := 1$.

To isolate WTF regions, we set all NCC values below a threshold θ to 0. Finally, we use morphological opening to eliminate small regions that likely do not belong to a WTF. Figure 7.4 (3rd column) shows example similarity maps computed with this method. While this method generally yields very precise results, we found that uniform image regions with high NCC can often occur randomly across the image. Moreover, this method is expensive and requires the choice of three parameters, namely the NCC window size r, the NCC threshold θ, and the size of the structuring element for morphological opening.

Inliers. Another way to find similar image regions exploits the fact that the homography *inliers*, i.e., corresponding local features that obey the homography transformation, are available to our detector at no extra cost since it is intended to be run after the spatial

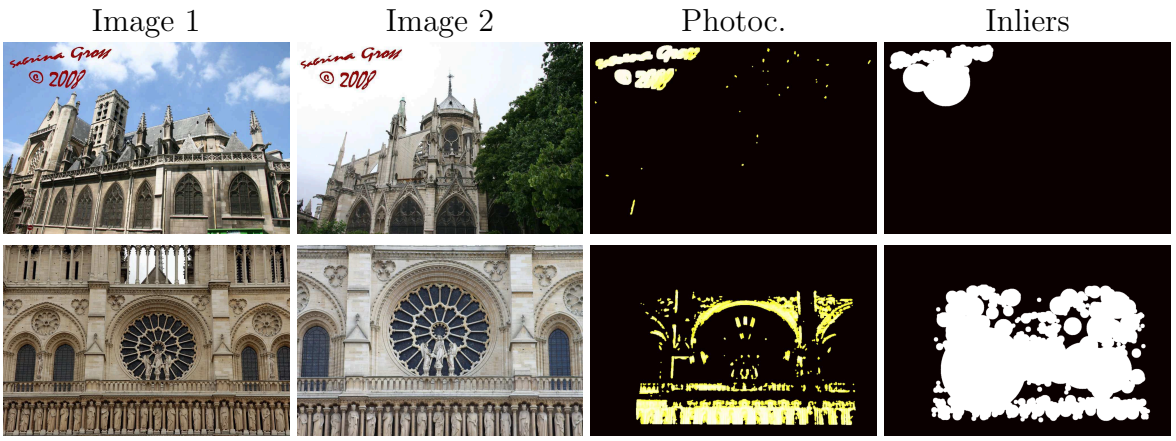


Figure 7.4: *Similarity maps computed with the* photoconsistency *and* inlier *methods.*

verification step in an image retrieval system (Sec. 3.2.2). Assuming that feature correspondences are a sufficient indication of the similarity of their respective image regions, we create the similarity map simply by forming the union of the interest regions of all inliers in image 1. That is, for each inlier, we set all pixels in its interest region to 1 and leave the rest at 0. In contrast to the photoconsistency-based method, the resulting similarity map is binary. This method is much faster than the photoconsistency-based method above and is completely parameter-free. While it yields less precise similarity maps than *photoconsistency*, it does not randomly fire in uniform regions. Figure 7.4 (4th column) shows some example similarity maps computed with this method.

7.2.2 Spatial Histogram of Similar Regions

Based on these similarity maps, we now compute feature vectors suitable as input for a classifier. It seems reasonable to assume that for WTFs, similar regions will appear mainly at the image borders, while for valid matches, similar regions will appear all over the image (Fig. 7.4). Therefore, we compute spatial histograms of the similarity maps. We investigate four histogram shapes (Fig. 7.5). The bins of the `dist2border` and `dist2centre` histograms are bands of the distance to the image border and center, respectively. The bins of `cake` correspond to the angle w.r.t. the center, and `dartboard` is the combination of `dist2centre` and `cake`. The value of a histogram bin is computed as the sum over all similarity map values in it, divided by the total number of pixels in the bin.

We also experiment with additional attributes, namely the *mean photoconsistency* (only for the *photoconsistency* method) and the *coverage*, i.e., the fraction of active pixels in the similarity map. We concatenate these values to the histogram. The resulting feature vector serves as input for AdaBoost with decision trees as weak classifiers. This delivered performance superior to both linear SVMs and SVMs with RBF kernels.

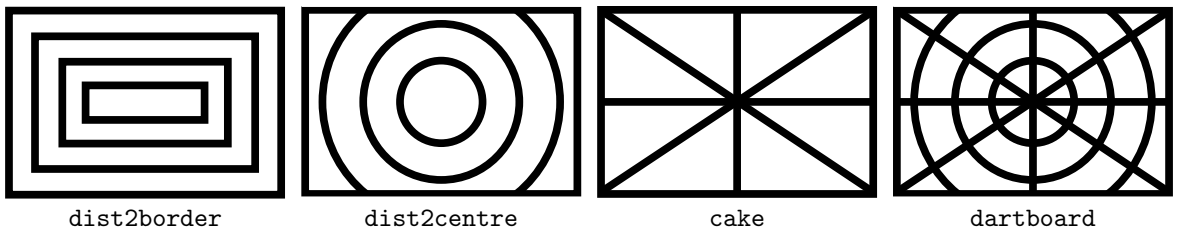


Figure 7.5: *Spatial histogram shapes.*

7.2.3 Two-way Matching

Sometimes the matching region between two images is at the border of one image but in the middle of the other (See Fig. 7.7b). In such a case, depending on which of the images we choose for computing the similarity map, the image pair can be falsely detected as a WTF. To prevent this, we compute the similarity maps and histograms in two directions; once by projecting image 2 onto image 1, and once by projecting image 1 onto image 2. We then only call the image pair a WTF if both feature vectors are labeled as positive by the classifier. This way, we enforce that the WTF regions are close to their learned positions in both images. This not only reduces the number of false positives, but also doubles the amount of training data, since each image pair now generates two training samples.

7.2.4 Integration into Image Retrieval and Clustering.

In visual-words-based *image retrieval* Philbin et al. (2007); Sivic and Zisserman (2003), potential matches of a query image are ranked w.r.t. their tf-idf score and then verified by fitting a geometric transformation. Our detector is simply used as a *second* verification step that rejects WTF matches. This integration directly benefits *image clustering* approaches that are based on a *matching graph*. This graph is built by performing image retrieval and linking matching images by edges. By performing WTF detection after image retrieval, false-positive edges in the matching graph can be avoided directly during graph construction.

7.3 Experiments and Results

We now first give a detailed analysis of our method and its parameters in a classification setting and then show how it can improve image clustering results by filtering false matches.

7.3.1 Dataset and Ground Truth

We collected a realistic dataset of 36,240 image matches for evaluating our WTF detector. Each image pair has a binary label: 90% of the matches are *correct*, or *non-WTF* matches, and 10% are *WTF* matches. We sampled the matches from the matching graph (Sec. 3.2.3) we computed for the Paris 500k dataset (Sec. 3.3) using four methods:

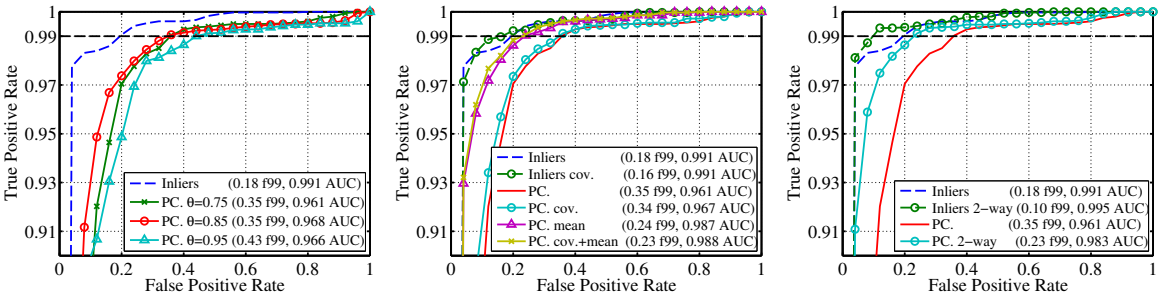
- We searched for users whose photos frequently match among themselves.
- We searched for image pairs where all inliers are in one of the four image corners.
- We clustered the dataset using Iconoid Shift (Ch. 4) and identified clusters containing multiple unrelated buildings, since these are typically caused by WTF matches that create invalid edges in the matching graph.
- We manually went through a large number of images from the dataset and collected the matches of all images containing a WTF.

Note that the first method exploits the availability of author data. While this data could also be of use in detecting WTFs, we deliberately designed our method based on just image data so it can also be applied in settings where this data is not available.

After having collected this initial pool of matches, we then manually inspected all of them and filtered out the remaining non-WTF matches. Because we found that some users were responsible for a large number of matches in the pool, we limited the number of WTF matches from the same user to 250 to avoid a bias towards certain kinds of WTFs. The resulting set of 3,624 image matches consists of 61% Watermarks, 23% Timestamps and 16% Frames. We then added 32,616 random non-WTF matches from the matching graph, so the final dataset has 10% WTF matches. For cross-validation, we then split up the dataset into 5 folds, taking care that WTFs from the same user are all in the same fold to avoid training on the test data. The dataset is publicly available at http://www.vision.rwth-aachen.de/projects/fixing-wtfs.

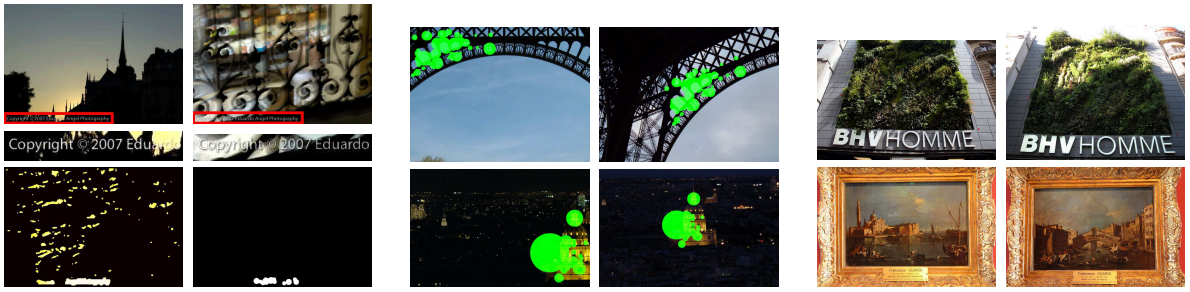
7.3.2 Evaluation Procedure

We now evaluate our method in a *binary classification* setting using 5-fold cross-validation. We plot receiver operator characteristic (ROC) curves by *vertical averaging*, i.e., we average the true positive rates of the five folds for each false positive rate. Additionally, we consider the area under the ROC curve (AUC) and the false positive rate at a true positive rate of 0.99, which we call *f99*. This is motivated by our target application, image clustering. While the effect of a few missing matches is negligible since an image cluster is usually very densely connected, a single false-positive match caused by a WTF can already result in the merging of two unrelated clusters (Fig. 7.9). Our primary goal is therefore to reach high recall. We now analyze the different parameters and design choices introduced in Section 7.2 and how they affect performance and efficiency.



(a) (b) (c)

Figure 7.6: *Effect of different parameters on classification performance. (Note that the y-axis ranges from 0.9 to 1.0.) (a)* Photoconsistency *vs.* inlier *similarity maps (using a* `dist2border` *histogram (Fig. 7.5) with 5 bins). (b) Effect of additional features. (using* θ*=0.75 for* photoconsistency*) (c) Effect of two-way matching.*



(a) (b) (c)

Figure 7.7: *(a) Top: WTF match. Middle: Closeup of WTF. Bottom left:* photoconsistency *map. Bottom right:* inlier *map. (b) False-positive WTFs that two-way matching prevents (similarity map overlaid in green). (c) Two unusual false-positive WTF detections in* Paris 500k*. Top: A planted facade resembling a frame. Bottom: Two paintings with similar* physical *frames.*

7.3.3 Photoconsistency vs. Inliers

The first pipeline step is detecting similar image regions (Sec. 7.2.1). We compare the *photoconsistency* and *inliers* methods using different NCC thresholds θ for *photoconsistency* (Fig. 7.6a). Surprisingly, *inliers* outperforms *photoconsistency* by a large margin (0.18 *f99* vs. 0.35 *f99*). Analyzing the similarity maps showed that the reason why *photoconsistency* is much more prone to false-positives is that high photoconsistency regions are likely to occur by accident, e.g., in uniform regions (Fig. 7.7a). *Inliers* does not have this problem, because interest points are mainly detected in textured regions and because SIFT matching is more discriminative than photoconsistency.

7.3.4 Efficiency

The *photoconsistency* method needs to compute the NCC by applying a sliding window over both images and computing their dot product. Let N be the number of pixels in the overlap of the warped images and r be the window size, then NCC computation requires $O(N * r^2)$ operations. The *inliers* method only needs to fill the interest regions of the inlier features. Let s be the average diameter of an interest region and R the number of interest regions, then the similarity map computation takes $O(R * s^2)$ operations. Since there are much less interest points than pixels ($R \ll N$), the *inliers* method is much faster. In our measurements, the average time to compute the feature vector for an image pair is 1.5 seconds for *photoconsistency* and 0.15 seconds for *inliers*. Both methods were implemented in Matlab, except for the NCC computation in the *photoconsistency* method, which was implemented in C. Timings were taken on a 2.7 GHz Intel Core i7 CPU. We expect significant speedups from a full C implementation, but NCC computation will remain the bottleneck of *photoconsistency*. Therefore, *inliers* is highly preferable both in terms of performance and efficiency.

7.3.5 Additional Features

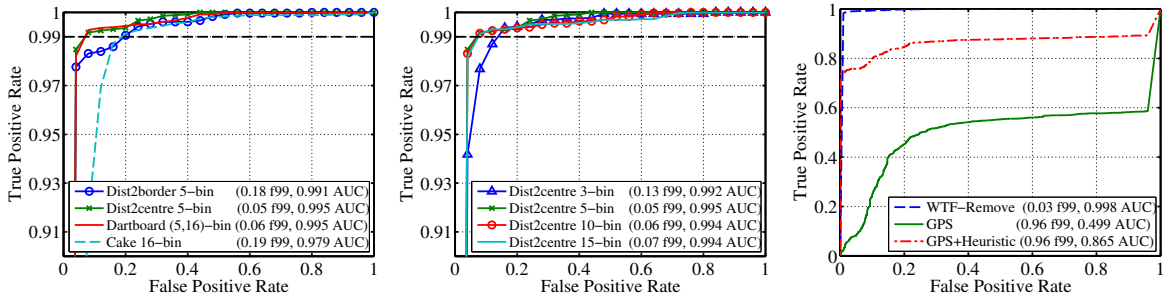
The effect of the additional features *coverage* and *mean* is shown in Figure 7.6b. While *coverage* has only a small effect on both methods, the *mean* strongly increases the performance of the *photoconsistency* method, because it helps eliminate false-positive detections due to accidentally photoconsistent regions.

7.3.6 Two-way Matching

Two-way matching (Sec. 7.2.3) yields a large performance improvement (Fig. 7.6c). The *f99* improves from 0.18 to 0.10 for *inliers* and from 0.35 to 0.23 for *photoconsistency*. This is because firstly, twice the amount of training data is available, and secondly, image matches where the similar region is at the center in one image, but at the border in the other are filtered out by this method. In the examples in Figure 7.7b, only the left images were considered when not using two-way matching, causing them to be falsely classified as WTFs.

7.3.7 Histogram Shapes

We now compare the performance of different shapes of spatial histograms (Sec. 7.2.2). As Figure 7.8a shows, `dist2centre` performs highest. Although `dartboard` with (5,16) bins achieves similar performance, it has a dimensionality of 80, whereas `dist2centre` only has a dimensionality of 5, making it preferable. Figure 7.8b shows the effect of the number of bins for the `dist2centre` histogram and the *inliers* method. The strong performance increase from 3 to 5 bins shows that a certain spatial resolution is required



(a) (b) (c)

Figure 7.8: *(a) Performance of different histogram shapes (using* inliers *and the respective best-performing bin counts). (b) Performance of different histogram sizes. (c) Comparison with two baseline methods. For our detector, we use the best performing setup:* `dist2centre`*, 5-bin, inliers, 2way. (Note that the y-axis is scaled differently in this plot, so the curves of the other methods are visible.)*

2-way	hist.	f99	AUC
	d2c	0.228	0.989
X	d2c	**0.184**	**0.993**
	d2b	0.355	0.961
X	d2b	0.228	0.983

(a) *Photoconsistency*

2-way	hist.	f99	AUC
	d2c	0.045	0.995
X	d2c	**0.034**	**0.998**
	d2b	0.183	0.991
X	d2b	0.095	0.995

(b) *Inliers*

Table 7.1: *Comparison of different detector settings.*

to reliably detect WTFs. However, more bins do not bring an advantage, which again allows us to keep feature dimensionality low.

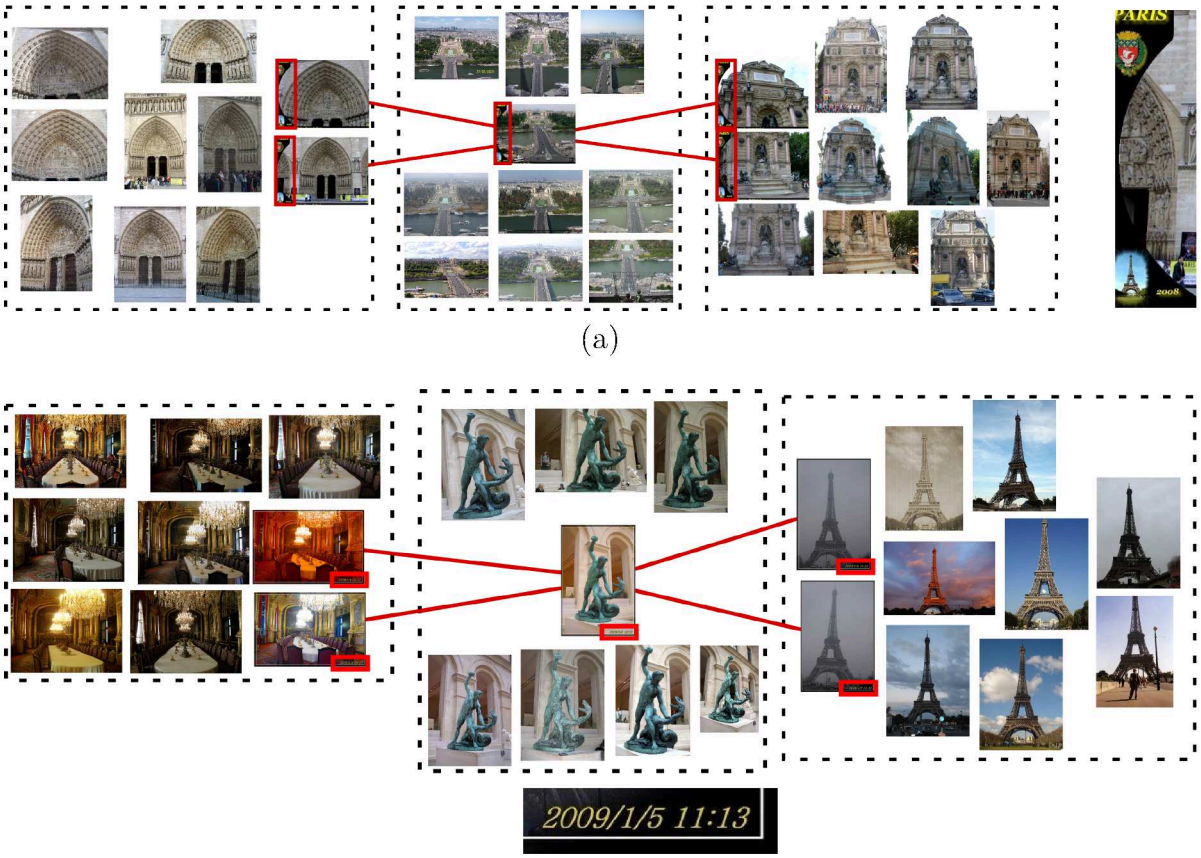
7.3.8 Summary

Table 7.1 summarizes our results. For both *photoconsistency* and *inliers*, `dist2centre` with 2-way matching performs highest. The best setup achieves 3.4% *f99*.

7.3.9 Comparison against Baselines

Since the task of detecting watermarks, timestamps and frames has not been addressed before and previous methods are not applicable to Internet photos (Sec. 7.1), we compare against two simple methods used in practice.

The first method is to use GPS tags to restrict the candidate matches for an image to geographically close images (Avrithis et al., 2010; Quack et al., 2008; Zheng et al.,



(a)

(b)

Figure 7.9: *Two clusters successfully split by our algorithm. All images in (a) and (b), respectively, were initially in the same cluster due to WTF matches (red, closeups on the right and bottom, respectively) that linked images of different buildings. Both clusters were split into three clusters denoted by the dashed rectangles.*

2009), assuming that images taken far apart cannot show the same object. If they match anyway, it must be a false positive. Since geotags are available for our images, we apply this method by classifying all image pairs with a geographic distance above a threshold t as WTFs. Figure 7.8c shows the ROC curve drawn by varying t, compared against our best performing setup. We found that the main reasons for the much lower performance of this method are: (i) Inaccurate GPS tags causing false-positive WTF detections. For example, two pictures of a sculpture in a museum may have far-away geotags due to bad GPS reception, causing the match to be classified as a WTF. (ii) Although two images have close-by geotags, they do not always show the same object. Hence, matches between them can still be caused by a WTF.

We also compare the performance of our detector with a heuristic method we have used in our experiments before, in combination with the GPS-based method. We removed

Figure 7.10: *A cluster in the* Oxford 105k *dataset that contains unrelated images due to false-positive matches caused by timestamps (example on the right). The crossed-out images were removed by our detector, leaving only the relevant images.*

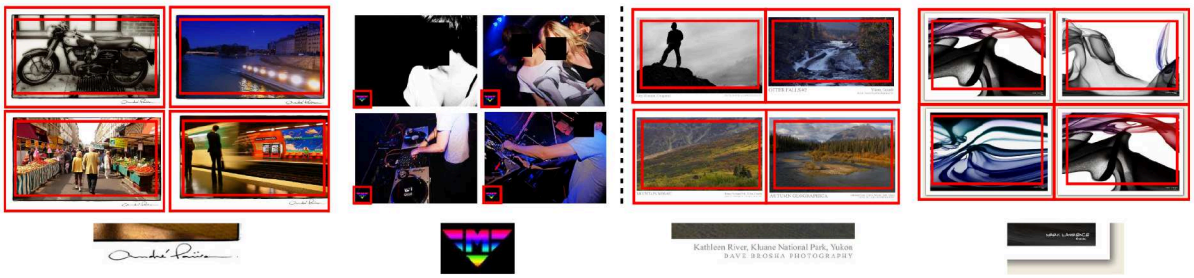


Figure 7.11: *Sample images from removed pseudo-clusters from* Paris 500k *(left)* Oxford 105k *(right). Closeups of the WTFs causing the clusters are shown below.*

image matches if at least 50% of its inliers are in the top or bottom 10% of the image. While this additional step drastically increases the *tpr*, its performance, especially the *f99*, is still not comparable to our WTF detector.

7.3.10 Application to Clustering

We now demonstrate how our detector can improve the results of image clustering and thus benefit applications such as landmark recognition or structure-from-motion. We evaluate our algorithm on two datasets, Paris 500k (Sec. 3.3) and Oxford 105k (Philbin et al., 2007). For each dataset, we build one baseline matching graph using standard image retrieval and one using image retrieval with subsequent WTF removal. We then cluster both graphs using Iconoid Shift (Ch. 4). The clustering based on the baseline matching graph is called *clustering 1* and the clustering based on the matching graph with WTF removal is called *clustering 2*. We now compare these clusterings for both datasets. For WTF removal, we use *inlier* similarity maps, `dist2border` histograms with 5 bins and two-way matching. We selected the classifier score threshold to an operating point of 0.99 *tpr* determined using cross-validation on the WTF dataset.

Results on Paris 500k. *Clustering 1* of Paris 500k has 14,254 clusters covering a total of 111,892 images. 87 clusters, and in particular the three largest clusters, are affected by WTFs. For example, the largest cluster (5,435 images) contains the Eiffel Tower, the Louvre, Notre Dame, the Arc de Triomphe, several paintings and sculptures, as well as many non-landmark tourist photos such as portraits or pictures of food. In *clustering 2*, 38 of these clusters are completely removed since they only contained WTFs (Fig. 7.11). Two WTF clusters survived unharmed, but they contained only 2 and 3

Figure 7.12: *Images from* Oxford 105k *that were in the same cluster because a timestamp (below) caused false-positive matches between them. The cluster was split into the four individual clusters denoted by dashed boxes.*

images respectively. 39 clusters, covering 17,857 images, were split into a total of 317 clusters. Only 14 of them still contained WTF matches, but all of these clusters were smaller than 5 images. The remaining 303 clusters were all pure, each containing only one object. Figure 7.9 shows two successfully split clusters. 8 clusters from *clustering 1* contained only a small fraction of WTF images, but were pure otherwise. In *clustering 2*, all WTFs were removed from these clusters.

Due to false positives, 79 non-WTF clusters were removed, but all of them were very small: 70 of them had only size 2 and the largest had size 7. False positive that are removed in larger clusters usually do not have any effect, since these clusters are densely connected. Most false positive WTF detections are due to image pairs that have matching features only at the border. This can happen, e.g., if there is extreme panning between the two photos such that they have only very little overlap. Figure 7.7c shows two rather unusual cases.

Results on Oxford 105k. The Oxford 105k dataset, used to evaluate image clustering in (Chum and Matas, 2010; Chum et al., 2009; Philbin and Zisserman, 2008), consists of 5k images of *Oxford* showing 11 landmarks, and 100k *distractor* images collected from Flickr by random keyword searches. *Clustering 1* of Oxford 105k has 6,225 clusters covering 17,599 images. Since the *Oxford* part of the dataset is relatively clean of WTFs, none of the Oxford building clusters were falsely merged. We found that 7 clusters in *clustering 1* were merged by WTFs, 8 contained only WTFs, and one cluster contained some WTFs but was otherwise pure. In *clustering 2*, 6 of the 7 merged clusters were

Figure 7.13: *Image pair from a WTF cluster that was not removed due to a common background object. The watermark in the lower right was hardly detected since the homography is on the poster.*

successfully split (see Fig. 7.12 for an example). In the remaining cluster, all photos had a common background object that caused them to match (Fig. 7.13). 7 of 8 clusters containing only WTFs were removed (see Fig. 7.11 for examples). The missed cluster contained 4 identical images with a frame, which were not detected because they had inliers over the whole image area. Finally, the cluster consisting partially of WTFs was successfully cleaned of unrelated images (Fig. 7.10). 203 clusters were removed although they did not contain WTFs, but all of them contained four or less images.

Summary. We have shown that our method prevents the formation of clusters whose images match only due to WTFs and prevents clusters to be merged due to WTFs. Though trained on Paris 500k, our detector was able to generalize to Oxford 105k where it was just as successful in removing WTFs.

7.4 Conclusion

In this chapter, we have presented a simple method to detect whether a match between two images was caused by a watermark, timestamp or frame (WTF) that is present in both images. If this is the case, the match should not be returned by an image retrieval system since the retrieved image is not relevant to the query and the match will cause harm in applications like image clustering or structure-from-motion.

Our method first warps the input images onto each other using their known homography and computes a map of their similar regions. We presented two methods for this, one based on photoconsistency and one based on inliers. Then, our detector computes a spatial histogram and summary statistics of the similar regions and classifies the resulting feature vector using AdaBoost.

We created a realistic dataset for training and evaluation of our detector and showed that it can achieve high performance while being computationally efficient enough to be integrated into the spatial verification phase of an existing image retrieval engine. We showed that when applied in an image clustering application, our detector can suc-

cessfully fix the problems such as falsely merged clusters, clusters containing unrelated images, and clusters that are only formed by WTFs.

One remaining problem of our method is that false-positive detections can be caused by an image pair that has extreme pan or tilt such that only the borders of the images match. Since our detector is based on spatial histograms of similar regions, such a match will "look" like a WTF match to our detector. As we found in our clustering evaluation these false-positives do not have a profound effect since images are typically linked via many paths in the matching graph. However, a method for avoiding such detections would be desirable.

Finally, our method does not detect all possible WTFs. Because the vast majority of WTFs are close to the image border, AdaBoost learns that spatial histograms with high values near the image border are likely to be WTFs. Therefore, our detector will miss watermarks or logos that are placed in the middle of the image. In order to distinguish such cases from correct matches, it might be necessary to not only model the position but also the appearance of WTFs, which should be possible since watermarks, timestamps and frames are designed to stand out from the image content.

8 Conclusion

In this thesis, we have addressed the problem of discovering landmarks in large collections of tourist photos from photo sharing websites such as Flickr or Panoramio.

Previous work has approached this task by partitioning the image collection into clusters, each corresponding to a landmark. In contrast, our *Iconoid Shift* algorithm casts this task as a *mode search* problem, where the modes, called *Iconoids*, correspond to the most frequently photographed views. This novel view of the problem has several appealing advantages. It has an *intuitive cluster definition*: a cluster is defined as the set of all images that overlap with an iconic view of a landmark. Iconoid Shift is *efficient*, because it only performs a local exploration of the matching graph. The algorithm is based on well-understood mode search methods and inherits their proven properties such as their *intuitive parameters* and *guaranteed convergence*. Finally, Iconoid Shift is *inherently parallel* and can be distributed to hundreds of machines.

Iconoid Shift, like most other previous landmark discovery approaches, is aimed at discovering clusters that correspond to *building-scale* objects. However, Internet photo collections contain much more fine-grained information, also revealing the many architectural details of the world's landmark buildings, like sculptures and reliefs on their facades and murals on their walls and ceilings. These details were missed by previous algorithms that performed a hard clustering at a single, fixed scale. In contrast, our *Hierarchical Iconoid Shift* algorithm produces an *overlapping clustering* and discovers clusters at *multiple scales*, resulting in a hierarchical description of a building. The algorithm shares the desirable properties of Iconoid Shift, but does not require the user to set the clustering scale by hand, making it completely *parameter-free*. The basis of Hierarchical Iconoid Shift is our *Hierarchical Medoid Shift* algorithm. Inspired by Scale Space theory, this algorithm extends Medoid Shift to perform a hierarchical clustering by following density maxima while continuously increasing the clustering scale.

In order to analyze the potential of landmark recognition systems based on landmarks that were automatically discovered in Internet photo collections, we performed a large-scale evaluation on a dataset of 500k images of Paris. We first inspected how many and

what kinds of objects can automatically be discovered and then considered each stage of a typical landmark recognition pipeline to find out where such systems perform well, which factors limit their performance, and how different parameter choices affect the recognition result. Our analysis showed that that performance can vary significantly across different object categories. For example, while recognition generally performs well for building-scale landmarks, the lack of reliable semantic information can limit the performance of smaller landmarks such as murals or building details. For each of the components of such a system, we analyzed several methods from the literature as well as some of our own methods and pointed out directions where progress can still be made.

8.1 Summary and Contributions

After introducing fundamental techniques in Chapter 3, we analyzed two existing landmark discovery algorithms based on min-hash (Chum and Matas, 2010) and Spectral Clustering (Philbin and Zisserman, 2008), respectively. This analysis brought to light some of the challenges that existing methods are facing. For example, the clustering algorithms and grouping criteria these approaches use are not ideally suited to the problem and often lead to an over- or under-segmentation. Moreover, some methods require the construction of a matching graph for the full dataset, which can become a computational bottleneck, especially when considering the rapid growth of Internet photo collections.

In Chapter 4, we introduced the *Iconoid Shift* algorithm that addresses the above challenges. We defined clusters as set of all images overlapping with a central, iconic view of a building, called *Iconoid*. Iconoids are the views of buildings most favored by photographers. We find these views by an iterative mode search procedure that seeks photos with minimum total *homography overlap distance* to other photos of the same object. This distance measure allows us to efficiently compute the overlap between photos without performing an explicit matching for each image pair, and enables computing the overlap of photos that do not form a match due to extreme lighting or perspective changes. We presented efficient algorithms to propagate pairwise overlaps in a *local* matching graph that is explored on-the-fly along the convergence trajectory of the mode search. In experiments on large-scale datasets of photos from Paris and Barcelona, we demonstrated that Iconoid Shift discovers views of landmarks that are often more frontal and centered than views selected by other algorithms and comparable to views selected by the authors of the Wikipedia articles of the respective landmark.

Based on this, we then presented the *Hierarchical Iconoid Shift* (HIS) algorithm in Chapter 5. HIS is based on *Hierarchical Medoid Shift* (HMS), a hierarchical extension of the Medoid Shift (Sheikh et al., 2007) clustering algorithm. HMS is inspired by Scale Space Filtering (Witkin, 1984) that tracks the evolution of infliction points of a signal while increasing the bandwidth of a Gaussian kernel applied to it. Similarly, HMS tracks the evolution of density maxima while increasing the bandwidth of the underlying kernel density estimator. The key insight of HMS is that, for certain kernels,

the density maximum cannot change unless a new point enters the kernel window. This allows the algorithm to achieve the same result as if the kernel were grown *continuously* by only visiting a finite number of events. HMS produces a dendrogram describing the density structure of the dataset, and each level of the dendrogram represents an overlapping clustering at a particular scale. The algorithm can easily be distributed to many machines, has the same worst-case complexity as Medoid Shift, but is completely parameter-free. HIS extends Iconoid Shift to apply HMS to the task of discovering landmarks and their details by alternately performing mode search and increasing the kernel scale. The latter is done efficiently by incrementally adding new images to the kernel window and propagating their overlap to all images already under the kernel support. We evaluated HIS on a large-scale dataset of 36 landmarks and showed that its resulting dendrograms capture the relationships of their details. Moreover, we outlined two potential applications, namely landmark building summarization and proposing new details to be added to Wikipedia articles.

In Chapter 6, we then considered a specific application of landmark discovery, namely the automatic construction of landmark recognition engines. While several papers have proposed such systems, and they are already in productive use in applications like Google Goggles, there has not been an evaluation of the performance they can achieve. We therefore performed a large-scale evaluation of such a landmark recognition pipeline to find answers to the following questions: How many and what kinds of objects are present in Internet photo collections and what is the difficulty of discovering objects of different object categories? How to decide which landmark was recognized given a list of retrieved images? How to efficiently represent the discovered objects in memory for recognition? Are the user-provided tags reliable enough for determining accurate object names? What are the factors effectively limiting the recognition of different landmark types? We invested significant effort into creating ground truths for landmark retrieval, recognition and semantic annotation. Our analysis resulted in a number of interesting insights. For instance, we found that the main performance bottleneck is semantic annotation, since user-provided tags are noisy and sparse, which particularly hurts accuracy for less frequently photographed objects. In a more fine-grained analysis on an object category level, we found that different bottlenecks exist for different categories. For example, some objects like murals are easy to recognize using standard image retrieval techniques, but semantic information is often missing or unreliable for them. For other objects, like museum exhibits, the opposite is the case: User-provided tags are more reliable, but their spatial structure and limited number of photos makes them hard to recognize visually. We also analyzed different methods for reducing the memory footprint of a landmark recognition engine by removing redundant photos from its index and found different reduction strategies to be optimal for different object categories. Our analysis showed that when using a seeding-based algorithm like Iconoid Shift, larger objects, like landmark buildings, are discovered already with few seeds while smaller objects are only discovered with more seeds. The number of seeds can therefore be used to trade off the efficiency of object discovery and the comprehensiveness of the resulting index.

Finally, in Chapter 7 we considered an emerging problem of image retrieval in Internet photo collections. More and more users are adding artificial overlays, like watermarks, timestamps or frames to their images, which cause false-positive image matches. In image clustering algorithms that rely on accurate image retrieval, these matches can in turn cause clusters of unrelated objects to be joined into one. We often observed this problem with Iconoid Shift and therefore presented a method to filter false-positive matches that are due to watermarks, timestamps or frames (WTFs). The method works by first computing a map of similar regions in a matching pair of photos and performing spatial binning. Then, we train an AdaBoost classifier on the resulting feature vectors to distinguish between WTF-matches and valid matches. The intuition here is that WTFs mostly cause image matches near the image border, while valid matches are more likely to occur towards the image center. The regions where WTFs occur are automatically learned by the classifier. For our evaluation, we created a dataset of WTF matches based on Paris 500k. We demonstrated that the method successfully avoids unrelated clusters from being merged, and prevents pseudo-clusters caused, e.g., by timestamps, from being formed.

8.2 Future Work

There are several potential future directions to extend and improve the work presented in this thesis:

Incorporating Image Weights. Our evaluation of Iconoid Shift has shown that the algorithm is well-suited for choosing iconic images of buildings or objects. However, Iconoid Shift only optimizes for the best *view*, but does not necessarily find the most *aesthetically pleasant* picture (Fig. 4.12). While the viewing angle plays a role, several other factors such as contrast, color or sharpness also affect how aesthetic an image is. One future direction could therefore be a generalization of Iconoid Shift that incorporates a per-image weight, where images with higher weights are more likely to be selected as Iconoids. This weight could be determined , e.g., by a scoring system trained on human judgments of how aesthetically pleasant an image is. This extension could also be used to avoid certain images from being selected as an iconic image. For example, an image with recognizable people in front of a touristic landmark might not be suitable to represent this landmark in a tour guide for anonymity reasons.

Incremental Clustering. In order to have an up-to-date index of the tourist attractions of the world, the operator of a landmark recognition engine might want to regularly re-run landmark discovery to add new popular buildings and objects, and to incorporate new images of existing ones. However, the rapidly increasing number of photos on photo sharing websites may soon make it computationally prohibitive to do this regularly. Therefore, an incremental version of Iconoid Shift or Hierarchical Iconoid Shift that

updates an existing clustering by incorporating new images and deleting old ones might be of practical use.

More precise Overlap Estimation. Currently, Iconoid Shift approximates the overlap between a pair of images by simply drawing bounding boxes around the inlier features of the homography between the images. In cases where the object of interest does not have rectangular shape, this will lead to an overestimation of the overlap. In cases where no inliers are present at the object borders, the overlap might be underestimated. Here, a more precise method of computing the overlap might be helpful. For example, the overlap region could be estimated as the convex hull around the inlier features or the union of their interest regions. This might improve the accuracy of the algorithm and make the estimated overlap regions more useful for determining the location of the object of interest in a photo.

Reduced Complexity. Although we have presented efficient algorithms for propagating image overlaps in the local matching graph, this step remains the main bottleneck of Iconoid Shift due to its quadratic complexity in the number of images under the current kernel window. This complexity cannot simply be reduced, since the homography overlap distance requires computing the overlap region between any pair of images under the kernel. Therefore, a future direction could be the development of other distance measures and propagation schemes that are equally well-suited to finding iconic views via mode search, but that allow performing the Medoid Shift minimization (Eq. 4.5) in sub-quadratic time.

Extension to other Object Classes. Iconoid Shift and other landmark discovery algorithms cannot discover arbitrary types of objects, because they rely on visual word based image retrieval and matching techniques. Due to the assumptions underlying visual word based image retrieval (Sec. 3.1), it can only match images of the same object *instance* (e.g., a specific painting), and can only handle *rigid* objects, i.e., objects that do not change their shape. Therefore, visual word based image retrieval has been successfully used for recognizing buildings, paintings, CD-covers, or movie posters, but is not suitable, e.g., for recognizing animals or plants. In principle, Iconoid Shift is not limited to rigid objects, and could be extended to more general object classes by replacing the underlying matching procedure by something more generic. However, recognizing that two images show the same (possibly deformable) object class, is a more fundamental problem of computer vision and therefore outside the scope of this thesis.

Hierarchical Medoid Shift Applications. We presented Hierarchical Medoid Shift (Sec. 5.2), an extension of the Medoid Shift algorithm (Sheikh et al., 2007) that we applied to discovering landmarks and their details in the Hierarchical Iconoid Shift algorithm (Sec. 5.3). However, HMS is not limited to this application and can in principle be used to perform hierarchical clustering of all kinds of data where a distance metric can be defined.

Building Knowledge Graphs. A potential application for Hierarchical Iconoid Shift is to generate *knowledge graphs* that encode relational information about buildings and their details. In such a graph, the full building as well as its details are nodes, and each edge encodes a relation between the connected nodes. Examples of such relations could be "is a part of", "is left of", or "is inside of". These relations could automatically be discovered by estimating the camera movement between two Iconoids in a dendrogram produced by HIS. More information like object names and categories could be added to the resulting graphs by parsing Wikipedia articles and other related websites about a building. The resulting graph could then be used to automatically infer answers to questions such as "What is the name of the leftmost gate on the west facade of Notre Dame?", or "How many spires does the Church on Spilt Blood have?".

Incorporate appearance into WTF detection. Our detector for invalid matches that are due to watermarks, timestamps and frames (Ch. 7) currently relies only on the spatial position of matching image regions to determine whether an image match is due to a WTF. This can lead to false-negative detections when the WTF is not near the image border, and can lead to false-positive detections if the images of a match only overlap by a small fraction. It might be possible to prevent these cases by incorporating appearance-based features, since some WTFs, like timestamps, have a characteristic appearance. This would further improve the accuracy of the detector and thus also the quality of the results of image retrieval and clustering.

Bibliography

Agarwal, S., Snavely, N., Simon, I., Seitz, S., and Szeliski, R. (2009). Building Rome in a Day. In *International Conference on Computer Vision*.

Agrawal, R., Imieliński, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD*.

Arandjelovic, R. and Zisserman, A. (2011). Smooth Object Retrieval using a Bag of Boundaries. In *International Conference on Computer Vision*.

Arandjelović, R. and Zisserman, A. (2012a). Name that Sculpture. In *ACM International Conference on Multimedia Retrieval*.

Arandjelović, R. and Zisserman, A. (2012b). Three things everyone should know to improve object retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Arya, S. and Mount, D. M. (1992). Approximate nearest neighbor queries in fixed dimensions. In *ACM-SIAM Symposium on Discrete Algorithms*.

Avrithis, Y., Kalantidis, Y., Tolias, G., and Spyrou, E. (2010). Retrieving Landmark and Non-Landmark Images from Community Photo Collections. In *ACM Multimedia*.

Baatz, G., Koeser, K., Chen, D., Grzeszczuk, R., and Pollefeys, M. (2010). Handling urban location recognition as a 2D homothetic problem. In *European Conference on Computer Vision*.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding (CVIU)*, **110**(3), 346–359.

Beaudet, P. R. (1978). Rotationally invariant image operators. In *International Joint Conference on Pattern Recognition*.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, **18**(9), 509–517.

Berg, A. C. and Malik, J. (2001). Geometric Blur for Template Matching. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Berg, T. and Berg, A. B. (2009). Finding Iconic Images. In *CVPR'09 Internet Vision Workshop*.

Bergamo, A., Sinha, S. N., and Torresani, L. (2013). Leveraging Structure from Motion to Learn Discriminative Codebooks for Scalable Landmark Classification. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Blanz, V., Tarr, M. J., Bülthoff, H. H., and Vetter, T. (1999). What object attributes determine canonical views? *Perception*, **28**(5), 575–600.

Broder, A. (1997). On the Resemblance and Containment of Documents. In *Compression and Complexity of Sequences*.

Cao, S. and Snavely, N. (2013). Graph-Based Discriminative Learning for Location Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Cao, S. and Snavely, N. (2014). Minimal Scene Descriptions from Structure from Motion Models. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chakravarthy, S. V. and Ghosh, J. (1996). Scale-based clustering using the radial basis function network. *IEEE Trans. on Neural Networks*, **7**(5), 1250–1261.

Chen, D., Baatz, G., Köser, K., Tsai, S., Vedantham, R., Pylvänäinen, T., Roimela, K., Chen, X., Bach, J., Pollefeys, M., Girod, B., and Grzeszczuk, R. (2011). City-scale landmark identification on mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Chen, X. and Zhang, H.-J. (2003). Photo time-stamp detection and recognition. In *International Converence on Document Analysis and Recognition*.

Cheng, Y. (1995). Mean shift mode seeking and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **17**(8), 790–799.

Chin, F. and Houck, D. (1978). Algorithms for Updating Minimal Spanning Trees. *Journal of Computer and System Sciences*, **16**(3), 333–344.

Chum, O. and Matas, J. (2010). Large-scale discovery of spatially related images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(2), 371–377.

Chum, O., Philbin, J., Isard, M., and Zisserman, A. (2007a). Scalable Near Identical Image and Shot Detection. In *Conference on Image and Video Retrieval*.

Chum, O., Philbin, J., Sivic, J., and Zisserman, A. (2007b). Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. In *International Conference on Computer Vision*.

Chum, O., Philbin, J., and Zisserman, A. (2008). Near Duplicate Image Detection: min-Hash and tf-idf Weighting. In *British Machine Vision Conference*.

Chum, O., Perdoch, M., and Matas, J. (2009). Geometric min-Hashing: Finding a (Thick) Needle in a Haystack. In *International Conference on Computer Vision*.

Comaniciu, D. and Meer, P. (2002). Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(5), 603–619.

Comaniciu, D., Ramesh, V., and Meer, P. (2000). Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Crandall, D., Backstrom, L., Huttenlocher, D., and Kleinberg, J. (2009). Mapping the World's Photos. In *World Wide Web Conference*.

Crandall, D., Owens, A., Snavely, N., and Huttenlocher, D. (2011). Discrete-continuous optimization for large-scale structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*.

DeMenthon, D. (2002). Spatio-Temporal Segmentation of Video by Hierarchical Mean Shift Analysis. In *SMVP*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Denton, T., Demirci, M., Abrahamson, J., Shokoufandeh, A., and Dickinson, S. (2004). Selecting Canonical Views for View-based 3D Object Recognition. In *International Conference on Pattern Recogntion*.

Duda, R. O., Hart, P. E., and Stork, D. G. (2000). *Pattern Classification*. Wiley-Interscience, ISBN: 978-0-471-05669-0.

Epshtein, B., Ofek, E., Wexler, Y., and Zhang, P. (2007). Hierarchical Photo Organization using Geo-Relevance. In *ACM Advances in Geographic Information Systems*.

Fischler, M. and Bolles, R. (1981). Random Sampling Consensus: A Paradigm for Model Fitting with Application to Image Analysis and Automated Cartography. *Communications of the ACM*, **24**, 381–395.

Frahm, J.-M., Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.-H., Dunn, E., Clipp, B., Lazebnik, S., and Pollefeys, M. (2010). Building Rome on a Cloudless Day. In *European Conference on Computer Vision*.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, **21**(1), 32–40.

Fumin, B., Aiguo, L., and Zheng, Q. (2004). Photo Time-Stamp Recognition Based on Particle Swarm Optimization. In *International Conference on Web Intelligence*.

Furukawa, Y. and Ponce, J. (2009). Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(8), 1362–1376.

Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R. (2012). Towards Internet-scale Multi-view Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Gammeter, S., Quack, T., and Van Gool, L. (2009). I Know What You Did Last Summer: Object-Level Auto-Annotation of Holiday Snaps. In *International Conference on Computer Vision*.

Gammeter, S., Quack, T., Tingdahl, D., and Van Gool, L. (2010). Size Does Matter: Improving Object Recognition and 3D Reconstruction with Cross-Media Analysis of Image Clusters. In *European Conference on Computer Vision*.

Grauman, K. and Leibe, B. (2011). *Visual Object Recognition (Synthesis Lectures on Artificial Intelligence and Machine Learning)*. Morgan & Claypool, ISBN: 978-1598299687.

Gronat, P., Obozinski, G., Sivic, J., and Pajdla, T. (2013). Learning per-location classifiers for visual place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hall, P. and Owen, M. J. (2005). Simple Canonical Views. In *British Machine Vision Conference*.

Hao, Q., Cai, R., Li, Z., Zhang, L., Pang, Y., and Wu, F. (2012). 3D Visual Phrases for Landmark Recognition . In *IEEE Conference on Computer Vision and Pattern Recognition*.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey Vision Conference*.

Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518.

Irschara, A., Zach, C., Frahm, J.-M., and Bischof, H. (2009). From structure-from-motion point clouds to fast location recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Jing, Y. and Baluja, S. (2008). VisualRank: Applying PageRank to Large-Scale Image Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **30**(11), 1877–1890.

Jing, Y., Baluja, S., and Rowley, H. (2007). Canonical Image Selection from the Web. In *Conference on Image and Video Retrieval*.

Johns, E. and Yang, G.-Z. (2011). From images to scenes: Compressing an image cluster into a single scene model for place recognition. In *International Conference on Computer Vision*.

Johns, E. and Yang, G.-Z. (2014). Generative Methods for Long-Term Place Recognition in Dynamic Scenes. *International Journal of Computer Vision*, **106**(3), 297–314.

Johnson, D. S. (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, **9**, 256–278.

Jégou, H., Douze, M., and Schmid, C. (2008). Hamming embedding and weak geometric consistency for large scale image search. In *European Conference on Computer Vision*.

Jégou, H., Douze, M., and Schmid, C. (2009a). On the burstiness of visual elements. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Jégou, H., Douze, M., and Schmid, C. (2009b). Packing Bag-of-features. In *International Conference on Computer Vision*.

Jégou, H., Douze, M., and Schmid, C. (2011). Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **33**(1), 117–128.

Kalantidis, Y., Tolias, G., Avrithis, Y., Phinikettos, M., Spyrou, E., Mylonas, P., and Kollias, S. (2011). Viral: Visual image retrieval and localization. *Multimedia Tools and Applications*, **51**(2), 555–592.

Kaufman, L. and Rousseeuw, P. J. (1990). *Finding Groups in Data; An Introduction to Cluster Analysis*. Wiley-Interscience, ISBN: 9780471878766.

Kennedy, L. and Naaman, M. (2008). Generating Diverse and Representative Image Search Results for Landmarks. In *World Wide Web Conference*.

Knopp, J., Sivic, J., and Pajdla, T. (2010). Avoiding confusing features in place recognition. In *European Conference on Computer Vision*.

Kuo, C.-M., Chao, C.-P., Chang, W.-H., and Shen, J.-L. (2008). Broadcast Video Logo Detection and Removing. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*.

Ladikos, A., Boyer, E., Navab, N., and Ilic, S. (2010). Region graphs for organizing image collections. In *RMLE10*.

Letessier, P., Buisson, O., and Joly, A. (2012). Scalable mining of small visual objects. In *ACM Multimedia*.

Leung, Y., Zhang, J.-S., and Xu, Z.-B. (2000). Clustering by Scale-Space Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(12), 1396–1410.

Li, A. (2006). Fast Photo Time-Stamp Recognition Based on SGNN. In *Advances in Neural Networks - ISNN 2006*, volume 3972, pages 316–321. Springer Berlin Heidelberg.

Li, X., Wu, C., Zach, C., Lazebnik, S., and Frahm, J.-M. (2008). Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. In *European Conference on Computer Vision*.

Li, Y., Wan, K., Yan, X., and Yu, X. (2006). Video Clock Time Reconition Based on Temporal Periodic Pattern Change of the Digit Characters. In *IEEE Conference on Acoustics, Speech and Signal Processing*.

Li, Y., Crandall, D. J., and Huttenlocher, D. P. (2009). Landmark classification in large-scale image collections. In *International Conference on Computer Vision*.

Li, Y., Snavely, N., Huttenlocher, D., and Fua, P. (2010). Location Recognition using Prioritized Feature Matching. In *European Conference on Computer Vision*.

Li, Y., Snavely, N., Huttenlocher, D., and Fua, P. (2012). Worldwide Pose Estimation Using 3D Point Clouds. In *European Conference on Computer Vision*.

Lindeberg, T. (1998). Feature detection with automatic scale selection. *International Journal of Computer Vision*, **30**(2), 79–116.

Lindeberg, T. and Garding, J. (1997). Shape-adapted smoothing in estimation of 3-d shape cues from affine deformations of local 2-d brightness structure. *Image and Vision Computing*, **15**(6), 415—434.

Lowe, D. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, **60**(2), 91–110.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Matas, J., Chum, O., Urban, M., and Pajdla, T. (2002). Robust wide-baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*.

Meisinger, K., Troeger, T., Zeller, M., and Kaup, A. (2005). Automatic TV logo removal using statistical based logo detection and frequency selective inpainting. In *European Signal Processing Conference*.

Mezuman, E. and Weiss, Y. (2012). Learning about Canonical Views from Internet Image Collections . In *Neural Information Processing Systems*.

Mikolajczyk, K. and Schmid, C. (2004). Scale and Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, **60**, 63–86.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(10), 1615–1630.

Mikulik, A., Radenovic, F., Chum, O., and Matas, J. (2014). Efficient Image Detail Mining. In *Asian Conference on Computer Vision*.

Müller, H., , Heuberger, J., and Geissbuhler, A. (2005). Logo and text removal for medical image retrieval. In *Bildverarbeitung für die Medizin 2005*.

Newman, M. and Girvan, M. (2004). Finding and Evaluating Community Structure in Networks. *Physical Review*, **E 69**(026113).

Ng, A., Jordan, M., and Weiss, Y. (2001). On Spectral Clustering: Analysis and an Algorithm. In *Neural Information Processing Systems*.

Nistér, D. and Stewénius, H. (2006). Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Olivia, A. and Torralba, A. (2001). Modeling the shape of the scene: a holistic representation of the spatial envelope. *International Journal of Computer Vision*, **42**(3), 145–175.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab.

Palmer, S., Rosch, E., and Chase, P. (1981). Canonical perspective and the perception of objects. *Attention and Performance*, pages 135–151.

Papadopoulos, S., Kompatsiaris, Y., Zigkolis, C., Vakali, A., and Kapiris, S. (2010). ClustTour: City Exploration by use of Hybrid Photo Clustering. In *ACM Multimedia*.

Paris, S. and Durand, F. (2007). A Topological Approach to Hierarchical Segmentation using Mean Shift. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Philbin, J. and Zisserman, A. (2008). Object Mining using a Matching Graph on Very Large Image Collections. In *Indian Conference on Computer Vision, Graphics and Image Processing*.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2007). Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Philbin, J., Sivic, J., and Zisserman, A. (2011). Geometric Latent Dirichlet Allocation on a Matching Graph for Large-scale Image Datasets. *International Journal of Computer Vision*, **95**, 138–153.

Quack, T., Leibe, B., and Van Gool, L. (2008). World-Scale Mining of Objects and Events from Community Photo Collections. In *Conference on Image and Video Retrieval*.

Raguram, R. and Lazebnik, S. (2008). Computing Iconic Summaries for General Visual Concepts. In *CVPR'09 Internet Vision Workshop*.

Raguram, R., Frahm, J.-M., and Pollefeys, M. (2008). A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In *European Conference on Computer Vision*.

Raguram, R., Wu, C., Frahm, J.-M., and Lazebnik, S. (2011). Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. *International Journal of Computer Vision*, **95**(3), 213–239.

Russell, B. C., Martin-Brualla, R., Butler, D. J., Seitz, S. M., and Zettlemoyer, L. (2013). 3D Wikipedia: Using Online Text to Automatically Label and Navigate Reconstructed Geometry. *ACM Transactions on Graphics*, **32**(6), 193:1–193:10.

Sato, T., Kanade, T., Hughes, E. K., and Smith, M. A. (1998). Video OCR for Digital News Archive. In *International Workshop on Content-Based Access of Image and Video Databases*.

Sattler, T., Leibe, B., and Kobbelt, L. (2009). SCRAMSAC: Improving RANSAC's Efficiency with a Spatial Consistency Filter. In *International Conference on Computer Vision*.

Sattler, T., Leibe, B., and Kobbelt, L. (2011). Fast image-based localization using direct 2d-to-3d matching. In *International Conference on Computer Vision*.

Sattler, T., Weyand, T., Leibe, B., and Kobbelt, L. (2012). Image Retrieval for Image-Based Localization Revisited. In *British Machine Vision Conference*.

Schindler, G., Brown, M., and Szeliski, R. (2007). City-Scale Location Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Shahab, A., Shafait, F., and Dengel, A. (2011). Bayesian Approach to Photo Time-Stamp Recognition. In *International Converence on Document Analysis and Recognition*.

Sheikh, Y., Khan, E., and Kanade, T. (2007). Mode-Seeking by Medoidshifts. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Silpa-Anan, C. and Hartley, R. (2008). Optimised kd-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Simon, I. and Seitz, S. (2008). Scene Segmentation Using the Wisdom of Crowds. In *European Conference on Computer Vision*.

Simon, I., Snavely, N., and Seitz, S. (2007). Scene Summarization for Online Image Collections. In *International Conference on Computer Vision*.

Sivic, J. and Zisserman, A. (2003). Video Google: A Text Retrieval Approach to Object Matching in Videos. In *International Conference on Computer Vision*, volume 2.

Sivic, J. and Zisserman, A. (2009). Efficient Visual Search of Videos Cast as Text Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **31**(4), 591–606.

Snavely, N., Seitz, S., and Szeliski, R. (2006). Photo Tourism: Exploring Photo Collections in 3D. In *ACM SIGGRAPH*.

Snavely, N., Seitz, S., and Szeliski, R. (2008a). Modeling the World from Internet Photo Collections. *International Journal of Computer Vision*, **80**(2), 189–210.

Snavely, N., Seitz, S. M., and Szeliski, R. (2008b). Skeletal graphs for efficient structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Szeliski, R. (2011). *Computer Vision; Algorithms and Applications*. Springer London, ISBN: 978-1-84882-934-3.

Tipping, M. and Schölkopf, B. (2001). A kernel approach for vector quantization with guaranteed distortion bounds. In *Artificial Intelligence and Statistics*.

Tola, E., Lepetit, V., and Fua, P. (2010). DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(5), 815–830.

Torii, A., Sivic, J., and Pajdla, T. (2011). Visual localization by linear combination of image descriptors. In *ICCV Mobile Vision Workshop*.

Turcot, P. and Lowe, D. (2009). Better matching with fewer features: The selection of useful features in large database recognition problems. In *ICCV Workshop on Emergent Issues in Large Amounts of Visual Data*.

Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, **3**(3), 177–280.

Tuytelaars, T. and Schmid, C. (2007). Vector Quantizing Feature Space with a Regular Lattice. In *International Conference on Computer Vision*.

Vatturi, P. and Wong, W.-K. (2009). Category Detection using Hierarchical Mean Shift. In *ACM Conference on Knowledge Discovery and Data Mining*.

Weinshall, D. and Werman, M. (1997). On View Likelihood and Stability. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**(2), 97–108.

Weyand, T. and Leibe, B. (2011). Discovering Favorite Views of Popular Places with Iconoid Shift. In *International Conference on Computer Vision*.

Weyand, T. and Leibe, B. (2013). Discovering Details and Scene Structure with Hierarchical Iconoid Shift. In *International Conference on Computer Vision*.

Weyand, T. and Leibe, B. (2015). Visual Landmark Recognition from Internet Photo Collections: A Large-Scale Evaluation. *Computer Vision and Image Understanding (CVIU)*, **135**, 1–15.

Weyand, T., Hosang, J., and Leibe, B. (2010). An Evaluation of Two Landmark Building Discovery Algorithms. In *ECCV'10 Workshop on Reconstruction and Modeling of Large-Scale Virtual Environments*.

Weyand, T., Tsai, C.-Y., and Leibe, B. (2015). Fixing WTFs: Detecting Image Matches caused by Watermarks, Timestamps, and Frames in Internet Photos. In *IEEE Winter Conference on Applications of Computer Vision*.

Witkin, A. P. (1984). Scale-space filtering: A new approach to multi-scale description. In *IEEE Conference on Acoustics, Speech and Signal Processing*.

Yan, W.-Q., Wang, J., and Kankanhalli, M. S. (2005). Automatic video logo detection and removal. *Multimedia Systems*, **10**(5), 379–391.

Yang, L., Johnstone, J., and Zhang, C. (2011). Efficient place recognition with canonical views. In *International Conference on Multimedia & Expo*.

Yin, P., Hua, X.-S., and Zhang, H.-J. (2002). Automatic Time Stamp Extraction System for Home Videos. In *International Symposium on Circuits and Systems*.

Yu, X., Cheng, J., Song, W., and He, B. (2013). An Algorithm for Timestamp Removal for Panorama Video Surveillance. In *International Conference on Internet Multimedia Computing and Service*.

Yuan, X.-T., Hu, B.-G., and He, R. (2012). Agglomerative Mean-Shift Clustering. *IEEE Trans. on Knowledge and Data Engineering*, **24**(2), 209–219.

Zhang, W., Cao, X., Qu, Y., Hou, Y., Zhao, H., and Zhang, C. (2010). Detecting and Extracting the Photo Composites Using Planar Homography and Graph Cut. *IEEE Trans. on Information Forensics and Security*, **5**(3), 544–555.

Zheng, Y.-T., Zhao, M., Song, Y., Adam, H., Buddemeier, U., Bissacco, A., Brucher, F., Chua, T.-S., and Neven, H. (2009). Tour the world: Building a Web-Scale Landmark Recognition Engine. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Curriculum Vitae

Tobias Weyand

E-Mail	weyand@vision.rwth-aachen.de
Date of Birth	13.05.1982
Place of Birth	Siershahn, Germany
Citizenship	German

Education

Aug. 2009 – Oct. 2014	Doctoral student at RWTH Aachen University, Computer Vision Group Supervisor: Prof. Dr. Bastian Leibe
Oct. 2001 – Aug. 2009	Computer Science Studies at RWTH Aachen University Degree: Dipl. Inform.
Aug. 1992 – Oct. 2001	Secondary School Martin Butzer Gymnasium, Dierdorf Degree: Hochschulreife

Professions

Oct. 2014 – present	Software Engineer at Google in Los Angeles
Jul. 2013 – Oct. 2013	Intern at Google in Los Angeles
Sep. 2012 – Nov. 2012	Intern at Google in Los Angeles
Apr. 2009 – Aug. 2009	Student research assistant at Computer Vision Group, RWTH Aachen University
Nov. 2005 – Apr. 2009	Student research assistant at Human Language Technology and Pattern Recognition Group, RWTH Aachen University
Jul. 2006 – Aug. 2006	Intern at LTU Technologies in Paris
Oct. 2003 – Apr. 2005	Student research assistant at Computer Graphics Group, RWTH Aachen University

Publications

T. Weyand, B. Leibe, *Visual Landmark Recognition from Internet Photo Collections: A Large-Scale Evaluation*, Computer Vision and Image Understanding (CVIU), Vol. 135, pp. 1-15, 2015

T. Weyand, C.-Y. Tsai, B. Leibe, *WTF-Remove: Eliminating Watermarks, Timestamps, and Frames in Large-Scale Image Collections*, IEEE Winter Conference on Applications of Computer Vision (WACV), 2015

T. Weyand, B. Leibe, *Discovering Details and Scene Structure Places with Hierarchical Iconoid Shift*, International Conference on Computer Vision (ICCV), 2013

T. Sattler, T. Weyand, B. Leibe, L. Kobbelt, *Image Retrieval for Image-Based Localization Revisited*, British Machine Vision Conference (BMVC), 2012

T. Weyand, B. Leibe, *Discovering Favorite Views of Popular Places with Iconoid Shift*, International Conference on Computer Vision (ICCV), 2011

T. Weyand, J. Hosang, B. Leibe, *An Evaluation of two Automatic Landmark Building Discovery Algorithms for City Reconstruction*, Reconstruction and Modelling of Large-Scale Virtual Environments (RMLE) workshop at the European Conference on Computer Vision (ECCV), 2010

T. Weyand, T. Deselaers, H. Ney, *Log-Linear Mixtures for Object Recognition*, British Machine Vision Conference (BMVC), 2009

T. Gass, T. Weyand, T. Deselaers, H. Ney, *FIRE in ImageCLEF 2007: Support Vector Machines and Logistic Regression to Fuse Image Descriptors for Photo Retrieval*, Workshop of the Cross-Language Evaluation Forum (CLEF), 2007

T. Deselaers, T. Weyand, H. Ney, *Image Retrieval and Annotation Using Maximum Entropy*, Workshop of the Cross-Language Evaluation Forum (CLEF), 2006

T. Deselaers, T. Weyand, D. Keysers, W. Macherey, H. Ney, *FIRE in ImageCLEF 2005: Combining Content-based Image Retrieval with Textual Information Retrieval*, Workshop of the Cross-Language Evalution Forum (CLEF), 2005

S. Bischoff, T. Weyand, L. Kobbelt, *Snakes on Triangle Meshes*, Bildverarbeitung für die Medizin, 2005