Content-Based Refinement of Keyword-Based Image Retrieval

Popke Altenburg

May 2011

Master Thesis

Artificial Intelligence
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands

Internal supervisor:
Dr. Marco Wiering (Artificial Intelligence, University of Groningen)

External supervisor:
Mathijs Homminga (Kalooga, Nl)

university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Contents

Chapter 1

Introduction

There is a lot of research in the field of computer vision. In general, the field is concerned with deriving all kinds of information from analyzing a, most often digital or digitized, image. Subfields include object detection and recognition, face detection and recognition (which is a subfield of the former), and handwriting recognition. Computer vision is applied in, among others, consumer photography (face detection in cameras and photo analysis on photo sharing websites), biometrics for security (fingerprint analysis, hand recognition and face recognition), medical imagery (detection of cancer tissue) and various (other) industrial applications, like handwritten address recognition on envelopes, or automated categorization of plants and vegetables. It is a relatively new scientific field, in part due to the fact that it is computationally intensive, and it is only in the last decades that computational power has become more and more abundantly available.

1.1 Image retrieval

The subject of this project is keyword-based image retrieval. This task is concerned with the retrieval of images from a large database, given some kind of input, like a verbal query. Images are returned that are related to the input in order of relevance. With image retrieval that is keyword-based, the relevance of an image to the input is determined on the basis of verbal tags attached to the images in the database. This can be a list of keywords, for example a bit of text that was found on the webpage the image was originally found on, or by keywords that were manually added, as in stock-photography databases. Most image search engines, like Google's, currently are (at least part) keyword-based.

A different kind of image retrieval is content-based image retrieval, where the input to the image search is visual, like an image, or a drawing. In that case, the retrieved images are related to the input-image by visual similarity.

1.2 Problem definition

With keyword-based image retrieval, the set of returned images can consist of different kinds of images. There is a variety of reasons that the verbal tags in an image database can match with the input query. Different types of image can be related to a verbal search query, for example due

to ambiguity about the meaning of the search query keyword. For keyword-based image retrieval on data that is retrieved from webpages or webgalleries, it is often difficult to rank the results on relevance. This is because the keywords found in the proximity of the image can for instance be added by the creator of the webpage, and so are not added necessarily for the purpose of image retrieval. This can cause the problem of the presence of images in the results that are not related to the input query. Within the context of this thesis, they will be called outliers. For this project, the goal is to identify these outliers. When outliers can be identified they can be removed from the results or ranked lower. The latter will most likely be better, as there is not always a clear definition of 'relevance' to a keyword.

An image can have different kind of properties that can be useful for determining the relevance to a search query. Besides the verbal tags linked to the image itself, there are properties like information about the webpage on which the image is found, or meta-data about the image inside the image file. For this project we consider the visual content of the returned images. We make the assumption that visual content is related to semantic meaning, and that relevant images share similarities. Outlier images could also have similarities among them, but they are assumed to share less similarities than the relevant images. The concept of outlier is defined as an image that has very few similarities to the rest of the images in the dataset.

1.2.1 Research question

The main research question is:

- Is it possible to use image content to identify outliers in a set of keyword-based retrievals?

This is approached using methods from the field of computer vision. By pairwise comparison of images in the set of retrieved images, clusters are being detected. Based on the cluster sizes, and whether images are members of a certain cluster or not, outliers are found in a group of keyword-based retrievals.

This leads to the sub-questions: What features in images are most descriptive of their image class, and discriminative between image classes? Can groups of images be efficiently clustered according to their extracted features? And how well can this be used for detecting outliers?

As was said above the focus will be on images in the real-world domain. Therefore it is important to have a strong generalizing ability within classes.

1.3 About the project

The project was done at the company Kalooga, situated in Groningen, NL. Kalooga is a company with fifteen to twenty employees, that is focused at advertising through visual content at third-party websites. Kalooga has a database of thumbnails from images that are gathered from photo-gallery websites publicly available on the internet. Currently, the database contains images from approximately twenty million photo galleries. At their frontpage, at kalooga.com, the company hosts an image search engine, where searches through the database can be executed.

Because the images that are searched are gathered from all kinds of different types of photo galleries publicly available on the internet, the problem of false positives in the search results is one of their concerns.

1.4 Outline

In chapter two, we will discuss existing methods from scientific literature that are used for this project. After that in chapter three we will discuss and motivate our approach. In the fourth chapter we will discuss the performance of the methods based on results of experiments. In chapter five we will analyse the results and conclude.

Chapter 2

Computer vision

In part due to the increased computational power available in the last decades, there is a lot of scientific research in the field of computer vision. Examples of fields of research are text recognition, face detection and recognition, image classification and object recognition. In this thesis, the focus is on image classification.

For this project, certain existing methods from computer vision are used, as well as methods from the field of pattern recognition in general. We will describe methods used for keypoint detection and description, and visual image comparison. In this chapter these methods will be discussed separately, and in chapter 3 we will refer back to them.

2.1 Keypoint based methods

In computer vision, keypoints (also called salient points, interest points, feature points or simply features) are often successfully used. Applications include: Matching of stereo pair images [Tuytelaars and Van Gool, 2000], content based image retrieval [Smeulders et al., 2002], object recognition [Lowe, 2004] and the like.

Keypoints are points in the image that are sparsely distributed over the image, and that have a unique visual quality. For example, often they are corners, or crossings, but they are generally not detected at lines or evenly colored patches, depending on the specified type of keypoint based method. The fact that keypoints are sparsely detected over the image is part of its purpose: the amount of data one has to work with is reduced from the entire image to a few specific coordinates, or patches that hopefully contain information of interest. A keypoint method is repetitive if keypoints on similar images are located at similar coordinates. A complete keypoint based method consists of both a detection step, as well as a description step. We will explain the two steps below.

Detection
With the detection step, the unique points in the image are identified and given at least the location of the keypoints and often additional information, like its scale and rotation. The rationale behind using keypoints is that they are generated using the same process in different images, so that the same object in different images contains the same (of similar) keypoints. This makes it easy for object detection algorithms to analyze images to find objects. Keypoint detection algorithms differ in the way the keypoints are generated. Stable algorithms are considered those that find

the keypoints on the same locations on objects in different images. Keypoints can be detected in images in a number of different ways. The key idea is that the locations of those keypoints are in a way consistent with a pattern in pixel values over different images, so that in different images with the same kind of patterns, the keypoints are detected at similar locations between images. Most modern keypoint detection methods are able to find keypoints that are invariant to scale, rotation and more, so called affine transformations, so that the same pattern can be detected under a variety of circumstances, like different viewpoints of the same object. For this project we use the Harris-Affine keypoint detection, described in section 2.1.2.

Description
With the description step, a representation is given of the image in a certain area around the keypoint, given its location, and often given its scale and rotation too. Often this description is an array of numbers that is derived in such a way that the rectilinear- or Euclidean distance between two of such arrays is a measure for the difference in visual appearance of its two keypoints. Some methods give a two-dimensional histogram of the gradients around the keypoint, of which the histogram-bins are arranged in a predefined pattern. For instance, the SIFT feature [Lowe, 2004] differs with the GLOH feature [Mikolajczyk and Schmid, 2005] in that with the former, the bins are distributed in a rectangular grid array, and with the latter, the bins are distributed circularly around the keypoint.

The use of keypoints is successful in tasks where patterns have to be found that are more or less exact matches, like finding correspondences in stereo images, or finding exactly the same visual objects in images, like book covers. However for finding matches in groups of images that are less identical, they are less useful. This is the case with finding images of a certain type of general object class, like a species of animal. Despite the fact that generalizing over a generic type of object class is one the largest problems this project has to overcome, the use of keypoint based methods is still found useful.

For this project, the widely used histogram of gradients descriptor from the SIFT method is used. Despite the fact that the GLOH-descriptor may perform better we do not find that the difference is very large, so we prefer the more generally used SIFT variant. The SIFT-descriptor will be explained in section 2.1.1.

Orientation gradient based methods
A lot of methods are based on finding and describing keypoints based on gradients in the lightness values of the image rather than the lightness values themselves. Instead of looking at the values of the pixels themselves, the first order derivative of the image is taken in both dimensions.

Many different keypoint detection and description methods exist, which perform differently on different aspects. Here we will discuss a few of them that have proven in literature to perform well [Tuytelaars and Mikolajczyk, 2008]. The decision of which methods are selected for this project are based on reported performances in literature.

2.1.1 SIFT keypoint detection and description

A popular keypoint detection and description method is SIFT. It is very useful for image based retrieval, like finding a specific object in an image. It is invariant to scale and rotation, and robust to a small degree of distortion. The SIFT feature is presented by Lowe in [Lowe, 2004], and became popular due to its speed and performance.

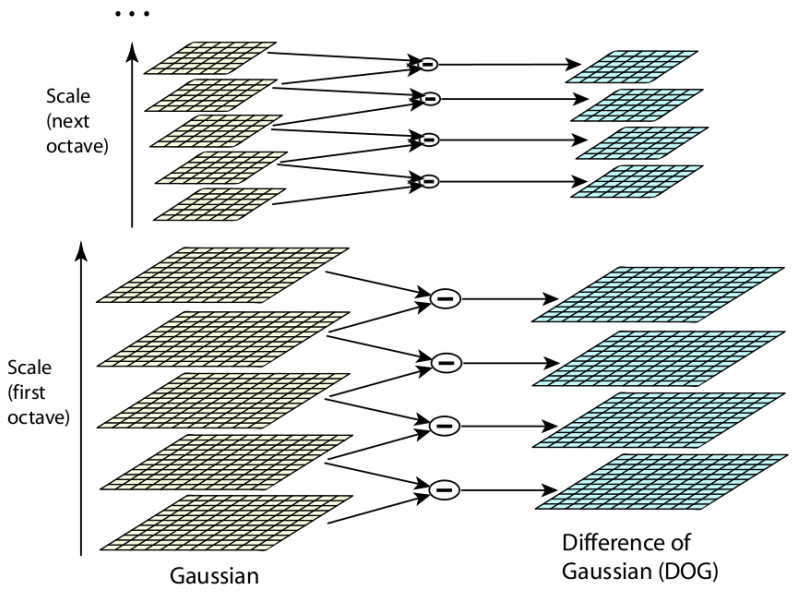


Figure 2.1: Sift keypoints are detected based on peaks in the 'Difference of Gaussians' data. Image taken from [Lowe, 2004].

For keypoint detection, SIFT uses the Difference of Gaussians detector, or DoG for short. DoG was first proposed in [Crowley and Parker, 1984]. This is a fast method and a close approximation of the Laplacian [Lowe, 2004]. In a comparison by [Mikolajczyk and Schmid, 2005], the scale-normalized Laplacian of Gaussians was found to produce the most stable image features. This is one of the reasons SIFT is fast, and can be used for realtime video analysis with multiple frames per second.

DoG works as follows: The image is convolved with Gaussian filters of different sizes, creating images with different scales of detail. Each pair of images with successive Gaussian filter sizes, are combined into a DoG image. This is done by subtracting the image with the smaller Gaussian filter from the one with the larger. The resulting DoG-images are stacked, and combined into a scale-space as illustrated in figure 2.1. In these images, local extrema are extracted, in both dimensions of the image, and the scale dimension at once.

Often, these local extrema are located at edges, which are unstable. Another filtering step is needed to avoid high responses at edges. At edges the principal curvature is large in one direction, and small in the perpendicular direction. So at edges, the ratio between the two principal directions is larger than at (more stable) corners. The eigenvalues of the Hessian matrix are proportional to the principal curvatures, and their proportion is computed from the trace and the determinant of this Hessian matrix [Harris and Stephens, 1988]. If this proportion is above a certain threshold, it is discarded.

From the remaining points, the orientation is determined. This is done so the descriptor can be normalized for its rotation, and thereby made rotation invariant. From sample points within a region around the keypoint the gradient magnitude and orientation is computed. Of these, the

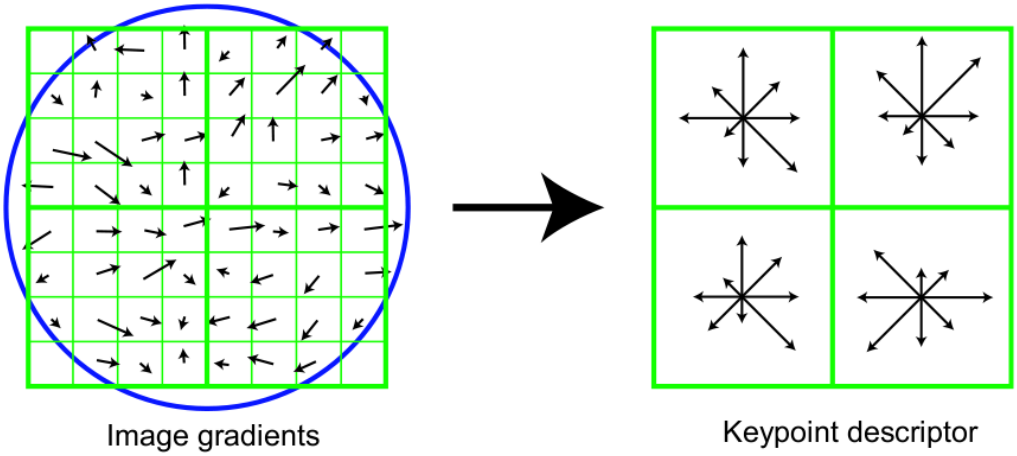


Figure 2.2: Illustration of the SIFT descriptor. The gradient magnitudes are weighted by a Gaussian window, indicated by the overlaid circle. In this example the region of oriented gradients is described by a histogram width of 2x2 subregions, whereas [Lowe, 2004] reports best performance with a 4x4 array of subregions, which is also used for this project. Image taken from [Lowe, 2004].

weighted average is taken, using a Gaussian window.

The keypoint descriptor from the SIFT-method is built up from the gradient magnitudes and orientations at each image sample point from a rectangular region around the keypoint. This rectangular region is scaled and oriented according to the determined scale and orientation of the detected keypoint. A histogram is created, describing the orientations in a 4x4 pattern of subregions. The orientations in each subregion are described by 8 histogram bins. This results in a vector of 128 values, describing the region around the keypoint. The orientation magnitudes, sampled from the image, are weighted by a Gaussian window, with a standard deviation equal to one half of the width of the descriptor region, to avoid sudden changes in the descriptor due to values near the edge, and to assign less emphasis on gradient magnitudes further away from the coordinates of the detected keypoint. To avoid boundary effects, within bins, the contributing value of each entry in the bin is multiplied by $1 - d$, where d is the distance of the sample from the central value of the bin, as measured in units of the histogram bin spacing. See figure 2.2 for an illustration.

2.1.2 Harris-Affine keypoint detection

For this project, Harris Affine keypoint detection is used. In the past years, there have been a number of comparison studies on keypoint detection methods [Schmid et al., 2000], [Mikolajczyk et al., 2005]. In these studies, different methods of finding interest points are compared and evaluated on different qualities. In [Schmid et al., 2000] five different keypoint detection methods are evaluated on repeatability and information content. In [Mikolajczyk et al., 2005] six different affine invariant keypoint methods are compared under different image conditions. Images were varied in viewpoint, scale, illumination, defocus and compression. A repeatability measure is taken to evaluate their

performances. It is concluded that the Hessian-Affine and Harris-Affine methods perform best. In [Mikolajczyk et al., 2005] it is shown that the Harris-keypoint detector performs well on stability and repeatability. This is demonstrated by applying the method to images of various scenes, shown from different viewpoints. According to a comparative study by [Schmid et al., 2000] it is considered most repeatable and most informative.

The Harris-Affine keypoint detector is based on the corner detector by [Harris and Stephens, 1988]. As an initial step, the Harris corner detector is used to extract keypoints from the image. It is based on the so called second moment matrix M, also called auto-correlation matrix. This matrix has the property that its eigenvalues represent gradients in two orthogonal directions in a neighborhood around a given point. If both eigenvalues are large, that means there are strong image gradients in orthogonal directions, which is usually the case around an image structure like a corner, junction or other points with high curvatures. The second moment matrix M is defined as

$$M = \sigma_D^2 g(\sigma_I) * \begin{pmatrix} I_x^2(\mathbf{x}, \sigma_D) & I_x I_y(\mathbf{x}, \sigma_D) \\ I_x I_y(\mathbf{x}, \sigma_D) & I_y^2(\mathbf{x}, \sigma_D) \end{pmatrix} \tag{2.1}$$

where

$$I_x(\mathbf{x}, \sigma_D) = \frac{\delta}{\delta x} g(\sigma_D) * I(\mathbf{x}) \tag{2.2}$$

$$I_y(\mathbf{x}, \sigma_D) = \frac{\delta}{\delta y} g(\sigma_D) * I(\mathbf{x}) \tag{2.3}$$

and

$$g(\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.4}$$

Here, $\mathbf{x} = (x, y)$ and $I(\mathbf{x})$ denotes point $\mathbf{x}$ by its lightness value. This matrix describes the gradient distribution in a local neighborhood of a point. The local image derivatives are computed with Gaussian kernels of scale σ_D (the differentiation scale). The derivatives are then averaged in the neighborhood of the point by smoothing with a Gaussian window of scale σ_I (the integrations scale).

[Harris and Stephens, 1988] define a measure for *cornerness* as follows:

$$cornerness = \det(M) - \lambda \times trace(M) \tag{2.5}$$

where $\det(M)$ is the determinant of M, $trace(M)$ is the trace of M, and with λ a typical value of 0.04. This measure is large when both eigenvalues of M are large, and this way, the eigenvalues do not need to be computed, which makes it faster. For each point in the image this *cornerness* is computed, and local maxima are taken as the detected keypoints.

Next a characteristic scale is determined for a keypoint. This is done so the keypoint can be normalized for scale to make it scale-invariant. The keypoints are taken at different versions of the image, each smoothed with Gaussian filters of increasing size, which determines its scale. Often the same keypoint is detected at multiple scales, so a measure is needed to determine the characteristic scale. In [Mikolajczyk and Schmid, 2002] it is shown experimentally that the Laplacian derivative function gives the best results for scale selection. Its function response is taken at all scales, and the scale that corresponds with extrema of this value is taken as the characteristic scale.

This was extended with invariance for other affine transformations in [Mikolajczyk and Schmid, 2002]. The affine shape of a keypoint is estimated using the second moment matrix, and then its region is normalized to a circular one. For this normalized region the new location and scale are detected. These steps are repeated until the eigenvalues of the second moment matrix are equal. With this method, the keypoints are invariant for affine translations, such as scale, rotation, aspect ratio and skew.

2.1.3 Others

Another notable keypoint detector is MSER [Donoser and Bischof, 2006]. This is more patch-based than gradient-based, so it can be used to complement gradient based detected keypoints.

2.2 Image correspondence problem

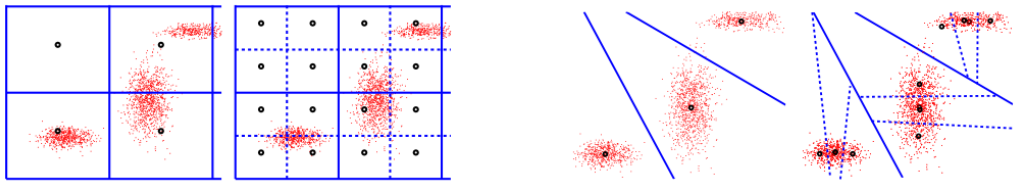
In order to find groups of images within a set, a measure for similarity between images is needed. How this measure is determined is subjective, and could be done in many ways. For our purpose, it should relate to the semantic content of the image. Methods that approach this problem are discussed in, among others, [Grauman and Darrell, 2006] and [Kim et al., 2008]. Different methods exist that are used to give a description of similarity between a pair of images. For this project, we have selected two of these methods that have been shown to give promising results. We will compare them for performance and speed is our experiments in chapter four.

2.2.1 Pyramid matching

In [Grauman and Darrell, 2005] pyramid matching or intersection is proposed to match pairs of images. Like this project, their method is also based on keypoint analysis of the images. In [Grauman and Darrell, 2006] they show promising results on experiments using a subset of the Caltech 256 dataset [Griffin et al., 2007], so we use it for this project.

Their matching method measures similarity between unordered, variable sized sets of vectors. It can be used as a kernel function. Keypoint descriptors are extracted from the images and the images are represented as the set of their feature descriptors. The descriptors have a fixed length, but the number of descriptors per image may vary. Because the descriptors have a fixed dimension d, we define them as points in d-dimensional space $\mathcal{V}$. Each image is represented as a set of points. We define an input space S, which contains sets of feature vectors of all images drawn from $\mathcal{V}$: $S = \{\mathbf{X} | \mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}\}$, where each feature $\mathbf{x}_i \in \mathcal{V}$, and $m = |\mathbf{X}|$.

As a next step, these points are converted into histograms, using hierarchical decomposition of the feature space. The histograms are created at different levels of detail, and are called multi-resolution histograms. The feature space is partitioned into a pyramid of non-uniformly shaped regions based on the distribution of the given data. So the bins in the pyramid follow the distribution of the keypoint descriptors in the set. This is done with hierarchical clustering. Hierarchical K-means clustering is used as the clustering method, and for each level in the pyramid, or tree, a random subset of the data is used for speed optimization. Each level is clustered into a number of clusters, k, which is the branching factor. After the top level is clustered, each cluster is recursively divided into k new clusters, until the maximum number of levels L is reached, or each point is assigned to a leaf in the tree. This is called a vocabulary guided pyramid, illustrated in figure 2.3 and is a one-time process. Using these divisions of the feature space, pointsets are then converted to



(a) Uniform bins (b) Vocabulary-guided bins

Figure 2.3: Illustration of vocabulary guided bins. Image taken from [?]

multi-resolution histograms. Each keypoint descriptor (that is a point in the pointset), contributes to the count of one histogram bin at each consecutive level. We define a histogram pyramid for input example $\mathbf{X} \in S$ as:

$$\Psi(\mathbf{X}) = [H_0(\mathbf{X}), \ldots, H_{L-1}(\mathbf{X})] \tag{2.6}$$

where $H_i(\mathbf{X})$ is a histogram vector over points in $\mathbf{X}$. The bins increase in size from the finest-level histogram H_0 to the coarsest-level histogram H_{L-1}.

From any two histogram pyramids, the approximate matching score is computed using a weighted intersection measure. Beginning at the bottom leaves of the pyramid, the number of points that share the same bin is counted. At each level upwards, only the new matches are counted, so that each point from one point set is matched against the point of the other point set only once, at the smallest bin that they share. The intersection $\mathcal{I}$ between two histograms is defined as

$$\mathcal{I}(P, Q) = \sum_{j=1}^{r} \min\{P_j, Q_j\} \tag{2.7}$$

where P and Q are histograms with r bins, P_j denotes the count of the j-th bin in P and Q_j the count of the j-th bin in Q. Consequently, the pyramid match similarity score $\mathcal{P}_\Delta$ between two input sets $\mathbf{Y}$ and $\mathbf{Z}$ is defined as

$$\mathcal{P}_\Delta(\Psi(\mathbf{Y}), \Psi(\mathbf{Z})) = \sum_{i=0}^{L-1} w_i \Big(\mathcal{I}(H_i(\mathbf{Y}), H_i(\mathbf{Z})) - \mathcal{I}(H_{i-1}(\mathbf{Y}), H_{i-1}(\mathbf{Z})) \Big) \tag{2.8}$$

where the number of new matches at each level i is weighted by $w_i = \frac{1}{d2^i}$, with d the dimension of descriptor space $\mathcal{V}$. See figure 2.2.1 for an example of matching of histogram pyramids.

This matching can be done in linear time, whereas optimal partial matching requires at least cubic time. The authors claim that their method works well with over ten dimensions, but they remain vague about normalizing results based on different numbers of keypoints. In their own experiments equal numbers of keypoints are used for two images.

2.2.2 Spectral matching and link analysis

In [Kim et al., 2008] a method is proposed to discover object categories from a set of images and learn their visual models. Their method is also based on image features in the form of keypoints. They claim it is unsupervised, however the number of categories does have to be supplied by the

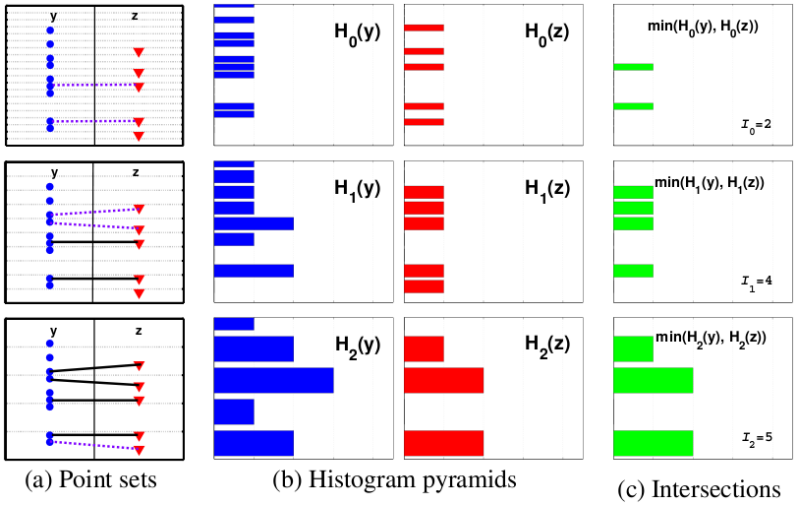


Figure 2.4: Illustration of multi-resolution histogram matching. Image taken from [Grauman and Darrell, 2005]. Two 1-D feature sets are used to form two histogram pyramids. Each row corresponds to a pyramid level, which is numbered 1,2 and 3 from top to bottom. In (a), set $\mathbf{Y}$ is on the left and set $\mathbf{Z}$ is on the right and points are distributed along the vertical axis. Light lines are bin boundaries, bold dashed lines indicate a new pair matched at that level, and bold solid lines indicate a match already formed at a finer resolution level. In (b), the multi-resolution histograms are shown and (c) shows the intersections $\mathcal{I}_i = \mathcal{I}(H_i(\mathbf{Y}), H_i(\mathbf{Z})) = \min\{H_i(\mathbf{Y}), H_i(\mathbf{Z})\}$. Here, $\mathcal{I}_i$ equals 2, 4 and 5 for the three levels 1,2 and 3 respectively. Consequently, the number of new matches counted for level 1 is 2, for level 2: $4 - 2 = 2$, and for level 3: $5 - 4 = 1$.

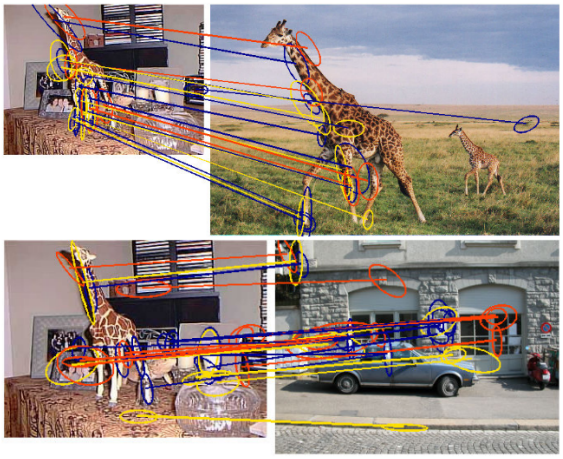


Figure 2.5: An example of matching an image to two other images, one of the same class, and one of a different class. Link color denotes quality of match. Only the top pair consists of correct matches. If consistently matched with more images of the same class, the link analysis techniques extract the correct matches from the false and inconsistent ones. Image taken from [Leordeanu and Hebert, 2005].

user. In [Kim et al., 2008] the output of the method is the set of discovered object categories, but for this project we use it to construct a similarity matrix of the images, so we omit the 'category discovery' and 'localization' steps in their article.

A large-scale graph is constructed capturing the interactions between all features across the entire image set. It is referred to as the *visual similarity network*, or VSN. Link analysis techniques are applied to this graph. This is done in order to find features that are matched a lot to features in other images. If a set of images contains images of the same visual category, a feature with a lot of matches to features in other images is considered important for its category. Using link analysis techniques is inspired by its use for internet search engines, where useful information is extracted by analyzing the interaction of elements, namely pages linking to other pages, rather than the information contained by the elements themselves. In [Kim et al., 2008], the PageRank [Brin and Page, 1998], and the vertex similarity algorithms [Blondel et al., 2004] are used as link analysis techniques. The graph consists of the pairwise relations between all features in the image set, but it is very sparse, so it is not unusable.

Visual similarity network

The set of images is denoted by $\mathbb{I}$. The VSN, the graph containing the feature interactions is represented by a graph $G = (V, E, W)$, where V is the set of vertices, E is the set of edges, and W the set of edge weights. The adjacency matrix of the VSN is denoted by M. Each node is denoted by a_i, representing the i-th keypoint, or feature, in image I_a in $\mathbb{I}$. The total number of keypoints is denoted by n, and the number of keypoints in image a by n_a.

Establishing the edges of the VSN

The edges in E between the nodes are established using spectral matching [Leordeanu and Hebert, 2005]. Using this method, the weight w_e in W of each edge e in E between two nodes in V is a result of (1) the individual match of the keypoints of both nodes and (2) the geometric consistency with the other keypoint matches in the same pair of images.

The method works as follows: For each pair of images in $\mathbb{I}$, a weighted adjacency matrix Q is built. Q is the adjacency matrix of the graph that has all potential feature assignments as nodes. It has one row and one column for each potential assignment, (a_i, b_j). Strongly connected clusters in this graph tend to be clusters of correctly assigned feature points. The problem amounts to finding the set of assignments that maximizes a defined inter cluster score. A spectral approach is used for finding the main clusters in this graph. First the eigenvector of M corresponding to its largest eigenvalue, the principal eigenvector, is taken, and then assignments of low association are rejected until the predefined (geometric of otherwise) constraints are met. For ease of notation, we abbreviate (a_i, b_j) by ij.

For each candidate assignment ij there is an associated score or affinity that measures how well both features match. This value is stored in Q at position $Q(ij, ij)$. Depending on this value, a lot of candidate assignments can be rejected initially, so Q is generally very sparse. [Leordeanu and Hebert, 2005] suggests using a threshold for this, but in [Kim et al., 2008] the top k of best matching features are used, which we will follow in our implementation. For k we use a value of 5. In [Leordeanu and Hebert, 2005] a score is used that is linearly decreasing with the L_2 distance between the two keypoint descriptors, but specifics are not given. We implement this as a linearly decreasing value between 0 and 1:

$$Q(ij, ij) = 1 - \frac{|\mathbf{d}_{a_i} - \mathbf{d}_{b_i}|}{\sqrt{m^2 \times |\mathbf{d}_{a_i}|}} \tag{2.9}$$

where $\mathbf{d}_{a_i}$ indicates the descriptor vector associated with keypoint a_i, and m is set as follows: $Q(ij, ij)$ should be between 0 and 1, so with the denominator, $\sqrt{m^2 \times |\mathbf{d}_{a_i}|}$ we indicate a value that is never lower than the numerator of the equation above. $|\mathbf{d}_{a_i}|$ indicates the number of values in a descriptor, and we could set m to the maximum value that a descriptor value can take, but we set it to 500, for which the mentioned denominator also never gets higher than the numerator, and the values are better spread over the range.

$Q(ij, i'j')$ is set to the pairwise geometric consistency score between correspondences ij and $i'j'$ such that $Q(ij, i'j')$ is lower if more deformation is needed to map the pair ij to the pair $i'j'$. These affinities are required to be non-negative, symmetric ($Q(ij, i'j') = Q(i'j', ij)$), and increasing with the quality of the match. The pairwise score is defined as

$$Q(ij, i'j') = \begin{cases} \frac{1}{4.5}\left(4.5 - \frac{(d_{ii'} - d_{jj'})^2}{2\sigma_d^2}\right) & \text{if } |d_{ij} - d_{i'j'}| < 3\sigma_d \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

where $d_{ii'}$ equals the Euclidean distance between the locations of keypoints a_i and $a_{i'}$. This is analog to the definition used in [Leordeanu and Hebert, 2005], but scaled between 0 and 1.

Next, the set, or cluster, of assignments C is extracted that maximizes the inter-cluster score $S = \sum_{ij,i'j' \in C} Q(ij, i'j')$ such that certain *mapping constraints* are met. Mapping constraints are rules that are defined such as allowing a keypoint from one image to match at most one keypoint from another image, or allowing a keypoint from one image to match more keypoints from another image. We use the mapping constraint that a keypoint can match at most one keypoint from another image, and that a keypoint can only be matched by at most one keypoint from another

image. Given two correspondences ij and $i'j'$, this mapping constraint is met only if $i \neq i'$ and $j \neq j'$.

We represent cluster C by an indicator vector x, such that $x(a) = 1$ if $a \in C$ and zero otherwise. The inter-cluster score S can then be rewritten as

$$S = \sum_{(ij),(i'j') \in C} Q(ij, i'j') = x^T M x \tag{2.11}$$

The optimal solution x^* is the binary vector that maximizes the score, given the mapping constraints:

$$x^* = \text{argmax}_x(x^T M x) \tag{2.12}$$

According to [Leordeanu and Hebert, 2005], the x^* that will maximize the inter-cluster score $x^T M x$ is the principal eigenvector of Q. $x^*(ij)$ is interpreted as the *association* of assignment ij with the best cluster C^*. Otherwise put, $x^*(ij)$ is interpreted as the confidence that ij is a correct assignment. Therefore, x^* is set to the principal eigenvector of Q.

The next step is to derive which elements are in cluster C so that the mapping constraints are met. This is done by iterating over the following steps:

1. Initialize cluster C as an empty set. Let L be the set of all candidate assignments.
2. Find $(ij)^* = \text{argmax}_{(ij) \in L}(x^*(ij))$. If $x^*((ij)^*) = 0$ stop and return cluster C. Otherwise add $(ij)^*$ to C and remove it from L.
3. Remove from L all potential assignments in conflict with $(ij)^* = (a_i, b_j)$ according to the mapping constraints. These assignments are of the form $(a_{i'}, b_{j'})$, where $a_{i'} = a_i$ or $b_{j'} = b_j$.
4. If L is empty, return C. Otherwise go back to step 2.

This returns the resulting set of correspondences. An estimate of the confidence of $(a_i, b_j) \in C$ is given by: $C_{ij} = \sum_{i'j' \in C} Q(ij, i'j')/|C|$. The correspondencies in C are taken as the edges of the VSN G between images $\mathbf{I}_a$ and $\mathbf{I}_b$. Each value C_{ij} in C is stored in the adjacency matrix M of G at $M(a_i, b_j)$, where $M(a_i, b_j)$ is the value of weight w_e of the edge $e = (a_i, b_j)$. This pairwise matching is done for each image pair, so that edges are established between keypoints of all images in $\mathbb{I}$.

Pairwise image matching

Based on the feature interactions the affinity matrix A is build, depicting the pairwise image similarity values. As a mechanism to estimate the relative importance of nodes in the graph, the PageRank algorithm [Brin and Page, 1998] is used. This algorithm generates a $n \times 1$ PageRank vector P by solving the equation:

$$P = (1 - \alpha)(M^T + D)P + \alpha v, \tag{2.13}$$

where M is the weight matrix of the VSN G, α is a constant close to zero (following [Kim et al., 2008], we use $\alpha = 0.1$), v is the *transport* vector ($= [\frac{1}{n}]_{n \times 1}$, uniform probability distribution over all nodes), and $D = vd^T$ (with d being the n-dimensional indicator vector identifying the nodes with outdegree 0). A PageRank vector P is initialized with $[\frac{1}{n}]_{n \times 1}$, and equation 2.13 is iterated until convergence.

For each image $\mathbf{I}_a$, a PageRank vector P_a is created, using the portion of M with only features from $\mathbf{I}_a$. This is done by setting $M(ij)$ to zero if both $i \notin \mathbf{I}_a$ and $j \notin \mathbf{I}_a$. According to [Kim et al., 2008], for all $i \notin \mathbf{I}_a$, a large $P_a(i)$ indicates that i is a highly relevant feature with respect to image $\mathbf{I}_a$.

Using the PageRank vectors, the affinity matrix A is created. The affinity, which we use as similarity measure, between two images I_a and I_b is defined as $A(a, b) = \sum_{b_j \in \mathbf{I}_b} P_a(b_j)$.

2.3 Cluster detection

The problem with the use of data that has no attached category data is one of unsupervised classification. This is also the case with the data that is used for this project, that is the set of retrieved images, in which outliers have to be identified. Cluster detection will be used to generate supposed classes, and outliers will be identified based on the connection to these classes, and the strengths of these connections. The cluster detection is done based on the pairwise similarities between images. How this is done exactly is thoroughly discussed in chapter 3.

Chapter 3

Methods

In this chapter we discuss the approach to the problem as defined in section 1.2.1. Our approach to outlier detection is as follows: Groups of images representing the same concept are searched for in a set of retrievals. If an image is not a member of a group, or if the strength of its membership is low enough, it is considered an outlier.

Therefore, a way to find groups defining a concept is needed. For this project this is approached by detecting clusters based on the pairwise similarity between images, about which we make the following assumption:

- If a set of images exists within which the pairwise similarities are high, then the cause for this is that the images in the group are of the same class.

This assumption is of course not very restrictive, as the truth of the assumption depends on the definition of the concepts stated in it, like the way similarity between two images is defined exactly. How we define this similarity is again dependent on the above assumption, but this circular dependency need not be a problem. Also it is to be noted that the above assumption does not restrict the same visual concept to have different groups of images (for instance there can exist both a group of images of cars facing sideways, and one of cars facing the front, which both represent the concept 'car').

Because concept detection is not the main focus of this project, we do not address the possibility of hierarchy between concepts (like an image of a taxicab is a member of the concepts 'taxicab', 'car' and 'wheeled vehicle').

So in order to define groups, a definition of similarity is needed. For this definition we define the following properties to hold:

- As stated above, the similarity should reflect membership of concepts or category of both images.
- The similarity can only be based on visual qualities of the image (the pixel values), instead of, for example, added keywords or metadata.

The first is needed to effectively search for concepts by looking for clusters in the set of images. The latter is needed because it is the aim of this project to investigate if and how this task can be done based on purely visual information. What remains an issue, is what level of generalization

should be assigned to the similarity measure. If the level of generalization is too low, the similarities between the different images are too low to find large clusters in a group of images with high visual variance. The similarity measure should generalize over images of the same class, but in order to do this purely based on visual information, the images over which is generalized should share certain visual qualities. Of course it is not said that these properties are always true simultaneously for all types of image, but it is intuitive to say that images of the same concept or category are visually similar.

As said in chapter 2 the problem of similarity is approached by using keypoints and their descriptions. Even if images of the same concept are not exactly visually similar, we still expect them to share certain keypoint features. Hence, the definition of similarity between two images is based on:

- The similarity of keypoints between them, and the number of similar keypoints.
- The geometric similarity of the coordinates of the keypoints.

In this chapter we will elaborate on the above stages of the method.

3.1 Overview

The outline of the proposed method is as follows. See figure 3.1 for a schematic overview.

First, from all images, the keypoints and their descriptions are extracted. Based on these keypoints, the images are compared to each other in order to define a similarity value between each pair of images. These similarity values are used for two things:

- The images are ranked in order to give images that have a higher chance of being an outlier, a lower ranking.
- Where possible, we try to detect clusters of images that are visually similar. That way each image can be labeled as being an outlier or not.

The result of the former is a sequence in which the images are positioned. This can then be used for image search, to place images that have a lower chance of being an outlier, higher in the list of search results. The result of the latter is a labeling of the images, that can be used in image search to exclude images from search results.

3.2 Image similarity

As a first step, of each image the keypoints are extracted. For this project we use a number of 100 keypoints per image with the highest saliency measure. For the Harris-Laplace keypoints this saliency measure is the *cornerness*-value as defined in the previous chapter. For each keypoint a descriptor is built as used in the SIFT-method. Next, for each pair of images, their similarity is computed. For this we use both pyramid matching and matching using link analysis, as discussed in chapter 2.

Both methods give as result a set of pairwise similarity values between all images.

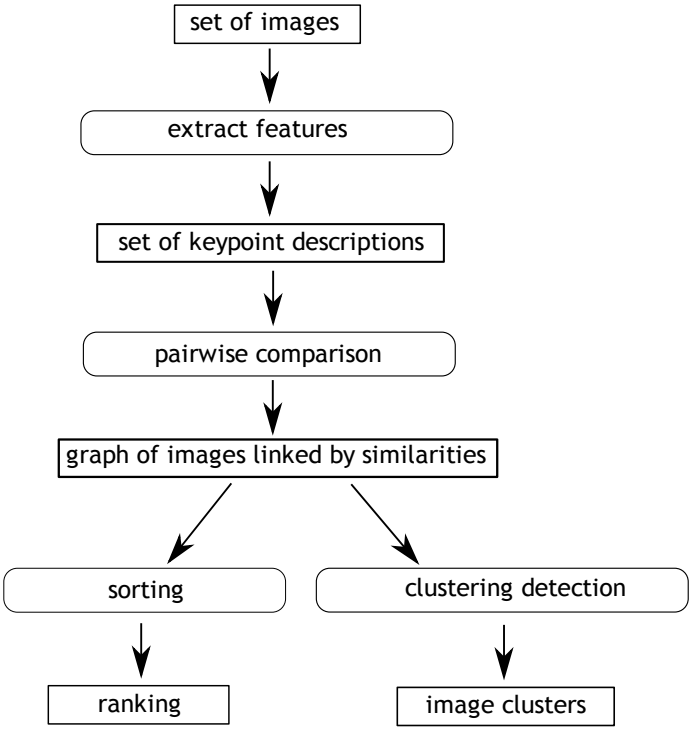


Figure 3.1: Schematic outline of the proposed method.

3.3 Sorting of images

Using the pairwise similarity data, the group of images is sorted, so the images that are outliers will be ranked lower. The goal is to place images that are important to the search query high in the list of results, and the outliers low. To sort the images, they are assigned a value based on which they are sorted. A higher value should express a higher chance that the image is related to the search query instead of being an outlier. Different ways of determining this value are used as discussed in the next sections.

3.3.1 Sum of similarity values

The value we assign to an image to sort the group of images is determined as follows: For the given image we add up all the similarity values to the other images and take this as the sorting value for that image. The intuition is that images that are outliers have less resemblance to other images in the group, so the total of the similarity values to other images is low.

Let $\mathbb{I} = \{I_a\}_{a=1,\ldots,m}$ be the set of images of size m, and let S be the matrix where all similarity values are stored in. $S(a, b)$ then denotes the similarity value between images I_a and I_b. The sorting value v_a for an image I_a is then defined as:

$$v_a = -S(a, a) + \sum_{b=1}^{m} S(a, b) \tag{3.1}$$

See figure 3.2 for an example of a sorted list of images. The images are shown with their similarity values to illustrate how the similarity values contribute to how the list of images is sorted.

3.3.2 Density measure

A more local way to do this is to only consider images with high similarity to a given image. This is used as a density measure. The density measure is determined by taking the closest n images to a given image, and taking the sum of those similarity values as density measure. Here, we take the inverse of the similarity as a measure of distance.

Let $d(a, b) = \frac{1}{S(a,b)}$ be the distance measure between images I_a and I_b, and let $\mathbf{c}_a(b) = (i_1, \ldots, i_{m-1})$ be the list of image indices sorted by $d(a, i_b)$, the distance of image I_{i_b} to image I_a, with $I_{\mathbf{c}_a(1)}$ being the closest to I_a and $I_{\mathbf{c}_a(m-1)}$ the furthest and $a \notin \mathbf{c}_a(b)$. The sorting value v_a for an image I_a then becomes:

$$v_a = \sum_{b=1}^{n} S(a, \mathbf{c}_a(b)) \tag{3.2}$$

where n is the parameter denoting the number of images used for the density measure. The intuition is that with images in high density areas there are more images that are visually similar, so that they are less likely to be an outlier. When sorting using this value, images with low similarity to the image to be sorted have little to no influence.

3.4 Cluster detection

Based on pairwise similarities between images we try to detect a cluster within the set of images. The images in a cluster should have high similarity to other images within that cluster, and less

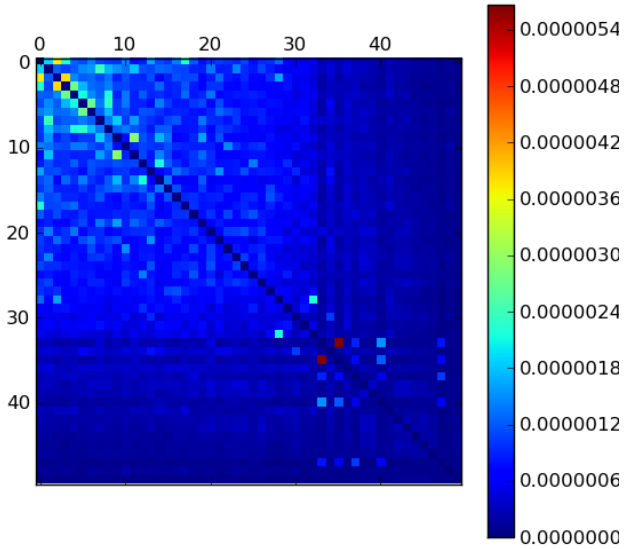


Figure 3.2: Example of the sequence of a sorted list of images, shown with their similarity values. This example is of category 'apple1', pyramid matching is used for similarity matching, and sorting is done by the sum of similarity values. In this experiment, 30 images are of class 'apple1'. The lower the numbered position in the sequence, the higher an image is ranked.

similarity to images outside of it.

The cluster detection is done as follows: For each image we determine if it is a member of a cluster. This is done by looking at the similarities with the rest of the images. If there is a group of images with a clearly stronger connection within the group than to other images, it is considered a cluster. We use the following algorithm:

For each image I_i in the set of images $\mathbb{I}$, we try to detect a cluster $\mathbb{C}_i$. A potential cluster is initialized with image I_i. Iteratively, images are added that have the highest sum of similarity values to the images that are added already.

First, we create a ranking $\mathbf{r}_i$ of the images in $\mathbb{I}$, based on the similarity values to the initial image I_i. $\mathbf{r}_i = (r_1, \ldots, r_m)$ is a vector containing the indices of the images in $\mathbb{I}$. Starting with the initial image, iteratively the next image in the ranking is found as follows: Of each image remaining to be ranked, similarity values to the images that are already ranked are summed, and the image with the highest sum of similarity values is the next in the ranking. r_1 is set to i and each of the following values $\{r_2, \ldots, r_m\}$ is selected iteratively as

$$r_j = \text{argmax}_{k \notin \{r_1, \ldots, r_{j-1}\}} \sum_{l=1}^{j-1} S(k, r_l) \tag{3.3}$$

The bottom graphs in figure 3.3 show two examples of a set of images and their similarity values positioned in this ranking order.

Based on this ranking we determine if the initial image I_i is part of a cluster $\mathbb{C}_i$ that is a subset of the image set $\mathbb{I}$. This is done as follows: For each position p in ranking $\mathbf{r}_i$, the imageset is split into a potential cluster $\mathbb{C}_p = \{I_{\mathbf{r}_i}, \ldots, I_{\mathbf{r}_{p-1}}\}$ and the set of images outside $\mathbb{C}_p$, $\mathbb{I} - \mathbb{C}_p = \{I_{\mathbf{r}_p}, \ldots, I_{\mathbf{r}_m}\}$.

For each potential cluster $\mathbb{C}_p$ at position p, we determine a measure for how well that cluster separates its images from the rest of the images. For this, we take two sets of similarity values:

- The set of all similarity values between the images in the set of images in cluster $\mathbb{C}_p$,

$$\mathbb{S}_{\text{in}} = S(i,j)_{\forall i \in \{\mathbf{r}_1, \ldots, \mathbf{r}_{p-1}\}, \forall j \in \{\mathbf{r}_1, \ldots, \mathbf{r}_{p-1}\}}$$

- The set of similarity values between the images in the cluster and the images outside the cluster,

$$\mathbb{S}_{\text{out}} = S(i,j)_{\forall i \in \{\mathbf{r}_1, \ldots, \mathbf{r}_{p-1}\}, \forall j \in \{\mathbf{r}_p, \ldots, \mathbf{r}_m\}}$$

Note that the similarity values between the images in the set of images outside the cluster are not used. For the practical reason that we are not interested in clusters that are almost empty or almost the size of the image set, we only use values of p between 5 and $m - 5$.

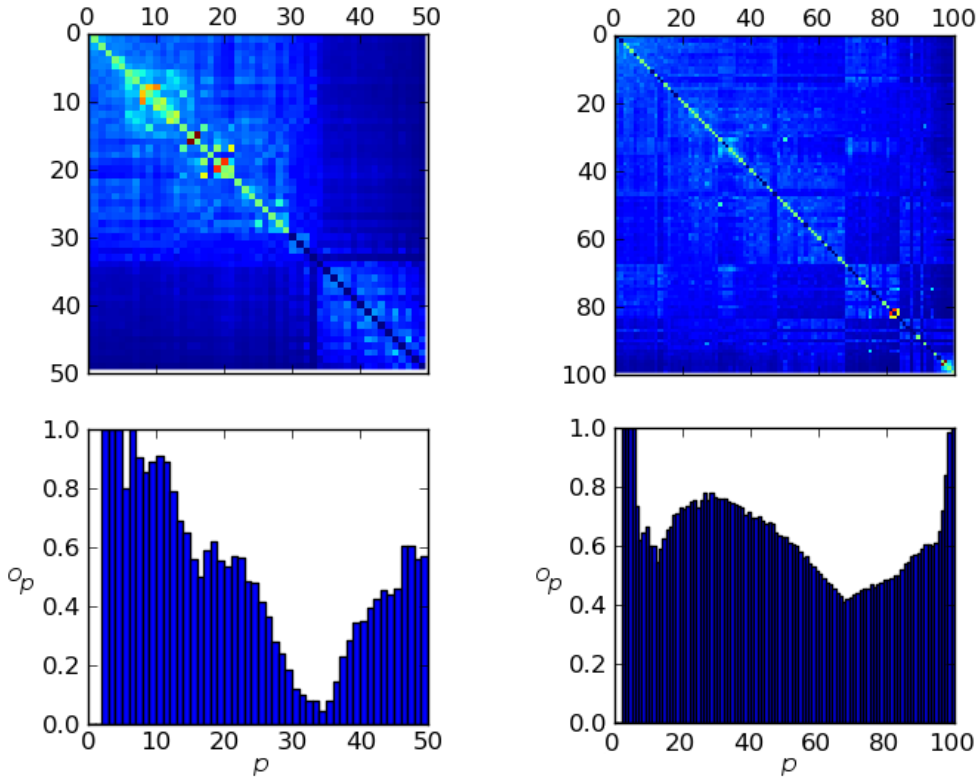
A histogram $\mathbf{H}(\mathbb{S}_{\text{in}} \cup \mathbb{S}_{\text{out}})$ is made from the similarity values in both groups, $\forall s \in \mathbb{S}_{\text{in}} \cup \mathbb{S}_{\text{out}}$. The histogram is created using bins with a fixed bin count and, consequently, a variable width. The fixed bin count we use is 10. The histogram $\mathbf{H}(\mathbb{S}_{\text{in}} \cup \mathbb{S}_{\text{out}})$ consists of a number of bins $(\langle n_{\text{in}}, n_{\text{out}} \rangle_1, \ldots, \langle n_{\text{in}}, n_{\text{out}} \rangle_x)$, where $\langle n_{\text{in}}, n_{\text{out}} \rangle_i$ is the ith bin, n_{in} is the number of similarity values in that bin from $\mathbb{S}_{\text{in}}$, n_{out} the number of similarity values from $\mathbb{S}_{\text{out}}$, and x equals the total number of bins in the histogram. $n_{\text{in}} + n_{\text{out}} = 10$ and x equals the total number of bins in the histogram. Based on this histogram, we define a measure of overlap between the two groups of similarity values. If a bin consists of similarity values of both groups, the overlap in that bin is defined as the number of similarity values of the smallest group in that bin. The overlap o_p of both groups of similarity values, based on the split at position p, in the histogram is defined as the sum

of overlaps in the individual bins, normalized by the size of the smallest set of similarity values. Let n^i_{in} and n^i_{out} denote the elements n_{in} and n_{out} in $\langle n_{\text{in}}, n_{\text{out}} \rangle_i$, the i-th bin in histogram $\mathbf{H}(\mathbb{S}_{\text{in}} \cup \mathbb{S}_{\text{out}})$. o_p is then defined as

$$o_p = \frac{1}{\min(|\mathbb{S}_{\text{in}}|, |\mathbb{S}_{\text{out}}|)} \sum_{i=1}^{x} \min(n^i_{\text{in}}, n^i_{\text{out}}) \tag{3.4}$$

where $|\mathbb{S}_{\text{in}}|$ and $|\mathbb{S}_{\text{out}}|$ denote the number of elements in respectively $\mathbb{S}_{\text{in}}$ and $\mathbb{S}_{\text{out}}$. If the overlap o_p is below a threshold t, then cluster $\mathbb{C}_p$ is considered a good cluster. For t we use a value of 0.2. If there is no o_p that results in a low enough overlap, a cluster $\mathbb{C}_i$ is considered not detected, based on the current initial image I_i. If there are one or more ranking positions p with an o_p below the threshold t, then the cluster $\mathbb{C}_p$ that has the lowest associated overlap o_p is chosen as cluster $\mathbb{C}_i$ for initial image I_i, with associated overlap value o.

Applying this alogrithm to all images in $\mathbb{I}$ results in a set of possibly one or more clusters. If so, the cluster $\mathbb{C}_i$ with the lowest associated value of overlap o_p is taken as the cluster that is detected in this imageset. If for no image I_i a cluster is detected, no cluster is detected in this set of images. See figure 3.3 for an example where a cluster is detected, and one where one is not detected.



(a) (b)

Figure 3.3: Two examples of cluster detection. The bar plot at the bottom row shows the overlap o_p in the histogram between two groups of similarity values given a split at position p. In figure (a) the values for o_p drop below threshold $t = 0.2$, so in this set of images a cluster is detected, whereas in figure (b) the opposite is true.

Chapter 4

Evaluation

In this chapter, the proposed methods are evaluated. Experiments are done on various datasets, to measure the performance of the methods. For similarity matching, we compare pyramid matching [Grauman and Darrell, 2005] against spectral matching combined with link analysis [Kim et al., 2008]. For image sorting, we compare the two sorting values mentioned in chapter three: sorting by the sum of similarity values, and by a density measure. As input we use sets of images that are selected manually, to control the parameters of the set. This way the method can be tested under various circumstances. The end results of experiments, as well as the results of intermediate steps will be shown in this chapter.

4.1 Datasets

We use the ETH-80 imageset [Leibe and Schiele, 2003], and the Caltech 256 dataset [Griffin et al., 2007].

4.1.1 ETH-80 dataset

The ETH-80 set consists of photos of 80 different objects. The objects are from eight categories: cow, car, apple, cup, dog, horse, pear and tomato. Each category consists of ten different objects. Each object is photographed from 41 different viewpoints, evenly distributed over the upper half-sphere (at distances of 22.5-26°). The viewing positions were obtained by subdividing the faces of an octahedron to the third recursion level. For collecting the views, an automated robot setup was used and a blue chromakeying background for easier segmentation. For this project, we use the images of only one object of each category, so the subset we use consists of eight image classes, where each class consists of 41 photos of the same object and each object is from a different category. For a sample of this set, see figure 4.1.

Because within each class, the images are photographs of the same object, the similarity of the images within classes is relatively high. Also all images in this dataset have the same evenly blue background.

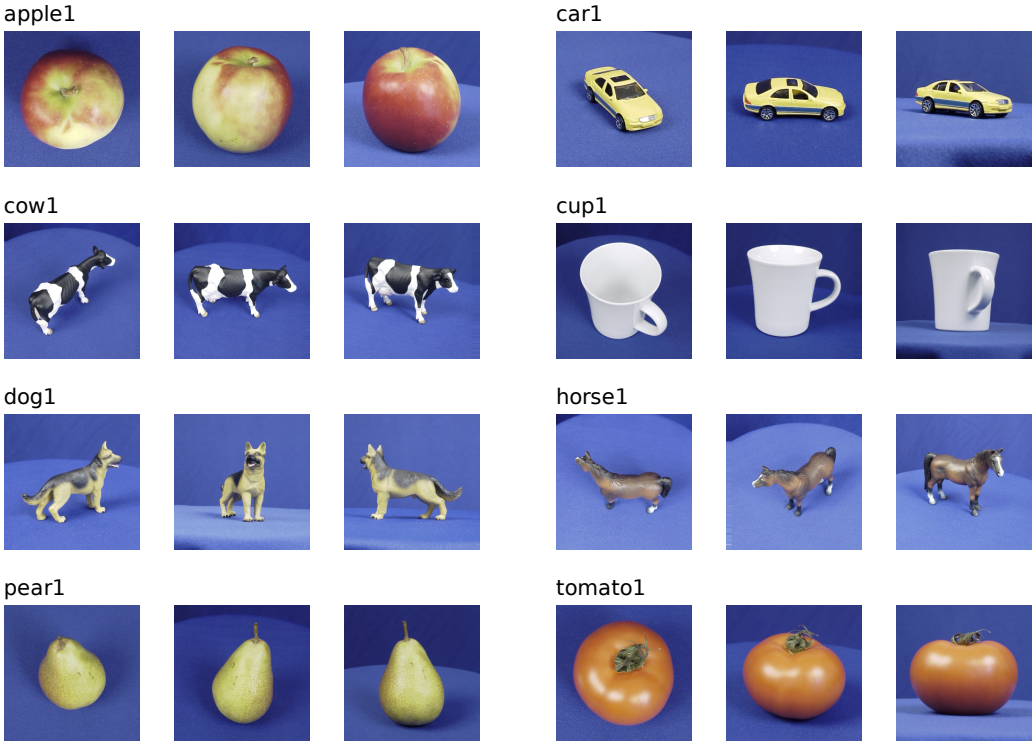


Figure 4.1: Sample of the eight classes ETH-80 dataset

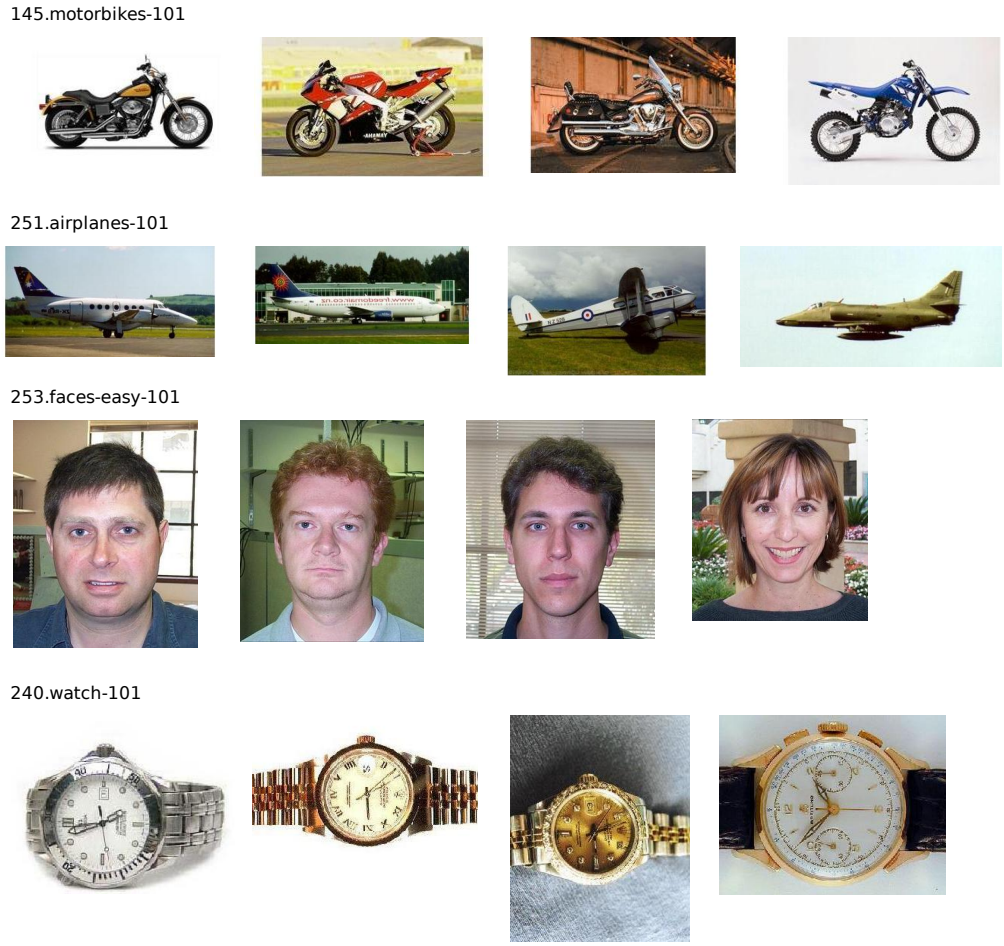


Figure 4.2: Sample of four classes from the Caltech 256 dataset. The photos in the faces category are made by the creators of the dataset. Note that for some categories the background is more cluttered than for others.

4.1.2 Caltech 256 dataset

This dataset contains images of 256 object categories (like watches, airplanes, motorcycles) with over 80 images per category each. It is a follow up of the Caltech 101 dataset. See figure 4.2 for some examples of this dataset. Most of these categories are collected using Google's image search. This is why most images within a class are from different objects, so the variation within each class is higher than in the ETH-80 dataset. Also, because the images are from different sources, the background of the images can vary. Some categories, however, mostly consist of photos of products (specifically made for catalogs and the like). In these cases often there is a clear and even background.

4.2 Implementation

For keypoint detection, we use the implementation by [Zhao, 2011]. A parameter for this implementation is the '**topk**'-option, with which the top number of most salient features can be set. For this project, a value of 100 is used. The actual number of extracted features is often not exactly equal to this value. This is probably because some keypoints have the same value of saliency. The cutoff should be at a point where one keypoint has a higher saliency value than the next, so the implementation has to choose the number of features that is closest to the value of this parameter.

For the pyramid matching, the implementation is available from the site of the authors of the method [Grauman and Darrell, 2005].

Matching using spectral matching and link analysis was implemented in the Python programming language based on the paper in which it is described [Kim et al., 2008]. For finding the first eigenvector of the sparse matrix containing the pairwise geometric relations between features, we use 'Sparsesvd' [Rehurek, 2011]. This is a python wrapper around 'SVDLIBC' [Rohde, 2011], which is a C library for computing the singular value decomposition of large sparse matrices, using the Lanczos algorithm.

4.3 Experiments

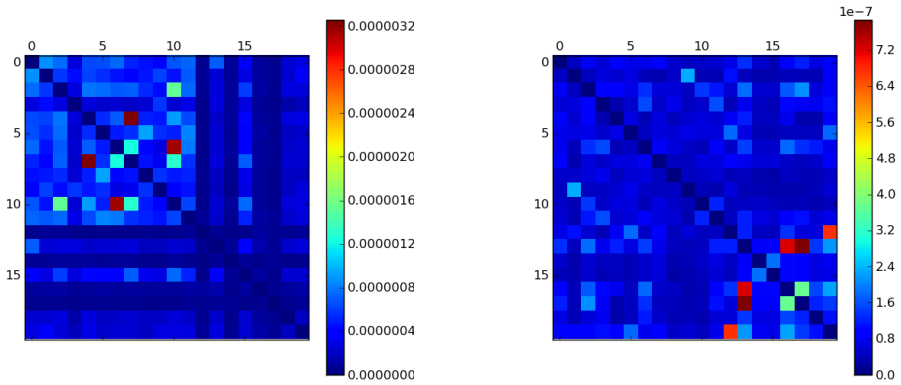
To show how the performance of the method varies, it is run a number of times under a range of different conditions.

4.3.1 Testsets

The method is evaluated using a number of different imagesets as input. These testsets are put together using different parameters so the performance of the method is evaluated under different conditions. Each testset is put together combining a number of images of one category of the dataset with images that are randomly selected from the other categories of the same dataset.

The conditions that are varied are:

- Two different datasets are used. The Caltech256 dataset has a relatively high within-class variation, whereas the Eth80 dataset has a lower within-class variation.
- Different proportions of images from one category are used. Proportions of 20%, 40%, 60% and 80% are used.



(a) (b)

Figure 4.3: Two examples of similarity results

For each combination of the above conditions, eight testsets are created, each based on a different category of the used dataset. Using two datasets, four different proportions, and eight categories, this results in 64 image sets, which are listed in table 4.1. The total number of images in each testset is 50.

4.3.2 Running experiments

First, keypoints are extracted from the images used in the experiments. For each image, the 100 most salient keypoints are taken.

Next, for each of the 64 testsets, pairwise similarity between images is applied with both the method of pyramid matching, and using the combination of spectral matching and link analysis. This results in 128 sets of similarity values. A set of similarity values contains the similarity values between all combinations of image pairs of the 50 images in the testset. See figure 4.3 for an example.

The following step is to apply both variants of the sorting step ('sorting by the sum of similarity values' and 'sorting by density measure'), and the 'cluster detection' step, as indicated by the schematic overview in figure 3.1. These steps are applied to all 128 sets of similarity values. With the sorting by density measure, a value of 15 is used for the value of parameter n denoting the number of closest images used for density estimation.

The output of a sorting operation is a ranking order of the images in the sorted testset. The output of a 'cluster detection', if a cluster if found, is a cluster of images, that is a subset of the image testset. In the next section, the results of the three steps are analyzed, summarized and discussed.

Dataset	Category	Proportion	Dataset	Category	Proportion
Eth-80	apple1	20%	Caltech256	145.motorbikes-101	20%
Eth-80	apple1	40%	Caltech256	145.motorbikes-101	40%
Eth-80	apple1	60%	Caltech256	145.motorbikes-101	60%
Eth-80	apple1	80%	Caltech256	145.motorbikes-101	80%
Eth-80	car1	20%	Caltech256	251.airplanes-101	20%
Eth-80	car1	40%	Caltech256	251.airplanes-101	40%
Eth-80	car1	60%	Caltech256	251.airplanes-101	60%
Eth-80	car1	80%	Caltech256	251.airplanes-101	80%
Eth-80	cow1	20%	Caltech256	253.faces-easy-101	20%
Eth-80	cow1	40%	Caltech256	253.faces-easy-101	40%
Eth-80	cow1	60%	Caltech256	253.faces-easy-101	60%
Eth-80	cow1	80%	Caltech256	253.faces-easy-101	80%
Eth-80	cup1	20%	Caltech256	240.watch-101	20%
Eth-80	cup1	40%	Caltech256	240.watch-101	40%
Eth-80	cup1	60%	Caltech256	240.watch-101	60%
Eth-80	cup1	80%	Caltech256	240.watch-101	80%
Eth-80	dog1	20%	Caltech256	219.theodolite	20%
Eth-80	dog1	40%	Caltech256	219.theodolite	40%
Eth-80	dog1	60%	Caltech256	219.theodolite	60%
Eth-80	dog1	80%	Caltech256	219.theodolite	80%
Eth-80	horse1	20%	Caltech256	182.lawn-mower	20%
Eth-80	horse1	40%	Caltech256	182.lawn-mower	40%
Eth-80	horse1	60%	Caltech256	182.lawn-mower	60%
Eth-80	horse1	80%	Caltech256	182.lawn-mower	80%
Eth-80	pear1	20%	Caltech256	222.tombstone	20%
Eth-80	pear1	40%	Caltech256	222.tombstone	40%
Eth-80	pear1	60%	Caltech256	222.tombstone	60%
Eth-80	pear1	80%	Caltech256	222.tombstone	80%
Eth-80	tomato1	20%	Caltech256	244.wheelbarrow	20%
Eth-80	tomato1	40%	Caltech256	244.wheelbarrow	40%
Eth-80	tomato1	60%	Caltech256	244.wheelbarrow	60%
Eth-80	tomato1	80%	Caltech256	244.wheelbarrow	80%

Table 4.1: Listing of the different testsets used.

4.4 Results

In this section, we will evaluate the output of the experiments, and discuss them per condition.

4.4.1 Sorting of images

The output of a sorting operation is a ranking order in which the images are positioned. The positions in the list are labeled 1 to, and including, n, where n denotes the total number of images in the image set that is used as input. Lower numbered positions in the list denote higher importance, like this is the case with a list of search results given by a search engine. With the ideal sorting, the output is a sequence where all the target images are positioned at the first t positions, where t is the number of target images in the image set, and the random images are positioned in the positions $t+1$ through n, with n being the total number of images in the set. The worst sorting is when the positions 1 through $n-t$ are occupied by random images and the positions $n-t+1$ through n by target images. As baseline we consider a ranking order with randomly assigned ranking positions for the images.

We use two ways of measuring the quality of a sorted sequence:

- **mean position** Take the mean of the positions of the target images. The lower this value, the better the sorting. The minimum value of this measure is the one measured for the ideal sorting, that is $1 + \frac{t-1}{2}$. The maximum value, the one for the worst sorting, is $n - \frac{t-1}{2}$. The baseline value is on average: $1 + \frac{n-1}{2}$

- **number of wrong classifications** This way, it is considered as a classification problem. An image is classified as target image when given a position in the range 1 to and including t, and as random image when given a position in the range $t+1$ to and including n. The number of wrong classifications is taken as a quality measure. For this measure, the minimum value (the one for the ideal sorting), is zero and the maximum value (the one for the worst sorting) is $\min(t, n-t) \times 2$. The baseline value (for a random sorting) is on average $(t - t/n \times t) + t/n \times (n-t)$.

Mean position

Of the 128 sorting results of 'sorting by sum of similarity values' as well as the 128 results of 'sorting by density measure' the mean position of the target images in the ranking order is calculated. This value is then normalized between the minimum and maximum values for each testset as follows:

$$\textit{(normalized mean position)} = \frac{\textit{(mean position)} - \textit{(minimum value)}}{\textit{(maximum value)} - \textit{(minimum value)}}$$

where for each image testset, the minimum and maximum values are calculated as explained. These results are summarized in figures 4.4 to 4.8.

Figures 4.4 to 4.7 show the performances plotted against the proportion of images in the testset that are of the same class. Each figure shows the results of a different combination of similarity matching and sorting variant. The data points of the dark lines are averaged results over all classes of one of the datasets Eth-80 or Caltech256. To visualize the spread of the data, the light lines also show the performances of individual classes.

Figure 4.4 summarizes the results obtained using pyramid matching as similarity value and 'sorting by sum of similarity' as sorting method, figure 4.5 shows the results from pyramid matching in combination with 'sorting by density estimation', figure 4.6 shows the results from using spectral matching and link analysis for the similarity matching and 'sorting by sum of similarity' as sorting method, and figure 4.7 summarizes the results from spectral matching and link analysis in combination with 'sorting by density measure'.

Comparison of datasets
Because the images in the Eth80 dataset have less within-class variance than those of the Caltech 256, we expect testsets with those images to perform better. This appears to be true for experiments where pyramid matching is used, whereas the Caltech 256 testsets show better performance where link analysis is used as similarity measure. Looking at figure 4.8, we see the gain of Eth80 testsets with pyramid matching is larger than the gain of Caltech256 testsets with link analyses.

Comparison of similarity measures
Pyramid matching as a similarity measure shows to give lower mean positions than when spectral matching and link analysis are used as a similarity measure.

Comparison of sorting method
Between the two sorting methods, no conclusive differences are observed in performance, based on these experiments.

Comparison of proportions
With pyramid matching as similarity value, performance increases with higher proportions of target-images. With spectral matching and link analysis used as similarity value performance is better with small proportions of target-images.

Error count

In this section, the number of wrong classifications is used as evaluation measure. For all four combinations of similarity matching and sorting value, perfomance results are shown in figures 4.9 to 4.12. Again the results are normalized between their maximum and minimum values. To see how the performance depends on each class, in figure 4.13 the results are shown averaged over each image category.

Comparison of datasets
Just as with the mean position as evaluation measure, the Eth-80 testsets perform better with pyramid matching as similarity value, and otherwise with spectral matching and link analysis used as similarity measure.

Comparison of similarity measure
Pyramid matching shows overall better performance than the combination of spectral matching and link analysis.

Comparison of sorting method
Also with this performance measure, the results between both sorting methods show no real differ-

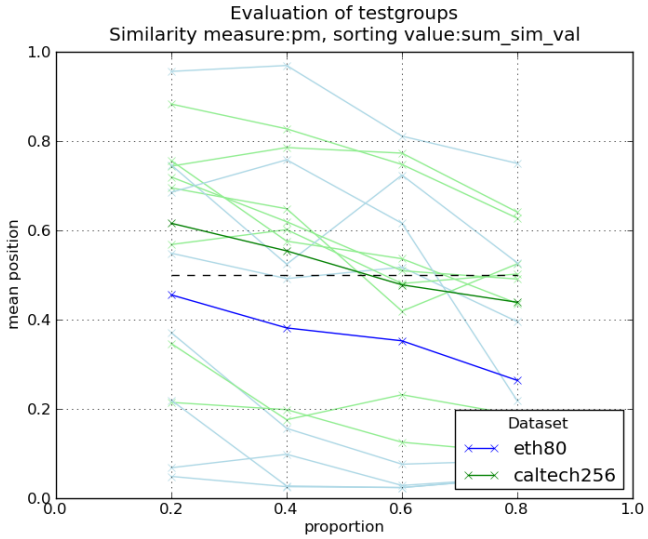


Figure 4.4: Performance measured as *normalized mean position*. Similarity matching is done with pyramid matching, and sorting is done by the sum of similarity values. Baseline performance at the four proportions is indicated by the dotted line.

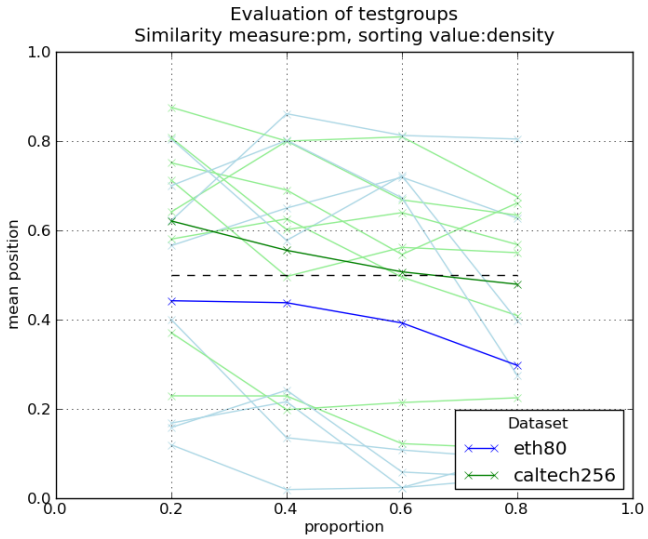


Figure 4.5: Performance measured as *normalized mean position*. Similarity matching is done with pyramid matching, and sorting is done with the density measure. Baseline performance at the four proportions is indicated by the dotted line.

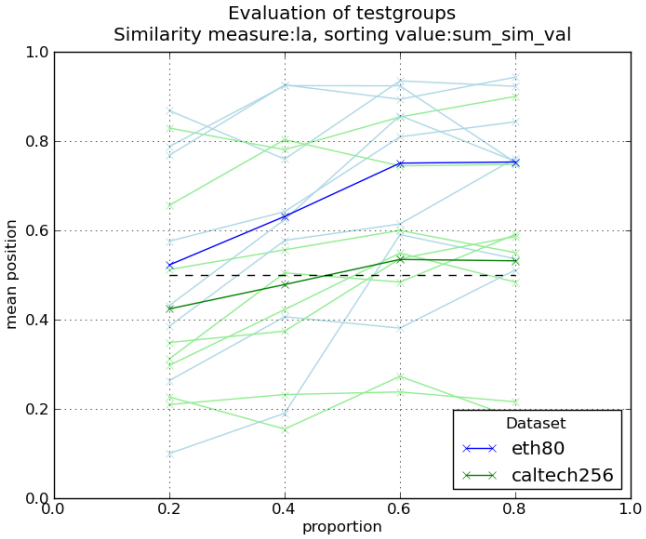


Figure 4.6: Performance measured as *normalized mean position*. Similarity matching is done with spectral matching and link analysis, and sorting is done by the sum of similarity values. Baseline performance at the four proportions is indicated by the dotted line.

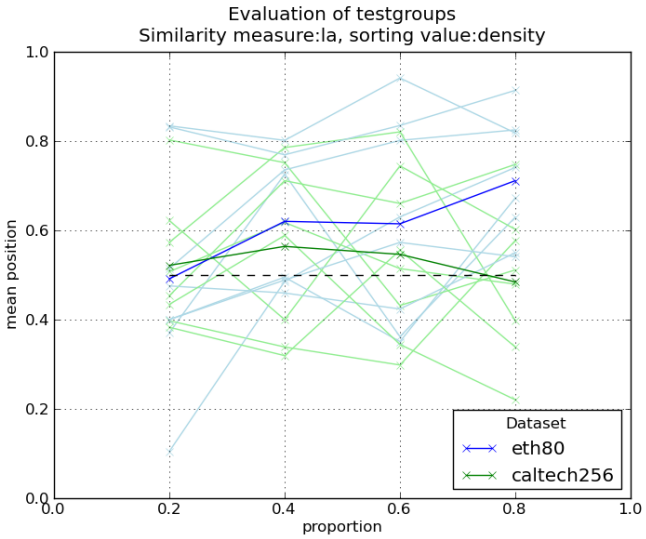


Figure 4.7: Performance measured as *normalized mean position*. Similarity matching is done with spectral matching and link analysis, and sorting is done by density measure. Baseline performance at the four proportions is indicated by the dotted line.

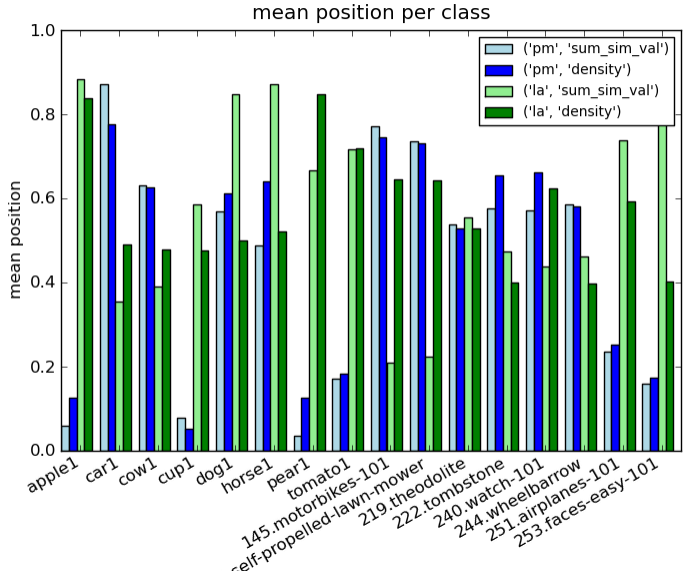


Figure 4.8: Performance measured as *normalized mean position*. Results are averaged over each image class.

ence.

4.4.2 Cluster detection

To all 128 sets of similarity values (both methods of similarity matching for each of the 64 testsets), cluster detection is applied. In some testsets a cluster is detected, in others a cluster is not detected. Of the clusters that are found, we report its size, which should be ideally the same as the number of target images in the testset, and its precision, which is the ratio of target images in the cluster to the total number of images in the cluster. Ideally a cluster should consist of only target images, so this precision should be one. The baseline values for the precision, are equal the proportion of target-images in the testset.

Tables 4.2 to 4.4 show for which image sets a cluster is detected. What immediately shows is that there are no clusters detected in the testsets where the similarity values are produced by spectral matching and link analysis. Clearly, this method of similarity matching is unable to separate the target images from the random images well. Figure 4.14 shows the number of clusters detected by proportion of target-images of the testset. Here it shows that it becomes more difficult to distinguish a cluster of target images from the total set if there are less randomly selected images in a testset. For the quality of the detected clusters, refer to figure 4.15. One of the things this figure shows is that the precisions all lie close to baseline values. Clearly the testsets from dataset ETH-80 show better results than those of the Caltech256 dataset. Figure 4.16 shows the precision of the images in the detected clusters averaged over all testsets of one category. This shows that the performance is still highly varying with the image category.

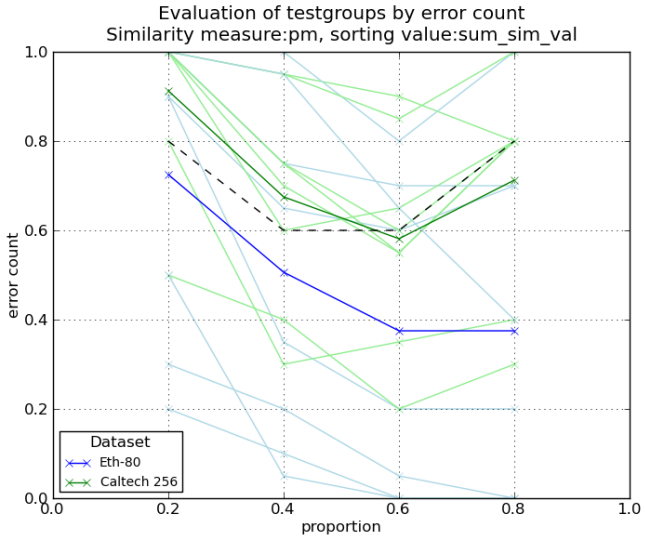


Figure 4.9: Performance measured in *normalized error count*. Similarity matching is done with pyramid matching, and sorting is done by the sum of similarity values. Baseline performance at the four proportions is indicated by the dotted line.

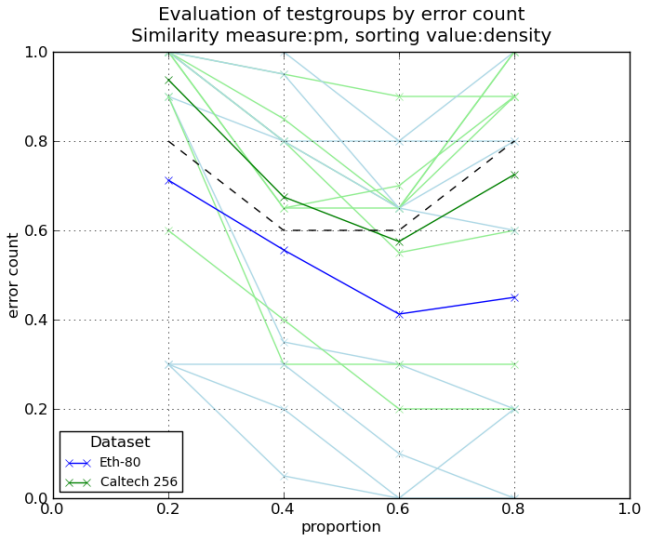


Figure 4.10: Performance measured in *normalized error count*. Similarity matching is done with pyramid matching, and sorting is done with the density measure. Baseline performance at the four proportions is indicated by the dotted line.

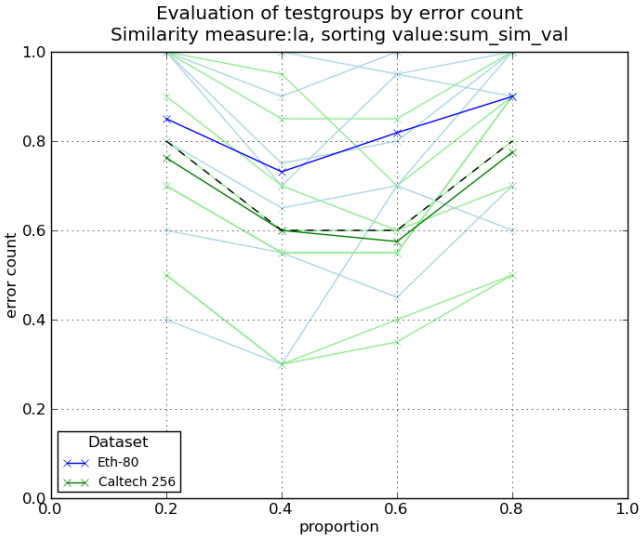


Figure 4.11: Performance measured in *normalized error count*. Similarity matching is done with spectral matching and link analysis, and sorting is done by the sum of similarity values. Baseline performance at the four proportions is indicated by the dotted line.

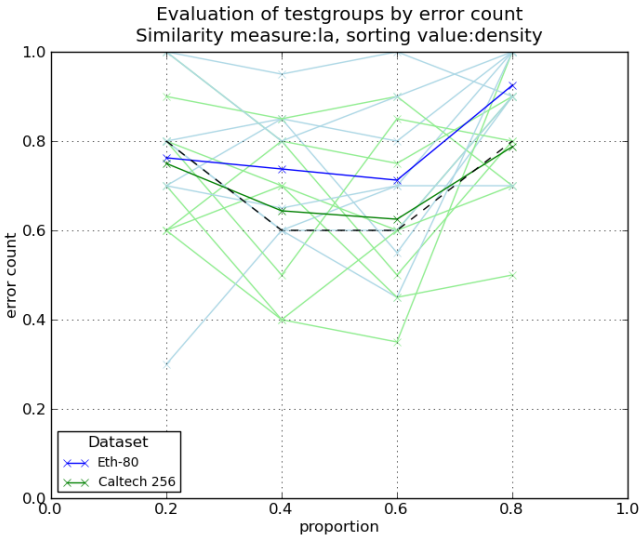


Figure 4.12: Performance measured in *normalized error count*. Similarity matching is done with spectral matching and link analysis, and sorting is done by the density measure. Baseline performance at the four proportions is indicated by the dotted line.

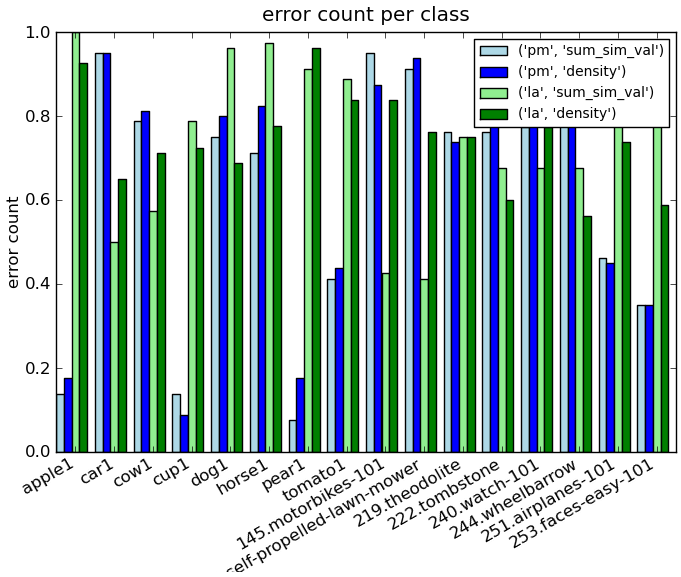


Figure 4.13: Performance measured in *normalized error count*. Results are summarized by image category.

Figure 4.14: Percentage of testgroups where a cluster is detected.

Dataset	Category	Proportion	Sim. method	Cluster detected	Size	Precision
Eth-80	apple1	20%	PM	X	23	0.43
Eth-80	apple1	20%	SM+LA			
Eth-80	car1	20%	PM	X	21	0.00
Eth-80	car1	20%	SM+LA			
Eth-80	cow1	20%	PM	X	16	0.00
Eth-80	cow1	20%	SM+LA			
Eth-80	cup1	20%	PM	X	31	0.32
Eth-80	cup1	20%	SM+LA			
Eth-80	dog1	20%	PM	X	14	0.00
Eth-80	dog1	20%	SM+LA			
Eth-80	horse1	20%	PM	X	15	0.00
Eth-80	horse1	20%	SM+LA			
Eth-80	pear1	20%	PM	X	14	0.71
Eth-80	pear1	20%	SM+LA			
Eth-80	tomato1	20%	PM	X	15	0.40
Eth-80	tomato1	20%	SM+LA			
Eth-80	apple1	40%	PM	X	25	0.80
Eth-80	apple1	40%	SM+LA			
Eth-80	car1	40%	PM	X	29	0.07
Eth-80	car1	40%	SM+LA			
Eth-80	cow1	40%	PM	X	41	0.41
Eth-80	cow1	40%	SM+LA			
Eth-80	cup1	40%	PM	X	13	1.00
Eth-80	cup1	40%	SM+LA			
Eth-80	dog1	40%	PM	X	30	0.43
Eth-80	dog1	40%	SM+LA			
Eth-80	horse1	40%	PM	X	39	0.51
Eth-80	horse1	40%	SM+LA			
Eth-80	pear1	40%	PM	X	28	0.71
Eth-80	pear1	40%	SM+LA			
Eth-80	tomato1	40%	PM	X	28	0.68
Eth-80	tomato1	40%	SM+LA			
Eth-80	apple1	60%	PM	X	31	0.94
Eth-80	apple1	60%	SM+LA			
Eth-80	car1	60%	PM			
Eth-80	car1	60%	SM+LA			
Eth-80	cow1	60%	PM	X	11	0.00
Eth-80	cow1	60%	SM+LA			
Eth-80	cup1	60%	PM	X	9	1.00
Eth-80	cup1	60%	SM+LA			
Eth-80	dog1	60%	PM	X	10	0.00
Eth-80	dog1	60%	SM+LA			
Eth-80	horse1	60%	PM			
Eth-80	horse1	60%	SM+LA			
Eth-80	pear1	60%	PM	X	32	0.94
Eth-80	pear1	60%	SM+LA			
Eth-80	tomato1	60%	PM	X	32	0.88
Eth-80	tomato1	60%	SM+LA			

Table 4.2: Listing of clusters detected by testset, part 1 of 3

Dataset	Category	Proportion	Sim. method	Cluster detected	Size	Precision
Eth-80	apple1	80%	PM	X	39	1.00
Eth-80	apple1	80%	SM+LA			
Eth-80	car1	80%	PM			
Eth-80	car1	80%	SM+LA			
Eth-80	cow1	80%	PM			
Eth-80	cow1	80%	SM+LA			
Eth-80	cup1	80%	PM	X	12	1.00
Eth-80	cup1	80%	SM+LA			
Eth-80	dog1	80%	PM			
Eth-80	dog1	80%	SM+LA			
Eth-80	horse1	80%	PM	X	47	0.85
Eth-80	horse1	80%	SM+LA			
Eth-80	pear1	80%	PM	X	44	0.91
Eth-80	pear1	80%	SM+LA			
Eth-80	tomato1	80%	PM	X	42	0.95
Eth-80	tomato1	80%	SM+LA			
Caltech256	145.motorbikes-101	20%	PM	X	29	0.03
Caltech256	145.motorbikes-101	20%	SM+LA			
Caltech256	251.airplanes-101	20%	PM	X	26	0.35
Caltech256	251.airplanes-101	20%	SM+LA			
Caltech256	253.faces-easy-101	20%	PM	X	40	0.25
Caltech256	253.faces-easy-101	20%	SM+LA			
Caltech256	240.watch-101	20%	PM	X	23	0.13
Caltech256	240.watch-101	20%	SM+LA			
Caltech256	219.theodolite	20%	PM	X	26	0.15
Caltech256	219.theodolite	20%	SM+LA			
Caltech256	182.lawn-mower	20%	PM	X	22	0.05
Caltech256	182.lawn-mower	20%	SM+LA			
Caltech256	222.tombstone	20%	PM	X	31	0.06
Caltech256	222.tombstone	20%	SM+LA			
Caltech256	244.wheelbarrow	20%	PM	X	31	0.13
Caltech256	244.wheelbarrow	20%	SM+LA			
Caltech256	145.motorbikes-101	40%	PM	X	17	0.00
Caltech256	145.motorbikes-101	40%	SM+LA			
Caltech256	251.airplanes-101	40%	PM	X	24	0.75
Caltech256	251.airplanes-101	40%	SM+LA			
Caltech256	253.faces-easy-101	40%	PM	X	23	0.65
Caltech256	253.faces-easy-101	40%	SM+LA			
Caltech256	240.watch-101	40%	PM	X	31	0.29
Caltech256	240.watch-101	40%	SM+LA			
Caltech256	219.theodolite	40%	PM	X	31	0.29
Caltech256	219.theodolite	40%	SM+LA			
Caltech256	182.lawn-mower	40%	PM	X	19	0.05
Caltech256	182.lawn-mower	40%	SM+LA			
Caltech256	222.tombstone	40%	PM	X	32	0.34
Caltech256	222.tombstone	40%	SM+LA			
Caltech256	244.wheelbarrow	40%	PM	X	31	0.35
Caltech256	244.wheelbarrow	40%	SM+LA			

Table 4.3: Listing of clusters detected by testset, part 2 of 3

Dataset	Category	Proportion	Sim. method	Cluster detected	Size	Precision
Caltech256	145.motorbikes-101	60%	PM	X	31	0.45
Caltech256	145.motorbikes-101	60%	SM+LA			
Caltech256	251.airplanes-101	60%	PM	X	42	0.71
Caltech256	251.airplanes-101	60%	SM+LA			
Caltech256	253.faces-easy-101	60%	PM	X	38	0.79
Caltech256	253.faces-easy-101	60%	SM+LA			
Caltech256	240.watch-101	60%	PM	X	36	0.61
Caltech256	240.watch-101	60%	SM+LA			
Caltech256	219.theodolite	60%	PM	X	47	0.64
Caltech256	219.theodolite	60%	SM+LA			
Caltech256	182.lawn-mower	60%	PM	X	29	0.41
Caltech256	182.lawn-mower	60%	SM+LA			
Caltech256	222.tombstone	60%	PM	X	17	0.53
Caltech256	222.tombstone	60%	SM+LA			
Caltech256	244.wheelbarrow	60%	PM			
Caltech256	244.wheelbarrow	60%	SM+LA			
Caltech256	145.motorbikes-101	80%	PM	X	32	0.69
Caltech256	145.motorbikes-101	80%	SM+LA			
Caltech256	251.airplanes-101	80%	PM	X	15	1.00
Caltech256	251.airplanes-101	80%	SM+LA			
Caltech256	253.faces-easy-101	80%	PM	X	43	0.91
Caltech256	253.faces-easy-101	80%	SM+LA			
Caltech256	240.watch-101	80%	PM	X	33	0.76
Caltech256	240.watch-101	80%	SM+LA			
Caltech256	219.theodolite	80%	PM	X	30	0.80
Caltech256	219.theodolite	80%	SM+LA			
Caltech256	182.lawn-mower	80%	PM			
Caltech256	182.lawn-mower	80%	SM+LA			
Caltech256	222.tombstone	80%	PM	X	32	0.78
Caltech256	222.tombstone	80%	SM+LA			
Caltech256	244.wheelbarrow	80%	PM			
Caltech256	244.wheelbarrow	80%	SM+LA			

Table 4.4: Listing of clusters detected by testset, part 3 of 3

Figure 4.15: Precision of images in clusters by target-image proportion and dataset. Baseline performance is indicated by the dotted line.

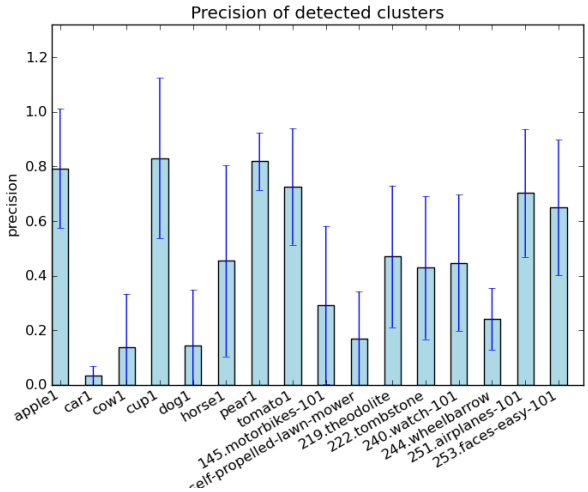


Figure 4.16: Precision of images in clusters by image category.

Chapter 5

Discussion

In this chapter we will discuss the methods and their results. For this project we compared a number of different methods in cascade to improve keyword based image search. The method consists of applying Harris-Affine keypoint detection to extract image keypoints, and SIFT keypoint description to assign keypoint descriptor vectors. These methods are selected based on performance in comparison studies. Based on the keypoint data of the images, similarity measures between image pairs are determined. Two methods for this are applied and tested for performance. Based on the similarity measures between pairs of images, inference about their relevance within the search results is done. This is done by sorting the results according to their similarity values, and by detecting clusters based on the similarity data. Through experiments on a number of different image testsets, put together based on two different image datasets, performance of both sorting and clustering was evaluated.

To answer the question which combination of methods shows the best performance: We found that for pairwise similarity matching of images, pyramid matching outperforms the combination of spectral matching and link analysis. For the mean position of target images of the sorted rankings, pyramid matching shows a clear advantage. For the averaged results in figures 4.4 and 4.5 the mean positions are at or below baseline values. Although for spectral matching and link analysis the values of mean position get better with lower proportions, these values are above or below the baseline value of 0.5. If we examine the distribution of individual performances, the results seem very much dependent on the image category. Some individual image classes perform very well, and are well below baseline values, but most image classes show performances around baseline values or higher. Due to the well performing individual classes, the averaged mean positions are below baseline values. In figures 4.4 and 4.5 it also shows that for the Eth-80 dataset, there are more well-performing image classes than of the caltech256 dataset, and therefore the averaged mean positions of this dataset are clearly lower. The results of spectral matching and link analysis show a more even distribution (figures 4.6 and 4.7). In these cases it shows that the results of the Eth-80 dataset are clearly worse than those of the Caltech256 dataset. Upon analyzing the individual performances, we see that the Caltech256 testsets show performances around the baseline value of 0.5, and that for the Eth-80 dataset there are a some testsets with mean positions higher than baseline, reaching its maximum value, resulting in a higher averaged mean position of target images.

For sorting a set of images based on similarity measures, we found no real difference between the ‘sorting by sum of similarity’ method, and the ‘sorting by density’ method. Again this varies

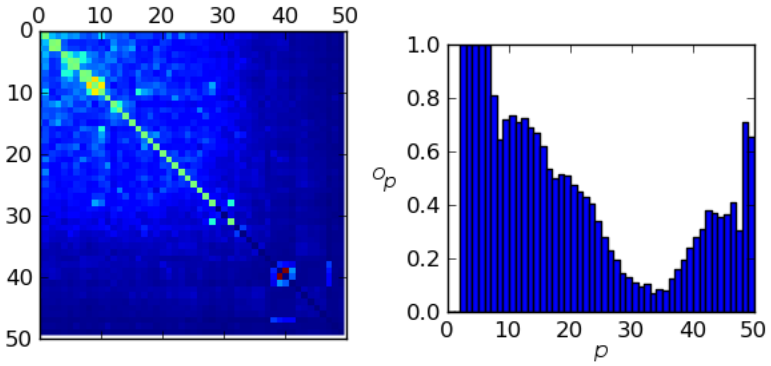


Figure 5.1: Clustering results for class 'apple1' of the Eth-80 dataset. Proportion of target images is 60%. Left, the images are shown in the ranking order for which the cluster is found, and right the values for overlap for each position in the ranking order is shown. Target images are indicated green on the diagonal.

per image class. 'Sorting by density' uses a sorting value based on only local similarity values, and 'sorting by sum of similarity' uses sorting values based on all similarity values related to an image. From this we conclude that using local values is enough, and adding similarity values of less similar images does not add much.

For cluster detection, it holds that if pyramid matching is used as a similarity measure, cluster detection can be applied to detect a subset of similar images within a bigger set of images. We observe that for larger proportions of target images, it becomes more difficult to separate clusters that are subsets of images from the total testset of images. As for the quality of the clusters, the testresults only show precisions above baseline for the clean, uncluttered images of the Eth-80 dataset. See figures 5.1 and 5.2 for two examples from the Eth-80 dataset: 'apple1', with high precision and 'car1', with low precision, as indicated in figure 4.16. For 'apple1' the images of the target-class cluster well. For 'car1' however, the within-class similarity appears to be lower than that of the randomly added images, and so the clustered images are mostly not of the target-class. Figures 5.3 and 5.4 show two examples from the Caltech256 dataset. It is visible that for this dataset, the similarity values are more noisy, and the target-images are more evenly spread over the cluster and outside the cluster. The results are indicative for the kind of data this method might be useful for: Low within-class variance and no background clutter. Photos of products from a webstore could fall in this category.

To answer the question if image content can be used to identify outliers in a set of keyword-based image retrievals: We found that given the proposed methods the results are still close to baseline values, and show improvement in the most artificial testsets, like that of the Eth-80 dataset. For practical situations however, using the proposed methods, we did not find large improvements in image retrieval. We conclude that the proposed methods are not robust enough to handle realistic data.

Whether visual information is useful to improve on keyword based image search still depends highly on the type of image that is used, and different methods should be tried to a certain type

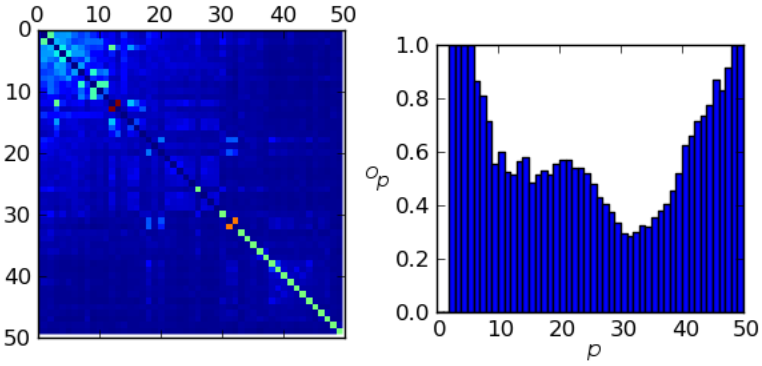


Figure 5.2: Clustering results for class 'car1' of the Eth-80 dataset. Proportion of target images is 40%. Left, the images are shown in the ranking order for which the cluster is found, and right the values for overlap for each position in the ranking order is shown. Target images are indicated green on the diagonal.

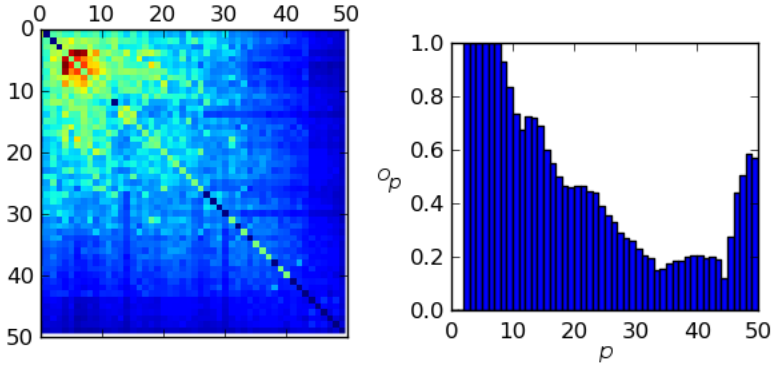


Figure 5.3: Clustering results for class '251.airplanes-101' of the Caltech256 dataset. Proportion of target images is 60%. Left, the images are shown in the ranking order for which the cluster is found, and right the values for overlap for each position in the ranking order is shown. Target images are indicated green on the diagonal.

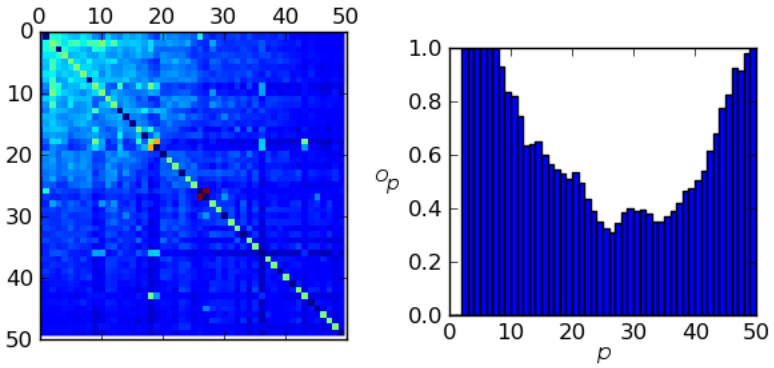


Figure 5.4: Clustering results for class '244.wheelbarrow' of the Caltech256 dataset. Proportion of target images is 60%. Left, the images are shown in the ranking order for which the cluster is found, and right the values for overlap for each position in the ranking order is shown. Target images are indicated green on the diagonal.

of image. The methods proposed in this thesis improve search results in some situations, although there is still a lot of noise in the results. For images with a lot of matching keypoints between them, the method should give good results.

Suggestions for future research

For this project, we selected and tested a limited number of existing methods from the field of computer vision. There is however, a great number of other methods in this field for analysing visual data, so for future research it might be interesting to do experiments with a different selection of methods.

For this project we used artificially put together image sets. This gives control over the parameters of the experiments, but might represent real world situations less accurately. Therefore for future research we suggest using datasets that are based on the results of a keyword-based search engine.

Bibliography

[Blondel et al., 2004] Blondel, V., Gajardo, A., Heymans, M., Senellart, P., and Van Dooren, P. (2004). A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *Siam Review*, 46(4):647–666.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine* 1. *Computer networks and ISDN systems*, 30(1-7):107–117.

[Crowley and Parker, 1984] Crowley, J. and Parker, A. (1984). A representation for shape based on peaks and ridges in the difference of low pass transform. *Transactions on Pattern Analysis and Machine Intelligence*, 6(2):156–170.

[Donoser and Bischof, 2006] Donoser, M. and Bischof, H. (2006). Efficient Maximally Stable Extremal Region (MSER) Tracking. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 553–560. IEEE Computer Society.

[Grauman and Darrell, 2005] Grauman, K. and Darrell, T. (2005). The pyramid match kernel: discriminative classification with sets of image features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 1458–1465. IEEE.

[Grauman and Darrell, 2006] Grauman, K. and Darrell, T. (2006). Unsupervised Learning of Categories from Sets of Partially Matching Image Features. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 19–25. IEEE Computer Society.

[Griffin et al., 2007] Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset.

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK.

[Kim et al., 2008] Kim, G., Faloutsos, C., and Hebert, M. (2008). Unsupervised modeling of object categories using link analysis techniques. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[Leibe and Schiele, 2003] Leibe, B. and Schiele, B. (2003). Analyzing appearance and contour based methods for object categorization. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2. IEEE.

[Leordeanu and Hebert, 2005] Leordeanu, M. and Hebert, M. (2005). A Spectral Technique for Correspondence Problems Using Pairwise Constraints. In *Proceedings of the Tenth IEEE International Conference on Computer Vision-Volume 2*, pages 1482–1489. IEEE Computer Society.

[Lowe, 2004] Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[Mikolajczyk and Schmid, 2002] Mikolajczyk, K. and Schmid, C. (2002). Indexing based on scale invariant interest points. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 525–531. IEEE.

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, pages 1615–1630.

[Mikolajczyk et al., 2005] Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., and Gool, L. (2005). A comparison of affine region detectors. *International journal of computer vision*, 65(1):43–72.

[Rehurek, 2011] Rehurek, R. (2011). Sparsesvd. http://pypi.python.org/pypi/sparsesvd.

[Rohde, 2011] Rohde, D. (2011). Svdlibc. http://tedlab.mit.edu/~dr/SVDLIBC/.

[Schmid et al., 2000] Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of computer vision*, 37(2):151–172.

[Smeulders et al., 2002] Smeulders, A., Worring, M., Santini, S., Gupta, A., and Jain, R. (2002). Content-based image retrieval at the end of the early years. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(12):1349–1380.

[Tuytelaars and Mikolajczyk, 2008] Tuytelaars, T. and Mikolajczyk, K. (2008). A survey on local invariant features. *Foundations and Trends in Computer Graphics and Vision,(3)*, pages 177–280.

[Tuytelaars and Van Gool, 2000] Tuytelaars, T. and Van Gool, L. (2000). Wide baseline stereo matching based on local, affinely invariant regions. In *British Machine Vision Conference*, pages 412–425. Citeseer.

[Zhao, 2011] Zhao, W.-L. (2011). Implementation of keypoint detection. http://www.cs.cityu.edu.hk/~wzhao2/lip-vireo.htm.