# Application of Text Mining on Spatial Visual Sentences

Xiwen Cheng

*July 23, 2015*
Version: 1.2.0

Leiden University


Leiden Institute of Advanced Computer Science
LIACS
Media Lab

Master Thesis

# Application of Text Mining on Spatial Visual Sentences

Xiwen Cheng

*Supervisor* Dr. Michael S. Lew

July 23, 2015

**Xiwen Cheng**
*Application of Text Mining on Spatial Visual Sentences*
Master Thesis, July 23, 2015
Supervisor: Dr. Michael S. Lew

**Leiden University**
*Media Lab*
LIACS
Leiden Institute of Advanced Computer Science
Niels Bohrweg 1
2333 CA Leiden, The Netherlands

# Abstract

Domestic photography has been booming since the introduction of personal devices equipped with cameras like smartphones. As a consequence users struggle in finding relevant pictures in the ever growing photo collections. Content-based image retrieval (CBIR) solves this and takes away burdens like manual tagging. CBIR methods are often inspired by text mining techniques and concepts. In this project the gap of image and text document semantics analogy is closed with the introduction of *Spatial Visual Sentences*. These visual sentences uses the Bag-of-Words (BoW) model to construct semantically meaningful word sequences based on segmentation techniques like Superpixels and Watershed combined with Canny Edge detector or Otsu Thresholding. To illustrate its effectiveness Latent Semantic Indexing (LSI) among other text retrieval techniques are applied to visual sentences against two datasets: UKBench and MIRFLICKR. Due to composition diversity of images from MIRFLICKR the proposed algorithm including BoW had trouble retrieving relevant results. However recall of objects in UKBench with Spatial Visual Sentences is almost as good as BoW. Spatial Visual Sentences essentially extends Visual Phrases with additional semantic properties by creating semantically meaningful word groups. This has proven to be quite effective for object recalls.

Keywords: Image retrieval, CBIR, text-based search algorithms, text retrieval, information retrieval, Latent Semantic Indexing, Latent Semantic Analysis, K-Means, Self Organizing Map, SURF, SIFT, Superpixels, Watershed, Canny, Otsu, Visual Phrases, Visual Sentences

# Acknowledgments

> *A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.*
>
> **— Alan Turing**

After countless hours of studying, discussions, contemplations, self-reflections and source code rewrites my goal of completing this thesis has finally been fulfilled. The success of this journey was possible thanks to my peers. Each one of them contributed in one way or another.

First and foremost I would like to thank my supervisor Dr. Michael S. Lew at LIACS for his guidance. His insights on CBIR and related topics were unmatched. The sessions we had discussing the progress of the thesis helped shape the initial naïve and rough idea into this final thesis. Even though the thesis dragged for a very long time Dr. Michael S. Lew promptly replied to my questions and uncertainties.

I am also very grateful for all the support I received in the past few years from family and friends. In particular J. Cheng, B. Hu, Msc. M. Gheorghe, BSc. D. Meulens, Bsc. C. Isidora and Msc. T. Christina. It was great explaining them what I was working on. The conversations we had were inspiring and fueled my thoughts in addressing and solving issues otherwise I might have missed.

My dearest former colleagues from LIACS namely BSc. A. Peymani, MSc. T. Dorsman, MSc. H. Wortels, MSc. E. Veger, B. Hamar de la Brethonière and BSc. T. Groentjes regularly reminded me I had unfinished business at the university. The periodic alumni meetups kept us connected even though we found our own places in society.

And finally I would like to thank my colleagues at Mendix Technology B.V. namely BSc. J.T. Waleson, BSc. F. Baalbergen B.Eng. J.M. van Kranenburg, B.Eng. P.M. van den Berg and many more for their concerns and understanding. Special thanks to MSc. R.M. Cefala for his valuable feedback; I really appreciate the hours he had spent reviewing this thesis.

Once again, thank you all for your support from the bottom of my heart!

– Xiwen Cheng

# Contents

# 1 Introduction

The amount of personal pictures has been growing rapidly over the past few decades. Flickr.com is a popular image hosting website among the photography community. Figure 1.1 shows since the inception of Flickr how many public photos were uploaded per month [1]. In the year 2014 1.84 million photos were uploaded on average per day. By May 2015 Flickr had over 10 billion [2] images. Not just the younger generation but also elders possess portable devices like smart-phones, tablets and dedicated cameras which enable them to take photographs within a hand reach. As a result users end up with enormous collections of photographs.

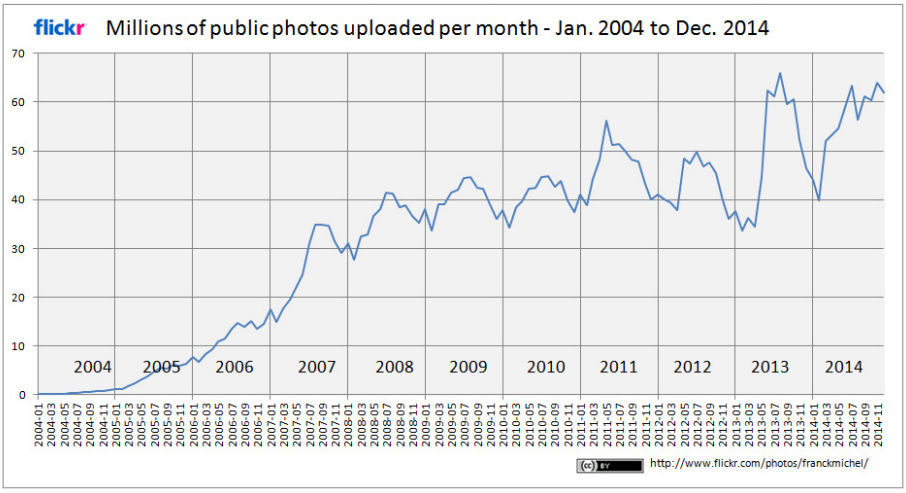



**Fig. 1.1:** Millions of public photos uploaded per month to Flickr between 2004 and 2014

Due to the sheer amount of digital media most users do not bother with manual tagging or cataloguing. There is therefore great demand for systems that can aid users in retrieving relevant images from big image collections with least possible input from the end-user. This research focuses on the former: retrieve relevant images based on a sample input image. Though this area has been researched extensively the Bag-of-Words (BoW) model as basis has been proven to be successful [Yan+07; Mül+10; Zag+11]. Just like BoW most derivatives [Sme+00; Liu+07; Zha+11; Yua+07; Zhe+06; Tho+10; TL12] including our method are inspired by text retrieval concepts. Our main contribution is to complete the semantic analogy between an image and a text document by introducing *spatial visual sentences*.

The spatial visual sentence concept introduced here fits in the analogy (Figure 1.2) between image and text document in semantic granularity drawn by Zheng et al [Zhe+06]. Images

[1] `https://www.flickr.com/photos/franckmichel/6855169886/in/photostream/`
[2] `http://blog.flickr.net/en/2015/05/07/flickr-unified-search/`

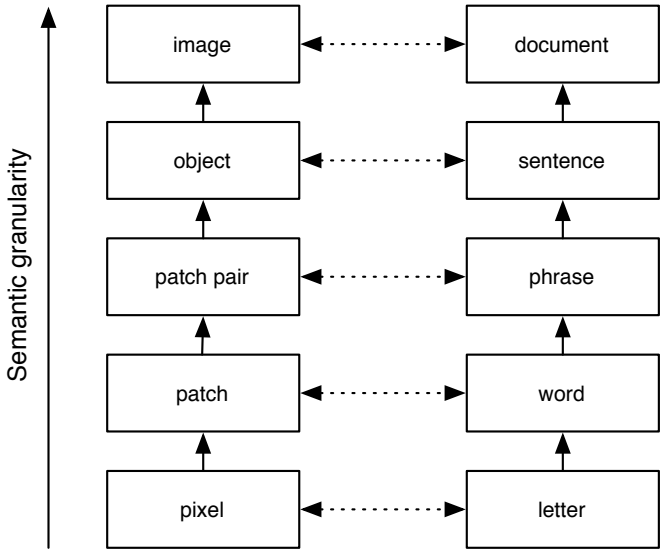



**Fig. 1.2:** Analogy between image and text document in semantic granularity [Zhe+06]

are encoded as a 2-dimensional array of *pixels*. This primitive is analog to *letters* of an alphabet. First level of abstraction is feature descriptor [Des+08] like SIFT and SURF (Section 3.1) are examples of methods which can represent a *patch* (pixels region) as a feature vector. Because such vector is computed based on a collection of pixels adjacent to each other it is similar to composing a *word* using letters. *Phrases* can be constructed by grouping words based on some criteria like visual word co-occurrences [Yua+07] or Euclidean distances. The final semantic gap is closed with *sentence* which in images context corresponds to an *object* or group of words with emergent visual semantics. This grouping method relies on image segments that are inspired by Gestalt principles [Kof13; Mal+01; Koo+11]. We coin this approach as *Spatial Visual Sentences* because it decomposes an image into meaningful group of words with respect to visual semantics using image segmentation techniques like Superpixels and edge detectors.

This thesis is organized as follow:

- Section 2 presents an overview of relevant text mining methods and concepts inspired by *Natural Language Processing*. Some of them are applied during the different phases of the proposed Spatial Visual Sentences algorithm.

- Section 3 outlines state of the art image processing techniques that are also used by our algorithm. Of each technique two or more variations are discussed for comparison in section 5.

- Section 4 dives into the details on how to compute Spatial Visual Sentences from pixels. It also addresses ways to compare visual sentences or images with each other using three similarity measures.

- Section 5 presents how the proposed algorithm performs in terms of recall and precision against two popular datasets and a mixture of algorithm configurations.

- Section 6 summarizes this thesis and draws some conclusions based on the results presented in section 5.

- Finally section 7 sheds some light on possible future work that could improve the proposed algorithm in terms of speed and accuracy.

Full source code including subsets of the datasets are publicly available [3].

[3] https://github.com/xiwenc/cbir-invenio

# 2 Text Mining

Text mining or *knowledge discovery from text* deals with computer assisted text analysis. Hotho et al summarized three possible definitions [Hot+05] of *text mining* in the research areas: Information Retrieval, Natural Language Processing and Information Extraction. In this article we will consider the following definition:

> Text mining is the application of algorithms and methods from the fields machine learning and statistics to texts with the goal of finding useful patterns.

The mining process can be simplified into two main phases:

- Preprocessing: Transform documents into structured data taking into account the syntax and semantics of natural languages where possible.
- Classification: Group documents or parts of it based on a similarity measure to speed up retrieval of similar content.

The rest of this section will describe these phases in more detail. Most text mining approaches apply the *Bag-of-Words* model to represent text documents. It extracts keywords from a document. The importance of a word within a document can be quantified using several known models like *probabilistic model*, *logical model* and *vector space model*.

## 2.1 Text preprocessing

A text document consists of one or more *pages*. Each page is composed of an ordered list of *phrases* separated by a *punctuation mark*. Multi-page documents can be easily transformed to single-page by concatenating all pages. For the sake of simplicity we will only consider single-page documents from now on forward. Each phrase is in turn an ordered list of *words* separated by *white spaces*. A word is an ordered list of terminals also known as characters or in more general: *symbols*.
The set of terminals specific to language is called the *alphabet*. And the set of words that can be composed using the alphabet based on grammar and spelling rules of a language is coined the *vocabulary*. We will use the *English* language as reference to illustrate some concepts in the rest of this section.

### 2.1.1 Stop Words Removal

*Stop words* is a group of words used to filter out before or after processing of natural language data. The idea is to reduce complexity of subsequent computational tasks by reducing the vocabulary size and discarding *non-important* data. Examples in English are *the*, *a*, *an* and *is*. It's important to find a balance in what to discard in order to improve performance because it could severe accuracy of phrase searches. Often the most frequently used words are chosen as they do not contribute to distinctiveness of query results.

### 2.1.2 Synonym, Stemming and Lemmatization

In linguistic there are several ways to group/summarize multiple words into a single term based on their semantics, syntax and context in which it is used. The word *better* is a *synonym* of *good* and vice versa. These two words have no commonalities in their construction at all yet they have similar meaning. Hence synonyms are language specific and is often presented as a static lookup table.

Words with similar composition like *cats* and *catty* can be mapped to the root *cat*. This process for reducing inflected or sometimes derived words to their stem word, base form or root is called *stemming*. Most algorithms that determine the stem word of a given target word uses lookup table that maps to inflicted forms or more flexible rules like *suffix-stripping algorithm*.

A more complex form of determining the stem of a word is *lemmatization*. The *part of speech* is first detected using the context in which the word occurs. Next the stemming rules are applied depending on the word category. For instance the word *meeting* can be a base form of a noun or the verb *to meet* depending on the context:

- *During last meeting*
- *We are meeting again tomorrow*

Additional linguistic preprocessing specifics to semantics add limited value to Bag-of-Words because co-occurrence of terms serve as an automatic disambiguation for classification purposes[LK02]. Nonetheless progress[Hot+03] is being made to exploit these concepts:

- **Part-of-Speech tagging**: classify the category of the word based on the context it is used in e.g. noun, verb, adjective, etc.
- **Text chunking**: create groups of adjacent words in a sentence e.g. *blue balloon*.
- **Word Sense Disambiguation**: Resolve ambiguity using the context in which a word is used. For instance *bank* could mean *financial institution* or *border of a lake or river*.

### 2.1.3 Keyword Selection

The words which make up the vocabulary can further be reduced by applying selection algorithms. One way of doing this is to select keywords based on their *entropy*[LS89]. The entropy of a word $t$ is defined as:

$$W(t) = 1 + \frac{1}{\log_2|D|} \sum_{d \in D} P(d,t) \log_2 P(d,t) \quad \text{with} \quad P(d,t) = \frac{\text{tf}(d,t)}{\sum_{l=1}^{n} \text{tf}(d_l,t)} \tag{2.1}$$

Where:

| | | |
|---|---|---|
| $D$ | = | the set of documents |
| $T$ | = | the set of all different terms $\{t_1, ..., t_m\}$ occuring in $D$ |
| $\text{tf}(d,t)$ | = | frequency of term $t$ in document $d$ |

Words that occur in many documents get low entropy value. In an ideal situation all the values are computed and sorted. The top $n$ can then be selected to be the index terms. Greedy approach[Hot+05] can also be applied to reduce required computational resources for large document collections.

### 2.1.4 Vector Space Model

Given a computed set of index terms the original document collection can be transformed to a more efficient data structure suitable for searching. The vector space model can represent documents as numerical feature vector with $m$-dimensions: $w(d) = (x(d,t_1), ..., x(d,t_m))$. Where $m$ is the number of distinct terms and $x(d,t)$ is the weighting function. The simplest form to encode the vector is using *booleans*:

$$x_b(d,t) = \begin{cases} true & \text{if } t \text{ occurs in } d \\ false & \text{if } t \text{ does not occur in } d \end{cases} \tag{2.2}$$

Salton et al[Sal+94] proposed a weighting scheme $x_s(d,t)$ which combines term frequency $tf(d,t)$, inverse document frequency $idf(t)$ and length normalization factor. By normalizing the weights against the document length each document has equal chance of being retrieved.

$$x_s(d,t) = \frac{\text{tf}(d,t) \times \text{idf(t)}}{\sqrt{\sum_{j=1}^{m} \text{tf}(d,t_j)^2 (\log(N/n_{t_j}))^2}} \tag{2.3}$$

Where:

| | | |
|---|---|---|
| $\log(N/n_t)$ | = | inverse document frequency idf(t) |
| $N$ | = | size of document collection D |
| $n_t$ | = | number of documents in D containing term t |

The similarity between two documents $d_1$ and $d_2$ can be computed using Euclidean distance:

$$S(d_1, d_2) = \sqrt{\sum_{k=1}^{m} |x(d_1, t_k) - x(d_2, t_k)|^2} \tag{2.4}$$

As the name already suggests: large distance value means less similar. While identical documents would yield a distance of $0$. This function can be used to retrieve similar documents based on a query encoded as another document $d_q$. Thus a search action equals to computing $S(d, d_q)$ for all $d \in D$.

## 2.2 Clustering

In data mining there are two main categories of structuring data: by means of classification and clustering. The former method requires a predefined set of classes and optionally example data that belongs to each class. Sebastiani [Seb02] discussed a plethora of approaches using machine learning. Examples are Naïve Bayes Classifier, Nearest Neighbor Classifier, Decision Trees and Support Vector Machines. Clustering algorithms tackles the issue from the opposite direction: partition the input data into $k$ clusters based on similarity measures. Given the set of feature vectors described in section 2.1.4 it is possible to automatically cluster the data. The rest of this section outlines two well known and effective clustering algorithms namely *k-means* and *Self Organizing Map*.

### 2.2.1 k-means

In the field of statistics and data mining, k-means is the most popular clustering method [VS10; Ste+00; SS12; KM13]. There are several variations [Pha+05; Sin+11] employing heuristics in order to improve the original computational complexity of $NP$-hard in finding the global optimum. The standard k-means algorithm starts by randomly initialize the centroids using the input dataset to be clustered. Next it iteratively assigns elements to partitions based on nearest centroid and recomputing the centroids using partition centres (Algorithm 1). The loop ends when the centroids don't change or after a fixed number of iterations.

**Algorithm 1** Standard k-means

Input: Set $D$, distance measure $dist$, number $k$ clusters
Output: Partition $P$ of $k$ disjoint subsets of $D$ such that $\cup_{p \in P} = D$ and $\cap_{p \in P} = \emptyset$

1: Initialize by randomly selecting $k$ elements from $D$ as starting centroids $C = \{c_1 ... c_k\}$
2: **repeat**
3: Assign $d \in D$ to $p_i$ if closest to centroid $c_i$ with respect to $dist$
4: Recalculate centroids $C$
5: **until** The centroids $C$ are stable
6: **return** P = $\{p_1, ..., p_k\}$

### 2.2.2 Self Organizing Map

*Self Organizing Map* (SOM), also known as Kohonen map [Koh82] is a special architecture of neural networks that clusters high-dimensional data vectors. It performs well for small datasets and relatively small number of clusters in comparison to k-means [Abb+08]. The clusters are arranged in a low-dimensional topology such that clusters nearby each other are more similar to each other than those further away. Unsupervised training of the network is possible because the learning process is aided by a similarity measure. The network structure has two layers (Figure 2.1): output and input layers.

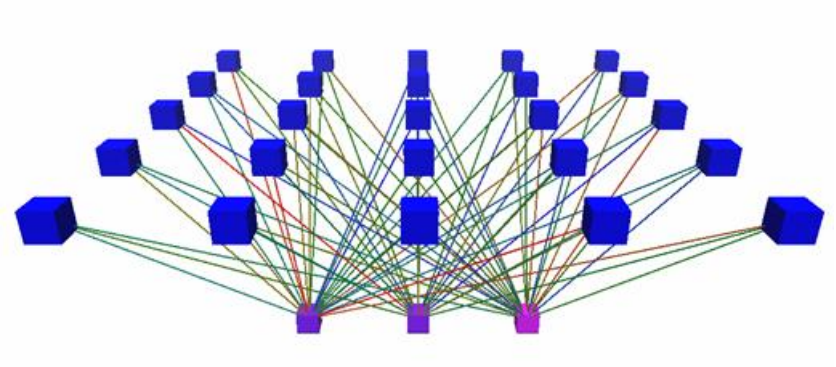


**Fig. 2.1:** Kohonen network structure: output layer (top) and input layer (bottom) [Unk15]

Neurons in the input layer correspond to the input dimensions $n$ of a document $d$. This input is presented as $\vec{I} = [t_1, ..., t_n]$ where $t_i$ is a term in $d$. The output layer consists of $k$ nodes each corresponding to a cluster center. All neurons in the input layer are connected to all nodes in the output layer. Thus each node $W$ contains $n$ weights $\vec{W} = [w_1, ..., w_n]$. Algorithm 2 depicts how the learning process works.

SOM starts by randomly assigning small initial values to each $\vec{W}_i \in W$ (Line 1). For every iteration a random input vector $\vec{I}$ selected from the training set $D$ (Line 6). The Best Matching Unit (BMU) $\gamma$ is determined by computing all the Euclidean distances between $\vec{I}$ and each $\vec{W}_i$ then select the node $W_i$ with the lowest distance (Lines 7-10). Next a radius $\sigma$ is computed based on the iteration number $y$ and a time constant $\lambda$ which starts with $\sigma_0$ and converges to $1$ over time (Line 11). Lines 12-16 updates neighbors of BMU $\gamma$ within radius $\sigma$ based on influence rate $\Theta$ and learning rate $L$ which decays exponentially. The algorithm finishes the training when the given number of iterations $m$ has been reached.

A trained Kohonen network classifies a new input vector $\vec{I'}$ the same way BMU is computed.

## 2.3 Latent Semantic Indexing

*Latent Semantic Indexing* (LSI) is an indexing and retrieval method that uses *singular value decomposition* to identify patterns in term relationships and concepts in unstructured text. It is based on the principle that words used in the same contexts tend to have similar meanings

**Algorithm 2** Self Organizing Map: Learning process

Input: $W$ nodes, training set $D$, max iterations $m$, map radius $\sigma_0$, learning rate $L_0$
Output: trained nodes $W$

Randomly initialize each $\vec{W}_i$
Iteration counter $y = 0$
Time constant $\lambda = m/log(\sigma_0)$
**repeat**
    Increase $y = y + 1$
    input vector $\vec{I} =$ select_random$(D)$
    **for all** Nodes $W_i$ **do**
        $\delta_i = S(\vec{I}, \vec{W}_i)$ (See equation 2.4)
    **end for**
    Best Matching Unit $\gamma = W_i$ with lowest $\delta_i$
    Neighborhood radius $\sigma = \sigma_0 e^{-y/\lambda}$
    **for all** Nodes $W_i$ within radius $\sigma$ **do**
        Influence rate $\Theta = e^{-S(\vec{\gamma},\vec{W}_i)/2\sigma^2}$
        Learning rate $L = L_0 e^{-y/\lambda}$
        Update $\vec{W}_i = \vec{W}_i + \Theta L(\vec{I} - \vec{W}_i)$
    **end for**
**until** $y >= m$
**return** $W$

(semantics). A good indication of the effectiveness of LSI is Google [Nam14; LL94; Dee+89; Beh+03] relying on it for information retrieval. Because of its mathematical approach it is independent of language and is therefore interesting to apply to images. For details and analysis of LSI refer to other publications [WH+04; Pap+98].

# 3 Content-Based Image Retrieval

*Content-Based Image Retrieval* (CBIR) is the application of computer vision techniques to image retrieval problems. Unlike *text-based image retrieval* which exploits the availability of textual information, CBIR purely relies on the ability to make sense of the actual content. Visual perception of the human brain is extremely sophisticated which enables it to identify visuals with semantics like recognizing written text or *kids playing with a beach ball*. The saying "*A picture is worth a thousand words*" clearly highlights the short coming of annotated images. Perceptions of one person may not be the same as someone else's. Also nowadays the sheer amount of personal digital imaging makes it an impossible task to properly annotate all the data. Hence there is a great demand for good CBIR systems.

A basic CBIR system (Figure 3.1) composes of two phases. First an image collection is preprocessed and transformed into a data structure suitable for searching. Then a user can present to the system a query image in which it will return a list of similar images using this data structure.

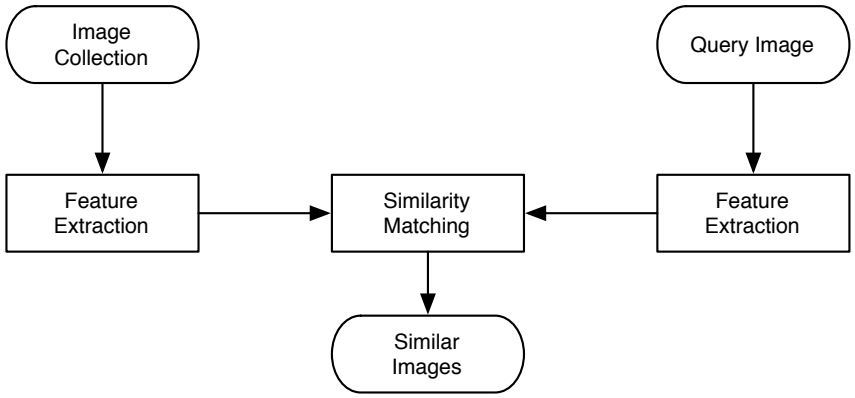



**Fig. 3.1:** Generic Content-Based Image Retrieval system

In this section we will dive into methods used during the preprocessing phase and possible data structure representation. The concepts and methods described were selected based on their relevance to be building blocks of the algorithm outlined in section 4. Please consult surveys on CBIR [TM08; Tuy+10; Liu+07; Lew+06] for more complete overview of existing research on this topic.

## 3.1 Image Features and Detectors

Images are coloured pixels arranged in a two-dimensional plane. Examining these pixels individually to make "sense" of them is extremely computationally expensive. Therefore raw image information is often transformed into an abstract and compressed representation known as *image features*. Image feature defined as interesting part of an image is the basis of many computer vision algorithms. The performance of these algorithms greatly depend on how good the features are. Feature effectiveness can be expressed in terms of the following properties[TM08]:

- *Repeatability*: High probability of identifying similar parts between two images taken of the same scene with different viewing conditions like angle and illumination.
- *Distinctiveness/informativeness*: Has to be descriptive enough to distinguish detected features from each other.
- *Locality*: Describes an image patch taking into account the size to mitigate potential occlusion problems.
- *Quantity*: Number of detected features should be sufficiently large to enable object detection.
- *Accuracy*: The same feature must be detected during location or scale change.
- *Efficiency*: Feature must be detected and computed within reasonable time crucial for real-time applications.

The most basic image features are based on color, texture, shape and location. *Global features* try to describe the image using a single vector [Zag+11] while *local features* computed per interest point in an image are capable of recognizing objects. Based on comprehensive surveys on CBIR [Lew+06; Liu+07; Dat+05] we will focus only on more state-of-the-art local feature descriptors which are affine invariant. This allows matching of feature vectors under different transformations like scale and rotations.

Prior to extracting descriptors the area or point of interest (also known as salient points) must be identified using a *detector*. There are four main categories of detectors: *Edges*, *interest points*, *regions of interest* and *ridges*. Canny[Can86] and Sobel[Sob90] are well-known examples of edge detector operators which find sets of points with strong gradient magnitude in the image. Blob detectors[For07; Den+07] focus on regions of interest in images which might be too smooth for corner detectors to find points of interest[Ros+08; RD06].

Local image patches extracted from the detected features are known as *feature vector* or *feature descriptor*. SIFT[Low99; Low04] and SURF[Bay+08] are known to perform well in terms of repeatability[Gil+10; JG09; Rub+11; MY09; Tuy+10]. These two has been

proven to be quite effective and therefore referenced frequently [Cal+10; Rub+11; Leu+11; Ala+12; Lou+00]. Hence the rest of this section is devoted to give a quick overview on how these two methods work. Please refer to the original publications [Low04; Bay+08] for full details. Juan et al [JG09] concluded SIFT is slow and doesn't perform well with illumination changes while it is invariant to rotation, scale changes and affine transformations. SURF is fast and performs as good as SIFT but is not stable to rotation and illumination changes.

### 3.1.1 Scale-Invariant Feature Transform

This section briefly outlines the *Scale-Invariant Feature Transform* (SIFT) algorithm introduced by Lowe[Low04]. It starts by identifying candidate locations and its corresponding scale that are invariant to scale change. The detection uses *scale-space extrema* in the *Difference-of-Gaussians* (DoG) function convolved with the image. Each sampled point is compared to its eight neighbors in the current scale and nine neighbors of both scale-up and scale-down (Figure 3.2). It is considered to be an candidate if it is a local extrema.

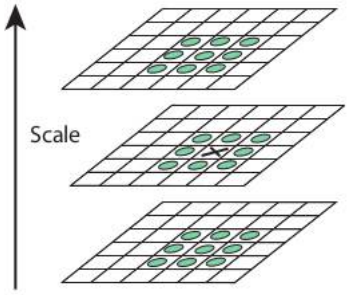



**Fig. 3.2:** Finding local extrema by comparing neighboring pixels across scales [Pro15a]

The set of found keypoints are then refined by rejecting low contrast extrema and edge keypoints because the DoG function has strong responses along edges.

A region around a keypoint is chosen based on the scale found in previous steps. Of this region an orientation histogram with 36 bins is created weighting the gradient magnitude and Gaussian-weighted circular window. Top 80% highest peaks of the histogram are selected yielding multiple keypoints of the same location and scale but different directions.

A keypoint descriptor is represented by a $4 \times 4 \times 8 = 128$ dimensional feature vector. The $16 \times 16$ array around the keypoint is divided into 16 sub-regions of $4 \times 4$. For each sub-region an 8 bin orientation histogram is created (Figure 3.3).

### 3.1.2 Speeded Up Robust Features

Bay et al [Bay+08] presented a high performance scale- and rotation-invariant interest point detector and descriptor named *Speeded Up Robust Features* (SURF). SURF approximates Difference of Gaussian with *Box filter*. Figure 3.4 shows an example of the approximation. Convolution with box filter can be calculated with integral images. Therefore it is possible to

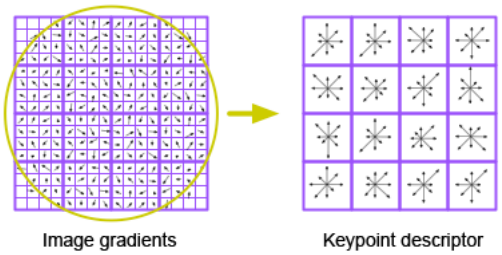



**Fig. 3.3:** 128 dimensional feature vector computed from image gradients [Ban15]

parallelize the computation for different scales. Furthermore SURF relies on determinant of Hessian matrix for both scale and location.

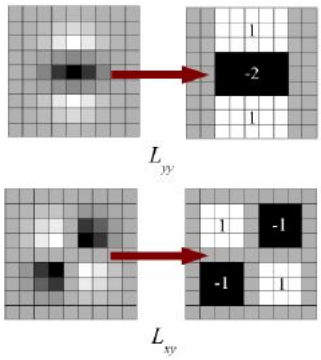



**Fig. 3.4:** Difference of Gaussian approximation with Box Filter [Pro15b]

Orientation assignment is computed using wavelet responses in horizontal and vertical directions for a neighborhood of size $6s$ where $s$ is the *scale*. Next the wavelet responses are weighted with a Gaussian then plotted as illustrated in figure 3.5. The dominant orientation is estimated by the sum of all responses within a window of angle 60 degrees. The longest vector over all windows defines the orientation of the interest point.

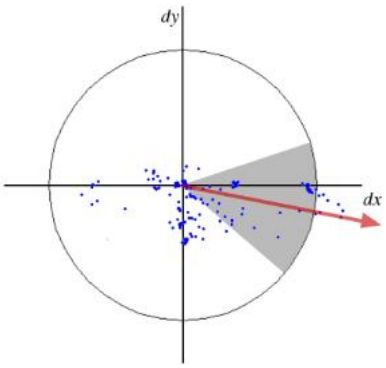



**Fig. 3.5:** Sliding orientation window to detect dominant orientation [Pro15b]

The SURF descriptor describes an interest area of $20s$ around the detected keypoint. This area is divided into $4 \times 4$ subregions. For each subregion a vector $v = (\sum dx, \sum dy, \sum |dx|, \sum |dy|)$ is calculated based on $5 \times 5$ samples. Where $dx$ and $dy$ are wavelet responses in horizontal

and vertical directions respectively. And $|dx|$ and $|dy|$ are the absolute values of the responses. By concatenating $v$ of all $4 \times 4$ a single descriptor vector is represented with 64 dimensions.

## 3.2 Image Segmentation

Image segmentation is the process of partitioning an image into multiple segments. Segments are groups of pixels which *belong* together based on some characteristics like colors, textures, position, intensities, etc. Ideally each segment represents a real-world object. There are numerous surveys [Yeo+05; Pen+13; LM01; Fre+02; YM12; Zha+08] on image segmentation that give very thorough overview of existing methods. There are two main type of methods: boundary-based use the discontinuity property of pixels in relation to its neighbors while region-based apply the similarity property of nearby pixels. Recent proposals [YM12] in improving segmentation performance suggest in combining both methods for more accurate segments.

In the rest of this section the methods *SLIC Superpixels* and *Watershed Transform* are discussed. Watershed requires the aid of markers to compute the segments. These methods compared by Al-Kubati et al [AK+12] are *Canny edge detector* and *Otsu Threshold*. To conclude the section the mentioned methods are applied on two sample images. The code used is based on implementations of segmentation methods in *OpenCV* and *Scikit-image*.

Other good segmentation alternatives are Graph Cut [YM12; Del+12] and Felzenszwalb's algorithm [FH04] based on pairwise region comparison. Due to the scope of this project we will not discuss all of them.

### 3.2.1 SLIC Superpixels

Achanta et al [Ach+10] introduced *Simple Linear Iterative Clustering* (SLIC) which can generate compact nearly uniform superpixels. Superpixel is a concept originally developed by Xiaofeng Ren and Jitendra Malik [RM03]. Each superpixel is a segment obtained from low-level grouping process.

According to Ren et al [Ren] a superpixel map has many desired properties:

- Computationally efficient: reduces complexity of images by using the pixel groups instead of individual pixels.
- Representationally efficient: a single pixel has at most 8 adjacent neighbors while superpixels can have many more which is more efficient to represent in models as $n$ relations.

- Perceptually meaningful: all pixels in a superpixel most likely has some uniform properties in common like color and texture.

The SLIC algorithm is summarized in algorithm 3. It is somewhat similar to *k-means* with regard to the body-construction of the algorithm. It starts by initializing $k$ cluster centers then iteratively assign pixels to the best matching centroid and recompute cluster centers until some acceptable residue error $E$ is reached. The distance measure takes into account the color and pixel position. For full details refer to the original publication [Ach+10].

**Algorithm 3** SLIC Superpixels segmentation [Ach+10]

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps $S$.
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
3: **repeat**
4: **for all** cluster center $C_k$ **do**
5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure.
6: **end for**
7: Compute new cluster centers and residual error $E$ {$L1$ distance between previous centers and recomputed centers}
8: **until** $E \leq$ threshold
9: Enforce connectivity.

### 3.2.2 Watershed Transform

The watershed transform method [BL79] belongs to the region-based class. It was inspired by geography. Consider gray-scale image to be a landscape which is flooded with water. At points where different water areas meet each other a dam is built forming watershed lines. In order for the algorithm to perform well the *marker*-based variation was introduced. Markers essentially guide the algorithm in finding the segments otherwise it would over-segment. Listing 1 shows how watershed can be used in *OpenCV*. The function prototype is $watershed(image, grayed, edges, min_ratio, max\_count)$:

1. $image$ is the original input image.

2. $grayed$ is the converted original input input to gray scale carrying only intensity information.

3. $edges$ is an image of the same dimension as input image of which is used as basis for the markers

4. $min\_ratio$ is the minimal ratio of a contour in $edges$ which must be exceeded by $\frac{contour\_area}{total\_image\_area}$. A value less or equal to 0 yields all detected segments regardless of the contour area.

5. $max\_count$ denotes the maximum number of segments to return from $image$ giving bigger segments more priority.

Sections 3.2.3 and 3.2.4 show how $edges$ can be constructed. A markers mask is constructed by tracing the contours in $edges$ and drawing them on a black backgrounds. Finally the watershed algorithm is applied to the original image together with markers.

Roerdink et al [RM00] looked into speeding up watershed transform by means of parallelization. They concluded the speedup to be achieved is pretty modest because of a global operation being the bottleneck.

### 3.2.3 Canny Edge Detector

The Canny edge detector operator uses multiple stages to detect a wide range of edges in images. We will not dive into the details of the algorithm but only highlights the parameters. Interested readers are advised to read the original publication [Can86] instead.

The function prototype of Canny edges in listing 1 is $canny(image, gaussian\_ksize, threshold1, threshold2)$. It accepts four parameters:

1. $image$ is the input image.
2. $guassian\_ksize$ is the filter size of the Gaussian kernel.
3. $threshold1$ is a threshold of hysteresis.
4. $threshold2$ is also a threshold of hysteresis.

Hysteresis is the thresholder used in Canny which accepts an upper and lower threshold limit while most thresholders use a single value limit. Setting the thresholds too low will miss details and too high will miss important information.

### 3.2.4 Otsu Threshold

Thresholding itself can be considered to be the simplest form of image segmentation. For every pixel in the input image if the intensity is lower than a given threshold $\tau$ it converts the pixel to black otherwise white. Otsu's [Ots75; Gre10] method finds a threshold $\tau$ where the sum of the foreground and background spreads is at its minimum.

The code in listing 1 also function prototype $otsu(image, gaussian\_ksize)$. It accepts two arguments:

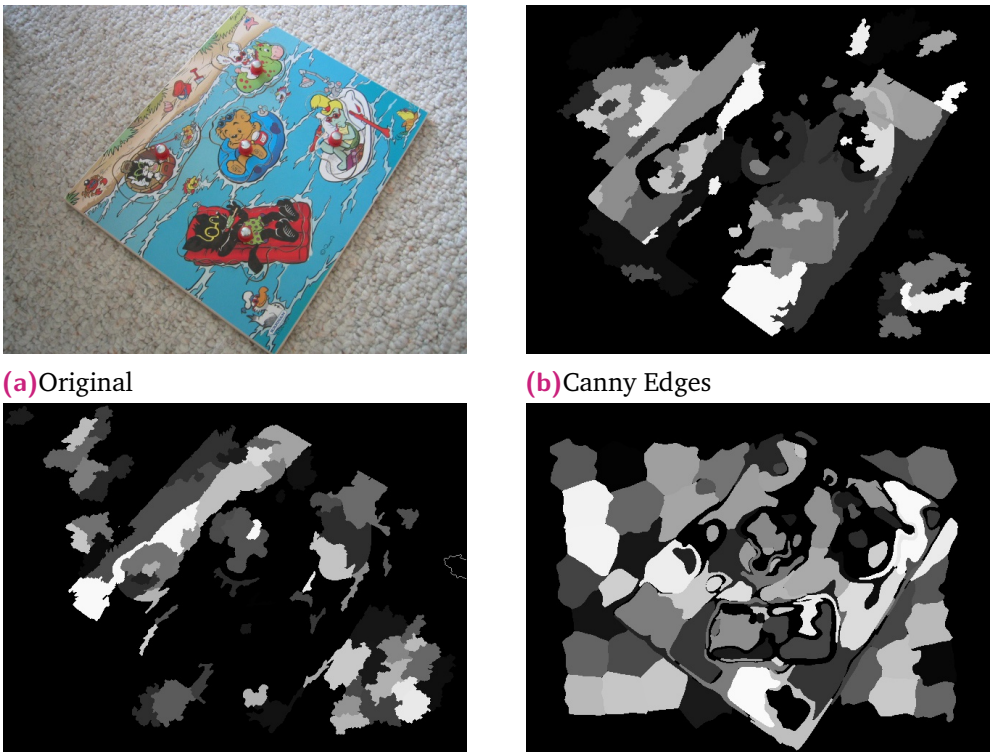
1. $image$ is the input image.
2. $gaussian\_ksize$ is the filter size of the Gaussian kernel.

### 3.2.5 Comparison

Figures 3.6 and 3.7 show the different results of two example images by applying SLIC Superpixels; Canny and Otsu together with the watershed algorithm. For these examples the following parameter values were used:

| | | | |
|---|---|---|---|
| $gaussian\_ksize$ | $=$ | $(7, 7)$ | Brush size used by Gaussian blur |
| $threshold1$ | $=$ | $20$ | First hysteresis threhold for Canny |
| $threshold2$ | $=$ | $100$ | Second hysteresis threhold for Canny |
| $min\_ratio$ | $=$ | $0.0001$ | Minimal ratio of the segment size (Section 3.2.2) |
| $max\_count$ | $=$ | $100$ | Maximum number of segments to derive |

The values were chosen purely based on human trial-and-error evaluation. Each segment is filled with a unique gray scale color (at most 255 variations). Without quantitative or reference data it's hard to conclude which method performs best. Hence we'll try to judge them based on intuition instead. In figure 3.6 the better results are computed by SLIC and the worst with Otsu. Same ranking applies to figure 3.7. It is very unlikely there is a single set of parameters that works perfectly for all possible input images.



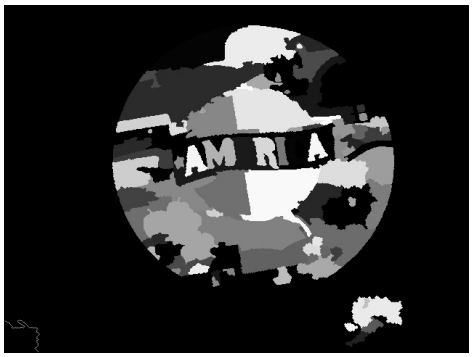
**(a)** Original **(b)** Canny Edges

**(c)** Otsu Threshold **(d)** SLIC Superpixels

**Fig. 3.6:** Segmentation of a board game


(a)Original

(b)Canny Edges



(c)Otsu Threshold

(d)SLIC Superpixels

**Fig. 3.7:** Segmentation of a bottle cap

## 3.3 Spatial Visual Phrases

If we consider a *pixel* of an image to be a *letter* in an alphabet then an *image patch* (being a set of pixels) is equivalent to a *word*. In computer vision it's commonly referred to as *visual word*. The Bag-of-words (BoW) model using the analogy of visual words has shown to perform pretty well as image retrieval method. It can be implemented using image features as described in section 3.1.

A major drawback of BoW is the lack of spatial relationship information of the words in the original image. Recent research [ZG08; Zha+09; Zha+11] has been conducted to close this *semantic* gap and therefore improve search accuracy by introducing *spatial visual phrases*. Spatial visual phrases are collections of $k$ visual words. The phrases are constructed based on the co-occurrence of two visual words. In section 4 we show how to push the text analogy further by introducing *visual sentences*.

# 4 Spatial Visual Sentences

To apply text retrieval algorithms on images based purely on content the concept of *spatial visual sentences* is introduced. A sentence can be defined [1] as:

**"A sequence of words capable of standing alone to make an assertion, ask a question, or give a command, usually consisting of a subject and a predicate containing a finite verb."**

Each spatial visual sentence is an ordered sequence of features (words) that belong together if they reside (spatial) in a common image segment (visual semantics). Such sentence could be interpreted as representing an *abstract* object with a sequence (1-dimensional) of visual features. The process of computing these sentences is inspired by concepts from text retrieval (Section 2) and CBIR (Section 3). The main goal of spatial visual sentences is to capture the semantics of images in a concise representation to allow for fast online querying. This is achieved by reducing the vast amount of mid-level descriptors (low-level being pixels themselves) to a higher level semantically meaningful groups.

Given a collection $C$ of images all the points of interests and their descriptors are extracted using a feature detector and extractor like SURF and SIFT discussed in section 3.1. Then a sample of $p$ descriptors are used as input for a clustering algorithm like k-means or SOM. The resulting $k$ cluster centers are used as the vocabulary $B$. For each image $d$ in $C$ segments are computed using SLIC superpixels, Otsu-watershed or Canny-watershed (Section 3.2). Each sentence is an ordered sequence of terms from vocabulary. A term (or word) is selected if the term is within the boundaries of the segment. The order of the terms is determined using the position the term was found in the original image. Algorithm 4 shows the full algorithm how an album $A$ is computed given $C$ as input. The rest of this section dives into the details of the main components/phases.

## 4.1 Visual Word Vocabulary

In order to reduce the complexity of searching through groups the descriptors in a group is mapped to a close representation mimicking the concept of having a vocabulary of a language. Given a collection of images all their features are detected and extracted. Feeding these features into a clustering algorithm like k-means or SOM yields $k$ clusters (Figure 4.1). Each cluster center is considered to be a word (or term) in the vocabulary. The vocabulary

[1] `http://www.wordreference.com/definition/sentence`

**Algorithm 4** Spatial Visual sentences

Input: Collection $C$ containing $n$ images
Output: Album $A$

Create album $A$
**for all** Image in $C$ **do**
    Compute and extract descriptors
**end for**
Create list of descriptors $Q$ by randomly selecting $p$ descriptors from the images
Compute the vocabulary $B$ with size $k$ by clustering $Q$ into $k$ clusters
Compute and flag noisy words in $B$
**for all** Image $d$ in $C$ **do**
    Compute the *words* in $d$ using $B$
    Identify the set of *segments* $S$ in $d$
    **for all** Segment $s$ in $S$ **do**
        Compute sentence $v$ by selecting word $w$ if its position is in $s$ and $w$ is not noisy
        $v$ is sorted by position (Starts from top-left and ends in bottom-right)
    **end for**
    Add $d$ with augmented sentences to $A$
**end for**
**return** $A$

size depicts the discriminative level and thus the effectiveness of the library to be searched: $0 \leq k \leq |C|$ where $|C|$ is the total number of features detected in the collection $C$. For $k = 0$ means no discriminative and thus equals to random search. And $k = |C|$ would be too discriminative resulting in zero speed-up. $k$ should be chosen such that there is a good trade-off between accuracy and speed. We propose the following equation:

$$k = |C| \cdot \gamma \tag{4.1}$$

Where $\gamma$ is the desired speed up expressed as a ratio of the original $|C|$ count.

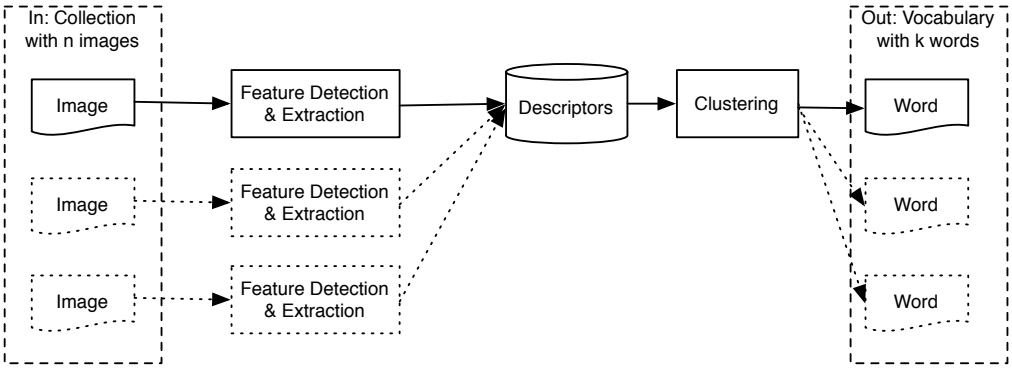



**Fig. 4.1:** Computing a vocabulary from a collection of $n$ images

For large collections the clustering process could be slow. Existing publications [Jeo+03; Now+06] suggest random sampling strategy is simple yet quite effective. Sampling can be done in two phases: images or descriptors. Image sampling is faster because a fraction of computational resources for descriptor extraction can be omitted and thus more *efficient*. However descriptors sampling is better in terms of descriptiveness of the resulting vocabulary because all descriptors have equal probability of being selected.

## 4.2 Visual Word Construction

Depending on the chosen feature descriptor each descriptor is represented as a vector $\vec{v}$ of undefined dimensions (64 for SURF and 128 for SIFT). During the visual word mapping phase each descriptor is matched against the provided vocabulary $B$ to find the best matching word $w \in B$. $w_i$ is stored as a natural number being the index of the word in $B$. Thus each image at this stage is represented as a vector $\vec{d} = (w_0, ..., w_n)$ allowing duplicates of $w_i$ in $\vec{d}$ because a word can occur multiple times in a document. After all images have been mapped to their respective $\vec{d}$ each word in $B$ is counted how often it occurs in the collection $C$. Words with high occurrences are considered to be stop-words (Section 2.1.1) and thus discarded from all $\vec{d} \in C$. Rare words (low occurrence) are also discarded because they can be considered as outliers and thus barely improve retrieval effectiveness. In the implementation these words are flagged as *noise* and kept in the vocabulary for future mappings. Efficiency is improved by skipping noisy words during construction of visual sentences.

## 4.3 Visual Sentence Construction

Each image in collection $C$ is segmented into $F$ fragments (See section 3.2). Each fragment $f_s \in F$ is defined as a set of vectors which together make up a contour. A segment is a vector $\vec{s} = (w_0, ..., w_n)$ for all $w_i \in B$ if $w_i$ is within the boundaries of $f_s$. The elements in $\vec{s}$ are sorted by the position of $w_i$ found in the original image. It begins with top-left and ends with bottom-right. This representation does not work well with transformed image like rotation or mirroring because they would yield different ordering. Depending on the application rotation can be fixed during preprocessing of scenery photographs using the horizon as reference. Or alternatively align with direction of main light source if there are shadows present. Figure 4.2 shows how an image is transformed into a list of sentences.

## 4.4 Similarity Measures

Using the computed album $A$ where each image $d$ is represented as a list of sentences $S = (\vec{s_0}, ..., \vec{s_n})$ it is possible to apply generic sequence matching algorithm and other language independent methods on two sequences. We will start with *difflib.SequenceMatcher* [2] that implements *Ratcliff/Obershelp pattern recognition* algorithm. Finally TF-IDF (Section 2.1.4)

[2] `https://docs.python.org/2/library/difflib.html`

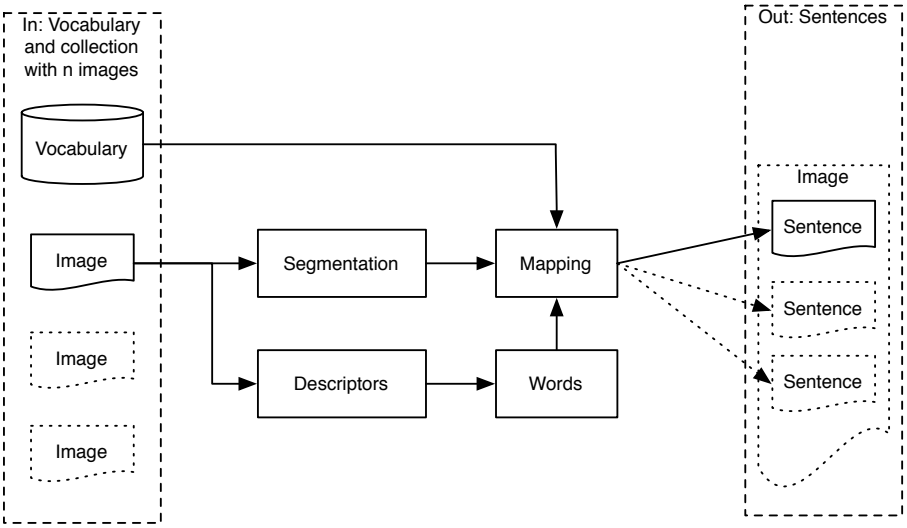



**Fig. 4.2:** Sentences of an image are computed using a global vocabulary, feature descriptors and segments derived from the image

and LSI (Section 2.3) implementations from *gensim* [3] are applied to verify the effectiveness. Similarity values are floats in the range of $[0...1]$. 0 being nothing in common and 1 means identical sequences.

### 4.4.1 Ratcliff/Obershelp

Ratcliff/Obershelp algorithm [Bla04] computes the similarity of two sequences. The ratio $\rho$ is computed with

$$\rho = \frac{2M}{T}$$

where $M$ is the number of matches and $T$ the total number of elements in both sequences. Matching characters are those in the longest common subsequence plus, recursively, matching characters in the unmatched region on either side of the longest common subsequence. Listing 2 shows how this algorithm is used to calculate the similarity of two images $image1$ and $image2$. The ratio $\rho_s$ of a sentence $s \in image1$ is the maximum ratio of $s$ against each sentence $s' \in image2$. The effective ratio $\rho'$ is the average of all the sentence ratios $\rho_i$ of $image1$.

### 4.4.2 Gensim: TF-IDF and LSI

Gensim [4] models work with a corpus. Corpus is a collection of vectors representing a document collection. An album $A$ as computed in section 4 can be transformed to a corpus in two ways:

1. CorporaOfImages: all words in the image are considered to part of a single vector

[3] `https://radimrehurek.com/gensim/tut3.html`
[4] `https://radimrehurek.com/gensim/index.html`

2. CorporaOfSentences: all words in the sentence of a particular image are considered to part of a single vector

Both methods have been implemented in listing 3. The methods implement the function *similarity(model, image)* which computes the similarity of a target image $I^T$ against the corpus. *CorporaOfImages* is simple because each image is equivalent to a vector. However *CorporaOfSentences* is slightly more complicated because the computation is done on sentences level. It keeps a mapping of $\vec{v} \rightarrow d$ where $\vec{v}$ is a vector in the corpus and $d$ is the image $\vec{v}$ belongs to. The effective ratio $\rho'(d)$ where $d \in A$ is the best model $M$ similarity value of all sentences $s \in I^T$ compared against all sentences $s' \in d$. Note that short $s \in I^T$ (lower than *min_length*, default is 5) are ignored. Listing 3 also includes wrapper implementations of two similarity models: TF-IDF and LSI. These models combined with a corpus can be used to compute the similarity of each image in $A$ against $I^T$.

# 5 Results and Evaluation

This section outlines how *Spatial Visual Sentences* are tested. We will evaluate the effectiveness with *f-score* ($F_1$) of randomly selected images from 2 datasets. The measures are defined as:

$$Precision = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{retrieved\}|}$$

is the fraction of retrieved documents that are relevant to the search. And

$$Recall = \frac{|\{relevant\} \cap \{retrieved\}|}{|\{relevant\}|}$$

is the fraction of documents that are relevant to the search has been retrieved.

$$F_1 = 2 * \frac{precision \cdot recall}{precision + recall}$$

combines both *precision* and *recall* as the *harmonic mean* called F-measure or F-score. In order to compare the performance between configurations we also introduced *Average F-Score* which is the average F-Score of all target images per algorithm configuration (See section 5.2).

The rest of this section presents the possible algorithm configurations, datasets and finally the results.

## 5.1 Datasets

To verify the effectiveness we will run the algorithm against 2 distinct datasets:

1. UKBench [NS06]: A collection of 10200 images. All images have dimensions $640 \times 480$. The set contains $\frac{10200}{4} = 2550$ groups. Each group of $4$ images (Figure 5.1) show the same object with different lighting, orientation and/or scale. Results are evaluated by checking if the retrieved images belong to the same group as the target image.

2. MIRFLICKR [HL08]: Dataset of 25000 Flickr [1] images. Each image is annotated with a list of *tags*. Evaluation of the results is done by checking the tags of the target image against those of the retrieved images. Figure 5.2 shows a few samples from the set. Unlike UKBench the image sizes, illumination, sceneries and compositions in MIRFLICKR are not uniform so it is very representative for real-world scenarios.

[1] `https://www.flickr.com/explore`

ukbench00164.jpg

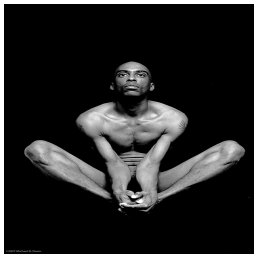
ukbench00165.jpg

ukbench00166.jpg

ukbench00167.jpg

**Fig. 5.1:** A group of 4 images showing the same subject from UKBench


im6637.jpg: daffodil, flower, fleur, yellow, trumpet, blue, light, sunlight, conservatory, 105mm, closeup, tripod, pictureproject, zixi, raw


im14091.jpg: 365days, d40, day359, nikon, raw, wednesday, bsb


im4761.jpg: tohonochul, flower, impressedbeauty


im455.jpg: rome, roma, objects, myfavourites, life, italy, eos350d, digital, canon, byfotorita, bw, white, black, 123bw, if, mani, hands, libro, book, candela, light, luce, fuoco, fire, occhiali, canoneos350d, cinquantinoinoino

**Fig. 5.2:** Four related images from MIRFLICKR with their respective tags

The results presented do not utilize the full original set. Due to high complexity only 200 images of each set were considered. Each dataset is split into a test-set of size 50 and training-set of size 150. The ratio of $1:3$ is consciously chosen to accommodate UKBench dataset described earlier. The set of relevant images from both datasets can be computed deterministically. Therefore the number of retrieved images is equal to the expected relevant images. By doing so both *precision* and *recall* values are somewhat normalized based on the target image. The ground truths per data set is computed based on their characteristics: Relevant images in UKBench are groups of 4 (one being the target image and 3 other ones are expected in the retrieved set) and an individual image in MIRFLICKR has user-defined tags. Sections 5.3 and 5.4 discussed these in more detail together with the acquired results.

## 5.2 Algorithm Configurations

Most phases of the algorithm presented in section 4 could have multiple implementations. In this section we summarize the variations creating a list of configurations of the algorithm to test against the datasets. Table 5.1 lists per phase and parameter which variations are available or interesting to evaluate. Strictly there are at least $2 \cdot 2 \cdot 3 \cdot 3 \cdot 2 \cdot 2 \cdot 1 = 144$ possible

| Phase/Parameter | 1 | 2 | 3 |
|---|---|---|---|
| Feature Descriptors | surf | sift | |
| Vocabulary clustering | kmeans | som | |
| Segmentation | otsu | canny | slic |
| Similarity measure | ratcliff | tfidf | lsi |
| Corpus mode | images | segments | |
| Vocabulary size | 1000 | 2000 | |
| Noise ratio (words discarded) | 0.20 | | |

**Tab. 5.1:** Parameter value and implementation variations

| Hardware | Description |
|---|---|
| CPU | Intel(R) Core(TM) i7-5600U CPU @ 2.60GHz |
| Number of logical cores | 4 |
| Memory | 12 GB |
| **Software** | **Version** |
| Operating System | Arch linux (Kernel 4.0.5) |
| Python | 2.7.10 |
| scikit-image | 0.11.3 |
| scipy | 0.15.1 |
| numpy | 1.9.2 |
| gensim | 0.10.3 |
| OpenCV | 2.4.10 |

**Tab. 5.2:** Test machine specifications

configurations because there are more value ranges available for *Vocabulary size* and *Noise ratio*. The results presented are based on 28 handpicked candidates and executed on a test machine described in Table 5.2. As comparison 4 additional configurations are included that implements the standard BoW algorithm varying feature descriptors and vocabulary size. So the total number of configurations sum up to 32.

## 5.3 Retrieval from UKBench

A test-set of 50 images were selected from 50 distinct groups to validate the effectiveness of the preselected configurations. The set of *relevant* images is derived from the file name: every 4 images of the sorted dataset is a sample of the same object. Given the relevant and retrieved images the three measures can be computed and shown in figure 5.3 presented as the average F-score over all target images per configuration. All of the standard BoW's perform almost twice as well as the best Visual Sentence configurations (0.913333 vs 0.508). Our best configurations are *sift-kmeans-2000-02-canny-lsi-segments* (0.508) and *sift-kmeans-2000-02-canny-tfidf-segments* (0.501333). The worst configurations all used the *SOM* vocabulary clustering together with SLIC and LSI. Overall most of the results are somewhat stable around an average F-score value of 0.45.

Figure 5.4 shows the average number of correctly retrieved images of 50 target images from UKBench. *ukbench00125.jpg* scored the best with an average relevant retrieval of 2.687500 out of 3. The worst one is *ukbench00118.jpg* with value 0.218750. The best case and worst

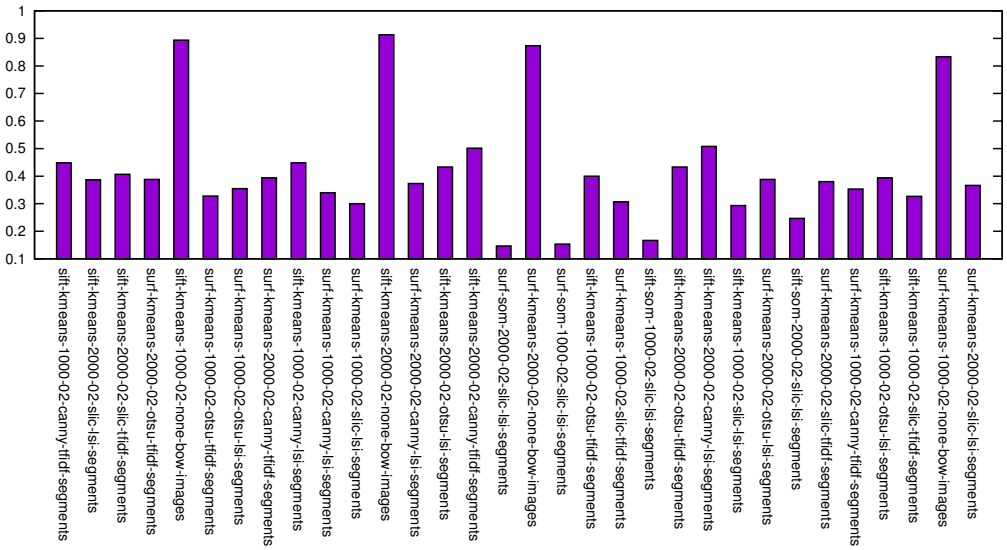



**Fig. 5.3:** Average F-Scores UKBench

case of retrieved images are shown in figures 5.5 and 5.6 respectively. Results from best case are perfectly retrieved while the worst case did not even find a single matching image out of 3 candidates.

## 5.4 Retrieval from MIRFLICKR

The MIRFLICKR dataset is a mixture of amateur and professional photographs. Unlike UKBench with its groups of 4 samples of the same subject this dataset is very unpredictable. Each sample in MIRFLICKR accompanies a list of user defined tags. Using the tags as metadata we can classify whether two pictures are related. For instance if the target image contains the tags *explore* and *flower* any other image with either *explore* or *flower* as tag is considered to be a candidate. This method is used to compute the set of *relevant* images. The dataset of 200 items is split up as 50 test images and 150 training images. They are all randomly selected from the original dataset. Figure 5.7 shows per target image on the x-axis the number of tags present of that particular image and the number of expected matches (relevant images) to be found in the training set. The images are sorted from least matches to most matches. This ordering is kept in the rest of the graphs so it's easier for comparison and reasoning.

Considering the randomness of the set creation lots of the test images had very few relevant images found in the training set (Figure 5.7). Also the diversity of the photographs makes it extremely hard to detect similarities in visual features. So therefore we will not focus on the absolute values but compare relative scores of the configurations.

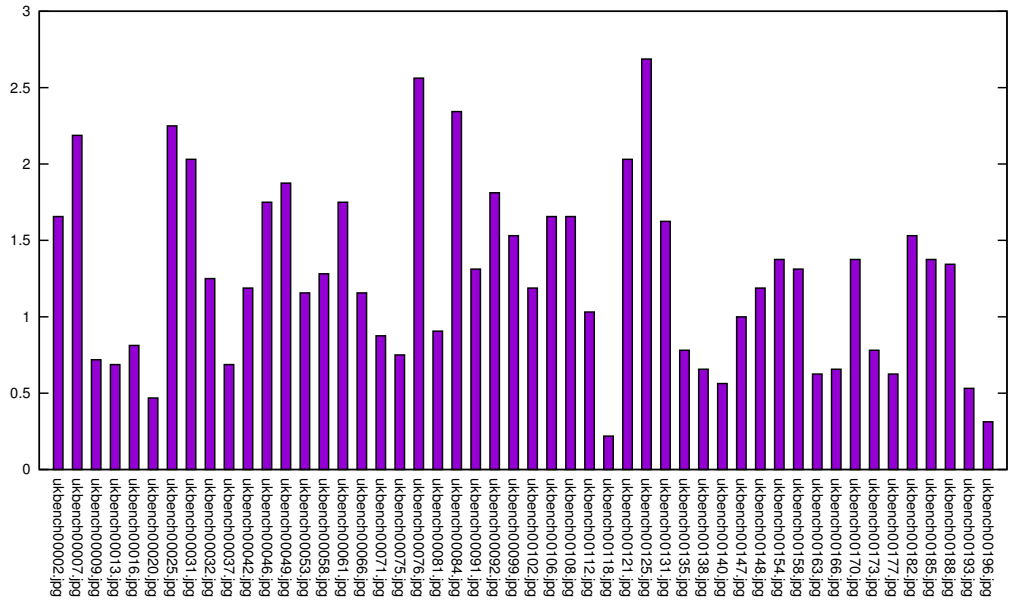
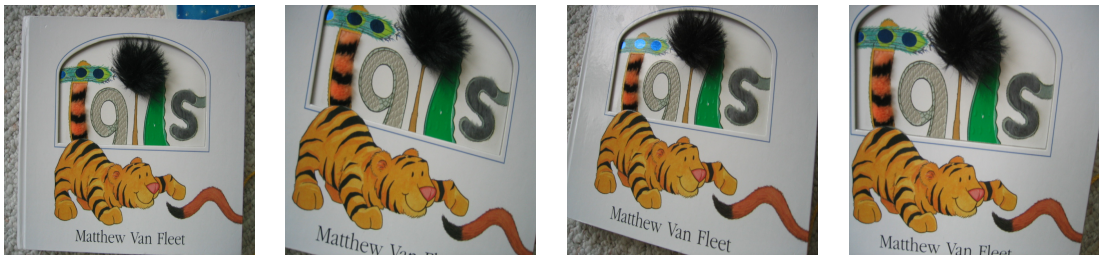



**Fig. 5.4:** Average correctly retrieved results of 50 images from UKBench




**ukbench00125.jpg (Target)** ukbench00126.jpg ukbench00124.jpg ukbench00127.jpg

**Fig. 5.5:** Overall best case: Top 3 retrieved images of *sift-kmeans-2000-02-canny-tfidf-segments* configuration of target image *ukbench00125.jpg* from UKBench


**ukbench00118.jpg (Target)** ukbench00149.jpg ukbench00144.jpg ukbench00133.jpg

**Fig. 5.6:** Overall worst case: Top 3 retrieved images of *surf-som-2000-02-slic-lsi-segments* configuration of target image *ukbench00118.jpg* from UKBench

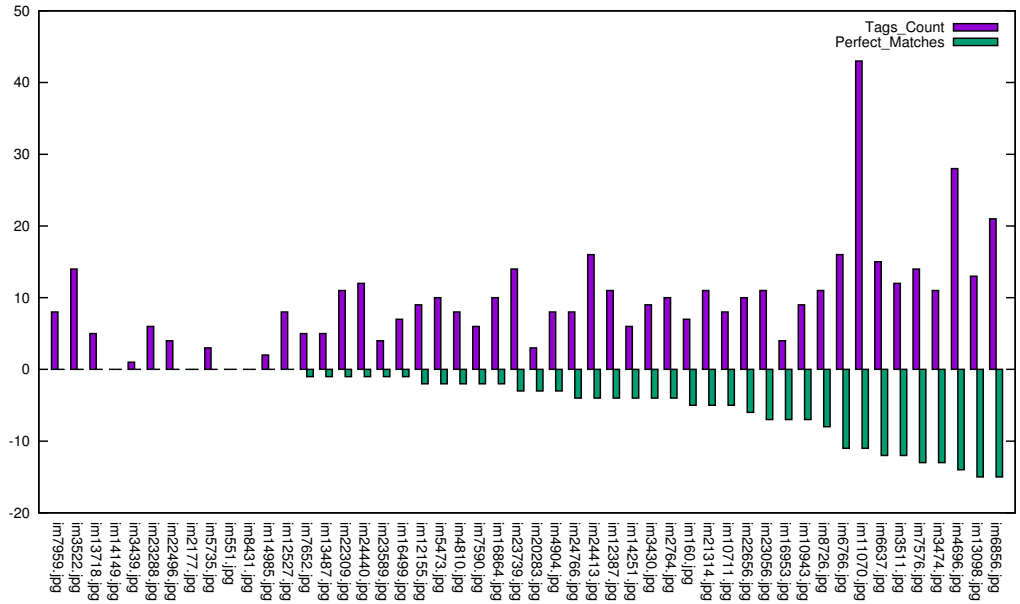



**Fig. 5.7:** 50 test images from MIRFLICKR sorted by the number of relevant images found in the training set

Average F-scores of all test images per configurations are shown in figure 5.8. The best configuration (surf-kmeans-2000-02-canny-tfidf-segments) scored the best with score 0.059066. And second best is BoW *sift-kmeans-1000-02-none-bow-images* with score 0.057416. The worst one is *sift-kmeans-1000-02-otsu-tfidf-segments* with score 0.006267.

Figure 5.9 shows the average correctly retrieved results of all configurations per test images. Overall it correlates with figure 5.7 because images to the right are expected to have higher probability of being retrieved correctly. For this dataset the image *im6856.jpg* has the highest retrieval value and *im7959.jpg* among others had zero tag correlations in the training-set. Retrieved images of both best and worst cases are depicted in figures 5.10 and 5.11. Albeit the best case results do show some visual similarities to the target image they had no tags-correlation at all just like the worst case.

Furthermore the top MIRFLICKR average F-score is very low compared to UKBENCH. They are respectively 0.913333 and 0.006267 differing by 2 orders of magnitude.

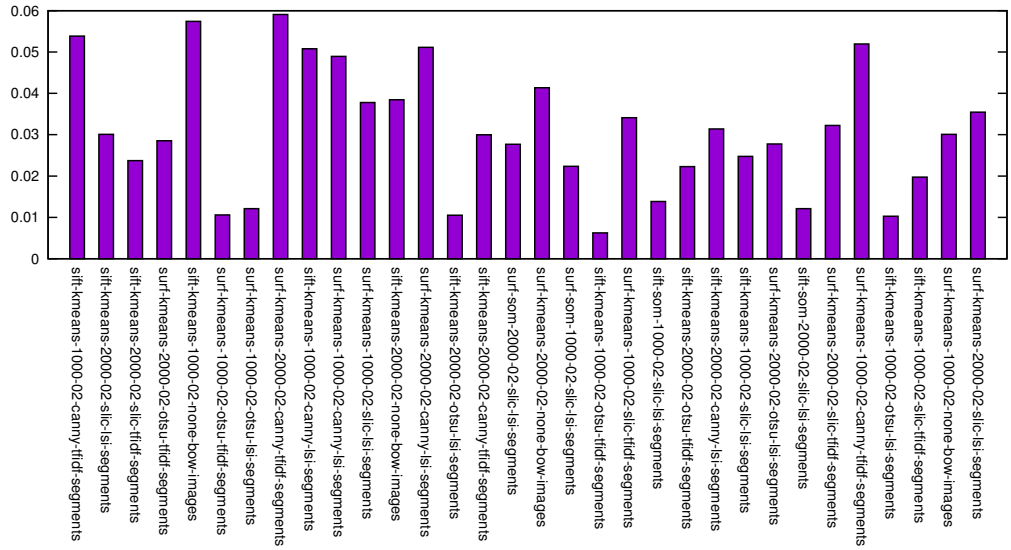



**Fig. 5.8:** Average F-Scores MIRFLICKR

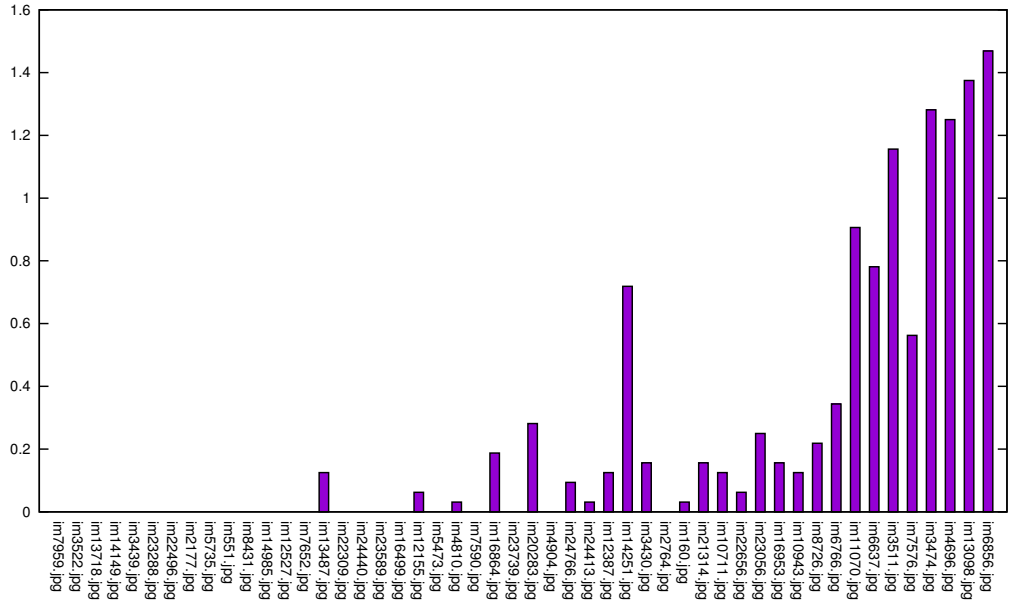



**Fig. 5.9:** Average correctly retrieved results of 50 images from MIRFLICKR

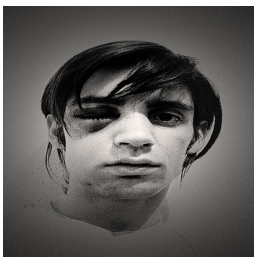
**im6856.jpg (Target):** nyc, newyork, newyorkcity, manhattan, midtown, empirestatebuilding, esb, puddle, water, reflection, upsidedown, pavement, yellow, blue, tamron1750, geotagged, maylookbetterflipped, goldstaraward, explore30jan08, i500, interestingness172

im17565.jpg: riat2007, fairford, airshow, military, aircraft, tattoo, riat, spectator, redarrows, formationflying

im15691.jpg: bento, obento, asian, lunch, tomato, roma, lobster, shellfish, crustacean, excapture, sakurakokitsa, sakurako, kitsa

im11546.jpg: sky, clouds, burbage, valley, derbyshire, canon, eos, 400d, interestingness, fern, redsky, explore8, tony, mangan, land

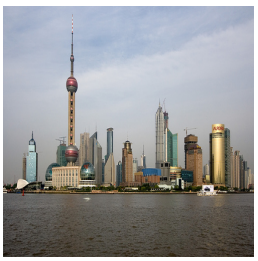
**Fig. 5.10:** Overall best case: Top 3 retrieved images of *surf-kmeans-2000-02-canny-tfidf-segments* configuration of target image *im6856.jpg* from MIRFLICKR



**im7959.jpg (Target):** etfautorretrato, autorretrato, golpe, ojo, moreton, hinchazon, cara, exhibetusfotos

im15745.jpg:

im7076.jpg: cityscape, night, road, lighttrails, streaks, moon, clouds, buildings, architecture, skyscrapers, city, urban, cranes, bridge, eureka, tower, eurekatower, rialto, eifel, charlesgrimesbridge, wurundjeriway, worldtradecentre, 3005, docklands, border, melbourne, victoria, australia, nikon, d300, 1855mmf3556gii



im22709.jpg: skyline, thebund, shanghai, pudong, china, canon, eos, 400d, canon400d, orientalpearl, jinmaobuilding, 1022mm

**Fig. 5.11:** Overall worst case: Top 3 retrieved images of *sift-kmeans-1000-02-otsu-tfidf-segments* configuration of target image *im7959.jpg* from MIRFLICKR

# 6 Conclusions

In this thesis we outlined methods related to text mining and image retrieval followed by a proposal of applying these methods to collectively implement *Spatial Visual Sentences*. This method aims to fill in the analogy of sentences in the context of text documents. While words equal to visual descriptors we derive a limited set of dictionary from the global word list applying concepts like stop words. Sentences are composed using this vocabulary and proximity of visual features (vocabulary words) that are located within the same image segment. We looked into several methods like Superpixels and watershed combined with Canny edges or Otsu threshold in partitioning an image into semantically meaningful segments. Having closed this semantic gap in the analogy between images and text documents we can leverage existing text algorithms like LSI and TFIDF to improve accuracy and hopefully also speed of offline retrieval of visually identical fragments between images.

The method has been tested against two popular datasets namely UKBench and MIRFLICKR in combination with a wide variety of configurations of the proposed algorithm. There are several configurations available because the algorithm is split up into different phases and each phase can be implemented independently. Overall we got promising results with UKBench because the dataset is focused on retrieval of the same object. In contrast MIRFLICKR composed of user-taken photographs from all over the world posed to be quite a challenge to retrieve relevant images. On the MIRFLICKR dataset, the best results were found using the configuration *surf-kmeans-canny-tfidf-segments*, however, the significance is unclear and should be further investigated. This could be related to the fact that the photographs share very little visual feature similarities. Therefore we can conclude that our algorithm is best suited to recall objects from a large collection. Although the recall results of our method were not as good as the standard Bag-of-Words model they were able to retrieve the candidates taken from different angles and scales just fine.

Because the scope of this project was purely closing this semantic gap hence we did not focus on delivering a high-performance implementation[1]. Therefore speed has not been investigated at all. In section 7 we describe possible continuations of this project.

[1] https://github.com/xiwenc/cbir-invenio

# 7 Future work

This project's focus was an attempt in closing the semantic gap of the analogy between text document and image. The results look promising for object recall however in the future we would like to look into the following questions and possible improvements:

1. **Speedup assessment and tuning**: The results presented in this thesis were mainly focused on effectiveness of the algorithm but not on the potential speedup compared to BoW. To do this a more exploratory approach like exhaustive search needs to be taken to identify which parameters constitute the global best. Evolutionary algorithms could also be applied where the fitness is a function of retrieval effectiveness and/or time required for the execution.

2. **Augment word/sentence representation with metadata**: Words in our vocabulary are defined by clustering feature descriptors. Accuracy could be improved further by extending the word vector or sentence vector with extra information like colour histogram, geolocation of the picture if present and time period when the photo was shot. These would increase descriptiveness and potentially reduce time complexity if this meta information is used as a quick filter. For instance if the target picture was taken during a road-trip through Europe in June 2014 the algorithm can narrow down the search space within that time period and location.

3. **Family photo album as dataset**: We tested the algorithm against UKBench and MIRFLICKR datasets of which fairly good results are achieved in the former but it pretty much failed with the latter. This implies it is best suited for object recalls. However to confirm this hypothesis we need a different kind of dataset: A perfect candidate would be a family photo album. The balance between objects (family members, cars, clothes, etc...) and scenery (beach, mountains, winter, house interior, etc... ) would yield better matches because of more feature descriptor similarities.

4. **Similarity measure**: The current proposed similarity measure is very optimistic because it uses similarities of best matches. As a result short sentences are more likely to yield high similarity values. We have tackled this issue by specifying a minimal sentence length in the algorithm. However a more weighted measure based on for example length of the sentence could give more accurate results.

5. **Relevance feedback application**: Spatial visual sentences can recall objects pretty accurately from large collections. To use this to our advantage users can query the system by means of sub-images. For instance the user Jim is looking for a picture of Melissa and him at the beach. A simple relevance feedback application would be: Present 10 random images from the collection to the user. If Jim recognizes a fragment of relevance like *Melissa* he marks that particular region of interest. In the

next iteration more relevant results would be presented to Jim; hopefully more picture rank up with Melissa in it. After a few iterations Jim would be able to identify the picture he sought. This example shows it is possible to construct complex queries and the sentences model fits pretty well in it.

# Appendix: Code

## segmentation

**Listing 1:** segmentation.py

```
import cv2
import numpy
import operator

from logger import logger


def watershed(image, grayed, edges, min_ratio, max_count):
    """ Applies watershed algorithm to 'image' with markers
    derived
        from 'edges'
    Args:
        image: original image
        grayed: grayed and optionally blurred version of 'image'
        edges: a binary image
        min_ratio: only contours in 'edges' with an area bigger
are used as
                markers
        max_count: maximum number of segments to derive
    Returns segments, markers, count
    """

    markers = edges.copy()
    _, markers1, _ = extract_segments(
        grayed,
        markers,
        min_ratio=min_ratio,
        max_count=max_count
    )
    markers32 = numpy.int32(markers1)
    cv2.watershed(image, markers32)
    watersheded = cv2.convertScaleAbs(markers32)
```

```
    _, edges = cv2.threshold(
        watersheded,
        1,
        255,
        cv2.THRESH_BINARY_INV
    )
    segments, markers, count = extract_segments(
        grayed,
        edges
    )
    return segments, markers, count


def canny(image, gaussian_ksize=(7, 7), threshold1=20, threshold2
=100):
    """ Computes Gaussian blurred grayscale and Canny edges
    Args:
        image = image array; use cv2.imread(...) to load from
file
        gaussian_ksize = filter size e.g. (5, 5)
        threshold1 = first threshold of the hysteresis procedure
        threshold2 = second threshold of the hysteresis procedure
    Returns (grayscale, edges)
    """
    if len(image.shape) == 3:
        grayed = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    elif len(image.shape) == 2:
        grayed = image
    else:
        raise Exception("Unsupported input image")

    blurred = cv2.GaussianBlur(grayed, gaussian_ksize, 0)
    edges = cv2.Canny(blurred, threshold1, threshold2)
    return (grayed, edges)


def otsu(image, gaussian_ksize=(7, 7)):
    """ Computes Gaussian blurred grayscale and Otsu threshold
    edges
    Args:
        image = image array; use cv2.imread(...) to load from
file
        gaussian_ksize = filter size e.g. (5, 5)
    Returns (grayscale, edges)
    """
    grayed = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(grayed, gaussian_ksize, 0)
```

```
    ret, edges = cv2.threshold(
        blurred,
        0,
        255,
        cv2.THRESH_BINARY + cv2.THRESH_OTSU
    )
    return (grayed, edges)


def seg_otsu_watershed(image, min_ratio=0.0001, max_count=100):
    grayed, edges = otsu(image)
    return watershed(image, grayed, edges, min_ratio, max_count)


def seg_canny_watershed(image, min_ratio=0.0001, max_count=100):
    grayed, edges = canny(image)
    return watershed(image, grayed, edges, min_ratio, max_count)


def seg_slic(image, min_ratio=0, max_count=100):
    from skimage.segmentation import slic
    from skimage.segmentation import mark_boundaries
    from skimage import img_as_ubyte

    grayed = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    segments = slic(image, sigma=5)
    black = numpy.zeros(image.shape, numpy.uint8)
    edges_sci = mark_boundaries(black, segments, color=(1, 1, 1))
    edges_gray = cv2.cvtColor(img_as_ubyte(edges_sci), cv2.
    COLOR_BGR2GRAY)
    ret, edges = cv2.threshold(
        edges_gray,
        80,
        255,
        cv2.THRESH_BINARY
    )
    segments, markers, count = extract_segments(
        grayed,
        edges,
        min_ratio,
        max_count
    )
    return segments, markers, count


def extract_segments(grayed, edges, min_ratio=0, max_count=100):
    contours, hierarchy = cv2.findContours(
```

```
    edges,
    cv2.cv.CV_RETR_TREE,
    cv2.cv.CV_CHAIN_APPROX_SIMPLE
)
markers = numpy.zeros(grayed.shape, numpy.uint8)
total_area = edges.shape[0] * edges.shape[1]
contours_total = len(contours)

contours_no_child = []
for i in range(contours_total):
    h = hierarchy[0][i]
    if h[2] == -1:
        contours_no_child.append(contours[i])
    else:
        logger.debug('Skipped child contour {i}'.format(i=i))

contours_filtered = []
for contour in contours_no_child:
    area = float(cv2.contourArea(contour))
    ratio = area / total_area
    if ratio >= min_ratio:
        contours_filtered.append((contour, area))
    else:
        logger.debug('Skipped contour with ratio %f' % ratio)

contours_filtered_sorted = sorted(
    contours_filtered,
    key=operator.itemgetter(1),
    reverse=True
)

assert(max_count < 255 - 2)
contours_count = min(len(contours_filtered_sorted), max_count
)
contours_capped = contours_filtered_sorted[:contours_count]
contours_no_tuple = [c for c, v in contours_capped]

# color 1: border
colors = range(2, 255)
import random
random.shuffle(colors)
for i in range(contours_count):
    cv2.drawContours(markers, contours_no_tuple, i, colors[i
    ], -1)
logger.info(
    'Segments found {total}, {count} satisfied min_ratio'.
    format(
```

```
            total=contours_total,
            count=contours_count
        ))
    segments = contours_no_tuple
    return segments, markers, contours_count
```

## similarity

Listing 2: similarity.py

```
class SentenceDiff(object):

    @staticmethod
    def distance(image1, image2):
        assert len(image1.sentences) > 0
        assert len(image2.sentences) > 0

        ratios = []
        for s in image1.sentences:
            ratios.append(max(map(
                lambda x: SentenceDiff.distance_sentence(s, x),
                image2.sentences
            )))

        ratio_total = sum(ratios)
        count = len(ratios)
        return ratio_total / float(count)

    @staticmethod
    def distance_sentence(sentence1, sentence2):
        sm = difflib.SequenceMatcher(
            None,
            [word.value for word in sentence1.words],
            [word.value for word in sentence2.words],
            autojunk=False
        )
        return sm.ratio()
```

## gensim

Listing 3: gensim.py

```
from gensim import models, similarities


def document(sequence):
    counts = {}
```

```
    for item in sequence:
        if item in counts:
            counts[item] = counts[item] + 1
        else:
            counts[item] = 1

    result = []
    for k, v in counts.iteritems():
        result.append((k, v))
    return result


class LSI(object):
    def __init__(self, corpus, num_features):
        self.lsi = models.LsiModel(corpus, num_topics=
        num_features)
        self.index = similarities.SparseMatrixSimilarity(
            self.lsi[corpus],
            num_features=num_features
        )

    def similarity(self, doc):
        sims = self.index[self.lsi[doc]]
        return sims


class TFIDF(object):
    def __init__(self, corpus, num_features):
        self.tfidf = models.TfidfModel(corpus)
        self.index = similarities.SparseMatrixSimilarity(
            self.tfidf[corpus],
            num_features=num_features
        )

    def similarity(self, doc):
        sims = self.index[self.tfidf[doc]]
        return sims


class CorporaOfImages(object):
    def __init__(self, album):
        self.album = album
        self.index = []
        self.corpus = []

    def get_corpus(self):
        if len(self.corpus) > 0:
```

```
            return self.corpus

        for image in self.album.images:
            sequences = []
            for sentence in image.sentences:
                sequence = sentence.export()
                sequences.extend(sequence)
            doc = document(sequences)
            self.corpus.append(doc)
            self.index.append(image)

        return self.corpus

    def similarity(self, model, image):
        sequences = []
        for sentence in image.sentences:
            sequence = sentence.export()
            sequences.extend(sequence)
        doc = document(sequences)
        sims = model.similarity(doc)

        fitness = {}
        for i, j in list(enumerate(sims)):
            filename = self.index[i].filename
            fitness[filename] = j
        return fitness


class CorporaOfSentences(object):
    def __init__(self, album, min_length=5):
        self.album = album
        self.index = []
        self.corpus = []
        self.min_length = min_length

    def get_corpus(self):
        if len(self.corpus) > 0:
            return self.corpus

        for image in self.album.images:
            for sentence in image.sentences:
                sequence = sentence.export()
                doc = document(sequence)
                if len(sequence) >= self.min_length:
                    self.corpus.append(doc)
                    self.index.append(image)
```

```
        return self.corpus

    def similarity(self, model, image):
        result = None
        for sentence in image.sentences:
            sequence = sentence.export()
            doc = document(sequence)
            if len(sequence) >= self.min_length:
                sims = model.similarity(doc)
                if result is None:
                    result = sims
                else:
                    result = [max(i, j) for i, j in zip(result,
                    sims)]

        if result is None:
            result = []

        fitness = {}
        for i, j in list(enumerate(result)):
            filename = self.index[i].filename
            if filename not in fitness:
                fitness[filename] = j
            else:
                fitness[filename] = max(fitness[filename], j)
        return fitness
```

# Bibliography


[Abb+08] Osama Abu Abbas et al. *Comparisons Between Data Clustering Algorithms.* 3. 2008, pp. 320–325 (cit. on p. 9).

[Ach+10] Radhakrishna Achanta, Appu Shaji, Kevin Smith, et al. *Slic superpixels*. 2010 (cit. on pp. 15, 16).

[Ach+12] Radhakrishna Achanta, Appu Shaji, Kevin Smith, et al. *SLIC superpixels compared to state-of-the-art superpixel methods*. 11. IEEE, 2012, pp. 2274–2282.

[AK+12] Ali Abdo Mohammed Al-Kubati, Jamil AM Saif, and Murad AA Taher. *Evaluation of canny and otsu image segmentation*. 2012, pp. 24–25 (cit. on p. 15).

[Ala+12] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. *Freak: Fast retina keypoint*. Ieee. 2012, pp. 510–517 (cit. on p. 13).

[Ami13] Nashruddin Amin. *Count and segment overlapping objects with Watershed and Distance Transform*. `https://opencv-code.com/tutorials/count-and-segment-overlapping-objects-with-watershed-and-distance-transform/`. Jan. 2013.

[AZ12] Relja Arandjelovic and Andrew Zisserman. *Three things everyone should know to improve object retrieval*. IEEE. 2012, pp. 2911–2918.

[Ban15] Ravimal Bandara. *Bag-of-Features Descriptor on SIFT Features with OpenCV (BoF-SIFT)*. `http://www.codeproject.com/Articles/619039/Bag-of-Features-Descriptor-on-SIFT-Features-with-O`. Feb. 2015 (cit. on p. 14).

[Bay+08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. *Speeded-up robust features (SURF)*. 3. Elsevier, 2008, pp. 346–359 (cit. on pp. 12, 13).

[Beh+03] Clifford A Behrens, Dennis E Egan, Yu-Yun Ho, Carol Lochbaum, and Mark Rosenstein. *Automatic recommendation of products using latent semantic indexing of content*. US Patent 6,615,208. Google Patents, 2003 (cit. on p. 10).

[BG09] Gertjan J Burghouts and Jan-Mark Geusebroek. *Performance evaluation of local colour invariants*. 1. Elsevier, 2009, pp. 48–62.

[BL79] Serge Beucher and Christian Lantuéjoul. *Use of watersheds in contour detection*. 1979 (cit. on p. 16).

[Bla04] Paul E Black. *Ratcliff/Obershelp pattern recognition*. 2004 (cit. on p. 24).

[Cal+10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. *Brief: Binary robust independent elementary features*. Springer, 2010, pp. 778–792 (cit. on p. 13).

[Can86] John Canny. *A computational approach to edge detection*. 6. IEEE, 1986, pp. 679–698 (cit. on pp. 12, 17).

[Dat+05] Ritendra Datta, Jia Li, and James Z Wang. *Content-based image retrieval: approaches and trends of the new age*. ACM. 2005, pp. 253–262 (cit. on p. 12).

[Dee+89] Scott C Deerwester, Susan T Dumais, George W Furnas, et al. *Computer information retrieval using latent semantic structure*. US Patent 4,839,853. Google Patents, 1989 (cit. on p. 10).

[Del+12] Andrew Delong, Anton Osokin, Hossam N Isack, and Yuri Boykov. *Fast approximate energy minimization with label costs*. 1. Springer, 2012, pp. 1–27 (cit. on p. 15).

[Den+07] Hongli Deng, Wei Zhang, E. Mortensen, T. Dietterich, and L. Shapiro. *Principal Curvature-Based Region Detector for Object Recognition*. 2007, pp. 1–8 (cit. on p. 12).

[Des+08] Thomas Deselaers, Daniel Keysers, and Hermann Ney. *Features for image retrieval: an experimental comparison*. 2. Springer, 2008, pp. 77–107 (cit. on p. 2).

[Erk12] Katrin Erk. *Vector space models of word meaning and phrase meaning: A survey*. 10. Wiley Online Library, 2012, pp. 635–653.

[FH04] Pedro F Felzenszwalb and Daniel P Huttenlocher. *Efficient graph-based image segmentation*. 2. Springer, 2004, pp. 167–181 (cit. on p. 15).

[For07] P.-E. Forssen. *Maximally Stable Colour Regions for Recognition and Matching*. 2007, pp. 1–8 (cit. on p. 12).

[Fre+02] Jordi Freixenet, Xavier Muñoz, David Raba, Joan Martí, and Xavier Cufí. „Yet another survey on image segmentation: Region and boundary information integration“. In: *Computer Vision—ECCV 2002*. Springer, 2002, pp. 408–422 (cit. on p. 15).

[Gil+10] Arturo Gil, OscarMartinez Mozos, Monica Ballesta, and Oscar Reinoso. *A comparative evaluation of interest point detectors and local descriptors for visual SLAM*. 6. 2010, pp. 905–920 (cit. on p. 12).

[Gre10] Andrew Greensted. *Otsu Thresholding*. `http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html`. June 2010 (cit. on p. 17).

[HL08] Mark J. Huiskes and Michael S. Lew. *The MIR Flickr Retrieval Evaluation*. Vancouver, Canada: ACM, 2008 (cit. on p. 27).

[Hot+03] Andreas Hotho, Steffen Staab, and Gerd Stumme. *Ontologies improve text document clustering*. IEEE. 2003, pp. 541–544 (cit. on p. 6).

[Hot+05] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. *A Brief Survey of Text Mining*. 1. 2005, pp. 19–62 (cit. on pp. 5, 7).

[Jeo+03] Jiwoon Jeon, Victor Lavrenko, and Raghavan Manmatha. *Automatic image annotation and retrieval using cross-media relevance models*. ACM. 2003, pp. 119–126 (cit. on p. 23).

[JG09] Luo Juan and Oubong Gwun. *A comparison of sift, pca-sift and surf*. 4. 2009, pp. 143–152 (cit. on pp. 12, 13).

[KM13] Stuti Karol and Veenu Mangat. *Evaluation of text document clustering approach based on particle swarm optimization*. 2. 2013, pp. 69–90 (cit. on p. 8).

[Kof13] Kurt Koffka. *Principles of Gestalt psychology*. Routledge, 2013 (cit. on p. 2).

[Koh82] Teuvo Kohonen. *Self-organized formation of topologically correct feature maps*. 1. Springer, 1982, pp. 59–69 (cit. on p. 9).

[Koo+11] Gert Kootstra, Niklas Bergström, and Danica Kragic. *Gestalt principles for attention and segmentation in natural and artificial vision systems*. eSMCs. 2011 (cit. on p. 2).

[Leu+11] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. *BRISK: Binary robust invariant scalable keypoints*. IEEE. 2011, pp. 2548–2555 (cit. on p. 13).

[Lew+06] Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. *Content-based multimedia information retrieval: State of the art and challenges*. 1. ACM, 2006, pp. 1–19 (cit. on pp. 11, 12).

[Liu+07] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. *A survey of content-based image retrieval with high-level semantics*. 1. Pergamon, 2007, pp. 262–282 (cit. on pp. 1, 11, 12).

[LK02] Edda Leopold and Jörg Kindermann. *Text categorization with support vector machines. How to represent texts in input space?* 1-3. Springer, 2002, pp. 423–444 (cit. on p. 6).

[LL94] Thomas K Landauer and Michael L Littman. *Computerized cross-language document retrieval using latent semantic indexing*. US Patent 5,301,109. Google Patents, 1994 (cit. on p. 10).

[LM01] L Luccheseyz and SK Mitray. *Color image segmentation: A state-of-the-art survey*. 2. 2001, pp. 207–221 (cit. on p. 15).

[Lou+00] E. Loupias, N. Sebe, S. Bres, and J.-M. Jolion. *Wavelet-based salient points for image retrieval*. 2000, 518–521 vol.2 (cit. on p. 13).

[Low04] David G Lowe. *Distinctive image features from scale-invariant keypoints*. 2. Springer, 2004, pp. 91–110 (cit. on pp. 12, 13).

[Low99] D.G. Lowe. *Object recognition from local scale-invariant features*. 1999, 1150–1157 vol.2 (cit. on p. 12).

[LS89] Karen E Lochbaum and Lynn A Streeter. *Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval*. 6. Elsevier, 1989, pp. 665–676 (cit. on p. 7).

[Mal+01] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. *Contour and texture analysis for image segmentation*. 1. Springer, 2001, pp. 7–27 (cit. on p. 2).

[MM12] Ondrej Miksik and Krystian Mikolajczyk. *Evaluation of local detectors and descriptors for fast feature matching*. IEEE. 2012, pp. 2681–2684.

[MY09] Jean-Michel Morel and Guoshen Yu. *ASIFT: A new framework for fully affine invariant image comparison*. 2. Society for Industrial and Applied Mathematics, 2009, pp. 438–469 (cit. on p. 12).

[Mül+10] Henning Müller, Paul Clough, Thomas Deselaers, Barbara Caputo, and Image CLEF. *Experimental evaluation in visual information retrieval*. Springer, 2010 (cit. on p. 1).

[Nam14] Rajesh Namase. *Latent Semantic Indexing: How Does the LSI Algorithm Work?* `http://www.techlila.com/latent-semantic-indexing/`. Apr. 2014 (cit. on p. 10).

[Now+06] Eric Nowak, Frédéric Jurie, and Bill Triggs. „Sampling strategies for bag-of-features image classification“. In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 490–503 (cit. on p. 23).

[NS06] D. Nistér and H. Stewénius. „Scalable Recognition with a Vocabulary Tree“. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. **oral presentation**. June 2006, pp. 2161–2168 (cit. on p. 27).

[Ots75] Nobuyuki Otsu. *A threshold selection method from gray-level histograms*. 285-296. 1975, pp. 23–27 (cit. on p. 17).

[Pap+98] Christos H Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. *Latent semantic indexing: A probabilistic analysis*. ACM. 1998, pp. 159–168 (cit. on p. 10).

[Pen+13] Bo Peng, Lei Zhang, and David Zhang. *A survey of graph theoretical approaches to image segmentation*. 3. Pergamon, 2013, pp. 1020–1038 (cit. on p. 15).

[Pha+05] Duc Truong Pham, Stefan S Dimov, and CD Nguyen. *Selection of K in K-means clustering*. 1. SAGE Publications, 2005, pp. 103–119 (cit. on p. 8).

[Pro15a] OpenCV Project. *Introduction to SIFT (Scale-Invariant Feature Transform)*. `http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html`. Feb. 2015 (cit. on p. 13).

[Pro15b] OpenCV Project. *Introduction to SURF (Speeded-Up Robust Features)*. `http://docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html`. Feb. 2015 (cit. on p. 14).

[RD06] E. Rosten and T. Drummond. *Machine Learning for High-speed Corner Detection*. 2006 (cit. on p. 12).

[Ren] X Ren. *Superpixel: Empirical Studies and Applications* (cit. on p. 15).

[RM00] Jos BTM Roerdink and Arnold Meijster. *The watershed transform: Definitions, algorithms and parallelization strategies*. 1. IOS Press, 2000, pp. 187–228 (cit. on p. 17).

[RM03] Xiaofeng Ren and Jitendra Malik. *Learning a classification model for segmentation*. IEEE. 2003, pp. 10–17 (cit. on p. 15).

[Ros+08] E. Rosten, R. Porter, and T. Drummond. *Faster and better: a machine learning approach to corner detection*. 2008 (cit. on p. 12).

[Rub+11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. *ORB: an efficient alternative to SIFT or SURF*. IEEE. 2011, pp. 2564–2571 (cit. on pp. 12, 13).

[Sal+94] Gerard Salton, James Allan, and Chris Buckley. *Automatic structuring and retrieval of large text files*. 2. ACM, 1994, pp. 97–108 (cit. on p. 7).

[Seb02] Fabrizio Sebastiani. *Machine learning in automated text categorization*. 1. ACM, 2002, pp. 1–47 (cit. on p. 8).

[Sin+11] Vivek Kumar Singh, Nisha Tiwari, and Shekhar Garg. *Document clustering using k-means, heuristic k-means and fuzzy c-means*. IEEE. 2011, pp. 297–301 (cit. on p. 8).

[Sme+00] Arnold WM Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. *Content-based image retrieval at the end of the early years*. 12. IEEE, 2000, pp. 1349–1380 (cit. on p. 1).

[Sob90] I. Sobel. *An Isotropic 3x3 image gradient operator*. Academic Press, 1990 (cit. on p. 12).

[SS12] Nidhi Singh and Divakar Singh. *Performance Evaluation of K-Means and Heirarichal Clustering in Terms of Accuracy and Running Time*. 3. Citeseer, 2012, pp. 4119–4121 (cit. on p. 8).

[Ste+00] Michael Steinbach, George Karypis, Vipin Kumar, et al. *A comparison of document clustering techniques*. 2000 (cit. on p. 8).

[Tho+10] Bart Thomee, Erwin M Bakker, and Michael S Lew. *TOP-SURF: a visual words toolkit*. ACM. 2010, pp. 1473–1476 (cit. on p. 1).

[TL12] Bart Thomee and Michael S Lew. „Interactive search in image retrieval: a survey“. In: *International Journal of Multimedia Information Retrieval* 1.2 (2012), pp. 71–86 (cit. on p. 1).

[TM08] Tinne Tuytelaars and Krystian Mikolajczyk. *Local invariant feature detectors: a survey*. 3. Now Publishers Inc., 2008, pp. 177–280 (cit. on pp. 11, 12).

[Tuy+10] Tinne Tuytelaars, Christoph H Lampert, Matthew B Blaschko, and Wray Buntine. *Unsupervised object discovery: A comparison*. 2. Springer, 2010, pp. 284–302 (cit. on pp. 11, 12).

[Unk15] Unknown. *Untitled*. `http://johannes.lampel.net/bll137-Dateien/image044.jpg`. Feb. 2015 (cit. on p. 9).

[VS10] T Velmurugan and T Santhanam. *Performance evaluation of k-means and fuzzy c-means clustering algorithms for statistical distributions of input data points*. 3. 2010, pp. 320–330 (cit. on p. 8).

[WH+04] Peter Wiemer-Hastings, K Wiemer-Hastings, and A Graesser. *Latent semantic analysis*. Citeseer. 2004, pp. 1–14 (cit. on p. 10).

[Yan+07] Jun Yang, Yu-Gang Jiang, Alexander G Hauptmann, and Chong-Wah Ngo. *Evaluating bag-of-visual-words representations in scene classification*. ACM. 2007, pp. 197–206 (cit. on p. 1).

[Yeo+05] NC Yeo, KH Lee, YV Venkatesh, and Sim Heng Ong. *Colour image segmentation using the self-organizing map and adaptive resonance theory*. 12. Elsevier, 2005, pp. 1060–1079 (cit. on p. 15).

[YM12] Faliu Yi and Inkyu Moon. *Image segmentation: A survey of graph-cut methods*. IEEE. 2012, pp. 1936–1941 (cit. on p. 15).

[Yua+07] Junsong Yuan, Ying Wu, and Ming Yang. *Discovery of collocation patterns: from visual words to visual phrases*. IEEE. 2007, pp. 1–8 (cit. on pp. 1, 2).

[Zag+11] Konstantinos Zagoris, Savvas A Chatzichristofis, and Avi Arampatzis. *Bag-of-visual-words vs global image descriptors on two-stage multimodal retrieval*. ACM. 2011, pp. 1251–1252 (cit. on pp. 1, 12).

[ZG08] Qing-Fang Zheng and Wen Gao. *Constructing visual phrases for effective and efficient object-based image retrieval*. 1. ACM, 2008, p. 7 (cit. on p. 19).

[Zha+08] Hui Zhang, Jason E Fritts, and Sally A Goldman. *Image segmentation evaluation: A survey of unsupervised methods*. 2. Elsevier, 2008, pp. 260–280 (cit. on p. 15).

[Zha+09] Shiliang Zhang, Qi Tian, Gang Hua, Qingming Huang, and Shipeng Li. *Descriptive visual words and visual phrases for image applications*. ACM. 2009, pp. 75–84 (cit. on p. 19).

[Zha+11] Yimeng Zhang, Zhaoyin Jia, and Tsuhan Chen. *Image retrieval with geometry-preserving visual phrases*. IEEE. 2011, pp. 809–816 (cit. on pp. 1, 19).

[Zhe+06] Qing-Fang Zheng, Wei-Qiang Wang, and Wen Gao. *Effective and efficient object-based image retrieval using visual phrases*. ACM. 2006, pp. 77–80 (cit. on pp. 1, 2).

# List of Figures

# List of Tables