Technische Universität Kaiserslautern
Fachbereich Informatik

Bachelorarbeit

Statistical Detection of Non-Scene Text for Content-based Video Retrieval

von

Markus Koch

11. März 2008

Themensteller:
Prof. Dr. Thomas Breuel
AG Bildverstehen und Mustererkennung

Betreuer:
Dipl.-Inf. Adrian Ulges

Erklärung

Ich versichere hiermit, dass ich die vorliegende Bachelorarbeit mit dem Thema "Statistical Detection of Non-Scene Text for Content-based Video Retrieval" selbstständig verfasst habe und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen wurden, habe ich durch die Angabe der Quelle, auch der benutzten Sekundärliteratur, als Entlehnung kenntlich gemacht.

Kaiserslautern, den 11. März 2008 Unterschrift

Kurzfassung

Mit der steigenden Bedeutung von digitalem Video wächst auch das Bedürfnis, die größer werdenden Videoarchive komfortabel durchsuchen und verwalten zu können. Ein Ansatz, der viel Beachtung erfährt, ist der Bereich des Content-based Video Retrieval. Bei diesen Systemen geht es darum, den Nutzern einen Zugriff abseits von benutzerdefinierten Schlüsselwörtern zu bieten und stattdessen den wirklichen Inhalt des Videos als Grundlage für die Navigation zu nutzen. Sie erkennen visuelle Eigenschaften wie Farbe und Bewegung, und setzen diese in Zusammenhang mit dem semantischen Kontext des Videos.

Nicht nur im Bereich von Online-Videoportalen fügen die Ersteller oder Vertreiber von Videomaterial oft nachträglich Bilder mit Text-Information (Nicht-Szenenbilder) ein, um den Ort, das Datum oder die Thematik ihrer Videos zu propagieren. Traditionelle Content-based Video Retrieval Systeme nutzen diese wertvolle Information jedoch in der Regel nicht.

Diese Arbeit beschreibt eine Ergänzung eines solchen Systems, welche darauf abzielt, Nicht-Szenenbilder zu erkennen, um den darin enthaltenen Text zu extrahieren. Dieser Nicht-Szenen Text wird durch einen Schlüsselwortabgleich auf verwertbare Information hin untersucht, die dann für die weitere Verwendung im System zur Verfügung steht.

Abstract

Digital video has received a massive boom over the past years, and the growing video archives make it necessary to develop alternatives to the traditional text search based on keywords. Content-based video retrieval aims at using the visual content of a video instead of focusing on user-generated metadata. To achieve this, these systems learn the interrelationships between low-level visual features like color or motion and the semantic context of the video.

Videos published on online video portals often contain non-scene images added by the creator or publisher of the video. The text found in these images usually contains information about the source, the date or the topic of the video. Traditional content-based video retrieval systems do not make a difference at this point, so this valuable information remains unused.

This thesis describes a modification of a content-based video retrieval system, which strives to detect these non-scene images and to extract the contained text. This non-scene text is then interpreted using a keyword spotting mechanism, and the text information obtained is used in the further processing of the video.

Contents

Chapter 1

Introduction

1.1 Digital Videos

During the last few years there has been a significant growth in the use of digital videos. Today, more people than ever before use digital videos to spread information, state their opinions, or even share private moments of their lives with others.

Video portals like YouTube[1] or Dailymotion[2] play an important role in this recent development, as they deliver an easily accessible and free-of-charge way for users world-wide to store and share their digital videos. With the rapid spread of broadband Internet access their popularity has increased enormously, resulting in strongly growing video databases and a massive raise of video streaming traffic over the last few years. According to a study released by Ellacoya Networks in 2007 [9], the market leader for streaming video, YouTube, is responsible for nearly 10 % of all traffic on the Internet.

YouTube currently hosts about 72,500,000 videos with approximately 200.000 new videos a day [28]. With the size of the video databases growing, the need for effective searching and browsing techniques also gains even more importance. Most video portals today allow the visitors to browse and search their video databases using tags and other user-generated metadata, thus relying on manual annotation of the videos. This approach needs a great amount of labor, and is subjective and inflexible.

[1] www.youtube.com
[2] www.dailymotion.com

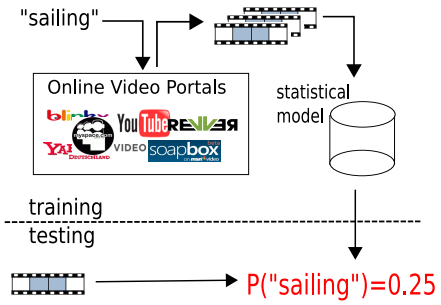


Figure 1.1: TubeTagger learns the interrelationships between the visual features and the semantic concept "sailing" by downloading a set of videos and training the statistical models using the extracted features.

1.2 Content-based Video Retrieval

A rather new way to access large video databases are content-based video retrieval systems. These system focus on using the actual visual content of videos for annotation and querying purposes, instead of relying on tags and keywords provided by users. They allow the users to perform search tasks based on visual similarities or descriptions of the visual content of a video. Therefore, the content of a video is extracted and represented in the form of visual features, which are used for indexing the videos. Some examples for visual features that are used in content-based video retrieval include color, texture and motion.

Many projects have emerged in this field, for example the Virage Video engine [15] and CueVideo [19] among the first ones. A newer example for a video retrieval system that provides a content-based access is InViRe, a project currently conducted by the German Research Center for Artificial Intelligence(DFKI)[3]. It features a special tagging system, TubeTagger, where the focus of this thesis lies.

Content-based video retrieval received a massive raise in attention over the last years, particularly because of the massive growth of the video databases. Since 2003, the annual video retrieval contest TRECVID[4] is held, which aims at evaluating and promoting the progress made in this field of research.

[3]http://www.dfki.de/web/forschung/iupr/projekte/
[4]http://www-nlpir.nist.gov/projects/trecvid/

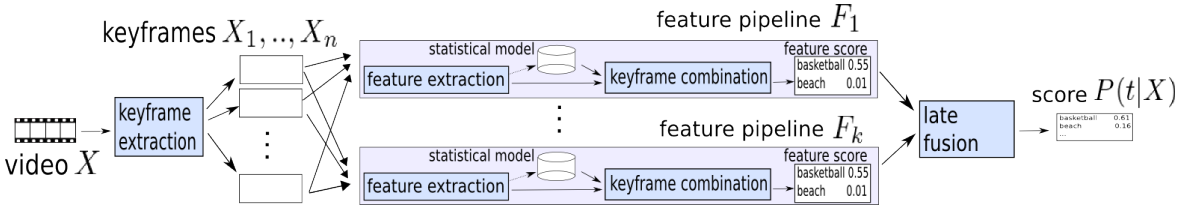


Figure 1.2: The tagging system TubeTagger in detail. Several features are extracted from a set of keyframes, which are used to train the statistical models. A new video is tagged by classifying its features and combining the results to a single score [26].

1.3 TubeTagger - a Tagging System for Video Retrieval

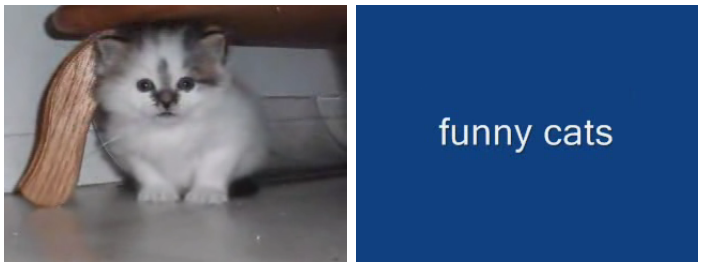
TubeTagger is a tagging system which works as a high-level semantic concept detection [25]. It uses several visual features extracted from a set of representative keyframes, namely color, texture, motion and visual words. During a training, the tagging system learns the interrelationships between these low-level visual features and the semantic contexts of the corresponding training videos. A suitable training set can be obtained by automatically downloading videos from a video portal like YouTube, whereas the tags provided by the users who uploaded the videos are used for annotation [26], see Figure 1.1.

A new video is being tagged by extracting the same features from a set of keyframes, which are then classified using the previously trained statistical models. For each feature a score is created, which are then combined to a single score for the video (see Figure 1.2). This score corresponds to the posterior $P(T|X)$ of the video X for each tag T, which is the probability that the video X belongs to a category T.

For each tag T a score is created. If a single tag is desired, the one with the highest score is chosen. This tag can then be used for different tasks, for example to support the traditional text search or to assist the user with tagging the video. Another possible application is the automatic tagging of the videos in a database.

1.4 Using Text Information for Tagging

So far, all extracted keyframes are used in the same manner. This means that TubeTagger does not make any difference between scene images (the "real" video data, captured with a camera) and images added subsequently by the user who published the video, which are referred to as "non-scene images" in the following.



(a) scene image

(b) non-scene image, containing the correct tag within the text

Figure 1.3: Sample keyframes extracted from a video of the category "cats". So far both keyframes are used in the same manner, the textual information contained in the non-scene image is not used. The goal is to extract this information and use it to improve the tagging of the video.

These non-scene images often contain some textual information about the origin or theme of the video (as seen in Fig. 1.3b), but this valuable information is currently unused. Another point is that these keyframes could even impair the results of the tagging, because their visual features are not related to the semantic context of the video.

In conclusion, two ways are proposed to improve the tagging system TubeTagger. The first one is to extract the textual information from the non-scene images, interpret it and take the results into account when it comes to create a score for a video. The second one is to withhold the non-scene images from both the training and classification process, so that they do not affect the tagging in a negative fashion.

Both issues can be addressed by detecting the keyframes containing non-scene text and processing them using a text extraction system to use this additional source of information.

Chapter 2

Method

2.1 Overview

In this thesis, a modification of the tagging system TubeTagger is proposed, consisting of two additional modules: a keyframe classification module to classify incoming keyframes, and a text extraction module for the extraction and interpretation of non-scene text (Fig. 2.1).

The frame classification module precedes the original tagging system. Each keyframe extracted from a video passes this module and is classified, either as a non-scene image or scene image. A keyframe classified as non-scene image is passed to the text extraction module, whereas any keyframe classified as scene image is forwarded to the tagging system and processed as usual.

The text extraction module extracts the text from the detected non-scene images and searches it for any useful information. This can be achieved by keyword spotting, which means that the extracted text is compared with the words in a dictionary of predefined tags. Finally, the text extraction module creates a score based on the results. This score is passed to the tagging system, which then combines it with the scores obtained from the other frames and creates a final vote for the video.

At lot of work has been done for extracting text from video images during the last years [7, 12, 13, 24, 27], but these methods aim at extracting text especially from scene images. In contrast, the method proposed here specifically aims at the user-added textual information in non-scene images.

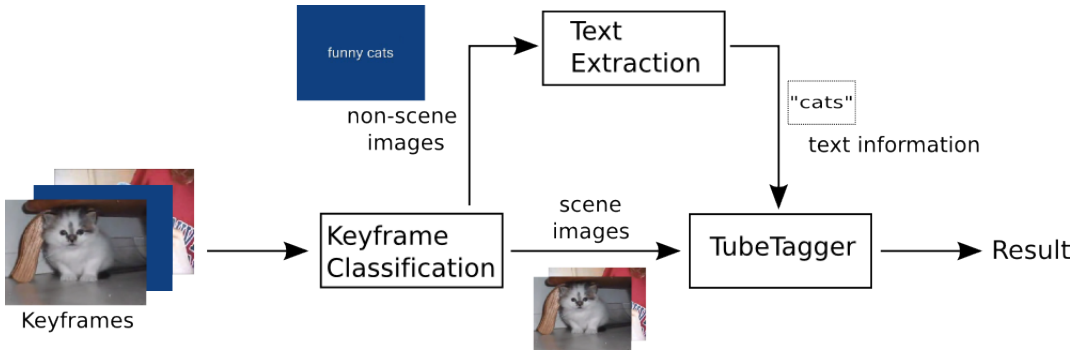


Figure 2.1: The proposed modification of the tagging system TubeTagger. The detected non-scene images are passed to the text extraction module, the scene images are processed as before. Finally, the scores from the text extraction module and TubeTagger are combined to a single score.

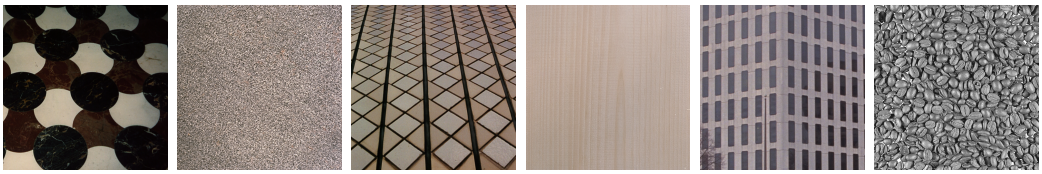
2.2 Detecting Non-Scene Images

The task of the keyframe classification module is to separate non-scene images from scene images. It consists of a classifier, which is trained in a supervised manner using a set of manually annotated keyframes. The annotation process and the data used are described in Section 3.1.

The performance of a classifier strongly depends on the features used for classification. Based on observations made on a small set of sample keyframes and some theoretical considerations, it was decided to use three kinds of visual features, which seem to be suited for separating non-scene images from scene images. Two of them, namely color histograms and Tamura texture features, have already been successfully used in TubeTagger. The third feature are gradient histograms, which are currently not utilized in the tagging process.

Also, two different types of classifiers are taken into consideration: a nearest-neighbor classifier and a classifier based on a simple thresholding over the entropy of the feature histograms. These two classifiers are tested together with the three different features mentioned above in a series of experiments, and the combination which provides the best results is used for the keyframe classification module.

In the following sections both the features and classifiers are introduced in more detail.



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

Figure 2.2: Examples for the first three Tamura features: high coarseness(a), low coarseness(b), high contrast(c), low contrast(d), high directionality(e), low directionality(f) [14, 8].

2.2.1 Feature Selection

Tamura Features

Tamura features [23] are a statistical texture representation, designed in accordance to the human perception of texture. They consist of six features, including coarseness, contrast, directionality, linelikeness, regularity, and roughness. However, experiments have shown that especially the first three features are important, thus they have been used in several image-retrieval systems like QBIC [10] and are used here as well. To obtain a histogram describing the texture, these three features are computed as follows [8]:

Coarseness
The coarseness is a measurement for the size of the texture elements in an image (see Fig. 2.2). The calculation takes several steps: First, for every point (x, y) the average $A_k(x, y)$ over a $2^k \times 2^k$ neighbourhood is computed:

$$A_k(x,y) = \frac{1}{2^{2k}} \sum_{i=x-2^{k-1}}^{x+2^{k-1}-1} \sum_{j=x-2^{k-1}}^{x+2^{k-1}-1} I(i,j)$$

where $I(i, j)$ is the image intensity at the point (i, j).

Then for every point (x, y) the differences between the non-overlapping averages on opposite sides of the point in both horizontal and vertical direction are calculated:

$$E_{k,h}(x,y) = |A_k(x + 2^{k-1}, y) - A_k(x - 2^{k-1}, y)|$$
$$E_{k,v}(x,y) = |A_k(x, y + 2^{k-1}) - A_k(x, y - 2^{k-1})|$$

For each point (x, y),

$$S(x, y) = \underset{k \in \{1..5\}}{\text{argmax}}\ max(E_{k,v}(x, y), E_{k,h}(x, y))$$

is computed, which results in the best size

$$S_{opt}(x, y) = 2^S$$

for each point (x, y).

Contrast
In an image, the contrast measures the dynamic range of grey levels and the polarisation of the distribution of black and white (see Fig. 2.2). The dynamic range is measured using the standard deviation of the grey levels, the polarisation using the fourth standardized moment, the so-called kurtosis:

$$F_{contrast} = \frac{\sigma}{\alpha_4^{1/4}} \qquad \text{where } \alpha_4 = \frac{\mu_4}{\sigma^4}$$

with σ being the standard deviation and μ_4 being the fourth moment about the mean of the grey values. To obtain pixel-per-pixel values, the contrast is computed over a 13×13 neighbourhood for each point (x, y).

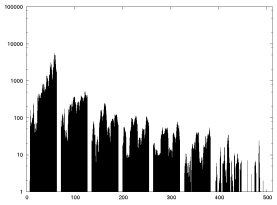
Directionality
The directionality is a measurement for the degree of orientation in the texture (see Fig. 2.2). To compute this feature, edges are detected in the image by a convolution with the Sobel operators G_h and G_v :

$$[\Delta_h(x, y), \Delta_v(x, y)] = [(G_h * I)(x, y), (G_v * I)(x, y)]$$

where

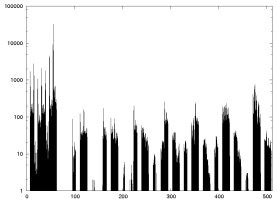
$$G_h = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \text{ and } \mathrm{G}_v = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(a) scene image



(b) Tamura histogram

(c) non-scene image



(d) Tamura histogram

Figure 2.3: Two sample keyframes with their Tamura features histograms.

Then the angle of the resulting gradient vector is computed for each point (x, y):

$$\theta(x,y) = arctan\left(\frac{\Delta_v(x,y)}{\Delta_h(x,y)}\right) + \frac{\pi}{2}$$

Now that for each pixel (x, y) a total of three values is available ($S_{opt}(x, y)$, $F_{contrast}(x, y)$ and $\theta(x, y)$), a three-channel image can be constructed (the so called tamura image) and a histogram is created:

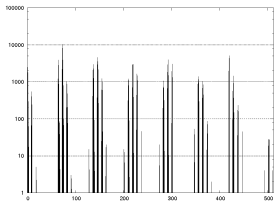
$$h_{tamura}(s, c, d) = N \cdot p(S_{opt} = s, F_{contrast} = c, \theta = d)$$

where N is the total number of pixels in the image. The Tamura features histograms used in this thesis are computed using the FIRE Image Retrieval Engine[1].

Taking into consideration that most non-scene images only consist of a plain unicolor background with some text in it, these three features seem to be well-suited for the task of discriminating the two classes. The large unicolor areas in a non-scene image result in a very high value for the coarseness $S_{opt}(x, y)$. For the same reason, the contrast $F_{contrast}(x, y)$ and the angle $\theta(x, y)$ are zero for most pixels, with the text areas being the only exceptions. This difference is reflected in the Tamura feature histograms (see Fig. 2.3).

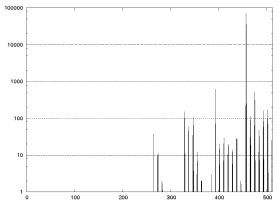
[1] http://www-i6.informatik.rwth-aachen.de/~deselaers/fire.html

(a) scene image



(b) color histogram

(c) non-scene image



(d) color histogram

Figure 2.4: Two sample keyframes with their color histograms. The histogram of the non-scene image is much more peaked than the histogram of the scene image.

Color Histogram

A color histogram is an effective and simple representation of the color content of an image. Every pixel in an image can be described by the three components in its color space (in this thesis red, green and blue components in RGB space are used). The color histogram denotes the joint distribution of the intensities of these three color channels and is defined as

$$h_{RGB}(r, g, b) = N \cdot p(R = r, G = g, B = b)$$

with R, G and B being the three color channels and N the total number of pixels in the image. Again, the FIRE Image Retrieval Engine is used to create these histograms.

Color histograms seem to be promising for discriminating non-scene images from scene images, because most of the non-scene images only contain very few different colors, with one color for the background being dominant. This can clearly be recognised in a color histogram (Fig. 2.4d), where one distinctive peak overtops the others by far. Scene images on the other hand tend to have a much wider range of colors, their color histograms are less peaked than the color histograms of non-scene images (Fig. 2.4b).

For the experiments in Section 3.2, color histograms with two different bin sizes are used. A smaller histogram consisting of a total of $8^3 = 512$ bins and a much larger one consisting of $24^3 = 13824$ bins.

Gradient Histogram

The gradient of an image measures the local change of the intensities, both in their magnitude and direction. Whereas the direction is used for the Tamura feature "directionality", this time the magnitude is computed for each pixel (x, y). Again, a convolution is performed, but this time two simpler masks are used:

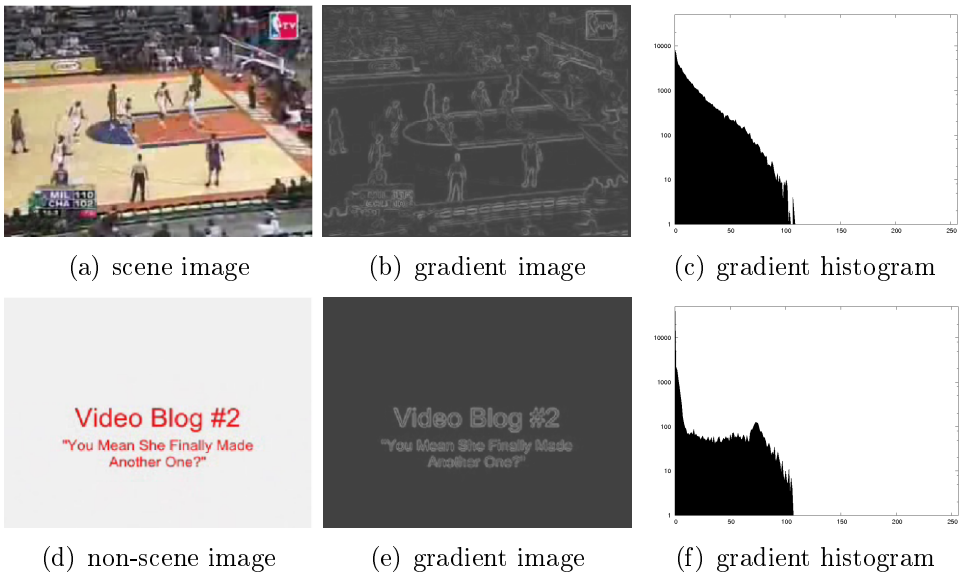
$$G_h = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \text{ and } \mathrm{G}_v = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Now we compute the magnitude g for each point (x, y)

$$g(x, y) := \sqrt{(\Delta_h(x, y))^2 + (\Delta_v(x, y))^2}$$

and create a histogram as a feature. In this thesis, a histogram with 256 bins is used.

The difference between non-scene images and scene images can be seen in the way the strength of the intensity changes over the image, which is measured by the magnitude of the gradient. Given that a non-scene image consists of few different intensities with the intensity of the background color being very dominant, most of the gradient image shows a magnitude of zero. Only at the transitions to the characters a change of the intensity is observable (Fig. 2.5e). In contrast, a scene image features changes in the intensity distributed over the whole image (Fig. 2.5b). This results in a smooth descent of the histogram (Fig. 2.5c), in contrast to the abrupt drop in the histogram of a non-scene image (Fig. 2.5f).



(a) scene image (b) gradient image (c) gradient histogram

(d) non-scene image (e) gradient image (f) gradient histogram

Figure 2.5: Gradient histogram examples. The scene image (a) shows intensity changes distributed over the whole image (b), whereas in the non-scene image (d) the changes are concentrated in the region around the text (e). The histograms (c,f) differ accordingly.

2.2.2 Classifier

Nearest Neighbor

The nearest neighbor (NN) rule is a commonly used method in pattern classification [6], providing satisfying results in many cases. Its setup is quite simple: By comparing a querying sample to a set of training samples (called prototypes) using a distance measurement, the training sample with the smallest distance - the so-called nearest neighbor - is found. It is then assumed that the test sample is of the same class as the found prototype.

When implemented in a naive way, NN has a time complexity of $O(dn)$, as every sample has to be compared in d dimension to all n prototypes used for training. To bypass this, approximate nearest neighbor search can be used to reduce the time complexity at negligible loss in terms of precision/error rates. The approximate nearest neighbor method used here makes use of kd-trees - a generalization of the binary search tree for the use in higher dimensions introduced by Bentley [2] - and has been successfully applied to visual matching before [18].

The kd-tree works as a space-partitioning data structure, organizing points in a k-dimensional space by hierarchical subdivision using splitting hyperplanes parallel to the coordinate axes. Each node of the tree is associated with a k-dimensional

rectangle and the points that lie within, with the root being the bounding rectangle and thus containing all points stored in the tree.

The construction of a kd-tree starts with the first cell, the root node. It is split into two parts using a hyperplane parallel to one of the coordinate axes and passing through the cell, with each part becoming a new node (child) of the tree. Every new node is being split as described above, until the number of points associated with a new node falls below a certain threshold (e.g. one). Such a node becomes a leaf and is not partitioned further.

Nearest neighbor search on a kd-tree [11] is performed by a simple recursive algorithm. The tree is searched depth-first, deciding at each node on which side of the corresponding hyperplane the query point lies. When reaching a leaf, the distances from the query point to each of the points in the leaf's cell are calculated and the smallest distance is saved. Then each of the previously visited parent nodes is examined to determine if it is possible that the respective other child contains a point closer to the query point than the point found so far. When returning from the root, the closest point found is returned.

An approximate nearest neighbour search can be performed by introducing an error bound ε and allowing the algorithm to only visit a further child if its distance is lower than the currently smallest distance found divided by $(1 + \varepsilon)$ [1].

Entropy thresholding

For the second classification method the well-known entropy measure is used. The entropy was first introduced by Shannon as a statistical feature measuring the amount of information of a system. For a discrete distribution over events $X = x_1, ..., x_n$ it is computed as follows:

$$H(X) = \sum_{i=1}^{n} P(x_i) \cdot log\, P(x_i)$$

where $P(x_i)$ is the probability of the i-th event. In case of a histogram referring to a probabilistic distribution $P(x_i)$ it is the normalized value in the i-th bin of the histogram.

A sharply peaked distribution results in a very low entropy value, whereas a distribution spread evenly over a large amount of bins has a much higher entropy. As seen in Section 2.2.1, the feature histograms of non-scene images tend to be much more peaked than the feature histograms of scene images, thus it is expected that their entropy values also differ by a large amount.

Thus, the classification problem is reduced to the problem of finding a threshold which seperates the two classes by the entropy values $H(X)$ of their feature histograms X:

$$class(X) = \begin{cases} \text{scene image} & if\, H(X) \geq T \\ \text{non-scene image} & if\, H(X) < T \end{cases}$$

The threshold T is proposed to separate the two classes such that the difference between the error rates of non-scene images and scene images is minimized over a training set, thus obtaining an equal error rate for both classes:

$$T = \underset{t}{\text{argmin}} \mid e_{train,scene}(t) - e_{train,non-scene}(t) \mid$$

where $e_{scene,text}$ is the quotient of wrongly classified scene images to the total number of non-scene images of the training set, and $e_{train,non-scene}$ is the same for non-scene images.

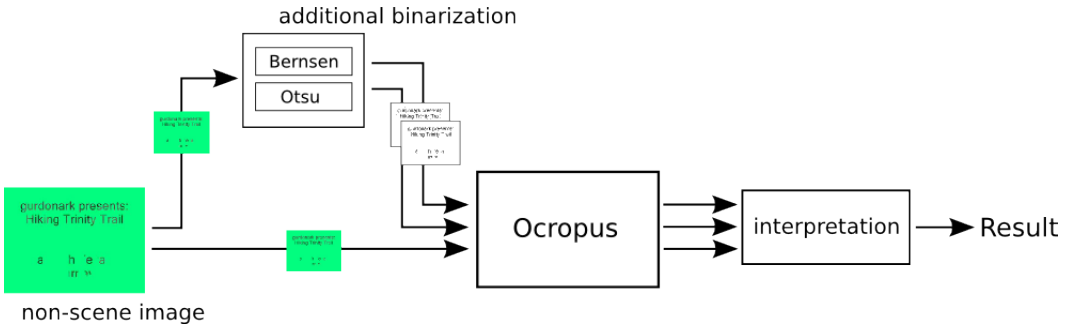


Figure 2.6: The text extraction module. Incoming keyframes are first binarized by the two additional binarization methods, then forwarded to the OCR system OCRopus. The extracted text is interpreted and a score is produced for the use in the tagging process.

2.3 Text Extraction

After the keyframes have been classified, the detected non-scene images are forwarded to the text extraction module. This consists of three parts: the Optical Character Recognition (OCR) system which extracts the text from the images, an additional binarization module, and an interpretation module which searches the found text for usable information, in our case video tags.

2.3.1 OCRopus

The OCR system used to extract the text is OCRopus[2]. OCRopus is an open source OCR project, mainly being developed for book scanning applications like the Google Book Search[3] and for desktop and office use. It combines several subsystems like preprocessing, layout analysis and statistical language modeling in a highly modular design [5]. The character recognition engine used by OCRopus is Tesseract[4].

OCRopus can be used in several ways, including the Lua scripting language[5] and a command-line tool. For the experiments in this thesis, the command-line tool is used.

[2]http://code.google.com/p/ocropus/
[3]http://books.google.com/
[4]http://code.google.com/p/tesseract-ocr/
[5]http://www.lua.org/

2.3.2 Preprocessing

Of all possible preprocessing steps the binarization - which means converting an image into a binary (black/white) representation - plays a key role when it comes to character recognition in color images. Given the fact that most non-scene images only feature very few different colors, the binarization of non-scene images is performed on the corresponding grey-scale images. But because many different character sizes can be found in non-scene images, it is necessary to take a closer look at binarization methods to achieve good recognition results.

Sauvola's Method

OCRopus features a local thresholding method by Sauvola[20] as its built-in binarization method. Thereby, "local" thresholding means that the threshold separating the background from the foreground is computed separately for each pixel. For Sauvola's method it is computed by

$$T(x,y) = \mu(x,y) \cdot \left[1 + k \cdot \left(\frac{\sigma(x,y)}{R} - 1\right)\right]$$

where $\mu(x,y)$ is the mean and $\sigma(x,y)$ the standard deviation of pixel values in a $w \times w$ neighbourhood around the pixel (x,y). R is the maximum value of the standard deviation and thus set to 128 for a grey-scale document, and k is a parameter set a priori, usually to a value between 0.2 to 0.5.

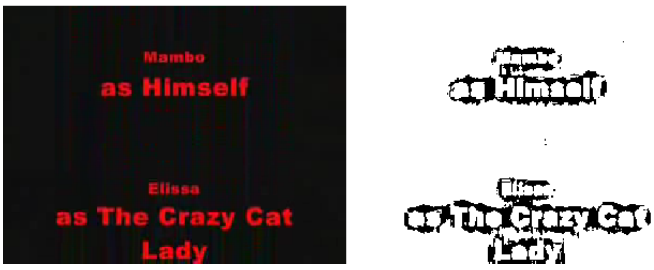
OCRopus uses a fast implementation of this method that utilizes integral images to compute the local means and standard deviations. In comparison to the original method introduced by Sauvola its running time is not dependent on the window size w [22].

Sauvola's binarization method delivers good results in combination with document images [21, 22], but often fails when applied to video keyframes. The reason for this lies in the way the method calculates the threshold. In unicolor areas where the standard derivation is 0, the threshold computes to

$$T(x,y) = \mu(x,y) \cdot (1-k) = \mu(x,y) - k \cdot \mu(x,y)$$

so with $k \in [0.2; 0.5]$ the threshold is always lower than the current intensity level.

That means that all unicolor areas are whitend, with the areas around the characters (depending on the window size) being the only exception. Consequently, this binarization method cannot be used solely for all keyframes, because many



(a) Original image (b) Binary image produced by Sauvola's method

Figure 2.7: Sauvola's method whitens unicolor areas, thus its application is limited to images with a bright background color.

non-scene images feature a darker color for the background than used for the text, which renders the method useless (see Fig.2.7).

To get a large proportion of the text extracted, another binarization technique is needed. However, first experiments showed that it is very difficult to find a binarization method which suites all non-scene images, given their large spread of different character sizes and the artifacts resulting from the video compression.

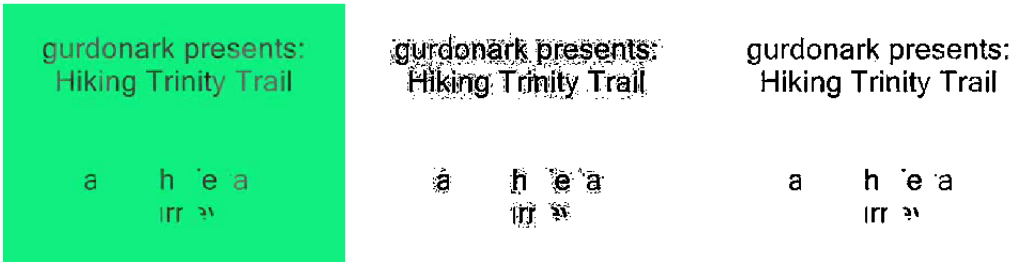
To avoid this, two different binarization techniques (one global and one local method) are used in addition and are applied simultaneously to all non-scene images, together with the built-in Sauvola method. So when it comes to character recognition, up to three differently binarized images are available. Also, up to three different text outputs are available for the interpretation module, increasing the chance of recognising a tag within the text of a non-scene image.

Otsu's Method

The first additional binarization technique used is Otsu's method [17], which tries to find an optimal global threshold by minimizing the weighted sum of the within-class variance from the foreground and background pixels and at the same time maximizing the between-class scatter. The global threshold computes as follows [21]:

$$T_{otsu} = \underset{T \in \{0..255\}}{\operatorname{argmax}} \frac{P(T)(1-P(T))(\mu_f(T)-\mu_b(T))^2}{P(T)\sigma_f^2(T)+(1-P(T))\sigma_b^2(T)}$$

where $\mu_f(T)$ and $\mu_b(T)$ are the means of the foreground and background region for a certain threshold T, with their corresponding standard deviations $\sigma_f(T)$ and $\sigma_b(T)$.



(a) Original image (b) Binarized by Otsu's method (c) Binarized by Otsu's method with the modified threshold

Figure 2.8: The threshold found by Otsu's method is not optimal, resulting in visible artifacts around the characters. A simple modification that moves the threshold towards the assumed background color eliminates these artifacts.

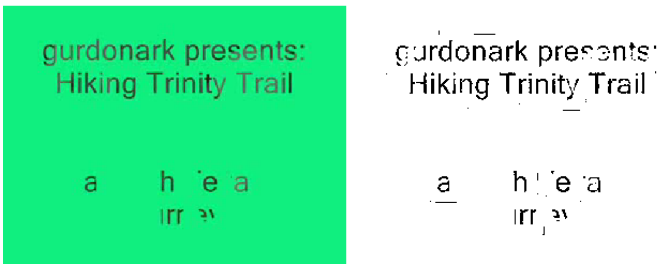
Otsu's method relies on an almost equal number of pixels in each of the two classes to deliver good results [21], a fact that is not given in most of the non-scene images. As to be seen in Figure 2.8b, the threshold found is not optimal, resulting in visible artifacts around the characters. These compression artifacts are a byproduct of the lossy compression of the video data and its low resolution.

To remove these artifacts and thus to improve the overall results, the threshold found by Otsu's method is modified as follows:

$$T_{new} = T_{otsu} - \frac{C}{|b - T_{otsu}|}$$

where T_{otsu} is the threshold found by Otsu's method and C is a predefined parameter. b is set to the intensity level in the corners of the image, based on the assumption that the background color of a non-scene image also applies to its corners. So the threshold is being moved towards the assumed background color, in relation to its difference from the old threshold and the parameter C. The larger the difference between the background color and the found threshold is, the more the threshold is adapted.

Several test runs have shown that this small modification eliminates the artifacts around the text region and therefore makes most of the non-scene text recognizable by OCRopus, as shown in Figure 2.8c. For C, a value of 30 has been set manually.



(a) Original image (b) Binarized by Bernsen's method

Figure 2.9: Example for a non-scene image binarized by Bernsen's method.

Bernsen's method

Bernsen's binarization [3] in contrast to Otsu's method is another local thresholding method, where the threshold for each pixel is computed separately, according to the mean of the minimum intesities $I_{min}(x, y)$ and maximum intensities $I_{min}(x, y)$ in a $w \times w$ region around the pixel (x, y):

$$T(x, y) = \frac{I_{max}(x, y) + I_{min}(x, y)}{2}$$

Additionally, if the contrast

$$C(x, y) = I_{max}(x, y) - I_{min}(x, y)$$

in the surrounding area of (x, y) falls below a certain threshold C_{min} , the treshold $T(x, y)$ is overwritten by a global value T_{global}. T_{global} is either set a priori or can be computed by a simple global thresholding method, like using the mean of the highest and lowest intensity over the whole image.

For the non-scene images, a contrast limit $C_{min} = 30$ and a window size $w = 20$ are used. An example can be seen in Figure 2.9.

2.3.3 Keyword Spotting

After the text is extracted from non-scene images, it is interpreted for the use in the tagging system. Therefore, the results of OCRopus are compared to a dictionary containing all possible tags for the videos ("keyword spotting").

The comparison is word-by-word, whereas the edit distance $d(x, y)$ between a candidate x and each word y in the dictionary is measured using the algorithm of Levenshtein [16]. It equals the minimum number of deletions, insertions or

substitutions of single characters necessary to transform one string into another. To make matches robust to errors in character recognition, for example the small letter "l" recognised as the number "1", a certain edit distance D_{max} is allowed for detecting matches:

$$match(x,y) = \begin{cases} true & \text{if d(x,y)} < \text{D}_{max}(y) \\ false & \text{if d(x,y)} \geq D_{max}(y) \end{cases}$$

For the experiments, the following value for D_{max} is used:

$$D_{max}(y) = \lfloor wordlength(y)/4 \rfloor$$

So the allowed edit distance grows by one every four characters of y, allowing more errors the larger the tag is.

The result are the names and quantities of the tags found within the keyframes of a video. To integrate this information with the results of the tagging system, for each tag T and video X the corresponding probabilities $P_{text}(T|X)$ are created. They are computed by putting the count of a tag T in relation to the total number of tags found in the video:

$$P_{text}(T|X) = \begin{cases} n/N & \text{if tag } T \text{ was found } n \text{ times in a total of } N \text{ tags found} \\ 0 & \text{if tag } T \text{ was not found} \end{cases}$$

To create a final vote in combination with the results of TubeTagger, the final score $P_{final}(T|X)$ is set to

$$P_{final}(T|X) = P_{text}(T|X) \cdot 1 + (1 - \sum_{T} P_{text}(T|X)) \cdot P_{tagging}(T|X)$$

with $P_{tagging}(T|X)$ being the vote of the tagging system. So if any tags have been found in the keyframes of a video, the voting is according to these tags and not according to the results of the tagging by visual features. This approach is chosen because it is very likely that a video with the tag T included into a non-scene image actually belongs to this category.

Chapter 3

Experiments

3.1 Dataset

The foundation of the following experiments is a set of several thousand automatically downloaded YouTube videos spread over 23 categories, namely: basketball, beach, building, cats, concert, crash, dancing, desert, eiffeltower, explosion, golf, helicopter, hiking, interview, race, riot, sailing, secondlife, soccer, swimming, talkshow, tank and videoblog. From these videos, about 97,430 representative keyframes have previously been extracted, using an adaptive clustering approach [4].

Frame classifier

For the classification experiments, two sets containing 4538 and 4539 keyframes have been chosen randomly. To be able to use them for training and test purposes they have been annotated manually. To accelerate this process, a tool has been developed which allows a quick manual detection and annotation of the non-scene images (see Fig. 3.2).

The statistics of these two sets are the following:

	set CL:A	set CL:B	both
non-scene images	114	118	232
scene images	4425	4420	8845
images total	4539	4538	9077

As can be seen, the percentage of non-scene images is quite low with about $2,61\%$, a fact that should be kept in mind when it comes to interpreting the results. Another interesting observation during the annotation was that the percentage of

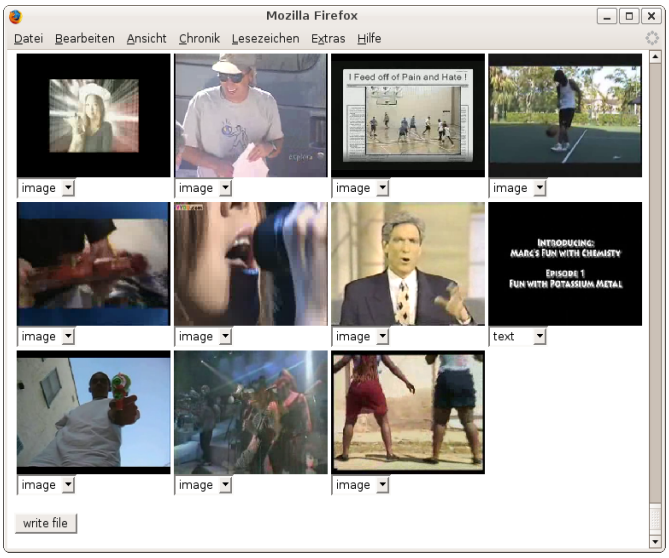


Figure 3.1: A screenshot of the annotation tool.

non-scene images from the extracted keyframes varies depending of the category of the video (see Fig.3.2). The most non-scene images were found in keyframes of the category soccer, whereas in the category eiffeltower there are none at all. Obviously, improvements through the use of textual information do not apply to all categories equally.

Text Extraction

To test the text extraction module, the non-scene images of both sets CL:A and CL:B have been merged and their ground truth notated where possible. Non-scene images containing non-western fonts or overlapping characters due to sliding effects were disposed. The resulting set is the following:

	set OCR
non-scene images	171
ground truth	6656 characters

TubeTagger

Finally, for the practical tests with the whole tagging system, two sets each containing of 1100 videos were used (50 per category, whereas the category building was not included in these already existing sets). To be able to evaluate the results, the test set contained the full ground truth.

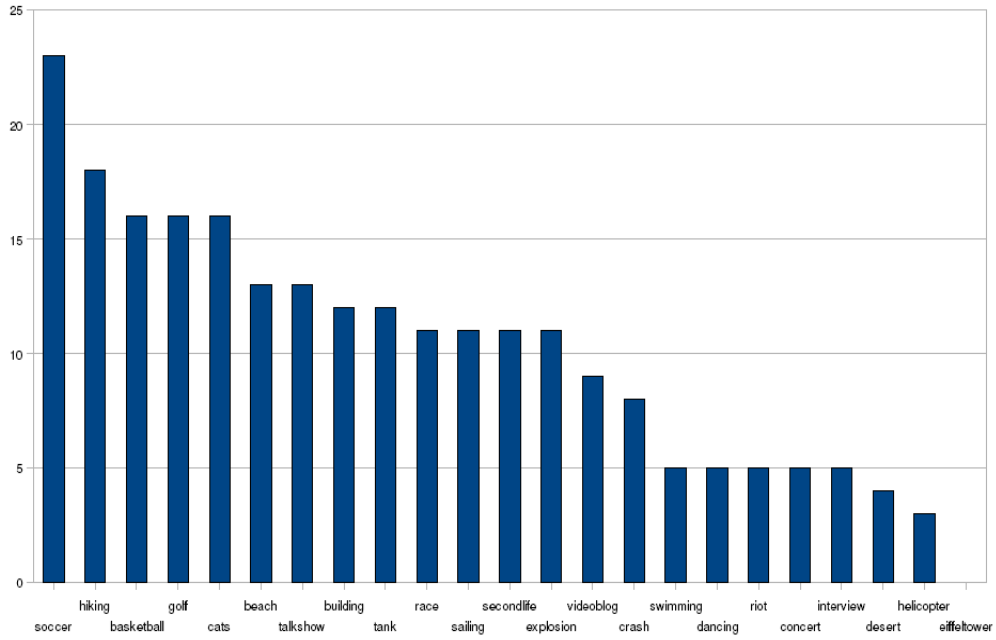


Figure 3.2: Breakdown of the 232 manually annotated non-scene images. They are not distributed evenly over all 23 categories, instead some categories contain much more non-scene images than others.

3.2 Experiments

3.2.1 Frame Classification

The first series of experiments is used to determine the best combination of classifier and feature. Therefore both classifiers - the nearest-neighbor and the entropy thresholding method (for both see Section 2.2.2) - are tested with the three proposed visual features in Section 2.2.1 .

Experiment 1 - NN Classification, unbalanced:

In the first experiment, the approximate nearest-neighbor classification is performed on a kd-tree, using the previously introduced Tamura features, color histogram and gradient histogram. The two sets CL:A and CL:B are used in a 2-fold cross-validation process, which means that every set is used once as the training set to create the tree and once as the test set. The resulting errors are averaged.

	test error %	
feature	non-scene image	scene image
Tamura features	34.23	0.43
color histogram	28.82	0.34
color histogram 24^3	25.43	0.28
gradient histogram	36.68	0.47

The first thing to be seen is that the error rates for non-scene images and scene images differ by a large amount. Taking into consideration that there are much more scene images than non-scene images (the annotated samples showed a 38 to 1 ratio), this implies that a lot of scene images must have similar visual features in comparison to the non-scene images we want to separate.

Besides this, the gradient histogram performs worst. The Tamura features perform a little better, but the best feature for this classifier are the color histograms. The larger color histogram even performs a little better than the smaller one, but at the cost of a significantly higher computational complexity.

Experiment 2 - NN Classification, balanced:

The goal for the second experiment is to balance the error rates for non-scene images and scene images. Therefore, a k-nearest-neighbor search is performed with increasing values for k, whereas a sample is classified as non-scene image if at least one of the k nearest neighbors is a non-scene image. This procedure corresponds to the introduction of a loss function and the adjustment of the costs until an equal error rate is achieved. Again, the sets CL:A and CL:B are used.

feature	k	test error%
Tamura features	65	15.39
color histogram	100	9.19
color histogram 24^3	300	9.11
gradient histogram	20	7.25

An equal error rate was achieved with different values for k, depending on the feature used. This time the gradient histogram delivers the best result with an error rate of 7.25%, whereas the Tamura features now take the last position. Both color histograms have similar error rates.

Experiment 3 - Entropy Thresholding:

In the next experiment, the classifier based on the entropy thresholding method is used. Again, the two sets CL:A and CL:B participate in a 2-fold cross-validation.

feature	training error %	test error %
Tamura features	15.09	15.27
color histogram	7.41	6.89
color histogram 24^3	6.12	5.89
gradient histogram	10.94	10.78

Given the fact that the threshold is found by minimizing the difference between the error rates for non-scene and scene images, the error rates do not diverge as in the case of the one-nearest-neighbor approach, so no adjustment like in Experiment 2 is needed to achieve an equal error rate.

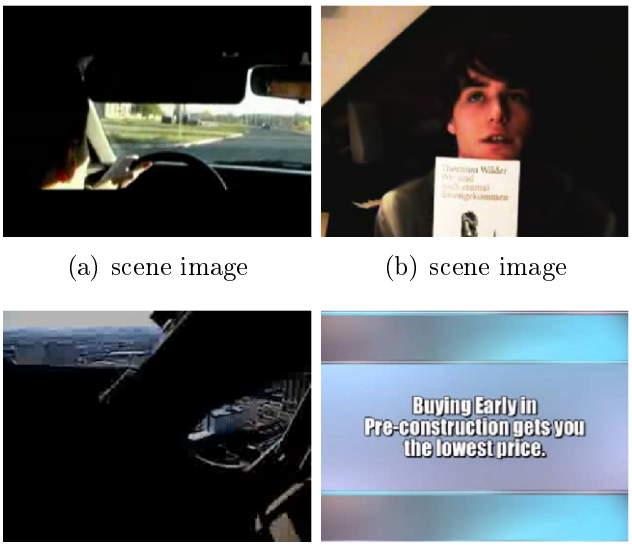
The color histograms perform better than with the nearest-neighbor approach of Experiment 2, but the results using the gradient histogram are worse. Again, the Tamura features have the highest error rates with about 15% and therefore do not meet the expectations.

Experiment 4 - Entropy Thresholding, modified gradient histogram:

An in-depth analysis of the gradient histograms revealed that the main difference between non-scene images and scene images lies within the first 50 to 70 bins. In this experiment, the histogram is cut at specific bins to see if the results improve when ignoring the higher bins. The classifier and sets used are the same as in Experiment 3.

cut at	training error %	test error %
no cut	10.94	10.78
50	8.55	8.17
25	6.96	6.74
10	7.22	7.28

Cutting the histogram improves the error rates, with the best result using a cut after the 25. bin. At this point, the error rate went down by about 4 % in comparison to the original histogram. Cutting the histogram further results in an increasing error rate.



(a) scene image (b) scene image

(c) scene image (d) non-scene image

Figure 3.3: Four examples of keyframes that are classified wrong with all three features.

So the best error rates could be achieved with the entropy thresholding used on the large color histogram, with a test error of 5.89 %. But the calculation and processing of such a large histogram is costly in terms of time and space complexity, which is why either the small color histogram or the cut gradient histogram should be used for the frame classification module. Both their error rates are only about 1 % higher than the error rates of the large color histogram. Compared to NN classification, the entropy thresholding method is preferable because the achieved equal error rates are lower.

So in a practical test an error rate of approximately 7 % is expected, which means that still a lot of scene images are classified as non-scene images. Taking a closer look at the wrong classified samples reveals that many scene images are misclassified by all three features. As can be seen in Figure 3.3, many of them have a lot in common with non-scene images. High contrast regions, large unicolor areas and few different colors are the main points based on which the three features distinguish the scene images from the non-scene images. This is why none of the three features is sufficient for the use with this kind of scene images. Also, some non-scene images are classified incorrectly. The non-scene image in Figure 3.3d for example features a complex multicolor background, a feature that is associated with scene images.

There are also some images that are misclassified by one feature, but classified correctly with another. For example the scene image in Figure 3.4: it features several different colors, therefore it is classified right if the color histogram is

Figure 3.4: Example a for non-scene image where one feature works but the other does not. This scene image is classified correctly by its color histogram, but incorrectly by its gradient histogram.

used. On the other hand, it features a high contrast around the person, thus by using the gradient histogram it is classified as non-scene image.

3.2.2 Text Extraction

The next series of experiments is carried out to determine the detection rates of the text extraction module. All components are explained in Section 2.3.

Experiment 5 - Text Extraction, simulation:

This experiment is a simulation of the keyword spotting method used for finding tags. Each keyframe of the OCR set is been binarized by the three binarization methods stated before, then the text is extracted by OCRopus.

For each result the edit distance to the ground truth is calculated. To state that in case of some non-scene images one binarization method might perform better than another one, multiple binarization results are combined by choosing the result with the smallest edit distance for each keyframe.

binarization by	edit distance	% of ground truth
Sauvola	3381	50.80
Bernsen	3290	49.43
Otsu	3502	52.63
Sauvola+Bernsen	2571	38.63
Sauvola+Otsu	2370	35.61
Bernsen+Otsu	2334	35.07
Sauvola+Bernsen+Otsu	2100	31.55

The three different binarization methods perform almost equally if used separately. Regarding the observed problems Sauvola's method has with a lot of the

keyframes and the high error rates, it seems that the other binarization methods also have problems with some keyframes. This means that none of the three binarization methods alone is sufficient for this task.

Combining different binarization methods decreases the edit distance, which means that keyframes where one method fails are often better binarized by another method. The overall improvement shows that the combination of different binarization methods is suitable for keyword spotting in non-scene images.

Experiment 6 - Text Extraction, keyword spotting

In the next experiment the keyword spotting with a small allowed edit distance is performed (as described in Section 2.3.3), and the OCRopus results are compared to a dictionary containing the 23 possible tags. Like in the simulation run, in case of multiple binarization results the one with the lowest edit distance is chosen. In this test set a maximum of 17 tags could be found.

type	tags found	correct	recall	precision
sauvola	9	9	52.94	100.00
bernsen	8	8	47.06	100.00
otsu	10	9	52.94	90.00
sauvola+bernsen	11	11	64.71	100.00
sauvola+otsu	13	12	70.59	92.23
bernsen+otsu	13	12	70.59	92.23
sauvola+bernsen+otsu	14	13	76.47	92.86

As expected and seen in the simulation, the combination of multiple binarization results delivers a higher recall rate for the tags. A combination of three different binarization methods makes it possible to detect more than 75% of the tags contained in the keyframes of the OCR set. Given a rather small allowed edit distance, the precision is always greater or equal than 90%.

3.2.3 Modified Tagging System

The experiments so far analyzed the two modules developed in this thesis - keyframe classification and text extraction - separately, to find an appropriate classification method and to evaluate the performance of the keyword spotting.

In the following experiments, the influence of these modules on the tagging sys-

tem is tested as described before to get an impression of how well this method works in practice. The keyframe classification module uses the entropy thresholding method as classifier, together with the color histogram as features. This combination delivers decent error rates with about 6-7% on the test set and is fast in terms of computing time. The text extraction module uses all three binarization methods, and keyword spotting is performed on the dictionary containing 22 tags (the category building is disposed).

Experiment 7 - TubeTagger, with keyframe classification module:

In this experiment, only the keyframe classification module (KC) is active. Incoming keyframes are classified and only keyframes classified as scene image are processed by the tagging system. The training and test set each consisting of 1100 videos are used, the result is the mean average precision (MAP) of the tagging system.

To compare the results, the same experiment is done without the keyframe classificator being active.

feature combination	w/o KC (MAP %)	with KC (MAP %)	$\Delta MAP\%$
color	23.2562	22.5729	-0.6833
texture	20.2136	19.7838	-0.4298
coloretexture	25.7652	24.5116	-1.2536
colortexturemotion-ef	27.7458	27.6787	-0.0671

The filtering of these keyframes decreases the mean average precision of the tagging system. The change is not radical, but still measurable. The main reason for this lies in the ratio between non-scene images and scene images, in the sets CL:A and CL:B it was about 1:38. To many scene images get sorted out compared to non-scene images.

Experiment 8 - TubeTagger, with text extraction module:

Now the results of the text extraction module are included in the tagging process. All keyframes classified as non-scene images are searched for tags. First, the scores of the text extraction are compared to the scores of the tagging system, to get an impression of how many videos are affected to which degree.

results when tagging by text	#	% of test set
correctly tagged (better tag)	24	2.18
correctly tagged (same tag)	15	1.36
correctly tagged	39	3.55
wrongly tagged (same tag)	4	0.36
wrongly tagged (worse tag)	1	0.09
wrongly tagged	5	0.45
videos tagged by OCR	44	4.00

The share of videos containing their tag in one of the non-scene images is quite low, only 4% of the total number of videos. But more than 88 % of the found tags contribute to the correct tagging of the video, and only one video is given a worse tagging than by the tagging system.

The scores are now combined with the scores of the tagging system. Note that all keyframes classified as non-scene image are passed to the tagging system, in addition to proceed them to the text extraction module. This is done to achieve the best possible results (see Experiment 7).

As far as features are concerned, color, texture and motion are combined in an early fusion.

feautre combination	w/o OCR (MAP %)	with OCR (MAP %)	$\Delta MAP\%$
colortexturemotion-ef	27.7458	30.9553	3.2095

Using the found tags in the tagging process improves the mean average precision by more than 3 %, an improvement similar to those achieved by using one additional visual feature for the tagging system. Considering the fact that only about 4 % of all videos are affected, this is a remarkable improvement.

Chapter 4

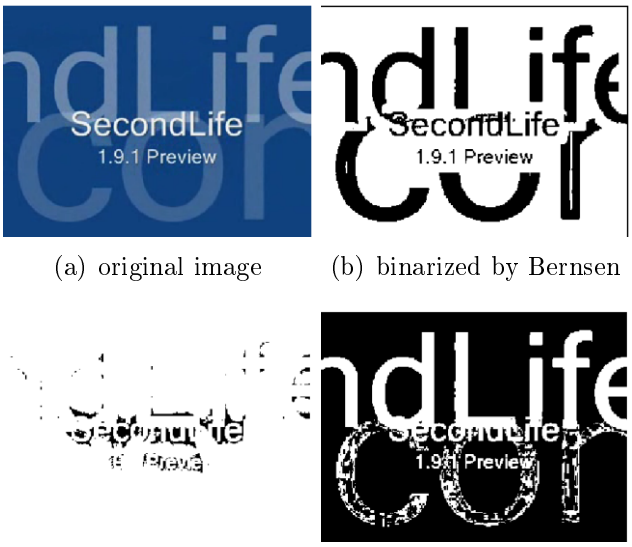
Discussion

The experimental results have shown that non-scene images can be separated from scene images with a decent error rate, using a simple classifier based on visual features. With color histograms or gradient histograms as features and the entropy thresholding method as classifier, an overall error rate of below 7 % could be archieved. This means that about 93 % of all non-scene text images are detected and forwarded to the text extraction module.

But given the 1:38 ratio of non-scene images to scene images observed during the annotation, a lot of scene images are incorrectly classified as non-scene images. This has a minor effect on the results of the text extraction module, because in scene images usually no character boxes are found by OCRopus and thus no text is extracted. And in case that some wrong text is found, a limited edit distance to the words in the dictionary is also needed for a match. It does produce additional load for the text extraction module, but compared to the time TubeTagger needs to tag a video it is negligible.

For the same reason, many scene images are disposed because they are misclassified. Assuming that the ratio from the annotated keyframes applies to all keyframes, from 1000 keyframes 974 would be scene images and 26 non-scene images. Given a 7% error rate on both classes, 24 of 26 non-scene images would be forwarded to the text extraction module and withhold from TubeTagger. But due to the error, about 68 scene images would be disposed in addition. This results in a decrease of the mean average precision.

The error rate of the keyframe classification module can surely be lowered by using more complex classifiers, but the main problem lies within the features. Based on these features, many scene images show similarities to non-scene images, like few different colors or large unicolor areas (see Figure 3.3). Therefore, more appropriate features are needed to reach better detection rates. Assuming that each scene image contributes as much information to the tagging system as a



(a) original image (b) binarized by Bernsen

(c) binarized by Sauvola (d) binarized by Otsu

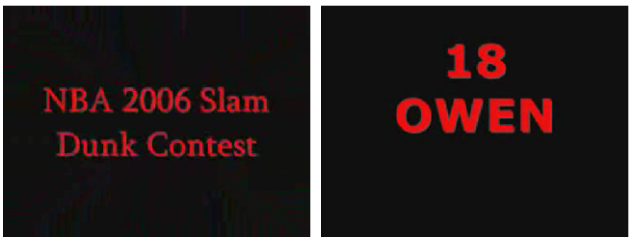
Figure 4.1: Some non-scene images are not binarized correctly with any of the three methods. Especially if they feature complex, multicolor text.

non-scene images contributes false information, we would need a combination of feature and classifier with an equal error rate as low as the percentage of non-scene text, which lies around 2-3%.

On the other hand, processing the detected non-scene text images to the text extraction module shows good results. With the simple approach of comparing the OCRopus results with all possible tags ("keyword spotting"), a significant improvement of the tagging system could be achieved, even if the percentage of videos containing their tag in text form is quite low with about 4 %. But using this information as proposed improves the mean average precision by 3%.

Some improvements of the text extraction module could be achieved by using better binarization techniques. Against the original assumptions, the binarization on gray-scale images is not sufficient for all non-scene images. Especially if the text in a non-scene images features different colors (see Figure 4.1), the binarization methods used in this thesis do not work. A color-based approach could help to improve the results on these non-scene images.

Another way to improve the results would be to extend the dictionary. It actually only contains the names of the 23 categories, but many videos contained other words with a strong semantic relation to these categories. For example, many sport videos contain the names of famous soccer players or other theme-related words (see Figure 4.2). A much larger dictionary with semantic associations could help here, as suggested by Zhuang et al. [29].



(a) non-scene image related to basketball

(b) non-scene image related to soccer

Figure 4.2: Both non-scene images contain text with a strong semantic link to their category. The shortcut "NBA" refers to the american basketball league, whereas "Owen" is the name of a famous soccer player.

As a conclusion, it is therefore proposed to include the keyframe classification and the text extraction module in TubeTagger, only without removing the detected non-scene text images from the tagging process unless better classification rates can be achieved. Therefore, a more appropriate combination of classifier and feature has to been found.

Bibliography

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Wu, *"An Optimal Algorithm for Approximate Nearest Neighbor Searching"*. Proc. 5th ACM-SIAM Sympos. Discrete Algorithms, pp. 573-582, 1994.

[2] J. L. Bentley, *"Multidimensional binary search trees used for associative searching"*. Communications of the ACM, 18(9) pp. 509-517, 1975.

[3] J. Bernsen, *"Dynamic Thresholding Of Gray Level Images"*. ICPR'86: Proc. Intl. Conf. Patt. Recog., pp. 1251-1255, 1986.

[4] D. Borth, A. Ulges, C. Schulze, T. Breuel, *"Keyframe Extraction for Video Tagging & Summarization"*. GI-Informatiktage 2008, Bonn, Germany.

[5] T. Breuel, *"The OCRopus Open Source OCR System"*. Proceedings IST/SPIE 20th Annual Symposium, 2008.

[6] T. M. Cover and P. E. Hart, *"Nearest Neighbor Pattern Classification"*. IEEE Trans. Inform. Theory, 13, pp. 57-67, 1967.

[7] D. Chen, J.M. Odobez, H. Bourlard, *"Text Detection and Recognition in Images and Video Frames"*. Pattern Recognition 37, pp. 595-608, 2004.

[8] T. Deselaers, *"Features for Image Retrieval"*. Master's Thesis, Human Language Technology and Pattern Recognition Group, RWTH Aachen University, Aachen, Germany, 2003.

[9] Ellacoya Networks Inc., *"Ellacoya Data Shows Web Traffic Overtakes Peer-to-Peer (P2P) as Largest Percentage of Bandwidth on the Network"*. http://www.ellacoya.com/news/pdf/2007/NXTcommEllacoyaMediaAlert.pdf June 2007.

[10] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, W. Equitz, *"Efficient and Effective Querying by Image Content"*. Journal of Intelligent Information Systems, Vol. 3, No. 3/4, pp. 231-262, July 1994.

[11] J. H. Friedman, J. L. Bentley, R. A. Finkel, *"An Algorithm for Finding Best Matches in Logarithmic Expected Time"*. ACM Trans. Math. Software, 3(3), pp. 209-226, 1977.

[12] C. Garcia, X. Apostolidis, *"Text Detection and Segmentation in Complex Color Images"*. Proceedings of 2000 IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. 2326-2330, 2000.

[13] C. Garcia, X. Apostolidis, *"Text Detection in Indoor/Outdoor Scene Images"*. Proc. First Workshop of Camera-based Document Analysis and Recognition, pp. 127-132, 2005.

[14] C. Graczyk, *"Vision Texture"*. http://vismod.media.mit.edu/vismod/imagery/VisionTexture/vistex.html March 05, 2007.

[15] A. Hampapur, A. Gupta, B. Horowitz, C.-F. Shu, C. Fuller, J.R. Bach, M. Gorkani, R.C. Jain, *"Virage Video Engine"*. Proc. SPIE Vol. 3022, pp. 188-198, 1997.

[16] V. I. Levenshtein, *"Binary Codes Capable of Correcting Deletions, Insertions, and Reversals"*. Soviet Physics Doklady 10, pp. 707-710, 1966.

[17] N. Otsu, *"A Threshold Selection Method from Gray Level Histograms"*. IEEE Trans. Syst. Man Cybern. SMC-9, pp. 62-66, 1979.

[18] R. Paredes, A. Perez-Cortes, *"Local Representations and a Direct Voting Scheme for Face Recognition"*. Workshop on Pattern Rec. and Inf. Systems, pp. 71-79, 2001.

[19] D. Ponceleon, A. Amir, S. Srinivasan, T. Syeda-Mahmood, and D. Petkovicn, *"CueVideo: Automated Multimedia Indexing and Retrieval"*. Proc. ACM Multimedia, vol. 2, Orlando, USA, 1999.

[20] J. Sauvola, M. Pietikainen, *"Adaptive document image binarization"*. Pattern Recognition 33(2), pp. 225-236, 2000.

[21] M. Sezgin, B. Sankur, *"Survey over image thresholding techniques and quantitative performance evaluation"*. Journal of Electronic Imaging Volume 13, Issue 1 , pp. 146-168, January 2004.

[22] F. Shafait, D. Keysers, T. M. Breuel, *"Efficient Implementation of Local Adaptive Thresholding Techniques Using Integral Images"*. Document Recognition and Retrieval XV, San Jose, CA, 2008.

[23] H. Tamura, S. Mori, T. Yamawaki, *"Textural Features Corresponding to Visual Perception"*. IEEE Transaction on Systems, Man, and Cybernetcs, vol. SMC-8, No. 6, pp. 460-472, June 1978.

[24] H. Tran, A. Lux, H. L. Nguyen, A. Boucher, *"A Novel Approach for Text Detection in Images Using Structural Features"*. Proceedings of the third International Conference on Advances in Pattern Recognition (ICAPR), Vol. 1, pp. 627-635, 2005.

[25] A. Ulges, C. Schulze, D. Keysers, T. Breuel, *"Content-based Video Tagging for Online Video Portals"*. Third MUSCLE / ImageCLEF Workshop on Image and Video Retrieval Evaluation, 2007. (accepted for publication)

[26] A. Ulges, C. Schulze, D. Keysers, T. Breuel, *"A System that Learns to Tag Videos by Watching Youtube"*. International Conference on Computer Vision Systems, 2008.

[27] Jie Xi, Xian-Sheng Hua, Xiang-Rong Chen, Liu Wenyin, Hong-Jiang Zhang, *"A Video Text Detection and Recognition System"*. IEEE International Conference on Multimedia and Expo (ICME 2001), Waseda University, Tokyo, Japan, 2001.

[28] *"YouTube"*. http://www.youtube.com/ February 26, 2007.

[29] Y. Zhuang, Y. Rui, T. Huang, S. Mehrotra, *"Applying Semantic Association to Support Content-based Video Retrieval"*. International workshop on Very Low Bitrate Video Coding(VLBV'98), UIUC, USA, 1998.