TAMPERE UNIVERSITY OF TECHNOLOGY

Antti Hietanen

LOCAL FEATURES FOR VISUAL OBJECT MATCHING AND VIDEO SCENE DETECTION

Master's Thesis

Examiners: Professor Joni-Kristian Kämäräinen

Examiners and topic approved in the Computing and Electrical Engineering Faculty Council meeting on 4th November 2015

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
ANTTI HIETANEN: **Local features for visual object matching and video scene detection**
Master of Science Thesis, 63 pages
September 2015
Major: Signal Processing
Examiner: Prof. Joni-Kristian Kämäräinen
Keywords: Local features, Feature detector, Feature descriptor, Object class matching, Scene boundary detection

Local features are important building blocks for many computer vision algorithms such as visual object alignment, object recognition, and content-based image retrieval. Local features are extracted from an image by a local feature detector and then the detected features are encoded using a local feature descriptor. The resulting features based on the descriptors, such as histograms or binary strings, are used in matching to find similar features between objects in images.
In this thesis, we deal with two research problem in the context of local features for object detection: we extend the original local feature detector and descriptor performance benchmarks from the wide baseline setting to the intra-class matching; and propose local features for consumer video scene boundary detection.
In the intra-class matching, the visual appearance of objects semantic class can be very different (e.g., Harley Davidson and Scooter in the same motorbike class) and making the task more difficult than wide baseline matching. The performance of different local feature detectors and descriptors are evaluated over three different image databases and results for more advance analysis are reported.
In the second part of the thesis, we study the use of Bag-of-Words (BoW) in the video scene boundary detection. In literature there have been several approaches to the task exploiting the local features, but based on the author's knowledge, none of them are practical in an online processing of user videos. We introduce an online BoW based scene boundary detector using a dynamic codebook, study the optimal parameters for the detector and compare our method to the existing methods. Precision and recall curves are used as a performance metric.
The goal of this thesis is to find the best local feature detector and descriptor for intra-class matching and develop a novel scene boundary detection method for online applications.

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
ANTTI HIETANEN: **Paikalliset piirteet visuaalisten objektien luokittelussa ja videon otosten leikkauskohtien tunnistamisessa**
Diplomityö, 63 sivua
Syyskuu 2015
Pääaine: Signaalinkäsittely
Tarkastaja: Prof. Joni-Kristian Kämäräinen
Avainsanat: Paikallinen piirre, Piirrehavaitsin, Piirrekuvaaja, Objektiluokkien samankaltaisuuden tunnistaminen, Otosten leikkauskohtien tunnistus

Paikalliset piirteet ja niiden kuvaajat ovat tärkeitä komponentteja monessa tietokonenäköön liittyvissä algoritmeissa kuten visuaalisten objektien kohdistamisessa, visuaalisten piirteiden tunnistamisessa ja sisältöpohjaisen informaation haussa. Paikalliset piirteet irroitetaan kuvasta piirreirroittimen avulla ja tämän jälkeen nämä piirteet koodataan piirrekuvaajan avulla. Piirrekuvaukset voivat olla esimerkiksi histogrammeja tai binäärisiä merkkijonoja, joita käytetään samankaltaisten piirteiden tunnistamisessa objekteista kahden eri kuvan välillä.

Tässä opinnäytetyössä tutkimme kahta eri ongelmaa liittyen paikallisten piirteiden käyttämiseen objektien tunnistamisessa: laajennamme alkuperäistä paikallisten piirteiden havaitsijoiden ja kuvaajien suorituskyvyn testausmenetelmää samasta objektista samaan luokkaan kuuluvien objektien tunnistamiseen ja hyödynnämme paikallisia piirteitä leikkauskohtien tunnistamiseen videosta.

Saman luokan sisäisten objektien ulkomuoto voi olla hyvinkin eri näköinen (esimerkiksi moottoripyöräluokan aliluokat Harley Davidson ja skootteri) ja täten tekee objektien tunnistamisesta hankalempaa kuin saman objektin tunnistamisesta eri ympäristössä. Eri piirteen havaitsijoiden ja kuvaajien suorituskyky arvioidaan työssä kolmen eri tietokannan avulla ja tuloksia analysoidaan yksityiskohtaisesti.

Opinnäytetyön toinen osa koostuu "piirrelaukku-menetelmän tutkimisesta ja sen hyödyntämisestä videon eri leikkauskohtien tunnistamisessa. Kirjallisuudessa on esiintynyt muutamia menetelmiä tehtävän ratkaisemiseksi hyödyntäen paikallisia piirteitä, mutta mikään niistä ei ole käytännöllinen online-järjestelmän kannalta. Seuraavien kappaleiden aikana esittelemme uuden piirrelaukku-menetelmän videoleikkausten tunnistamiseen hyödyntäen muuttuvaa koodikirjastoa, etsimme optimaaliset parametrit menetelmälle ja vertaamme sitä jo olemassa oleviin menetelmiin. Suorituskyvyn mittaamiseen käytämme "precision and recall"käyrää.

Tämän opinnäytetyön tavoite on löytää paras piirrehavaitsin ja -kuvaaja tunnistamaan eri objektit samasta luokasta ja kehittää uudenlainen leikkauskohtien tunnistusmenetelmä reaaliaikaisia järjestelmiä varten.

PREFACE

First of all, I wish to thank my supervisor Professor Joni-Kristian Kämäräinen for giving me an opportunity to work in his research group and for his guidance and patience during the work. I wish to also thank Jukka Lankinen D.Sc. (Tech) for his help and support. I want to thank the whole computer vision group at Tampere University of Technology providing an enjoyable work environment.

Finally, I want to thank my family and especially my love, Pinja.

Tampere, September 21th, 2015

Antti Hietanen

CONTENTS

ABBREVIATIONS AND SYMBOLS

σ	Gaussian kernel size
σ_n	Number of edge pixels in frame n
λ_1	Eigenvalue calculated form **M**
λ_2	Eigenvalue calculated form **M**
τ	Threshold value
$\tau(\mathbf{P};\mathbf{x},\mathbf{y})$	Binary test in BRIEF
ϵ_0	Overlap error
θ	Orientation
b	Color component
c_i	Correspondence score
ECR_n	Edge change ration in frame n
F	F-measure
$D(\mathbf{x},\sigma)$	Difference-of-Gaussian
$G(\mathbf{x},k\sigma)$	Gaussian kernel
g	Color component
Det	Determinant
$\mathbf{H}$	Hessian matrix
H	Tomography of two images
$L(\mathbf{x})$	Intensity function of Gaussian smoothed image
$L_x(\mathbf{x})$	Partial derivative in x direction of $I(x)$
$L_y(\mathbf{y})$	Partial derivative in y direction of $I(x)$
$L_{xx}(\mathbf{x})$	Second-order partial derivative in x direction of $I_x(x)$
$L_{xy}(\mathbf{x})$	Second-order partial derivative in y direction of $I_x(x)$
$L_{yy}(\mathbf{x})$	Second-order partial derivative in y direction of $I_y(x)$
C	Centroid of a patch
K	Number of best matches
$\mathbf{M}$	Second moment matrix
m_i	Number of matches
m_{pq}	Moment of a patch in ORB detector
N	Number of correct matches in Coverage-N measurement
$\mathbf{P}$	Patch in BRIEF
P	Precision
R_μ	Detected elliptic region
R	Recall
r	Color component
S	Size of an image patch

$\mathbf{x}$	Coordinate point
T	Threshold value
Tr	Trace
X_n^{in}	Number of entering edge pixels in frame n
X_{n-1}^{out}	Number of exiting edge pixels in frame $n-1$

2D	Two dimensional
3D	Three dimensional
BoW	Bag-of-Words
BoF	Bag-of-Features
BRIEF	Binary Robust Independent Elementary Features
DoG	Difference-of-Gaussian
HoG	Histogram of Oriented Gradients
LoG	Laplacian-of-Gaussian
NIST	National Institute of Standards and Technology
ORB	Oriented BRIEF
PCA	Principal Component Analysis
RGB	Red-Green-Blue Color Space
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine
TRECVid	TREC Video Retrieval Evaluation

1 INTRODUCTION

As humans, we have powerful ability to perceive the three-dimensional world around us. You can immediately recognise different flowers from the vase and maybe even name their sort. Looking at unfamiliar object, for instance an animal or food, you can draw a conclusion of the target by comparing it to your earlier experiences with objects having similar colour, texture or shape appearance. There have been a great deal of work trying to understand how the human visual system works but the solution remain still unaccomplished [1].

Despite the difficulties in modelling human visual system, over the years there have been significant progress in the field of computer vision [2, 3]. Development of computer vision have given us great potential to change the way we interact with computers and help us to augment the physical real-world. Researchers have developed mathematical techniques which computers utilize for recovering the three-dimensional shape and appearance of objects in images. We have already made reliable techniques to make panoramic view of an environment from thousands of partially overlapping photographs. Another popular application is for instance face recognition which can be found from social media applications and surveillance systems.

In Figure 1 is shown some active topics of the field of computer vision during different decades. In the 1970s the computer vision was distinguished from the common image

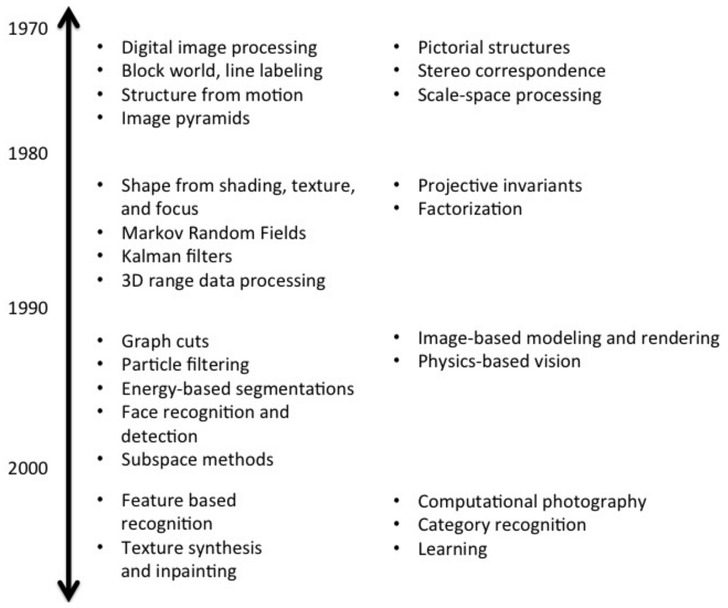


Figure 1. Active topics over different decades on computer vision research [3].

processing by the desire of recovering the three-dimensional structure of the world from images and use this information to full scene understanding. The earliest attempts to extract information from the scene was extracting edges and then inferring the 3D structure of an object or a "block world" from the topological structure of the 2D lines [4].

In the 1980s lot of efforts was putted to improve existing algorithms and mathematical techniques. Moravec [5] introduced *interest operator* based on the intensity variation in the local neighbourhood of a pixel. Harris and Stephen improved this method in 1988 by presenting the *Harris* detector [6].

One of the most noteworthy result achieved in the 1990s was development in the area of image-based modelling and rendering due to progress on computer graphics [7] which lead to numerous image manipulation techniques such as image morphing and view interpolation.

Interest points and other feature-based techniques have been the trend for object recognition during the 1900s and the 2000s. Many successful approaches have been presented, such as SIFT [8], which was well accepted and is still used in many works as a baseline for new approaches. One of the active topics of 2000s were *category-level* or *intra-class* object recognition [9, 10, 11], where the problem is to classify an object to a particular general class such as "dog", "car" or "bicycle". A popular algorithm for category recognition is the Bag-of-Words (also noted as Bag-of-Features, Bag-of-Keypoints, and BoW) approach, where the term *bag of keypoints* was first used in 2004 by Csurka *et al.* [9]. During this decade more and more interest have gained the deep neural architectures [12].

1.1 Object recognition using local features

In literature, the term *recognition* or generally *detection* refers to detecting an instance of a particular object class under many viewing conditions in unconstrained settings (Figure 2). A common way is to extract local features by a feature detector, encode them into descriptions by a feature descriptor, and use the descriptions as an input data for certain task such as object categorization or image stitching.

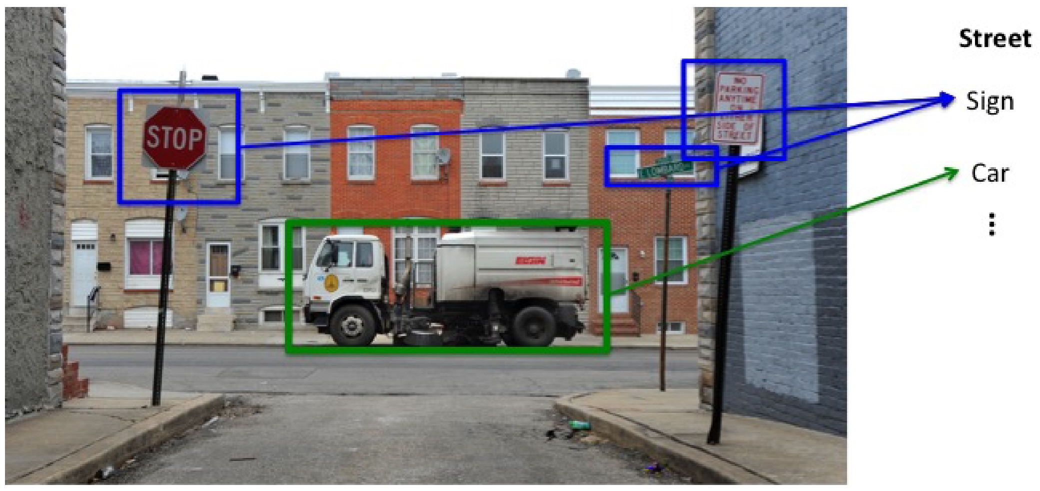


Figure 2. An example of object recognition.

1.1.1 Local features

A local feature is an image pattern which differs from its immediate neighbourhood [13], for instance a point on an image where a small shift in multiple directions result large intensity changes in the texture. In addition to texture, changes in other image properties, such as colour, can result an interest point depending on an algorithm. These detected interest points are then encoded by a feature descriptor to *descriptions*, which are ultimate feature of an application.

Why someone should use local features? One could be interested in specific type of features: in the earlier development of local features, they were used to detect corners and edges, such as the Harris detector [6]. A typical application could be a detection of road edges from an aerial image. Secondly, one might not be interested in the actual representation of an interest point itself, more like the accurate location of it and that it can be found under various transformations. Depending on a algorithm, local features can provide different type of features and be robust to various image transformations and hence local features are used as anchor points needed in many matching tasks, for example, in object retrieval [10, 14, 15] and mosaicing [16]. Thirdly, slightly related to the second point, using local features allows us to detect objects without segmentation because local features are highly tolerant to occlusion and clutter which are the main reason for segmentation.

1.1.2 Local features in visual object categorization

In visual object categorization (VOC), the task is to find category for an object in a given image. The task can be hard due to numerous transformations in the image (described in Section 1.1.3) and presence of multiple instances of the same and different category objects in the image. VOC methods can be divided into two different learning approaches: *unsupervised* and *supervised*.

- *Supervised*: Before the actual detecting process, a subset of local features from object category are used to train a model which represent instances of this object category. The model should be as discriminative as possible in order to distinguish two or more different categories.
- *Unsupervised*: In a unsupervised approach, we are not provided local features with correct category labels to train the model. The task is to find the hidden characteristics of the object category, which makes the unsupervised learning more challenging task than supervised.

Why then to choose unsupervised learning? In real life tasks the common unambiguous characters of some object category are not know or they are impossible or impractical to attain. Also by unsupervised learning one can explore data that is considered to be purely unstructured and find hidden patterns from there.

A typical unsupervised approach is to cluster unseen data to different clusters. Clustering is a technique to group similar multidimensional data to the same group (cluster) by means of some metric. One of the most influential approach in VOC which utilize clustering is BoW. In VOC systems, BoW uses clustering as an initialization step (form codewords of the codebook) and in the building process of code histograms. A codeword is calculated from an unseen data set of feature descriptors usually by the k-means algorithm [3], which searches k clusters from unpartitioned data. Then, a codeword is defined as the center of the learned cluster. In the initialisation phase of a codebook, one calculates the mean value of each cluster which are then selected to represent our codebook, a vector of length k. Now, one can calculate a code histogram of an image by extracting local features from the image and assigning the features to the nearest codeword in the codebook. This histogram can then be compared to another image histogram calculated in the same way.

1.1.3 Challenges in object recognition

In real life detecting things such as cars and traffic signs in a street image is often a challenging task because of view point variation, difference of object scales or varying illumination conditions (Figure 3). In Figure 4 is shown different instances of the object class chair. Such a large intra-class variation in appearance and shape makes the detection challenging and this is studied in Chapter 3. Other obstacles in object detection are occlusion, where the object is partially hidden by some other object and background clutter, where the background of the object can severely affect the detection performance (Figure 5). An ideal system, which output does not depend to these transformations is said to be *invariant*, such as invariant to rotation or rotation-invariant.

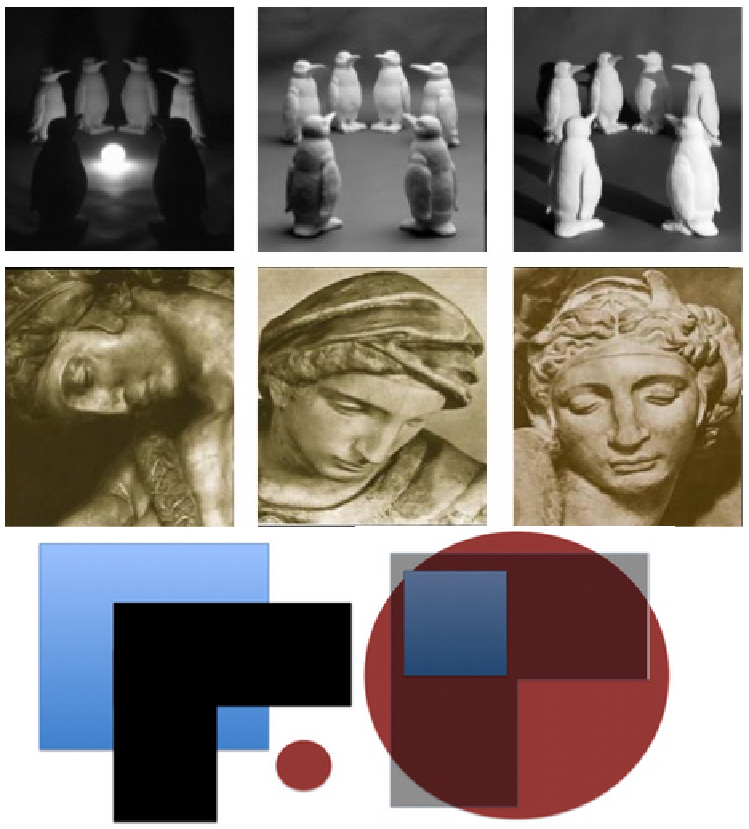


Figure 3. Different objects under illumination change (top), view point variation (middle), and scale change.

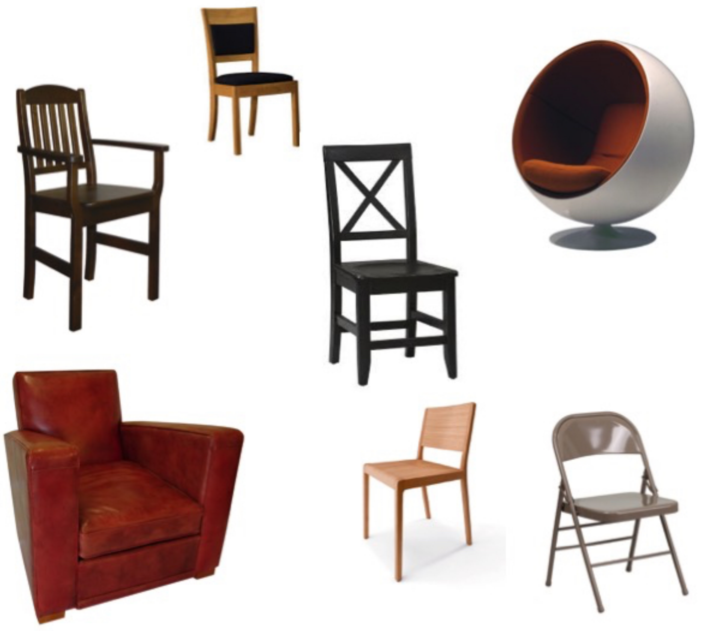
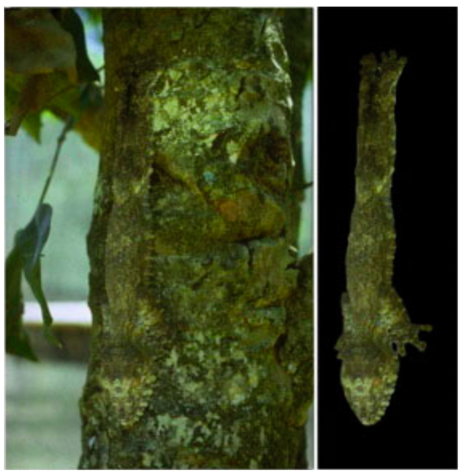


Figure 4. Different instances of the same object category.


(a)

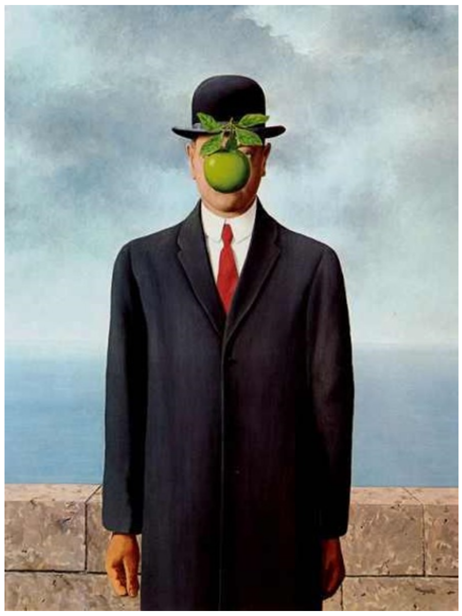

(b)

Figure 5. a) Background clutter: A gecko is camouflaged to be part of the tree and thus hard to detect [17]. b) Occlusion: A painting by Réne Magritte where a man face is occluded by an apple.

1.2 Structure of the thesis

In section 1 we give the reader a general view of computer vision, visual object detection and categorization. After that the thesis has two main parts.

The first part consist of an introduction of well known and efficient detectors and descriptors in Chapter 2. Performance of these detectors and descriptors in matching task for visual objects classes is studied in Chapter 3. The second part of the thesis starts with description of Bag-of-Features object classification which is described in Chapter 4. Second part ends to an application exploiting Bag-of-Features method in a video scene detection in Chapter 5. A short conclusion of the work done is given in Section 6.

1.3 Goals and restrictions

In the first part, the main goal is to find the best methods for local feature detection and matching. The evaluations is done for local feature detectors and descriptors. The problem is to distinguish instances of the same class and instances from different class from each other. This is a different task such as the wide-baseline matching where the matching is performed for the same objects with different view points. Also for fair comparison of local feature detectors and descriptors, the meta-parameters of different approaches must be configured.

In the second part, the main interest is to develop a novel online video scene boundary detector using local features and Bag-of-Features. The problem is to make a accurate method which detects a cut location between two different scenes, for instance video changing from a ballet to a basketball match, and works in online so that video manipulation can be done at the same time while the video is processed. In literature there have been different approaches to solve the problem and some of them have been already successful, but none of them are practical for an online application.

2 LOCAL FEATURE DETECTORS AND DESCRIPTORS

In this section we will briefly introduce the properties of local feature detectors and descriptors. We will also give short descriptions of the methods which will be used in Chapter 3 for visual object class matching.

2.1 Introduction

Local feature detectors and their descriptions are the building blocks of many computer vision algorithms. A detector provides detected regions or points for a descriptor which describes them in a certain form. Once we have obtained the features and their descriptions from some query image we can compute features from another images and find similar parts between the images (Figure 6). Local descriptors have been used successfully in many applications such as wide baseline matching [18], object recognition [19], texture recognition [20], robot localization [21], image and video data mining [22], and panoramic photography [16].

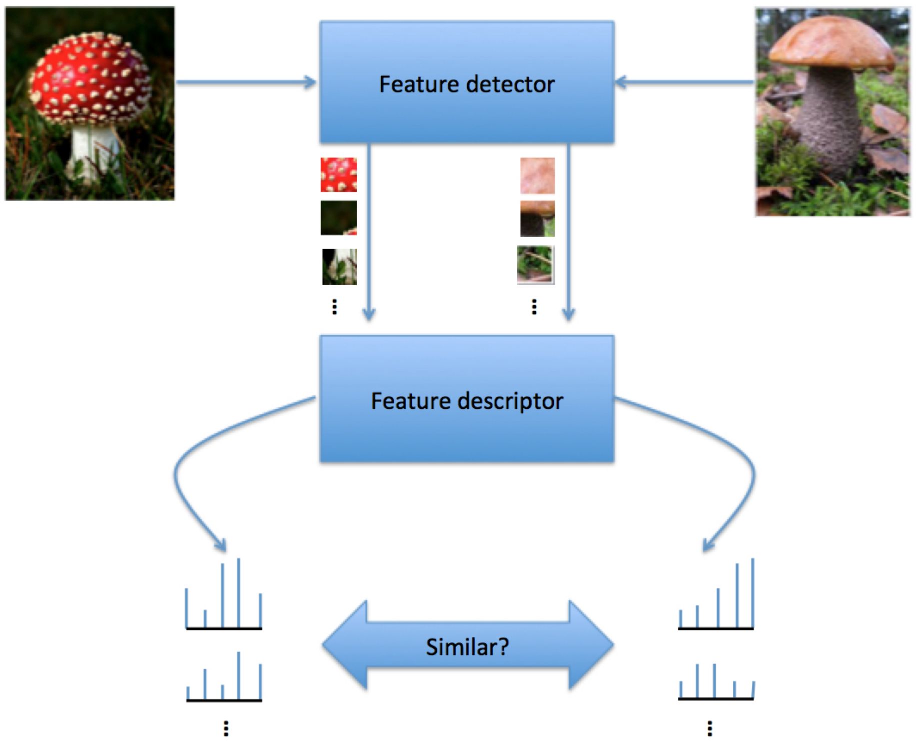


Figure 6. The process of finding similar features between two image.

From an image one can extract two kind of low-level features: *global* features or/and *local* features. Global features are usually computed from every pixel of the image and they represents the whole image by a single feature element, such as a vector. These kind of features are for example RGB histograms [15] (concerning the colour information of the image) or calculating a Grey Level Co-occurrence Matrix of the image (concerning the textural features of the whole image) [23]. On the other hand local features are computed from multiple locations on the image and as a result we get multiple feature vectors from a single image. One advantage of local features over global features is that they may be used to recognize the object despite significant clutter and occlusion and thus do not require a segmentation of the object from the background unlike many texture features. Advantages of global features are simplicity of the implementation and a compact representation of image features. Combining global and local features we can gain even a 20% boost to performance than using features separately [24]. In this thesis we will now focus on local features.

The terminology in the literature can be sometimes confusing because of the many different terms refer to the same thing. In this thesis, we will mainly use local features and keypoints, but also interest points and interest regions are used in the literature. For the descriptors there exists much less variety and often descriptors and descriptions are used. In the thesis, we will use the term local feature descriptors for description of local features.

2.2 Local feature detector properties

Feature detection is a low-level image processing operation which output serves as an input for descriptor. Because feature detector is the initial component in the whole pipeline process of object recognition, the overall result will be as good as the feature detector. The detectors can be roughly divided into four different categories: *edge detector*, *corner detector*, *blob detector*, and *region detector*. In this thesis we are focusing on blob detectors which are discussed in Section 2.4. The detector example images in following sections are computed using VLFeat toolbox [25] by Andrea Vedaldi in Matlab and OpenCV libraries.

In [13] are listed the following six properties (*repeatability* being the most important one) what an ideal local feature has and what a good local feature detector should search:

- *Repeatability:* Between two images taken from the same scene (with various transformations) the detector should extract the same visual features. Repeatability can be obtained by two ways: either by *invariance* or by *robustness*.

- *Distinctiveness/informativeness:* Different detected features should be discriminative enough so they can be distinguished and matched easily.
- *Locality:* The features should be local enough to tolerate feature deformation such as occlusion, noise, scale and rotation.
- *Quantity:* The number of detected features should be large enough, even from on small object. Ideally the number of features should be adjustable by a threshold.
- *Accuracy:* The properties concerning the detected feature localization (location, scale and shape) should be accurate.
- *Efficiency:* A faster implementation of a detector is always worth to pursue. In a real-time application it is a crucial property.

In Figure 7 is shown extracted features on the same object in different images. From the images we can see that exactly the same object characteristic features are found despite scale, viewpoint and minor contrast changes.

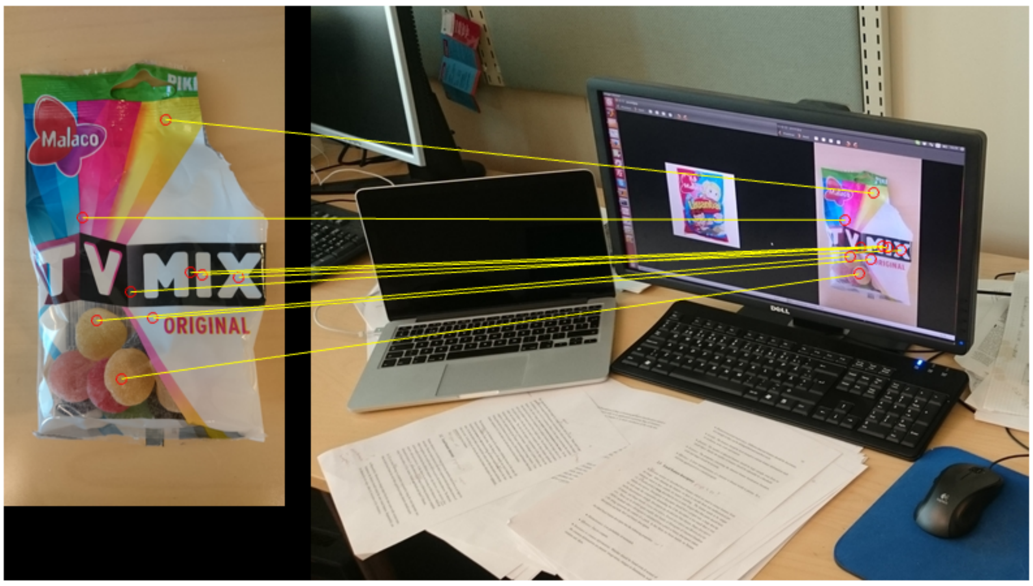


Figure 7. The detected SURF [26] features on two different images. The yellow lines show which regions match in descriptor space.

2.3 Local feature descriptor properties

In the previous section we described local features which are used to extract features from an image. After obtaining the features, we have to encode them into mathematical presentation which is then suitable for feature matching. The feature needs to be unique i.e. if a similar point is described in two or more images then that point should have similar *description*. Description or descriptor should have a proper size, a large descriptor will make the computation demanding but if the descriptor is too small then it may not be discriminative enough. Descriptors are divided into four classes: *Shape descriptors* (Contour- and Shape-based), *Colour descriptors*, *Texture descriptors* and *Motion descriptors*. These can be divided again roughly into family of Histogram of Oriented Gradient (HOG) methods where a histogram is calculated from a patch or binary descriptors where a bit string represents the content of a patch. In this thesis we focus mainly on Texture descriptors and both, HoG and binary descriptors.

An ideal local feature descriptor has the following properties [27]:

- *Distinctiveness:* Low probability of mismatch.
- *Efficiency:* Fast to compute.
- *Invariance to common deformations:* Matches should be found even if several of the common deformations are present: image noise, changes in illumination, scale, rotation and, skew.

2.4 Early work on local features

2.4.1 Hessian detector

Hessian detector proposed by Beaudet *et. al* [28] in 1978 is one of the first published blob detection algorithms in the literature. Regions detected by the Hessian algorithm are shown in Figure 8. The family of Hessian-based detectors consist of the original Hessian keypoint detector, Hessian-Laplace, and Hessian-Affine. The Hessian detector is based on the 2×2 *Hessian* matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} L_{xx}(\mathbf{x}) & L_{xy}(\mathbf{x}) \\ L_{xy}(\mathbf{x}) & L_{yy}(\mathbf{x}) \end{bmatrix}, \tag{1}$$

where the terms L_{xx}, L_{xy}, and L_{yy} denote the second-order partial derivatives of $L(\mathbf{x})$ at location $\mathbf{x} = (x, y)$ i.e., the gradients in the image in different directions. Before we calculate the second order derivatives the image is smoothed by taking the convolution between the image I and a Gaussian kernel with scale σ:

$$L(\mathbf{x}) = G(\sigma) \otimes I(\mathbf{x}) \tag{2}$$

To find keypoints, the determinant of the Hessian is computed as:

$$\text{Det}(\mathbf{H}) = L_{xx}L_{yy} - L_{xy}^2 \tag{3}$$

The determinant of the Hessian matrix is calculated in every point $\mathbf{x}$ in the given image. After obtaining the determinant for every pixel, we search for points where the determinant of the Hessian becomes maximal. This is done by using 3×3 window where the window is swept over the entire image, keeping only pixels whose value is larger than the values of all 8 immediate neighbours inside the window. Then, the detector returns all the remaining locations whose value is higher than a pre-defined threshold value τ and these will be selected as keypoints.

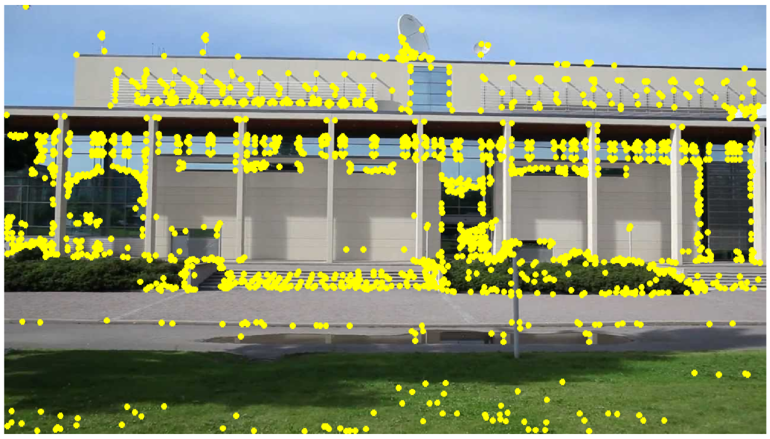


Figure 8. Local features detected by the original Hessian detector. Note the approach convention to detect features from object contours and corners.

However this approach is not robust to various transformations in the images. *The Hessian-Laplacian* detector was introduced to increase robustness and discriminative power of the original Hessian detector. It combines the Hessian operator's specificity for corner-like structures with the scale selection mechanism presented by Lindeberg [29]. The Lindeberg mechanism searches for scale space extrema of a scale-normalized *Laplacian-of-Gaussian* (LoG):

$$S = \sigma^2(L_{xx} + L_{yy}), \tag{4}$$

where σ^2 is the scale factor. In the Hessian-Laplacian method we build separate spaces for the Hessian functions and the Laplacian. Then we use Hessian to detect candidate points from different scale levels and select those candidates to our keypoints which the Laplacian simultaneously attains an extremum over scale. Now, the obtained keypoints will be robust to changes in scale, image rotation, illumination and camera noise.

To achieve invariance against viewpoint changes an extended version of Harris-Laplacian have been pronounced: *Hessian-Affine* detector [30, 31]. First, we use the scale-invariant Hessian to find initial keypoints and estimate the shape of the region with the second moment matrix **M**:

$$\mathbf{M} = G(\sigma)\begin{bmatrix} L_x^2(\mathbf{x}) & L_xL_y(\mathbf{x}) \\ L_xL_y(\mathbf{x}) & L_y^2(\mathbf{x}) \end{bmatrix}, \tag{5}$$

where we can compute the eigenvalues λ_1 and λ_2 by the following equations:

$$\text{Tr}(\mathbf{M}) = \lambda_1 + \lambda_2, \tag{6}$$

$$\text{Det}(\mathbf{M}) = \lambda_1\lambda_2. \tag{7}$$

The obtained eigenvalues from Equations 6 and 7 gives us an elliptical shaped region, corresponding to a local affine deformation. The elliptical shape is normalized to circular one and the point location and scale is recovered. Again the moment matrix is calculated from the normalized region and this process is iteratively continued until the the eigenvalues of the matrix are equal.

2.4.2 SIFT detector and descriptor

The SIFT (Scale-Invariant Feature Transform) detector was first introduced by David Lowe in 1999 [8] and later improved in 2004 [32]. In contrast to Harris [6] method where corners are localized when there is low auto-correlation in all direction, SIFT localizes features where there are "blob-like" structures in the image. Lowe aimed to create a detector which would be a invariant to translation, rotation and scale. The SIFT algorithm builds a scale-space representation of the original image. This is achieved by a Difference of Gaussian (DoG) approach combined with interpolation over the scale-space which leads to the locations of stable keypoints in that scale-space representation of the image. After the localization, each keypoint is assigned an orientation, which leads to the desired rotation invariance. Below are listed the key steps in SIFT algorithm:

1. Scale-space extrema detection: Image is convoluted with Gaussian filters on different scales and then the difference of successive Gaussian-blurred images are taken. This is called Difference of Gaussians (DoG) can be written as:

$$D(\mathbf{x}, \sigma) = L(\mathbf{x}, k_i\sigma) - L(\mathbf{x}, k_j\sigma), \tag{8}$$

   where $L(\mathbf{x}, k_k\sigma)$ is the convolution of the original image $I(\mathbf{x})$ with Gaussian blur $G(\mathbf{x}, k_k\sigma)$ at scale $k_k\sigma$. Once we have obtained DoG images, keypoint candidates are identified as local minima/maxima of the DoG images over all scales.

2. Keypoint localization: After the scale-space extrema detection we have gathered lots of keypoint candidates and we have to perform an eliminations of the unstable keypoints. A detailed fit to nearby data is performed to determine location, scale, and ratio of principle curvatures. In Lowe's first method the keypoints accurate position and scale of a central sample point was acquired. In more recent work, the author fit a 3D quadratic function to improve interpolation accuracy. In addition, the Hessian matrix was used to eliminate edge responses with poorly determined locations.

3. Orientation assignment: In this step we achieve rotation-invariance by assigning to each keypoint one or more orientations based on local image gradient directions. We create a histogram of local gradient directions at selected scale. Histogram is then smoothed by every sample's gradient magnitude and by a Gaussian weighted circular window. Finally canonical orientation is assigned at peak of smoothed histogram where each key specifies stable 2 dimensional coordinates. In Figure 10 is shown detected SIFT features with the orientation.

The SIFT keypoint descriptor computation is shown in Figure 9. The left image of the figure with small arrows represents the gradient magnitude and orientations at each sample locations. The circle around the samples is a Gaussian weighting function. It is used to weight the gradient magnitudes of each sample point in a way that it gives less emphasis to gradients that are far from the center of descriptor. The right part of the figure represents the 2×2 descriptor matrix of the 8×8 neighborhood. Every cell in the matrix contains accumulated gradients to 8 directions from the corresponding 4×4 subregions.

However, the best results are typically achieved with 4×4 descriptor matrix with 8 orientations [32]. In this case our descriptor has $4 \times 4 \times 8 = 128$ dimension in total. Finally the descriptor vector is normalized to enhance invariance to affine changes in illumination.

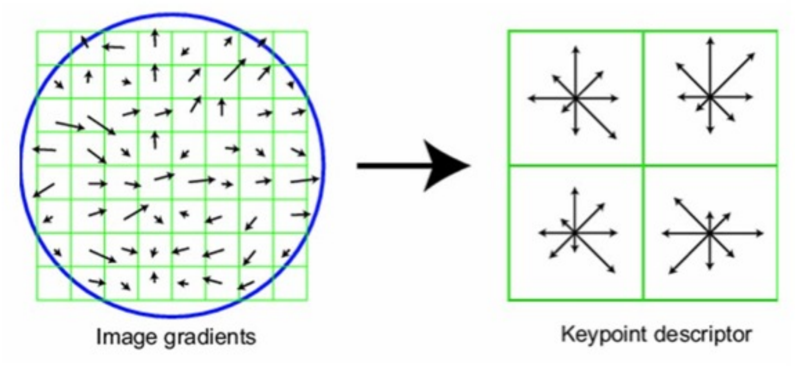


Figure 9. A 2×2 SIFT descriptor (right) computed from 8×8 sample neighbourhood (left).

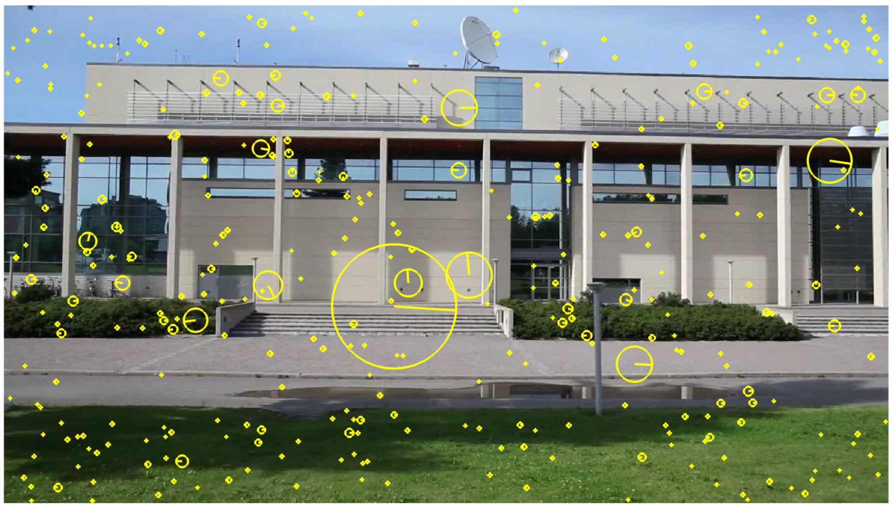


Figure 10. Detected SIFT features and their orientations.

2.4.3 Dense sampling

Dense sampling is one of the most commonly used low-level image representation method, which uses a fixed pixel interval (horizontally and vertically) between sample points i.e., points are sampled on a regular dense grid. The chosen length of the interval between sample points determines the number of patches to be sampled from the image. The interval is usually set that different patches are overlapping 50% [33]. Keeping the interval value same for same size images the "detector" generates constant amount of samples despite for instance contrast shifts and thus dense sampling on a regular grid results in a good coverage of the entire object or scene. Other benefit of dense sampling is a regular spatial relationship between sampled patches where as local feature detectors find interest points only from specific locations (corners, blobs, etc.) which makes it hard to model spatial configuration of features (for instance the spatial relationship between eyes and mouth on human face). However, dense sampling cannot reach the same level of repeatability as local features do. In [34] it is shown that dense sampling outperforms local features on Bag-of-Words based classification. In Figure 11 is illustrated the sampled patches having 50 pixel distance to each other.

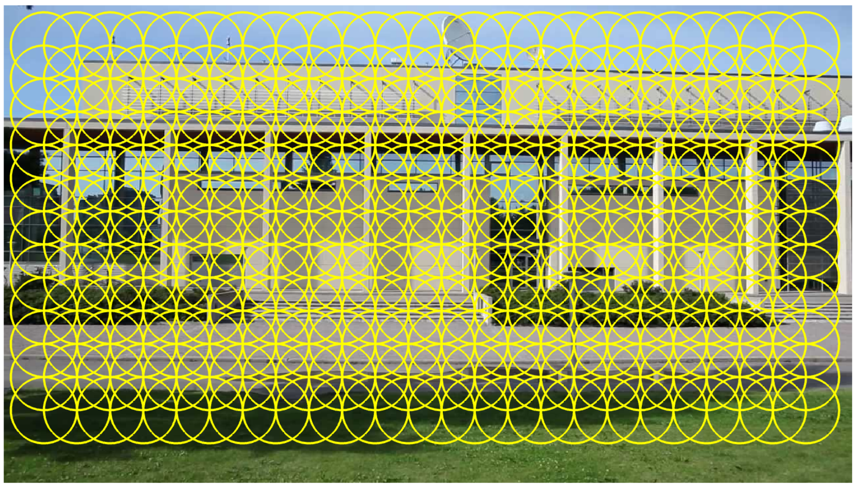


Figure 11. Dense sampling.

2.5 More recent and efficient implementations

2.5.1 BRIEF descriptor

BRIEF (Binary Robust Independent Elementary Features) was one of the first published binary descriptor by Calonder *et al.* [35]. Binary descriptors try to provide an alternative method to the widely used floating point descriptors such as SIFT. BRIEF was originally made to beat SURF [26] and U-SURF (upright version of SURF) descriptors on recognition performance, while only using a fraction of the time required by either. BRIEF can produce a very good results with only 256 bits, or even 128 bits, which is a significant advantage when millions of descriptors must be stored.

The approach main assumption is that image patches could be effectively classified on the basis of a relatively small number of pairwise intensity comparisons. The descriptors are binary strings where each bit represents a simple comparison between two points inside a patch. Before the binary tests we have to first smooth the patch using the Gaussian kernel of size σ. More precisely, we can define test τ on path **P** of size $S \times S$ as

$$\tau(\mathbf{P};\mathbf{x},\mathbf{y}) := \begin{cases} 1 & \text{if } \mathbf{P}(\mathbf{x}) < \mathbf{P}\,(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where **P**(**x**) is the pixel intensity in a smoothed version of **P** at **x** = $(u, v)^T$. The BRIEF descriptor is defined as a vector of *n* binary test:

$$f_n(\mathbf{P}) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x, y) \tag{10}$$

where a typical value for n is 128, 256 or 512 [35]. The only properties that have to be taken account when creating the descriptors are the kernels which are used to smooth the patches and the spatial arrangement of the (**x**, **y**)-pairs.

Smoothing of the patches is a necessary step in the process of obtaining the BRIEF descriptors. BRIEF is highly sensitive to noise due method's pixel-to-pixel test protocol. To make approach more robust and increase the stability and the repeatability different smoothing kernel sizes were studied by Calonder *et al.* [35]. They found out that a practical value for Gaussian kernel was 2. Spatial arrangement of the binary tests were their second experimented parameter. They tested five different sampling geometries, where the Uniform $(-\frac{S}{2}, \frac{S}{2})$ and the Gaussian $S(0, \frac{1}{25}S^2)$ (S was the size of the image patch) distributions were the best methods.

The authors reported the performance results of BRIEF against OpenCV implementation of SURF. The performance rate histograms were almost superposed and there were no significant differences in any categories. The speed comparison showed clear differences among methods and BRIEF was 35- to 41- fold faster building descriptors over SURF where the time for performing and storing the tests remains virtually constant. For matching the speed-up was 4- to 13-fold. U-SURF computation time was between these two, but still far away from the BRIEF results. However, BRIEF does not tolerate well orientation and rotation and with a bigger test set of different objects it might not compete with SIFT and SURF as it was noted by the authors.

2.5.2 ORB detector

ORB (Oriented FAST and Rotated BRIEF) is a fast local feature detector first introduced by Ethan Rublee, Vincent Rabaud, Kurt Konolige and Gary Bradski [36]. ORB is one of the most efficient detector algorithm in the field of image processing [37] and it is based on FAST (Features from Accelerated Segment Test) keypoint. ORB can also be used to compute the visual descriptors and the approach is based on a steered version of BRIEF with an additional learning step. The authors main idea was to develop a method which is a computationally-efficient replacement to SIFT that has similar matching performance, is less affected by image noise, and is capable of being used for real-time performance.

ORB starts the search of local features by FAST corner detector. It is a computationally efficient and produces high quality features. After we have retrieved the features we apply a Harris corner measure to find the top N features among them. We apply also a scale pyramid to get multiscale-features. However, the FAST does not compute the orientation so we do not have rotation-invariant features yet. Authors decided to use Rosin's [38] orientation measure, the *intensity centroid*, to include rotation invariance. First we define the moments of a patch as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y), \tag{11}$$

and centroid can be determined as

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}). \tag{12}$$

The corner orientation is then the angle of the vector $\overrightarrow{OC}$, where O is the corner's center. The orientation is simply calculated as:

$$\theta = atan2(m_{01}, m_{10}), \tag{13}$$

where atan2 is the quadrant-aware version of arctan. To improve the rotation invariance of this measure one should compute x and y remaining within a circular region of radius r, where r is the size of the patch. Below (Figure 12) are shown the detected features by ORB detector.

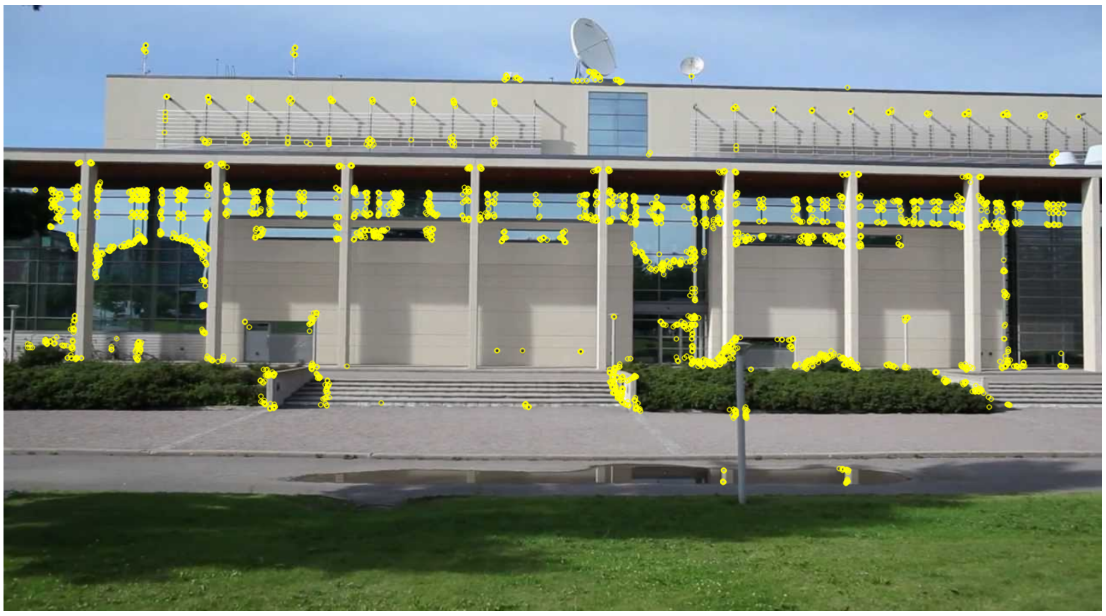


Figure 12. Detected ORB features.

3 COMPARISON OF FEATURE DETECTORS AND DESCRIPTORS FOR VISUAL OBJECT CLASS MATCHING

In Chapter 2 we presented local feature detectors and descriptors which are used in many computer vision problems such as panorama image stitching [16], robot localization [39], and wide baseline matching [18]. In all these cases, the feature correspondences are needed to match several views of the same scene. Local feature detectors and descriptors performance for these kind of problems have been already well covered by Mikolajczyk [30], Mikolajczyk and Schmid [27] and in the more recent paper [37] by Miksik and Mikolajczyk. Another interesting application would be to match two views of different objects from the same class. For instance we want to classify a mountain bicycle and a trial bike to be related because they are both from the same class bicycle. Although both are objects from the same class, their visual appearance can be very different, and thus, the original evaluation of detectors and descriptors are not directly applicable.

Various methods have been proposed for detecting interest points/regions and to construct descriptors from them, most of which are designed with a different applications in mind. In [30, 27] and [37] the authors evaluated and compared the most popular and recent detectors and descriptors. The detector were evaluated by their repeatability rations and total number of correspondences over several views of scenes and with various imaging distortion types. The descriptors were evaluated by their matching rates for the same views. Comparisons concerning object classification tasks were reported in [20] and [19], but in these works the evaluation was tied to a single methodological approach, namely visual Bag-of-Words (BoW). Moreover, many fast descriptors have been proposed recently: SURF [26], FREAK [40], ORB [36], BRISK [41], BRIEF [35]

In this chapter we will extend the original detector and descriptor benchmarks in [30, 27] to intra-class repeatability and matching. We evaluate the popular and efficient detectors and descriptors and their various implementations with the proposed framework. The detectors repeatability rates and the total number of correspondences for different objects from the same class were evaluated over 750 image pairs from different image databases. The descriptors were evaluated by their matching rate for the same data set. In addition, we investigate the effect of using multiple best matches ($K = 1, 2, ...$) and introduce an alternative performance measure: *match coverage*.

3.1 Previous work

The evaluation method described in this chapter is basically an extension of the evaluation framework from [30, 27]. The benchmark framework provides a fair way to compare detectors and descriptors by evaluating the overlap of the detected areas of interest as well as how well these regions actually match. We will follow the same evaluation principles as in the framework in our experiments: *repeatability* for detectors and *match count/ratio* for descriptors. Most of the results are included to the recent paper [42].

We refer to these repeating and matching regions as "category-specific landmarks". A qualitative measure to visualise descriptors "(HOGgles)" was recently proposed by Vondrick *et al.* [43], but its main use is in image-wise debugging of existing methods. More quantitative evaluations were reported by Zhang *et al.* [20] and Mikolajczyk *et al.* [19], but these were quite heuristic and tied to a single methodology, the visual Bag-of-Words (BoW) [10, 9]. In this work, we show that the original evaluation principles can be adopted to obtain similar quantitative performance measures in general, comparable and intuitive forms used in the original works of Mikolajczyk *et al.*, and not tied to any specific approach.

3.2 Performance measurement protocol

We believe that the general evaluation principles in [30, 27] hold also in the visual object categorization context:

1. Detectors which return the same object regions for class examples are good detectors – *detection repeatability*.
2. Descriptors which match the same object regions between class examples are good descriptors – *match count/ratio*.

For detectors the *detector repeatability rate* is the most important value to measure performance of a detector [13]. We start the calculations by comparing all the detected regions in two image pairs. If the *overlap error* is smaller than ϵ_0 then two regions are deemed to correspond:

$$1 - \frac{R_{\mu_\alpha} \cap R_{(H^T \mu_b H)}}{R_{\mu_\alpha} \cup R_{(H^T \mu_b H)}} < \epsilon_0 \tag{14}$$

where R_μ represents the detected elliptic region defined by $x^T \mu x = 1$ and H is the homography relating the two images. Before calculating the overlap error we have to normalize the corresponding areas. This is done because the bigger the ellipses are, the smaller is the overlap error in the measurement and vice versa (Figure 13). To compensate

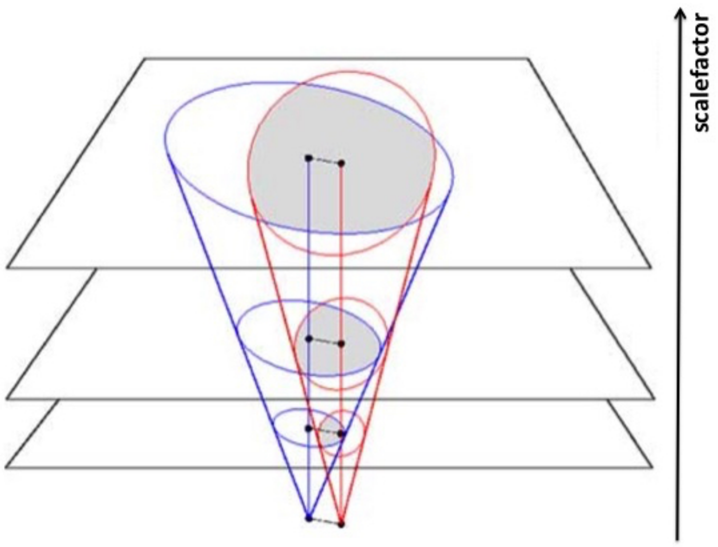


Figure 13. Size of the ellipses have an effect on the overlap measurement [30].

for the effect of regions of different sizes, Mikolajczyk determined the scale factor that transforms reference image into a region of normalized size (in the experiments normalized to a radius of 30 pixels). After that, we apply the determined scale factor to both the region in the reference image and the region detected in the other image which has been mapped onto the reference image by an estimated unknown affine homography. Now, the actual overlap error as stated in Equation 14, can be computed. In Mikolajczyk experiment the overlap error was set to 40%. After we have obtained all the correspondence regions, we can calculate the *repeatability rate* as:

$$repeatability\ rate = \frac{\#\ of\ correspondences}{min(\#\ of\ reg\ in\ img\ A, \#\ of\ reg\ in\ img\ B)} * 100 \tag{15}$$

Taking the minimum of detected regions in image A and B we will only include regions that are found in both images.

As we stated earlier a good descriptor should be discriminative to match only correct regions and also it should be robust to some small appearance variations between the examples. The descriptor performance were obtained in the Mikolajczyk and Schmid paper [27] by computing statistics of the correct and false matches. Our descriptor performance will be evaluated by the average number of matches and the median number of matches. In our work descriptors results are expected to be weaker due to increased appearance variance. For instance, scooter and road bikes are both found in the Caltech-101 motorbike category, but their pair-wise similarity is much weaker than between two scooters or two road bikes.

3.3 Data

Both detectors and descriptors were evaluated using three different image databases: Caltech-101 [44], R-Caltech-101 [45] and ImageNet [46]. Caltech-101 image database contains images and annotations for bounding boxes and outlines enclosing each object. We chose Caltech-101 because it is popular in papers related to object categorization and contains rather easy images for benchmarking. We selected ten different classes from the database to get good view of the performance over different content. Every image is scaled not to exceed 300 pixels in width and height. The classes are *watch, stop_sign, starfish, revolver, euphonium, dollar_bill, car_side, air planes, Motorbikes* and *Faces_easy*.

The Caltech 101 database however has some weaknesses: the objects are practically in a standard pose and scale in the middle of the images and background varies too little in certain categories making it more discriminative than the foreground objects. To make our benchmark process more challenging we will use the randomized version of the Caltech-101 database where we use the same classes but with randomized background, object orientation and locations. Annotations for bounding boxes and outlines are provided.

To experiment our detectors and descriptors with more recent images, we chose to include ImageNet image dataset to our evaluation. ImageNet provides over 100,000 different meaningful concepts and millions of images. However, bounding boxes, outline coordinates, and landmarks for the objects were not provided and we had to mark them manually. We selected *Watch, Sunflower, Pistol, Guitar, Elephant, Camera, Boot, Bird* and *Aeroplane* object categories to be in our testing process.

3.4 Comparing detectors

3.4.1 Annotated bounding boxes and contour points

In our experiment annotations for object bounding boxes and contour points are given for each images. Box coordinates are given as vector of 4 elements describing the top left and bottom right corner of the box. The outlines of objects for Caltech-101 and ImageNet image set are $2 \times n$ matrices containing $n\,(x, y)$-pairs for contour points. Since we are only interest of benchmarking how well detected features found from the objects match to different object from the same category, detected features outside the object contour are discarded. With more challenging randomized Caltech-101 we only used bounding boxes so some background features around the objects will be detected. In Figure 14 is shown the bounding box and the contour of an image from Caltech-101 dataset, all the detected SIFT features and the remaining ones after the elimination process.

3.4.2 Annotated landmarks and affine transform

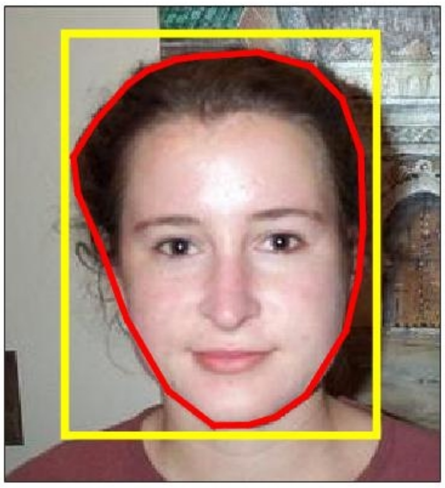
For every image there are manually annotated landmarks (5-12 landmarks per category) which are marked to the sample image locations, which are semantically similar between the same category images. By these landmarks and estimating the pair-wise image transformations using the direct linear transform [47] we can establish affine correspondence between image pairs. In Figure 15 is shown two examples of landmark locations on an object and how well they match with other object's landmarks.

3.4.3 Selected local feature detectors

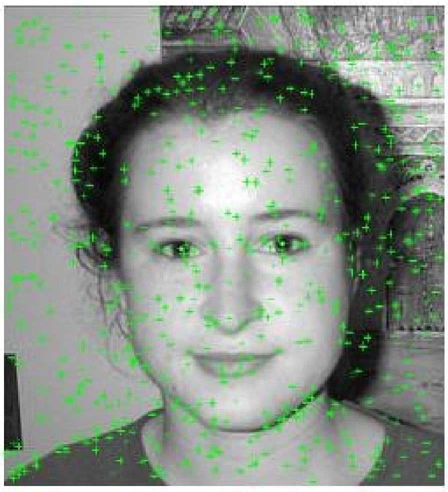
The detectors for the experiment were selected from the earlier study [48] where the performance of different detectors and descriptors were evaluated. Detector evaluations included nine publicly available detectors:

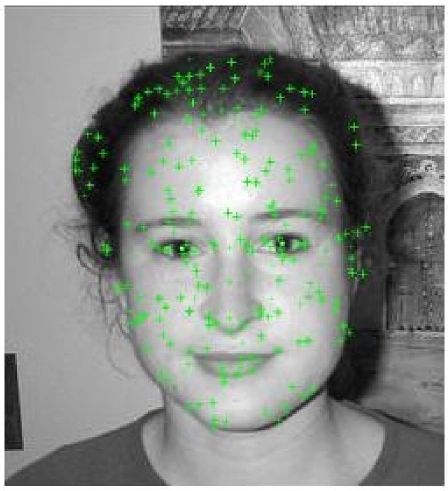
1. Detectors from Zhao's Lip-vireo toolkit [1]: difference of Gaussian (*dog-vireo*), Laplacian of Gaussian (*log-vireo*) and Hessian affine (*hesslap-vireo*).
2. Three implementations by Mikolajczyk[2]: Hessian-affine detectors (*hessaff*) and

[1]http://code.google.com/p/lip-vireo
[2]http://featurespace.org



(a)





(b) (c)

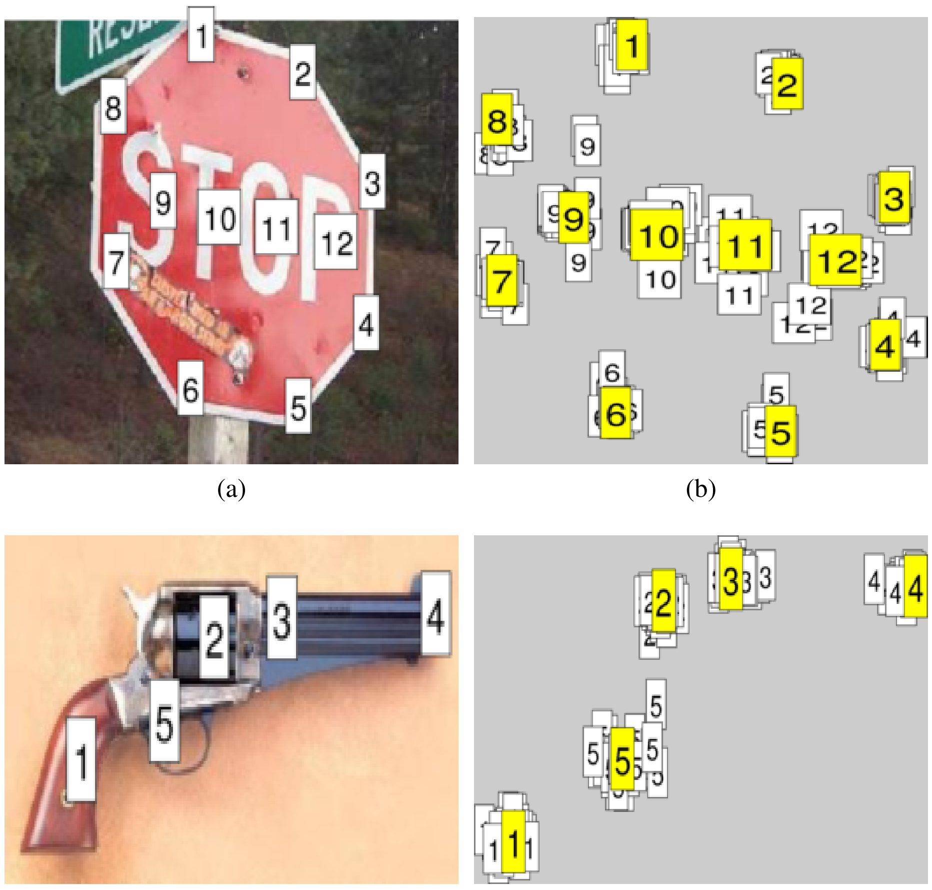
Figure 14. Elimination of the features outside the object: a) the bounding box (yellow line) and the contour (red line) of the face, (b) the detected SIFT features, and c) the remaining features after elimination.

(*hessaff-alt*) and Speeded-up robust feature detector (*surf*).

3. Two detectors from VLFeat toolbox[3]: Maximally stable extremal regions (*mser*) and SIFT (*sift*).

The results indicated that *hesslap-vireo*, *dog-vireo* and *surf* were the best detectors based on the repeatability rate which was over 30% with all the methods. However, the *hesslap-vireo* provided much more correspondences (57) on average compared to for instance the highest repeatability rate (30%) *dog-vireo* which had 16 average on correspondences. Considering the repeatability rate and the average correspondences equally the best detectors were *hesslap-vireo* $(30.6\%, 57.4)$, *hesaff* $(25.4\%, 47.8)$ and *log-vireo* $(26.3\%, 46.5)$.

[3]http://vlfeat.org



(a) (b)

(c) (d) Subfigure 3 caption

Figure 15. Two objects from different classes with annotated landmarks (the leftmost images) and 50 examples (affine) projected into a single space (denoted by the yellow tags).

In this thesis, we report the results for the hesaff detector (*hesaff*) and VLFeat implementation of SIFT.

In additions to these we were interested to evaluate some of the more recent and efficient detectors from [37]: BRIEF [35], BRISK [41] and ORB [36]. The best results were obtained using the ORB OpenCV implementation [4] which results are reported in this thesis. Moreover, dense sampling has replaced detectors in the top methods (Pascal VOC 2011 [49]) and we added the dense SIFT in VLFeat toolbox to our evaluations (*dense*).

3.4.4 Performance measures and evaluation

For the detector performance evaluation, the test protocol is similar to Mikolajczyk benchmark [30] which main points were discussed in section 3.2. Because we are focusing on

[4]http://opencv.org/

object matching more than a general feature similarity measurement, we will make an exception to the protocol that only interest points inside the contour will be selected. For each image pair, points from the first image are projected onto the second image by the affine transformation estimated using the the annotated landmarks described in the section 3.4.2. The interest points (regions) are described by 2D ellipses and if a transformed ellipse overlaps with an ellipse in the second image (more than a selected threshold value) a correct correspondence is recorded. The reported performance numbers are the average number of corresponding regions between image pairs and the total number of detected regions. The detector performs well if the total number of detected regions is high and most of them overlap with the second image corresponding regions. We adopt the parameter setting from [30]: 40% overlap threshold and normalization of the ellipses to the radius of 30 pixels. The normalization is required since the overlap area depends on the size of the ellipses. The algorithm for evaluating detectors is shown below:

Algorithm 1 The detector benchmark procedure

```
 1: for all image pairs do
 2:     Extract local features
 3:     Use foreground masks to filter out features outside the object
 4:     Compute the true transformation matrix $H$ for image pairs
 5:     Transform all detected regions onto the second image using $H$
 6:     for all regions do
 7:         if Overlap is more than a threshold $t$ with some of the regions in the other
    image then
 8:             Increment the correspondency score $c_i$
 9:         end if
10:     end for
11:     Calculate the repeatability rate $r_j$ using $c_i$
12: end for
13: Return the correspondences c and repeatabilities r
```

3.4.5 Results

It is noteworthy that this evaluation differs from the earlier studies [48] in the sense that instead of using the default parameters for each detector we adjusted their meta-parameters to return on average 300 regions for each image. This makes the evaluation fair for all the detectors because of the number of detected regions will surely have an effect to number of correspondences between image pairs. The adjustment of the meta-parameters is discussed more in section 3.4.6. The results of the detector experiment are reported in Table 2 and Figure 16. From the results we can see that the *starfish* and the *revolver* categories were the hardest ones for all the detectors. Performance of dense sampling in *Faces_easy* category is very good: it provides a lot of correspondence regions compared to other methods and the same regions are mostly found in both images.

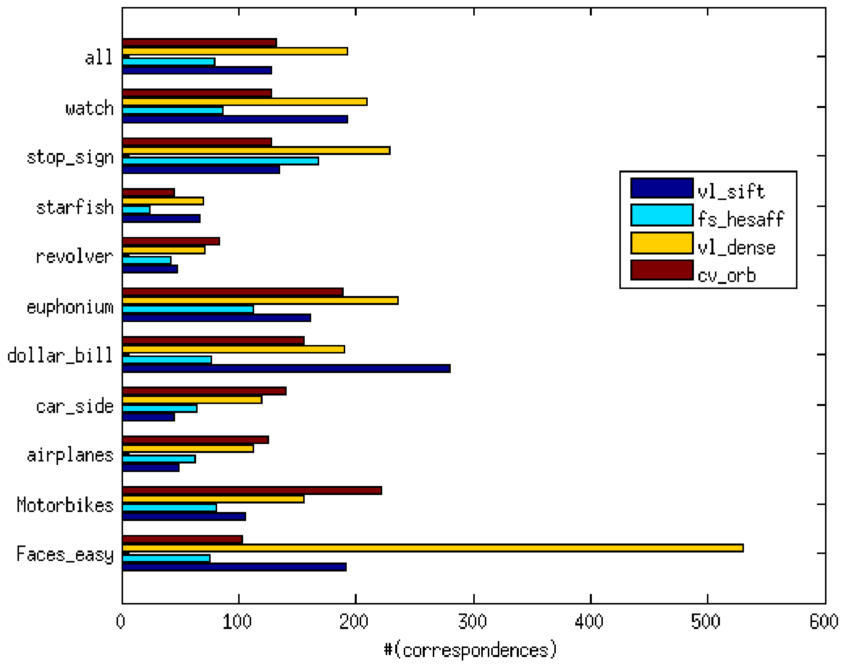
With the adjusted meta-parameters the difference between the detectors is less significant than in the earlier evaluation [48] and the previous winner, Hessian-affine, is now the weakest. With the default parameters Hessian-affine returns almost five times more features than for instance SIFT, which made the evaluations too biased against the other detectors. The original SIFT detector performance without the parameter adjustment would be by order of magnitude worse. The new winner in the detector benchmark is clearly the dense sampling with a clear margin to the next best detector ORB. However, when computational time is crucial, the ORB detector seems attempting due to its speed.

Table 2. The Overall result table of detector evaluation.

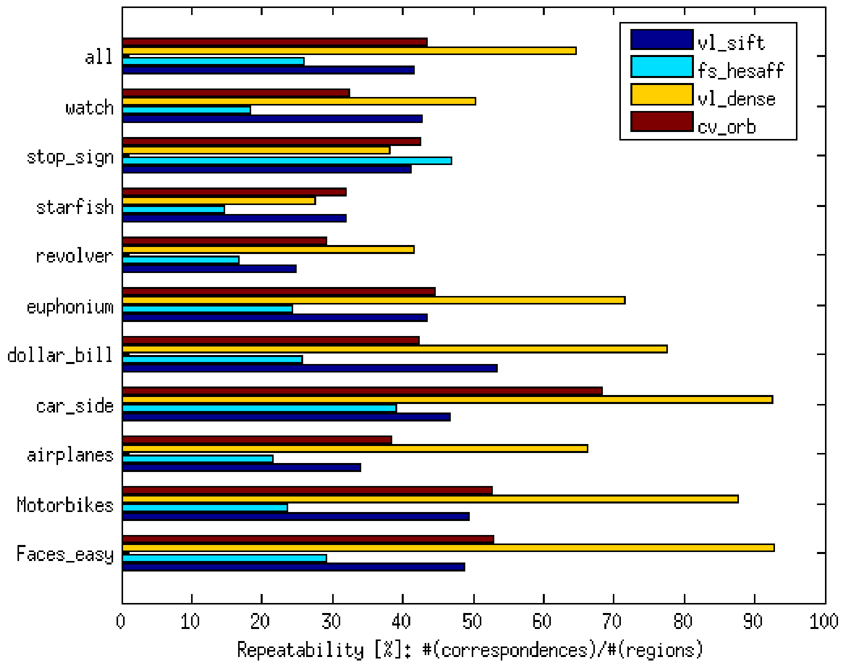
Detector	*Avg # of corr.*	*Avg. rep. rate*
vl_sift	127.5	41.6%
fs_hessaff	79.3	26.0%
cv_orb	132.0	43.5%
vl_dense	192.3	64.6%

3.4.6 Detecting more regions

In the previous section, we adjusted detector meta-parameters to return on average 300 regions for each image. That made detectors produce very similar results while using the default parameters in our previous work lead to completely different interpretation. It is interesting to study whether we can exploit meta-parameters further to increase the number of corresponding regions. For ORB we adjusted the edge threshold, for Hessian-affine the feature density and the Hessian threshold, for SIFT the number of levels per octave, and for the dense the grid step size. We computed the detector repeatability rates as the functions of the number of detection regions and the results are reported in Figure 17. Figure also shows the number of returned regions by default parameters with the black dots. As expected the meta-parameters have almost no effect to the dense detection while Hessian-affine, ORB and especially SIFT clearly improve as the number of the regions increase (SIFT regions saturate to the same locations approximately at 600 detected regions). For the most difficult classes (*starfish* and *revolver*) the performance of the detectors could be increased by a iterative search of the optimal parameters but this could



(a)



(b)

Figure 16. Detector evaluation in object class matching. Meta-parameters were set to return on average 300 regions. (a) average number of corresponding regions and (b) repeatability rates.

compromise the performance of other category classes.

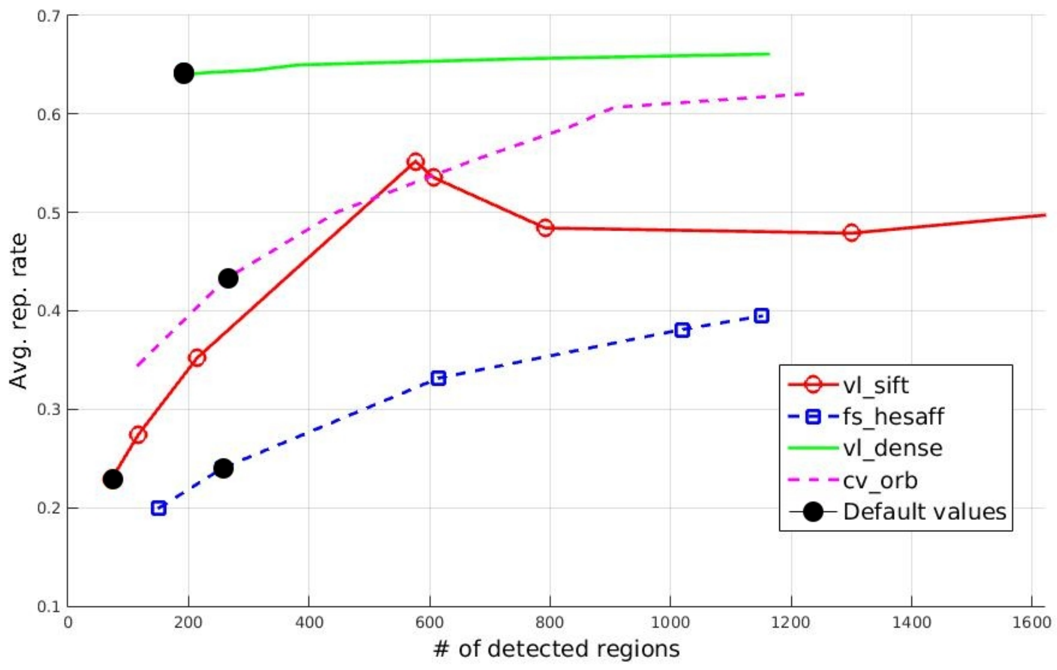


Figure 17. Detector repeatability as the function of the number of detected regions adjusted by the meta-parameters

3.5 Comparing descriptors

A good region descriptor for object matching should be discriminative to match only correct regions, and also tolerate small appearance variation between the examples. These are general requirements for feature extraction in computer vision and image processing. The descriptor performance were obtained in the original work [27] by computing statistics of the correct and false matches. Between different class examples, descriptor matches are expected to be weaker due to increased appearance variation.

3.5.1 Available descriptors

In this experiment we used detector-descriptor pairs. The earlier studies included the following detector-descriptors combinations:

1. Hessian-affine and SIFT

2. Hessian-affine and steerable filters
3. Vireo implementation of Hessian-affine and SIFT
4. Original SIFT detector and SIFT descriptor
5. Alternative (Vireo) implementation of SIFT and SIFT
6. SURF and SURF

Within their default parameters the combinations 1) and 2) utilizing Mikolajczyk's implementation of Hessian-affine detector were clearly superior to other methods [48], but here we adjust the same meta-parameters as described in section 3.4.6 to return the same average number of regions (300).

To these experiments, we also include the best fast detector-descriptor pair: ORB and BRIEF. The following combinations will be reported: *vl_sift+vl_sift* (FeatureSpace implementation), *fs_hessaff+fs_sift* (FeatureSpace implementation), *cv_orb+cv_brief* (OpenCV implementation), *cv_orb+cv_sift* (OpenCV, to compare SIFT and BRIEF), *vl_dense+vl_sift* (VLFeat implementation). It should be noted that available descriptors are not guaranteed to work well with different implementations of detectors. Thus we will use in our evaluation pair-wise detector-descriptor combinations only. Because the chosen detector has also impact on the performance of the descriptor, we wanted to test *vl_sift* descriptor with two different detectors. We also tested the RootSIFT descriptor from [50] which should lead to performance boost of SIFT by using a square root (Hellinger) kernel instead of the standard Euclidean distance. In their experiment this led to a notable performance increase but in our case it provided insignificant difference to the original SIFT (mean: $3.9 \rightarrow 4.2$, median: $1 \rightarrow 1$).

3.5.2 Performance measures and evaluation

In the earlier work [48] they used a simplified version of the Mikolajczyk's descriptor performance measure: the ellipse overlap was replaced by normalized centroid distance of the matching regions. However, the results by the simplified rule turned out to be too optimistic and in this work we adopt the original measure.

The rule is the same as with the detectors, if the best matching regions have sufficient overlap the match is correct. Descriptors are computed for all detected regions (foreground only). Images are processed pair-wise and the best match for each region is selected from the full distance matrix. It is worth noting that the rule proposed in [32] for discarding "bad regions" (ratio between the first and the second best is less than 1.5) is not used

Algorithm 2 Descriptor performance evaluation procedure

```
1:  for all image pairs do
2:      Extract local features
3:      Use foreground masks to filter out features outside the object
4:      Compute the true transformation matrix $H$ for image pair
5:      Transform all detected regions onto the second image using $H$
6:      for all regions do
7:          Find the $N$ closest descriptors associated with the region
8:          if region overlap is more than a threshold $t$ then
9:              Increase the number of matches for $m_i$
10:         end if
11:     end for
12: end for
13: Return the total number of matches $\mathbf{m}$
```

since it results complete failure. We used the default ellipse overlap threshold 50% from [27] which is little bit looser than in detector evaluation, but also more strict threshold were tested. Our performance numbers are the average number of matches and median number of matches. In [27] recall versus 1-precision was used for quantitative evaluation of descriptor matching. Our case however is different to the wide baseline matching where image pairs are tested one by one. We decided to report the average number of matches that is more compact and intuitive for a large set of image pairs containing category instances and the average numbers of matches were also reported by Mikolajczyk and Schmid. The Algorithm 2 also provides an alternative measure, *coverage*, which measures the number of images "covered" with at least the given number of correspondences N (coverage N). In the detector evaluation the mean and median numbers were almost the same, but here we report the both since for the descriptors there is significant discrepancy between the mean and median numbers.

3.5.3 Results

The average number of matches for the descriptor evaluation is shown in Figure 18 and the overall results including the median, the results for more strict overlap, and computation time are reported in Table 3. For many classes, the mean and median numbers are very low, and for instance the starfish category is extremely hard for every descriptor. However, with dense grid sampling coupled with SIFT we get decent results for most of the categories and its performance is superior compared to all other descriptors, achieving the average of 23.0% matches per class and median of 10.0% matches. The second best descriptor is FeatureSpace implementation of *fs_hessaf*+*fs_sift* and the rest of the descriptors are near behind with minor performance decrease. The more strict overlaps, 60% and 70%, provide almost the same numbers verifying that the matched regions do match well also spatially.

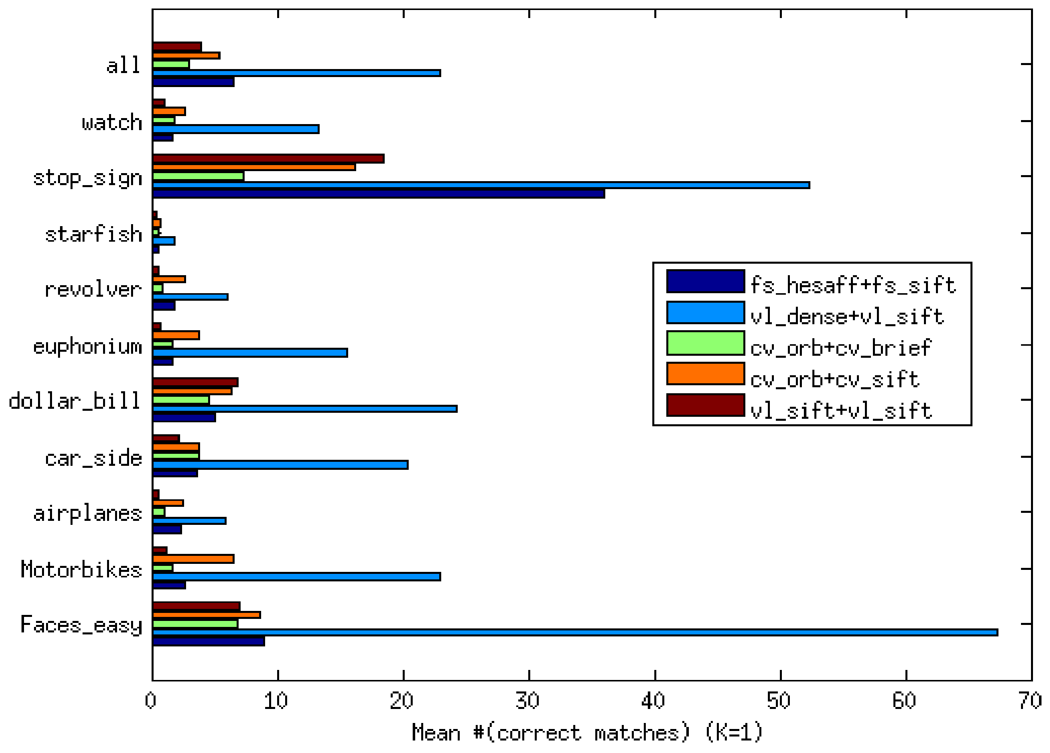


Figure 18. The average number of matches per class in descriptor evaluation when using the default threshold overlap 50%.

Table 3. Overall result table of description evaluation. The default overlap threshold is 50% [27], 60% and 70% results demonstrate the effect of the more strict overlaps. The computation time are average detector and descriptor computation times for one image pair.

Detector+descriptor	*Avg #*	*Med #*	*Avg # (60%)*	*(70%)*	*Comp. time (s.)*
vl_sift+vl_sift	3.9	1	2.8	1.6	0.15
fs_hessaff+fs_sift	6.5	2	5.9	4.9	0.22
vl_dense+vl_sift	23.0	10	22.3	20.2	0.76
cv_orb+cv_brief	3.0	1	2.9	2.7	0.11
cv_orb+cv_sift	5.4	2	4.8	4.1	0.37

The best results were obtained for the stop signs, dollar bills and faces, but the overall performance is poor. The best discriminative methods could still learn to detect these categories, but it is difficult to imagine naturally emerging "common codes" for other classes except the three easiest. It is surprising that the best detectors, Hessian-affine and dense sampling, were able to provide 79 and 192 repeatable regions on average, but only roughly 10% of these match in the descriptor space. Despite the fact that the SIFT detector performed well in the detector experiment, its region do not match well in this experiment. The main conclusion is that the descriptors that are developed for wide baseline matching do not work well in the matching regions between different class examples.

3.5.4 The more the merrier

Similar to Section 3.4.6 we studied how the average number of matches behaves as the function of the number extracted regions. This is justified as some work [34] claim that the number of interest points extracted from the test images is the single most influential parameter governing the performance. The result graph is shown in Figure 19.

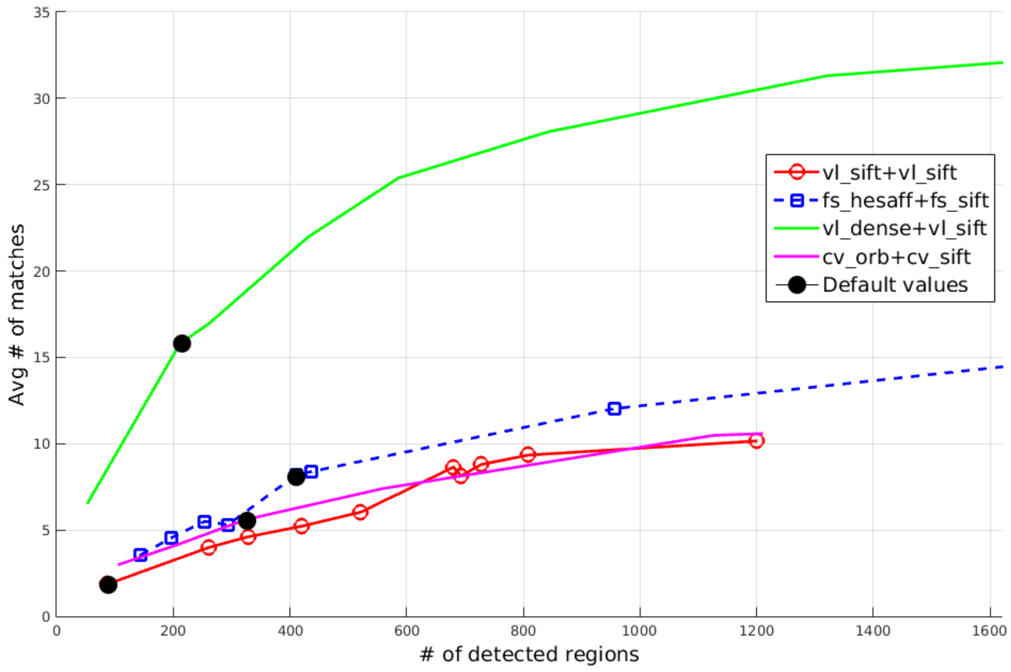


Figure 19. Descriptors' matches as function of the number of detected regions controlled by the meta-parameters (default values denoted by black dots).

The results show that adding more regions by adjusting the detector meta-parameters provides only minor improvement to the average number of matches. Clearly, the "best regions" are provided first and the dense sampling performs much better indicating that what is "interesting" for the detector is not necessarily a good object part.

3.6 Advanced analysis

In this section, we address the open questions raised during the detector and descriptor comparisons in Section 3.4 and 3.5. The important questions are: why only a few matches are found between different class examples and what can be done to improve that? Why dense sampling outperforms all interest point detectors and does it have any drawbacks? Do our results generalize to other data sets.

3.6.1 ImageNet classes

To validate our results, we selected 10 different categories from the state-of-the-art object detection database: ImageNet [46]. The configuration set up was the same as in the section 3.5: the images were scaled to the same size as the Caltech-101 images, the foreground areas were annotated and the same overlap threshold values were tested. The results for the ImageNet classes are reported in Figure 20 and overall results in Table 4. The average number of matches is roughly half of the number of matches with Caltech-101 images which can be explained by the fact that the data set is more challenging due to 3D view point changes. However, the ranking of the methods is almost the same: dense sampling and SIFT is the best combination and the SIFT detector and descriptor pair is the worst. The results validate our findings with Caltech-101.

Table 4. Overall result table of description evaluation using ImageNet dataset. The default overlap threshold is 50% [27], 60% and 70% results demonstrated the effect of the more strict overlaps.

Detector+descriptor	*Avg #*	*Med #*	*Avg # (60%)*	*(70%)*
vl_sift+vl_sift	1.2	0	0.7	0.3
fs_hessaff+fs_sift	3.4	2	2.8	1.9
vl_dense+vl_sift	12.4	7	11.6	10.2
cv_orb+cv_brief	2.2	1	1.9	1.5
cv_orb+cv_sift	3.9	2	3.3	2.5

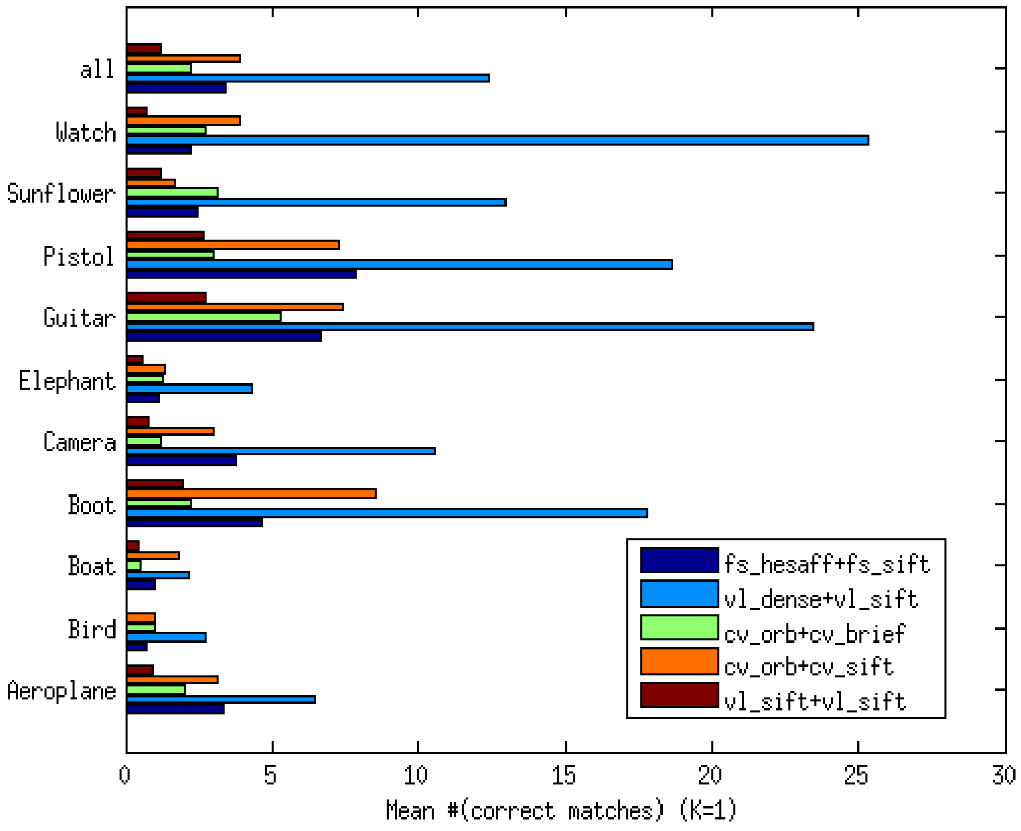


Figure 20. Average number of matches per class using ImageNet database images (overlap threshold 50%).

3.6.2 Beyond the single best match

In object matching, assigning each descriptor to several best matches, "soft assignment" [51, 52, 53], provides improvement and we want to experimentally verify this finding using our framework. The hypothesis is that the best matches in descriptor space are not always correct between two image pairs, and thus, not only the best, but a few best matches can be used. This was tested by counting matches as correct if they were within the K best and the overlap error was under the threshold. To measure the effect of multiple assignments, we establish a new performance measure: *coverage*. Coverage corresponds to the number of image pairs for which at least N matches have been found (*coverage-N*) and this measure is more meaningful than the average number of matches since there was strong discrepancies between the average and median numbers. We tested the multiple assignment procedure by accumulating matches over $n = 1, 2, ..., K$ best matches. The corresponding coverage for $K = 1, 5, 10$ are shown in Figure 21 and Table 5. Obviously, more image pairs contain at least $N = 5$ than $N = 10$ matches. With $K = 1$ (only the

Table 5. Average number of image pairs for which $N = 5, 10$ matches were found using $K = 1, 5, 10$ nearest neighbors.

	Coverage-($N = 5$)			*Coverage-($N = 10$)*		
Detector+descriptor	*K=1*	*K=5*	*K=10*	*K=1*	*K=5*	*K=10*
cv_orb+cv_sift	7.9	16.7	23.0	3.6	11.1	15.7
vl_dense+vl_sift	16.0	19.5	19.8	12.9	18.1	19.6
cv_orb+cv_brief	4.5	13.3	17.9	2.1	9.5	13.2
fs_hesaff+fs_sift	7.3	17.9	20.4	3.5	12.7	17.7
vl_sift+vl_sift	4.3	8.0	11.3	2.5	4.3	6.0

best match) the best method, VLFeat dense SIFT, finds at least $N = 5$ matches in 16 out of 25 image pairs and 13 for $N = 10$. When the number of best matches is increased to $K = 5$, the same numbers are 19 and 18, respectively, showing clear improvement. Beyond $K = 5$ the positive effect diminishes and also the difference between the methods is less significant.

3.6.3 Different implementations of the dense SIFT

During the course of work, we noticed that different implementations of the same method provided slightly different results. Since there are two popular implementations of dense sampling with the SIFT descriptor, OpenCV and VLFeat (two options: slow and fast), we compared them. The result corresponding to the previous experiments in section 3.5 are shown in Figure 22. There are slight differences in class results due to implementation differences, but the overall performances are almost equal. However, the computation time of the VLFeat implementation is much smaller compared to the OpenCV. In addition, the VLFeat fast version is roughly six times faster than the slower version of SIFT.

3.6.4 Challenging dense sampling: r-Caltech-101

With dense sampling the main concern is its robustness to changes in scale and, in particular, orientation, since these are not estimated similar to interest point detection methods. In this experiment, we replicated the previous experiments with the two dense sampling implementations and the best interest point detection method using the randomized version of the Caltech-101 data set: r-Caltech-101 [45]. R-Caltech-101 contains the same objects (foreground), but with varying random Google backgrounds and the objects have been translated, rotated and scaled randomly. These are illustrated in Figure 23. An ex-

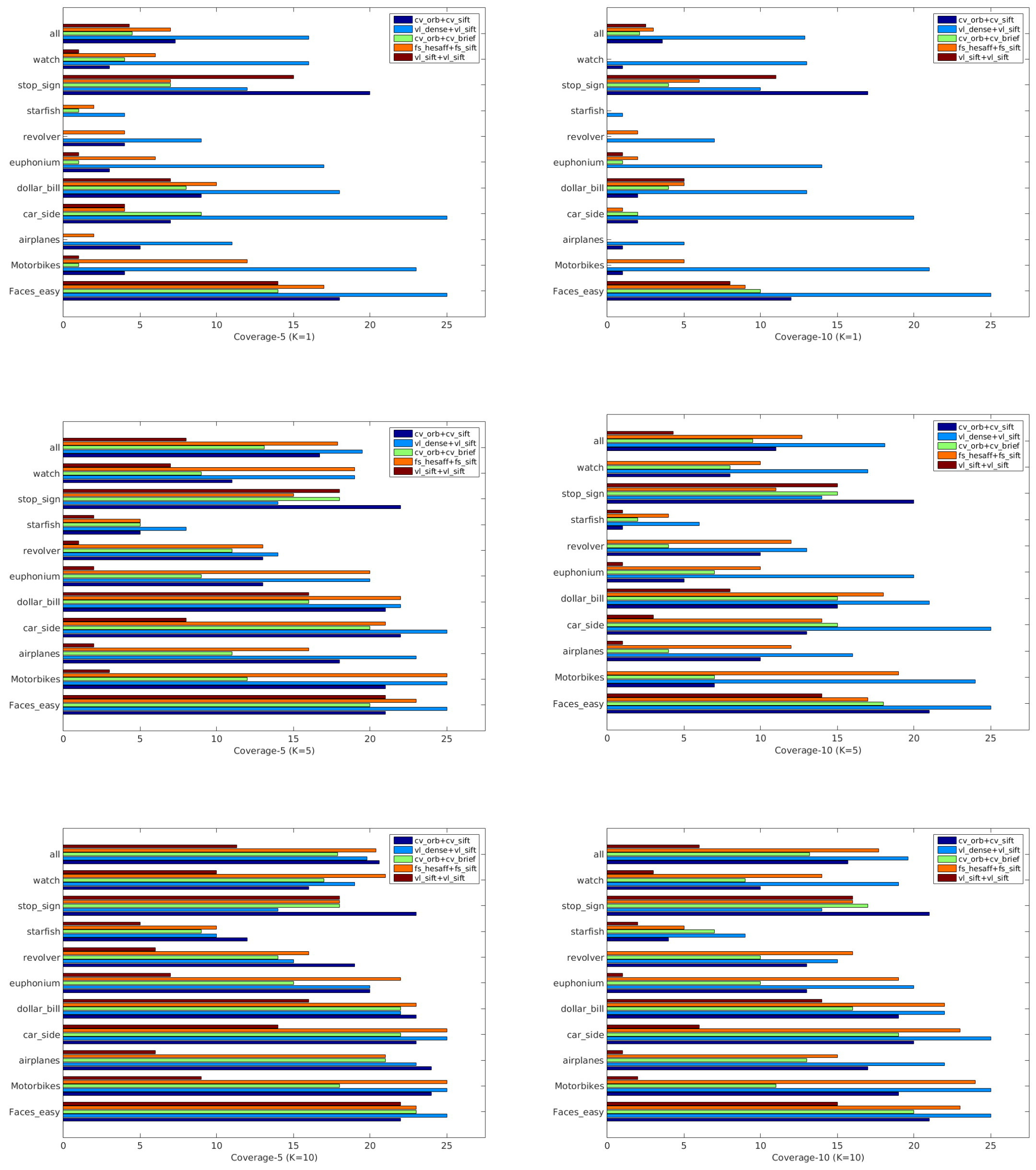


Figure 21. Number of image pairs for which at least $N = 5, 10$ (left column, right column) descriptor matches were found (*Coverage-N*). $K = 1, 5, 10$ denotes the number of best matches (nearest neighbours) counted in matching (top-down).

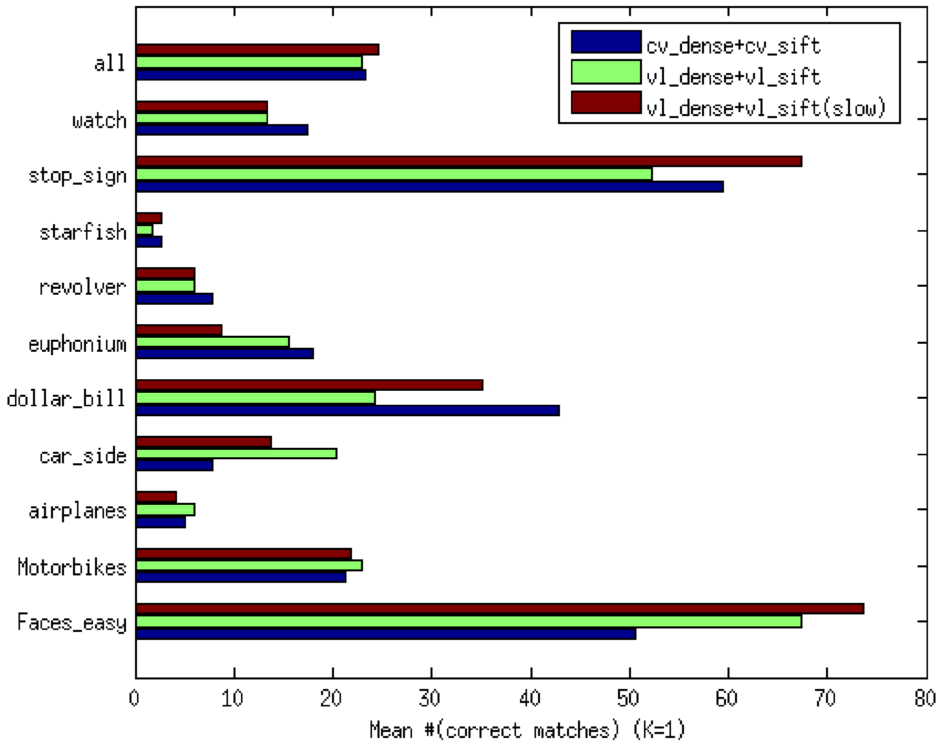


Figure 22. Average number of matches per class using ImageNet database images (overlap threshold 50%) and different dense SIFT implementations.

ception to the previous experiments is that we discard features outside the bounding boxes instead of using the more detailed object contour. The detector and descriptor results of this experiment are reported in Figure 24. Now it is clear that artificial rotations affect to the dense descriptors while Hessian-affine is unaffected (actually improves). It is noteworthy that the generated pose changes in r-Caltech-101 are rather small ($[-20°, +20°]$) and the performance drop could be more dramatic with larger variation. An intriguing research direction is detection of scaling and rotation invariant dense interest points.

3.7 Discussion

In this chapter, the well accepted and highly cited interest point detector and descriptor performance measures by Mikolajczyk *et. al* [30, 27], the repeatability and number of matches, were extended to measure intra-class performance with visual object categories. The most popular state-of-the-art and more recent efficient detectors and descriptors were compared using the Caltech-101, r-Caltech-101 and ImageNet image datasets. Interest points and regions have been the low-level features in visual class detection and classi-

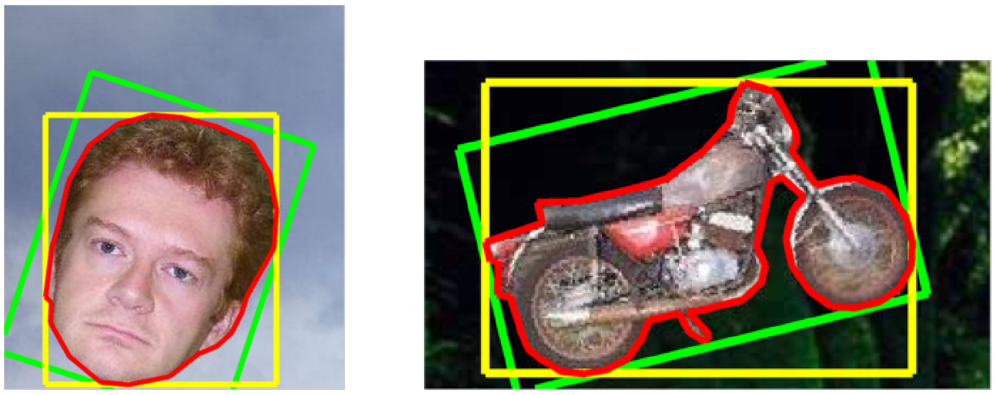


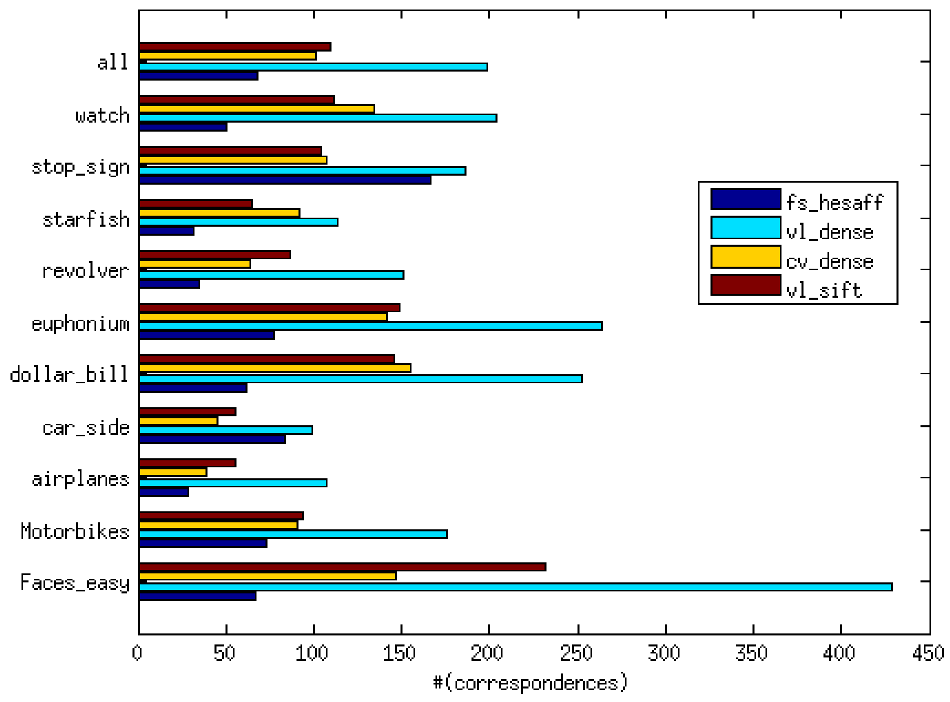
Figure 23. The r-Caltech-101 versions of the original Caltech-101 images (original bounding box shown by green).

fication for a decade [10]. Recently, supervised low-level features, such as convolution filters in deep neural networks [12], have gained momentum, but we believe that the unsupervised detector-descriptor approach can be developed further by identifying and improving bottlenecks.

With our proposed framework we identified that dense sampling outperforms interest point detectors with a clear margin. It is the most reliable in the terms of repeatability rate and it also has the highest number of correspondences between image pairs. One of the most interesting finding was the number of detected features' relationship to the detection performance. The earlier winner Hessian-affine was surprisingly the weakest detector because of the adjustment of meta-parameters. The descriptor experiment showed that the original SIFT is the best descriptor including the recent fast descriptors. The descriptor experiment also showed that the choice of the detector which will be paired with the descriptor has a large impact to the results.

Generally, the detectors performed well, but descriptors' ability to match parts over visual class examples collapse. Also it is noteworthy to say that despite the fact that dense sampling performed well in the general evaluations, the method is fragile to object pose variation, while the Hessian-affine is the most robust against pose variations. With an alternative performance measure, *coverage-N*, it was shown that all the methods converge to perform equally and using multiple, even a few, best matches instead of the single best match provides significant performance boost.

The findings advocate new research on i) optimization of the detector meta-parameters for each visual class, ii) specialized descriptors for visual class parts and regions, iii) dense scaling and rotation invariant interest points, and iv) alternative matching methods for multiple best matches. Some results already exist. For example, BoW codebook de-



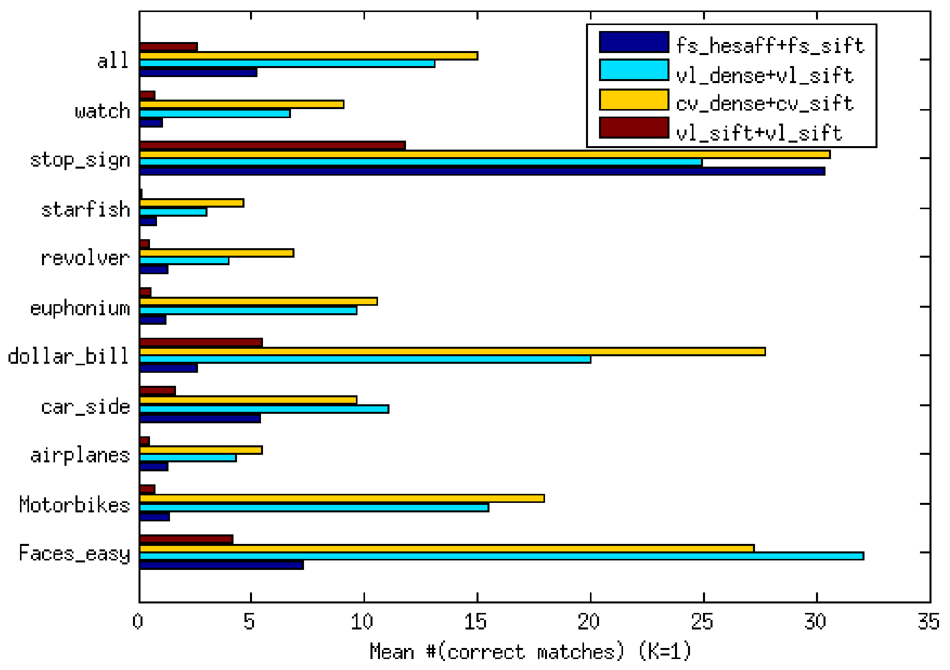


Figure 24. R-Caltech-101: detector (top) and descriptor (bottom). The detector results are almost equivalent to Fig. 16. In the descriptor benchmark (cf. with Fig.18) the Hessian-affine performs better (mean: 3.4 $\rightarrow$ 5.2) while both dense implementations, VLFeat (23.0 $\rightarrow$ 13.1) and OpenCV (23.3 $\rightarrow$ 15.0) are severely affected.

scriptors can be enhanced by merging descriptors based on co-location and co-activation clustering [54] or by learning [55], dense interest points have been proposed [33], and soft-assignment (to have less detected features, but assign each local feature to multiple best matches or codes) has been shown to improve BoW codebook matching [51]. Moreover the success of the standard SIFT in our experiments justifies further development of more effective visual class descriptors. Investigating these potential research directions benefits from our evaluation framework that can be used for automatic validation and optimization.

4 VIDEO SCENE BOUNDARY DETECTION USING BAG-OF-WORDS

In this chapter, the local features are adopted in the domain of video scene boundary detection, which is the second main application in this thesis. The shot and scene level boundary detection is used usually as a preprocessing step before higher level processing, such as video summarization [56] and content based retrieval [10]. In the following sections, a brief introduction to previous shot boundary detection methods is given and detection based on visual Bag-of-Words is introduced. An experimental Bag-of-Words approach is tested in experiments on scene boundary detection in the next chapter.

4.1 Introduction

In the recent years, development of software, cameras (especially small cameras such as mobile or action cameras) has enabled creation of a large amount of digital video content. Services like Youtube[5] and Vimeo[6] provide ways for people to upload their own home videos for everyone to see, but they lack possibilities for automatic video editing. Video applications, which are considerably growing, have initiated growing demand for innovative technologies and tools for indexing, browsing, summarization, and retrieval of video data. Automatic video tools do not necessary provide anything that humans can not do, but the tools can save us a lot of time. For instance one can ask an application to search shots where a certain object appears. Another useful applications would be detection of sudden content variation, which are already well exploited in surveillance systems [57].

In order to automate the indexing, retrieval, and management of videos, a great deal of research has been done on content-based video retrieval over the last decade [58, 15, 14]. Structural analysis of the video is usually the first processing step and thus the performance of the detection algorithm has a big impact to the outcome of the whole pipeline of the process. Among the various structural levels (i.e., frame, shot, scene, etc.), shot level organization has been regarded suitable for browsing and content-based retrieval [59]. In simple terms idea of boundary detection is to find all the possible positions from a video where consecutive shots or scenes have different visual content.

[5]http://www.youtube.com/
[6]http://www.vimeo.com/

4.2 Hierarchy of video

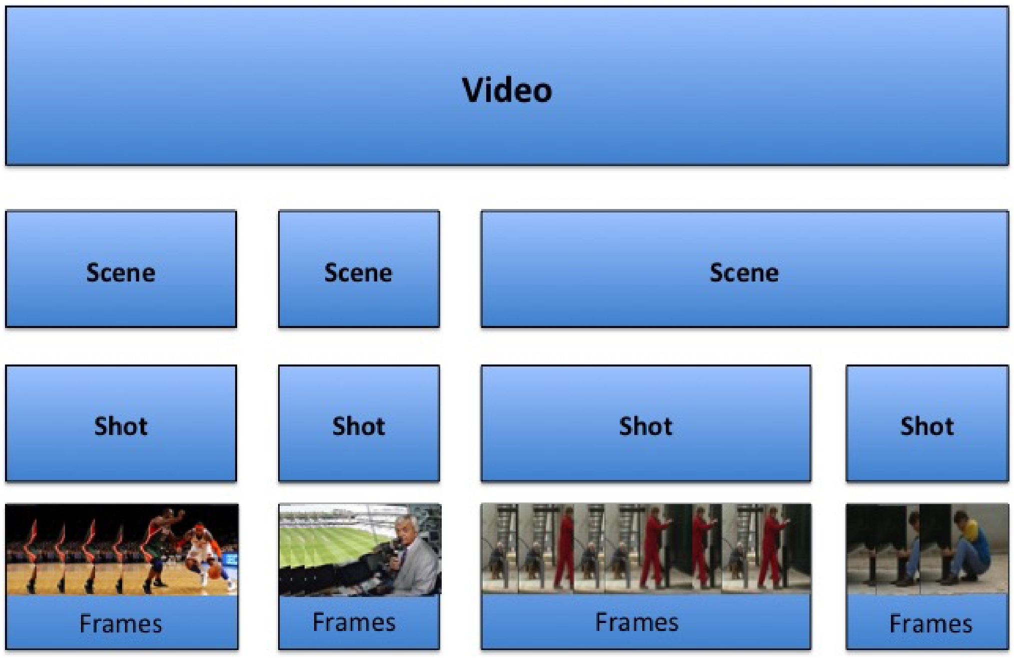
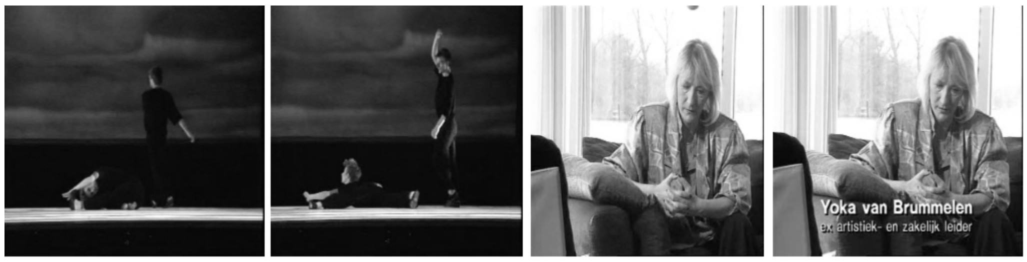


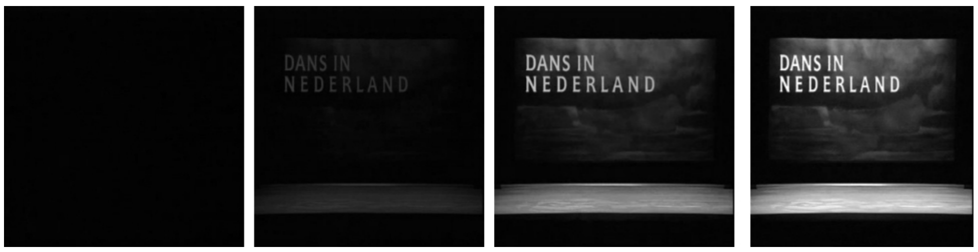
Figure 25. Hierarchical compositional structure of a video.

In Figure 25 is illustrated the composition of the video. A video is a stream of frames, usually combined with an audio stream. When frames are frequently changing in a speed of 20-30 frames per second, it will give us an illustration of a moving picture. The number of images displayed in a second is called the *frame rate* or *frames per second* (*fps*). A typical frame rate of a video is 25 fps. A higher frame rate indicates more accurately flowing motion but as a drawback it requires more storing capacity. An edited video is composed of multiple *scenes*, which are separated from each other by a change of a time or location. A video *shot* is defined as a sequence of frames captured by one camera in a single continuous action in time and space [60]. A scene is typically an event where for instance two people are having a conversation. Now two different shots here could be a camera movement or zooming from one person to another. Different scenes are usually easier to detect than different shots due to bigger variations in the visual content.

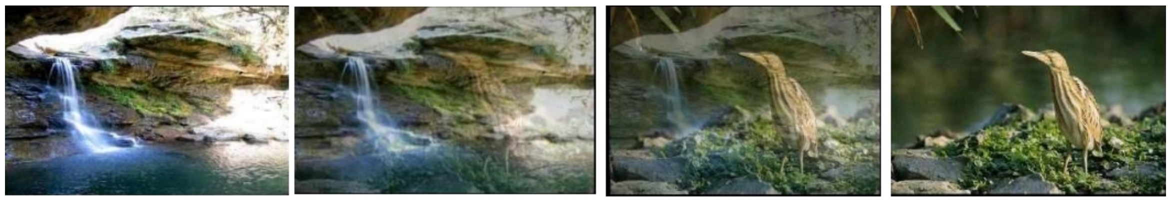
Adjacent scenes can be separated from each other by an abrupt transition (hard cut) or a gradual transition (soft cut) as shown in Figure 26. In an abrupt transition a frame n from the scene k is immediately followed by a frame $n+1$ from the scene $k+1$. In a gradual transition scenes or shots k and $k+1$ are concatenated in a milder manner such as fadeout/fade-in, dissolve, geometrical transition (zooming out or in, camera movement etc.), or artistic transition (computer effects). Research for detecting hard cuts have yield good results but detecting different gradual transition classes have been much more challenging task [22].



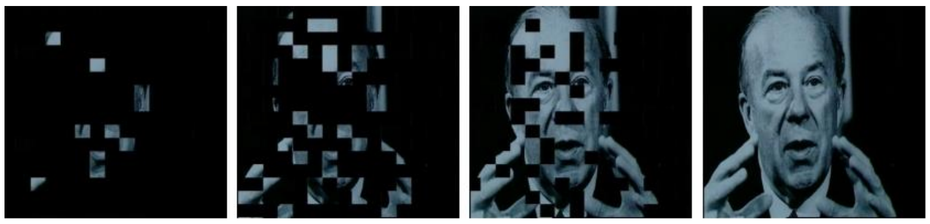
(a)



(b)



(c)



(d)

Figure 26. Examples of transitions: a) hard cut is an instant transition from one scene to the next with a clear cut point, while a soft cut as for example b) fade-in, c) dissolve or d) an artistic transition, where the scene changes gradually and the exact cut point is harder to determine.

4.3 Scene boundary detection

The basis of the scene boundary detection is to detect visual discontinuities along the time domain. In the detection process, it is necessary to extract visual features that measure the degree of similarity between frames. The measure, denoted as $d(n, n+k)$, is related to the difference or discontinuity between a frame or a frame set n and $n+k$ where $k \geq 1$. Then the difference value between n and $n + k$ is compared against a threshold T and if the the threshold is exceeded, a scene boundary between n and $n + k$ is detected. To be able to draw reliable conclusion about the presence or absence of a scene boundary between n and $n + k$, we have to use methods that are as discriminating as possible. This means that our method should report us a clear separation between measurements performed *within scenes* and *at scene boundaries*. The problem of separating a true scene boundary from dissimilarity peaks within the scene is illustrated in Figure 27. In the scene 2 (in Figure 27) when using a loose threshold value there can exists multiple scene boundaries due to high amount of dissimilarity peaks within the scene. On the other hand, using a more strict threshold value, the boundary between scenes 3 and 4 will be missed. Varying values $d(n, n + k)$ inside a single scene make it difficult to decide about the presence or absence of scene boundaries without detection mistakes, i.e., *missed* or *falsely detected* boundaries. There is a large selection of different methods for computing the value of $d(n, n + k)$ and these methods are discussed in more detail in the next section. The most common way to compare frames or frame sets is to use histogram based methods with some distance-metric which calculates the content difference between n and $n + k$.

4.4 Previous work

One of the first published studies about different shot boundary detection techniques were by Boreczky *et. al* [14] and Rui *et. al* [61], including such as full image pixel differences, statistical differences, histograms, edge tracking, motion vectors and feature-based methods. These methods can be combined to get multiple feature information about the video.

The pixel difference method compares pixel difference between consecutive frames by:

$$D(\mathbf{x}) = I_i(\mathbf{x}) - I_{i+1}(\mathbf{x}), \tag{16}$$

where D is the difference value, I_i and I_{i+1} are consecutive frames, and $\mathbf{x} = (x, y)$. The method declares two frames to be different if the pixel difference value $D(\mathbf{x})$ between

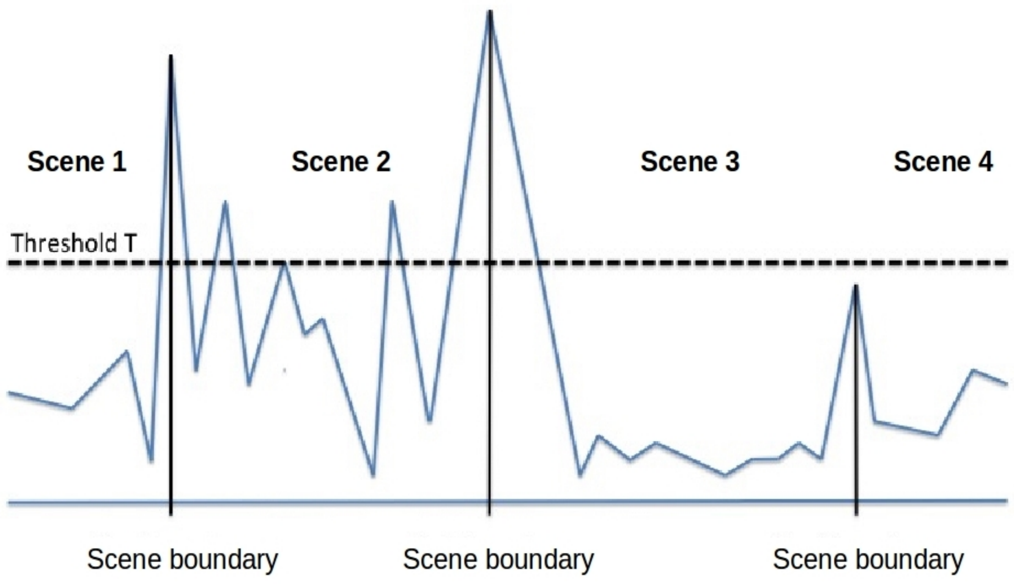


Figure 27. The problem of finding the true cut boundaries by thresholding distance values $d(n, n+k)$.

consecutive frames is above threshold value T_1 and the number of pixels above T_1 exceed some another threshold value T_2 . Pixel-based methods are very sensitive to motion and object movement in some direction which causes large pixel differences. Thus some kind of motion compensation should be used with the method. The statistical approach extends the idea of the previous method by breaking the images into regions and comparing statistical measures of the pixels in those regions. This compensates the camera motion and noise to certain degree.

In the histogram method, we make a vector where each entry stores the number of certain pixel values in the given image. Consecutive frames' histogram dissimilarity is measured and then compared against a threshold value. A typical method is to use colour histogram, such as RGB histogram, which assumes that color content of various scenes is different. The colour histogram distance is calculated as:

$$D = \sum_{r}^{2^B-1} \sum_{g}^{2^B-1} \sum_{b}^{2^B-1} (I_i(r,g,b) - I_{i+1}(r,g,b)), \tag{17}$$

where r, g, b are color components. Each color component is quantized to 2^B different values. Small values of B (2 or 3) [15] are already good to make the method rather robust to noise, lighting and view point changes while having a small memory storage requirement. Drawback of the color histogram is that it does not take into account the space information concerning the color. For instance, an image containing a red flower surrounded by green grass will have a similar RGB histogram with an image containing a person with red shirt playing golf in a golf course.

In edge tracking shot boundaries were detected by looking for large edge change percentages (percentage of edges that enter and exit between the two frames). The edge change ratio (ECR) is defined as:

$$ECR_n = max(X_n^{in}/\sigma_n, X_{n-1}^{out}/\sigma_{n-1}), \tag{18}$$

where σ_n is the number of edge pixels in frame n, X_n^{in} and X_{n-1}^{out} the number of entering and exiting edge pixels in frames n and $n-1$ respectively. By discarding edges that appear in one image which have edge pixels nearby in the other image we can tolerate some object motions. In addition, edge tracking should provide distinguish patterns between different type of transitions in a video.

Motion vectors have been used successfully for detecting camera zoom or pan from the scene. The block motion vectors can be extracted from the MPEG compressed video, but the selected features are usually inappropriate for image processing purpose. Boreczky *et. al* [14] came to the conclusion that simpler algorithms outperformed more complicated ones, because the complicated algorithms' parameters had to be tuned and the algorithms were sensitive to the threshold value. Also combining multiple methods could provide more accurate results.

Local features have been used in video shot boundary detection. Local features computed from an image are usually feature vectors, and the scene change is detected by computing vector dissimilarity between consecutive frames. The approach somehow overlaps with previously introduced methods because feature vector can represent a histogram. Baber *et. al* [62] used SURF features in shot boundary detection and the method was found robust to various transformations (such as camera and object motion), but its performance suffered in extreme light or dark scenes. Li *et. al* [63] introduced a local feature based SIFT shot boundary detector which utilize SVM after feature extraction to match local features in consecutive frames. Luo *et. al* [64] presented another approach which used the Kernel PCA and clustering based feature extraction coupled with the SIFT Flow algorithm. Many authors [65, 66, 67, 62] have also presented a moving window approach where the cut is usually determined from pre- and post-frames related to the current frame.

4.5 Visual Bag-of-Words

Bag-of-Words (BoW) or Bag-of-Features (BoF) is a popular visual descriptor used for visual data classification. In general, Bag-of-Words is a sparse vector of occurrence counts of words; that is, a sparse histogram over the vocabulary. In a computer vision related task, a bag of visual words of features is a sparse vector of occurrence counts of a vocabulary of local image features. Publications in computer vision uses a codebook and a codeword terms instead of vocabulary and word. In this thesis we adopt them as well.

The seminal works of the visual Bag-of-Words (BoW) are presented in [10] and [9]. The codebook generation and utilization with the visual Bag-of-Words in visual object categorization can be seen in Figure 28. In general, the process with visual Bag-of-Words can be separated into two parts: 1) obtaining the visual codebook and 2) building the code histogram. In the first part of BoW, the salient local image features (interest points) are first extracted with a special detector (e.g. SIFT) or fixed size patches are selected using dense sampling on a regular grid. Then, these keypoints are described with a descriptor, the SIFT descriptor being the most popular one. Finally, a codebook is generated by clustering the descriptors into a fixed number of codes, typically by the k-means algorithm [3]. In the second part, after the feature extraction, an image feature (code histogram) is generated by computing the histogram of the codes appearing in the image. This is done by assigning every descriptor from the image to the closest codeword in the codebook. Now, the matching between two images (frames), can be performed by calculating the histogram similarity.

The scene boundary detection methods using local feature histograms is only a few. Li *et al.* [63] computed SIFT regions and descriptors and [62] used SURF descriptors to detect cuts, but both of them did not utilize a codebook. They searched the matches directly between consecutive frames or within a query window. Similar approaches which utilize a codebook were proposed by Sivic and Zisserman [10] and Lankinen and Kämäräinen [69]. Sivic and Zisserman, and also Li approaches however are extremely time consuming due to random sampling based matching. Sivic and Zisserman run the matching only for key frames of every shot as their application was content retrieval and Li *et al.* [63] did not report the computation time for their method. Lankinen and Kämäräinen [69] and Sivic and Zisserman [10] built a statistic visual vocabulary from a big subset of frames (about $5 - 10\%$ of the entire frame count) from the given video. Thus building the codebook in the beginning of the processing will be very time consuming and will be devastating property concerning any online applications.

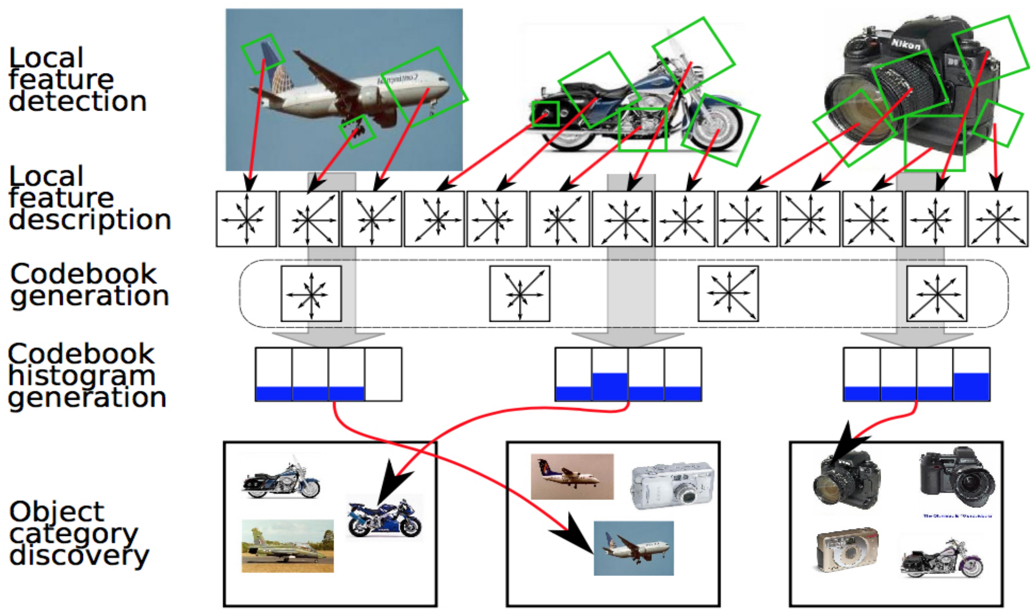

Local feature detection
Local feature description
Codebook generation
Codebook histogram generation
Object category discovery

Figure 28. The visual object classification with Bag-of-Words.[68]

In the next chapter, we present an experimental BoW method, which will use a dynamic visual codebook. Idea is to develop a novel initial step for a parallel video editing system where an end user can manipulate a given video simultaneously while the system is still processing the video incrementally. The codebook will be initialized with a small amount of features in the beginning of the processing and the codebook is re-estimated when a cut is detected.

5 EXPERIMENTS ON DYNAMIC BAG-OF-WORDS FOR SCENE BOUNDARY DETECTION

In the previous chapter a brief introduction to Bag-of-Words approach was given and the commonly used video shot boundary detection algorithms were described. In this chapter, a novel BoW method is introduced for processing massive amounts of frames from a video: dense sampling and SIFT descriptor for feature detection and representation, k-mean clustering for codebook generation, a brute-force matcher to find closest codewords for descriptors, L1-normalization of codebook histogram, and the Euclidean distance for code histogram matching. The method is evaluated using the TRECVid 2007 dataset. Our main target is to present a customer-level scene boundary detector which is useful in online applications, such as being a pre-processing step in an online video summarization software. We evaluate our method with different codebook sizes, which is one of the most important parameters in BoW categorisation. The algorithm processes a given video by moving query window and in the experiments, we will use different window sizes and study the window size relation to the cut accuracy. In addition, we tune our algorithm to work as fast as possible by searching the minimum feature number that is needed to build the codebook and the code histogram. In the experiments recall and precision curve is our choice of accuracy metric. Finally the presented method is compared against the similar approach by Lankinen and Kämäräinen [69] which uses a static codebook.

5.1 TRECVid 2007 dataset

The TREC Video Retrieval Evaluation[7] (TRECVid) 2007 represents the seventh running of a TREC-style video retrieval evaluation, which goal is to encourage research in information retrieval by providing a large test collection via open, metrics-based evaluation. The TRECVid consist of The Netherlands Institute for Sound and Vision (NIST) gathered 400 hours of news magazine, science news, news reports, documentaries, educational programming and archival video in MPEG-1 format. TRECVid also provides a forum for the organizations for comparing results and for submitting the results of the experiments back to the coordinator. NIST collects all the submitted results for automatic evaluation and eventually the overall results are made publicly available on the TRECVid webpage.

TRECVid 2007 uses approximately 100 hours of data from the whole TRECVid archive and the test data was the first time distributed via the Internet. The TRECVid 2007 eval-

[7]http://trecvid.nist.gov/

uation was divided into three different categories: shot boundary detection, high-level feature extraction, and search task. The data was divided as follows:

- 6 hours for the shot boundary task
- 50 hours for development of search/feature detection
- 50 hours for test of search/feature detection

The shot boundary task is included in TRECVid as an introductory problem, the output of which is needed for most higher-level tasks. The number of videos for the shot boundary detection task in TRECVid 2007 is 17, containing 6 hour of video and approximately 4.4 GB of data [22]. The test videos contained a total of 637 805 frames, having 25 frame rate and 2317 shot transitions (abrupt or gradual) [70]. Thus the 2007 shots are 275.3 frames long on average which is over 100 frames per shot longer than than in TRECVid 2006 shot boundary task (157.7 frames/shot). The TRECVid 2007 dataset contains also the ground-truth which describes all the transitions and divides each of them to one of the following categories:

- *cut*: no transition, i.e., the last frame of one shot followed immediately by the first frame of the next shot with no fade or other combination.
- *dissolve*: shot transition takes place as the first shot fades out *while* the second shot fades in.
- *fadeout/in*: shot transition takes place as the first shot fades out and *then* the second fades in.
- *other*: everything not in the previous categories e.g., diagonal wipes.

The distribution of transition types was 90.8%, 5.4%, 1%, and 3.7% for hard cuts, dissolves, fades to black and back and others respectively [70].

Over the years, the following measures for evaluating a shot boundary detection algorithm were calculated by the NIST software: *inserted transition count, deleted transition count, correction rate, deletion rate, insertion rate, error rate, quality index, correction probability, recall, and precision.* From these precision and recall were the primary measures used in the presentation of results at TRECVid and those will be used in our evaluations too. In [22] five frame difference for abrupt and gradual transitions was allowed between the estimation and the ground-truth data. Because in our implementation a given video is processed by a small window, we make an exception to the performance protocol and the estimated cut location is counted as correct if the ground-truth location of the cut is less than a frame window length away. If we are interested of a single measurement value, recall (R) and precision (P) were combined with equal weights in the F-measure:

$$F = \frac{2 \times R \times P}{R + P} \tag{19}$$

5.2 Experiment

In this section we give a brief introduction to our experimental Bag-of-Words approach, describe the optimal configuration set-up and report the results. The experiments were conducted with the TRECVid 2007 Shot Boundary data set including all the 17 videos. In the evaluation, the precision and recall curves were calculated using the ground-truth. The evaluation curves were computed by iteratively testing all the possible values of the threshold. The implementation was programmed by using C++ and OpenCV [71] libraries that provided optimized algorithms for large data sets and for high dimensional feature processing.

5.2.1 Dynamic visual codebook approach

In [69] and [10] the use of BoW on shot boundary detection was utilized by a visual codebook which was created from a huge junk of features in the beginning of processing a given video. This however is not ideal for a customer-level application because the training phase of the codebook will take a lot of time before we even can start the actual cut detection. Thus we will introduce a dynamic codebook approach where the codebook will be trained only by a small portion of the features which other proposed approach used. The dynamic codebook will be rebuilt every time algorithm detects a cut and the frames from the new scene will be used in the construction phase. Processing the video in this manner, we can in parallel show the already processed material while at the same time the method continues to process the video. The scene boundary detection algorithm is given in Algorithm 3.

Algorithm 3 Scene boundary detector

```
 1: Initialize_codebook = true
 2: for $i \leftarrow 1$ to video_length step window_size do
 3:     for Frames $j \leftarrow i$ to $i + window\_size$ do
 4:         Extract dense interest points
 5:         Form the descriptors
 6:         Store the descriptors desc_all ← descriptors
 7:     end for
 8:     Select a subset of all the descriptors: desc ⊂ desc_all
 9:     if Initialize_codebook = true then
10:         Train the codebook cb using desc
11:         Form the comparison histogram compHist
12:         Set Initialize_codebook to false and continue
13:     end if
14:     Form the histogram hist using codebook cb and the descriptor vector desc
15:     Calculate the distance d = hist-compHist
16:     if d $\geq$ T then
17:         Mark a scene boundary to the first frame of the window: cut(k):=i-window_size
18:         Set Initialize_codebook to true
19:     end if
20: end for
21: Return the vector of scene boundaries cut
```

In the algorithm we will process the given video by a moving sliding windows which has the length of *window size*. We resize the images (frames) to be 300 $\times$ 300 and transform them to gray scale images. From the frames in the window we extract interest points using dense sampling. Our decision to use dense sampling instead of local features (such as corners or blobs) is based on the comparative evaluation of detection methods by Fei-Fei *et. al* [72]. They showed that dense features work better for scene classification and thus it should be a suitable approach for scene boundary detection. Distance between two patch is set to 10 pixels and patch size to 20 pixel giving us approximately 850 features per each frame. Dense sampling coupled with SIFT descriptor provided promising results in Section 3 and we will form descriptors for the interest points by using SIFT. Using SIFT each descriptor is a 128-vector. When we are using other detection method than dense sampling (for instance SIFT detector) a situation can occur, where the algorithm has not find enough visual feature candidates for the codebook.

The visual codebook is generated using the extracted descriptors from the frames in the current window and using the k-means algorithm. The histograms (*compHist* and *hist*) are constructed using the brute-force matcher which finds the best matches in the codebook *cb* for every extracted descriptor. It is worth to mention that the matcher parameter *normType* should be set to *NORM_L1*, what increases the results by $0.05 - 0.3$ when using SIFT descriptors. For each frame the histogram is normalized by:

$$h_n = \frac{\textit{number of descriptors closest to } n}{\textit{total number of descriptors}} \quad n = 0, 1, .., L-1 \tag{20}$$

where n denotes a histogram bin. Normalization is a necessary step and without it for instance a visual code histogram of an image containing a human face differ a lot from an image containing multiple faces of the same person.

The distance between two code histograms is calculated using the Euclidean distance:

$$D(A, B) = \sqrt{\sum_{i=0}^{L-1} (h_i(A) - h_i(B))} \tag{21}$$

The parameters for the method are the threshold value T, size of the visual codebook, and the window size. The low values of T result to high recall but low precision and vice versa.

5.2.2 Optimal window size

The window size determines which frames will be included into the training phase of the codebook and how many frames will be used in the building process of code histograms. More precisely the size of the window tells how wide is our comparison angle in the time domain. All the TRECVid2007 videos are displayed at 25 frames per second and thereby with frame sizes 25, 50, and 100 we will use approximately 1, 2, and 4 second time intervals from the video respectively. In the experiments, all frames inside a window were used, but we randomly selected 1000 features among them to reduce the computation time. Figure 29 shows the results of varying frame sizes. The bigger frame size indicates better results, but it makes the comparison biased: bigger the frame, higher the probability that a true cut will be "accidentally" included. For instance an arbitrary video having scenes duration smaller than 100 frames and selecting the smallest possible threshold value with a frame size 100, the algorithm will "detect" all the possible cut locations. However, increasing the frame size over 400 we can no longer detect all the possible cut location with TRECVid 2007 dataset because a frame will eventually contain multiple cut location and only a one of them will we noticed. In the following experiments we will use the window size of 25 to get the most accurate view of the functionality of the algorithm with different parameter settings.

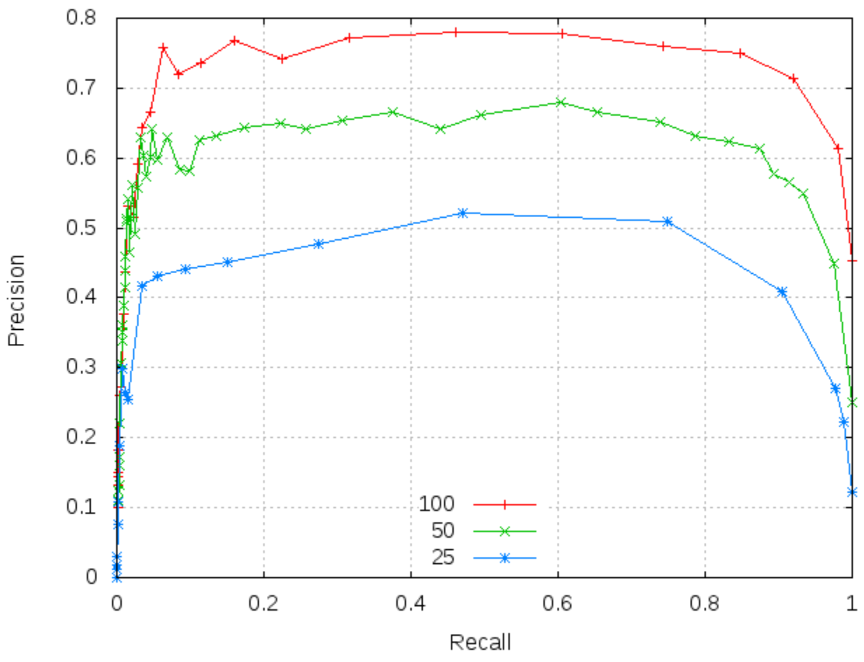


Figure 29. Scene boundary detector with varying frame size.

5.2.3 Optimal codebook size

One of the biggest bottlenecks of the algorithm is the size of the codebook. The computation time increases linearly with the codebook size (Table 6). The codebook is generated using the k-means algorithm which returns us the center points (codewords) for the codebook. To make our method as efficient as possible we try to find the smallest possible codebook size that is still discriminative enough to distinguish different scenes.

Table 6. Computation times of OpenCV implementations of k-means algorithm for 84100 SIFT features.

# centers	*10*	*20*	*50*	*100*	*1000*	*2000*
Comp. time (s.)	0.632	2.813	7.905	14.994	144.665	282.151

In Figure 30 is shown the results and the k-means algorithm was run with three different number of centers: 10, 100, and 1000. In the experiment, window size was set to 25 and due to time limit for the work only a subset (2/3 of the videos) were used in the evaluations. The result indicates that bigger codebook sizes do not necessarily mean better results and even using only ten centers the algorithm still works well. It was however noted that the same threshold value does not work well with different codebook sizes: smaller the codebook, greater the threshold value needs to be to prevent false alarms.

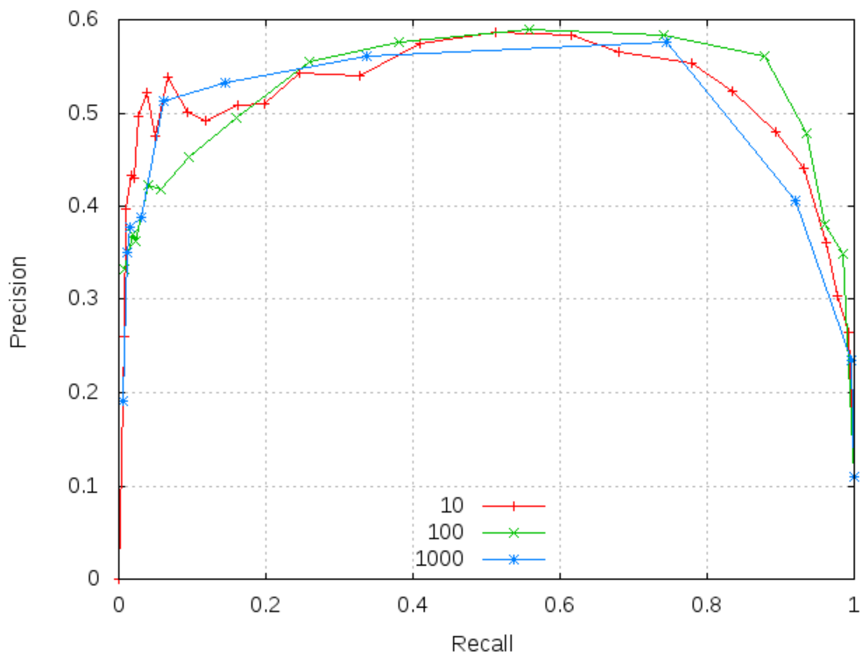


Figure 30. Scene boundary detection method with different codebook sizes.

5.2.4 Optimal number of features

In addition to the codebook size, the number of features used in the k-means algorithm to build the codebook affects the computation time a lot. Also with the knowledge of optimal number of features, we can reduce the feature extraction time. In this experiment we find the smallest feature number to build the codebook (the codebook size determines the minimum possible feature number) and the code histograms which gives us reliable results. The experiment was run with three different number of features: 100, 1000, and 10000. The same configuration settings were used as in the earlier experiments and the codebook size was fixed to 100 centers. The result are reported in Figure 31. The results indicate that using only 1000 features (from total of 21025 features) we can still achieve good results. Below that, the algorithm performance starts to degrade significantly.

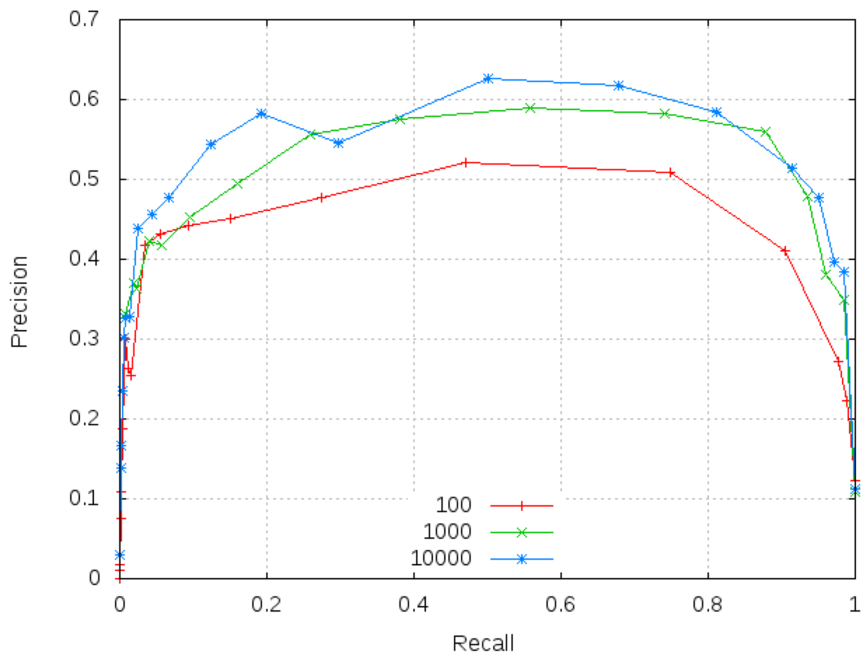


Figure 31. Scene boundary detection method with different number of features for a window size 25.

5.2.5 Comparison to existing methods

In the final experiment the method was compared against the statistical codebook Bag-of-Words approach by Lankinen *et. al* [69]. Both methods were evaluated using all the TRECVid 2007 videos. The choice of the codebook size was the same for all the methods: 100. The general codebooks in [69] were generated using features from the ImageNet database whereas the specific codebooks were generated using selected features from the given video. Again, the window size of 25 was used and 1000 features were randomly selected from the frames. The scene boundary detection results are shown in Figure 32.

The results indicate that our method cannot compete against the specific codebook approach in the low recall area. The reason for this is perhaps that the specific codebooks in the beginning of the processing are constructed using features extracted from selected frames over the whole video content (one frame per second) and thus the codebook comes very universal. In our implementation the codebook is typically generated (when the cut happens) using features of a specific video scene, for instance hiking in the Alps, making the codebook represent only a fraction of the content of the video. However, because we are more interested in the approaches performance in the high recall area, our method outperforms the general codebook clearly and when the recall is over 0.9, dynamic BoW method has the highest precision value.

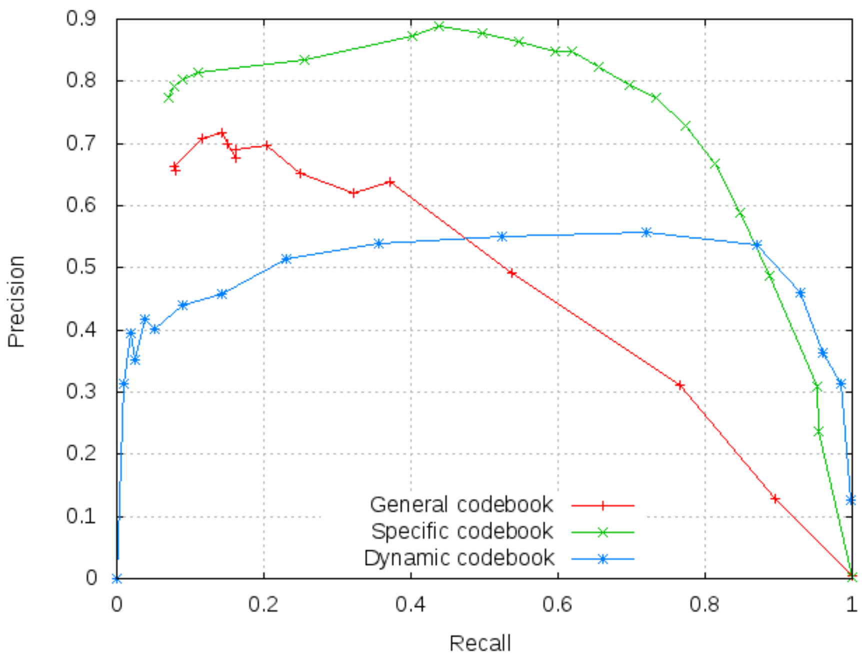
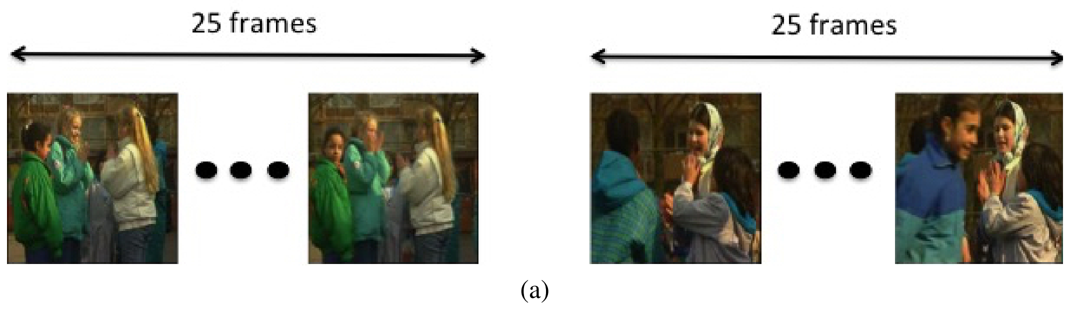


Figure 32. Scene boundary detector comparison with the existing methods.

5.3 Discussion

In this chapter video scene boundary detector using dynamic visual codebook was introduced and iterative search of optimal parameters for the method was done. Rapid evolve of smartphones, tablets, and outdoor activity purposed cameras have created great need of customer-level video applications. Shot and scene level boundary detectors serve as the initial step in variety of applications (such as video summarization and object retrieval) and thus their performance is crucial. A 20 minute video contains roughly 35000 frames which makes the processing very time consuming. The performance of the method was measured by the precision and recall curve. The parameter search was conducted using a subset of the videos ($2/3$ of total number) in TRECVid 2007 dataset and the final comparison with existing methods was evaluated using the whole dataset (17 different videos).

During our experiment we found out that codebook size of 100 or even 10 is enough to make the method as discriminant as possible. Raising the codebook size above 100 did not bring any notable performance increase. It was also noted that lowering the number of features used in codebooks and code histograms building below 1000 starts to reduce the performance gradually. Frame size impact for the accuracy was hard to determine because short scene duration time of the videos. This made longer windows seem to work better but with a lengthy video having long scene cuts the performance of algorithm with different frame sizes is likely to be equal. Judging by our results, our method precision lacks behind the state-of-the-art methods on average in the lower recall area. However, when the methods were tuned to be more sensitive, the dynamic BoW method achieves the best results. The dynamic BoW performance in the low recall area could be compensated by optimizing dense sampling (the grid size and the distance between two patches) and SIFT (number of octave levels, size of Gaussian kernel etc.) parameters but due to time constraints it will be left as future work. A true cut was typically missed between similar scenes having lots of similar features (people, background etc.) and where the camera has zoomed or the same scene is shot from another direction. In our case we are interested in distinguish two different scenes which are shot in different time and place (eating breakfast, driving a car etc.), and thus missing cuts between shots like illustrated in Figure 33) is irrelevant. Typically false cut was detected when a camera focus changed from a view to another within a single shot. To prevent this kind of false alarms (Figure 34a) would require much more advanced algorithm. Also in some videos there existed false cuts within an evident single shot (Figure 34b).



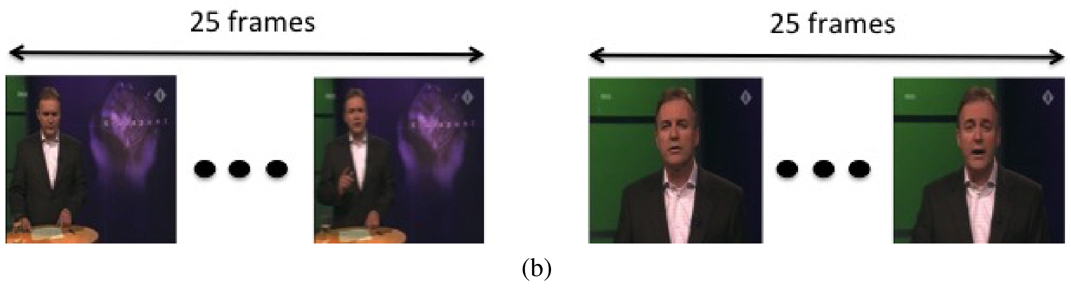
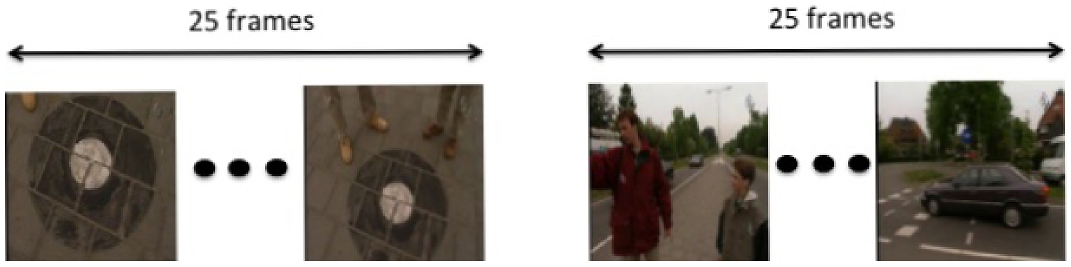
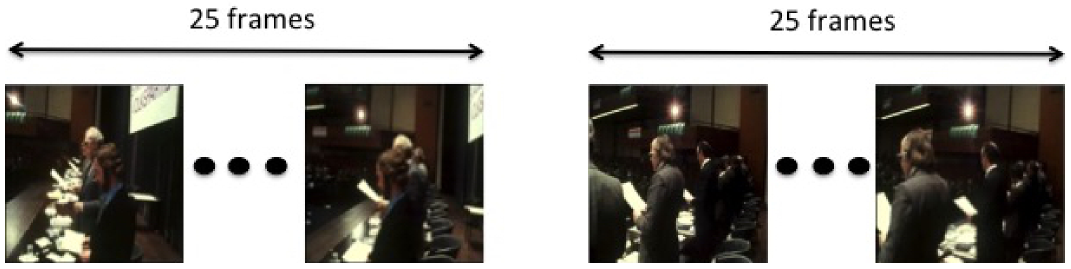


Figure 33. Missed cuts from the TRECVid 2007 videos a) BG_35187 and b) BG_36537.

Because of development of video editing applications, cameras and the entire video industry the characteristics of today's video materials have changed significantly. TRECVid 2007 contains rather poor quality videos without modern video effects, thus existing methods should be re-evaluated against more sophisticated data sets.



(a)



(b)

Figure 34. False alarms from the videos a) BG_2408 and b) BG_36028.

6 CONCLUSION

In this thesis, two applications of local features were investigated. The first goal was to find the best local feature detector and descriptor for visual object categorization task by extending the well known benchmark by Mikolajczyk *et al.* [30, 27]. The second task was to implement a customer-level video scene boundary detector using the visual bag-of-words method.

For the local detector and descriptor evaluation for intra-class matching the evaluation framework was set up and the meta-parameters of different detectors and descriptors were investigated. The detector evaluation provided several important findings with regard to object class matching: 1) dense sampling outperformed its rival interest point detectors with a clear margin; 2) object pose variation degrades dense sampling performance while the best interest point detector (Hessian-affine) is unaffected; 3) with the adjustment of the detectors' meta-parameters, the detectors produce very similar results. The descriptor evaluation showed that performance of descriptors to match similar features between intra-class objects is poor and the following conclusions were made: 1) the original SIFT is still the best descriptor; 2) only a fraction of the corresponding regions match in the descriptor domain; 3) using multiple, even a few, best matches instead of the single best has significant effect on the performance. In addition to investigations of descriptors for intra-class matching, future work include research on combinations of interest points and dense sampling [33] and alternative matching methods for multiple best matches.

A scene boundary detector using dynamic visual codebook was implemented to respond to the need of customer-level initial stage procedure for video applications, such as video summarization. During the experiments the optimal configuration set up was iteratively searched by plotting recall and precision curves and the following discoveries were made: 1) bigger codebook size does not necessary mean better performance; 2) only a fraction of the features used inside the query window is enough to give good results; 3) dynamic codebook can still achieve good precision on the high recall area. The evaluations indicated that dynamic codebook does not outperform on average the state-of-the-art methods in retrieving only a few key scenes but when the algorithms are configured to be more sensitive against dissimilarity peaks, the dynamic BoW method outperforms all the other methods.

In the future work one should focus on improving the dynamic BoW approach. For instance does an iterative search of optimal parameters for the dense sampling and SIFT improve the results. Also the testing process is very time consuming due to fact that the TRECVid 2007 video set has 17 videos having tens of thousands of frames each and collection of more sophisticated video test material is advised.

REFERENCES

[1] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Henry Holt and Co., Inc., New York, NY, USA, 1982.

[2] Bernd Jähne and Horst Haussecker, editors. *Computer Vision and Applications: A Guide for Students and Practitioners*. Academic Press, Inc., Orlando, FL, USA, 2000.

[3] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[4] Lawrence G. Roberts. *Machine Perception of Three-Dimensional Solids*. Outstanding Dissertations in the Computer Sciences. Garland Publishing, New York, 1963.

[5] Hans Peter Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford University, Stanford, CA, USA, 1980.

[6] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[7] Steven Seitz and Richard Szeliski. Applications of computer vision to computer graphics. *Computer Graphics*, 33(4):35–37, November 1999. Guest Editors' introduction to the Special Issue.

[8] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[9] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[10] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[11] Jean Ponce, Martial Hebert, Cordelia Schmid, and Andrew Zisserman, editors. *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*. Springer, 2006.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[13] Tinne Tuytelaars and Krystian Mikolajczyk. *Local Invariant Feature Detectors: A Survey*. Now Publishers Inc., Hanover, MA, USA, 2008.

[14] J. S. Boreczky and L. A. Rowe. Comparison of Video Shot Boundary Detection Techniques. In *Storage and Retrieval for Still Image and Video Databases IV*, Los Angeles, California, January 1996.

[15] R. Lienhart. Comparison of automatic shot boundary detection algorithms. In *Storage and Retrieval for Image and Video Databases*, pages 290–301, January 1999.

[16] M. Brown and D. G. Lowe. Recognising panoramas. In *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*, ICCV ’03, pages 1218–, Washington, DC, USA, 2003. IEEE Computer Society.

[17] Mark J. Brady and Daniel Kersten. Bootstrapped learning of novel objects. *Journal of Vision*, 3(6):2, 2003.

[18] Tinne Tuytelaars and Luc Van Gool. Matching widely separated views based on affine invariant regions. *International Journal of Computer Vision*, 59(1):61–85, August 2004.

[19] Krystian Mikolajczyk, Bastian Leibe, and Bernt Schiele. Local features for object class recognition. In *Proceedings of the 10th IEEE International Conference on Computer Vision*, pages 1792–1799, 2005.

[20] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: a comprehensive study. *International Journal of Computer Vision*, 73(2):213–238, june 2007.

[21] S. Se, D. Lowe, and J. Little. Global localization using distinctive visual features. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 226–231 vol.1, 2002.

[22] Alan F. Smeaton, Paul Over, and Aiden R. Doherty. Video shot boundary detection: Seven years of TRECVid activity. *Computer Vision and Image Understanding*, 114(4):411 – 418, 2010. Special issue on Image and Video Retrieval Evaluation.

[23] A. Eleyan and Hasan Demirel. Co-occurrence based statistical approach for face recognition. In *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*, pages 611–615, Sept 2009.

[24] Dimitri A. Lisin, Marwan A. Mattar, Matthew B. Blaschko, Mark C. Benfield, and Erik G. Learned-miller. Combining local and global image features for object class recognition. In *Proceedings of the IEEE CVPR Workshop on Learning in Computer Vision and Pattern Recognition*, pages 47–55, 2005.

[25] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[26] H. Bay, A. Ess, T. Tuytelaars, and L. van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, June 2008.

[27] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[28] P. R. Beaudet. Rotationally invariant image operators. In *Proceedings of the 4th International Joint Conference on Pattern Recognition*, pages 579–583, Kyoto, Japan, November 1978.

[29] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30:79–116, 1998.

[30] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, November 2005.

[31] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. In *International journal of computer vision*, volume 1, pages 63–86, 2004.

[32] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[33] T. Tuytelaars. Dense interest points. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2281–2288, June 2010.

[34] Eric Nowak, Frédéric Jurie, and Bill Triggs. Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision*. Springer, 2006.

[35] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *Proceedings of the 11th European*

Conference on Computer Vision: Part IV, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.

[36] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[37] O. Miksik and K. Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2681–2684, Nov 2012.

[38] Paul Rosin. Measuring corner properties. In *Computer Vision & Image Understanding, Vol.73, No.2*, pages 291–307, 1999.

[39] Kevin Murphy, Antonio Torralba, Daniel Eaton, and William Freeman. Object detection and localization using local and global features. In Jean Ponce, Martial Hebert, Cordelia Schmid, and Andrew Zisserman, editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science*, pages 382–400. Springer Berlin Heidelberg, 2006.

[40] Raphael Ortiz. Freak: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society.

[41] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society.

[42] A. Hietanen, J. Lankinen, J.-K. Kamarainen, A. Buch, and N. Krüger. A comparison of feature detectors and descriptors for object class matching. *Neurocomputing*, 2015. In Press.

[43] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing Object Detection Features. *International Conference on Computer Vision*, 2013.

[44] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[45] T. Kinnunen, J.-K. Kamarainen, L. Lensu, J. Lankinen, and H. Kälviäinen. Making visual object categorization more challenging: Randomized Caltech-101 data

set. In *20th International Conference on Pattern Recognition (ICPR2010)*, Istanbul, Turkey, 2010.

[46] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255, June 2009.

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

[48] J. Lankinen, V. Kangas, and J.-K. Kamarainen. A comparison of local feature detectors and descriptors for visual object categorization by intra-class repeatability and matching. In *21th International Conference on Pattern Recognition (ICPR2012)*, Tsukuba Science City, Japan, 2012.

[49] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

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

[51] Ankur Agarwal and Bill Triggs. Hyperfeatures – multilevel local coding for visual recognition. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 30–43. Springer Berlin Heidelberg, 2006.

[52] Tinne Tuytelaars and Cordelia Schmid. Vector quantizing feature space with a regular lattice. In *ICCV*, pages 1–8. IEEE, 2007.

[53] Ken Chatfield, Victor Lempitsky, Andrea Vedaldi, and Andrew Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. In *Proceedings of the British Machine Vision Conference*, pages 76.1–76.12. BMVA Press, 2011. http://dx.doi.org/10.5244/C.25.76.

[54] Bastian Leibe, Alan Ettlin, and Bernt Schiele. Learning semantic object parts for object categorization. *Image Vision Comput.*, 26(1):15–26, January 2008.

[55] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *Pattern Analysis and Machine Intellingence*, 2014.

[56] Ba Tu Truong and Svetha Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1), February 2007.

[57] Jacinto C. Nascimento and Jorge S. Marques. Performance evaluation for object detection algorithms for video surveillance. In *IEEE Transaction on Multimedia*, 2006.

[58] Darin Brezeale and Diane J. Cook. Automatic video classification: A survey of the literature. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(3):416–430, 2008.

[59] M. Cooper, T. Liu, and E. Rieffel. Video segmentation via temporal pattern classification. *Multimedia, IEEE Transactions on*, 9(3):610–618, April 2007.

[60] A. Hanjalic. Shot-boundary detection: unraveled and resolved? *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(2):90–105, Feb 2002.

[61] Yong Rui, T.S. Huang, and S. Mehrotra. Exploring video structure beyond the shots. In *Multimedia Computing and Systems, 1998. Proceedings. IEEE International Conference on*, pages 237–240, Jun 1998.

[62] J. Baber, Nitin Afzulpurkar, M.N. Dailey, and M. Bakhtyar. Shot boundary detection from videos using entropy and local descriptor. In *Digital Signal Processing (DSP), 2011 17th International Conference on*, pages 1–6, July 2011.

[63] Jun Li, Youdong Ding, Yunyu Shi, and Wei Li. A divide-and-rule scheme for shot boundary detection based on SIFT. *International Journal of Digital Content Technology and its Applications*, 4(3):202–214, 2010.

[64] Ye Luo, Ping Xue, and Qi Tian. Scene alignment by SIFT flow for video summarization. In *Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on*, pages 1–5, Dec 2009.

[65] T. Lu and P.N. Suganthan. An accumulation algorithm for video shot boundary detection. *Multimedia Tools and Applications*, 22(1):89–106, 2004.

[66] Efthymia Tsamoura, Vasileios Mezaris, and Ioannis Kompatsiaris. Gradual transition detection using color coherence and other criteria in a video shot meta-segmentation framework. In *International Conference on Image Processing*, pages 45–48. IEEE, 2008.

[67] S. M. M. Tahaghoghi, Hugh E. Williams, James A. Thom, and Timo Volkmer. Video cut detection using frame windows. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science - Volume 38*, ACSC '05, pages 193–199, Darlinghurst, Australia, Australia, 2005. Australian Computer Society, Inc.

[68] T. Kinnunen, J.-K. Kamarainen, L. Lensu, and H. Kälviäinen. Bag-of-features codebook generation by self-organisation. In *7th International Workshop on Self-Organizing Maps (WSOM2009)*, St. Augustine, FL, USA, 2009.

[69] J. Lankinen and J.-K. Kämäräinen. Video shot boundary detection using visual bag-of-words. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, Barcelona, Spain, 2013.

[70] Paul Over, George Awad, Wessel Kraaij, and Alan F. Smeaton. TRECVID 2007–overview. In *TRECVID 2007 workshop participants notebook papers, Gaithersburg, MD, USA, November 2007*, 2007.

[71] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[72] Fei-Fei Li and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 524–531, Washington, DC, USA, 2005. IEEE Computer Society.