Lappeenranta University of Technology
Faculty of Technology Management
Degree Program in Information Technology

Master's Thesis

Ville Kangas

COMPARISON OF LOCAL FEATURE DETECTORS AND DESCRIPTORS FOR VISUAL OBJECT CATEGORIZATION

Examiners:	Professor Joni Kämäräinen
	Jukka Lankinen M.Sc (Eng)
Supervisors:	Professor Joni Kämäräinen
	Jukka Lankinen M.Sc (Eng)

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Degree Program in Information Technology

Ville Kangas

Comparison of Local Feature Detectors and Descriptors for Visual Object Categorization

Master's Thesis

2011

69 pages, 32 figures, 2 tables and 3 appendices.

Examiners:	Professor Joni Kämäräinen
	Jukka Lankinen M.Sc (Eng)

Keywords: local feature, interest point, feature detector, feature descriptor, visual object categorization, machine vision, computer vision

Local features are used in many computer vision tasks including visual object categorization, content-based image retrieval and object recognition to mention a few. Local features are points, blobs or regions in images that are extracted using a *local feature detector*. To make use of extracted local features the localized interest points are described using a *local feature descriptor*. A descriptor histogram vector is a compact representation of an image and can be used for searching and matching images in databases. In this thesis the performance of local feature detectors and descriptors is evaluated for object class detection task. Features are extracted from image samples belonging to several object classes. Matching features are then searched using random image pairs of a same class. The goal of this thesis is to find out what are the best detector and descriptor methods for such task in terms of detector repeatability and descriptor matching rate.

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Ville Kangas

Paikallisten piirteiden havaitsemis- ja kuvaamismenetelmien vertailu kohteiden visuaalisessa luokittelussa

Diplomityö

2011

69 sivua, 32 kuvaa, 2 taulukkoa ja 3 liitettä.

Tarkastajat:	Professori Joni Kämäräinen
	DI Jukka Lankinen

Hakusanat: paikallinen piirre, piirrehavaitsin, piirremäärittelijä, visuaalinen kohteiden luokittelu, konenäkö, tietokonenäkö
Keywords: local feature, interest point, feature detector, feature descriptor, visual object categorization, machine vision, computer vision

Paikallisia piirteitä käytetään monissa konenäön sovelluksissa, kuten visuaalisessa kohteiden luokittelussa, sisältöperusteisissa hakusovelluksissa ja kohteiden tunnistuksessa. Paikalliset piirteet ovat tavallisesti pisteitä tai alueita, jotka irrotetaan kuvasta erityisen piirrehavaitsijan avulla. Löydetyt piirteet kuvataan kuvaamisvektorilla. Tässä työssä vertailtiin eri havaitsemis- ja kuvaamismenetelmien suorituskykyä kohteiden luokantunnistustehtävässä. Testikuvien joukossa on yhteensä 250 satunnaisesti valittua näytekuvaparia kymmenestä eri luokasta. Suorituskykytestit suoritettiin aina kuvaparin välillä. Havaitsijatestiä varten kuvaparin molemmista näytekuvista irrotettiin piirteet, jonka jälkeen piirteitä kuvaavien ellipsien päällekkäisyyksiä tarkastelemalla saatiin laskettua vastaavien piirteiden määrä. Vastaavasti kuvaamismenetelmien testissä kuvaparien väliltä etsittiin kuvattujen piirteiden väliltä osumia. Työn tarkoituksena on selvittää, mikä havaitsemis- ja kuvaamismenetelmistä on paras vaihtoehto kohteidenluokittelutehtävään.

CONTENTS

ABBREVIATIONS AND SYMBOLS

CAD	Computer-aided Design
DLT	Direct Linear Transform
DoG	Difference-of-Gaussian
LoG	Laplacian-of-Gaussian
MSER	Maximally Stable Extremal Regions
PCA	Principal Component Analysis
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
VOC	Visual Object Categorization

$\mathbf{H}$	Hessian matrix
$I(\mathbf{x})$	Image intensity at location $\mathbf{x} = (x, y)$
$\mathbf{M}$	Second-moment matrix
R	Harris cornerness
s	Euclidean distance
$\mathcal{S}$	Range of values
$\mathbf{v}$	Descriptor vector
$\hat{\mathbf{v}}$	Normalized descriptor vector
λ_1, λ_2	Eigenvalues in two dimensions

1 INTRODUCTION

The number of digital images in many kinds of archives starting from home computers is rising rapidly. People carry cameras around constantly. The potential of consumer-grade cameras has grown and almost everyone knows how to save and share photos over the Internet. The wide use of digital cameras establishes a demand for intelligent systems to organize, search and categorize images.

The development of software has made a lot of very sophisticated software available for advanced users. The requirements of the used digital photograph editing and organizing software are still going to be more demanding in the future. Increasing computational resources will make more and more complex processing possible.

As a part of many successful image categorization and searching solutions, *local features* are used. Local features can be thought as patterns in images that differ from the immediate neighborhood. Such a pattern can be a corner, blob or region. To be able to exploit found local features, the immediate neighborhoods need to be described in an efficient and compact way to make it possible to match it to similar patterns in other images. A *local feature descriptor* creates the description.

In this thesis main focus is on object recognition and categorization tasks, where there are several sample images from a same class but not from a same scene. An evaluation and comparison of the best available local feature detectors and descriptors was done using an existing test protocol for detectors, and a novel test scheme for descriptors.

1.1 Background

Research on local features has been active over the last few decades. A quick overview of the timeline is given for both local feature detecting and description methods.

1.1.1 Local features in literature

The use of local features has become more and more popular. Increased computational power gives a lot more potential for applications analyzing and categorizing digital images. Despite the limitations of computational resources in the past decades, the history of local features is quite long. The theory used in current state-of-art methods was introduced during the last few decades starting from 1970s, when the first actual detectors were implemented. In this section, the major development steps of local feature detection methods from the 1970s are discussed.

In the literature, as mentioned by Tuytelaars et al. [1], theory behind local feature detection goes back in 1954. Attneave [2] worked on the information theory and observed that shape information is concentrated on points where object contour changes its direction maximally. Early feature detection was concentrated on detecting simple objects in CAD images or other very structured drawings rather than natural scenes [1]. Hessian detector, proposed by Beaudet 1978 [3], is based on the Hessian matrix. It detects blobs and ridges and works well with simple drawings with no background clutter. The Hessian detector is discussed in more detail in Section 2.3.

Early work on the local features with natural scenes was done by Moravec from late 1970s [4]. Moravec's *interest operator* is based on the intensity variation in the local neighborhood of a pixel. An improved version, Harris corner detector, was proposed by Harris and Stephens [5]. Since some of the currently used methods are based on Harris corners, the method is described in Section 2.4.

Methods proposed in 1970s and 1980s were not scale-invariant in the sense that same features would be extracted if an image is re-scaled. Several approaches were proposed, e.g. using multi-scale variations of existing methods [6]. Important work on what current scale-invariant methods are based was done in 1990s by Lindeberg [7, 8].

The methods compared in this thesis are published during the late 1990s and 2000s. The detectors include *Harris-Affine* and *Hessian-Affine* from Mikolajczyk and Schmid [9], *Maximally stable extremal regions (MSER)* from Matas et al. [10], *Speeded-up robust features (SURF)* from Bay et al. [11], *Difference-of-Gaussian (DoG)* from Lowe [12] and *Laplacian-of-Gaussian (LoG)* from Lindeberg [8].

1.1.2 Feature description and matching

The task of a local feature descriptor is to describe local image region such a way that it can be distinguished from other regions and also matched effectively with those that are similar enough. The descriptor is usually built after a feature has been detected using a local feature detector. In the following, a rough overview is given over the development of feature description methods in the literature.

The history of local feature description can be thought to start from the 1970s. Finding basic correspondences between two images has been studied since then. The use of local features makes it computationally easier to find candidate matches instead of exhaustive search on the images. Later the performance has become even more important when images are retrieved from large databases by finding images having similar set of descriptors.

During the 1980s and 1990s, many methods based on shape, appearance, frequency content and intensity changes have been used to build local feature descriptors [13]. There are other categories, but most of the descriptors can be put to at least one of these categories. During the 1990s there was already some success in many applications of local features.

The most widely known descriptor today, Scale-invariant Feature Transform (SIFT), was introduced by Lowe in 1999 [12]. Since then, many variations of SIFT have been proposed, like PCA-SIFT [14] and GLOH [9]. The current descriptors, such as the SIFT, have reached a level of maturity where the descriptor matching can be considered as a standard tool, ready for use in many applications. Since the performance has been quite good for a while now, interest has switched to developing faster methods for matching. One of the recent descriptors is SURF [11]. SURF provides both a robust detector and descriptor for very fast detection and description of features.

In this thesis, the selected set of descriptors contains SIFT from Lowe [12] and SURF from Bay et al. [11]. The descriptors were selected based on their performance in earlier studies and preliminary tests for this work. For SIFT there were two different implementations that were used with several detector implementations in the tests.

1.2 Applications of local features

In this thesis local feature extraction and description methods are evaluated and compared. To give motivation for such comparison, some of the most important application areas for local features are shortly introduced. The target is to shortly describe how local features can be used in these application areas.

1.2.1 Wide-baseline stereo matching

Wide-baseline matching is one of the natural applications for local features. It means finding corresponding pixels from two images taken from a same scene but from different viewpoints. The word baseline originally refers to the distance between the two cameras in a stereo system, but later the expression "wide baseline" is used to denote situations where significant change occurs between two images [15]. In wide-baseline matching, several assumptions are often made [15]:

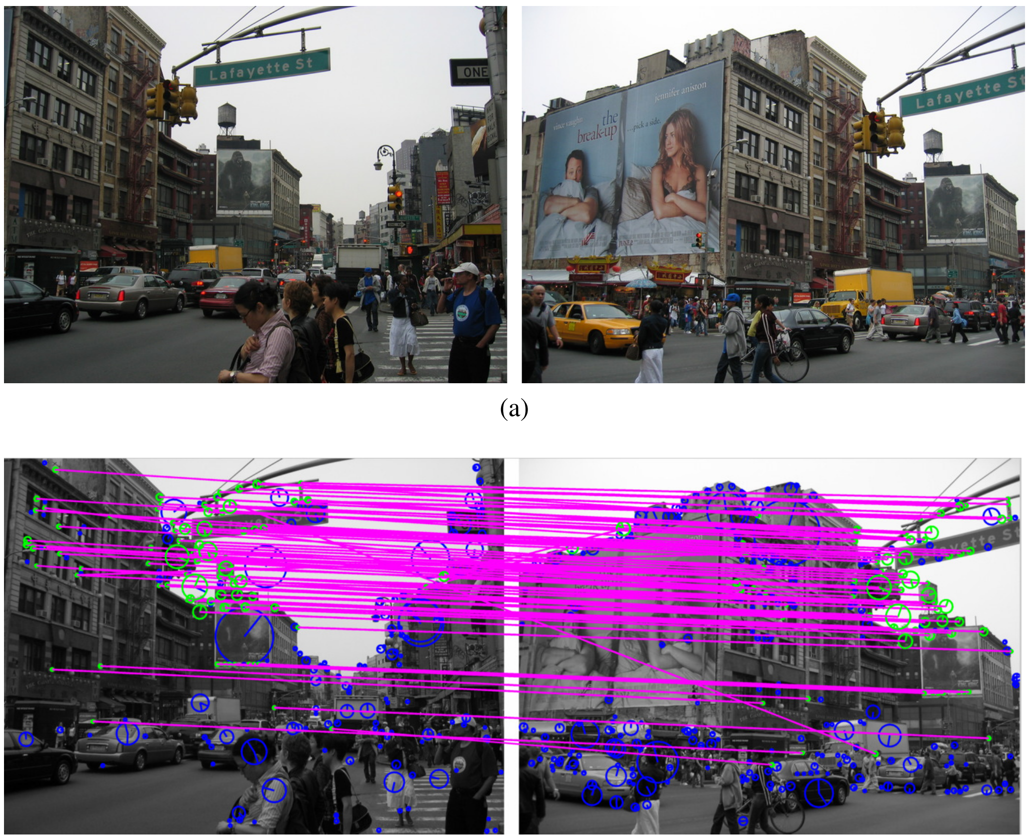
- A corresponding point for any of the original points may be anywhere in the other image
- Neighborhoods cannot be directly compared around corresponding points since the neighborhoods have been transformed by the camera movement
- The camera may be moved in any way, but the same objects must be visible in both images and viewed from "the same side"
- Camera parameters including focal length and other internal parameters may have changed
- Objects visible in one image may be occluded in the other.

The problem of finding corresponding features from two images of the same scene has been studied at least since 1970s. Some success in image matching and registration was achieved in the 1980s [16]. During the 1990s and 2000s the wide-baseline matching using local features has become quite reliable as several robust local feature detectors and descriptors exist. A matching example is given in Fig. 1. Circles in green are matched features and circles in blue are features without matches. Pink lines are drawn between two matching features. It is clearly visible that there are only a few "incorrect" matches which are not real correspondences.

1.2.2 Visual object categorization

In visual object categorization (VOC), the problem is to find categories for a given image set. If a set of classes is known beforehand, a system is supervised, and otherwise unsupervised. Use of local features has become quite popular in such task.

Local feature descriptors are usually used to build a *codebook* by clustering descriptor data. Descriptors are vectors, containing for example gradient histograms of local feature pixel neighborhoods. Clustering is statistical technique for grouping similar data. Training data is used to group similar samples that form groups, clusters. When local



(a)

(b)

Figure 1. (a) An image pair showing a few buildings from two different viewpoints and (b) SIFT matches showing the corresponding features.

features are used in visual object categorization, usually descriptor vectors are compared and descriptors that are similar enough, form clusters. Each cluster represents a word in the codebook. After training the system, new images can be classified by comparing new descriptors to words in the codebook. Images belonging to a same class often contain quite similar set of codewords of the codebook and can thus be distinguished from the other classes.

1.2.3 Object recognition

In an object recognition task there are usually several object classes and the task is to find out to which class an object in a given image belongs to. Depending on the task, one, several or no objects may appear in an image. Local features are used for many kinds of object recognition tasks. How local features are used, is task-dependent. In the following, two different kinds of object recognition system types are shortly described to show examples of using local features in object recognition.

One example is recognition of an object in two different scenes. An object may be partially occluded in the other scene, and local features have obvious advantages over methods based on appearance of a whole object. SIFT descriptors can be used and even partially occluded objects can be recognized [12]. However, the use of local features is feasible only when objects have at least some textures or other visual details to form distinctive descriptions.

An other example of using local features is as intermediate level of representation in an object class recognition task [17]. Based on the local features and their locations, very compact representation of an image can be formed. The idea is that depending on the approach, feature descriptions and locations are used to calculate a signature for a class. Clustering can be used to form classes from training data. After training the system, new samples can be classified using trained clusters.

1.3 Goals and restrictions

The main goal of this thesis is to identify the best methods for local feature detection and matching. The problem in this thesis is defined as matching different objects belonging to a same category, e.g. motorbikes, faces or stop signs. The task is a bit different than in the case of wide-baseline stereo matching where there is a single object (scene) and multiple view points. The main goal is to find out how well the current state-of-art detectors and descriptors can perform in this kind of task.

1.4 Structure of the thesis

Section 1 is an introduction to use of local features in machine vision tasks. Sections 2 and 3 will give details about local feature detectors and descriptors. Performance evaluation methods for detectors and descriptors are described in Section 4. The experiments that were done are explained in Section 5. The results are discussed in Section 6 and finally a short conclusion of the work done is given in Section 7.

2 LOCAL FEATURE DETECTORS

When it comes to local features, local feature detectors do the part of the process where local features are extracted. Many robust approaches for such task exist. First, a few general remarks about feature detectors are given.

2.1 Terminology

Several established terms related to local features in the literature may be a bit misleading [1]. The terms are widely used in the literature, but to avoid confusion they are now clarified. First, the term *local feature* is used in this thesis. There are many other terms used in the literature, e.g. *interest points*, *interest regions* and *keypoints*. The problem is that some detectors detect corners, i.e. points and some others blobs or ridges, i.e. regions. In that sense, both, interest region or interest point are incorrect for some of the features. The term local feature covers all the features and is thus the correct one.

Feature detector is not actually a detector, since it does not have *a priori* knowledge on the blobs, edges etc. in the image. The correct term would be extractor, but since *detector* is widely used, it will be used in this thesis also.

Similar problem exist with terms “invariant” and “covariant”. If a function is invariant to some transform, its value remains the same, i.e. it is *invariant*. If a function commutes with a transform, its output will change as the transform imposes, so it is “covariant”. Many local feature detectors are then actually covariant to scale and affine transforms, not invariant. But as above, the term “invariant” is widely used and it is used in this thesis as well.

2.2 Properties of an ideal local feature

It is not obvious what kind of local features are ideal. A way to detect semantically meaningful parts sounds ideal, but it is not feasible in practice. It means that local features are just patterns found in images. Tuytelaars and Mikolajczyk have listed six properties of an ideal local feature scheme: [1]

- *Repeatability:* When features are extracted from two sample images, a high proportion of the same features should be found in both images.

- *Distinctiveness:* Patterns found should be informative, i.e. distinguishable from the other local features when matched.
- *Locality:* The features should be local enough to be found from two images taken from a different view point of the same scene. For such features deformations such as noise, scale and rotation between the images can be handled.
- *Quantity:* Sufficiently large number of features is usually beneficial. Ideally a detector has also a threshold value that makes it possible to detect more or less features depending on the application. Still, of course, number of features should reflect the amount of shapes and interesting areas in an image.
- *Accuracy:* The location of the detected features should be accurately placed in image coordinates. The same also applies for scale and shape adaptation.
- *Efficiency:* The quicker the better. Important for time-critical applications.

2.3 Hessian detector

Hessian detector was proposed by Beaudet [3]. The detector is based on the *Hessian* matrix $\mathbf{H}$, i.e. partial second-order derivatives of the image intensity function $I(\mathbf{x})$. The terms I_{xx}, I_{xy} and I_{yy} denote the partial second-order derivatives of $I(\mathbf{x})$ at location $\mathbf{x} = (x, y)$:

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

In Fig. 2, the second derivatives and the determinant of the Hessian matrix are shown. In Fig. 4, an example determinant image is shown. White spots in the image denote local maxima of determinant response when $I_{xx}I_{yy} - I_{xy}^2$ is calculated over the image. Finally the local peaks higher than predefined threshold are selected. To make illustration clearer the image has been filtered to remove lower maxima. Blobs detected by Hessian detector are shown in Fig. 4.

2.4 Harris detector

Harris corner detector was proposed by Harris and Stephens [5]. The basic assumption of Harris detector is that at a corner, the intensity of an image will change in multiple

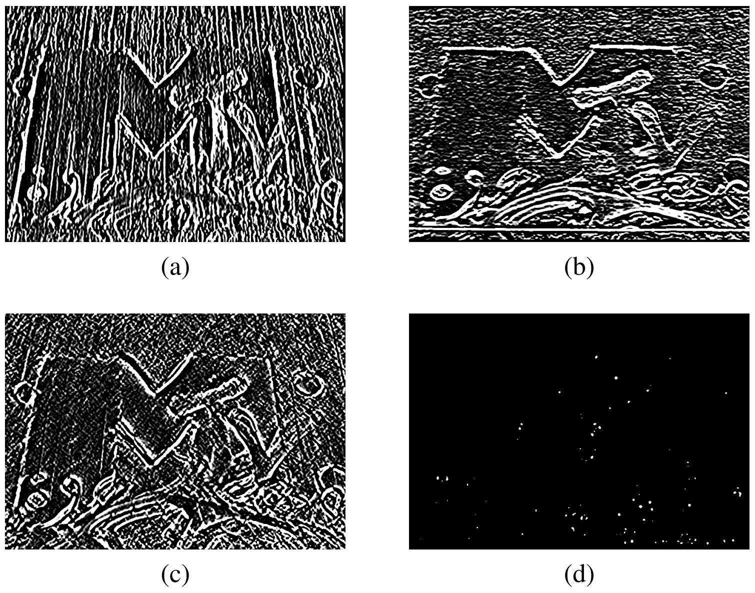


Figure 2. Components of the Hessian matrix. (a) I_{xx} (b) I_{yy} (c) I_{xy} (d) $I_{xx}I_{yy} - I_{xy}^2$.

directions. The detector is based on the *second moment matrix* $\mathbf{M}$, that describes the intensity change in the local neighborhood of a point $\mathbf{x} = (x, y)$:

$$\mathbf{M} = g(\sigma) \begin{bmatrix} I_x^2(\mathbf{x}) & I_xI_y(\mathbf{x}) \\ I_xI_y(\mathbf{x}) & I_y^2(\mathbf{x}) \end{bmatrix} \tag{2}$$

with

$$I_x(\mathbf{x}) = \frac{\partial}{\partial x} I(\mathbf{x}) \tag{3}$$

$$I_y(\mathbf{x}) = \frac{\partial}{\partial y} I(\mathbf{x}) \tag{4}$$

$$g(\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{5}$$

First, derivatives $I_x(\mathbf{x})$ and $I_x(\mathbf{x})$ are computed in x and y directions. Then, ${I_x}^2, I_xI_y$ and ${I_y}^2$ are calculated. Finally the derivatives are smoothed using Gaussian window of size σ. Each step is illustrated in Fig. 3. Full derivation of the second moment matrix $\mathbf{M}$, also known as auto-correlation matrix, is shown in [18, 19]. It has several properties that make finding corners efficient. The corners can be found in an image where the signal change is significant in both directions, i.e. the points where both eigenvalues are large.

Locations of corners are quite simple to calculate after the second-moment matrix for a

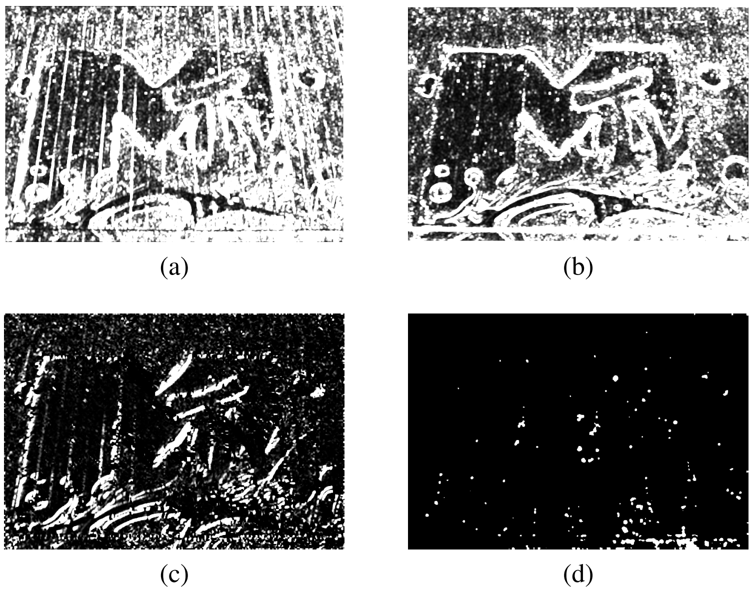


Figure 3. Components of the second moment matrix: (a) $I_x I_x$, (b) $I_y I_y$, (c) $I_x I_y$ and (d) $det(\mathbf{M}) - \alpha\, trace^2(\mathbf{M})$.

point has been composed. First, the 2×2 second-moment matrix $\mathbf{M}$ can be written as:

$$\mathbf{M} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}. \tag{6}$$

Harris proposed the *cornerness* measure R, that describes the cornerness of the local neighborhood of a point. R is computed using the trace and determinant of the matrix $\mathbf{M}$. In the following equations the calculation of cornerness based on the second moment matrix is shown. The trace and the determinant of such matrix are straightforward to calculate:

$$tr(\mathbf{M}) = \lambda_1 + \lambda_2 = A + B \tag{7}$$

$$det(\mathbf{M}) = \lambda_1 \lambda_2 = AB - C^2. \tag{8}$$

Cornerness can then be easily derived. Actually there is no need to calculate the eigenvalues λ_1 and λ_2 since:

$$R = det(\mathbf{M}) - \alpha trace^2(\mathbf{M}) \tag{9}$$

$$= \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 \tag{10}$$

$$= AB - C^2 - \alpha(A + B)^2. \tag{11}$$

A constant α is used for balancing the terms in the equation. Typical value for α is 0.04

[1]. Cornerness measure R can be used to roughly classify the responses found (also behavior of eigenvalues in such cases are denoted):

1. $R \approx 0 \quad (\lambda_1 \approx \lambda_2)$: a flat region is found
2. $R < 0 \quad (\lambda_1 \gg 0 \vee \lambda_2 \gg 0)$: an edge is found
3. $R > 0 \quad (\lambda_2 \gg 0 \wedge \lambda_2 \gg 0)$: a corner is found.

If only one of the eigenvalues is significantly higher than zero, an edge is found. A corner is considered to be found only if both eigenvalues are significantly larger than zero. Finally local maxima of R above a given threshold are considered as found Harris corners. An example of the corners detected by Harris detector are shown in Fig. 5.

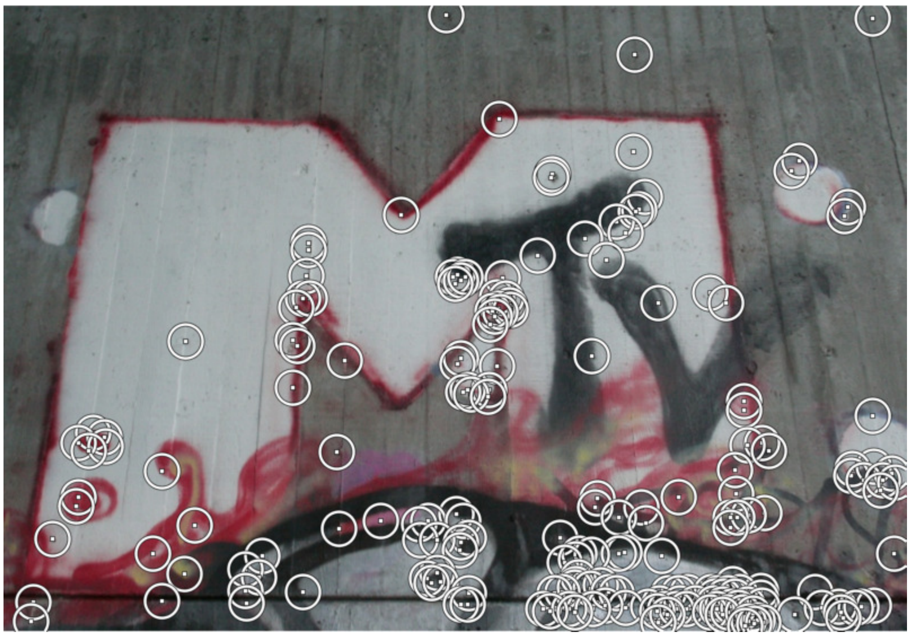


Figure 4. Local features detected with Hessian detector.

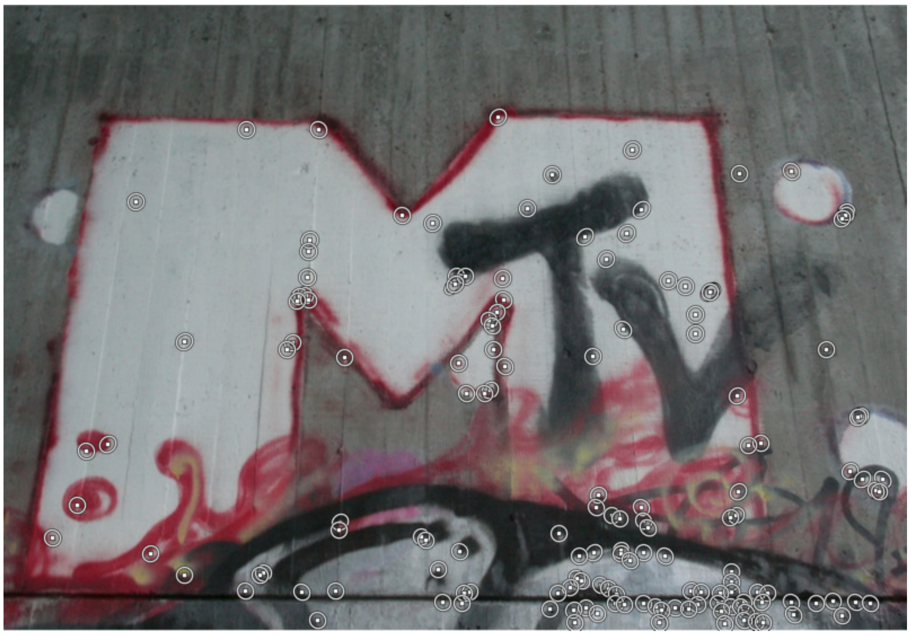


Figure 5. Local features detected with Harris detector.

2.5 Harris-Laplace and Harris-Affine detectors

With Harris corner detector, scale changes make detecting more difficult. If scale change is known, detected corners can be scaled, but if the scale change is unknown, the only option is to use multi-scale approach, where corners are detected in multiple scales. This approach also has a disadvantage: it is not feasible to have ten times more features because of the scale adaptation.

Another challenge is a viewpoint change. If images of the same scene are taken from different viewpoints, neighborhood of a local feature changes. This Harris detector cannot adapt to. To overcome these limitations, Harris-Laplace, scale invariant local feature detector, and later Harris-Affine, a scale and affine invariant detector have been developed by Mikolajczyk and Schmid [9].

2.5.1 Automatic scale selection

Scale invariant local features can be obtained by searching stable features in many scales, by building a *scale-space presentation* of an image [20]. A scale-space presentation contains a family of smoothed images with many different *scales*, σ. Different resolution levels L are given by convolutions with the Gaussian kernel:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{12}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2}\, e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{13}$$

where $G(x, y, \sigma)$ is the Gaussian kernel in the scale σ and I represents the original image. While the original scale being σ_0, scales for various levels are obtained by:

$$\sigma_n = k^n \sigma_0 \tag{14}$$

where n is a scale level and k is the factor of scale change between two successive scales. In the original version of Harris-Laplace detector, values for the initial scale σ_0 and the factor k between the scales, were set to 1.5 and 1.2, respectively. [21] The scale-space presentation can be used to find the *characteristic scale* for a local feature. The characteristic scale is found in local maxima of $F(\mathbf{x}, s_n)$ where F is the normalized Laplacian:

$$\mid LoG(\mathbf{x}, \sigma_n) \mid = {\sigma_n}^2 \mid L_{xx}(\mathbf{x}, \sigma_n) + L_{yy}(\mathbf{x}, \sigma_n) \mid . \tag{15}$$

Since the response of a derivative of Gaussian decreases as the σ increases, to compensate the weakening response, the derivatives are multiplied with σ. Since Laplacian operator uses second derivatives, the normalization is done using σ^2 as shown in Eq. 15. The full description of the scale-adaptation is given in [8].

The same automatic scale-selection method is used for both Hessian- and Harris-based scale-invariant and affine-invariant detectors. For demonstration, in Fig. 6, features detected with Hessian-Laplace in two images at different scales. The same characteristic scale is recognized for all features in the both images. The Hessian threshold is set to 4000 to give only the strongest responses, making the illustration clearer.

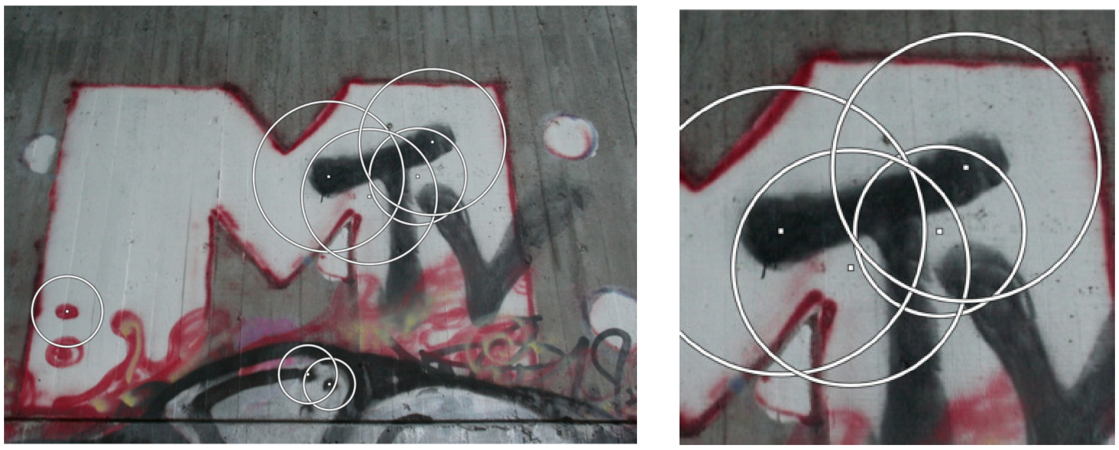


Figure 6. Hessian-Laplace features detected in two scales.

2.5.2 Affine adaptation

Extending scale-invariant detector to be affine-invariant makes it possible to detect the same features when a picture of a same scene is taken from an arbitrary location. That may introduce changes in the scale, orientation and shape of a local feature. An affine-invariant detector is able to find the same features despite the affine transform.

Affine adaptation algorithm used by Mikolajczyk and Schmid contains several steps and is based on properties of the second moment matrix and affine transform. The full description of the adaptation process is described in [22, 9]. In the following, the most important steps are shown.

The following description of the Harris-Affine procedure is directly from [1]:

1. Detect the initial region with the Harris-Laplace detector.
2. Estimate the affine shape with the second moment matrix.
3. Normalize the affine region to a circular one.
4. Re-detect the new location and scale in the normalized image.
5. Go to step 2 if the eigenvalues of the second moment matrix for the new point are not equal.

The main goal of affine adaptation is to find regions from the images that are related through an affine transform A. For a given point $\mathbf{x}$, that is present in both images referred as R and L, the relation is then

$$\mathbf{x_R} = A\mathbf{x_L}. \tag{16}$$

To make it possible to find same structures invariant to the transform, a local feature that is affine-invariant, must be normalized. The second moment matrices $\mathbf{M}_R$ and $\mathbf{M}_L$ for two features representing the same local structure, found in two different images, contain the shape information about the feature in two arbitrary positions. The normalized features are identified by the both eigenvalues being equal, i.e. the ellipse is normalized to a circle. The second moment matrix holds the shape information and the eigenvalue equalization can be obtained by taking square root of the matrix: $\mathbf{M}^{1/2}$. Therefore, using their properties, the relation can be written

$$\mathbf{M}_R{}^{1/2}\mathbf{x}_R = R\mathbf{M}_L{}^{1/2}\mathbf{x}_L \tag{17}$$

where R is the rotation between the two features. To obtain affine invariant points and regions, an iterative algorithm is used. The iterative algorithm is used to find features that have equal eigenvalues. The algorithm forms a shape for the local feature with equal eigenvalues. In each step a new second moment matrix, that has more equal eigenvalues, is calculated. Convergence criterion ϵ_C of the iterative algorithm is based on the *isotropy measure*:

$$1 - \frac{\lambda_{min}(\mu)}{\lambda_{max}(\mu)} < \epsilon_C. \tag{18}$$

The isotropy measure can have values in the range $[0...1]$. Value of 0 means that the eigenvalues are equal, and the bigger the value goes, the bigger is the difference. Typical allowed error used in the convergence criterion, is: $\epsilon_c = 0.05$.

Illustration of normalizing ellipses is shown in Fig. 7. Both neighborhoods of the features, $\mathbf{x}_L$ and $\mathbf{x}_R$, shown in Figs. 7(a) and 7(c) are normalized using second moment matrices

$\mathbf{M}_L$ and $\mathbf{M}_R$:

$$\mathbf{x}_L \longrightarrow \mathbf{M}_L^{(1/2)}\mathbf{x}'_L \quad (19)$$

$$\mathbf{x}_R \longrightarrow \mathbf{M}_R^{(1/2)}\mathbf{x}'_R. \quad (20)$$

After normalization, skew and stretch are removed and there is just pure rotation R between the two normalized neighborhoods of features (Figs. 7(b) and 7(d)):

$$\mathbf{x}'_L \longrightarrow R\mathbf{x}'_R. \quad (21)$$

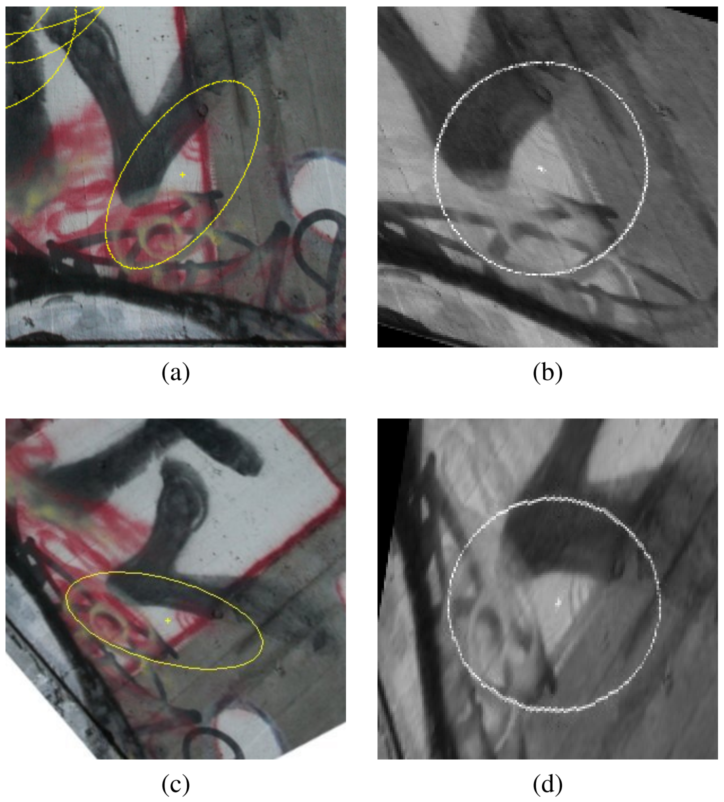


Figure 7. Harris-Affine features are normalized to circles. After normalization of ellipses (a) and (c), shapes of the two features in (b) and (d) are similar.

2.6 Hessian-Laplace and Hessian-Affine detectors

Hessian-Laplace and Hessian-Affine use the same techniques as their Harris corner-based counterparts, Harris-Laplace and Harris-Affine. The difference is that Hessian-detectors

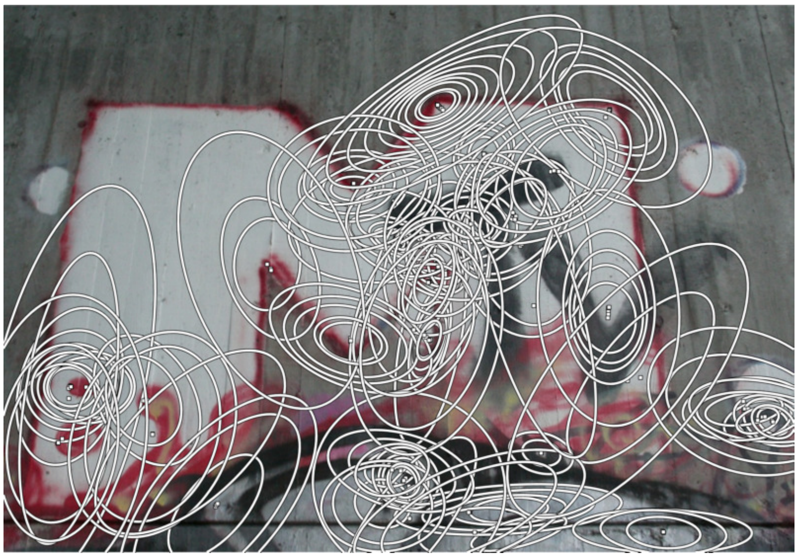


Figure 8. Local features detected with Harris-Affine detector.

are based on the Hessian matrix described in Section 2.3 instead of Harris corners. Blobs detected using Hessian-Affine are shown in Fig. 9. It can be seen in Figs. 8 and 9 that in practice many of the detected locations seem to be same for both detectors. Still, the both methods have different definitions and usually the center points of detected features are not exactly the same. Some parts in images may be formed such way that both Harris-Affine and Hessian-Affine features are found from the same locations.

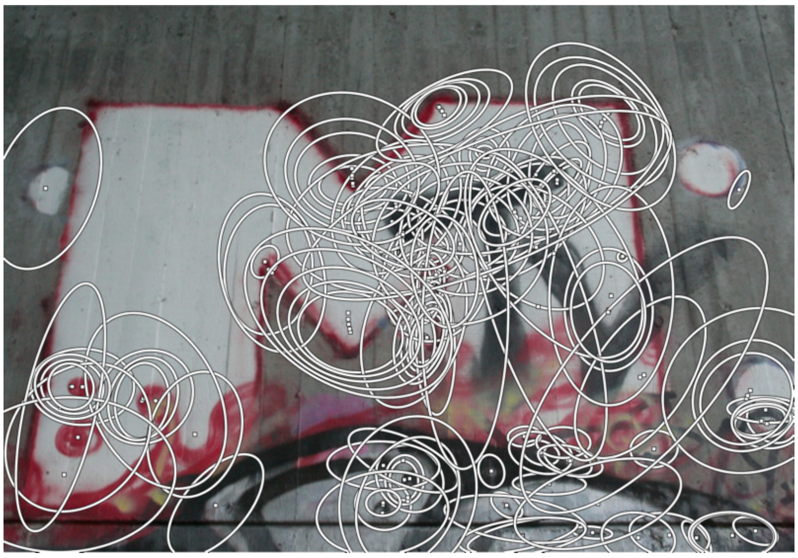


Figure 9. Local features detected with Hessian-Affine detector.

2.7 Maximally Stable Extremal Regions (MSER) detector

Maximally Stable Extremal Regions, MSER, was proposed by Matas et al. [10]. The algorithm finds areas where intensity change is minimal, i.e. areas that are constantly brighter or darker than an outer boundary of a region. The idea of the algorithm is in sequential thresholding the image with all possible values, i.e. $\mathcal{S} = \{0, ..., 255\}$ for 8-bit gray-scale image. The maximally stable extremal regions stay spatially stable, i.e. shape does not change.

The computational complexity of the original algorithm is $O(n\ log(log(n)))$, i.e. almost linear. A more efficient version with the worst-case $O(n)$ is proposed by Nistér and Stewénius [23].

Challenging part with MSER is potential complexity of found regions. Typically found regions are converted to ellipses and information about the shape is lost. In Fig. 10 MSER regions (a) and fitted ellipses (b) are shown. In the example pair it is clearly visible that some of the regions can be represented correctly using ellipses and some not. However, in the matching task the most important property is spatial stability of the fit, i.e. ellipse description should be invariant to affine transformation.

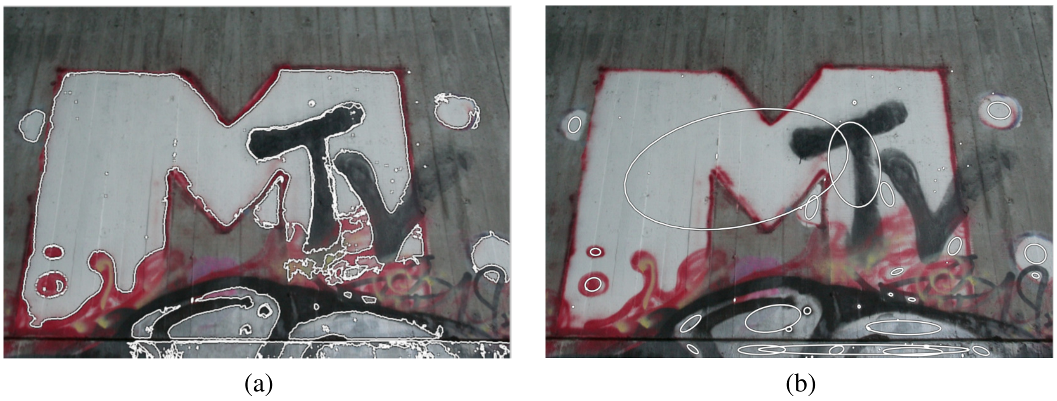


Figure 10. Local features detected with MSER. (a) The original MSER regions and (b) the fitted ellipses.

2.8 Speeded-Up Robust Features (SURF) detector

Speeded-Up Robust Features have been proposed by Bay et al. [11]. It is intented to be extremely fast but still not sacrifice detector or descriptor performance. Low computational complexity is achieved by using *integral images*, introduced by Viola and Jones [24]. They provide a way to calculate responses for box-type filters in constant time. After building the structure in the beginning, response for any boxfilter of any size inside the image can be built in constant time:

$$\sum = A - B - C + D. \tag{22}$$

An example image is shown in Fig. 11.

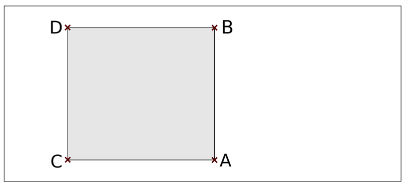


Figure 11. With integral images, only four operations are needed to calculate the sum of intensities inside any rectangular area in the image.

The base technique for local feature detection in SURF is the Hessian matrix, shown in Eq. 1. In the case of SURF the Hessian matrix is approximated roughly using simple box filters of size 9×9. The box filter approximations are shown in the right side of Fig. 12. Responses similar to those shown in Fig. 2 are computed with:

$$det(\mathbf{H}_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2. \tag{23}$$

Weight w is used for balancing the effect of the terms in the equation. In SURF, constant $w = 0.9$ is used is used for all the scales [11].

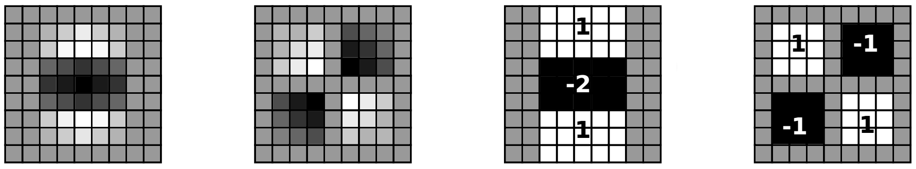


Figure 12. From left to right: the discretized second-order gaussian derivatives L_{yy} and L_{xy} and SURF approximations for L_{yy} and L_{xy}.

A scale-space representation is needed to detect features in various sizes. As with DoG, the scale-space is divided to octaves. An octave is a set of images filtered with increasing kernel size. In each octave, the filter is scaled up by the factor of 2. The fast filter response calculation allows SURF to scale up the filters instead of down-scaling the image. The scales for first three octaves are shown in Fig. 13. Numbers in the boxes are filter sizes used, i.e. for the first octave, fhe filter sizes are 9×9, 15×15, 21×21 and 27×27. The filter size doubles on each octave (as shown in the x-axis), i.e. on the first octave the filter sizes are increased by 6 in each step, in the second octave by 12 in each step and finally in third octave by 24 in each step.

Finally local features are selected as local maxima in $3 \times 3 \times 3$ neighborhood in the scale-space. A fast method for non-maximum suppression proposed by Neubeck and Van Gool is used to locate these extrema points [25].

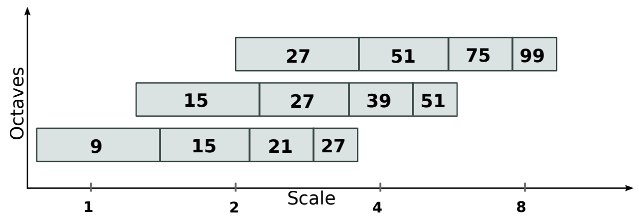


Figure 13. The filter sizes of the first three SURF octaves. Each octave contains 4 filters with various sizes. The form of the filters is square, i.e. filter size 15 means that filter size 15×15 is used.

2.9 Laplacian-of-Gaussian (LoG) and Difference-of-Gaussian (DoG) detectors

Laplacian-of-Gaussian proposed by Lindeberg [8] is based on the second derivatives of the Gaussian. Difference-of-Gaussian provides a good approximation for Laplacian and is more widely used because of its better computational performance [12]. The use of the LoG or DoG approach eliminates the need of calculating the derivatives in x and y directions [26]. Plots of both functions are shown in Fig. 14.

The first phase in Difference-of-Gaussian approach is to build a pyramid of filtered images for the scale-space analysis. The pyramid consists of octaves which are images subsampled by the factor of 2. An octave consists of images convolved with the Gaussian kernel using multiple increasing values of σ. The edges are then found in images formed by subtracting the image convolved with $(k-1)\sigma$ from the images convolved with $k\sigma$. DoG-pyramid is shown in Fig. 15. The approximation of Laplacian-of-Gaussian can be written as

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G \tag{24}$$

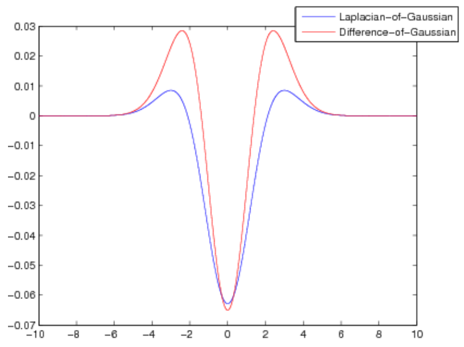


Figure 14. Difference-of-Gaussian can be used as an approximation of Laplacian-of-Gaussian.

where $k = \sqrt{2}$. Lindeberg has shown that the Laplacian must be normalized by the factor σ^2 to achieve true scale invariance [8]. An example image in two scales and a DoG image are shown in Fig. 16.

The edges can be identified as local maxima in the scale-space. Layers of a scale-space are illustrated in Fig. 15. Each pixel in each DoG-layer is compared to 9 surrounding pixels on the layer above, 8 pixels on the same layer and 9 pixels on the layer below. If the pixel has higher value than any of its neighbors, it is selected as a local maximum. The

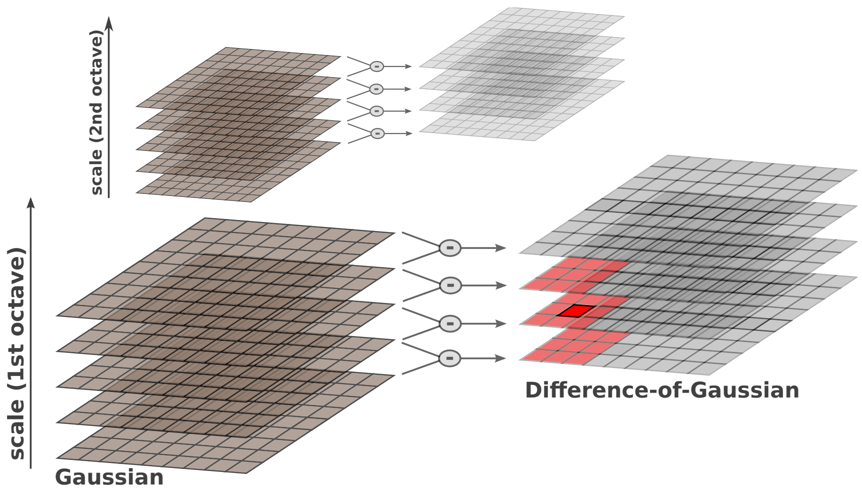


Figure 15. The pyramid contains images smoothed by Gaussian with scale $k\sigma$. Each higher octave contains smoothed images subsampled by the factor of 2. Difference-of-Gaussian images are calculated from adjacent smoothed images.

Laplacian approximation used gives a strong response along the edges. In those locations the local maxima may be weak because the intensity change occurs only in one direction. For that reason, redundant responses must be eliminated to have feasible results. The elimination is based on the properties of the Hessian matrix $\mathbf{H}$:

$$\mathbf{H} = \begin{bmatrix} I_{xx}(\mathbf{x}) & I_{xy}(\mathbf{x}) \\ I_{xy}(\mathbf{x}) & I_{yy}(\mathbf{x}) \end{bmatrix} \tag{25}$$

and Harris cornerness measure R that was discussed in Section 2.4. For DoG, second derivatives in $\mathbf{H}$ are approximated by calculating differences of sample points in the local neighborhood of a maxima. The trace and the determinant for the $\mathbf{H}$ are the sum and the product of the two eigenvalues. Both were also discussed earlier and are given in Eqs. 7 and 8.

As with the Harris detector, only the ratio of the two eigenvalues is needed. The relation between the two eigenvalues λ_1 and λ_2 can the be written as $\lambda_1 = r\lambda_2$. Then,

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} = \frac{(r\lambda_2 + \lambda_2)^2}{r{\lambda_2}^2} = \frac{(r+1)^2}{r}. \tag{26}$$

Now the measurement $(r+1)^2/r$ is at minimum when the eigenvalues are equal. Therefore, the measure can be used to filter out local maxima with the change mostly occurring in one direction. Eliminating the redundant local maxima is then straightforward:

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r+1)^2}{r}. \tag{27}$$

If the result is false, a maximum is discarded. With the measurement, Lowe reports that value $r = 10$ was used for the maximum allowed ratio between the eigenvalues in the original implementation [26].

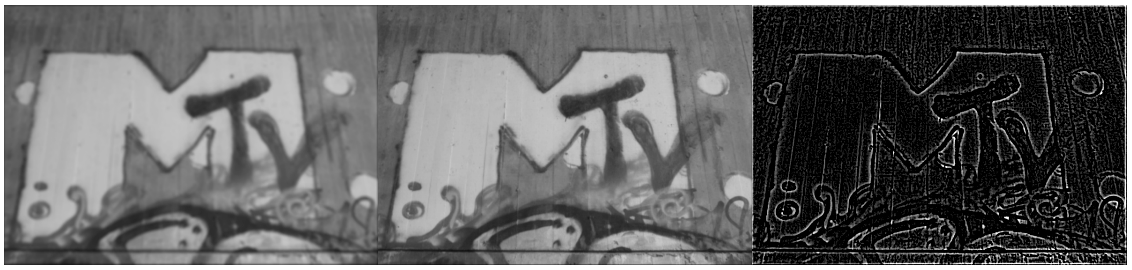

(a) (b) (c)

Figure 16. (a) $I(k\sigma)$, (b) $I(\sigma)$ and (c) $I(k\sigma) - I(\sigma)$.

3 FEATURE DESCRIPTORS

Local feature detectors are used to find areas of interest in the images. A local feature descriptor is used to build a description of a local feature.

3.1 Properties of ideal local feature descriptor

An ideal local feature descriptor has several properties:

- *Distinctiveness:* Low probability of mismatch
- *Efficiency:* Must be computationally efficient to calculate
- *Invariance to common deformations:* Matches should be found even if several of the common deformations are present:
  - image noise
  - changes in illumination
  - scale
  - rotation
  - skew

Although these are features of an ideal descriptor, the methods used in this comparison fullfil these requirement quite well. In the context of object categorization, visual variation of local features makes the task more challenging. The most challenging "deformation" is visual variation of local features between image samples. Features in same spatial locations vary in shape, color and size making the features look different.

3.2 Scale-Invariant Feature Transform (SIFT)

Scale-Invariant Feature Transform (SIFT) is a combination of a detector (DoG, discussed in Section 2.9) and a descriptor, SIFT key. It is a robust descriptor that can provide highly distinctive description of local features extracted using DoG or some other detector. In the following, construction of SIFT descriptors is explained.

In this phase, local features are already detected using the DoG detector. When SIFT features are detected, scale-space presentation is used to find local extrema, i.e. characteristic scales for features. A detected feature belongs then to some level σ of the pyramid and to a pixel location $L(x, y, \sigma)$ of the scale-space.

The gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$ are computed in every pixel location $L(x, y, \sigma)$ belonging to a selected neighborhood of a detected local feature. The computation is made using simple pixel differences [26]:

$$m(x, y) = \sqrt{\left(L\left(x+1, y\right) - L\left(x-1, y\right)\right)^2 + \left(L\left(x, y+1\right) - L\left(x, y-1\right)\right)^2} \tag{28}$$

$$\theta(x, y) = tan^{-1}\left(\frac{L\left(x, y+1\right) - L\left(x, y-1\right)}{L\left(x+1, y\right) - L\left(x-1, y\right)}\right). \tag{29}$$

An orientation histogram containing 36 bins for a feature is formed to cover gradient directions and magnitudes in sectors surrounding a feature. The whole 360 degrees around a feature is covered, using 10 degrees for each bin. Gradient samples are weighted by their distance using Gaussian-weighted circular window with $\sigma = 1.5$ and by their magnitude. Each of the 36 bins contains a measurement of intensity change in one direction. The highest of those measurements is selected as dominant orientation of the descriptor. If other peaks with a value over 80% of the highest are found, they are considered as additional descriptors and found local maxima are used as the dominant orientations of the additional descriptors.

The descriptor construction is illustrated in Fig. 17. Arrows in the first part of the image present the gradient magnitudes and orientations calculated earlier. They are later rotated according to the dominant orientation. The circle around the local pixel neighborhood illustrates the Gaussian weights that are applied to gradients to make the nearest gradients the most significant. In the middle of the figure, a 2×2 SIFT descriptor is shown. Each of the four cells contains accumulated gradients to 8 directions calculated from 4×4 sample array. Although other sizes can be used, usually 4×4 sample arrays are used with SIFT, because they are reported to give the best results [26]. A final built SIFT descriptor is a vector with 128 dimensions. All vector components are 8-bit unsigned integers, i.e. range of values $\mathcal{S} = \{0, ..., 255\}$.

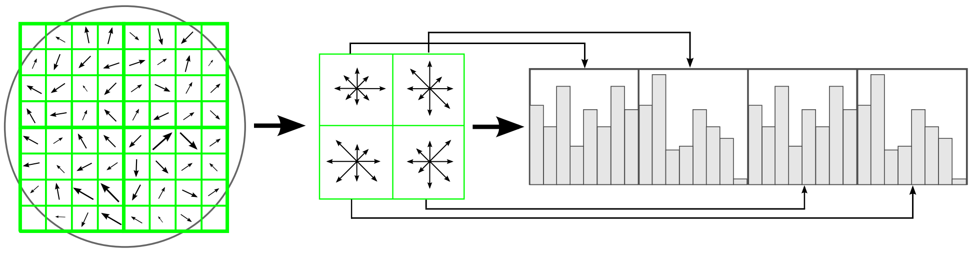


Figure 17. Building a SIFT descriptor using gradients around a detected feature. First, gradients are weighted using a Gaussian (the circle in the image). Then, in the middle, gradient orientations in 4×4 grid are accumulated to form the descriptor. Finally a normalized histogram containing the magnitudes to each direction in each cell can be computed.

3.3 Speeded Up Robust Features (SURF)

Building a SURF descriptor is very fast as the feature extraction discussed in Section 2.8. Integral images are exploited also in descriptor building to calculate filter responses very quickly. SURF descriptor is rotation-invariant. Invariance is achieved by finding reproducible orientation for the local neighborhood of a feature. When the scale of a detected feature is s, Haar wavelet responses for circular neighbordhood of size $6s$ are calculated. Haar wavelets are simple filters shown in Fig. 18. After calculating the filter responses, the local neighborhood is weighted with Gaussian, $\sigma = 2$, to make the nearest intensity changes the most significant.

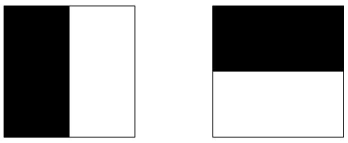


Figure 18. Haar-wavelets. Responses to Haar-wavelets are very efficient to calculate using integral images.

Wavelet responses are handled as points in space, as shown in Fig. 19. X- and Y-axes represent response strengths in horizontal and vertical directions, respectively. A sliding window of size $\frac{\pi}{3}$ (gray sector in the figure) is used around the feature surroundings to calculate sum of horizontal and vertical response strengths. Sums of response strengths are used to calculate a local orientation vector for each direction. The longest such vector is finally selected to represent the dominant orientation of a descriptor.

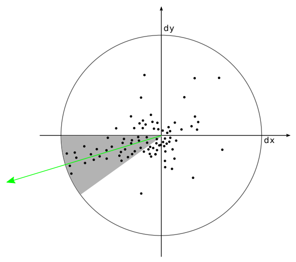


Figure 19. Wavelet response strengths in horizontal and vertical directions are shown as dots in space. Gaussian-weighted responses inside the sliding window are summed to form a local orientation vector. The longest such vector (green arrow) is selected to represent the dominant orientation.

Responses of Haar wavelets are used also in building the actual descriptor. First an area around the keypoint (detected in scale s) of size $20s$ is selected. That region is split up into 4×4 sub-regions. For each sub-region, Haar wavelet responses are calculated for

5×5 blocks from a grid of sample points. In practice, however, to decrease computational complexity, the wavelet responses are calculated using the unrotated image and then approximated for various descriptor orientations as needed.

All 16 sub-regions that form the whole region are shown in Fig. 20. First, as the circle in the figure denotes, the responses around the keypoint are Gaussian-weighted ($\sigma = 3.3s$). Each of these 16 sub-regions contain 2×2 smaller regions where response strengths are summed. A feature vector $\mathbf{v}$ calculated from these response strength sums of sub-regions is then:

$$\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|). \tag{30}$$

When the 16 vectors are combined, $16 \times 4 = 64$ dimensional vector is formed. By definition, SURF sums are invariant to illumination changes. To be invariant to contrast (scaling factor), a feature vector is finally turned into a unit vector, i.e. the vector is divided by its length:

$$\hat{\mathbf{v}} = \frac{\mathbf{v}}{\|\mathbf{v}\|}. \tag{31}$$

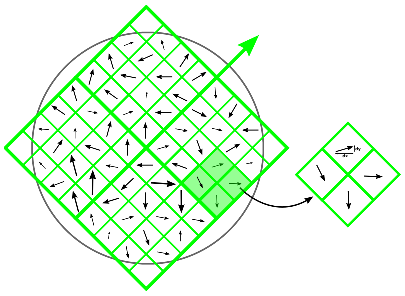


Figure 20. A SURF descriptor is divided into 16 sub-regions that all contain 2×2 smaller regions. Sums for these sub-regions (like detached square in the figure) are formed to build feature vector $\mathbf{v}$, as shown in Eq. 30.

4 DETECTOR AND DESCRIPTOR PERFORMANCE EVALUATION

Performance evaluation of local feature detectors and descriptors is a challenging task. Mikolajczyk et al. have done studies on both detectors and descriptors [27, 13] and their evaluation protocol for detectors was used for detector evaluation. For descriptor performance, a novel test scheme was developed. In the following, Mikolajczyk's approach is explained and later differences to this work.

4.1 Mikolajczyk's test protocol for detectors

For detector performance evaluation, a test protocol developed by Mikolajczyk et al. [27] was used. In the original comparison, there were several images from a few scenes. Various deformations were present in the images, such as changes in viewpoint angle, changes in scale, blur, JPEG compression and changes in lighting conditions. The performance evaluation in that case meant how well the same local features could be found in the same spatial locations despite a deformation, i.e. how well the detectors can cope with deformations that are often present in the real world images.

4.1.1 Ground truths

In Mikolajczyk's test, a reference image is compared to several deformed images. There is a known, pre-calculated affine homography between the images compare. Features of each deformed test image are then projected to the reference image for comparison.

Examples of the test images are shown in Fig. 21. In these sample images, the deformation between samples is rotation. In Fig. 21(a), there is a reference image. In Figs. 21(b) and 21(c) there are the first two images that features of the reference image are tested against. In the original test various types of scenes were used. For rotation, scale change, viewpoint change and blur tests two types of test scene images were used. One of the scenes was structured and containing well-distinctive regions, as a graffiti shown in Fig. 21. The other one contains finer texture-like structures, such as bricks on a wall or leaves in a tree. All the test images are available in the Internet [28].

Figure 21. Examples of viewpoint change test images used in [13].

4.1.2 Correspondences

When features from two images are compared, there are two groups of ellipses in various sizes and poses. An exhaustive search is done to find overlapping ellipses. In this work, as in Mikolajczyk's original tests, 40% overlap error was allowed. In overlap measurement, the sizes of ellipses have an effect on results. The bigger the ellipses are the smaller is the overlap error in the measurement. For that reason, all the ellipses are normalized to a radius of 30 pixels before calculating the overlap error. The effect of rescaling ellipses for the overlap measurement is illustrated in Fig. 22. The effect of growing relative size of the overlapping gray area is clearly visible.

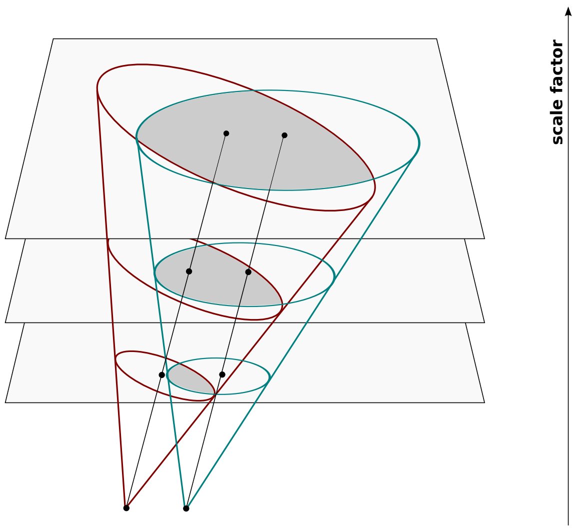


Figure 22. Rescaling an ellipse has an effect on the overlap measurement.

Number of correspondences is a measurement that tells the number of ellipse pairs found whose overlap error is below 40%. It is not usually feasible to use this measurement by itself, because it does not take into account the percentage of the overlapping ellipses.

4.1.3 Repeatability

Repeatability is the most important measure for detector comparison. When the number of correspondences is known, it is straightforward to calculate the *repeatability rate*:

$$repeatability\, rate = \frac{\#\, of\, correspondences}{min(\#\, of\, reg\, in\, img\, A, \#\, of\, reg\, in\, img\, B)} \cdot 100\%. \qquad (32)$$

For the region count, only regions present in both images are included. Features in both images are projected to each other using a known homography, and some of the features are not necessarily present in both images after the transform.

4.1.4 Testing procedure

First, features are detected in all test images in a set using all the detectors. Then, the features in the first image are compared to each other image in the set. Features detected in both images are projected to each other using the known homography between the images. Features that are not present in both images are discarded. For features present in both images, correspondences are searched by comparing each ellipse to all ellipses in the other image. If overlap error for two ellipses is less than 40%, a correspondence is found. When all the correspondences are found, it is straightforward to calculate the repeatability rate using Eq. 32.

4.2 Detector evaluation in this work

There are several differences in evaluation in this work to Mikolajczyk's original comparison. The assumption is that features found in the reference image are found also in the other image in the same spatial location. Similar assumption was made in this work when detector performance was evaluated. There are sample images belonging to certain classes, such as motor bikes, faces or dollar bills. It is assumed that spatially similar locations in these samples look similar and local features sharing approximately the same shape can be found.

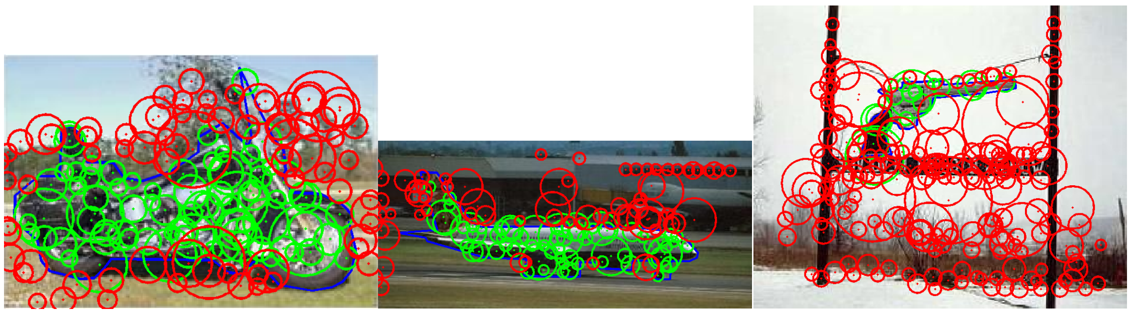
4.2.1 Image pairs

In Mikolajczyk's comparison, features in an other, deformed image were compared to features in a reference image of that test set to see how well the detectors can cope with several types of deformation. In this work, there is no reference image. In the main tests there are 25 randomly selected image pairs from each of the ten classes. Features are then detected in both of the images in a pair.

4.2.2 Object outlines and landmarks

In this work, only objects in the test images are considered. In the test images outlines of the sample objects are known. There are $2 \times n$ matrices containing n (x, y)-pairs for locations of the object contour. In the detector tests, features with centroids outside of the object outline are discarded. Features outside the object outlines contain no information about the objects. Some random corresponding features could be found in some cases, but it has no use. In Mikolajczyk's test protocol, only features that are present in both images after features are projected to each other, are selected. In this work, objects are fully visible even after features in the first image have been projected to the other one. And since only features inside the object outlines are accepted, no checking needs to be done for presence of the features left.

Three examples of accepted and discarded SURF features are shown in Fig. 23. Object outline is drawn in blue. Features are drawn in green if inside and in red if outside of an object outline.



(a) (b) (c)

Figure 23. The selected SURF features are shown in green, object outlines in blue and discarded SURF features in red. Sample images are representing classes (a) motorbikes, (b) airplanes and (c) revolvers.

The ground truths are used to eliminate features outside of the object contour because object detection is not in the scope of this work. The other ground truth information is "landmarks" which are marked in every sample image to annotate semantically similar parts of the sample objects. A few examples of landmark locations are shown in Fig. 24.

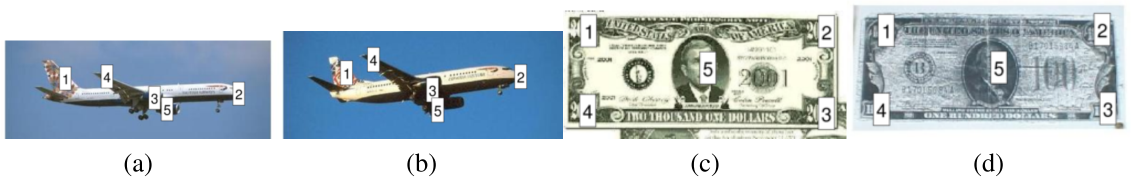


Figure 24. Landmark examples for (a)-(b) two airplanes and (c)-(d) two dollar bills.

4.2.3 Affine transform

The performance evaluation of detector methods is based on finding features in same spatial locations from two samples of a class forming an image pair. If a pose and location of the object are not exactly the same in two images, the assumption that features are located in same parts of the image is not true. To overcome this limitation, landmarks are used to calculate an estimate for unknown affine homography between the two sample objects. An estimate of an affine homography was also used in Mikolajczyk's original comparison.

An affine transform is a weak perspective transform. In a projective transform, straight lines stay straight. If parallel lines remain parallel, an affine transform with six degrees of freedom is enough. An affine homography between two point sets $\mathbf{x}$ and $\mathbf{x}'$ is defined as [29]:

$$\mathbf{x}' = \mathbf{H}_A \mathbf{x} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x} \tag{33}$$

where parameters of $\mathbf{A}$ determine a rotation, shearing, scaling and parameters of $\mathbf{t}$ determine a translation. An unknown affine homography between two point sets can be acquired exactly if three point correspondences are defined. If less points are used, there are many solutions and if more are used, an exact solution may not exist. An exact solution $\mathbf{H}_A$ for three points can be defined as [29]:

$$\begin{bmatrix} x'_1 & x'_2 & x'_3 \\ y'_1 & y'_2 & y'_3 \\ 1 & 1 & 1 \end{bmatrix} = \mathbf{X}' = \mathbf{H}_A\mathbf{X} = \mathbf{H}_A \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \Rightarrow \mathbf{H}_A = \mathbf{X}'\mathbf{X}^{-1} \tag{34}$$

where $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$ are the original points and $\{(x'_1, y'_1), (x'_2, y'_2), (x'_3, y'_3)$ are the corresponding points in a transformed domain. As visible in Eq. 33, the last row of $\mathbf{H}_A$ always contains values 0, 0 and 1 in affine transform.

Landmarks are sets of (x, y)-pairs describing certain parts of objects in a class that can be used to estimate a homography between two sample objects. As usually five or six correspondences are defined by landmarks, it is preferable to use the information from every point. Direct linear transform (DLT) can be used to estimate an unknown homography. Similarity equations are formed as homogeneous linear equations that can be solved [29]. In this work for calculating homography estimations, publicly available homography estimation toolbox was used [30].

An example image pair is shown in Fig. 25. Two object samples are shown in Figs. 25(a) and 25(b). Blue lines around the objects are the outlines. Red circles in both images (a) and (b) are features that are discarded because their location is outside of the object outline. In Fig. 25(c) red circles are the original features shown in green in Fig. 25(b). The green circles in Fig. 25(c) are the circles projected from Fig. 25(a). It is clearly visible that the transform overlays the features adequately. The spatial information can be used e.g. to validate descriptor matches.

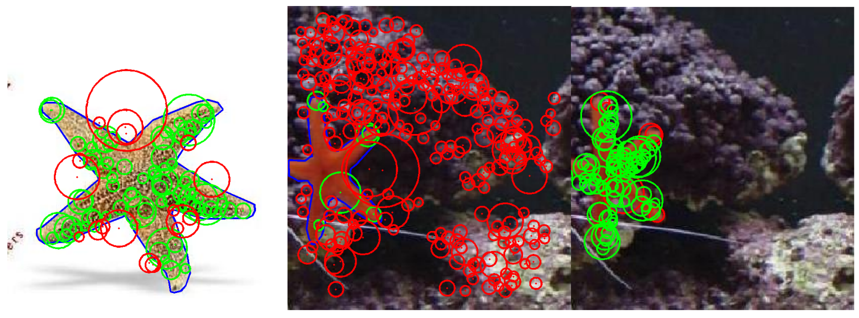

(a) (b) (c)

Figure 25. Accepted features (green) from (a) are projected to the other image (b) in the pair to overlay both features (c). In (c) accepted features from (a) projected to (b) are shown in green color and accepted features in (b) are shown in red color.

4.2.4 Testing procedure

For all image pairs and chosen detectors:

1. Calculate homography between an image pair based on the landmarks
2. Extract local features from the both images
3. Select in the both images only the features whose centroids are located inside the region defined by Caltech_101 outlines
4. Find corresponding regions, i.e. the overlapping ellipses between the two images
5. Calculate the repeatability rate using Eq. 37

There are some number of image pairs randomly selected from a database. Features in all the images are detected using all given detectors. As explained above, homographies for image pairs are estimated using landmarks. A publicly available script from Mikolajczyk is used to calculate correspondences and repeatability rates as explained in Section 4.1.4 [28].

4.3 Mikolajczyk's descriptor evaluation

Several steps of descriptor performance evaluation in [13] are similar to detector performance evaluation described in Section 4.1. In an image set, the first image is used as a reference image. Detected features and their descriptions in other images are compared to the first one. A known homography between an image pair is used to project found features to each other. Features present in both images after the projection procedure are selected and correspondences are searched.

In descriptor performance evaluation, the found feature correspondences are used as ground truth for descriptor evaluation. Descriptor matches are first searched using three methods explained in Section 4.4.1. If also source features (ellipses) of matching descriptions overlap, a match is considered to be correct. If features do not match, a match is a false match.

For all detectors and image pairs:

1. Estimate the homography between the image pair
2. Extract local features from the both images
3. Project features from the first image to the second and vice versa
4. Select features that are present in both images after the projections
5. Calculate overlap errors from each feature to all other features

6. Select features with overlap error below 40% as correspondences
7. Calculate repeatability using Eq. 32

4.4 Descriptor evaluation in this work

The basis for descriptor performance evaluation is different than in the original comparison by Mikolajczyk and Schmid [13]. Their goal was to find best methods for finding matches between images taken from different view points of a scene. In this evaluation the goal is to find out if the matching can be extended to visual object categorization where matches are searched between two samples of a same class.

Descriptor tests are based on matches that are found between two images. Numbers of matches are used to calculate averages and medians for datasets and classes. Also a coverage measure is introduced to see how compherensive the matching performance of a descriptor is.

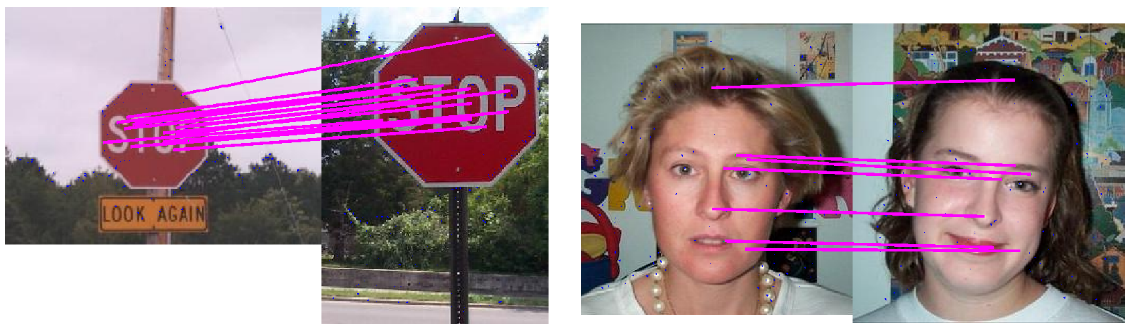
4.4.1 The definition of match

A local feature descriptor is a vector that describes a feature extracted from an image. Often matching descriptors are searched to find corresponding regions from a database or an other image. A matching example is shown in Fig. 26. In Fig. 26(a) stop signs are matched using SIFT-descriptors built on Laplacian-of-Gaussian features and in Fig. 26(b) faces are matched using SURF-descriptors built on SURF features. Pink lines in the example show a connection between two regions that are found to be visually similar enough in both images and fulfill the spatial requirements. When local feature descriptors are matched, distances between descriptors are calculated. Usually Euclidean distances are used and distance s of descriptors $\mathbf{a}$ and $\mathbf{b}$ is defined as

$$s(\mathbf{a},\mathbf{b}) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2} \tag{35}$$

where n is the number of dimensions, e.g. for the SIFT it is 128.

Mikolajczyk tested three matching strategies. If a simple threshold is used, distances to other descriptors are calculated and if distance is below the threshold, a match is found. In the nearest-neighbor strategy, only the nearest neighbor of all the other descriptors is



(a) (b)

Figure 26. A matching example containing (a) LoG-vireo+SIFT-vireo descriptor matches on stop signs and (b) SURF+SURF descriptor matches on faces.

considered as a match, if the distance to it is below the threshold. The problem of these thresholds is that it is not trivial to select a threshold value, since the distance variation between descriptors is usually quite high. The third option is to use nearest-neighbor strategy with relative threshold, using the ratio of distances between the nearest and second-nearest neighbors.

In this work, the last one is used. The threshold value is 1.5. For every descriptor, distances to all descriptors from the other image are calculated and sorted. If the distance to the nearest descriptor is smaller than 1.5 times the distance to the second-nearest, a match is found.

There are two images from the Caltech car side class. They form one of the pairs in the dataset. In Figs. 27(a) and 27(b), the results for different parameters used in the test are shown. Spatial distance limit s is 0.05, i.e. 5% of the image diagonal, and descriptor distance threshold d is 1.5. The number of matches is low, but the significance of the matches is high. It can be seen in Figs. 27(c)-(h) that when the descriptor match threshold is lowered, the number of matches goes up quite fast. When that is combined with less strict spatial limit, the results can be seen in Figs. 27(i) and 27(j). Most of the “matches” are implied by the small structures of the background.

4.4.2 Spatial validation of matches

Matches are defined as explained above. In pure descriptor matching there is no information about spatial location of descriptors in the sample images. Usually it is preferable that matches are semantically correct, i.e. “correct” matches often represent similar structures in sample images. In Fig. 26 this kind of visual similarity of structures is clearly visible. The situation where less strict spatial and descriptor matching threshold values generate matches that do not share visual similarity is shown in Fig. 27.

Since pure descriptor matching does not imply any spatial restrictions for matches, all descriptor matches are validated after candidate matches have been found. As with features earlier, matching locations of the first image in an image pair are projected to the second one using a known homography between the images. After the transform, Euclidean distance s between the two locations is calculated. Since sample images vary in size, image

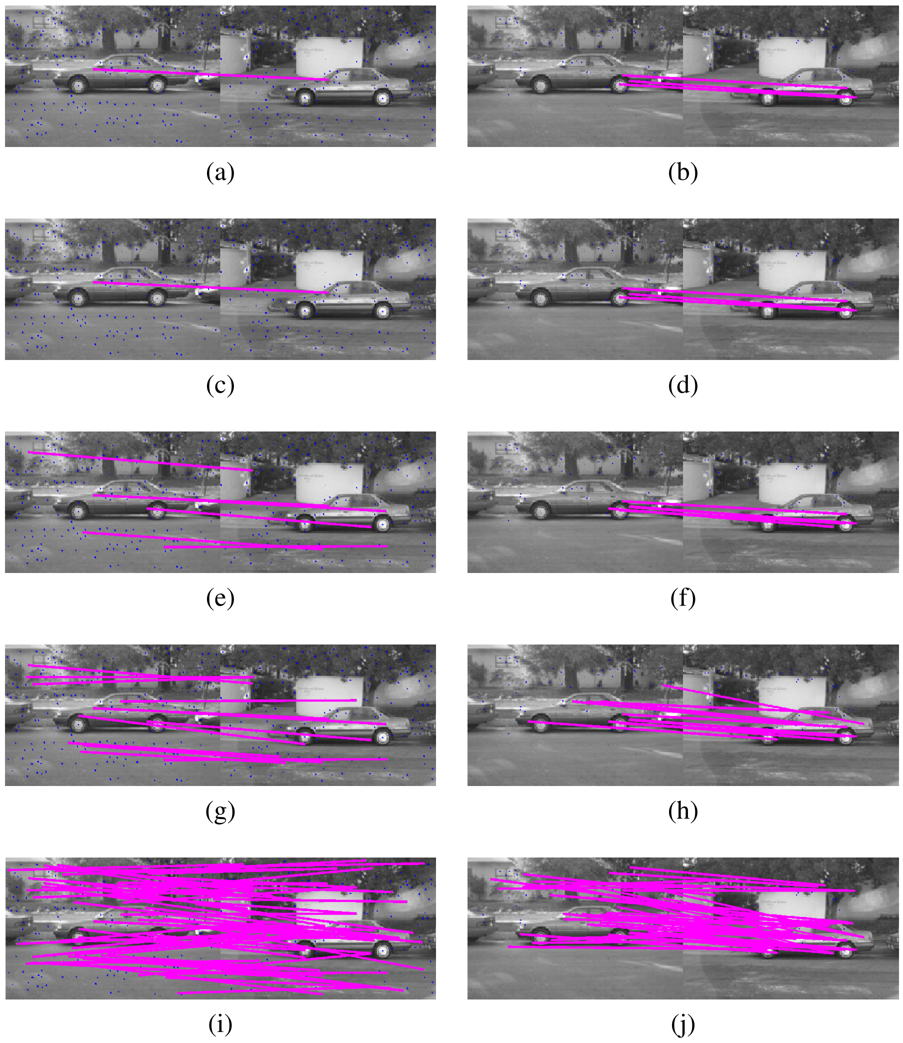


Figure 27. Example matches using SIFT descriptor on the left and SURF descriptor on the right side, with various parameters: (a) SIFT $d = 1.5, s = 5\%$, (b) SURF $d = 1.5, s = 5\%$, (c) SIFT $d = 1.5, s = 10\%$, (d) SURF $d = 1.5, s = 10\%$, (e) SIFT $d = 1.2, s = 5\%$, (f) SURF $d = 1.2, s = 5\%$, (g) SIFT $d = 1.2, s = 10\%$, (h) SURF $d = 1.2, s = 10\%$, (i) SIFT $d = 1.0, s = 15\%$ and (j) SURF $d=1.0, s=15\%$.

diagonal d is used for threshold scaling. A coefficient k is used to tune test settings. If the following is true, a candidate match is accepted as validated match:

$$s < kd. \tag{36}$$

The value for k in the experiments was 0.05, i.e. 5% of the image diagonal.

4.4.3 Number of matches and coverage

Performance of the descriptors is determined using plain numbers of matches. Number of matches found for each image pair in a testset using each detector and descriptor combination are calculated. Averages and medians can be calculated for individual classes and a whole testset. The most important figures are per-class median and average number of matches. E.g. per-class average tells how many matches in average is found from a sample pair belonging to a class. Using these figures, it is simple to find out which is the best detector for a given class or to see which descriptor gives the best results for a whole testset.

A *coverage* is an other way to express descriptor performance. It is a number of sample image pairs with at least one match found. It is also used in combination with numbers, i.e. coverage-5 tells a number of image pairs with at least 5 matches.

4.4.4 Testing procedure

For all image pairs and chosen detectors and descriptor pairs:

1. Calculate homography between an image pair based on the landmarks
2. Detect local features in both images and build descriptors for detected regions
3. Select in the both images only the features whose centroids are located inside the object outline
4. Find matching descriptors
5. Project the features of the first image to the other image
6. Validate matches using spatial distance between two matching features

5 EXPERIMENTS

Experiments were divided into two separate test setups. The intention of the first test was to measure the performance of the local feature detectors. In the second test, the performance of the local descriptors was measured. For both the tests, MATLAB scripts were written to run the tests. In the detector test also code from K. Mikolajczyk [27] was used.

5.1 Data

Caltech-101 is a popular dataset for object recognition and categorization systems performance evaluation. It contains objects belonging to 101 categories. There are 40-800 sample images per category. Object outlines for every sample are annotated. [31]

All the tests in this work were done using 250 image pairs from Caltech-101 dataset. Both tests require “landmarks” to be annotated beforehand. The landmarks are manually marked locations in images that can be used to estimate the transform from an image to another. Landmarks are available for ten classes, and 25 image pairs from each class were randomly selected to build the dataset.

Example images from Caltech-101 are included as Appendix 1. Two examples of images pairs for each class are shown in Fig. A1.1. There are four images in each row. The first two images in each row form an image pair and the next two form an other pair. All these samples are real image pairs from the dataset used in this work.

5.2 Setup for the local feature detector tests

The test setup for detector performance evaluation includes required binaries for detector implementations from various sources, and both MATLAB and C code from Mikolajczyk to evaluate the performance of a detector. A MATLAB script was written to control test execution, calculate affine transforms between the image pairs and to visualize detector performance. When comparing the detectors, the first measure is how many *corresponding regions*, i.e. ellipses that can be found in the both images after the regions in the first image are projected to the second image and vice versa. For the two ellipses that are considered to be corresponding, overlap error must be less than 40%.

It is important to notice that ellipses were normalized to an even size, a diameter of 30 pixels, before calculating the overlap to eliminate the effect of ellipse size. The other restriction is that only features with centroid point inside the object outline are considered. Others are discarded, because corresponding object parts want to be found and randomly corresponding background features are not object parts.

The most important measure that is used for comparing the detectors is the *repeatability rate*. The formula is given in Eq. 37. The number of correspondences in both images are compared, and the smaller of the numbers is used as minimum when calculating the repeatability. In this case only the features that are present in the scene in the both images after the transform are considered.

$$repeatability\, rate = \frac{\#\, of\, correspondences}{min(\#\, of\, reg\, in\, img\, A, \#\, of\, reg\, in\, img\, B)} \cdot 100\% \qquad (37)$$

The number of correspondences found is a raw measure and gives additional information about the results of repeatability comparison. The repeatability measurement tends to give better results if the number of correspondences is higher. If such problem is believed to bias the results, the number of correspondences can be used to determine what is actually causing the high score.

5.3 Selected detector implementations

Nine different detectors were tested. The detectors are based on five different approaches. There were three implementations of Hessian-Affine, referred to as *hesaff*, *hesaff-alt* and *hesslap-vireo*. The two preceeding ones are from Mikolajczyk, hesaff-alt being the original implementation that was used in [27]. Hesaff is a newer version that is implemented using a slightly different parameters. Hesslap-vireo is implemented by Zhao [32] and is only a partial implementation of Hessian-Affine and is thus referred as Hessian-Laplace. *Harlap-vireo* is based on the Harris corners and it is a partial implementation of Harris-Affine.

Detectors referred as *dog-vireo* and *sift* use Difference-of-Gaussian to approximate the Laplacian. The *log-vireo* uses real Laplacian values instead of an approximation. Dog-vireo and log-vireo are implemented by Zhao [32] and sift is vlfeat's implementation [33].

The other detectors included were SURF and MSER, referred as *surf* and *mser*. The set of detectors is selected based on their performance on earlier studies and in the preliminary tests for this comparison. All the selected detectors are listed in Table 1.

Table 1. Selected detectors for detector evaluation.

Detector	Referred as	Implementation
SURF	surf	ETH [11]
MSER	mser	featurespace [34]
DoG	sift	vlfeat [33]
Hessian-Affine	hesaff	featurespace [34]
Hessian-Affine	hesaff-alt	featurespace [34]
LoG	log-vireo	LIP-VIREO [32]
Hessian-Laplace	hesslap-vireo	LIP-VIREO [32]
Harris-Laplace	harlap-vireo	LIP-VIREO [32]
DoG	dog-vireo+sift-vireo	LIP-VIREO [32]

5.4 Detector results

In the first test the detectors were evaluated. In the detector test results seemed quite good, as average repeatability rates for the whole dataset were commonly in the range of $20 - 35\%$, as shown in Fig. 28. In Figs. 28(b)-(c) average and median repeatabilities are shown for each class and for the whole dataset. In Figs. 28(c)-(d) numbers of correspondences are shown. The repeatability is the most important measure, but it is important to notice that the number of correspondences is also useful for interpretation of the results and identifying properties of the detectors.

The top three detector implementations were dog-vireo, surf and hesslap-vireo, with a bit over 30% of average repeatability rate. Dog-vireo and surf got the best repeatability rates in most categories. Hesslap-vireo was almost as good in repeatability, but outperformed the other two clearly in the number of correspondences. Therefore, hesslap-vireo was the best detector. The performance of the next four, dog-vireo, surf, hesaff and log-vireo was almost equal. Within these four, dog-vireo and surf got the best repeatability rates, but higher number of corresponding features was detected by hesaff and log-vireo.

After the best five detectors, also the next three could be identified. Harlap-vireo, hesaff-alt and sift got almost identical results for repeatability rate, around $20 - 25\%$, while number of corresponding features was the highest for harlap-vireo. In the test clearly the worst detector was mser. It got the lowest scores for the both repeatability and number of correspondences.

More detailed test results for individual classes are shown in Appendix 3. Repeatability rates are shown in Fig. A3.1 and numbers of correspondences in Fig. A3.2. The results are shown using boxplots, i.e. the most important characteristics of the data are shown in the plots. Whiskers, i.e. vertical dashed lines with horizontal endpoints, show the minimum and maximum values of the data. Measurements considered as outliers are marked with crosses in the plots. Boxes over the whiskers are used to show percentiles of 25th and 75th, meaning that 25 percent of the measurements are below the 25th percentile. Similarly 25 percent of the measurements are above the 75th percentile. Median values of measurements are shown as horizontal lines inside the boxes.

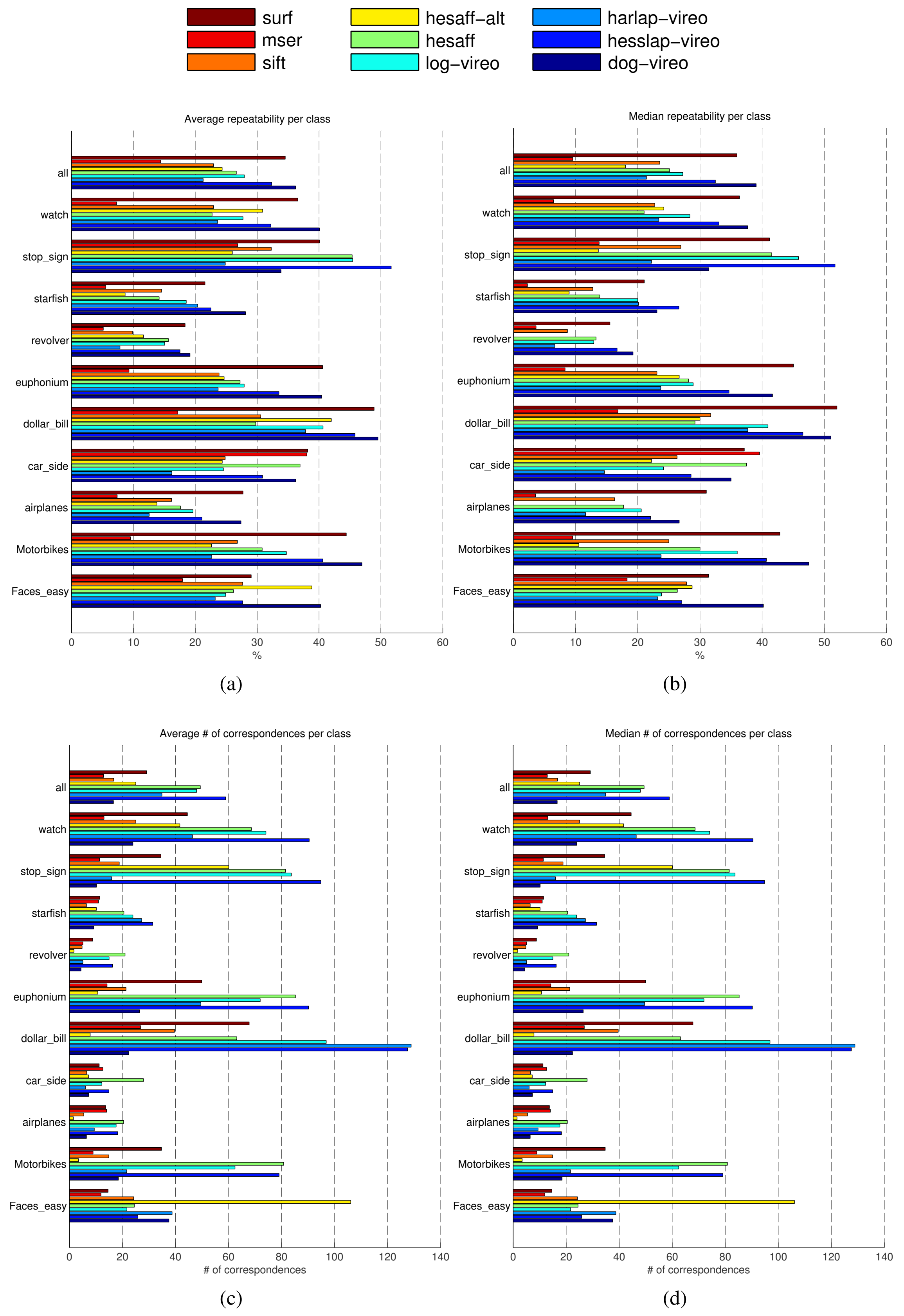


Figure 28. (a) Average and (b) median repeatability rates for classes. (c) Average and (d) median number of correspondences for classes.

5.5 Setup for the local feature descriptor tests

A test protocol was needed for the descriptor test, and Mikolajczyk's test procol used in [27] would have been an obvious choice. However, it turned out that the Mikolajczyk's test protocol was not suitable for this work, since matching performance was worse than expected from the beginning. If Mikolajczyk's test protocol would have been used, the small number of matches compared to number of all features would have made comparison of various methods hard. The chosen, newly-developed approach, gives results which make comparison of various methods straightforward. The task in Mikolajczyk's comparison was to see what is the best of the several good methods. In this work the question was that can any of these methods can provide more than just a few random matches?

Usually when descriptors are used in object recognition or similar tasks, more than a few matches are required. Higher number of matches makes any decisions more reliable. For that reason, the descriptors are evaluated based on the number of matches that can be found between the two images that form a pair. Preliminary tests showed that numbers for the actual test will probably be quite low and for that reason the approach is just to look for number of matches.

5.6 Selected descriptor implementations

Six detector and descriptor combinations were chosen to this evaluation. SIFT is an obvious choice, since it is the most popular one and used widely. Two implementations of SIFT were used, vlfeat's [33] and Zhao's [32] implementations, referred as sift and sift-vireo. An other descriptor method used is SURF. All the selected detector and descriptor pairs are listed in Table 2. First combination, sift+sift uses vlfeat's implementations of

Table 2. Selected detector and descriptor pairs for descriptor evaluation.

Detector	Descriptor	Referred as	Implementation
SIFT	SIFT	sift+sift	vlfeat [33]
Hessian-Affine	SIFT	hesaff+sift	featurespace [34]
MSER	SIFT	mser+sift	featurespace [34]
DoG	SIFT	dog-vireo+sift-vireo	LIP-VIREO [32]
Hessian-Laplace	SIFT	hesslap-vireo+sift-vireo	LIP-VIREO [32]
SURF	SURF	surf+surf	ETH [11]

SIFT descriptor and DoG detector. Hesaff+sift and mser+sift features and descriptors are acquired by using the binary used in Mikolajczyk's tests [27]. Dog-vireo+sift-vireo and hesslap-vireo+sift-vireo are referring to Zhao's implementation of many popular detectors and descriptors, lip-vireo [32]. In dog-vireo+sift-vireo, algorithm similar to original SIFT is used. Hesslap-vireo is a partial implementation of Hessian-Affine, and the combination hesslap-vireo+sift-vireo is very similar to hesaff+sift. Surf+surf is a combination of the SURF detector and descriptor.

The GLOH is an extension to SIFT that performed well in [13] and [17]. It involves dimensionality reduction using PCA, representing the original 272-dimensional vectors using only 128 dimensions. To be able to use PCA successfully, a training is necessary. Since different data was used in [13], the basis file provided was not suitable for the data used in this work. Several technical problems made it not possible to use GLOH. Therefore, the GLOH was left out from the tests.

5.7 Descriptor results

In the second test descriptors were evaluated and statistics about successful matches were collected. In Fig. 30 descriptor test results are shown.

In Figs. 30(a)-(b) coverage-1 and coverage-5 are shown. Since there were 25 image pairs from each class, the maximum result for both coverages is 25. Also an average performance for each method is given and is referred as "all" in the coverage figures. Surf got the highest coverage-1, but coverage-5 and average number of matches show that hesaff+sift is the best combination. With surf, it was possible to find a few matches in even quite difficult cases, but in less difficult cases it did not perform as well has Hessian-affine-based combinations hesaff+sift and hesslap+sift-vireo.

In Figs. 30(c)-(d) average and median matches per class are shown. Average and median over all 250 samples are referred as "all". The highest average, 2.4 matches per image pair was achieved by hesaff+sift, but median was 0, coverage-1 being $124/250$, i.e. for more than half of the image pairs the result was 0 matches. On the other hand, for surf the median was 1, while coverage-1 was $144/250$ and average number of matches was 2.1. Dog-vireo+sift-vireo and sift+sift performed almost equally compared to hesaff+sift, but got in easier categories significantly less matches. As in the detector test, mser+sift was the worst combination. It seems that it is less suitable than others to this kind of task, because the found regions cannot be exactly the same.

The overall performance of descriptors indicates that any of these methods is not suitable for this kind of task. In the results, surf's better tolerance for noise was visible through some matches in very difficult matching pairs. In Figs. 29(a)-(d) it is clearly visible that coverage results decreased quickly as more matches were required. For coverage-1 the highest value was 144 by surf but for coverage-5 only hesaff+sift was able to get a result over 50, i.e. from only a bit over 20% of the image pairs at least 5 matches were found.

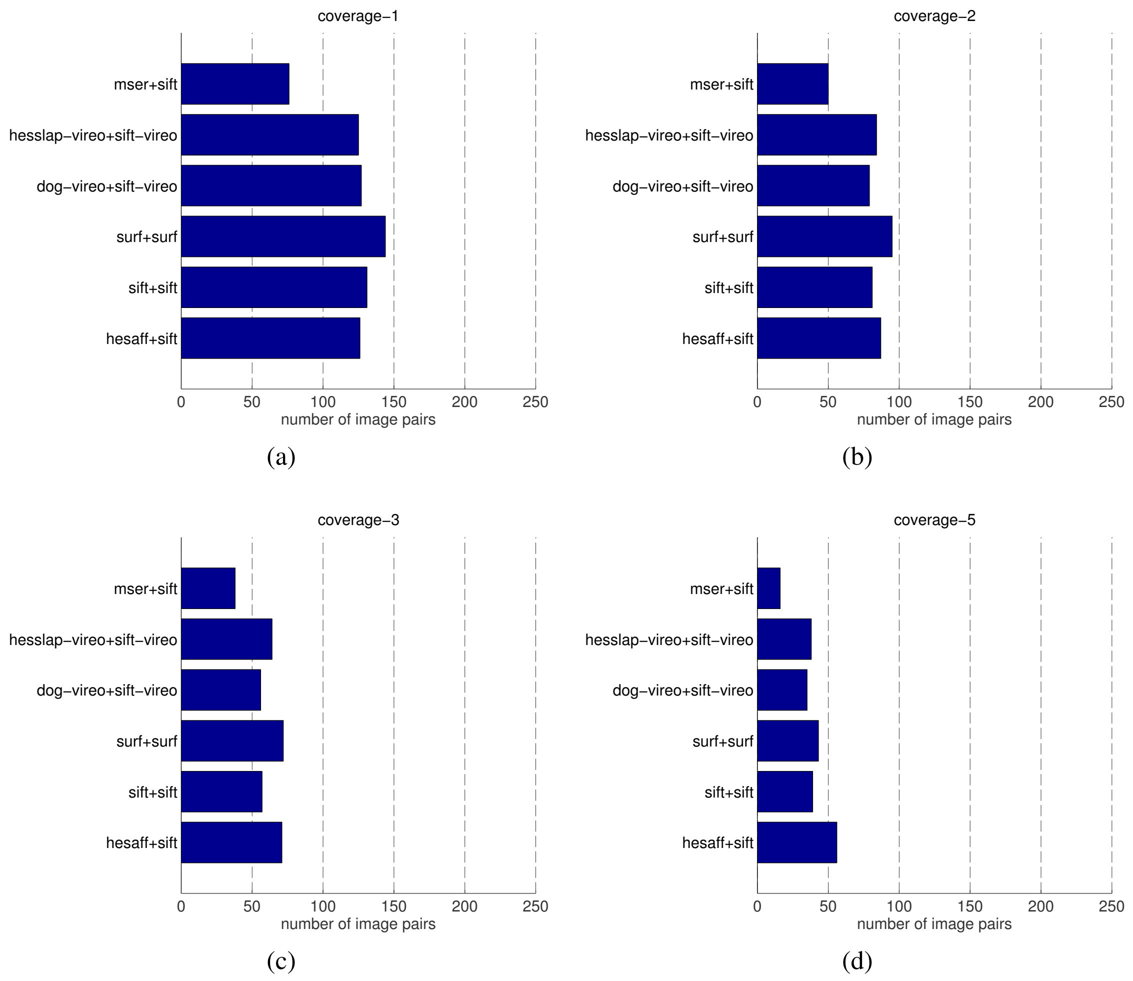


Figure 29. Coverages for (a) 1, (b) 2, (c) 3 and (d) 5 matches required.

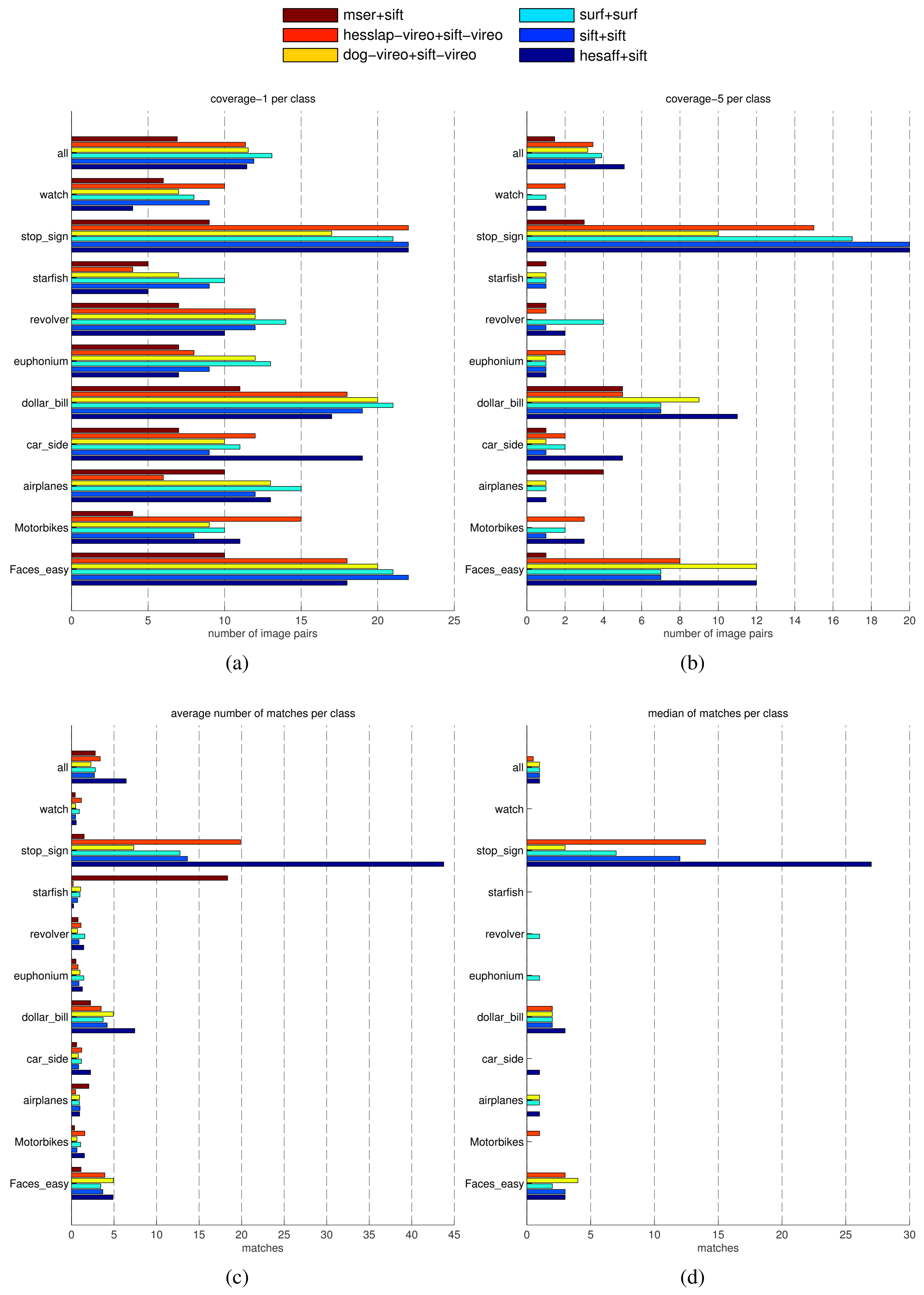


Figure 30. (a) Coverage-1 and (b) coverage-5 per class. (c) Average and (d) median number of matches per class.

5.8 Descriptor combinations

Since the number of matches was very low, combinations of methods were tested. In Fig. 31 performance boost is shown for the best single descriptors and the best combinations of two and three descriptors. In Fig. 31(a) average matches per image are shown. For each single descriptor the average was ≈ 2 and with two and three methods, the averages were ≈ 4 and ≈ 6, respectively. The median of matches per image behaved similarly, results being 1, 2 and 3 for one, two and three method combinations. In Fig. 31(b) same information is shown using absolute value for the whole dataset. With the best single descriptors $400 - 500$ matches were found. $950 - 1100$ and $1400 - 1600$ matches were found when two and three methods were combined.

In Fig. 31(c) coverage-1 results of all the best combinations are shown. For the best single method, surf, the coverage-1 was $144/250$. For the best combination of two methods, surf+surf and dog+sift-v, the coverage result was $177/250$. The highest coverage-1, $198/250$ was obtained by the combination of surf, dog-vireo+sift-vireo and hesslap-vireo+sift-vireo.

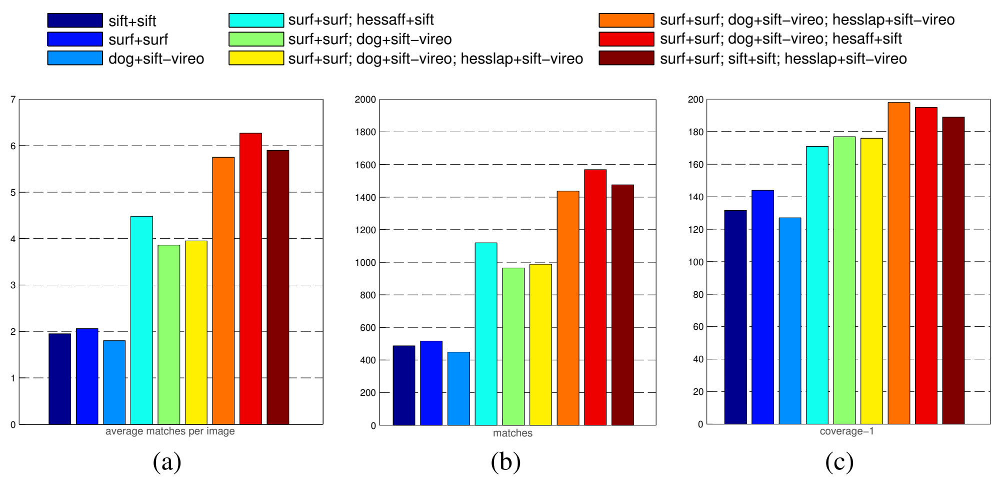


Figure 31. Key figures for the whole dataset: (a) average matches per image, (b) number of matches and (c) coverage-1.

5.9 Verification of descriptor test results using a different testset

To verify the test results for Caltech images, an other set with 119 high quality images was used. There were sample images from 12 categories. From phones category there were nine sample pairs and from all the other categories the number of image pairs was ten. The testing procedure was exactly the same that was used before for the descriptor test. Example images from the dataset are shown in Appendix 2. An image pair from all 12 categories is shown in Fig. A2.1.

Results of the verification test are shown in Fig. 32. The maximum result for per-class coverages is 10 for all the classes but phones, for which it is 9. Category “all” refers to the average over all the categories. As in the main descriptor test, hesaff+sift was the best combination. In coverage-5 hesaff+sift is the best and in coverage-1 the second best. As in the main descriptor test, the best over-all results were obtained in traffic sign classes, stop signs and crosswalk signs.

The second and third best methods were sift+sift and surf+surf which performed almost equally. Coverage-wise these two were approximately as good as hesaff+sift. Hesaff+sift outperformed the others by reaching high number of matches in traffic sign classes. After the top three methods, the other three managed to get almost equal measurements in the test. Each of mser+sift, hesslap-vireo+sift-vireo and dog-vireo+sift-vireo was able to find at least one match from $40 - 50\%$ of all the sample images.

There were some differences to the first descriptor tests. In the verification test, the methods can be divided to two categories based on their performance in the test. In the first descriptor test all the methods performed almost equally, except that mser+sift was clearly the worst. In the verification test mser+sift was not the worst, but belonged to the second group of methods with lower performance.

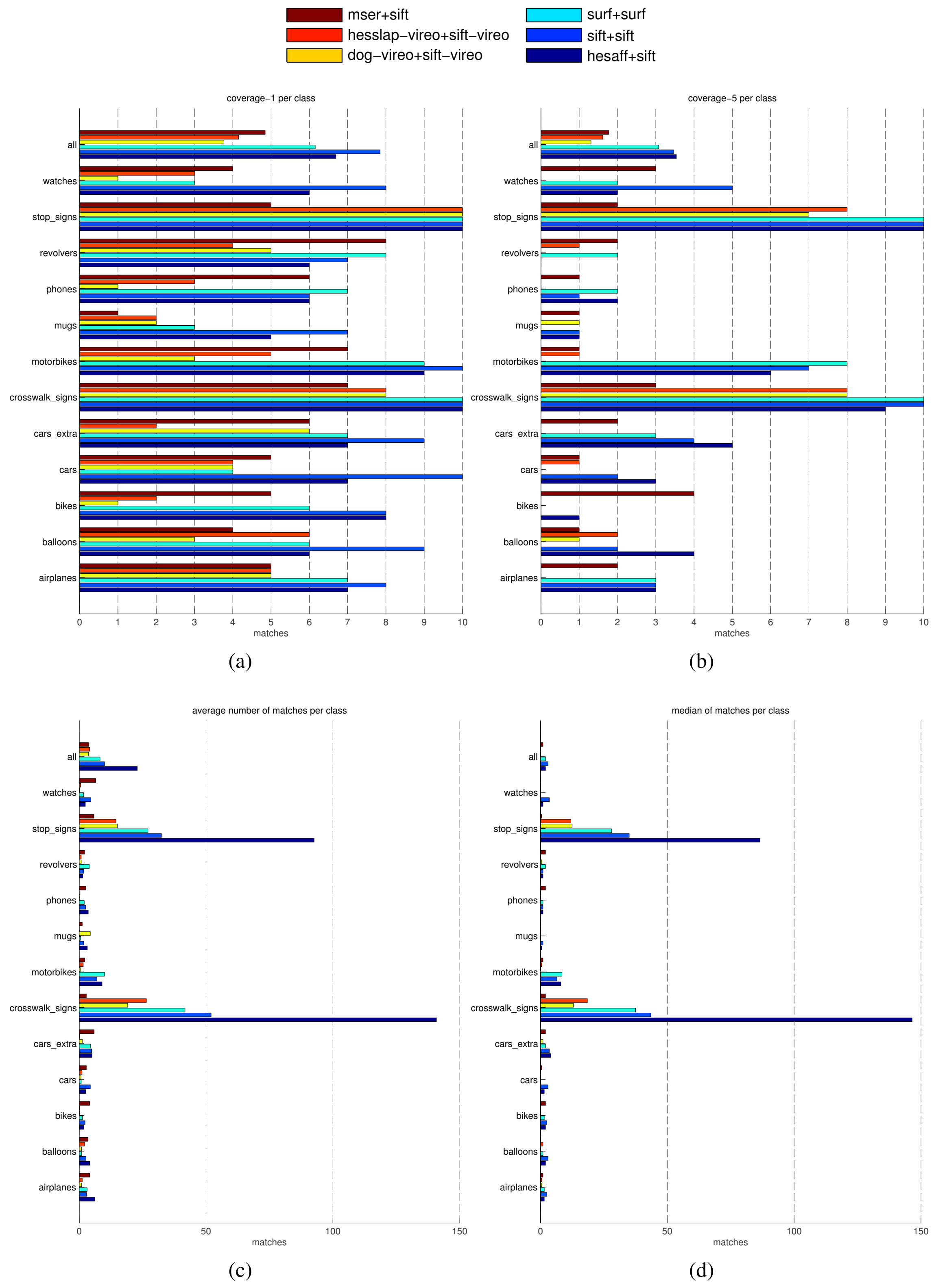


Figure 32. (a) Coverage-1 and (b) coverage-5 per class. (c) Average and (d) median number of matches per class.

6 DISCUSSION

The results for the tests are quite intriguing. The local feature detectors showed to perform quite well in the context of visual object categorization. On the other hand, the performance of the local feature descriptors seemed worse than expected. In the following, the possible reasons behind the both are discussed.

6.1 Detector performance

Local feature detectors do not have any kind of intelligence. They are just based on the intensity changes of pixels in digital images. The detected features are corners, blobs or regions depending on the detector method used. When this kind of technique is used to find similar features from the two different objects in the same category, an assumption is that objects in the same category share at least some sufficiently similar shapes and that these shapes are in approximately same spatial location in 2D images.

The results of the detector test show that the assumption is quite feasible. The repeatability rates are significantly lower than in the original Mikolajczyk's case when matching was done between views of a same scene. For most of the detectors, the median number, i.e. for half of the sample images, number of correspondences found was $15 - 50$ per image pair for the whole dataset and the median repeatability rate $20 - 35\%$, which seems adequate performance.

To verify the results, also a test was done using incorrect image pairs from various classes, i.e. the compared samples were not from the same class. The results show that the repeatability for that kind of verification test set was practically zero for these detectors. Based on that it is rather safe to assume that the repeatability rates for the detectors are not based on random overlaps with the ellipses, but on the fact that objects from the same categories share similarities as assumed in the first place.

6.2 Descriptor performance

While detectors performed quite well in the tests, the match numbers for the descriptors were poor. The medians of matches per image showed that the typical number of matches per image is only one for the best methods. Average is slightly higher, since for example for stop signs, faces and dollar bills the results are significantly better than for the other classes. Classes that have some man-made structures contain many structures that look practically the same in all objects in the class. In caltech-250 dataset the stop signs are a typical example of this kind of object set. It can be clearly seen in Figs. 30(c) and 30(d) that the stop signs are the only category for which the studied descriptors perform moderately. Faces and dollar bills are two other categories with more matches than one. The median is $2 - 3$ matches per image for them, which is still not a sufficient number of matches for most applications.

In other classes the results were even worse, median for the whole dataset being one match per image for the best methods. Combining various methods boosted the performance. The best two and three-method combinations reached medians 2 and 3. Percentage wise the increase seems high, but still a lot more matches would be needed. The insufficient performance of local feature descriptors is such bad, that for many applications sufficient number of matches can not be found using any of the current detector and descriptor pairs.

7 CONCLUSIONS

The goal of this thesis was to find the best local feature detector and descriptor for visual object categorization task by comparing existing methods using a suitable test scheme. Suitability of these methods for such a task was not known exactly beforehand.

Test setups for the both detectors and descriptors were created. For the detectors, the test setup was similar to one developed by Mikolajczyk [27]. Because of the different situation than in [13], no suitable test protocol for descriptor comparison was found. A test scheme was developed, as described in Section 4.4.

For the detectors, the test results were adequate. The repeatability rates were not high if compared to typical wide-baseline scenario, but still $20 - 35\%$. A bit surprisingly the highest scores were obtained by SURF and Zhao's DoG implementation. One of the main discoveries was that there are significant performance variations between various implementations of the same methods.

When it comes to descriptors, the performance was worse than expected. For classes where visual variation between samples is high, only a few matches were found between the image pairs. For classes that share more similarities between samples, like human faces or stop signs, the results were significantly better. In some cases, the performance can be boosted by combining the matches from several methods. The best descriptors were SURF that was the best for difficult categories, and Hessian-Affine that performs well for the easier categories. The interesting fact is, that the well performed SURF and DoG are computationally the fastest ones. In addition to being very fast, SURF is able to cope very well with the noise [11], and in this case, with visual variation of the samples.

The results of this work imply that a better descriptor for visual object categorization purposes is needed. The current performance of the descriptors is not enough for most applications. Could the good performance of the detectors imply that a better "voc-descriptor" could be built on them?

REFERENCES

[1] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[2] F. Attneave. Some informational aspects of visual perception. *Psychological review*, 61:183–193, 1954.

[3] P. R. Beaudet. Rotationally invariant image operators. In *Proceedings of the International Joint Conference on Pattern Recognition*, pages 579–583, 1978.

[4] H. Moravec. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. PhD thesis, Stanford University, 1980.

[5] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[6] J. L. Crowley and A. C. Parker. Multiple resolution representation and probabilistic matching of 2-d gray-scale shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(1):113–121, 1987.

[7] T. Lindeberg. Scale-space for discrete signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):234–254, 1990.

[8] T. Lindeberg. Feature detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[9] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 1(60):63–86, 2004.

[10] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 384–393, 2002.

[11] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *International Journal of Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[12] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, pages 1150–1157, 1999.

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

[14] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pages 230–235, 1998.

[15] D. Tell. *Wide baseline matching with applications to visual servoing*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, June 2002.

[16] L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[17] K. Mikolajczyk, B. Leibe, and B. Schiele. Local features for object class recognition. In *IEEE International Conference on Computer Vision*, pages 1792–1799, 2005.

[18] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. In *Proceedings of the International Conference on Computer Vision*, pages 230–235, 1998.

[19] C. Schmid, R. Mohr, and C. Bauckhage. Comparing and evaluating interest points. extended version. `ftp://ftp.inrialpes.fr/pub/movi/publications/schmid_iccv98_ext.ps.gz` (Retrieved: August 2010).

[20] A. P. Witkin. Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[21] K. Mikolajczyk and C. Schmid. Indexing based on scale-invariant interest points. In *Proceedings of the International Conference on Computer Vision*, pages 525–531, 2001.

[22] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the European Conference on Computer Vision*, pages 128–142, 2002.

[23] D. Nistér and H. Stewénius. Linear time maximally stable extremal regions. In *European Conference on Computer Vision*, pages 183–196, 2008.

[24] P. Viola and M. Jones. Robust real-time feace detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[25] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *Proceedings of the International Conference on Pattern Recognition*, pages 230–235, 1998.

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

[27] 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, 2005.

[28] K. Mikolajczyk. `http://www.robots.ox.ac.uk/~vgg/research/affine/` (retrieved: Nov 2010).

[29] J.-K. Kamarainen and P. Paalanen. Experimental study on fast 2d homography estimation from a few point correspondences. Research Report 111, Lapppeenranta University of Technology, Department of Information Technology, 2009.

[30] J.-K. Kamarainen. Homogr toolbox. `http://www.it.lut.fi/project/homogr/`.

[31] L. Fei-Fei, M. Andreetto, and M. Ranzato. Caltech-101 dataset. `http://www.vision.caltech.edu/Image_Datasets/Caltech101/` (retrieved: Nov 2010), 2003.

[32] W. Zhao. LIP-VIREO local interest point extraction toolkit. `http://vireo.cs.cityu.edu.hk` (retrieved: Nov 2010).

[33] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/` (retrieved: Nov 2010), 2008.

[34] K. Mikolajczyk, T. Tuytelaars, J. Matas, C. Schmid, and A. Zisserman. `http://www.featurespace.org` (retrieved: Dec 2010).

Appendix 1. Caltech-101 example image pairs

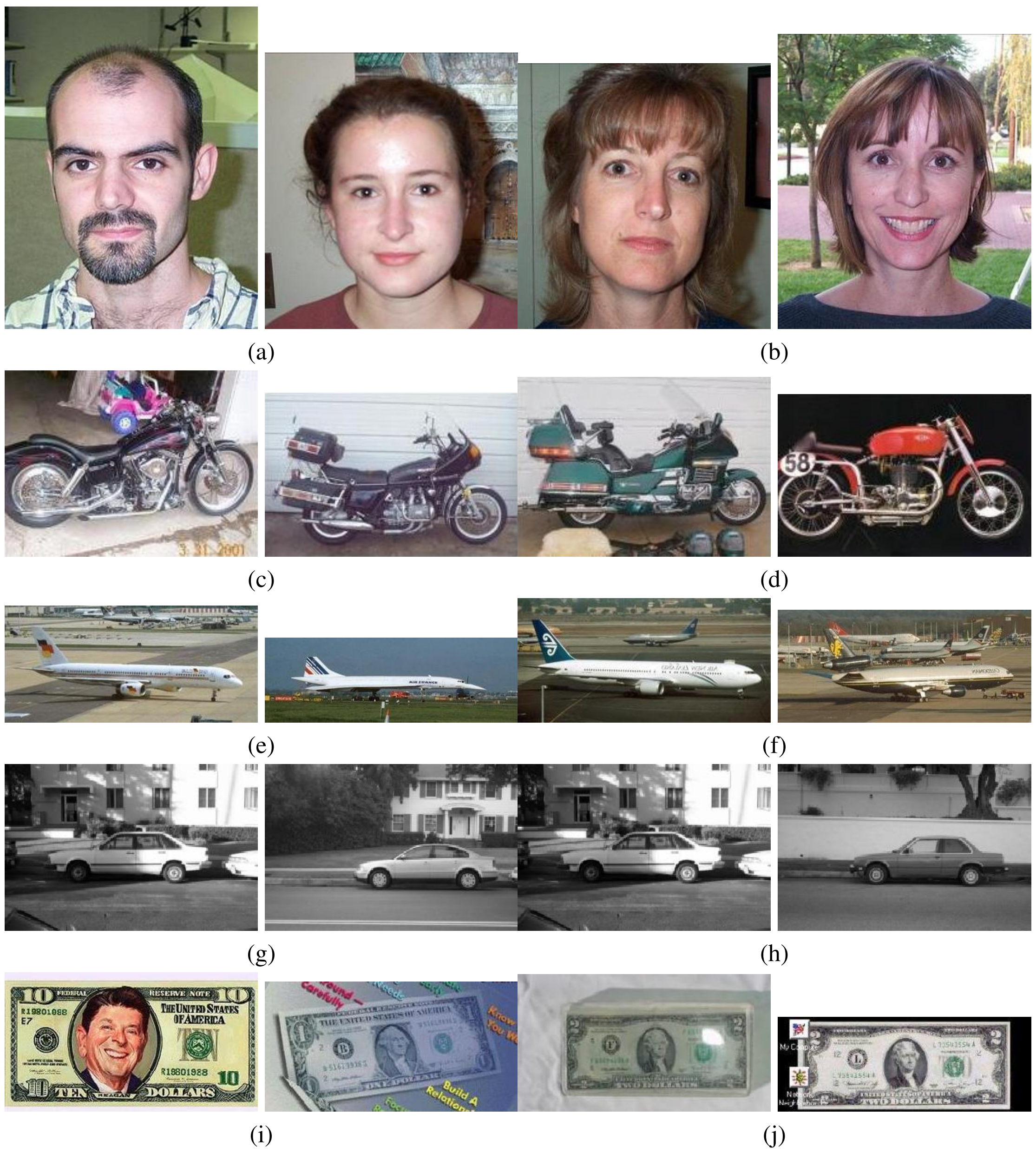


Figure A1.1. Example image pairs from (a)-(b) faces, (c)-(d) motorbikes, (e)-(f) airplanes, (g)-(h) car sides and (i)-(j) dollar bills.

(continues)

Appendix 1. (continued)

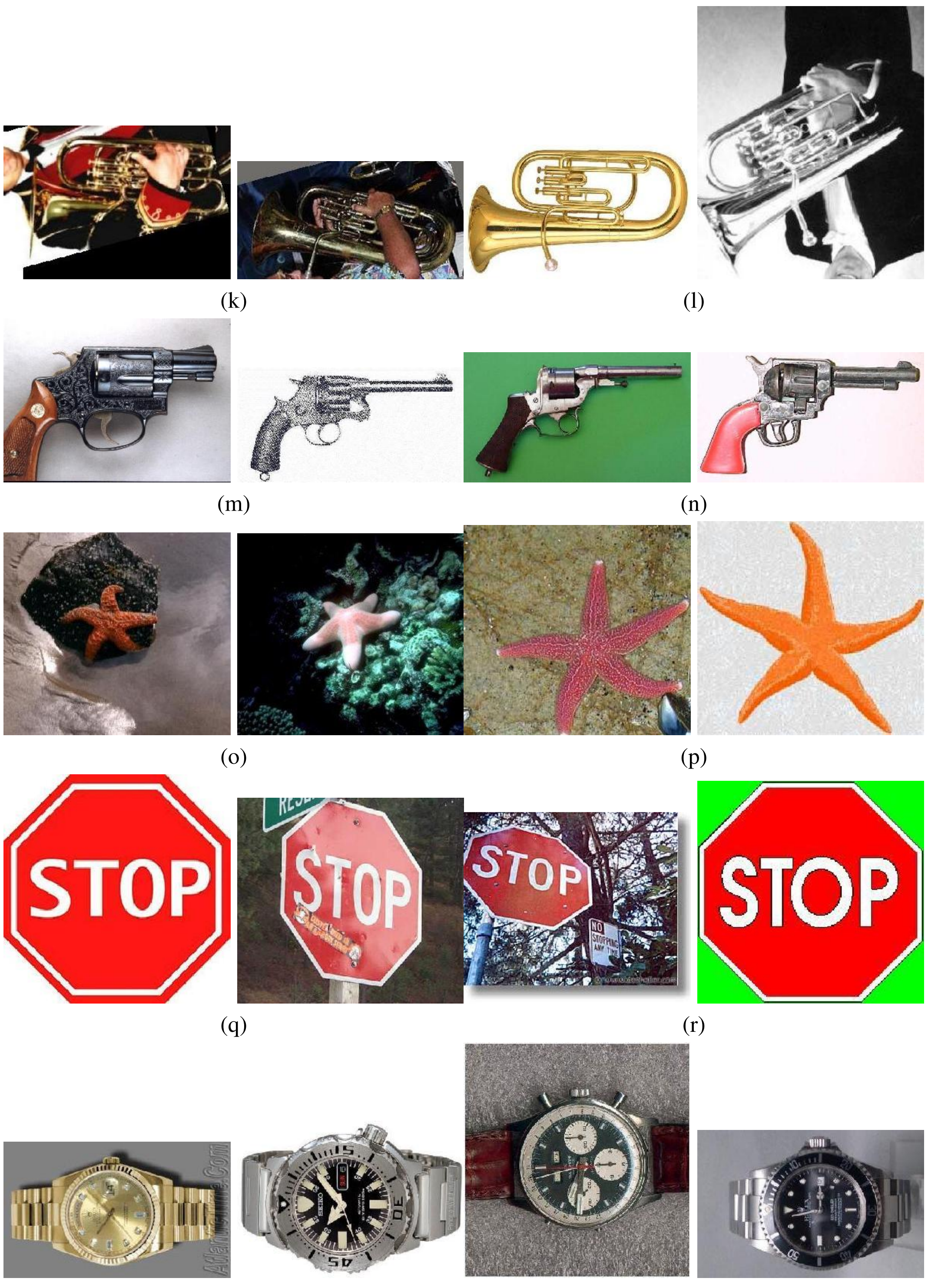


Figure A1.1. Example image pairs from (k)-(l) euphoniums, (m)-(n) revolvers, (o)-(p) star fish, (q)-(r) stop signs and (s)-(t) watches.

Appendix 2. MinnaImageDatabase example image pairs

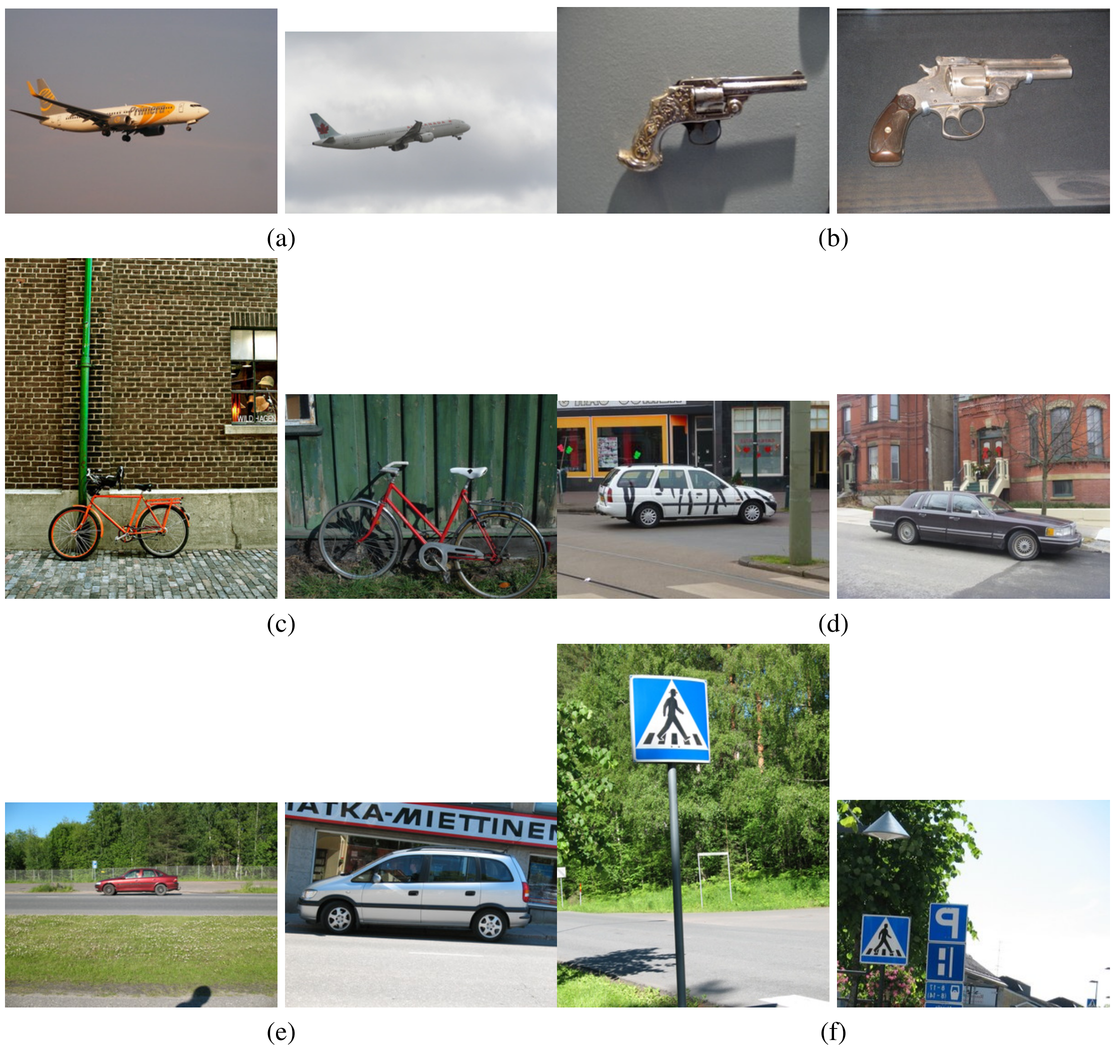


Figure A2.1. Example image pairs from (a) airplanes, (b) revolvers, (c) bikes, (d) cars, (e) cars extra and (f) crosswalk signs.

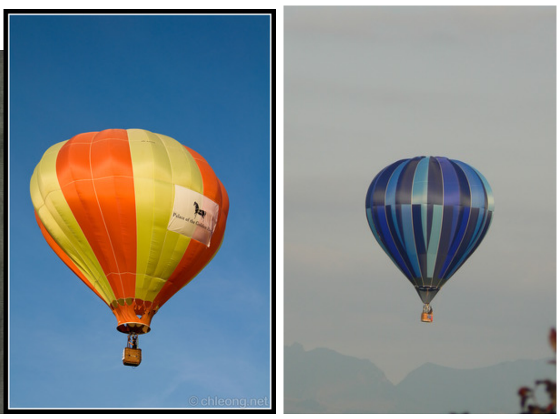
(continues)

Appendix 2. (continued)

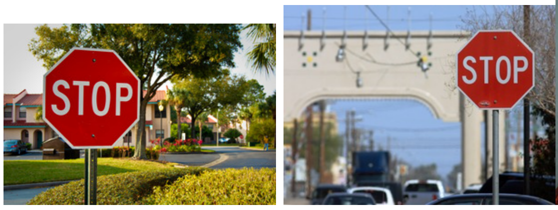
(g)

(h)

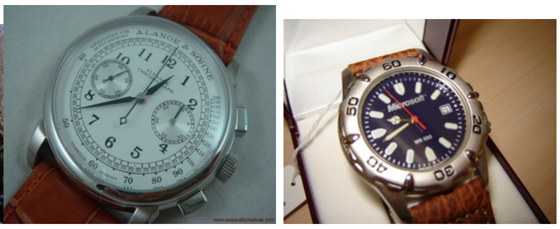
(i)



(j)



(k)



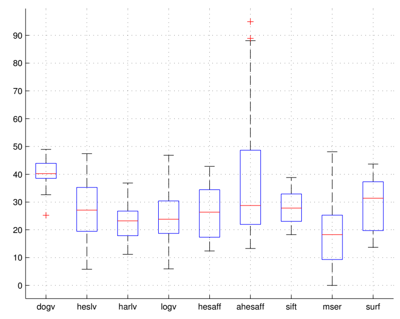
(l)

Figure A2.1. Example image pairs from (g) motorbikes, (h) mugs, (i) phones, (j) balloons, (k) stop signs and (l) watches.

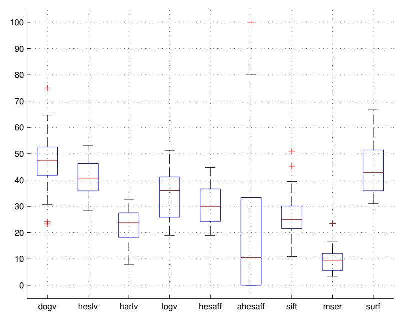
Appendix 3. Detector test results for individual classes

Abbreviations used in the figures		
dogv = dog-vireo	logv = log-vireo	sift = sift
heslv = heslap-vireo	hesaff = hesaff	mser = mser
harlv = harlap-vireo	ahesaff = hesaff-alt	surf = surf

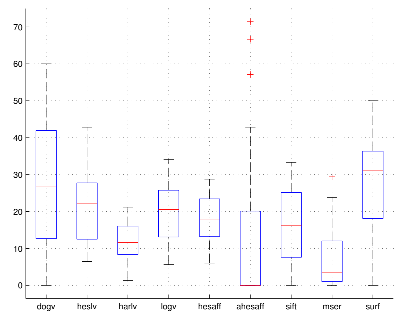
3.1 Repeatability rates for individual classes



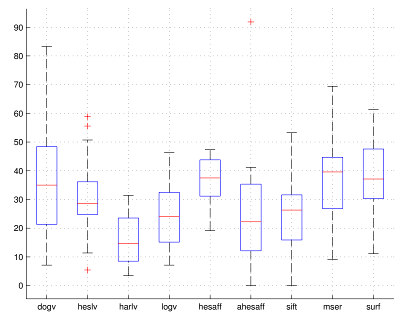
(a)



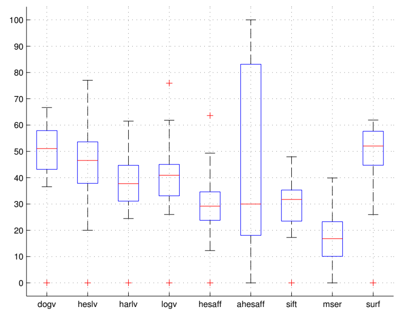
(b)



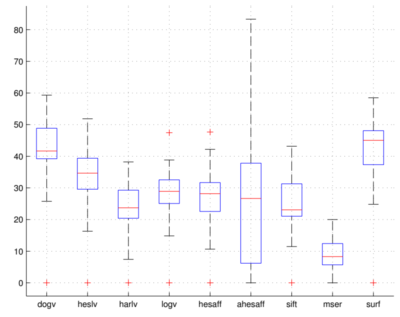
(c)



(d)



(e)

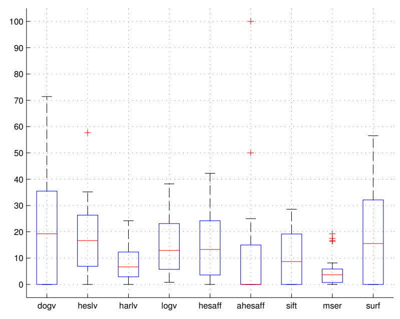


(f)

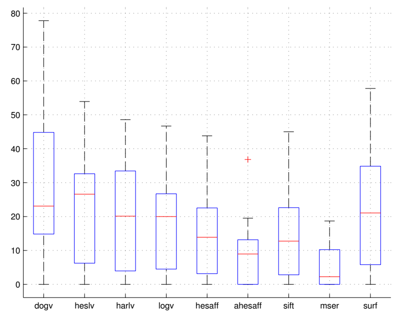
Figure A3.1. Repeatability rates for classes (a) faces, (b) motorbikes, (c) airplanes, (d) car sides, (e) dollar bills and (f) euphoniums.

(continues)

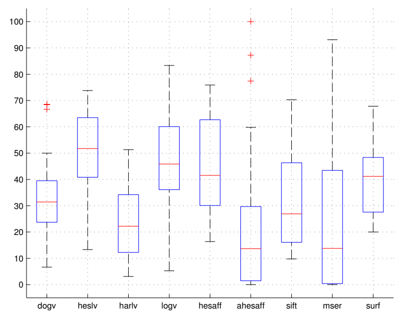
Appendix 3. (continued)



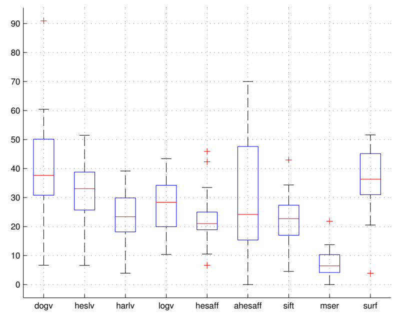
(g)



(h)



(i)



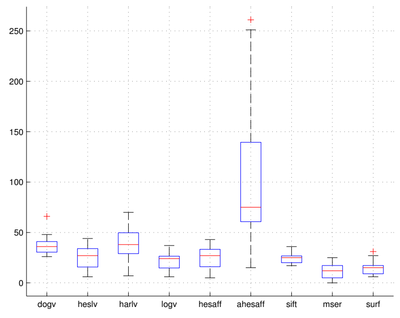
(j)

Figure A3.1. Repeatability rates for classes (g) revolvers, (h) starfish, (i) stop signs and (j) watches.

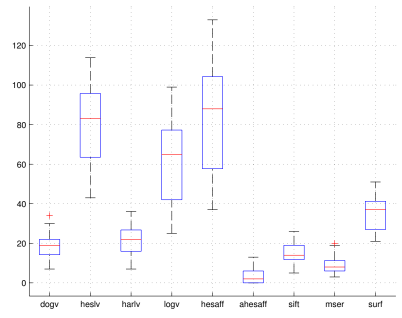
(continues)

Appendix 3. (continued)

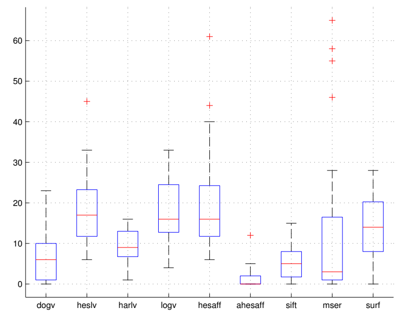
3.2 Number of correspondences for individual classes



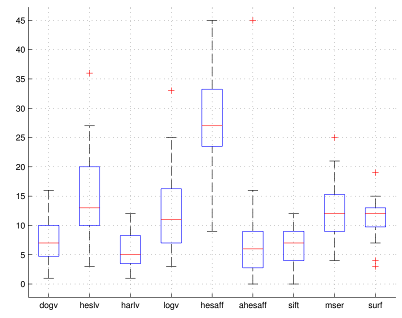
(a)



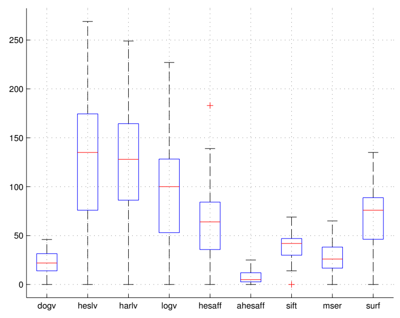
(b)



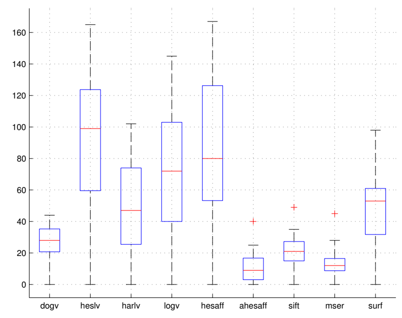
(c)



(d)



(e)

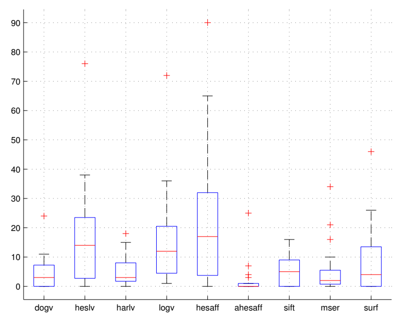


(f)

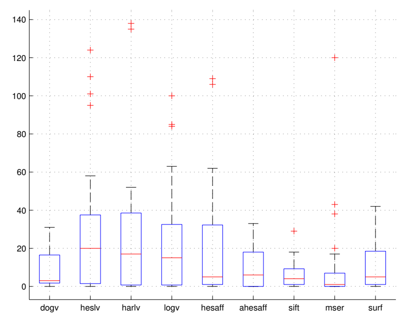
Figure A3.2. Number of correspondences for classes (a) faces, (b) motorbikes, (c) airplanes, (d) car sides, (e) dollar bills and (f) euphoniums.

(continues)

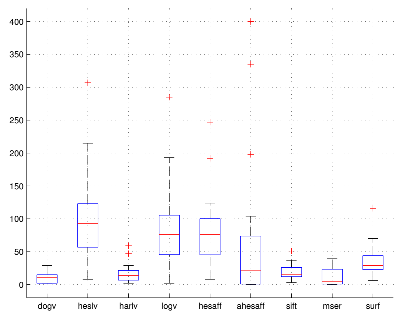
Appendix 3. (continued)



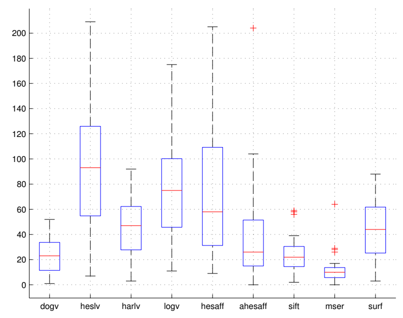
(g)



(h)



(i)



(j)

Figure A3.2. Number of correspondences for classes (g) revolvers, (h) starfish, (i) stop signs and (j) watches.