New Hierarchical Approaches in Real-Time Robust Image Feature Detection and Matching

M. Langer and K.-D. Kuhnert
University of Siegen
Germany

1. Introduction

The detection of robust image features of high distinctiveness forms a ubiquitous problem in digital image processing. Robust image features are the first step and the key to reliably detect patterns or objects in images, which subsequently leads to object classification and semantic interpretation of the objects in a given image.
The ideal situation for object recognition and classification is to find unambiguous traits of patterns or objects that are to be processed. For anyone delving into this matter, the subject presents itself as a challenging task. Besides the often not obvious object traits, there are a lot of other issues that have to be taken into account. For instance, lighting conditions can vary as well as the objects' scales and rotations; image noise or partial occlusion also is likely to occur. Unless one conducts experiments under laboratory conditions, where some of these problems might be ruled out, all these issues have to be addressed properly. So the challenging task is to find image features that are distinct and robust under the varying conditions just stated before.
An obvious method for detecting objects is to examine their shape and, as an abstraction of it, their *contours*. Contour matching usually works well, if the distinct object classes have strong variations in their shape (like the shape of any automobile is quite different compared to the shape of a human). The contour can be represented by a classic chain-code (or its many derivates). For the matching process both contour representations undergo a transformation to achieve scale and rotation invariance and are then compared to each other. The comparison can either be done directly on the transformed image coordinates or on measures deduced from the earlier representation (i.e. the moments, distances, polygonal, or Fourier descriptors yielded from the contour points).
Besides those methods, one popular approach is to detect certain local *interest points* at distinctive locations in the image, such as corners, blobs, or T-junctions. The interest point detector is supposed to be repeatable under varying viewing conditions (e.g. different lighting or different viewing angles). Having found distinctive interest points in an image, the interest points are examined more closely regarding their neighbourhood. Taking the neighboring pixels into account forms an *image feature* or *image descriptor*, which has to fulfill certain criteria regarding distinctiveness, robustness against noise, detection errors, and image deformations. These image features can then be matched to features computed from other images. Matching features indicate a high likeliness of correspondence between the two images. Hence patterns or objects in one image can be detected in other images

Source: Computer Vision, Book edited by: Xiong Zhihui,
ISBN 978-953-7619-21-3, pp. 538, November 2008, I-Tech, Vienna, Austria

employing this method. An advantage of this method is that detection of interest points takes place at one of the first stages of the image processing pipeline: in *scale-space*. All the interest point detectors and image descriptors discussed in section 3 will exploit the scale-space representation of an image. Hence a short review of scale-space computation is provided in the following section.

2. Structural information and scale-space representation of an image

In this section a short review about scale-space construction on digital images is given. One of the most thorough and comprehensive researches on the topic of scale-space representation was done in (Lindeberg, 1994). In his work, Lindeberg first states the principle problem about detecting the characteristic structure of objects in images regarding the scale level the image was taken. A good example is the structure of a tree that varies over different scaling levels. At a very coarse level, the observer may only recognize a green blob-like structure of the treetop, whereas at finer scales, branches and leaves may be noticeable. As the scale-space is a continuous and infinite space, this can go as far as the tree may degenerate to a single unnoticeable green spot or to a level where leaf-fibres can be examined. Which scale intervals are best suited totally depends on the image processing task at hand. A method for automatic scale selection for characteristic image features in this regard is presented in (Lindeberg, 1998).
Scale-space representations span a broad range of applications in image processing (i.e. feature detection, feature classification, shape computation, or even stereo matching). In this work, though, we are mainly interested in exploiting scale-space for object recognition. It will be shown that scale-space representations of images allow for effective extraction of distinctive image features, which can be matched with features from other images in the context of object recognition. To compute these distinctive features one has to transform the image into scale-space first, so that the interest point detectors and image descriptors introduced in section 3 can build upon this representation. The *multi-scale* representation of a digital image is well-known and widely used in image processing in forms of quad-trees, image pyramids, or (the relatively more recent) wavelets. The scale-space is just a special type of a multi-scale transformation.

2.1 Scale-space construction of continuous signals

The scale-space of a digital image is directly related to the scale-space in signal processing, whereas the scale-space representation of a one-dimensional signal is defined by *embedding* this signal into a one-parameter family of derived signals computed by convolution with a one-parameter *Gaussian kernel* with increasing width (Witkin, 1983). In (Lindeberg, 1994) a proof is provided that the Gaussian kernel is a *unique* choice for scale-space processing. The Gaussian kernel alone fulfills all necessary requirements for scale-space computation like basical linearity, spatial shift and scale invariance, and preserving the constraint that no new structures (maxima or minima in the image) will be created. It is also shown that the scale-space abides by the laws of a mathematical semi-group. The formal mathematical notion for constructing the scale space is as follows: Let $f : \mathbb{R} \rightarrow \mathbb{R}$ stand for the original signal. Then, the scale space representation $L : \mathbb{R}^N \times \mathbb{R}_+ \rightarrow \mathbb{R}$ is defined by $L(\cdot;0) = f$ and

$$L(\cdot;t) = g(\cdot;t) * f \tag{1}$$

where $t \in \mathbb{R}_+$ is the scale parameter, and $g : \mathbb{R}^N \times \mathbb{R}_+\backslash\{0\} \rightarrow \mathbb{R}$ is the Gaussian kernel. The kernel itself is defined by

$$g(x;t) = \frac{1}{(2\pi t)^{N/2}} e^{-x^T x(2t)} = \frac{1}{(2\pi t)^{N/2}} e^{-\sum_{i=1}^{N} x_i^2/(2t)} \qquad (x \in \mathbb{R}^N, x_i \in \mathbb{R}). \quad (2)$$

$\sigma = \sqrt{t}$ is the standard deviation of kernel g. It is a natural measure of spatial scale in the smoothed signal at scale t.
Another equivalent way to construct the scale-space L is by solving the differential heat diffusion equation

$$\partial_t L = \frac{1}{2}\nabla^2 L = \frac{1}{2}\sum_{i=1}^{N} \partial_{x_i^2} L \quad (3)$$

with initial condition $L(\cdot;0) = f$, which is the well-known physical formula describing how a heat distribution L evolves over time t in a homogeneous medium with uniform conductivity, given the initial heat distribution stated above. From Eq. (1)-(3) multi-scale spatial derivates can be defined by

$$L_{x^n}(\cdot;t) = \partial_{x^n} L(\cdot;t) = g_{x^n}(\cdot;t) * f, \quad (4)$$

where g_{x^n} denotes a derivative of some order n.
The main goal of the scale-space construction is that fine scale information (i.e. amplitude peaks or noise at high frequencies) shall vanish with increasing scale parameter t. Witkin (Witkin, 1983) noticed that new local extrema cannot be created, if the scale-space is contructed in the above described manner. Because differentiation and convolution are commutative,

$$\partial_{x^n} L(\cdot;t) = \partial_{x^n}(g(\cdot;t) * f) = g(\cdot;t) * \partial_{x^n} f, \quad (5)$$

this non-creation property holds for any nth degree spatial derivative. This feature enables detection of significant image structures over scales and is exploited by the algorithms described in section 3.
An important statement in (Koenderink, 1984) is that without a priori knowledge of specific image structures, the image has to be processed at *all* scales *simultaneously*. In this regard the images forming the scale-space are *strongly coupled* and not just a set of unrelated derived images.

2.2 Scale-space construction of discrete signals

Because digital image processing deals with discrete signals, the equations presented above need to be adapted. The interesting question here is how this adjustment needs to be performed so that the scale-space properties are preserved. It turns out that there is only one way to construct a scale-space for discrete signals. For an in-depth look into the derivation of the following formula see (Lindeberg, 1994). Given a signal $f : \mathbb{Z} \to \mathbb{R}$ the scale-space representation $L : \mathbb{Z} \times \mathbb{R}_+ \to \mathbb{R}$ is given by

$$L(x;t) = \sum_{n=-\infty}^{\infty} T(n;t) f(x-n) \quad (6)$$

where $T : \mathbb{Z} \times \mathbb{R}_+ \to \mathbb{R}$ is a kernel named the “discrete analogue of the Gaussian kernel”, which is defined in terms of a modified Bessel function given by $T(n;t) = e^{-\alpha t} I_n(\alpha t)$.

It can be shown that the discrete scale-space family $L : \mathbb{Z}^N \times \mathbb{R}_+ \to \mathbb{R}$ of a discrete signal $f : \mathbb{Z}^N \to \mathbb{R}$ must satisfy

$$\partial_t L = \alpha_1 \nabla_3^2 L \tag{7}$$

$$\partial_t L = \alpha_1 \nabla_5^2 L + \alpha_2 \nabla_{\times^2}^2 L \tag{8}$$

in one and two dimensions for constants $\alpha_1 \geq 0$ and $\alpha_2 \geq 0$. ∇_5^2 and $\nabla_{\times^2}^2$ denote two discrete approximations of the Laplace operator. They are defined by

$$(\nabla_5^2 f)_{0,0} = f_{-1,0} + f_{+1,0} + f_{0,-1} + f_{0,+1} - 4f_{0,0}\,,$$

$$(\nabla_{\times^2}^2 f)_{0,0} = \frac{1}{2}\left(f_{-1,-1} + f_{-1,+1} + f_{+1,-1} + f_{+1,+1} - 4f_{0,0}\right).$$

The function subscripts are the indices at which position in the image an intensity value shall be taken. With $\alpha_2 = 0$, the two-dimensional scale-space representation of a digital image is given by convolution with a one-dimensional Gaussian kernel along each direction. What remains to say is that the spatial derivatives of the Gaussian kernel can be approximated by *differences of Gaussian* (DoG) kernels at different spatial locations.

2.3 Image features from scale-space representations

To exploit scale-space for image feature detection, one has to deal with differential geometry. Some methods are required for further processing the output of the Gaussian derivative operators to gain meaningful and distinct image features. It is mandatory to base the analysis on image descriptors that do not depend on the actual coordinate system of the spatial and intensity domain, because a single partial derivative contains no useful geometric information. So it is required that the scale-space representation shall be *invariant* with respect to *translation*, *rotation*, and *scale changes*. Unfortunately complete affine invariance (i.e. non-uniform rescaling) is harder to achieve. This issue is also addressed in (Lindeberg, 1998).

Scale-space representation of an image is especially well-suited to detect sub-pixel *edges*, *junctions* and *blob-like* structures. For this, it is helpful to define a local orthonormal coordinate system for any point of interest in the image. A useful definition would be that for any Point P_0, normal *(x,y)* image coordinates are translated into a *(u,v)* coordinates with the *v*-axis aligned along the gradient direction and the *u*-axis perpendicular to it (which also aligns with the tangent orientation). This leads to $e_u = (\sin\alpha, -\cos\alpha)^T$ and $e_v = (\cos\alpha, \sin\alpha)^T$ for example, where

$$e_v|_{P_0} = \begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} = \frac{1}{\sqrt{L_x^2 + L_y^2}} \begin{pmatrix} L_x \\ L_y \end{pmatrix}. \tag{9}$$

In terms of Cartesian coordinates, the local directional operators can be expressed by

$$\begin{pmatrix} \partial_u \\ \partial_v \end{pmatrix} = \begin{pmatrix} \sin\alpha & -\cos\alpha \\ \cos\alpha & \sin\alpha \end{pmatrix} \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}. \tag{10}$$

Note that $L_u = 0$, because of e_u being parallel to the tangent at point P_0.

With these definitions *edge detection* in scale-space can be computed by a non maximum suppression in the gradient direction. The following conditions need to be fulfilled:

$$\begin{cases} L_{vv} = 0 \\ L_{vvv} < 0 \end{cases}. \tag{11}$$

With Eq.(9) and (10) it is possible to convert these statements back to Cartesian coordinates. *Junction detection* is expressed in terms of curvature of level curves in gray level images. In derivative terms the curvature can be expressed by

$$\kappa = \frac{L_{uu}}{L_v}. \tag{12}$$

The curvature is usually multiplied by the gradient magnitude L_v raised to some power *k*. This provides stronger responses at edges. In (Brunnström, 1992) *k=3* is chosen leading to

$$|\tilde{\kappa}| = \left|L_v^2 L_{uu}\right| = \left|L_y^2 L_{xx} - 2L_x L_y L_{xy} + L_x^2 L_{yy}\right|. \tag{13}$$

A point P_0 has to fulfill the following conditions to be considered a junction-point:

$$\begin{cases} \partial_u(\tilde{\kappa}) = 0, \\ \partial_v(\tilde{\kappa}) = 0, \\ \mathcal{H}(\tilde{\kappa}) = \tilde{\kappa}_{\mathcal{H}} = \tilde{\kappa}_{uu}\tilde{\kappa}_{vv} - \tilde{\kappa}_{uv}^2 > 0, \\ \operatorname{sign}(\tilde{\kappa})\tilde{\kappa}_{uu} < 0. \end{cases} \tag{14}$$

Blob detection can be performed by calculating the zero-crossings of the Laplacian, where

$$\nabla^2 L = L_{uu} + L_{vv} = L_{xx} + L_{yy} = 0. \tag{15}$$

All the introduced feature detectors in section 3 first do a scale-space transformation and then employ Eq. (11)-(15) to detect interest points. At these points the local image descriptors are then computed and used for the later matching process.

3. Technology review of interest point detectors and image descriptors

Usually the computation of distinctive image features is split into two phases. First, interest points are computed at locations that are considered to be promising for the later descriptor calculation. A reliable (and in this regard reproducible) interest point detector is crucial for the feature determination. Then, after the locations of the interest points have been determined, for each such point a local image descriptor is computed by processing the neighbouring intensity values of the current interest point's location. This yields a set of image features that are supposed to be distinctive to the objects in the image. These features can then be used in object recognition or object tracking by matching features yielded from one image to those from another.

3.1 Interest point detectors

Classic interest point detectors used simple attributes like edges or corners. Among these, the probably most widely used, is the Harris corner detector (Harris, 1988). It is based on eigenvalues of the second-moment (also called auto-correlation) matrix. Unfortunately, this detector lacks scale-invariance.

Lindeberg tried to extend the Harris corner detector experimenting with the determinant of the Hessian matrix as well as the Laplacian to detect blob-like shapes (Lindeberg, 1998). He also examined methods for automatic scale selection for image feature computation.

(Mikolajczyk & Schmid, 2004) refined this method exploiting a combination of the Harris detector to determine the location, and the Laplace matrix for scale selection of an image feature. It is basically a combination of the Harris corner detector with the automatic scale selection methods proposed by Lindeberg. This yields the desired scale-invariance. Their image descriptors proved to be highly distinctive, scale invariant, rotation invariant, and highly tolerant against illumination effects.

(Lowe, 2004) mainly focused on speeding up this computational costly processing by replacing the Laplacian of Gaussian (LoG) by a Difference of Gaussian filter (DoG). In 2004 he presented his method called SIFT (scale invariant feature transform) to the public. SIFT features embody the same tolerance to (uniform) changes in scale, rotation, and illumination effects. This also qualifies SIFT as an ideal candidate for pattern and object recognition.

The main difference between SIFT and the feature detector described in (Mikolajczyk & Schmid, 2004) is that the former interest point detector aims at blob-like structures and the latter at corners and highly textured points in images. Thus it can be concluded that the two interest point detectors generate complementary feature.

One basic problem with these sophisticated interest point detectors is that they are computationally complex and hence cannot match hard real-time constraints (i.e. analyzing a video stream of a constant 25fps online). This problem was addressed in (Bay *et al.*, 2006). The group showed that SIFT could be speeded up further almost without losing any of its matching quality. Their work is based on certain approximations (and simplifications) of Lowe's work and was coined "*speeded up robust features*" (SURF). For instance, the research group proved that the approximation of the LoG by a DoG filter can be pushed even further to a *difference of means* (DoM) filter. This filter can be implemented very efficiently exploiting integral images, thus achieving constant runtime behavior. Also the components of the resulting feature vector are cut in half, which provides faster matching speed between different feature vectors.

Usually some interpolation steps are applied over scale-space to accurately locate the interest points before the image descriptor is computed. In (Mikolajczyk & Schmid, 2004) an iterative approach is taken based on displacement calculations using Eq. (16) by starting at the initially detected point for affine normalized windows around this point. In (Lowe, 2004) a 3D quadratic function is fitted to local sample points to interpolate the location of the maximum. This is employed by using the Taylor expansion introduced in (Brown & Lowe, 2002).

A common drawback to all these interest point detectors is their strong responses at edges or contours. This can be mitigated to a certain degree by selecting the scale at which the trace *and* the determinant of the Hessian matrix assume a local extremum.

3.2 Image descriptors

Image descriptors are computed from the location around the previously detected interest points.

(Mikolajczyk & Schmid, 2004) use Gaussian derivates computed in the local neighborhood of each interest point. They convolve these derivatives on small image patches normalized with a matrix U of the form

$$U = \prod_{k} \left(\mu^{-\frac{1}{2}}\right)^{(k)} U^{(0)}. \tag{16}$$

with μ denoting the (scale-space adapted) second moment matrix of the Harris corner detector and k denoting the step width of this iterative algorithm. $U^{(0)}$ is the initial concatenation of square roots of the second moment matrices. To gain invariance to rotation, the derivatives are aligned at the direction of the gradient. The gradient orientation at each interest point is averaged with the gradient orientations in the neighborhood and goes into the descriptor.

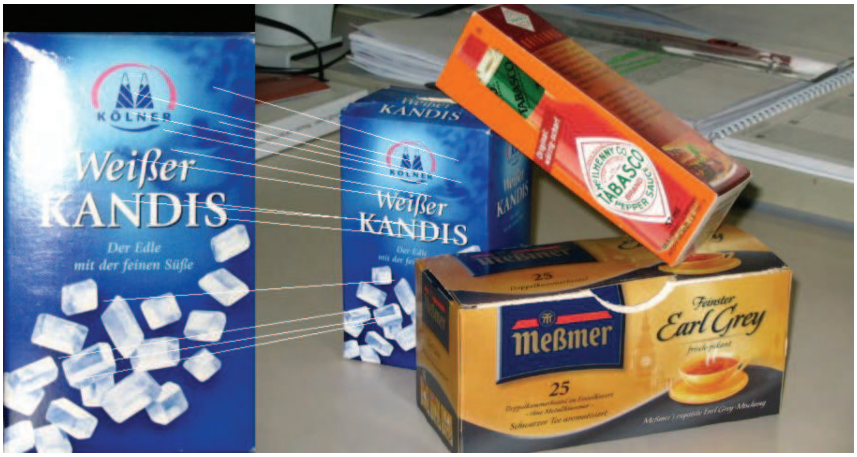

Fig. 1. SIFT feature matches applied to a self recorded scene. The blue sugar box on the left side of the image was found in the right complex scene. The SIFT matching software yielded 16 matches (shown as white lines). Note that only 3 matches would have sufficed to correctly identify an object. Also note the partial occlusion and difficult lighting conditions under which the images were taken. The digital camera used to record both images was a Canon Powershot A75 with 3.2 megapixels. The camera was set to automatic mode. The images also underwent a lossy JPEG compression.

Lowe determined his descriptor components by computing the gradient magnitude and orientation for each sample point neighboring the interest point. These values are weighted by a Gaussian window. The samples are then accumulated into orientation histograms, which summarize the contents of 4x4 subregions around the interest point. Over each such subregion the 8 predominant gradient orientations are computed. These values make for the image descriptor. Hence the descriptor consists of 128 components. A visual example of the image feature matching with our own implementation of the SIFT operator is shown in Fig. 1.
The SURF descriptor is determined by the following steps. First, a square region is constructed around the interest point and aligned along the dominant orientation obtained by haar wavelet filter responses at the interest point. The side length of the wavelet is four times the scale level at which the point was detected. The dominant orientation is estimated

by calculating the sum of all responses within a sliding window covering an angle of $\frac{\pi}{3}$. Then, the horizontal and vertical responses over these windows are summed and contribute to the components of the vector of the estimated orientation. The size of this sliding window needs to be determined experimentally (see Bay *et al.*, 2006). The square region size around the interest point is of size 20s (s denoting the scale parameter). The region is split into 4x4 subregions like in (Lowe, 2004). For each such subregion a few simple features (again employing the directional Haar wavelet responses) at 5x5 regularly spaced sample points are computed. Finally, the responses are accumulated over the regions and provide a first set of entries to the image descriptor. Also the sum of the absolute response values is calculated. This yields a four-dimensional vector for each subregion of the form $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$, with d_x denoting the Haar wavelet response in x-direction and d_y respectively. Such vectors form a single descriptor vector for all four subregions. Hence the resulting descriptor vector is of size 64 – half the size of a SIFT-descriptor.
There are yet some derivations the introduced image descriptors, i.e. (PCA-SIFT, U-SURF, GLOH etc.) that are not discussed here. These derivates mainly vary in the way how the final descriptor computation is done aiming at lesser component numbers than the original version to speed up object matching.

4. Performance evaluation of local image features

This section is contributed to a short overview of the performance of local image features regarding their *object recognition ratio* and their *computational speed* when implemented on modern computers. Also the issue of exploiting image features for *object classification* is addressed at the end of this section.
In (Leibe & Schiele, 2003) a systematic comparison between contour based, shape based, and appearance based approaches regarding their effectiveness for object recognition is performed. For their evaluation they constructed an own database of small objects, coined the *ETH-80* database. It consists of 80 objects from 8 self chosen categories (i.e. fruit and vegetables, animals, human-made small objects, and human-made big objects). For each category 10 objects are provided that span in-class variations while still clearly belonging to that category. The test mode is of the form leave-one-object-out-crossvalidation, which means that the object recognition system is trained with 79 objects (randomly taken out of the 8 categories) and tested against the one remaining object. The measurements were performed over object sets taken of each category and the results were averaged.
The algorithms used for testing were

- a simple *colour histogram* driven approach,
- *texture based* methods over scale-space (which directly corresponds to the image feature detectors from section 3.1-3.2); these are split into a *rotation variant* method (first order derivatives in *x* and *y* direction) and a *rotation invariant* method (gradient magnitude and Laplacian over 3 scales – also see (Love, 2004)),
- *global shape*: PCA-based methods either based on *one single global eigenspace* for all categories or *separate eigenspaces* for each category,
- and *local shape* using the contours of each object represented by a discrete set of sample points; these points are later matched with points from other images with a *dynamic programming* and a *one-to-one point matching* (using a greedy strategy) approach.

The results found in (Leibe & Schiele, 2003) show that contour-based (local shape) methods perform best, with an average object recognition ratio of 86.4%. Second place were global-shape based PCA variation with 83.4% and 82.9% respectively, but with the texture histograms only slightly behind with 82.2%. As expected, colour histograms performed worst with just 64.9%. Yet, nothing is stated about the different runtime behaviour of the different methods, because this paper deals with quantitative evaluation of the recognition ratio. It can reasonably be assumed that the global and local contour matching algorithms, besides showing the best recognition ratio are also the most costly ones to compute.
Leibe & Schiele also examined combinations for the above described methods. Cascading object recognition systems were constructed using different permutations of these algorithms. They showed that certain combinations could raise the overall object recognition ratio up to 93%. In section 5.1 we show our own approach for a cascading object recognition system aimed to fulfil certain real-time constraints, yet preserving high recognition ratios.
(Mikolajczyk *et al.*, 2005a) evaluated interest region descriptors among themselves and how they perform in different situations; they concentrated especially on all kinds of affine transformations, blurring, and JPEG compression artefacts. They concerned with all the image feature detectors described in section 3 and their common variations and also proposed their own extension to the SIFT operator. In their results section they conclude that the *gradient orientation and location histogram* (GLOH) – a variation of the original SIFT operator – performs best regarding object recognition ratio, closely followed by SIFT. We also employ SIFT features for our cascading object recognition system (see section 5.1).
Another highly noteworthy contribution is presented in (Mikolajczyk *et al.*, 2005b), where local image features are examined regarding their applicability on *object classification*. The interesting question here is, whether the image features, which perform well in scenarios like pattern matching and object recognition, contain useful information that can be applied as features for object classification. They pursue a cluster-driven approach of image features for classification. They computed the distribution of features in the cluster and defined a similarity measure between two clusters by

$$\frac{1}{NM}\sum_{m}^{M}\sum_{n}^{N}(f_{km}-f_{lm})^2=\sigma_k^2+\sigma_l^2+(\mu_k-\mu_l)^2\leq v \tag{17}$$

with N and M denoting the numbers of features in clusters k and l; μ_k and μ_l represent the cluster centres; σ_k^2 and σ_l^2 denote the variances, and v an experimentally determined threshold value.
Mikolajczyk *et al.*, evaluated image feature detectors employing Hessian-Laplace and Salient region, Harris-Laplace, SIFT, PCA-SIFT, and GLOH detectors. The challenging object classification task was to detect pedestrians crossing a street in an urban scenario. Again, the GLOH descriptors exploiting regions around interest points found by Hessian-Laplace obtained the best results; salient region detectors also performed well. Hence, it can be concluded that scale-space image feature detectors are also applicable to object classification beyond their originally intended domain (pattern and object matching).

5. Extending local image features for real-time object recognition

All the previously described image features from section 3 show excellent robustness on object recognition under real-world conditions. Regarding object recognition by matching features of different images, the image features prove to be

- scale-invariant,
- rotational invariant,
- *partially* affine invariant (non-uniform scale changes are problematic),
- highly tolerant against changes in illumination, and
- insensitive to noise (to a certain reasonable degree).

Yet, one important issue has not been fully addressed: the real-time processing of image features or feature matching respectively. In many scenarios, online computation of image features for real-time object matching is desirable. In this regard it is common to imply that a constant video stream needs to be analyzed in real-time, which in this case would mean that the image feature computation software had to cope with 25fps in camera-like image resolutions. The high robustness of the scale-invariant image features comes at the prize of high computation-time, which renders them applicable only partially to these scenarios.
Constructing the scale-space and finding reliable interest points is one of the bottlenecks in this case; another is the component size of the image descriptors, which have to be used for the later matching process. Many extensions to the image features have been applied to address this problem and especially the latest of these, the SURF operator, aims at better runtime speed. The idea in (Bay *et al.*, 2006) is to simplify the costly difference of Gaussian (DoG) filter, which is used during scale-space computation, to a *difference of means* (DoM) filter. The DoM is just a simple box filter. Bay *et al.* showed that this approximation of the DoG filter is permissible, because the DoG filters, which are supposed to be infinite, are actually cropped when they are applied to the image. Thus DoG filters are, when it comes to applying them in discrete digital image processing, an approximation of the theoretical DoG filters themselves. The DoM filter has the nice property that it can be computed very efficiently employing integral images. An integral image is defined by

$$I_\Sigma(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j). \tag{18}$$

After I_Σ is constructed, it takes just four additions to calculate the DoM filter of *any size* at any point. This is a major saving of computation time compared to the full application of a Gaussian filter at any point.
Bay *et al.* also reduce the length of the image descriptor to 64 components in contrast to the 128 components of the SIFT operator. This saves time when feature matching is applied, because during the matching process each image descriptor from one image has to be compared (usually by employing some common metric, i.e. the Euclidian distance) to each descriptor of the image that is to be matched.
These measures mitigate the problem of time consuming computations and in fact the SURF operator with all its optimizations only needs 33% the time of the SIFT operator to perform its tasks.

5.1 Cascading object recognition approach

Yet, we seek even better run-time performance than the optimized SURF operator. The SURF operator still cannot satisfy the hard real-time constraints stated in the last section. So we came up with a cascading approach for object recognition. The precondition to our

approach is that *a priori knowledge* of the objects to be recognized can be assumed, which is the case in many classic object recognition and classification scenarios. In particular you need to know with which objects the system is supposed to deal with to construct sensible training data sets for an initial teaching of the system.
The main idea of our contribution is to employ a *two-stage cascading approach*: a fast decision-tree based pre-classifier, which sorts out the objects to recognize with a recognition ratio of 66%-75% accuracy, and a later performed image feature matching exploiting SIFT and SURF operators. The recognized objects from the first stage need not be processed by the costly operators anymore. The time savings go directly into equation

$$t_o = t_p + (1 - r)t_s + t_d \tag{19}$$

with t_0 denoting the overall system processing time, t_p the time for pre-computation (system initialization etc.), t_s the time for the SIFT/SURF matching process, and t_d the dead-time of the system (consumed e.g. by memory management and other administrative tasks). r is the ratio of correctly classified objects from the earlier pre-processing stage. Considering the fact that $t_p << t_s$, applying the pre-computation yields better overall timing.
We use the Coil-100 image database[1] as a testbed for our algorithms. This library consists of 100 different, small, everyday use objects, which have about the same size. Each object is rotated around the Y-axis 72 times at a 5° angle each step, thus covering the full 360 degree, which yields 7,200 object view images in total. Each image has a size of 128x128 pixels. The methods and results presented in the following sections all refer to this database.

5.2 Decision tree construction

Decision trees are a classic tool rooting in artificial intelligence and are mainly used for data mining purposes. Decision trees in our work are exploited to classify objects. For an in-depth look into the structure of general decision trees and methods to generate them the reader is referred to (Quinlan, 1983), who was the first to introduce decision trees in his work and (Russel, 2003), who provides a quick and practice oriented overview of decision trees.
We use a fast classification method in the first stage for our object recognition software exploiting *binary decision trees*. Decision trees, once trained, provide means to classify large amounts of objects (several thousands) in just a few milliseconds. The idea is to recursively divide the feature space into sets of similar sizes and feed these sets to nodes into the decision tree. To classify an object, the object features are matched with the class feature sets each node holds. Each node decides whether an object belongs to one (or none) of its two class-feature sets. The decision made by the node determines the next sub-tree, which has to deal with the classification. This process is repeated until a leaf-node is reached. The feature represented by this leaf node yields the class the object belongs to. Because the tree needs not necessarily be built up completely, whole class or feature sets can be represented by leaf nodes.
We pursue a simple colour cluster driven approach to train the decision tree based classificators. The training data sets are taken out of the Coil100 image database. Although

[1] http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php

yielding high performance, the training of the decision trees proved to be difficult regarding classification quality, which is at 66% of correctly classified objects on average. This is mainly due to the simple colour based features we use. To gain the features, we apply Alg. 1.

Algorithm 1: Generating a decision tree

```
Data: training sets $S$ of images
Result: generated decision tree $d$
begin
    $C \longleftarrow \emptyset$
    forall $I \in S$ (with $I \subset S$) do
        $G \longleftarrow \emptyset$
        repeat
            $A \longleftarrow P \longleftarrow \emptyset$
            forall $i \in I$ do
                $P \cup$ GaussPyr($i$)
                $avg_{RGB} \longleftarrow P(i)|_{max.\ pyramid\ level}$
                $A \cup avg_{RGB}$
            $G \cup$ CalcCOGs($A$)
        until $|S| = 2$
        $C \cup G$    (C holds all CoG subsets)
    $d \longleftarrow$BuildDecisionTree($C$)
    return $d$
end
```

Each node divides the center of gravity set (and so the associated class set) into half. This way we achieve a runtime complexity of $\mathcal{O}(M \log(N))$ with N denoting the number of classes and M the number of objects to classify.
The problem of low classification quality can be mitigated by moving to *decision forests* (see Zhao, 2005). Decision forests integrate more than one decision tree into their decision process. These trees are trained with varying datasets which may lead to different classification results. Among all the trees in the forest a quorum is made and the decision taken which should be correct most likely; usually this is done by proclaiming the largest subset of trees, which made the same decision, the winning subset using their decision as the classification result. Applying this method raises the correctly classified objects rate to 75%, but also increases the runtime complexity to $\mathcal{O}(kM \log(N))$ with k denoting the number of trees in the forest.
The rate of correctly classified objects in this stage may seem to be quite low compared to other more sophisticated classificators found in literature, but it is important to note that we employ the pre-stage system only to speed up the later main-stage object recognition process. Every correctly recognized object at the end of this phase means, that no costly image transformation, and no exhaustive search through an image feature database comparing all the descriptor vectors with each other needs to be performed. Hence a classification rate of 66% to 75% means an overall time reduction of the entire system at almost the same rate. As we will show next, this will get us closer to the desired real-time behavior of the object recognition system.

5.3 Performance evaluation

The two-stage image feature matching approach yields a very efficient object recognition method providing highly accurate results. The pre-processing of the algorithm drastically reduces the search-space at minimal computational costs of a few milliseconds.

A typical feature match of a Coil-DB image with the SIFT feature database performs at 75ms. Applying the pre-processing, which yields 75% of correctly classified objects, which is the average rate for well trained decision forests in our implementation, the processing time reduces to 21ms. This time reduction is correlated linearly to the rate of correctly classified objects of the pre-processing phase and follows Eq. (19).

feature distance	correct positive	false positive
40,000	49.30%	0.00%
60,000	65.38%	0.09%
80,000	85.66%	1.74%
100,000	95.10%	8.85%
120,000	97.55%	31.78%

Tab. 1. Results of the object recognition process employing SIFT image features.

Tab. 1 shows some results regarding the object recognition ratio applying SIFT. It clearly shows the high accuracy and reliability employing SIFT features. The matching parameter denotes the threshold that is taken for feature comparison. Features are compared to each other using the squared Euclidian distance over the 128 dimensional feature vectors. Only feature distances that are below this threshold are considered to be equal enough to yield a match. As it can be seen, at a certain value of the threshold (in this particular case 100,000) we yield a rate of correctly classified objects of 95%. Increasing the threshold value any further increases the false positive rate of the matched features drastically (from 9% to 32%) yet improving the correct positive rate at only 2%. We averaged our results over several objects that were taken out of the Coil-100 database randomly. The images we used were to be found in a subset of the database that consisted of all objects and object views from an angle of 60°. Angles between 60° and 90° cannot be reliably matched with the SIFT operator anymore. Values beyond 90° are impossible to use with SIFT, because the object views now show their backside of the object whereas the reference object view shows its front side. For SIFT this is equivalent to a total occlusion of the object. The optimal feature distance for a concrete image set is determined experimentally and changes according to different environments.

Exchanging the SIFT with the SURF operator the values from Tab. 1 stay almost the same (with only slight deviations), except for the even better run-time performance of the system. We could verify that the SURF operator is indeed almost 3 times faster than SIFT as stated in (Bay *et al.*, 2006). The overall computation time of 21ms reduces to 10ms, if SIFT is replaced by SURF in the main-processing stage.

We also took our own image sets with standard CMOS cameras at standard sizes of 320x240 pixels. The object recognition system is capable of processing at a rate of 11fps (SIFT) and 16fps (SURF). This gets us closer to the desired real-time behaviour of 25fps, yet exploiting all the benefits of the SIFT and SURF operators.

6. Conclusion and future work

In this chapter we gave an overview about current object recognition methods exploiting scale-space transformation and showed an extension to the very reliable and accurate methods SIFT and SURF. We focused on speeding up these two operators even further targeting at real-time behaviour (at least 25fps for 320x200 pixel sized images). We used a pre-processing step exploiting decision trees to sort out as much data as possible in an early stage, trying to employ the costly SIFT and SURF matching process only at falsely recognized objects. We showed that this method is capable of further reducing overall computation time by the formulas given in section 5.1.
We intend to use this hierarchical approach for use in an *analysis by synthesis* system (Todt, 2008). This system is supposed to reliable detect real-world objects from synthetic three dimensional scene models, which are generated by a photo-realistic 3D lumigraph renderer for the synthesis part. The generation of this renderer uses the recently appeared PMD cameras (see Stommel & Kuhnert, 2006; Kuhnert *et al.*, 2007). Because this is an iterative computational process approximating, the best synthesis parameters for a given real-world object, one faces hard real-time constraints. We are confident, that our approach presented in this paper embodies the potency to fulfil these constraints.
We also plan on implementing this object recognition system on our mobile outdoor robot AMOR (see Kuhnert & Seemann, 2007; Seemann & Kuhnert, 2007) for passive object tracking purposes. It is supposed to support the active laser scanner sensors in tracking a vehicle that drives in front of AMOR. This is done by analyzing a constant video stream taken from a camera on the front side of the robot. Hard real-time constraints apply here as well.
There are still many optimizations remaining, e.g. software parallelization exploiting modern multi-core processors has yet to be implemented. The filter operations in particular are excellent candidates for this approach, because there are no data dependencies. This also leads to purely GPU based image filter algorithms (see Staudt, 2008). Modern graphic cards have shown high potential in processing large streams of independent data. We hope in implementing these optimizations, the costs for image feature computation can be reduced even further and that we get closer to the above stated real-time behaviour.

7. References

Bay, H.; Tuytelaars, T. & Gool, L. V. (2006). Surf: Speeded up robust features. *9th European Conference on Computer Vision*, 2006

Brown, M. & Lowe, D. G. (2002). Invariant features from interest point groups. In *British Machine Vision Conference*, Cardiff, Wales, pp. 656-665, 2002

Brunnström, K.; Lindeberg, T. & Eklundh, J.-O. (1992). Active detection and classification of junctions by foveation with a head-eye system guided by the scale-space primal sketch. *In Proc. 2nd European Conf. on Computer Vision* (G. Sandini, ed.), vol. 588 of *Lecture Notes in Computer Science*, pp. 701-709, Springer-Verlag, 1992

Harris, C. & Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the Alvey Vision Conference*, pp. 147 – 151, 1988

Koenderink, J. J. (1984). The structure of images. In *Biological Cybernetics*, 50, pp. 363-370, 1984

K.-D. Kuhnert, M. Langer, M. Stommel, & A. Kolb. (2007). Dynamic 3D-Vision, In *volume 4 of Vision Systems, Applications*, chapter 18, pages 311–334. I-Tech Education and Publishing, Vienna, Austria, June 2007. ISBN 978-3-902613-01-1.

Kuhnert, K.-D. & Seemann, W. (2007). Planning of minimal-time trajectories for high speed autonomous vehicles. In *The 2007 IEEE Intelligent Vehicle Symposium* (IV'07), Istanbul, Turkey, June 13-15, 2007.

Lindeberg, T. (1994). Scale-space theory. A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, 21(2), pp. 224-270, 1994

Lindeberg, T. (1998). Feature detection with automatic scale selection. *Int. Journal of Computer Vision*, 30, 2, pp. 79—116, 1998

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, 20, pp. 91–110, 2004

Mikolajczyk, K. & Schmid, C. (2001). Indexing based on scale invariant interest points. *ICCV*, 1, pp. 525–531, 2001

Mikolajczyk, K. & Schmid, C. (2004). Scale & affine invariant interest point detectors. In *Int. Journal of Computer Vision*, 60(1), pp. 63-86, Kluwer Academic Publishers, 2004

Mikolajczyk, K. & Schmid, C. (2005a). A performance evaluation of local descriptors. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10), October, 2005

Mikolajczyk, K.; Leibe, B. & Schiele B. (2005b). Local features for object classification. In *Tenth IEEE International Conference on Computer Vision (ICCV 2005)*, 2, pp. 1792-1799, 17-21 Oct, 2005

Quinlan, J. (1983). *Machine Learning: an artificial intelligence approach*, pages 463–482. Morgan Kaufmann, 1983

Russel, S. & Norwig, P. (2003). *Artificial Intelligence - A Modern Approach*, Prentice Hall, pp. 653—664, 2003

Seemann, W. & Kuhnert, K.-D. (2007). Design and realization of the highly modular and robust autonomous mobile outdoor robot amor. In *The 13th IASTED International Conference on Robotics and Applications*, Würzburg, Germany, August 29-31, 2007.

Staudt, A.; Langer M. & Kuhnert, K.-D. (2008). Comparison of two real-time image processing system approaches. *Proceedings of the 10th IASTED International Conference on Computer Graphics and Imaging,* 2008

M. Stommel & K.-D. Kuhnert. (2006). Fusion of stereo-camera and PMD-camera data for real-time suited precise 3D environment reconstruction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4780–4785, October 9-15, 2006. Also presented at the Robotic 3D Environment Cognition Workshop at the *Spatial Cognition*, Bremen, Germany, September 24-28, 2006.

Todt, S.; Langer, M.; Rezk-Salama, C.; Kolb, A. & Kuhnert, K.D. (2008). Spherical light-field rendering in application for analysis by synthesis. *IJISTA N3/4*, 5, 2008

Witkin, A. P. (1983). Scale-space filtering. In *Proc. 8th Int. Joint Conf. on Art. Intell.*, Karlsruhe, Germany, pp. 1019-1022, Aug. 1983

Zhao, H. & Sinha, A. (2005). An efficient algorithm for generating generalized decision forests. In *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35, pp. 754–762, 2005