Robust Local Features and their Application in Self-Calibration and Object Recognition on Embedded Systems

Clemens Arth, Christian Leistner, Horst Bischof
Graz University of Technology
Institute for Computer Graphics and Vision
Inffeldgasse 16/2, 8010 Graz, Austria
{arth,leistner,bischof}@icg.tu-graz.ac.at

Abstract

In recent years many powerful Computer Vision algorithms have been invented, making automatic or semi-automatic solutions to many popular vision tasks, such as visual object recognition or camera calibration, possible. On the other hand embedded vision platforms and solutions such as smart cameras have successfully emerged, however, only offering limited computational and memory resources.

The first contribution of this paper is the investigation of a set of robust local feature detectors and descriptors for application on embedded systems. We briefly describe the methods involved, i.e. *the DoG (Difference of Gaussian) and MSER (Maximally Stable Extremal Regions) detector as well as the PCA-SIFT descriptor, and discuss their suitability for smart systems and their qualification for given tasks. The second contribution of this work is the experimental evaluation of these methods on two challenging tasks, namely fully embedded object recognition on a moderate size database and on the task of robust camera calibration. Our approach is fortified by encouraging results we present at length.*

1. Introduction

In the last few years, Computer Vision has become one of the most powerful tools in engineering. Industrial and commercial demands are further pushing the development of high-performance vision algorithms, bringing up efficient solutions to existing problems and also many new applications into everyones everyday life. This has led to the deployment of large networks of cameras and, in turn, a demand for local processing [4]. Therefore and due to their flexibility, scalability as well as passive operation special interest has been placed on smart camera systems, *e.g.* for industrial and surveillance applications.

Robust local features and descriptors have already been successfully applied to related tasks like, auto-calibration of cameras or object recognition. They are designed to be invariant to illumination changes, image scale and rotation. Therefore, these features may be robustly matched over a substantial range of affine distortion and change in 3D viewpoint. Clearly these properties come at high demands in terms of computational power and memory. Hence, until now in resource constrained embedded systems, local feature and descriptor based systems for calibration or recognition have been mostly avoided.

In general, most Computer Vision algorithms are designed for use on standard desktop computers, having to meet almost no constraints in terms of memory and computational power consumption. Additionally, they mostly do not fit the fixed-point and SIMD architectures of embedded systems (*e.g.* fixed point DSPs). Altogether, this has often made the full employment of state-of-the-art algorithms a tedious task on resource constrained embedded devices. As embedded systems are limited in enrolling their full potential without taking advantage of the state-of-the-art algorithms, we concentrate our work on connecting these two worlds in order to benefit from both approaches.

In this work we investigate a set of highly-efficient robust local features and descriptors used in a wide range of popular Computer Vision applications. We discuss the suitability of these methods for implementation and usage on embedded systems and experimentally show their efficiency on two applications, namely a fully embedded object recognition system and a camera calibration framework. Our encouraging results justify the usage of these algorithms for specific problems in the world of smart sensors.

Section 2 gives an overview about related work in the area of embedded systems. In section 3 we briefly discuss the criteria for choosing algorithms for given tasks. Furthermore, we shortly describe our region detection and description algorithms, together with the algorithms for descriptor matching and for epipolar geometry calculation. An exper-

imental evaluation of our implementations and examples of two applications, namely object recognition and camera calibration, is given in section 4. The remainder of the paper in section 5 contains some concluding remarks and an outlook on future work.

2. Related Work

In recent years smart cameras have attracted interest of many research groups with yielded applications in traffic monitoring [1, 5], home care [12], gesture recognition [21], to mention a few. Many applications demand for well calibrated systems while fast and easy deployment still has to be guaranteed. Thus, solving the challenges of self-localization and self-calibration is very important for smart camera networks.

Both tasks (see also section 3.6) have shown to be partially solvable using point correspondences and, thus, local features and descriptors. Clearly, the performance of algorithms based on point correspondences is highly dependent on the quality of the detection process and on the type of descriptors used. Although Mikolajczyk *et al*. [18, 19] have shown that *Difference of Gaussian* (DoG) keypoints in combination with *Scale Invariant Feature Transform* (SIFT) descriptors [14] have proven to be very effective in terms of detectability and repeatability, most embedded systems use simpler corner detectors [10] or use additional active LEDs [2, 13] to perform calibration. Yet, Cheng *et al*. [6] as well as Mallett [15] were among the first to apply SIFT in embedded systems in order to perform multi-camera self-localization and calibration. In the work of Cheng *et al*. DoG keypoints are detected on high resolution images, and a PCA-based compression method is performed on the corresponding descriptors to reduce the amount of data to be transmitted between camera nodes. The feasibility of their approach was shown in a multi-camera simulation to determine vision graphs and camera localization. Note that this way of using PCA is remarkably different from the one used to generate PCA-SIFT descriptors as introduced in section 3.4. Furthermore, we point out that the huge amount of computational work in this approach presents a big issue and that the implementation of parts of the algorithms on a smart camera network presents a big challenge.

While the usage of local features and descriptors is not limited to the task of camera calibration, they have not been widely applied in the area of embedded systems by now, *e.g*. for object recognition. Yet, to the best of our knowledge there exists no completely embedded object recognition system which is based on local interest regions and descriptors. The only somehow related work we are aware of is the work of Yeh *et al*. [22]. In this work two images are taken with a camera equipped to a mobile phone, one image with and one without the object sought. An interactive segmentation tool is used to isolate the object and to submit its image as a query to a web-database. After recognizing the object, the database engine provides the user with useful information about the object, be it a famous building or a shopping item. At least the step of object segmentation - and thereby describing the object - is performed on the smart phone. The main strength of this algorithm is that it is in general not limited to any type of object since the recognition is done remotely using a more powerful device. We still believe that the usage of local features and descriptors could make the framework more efficient, at least in terms of communication costs via compression of the amount of data to be transmitted.

To sum up, there is only little literature about the usage of the algorithms described in the context of smart cameras. We want to bring local features and descriptors into this field more generally and intend to show that they can be successfully applied in various applications related to embedded systems.

3. Algorithm Selection

In this section we first describe the criteria for choosing dedicated algorithms for given tasks based on our hardware platform. Furthermore, we describe the algorithms we have selected due to their suitability to solve the two examples given. We justify the usage of these algorithms and outline their special relevance and qualification for usage on smart systems. We also shortly describe two descriptor matching methods and the robust algorithm for calculating the epipolar geometry from a given image pair.

3.1. Hardware Constraints and Selection Criteria

Our hardware platform is similar to the one used in [1] and represents a typical and popular set-up used in many applications. Hence, all algorithms run on a single Texas Instruments^{TM} TMS320C6414 DSP running at 600MHz with 1MB internal cache, the amount of external memory is 16MB.

Given a special task, the challenge is choosing the best algorithms currently available to solve the problem most efficiently under consideration of additional hardware constraints. Clearly the selection of algorithms has to be done according to special application dependent criteria too. Anyway, the best choice of algorithms must result in a system which is optimized in more than one aspect. For both applications we present in the next section there is more than one aspect to be considered during system design. In the first case, it is important that the recognition performance is good (choosing the right type of object) even under adverse conditions, while the time spent for recognition should be minimized. In the second case, for camera calibration, communication costs between individual camera entities should be minimized, but the overall number of

correct point correspondences (the matching performance) should still be kept at a high level to guarantee for a good calibration result.

In the following we describe the set of algorithms we have chosen for our two tasks to be solved most efficiently and note their properties which make them suitable for our purposes.

3.2. DoG Keypoints

The first algorithm we have investigated is Lowe's *Difference of Gaussian* (DoG) detector, which can be used to obtain rather accurate keypoints with high repeatability [14].

The DoG-detector takes the differences of Gaussian blurred images as an approximation of the scale normalized Laplacian and uses the local maxima and minima of the responses in scale space as an indicator for keypoints. The DoG-detector mainly delivers keypoints which indicate the presence of blob-like (more or less circular) structures in images. For each keypoint a circular region around the keypoint is cropped whose size is dependent on the scale factor delivered during detection. By summing up the gradients in the image patch, the main gradient direction is determined and assigned as orientation to the keypoint. A subsequent descriptor calculation is done on the image patch after rotating it to 0°.

A nice feature of the DoG detector is that it is almost purely based on image filtering and addition/subtraction operations. While a clever arrangement of filtering and search operations makes the algorithm also efficient in terms of memory usage, the algorithm is very well suited for DSP platforms, as they are mainly designed for fast filter operations. Moreover, the filtering can be implemented in fixed-point which results in a significant performance increase.

3.3. MSER

MSER stands for *Maximally Stable Extremal Regions* and was first proposed by Matas *et al*. [16]. This region detector is complementary to the DoG-detector and is based on searching for regions which possess an extremal property of the intensity function inside and on their outer boundary.

In short, the MSER detector searches for regions which are brighter or darker than their surroundings, *i.e*. are surrounded by darker, vice-versa brighter pixels. First, pixels are sorted in ascending or descending order of their intensity value, depending on the region type to be detected. The pixel array is sequentially fed into a union-find algorithm and a tree-like shaped data structure is maintained, whereas the nodes contain information about pixel neighborhoods, as well as information about intensity value relationships. Finally, nodes which satisfy a set of predefined criteria are sought by a tree walking algorithm, which in our case has to be iterative due to our architectural hardware constraints.

The big advantage of the MSER algorithm is that it is efficiently computable - at least on conventional desktop computers - and that the regions to be found are not restricted anyhow in terms of area or shape. Moreover, it is possible to identify regions across very large viewpoint changes because the extremal property of the regions in general does not change. For ease of implementation we have not implemented full-featured *Local Affine Frames* [17], but used the ellipse fitting approach of Mikolajczyk [19]. After fitting an ellipse to the region, the image patch below the ellipse is deskewed and rotated to 0° for the calculation of the descriptors [1].

An appealing feature of this algorithm is that it does not need any floating point arithmetics to be performed. However a union-find based algorithm creates a tree-like data structure, and though recursive algorithms are not suitable for DSP platforms, an iterative tree-walking algorithm has to be used. The shape of the tree is heavily dependent on the image processed, thus the runtime of the algorithm can not be estimated easily. Moreover, for building the data structure a large amount of memory is needed. The reason for choosing the MSER algorithm for implementation on the DSP is that its superior performance for identifying distinctive regions simply votes out all disadvantages. Moreover, when runtime is not a critical factor, the MSER algorithm might still be a valuable option.

3.4. PCA-SIFT

Ke and Sukthankar [11] proposed to use a compact descriptor based on eigenspace analysis, the so called PCA-SIFT descriptor. This descriptor has less in common with the original SIFT descriptor, proposed in [14], as one might suppose. They calculated a PCA (*Principal Component Analysis*) eigenspace on the gradient images of a representative number of over 20000 image patches. The descriptor of a new image tile is generated by projecting the gradients of the tile onto the precalculated eigenspace keeping only the d most significant eigenvectors.

This descriptor has several advantages, especially for our utilization. First, the descriptor is much more compact, because they have proven the $d = 36$ dimensional descriptor to exhibit the same discriminatory power as the 128-dimensional SIFT descriptor. A second big advantage is, that a further decrement of d results in only a slight loss in discriminatory power, thereby making the descriptor calculation itself scalable.

Fortunately, for application in a smart camera network the reduction of the descriptor length results in three more favorable effects; first, transmission costs are obviously reduced by a factor of (at least) 4. Secondly, the matching

[1] The drawback of this method is that two descriptors have to be calculated for each region, one for 0° and one for 180° due to the ambiguous orientation of the ellipse.

effort is also massively reduced because of the shortened descriptor lengths. Finally, the amount of storage for the descriptors is also reduced by a factor of ≥ 4 because of the smaller amount of memory needed to store the descriptors.

3.5. Descriptor Matching

An important part of most systems using local features is a descriptor matching engine. Efficient descriptor matching is a challenge on its own and a lot of distance metrics exist. One very popular metric is the *Euclidean distance*, which is defined as

$$dist(X,Y) = sqrt(\sum_{i=1}^{N}(x_i - y_i)^2) \quad (1)$$

with X and Y being vectors of length N, and x_i, y_i being the i-th element of vector X, respectively Y. Matching of descriptors is relatively expensive. The naive exhaustive search has a complexity of $\mathcal{O}(nmd)$ with n being the number of descriptors in a database, m the number of descriptors to be matched and d the dimension of the descriptor vectors. Making d smaller is one possible solution to reduce the computational load, using k-d trees and approximate-nearest-neighbor search algorithms is another one.

We have evaluated both matching approaches, a k-d tree with nearest-neighbor search and the exhaustive search method. Furthermore, we decided to use the *sum of squared distances* metric, which is very similar to the Euclidean metric except that the normalizing square root is omitted. Extracting the root is a very expensive step on a DSP; while the order of elements is not changed, whether the root is taken or not, we can safely omit this operation and still find the nearest-neighbor. Regarding our tasks, this is the crucial matter to make our object recognition system real-time capable, as will be shown later.

3.6. Epipolar Geometry

For calculating the epipolar geometry between an image pair, point correspondences between the images have to be established. Once having enough robust potential point correspondences it is possible to compute the cameras extrinsic parameters and estimate the fundamental matrix **F**, where $x'^T Fx = 0$ and x' and x are the corresponding features in the first, respectively the second image. Depending on the quality of the matches, this works for both stereo and wide-baseline setups [3]. Note, however, that in order to achieve high accuracy, point correspondences in general position should be distributed uniformly. Hence, for slightly overlapping views, different methods, *i.e.* [20], have to be applied.

For most scenarios, though, one of the simplest yet efficient ways to estimate F is the normalized 8-point algorithm more precisely described in [9]. In order to handle the many possible outliers an iterative matching method RANSAC (RANdom SAmple Concensus) [7] is applied. For *n* iterations RANSAC takes randomly eight points and calculates the fundamental matrix using the 8-point algorithm. After that a distance *d* for each putative correspondence is calculated. We used the Sampson distance measure which yields quite good results also described in [9]. Then the number of inliers consistent with **F** is determined. Finally, **F** with the largest number of inliers is taken. For a more precise algorithm overview again see [9].

Because the calculation of the fundamental matrix is very sensitive to outliers, a robust outlier detection algorithm is necessary. The special qualification of the RANSAC based outlier detection algorithm for our purposes is that it is easy to implement, and that it does not require us to store large amounts of data.

4. Experimental Section

Now we will evaluate our algorithms on the two challenging tasks given, namely object recognition and camera calibration. First we will describe the datasets and experimental setup used, then we list the timings for each separate module on our embedded platform, and afterwards we elucidate and discuss the results of our algorithms.

Figure 1. A subset of the 50 randomly selected objects in our database from the ALOI (*Amsterdam Library Object Images*), viewed from 0°.

4.1. Object Recognition

In this application a small-size database is deployed on our embedded system. The objects are described from one frontal view only and the recognition performance of the system is evaluated over a total viewpoint range of -60° to $+60^\circ$. The important factors during system setup are to guarantee that the average time needed for recognizing an object is below one second, and that the data buffers fit into the internal memory of the DSP.

Dataset and System Setup For our experiments we randomly choose a 50 object subset of the ALOI database [8]. All object images are resized to CIF resolution (352x288), for training as well as for the performance evaluations of our system. We take the objects from 0° frontal view only to extract our descriptors for the database. For each image

Figure 2. The four background images onto which we have projected the objects to further challenge our recognition system.

we first apply the DoG detector to acquire highly distinctive keypoints. The PCA-SIFT descriptor is subsequently used to describe the region patches around the keypoints in a flexible, yet efficient manner. The descriptors are appended to our database together with the object id. The final database is a list containing 7395 descriptors (≈ 148 descriptors per object on average). The step of database generation is done on a desktop computer and the database is then transfered to the DSP platform as a raw block of data. For testing the performance of our system, we supply the query objects from $\pm 60^\circ$ viewpoint change in steps of 10° to also challenge the descriptor calculation and matching algorithms. Furthermore, we project the objects onto four different background images to further complicate the task. During the recognition phase, the DoG and PCA-SIFT algorithms are applied on these images (see figure 3 for an example). For the keypoints and the corresponding descriptors the nearest neighbors in the descriptor space are determined using exhaustive matching or using a k-d tree. As each descriptor in the database is linked to an object id, the object votes are summed up in a 50-dimensional histogram and the final result is the object with the maximum votes in the histogram.

A subset of our 50 objects in our database is depicted in figure 1, the background images are depicted in figure 2. To illustrate the difficulty of the task, the projection of all views of a single object projected onto the "graffiti" scene is shown in figure 3.

Recognition Performance Evaluation For the performance evaluation of an object recognition system there is a list of determining factors. On the one hand it is desirable that the system is able to act in real-time. On the other hand, the recognition power of the system has to be satisfying too,

	-60°	-50°	-40°	-30°	-20°	-10°	0°
No BG	32%	40%	62%	86%	96%	100%	100%
Graffiti	28%	34%	54%	82%	92%	94%	100%
Boat	22%	28%	56%	88%	88%	96%	100%
Sea	22%	36%	62%	86%	94%	100%	100%
Trees	28%	30%	54%	82%	88%	96%	100%

	10°	20°	30°	40°	50°	60°	
No BG	100%	98%	98%	70%	46%	38%	
Graffiti	100%	92%	86%	64%	42%	36%	
Boat	100%	92%	86%	62%	38%	22%	
Sea	100%	96%	92%	64%	38%	32%	
Trees	100%	92%	92%	58%	40%	28%	

Table 1. Detailed results for our recognition setup. The recognition rate without background clutter is only slightly better than the ones with background clutter. This owes to the distinctiveness of the PCA-SIFT descriptor and proofs our choice favorable.

because *real-time* is really meaningless for a system without sufficient functionality. To state it differently, *real-time* is a flexible term given environmental conditions. In our current setup there was no critical time limit to meet, thus a good recognition performance was the primary goal. Anyway, a time span of more than a second might be too long in an other scenario, making a user of the system impatient having to wait for the result.

In figure 4 the results for the recognition performance evaluation of our system using 36-dimensional PCA-SIFT are depicted. As can easily be seen, the performance is really satisfying, even in the case where background clutter is present. For better visibility, we averaged all recognition results with background images and drew only these averaged results. A detailed listing of results can be found in table 1. The recognition performance is between 80% and 100% for $\pm 30^\circ$, but notably drops when the change in viewpoint is more severe. These results are consistent with the results presented in [18, 19].

Time and Memory Consumption In table 2 the final timing results of our object recognition system on our platform are listed. We divide the DoG keypoint detection algorithm into several stages, where the timing results are based on an average point detection rate per 352x288 image of 198 keypoints (object projected onto arbitrary background). The DoG detector is quite fast due to the suitability of filter-based algorithms for application on DSPs. Nevertheless, the assignment of the orientation involves calculation of gradients and summation over subwindows, which takes a little longer. The PCA-SIFT Descriptors are calculated on 41x41 image patches centered at the keypoint, as in the work of Ke and Sukthankar [11]. This value also includes the time necessary for rotating, bilinear resizing and cropping the patch from the original image. The time spent altogether is comparable to the one for orientation assignment. As mentioned in section 3.5, we evaluated both matching methods, exhaustive search and a k-d tree with nearest-neighbor search.

Figure 3. A single object viewed from of -60° to $+60^\circ$ in steps of 10°, projected onto the "graffiti" background (the frontal view is depicted twice).

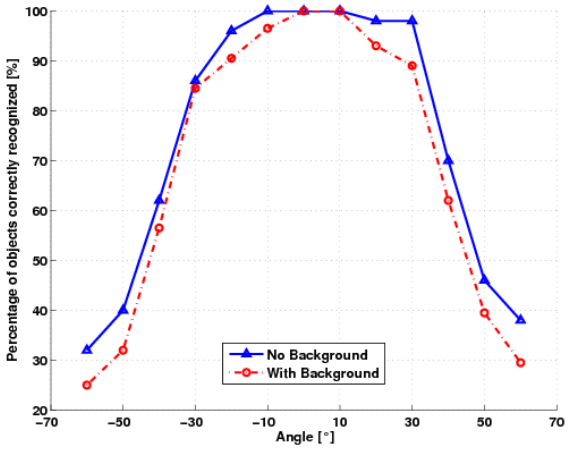

Figure 4. Recognition performance for a database of 50 objects over a viewpoint change of $\pm 60^\circ$. For better visibility we averaged the recognition rate of all background results into one single plot. The recognition results for full range and all separate background results are given in table 1.

On average 198 descriptors are matched against 7395 in our database. Obviously the k-d tree is much more efficient, although matching is still the most time consuming step in our system. The high standard deviation in orientation assignment and descriptor matching is an indicator for the high variation of the number of keypoints detected in the images, which ranges from only 8 keypoints up to 709 keypoints at max. Recognizing an object takes about 0.88 seconds on average. Please note, that almost 70 % of the total time is consumed during the matching step. This is an indication that the matching procedure should further be optimized, *e.g.* by using *Best-Bin-First* as proposed by Lowe [14].

The amount of memory is an evident result of the PCA data ($\approx 220kB$) and the descriptor database ($\approx 260kB$) for recognition using *short*, respectively *char* datatypes for storage. Storing the descriptors in a k-d tree instead of accessing the data sequentially by using exhaustive search additionally requires about 145 kB.

	avg. time [ms]	std. dev.
Scale Space Generation	29.16*	0.013*
Extrema Search	21.66*	0.947*
Keypoint Orientation Assignment	111.84*	11.427*
PCA-SIFT Calculation	107.39*	53.779*
Matching (k-d tree)	607.06*	306.988*
Matching (exhaustive)	930.42	488.135
Total:	**877.11***	**372.188***
PCA-SIFT Calculation / Descriptor	0.56	0.071
Matching (1 vs. 7395) (k-d tree)	3.07	0.219
Matching (1 vs. 7395) (exhaustive)	4.69	0.015

Table 2. Timing results of our object recognition system, based on an average keypoint detection rate of 198 keypoints. For completeness we have also included the results for calculating a single descriptor and for matching the descriptor against our database. Since we have evaluated two descriptor matching approaches, we have also included the time needed for descriptor matching using exhaustive search. The values marked with an asterisk are considered to calculate the average total time needed for recognizing a single object.

4.2. Camera Calibration

Our second test scenario is camera calibration. In the following we will describe our experimental setup and give notes about our configuration choices and why we have done so. While camera calibration usually has to be done only once during deployment, setup time is not necessarily a critical factor. It is much more important that the number of point correspondences is high enough, and that the major amount of correspondences is correct. Moreover, in a camera network it is important to minimize the amount of data to be transmitted.

System Setup As the MSER detector has been proven to be a good choice for the task of wide-baseline camera calibration, we choose the camera calibration task to test its performance together with the PCA-SIFT descriptor on our platform. A limiting factor in this task is image size and resolution. On the one hand, it is hard to calibrate from

low resolution images, on the other hand, the usage of high resolution images results in a higher memory consumption, which is critical especially on embedded systems. Thus, we decided to split the 680x510 test images, which are depicted in 6, into 4 tiles, each of size 352x288 with a small overlap area. The images are separated by an approximately 30° viewpoint change. We are aware that this configuration is neither wide-baseline, nor does it provide a good testbed for proofing the strengths of the MSER detection algorithm together with a robust calibration algorithm. Anyhow, we simply want to demonstrate the approach so we did not choose a more difficult scenario. Furthermore, it is common to tile images into smaller parts if memory and computational resources are limited, thus this setup makes it possible to process and run our framework without much additional programming overhead.

Figure 5. Image tiles cropped with overlayed MSER detection results for positive regions and negative regions on the left, respectively right side. The images are separated by an approximately 30° viewpoint change. As can easily be seen the regions found by the algorithm are almost the same for both views.

We process both sets of image tiles sequentially on a single instance of our platform, only storing and exchanging the region coordinates and the corresponding descriptors as if they were passed between separate smart cameras. After calculating positive and negative MSERs, we calculate the PCA-SIFT descriptors on the deskewed patches for both types of regions separately. Also the subsequent matching of the descriptors of each type is done separately to avoid additional wrong matches. The descriptors are matched using exhaustive search and putative point correspondences are established. The RANSAC-based fundamental matrix calculation algorithm is finally used to eliminate outliers and to calculate the epipolar geometry of the image pair.

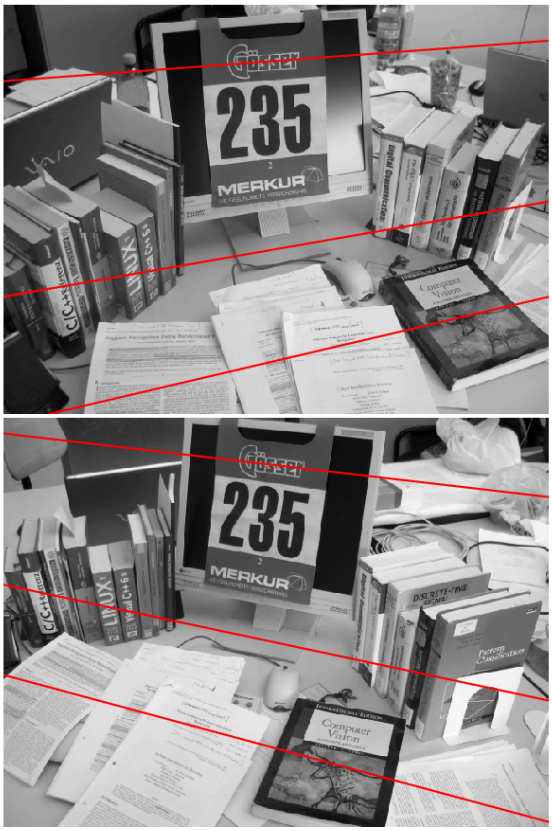

Figure 6. Our calibration scenario, on which we have calculated the fundamental matrix, and three corresponding epipolar lines

	avg. time [ms]	std. dev.
Tree Data Structure Building	412.96	67.324
Tree Walking Algorithm	2572.86	712.395
Ellipse Fitting	51.38	28.293
PCA-SIFT Calculation	343.17	144.945
Total:	**3380.37**	**768.890**
Ellipse Fitting / MSER	0.13	0.021
PCA-SIFT Calculation / Descriptor	0.57	0.010

Table 3. Timing results for the MSER detection algorithm. The results are obtained detecting positive or negative regions separately for about 358 (positive or negative) regions and an average number of 597 descriptors to be calculated.

Calibration Results In figure 5 the results of our MSER detection on two sample tiles of our image pair for positive and negative regions are depicted. The detection algorithm enumerates the same parts in both images as interest regions. In our calibration experiment the algorithm altogether detects 951 regions in the first image and 976 regions in the second image respectively. In figure 6 three corresponding epipolar lines are overlayed on our calibration test images. The average reprojection error is in the range of a few pixels.

Timing Results In table 3 the timing results for our implementation of the detector are listed. We ran the detection algorithm on the images used for our object recognition experiment. All results are based on an average detection of 358 regions per image and 597 descriptors[2]. We have not listed the amount of time necessary for calibrating an image pair with our complete calibration framework. The reason for doing so is that the time span needed is heavily dependent on the image data, *i.e*. on the number of descriptors to be matched, and especially on the number of RANSAC iterations needed for robustly identifying and discarding outliers. For our calibration scenario, it takes our system less than a minute to calculate the epipolar geometry.

The runtime of all parts of the MSER detection algorithm is heavily dependent on the image data. Furthermore, it is somewhat slower than the DoG detector, which owes to its algorithmic workflow. Random memory accesses, the necessity of linked lists and the tree-like shape of the data structure disunites the architectural strengths of the platform and the algorithm. Nevertheless, MSER is one of the most popular approaches and has been shown to perform very well for this task. Due to the fact that camera calibration has to be done only once during setup, the time needed for detection is not critical and thus the algorithm can be used for this purpose without any complaints.

5. Conclusion

In this paper we presented our investigation of a set of local features and their suitability for embedded systems. We used the best state-of-the-art detectors, MSER and DoG, for selection of interest regions, and combined them with one of the most promising descriptors, the PCA-SIFT descriptor. All algorithms were fully implemented and tested on a single-chip based embedded platform and their suitability was shown on the popular tasks of camera calibration and object recognition. Doing so we further narrowed the gap between high-level state-of-the-art vision and resource constrained embedded systems.

Future work will both concentrate on additional algorithm improvements and a system expansion to additional applications such as multi-camera tracking. Moreover, we aim to employ better region descriptors such as local affine frames in order to allow wide-baseline calibration and localization.

References

[1] C. Arth, H. Bischof, and C. Leistner. TRICam: An Embedded Platform for Remote Traffic Surveillance. In *CVPR (ECV Workshop)* , 2006.

[2] A. Barton-Sweeney, D. Lymberopoulos, and A. Savvides. Sensor localization and camera calibration in distributed camera sensor networks. In *IEEE BaseNets* , 2006.

[3] H. Bay, V. Ferrari, and L. van Gool. Wide-baseline stereo matching with line segments. In *CVPR* , pages 329–336, 2005.

[4] M. Bhardwaj, A. Chandrakasan, and T. Garnett. Upper bounds on the lifetime of sensor networks. In *ICC* , pages 785 – 790, 2001.

[5] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed Embedded Smart Cameras for Surveillance Applications. *Computer*, 39(2):68–75, 2006.

[6] Z. Cheng, D. Devarajan, and R. J. Radke. Determining vision graphs for distributed camera networks using feature digests. *EURASIP Journal on Advances in Signal Processing*, 2007.

[7] M. A. Fischler and R. C. Bowles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 24(6):381–395, 1981.

[8] J.-M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders. The Amsterdam library of object images. *IJCV* , 61(1):103–112, 2005.

[9] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[10] J. Jannotti and J. Mao. Distributed calibration of smart cameras. In *Workshop on Distributed Smart Cameras* , 2006.

[11] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *CVPR* , pages 506–513, 2004.

[12] A. Keshavarz, A. M. Tabar, and H. Aghajan. Distributed vision-based reasoning for smart home care. *Workshop on Distributed Smart Cameras (DSC06)*, 2006.

[13] H. Lee and H. Aghajan. Collaborative node localization in surveillance networks using opportunistic target observations. In *VSNN* , 2006.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV* , 60(2):91–110, 2004.

[15] J. Mallett. *The Role of Groups in Smart Camera Networks*. PhD thesis, MIT, February 2006.

[16] J. Matas, O. Chum, U. Martin, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *BMVC, vol.1*, pages 384–393, London, September 2002.

[17] J. Matas, S. Obdrzalek, and O. Chum. Local affine frames for wide-baseline stereo. *ICPR* , 04:40363, 2002.

[18] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI* , 27(10):1615–1630, 2005.

[19] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *IJCV*, 65(1-2):43–72, 2005.

[20] R. Pflugfelder and H. Bischof. Fundamental matrix and slightly overlapping views. In *ICPR* , 2006.

[21] W. Wolf, B. Ozer, and T. Lv. Smart cameras as embedded systems. *Computer*, 35(9):48–53, 2002.

[22] T. Yeh, K. Grauman, K. Tollmar, and T. Darrell. A picture is worth a thousand keywords: image-based object search on a mobile platform. In *CHI Extended Abstracts*, pages 2025–2028, 2005.

[2] Descriptors are only calculated on regions for which the extracted image patch completely lies in the original image. Thus the average number of descriptors is not twice the number of regions found.