Bauer, J., Sünderhauf, N., & Protzel, P. (2007). Comparing Several Implementations of Two Recently Published Feature Detectors. In Proc. of the International Conference on Intelligent and Autonomous Systems, IAV, Toulouse, France.

COMPARING SEVERAL IMPLEMENTATIONS OF TWO RECENTLY PUBLISHED FEATURE DETECTORS

Johannes Bauer, Niko Sünderhauf, Peter Protzel

Department of Electrical Engineering
and Information Technology
Chemnitz University of Technology
09111 Chemnitz
Germany
johannes.bauer@5pm.de
{*niko.suenderhauf, peter.protzel*}*@etit.tu-chemnitz.de*

Abstract: Detecting, identifying, and recognizing salient regions or feature points in images is a very important and fundamental problem to the computer vision and robotics community. Tasks like landmark detection and visual odometry, but also object recognition benefit from stable and repeatable salient features that are invariant to a variety of effects like rotation, scale changes, view point changes, noise, or change in illumination conditions. Recently, two promising new approaches, SIFT and SURF, have been published. In this paper we compare and evaluate how well different available implementations of SIFT and SURF perform in terms of invariancy and runtime efficiency.

1. INTRODUCTION

Recently, two promising approaches to detect salient regions in images have been published: SIFT, Lowe (2004) (Scale Invariant Feature Transform) and SURF, Bay et al. (2006) (Speeded Up Robust Features). Both approaches do not only *detect* interest points or so called features, but also propose a method of creating an invariant *descriptor*. This descriptor can be used to (more or less) uniquely identify the found interest points and match them even under a variety of disturbing conditions like scale changes, rotation, changes in illumination or viewpoints or image noise. Exactly this invariancy is most important to applications in mobile robotics, where stable and repeatable visual features serve as landmarks for visual odometry, SLAM, or support object recognition. Of course, there was and still is a variety of other methods available that claim to achieve the same goals.

Comparative work done by Mikolajczyk and Schmid (2005) is a good reference for a list of different other interest points detectors and descriptors and proves that SIFT outperforms other algorithms in terms of invariancy. The authors of SURF, Bay et al. (2006), in return claimed SURF to be superior to SIFT in terms of runtime efficiency while still yielding comparably good results with regards to feature point quality.

We felt a need to compare SIFT against SURF using images taken in a natural outdoor environment, as this is the environment our robots operate in. We also wanted to compare different available implementations of SIFT to reveal possible differences in their performance. This paper presents our results.

2. COMPARED FEATURE DETECTORS AND DESCRIPTORS

We compared and evaluated three different implementations of SIFT and two different SURF parameter settings.

- SIFT
  - · original implementation by David Lowe
  - · SIFT++
  - · LTI-lib SIFT
- SURF
  - · SURF
  - · SURF -d

2.1 Harris Corner Detector

In addition to the different SIFT and SURF implementations, the Harris Corner Detector by Harris and Stephens (1988) was used as a comparison how a very naive approach would perform in the given test cases. It is a well known feature point detector and has been used for many years in computer vision. It only *detects* interest points in the image but does not calculate a *descriptor*. A very simple and naive strategy, that formed the descriptor from the raw grey values of the pixels in a certain neighborhood of the identified interest point was chosen.

2.2 Original SIFT by Lowe

The SIFT algorithm as published by Lowe (2004) includes both a keypoint *detector* and *descriptor*. Keypoint *detection* is done by building a scale-space representation of the original image. (See Lindeberg and ter Haar Romeny (1994) for an introduction into scale-space theory.) This is achieved by repeatedly convolving the image with a Gaussian function. A Difference of Gaussian (DoG) approach combined with interpolation over the scale-space leads to the locations of stable keypoints in that scale-space representation of the image. After that localization, each keypoint is assigned an orientation, which leads to the desired rotation invariancy. The keypoint *descriptors* are calculated from the local gradient orientation and magnitudes in a certain neighborhood around the identified keypoint. The gradient orientations and magnitudes are combined in a histogram representation, from which the descriptor is formed as a normalized vector of 128 elements.
Se et al. (2002) and Lowe (2004) demonstrated how SIFT can be used for object recognition or vision based SLAM.
Lowe's original implementation is freely available as a closed-source library written in C from Lowe's website[1].

2.3 SIFT++

SIFT++ is a free and open C++ implementation of the SIFT detector and descriptor and can be downloaded from the author's homepage[2]. It has been developed by Andrea Vedaldi from the Vision Lab of the University of California. A free and open Matlab version of SIFT is also available from the same author.

2.4 LTI-lib SIFT

LTI-lib[3] is a LGPL-licenced C++ library that contains algorithms and datastructures frequently used in computer vision. LTI-lib includes routines for image processing, linear algebra, classification and clustering as well as visualization and drawing tools. Two classes implement the SIFT detector and classificator: `lti::pyramidLocationSearch` and `lti::loweGradientFeature`.

2.5 SURF

SURF (Speeded Up Robust Features) has been recently published by Bay et al. (2006). Like SIFT, the SURF approach describes a keypoint *detector* and *descriptor*. Keypoints are found by using a so called Fast-Hessian Detector that bases on an approximation of the Hessian matrix for a given image point. The responses to Haar wavelets are used for orientation assignment, before the keypoint descriptor is formed from the wavelet responses in a certain surrounding of the keypoint. The descriptor vector has a length of 64 floating point numbers but can be extended to a length of 128. As this did not significantly improve the results in our experiments but rather increased the computational costs, all results refer to the standard descriptor length of 64.
SURF is available as a precompiled library, where the core algorithm is closed source. It can be downloaded from the author's website[4].

2.6 SURF -d

SURF -d is the normal SURF algorithm with initial image resolution doubling activated.

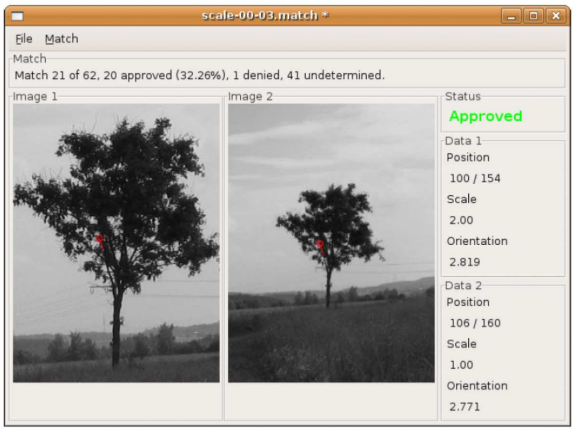
[1] http://www.cs.ubc.ca/~lowe/keypoints/
[2] http://vision.ucla.edu/~vedaldi/code/siftpp/siftpp.html
[3] http://ltilib.sourceforge.net
[4] http://www.vision.ee.ethz.ch/~surf/

3. EVALUATION

We tested all of the above algorithms with a dataset of images to confirm and evaluate their invariancy against

- rotation
- scale change
- image noise
- change in lighting conditions
- change of view point

The dataset of images was created from natural outdoor scenes. Starting from an initial image, the images were altered according to the performed test, i.e. they were rotated or the camera zoomed closer to the object etc. Feature points and their descriptors were determined in the initial and the secondary images. As both SIFT and SURF descriptors can be interpreted as integer or floating point vectors respectively, the matching strategy was to find the descriptor from the initial image that had the smallest euclidean distance to a given descriptor in one of the secondary images. A small tool written in Python (fig. 1) was used to determine which of the found matches were correct. The total number of correct matches was recorded for the evaluation as well as the *ratio* of correct and incorrect matches. The later one tells more about the quality of the implementation than the mere number of keypoints. Finally, the runtime efficiency was compared by measuring the time the algorithms took for identifying and matching the keypoints.


scale-00-03.match *

File Match

Match
Match 21 of 62, 20 approved (32.26%), 1 denied, 41 undetermined.

Image 1 | Image 2

Status
Approved

Data 1
Position
100 / 154
Scale
2.00
Orientation
2.819

Data 2
Position
106 / 160
Scale
1.00
Orientation
2.771

Fig. 1. This tool was used to determine the correct and incorrect matches.

3.1 Image Dataset

Figures 2 to 4 show the images that were used in the several test cases.

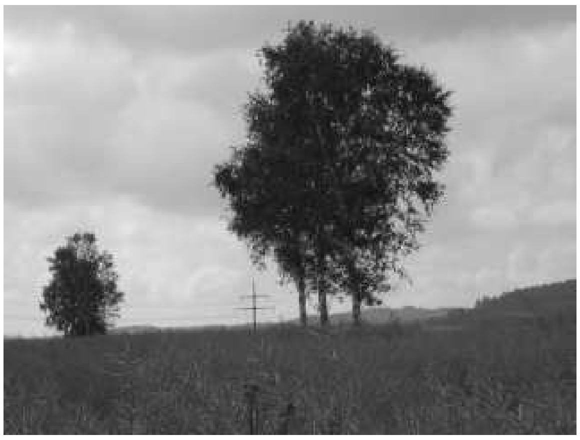


Fig. 2. The initial image used during the test for rotation invariance. It was rotated in steps of 30°.

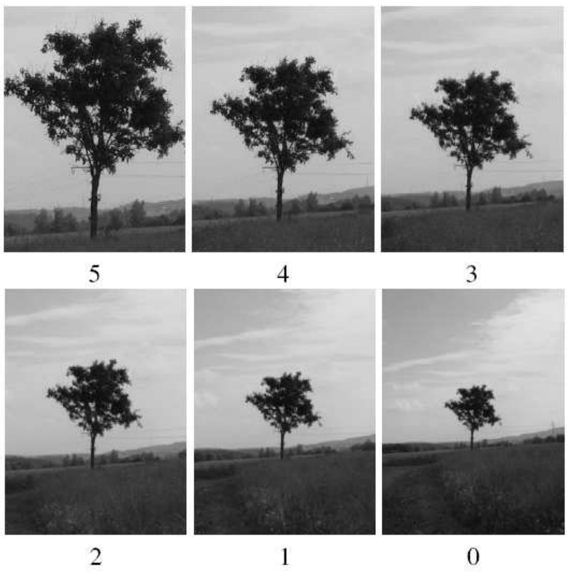


Fig. 3. The images used for the scale invariancy test. Image 5 was chosen as the initial image, i.e. the keypoints found in all other images were matched with the ones from image 5.

3.2 Results

The first thing that became obvious during the tests was that the total number of keypoints is generally higher for SIFT and SIFT++ than it is for SURF or SURF-d. However, the quality of the matches is almost equal for the four implementations (SIFT, SIFT++, SURF, and SURF-d), with small advantages for the two SIFT algorithms.
The LTI-lib implementation of SIFT seems to be erroneous or at least does not follow Lowe's original algorithm as close as SIFT++ does: The ratio of correct matches was usually far worse for LTI-lib SIFT than for Lowe's SIFT or SIFT++.
As expected, the naive Harris approach was not able to compete against the other approaches. Due to the deficiencies of Harris and LTI-lib SIFT we are not going to mention their results in detail here.

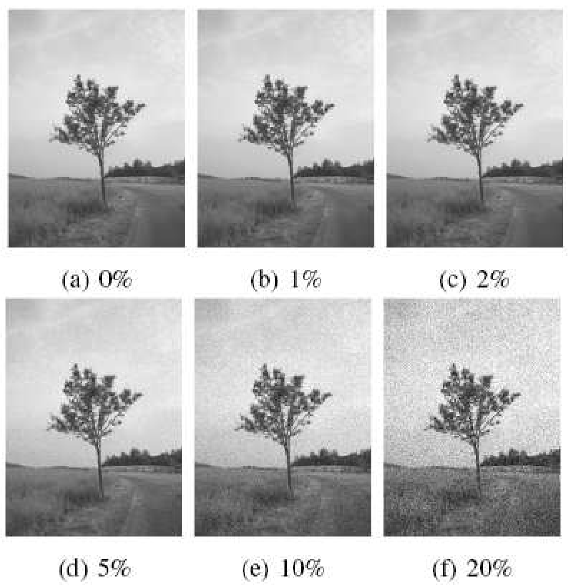


Fig. 4. A series of images used for the noise invariancy test showing increasing amount of noise.

The test for rotation invariance was passed without problems by the remaining four implementations, as expected. The ratio of correct matches was well above 95% for all algorithms. Figure 5 illustrates the results of the rotation test.

Scale changes hardly influence the matching quality of the original SIFT implementation. Figure 6 shows the quality of the matches found by SURF and especially SIFT++ drops at the highest scale change while Lowe's original SIFT algorithm stays well close to the 100% margin. As expected, the total number of matches decreases with increasing scale change.

As can be seen from figure 7, image noise does hardly influence the high quality of keypoint matching but increasing noise significantly reduces the total number of matched keypoints by all algorithms (except standard SURF which shows only a slight decrease). All approaches performed almost identically well, the ratio of correct matches was around 100% to 95%.

All algorithms can cope with changing illumination conditions up to a certain extend where simply not enough keypoints are generated. This breakdown is clearly visible in figure 8. Illumination level 8 corresponds to the initial image. Level 0 is the darkest image, where hardly any feature was found, except for SIFT++. The illumination changes were not generated artificially, but the images were rather taken during sunset.

Changing the viewpoint, i.e. observing an object under a different angle, has a large negative impact to all algorithms. The total number of matched features drops significantly even after viewpoint changes of only 10 degrees. This early breakdown may be caused by the motif that was used during the test (again a natural scene with a tree and some bushes beneath it), as the authors of SIFT and SURF report viewpoint invariancy of up to 30 degrees.

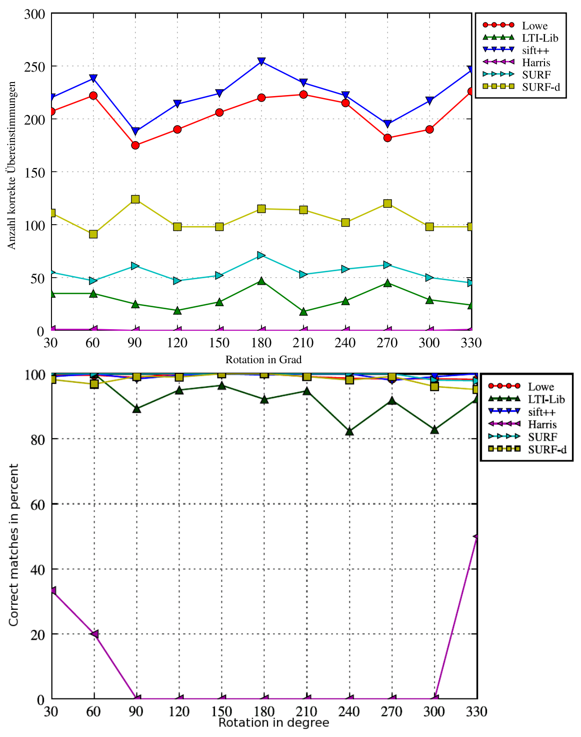


Fig. 5. Results of the test for rotation invariance. Notice that Lowe, SURF, SURF-d and SIFT++ yield almost identically good results. LTI-lib is notably worse, Harris performs out of the question.

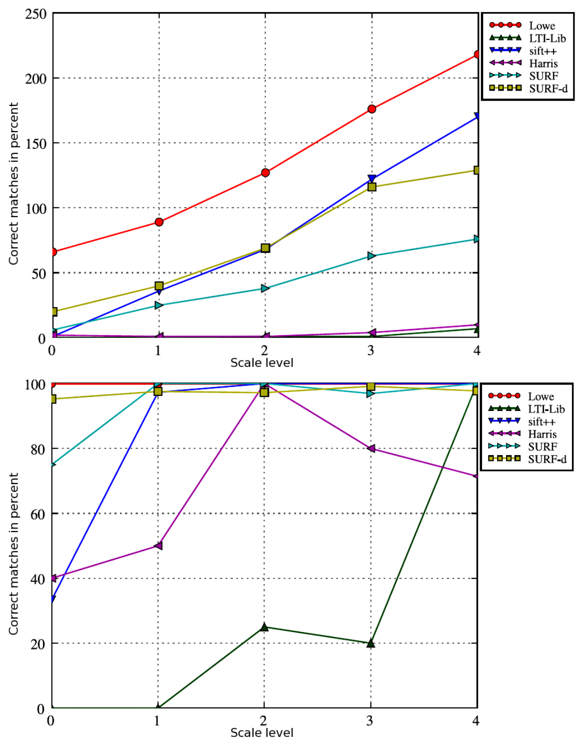


Fig. 6. Results of the scale invariancy test.

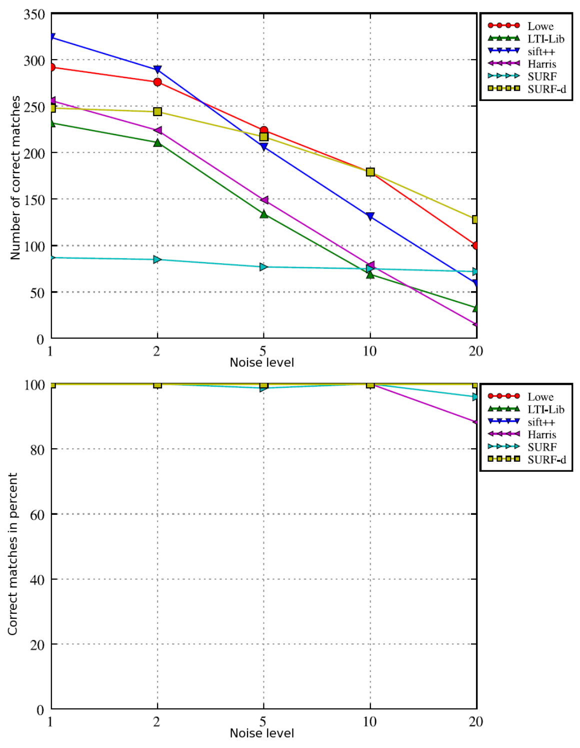


Fig. 7. Results of the scale invariancy test.

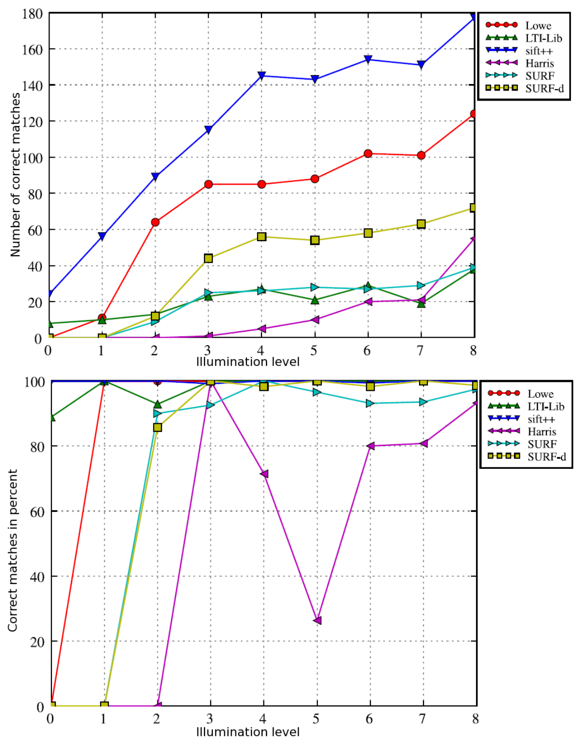


Fig. 8. Results of the test of invariance to changing illumination conditions.

4. CONCLUSION

During all of our tests, Lowe's SIFT and SIFT++ performed best in terms of match ratio and total number of correct matches. Regarding the ratio of correct to incorrect matches, SURF -d and SURF followed very closely. Although they usually produce fewer keypoints, this is not a disadvantage

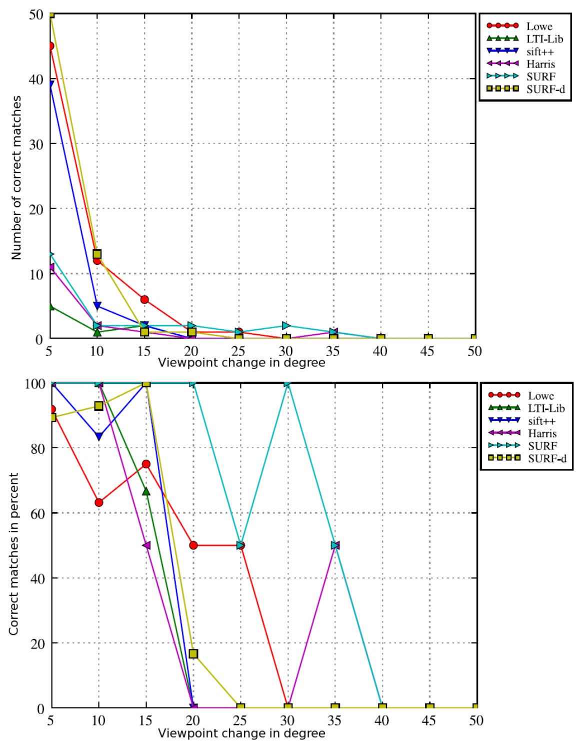


Fig. 9. Results of the viewpoint invariancy test.

in our eyes: In many applications there is simply no need for 300 feature points per image. What is more important is that the feature points can be calculated quickly and that they can be recognized and matched with feature points from other images in an efficient way.
Fig. 10 plots the number of correctly found matches per second for all of the conducted tests and algorithms. It can be clearly seen that both versions of SURF are more performant than SIFT and produce much more correct matches per time interval. So although the quality and total number of the created keypoints and their descriptors are slightly better for SIFT, one has to pay for this with a disproportionate increase in computing time. As, in our opinion, the better runtime performance outweights the slightly better feature quality, we switched to using SURF instead of SIFT in several of our projects like visual odometry or large-scale visual FastSLAM as it is described in Neubert et al. (2007).
To conclude, we showed that SURF is indeed superior to SIFT and all its different implementations. If for some reason SIFT is preferred anyway, the open SIFT++ implementation is a good alternative to the closed original library version. The LTI-lib implementation of SIFT seems erroneous and should not be used.

REFERENCES

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In

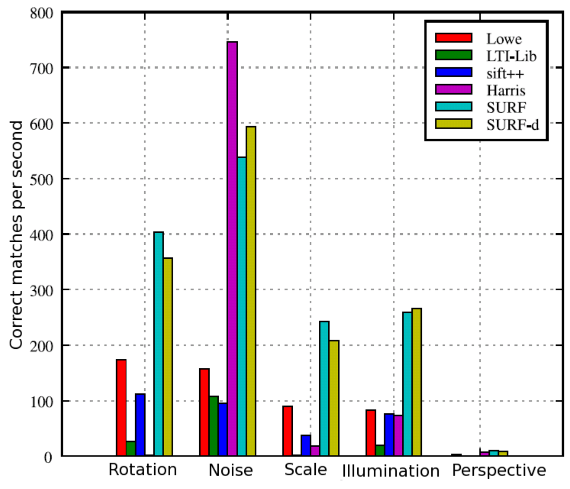


Fig. 10. A series of images used for the noise invariancy test.

Proceedings of the ninth European Conference on Computer Vision, May 2006.

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

T. Lindeberg and Bart M. ter Haar Romeny. Linear scale-space. In Bart M. ter Haar Romeny, editor, *Geometry-Driven Diffusion*, pages 1–77, Dordrecht, Netherlands, 1994. Kluwer Academic Publishers.

David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. In *International Journal of Computer Vision, 60, 2*, pages 91–110, 2004.

Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(10):1615–1630, 2005. URL `http://lear.inrialpes.fr/pubs/2005/MS05`.

Peer Neubert, Niko Sünderhauf, and Peter Protzel. Fastslam using surf features: An efficient implementation and practical experiences. In *Proceedings of the International Conference on Intelligent and Autonomous Vehicles, IAV07*, Tolouse, France, September 2007.

Stephen Se, David G. Lowe, and Jim Little. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. In *International Journal of Robotics Research, 21, 8*, pages 735–758, 2002.