Int J Comput Vis (2011) 94:335–360
DOI 10.1007/s11263-011-0431-5

Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking

Steffen Gauglitz · Tobias Höllerer · Matthew Turk

Received: 21 April 2010 / Accepted: 7 March 2011 / Published online: 31 March 2011

Abstract Applications for real-time visual tracking can be found in many areas, including visual odometry and augmented reality. Interest point detection and feature description form the basis of feature-based tracking, and a variety of algorithms for these tasks have been proposed. In this work, we present (1) a carefully designed dataset of video sequences of planar textures with ground truth, which includes various geometric changes, lighting conditions, and levels of motion blur, and which may serve as a testbed for a variety of tracking-related problems, and (2) a comprehensive quantitative evaluation of detector-descriptor-based visual camera tracking based on this testbed. We evaluate the impact of individual algorithm parameters, compare algorithms for both detection and description in isolation, as well as all detector-descriptor combinations as a tracking solution. In contrast to existing evaluations, which aim at different tasks such as object recognition and have limited validity for visual tracking, our evaluation is geared towards this application in all relevant factors (performance measures, testbed, candidate algorithms). To our knowledge, this is the first work that comprehensively compares these algorithms in this context, and in particular, on video streams.

Keywords Interest point detectors · Feature descriptors · Visual tracking · Dataset · Evaluation

S. Gauglitz (✉) · T. Höllerer · M. Turk
Dept. of Computer Science, University of California, Santa Barbara, Santa Barbara, CA 93106-5110, USA
e-mail: sgauglitz@cs.ucsb.edu

T. Höllerer
e-mail: holl@cs.ucsb.edu

M. Turk
e-mail: mturk@cs.ucsb.edu

1 Introduction

Visual tracking is a core component for many applications including visual odometry (Cheng et al. 2006; Nistér et al. 2004), visual Simultaneous Localization and Mapping (SLAM) (Davison et al. 2007) and Augmented Reality (AR) (Klein and Murray 2007). Although the specific requirements of these systems may vary with the application, they all require the underlying visual tracking to be robust, accurate, and fast enough to be computed in real time. While some of the underlying techniques and algorithms have been known for a longer time, visual tracking in real time is a fairly young endeavor, enabled by the rapid increase in computation power of modern hardware and the availability of compact and cheap cameras.

While some visual odometry systems work with optical flow, most tracking applications use feature-based visual tracking. In this case, interest point detection and feature description are the first steps of the system. Many algorithms have been proposed to accomplish these tasks, and existing visual tracking systems use differing approaches. There is however little recent literature quantitatively comparing these approaches, and existing evaluations are mostly geared towards other applications such as object recognition, which limits the significance of obtained results for visual tracking. In particular, we are not aware of any work that compares the respective algorithms on video streams, which is the setup of interest for visual tracking.

Main Contributions The main contributions of this work fall into the following two categories:

1. We present a carefully designed dataset with ground truth. It comprises 96 video streams displaying six differently textured planar tracking targets with a total of 6889 frames, featuring geometric distortions (zoom

and rotation around three different axes), nine accurately controlled levels of motion blur, as well as different lighting conditions, with all frames affected by natural amounts of noise. This dataset may serve as a testbed/benchmark for a variety of algorithm classes related to visual tracking, including: interest point detectors, feature descriptors, detector-descriptor-based tracking, optical flow-based tracking, tracking-by-detection, template- or model-based tracking, feature matching and outlier removal strategies (Sect. 3).
2. We present an extensive evaluation of detector-descriptor-based tracking, including two recognition-by-classification methods. Our evaluations use the above dataset and encompass four stages: (a) evaluation of each algorithm's parameters in order to quantify their effect on tracking performance and determine the optimal configuration for our task, (b) comparison of detectors in isolation, (c) comparison of descriptors (including classifiers) in isolation, (d) comparison of each detector-descriptor combination as a tracking solution (Sects. 5 and 6).

In this order, the work develops from broadly applicable towards more focused: the dataset was specifically designed to be applicable to a broad spectrum of problems. The isolated evaluation of the detectors (stage (b)) is also relevant to several applications (specifically as interest point detection is likely to be the first step of any processing chain in which it is used), while the last stage (d) is clearly focused on feature-based visual tracking and includes certain assumptions (see Sects. 5 and 6) that might not be valid in other contexts. The isolated evaluations will also prove useful to analyze the results of the last stage (d) in detail.

Our work builds upon an improved version of the evaluation setup introduced in Gauglitz et al. (2009).

Differentiation Visual tracking is a broad term that is used for a variety of problems. While some of our results are applicable to a wider scope, we would like to clarify our notion of visual tracking and thus the scope of this work as follows: We are primarily interested in six-degree-of-freedom pose estimation of the observer (or vice versa, a single or multiple well-defined objects with respect to the observer), i.e., *inside-out* tracking or camera tracking. We describe a number of applications which fall into this category in Sect. 2.2. Despite the obvious commonalities, this is very different from *outside-in* tracking (such as tracking humans by way of a static video surveillance camera) in various aspects—e.g., the number of spatial degrees of freedom, the size of the target object, assumptions about deformations and static background, and treatment and significance of moving objects—and therefore other algorithmic approaches and representations are interesting. See Yilmaz et al. (2006) for a survey on object tracking.

Restriction to Planar Textures The proposed setup (in particular, our method of obtaining ground truth, cf. Sect. 3.1) limits our dataset and hence our evaluations to planar textures. This is an important restriction, and datasets that include general 3D objects have been proposed (Moreels and Perona 2007; Winder and Brown 2007). However, as we will discuss in Sect. 2.1, none of these setups feature a moving camera and hence the datasets lack artifacts that are crucial for the evaluation of visual tracking. It is our view that, especially in its application to inside-out tracking, the restriction to planar scenes does not devalue the comparison, in particular since we focus on the first, local, image-based steps of tracking, *not* the pose estimation part. Here, the difference of 2D vs. 3D is less relevant, since these steps are inherently two-dimensional—at this point, no structural information about the scene is known anyways—and many conditions that are not explicitly modelled and hence challenge the algorithms' robustness are represented in our dataset (e.g., out-of-plane rotation). Effects not represented (such as extreme local non-planarity and occlusion) cannot be handled by any of the state-of-the-art approaches on this level and have to be handled by the further steps (i.e., treated as outliers) in either case.

Outline This paper is structured as follows: In Sect. 2, we discuss literature on datasets, systems that employ visual tracking, algorithms for interest point detection and feature description, as well as existing comparisons of these algorithms. Section 3 describes the collected dataset in detail, including the method of how ground truth was obtained. In Sect. 4, we provide an introduction to detector-descriptor-based visual tracking, including a compilation of published systems and a short review of each detector and descriptor that is included in the evaluation. Section 5 details the setup for the evaluation, including exact definitions of the performance measures and implementation details. Section 6 presents and analyzes the obtained results. Finally, Sect. 7 presents our conclusions.

2 Related Work

2.1 Datasets

In many domains, certain datasets have successfully been established as de-facto standards and used to compare and evaluate state-of-the-art algorithms; for example, the Middlebury dataset for multi-view reconstruction (Seitz et al. 2006). The dataset of Mikolajczyk et al. (2005) has been used for several comparisons of locally invariant detectors and descriptors. However, these sets consist of only a few static, high-resolution images per sequence with rather large baseline distances, which limits their usefulness (and the significance of obtained results) for visual tracking.

Both Moreels and Perona (2007) and Winder and Brown (2007) created datasets of three-dimensional objects that can be used for descriptor design and evaluations, but neither setup includes (or is feasible to do for) a moving camera, and absence of video sequences with typical artifacts such as jitter and motion blur limits their usefulness for tracking as well.

Baker et al. (2007) produced image sequences including moving objects that were designed to evaluate dense optical flow. However, due to both the length of the sequences (eight images per sequence) and the appearance of the scenes, they are not well-suited to evaluate inside-out tracking.

Zimmermann et al. (2009) collected image sequences of three different objects with approximately 12,000 images and made them available including ground truth. Lieberknecht et al. (2009) presented a dataset of 40 sequences featuring eight different textures in five different motion patterns each. This dataset might be the most similar to ours in terms of purpose and scope (although the ground truth is not made available). However, their motion patterns all combine several motions (much like our pattern "unconstrained," see Sect. 3.2) and are, effectively, rather similar to each other. Hence, as with Zimmermann et al. (2009)'s sequences, they do not allow a detailed analysis for different conditions or tailoring of the testbed to custom situations in which one wishes to emphasize performance on certain conditions and/or exclude conditions that are known not to occur: For example, in-plane rotation will be crucial in some applications, but largely irrelevant and thus sensible to exclude for visual odometry with a robot-mounted camera.

2.2 Applications of Visual Tracking

Visual Odometry The idea of using vision-based tracking for navigation of an autonomous robot can be traced back to Moravec (1980) and Matthies and Shafer (1987). To increase robustness, other information such as GPS data (Nistér et al. 2004) or coarse map information (Levin and Szeliski 2004) may be integrated. Cheng et al. (2006) described how visual odometry has been in use on NASA's Mars Exploration Rovers since 2004. This case is especially interesting due to the high level of autonomy and the limited computation power available on board the rovers.

While the systems mentioned above are feature-based, i.e., they keep track of a sparse set of "landmarks," visual odometry can also be accomplished using optical flow (Lee and Song 2004; McCarthy 2005). However, estimating movement from optical flow is prone to long-term drift, as the motion estimates and therefore the errors in the estimates are integrated over time, and is only feasible for smooth motion (DiVerdi and Höllerer 2008; Campbell et al. 2004).

Simultaneous Localization and Mapping SLAM was first explored using sensors such as laser range finders (Montemerlo et al. 2002, 2003). Pioneering work on visual SLAM, i.e. using (only) cameras as sensors, has been done by Davison et al. (2007). Further research explored the use of particle filters (Eade and Drummond 2006b), features at multiple scales (Chekhlov et al. 2006, 2007), and edge-based features (Eade and Drummond 2006a; Klein and Murray 2008), as well as recovery from tracking failure (Williams et al. 2007).

Augmented Reality Tracking for AR was first investigated using fiducial markers (Kato and Billinghurst 1999; Fiala 2005). Later systems used markerless environments, but require a priori information, for example a 3D point model of the scene (Skrypnyk and Lowe 2004), and/or additional inputs such as accelerometers (Bleser and Stricker 2008). Tracking of arbitrary, but a priori known targets has been demonstrated despite occlusion and even deformation (Lepetit and Fua 2006), for multiple targets (Park et al. 2008), and even on current generation mobile phones (Wagner et al. 2008, 2009). Abandoning markers, Lee and Höllerer (2008) used the user's outstreched hand to establish a coordinate frame for unprepared environments. Finally, Klein and Murray (2007) proved that real-time tracking for AR is possible without any prior information about the scene. With slight modifications and concessions regarding robustness, this system was shown to work on a mobile phone (Klein and Murray 2009).

Further applications include the online construction of panoramas (DiVerdi et al. 2008; Wagner et al. 2010).

2.3 Interest Point Detectors

Corner Detectors Corners are among the first low-level features used for image analysis and in particular, tracking (Moravec 1980). Based on Moravec's, Harris and Stephens (1988) developed the algorithm that became known as the Harris Corner Detector. They derive a "corner score" from the second-order moment image gradient matrix, which also forms the basis for the detectors proposed by Förstner (1994) and Shi and Tomasi (1994). Mikolajczyk and Schmid (2001) proposed an approach to make the Harris detector scale invariant. Other intensity-based corner detectors include the algorithms of Beaudet (1978), which uses the determinant of the Hessian matrix, and Kitchen and Rosenfeld (1982), which measures the change of direction in the local gradient field. A more exhaustive list especially of intensity- and contour-based detectors can be found in the evaluation of Schmid et al. (2000).

To avoid costly window or filter operations, Trajkovic and Hedley (1998) developed a detector that does not rely on discrete image derivatives. Instead, the pixel value at the center of a discretized circle is compared to the values on the circle. Rosten and Drummond (2005, 2006) developed this idea further and sped up the process by reducing the number of pixel tests with machine learning techniques.

Blob Detectors Instead of trying to detect corners, one may use local extrema of the responses of certain filters as interest points. In particular, many approaches aim at approximating the Laplacian of a Gaussian, which, given an appropriate normalization, was shown to be scale invariant if applied at multiple image scales (Lindeberg 1994). Lowe (1999, 2004) proposed to select the local extrema of an image filtered with differences of Gaussians, which are separable and hence faster to compute than the Laplacian. The Fast Hessian detector (Bay et al. 2008) is based on efficient-to-compute approximations to the Hessian matrix at different scales. Agrawal et al. (2008) proposed to approximate the Laplacian even further, down to bi-level octogons and boxes. Using slanted integral images, the result can be computed very efficiently despite a fine scale quantization. Ebrahimi and Mayol-Cuevas (2009) observed that the area underneath the filter kernel changes only very slightly from one pixel to the next and thus propose to further accelerate the process by skipping the computation of the filter response if the response for the previous pixel is very low. They report significant speed-ups with only minor repeatability losses. Their idea (which, in a simpler, unconditional fashion has also been implemented by Bay et al. 2008) seems promising and could similarly be implemented for other detectors as well.

Affine-Invariant Detectors In recent years, detectors have been proposed that are invariant to affine changes (Mikolajczyk and Schmid 2002; Schaffalitzky and Zisserman 2002; Tuytelaars and van Gool 2000; Matas et al. 2002; Kadir et al. 2004). Affine-invariant detectors provide higher repeatability for large affine distortions (Lowe 2004; Mikolajczyk and Schmid 2002), but are typically expensive to compute (Mikolajczyk et al. 2005; Moreels and Perona 2007).

2.4 Feature Descriptors

Early Approaches A variety of features derived from the local image intensities have been proposed to derive robust feature descriptors. Early ideas include derivatives for rotationally invariant features (Schmid and Mohr 1997), derivatives of Gaussians of different order (Freeman and Adelson 1991), filter banks derived from complex functions (Schaffalitzky and Zisserman 2002), phase information (Carneiro and Jepson 2003), and others. Many of these have been evaluated by Mikolajczyk and Schmid (2005).

SIFT and Follow-Up Works The Scale-Invariant Feature Transform (SIFT) by Lowe (1999, 2004) is probably the most well-known and widely used local descriptor, and has stimulated several follow-up works. It achieves invariance to changes in scale and rotation by operating in a local reference frame relative to a dominant scale and rotation that is computed from the image. The descriptor is based on local gradient histograms, sampled in a square grid around the keypoint. Based on the same local reference frame, Ke and Sukthankar (2004) applied principal component analysis (PCA) to the image gradients to derive a more compact representation. The "shape context" descriptor of Belongie et al. (2002) is based on an edge histogram, sampled from a log-polar grid. Combining above ideas, Mikolajczyk and Schmid (2005) proposed an extension to SIFT by using log-polar grids and applying PCA to reduce the dimensionality. Bay et al. (2008) proposed SURF as a faster alternative to SIFT, adopting similar approaches for scale and rotation invariance combined with efficient approximations to speed up the computation: The descriptor is derived from responses to box filters instead of the computationally more expensive Gaussian filters.

Learning a Descriptor Rather than testing different "*ad hoc*" approaches, Winder and Brown (2007) broke up the descriptor creation pipeline into several modules and used learning approaches to derive the optimal choice for each of them. Many of their building blocks are derived from or inspired by ideas mentioned above (Winder and Brown 2007, Winder et al. 2009).

Keypoint Recognition Using Trained Classifiers Lepetit and Fua (2006) proposed a new approach to robust keypoint recognition: they formulate keypoint matching as a classification problem using Randomized Trees as classifiers. Özuysal et al. (2007) simplified this approach structurally by adopting a naïve Bayes approach, thus simplifying the trees to "ferns." Taylor et al. (2009) presented another training-based keypoint recognition approach, which, during the training step, builds coarsely quantized histogram-based representations. Their approach requires only 44 bytes of memory per trained feature and allows for very fast matching using bitmasks. As a drawback, all of these classifiers need a training phase, although Calonder et al. (2008) showed that by abandoning the strict "classification" interpretation, the results may be re-interpreted and used in a classic descriptor paradigm as well.

2.5 Evaluations of Detectors and Descriptors

Schmid et al. (2000) compared interest point detectors on two still images under changes in rotation, viewpoint and illumination, as well as with artificially added image noise.

They found that the Harris Corner Detector outperformed other existing approaches at that time. Mikolajczyk and Schmid (2004) and Mikolajczyk et al. (2005) compared affine invariant detectors. Mikolajczyk and Schmid (2005) compared feature descriptors on sets of images. They found SIFT and their extension to SIFT (see above) to be superior to other approaches. Moreels and Perona (2007) explored the performance of combinations of detectors and descriptors with a testbed of three-dimensional objects rather than flat pictures.

While the objective of Winder and Brown (2007) and Winder et al. (2009) is to design a descriptor rather than evaluation of fundamentally different algorithms, their approach of learning the optimal descriptor design requires a similar framework and may be seen as evaluation of a family of descriptors with a certain parameterization.[1]

However, all comparisons mentioned above are geared towards object recognition and image retrieval rather than tracking. This becomes clear from the chosen testbeds, the performance measures chosen to evaluate the algorithms, and the set of detectors and descriptors that are tested. Execution time, a criterion crucial for designing real-time systems, receives very little or no attention, and reported execution times are in the order of seconds to several minutes, which is intractable for real-time tracking.

In contrast, the evaluation in this work aims at visual tracking in all of the factors mentioned above. Most notably, the performance measures are chosen with respect to the application of visual tracking and the testbed, which will be detailed in the next section, consists of video streams with several thousand frames affected by noise and motion blur rather than a set of high-resolution, low-noise still images.

3 Dataset Design

3.1 Establishing Ground Truth

To evaluate algorithms on images taken with a moving camera, ground truth information is needed, specifying which point x_j in frame j corresponds to point x_i in frame i. For general 3D scenes, this is very difficult to obtain without an accurate 3D model of the scene, but for planar scenes, x_i and x_j are related by a homography $H_{ij}(q) \in \Re^{3x3}$ (Hartley and Zisserman 2004; Schmid et al. 2000):

$$x_j = H_{ij}(q) \cdot x_i \tag{1}$$

Here, $x_{i/j}$ are in homogeneous coordinates: $x_i = (x, y, 1)^T$ and refer to coordinates in the undistorted frames, that is, after distortions introduced by the lens have been corrected for using the (known) internal camera calibration. Solving for H_{ij} may be done by projecting a known pattern onto a static scene (Schmid et al. 2000) or indicating reference points manually (Rosten and Drummond 2006; Zimmermann et al. 2009). Another option is to use fiducial markers that can be detected automatically (Fiala 2005). However, they occlude a significant part of the image (rendering it unavailable for the evaluation), require a minimum viewing angle for detection, and their detection uses algorithms similar to the ones to be evaluated, which potentially biases the result.

For this work, we fabricated a precisely milled acrylic glass frame which holds the planar texture and four bright red balls (15 mm diameter) as markers placed such that their center is in the plane of the texture. The markers were chosen as their detection is more robust to blur (no corners/edges have to be detected) and out-of-plane rotation (they look the same from any direction) than planar fiducial markers. With an initial estimate of their position, the whole textured area can be used to improve the accuracy of the estimation (see step 5 below). The markers are $13 \times 10.5''$ apart, situated outside of the area for the texture, which is $11 \times 8.5''$ (standard letter format). The area of the texture itself is $9.5 \times 7''$, of which a margin of $0.75''$ is subtracted to avoid border effects. This leaves an area of $8 \times 5.5''$ to be detected/tracked by the algorithms under evaluation. This setup can be seen in Figs. 1 and 3.

We then implemented the following semi-automatic algorithm to detect and track markers and texture in the videos:

1. The position and size of the balls are manually indicated in the first frame of the sequence.
2. An adaptive color model in HSV color space is initialized, which, applied to a new frame, produces a "probability map" that a given pixel belongs to the colored ball (Fig. 1 middle). The most probable positions of the balls are then identified using template matching with distance constraints.
3. The color model is adapted to the appearance of the balls in the new frame. For subsequent frames, a mixture model using both the model from the first and the previous frame is used to avoid long-term drift.
4. The position of each ball individually is refined using "inverse-compositional" image alignment (Baker and Matthews 2001) with 3 degrees of freedom (x, y, scale) between the current and the previous frame, and the homography between the current image and a canonical reference frame is computed (Fig. 1 right).
5. Finally, the homography is refined using image alignment between the reference frame and the current frame (Fig. 2).

[1]Note that their work concentrates on descriptors—the result of the interest point detector is already built-in into their dataset.

This algorithm was embedded into an interactive tool allowing the user to inspect the result and correct and re-start the algorithm if needed.

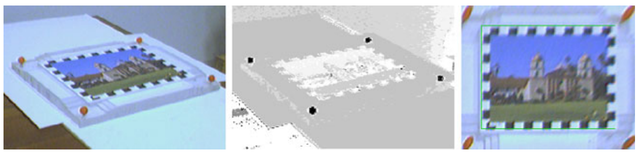


Fig. 1 Adaptive color model: The *image in the middle* shows the "probability map" that the adaptive color model generated for the image *on the left*. The image is then warped into a canonical frame in which the balls form a rectangle (*right image*)

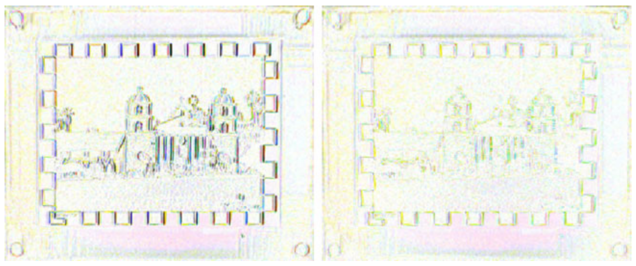


Fig. 2 Result of image alignment: Difference between current frame and reference frame before (*left*) and after image alignment (*right*). Images are shown inverted (i.e. white = image difference 0) and with increased contrast. The alignment was substantially improved. The residuals are due to change in appearance (lighting effects, motion blur), sensor noise and interpolation artifacts

Overall, this semi-automatic tracking system produced very stable warped video streams despite extreme viewing angles and motion blur with a manageable amount of manual labor. Examples of its output are depicted in Fig. 3.

3.2 Dataset

The testbed consists of 96 video streams, showing six different planar textures in 16 different motion patterns each, all recorded with a Unibrain Fire-i camera with a resolution of 640 × 480 pixels. The textures are shown in Fig. 4, and the motion patterns are as follows:

- *Unconstrained*: free movement of a hand-held camera, unconstrained except that the object of interest has to stay in the field of view. The motion is mostly smooth, some parts exhibit quick movements and motion blur. Figure 5(a) shows a reconstruction of one of the flight paths (6 × 500 frames).
- *Panning*: located about 1 m from the object of interest, the camera pans sideways, effectively causing the object to move sideways with very little distortion (6 × 50 frames).
- *Rotation*: located about 1 m from the object of interest, the camera rotates around its optical axis from 0° to 90°, resulting in in-plane rotation of the object (6 × 50 frames).
- *Perspective distortion*: starting roughly perpendicular above the object, the camera goes down in an arc, resulting in perspective distortion (out-of-plane rotation) of

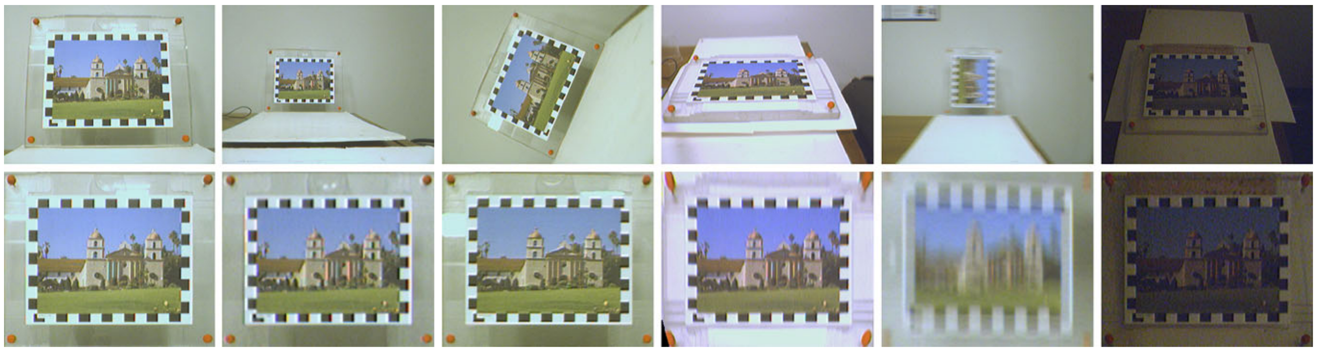


Fig. 3 *Top row*: a few examples of the 6889 frames in the testbed; *bottom row*: the same frames, warped to the reference frame. These examples illustrate the challenges that the dataset encompasses: scale changes (*first two images*), rotation, perspective distortion, motion blur (here: fastest setting), lighting (here: darkest condition). The black-and-white pattern on the border was added to improve the image alignment result (cf. Sect. 3.1), algorithms to be evaluated may use only the area inside

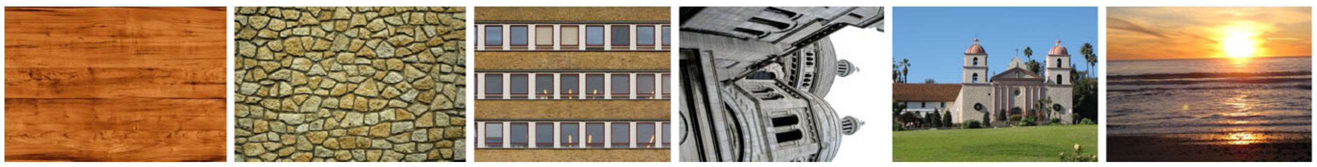


Fig. 4 Used textures. *From left to right*: "wood," "bricks," "building," "paris," "mission," "sunset"

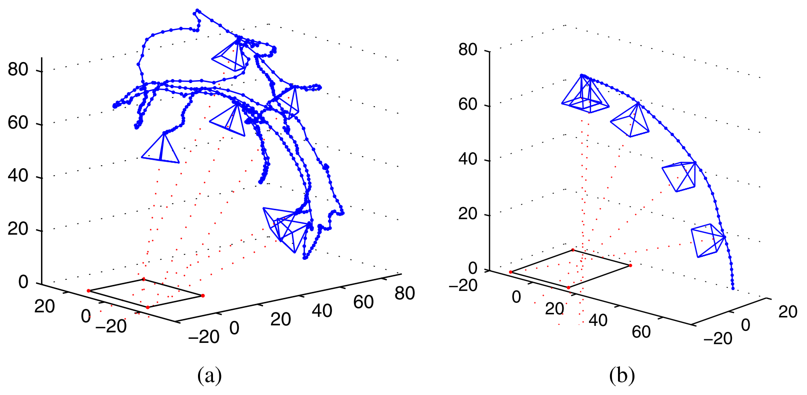


Fig. 5 Flight paths of selected video streams, all axes in centimeters. (**a**) Unconstrained (with texture "building"), (**b**) perspective distortion (with texture "bricks")

the object, cf. the flight path shown in Fig. 5(b) (6 × 50 frames).

– *Zoom*: the camera moves perpendicularly away from the object, from 60 cm to 130[±10] cm (6 × 50 frames).
– *Motion blur*: mounted to a pan-tilt unit to precisely control its speed, the camera pans sideways with nine different speed settings. The speeds are the 1- to 9-folds of 0.02778°/s, or, equivalently, 1- to 9-folds of about 5.1 pixels per frame (6 × 9 sequences of length 13–89, varying length due to the varying time it takes for the object to disappear).
– *Static lighting*: the camera is statically mounted on a tripod and observes the scene under four different lighting conditions. The transition from one condition to the next is *not* included (6 × 4 × 20 frames).
– *Dynamic lighting*: the camera is statically mounted on a tripod and observes the scene transitioning from bright lighting to dark (a screen being moved in front of a soft lamp) and back (6 × 100 frames).

The motion patterns "panning" through "zoom" were conducted with the camera mounted to an appropriately mechanically guided, but manually operated contraption, hence they are not exactly the same among the different textures and contain certain amounts of motion blur and jitter. As these conditions are exactly the same for all algorithms and we desire robustness against all kinds of motions, this does not affect algorithm comparison. All videos are encoded with the lossless HUFFYUV codec. In total, the dataset consists of 6889 frames.

The camera movement is reconstructed from the position of the target texture for the purposes of illustration (Fig. 5) and binning algorithm performance according to the relative change in camera positions.

To evaluate algorithms, the sequences can be used in consecutive order, thus simulating continuous tracking during smooth motion, as well as in randomly (or otherwise) sampled order, thus evaluating robustness against larger baseline distances.

3.3 Distribution

Our dataset is publicly available, including all necessary material. In particular, we are making available:[2]

– the 96 video sequences themselves,
– the camera calibration needed to correctly undistort the frames,
– the computed ground truth for each frame as well as the reconstructed camera paths,
– frame indices indicating proposed frame pairs for evaluating tracking in the case of larger baseline distances along with the indices of the "bins" into which each frame pair falls according to the relative camera pose between the respective two frames.[3]

The next section provides an introduction to detector-descriptor-based tracking, including a compilation of existing systems, and briefly reviews the algorithms that are included in the evaluation. Section 5 will then describe our evaluation setup in detail.

4 Detector-Descriptor-Based Visual Tracking

In recent years, many visual tracking systems have been proposed. They differ in motivation, aim, implementation and algorithms that are used. However, they can generally be broken down into similar main components and largely follow a structure as depicted in Fig. 6. In Sect. 4.1, we will show how state-of-the-art systems fit into this structure.

One cycle of a visual tracking algorithm can be summarized as follows: With a new frame captured from the video source, an interest point detector is applied to detect candidate points for tracking. For each of the candidates, a feature descriptor is computed. This descriptor is then sought

[2]http://ilab.cs.ucsb.edu/tracking_dataset_ijcv/.

[3]The significance of this will become clear in Sect. 5. Use of these indices is optional but needed to directly compare with the results presented in Sect. 6.

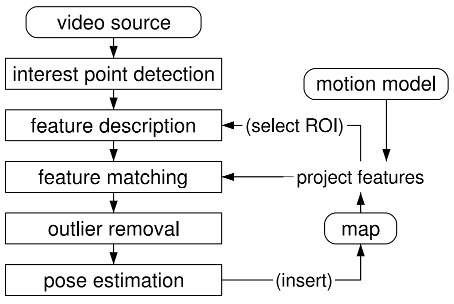


Fig. 6 Structure of a detector-descriptor-based visual tracking system

to match with the descriptors of previously (e.g., in the last frame) encountered features. Usually, the matching is constrained to those of the known features that are predicted to lie close to the encountered position ("active search" Davison et al. 2007 or "gated search" Klein and Murray 2007). Here, the interest point detector is used to thin out the field of potential candidates. However, if the cost of creating the description per pixel is very small, the interest point detection can be omitted and the matching can instead be performed over all pixels inside the predicted region (Davison et al. 2007; Eade and Drummond 2006b).

Usually, the number of feature matches is much greater than the degrees of freedom that have to be estimated. Thus, the system is highly overdetermined, which allows for the removal of outliers before the pose is estimated.

4.1 Existing Visual Tracking Systems

Table 1 lists existing feature-based visual tracking systems along with the algorithms that are used in each of them for the main components depicted in Fig. 6. This compilation is not meant to be exhaustive, and the short bullet points do not do justice to specific features and contributions of the listed systems. Rather, it is meant to give an overview of the applications of visual tracking and the algorithms that have been employed for different components.

As seen from the compilation in Table 1, different interest point detectors and feature descriptors have been used in visual tracking systems. Wherever explicit timings are available (e.g. Carrera et al. 2007; DiVerdi et al. 2008; Klein and Murray 2007; Lee and Höllerer 2008; Wagner et al. 2008), they indicate that a significant part of the overall processing time is spent on feature detection, description and matching. These observations and the lack of independent comparisons in the context of real-time visual tracking motivate the evaluations in this work.

4.2 Interest Point Detectors

Due to the high dimensionality of image data, tracking every single pixel is computationally prohibitive and incorporates a lot of redundancy, as pixels do not move independently. Instead, a sparse set of features is extracted from the image. Although work has been done to detect and integrate other features like edges (Eade and Drummond 2006a; Klein and Murray 2008), features that build upon or around single points are most commonly used for tracking.

There is no clear-cut definition as to what makes a point "interesting," although attempts have been made, for example, via the entropy of the local neighborhood (Schmid et al. 2000), and detection of such points is only an intermediate step in any application. The most pragmatic definition is "the right features are exactly those that make the tracker work best" (Shi and Tomasi 1994). This means that any set of points is acceptable, but the results ought to be consistent, i.e.: in images that show the same scene, the algorithm should detect the same points. This is especially relevant for visual tracking and leads to the criterion of repeatability, which will be defined more formally in Sect. 5.1.

The following sections review the detectors which are included in the evaluation in this work.

4.2.1 *Harris Corner Detector*

Based on Moravec's corner detector (Moravec 1980), Harris and Stephens (1988) developed the following algorithm: Given an image I, the algorithm first computes the following matrix for every pixel (x, y), which is an approximation to the local auto-correlation function of image I:

$$M(x,y) = \begin{bmatrix} \sum_{u,v} w_{u,v}\cdot[I_x(x_r,y_r)]^2 & \sum_{u,v} w_{u,v}\cdot I_x(x_r,y_r)I_y(x_r,y_r) \\ \sum_{u,v} w_{u,v}\cdot I_x(x_r,y_r)I_y(x_r,y_r) & \sum_{u,v} w_{u,v}\cdot[I_y(x_r,y_r)]^2 \end{bmatrix} \quad (2)$$

I_x and I_y denote the derivatives of image I, $(x_r, y_r) := (x+u, y+v)$, and $w(u, v)$ is a window and weighting function. In the simplest case, $w(u, v)$ can be a binary rectangular window. Harris and Stephens (1988) propose to use a Gaussian window $w(u, v) = \exp\{-(u^2 + v^2)/2\sigma^2\}$.

The eigenvalues λ_1, λ_2 of M are proportional to the principal curvatures of I. Based on them, the region can be classified as either uniform (λ_1 and λ_2 small), edge region (one small, one large) or corner region (both large). To avoid the explicit computation of λ_1 and λ_2, Harris and Stephens (1988) propose the following corner score, which is derived based on the eigenvalues, but can be expressed without them:

$$\begin{aligned} c(x,y) &= \lambda_1\lambda_2 - k\cdot(\lambda_1+\lambda_2)^2 \\ &= \det(M(x,y)) - k\cdot[\operatorname{trace}(M(x,y))]^2 \end{aligned} \quad (3)$$

Table 1 Existing feature-based visual tracking systems. This compilation is not exhaustive

Reference	Objective/prior knowledge or additional inputs[a]	Detector[b]	Descriptor[b]	Matching, outlier removal, pose estimation[c]
Bleser and Stricker (2008)	tracking/3D model, IMU	FAST	patch, warped	SSD, Kalman filter
Carrera et al. (2007)	tracking/known target	Harris	SURF	UKF
Chekhlov et al. (2007)	SLAM/–	Shi-Tomasi	SIFT-like	UKF
Cheng et al. (2006)	odom./stereo, wheel odom., IMU	Harris or sim.	patch	NCC, RANSAC, LSE
Davison et al. (2007)	SLAM/initialization target	Shi-Tomasi[d]	patch (11×11), warped	NCC, EKF
DiVerdi et al. (2008)	panorama creation/–	Shi-Tomasi	Optical flow & SURF[e]	RANSAC (Horn 1987)
Eade and Drummond (2006b)	SLAM/–	FAST[d]	patch, warped	NCC, particle filter
Klein and Murray (2007)	SLAM/–	FAST	patch (8×8), warped	SSD, Tukey M-estimator
Lee and Höllerer (2008)	tracking/init. with user's hand	DoG	Optical flow & SIFT[e]	RANSAC, Kalman filter
Lepetit and Fua (2006)	tracking-by-det./known target	cf. reference	Randomized Trees	RANSAC, LSE, P-n-P
Nistér et al. (2004)	odometry/–	Harris	patch (11×11)	NCC, RANSAC
Özuysal et al. (2007)	tracking-by-det./known target	cf. reference	Ferns	RANSAC
Park et al. (2008)	tracking-by-det./known targets	not specified	Ferns	RANSAC, P-n-P
Se et al. (2002)	SLAM/trinocular camera	DoG	[scale, orientation]	LSE, Kalman filter
Skrypnyk and Lowe (2004)	tracking/known scene	DoG	SIFT	RANSAC, non-lin. LSE
Taylor et al. (2009)	tracking-by-det./known target	FAST	trained histograms	PROSAC
Wagner et al. (2009)	tracking/known targets	FAST[d]	patch & reduced SIFT[e]	NCC, PROSAC, M-estim.
Wagner et al. (2010)	panorama creation/–	FAST[d]	patch (8×8), warped	NCC, M-estimator
Williams et al. (2007)	recovery for SLAM/–	FAST	Randomized lists	JCBB, EKF

[a]Unless specified otherwise, the system aims to track or recover the full 6 degree-of-freedom pose and uses a single camera as the only sensor. Abbreviations: IMU = inertial measurement unit, SLAM = simultaneous localization and mapping

[b]Abbreviations used in these columns will be explained in Sects. 4.2 and 4.3, respectively

[c]Abbreviations: EKF = Extended Kalman Filter, JCBB = Joint Compatibility Branch-and-Bound (Neira and Tardos 2001), LSE = Least-squares estimation, NCC = normalized cross-correlation, PROSAC = progressive sample consensus (Chum and Matas 2005), P-n-P cf. (Moreno-Noguer et al. 2007), RANSAC = random sample consensus (Fischler and Bolles 1981), SSD = sum of squared distances, UKF = Unscented Kalman Filter (Julier and Uhlmann 1997)

[d]Detector is only used to discover new features. Matching is performed against all pixels around the predicted feature position

[e]Hybrid approach: patch/Lucas-Kanade optical flow for frame-to-frame tracking, SIFT/SURF for long-term matching/detection

Subsequently, 8-neighborhood non-maximum suppression is applied and candidates with a response of less than a predefined percentage θ of the maximum response encountered, $c(x, y) < \theta \cdot \max_{x,y}\{c(x, y)\}$, are filtered out.

4.2.2 *Shi-Tomasi's "Good Features to Track"*

Based on a theoretical analysis of which features will be "good to track," Shi and Tomasi (1994) derive an image motion model for affine motion and pure translation, which they use for tracking and monitoring the tracked features. For tracking, they suggest using the translation model, where the matrix involved is equivalent to M (2). With the same reasoning as above, the eigenvalues λ_1, λ_2 of M are computed and a candidate point is accepted if

$$c(x, y) = \min(\lambda_1, \lambda_2) > \theta \cdot \max_{x,y}\{c(x, y)\} \qquad (4)$$

Compared to the Harris score (3) this requires an additional square root operation per pixel.

4.2.3 *Difference of Gaussians (DoG)*

The approach of detecting local extrema of the image filtered with differences of Gaussians (DoG) was introduced by Lowe (1999, 2004) as part of SIFT. The descriptor will be reviewed in Sect. 4.3.2.

To achieve invariance against changes in scale, the detector builds a pyramid of images by convoluting the image I with differences of Gaussian filters at different scales σ:

$$\begin{aligned} DoG_{k,\sigma}(x, y) &= G(x, y, k\sigma) - G(x, y, \sigma) \\ &= \frac{1}{2\pi(k\sigma)^2} \cdot e^{-(x^2+y^2)/2(k\sigma)^2} \\ &\quad - \frac{1}{2\pi\sigma^2} \cdot e^{-(x^2+y^2)/2\sigma^2} \end{aligned} \qquad (5)$$

In practice, this is done by first convoluting with the Gaussian kernels $G(\sigma)$ and then computing differences of the resulting images:

$$I_0 = I * DoG_{k,\sigma_0} = I * G(k\sigma_0) - I * G(\sigma_0)$$

$$I_1 = I * DoG_{k,k\sigma_0} = I * G(k^2\sigma_0) - I * G(k\sigma_0)$$

$$\vdots$$

As interest points, the algorithm selects local extrema, which are found by comparing each sample to its eight neighbors in the current image I_n and the 18 neighbors "above" (in I_{n-1}) and "below" (in I_{n+1}). The interest point locations are then refined to subpixel accuracy by fitting a parabola to the sample point and its immediate neighbors (Brown and Lowe 2002). Interest points with low contrast, i.e., $|I_{\hat{\sigma}}(\hat{x})| < \theta_{\text{contr}}$, where $\hat{x}$ and $\hat{\sigma}$ are the refined extremum location and scale, are rejected. The ratio of the principal curvatures are estimated using the eigenvalue approach from Harris and Stephens (1988) (cf. Sect. 4.2.1), and feature points with an "edge response," i.e., where the ratio of the two principal curvatures is smaller than a threshold θ_{edge}, are rejected as well.

4.2.4 *Fast Hessian*

The Fast Hessian detector, proposed by Bay et al. (2008) as a basis for SURF, is based on the determinant of the Hessian matrix, which at scale σ is defined as follows:

$$H(x,y,\sigma) = \begin{bmatrix} \frac{\partial^2}{\partial x^2} G(\sigma) * I(x,y) & \frac{\partial}{\partial x}\frac{\partial}{\partial y} G(\sigma) * I(x,y) \\ \frac{\partial}{\partial x}\frac{\partial}{\partial y} G(\sigma) * I(x,y) & \frac{\partial^2}{\partial y^2} G(\sigma) * I(x,y) \end{bmatrix} \quad (6)$$

As convolution with the Gaussian second order derivatives is very costly especially for higher scales, Bay et al. approximate them by filters that are composed of simple box filters (Fig. 7 left) and can therefore be computed in constant time using the integral image (Viola and Jones 2001). The computed candidate score then is

$$c(x,y,\sigma) = D_{xx}(\sigma) \cdot D_{yy}(\sigma) - (0.9 D_{xy}(\sigma))^2 \approx \det[H(x,y,\sigma)] \quad (7)$$

where D_{xx}, D_{xy} and D_{yy} are the results of convoluting the image with the filters depicted in Fig. 7 (left), and the factor 0.9 is added to approximate $\det[H(x,y,\sigma)]$ more closely. $3 \times 3 \times 3$-neighborhood non-maximum suppression and subpixel refinement are then applied as for the DoG detector. Likewise, candidates with c below a threshold θ are rejected. To speed up the computation, one may optionally increase the sampling intervals, i.e., compute c only for every nth pixel (Bay et al. 2008).

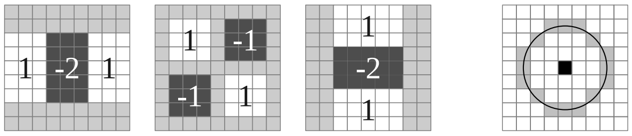


Fig. 7 *Left*: Filters composed of box filters as used by Fast Hessian as approximations to second order derivatives of Gaussians. Weights of *black* and *white regions* as denoted, *grey regions* have weight zero. Figure adapted from Bay et al. (2008). *Right*: Bresenham circle. The *black point* is the current candidate point p, the 16 *grey points* are the discretized approximation of the outlined circle around it. Figure adapted from Rosten and Drummond (2006)

4.2.5 *Features from Accelerated Segment Test (FAST)*

Rosten and Drummond (2005, 2006) developed a high-speed corner detector which they coined FAST, for Features from Accelerated Segment Test. The algorithm operates on a discretized circle around a candidate point p as shown in Fig. 7 (right). p is classified as a corner if there exists a contiguous arc of at least nine pixels that are all brighter or all darker than p by a threshold t. The algorithm was further accelerated by training a decision tree to test as few pixels as possible for classifying a candidate pixel as corner or non-corner. With this decision tree, only 2.26 pixels are tested for each candidate on average, whereas with the naïve algorithm, 2.8 are tested (Rosten and Drummond 2006).

In contrast to all aforementioned detectors, detection with the FAST algorithm does not inherently provide a measure of the "strength" of a feature. In order to apply non-maximum suppression, the following score is computed for each candidate point:

$$c(p) = \max\left\{ \sum_{q \in S_+} |I_q - I_p| - t,\ \sum_{q \in S_-} |I_q - I_p| - t \right\} \quad (8)$$

where S_+ is the subset of pixels on the circle that are brighter than p (by t) and S_- the subset of pixels that are darker than p (by t) (Rosten and Drummond 2006).

4.2.6 *Center-Surround Extrema (CenSurE)*

Like Lowe (2004)'s DoGs, the filters designed by Agrawal et al. (2008) aim at approximating a Laplacian of a Gaussian filter, though simplified further: In the first step, the filter is reduced to a bi-level filter, i.e., with filter values -1 and 1. Approximating the Laplacian of a Gaussian then yields a torus-shaped filter kernel as depicted in Fig. 8 (left). As this filter is computationally rather expensive, three approximations are proposed, each getting less symmetric, but easier to compute (Fig. 8 right).

As mentioned above, box filters can be efficiently computed at any scale using an integral image. For octagons and hexagons, Agrawal et al. (2008) propose the use of *slanted*

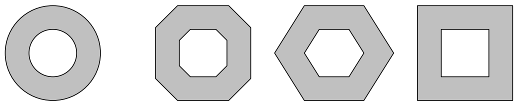


Fig. 8 CenSurE's bi-level filters. The circle (*left*) is the ideal, fully symmetric bi-level approximation of the Laplacian. *From left to right*, the approximations are coarser (less symmetric), but easier to compute: octagon, hexagon, square. Figure adapted from Agrawal et al. (2008)

integral images, where the value stored at each pixel (x, y) is the sum of intensities in the trapezoidal area above (x, y). Octagons and hexagons can then be decomposed into a few trapezoids and thus also be efficiently computed at any scale.

$3 \times 3 \times 3$-neighborhood non-maximum and edge response suppression are applied in a similar fashion as in the DoG and Fast Hessian detectors. Agrawal et al. (2008) conclude that the octagon filter represents the best trade-off between speed and repeatability.

4.3 Feature Descriptors

After "interesting points" have been identified, a description has to be found that can be used to identify and match them across images. Ideally, this description is unique to every world point X, but identical for all views $x_i = P_i X$ of X. This ideal case is only possible in extremely simplified environments, but proposed descriptors aim to capture the texture of the local neighborhood while being invariant to changes in illumination, scale and rotation.

As for the interest point detectors, the following sections review the descriptors which are included in the evaluation.

4.3.1 Image Patch

The most straightforward description of a feature point is the image patch around the point itself. Computation of this "descriptor" only requires (sub)sampling the image at a given location. It does not have any of the desired invariance properties described above, although invariance to uniform illumination changes can be achieved by normalizing the patch's intensity values. Tracking systems that use image patches for matching either assume that the viewpoint (relative position and orientation of camera and patch), and therefore the patch appearance, has changed very little since it was last observed (Cheng et al. 2006; Nistér et al. 2004), or use an estimate of the current camera position to warp the last appearance of the patch to what it would look like if viewed from the estimated camera position (Davison et al. 2007; Eade and Drummond 2006b; Klein and Murray 2007).

4.3.2 Scale Invariant Feature Transform (SIFT)

For each keypoint p, the SIFT algorithm (Lowe 1999; Lowe 2004) first assigns an orientation α_p in order to make the descriptor invariant to image rotation: The gradient magnitude m and orientation α are computed for each pixel around p, and a histogram of these orientations, weighted by m and a Gaussian window around p, is computed. α_p is set to the highest peak in the histogram.

The scale σ_p and orientation α_p of keypoint p now define a local coordinate system in which the subsequent steps operate. A new orientation histogram (with all orientations now relative to α_p) with B bins is computed for each subregion of a $N \times N$ grid around p, again weighted by the respective m and a Gaussian window around p. The descriptor finally consists of the $N \cdot N \cdot B$ histogram values. Default values for N and B are 4 and 8, respectively, yielding a descriptor of length 128, but this work will evaluate other choices as well.

4.3.3 Speeded Up Robust Features (SURF)

Similar to SIFT, SURF (Bay et al. 2008) first assigns an orientation to each keypoint: A circular region around the keypoint is convoluted with two Haar wavelets. The size of region and wavelets as well as the sampling step are dependent on the scale σ at which the keypoint was detected. The filter responses, weighted with a Gaussian around the keypoint, are then represented as vectors in a two dimensional space and summed up with a rotating angular window. The longest resulting vector determines the orientation of the keypoint.

As for SIFT, scale and orientation now define a local coordinate system for the computation of the descriptor. A square region around the keypoint is split up into 4×4 subregions and the following feature vector is computed:

$$\left[\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right] \tag{9}$$

d_x and d_y are the filter responses to the Haar wavelets and all sums are over regularly spaced sample points in the respective subregion. Concatenating the feature vectors of each of the subregions yields the SURF descriptor of length $4 \cdot 4 \cdot 4 = 64$ (Bay et al. 2008).

4.3.4 Keypoint Classification with Randomized Trees

Lepetit and Fua (2006) formulate recognition of keypoints as a classification problem using Randomized Trees as trained classifiers. In particular, the set of all possible appearances of an image patch i is considered one class C_i. While creating a "forest" of Randomized Trees, each node n gets assigned two randomly chosen pixel displacements p_n, q_n. For a candidate keypoint x, the node tests if the pixel at $x + p_n$ is darker than the pixel at $x + q_n$. If so, the keypoint is referred to the left child, otherwise to the right child, until it finally reaches a leaf.

During training, a set of views is generated for each keypoint ("class"), depicting the changes that the classifier shall be robust against (e.g., affine warps and Gaussian noise). Each sample is "dropped down" each of the trees, eventually reaching one of the leaves. Afterwards, a posterior probability for each class C_i and leaf l_j can be extracted, denoting the probability that a sample of C_i reaches leaf l_j, and, with Bayes' theorem, the probability that a sample that reaches leaf l_j belongs to class C_i (Lepetit and Fua 2006).

Using trained classifiers has a few conceptual advantages over descriptor-based matching:

- For recognition, a keypoint is simply "dropped down" the classification trees, which is faster than computing a complex descriptor (such as SIFT or SURF) plus matching against a potentially large database.
- The classifiers can be trained to be robust against exactly the conditions that are expected to occur, including random background if looking for an isolated object.
- During the training step, the algorithm can select the keypoints that are the most stable under the presented viewing conditions.

The drawbacks are the requirement of a training step, but also the limited scalability: as discussed further in Sect. 6.2.1, the number of keypoints is inherently limited.

4.3.5 Keypoint Classification with Ferns

The Randomized Trees of Lepetit and Fua (2006) represent the full joint probabilities for each possible combination of node test results (thus branching out to 2^n leaves for n node tests). To compress this representation, one may employ a naïve Bayesian approach and assume that the node tests are independent of each other (the "tree" then collapses into a single "string" of nodes). To avoid either extreme, Özuysal et al. (2007) follow a Semi-Naïve Bayesian approach and model only some of the dependencies by partitioning the node tests into groups: within each group, the joint probabilities are fully modelled, while each group is assumed to be independent from the other groups. Thus, the classifier trees now structurally look like "ferns." Training and classification are conducted as for Randomized Trees.

5 Setup for Evaluating Detectors and Descriptors for Tracking

5.1 Evaluation of Interest Point Detectors

Tested Interest Point Detectors The interest point detectors that were evaluated are Harris (Harris and Stephens 1988), Shi-Tomasi (Shi and Tomasi 1994), DoG (Lowe 1999), Fast Hessian (Bay et al. 2008), FAST (Rosten and Drummond 2006), and CenSurE (Agrawal et al. 2008). As discussed in Sect. 2, many other detectors have been proposed in the literature. The reasons for the selection above are as follows: these detectors are considered to be state-of-the-art and have been widely used (Harris, Shi-Tomasi, SIFT), they were previously shown to outperform other detectors (Schmid et al. 2000; Rosten and Drummond 2006; Bay et al. 2008; Agrawal et al. 2008), and they are used in state-of-the-art visual tracking systems (cf. Table 1).

Performance Measures As motivated in Sect. 4.2, the criterion that is most relevant for visual tracking as well as for other domains (Bay et al. 2008) is repeatability (Schmid et al. 2000):

$$\text{repeatability} = \frac{|\{(x_a \in S_i, x_b \in S_j)| \|H_i \cdot x_a - H_j \cdot x_b\| < \varepsilon\}|}{|S_i|} \qquad (10)$$

where S_i, S_j are the sets of points detected in frames i and j, respectively, and H_k is the homography between the kth frame and the reference frame. Even with perfect ground truth, a re-detected point will not be at exactly the same position as in the old frame: This is most obvious in the case of detectors that do not work with subpixel refinement, where the detected point has to "jump" to the next pixel at some point during object or camera movement (a more detailed list of reasons may be found in Rosten and Drummond 2006). For our evaluations, ε is set to 2.

It should be noted that Schmid et al. (2000) define the denominator of (10) as $\min\{|S_i|, |S_j|\}$. However, this leads to a seemingly perfect repeatability of 1 if the detector returns, for example, 100 points in the first frame and only one point in the second frame, as long as this single point is among the 100 detected earlier, which is misleading. Our choice of denominator comes at the price of a non-symmetric performance measure, but ensures that it reflects the way that a tracking system operates: it tracks *from* one frame *to* the next, hence a non-symmetric performance measure seems acceptable.

Unfortunately, the repeatability criterion is biased in favor of algorithms that return many keypoints: as a trivial example, an algorithm that simply "detects" every single pixel has a repeatability of 1. Alternative measures have been proposed (Mohanna and Mokhtarian 2006), but they rely on subjective decisions and are not relevant for the application of visual tracking. One possibility is to adjust the parameters for the comparison so that different detectors return the same number of points (see, e.g., Bay et al. 2008), but this assumes that all detectors work equally well for any number of features, which is not the case as Fig. 10(a) will show, and the difference in performance outweighs the criterion's bias. Additionally, the number of features that a detector works well with might be an important factor in the decision of which detector to use in a specific application, so instead of trying to equalize this value, we will report it together with the repeatability.

Moreover, the repeatability criterion as given in (10) allows arbitrarily small sets of points: A single, perfectly stable point will produce a repeatability of 1. However, if we want to use the detector for a tracking system that estimates a pose with d degrees of freedom, we need at least $d/2$ correctly re-detected (and identified) points. In our setup, we require 4 points to correctly obtain the homography. Therefore, we amend (10) and set the score to 0 if the number of re-detected points is smaller than 4. This is especially important during the first stage of our evaluations, in which we determine the parameter configuration to be used: In preliminary evaluations, we observed that DoG and Fast Hessian obtained high repeatability scores for very high thresholds, resulting in very few features. Choosing the parameters such that repeatability as given in (10) is maximized thus frequently led to less than 4 re-detected points. The above correction alleviates that problem and ensures that maximizing repeatability also maximizes the chance of tracking success.

5.2 Evaluation of Feature Descriptors

Tested Feature Descriptors The feature descriptors that were evaluated are image patch, SIFT (Lowe 1999), SURF (Bay et al. 2008), classification via Randomized Trees (Lepetit and Fua 2006), and classification via Ferns (Özuysal et al. 2007). The reasons for this selection are analogous to the arguments for choosing the detectors as listed in Sect. 5.1: The chosen descriptors are widely used, were shown to outperform others and/or perform comparably (Mikolajczyk and Schmid 2005; Bay et al. 2008), and are used in state-of-the-art visual tracking systems (cf. Table 1).

The keypoint classifiers were given the first frame of the "unconstrained" motion pattern of the respective texture (along with the correct warp) for training. The same detector was used during both training and evaluation. As for the descriptors, several parameter configurations were evaluated (including parameters regarding the training step).

Matching Paradigm for Descriptors To simulate a tracking system with pose estimation affected by uncertainty, we match each descriptor against all descriptors from the previous frame that fall within a circular region of interest around the current descriptor's location. We set this radius to 50 pixels due to the consideration that, if simulating a frame-to-frame tracker, we assume that a pixel can hardly move farther in a single frame, as otherwise motion blur will likely prohibit successful tracking (cf. repeatability results on motion blur, Fig. 10(h)). This value is of course application-specific and, in this evaluation, arbitrary to a certain extent, but the size of this region relates to the uncertainty inherent to the system and is thus independent of the specific detector/descriptor used.[4] Its influence on the descriptors' performance will be evaluated (and shown to be limited) in Figs. 13(a) and 16.

The first nearest neighbor (1NN), i.e., the descriptor with the smallest distance in descriptor space, is identified and established as its match.

Performance Measures The performance criterion used is

$$\text{precision of 1NN} = \frac{\text{number of correct matches}}{\text{number of correct} + \text{false matches}} \qquad (11)$$

Other criteria can be used, for example recall (ratio of retrieved matches to number of relevant candidates within a certain database) or precision of the first k nearest neighbors. Whether or not a criterion is relevant depends on the matching strategy that is used (Mikolajczyk and Schmid 2005): as most real-time tracking systems (Sect. 4.1) evaluate only the 1NN, the precision of the 1NN is the most relevant criterion in this context. The classification provided by the classifiers may be interpreted as "one-dimensional descriptor vector," and the classification precision (more accurately, the "best guess" classification precision) is exactly equivalent to (11).

To evaluate the performance of the descriptor independent of the detectors' repeatability, we simulate the ideal detector by detecting points in the first frame and then re-projecting them into all subsequent frames using the ground truth warp (Calonder et al. 2008). Hence, each point does indeed have a unique correct match. This artificial constraint will be removed in the next stage of the evaluation.

Descriptor vs. Classifier Paradigm It is important to note that there are crucial differences in the problem as posed for the descriptors (patch, SIFT, SURF) compared to the trained classifiers (Randomized Trees and Ferns): By using classifiers, one inherently employs tracking-by-detection, or *model-based* tracking (the model being the image(s) that were used to train the classifiers). While descriptors may be used in a model-based fashion as well (by computing descriptors for a reference image and then matching against those instead of matching against the previous frame), this is not advisable in the general case, as the descriptors have to be computed in every frame either way and this merely throws away the advantage of short baseline distances.

This difference has important implications for the interpretation of our results:

[4] To reduce it, one has to improve the pose prediction, increase the framerate, and/or restrict the allowed movements, which are all external factors from the point of view of this evaluation.

– While for the descriptors, one needs the result of two frames to evaluate precision, the classifiers always "match" against the trained model, i.e., one "new" frame is sufficient for evaluations, and the only sensible measure of baseline distance is to the viewpoint of the training frame. It is therefore important to note that, in all figures in which results are grouped by the relative position or orientation of two frames, these results refer to *different pairs of frames* for the descriptors as compared to the classifiers.
– While it is straightforward to constrain the region that a descriptor vector is matched against (and given the task of tracking reasonable to do so), this is less straightforward for the classifiers, which will simply classify each point according to their model. This means that the number of candidates is inherently larger for the classifiers. Note however that Fig. 13(a) will show that the influence of this factor is limited.
– Lastly, given "simple" circumstances, it is possible that all keypoints can be matched correctly using descriptor vectors (i.e., precision = 1), whereas the number of keypoints that can be recognized correctly using a classifier is inherently limited to the number of keypoints that the model was trained to recognize (here: 600, which was determined to be a good parameter in Fig. 12(a)). If more points are detected in a particular frame, the precision will necessarily drop, even with a "perfect" classifier.

These differences have to be kept in mind when interpreting, e.g., Fig. 13.

5.3 Evaluation of Detector-Descriptor-Based Tracking

As mentioned before, neither feature detection nor description are isolated tasks. Instead, our goal is to reliably track an object of interest, or, equivalently, estimate the (relative) pose of scene and observer. In our setup, this is equivalent to estimating the homographic warp between the frames. Although we argued for why the detector's repeatability and the descriptor's precision are the most important performance measures for the two individual stages, we want to evaluate the actual success rate for tracking for any detector-descriptor combination.

Tested Solutions We tested each possible combination of the above detectors and descriptors. Additionally, we employed RANSAC (Fischler and Bolles 1981), which is frequently used (cf. Table 1) and well-suited for this task, as an outlier removal step (cf. Fig. 6), and computed the homographic warp from the largest set of inliers found after 200 iterations. Note that our focus remains on evaluation of the (combination of) detector and descriptor; the use of RANSAC as well as specific parameters are arbitrary albeit reasonable choices for a possible application scenario.

As before, descriptors are sought to match against descriptors from the previous frame in a circular region of interest of 50 pixel radius to simulate a tracking system with pose estimation affected by uncertainty.

Performance Measures As a performance metric we measure the distance between the true (ground truth) position of the red markers and their projected position using the estimated homography. We define a frame to be successfully tracked if the average error between the estimated and the true positions of the ground truth markers is less than five pixels. This yields a higher-level metric than the ones employed above, in particular, it yields one value per frame instead of one per keypoint.

5.4 Testbed

We used the dataset described in Sect. 3. As briefly mentioned above, there are two different use cases: To simulate continuous tracking, the performance measures (repeatability, precision, tracking success) are evaluated on consecutive frames. To evaluate robustness against larger baseline distances, we randomly chose $10n$ frame pairs of each sequence, where n is the length of the sequence, and evaluated for these frame pairs (note that this does not make a difference for the classifiers). The pairs were binned by the relative pose change between the two frames (e.g., angle of rotation for in-plane rotation), and the obtained performance results were averaged for each bin. In the context of tracking, this is relevant for tracking-by-detection, recovery after tracking failure over a few frames, or re-visiting a previously mapped scene (closing of a loop). If both conditions are shown in the same graph, the latter will be indicated by dashed lines.

5.5 Implementation

The software framework for this evaluation was implemented in C++. For the evaluated algorithms, the implementations of the original authors were used where available (Fast Hessian+SURF,[5] FAST,[6] classification with Randomized Trees,[7] classification with Ferns).[8] Implementations of Harris and Shi-Tomasi are provided with the OpenCV library.[9] The Harris implementation by Edward Rosten, which was used in the comparison in Rosten and Drummond (2006) and which he kindly made available to us, was found to

[5]http://www.vision.ee.ethz.ch/~surf/.
[6]http://svr-www.eng.cam.ac.uk/~er258/work/fast.html.
[7]http://cvlab.epfl.ch/software/bazar/index.php.
[8]http://cvlab.epfl.ch/software/ferns/index.php.
[9]http://sourceforge.net/projects/opencvlibrary/—note that taking the library functions as-is does not provide control over the windowing kernel.

be equivalent (i.e., detect the same set of points), which we take as an indication that both implementations are correct and implement exactly the original algorithm. For the comparison, OpenCV's implementation was used, as it was slightly faster. The original DoG+SIFT implementation is only available as a binary executable[10] which does not allow the flexibility needed for this evaluation. Instead, two publicly available SIFT implementations were used, from Rob Hess[11] and Andrea Vedaldi.[12] Due to subtleties in, for example, location refinement and thresholding, the implementations are not exactly equivalent. The comparison shows results for Vedaldi's implementation, which was found to perform slightly faster and better. For CenSurE, we use an implementation from WillowGarage, which is "[...] based on CenSurE with some modifications for increased speed and stability."[13] The patch descriptor only requires (sub)sampling the image at a given location.

It should be noted that, unlike in some of the visual tracking systems in Table 1, the image patch does *not* get warped according to its expected appearance in the target frame. As none of the descriptors claim to be fully invariant to perspective distortion, this step would improve the performance for all of them, so the (in)ability of performing a correct estimate is a property of the system and independent of this evaluation. With respect to a system that predicts the camera pose and warps the image accordingly, the tests below therefore show how much error can be tolerated. Similarly, we employ the corner detectors only on the full-scale image—for some applications, it might be desirable to embed them into a multi-scale framework (see, e.g., Chekhlov et al. 2006; Klein and Murray 2007; Wagner et al. 2009).

Execution time is measured only for the core part of the algorithms, that is, without the unifying interface needed for the evaluation or any initialization or memory allocation steps that could be moved to a pre-processing step.

Reported timings refer to an IBM Thinkpad T60 with a 1.83 GHz Dual Core CPU (only one core used for the computations) running Linux.

6 Results and Analysis

6.1 Interest Point Detectors

6.1.1 Parameter Configurations of Detectors

A total of 25 parameters for the various detectors were evaluated in terms of their impact on the performance of the algorithm, namely, execution time, number of detected features and repeatability. We found that, while some parameters have a clear optimal value and/or only limited impact (within a certain range of reasonable values) on two or all three measures, some parameters incorporate a tradeoff between, for example, time and repeatability, and that ill-chosen values may significantly reduce the performance. Due to space limitations and the amount of data from this particular stage, the results are not presented here; the interested reader is referred to a technical report (Gauglitz et al. 2010). We then fixed one set of parameters for the following comparisons. The numerical values can be found in Table 3 in the Appendix.

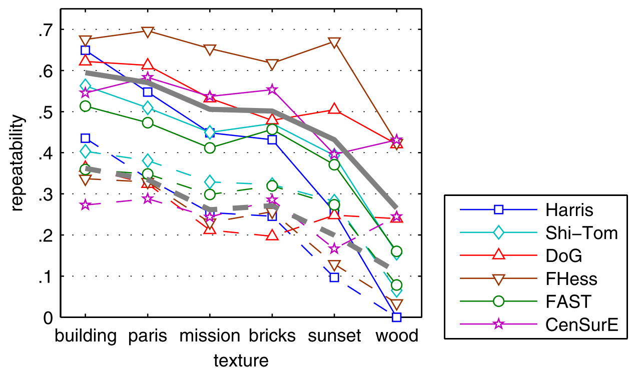


Fig. 9 Repeatability on the different textures, motion pattern "unconstrained." Here, *solid lines* indicate performance for consecutive frames, *dashed lines* indicate performance for randomly selected pairs (5000 per texture). The *bold gray lines* indicate the average over all detectors

6.1.2 Comparison of Detectors

Figure 10 presents the obtained detector comparison results, specifically, the repeatability under various conditions.

Figure 10(a) shows the repeatability vs. the number of detected points (in the textured region of interest) achieved by the detectors by varying the respective thresholds. For all following experiments, the threshold was set to the value given in Table 3 (indicated by the respective marker in Fig. 10(a)).

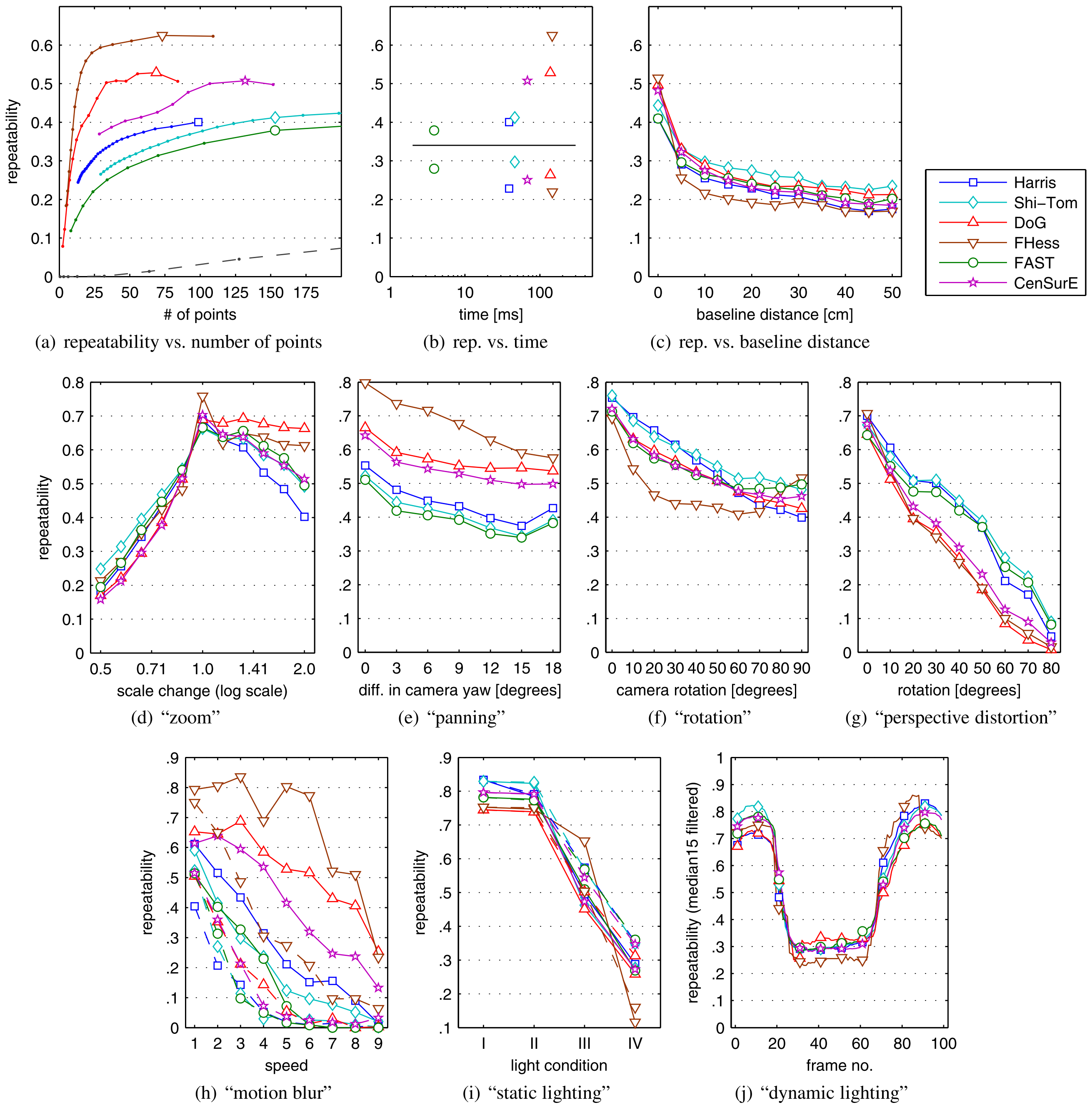
Figure 10(b) explores the tradeoff repeatability vs. execution time, both in the case of consecutive frames and random frame pairs. For the latter, the repeatability of all detectors decreases significantly, however, the performance of the "center-oriented" detectors, especially Fast Hessian, drops much farther. This result is broken down by texture in Fig. 9, and by the relative baseline distance in Fig. 10(c). It can be analyzed further using the isolated motion patterns Figs. 10(d)–(g): while large perspective distortion is the biggest challenge for all detectors, the corner detectors maintain higher repeatability for medium angles. As perspective distortion occurs frequently in many "real-world"

[10]http://www.cs.ubc.ca/~lowe/keypoints/.

[11]http://web.engr.oregonstate.edu/~hess/.

[12]http://vision.ucla.edu/~vedaldi/code/siftpp/siftpp.html.

[13]http://pr.willowgarage.com/wiki/Star_Detector.

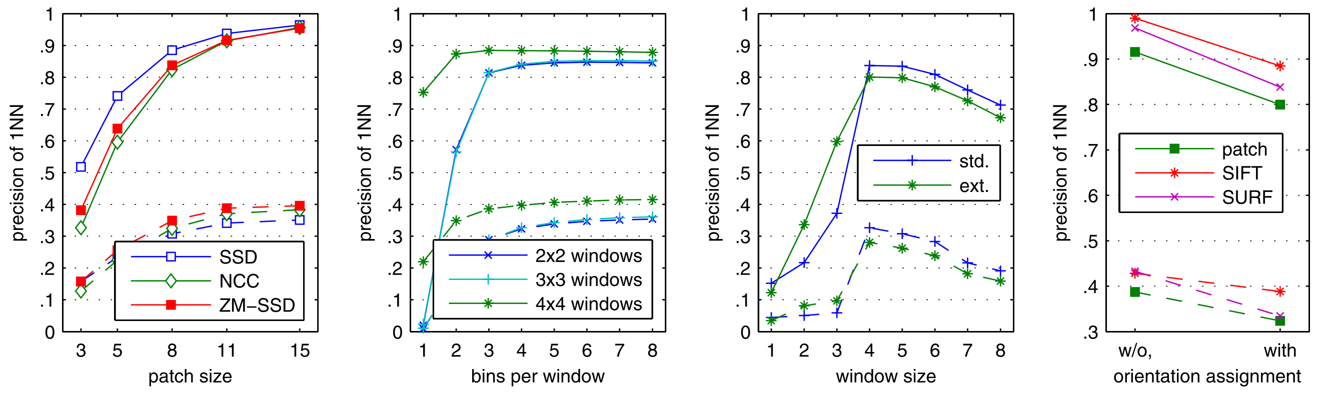


(a) repeatability vs. number of points (b) rep. vs. time (c) rep. vs. baseline distance

(d) “zoom” (e) “panning” (f) “rotation” (g) “perspective distortion”

(h) “motion blur” (i) “static lighting” (j) “dynamic lighting”

Fig. 10 Repeatability (Rep.) of detectors under various conditions. (**a**)–(**c**) use the motion pattern “unconstrained”; for (**d**)–(**j**), the subcaptions indicate the used motion pattern. All results averaged for all textures and frame pairs. (**a**) Rep. on consecutive frames vs. number of detected points (in the region of interest), varying the respective threshold parameter. For comparison, the *dashed gray line* indicates the repeatability of randomly selected points, thus clearly visualizing the criterion’s bias. (**b**) Rep. vs. execution time, for all consecutive frame pairs (above line) and 30 000 pairs of randomly selected frames (5000 per texture) (*below line*). (**c**) Rep. on random frame pairs as a function of the baseline distance between the two frames. (**d**)–(**g**) Rep. as function of geometric changes. For each figure, 5 × 500 random frame pairs (500 per texture) of the respective motion pattern were evaluated and then binned and averaged according to the relative change in the camera’s position between the two frames. (**h**) Rep. in the case of motion blur, both during motion (*solid lines*) and compared to the first, still frame (*dashed lines*). For absolute values of the speeds 1–9 refer to Sect. 3.2. (**i**) Rep. for the four different light conditions, both within one condition (*solid lines*) and compared to the first condition (*dashed lines*). (**j**) Rep. for dynamic light changes. Here, the repeatability is shown over time (i.e. frames) and filtered with a median filter of length 15 to make the figure legible (note that (**j**) is the only figure that shows repeatability for single frames). While on average, all detectors show the same behavior, detailed analysis of lighting changes for each texture (not shown) reveal differences



(a) patch (b) SIFT (c) SURF (d) orientation assignment

Fig. 11 Precision for different descriptor variants, averaged for all detectors and textures; consecutive frames (*solid lines*) and 5 × 500 randomly selected frame pairs (*dashed lines*); motion patterns "unconstrained." (**a**) Precision of matching with an image patch, varying the patch's size and the similarity measure (SSD = sum of squared differences, ZM SSD = zero mean SSD, NCC = normalized cross correlation). (**b**) Precision of matching with SIFT, varying the number of subwindows and the bins for each of the windows. (**c**) Precision of matching with SURF, varying the size of the subwindows. (**d**) Comparison of descriptors with and without orientation assignment. Here, the orientation assignment of SURF is used for patch as well

camera paths as well as our pattern "unconstrained," this result is important.

The difficulty of the corner detectors to cope with motion blur (Fig. 10(h)) and the texture "wood" (Fig. 9) is apparent.

Fast Hessian performs best for small baseline distances for almost every texture (Fig. 9), as well as for panning motion (Fig. 10(e)) and motion blur (Fig. 10(h)).

None of the detectors cope well with the increased noise level of the darker static lighting conditions (Fig. 10(i)). Figure 10(j) is the only figure that shows repeatability over time, namely, while the lighting is changed (note that the output is smoothed with a median filter to make the figure legible). While all detectors show the same behavior on average, detailed analysis of lighting changes for each texture (not shown) reveal differences: interestingly, there are two textures in which Fast Hessian outperforms all others and maintains reasonable repeatability throughout the light change, but two others where it fails to find any features and is outperformed by all others.

6.2 Feature Descriptors and Classifiers

6.2.1 Parameter Configurations of Descriptors and Classifiers

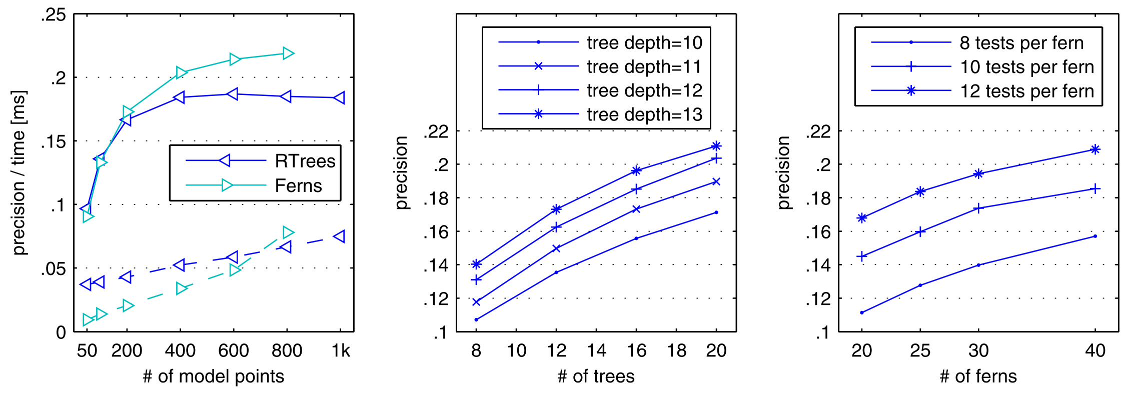
As for the detectors, we first evaluated different parameters for the descriptors. Figure 11(a) reveals that the gain of increasing the patch size beyond 11 pixels is very small—indeed, existing systems use patch sizes of 8 or 11 (cf. Table 1). In the following, a normalized patch of size 11 and sum of squared differences (SSD) as similarity measure are used.

Figure 11(b) shows the matching precision of SIFT while varying the number of subwindows and the number of bins per window. The default parameters (4 × 4 and 8, respectively) result in high precision, but a long descriptor vector (128). For the comparison, we decided to use a configuration of 4 × 4 windows and 3 bins, which performs almost as well, and decreases storage requirements and time needed for matching by 63%.

According to Fig. 11(c), SURF performs best with the default of window size 4 and the 'standard' configuration (for details on the 'extended' configuration see Bay et al. 2008) and is hence used for the comparison.

Figure 11(d) shows that for large parts of our testbed, orientation assignments are, on average, not beneficial for any of the descriptors, as in-plane rotation does not occur frequently enough to outweigh the disadvantage of discarding the original orientation. With the exception of Figs. 13(f) and 17(g), it is therefore not used in the evaluation. These two figures hence deserve special attention for applications that require rotation invariance.

The keypoint classifiers using Randomized Trees and Ferns have a whole array of parameters concerning the training (number of views generated during different phases, number of model points to train for), properties of the Trees/Ferns (number, depth) and the underlying image region (size of patch). While some parameters only influence the training phase (e.g., number of training samples), other parameters define memory requirements and running time. Figure 12 shows three examples of the parameter evaluations. The number of model points, N, is especially interesting in two respects: First, it includes a trade-off between the upper bound on how many points can be correctly classified, regardless of the viewing conditions (cf. Sect. 5.2), and the expressiveness of the classification (too many classes will "blur" the classification results), which can be observed in



(a) R. Trees & Ferns (b) Randomized Trees (c) Ferns

Fig. 12 Impact of different parameters for the classifiers on their classification precision, averaged for all frames, textures and detectors, motion pattern "unconstrained." (**a**) Number of model points, *dashed lines* depict the time needed to classify a single point. The superlinear increase in the time for Ferns might be caused by extreme memory requirements, the test with 1000 points aborted on our test machine. (**b**) Number of trees and depth and (**c**) number of ferns and depth

Table 2 Time to compute a single descriptor. Note that this table conceals an important advantage of the keypoint classifiers: their result is a *classification*, which may be fed directly into outlier removal/pose estimation algorithms, while the descriptors first have to be matched against descriptors of previously seen keypoints. The absolute values are implementation and hardware specific, but the relative order of magnitude is unlikely to change unless special hardware acceleration is used

Descriptor	Patch	SIFT	SURF	R. Trees	Ferns
Time [s]	5×10^{-6}	1×10^{-3}	2×10^{-4}	7×10^{-5}	5×10^{-5}

Fig. 12(a) for Randomized Trees.[14] Second, it heavily influences the running time, as for each classification, vectors of length N have to be added (one for each Tree/Fern in the set) and the optimal value extracted.

The number of tests per Tree/Fern and the number of Trees/Ferns influences especially the memory requirements, and extension beyond the depicted ranges was found to be impractical. The performance was reported to level off soon beyond this range by Lepetit and Fua (2006) and Özuysal et al. (2007), respectively.

The parameter values used for the comparison can be found in Table 4 in the Appendix.

6.2.2 Comparison of Descriptors and Classifiers

Figure 13 presents the obtained descriptor/classifier comparison results, specifically, the precision under various conditions.[15] Table 2 lists the corresponding execution times. For interpretation of these results, it is important to keep the differences between descriptors and classifiers in mind, namely, (1) results grouped by baseline or orientation distances refer to *different frame pairs*, (2) for the descriptors, the matching is constrained to a certain region around the feature, and (3) the precision of the classifiers is inherently limited if the number of points detected on the current frame exceeds the number of model points (cf. Sect. 5.2). While (1) should not influence the value, and the influence of (2) is shown to be limited in Fig. 13(a), (3) may, depending on the detector, reduce the precision significantly. We nevertheless opted for showing both results together for the sake of space and because, if the above limitations are kept in mind, one might still gain useful insights from the comparison.

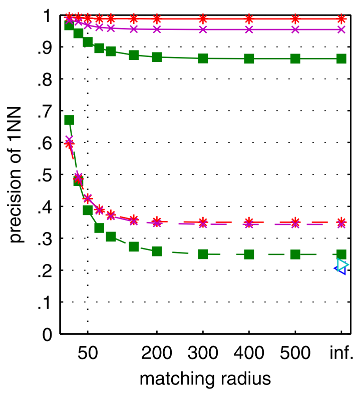
Figure 13 shows that, assuming the perfect detector, the precision for all descriptors is rather similar in most conditions, and very high for short baseline distances (Figs. 13(b)–(d)). The radius of the search region, and therefore the number of candidate features that a new feature is matched against, has limited impact (Fig. 13(a)). As mentioned in Sect. 5.2, the radius is set to 50 pixels for all other experiments due to considerations about the application of visual tracking.

In-plane rotation of more than about 10° necessitates orientation assignments (Fig. 13(f)), and, as for the detectors, perspective distortion is the hardest challenge for all descriptors (Fig. 13(g)).

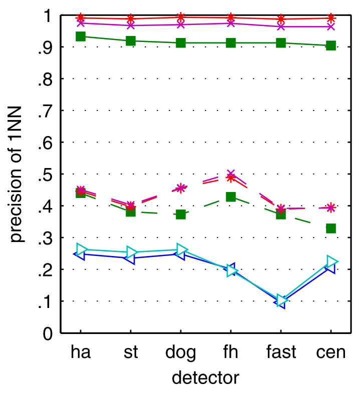
Matching *during* fast motion works very well with all three descriptors even for extremely blurred frames (solid

[14]Özuysal et al. (2007) show the same effect, though for larger N, for Ferns.

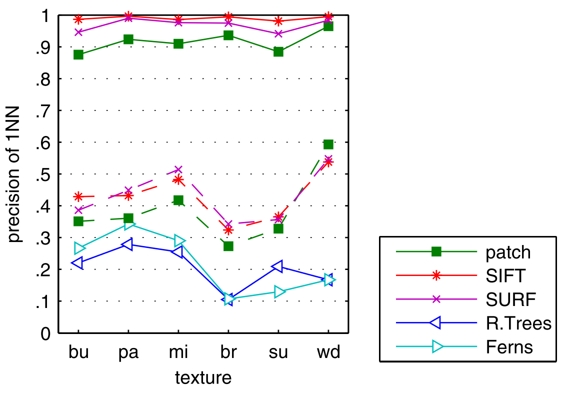
[15]Note that the condition "zoom" was omitted: the results would be misleading, as it is the *detector* that is responsible for identifying the "scale" of the keypoint; however, due to our setup of reprojecting the keypoints, this information is lost. Refer to Fig. 17(e) for tracking success including all information.



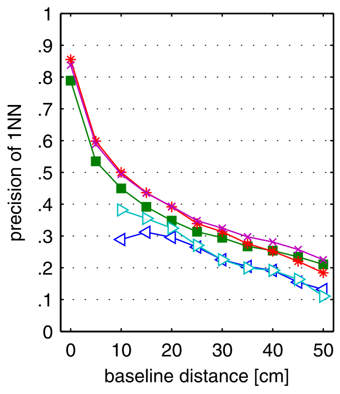
(a) impact of size of search region



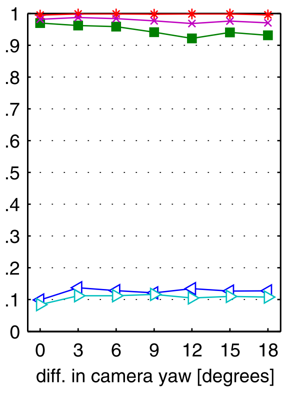
(b) averaged over textures



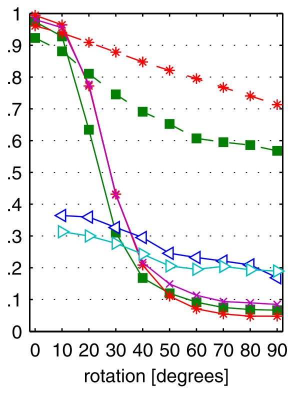
(c) averaged over detectors



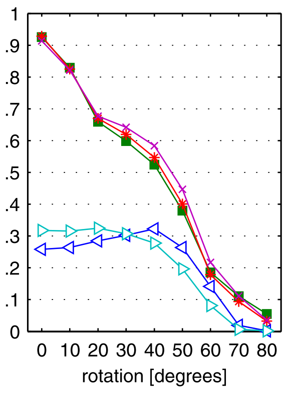
(d) prec. vs. baseline distance



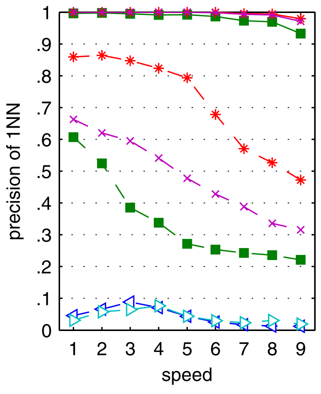
(e) “panning”



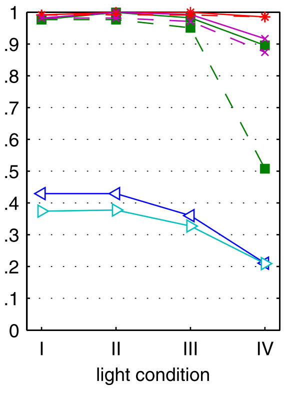
(f) “rotation”



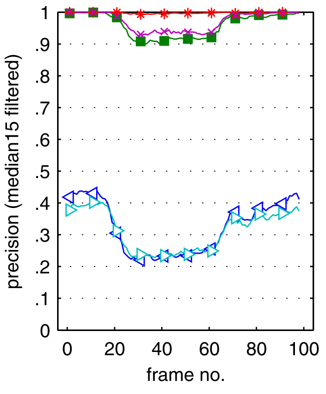
(g) “perspective distortion”



(h) “motion blur”



(i) “static lighting”



(j) “dynamic lighting”

Fig. 13 Precision (Prec.) of descriptors and classifiers under various conditions. (**a**)–(**d**) Use the motion pattern “unconstrained”; for (**e**)–(**j**), the subcaptions indicate the used motion pattern. All results averaged for all respective frame pairs and textures. Note that there are important differences in how the problem is posed for the descriptors vs. the classifiers, which favor the descriptors in this figure. See Sect. 5.2 for details. (**a**) Prec. vs. size of the search region. For all remaining experiments, the radius is set to 50 pixels. (**b**) Prec. broken down by detector-descriptor combination, *solid lines*: consecutive frames, *dashed lines*: 5 × 5000 random frame pairs. (**c**) Prec. broken down by textures, *solid/dashed lines* as in (**b**). Here, the reason that “wood” performs so well is that there are very few keypoints, so that the number of candidate points (from which the correct one has to be identified) is very small. (**d**) Prec. vs. baseline distance. (**e**)–(**g**) Prec. as function of geometric changes. For each figure, 5 × 500 random frame pairs of the respective motion pattern were evaluated. In (**f**), precision is evaluated both with orientation assignment (*dashed lines*) and without (*solid lines*, used for all other experiments due to the result of Fig. 11(d)). (**h**) Prec. in the case of motion blur, matching against the first, *still* frame (*dashed lines*) and between frames under motion (*solid lines*). For absolute values of the speeds 1–9 refer to Sect. 3.2. (**i**) Prec. for the four different light conditions, both within one condition (*solid lines*) and compared to the first condition (*dashed lines*). (**j**) Prec. for dynamic light changes, shown over time (i.e. frames) and filtered with a median filter of length 15 to make the figure legible (note that (**j**) is the only figure that shows precision for single frames)

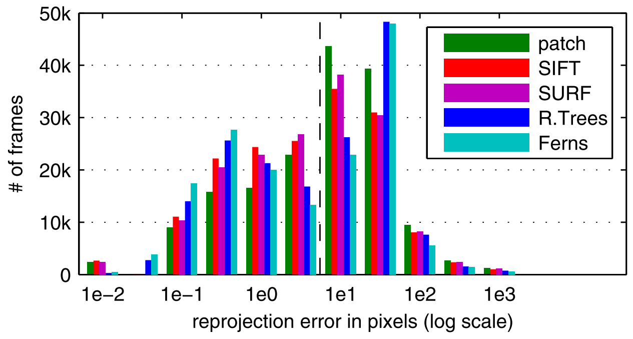


Fig. 14 Reprojection error by descriptors

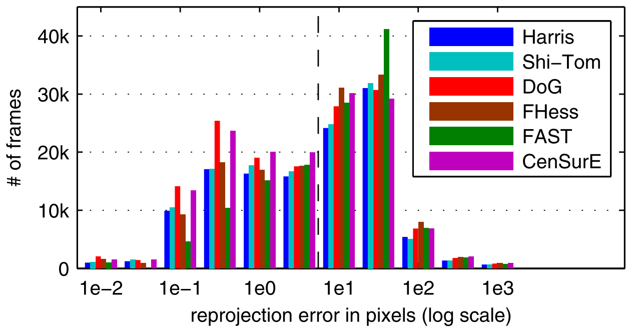


Fig. 15 Reprojection error by detectors

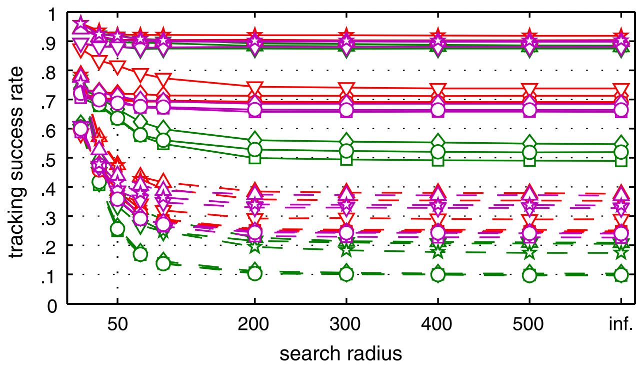


Fig. 16 Tracking success rate vs. size of the search region. Here, marker *shape* indicates detector (see legend in Fig. 10), *color* indicates descriptor (see legend in Fig. 13). For all remaining experiments, the radius is set to 50 pixels

lines in Fig. 13(h)), and SIFT outperforms others if matching from a non-blurred to a blurred frame (dashed lines in Fig. 13(h)). The latter is somewhat surprising given that the gradients on which SIFT relies should be significantly altered due to the (directed) motion blur.

Matching within or across static light changes (Fig. 13(i)) and even during transition (Fig. 13(j)) poses little problems for all of the descriptors (despite high noise levels for low light condition), even the simple intensity normalization for patch seems sufficient for all but the darkest light condition. SIFT is the most robust under light changes.

While the absolute precision values for the classifiers are low, they confirm that due to the training step, the impact of pose changes is reduced and the performance is rather constant throughout a range of baseline distances (Fig. 13(d)), in-plane rotation (Fig. 13(f)) and perspective distortion (Fig. 13(g)).

6.3 Tracking Success of Detector + Descriptor

Histograms of the tracking accuracy (i.e., reprojection error) obtained by the various combinations are shown in Figs. 14 and 15, broken down by descriptors and detectors, respectively. As mentioned above, in the following we define a frame to be "successfully tracked" if the reprojection error is less than 5 pixels (dashed line in Figs. 14 and 15).

Figure 16 illustrates the influence of the size of the search region. As before, the radius is set to 50 pixels for all remaining experiments.

Figure 17(a) shows the success rate for each detector/descriptor combination, both for consecutive frames (i.e., frame-to-frame tracking) and random frame pairs. Especially for frame-to-frame tracking, the difference among the three descriptors (patch, SIFT, and SURF) is very small and overweighted by the differences among detectors. Combinations with CenSurE as detector perform very well, which is due to the very good performance on the "wood" texture, as may be concluded from Fig. 17(c). As already indicated in Fig. 9, the corner detectors (Harris, Shi-Tomasi, and FAST) are not able to cope with this texture and lead to tracking failure for any descriptor. CenSurE and DoG are the best detectors for use with the classifiers, while using FAST is apparently not a good idea (Fig. 17(a)).

Looking at the results of individual combinations broken down by baseline distance (Fig. 17(d)), {DoG/CenSurE} + {Randomized Trees/Ferns} now clearly outperform all other solutions for baseline distance of more than 10 cm, with the best non-trained solution being {DoG/CenSurE} + SIFT. The computationally cheapest solution, FAST+patch, fails in roughly 25% more cases than DoG+SIFT (i.e., the classic SIFT combination). The textures "wood" and "sunset" are responsible for a significant part of this (Figs. 17(b) and 17(c)).

While the classifier-based solutions do not perform well in the case of panning (Fig. 17(f)) and are less robust than {DoG/CenSurE} + SIFT in the case of in-plane rotation, they clearly outperform descriptor-based solutions in the case of perspective distortion (Fig. 17(h)), where {DoG/CenSurE} + {Randomized Trees/Ferns} perform close to perfect up to 50° out-of-plane rotation.

In the case of motion blur (Fig. 17(i)), all classifier-based solutions perform rather poorly, as do all combinations with corner detectors. Fast Hessian + {patch/SIFT/SURF} perform best, both for tracking *during* motion and a starting motion, which is not unexpected given the results of Fig. 10(h).

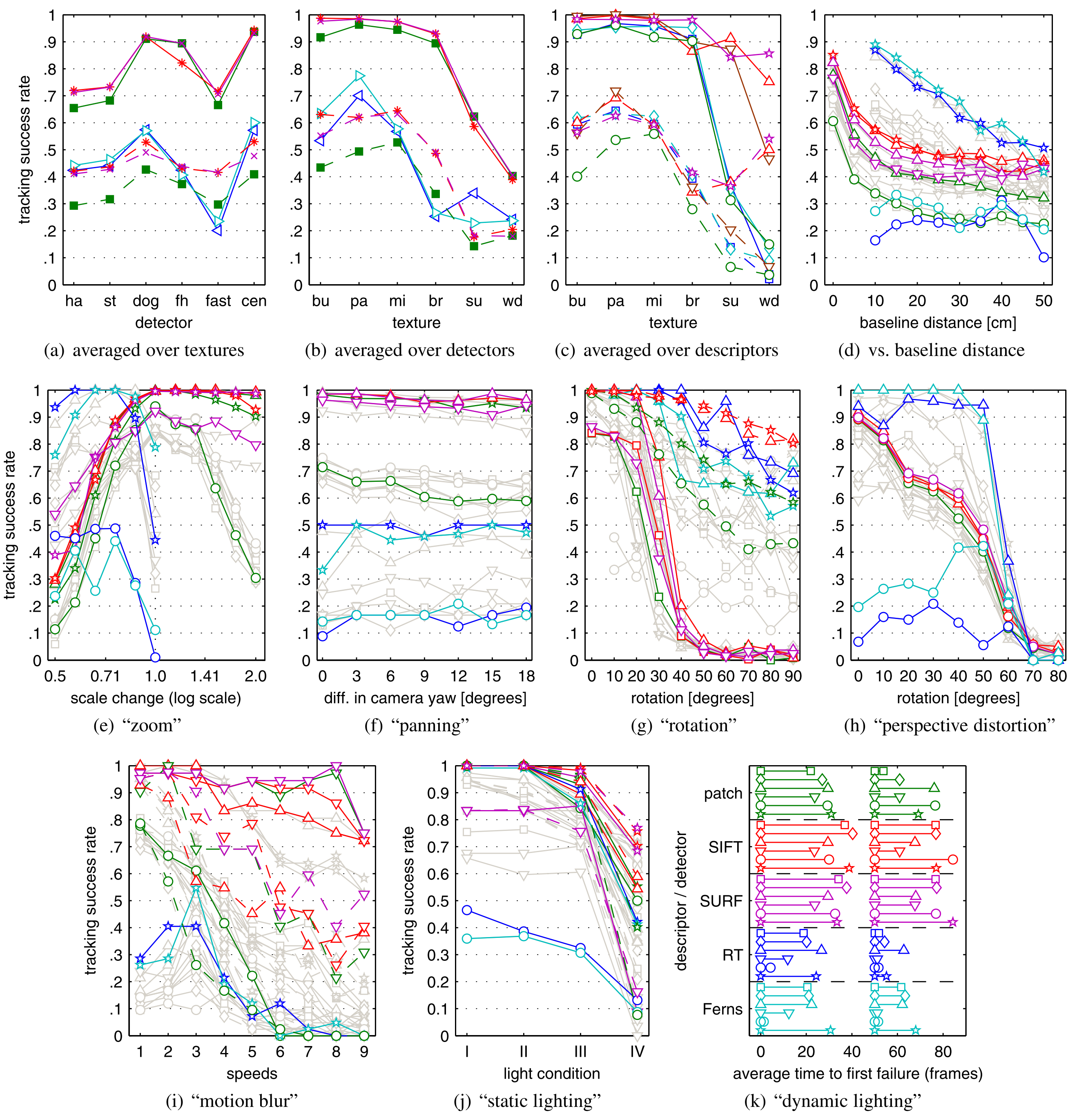
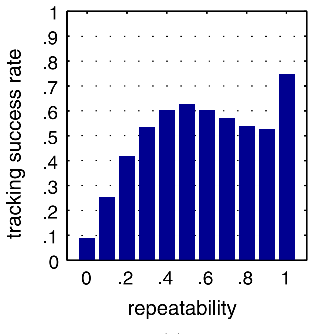


Fig. 17 Tracking success rate (Tsr.) of all detector-descriptor/classifier combination under various conditions. (**a**)–(**d**) use the motion pattern "unconstrained"; for (**e**)–(**j**), the subcaptions indicate the used motion pattern. All results averaged for all respective frame pairs and textures. In (**d**)–(**j**), the *shape* of the marker indicates the detector (legend as in Fig. 10) and the *color* indicates the descriptor (legend as in Fig. 13). To reduce clutter, only a few combinations are highlighted and identifiably marked, although the results for all 30 combinations are shown in light gray to convey the spread of performance. (**a**) Tsr. averaged for all textures and broken down by descriptors and detectors (legend as in Fig. 13). (**b**) Tsr. averaged for all detectors and broken down by descriptors and textures (legend as in Fig. 13). (**c**) Tsr. averaged for all descriptors and broken down by detectors and textures (legend as in Fig. 10). (**d**) Tsr. as a function of the baseline distance (motions "unconstrained," 5 × 5000 random frame pairs). (**e**)–(**h**) Tsr. as function of geometric changes, 5 × 500 random frame pairs of the respective motion pattern. (**i**) Tsr. for different levels of motion blur, both during motion (*solid lines*) and to the first, still frame (*dashed lines*). (**j**) Tsr. for the four different lighting conditions, both within one condition (*solid lines*) and compared to the first, brightest condition (*dashed lines*). (**k**) Time until tracking fails (averaged over textures) during the lighting changes from light to dark (*left half*) and back (*right half*). Figures 14–19 are best viewed in color, please refer to the online version of this article



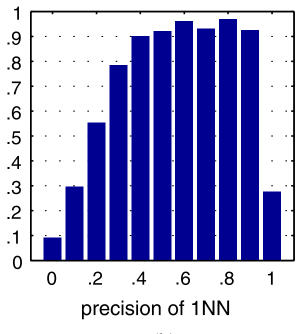


Fig. 18 Tracking success as a function of the respective frame (pair)'s (**a**) repeatability and (**b**) 1NN precision. The reason for the low success rate in the case of 100% precision is the fact that in these failed frames, too few points were detected: thus even though all of them were correctly matched, the homography cannot be recovered. In this graph, this effect is so strong because the likelihood of 100% precision is very small, but the smaller the number of points, the higher the (random) chance of matching all of them correctly

For tracking in low light conditions (Fig. 17(j)), again the CenSurE-based solutions perform best. The results for tracking during light change (Fig. 17(k)) are rather mixed, but confirm several trends mentioned above, e.g., solutions with SIFT perform well (though, interestingly, better with the corner detectors), and Randomized Trees/Ferns perform best with CenSurE as detector.

6.4 Further Analysis

Figure 18 show the average tracking success rate given a certain level of repeatability (a) and precision (b). Note that these figures do not necessarily display a strict causal relationship: several artifacts can be observed that can only be explained with other influences, e.g., the descriptor's precision which limits the repeatability's impact, and the number of points, which may cause tracking to fail despite high precision. However it becomes clear that if either repeatability or precision fall below 0.3, it significantly reduces the chances of successful tracking.

Next, we investigate the impact of the next step in the processing chain (cf. Fig. 6), namely, outlier removal. As mentioned above, our use of RANSAC as well as specific parameters is arbitrary and other algorithms such as MLESAC (Torr and Zisserman 2000), PROSAC (Chum and Matas 2005) or M-estimators (Zhang 1997) could be used (cf. Table 1). Figure 19 shows the tracking success rate of all detector-descriptor combinations as a function of the number of RANSAC iterations allowed. Especially the combinations based on detectors that return many keypoints (the corner detectors and CenSurE, cf. Fig. 10(a)) benefit from more iterations, which will generally be a computationally more efficient solution than opting for a more expensive descriptor (cf. Table 2). However, the increase is limited. Given enough iterations and with certain assumptions with respect to error

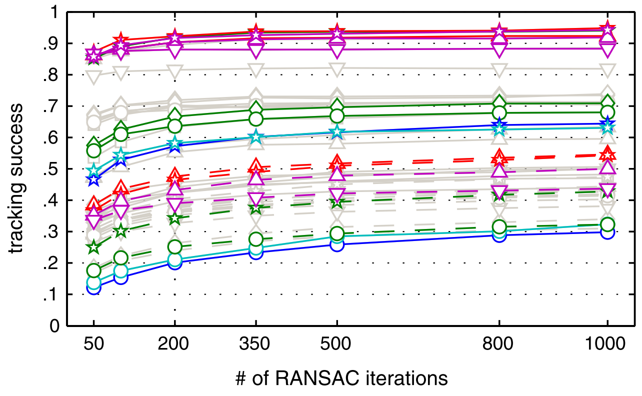


Fig. 19 Impact of the number of iterations for RANSAC on the tracking success rate. Legend as in Fig. 17 (*shape* indicates detector, *color* indicates descriptor)

measures, RANSAC is effectively similar to MLESAC and PROSAC, hence Fig. 19 may serve as a rough indication for their chances of success.

7 Conclusions

We have presented an extensive dataset of videostreams that we make available with all data necessary to run evaluations of different kinds. In terms of conditions that the dataset includes (textures with different levels of difficulty, geometric distortions, levels of motion blur, lighting conditions), it is more extensive than other available datasets. We hope that other researchers will find it useful as a testbed for a variety of algorithm classes in the context of visual tracking.

Using this dataset, we have presented a comprehensive evaluation of detectors and descriptors for feature-based visual tracking: we first evaluated an array of algorithm parameters as to their impact on tracking performance in order to obtain the best configuration for each algorithm, then compared both detectors and descriptors in isolation, and finally compared a total of 30 detector-descriptor combinations as tracking solutions. To our knowledge, this is the first work that compares these algorithms specifically in the context of visual tracking.

The value of this evaluation lies in the abundance of data presented in Figs. 9–19, and in presenting quantitative evidence for many trends. Given the many different conditions on which the algorithms were evaluated, unknown requirements of specific applications and the trade-offs involved, it is difficult to derive universally valid recommendations or proclaim a single "winner." Also, some of the results (such as: combinations with SIFT tend to be more robust than combinations using an image patch) are hardly surprising. However, there are many detailed insights that emerge from this analysis, including the following:

The tested center-oriented detectors (DoG, Fast Hessian, CenSurE) provide higher repeatability than the corner detectors (Harris, Shi-Tomasi, FAST) for smooth motion and short baseline distances, but they are more susceptible especially to perspective distortion. Fast Hessian and Fast Hessian-based tracking solutions cope best with strong motion blur. In terms of size of the image patch descriptor, a point of diminishing returns is reached around 11×11 pixels. The dimensionality of SIFT can be significantly reduced without affecting its precision by reducing the number of bins of the underlying histograms. For a perfect detector, the matching precision of different descriptors would be similar, but differences emerge in combination with different (obviously non-perfect) detectors. Interestingly, CenSurE + SIFT performs slightly better than DoG + SIFT. A non-trained tracking solution that handles out-of-plane rotation beyond 20° satisfactorily has yet to be found—this confirms findings by Moreels and Perona (2007) in a different context. Trained classifiers clearly outperform descriptors for medium baseline distances (especially in the case of out-of-plane rotation), but only if combined with an appropriate detector: Randomized Trees and Ferns perform especially well with CenSurE, and especially badly with FAST.

We also presented data on computation times for all algorithms, for different textures, influence of the search radius, the computation allotted for outlier removal, and correlation between the low-level metrics repeatability and 1NN precision and the high-level metric tracking success.

Our data may be used in several different ways: First, it shows what level of performance to expect and which claims about the algorithms can be confirmed in practice and which ones cannot, in particular for the image quality available in real-time tracking. Second, it provides quantitative support for the decision as to which detector or descriptor to choose for a particular tracking problem, how to choose parameter values, and what tweaks are most promising to increase performance of a given system. Third, especially due to the detailed multi-stage evaluation approach, it provides insights into strengths and drawbacks of the algorithms, and may stimulate ideas of how to improve existing algorithms, how to combine them (both at algorithm and/or system level), or which avenues are promising for future research.

In terms of limitations of our work, it should be noted that, despite the high-level metric of tracking success, our evaluations are still to be considered analytical in nature: they focus on insights on the different algorithms rather than recommending the perfect tracking solution. For the latter, application-specific requirements such as supported motions, processing power and required framerate would have to be known, and the evaluations would have to encompass even more components (e.g., outlier removal strategies, pose estimators and predictors), system integration features such as multi-scale frameworks or patch warping (cf. Sect. 5.5), as well as non-feature-based solutions such as alignment-based techniques (Benhimane and Malis 2004; Adams et al. 2008). Moreover, although we report execution times, we do not explicitly explore the trade-offs involved in budgeting a fixed amount of time for different components or voluntarily adjusting the framerate. We hope that the presented dataset will prove useful for future work in these areas.

Acknowledgements We thank Jürgen Roßmann and Petra Krahwinkler (RWTH Aachen Univ.) for their supervision of SG's earlier work on this topic, Rob Hess (Oregon State Univ.) and Andrea Vedaldi (UCLA) for their work of creating open SIFT implementations, and Nicole Holstrom and Andy Weinberg (UCSB College of Engineering Machine Shop) for their help in manufacturing our acrylic glass frame. This work was partially supported by a Chancellor's Fellowship from UCSB, NSF CAREER grant IIS-0747520, and ONR grant N00014-09-1-1113.

Appendix

Tables 3 and 4 list the parameter values that were used in the comparisons for detectors and descriptors/classifiers, respectively, except where specifically indicated otherwise in the respective figure caption.

Table 3 Parameter values for the interest point detectors

Detector	Parameter	Value
Harris	k	0.15
	threshold θ	0.001
	$w(u, v)$	$\exp\{-(u^2+v^2)/2\sigma^2\}$
	σ	2
	kernel size	2σ
	gradient kernel	Sobel
Shi-Tomasi	threshold θ	0.022
	$w(u, v)$	$\exp\{-(u^2+v^2)/2\sigma^2\}$
	σ	1.5
	kernel size	1.5σ
	gradient kernel	Sobel
DoG	octaves	4
	levels per octave	3
	σ_0	1.6
	θ_{edge}	10
	θ_{contr}	0.02
Fast Hessian	octaves	4
	threshold θ	4
	sampling step n	1
	initLobe	3
FAST	threshold t	20
CenSurE	scales	6
	response threshold	6
	line thresh. (binarized)	10
	line thresh. (projected)	10

Table 4 Parameter values for the descriptors and classifiers

Descriptor	Parameter	Value
Patch	size	11 × 11
	avg. intensity normalized to zero	
SIFT	subwindows	4 × 4
	bins per window	3
SURF	standard scheme (not extended)	
	window size	4
Rand. Trees	# of views to detect stable points	3000
& Ferns	# of points in model N	600
	minimum detection rate	0
	# of views for training	10000
	patch size	32
Rand. Trees	# of trees	20
	tree depth	12
Ferns	# of Ferns	30
	# of tests per Fern	12

References

Adams, A., Gelfand, N., & Pulli, K. (2008). Viewfinder alignment. *Computer Graphics Forum*, *27*(2), 597–606. doi:10.1111/j.1467-8659.2008.01157.x.

Agrawal, M., Konolige, K., & Blas, M. R. (2008). CenSurE: Center surround extremas for realtime feature detection and matching. In *Proceedings of the European conference on computer vision (ECCV'08)* (Vol. 5305, pp. 102–115). doi:10.1007/978-3-540-88693-8_8.

Baker, S., & Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'01)* (Vol. 1, pp. 1090–1097).

Baker, S., Scharstein, D., Lewis, J. P., Roth, S., Black, M. J., & Szeliski, R. (2007). A database and evaluation methodology for optical flow. In *Proceedings of the IEEE intl. conference on computer vision (ICCV'07)* (pp. 1–8). doi:10.1109/ICCV.2007.4408903.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, *110*, 346–359. doi:10.1016/j.cviu.2007.09.014.

Beaudet, P. R. (1978). Rotationally invariant image operators. In *Proceedings of the intl. joint conference on pattern recognition* (pp. 579–583).

Belongie, S., Malik, J., & Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(4), 509–522. doi:10.1109/34.993558.

Benhimane, S., & Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. In *Proceedings of the IEEE/RSJ intl. conference on intelligent robots and systems* (pp. 943–948).

Bleser, G., & Stricker, D. (2008). Advanced tracking through efficient image processing and visual-inertial sensor fusion. In *Proceedings of the IEEE virtual reality conference (VR'08)* (pp. 137–144). doi:10.1109/VR.2008.4480765.

Brown, M., & Lowe, D. (2002). Invariant features from interest point groups. In *Proceedings of the British machine vision conference (BMVC'02)*.

Calonder, M., Lepetit, V., & Fua, P. (2008). Keypoint signatures for fast learning and recognition. In *Proceedings of the 11th European conference on computer vision (ECCV'08)*, Marseille, France.

Campbell, J., Sukthankar, R., & Nourbakhsh, I. (2004). Techniques for evaluating optical flow for visual odometry in extreme terrain. In *Proceedings of the IEEE/RSJ intl. conference on intelligent robots and systems* (Vol. 4, pp. 3704–3711).

Carneiro, G., & Jepson, A. D. (2003). Multi-scale phase-based local features. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'03)* (Vol. 1, pp. 736–743).

Carrera, G., Savage, J., & Mayol-Cuevas, W. (2007). Robust feature descriptors for efficient vision-based tracking. In *Proceedings of the 12th Iberoamerican congress on pattern recognition* (pp. 251–260). doi:10.1007/978-3-540-76725-1_27.

Chekhlov, D., Pupilli, M., Mayol-Cuevas, W., & Calway, A. (2006). Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In *Proceedings of the 2nd intl. symposium on visual computing*.

Chekhlov, D., Pupilli, M., Mayol, W., & Calway, A. (2007). Robust real-time visual SLAM using scale prediction and exemplar based feature description. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'07)* (pp. 1–7). doi:10.1109/CVPR.2007.383026.

Cheng, Y., Maimone, M. W., & Matthies, L. (2006). Visual odometry on the mars exploration rovers—a tool to ensure accurate driving and science imaging. *IEEE Robotics & Automation Magazine*, *13*(2), 54–62. doi:10.1109/MRA.2006.1638016.

Chum, O., & Matas, J. (2005). Matching with PROSAC—progressive sample consensus. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'05)* (pp. 220–226). doi:10.1109/CVPR.2005.221.

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(6), 1052–1067. doi:10.1109/TPAMI.2007.1049.

DiVerdi, S., & Höllerer, T. (2008). Heads up and camera down: A vision-based tracking modality for mobile mixed reality. *IEEE Transactions on Visualization and Computer Graphics*, *14*(3), 500–512. doi:10.1109/TVCG.2008.26.

DiVerdi, S., Wither, J., & Höllerer, T. (2008). Envisor: Online environment map construction for mixed reality. In *Proceedings of the IEEE virtual reality conference (VR'08)* (pp. 19–26). doi:10.1109/VR.2008.4480745.

Eade, E., & Drummond, T. (2006a). Edge landmarks in monocular SLAM. In *Proceedings of the 17th British machine vision conference (BMVC'06)*, Edinburgh (Vol. 1, pp. 7–16).

Eade, E., & Drummond, T. (2006b). Scalable monocular SLAM. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'06)* (Vol. 1, pp. 469–476). doi:10.1109/CVPR.2006.263.

Ebrahimi, M., & Mayol-Cuevas, W. (2009). SUSurE: Speeded up surround extrema feature detector and descriptor for realtime applications. In *Workshop on feature detectors and descriptors: the state of the art and beyond. IEEE conference on computer vision and pattern recognition (CVPR'09)*.

Fiala, M. (2005). ARTag, a fiducial marker system using digital techniques. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'05)* (Vol. 2, pp. 590–596), Washington, DC, USA. doi:10.1109/CVPR.2005.74.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395. doi:10.1145/358669.358692.

Förstner, W. (1994). A framework for low level feature extraction. In *Proceedings of the 3rd European conference on computer vision (ECCV'94)*, Secaucus, NJ, USA (Vol. II, pp. 383–394).

Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *13*(9), 891–906. doi:10.1109/34.93808.

Gauglitz, S., Höllerer, T., Krahwinkler, P., & Roßmann, J. (2009). A setup for evaluating detectors and descriptors for visual tracking. In *Proceedings of the 8th IEEE intl. symposium on mixed and augmented reality (ISMAR'09)*.

Gauglitz, S., Höllerer, T., & Turk, M. (2010). Dataset and evaluation of interest point detectors for visual tracking (Technical Report 2010-06). Department of Computer Science, UC Santa Barbara.

Harris, C., & Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the 4th ALVEY vision conference* (pp. 147–151).

Hartley, R., & Zisserman, A. (2004). *Multiple view geometry in computer vision* (2nd ed.). Cambridge: Cambridge University Press.

Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America, A, Optics, Image Science & Vision*, *4*(4), 629–642.

Julier, S. J., & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In I. Kadar (Ed.), *Proceedings of the SPIE conference on signal processing, sensor fusion, & target recognition VI* (Vol. 3068, pp. 182–193). doi:10.1117/12.280797.

Kadir, T., Zisserman, A., & Brady, M. (2004). An affine invariant salient region detector. In *Proceedings of the 8th European conference on computer vision (ECCV'04)* (pp. 228–241).

Kato, H., & Billinghurst, M. (1999). Marker tracking and HMD calibration for a video-based augmented reality conferencerencing system. In *Proceedings of the 2nd IEEE and ACM intl. workshop on augmented reality (IWAR'99)* (p. 85), Washington, DC, USA.

Ke, Y., & Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'04)* (Vol. 2, pp. 506–513). doi:10.1109/CVPR.2004.183.

Kitchen, L., & Rosenfeld, A. (1982). Gray-level corner detection. *Pattern Recognition Letters*, *1*(2), 95–102. doi:10.1016/0167-8655(82)90020-4.

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 6th IEEE and ACM intl. symposium on mixed and augmented reality (ISMAR'07)*, Nara, Japan.

Klein, G., & Murray, D. (2008). Improving the agility of keyframe-based SLAM. In *Proceedings of the 10th European conference on computer vision (ECCV'08)*, Marseille, France (pp. 802–815).

Klein, G., & Murray, D. (2009). Parallel tracking and mapping on a camera phone. In *Proceedings of the 8th IEEE intl. symposium on mixed and augmented reality (ISMAR'09)* (pp. 83–86). doi:10.1109/ISMAR.2009.5336495.

Lee, S., & Song, J. B. (2004). Mobile robot localization using optical flow sensors. *International Journal of Control, Automation, and Systems*, *2*(4), 485–493.

Lee, T., & Höllerer, T. (2008). Hybrid feature tracking and user interaction for markerless augmented reality. In *Proceedings of the IEEE virtual reality conference (VR'08)* (pp. 145–152). doi:10.1109/VR.2008.4480766.

Lepetit, V., & Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(9), 1465–1479. doi:10.1109/TPAMI.2006.188.

Levin, A., & Szeliski, R. (2004). Visual odometry and map correlation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'04)* (Vol. 1, pp. 611–618). doi:10.1109/CVPR.2004.266.

Lieberknecht, S., Benhimane, S., Meier, P., & Navab, N. (2009). A dataset and evaluation methodology for template-based tracking algorithms. In *Proceedings of the IEEE intl. symposium on mixed and augmented reality (ISMAR'09)*.

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

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *Proceedings of the IEEE intl. conference on computer vision (ICCV'99)*, Corfu (pp. 1150–1157).

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

Matas, J., Chum, O., Urban, M., & Pajdla, T. (2002). Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British machine vision conference (BMCV'02)* (pp. 384–393).

Matthies, L., & Shafer, S. A. (1987). Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, *3*(3), 239–248.

McCarthy, C. D. (2005). Performance of optical flow techniques for mobile robot navigation (Master's thesis). Department of Computer Science and Software Engineering, University of Melbourne.

Mikolajczyk, K., & Schmid, C. (2001). Indexing based on scale invariant interest points. In *Proceedings of the IEEE intl. conference on computer vision (ICCV'01)* (Vol. 1, p. 525). doi:10.1109/ICCV.2001.10069.

Mikolajczyk, K., & Schmid, C. (2002). An affine invariant interest point detector. In *Proceedings of the 7th European conference on computer vision (ECCV'02)* (pp. 128–142), London, UK.

Mikolajczyk, K., & Schmid, C. (2004). Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, *60*(1), 63–86. doi:10.1023/B:VISI.0000027790.02288.f2.

Mikolajczyk, K., & Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1615–1630. doi:10.1109/TPAMI.2005.188.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T., & van Gool, L. (2005). A comparison of affine region detectors. *International Journal of Computer Vision*, *65*(7), 43–72.

Mohanna, F., & Mokhtarian, F. (2006). Performance evaluation of corner detectors using consistency and accuracy measures. *Computer Vision and Image Understanding*, *102*(1), 81–94. doi:10.1016/j.cviu.2005.11.001.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI national conference on artificial intelligence* (pp. 593–598).

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2003). FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the intl. joint conference on artificial intelligence (IJCAI'03)* (pp. 1151–1156).

Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover (Technical Report CMU-RI-TR-80-03). Robotics Institute, Carnegie Mellon University.

Moreels, P., & Perona, P. (2007). Evaluation of features detectors and descriptors based on 3D objects. *International Journal of Computer Vision*, *73*(3), 263–284. doi:10.1007/s11263-006-9967-1.

Moreno-Noguer, F., Lepetit, V., & Fua, P. (2007). Accurate non-iterative o(n) solution to the pnp problem. In *Proceedings of the IEEE international conference on computer vision (ICCV'07)* (pp. 1–8). doi:10.1109/ICCV.2007.4409116.

Neira, J., & Tardos, J. D. (2001). Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, *17*(6), 890–897. doi:10.1109/70.976019.

Nistér, D., Naroditsky, O., & Bergen, J. (2004). Visual odometry. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'04)* (Vol. 1, pp. 652–659). doi:10.1109/CVPR.2004.1315094.

Özuysal, M., Fua, P., & Lepetit, V. (2007). Fast keypoint recognition in ten lines of code. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'07)*, Minneapolis, Minnesota, USA. doi:10.1109/CVPR.2007.383123.

Park, Y., Lepetit, V., & Woo, W. (2008). Multiple 3D object tracking for augmented reality. In *Proceedings of the 7th IEEE and ACM intl. symposium on mixed and augmented reality (ISMAR'08)* (pp. 117–120). doi:10.1109/ISMAR.2008.4637336.

Rosten, E., & Drummond, T. (2005). Fusing points and lines for high performance tracking. In *Proceedings of the IEEE intl. conference on computer vision (ICCV'05)* (Vol. 2, pp. 1508–1511). doi:10.1109/ICCV.2005.104.

Rosten, E., & Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proceedings of the IEEE European conference on computer vision (ECCV'06)* (Vol. 1, pp. 430–443). doi:10.1007/11744023_34.

Schaffalitzky, F., & Zisserman, A. (2002). Multi-view matching for unordered image sets, or "How Do I Organize My Holiday Snaps?". In *Proceedings of the 7th European conference on computer vision (ECCV'02)* (Vol. 1, pp. 414–431), London, UK.

Schmid, C., & Mohr, R. (1997). Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*, 530–535.

Schmid, C., Mohr, R., & Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, *37*(2), 151–172.

Se, S., Lowe, D., & Little, J. (2002). Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The International Journal of Robotics Research*, *21*(8), 735–758. doi:10.1177/027836402761412467.

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'06)* (Vol. 1, pp. 519–528). Los Alamitos: IEEE Computer Society. doi:10.1109/CVPR.2006.19.

Shi, J., & Tomasi, C. (1994). Good features to track. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'94)* (pp. 593–600). doi:10.1109/CVPR.1994.323794.

Skrypnyk, I., & Lowe, D. G. (2004). Scene modelling, recognition and tracking with invariant image features. In *Proceedings of the 3rd IEEE and ACM intl. symposium on mixed and augmented reality (ISMAR'04)* (pp. 110–119). doi:10.1109/ISMAR.2004.53.

Taylor, S., Rosten, E., & Drummond, T. (2009). Robust feature matching in 2.3us. In *Workshop, IEEE conference on computer vision and pattern recognition* (pp. 15–22). doi:10.1109/CVPRW.2009.5204314.

Torr, P. H. S., & Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, *78*(1), 138–156. doi:10.1006/cviu.1999.0832.

Trajkovic, M., & Hedley, M. (1998). Fast corner detection. *Image and Vision Computing*, *16*(2), 75–87. doi:10.1016/S0262-8856(97)00056-5.

Tuytelaars, T., & van Gool, L. (2000). Wide baseline stereo matching based on local, affinely invariant regions. In *Proceedings of the British machine vision conference (BMVC'00)* (pp. 412–425).

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'01)* (Vol. 1, p. 511). Los Alamitos: IEEE Computer Society. doi:10.1109/CVPR.2001.990517.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., & Schmalstieg, D. (2008). Pose tracking from natural features on mobile phones. In *Proceedings of the 7th IEEE and ACM intl. symposium on mixed and augmented reality (ISMAR'08)*, Cambridge, UK.

Wagner, D., Schmalstieg, D., & Bischof, H. (2009). Multiple target detection and tracking with guaranteed framerates on mobile phones. In *Proceedings of the 8th IEEE intl. symposium on mixed and augmented reality (ISMAR'09)* (pp. 57–64). doi:10.1109/ISMAR.2009.5336497.

Wagner, D., Mulloni, A., Langlotz, T., & Schmalstieg, D. (2010). Real-time panoramic mapping and tracking on mobile phones. In *Proceedings of the IEEE virtual reality conference (VR'10)*.

Williams, B., Klein, G., & Reid, I. (2007). Real-time SLAM relocalisation. In *Proceedings of the IEEE intl. conference on computer vision (ICCV'07)* (pp. 1–8). doi:10.1109/ICCV.2007.4409115.

Winder, S., & Brown, M. (2007). Learning local image descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'07)* (pp. 1–8). doi:10.1109/CVPR.2007.382971.

Winder, S., Hua, G., & Brown, M. (2009). Picking the best daisy. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR'09)* (pp. 178–185). doi:10.1109/CVPRW.2009.5206839.

Yilmaz, A., Javed, O., & Shah, M. (2006). Object tracking: A survey. *ACM Computing Surveys*, *38*. doi:10.1145/1177352.1177355.

Zhang, Z. (1997). Parameter estimation techniques: a tutorial with application to conic fitting. *Image and Vision Computing*, *15*, 59–76.

Zimmermann, K., Matas, J., & Svoboda, T. (2009). Tracking by an optimal sequence of linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*, 677–692. doi:10.1109/TPAMI.2008.119.