# Interactive Feature Growing for Accurate Object Detection in Megapixel Images

Julius Schöning(✉), Patrick Faion, and Gunther Heidemann

Institute of Cognitive Science, Osnabrück University, Osnabrück, Germany
juschoening@uos.de

**Abstract.** Automatic object detection in megapixel images is quite inaccurate and a time and memory expensive task, even with feature detectors and descriptors like *SIFT*, *SURF*, *ORB*, and *KAZE*. In this paper we propose an interactive feature growing process, which draws on the efficiency of the users' visual system. The performance of the visual system in search tasks is not affected by the pixel density, so the users' gazes are used to boost feature extraction for object detection.

Experimental tests of the interactive feature growing process show an increase of processing speed by 50 % for object detection in 20 megapixel scenes at an object detection rate of 95 %. Based on this method, we discuss the prospects of interactive features, possible use cases and further developments.

**Keywords:** Feature growing · Interactive object detection · Eye tracking · Multivariate detectors

## 1 Introduction

In available datasets [7,8,14,25] the resolution of images for e.g. feature and interest point detection are mostly way below common camera resolution between 5 and 20 megapixel (mp). Nevertheless, feature detection and description are usually key components in modern computer vision application like 3D reconstruction [18,19], object recognition, as well as scene awareness [16,17] and understanding [22]. One common way to start bottom-up analysis of still images is to apply feature detectors, such as *SIFT* [9], *SURF* [3], *ORB* [15], and *KAZE* [2]. For video data, these techniques have been extended with time as an additional dimension, cf. *3D SIFT* [20].

In this paper, we propose to extend available feature detectors and descriptors by integrating multimodal information generated by user interaction. We expect that interactive feature detectors and descriptors yield a minimum of feature points while maximizing information content and robustness. Further, downsampling of images will be avoided and computation time reduced. To get an

**Electronic supplementary material** The online version of this chapter (doi:10.1007/978-3-319-46604-0_39) contains supplementary material, which is available to authorized users.


G. Hua and H. Jégou (Eds.): ECCV 2016 Workshops, Part I, LNCS 9913, pp. 546–556, 2016.
DOI: 10.1007/978-3-319-46604-0_39

idea of the potential of interactive features, we perform object detection in scenes of 19.9 mp (5152 × 3864) resolution. For highlighting the need of high scene resolution, our data set consists of both very small and very large objects within various scenes. Starting with a brief overview of used features, we benchmark the detection rate and processing time of *SURF*, *SIFT*, *ORB*, and *KAZE* features in Sect. 3. Based on the detector with the best detection rate, the interactive feature growing process is introduced in Sect. 4. This guided process uses the effectiveness of the human vision system in search tasks. In experimental tests, we are able show that processing time is halved and detection rate even slightly increases. Finally, we discuss possible applications, improvements and further work.

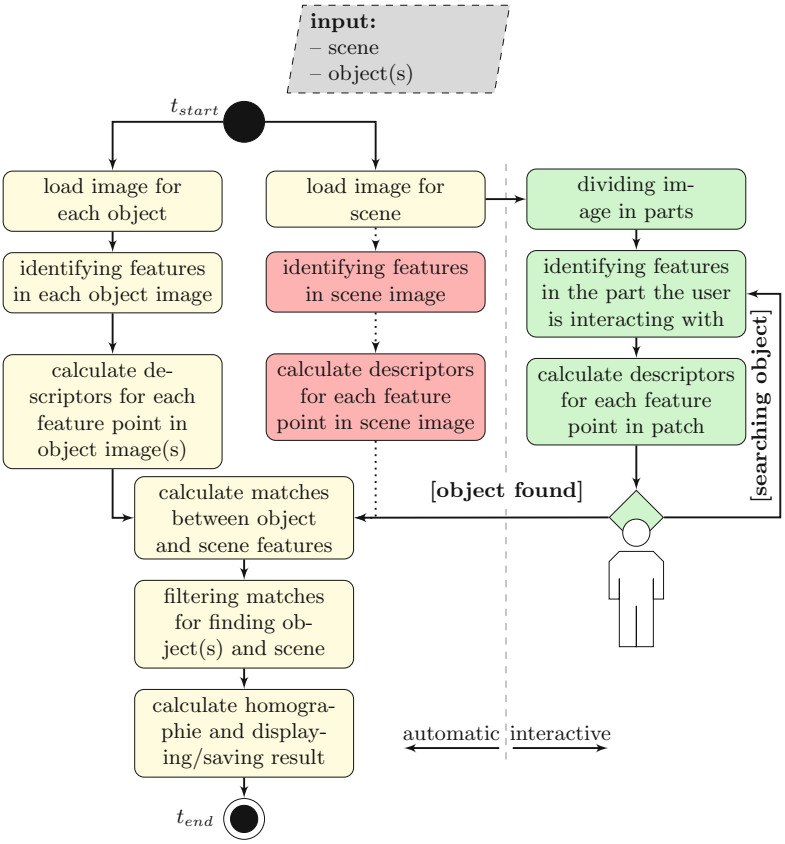



**Fig. 1.** Automatic and interactive object detection process—red blocks: processing time critical path of automatic feature detection and description process, especially, if high resolution scene images are used; green blocks: proposed interactive feature growing using, e.g., users' gaze information. By this additional information generated by user interaction, the number of feature points is reduced. (Color figure online)

## 2 State of the Art

In the following, we briefly summarize the feature processing stages and four common feature detection and description methods, that we used in our experiments.

Object recognition based on local features usually follows three processing stages [21, Chap. 4]:

- detection—identify the location of features,
- description—calculate a quantitative description of every feature,
- matching—find a best match between feature sets of different images, cf. yellow and red blocks of Fig. 1.

In order to integrate the user information into the whole process, one has to start at the detection level already. From there on, any classical combination of detection and description can be used.

Lowe [9] proposed the *Scale-invariant Feature Transform* (*SIFT*). Its basic idea is to detect features as scale-space extrema in a pyramid of difference-of-Gaussian (DoG) filtered versions of the image. The DoG pyramid corresponds to a fast approximation for a Laplacian of Gaussian pyramid, which can be used effectively to detect corners in images. The descriptor algorithm incorporates local image gradients in a neighborhood around the keypoint into 4×4 orientation histograms with 8 orientation bins, leading to a $4 \times 4 \times 8 = 128$ dimensional descriptor. By using the scale space and gradient information, *SIFT* becomes invariant to image scaling and rotation. It is still one of the slowest algorithms for feature detection and description, mostly because the calculation of the DoG pyramid takes much time [13].

*Speeded Up Robust Features* (*SURF*) is another feature proposed by Bay et al. [3]. It also detects features as extrema in scale-space, but uses a fast approximation of the Hessian matrix, which is computed on integral images. For description, the responses of Haar-wavelets are recorded and quantized into 64 dimensions. The *SURF* algorithm needs roughly one third of the time of *SIFT* [3] with comparable detection performance. Still, both methods can only deal with linear illumination changes [13] and are therefore not completely invariant to photometric transformations.

Another robust feature is the *Oriented FAST and Rotated BRIEF* [15] (*ORB*), which combines two existing approaches. It employs the *FAST* detector, which detects points with an intensity difference between the center pixel and surrounding pixels greater than a predefined threshold. For *ORB*, this technique is applied on multiple levels of a scale pyramid. An orientation measure is added, which is lacking in *FAST*. The descriptor works with an orientation invariant extension of the *BRIEF* descriptor, which uses binary tests between pixels in an image patch. In principle, the *ORB* algorithm is not as robust as *SIFT* or *SURF*, but orders of magnitude faster, making it very useful for real-time applications [13].

The most recent method considered here is *KAZE* [2]. The fundamental difference to the previous ones is the computation of a nonlinear scale-space by

diffusion filtering. Here, feature points are detected with Hessian matrices as well. The description is performed with the *M-SURF* descriptor, which is similar to *SURF*, but is adapted to the nonlinear scale space. *KAZE* does not offer much speedup and resides somewhere between *SIFT* and *SURF* in terms of computation time, but is more accurate even under non-linear transformations [2].

## 3 Feature Detectors and Descriptors

Due to the lack of available high-resolution image sets, that contain scenes with both small and large objects, we created our own data set for the evaluation of feature detectors and descriptors.

As shown in Fig. 2, it consists of four scenes—19.9 mp each— and nine objects with a large variation in resolution, scale and orientation. To provide an unbiased benchmark of the detection rate, the *OpenCV* implementation of *SIFT*, *SURF*, *ORB* and *KAZE* were used with their predefined parameters. Special boosted implementations have been deliberately left out. With all aforementioned features, an automatic object detection system corresponding to the yellow and red process blocks of Fig. 1 has been implemented. The feature matching between object and scene is done by a *Fast Library for Approximate Nearest Neighbors* (*FLANN*) [11] based matcher, followed by an outlier filter. Finally, the homography of the object in the scene is calculated and the resulting image is saved.

The random initialization of the algorithms has a minor effect on detecting the object. To reduce this effect, all twelve scene-object detection tasks[1] were repeated ten times.

As shown later in Fig. 3(a), the performance of *KAZE* is significantly better than *SIFT*, *SURF* and *ORB*. This result reflects the findings of Alcantarilla et al. [2]. Despite its nonlinear scale space, *KAZE* is computationally expensive, as shown later in Fig. 3(b). Nevertheless, because of its detection performance and its higher number of found corresponding features, *KAZE* is used for the implementation of our interactive feature growing process.

## 4 Interactive Feature Growing

The main idea of interactive feature growing is to detect and describe features, based on users gazes, during an object search task. Thus, a new time variant dimension—the users' gaze—is considered for the feature calculation, such that the information of the image is enriched by user interaction. Owing the fast response time of subjects in visual search tasks [6,24], our hypothesis is that users' gaze fixations on certain pixels contain information for boosting object detection. In consequence, an error-free object detection with a minimum of user keyboard and mouse interaction, as demonstrated in the supplemental video, can be realized.

[1] detect objects (e)-(h) in scene (a); detect objects (e),(g),(h) in scene (b); detect objects (i),(j) in scene (c); detect objects (k)-(m) in scene (d); cf. Fig. 2.

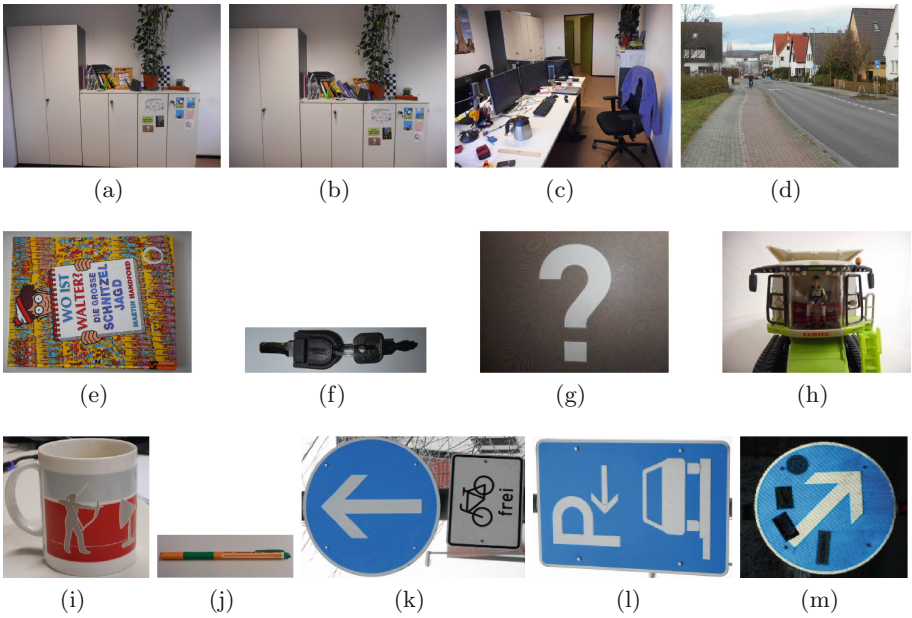



**Fig. 2.** Data set for scene-object detection tasks (See footnote 1) comprising four scene (a) to (d) each at a resolution of 5152 × 3864 pixel (19.9 mp) and nine object (e) to (m) in various resolution, scale and orientation

### 4.1 Process

To realize a first simple, temporal invariant implementation of interactive feature growing, we redesigned the normal automatic object detection process (Fig. 1, left hand side). Therefore, the path which is time critical for high resolution (red blocks) has been substituted by an interactive path (green blocks).

As a preprocessing step, the whole image is divided into a grid of small patches. In the current implementation, a simple rectangular grid is used, but meaningful image segmentation methods [4,5] could also be helpful. When the user interacts with a grid cell, the features in this cell are detected and described. The processing of image parts continues until the user finds the object in the scene. Then, the interactive process turns back to the automatic process and matches the features.

Additionally, after displaying the result, the user can directly decide whether the result is correct. Otherwise, the interactive process can be repeated.

### 4.2 User Interface

Considering user capacity for processing information [10], the user interface (UI) displays a maximum of five elements. Once the application is started, only a fixation element and a separator is displayed. All operating system elements

are also hidden. As soon as the user presses the "start/found object" key, the scene is shown at the position of the fixation element. The object of interest for the detection task is displayed on the upper left. To provide the important instantaneous feedback in about a tenth of a second [12, Chap. 5], processed image parts are grayed out. The users' gaze point is visualized at all times during interactive key point growing. If the object of interest is found by the user, she/he presses the "start/found object" key again. After a brief processing time, the user can evaluate the result and is able to redo the current scene if the matching failed. A demonstration video of the UI is available within the supplemental material of this paper[2].

### 4.3 Implementation

The coding for the prototype and benchmark software was in *C++* with *QT 5.3*, *OpenCV 3.0.0* and *Tobii EyeX SDK 1.6* as third-party components. To achieve a fast responding UI, the *QT* framework in conjunction with *OpenCV* is used. On this basis, a low cost eye-tracking device, *Tobii EyeX* [23], is implemented as singleton to capture the pixel at which the user is looking. To avoid blocking process elements within the UI and gaze capturing, the *KAZE* feature detection and description is encapsulated as an independent thread. This design facilitates a continuous sampling of gaze data, such that the feature growing works even for very small eye movements.

## 5 Experimental Evaluation

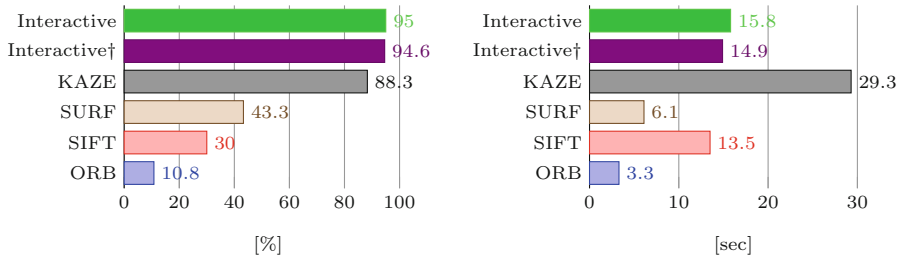
In order to test our hypothesis and evaluating the interactive approach, the following tests were designed and performed.

### 5.1 Setup

As described in Sect. 3, the automatic tests are done with twelve randomized scene-object detection tasks, which were repeated ten times. Based on this procedure, the evaluation of the interactive feature growing was done with ten untrained subjects, using the randomized scene-object detection tasks. Taking a possible learning curve of the untrained subjects into account, the results in Fig. 3, and in the supplementary material contain the measurements with and without (marked with †) the first of twelve tasks.

To embed the human visual system in a natural manner, an eye-tracking device is used for machine interaction. This device requires calibration for every subject. After the calibration with the *Tobii* tool, the subject performs the searching task using our implementation of interactive feature growing. The subjects were told to look at the object and find it in the scene as fast as possible. When they think they found it, they should press the "start/found object" key.

[2] see also https://ikw.uos.de/%7Ecv/publications/EPIC16.




(a) average detection rate over all tasks (b) average processing time over all tasks

**Fig. 3.** Average results over all twelve scene-object detection tasks (See footnote 1). † first task of each subject is excluded with respect to the learning curve.

If the object was not detected correctly in a scene, the subject was able to repeat this scene three times. In this case, the final reported processing time is the sum over all retrials. A chinrest was used during the calibration and performing phase to minimize errors due to head movements of the untrained subjects.

The processing time, in the automatic, as well as in the in interactive case, is defined as difference between the stopping $t_{end}$ and starting point $t_{start}$, cf. Fig. 1. All tests were performed on the same *Intel i7-3770* computer with 24 GB RAM.

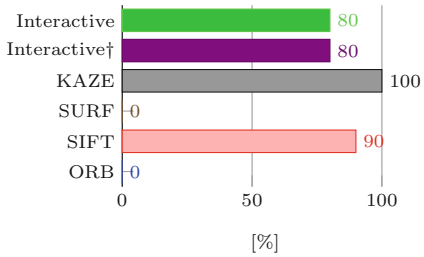
### 5.2 Results

Interactive feature growing outperforms even the best automatic methods *KAZE* with respect to the detection rate, see Fig. 3(a). On average, it has halved the processing time of *KAZE* to a time comparable with *SIFT*, cf. Fig. 3(b). For specific object-scene combinations, our interactive feature growing process is 29 %–69 % faster than *KAZE* with a slightly better detection rate. The detailed results and statistics of each scene object combination are presented in a supplementary pdf document.

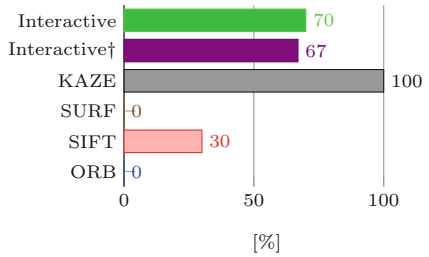
In two object-scene combinations users were not able to get the algorithm to identify the right location, even after three retries. These object-scene combinations are illustrated in Fig. 4 – cf. detailed results Figs. 6 and 7 in supplementary document.

## 6 Discussion

On average, interactive feature growing outperforms the automatic versions of feature detection algorithms with respect to the detection rate. However, some cases are problematic, cf. Fig. 4. By looking at these cases more closely, we recognize that users only fixate their gazes on a small specific area of the object within the scene and do not cover the complete object with their gazes. As consequence, the number of features and descriptors were not sufficient to detect

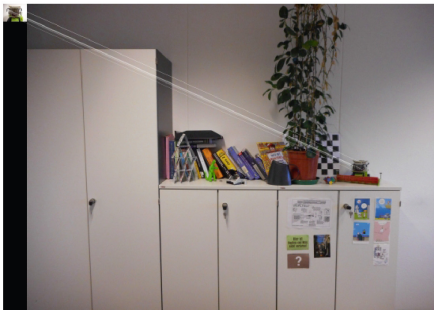



(a) detection rate: detecting Object 2(g) in Scene 2(b)




(b) detection rate: detecting Object 2(h) in Scene 2(b)

(c) exemplary result: detecting Object 2(g) in Scene 2(b)



(d) exemplary result: detecting Object 2(h) in Scene 2(b)

**Fig. 4.** Examples where the matching rate of interactive scene-object detection is below automatic approaches. This effect is caused by the rectangular processing grid.

the object in the scene. In addition, the fragmentation of the object through the grid resulted in even less features in the areas of the grid-lines. In further tests, we were able to reduce this effect by parameterization of the rectangular processing grid. Still, in order to avoid the problem even better, one should apply superpixels or segmentation based processing grids instead. Thus looking at one part of an object would lead to features being computed over the whole area of the object.

Another major point of discussion is why we omitted scaling down the scenes by some factor. Downscaling images until the task is just possible for humans, like it is common for image classification using deep convolutional neural networks, will reduce the processing time of the scene (Fig. 1, red blocks) drastically. But the varying size and resolution of the objects makes it almost impossible to define a scaling factor for the scenes, which works over the whole scene-object detection task. Scaling the images down too much would lead to drastically reduced detection rate for small objects. This behavior might also be an explanation on why *SIFT* does show such a low detection rate for objects with low resolution, like object 2(f), (h), (i), and (k).

Every interactive approach needs a user as operator over the complete processing time, which leads to the issue of use cases for such an expensive

method. A potential use case is the creation of accurate ground truth annotations, which are essential in a large number e.g. for training of deep neuronal networks or of classifier ensembles. Currently, many of these ground truth data sets were created manually or operators review the results of automatic annotations to filter out incorrect data. Another use case can be found in situation where a user already exists, e.g. in research with driver assistance systems. Interactive feature growing could on the one hand boost detection of objects in front of the car, e.g. street signs. On the other hand it could support evaluation of driver behavior, e.g. in assessing if the driver missed some important signs or was distracted by other objects.

Regarding our hypothesis—that users' gaze fixations on certain pixels contains information for boosting object detection—we show in the results, that interactive identification of features can significantly improve the detection rate. By incorporating additional gaze features, like saccades, smooth pursuit, or total time spend on a certain pixel, we assume that interactive feature growing could be improved further. This leads to a more selective identification of features. In the end new feature descriptors using users' gazes could increase the detection rate of interactive feature growing to 100 %.

## 7 Conclusion and Future Work

In this paper, an interactive feature growing process has been presented. With a still naive and not yet optimized prototype of this novel method, the processing time for applying computationally expensive features on high quality images (20 mp) can be drastically reduced. In contrast to automatic methods, only the necessary areas of an image were processed. The decision, which parts of an image are necessary for e.g. a search task, is made by users' conscious and unconscious experiences in visual searching tasks. To support object detection, she/he uses her/his experiences of the real world, which include conceptual knowledge like "a pen is commonly placed on a table".

As further work, we plan to combine interactive feature growing with the FREAK [1] feature descriptor to get a fully bio-inspired system for scene understanding in high resolution images. Further, we want to investigate the impact of different image partition grids and temporal interaction of the user to describe completely new feature types. In addition to still images, the implementation of interactive feature growing for high definition video data is planned to improve our semi-automatic ground truth annotation *iSeg* [16,17].

**Acknowledgments.** This work was funded by German Research Foundation (DFG) as part of the Priority Program "Scalable Visual Analytics" (SPP 1335).

## References

1. Alahi, A., Ortiz, R., Vandergheynst, P.: FREAK: fast retina keypoint. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)

2. Alcantarilla, P.F., Bartoli, A., Davison, A.J.: KAZE features. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7577, pp. 214–227. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33783-3_16
3. Bay, H., Tuytelaars, T., Gool, L.: SURF: speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006). doi:10.1007/11744023_32
4. Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., Van Gool, L.: SEEDS: superpixels extracted via energy-driven sampling. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7578, pp. 13–26. Springer, Heidelberg (2012). doi:10.1007/978-3-642-33786-4_2
5. Chen, H.-P., Shen, X.-J., Long, J.-W.: Histogram-based colour image fuzzy clustering algorithm. Multimedia Tools Appl. **75**, 11417–11432 (2016). doi:10.1007/s11042-015-2860-6
6. Eriksen, C.W., Schultz, D.W.: Information processing in visual search: a continuous flow conception and experimental results. Percept. Psychophys. **25**(4), 249–263 (1979)
7. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results (2012). http://host.robots.ox.ac.uk:8080/pascal/VOC/voc2012/
8. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical report (2007)
9. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004). doi:10.1023/B:VISI.0000029664.99615.94
10. Miller, G.A.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol. Rev. **63**(2), 81 (1956)
11. Muja, M., Lowe, D.G.: Fast approximate nearest neighbors with automatic algorithm configuration. Proc. Int. Conf. Comput. Vis. Theor. Appl. (VISAPP) **2**, 331–340 (2009)
12. Nielsen, J.: Usability Engineering. Morgan Kaufmann, San Francisco (1993)
13. Oliveira, I.O.d., Ono, K.V., Todt, E.: IGFTT: towards an efficient alternative to SIFT and SURF. In: International Conferences in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG), Full Papers Proceedings, pp. 73–80 (2015)
14. Romberg, S., Pueyo, L.G., Lienhart, R., van Zwol, R.: Scalable logo recognition in real-world images. In: Proceedings of the 1st ACM International Conference on Multimedia Retrieval, ICMR 2011, pp. 25: 1–25: 8. ACM, New York (2011)
15. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. In: International Conference on Computer Vision (ICCV) (2011)
16. Schöning, J., Faion, P., Heidemann, G.: Semi-automatic ground truth annotation in videos: an interactive tool for polygon-based object annotation and segmentation. In: Proceedings of the 8th International Conference on Knowledge Capture (K-CAP), K-CAP 2015, pp. 17: 1–17: 4. ACM, New York (2015)
17. Schöning, J., Faion, P., Heidemann, G.: Pixel-wise ground truth annotation in videos - an semi-automatic approach for pixel-wise and semantic object annotation. In: Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods (ICPRAM), pp. 690–697 (2016)
18. Schöning, J., Heidemann, G.: Evaluation of multi-view 3D reconstruction software. In: Azzopardi, G., Petkov, N. (eds.) CAIP 2015. LNCS, vol. 9257, pp. 450–461. Springer, Heidelberg (2015). doi:10.1007/978-3-319-23117-4_39

19. Schöning, J., Heidemann, G.: Interactive 3D modeling - a survey-based perspective on interactive 3D reconstruction. In: Proceedings of the 4th International Conference on Pattern Recognition Applications and Methods (ICPRAM) pp. 289–294 (2015)
20. Scovanner, P., Ali, S., Shah, M.: A 3-dimensional sift descriptor and its application to action recognition. In: Proceedings of the 15th ACM International Conference on Multimedia, MM 2007, pp. 357–360. ACM, New York (2007)
21. Szeliski, R.: Computer Vision: Algorithms and Applications. Springer, London (2011). doi:10.1007/978-1-84882-935-0
22. Tanisaro, P., Schöning, J., Kurzhals, K., Heidemann, G., Weiskopf, D.: Visual analytics for video applications. It-Inf. Technol. **57**, 30–36 (2015)
23. Tobii, A.B.: Tobii EyeX controller (2016). http://www.tobii.com/xperience/products/
24. Trick, L.M., Enns, J.T.: Lifespan changes in attention: the visual search task. Cogn. Dev. **13**(3), 369–386 (1998)
25. Xiao, J., Ehinger, K.A., Hays, J., Torralba, A., Oliva, A.: SUN database: exploring a large collection of scene categories. Int. J. Comput. Vis. **119**(1), 3–22 (2016). doi:10.1007/s11263-014-0748-y