Baltic J. Modern Computing, Vol. 8 (2020), No. 1, 174-181
https://doi.org/10.22364/bjmc.2020.8.1.10

Analysis of Suitable Natural Feature Computer Vision Algorithms for Augmented Reality Services

Arnis CĪRULIS[1], Kristaps BRIGMANIS-BRIĢIS[1], Gatis ZVEJNIEKS[2]

[1]Virtual and Augmented Reality Laboratory, Sociotechnical Systems Engineering Institute of the Vidzeme University of Applied Sciences, Cēsu str. 4, Valmiera, Latvia, LV-4200
[2]SIA Overly, Kr. Valdemara str. 115-5, Riga, Latvia

arnis@va.lv, kristaps.brigmanis@va.lv, gatis@overly.lv

Abstract. The first step in working with object augmentation in an augmented reality system is to identify the target object, so its pose in respect to the camera can be determined for precise and accurate augmented content generation over the target object.
In modern augmented reality systems natural feature detection algorithms are widely used for detecting, identifying and tracking planar textured objects. All-natural feature algorithms detect interest points or keypoints (detector) in an image (scene) and/or calculate descriptors for keypoints (extractor). Algorithms can include both parts, detection and extraction, and can have just one of them realized.
There is a variety of algorithms available nowadays for developers to use. Starting from floating point descriptor-based ones as SIFT and SURF and a row of binary descriptor-based algorithms such as BRIEF, ORB, BRISK, FREAK, KAZE, A-KAZE, LATCH. In addition, there are algorithms which only detect interest points, as FAST or A-GAST. Furthermore, it is possible to use one algorithm for keypoint detection and afterwards use another for descriptor extraction.
Given such a variety of available algorithms, it is necessary to analyse them by understanding their working principles, so they can be classified by their strengths and weaknesses and in what situations the use of one or another algorithm is more appropriate. Since it is possible to use combinations of algorithms, a table of possible cases is provided.
For clarity we must mention that various algorithms, which are not mentioned here, are available but we take an overview of the above listed as all of them are included in the OpenCV library and are widely used in the industry.

Keywords: Augmented Reality, Computer Vision, Natural Feature Detection Algorithms.

1. Introduction

The latest AR solutions use close range solutions utilizing spatial mapping and depth cameras instead of a marker-based approach. A surrounding 3D model is constructed in real time, thereby allowing precise placement of virtual objects at distances from ~10 centimetres to ~5 meters. This approach offers high precision, participant mobility and a high immersion level in typical living room or office conditions (indoor, close range). In the last two years successful devices like Microsoft HoloLens, Lenovo Phab2 and Asus ZenfoneAR were developed. Performance and real time space recognition with depth camera sensors offers possibilities for new and engaging scenarios. Unfortunately, such

devices are too expensive for the typical consumer and end user and we can not talk about involvement of large social groups. This is a disadvantage and partly a reason why Google suspended its Tango project and switched to CoreAR, which uses the standard camera of a smartphone (Kastrenakes, 2017). Marker and image-based AR solutions are still widely used in various areas. High quality cameras and fast processors can provide accurate image recognition and virtual object placement. More and more solutions in AR are offered by computer vision algorithms and we can still be participants in various scenarios, related to architecture, healthcare, logistics, manufacturing, education, marketing, entertainment and other areas. This paper is linked to the collaboration with industry (Ltd. Overly), where the goal was to develop improved image-based marker detection for their services.

2. Importance of computer vision and marked based augmented reality

As stated previously, nowadays computer vision is used in numerous fields and disciplines where benefit can be gained from object detection, classification, tracking, pose estimation etc. in images or video frames.

In industry several software-development kits (SDK) are available for augmented reality solution developers. To well-known and widely used ones “Vuforia”, “Wikitude”, Apple “ARKit” and Google “ARCore” can be enlisted. In augmented reality it is important for the afore-mentioned SDK’s to provide application, image or video frame processing. Another crucial factor is user experience in conjunction with augmented content creation and display to user, as well as interactivity. Here 3D authoring tools, such as “Unity”, can help. All mentioned SDK’s are used as plugins in Unity, providing a powerful complete tool for augmented reality application creation. Still there are some limitations, issues and obstacles that stem from the SDK’s closed code. The main is the inability to make changes in SDK code for more specific individual solutions and blockage of device’s video camera to other applications while using a particular SDK.

There is another approach to use the computer vision open source library “OpenCV” for solving particular application issues related to computer vision. The library has C++ interfaces and is written in optimized C/C++ and thus is platform independent. It consists of more than 2500 optimized algorithms, which include a notable set of classic and state-of-the-art computer vision and machine learning algorithms that can be used to also detect and track objects in an image. As the “OpenCV” library is a BSD license product, it is easy to adopt and to modify the source code. (https://opencv.org/about.html)

3. Analysis of natural feature computer vision algorithms

Natural feature algorithms consist of two main steps, which are the interest point or keypoint detection and keypoint descriptor calculation. Keypoints are distinct points in object texture such as edges, corners, blobs etc., which can be easily located and stand out from rest of the image. Meanwhile a keypoint descriptor characterises it, so a keypoint can be located in an image and distinguished from other keypoints, and, more

importantly, matched to itself in a different image even if transformations like translation, scale changes and rotation are present.

There are numerous natural feature algorithms available for use in modern computer vision applications. These algorithms each implement different approaches and methods for keypoint detection and keypoint descriptor calculation. Besides distinct approaches for keypoint detection we can divide natural feature algorithms into two main groups by the calculated descriptor type used: floating and binary type descriptors.

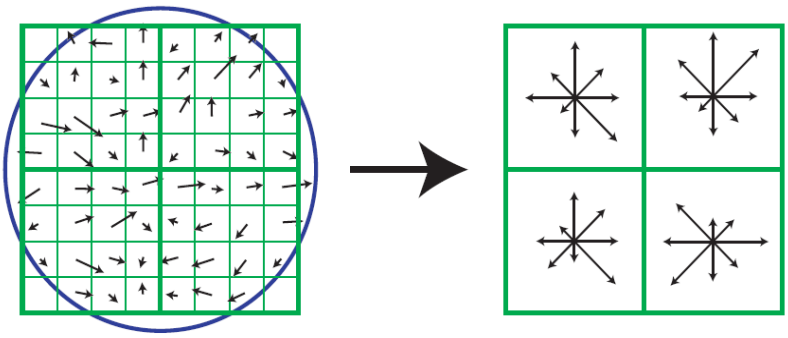
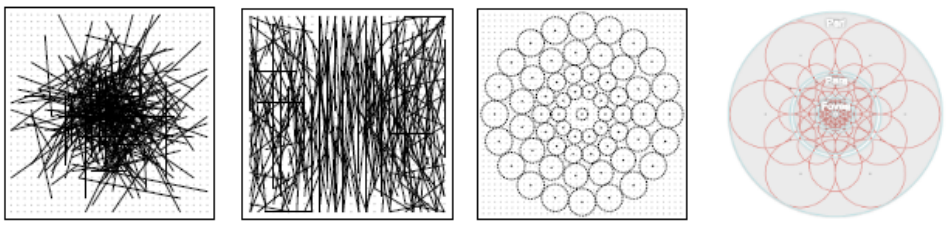


Fig.1. SIFT keypoint descriptor computing from 8x8 set of samples around keypoint. (Lowe, 2004)

One of the most powerful, including scale and rotation invariance and persistency to illumination changes, natural feature algorithm is Scale Invariant Feature Transform (SIFT), introduced in (Lowe, 2004). The SIFT algorithm contains four main steps where at first keypoints are identified in the image by searching for pixels that represent extrema of the Difference-of-Gaussian (DoG) scales-space. In the next step unreliable keypoints are removed, like keypoints with poorly determined location (along edges) or with low contrast. In the third step, orientation is assigned to each keypoint, therefore keypoint descriptors can be represented relative to a keypoint's orientation, and thus a keypoint is invariant to image rotation. As showed in Fig.1, the keypoint descriptor is calculated by computing gradient magnitude and orientation at each image point in a region around the keypoint. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 2x2 sub regions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region (Lowe, 2004).

As the SIFT algorithm is computation expensive Speeded Up Robust Features (SURF) algorithm was introduced in (Bay et al., 2006). SURF detection of keypoints uses the determinant of the Hessian matrix, while the descriptor calculation is done by summing Haar wavelet responses at the regions of interest (Bay et al., 2006). Despite the SURF algorithm being much faster than SIFT, it still is computation expensive and as keypoint descriptors are type of so called floating-point, the same as SIFT, which takes up much memory and lot of computation to compare and match them, these algorithms are not applicable for real time object tracking applications.



(a) (b) (c) (d)

Fig.2. Descriptor calculation pattern of (a) BRIEF, (b) ORB, (c) BRISK, (d) FREAK.

This is where binary descriptor-based natural feature algorithms are useful, as descriptor calculation is much less expensive and, due to its binary nature, they can be more effectively compared and matched. The speed comes at a price, and that is robustness and performance.

One of the first natural feature algorithms using binary descriptors is Binary Robust Independent Elementary Features (BRIEF) introduced in (Calonder et al., 2010). This algorithm does not include a keypoint detection phase, so it can be coupled with any other keypoint detector. Descriptor calculation is based on intensity comparison of random pixel pairs in a patch centred around a keypoint (see Fig.2 (a).). The resulting comparison forms a binary string that can be compared to other descriptors and matched very quickly. Due to its simplicity, BRIEF is not scale and rotation invariant, and that limits its usage in augmented reality applications.

To overcome the lack of BRIEF rotation invariance Oriented FAST and Rotated BRIEF (ORB) was introduced in (Rublee et al., 2011). This algorithm uses modified FAST (Rosten et al., 2010) corner detector for keypoint detection, where Harris corner measure is applied for each keypoint location, thus providing non-maximal suppression within the image. For detection of different size keypoints limited image scale pyramid is used. For keypoint descriptor calculation modified BRIEF is used, where local orientation is computed through the use of an intensity centroid, which is weighted averaging pixel intensities in the local patch which does not correspond to the centre of keypoint. As in BRIEF, descriptor calculation is based on intensity comparison on pixel pairs, and pair selection (see Fig.2 (b)) is constructed by machine learning, maximizing descriptor's variance and minimizing the correlation under various orientation changes (Heinly et al., 2012).

To introduce scale invariance Binary Robust Invariant Scalable (BRISK) algorithm was introduced in (Leutenegger et al., 2011). For keypoint location detection AGAST (Mair et al., 2010) corner detector is used. To add scale invariance, keypoints are detected in a scale-space pyramid, performing non-maxima suppression and interpolation across all scales (Heinly et al., 2012). As for BRIEF and ORB, keypoint descriptor is calculated based on brightness intensity comparison of pixel pairs. Distinct is principle how pixel pairs are selected. A concentric ring-based sampling pattern (see Fig. X (c)) is used to select pixel pairs. Close distance pairs are used for descriptor calculation, long distance pairs used to determine orientation. Due to its complexity compared to BRIEF and ORB, BRISK requires more computation power.

Fast REtinA Keypoint (FREAK) was introduced in (Alahi et al., 2012). This algorithm lacks a feature detection step, so existing keypoint detectors can be used. The author of FREAK suggests using AGAST corner detector. As afore-mentioned binary algorithms, FREAK also computes descriptors based on brightness intensity between pixel pairs. The pixel sampling pattern (see Fig.2 (d)) is inspired by the human retina so that sampling points represent centres of receptive fields.

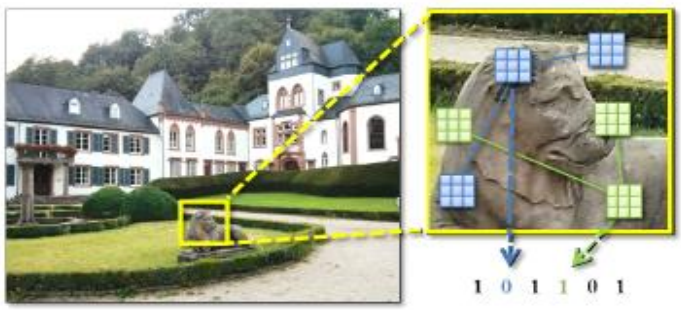


Fig. 3. Visualization of LATCH descriptor calculation with three-pixel patches in region of keypoint.

Learned Arrangements of Three Patch Codes (LATCH) was introduced in (Levi et al., 2016). As BRIEF and FREAK, LATCH also does not have its own keypoint detection, so existing keypoint detectors have to be used. Instead of comparing pixel pairs, for descriptor calculation, LATCH compares triplets of patches in region of keypoint (see Fig.3). In each triple of patches one patch is marked as 'anchor', the other patches are marked as 'companions'. Similarity between "anchor" and "companions" is calculated by the sum-of-squared differences (SSD). If the first "companion" is more similar to the "anchor" than the second, then "1" is written in the resulting bit, otherwise a "0" is entered.

Accelerated KAZE (A-KAZE) algorithm was introduced in (Alcantarilla et al, 2013) to provide robust feature detection and descriptor extraction. For keypoint detection, A-KAZE uses fast explicit diffusion schemes for building nonlinear scale space considering anisotropic diffusion. As for descriptor calculation, A-KAZE uses Modified-Local Difference Binary (M-LDB) that uses gradient and intensity information from the nonlinear scale space.

In Table 1. a summary of above listed natural feature algorithms is provided.

Table 1. Summary of natural feature algorithm properties

Algorithm	FD, DE	Descriptor type	Rotation invariance	Scale invariance	Patented
SIFT	FD, DE	Floating	Yes	Yes	Yes
SURF	FD, DE	Floating	Yes	Yes	Yes
BRIEF	DE	Binary	No	No	No
ORB	FD, DE	Binary	Yes	No	No
BRISK	FD, DE	Binary	Yes	Yes	No
FREAK	DE	Binary	Yes	Yes	No
A-KAZE	FD, DE	Binary	Yes	Yes	No
LATCH	DE	Binary	Yes	Yes	No
FD – feature detection DE – feature descriptor extraction (calculation)					

4. Suitable algorithm approbation in open source software library

The aim is to create a software module which is platform independent, performs object detection and tracking, does not block device's video camera and is coded using "OpenCV" library so it is possible to modify the source code.

The above-listed prerequisites require creation of a library written in C++, which provides platform independence and can be implemented in any 3D authoring tool. Device's camera is controlled from a 3D authoring tool, so the camera is freely available

for the developer to do any other manipulations with its input, also frames from camera input can be passed to the module for processing at any time.

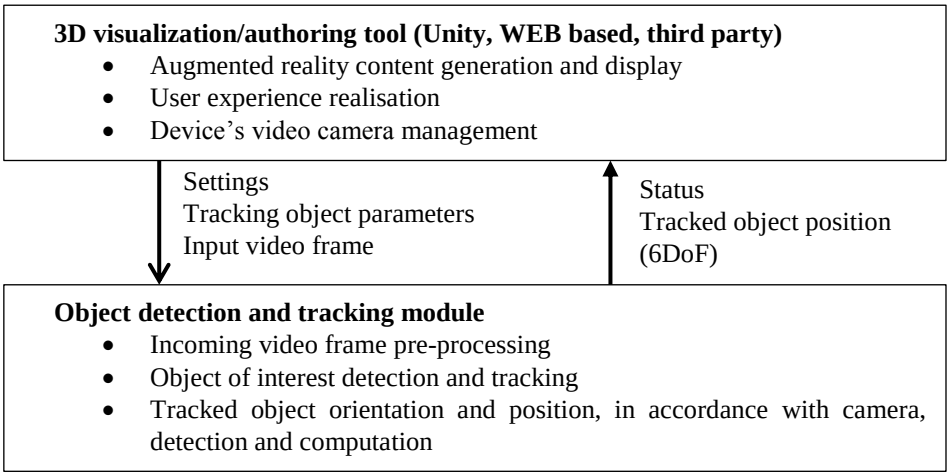


Fig. 4. Conceptual schematics of module and authoring tool interaction

The graphical interface of the 3D authoring tool and developed module interaction is shown in Fig. 4. The 3D authoring tool must provide initial settings for the tracking module and tracked object parameters, which can be in the form of a tracked object image or XML file with tracked object data, such as dimensions, keypoints and descriptors. For every frame where object detection and tracking is necessary, the input frame from a device's video camera must be forwarded to the tracking module where object detection, tracking and pose estimation is performed. As a result, the tracking module returns its status to the 3D authoring tool, whether an object has been detected. If yes, object position in reference to the device's camera is returned. Object position is formed in 6 Degrees of Freedom, which includes object translation in frame by x, y, z axis and objects rotation by x, y, z axis in reference to the video camera of the device.

In the tracking module tracking will be separated in two cases, initial object detection and continuous object tracking (see Fig. 5).

First is initial object detection, which is performed if an object has not been detected in previous frames. In this case a more precise and robust natural feature tracking algorithm is used to detect an object in a scene. After initial object detection, separate interest points are calculated, which are used in later frames for continuous object tracking using a modified iterative version of the Lucas-Kanade optical flow in pyramids (Bouguet, 2001).

If object initial detection has been performed, then in continuous frames interest point motion in reference to the previous frame is tracked using a sparse optical flow algorithm. Next, the faster but not as precise natural feature algorithm is used to detect the tracking object in the input frame region tracked by the sparse optical flow algorithm, thus processing just part of the input frame and improving execution time. As in the first case, separate interest points are calculated for next frame optical flow tracking.

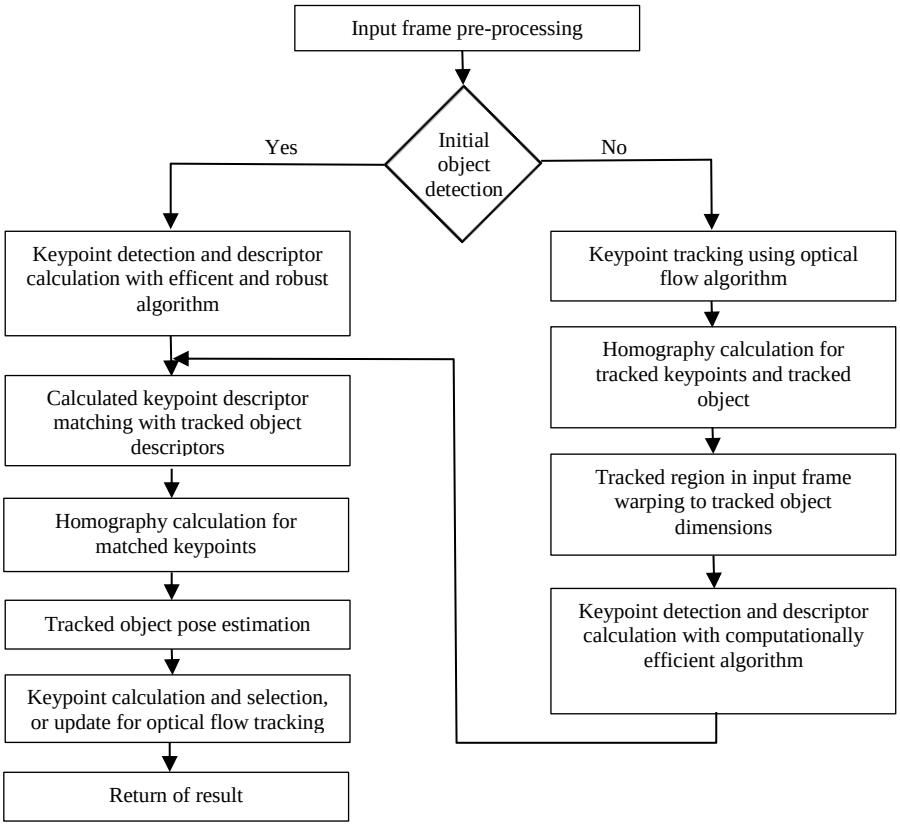


Fig. 5. Tracking module conceptual processing schematics.

5. Conclusions

The approach offered in this paper provides a functional software module for various AR implementations. This module is platform and hardware independent. It can run on Windows, Android and Apple computers. Compared to other solutions, this software does not block access to the video camera and it is accessible for other applications as well. To develop this marker recognition module, function separation is considered; it means that this module only takes responsibility for object detection and tracking. The module is open source and it uses only open source libraries.

Future challenges include improvement of tracking stability and persistence, collision detection among many markers in image database, performance improvements on mobile devices, simultaneous tracking of several objects.

References

Alahi, A., Ortiz, R., Vandergheynst, P. (2012).Freak: Fast retina keypoint. In Proc. IEEE.Conf. Comput. Vision Pattern Recognition, pp. 510–517.

Alcantarilla, P. F., Nuevo, J., Bartoli, A. (2013). Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces, Bristol, UK.

Bay, H., Tuytelaars, T., Van Gool, L. (2006). "SURF: Speeded up robust features," in Proceedings of the European Conference on Computer Vision, pp. 404–417..

Bouguet, J. Y. (2001). "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm", Intel Corporation, Vol. 1, no. 2, pp. 1-9.

Calonder, M., Lepetit, V., Strecha, C., Fua, P. (2010). Brief: Binary robust independent elementary features. In European Conf. Comput. Vision, pp. 778–792. Springer.

Heinly, J., Dunn, E., Frahm, J.M. (2012). Comparative evaluation of binary features. In: Proceedings of the European Conference on Computer Vision, Springer, pp. 759–773.

Kastrenakes, J. (2017). Google's Project Tango is shutting down because ARCore is already here, Available at: *https://www.theverge.com/2017/12/15/16782556/project-tango-google-shutting-down-arcore-augmented-reality*

Leutenegger, S., Chli, M., Siegwart, R. Y. (2011). Brisk: Binary robust invariant scalable keypoints. In Proc. IEEE Int. Conf. Comput. Vision, pp. 2548–2555.

Levi, G., Hassner, T. (2016). LATCH: Learned Arrangements of Three Patch Codes, IEEE Winter Conference on Applications of Computer Vision (WACV).

Lowe, D. G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints" *IJCV*, vol. 60, no. 2, pp. 91–110.

Mair, E., Hager, G. D., Burschka, D., Suppa, M., Hirzinger, G. (2010). Adaptive and generic corner detection based on the accelerated segment test, Proceedings of the 11th European conference on Computer vision: Part II.

Rosten, E., Porter, R., Drummond, T. (2010)."Faster and better: A machine learning approach to corner detection," IEEE Trans. Pattern Anal. Mach. Intell, vol. 32, no. 1, pp. 105-119.

Rublee, E., Rabaud, V., Konolige, K., Bradski. G. (2011). Orb: an efficient alternative to sift or surf. In Proc. IEEE Int. Conf. Comput. Vision, pp. 2564–2571.

Received February 28, 2020, accepted March 17, 2020