Acta Polytechnica CTU Proceedings **2**:8–14, 2015
doi:10.14311/APP.2015.1.0008

available online at http://ojs.cvut.cz/ojs/index.php/app

ON FPGA BASED ACCELERATION OF IMAGE PROCESSING IN MOBILE ROBOTICS

Petr Čížek*, Jan Faigl

Department of Computer Science, Faculty of Electrical Engineering, CTU in Prague, Technická 2, 166 27 Prague, Czech Republic

* corresponding author: petr.cizek@fel.cvut.cz

Abstract. In visual navigation tasks, a lack of the computational resources is one of the main limitations of micro robotic platforms to be deployed in autonomous missions. It is because the most of nowadays techniques of visual navigation relies on a detection of salient points that is computationally very demanding. In this paper, an FPGA assisted acceleration of image processing is considered to overcome limitations of computational resources available on-board and to enable high processing speeds while it may lower the power consumption of the system. The paper reports on performance evaluation of the CPU–based and FPGA–based implementations of a visual teach-and-repeat navigation system based on detection and tracking of the FAST image salient points. The results indicate that even a computationally efficient FAST algorithm can benefit from a parallel (low–cost) FPGA–based implementation that has a competitive processing time but more importantly it is a more power efficient.

Keywords: FPGA, system–on–chip, image processing, FAST feature detector, visual navigation.

1. Introduction

One of the key indicators of the intelligent behaviour of a mobile robot is its ability to react promptly in complex situations and make fast and correct decisions regarding the robot goals. On micro-robotic platforms, this is a very comprehensive task due to limited computational resources. A typical example of micro-robotic platforms are micro aerial vehicles (MAVs) [1], small legged robots [2] or robots used to study swarm intelligence [3]. They can be characterized by constrained dimensions and payload, which is directly related to the limited battery capacity. Altogether, these parameters determine an available computational power on-board of the mobile robot. A direct trade-off between the computational capabilities and power consumption can be identified, characterizing the maximum time for which the robot can perform its mission.

This trade-off between computationally demanding and power efficient decision-making techniques is especially recognizable in navigation algorithms that rely on computer vision methods based on processing of a large amount of visual data. Salient point extractors are popular techniques of machine perception in the mobile robot navigation tasks; however, not all available approaches are currently suitable for micro-robotic platforms due to the limited on-board computational resources. Moreover, a practical deployment of a mobile robot always demands a real-time performance of decision-making algorithms, which imposes further restrictions on the applicable approaches regarding the available resources.

Two fundamental approaches how to deal with limited computational resources on micro-robotic platforms can be considered. The first approach is a simplification of the computationally demanding processing, e.g., by introducing approximations of the demanding method and simplifying the whole principle, which can be a daunting task as it may not be always possible. The second approach is a utilization of the optimized implementations for newly available features of the modern processors, e.g., a dedicated optimization for special (typically SIMD–single instruction multiple data) instructions of the new CPUs. Besides, dedicated co-processors can be utilized to speedup the computationally demanding calculations by massively parallel processing that is available using conventional graphics cards, e.g., using CUDA or OpenCL. Although modern graphics cards are able to provide superior peak performance, they also have high power consumption requirements (in order of hundreds of Watts) and thus they are not suitable for micro-robotic platforms.

Therefore, as an alternative solution for parallel based and power-efficient computations, it is a more suitable to consider Field-Programmable Gate Array (FPGA) technology to develop a custom architecture specifically designed for the particular computational task, which, in the end, can be very power and computationally efficient while also small in dimensions and cheap in development.

However, FPGA-based solutions need a significantly different approach for an efficient implementation of the particular algorithms than a conventional CPU. Thus, the deployment time can be high compared to a gain from the implementation of a custom design computational architecture for the FPGA.

In this paper, we report our results on a comparison

of the selected image processing techniques suitable for embedded computation of the visual navigation task implemented on a conventional on-board platform and a dedicated solution based on FPGA processing of the image data. In particular, we consider a detection of the salient points in the image. The salient points are image patterns which differs from their local neighbourhood and are expected to be reliably and repeatably detectable, preferably invariant to camera viewpoint changes. Such features provide a mobile robot with a limited set of environmental anchor points which are then utilized for a vision-based navigation.

Feature based methods consist of three stages. The first stage is a detection of features to identify salient points in the image. Then, for each detected feature its descriptor is calculated to describe the local image surrounding of the feature in order to distinguish individual features. The third stage is a process of establishing feature correspondences which is called feature matching. The matching is based on a comparison of the feature descriptors to determine whether the features correspond to the same salient object already detected in the environment.

Regarding the computational complexity of the feature detection, we can classify the features to artificial, like blobs or patterns, and naturally occurring in the environment. For the field robotics, natural landmarks are more important; however, artificial landmarks are much easier to detect. One of the foundational feature detectors is the Scale-invariant feature transform (SIFT) algorithm [4], which is unfortunately computationally very demanding, and therefore, a Speeded-up robust features (SURF) algorithm has been proposed [5] to provide salient points with less computational requirements.

Although SURF is less demanding than SIFT, it can be still computationally restrictive on small platforms. That is why researchers investigate other methods how to detect and describe salient objects in the environment by a computationally efficient algorithm such as the FAST feature detector [6]. FAST is widely adopted by robotic community for its low computational complexity. It has been deployed in several visual navigation methods, e.g., [7, 8], and it also provides a base for different feature extractors [9, 10].

Following our intention to have a power and computationally efficient image processing available on a mobile robot, we consider FAST as a suitable algorithm for visual navigation task computed on-board. Therefore, we compare a performance of the FAST feature extraction implemented on a conventional embedded platform with a CPU based on an ARM architecture and a custom implementation of the FAST detector on an FPGA-based dedicated co-processor. The comparison indicates that even a computationally efficient feature extraction based on the FAST algorithm can benefit from parallel FPGA-based implementation that has a competitive processing time and it is a more power efficient.

FIGURE 1. Hexapod walking robot platform

The paper is organized as follows. An overview of the most related reports on enhanced computationally and power efficient vision-based navigation approaches based on FPGA co-processor is presented in Section 2. The proposed evaluation of the FPGA-based image processing is considered in the context of a monocular vision-based teach-and-repeat autonomous navigation [11] that is briefly described in Section 4 together with the main idea of the FAST feature detection and the selected BRIEF feature descriptor [12]. Results on the evaluation of the computational burden reduction using an embedded platform of the hexapod walking robot (see Figure 1) and FPGA–based solution are presented in Section 5. Concluding remarks and future work is dedicated to conclusion in Section 6.

2. Related work

This paper is focused on FPGA-based implementations of computer vision techniques that are utilized in mobile robot navigation tasks. Several approaches have been published and their authors reported improvements of the computational and power efficiency in vision-based navigation tasks by a dedicated implementation on FPGA.

Probably one of the first FPGA based implementation of the SURF feature extraction has been introduced in [13], where an embedded module capable of image processing suitable for mobile robotics is presented. The module implements a SURF feature extraction [14] from images with the resolution of 1024×768 pixels. The reported frame rate is 15 fps, which can be considered low in a comparison to the 24 fps GPU-based implementation of SURF; however, the power consumption is only 6 Watts. Notice, the FPGA implementation outperforms a pure CPU implementation running on an Intel Atom dual-core processor, which is able to process only a single frame per second.

Another FPGA-based module for stereo image processing has been recently introduced by the ETH

Computer Vision and Geometry group [15]. It is based on the FPGA and ARM Cortex A9 quad-core CPU and calculates dense disparity images with the resolution of 752×480 pixels at the 60 frames per second. The disparity estimation relies on the semi global matching approach, which is computationally a very intensive task. The reported power consumption of the module is 5 Watts. In [16], the same authors extend their module by implementing a reactive based collision avoidance method for MAVs and tested it in an outdoor environment.

Authors of [17] presented a miniature module with a low-cost FPGA and MCU for visual navigation of MAVs. They utilize a reduced version of the PTAM algorithm [8] (mapping maximum of 200 features due to memory limitations) for an estimation of the visual odometry. Their approach is based on the FPGA implementation of the FAST features detection and BRIEF [12] feature description. The MCU is utilized for visual odometry calculation and the whole system operates at 30 frames per second on images with the resolution of 160×120 pixels.

In [18], authors consider the FPGA to synchronize data from the IMU and FAST feature extraction from a camera stereo pair for a robust inertial assisted real-time visual SLAM. The reported update rate is about 20 frames per second for images with the resolution of 752×480 pixels. Similar results with a slightly different approach are presented in [19].

An embedded FPGA–based computer vision module has been presented in [20]. The proposed FPGA architecture uses an efficient pipelining for the FAST feature detection in a video stream with the resolution of 752×480 pixels at the frame rate of 60 frames per second, which is utilized in the visual teach–and–repeat navigation method.

Regarding the physical dimensions of the aforementioned modules, they are all based on system–on–chip (SoC) solutions on FPGA development boards which (in all cases) do not exceed 12 cm × 8 cm. The power consumption of these modules is less than 10 Watts while all of them exhibit a real-time performance in the particular robotic navigation task. The SoC architecture combines the advantages of the FPGA with a regular CPU in order to reduce deployment costs and improve performance of the system. Its main aspects are described in Section 3.

All of the discussed approaches seem to be a suitable choice for deployment on a micro-robotic platform. However, the most promising are the approaches ([17–20]) based on the FAST feature detector [6] because this detector is widely adopted by the robotic community due to its low computational complexity. Hence, it represents a base for further methods. For example, it is a source of the parallel tracking and mapping (PTAM) method [8] that can be considered as a visual monocular odometry (or SLAM – simultaneous localization and mapping) technique, also suitable for MAVs, albeit the original PTAM method was designed to be used in small augmented reality workspaces. The method relies on tracking and mapping of FAST features for the 6DOF position estimation in the environment.

FAST has been recently utilized in the semi-direct monocular visual odometry [7] in which FAST features are tracked and used to build a map that is further used for a visual odometry estimation. The algorithm was tested on MAV equipped with the Odroid-U2 ARM quad-core computer achieving computational speed of 50 frames per second. Despite of the CPU implementation, the reported power consumption of the system is 10 Watts, which is a competitive to the one of the first FPGA module for the SURF detection introduced in [13].

Therefore, we consider the FAST features detection on the ARM–based computer and FPGA–based implementation [20] in this evaluation study of the potential benefits of the parallel (FPGA–based) implementation of the image processing for visual navigation.

3. Processor centric design

A combination of the FPGA and CPU allows to accelerate a computer vision application and also allows to carry out more complex tasks by an improved system. The main idea of combining these two types of computational architectures is the processor centric system–on–chip co-design, which allows to exploit benefits of both architectures. Generally, a CPU is a more suitable for general purpose computations and it is also a more easily programmable. However, a parallel nature of the FPGA fabric makes it well suitable for accelerating demanding or repetitive computational operations performed by the CPU. Besides, it is also suitable for online signal processing and sensory interfacing [21].

There are two principal ways how to incorporate both the FPGA and CPU in a single embedded design. It is possible to have both of them soldered on the printed circuit board and utilize a communication channel like SPI for a fast data transmission between them. However, much more efficient is to utilize the system-on-chip (SoC) solution, where the CPU is a part of the FPGA chip. The CPU can be either softcore or hardcore. The FPGA fabric is extremely versatile, and therefore, it can implement even a CPU in its fabric, which is called a softcore CPU. On the other hand, manufacturers started to incorporate the CPU in the design of the chip to further exploit the FPGA–CPU co-design possibilities. In that case, the CPU is etched into the silicon of the chip together with the FPGA fabric and such a solution is then referenced as the hardcore CPU.

Nowadays hardcore processors are usually based on multi-core ARM architecture and they are running at the units of GHz. In the case of the softcore processors, their speed is limited to around 200 MHz due to the FPGA fabric limitations. The main advantage of the both hardcore and softcore processors is that they

can be directly programmed in the C programming language. In addition, they can host a real-time operating system, which makes their programming for time critical applications even simpler.

From the processor point of view, the FPGA fabric can act either as a memory mapped slave device or it can be directly connected to the CPU core and accelerates the processor instructions. In that case, the instruction set of the CPU is extended by custom designed instructions.

The main advantage of FPGA–based SoC systems is their versatility and reconfigurability which may be of a particular use in the field of mobile robotics. The programmer can choose components which assemble the system according to the needs of the particular application. The FPGA resources allow to build both internal computational units and IO communication interfaces, which can be directly connected to the main CPU. Moreover, FPGA always posses a lot of general purpose IO pins which can be used for any communication not exceeding maximum frequency of the given FPGA chip (usually about 300∼350 MHz) regardless the communication, which can be serial like SPI and I^2C or parallel. In addition, specialized hard-core bus endpoints for a high-speed communication are common integral parts of nowadays FPGAs, e.g., Gigahertz serial endpoints like PCIe, SATA or parallel DRAM interfaces.

4. Bearing-only Visual Navigation

The navigation algorithm used for the evaluation of implementations of the FAST feature detector on the CPU and FPGA is based on the teach-and-repeat navigation algorithm proposed in [11]. The navigation is based on tracking previously mapped visual features that are used for a bearing-only navigation. The approach relies on the relative localization provided by the odometry, which would eventually integrate unbounded position error over time. However, the approach considers the odometry only locally for traversing a short straight line segment along which detected visual landmarks are utilized to correct the robot heading and thus suppresses the odometric error. The corrections of the heading are based on a modus of the horizontal displacements of the tentative correspondences between previously mapped and the currently perceived visual features. The modus is found by a histogram voting as it is visualized in Figure 2.

The tentative correspondences are established using the FAST feature detector and the BRIEF feature descriptor [12] which are briefly described in the following subsections.

4.1. Features from Accelerated Segment Test

The FAST feature detector [6] belongs to the family of the so called appearance based detectors because it

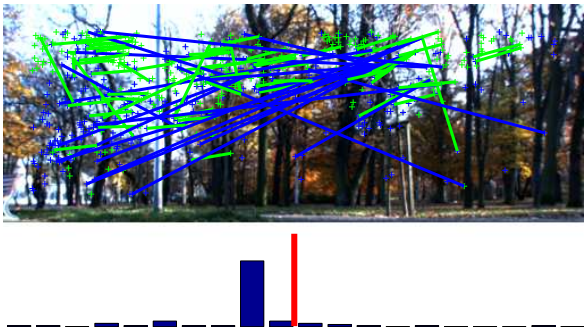

Figure 2. SURFnav navigation algorithm. Matched features from the current and previously learned image. The corresponding navigation histogram is depicted at the bottom.

searches the image for corner-like structures. The detector is optimized with respect to the computational complexity. The algorithm uses a set of comparisons between the center pixel p and pixels defined on the 7×7 neighbourhood using the circle in the image determined by Bresenham's circle algorithm, which is shown in Figure 3.

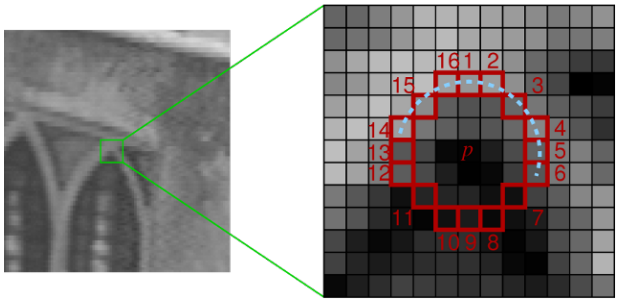

Figure 3. FAST feature detection. Courtesy of [6].

At first, the pixels on the circle are labelled dark or bright depending on their relative brightness to the central pixel. The candidate pixel is considered as a feature if there are n contiguous bright or dark pixels in the circle. For $n \geq 12$ it is possible to use a rapid rejection method for a faster outlier rejection which leads to a common setting of $n = 12$. However, the original paper [6] approved that the best repeatability is exhibited for the setting of $n = 9$. The corner score, which is necessary for the non-maxima suppression, is calculated as the sum of the absolute differences between the pixels in the contiguous arc and the central pixel.

The FPGA architecture presented in [20] uses efficient pipelining for the FAST feature detection acceleration. The image data are processed as an online stream and the architecture performs all the steps of the algorithm, i.e., the corner detection, corner score calculation, and non-maxima suppression, in parallel as data are transmitted from the image sensor. The architecture also allows to use different values of n without an impact on the computational speed or used

resources.

4.2. Binary robust independent elementary features

Once features are detected, individual salient points are characterized by descriptors. The purpose of the descriptor is to describe the image neighbourhood of the point and thus characterize the salient object of the environment. In the proposed evaluation, the selected descriptor is the BRIEF descriptor which stands to the binary feature descriptor [12]. It is based on pairwise intensity comparisons of pixels inside an image patch surrounding the located feature. These comparisons form a set of unique binary tests which are subsequently stored into a q-dimensional bit vector. The pairwise comparisons can be chosen either randomly or evolutionary, e.g., they can be trained by methods of reinforcement learning [22], which optimize the descriptor for the particular environment. Typically used values of q are 64, 128, and 256 bits that correspond to the BRIEF8, BRIEF16, and BRIEF32 variants of the feature descriptor, respectively.

The descriptor similarity is evaluated using the Hamming distance that computes how many bits of two given feature vectors are different.

5. Evaluation

The main motivation behind the utilisation of FPGA–based systems in visual navigation tasks is a latency reduction in the robotic navigation system which is especially recognizable in experiments performed in dynamic or confined environments, which impose imminent threats to the mobile robotic platform [1, 23] and the reduction of the computational load thus power consumption of the mobile robot platforms.

This section presents the results of the experimental evaluation of the impact of the FPGA–based co-processor utilization on the reduction of the computational burden of the feature extraction process in the monocular vision–based teach–and–repeat autonomous navigation. The particular parts of the whole navigation system under the evaluation are the feature extraction chain and the computationally most demanding parts of the navigation algorithm: the feature detection and description; features matching to the map previously created in the teach mode; construction of the navigational histogram; and determination of the robot heading correction based on the maximum peak in the histogram voting method.

For the feature extraction the FAST feature detector [6] is used with the setting of $n = 12$, while 256 bit long BRIEF [12] (BRIEF32) is utilized as the feature descriptor. The number of features is set to approx. 200 features per image and the pre-learned map in the navigation pipeline evaluation contains 200 features.

The pure CPU based implementation running on the Odroid U3 board [24] was compared to the

(a) . Outdoor urban env. (b) . Indoor lab env.

Figure 4. Testing environments.

FPGA–based SoC implementation on Terasic DE0-nano board [25]. In particular, the hardware and software used in the evaluation are as follows:

Odroid U3 – A small embedded micro-computer suitable for robotic applications. It features 1.7 GHz ARM Cortex A9 quad-core microprocessor (Samsung Exynos4412 Prime) running the Ubuntu 12.04 Linux operating system. The attached camera is the Mobius ActionCam with the resolution of 640×480 with 60 fps. The implementation of the navigation algorithm was based on the Open Computer Vision library [26] (OpenCV 2.4.9) implementations of the FAST feature detector and BRIEF32 descriptor.

DE0-nano board – An embedded module with the Altera Cyclone IV EP4CE22 FPGA equipped with the APTINA MT9V034 grayscale camera sensor with the global shutter and the resolution of 752×480 providing images with 60 fps. The implementation of the navigation algorithm follows the approach described in [20]. It utilizes the bare-metal programmed NIOS IIe softcore processor clocked at 150 MHz for the feature description, matching, and histogram voting and a dedicated FPGA–based parallel co-processor for the FAST feature detection.

The proposed evaluation consists of the focused examination of the feature detector and descriptor and examination of the performance of the whole navigation system. The tests were performed in field conditions both in outdoor urban (Fig. 4a) and indoor lab (Fig. 4b) environments.

First, we evaluated the performance of the FPGA–based and CPU–based implementations of the FAST feature detector and computation of the BRIEF descriptor. Two criteria are considered in this evaluation: correctness of the FPGA–based implementation and computational requirements.

Regarding the correctness of the FPGA–based implementation, it provides identical results to the CPU–based counterpart for the detected features and also the same descriptors. This test was performed on artificially generated data in order to ensure the correctness of the results. The data consists of a checkerboard pattern with grayscale gradient which allows to detect various number of corners depending on the predefined threshold.

Concerning the computational requirements of the

feature extractor, the Odroid implementation needs (in average) 17.07 ms for approximately 200 FAST feature detections with BRIEF32 descriptors in an image with the resolution of 640×480 pixels. The DE0-nano implementation exhibits similar performance with the processing time of 16.91 ms for 200 feature detections and descriptions.

The navigation stack performance benchmarking consists of the feature extraction, feature matching against the pre-learned map, and the histogram voting method to establish the heading correction of the robot. The achieved results are summarized in Table 1.

Platform	**Odroid U3**	**DE0-nano**
CPU cores	4	1
CPU clock	1.7 GHz	150 MHz
FPGA	-	22320**
FPGA usage	-	28%
Feature extraction	17.07 ms	16.91 ms
Navigation pipeline	35.89 ms	24.38 ms
Power consumption	8.13 W	1.82 W

** The number of available logic elements.

TABLE 1. Navigation pipeline processing results of Odroid U3 and DE0-nano SoC platforms

5.1. Discussion

The results presented in Table 1 indicate that the FPGA–based SoC approach does not significantly reduce the required time for feature extraction; however, the power consumption is reduced by 4.5 times magnitude in comparison to the purely CPU–based implementation.

Regarding the update rate of individual methods the FPGA–based implementation is capable to process each other image in the 60 fps camera stream, which gives about 30 processed images per second while the CPU–based implementation can process 20 frames per second if we consider a uniform image retrieval; otherwise the update rate of the CPU–based implementation is 27 fps but images are dropped at non-deterministic way which leads to uneven distribution of information in time.

The FAST feature detector of the DE0-nano implementation performs feature detection online on the stream of camera data, which implies that the image coordinates of all salient points in the image are known during the readout of a single image from the camera sensor. The feature description and matching to the pre-learned map is performed in an instant when the feature is detected in the camera stream. However, 256 bit long BRIEF32 descriptor calculation together with matching to approx. 200 features from pre-learned map on the 150 MHz single core CPU makes it impossible to finish all the computations during the readout of one single image. Therefore, each other frame in a 60 Hz camera stream is skipped, which allows the algorithm to finish all the calculations in less than 32 ms during the readout of the next image.

6. Conclusion

In this paper, we consider a problem of decreasing the computational load of the embedded computational resources utilized in the vision-based navigation tasks. The presented results show that although the processing times of both the FPGA–based and CPU–based implementations are almost similar most-likely due to the overall low complexity of the used feature extraction, the FPGA–based system–on–chip design can significantly reduce the power consumption of the embedded processor in comparison to a purely CPU–based solution.

In order to further verify and quantify the results, we plan to thoroughly benchmark each part of the visual navigation algorithm stack and incorporate the FPGA-based image processing in a more complex navigation task of a visual SLAM, where the computational power of the FPGA platform can be of potential benefit.

Acknowledgements

The presented work has been supported by the Czech Science Foundation (GAČR) under research project No. 13-18316P.

References

[1] N. Michael, D. Mellinger, Q. Lindsey, V. Kumar. The grasp multiple micro-uav testbed. *Robotics Automation Magazine, IEEE* **17**(3):56–65, 2010. DOI:10.1109/MRA.2010.937855.

[2] D. Belter, P. Skrzypczyński, K. Walas, D. Wlodkowic. Affordable Multi-legged Robots for Research and STEM Education: A Case Study of Design and Technological Aspects. In *Progress in Automation, Robotics and Measuring Techniques*, vol. 351 of *Advances in Intelligent Systems and Computing*, pp. 23–34. Springer International Publishing, 2015. DOI:10.1007/978-3-319-15847-1_3.

[3] F. Arvin, J. Murray, C. Zhang, et al. Colias: an autonomous micro robot for swarm robotic applications. *International Journal of Advanced Robotic Systems* **11**(113):1–10, 2014. DOI:10.5772/58730.

[4] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **60**(2):91–110, 2004. DOI:10.1023/B:VISI.0000029664.99615.94.

[5] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding* **110**(3):346–359, 2008. DOI:10.1016/j.cviu.2007.09.014.

[6] Rosten, Edward and Drummond, Tom. Machine learning for high-speed corner detection. In *ECCV 2006*, vol. 1, pp. 430–443. DOI:10.1007/11744023_34.

[7] C. Forster, M. Pizzoli, D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *ICRA 2014*, pp. 15–22. DOI:10.1109/ICRA.2014.6906584.

[8] G. Klein, D. Murray. Parallel Tracking and Mapping for Small AR Workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pp. 225–234. DOI:10.1109/ISMAR.2007.4538852.

[9] S. Leutenegger, M. Chli, R. Siegwart. BRISK: Binary Robust invariant scalable keypoints. In *ICCV 2011*, pp. 2548–2555. 2011. DOI:10.1109/ICCV.2011.6126542.

[10] E. Rublee, V. Rabaud, K. Konolige, G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV 2011*, pp. 2564–2571. DOI:10.1109/ICCV.2011.6126544.

[11] T. Krajník, J. Faigl, M. Vonásek, V. Kulich, et al. Simple yet Stable Bearing-only Navigation. *Journal of Field Robotics* 2010. DOI:10.1002/rob.20354.

[12] M. Calonder, V. Lepetit, C. Strecha, P. Fua. BRIEF: binary robust independent elementary features. In *ECCV 2010*, pp. 778–792. Springer. DOI:10.1007/978-3-642-15561-1_56.

[13] T. Krajník, J. Šváb, S. Pedre, et al. FPGA-based module for SURF extraction. *Machine vision and applications* **25**(3):787–800, 2014. DOI:10.1007/s00138-014-0599-0.

[14] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool. Speeded-up robust features (SURF). *Computer vision and image understanding* **110**(3):346–359, 2008. DOI:10.1016/j.cviu.2007.09.014.

[15] D. Honegger, H. Oleynikova, M. Pollefeys. Real-time and Low Latency Embedded Computer Vision Hardware Based on a Combination of FPGA and Mobile CPU. *IROS 2014* DOI:10.1109/IROS.2014.6943263.

[16] H. Oleynikova, D. Honegger, M. Pollefeys. Reactive Avoidance Using Embedded Stereo Vision for MAV Flight. In *ICRA 2015*, pp. 50–56. IEEE. DOI:10.1109/ICRA.2015.7138979.

[17] R. Konomura, K. Hori. Visual 3D self localization with 8 gram circuit board for very compact and fully autonomous unmanned aerial vehicles. In *ICRA 2014*, pp. 5215–5220. DOI:10.1109/ICRA.2014.6907625.

[18] J. Nikolic, J. Rehder, M. Burri, et al. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In *ICRA 2014*, pp. 431–437. DOI:10.1109/ICRA.2014.6906892.

[19] G. Zhou, J. Ye, W. Ren, et al. On-board inertial-assisted visual odometer on an embedded system. In *ICRA 2014*, pp. 2602–2608. DOI:10.1109/ICRA.2014.6907232.

[20] P. Čížek. *Embedded module for image processing.* Master's thesis, CTU, Faculty of Electrical Engineering, 2015.

[21] García, Gabriel J and Jara, Carlos A and Pomares, Jorge and Alabdo, Aiman and Poggi, Lucas M and Torres, Fernando. A survey on FPGA-Based sensor systems: towards intelligent and reconfigurable low-power sensors for computer vision, control and signal processing. *Sensors* **14**(4):6247–6278, 2014. DOI:10.3390/s140406247.

[22] T. Krajník, P. deCristoforis, M. Nitsche, et al. Image features and seasons revisited. *ECMR 2015 – to appear* .

[23] S. Lupashin, M. Hehn, M. W. Mueller, et al. A platform for aerial robotics research and demonstration: The flying machine arena. *Mechatronics* **24**(1):41–54, 2014. DOI:10.1016/j.mechatronics.2013.11.006.

[24] Collective of authors. Hardkernel co., Ltd.@ONLINE. `http://www.hardkernel.com`, cited on 2015/08/17.

[25] Collective of authors. Terasic, Inc.@ONLINE. `http://www.terasic.com.tw`, cited on 2015/08/17.

[26] Bradski, G. and Kaebler, A. *Computer vision with the OpenCV library.* O'Reilly Media, 2008.