J Real-Time Image Proc
DOI 10.1007/s11554-014-0449-3

SPECIAL ISSUE PAPER

A real-time multi-camera vision system for UAV collision warning and navigation

Ákos Zarándy · Mate Nemeth · Zoltan Nagy · Andras Kiss · Levente Santha · Tamás Zsedrovits

Received: 1 April 2014 / Accepted: 19 August 2014

Abstract A real-time vision system with multiple cameras was developed for UAV collision warning and visual navigation for fixed-wing small- or medium-sized aircrafts. The embedded vision system simultaneously acquires images using five cameras, stores, and evaluates the visual data with an FPGA-based multi-core processor system. The system was designed to fulfill the strict size, power, and weight requirements arising from UAV on-board restrictions. The hardware parameters of the system and the performance of the algorithm are compared to the state of the art.

Keywords Real-time vision system · Embedded system · UAV · Multi-camera · FPGA · Multi-core processing · Collision warning · Visual navigation

1 Introduction

Unmanned aerial vehicle (UAV) technology has reached an advanced level, which enables them technically to fly autonomously a predefined path and complete different missions. However, legally they are not allowed to fly fully autonomously, since flight authorities identified various safety shortcomings [1]. One of the problems is that they are not robust enough due to the lack of on-board sensor and actuator redundancies. Another missing capability is the collision avoidance [2–5], because the GPS-based control and navigation system drives the UAV practically blindly, hence it can collide with any other aircraft, or with any stationary object, which is not correctly on the map (new building, antenna tower, pillar of a bridge, crane, ski lift, etc.). The vision system, introduced in this paper, was designed to be able to identify and track a remote aircraft in mid-air and measure its size. Moreover, it can calculate the ego-motion of the UAV by tracking multiple stationary points on the ground.

One can ask, why vision is used, rather than GPS-based active radio transponders (e.g., Traffic Alert and Collision Avoidance System, TCAS) or radars. The answer is that due to the equivalent level of safety (ELOS) principle the probability of a mid-air collisions should be as low as 10^{-9} per flight hour, which assumes the usage of a layered collision avoidance approach, meaning that some of these systems should be simultaneously used, depending on the size of the UAV and on the used air segment.

After the need for vision-based collision avoidance system became obvious, several teams have launched projects for building visual remote aircraft sensors [2, 6–8]. However, as it turns out from the literature, these teams have been designing algorithms on PCs, and even the tests are conducted on PCs, which are carried by manned flights, or off-line. In contrast, the vision-based remote aircraft detector system introduced here is designed for medium- or small-sized (<20 kg) fixed-wing UAVs flying in open airspace rather than navigating between tall buildings in urban areas.

An on-board vision system for a medium- or small-sized UAV should fulfill numerous tough specification criteria. Its resolution and field of view (FOV) should be high enough to identify intruder aircraft from large distance; it should be able to perform real-time processing; its size,

Á. Zarándy (✉) · M. Nemeth · Z. Nagy · A. Kiss · L. Santha
Institute for Computer Science and Control, 13-17 Kende Street, Budapest H-1111, Hungary
e-mail: zarandy.akos@sztaki.mta.hu

Á. Zarándy · M. Nemeth · Z. Nagy · A. Kiss · T. Zsedrovits
Pazmany Peter Catholic University, Budapest, Hungary

weight, and power consumption parameters should satisfy on-board UAV operation requirements; and finally, it should be affordable. From a functionality point of view, it is expected (1) to detect intruder aircraft, which is on a collision course; (2) calculate the attitude (Euler angles: yaw, pitch, and roll) of the aircraft by calculating the differential orientation changes between consecutive frames; and (3) store all the acquired images in full resolution for archiving and for off-line testing purposes.

The paper is organized in the following way: first the state of the art and the related work in this field is described (Sect. 2), and comparison with other similar systems is provided. Then, system specifications are given in Sect. 3. After that, the vision-based remote aircraft detector system is described in Sect. 4. In Sect. 5, the algorithmic components are described, while in Sect. 6, the multi-core processor array implementation is shown.

2 Related work

Naturally, avoiding mid-air collisions is not a new problem. Traditionally, there are two different approaches to address airborne collision avoidance. The first assumes cooperation among the airplanes. In this case, each aircraft transmits its position, velocity, and planned route, and based on a predefined protocol they avoid approaching each other. The current system in use in air traffic management is called TCAS (traffic collision avoidance system) [9]. A next generation of it, called ADS-B (automatic dependent surveillance-broadcast) is currently being introduced, and will be mandatory in most of larger aircraft from 2020 [10]. Though cooperative approaches are relatively simple, and does not require sensing of remote aircrafts, the US and European agencies require having a non-cooperating solution on board as well.

Modern large airliners utilize sophisticated radar and computer systems, which identify the position and the velocity of intruder aircrafts, warn the pilot if they are on a collision course, and even make an avoidance maneuver automatically if the pilot does not react. However, this solution cannot be applied to small aircraft due to economic and weight considerations.

For large UAVs, sensor fusion is a commonly used approach, to make the collision avoidance system operational in all flight conditions. The system, described in [11], is based on pulsed Ka-band radar, two kinds of visible cameras, two IR cameras, and two PCs. For small UAVs vision only systems are currently developed in different places. One is described in [6], in which a single 1024×768 resolution camera and a full-sized PC with GPU is used to identify the intruder. Multiple systems can be found in the literature which applies different kinds of miniature PCs [12, 13]. In paper [12], an embedded PC with a 1.6 GHz Intel Atom processor-based UAV vision system is described. It is equipped with a 752×480 (WVGA) camera pair, and the system can calculate visual inertial state estimation with 20 Hz. In paper [13], a pxCOME industrial board with Intel core 2 Duo is described. It weights 230 g, consumes 27 W, and serves 2 stereo camera pairs with 752×480 resolution each. It can calculate localization, pattern recognition, obstacle avoidance. Papers [14, 15] describe FPGA-based real-time vision system for stereo vision applications. Both of them apply an Intel Atom card as well, to implement post-processing and decision making. The first system is specially designed for a UAV to perform visual/inertial sensor fusion, while the second is a more general image processing system. Both of the systems use camera pairs, and they can reach 20 and 60 Hz frame rate, respectively. The system, introduced in the paper, can process the images of five pieces of WVGA or 1280×960 cameras, by utilizing only a single FPGA. As it is shown in Table 1, the weight and power consumption data of our systems are better.

The system, described in [16], is heavier and more power hungry; however, algorithmically it is the most similar to our system, because it is dedicated for remote aircraft detection as well. The detection results comparison is described in Sect. 5.1, where the performance of our algorithm is analyzed.

3 System specification

The vision system has two important roles, namely the collision warning and the attitude data calculation. From these, the more demanding task from image acquisition (resolution and view angle) point of view is the collision warning, because detection of potentially dangerous intruder aircraft in time requires the permanent monitoring of the field of view of $220° \times 70°$ in front of the UAV [17, 18] with high resolution as it is shown below. Figure 1 illustrates the requirements of a safe avoidance. According to the flight safety requirements [17], there should be a certain separation volume around each and every aircraft, in which no other aircraft can be. The size of the separation volume (separation minima) depends on the particular air segment and on the type of the aircraft.

To be able to avoid the separation minima, the intruder should be detected from a distance, which is not smaller than the traffic avoidance threshold. If the intruder is not detected before crossing the traffic avoidance threshold, but detected before the collision avoidance threshold, the collision can be still avoided. For human pilots, 12.5 s response time is recommended to safely avoid a collision [3]. Since there are now standardized requirements, so far for UAVs, the goal is to reach the human safety

Table 1 Comparison of different vision systems

	[12]	[13]	[14]	[15]	[16]	System 1	System 2
Weight (cameras + proc. system)	~150 g	~230 g	~250 g	~250 g	~450 g	138 g	226 g
Power	6 W	27 W	~10 W	11 W	59 W	3 W	14 W
Number of cameras	2	4	2	2	1	5	5
Resolution of the cameras	752 × 480	752 × 480	752 × 480	5 megapixel	1024 × 768	752 × 480	1280 × 960
Frame rate	20 Hz	60 Hz	20 Hz	resolution dependent	11 Hz	20 Hz	20 Hz
Functionalities	Stereo vision, visual state estimation	Stereo vision, obstacle avoidance	Stereo vision/ inertial visual navigation	Stereo vision/general purpose	Distant airplane detection	Distant airplane detection, navigation	Distant airplane detection, navigation
Video storage	–	–	–	–	SSD	SSD	SSD
Processor	Intel Atom	Intel core 2 Duo	Spartan 6 + Intel Atom	Spartan 3A XC351800A, + Intel Atom	Intel core 2 Duo + Geforce 9600GT	Spartan 6 lx45t	Spartan 6 lx150t

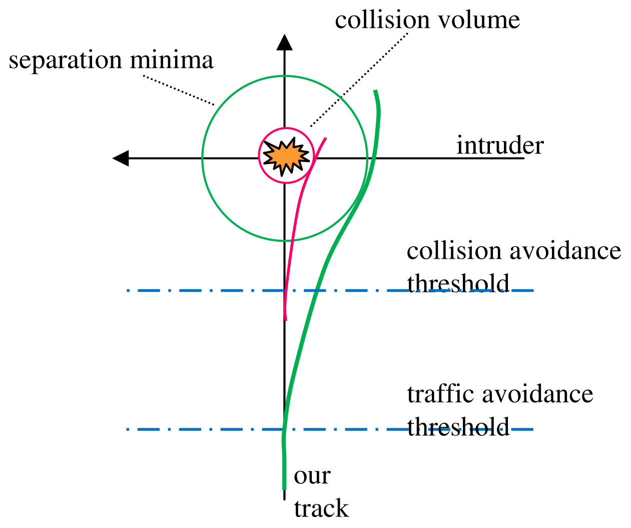

Fig. 1 Traffic (*green*) and collision (*magenta*) avoidance courses

requirements, that is to follow the "equivalent level of safety" principle. Naturally, to avoid scaring the pilots and the passengers of the other aircraft, and to increase the safety level, earlier initialization of the avoidance maneuver is required, which certainly assumes earlier detection. Since the tracks of the small- and medium-sized UAVs do not interfere with airliners, or high-speed jets, it has to be prepared for other UAVs and smaller manned airplanes, like the Cessna 172. This means that the maximal joint approaching speed is 100 m/s; therefore, we need to detect them from 1,250 m (12.5 s before collision), to be able to safely avoid them. In these cases, the radius of the separation minima is 660 m (~2000 ft) and of the collision volume is 160 m (~500 ft).

According to the experiments, in order to perform robust visual detection of an aircraft, it should be at least 3–4 pixels large in the captured image. For a single engine land class aircraft (Cessna 172), with 10 m wingspan and length, a minimum 0.1 degree/pixel resolution is required. This means an overall minimum 2200 × 700 pixel resolution. (A 10 m-sized object is 3 pixels large from 1,909 m with this resolution.)

Another important system requirement is the speed. The control system is expecting 20 navigation parameter updates in a second; therefore, the frame rate should be a minimum 20FPS. Naturally, the image processing part of the vision system should be able to perform the complete processing at this speed also.

For real-time attitude calculation, the same resolution and speed, but smaller FOV is satisfactory. Therefore, the system with the above specification can calculate the angular changes of the aircraft orientation as well.

The system should be able to fit and operate on a UAV platform, which introduces strong limitations to its size, weight, and power parameters. The target was to fit the device to a medium-sized fixed-wing UAV with a 3 m wingspan that limits the weight to a maximum of 300 g. Another important requirement is that the vision and storage system should be resonance tolerant.

4 System description

In this section first, the selection of the main components are described, then the system architecture, the interconnections of the components, the power distribution, and the system integration are shown.

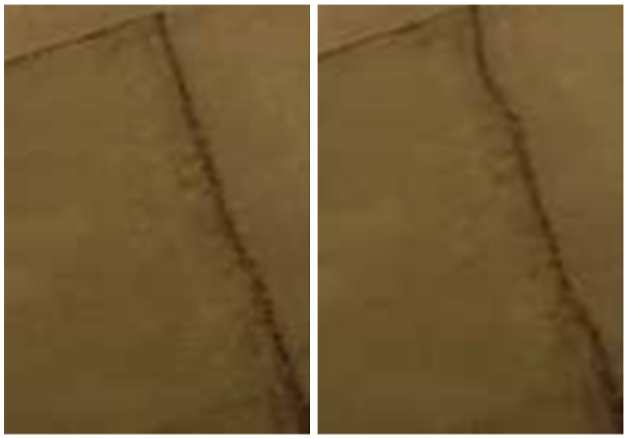

Fig. 2 Aerial frames captured with rolling shutter camera from a vibrating UAV platform. The straight edge of the airfield is strongly distorted due to the vibration of the camera

4.1 Camera

The key component of a special purpose vision system is the camera. During the design phase, one has to consider different types of camera. One would think that the most straightforward solution would be to use one piece of high resolution (like 2,500 × 1,000) camera with low distortion ultra-wide angle optics. However, the problem with this setup is that the size and the weight of the camera and especially the ultra-wide view angle optics are beyond the acceptable limits. Therefore, multiple small cameras were applied. Three different classes of cameras have been studied:

1. Micro-cameras with integrated lenses (mobile phone class);
2. Miniature cameras with S-mount (M12) lenses;
3. Small industrial cameras with C-mount or CS-mount lenses.

One important criterion is that the camera should have a global shutter. A rolling shutter camera does not provide a geometrically coherent image on a vibrating platform, because different horizontal bands of the images are captured in different time instants with different camera axes. Figure 2 shows the distortions on a few consecutive frames captured with a rolling shutter camera from the UAV platform showing the corner of the rectangle-shaped airfield.

Another important criterion is that the cameras should be triggerable, because the images have to be captured in known time instances. The remote aircraft positions derived from the camera images are relative to the UAV, and the Inertial Measurement Unit (IMU) contains the exact position and attitude of the UAV, hence the synchronization of these units is vital.

4.1.1 Micro-cameras

The advantage of micro-cameras is that they are inexpensive, they have sufficient pixel resolution (up to 13 megapixels; e.g., Sony Exmor module [19]), and they are ultra-compact, ultra light (~3–5 g), and low power. However, the price of miniaturization is the rolling shutter operation. The reason is that the global shutter sensors have larger pixel pitch; hence, the sensor diameter is larger that calls for large optics. Moreover, the integrated optics are not replaceable, and its quality is sometimes poor. Though some of them, like the one in the iPhone5, have surprisingly high angular resolution with the five pieces of aspheric lens system (0.05°/pixel). Moreover, the problem of these miniaturized cameras is that it is practically impossible to buy them in low quantity, it is very difficult to find the manuals, and the largest problem is, that they become obsolete in less than two years, practically by the end of the design cycle. These cameras are usually not externally triggerable. They have parallel raw data digital output.

4.1.2 Miniature cameras with S-mount (M12) lenses

Miniature cameras are good candidates for low volume, low weight applications. In this camera class, the optics are replaceable, and one can find high resolution (megapixel) lightweight (~5 g) optics [20] for them with different viewing angles. The resolution of the rolling shutter versions are far beyond 5 megapixels, while the global shutter imagers have lower resolutions, like WVGA [21] or 1.2 Megapixels [22]. Here the typical power consumption is less than 200 mW, and the weight is around 10 g including optics. These cameras are usually externally triggerable. They have either parallel raw data or USB output formats.

4.1.3 Small industrial cameras with C-mount or CS-mount lenses

There are a very large number of cameras in this class. One can find them in different resolution (from VGA up to 8 megapixels), size (from 3 × 3 × 3 cm), weight (from 40 g), and both rolling and global shutter types [23]. However, here the weight of a precision lens is significant (60–200 g) [24]; hence, the overall weight is above 100 g. This weight is much larger than the cameras in the second category, but on the other hand, the precision of the lens, the optical alignment, and the overall image quality is much better. The power consumption of these cameras is in the watt range rather than a few hundreds of milliwatts. These cameras are externally triggerable, and they have typically GigE, USB 2, USB 3, Camera Link, or FireWire (IEEE 1394) output formats.

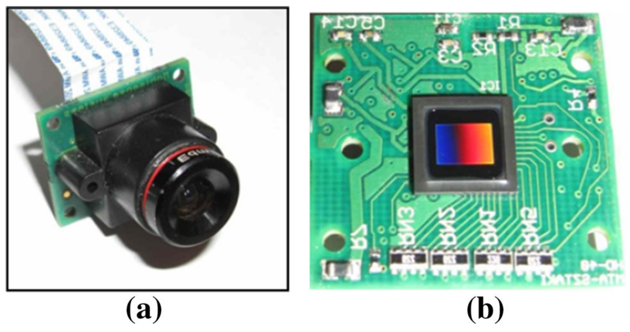

Fig. 3 The off-the-shelf WVGA camera module (**a**) and the 1.2 Mpixel camera module, designed by the authors (**b**)

4.1.4 Selection

Since triggerable camera with global shutter sensor is needed, cameras only from the second or the third category can be selected. The second category makes it possible to build a vision system for medium-sized UAVs, where the weight of the vision system should not exceed 300 g.

The other selection criterion was the data interface. We wanted to avoid GigE and USB camera, because they have relatively large power consumption. Moreover, embedded systems do not have multiple of these interfaces, and it is difficult to implement them due to their complex protocols.

Five pieces of WVGA (752×480) cameras (MBSV034M-FFC from Mobisens) [21] (Fig. 3a) have been selected to cover the required resolution with necessary overlap. For this 1/3 inch camera module, a 3.66 mm focal length High Resolution Infinite Conjugate μ-Video^{TM} Imaging Lenses from Edmund Optics was chosen[24].

Besides this off-the-shelf camera module, we have also designed an advanced one (Fig. 3b), which uses a 1280×960 sized Aptina sensor (MT9M021). The advanced feature of our camera module design is that it applies the DS90UB913Q serializer on the camera module. This Texas Instruments chip was designed directly for the serialization of digital megapixel camera output data. It enables the use of two wires for transmitting all the necessary signals: the power, the digital video stream, the trigger, the reset, and the I^{2}C bus, which is needed for setting up the sensor chip. Using this new camera module, the number of the required cameras to reach the same performance can be reduced, which simplifies the system. Moreover, since the serial cables can be up to a couple of meters long, the processor and the sensors can be placed to arbitrary positions, which relaxes the placement constraints of the system on the UAV. This is a critical issue due to weight balancing considerations on a UAV.

4.2 Processor selection

Nowadays high-performance image processing platforms are based either on GPUs, DSPs, or FPGAs. In the case of strict power, weight, and size budgets, the power hungry GPU platforms with their heavy cooling radiators cannot be an option, even though there have been some platforms already developed for military UAVs [25].

By comparing DSPs and FPGAs, the DSPs are more flexible, and their programming time is much shorter; however, the processing performance of FPGAs is much higher. Since a five-camera data acquisition, processing and storing system requires high computational speed and flexible data communication channels, the selection of the FPGA solution was the best choice. A small form factor FPGA board with a Spartan 6 FPGA (EXPARTAN-6T, [26]) have been selected, which has enough user IO ports to collect the data from the five cameras, and has SATA interface to save the acquired image flows. This FPGA board with the WVGA cameras is the computational platform of the first vision system version (System 1).

Major FPGA vendors recently introduced a new class of devices specially aiming reconfigurable SoC systems where a powerful embedded ARM microprocessor and an FPGA fabric is implemented on the same die. The Xilinx Zynq architecture contains a Dual ARM subsystem called Processing System (PS) and a Programmable Logic (PL) part. The PS part contains two ARM Cortex-A9 microprocessors, L1 an L2 cache memories, an on-chip SRAM memory, DMA engines and several peripheral controllers such as I^{2}C, SPI, CAN, USB, Ethernet, and memory interfaces. The PL part is based on the Xilinx Atrix or Kintex architecture. The two parts are connected by several AXI4 compatible interfaces optimized for both low latency access of the configuration registers of the peripherals and high throughput memory accesses.

System level tasks in the system, such as I^{2}C communication, programming of the DMA engines etc., can be performed efficiently using a conventional microprocessor such as the Microblaze soft processor core from Xilinx, which can be implemented easily in Spartan 6 FPGAs. Unfortunately, this soft processor core is implemented using the general logic resources of the FPGA reducing area available for the computationally intensive image processing tasks; moreover, its computational performance is very limited due to its ~100 MHz clock frequency. The Zynq architecture provides a good solution for this problem. The existing image processing blocks designed for the Spartan-6, which use the same AXI interconnect system, can be migrated to the new system. The hard ARM processor cores can operate on nearly an order higher clock frequency than the Microblaze processors (800 MHz), while its power consumption remains in the same range.

The more powerful processors enable us to run more complex algorithms on the recorded image flow, which means that the less computationally intensive image processing tasks can be implemented on the ARMs, which saves significant development time, and makes the system more flexible. The short term goal is to replace the Spartan 6 with the Zynq FPGAs, and the WVGA cameras with the 1.2 megapixel ones (System 2).

4.3 Data storage unit

In an airborne application where vibration is a critical issue, data storage can be implemented using different types of flash memory devices, such as memory cards, USB sticks, or solid state drives (SSD). In the discussed application, the data rate to be saved is $5 \times 752 \times 480 \times 20 = 36$ Mbyte/sec (2.1 Gbyte/min) raw data, assuming five cameras, WVGA image size, and 20 fps. Even though data compression is a widely used option for image storage, this is not a viable option in the proposed application, as very small remote objects need to be identified, thus, the artifacts introduced by the compression become intolerable.

Therefore, we needed a device that can cope with 36 Mbyte/sec data flow, which is far beyond the write speed of an SD card (2–10 Mbyte/sec) or even USB stick (4–25 Mbyte/sec). Moreover, up to 20 min of flight data during a test data acquisition flight have to be stored. Hence, 45 Gbyte data storage space is needed. This fits to a small sized SSD (64 Gbyte). The system enables easy up scaling, since SSDs go up to 600 Gbyte. Therefore, it is possible to save a couple of test flights onto a single SSD.

Since the cameras are controlled by the FPGA and the image capturing and saving has to be synchronized to each other, an efficient SATA core needed to be implemented on the FPGA. There were several criteria for this interface:

- compatibility with all the SATA devices;
- capability to handle image flows coming from up to five cameras;
- adjustable storage frame size on the SSD to be able to tune it to input image sizes;
- continuous data saving capability;
- several test flights need to be stored one after the other without overwriting each other.

In our implementation, the camera interface converts the parallel data to an AXI Stream structure. Then, a stream gatherer unit merges the data streams coming from the five cameras into one single stream. This is done via line-by-line, concatenating the synchronously arriving lines coming from the synchronized cameras. The size of the merged frames is: number of cameras × camera width × camera height. The ends of theses elongated frames are labeled with end of frame flags.

The implemented SATA core has an AXI Stream input also. This is useful, if some parts of the results of the data stream are meant to be stored in addition to the raw camera images. The SATA core implements the Command, Transport, and Link Layers of the SATA protocol and provides a Physical Layer Wrapper for the GTX transceivers. The Physical Layer Wrapper also includes an Out of Band Signaling (OOB) controller state machine, which deals with initialization and synchronization of the SATA link. The core can interface with SATA 2 style hard drives as well as Flash-based solid state drives or even older SATA 1st gen. hard drives.

The SATA core provides a simple interface to issue READ/WRITE sector commands. Its data interface is organized like a 32-bit FIFO. The output data is delivered at a speed of 4 bytes at 75 MHz (user output clock) for a theoretical peak bandwidth of 300 MB/s (SATA 2), which is enough for storing the raw data of the cameras even with reserve.

The developed SATA core receives a WRITE command, which has a physical address in the SSD and a data package size. Equation (1) shows the calculation method of the maximum frames to be stored in one data package.

$$\mathrm{fr} = \frac{\mathrm{max\_sector\_num} \times \mathrm{sector\_size}}{\mathrm{img\_height} \times \mathrm{img\_width} \times \mathrm{camera\_num}} \tag{1}$$

where

the max_sector_num is 65,535,

the sector size is 512 byte

the img_height and img_width are the parameters of one camera.

The camera_num and the frame size also determine the maximum sector number to use. This restriction is applied because a write instruction must be finished by the end of a concatenated frame. To begin a new instruction, a new sector address has to be set and the SATA core needs to wait for the "ready for new command" message from the SSD. This preparation can be done within the blank period of a frame, however, not in the middle of a frame. The number of WRITE commands should be specified by the *command number* parameter. All the above/required parameters can be programmed via the processor core (MicroBlaze or ARM processor).

The SATA core can perform READ instructions as well. This can be useful, if the captured images need to be displayed or used as an input data source for off-line testing. Before performing a READ instruction, the initialization of the sector address and the sector number must be set correctly. The data can be streamed through a HDMI port to a display or can be sent to a host PC via Ethernet, or directly processed by the FPGA.

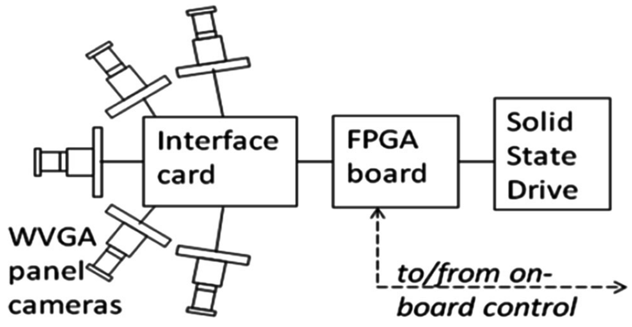

Fig. 4 The block diagram of the vision system

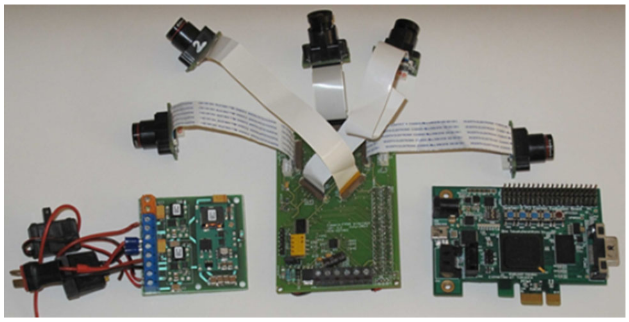

Fig. 5 Photo of the components of the vision system. (*Left to right* power module, interface board, FPGA board)

4.4 System architecture and interconnections

The block diagram of the System 1 with the WVGA camera modules is shown in Fig. 4. It contains off-the-shelf components (the cameras, the FPGA board, and the SSD), and a custom-designed interface card. The cameras and the interface card are connected with 30-wire Flexible Flat Cable (FFC) (Fig. 5). The interface card is connected to the FPGA card with a board-to-board 80 pin connector. The SSD is connected to the FPGA board with SATA cable.

In the new version (System 2), the five pieces of 1.2 megapixel cameras will be connected via serial cables to the interface card.

4.5 Operation

The cameras are initialized through separate I^{2}C buses. They run synchronized. Their integration times are the same, and they receive the same system clock, and exposure trigger signal. Therefore, the individual frames are captured at the same time.

The vision system is connected to the on-board control computer of the UAV through another I^{2}C bus. Through these connections, the vision system receives the attitude estimation calculated by the navigation computer, and

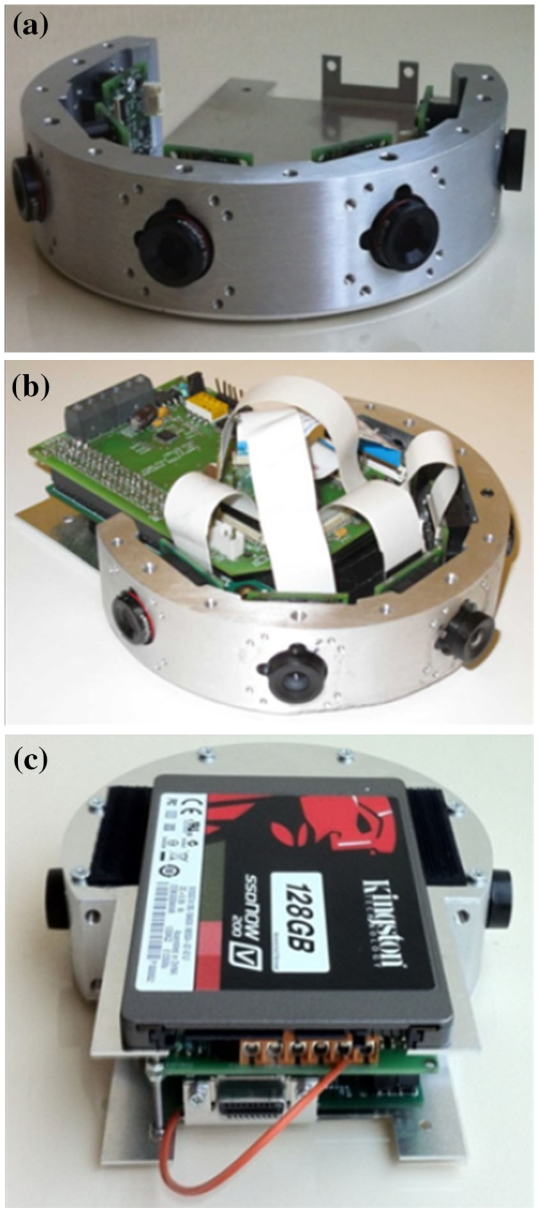

Fig. 6 Camera holder aluminum frame with the cameras (**a**). Open system (**b**). The entire vision system without the power units (**c**)

based on it, the vision system calculates its own, more precise yaw, pitch, and roll figures, which are sent back to the navigation system. In case of intruder aircraft detection, its position and size information is sent to the navigation and control computer as well, to initialize an avoidance maneuver if necessary.

4.6 Power supply

The total power consumption of the vision system is about 4.2 W. Significant part of it is consumed by the SSD, which is 1.2 W alone [27]. The energy source of the entire

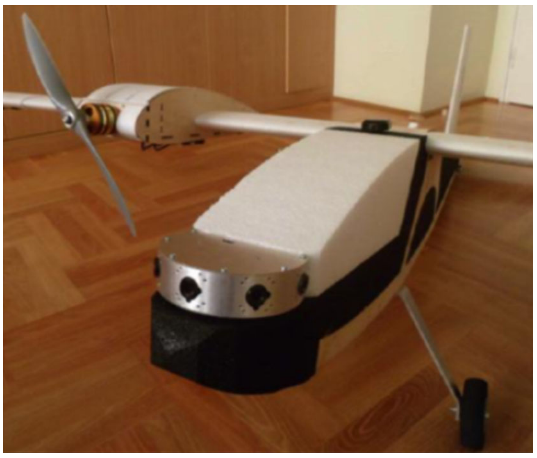

Fig. 7 The vision system mounted on the nose of the aircraft

system is a 1,200 mAh 7.4 V Lithium Polymer battery (2S1P). It can provide continuously 30 amps (25 °C), which ensures that the battery will not be overloaded. It enables close to 2 h continuous operation. Three DC–DC converter modules were applied, a 5 Amp one for the FPGA board, and two pieces of 2 Amp ones for the SSD and the cameras, to reduce cross-coupling of noises trough power lines (Fig. 5).

4.7 System integration

Physical system integration is always a key point of a complex embedded system. It is especially true for an airborne vision system with multiple cameras, where the relative camera orientations are critical. Therefore, a horseshoe-like solid aluminum frame was constructed for holding the cameras and canceling any cross-vibrations (Fig. 6a). The interface and the FPGA cards were put in and behind the horseshoe between two aluminum planes (Fig. 6b, c). The vision system is mounted to the nose of a two engine aircraft on a way that the axis of the front camera is aligned with the horizontal axis of the aircraft (Fig. 7).

5 Algorithmic components

5.1 Remote aircraft detection

The primary goal of the vision system is the identification of the remote (intruder) aircraft, track them, and measure their virtual sizes in pixel. Let us assume that the background is either clear sky or cloudy sky. The structure of the clouds can be either dense with relatively few boundaries with low contrasts [(c) Fig. 8a], or may contain highly structured clouds with sharp, high contrast boundaries

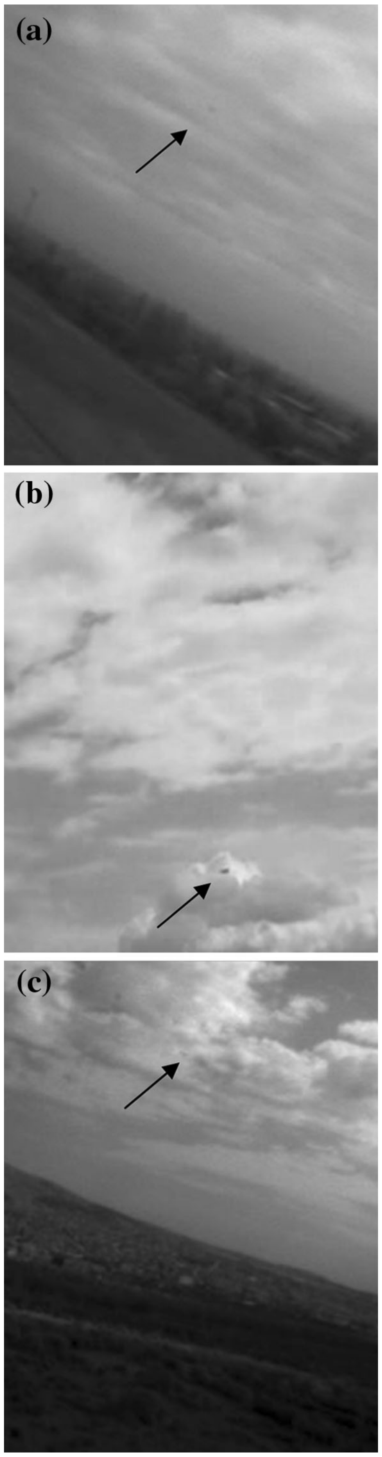

Fig. 8 Typical situations in remote aircraft detection. The background is covered with dense low contrasts cloud system (**a**). The background is constructed of highly structured cloud system with high contrasts at the cloud boundaries (**b**). Distant aircraft generate a small spot in a highly structured high contrast cloud system

[(c) Fig. 8b, c]. In these cases, the gradients are relatively low in the internal parts of the clouds. Therefore, the algorithm is based on finding the dark or bright small spots

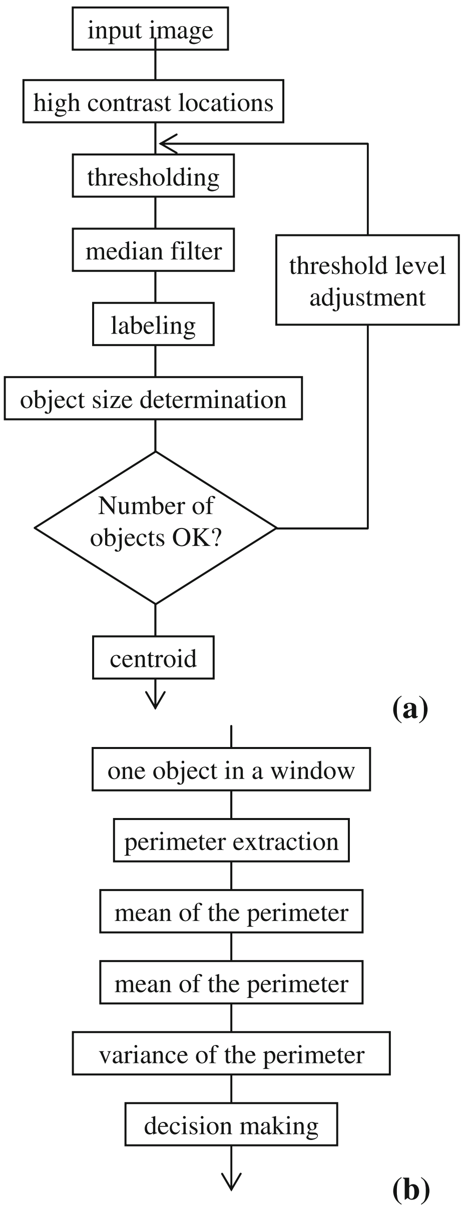

Fig. 9 Flow-chart showing the major steps of the intruder aircraft detector algorithm. The pre-processing part of the algorithm (**a**) is executed on the entire frame, while the post-processing part (**b**) is executed on ROI windows containing individual objects

in front of semi-flat background. Naturally, the algorithm will lose an aircraft at the high contrast boundaries, but the tracking algorithm will keep the target for those few frames, while it crosses the edge.

The algorithm is constructed of three major phases: (1) pre-processing, in which the suspicious regions are identified, (2) ROI (region of interest) processing, in which those spots are kept, which are in a semi-homogenous region, and finally (3) tracking is applied to discard the non-continuous traces.

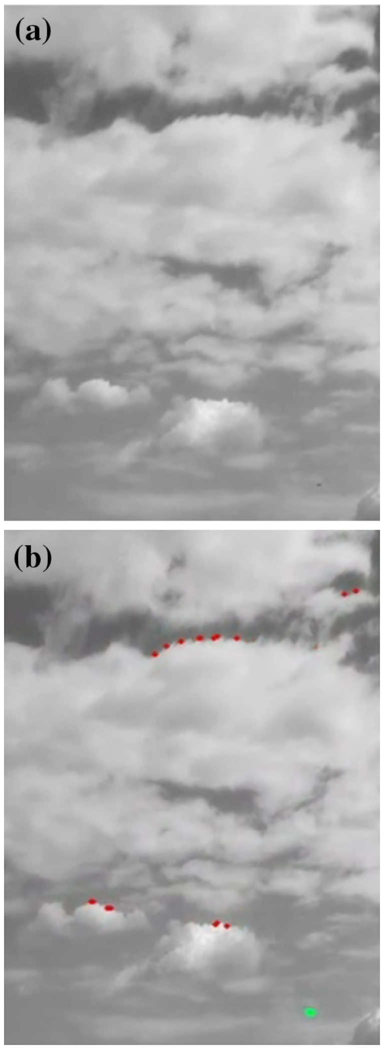

Fig. 10 Raw image (**a**) and the detection result (**b**). The pre-processing algorithm found all the marked points. Some of those are cloud boundaries, the other is aircraft. Then, the post-processor discarded the cloud boundaries (*red*), and kept the aircraft (*green* in the lower right corner)

The algorithm starts with the pre-processing phase (Fig. 9a), where all the operators are applied to the entire frame. The first step is the identification of the small objects (called candidate objects). It is done with a 5×5 zero-sum convolution kernel followed by a thresholding. Then, the single-point-sized candidate objects, considered as noise, are deleted with a median filter. The remaining objects are labeled and their sizes are calculated. After the object number is calculated, the threshold number for the next frame is recalculated. The goal is to keep the number of the found objects in the range of 20–30, because this is the number that the implemented processor architecture can handle without dropping frames. The last step of the pre-processing is the centroid calculation.

The second major part of the algorithm is the post-processing (Fig. 9b), where the ROIs, identified in the first phase are further analyzed. The operators are calculated in windows (foveae), which are cut out from the full frame with the centroid points in the middle. This fovea processing approach saves significant processing power, which is very important in a low power embedded image processing system, where the real-time operation is critical. The goal of the post-processing is to discard those candidate objects, which are most likely not intruder aircraft. As it can be seen in Fig. 10, the wrong candidate points are typically cloud boundaries. These false positive points can be filtered out with the following algorithm. The pixel values lying on a circle with a given radius around the candidate points are extracted, and the variances of the series of pixels are calculated. If the object is on a cloud boundary, part of its perimeter pixels will be bright cloud pixels, others will be darker sky pixels, and hence the variance will be high. On the other hand, a candidate object against smooth background is assumed to be a flying object.

In the third phase, a JVC (Jonker-Volgenant-Castanon) multi-target tracking algorithm [28] is run. In this phase, those objects are extracted, which are found on a continuous track, and the others are discarded.

The evaluation of the remote aircraft detection algorithms is complex and not a well-established process due to the lack of standardized methodology and standard databases. Therefore, different research groups use different benchmark sequences with different kinds of aircrafts, from different distances, with different approaching angles, under different weather conditions. For example, in [28] the authors claimed that they could detect Boomerang type UAV from 412 to 881 meters under different weather conditions and approaching angles in different scenarios. Using these data, we calculated that in the former case, the aircraft was seen under 0.3°, which was 6 pixels in their camera system, while it was 0.1° (2 pixels) in the latter case.

The algorithm performance evaluation approach introduced in this paper is different. Our concept is to measure the minimum detectable spot size generated by the intruder aircraft on the camera image. The advantage of this approach is that (1) this data characterize the algorithm (2) and on the other hand, from this data, one can estimate the detection distance of different intruders in different situations with a given camera. Naturally, we have to keep in mind, that some parts of the complex silhouettes (e.g., wings viewed from the front) of a distant aircraft often causes sub-pixel problems (Fig. 11). The pixel representation of a remote aircraft strongly depends on its attitude [29].

During the algorithm development and enhancement phase, simulated sequences have been used, and real sequences captured from a UAV or from the ground (Note; it is rather difficult to fly two fixed-wing UAVs such that, one is in the field of view of the other, keeping booth of them in remote controllable distances). For evaluation purposes, three annotated sequences with different weather conditions have been selected. These are called benchmark scenarios. Each snapshot in (c)

Figure 8 shows a typical situation from these scenarios. Performance analysis of the different scenarios is summarized in Table 2. As it can be seen, the false tracked targets often appear in the detection results in case of complex scene with strongly structured clouds (Fig. 9). However, these false positive objects can be filtered out, because the tracked object data is entering to a 3D track calculator, which decides whether a track is a collision track or not. The falsely tracked objects are typically stationary cloud pieces. Since these spots are drifting with the optical flow generated by the ego-motion of the UAV, it is easy to distinguish them from moving objects (aircrafts) by this 3D track calculator.

5.2 Visual attitude calculation

The secondary goal of the system is to support the navigation system by calculating angular velocity figures. This is done by a camera pose estimation algorithm [30–32]. The input of these algorithms is the displacement of a minimum number of some stationary ground points on two consecutive frames, and the output is the angular ego-motion of the camera and its relative displacement. Therefore, first, feature points on the image have to be found, and then, the displacement has to be calculated with matching the feature points' neighbourhoods. Before starting the huge work of the FPGA implementation of the algorithms, research was pursued to analyze the accuracy aspects, and to understand the expectable capabilities and the limitation.

5.2.1 Angular and linear velocity ranges

One of the first issues is the range of the angular and linear velocity the vision system can follow. This should be considered from two points of view. First of all, the motion blur has to be kept in a tolerable range, practically below one pixel. The angular speed of an acrobatic quad or fixed-wing UAV can be even 720°/sec; however, a medium size fixed-wing UAV is designed for maximum 180°/sec, and its typical angular velocity is 50°/sec. (We reached 75°/sec pitch speed with our 3.2 m wingspan foam UAV as a maximum.) On a bright sunny day, one can go down with

Table 2 Performance analysis of the algorithm on different scenarios

Scenario	Description	Suspicious locations	Candidate objects/frame		Validated tracks	
			Distant aircraft found:	Average false candidate object number/frame:	Distant aircraft tracked:	False target tracked:
#1	Captured from hand, contains a approaching airliner (spot size from 4 × 5 to 7 × 4 pixels) with mixed cloudy and blue sky background	20–30	94.3 %	0.5 (max: 2)	78.7 %	8.6 % (max 1 object)
#2	Captured from a UAV, contains a approaching airliner (spot size from 3 × 4 to 5 × 3 pixels) with dense cloudy background	20–30	89 %	2 (max 11)	89 %	31.9 % (max: 3 objects)
#3	Captured from a UAV, contains a moving away UAV (spot size from 6 × 3 to 3 × 2 pixels) with mixed strongly structured clouds and blue sky background	20–30	86.4 %	4 (max: 12)	93.2 %	100 % (max: 5 objects)

Percentages indicate the ratio of the frames to the total number of frames on what the detection truly or falsely happened

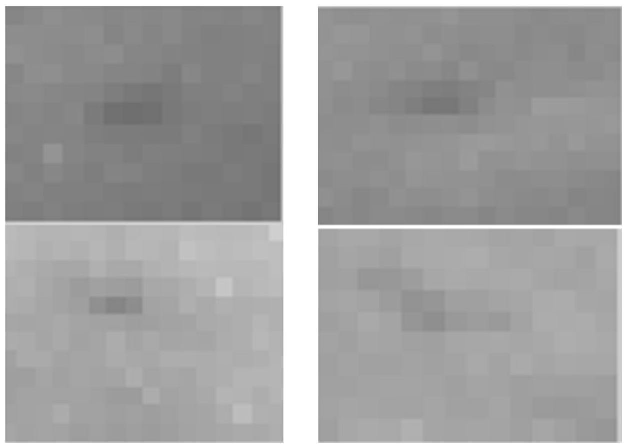
Fig. 11 Distant aircraft signatures. The bodies of the aircrafts make larger contrast (20–30 LSBs); however, the wings, wheel, or tails make it appear sub-pixel size, hence it generates lower contrast (10–15 LSBs). We consider the size of these pots as 3 × 4, 3 × 5, 3 × 4, and 3 × 5, respectively

the exposure time to 1 ms, while it goes up 20 ms on a dark cloudy day. The pixel shift can be calculated as:

$$s_r = \frac{\omega_e \times t_{exp}}{\alpha_{res}} \tag{2}$$

where:

- s_r: is the shift in pixels caused by the angular speed
- ω_e: is the camera ego rotation
- t_{int}: is the exposure time
- α_{res}: is the view angle of a pixel (0.1° in our system)

If we assume 1 ms exposure time, 50°/sec angular velocity, and 0.1 pixel resolution, we get 0.5 pixel shift during the exposure, which does not cause motion blur. However, with the drop of the light intensity, the motion blur starts increasing, hence, the accuracy of the system decreases. It can be even worse in badly illuminated indoor situations.

The linear velocity of the small- and medium-sized UAVs is in the range of 20–40 m/s. The altitude of the flight is 30–300 m from the ground. From these data, and the internal parameter of the vision system, the shift in pixels, caused by the velocity, is calculated.

$$l_g = \frac{h}{\sin\left(\frac{\beta}{2}\right)} \tag{3}$$

$$d_p = l_g \times \mathrm{tg}(\alpha_{res}) = \frac{h \times \mathrm{tg}(\alpha_{res})}{\sin\left(\frac{\beta}{2}\right)} \tag{4}$$

$$s_v = \frac{v_e \times t_{fr}}{d_p} = \frac{v_e \times t_{int} \times \sin\left(\frac{\beta}{2}\right)}{h \times \mathrm{tg}(\alpha_{res})} \tag{5}$$

where:

- s_v: is the shift in pixels caused by the linear velocity of the aircraft
- h: is relative elevation of the UAV to the ground
- β: is vertical view angle of the camera of the vision system
- l_g: closest ground location on the camera image (side looking cameras with horizontal axis and β vertical view angle, 70° in our system)
- d_p: is pixel size in meter on the ground
- v_e: is the linear velocity of the aircraft

Let us assume a flight at 30 m in a sunny day, the shift will be in the sub-pixel range. However, during landing and takeoff, or in a darker situation, motion blur might occur.

After analyzing the intra-frame shift, the inter-frame shift has to be calculated as well. This value indicates, how large the displacement is between two frames. If we substitute the frame time (reciprocal of frame rate) into (2) and (5), we get the displacement value. Under normal flight conditions, it is between 5 and 30 pixels, hence, the displacements are in the range of the WVGA or megapixel cameras. Note, that the displacement is the same for both the side and the front looking cameras, because it depends on the distance of the observed objects and the velocity of the camera.

5.2.2 *Analysis of camera pose estimator algorithms*

Let us assume that the extraction and the matching of the feature points are feasible as it can be seen in the previous calculation, then a camera pose estimation algorithm can be chosen. Four different camera pose estimator algorithms were implemented and the achievable accuracies and computational complexities were analyzed. The camera pose estimator algorithms were the five-point algorithm [30, 33, 34], the eight-point algorithm [35], the MLESAC algorithm [36], and a scene homography-based algorithm [35]. The camera pose estimator algorithms were analyzed both with synthesized and real measured data captured during the flight of the UAV. The performance can be tested precisely with the synthesized data, because in case of real flight data, the ground truth is not known precisely, due to the lack of expensive high precision leaser gyroscope (Fig. 12). However, it was shown in our research paper that more accurate navigation data can be calculated when the camera, the gyroscope, and the GPS data are combined than without involving the camera data [31, 32, 37].

Using synthesized data, the reachable precision with different algorithms have been analyzed. The error of the angular velocity versus the accuracy of the input points is summarized in Table 3. As it can be seen, the five-point algorithm is the most precise. If the feature point coordinates are given with double precision, the error is in the range of the numerical error. If the point coordinates are discretized, like in a real camera image, the improvement of the error of the four algorithms calculated with different cameras are in a linear relation with the degree per pixel value of the actual cameras. It can be also seen that the homography-based algorithms is close to the five-point algorithm.

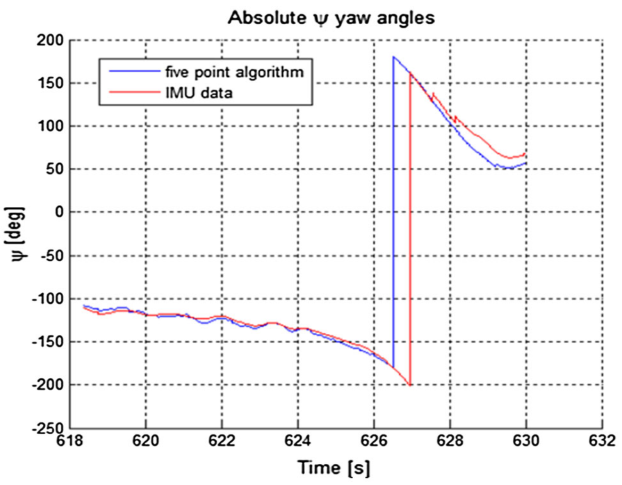

Fig. 12 Yaw angles of a flight captured by the IMU unit (fused gyroscope and GPS data) and calculated from the five-point algorithm

From computational point of view, the five-point algorithm is the most demanding, as it requires either the computation of the roots of a 10th degree polynomial or the computation of the Gröbner basis of a 10×20 matrix. Though the latter is faster, it is still challenging in an embedded system since its computational need is very high. The simplest is the homography, which requires the solution of a few straightforward equations and a singular value decomposition (SVD) on a $9 \times n$ matrix, where n is the number of coordinate pairs, which is most of the time is around 20. In a typical situation, the difference of the computational demand of the homography is two orders of magnitude smaller than the computational demand of five-point algorithm in the number of the multiplications.

6 Multi-core processor architecture in the FPGA

The image processing system should execute the following parallel tasks:

- identifying and tracking intruder aircraft;
- calculating the attitude changes of the aircraft;
- communicating with the control and navigation processor of the UAV;
- transferring the raw image data toward the SSD.

All of these functionalities are handled by a custom designed multi-core processor architecture implemented in a Spartan 6 LX150T FPGA. The basic concept of the processor design was to mimic the human fovea vision in a way that a pre-processor examines the entire frame and identifies those locations, which need more attention. Then, the focus of the processing is shifted to these locations one after the other, similarly as the fovea focuses to different important details of a scene.

The architecture of the system is shown in Fig. 13. The multi-core fovea processor system is constructed of three main blocks: Pre-Processor unit (Adaptive thresholding block and the Labeling and Centroid block), the Fovea Processor unit and the Microblaze processor. The system is controlled by a Xilinx MicroBlaze processor which is a general purpose 32-bit soft-core processor implemented in Xilinx FPGAs. It has relatively low computational power (~100 MHz clock speed), which means that it cannot perform image processing tasks. It can be used as the

Table 3 Mean of the error produced by the different camera pose estimator algorithms

	Absolute displacement figures (double precision)	Discretized displacement figures (pixelized, 0.09°/pixel)	Discretized displacement figures (pixelized, 0.05°/pixel)	Discretized displacement figures (pixelized, 0.04°/pixel)
Five-point algorithm	9.17×10^{-11}	7.40×10^{-2}	5.42×10^{-2}	4.03×10^{-2}
Homography	1.97×10^{-2}	7.41×10^{-2}	5.64×10^{-2}	5.09×10^{-2}
Eight-point algorithm	2.83×10^{-3}	4.99×10^{-1}	3.55×10^{-1}	2.82×10^{-1}
MLESAC algorithm	3.79×10^{-3}	2.63×10^{-1}	1.99×10^{-1}	1.61×10^{-1}

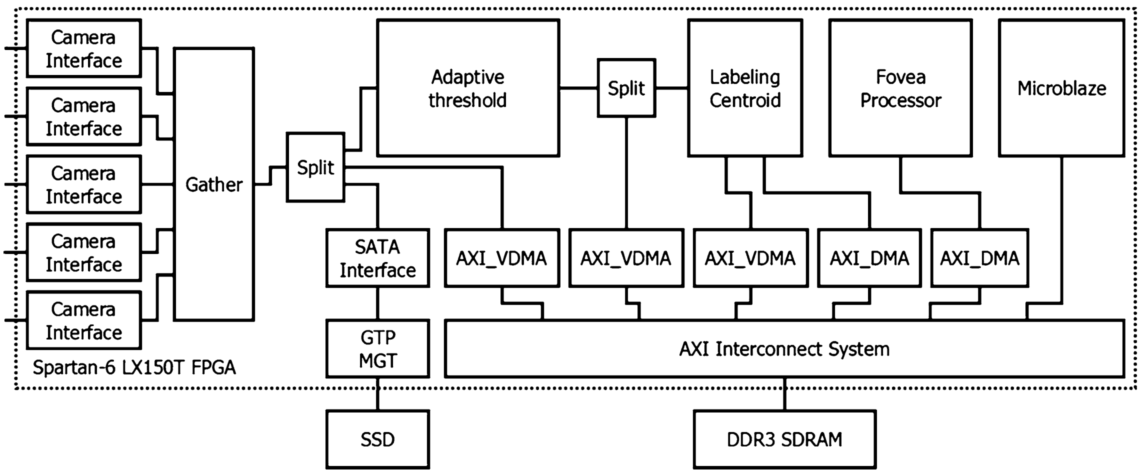

Fig. 13 The block diagram of the image processing architecture

control processor of the system, and also to perform some decision making and communication. The computationally intensive pre-processing of the incoming image flows is performed by the specialized streaming Adaptive Threshold and Labeling and Centroid Units. The MicroBlaze then goes through the identified locations and performs fovea (region of interest, ROI) processing one after the other, by instructing the binary and grayscale Fovea Processor to cut out the required windows, copy them into the internal block memories of the FPGA, and execute the program sequences. The fovea processor is a single powerful unit, which handles the selected regions one after the other.

As it is shown in Fig. 13, the five parallel 8-bit data flows arriving synchronously from the cameras are combined to one, time multiplexed 8-bit data flow by the Gather unit. The pixel clock frequency of the cameras can be between 15 and 25 MHz while the Gather and the Pre-Processor unit is operating on 150 MHz to avoid blocking. The combined data flow goes to the SATA core, the on-board DDR3 SDRAM via a standard Xilinx AXI Video DMA IP core and to the full-frame streaming pre-processor as well.

6.1 The pre-processor unit

The pre-processor has a streaming architecture, meaning that it cannot randomly access the entire frame, but it receives it row-wise sequentially as the image is read out from the Gather unit. To be able to calculate neighborhood operators, it collects a few lines of the frame and processes those lines together. As it was shown in the previous section, the first step of the pre-processing is an adaptive thresholding step where a 5×5 neighborhood is considered. As the data stream flows through the processor, it finds high contrast locations. The resulting binary image is saved to the on-board memory and streamed into the Labeling and Centroid Unit.

Labeling connected components of an image using line-by-line streaming architecture requires two stages because 'U' shaped objects are detected first as two separate objects, which should be merged in the second stage. In this case, the potential intruder aircraft are selected by the size of the labeled objects, while its position is described by the (x, y) coordinates of its centroid. Bailey and Johnston [38] described a method to compute centroids of the labeled components by a single pass algorithm. In the Labeling and Centroid Unit, a similar architecture is used. The indexes of connected objects found in the current line are stored in a small label stack, while the connected objects found previously are stored in a merger table. The usable size of the merger table is 255 elements therefore the merger table is 8-bit wide. The merger table is updated at the end of each line during the synchronization period of the cameras. The labels to be merged are processed in a reversed order (the first merge is processed last by reading it from the merge stack) and previous merges are also checked to prevent long chain of merged objects in the merger table.

To compute the centroid of the merged objects the sum of the x and y coordinates and the number of pixels should be saved, which requires three additional 32-bit fields in the merging table. The merging of these partial results of the centroid computation is executed by the MicroBlaze microprocessor. At the end of the frame, an AXI DMA is used to save the contents of the merger table into an on-chip BRAM, which can be accessed by the Microblaze microprocessor and an interrupt is generated to start finalization of the centroids. The 32-bit wide AXI interconnect running on 100 MHz clock frequency, which means that the merger table can be transferred in 10.2 µs and in the worst case the final centroids can be computed in 100 µs. Based on the number of objects, a new threshold value is set to the Adaptive Threshold Unit before the next frame arrives, to keep the number of candidate objects in a manageable range (between 20 and 30).

6.2 The fovea processor unit

The fovea processor unit contains grayscale and binary window processors, and another block, responsible for the calculation of the variance of the pixels on the perimeter of the candidate objects.

Grayscale operators are performed by the block shown in Fig. 14a. As opposed to the high resolution full frames, which were stored in external DRAM, the 128 × 128 sized foveae are stored in internal block RAMs (BRAM) of the FPGA. The number of foveae can be configured according to the requirements of the image processing algorithm. Fast off-chip DRAM access is provided by a Direct Memory Access (DMA) engine which can cut out the 128 × 128 sized foveae from the input image. For efficient utilization of the available memory bandwidth the *x* coordinate of the top left corner of a fovea must fall on a 32-pixel boundary.

The arithmetic block of the processor contains an array of highly optimized processing elements (PE) from which only one is active during the operation. The PE array has a modular structure where existing functions can be easily removed and new functions can be easily inserted before synthesizing the block according to the requirements of the image processing algorithm. The supported operations are

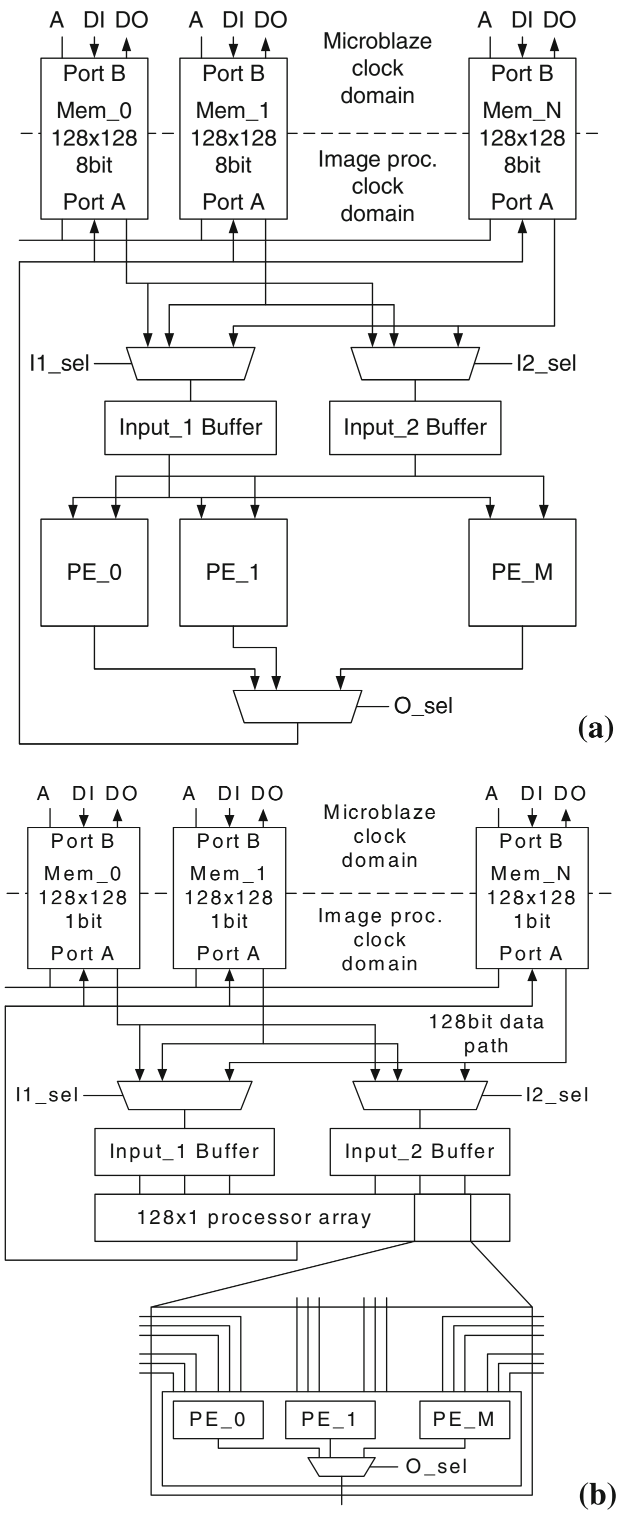

Fig. 14 The block diagram of the grayscale (**a**) and the binary (**b**) fovea (window) processor blocks

convolution, thresholding, and arithmetic operations such as *addition, subtraction, multiplication,* and *absolute value computation*. These operations are used for matching areas around feature points and identifying displacements.

The architecture of the binary fovea processor block is similar to the grayscale processor block as shown in Fig. 14b. Here the internal BRAMs store the 128 × 128 sized binary foveae also. However, the image processing algorithms usually require more binary and morphological operators than grayscale ones; therefore, the binary image processor is designed for higher performance. Each BRAM is configured with a 128-bit wide data bus and all the pixels in a row are computed in parallel here. The supported operations are *erosion, dilation, single pixel removal, reconstruction*, and *two input-one output logic operations* such as *AND*, *OR*, and *XOR*. The *Global OR* and *Change Signals* are also implemented. Detailed description of the arithmetic and binary blocks can be found in [39].

In addition to the spatial image processing operations, the grayscale processor is extended with a specialized unit to accelerate tracing of the boundary of the objects, which is an important step to avoid false positive intruder detection near the edges of the clouds. The operation works on the original grayscale and the labeled image which is loaded into the BRAMs of the grayscale processor. The boundary is traced counterclockwise and pixels of the object are searched in the eight connected neighborhood. As the boundary of the objects is traced on the binary image, the grayscale values of the perimeter are stored to a FIFO.

The architecture is designed for performance therefore reading from the BRAMs is pipelined and has two clock cycles latency. The entire neighborhood of the actual pixel is searched at worst in ten clock cycles. In this application, the boundaries of relatively small objects are traced; therefore, the length of the boundary path is restricted to 255 pixels. Time to trace the longest path can be executed at most 2,550 clock cycles or 17 µs when the processor is operating on 150 MHz clock frequency.

The mean value and variance of the pixels of the traced path are also computed by this unit to reduce the load on the Microblaze processor. During the mean calculation a serial fixed point divider (8-bit integer and 16-bit fractional part) from the Xilinx Core Generator is used to save FPGA resources. When computation of the mean value is completed the variance computation is started, which utilizes the multiply accumulate function of a DSP48 slice in the first stage. Computation of the mean and variance requires at most 345 clock cycles or 2.3 µs. The results are placed into two status registers accessible by the Microblaze and an interrupt is generated.

7 Conclusions

In this paper, a five-camera vision was introduced. The system was designed to be able to operate on fixed-wing small- and medium-sized UAV platforms. It has low power consumption, builds up from cheap parts. It is vibration tolerant and compact. Its role is the real-time vision-based collision warning and attitude (orientation angle) calculation, and visual flight data acquisition. The parameters of the hardware and the performance of the algorithm is compared to the state of the art.

Acknowledgments The ONR Grants (N62909-11-1-7039, N62909-10-1-7081) is greatly acknowledged., The authors express their thanks to grants TÁMOP- 4.2.1.B-11/2/KRM-2011-0002 and TÁMOP-4.2.2/B-10/1-2010-0014.

References

1. Felder, W.: Unmanned System Integration into the National Airspace System. Philadelphia, PA, USA: Keynote Presented at ICUAS 2012, June (2012)
2. Dey, D., Geyer, C., Singh, S., Digioia, M.: Passive, long-range detection of aircraft: towards a field deployable sense and avoid system. Field and Service Robotics, Springer Tracts in Advanced Robotics, 2010, Vol. 62, pp. 113–123 (2010). doi: 10.1007/978-3-642-13408-1_11
3. Federal Aviation Administration: Fact sheet—unmanned aircraft systems (UAS) (2010)
4. Department of Defense: Unmanned aircraft system airspace integration plan. Tech. Rep. March, Department of Defense (2011)
5. Federal Aviation Administration: Integration of unmanned aircraft systems into the National Airspace System Concept of Operations (2012)
6. Mejias, L., McNamara, S., Lai, J., Ford, J.: Vision-based detection and tracking of aerial targets for UAV collision avoidance. International Conference on Intelligent Robots and Systems (IROS), pp. 87–92 (2010)
7. Dey, D., Geyer, C., Singh, S., Digioia, M.: Passive, long-range detection of aircraft: towards a field deployable sense and avoid system. Robotics Institute. Paper 962 (2009)
8. Mejias, L., Ford, J. J., Lai, J. S.: Towards the implementation of vision-based UAS sense-and-avoid. In: Proceedings of the 27th International Congress of the Aeronautical Sciences (ICAS 2010), 19–24. September 2010, Acropolis Conference, Centre, Nice (2010)
9. Livadas, C., Lygeros, J., Lynch, A., Nancy, : High-level modeling and analysis of the traffic alert and collision avoidance system (TCAS). Proc. IEEE **88**(7), 926–948 (2000)
10. Federal Aviation Administration: Fact sheet—automatic dependent surveillance-broadcast (ADB-S) (2010)
11. Fasano, G., Accardo, D., Moccia, A., Carbone, C., Ciniglio, U., Corraro, F., Luongo, S.: Multi-sensor-based fully autonomous non-cooperative collision avoidance system for unmanned air vehicles. J. Aerosp. Comput. Inf. Commun. **5**(10), 338–360 (2008)
12. Achtelik, M., Stephan W., Siegwart, R.: Onboard IMU and monocular vision based control for MAVs in unknown in-and outdoor environments. Robotics and automation (ICRA), 2011 IEEE international conference on. IEEE, (2011)
13. Meier, L., Tanskanen, P., Heng, L., Lee, G.H., Fraundorfer, F., Pollefeys, M.: PIXHAWK: a micro aerial vehicle design for autonomous flight using onboard computer vision. Auton. Robots **33**(1–2), 21–39 (2012)
14. Nikolic, J., Burri, M., Rehder, J., Leutenegger, S., Huerzeler, C., Siegwart, R.: A UAV system for inspection of industrial facilities. In: Aerospace Conference, 2013 IEEE (pp. 1–8) IEEE (2013, March)

15. Ahlberg, C., Lidholm, J., Ekstrand, F., Spampinato, G., Ekstrom, M., Asplund, L.: Gimme-a general image multiview manipulation engine. In: Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on (pp. 129–134) IEEE (2011, November)
16. Mejias, L., McNamara, S., Lai, J., Ford, J.: Vision-based detection and tracking of aerial targets for UAV collision avoidance. In: Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on (pp. 87–92) IEEE (2010, October)
17. International Civil Aviation Organization: Air traffic management. ICAO Doc 4444, fifteenth edition (2007)
18. Zarándy, Á., Zsedrovits, T., Nagy, Z., Kiss, A., Roska, T.: Onboard see-and-avoid system. Conference of the Hungarian Association for Image Processing and Pattern Recognition (Kepaf 2013) pp. 604–617 Bakonybel, (2013)
19. http://www.framos.com/products/en/cameras/camera-modules/fcb-ma130-15447.html (2014). Accessed 2 Sept 2014
20. http://www.sedeco.nl/sedeco/index.php/lenses/smount (2014). Accessed 2 Sept 2014
21. http://www.mobisensesystems.com/pages_en/aptina_modules.html (2014). Accessed 2 Sept 2014
22. https://www.leopardimaging.com/LI-USB30-M021.html (2014). Accessed 2 Sept 2014
23. http://www.ptgrey.com/products/index.asp (2014). Accessed 2 Sept 2014
24. http://www.edmundoptics.com/imaging/imaging-lenses/ (2014). Accessed 2 Sept 2014
25. http://defense.ge-ip.com/products/gpgpu/c497 (2014). Accessed 2 Sept 2014
26. http://www.tokudenkairo.co.jp/exp6t/ (2014). Accessed 2 Sept 2014
27. http://www.legitreviews.com/article/1980/1/ (2014). Accessed 2 Sept 2014
28. Zsedrovits, T.: Visual sense and avoid for fixed-wing small size unmanned aerial vehicles. Dissertation. Pazmany Peter Catholic University, Budapest, Hungary, 2014. https://itk.ppke.hu/uploads/articles/162/file/Zsedrovits%20Tamas%20dissertation.pdf. Accessed 2 Sept 2014
29. Zsedrovits, T., Zarandy, A., Vanek, B., Peni, T., Bokor, J., Roska, T.: Estimation of relative direction angle of distant, approaching airplane in sense-and-avoid. J. Intell. Rob. Syst. **69**(1–4), 407–415 (2013)
30. Chu, T., Guo, N., Backén, S., Akos, D.: Monocular camera/IMU/GNSS integration for ground vehicle navigation in challenging GNSS environments. Sensors (Basel, Switzerland), vol. 12, no. 3, pp. 3162–85, Jan. 2012
31. Zsedrovits, T., Bauer, P., Zarandy, A., Vanek, B., Bokor, J., Roska, T.: Error analysis of algorithms for camera rotation calculation in GPS/IMU/camera fusion for UAV sense and avoid systems. ICUAS (2014)
32. Zsedrovits, T., Bauer, P., Zarandy, A., Vanek, B., Bokor, J., Roska, T.: Towards real-time visual and IMU data fusion. Presented at the AIAA Guidance, Navigation, and Control Conference and Exhibit, (2014)
33. Ristic, B., Vo, B.N., Clark, D., Vo, B.T.: A metric for performance evaluation of multi-target tracking algorithms. Signal Process. IEEE Trans. **59**(7), 3452–3457 (2011)
34. Nistér, D.: An efficient solution to the five-point relative pose problem. IEEE Trans. Pattern Anal. Mach. Intell. **26**(6), 756–777 (2004)
35. R. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision. Cambridge University Press, 2004
36. Torr, P.H.S., Zisserman, A.: MLESAC: a new robust estimator with application to estimating image geometry. Comput. Vis. Image Underst. **78**(1), 138–156 (2000)
37. Stewénius, H., et al.: Recent developments on direct relative orientation. ISPRS J. Photogramm. Remote Sens. **60**(4), 284–294 (2006)
38. Bailey, D. G., Johnston, C. T.: Single pass connected components analysis. In: Proceedings of image and vision computing New Zealand 2007, pp. 282–287, Hamilton, New Zealand, December 2007
39. Nagy, Z., Kiss, A., Zarándy, Á., Vanek, B., Péni, T., Bokor, J., Roska, T.: Volume and power optimized high-performance system for UAV collision avoidance ISCAS-2012, Seoul, Korea (2012)

Dr. Ákos Zarándy spent 25 years with the scientific research and development of various array processor architectures, implementations, real-time image processing problems, and applications. He obtained his Ph.D. from the Hungarian Academy of Sciences in 1997. During his Ph.D. studies, he spent more than 2 years at the University of California, Berkeley. He led several successful research and development projects, including vision system development, locally adaptive sensor development, and solved ultra high-speed vision problems. He works for the Institute for Computer Science and Control of the Hungarian Academy of Sciences (MTA-SZTAKI). He is also a professor at Pazmany Peter Catholic University from 2011. He is the author or co-author of 34 pier reviewed scientific papers published in international scientific journals.

Mate Nemeth received his M.Sc degree in 2013 from the Budapest University of Technology and Economics, and currently a Ph.D. student at Pazmany Peter Catholic University, and also at the Institute for Computer Science and Control of the Hungarian Academy of Sciences (MTA-SZTAKI). His area of interest is real-time vision system implementation, and multi-core processor architectures implementation in FPGA.

Zoltan Nagy received both his M.Sc and his Ph.D. degrees in information technology from the University of Veszprém in 1999 and 2008, respectively. Since 2007, he is with the Institute for Computer Science and Control of the Hungarian Academy of Sciences. Since 2010, he is a part-time associate professor in the Pázmány Péter Catholic University. His research interests are computer architectures, digital circuit design, field programmable gate arrays, reconfigurable computing, and high-performance computing.

Dr. Andras Kiss is a lecturer at the Faculty of Information Technology at Pazmany Peter Catholic University, Budapest, Hungary. He received his Ph.D. at the doctoral school of Pazmany Peter Catholic University in 2012. He is teaching digital architectures at the University. In addition to teaching, Dr. Andras Kiss is a research fellow of the Institute for Computer Science and Control of the Hungarian Academy of Sciences. His research topic is mapping computationally intensive algorithms into FPGAs.

Levente Santha received his B.Sc degree in 2014 from the Budapest University of Technology and Economics (BME), and currently an M.Sc student. Parallel to his studies, he is actively taking part in research and development activities at the Institute for Computer Science and Control of the Hungarian Academy of Sciences (MTA-SZTAKI). His area of interest is communication protocols and their implementation in FPGA.

Tamás Zsedrovits has received his M.Sc in 2009 from Pazmany Peter Catholic University. Since that, he is working on his Ph.D. at the same university. His area of interest is UAV vision algorithms for collision avoidance and vision-based navigation. He is expected to receive his Ph.D. degree in 2014. He spent one year at Notre Dame University during his Ph.D. studies.