On-board see-and-avoid system

Ákos Zarándy*†, Tamás Zsedrovits†, Zoltán Nagy*†, András Kiss*†, Tamás Roska*†

*Computer and Automation Research Institute of the Hungarian Academy of Sciences (MTA-SZTAKI) Budapest, Hungary
†Pázmány Péter Catholic University, The Faculty of Information Technology, Budapest, Hungary.

zarandy.akos@sztaki.mta.hu

Abstract. The algorithmic and the hardware aspects of a visual see-and-avoid (SAA) system are introduced in this paper. The SAA system is designed to operate on small and medium sized unmanned aerial vehicles (UAVs), and to be able to detect and avoid small manned and unmanned aircrafts. The system is designed to be able to perform real-time capturing and analysis of a 4650×1200 sized video by using a many-core cellular processor array implemented in a Spartan 6 (XC6SLX45T) FPGA.

Keywords: UAV, UAV collision avoidance, visual navigation, FPGA, many-core processing, embedded system, foveal processor architecture.

1 Introduction

The Unmanned Arial Vehicles (UAVs) went through an intensive development period in the last decade and in many (mostly military) applications they have proved their reason for existence. Nowadays more and more commercial application opportunities are opening for UAVs, however the process is slowed down due to the lack meeting certain safety standards. One of these safety standards is the automatic collision avoidance system, which is required for the autonomous UAVs as well as the remotely piloted ones [1,4].

Radar based automatic collision avoidance system exists for a long time and all the larger commercial airliners are already equipped with it. Recently the large remotely piloted military UAVs (e.g. Predator, Global Hawk) are also equipped with multiple sensory (TCAS, and radar) based collision avoidance system. These systems work perfectly on large aircrafts, however, radar based solutions are too expensive and too bulky for small or medium sized UAVs. An affordable alternative of radar based detection of the intruder aircraft is the visual detection. In the paper, we are presenting a visual based collision avoidance system, which is designed for small and medium sized UAVs.

Our prototype is about 0.5kg, and consumes less than 5W. It is designed to use 5 pieces of 1.2 Megapixel miniature cameras, an FPGA board with a Spartan 6 (XC6SLX45T), and a 128Gbyte Solid-State Disk (SSD) drive for recording raw video data. The final paper will describes the requirements, then the system will be introduced, including the image processing algorithms, and the many-core processor architecture implemented in the FPGA.

The system is developed at the Computer and Automation Institute of the Hungarian Academy of Sciences (MTA-SZTAKI), and at Pazmany Peter Catholic University with the support of the Office of Naval Research.

2 Requirements

The basic requirement of the system is to detect the other aircraft (intruder) in time, and provide a relatively accurate measurement data of its position and velocity vector, which enable the control and navigation system of our vehicle to safely avoid it. Fig. 1 illustrates the requirements of the safe avoidance. According to the flight safety requirements [4], there should be a certain separation volume around each aircraft, in which nothing else can be. The size of the separation volume (separation minima) differs from airplane to airplane and situation to situation.

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 second before collision is the last time instant, when collision can be avoided with high probability [3]. 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 size UAVs do not interfere with streamliners, or high speed jets, we have to be prepared for other UAVs and Cessna 172 type manned crafts. This means that the maximal joint approaching speed is 100 m/s, therefore we need to detect them from 2000 meters (20 seconds before collision), to be able to safely avoid them.

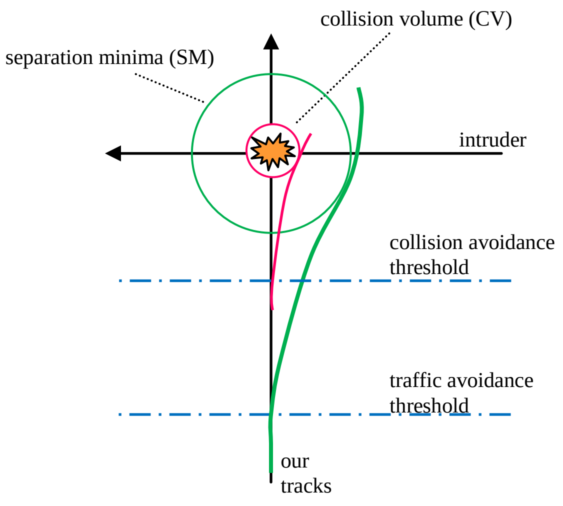


Fig. 1. Traffic and collision avoidance

To be able to perform robust visual detection of an aircraft, it should be at least 5 pixels large in the captured image. For a Cessna 172 class aircraft, with 10 meters wingspan, 0.05 degree/pixel resolution is minimum required.

3 The Sense-and-Avoid system

The control system contains a closed loop with the embedded collision avoidance capability based on visual detection of the approaching aircraft (Fig. 2). The organization of the system is as follows. The input images are recorded by the *Cameras*. The recorded pictures are transmitted by the *Image Acquisition* block to the *Preprocessing* block by which the pictures are filtered. The next step of the processing is the *Detection*. The images are processed by image processing algorithms to detect the approaching objects. *Data Association & Tracking* is responsible for the combination of the orientation and angle of attack data of the approaching object calculated by *Detection* and the own position and inertial data measured by onboard *INS/GPS* (Inertial Navigation System/Global Positioning System).

The second part of the system is the flight control. According to the combined data the relative motion of the approaching object is predicted by *Motion Prediction*. If a risky situation is identified by *Collision Risk Estimation & Decision* a modified trajectory is generated by *Trajectory generation* and the avoiding maneuver is controlled by *Flight Control*. In this paper the image capturing and processing part of the flight control system is presented.

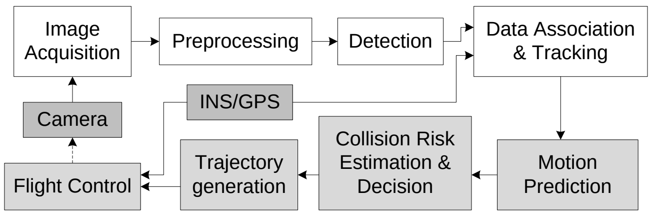


Fig. 2. Diagram of the control system

4 Image processing algorithms

4.1 Test images

To analyze and demonstrate the functionality of a vision based system it is important to have real or at least realistic camera images available. Three types of videos are used to test the developed image processing algorithm:

Series of images are generated by an appropriate simulator, which is able to provide realistic 3D views from the flight scenarios (Fig. 3). Here we know the exact 3D relative positions and the size of the intruder at every t time instant, therefore we can measure the accuracy of the calculations.

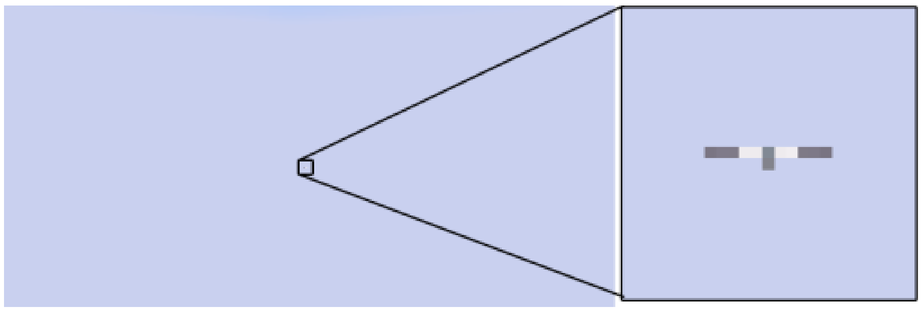


Fig. 3. Generated input image (2200x1100 pixel) from simulator; the square shows the location of the intruder, on the right side the enlarged image of the intruder

Video sequences of flying aircrafts captured from the ground. These sequences are captured at airfields and they contain airplanes with various sizes. Some of these sequences are captured from tripods, other from hand, to be able to mimic the movement of the aircraft. The exact position of the intruder may or may not be known.

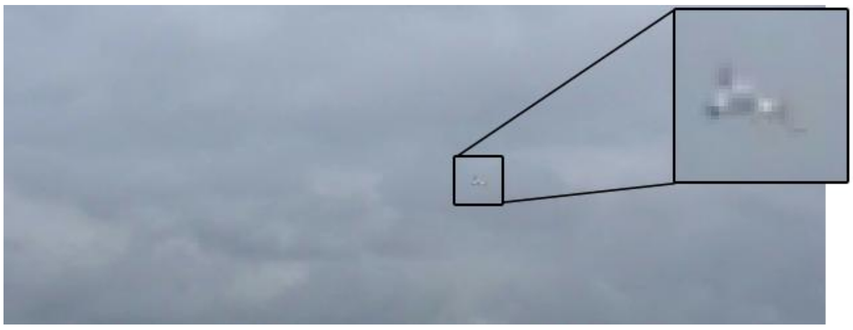


Fig. 4. Image of a remote aircraft in a slightly cloudy situation

Video sequences from real flight scenarios, where the videos are recorded by cameras equipped to the UAV (Fig. 5). In these cases sensor data of the on-board inertial navigation system (INS) is also recorded on a synchronized way.

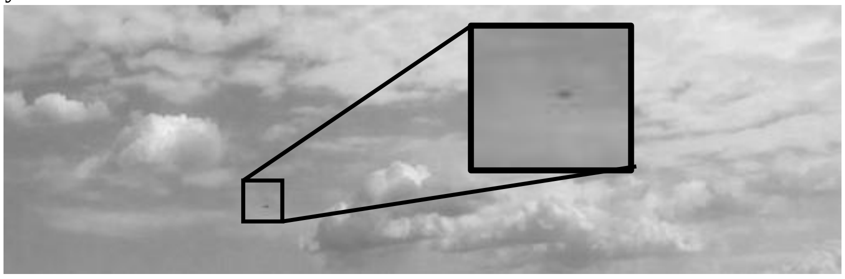


Fig. 5. Image of remote aircraft captured from our UAV.

4.2 Test environment

The algorithm development process contains subsequent parameter refining and new algorithm blocks insertion steps. To be able to compare the efficiency of the different algorithm versions with a different parameter and algorithmic block sets, one should do many test runs on different videos and evaluate the results. To be able to do it automatically, we have developed a testing framework, which automatically executes the algorithm on a number of preselecting video sequences. The video sequences are previously annotated, therefore the ground truth (the position and the size of the intruder aircrafts, and the silhouette) is known, hence the testing framework can reach a full numeric and graphic evaluation of an algorithm version. In this way, automatic evaluation of dozens of different parameter sets can be executed and compared, to be able to efficiently optimize the algorithm.

4.3 Multi-adaptive foveal image processing algorithm

In our system first, we are implementing an image processing algorithm, which is designed for recognizing airplanes against blue sky or cloudy backgrounds. From the very beginning of the algorithm design, we kept in mind the strict power, volume and other constraints of an air-born UAV application. To be able to fulfill these constraints, we decided to use many-core *cellular array processor*, implemented in FPGA. Therefore, we have selected FPGA friendly topographic operators with local interconnections, which well fit in this environment.

We applied a multi-adaptive foveal concept, means that first, we examine the entire image (pre-processing) for candidate airplane locations and then, these candidate positions are further examined by focusing our attention (fovea) to these location one after the other (post processing). In the pre-processing phase, we apply a 7x7 zero sum convolution kernel for enhancing the contrast. In this image, the edges of the clouds have the largest contrast values, hence a simple global thresholding does not work for aircraft identification. Therefore, we apply a both locally and globally adaptive threshold level. The threshold level is locally modified by the edge density of the image. In this way, the threshold level is higher at the cloud edges, and lower in the smooth areas. We add a global constant to this locally adaptive space variant threshold value, which is needed to keep the found candidate pixels in a certain range, to be able to handle it with an on-board device real-time. Fig. 6a shows the flowchart of the pre-processing algorithm. Fig. 7a and b show as the number of the candidate points goes down below 10 as the global adaptation mechanism sets the appropriate threshold value after a few frames.

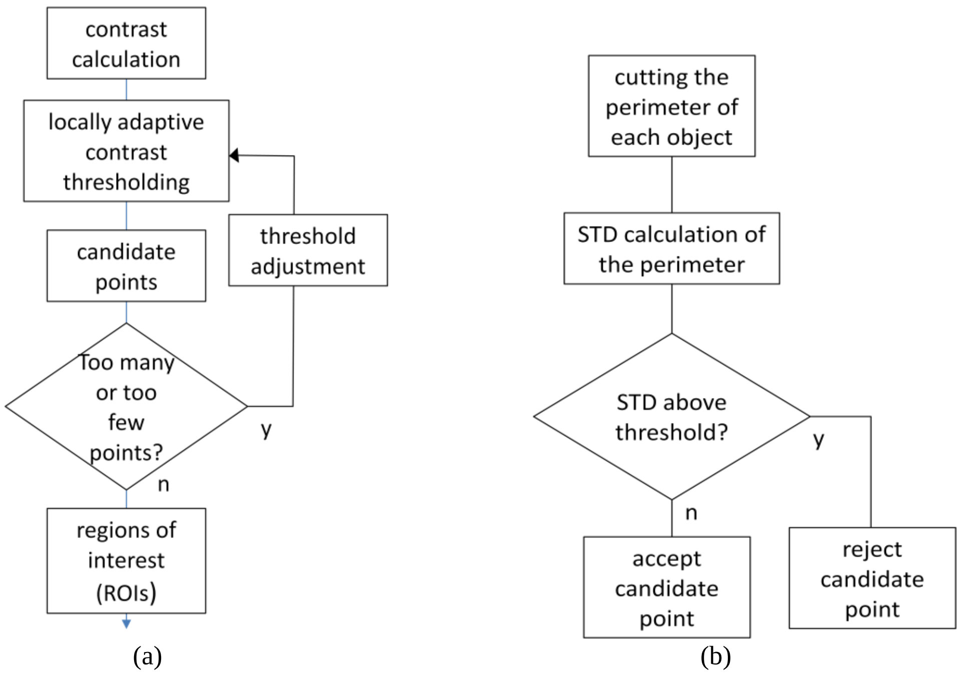

contrast calculation
locally adaptive contrast thresholding
candidate points
threshold adjustment
Too many or too few points?
y
n
regions of interest (ROIs)
(a)
cutting the perimeter of each object
STD calculation of the perimeter
STD above threshold?
y
n
accept candidate point
reject candidate point
(b)

Fig. 6. Flowchart of the image pre-processing (a), and the post-processing (b) algorithms

The task in the post processing algorithm is to separate the still remaining high contrast edges of the clouds from the small singular dark objects. This is done on a way that a circle is drawn around the candidate target, and the standard deviation of the pixel train on the perimeter of the circle is calculated. If we are at a boundary of cloud, then the deviation will be high, because half of the points are in the blue sky, the others are in the cloud. However, when we are in a local flat area (either in front of blue sky or middle of a cloud) the deviation will be low (Fig. 7 c-f). Naturally, the aircrafts located right on a strong edge of a cloud will be lost. This problem is handled by applying tracking.

5 Hardware implementation

The block diagram of the remote airplane detector system is shown in Fig. 8. It contains the cameras an interface board, and FPGA board, a solid state drive. We have selected 5 pieces of 1.2 megapixel grayscale global shutter cameras as the image sources. The angle between the neighboring cameras is 47.5 degrees. The joint image is covers $239^{o}\times66^{o}$ field of view with 1.5 degree of overlap. The entire resolution of the final system will be 4650×1280. (Currently, we use 5 pieces of MBSV034M-FFC type wVGA cameras from Mobisense [12], because the 1.2 megapixel versions are not yet available.) The advantage of this multiple camera system is, that the distortion of the image is negligible compared to a fish-eye objective. Moreover, a high resolution camera with a high resolution, ultra large view angle precision lens would cost tens of thousands USD, and would weight kilos. To be able to hold the cameras in the required position, and avoid cross camera vibration, we have designed and manufactured a solid aluminum camera holder shown in Fig. 9.

An off-the-shelf FPGA card was selected (EXPARTAN-6T) [13], which had an appropriate compact design, and could handle solid-state-disk-drive (SSD). To be able to interface the ribbon cables of the cameras to the FPGA card, we have designed and manufactured an interface board.

The SSD is a key element of the system, which enables to record this high resolution, low distortion test image sequences from different approaching situations.

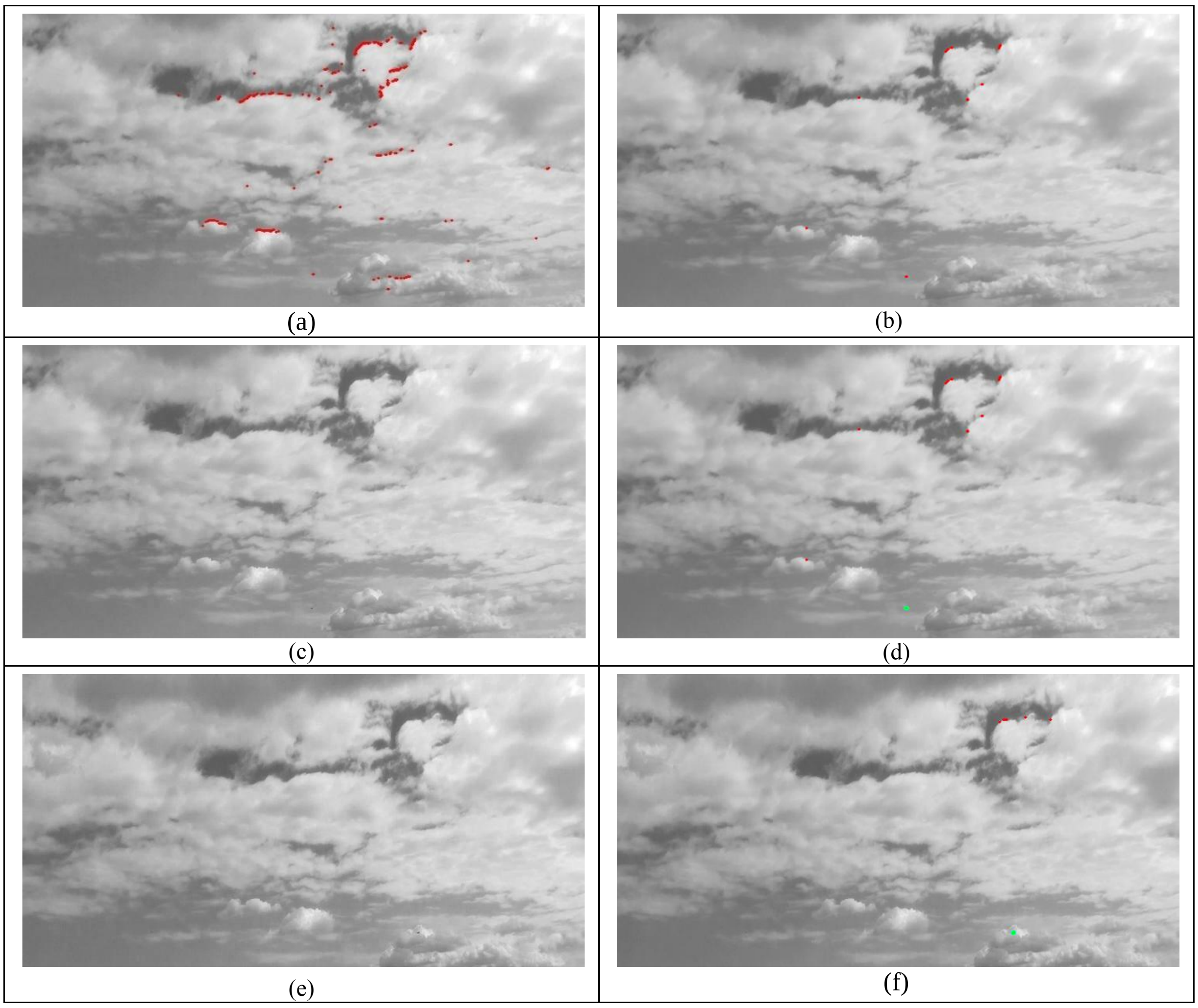
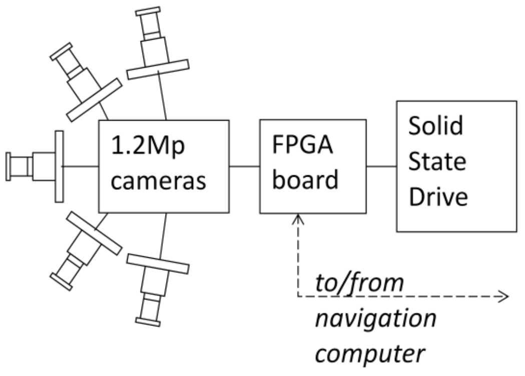


Fig. 7. Large number of candidate points (red) before the correct global threshold value is found (a). Regular number of candidate points (b). Airplane in front of a local clear background (c), and the result of the detection with the post processing (green) (d). Airplane in front of clouds: raw: (e), detection: (f).

5.1 Many-core processor architecture in the FPGA

The on-board image processing system should execute several parallel tasks. Each task of the algorithm has a dedicated execution unit designed for the specific functionality of the task. Operation of the different units is synchronized by a Xilinx Microblaze soft processor core [7,8]. The system can handle several cameras which are connected to the FPGA directly. The processing is done in two steps. First the suspicious objects which are considered to be candidate remote aircrafts are detected during the full-frame preprocessing step. Then, windows containing these objects are cut out from the high resolution images. These windows are called foveas. The foveas including the suspicious objects are further processed by the gray-scale and binary foveal processors in the second step. In this step, the most computationally intensive aircraft identification parts of the algorithm are executed on the foveas only. Block diagram of the on-board image processing and motion estimation system is shown in Fig. 10.



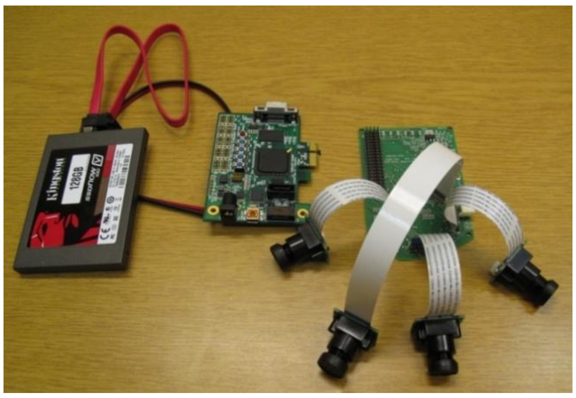


Fig. 8. Diagram of the image capturing, processing, and storing hardware components (upper). The communication with the navigation computer is done by using I^2C bus. The physical hardware components (lower)

5.2 Image preprocessing system

The main task of the image preprocessor part is to provide a physical interface for the cameras, receive pixel data and identify suspicious objects. The main parts of the image preprocessor are shown in Fig. 11. The incoming pixel streams are processed immediately and the results along with the original pixel data are saved into the DRAM memory connected to the FPGA. Depending on the size of the off-chip memory several input frames can be saved for further analysis. Resolution, frame rate and the pixel clock of the cameras do not typically match neither the clock frequency of the Microblaze processor's nor the clock of the memory interface. Therefore the image preprocessor works on three different clocks while synchronization between the clock domains are performed by using FIFO buffers.

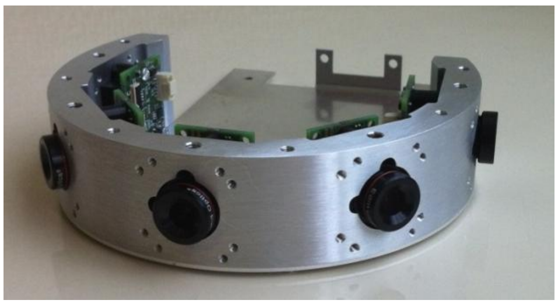


Fig. 9. Solid aluminum camera holder for avoiding cross vibrations

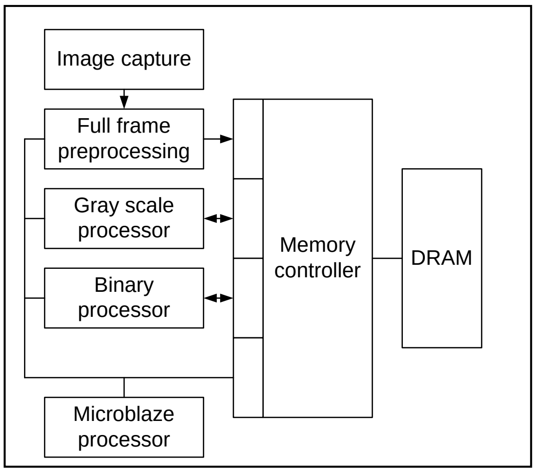


Fig. 10. Block diagram of the proposed image processing architecture

The incoming frames are converted to binary images by using an adaptive threshold operation where the local average calculation can be achieved in either 3×3, or 5×5 or 7×7 windows.

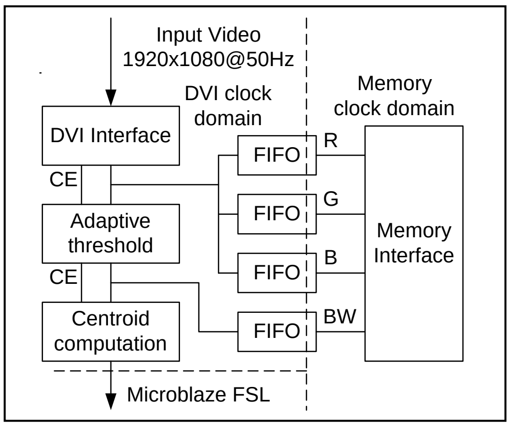


Fig. 11. Architecture of the full frame preprocessor

The next step is detection of the coordinates of the suspicions objects. First, the white pixels are counted in all non-overlapping 32×32 pixel sized parts of the thresholded image. The horizontal and vertical coordinates and the number of the found white pixels are sent to the Microblaze processor where the coordinates of the final 128×128 sized foveas are computed. These foveas are further investigated by the gray-scale and binary processors.

5.3 Grayscale processor

Grayscale operators of the algorithm are performed by the unit shown in Fig. 13. As opposed to the high resolution full frames, which were stored in external DRAM, the 128x128 sized foveas are stored in internal block RAMs (BRAM) of the FPGA. The number of foveas can be configured according to the requirements of the image processing algorithm. Fast offchip DRAM access is provided by a Direct Memory Access (DMA) engine which can cut out the 128×128 sized foveas from the input image. For efficient utilization of the available memory bandwidth, the top left corner of a fovea must fall on a 32 pixel boundary. The on-chip memories should also be accessed by the Microblaze processor to make decisions based on the results of the image processing algorithm. However, the grayscale image processing unit can run on higher clock frequency than the Microblaze processor, therefore the on-chip BRAM memories are used in dual-ported configuration where each port is connected to a different clock domain as it is shown in Fig. 12.

Foveas are processed by an architecture similar to the Falcon emulated digital CNN-UM processor [11]. However, the arithmetic unit of the processor here contains an array of highly optimized Processing Elements (PEs) from which only one is activated during an 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 unit according to the requirements of the image processing algorithm. The utilization of the partial reconfiguration feature of the modern FPGAs makes it possible to load a new mix of PEs to the FPGA without interrupting other functions of the device. This can be very useful for example to adapt to changing visibility conditions on-the-flight.

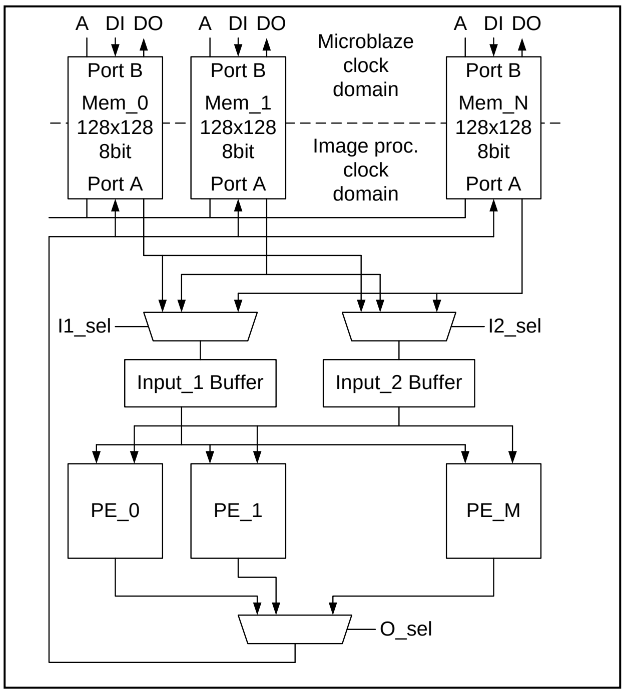


Fig. 12. Architecture of the grayscale processor

The supported operations are diffusion, local average calculation, orientation selective edge detection, thresholding, and arithmetic operations such as addition, subtraction, multiplication and absolute value computation.

Operation of the processor is initiated by the Microblaze processor, by sending an instruction word which contains the address of the three distinct memories (two sources and the target) and one processing element. The input memories provide data to fill up the input buffers where 3×3 sized neighborhood is generated for each pixel. These 9 neighboring pixels are handled in one single step by the processor elements. The result of the selected operator is saved into the third memory. Completion of the operation is indicated by generating an interrupt for the Microblaze processor.

Result of the operation can be quickly evaluated by the Microblaze processor by checking the global status signals. Completely white and black result is indicated by the global white and black signals while steady state of an iterative operation can be checked by the global change signal.

5.4 Binary processor

The architecture of the binary processor is similar to the grayscale processor. Here the internal BRAMs store the 128×128 sized binary foveas also. They are accessible both by the Microblaze and the binary processors as shown in Fig. 14. However, the image processing algorithm requires more binary and morphological operators than grayscale operators, therefore the binary image processor is designed for higher performance. Port A of each BRAM is configured as a 128bit wide data bus and all the pixels in a row are computed in parallel here (Fig. 13.). Due to architectural restrictions of the Spartan-6 FPGA data bus of the BRAMs, one BRAM can provide maximum 36 bits parallel only, therefore four pieces of 18kbit BRAMs are required for each memory blocks. In these blocks, four binary images can be stored at a time. The input buffers store three lines from the binary image and the processing is carried out by a linear array of 128 binary processors.

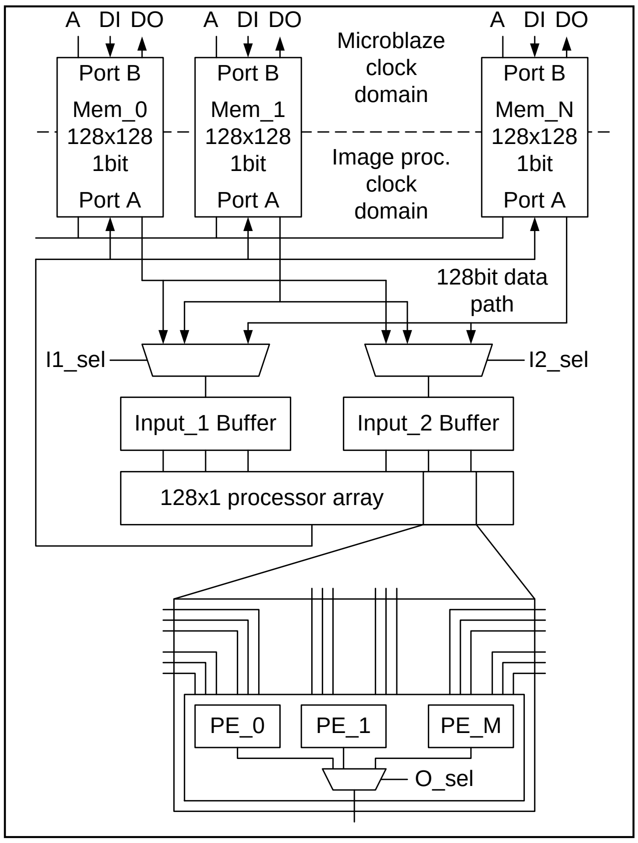


Fig. 13. Architecture of the binary processor

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 white, black and change signals are also implemented.

6 Implementation area, performance

The prototype of the system is implemented on a Xilinx SP605 evaluation board which is equipped with a XC6SLX45T Spartan-6 FPGA, 128MB DDR3 DRAM memory, Gigabit Ethernet interface, one FMC-LPC connector and several other peripherals. Camera signals are connected via an FMC daughter board. Implementation details are described by using an example system configured to handle one HD resolution (1920×1080@50Hz) image flow. In the final system, 5 pieces of 1.2megapixel cameras will be used. Area requirements and power consumption of the full system are summarized on Table. 1.

The system use standard Xilinx IP cores to access devices on the prototyping board such as the on-chip hard memory controller block (MCB DDR3) and the soft Gigabit Ethernet IP core (Soft TEMAC) and other IPs for system level tasks such as clock management, interrupt controller, etc. The Microblaze processor part of the system is running on a moderate 66MHz clock frequency while the on-board DDR3 DRAM chip has a 400MHz clock frequency providing 1.6GB/s peak memory access bandwidth. The full frame preprocessing part is running on 165MHz pixel clock. The grayscale processor is configured to use all PE types described in Section II-B and grayscale foveas can be stored in four on-chip memories. The binary processor also contains all PE types described in Section II-C and four BRAM units where used to store 16 binary foveas. Both units are operating on 150MHz clock frequency which is limited by the BRAMs and the dedicated multipliers of the FPGA.

Each grayscale operation can be executed in 16,384 clock cycles which requires 110μs. This means that about 9100 operations/sec can be reached using 150MHz clock frequency. The grayscale processor consumes relatively small area therefore its performance can be improved by using four arithmetic units if required.

The binary operations can be executed significantly faster, only 0.86μs is needed for one operation. This equals to more than a million pieces of 128×128 sized binary operations in a second!

Assuming three pieces of 16.6Hz HD image flow, more than 180 grayscale and 23,000 binary operations can be executed on each frame. The current image processing algorithm executes 4 grayscale and about 50 binary operations, therefore approximately 45 foveas can be examined on each frame.

The system occupies about one half of the FPGA where the image processing system requires about one quarter of the chip and the Microblaze subsystem, the memory controller and the Gigabit Ethernet MAC can be implemented on one other quarter. The remaining free space makes it possible implement more functions on the FPGA such as collision estimation and trajectory generation.

Table 1. The resource requirements and the power consumption of the system.

Module	**Slice Reg**	**LUTs**	**LUTRAM**	**BRAM/FIFO**	**DSP48A1**	**Power**
MCB_DDR3	1978	2172	45	0	0	0,195
Soft_TEMAC	2636	2452	109	6	0	0,00548
image_proc_0	1737	3388	66	16	0	0,00973
image_proc_gray_0	604	622	100	32	2	0,00462
microblaze_0	1056	1363	113	0	3	0,00507
vga_in_ctrl_0	1751	1896	344	2	0	0,0223
other	1421	1356	48	4	0	0,211
System total	11183	13249	825	60	5	0,454
Spartan-6 XC6SLX45T (available)	54320	27288	6408	116	58	
used	20,59%	48,55%	12,87%	51,72%	8,62%	

Dynamic power consumption of the system is estimated by the Xilinx Power Analyzer and the results are summarized on Table I. Quiescent power consumption of the system is 0.6W which is more than half of the total power consumption. On the FPGA the clock generation and the memory interface require the majority of the power. Total power consumption of the system is around 5W which fits well into the power budget of a UAV.

7 Conclusions

A sense and avoid system for unmanned aerial vehicles was introduced. The system is designed to be able to identify 10 meter sized aircrafts from at least 2000 meters under regular daylight image conditions. The detection is based on visual information, which requires the real-time processing of 120 megabyte/sec of image data. This high performance computation is done by a many-core cellular processor array, which is implemented in a Spartan 6 FPGA.

8 Acknowledgment

The ONR Grants (N62909-11-1-7039, N62909-10-1-7081) is greatly acknowledged.

References

1. http://en.wikipedia.org/wiki/Aircraft_collision_avoidance_systemsAFRL Sense and avoid
2. Debadeepta Dey, Christopher Geyer, Sanjiv Singh and Matt Digioia “Passive, Long-Range Detection of Aircraft: Towards a Field Deployable Sense and Avoid System” Field and Service Robotics, Springer Tracts in Advanced Robotics, 2010, Volume 62/2010, 113-123, DOI: 10.1007/978-3-642-13408-1_11
3. John Lai, Luis Mejias, Jason J. Ford, “Airborne vision-based collision-detection system”, Journal of Field Robotics, Volume 28, Issue 2, pages 137–157, March/April 2011
4. ICAO Doc 4444 (Air Traffic Management)
5. T. Zsedrovits, Á. Zarándy, B. Vanek, T. Péni, J. Bokor, T. Roska, „Collision avoidance for UAV using visual detection” ISCAS-2011, Rio de Janeiro, Brasil
6. Z. Nagy, A. Kiss, Á. Zarándy, B. Vanek, T. Péni, J. Bokor, T. Roska “Volume and power optimized high-performance system for UAV collision avoidance” ISCAS-2012, Seoul, Korea
7. Z. Voroshazi, A. Kiss, Z. Nagy, and P. Szolgay, “Implementation of embedded emulated-digital CNN-UM global analogic programming unit on FPGA and its application,” International Journal of Circuit Theory and Applications, vol. 36, no. 5-6, pp. 589–603, 2008.
8. “Xilinx products homepage,” [Online] http://www.xilinx.com, 2011.
9. B. Vanek, T. Péni, T. Zsedrovits, Á. Zarándy, J. Bokor and T. Roska., "Vision only Sense and Avoid system for small UAVs”, *Am.Control Conference 2011*
10. Ákos Zarándy, Tamás Zsedrovits, Zoltán Nagy, András Kiss, Tamás Roska “Visual sense-and-avoid system for UAVs”, CNNA 2012, Turin, Italy.
11. Z. Nagy and P. Szolgay, “Configurable Multi-layer CNN-UM Emulator on FPGA,” *IEEE Transaction on Circuit and* Systems *I: Fundamental Theory and Applications*, vol. 50, pp. 774–778, 2003.
12. http://www.mobisensesystems.com/pages_en/aptina_modules.html
13. http://www.tokudenkairo.co.jp/exp6t/
14. MIDCAS 2012 Workshop: http://www.eda.europa.eu/info-hub/publications