CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2016; **00**:1–14
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Real-time surveillance detection system for medium-altitude long-endurance unmanned aerial vehicles

A. Amanatiadis*, L. Bampis, E. G. Karakasis, A. Gasteratos and G. Sirakoulis

All the authors are with the School of Engineering, Democritus University of Thrace, GR-67132, Xanthi, Greece

SUMMARY

The detection of ambiguous objects, although challenging, is of great importance for any surveillance system and especially for an Unmanned Aerial Vehicle (UAV), where the measurements are affected by the great observing distance. Wildfire outbursts and illegal migration are only some of the examples that such a system should distinguish and report to the appropriate authorities. More specifically, Southern European countries commonly suffer from those problems due to the mountainous terrain and thick forests that contain. UAVs like the "Hellenic Civil Unmanned Air Vehicle - HCUAV" project have been designed in order to address high altitude detection tasks and patrol the borders and woodlands for any ambiguous activity. In this paper, a moment-based blob detection approach is proposed that utilizes the thermal footprint obtained from single infrared (IR) images and distinguishes human or fire sized and shaped figures. Our method is specifically designed so as to be appropriately integrated into hardware acceleration devices, such as GPGPUs and FPGAs, and takes full advantage of their respective parallelization capabilities succeeding real-time performances and energy efficiency. The timing evaluation of the proposed hardware accelerated algorithm's adaptations shows an achieved speedup of up to 7 times, as compared to a highly optimized CPU-only based version.

Received . . .

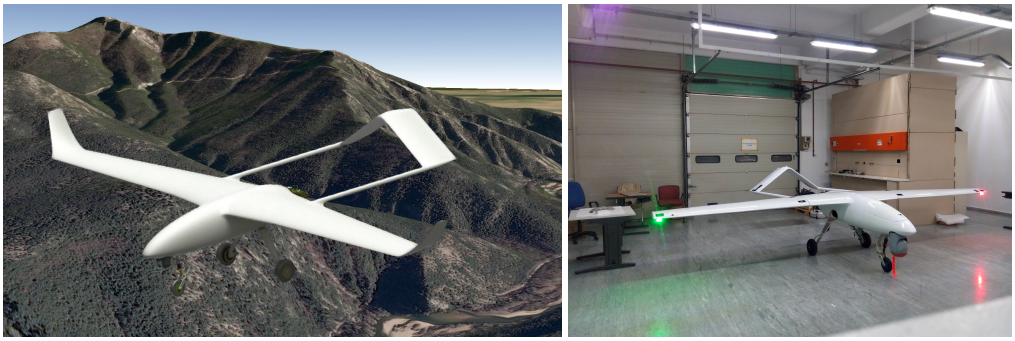
KEY WORDS: Aerial surveillance detection system; Low-power implementation; UAVs; GPGPU; FPGA

1. INTRODUCTION

The use of images and video obtained by Unmanned Aerial Vehicles (UAVs) patrolling harsh or difficult to access territories, such as steep mountains or dense forests, can be beneficial for early fire detection, illegal immigration prevention and search and rescue operations. With the technological advancements in the field of UAVs, it is possible for the corresponding agencies of countries that face problems such as wildfires and illegal immigration to obtain and operate a small number of UAVs assigned to patrolling areas of interest. In this paper, we present algorithms developed to automate human and fire detection using images acquired by a UAV equipped with full-color and thermal cameras.

In the existing bibliography, human detection from aerial imagery either makes use of tracking moving objects [1, 2] or depends on detecting the distinct shape of the human figure [3, 4]. In [1] the detection of moving humans is based on dense optical flow, using the deviation of all pixels from the anticipated geometry between succeeding images to differentiate between background and moving objects, while tracking-by-detection [2] is used to increase the probability of a positive human detection.

*Correspondence to: aamanat@ee.duth.gr



(a) (b)

Figure 1. The Hellenic Civil Unmanned Aerial Vehicle. a) Conceptual operation; b) Developed prototype.

The Head-Shoulders-Torso (HST) model [3] is used in order to determine whether a detected blob is actually a human. However, in order for this model to be effective, the images depicting the possible targets have to be at an angle, so that the human anatomy is visible. Cascaded Haar classifiers [4] are used to detect humans in static images from UAVs taken at a 45 degree angle to the ground plane, making use of the distinct human figure. Finally, the characteristics of the shadow cast by objects can be used in order to determine if it corresponds to a human [5].

Moreover, infrared images can be used to enhance the performance of the algorithms, especially if the background is homogeneous in terms of temperature variation. In [4] infrared images are used in combination with full-color ones in order to cross-reference results and eliminate false positives. Infrared images are used in [6], in order to detect pedestrians in non-aerial images. In [7] regions of interest are extracted from infrared photographs and then pedestrian detection is performed in the corresponding region of a full-color image of the same area.

Fire detection on color images is based on chromatic features, since the distinct orange color of the flame is easy to detect. Using color data extracted from images of fire [8, 9, 10, 11], as well as the shape of the flames, highly efficient fire detection can be achieved. Additionally, detection of the smoke plume created by a fire can help the early detection and prevent its spread [12]. Moreover, using infrared images to detect fires can be very effective, due to the high temperature of the flames [13, 14]. Since in this work we are addressing the problem of fire detection in cases of middle altitude flights, where the weather or environmental conditions (e.g. clouds or dense vegetation of high trees) may affect the color optical information, only the infrared spectrum is utilized.

This paper as an extension of [15] includes three different implementations of the proposed algorithm and is structured as follows. Section 2 is dedicated to the HCUAV project, the UAV system for which our method was developed. In Section 3 some of the required notions for the rest of the presented work are explained. Our whole detection approach is described in Section 4, while Section 5 contains the implementation and parallelization techniques applied for the three optimized versions of the presented algorithm. Section 6 presents our experimental results and finally, Section 7 draws our conclusion.

2. HCUAV PROJECT

Since UAVs are mainly characterized by operational altitude and flight endurance, there are three main classes in which can be categorized: 1) Micro Aerial Vehicles (MAVs), 2) Medium Altitude, Long Endurance (MALE) and 3) High Altitude, Long Endurance (HALE) vehicles. Between them, MAVs present limited (low altitudes (up to 350 m) and battery capacities (5-30 min)) but promising performance. They have been successfully utilized in real life missions supporting rescue operations and damage analysis. On the other hand, MALE vehicles are characterized by significantly higher altitudes (up to 9000 m) and many hours of flight operation covering extended regions and having

DOI: 10.1002/cpe

(a) (b)

Figure 2. The HCUAV electro-optical airborne sensor system. a) Infrared and day camera gimbal with its electronics compartment; b) Actual test flight.

a key role in defense, security and surveillance operations. Finally, the class of HALE vehicles is characterized by the largest and most sophisticated UAVs, which are capable of flying at heights up to 20.000 m or more and operational duration that is measured in tens of hours, while they can be used for gathering data at global scale.

The HCUAV project [16] aims to merge the best characteristics of MAVs and HALE vehicles for developing a cost efficient MALE UAV with operational flight altitude up to 2000 m, which is going to combine comparatively low-cost payload and sophisticated image processing algorithms for accurate remote sensing and surveillance [17]. The main objective of this project is fire detection in forested areas and border control thus, its characteristics such as flight duration, covered region, and flight velocities were adapted correspondingly. Regarding the layout, a propeller-driven pusher configuration with a boom-mounted inverted V tail carrying an internal combustion engine as shown in Fig. 1 has been selected. In contrast with HALE vehicles where the operational altitude remains static, the corresponding altitude of HCUAV turns out to be a 3D mission path considering the mountainous nature of the regions of surveillance in conjunction with the algorithm's requirements.

For the electro-optical sensor system the CONTROP TR-STAMP gimbal was used which entails a three axis gyro stabilized system along with an infrared and a day camera as shown in Fig. 2. The control of the gimbal was achieved through the VISCA protocol based on the RS232 serial communication. The analog video output of the sensor gimbal was encoded using the AXIS Q7424-R video encoder.

3. PREREQUISITES

3.1. Ground Sample Distance (GSD)

The ground sample distance (GSD) is defined as the distance between the projections to the ground of two consecutive pixels centers and is directly related to the camera's spatial resolution. Let us assume a camera with a sensor resolution $M \times N$. Let us further assume that the character L represents either the vertical (M) or the horizontal (N) sensor resolution. Then the GSD that corresponds between the pixels $x-1$ and x is represented as $GSD(x)$, $x \in [1,\ L-1]$ (expressed in centimeters) and is given by Eq. (1).

$$GSD(x) = H \cdot [T(x) - T(x-1)] \cdot 100 \tag{1}$$

$$T(x) = \tan\left(\frac{FOV}{2} \cdot \frac{2x-L}{L} + \beta\right) \tag{2}$$

DOI: 10.1002/cpe

where FOV (in radians) is the field of view (horizontal or vertical), H (in meters) is the height of the camera above the average ground elevation and β (in radians) is the angle between the camera's principal axis and the plumb line (i.e. the angle of tilt).

In the general case, a human area could be represented by a blob with $\alpha_l \cdot \left(\frac{\text{Human Height (cm)}}{GSD(\frac{x}{2})} \times \frac{\text{Human Width (cm)}}{GSD(\frac{x}{2})} \right)$ pixels and when the camera has vertical orientation, this area could be represented by a blob of $\alpha_s \cdot \left(\frac{\text{Human Width (cm)}}{GSD(\frac{x}{2})} \right)^2$ pixels, where α_l and α_s are free parameters that depend on the problem. Although that this is only a gross approximation, it could efficiently serve as a size threshold for discarding small and large objects.

3.2. Chebyshev Image Moments

Every $M \times N$ image can be perceived as a piecewise continuous function f, which can be approximated using polynomial-based spectral expansions, as it can be seen in the following eq. (3):

$$f(j,i) \approx \sum_{k=0}^{N-1} \sum_{t=0}^{M-1} Kernel_{kt}(y_j, x_i) \cdot C_{kt} \tag{3}$$

where $Kernel_{kt}(y_j, x_i)\cdot$ is a separable basis function and $\{ C_{kt} | \, k \in [0,\, N-1] \,,\, t \in [0,\, M-1] \}$ is the set of the expansion coefficients. In this work the continuous Chebyshev polynomials [18] of the first kind, given by eq. (4), are used as the basis orthogonal set $\{ P_k(x) | \, x \in [-1,\, 1],\, k \in [0,\, max(N,\, M)-1] \}$ in order to calculate the expansion coefficients $\{C_{kt}\}$ (or else the Chebyshev image moments of order $(k,\, t)$) .

$$C_{kt} = W_k \cdot W_t \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} P_k(x_i) \cdot P_t(y_j) \cdot f(j,i) \tag{4}$$

where W_k and W_t are weighting factors, while

$$P_{k+1}(x) = 2xP_k(x) - P_{k-1}(x)$$

$$P_0(x) = 1 \;\;,\;\; P_1(x) = x$$

and

$$W_k = \begin{cases} \frac{1}{N} & k = 0 \\ \frac{2}{N} & 1 \le k < N \end{cases} \;\;,\;\; W_t = \begin{cases} \frac{1}{M} & t = 0 \\ \frac{2}{M} & 1 \le t < N \end{cases}$$

$$x_i = cos\left(\pi \frac{i + \frac{1}{2}}{N} \right) \;\;,\;\; i = 0, 1, ..., N-1$$

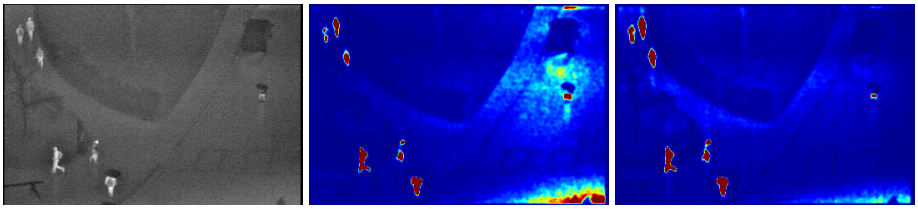
$$y_j = cos\left(\pi \frac{j + \frac{1}{2}}{M} \right) \;\;,\;\; j = 0, 1, ..., M-1$$

Calculating the Chebyshev image moments C_{kt} (i.e. the expansion coefficients of the Chebyshev spectral expansion of the image f) is a computationally demanding process. Except the direct way, there is a number of different implementations of eq. (4) that could be selected in order to achieve a fast computation scheme. The simpler and most straightforward implementation in order to reduce the number of multiplications is expressed by the formula:

$$C_{kt} = W_k \cdot W_t \cdot \sum_{i=0}^{N-1} P_k(x_i) \sum_{j=0}^{M-1} P_t(y_j) \cdot f(j,i) \tag{5}$$

which in fact exploits the separable nature of the basis function and it is very probably the most frequently implemented form. An alternative implementation could make use of image block or slice representation [19], which, taking into consideration the distribution of pixel intensities, could

DOI: 10.1002/cpe



(a) (b) (c)

Figure 3. The resulted enhanced images from (b): the wavelet-based method used in [20] and (c): the proposed moment-based method. The original image is illustrated in (a).

achieve even faster calculation speeds. In this work, we select to implement a matrix-based version of eq. (5), which is expressed in eq. (6):

$$C = \mathcal{W}. * \left[A \cdot (B \cdot f)^T \right] \tag{6}$$

where $(\,)^T$ represents the transpose of a matrix, $.*$ represents the element-wise matrix multiplication, A is a $n \times N$ matrix, the elements of which are given by $A_{ki} = P_k(x_i)$ (where $0 \leq n, i < N$ and $0 \leq k < n$), B is a $m \times M$ matrix, the elements of which are given by $B_{tj} = P_t(y_j)$ (where $0 \leq m, j < N$ and $0 \leq t < m$), $\mathcal{W}$ is a $n \times m$ matrix the elements of which are given by $\mathcal{W}_{kt} = W_k \cdot W_t$, f is the image matrix with dimensions $M \times N$ and $C = [C_{kt}]$ is the resulting $n \times m$ matrix of the image moments. This form has been selected due to its ease of extension into GPGPU programming. In the same context, the image reconstruction of eq. (3) can be represented in matrix form as

$$\hat{f} = (C \cdot B)^T \cdot A \tag{7}$$

3.3. Moment-Based Image Contrast Enhancement

In the previous subsection, the calculation of the Chebyshev image moments has been presented. In this one, an image contrast enhancement technique, which is based on Chebyshev moments, is going to be explained. The used method has been based on a similar methodology [20], where wavelets are used instead of moments. Wavelet-based methods have proven effectiveness in contrast enhancement. However, their hierarchical structure (levels of coefficients that correspond to different frequency bands; the maximum number of levels depends on how many times an image can be downscaled by a factor of 0.5) does not allow much freedom on properly selecting low-frequency components. Based on this fact, a moment-based perspective, which presents similar results with the wavelet-based one and leads to a more flexible scheme, is adopted. Fig. 3 presents indicative outputs of the wavelet- and moment-based methods for contrast enhancement. The steps of this method are presented below:

Step 1: Initially, the Chebyshev moments of the $M \times N$ image f are calculated up to order (n, m) using eq. (6), resulting in a $n \times m$ matrix C.
Step 2: The second step is to make zero the low order moments which represent low-frequency information. All the moments up to order (O_n, O_m) are changed to zero, that is $C_{kt} = 0 \ \forall \ 0 \leq k \leq O_n$ and $0 \leq t \leq O_m$.
Step 3: A new $M \times N$ image $\hat{f}$ is reconstructed by using the inverse transform illustrated in eq. (7) and is normalized to range $[0, 255]$.
Step 4: The reconstructed image matrix $\hat{f}$ is used in order to define a new $M \times N$ matrix as a mask. This matrix is given by eq. (8) and (9).

$$mask_{ji} = \frac{G_{ji}}{\max(G_{ji})} \tag{8}$$

$$G_{ji} = e^{\alpha \cdot \hat{f}(j,i)} \tag{9}$$

DOI: 10.1002/cpe

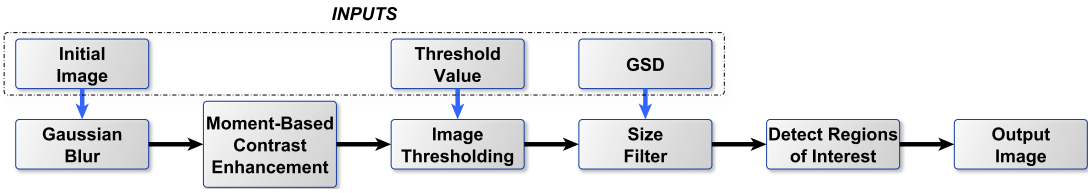


Figure 4. Human and Fire Detection Algorithm.

Algorithm 1 Steps of Detection Algorithm.

Require: $\boldsymbol{I}$: Initial Image
Require: $\boldsymbol{Th}$: Threshold Value
Require: $\boldsymbol{GSD}$: Ground Sample Distance
1: $\boldsymbol{I_{gb}} \leftarrow$ Gaussian_Blur($\boldsymbol{I}$)
2: $\boldsymbol{I_{ce}} \leftarrow$ Moments_Based_Contrast_Enhancement($\boldsymbol{I_{gb}}$)
3: $\boldsymbol{I_{th}} \leftarrow$ Thresholding($\boldsymbol{I_{ce}}, \boldsymbol{Th}$)
4: $\boldsymbol{I_{sf}} \leftarrow$ Size_Filter($\boldsymbol{I_{th}}, \boldsymbol{GSD}$))
5: $\boldsymbol{I_{dr}} \leftarrow$ Detect_Regions_of_Interest($\boldsymbol{I_{sf}}$)
return $\boldsymbol{I_{dr}}$

where α is an application oriented parameter and it should be selected empirically. For the purposes of this work, the value $\alpha = 0.02$ is used.
Step 5: Finally, the enhanced image f_E is calculated according to eq. (10).

$$f_E(j,i) = f(j,i) \cdot mask_{ji} \tag{10}$$

4. PROPOSED SURVEILLANCE DETECTION TECHNIQUE

This section aims to present the algorithm which is used for human and fire detection. The algorithm is composed of five main steps as it can be seen in Fig. 4 and Alg. (1), while its output is illustrated in Fig. 5. Those steps are explained as follows:

Step 1: The first step is the Gaussian blurring of the input thermal image in order to produce smoother blob borders.
Step 2: The second step applies contrast enhancement, which is based on the Chebyshev image moments, to the filtered image. This step results in an image in which objects with high temperature are highlighted.
Step 3: The third step applies a threshold ($\boldsymbol{Th}$) to the intensities of the enhanced image and produces a binary output where all the pixels intensities bellow this threshold are set to zero, while the rest ones are set to one.
Step 4: In the fourth step a size filter, which takes into account the Ground Sample Distance (GSD), is applied in order to estimate the larger and the smaller object size for keeping only human or fire-sized objects.
Step 5: Finally, in the last step human or fire motion characteristics are further considered for increasing the possibility of detecting specific regions of interest, as to be further explained in the following subsection.

4.1. Human and Fire Detection

The human detection process using images taken by a UAV at heights up to two kilometers is a challenging task mainly due to human-shape restrictions that are based on the target's projection to image plane in conjunction with size and payload restrictions. Let us assume a camera sensor

DOI: 10.1002/cpe

(a) (b)

Figure 5. Result of human detection algorithm.

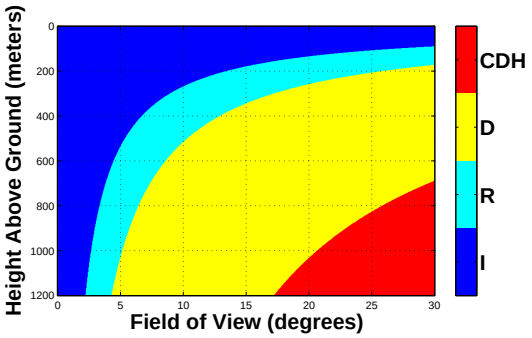


Figure 6. Human Detection (D, yellow), Recognition (R, cyan) and Identification (I, blue) for a range of FOV and flight heights according to Johnson's criteria. In the red (CDH) area, no human-sized object can be detected. The assumed camera has resolution 480×720, the tilt angle is $\beta = 0^o$ and the GSD is calculated using eq. (1) for the camera's principal point.

with pixel pitch $3.5\,\mu m$, resolution 480×720 pixels and focal length $\mathcal{F} = 28\,mm$, that is on-board in the UAV at height $H = 1200$ meters above the average ground elevation, the field of view of which is calculated by $FOV = 2 \cdot \tan^{-1}\left(\frac{d}{2 \cdot \mathcal{F}}\right) = 2 \cdot \tan^{-1}\left(\frac{\frac{3.5}{1000} \cdot 720}{2 \cdot 28}\right) \approx 5.15^o$, where d is the horizontal or vertical sensor size. Assuming that the camera's Principal Point (PP) is located at the image coordinates $(240, 360)$ and that the tilt angle is $\beta = 0^0$ (vertical orientation), the ground sample distance for its principal point is approximately $GSD_{pp} \approx 15\,cm$ according to eq. (1), with $L = 720$ and $x = 360$. Considering that an average human size according to Johnson's criteria [21] is $180cm \times 50cm$ and taking into account the vertical camera's orientation, a human is projected to the image plane as a blob that consists of approximately $\left(\frac{\text{human width (cm)}}{GSD_{pp}}\right)^2 = \left(\frac{50}{15}\right)^2 \approx 11$ pixels. Additionally, taking into consideration Johnson's criteria which assume that for detection, recognition and identification 2, 8 and 16 pixels/meter are required, respectively. The described predefined payload is capable of detecting human-sized objects on the basis of their emitted thermal radiation. This example makes clear that the human shape characteristics cannot be used, at least directly, for detecting and recognizing human figures. Figure (6) presents some combinations of FOV and H, where the detection, recognition or identification of a human-sized object may or may not be achieved for the previously mentioned camera. This is the reason why the proposed algorithm has been designed mainly as a blob detector applied to thermal images and is kept simple enough in order to achieve high frame rates when implemented on a GPGPU or FPGA unit.

From high altitudes, humans and other human-sized objects, like big animals, are projected as hot spots or small bright blobs (small-sized objects with obscure shape characteristics) on the image plane, where their thermal footprint differs from the background. Although there is a difference between the detected temperature that corresponds to fire areas and that of human objects (which in fact is taken into consideration when the thresholding procedure takes place in order to separate hot objects from their surrounding area), the corresponding blob's size is an additional important feature for separating human- from fire-sized objects, since in forested areas the later cover tens

DOI: 10.1002/cpe

of m^2. For the case of human detection, the size thresholds have previously been mentioned (see subsection 3.1), for the case of fire detection the size threshold uses a predefined area as the smaller detected region of interest. This area can be calculated in pixels using the formula $\alpha_f \cdot \left(\frac{\text{area length (cm)}}{GSD_{pp}}\right) \cdot \left(\frac{\text{area width (cm)}}{GSD_{pp}}\right)$, where a_f is a free parameter that is dependent on the problem.

5. ALGORITHM'S IMPLEMENTATIONS

One of the main contributions of this work is the capability of the proposed algorithm to be implemented on hardware accelerators. In this section, two parallel versions of the technique are described, together with a reference point of a serialized one.

5.1. CPU-only Implementation

In order to create a valid reference point for comparing the proposed algorithm's accelerated implementations, a CPU-only based version was developed. More specifically, all the steps described in Section 4 were computed, making use of every available CPU core (one thread per core) of the given device and utilizing several highly optimized programming libraries . We made use of the Boost library [22] in order to create a multi-threaded architecture. In addition, the Eigen [23] and OpenCV [24] libraries were utilized, undertaking every linear algebra and computer vision functionality respectively. Using this CPU-only version, we were able to profile the proposed methodology, identify its bottlenecks and properly address them by the assessed hardware accelerators (GPU and FPGA).

5.2. CPU and GPU Implementation

As it has already been mentioned, the overall algorithm's architecture have been designed in order to be efficiently implemented on a GPGPU. In this work, the general purpose parallel computing architecture of CUDA was selected as a means of assigning calculations to the GPU. In general, the architecture of a GPU complies with the "Single Instruction on Multiple Threads" (*SIMT*) model[†], which essentially means that each one of the available parallel processing units is restricted to execute the same instruction on every clock circle. Violating the above rule, causes a serialization of the individual processing threads and leaves the majority of the available resources unoccupied. CUDA provides several programming utilities, each of which can assist the efficiency of a given implementation, with their description available in [25].

The first step of the presented detection approach includes the application of a Gaussian blurring filter on each given thermal image. In the discretized space of an image, such a blurring can be achieved by applying a Gaussian convolution kernel, of size 3×3, over the input frame's pixels. A straightforward implementation of such a filtering approach is described in [26, 27]. According to that, the 3×3 kernel is firstly divided into two one-dimensional filters ($F_1 \in \mathbb{R}^{3\times1}, F_2 \in \mathbb{R}^{1\times3}$). Then, the Gaussian blurring can be implemented by recursively applying the convolution operation between the input image and each individual kernel F_1 and F_2. The aforementioned approach, although scales well for a GPGPU-based implementation, involves a lot of redundant and misaligned memory access (especially in the case of the F_1 convolution). Thus, here we adopt a more efficient technique by taking advantage of the CUDA Shuffle Functions introduced by Kepler GPU architecture. Using those Functions, the GPU threads are capable of sharing register values as long as they belong to the same warp. Thus, N individual threads are assigned for processing in parallel each pixel of the image's i-th row, accessing the corresponding pixel value, applying the respective filter kernel weight and passing that product to their neighboring threads. This way, the application of the horizontal filtering component is achieved without employing any of the higher latency global or shared GPU memory. The same procedure is then executed for the next image row $(i + 1)$ by the

[†]The extension of the more traditional "Single Instruction on Multiple Data" (*SIMD*) model.

DOI: 10.1002/cpe

same threads, only this time, the corresponding filtered image values from row i are still stored in the used registers, liberating once more our implementation from the requirement of accessing the slower types of memory. This technique is utilized here for addressing the Gaussian blur filtering, though, in fact, it can also be extended to any kind of 2-dimensional kernel applications.

In order to enhance the blurred image's contrast, the second step of our model makes use of Chebyshev moments. Using eq. (6), the calculation of the image moments C is straightforward since it consists of simple linear algebraic operations. Given a specified image resolution, the terms A and B can be considered as constants. Thus, we choose to pre-calculate and store in the GPU's texture memory the terms $\mathcal{W}$, A and B^T, meaning that the operations executed on-line for every input image are downgraded into two matrix multiplications, one transposition and one element-wise multiplication ($C = \mathcal{W}. * \left[A \cdot f^T \cdot B^T\right]$). Note that the on-line image transposition can be efficiently implemented using [28]. The basic idea behind this approach is based on simple read and write operations with each parallel thread undertaking the transposition of multiple matrix elements. This way, the workload of the thread management unit is reduced, while still ensuring a full device occupancy. Initially, the given matrix is separated into tiles of equal size, each of which is assigned to an individual GPU block of threads. In addition, each of those blocks retains a two-dimensional chuck of shared memory, which refers to a GPU memory type of low latency. Thus, every thread is responsible for copying one element from every tile into the shared memory and subsequently re-assigning it to its appropriate transposed location of the resulting matrix. This two-step approach, although seems to increase the required computational steps, is adopted in order to avoid the high-latency memory accesses of misaligned data. After the required transpositions, the terms corresponding to moments of up to (O_n, O_m) order are zeroed using one thread for each respective value. This zeroing operation is not ideal, since the accessing memory alignment restrictions are not absolutely met, though it is still much faster than a serialized version of the procedure.

Subsequently, we need to reconstruct image $\hat{f}$. Equation (3) may seem really costly in terms of computations but it can take great advantage of a parallel implementation. Since each value of matrix C_{kt} needs to be accessed multiple times, we choose to bind a texture memory over the buffer containing C_{kt}. This texture is binded only once on the beginning of our on-line application and provides lower latency to the read operations. Then, the parallel reduction algorithm is applied, in order to achieve the required 2-dimensional summation, through the means of CUB library [29]. CUB is a highly optimized CUDA-based library that provides several functionalities, while it is tuned for multiple versions of GPU architectures. Thus, the reduction is achieved by a combination of the CUDA Shuffle Functions, Grid Stride Loops and Atomic operations, all of them specifically optimized so as to achieve the maximum possible throughput. In addition, a parallel version of eq. (7) was developed, making use of the same transposition technique applied for eq. (6) and described above, in order to assess every possible optimization scenario. Through our experiments, we concluded that the first approach achieved a higher speedup and thus adopted by our method. Finally, during the above calculations, we keep track of the maximum and minimum values achieved and we normalize the aforementioned products in the range of $[0, 255]$.

With the reconstructed image $\hat{f}$ in hand, we now proceed to the calculation and application of the enhancement mask. Looking at eq. (8) it is clear that the maximum value of G_{ji} needs to be computed. Since the maximum value of $\hat{f}$ ($\hat{f}_{max}$) is already obtained by our previous steps, we make use of eq. (9) and we calculate $max(G_{ji}) = e^{\alpha \cdot \hat{f}_{max}}$. Finally, the thresholded image f_{th} is calculated by the same kernel, through the concatenation of eq. (8) to (10) and using $M \times N$ threads, as:

$$
\begin{aligned}
f_E(j,i) &= f(j,i) \cdot \frac{e^{\alpha \cdot \hat{f}(j,i)}}{\max(G_{ji})} \\
f_{th}(j,i) &= \begin{cases} 1 & f_E(j,i) > \boldsymbol{Th} \\ 0 & f_E(j,i) \leq \boldsymbol{Th} \end{cases}
\end{aligned}
\tag{11}
$$

The rest of the proposed method's steps require the formulation of pixel groups with values equal to 1 in order to obtain the human or fire-sized objects. This procedure does not comply with the

DOI: 10.1002/cpe

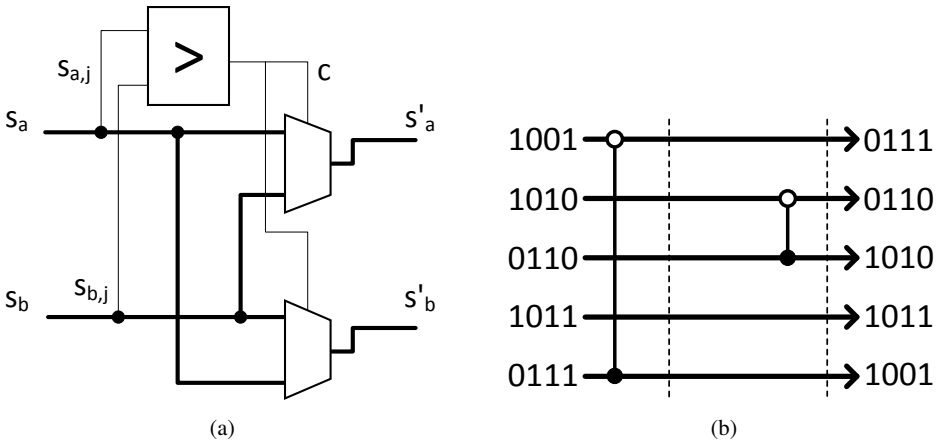


Figure 7. Blob size comparison hardware implementation: a) basic compare-swap element b) pipelined MSB sorting network, where the operator represents the activated compare-swap elements.

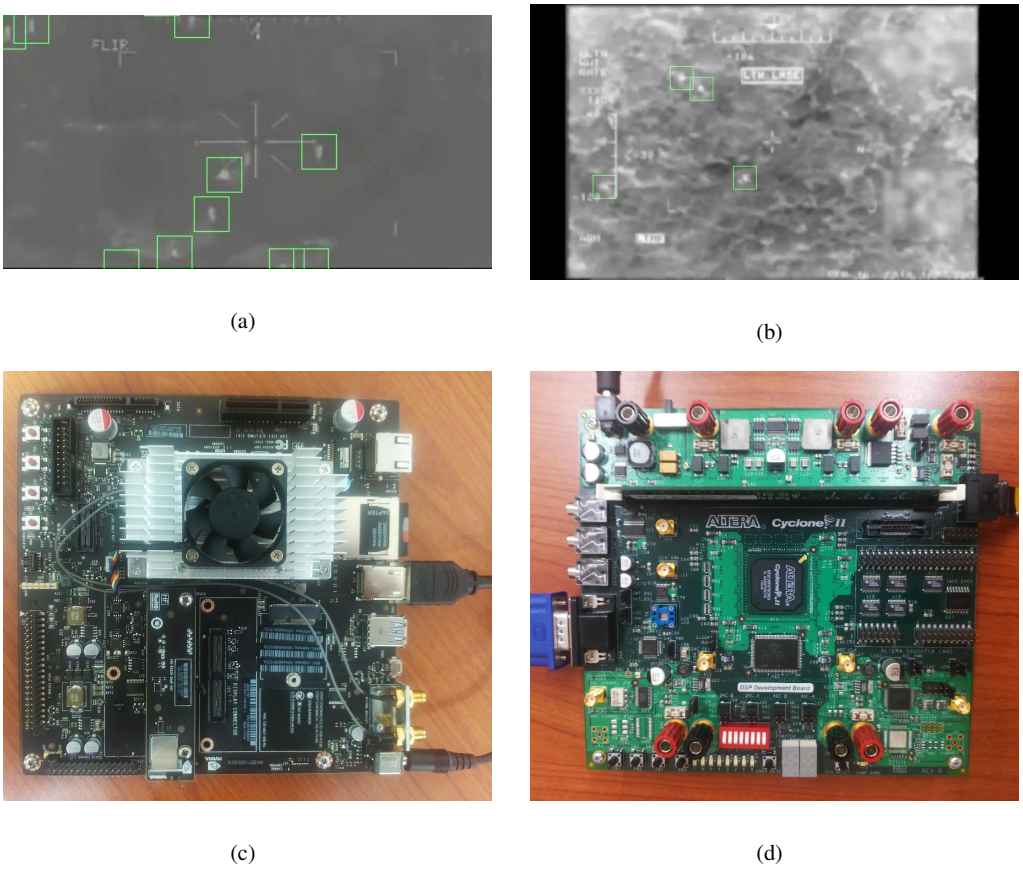
parallelization properties of a GPU, since it requires non-neighboring threads to communicate constantly so as to determine whether their corresponding values f_{th} should be considered as members of a specific object. Following the implementations described in [30, 31, 32] we adopt a pipeline scheme and we make use of both available processing units of a given system, e.i. the CPU and the GPU. More specifically, given a constant image stream, at the same time that the GPU is executing all the required steps described above for the input image I_k, the CPU is operating on the previous image I_{k-1} and creates the human or fire-sized objects, applies the size filtering and additionally checks for the requirements presented in Section 4.1. Without this technique, the CPU would be restricted to remain idle, waiting for the GPU to return the computed results. In this way though, a more efficient algorithm architecture is provided, allowing all the system's processing units to be simultaneously utilized.

5.3. FPGA Implementation

Since memory and computational resources are very limited due to the volume and power restrictions of the HCUAV, a dedicated hardware for the surveillance detection was also considered as an alternative option. As a result, an efficient implementation on a Field Programmable Gate Array (FPGA) is also proposed and realized in a low-cost development board. FPGA-based computation engines appear to be very attractive for such unmanned aerial detection systems, since the fundamental image processing steps for such systems, comprise a uniform structure composed of many finite state machines, thus matching the inherent design layout of FPGA hardware. Furthermore, tests based on double-precision floating point operations' (add/multiply) performance showed that FPGAs are competitive to multicore processors. More specifically, an AMD dual-core 2.8 GHz Opteron processor has a peak of 11.2 Gflop/s and an Intel 3 GHz quad-core Xeon has a peak of 24 Gflop/s. However, it has been evaluated that theoretically Xilinx Virtex4 LX200 and Altera Stratix II EP2S180 can perform 15.9 and 25.2 Gflop/s, whereas Xilinx Virtex5 LX330 and Altera Stratix III EP3SE260 perform 28 and 50.7 Gflop/s, respectively. Finally, as FPGAs can be completely dedicated to a particular function, in several cases they are more energy efficient than general-purpose CPUs. Therefore, although each implementation has its particular features, the merits of surveillance applications on FPGA platforms along with low power consumption, compactness, and portability, completely justify in many cases such an option [33].

The proposed detection system was designed, compiled and simulated using the software package Quartus II Programmable Logic Development System of Altera Corporation. The device chosen was the Cyclone II EP2C45 utilizing the accompanying DSP development board.

DOI: 10.1002/cpe



(a) (b)

(c) (d)

Figure 8. The two proposed and evaluated implementations. a) and c) The CPU+GPU parallel implementation running on the Jetson TX1 Embedded System Developer Kit; b) and d) The FPGA hardware implementation running on the Cyclone II DSP Development Board.

The proposed architecture was based on a sequence of pipelined stages in order to reduce computational time [34]. Parallel processing has been employed wherever was possible in order to further accelerate the process. The video stream constitutes the input data of the system, with the Gaussian blurring. A line buffer loads and stores each time two lines of the input image in order to perform the blurring. The filter window block calculates the filter mask area and the process is performed only once, in the first step of the process. However, the filter window coefficient unit calculates each time the new filter mask coordinates and the new spatial coverage for the contrast enhancement. The filter window coefficients unit renews the four coordinates of the filter windows vertex along the horizontal direction.

In the next step, the source pixel intensity values are imported in parallel into the next unit together with threshold number for producing the binary image output. The purpose of this unit is to compare the pixel's intensity together with the threshold number.

For the subsequent size filtering, a novel sorting method is proposed. Since fires are considered as an upper threshold bound and humans as a lower threshold bound, a sorting method of all the detected pixel groups can be applied. Let $s_a = \{s_{a,k-1}, s_{a,k-2} \ldots, s_{a,1}, s_{a,0}\}$ and $s_b = \{s_{b,k-1}, s_{b,k-2} \ldots, s_{b,1}, s_{b,0}\}$ be the two k-bit non-negative binary values denoting the size of the detected blob, where $s_{a,j}$ and $s_{b,j}$ denote the j-th bit of s_a and s_b respectively. A bit-wise analysis can define the relationship between $s_{a,j}$ and $s_{b,j}$ of $s_{a,j} > s_{b,j}$ if $s_{a,j} =$'1' and $s_{b,j} =$'0' (and vice-versa) in case that the first $(k-j-1)$-th Most-Significant-Bits (MSBs) have the same value. The proposed implementation performs a recursive compare operation between the same bits of each $s_{a,j}$ and $s_{b,j}$ values from the MSB to Least-Significant-Bit (LSB). The basic sorting element used is

DOI: 10.1002/cpe

Table I. Tested devices' specifications

	Cyclone II DSP Development Board	Jetson TX1 Development Board
Weight	331 gr	474 gr
Power Consumption	5-60 W	5-20 W
Clock Frequency	100 Mhz	CPU: 1734 Mhz / core GPU: 72 Mhz / core
Available Processing Units	68,416 $Logic\ Elements\ (LEs)$ 150 $Embedded\ 18 \times 18\ multipliers$	4 $CPU\ cores$ 256 $CUDA\ cores$

Table II. Average execution time for each one of implemented algorithm's versions.

	Average Time (ms/frame)					
	CPU-only		CPU+GPU		FPGA	
	800x600	1280x1024	800x600	1280x1024	800x600	1280x1024
1) Gaussian Blurring	10.6	22.9	0.2	0.5	0.1	0.3
2) Moments Based Contrast Enhancement	43.3	109.3	3.1	7.9	6.8	15.8
3) Thresholding	2.8	6.7	0.1	0.3	0.0	0.0
4) Size Filter	7.2	15.4	7.2	15.6	3.2	6.9
5) Regions of Interest detection	6.5	14.5	6.5	16.6	2.4	5.5
Whole algorithm	70.9	169.2	14.1	32.7	12.5	28.5

a compare-swap unit which compares the two j-th bit-values of $s_{a,j}$ and $s_{b,j}$ and swaps the whole k-bit inputs [35]. The compare-swap unit is shown in Figure 7(a), where the result of the comparator is used to control both multiplexers for adjusting the input operands in order to perform the sorting.

The main group and each subgroup, are sorted with respect to the leftmost bit by a sorting network which repeatedly scans top-down and bottom-up to find '1' and '0', respectively, until scan indices cross. This sorting network utilizes the compare-swap elements and exploits a high degree of parallelism. Pipeline registers are added after each compare-swap element as shown in Figure 7(b), where only two swaps were needed to perform the illustrated sorting.

6. EXPERIMENTAL EVALUATION

For our timing evaluation, we made use of two processing systems: the Jetson TX1 Embedded System Developer Kit [36] running 64-bit version of Ubuntu 16.04 and the Cyclone II DSP Development Board. Both of those devices provide high computational capabilities, while still retaining low power consumption and weight. Table I contains a brief overview of their specifications. Those characteristics makes them perfect candidates for on-line and on-board applications of UAVs. Both CPU-only and CPU+GPU implementations are based on C++ and executed in the Jetson TX1 board, while the FPGA version is based on VHDL and MegaCore functions tested on the Cyclone II DSP Development Board. The two instances of our running application, applied on the testing videos, are shown in Fig. 8(a) and Fig. 8(b), respectively.

DOI: 10.1002/cpe

The results of our timing experiments for the main processing steps of the presented method are summarized in Table II. Note that two different input resolutions were tested, e.g. 800×600 and 1280×1024 in order to evaluate our implementations' scalability. The table contains the average execution time observed after testing 10000 input frames. As it can be seen, each one of the parallel versions (CPU+GPU and FPGA) achieve real-time performances justifying the employed algorithms' efficiency. It is important also to note that the processing time required by the GPU to compute steps 1 through 3 in not perceptible by the overall application of the CPU+GPU version since the respective calculations are pipelined with the CPU's steps 4 and 5. The FPGA implementation outperforms almost every step, apart from the Moments Based Contrast Enhancement, where the CPU+GPU implementation presents the best individual performance. The effectiveness of the FPGA implementation is mainly due to the full exploitation of the algorithm's dataflow through parallelism, pipelining and the used bit-wise operations. The CPU-GPU implementation presents comparable results, however, the used control flow instructions presented an overhead in the performance since the different execution paths of the algorithm needed to be serialized, increasing the total number of executed instructions.

Furthermore, we used the OSU thermal image dataset [37] in order to evaluate the effectiveness of the detection algorithm. The dataset includes 284 images organized in 10 sequences that correspond to different weather conditions. We applied the proposed algorithm to the entire dataset by keeping the same values for its parameters (threshold value, size filter, contrast parameter) for all the sequences in order to present its robustness to different conditions. It should be noted that we could further improve the results by adapting the aforementioned parameters specifically for each sequence. The results show a rate of 92.94% of true positive, 4.31% false positive and 7.06% false negative detection.

7. CONCLUSIONS

Human or fire-sized object detection from high altitudes, challenging as it is, constitutes a demanding procedure in terms of computational power. In this work, an efficient pipeline for human and fire detection was implemented, capable of running in real-time and on-board of a middle altitude UAV. The whole algorithm was designed and tested having in mind the parallelization characteristics of a modern GPU and FPGA system, taking advantage of their entire processing power. Comparative results justify the superiority of the proposed parallel approaches as compared to the performance of a serialized version, while proving the applicability on real life missions.

ACKNOWLEDGEMENT

The work presented in this paper is part of the 11SYNERGASIA_9_629 "Hellenic Civil Unmanned Air Vehicle - HCUAV" research project, implemented within the framework of the National Strategic Reference Framework (NSRF) and through the Operation Programme "Competitiveness & Entrepreneurship – SYNERGASIA 2011". The research project is co-financed by National and Community Funds, 25% from the Greek Ministry of Education and Religious Affairs - General Secretariat of Research and Technology and 75% from E.U. - European Social Fund.
The authors gratefully acknowledge the support of NVIDIA Corporation with the donation of the Jetson TX1 used for this research.

REFERENCES

1. Maier J, Humenberger M. Movement detection based on dense optical flow for unmanned aerial vehicles. *International Journal of Advanced Robotic Systems* 2013; **10**.
2. Schmidt F, Hinz S. A scheme for the detection and tracking of people tuned for aerial image sequences. *Photogrammetric Image Analysis*. Springer, 2011; 257–270.
3. Oreifej O, Mehran R, Shah M. Human identity recognition in aerial images. *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, IEEE, 2010; 709–716.
4. Gaszczak A, Breckon TP, Han J. Real-time people and vehicle detection from uav imagery. *IS&T/SPIE Electronic Imaging*, International Society for Optics and Photonics, 2011; 78 780B–78 780B.

DOI: 10.1002/cpe

5. Reilly V, Solmaz B, Shah M. Shadow casting out of plane (scoop) candidates for human and vehicle detection in aerial imagery. *International journal of computer vision* 2013; **101**(2):350–366.
6. Wang J, Zhang Y, Lu J, Li Y. Target detection and pedestrian recognition in infrared images. *Journal of Computers* 2013; **8**(4):1050–1057.
7. Flynn H, Cameron S. Multi-modal people detection from aerial video. *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*, Springer, 2013; 815–824.
8. Huang PH, Su JY, Lu ZM, Pan JS. A fire-alarming method based on video processing. *Intelligent Information Hiding and Multimedia Signal Processing, 2006. IIH-MSP'06. International Conference on*, IEEE, 2006; 359–364.
9. Martinez-de Dios J, Arrue BC, Ollero A, Merino L, Gómez-Rodríguez F. Computer vision techniques for forest fire perception. *Image and vision computing* 2008; **26**(4):550–562.
10. Celik T. Fast and efficient method for fire detection using image processing. *ETRI journal* 2010; **32**(6):881–890.
11. Vipin V. Image processing based forest fire detection. *International Journal of Emerging Technology and Advanced Engineering* 2012; **2**(2):2250–2459.
12. Den Breejen E, Breuers M, Cremer F, Kemp R, Roos M, Schutte K, De Vries JS. Autonomous forest fire detection. *Proc. 3rd Int. Conf. on Forest Fire Research*, Citeseer, 1998; 2003–2012.
13. Töreyin BU, Cinbiş RG, Dedeoğlu Y, Çetin AE. Fire detection in infrared video using wavelet analysis. *Optical Engineering* 2007; **46**(6):067 204–067 204.
14. Verstockt S, Dekeerschieter R, Vanoosthuise A, Merci B, Sette B, Lambert P, Van de Walle R. Video fire detection using non-visible light [short version]. *6th International seminar on Fire and Explosion Hazards (FEH-6)*, University of Leeds. School of Civil Engineering, 2010; 49–50.
15. Giitsidis T, Karakasis EG, Gasteratos A, Sirakoulis GC. Human and fire detection from high altitude uav images. *2015 23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2015; 309–315.
16. Amanatiadis A, Karakasis E, Bampis L, Giitsidis T, Panagiotou P, Sirakoulis GC, Gasteratos A, Tsalides P, Goulas A, Yakinthos K. The HCUAV project: electronics and software development for medium altitude remote sensing. *IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2014; 1–5.
17. Bampis L, Karakasis EG, Amanatiadis A, Gasteratos A. Can speedup assist accuracy? an on-board gpu-accelerated image georeference method for uavs. *In Proc. Springer International Conference on Computer Vision Systems*, 2015; 104–114.
18. Karakasis EG, Bampis L, Amanatiadis A, Gasteratos A, Tsalides P. Digital elevation model fusion using spectral methods. *IEEE International Conference on Imaging Systems and Techniques (IST)*, 2014; 340–345.
19. Papakostas GA, Koulouriotis DE, Karakasis EG. A unified methodology for the efficient computation of discrete orthogonal image moments. *Information Sciences* 2009; **179**(20):3619–3633.
20. Arodź T, Kurdziel M, Popiela TJ, Sevre EO, Yuen DA. Detection of clustered microcalcifications in small field digital mammography. *computer methods and programs in biomedicine* 2006; **81**(1):56–65.
21. Johnson J. Analysis of image forming systems. *Image Intensifier Symposium, AD 220160*, US Army Engineer Research and Development Laboratories, Fort Belvoir, Va. 1958; 244–273.
22. Boost, peer-reviewed portable C++ libraries. URL `www.boost.org`.
23. Eigen, C++ template library for linear algebra. URL `eigen.tuxfamily.org`.
24. OpenCV, Open source Computer Vision library. URL `www.opencv.org`.
25. Hwu WM, Rodrigues C, Ryoo S, Stratton J. Compute unified device architecture application suitability. *Computing in Science & Engineering* 2009; **11**(3):16–26.
26. Kharlamov A, Podlozhnyuk V. Image denoising. *NVIDIA* 2007; .
27. Podlozhnyuk V. Image convolution with cuda. *NVIDIA Corporation white paper, June* 2007; **2097**(3).
28. Ruetsch G, Micikevicius P. Optimizing matrix transpose in CUDA. *Nvidia CUDA SDK Application Note* 2009; **28**.
29. CUB, C++ library for every layer of the CUDA programming model. URL `nvlabs.github.io/cub`.
30. Amanatiadis A, Bampis L, Gasteratos A. Accelerating image super-resolution regression by a hybrid implementation in mobile devices. *IEEE International Conference on Consumer Electronics*, 2014; 335–336.
31. Amanatiadis A, Bampis L, Gasteratos A. Accelerating single-image super-resolution polynomial regression in mobile devices. *IEEE Transactions on Consumer Electronics* 2015; **61**(1):63–71.
32. Bampis L, Iakovidou C, Chatzichristofis SA, Boutalis YS, Amanatiadis A. Real-time indexing for large image databases: color and edge directivity descriptor on GPU. *Springer The Journal of Supercomputing* 2015; **71**(3):909–937.
33. Georgoudas IG, Kyriakos P, Sirakoulis GC, Andreadis IT. An fpga implemented cellular automaton crowd evacuation model inspired by the electrostatic-induced potential fields. *Microprocess. Microsyst.* Nov 2010; **34**(7-8):285–300.
34. Andreadis I, Amanatiadis A. Digital image scaling. *IEEE Instrumentationand Measurement Technology Conference Proceedings*, vol. 3, 2005; 2028–2032.
35. Koch D, Torresen J. Fpgasort: a high performance sorting architecture exploiting run-time reconfiguration on fpgas for large problem sorting. *ACM/SIGDA international symposium on Field programmable gate arrays*, 2011; 45–54.
36. NVIDIA Jetson TX1 Embedded System Developer Kit. URL `http://www.nvidia.com/object/jetson-tx1-dev-kit.html`.
37. Davis J, Keck M. A two-stage template approach to person detection in thermal imagery. *Workshop on Applications of Computer Vision*, 2005.

DOI: 10.1002/cpe