Journal of Electronic Imaging

JElectronicImaging.org

Exploration of preprocessing architectures for field-programmable gate array-based thermal-visual smart camera

Muhammad Imran
Bernhard Rinner
Sajjad Zandi Zand
Mattias O'Nils

Muhammad Imran, Bernhard Rinner, Sajjad Zandi Zand, Mattias O'Nils, "Exploration of preprocessing architectures for field-programmable gate array-based thermal-visual smart camera," *J. Electron. Imaging* **25**(4), 041006 (2016), doi: 10.1117/1.JEI.25.4.041006.

Exploration of preprocessing architectures for field-programmable gate array-based thermal-visual smart camera

Muhammad Imran,[a,*] Bernhard Rinner,[b] Sajjad Zandi Zand,[a] and Mattias O'Nils[a]
[a]Mid Sweden University, Department of Electronics Design, Holmgatan 10, Sundsvall 851 70, Sweden
[b]Alpen-Adria-Universität, Institute of Networked and Embedded Systems, Lakeside Park B02b, Klagenfurt 9020, Austria

Abstract. Embedded smart cameras are gaining in popularity for a number of real-time outdoor surveillance applications. However, there are still challenges, i.e., computational latency, variation in illumination, and occlusion. To solve these challenges, multimodal systems, integrating multiple imagers can be utilized. However, trade-off is more stringent requirements on processing and communication for embedded platforms. To meet these challenges, we investigated two low-complexity and high-performance preprocessing architectures for a multiple imagers' node on a field-programmable gate array (FPGA). In the proposed architectures, majority of the tasks are performed on the thermal images because of the lower spatial resolution. Analysis with different sets of images show that the system with proposed architectures offers better detection performance and can reduce output data from 1.7 to 99 times as compared with full-size images. The proposed architectures can achieve a frame rate of 53 fps, logics utilization from 2.1% to 4.1%, memory consumption 987 to 148 KB and power consumption in the range of 141 to 163 mW on Artix-7 FPGA. This concludes that the proposed architectures offer reduced design complexity and lower processing and communication requirements while retaining the configurability of the system. *© 2016 SPIE and IS&T* [DOI: 10.1117/1.JEI.25.4.041006]

Keywords: smart camera; preprocessing; architecture; thermal; field-programmable gate array.

Paper 15943SS received Dec. 31, 2015; accepted for publication Mar. 25, 2016; published online May 2, 2016.

1 Introduction

The advancement in related technological fields has enabled the imaging sensors for ubiquitous applications, ranging from indoor navigation and machine vision to outdoor surveillance.[1,2] These developments have resulted in the emergence of concepts such as wireless sensor networks (WSN), which consist of spatially distributed smart camera nodes. The smart camera nodes are standalone devices able to perform analytics by processing data locally on the node and then transmitting useful information to the end user. Because of the scalability and easy integration into existing infrastructure, WSNs are expected to be deployed for applications which are characterized by low latency, limited power availability, and low-rate wireless bandwidth.[3]

In relation to WSNs, vision sensors are commonly used in smart camera motes[1,4] because of their complementary metal-oxide-semiconductor (CMOS) process technology, versatility, low cost, easy integration, and the availability of ready-to-use imaging libraries. However, vision sensors are sensitive to the visible range of the electromagnetic spectrum, therefore, in poor ambient lighting conditions, in the absence of external lighting sources, the objects do not reflect enough light for the vision sensors. The challenges become more stringent for vision sensors under certain environmental conditions, e.g., rain and horizontal obscuration. Even when weather conditions are good and there is sufficient lighting, the vision sensors are prone to produce false positive results because of illumination variation, direct light, shadows, reflections, changing background, cluttered background, and occlusion.[4,5]

To meet these challenges, there is an interest in augmenting vision sensors with beyond-vision-range sensors such as lasers and thermal sensors. Figure 1 shows visual and thermal images of the same scene from vision and thermal sensors. In the visual image, the infrastructure, i.e., the road, pole, and construction frame, is detailed enough, whereas the persons in the picture appear occluded by the frame and cluttered with background. In the thermal image, however, the persons are detailed enough, but the infrastructure information is not sufficient, which is a result of thermal images being obtained by sensing the radiation emitted from the objects. The emitted radiation is a
ffected by the emissivity of the objects, temperature changes,[5] and water droplets.[6] Therefore, it is important to combine the two sensors, as multisensor imaging data provide complementary information of the same activity region. This combination will help in robust detection of objects in surveillance applications.[5–7]

For real-time surveillance applications, the processing needs to be performed on a processor that can handle two streams of images with low latency. Microprocessor-based platforms are expected to have high latency because of memory-bound and sequential operations.[8–10] Therefore, using field-programmable gate array (FPGA) for processing is a suitable choice for the handling of parallel processes because of the inherited parallelism.

*Address all correspondence to: Muhammad Imran, E-mail: muhammad.imran@miun.se

1017-9909/2016/$25.00 © 2016 SPIE and IS&T

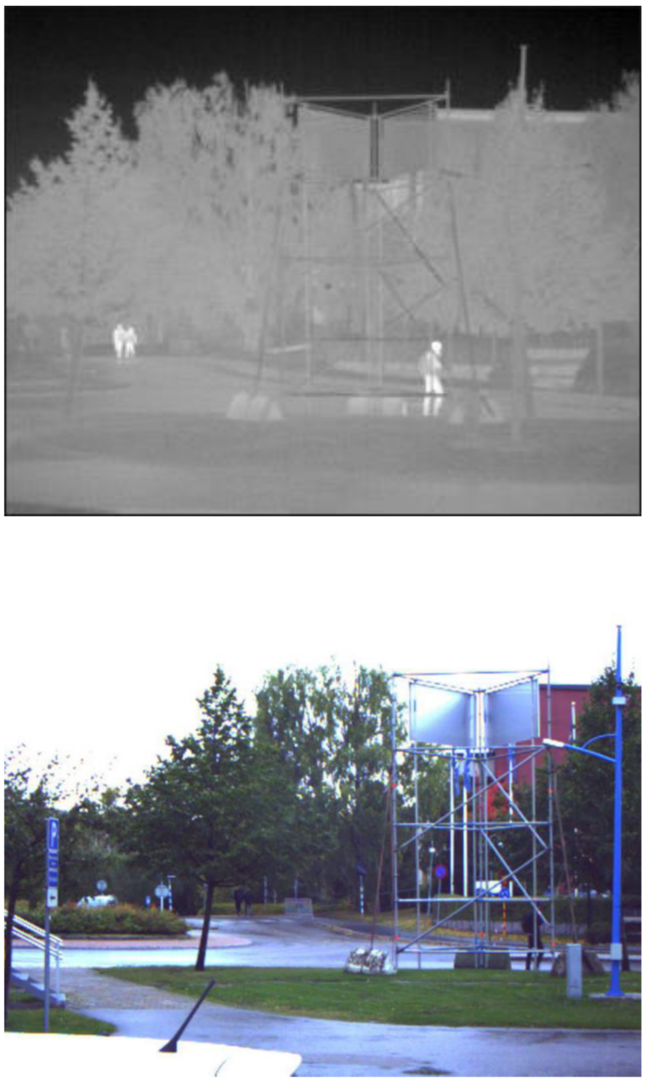


Fig. 1 Visual and thermal images.

Integrating two different image sensors poses challenges of design complexity, synchronization, a large amount of data, and high-power consumption. Traditional approaches to handling such challenges are based on the two implementation strategies listed here.

- Acquire images and transmit compressed images in order to perform computation and analytics on a client device with sufficient resources [Fig. 2(a)].[11]
- Acquire images, process them and transmit feature information to a client device where analysis is performed [Fig. 2(b)].[12,13]

In the first approach, the computational load on the node is minimal but the communication requirement is enormous, which increases communication latency. In the second approach, the communication requirement is reduced as final features are transmitted. However, the computational load is high, which results in increased processing latency for microprocessor-based systems and increased implementation complexity for FPGA-based systems.[8]

In comparison to the aforementioned two approaches, our approach, shown in Fig. 3, considers computation load

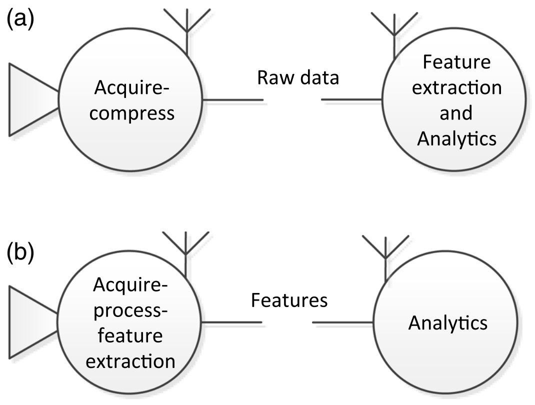

(a)
Acquire-compress
Raw data
Feature extraction and Analytics
(b)
Acquire-process-feature extraction
Features
Analytics

Fig. 2 Traditional implementation strategies for smart cameras. (a) Most processing on the client device. (b) Most processing on the smart camera node.

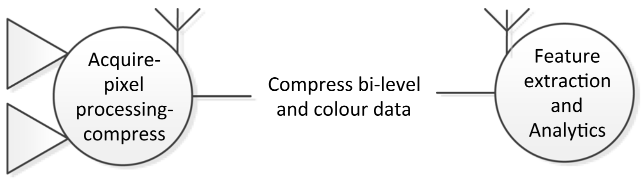

Acquire-pixel processing-compress
Compress bi-level and colour data
Feature extraction and Analytics

Fig. 3 Vision processing based on intelligence partitioning.

partitioning between two points, a camera node, and a client device. The preprocessing pixel level tasks, such as background modeling, subtraction, segmentation, morphological operation, and region of interest extraction are performed on the camera node, and high-level tasks such as feature extraction and classification on the client device.

1.1 *Problem Description*

Current research focusing on embedded smart cameras is mostly related to single imager where common challenges include computational complexity, high processing, and communication energy consumption in order to extend the lifetime.[8,10] Employing embedded smart cameras for real-time applications puts further constraints on latency and robust surveillance.

This in turn motivates the use of quick processing engines such as FPGA, which can perform computation in far less time as compared to microprocessor-based processing platforms.[8] However, implementation complexity on the register transfer level (RTL) for FPGA is greater. The robust surveillance requires multimodel sensor integration, which increases the challenges associated with processing, communication, latency, and power. In order to address these challenges, this paper explores two preprocessing architectures for FPGA-based smart cameras that have reduced implementation complexity, as well as processing and communication latencies.

In our proposed approach, a region of interest (ROI) is extracted using simple imaging tasks. The ROI is then compressed and transmitted more frequently, whereas the full-size image is transmitted less frequently in order to intelligently monitor the area. This approach helps to simplify design complexity, reduce processing and communication requirements for multiple imagers' systems on an FPGA-based platform. Depending on the classification algorithms on the client side, an architecture with color regions of interest or binary region of interest data can be transmitted from the camera node. The performance, power consumption, and output data shows that the solution is comparable to a single imager smart camera node, with the added advantage of robustness in detection under different environmental conditions.

The remainder of this paper is organized as follows. Section 2 presents related work, Sec. 3 discusses the experimental setup and test cases. Section 4 presents the processing flow and Sec. 5 discusses the architecture exploration and results. Finally, Sec. 6 summarizes the conclusions.

2 Related Work

The increasing interest in embedded smart camera research has broadened the scope by attracting different types of applications, mainly because of features such as on-board computations, wireless communication, and easy integration of peripherals. The increasing demand for on-board computations has motivated the usage of FPGA-based architectures for certain applications which require high performance, low energy, and reconfigurable solutions.[8,10,14] In this paragraph, we present a sample of work from published literature in the area of embedded smart camera and multispectral image processing.

Kandhalu et al.[15] presented a platform called DSPcam, which performs local processing in order to detect an event and annotate the video stream for the operator at the observation station in the network. Gasparini et al.[14] used a bilevel CMOS vision sensor of 128×64 resolution to capture images and then perform binary processing on a FLASH-based FPGA. The authors proposed design principles for VSN in the context of a long lifetime. Bourrasset et al.[10] proposed an FPGA-based smart camera mote which can be reconfigured by using internet protocol. The authors have reported a tool chain which can accelerate the development process on FPGA by avoiding the low-level programming. Rinner and Winkler[13] argued that privacy aware approaches require processing of tasks near the image sensor in order to transmit processed information. The authors demonstrated a proof-of-concept system showing cartoon images by embedding imaging tasks close the image sensor, thereby reducing the identity information.

The aforementioned embedded smart camera considered a single imaging sensor, which has a different set of challenges as compared with multispectral imagers, which are discussed in the following paragraphs.

Magno et al.[3] introduced a multimodal wireless smart camera equipped with a pyroelectric infrared sensor and a solar energy harvester. The authors have shown that the energy harvester together with a heterogeneous combination of sensors provide better battery life for the camera node. Goel et al.[4] proposed a multispectral camera using time-multiplexed illumination, which highlights salient features of a scene by using multiple wavelengths. The proposed camera system uses a CMOS sensor with a sensitivity from 350 to 1080 nm integrating narrow-band light-emitting diodes (LEDs) to create 17 different spectral bands, ranging from 450 to 990 nm. However, this camera system requires objects to be in close proximity to the camera in order to have good exposure for LED illumination. Mouts et al.[16] evaluated moving object detection techniques by using a pixel-based statistical approach. The authors evaluated the effect on the detection accuracy when thermal and visual image fusion is performed before and after subtraction. Ibrahim and Wirth[7] developed a visible and IR data fusion technique using the contoured transform. The proposed solution is based on a region-based fusion technique reported to produce output images for better human and machine interpretation. The authors' focus was to preserve the input image's spectral components, such as chromaticity of visual images, as compared to pixel-based fusion approaches.

The literature mentioned above shows that current approaches for combining thermal and visual information are based on fusion techniques, which can be categorized into three types[5] based on the processing location in the imaging flow, which consists of functions ranging from image acquisition to decision level. The three fusion approaches include:

- Pixel-based fusion
- Feature-based fusion
- Decision-level fusion

However, these techniques focus on efficient fusion of information from two image streams without considering the resource limitation of embedded smart cameras. Migrating traditional algorithms to an embedded platform results in a new set of challenges because of resource constraints, development time, and availability of support and skills.

In comparison to the aforementioned three approaches focusing on processing algorithms, our approach considers implementation on an FPGA-based platform. In this approach operations are performed on thermal images because of the lower spatial resolution and the higher contrast as compared to visual images. This enables us to transmit compressed ROI data more frequently and full-size images occasionally in order to monitor the area intelligently. In this way, design complexity, processing and communication requirements for embedded smart cameras, integrating multiple image sensors are reduced.

3 Experimental System and Test Case

Before describing our proposed approach, the experimental setup involving two image sensors, their respective parameters, such as fps, focal length, and spatial resolutions, are described. The section also gives a brief overview of the selected FPGA device, calibration method and test case used for this work.

3.1 *Hardware Description and Method for Calculating Latency*

For thermal imaging, a tamarisk 320 camera,[17] based on uncooled VOx micro-bolometer sensor technology, with a spectral band of 8 to 14 μm has been used. This camera has a 19 mm focal length lens, a spatial resolution of 320×240 and a pixel pitch of 17 μm. The clock frequency with parallel video mode is 10 MHz.[17] For visual imaging, an IDS CMOS UI camera[18] has been used. This camera has a 12-mm focal length lens, a spatial resolution of 1280×1024 and a pixel pitch of 5.3 μm. For resource estimation, an Artix 7 series FPGA device 7a200tfbg484[19] has been used. The device has 134600 logics and 730 (18 Kb) or 365 (36 Kb) BRAMs.

It is important to mention that the processing latency for the two architectures is calculated by using

$$T = (\text{Row} \times (\text{Col} + \text{Ls}) + \text{Lt})/f \quad (\text{sec}), \tag{1}$$

where Lt is the latency of each task, f is frequency of the thermal camera, Row represents the rows, and Col represents the columns of thermal camera, and Ls represents the low line sync.

3.2 *Calibration for Image Rectification*

The proposed system uses two imagers with different characteristics, such as different kinds of sensors, resolutions, and optics. These characteristics, in addition to the baseline, result in disparity between the two images. Therefore, we have used a one-time calibration method which will transform the images to emulate a common image plane so that objects appear of the same size and on the same plane in images. This calibration method involves small incandescent bulbs used as reference points, simple scaling tasks, and zero padding. A scaling factor for both horizontal and vertical direction is required in order to make the object sizes of the thermal image similar to the object sizes of visual image. For scaling in the horizontal direction, Eq. (2) is used, and for scaling in the vertical direction, Eq. (3) is used. *Rv1*, *Rv2*, and *Rv3* are reference points on visual images as shown in Fig. 4(a), and *Rt1*, *Rt2*, and *Rt3* are three reference points on the thermal images shown in Fig. 4(b). Each reference point's horizontal coordinate value is represented by x and vertical coordinate value by y. These reference points will facilitate the calculation of the upscaling factor for the thermal camera.

$$\text{Horizontal scaling factor (Hs)} = \frac{Rv2_{(x2)} - Rv1_{(x1)}}{Rt2_{(x2)} - Rt1_{(x1)}}. \tag{2}$$

$$\text{Vertical scaling factor (Vs)} = \frac{Rv3_{(y3)} - Rv1_{(y1)}}{Rt3_{(y3)} - Rt1_{(y1)}}. \tag{3}$$

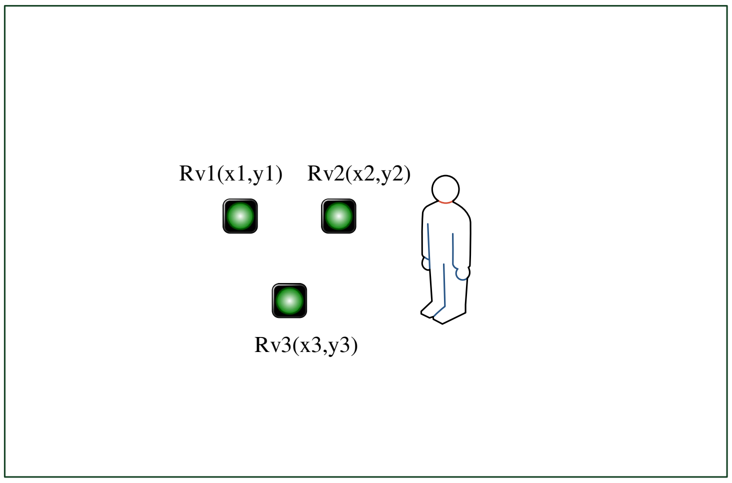
The new rows and columns are calculated according to

$$\text{Rows\_new} = \text{Hs} \times \text{Rows\_original}. \tag{4}$$

$$\text{Columns\_new} = \text{Vs} \times \text{Columns\_original}. \tag{5}$$

After scaling up the thermal image, it is zero padded in order to match the size of the visual image. In connection to this, the number of blank rows and columns are calculated by using Eqs. (5) and (6), respectively. The upscaled image, including reference points as well as horizontal and vertical blanking, is shown in Fig. 4(c).

$$\text{Blank columns} = Rv2_{(x2)} - Rt2s_{(x2)}. \tag{6}$$



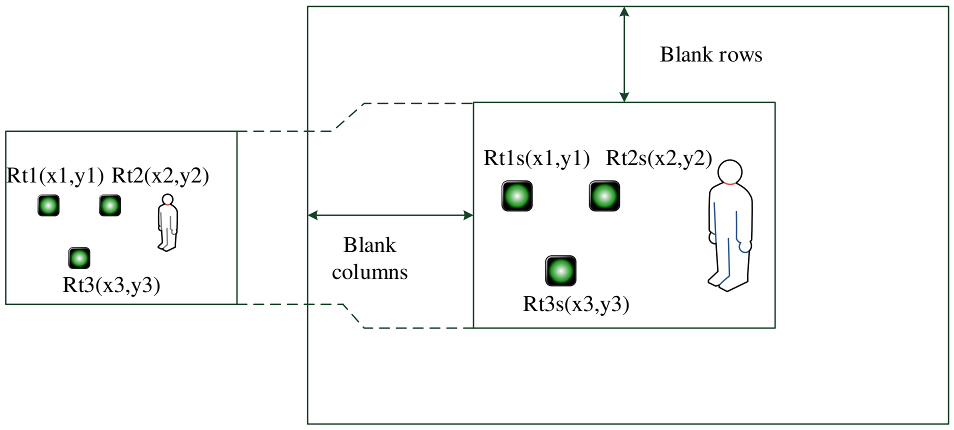


Fig. 4 Calibration of thermal and visual images. (a) Reference points on the visual image. (b) Reference points on the thermal image. (c) Scaled version of the thermal image.

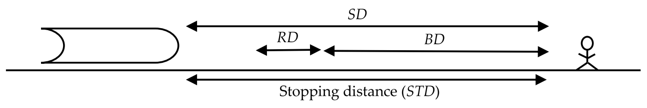


Fig. 5 Illustration of the train stopping with respect to object.

$$\text{Blank rows} = Rv2_{(y2)} - Rt2s_{(y2)}. \tag{7}$$

3.3 *Test Case: Rail Track Monitoring*

The test case for this work is surveillance of railway tracks' surroundings in areas where increased safety for people passing by and who work or live in the vicinity is needed. In Sweden alone, ∼100 people a year lose their lives on rail tracks.[20] Because of the high speed of trains today, pedestrians and drivers have little time to react to avoid collision. To avoid collision, it is vital that the train operator receives information before approaching safety-critical areas so that she/he can make the best possible decision. The stopping distance of the train depends on different variables such as number of braking elements, reaction time, response time, mass distribution, geography of the track, and dynamic friction.[21] These variables led us to the simplified assumption shown in Fig. 5, where stopping distance depends on the sighting distance, i.e., when the train operator sees an object, the reaction distance, and braking distance. Our goal is to propose an imaging-based solution that detects objects in different weather conditions and alerts the train operator in real-time so that the train operator has robust information of the surroundings. This in turn will help to increase the sighting distance and reduce the reaction time. In the proposed approach, the smart cameras at fixed locations will transmit processed data to the train operator in real-time. This helps us to simplify the system parameters, such as the calibration of the two image sensors and the limiting distance at which objects can be correctly classified.

4 Preprocessing Flow for IR-Visual Smart Cameras

Based on the approach shown in Fig. 3, two preprocessing flows, shown in Figs. 6 and 7, are analyzed for hardware implementation. The preprocessing flow for architecture 1, shown in Fig. 6, has two types of output data transmission: compressed color ROI of activity region and complete color compressed image. The preprocessing flow for architecture 2, shown in Fig. 7, has two types of output data: compressed binary ROI of activity region and complete color compressed image. The selection of the processing flows depends on the classification algorithms used on the client device. One architecture can be employed for systems which classify objects based on color data and other can be employed for systems which classify objects based on binary data. In the following, a detailed discussion of each task is presented.

4.1 *Background Modeling and Subtraction*

In this task, two existing low-complexity background modeling and subtraction techniques, background generation with a temporal low-pass for infinite impulse response (IIR) filtering[12] and progressive background generation,[22] have been evaluated with respect to thermal imagery. Based on the analysis, the background modeling techniques have been modified to develop a hybrid technique which was tested on different data sets. After in-depth analysis, the proposed technique is modeled at RTL level and implemented on FPGA.[9] The hybrid background is represented by

$$\mathrm{Bg}(t) = \begin{cases} \mathrm{Mbg} & \text{if } |\mathrm{In} - \mathrm{Bg}(t-1)| < \varepsilon \\ \mathrm{Lbg} & \text{else} \end{cases}, \tag{8}$$

$$\mathrm{Mbg} = \min(\mathrm{In}(t), \mathrm{Bg}(t-1)) \quad \text{or} \quad \mathrm{avg}(\mathrm{In}(t), \mathrm{Bg}(t-1))$$
$$\mathrm{Lpg} = \alpha\,\mathrm{In} + (1-\alpha)\mathrm{Bg}(t-1)$$

where In is the current image, Bg is the generated background, Mbg represents background pixels when the difference between the current and the previous background is smaller than a constant ε. In this study, the value of ε is selected as 25 based on repetitive experiments on the thermal image data set.[23] The static objects with a temperature similar to that of humans are considered as a part of the background because of the small intensity difference between successive images. Therefore, to generate Mbg, we use a nonlinear min filter. Other alternatives for calculating the Mbg value is averaging of the two pixel values, i.e., the value of the current or previous frame pixel value. Lbg is generated by using low-pass first-order recursive filter when the difference between the current and the previous background is greater than a constant ε. In this part, the background image is updated by integrating new incoming pixel data into the current background to adapt the temperature variation and motion changes caused by high frequency background objects like trees and shrubs etc. In this case, α is the adaptation coefficient and it is selected as 0.15 in order to ensure that artificial tails behind moving objects are not created.[12] The value of α is selected based on repetitive experiments on the thermal image data set.[23]

4.2 *Segmentation*

The background subtraction tasks efficiently remove the background and uneven variation in thermal images. Therefore, a global thresholding with a manually selected threshold value of 28 has been used because of the uniformity in the subtracted thermal image. The manual threshold is selected after thorough experiments. This helped to reduce the complexity in hardware implementation.

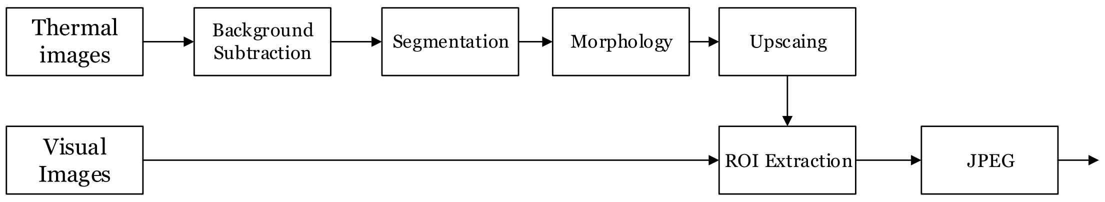


Fig. 6 Preprocessing flow for architecture 1.

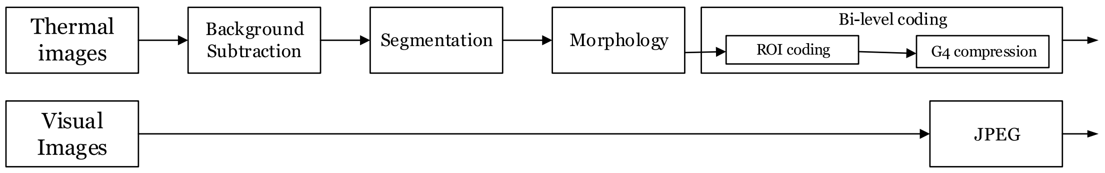


Fig. 7 Preprocessing flow for architecture 2.

4.3 *Morphology*

In morphology, unwanted pixel noise is removed by using morphological operation of erosion and dilation. A window size of 3×3 is selected in order to perform the operation.

4.4 *Bilevel Region of Interest Coding*

In bilevel ROI coding, image data is reduced by using compact ROI extraction techniques, ITU-T G4 coding scheme and zero-to-end symbol run length coding.[8]

4.4.1 *Compact region of interest extraction*

In ROI extraction, binary regions of objects are extracted by removing the rows without objects and retaining the columns as shown in Fig. 8. The columns are passed through because the data after ROI is compressed by using the ITU-T G4, in which the position of each changing picture element is calculated, rather than alternating black and white runs. The rows containing no object pixels are represented by zero and the row containing objects are represented by one. These sequences of 1s and 0s are encoded by using a version of a zero-to-end run length coding technique.[8] The run length codes representing the presence of objects in the rows are transmitted along with the compressed Huffman codes of the ROI image to the client device.

4.4.2 *ITU-G4 compression*

For bilevel image coding, the ITU-T G4 compression scheme has been selected based on the reported study.[24] The ITU-T G4 uses a two-dimensional line-by-line coding method in which vertical features in the source image are used to achieve a better compression ratio.

4.4.3 *Data formatting*

The output data format for bilevel ROI coding is shown in Table 1 where "Huffman codes" are appended with "row runs" of variable length followed by two bytes of information showing the data length of row runs. This data format will enable the reconstruction of the image with objects placed in the original location.

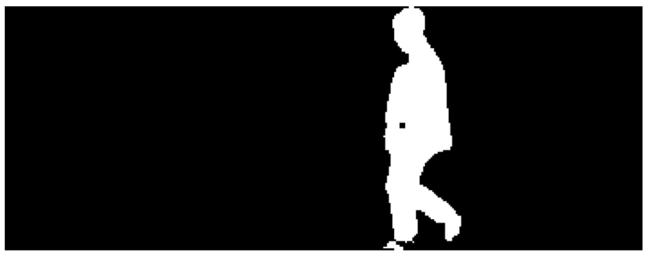


Fig. 8 Bilevel ROI extraction.

4.5 *Upscaling and Zero Padding*

There are several algorithms available for upscaling images such as the nearest neighbor, bilinear, bicubic, quadratic cubic, winscale, Lagrange, and Gaussian scaling algorithms. Due to the unique characteristics and wide applications of image scaling, a separate study of their evaluation methods is essential. Some of these techniques, such as the Gaussian method, offers good quality but have higher computational complexity.[25] In our approach, a bilevel segmented image requires upscaling. Therefore, a low-complexity nearest neighbor technique has been selected. This method has a good high frequency response and reduces the blurring effect.[18] Following upscaling, zero padding is applied in real-time on pixels retrieved from memory. This task requires storage of a binary thermal image in internal memory in order to synchronize the timings of thermal and visual images.

4.5.1 *Memory requirement*

The proposed architecture 1 using an upscaling algorithm, requires a memory of 75 KB in order to store a segmented thermal image with a bit-width of 1. The memory is essential in this case as the upscaling is done in both the vertical and horizontal axis by a factor of 2×2. Access to the pixel of the original image is vital to repeat the row once more; at each clock cycle one new pixel is read from the thermal camera. The segmented image data is written with the thermal camera pixel clock, whereas the stored data is read by a visual camera pixel clock. Using a visual clock for the reading process from the block RAM (BRAM) helps the synchronization of timing issues between the upscaled thermal pixels and the incoming visual pixels. In order to prevent overwriting the memory contents that have not been read yet, the writing counter stops when the memory is full and starts again when the reading counter reaches the last entry of the memory. In such cases, the data of this frame is not written, and the writing counter starts with the next frame synchronization signal.

4.6 *Color Region of Interest Coding*

Color ROI coding consists of three parts, (1) extraction of color ROI, (2) JPEG coding, and (3) rows and columns runs calculation

Table 1 Bilevel ROI data format.

Huffman codes	Rows runs	Row runs bytes count

4.6.1 *Compact region of interest extraction*

In this task, the thermal image is preprocessed in order to segment the foreground from the background as shown in Fig. 9(c). In addition, the thermal image is upscaled and zero padded in order to match the visual image. Following this, color ROI is extracted by performing a logical "AND" between the pixels of the processed thermal images and the pixels of the visual raw images. In this way, the unwanted details are removed as shown in Fig. 9(d). Next, the rows and columns containing no objects are discarded. The resultant compact ROI extracted image is shown in Fig. 9(e).

For pixel-based processing, this task is completed in two stages. In the first stage, rows with objects are retained, while rows with no objects are discarded, as shown in Fig. 10(b). In the second stage, columns are traversed for the presence of objects, and columns with objects are retained whereas columns with no objects are discarded, as discussed in Fig. 10(c).

The rows and columns with objects are represented by 1s, and row and columns with no objects are represented by 0s. In the first stage, the sequence of 1s and 0s for rows are coded and in the second stage, 1s and 0s for columns are coded by using zero-to-end symbol run length coding. This process is shown in Fig. 9(d).

4.6.2 *Memory requirement*

The color ROI extraction requires storage of the image part in the memory for the second stage. In the first stage, rows

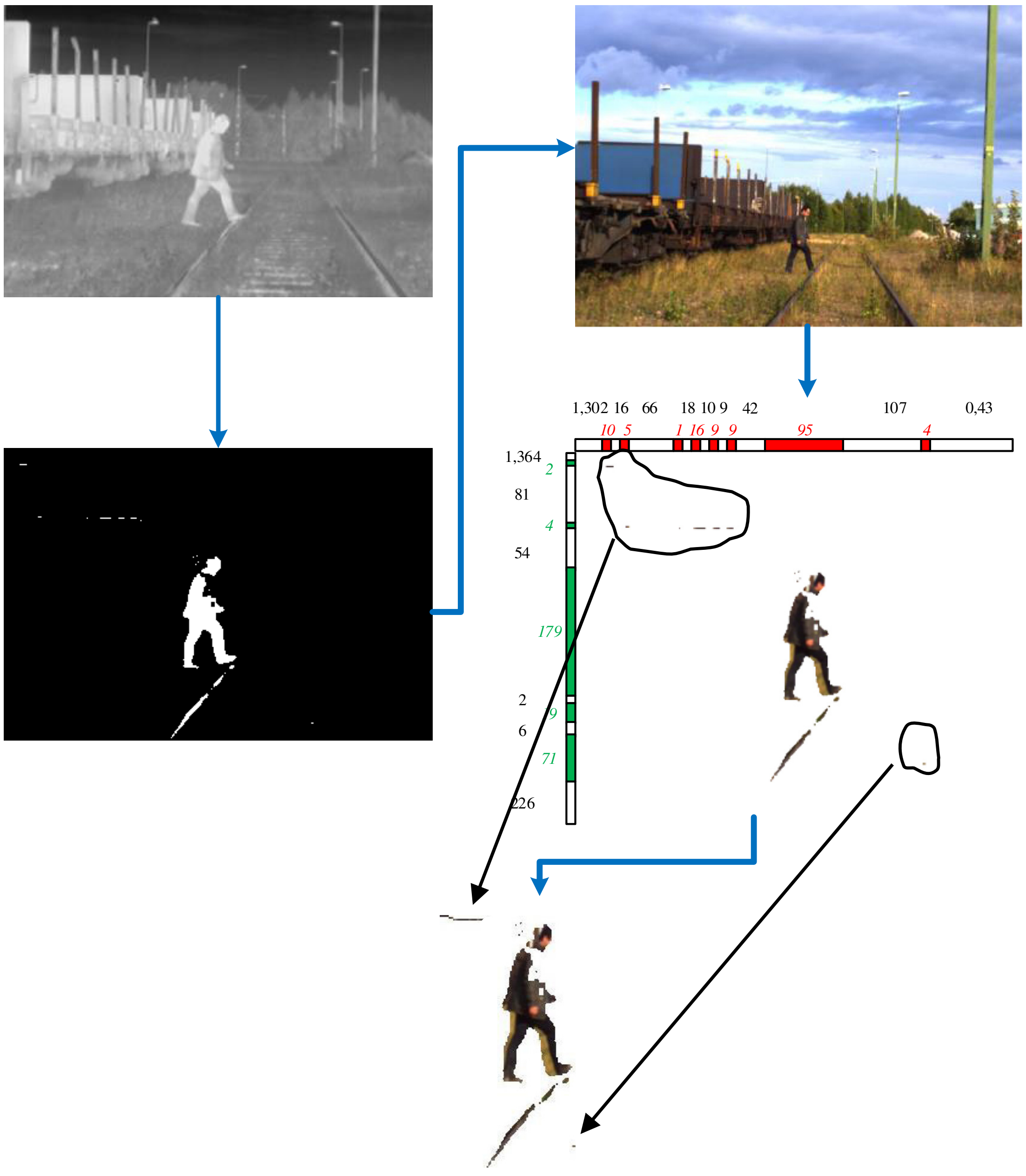


Fig. 9 Illustration of color ROI extraction. (a) Thermal image. (b) Visual image. (c) Preprocessed thermal image. (d) Masked image of c and b. (e) Compact ROI extraction.

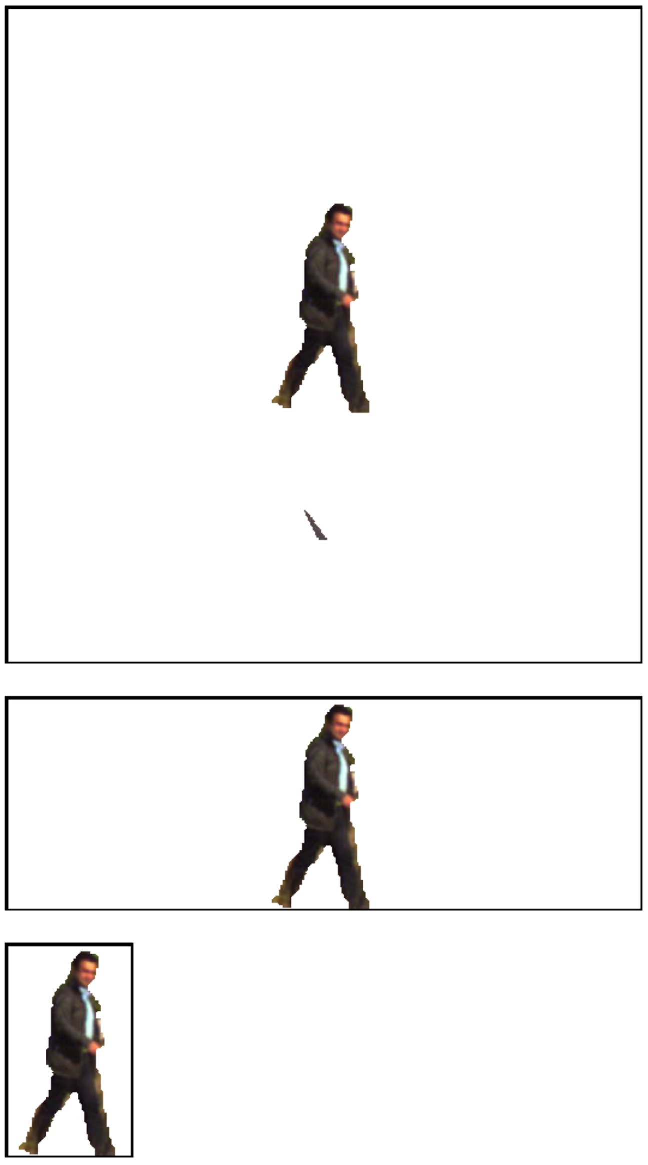


Fig. 10 Compact ROI extraction illustration. (a) Full-size image. (b) ROI with columns retained. (c) Compact ROI after removing rows and columns.

that do not contain objects are removed, and then extracted ROI is stored in the internal memory. The resultant image is shown in Fig. 10(b). In the second stage, the columns containing no objects are discarded. The resultant image is shown in Fig. 10(c). The memory requirement is estimated by using Eq. (9). Using the mentioned experimental setup configuration, columns are 1280, the average expected number of rows are 225, and bits per pixel are 24. Therefore, the memory requirement is 6.6 MB

$$\text{mem} = \text{columns} \times \text{expected\_rows} \times \text{bitsperpixels}. \tag{9}$$

4.6.3 *JPEG coding*

For color image coding, the JPEG compression scheme has been selected because it is commonly used, and it offers the option of selecting a suitable image quality and compression ratio. In connection, we evaluated the compression scheme in MATLAB with respect to compression ratio and

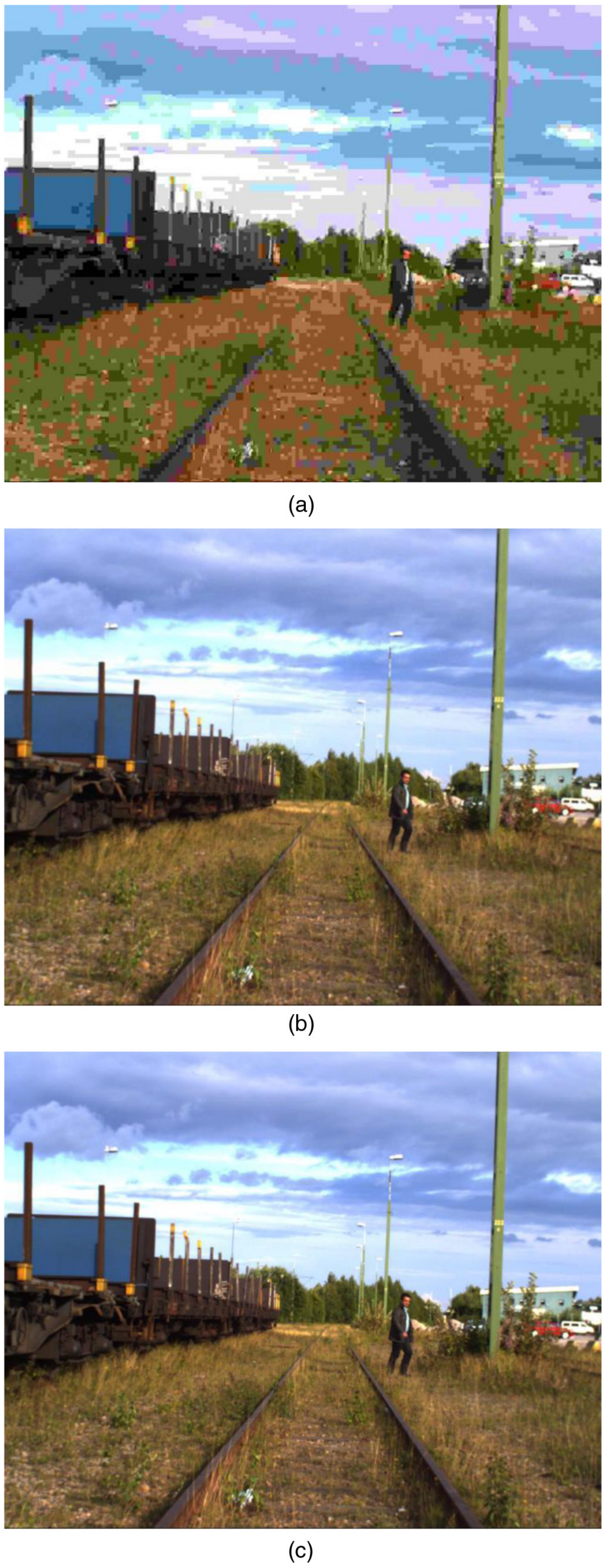


Fig. 11 Selection of JPEG compression quality factor. (a) Compression with quality factor of 1. (b) Compression with quality factor of 10. (c) Compression with quality factor of 100.

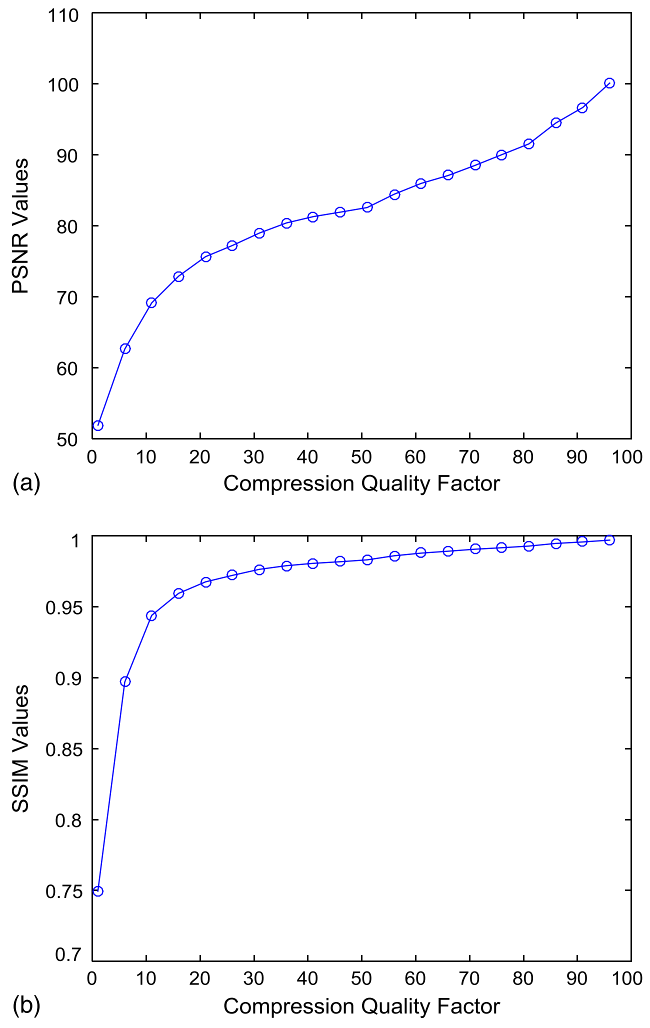


Fig. 12 (a) PSNR and (b) SSIM values with different compression quality factors.

image quality by selecting different (quantization matrix) quality factors, ranging from 0 to 100, where 0 means lower quality and higher compression, and 100 means higher quality and lower compression. The visual inspection (Fig. 11), peak signal-to-noise ratio (PSNR), and structure similarity index (SSIM) values (Fig. 12) show that a quality factor of 10 is suitable and does not result in major loss of image details.

4.6.4 *Data Formatting*

The output data format for color ROI coding is shown in Table 2. The JPEG coding is appended with "row runs" and "column runs" of variable length followed by 2 bytes for each row and column length information.

4.7 *Resource utilization and latency of tasks*

The resource utilization and latency of each task in the preprocessing flow is shown in Table 3. The resource estimation is considered with respect to Artix 7a200tfbg484 FPGA. This device can implement BRAM in two formats, either as 18 KB, which are 740 in number, or 36 KB, which are 365 in number. In Table 3, 18 KB BRAMs is represented by the * symbol, whereas 36 KB BRAMs is represented by the + symbol. In relation to the final implementation, logic resources of any specific strategy can exceed the combination of the individual functions because the integration and synchronization requires extra logics.

Table 2 Color ROI data format.

ROI JPEG data	Row runs	Column runs	Row runs bytes count	Column runs bytes count

5 Evaluated Architectures and Discussion

Two architectures have been evaluated for embedded smart cameras on FPGA, taking the implementation complexity on hardware, data reduction, and resource utilization into consideration. In both architectures, background modeling and subtraction is an integral and important preprocessing step; therefore, the background modeling and subtraction algorithm is discussed before describing the two architectures.

5.1 *Background Modeling and Subtraction*

The implementation complexity on hardware platforms such as FPGA requires development of a low-complexity background subtraction technique which can effectively

Table 3 Resource utilization and latency of individual tasks (Artix 7a200tfbg484).

Vision tasks	Logics used	% used	BRAMs 18 K*/36 K+	DSP	Latency (clk cycles)
Image capture	126	0.09	NA	NA	4
Background storage model and subtraction	149	0.11	24*	2/740	76802
Segmentation	6	0.00		NA	1
Morphology	134	0.10	4+	NA	650
Upscaling and zero padding	235	0.17	1+ 2*	NA	2560
Bilevel ROI coding	NA	NA	NA	NA	NA
Compact ROI extraction	276	0.21		NA	641
ITU-T G4	2946	2.19	3+	NA	327
Data formatting	5	0.00		NA	1
Color ROI coding	NA	NA	NA	NA	NA
Compact ROI extraction	414	0.31	188*	NA	1
JPEG coding [26]	1594	1.18	5+	11/275	2064
Data formatting	27	0.02		NA	1
Rs232 communication	167	0.12	1+	NA	4

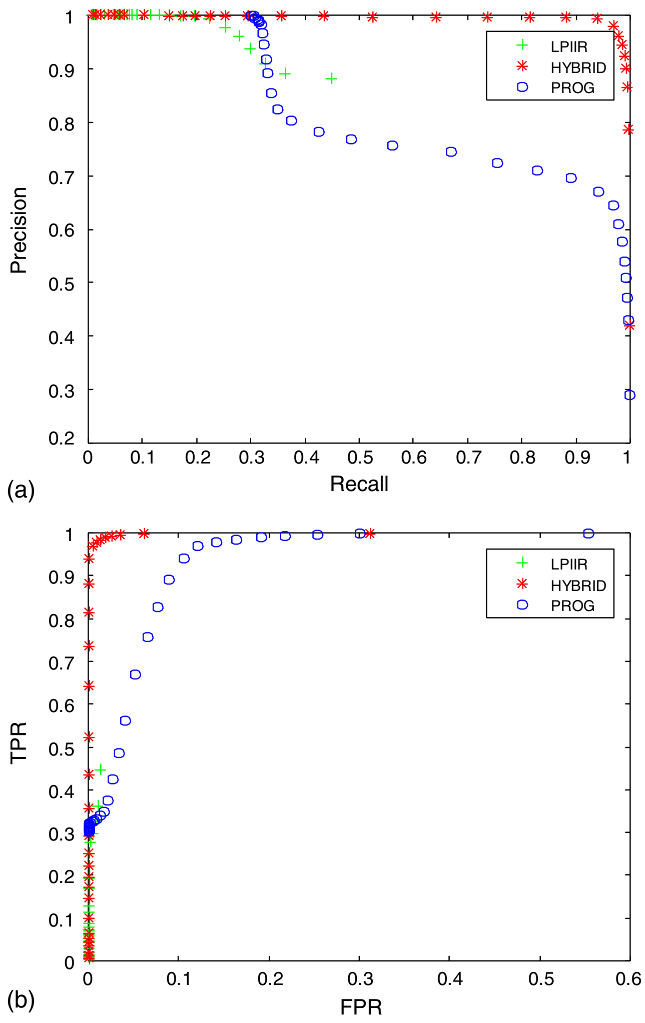


Fig. 13 ROC curve for LPIIR, hybrid, and progressive background subtraction algorithms. (a) Precision versus recall. (b) TPR versus FPR.

remove the background noise while maintaining a reduced computational and memory requirement in order to fit on available FPGA resources. We have performed an in-depth analysis on different sets of thermal images, characterizing signal-to-noise ratio challenges, e.g., motion of high frequency background objects, temperature variation and camera jitter etc.[9] The investigated result is complemented by performance analysis by using ROC curves.

5.1.1 *Performance analysis using receiver operating curve curves*

The performance of the three techniques including LP IIR,[12] hybrid,[9] and progressive generation[22] techniques are evaluated by using receiver operating curve (ROC) analysis. Each unique point on the ROC curve represents a threshold value for the image. The experiments were performed on a host computer for image data sets.[23] ROC curves for the three background subtraction techniques are shown in Fig. 13. It is evident that the hybrid technique has high precision and high recall as compared with the other two methods, as shown in Fig. 13(a). The ROC curves with true positive rate (TPR) versus false positive rate (FPR), shown in Fig. 13(b), also depict that TPR is high and FPR is low for our proposed algorithm as compared to the other two published systems.

5.1.2 *Background subtraction architecture*

The computational architecture of the proposed background modeling and subtraction is shown in Fig. 14. The first incoming input image is stored in the BRAMs without any processing. Following the second frame, the current frame and the frame previously stored in BRAMs is used to generate a background model (BG model) by using the operations discussed in Sec. 4.1. The multiplexer will allow the first frame in the initial setup configuration and then always store the new modeled backgrounds in the BRAMs. For the subtraction operation, the current frame is subtracted from the generated background image on pixel level while simultaneously updating the background model in the BRAMs. In the following, the two architectures are presented.

5.2 *Architecture 1: Medium Complexity and Relatively High Memory Requirement*

In this architecture, as shown in Fig. 15, preprocessing imaging tasks are performed on the thermal stream of images because of the lower spatial resolution and better contrast. This in turn helps to reduce the processing and communication requirement. After detecting the area of interest in the thermal stream of images, a region of interest in the visual images are extracted by using the masking (AND) operation with visual images. The visual ROI image is coded by transmitting color ROI coded data as discussed in Sec. 4.6. The activity information is transmitted frequently by using color ROI coding because of the reduced data for communication purposes. In addition to the ROI image, depending on the requirement (periodicity, time, or event triggering), a

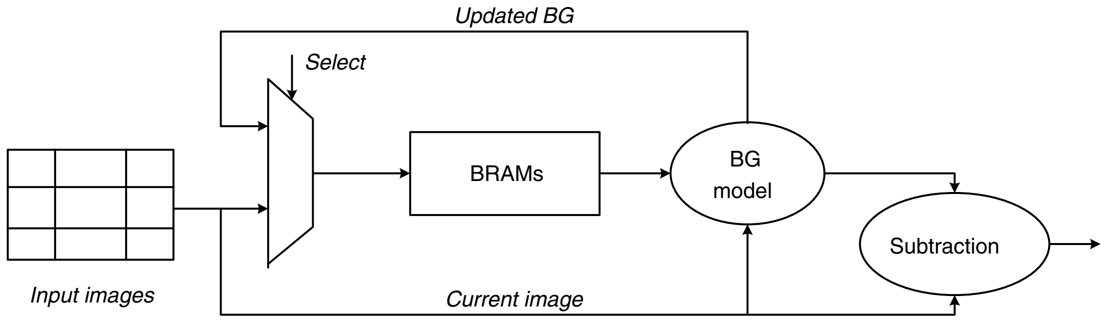


Fig. 14 Architecture for background modeling and subtraction.

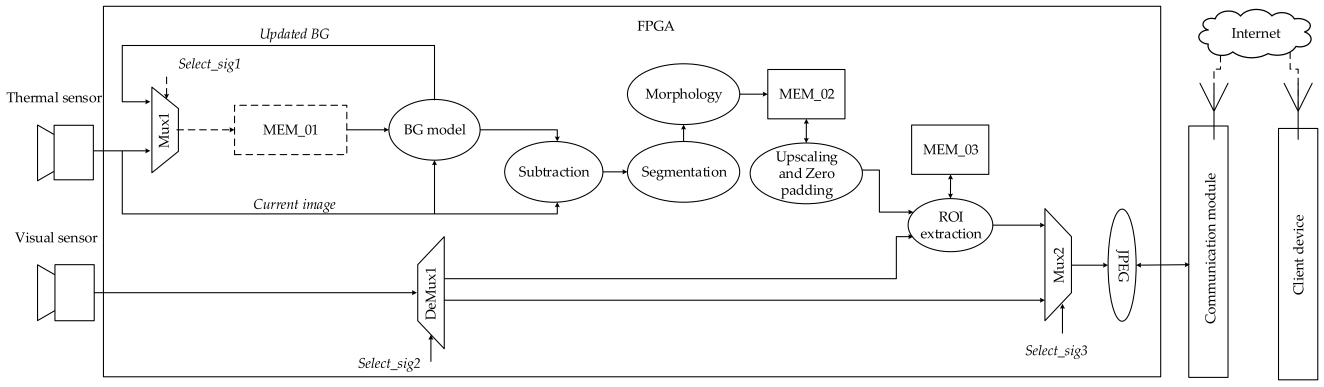


Fig. 15 Preprocessing architecture 1: medium complexity and relatively high memory requirement.

complete image will be transmitted for situational awareness and updates of changes in infrastructure.

5.2.1 *Memory requirement*

Architecture 1 requires memory buffers in three stages of the imaging pipeline flow. In the first stage, memory (MEM_01) is required for the storage of the thermal image, in the second stage, memory (MEM_02) is required for the synchronization of the thermal and visual images, and in the third stage, memory (MEM_03) is required for the extraction of the ROI in the two stages, as discussed in Sec. 4.6.1. The bit-widths of MEM_01, MEM_02, and MEM_03 are 8, 1, and 24, respectively.

5.3 *Architecture 2: Low Complexity and Low Memory Requirement*

In architecture 2, as shown in Fig. 16, the bilevel ROI coded data is transmitted frequently whereas color images are transmitted occasionally depending on the requirements of the client device. In this architecture there are lightweight tasks in the preprocessing flow and the memory requirement is lower, as compared to architecture 2. The difference between the two, is that instead of color ROI, binarized ROI will be transmitted from the camera node. This in turn requires a binary classifier on the client device.

5.4 *Resource Utilization, Latency, Power Consumption and Output Data of the Two Architectures*

The resource utilization in terms of logics, memory and DSP utilization, processing latency, and output produced data of the two architectures are shown in Table 4. The logic utilizations of the two architectures show that a smaller FPGA device can be utilized, provided the device has sufficient internal memory (BRAMs) as required by the two architectures. This in turn will help to circumvent the quiescent power consumption associated with the unused FPGA fabric.

Depending on the requirements of the client side, architecture 1 can be used if the classification is performed on color images and architecture 2 can be used if the classification is performed on binary images on the client device. Architecture 1 requires approximately 6.6 times greater memory and produces ∼32 times more output data, at a cost of ∼2 times more logics, as compared with architecture 2. The large amount of output data will contribute to greater communication latency and costs.[24] The output data could vary depending on the movement and distance of the object, therefore, standard deviation from average output data for single objects is calculated. The percentage variation in standard deviation from average output data is reported to be less than 24% for both architectures.

It is worth mentioning that the preprocessing architectures could be applied to multiple object scenarios. However, this would require that the classification algorithm on the client device can handle multiple objects in a single image. The performance of the system is 53 frames/s, which is dependent on the clock frequency of the image sensors instead of separate system clock frequency. This helps to avoid unnecessary synchronization and reduce the power consumption as switching activity occurs according to the speed of the incoming pixels clock. The power consumption is reported to be 163 and 141 mW for architecture 1 and architecture 2,

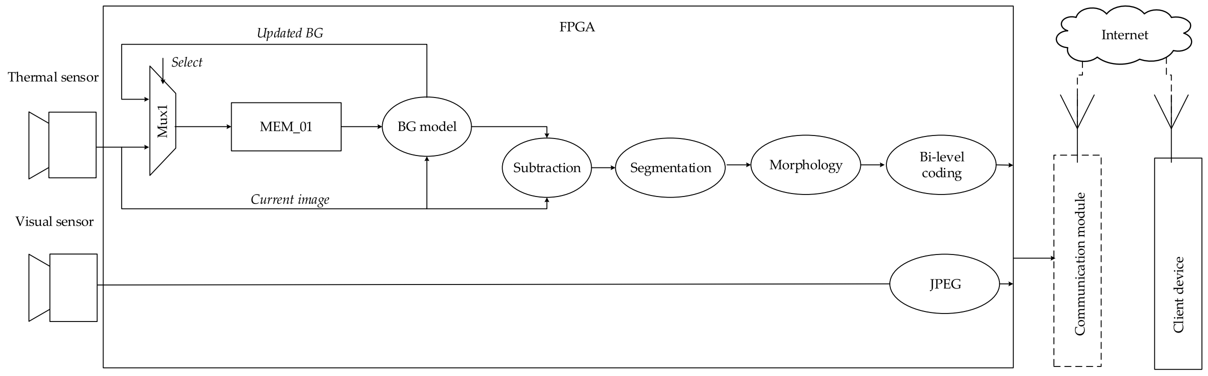


Fig. 16 Preprocessing architecture 2: low complexity and relatively small memory requirement.

Table 4 Resource utilization, power consumption, performance, and output data.

						Output data			
						Frequently (Bytes)		Occasionally (KB)	
	Logics	DSP	Memory 18 Kbits* 36 Kbits+	Power consumption (mW)	Processing latency (ms), FPS	Average	Standard deviation	Average	Standard deviation
Architecture 1	2852	13	15*, 212+	163	18.7, 53 FPS	4468.9	1101.3	38.5	1.2
Architecture 2	5638	13	14*, 26+	141	18.8, 53 FPS	142.5	33.6	38.5	1.2

respectively. This concludes that suitable design and architecture methods can enable existing FPGA devices to handle the processing of two image streams for real-time applications.

5.5 *Comparison with Existing Systems*

Individual algorithm parameters such as efficiency and complexity have been compared before considering it for hardware implementation. The resultant images after background modeling subtraction and JPEG coding algorithms have been compared with ground truth before selecting the algorithm for implementation. The comparison result is shown in Secs. 4.6.3 and 5.1.1. As discussed in related work, the current research trend of multispectral camera systems is processing algorithms, rather than hardware implementation. Therefore, direct comparison with other systems is challenging because of the unavailability of published results from the hardware implementation perspective. However, the future potential of thermal cameras and quick processing engines, e.g., FPGA in safety applications, such as the automotive industry and industrial automation,[27,28] will attract the research community to this area. Nonetheless, this work will serve as a reference system for future hardware implemented multispectral imaging systems.

6 Conclusion

This paper explored suitable preprocessing architectures for FPGA-based embedded smart cameras integrating thermal and visual image sensors. The need for such a system arose from outdoor surveillance applications with real-time and robust surveillance requirements. This work integrates thermal and visual image sensors with FPGA so that the solution has lower processing time and is able to work in different weather and environmental conditions. In relation to this, two preprocessing architectures are explored, which can effectively segment the objects from the background and reduce the amount of transmission data with the help ROI extraction and image coding schemes. Both architectures transmit data in two modes (1) frequent transmission and (2) occasional transmission in order to transmit activity information and changes in the environment. Depending on the classification algorithms on the client device, one architecture transmits color ROI data frequently and other architecture transmits bilevel ROI data frequently. The architecture with binary ROI consumes 6.6 times less memory, resulting in 32 times less output data. A rigorous experimental data on pedestrian detection application shows that the proposed architectures can offer a frame rate of 53 frames per second which makes the underlying system suitable for real-time applications.

Acknowledgments

The authors would like to acknowledge Mid Sweden University's Qualification Program, the Swedish Knowledge Foundation and Ericsson Research Foundation for funding this research work.

References

1. M. Imran et al., "Analysis and characterization of embedded vision systems for taxonomy formulation," *Proc. SPIE* **8656**, 86560J (2013).
2. M. Reisslein, B. Rinner, and A. Roy-Chowdhury, "Smart camera networks," *Computer* **47**, 23–25 (2014).
3. M. Magno et al., "Multimodal video analysis on self-powered resource-limited wireless smart camera," *IEEE J. Emerging Sel. Top. Circuits Syst.* **3**, 223–235 (2013).
4. M. Goel et al., "Hyper cam: hyperspectral imaging for ubiquitous computing applications," in *Proc. of ACM Int. Joint Conf. on Pervasive and Ubiquitous Computing*, pp. 145–156 (2015).
5. T. T. Zin et al., "Fusion of infrared and visible images for robust person detection," in *Image Fusion*, O. Ukimura, Ed., InTech (2011).
6. Axis Communication AB, "Some like it hot thermal cameras in surveillance table of contents," White Paper, http://www.axis.com/ (2009).
7. S. Ibrahim and M. Wirth, "Visible and IR data fusion technique using the contourlet transform," in *Int. Conf. Computational Science and Engineering*, Vol. **2**, pp. 42–47 (2009).
8. M. Imran et al., "Architecture of wireless vision sensor node with region of interest coding," in *IEEE Int. Conf. on Networked Embedded Systems for Every Application*, pp. 1–6 (2012).
9. M. Imran et al., "Low complexity FPGA based background subtraction technique for thermal imagery," in *Proc. of the 9th Int. Conf. on Distributed Smart Cameras*, pp. 1–6 (2015).
10. C. Bourrasset, J. Serot, and F. Berry, "FPGA-based smart camera mote for pervasive wireless network," in *Seventh Int. Conf. on Distributed Smart Cameras*, pp. 1–6 (2013).
11. S. Soro and W. Heinzelman, "A survey of visual sensor networks," *Adv. Multimedia* **2009**, 1–21 (2009).
12. A. Kandhalu, A. Rowe, and R. Rajkumar, "DSPcam: a camera sensor system for surveillance networks," in *Third ACM/IEEE Int. Conf. on Distributed Smart Cameras*, pp. 1–7 (2009).
13. B. Rinner and T. Winkler, "Privacy-protecting smart cameras," in *Proc. of the Int. Conf. on Distributed Smart Cameras*, pp. 5 (2014).
14. L. Gasparini et al., "An ultralow-power wireless camera node: development and performance analysis," *IEEE Trans. Instrum. Meas.* **60**, 3824–3832 (2011).
15. A. Kandhalu, A. Owe, and R. Ajkumar, "DSPcam: a camera sensor system for surveillance networks," in *Third ACM/IEEE Intl. Conf. on Distributed Smart Cameras*, pp. 1–7 (2009).
16. T. Mouats and N. Aouf, "Fusion of thermal and visible images for day/night moving objects detection," *Sensor Signal Processing for Defence*, Edinburgh, pp. 1–5, IEEE (2016).
17. D. R. S. Technologies, "Tamarisk®320 17 μm 320 × 240 long wave infrared camera electrical interface control document," www.drsinfrared.com (2013).
18. IDS Imaging Development Systems GmbH, "UI-1640SE-C-HQ specification sensor UI-1640SE-C-HQ," pp. 1–2, www.en.ids-imaging.com/ (2015).
19. Xilinx, Inc, "Artix-7 family and Xilinx power tools tutorial, Xilinx ISE 14.7," www.xilinx.com/ (2010).
20. Transportstyrelsen, "Trafiksäkerheten i Sverige," *Statistik och analys över järnväg, luftfart, sjöfart och väg för 2014, Rapporten*, www.transportstyrelsen.se/ (2015).

21. D. Barney, D. Haley, and G. Nikandros, "Calculating train braking distance," in *Proc. of the Sixth Australian Workshop on Safety Critical Systems and Software*, Vol. **3**, pp. 23–30 (2001).
22. Y.-C. Chung, J.-M. Wang, and S.-W. Chen, "Progressive background images generation," in *15th IPPR Conf. on Computer Vision, Graphics and Image Processing* (2002).
23. STC thermal images, http://apachepersonal.miun.se/~muhimr/STC_thermal_image_sets.zip (2014).
24. M. Imran, M. O'Nils, and N. Lawal, "Energy driven selection and hardware implementation of bi-level image compression," in *Proc. of the Int. Conf. on Distributed Smart Cameras* (2014).
25. M. Imran et al., "Low complexity background subtraction for wireless vision sensor node," in *2013 Euromicro Conf. on Digital System Design*, pp. 681–688 (2013).
26. CAST, Inc., "JPEG-C baseline JPEG codec," http://www.xilinx.com/ (2015).
27. A. Corporation, "A safety methodology for ADAS designs in FPGAs in ADAS applications," White Paper, pp. 1–18 (2013).
28. C. V. Systems, "BMW incorporates thermal imaging cameras in its cars," *Application Story* http://www.flir.com/ (2016).

Muhammad Imran is a research assistant in the STC research center, Mid Sweden University, Sweden. He received his MS degree in electrical engineering from Linköping University, Sweden, in 2007 and a PhD in electronics design from Mid Sweden University, Sweden, in 2013. His research interests include design, development, and implementation methodologies for embedded vision processing systems, especially low energy and performance aspects of embedded smart camera systems both on software and hardware platforms.

Bernhard Rinner received his MSc and PhD degrees in telematics from Graz University of Technology, Austria, in 1993 and 1996, respectively. He is a full professor and chair of pervasive computing at Klagenfurt University. He held research positions with Graz University of Technology from 1993 to 2007 and with the Department of Computer Science, University of Texas at Austin, from 1998 to 1999. His current research interests include embedded computing, embedded video and computer vision, sensor networks, and pervasive computing. He has been cofounder and general chair of the ACM/IEEE International Conference on Distributed Smart Cameras. He is a senior member of IEEE.

Sajjad Zandi Zand received his BS degree in electronics design from Mid Sweden University, Sweden, in 2015. Currently, he is pursuing master-by-research degree in the STC Research Center, Mid Sweden University, Sweden.

Mattias O'Nils received his BSc degree in electrical engineering from Mid Sweden University in 1993 and his Licentiate/PhD in electronics system design from the Royal Institute of Technology in Stockholm, Sweden, in 1996 and 1999, respectively. He is a professor of the department and leads a research center STC at Mid Sweden University. His research interest includes hardware/software partitioning, methods for performance optimization by migration of functionality to hardware, specification, design, and verification of hardware/software interfaces, low-power optimization of FSM implementation, compiler technology memory optimization, and synthesis for real-time video systems.