An Ultra-Low-Power Contrast-Based Integrated Camera Node and its Application as a People Counter

Leonardo Gasparini
Dept. of Information Engineering
and Computer Science
University of Trento,
38123 Povo (TN), Italy
gasparini@disi.unitn.it

Roberto Manduchi
Dept. of Computer
Engineering
University of California
Santa Cruz, CA 95064, USA
manduchi@soe.ucsc.edu

Massimo Gottardi
Smart Optical
Sensors and Interfaces
Fondazione Bruno Kessler
38123 Povo (TN), Italy
gottardi@fbk.eu

Abstract

We describe the implementation in a self-standing system of a novel contrast-based binary CMOS imaging sensor. This sensor is characterized by very low power consumption and wide dynamic range, which makes it attractive for wireless camera network applications. In our implementation, the sensor is interfaced with a Flash-based FPGA processor, which handles data readout and image processing. This self-standing camera node is configured as a system for counting persons walking through a corridor. Simple features are extracted from each image in a video stream at 30 fps. A classifier is designed based on the temporal evolution of these features, which is modeled as a Markov chain. The video stream is then segmented into intervals corresponding to individual persons crossing through the field of view. Experimental results are shown in cross-validated tests over real sequences acquired by the camera.

1. Introduction

The past decade has witnessed a spate of research projects in the area of low-power, wireless camera networks for surveillance and monitoring. The general trend is to embed as much processing as possible within the camera node, in order to minimize the amount of data to be transmitted. However, the power consumption distribution between image acquisition, processing and communication needs to be carefully assessed. For example, depending on the hardware employed in the node, the energy consumed for processing an image frame may exceed the energy required to transmit the frame [7].

One possible way to limit power consumption in a sensor node is to reduce the amount of data from the camera, without sacrificing relevant information content. This can be achieved through CMOS Active Pixel Sensors (APS) technology, which enables embedding of some image processing components within the sensor itself. For example, contrast-based cameras [2, 11] extract a coarse edge representation of the image, resulting in reduced data rate from the sensor and thus reduced power dissipation. For some applications, this succinct image representation is sufficient (and perhaps even desirable).

A novel APS camera architecture that is amenable to battery-powered systems due to its extremely limited power consumption (of approximately 100 μW at 50 fps) was recently introduced by Gottardi *et al.* [3]. The sensor has a resolution of 64 $\times$ 128 pixels and a variable frame rate of up to thousands of fps. It produces binary contrast images at a wide range of illumination, thanks to an inherent mechanism of self-exposure. The camera can perform on-sensor differencing of the contrast data between two consecutive images, which provides a crude but effective procedure of background removal and event detection. This feature also provides a simple mechanism to automatically switch the camera from Idle mode to Active mode when moving objects are detected in the scene. During Idle mode, the camera keeps acquiring data but rather than dispatching it to the output, it only produces the number of pixels where some motion was detected. This strategy reduces power consumption to about 30 μW.

In this paper we describe the implementation of this contrast-based camera as a self-standing counter for persons transiting through a “gate” (a corridor or a door). This simple application highlights the main features of the sensor and provides an example of integration with a processing unit. People counting is normally performed using wire-trapping with one or more fixed photocells and illuminators. Other sensing modalities have been proposed, including PIR sensors [4] and pressure sensors [8]. Camera-based systems for people counting are attractive because they can be very compact and may require little or no calibration (compared, for example, to wire-trapping). These characteristics make them suitable for impromptu surveillance operations. If equipped with a radio (e.g. ZigBee) to periodically transmit the value of the counters, a battery-operated camera-based people counter would represent an ideal solution when there is the need to monitor a certain portion of space and wiring the system is not an option.

Vision-based human detection and tracking has been studied for decades. Rather than a full-fledge detector or tracker, our system simply counts the number of people (or moving objects) transiting in one direction or another. This is similar in spirit to the work by Teixeira and Savvides [13], who used an histogram-based algorithm on an address-event imager architecture [12], or to the work by Lefloch [6], who used simple blob representation. We use a Flash-based Field Programmable Gate Array (FPGA) for data readout from the camera and for data processing, producing symbolic information indicating the passage of a person through the gate and the direction of motion.

The on-sensor processing capabilities of the imager enable a well-defined subdivision of tasks between the sensor and the processing unit. The sensor detects when there is something moving in the scene, and computes a coarse edge map of moving objects. The processor analyzes the sequence of frames thus produced during active periods, to determine when a person enters or exits the scene and its direction of motion. From each frame (at a rate of 30 fps), it extracts a very simple feature: the presence of a moving person within two disjoint image areas (dubbed Virtual Inductive Loops [14]). In spite of the extreme simplicity of this representation (only 2 bits defining the "state" of a frame), the evolution of this feature through time is shown to be informative enough to enable satisfactory counting. We model the sequence of maximal segments of frames with the same state as a second order Markov chain, which allows us to quickly compute the likelihood of a given sequence of frames. Then, the video is segmented into labeled intervals corresponding to individual persons crossing through the visual field, using a maximum-likelihood criterion. This operation is seen to be equivalent to finding the maximum cost path in a graph, which is computed using the Dijkstra algorithm. Quantitative benchmarking of the segmentation performance are presented based on two long sequences of images acquired in different environments. The overall system (including camera and processor) consumes about 9 mW. This is almost two order of magnitude less than the power consumption of other low-power integrated camera systems reported in the literature [10, 5, 1].

2. The Binary Contrast-Based Camera

The imaging sensor used in our prototype is organized in an array of 128 x 64 pixels. The chip was integrated in a 0.35 μm process DP TM CMOS featuring a pixel size of 26 μm $\times$ 26.5 μm with a fill factor of 20%. The die size, with pads, is of 11 mm^2.

A certain amount of image processing is embedded both at the pixel-level and at the column-level [3]. At the pixel level, the spatial contrast is extracted during exposure time, over a kernel of three neighbouring pixels. The sensor measures and binarizes the contrast V_C, defined as the difference in voltage between the most illuminated and the least illuminated pixel in the triplet, when the least illuminated one has voltage equal to a preset value. It can be shown that the contrast V_C is linearly related to the ratio of irradiances between the most illuminated and the least illuminated pixel in the triplet. By measuring *ratios* of irradiances, rather than the irradiance itself, the sensor is able to give meaningful results at a wide illumination dynamic range. If the contrast V_C is above a certain user-defined threshold, the reference pixel stores a '1' into the local memory, otherwise a '0' will be frozen in. This can be seen as a crude form of edge map.

At the column level, the visual information is binary, hence the readout process can be accomplished by detecting and dispatching only the asserted pixels, rather than delivering every single pixel. A characteristic of this sensor is that the pixel array is equipped with two bit-lines, one for the current binary value (BTLA), the other for a *reference* binary value (BTLB). The system compares BTLA with BTLB at each pixel producing three possible values of *disparity*: zero, positive, or negative. This allows one to compare the current edge image with a previously stored binary image, directly on the sensor.

In order to minimize the activity at the sensor interface, the readout phase has been chosen to be asynchronous. The whole image is scanned, row by row. As soon as a (negative or positive) disparity is detected (i.e., the pixel is asserted), the column address of the current pixel is dispatched to the output, along with the sign of the disparity. A clock pin at 80 MHz signals when a new asserted pixel has been found and its address and sign are ready for readout. An End of Frame pulse is produced when all pixels in the frame have been checked.

The entire sensor array takes about 150 μs to be read out, regardless the number of asserted pixels. This means that, under proper illuminating conditions, the maximum achievable frame rate is of 6000 fps. Since only the addresses of the asserted pixels are provided in output, this sensor achieves a data bandwidth reduction of about 70%–80% with respect to a traditional raster-scan imager in typical condition and using the Motion mode discussed next.

2.1. Edge vs. Motion mode

The imager allows one to store a reference binary image, which is compared with the edge image being acquired. This can be useful when one is interested in detecting only moving objects in the scene. If the camera is fixed in front

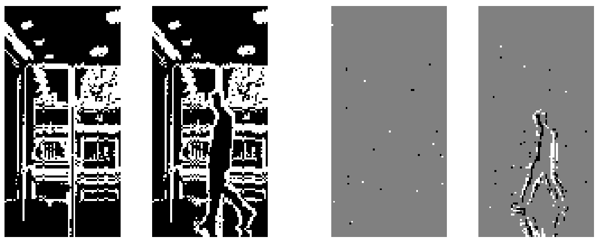

Figure 1. Images taken by the our camera from the same location. Each pair includes an image of a static background and one image with a person walking by. Note that in Motion mode each pixel can take three values, since it is the difference between two binary values. The background is effectively removed in Motion mode (the remaining active pixels in the static background are due to noise).

of a static background, a simple way to detect textured moving image areas is to compute the pixel-by-pixel difference between two consecutive frames. This can be achieved by means of on-sensor processing by simply loading as reference image the edge image of the previous frame (*Motion* mode). A comparison of Motion mode vs. Edge mode (i.e., the contrast binary data produced by the sensor) is shown in Fig. 1. Note that each pixel in an image in Motion mode has three possible values, since it is generated by the difference between two binary values. In general, only a small portion of pixels in the image, typically corresponding to the edges of moving objects, are asserted in Motion mode.

The Motion mode is useful to remove edges belonging to a static background, thus enabling focusing of subsequent processing only on moving areas. When used in conjunction with the Active/Idle mode strategy described below, it enables power-efficient sensor control strategies.

2.2. Active vs. Idle Mode

The sensor can be externally set to function in *Active* or *Idle* mode. In Active mode, addresses of asserted pixels, their signs, and other control signals are dispatched in output. The average power consumed at 50 fps, when 25% of the pixels are asserted, is of 100μW. This consumption is one to two order of magnitudes less than for equivalent sensors [3]. In Idle mode, this data is not dispatched, and thus is not accessible. The only data accessible from outside is the value of a counter register, which is incremented each time an asserted pixel is found, and reset at the end of a frame.

The Idle mode can be exploited when the sensor is set to Motion mode and the scene is expected to be static for extended periods of time. When the scene is static, only very few pixels are asserted in Motion mode at each frame (See Fig. 1). An external processor may periodically check the value of the asserted pixel counter, compare it with a threshold, and decide to switch the sensor to Active mode only when a significant number of asserted pixels is found, meaning that there is some activity in the scene worth observing. This represents a very simple and power-efficient assisted wake-up system. Note that by keeping the sensor in Idle mode for as long as possible, significant power saving can be achieved, as the sensor in Idle mode consumes only 30% of the power consumed while in Active mode [3].

3. An Integrated People Counter

We developed an integrated system based on the sensor described in Sec. 2, customized as an autonomous counter for persons passing through a “gate”, such as a door or a corridor. The camera node is attached to the ceiling, placed such that its field of view encompasses the width of the corridor or of the door. Persons are seen entering from the top or bottom of the image and exiting through the opposite end. The task is to count the number of persons transiting in each direction. It should be clear that this is just one possible (and simple) way to use this system; the camera node could be reconfigured for many other monitoring tasks.

Data is processed at full frame rate (30 fps), while using a relatively simple image analysis procedure. The dynamic characteristics of the application dictate this frame rate. A person walking at normal speed can transit through the field of view of the camera in as little as half a second. Robust estimation of the direction of motion requires analysis of several image frames, hence the need for frame rates in excess of 10 fps. Furthermore, when two persons transit through the field view in close sequence, a high frame rate is necessary to correctly separate the two, otherwise the system may incorrectly count just one passage.

The algorithm for people counting based on the binary data produced by the camera is presented in the next two sections.

3.1. Image Summarization

We borrow the concept of “Virtual Inductive Loops” (VIL) from Viarani [14], who introduced it for visual traffic monitoring. A VIL mimics the action of an inductive loop, such as those embedded in the road pavement for car counting. In our case, a VIL is simply a particular, typically rectangular region of the image. The system counts the number of asserted pixels in a VIL, reporting a binary ‘active’ or ‘inactive’ state for the VIL according to whether the number of asserted pixels is above a fixed threshold. This threshold is chosen empirically based on the noise characteristics of the system. In order to increase robustness, hysteresis is implemented. If the image has N_{VIL} virtual loops, the overall state is summarized by a binary word with N_{VIL} bits. In our experiments, we used 2 VILs, placed as shown in Fig. 2. This is admittedly a crude summarization (a 2-

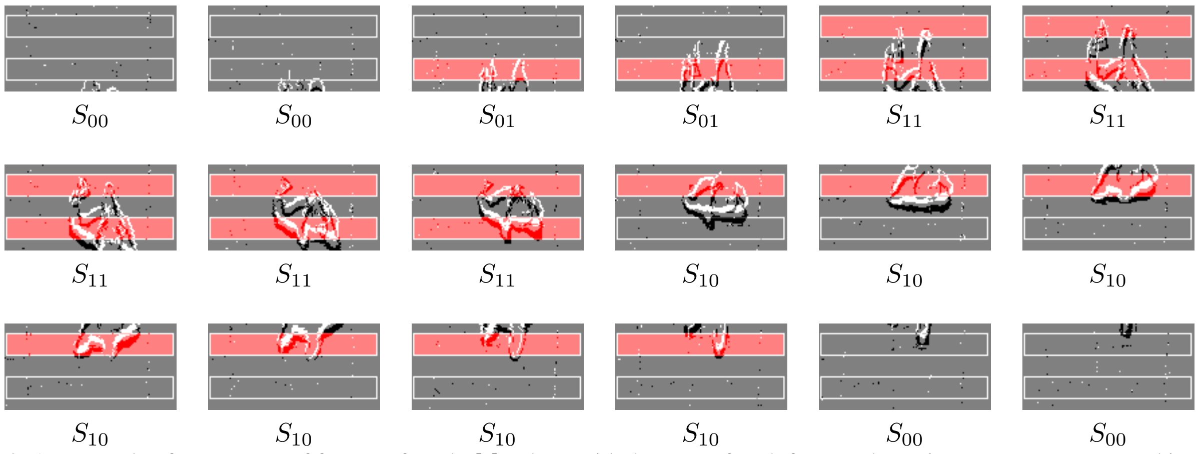

Figure 2. An example of a sequence of frames of mode M_2, along with the state of each frame. The active VILs are represented in red.

bit *state*) of the binary image content. Yet, this representation is sufficient to accomplish our task with good accuracy. In fact, this summarization procedure has two main advantages. Firstly, the sequence of 2-bit states can be processed very efficiently by the low-power FPGA sensor node architecture. Secondly, counting the number of asserted pixels within a VIL can be achieved *without* the need to transfer the whole binary image data. The FPGA simply counts the number of asserted pixels as they are produced by the sensor, checking their address to ensure that it is within either VIL.

In order to simplify our representation further, we group consecutive frames characterized by the same 2-bit state into *segments*. Thus, the video is represented by the sequence of segments $(S^1, S^2, \ldots, S^N)$, where superscripts represent the index of the segment in the sequence. When we want to indicate a particular value for a segment, we write its 2-bit representation in the subscript (e.g. S^i_{01}). Each segment contains frames with the same state, and two subsequent segments have frames with different states.

We will say that a frame is of *mode* M^m_1 when it is an image of the m-th person seen in the scene, and this persons is transiting from the top to the bottom of the image. Mode M^m_2 represents the m-th person seen in the scene, who is moving from the bottom to the top of the image. In the following, when the index m of the person transiting is irrelevant, we will drop the superscript, in which case mode M_1 and M_2 simply represent the fact that a persons is walking in a particular direction. For example, Fig. 2 shows a sequence of frames of mode M_2[1]. We also introduce mode M_0 representing sequences with no persons visible in the scene. An *interval* is a maximal sequence of segments with the same mode M^m_k. Note that modes and thus the extent of each interval are not directly observable, and must be inferred from the observable states.

[1]Although the sequence of states in Fig. 2 is fairly typical for a mode M_2, much more complex sequences can be observed. This is because different parts of one's body may activate the VILs in various orders.

3.2. Mode Assignment

The goal of our algorithm is to assign to each segment its correct mode. The algorithm is based on a statistical generative model. We assume that, within an interval, the sequence of segments being generated forms a Markov chain. However, a Markov chain of the first order would not be sufficiently descriptive. A Markov chain of the first order is such that $P(S^n|S^{n-2}, S^{n-1}, M_i) = P(S^n|S^{n-1}, M_i)$, where $P(S^n|S^{n-2}, S^{n-1}, M_i)$ is the probability of the segment S^n appearing after segments S^{n-2} and S^{n-1} in an interval of mode M_i. But this is unrealistic in our case, as shown by the following example. The conditional probability $P(S^3_{01}|S^2_{00}, M_2)$ can be expected to be close to 0.5 (for example, it may correspond to the beginning of the sequence, as a person enters from below and crosses over the second VIL). This is not the case, though, for $P(S^3_{01}|S^1_{10}, S^2_{00}, M_2)$, which is bound to take a very small value since it is an unlikely sequence during upwards motion. Using a second-order Markov chain removes this ambiguity.

Assume that a sequence $\mathbf{S} = (S^1, S^2, \ldots, S^N)$ is contained in an interval of mode M^m_k (as denoted by the writing $\mathbf{S} \subset M^m_k$). The likelihood of observing $\mathbf{S}$ under this assumption is:

$$P(\mathbf{S}|\mathbf{S} \subset M^m_k) = P(S^1|M^m_k)P(S^2|S^1, M^m_k) \cdot \cdot \prod_{n=3}^{N} P(S^n|S^{n-2}, S^{n-1}, M^m_k) \quad (1)$$

In addition, we introduce two fictitious (and thus non-observable) states, S_I and S_F, that are assumed to occur at the beginning and at the end (respectively) of each interval. If an observed sequence $\mathbf{S} = (S^1, S^2, \ldots, S^N)$ is assumed to encompass a whole interval (as denoted by $\mathbf{S} = M^m_k$), then the probability of observing $\mathbf{S}$ should be modified as follows:

$$P(\mathbf{S}|\mathbf{S}=M_k^m) = P(S^1|S_I, M_k^m)\cdot \quad (2)$$
$$\cdot P(S^2|S_I, S^1, M_k^m)\cdot \prod_{n=3}^{N} P(S^n|S^{n-2}, S^{n-1}, M_k^m)\cdot$$
$$\cdot P(S_F|S^{N-1}, S^{N-2}, M_k^m)$$

In practice, $P(S_i|S_I, M_k^m)$ and $P(S_j, S_i|S_I, M_k^m) = P(S_j|S_I, S_i, M_k^m)P(S_i|S_I, M_k^m)$ represent the probability that state S_i or the sequence (S_i, S_j) are seen at the beginning of an interval. $P(S_F|S_i, S_j, M_k^m)$ represents the probability that the sequence (S_i, S_j) is seen at the end of an interval. Note that the marginal probability $P(S_I|\mathbf{S}=M_k^m)$ is equal to 1, since the first state in an interval is always S_I. The first and second order conditional probabilities in (2) can be learned from labeled training data sets.

The conditional probabilities in (1) and (2) are independent of the person index m. However, the segments must all belong to the same mode M_k^m. For example, consider the case with two persons traversing right after each other in the same direction. The first person, with index $m = 1$, is seen in the intervals $\mathbf{S}^1$ (S^1 through S^R), while the second person ($m = 2$) is seen in $\mathbf{S}^2$ (S^{R+1} through S^N). In this case,

$$P((\mathbf{S}^1, \mathbf{S}^2)|\mathbf{S}^1 = M_k^1, \mathbf{S}^2 = M_k^2) \quad (3)$$
$$= P(\mathbf{S}^1|\mathbf{S}^1 = M_k)P(\mathbf{S}^2|\mathbf{S}^2 = M_k)$$

where $(\mathbf{S}^1, \mathbf{S}^2)$ is the juxtaposition of the two segment sequences, and each sequence is equal to a distinct interval.

In our model we assume that intervals with mode M_0 contain only one segment (of state S_{00}). Although this is not always the case in practice (noise may activate a VIL even in the absence of persons transiting), this assumption simplifies our algorithm considerably. In addition, we assume that intervals of mode other than M_0 have at least two segments. Finally, we assume that any given interval cannot have length more than a pre-determined value $T_{\max}$, where $T_{\max}$ is defined in terms of frames, rather than of segments. Basically, $T_{\max}$ is the length (in frames) of the longest observed interval. In our tests, we set $T_{\max}$ = 60 frames (2 seconds at 30 fps).

Let $\mathbf{S}$ be a candidate interval that has been assigned to mode M_k^m. We can compute its likelihood as by (2)

$$P_k^m = P(\mathbf{S}|\mathbf{S} = M_k^m) \quad (4)$$

For a given assignment, we can segment the sequence of frames into the list of intervals $(\mathbf{S}^1, \mathbf{S}^2, \ldots, \mathbf{S}^M)$, where the m-th interval $\mathbf{S}^m$ is assigned to mode $k(m)$, and has likelihood $P_{k(m)}^m$. Armed with these definitions, we can state our problem:

Problem: Find the assignment of modes to segments that maximizes $\prod_{m=1}^{M} P_{k(m)}^m$.

Differently stated, we find the sequence of people crossing in both directions that maximizes the likelihood of the observed data. Then the number of people who traversed from top to bottom (or viceversa) in a certain period of time is equal to the number of distinct indices m with $k(m)$ equal to 1 (or 2).

In order to tackle this problem, it is useful to represent the set of segments and intervals as an acyclic, directed multigraph. Each node represents a segment. A directed edge from node S^i to node S^j with $j > i$ represents the sequence $\mathbf{S} = (S^{i+1}, S^{i+2}, \ldots, S^j)$. Directed edges are formed between each node and any subsequent node that could legally be part of the same interval (remember that segments in an interval must have the same mode). More precisely, node S^i has directed outwards edge pairs (one for M_1 and one for M_2) to all nodes S^j with $j - i \geq 2$ and such that the number of frames between the beginning of S^{i+1} and the end of S^j is no larger than $T_{\max}$. If a node is of type S_{00}, it only has one directed outwards edge to the next segment in the sequence. An edge representing an interval $\mathbf{S}$ for mode M_k has cost equal to $P(\mathbf{S}|\mathbf{S} = M_k)$. Note that the number of edges in the graph is bounded from above by $N_{\text{segm}} \cdot T_{\max}$, where N_{segm} is the total number of segments in the video. This is a very conservative upper bound, since it is highly unlikely that all segments are made up by a single frame. In practice, within an interval of $T_{\max}$ frames, there are about 5 segments on average.

Thus, our problem becomes one of finding the maximum cost path in the multigraph from the first to the last node (segment), where the cost of a path is the product of the costs of the edges in the path. Since all costs are positive and less than or equal to one, this problem is equivalent to one of finding a minimum cost path where the path cost is the sum of the (non-negative) negative logarithms of $P(\mathbf{S}|\mathbf{S} = M_k)$. Note that if two nodes have multiple edges linking them, only the edge with the maximum cost needs to be kept. Dijkstra's algorithm can then be used to find the maximum cost path in the resulting graph, with a complexity of $\mathcal{O}(N_{\text{segm}})$ (since the number of edges grows linearly with the number of nodes).

4. Hardware Prototype

We implemented the people counting algorithm described above on an FPGA based prototype board. The board consists of an Actel IGLOO M1-AGL600 FPGA, two 512 kB SRAMs, two 8 MB Flash memories, a crystal oscillator, a programmer for the FPGA, a USB to RS-232 converter chip, expansion connectors, leds and switches. A block scheme of the system is depicted in Fig. 3.

The hardware core of the unit is represented by the FPGA. Most FPGAs are built on volatile SRAM technology, which means that the device needs to be reconfigured each time it is powered up. This considerably increases the

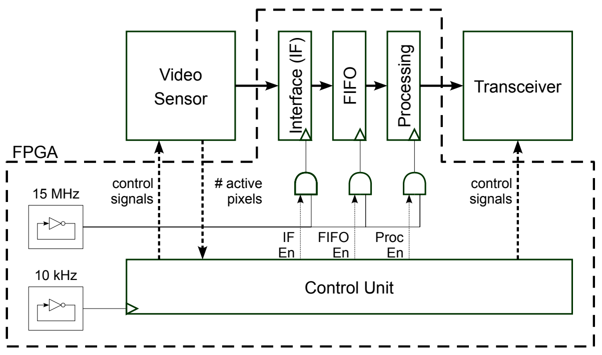

Figure 3. Block scheme of the node.

complexity of the system, with substantial effects on the power requirements of the whole system. A SRAM-based device must be paired with a non-volatile memory from which it loads its configuration, thus increasing the number of components required on the board. Moreover, configuration information loading is a power-consuming process. The Actel IGLOO Flash-based FPGA family overcomes these limitations. Thanks to Flash technology, these devices preserve their configuration when powered down. In addition, they are characterized by ultra-low static consumption: the power consumed by the device when the clock is not toggling is negligible. These advantages are obtained at the expense of a higher cost per unit and a lower supported clock speed with respect to an equivalent SRAM-based FPGA.

Clock speed is critical when communication with other devices is asynchronous. This is the case for the binary camera used in this prototype, which provides output data at rates as high as 80 MHz. The FPGA communicates with the camera through one of the expansion connectors available on the board. This provides the sensor with the control signals required for image acquisition and for transmitting the address and the sign of the non-zero pixels. Data from the sensor may then be directly processed, or it may be stored in an internal soft memory for later analysis. Unfortunately, the speed at which the sensor data is received cannot be supported by the memory/processing system directly. Therefore, part of the logic available in the FPGA is dedicated to slowing down the access to the soft memory through a serial to parallel converter. In this way, only a small portion of the program present in the FPGA needs to execute at a high speed, while the remaining parts may be clocked at lower frequencies.

To exploit the low power capabilities of the FPGA, we use two clock sources, one running at a a few kHz, and one at a few MHz. Different components implemented in the FPGA are clocked with one of the two clocks depending on the complexity and priority of the task that they perform. We implemented both clocks as ring oscillators internal to the FPGA, avoiding the need for external oscillators.

The control unit (which is always running) is a Finite State Machine (FSM) that generates the control signals for all the other components, internal and external to the FPGA. It is responsible for setting the camera in Idle or Active mode, enabling significant power saving during idle periods. It enables the memory interface, the FIFO and the processing unit when needed, and controls the transceiver. Cyclically, with a period equal to the frame period, it activates and deactivates the other components by providing reset signals and controlling their clock sources through a number of AND gates. This prevents the internal circuitry to switch its state when not needed, thus minimizing dynamic power consumption. The cycle begins with the generation of the timing signals needed by the camera to acquire a new frame. During acquisition, only the camera and the control unit need to be on. Then, the FPGA activates the internal FIFO memory and its interface before starting the readout process. Upon completion of this task, the control unit resets the sensor and disables the FIFO interface, while enabling the processing unit. The control unit then waits for the algorithm to be executed before turning off the processing unit and transmitting the results.

The FIFO memory is an 8 kBit soft memory that can contain the whole image. Note that, as discussed in Sec. 3.1, it is not necessary to store the image from the camera for the functioning of our people counter system: only the number of asserted pixels in the two VILs needs to be accessed. We implemented the internal storage in order to facilitate system reconfiguration for other types of algorithms involving some form of image processing, as well as to facilitate debugging by transmitting the whole image to a host PC.

The people counting algorithm is executed by the processing unit. At each frame being received, the state S is computed, by counting the number of asserted pixels in each VIL. The graph is generated with the aid of three FIFO memories. The first FIFO is used as temporary storage space for the generation of edges and the associated costs to the current node. The second one stores the cumulative costs of the paths. The third one stores, for each segment, the address of the previous in the maximum cost path, along with the mode of the edge. We implemented a set of multipliers to compute the likelihood of a candidate interval for all the modes. The parallel structure helps to reduce the time required to terminate the whole procedure and simplifies the finite state machine that controls the process. Another multiplier is then required to extract the total cost of a path. The algorithm also requires some comparators to find the mode that maximizes the likelihood of each interval and the path the maximizes the total cost. Data is represented with 8 bits, except for the cumulative path probability that is stored at 64 bits. Off-line simulation has shown that choice provide enough resolution for this task.

As discussed in Sec. 3.2, the algorithm has complexity of $\mathcal{O}(N_{\rm segm})$. In our FPGA implementation, execution requires 42 cycles for each new segment. Suppose for example that 5 persons traverse the field of view within a period of $T_{\max}$ frames, and that each passage generates 5 segments. Then, if the processing unit is clocked at 10 MHz, the algorithm takes about 100 μs to generate all intervals.

The duty cycle of algorithm execution can be computed as follows. During each frame period, the memory interface, FIFO and processing unit is active for about 250 μs. This represents 0.8% of the frame period, assuming a frame rate of 30 fps. Hence, the duty cycle is of only 0.8%. This translates in very low power consumed by the FPGA, which can be put in reduced-power mode for 99.2% of the time.

The whole implementation occupies 5101 out of 13824 VersaTiles (36%). VersaTiles are Actel Igloo's basic elements which can be customized to generate the required function. The implementation also uses 48 out of 235 I/O lines (20.43%).

The development board allows us to monitor the current flowing through the FPGA core and its I/O banks, and the camera. The FPGA core needs a supply voltage of 1.2 V, while the I/O banks and the camera operate at 3.3 V. According to our measurements, these elements consume less than $9\mathrm{mW}$ altogether. This estimate is based on the assumption of a continuous flow of people through the camera's field of view. If the flow of people is intermittent (a more realistic hypothesis), the Idle mode of the camera can be exploited for reduced power consumption. We should also point out that the camera is mounted on a board together with other elements required for debugging, such as trimmers, which, while not strictly necessary, consume a significant amount of power. Removing these elements would reduce power consumption even further.

In our current prototype, the output data is sent to a host PC through the RS-232 interface. The next version of this system will have a board equipped with only the essential components. The RS-232 interface will be replaced with a wireless interface, based on the IEEE 802.11 or the IEEE 802.15.4 standards.

5. Experimental Evaluation

Two videos (Video 1 and Video 2) were acquired with the binary camera. Each video was hand-labeled, resulting in a list $\bar{\mathcal{S}} = (\bar{\mathbf{S}}^m_{\bar{k}(m)})$ of "ground truth" intervals (the m-th interval is labeled with mode $M_{\bar{k}(m)}$). In some instances, more than one person was seen at the same time in the camera's field of view. These sequences are labeled with a mode of type M_3. Note that our algorithm can only identify sequences of mode M_0, M_1 or M_2, so a situation with M_3 will produce an error. Video 1 contained 186 intervals, 23 of which of type M_3. Video 2 contained 193 intervals, 37 of which of type M_3. The two videos were taken in different locations with different light conditions. In the case of Video 1, persons transiting in the scene casted shadows that were well visible in the binary images. No shadows were noticeable in Video 2.

5.1. Method of Benchmarking

Our algorithm produces a list of intervals $\mathcal{S} = (\mathbf{S}^n_{k(n)})$, with the n-the interval of mode $M_{k(n)}$. In order to assess the quality of classification, one needs a way to compare the list of hand-labeled and automatically produced intervals. The approach we use is to find the set of unique ordered assignments between elements of $\bar{\mathcal{S}}$ and of $\mathcal{S}$ that maximizes a certain criterion. By "unique" we mean that one interval of $\bar{\mathcal{S}}$ can be assigned to at most one interval of $\mathcal{S}$, and vice-versa. By "ordered" we mean that if $\bar{\mathbf{S}}^{m_1}$ is assigned to $\mathbf{S}^{n_1}$ and $\bar{\mathbf{S}}^{m_2}$ is assigned to $\mathbf{S}^{n_2}$ with $m_2 > m_1$, then $n_2 > n_1$. The criterion to be maximized is the sum of the overlap (in frames) of the sequences in each assignment. More precisely, let the assignments be represented by the set of I pairs (m_i, n_i), and let O_i represent the overlap (in frames) between $\bar{\mathbf{S}}^{m_i}$ and $\mathbf{S}^{n_i}$. We seek the unique ordered assignments that maximize $\sum_{i=1}^{I} O_i$. This task is equivalent to the problem of *global sequence alignment*, a well-studied topic in bioinformatics. We use the Needleman-Wunsch algorithm [9], with scores equal to O_i and zero gap penalty. This algorithm solves global sequence alignment using dynamic programming. Once the sequences are aligned, we compute the rate of "correct detection" (matches $[\bar{\mathbf{S}}^m_{\bar{k}(m)}, \mathbf{S}^n_{k(n)}]$ with $\bar{k}(m) = k(n)$), "misses" (sequences of $\bar{\mathcal{S}}$ that do not have a match in $\mathcal{S}$) and "false alarms" (sequences of $\mathcal{S}$ that do not have a match in $\bar{\mathcal{S}}$).

5.2. Classification Performances

Experimental tests were conducted with one of the two videos used as training set (to learn the state conditional probabilities) and the other one for testing the algorithm. As mentioned earlier, the videos contained intervals with more than one person visible in the scene (M_3). These intervals cannot be correctly identified by our algorithm, which is only trained on modes M_1 and M_2. We presents two sets of results. The first set is computed after removing all intervals of type M_3 from the videos; the second set includes intervals of type M_3. The results of the first test are shown in Tab. 1 in terms of Error rate, Missed detection rate, and False alarm rate, as defined in Sec. 5.1. It is seen that, although the error rate is negligible, some missed detections and false alarms are reported. Upon visual analysis of the data, it was ascertained that occurrences of missed detection were mostly due to poor lighting condition. The rate of false alarms is negligible when the algorithm is trained on

Table 1. Classification performances (M_3 intervals removed)

Training set	Video 1	Video 2
Test set	Video 2	Video 1
Error rate	0%	0%
Missed detection rate	3%	5%
False alarm rate	0%	14%

Table 2. Classification performances (M_3 intervals included)

Training set	Video 1	Video 2
Test set	Video 2	Video 1
Error rate	15%	14%
Missed detection rate	8%	5%
False alarm rate	15%	7%

Video 1 and tested on Video 2, but fairly high (14%) when the training and test sets are reversed. The reason for this is that, as observed earlier, Video 1 contains several visible moving shadows. These shadows are sometimes incorrectly interpreted as additional people in the scene by the algorithm when trained on Video 2, which did not have moving shadows.

As expected, if intervals with mode M_3 are considered (Tab. 2), the error rate grows dramatically. The algorithm, when presented with an interval of mode M_3, typically assigns one or more intervals of type M_1 and M_2 to it, resulting in errors and possibly false alarms. This also explains why the rate of false alarms increases with respect to the previous test. Note that introducing intervals of type M_3 also increases the missed detection rate when the system is trained on Video 1.

6. Conclusions

We have presented an integrated camera-processor node that was configured to detect and count persons transiting in either direction through the camera's field of view. The power consumed by this system is extremely low (less than 10 mW) which allows for very long lifetime when battery-operated. Future work will address the integration of this system in a node that includes a ZigBee transceiver, which could be used to periodically communicate the results of the people counter.

The algorithm is based on statistical modeling of the dynamics of the appearance of a person as it transits through the scene. Perhaps its biggest drawback is that it cannot detect the presence of multiple persons in the scene. We plan to increase the number of modes in order to account for various modalities (such as two persons moving in one direction). This may require increasing the number of VILs to allow for better representation of the image content. In addition, we plan to consider the temporal length of each segment of frames as a feature to be used in the statistical model. All of these new features can be easily accommodated in the current Flash-FPGA implementation.

References

[1] P. W.-C. Chen, P. Ahammad, C. Boyer, S.-I. Huang, L. Lin, E. J. Lobaton, M. L. Meingast, S. Oh, S. Wang, P. Yan, A. Yang, C. Yeo, L.-C. Chang, D. Tygar, and S. S. Sastry. CITRIC: A low-bandwidth wireless camera network platform. Technical Report UCB/EECS-2008-50, EECS Department, University of California, Berkeley, May 2008. 2

[2] J. Costas-Santos, T. Serrano-Gotarredona, R. Serrano-Gotarredona, and B. Linares-Barranco. A spatial contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems. *IEEE Transactions on Circuits and Systems I*, 54(7):1444–1458, July 2007. 1

[3] M. Gottardi, N. Massari, and S. Jawed. A 100 μw 128x64 pixels contrast-based asynchronous binary vision sensor for sensor networks applications. *IEEE Journal of Solid-State Circuits*, 44(5), May 2009. 1, 2, 3

[4] K. Hashimoto, K. Morinaka, N. Yoshiik, C. Kawaguchi, and S. Matsueda. Peopple count system using multi-sensing applications. In *IEEE International Conference on Solid-State Sensors and Actuators*, 1997. 1

[5] S. Hengstler, D. Prashanth, S. Fong, and H. Aghajan. Mesheye: A hybrid-resolution smart camera mote for applications in distributed intelligent surveillance. In *IPSN '07*, pages 360–369. ACM Press, 2007. 2

[6] D. Lefloch. Real-time people counting system using video camera. Master's thesis, Universite de Bourgogne, 2007. 2

[7] C. Margi, R. Manduchi, and K. Obraczka. Energy consumption tradeoffs in visual sensor networks. In *24th Brazilian Symposium on Computer Networks (SBRC 2006)*, 2006. 1

[8] T. Murakita, T. Ikeda, and H. Ishiguro. Human trackingn using floor sensors based on the Markov chain Monte Carlo method. In *IEEE ICPR*, 2004. 1

[9] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970. 7

[10] M. Rahimi, R. Baer, O. I. Iroezi, J. C. Garcia, J. Warrior, D. Estrin, and M. Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *SenSys 2005)*, 2005. 2

[11] P.-F. Ruedi, P. Heim, F. Kaess, E. Grenet, F. Heitger, P.-Y. Burgi, S. Gyger, and P. Nussbaum. A 128 $\times$ 128 pixel 120-db dynamic-range vision-sensor chip for image contrast and orientation extraction. *IEEE Journal of Solid-State Circuits*, 38(12):2325–2333, December 2003. 1

[12] T. Teixeira, E. Culurciello, J. H. Park, D. Lymberopoulos, A. Barton-Sweeney, and A. Savvides. Address-event imagers for sensor networks: evaluation and modeling. In *IPSN '06*, pages 458–466, New York, NY, USA, 2006. ACM. 2

[13] T. Teixeira and A. Savvides. Lightweight people counting and localizing in indoor spaces using camera sensor nodes. In *ICDSC '07*, pages 36–43, 2007. 2

[14] E. Viarani. Extraction of traffic information from images at DEIS. In *ICIAP'99*, pages 1073–1076, 1999. 2, 3