Design guidelines for embedded real time face detection application

White paper for Embedded Vision Alliance

By Eldad Melamed

Much like the human visual system, embedded computer vision systems perform the same visual functions of analyzing and extracting information from video in a wide variety of products.

In embedded portable devices such as Smartphones, digital cameras, and camcorders, the elevated performance has to be delivered with limited size, cost, and power. Emerging high-volume embedded vision markets include automotive safety, surveillance, and gaming. Computer vision algorithms identify objects in a scene, and consequently produce one region of an image that has more importance than other regions of the image. For example, object and face detection can be used to enhance video conferencing experience, management of public security files, content based retrieval and many other aspects.

Cropping and resizing can be done to properly center the image on a face. In this paper we present an application that detects faces in digital image, crops the selected main face and resize it to a fixed size output image (see figure 1).

The application can be used on a single image or on a video stream, and is designed to run in real time. As far as real-time face detection on mobile devices is concerned, appropriate implementation steps need to be made in order to achieve a real-time throughput.

This paper presents such steps for real-time deployment of face detection application on programmable vector processor. The steps taken are general purpose in the sense that they can be used to implement similar computer vision algorithms on any mobile device.

2 Maskit Street, P.O. Box 2068, Herzelia, 46120, Israel
t: +972 9 961 3700 | f: +972 9 961 3800 | info@ceva-dsp.com | www.ceva-dsp.com

Figure 1 – CEVA face detection application

While still image processing consumes a small amount of bandwidth and allocated memory, video can be considerably demanding on today's memory systems.

At the other end of the spectrum, memory system design for computer vision algorithms can be extremely challenging because of the extra number of processing steps required to detect and classify objects. Consider a thumbnail with 19x19 pixels size of face pattern. There are 256^{361} possible combinations of gray values only for this tiny image, which impose extremely high dimensional space. Because of the complexities of face image, explicit description of the facial feature has certain difficulties; therefore other methods which are based on a statistical model have been developed. These methods consider human face region as one pattern, construct classifier by training a lot of "Face" and "non-face" samples, and then determine whether the images contains human face by analyzing the pattern of the detection region.

Other challenges that face detection algorithms must overcome are: pose (frontal, 45 degree, profile, upside down), presence or absence of structural components (beards, mustaches, glasses), facial expression, occlusion (faces may be partially occluded by other objects), image orientation (face appearance directly vary for different rotations about the camera's optical axis), and imaging conditions (lighting, camera characteristics, resolution).

2 Maskit Street, P.O. Box 2068, Herzelia, 46120, Israel
t: +972 9 961 3700 | f: +972 9 961 3800 | info@ceva-dsp.com | www.ceva-dsp.com

Although many face detection algorithms have been introduced in the literature, only a handful of them can meet the real-time constraints of mobile devices. While many face detection algorithms have been reported to generate high detection rates, very few of them are suitable for real-time deployment on mobile devices such as cell-phones due to the computation and memory limitations of these devices.

Normally, real-time implementations of face detection algorithms are done on PC platforms with relatively powerful CPUs and large memory sizes. The examination of the existing face detection products reveal that the algorithm introduced by Viola and Jones in 2001 has been widely adopted. This is a breakthrough work which allowed appearance-based methods to run in real-time, while keeping the same or improved accuracy.

The algorithm uses a boosted cascade of simple features, and can be divided to three main components: (1) Integral graph - efficient convolution for fast feature evaluation; (2) Use Adaboost for feature selection and sort them in the order of importance. Each feature can be used as a simple (weak) classifier; (3) Use Adaboost to learn the cascade classifier (ensemble of weak classifiers) that filters out the regions that most likely do not contain faces. Figure 2 is a schematic representation of the cascade of classifiers. Within an image, most sub images are non-face instances.

Based on this assumption we can use smaller and efficient classifiers to reject many negative examples at early stage while detecting almost all the positive instances. More complex classifiers are used at later stage to examine difficult cases.

Example: 24 stages cascade classifier
2 feature classifier in the first stage => rejecting 60% non-faces while detecting 100% faces
5 feature classifier in the second stage => rejecting 80% non-faces while detecting 100% faces
20 feature classifier in stages 3, 4, and 5
50 feature classifier in stages 6 and 7
100 feature classifier in stages 8 to 12
200 feature classifier in stage 13 to 24

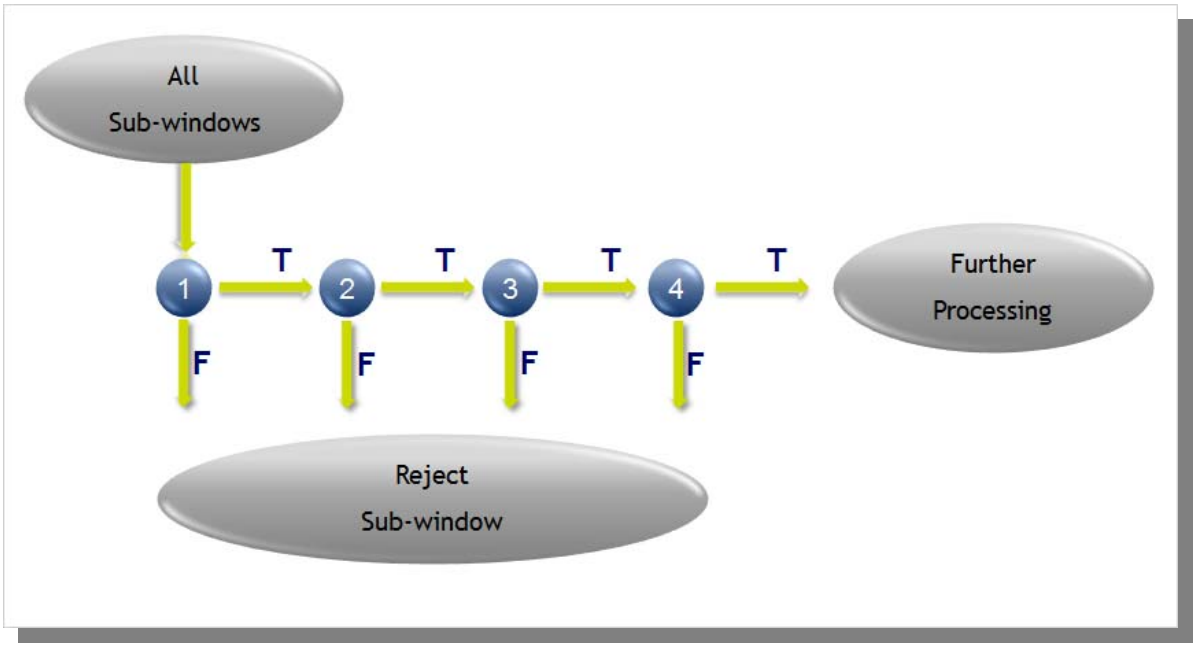

Figure 2 - The Cascade of Classifiers

During the first stage of the face detection algorithm, rectangle features can be computed very rapidly using an intermediate representation called integral image. As shown in figure 3 the value of the integral image at point (x,y) is the sum of all the pixels above and to the left. The sum of pixels within D can be computed as 4+1-(2+3).

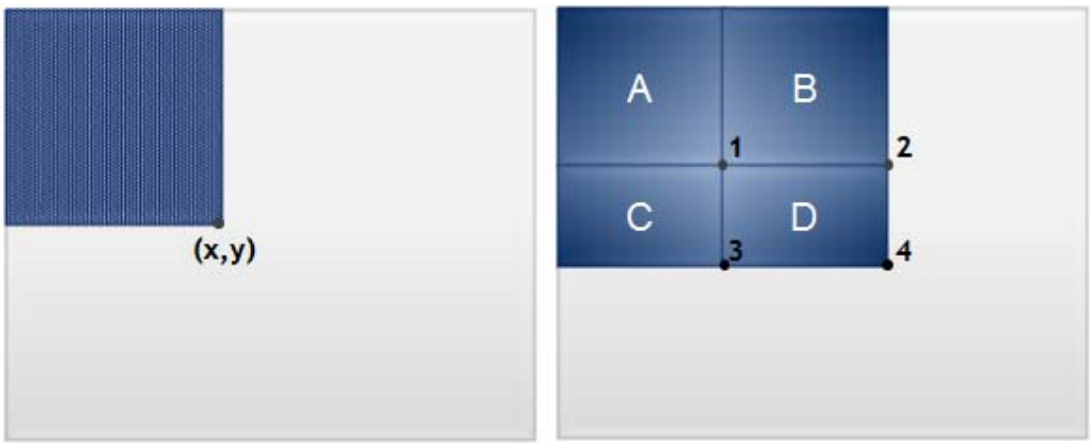

Figure 3 – Rapid evaluation of rectangular features by integral image

2 Maskit Street, P.O. Box 2068, Herzelia, 46120, Israel
t: +972 9 961 3700 | f: +972 9 961 3800 | info@ceva-dsp.com | www.ceva-dsp.com

To implement a real time face detection application on embedded device there is a need for a high-level of parallelism, combining instruction-level and data-level parallelism. Very Long Instruction Word (VLIW) architectures allow a high level of concurrent instruction processing, providing extended parallelism as well as low power consumption.

Single Instruction Multiple Data (SIMD) architectures enable single instructions to operate on multiple data elements resulting in code size reduction and increased performance. Using vector processor architecture accelerates these integral sum calculations by a factor of the parallel number of adders/subtractors. If a vector register can be loaded with 16 pixels, and these pixels can be added to the next vector simultaneously, the acceleration factor is 16. Evidently, adding similar vector processing unit to the processor doubles this factor.

During the next face detection stages, the image is scanned at multiple positions and scales. Adaboost strong classifier (which is based on rectangle features) is applied to decide whether the search window contains a face or not. Again, a vector processor has obvious advantage – the ability to simultaneously compare multiple positions to threshold.
Under the assumption that within an image, most sub images are non-face instances, more available parallel comparators mean faster acceleration.
For example, if the architecture is designed with the ability to compare 2 vectors of 8 elements each in 1 cycle, the rejection of 16 positions sub images will take only 1 cycle. To ease data loading, and to use the vector processor load/store unit efficiently, the positions can be spatially close one to another.

In order to obtain highly parallel code, the architecture should support instruction predication. This enables branches caused by if-then-else constructs to be replaced with sequential code, thus reducing cycle count and code size. Allowing conditional execution, with the ability to combine conditions, achieve a higher degree of efficiency in control code. Moreover, non-sequential code, such as branches and loops, can be designed with a zero cycle penalty without requiring cumbersome techniques such as dynamic branch prediction and speculative execution that drive up the power dissipation of RISC processors.

2 Maskit Street, P.O. Box 2068, Herzelia, 46120, Israel
t: +972 9 961 3700 | f: +972 9 961 3800 | info@ceva-dsp.com | www.ceva-dsp.com

One of the key challenges in the application is memory bandwidth. The application needs to scan each frame of the video stream to perform the face detection. A video stream cannot be stored at the tightly coupled memory (TCM), because of its large data size. For example, 1 high definition frame in a YUV 4:2:0 format consumes 3 Mbytes of data memory.
The high memory bandwidth causes higher power dissipation and involves more expensive DDR memory, contributing to a more costly bill of materials. An elegant solution is to store the pixels using data tiling, whereby 2-dimensional tiles are accessed from the DDR in a single burst, vastly improving the efficiency of the DDR. Direct memory access (DMA) can transfer data tiles between external memory and the core's memory subsystem. During the final face detection application stage, the sub image that contains the detected face is resized to a fixed size output window.

This process of image resizing is also used during the detection phases, when the image is scanned at multiple scales. Resizing algorithms are widely used in image processing for video up-scaling and down-scaling. The algorithm that is implemented in the face detection application is the bi-cubic algorithm.

Cubic convolution interpolation determines the gray level value from the weighted average of the 16 closest pixels to the specified input coordinates, and assigns that value to the output coordinates. First, four one-dimension cubic convolutions are performed in one direction (horizontally) and then one more one-dimension cubic convolution is performed in the perpendicular direction (vertically). This means that to implement a two-dimension cubic convolution, a one-dimension cubic convolution is all that is needed.

A vector processor core that has powerful load-store capabilities to quickly and efficiently access the data is a crucial feature for such applications, where algorithms operate on blocks of data. The resizing algorithm optimization can be satisfied by capability to access 2-dimensional blocks of memory from the memory in a single cycle.

This feature allows the processor to efficiently achieve high memory bandwidth without the need to load unnecessary data or burden computational units with performing data manipulations.

2 Maskit Street, P.O. Box 2068, Herzelia, 46120, Israel
t: +972 9 961 3700 | f: +972 9 961 3800 | info@ceva-dsp.com | www.ceva-dsp.com

Furthermore, a capability to transpose a block of data, during data access, without any cycle penalty, enabling a transposed block of data to be accessed in a single cycle, is extremely practical for the implementation of the horizontal and vertical filters. The horsepower of the processor is a result of its ability to perform powerful convolutions, allowing parallel filters to be performed in a single cycle.

An example for efficient solution is loading 4x8 block of bytes in one cycle, and then performing the cubic convolution in the vertical direction using 4 pixels for each iteration. The 4 pixels are pre ordered in 4 separate vector registers, so we can get 8 results simultaneously.
These intermediate results are then processed exactly the same, but with loading the data in transposed format, so the horizontal filter is done. In order to preserve results accuracy, initialization with a rounding value and post-shift of the result is needed. The filter configuration should enable these features without requiring a dedicated instruction.

Overall, this kind of parallel vector processing solution kernel can be balanced between the load/store unit operations and the processing units. Generally, the data bandwidth limitations and the cost of processing units in means of power consumption and die area restricts the implementation efficiency; yet it is clear that major acceleration over scalar processor architectures is achieved.

A Multipurpose, Programmable HD Video and Image Platform for Multimedia Devices

CEVA-MM3000 is a scalable, fully programmable multimedia platform that can be integrated into SoCs to deliver 1080p 60fps video decode and encode, ISP functions and vision applications, completely in software. The platform consists of two specialized processors, a Stream Processor and a Vector Processor, combined into a complete multi-core system, including local and shared memories, peripherals, DMA and standard bridges to external busses. This comprehensive multi-core platform was designed specifically to meet low-power requirements for mobile devices and other consumer electronics.

The Vector Processor includes two independent Vector Processing Units (VPUs). The VPUs are responsible for all vector computations. These consist of both inter-vector operations (using single instruction multiple data) and intra-vector operations. The inter-vector instructions can operate on sixteen 8-bit (byte) elements or eight 16-bit (word) elements, and can use pairs of vector registers to form 32-bit (double-word) elements. The VPU has the ability to complete eight parallel filters of six taps in a single cycle.

While the VPUs serve as the computational workhorse of the Vector Processor, the Vector Load and Store Unit (VLSU) serves as the vehicle for transferring data from/to the Data Memory Sub-System to/from the Vector Processor. The VLSU has a 256-bit bandwidth for both load and store operations, and supports non-aligned accesses. The VLSU is powered with the capability to access 2-dimensional blocks of data in a single cycle, supporting various block sizes.

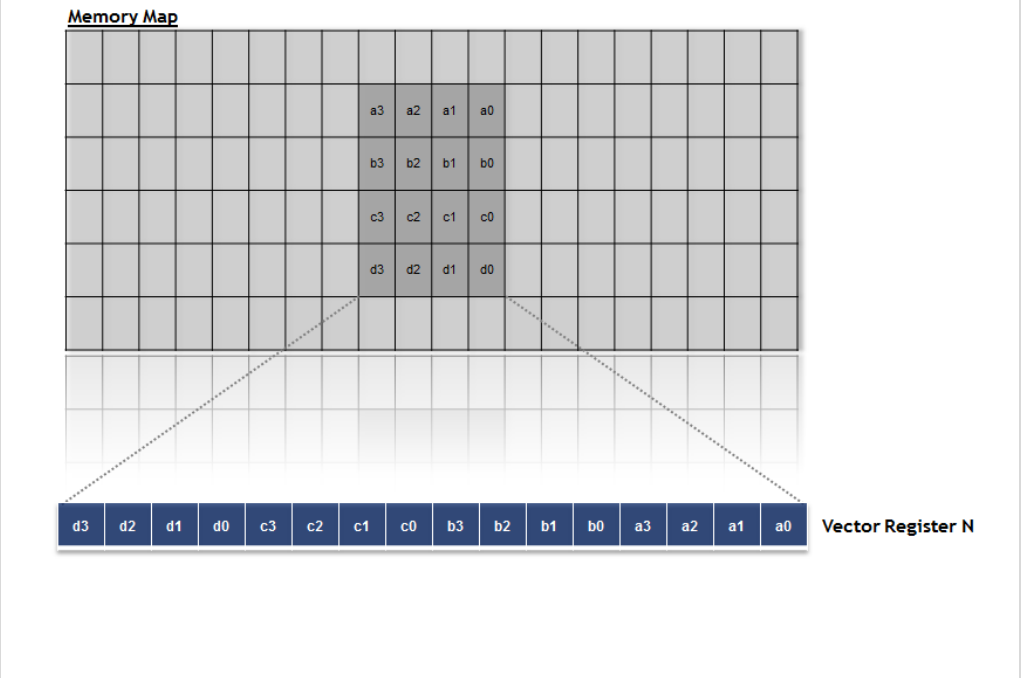

Figure 4 - loading a 4x4 block of pixels

2 Maskit Street, P.O. Box 2068, Herzelia, 46120, Israel
t: +972 9 961 3700 | f: +972 9 961 3800 | info@ceva-dsp.com | www.ceva-dsp.com

To ease the task of the VPUs, the VLSU can flexibly manipulate the structure of the data when reading/writing the vector registers. A block of data can be transposed during data access without any cycle penalty, enabling a transposed block of data to be accessed in a single cycle. The transpose function can be dynamically set or cleared. In this way, the same function can be re-used for both horizontal and vertical filters, saving development and debug time of each filter, while reducing the program memory footprint.

Conclusion

An embedded vision application like face detection with cropping and resizing can be one example from the diversity of algorithms that can be efficiently implemented for consumer devices with the CEVA-MM3000 platform. The future predicts growing demand for similar and more complicated applications; all can utilize the programmability and scalability of the CEVA-MM3000 architecture.

2 Maskit Street, P.O. Box 2068, Herzelia, 46120, Israel
t: +972 9 961 3700 | f: +972 9 961 3800 | info@ceva-dsp.com | www.ceva-dsp.com