**Enabling seamless video processing in smart surveillance cameras with multicore**

**Abstract—**

Smart video surveillance is an area of research focus in smart city technology. Smart camera design for this task needs to perform seamless video processing. Multicore is one solution to achieve high performance. In this paper, we propose a pipelined parallel architecture for smart video surveillance that is appropriate for implementation on a multicore environment. The architecture comprises of modules for video frame acquisition and image processing operations performed in sequence on an image frame. Successive lines of a frame are processed in a pipeline on the multicore. Embedded system realization on a multicore XMOS microcontroller runs the drivers for interfacing image sensor and LCD on different cores along with the various stages of the image processing pipeline. The realization achieves a frame rate of 8 frames/second for an image size of 480×272. Further, the solution is area-efficient without the need for a large external memory and is based on a single XMOS sliceKIT with support (in the form of compact slices) for camera, LCD and other units.

1. **INTRODUCTION**

Security is a main concern in the development of smart cities. Smart cameras need to be deployed for advanced video surveillance [1]. These embedded systems demand seamless video processing. The processing necessarily involves computationally-intensive operations such as motion tracking, face detection and activity recognition. In the past, speedup has been achieved in a processor by increasing clock speed. Multicore processors are the new direction semiconductor companies are focusing on to get a boost in the performance. A multicore system [2] consists of multiple conventional processors on a single chip, thus giving a homogenous processing platform. General purpose multicore processors tend to use shared memory architectures. Computer vision algorithms are characterized by fine grain of task granularity along with a regular dataflow stream. They are excellent candidates for multicore systems [3]. In addition, interfacing with various devices to build an embedded vision system necessitates efficient data acquisition and delivery. A multicore environment provides high degree of parallelism to accomplish these needs. In this paper, we present a speed and area efficient pipelined system for video processing. We demonstrate the feasibility of the system for surveillance applications like face detection and motion tracking. In face detection, considerable work has been done and several algorithms have been proposed [4]. Most of these algorithms are based on features such as Haar, skin pixels, histograms and shape. Extracting the desired information from voluminous video data is computationally intensive. In addition, the real-time processing requirement calls for a high-performance implementation of face detection. Embedded implementation of face detection is quite challenging. Some recent work on VLSI implementation of face detection are available [5], [6], [7], [8]. Of late, multicore processor has been given greater attention due to its ability to realize a parallel system with ease achieving faster time-tomarket when compared to FPGAs. Work on multicore realization of face detection is limited. A multicore architecture, based on ASIP, for Viola and Jones’s algorithm is proposed in [9]. The architecture was prototyped on Altera FPGA with an external memory. In another work [10], the same algorithm was realized on a multicore system with 64 ARM v5 processors and a shared memory. In these implementations, the memory requirement is high as it needs storing the entire video frame and the intermediate results.

1. **APPLICATIONS TO SMART VIDEO SURVEILLANCE**

We have demonstrated the feasibility of the proposed pipeline for the surveillance applications given below. A. Face Detection The proposed face detection method is based on a skin pixel detection algorithm. Every pixel is compared with the standard range of human skin colors to identify the pixels corresponding to faces, and the image is thresholded to get a binary image. We apply a skin color based detection method as described in [16] to identify the skin pixels, as this method allows fast processing and is extremely robust, while also being memory-efficient and fairly accurate. Connected component analysis is performed on the binary image. This gives the location of the face in terms of bounding box coordinates. This information can be utilized for several applications, such as biometrics and motor control in vision based robotics. We have used this method to isolate a face and display it on an LCD screen. The proposed pipelined face detection method consists of four stages, viz., 1) Video frame acquisition 2) Skin pixel detection 3) Binary morphology 4) Connected component analysis The first stage involves acquisition of the lines of the image from the camera, and converting it into an RGB image. As a result of the color filter array present on the image sensor, the data received from every pixel only corresponds to one color in Bayer Pattern [17]. This data is used to assign R,G,B values to every pixel by performing demosaicing techniques, as described in [18]. We have used the Pixel Double Bayer interpolation algorithm, where the values of the nearest neighbors are directly taken for the missing colors of every pixel. This method was implemented using two lines of data received from the image sensor. The second stage identifies the skin pixels in the image and performs thresholding to obtain a binary image. The thresholding is based on the following rules, as described in [16]: • R > 95 AND G > 40 AND B > 20 • max{R,G,B} − min{R,G,B} > 15 • |R − G| > 15 • R > G AND R > B The binary image giving the identified skin pixels is passed through morphological processing to remove the spurious skin pixels and also to close the holes in the skin pixel components. The morphological operation of closing is applied to the image, to achieve this. Closing consists of the binary dilation and erosion operations applied sequentially, as described below: A • B = (A ⊕ B) B where A is the binary image, B is the structuring element, • represents the closing operation, and ⊕ and represent the dilation and erosion operations respectively.