ADAPTIVE IMAGE SENSOR SAMPLING FOR LIMITED MEMORY MOTION DETECTION

David Gibson[1], Henk Muller[2] and Neill Campbell[1]
[1]*Computer Science, University of Bristol, Bristol, U.K.*
[2]*XMOS Ltd, Bristol, U.K.*

Keywords: Motion Detection, Programmable Sensors, Low Memory.

Abstract: In this paper we propose that the combination of a state-of-the-art high frequency, low energy demanding microprocessor architecture combined with a highly programmable image sensor can offer a substantial reduction in cost and energy requirement when carrying out low-level visual event detection and object tracking. The XMOS microprocessor consists of a single or multi-core concurrent architecture that runs at between 400 and 1600 MIPS with 64KB per-core of on chip RAM. Modern highly programmable image sensors such as the Kodak KAC-401 can capture regions-of-interest (ROI) at rates in excess of 1500fps. To compare the difference between two 320 by 240 pixel images one would usually require 150KB of RAM, by combining the above components as a computational camera this constraint can be overcome. In the proposed system the microprocessor programs the sensor to capture images as a sequence of high frame rate regions-of-interest. These regions can be processed to determine the presence of motion as differences of ROIs over time. By providing additional cores extensive image processing can be carried out and ROI pixels can be composited onto an LCD to give output images of 320 by 240 pixels at near standard frame rates.

1 INTRODUCTION

There is increasing interest in low cost computer vision systems with a wide range of applications including gesture based user interfaces, surveillance, automotive systems and robotics. As the complexity of consumer, sensing and military systems increase the demands on energy resources becomes critical for high-level computing performance. Vision systems are proving to be extremely valuable across a range of applications and to be able to efficiently process visual information offers a huge advantage in the functionality of such systems.

Traditional computer vision systems typically consist of a camera continually capturing and transmitting images at a fixed frame rate and resolution with a host computer sequentially processing them to obtain a result such as the trajectory of a moving object. A major drawback of this pipeline is that large amounts of memory are required to store the image data before it is processed, especially as frame rate and image resolution increase. Additionally large amounts of the image data is transmitted to the host for processing regardless of the amount of information contained in this data. In the case of object tracking computer vision algorithms work towards creating a concise description such as a group of pixels at a certain location is moving in a particular way. Often the object is relatively small compared to the whole image and the background maybe static. In cases like this the traditional computer vision processing pipeline could be considered as being highly inefficient as large amounts of image data are being captured, transmitted to the host, stored in memory and being processed on a per-pixel basis while most of the visual information comes from a small number of changing pixels. In such cases most of the image data is discarded as it contains no useful information.

In the case of a scene with an object moving across a static background most of the image data changes very little while some pixel areas might change rapidly or move a different speeds. The fixed temporal sampling rate of standard camera systems cannot take this into account and artifacts such as motion blur and temporal incoherence are introduced. These artifacts consequently confound down stream processing necessitating ever more complex computer vision algorithms to overcome these imaging effects.

In (Shraml, S. and Belbachir, A. N., 2010) a so-called Dynamic Vision System (DVS) (Lichtsteiner,

P. and Posch, C. and Delbruck, T., 2007) is used to overcome the issues discussed above. The DVS has very low latency and only delivers pixel difference information when pixel luminance values change. While this is an exciting new technology it has two major drawbacks; it is low resolution (only 128 by 128 pixels) and only image differences are captured so a traditional image cannot be created.

In this paper we propose a system that conceptually fits between the DVS of above and a traditional computer vision capture system. By scanning regions of interest on the surface of the sensor at high sampling rates, piece-wise image processing can be carried out in very small amounts of memory. The first level of processing involves temporal ROI differencing to determine whether any activity has occurred in the current ROI. This temporal ROI differencing can be used to provide a powerful technique for controlling the amount and rate at which data is captured by the sensor, the amount and extent of further image processing that is performed and whether pixel data should be transmitted, stored or displayed. The potential of the low-level ROI differencing is demonstrated by two prototype hardware systems built to investigate the combination of XMOSs' high frequency microprocessor and a highly programmable image sensor.

2 SYSTEM ARCHITECTURE

The XMOS[1] XCore is a multi-threaded processing component with instruction set support for communication, I/O and timing. Thread execution is deterministic and the time taken to execute a sequence of instructions can be accurately predicted. This makes it possible for software executing on an XCore to perform many functions normally performed by hardware, especially DSP and I/O. An initial prototype was constructed to test the feasibility of the system described above, Figure 1. As a minimal system it consisted of a single core XMOS XS1-L1 processor running at 400 MIPS with four fast threads directly connected to a Kodak KAC-401 WVGA image sensor[2]. The XMOS architecture allows for four fast threads per core and while more threads are available they will share the system resources between them. In this application the high clock speed of the image sensor dictated that only fast threads could be used. The Kodak KAC-401 image sensor is highly

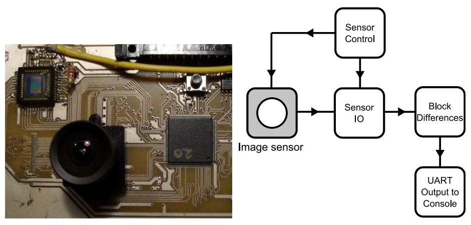

Figure 1: The initial prototype (left) consisting of a single core XMOS XS1-L1 and a Kodak KAC-401 image sensor. The sensor is directly connected to the processor, a M12 lens is shown to give a reference to the scale of the components and would normally cover the sensor. A four-threaded architecture (right) with the thread connectivity is described. Data is received by the XCore via bi-directional hardware ports and communication between threads is carried out via bi-directional channels.

programmable and provides features such as pixel binning (image sub-sampling), region-of-interest positioning, frame and row delays, digital and analogue gain adjustment, variable bit-depth, etc. The default clock speed is 25MHz and the sensor registers are updated using the I2C protocol. To program the sensor intermediate registers are written to and an update register is set, an update is triggered which then takes a single frame cycle to write from the intermediate registers to the main register set.

Throughout this paper the pixel depth was set to 8 bits and ROI were set to 64 by 40 pixels and programmed as a 5 by 6 ROI grid to cover the exposed sensor surface. The sensor resolution was set to 640 by 480 pixels and 2 by 2 pixel binning was used to give an effective image resolution of 320 by 240 pixels. The XMOS processors were programmed via a JTag connection to a host PC, the JTag connection includes a 10Kbs UART which was used in the initial prototype.

The initial prototype consisted of one thread to read from the sensor two consecutive ROI at a given grid location, each 64 by 40 pixels and consisting of 2560 bytes. This data is passed to a second, image processing, thread which compares the buffered pixels values of the ROI pair. As there were not enough resources for displaying results on an LCD, the results of the ROI comparison are transmitted as a bit pattern via a third thread running a UART to the host side console via the JTag connection. In a synchronized and concurrent manner the fourth thread programs and updates the sensor to capture the next ROI of the grid. As will be discussed in the results section, the output of the initial prototype responded as expected to basic stimuli such as passing a hand over and above the sensor, this lead to the development of a phase two prototype consisting of a multi-core architecture and LCD.

[1]www.xmos.com

[2]Technical datasheet obtained in 2010, "MTD-PS-1170 KAC-00401_Revision 1.0 MTDPS-1070.pdf", no longer available from Kodak.com.

In order to further explore the concepts described above a second prototype was built. This system comprised of the same image sensor but with a number of quad-core processors, ethernet and a LCD display. A schematic of this more extensive system is shown in Figure 2.

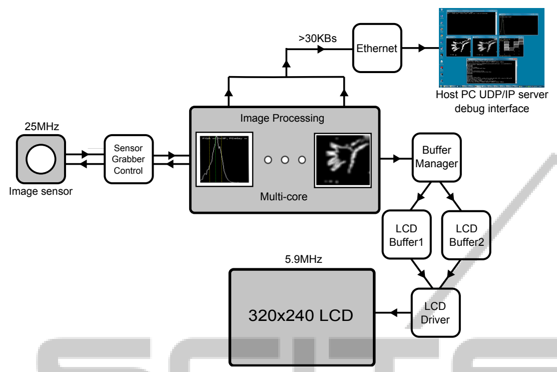

Figure 2: The system architecture of the second phase prototype, roughly from left to right; Pixels are read in from the sensor as a series of sub-images and passed onto the image processing sub-system. The results of the image processing sub-system are used to re-program the frame delay, gain and sub-image position and size registers of the image sensor. Image processing state spaces are transmitted to an ethernet delegate and the raw pixel data is passed through, via the frame buffering sub-system, to the LCD. The internal states of the image processing sub-system are visualized in their various forms using a windows based interface.

A detailed representation of the multi-core image processing sub-system is shown in Figure 3. The system proceeds as follows; the sensor is programmed to grab a ROI, data is read from the grabbing thread into the first image handling thread, while, concurrently, image data is passed on to a histogram analysing thread. In this thread a histogram is created for every other whole image and it's mean is compared to a pre-defined ideal mean value. A proportional integral derivative (PID) control is used to adjust the frame delay and analogue gain so as to move the current mean towards the ideal, the updated values are sent back to the grabber thread where they are used to update the sensor registers. Meanwhile, the image handler passes the ROI pixel data onto the a set of parallel threads to compute the integral images from each ROI image. The first image handler also requests the next ROI from the sensor. In the current implementation the integral ROI images are just used to create an 8 by 8 mean filtered representation of each 64 by 40 pixel ROI. Each 8 by 8 mean ROI representation is added to a 40 by 48 array to give low resolution representation of the high resolution 5 by 6 sensor grid sampling. The 40 by 48 array is considered as a set of observations which is compared to the previous frame of observations, the difference of which gives a motion detection map. The motion detection map is smoothed by a 3 by 3 gaussian filter each frame to generate a motion-history-image style representation (Davis and Bobick, 1997). The image difference and gaussian filter parts of the architecture in Figure 3 allow for results of statistical analysis to be feed back to the first image handler and sensor via the second image handler. The second image handler also channels the original pixel data on to the LCD output system. The results of parts of the image processing architecture are concurrently passed on to an ethernet delegate which transmits data to a server running on the host PC system. The later has proved invaluable for debugging, algorithm prototyping and visualisation of the internal states of the image processing system.

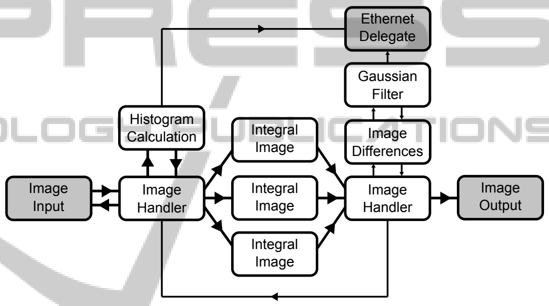

Figure 3: The image processing sub-system with parallel and concurrent thread usage which is distributed across a single quad-core XMOS processors.

3 RESULTS

In Figures 2 and 3 each small block effectively represents a single thread of the system. Consequently all of the image capture, sensor programming, image processing and LCD management can fit on a single 16 thread quad-core processor running at 1600 MIPS. The ethernet connection requires two more threads. The initial prototype generated a bit pattern where each bit represented a ROI and was set for motion detected and off for no motion. The system output behaved as expected when passing a hand over the sensor. However, using the ethernet and LCD output of the second prototype gave a much greater insight to the performance and functionality of the system. Table 1 shows timings of the system. The histogram processing is used to attempt to obtain a reasonable intensity balance for image pixels in varying light conditions. The system works in two modes; good light and bad light modes and the frame delay is allowed to change within a fixed range to best adapt to these conditions as reflected in the F. delay values of Table 1.

Table 1: Timing information of the multi-core system. F. delay is the frame delay used to increase exposure time. Sen. ROI is the rate at which ROI frames are captured by the sensor per second. Sys. ROI is the rate at which ROI images actually pass through the entire system per second. LCD and PC are the rate, in frames per second, at which 320 by 240 pixel images and 40 by 48 motion representations reach the LCD and PC (via ethernet) respectfully.

F.delay (μs)	Sen. ROI	Sys. ROI	LCD	PC
285	436	305	10.2	8.5
162	1442	489	16.3	15.3
113	1700	643	21.3	18.5

The Sensor ROI and System ROI values of Table 1 show the potential rate of ROI sensor capture and actual rate of ROI processing for the system. Clearly the system cannot process image data at the sensor ROI grabbing rate leaving the sensor to free run until the system is ready to grab the next ROI. ROI transfer times from the sensor to the system are 46μs and sensor ROI programming times are 26μs. The LCD values show the 320 by 240 pixel frames per second rate of output to the LCD and the PC values show the rate at which 40 by 48 motion representations are received by the host PC via the ethernet connection. The gaussian filtering is computed in a naive manner and by using the algorithms of (Wells, W. M., 1986) system performance and functionality could be increased. It should be noted that there has been no explicit optimisation applied to the system software which is written in the XMOS XC language, an extension of C. The above timings are given as an initial report of results and much more analysis is required to fully understand the true performance of the system.

4 CONCLUSIONS

In this paper we have shown that by leveraging the programmability of an image sensor, motion detection can be carried out at near standard frame rates at an effective resolution of 320 by 240 pixels using a single-core four thread processor with just 64KBs of RAM. Further we have shown that by using a multi-core architecture motion detection and various additional image processing can be carried out at near real time rates at an effective resolution of 320 by 240 pixels using a distributed system with no more than four unshared blocks of 64KB of RAM. It is expected that with further development the proposed system will be able to compute higher-level computer vision algorithms such as optical flow (Barron, J. L. and Fleet, D. J. and Beauchemin, S., 1994), point tracking (Shi, J. and Tomasi, C., 1994), gesture recognition (Shotton, J. and Fitzgibbon, A. and Cook, M. and Sharp, T. and Finocchio, M. and More, R. and Kipman, A. and Blake, A., 2011) and face detection (Viola, P. and Jones, M. J. and Snow, D., 2005). Key contributions of this paper include leveraging the programmability of modern image sensors and the use of high frequency low power XMOS processors.

ACKNOWLEDGEMENTS

This work was sponsored by an EPSRC Knowledge Transfer Secondment help by the Research, Enterprise and Development department of the University of Bristol.

REFERENCES

Barron, J. L. and Fleet, D. J. and Beauchemin, S. (1994). Performance of optical flow techniques. In *International Journal of Computer Vision*, volume 12, pages 43–77.

Davis, J. and Bobick, A. (1997). The representation and recognition of action using temporal templates. In *International Conference on Computer Vision and Pattern Recognition*.

Lichtsteiner, P. and Posch, C. and Delbruck, T. (2007). An 128x128 120db 15us-latency temporal contrast vision sensor. In *IEEE Journal Solid State Circuits*.

Shi, J. and Tomasi, C. (1994). Good features to track. In *International Conference on Computer Vision and Pattern Recognition*.

Shotton, J. and Fitzgibbon, A. and Cook, M. and Sharp, T. and Finocchio, M. and More, R. and Kipman, A. and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *International Conference on Computer Vision and Pattern Recognition*.

Shraml, S. and Belbachir, A. N. (2010). A spatio-temporal clustering method using real-time motion analysis on event-based 3d vision. In *International Conference on Computer Vision and Pattern Recognition*.

Viola, P. and Jones, M. J. and Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. In *International Journal of Computer Vision*, volume 63, pages 153–161.

Wells, W. M. (1986). Efficient synthesis of gaussian filters by cascaded uniform filters. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 8, pages 234–239.