

# Real-time Positioning for Augmented Reality on a Custom Parallel Machine

D. J. Johnston, M. Fleury, A. C. Downton, and A. F. Clark

Department of Electronic Systems Engineering

University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, U.K.

tel: +44 - 1206 - 872817

fax: +44 - 1206 - 872900

e-mail fleum@essex.ac.uk

## Abstract

Augmented Reality(AR) is frequently implemented using vision processing for target recognition, but performance that is simultaneously robust and real-time is still elusive for larger frame sizes. The processing requirements of a particular AR system developed at the University of Essex (the Video Positioning System or VPS) were analysed. It has been found that the critical region-based processing steps could be parallelised, despite the resulting complex accumulation of intermediate results. The paper presents the parallel algorithms involved and the performance achieved. Comparison is made with more traditional edge-based systems, which may execute somewhat faster but are not as robust. The success of the parallelisation overcomes this performance limitation, and suggests a future production route.

**Keywords:** Augmented reality, parallel computing, position finding.

## 1 Introduction

In [1], Augmented Reality (AR) is defined as: 1) combining real and virtual 2) being interactive in real-time 3) involving registration in 3-D. Augmented environments are a mobile form of Graphical User Interface (GUI) divorced from the desktop computer. Unlike Virtual Reality (VR), the user is not immersed in an artificial environment, but in the real world with objects or information superimposed. Because the immersion is not complete, current systems need not insist on photorealism and, subject to the constraint of a completed display, may run largely unhindered by rendering overhead. On the other hand, for some less common AR applications rendering latency is less of an issue than that of quality but completion of positioning registration is still vital, and robust positioning remains an essential requirement.

This paper examines how real-time indoor positioning performance can be achieved on a custom parallel computer for systems requiring accurate 6D location ( *i.e.* position and angular orientation which we abbreviate as “position”). The aim is to produce a generalised AR system in the sense that accurate estimation of a mobile user’s position relative to a set of indoor targets is accomplished in real time. For reasons of cost, a passive, vision-based system called the Video Positioning System (or VPS) has been developed. For real-time performance, speed-up of the positioning calculations is crucial, as even high-end PCs are not able to achieve the desired performance for full-PAL frames. Even the current *de facto* AR software, called the ARToolkit, would process full PAL frames at 7.2 fps on the 933 MHz PC used in a comparative trial [2]. However, developments in hardware for embedded systems make it likely that an alternative platform will become available.

In our investigations, the serial version the VPS positioning algorithm was transferred onto a custom parallel machine with four PowerPC microprocessors acting as compute engines, forming a coprocessor to a PC host. It turns out that the compute intensive part of the positioning algorithm, pre-processing of the image, can be readily decomposed through geometric parallelism (each node working on an equal part of an image) and so load-balancing is intrinsic. The overall time complexity of the target detection algorithm is independent of the number of targets in the scene, the number of targets being searched for, and their spatial arrangement - unlike the other systems available [2]. The paper argues that the extra computational load of the VPS compared to similar systems, is more than worthwhile given the high levels of robustness provided, which are directly derived from the nature of the algorithms used.

The work described in this paper was carried out as part of a larger project to establish a general methodology for developing high-bandwidth, real-time embedded applications that run on parallel hardware. While leading edge uniprocessors are approaching real-time performance for full-sized PAL frames, the use of multiple processors behind the leading edge (which are by definition the resources found most commonly *in situ*) allows the effectiveness of parallelisation to be assessed. Any parallel development methodology must be capable of delivering provably effective speed-ups; to fail to achieve this would be to undermine the whole reason for parallelism in the first place. The requirements of Augmented Reality presented an ideal test application with a usefully practical goal of real-time performance.

The rest of this paper is organised as follows. Section 2 describes the performance of AR systems of the type which form the focus of this paper. The VPS is addressed from the same external user's point of view. Section 4 outlines the main hardware and software features of the custom parallel computer that has been developed for data-dominated embedded applications such as the VPS. Section 5 is the insider's view of the VPS with an analysis of the processing steps. Then, Section 6 presents the performance of AR processing before and after parallelisation. Finally, Section 7 draws some conclusions.

## 2 Background

### 2.1 Real-time AR

AR relies on registration of virtual objects within the user's real-world view. As the virtual- and real-view are synchronised, AR systems should operate in real-time to keep pace with a mobile user, or more narrowly the motion of a user's head-mounted display. Real-time performance depends on: latency in rendering virtual objects, within 20/16.67 ms for PAL/NTSC video; and latency in finding the position of the real-world scene, for some AR tasks within a few millimetres.

As commodity graphics cards have a three-stage pipeline to cope with the compute-intensive operations in the per-pixel final stage, a frame normally suffers a three-fold frame lag through a graphics system.<sup>1</sup> Reducing the latency to one frame requires custom hardware and software [3]. Various

---

<sup>1</sup>One technique is to combine the virtual overlay frame with the real world by means

techniques have been used to offset rendering latency, including: delaying real-world video, for all-video schemes; warping or deflecting previously rendered images; and predicting over the short term the change in scene/user gaze [4].

In AR, wire-frame outlines, text labels, and pointers can be added to the view as an alternative to virtual objects. If additional information is to be displayed, for example, giving expert advice to a field worker [5], then positioning may not be as important as identification of the context. Identification of context is important where the local orientation must be linked to world coordinates. There are also a number of AR applications which do not involve much mobility, for example: direction of workers wiring an aircraft [6]. In fact, if the application is closely defined, compromises can be made in regard to positioning, identification, and update delay.

However, if the intention is in some way to deceive the eye of the viewer, as in a virtual TV studio [7], then precise positioning is vital. An AR system that can be applied to multiple applications, will need to perform well in positioning, identification, and update delay, as well as include mobility. This goal will only be possible in room-like environments, indeed one well-known application, adding virtual furniture to a room for interior design [8], is precisely that. The aim of the current work [9] is to create a generalised AR system.

## 2.2 AR positioning methods

The position of the user within an indoor environment <sup>2</sup> can be established by active or passive targets. Targets have the ability to provide identification as well as positioning, whereas vision-based tracking systems, such as through optical flow [11] or through working in the frequency domain [5, 12], do not.

Active indoor sensing systems, such as through ultrasonic range finders (to 3 cm accuracy) [13], are being constructed to form a sentient environment providing a centrally consistent scene, whereby the user's environment becomes the interface to the computer. A network of ceiling-mounted, ultra-

---

of hardware chroma-keying at the output of the display.

<sup>2</sup>Precise positioning within most outdoor scenes is largely established to 2-3 metres accuracy by means of the satellite-based Differential Global Positioning System (DGPS) [10], except in urban canyons.

sonic receivers is required, along with a fixed network to deliver identifying radio signals to any target. Other tagging technologies exist: active badges, based on infra-red line-of-sight signals [14]; embedded radio-frequency (RF) tags [15]; and magnetic resonance tags [16]; but these appear more suitable for tracking than positioning.<sup>3</sup> Active systems have the advantage, in some cases, that they can be hidden from the observer, making social integration easier.<sup>4</sup>

However, active positioning systems are not ubiquitous in the sense that tags cannot combine low cost with accuracy of measurement. If the AR interface is a wearable or portable device, ‘untethered AR’, then a decentralised solution is desirable. This is why passive, vision-based, systems are the subject of this paper. For more information on AR positioning methods the reader is referred to Chapter 3 of “Fundamentals of Wearable Computers and Augmented Reality” [17].

### 2.3 Vision-based positioning and identification

A passive computer vision system can permit precise measurements, and can be designed so that targets can be run off on an ordinary printer, although infra-red LEDs have also been used. Its main weakness is the need for line-of-sight views of the targets. To achieve a passive system, targets are placed within a room or corridor, as an extension of camera calibration techniques [18]. If a vision-based system is employed without controlled lighting, then the computation requirements are significant, presently requiring either hardware pre-processing or parallel processing to achieve real-time (video-rate) updates.

In vision systems both with assisted target recognition [7] (for a virtual studio) from the BBC (British Broadcasting Corporation), and without props [19] from Sony, hardware pre-processing is proposed.

---

<sup>3</sup>Another requirement is that the range should be limited to a micro-cell, conveniently formed by a room’s walls, so that no more than one receiver can be activated. Range is found by timing pulses, as a single signal strength measurement may well be affected by multi-path reflections in an indoor environment.

<sup>4</sup>Confidentiality of tracking information remains an issue.

### 2.3.1 UNC AR System

In the University of North Carolina's (UNC) pioneering AR system [4], an array of ceiling-mounted infra-red LED's of known location and geometry is tracked by four head-mounted optical sensors. The system was resilient against head-motion through correction by inertial sensors, and predicted dynamic head motion through Kalman filtering. This system used the Pixel-Planes 5 parallel graphics engine [20] for rendering the display, and a 50 MHz i860 4-way vector processor to perform positioning calculations.

### 2.3.2 ARtoolkit

The University of Washington have produced an 'ARtoolkit' with generic target detection software [21]. Targets are placed in pre-determined and measured locations, requiring a survey, though using a planar surface with gridded locations significantly ameliorates the problem. The target design is a simple dense black square border. Each is distinguished by a different bitmap image within the inner square. Typical images are kanji characters or letters of the alphabet, and these are used to provide orientation information for the target as well as identification. This software is available for free down-loading<sup>5</sup>, and is in widespread use in AR systems.

A number of authors [22, 23] report the ARToolkit getting confused between fiducials with similar patterns within the surrounding black square. To remedy this problem, [23] suggests a systematic way of producing fiducial images based on the Discrete Cosine Transform (DCT) basis functions instead of using an arbitrary set of bitmap images. Another benefit of this solution is that only a single DCT needs to be performed to find the most likely target, instead of a series of correlation coefficients having to be calculated - one for each known image to be tested against each unknown fiducial located in the scene. This method produces in the order of 200 distinguishable fiducials but sensitivity to errors are reported in the higher frequency DCT function designs. The ATK reports targets with a confidence measure for marker decoding, but [2] suggests this value may not be quite reliable. [2, 49] raise some concern over the positional accuracy of the ARToolkit in locating image points, though [2] concludes that the ARToolkit performs well across the board in particular being easy to use, and this no doubt explains its widespread adoption.

---

<sup>5</sup>[http://www.hitl.washington.edu/research/shared\\_space/download/](http://www.hitl.washington.edu/research/shared_space/download/) extant 17/07/03

A typical processing time for a frame of  $320 \times 240$  pixels with 10 targets [2] on a 933 MHz Pentium III is 24 ms. This varies with the number of targets visible and their size within the scene.

### 2.3.3 Sony Cybercode and other barcodes

The CyberCode tagging system [24] employs easily produced 2D barcodes, with positioning based on four feature points located by guide bars on a single target. Targets are intended to be read with a low-resolution ( $< 100$  K pixels) CCD camera, equipped with a lens of normal focal length, as available on some laptop computers such as the Sony VAIO-C1, the Sharp Zourous, or the Epson Locatio, giving 24- or 48-bit identity recognition in normal lighting conditions. The intended applications include: a pointing device which can display information on identified objects, InfoPoint; and a device that can annotate a route through a building, navicam. Therefore, the Sony system can be characterised as a generalised AR system.

The Sony system timings are cited as 15-30 Hz screen update on an SGI 02 (MIPS-R10000 at 175 MHz). On a mobile PC with attached camera, namely the Sony VAIO-C1 with a 200 MHz Mobile Pentium with MMX, the rate is 15 frame/s [24], though presumably on a small image size, which might affect target searching. It is suggested in [24] that the connected components algorithm on the Sony system could be accelerated by hardware.

These performance figures do not account for differences in the algorithms, accuracy, and functionality, or indeed what optimisations have been applied to the code.<sup>6</sup> There are also significant differences in I/O bandwidth, for example transferring video images over a PCMCIA (Personal Computer Memory Card International Association) interface on mobile devices [24]. Additionally, the desired frame rate and frame size can vary depending on the intended application. Completely flicker-free display may require an 80 Hz frame refresh rate, whereas rates as low as 12 Hz may be sufficient for some systems. Progressive capture, suitable for target searching, can occur at 30 Hz on a CCD camera, with interlacing artificially introduced by removal of alternate lines if broadcasting is an aim.

Similar systems based on square targets, with barcode-type encodings are the Hoffman marker system (HOM [25]) developed at Siemens AG in 1994 but recently turned into a software library; the IGD marker system

---

<sup>6</sup>Note, however, that the structure of the complete positioning algorithm in many systems [7, 11, 24] appears broadly equivalent (see Section 5.2).

[26], developed by IGD and available to participants in the ARVIKA project; the SGR [25] marker system developed at Siemens Corporate Research. [23] reports that barcode-style targets require a high resolution camera to operate reliably.

The comparative survey of passive visual targets [2] restricts practical evaluation to 4 systems based on square targets (*i.e.* ATK, HOM, IGD, SCR). These are evaluated under generous illumination conditions (against a plain background, closed curtains, all lights switched on) so this dimension of robustness is not tested. However, geometric robustness is investigated (targets under perspective distortion, at a distance and with a badly focused camera). Recognition in the HOM system is given a confidence value between 0 and 6, with  $\leq 2$  suggesting the marker has been detected with high confidence. The false positive rate of IGD is not discussed perhaps because under evaluation it could not be coaxed into finding more than one target per scene. SCR regards a target as being recognised or not. The processing time for a  $320 \times 240$  pixel image with 10 targets in the scene [2] are 35.3 ms for HOM and 21.9 ms with SCR, not significantly different from that of the ARToolkit.

#### 2.3.4 BBC Virtual Studio

The concept of the BBC's virtual-studio camera positioning system originates in [4], and indeed includes AR capability via live computer graphic rendering in the cameraman's console. Positioning within a large studio of a freely roaming, hand-held camera to an accuracy of  $\pm 1\text{mm}$  is required in order to register the camera's view with a virtual background. An additional, ceiling-pointing camera locates circular 9-bit bar-code targets rather than point source infra-red LEDs. Notice that to increase the number of bits of a circular target requires increasing the resolution of the imaging system. To achieve real-time processing (at TV output frame rate), special-purpose hardware is required to perform filtering and thresholding, as well as a digital signal processor to locate the targets in the shot, identify the bar codes, and calculate the camera position in 3D. In addition, a high-contrast image is produced by manufacturing the targets from a retro-reflective material (coated with miniscule glass beads) which reflects light from LEDs mounted around the camera lens.

The BBC system runs at twenty camera measurements/s on an SGI O2, whereas it can achieve a progressive scan rate of 50/60 frames/s but only with hardware assisted target detection and DSP (Digital Signal Processor)

assisted position location. However, initial target detection takes a few seconds to locate a target within an image. Thereafter, the search region (and search time) is reduced from frame-to-frame, due to the known confines of camera motion. An earlier in-shot version of the BBC system works with multiple markers on one target. Four or more markers allow location of a target, without the need to find multiple potential targets, which anyway may not appear within the same image. The BBC system is a commercial robust system already in active use.

### **2.3.5 TRIP**

The TRIP vision system [27], initially developed at the Olivetti Research Laboratory in Cambridge, for identification and location of a target also employs bi-level targets based on concentric circles, whereby sectors of each ring are black. A white annular band between each sectorised ring synchronises read-off of the target code. Code detection is via edge detection. The targets can be produced by laser printer and, therefore, are low-cost and versatile. However, the form of image processing is different and the suggested applications are more to do with location of people and objects than of augmented reality.

### **2.3.6 VPS**

The Video Positioning System (VPS) was developed at the University of Essex, UK to determine camera location for indoors AR. The targets [9] are designed to have highly distinctive patterns when the scene in which they occur is turned into a region representation called a Region Adjacency Graph (RAG). Each target has a common “key pattern” (used to find the targets) combined with a unique “identifier pattern” (used to distinguish the targets). The RAG representation of the key is detected within the RAG representation of the scene, by standard graph searching.

Forbidding sub-centimetre detail produces 449 resolveably distinct targets which can fit on an A4 sheet of paper: considerably more than the rationalised ARToolkit scheme [23] but of the same order of magnitude. The VPS targets are much more even in response than those of the rationalised ARToolkit scheme which are sensitive to errors in the higher frequency DCT function designs. If the VPS targets are physically resolvable they will generally be detected; if they are detected they will generally be

identified correctly.

Some work on the reliability of the VPS has been carried out in [28]. 10,000 natural images were fed into VPS, and no targets were detected *i.e.* no false positives. In the light of this “over-engineered” quality of the VPS, the robustness profile of the VPS was explored by allowing the graph-searching condition to be relaxed for non-exact matches and by using a variety of three-level graphs as the key RAG. Certain combinations of tolerance levels and graph patterns, do usefully reduce the rate of false negatives (under partial occlusion conditions, *etc*) at little false positive cost. Overall, the method is shown to be robust to illumination variation and false detection. A novel initial thresholding algorithm is reported [28] to outperform existing adaptive two-level thresholding algorithms which may speed the performance of the VPS but no performance figures are provided.

The VPS uses region-based techniques to determine image coordinates to sub-pixel accuracy. Experiments described in [9] conclude the position measurement accuracy sufficient for AR applications.

VPS, like SCR, only regards targets as being recognised or not. The choice is between a system that reliably detects targets with no false positives like the VPS, or one which presents confidence values (HOM and ATK). Using a system of the latter variety is difficult. How are the different confidence metrics of different systems to be used by an application program without it becoming system dependent?

The processing time for a  $384 \times 288$  image was 32 ms on our four-processor parallel system, which scales to an equivalent uniprocessor time on a 933 MHz Pentium III for a  $320 \times 240$  image of 70 ms. However, unlike other systems the performance does not drop with the number of visible targets or the size of these targets within the scene as a graph searching algorithm is employed.

## 3 System choices

### 3.1 Choice of hardware

Our choice of hardware is designed to reflect existing practice within the custom embedded field through the use of Commercial Off-The-Shelf (COTS) processors connected by a high speed network. Possible future parallel targets are shared memory multi-processors and Field Programmable Gate

Arrays (FPGAs). It is suspected that a bus-dependent architecture and caching effects would restrict scalability, so we are working instead with FPGAs, which are capable of massive parallelism but of a limited type of computation. A migration path to the use of FPGAs is discussed in the Conclusion.

We decided upon a network of four PowerPC 7400 processors connected by a Myrinet network. The PowerPC 7400 is a low-power microprocessor that typically consumes 5.0 W at 500 MHz according to the manufacturer's specification and consequently is suitable for embedded applications. The choice to go for the high speed of Myrinet is especially significant, as past experience has demonstrated that, for data-intensive applications, performance is severely limited if the processor's compute rate is not balanced by adequate I/O bandwidth.

### **3.2 Choice of low level software**

The positioning algorithm is one of the applications implemented on this machine to test an evolving high-level software component protocol stack. The development of a low-level protocol stack for communication over Myrinet to facilitate the application is outlined in this paper. Rather than use a real-time operating system (OS), such as VxWorks [29], which would also have cost implications, the communication software runs from the Linux OS. A subset of MPI (Message-Passing Interface) [30, 31] commands brings standardisation, and provides a programmer-level access, while it is planned that the stack will be extended further upwards to give algorithm developer level access in due course.

## **4 A Custom parallel machine for AR**

### **4.1 Hardware**

Apart from a clean micro-architecture, unencumbered by the need to support a legacy programming model, the PowerPC, though still a commodity processor like the Pentium series, has versions with specialist features such as fused multiply/add and integral memory control unit, which can act independently of the system bus. Such features may have led to its choice as an

embedded core within the Xilinx Virtex-II Pro dense FPGA, which can contain up to four 300 MHz PowerPC 405 cores. The Virtex-II FPGA family, with between 3,168 and 50,382 logic cells, also has a range of built-in components such as an 18-bit hardware multiplier. Note that bandwidth scaling is accomplished on the FPGA: internally, by means of a 64-bit soft IBM CoreConnect bus at 100 or 133 MHz; and, externally to off-chip devices, by 16 RocketIO blocks each capable of 3.125 Gbps.

Therefore, the custom parallel machine on which the positioning algorithm runs, can also act as a development environment for such devices. Extending the PowerPC boards to include an FPGA on our current machine can be accomplished through a PMC (PCI Mezzanine Card). Communication between the nodes uses Myricom's switched parallel bus network, Myrinet [32].

Figure 1 is a block diagram of our machine with parallel all-to-all communication provided by the Myrinet crossbar switch. A photograph is shown in Figure 2. The PowerPC 7400s are clocked at 450 MHz. Each has a 2 MB level 2 cache connected by a back-side bus to the processor, and 256 MB SDRAM main memory. Boot-up of the PowerPCs is accomplished by means of the Ethernet connection via board-support package (BSP) software. A Myrinet PCI (Peripheral Component Interface) 64-bit bus card resides on the PC host, which is an 800 MHz Pentium III with 128 MB memory. Myrinet PMCs on each PowerPC board connect to a microstrip ribbon cable full-duplex 2.0 Gbps to form a SAN (System Area Network), *i.e.* confined to a VME enclosure.<sup>7</sup> Each Myrinet PMC has its own 133 MHz (or 200 MHz) RISC LANai communications processor along with receive and send buffers, as described for an earlier version of Myrinet in [33]. Communication between LANai and processor is via Direct Memory Access (DMA) to the LANai's and PowerPC address space on a per-process basis [34], thus avoiding excess memory-to-memory copies or CPU involvement, as would otherwise typically occur in a TCP/IP stack.

With customised software Myrinet in the manufacturer's benchmarks [35] sustained a maximum one-way data rate of about 2 Gbps. Using Myricom's GM message-passing API, a sustained data rate of about 1.8 Gbps with latencies was found. However, the data rate drops to about 250 to 1,765 Mbps when a TCP/IP stack is layered over GM. It is important to note that these bandwidth figures only relate to messages of about 10 KB,

---

<sup>7</sup>The 50/125 multimode fibre alternative allows connections over 10 m and full scalability of the network.

as there is rapid fall-off in performance for smaller messages, falling to half the rate for 1.2 KB, and less than 80 Mbps for messages less than about 100 B, due to packet handling and DMA costs.

In order to provide communication scaling across the system commensurate with Myrinet's bandwidth, a 66 MHz 64-bit PCI bus is sensible, with peak bandwidth of about 4 Gbps. However, some PowerPC boards only employ a 33 MHz local PCI 32-bit bus, with peak bandwidth of about 1 Gbps. There are a number of proprietary I/O buses, such as Raceway from DNA Enterprises Inc., effectively providing an external PCI bus. Because of this situation, four-processor cards with a dedicated bus to the enhanced PCI specification on one PMC site<sup>8</sup> were used in our experiments instead, though each is currently populated with one processor per card. The PowerPC system bus is rated at 64-bits/100 MHz, and can potentially allow shared-memory communication through 256 MB SDRAM (Synchronous DRAM) [36].

In our benchmarking of this application, the PowerPC was found to be 3% faster than the host processor, though on a MHz-by-MHz basis this equates to 83% faster. Note that these figures were obtained with compiler optimisation switched on. Without these optimisations the situation was reversed with the host processor being 38% faster. The PowerPC is big-endian machine whereas the Pentium is little-endian. The hidden cost of byte swapping (for all but grey-scale images) is, therefore, sensibly confined to the faster processor. Note that it is also possible to make use of byte-swapping support in the instruction set or setting a memory region view attribute, for example on the embedded PowerPC 405GP [37].

## 4.2 Low level software

Though a number of versions of Linux or toolkits for Linux for the PowerPC are available such as BlackLab Linux from Terrasoft, these are intended for the Apple iMAC, for which there is already system development software such as RPM (Redhat Package Module). The time to recreate this support for the present system was felt to be prohibitive. Moreover, many of the features of these alternative linuxes are not required for real-time systems. Therefore, the Linux BSP from Transtech Ltd along with a basic Linux kernel, HardHat Linux, were deployed. Unfortunately, the utilities supplied with this kernel were extremely minimal, causing myriad difficulties in de-

---

<sup>8</sup>The other PMC site is at the lower rating and is not dedicated.

velopment. For example even supplying standard networking utilities to allow remote working proved impossible. Instead, it was necessary to cross compile the portable “netutils” software designed to have minimal system dependencies. Real-time versions of Linux are in active development [38].

Myrinet’s GM API [39] is a lightweight communication layer, a variant of Active Messages [40], in theory providing reliable ordered connections, as well as addressing and wormhole routing on larger networks. Messages, which can be a theoretical maximum length of  $2^{31} - 1$  bytes, have two priority levels, allowing deadlock resolution. Client software access control to both send and receive buffers on the LANai board (Figure 3) is via a token system, similar to that used on some ATM networks. Send is non-blocking but completion of a message send is only notified when a receive-event queue is polled. Similarly, a receive enable is non-blocking, but the same receive-event queue must be polled to determine if the required message has arrived.

In our benchmarking of the Myrinet connections, using Myricom’s point-to-point test software, with GM the maximum bandwidth achieved (for a message size of about 4 kbits) was: 352 Mbps for communication from PowerPC to PowerPC, 392 Mbps PowerPC to the host Pentium, and 352 Mbps from Pentium to PowerPC. This bandwidth, approximately 0.4 Gbps, is clearly below Myricom’s testbench results, and strongly points to a performance bottleneck on the PowerPC, as the bandwidth improves when the high-specification Pentium is a receiver.

GM does not provide higher-level communication primitives such as scatter/gather or indeed those from the reduction family, nor indeed does GM provide any standardisation. However, MPI 1 & 2, as for example implemented in MPICH [41], can support a bewildering variety of messaging types. It is not always clear which of these message types have an efficient implementation for any one parallel machine. Though a port of MPICH has been made to *de facto* versions of Linux, this would not transfer to HardHat Linux, and the development work in affecting a port was felt to be prohibitive. Therefore, a subset of MPI calls were mapped directly onto GM. Just one ‘send’ and ‘receive’ were implemented, together with *inter alia* a one-to-all scatter and gather, rather than the process group restriction, which can require dæmons to manage group membership. Initialisation of GM is masked within an MPI initialisation call. It is intended that a set (possibly a complete set [42]) of parallel structures will eventually be mapped onto this MPI subset itself to provide a high-level development environment capable of porting software from one platform to another as the earlier platform becomes obsolete.

MPI’s message-passing is unsuited to hardware, and, in fact, CSP (Com-

municating Sequential Processes) channel semantics are more likely to map to hardware and software alike [43], which would be needed if FPGAs were added. Conveniently, however, conventional MPI allowed algorithmic development on other Linux machines before transfer to the parallel machine.

## 5 System details

Our VPS augmented reality system addresses the camera “pose estimation” problem using passive visual targets, and was produced before the ARToolkit emerged. The functioning of the VPS is qualitatively different in a number of respects from other such systems, and this section explores these differences and the advantages they confer.

### 5.1 Target design

A comparative study of visual markets system [2] uses the four evaluation criteria: (1) usability; (2) performance efficiency; (3) positioning accuracy and (4) reliability under adverse conditions. A paper on fiducial design [23] specifies the criteria:

- (a) unambiguous position and orientation
- (b) orientation independent
- (c) target image unlikely to be confused with large space of same
- (d) easy to locate and identify
- (e) image must work over wide camera capture range

The design criteria for the VPS were: diversity, distinguishability, detectability and locatability. The diversity refers to the range of targets, while distinguishability refers to the ability to differentiate between these targets. Figure 4 shows the semantic relationships between these criteria schema. It can be observed that where the semantics are related, the VPS criteria are more powerful than the other two sets and orthogonal and this section uses these to assess target design.

Figures 5 & 6 show sample targets. The targets are printed on paper by a PostScript printer. A range of alternative black-and-white target designs, including spiral and concentric circles were initially tested for detectability.

In the current system, location of one or more targets within an image frame is achieved by constructing a region adjacency graph (RAG) for each complete image, as shown in Figure 7. A RAG represents a region decomposition of a scene as a graph, in which each region is a node and each arc represents the adjacency relationship between regions. In addition, each node may be labelled by the colour of the corresponding region.

Each target has been designed to have a distinct and unambiguous form when converted to a RAG. The corresponding RAG forms of the targets in Figures 5 & 6 are shown in Figure 7. There are two parts to the target design: the border is used to detect the targets while the centre is used to distinguish one target from another. The border is constant from one target to another and is known as the key. The centre varies from target to target; this is known as the identifier or ID. Finding the targets is as simple as searching the scene RAG for instances of the key RAG. Identifying targets involves examining the ID RAGs connected directly to each key RAG found. Full details of the design of the patterns so as to maximise diversity can be found in [9]. A string representation for each target (in addition to the PostScript) allows them to be handled inside software. The standard set of 449 A4 targets is based on potentially scaled (sub-)colourings of an  $11 \times 11$  grid *i.e.* the ID pattern is drawn on an  $11 \text{ cm} \times 11 \text{ cm}$  area with no detail allowed to fall below 1 cm. The full set, but smaller number, of targets that can be generated by effectively colouring a  $9 \times 9$  grid is illustratively shown in Figure 8.

To find the position of a camera from an image requires at least four distinct fiducial points. In the current design, there are four rectangular markers embedded in the corners of each target supplying the fiducial points. These are distinguishable as the target design has no rotational or reflective symmetry. Thus, it is not necessary to use a set of targets as fiducial points. In comparison, the BBC out-of-shot system needs a number of targets to find the relative position.

In contrast with both BBC, CyberCode and other schemes, identification of an individual target is not by an embedded bar-code. In general, complex 2D barcodes [44] require close inspection under ideal illumination conditions by laser scanners or CCD cameras with short focal lengths. However, RAGs have the advantage that a wide range of distinguishable patterns are possible under low-resolution conditions, chiefly because identification is not by edges but by regions.

Note that VPS target detection and identification is contingent on topological rather than geometric information and so is immune to lens distortion. A target recognition system based on a panoramic view camera is

described in [22]. This uses the ARToolkit and the panoramic image has to be “undistorted” before the ARToolkit can work. The VPS could work directly with the panoramic image - only the resulting coordinate information would have to be undistorted mathematically.

## 5.2 The positioning algorithm

The stages in the algorithm are:

- Pre-process the image so that it is in the form of a RAG.
- Locate a target in the image by means of the RAG. This target may be already matched in a previous frame, or a new target that has come into view.
- Locate the four fiducial points within the target.
- Perform geometry calculations to find the position of the camera relative to the target.

## 5.3 Pre-processing

This stage has four sub-stages:

**Filtering** To account for varying image contrast and intensity, adaptive filtering is performed. The variance and mean pixel intensities within an odd-sized square window are found through calculation of running means, as the window is slid over the image. If the pixel intensity is within a given number (under command line control) of standard-deviations from the mean then the pixel is classified as grey. Otherwise, if the pixel’s intensity is below the mean intensity of the window it is classified as black, and if above as white. Image boundaries were mirrored (Figure 9 with an earlier target design), to make filtering possible near image edges, though there are other possibilities such as clamping to the boundary value. A small filter window, while reducing the computational burden, may fit entirely within a target region, resulting in a ‘speckling’ of black or white pixels through relatively small changes in individual pixel intensity in a conventional bi-level categorisation as in Figure 10. However, such speckling can be usually avoided as these regions generally come out as grey in the tri-level

categorisation as in Figure 11. As a later stage in the processing, the inherent ambiguity in grey regions will be removed (see Create RAG)

The filter window used for timing experiments reported in this paper is  $21 \times 21$  pixels, corresponding to a border region width of 10 pixels. This size was the minimum that gave reliable region detection in all circumstances *before* the noise reduction technique based on grey regions was introduced *i.e.* one could probably get away with a smaller, less computationally demanding filter. However, the use of a running mean algorithm reduces the computational effect of changing the size of the filter, because a single column of new entries is introduced into the window (and taken out) rather than recalculating for the whole of the window coverage. The overall computational complexity reduces from  $O(i_w i_h f_w f_h)$  to  $O(i_w i_h)$ , where  $i_w$ ,  $i_h$ ,  $f_w$  and  $f_h$  represent the image and filter widths and heights respectively. This reduction also includes the separability of the 2D filter though it has to be recognised that extra set-up costs are incurred by the running mean.

**Colouring** The following well-known ‘painter’s algorithm’ for finding connected components [45] was used to identify regions within the filtered image. Four-way connectivity was found to be adequate for this application.

The tri-level image is traversed by scanline from top to bottom, and from left to right. If the current pixel’s intensity differs from both the left- and upper-neighbours’ intensities assign a new id. or colour, and equivalence set. If the neighbour’s intensities are the same but the colours are different select the lowest value from the equivalence set of colours, otherwise adopt the neighbours’ colour. Subsequently, replace each pixel’s colour by the lowest value in its colour equivalence set, which involves sorting equivalence relation chains.

**Region statistics** A region table stores per-region information: the original tri-level colour; the new colour value ( $\in \{1, \dots, n\}$  where  $n$  is the number of regions); region bounding box; and the centroid. The bounding box is found by simply observing maximum and minimum region extents. Additionally, regions with very few pixels (through a user defined threshold) are set to a “notional” colour 0 and ignored in subsequent processing.

**Create RAG** The image is again scanned as before. A graph is formed by making an arc whenever either the left or upper pixel colour differs

from the current pixel colour, though ignoring colour 0. The result is a RAG with nodes of three different intensities, superimposed as in Figure 12 . To avoid the indeterminacy presented by the grey regions, all nodes representing white connected to a grey node are merged, and similarly for the nodes representing black. The grey nodes are then removed. This rationalisation of the graph is to facilitate noise-resilient location of a black and white target regions. The removal of grey nodes resulting from the image in 12, can be shown as the transition from the RAG in Figure 13 to that in Figure 14. The areas of the scene RAG which correspond to the target ID, the target key and the remaining parts of the scene are annotated.

#### 5.4 Location of target in the image

A RAG can be represented as a set of 2-D coordinates, *e.g.* (2, 28) connects region 2 to region 28 or as a 2D array where entry  $(i, j)$  is marked with a 1 if an arc exists. A recursive graph-searching algorithm was used to find the generic target key sub-graph. Though in general finding a sub-graph within a graph is known to be NP-complete, *e.g.* [46], given that the target key is a tree and given the limited size of the scene graph, in practice, the search was tractable. The individual target ID RAG is connected to the generic key RAG. However, it was also found to be necessary [9] to sort regions into area order in order to resolve identification ambiguities. The graph searching was prototyped in the functional language ML and then ported to ‘C’ and incorporated within the image processing software. Prototyping the graph searching code in ML facilitated software development. *Ab initio* development of graph searching code in ‘C’, would have been difficult.

#### 5.5 Location of fiducial points

Once the targets are found their 2D positions are taken as the previously calculated centroids of the corresponding regions in the tri-level image. The centroids of the four regions acting as markers are used as the fiducial points.

#### 5.6 Geometry calculations

The nub of the positioning problem is that, given the pixel coordinates  $(u_i, v_i)$  within an image of the projection of a number of known fiducial

points  $(X_i, Y_i, Z_i)$  in the real world, how can the position of the camera be found? The projection relationship between each image and world point is:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

This may be expressed more compactly as

$$\mathbf{u} = \mathbf{P}\mathbf{X} \quad \text{where} \quad \mathbf{P} = \mathbf{C}[\mathbf{R}|\mathbf{T}]$$

$\mathbf{P} \in \mathbb{R}_{3 \times 4}$  is called the camera projection matrix which can be decomposed as shown.  $\mathbf{C} \in \mathbb{R}_{3 \times 3}$  is an upper triangular matrix called the camera calibration matrix and encapsulates the camera dependent aspects of the projection. When  $\mathbf{C}$  is known, the camera is said to be calibrated.  $\mathbf{R} \in \mathbb{R}_{3 \times 3}$  is the rotation matrix and  $\mathbf{T} \in \mathbb{R}_3$  the translation vector that together give the transformation between the world and camera coordinates system.

The problem can be re-expressed as determining  $\mathbf{R}$  and  $\mathbf{T}$ . Each point correspondence places two constraints on the 12 elements of  $\mathbf{P}$ . As  $\mathbf{P}$  is not unique up to scale there are 11 unknowns, and 6 (or more) point correspondences are therefore required to solve for  $\mathbf{P}$ . Implementations now solve the system of equations, either as herein by linear methods, or by non-linear methods [11]. Subsequent determination of  $\mathbf{R}$  and  $\mathbf{T}$  from  $\mathbf{P}$  is known from the camera calibration problem [47]. A valid solution can be obtained provided the fiducial points are not coplanar.

In the special case that fiducial points are coplanar (*i.e.* on a flat ceiling), then the origin of the camera coordinate system can be shifted so that the  $Z$  coordinates are all zero. Terms then drop out resulting in just 7 unknowns in  $\mathbf{P}$ , and hence just 4 fiducial points are required. However, given the above constraint, the camera calibration matrix must be known in advance to work out  $\mathbf{R}$  and  $\mathbf{T}$  in this instance.

The VPS can work whether the targets are coplanar or not. Accuracy is improved with a non-coplanar target set, but using a pre-computed camera calibration matrix (forced by coplanar operation) is also more accurate than calculating one on a frame-by-frame basis. Composition on the optical axis (essentially directly augmenting the video image) does not require matrix factorisation at all and this gives the best results.

## 6 Parallelisation Results

### 6.1 Serial algorithm

The application was benchmarked on the host Pentium with compiler optimisation. The performance at half-PAL resolution<sup>9</sup> was 8.45 fps, and at full-PAL resolution (768 × 576 pixels) was 1.97 fps. Performance on a PowerPC, also with compiler optimisation turned on, was 8.72 fps. Performance for full-PAL on a single PowerPC was 1.71 fps when accessing a frame sequence from local memory. If the frame sequence was obtained via the host PC rather than directly then the rate dropped to 1.48 fps.

The half-PAL result is, therefore, on the Pentium and PowerPC respectively 7% and 17% faster than linear extrapolation from the full-PAL result might lead one to expect. The relative discrepancy is most likely to be (though not proven to be) an effect connected with a favourable caching regime, for example data access patterns *vis-à-vis* cache line size or total cache size. The PowerPC's level 2 cache is 2 MB, whereas the host PC level 2 cache is only 256 KB. It would appear that the PowerPC on the custom machine were more susceptible to caching effects than the host Pentium.

GNU compilers were used for both the Pentium and the PowerPC. The 'C' code for the PowerPCs was cross-compiled on the Pentium host. On both processors, full level-nine optimisation was turned on, as was loop unrolling, after checking which compiler flag settings led to improvements for this application. The effect of turning on optimisation and setting flags was dramatic on the PowerPC, leading to a doubling of performance, and indeed, for half-PAL outstripped the Pentium despite what one might expect from a superficial consideration of clock speeds.

It was found useful to test in both half- and full-PAL resolution. The choice of half-PAL size avoids the effect of interlacing and reduces the search area by three quarters. Interlacing in full-PAL images may result in regionalised motion blur, as a result of combining two frames offset in time from each other. However, full-PAL resolution has the advantage that distant objects can be resolved more readily.<sup>10</sup>

---

<sup>9</sup>The term half-PAL resolution in this paper is used to distinguish from CIF (Common Interchange Format) at 360 × 288 active pixels, and should not be confused with PAL signal transmission format.

<sup>10</sup>We did not use progressive scan mode because the camera available to us did not have

The obvious conclusion from testing on a uni-processor was that neither half-PAL nor full-PAL resolution processing can be performed at video rate, either 25 Hz or 30 Hz respectively for UK and USA television systems. The next step was to examine which parts of the application could be parallelised. Table 1 summarises the results of profiling the positioning code using the Unix utility `gprof`.

Despite the complexity of the geometry calculations and relative attention given to this topic in the literature, the time taken in calculation is relatively trivial in comparison to the pre-processing stage, in which per-pixel calculations take place. The scaling between the colouring and filtering sub-stages is no doubt due to the fact that the former algorithm only inspects the adjacent upper and lower pixels, whereas the latter algorithm, despite optimisation, inspects the surrounding region. Note that the graph searching takes a minimal (0.2%) of the processing. The time complexity of the search is given by  $O(nm)$  where  $n$  and  $m$  are the cardinality of the scene and fiducial trees. Note that bi-level images yield RAGs which are trees. So while this time complexity is worrying in theory, practice suggests fretting about it is unproductive.

## 6.2 Parallel algorithms

As approximately 60% of the cost of the serial version is confined within the filtering sub-stage, then parallelising the filtering stage is the priority. Filtering can be made self-contained by overlapping sub-image boundaries by half the size of the filtering window. Given the better communication performance as message sizes become larger and the deterministic nature of computation, splitting the image into as few non-overlapping sub-images as possible to minimise the number of border data exchanges also has its attractions. The alternative would be to farm out sub-images, *e.g.* image strips, but this approach [48] works best when buffering is used to allow overlap of communications with computation, and when load-balancing is needed to counter non-constant computation times. The main disadvantage of geometric parallelism is the latency arising from the initial distribution of data, and its subsequent collection.

In Figure 15, the host holds the image which has been divided into four to match the number of compute nodes. The host is responsible for an initial scatter of the image blocks and a final gather, using MPI commands.

---

sufficiently fast shutter speed for this mode, resulting in conventional motion blur.

Results are exchanged across the image block internal boundaries by the PowerPCs in successive phases, while, in order to maintain the symmetry of the algorithm the image block edges are mirrored at external boundaries (Section 5.3).

The three other pre-processing stages: colouring, region statistics, and RAG creation are also parallelisable. Again the image remains divided into four, and for each sub-image a RAG and region table are ultimately produced. Unfortunately, a consistent colouring scheme is not maintainable across the sub-images. Consider the case of two adjacent sub-images, then the adjoining edge pixels between two sub-images are scanned, vertically or horizontally depending on the edge orientation. If adjacent pixels have the same tri-level value then an equivalence relation is set up between the regions in which the two pixels reside. In essence a consistent renumbering is required: region 5 in one image is probably unconnected with region 5 in the adjoining one. The RAGs are then merged, accounting for the newly established equivalence relationships. The region tables are merged by super-setting bounding boxes, and computing new centroids as the weighted sum of component centroids. Only one of the two processors holds the resulting accumulated region table and RAG.

In Figure 16, a generalisation of the binary merging scheme for a  $4 \times 5$  processor array is shown, numbered with the steps in the overall accumulate operation. The example demonstrates that the accumulate can be performed when array dimensions are not an exact power of two. At each exchange of data, the accumulated region table, the accumulated RAG, and such edge data that is needed for forming the accumulations in the next *and* subsequent accumulations is passed. The final accumulate presents the aggregated region table and RAG to an individual PowerPC, which subsequently processes the remaining steps of the algorithm in serial fashion.

### 6.3 Comparative performance

Parallelisation proceeded up to the point that a real-time rate was achieved. The results for varying number of PowerPCs (PPC) for half-PAL is summarised in Table 2, when only the filtering algorithm was parallelised. Efficiency is defined as  $\frac{100 \cdot n \cdot s}{p}$ , where  $s$  is the time for one processor to complete the task, and  $p$  is the time for  $n$  processors working in parallel to complete the task. Table 3 shows the corresponding performance after all pre-processing was parallelised according to the algorithms in Section 6.2. It will be seen that American real-time rates are now achieved, and what is more there is

little slack in the performance.

Turning to full-PAL resolution performance, Table 4 & 5 summarise the results. The result shows that real-time performance is still out of reach on the custom parallel computer, though it should be appreciated that a very large filter of  $21 \times 21$  pixels was used in the timing trials. However, the efficiency figures, which remain similar across the two image sizes, suggest that adding processors could result in real-time performance.

The communication performance is summarised in Table 6, found by averaging timings over a number of frames. Each timing was taken on a single processor, and then the total bandwidth obtained from the known amount of data transferred. Whereas the same message sizes occurred in scatter and gather operations, for the gather, messages were transferred in parallel across the Myrinet links, explaining the higher bandwidth. Two bytes per pixel were employed in the swap operation, with communication parallelism. The accumulate bandwidth figure was reduced due to the smaller message sizes and due to inclusion of intermediate processing steps in the timings. These figures compare favourably with the benchmark for GM communication alone, running on the custom machine without MPI (Section 4). The bandwidths returned, though considerably less than might have been hoped for, given Myricom's raw benchmarking figures, are still an order of magnitude greater than parallel machines have had to be content with in the past, and effectively created a compute-intensive application from one that would have been communication limited.

## 7 Conclusion

AR is significantly more compute intensive than VR, yet it is even more critical that the associated calculations (such as the position finding of this paper) are performed at video rate. Just as 3D graphics cards have been developed to meet the requirements of computer games (and by extension VR), AR demands the equivalent to accelerate vision-based processing. This paper is an attempt to scope such hardware, as the real world component of AR introduces particularly demanding real-time constraints.

Having developed the VPS<sup>11</sup> (a general-purpose software AR toolkit), it was found that on general-purpose processors, performance was significantly below real-time. However, through parallelisation of this software, real-

---

<sup>11</sup>Version 1.1 is available at <http://privatewww.essex.ac.uk/~djjohn/>

time performance has easily been achieved for half-PAL resolution images, leaving time for further processing after positioning is accomplished. It is also established that comparable improvements in performance for full-PAL resolution images are readily achievable if further processors were deployed, such is the efficiency of the parallelisation.

The efficiencies achieved are attributed to the custom computer that has been constructed with a more than sufficient interconnect bandwidth. In fact, results also show that the PowerPC nodes are comparable in performance to Intel processors running at nominally faster clock rates, with the advantage of reduced power usage.

In a passive, vision-based system, position is found firstly by target identification and secondly by computing camera orientation relative to the target. Rectangular targets have been designed which result in robust detection and identification through region-based image processing. As a result of a top-down analysis of the algorithm set, spatial filtering of the video image, and extraction of target identification information were identified as the keys to performance. Accelerating the filtering algorithm alone is not sufficient but with the additional parallelisation of region generation, and Region Adjacency Graph (RAG) construction, the desired result was achieved.

The RAG approach employed is significantly different to that of other systems, both the typical barcode-based system and that of the widely used AR-Toolkit. It possesses significant advantages in robustness, particularly in immunity to geometric distortion, lighting conditions, and misidentification, at the expense of a relatively computationally expensive initial filtering stage. The VPS was timed using a large  $21 \times 21$  filter which is probably unnecessary in many circumstances especially given the graph-based noise reduction stage later in the image processing pipeline. Note that the filter size is under user control.

We are aware that the custom machine described does not yet fulfil the goal of general-purpose AR (*i.e.* interpreting sequences of full-PAL at real-time rates), but consider the current implementation as an encouraging prototype demonstrating scalability. The VPS software functions reliably, and at half-PAL, is quite capable of working on a modern high-end PC at interactive rates and has been successfully used in a number of AR applications. Further work is to transfer the application onto a commodity device, namely the Virtex-II Pro from Xilinx, as it is anticipated that the spatial filtering algorithm will be suitable for data parallel processing on the FPGA component. In more general terms, by abstraction of typical communication patterns as present in this application it is intended to construct a higher-level implementation environment based around the notion

of software components for parallel applications.

## **Acknowledgement**

This work is being carried out with assistance from an EPSRC/DERA Grant Reference No. GR/N20980.

## References

- [1] R. T. Azuma. A survey of augmented reality. In *Computer Graphics'95*, pages 1–38, 1995. Course Notes # 9.
- [2] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in ar systems: A comparative study. In *Proceedings of the International Symposium on Mixed and Augmented*, pages 97–106, 2002.
- [3] M. Olano, J. D. Cohen, M. R. Mine, and G. Bishop. Combatting rendering latency. In *Symposium on Interactive 3D Graphics*, pages 19–24, 204, 1995.
- [4] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through HMD. In *SIGGRAPH'94*, pages 197–204, 1994.
- [5] L-T. Cheng and J. Robinson. Dealing with speed and robustness in video-based registration on a wearable computing platform. In *2<sup>nd</sup> International Symposium on Wearable Computers*, 1998.
- [6] D. Sims. New realities in aircraft design and manufacture. *IEEE Computer Graphics and Applications*, 14(2):94, 1994.
- [7] G. A. Thomas, J. Jin, T. Niblett, and C. Urquhart. A versatile camera position measurement system for virtual reality TV production. In *International Broadcasting Convention (IBC'97)*, pages 284–289, 1997.
- [8] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time vision-based tracking for augmented reality applications. In *Symposium on Virtual Reality Software and Technology (VRST-97)*, pages 87–94, 1997.
- [9] D. J. Johnston. *Position Sensing and Augmented Reality*. PhD thesis, University of Essex, 2002.
- [10] B. W. Parkinson and J. J. Spilker Jr., editors. *Global Positioning Systems: Theory and Applications*, volume 1. American Institute of Aeronautics, 1995.
- [11] D. LaRose. A fast, affordable system for augmented reality. Master's thesis, Carnegie Mellon University, 1998. Report No. CMU-RI-TR-98-21.

- [12] L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.
- [13] A. Hopper. Sentient computing, 1999. Royal Society Clifford Patterson Lecture.
- [14] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.
- [15] R. Want, K. P. Fishkin, and B. L. Harrison. Bridging physical and virtual worlds with electronic tags. In *CHI'99 Proceedings*, pages 370–377, 1999.
- [16] J. Paradiso and K-Y. Hsiao. Swept-frequency, magnetically-coupled resonant tags for realtime, continuous, multiparameter control. In *CHI'99*, pages 212–213, 1999.
- [17] W. Barfield and T. Caudell. *Fundamentals of Wearable Computers and Augmented Reality*. Lawrence Erlbaum Assoc, 1<sup>st</sup> edition, 2000.
- [18] J. P. Mellor. Realtime camera calibration for enhanced reality visualization. In *Computer Visions, Virtual Reality, and Robotics in Medicine'95 (CVRMed '95)*, pages 471–475, 1995.
- [19] J. Rekimoto. Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings of the Asian Pacific Computer Human Interfaces (APCHI'98)*, 1998.
- [20] H. Fuchs and J. Poulton. Pixel-Planes: A VLSI-oriented design for a raster graphics engine. *VLSI Design*, 2(3):20–28, 1981.
- [21] M. Billinghurst, H. Kato, and I. Poupyrev. Magic book: Transitioning between reality and virtuality. In *CHI'01*, 2001.
- [22] M. Fiala. Artoolkit applied to panoramic vision for robot navigation. In *16th International Conference on Vision Interface*, 2003.
- [23] C. B. Owen, F. Xiao, and P. Middlin. What is the best fiducial? In *The First IEEE International Augmented Reality Toolkit Workshop*, pages 98–105, 2002.
- [24] J. Rekimoto and Y. Ayatsuka. Cybercode: Designing augmented reality environments with visual tags. In *Designing Augmented Reality Environments (DARE 2000)*, 2000.

- [25] M. Appel and N. Naveb. Registration of technical drawings and calibrated images for industrial augmented reality. *Machine Vision and Applications*, 13(3):111–118, 2002.
- [26] Arvika project: Augmented reality for development, production and servicing. Details on [http://www.arvika.de/www\\_extant](http://www.arvika.de/www_extant) 17/07/03.
- [27] López de Ipiña D. Video-based sensing for wide deployment of sentient spaces. In *2<sup>nd</sup> PACT Workshop on Ubiquitous Computing*, 2001.
- [28] E. Costanza and Robinson J. A. A region adjacency tree approach to the detection and design of fiducials. In *Vision, Video, and Graphics 2003*, 2003.
- [29] Wind River Systems Inc. *VxWorks 5.1 Programmers Manual*. Wind River Systems Inc., 1010 Atalantic Avenue, Alameda, CA, 1993.
- [30] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI Portable Parallel programming with the Message-Passing Interface*. MIT, Cambridge, MA, 1995.
- [31] W. Gropp, B. Lusk, and S. Thakur. *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT, Cambridge, MA, 1999.
- [32] N. Boden, D. Cohen, R. Feldman, A. Kulawik, C. Seitz, J. Seizovic, and S. Wu. Myrinet: A gigabit-per-second local area network. *IEEE Micro*, 15(1):29–38, 1995.
- [33] D. E. Culler, J. P. Singh, and A. Gupta. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann, San Francisco, 1999.
- [34] B. N. Chun, A. M. Mainwaring, and D. E. Culler. Virtual network transport protocols for Myrinet. *IEEE Micro*, 18(1):55–63, 1998.
- [35] Myricom Inc. GM 1.4 API Performance, 2001. Available from <http://www.myricom.com/myrinet/performance/index.html>.
- [36] W. W. Smith. Radar processing on the DNA VQ750 multi-processor architecture, 2000. Application note from DNA Enterprises Inc.
- [37] IBM Microelectronics Inc. PowerPC 405GP endianness and bit naming conventions, 2000. Application note from <http://www.chips.ibm.com>.

- [38] V. Yodaiken. The RTLinux manifesto. In *The 5th Linux Expo, Raleigh, NC*, 1999.
- [39] Myricom Inc. *The GM Message Passing System*. Myricom Inc., 325 N. Santa Anita Ave., Arcadia, CA 91024, 2000.
- [40] T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauer. Active Messages: A mechanism for integrated communication and computation. In *19<sup>th</sup> Annual International Symposium on Computer Architecture*, pages 256–266, 1992.
- [41] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message-passing standard. *Parallel Computing*, 22(6):789–828, 1996.
- [42] D. B. Skillicorn and D. Talia. Models and languages for parallel computation. *ACM Computing Surveys*, 30(2):123–169, 1998.
- [43] M. Fleury, R. P. Self, and A. C. Downton. Hardware compilation for software engineers: An ATM example. *IEE Proceedings Software*, 148(1):31–42, 2001.
- [44] R. Adams. 2-dimensional bar code page, 2001. Survey of bar-codes available at <http://www.adams1.com/pub/russadam/stack.html> *extracted 17/07/03*.
- [45] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, New York, 1995.
- [46] A. Blum and R. L. Rivest. Training a three node neural network is NP-complete. In *IEEE Conference on Neural Information Processing Systems*, 1988.
- [47] O. Fougeras. *Three-Dimensional Computer Vision*. MIT, Cambridge, MA, 1993.
- [48] M. Fleury, A. C. Downton, and A. F. Clark. Performance metrics for embedded parallel pipelines. *IEEE Transactions on Parallel and Distributed Systems*, 11(11):1164–1185, 2000.
- [49] P. Malbezin, W. Piekarski, and B. H. Thomas. Measuring ARToolkit Accuracy in Long Distance Tracking Experiments. In *1<sup>st</sup> International Augmented Reality Toolkit Workshop, Darmstadt, Germany*, 2002.

Function	Percentage
Filtering	57.31
Colouring	15.85
Region statistics	12.44
RAG creation	9.84
RAG search	0.20
Geometry calculations	0.72
Utility functions	3.62

Table 1: Profiled performance of position finding code

No. of PPC	Frame/s	Speed-up	Efficiency %
1	8.72	1.00	100.0
2	11.95	1.37	68.5
4	19.61	2.25	56.2

Table 2: Performance for half-PAL resolution, partial parallelisation

No. of PPC	Frame/s	Speed-up	Efficiency %
1	8.72	1.00	100.0
2	16.38	1.90	93.9
4	31.51	3.61	90.3

Table 3: Performance for half-PAL resolution parallelisation

No. of PPC	Frame/s	Speed-up	Efficiency %
1	1.72	1.00	100.0
2	2.44	1.42	71.3
4	3.91	2.27	66.0

Table 4: Performance for full-PAL resolution, partial parallelisation

No. of PPC	Frame/s	Speed-up	Efficiency %
1	1.72	1.00	100.0
2	3.26	1.90	94.8
4	6.12	3.56	90.0

Table 5: Performance for full-PAL resolution parallelisation

Operation	Total bandwidth Mbps	Message size KB
Scatter	232	81
Gather	384	81
Swap	206	14
Accumulate	48	27

Table 6: Communication performance for half-PAL resolution

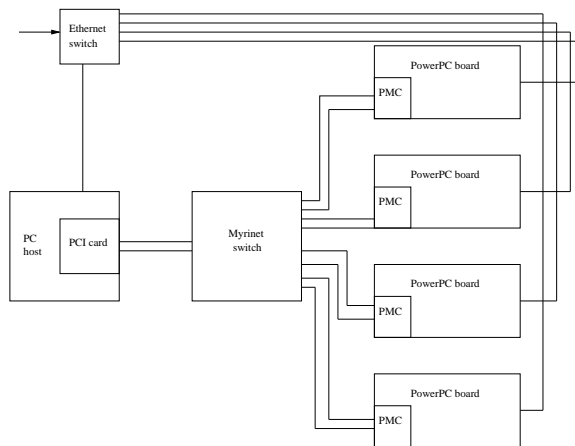


Figure 1: Diagram of Custom Parallel Machine

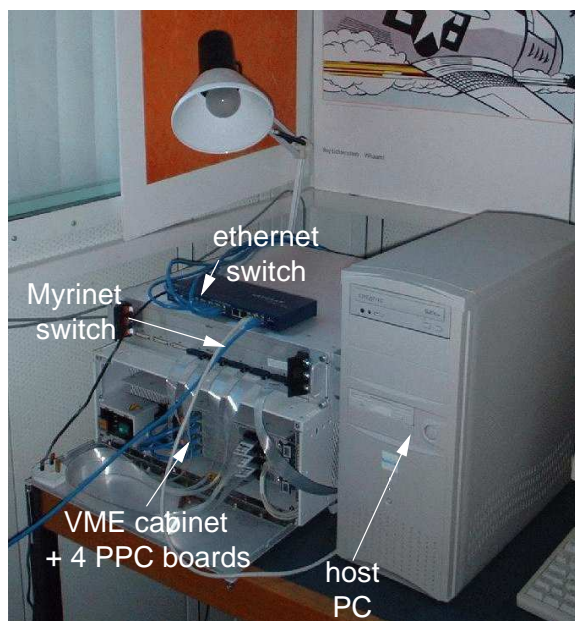


Figure 2: Photograph of Custom Parallel Machine

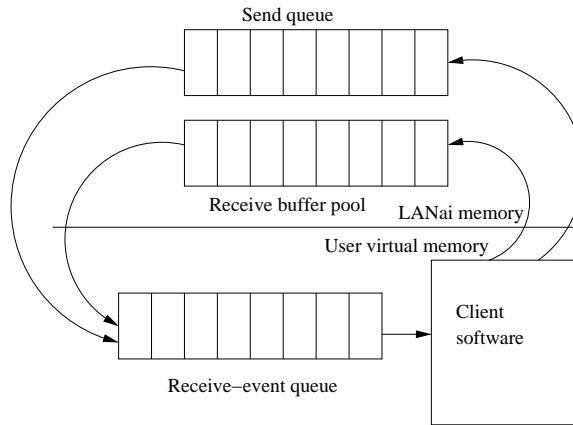


Figure 3: Communicating with GM

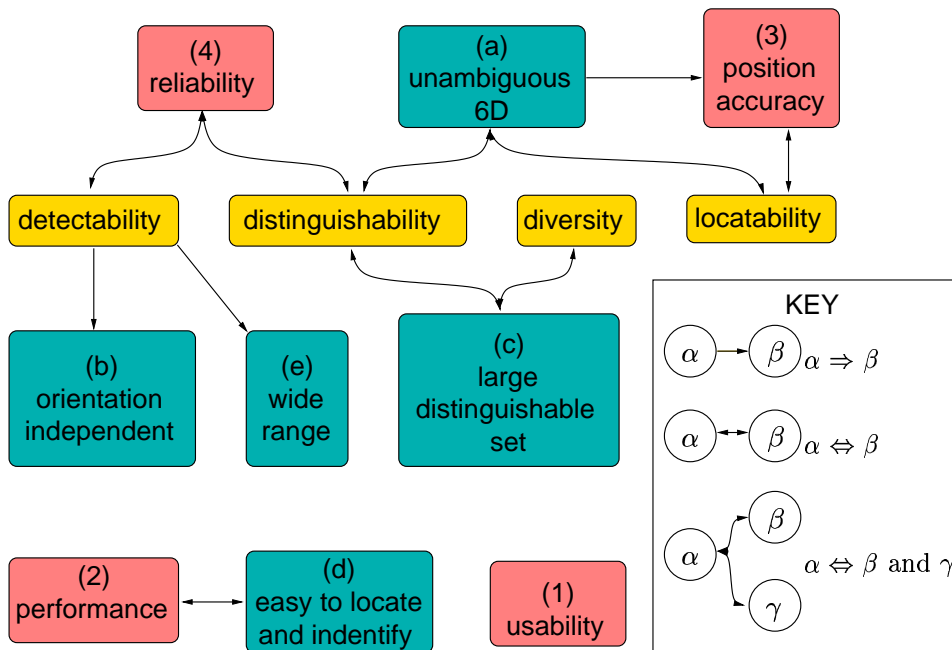
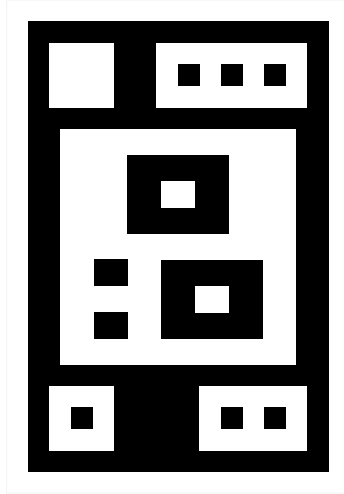
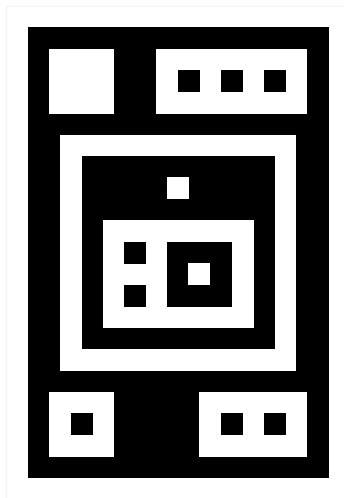


Figure 4: Semantic Relations Between Criteria



22

Figure 5: Example target A



125

Figure 6: Example target B

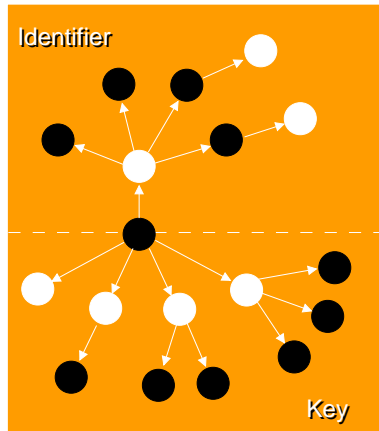


Figure 7: Example RAG (Region Adjacency Graph) with identifier and generic key

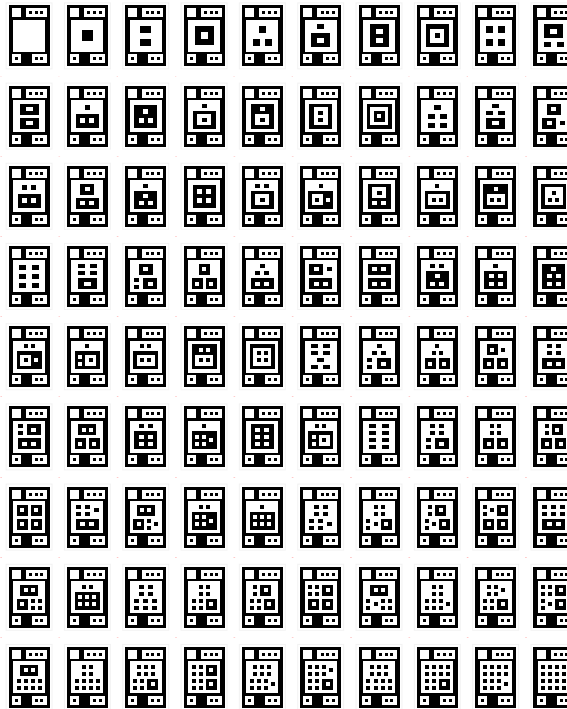


Figure 8: Target set

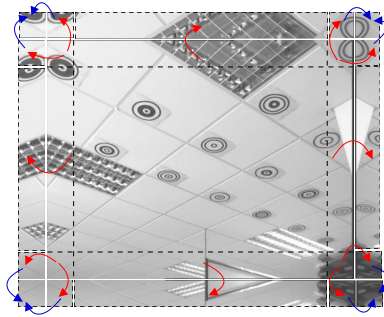


Figure 9: Treatment of edge boundaries (halo) in filtering operation

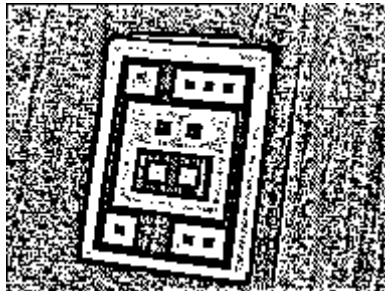


Figure 10: bi-level processed image

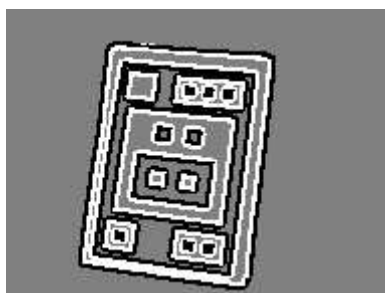


Figure 11: tri-level processed image

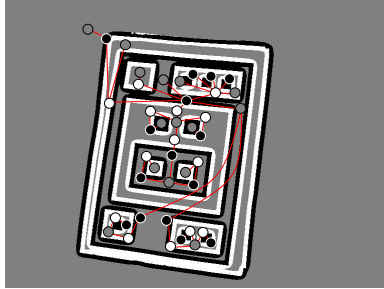


Figure 12: RAG superimposed on processed image

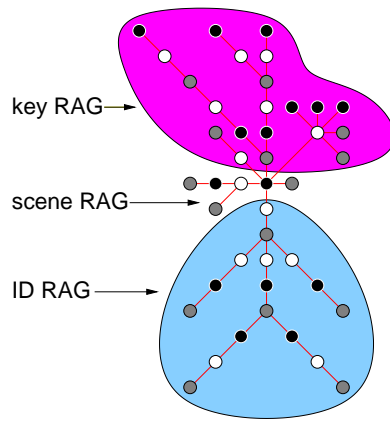


Figure 13: RAG with rationalised layout

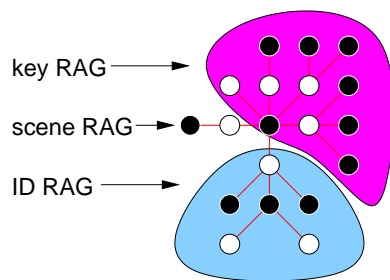


Figure 14: RAG with rationalised layout — grey nodes removed

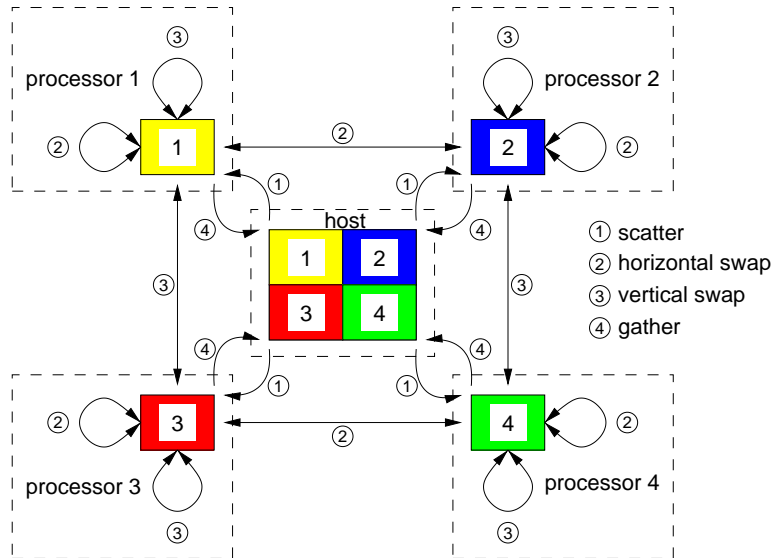


Figure 15: Parallel communication for filtering algorithm

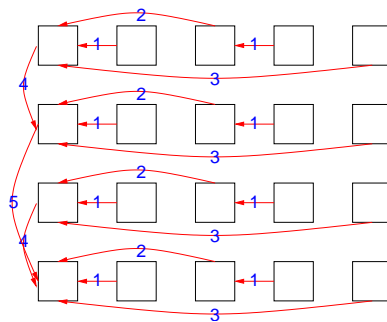


Figure 16: Parallel communication for accumulate algorithm

## List of Tables

1	Profiled performance of position finding code . . . . .	31
2	Performance for half-PAL resolution, partial parallelisation . . . . .	31
3	Performance for half-PAL resolution parallelisation . . . . .	31
4	Performance for full-PAL resolution, partial parallelisation . . . . .	31
5	Performance for full-PAL resolution parallelisation . . . . .	32
6	Communication performance for half-PAL resolution . . . . .	32

## List of Figures

1	Diagram of Custom Parallel Machine . . . . .	33
2	Photograph of Custom Parallel Machine . . . . .	33
3	Communicating with GM . . . . .	34
4	Semantic Relations Between Criteria . . . . .	34
5	Example target A . . . . .	35
6	Example target B . . . . .	35
7	Example RAG (Region Adjacency Graph) with identifier and generic key . . . . .	36
8	Target set . . . . .	36
9	Treatment of edge boundaries (halo) in filtering operation . . . . .	37
10	bi-level processed image . . . . .	37
11	tri-level processed image . . . . .	37
12	RAG superimposed on processed image . . . . .	38
13	RAG with rationalised layout . . . . .	38
14	RAG with rationalised layout — grey nodes removed . . . . .	38
15	Parallel communication for filtering algorithm . . . . .	39
16	Parallel communication for accumulate algorithm . . . . .	39