Hand Tracking For Low Powered Mobile AR User Interfaces

Ross Smith[1], Wayne Piekarski[1], and Grant Wigley[2]
Wearable Computer Lab[1] / Reconfigurable Computing Laboratory[2]
School of Computer and Information Science
University of South Australia
Mawson Lakes, SA, 5095, Australia
ross@cs.unisa.edu.au, wayne@cs.unisa.edu.au, wigley@cs.unisa.edu.au

Abstract

Mobile augmented reality systems use general purpose computing hardware to perform tasks such as rendering computer graphics, providing video overlay, and performing vision tracking. Our current Tinmith-Metro modelling system implements a user interface which is based on tracking the motions of gloves worn by the user, but is implemented inefficiently in a mobile laptop carried on a backpack by the user. This paper describes how we have developed a tracking algorithm which is suitable for implementation in a field programmable gate array. This implementation uses minimal power and will allow future miniaturisation of our mobile backpack equipment. We present the results of studies conducted outdoors to find the most appropriate marker type to use, and also the overall results that were achieved during testing.

Keywords: augmented reality, wearable computers, field programmable gate arrays, mobile user interfaces.

1 Introduction

Our current main area of research has been in developing mobile outdoor augmented reality systems, particularly those that allow the user to interact with the 3D environment directly. Our Tinmith-Metro software allows users to perform real-time 3D modelling in outdoor environments using the body and the hands as the user interface (Piekarski 2001) (Piekarski 2003). Tinmith-Metro runs on a backpack computer worn by the user (shown in Figure 1) and operates completely autonomously in an outdoor environment. Since the system is mobile, it is important that the interface to control it is portable but yet still intuitive to use. Our user interface relies on motions of the user's hands to provide an intuitive set of controls to interact with the 3D environment. A cursor is implemented by tracking markers on the gloves worn by the user, and menus are selected by pinching different fingers with the thumbs.

Our existing systems have relied on the use of fiducial markers placed on the tips of the thumbs of gloves worn by the user, as shown in Figure 2. These fiducial markers are tracked using ARToolKit software (Kato 1999), and while this approach worked well for our earlier systems there were a number of drawbacks. Firstly, ARToolkit uses a complex algorithm that provides a full six degrees of freedom (6DOF) tracking and is very sensitive to error, while only two dimensional tracking is required for our user interface. Tracking failures are quite common in typical outdoor environments where conditions are constantly changing and quite harsh. Images with bright or dark backgrounds, or specular highlights on fiducial markers, are common in varying sunlight and the tracking tends to fail quite often. Secondly, the ARToolKit as well as the live video input and overlay also requires a considerable portion of the processor in the mobile laptop. By removing the need for these tasks to be performed on the laptop we can use smaller and less powerful computers, and free up the processor for other useful tasks. Therefore in this paper, we describe how we have used specialised hardware to implement functionality formerly provided by a general purpose laptop processor. Instead of tracking complex fiducial markers described previously, simple coloured balls that are immune to most lighting problems are used instead, as shown in Figure 2. Robust tracking is critical for developing applications that are easy to use because tracking glitches can cause very confusing side effects for the user.

Most augmented reality (AR) research that has been published to date relies on the use of general purpose computing hardware to perform computations, render computer graphics, and provide video overlay functionality. Systems that rely on this general purpose computing hardware will be larger in size and consume more power than ones which have devices customised for specific tasks. When working in indoor environments, issues such as weight and size restrictions and power consumption are rarely considered as the systems are not required to be mobile. When working outdoors however, these issues are very important as the user may be required to carry the system around with them.

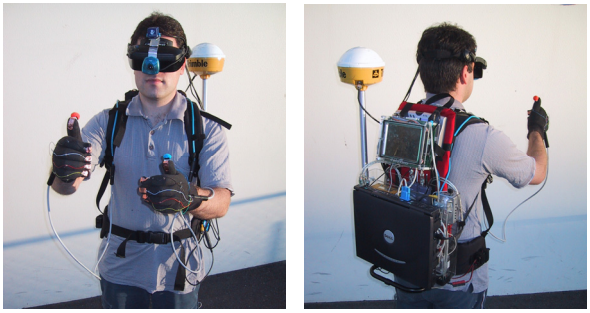

Figure 1 - Tinmith backpack with the RC200 and video overlay devices mounted at the top, and the user wearing gloves with coloured markers attached for the hardware vision tracker

In this paper we describe how we have overcome the drawbacks of the existing tracker in an effort to miniaturise and optimise the overall system. This includes the use of a reconfigurable computer containing a field programmable gate array (FPGA) to perform the hand tracking, and a video overlay device to perform the required AR overlay. By transferring these tasks from the laptop to custom hardware, we can use a slower general purpose microprocessor and less sophisticated 3D graphics chipset which may ultimately result in reduced size and power consumption of the laptop and possibly remove the need for it altogether in the future.

This paper is divided into four main sections. Section 2 discusses technologies and research that are relevant to this paper, including reconfigurable computers with FPGAs, the current problems with outdoor hand tracking, and previous vision tracking techniques. In section 3 we describe how the hand tracker is implemented on the reconfigurable computer, including the selection of the best algorithm, the marker design, and the use of YUV colour spaces to extract out the desired features. Section 4 details the experiments performed into selecting a suitable marker colour and threshold range, and the performance of the tracker outdoors. In section 5, we present how the new hand tracking system was integrated into the current Tinmith modelling software, including the implementation of low powered video AR overlay. Finally, we finish the paper with a summary of the work presented.

2 Background

In this section we will discuss technologies and research that are relevant to this paper, including reconfigurable computers, FPGAs, current problems with outdoor hand tracking, and previous vision tracking techniques.

2.1 Reconfigurable computing

Not all applications can be solved through the use of software. Many real-time applications require algorithmic speedup that only dedicated hardware can provide. Application specific integrated circuits (ASIC) are a well-known category of customised hardware that can provide this algorithmic speedup. ASICs however are not desirable in all situations as they can be very costly for short production runs, require large amounts of engineering expertise to design, can take many months to design and verify, and can not be modified once fabricated (Robles 2003).

Figure 2 - Original ARToolkit marker shown attached to the left glove, and new robust ball-shaped coloured markers shown attached to right glove.

An alternative to an ASIC that does not have these drawbacks but retains a similar algorithmic speedup is a Field Programmable Gate Array (FPGA). An FPGA consists of an array of uncommitted logic and wire resources that can be configured by the end user repeatedly through a form of hardware programming. A reconfigurable computer (Compton 2002) combines an FPGA with other ASICs to provide a platform that retains the flexibility of software but gains the speedup of hardware. The algorithm requiring the hardware speedup is configured onto the FPGA while other ASICs provide support hardware that is not suited to the FPGA architecture; for example, video capture and floating point computation. As the algorithms are performed in custom hardware, there is a possibility for a reduction in power consumption and improved performance as compared to if they were used on a general purpose computer.

Programming hardware circuits for FPGAs involves the use of a hardware description language (HDL). The two traditional HDLs are Very High Speed Integrated Circuits Hardware Description (VHDL) (Ashenden 1990) and Verilog. A new set of HDLs has recently become popular to create hardware circuits however. These HDLs are subsets of common software programming languages such as C (Bazargan 2000), and examples include Handel-C (Celoxica 2003), System-C (Bhasker 2002), and Hardware Join Java (Hopf 2002). An advantage of these languages as compared with the traditional HDLs is the ability to reduce the design time since they use a similar syntax to traditional languages, but extended to support hardware circuits. This was demonstrated by Loo (Loo 2002) as students with limited or no VHDL experience were able to develop hardware applications within weeks.

In the wearable computing and augmented reality domains, the use of FPGAs is still quite rare. This is perhaps due to the extra complexity of implementing hardware in VHDL or Verilog rather than software. Plessel et al (Plessl 2002) described a wearable system that performs simple tasks such as audio and video decoding through the use of reconfigurable modules located on an FPGA. As particular applications are required, the FPGA loads the appropriate hardware module and performs the task in hardware. Luk et al (Luk 1998, Luk 1999) used a reconfigurable computer to support basic functions for augmented reality applications: video mixing, image extraction and object tracking. The image extraction and object tracking stages in this system were performed using a fixed position camera, which significantly reduces the difficultly in performing these tasks. Matsushita et al (Matsushita 2003) described ID Cam, which uses custom hardware and high speed cameras to extract identification codes from flashing beacons in a scene. The camera contained custom silicon to perform most of the high speed extraction, and an FPGA was used to process the final result.

2.2 Outdoor hand tracking

When the Tinmith-Metro modelling system was originally designed (Piekarski 2001) (Piekarski 2003), the user interface was built up around the use of gloves as the input device. The user is able to change the environment using their hands, which is a very intuitive control mechanism. Tracking the position of the user's hands is made particularly difficult when working outdoors, with known problems such as power consumption, size, weight, and support infrastructure. Some examples of these problems are: accelerometers drift over time and provide inadequate registration; infrared based systems are unreliable due to the large amounts of radiation generated by the sun; active magnetic tracking relies on using large non-portable units to generate magnetic fields and are affected by equipment carried on the backpack. One tracking system that has been demonstrated outdoors is the WearTrack system (Foxlin 2000) which adds ultrasonic transmitter and receiver equipment to a wearable computer.

Our original outdoor tracking implementation (Piekarski 2004) employs optically-based vision tracking however. This approach does not require any extra hardware since the user already wears a head mounted camera to provide the video AR overlay for the rest of the system. An additional advantage is that since the user sees the same view as the tracking software, it is possible to achieve accurate registration. The ARToolKit libraries (Kato 1999) are used to perform full six degree of freedom tracking of 2 cm x 2 cm paper fiducial markers placed on the thumbs of the gloves. Only the position values are used, as the accuracy of the rotation values is not adequate. Performing this tracking on the laptop is quite CPU intensive however, and prevents us from using more efficient and smaller equipment.

2.3 Previous vision tracking algorithms

Vision tracking has been a popular field of research for many years and has had a wide range of contributions. Rasmussen et al (Rasmussen 1996) reviewed a number of different tracking techniques including edge detection, region based correlation, and blob tracking. They explain that the tracking of simple blobs is much simpler than other techniques, and we believe that this will assist with its implementation on an FPGA. The authors described how most existing algorithms for blob tracking rely on static or selective colour distribution to segment an image accurately. They defined a custom colour space to assist with an accurate threshold but did not describe other already available colour spaces which might be easier to implement. Brusey et al (Brusey 2000) discussed the problems with recognizing images with colour alone in many colour spaces, and that in all cases there are different objects which are not distinguishable. They present a decision tree approach with a custom colour space that separates brightness, using individual colour channels to make decisions.

There are a number of universities who participate in the RoboCup robot soccer competitions (Bandlow July 1999) (Wang 2001). These competitive events rely heavily on the tracking of coloured markers for estimating the positions of all the robot players. Since the lighting conditions are fixed, many of the competititors perform simple thresholding in RGB space, although other colour spaces are also used. For other applications of these trackers, Jebara et al. (Jebara 1997) implemented a system which allows a user wearing a HMD to visualise predicted ball motions in a game of billiards. A vision tracking system was used to automatically capture the locations of the balls in real time. Cipolla (Cipolla 1993) implemented an indoor vision tracker using motion parallax to estimate the 3D pose of gloves worn by a user. Dorfmuller-Ulhass et al (Dorfmuller-Ulhaas 2001) described the use of blob tracking with retro-reflective markers and an infra-red light to detect the rotations of various joints in the hands. They discuss how they initially used rings around the fingers, but found that with blobs the centres were much easier to locate.

3 FPGA based hand tracking

In this section, we describe how the hand tracker is implemented using a reconfigurable computer, and the modifications made to improve its performance in an outdoor environment. Firstly we discuss what tracking algorithm was used and why. We then outline the colour segmentation step used, followed by blob detection, and finally the FPGA implementation.

3.1 Overview

Many existing tracking techniques are based on one of the following methods: edge extraction, region based correlation or template matching, and segmentation techniques (Dorfmuller-Ulhaas 2001) (Rasmussen 1996). Region-based correlation compares known templates to the video stream to locate markers, but incurs a high computational cost. Edge extraction may be performed at a low cost, however the remaining post-processing slows the overall performance. We have therefore focused on segmentation techniques as they can be implemented in real time on an FPGA the easiest.

Tracking techniques may be further distinguished as having either background or foreground constraints (Dorfmuller-Ulhaas 2001). Background constraint systems rely on a simple uncluttered environment or may perform background subtraction based on a static view. These systems do not apply well to our environment since our background scene is constantly changing with user motion. We therefore threshold with a foreground constraint, whereby coloured balls are attached to the gloves of the user. The colours allow the system to distinguish the balls from the environment, and then blob detection is used to work out the centre of the ball. We use spherical markers because they appear the same from all angles.

Although there are many vision tracking algorithms, not all of them can easily be implemented on an FPGA. The algorithm selected must be suitable for implementation in parallel, and not have a gate count larger that the target FPGA. Some operations such as floating point computation will consume larges amounts of FPGA area and should be avoided.

3.2 Implementation

In this sub-section we will outline how the reconfigurable hardware hand tracking system was implemented. We discuss what implementation language it was written in and why, what platform was used and why, and detail each of the components of the algorithm itself.

3.2.1 Language

Programming hardware circuits for FPGAs traditionally involves the use of a hardware description language (HDL). The two main HDLs are Very High Speed Integrated Circuits Hardware Description Language (VHDL) and Verilog. One of the major problems of these HDLs is the advanced level of hardware knowledge required to use them. A new set of HDLs has recently become popular to create hardware circuits. These HDLs are subsets of common software programming languages and examples of these languages include Handel-C (Celoxica 2003), System-C (Bhasker 2002), and Hardware Join Java (Hopf). The advantage of these languages as compared with the traditional HDLs is the ability for software engineers to use them and concentrate on the specifications of the implementation rather than the code semantics (Rao 2004). This was demonstrated in the paper by Loo (Loo 2002) as students with limited or no VHDL experience were able to develop complex hardware applications in Handel-C within weeks.

Handel-C is a hardware description language based on an extended subset of the standard ANSI-C software programming language. The major advantage of it is there are no intermediate stages and it allows hardware to be directly targeted from software. This provides the necessary features that allow software engineers to easily develop hardware applications. Handel-C comes packaged with the development environment DK1. DK1 does not provide synthesis but is able to produce output files in EDIF format. From this, the traditional place and route tools of the target device can be used to produce bitstreams. A more detailed description on Handel-C is available in the reference manual (Celoxica 2003).

3.3 Platform

The Celoxica RC200 reconfigurable computer was chosen as the target platform for the hand tracking system. This is shown attached to the top of the backpack in Figure 1. The hardware consists of a Xilinx Virtex II 1000 FPGA, 8 Megabytes of external memory, programmable clocks, TFT touch sensitive screen, Ethernet, audio, video out, VGA out, video in, parallel, and RS-232 serial ports. The RC200 is a single board with dimensions of 190mm x 150mm and can run from a 12V power source. The platform contains a Phillips video capture device and provides synchronous streaming of pixels to the FPGA. This particular platform was selected because it has a dense FPGA, supports streaming video, has a serial port for connection to the host, and has an extensive Handel-C application programming interface.

3.4 Algorithm

Our goal with the algorithm was to select a simple reliable algorithm which provides 2DOF tracking capability. There are many different vision tracking algorithms but not all of them can be easily implemented on an FPGA. We have avoided using techniques that require complex floating point calculations in an effort to minimise the area used on the FPGA. We found that segmentation could be performed using very few gates. To provide a unique target that can be tracked by the algorithm, we have decided to use coloured balls mounted onto the thumbs of gloves worn by the user. Figure 2 shows both the original gloves with fiducial markers, as well as the new design which contains a single coloured ball. The marker ball is made of furry material which reduces reflections, and the shape is designed to provide highly robust object detection.

The first step in separating the coloured ball from the image is to threshold the image based on colour. The naïve approach is to use an RGB colour space to nominate a range of colours along each axis. The limitation of RGB is that brightness is encoded into all three channels, and brightness runs diagonally from black to white. Therefore it is not possible to specify certain ranges of colours in RGB that are invariant to brightness, which is important when working in uncontrolled lighting outdoors. Colour spaces are a traditional area of computer graphics (Brusey 2000) (Foley 1990) and there are a wide range available. When encoding colour for hardware such as a television, schemes such as YUV, YIQ, and YCrCb are used. The Y channel represents brightness (luminance) while colour values are stored in the other two chromaticity components. When representing colours in the user-oriented manner often used by artists, colour specification schemes such as HSV and HLS are used. The H channel specifies colour in the form of a single hue angle on a colour wheel, while the other two channels specify brightness and saturation type values. In all of the above colour spaces, brightness is separated from colour and makes it suitable for use in robust colour segmentation. The limitation of the H channel is that it is an angle and requires trigonometric operations or lookup tables to be used.

There are a number of different ways to find the centre of the segmented region, and we have explored the statistical filters median, mode, and mean. The median algorithm implements an array of buckets for every row and column in the image - as pixels are found the matching buckets are incremented. At the end of this process the buckets in the row and column arrays are individually traversed and added up until the total reaches half the total number of hits - when this occurs then the median pixel is found. The mode algorithm implements a similar bucket algorithm as median, except that the row and column buckets with the most hits is selected as the mode coordinate. The mean accumulatively adds up the X and Y locations of the accepted pixels and divides each of these by the number of accepted pixels. We studied the accuracy of each of the filters with a variety of software test cases on a PC and found they all performed reasonably well but with varying failure conditions. We made our

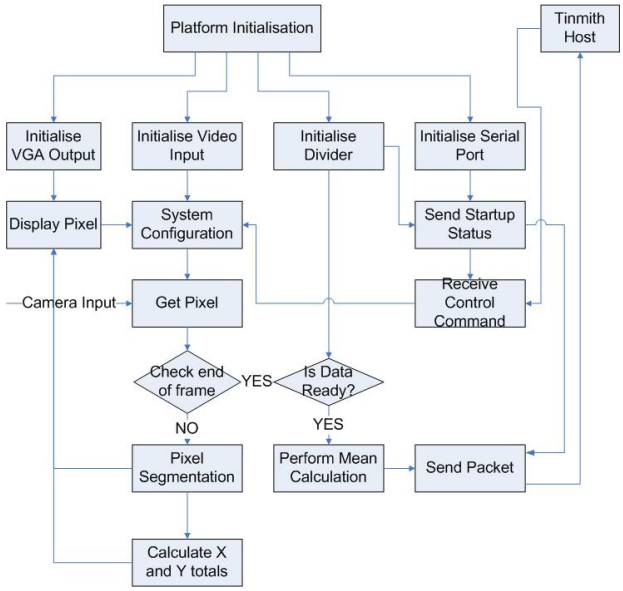

Figure 3 - Flowchart of the parallel hand tracking architecture implemented in hardware

final decision based on which was best suited to implementation in hardware. The median and mode algorithms both require two arrays of buckets to be maintained in memory, and random memory access times on the RC200 are slow in comparison to the time it takes to traverse the pixel array linearly. However the mean algorithm only requires three counters to be maintained with Xtotal, Ytotal and the number of hits; thus we decided mean is the best suited filter to implement.

3.5 Circuit Design

The application we have written consists of four parallel processes: video input, VGA output, mean calculation (division), and RS-232 serial communications, each of which will be discussed in more detail in the following sub-sections. Although the four processes are run in parallel to each other, they are all closely interlinked using control flags for accessing common data values, as depicted in Figure 3. The flow of the system starts with the video stream; a pixel is read and evaluated according to the threshold values. For all the values that are accepted, a running total of the X and Y pixel locations is stored. When the end of the frame is reached the mean calculation process computes the centre of mass of the accepted pixels. The result from this is then sent via the RS-232 serial port and displayed to the TFT display for debugging using a cross to indicate the location of the marker being tracked. The TFT screen is typically used as a debugging device when tuning the tracker outdoors. The pseudo code for our design has been presented in Figure 4, which outlines the flow of the system on a pixel by pixel basis.

3.5.1 Video input

The video input process captures a video stream from the Phillips SAA7113H chip. This provides a synchronous stream of pixels which we evaluate in real time. The pixels can be captured in a range of formats, but we have chosen YCrCb as this separates brightness and colour information which makes it well suited to outdoor tracking. As each pixel is captured, the scan position is used to evaluate what path will be executed in the circuit. The first and most common path is executed when a pixel is read and at this point segmentation is performed to determine if it falls between the threshold values. When a pixel is within the threshold values, the X and Y locations are processed using the decided algorithm until the end of the frame. The second path is executed when the X and Y scan values indicate the end of a frame (X = 720 and Y=576). Finally, a shared control flag is set to indicate the frame is complete and the overall calculation process can begin.

3.5.2 Mean calculation (division)

The purpose of the mean calculation process is to perform division on the results calculated in the section above. The mean calculation process runs as an endless loop, and a control flag is used to signal when new values are ready and the division can be preformed. When this occurs the X and Y totals are divided by the frame hit counter and provides the final average X and Y results. When this step is completed another control flag is set indicating the results are ready for the video output process. Finally, the results are sent to the RS-232 circuit ready for transmission to the Tinmith system.

3.5.3 Video output

The video output process is used to display a picture on the TFT screen and the VGA out of the RC200. We have used the results from the mean calculation to display the location of the blob being tracked, as shown in Figure 5. This video output is not essential for the operation of the algorithm but has proven to be a valuable tool when tuning the different coloured blobs. It also means the RC200 is a stand alone tracker not relying on the accompanying laptop computer to demonstrate its operation.

3.5.4 RS-232 serial communications

The RS-232 serial port is used to send results to the host computer, as well as receive commands used to configure the RC200. When the RC200 is reset, a set of system initialisation packets are sent to the host indicating the status of the RC200. The RC200 then enters normal operation where it reads incoming commands used for system configuration and sends tracking information to the host. Incoming commands include setting the threshold values for tracking of different coloured blobs, setting the camera input port, and running in debugging mode when the

```
While (True)
  For each pixel in the image
    If pixel falls between segmentation ranges
      Add x coordinate to xtotal
      Add y coordinate to ytotal
      Increment hit counter by one
    End If
  End For

  Calculate xmean with xtotal divided by hits
  Calculate ymean with ytotal divided by hits
  Send results to PC via RS-232 port
End While
```

Figure 4 – Pseudocode for the FPGA vision tracking processing algorithm

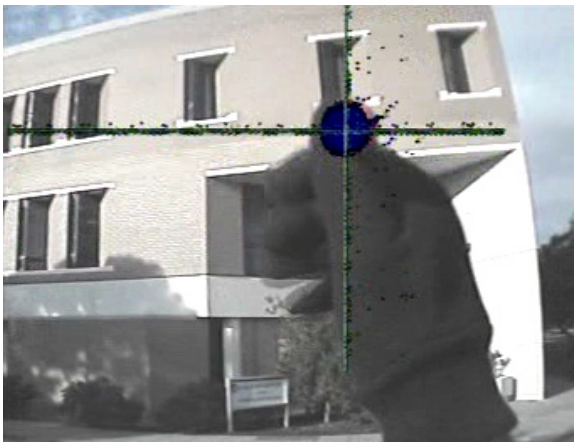

Figure 5 - Output from the RC200 showing image threshold and the calculated marker centre point, combined with the view of the outdoor environment

image segmentation is displayed to the TFT screen. Outgoing tracking data is sent when the mean calculation process indicates a new result is ready. The system sends approximately 25 updates a second, which is the standard PAL refresh rate provided by the Phillips capture chip.

3.6 Performance

Once the hand tracking algorithm was implemented on the FPGA, both the clock speed and device utilisation were recorded. These values were captured from the output files generated from the Xilinx place and route tools. Including the algorithm and a 32 bit divider circuit, the total device utilisation was 83%, or 4263 out of 5120 slices of the FPGA. The theoretical maximum clock speed was 105.27MHz, however the actual clock speed of the system was set to 25.175MHz due to a requirement of the TFT display device.

To determine the actual device utilisation of the hand tracking algorithm itself without the divider circuit (something that could be easily performed on the host), the application was recompiled with it removed. The percentage of FPGA consumed dropped to 23% or 1177 slices. Therefore, the 32-bit divider circuit consumed 3086 slices or 60% of the FPGA.

Our first implementation of the tracking algorithm performed the segmentation on the FPGA and passed values to the laptop for division. This immediately reduces the area used on the FPGA however we then moved the division to the FPGA to make it a stand alone unit supporting our goals of removing the laptop from the wearable computer altogether.

3.7 Power considerations

Since the RC200 is designed for development, it contains many supporting hardware components that are not used but still consume a noticeable amount of power. For example, the optional LCD screen is useful for debugging but the display and backlight contribute to a total consumption of 9 W (750 mA at 12 V) of power and so would be disabled in actual use. During the operation of our tracker without the display, the power used was measured as being 4 W (330 mA at 12 V), which is considerably less. However, this measurement is based on the development board which has many other features we have not used. A more efficient solution could be achieved in a custom circuit board which is comprised of only the FPGA, video capture, and serial UART chips attached. We used the Xilinx XPower tools to estimate the power consumption of the FPGA chip by itself, and the calculated result was 483 mW. This power value is significantly lower than those achieved by the entire RC200 development board. We consulted the data sheet for the Phillips SAA7113H video capture chip which is used in the RC200, and found that its power consumption was rated at less than 500 mW. Therefore, it should be possible to develop a board which contains only a Virtex II FPGA and a Phillips SAA7113H that consumes less than 1 W of power.

4 Outdoor experiments

After the implementation of our algorithm on the FPGA, we tested it outdoors to evaluate its effectiveness. The first step is to pick suitable colours to track, and then test to see how well these perform in actual outdoor environments.

4.1 Suitable colour selection

As was previously mentioned, we require the use of unique colours to separate the marker balls from the background environment. Initially we tried to use small ping pong balls, but found that the surface was too shiny and contained large specular highlights. These specular highlights show up as white on the video camera which exceed its capabilities, and are not visible as any particular colour. To prevent these highlights, we used furry balls which are lit much more evenly, even when used in the brightest direct lighting conditions. A variety of different coloured balls of sizes 1 cm, 2 cm, and 3.5 cm were evaluated to determine which colour would be the most suitable to be distinguished from the background environment.

Our first experiment was to build up a map of the colours that are present in typical outdoor environments on our campus. We used a camera outside (a FujiFilm Finepix 40i) and captured over 50 images of various scenes such as trees, grassy areas, buildings, and signs. We extracted all the RGB values from the images and converted them into YUV space. Figure 6 shows plots in U and V of the presence of different colours in the sampled environments as dark points, with various primary and secondary colours also labelled for reference. Figure 6 also shows another larger and lighter coloured region indicating the possible range that the camera is capable of operating to. Also present is a small streak from the centre of the UV plot, indicating what a green ball looks like when viewed with a black background. From these tests, we learned that saturated colours are not as common as we expected in the environment - saturated colours are present on the outer boundaries of the depicted hexagon, while greys are at the centre. Most colours in the environment are slightly grey, and so very bright and saturated objects should be distinguishable from the background. We should also try

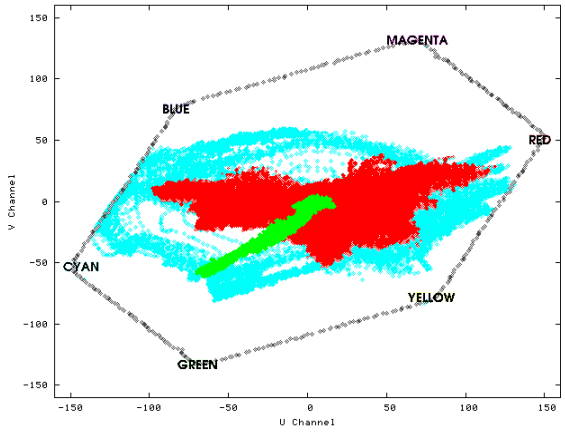

Figure 6 - All colours that are present in a selection of outdoor photos are shown in dark pixels, plotted in UV coordinates. The brighter area depicts the possible operating range of the camera, and the small light streak represents a green marker held in front of a black background

and use regions of the colour space in Figure 6 that are not densely populated with colours from the environment.

4.2 Colour range specification

Our implementation on the RC200 is tuneable via the RS-232 port so that the laptop keyboard can be used to configure the ranges to accept on the YCrCb channels. We tested a range of coloured balls (green, blue, orange, yellow, red, and white) to evaluate which would produce the best results in outdoor environments. To tune colours, the YCrCb filter is opened up to full range (0-255 with 8 bits per channel) and the thresholded image is viewed on the RC200 TFT screen. The minimum (min) value for the Cr channel is then adjusted until just before it begins to filter out the blob. The maximum (max) value is adjusted in the opposite direction until the ball is just accepted. The min and max values are recorded and then opened back up to full range. The same process for min and max is then performed for the Cb channel so that the ball is thresholded. The Y channel is slightly restricted to 10-240 so that it removes out black and white pixels (which contain no real colour information) and then the Cr and Cb ranges are both set to the measured values. We then test these colours against the environment to see if there are matches against any other objects.

During our testing, we noticed that the CCD camera used sees colours in the environment different than our own eyes because the sensor operates over different light wavelengths. When looking at the output, some colours such as blue appeared with a noticeable green colour. When selecting coloured balls to use, the colour the camera sees (and not what the human sees) must be taken into consideration. The following summarises the colours that we tested outdoors - *Green:* When testing the green balls, we noticed that they conflicted with many of the leaves in the trees nearby, but not the grass. Under close inspection the blades of grass actually contain quite a lot of yellow and so there was no conflict. Unfortunately, the amount of noise from the trees was enough to affect the tracker. *Blue:* The blue coloured balls occasionally conflicted with the sky under certain brightness conditions, such as when the camera's auto adjustment darkened the overall image. The blue ball appeared to be a turquoise colour in the camera, but when we tried a more pure blue colour it still conflicted with the sky colour. *Orange:* The orange colour generated the best results with our tracking system. After calibrating it we were not able to find any other objects in the environment of a similar colour that would cause the tracker to operate incorrectly. The only time we could cause a conflict was when looking at a pedestrian crossing sign, which is understandable considering it was a similar shade of orange. *Yellow:* The yellow marker balls experienced slight conflicts with the grass, which as mentioned previously contained large amounts of yellow. This colour would be suitable for use in environments without grass however, such as on concrete or dirt perhaps. Another problem with yellow is that it is similar to orange and it is not possible to separate these two colours from each other with their YCrCb ranges. *Red:* We experienced our second best result with the red coloured balls. There were no conflicts for this colour with the rest of the environment, except for a stop sign on campus. The results were not quite as good as orange, but this could perhaps be improved with further tuning of the YCrCb ranges. Once again, this colour is similar enough to orange that it too prevents them from being used together.

From our experiments, we discovered that trying to provide highly saturated colours for the camera was more difficult than first thought. Even when using highly saturated coloured balls, the camera tends to capture them with a slightly washed out colour which slightly reduces their distinctiveness. We would like to try out new colours, especially those around the cyan or magenta areas because they are the most different from orange.

4.3 Hand tracker results

The purpose of this project was to develop a separate hand tracker using an FPGA to integrate with the Tinmith-Metro software. The RC200 contains an RS-232 serial port and we transmit 10 byte packets to the PC for each update to indicate the X and Y coordinates of the cursor, as well as the number of pixels used in the calculation as a confidence factor. The RC200 captures frames at the PAL refresh rate of 50 Hz, but only provides frames to the FPGA at 25 Hz due to interlacing in the video signals. The RC200 processes frames in real-time and so the results are available within $1/25^{th}$ of a second, although there is additional delay added by transmission across the RS-232 cable and in the host laptop operating system. Figure 5 shows a capture of the thresholded overlay and extracted centre point from the RC200, combined with the view from the camera.

When the 2D cursor is plotted on the display in the Tinmith-Metro software, there is some slight lag with the cursor. This lag is noticeable because the RC200 and the video overlay hardware operate at PAL refresh rates, while the laptop is slightly behind with its processing and rendering of the 3D augmented reality overlay. Previously this effect was not noticeable because the entire output was delayed, but now some parts of the display are faster than others. We could possibly add a delay to the

video stream so that the lag is synchronised, but then this would make the entire system lag from the physical world.

In terms of accuracy, all of the marker sizes tested produced cursors which were within the bounds of the marker, assuming that the signal to noise ratio is relatively high. When smaller markers are used, the signal to noise ratio is reduced, and the accuracy will suffer while still being small enough to point to objects accurately. Of the 1 cm, 2 cm, and 3.5 cm markers tested, we decided to use the 2 cm markers since they are the size of the finger tips.

4.4 Tracker operation and comparison

The user interface in our Tinmith-Metro modelling system is made up of three components: a cursor based on the tracking of the user's thumbs; a command entry system where the user's fingers control menu operations; and an AR HMD which displays information to the user. Each of the user's fingers are mapped to a menu option which is selected by pressing the appropriate finger to the thumb. Figure 7 shows our AR software in operation outdoors, and the menu strips are visible in the bottom left and right corners. This menu system is used to control all aspects of the operation of the system, such as performing 3D modelling tasks. Using the cursor, a number of different manipulation operations are supported, including selection, rotation, translation, scaling, carving, and painting operations. Since the thumbs are used as a cursor, the fingers can still be used to select menu options simultaneously.

The original ARToolkit markers (shown in Figure 2) are a black and white pattern printed on paper and attached to cardboard. Our new markers were chosen specifically for their furry matte surface to minimise specular highlights experienced with the ARToolkit markers. Also they appear to be the same spherical shape from all angles disregarding occlusions. From the user's perspective, both the old and the new trackers are very similar, the only difference is the type of marker attached to the thumb of the glove. In terms of the application and interface to the user however, there is no noticeable difference.

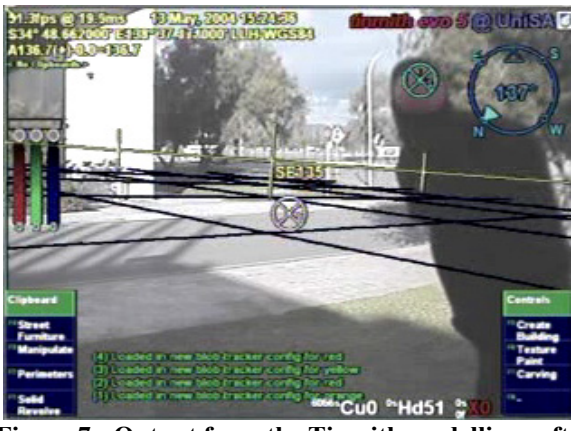

Figure 7 - Output from the Tinmith modelling software, showing the augmented reality overlay. The cursor representing the detected marker location is being drawn using values from the RC200.

In comparison to our previous ARToolKit based tracker, our new tracker appears to be more robust in outdoor environments under most conditions. It is able to more easily survive extreme lighting conditions such as when the sun is almost in the field of view of the camera, which is very common. The tracker's main weakness is operating under twilight conditions when the camera is unable to distinguish colours in the environment as easily, while ARToolKit uses only black and white fiducials. However, the ARToolKit tracker fails in scenes that are half bright and half dark, and with specular highlights on the flat markers.

5 Video overlay implementation

Augmented reality has been traditionally implemented using either optical or video based combination (Rolland 2000). In optical combination AR, a half silvered mirror or prism is used to merge the light from the physical world with the image from an internal LCD or CRT display. In video combination AR, a video camera captures the physical world and this is merged with a virtual image electronically (Bajura 1992). The Tinmith backpack system is configurable to operate in either optical or video overlay mode with a menu option. In our early designs, optical overlay was used to avoid burdening the laptop with video combination tasks. As laptop hardware improved, we added vision tracking of the hands. To implement this tracking, a 1394 Firewire camera was added to supply video to the laptop, since other interfaces such as composite or S-Video were not available. With the video signal being sent to the laptop, using the 3D hardware to also perform video overlay was made possible. We currently prefer video overlay since it makes the system much easier to produce accurate registration and demonstrate to spectators. Optical overlay is much simpler to render however, since the software draws an empty background wherever optical overlay is desired.

With the new developments in this paper, the FPGA is now used to perform the vision tracking. To provide the video AR capability however, the video signal still needs to be passed to the laptop for it to be combined with the overlay image. This would be extremely wasteful of the laptop CPU, and our eventual goal is to try and reduce our reliance on these inefficient devices. To perform video overlay, we now use a GrandTec MagicView device (GrandTec 2004), which is able to genlock a VGA signal with a live video source and combine them based on either a luminance or a colour key. The overall operation of our system is depicted in Figure 8, with the signal from the video camera replacing parts of the VGA image where the pixels are black. The main benefit of the GrandTec MagicView is that it uses only 1.9 W of power (380 mA at 5V) and requires less rendering from the laptop. Improved power savings in the laptop can be made because it only needs to render a black background instead of capturing, processing, and rendering incoming video streams.

The quality of the video signal produced by the video overlay device is excellent for use in AR. The video overlay device operates at the same PAL 50 Hz refresh rate as the video camera, and introduces what we estimate would

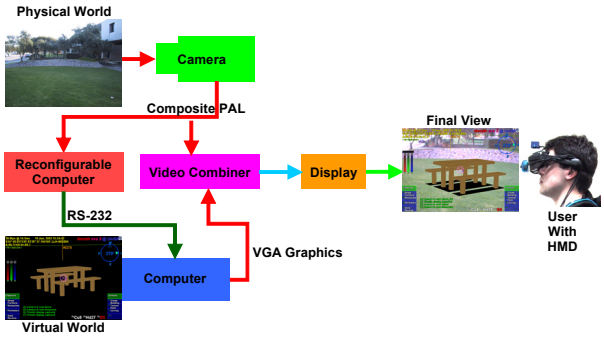

Figure 8 - Overall system schematic, showing the RC200 integrated with the general purpose laptop, and video AR implemented using hardware overlay of the two video signals

be a single frame of delay to synchronise against the VGA signal. The end result is an AR overlay which is just as good as connecting the camera directly to the HMD, without the jitter and pixellation associated with PC based cameras. Since the video overlay device replaces a specific colour in the image, where the overlay meets the video there is some slight bleeding caused by blur in the converted VGA image. When viewing wireframes this is noticeable, but is not a major problem. Using high quality PAL 720x576 signals for display is also quite comparable to SVGA 800x600 signals - our Sony Glasstron PLM-700E display renders PAL with only a slightly noticeable degraded quality compared to SVGA. By using this video overlay device, we are able to obtain all the benefits of video AR while mobile but at the much simpler performance cost of implementing optical AR.

An alternative technique that we are currently exploring is using the FPGA for both vision tracking and video overlay. The RC200 currently only has one capture chip, but it does have the ability to switch between multiple inputs very quickly. We managed to get the RC200 to combine frames from two sources but there were synchronisation problems where after a short period of time the frames would become corrupted. In the future, we hope to implement this properly on the new Celoxica RC300, which provides dual video capture inputs and would therefore provide the desired functionality in hardware.

The hardware tracking system described in this paper has been fully integrated into our latest Tinmith backpack design, which is shown in Figure 1. The existing design contains a Pentium-III 1.2 GHz laptop with Nvidia GeForce2 for rendering the 3D overlay. An InterSense InertiaCube 2 sensor measures the orientation of the user's head, and a Trimble Ag132 GPS measures position within an accuracy of 50 cm. A small PAL resolution CCD colour video camera captures the physical world and the system output is viewed on PAL resolution IO-Glasses. Currently, we have not had a chance to redesign our backpack to take advantage of the power and space savings that are possible. Our next step is to acquire much smaller laptop or embedded PC hardware with reduced functionality, and then build a new design around this. This reduced hardware will still support the same functionality and performance that our previous system provided.

6 Conclusion

In this paper, we have described how we have transferred a CPU intensive vision processing task from our general purpose laptop to a specialised reconfigurable computer. The reconfigurable computer uses a field programmable gate array device to implement the vision processing task in dedicated hardware, both improving performance and reducing the amount of power consumed. We have further improved the system so that video overlay for augmented reality is performed using a specialised hardware overlay device to remove the need for the laptop to ever process video streams at all. With these optimisations the laptop no longer requires a 3D graphics chipset of the same calibre as used previously, allowing the use of smaller and more power efficient laptops in the future. In this paper, we described vision tracking algorithms that are suitable for implementation on an FPGA, and how they can be used to track simple ball shaped markers. The colour thresholding technique that we use is able to operate under a wide range of lighting conditions and deliver robust tracking to our applications. We performed experiments to find out what colours are commonly found in the environment, and then to test colours against the environment to see if they were suitable to use. Based on the results of our work, we have integrated our new vision tracker successfully into the Tinmith-Metro mobile outdoor AR system.

7 References

Ashenden, P. (1990): *The VHDL Cookbook.* 1st ed, Adelaide, University of South Australia, 1990.

Bajura, M., Fuchs, H., and Ohbuchi, R. (1992): Merging Virtual Objects with the Real World: Seeing Ultrasound Imagery Within The Patient. In *Int'l Conference on Computer Graphics and Interactive Techniques,* pp 203-210, Chicago, Il, Aug 1992.

Bandlow, T., Klupsch, M., Hanek, R., and Schmitt, T. (July 1999): *Fast Image Segmentation, Object Recognition and Localization in a RoboCup Scenario.* Jul 1999.

Bazargan, K., Kastner, R., Ogrenci, S., and Sarrafzadeh, M. (2000): A C to Hardware/Software Compiler. In *IEEE Symposium on FPGAs for Custom Computing Machines* Napa Valley, Ca, USA, April 2000.

Bhasker, J. (2002): *A SystemC Primer.* Star Galaxy Publishing, 2002.

Brusey, J. and Padgham, L. (2000): *Techniques for obtaining robust, real-time, colour-based vision for robotics.* Springer-Verlag.

Celoxica (2003): *Handel-C.* http://www.celoxica.com

Cipolla, R., Okamoto, Y., and Kuno, Y. (1993): Robust Structure from Motion using Motion Parallax. In *4th Int'l Conference on Computer Vision,* pp 374-382, Berlin, Germany, May 1993.

Compton, K. and Hauck, S. (2002): *Reconfigurable Computing: A Survey of Systems and Software.* ACM Computing Surveys (CSUR), Vol. 34, No. 2, pp 171-210, 2002.

Dorfmuller-Ulhaas, K. and Schmalstieg, D. (2001): Finger tracking for interaction in augmented environments. In *2nd Int'l Symposium on Augmented Reality,* New York, NY, Oct 2001.

Foley, J., van Dam, S., Feiner, S., and Hughes, J. (1990): *Computer Graphics: Principles and Practice.* 2nd ed, Reading, Ma, Addison-Wesley, 1990.

Foxlin, E. and Harrington, M. (2000): WearTrack: A Self-Referenced Head and Hand Tracker for Wearable Computers and Portable VR. In *4th Int'l Symposium on Wearable Computers,* pp 155-162, Atlanta, Ga, Oct 2000.

GrandTec (2004): *MagicView.* http://www.grandtec.com/magicview.htm

Hopf, J., Itzstein, G., and Kearney, D. Specification of Concurrent Reconfigurable Hardware using Hardware Join Java. In *IEEE Int'l Conference on Field-Programmable Technology,* Hong Kong SAR, China, 2002.

Hopf, J., Itzstein, G., and Kearney, D. (2002): Specification of Concurrent Reconfigurable Hardware using Hardware Join Java. In *Int'l Conference on Field-Programmable Technology,* Hong Kong 2002.

Jebara, T., Eyster, C., Weaver, J., Starner, T., and Pentland, A. (1997): Stochasticks: augmenting the billiards experience with probabilistic vision and wearable computers. In *1st Int'l Symposium on Wearable Computers,* pp 138-145, Cambridge, Ma, Oct 1997.

Kato, H. and Billinghurst, M. (1999): Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *2nd Int'l Workshop on Augmented Reality,* pp 85-94, San Francisco, Ca, Oct 1999.

Loo, S., Wells, B., and Kulick, J. (2002): Handel-C for Rapid Prototyping of VLSI Coprocessors for Real Time Systems. In *Southeastern Symposium on System Theory,* Huntsville, Al, March 2002.

Luk, W. (1998): A Reconfigurable Engine for real-time Video Processing. In *Field Programmable Logic and Applications,* pp 169-178, 1998.

Luk, W., Lee, T., Rice, J., and Cheung, P. (1999): Reconfigurable Computing for Augmented Reality. In *7th IEEE Symposium on Field-Programmable Custom Computing Machines,* pp 136-145, Napa, Ca, 1999.

Matsushita, N., Hihara, D., Ushiro, T., Yoshimura, S., Rekimoto, J., and Yamamoto, Y. (2003): ID CAM: A Smart Camera for Scene Capturing and ID Recognition. In *2nd Int'l Symposium on Mixed and Augmented Reality,* pp 227-236, Tokyo, Japan, Oct 2003.

Piekarski, W., Avery, B., Thomas, B. H., and Malbezin, P. (2004): Integrated Head and Hand Tracking for Indoor and Outdoor Augmented Reality. In *IEEE Virtual Reality Conference,* Chicago, Il, Mar 2004.

Piekarski, W. and Thomas, B. H. (2001): Tinmith-Metro: New Outdoor Techniques for Creating City Models with an Augmented Reality Wearable Computer. In *5th Int'l Symposium on Wearable Computers,* pp 31-38, Zurich, Switzerland, Oct 2001.

Piekarski, W. and Thomas, B. H. (2003): Interactive Augmented Reality Techniques for Construction at a Distance of 3D Geometry. In *7th Int'l Workshop on Immersive Projection Technology / 9th Eurographics Workshop on Virtual Environments,* Zurich, Switzerland, May 2003.

Plessl, C., Enzler, R., Walder, H., Beutel, J., Platzner, M., and Thiele, L. (2002): Reconfigurable Hardware in Wearable Compting Nodes. In *6th Int'l Symposium on Wearable Computers,* pp 215-222, Seattle, Wa, 2002.

Rao, D. and Venkatesan, M. (2004): An Efficient Reconfigurable Architecture and Implmentation of Edge Detection Algorithm using Handel-C. In *ITCC04,* pp 846, Las Vegas, NV, 2004.

Rasmussen, C., Toyama, K., and Hager, G. (1996): *Tracking Objects By Color Alone.* New Haven, CT, Yale University - Department of Computer Science, Jun 1996.

Robles, R. (2003): *ASIC Versus Reconfigurable Compute Fabric (RCF) Solutions.* Denver, Co, Motorola, pp 8, March 2003.

Rolland, J. P. and Fuchs, H. (2000): *Optical Versus Video See-Through Head-Mounted Displays in Medical Visualization.* Presence: Teleoperators and Virtual Environments, Vol. 9, No. 3, pp 287-309, 2000.

Wang, C., Wang, H., Soh, W. Y. C., and Wang, H. (2001): A Real Time Vision System for Robotic Soccer. In *4th Asian Conference on Robots and its Applications,* Singapore, Jun 2001.