Markerless Augmented Reality based Cameras using System-on-Chip Technology

Veronica Teichrieb, João Marcelo X. N. Teixeira João Paulo S. do M. Lima, Judith Kelner
Federal University of Pernambuco – UFPE, Computer Science Center – CIn
Recife, Pernambuco, Brazil
vt@cin.ufpe.br, jmxnt@cin.ufpe.br jpsml@cin.ufpe.br, jk@cin.ufpe.br

ABSTRACT

Markerless augmented reality (MAR) aims to integrate 3D virtual objects into the real environment in real-time, enhancing the user's perception and interaction. It differs from marker based augmented reality (AR) systems basically by the method used to place virtual objects in the real world. In MAR, the real environment may be used as a marker that can be tracked in order to position virtual objects. Generally, the techniques used are based on computer vision, image processing, and computer graphics, and a major issue related to the field is robust, yet precise real-time object tracking and registration. Several AR solutions using general purpose devices have been developed. Such processing is done by software, making it difficult to obtain real-time results without compromising resolution and frame rate, and requiring the use of high clock frequencies that consequently lead to higher costs and power consumption. This paper introduces MARCam, an FPGA based solution that allows the development of embedded MAR applications. This framework allows the development of compact hardware with wearable capabilities for applications requiring user mobility through unknown environments and real-time dedicated processing. One of the most promising applications of this technology is guidance in training systems. MARCam can utilize a Structure from Motion (SfM) based technique. Instead of relying on previously obtained information about the real scene, SfM based techniques estimate the camera displacement without a priori knowledge of the environment. These methods are also able to retrieve the structure of the scene in real-time, with various levels of detail. Due to this, it is possible to reconstruct a totally unknown environment on the fly. MARCam's architecture is divided into many circuit modules, each one responsible for a specific task. This way, one can quickly arrange an assembly of modules and have in short time a fully-working, dedicated MAR system.

ABOUT THE AUTHORS

Veronica Teichrieb is a Visiting Researcher at the UFPE's CIn, where she co-leads the Virtual Reality and Multimedia Research Group (GRVM). Current GRVM research areas include augmented reality, graphics and physics modeling and simulation, and 3D interaction. Veronica earned her master's degree in Computer Science from the UFPE (1999), and a Ph.D. in Computer Science from UFPE in 2004. In 2001, she was a visiting researcher at the Aero-Sensing Radarsysteme GmbH, Germany, during a doctoral sandwich-program.

João Marcelo X. N. Teixeira is following a M.S. degree course in Computer Science at the CIn of the UFPE, in Pernambuco – Brazil, since 2007. Currently he is a researcher in the GRVM of the UFPE.

João Paulo S. do M. Lima is following a degree course in Computer Science at the CIn of the UFPE, in Pernambuco – Brazil, since 2003. Currently he is a researcher in the GRVM of the UFPE interested in virtual reality and augmented reality.

Judith Kelner is an Associate Lecturer at the UFPE's CIn, where she leads the GRVM. Judith earned her master's degree in Computer Science from the UFPE (1981), and a Ph.D. in Computer Science from University of Kent at Canterbury in 1993.

Markerless Augmented Reality based Cameras using System-on-Chip Technology

Veronica Teichrieb, João Marcelo X. N. Teixeira **João Paulo S. do M. Lima, Judith Kelner**
Federal University of Pernambuco – UFPE, Computer Science Center – CIn
Recife, Pernambuco, Brazil
vt@cin.ufpe.br, jmxnt@cin.ufpe.br **jpsml@cin.ufpe.br, jk@cin.ufpe.br**

INTRODUCTION

Markerless Augmented Reality (MAR) superimposes virtual information – 2D or 3D, textual or pictorial – onto real world scenes. Nowadays, Augmented Reality (AR) applications are used in different fields, such as entertainment, medicine, manufacturing and repair, and education. The technical challenges lie in determining, in real-time, what should be shown where, and how. The latter problem is especially important when the visual appeal of the result is crucial. Then substantial effort must go into seamlessly fitting the information into the scene, both geometrically and photometrically (occlusions, shadowing, mutual reflections, chromatic adaptation to scene illumination, and so on). Even under simplified conditions these problems are not trivial. The system must recognize where it is, which information is to be superimposed, and where. Beyond that, it should present realistic interaction between virtual and real elements.

In this paper we deal with the two first issues, that mean what to place and where to place it, with the goal of achieving acceptable processing performance, user autonomy, and system mobility. The concept of MARCam, which is introduced in this paper, is a smart camera built upon the System-on-Chip (SoC) technology. It comprises a dedicated hardware containing an entire MAR pipeline implemented as a set of hardware components (cores). The use of dedicated hardware allows overall better performance compared to software implementations, and with the potential of exploiting real parallelism of concurrent tasks. These systems also offer the advantages of low power consumption while still handling high resolution images.

Our present prototype uses a Field Programmable Gate Array (FPGA) for image acquisition, image processing and scene display, in order to develop embedded AR applications. Current results in some cases show a performance gain ~30,000 times faster than comparable functions implemented in software alone. After performing some successful validation case studies, we have started to implement MAR components.

This paper is organized as follows. The section "Background" describes some basic concepts regarding to MAR and SoC technologies. The section "Related Work" presents current major research in this topic. The section "MARCam" introduces our MAR camera prototype, and the "Results" section offers detailed information regarding results already obtained and ongoing implementations. The section "Potential Application Scenario" draws a scenario using MARCam for guidance and training. Finally, some concluding remarks are discussed in section "Conclusions and Future Directions".

BACKGROUND

This section describes the basic concepts involved in the two main topics comprising this work: Markerless Augmented Reality and System-on-Chip. Special attention is given to Structure from Motion (SfM) based MAR techniques, as they are the most adequate approach for MARCam purposes.

Markerless Augmented Reality

AR systems integrate virtual objects into a 3D real environment in real-time, enhancing the user's perception of, and interaction with, the real world (Bimber & Raskar, 2005). Its basic difference from MAR systems is the method used to place virtual objects in the real world. The Markerless approach is not based on the use of traditional artificial markers. Such markers are placed in the real world to support position and orientation tracking by the system. In MAR any part of the real environment is used as a marker that can be tracked in order to position and orient virtual objects. Therefore, there are no ambient intrusive markers that are not really part of the environment. MAR counts on robust trackers to accomplish this (Comport, Marchand, Pressigout, & Chaumette, 2006). Another advantage is the possibility of extracting from the environment characteristic information that may later be used by the MAR system.

Nonetheless, tracking and registration techniques are more complex in MAR systems. Another disadvantage emerges in online MAR since it presents more restrictions.

Techniques developed for MAR can be classified in two major types: Model based and SfM based (Figure 1). With model based techniques, knowledge about the real world is obtained before tracking occurs and is stored in a 3D model that is used for estimating camera pose. In SfM based approaches, camera movement throughout the frames is estimated without any previous knowledge about the scene, but is acquired during tracking.

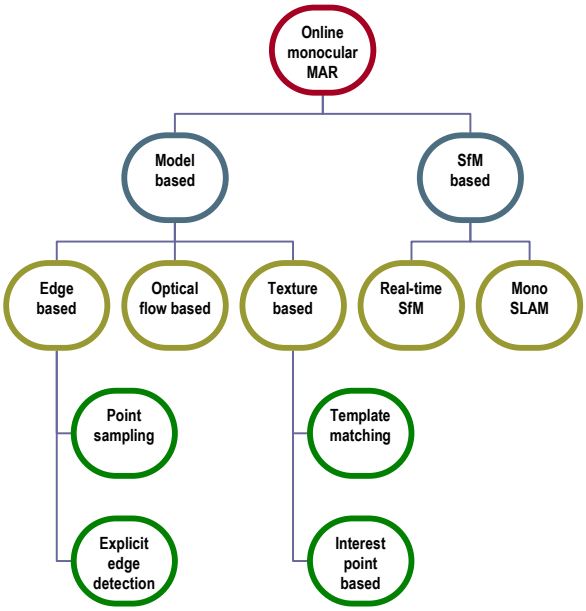

Figure 1. Online Monocular MAR Taxonomy

SfM is a classic technique used in computer vision to perform 3D reconstruction (Pollefeys, 1999). Its traditional implementation follows a suggested pipeline, and is not concerned with real-time constraints. SfM produces great results relative to the final mesh generated by the entire process, but some algorithms present in its pipeline require a lot of processing time to finish their work, and are thus unsuitable for real-time applications. Usually, the SfM pipeline is composed of the following phases: feature tracking, camera pose hypothesis generation, pose evaluation and refinement, self-calibration and 3D reconstruction.

Basically, in order for SfM to support real-time constraints, some of these phases have to be skipped or replaced by other algorithms that still maintain the robustness of this technique. An overview of the real-time SfM method is depicted in Figure 2. In Nistér's implementation of real-time SfM, he introduced some modifications to the pipeline relative to camera pose hypothesis generation, evaluation and refinement, creating a brand new solution based on the classic Random Sample Consensus (RANSAC) refinement algorithm (Nistér, 2005). He used this new algorithm for acquiring a sufficiently good pose estimate that could be fed into a bundle adjustment procedure, such as the Levenberg-Marquadt one (Triggs, McLauchlan, Hartley, & Fitzgibbon, 2000), in order to generate a precise resulting pose. This algorithm works similar to original RANSAC, but in a preemptive way, stopping evaluation of pose hypothesis that are not promising. Since Nistér's approach works with a calibrated camera, the self-calibration step is not performed and camera pose hypothesis generation is done with the five-point (Nistér, 2004) and three-point (Haralick, Lee, Ottenberg, & Nölle, 1994) methods. These methods consist in solving a linear equation, considering the number of degrees of freedom given by the metric reconstruction. Therefore, to compute the camera translation and orientation, only five or three pairs of correspondent points from two consecutive frames are used.

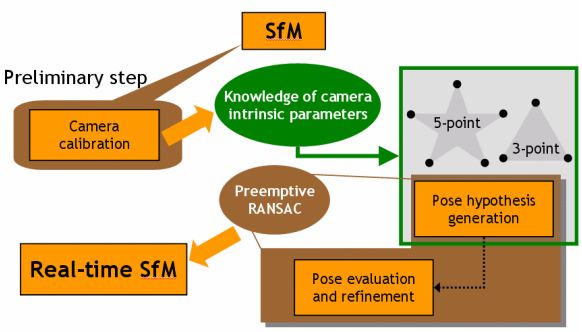

Figure 2. Real-Time SfM Overview

These modifications to the original SfM pipeline removed some bottlenecks and speeded up the entire process. This minimizes the delay in reconstructing a rigid scene, getting closer to real-time 3D reconstruction. Since the real-time constraint is supported by SfM, it has been used in MAR systems (Lourakis & Argyros, 2005). Real-time SfM can offer more information about the entire scene, and may provide data to improve the MAR system. Features like occlusion of virtual objects by real ones and physical-based interaction between them can be exploited.

System-on-Chip

In the area of embedded systems there is a novel technology that aims to satisfy the desire of every hardware engineer: reduced cost-per-function, increased system performance, longer battery life, higher-end features in lower-cost systems, smaller form factors to enable designs with lower temperatures, easier design and manufacturing, and increased reliability. System-on-Chip, or simply SoC is a reasonable attempt at all of these goals. The main idea is that it integrates a complete system, created as a set of components (intellectual properties, or IP cores, as they are known), or sub-systems, on a single piece of silicon. In the past, a user had to utilize many devices for obtaining the desired result. Currently, the goal is to reach the same, or an even better result, with the use of a single device.

As requisites, SoC technology presents three basic demands. The first one is a common process for handling analog, digital and radio frequency signals, so that various systems can communicate with each other. The second requisite is that there must be a simple interface for connecting the assembled components, following common busing techniques. At last, there is a need for end-equipment systems expertise, since application domain knowledge is applied directly to the development environment, for generating the SoCs.

RELATED WORK

As far as we know, there is no flexible solution considering the use of a hybrid hardware and software platform to develop marker based AR applications, neither a complete hardware based one. Indeed, our research did not find any AR embedded system that exploits the Markerless approach. Most existing AR solutions are still not accessible for the general audience, because they are still in research phase and/or dedicated to a specific application domain (Umlauf, Piringer, Reitmayr, & Schmalstieg, 2002), (Wagner, Pintaric, Ledermann, & Schmalstieg, 2005), (Matsushita et al., 2003). For instance, ID CAM (Matsushita et al., 2003) is an ID recognition system with an optical beacon and a fast image sensor with sufficient space resolution and robustness for long-distance recognition. The ID CAM contains an optical lens, a fast CMOS (Complementary Metal–Oxide–Semiconductor) image sensor, a controlling FPGA, and an USB interface to output scene images and the IDs to a computer.

In (Toledo, Martinez, Garrigos, & Ferrandez, 2005), a fully FPGA based AR application is developed for visual impaired individuals affected by tunnel vision. A cellular neural network extracts the contour information and superimposes it on the patient's view. This work, however, performs very simple image processing operations when compared to our developed platform.

Regarding MAR, there are already some important contributions related to interest point based techniques (Vacchetti, Lepetit, & Fua, 2004) and tracking of corners and edges (Fung & Mann, 2005) implemented for the Graphics Processing Unit (GPU). In (Sinha, Frahm, Pollefeys, & Genc, 2006), there is an implementation of Scale Invariant Feature Transform (SIFT) and Kanade Lucas Tomasi (KLT) feature selection algorithms for GPU.

AR has been used for training and guidance purposes. One example is the Primordial Soldier, by Primordial (2007). This system uses Global Positioning System (GPS) and Geographical Information System (GIS) technologies for soldier tracking and battlefield georeferencing, and AR based interfaces that guide soldiers during rescue missions and assist them during training how to act at the battlefield against the enemy.

MARCam contribution relies on the use of a completely hardware based infrastructure for the development of MAR systems targeting different application domains, in which training and guidance are some of the most promissing ones.

MARCAM

This section introduces MARCam, giving detailed information regarding its architecture.

System Overview

MARCam is a platform that comprises a dedicated hardware containing an entire AR pipeline implemented as a set of components. Figure 3 shows the development environment used in the project, where an image sensor (lower right corner) was connected to an FPGA-based development board.

MARCam overview is illustrated in Figure 4. It exploits the use of FPGA in order to develop AR applications, covering three distinct steps: image acquisition, image processing and scene exhibition.

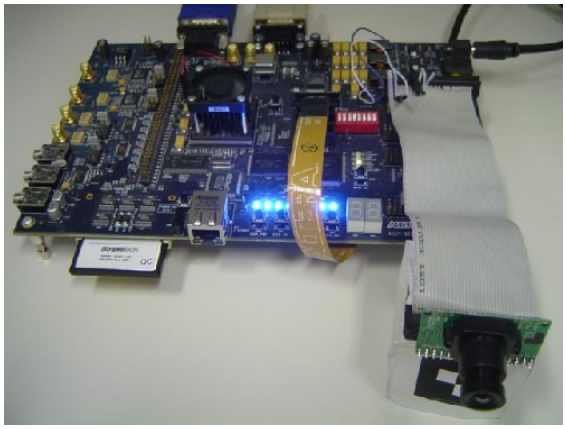

Figure 3. MARCam Development Environment

Each implemented function is encapsulated in a component that defines all the inputs and outputs needed by the algorithms. The component-oriented approach adopted brings the opportunity to choose from a components library only the necessary modules required by the AR application project, which can be connected together to build a complete system.

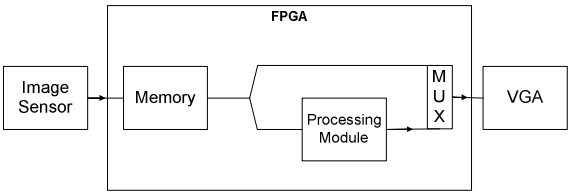

Figure 4. MARCam Overview

Architecture

A typical AR system should run at least at 15 frames per second (fps) with image resolution of $320x240$ pixels. MARCam is designed to fulfill these requirements.

The proposed architecture consists of an image sensor that acquires images from the environment, a VGA monitor that displays the enhanced visualization, and an FPGA as processor, controller and storage unit. The architecture design is shown in Figure 5.

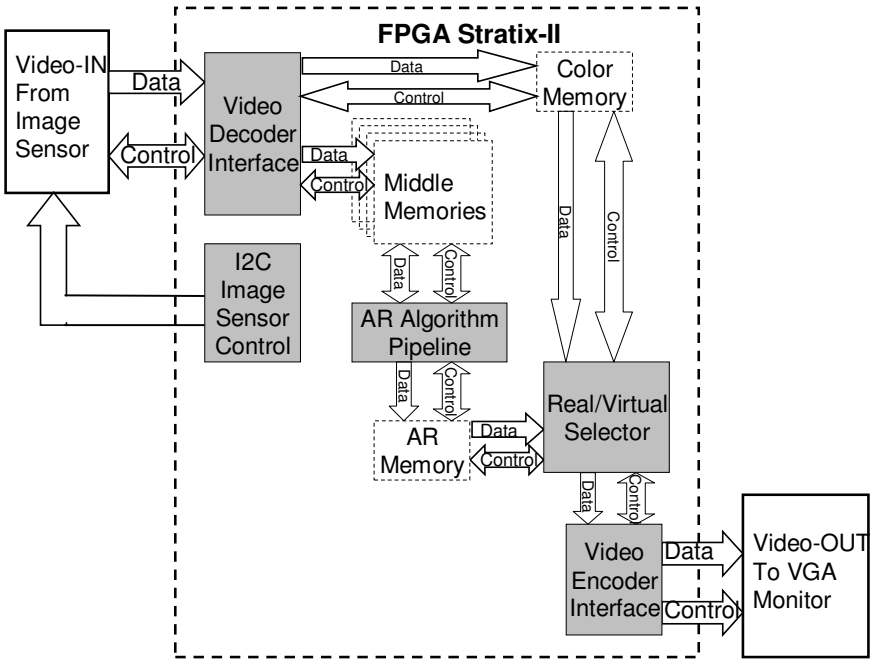

Figure 5. MARCam Architecture Block Diagram

The Video-IN block converts analog video input to a digital RGB format. The Omnivison's CMOS circuit OV7620 is a single-chip video/imaging camera designed to provide a high level of functionality in a single package. This device incorporates a $640x480$ image array operating at up to 30 fps. The process unit is an Altera Stratix II FPGA with 48,352 Adaptative Look-Up Tables - ALUTs (equivalent to 60,440 Logic

Elements - LEs), 2,544,192 bits of RAM memory, 36 DSP blocks and 144 multipliers. The Video-OUT block converts the digital RGB format to analog video output for VGA monitors. The VGA interface is a Triple Video D/A converter $3x8$ bits at 180 megapixels per second.

Since the FPGA internal memory blocks impose a severe storage limitation, the input resolution of $320x240$ was chosen and later converted to a $640x480$ video output. This limitation is due to temporary prototype issues we have while accessing external memory; at the moment MARCam uses only internal memory.

The basic infrastructure of our architecture is represented by the following cores. Their use is mandatory in every application.

- *I2C Image Sensor Control*: this block uses the I2C protocol to control all required camera functions, including: exposure control, gamma, gain, white balance, color matrix, color saturation, hue control, and windowing, among others.
- *Video Decoder Interface*: receives image sensor signals and controls the storage of acquired images in the color memory. The image sensor's output is basically composed by three electric signals: vertical, horizontal and pixels synchronism. They indicate, respectively, when a frame and a line finish, and when a pixel is available at the bus.
- *Color Memory*: stores a $320x240$ real world image in RGB666 format. Stratix II devices have three sizes of embedded RAM blocks. This color memory uses all of the 2 M-RAM blocks (64K x 18) to print the first 200 lines of the frame, and uses also 50 M4K blocks (12.8K x 18) to print the last 40 lines of the frame. This totals 58% of the FPGA's embedded RAM blocks.
- *Middle Memory*: intermediary memories needed to process AR algorithms. The number of middle memories and the word length depends on the algorithm used.
- *AR Memory*: stores pixels resulting from AR algorithms.
- *AR Algorithm Pipeline*: implements AR tasks. This module corresponds to the user application and it is developed according to the desired functionality.
- *Real/Virtual Selector*: based on AR Memory pixels, this core functions like a multiplexer and decides which value will be sent to the Video-OUT block.
- *Video Encoder Interface*: sends digital RGB signals (real world image or virtual objects pixels) and VGA control signals (vertical or horizontal synchronism, for example) to the Video-OUT block. At this stage, our system's output is VGA, so that the board can be connected to both a Head Mounted Display (HMD), or a common video monitor. At a second stage, we plan to include an embedded Liquid Crystal Display (LCD).

Implementation flexibility provided by a hardware description language, such as Very High Speed Integrated Circuits Hardware Description Language (VHDL), makes scalability possible every time it is necessary to duplicate a component in order to improve processing performance.

RESULTS

Since MARCam intends to be a framework for enabling the development of embedded AR systems, it must provide a set of modules representing the functions commonly used by these applications. After specifying the framework architecture, we divided the development of its modules in three basic steps: 1) software implementation, 2) hardware translation, and 3) hardware optimizations. At first, the functionality is implemented in software and some tests are performed to verify its correctness. The following process consists in transforming the software source code to a hardware description language, like VHDL, using a machine state approach. Lastly, some optimizations regarding hardware specific characteristics are performed, in order to take advantage of the proximity to the hardware, like real parallelism, for example (Keating & Bricaud, 2002).

Implemented Components

For the initial development of the MARCam infrastructure, a number of hardware components used for marker-based AR applications were developed, since they are less complex than MAR ones. They were useful for ensuring the viability of the proposed embedded AR development platform. In addition, the same infrastructure can be exploited by hardware based MAR applications.

Several image processing components with different purposes were implemented in this infrastructure. These components perform typical image processing functions and they are intended to be used for designing AR applications. Currently we have implemented

components with the following functionalities: binarization, gray scaling, labeling, mean filtering, edge detection, generic convolution, centroid estimation, square detection and 3D object wireframe renderization (named Hardwire).

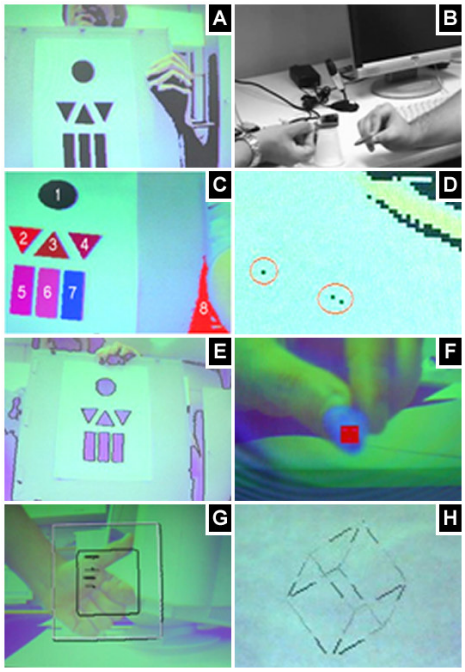

Figure 6. Implemented Components: (a) Binarization; (b) Gray Scaling; (c) Labeling; (d) Mean Filtering; (e) Edge Detection; (f) Centroid Estimation; (g) QuadDetector; (h) Hardwire

Most image processing algorithms applied in AR do not handle color images. As a result, the original image has to be converted to a more suitable format, such as gray scale or binary (Fiala, 2004). Figure 6 (a) shows the binarization results with the white color representing the real world scene and black corresponding to pixels above the threshold. Hence, when a white pixel is found in memory, the real world scene pixel is rendered on the screen. Figure 6 (b) illustrates the gray scaling results.

Another implemented function is the labeling algorithm. Binary image labeling refers to the act of assigning a unique value to pixels belonging to the same connected region (Gonzalez & Woods, 1992). This algorithm is often exploited in marker based AR applications in order to segment marker regions for latter recognition (Fiala, 2004). Figure 6 (c) displays the result of this process, in which the numbers represent the same connected regions.

Another implemented function is the mean filter. This replaces each pixel value in an image by the mean value of its neighbors, including itself (Gonzalez & Woods, 1992). It is often used to reduce noise in images, eliminating pixel values that are unrepresentative of their surroundings. Figure 6 (d) illustrates some image noise that can be removed by this filter.

In a digital image, points that are affected by sudden luminous intensity changes are identified by edge detection algorithms (Gonzalez & Woods, 1992). Usually these changes suggest points of interest in real world captured images that can represent depth discontinuity, surface orientation discontinuity, material property changes, and scene illumination variations. AR solutions widely use edge detection in tasks such as marker tracking and feature matching (Fiala, 2004). Figure 6 (e) shows the outcome of this process, in which all detected borders are contoured in black.

Applying a convolution mask on an image is the process of changing the pixel value to other values based on the neighbors of the pixel. In order to use any convolution kernel, we included a block where convolution mask values can be easily changed. $3x3$ convolution masks are often used in image processing.

A function that finds the center of a colored object was also implemented. The purpose of this centroid estimator component is finding the center of a region (x and y coordinates) containing more pixels regarding to a specific color (blue, in the example of Figure 6 (f)).

Actually, marker recognition is widely used by AR applications, and most of these markers have a quadrangle shape (Fiala, 2004). The QuadDetector component is responsible for detecting squares in real-time on images captured by a camera. Figure 6 (g) illustrates the traced border (in black), together with the detected square (in white).

Lastly, the Hardwire component renders 3D wireframe objects. This process applies a series of coordinate transformations (world to view and projection transforms) and then displays these objects on screen, as shown in Figure 6 (h), using Bresenham's line drawing algorithm (Foley, Van Dam, Feiner, & Hughes, 2005). Hardwire's purpose is to provide fast prototyping object visualization so that the developer can validate his/her AR system.

Performance Analysis

The VHDL hardware description language was used for modeling the MARCam architecture. The Quartus II tool from Altera was used for place and route, and for device programming. The selected target technology was the Stratix II from Altera (EP2S60F1020C4). This device has 719 GPIO (General-Purpose I/O) pins and MARCam uses only 57 pins (8%) for video input and output.

In order to evaluate the prototype performance in hardware, an equivalent implementation for each algorithm was developed in software, using the C programming language. Therefore, analysis could be done by comparing each algorithm's performance gain on embedded hardware versus general purpose processors.

The entire evaluation process used a unique image to test all modules, in order to keep processing time precise (for example, edge detection time varies if the input image changes). A tool was developed for converting PNG images to a VHDL module that could be used as the input image on hardware tests. If the input image is not changed, the hardware module always performs its task in a fixed time. Because of that, the hardware processing time acquisition was realized by running a single test.

Since the software simulation runs on the top of an operating system, there is no absolute control over the processing time. Thus, a statistical approach is necessary to evaluate the software tests. First, a selection of samples from each simulation was gathered, using equation 1, in order to find the ideal number of samples according to the simulation parameters. This formula is based on the confidence level and the simulation precision desired (Jain, 1991).

$$n = \left(\frac{100 \times Z \times S}{r \times \bar{x}} \right)^2 \quad (1)$$

Where:

n	ideal number of samples
Z	value found on normal distribution table
S	standard deviation of samples
r	sample precision
x	sample average

Next, a new selection of samples was gathered, based on the outcome obtained through the statistical method, and then analyzed. Table 1 shows the time results got on both, software and hardware targets.

Table 1. HW/SW Performance Comparison

	Time duration (10^{-6}s)		Ratio
Process	**Software (2.01GHz)**	**Hardware (100MHz)**	**sw/hw (100MHz)**
Binarization	721.99	768.00	18.89
Gray Scale	615.49	768.00	16.10
3*x*3 Filter	12,789.08	8.44	30,428.57
Mean Filter	468.36	5.37	1,751.15
Edge Detection	447.29	4.60	1,951.06
Labeling	638.19	3.54	3,623.64
Centroid	201.18	768.00	5.26
QuadDetector	646.56	470.23	27.63

It is important to observe the difference between targets' clock frequencies. The computer used on software tests was an AMD Athlon 64 3200+, with a 2.01GHz processor and 1GB of RAM memory. The operating system running was the Windows XP Professional Version 2002, Service Pack 2, and the C language development IDE was Microsoft Visual Studio .NET 2005 Professional Edition. The clock frequency used on the prototyping board was 100MHz.

The fourth column of Table 1 shows the ratio between the clock counts obtained from both software and hardware tests. Considering that the same clock frequency is used (100MHz, for example), the worst hardware result acquired is 5 times better than the software's performance (object's centroid estimation time). Because of the number of memory accesses required, the 3*x*3 Filter module presented the highest gain in performance under the hardware implementation (~30 thousand times more effective).

Case Studies

Two AR proofs of concept were implemented using the MARCam platform, in order to evaluate its feasibility. Due to this, an AR application prototype (the well-known Pong game) was quickly developed without taking too much modularization into account. The second case study (an object recognition demo) connects some of the existing core components to build another application, showing that it is possible to have a modularized model for the design of hardware based AR systems.

The Pong game was implemented as the first case study. The game is based on a ricocheting ball that is prevented from colliding with the side edges of the screen by user-controlled "paddles" near the left and right screen edges. If a player is not able to prevent this

collision, a point is lost and the screen edges blink in red, announcing the collision.
Figure 7 illustrates the moment that a collision happened.

Figure 7. Pong Game Illustrating a Collision Detection

The object recognition application identifies blue objects in the real environment and draws a rotating cube over them on screen. Basically, two function components were used to achieve this result: Centroid Estimation and Hardwire.

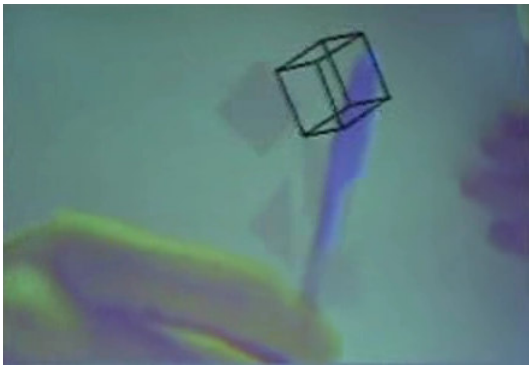

Figure 8. Object Centroid Estimation and Cube Rendering

The application pipeline works as follows: the image captured by the camera is sent to the centroid estimation component, where the x and y coordinates relative to the object position are determined. This information is then sent to the Hardwire component, and the wireframe cube is rendered at the supplied position over the real world image. The result of this process, shown in Figure 8, is then presented to the user.

Ongoing Components

There are two additional MAR modules currently under development: a SfM module and a model based one. Both are concerned to the first development stage: the software implementation (as mentioned in section Results).

The chosen model-based technique is an edge-based one. The main purpose of the edge-based module, which is based on the RAPiD's algorithm (Lepetit & Fua, 2006), is providing a Markerless tracker capable of determining the camera pose based on the correspondence between a projected object CAD model and its real image. The algorithm samples a small amount (about 10%) of object visible edge points and tries to find their matching positions in the input image. Figure 9 illustrates the matching of the samples. The green edges refer to the real image and the red ones to the object CAD model.

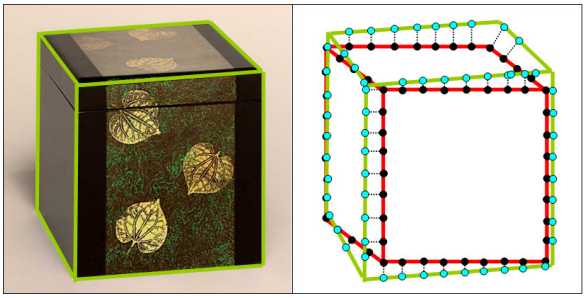

Figure 9. Edge Based Tracking Point Matching Example

Next, it assumes that the camera pose differs only by a small translation and a small rotation from the predicted position in 3D. The projection of the object onto the image plane is linearised and the camera pose correction is found using standard linear algebra techniques. All these calculations are performed using OpenCV library (Bradski & Pisarevsky, 2000).

Based on the test results using the modules implemented, we have assured the feasibility of the edge-based tracker module. This module requires a relatively small amount of memory for CAD model storage and basically a sequence of matrix multiplications. Once the software version is fully implemented, its hardware counterpart can be developed without significant effort.

The implementation of the SfM based technique is currently focusing on the first step in the SfM pipeline: feature matching. Feature matching can be divided into two subtasks: feature selection and feature tracking. Feature selection is the extraction of interest points from the real scene. Feature tracking is the following of interest points positions in subsequent video frames. Feature selection is accomplished using the Good Features to Track algorithm (Shi & Tomasi, 1994), while feature tracking is done using the Kanade Lucas optical flow estimator (Lucas & Kanade, 1981).

The analysis of existing implementations of the algorithms mentioned above shows that the amount of memory needed for implementing them in hardware is available at MARCam infrastructure. Furthermore, the involved image processing tasks can be developed using only plain mathematical functions, such as eigenvalues and gradients.

POTENTIAL APPLICATION SCENARIO

MARCam may be prototyped as a compact, lightweight and autonomous camera dedicated to processing marker based and Markerless AR mobile applications. Therefore, it may be adequately used for user AR based training and guidance in wide range environments, principally outdoor. Next we present an example that illustrates the use of MARCam technology in the context of a battlefield training application.

The main idea is of a soldier in a battlefield interacting with an unknown environment and making decisions dinamically. During training the soldier has to deal with virtual enemies positioned in the environment according to the reconstructed world through real-time SfM technique. Since SfM retrieves users' position, he/she does not require a GPS to track his/her position, neither the environment needs to be georeferenced previously. There is only a requirement for a tracker responsible for gathering the weapon's position and orientation, such as a 3D pointing device. In addition we can have a reconstruction of the environment that allows the application to deal with real-virtual object interactions.

Figure 10 illustrates a scenario where a user shoots a virtual enemy (highlighted in red) positioned randomly in the scene. The system is able to know if he/she hit the target. It is important to mention that the system is able to detect collisions between the virtual bullet and real obstacles. In case he/she misses, the AR interface shows an orange mark indicating the position where the virtual bullet hit, allowing further user error situations verification. Based on these marks it is possible to perceive the hit/miss rate on the user interface. It is also possible to place virtual obstacles registered with the real world, which behave similar to real ones. Cyan color indicates some of the recognized edges of environment objects.

Figure 10. Battlefield Training Scenario

In order to fulfill the presented scenario requirements, MARCam allows user mobility through the battlefield. Besides that, aiming soldier's freedom of movement, the platform has to be compact and lightweight. Due to MARCam's specific processing nature, it consumes low power, allowing soldier autonomy necessary when he/she is immersed in an outdoor environment. Finally, a complex scenario such as the one described here demands a high processing load, justifying the use of a hardware-based solution. This factor supports applications that target an efficient execution performance and an adequate visualization frame rate.

CONCLUSIONS AND FUTURE DIRECTIONS

This paper presented a novel platform for developing hardware based AR systems. The infrastructure showed to be adequate for building AR applications targeted to low level implementations. It was also shown that it is possible to use efficiently a component oriented design model in the development of an embedded AR project. The performance results obtained when using the platform are promising, since they were far better than the ones obtained with software based implementations.

Regarding MAR, the two techniques chosen to be implemented present different levels of complexity. Actually, we already verified feasibility of

implementing both algorithms, since they require a relatively small amount of resources.

As future work, power consumption evaluation should be done, comparing the hardware consumption with the ones obtained from desktop, notebook and PDA applications. The existing modules will be refined and the currently ongoing components will be concluded. After that, case studies will be performed regarding embedded MAR applied to a specific knowledge domain, such as training and guidance.

In order to smooth the progress of the development of such applications, an authoring tool for hardware based AR applications might be considered, where the user would be able to choose and link the needed components using a GUI (Graphical User Interface). An infrastructure for accessing external memory from the FPGA is also under definition, hence increasing the amount of system memory, instead of relying on restricted internal one. New image capture and scene exhibition components are planned, interfacing with elements such as USB camera, GPU and LCD display.

ACKNOWLEDGEMENTS

The authors want to thank MCT and CNPq, for financially supporting this research (process 507194/2004-7).

REFERENCES

Bimber, O., & Raskar, R. (2005). *Spatial Augmented Reality: Merging Real and Virtual Worlds*, Wellesley, Massachusetts: A K Peters.

Bradski, G., & Pisarevsky, V. (2000). Intel's Computer Vision Library: Applications in Calibration, Stereo Segmentation, Tracking, Gesture, Face and Object Recognition. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 796-797.

Comport, A. I., Marchand, E., Pressigout, M., & Chaumette, F. (2006). Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Transactions on Visualization and Computer Graphics, 12*(4), 615-628.

Fiala, M. (2004). ARTag Revision 1, a Fiducial Marker System Using Digital Techniques. Technical Report NRC 47419, National Research Council Canada.

Foley, J. D., Van Dam, A., Feiner, S. K., & Hughes, J. F. (2005). *Computer Graphics: Principles and Practice*, USA: Addison Wesley.

Fung, J., & Mann, S. (2005). OpenVIDIA: parallel GPU computer vision. *Proceedings of the International Conference on Multimedia*, 849-852.

Gonzalez, R., & Woods, R (1992). *Digital Image Processing*, Reading, Massachusetts: Addison Wesley.

Haralick, R., Lee, C., Ottenberg, K. & Nölle, M. (1994). Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem. *International Journal of Computer Vision*, *13*(3), 331-356.

Jain, R (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*, New York City, New York: Wiley-Interscience.

Keating, M., & Bricaud, P. (2002). *Reuse Methodology Manual for System-on-a-Chip Designs*, Springer.

Lepetit, V., & Fua, P. (2006). Monocular model-based 3D tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision*, *1*(1), 40-47.

Lourakis, M., & Argyros, A. (2005). Efficient, Causal Camera Tracking in Unprepared Environments. *Computer Vision and Image Understanding, 99*, 259-290.

Lucas, B. & Kanade, T. (1981). An Iterative Image Registration Technique with an Application to Stereo Vision. *Proceedings of the International Joint Conference on Artificial Intelligence*, 121-130.

Matsushita, N., Hihara, D., Ushiro, T., Yoshimura, S., Rekimoto, J., & Yamamoto, Y. (2003). ID CAM: a smart camera for scene capturing and ID recognition. *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, 227-236.

Nistér, D. (2004). An Efficient Solution to the Five-Point Relative Pose Problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *26*, 756-770.

Nistér, D. (2005). Preemptive RANSAC for Live Structure and Motion Estimation. *Machine Vision and Applications, 16*, 321-329.

Pollefeys, M. (1999). Self-calibration and metric 3D reconstruction from uncalibrated image sequences. Ph.D. Thesis, ESAT-PSI, Katholieke Universiteit Leuven.

Primordial, (2007). *Battlefield Augmented Reality System.* Retrieved June 14, 2007, from http://www.primordial.com

Rajsuman, R (2000). *System-on-a-Chip: Design and Test*, Norwood, Massachusetts: Artech House, Inc.

Shi, J., & Tomasi, C. (1994). Good Features to Track. *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 593-600.

Sinha, S., Frahm, J.-M., Pollefeys, M., & Genc, Y. (2006). GPU-Based Video Feature Tracking and

Matching. Technical Report 06-012, Department of Computer Science, UNC Chapel Hill.

Toledo, E., Martinez, J., Garrigos, E., & Ferrandez, J. (2005). FPGA implementation of an augmented reality application for visually impaired people. *Proceedings of the International Conference on Field Programmable Logic and Applications*, 723-724.

Triggs, B., McLauchlan, P., Hartley, R., & Fitzgibbon, A. (2000). Bundle Adjustment – A Modern Synthesis. *Vision Algorithms: Theory and Practice*, 298-372.

Umlauf, E., Piringer, H., Reitmayr, G. & Schmalstieg, D. (2002). ARLib: the augmented library. *Proceedings of the IEEE International Augmented Reality ToolKit Workshop*, 2 pp.

Vacchetti, L., Lepetit, V., & Fua, P. (2004). Stable real-time 3D tracking using online and offline information. *IEEE TPAMI*, *26*, 1385-1391.

Wagner, D., Pintaric, T., Ledermann, F., & Schmalstieg, D. (2005). Towards massively multi-user augmented reality on handheld devices. *Proceedings of the International Conference on Pervasive Computing*, 208-219.