

RESEARCH

Open Access

Dedicated object processor for mobile augmented reality - sailor assistance case study

Jean-Philippe Diguet^{1,2*}, Neil Bergmann² and Jean-Christophe Morgère¹

Abstract

This paper addresses the design of embedded systems for outdoor augmented reality (AR) applications integrated to see-through glasses. The set of tasks includes object positioning, graphic computation, as well as wireless communications, and we consider constraints such as real-time, low power, and low footprint. We introduce an original sailor assistance application, as a typical, useful, and complex outdoor AR application, where context-dependent virtual objects must be placed in the user field of view according to head motions and ambient information. Our study demonstrates that it is worth working on power optimization, since the embedded system based on a standard general-purpose processor (GPP) + graphics processing unit (GPU) consumes more than high-luminosity see-through glasses. This work presents then three main contributions, the first one is the choice and combinations of position and attitude algorithms that fit with the application context. The second one is the architecture of the embedded system, where it is introduced as a fast and simple object processor (OP) optimized for the domain of mobile AR. Finally, the OP implements a new pixel rendering method (incremental pixel shader (IPS)), which is implemented in hardware and takes full advantage of OpenGL ES light model. A GP+OP(s) complete architecture is described and prototyped on field programmable gate-array (FPGA). It includes hardware/software partitioning based on the analysis of application requirements and ergonomics.

Keywords: Processor Architecture; Communication/memory optimization; 3D graphics; Inertial sensors; 3D positioning

1 Introduction

Recent breakthroughs in augmented reality (AR) display will bring a lot of new applications in the future. However, this also means an emerging challenge, which is the design of low-cost, low-footprint, and low-power systems to be embedded in see-through glasses. On the one hand, most of research works in related conferences (e.g. ISMAR), do not focus on embedded system design but on specific AR issues such as simultaneous localisation and mapping or virtual object handling. On the other hand, emerging see-through glasses are considered as a new peripheral connected to smartphones, which can execute AR applications. The contributions of this work are twofold. The first one is an in-depth study of algorithms for the target AR application in the domain of sailor assistance. The second one is the design of an architecture

that fits with application requirements. The application domain is outdoor AR without any equipment in the user environment. Such applications require the management and the fusion, at runtime, of multiple information flows (position/attitude sensors, points of interest, ...) and simultaneously the processing of graphical objects. The paper is organized as follows. In Section 2, we present our motivations for this research field and the specificities of the target applications. In Section 3, we present current technologies for AR and a state of the art of usual implementations. Section 5 describes the main steps of our original approach. Our solution is based on the adaptation of previous positioning solutions to the application context, on algorithmic transformations to improve data reuse and reduce processing load, and on a new architectural solution for object drawing. Section 6 presents our hardware/software architecture based on data locality and bandwidth optimization. Our architecture is dedicated but flexible and optimized according to AR applications;

*Correspondence: jean-philippe.diguet@univ-ubs.fr

¹Lab-STICC, CNRS, Université de Bretagne Sud, 56100 Lorient, France

²The University of Queensland, 4072 Brisbane, Australia

we come up with a coprocessor-based solution that provide a possible IP for mass market products. In Section 7, we present our prototyping results on field programmable gate-array (FPGA) and discuss performances estimations. Finally, we conclude and draw some overall insights.

2 Outdoor AR applications: case of sailor assistance

AR by itself is not a new topic, but many challenges remain unsolved. This is especially true in mobile and outdoor contexts, where field markers are not applicable and video-assisted model-based tracking is difficult to implement in real-life luminosity conditions. The proposed application set is based on the following observations. First, designers already have at their disposal a huge amount of recorded and classified data of geolocalization. Second, these data can be added to a user's field of vision by means of see-through glasses. Third, an embedded system can compute 3D objects that fit with the landscape, if appropriate sensors can provide the user attitude. Fourth, we consider an application-based distant objects that do not require an extremely accurate positioning, so no camera is needed for the pose estimation. Finally, basic ergonomics rules for positioning applications require simple and meaningful objects in the user field of view. In this context, many applications, in civilian and military domains, can be designed to improve security and navigation decisions with different hands-free and low-footprint devices. In this context, we consider the particular but complicated case study of sailor assistance from which can be derived various requirements for a generic embedded system. This case study is illustrated in Figure 1.

On a boat, the understanding of the position is vital when approaching sensitive environments such as coasts, open sea reefs, high-density navigation zones, or harbor channels. These situations are true not only for small sailing or motorboats but also for large vessels where the navigation crew is limited with respect to boat sizes. In this kind of large ships, it is also recommended to combine visual checking, based on real environment observations, and instrument piloting. Current methods consist of going back and forth between map analysis and visual observations. However, matching map indications with a real environment can be tricky and error prone. It also represents a loss of time that can be precious in case of emergency. Finally, this matching can be simply impossible when the visibility is very bad. This is a relevant case study since a ship is a very unstable system. All the continuing motions have various parameters depending on boat speed, user moves, and ocean oscillations. Swell periods can vary between 0.05 and 0.1 Hz. However, this is also a domain where a lot of data are available. The first category includes static seamarks objects. The second one is dynamic but can be estimated, and in this

category, we find for instance the ocean streams and the tide-dependent shallow areas. The third one is provided by the automatic identification system (AIS) that broadcasts the positions, the heading, and the ID of boats or any maritime objects in the surroundings. All these data can be added to the user fields of vision according to position and attitude estimations. Then, we have boat-positioning data, which include global positioning system (GPS) measurements, speed, trim, heel, and heading. All these data, which can be encrypted in sensitive or military applications, can be obtained through a wireless network that does not require high bandwidth capacities. However, these data are useful but not sufficient since it is necessary to know the user attitude, which is defined with the head angular positions. These data have to be provided by embedded sensors that must be integral with the glasses. Redundancy between boat and user data can also be usefully combined to improve accuracy. For instance, the on-glasses accelerometers can be combined with the ship GPS to estimate local moves on a long vessel.

The intensity of the see-through device is also an important constraint in such outdoor applications. It must be compliant with the weather conditions such as rain, fog, or intense light reflection in sunny conditions; moreover, the system must also be usable at night.

Finally, ergonomics are of the greatest importance in navigation applications where the objective is to provide the user with useful and relevant information. Feedbacks from potential users we are working with, in both civilian and military domains, confirm that the number of objects must be limited and that object forms must be simple. These requirements let us consider that a graphics processing unit (GPU) is actually not necessary, it also means opportunities for implementation optimizations.

3 State of the art

Recent technology breakthroughs are enabling new kinds of applications, hereafter we discuss this evolution in three parts: display, applications, and embedded systems.

3.1 AR technologies

The first breakthrough occurred in the domain of sensors for position, speed, acceleration, and attitude (yaw, pitch, roll) measurements. For a long time, the size, and cost of such devices have limited their use to navigation instruments in aircrafts and satellites. However, microelectromechanical systems (MEMS) technologies are now providing integrated and low-cost inertial measurement unit (IMU) solutions [1,2] that make possible the design of mobile consumer systems. The most widespread solution is based on the association of two kinds of MEMS devices: a 3-axis accelerometer sensor and a 3-axis magnetometer sensor. The combination of these sensors can provide the estimation of body translations and attitude, it

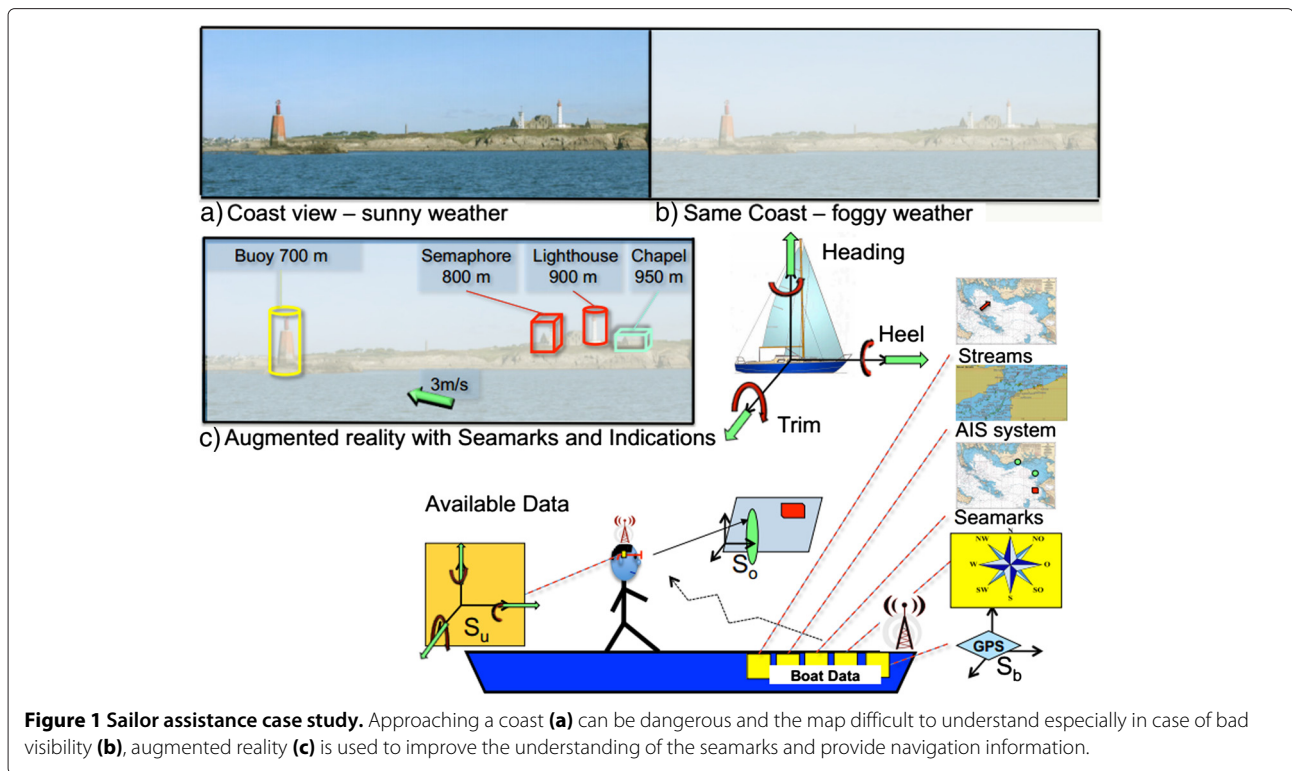


Figure 1 Sailor assistance case study. Approaching a coast (a) can be dangerous and the map difficult to understand especially in case of bad visibility (b), augmented reality (c) is used to improve the understanding of the seamarks and provide navigation information.

means 3-axis inclinations and so compass capabilities. ST has introduced in 2010 a new single module solution integrating these two kinds of MEMS sensors, with significant improvements in terms of accuracy and linearity. More recently, gyroscopes, that return angular velocities, have also been proposed in integrated versions. InvenSense has unveiled in 2010 an IMU including a 3-axis integrated gyroscope (angular speed) combined with a 3-axis accelerometer. In 2011, ST presented the iNemo engine that includes 3-axis linear accelerometers, 3-axis angular speed measures, a magnetometer (heading), and a barometer (altitude). Like the InvenSense solution, the whole device uses a 32 bit processor to filter motion sensor data but the access to this processor is very limited. Moreover, integrated gyroscopes consume one order of magnitude more power than magnetometers and accelerometer. We present solutions to these issues in Section 5, and we show that gyroscopes are not needed in our context and that a simple softcore processor synthesized on a FPGA can run motion estimation algorithms.

The second key technology that opens new horizons to AR applications comes from the domain of head mounted displays (HMD). New see-through glasses (see Figure 2) are now affordable. Companies like Vuzix, Optinvent, Laster, or Lumus have developed prototypes or already commercialized some products with some limitations. The arrival of Google on this market will also boost AR-related applications. In [3], the authors present

an interesting overview that compares the different optic technologies and patents. This type of device paves the way to future AR reality applications on a mass market. Moreover, in 2011, the first prototype of a single-pixel lens has been demonstrated [4]. However, one of the main issues in our context is the need of high luminosity, Laster proposes a mask with a very important intensity based on a backlight LCD, it offers 5,000 cd/m² whereas usual OLED-based solutions provide intensity around 200 to 400 cd/m².

3.2 Outdoor applications

In our outdoor context, efficient marker-based solutions [5] cannot obviously be used. Relevant markerless solutions based on image processing [6,7] could be proposed instead, and the solution described in [8] is for instance promising on mobile platforms. However, it is not applicable in our outdoor conditions and other issues such as the important distance of targets and boat motions disqualify this approach. For outdoor positioning and orientation, some solutions based on data have been introduced but are first based on handheld devices (e.g., Android and iOS) and secondly are not accurate enough to place an object at the right place. A solution, based on a differential GPS, was proposed in [9] with inclinometer and magnetometer to display text information on campus buildings. The authors noticed the acceptable inaccuracy with regard to the application requirements, but they also indicated

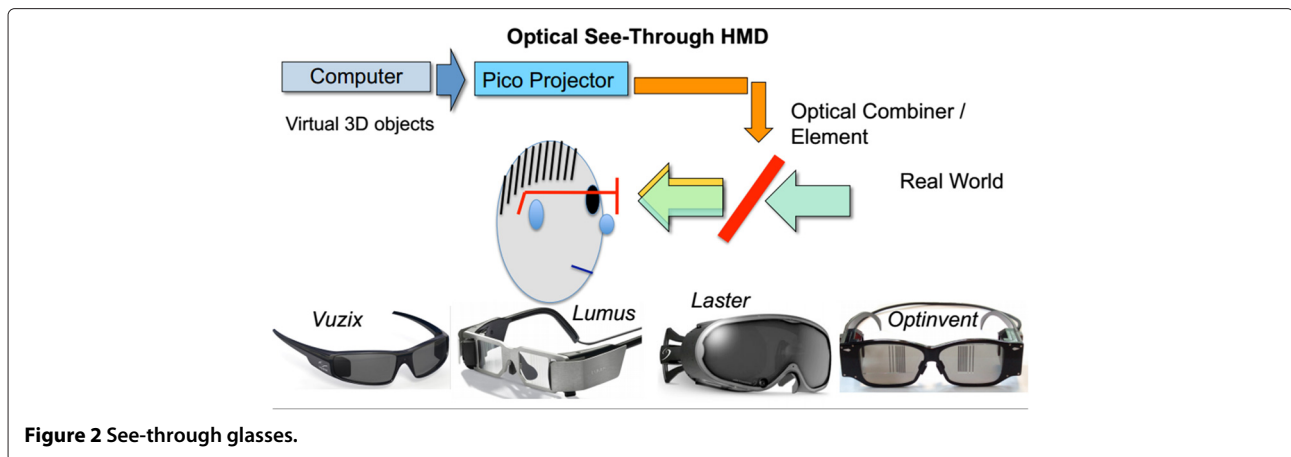


Figure 2 See-through glasses.

issues with visibility in sunny conditions and difficulties with the inertial sensors. Sensors and related signal processing have been improved since, but visibility and integration still need to be solved.

3.3 Embedded system architectures

Miniaturization and power consumption are the first constraints for the design self-content mobile AR systems. Object positioning and drawing require computation resources, but control capacities are also necessary to handle data acquisition from sensors and communication protocols. The implementation of the first prototypes for such applications was based on backpacked laptops with 3D graphics cards [9,10]. Advanced mobile multi-processor architectures are now available (e.g. OMAP5, Snapdragon, Atom N2800, ST-Ericsson Nova, Apple A5, ...), and they are typically based on cortex ARM cores or Intel Atoms with specialized graphics and video coprocessors. The main advantage of such architectures is the availability of software development frameworks. This kind of platforms is typically embedded in recent AR systems that also integrate IMU sensors: OMAP3530 for Optinvent and OMAP4430 for Vuzix and Google. However, they are mainly used to handle the video stream and the wired or wireless communication with a smartphone that remotely runs the AR application. As a matter of fact, high-resolution video games and video and image processing for object identification and complex online 3D graphics computations would justify such processing resources. However, the type of AR applications we are targeting requires simple objects instead. Secondly, they do not need cameras and complex pose computation that requires image processing. So we believe and we demonstrate that such general-purpose processor (GPP)+GPU solutions are actually oversized regarding power autonomy constraints. We present in Section 4 an implementation of such a solution based on the Nova system-on-chip (SoC). Another possible solution

is provided by reconfigurable architectures that enable specifically optimized and low-frequency designs. These rely on hardware/software design methodologies and recent high-performance FPGAs. These FPGAs are often power-hungry; however, the roadmap of FPGAs is clearly focused on this power issue with the aim to address the embedded system market. Recent hybrid ARM/FPGA architectures such a Xilinx/Zynq open new perspectives. On-chip memory capacity is also a key issue where significant progress has been made. For example, the Artix Xilinx low-power, low-cost family embeds up to 12 Mbits of block RAM. Regarding GPUs on FPGAs, Xylon has added a 3D graphics module to the Logibrick library available under license in an early access version for evaluation. The architecture relies on a three-stage pipeline. This solution is a simplified version of the usual graphics pipeline and is designed for general purpose OpenGL embedded system (ES) applications. It shows that low-frequency dedicated architectures can be designed for this purpose. In [11], the authors present a GPU-inspired and multi-threaded softcore architecture, which is programmable with the NVIDIA Cg language. The aim is to simplify the use of FPGA-based acceleration boards for high-performance computing. Our approach is different, it is dedicated and designed for embedded systems and AR applications with a high focus on data locality optimization for minimizing data transfers.

4 Mobile platform-based standard implementation

We have designed an AR system based on standard platform and software to evaluate the impact of the embedded system compared to the AR display device.

4.1 AR display

As presented in Section 2, our class of applications requires optical see-through displays 2. Considering weather conditions (light reflection, fog, humidity), it

turned out that Laster was providing the better solution for our outdoor conditions. The maximum light intensity of the product (ski mask) is about 5,000 cd/m² in color mode and monochrome is even higher. The resolution and the field of view are suitable for a see-through device: 800 × 600 and 40°×30°, respectively, which is better for light intensity (×10) and field of view (×2) than other solutions. The display technology is based on a LCD panel with LED backlight to reach expected high luminosity that cannot yet be delivered with AMOLED technologies.

4.2 SoC and development board

We have implemented a first version of the application on a ST-Ericsson *Snowball* board, which includes the A9500 SoC (see Figure 3). It is a GPP+GPU architecture that implements a double cortex A9 with a flexible clock frequency between 200 MHz and 1.0 GHz and a GPU Mali-400 MP. It also includes a wireless communication chip that provides Bluetooth (BT), WiFi, and GPS. Head tracking can be performed with the MEMS available on the board, which are an accelerometer/magnetometer 3-axis (LSM303DLHC from ST) and a gyroscope 3-axis (L3G4200D from ST). The GPS data are supposed to be available from the boat or user throughout the low-bandwidth wireless (BT) connection, so the GPS and the WIFI are switched off.

4.3 OS and software

The mobile OS is Android; it offers a large amount of application programming interfaces (APIs) that simplify GPU programming with OpenGL ES 2.0 API [12] and allows to control and acquire MEMS data (*sensorManager*) to get position data from a GPS (*LocationListener*) and some managers to handle wireless connectivity (BT, WiFi). As a proof of portability, we

have also implemented the application on a Galaxy S3 smartphone, which is based on the same GPU. In AR applications, the main functionality is the graphic service and the system has to draw 2D or 3D objects with different colors and textures. The second service is the attitude computation or head tracking, and this step is needed to place right objects in right orientation on the display. A special task with MEMS is implemented and it is mainly based on a standard 7-states extended Kalman filter [13]; we come back on this type of filter in Section 5. The MEMS data rate is set to 50 Hz. Finally, a wireless service was developed to acquire data from the boat network (GPS, AIS, map sea-marks, wind and stream grids) when those data are available from a PC on the boat. We used the NMEA protocol commonly used in marine electronics and BT for wireless communications, that do not require high bandwidth.

4.4 Power measurements

We measured the power consumption of the Snowball board and the Laster mask with different configurations, and the results are given in Table 1. The mask has five levels of luminosity, and the highest one consumes 130 mA. We then observe that the power consumption of the embedded system is much larger than the AR display, and it consumes 310 mA with a basic kernel and reaches 500 mA with Android, BT, MEMS (without gyroscope), and the application running with a set of simple objects. Expressed in watts, the SB board consumed 2.5 times more power than the mask with the highest luminosity. We note that the Bluetooth connection (version 2.1-EDR) significantly increases the power consumption (110 mA), in practice, this overhead can be drastically reduced with a low-power version (e.g., 4.0).

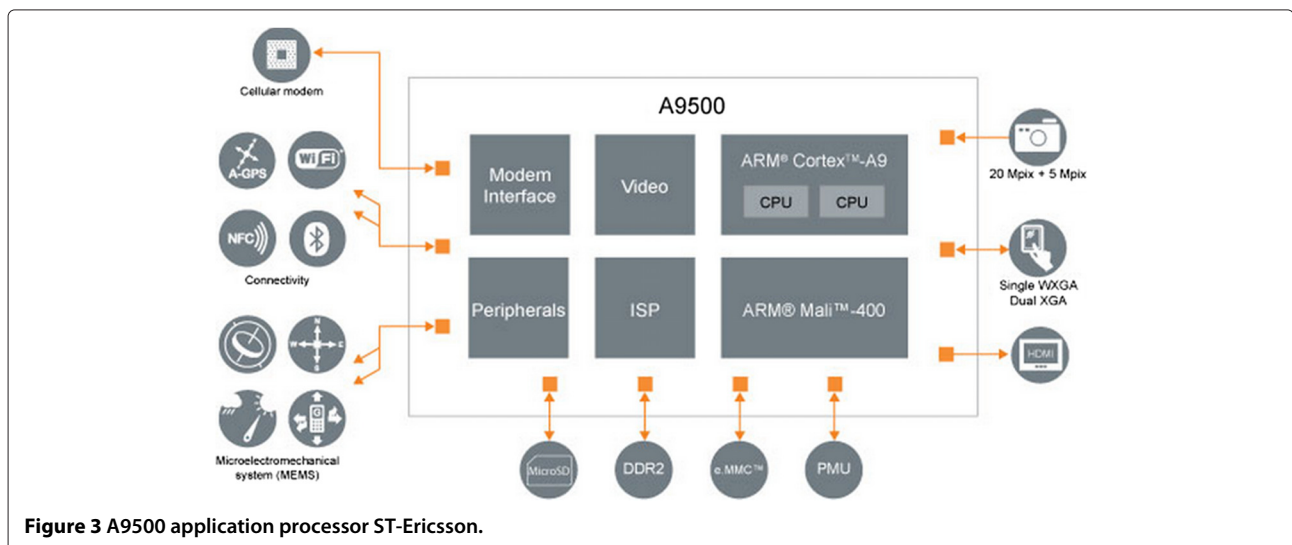


Figure 3 A9500 application processor ST-Ericsson.

Table 1 Power measurements: Snowball 1 GHz (SB), Laster AR Mask, Androïd (OS), BT (7 Kb/10s), no background, simple objects

	MASK	SB uboot	SB OS	SB, OS BT	SB, OS BT, EKF MEMS	SB, OS BT, EKF MEMS objects
Power (mA)	[100–130]	310	350	460	470	500
Voltage (V)/Power (W)	7.2/[0.72–0.94]	5/1.55	5/1.75	5/2.30	5/2.35	5/2.5

4.5 Conclusion and optimization opportunities

We can draw the following conclusions from this experiment. First, contrary to common beliefs, the power consumption of the embedded system is dominant even if a high-intensity AR LCD display is used. It means that the embedded system represents real opportunities for power optimizations. From a general point of view, 3D graphics represent complex and greedy tasks but optimizations and simplifications are possible if we consider ergonomics constraints and requirements of the most promising AR outdoor applications with see-through glasses. Considering the application context (orientation and notifications), we can point out three of them. Firstly, object distance means that we can relax the accuracy constraint since the size of the objects is decreasing with distance. Secondly, there is no background and a limited number of simple objects; the background is the real world, and ergonomics impose that a few simple objects can be drawn at the same time. Finally, useful orientation objects are static or move slowly if they are far away. All the previous features provide a rationale for a simplified implementation of 3D graphics that may be usefully exploited to optimize and specialize the design of the embedded system. Our solution follows this approach and is demonstrated on FPGA devices.

5 Application: algorithm and optimizations

In the following sections, we present the design choices to specify and implement the application flow described in Figure 4. We address the three following points: i) object positioning according to sensors and application context, ii) object drawing, and iii) some optimizations that have been introduced according to the application context.

5.1 User attitude modeling

The aim of stage I in Figure 4 is to compute user orientation and position. Our objective was to develop a robust and gyro-free solution based on magnetometer, GPS, and accelerometer. The choice of a gyro-free solution is motivated by the cost, the footprint, and the power consumption, which is one-order of magnitude larger

compared to magnetometers and accelerometers. There were two difficulties. The first one was the state of the art that was mainly related to aircraft, automotive systems, or AR applications with computer-based implementation without great concerns about size and power consumption issues [14]. The second point is related to the filtering and estimation problem, actually determining position, speed, and attitude from a set of noisy sensors is a non-linear problem that cannot be solved with traditional Kalman filters. We have studied various kinds of alternative solutions for non-linear filters, which were based on EKF (extended Kalman filter) [15], UKF (unscented Kalman filter) [16], or UPF (unscented particle filter) [17]. According to the current project environment and constraints, it turned out that applying Wahba’s method can solve the question of the gyroscope. This technique has been applied in aeronautics and avionics domain in [18] and considers gravity and magnetic field as the two required non-collinear vectors. It is based on the quaternion modeling, that also offers interesting properties such as the computation complexity for rotations, the stability in the presence of coding and rounding errors and the inherent robustness regarding the gimbal lock problem. Our complete solution for position and attitude estimation is described in Figure 5. It is finally based on a low complexity 6-states EKF (I-B(2)) algorithm for speed and position estimations. This EKF is loosely coupled with a low-frequency GPS and gets the body attitude data, as a quaternion vector, from a 6-states KF algorithm that implements Wahba’s method (I-B(1)) and used a linear acceleration derived from the GPS speed (I-B(4)). The proposed EKF-6 algorithm was previously applied in [19] with gyroscope data that are removed in the proposed version. Moreover, the acceleration data are combined with data from GPS after filtering. The complete solution also relies on a robust method for the auto calibration of the magnetometer [20]. It leads to a complex 14-states EKF algorithm; however, it is used only once at start time or with a very low frequency if the environment is changing. Note that the solution can easily be augmented with new data. It means that if gyroscope data are available with

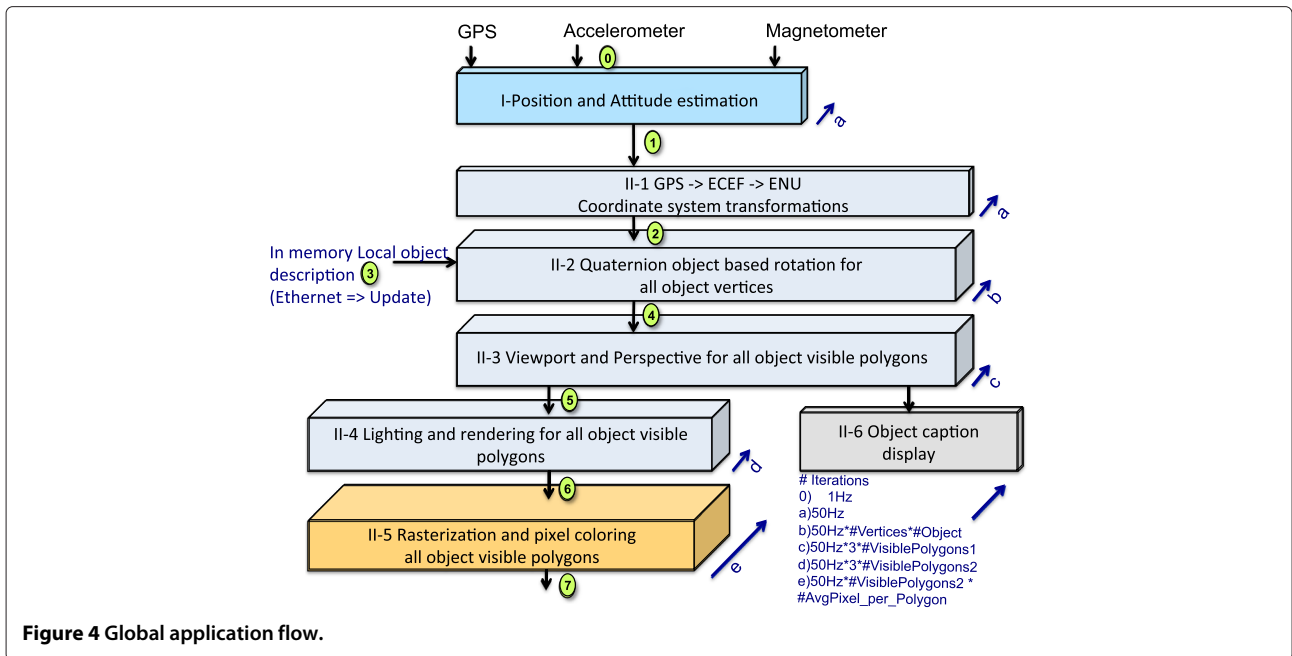


Figure 4 Global application flow.

reasonable cost and footprint, they can be added to the model.

5.2 Virtual object: geometry stage

The objective is to select only objects of interest in the context of the user. The definition of interest is a real question we study with different categories of users, but it is out of the scope of this paper. In this work, we consider three configurable parameters but various rules could be introduced as well. The first one is obvious; this is the position in the user field of view. The second is the minimum size of object after 2D projection. The third one is the choice of accessible objects stored in the object

memory. This selection is implemented in software and can be configured according to confidentiality issues or contextual search criterions. Hereafter, we summarize the steps of the geometry stage.

From GPS to ENU coordinates. Given the GPS coordinates, two successive transformations are applied to position and orientation data (stage II-1). The first one is the step from GPS to earth-centered, earth-fixed (ECEF) and then from ECEF to east north up (ENU).

Object rotation based on user attitude. The user attitude, namely the three angles that define user head orientation,

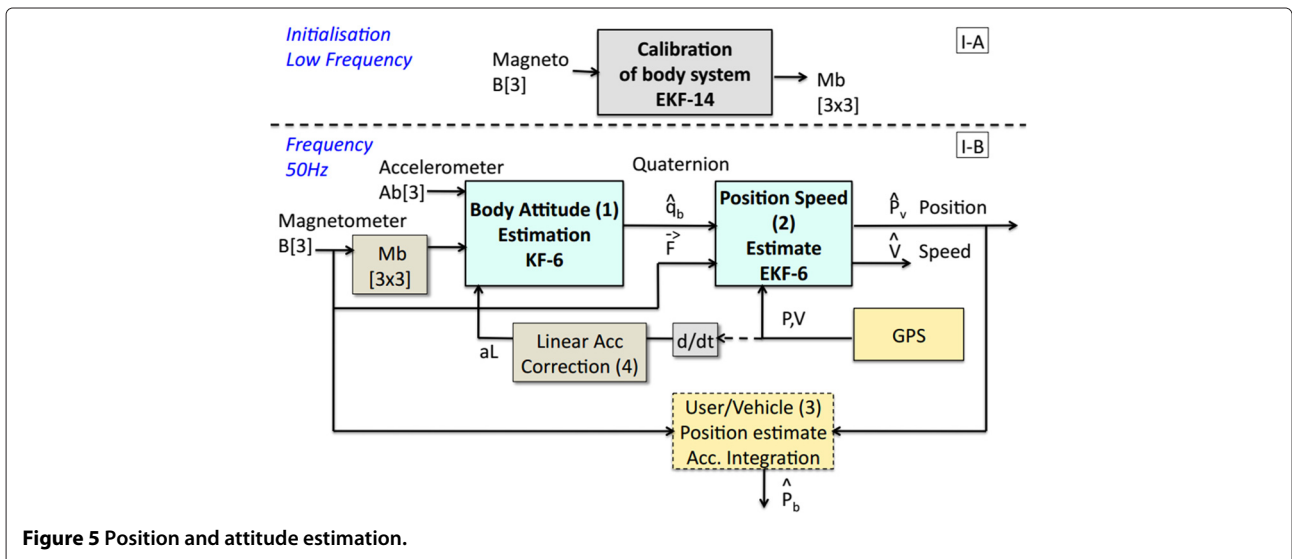


Figure 5 Position and attitude estimation.

is modeled with the quaternion formalism ($q = q_0 + q_1.i + q_2.j + q_3.k$). Then, a rotation (from V to V') is applied in stage II-2 to every point of every visible object according to the following operation:

$$V' = qVq^{-1} \text{ where } q^{-1} = q_0 - q_1i - q_2j - q_3k. \quad (1)$$

In-viewing frustum test, 3D→2D. The objective of the 2D projection step is firstly the selection of visible points and objects within the user field of vision and secondly the computation of 2D coordinates after a perspective projection as shown in Figure 6. An object is considered as visible if it is in the field of vision, which is defined with horizontal and vertical angles. It must also have a size bigger than a minimum sphere, which is defined by a ray R. The viewport operation is based on these bounding values and requires, for every point of all relevant objects, four multiplications and four tests. Then, the projection can be computed with three divisions per object and then four products for every point of all visible objects.

5.3 Virtual object drawing

Three visibility tests. The two first tests are applied in stage II-3 before drawing any object. The first one has been described in the previous section. The second test addresses the sign of polygon normal vectors. Considering opaque surfaces, if the Y' value N_y of the polygon normal vector in the user coordinates system is negative, then the polygon is not visible. Based on vertices' coordinates, this value is computed for each polygon and requires three multiplications and four additions. The third test is the well-known Z-buffer test (Y-buffer with our conventions) in implemented. The aim is to avoid drawing polygons, which are hidden by some closer ones. It is based on an array $A[i,j]$ that stores the smaller Y value of the closer polygon point located at the address (i,j) , where i and j correspond to the i th line and j th column of the display. We have implemented this test in stage II-5 with a dedicated module that computes the address and performs the comparison and update; details are given in Section 7.

Light modeling and optimization. Models used in stage II-4 are based on the barycenter method from OpenGL ES.

The light model for each vertex of the triangle polygon is given in Figure 7. Once the three vertex lights are computed (L^1, L^2, L^3), the barycenter method can be applied to fill the triangle A,B,C and compute each vertex light as follows:

$$L_c(x, z) = A(x, z)L_c^1 + B(x, z)L_c^2 + C(x, z)L_c^3 \quad (2)$$

where: $c = R, G, B$, $A(x, z) = \frac{F_{23}(x,z)}{F_{23}(x_1,z_1)}$, $B(x, z) = \frac{F_{31}(x,z)}{F_{31}(x_2,z_2)}$, $C(x, z) = \frac{F_{12}(x,z)}{F_{12}(x_3,z_3)}$, and $F_{ij}(x, z) = (z_i - z_j)x + (x_j - x_i)z + x_i.z_j - x_j.z_i$.

The implementation of this method has been optimized as follows: F_{ij} can be computed only once for each triangle and $K_{px}(c)$ and $K_{pz}(c)$ can be defined as unique color increments on X and Z axes, respectively:

$$-K_{px}(c) = \frac{L_1(c)((z_2 - z_3))}{F_{23}(x_1, z_1)} + \frac{L_2(c)((z_3 - z_1))}{F_{31}(x_2, z_2)} + \frac{L_3(c)((z_1 - z_2))}{F_{12}(x_3, z_3)}$$

$$-K_{pz}(c) = \frac{L_1(c)((x_3 - x_2))}{F_{23}(x_1, z_1)} + \frac{L_2(c)((x_1 - x_3))}{F_{31}(x_2, z_2)} + \frac{L_3(c)((x_2 - x_1))}{F_{12}(x_3, z_3)}$$

where $1/F_{ij}$ is computed only once per polygon to remove divisions. Finally, we obtain a simple algorithm based on fixed increments on both X and Z axes:

$$L_c(x + 1, z) = L_c(x, z) + K_{px}(c) \quad (3)$$

$$L_c(x, z + 1) = L_c(x, z) + K_{pz}(c). \quad (4)$$

This reorganization of computation has a strong impact on complexity since 3 DIV, 21 MULT, and 12 ADD are required once per visible polygon and then only 3 additions per polygon point.

Scan-line and incremental pixel shading. The process of rasterization (stage II-5) consists of mapping the real pixel addresses on the discrete display grid. The idea of the method is based on the use of the well-known Bresenham algorithm [21] for line drawing that eliminates divisions. In our particular case, we consider triangles so the algorithm first sorts out the three triangle vertices (yellow point in Figure 8) on the X -axis. Then, it simultaneously runs two Bresenham algorithms and draws two of the three segments (Pt1-Pt2 and Pt1-Pt3) and fills the triangle with K_{px} and K_{pz} color increments (dotted lines:

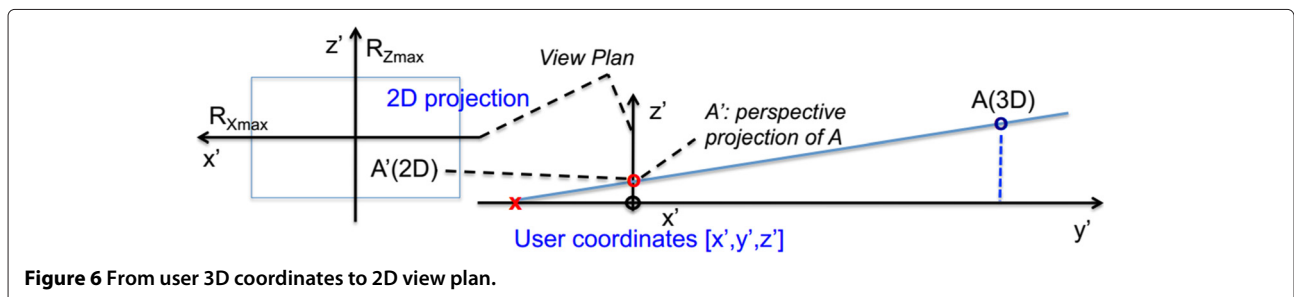
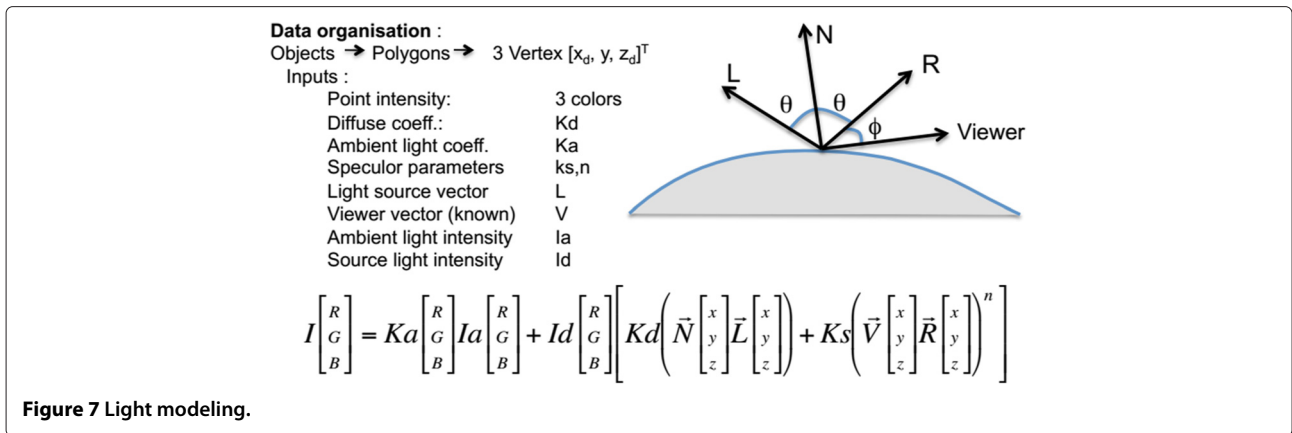


Figure 6 From user 3D coordinates to 2D view plan.



pixel coloring order). Once a triangle point is reached, the two remaining segments are considered and the same method is applied (Pt3-Pt2 and Pt3-Pt1). The main algorithm is the control of the pixel-shading method that also calls three key procedures. *InitDrawPolygon()* computes the three triangle vertex colors, *InitCoeffPolygon()* computes $K_{px}(c)$ and $K_{pz}(c)$ color increments, and *IncDraw()* writes the pixel value in the RGB format.

6 Embedded system architecture

In this section, we present our design choices based on our analysis of the applications requirements.

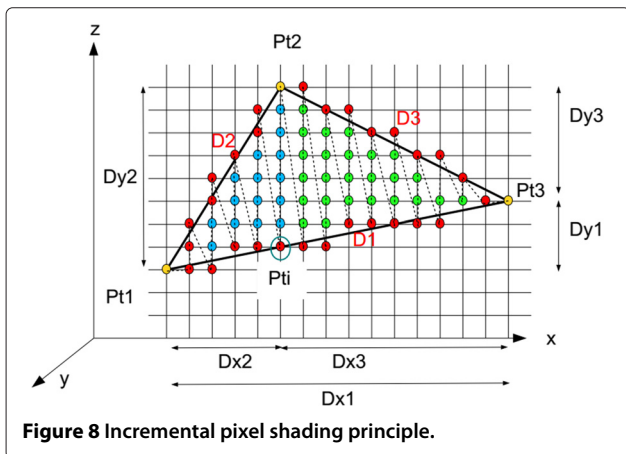
6.1 Data locality and bandwidth optimization opportunities

Data transfer optimization. The correlation between data locality or data reuse and power consumption and performance is a well-identified point [22] in the domain of embedded systems and data flow applications. Data transfers have a major impact on both performances and power optimization. So the scarcity of on-chip storage compared to off-chip capabilities imposes different kinds of optimizations. The first one addresses the problem of

algorithm organization and transformation so that the life spans of temporary variables are minimized. The second aspect is related to the architecture, and it includes local memory storage capacities, bandwidth, and memory access parallelism. The last point is the mapping of data on available memories and the scheduling of read/write operations. We have developed a precise model of the data transfers in the application described in Figure 4. This model is based on a complete set of parameters including the number of object and polygons for instance. Results for a typical case are presented in Section 7.1 and show that the update of the pixel memory represents more than 90% of the data transfers in practice (see Figure 9). It means that the computation of pixels has to be done so that data reuse is optimized. However, it does not mean that other transfers can be neglected, and the efficiency of computations strongly depends on the availability of data. In the following, we detail each step.

Sensor data and coordinates ((0),(1),(2) in Figure 4). Sensor data (accelerometers and magnetometers) as well as computed position and attitude coordinates before and after transformation are read only once at a rate of 50 Hz, and the amount of data is limited but of major importance since these data are used to compute all visible objects. In the application context, the required bandwidth is around 20 kB/s.

Access to object definition data (3). A set of typical objects can be stored in a local RAM memory and read when they appear in the user field of view. The content of the memory can be updated from servers, through a wireless network connection with contextual and application-related objects, but this transfer is very punctual and object memory can, in practice, benefit from a very high degree of data reuse. In this study, we consider that data are available in memory. Regarding memory accesses, the objective is to optimize the reuse rate of object data. We will see in the following how object data (position, colors and light



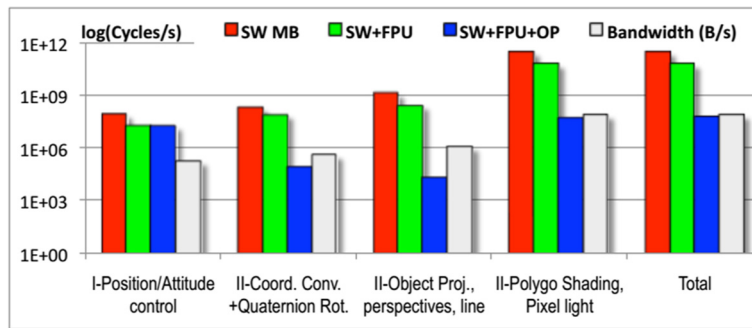


Figure 9 Performance summary considering: 8 objects, 16 vertex/Obj., 13 triangles/Obj., VGA display, average 2D object size: 1/20 VGA, IMU acquisition rate: 50 Hz, GPS rate: 5 Hz.

vectors of polygons points) are organized; it is depicted in Figure 10.

Transformed, filtered objects (4),(5). The same kind of optimization is applied to object data once they have been transformed in the new coordinates system according to user position and attitude. In our approach, all objects are first transformed before pixel computation and drawing, which are then applied to all objects. So the object memory is used to store updated object data, after geometric operations, and reused later. Again, the objective is to optimize data reuse and avoid multiple read operations. Another important aspect is object filtering; only visible object and polygons are updated according to object presence in the user field of view and object allowed minimal size.

Rasterization and pixel coloring (6),(7). Pixel computation is the greediest task in terms of bandwidth. Based on polygon values and color increment described in Section 5.3, pixels are updated in a specific video buffer. In our implementation, the data memory is distinct from the memory where this buffer is implemented. Another important point relies on the optimization of write operations to the video buffer; the default value of pixels is transparent, and only pixel related to visible objects is updated. In AR application, the space occupied by objects is very limited (e.g., 10%) compared to the display size (e.g., 64 × 480 VGA) and so important savings can be obtained. Another buffer is also required to store depths of objects (usually called Z-buffer), and the size of this memory depends on the visibility distance to be considered in the application. In our work, this buffer (called Y-buffer) is implemented in a dedicated memory.

6.2 Design methodology

Considering the mobile AR with distant and simple objects, the project objectives and the optimization opportunities, the first step was the analytical estimation

of performances. The second one was a projection on various architecture models on Xilinx FPGA that led to hardware (HW)/software (SW) partitioning decisions. The next step was the specification of the heterogeneous architecture and the coding with a hardware-description language (VHDL) at register-transfer level (RTL). We will see in Section 7 that hardware implementations are necessary. In the next section, we detail the final heterogeneous architecture model.

6.3 Multiple OP heterogeneous architecture

The system architecture is fully specified and tested at a cycle level, and the VHDL implementation is completed and synthesized; moreover, HW/Linux interfaces have been specified. The architecture, obtained after specialization and hardware/software partitioning, is described in Figure 11. The result is compact and flexible. The architecture is mainly built around a softcore, which is a MicroBlaze (MB) in this study, running Linux (Petalinux) to simplify the access to standard peripherals (I2C, UART, Ethernet) used for network access and communications with sensors. According to the specific AR context, we made the choice to consider positioning and graphics at the object level. So the processor is enhanced with some instances of a new accelerator called object processor (OP) that handles object positioning and drawing. The complete OP specification represents 16,000 lines of original VHDL code. Each OP is in charge of one or more objects. Video buffers are stored in a DDR-SDRAM, and memory access is implemented with the multi-port memory controller (MPMC) Xilinx fast memory controller. Each OP can access the video buffer through a dedicated video frame buffer controller (VFBC) port; each port is connected to a 32 bits FIFO. The Xilinx MPMC component allows for eight ports, which means that up to four OPs can be implemented. The Y-buffer may not be necessary; it actually depends on the number of simultaneously visible objects. However, if we consider a 8-km visibility along the Y-axis with a 1-m resolution, which is a very good

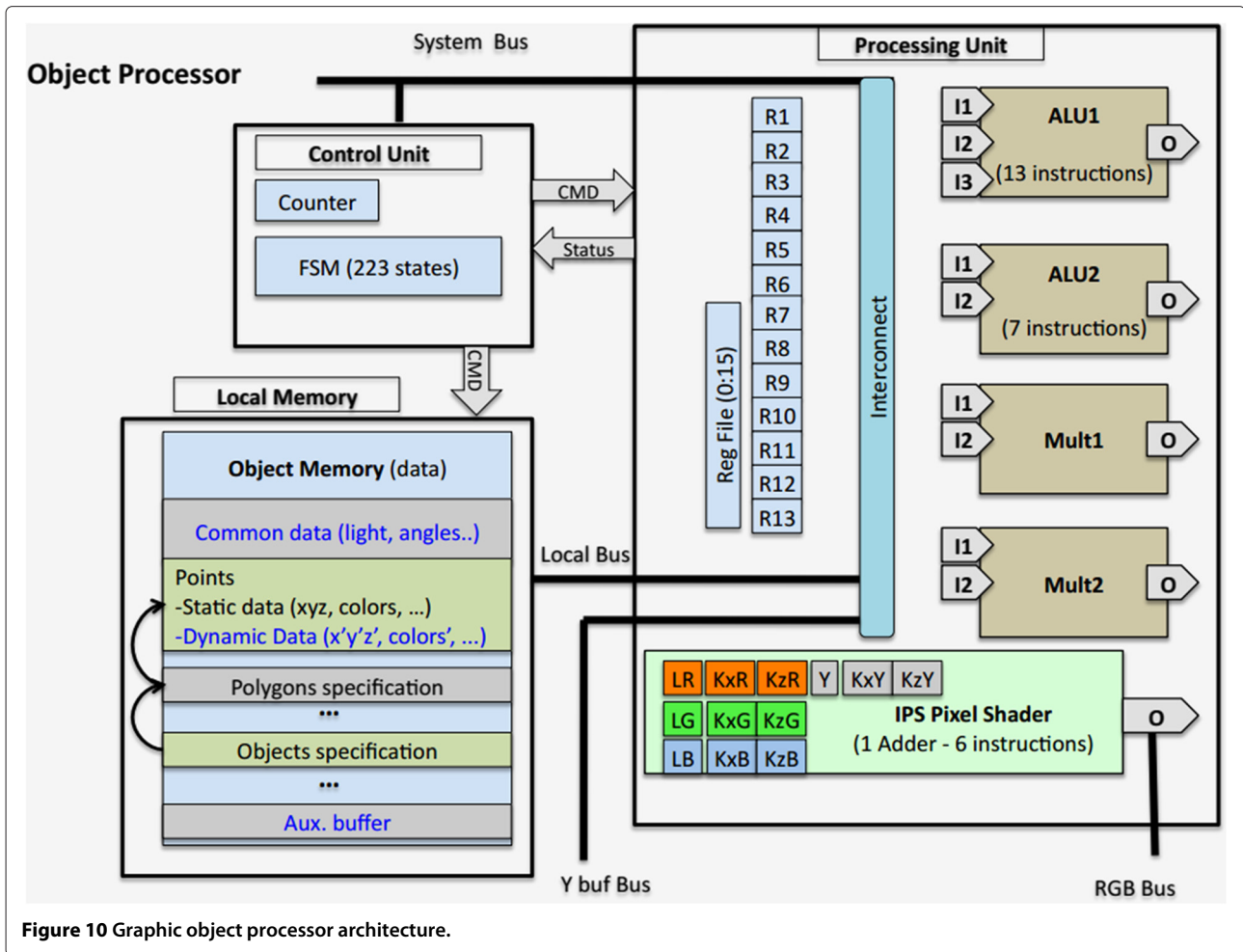


Figure 10 Graphic object processor architecture.

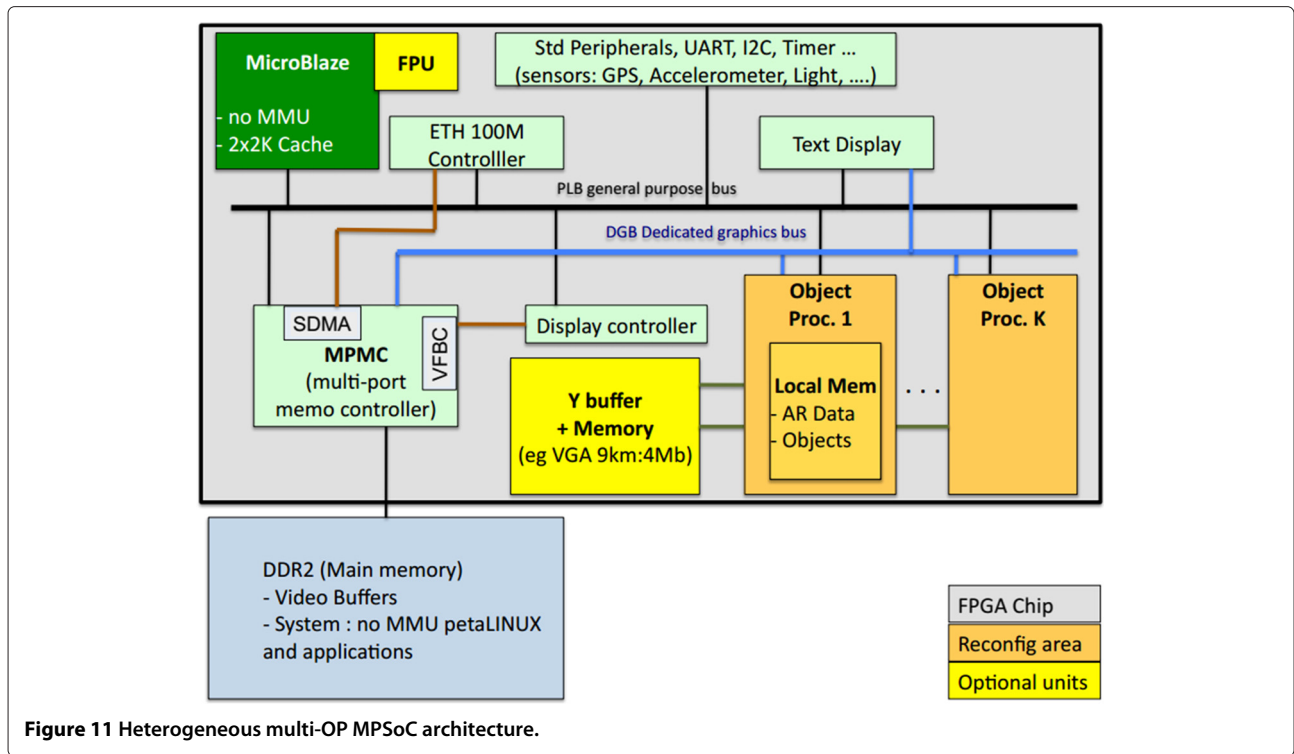
assumption according to [23] (5 km max), then a 4 Mbits memory is required. Regarding the Artix low-cost, low-power device (12 Mbits on-chip memory), this solution can be implemented on a single reconfigurable chip.

6.4 OP architecture

Control Unit. The OP component is the main architectural contribution of the proposed design, and it is described in Figure 10. Given new user attitude and position from the processor where positioning algorithms are implemented, each OP individually updates the position and the drawing of the objects where it is in charge of. The CPU stores object initial coordinates and features (polygons geometry, colors, and so on) in the OP local memory whose content is detailed in Figure 10. The CPU can decide the load of each OP, and the choices of objects to be drawn according to user priorities and field of vision. The OP is a strongly optimized N bits architecture, where N can be decided at design time according to accuracy constraints (e.g. $N = 16$ bits, 11 integer, and 5 rational). The design has been focused on data locality and bandwidth optimization, and the whole computation is controlled

with a 223-states finite state machines (FSM) organized in five main stages (see Figure 12): 1) *attitude and position data acquisition and coordinate system transformations*, 2) *object scaling and 2D projection*, 3) *visibility tests and vertex color computation*, 4) *polygon shape tests and color increment computation*, and 5) *polygon drawing*.

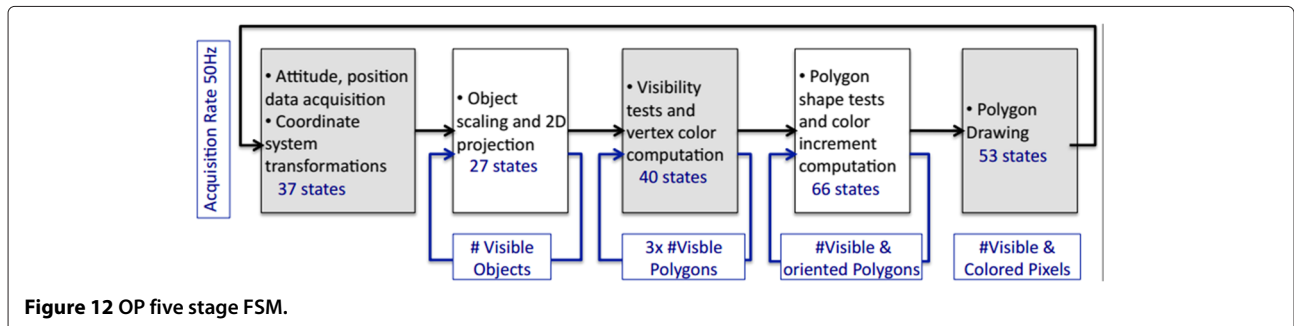
Processing unit. An OP implements two specific arithmetic and logic units (ALU), which are designed according to AR algorithm requirements. The first one has 3 inputs and 13 specific but simple instructions (e.g., fast implementation of division or increment operations), and the second one has 2 inputs and 7 instructions (e.g., decrement with zero test). The architecture also includes two multipliers, 13 general-purpose registers and a register file with 16 places. This is an important point regarding data bandwidth and concurrent access requirements. A new component called the incremental pixel shader (IPS) is introduced to apply the final step of the proposed method, which is based on horizontal and vertical color increments jointly with the double Bresenham method. It is mainly composed of registers with X , Y , and Z increments, an



ALU and a controller. Given K_{px} and K_{pz} , IPS can increment polygon pixel values and this component directly computes RGB values for each pixel with six increments or move instructions.

Interconnections and memories. The OP architecture is based on five concurrent communication buses to face bandwidth requirements. The first bus is 32 bits wide; it is connected to the local memory. The size of the memory is flexible according to application requirements (e.g., 5 objects with 20 points and 10 polygons each and $N = 16$ bits requires 39 Kbits). Moreover, the bus controller provides 1×32 , 2×16 , or 4×8 bits accesses. This flexibility is intensively used to concurrently transfer 16 and 8 bits data. The local memory is depicted in Figure 10, and it is organized in five regions. The first one contains common data including ambient light direction, number of objects, and numbers of polygons. The second region

contains point positions and light values, the static part stores initial point positions, and the dynamic part is used for updated positions in the user coordinate system. The third region contains specification of polygons as a list of point addresses. The fourth region stores the specification of objects as a list of polygon addresses; it also includes some specific coefficients for the object light modeling. Finally, the last region is a buffer used for temporary data. The second bus is 16 bits wide and connected to the register file that can be accessed simultaneously with processing unit registers. The register file length is 16, and it is used for the storage of temporary data produced and reused during processing. The third bus is the access to the Y-buffer, which is interfaced through a specific module designed to optimize the test of pixel depths. It works as follows: during the execution of the fourth stage of the processing flow (polygon shape test, color increments), the Y values of polygons are transferred to the Y-buffer



interface that sends back a notification if the tested polygon is hidden by closer ones. In such a case, the current polygon processing is interrupted and a jump is executed to the next polygon processing. Finally, the fourth connection is a RGB bus (24 bits pixel values), which is provided to access the video buffer.

Security aspects. Some applications require the object information to be kept secrets, if the softcore (CPU) is hacked then the main weakness is the link between the CPU and the OP local memory where objects are stored. So the access must be designed carefully. The local memory can be updated in various ways. First, the memory can be loaded at design time or at configuration time (from a encrypted Flash file); in such cases, the architecture does not implement any physical link between the CPU and the local memory, and in that case, no read or write access is possible from the CPU. When new objects must to be loaded according to the user environment, the Linux solution can provide the required secure network access (SSL), and in this case, only write operations are physically implemented and no program running on the CPU can read local memory data. If the chip is considered as a secure area, then this solution is efficient since no sensitive data are stored on external memory except the configuration that may be stored in on a encrypted flash memory [24].

H-MPSoC reconfiguration. The architecture is such that multiple OPs can run in parallel since each of them works with a given collection of objects. The CPU and OPs are quite independent, since they rely on shared memory communications, so that the CPU can feed the OP local memories with new position data. So the architecture can be dynamically and partially reconfigured according to the number of objects and real-time constraints. If new FPGA devices can implement real low-power modes when the power consumption is negligible for unconfigured areas, then such a solution makes sense in terms of energy efficiency. As described in Figure 11, each OP must be placed in a dynamically reconfigurable area. Technically speaking, this solution is viable, and the critical parts lie in the implementation of the two bus interfaces (VFBC, Y-buffer) that must be isolated from the reconfigurable area.

7 Results

In this section, we provide and discuss results related to performance estimations, bandwidth use, implementation cost on FPGA, and power consumption.

7.1 Performances

First, the whole application flow has been described with a parameterized performance model that enables

the counting of operations and transfers according to 25 variables such as the display format, the number and size of objects, etc. Then, this model has been validated with some real profilings carried out on the target MB with and without floating point unit (FPU). The performance model of the hardware implementation is straightforward since the number of steps of the FSM is fully specified. Figure 9 gives the results with the following configurations:

- ‘SW’: fully software implementation on a MB.
- ‘SW+FPU’: MB with a floating-point unit.
- ‘SW+FPU+OP IP’: MB with FPU and one OP unit.

In this example, we consider a case, which goes beyond our case study requirements. Thus, in this configuration, we consider 8 objects, 16 vertices, and 13 polygons per object, a VGA display, an average 2D object size equal to 1/20 VGA and a sensor (IMU MEMS) acquisition rate of 50 Hz while the GPS acquisition rate is 5 Hz. If we consider a 100 MHz clock ($100 \cdot 10^6$ cycles), we observe that the positioning/attitude control part of the application requires 70% of available cycles with the SW solution. The margin is too small to be safe within a Linux system where additional user processes are necessary. Moreover, a FPU unit is of real value for matrix operations that have high precision requirements. The second solution ‘SW+FPU’ can also be considered for the object positioning (steps 1.1, 1.2, 1.3) that requires around 2 M of cycles, but the quaternion-based rotation of objects will be too greedy (66 M cycles) and would lead to a total of 80% of CPU use. As expected, the graphic part of the application is definitely out of the scope of any processor-based implementation, except for the 2D projection steps (3.8 M). But combined with previous application requirements, this operation cannot be mapped on the processor and the improvement would not make sense in terms of data locality. This means that the OP HW IP will handle all the graphic steps with a clear interface to the software positioning part that can feed the OP with filtered attitude and position data. The OP processor is then in charge of adapting object drawing according to new data. Figure 13 gives the evolution of performances for the three configurations. They are linear with the number and the size of objects, and the size is given as a ratio of VGA frame. We observe that the performances of the OP implementation remain three orders of magnitude better than SW ones. These features can also be used to decide the number of OP to be implemented if necessary. Based on the previous assumptions, 40 M cycles are required for a complete execution of the graphics part. Tables 2 and 3 give the implementation results of the OP HW IP on different Xilinx device families: Virtex 5 and 6 and Spartan 6. The last one exhibits the lowest clock frequency (73 MHz), which

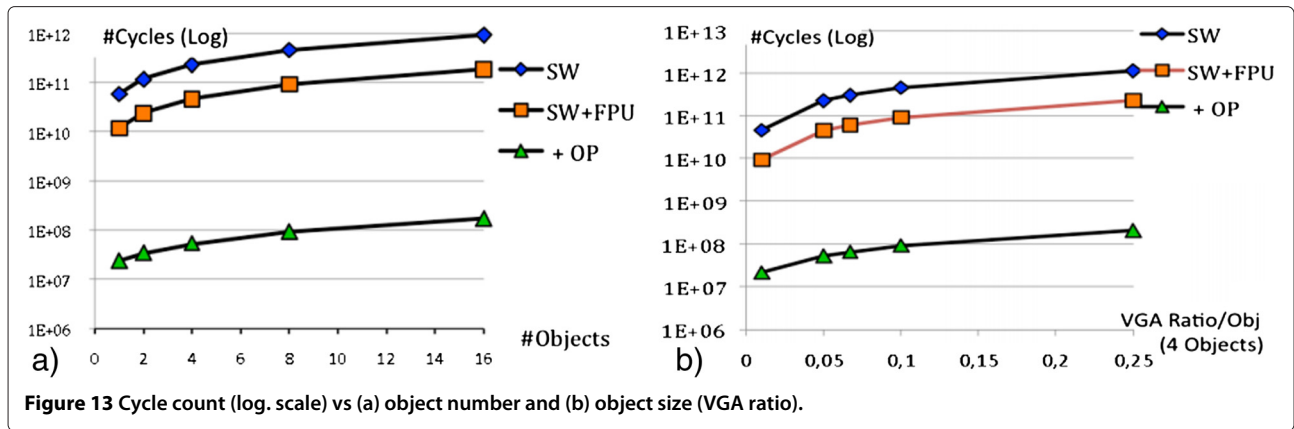


Figure 13 Cycle count (log. scale) vs (a) object number and (b) object size (VGA ratio).

means 73 M available cycles and consequently a use rate equal to 55%. In the two other cases, the clock frequency is 120 MHz and the use rate drops to 33%. As a conclusion, we observe that a viable solution is a MB with a FPU, running an embedded Linux OS and enhanced with a HW OP IP. This architecture offers the expected performances to run augmented reality applications. We have also tested the Zynq case, which is different since this is an hybrid device based on a ARM dual-core Cortex A9 combined with FPGA (Artix 7 or Kintex 7) on a single chip. It means that a MB is not necessary anymore and only OP IPs are synthesized; moreover, high-speed AXI links and a shared L2 cache are available for A9/FPGA communications, and finally, the processor can handle the video buffer.

Italicized (x) correspond to estimated data since the architecture is different.

7.2 Bandwidth use

The main design constraint was the access to storage units as well as the efficient ordering of computation so that data reuse is optimized. Figure 14 shows the use of local memory (RAM) and register file (RF) buses in each of the five control stages. R, W, and total means read, write, and read+write accesses. One corresponds to 100% use, meaning that a transfer occurs every cycle. We observe that the bandwidth use of the register file is over 50% in stages 1 and 2 and close to 100% in stages 3 and 4. It is not used in stage 5 anymore, and this free slot can be used, if necessary, to update the memory content with new objects. The use of the register file bus demonstrates the efficiency of

the implementation, and it is very close to 100% in stage 5 and over 50% in all stages. RGB out is close to 100% as well in stage 5, and it corresponds to the update of the video buffer, implemented in DDR, with new values of object pixels.

7.3 Implementation cost

The OP processor is specified as a VHDL code at RTL level. This code has been synthesized, placed, and routed for four Xilinx devices. Synthesis results are given in Table 2, and projections on different FPGA devices and families are given in Table 3. The MB configuration implements a 4 K I/D cache and usual peripheral controllers (UART, MPMC, Flash, Ethernet, VGA). Table 2 gives the number of slices and BRAM blocks required for MB+FPU and OP implementation and the maximum frequency clock after place and route. Figure 15 shows the required size of local memory that grows linearly with the number of objects and polygons. We observe that a OP is half the cost of a MB+FPU and that all implementations can reach 100 MHz except for OP on Spartan 6 (73 MHz). Note also that the results do not include the Y-buffer memory, which are not synthesized on the FPGA in these cases. However, it can be implemented as a buffer in the DDR, and the consequence is that the computation of hidden polygons may be interrupted a few cycles later. Table 3 gives the number of OP that may be implemented on the different targets according to the number of available slices and BRAM blocks. It also gives a ratio that includes the Y-buffer size considering a 12 bits depth (e.g., 5 Km, 1.25 m resolution) and a VGA display (3.68 Mb). The given

Table 2 MB+FPU and OP implementations: synthesis results on FPGA

Slices/Bram/F	Virtex 5	Virtex 6	Spartan 6	Zynq (Artix)
Nb/Nb/MHz				
MB+FPU	3,305 / 20 / 120	2,741 / 40 / 150	1,842 / 8 / 100	- / - / 866
HW IP OP	1,231 / 2 / 120	1,048 / 2 / 120	1,233 / 2 / 73	4,016 (LUTs) / 2 / 88

Number of slices or LUT slices/BRAM blocks/maximum frequency.

Table 3 Device choice and opportunities

	Nb of available		Nb of MB	Nb of OP	Used slices ratio	Used Bram ratio	Used Bram +Y-buff ratio	Nb of MPMC	Used BW ratio
	Slices	Bram18							
Virtex 5									
Min(LX30)	4,800	64	1	1	0.95	0.34	3.86	1	0.08
LX50	7,200	96	1	2	0.80	0.25	2.59	1	0.15
LX85	12,960	192	1	4	0.63	0.15	1.32	1	0.30
LX110	17,280	256	1	8	0.76	0.14	1.02	2	0.60
	17,280	256	1	4	0.48	0.11	0.99	1	0.30
LX155	24,560	384	1	12	0.74	0.11	0.70	2	0.90
Max(LX330)	51,840	576	1	12	0.35	0.08	0.47	2	0.90
Virtex 6									
Min(LX75)	11,640	312	1	4	0.60	0.15	0.88	1	0.30
LX130	56,880	528	1	12	0.27	0.12	0.55	2	0.90
Spartan 6									
LX25	3,758	52	1	1	0.82	0.19	4.52	1	0.08
LX45	6,822	116	1	3	0.81	0.12	2.06	1	0.23
LX75	11,662	172	1	6	0.79	0.12	1.42	2	0.45
Max(LX100)	15,822	268	1	8	0.74	0.09	0.93	2	0.60
Zynq (Artix)									
XC7Z010	17,600	60	0	2	0.46	0.07	(3.82)	(1)	(0.15)
XC7Z020	53,200	140	0	6	0.45	0.09	(1.69)	(2)	(0.45)

ratios represent the used resources divided by the available ones on the chip. The used bandwidth metric takes into account all transfers between OPs and the external DDR memory including Y-buffer accesses, and this metric is strongly dominated by pixel write accesses for updates (>99%). Note that we assume a very high video rate equal to the sensor rate, i.e., 50 Hz. The Xilinx memory controller (MPMC) can manage up to eight memory ports with different protocols (FIFO, DDR, cache,...), four

ports are used by the MB, so four remain available for OP accesses to the DDR.

7.4 Implementations discussions

We can draw several conclusions from these results. We first observe that the smallest Virtex 5 could theoretically implement one OP, but a usage of 95% of slices puts too much constraint on the routing step. However, the next device (LX50) can implement two OPs. A LX110 is

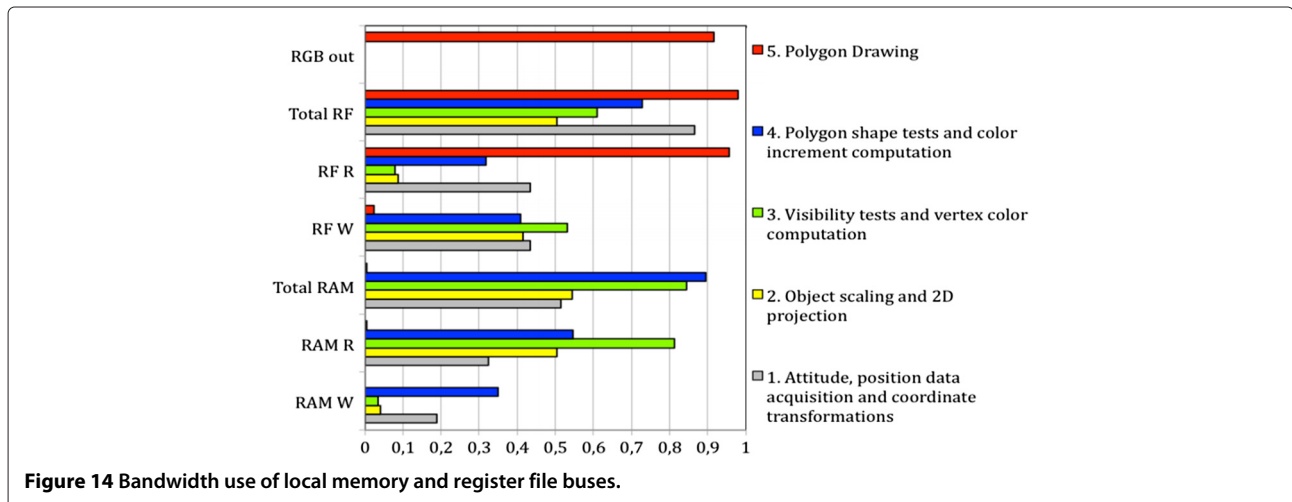


Figure 14 Bandwidth use of local memory and register file buses.

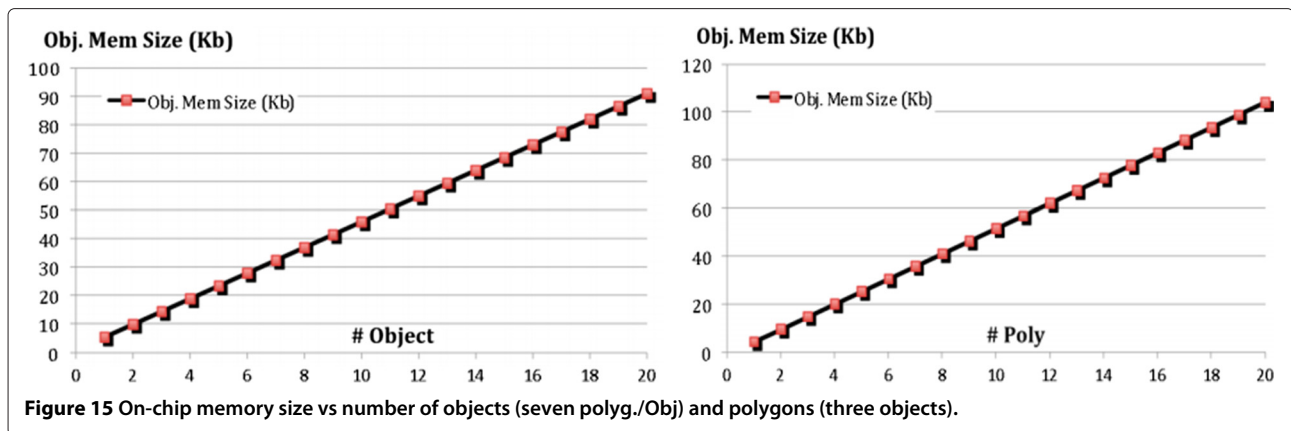


Figure 15 On-chip memory size vs number of objects (seven polyg./Obj) and polygons (three objects).

required to fully implement the Y-buffer on chip, such a chip can implement four OPs. With the next generation of Virtex 6, we note that the smallest device (LX75) is large enough to design a system with four OPs with an on-chip Y-buffer. A LX 130 can theoretically implement 48 OPS but is limited to 12 according to the memory bandwidth capacities. Virtex are expensive FPGAs while the Spartan family has been designed for low-cost design that fits the context of embedded systems. We observe that a small LX25 Spartan 6 can implement one OP and a LX45 provides enough resources for three OPs. The first solution that allows for an in-chip Y-buffer is the largest Spartan that can implement 12 OP according to bandwidth constraints. So low-cost solutions with high performances are possible and more can be expected.

Xilinx has released in 2011 a new device generation that relies on a 28-nm technology and brings significant power savings. This is especially true with the low-cost, low-power Artix 7 family that opens interesting perspectives. The positioning/control part of the application can be mapped on one A9 core, while two OPs can be synthesized on smallest Zynq device (XC7Z010).

7.5 Power and technologies considerations

Profiling-based power estimation shows that the power consumption of an OP is 100 mA on a Zynq Z020 with a maximum activity and a 88-MHz clock frequency. The power consumption of the ARM processor running Linux is 185 mA on the same device with the same voltage and a 866-MHz clock frequency. It means that a total power consumption remains lower than 300 mA with an OP that fully occupy the available 88 Mcycles. In our case study, only 40 M are necessary, so about 50% of power savings could be obtained if power gating was efficiently applied.

If we compare with the Snowball implementation, we can consider that the BT and MEMS power consumption remain identical. So the power difference is mainly due the Zynq and GPP+GPU (A9+Mali) implementations and

corresponds to 95 mA, and it leads to power gains larger than 200 mW.

We also made an estimation about a full implementation of the application on the Zynq ARM processor, with the best conditions, including no cache miss and no pipeline stall, we obtain about 600 Mcycles to be compared with the 40 Mcycles of the OP coprocessor, if we consider the relative clock frequencies (866/88 MHz) and power consumption only during computation, the OP energy efficiency is finally 15 times better.

This short study show that FPGA offers credible solutions. The power consumption of the OP coprocessor synthesized on a small Spartan LX25 is 85 mA (0.111 W) for instance while the MB with FPU consumes 1.12 W. However, in the case of mass product devices, the OP coprocessor will be delivered as an IP available in an ASIC library. Then, the final architecture can be based on a standard embedded (e.g. ARM) processor with a FPU and one or more OP coprocessors. Without the overhead of reconfiguration capabilities and the possibility to take advantage of power gating, a gain of one order of magnitude can be expected compared to GPP+GPU solutions. Such a solution would become viable in case of an exploding demand of chips to design self-content and energy efficient embedded systems for AR applications. Another promising solution is the use of upcoming low-power FPGA based on non-volatile memory with power gating capabilities [25]. This technology allows to switch-off coprocessors when the computation is complete, which means impressive power efficiency with reconfiguration capabilities.

8 Conclusions

Few research work has been conducted in the domain of embedded system design for mobile augmented reality applications in the context of emerging light see-through HMD. In this project, we have conducted an in-depth algorithm/architecture study and designed a complete system according to strong footprint constraints. The

solution that has been developed is dedicated but flexible. The approach has deliberately been focused on standard protocols and interfaces; it can be interconnected with usual inertial sensor and communication peripherals. This work results in a new approach for the design of AR-specific embedded and reconfigurable systems with three main contributions. The first one is the choice and the full specification of a gyroscope-free set of algorithms for position and attitude estimation, and this solution relies on the association and the adaptation, to the AR domain, of different previous contributions. The second one is the embedded system architecture, where it is introduced as a fast and simple object processor (OP) optimized for the domain of mobile AR. The architecture is especially optimized for data reuse and flexible since objects can be distributed on a given number of OPs. Finally, the OP implements a new pixel rendering method (IPS) implemented in hardware and that takes full advantage of OpenGL ES light model recommendations. The whole architecture has been implemented on various FPGA targets. The results demonstrate that expected performances can be reached and that a low-cost FPGA can implement multiple OPs. This solution is viable when reconfigurable architectures make sense, but the ultimate solution for mass market products would be a chip including a GPP and multiple OPs with power gating capabilities. Current undergoing work focuses on the choice of objects to be displayed in the user field of view. This study is made jointly with ergonomists and based on user feedbacks. The choice of objects to display will be implemented as a software service that will be in charge of the local memory content.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

This work has been supported by the DGA (french defense department) and has greatly benefited from discussions with Dr. John Williams about system architecture and Linux implementation on FPGA.

Received: 2 August 2014 Accepted: 7 November 2014

Published online: 21 January 2015

References

1. D Li, RJ Landry, P Lavoie, Low-cost MEMS Sensor-based attitude determination system by integration of magnetometers and GPS: a real-data test and performance evaluation. *IEEE Position Location and Navigation Symposium*, 1190–1198 (2008)
2. S Nasiri, A critical review of MEMS gyroscopes technology and commercialization status. Technical report, InvenSense (2010). <http://invensense.com/>
3. K Mirza, K Sarayeddine, Key challenges to affordable see through wearable displays: the missing link for mobile mass deployment. Technical report (2012)
4. AR Lingley, M Ali, Y Liao, R Mirjalili, M Klöner, M Sopanen, S Suihkonen, T Shen, BP Otis, H Lipsanen, BA Parviz, A single-pixel wireless contact lens display. *J. Micromechanics Microengineering*. **21**(12), 125014 (2011)
5. A Herout, M Zacharias, M Dubska, J Havel, in *Mixed and Augmented Reality (ISMAR) 2012 IEEE Int. Symp. On. Fractal marker fields: No more scale limitations for fiducial markers*, (2012), pp. 285–286
6. AI Comport, E Marchand, M Pressigout, F Chaumette, Real-time markerless tracking for augmented reality: the virtual visual servoing framework. *IEEE Trans. Visualization Comp. Graph.* **12**(4) (2006)
7. J Karlekar, SZ Zhou, W Lu, ZC Loh, Y Nakayama, D Hii, in *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium On. Positioning, tracking and mapping for outdoor augmentation*, (2010), pp. 175–184
8. X Yang, TT K-Cheng, in *International Symposium on Mixed and Augmented Reality (ISMAR)*. LDB: An ultra-fast feature for scalable augmented reality on mobile devices, (2012)
9. S Feiner, B MacIntyre, T Hollerer, A Webster, in *Wearable Computers, 1997. Digest of Papers, First International Symposium On. A touring machine: prototyping 3D mobile augmented reality systems for exploring the urban environment*, (1997), pp. 74–81
10. MA Livingston, LJ Rosenblum, SJ Julier, D Brown, Y Bailloit, J Edward, S II, JL Gabbard, D Hix, in *In Interservice/Industry Training, Simulation, and Education Conference. An augmented reality system for military*, (2002), p. 89
11. J Kingyens, JG Steffan, The potential for a GPU-like overlay architecture for FPGAs. *Int. J. Reconfigurable Comput.* **2011** (2011). doi:10.1155/2011/514581
12. A Munshi, D Ginsburg, D Shreiner, *OpenGL(R) ES 2.0 Programming Guide*. (Addison-Wesley, Boston, 2008)
13. JL Marins, X Yun, ER Bachmann, RB Mcghee, MJ Zyda, in *Intelligent Robots and Systems, 2001. Proceedings 2001 IEEE/RSJ International Conference On. An extended Kalman filter for quaternion-based orientation estimation using marg sensors*, vol. 4, (2001), pp. 2003–20114
14. A Waegli, J Skaloud, P Tome, J-M Bonnaz, in *ION-GNSS 2007. Assessment of the integration strategy between GPS and body-worn MEMS sensors with application to sports*, (2007)
15. KH Kim, JG Lee, CG Park, Adaptive two-stage extended Kalman filter for a fault-tolerant INS-GPS loosely coupled system. *Aerospace Electron. Syst. IEEE Trans.* **45**(1), 125–137 (2009)
16. E-H Shin, N El-Sheimy, in *Position Location and Navigation Symposium, 2004. PLANS 2004. An unscented Kalman filter for in-motion alignment of low-cost IMUs*, (2004), pp. 273–279
17. W Koo, S Chun, S Sung, YJ Lee, T Kang, in *National Aerospace & Electronics IEEE Conference (NAECON)*. In-flight heading estimation of strapdown magnetometers using particle filters, (2009)
18. D Gebre-Egziabher, GH Elkaim, JD Powell, BW Parkinson, in *Position Location and Navigation Symposium, IEEE 2000. A gyro-free quaternion-based attitude determination system suitable for implementation using low cost sensors*, (2000), pp. 185–192
19. J Bijker, W Steyn, Kalman filter configurations for a low-cost loosely integrated inertial navigation system on an airship. *Control Eng. Pract.* **16**(12), 1509–1518 (2008)
20. P-F Guo, H Qiu, Y Yang, Z Ren, in *Position Location and Navigation Symposium (PLANS)*. The soft iron and hard iron calibration method using extended Kalman filter for attitude and heading reference system, (2008)
21. J Bresenham, Algorithm for computer control of a digital plotter. *IBM Syst. J.* **4**(1), 25–30 (1965)
22. S Wuytack, J-P Diguët, F Catthoor, H De man, Formalized methodology for data reuse exploration for low-power hierarchical memory mappings. *IEEE Trans. VLSI Syst.* **6**(4), 529–537 (1998)
23. M Franklin, in *RTO Human Factors and Medicine Panel (HFM) Workshop. The lessons learned in the application of augmented reality* (NATO West Point, NY, USA, 2006)
24. R Vaslin, G Gogniat, J-P Diguët, E Wanderley, R Tessier, W Bursleson, A security approach for off-chip memory in embedded microprocessor systems. *Microprocess. Microsyst.* **33**(1), 37–45 (2009)
25. D Suzuki, M Natsui, A Mochizuki, S Miura, H Honjo, K Kinoshita, H Sato, S Ikeda, T Endoh, H Ohno, T Hanyu, Fabrication of a magnetic tunnel junction-based 240-tile nonvolatile field-programmable gate array chip skipping wasted write operations for greedy power-reduced logic applications. *IEICE Electron. Express*. **10**(23) (2013)