Journal of Network and Computer Applications 45 (2014) 181–190

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Distributed FPGA-based architecture to support indoor localisation and orientation services

Julio Dondo *, Felix Villanueva, David Garcia, David Vallejo, Carlos Glez-Morcillo, Juan Carlos Lopez

Escuela Superior de Informtica - University of Castilla-La Mancha, Ciudad Real, Spain

ARTICLE INFO

Article history:
Received 9 June 2013
Received in revised form
20 May 2014
Accepted 20 July 2014
Available online 1 August 2014

Keywords:
Indoor localisation and orientation
Dynamic reconfiguration
FPGA
Partial reconfiguration
Distributed systems

ABSTRACT

This paper introduces a proposal for an indoor localisation and orientation distributed service built on a dynamically reconfigurable platform. The integration of cameras in consumer electronic devices such as mobile phones, tablets, etc. allows the adoption of new methods based on video streaming analysis by a mobile device video camera that can enhance two essential features for a successful navigation experience: localisation and orientation. These features are important in services such as life assistance, direct marketing or localisation. The proposed infrastructure is based on the integration of heterogeneous resources under the umbrella of the distributed object paradigm. Our ultimate goal is to provide an efficient implementation of multi-user indoor localisation and orientation services.

1. Introduction

Indoor localisation of personal consumer electronics devices (e.g., smart phones, tablets and laptops) is a key problem, which, if addressed with enough accuracy and consideration of the orientation of device (e.g., the user), will lead to other advanced services. Indoor navigation, proximity marketing, security concerns, personal localisation, multitude monitoring, augmented reality, etc. are examples of services that require a localisation and orientation component. This group of services establishes a set of requirements in terms of accuracy and efficiency for a localisation service.

As we will see later, one of the most promising approaches in localisation/orientation of users in indoor environments is the analysis of video streaming from the consumer electronic devices that users carry. While analysing the video stream of the environment where the user is, we can extract the position and orientation of the user with several degrees of accuracy as a function of the analysis algorithm.

The algorithms used in this application field have two common characteristics; first, they require a previous exhaustive characterisation of the environment, which generates a large database. As it will be detailed later, this characterisation of the environment extracts a set of points of interest (corners, windows, etc.) that are used to match against those extracted from the images taken by the user. The general idea is to find those points of interest in the users video streaming to infer his/her position/orientation. The other characteristic in common is the computer power requirements of these algorithms to perform an efficient video analysis and an exact comparison with the database of the environment.

These two characteristics make the implementation of the algorithms in consumer electronic devices undesirable due to the inefficiency of transmitting the environmental database to all devices present in that environment. Airports, railway stations, commercial centres, public services building, etc. are examples of the environments where these types of applications can be deployed. Public security concerns supply other limitations and restrictions to sending the database to all consumer electronics devices.

In this paper we introduce an approach in which the video streaming analysis is performed by a set of specialized hardware which after processing send the localisation information to users with extra information to facilitate user localisation/orientation based on augmented reality (AR).

Our main motivation for this work was the Elcano project (Villanueva et al., 2011). This project was devoted to guiding handicapped people in indoor environments. The indoor positioning

* Corresponding author.
E-mail addresses: Juliodaniel.Dondo@uclm.es (J. Dondo), Felix.Villanueva@uclm.es (F. Villanueva), Dave.Alcarin@gmail.com (D. Garcia), David.Vallejo@uclm.es (D. Vallejo), Carlos.Gonzalez@uclm.es (C. Glez-Morcillo), Juancarlos.Lopez@uclm.es (J.C. Lopez).

http://dx.doi.org/10.1016/j.jnca.2014.07.029
1084-8045/

information of the Elcano project was delegated to different modules using different technologies (Wi-Fi indoor localisation, tag identification, etc.). In that project, we realised that the only methodology which provides us with user localisation information in a sufficiently precise manner is the analysis of video streaming provided by the user's consumer electronic device. Additionally, this methodology is the only one that provides us with the user's orientation information without any special hardware and/or modification of the user's device.

The specialized hardware-based solution we propose in this paper is based on the grid computing model, where through the use of heterogeneous grid formed by reconfigurable devices, mainly FPGAs, integrated under the umbrella of the distributed object paradigm with general purpose processors, we make an efficient implementation of indoor localization and orientation services. We model a multi-user architecture for including in user devices a generic application which sends video streaming to the FPGA Grid getting its position and orientation. The application running in the users mobile device is generic and can be used for position and navigation in any building. The FPGAs accelerate the tasks of localisation algorithm thus allowing us to attend multi-user requests simultaneously without loss of performance. This feature of multi-user requests' attendance is performed deploying as many accelerator as needed using partial reconfiguration capability of FPGA.

This paper extends the work presented in Villanueva et al. (2013), where we presented an initial architecture. In this paper, we show the final architecture in detail, together with the performance results of the prototypes. As we will see later, the results are really promising and our architecture could be used for numerous applications. Additionally, the authors of this paper are convinced that if consumer electronic devices like smart glasses succeed, any smart environment will implement architecture broadly similar to the one presented in this work.

This paper is structured as follows: Section 2 presents similar efforts focusing on algorithms used for position and/or orientation by mean video streaming analysis. Section 3 addresses the high performance computing platform and the proposed approach. Then, Section 4 describes the works, overall goals and components and shows the prototype and performance of our design. Finally, in Section 5, we summarise main conclusions of the work and future directions.

2. Related works

Location and orientation represent crucial data to provide users with services such as indoor navigation, proximity marketing or even augmented reality. Unfortunately, each service has specific needs (e.g. the number of variables to provide) so that different approaches must be considered to provide an efficient and accurate service.

One of the most important variables is the environment, which can be indoor or outdoor. On the other hand, the user location and orientation is based on the tracking process. Depending on the tracking devices in use, different techniques can be adopted (Papagiannakis et al., 2008; DiVerdi and Hollerer, 2007), such as optical (marker-based or markerless), GPS, Wi-Fi, RFID, accelerometers, inertial, and hybrid. As previously mentioned, these technologies have different levels of efficiency and accuracy and depend on the type of system being developed. Particularly, optical tracking aims at obtaining the camera pose with 6-depth of field (DOF) (location and orientation by means of a 4×4 matrix).

This work is especially concerned with real-time, optimal markerless approaches (Comport et al., 2006) in indoor environments. Optimal markerless approaches are becoming more and more popular because there is no need to deploy markers throughout the environment to carry out the tracking process. Integrating markers in prototypes can be a suitable choice to perform experiments and debug prototypes. However, the use of these markers (or fiducials) does not scale well when working on large environments. On the other hand, the recent advances in image processing have greatly contributed to boosting the development of camera-based location and orientation systems.

Within the context of optimal markerless techniques, the Features from Accelerated Segment Test (FAST) Corner Detection algorithm (Rosten and Drummond, 2005, 2006) represents the common ground for a large number of real-time 3D model-based tracking techniques. The main motivation for developing FAST, according to the authors, was the need for systems to address the rapid translations, rotations and accelerations that take place when using tracking devices, particularly in indoor environments.

The main advantage to using tracking methods based on feature points is that they provide a descriptor that can be used to perform identification between consecutive frames. Such descriptors must be as robust as possible, avoiding changes as a result of the environment variations (e.g., illumination and/or perspective). On the other hand, the relationship between edges (i.e., edge-based tracking) is computed according to proximity.

The FAST algorithm can be considered as an evolution of the technique proposed by Vacchetti et al. (2004). The authors claim that, in scenarios with a large number of feature points, it is not realistic to maintain an updated list of key frames. Within this context, the technique discussed in Rosten and Drummond (2005) is based on edges and corner points computed by the FAST algorithm, providing a model that estimates the movements between consecutive frames. In addition, this model allows dealing with significant movements by computing the probability for each detected characteristic (feature point or edge) to previously appear.

From the point of view of hardware, the use of low-cost devices has been explored during the last few years to improve the performance of location and orientation systems. For example, the work discussed in Hedley et al. (2008) proposes a novel radio location system for indoor tracking and navigation designed for emergency first responders. Within the context of image analysis, other authors, such as in Cho et al. (2006), make use of a particle filter to track moving objects in real time. Speed is again improved, thanks to the use of Field Programmable Gate Arrays (FPGAs).

In the application area of augmented reality systems, there are several proposals that address the challenge of robust tracking by means of reconfigurable hardware. For instance, in Piekarski et al. (2004), the authors discussed a hand tracking solution in a reconfigurable computer to not only reduce power consumption but also improve the efficiency of the tracking system. Related to the work proposed in this paper, in Guimarães et al. (2007), a platform to support developers of augmented reality applications is discussed. The infrastructure of this platform is built on top of FPGAs to contribute to high-speed processing and low power consumption. Instead of focusing on a particular algorithm, the authors state that multiple image processing algorithms can be implemented and integrated into the platform.

Although the use of low-cost hardware is becoming more and more relevant to address tracking and, therefore, localisation and orientation, there are other approaches that make use of parallel algorithms instead of computing single frames on different devices. The work discussed in Johnston et al. (2005) presented a number of parallel algorithms to address real-time positioning in augmented reality systems. Furthermore, the authors made a quantitative comparison between their algorithms and more traditional edge-based systems.

The use of FPGA has been shown to be more efficient for image processing than other approaches such as GPUs and/or CPU (Sriram et al., 2010). In the field of feature extraction, a good example of these comparisons can be found in Possa et al. (2013). In this work a hardware implementation of several algorithms for corner and edge detection has been implemented using FPGA and several comparisons have been made against GPU and CPU approaches. However all these works do not use dynamic reconfiguration capabilities in order to scale multi-stream processing as presented in this work. An approach for indoor location based on SLAM (Simultaneous localisation and Mapping) algorithm is presented in Zubiel and Long (2012), again authors only use the improved performance of FPGA implementation dealing with multiuser environment assigning a device per stream. However, it is not oriented to massive use of people with standard devices such as smart phones or tablets, but to track the location of a firefighter indoors using specialized devices formed by a camera plus an FPGA for image processing in order to reduce the amount of data to be transferred .

Besides the improvement shown in several features; such as the performance, the efficiency, and the number of dedicated resources; the most important FPGA vendors have included a special characteristic to some of their products: the dynamic reconfigurability facility. One of the benefits motivated by the dynamic reconfigurability is the fact that the designer can dynamically insert new functionalities without redesigning the system or moving to a bigger device. Furthermore, it is possible to adapt the FPGA to different scenarios, by modifying the functionality or the performance of some tasks running on it. In addition, the dynamic reconfiguration reduces costs and area enabling smaller designs by time-multiplexing portions of the available hardware resources. An example of this fact is presented in Manet et al. (2008), in which a single hardware platform can support different waveforms by using one or another at run-time according to the user/application requirements. The dynamic reconfiguration feature provides additional advantages, such as reducing the bitstream storage needs, the synthesis time or accelerating the computation (Dye, 2011; Kao, 2005). The characteristic of dynamic reconfigurability is very convenient for all those applications that require real-time adaptability (Viswanathan et al., 2012). An example of using dynamic reconfigurability in features detection is presented in Mühlbauer and Bobda (2006) where feature trackers are implemented for object tracking purposes. In this work dynamic reconfiguration is used to increase the flexibility of the feature tracker changing the input module according to the source of the incoming image stream. This work does not use partial reconfiguration to replicate feature tracker modules in order to attend multiuser requirements.

The work presented in this paper makes use of the advantage offered by FPGAs in terms of power consumption, parallelism and reconfiguration capability. It integrates CPU computational power with FPGA to accelerate the execution of algorithms involved in localisation and orientation for indoor services to create an efficient and scalable service provider. The scalability is based on the replication capacity of the part of the algorithm implemented in hardware to serve more users at same time. Due to the acceleration of processing produced by the implementation in hardware, this component can attend more users simultaneously and can be replicated in the FPGA on demand. This on-demand replication means that the FPGA can instantiate or remove components according to the demand. In that way, the device only uses the required resources, avoiding inactive resources that consume power. The instantiation on demand is performed using a component named the Reconfiguration Engine that partially configures the FPGA through its capability of implicit reconfiguration as described later in Section 4.2.

3. Architectural overview

This project has two candidate approaches for architecture design, depending on where the image processing is performed and where the service for localisation and orientation is located. First, the centralised service approach implies that each device sends a video stream to a processing node (e.g., a CPU plus an FPGA) that implements visual tracking and returns the position information to the device. In this approach, the main advantage is that the user only needs to install a generic application in their consumer electronic device.

All the work to compute the algorithm is performed in a server and sent to the user when required. From a security point of view, the environment keeps all critical confidential information within the system. The main drawback of this approach is scaling the information technology IT infrastructure of the environment according to the expected number of users.

The second approach is embedding the algorithm in the user device using a low power/low cost FPGA. This approach requires downloading the environment characterisation to each portable device. From a practical point of view, in short term this work takes the centralised approach, but the hardware implementation of the algorithm supports both approaches.

For a model of either of the two approaches to the architecture, an object-oriented distributed middleware is used. Using this type of middleware avoids encountering platform heterogeneity, and the network is transparent from the developers point of view. In fact, from the developers point of view, all resources of the environment are modelled as software objects, independent of where they are running, how they are implemented and which programming language was used. Even hardware algorithms running in an FPGA are accessed remotely as local software objects, thanks to our extension of this type of middleware (Barba et al., 2010).

An endpoint is defined as the address of a distributed software object in the IT environment architecture. Any software object is identified by an endpoint, which in its simpler format is an IP address together with a TCP port number. The methods of a software object can be invoked knowing the endpoint of that specific software object and the class to which that object belongs. Together with the endpoint, it is necessary to define an object interface; this interface is the contract for anyone who wishes to use the object. The key interfaces of our system are

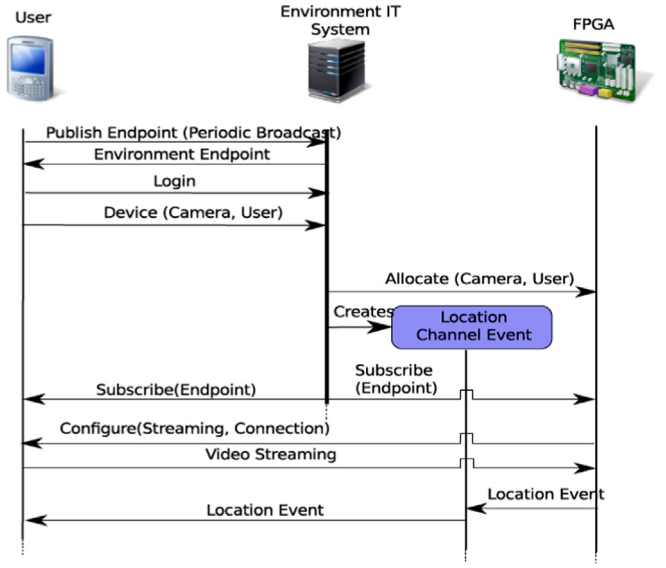

Fig. 1. Simplified invocation schema in the architecture.

```
interface Bootstrap{
int Publish(Endpoint e);
};

interface Configure{
int Login(Credentials c, User u);
int Device(Camera c, User u);
};

interface HPC{
int Allocate(Camera c, Endpoint e);
int Subscribe(Endpoint e);
};

interface MobileDevice{
int Environment(Endpoint e);
int Subscribe(Endpoint e);
int Configure(Endpoint streaming,Connection ch);
int LocationEvent(Point x);
};
```

To improve the readability of the interfaces, superfluous details (exceptions, error return values, etc.) and class definitions have been removed. Structures such as Credentials, User, Camera and Connection can be interpreted as dictionaries with a set of ⟨key, value⟩ pairs (really, they are classes). The Point structure represents position and orientation of a device:

```
struct Point{

int userID;
float x; // x coordinate
float y; // y coordinate
float z; // z coordinate
matrix coord; // 3x3 rotation matrix
};
```

We can obtain a global simplified overview of the interaction between these interfaces in Fig. 1. When a user enters an indoor environment with a mobile phone or tablet, the device associates with the indoor wireless network. As soon as the device is connected to the network, the app previously installed on the mobile device, which is generic and valid for any environment with the proposed architecture deployed, starts to send a periodic broadcast, publishing its endpoint.

When the environment IT system detects a broadcast from a new device, it uses the endpoint published in the broadcast message to invoke a method in the object running in the users device with the environment endpoint. At this moment, both the users device and the environment IT system can communicate because both have knowledge about each others endpoint. Just after this discovery phase, the device can communicate its credentials, user characteristics and camera profile to the environment. The credentials provide for a secure login to the system and the user characteristics can improve the personalised behaviour of the system.

With the camera profile, the environment IT system makes a resource allocation to start the users video streaming analysis. Additionally, it creates a Location Channel Event (LCE) and sends subscription information to the users app and FPGA-based system. The users device and the FPGA-based system subscribe to the LCE (for simplification, this phase is omitted in Fig. 1). At this moment, the FPGA-based system sends the characteristics of the stream (e.g., size, frames per second, format, etc.) and the endpoint where the users app has to send the stream. Finally, the users app sends video streaming to the configured endpoint. As soon as the FPGA-based module identifies the location of the users device according to the streaming video, it sends location events to the LCE. This LCE propagates this location event to all peers subscribed to the channel event as consumers, including the user devices.

The localisation information is sent to the user in the form of augmented reality (AR), using the maps created by applying Parallel Tracking and Multiple Mapping (PTAMM) methods (Klein and Murray, 2007). This method performs the tracking of common features separately between the maps and the mapping process. The tracking method uses the information provided by the current map to find a camera pose for the new frame that is being evaluated.

4. Experimental results

The approach presented in this paper has been prototyped and tested in a reconfigurable hardware-based platform. This prototype includes a PC and a grid of FPGAs (R-Grid) connected to the processor through Gigabit Ethernet.

The R-Grid system (Dondo et al., 2011) is a research system developed to exploit the advantages of applications with parallelizable algorithms offering multi-user and multi-application computational environments over the same scalable group of hardware resources. This set of hardware resources is formed basically for different classes of partially reconfigurable XILINX FPGAs, modeled in such a way that allows the integration of the heterogeneous resources, offering a simple and efficient computational model to the end-user. Each FPGA is divided into two parts: (a) a dynamic area forming the user space for the deployment of applications, and (b) a static area that implements the reconfiguration service, interfaces and communication mechanisms, necessary for the deployment and for the access of the user applications.

In the CPU, the activities concerning administration, user registry and operations of the environment IT system are performed. Once the FPGA-based system has sent the characteristics of the stream and the endpoint to the users app, the user can start sending video.

4.1. Video streaming performance

With our platform, as we will see later, the bottleneck of the system will pass from computing power capacity to wireless network capacity.

Figure 2 shows the throughput of a single MPEG4 connection from a tablet to the processing node using 802.11n. For the MPEG4 characterisation, we started our streaming application and the user picked up the tablet and was immobile between 1 and 30 s. Then, the user started to move around, making a video stream of the environment. Thus, the throughput increases significantly after 30 s. The frames per second value used in this video stream are 50 and the size of the image is 640×480 pixels.

These results allow us to be very optimistic according to the number of concurrent users in the same access point using the proposed platform.

In this approach, the most time-consuming parts of the different algorithms used for tracking are accelerated through the design of specific hardware modules and implemented using the partial reconfiguration capability of FPGAs.

4.2. The partial reconfiguration service

As mentioned before, each FPGA forming part of the R-Grid system is formed by two main parts (see Fig. 3): (i) a static part that contains the mechanism for partial reconfiguration and all the components necessary for proper operation of the system (a network interface to provide communication with external world and an embedded microprocessor to perform part of the tracking algorithm); (ii) a dynamic reconfigurable part where accelerated segments of the algorithm are instantiated.

This proposal uses two main components for the partial reconfiguration process: a Reconfiguration Controller and a Reconfiguration Engine. Both are entirely implemented in hardware in the static part of the FPGA. They provide the necessary tools to address dynamically reconfigurable designs at run time for operating systems or standalone applications (Dondo et al., 2013).

The reconfiguration process is managed by the Reconfiguration Controller, which has the responsibility to execute the necessary tasks for reconfiguration. This component offers a set of reconfiguration services through a simplified interface, including bitstream transference, the start and stop of individual instances, status

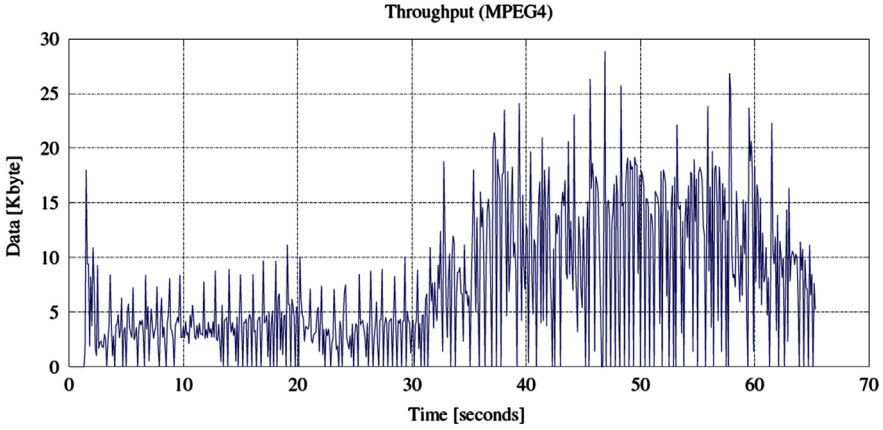

Fig. 2. Throughput of a single MPEG4 connection.

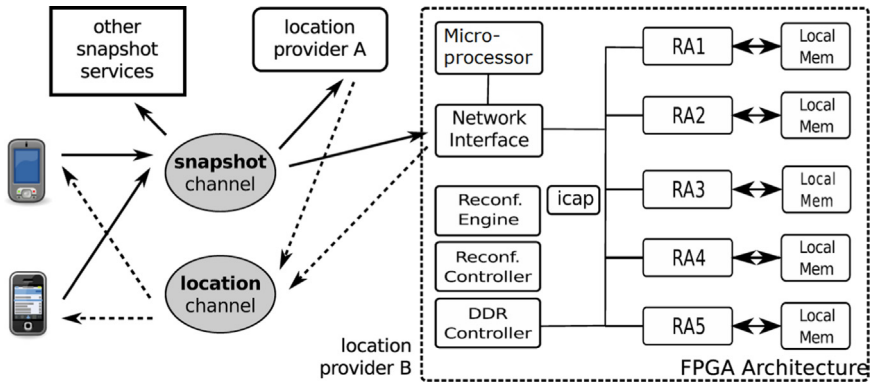

Fig. 3. Architecture overview.

requests, and location and relocation of components. From the architecture point of view, the implementation of the reconfiguration service through a dedicated hardware component instead of using a software approach represents two orders of magnitude in the improvement of the reconfiguration speed (Dondo et al., 2013).

Each hardware-accelerated module is implemented in one reconfigurable area (RA) (see Fig. 3). The Reconfiguration Engine acts as a dedicated Direct Memory Access (DMA), taking the partial bitstream from memory and sending it to the Internal Reconfiguration Access Ports (ICAP). This architecture allows the system to instantiate or replicate modules according to the application requirements, a very useful characteristic for those systems where power consumption is critical.

Each reconfigurable module can be replicated according to the demand by invoking a method in the Reconfiguration Controller, which according to the available reconfigurable resources, starts the reconfiguration through the Reconfiguration Engine component. The main advantage of real-time systems designed using FPGAs is that the processes should not compete for processing time because they are inherently parallel. They may, however, compete for area or the use of specific resources (BRAM, DSP), especially when shared.

Moreover, the dynamic reconfiguration capability of FPGAs facilitates resource distribution, reduces resource demand and increases the use of shared resources over the system life cycle, assuming (i) a negligible cost of context change and (ii) the independence of the tasks composing the system. Therefore, the dynamic reconfiguration process must be carried out in accordance with the execution scenario.

4.3. The acceleration of the algorithm

In this work, we have implemented the PTAMM tracking method using a grid of FPGAs to accelerate a certain part of the method. The part chosen for acceleration was the feature detection activity using FAST. The PTAMM builds a map of the scene, matching objects that are observed as matched features in images taken from different points. Simultaneously, it uses this map and the current image to make an estimation of the camera pose. During the initialisation phase, PTAMM builds an initial map using two frames of the same scene taken from two different points and separated by a known distance to set the scale of the map. Once both images have been captured, Faugeras and Lustman (1998) method is used to compute a new camera pose and to triangulate scene points (Castle et al., 2008). FAST is used to detect feature points in the first frame and these points will be searched for in all incoming frames. These images, called keyframes, are processed by FAST to detect features that will be used to match with new keyframes.

Once the image arrived to the FPGA, the original frame is first converted to 8-bit data (grey-scale image) and this new frame is the input of the FAST algorithm. To obtain a better performance, we developed a hardware-based FAST module (HW-FAST) that runs the FAST algorithm for corner detection, developed so that it can determine a corner from a parameterisable set of surrounding pixels without incremental computational time. In our approach, each pixel is evaluated in one clock cycle without the need to perform compression on the patterns and without implementation of a decision tree because the comparison is performed for all pixels at same time.

In Dohi et al. (2011), we found an implementation of the FAST algorithm based on corner pattern compression detection. Our work is based on the implementation performed in Kraft et al. (2008) but without non-maximum suppression. The implementation performed in this work splits the algorithm into two main parts. The first part contains a mechanism to select the pixel to be evaluated and the set of 16 surrounding pixels that will be used to determine if the central pixel is a corner or not. The second part is formed by a parameterisable comparator that evaluates the selected set of pixels.

The first part is performed in two stages: initialisation and selection. During the initialisation stage, the mechanism prepares the video stream for processing. As FPGAs receive video streaming sequentially, to analyze each point and its corresponding 16 surrounding pixels it is necessary first to store the video streaming in FPGA internal memory. For a video resolution of pixel dimensions $a \times b$, the first seven lines are stored in seven, 8-bit-wide a-depth FIFO stacks, during the initialisation process. Once the initialisation is finished, the first bytes of each one of the seven FIFOs are shifted to seven 8-bit registers as depicted in Fig. 4. These 49 registers will be used to select the 17 pixels for treatment. The value of registers 0–15 (depicted in the figure) plus the central pixel are sent to the comparison stage. Once one set of 17 pixels is selected, the seven FIFOs send a new pixel into the chain of seven registers and a new pixel feeds the FIFOs structure in the next clock cycle. Then, a new set of pixels can be selected from the same 17 registers. This approach allows for evaluating 17 pixels each clock cycle. During each clock cycle, a new pixel enters into the FIFO stack until the frame is completely evaluated.

In the second part (the comparison stage), the value of the central pixel is incremented and decremented by a threshold value and both results will be used for comparing with the surrounding pixels to determine if the central pixel is a corner. Then, the 16 pixels previously selected are compared with the new value of the central pixel. These 16 comparisons are performed at the same time in one clock cycle, and the result (a 1 for yes or a 0 for no) is stored in two 16-bit registers, one for the incremented pixel and the other one for the decremented pixel value. Each bit of this 16-bit register indicates the result of the comparison. The next step consists in checking if n consecutive pixels of these 16 pixels are ones, indicating that the central point evaluates to a definite corner.

The comparison and the evaluation of ones are performed in a two-stage pipeline as shown in Fig. 5. Stage 1 is formed by two sets of sixteen 8-bit comparators. The output of each comparator is a bit (1 or 0) depending on the result of comparison. For the first group, each comparator outputs a 1 if the surrounding pixels are greater than the central pixel plus the threshold. For the second group, each comparator outputs a 1 if the surrounding pixels are less than the central pixel minus the threshold. In Stage 2, each 16-bit vector formed by the results of comparators is evaluated to determine if there are 9 consecutive bits equal to 1, indicating that the evaluated pixel is a corner. This pipelined implementation allows us to determine in each clock cycle if a pixel is a corner or not.

The implementation described in this section allows us to evaluate a characteristic point without considering the decision tree as described in the software version. The main difference between this approach and the work presented in Kraft et al. (2008) relies on its implementation and the way that this implementation makes use of the partial reconfiguration characteristics of the FPGA described in Section 4.2.

This characteristic is used in two ways, firstly to define the type of implementation by means of the reconfiguration of parameters such as the size of FIFOs needed to store the image as it is giving online support for different image sizes, the amounts of surrounding pixels to be considered, and the threshold value; and, secondly, to instantiate as many HW-FAST components as needed according to the amount of users, allowing the provision of a very efficient location service. This efficiency can be measured characterizing the time consumed in evaluating each frame compared with the

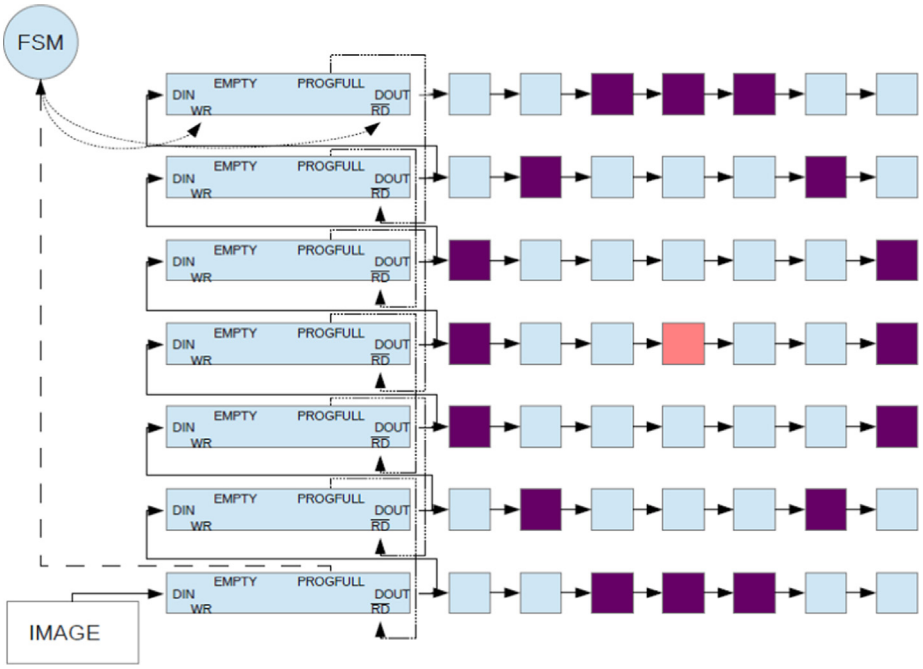

Fig. 4. Data organisation for corner detection.

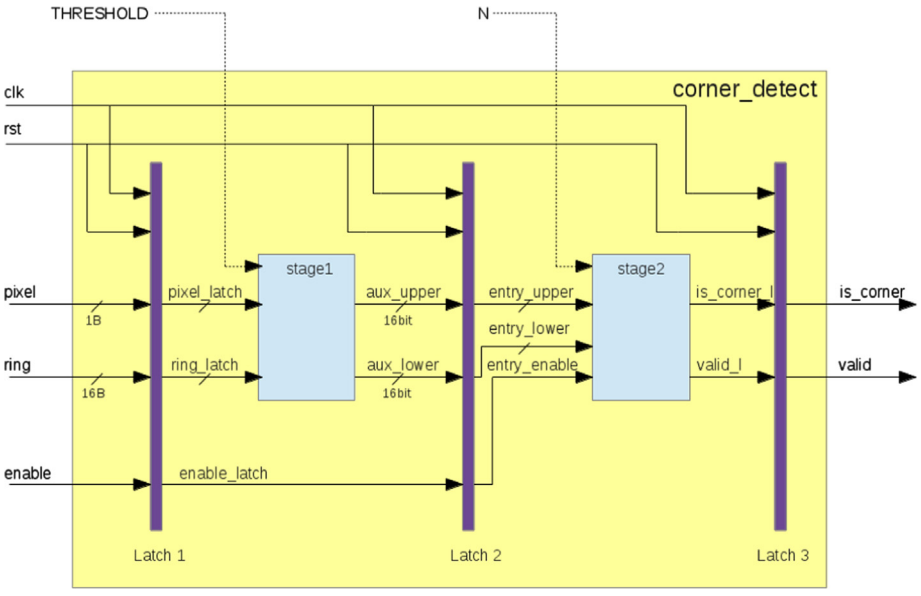

Fig. 5. Corner detection pipelining.

software version and determining the amount of users that this approach can assist simultaneously. Besides, as it is described in the next section, this approach have also designed the HW-FAST component in such a way that allow its integration and replication in one FPGA or in a grid of FPGAs.

The total time consumed in the evaluation of a frame is represented in the following equation:

$$Teval(clockcycles) = T\_init + T\_frame \quad (1)$$

T_init is the time that take the initialisation stage to fill the FIFOs, and *T_frame* is the time consumed during the evaluation of each point of the picture:

$$T\_init = a*7 \quad (2)$$

$$T\_frame(clockcycles) = (a-6)*(b-6) \quad (3)$$

where a and b represent the number of columns and files of the frame, respectively.

Then, for a frame with a 640×480 resolution, the algorithm will take 300,516 clock cycles for pixel selection plus 4480 clock cycles of initialisation.

The HW-FAST component was implemented in Virtex5 and Virtex6 FPGAs running at 200 MHz. At this frequency, it is possible to evaluate a frame for every 1.5 ms. Therefore, one HW-FAST module is able to attend the requirements of HD streaming video. Moreover, the power consumption of the FPGA is lower compared with CPU power consumption. In Fig. 6, a comparison of processing times using the HW-FAST approach and a software execution

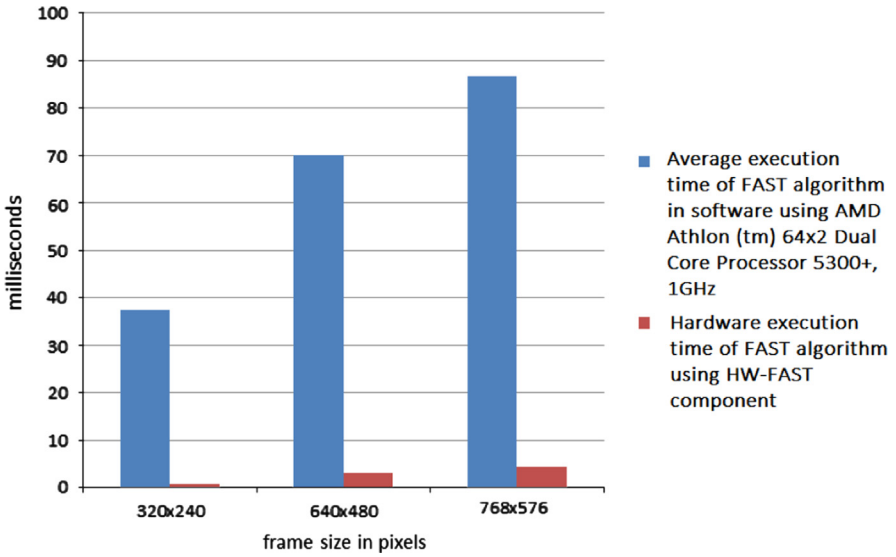

Fig. 6. Comparison of FAST algorithm execution time using author-compiled software vs. HW-Fast component.

of the Fast algorithm using the author-compiled code are shown. As the reader can see, the HW-Fast approach is one order of magnitude faster than software approach.

This approach can be used in two different ways. One for a centralised location and orientation service provider in which all logged users can make use of the service, or in a distributed manner in which each personal device has this service implemented in a low power FPGA. In centralised services, each HW-FAST component can attend more than 50 users simultaneously. For an efficient use of the HW-Fast component, the service provider sends the video stream to the component and the component sends back a file with the coordinates of each characteristic point. With these coordinates, the server recomposes the image, adding these points. With these points added to the map, in the mapping phase of the PTAMM methods, the location is identified and the Augmented Reality components are added to the image to compose the service output.

4.4. *Multi-user HW-FAST service*

One main challenge founded in the implementation of the location service described in this paper was scalability, which is a key point in this kind of systems. Implementing HW-FAST components as many as needed requires to give the support for reconfigurability on demand. On demand means to provide with the service to instantiate as many accelerators of the algorithm as needed, when the application and the number of users require it. For that, the reconfiguration service receives the reconfiguration invocation in order to instantiate the corresponding new accelerator to attend new users. In case that the number of users is increased, one FPGA may not be enough to provide the required services.

To solve this limitation the grid of FPGAs described previously was used to deploy several HW-FAST components in a distributed fashion. The grid was implemented using a standard PC acting as a server to which several FPGAs are connected through Gigabit Ethernet. The grid is managed by this system server which sends each user data-streaming to different HW-FAST components according to demand.

To provide a service addressing multi-requirements, besides the HW-FAST component, each FPGA contains a set of components related to the data administration as can be seen in Fig. 7. In this figure the static part of the FPGA contains the partial

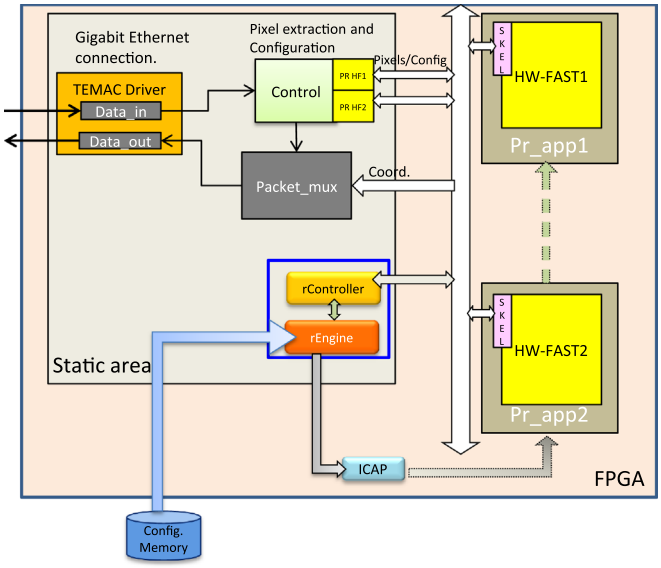

Fig. 7. Block diagram describing multi-HW-FAST implementation on FPGAs.

reconfiguration mechanism, the Ethernet port to perform the communication between FPGA and the server, the Packet-Mux component, and the Control component. The Control component separates the payload of the received message and sends the pixels of the image and the HW-FAST configuration data, such as the threshold and the amount of pixels to calculate the feature point, to each HW-FAST component. This new frame is the input of the HW-FAST component, which after completion of its work sends back a file with the coordinates of each characteristic point. The Packet-Mux component creates each file of coordinates when several HW-FAST components are working simultaneously, and creates the data packets to be sent through the Ethernet channel to each user.

The HW-FAST components, the Control and Packet-Mux component have been prototyped over the Xilinx Virtex 5 and Virtex 6 FPGA-based boards. These FPGAs were chosen owing its partial reconfiguration feature. For these tests in each FPGA several reconfigurable areas have been defined as represented in Fig. 7. In areas prApp1 and prApp2 we implemented two HW-FAST components allowing us to serve two groups of clients

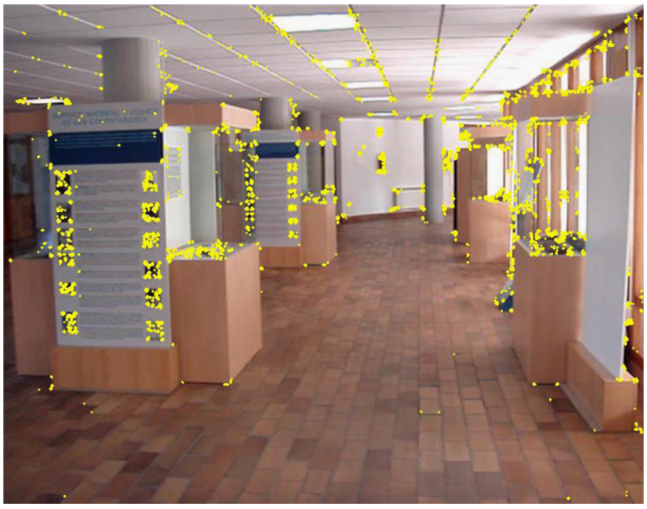

Fig. 8. Original image with features detected by HW-Fast in yellow. (For interpretation of the references to colour in this figure caption, the reader is referred to the web version of this paper.)

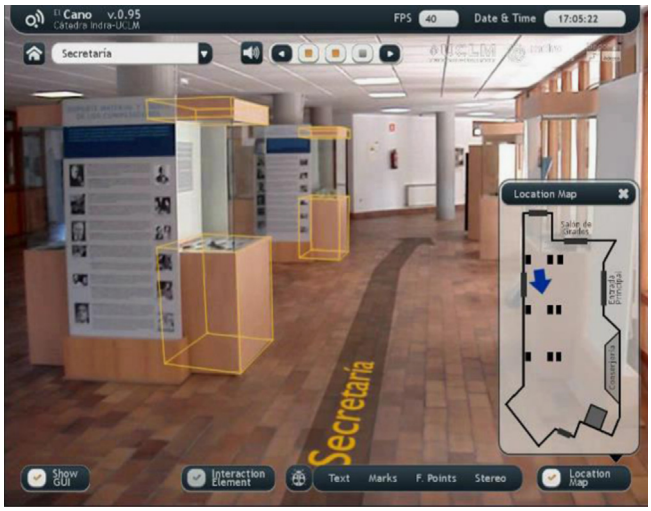

Fig. 9. Sample output of a localisation and orientation service. AR features have been added to the original image.

simultaneously. The system is designed in such a way that allows the implementation of more HW-FAST cores depending on the demand implementing more corner detection in reconfigurable areas when required.

As was established before, the implemented algorithm has the capacity to attend more than 50 users simultaneously. However, with this implementation the bottleneck of this type of systems passes from computing processing requirements to wireless network. The amount of concurrent users of our system is not limited by processing capability of FPGA but by communication mechanism between the user and the server, in this case the Gigabit Ethernet Channel. If the user portable device sends a streaming of video with a rate of 30 frame per seconds of 640×480 image resolution per frame, the Gigabit Ethernet channel will allow us to provide services only to around 10 users simultaneously. One way to solve this inconvenient could be reducing the size of wireless cells and incrementing its number, increasing in this way the number of user requirements addressed simultaneously.

An example of the localisation and orientation service provided by the architecture detailed in this work is presented in Figs. 8 and 9. In Fig. 8, the HW-Fast component has been applied to the original image to find feature points. Figure 9 shows the final image where the AR components were added to indicate the final destiny to the user considering his/her current position.

Table 1
Hardware cost of reconfiguration service and HW-Fast component.

Resources	*Reconf. Controller*	*Reconf. Engine*	*Hw−FAST Component*
Slices	1360	788	6617
FF	357	749	9250
LUTs	1830	1279	5241

The detection difference between the presented approach and the original version of the FAST algorithm in software is less than 2% of the detected corners. The following table shows the use of resources in the FPGA for both the reconfiguration service and the HW-Fast component (Table 1).

5. Conclusions

In this work, an infrastructure for an indoor localisation and orientation system has been presented, in which part of the service is implemented in FPGAs and is dynamically reconfigurable. This implementation is suitable for two types of environments: one offering centralised services in which an IT system provides this service for users with portable consumer electronic devices (e.g., a smartphone or tablet) who only have to install a generic application; and also a distributed system in which the service is performed in each device, developed with a low-power FPGA to comply with the computational exigencies of the service and saving on power consumption.

The main advantage of the proposed approach is the use of the partial reconfiguration capability of the FPGA to implement part of the algorithm in hardware. With the HW-Fast component defined for this tracking process, the computational time to evaluate each pixel is the same, one clock cycle, and is independent of the number of surrounding pixels used to identify a corner. This value is parameterisable according to the system requirements. Besides, dynamic reconfiguration of the FPGA allows us to scale the service to new users according to the demand. The work described in this paper also describes how to extend the service using a grid of partially reconfigurable FPGAs in order to provide multiuser support for larger systems.

As main contributions of this work we can summarize the following:

- In this work, the first multiuser object-oriented distributed platform for localisation and orientation services have been designed and implemented. One of the main contributions is a set of interfaces devoted to play a similar role than POSIX Interface in operating systems world. Any vendor could implement any of the parts of the localisation system (mobile client, algorithm, etc.) and be inter-operable with the rest of the architecture.
- One of the main issues of these types of architectures is the performance of localisation algorithm. This works presents the first FPGA-based grid approach for enabling a real multi-user architecture and improving algorithm execution. As far as we know, we are the first work where reconfigurable capacity of the FPGAs is used in this application field.
- The accuracy of this localization and orientation architecture depends on the implemented algorithm. The chosen algorithm PTAMM is one of the most accurate algorithms in this kind of issues. From an engineering point of view we go a step further providing with the whole architecture to be usable.

- To pass from a software algorithm to a multiuser hardware implementation is a research process that includes analysis of parallelisation issues and characterization of performance and communication aspects of different implementation alternatives.

With our work the bottleneck of the localisation and orientation service moves from computing processing requirements to wireless network, where the amount of concurrent users is limited now by the communication mechanism between the user and the server. One way to solve this inconvenience could be reducing the size of wireless cells.

This approach is suitable to be extended to new services such as evacuation information in disaster cases, or to guide handicapped people in indoor environments by means of AR features, offering to the user new information about the characteristic of the environment.

Acknowledgements

This work is supported by Spanish Government. Science and Innovation Department under project DREAMS (TEC2011-28666-C04-03), and by the Catedra Indra, University of Castilla-La Mancha.

References

Barba J, Rincon F, Moya F, Lopez J, Dondo J. Object-based communication architecture for system-on-chip design. In: Proceedings of the 2010 design of circuits and integrated systems. p. 374–9.

Castle R, Klein G, Murray D. Video-rate localization in multiple maps for wearable augmented reality. In: Wearable Computers, 2008. ISWC 2008. 12th IEEE international symposium on. p. 15–22.

Cho JU, Jin S-H, Pham XD, Jeon JW, Byun J-E, Kang H. A real-time object tracking system using a particle filter. In: 2006 IEEE/RSJ international conference on intelligent robots and systems. p. 2822–7.

Comport AI, Marchand E, Pressigout M, Chaumette F. Real-time markerless tracking for augmented reality: the virtual visual serving framework. IEEE Trans Vis Comput Graph 2006;12:615–28.

DiVerdi S, Hollerer T. Groundcam: a tracking modality for mobile mixed reality. In: IEEE Virtual Reality Conference 2007 March 10-14, Charlotte, North Carolina, USA, 2007.

Dohi K, Yorita Y, Shibata Y, Oguri K. Pattern compression of fast corner detection for efficient hardware implementation. In: Proceedings of the 2011 21st international conference on field programmable logic and applications, FPL '11. p. 478–81.

Dondo J, Barba J, Rincon F, Sanchez F, Fuente D, Lopez J. Distributed reconfigurable hardware for image processing acceleration. In: 2011 international conference on P2P, parallel, grid, cloud and internet computing (3PGCIC). p. 231–6.

Dondo JD, Barba J, Rincón F, Moya F, López JC. Dynamic objects: supporting fast and easy run-time reconfiguration in fpgas. J Syst Archit 2013;59:1–15.

Dye D. Partial reconfiguration of Xilinx FPGAs using ISE design suite. Technical Report. Xilinx; 2011.

Faugeras OD, Lustman F. Motion and structure from motion in a piecewise planar environment. Int J Pattern Recognit Artif Intell 1988:485–508.

Guimarães GF, Lima JaPSM, Teixeira JaMXN, Silva GD, Teichrieb V, Kelner J. Fpga infrastructure for the development of augmented reality applications. In: Proceedings of the 20th annual conference on integrated circuits and systems design, SBCCI '07. p. 336–1.

Hedley M, Humphrey D, Ho P. System and algorithms for accurate indoor tracking using low-cost hardware. In: 2008 IEEE/ION on position, location and navigation symposium. p. 633–40.

Johnston D, Fleury M, Downton A, Clark A. Real-time positioning for augmented reality on a custom parallel machine. Image Vis Comput 2005;23:271–86.

Kao C., Benefits of partial reconfiguration, Xcell Journal, Xilinx Corporation, 2005.

Klein G, Murray D. Parallel tracking and mapping for small ar workspaces. In: Proceedings of the 2007 sixth IEEE and ACM international symposium on mixed and augmented reality. ISMAR '07. p. 1–10.

Kraft M, Schmidt A, Kasinski AJ. High-speed image feature detection using FGPA implementation of fast algorithm. In: VISAPP (1)'08. p. 174–9.

Manet P, Maufroid D, Tosi L, Gailliard G, Mulertt O, Di Ciano M, et al. An evaluation of dynamic partial reconfiguration for signal and image processing in professional electronics applications. EURASIP J Embed Syst 2008;2008:1:1–1:11.

Mühlbauer F, Bobda C. A dynamic reconfigurable hardware/software architecture for object tracking in video streams. EURASIP J Embed Syst 2006;2006.

Papagiannakis G, Singh G, Magnenat-Thalmann N. A survey of mobile and wireless technologies for augmented reality systems. Comput Animat Virtual Worlds 2008;19:3–22.

Piekarski W, Smith R, Wigley G, Thomas B, Kearney D. Mobile hand tracking using FPGAs for low powered augmented reality. In Proceedings of the eighth international symposium on wearable computers, ISWC '04. p. 190–1.

Possa PR, Mahmoudi SA, Harb N, Valderrama C, Manneback P. A multi-resolution FPGA-based architecture for real-time edge and corner detection. IEEE Trans Comput 2013;99:1.

Rosten E, Drummond T. Fusing points and lines for high performance tracking. In: Proceedings of the tenth IEEE international conference on computer vision, vol. 2, ICCV '05. p. 1508–15.

Rosten E, Drummond T. Machine learning for high-speed corner detection. In: Proceedings of the ninth European conference on computer vision—Volume Part I. ECCV'06. p. 430–43.

Sriram V, Tsoi K, Luk W, Cox DD. A design-space exploration of biologically-inspired visual object recognition algorithms using CPUs, GPUs and FPGAs. In: Many-core and reconfigurable supercomputing; 2010.

Vacchetti L, Lepetit V, Fua P. Combining edge and texture information for real-time accurate 3d camera tracking. In: Proceedings of the 3rd IEEE/ACM international symposium on mixed and augmented reality. ISMAR '04. p. 48–57.

Villanueva F, Martinez M, Villa D, Gonzalez C, Lopez J. Elcano: multimodal indoor navigation infrastructure for disabled people. In: 2011 IEEE international conference on consumer electronics (ICCE). p. 611–2.

Villanueva F, Dondo Gazzano J, Villa D, Vallejo D, Mora C, Morcillo C, et al., Distributed architecture for efficient indoor localization and orientation. In: 2013 IEEE international conference on consumer electronics (ICCE). p. 57–8.

Viswanathan V, Ben Atitallah R, Dekeyser J-L, Nakache B, Nakache M. Dynamic reconfiguration of modular i/o ip cores for avionic applications. In: 2012 International conference on reconfigurable computing and FPGAs (ReConFig). p. 1–6.

Zubiel M, Long N. Firefighter indoor navigation using distributed SLAM (FINDS). Technical Report. East Lansing, Michigan: Faculty of the Worcester Polytechnic Institute; 2012.