A Software Architecture for Distributed AR Applications

Gregory de Oliveira Feijó* Leonardo Pavanatto Soares Vicenzo Abichequer Sangalli Thomas Volpatto de Oliveira Márcio Sarroglia Pinho

Virtual Reality Group - Computer Science School – PUCRS - Brazil

ABSTRACT

The recent advances in hardware technology expanded the scope of Augmented Reality applications to mobile devices such as tablets and smartphones. Given their limited battery resources, distributed computing becomes a tempting approach to develop AR applications. Although frameworks that support the development of distributed AR systems exist, they are mostly complex in their nature. In this article, we present a software architecture that mainly targets distributed AR systems. As a proof of concept, we developed three different applications based on this architecture. The applications use different devices such as smartphones, webcams and head mounted displays (HMD) and allows the user to interact with the system in many different ways, providing the user with realistic AR experience.

Keywords: Augmented Reality, Distributed Computing, Client-Server, Mobile, Software Architecture.

1 INTRODUCTION

Augmented Reality is commonly associated with the superposition of virtual elements in real images so that our perception of the real world is enhanced with virtual elements generated by computer devices such as desktops, laptops and mobile devices. According to Danado et al. [1], an AR system should provide a solution for the following tasks in order to create more realistic AR experiences:

- Registration of computer generated images of virtual elements on real images;
- Identification of the user position;
- Information retrieval;
- Data presentation.

Although the processing power of mobile devices is increasing over time, the real time processing of all these tasks and its limited battery duration makes AR impractical for a variety of applications. Distributed computing can overcome these limitations by splitting the application into modules that are responsible for small portions of the whole process. By cooperating together, the user can remotely interact with the system, save battery in case of mobile devices and still have a realistic AR experience.

There are some frameworks out there that use different architectures for AR applications[2][3]. The DWARF framework [2], for example, is a component-based framework based on the concept of distributed services in which each service is controlled by a service manager. The service manager defines the type of information the service needs, what information it can offer to other services and it establishes the connection between local and remote services. The main drawback of the DWARF framework is that it focuses on software engineering concepts to organize the existing components. This approach leads to a high software complexity and may be a hindrance for applications that need specialized components that are not provided by the framework. The MORGAN framework [2], is also a component-based framework but it focus on multi-user applications. The system is composed of a set of components that subscribe to input devices such as trackers. A particular component called *broker* is responsible to allow the manipulation of all the framework's components. The framework also implements its own rendering engine, which may be an obstacle if a particular project demands specific rendering tools.

In this article, we present a distributed software architecture for AR applications. The architecture is based on a Client-Server style that focuses on simplicity and easy implementation and allows the user to naturally interact with the real environment. Different from the architectures used in the other frameworks [2][3], our architecture makes it easier to incorporate specialized components since it only needs a client-side and a server-side implementation. Furthermore, we do not restrict the use of specific technologies as is the case of the MORGAN framework [2]. The following sections of this article presents a brief description about the distributed architecture and, as a proof of concept, two applications were implemented using the presented architecture.

2 ARCHITECTURE DESCRIPTION

The architecture of the system is shown in Figure 1. The system is based on a Client-Server architecture with three main processes and a few central objects, as described in the two following sub-sections.

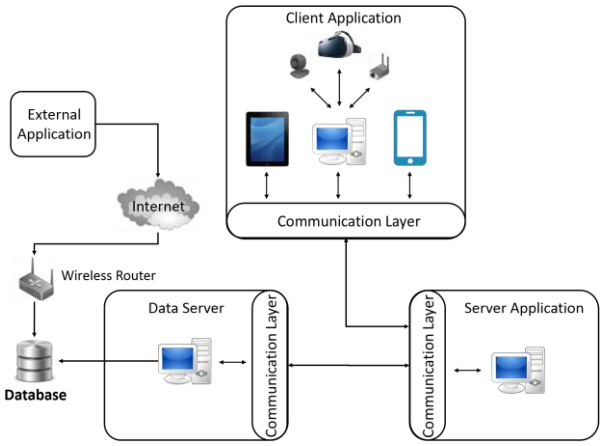

Figure 1: AR System Architecture

2.1 Objects

The objects are the smallest units in the architecture and they are responsible for specific tasks, such as tracking, interaction, communication and the manipulation of real objects.

Tracking objects are responsible for obtaining both the position and orientation of a given device or object of interest. *Interaction* objects specify the types of interaction the user can perform inside the system. The interaction may be to point to a real object, to select a real object on the mobile screen or performing a sequence of tasks, for example. *Communication* objects are the most important objects in the architecture since they are responsible to perform information exchange between the existing processes. For

*gregory_feijo@hotmail.com

every kind of information to be shared among application processes, it should exist both client and server objects. The server objects are responsible for packing the data and sending it through the network while the client objects waits for incoming data from the server object and makes it visible to the endpoint process. *Data* objects specify how the real objects are recognized inside the system. They mainly serve as containers for real objects information.

2.2 Processes

Three main processes constitute the architecture: Server Application, Client Application and Data Server. Others may be created according to the target application needs.

The **Server Application** is the core of the AR system and runs in a remote computer with high computational resources. This process should establish a connection with a Data Server in order to maintain all objects information available to clients and up-to-date. The Server should also manage multiple user connections by creating the necessary *communication* objects on the fly (tracking clients, interaction clients and object server). It is also responsible to perform the client desired interaction between users and the existing real objects, leaving the application running in the users' devices to be as light as possible. Whenever a user is interacting with a real object, the server can forward the object's information to the Client Application for further processing.

The **Client Application** is the application running in the users' device. At startup, the client establishes a connection with the Server Application and creates the necessary *communication* objects in order to exchange information. The Client Application is responsible to specify the type of interaction it desires and perform the data presentation to the user, whenever a new object is received from the server.

The **Data Server** works as a bridge between the AR system and any other system that manages the real world objects. The Data Server continuously checks the database for updates and forwards any updated information to the Server Application. The information can be anything meaningful for a given application, such as the object's description, images, multimedia, and so on.

3 Demonstration

As a proof of concept, we developed three applications based on our architecture. For the demonstrations, the *tracker* object is implemented based on the ARToolkit library (http://artoolkit.org), but other technologies could also be used. The *communication* object is performed using the VRPN library (http://vrpn.org). Both *data* and *interactions* objects are implemented differently depending on the application.

The first is a maintenance task application applied to a printer (Figure 2a). In this demo, the *data* is a container for an image that contains textual description about the task to be performed. The *interaction* specifies the marker visible to the user so the server is aware of the next step to forward to the client. The user wears a HMD with a webcam attached and both connected to a desktop machine. Whenever the user sees a marker, the *tracker* retrieves its position and it is sent with the visible marker to the Server Application by a *communication* object and remotely processed. The server forwards back the image for the next step of the task.

In the second application (Figure 2b), the *data* is a container for boxes spread around a pre-defined scenario and the *interaction* is performed by pointing a webcam in the direction of the real object the user is interested in. The user also wears a HMD and both its position and the pointer's are tracked by *tracker* objects and forwarded to the Server Application whenever available. The server then, applies a ray casting algorithm in order to find the object and forwards its information to the client. In this application, the objects are in fixed positions.

The third application (Figure 3) is quiet similar to the second, but the user interact with the system using a mobile device for both pointing and displaying the real objects information. Furthermore, a *tracking* object is constantly tracking the real objects position and feeding the database with updated information. The Data Server is clearly more active in this application since it propagates the updates to the Server Application.

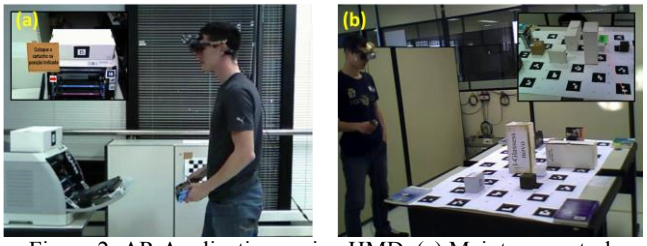

Figure 2: AR Applications using HMD. (a) Maintenance task applied to a printer. (b) Real object pointer application.


Figure 3: Real object pointer application using a mobile device.

4 Conclusion

We presented a simple yet effective software architecture for distributed AR systems. The architecture is composed of three main processes that use a few central objects. Each object individually contributes for a small portion of the AR system and, together, they successfully deliver the user a realistic Augmented Reality experience. As a proof of concept, three different applications were developed using the central objects presented, allowing a user to interact with the system in different ways.

5 Acknowledgments

Our research is funded by FINEP NAVITEC Project. Grant 01.13.0360.00, Ref.: 1233/13

References

[1] Danado, J., Dias E., Romão, T., Correia, N., Trabuco A., Santos C., Araújo D., Duarte, P., Rebocho, R., Palmeiro, J., Serpa, J., Costa M. and Câmara, A.,"Mobile Augmented Reality for Environmental Management(MARE)". In *EUROGRAPHICS 2003*, 1-6, pp.121-128.

[2] Bauer, M.; Bruegge, B.; Klinker, Gudrun; MacWilliams, A.; Reicher, T.; Riss, S.; Sandor, Christian; Wagner, M., "Design of a component-based augmented reality framework,". *IEEE and ACM International Symposium on Aumented Reality* , pp. 45-54, 2001.

[3] Ohlenburg, J., Herbst, I., Lindt, I., Fröhlich, T. and Broll, W. The MORGAN framework: enabling dynamic multi-user AR and VR projects. ACM VRST 2001, pp. 166-169.