Abstract—Immersive environments integrate real and virtual elements enabling a high interaction between them. To make this task possible, it is usually necessary to perform the reconstruction of real objects with images captured from multiple cameras, using 3D modeling and texturing techniques. Thus, the acquisition of images is one of the pillars of immersive environments. This work aims to study the integration of hardware and software to compose an infrastructure capable of capturing sequences of images from multiple cameras in real-time and synchronously. The methodology for the development of the system consists primarily in a review of the related literature, identifying the key components and technologies involved in this kind of system. Then, a detailed study of the identified fundamental components was realized. Finally, a prototype of the system was proposed. It was design and development as a hardware and software infrastructure, suitable to be used in future works as the base for immersive environments or other complex systems that also require multiple camera acquisitions.

Keywords—image acquisition; multiple cameras; synchronization; acquisition frameworks.

I. INTRODUCTION

Immersive environments that integrate motion capture and mixed reality allow a great level of interaction between real users and virtual applications. Such environments can create a virtual world from the synthesis of virtual objects. This can be based on the reconstruction of the real objects from their images captured by multiple cameras and processed through 3D modeling and texturization techniques.

The virtual reality allowed by such environments enable the creation of many complex applications into fields like, telepresence, games, physics phenomenon modeling, teaching tools, medical and others. The creation of a virtual environment with 3D modeling is only possible due to a base component used in such systems, the image acquisition infrastructure. It is the part of the system responsible for the acquisition of real world objects images, allowing their modeling, texturing and reconstruction.

To achieve the 3D reconstruction of a determined scene, using multiple cameras, it is necessary the development of a robust image acquisition infrastructure, concerning mainly the cameras synchronization and calibration. The cameras are employed, disposed at different viewpoints forming a three-dimensional acquisition space used for capturing images of real objects.

The image acquisition through multiple cameras do not only brings contributions just to immersive environments applications. Such infrastructure can also be applied in object recognition and tracking, borders surveillance, target tracking for intrusion detection systems, etc. Thus, the wide range of applications for acquisition infrastructures, in various segments of industry and academia, is one of the main motivations for the study and development of these systems.

In this context, the main objective of this work is to propose a prototype of an image acquisition infrastructure using multiple cameras, describing and comparing related works and major technologies involved in such systems. Thus, present paper is structured as follows. In Section II relevant related works found in the literature are presented. A detailed review of multi-camera acquisition systems fundamental parts is presented in Section III. The developed prototype, the performed experiments and the obtained results are detailed in Section IV. Finally, Section V brings the conclusions and future works.

II. RELATED WORK

There are several works in the literature which treat the acquisition of images with multiple cameras. Based on this, a study was conducted on the more significant works on the area.

One of the seminal works on multi-camera acquisition was the development of a stereo vision system in early 1990, according Kanade and Narayanan [1]. Since that, several projects in the area began to emerge. In 1993, Kanade built a number of machines that converted from a scene input to a map of 256x256 pixels (8 bits deep) at a rate of 30 frames per second [2]. Based on these machines, in mid-1994, Kanade built a system with 10 cameras, which allowed the observation of dynamic scenes from multiple points of view. This system was expanded in 1995 to a dome with 51 cameras. The system was analog and offline, the images were synchronized (adding up time references) and stored on tape for further processing. Since 1998 the system has become completely digital, using only digital cameras and online acquisition.

Moezzi et al. [3] designed an interactive video system with multiple perspectives using a combination of static models, detection of changes and shape triangulation to model and
navigate through large spaces. Later, techniques for image-based rendering (IBR) for capturing objects from multiple points of view have become a topic of intense study, initially for individual static objects and then to dynamic scenes. 3D modeling projects applied various modeling techniques for reconstruction of buildings, sites and mines [4]. The Digital Michelangelo Project [5] and the Great Buddha [6] used high-quality rangefinders (precision devices intended for measuring distances in real time) for archiving and preservation of cultural heritage.

A little later, other systems have emerged and boosted the capture of dynamic events, such as the Blue-c [7], the 3D video recorder of ETH Swiss Federal Institute of Technology [8], the Free-Viewpoint Video of Max Planck Institute for Computer Science [9], the rendering system based on video from Microsoft Research [10], the platform GrImage [11], the motion capture software Vicon Nexus [12], the system TEVEE [13], and finally, the commercial multi-capture software from NorPix the StreamPix [14]. Another system was proposed from University of Nagoya, which capture dynamic events and has an array with thousands of cameras [15]. This system is used for TV systems with free point of view. The main interfaces found in the market and mentioned in the reviewed literature as appropriate to be employed in a multi-camera image acquisition system, as well as a comparison between them, are presented next. 

a) IEEE 1394 (Firewire): IEEE 1394 [16] is the designation for a standard serial bus for high performance. The design of this bus was originated at Apple Computer Inc., as a diagnostic tool. The bus architecture was registered by Apple under the name "FireWire". In 1995, the IEEE standardized the bus specification under the number 1394 - hence the name IEEE 1394. Another trade name for this bus is i.LINK, used since 1997 by Sony in its PlayStation 2 [17]. Along the time, variations of the pattern 1394 appeared, adding new and more modern features, such as the patterns 1394a [18] and 1394b [19].

b) Camera Link: Camera Link [20] is a standard serial communication protocol designed for computer vision applications based on the interface Channel Link [21], from National Semiconductor [22]. The Camera Link was designed with the purpose of standardizing video products, scientific and industrial applications, including cameras, cables and capture cards. The standard is maintained and administered by Automated Imaging Association (AIA) [23], the global industry trade group for vision systems. Camera Link is a trademark of AIA.

Camera Link is the best option in terms of performance for connecting a camera into a capture card, achieving a throughput up to 850 MB/s. The main disadvantage of this technology is its cost, which is much higher than a FireWire system, for example.

c) GigE Vision: The GigE Vision [24] is an interface standard that provides an open framework, high performance and scalable for streaming images and control devices over Ethernet networks. It provides an environment for networked vision systems based on interchangeable client-server architectures, allowing the connection of multiple cameras to multiple computers.

The GigE Vision standard is property of AIA [23] and was developed by a group of enterprises belonging to all sectors of vision systems industry. The aim was to establish a standard that allows camera manufacturers and software developers to integrate their products through the bus Gigabit Ethernet. It is the first standard that allows images being transferred at high speeds over long cables. The GigE Vision standard defines both, the behavior of the host and of the camera [25].

d) USB3 Vision: USB3 Vision [26] is an interface standard for cameras, built over the USB 3.0 specification [27], and based on existing standards such as GigE Vision and standards for Gigabit Ethernet devices. The USB 3.0 standard, also known as SuperSpeed USB, is the second major revision of the USB standard made until nowadays. The USB 3.0 is based on USB 2.0 [28], achieving a maximum throughput considerably larger than its predecessor and providing 2.0 to 4.5 watts of power for peripheral devices [29].

At the current rate of implementation, all new produced computers will support the USB 3.0 standard [30] until 2015. Considering this, vendors can build applications and deploy them on almost any computer without the need for additional hardware. Cameras, for example, could be connected directly into any standard USB port without the need of a specific capture card, which reduces the cost the system.

e) Comparison Between the Interfaces: One of the most important steps in a project of a multiple cameras acquisition system is the correct choice of the best interface for the connection between the camera and the computer. There is no interface technology on the market that is the most indicated for all types of system. There are different buses technologies, with advantages and disadvantages, depending on the application that are being considered. Therefore, to determine the most suitable interface for each system it is necessary to take into account mainly the characteristics of application scenario.

The analysis of the above mentioned interfaces is summarized it Table IV. Such table compares the interfaces through their main characteristics concerning several relevant aspects, what is a contribution of the present work.
B. Synchronization

Multiple cameras acquisition systems continuously capture several frames from different viewpoints. These images need to be captured synchronously and receive a time reference (timestamp) for chronological identification.

Applications that perform the 3D reconstruction of the scene require a high level of synchronization to enable the correct reconstruction of such scenes. Images from multiple cameras need to be captured at exactly the same moment, especially if the scene is not static.

It is also necessary to calculate the acceptable time interval between the images captured. In some cases the acquisition process must be controlled externally to ensure this interval.

In this context, the timing mechanisms are essential in multicamera systems, because they are responsible for controlling the timing and time referencing on the captured frames. In the literature, several synchronization strategies have been found. The four most important are listed below:

a) Hardware Trigger: it is based on the generation of a dedicated electrical signal (TTL signal) from an external hardware (trigger) module.

b) Firewire Bus: some Point Grey [31] cameras are capable of perform automatic synchronization when connected to the same Firewire bus.

c) Computer Trigger: it is also based on electrical signal (TTL signal), but the signal is generated and controlled by software on a computer.

d) Software: the acquisition software controls the timing of capture, sending a specific software command simultaneously for each camera.

e) Comparison Between the Synchronization Methods: Table I summarizes the analyzed synchronization strategies concerning several important aspects. Such table is a contribution of the present work.

C. Acquisition Software

One of the key components of a system for image acquisition is the software that capture and index the frames. Developing this software from scratch would be an arduous task, which would require much time and engineering effort. Thus, the great majority of systems found in the literature are developed based on some pre-existing APIs. In this context, this section discusses some major frameworks currently available for camera control, image capture and time indexing.

a) OpenCV: OpenCV [35] is a computer vision library widely used in the acquisition, processing and analysis of images. Adopted worldwide, it has a user community of over 47 thousand people. It comprises more than 2,500 computer vision algorithms, has extensive documentation and a large number of code samples. It was designed to have computational efficiency and a strong focus on real-time applications.

b) DirectShow: DirectShow [36] is a multimedia framework produced by Microsoft that allows the execution of various operations with media files or streams. It is an extensible framework based on filters (filter chain). It is able to capture, process and render images and sounds. It splits a complex multimedia task in a sequence of processing fundamental steps, known as filters.

c) Linux Media Infrastructure API: Linux Media Infrastructure API [37] is a common framework for multimedia, used in kernel level (drivers) or user space (applications). It is used in digital video broadcasts, as well as to capture and output videos.

d) Harpia: Harpia [38] is a framework for processing signals (images), and remote vision systems management. It has basic features common for image capturing, processing and rendering. It also has a simple intuitive interface. The programming is graphical oriented using block diagrams and process flows.

e) AVT Fire4Linux: AVT Fire4Linux [39] is a comprehensive framework optimized for IEEE 1394 cameras from Allied Vision Technologies (AVT). Based on generic drivers for Linux and standard libraries for IEEE 1394 cameras. It has specific methods capable of controlling all the major features on AVT IEEE 1394 cameras, providing an API to control and capture images in conformity with the standard IIDC (DCAM) [40].

f) Comparison Between the Acquisition Softwares: Table II summarizes the analyzed acquisition softwares frameworks, concerning several important aspects. Such table is a contribution of the present work.

IV. EXPERIMENTS AND RESULTS

As the main result of the present work, a prototype of the multi-camera system proposed infrastructure was developed. Fig. 1 and Fig. 2 show a partial view of the system. The complete prototype is composed by: (1) a computer, (2) acquisition software, (3) four AVT Guppy F-036 cameras

---

**TABLE I** COMPARISON BETWEEN INTERFACES

<table>
<thead>
<tr>
<th>Features/Interfaces</th>
<th>IEEE 1394b</th>
<th>Gigabit Ethernet</th>
<th>USB 3.0</th>
<th>Camera Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard</td>
<td>H.264, DCAM</td>
<td>Gigabit Ethernet</td>
<td>USB 3.0</td>
<td>Camera Link</td>
</tr>
<tr>
<td>Speed (Gbps)</td>
<td>1.25 GB/s</td>
<td>10 Gbps</td>
<td>5 Gbps</td>
<td>2 Gbps</td>
</tr>
<tr>
<td>CPU Usage</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>None</td>
</tr>
<tr>
<td>Cost</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>None</td>
</tr>
<tr>
<td>Interfaces difficulty</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>None</td>
</tr>
<tr>
<td>Supply (V)</td>
<td>5 V</td>
<td>12 V</td>
<td>5 V</td>
<td>None</td>
</tr>
<tr>
<td>Cable length (m)</td>
<td>50</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Monitors Cameras</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Regular</td>
</tr>
<tr>
<td>Maximum cameras</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Regular</td>
</tr>
<tr>
<td>Plug and Play</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Interface Card</td>
<td>High 1394b</td>
<td>High 1394b</td>
<td>High 1394b</td>
<td>Medium</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Acceptance</td>
<td>Increasing</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
</tbody>
</table>

**TABLE II** COMPARISON BETWEEN SYNCHRONIZATION METHODS

<table>
<thead>
<tr>
<th>Metric/Method</th>
<th>Hardware Trigger</th>
<th>Firewire Bus</th>
<th>Computer trigger</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>99.9%</td>
<td>99.9%</td>
<td>99%</td>
<td>OS</td>
</tr>
<tr>
<td>Imposition</td>
<td>High</td>
<td>None</td>
<td>High</td>
<td>None</td>
</tr>
<tr>
<td>Cost of implementation</td>
<td>High</td>
<td>None</td>
<td>High</td>
<td>None</td>
</tr>
<tr>
<td>Difficulty</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Interface Card</td>
<td>Any</td>
<td>Camera Link</td>
<td>Any</td>
<td>Any</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Hardware trigger</td>
<td>All</td>
<td>Camera Link</td>
<td>Any</td>
<td>None</td>
</tr>
<tr>
<td>Software trigger</td>
<td>Firewire</td>
<td>Firewire</td>
<td>Any</td>
<td>None</td>
</tr>
<tr>
<td>Input trigger</td>
<td>Free</td>
<td>Camera Link</td>
<td>Any</td>
<td>None</td>
</tr>
<tr>
<td>Accuracy</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>Data transfer rate</td>
<td>80 MB/s</td>
<td>680 MB/s</td>
<td>440 MB/s</td>
<td>Decreasing</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>Good</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
</tr>
<tr>
<td>Network</td>
<td>USB3 Vision</td>
<td>IEEE1394</td>
<td>All</td>
<td>Any</td>
</tr>
<tr>
<td>Speed</td>
<td>Hardware</td>
<td>Computer</td>
<td>Software</td>
<td></td>
</tr>
</tbody>
</table>

---
TABLE III
COMPARISON BETWEEN ACQUISITION SOFTWARES

<table>
<thead>
<tr>
<th>Metrics/API</th>
<th>OpenCV</th>
<th>Direct Show</th>
<th>Linux Media API</th>
<th>HARPIA</th>
<th>AVT Fire4Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS</td>
<td>Linux</td>
<td>Windows</td>
<td>Linux and Windows</td>
<td>Linux and Mac OS X</td>
<td></td>
</tr>
<tr>
<td>Languages</td>
<td>C/C++, Python, Java, MATLAB, C#</td>
<td>Supports the COM framework</td>
<td>C/C++</td>
<td>Blocks Diagram</td>
<td>C/C++</td>
</tr>
<tr>
<td>Licensing</td>
<td>Open Source (BSD)</td>
<td>Not needed</td>
<td>Open Source</td>
<td>Open Source</td>
<td>Open Source (LGPL)</td>
</tr>
<tr>
<td>Buses and cameras</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>IEEE1394 and A VT cams</td>
</tr>
<tr>
<td>Standard vision system</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>IEEE1394 (DCAM)</td>
</tr>
</tbody>
</table>

connected to a IEEE 1394a bus, (4) two AVT Stingray F-046 cameras connected to a IEEE 1394b bus, (5) camera brackets, and (6) a monochromatic background.

The cameras were set around an area of approximately one cubic meter (1 m³). The floor and the wall of the environment were covered with a standard color in order to facilitate the posterior background subtraction of captured images.

![Fig. 1. Two partial views of the system prototype - cameras, brackets, monochromatic background and some toy objects.](image1)

Fig. 1. Two partial views of the system prototype - cameras, brackets, monochromatic background and some toy objects.

Several tests of the system were realized. Fig. 3 shows, as an example, the frames captured by the six cameras in one time instant. In the acquisition volume there were some static and moving toys. In the Sections below the main results concerning the acquisition software implementation and the synchronization are detailed.

A. Acquisition Software Implementation

The acquisition software was implemented based on AVT Fire4Linux library. Sample codes of the library package were used as a starting point for the implementation of the software, as they show how to use the main functions of AVT Fire4Linux. Additionally, this package has a complete documentation of all framework functions.

The acquisition software was developed as a C++ class, called CMultiAcquisition, which allows the configuration of multiple cameras and automatically makes the management of multiple acquisitions and the disk storage of the captured images.

For a better understanding of the class as a whole, Fig. 4 presents its UML class diagram. Through the diagram it is possible check out the interface (public methods) of the class used to: (1) setup the cameras, (2) setting the saved images to disk, and (3) set operating modes.

B. Synchronization Strategy

The software synchronization approach was adopted in this work. In this strategy, a capture command, which behaves as a trigger signal, is sent simultaneously to all the cameras, enabling them to capture images synchronously.

To check the time difference between the frames of the different cameras it was used a testing methodology based on a stopwatch. In this methodology, all cameras must view the same timer, so the timer value shown in the images indicates the exact instant when the images were captured.

Four cameras were used for testing. They were connected to two different buses: (1) two cameras AVT Guppy F-036, connected to an IEEE 1394a bus, and (2) two cameras AVT Stingray F-046 connected to an IEEE 1394b bus.

Analyzing the results of the tests, it was noticed that the two Stingray cameras were always synchronized in the order of milliseconds, regardless the use of synchronization via software.

The Guppy cameras presented a timing accuracy of about 10 milliseconds between each other, when no synchronization...
was employed. Using the synchronization via software, the accuracy between the two Guppy cameras strangely falls to an average of 30 milliseconds. Which leads to the conclusion that the software synchronization strategy was not effective or that it is less efficient than the "natural" synchronization of the bus.

But when the time differences between the cameras into distinct buses are compared, one can realize the importance of the applied synchronization. Analyzing sets of frames captured without synchronization, it is noticed that the frames from 1394a bus are completely disparate regarding the ones from the 1394b bus. The time difference between captures increases along the time. This indicates that the cameras into 1394b bus are capturing in a greater rate greater than that ones into bus 1394a. Without any synchronization, there is no control over the time of acquisition.

Despite the precision between captures of the four tested cameras reach values around 100 milliseconds - which seems to be an extremely high value - all cameras are being controlled by software and thus all them have the same frame rate. Never a frame "t + 1" of a camera will be captured before a frame "t" from another camera. This problem was happening when no synchronization was being applied.

The downside of synchronization by software is that it adds a significant delay in the capture, as the cameras need to wait for the command from the software at each cycle, decreasing the maximum frame rate achieved. In the tests, the average frame rate was reduced from 30 to 4 frames per second. Furthermore, all cameras will have their rates limited by the camera with lower rate.

V. CONCLUSION AND FUTURE WORKS

The result of this work was a detailed study of the main technologies involved in multi-camera acquisition systems, as well as the proposal and implementation of a hardware and software infrastructure prototype for image acquisition. It can be the basis for many vision systems such as immersive environments, pedagogical applications, among others. As another important contribution, the main characteristics of interfaces, cameras synchronization and acquisition softwares approaches, concerning relevant aspects, were analyzed and systematic summarized. Such information can guide the choice of the best approaches for different applications scenarios.

Based on experiments, it is concluded that the constructed system is capable of capturing synchronized images from multiple cameras, even when different cameras models are used. However, the timing reached is not yet ideal for more complex vision systems, since its accuracy is on the order of milliseconds. For a more precise synchronization an external triggering hardware is required. Software synchronization is also time-consuming, reducing greatly the maximum frame rate achieved by the acquisition software.

Many other aspects and technologies, besides interfaces, synchronization and acquisition softwares, are involved in the built of multi-camera systems. So, clearly, a more comprehensive study, covering other components such as cameras models and calibration, would be a possible future work. Another point to be explored would be the possibility of the acquisition software. A graphical user interface (GUI) could be added, letting it more friendly, intuitive and reducing the workload of the user. Finally, a future improvement that would add a great value to the architecture, would be the use of an external hardware trigger approach. This would allow the use of the system for more complex vision systems that require more rigorous levels of synchronization.

REFERENCES