INSPECTION OF ELECTRICAL EQUIPMENT ASSISTED BY AUGMENTED AND DIMINISHED REALITY

BY

CRYSTIAN WENDEL MENESES LEÃO

MASTER THESIS

Recife

20/02/2012

Crystian Wendel Meneses Leão

Inspection of electrical equipment assisted by augmented and diminished reality

Thesis presented as a partial requirement to obtain the Master title, by the Graduate Program in 2012 from the Centro de Informática from the Universidade Federal de Pernambuco.

Advisor: Judith Kelner, PhD

Co-Advisor: João Marcelo Xavier Natário Texeira, MSc

Recife

20/02/2012

I dedicate this thesis to everyone that somehow contributed to it, especially to my parents, my close friends, my advisor and co-advisor.

ACKNOWLEGMENTS

I would like to formally say thank you to all that somehow contributed to this master thesis and all the education I had before it. I am sure that these two years of dedication will worth each second in the rest of my life. I would like to say thank to my advisor, and to my co-advisor for the huge help with this. I would like to say sorry to them too, for all my faults during these two years inside the Lab. I should not forget to thank for all the work, fun and friendship inside GRVM. I would like to say thank you to the GRVM itself, all of the researches held inside it contributed somehow to build the knowledge I have now. I would like to say thank you to my parents, my family and friends for staying at my side (or far from it, because of this work).

I would like to thank the CNPq, the agency that under the process number 133688/2010-0 has supported me through these two years, and GRVM that supported me as well. I would like to thank FADE, UFPE, CIn, and all the teachers I had the pleasure to attend their classes.

At last, I send a special thank you to the design team at GRVM, for the great pictures inside this work and the chapter 5!

ABSTRACT

This research proposes a system to aid maintenance, inspection and training procedures involving electrical equipment. The aid happens with the addition of visual highlights, audio, video and image files, and through the removal of unwanted parts of the target equipment that can divert the operator's focus from the desired place in a specific moment. This system will run on a wearable computer that will show the information to the operator through a display attached to his/her head. This thesis covers the proposal of a tool to build new procedures too, which uses just pictures of the target equipment, and a wearable platform to be used in the field.

RESUMO

Esta pesquisa apresenta um sistema para auxílio na manutenção, inspeção e treinamento de equipamentos elétricos. O auxílio acontece através da adição de indicadores visuais registrados, arquivos de áudio, vídeo e imagem, e através da remoção de partes indesejadas do equipamento que possam tirar o foco principal do operador em um determinado momento. Este sistema deverá executar em um computador vestível, e as informações serão exibidas com o auxilio de um display acoplado na cabeça do operador. A proposta de arquitetura também engloba o software que constrói novos roteiros de manutenção para novos equipamentos, utilizando apenas imagens do mesmo, e uma plataforma vestível, para ser usada em campo.

IMAGE LIST

TABLE LIST

ACRONYM LIST

AR – Augmented Reality
CHESF – Companhia Hidro-Elétrica do São Francisco
CPU – Central Processing Unit
DR – Diminished Reality
GPGPU – General-Purpose Computing on Graphics Processing Unit
GPS – Global Positioning System
GPU – Graphics Processing Unit
HMD – Head-Mounted Display
MAR – Markerless Augmented Reality
OST – Optical See-Through
RAM – Random Access Memory
SfM – Structure from Motion
VR – Virtual Reality
VST – Video See-Through
W3C – World Wide Web Consortium
XML – Extensible Markup Language

SUMARY

1 INTRODUCTION

This section introduces this master thesis. It will briefly describe the motivation of the work itself, state the problem that we will focus on solving and shortly discuss the proposed solution. In the end, it will explain the structure of the next chapters.

1.1 MOTIVATION

Maintenance, inspection and training activities are essential inside an industrial context. While inspection activities are of great importance to prevent any equipment from failing during activity, maintenance is crucial to repair them when those fails occur and training must be done in order to create skilled labor to perform those activities.

When those activities are not well performed, many problems can occur. Errors during inspection can underestimate impending problems, which can lead to broken equipment. Errors during maintenance can cause harm to those equipment, and a bad training can create unskilled labor, that leads to the two previous problems.

Because of that, there is a critical need for approaches focusing on the reduction of the error rate involving inspection and maintenance procedures. The solution adopted, as consequence, may also improve the quality of training regarding the engineers, technicians and other possible target users.

When it comes to electrical equipment, there are many recurrent problems that make maintenance and inspection activities to be performed periodically, like failing connections, rusting, overheating, misuse or abrasion. Those events require maintenance and repair efforts from trained professionals, in a continuous way. Thus, technology could help to reduce costs in those scenarios in different ways, such as reducing error rate and preventing the equipment from suffering from these events, enabling less trained professionals to perform the activity, and helping the training process itself, reducing costs related to formal teachers.

1.2 PROBLEM DEFINITION

Many efforts have been made in order to reduce errors during the execution of maintenance procedures with different types of technologies. Nowadays, with all portable multimedia devices becoming more accessible, maintenance procedure documentation can be not only on books and manuals, but also in cellphones or tablets, allowing the inclusion of multimedia content, such as video and audio explanations.

There are many works in the literature using advanced technologies that help the user by displaying content directly over the target of the procedure, indicating exactly where user should guide his/her attention, and what user should do in a given moment. These complex systems exploit computer vision techniques to keep track of the target of the maintenance and use some display attached to the head of the worker to show the virtual information over it.

Nevertheless, those systems found in the literature need sophisticate three-dimensional data about the equipment in order to track them. In addition, none of them aids the user by removing unwanted parts of the scene neither by focusing on the development of a full platform, including the wearable computer, a system that aids the maintenance and a system that builds the aided maintenance procedures.

1.3 OBJECTIVES

The main objective of this work is to propose a general architecture to a system that aid maintenance procedures by adding visual cues and removing unwanted information over the target equipment, in order to guide the user performing the maintenance, avoiding attention deviation and reducing errors. One particularity of this approach that is worth to be mentioned here is that it will just use pictures of the target equipment in order to build the maintenance procedures, instead of three-dimensional models commonly used in the literature. This way, the obtained result is similar without all the effort necessary to create the complex 3D models used for performing the tracking.

In a second stage, the proposed architecture will be developed in a partnership between an electrical supply company and our research Lab, GRVM. Inside the scope of this master thesis, three components of the whole project were implemented: the wearable platform, which is an extension of a known platform, the software that aids the maintenances and the one that creates and edits them. This work focuses on the third one.

At last, a group of specialist designers will evaluate the system that builds the procedures by testing a simple scenario of utilization following a known evaluation technique, pointing out all the found issues and ordering them accordingly to a famous prioritizing schema for software tools. As the previous step will generate an ordered list of issues, where we can identify which one has more importance, we will use this list as a basis for future work on this component.

1.4 STRUCTURE OF THE DISSERTATION

This chapter has shortly described what will be discussed in this thesis, which problem will be solved and the chosen approach to solve it. Chapter 2 will go deeper into the main technical subjects of this thesis: Augmented Reality, Diminished Reality and General-purpose Programming on Graphics Processing Units. Chapter 3 will explain why one should spend efforts on assisting maintenance, repair and training procedures with Augmented and Diminished Reality techniques and will discuss about the research state of the art involving these themes. At the end, a generic architecture for a solution to assist maintenance will be presented.

In chapter 4, a case study with a Brazilian electrical supply company called CHESF will be detailed, including the chosen technologies, hardware components, and the most important parts of the solution. In chapter five, the solution built for CHESF will be evaluated considering usability aspects, and some general conclusions will be presented. The last chapter present overall conclusions, discuss briefly about the initial strands of this work, the published works related to it and future work.

2 BASIC CONCEPTS

This section introduces some basic concepts related to this work. They are presented as individual sections, describing about the state of the art techniques and applications involving the main technical subjects of this work: augmented reality, inpaint, diminished reality and general-purpose computing on graphics processing units.

As the ambition of this thesis is to propose and to analyze a solution that goes to aid mechanicals in maintenance, inspection and training activities exploiting augmented and diminished reality technologies. This chapter introduces augmented reality, inpaint, as a basis for diminished reality, diminished reality itself and general-purpose computing on graphics processing units, as the use of this technology was mandatory in order to improve the overall performance of the work in the desired platform.

2.1 AUGMENTED REALITY

Mixed reality applications are those where virtual and real elements coexist. Depending on the proportion in which real and virtual elements are arranged, the applications can be classified as Virtual Reality (VR) or Augmented Reality (AR) applications. While in VR applications the world is generated by a computer and most of the objects are virtual, in AR ones virtual elements are inserted into the real world in a way that they seem to be part of it [1].

2.1.1.CONTEXTUALIZATION

In order to visualize the virtual information inserted in real time in a real environment, there is a considerable amount of options available, as head mounted displays, monitors, projectors and mobile phones. Figure 1 shows the main technologies used to display data in AR applications.

Figure 1 - Main technologies used as displays in VR applications.

To insert properly some virtual elements in real environments, the application must correctly position the object relatively to the real world, namely registration. In order to do this, certain types of sensors can be used to get some information about the environment, such as GPSs, movement sensors, thermal imagers, ultrasound, cameras, and others. Those applications use these sensors as a way of gathering useful information about the real world in order to know where and how to place the virtual objects inside the scene.

AR applications that use cameras as the main source of information, specifically the ones that use just one image sensor, make use of monocular computer vision techniques. Generally, such techniques are important for the scientific community because the hardware required is a low cost one.

Independent of the kind of sensor utilized as the source of environment data, AR can be divided into two main groups: marker and markerless techniques. Marker techniques can be distinguished because they add some synthetic object into the environment in order to make the registration process simpler. A camera-based AR system that uses an artificial colored object, some type of structured light or a thermal imaging based system that needs some artificial heat source can be considered marker-based systems.

In the other hand, markerless systems use only the “natural” features present in the environment in order to make the registration process feasible. A markerless system exploits computer vision techniques to search for natural features in the world, use GPS, compass and some movement sensor to know exactly where the user is looking at. They can even perform a real time 3D

reconstruction with a depth sensor, with multiple cameras or with a single one, and use the 3D information to insert the virtual objects properly.

MAR systems that make use of just one camera to perform registration are called monocular markerless augmented reality systems, and this class of systems can be classified in two major categories based on their techniques: model-based and structure from motion based. Model-based techniques have previous information about the object to be tracked, and the main restriction about those techniques is that this object must be visible in the scene in order to the registration process to be successful. In the other hand, structure from motion techniques use temporal information to estimate the camera position and motion in real time, and sometimes they go even further: it is possible to build a world 3D model in real time.

Among the existing model-based techniques, three major groups are distinguished: edge-based, texture-based and optical flow based. One advantage of model-based techniques is that the system can use the information about the object being tracked to make interactions between it and virtual objects.

In the edge-based group, a 3D model of the object to be tracked must be given to the system. The tracking process usually starts with an initialization step, in which the initial position of the object is known. In the subsequent frames the object is tracked based on its previous position, using information about the 3D model's edges and detected edges on the image. Tracking the object using its 3D model allows the application to handle occlusions, collisions and other physical based interactions. Algorithms used by edge-based MAR systems usually project the 3D model of the object at the previously estimated position in the next frame, and search for the object in the neighborhood of the projected 3D model.

Texture-based MAR systems are capable of tracking an object in a scene using just texture information. These systems need an initialization step, in which texture information about the object must be given to the system. Based on that information they will track the object in each subsequent frame.

Texture-based MAR techniques could be divided into two major groups: template matching and interest point based. While in template matching the system works by applying distortion models to a reference image and searching for the

completely distorted image in the video sequence, interest point based performs a training step where it chooses some keypoints in the reference image and search for these points in the video sequence. As it finds matching points, it calculates the perspective transformation needed to take the found points on the video frame to the original image.

Optical flow techniques differ from the others because they just need an initialization step, where the virtual object should be manually placed in the scene, and they are capable of keeping the track over the frames of the input video just using temporal information. The algorithm keeps the keypoints found in one frame and searches them in the next one. The next step is to re-estimate the pose of the tracked object based on the keypoints found in the new frame. These techniques usually accumulate jitter and illumination deviations and can cause major errors in the entire process.

The third and last group in model-based MAR systems is called Structure from Motion (SfM). Despite the other techniques that rely on previously obtained information about the environment, SfM is capable of estimating the camera displacement without any, a priori, knowledge about the environment, just using temporal information. Some SfM algorithms are capable of retrieving the 3D structure of the world in real time as well. SfM based methods just uses temporal information and does not need an offline initialization phase.

This classification of AR applications, as stated before, is illustrated in Figure 2, as found in [2].

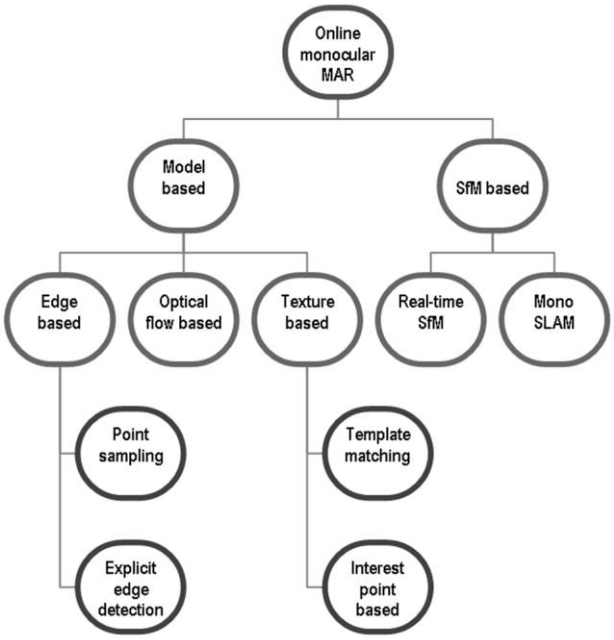

Figure 2 - The classification of MAR techniques as described in [2].

2.1.2.STATE OF THE ART

AR has been successfully used in many different areas such as advertisement [3], games [4], medicine [5] and maintenance [6][7][8][9][10][11][12][13][14]. A significant effort has been made in order to find out easier ways to develop AR applications with low cost hardware as well as without setup process, using, for example, SfM techniques.

In [15], a parallel version of a well-known SfM technique is used to make a dense live reconstruction of the environment that will serve as basis for an augmented reality application. The algorithm developed detects matches of keypoints in subsequent frames and uses this information to make the registration process and build a dense point cloud representing the world. The next step is to group these points into clusters and refine them in parallel, so that the algorithm can keep collecting new points of the environment. When the algorithm finishes the processing step of a cluster of points there will be a 3D model of a piece of the world, and with that in hand, the augmented objects can be placed in the scene.

Another work with an interesting MAR application is the one described in [16], the authors make use of a Kinect sensor to perform a live dense

reconstruction of the environment and augment the world using the data built. The Kinect sensor has a camera, an infrared emitter and an infrared camera (both camera resolutions as well as other technical details about the sensor can be found in [17]). While the infrared emitter and camera is used to get the depth information about the world, the algorithm of [16] builds a 3D model with this data, refine and increase the model at each new frame (as new portions of the world become visible). The 3D model is used to insert virtual objects, and make some physical interaction between them. Figure 3 shows the work of [15] (top images) and [16] (bottom images).

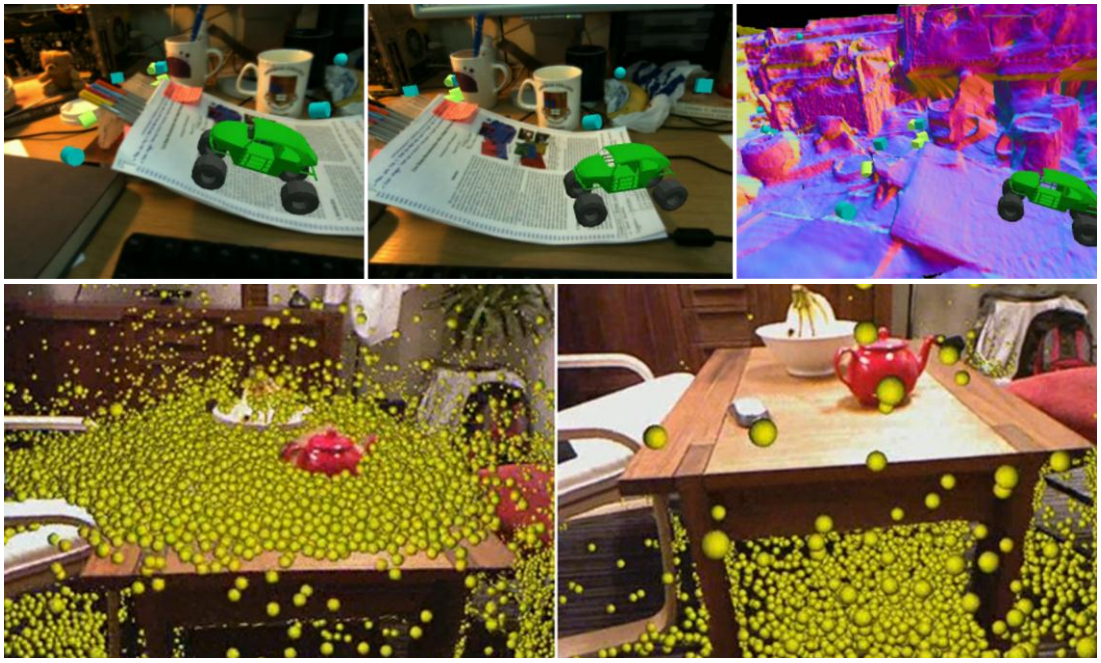

Figure 3 - MAR techniques of [15] and [16].

In [18], Lepetit et al. develop a model-based MAR technique, called Ferns, that uses just texture information to track an object. This algorithm searches the image for keypoints and classifies them according to previously obtained information about keypoints present in the target object. After the classification step, the system chooses the best classified points and make the registration process with that information. This technique will be depicted in section 4 .

An example of a medical application of AR can be found in [5]. In this work, the authors build a system to aid fixation of reconstruction plates (deformable metal plates) in bone fracture sites with irregular anatomical morphology. The system

uses 3D models of intact contralateral bones to generate 3D models of the implants and render them in an AR environment to guide preoperative implant.

Advertisements with augmented reality are not a new idea. Tarumi, H. et al. [19] develops an idea of advertisement and general information in the air. The Space Tags, the name of the virtual information in this system, can be placed at any geographical location in the world by anyone, just like a copy and paste system, and the information would be in that location for a limited period of time. A recent effort on AR in the area of advertisement is the work in [3], in which the BMW company lets the user to interact with a 3D model of a BMW Z4 by printing a marker and downloading a software tool that allows the user to drive the car with the keyboard.

AR has been successfully used in games too, as described in [4]. This game, called Eye of Judgment™, consists in a card game where the players must fight against each other using the cards. Each player must have real cards to play and each card has some markers on it so that they can be tracked by the PlayStation Eye® and augmented with a 3D model representing the card. The 3D models interact with the other cards in the game and with the user too. Figure 4 shows results of [3] (the advertisement of BMW on the left) and [4] (the Eye of Judgment™ game on the right).

Figure 4 - AR works of [3] and [4].

Regardless of the application, the concept of AR states that its applications must add some virtual information to the environment in real time. This information can be of any kind, such as a sound, an image, a video, a 3D model, but it must be inserted into the real world to add some information to the user. In general, this virtual information is a projected image or 3D model, and it is inserted in a display.

In the other hand, image-inpainting techniques modify the scene by gathering visual information in the scene itself and painting some pixels in it, in order to recover damaged parts or remove unwanted ones.

2.2 INPAINT

Automatic digital inpainting, or just inpaint, are techniques capable of restoring damaged image or video using image interpolation. According to [20], inpaint algorithms try to restore a region in an image with information acquired outside the region. Inpaint methods can be used to restore damaged images or video sequences, as said before, but it also can be used for intentionally removing unwanted patterns in image or videos.

Current inpaint algorithms may rely on many different techniques. They may be based on extrapolation of neighboring pixels, recovery of edges, curvature-driven diffusions, inpaint from multiple viewpoints [21] and even texture synthesis, as described in [22].

The most natural approach to solve the inpaint problem, to recover the pixel information of missing areas in digital images, is to mimic how professional image restorators manually perform inpaint. The authors of [23] state that these workers extend patterns from the edges of the region to be inpainted and then fill in intra-region accordingly. Two well-known techniques that follow these principles can be found in [20] and [24], and both have implementations available in the Open Source Computer Vision Library (OpenCV) [25].

Instead of using the conventional approach, another group of inpaint techniques treats the inpaint problem as a classification problem. In those kinds of techniques, the region to be inpainted is divided into smaller sub-regions and the algorithm searches the image for similar patches outside the region, choose one of the matched regions based on some criteria and copy the color information to the target area. A remarkable technique that follows this principle can be found in [22]. In this paper, the authors develop a generic technique for texture synthesis that is well suitable for inpaint purposes, but unfortunately does not run in real time. Figure

5 illustrates the work presented in [22], where an image with damaged borders is recovered using this algorithm.

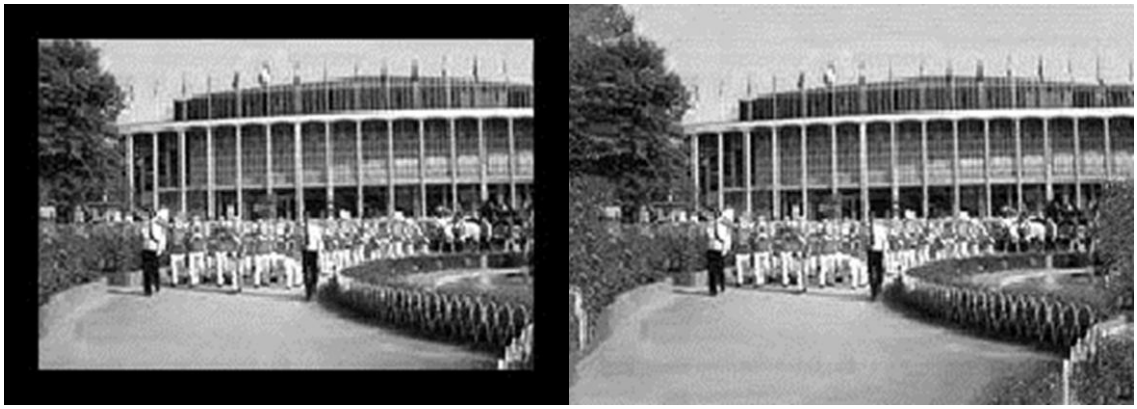

Figure 5 - Inpaint work of [22].

Inpaint techniques were initially proposed to recover damaged images, but many efforts have been made to develop good and fast video inpainting techniques. Video inpaint techniques can use previously mentioned image inpaint methods, gathering information just from the image frame to be inpainted, or rely on multiple cameras or temporal information, gathering information to inpaint a frame from the previous or from frames with the same timestamp, but at different points of view.

The video inpaint algorithms, as stated before, can take advantage of multiple cameras to perform the inpaint technique. When we record a scene with multiple cameras and want to remove an object in one of the views, it is possible that one of the other views currently represent the actual background behind the object. Those techniques usually try to match the views of all cameras so that they can take the background information from one of the views and use in another. . When using those techniques, the region to be inpainted should be marked in a frame of one of the video sequences, and the algorithm should match the region on the other video sequences so that the information to inpaint the object can be found. Besides matching the region to be inpainted in all video sequences, the algorithm will have to find a way to gather color information and fill in the region in the desired frames. A recent work that makes use of this type of technique is described in [26], which uses a grid of rectangles to help the inpaint process in a multi-view video sequence. In an initialization step, a user marks the region to be inpainted and, after that, the algorithm finds the region in the other views using the

mask clone method and it overlays the target region with a simple grid of points. Then the algorithm finds, for each rectangle grid, a matching patch in the other views, so that it can transform and paste them in the target frame. Figure 6 shows this work; the top images show two different views at the same time, the middle images show the selection of the target area to be inpainted, and the bottom images show the results.

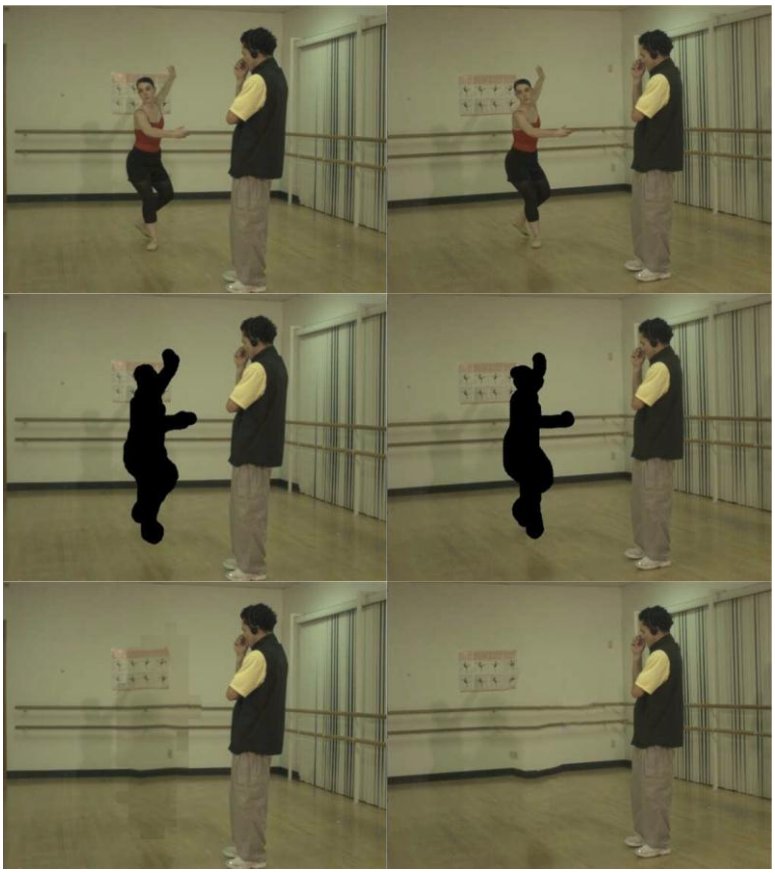

Figure 6 - Video inpainting of [26].

Temporal information can be useful to video inpainting algorithms too, since more information about the scene not present in the current frame can be found in any of the previous or future ones. One recent research describing the use of this kind of information was detailed in [27], where the author develops a technique to inpaint a video sequence with some constraints in camera motion. In this work, an inpaint algorithm was developed in a way that it can fill the background or any moving object, even with another object in front of or behind it.

The adopted method consists of a pre-processing step plus two inpaint steps. In the pre-processing step, each frame is segmented into foreground and background and three mosaic images are built using the segmented data. The mosaics will be useful to optimize the search step in the inpaint process. To perform the inpaint, first, the foreground objects that are occluded by the target area are reconstructed, when possible, by gathering information about moving objects on the foreground in previous frames, using a priority-based scheme. Then, the algorithm fills the remaining holes in the background by aligning previous frames and copying color information when possible. The remaining pixels are filled using a texture synthesis extended to spatiotemporal domain. Figure 7 shows the described technique; the top images show three views at the same time, the middle ones show the selection of the area to be inpainted and the bottom ones show the results.

Figure 7 - Video inpainting of [27].

Inpaint techniques can be used to recover damaged images and videos, and can be used to remove some unwanted parts on them. Optional characteristics of inpaint techniques are requesting of user information, real time performance and tracking of the inpaint area. On the other side, Diminished Reality techniques

demand tracking of objects or unwanted patterns on the environment and removal those from the scene in real time.

2.3 DIMINISHED REALITY

While AR applications are used to insert virtual information on the environment in real time, Diminished Reality (DR) relies on the opposite: to hide an object from the scene in real time, to provide convenience to the user and make him immersed in the environment [28]. DR applications can be used to remove objects synthesizing the background area occluded by them, or displaying the actual information behind it, using information from other cameras or previous frames.

DR applications follow the same rules as AR applications: the system must track a target in the scene in order to retrieve the position of the camera(s) in the environment and it must modify the scene correctly in real time. It is possible to use any tracking technique that is able to discern between different objects, because the system must keep tracking the object to be diminished.

The diminished information is visualized, in general, through a common monitor or a mobile one, as described in [29], but other unusual devices can be used to this purpose. As example, a projection-based DR system is developed in [28]. This system uses image completion to synthesize the area where the target object is, and employs radiometric compensation techniques for seamless projection onto the object. Figure 8 illustrates this work (original scene is on the left).

In [29], a fast approach for a DR technique is presented. It requires a simple setup and does not make use of any pre-processing or information about structure and location of the targets. This approach is based on the identification of the object to be removed using an active contour algorithm for object selection and object tracking and an image completion algorithm based on patches filled with information from the rest of the image. This algorithm uses CPU parallelization to achieve good performance results, and it presents good visual results as well.

Figure 9 shows some results of this work: the original scene on the left, the selection on the middle and the result on the right.

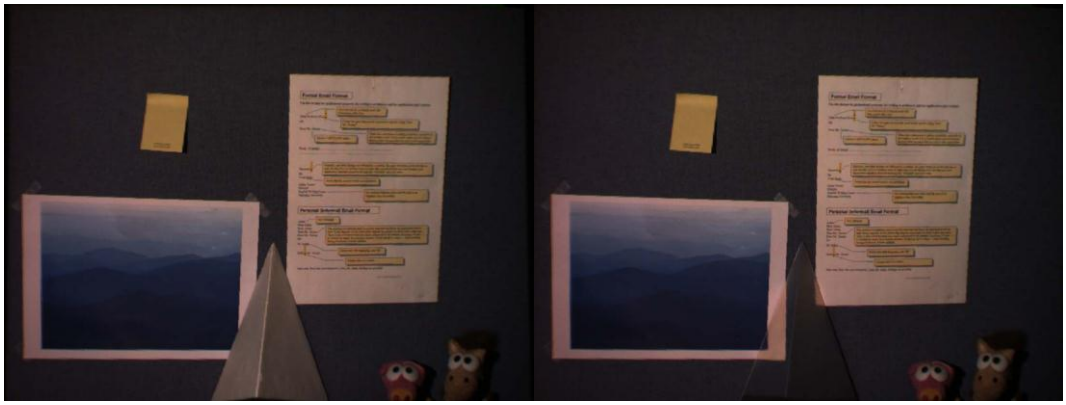

Figure 8 - The projection-based DR of [28].

Figure 9 - The DR techniques described in [29].

Another relevant work about DR is the one presented in [30], in which the authors develop a method to calibrate multiple handheld cameras that will be used by the application. This application removes some moving objects on the scene by segmenting the images onto background and foreground. By using information from all cameras, they are able to render the background without any moving objects, showing only the desired ones.

An innovative work that combines AR and DR to neither remove nor add objects in the scene is presented in [31]. In this work, the authors aim to modify a real object in real time in a scene by diminishing it in a first moment and then superimposing it with a purposely-modified version of it, with a dynamic texture applied onto it. The result of this AR + DR system is a virtual modified object that looks like the real one, as seen in Figure 10 the original cube on the top-left image, and some modifications on the five another ones.

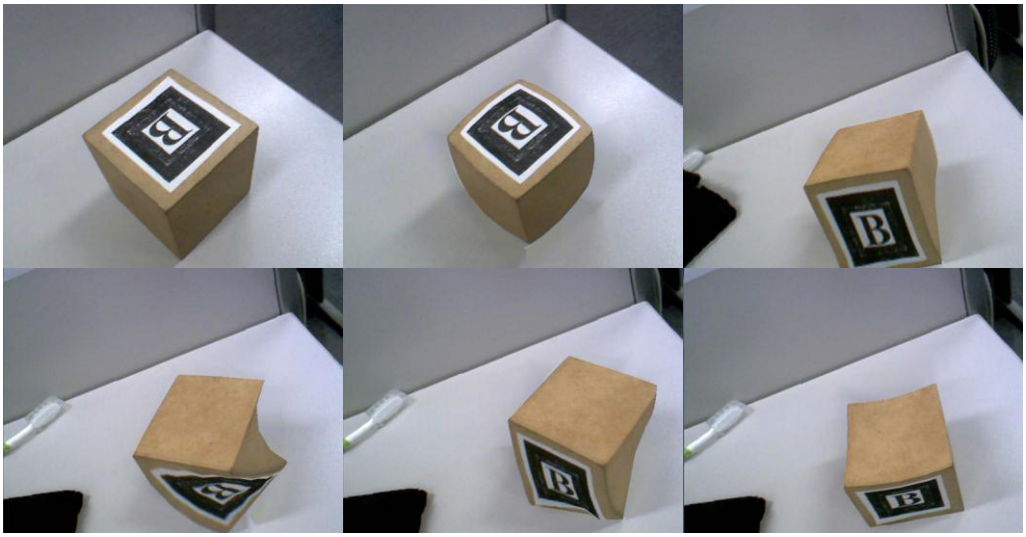

Figure 10 - The AR + DR system described by [31].

As mentioned before in the work of [29], CPU parallelization can be used to achieve better performance results, when the algorithm to be optimized can fit a parallel implementation. However, CPU parallelization has a serious limitation: the number of processors to be used. Although CPU processors have higher clock speed than Graphical Processing Unit (GPU) processors, the limitation on the number of processors is still a problem to be solved. On the other side, GPU processors do not have a high clock speed (in comparison to the CPU ones), but one single video card can have hundreds of GPU processors working in parallel at the same time.

2.4 GENERAL-PURPOSE COMPUTING ON GRAPHICS PROCESSING UNITS

The ever-increasing demand for high-speed graphics processing driven by the rapidly growing computer games industry has led to the development of powerful, parallel, multi-threaded, many-core processors called GPUs. As an example, the NVIDIA GeForce GTX 590 has 1024 cores in a single chip, and a peak performance of about 2490 GFLOPS [32]. Added to this performance, it is also possible to combine two or more GPUs in a single workstation, making a desktop computer become a massively parallel supercomputer [33].

Until some years ago, all the processing power from GPUs could only be used in graphics applications, since the available programming models for GPU

programming were graphics-specific, such as OpenGL and DirectX. Many problems could benefit from the use of Graphics cards, but the problems involving the adoption of the available programming models were a big obstacle.

As an example of these graphics cards capabilities, a parallelized GPU version of a well-known computer vision library, the OpenCV, is detailed in [34]. They used OpenGL to build the library, and the results were up to 18 times faster than the original CPU version.

The introduction of general-purpose programming models, such as CUDA [35] and OpenCL [36], has made General-purpose Programming on GPUs (GPGPU) a feasible alternative. Recent researches show that GPGPU programming can be used to speed up a variety of different algorithms in many areas of knowledge: linear algebra [37], image processing [34], and life sciences [38].

2.4.1.CUDA

In November 2006, NVIDIA introduced CUDA™, a general-purpose parallel computing architecture that leverages the parallel computing engine present in NVIDIA GPUs to solve many computational problems in a more efficient way than on a CPU. CUDA allows the developer to use C, CUDA FORTRAN, OpenCL and DirectCompute as high level programming languages, and it provides a software environment as well [39].

The CUDA parallel programming model is designed to enable GPGPU while maintaining a low learning curve for programmers used to languages similar to C. It has three key abstractions that enable this easiness: a hierarchy of thread groups, shared memories, and barrier synchronization.

The CUDA architecture guides the programmer to divide the problem to be solved into coarse sub-problems that can be solved independently by blocks of threads, and each sub-problem into finer pieces that can be solved cooperatively in parallel by all threads in the block. This decomposition is highly scalable because one block can be scheduled on any available processor core, in any order,

sequentially or concurrently, in a way that a compiled CUDA program can execute on any number of CUDA cores that is available now. Figure 11 illustrates how a problem is divided, and how it fits into two different GPUs. This figure shows a multithreaded CUDA program fitting in two different GPUs, one with 2 cores and another one with 4 cores.

A recent work involving CUDA technologies and a MAR technique is presented in [40]. In this research, the authors develop a CUDA version of Ferns that achieves results up to 3 times faster than the original version presented in [18]. This work will be explained with more details in section 4 , when the proposed system`s architecture by this thesis will be described.

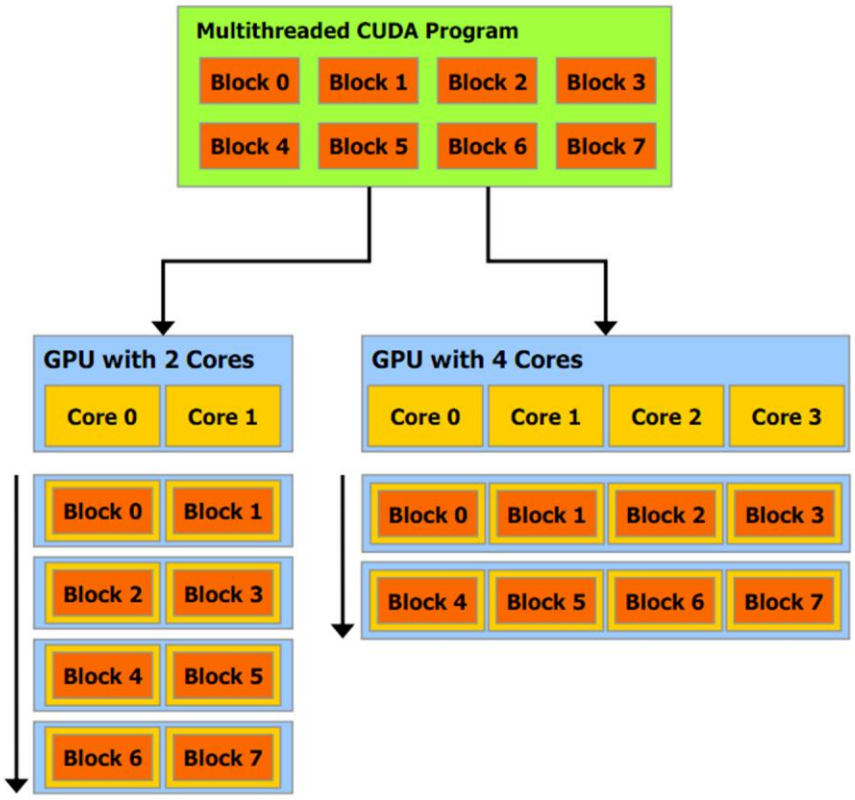

Figure 11 - CUDA approach to run a multithreaded program, as in [39].

In [41], the authors develop a DR system with multiple cameras that use Projective Grid Space (PGS) for calibration. A plane-sweep algorithm generates a depth map and removes unwanted objects, making use of GPU programming to achieve real time results.

In fact, all the mentioned works and techniques are of great importance to this thesis, as the work proposed here uses all these technologies. In order to accomplish the final objective, to aid maintenance, inspection and training procedures using AR and DR technologies using a wearable platform, we make use of GPGPU to guarantee a good performance of the application.

3 AR AND DR APPLIED TO MAINTENANCE, INSPECTION AND TRAINING

Inspection, maintenance and training are important activities regarding the manipulation of electrical and mechanical equipment. While inspection and maintenance are essential to prevent unexpected issues that can cause trouble to users and unnecessary costs to the company that possesses them, training is the key to make workers able to perform maintenance and inspections properly.

In general, trained workers executing established procedures to documented designs in relatively static and predictable environments conduct the activities in this domain. These procedures are typically organized into sequences of quantifiable tasks targeting a particular item in a specific location. These characteristics and others form a well-defined space, where a variety of systems and technologies can offer assistance [42].

This type of assistance is desirable, even for the most experienced worker, for two main reasons: navigating and performing repair and maintenance procedures imposes significant physical and cognitive requirements. The physical requirements come as he/she has to move his/her body, neck and head in order to locate and orient to the task. Assistance to optimize this part could save worker's time and energy and thus, prevent errors caused by fatigue. Cognitive requirements are needed because the person executing the procedure must spatially frame each task in a presumed model of the larger environment and map its location to the real world. He/She also has to correctly interpret and comprehend the tasks, and a help doing the mentioned parts could avoid mental fatigue and thus errors caused by it too.

In this context, AR and DR technologies can assist maintenance, inspection and training processes because of the fact that such techniques can add or remove visual information onto the target of the process and on its surroundings. AR systems could help the worker localize the target by adding visual information onto it and showing multimedia information related to the procedure being executed. DR systems could help the worker by removing unimportant visual information near the

target, avoiding diversion of attention and could be useful to maintain worker's focus onto target too.

3.1 STATE OF THE ART

Many researches have been done in order to assist maintenance, inspection and training processes with AR and DR technologies. In general, AR techniques add multimedia information over the target, showing the steps to be performed in an optimized way or directing attention to it, while DR techniques try to remove some information before adding another with AR techniques.

In [6], Caudell and Mizell proposed a seminal AR prototype to assist in assembling aircraft wire bundles. In the work of [7], it was found that the prototype performed as well as baseline techniques, but faced several practical and acceptance problems. Reiners et al. [8] created a prototype AR system that featured a tracked monocular optical see-through (OST) HMD that was used to present instructions for assembling a car door.

In [9], Tang et al. showed that using AR to assist the process of assembling toy blocks is more effective when registered instructions are displayed with a tracked stereoscopic OST HMD, compared to traditional media. The work of [10] demonstrated that subjects assembled toy blocks more quickly while viewing registered instructions on a video see-through (VST) HMD than on non-registered variants. Other works involving assembling with AR can be found at [11] [12] [13].

An interesting work about AR and inspection can be found at [14], where Ockermann and Pritchett develop a system to aid pilots performing preflight aircraft inspection by presenting instructions on an unregistered OST HMD. They have also demonstrated that the pilots had an undesired overreliance on the computer-generated instructions, sometimes ignoring previous knowledge about the procedure itself.

A work about attention guidance and AR is showed in [43], an application called Attention Funnel. They presented a system to guide the attention of the user using 3D line drawings, representing a vector funnel, from a screen-fixed position to

a possibly off-screen destination, as shown in Figure 12. They demonstrated that this approach leads to a 22% reduced search time.

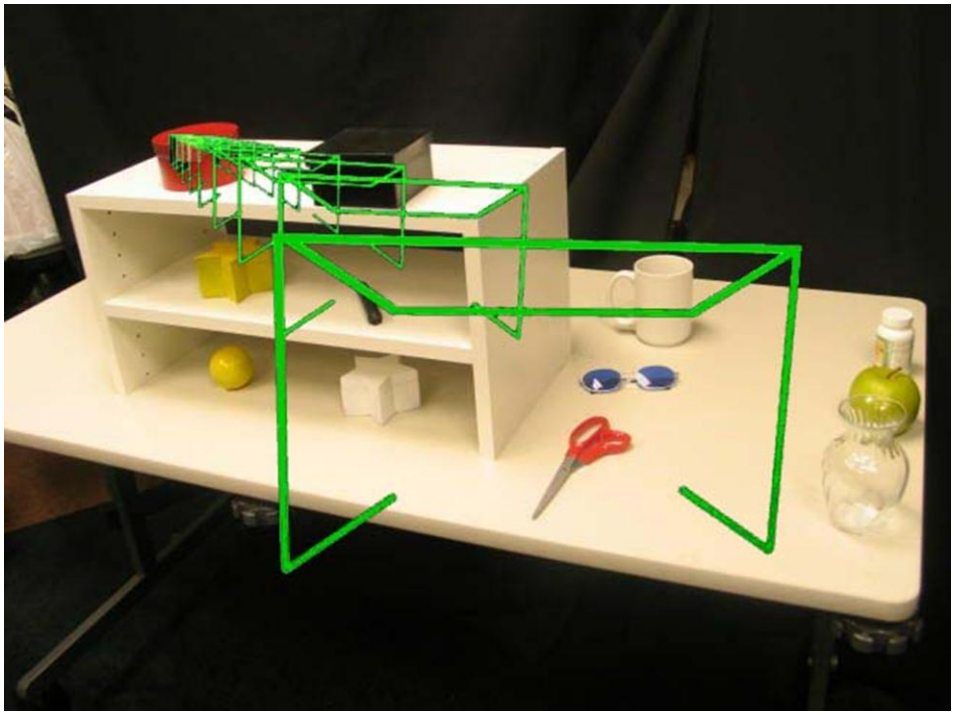

Figure 12 - The vector funnel application, as in [43].

A recent work that includes a vast description of the state of the art, a deep analysis of an existing AR solution to maintenance and good results can be found at [42]. It states that AR can reduce the time required to locate an object involved in a task, it can reduce head and neck movements during a repair activity and, according to authors, mechanics may be willing to tolerate shortcomings with HMD technology in case the system provides some help to the activity. They noticed in their tests that the virtual content can sometimes bother the mechanics, if it is in the front of the object being repaired, and mentioned that to solve this problem, all AR applications should have an easy way to dismiss the virtual content. Figure 13 shows the work of [42], in which a soldier is aided with some virtual instructions.

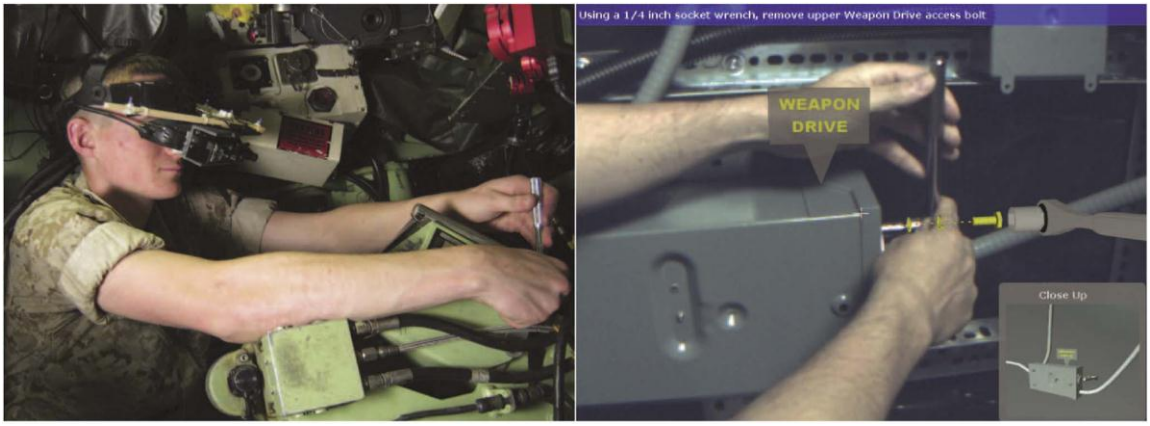

Figure 13 - The AR system to aid maintenance procedures of [42].

The works mentioned before offer a solution to maintenance/repair/training using AR technology, but many important problems remain open. More details about these problems and a possible solution to them will be discussed in the following sessions.

3.2 REMAINING PROBLEMS

There are many problems related to systems that try to aid maintenance, inspection or training procedures. The visual pollution of the environment is not treated in any of them, the easiness in the process of creating new maintenance procedures and getting the data required to create them are examples of those problems.

Visual pollution of the environment where the equipment is located, or even on the equipment itself can be a problem, if it starts to attract the attention of the person performing the inspection. Some tasks require concentration on a single point on the target, and distractions that could deviate the attention must be avoided.

The whole process involving systems that aid maintenance procedures is complex. These processes must start with the documentation of the maintenance itself, so that it can be imported to the system and the virtual instructions can be shown to the operator.. In order to allow the target equipment to be tracked, some information about the object must be provided to the system, and the type of this

information could be a problem too. Most of the works mentioned in the previous section uses specific model-based AR techniques to track the equipment and adds a constraint that determines that any equipment being a target of a maintenance-assisted procedure must have a 3D model to be used as an input to the system.

The next section proposes a generic solution that utilizes a tracking technique based on textures instead of 3D models. It has an editor that enables the user to create new procedures in an easy way, positioning virtual cues, highlights, and marking some information to be removed from the scene, in real time. The wearable platform that the mechanic should use when performing the maintenance is also proposed.

3.3 PROPOSED SOLUTION

The solution proposed in this thesis is more than a software solution for tracking some previously known equipment, and showing stored instructions in a registered HMD. It consists of an architecture involving three components, each one with its own internal architecture, that intercommunicate to solve the desired problem: a software, here called Editor, a wearable platform, here called miva, and another software, called Executer. The Editor is a system that allows any experienced worker to create maintenance procedures using a built-in workflow editor and pictures of the target equipment, in order to place the instructions as a result. The wearable platform consists of a computer, a battery pack, an HMD, a camera and an input device, arranged in a single platform. At last, the Executer is a system that is capable of reading the maintenance procedures created by the Editor and providing them on the field. It must track the equipment using the camera and show the stored instructions in the right place.

First, a maintenance procedure is created in the Editor, using just pictures of the equipment. After that, a file is generated, representing all the steps and containing the data needed to track the targets. This file is loaded into the Executer, which will run on miva platform, and the procedure can be executed. Figure 14 illustrates this communication scheme in detail: the engineer will feed the Editor

with the maintenance data, the Editor will generate the digital data and the Executer will load it, helping the worker in the maintenance activity.

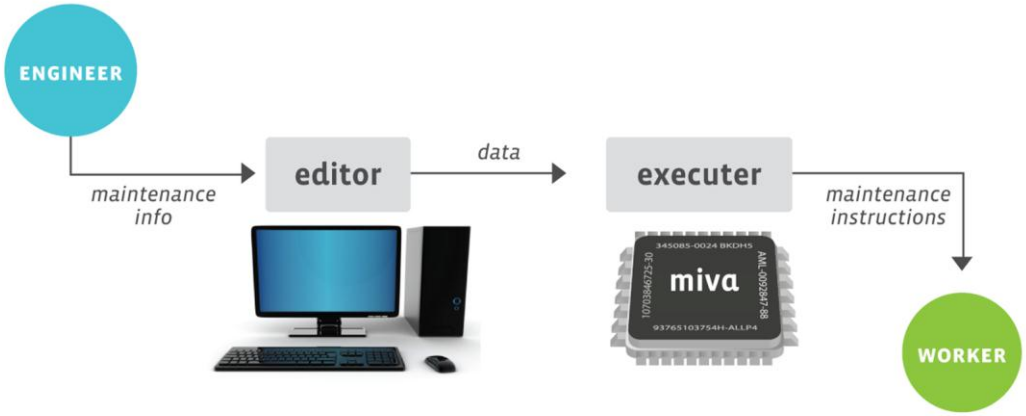

Figure 14 - The communication scheme between different parts of the proposed solution.

3.3.1.THE EDITOR

The editor is responsible for creating the data that will be used later by the Executer. It consists of a workflow editor, where the user can model a maintenance procedure, a detector trainer that must receive images as inputs and generate the data needed to track the targets, and a generator that will gather all data from the procedure and the trainer and generate a file readable by the Executer.

In the workflow editor, the user must be capable of creating steps, defining inputs of data to be entered by the person performing the maintenance, creating conditional Steps that will occur depending on the data gathered on previous steps, and defining highlights to be rendered over the target equipment, showing important information about the current step. Different contents can be added as information attached to the current step, such as video, audio or an image file.

The main objective of the detector trainer is to generate the data needed for tracking the equipment during the maintenance. The user must be capable of loading an image to the system, selecting a region of interest within the image, and the system must be capable of telling if the tracking can be done with the portion of the image selected. If so, it generates the data to be used by the Executer. Figure

15 shows the architecture of the Editor in detail: the user will enter all the data in the visual editor; then, there will be a refining process between the trainer and the visual editor, so that the user can select good images to generate the tracking data. In the end, the generator will group the information so that the Executer can read it

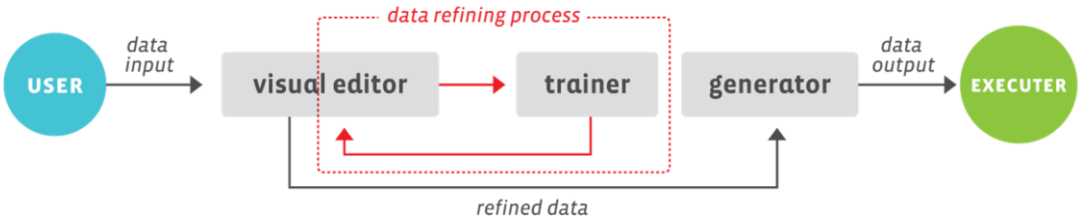

Figure 15 - The Editor's architecture.

3.3.2.MIVA PLATFORM

The wearable platform called miva, used in this thesis, is an evolution of these two previous platforms [44], [45]. miva, which means Mobile, Interactive, Virtual and Autonomous, was built to serve as a lightweight, but powerful computer platform that can be used by a worker aiding a specific procedure with AR data, but without imposing hard restrictions on his/her movements. This platform consists of a wearable computer that will be used on the field.

miva is composed of a small computer (later attached to a belt), a battery pack, an input device (later attached to the left or right arm of the worker), allowing him/her to enter input data to the system without imposing any restriction on his/her movements, an HMD to show virtual information, and a camera. All these components must work together in order to get the system working properly. The components of the platform and their interaction can be seen in Figure 16. The Arrows represent all the connections between the components.

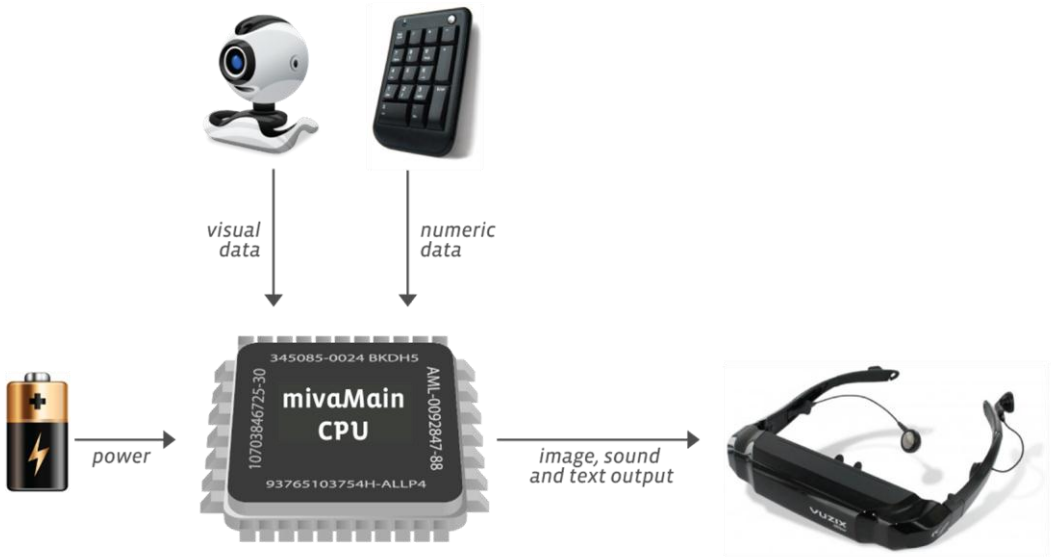

Figure 16 - All components of miva.

3.3.3.THE EXECUTER

The Executer can be considered the most important part of the proposed solution, because it is the one that will assist on the maintenance process. It must be capable of reading the data written by the Editor, track the objects that were previously chosen, and show all the registered cues and instructions stored in the procedure data in real time.

As this software will run on miva, it will use the camera and the numerical input device as input devices and the HMD as an output device, showing the scene with all the information added to it. It must also provide an easy way to remove all the virtual information from the scene too.

The topics discussed until now form the architecture of a system that aids maintenance procedures with AR technology, but it is just architecture. Each system to be implemented based on this architecture should utilize a tracking technique that better suits target's characteristics, it should have the peculiarities inherent to the maintenance being aided and the miva platform should be adapted to support the environment where the activity will be executed. The next chapter discusses the implementation of this architecture in a case study with CHESF, the

São Francisco Hydroelectric Company, which is the main electricity production company located in the northeast part of Brazil.

4 MIVAMAIN

As explained before in chapter 3, this chapter will describe mivamain, an implementation of the proposed architecture applied as a use case to CHESF, an electrical supply company in Brazil. Here it will be depicted which hardware were used in the platform as well as which technologies were chosen to be used in this case study.

The beginning of this case study has started with the definition of an initial planning, and all the described features and technologies chosen are in agreement with that planning.

4.1 MIVA-PLATFORM

The miva platform, as assembled to CHESF, has a miniaturized desktop computer: Zotac Mag. This computer has a 160GB hard drive, 2GB of RAM DDR2-800, an Intel® ATOM 330 (dual-core) processor, with a NVIDIA ION chipset. The dimensions of the Zotac Mag are 186mm x 189mm x 38mm, which makes it ideal for a mobile platform. The NVIDIA ION chipset has an integrated CUDA enabled GPU, with 16 processors, each one with a clock of 450 MHz. As it works similar to desktop computer, and it was not designed to be a mobile platform, therefore it has no battery, or another power source. For this reason, it was also included in the platform an external battery pack.

As input device, it was chosen a simple numeric keyboard to be attached later to the user's arm. The keyboard may be used to provide any input required by the application and to navigate through the Steps. As output device, some kind of HMD should be accomplished to the platform in order to allow the user to visualize the virtual information without having to hold any visualization device. For this case study, a monocular optical see through HMD was used without the see through feature. The HMD is a Liteye LE-750A, with a maximum resolution of 800x600 pixels.

The platform (as described before) is an extension of the works found in [44] and [45]. In this work, the authors have built a wearable computer, called miva. In its first version, this computer were in an acrylic backpack while in the second one it was attached in a belt, giving much more freedom to the user´s movements. Both of them used a data glove as input, and a HMD as output device. In our extension of this platform, the computer and the battery pack will be attached to the belt, as in the second version, but it will have a numeric keyboard too, to be attached to the forearm of the wearer. As in the previous versions, the HMD will be placed on user's head. Figure 17 shows the two versions of miva, the former on the left, and the second on the right.

Figure 17 - Miva versions.

4.2 SOFTWARE ARCHITECTURE

This description of the system's architecture will discuss the main decisions on what technologies were used in each one of the components of the Editor and the Executer. In addition, some former alternatives used during this thesis will be briefly discussed.

4.2.1.EDITOR

As stated before, the Editor, one of the three parts that compose the proposed architecture, has its importance. It will be a software tool, running on a desktop computer, which will be used by trained engineers to build and modify maintenance procedures to be later executed in field, by the technicians. It will be responsible for creating and saving the procedures in a known file format, so that the Executer can recognize it. In addition, it must generate all the data needed by the Executer to track the targets used on the procedures.

The first step in the creation of this tool was the definition of a maintenance procedure in a simple digital format. Therefore, it was proposed the following definition: a whole maintenance is a set of Steps, with one of them marked as the initial one. Each Step must have a label that is a unique identifier automatically generated by the tool, and a name. Optionally, it can have a description, an image file, a video file, and a sound file.

Each Step can query the user for an Input value. This Input value is unique in the whole procedure, and will be set just once. It was defined that this Input could be a numeric or yes/no (Boolean) value; it must have a name and optionally a measure unit. These values can be used to decide which is the next Step, and latter generate a report at the end of the maintenance process.

Every Step, except the ones marked as final Steps, should have a Branch. A Branch, in this context, is a unidirectional link between two Steps, representing a transition from a Step to another one. A Step can have more than one Branch, but one of them must have no Conditionals.

In our scheme, a Conditional is a test between two values that can be attached to a Branch to ensure that the maintenance procedure will only jump between the two Steps within the Branch only in case the test is true. A Conditional is a relational (equals, different, less than, more than, less or equals than, more or equals than) or logical (AND, OR, XOR, XNOR) operation between two previously asked Inputs, or between an Input and a constant value. In addition, more than one Conditional can be attached to a Branch, and all of them must be true in order to the maintenance jump between the two Steps in the Branch.

Targets can be defined also in the maintenance process, and they represent equipment that must receive visual hints over it, using augmented reality techniques, or ones that must have some part of it removed from the scene, with diminished reality algorithms. A Target is defined by an image, a unique name, and a detection quality, that will later influence in the speed and accuracy of the detection.

After defining the Targets, it is possible to assign them to as many Steps wanted, and some visual Highlights can be placed over them in each individual Step. In the proposed model, a Step can have a Target, which must be selected from the existing ones, and the Target can have one or more Highlights, that will appear over the Target during the maintenance. Each Highlight has a position, two decimal values relative to the target area, where 0.0, 0.0 represent the top right corner, 1.0, 1.0 represent the left lower corner. Any value out of this interval is outside the target area. A value for the size can be specified too, as a scale factor. A 1.0 value maintain its original size, 80x80 pixels, a 2.0 value double its size and any value smaller than one will reduce it.

As previously proposed, this software will use Diminished Reality to remove unimportant information from the screen, aiming to increase user's attention. Therefore, it was stated that every step that has a target can have also a Diminished Reality Mask. In this scheme, this mask is represented as an image, with the same size as the Target that it will diminish, with white and black pixels. In this context, white pixels represent the area to be diminished.

The result produced by the Editor is saved in XML (Extensible Markup Language) format. XML is a simple and very flexible text format, maintained by W3C (World Wide Web Consortium), widely used on the Internet [46]. The XML file contains all textual data about the maintenance, and a relative path to all external files. Figure 18 shows an example file, representing a simple maintenance with the elements described before (some elements were removed to reduce the figure size).

```
<Activity ResourceFolderName="Desktop" InitialStepLabel="L00001">
  <StepGroup>
    <Step label="L00001" title="Step 1" text="text" inputName="voltage">
        <Target targetDataName="equipment01">
        <HighlightGroup>
          <Highlight type="Arrow" x="0.23" y="0.54" size="2.2" />
          <Highlight type="Arrow" x="0"    y="0"    size="1"   />
        </HighlightGroup>
      </Target>
      <BranchGroup>
        <Branch destinationLabel="L00002">
          <ConditionGroup>
            <Condition operator="GreaterThan">
              <LeftOperand>
                <inputName>voltage</inputName>
              </LeftOperand>
              <RightOperand>
                <numberValue>40</numberValue>
              </RightOperand>
            </Condition>
          </ConditionGroup>
        </Branch>
        <Branch destinationLabel="L00003"/>
      </BranchGroup>
    </Step>
    <Step label="L00002" title="final 2" text="descrip" imageFile="eqp.jpg"/>
    <Step label="L00003" title="final 1" text="descrip" videoFile="int.avi"/>
  </StepGroup>
  <TargetDataGroup>
    <TargetData name="eqp" detectionQuality="High" imageFile="md.bmp"
trainingDataPath="eqp"/>
  </TargetDataGroup>
  <InputGroup>
    <Input name="voltage" type="Number" measureUnit="KW" />
  </InputGroup>
</Activity>
```

Figure 18 - An example of an XML file with a simple maintenance procedure.

C# and the .NET Framework [47] were chosen as the main technologies used to develop this tool. In addition, the IDE Visual Studio 2010, from Microsoft was used, and the project is stored in a cloud based SVN server.

To ensure the quality of this software, various design patterns were used during the development. As an example, Façade, Singleton and Multi-tier were used in the code. In addition, Unit Tests were performed with the help of a built in tool inside Microsoft Visual Studio 2010.

Figure 19 illustrates the Editor. The system has a main editor, where it is possible to notice the Steps being represented by orange boxes, with a label written over them, gray arrows representing Branches, which connect the Steps within the maintenance. Details about the selected Step can be seen on the lower panel, by choosing the proper detail node in the tree view, on the lower left corner. In order to

build the proposed workflow editor, we have incorporated in this project a general workflow editor, found in [48].

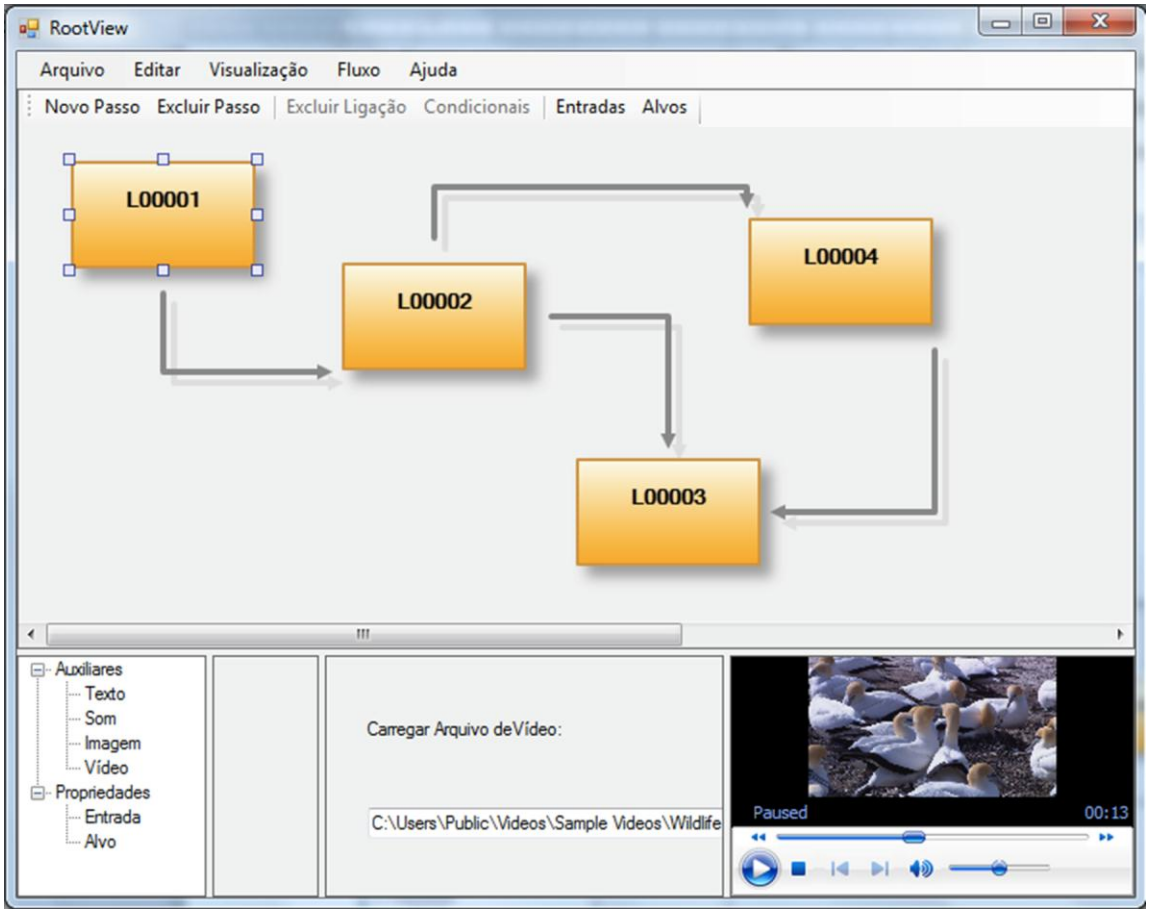

Figure 19 - The Editor's main screen.

The main actions involving the creation of Steps, Inputs and Targets can be executed through the buttons in a panel below the main menu. In this panel, there is a button to create a Step or remove the selected one, a button to delete the selected Branch, a button to manage the Conditionals of the selected Branch, and two more buttons, one to manage the Inputs, and other to manage the Targets of the maintenance. To create a Branch between two Steps, one must drag the mouse pointer from the desired connection square on the source Step (each Step has eight, one in each cardinal direction) to the desired connection square in the destination Step.

Figure 20 illustrates two secondary screens of the tool: the Input screen (bottom) and the Target screen (top). In both screens, the user can see a list on the left, and a box to add a new item on the right side.

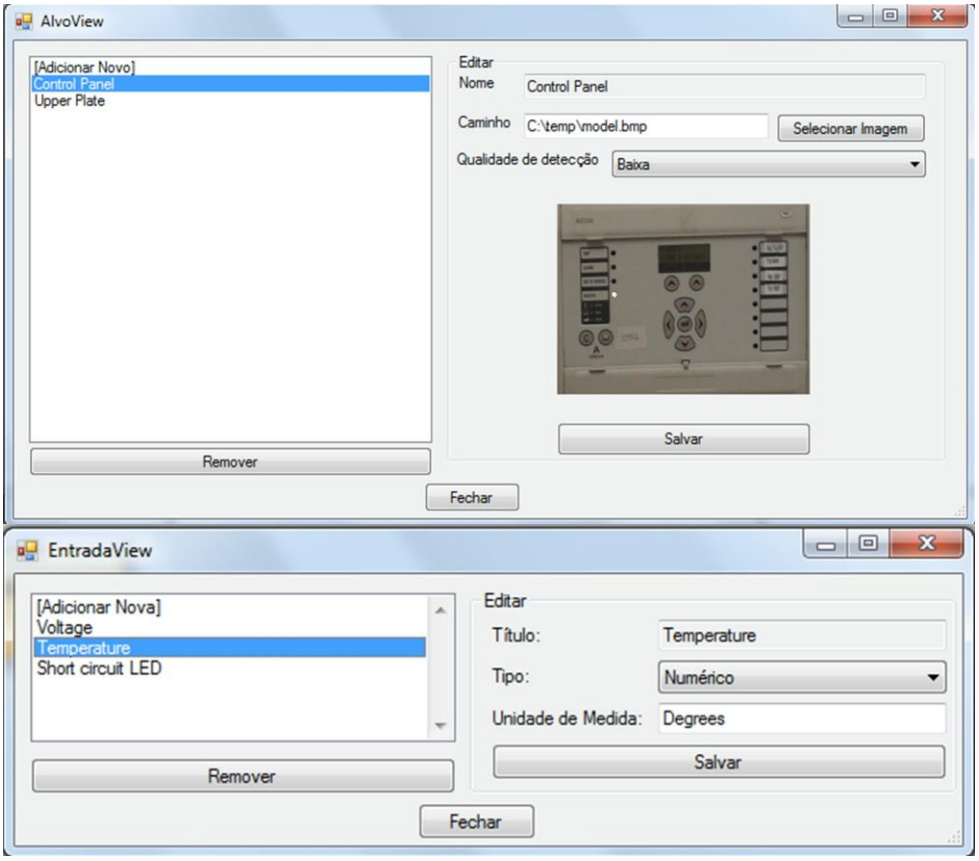

Figure 20 - The Target and the Input screen.

Figure 21 shows the Target screen. In this screen, there is a combo box to choose which target should be used in the selected Step on the left and a button to add Highlights to the Target on the right. The added Highlights, represented by the arrows, can be resized, rotated and moved around the screen. Note that the Highlights out of the Target area will be out of it during the execution of the maintenance procedure. In this interface there is also a button leading to the DR mask screen.

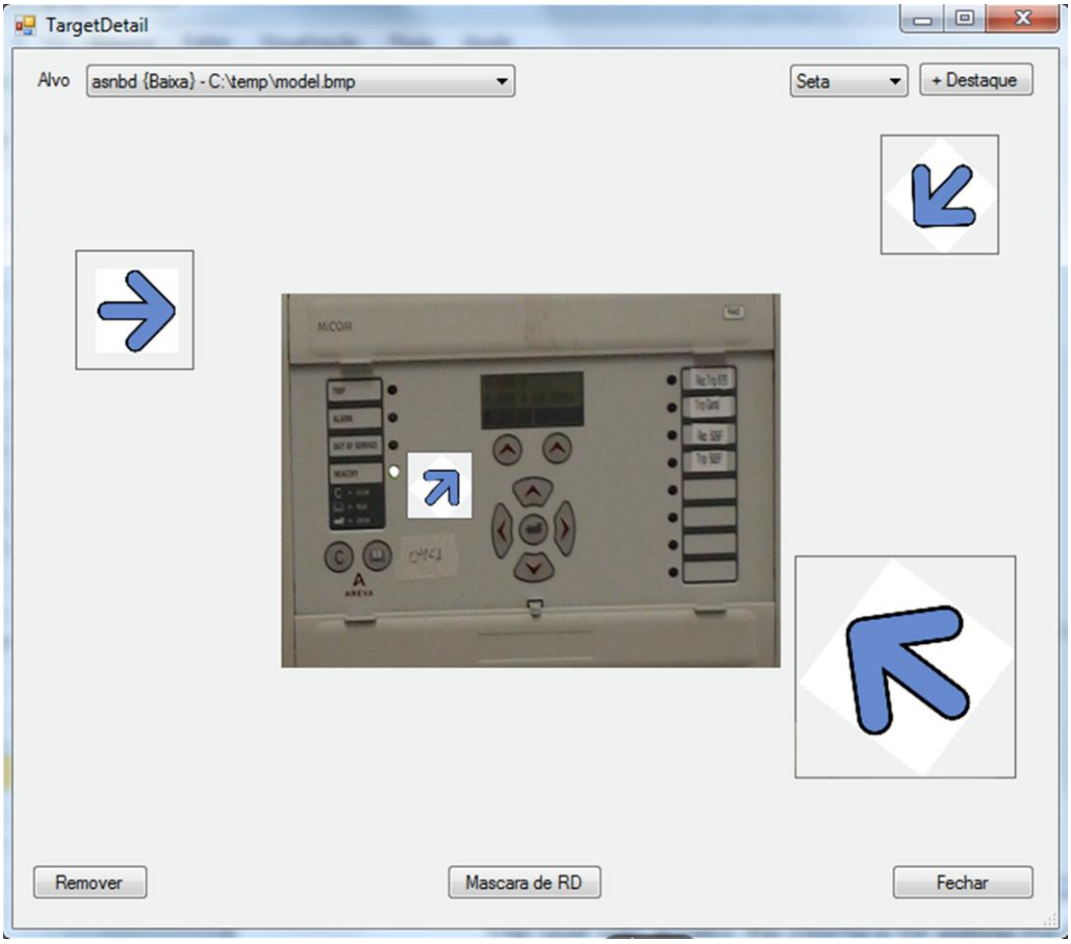

Figure 21 - The target screen.

In the DR mask screen, see Figure 22, it is possible to edit the mask by choosing the width of the pen with a slider, located at the right side of the window. Then, the user can paint the image with the mouse pointer and see the final mask on the right side. A button for clearing the mask is also available. In the image, the user has already painted a square in the middle of the target, covering a digital display, and two vertical dashes, covering some green LEDs.

Back in the Editor screen, when the user selects a Branch in the workflow editor, he/she can use a button in the panel bellow the menu bar to visualize and edit the Conditionals within that Branch. In this Conditional screen, shown in Figure 23, the user can add Conditionals to a list, choosing between a Boolean or relational Conditional and choosing one of the previously added Inputs or some constant value.

Figure 22 - The DR mask screen.

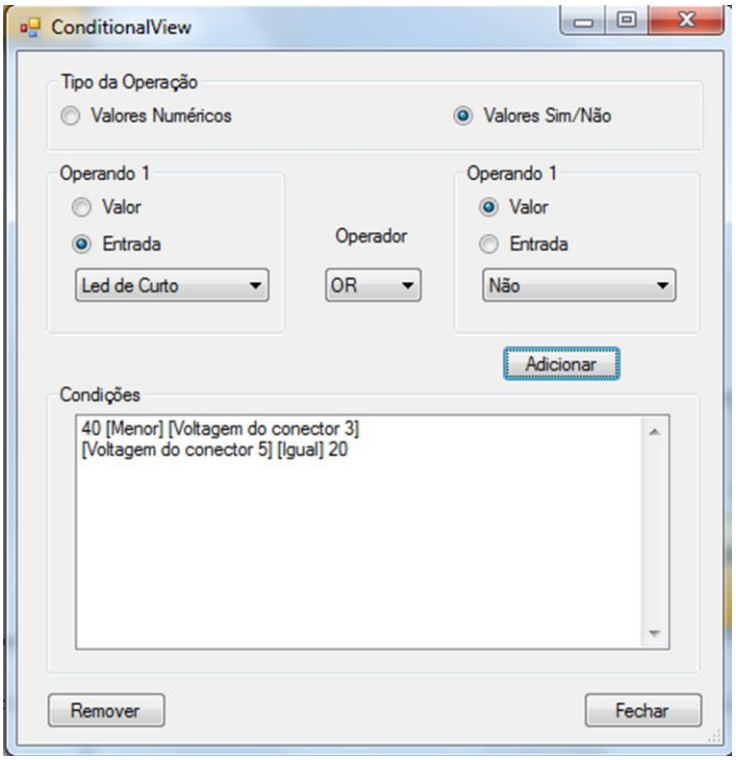

Figure 23 - Conditional screen.

When the user is going to save the procedure, the tool must look for logical errors, as using an Input value before it was asked for, and save an XML file with all the data. The external files are saved in a folder called Resources, in the same

folder as the XML file, and all the paths to them are relative. All the data needed to track the Targets is generated at the save time.

4.2.2.EXECUTER

The Executer, as it was previously stated, is the subsystem capable of reading the XML file created by the Editor, allowing the user to perform the maintenance assisted by it. It must read the XML, so the user can navigate through the Steps, detect the current Target, if present on the scene, adding the necessary Highlights, and ask for user input, when needed.

Unlike the Editor, this tool was built using C and C++ as development languages, and Microsoft Visual Studio 2010 integrated with QT Creator [49] were chosen as IDE for the development. In order to read the XML file created by the Editor, it was used TinyXML, a simple and small XML parser written in C++, found at [50].

The computer vision technique chosen to detect the maintenance targets was Ferns. This technique, created by Vincent Lepetit et al., is the succession of another technique, called Randomized Trees, in which the recognition problem is treated as a classification one. In their work, they prove that patches of an image can be classified by simply using randomized binary tests of luminance between pixels, grouped into structures called Ferns, which partition the space between all possible likelihoods. They treat all possible deviations of a patch as a class, and Ferns forms a probability distribution over those classes. A single group of tests cannot successfully classify a patch, but by utilizing many groups, or Ferns, as they called them, it is possible to obtain good results.

According to [18], given a patch, the algorithm tries to find the corresponding class that it belongs. Take $c_i, i = 1, \dots, H$ as all the classes, and $f_j, j = 1, \dots, N$ the results of each one of the N binary tests. Formally, they are looking for

$$\hat{c}_i = \underset{c_i}{argmax}\, P(C = c_i | f_1, f_2, \dots, f_N), \qquad (1)$$

where C is a random variable that represents a class. According to Bayes theorem, we can deduce that

$$P(C = c_i|f_1, \dots, f_N) = \frac{P(f_1,\dots,f_N|C=c_i)P(C=c_i)}{P(f_1,\dots,f_N)}. \quad (2)$$

Assuming that $P(C)$ follows a uniform distribution and that the denominator is a scale factor that does not depend on the actual class, we can assume that the problem is to find the solution for

$$\hat{c}_i = \underset{c_i}{argmax} P(f_1, f_2, \dots, f_N|C = c_i). \quad (3)$$

The previous formula could be understood as the following: for each class and for each possible result of each Fern, we calculate the probability of, given a patch that belongs to that class, the test results to be equal to the actual one. The main problem of this approach is the need for storing a probability value for each class, for each possible result in our tests, which corresponds to $2^N * H$ values.

Using a Semi-Naïve Bayesian approach it is possible to group the tests in small groups, and assume statistical independence between them. With this change, we have

$$P(f_1, f_2, \dots, f_N|C = c_i) = \prod_{K=1}^{M} P(F_K|C = c_i), \quad (4)$$

where the N tests were divided in M groups, with N/M tests, and F_K represents the result from the tests of the K^{th} Fern. By utilizing this approach, it is necessary to store only $M * 2^{N/M} * H$ values.

In a software engineering view, the algorithm can be divided in two parts: training and recognition. In the training part, the aim is to build the probability distribution of the classes, for each possible result of each Fern. Furthermore, the algorithm generates a big amount of homographies of the target image, extracting the most stable keypoints of them. For each one of these keypoints, the class which it belongs is known - as we have generated the homographies. Therefore, we build a probability distribution using the results of the tests for each keypoint detected in each generated homography. Figure 24 shows an example of a training procedure for three Ferns, each one with three different tests and five classes of keypoints. In this context, each color represents a class, and the patch being tested against the

Ferns belongs to the red one. Figure 25 shows the binary tests of the Ferns and the probability distributions at the end of the training, as proposed in [18].

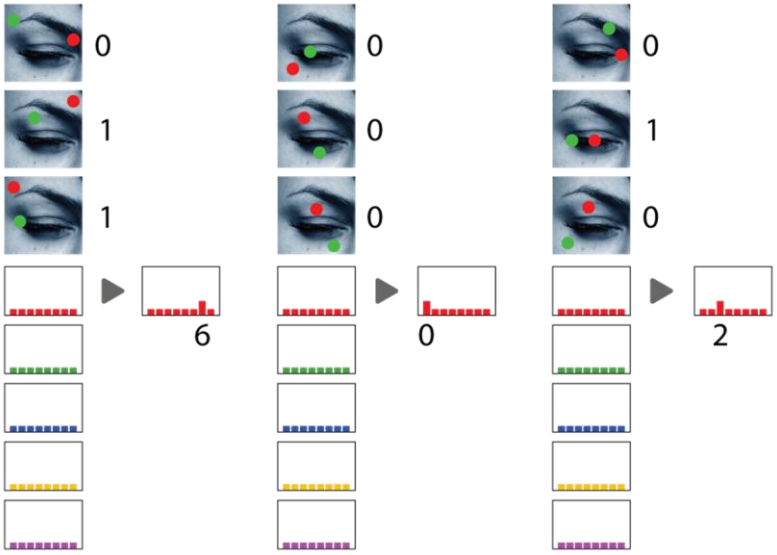

Figure 24 - A patch passing through the Ferns, during the training phase.

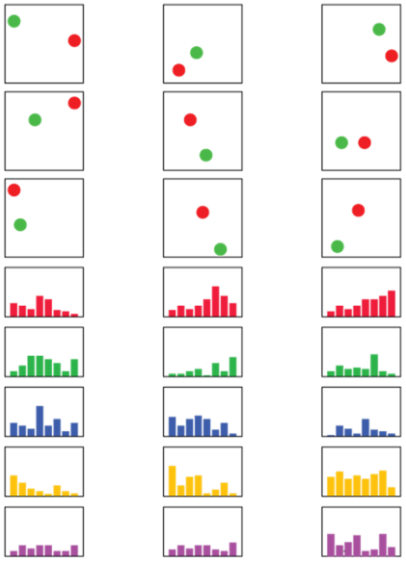

Figure 25 - Three Ferns and their probability distributions.

In order to recognize a given patch, we should pass the patch through each Fern, sum the distribution values for each corresponding class, and choose the highest one, as Figure 26 shows a patch passing through the Ferns. In this example, the most likely class is the red one.

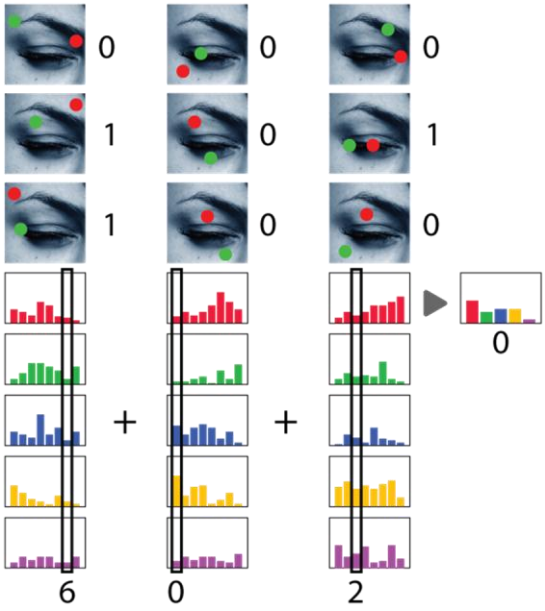

Figure 26 - Classification process for a given patch.

In order to improve the overall performance of this technique, we have used a parallel implementation of this algorithm, which can achieve results up to three times faster than the original version found in [40].

The parallelization exploited CUDA version 1.1, and was focused on the optimization of the recognition step of the Ferns algorithm. This stage of the algorithm has three main bottlenecks that can be targets using a parallel optimization: Gaussian pyramid build, keypoint extraction and Ferns classification.

In the first of these three phases, the original image is transformed several times to form a data structure called Gaussian pyramid, which will be used later when performing the keypoint extraction. This structure contains eight images: the original image, three downsamples of it, each one with half size of the previous, and a smoothed version of each one of the four images. Figure 27 shows a Gaussian pyramid of an example image. The original image is the top left one.. This bottleneck was fully implemented in CUDA, and the parallelization occurred inside each downsample and smoothing operation: the processing of each few pixels was made by a different thread.

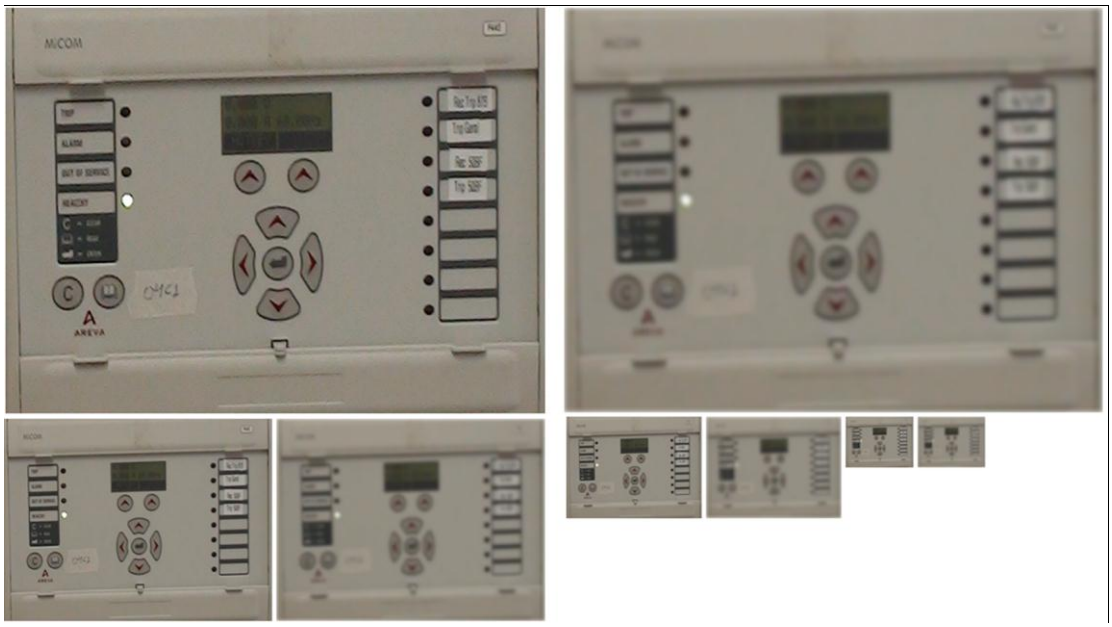

Figure 27 – An image and its Gaussian pyramid.

In the second step, the keypoint extraction, the algorithm must take each smoothed image of the Gaussian pyramid and calculate which pixels are strong candidates to be keypoints, passing through the Ferns in the next step.. This operation can be divided into two sub-steps: Laplacian Calculation for each image and Extraction of the strongest keypoints. Both the Laplacian Calculation and the Keypoint Extraction were, in addition, fully parallelized.

The last stage of the algorithm, the Ferns classification itself, is the biggest bottleneck in the whole process. It can be divided into two minor stages: the binary tests and the classification. Although these two parts were executed in one single step in the original implementation, we have chosen to divide them into two different parts, each one running in a different moment, for optimization purposes. In this implementation, each binary test of each Fern is processed in a different thread. In a second moment, all the test results are condensed, each class being processed in a different thread.

This parallelized version of the Ferns technique was chosen to correctly recognize the targets, allowing the application to place the visual highlights over it and to remove unimportant information from the target. In order to implement the Diminished Reality step on the Executer, we have decided to use the inpaint technique from [24], which has an OpenCV implementation available.

The Executer interface is showed in Figure 28. There are three images, the left one shows the actual interface of the Executer and the two on the right show the Executer using DR to remove one of the gauges at the panel of a small airplane. At the Executer interface, we can see textual and image data of the actual step on the top left side, and a Highlight being exhibited over the target. On the right, we can see the original image at the bottom, in which a panel with three gauges is showed. In the top image, we can see that the central gauge is missing.

Figure 28 - The Executer's interface and the Executer diminishing a Target.

Figure 29 and Figure 30 show the execution pipeline of the Executer. The first image shows what happens, in a resumed approach, for each frame retrieved from the camera: the system track the Target, if there is one in the current Step; if the tracking is successful, it apply the inpaint technique to remove the unwanted parts, and add the visual highlights, if the initial XML file had specified it. The second image shows the internal architecture of the software. After it loads the XML file, with all the necessary information about the maintenance, it creates an internal database with that data. Therefore, for each Step, starting from the initial one, specified in the XML file, the Executer must search in the database for media information, tracking data, and any input related to it, and show on the screen. As showed in Figure 30, if the tracking is successful, AR and DR operations are performed. If the Step asks for an user Input, it will wait until the user enter it; after it enter the data, or if the Step did not ask for one, the user can ask for the next step. At this moment, the Executer should check all the Branches of the Step and checking the Conditions in it. The first Branch in which all the Conditionals are true, or the Branch without Conditional, if none attends the Conditionals, will be selected.

The Maintenance will select the Step correspondent to the designated Branch, and will repeat the described steps until a Step without Branches is found.

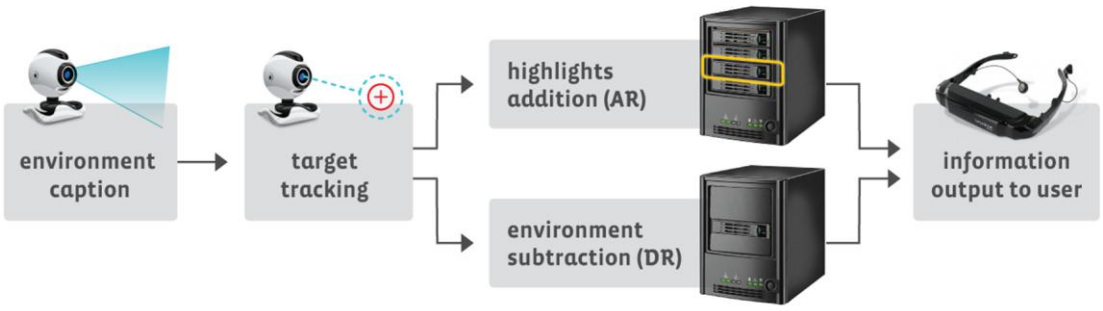

Figure 29 - The Executer pipeline runs for each frame acquired from miva's camera.

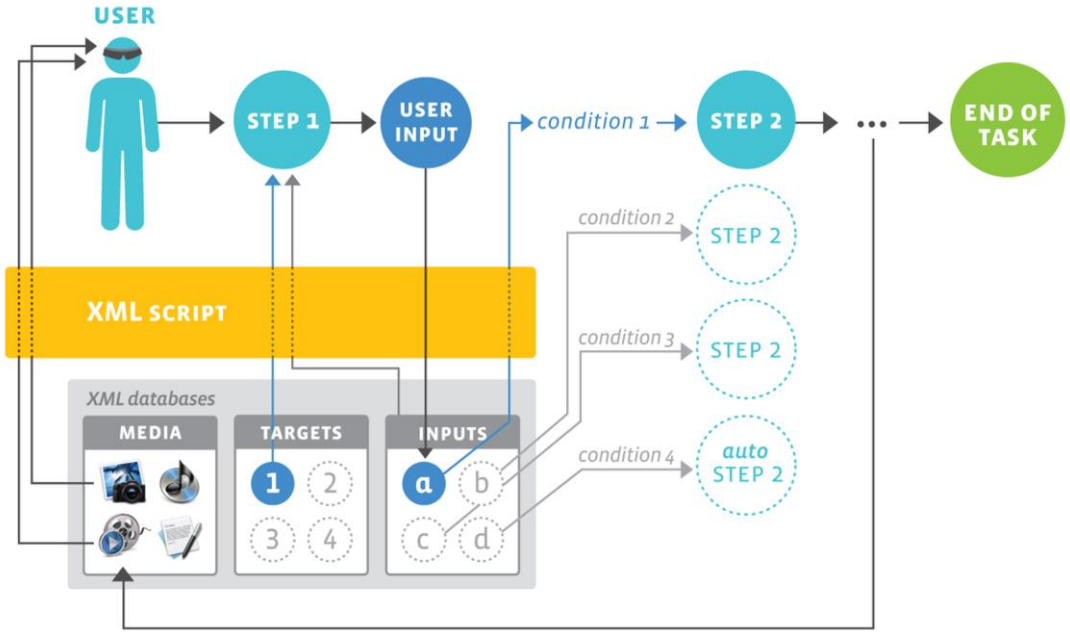

Figure 30 - Executer pipeline and its internal architecture.

4.3 SOME CONSIDERATIONS

During the development process of the system, many changes occurred, until we could reach the actual stage of it. In addition, it is worth to mention that it is not the final version.

The XML file structure, as an example, has passed through several modifications. We have included Branches, Conditionals to the Branches, Inputs

and Targets in each step. In another moment, we have taken the Inputs and the TargetDatas, which were inside each Step before, and we have grouped them inside the procedure in another place, putting just a reference to them inside each Step. This has reduced the amount of data stored in each maintenance file.

The parallelized version of the Ferns algorithm has passed through many modifications too. Since the beginning of the optimization process, we have tried different arrangements of the single parts to be parallelized. One of the biggest obstacles in this optimization process is the target platform that will run the Executer: it can only run in an old version of the CUDA SDK (specifically the one with 1.1 capability) due to its available GPU.

The Diminished Reality algorithm had also a previous version. Before the OpenCV implementation of the technique found in [24], we have implemented our own inpaint algorithm, as found in [31]. It was a simple alternative that reached a good result, but its time was really close to OpenCV's when the area to be diminished was small, and the alternative of OpenCV has a better visual result.

In the Editor, different alternatives were tried in diverse parts of the architecture. The main panel, where we can see the steps and branches, for example, is not the first version of it. In our first alternative, we have tried to implement a panel just using C# and Windows Forms components. Later, we decided to change the implementation to include WPF in this singular part of the interface.

5 USABILITY EVALUATION

In this chapter, we will present a usability evaluation of the proposed solution with specialized designers so that they can point out the principal improvements and changes needed by a new version of this tool. The evaluation held in the scope of this thesis will cover only the Editor software. The reason to skip the evaluation of the Executer and the miva platform is because both are in an initial version, and a relevant evaluation could not be done at this moment.

5.1 MAINTENANCE, INSPECTION AND TRAINING

In the context of CHESF, the utilization scenario of the developed tool will be simple: a qualified engineer will build the maintenance, inspection or training procedures with the Editor, ensuring that all the steps, highlights and descriptions will help the worker to accomplish the desired work. After this creation phase, any worker will be able to realize the procedures using the Executer running on miva platform. Therefore, the Editor evaluation consists of a simple scenario, in which an engineer builds a short maintenance procedure that covers the main features of the tool.

5.2 EVALUATION METHODOLOGY

In order to evaluate this scenario, we will adopt a two-step methodology: evaluate the system with a cognitive walkthrough [51] and order of issues found in a priority list [52]. In the cognitive walkthrough, we will present the tasks that the specialists will have to achieve and, during the execution of those tasks, each objective will be taken into account. Then, they will have to point out every minor part that in which they have found some issue, and give them a grade from one to four, where one represents a minor problem and four represents a bigger one, following the Nielsen approach, as in [52].

We have chosen this approach, to test the system using a cognitive walkthrough using specialized designers, because this is the suggested evaluation

when the proposed tool is not in a final stage of development. In this type of evaluation, we can test a feature of the tool even if it is not complete. In this case, as the Editor is not in its final version, we have found this to be the suitable alternative at this moment.

As mentioned before, the evaluation will use a cognitive walkthrough. In this evaluation, the designers will be briefly presented to the system interface, to the tasks they will have to perform, but not to how they should perform them. Then, each designer uses the tool to perform the desired objective, and they write down each point they felt some difficult to do. After all the designers finish this phase, they discuss all the problems found, and they decide which issues should be in the final list.

Once the designers perform the cognitive walkthrough and build the list of issues, they discuss again, this time trying to order all the problems in a priority list. They take into account that we will use this priority list as an order for future improvements and corrections, and they organize them based on a score from one to four, following the Nielsen approach, as in [52]. A one score means a low-severity problem, that according to Nielsen is mostly a cosmetic or irritant problem that does not prevent the operator from using the system, but many of these problems could discourage him. A two or three score stands for medium-severity problems: the ones that can frustrate and confuse the user. A four, the worst score, means a high-severity problem: after facing this issue, the operator will not be able to accomplish the desired activity.

5.2.1.SCENARIO DESCRIPTION

The scenario to evaluate the Editor consists of the assembly of a simple maintenance procedure that involves all the features that were proposed in the initial planning. The user will have to build a procedure with five steps, three inputs and one target that will be the target to all the steps. All the steps must have a title and a textual description too.

The first Step must have a numeric Input, a Highlight, a sound file, and a Branch to the second Step. The second Step should have another numeric Input, a

video file, two Highlights and two Branches: one with a Conditional ensuring that the first Input will be bigger than the second one, and another without Conditionals. The former Branch will lead to the third Step and the second one to the fifth Step.

The third Step will ask for a Boolean Input, will have no Highlights, but will have a DR mask marking a circle in the center of the target, to be later diminished. It will have two Branches too: one with the Conditional “Boolean Input OR false” and one without Conditionals. The former will lead to the fifth Step, and the second will take the maintenance to the fourth. The fourth Step will just have two Highlights, a DR mask with four squares, one in each extremity of the target, and a Branch going to the fifth Step. The fifth and last step will have just one Highlight. Figure 31 shows in a resumed way the described scenario. The orange squares are the Steps, the purple balloons are Inputs and the arrows are Branches. While red arrows represent Branches with Conditionals, the blue one is the one without them. Each green box represents a Conditional within a Branch.

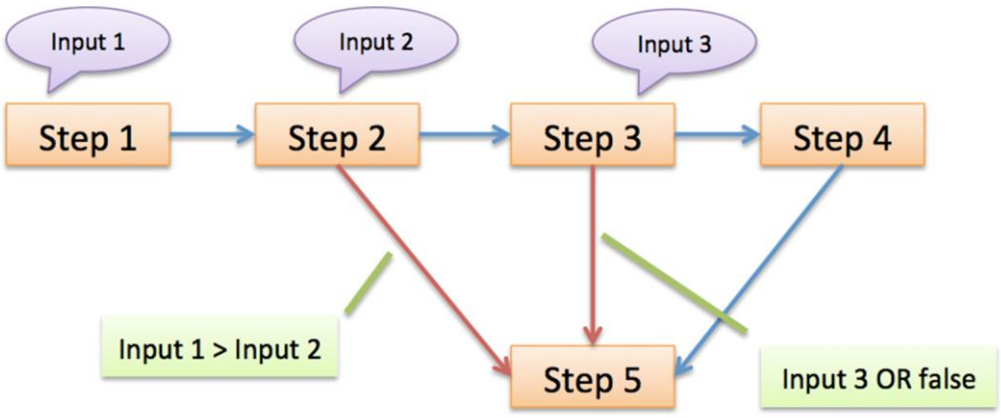

Figure 31 – The scenario used in tests.

5.2.2.EVALUATION

The specialized designers tested the system separately in periods of approximately thirty minutes. For this test, three specialists have performed the evaluation and all of them used the same platform. During the evaluations, one member of the development team was supervising them, providing assistance when needed.

Later, during the second step, the three specialists discussed together without any member of the development team, and they have taken a period of approximately four hours to accomplish this phase. After the execution of the tests, they gave as result the priority list, with an explanation of each issue found and a score of completeness, plus a list of missing features.

5.3 RESULTS

In Table 1 we can see a resume of the ordered issue list compiled by the specialists, where the score is on the left column and the description on the right. The issues are also grouped into two major classes: bugs and improvement points. Bugs are minor unexpected problems with features that were implemented but are not working, while improvement points are faults in the initial project that result in usability problems.

Most items presented as improvement points had a score of two or one (sixty percent of them, against 25.7 percent of score three and 13.3 percent of score four). Both problems pointed as high-severity ones are easily fixed just with some interface changes, as well as several problems with a three, two or one scores.

Based on this usability analysis, performed by specialists, we could conclude that the Editor software is mostly functional, and its features allowed the users to accomplish the most important activities related to the main objective of this system: build maintenance procedures.

The tests occurred in a short time and the users needed just a few minutes with the tool to get used to it, and built the whole maintenance of the scenario without major problems.

Table 1 - Issues list, with their respective score.

Score	Description
	Bugs
3	When adding Conditionals, some incorrect ones can be added.
3	The Steps on the workflow editor are not draggable, if we click on the label.
3	When one creates a new Input, they are not automatically added to the combo box of the Steps.
2	When one change the Target, while adding some Highlights, some unexpected effects happens.
2	The Highlight screen can be opened, even without any added Target.
1	Sound and video files, when added to a Step, starts playing when we change the panel's screen.
1	When one creates a Boolean Input, he/she should not be able to set the Measure Unit.
	Improvement Points
4	The lower panel should have buttons to create Inputs and Targets.
4	The lower panel should disappear when no Step is selected.
3	In the Conditional screen, we should be able to edit them.
3	In the DR Mask Screen, an undo button should be available.
3	When setting the Input of a Step, an option "None" should be available in the combo box, instead of the button "Clean".
3	The detail screens (Input, Target, Highlight, DR Mask and Condition) should follow a pattern: a Close button, or a Save button.
2	In the Input and Target screens, we should be able to edit their names too.
2	The squares on the corners of a Step, where we drag new Branches, should be circles.
2	The Input and the Target screen should show the list with the actual items on the right, and the fields to add a new on the left.
2	The mouse icon when rotating a Highlight should be a circular arrow.
2	Add a new Highlight is not intuitive (one must select the Highlight screen on the lower panel, set the Target for that Step and add it with a button "+ Highlight)"
2	The Steps should come with a provisory title (Insert title here) instead of an automatic label.
2	The Steps should allow the user to double click them, to edit the title right in the workflow editor.
1	Instead of 8 places to drag Branches in a Step, 4 should be enough.
1	The mouse pointer, in the DR Mask screen, should show the size of the pencil while moving it.

6 CONCLUSION

In this work, we have proposed the architecture of a system to aid maintenance procedures by adding visual cues and removing unwanted parts using AR and DR techniques and the architecture of a tool to build those aided procedures. We have also proposed an extension of a known wearable computer scheme, including all the necessary peripherals to be used in the field with the system itself.

In a second moment, we have made a case study with CHESF, in which we have chosen and acquired the hardware components to assemble a simple wearable platform, called miva, and we have implemented the proposed system in two parts. The Editor, a tool in which an engineer could build maintenance procedures using just pictures of the target equipment, and the Executer, a system to be executed on miva platform that is able of assisting the workers in the execution of the procedure by adding the visual cues and removing the unwanted parts that the engineer have specified.

A Chesf case study is part of a bigger project, called mivamain, in a partnership between the Virtual Reality and Multimedia Research Group (GRVM) from Centro de Informática - Universidade Federal de Pernambuco and CHESF. This project officially started in the beginning of 2012 and will take two years of development. Therefore, the miva platform and both software tools, Editor and Executer, are not in a final stage of development and some improvements will be done during the next 22 months.

In the evaluation of the system, usability specialists could evaluate the Editor software with a cognitive walkthrough, and point issues in a priority list based on their severity, so that they could be focused in the next steps of the Editor's development.

Since the beginning of this master thesis, many different researches were held, until we could focus on the actual work. Even though, each one of the previous researches had its own contributions to this work. In the first quarter of 2010, we started two research strands: one involving the optimization through

parallelization and GPGPU applied to a known AR technique, called Ferns, and another involving the use of AR and DR in a new approach that would modify the geometrical appearance of real objects in a scene, in real time. Both researches were published in the Workshop on Virtual and Augmented Reality (WRVA), [40] and [53].

The former work, involving AR and DR, was continued in an integration with NVIDIA PhysX and generated three other publications: a poster [54] and a demonstration [55], exhibited during the three days of the IEEE VR 2011, in Singapore, and a full paper [31] at the Symposium on Virtual and Augmented Reality, in Uberlândia, 2011, this full paper won the best paper award.

Later, we decided to focus the knowledge acquired from these works to build an architecture that uses AR and DR to aid some dangerous and relevant activity. Maintenance, inspection and training were chosen after we started the negotiations with CHESF to build this architecture and later a tool, to fulfill the needs inside its context: perform those activities on electrical equipment.

6.1 SOME CONTRIBUTIONS

There are three main contributions of this work that are worth to be cited here: the optimization of Ferns technique, a tool to aid maintenance procedures with AR and DR technologies and an architecture to aid those procedures without the need of a three dimensional model of each target to be tracked.

The optimization of the Ferns technique, as fully described in [40], has optimized the original technique from Lepetit et al. [18] up to three times by parallelizing the main bottlenecks of the original algorithm inside a GPU.

There is a contribution of a tool to aid maintenance with AR and DR technologies because in the literature, there is no work focusing on such aspects. Furthermore, all existing software similar to the proposed one, in which AR technologies aid maintenance procedures, use complex 3D models of the target equipment and thus imposes a strong restriction on which equipment's maintenance we can use the AR systems.

6.2 FUTURE WORK

As this work is part of a bigger project that will be executed for the next two years, there is a lot of future work to be done. The Editor, as described in the proposed architecture, in chapter 3, is composed of three main parts: the workflow editor, the trainer and the generator. The workflow editor is in an advanced stage of development, as well as the generator, but the trainer, responsible for training an image to be later recognized in the Executer, is not ready. For test purposes, we have used an external tool to generate the data to track all the targets. In addition, the chapter four, where we held a usability evaluation, gave general directions on which parts of the Editor we must focus in order to continue this development.

In the other hand, the Executer has more parts that are unfinished. It can read the XML file, load a maintenance procedure, display the visual highlights on the tracked target, and navigate through the loaded Steps, but in a simplified way. It still does not take into account the Conditionals in each Step, and it cannot load the media data within the Step (sound, video and image files). Furthermore, the removal of unwanted parts is working, but it does not work inside the main application yet - there is a separate application where we can see the results of the diminished target, as seen in Figure 28.

The miva platform, as will be assembled to CHESF, has all the necessary components to work in field, but it is not completely assembled. We must build the belt that will hold the computer and the battery pack, the camera must be assembled with the HMD, and both should pass through a calibration process to ensure that the images seen on the HMD will match the real environment in the right position. At the end, usability specialists must do a deep usability analysis on the miva before the utilization by the final users.

REFERENCES

[1] MILGRAM, P.; KISHINO, F. **A Taxonomy Of Mixed Reality Visual Displays**. 1994.

[2] TEICHRIEB, V. et al. **A Survey of Online Monocular Markerless Augmented Reality**. *International Journal of Modeling and Simulation for the Petroleum Industry*, v. 1, p. 1-7, 2007.

[3] BMW Augmented Reality. Available at: <http://www.bmw.com/com/en/owners/service/augmented_reality_introduction_1.html>. Feb. 14, 2012.

[4] THE EYE OF JUDGMENT™. Available at: <http://us.playstation.com/games-and-media/games/the-eye-of-judgment-ps3.html>. Feb. 14, 2012.

[5] FANGYANG, S.; YUE, S.; YUE, Q. **AR aided implant templating for unilateral fracture reduction and internal fixation surgery**. In: IEEE VIRTUAL REALITY CONFERENCE, 2011, Singapore. *Proceedings…* 2011. p.179-182. doi: 10.1109/VR.2011.5759459.

[6] CAUDELL, T. P.; MIZELL, D. W. **Augmented reality: an application of heads-up display technology to manual manufacturing processes**. In: INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, 25., 1992. Hawaii. *Proceedings…* 1992. v. 2, p.659-669. doi: 10.1109/HICSS.1992.183317.

[7] CURTIS, D. et al. **Several devils in the details: making an AR application work in the airplane factory**. In: INTERNATIONAL WORKSHOP ON AUGMENTED REALITY, 1998. Bellevue, WA, USA. *Proceedings…* Natick, MA, USA: A. K. Peters, Ltd., 1999. p. 47-60.

[8] REINERS, D. et al. **Augmented Reality for Construction Tasks: Doorlock Assembly**. In: INTERNATIONAL WORKSHOP ON AUGMENTED REALITY, 1998. Bellevue, WA, USA. *Proceedings…* Natick, MA, USA: A. K. Peters, Ltd., 1999. p. 31-46.

[9] TANG, A. et al. **Comparative effectiveness of augmented reality in object assembly**. In: INTERNATIONAL CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 14., 2003. Fort Lauderdale, FL, USA. *Proceedings…* 2003. p. 73-80. doi: 10.1145/642611.642626.

[10] ROBERTSON, C. M.; MACINTYRE, B.; WALKER, B. N. **An evaluation of graphical context when the graphics are outside of the task area**. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON MIXED AND AUGMENTED REALITY, 7., 2008. Cambridge, UK. *Proceedings…* Washington, DC, USA: IEEE Computer Society, 2008. p. 73-76. doi: 10.1109/ISMAR.2008.4637328.

[11] ZAUNER, J. et al. **Authoring of a Mixed Reality Assembly Instructor for Hierarchical Structures**. In: INTERNATIONAL SYMPOSIUM ON MIXED AND AUGMENTED REALITY, 2., 2003. Tokyo, Japan. *Proceedings…* 2003. p. 237-246.

[12] NILSSON, S; JOHANSSON, B. **Fun and Usable: Augmented Reality Instructions in a Hospital Setting**. In: AUSTRALASIAN COMPUTER-HUMAN INTERACTION CONFERENCE, 3., 2007. Adelaide, South Australia, Australia. *Proceedings…* New York, NY, USA: ACM, 2007. p. 123-130.

[13] SALONEN, T.; SÄÄSKI, J. **Dynamic and Visual Assembly Instruction for Configurable Products Using Augmented Reality Techniques**. *Advanced Design and Manufacture to Gain a Competitive Edge*. Springer London, 2008. p. 23-32.

[14] OCKERMAN, J. J.; PRITCHETT, A. R. **Preliminary Investigation of Wearable Computers for Task Guidance in Aircraft Inspection**. In: INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, 2., 1998. Pittsburgh, PA, USA. *Proceedings…* 1998. p. 33-40.

[15] NEWCOMBE, R. A.; DAVISON, A. J. **Live dense reconstruction with a single moving camera**. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 23., 2010, San Francisco, CA. *Proceedings…* 2010. p.1498-1505. doi: 10.1109/CVPR.2010.5539794.

[16] KinectFusion: Real-Time Dynamic 3D Surface Reconstruction and Interaction. Available at: <http://www.siggraph.org/s2011/content/kinectfusion-real-time-dynamic-3d-surface-reconstruction-and-interaction>. Feb. 14, 2012.

[17] Microsoft kinect SDK. Available at: <http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/>. Feb. 14, 2012.

[18] OZUYSAL, M. et al. **Fast Keypoint Recognition Using Random Ferns**. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 32, n.3 , p. 448-461, Mar. 2010. doi: 10.1109/TPAMI.2009.23.

[19] TARUMI, H. et al. **SpaceTag: An Overlaid Virtual System and Its Applications**. In: INTERNATIONAL CONFERENCE ON MULTIMEDIA COMPUTING AND SYSTEMS, 6., 1999, Florence, Italy. *Proceedings…* 1999. p. 207-212.

[20] AU, W.; Takey, R. Image Inpainting with the Navier-Stokes Equations. Available at: <http://www.eecs.berkeley.edu/~rrtakei/gradProj/930project.pdf>. Feb. 14, 2012.

[21] SHIH, T. K.; CHANG, R. C. **Digital inpainting - survey and multilayer image inpainting algorithms**. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY AND APPLICATIONS, 3., 2005, Sydney, Australia. *Proceedings…* 2005. v.1, p. 15- 24. doi: 10.1109/ICITA.2005.169.

[22] EFROS, A. A.; LEUNG, T. K. **Texture synthesis by non-parametric sampling**. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 7., 1999. Kerkyra, Greece. *Proceedings…* 1999. vol.2, p.1033-1038. doi: 10.1109/ICCV.1999.790383.

[23] BERTALMIO, M. et al. **Image inpainting**. In: ANNUAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES (SIGGRAPH), 27., 2000, New York, USA. *Proceedings…* New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000. p. 417-424. doi: 10.1145/344779.344972.

[24] TELEA, A. **An Image Inpainting Technique Based on the Fast Marching Method**. *Journal of Graphics Tools*. v. 9, n. 1, p. 25-36. 2003.

[25] OpenCV: Open Source Computer Vision library. Available at: <opencv.willowgarage.com>. Feb. 14, 2012.

[26] LEE, S. Y. et al. **An object inpainting algorithm for multi-view video sequences**. In: IEEE INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, 15., 2008, San Diego, CA, USA. *Proceedings…* 2008. p. 533-536.

[27] PATWARDHAN, K. A.; SAPIRO, G.; BERTALMIO, M. **Video Inpainting Under Constrained Camera Motion**. *IEEE Transactions on Image Processing*, v.16, n.2, p. 545-553, Feb. 2007.

[28] SEO, B. et al. **Projection-Based Diminished Reality System**. In: INTERNATIONAL SYMPOSIUM ON UBIQUITOUS VIRTUAL REALITY, 2008, GIST, Gwangju, South Korea. *Proceedings…* Washington, DC, USA: IEEE Computer Society, 2008. P. 25-28. doi: 10.1109/ISUVR.2008.21.

[29] HERLING, J.; BROLL, W. **Advanced self-contained object removal for realizing real-time Diminished Reality in unconstrained environments**. In: IEEE INTERNATIONAL SYMPOSIUM ON MIXED AND AUGMENTED REALITY, 9., 2010, COEX, Seoul, Korea. *Proceedings…* 2012. p. 207-212. doi: 10.1109/ISMAR.2010.5643572.

[30] JARUSIRISAWAD, S.; SAITO, H. **Diminished Reality Via Multiple Hand-Held Cameras**. In: ACM/IEEE INTERNATIONAL CONFERENCE ON DISTRIBUTED SMART CAMERAS, 1., 2007, Vienna, Austria. *Proceedings…* 2007. p. 251-258. doi: 10.1109/ICDSC.2007.4357531.

[31] LEAO, C. W. M. et al. **Geometric Modifications Applied to Real Elements in Augmented Reality**. In: SYMPOSIUM ON VIRTUAL REALITY, 13., 2011, Uberlândia, MG, Brazil. *Proceedings…* 2011. p. 96-101 doi: 10.1109/SVR.2011.29.

[32] NVIDIA GeForce Graphics Processors. Available at: <www.nvidia.com/object/geforce_family>. Feb. 14, 2012.

[33] MURTHY, G. S. et al. **Optimal loop unrolling for GPGPU programs**. In: IEEE INTERNATIONAL SYMPOSIUM ON PARALLEL & DISTRIBUTED PROCESSING, 24., 2010, Atlanta, GA, USA. *Proceedings…* 2010. p. 1-11. doi: 10.1109/IPDPS.2010.5470423.

[34] FARRUGIA, J.-P. et al. **GPUCV: A Framework for Image Processing Acceleration with Graphics Processors**. In: IEEE INTERNATIONAL CONFERENCE ON MULTIMEDIA AND EXPO, 2006, Toronto, Ontario, Canada. *Proceedings…* 2006. p. 585-588. doi: 10.1109/ICME.2006.262476.

[35] CUDA: Parallel Programing Made Easy. Available at: <www.nvidia.com/object/cuda_home_new>. Feb. 14, 2012.

[36] OpenCL - The open standard for parallel programming of heterogeneous systems. Available at: <www.khronos.org/opencl/>. Feb. 14, 2012.

[37] BUATOIS, L.; CAUMON, G.; LEVY, B. **Concurrent number cruncher: a GPU implementation of a general sparse linear solver**. *International Journal on*

Parallel Emergent Distributed Systems. v. 24, n. 3, p. 205-223, jun. 2009. doi: 10.1080/17445760802337010.

[38] SAMUEL S. S. et al. **Accelerating advanced mri reconstructions on gpus**. In: ACM CONFERENCE ON COMPUTING FRONTIERS, 5., 2008. New York, NY, USA. *Proceedings…* 2009. p. 261-272. doi: 10.1145/1366230.1366276.

[39] NVIDIA CUDA Programing guide. Available at: <http://developer.download.nvidia.com/compute/cuda/4_0/toolkit/docs/CUDA_C_Programming_Guide.pdf>. Feb. 14, 2012.

[40] LEAO, C. W. M. et al. **Melhorando o desempenho do rastreamento de pontos de interesse em imagens através do paralelismo em GPU**. In: WORKSHOP DE REALIDADE VIRTUAL E AUMENTADA, 7., 2010. São Paulo, SP, Brazil. *Proceedings…* 2010. p. 90-95.

[41] HOSOKAWA, T.; JARUSIRISAWAD, S.; SAITO, H. **Online video synthesis for removing occluding objects using multiple uncalibrated cameras via plane sweep algorithm**. In: ACM/IEEE INTERNATIONAL CONFERENCE ON DISTRIBUTED SMART CAMERAS, 3., 2009. Como, Italy. *Proceedings…* 2009. p. 1-8. doi: 10.1109/ICDSC.2009.5289380.

[42] HENDERSON, S.; FEINER, S. **Exploring the Benefits of Augmented Reality Documentation for Maintenance and Repair**. *IEEE Transactions on Visualization and Computer Graphics*, v. 17, n. 10, p. 1355-1368. Oct. 2011. doi: 10.1109/TVCG.2010.245.

[43] BIOCCA, F. et al. **Attention Funnel: Omnidirectional 3D Cursor for Mobile Augmented Reality Platforms**. In: CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 17., 2006. Montréal, Québec, Canada. *Proceedings…* 2006. p. 1115-1122.

[44] TEIXEIRA, J. M. et al.**miva: Constructing a Wearable Platform Prototype**. In: SYMPOSIUM ON VIRTUAL AND AUGMENTED REALITY, 2007. Petrópolis, SP, Brazil. *Proceedings...* 2007. p. 68-83.

[45] VASCONCELOS, L. et al. **Desenvolvimento de uma Plataforma Móvel para Inspeção Termal de Equipamentos Baseada em Realidade Aumentada**. In: ERGODESIGN USIHC, 11., 2011. Manaus, AM, Brazil. *Proceedings*… 2011.

[46] Extensible Markup Language (XML). Available at: <http://www.w3.org/XML/>. Feb. 14, 2012.

[47] NET Framework Developer Center. Available at: <msdn.microsoft.com/en-us/netframework/aa496123>. Feb. 14, 2012.

[48] WPF Diagram Designer - Part 3. Available at: <http://www.codeproject.com/Articles/23871/WPF-Diagram-Designer-Part-3>. Feb. 14, 2012.

[49] QT Creator IDE and Tools. Available at: <qt.nokia.com/products/developer-tools/>. Feb. 14, 2012.

[50] TinyXML. Available at: <http://www.grinninglizard.com/tinyxml/>. Feb. 14, 2012.

[51] ROWLEY, D. E.; RHOADES, D. G. **The Cognitive Jogthrough: A Fast-Paced User Interface Evaluation Procedure**. In: INTERNATIONAL CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 3., 1992. Monterey, CA, USA. *Proceedings...* 1992. p. 389-395.

[52] NIELSEN, J.; LORANGER, H. **Prioritizing Web Usability**. 1st edition. New Riders Press, apr.30, 2006. 432 p.

[53] LEAO, C. W. M. et al. **Modificações geométricas aplicadas a elementos reais em aplicações de RA**. In: WORKSHOP DE REALIDADE VIRTUAL E AUMENTADA, 2010, 7., São Paulo, SP, Brazil. *Proceedings...* 2010. p. 242-246.

[54] LEAO, C. W. M. et al. **Altered reality: Augmenting and diminishing reality in real time**. In: IEEE VIRTUAL REALITY CONFERENCE, 12., 2011. Singapore. *Proceedings…* 2011. p. 219-220. doi: 10.1109/VR.2011.5759477.

[55] LEAO, C. W. M. et al. **Demo — Altered reality: Augmenting and diminishing reality in real time**. In: IEEE VIRTUAL REALITY CONFERENCE, 12., 2011. Singapore. *Proceedings…* 2011. p.259-260. doi: 10.1109/VR.2011.5759497.