Computer Graphics and Computer Vision: some practical applications

Fernando Silva*, Nuno Rodrigues***, José Pereira***, Nuno Teixeira***,

Luís Almeida**, Nuno Martins* and César Páris*

***Abstract*—This paper presents three academic works, developed by three computer science students, using computer graphics, image processing and computer vision techniques. The first one is in the area of virtual environments representation and navigation, a second one in the area of cultural heritage and a third one in the area of game programming. All of them use camera model concepts to render graphics objects, to capture images or for both. Several techniques and algorithms are used, either for digital image capturing and calibration, image processing, feature detection, stereo visualization, collision detection, user interaction or for optimization issues. The results achieved so far serve as a proof of concept for the implemented prototypes. The participation of graduating students in projects for technology development has proved to increase their motivation, becoming a good practice for learning issues.**

***Index Terms* — Computer graphics, Computer vision, Image processing, Panoramic image, Virtual reality, Collision detection, Movement detection.**

I. INTRODUCTION

Personal computers have been used for several purposes, and during the most recent years have been considered by the general public as another household gadget. They are used at home for organizing accounts, letter writing, student homework and games. Many persons consider this a major revolution. From the mid-1980s there has been a public enthusiasm for virtual reality, and some people has considered this technology as another revolution that will change the way we work and communicate with other human beings [1]. Science fiction has already contributed to divulging opinions about creating and exploring synthetic 3D virtual reality worlds where it is possible to interact with anything on a virtual level. Like virtual reality, augmented reality is becoming an emerging platform, and novel approaches have enabled new application areas for museums, edutainment, research, industry and the art community [2].

In recent years there has been decisive hardware developments, mainly the graphics cards, allowing running in real time complex applications using commodity PC based systems. The computer graphics community has also been developing better, faster and more complex animation methods to create synthetic 3D virtual environments [3].

More realistic virtual environments can be created by incorporating data obtained from physical objects, using some type of sensor like a video camera. Image data need to be processed and interpreted in order to extract information to create models, what leads to the computer vision area. Computer vision techniques are mixed with computer graphics techniques to create immersive scenes. Computer vision techniques are being used in computer graphics to collect and to model graphic scenes. Computer graphics techniques are used to bound pattern recognition in computer vision [4]. Additionally, image-processing techniques are going out research laboratories to be used in the handling of photographs by personal working on other fields. Key elements for the successful merger of these techniques are the appropriate models to describe geometry, dynamics and all the attributes of objects and scenes.

3D virtual environments have been used on the promotion, management and preservation of architectural and archaeological cultural heritage [5]. Some projects have been developed in Portugal, or with participation of Portuguese partners, in this area, supported by own, national and EU funds [6][7]. Nevertheless, opportunities for new application arise as the cultural heritage specialists make new studies or work on-site, enabling the implementation of new applications for analyses, virtual representation (or reconstitution) or digital registration of their daily findings. Computer graphics applications are proven valuables supports for such activities.

In the next sections three academic works are presented, which were carried out during a seminar project, in the third year (two of them) and in the fifth year (one) of the double phased degree in computer sciences and systems at Instituto Superior de Engenharia de Coimbra: "*Stereo Visualization and Navigation", "Acquisition, Calibration and Processing of Digital images: Application for Archaeology"* and *"Playing with Virtual Objects and Real Hands"*. All of them served, by one hand, for the students to apply concepts and know-how acquired during the years of their graduation, and, on the other hand, for implementation of new solutions and optimizations that can serve as the first step for professional applications.

* Instituto Superior de Engenharia de Coimbra (ISEC), Rua Pedro Nunes, 3030-199 Coimbra; (e-mail: fsilva@isec.pt, ncmartin@isec.pt, cparis@isec.pt).
** Centro de Computação Gráfica, Campus Universitário do Isec, Rua Pedro Nunes, 3030-199 Coimbra; (e-mail: luis.almeida@ccg.pt).
*** Computer Science students at ISEC, Rua Pedro Nunes, 3030-199 Coimbra; (e-mail: {a21130094,a200505023, a21120210}@alunos.isec.pt).

II. STEREO VISUALIZATION AND NAVIGATION

Navigating in immersive virtual environments (VE) can be an alternative to the physical visit to places that are not easily accessible for some reason. Even when we are close to or inside a building, it can be interesting and useful to make a virtual visit for a quick or a more detailed observation. And this is especially true when there is the availability of a stereo visualization and an easy interaction, providing a good usability and a great sense of immersion.

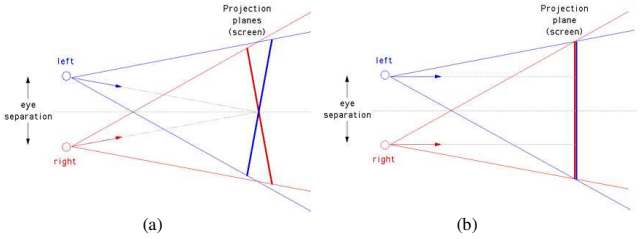
This section presents one approach to virtually visit large buildings, providing stereo visualization, navigation using keyboard, mouse and head tracker interaction, collision detection, and good performance in frames per second (fps) on commodity PC systems. The model is stored in the VRML format, and it is used the OpenGL graphic system and a free usage library (GLEW) [8] to develop the application.

A. Problem description

This work reports the development and implementation of a virtual visit to the building of Informatics and Systems Engineering Department. This is a medium size building, with three floors, three tiers each and with several rooms. A large number of rooms has several objects inside (chairs, desks, computers, etc.), and are visually communicating with the others through doors and windows. The first problem to face is to give the user a good feeling of reality and immersion. To reach this goal, it was developed a stereo solution, and included collision detection. A second problem to face is the need to provide a real time display in frames per second, in order to view the scene without bounces or delays. This leads to the need of optimizing all the viewing process, from the geometric manipulation to the rendering process. In the remainder of this section, some background about the stereo projection used is briefly exposed, two approaches for collision detection when navigating are presented, the clipping process in the 3D stage is explained, some questions about solutions for illumination are discussed, the optimization in the rendering process and the way the user can interact to navigate are described.

B. The stereo viewing

The stereo viewing is formed by simulating the sense of depth in the image projected onto a surface, the projection plane. Several techniques of stereo viewing are based on the same principle, which consists on the utilization of two projections of the same scene from two centers of projection slightly displaced horizontally. Therefore, the two images created are overlapped continuously or switched alternatively, according to the technique used. At the end, if the user sees the left image with the left eye and the right image with the right eye, his brain will process the images as if there were objects closer and objects more distant from the user, creating an illusion of depth.



(a) (b)

Fig. 1 – Stereo viewing, using (a) toe-in and (b) off-axis projection.

It is usual to point out two methods to make stereo views: *toe-in* and *off-axis* projections [9]. T*oe-in* projection uses a symmetrical visualization volume for left and right projections, as shown in Fig 1(a), but creates depth visualization inaccuracies. The *off-axis* projection creates artifact free images, unlike the *toe-in* projection. O*ff-axis* projection was used, which constrains to compute an asymmetrical visualization volume for left and right projections, as shown in Fig 1(b). In this implementation, passive stereo or active stereo viewing can be displayed, to be seen with two-color glasses or with shutter glasses).

C. The collision detection

The sense of immersion can be highly compromised if the user is allowed to walk through walls, closed doors and other real-life impassable obstacles. In order to provide a reasonably realistic experience, that kind of behavior must be prevented, and the paths that the virtual visitor is allowed to tread should be the same as if he was in the real site. At his simpler form, a virtual visitor or observer is considered as a 3D point, corresponding to the position of the virtual camera. The first step is conferring this entity some kind of volume, so the fact that the observer actually occupies space can be simulated. In order to keep performance in mind, the actual tree-dimensional volume to use must be one that, while providing a reasonable approximation to the actual space occupied by a real person (the observer), is still sufficiently computational inexpensive to work with, so it won't compromise those performance requirements.

The first approach to define this volume was an Axis Aligned Bounding Box (AABB), mostly because of its inherent simplicity [10]. As the box is always aligned with the three-dimensional world axis, most of the computations reduce to one-axis problems. This approach turned to be somewhat poor in terms of realism, as the approximation to a real person volume is clearly too rough to meet the realism criteria it is supposed to achieve. This was especially notorious when the user tries to pass through tight places, such as doorways. As a result, it was tried a second approach, using a capsule as a bounding volume for the observer. This time, the behavior of the program was much better, and the performance hit wasn't noticeable, as the calculations for intersections between the capsule and the virtual world polygons can take advantage of the fact that a capsule can be seen as a "line with thickness", and, as such, reasonably simple algorithms can be used for subsequent calculations.

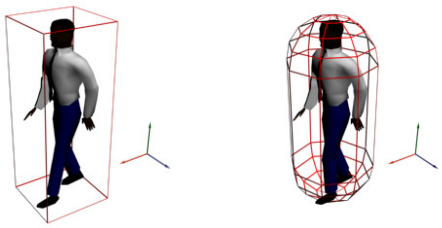


Fig. 2 – AABB and capsule.

Once the bounding volume is defined, the matter of preventing the observer from behaving "ghostly", passing though walls and other kind of bizarre actions, reduces to testing for collision between the bounding volume and the virtual world underlying geometry, before updating the camera position. At the lowest level, this is achieved through triangle-capsule intersection tests.

Of course, in a model containing several hundred thousand polygons, testing each one for intersection with the bounding capsule does have a serious performance hit, too serious to permit real-time rendering. As such, some kind of technique must be used to decrease the number of intersection tests needed. The approach followed in this project is founded in the use of *octrees* as a method for tree-dimensional space subdivision. Using an hierarchical data structure, like *octrees*, to manage the geometry allows the early discarding of a great number of polygons, namely those that are too far away from the observer to be an obstacle, and, as a result, permits that the number of polygons to be tested against the capsule can be maintained at a reasonable level, and cope with the performance required for real-time rendering and interactivity.

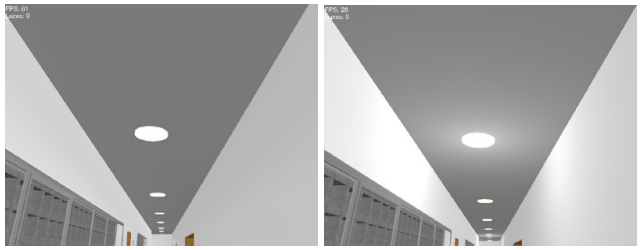
D. The 3D clipping solution

If the geometric model of the building is composed by about three hundred thousand triangles, it is worth searching out if there is a way to limit the number of triangles to render simultaneously in the visualization of each frame. Two decisions were made: to simplify the complexity of the model, and to limit the amount of geometric objects to display in each moment. The first one was achieved by eliminating excessively detailed description of the model. The second was achieved in the modeling of the geometry, by dividing the space and arranging it in *sections* and *portals*. At any moment, only the section that the virtual camera can see must be rendered, unless the existence of a *portal* belonging to that section allows the virtual camera seeing another section through it. The concept of *portal* arises as a consequence of existing doors and windows, which behave as connections between sections. This concept, already described in literature like in [11], was implemented.

E. Illumination

The basic illumination technique used by OpenGL is the so-called Gouraud shading, which uses the Phong illumination model [12]. But with Gouraud shading, the Phong illumination equation is used to compute the color only in the vertices, and the remaining colors are computed by bilinear interpolation of the bounding vertices colors. This creates usually a problem, mainly when the geometry is described by large polygons, as is the present case. For example, if a light source is near the center of a large polygon, the color at this point is computed by interpolation of weak light intensities in the vertices of the polygon that are far away from the center, resulting in a lower color value and in a lack of realism. Moreover, OpenGL provides only eight light sources, what is not so much for this case. One solution to these problems is to use a multipass rendering: the scene is rendered several times in an additive blending way, and eight new light sources are used in each pass.

A better solution is to use the *Graphics Processing Unit (GPU)*, taking advantage of his special capability to compute graphics objects. Programming vertex shaders and fragment shaders has proved to result in a more realistic image, as can be seen in fig. 3.



(a) (b)

Fig. 3 – Illumination, using (a) Gouraud shading and (b) fragment shading.

The performance in frames per second slows down with the utilization of fragment shaders. However, optimizing the code and the program organization, it is possible to maintain a frame rate high enough to offer a smooth visualization.

F. Rendering the model

To display the model with a good performance, in addition to the early 3D clipping solution mentioned above, it was necessary to employ some organization and programming skills. The first one was to organize objects with the same attributes (as material, texture, blending) in order to maintain as much as possible the internal OpenGL state while rendering the sections for each frame, thus decreasing the number of changes of internal OpenGL state. It was implemented the concept of *FaceBuffer*, a C++ class that encapsulates this behavior. It was verified a clear gain in performance.

Fig. 4 – Visualization of a scene with several portals using FaceBuffers

The next improvement was to use *Vertex Arrays*, a mechanism used in OpenGL that consists of assembling related entities (vertices, normal vectos, texture coordinates) in memory blocks that can be transferred together to the rendering process. This gives a good increase in performance. However, this method can be superseded by the mechanism of using *Vertex Buffer Objects* (VBO), in which the same data as used in *Vertex Arrays* is now stored in the GPU's internal memory, avoiding data transfer from RAM to the graphic card VRAM.

The visualization of the model of a building can profit from the fact that the geometry doesn't change. In that case, it is strongly advisable to use *Display Lists*, in which the geometry is stored as a group of OpenGL commands for redrawing the same geometry multiple times.

Fig. 5 – Visualization of the scene with VBOs and display lists

The use of several techniques in sequence allowed the increase in performance from some frames per second to over 150 fps, which can be considered as a dramatic optimization.

G. User interaction

Navigation is achieved by using mouse, keyboard and head tracker interaction. The virtual camera position and orientation is controlled by processing mouse movements and the keyboard arrow keys, and complemented with the tracker interaction. It was used an infrared tracker, the *TrackIR™* from *NaturalPoint®*, that has 6 degrees of freedom (DOF), 3 for *Yaw*, *Pitch* and *Roll* rotations and another 3 for 3D position (*x, y, z*) [13].

H. Results and discussion

The results are plainly good. The visualization is made without processing delays, and the stereo viewing and tracker interaction provide a deep feeling of immersion. The usage of a professional graphic card and an active stereo projector will provide an impressive viewing experience.

III. Acquisition, Calibration and Processing of Digital Images: Application for Archaeology

Digital image analysis and processing has been a computer graphics research and development (R&D) topic for decades [14][15]. Nevertheless, the state of the art in terms of techniques and algorithms in this area continue to be applicable. Continuous developments and optimizations for new applications, like for example those applicable for cultural heritage, are actually running allover the world. Some activities for cultural heritage studies and preservation imply the analysis and registration of information for post-study or dissemination among specialists or public in general. That is the case of archaeology, in particular the study of mosaics used centuries ago for houses decoration. Part of the study consists on registering the contours of the tesseras forming the mosaic, as accurately as possible, by drawing it on plastic sheets or transparent paper, by hand. This is done on-site, covering the mosaic panels with the sheets and drawing the tesseras, one by one, using a pencil. This work is as important for archaeologists as it is time consuming and uncomfortable, besides depending on climate conditions and human capability for drawing accurately. Despite the existence of commercial solutions for contour and feature extraction on digital images [16], they neither provide completely the functionalities nor fulfill the needs for all kind of professionals. This section presents one approach, and first results, for application of existing computer graphics algorithms and techniques for automation of tessera contours extraction from digital images of mosaics. Typically a mosaic panel cannot be captured at once in a single digital image. Several images are acquired and stitched, originating one final panoramic digital image. Images acquisition is the first step of a sequential process, starting with camera calibration, images stitching, edge contour detection and export of information for further usage.

A. Camera calibration

The process of tessera contour calculation starts by mosaic images acquisition. It can be done using a normal photo camera, whose intrinsic and extrinsic parameters are relevant for inference of relation between points in 2D images and the corresponding points in 3D world. The knowledge of this correspondence permits the establishment of metric associated to the 2D image, and therefore the scale generation and accurate determination of objects dimensions. The result of the calibration process is the c*alibration matrix*, obtained from the intrinsic camera parameters (focal length, optical center, skew, etc.) and *pose matrix*, obtained from the extrinsic parameters (information about camera position and orientation in 3D world) [17][18]. The extrinsic parameters of the camera are ignored, in this particular application, as the metric relation only was needed.

In this prototype we used a calibration toolbox for Matlab®. For camera calibration, a set of images from a pattern with known metric is required, typically a chessboard. They are captured with the same focal length.

1) Calibration matrix

The calibration matrix, C, given by

$$C = \begin{bmatrix} fk_x & \gamma & C_x \\ 0 & fk_x & C_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (1)$$

is one of the calibration results. It contains the intrinsic parameters for a given camera, like the focal length (f), the metric relation between image and 3D world, in each image axis, (k_x and k_y), the projection of the optical center in the image (c_x, c_y) and the skew (γ). It is an upper triangular matrix.

2) Results validation

Calibration results validation is important for avoidance of posterior errors. The validation consists on comparing the projected coordinates of some 3D points with its previously known 2D positions in the image. For this to result, the camera cannot be affected by any 3D transformation (translation or rotation). Therefore, the matrix of extrinsic parameters, that multiplies the calibration matrix, is a diagonal matrix with unitary entries. The distance from the camera to the captured object (chess board) must be known, and the focal distance must be the same as used for calibration images capture.

3) Metric calculation.

The process of associating a metric to an image is the same as giving to a map a scale. In this case, the scale is between the 2D dimensions (pixels) and the 3D world units. The value corresponds to the k_x and k_y, for each image axis. These parameters are elements of (1). For simplification, the same value is often assumed for k_x and k_y (the image pixels are considered squared).

Using fk_x or fk_y from the calibration matrix, and knowing the focal length, f, one can obtain the metric relation by dividing those values.

B. Computing images correspondence

Adjacent images obtained by the camera must be taken with some percentage of overlap. It is of fundamental importance for identification of common areas between images and establishes the correct orientation for both, when stitching for final panoramic image generation.

After images acquisition, covering the total mosaic area, correspondence points must be determined between pairs of images, so that the correspondence matrix, named homography, can be determined, enabling their accurate merge (stitch). The first approach is to request the user to manually indicate correspondent points in both images. Nevertheless, typically, several images are used for representation of a single mosaic panel, what implies the implementation and usage of algorithms for automatic estimation of corresponding points among the images. One possible process is to detect corners using Harris algorithm [19], then select the corresponding points using "monogenic phase" algorithm [20], and identify the points that best fit the matching between images, using the RANSAC algorithm [21].

1) The Harris detector

The Harris corner detector was applied for estimation of points in the image that can potentially be correspondent with others in the image to be stitched, as presented in fig. 6. In terms of concept, the corners are information considered stable on a sequence of images. Therefore, its usage for image correspondence estimation, pattern matching and other operations over digital images, is very useful and reliable.

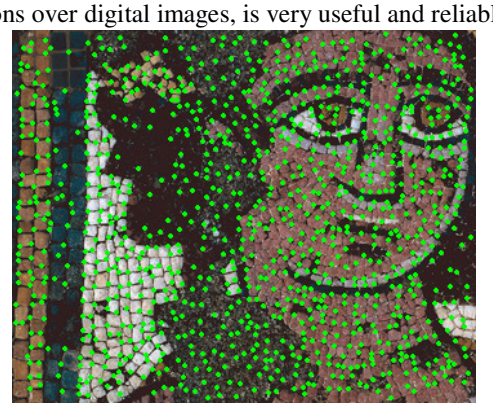


Fig. 6 – Example of points detected with Harris corner detector

2) Choice of points by "monogenic phase"

The identification of common points in two images can also be implemented on the basis of the "monogenic phase" theory. It stats that one signal has very specific characteristics, permitting the evaluation of analogies for a sequence of signals [20]. For images it permits the evaluation of correspondence between two of them, evaluating, for each pixel on the image considered as the base image, the best correspondence on the other image. For evaluation it uses the difference of local phase for the pixel under test (see fig. 7). For large images, this calculation is time consuming. One solution is to apply this methodology over the points previously calculated with Harris detector.

Fig. 7 – Example of points detected with "Monogenic phase".

3) Correspondent points estimation with RANSAC algorithm

From the "Monogenic phase" a number of incorrect points can result. A more precise filter must be applied, reducing as much as possible the number of incorrect correspondences. The RANSAC algorithm is used generically for estimation of robust models, in terms of toleration or elimination of erroneous information: in the present case, points not correct in terms of correspondence between images.

The RANSAC was applied over the set of points obtained after the two previous phases obtaining a new set of points with smaller percentage of incorrect matches (fig. 8).

Fig. 8 – Points estimated with RANSAC algorithm.

C. Panoramic images construction

Panoramic images are the result of a constructive process, consisting on stitching several images, taking one as the base [22]. Pairs of images to be stitched must have a percentage of common area, so that the corresponding homography can be calculated, enabling the planar geometric transformation for making the axis on both to coincide, for correct alignment. After the planar transformation, the images are ready to be stitched (fig. 9).

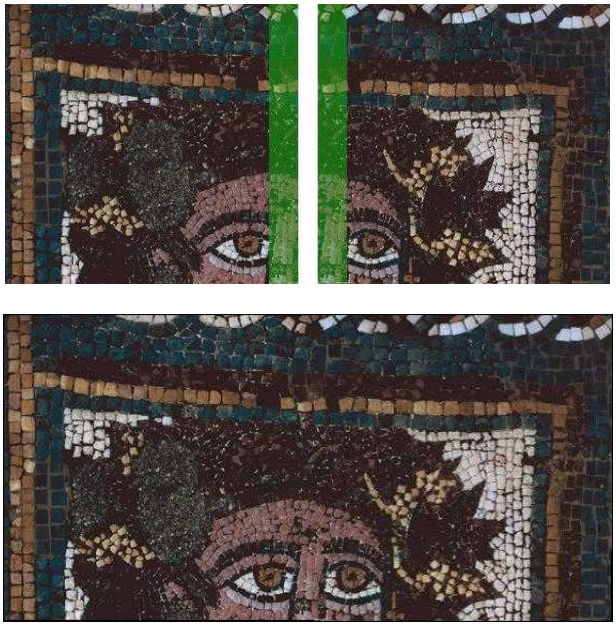


Fig. 9 – Example of panoramic image from two originals.

1) The transformation matrix

A homographic matrix (H) can represent the geometric transformation between two images. In the context of this work just a particular type of 2D planar transformations will be considered, affine transformations [23].

So, considering two images I_1 and I_2 and two points $P_1(x,y) \in I_1$ and $P_2(x',y') \in I_2$, the transformation matrix relating P_1 and P_2 is as follows (k is the homogeneous coordinate):

$$kP_2 = HP_1 \Leftrightarrow k\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (2)$$

Knowing correspondent points in each image, the estimation of H is the estimation of six variables, because the homography is defined up a scale factor. Therefore, it is necessary, at least, three points for the matrix determination. The user can input the points manually, but for panoramic images created from several partial images, that process is not affordable. The best solution resides on automatically selecting points determined by the algorithms referred in section B. From the transformation, if wrong points are used, some "ghost effects" can result on final stitched image. One hypothesis for transformation matrix optimization is to use the Levenberg-Marquart algorithm [24], for minimization of quadratic error of intensity between the two images:

$$E = \min \sum \left[I'(x',y') - I(x,y) \right]^2 \quad (3)$$

2) Merging the images

After determination of transformation matrix, the objective is to generate the panoramic image. The union of images is always done between two that are overlapping each other in at least 5% (minimum considered for algorithms performance), being one the base and the other the image to be "merged". The first step is to transform the coordinate's axis of the image to be "merged" in a way to make it coincident with the axis of the basis image, using the homography between these two images.

After that the junction is done, by simply overlapping the image over the base image at the right position. The panoramic image can present some "mottling" effect, or other effects, at the junction edge region. In order to avoid such effects, the merging of images is based on the weight of the pixels for final image (see fig. 8).

D. Geometry extraction

Mosaic geometry extraction consists of image interpretation for extracting the geometry for each of its constituting elements (tesseras). It is the same to say, tesseras contour detection and its transformation in lines that can be stored as vectors (2D or 3D). The main steps for this purpose are shown as follows.

1) Edge (contour) detection

Contours are limits or borders of regions inside the image. They are identified on images as thin areas where a remarkable contrast of intensity exists among adjacent pixels. In mathematical terms, a contour is detected when exists a variation of light intensity exceeding some established limit [25]. Among the several algorithms for image contours detection, the choice for this work was the Canny detector, which works like this:

a) Eliminate possible noise on image using a Gaussian filter;

b) Calculate horizontal (G_x) and vertical (G_y) gradient for each pixel. The gradient indicates the probability for each pixel to make part of a contour;
c) Estimate the pixel orientation relative to the contour edge. The orientation is represented by the angle resulting from

$$\theta = tg^{-1}(Gy/Gx) \quad (3)$$

d) Pixels not considered as contours must be eliminated (there are isolated pixels not integrating any contour);
e) Eliminate the streaking effect, which happens when a contour line is not continuous.

The application of this tool can be seen in fig. 10 and fig. 11.

Fig. 10 – Exemplification of edge detection using Canny detector

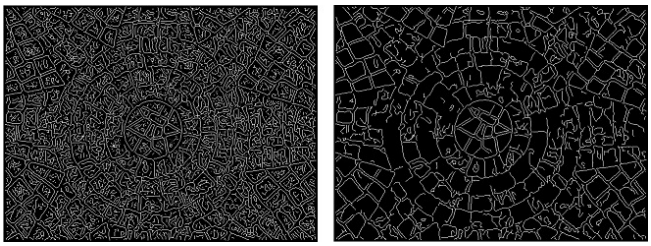


Fig. 11 – Example of application of filter for streaking elimination

2) Generating contour lines

The information resulting from edge contour detection is essentially a huge set of points. One possibility for making this information useful for 3D geometric mosaic representation and manipulation is to transform the 2D points into 3D lines. Using the points calculated for each contour and joining them in a single 3D polyline can do that. The process has two main steps, and corresponding algorithms, as proposed by Peter Kovesi [26]:

a) Create lists of points, each one corresponding to one contour in the image. If one point doesn't have another adjacent point, then a new list is created.
b) After lists creation, another algorithm will try to join the maximum of points with one straight line, using the test criteria of deviation of points from the optimal line. Several lines can result that are part of the same contour. A post-processing must be implemented for joining the straight lines with adjacent end-points.

3) Geometry information conversion

For 3D geometric representation and manipulation of the extracted contour edges, an appropriated format must be adopted for information export. The chosen format is the VRML, which is suitable for organizing and grouping sets of 3D elements for a virtual world representation. It is also a standard format implemented in the majority of available 3D software, thus enabling to import 3D edge contours for further utilization in other systems. The conversion of contour edge information for 3D format is important for further manipulation, like 3D reconstruction of mosaics from photos, archaeological studies and publication on Internet.

In the context of this work, the information was converted for VRML using the nodes *IndexedLineSet*, *coord* and *coordIndex.*

The conversion to VRML of the resulting mosaic is presented in fig. 12.

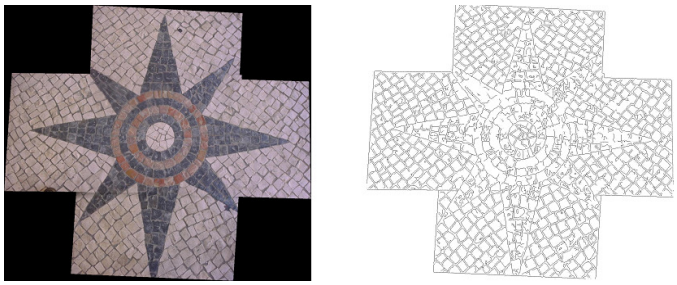


Fig. 12 – Example of a contour extraction from panoramic image

E. Comments and Future Work

The prototype implemented and tested proved this solution to be a useful advance for archaeological information registration, particularly concerning the mosaics. Some artifacts or incorrect contours can result from the automatic extraction of mosaic tesseras geometry, which can be adjusted or minimized by manually editing the final 3D information. Nevertheless, the gain of time and human resources effort for mosaic drawing is a remarkable result. The prototype proved the possibility of registering mosaics information, with high accuracy, without human intervention other than images capture and software functionalities operation.

Algorithms optimization is mandatory for better performance and reduction of processing time. Optimization can also derive from algorithms implementation using a fast compiled programming language, like C++, instead of using Matlab®. Specific hardware must be designed and constructed for camera installation during images acquisition, avoiding potential errors from "bad" images.

Furthermore, the prototype can be extended for application in other contexts of cultural heritage information registration and study, like for example walls or columns in ruins.

IV. Playing with Virtual Objects and Real Hands

A. Introduction

The merging of real world and virtual worlds to produce a new environment where physical and virtual objects can co-exist and interact is called Mixed Reality, which can be better described as a mix of augmented reality and virtual reality [2]. Mixed Reality is a topic of research that registered a notorious and growing development in recent years and is present in a number of areas such as medicine, architecture, robotics, industry and tourism. As a consequence of its constant growth

innovative applications were implemented that are changing the way of perceiving the use of virtual environments mixed with real environments to simulate and to optimize activities that happen in a real world.

To enter this mixed world, the end-user can make use of several hardware devices. Some common devices are video cameras, projectors, head mounted displays and data gloves. These devices create communication channels, allowing the user to generate events and to send messages, and/or receive information generated by the system as a response of its actions. This new way of communicating is generating new human-computer interaction paradigms.

The remainder of this section exposes an academic work carried out during a seminar project. The objective of the seminar was to create a set of simple games through the use of a mixed reality environment. The interaction uses a conventional web cam to capture user movements, and uses as a reference, the same ideas seen in SONY PLAYSTATION® famous game – Eye Toy® [27].

B. Objective

The objective of the project was to create a virtual environment and capture the user interaction using cameras, in order to play games. The main tasks to solve are:

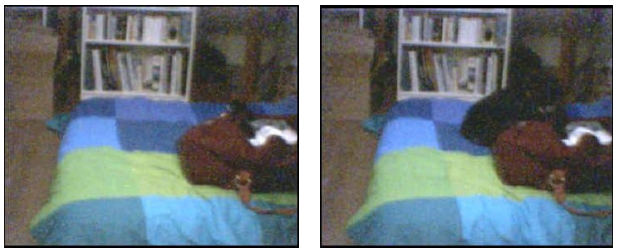
1) Video capture through a small USB camera located in front of the user, to acquire images in real time;

2) Movement detection, to determine where the movement is, using sequences of images captured by the camera;

3) Virtual environment creation, by drawing 3D graphic scenes, where virtual objects move in a consistent or random way;

4) User interaction, to relate the detected movement with actual actions in the virtual scene.

On its final configuration, the project resulted in the development of a group of games for one player, which can be easily extended for two players.



(a) (b)

Fig. 13 - Captured images: (a) before movement and (b) after the movement.

C. Implementation

The project was developed in Visual Studio.Net 2003, using C/C++ language and the OpenGL graphic system.

1) Video capture

The video capture was developed based on the "*Video for Windows*" library. This library contains a set of functions that makes easily feasible the acquisition of real time images.

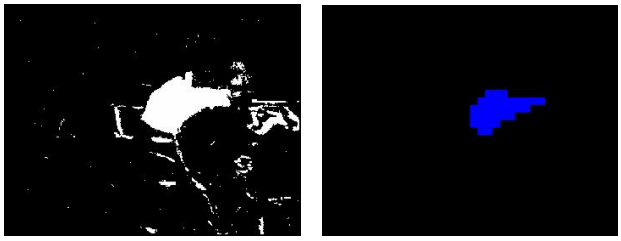
In order to get this, it is needful to establish a connection between the USB port where the camera is connected, and the program memory where the images will be placed. In the setting up of this connection, an acquisition rate must be established and the video source and the size of the images must be defined.

Upon these initializations, the images become available to be handled and to detect the user's movements.

2) Movement detection

The explanation of how to detect the movement will be presented now, using a simple example. Imagine that the camera is capturing the scene shown in Fig. 13(a). Meanwhile, an object appeared on the bed, as can be seen in Fig. 13(b).

The coming up of an object into the scenery will be detected as a movement. In order to get it, the difference between the color values of the monochromatic images in Fig. 13(a) and Fig. 13(b) is computed. This difference, marked with the white color in Fig. 14(a), defines the changes between the images, or in other words, defines the movement occurred.



(a) (b)

Fig. 14 - Processed images: (a) when applying the subtraction to the images in figure 13, (b) when applying the noise removal process to image (a).

Then, a noise removal is applied, eliminating the small no significant areas and considering them as no movements. In order to get this, a run through the image that contains the differences is made, until finding a white pixel (candidate to movement). By then, a search is made to see if nearby there are a meaningful number (defined by the programmer) of white pixels that can be considered as movement. If so, all of those pixels are marked as movement. If not, all of them are marked as without movement (as black pixels). The application of this step to the image of Fig. 14(a) results in the image of Fig. 14(b).

At this point, the localization of the movement is mapped into coordinates of the virtual scene, applying a relation defined when the size of the application window is set.

3) Virtual environment creation

The OpenGL library functions were used for creation of a virtual environment. The games developed are simple as well as their scenes, which are drawn using simple geometric forms such as spheres, plans and cubes. This optimizes the time needed for rendering the virtual scene, freeing more time for the "critical" processing, the movement and collision detection.

4) User interaction

To interact with the virtual world of the developed games, choosing an option or hitting a ball, the player must use his hands. The collision detection is calculated from the information of user's movement and the location of the animated synthetic objects. This is the mix of virtual and real worlds. The collisions are used to alter the movement of the virtual objects according to the angles of collision with the hands.

In one of the games, a bouncing ball runs inside a virtual room around a stationary physical object placed within the angle of view and captured by the camera.

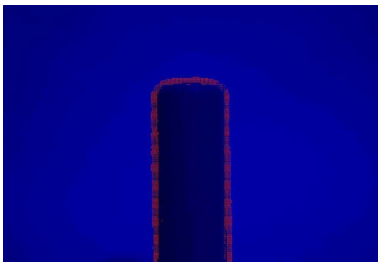


Fig. 15 - 2D profile of an object computed from an image.

After detecting the object 2D profile, a synthetic revolution volume is created, based on the acquired profile, and the virtual ball bounces continuously colliding with the room walls and the virtual object created from the real object captured by the camera.

V. CONCLUSION

In the previous sections three examples of application of computer graphics, image processing and computer vision algorithms and techniques were presented, in a practical and useful way. The work presented resulted in three independent prototypes that can be demonstrated and used for real utilization by users in general, which is guaranteed by the tests and achieved results. Despite the implementation work has been done by graduating students, the final solutions present high level of quality, fulfilling the proposed objectives.

As a final note one can conclude about the importance of this kind of projects for concepts consolidation and students' preparation for immediate work on software houses, R&DT institutions or even for continuation of their academic studies: the work of research, algorithms and software conception and implementation is of primordial importance for their training for professional career.

ACKNOWLEDGEMENTS

The authors wish to thank *Laboratório de Investigação e de Inovação Tecnológica* for providing some resources to develop the presented work.

For the cultural heritage project, an acknowledge is due to *Centro de Computação Gráfica,* for technical accompaniment and material support for the development, and *to Conimbriga Monographic Museum*, for initial specification and provision of mosaics images for tests.

REFERENCES

[1] J. Vince; "Virtual Reality Systems"; Addison-Wesley; 1995.

[2] O. Bimber, R. Raskar, "Spatial Augmented Reality: Merging Real and Virtual Worlds", A K PETERS, Ltd, 2005

[3] A. Leonardis, F. Solina and R.Bajcsy, "Confluence of Computer Vision and Computer Graphics", NATO Science Series, 3. High Technology – Vol.84, Kluwer Academic Publishers, 2000.

[4] A. Watt, F. Policarpo, "The Computer Image", Addison Wesley, 1998.

[5] A. Marcos, P. Bernardes and V. Sá; "Multimedia Technology and 3D Environments Used in the Preservation and Dissemination of Portuguese Cultural Heritage" in: Educational Technology, Méndez Vilas A., Gonzáles J.A., Zaldívar I.S., (Eds), Junta da Estremadura, ISBN Colección-84-95251-76-0, Tomo--84-95251-79-5, pags. 1335-1339 (vol III).

[6] M. Martins, P. Bernardes; "A Multi-Disciplinary Approach for Research and Presentation of Bracara Augusta's Archaeological Heritage" in: Archeologia e Calcolatori, vol. 11, pp. 347–357, 2000.

[7] V. Vlahakis, L. Almeida, et al, "Challenges and Solutions of a Personalized Augmented Reality Guide for Archaeological sites", "Computer Graphics in Art, History and Archaeology" Special Issue of the IEEE Computer Graphics and Applications Magazine, Sept.-Oct. 2002.

[8] http://glew.sourceforge.net/.

[9] http://local.wasp.uwa.edu.au/~pbourke/stereographics/stereogl/

[10] D. Voorhies; "Triangle-Cube Intersection"; Graphics Gems III, edited by David Kirk, Academic Press, 1992, p. 236 - 239.

[11] R. J. M. Silva, G. N. Wagner, A. B. Raposo, M. Gattass; "Experiência de Portais em Ambientes Arquitetônicos Virtuais"; VI Symposium on Virtual Reality (SVR 2003), p. 117 a 128.

[12] Woo, Neider, Davis, OpenGL Programming Guide, Addison Wesley Developers Press, 1997.

[13] OptiTrack API, OptiTrack SDK, at http://www.naturalpoint.com

[14] R. C. Gonzalez, R. E. Woods, "Digital Image Processing", Addison-Wesley Pub (Sd); 3rd edition, 1992.

[15] J. R. Parker, "Algorithms for Image Processing and Computer Vision", John Wiley & Sons Ed., 1997.

[16] SIMAGIS® - http://www.smartimtech.com/

[17] R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision", Cambridge University Press, 2000, pp. 138-183

[18] R. Tsai, "A Versatile Camera Calibration Technique for 3D Machine Vision", IEEE J. Robotics & Automation, RA-3, No. 4, August 1987, pp.323-344

[19] K. G. Derpanis.: http://www.cse.yorku.ca/~kosta/CompVis_Notes/harris_detector.pdf

[20] M. Felsberg, "Disparity from Monogenic Phase", Proceedings of the 24th DAGM Symposium on Pattern Recognition, Springer-Verlag, 2000, pp. 248 - 256

[21] M. Fischler and R. Bolles, "Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography", Communications of the ACM, 24(6):381–395, 1981.

[22] R. Szeliski, Heung-Yeung. "Panoramic Image Mosaics", Microsoft Research.

[23] R. Szeliski. "Video Mosaics for Virtual Environments", IEEE Computer Graphics Applications, Vol. 16, No, 2: March 1996, pp. 22-30

[24] S. Roweis, "Levenberg-Marquardt Optimization", at: http://www.cs.toronto.edu/~roweis/notes.html.

[25] J. F. Canny, "A computational approach to edge detection", IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6): 112–131, 1986.

[26] P. D. Kovesi. "MATLAB and Octave Functions for Computer Vision and Image Processing", School of Computer Science & Software Engineering, The University of Western Australia. Available from: http://www.csse.uwa.edu.au/~pk/research/matlabfns/

[27] http://www.us.playstation.com/Content/OGS/SCUS-97319/Site/