Real time projection of video images in virtual scenarios

Rui Gonçalves
FEUP LSTS
Rua Dr. Roberto Frias
4200-465 Porto
Portugal
Phone +351 22 508 1539
Fax +351 22 508 1443
E-Mail rjpg@fe.up.pt

A. Augusto de Sousa
FEUP/INESC Porto
Rua Dr. Roberto Frias
4200-465 Porto
Portugal
Phone +351 22 508 1827
Fax +351 22 508 1443
E-Mail aas@fe.up.pt

Gil Gonçalves
FEUP LSTS
Rua Dr. Roberto Frias
4200-465 Porto
Portugal
Phone +351 22 508 1690
Fax +351 22 508 1443
E-Mail Gil@fe.up.pt

João Sousa
FEUP LSTS
Rua Dr. Roberto Frias
4200-465 Porto
Portugal
Phone +351 22 508 1690
Fax +351 22 508 1443
E-Mail jtasso@fe.up.pt

Paulo Dias
FEUP LSTS
Rua Dr. Roberto Frias
4200-465 Porto
Portugal
Phone +351 22 508 1539
Fax +351 22 508 1443
E-Mail pdias@fe.up.pt

Abstract

There are many applications that map the reality onto virtual environments, through the projection of real images. One of the most popular examples is Google Earth which is applied to geography and other examples include Nasa World Wind, and 3D scanners applications. Implicit in these methods is a perfect alignment in the camera positioning between the real and the virtual worlds. This paper presents a generic algorithm to be applied in the context of robotics, more precisely in underwater vehicles (ROVs-Remote Operated Vehicles) equipped with video cameras and used for inspections of underwater infrastructures. The method uses the images and position data of the vehicle to map the virtual objects representing the infrastructures. We are also planning to use this technology, in unmanned aerial vehicles (UAVs) to produce terrain mapping in a personalized way for a limited area.

1. Introduction

The projection method we present consists on identify which virtual objects are in the depth of view of a camera and map them with the respective projected image in an adequate way. There is a source (or focus) from where the image is propagated until it intercepts the objects. When not done in real time, it is usual to see this functionality implemented with the *renderer* technique *ray-tracing* (e.g. in 3D-Studio, Lightwave, POVRay, etc.). In *ray-tracing* the projection of images drifts naturally, once this method of rendering seats intensively on calculations of geometric intersections between light rays and objects. The problem is the significant difficulty to apply this technique in real time, since it is very CPU consuming.

In real time applications, there has been a trend to simulate the projection effect. Some techniques are implemented using overlapping of textures (multi-texturing) by hardware [UENG] which normally have a limited texture layers and causes an implicit constrain to the number of projections.

We are applying a method where the projected image is able to remain correctly placed and deformed without great computational burden. This method adds a new 3D object (polygons) to the scene being similar to the stencil shadows technique used to project shadows by means of „shadow polygons“. The projection creates a new grid object that adapts near to the surface of the existing objects.
The grid projection object has a settable resolution for different accuracy in the adaptation. The grid segmentation also addresses the issue of perspective correction, as will be explained. This solution may not work well if the objects that are receiving the projection are too complex or with discontinuous surfaces. We intend to use this functionality in simple objects representing walls, pillars and terrain superficies. The framework receiving this functionality is being developed (in Java/Java3D) to work in any common PC at real time mission processing.

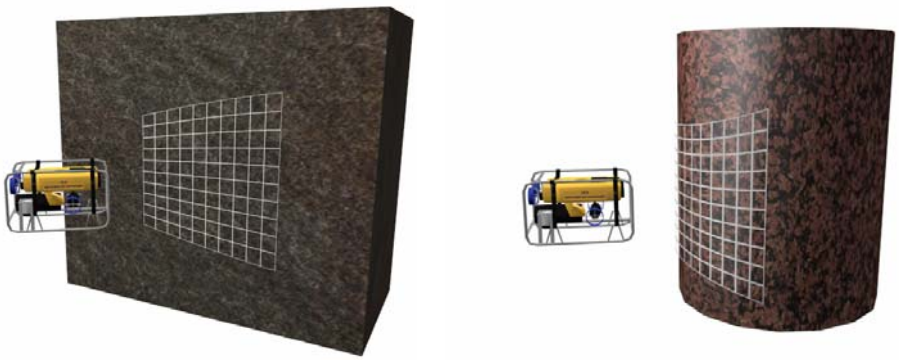

Figure 1: The grid projection object generated using the ROV position data sensors

2. Project Main Goals

The algorithm has applicability in the framework Neptus [NEPTUS06; NHOME] which main function is to support the coordinated operation of heterogeneous teams, including autonomous and remotely operated underwater, surface, land, and air vehicles and people. This framework is being developed at the Decision and Control Group Engineering (DCEG) of the Department of Electrotecnic Engineering and Computers (DEEC) in the Engineering Faculty of Porto University. The work is fit in the activities at one of the laboratories, the Underwater Laboratory of Systems and Technology (USTS).

This framework (Neptus) has one package of components for three-dimensional visualization. For each mission to be performed it is always constructed a 3D virtual representation of the mission site in question. To construct the 3D virtual environment it is used the Mission Planner application (Figure 2) specific editor, derived from the Neptus framework. Neptus uses a XML format to define geo-referenced maps. Besides the basic geometrics types used by this editor (e.g. parallelepipeds, cylinders, ellipsoids, planes, terrain objects generation …), 3DS, VRLM, X3D and Java3D file formats can be placed inside Neptus maps. Objects can be covered with generic textures (metal, rock…) normally representing structures to inspect that are visually unknown.

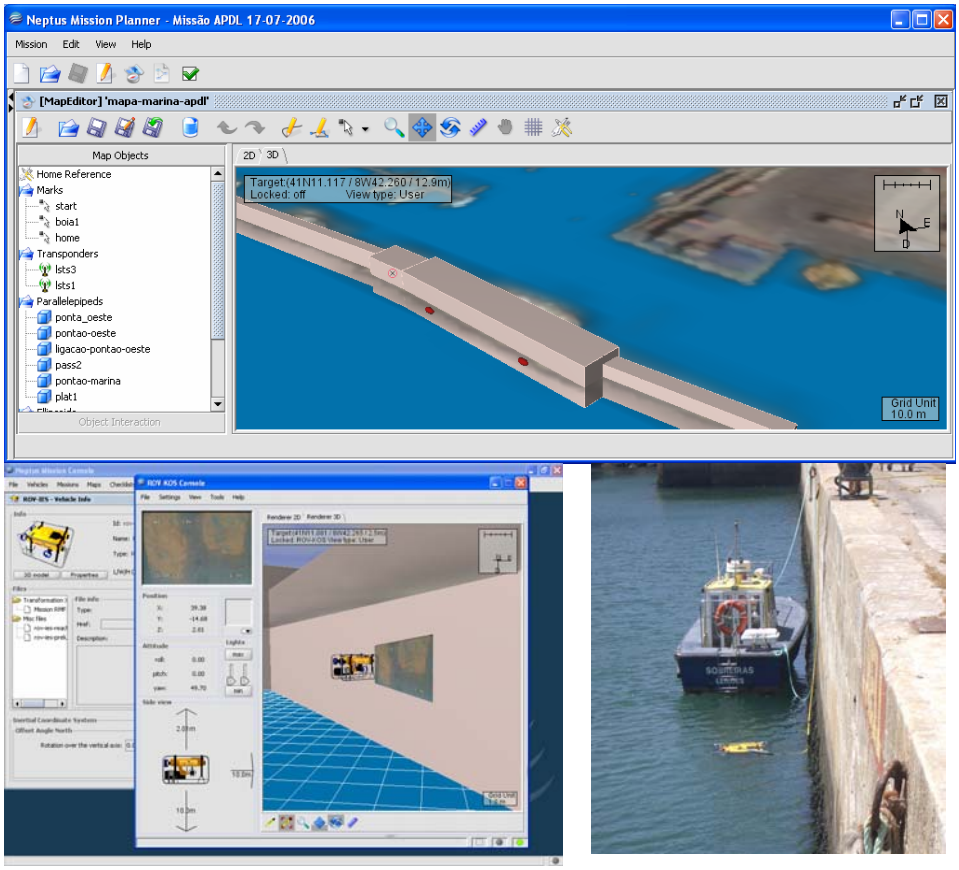

Figure 2: Map Editor at top, ROV console at bottom left and ROV real image

Once a generic 3D map of the area is constructed, vehicles equipped with cameras to verify the structures are used during the mission. For the area corresponding to figure 2 “Porto de Leixões”, it was used a ROV type vehicles and video images of the submerged walls have been collected. Later, a revision of video data is made to find any problem in the inspected structure, through a less practical search, in the recorded film. Comparing the film time with the ROV positioning logs over the time it is known an approximate localization of any defect found [MRA06].

The goal of this work consists in defining a way to map the video image in the 3D environment, making it possible to map virtual walls and others structures with textures obtained from the caught image of video. This way it is possible to see the real wall image in the 3D environment with manipulation in real time instead of searching in the video stream.

3. System Description

the Neptus framework has a sub application to mount monitoring/control consoles by placing and configuring Real Time information panels (compass panel, motors panel, sensor panels, video panel, Renderer3D panel …).This panels can be developed and added to Neptus framework as plug-ins.

For the video projection functionality it is intended that the users have a simple interface in the Renderer3D panel to associate one video panel, placed in the console, to one vehicle in the Renderer3D panel (the consoles can be created to monitor several vehicles in simultaneous). The Renderer3D panel plug-in makes use of the Neptus framework API to list and access the video panels present in the console. This way the Renderer3D does not connect directly to the hardware video capture devices but only to the existing video panels in the console to extract the frames images. While the video images are fed, to video panels, using capture devices the state of the vehicles (position and attitude) comes through the network interface. Whenever a vehicle state message arrives to the console the projection object have to be reformulated and in a optional refresh rate the video texture has to be actualized on the this projection object. Notice that there is no synchrony between the data state of the vehicles and the video images arrived to the consoles.

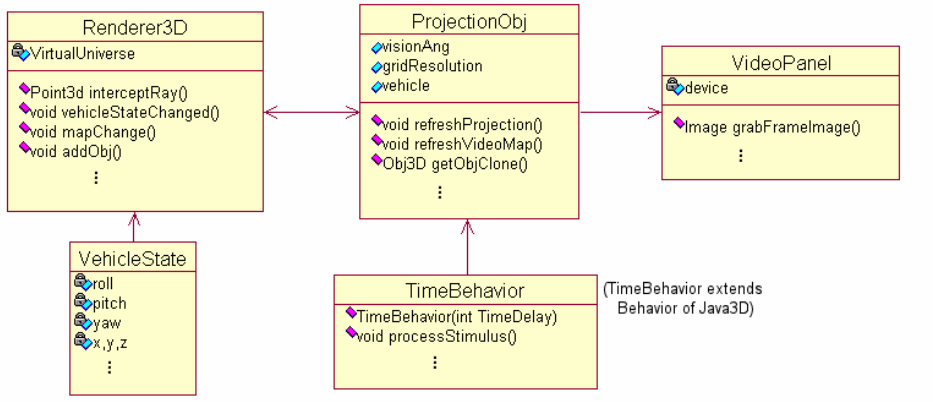

Figure 3: Class diagram of the base architecture for the video projection implementation.

In Figure 3 the Renderer3D class centralizes all the functionalities related to 3D viewing and has all data related to the virtual world. It has an interface to add/remove map objects and actualize the vehicle positions information (VehicleState). The ProjectionObj class keeps the information related to the grid projection object. The TimeBehavior class work as a timer, initialized in the 3D engine, that calls periodically the refreshVideoMap() method of the ProjectionObj class to ask the VideoPanel for a new Frame and change the texture. The VideoPanel class represents the Video panels placed in the console.

3.1 Positioning the vehicle

The method to extract the vehicle positioning, in underwater environment, is considered a complex issue to overcome. The position detection process is made by sonar triangulation. The estimated data state position of the vehicle can be elevated up to 10Hz in a reliable way, after applications of filters. The attitude data

is simpler to extract and it is made by an IMU sensor. All this data is combined in a single message and send to the consoles (ground station) through an “umbilical cable” that directly connects the vehicle to the Neptus console. In UAV’s vehicles type the positioning is made by GPS sensors and transmitted by WIFI.

3.2 Capturing video images

Neptus framework uses the Java Media Framework API to connect with the video capture devices. The video capture frequency depends on the device type. For video signal conversion from the ROV camera to digital format it is proximally 20Hz. In the console component panel, using JMF API, it is possible to ask any time for the current video frame of the component. The frame is easily converted to a texture and used by the Neptus 3D engine. Notice this video plug-in component work in Windows and Linux (as well as the rest of the system).

3.3 Java3D implementation

The Neptus Framework 3D base engine used is Java3D in retained mode that means we have a graph representing the scene (for more details see [JAVA3D]). The projection algorithm makes use of the Java3D functionalities for the interceptions calculations. To build the grid projection object we used the utils.picking.PickTool Java3D class that allows us to define an origin and direction of a ray to return the interception point in the 3D world. The objects in the scene graph of the Java3D engine have to be assigned with the ALLOW_INTERSECT flag property on. This way we can select the objects that are able to receive projections. The PickTool object must also be set to work in PickTool.GEOMETRY mode. That means the interceptions are made using the objects polygons and not by its bounding boxes (by default).

To minimise the CPU usage, in the interceptions calculations, the grid projection resolution should be as low as possible. If the projection is made into a plan object the grid can be reduced to four points only, generated by four rays. But if the projection angle, in the plan object, is too different from the normal vector of the plan the resolution should be sufficient high to make the perspective correction of the video images (Figure 4). Also if the object receiving the projection is irregular (i.e. terrain object) the resolution should be high enough to make a perfect adaptation of the grid projection.

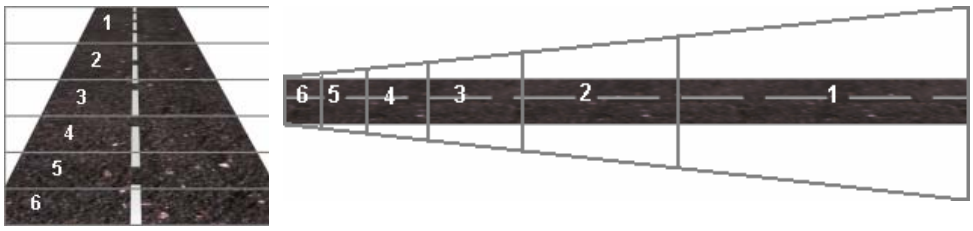

Figure 4: Grid segmentation used to make the perspective correction when the video image is projected.

For the video texture image refreshing in the projection object was created a Java3D behavior class. This type of Java3D objects are nodes used in the scene graph to make dynamic operations automatically by the engine. Our TimeBehavior (Figure 3) class, extended to Java3D Behavior, is simply used to inform the ProjectionObj class to periodically actualize the texture with the video image in synchrony with Java3D render rate.

4. Conclusion and future work

The presented method is the first solution for this functionality applied in Neptus,where the speed limitation in the interception calculation has been consider. The stencil shadow volumes algorithm, normally applied in shadows creation, is being studied for the possibility of implementation and adaptation in Java3D to produce the video projections in more complex and irregular objects. Others engines with support to produce shadows by default are being considered as well, like JavaMonkeyEngine [JME]. In the near future the presented method will be tested in UAVs type of vehicles that are being developed at USTL (another approach in [SOIA04]).

5. References

[JAVA3D] "Java3D Home Page " 2007, <http://java.sun.com/products/java-media/3D/>

[NEPTUS06] J. Pinto, Dias, P.S., R. Gonçalves, G. M. Gonçalves, J. B. Sousa and F. Lobo Pereira (2006), Neptus – A Framework to support the Mission Life Cycle. In: *IFAC 2006 The 7th Conference on Manoevring and Control of Marine Craft*, Lisbon, Portugal, September 20-22.

[NHOME] Neptus Framework, <http://whale.fe.up.pt/neptus/>

[NPS] Lee, C. S. (2004), NPS AUV Workbench: Collaborative Environment for Autonomous Underwater Vehicles (AUV) Mission Planning and 3D Visualization. MSc Thesis, Naval Postgraduate School, Monterey, U.S.A., March 2004

[MRA06] Dias, P. S., J. Pinto, G. M. Gonçalves, R. Gonçalves, J. B. Sousa and F. Lobo Pereira (2006b), Mission Review and Analysis. In: *Fusion 2006 The 9th International Conference on Information Fusion*, Florence, Italy, July 10-13.

[PEARTH] Pict'Earth, <http://www.pict-earth.com/>

[SOIA04] Henri Eisenbeiss (2004), A mini Unmanned Aereal Vehicle (UAV): System Overview and Image Acquisition. International Workshop on*: Processing and Visualization using High-Resolution Imagery*, Pitsanulok, Thailand, November 18-20.

[UENG] UnrealEngine, <http://udn.epicgames.com/Two/ProjectorsTableOfContents.html>