Real-Time Aircraft Radar Simulator for a Navy Training System

J. P. D'AMATO, C. GARCÍA BAUZA, G. BORONI, M. VÉNERE

Universidad Nacional del Centro and CICPBA-CONICET-CNEA, 7000 *Tandil, Argentina*

Received 1 February 2010; accepted 19 September 2010

ABSTRACT: A real-time aircraft radar simulation applied in naval training is presented. The implemented algorithm works on a 3D synthesized environment, described by large sets of polygons—a typical scene can have a million of triangles—and the lobe is discretized with a cluster of rays. The radar display is recreated solving fast ray–polygon intersections (a variation of Ray-Shooting), and mapping them on the screen. This work proposes a new polygonal simplification method and a geometric classification algorithm in order to solve the intersections efficiently. This methodology leads to high fidelity images and real-time radar simulation, which operates at the specified 15 revolutions per minute rate. The results were tested with trained aircraft pilots. © 2010 Wiley Periodicals, Inc. Comput Appl Eng Educ 21:606–613, 2013; View this article online at wileyonlinelibrary.com/journal/cae; DOI 10.1002/cae.20505

Keywords: aircraft radar simulation; geometric algorithms; training

INTRODUCTION

Training simulators is today a common tool for preparing people for situations [1,2]. There is a great number of well-known commercial products in different areas such as FRASCA Mentor in aviation and TRANSAS Navi-trainer in shipping, among others. The realism of the visualization, the look-and-feel of the instruments and the vehicle behavior, define the quality of the simulators.

In marine and aircraft vehicles, one of the most important devices to be simulated is the radar, this has promoted many researches since [3–5]. Basically, radar (see Refs. 6 or 7) emits and listens pulses of radio waves or microwaves up to a certain distance or range. These pulses should be collimated as much as possible in a way that the returned echoes come from the direction that is being observed. Even though, the generated pulses turn into a main lobe and several secondary lateral ones. The width of the lobes determines the radar quality, the more narrow higher quality, because the objects placed in the main direction of the lobe will be correctly recognized, but the rest will not.

In ship radars, the lobe is horizontally emitted in different directions, while it is turning at a rate of 14–20 revolutions per minute, generating about 4,000 emit–listen processes for each 360° around (see Ref. 8). In aircraft radar, the scan has a horizontal (azimuth) and vertical (tilt) orientation, and the device is smaller than the ship ones, so the lobe is wider. Radars of this type make 1,000 emit–listen processes in 360°, while it turns around 10–15 times per minute.This kind of airborne radar is applied to early warning and sea surveillance and it's installed on airplanes like Tracker S-2T (Fig. 1), property of the Argentine Army. The radar antenna is housed in a circular radome, which rotates continuously (rotodome).

To recreate the radar computationally the lobe is approximated as a ray beam and the targets as a polygon set in a three-dimensional space. An emission–reception process is meant to find the rays–polygons intersections, and then displaying on the screen the first intersection of each ray.

Similar approach can be find in the work of Ref. 9 which explores in detail this method, testing on synthetic metallic objects and getting good experimental and theoretical results [10]. Implements a similar algorithm to generate SAR images (Synthetic Aperture Radar) on a distributed architecture to obtain the necessary performance [4]. Works with graphic cards (GPUs) for the real-time simulation of a SAR Radar, but with a very constrained physic model.

Our simulator must runs in a distributed application environment directed by an instructor module, which handles position, velocity, and direction data of every vehicle in the scene; besides environment variables such as sea state, time of day, weather conditions, etc. Other applications, like the plane console, update this information after every user command event. At a defined elapsed time, all applications received global scenario data in order to update local state. Data that represents line coast and topography could be also detected by radar, but unlike mobile targets they are stored locally to avoid network congestion. The context of the modules in the simulator is shown in Figure 2.

Correspondence to J. P. D'Amato (jpdamato@exa.unicen.edu.ar).

Figure 1 Tracker S-2T plane. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

The paper is organized as follows. The second section describes how the scenes and the lobe are modeled. Third and fourth sections show implementation details of the proposed algorithm. Fifth section presents the results of testing the algorithm in a complex scene. Finally, sixth section concludes the paper with a discussion of results and a glimpse of future works.

THE DISCRETE MODEL

Aircraft radar, unlike naval and mostly terrestrial radars, takes into account height position of the vehicle, what makes necessary that data should by modeled in a three-dimensional space. The operation of aircraft radar consists in the emission of electromagnetic pulses that compose a lobe or beam, and the listening of the echo generated after the bounce on land surfaces and potential targets within the detection area. The extent area to which an object reflects or scatters radio waves is called its radar cross section.

As shown in Figure 3, the scanning beam rotates around azimuth direction and has an inclination over the horizon, called tilt. The time a pulse takes in returning is used to infer the distance to the object that originated the echo and the intensity is proportional to the refraction properties of the target. In case the object is soft (e.g., sand) the returned signal is weak, if it is hard (rock, metal) the signal is stronger. This property can be represented on solid objects using a reflectivity factor μ for each triangle of the mesh.

To simulate the behavior of the scanning beam a simplified algorithm of Ray-Shooting, with certain modifications, is used. These modifications are mainly in the structures and geometric classification of objects in the scene. The lobe is discretized in the XZ plane (tilt's plane) on a set of rays separated by a constant angle, $\Delta\beta$. The Figure 3 shows the lobe and its discretization in rays.

How many rays must be used will depend on the resolution of the radar screen. Figure 4(Left) shows an example were a large $\Delta\beta$ is chosen. As can be seen, there are strips of pixels that are not visited by any ray. If the chosen angle is very small, this problem will not appear, but at a cost of greater calculation time by having to compute a greater number of rays than needed. For standard aircraft radar of 320 pixels wide and a beam lobe of 4° the proper number of rays is 1,200 in azimuth and 120 in tilt.

The objects in the scene are modeled using a surface representation with triangular elements. The system represents each possible target in the scene (ships, buoys, or submarines) with detailed 3D models, necessary for a proper visualization. In the case of the radar these level of details are not perceived, so we generate simplified models with around 50 triangles as shown in Figure 5. That implies a significant reduction in the number of intersections that must be computed.

A special treatment is necessary for the representation of the topography. As a starting point, we use DEM files obtained from a database of the USGS EROS Collaboration Program/NASA SRTM (Shuttle Radar Topography Mission). These DEMs are regular grids with values of land heights in each pixel. Each file cover a region of 1° in longitude and 1 in latitude using a grid with 90 m cells. A typical training scenario has 4° latitude per 4° longitude, which gives meshes with millions of elements.

These kinds of meshes are useless for a real-time application and must be simplified in some way. To do that we pre-process the original mesh, removing elements from the regions where changes in the slopes are not high. This can be done with the algorithm described in Ref. 11, using a quadtree for the definition of the level of detail in each region and pre-defined templates for the discretization of the resulting terminals.

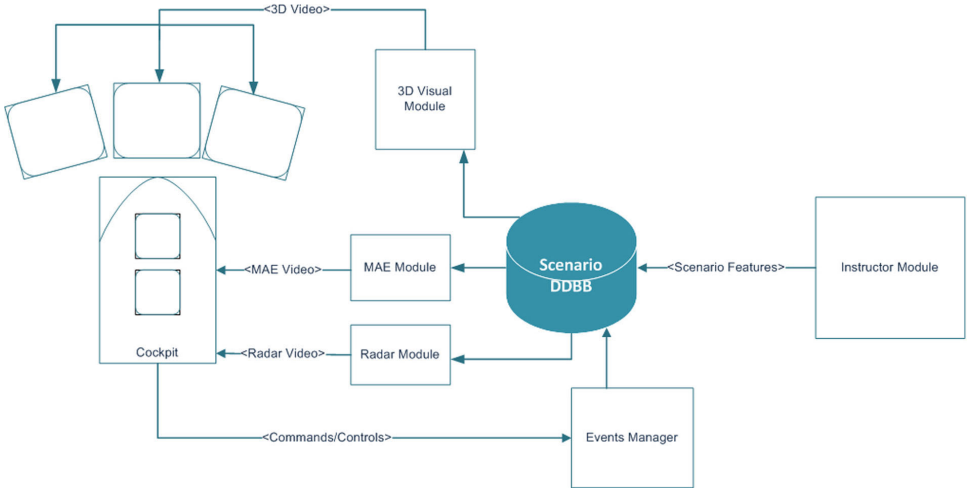

Figure 2 Distributed application environment.

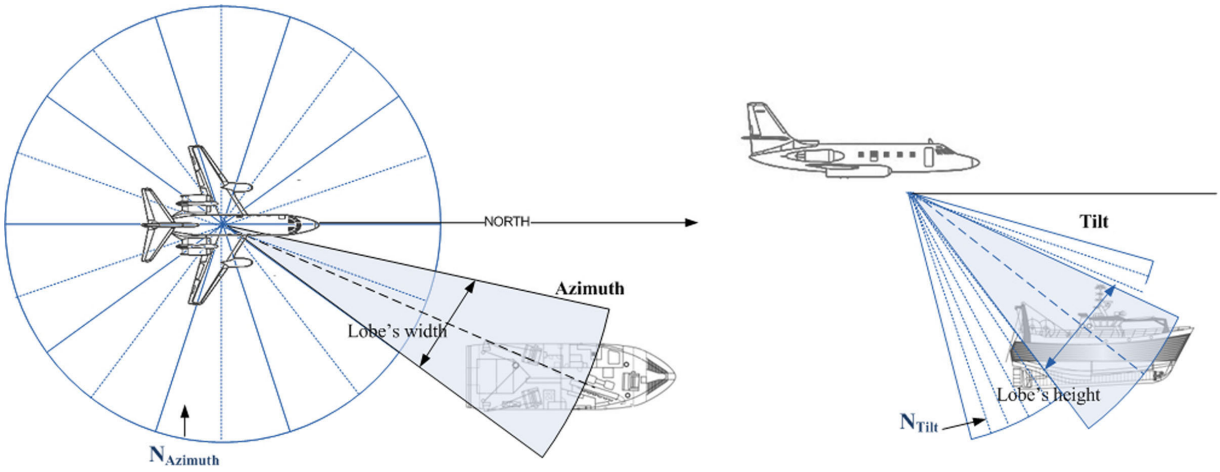

Figure 3 Plan view (XY or Azimuth's plane) and side view (XZ or Tilt's plane) of radar's lobe.

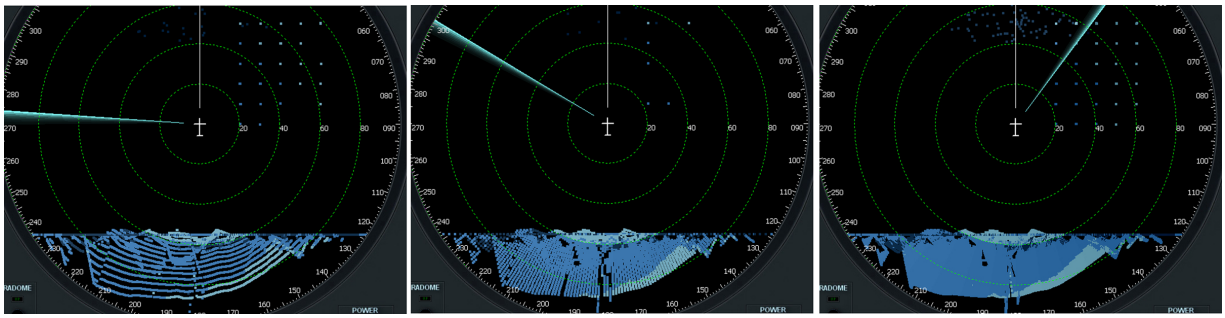

Figure 4 (Left) Insufficient number of rays in tilt. (Center) Insufficient number of rays in azimuth. (Right) Proper number of rays. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Another problem that must be solved is the union of neighbors DEMs. Usually, the borders between DEMs will show C0 discontinuities, generating holes in the terrain surface, and producing black lines in the final radar image. We use the idea proposed in Ref. 12 that assure a C0 continuity adding points at the border when necessary and merging the position of the "coincident" points (one point of a DEM that are near a point of the neighbor DEM).

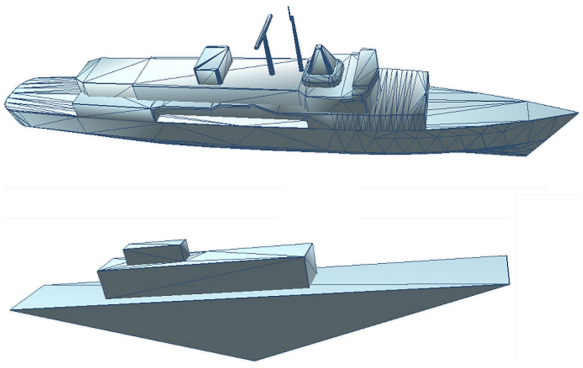

Figure 5 (Top) Mesh used in the 3D module. (Bottom) Simple boats representation used in the Radar.

Resulting meshes after applying the algorithm have around 10^5 triangles, but the amount varies according to terrain altitude variation. In Figure 6, a mesh built from a 4×4 grid of DEMS is shown, applying a color scheme to indicate different Latitude/Longitude files. Even simplified, a scene is composed by hundreds of thousands of triangles, what makes necessary to structure this data into a hierarchy that could solve efficiently the ray–object intersection algorithm.

RADAR SIMULATION ALGORITHM

As mentioned above, the implemented technique for the simulation is a Ray-Shooting algorithm, like proposed in Ref. 13. The core of the technique is the calculation of intersections of elements in the space, in our case triangles, with rays coming from the position of the radar antenna. For each ray, the closest intersection to the radar is stored while the rest intersections are ignored. Finally, the obtained points are projected onto the screen depending to the distance and colored according their returned intensity.

Each beam is represented by the following information:

- Origin: aircraft's antenna position in 3D space.
- Tilt angle: component that indicates the direction of the ray according to the aircraft's horizon.

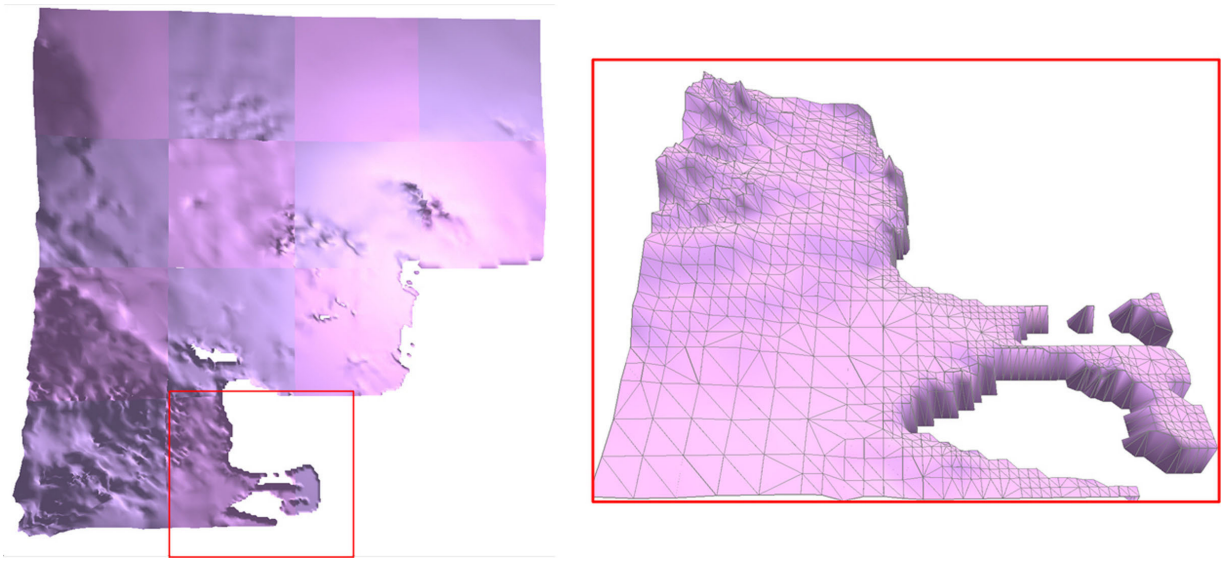

Figure 6 (Left) Mosaic of DEMs used in a training scenario. (Right) 3D detailed view of a region. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

- Azimuth angle: component that indicates the ray direction according to absolute north.

The computational cost for this straight forward algorithm is $O(N_{pol}\, N_{tilt})$, where:

- N_{pol}: Amount of polygons used to describe the scene and targets.
- N_{tilt}: Amount of rays used to discretize the lobe.

This cost is for each emission-listening process. Considering a standard radar that operate at 15 revolutions per minute with 1,000 emissions per revolution, a standard scenario with 10^5 elements, and a lobe discretization with 200 rays, will give a rate of 5×10^9 intersection ray–triangle per second. This is expensive enough to justify the use of some efficient search scheme.

The first and simplest idea to reduce this cost is to discard elements that are outside the visible range. The so called "radar's horizon" (HR), indicates the maximum distance from which the target will be achieved by the radar and meets.

$$\mathrm{HR} = \mathrm{factor}\sqrt{H_{\mathrm{Emitter}} + H_{\mathrm{Reception}}} \leq A_{\mathrm{R}}, \qquad (1)$$

where factor is a known constant for each type of simulated radar, A_{R} is the range of the radar and H_{Emitter} and $H_{\mathrm{Reception}}$ indicate, respectively height of radar and target. Elements that do not satisfy this equation are discarded and not considered in the intersections algorithm. This idea work well and eliminates a large numbers of triangles, but will not solve the computational cost problem.

THE CLASSIFICATION METHOD

To achieve a real-time representation of the radar image it's essential to have an efficient structure for classifying and searching the elements in the scene. Also the time required to update this structure should be minimum, because in a dynamic environment where all the elements relative to the radar are moving, a re-classification will be necessary at every time step.

We tried some classical classification algorithms, like grids and Kd-trees [14]; these structures works fine in the searching step but results expensive under frequently data updates. According to the movement of the lobe, the proposed solution is a classification strategy based on mapping triangles onto a semi-spherical surface. This algorithm takes all the polygons of the scene, for each triangle estimates its bounded box and calculates the corresponding maximum and minimum tilt and azimuth angles, as shown in Figure 7. Each triangle is then referenced from a two-dimensional matrix structure indexed by azimuth and tilt.

A triangle may cover more than one cell. For example, a triangle that is in the range of 16–18° tilt and 345° azimuth, will be stored in the cells [16] [345], [17] [345], and [18] [345]. Each

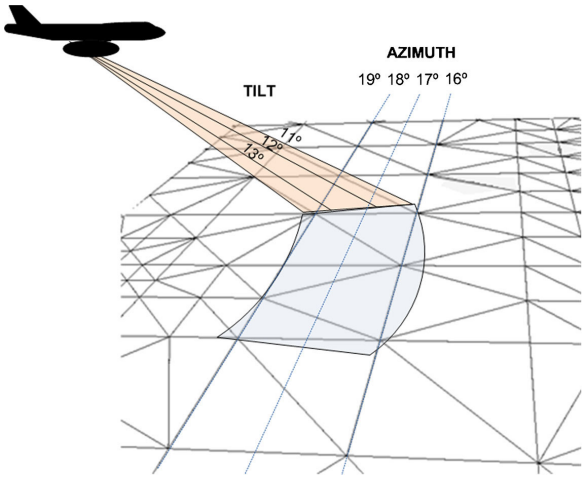

Figure 7 Triangles set classified by tilt and azimuth.

triangle is stored in the structure only if it meets the horizon radar equation [see Eq. (1)].

The search time in this new structure is almost constant. Each ray gains direct access to the elements that are in its direction. The total ray evaluation time looks like:

$$T_R = K_B + \frac{1}{2}\left(\frac{N}{C_A + C_T}\right)T_I, \quad (2)$$

where K_B is the search time in the structure, T_I the cost of solving an intersection, N is the amount of triangles, C_A and C_T are the discretization of space in Azimuth and Tilt, respectively. The amount of cells we use for azimuth range from 540 to 720 (azimuth moves between 0° and 360° with 0.75° and 0.5° step), and for tilt is 104 (tilt moves between −26° and 26° above the aircraft horizon with a step of 0.5°).

Ideally, if elements are uniformly distributed in space, each cell would contain the same amount of triangles and the time T_R is nearly constant and known. At each time step, the algorithm requires the update of the simulation's state to reflect the movement of the aircraft and targets, which leads to a re-classification before solving rays intersections. These operations can be split in two concurrent threads, taking advantage of multi-core hardware.

The time required to update the classification matrix is proportional to the amount of triangles in the scene: for each element, only its azimuth and tilt relative to the aircraft are computed, then data are stored in the structure at a constant time. To allow simultaneous reading and modifying data, a double matrix or buffer scheme is used. The read-only matrix, called active matrix, is accessed by the intersection calculation module. On the second matrix, called temporary matrix, elements re-classification is performed after radar and targets updated information is received. Once classification is ready, matrixes are exchanged, active matrix becomes the temporary array and vice versa. This step is unnoticeable to the user.

It is easy to show that the computational cost of the proposed algorithm is $O(N_{pol} + N_{tilt})$ instead of the original $O(N_{pol}\, N_{tilt})$. The first-term $O(N_{pol})$ correspond to the classification process, and the second $O(N_{tilt})$ to the calculation of the rays–triangles intersections.

GENERATION OF THE RADAR IMAGE

The image of the radar screen is generated pixel by pixel, scanning all rays (N_{tilt}) for every azimuth direction in a sequentially and synchronized clock-wise rotation. The highest intersection ray–polygon detected corresponds to a specific pixel in the radar screen.

The intersected triangle returns a value of echo intensity that depends on the type of material it had been assigned; this echo strength is associated with a specific color, according to a color table defined for each kind of radar. In general, strong signals (corresponding to hard land sectors or metal hulls of ships) are red, intermediate intensity values (soft land) are yellow, and low ones (sand) are blue. Figures 8 and 9 show some radar images obtained using this scheme.

The model as proposed can simulate most of the physics involved in the process, but there are some phenomena that cannot be treated in this way. The most important is the simulation of the reflectivity on the water surface under windy conditions. In this case, the surface must be represented using an excessive large number of triangles to model the waves and its interaction with objects. Instead of using many triangles, a distribution that emulates the height of waves encoded as a texture would be a cheaper alternative: for each ray that intersects a triangle, the intersection position is mapped to texture coordinates and the corresponding value is read. The distribution chosen is a Perlin Noise generated dynamically according to sea state, as shown in Figure 10.

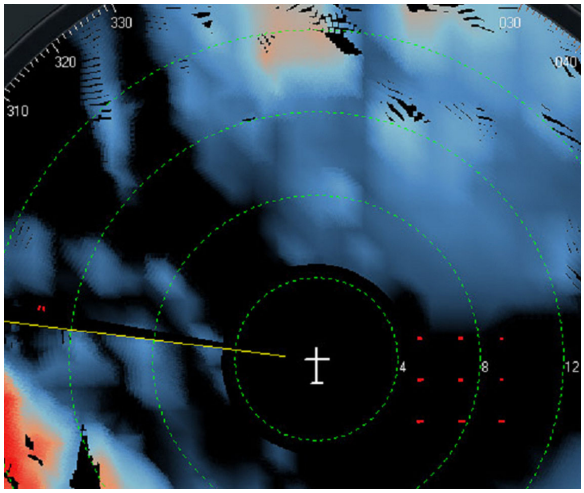

Figure 8 Enlarged display of terrain and vehicles (red dots) using a gradient color scheme. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

RESULTS

Some exhaustive tests with a complex scene and varying radar ray resolution were carried out to evaluate the proposed technique. The quality and realism of the images was validated by aircraft pilots from the Argentine Army.

We include here a test performed in the worst case scenario for the simulation, where the plane is completely surrounded by land and some targets, all represented by 10^5 triangles. This kind of radar operates between 10 and 20 revolutions per minute, with around a thousand emission-listening processes for each revolution; therefore, the computational cost for the simulation must not be >3 ms for each emission-listening process (or 3 s per revolution).

To achieve this performance, we first analyze the required resolution for the classification process in order to obtain few triangles in each cell. As can be seen in Figure 11, in this case a number of 500 sectors for azimuth (and also 120 for tilt to work with proportional cells) is sufficient to obtain <20 triangles in average per cell. Although the classification process is done in parallel on another processor, it is important to remark that the complexity of this algorithm is linear with the number of triangles, and the computer time involved is negligible compared to the ray–triangle intersection computation.

Using a classification of 540 × 160 sectors and 1,200 emission-listening processes in each revolution, we obtain an overall computation time of 1,980 ms for the scenario shown in Figure 6, enough for 30 revolutions per minute (more than necessary). The test was done on a simple dual core PC with a 2.8 GHz clock.

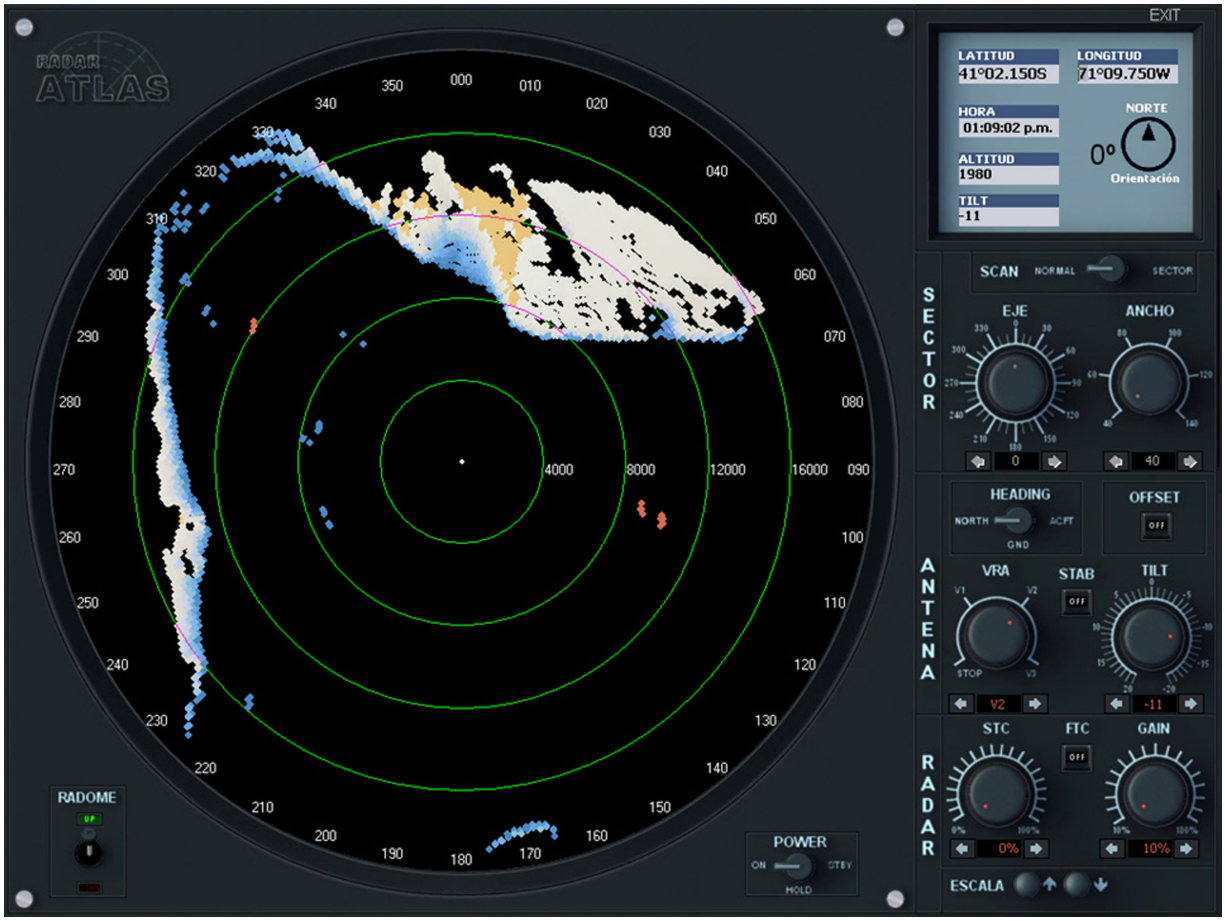

Figure 9 Radar's user interface. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

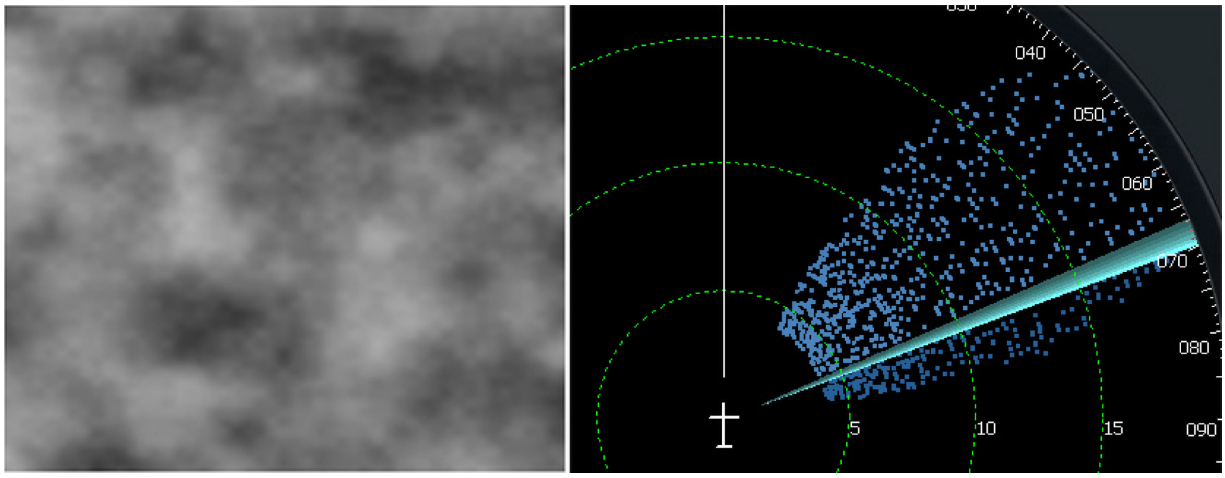

Figure 10 (Left) Texture pattern choose (Right) radar image obtained. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

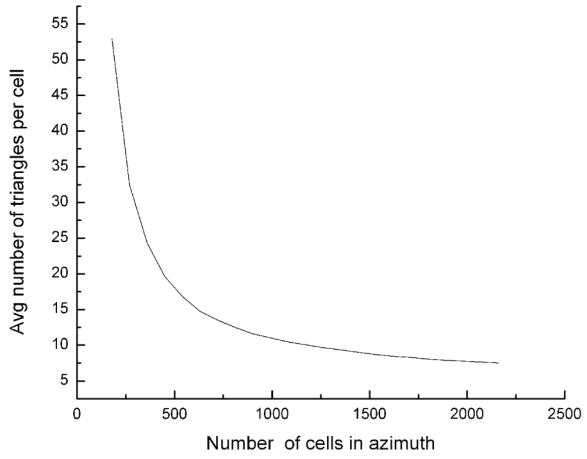

Figure 11 Classification and total time for a whole radar's revolution.

According to expert user's reviews, the real-time images obtained were suitable for training. Figure 8 shows enlarged display of terrain and vehicles using a gradient color scheme. Ships, which reflect the largest amount of energy are seen as red points, while colors assigned to terrain vary in proportion to the hardness of the surface (near the coast the signal becomes weaker). Figure 9 shows the radar user interface with the standard controls such as GAIN or STC (to reduce the sea clutter on the display) and change the tilt of the antenna or the scanning region. Finally, Figure 12 shows colored echoes with a discrete color scheme. All the images correspond to a 200 × 200 mile region in the south coast of Argentina, while the plane flies at 18,000 feet.

CONCLUSIONS

An aircraft radar simulator was efficiently implemented and integrated to a training system. The proposed classification algorithm has achieved the specified times, even in complex and large training scenarios, and it also proved to be a robust and temporally precise technique suitable for being executed on multi-core processors and within a collaborative environment.

As current lines of work, the algorithm calculation is being implemented in graphic cards (GPUs) and the simulation model is being extended, incorporating new variables such as channeling temperature and rebound effects.

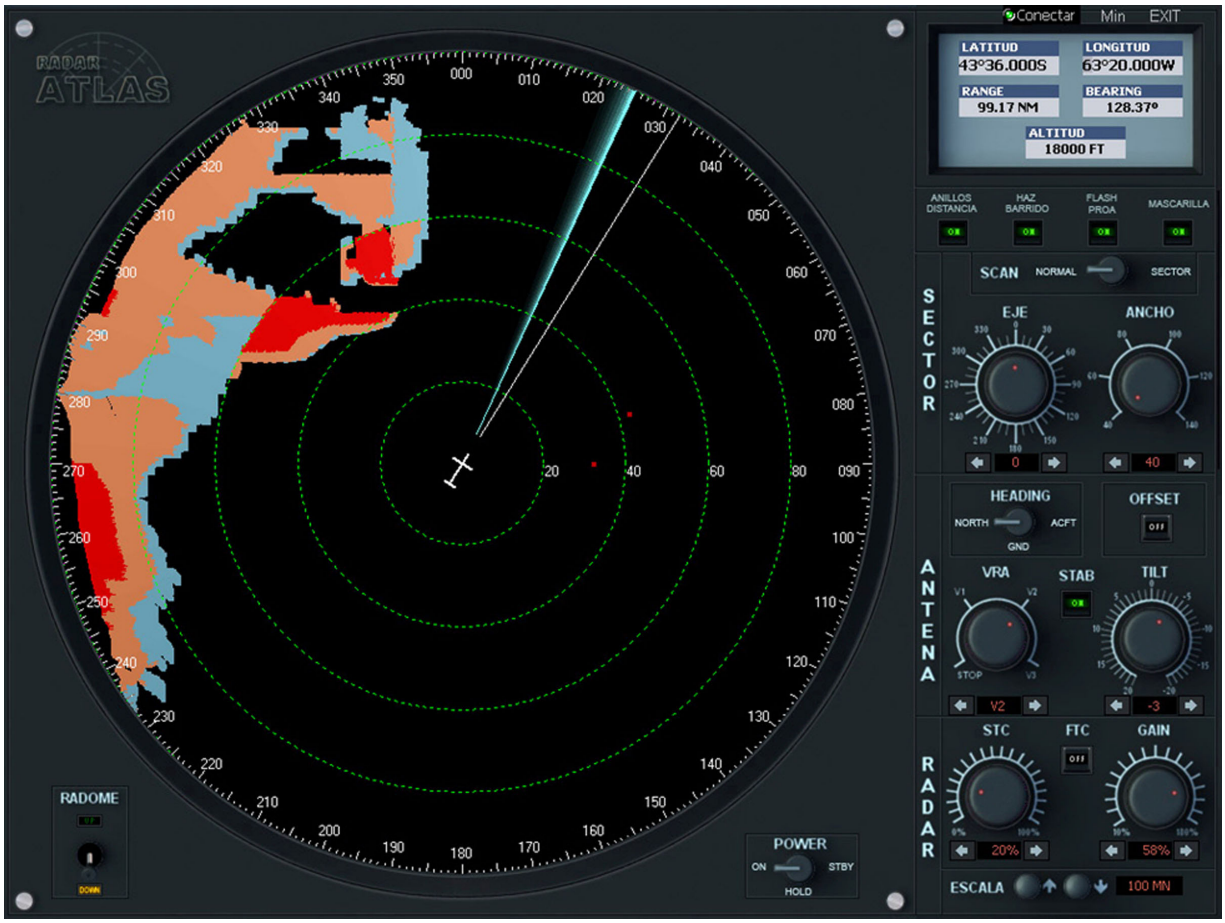

Figure 12 Colored echoes with a discrete color map. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

ACKNOWLEDGMENTS

This project was funded by the Argentine Armed Forces and is part of the Atlas II project which intends to recreate all instruments of ships and aircrafts used in Argentine.

REFERENCES

[1] R. Lechner and C. Huether, Integrated live virtual constructive technologies applied to tactical aviation training, Interservice/Industry Training, Simulation and Education Conference (IITSEC), 2008.
[2] P. A. Roman and D. Brown, Games—Just how serious are they?, Interservice/Industry Training, Simulation and Education Conference (IITSEC), 2008.
[3] G. L. Bair, Advances in airborne radar simulation, Proceedings of The 9th International Training and Education Conference, 1998, Lausanne, Switzerland.
[4] T. Balz, Real-time SAR simulation of complex scenes using programmable graphics processing units, ISPRS Commission VII Symposium ''Remote sensing: From pixels to processes,'' Netherlands, 2006.
[5] R. Boyell and H. Ruston, Hybrid techniques for real-time radar simulation, Proceedings of the AFIPS Joint Computer Conference, 1963, pp. 445–458.
[6] C. Coock and M. Bernfeld, Radar signals: An introduction to theory and applications, Artech House Incorporated, USA, 1994. ISBN 0890067333.
[7] H. Meikle, Modern radar system, Artech House Incorporated, USA, 2001. ISBN 1580532942.
[8] J. N. Briggs, Target detection by marine radar, The Institution of Electrical Engineers, London, UK, 2004. ISBN 0863413595.
[9] G. Ramière, Couplage des méthodes asymptotiques et de la technique de lancer de rayons pour le calcul du champ rayonné par des objets metalliques 3D complexes, PhD thesis, Paul Sabatier University, Toulouse-France, Sept. 2000.
[10] H. J. Mametsa, S. Laybros and A. Bergès, Asymptotic formulations and shooting ray coupling for fast 3D scattering field evaluation from complex objects and environment—Applications, ICEAA, Turin (Italy), September, 2003.
[11] M. V. Cifuentes, M. J. Vénere and A. Clausse, Interactive remeshing for navigation over landscape models, Latin American Applied Research, 40 (2010).
[12] M. Vénere, M. V. Cifuentes, J. P. D'Amato and C. García Bauza, Editor de Escenarios para Aplicaciones de Realidad Virtual, VI Simposio Argentino de Tecnología en Computación, 2005.
[13] V. Havran, Heuristic ray shooting algorithms, Technical Report (Ph.D. Dissertation), Faculty of Electrical Engineering, Czech Technical University, Prague, Nov. 2000.
[14] K. R. Subramanian and D. S. Fussel, A search structure based on k-d trees for efficient ray tracing, Technical Report (Ph.D. Dissertation), The University of Texas at Austin, Dec. 1990.

BIOGRAPHIES

Juan Pablo D'Amato is a Systems Engineer since 2004 and he's currently doing a PhD in computational geometry at the UNICEN University (Tandil, Argentina). He's professor assistant in Computer Graphics courses and has worked in the development of real time systems and simulators for the Argentine Army Forces. His main research interests include geometry, high performance, visualization and virtual reality applied to training.

Cristian García Bauza has been a System Engineer since 2006. He's doing a PhD in fluids simulation for real time applications at the National Southern University (Bahía Blanca, Argentina). He's professor assistant in Computer Graphics courses and has worked in the development of vision systems and simulators for the Argentine Army Forces. His main research interests deal with natural effects in visualization, virtual reality, physics simulation and graphic engines architectures.

Gustavo Boroni received his System Engineer degree and PhD in Computer Sciences at UNICEN University (Tandil, Argentina). He's professor assistant in Programming and Systems Communications area at UNICEN University. He's also an Assistant Researcher at the National Council of Scientific and Technological Research (CONICET) since 2010. He has worked and published several articles in numeric methods, real time simulation and computer graphics.

Marcelo Vénere is a PhD in Nuclear Engineering from Balseiro Institute, Argentina. He's an Associated Professor in Computer Graphics and Algorithms courses at the UNICEN University (Tandil, Argentina). He is leading several projects in arterial fluid simulation, real time training and computer graphics; among other topics. He is the Co-Director of the PLADEMA Research Institute of the UNICEN University at Tandil, Argentina.