Real-Time SAR Simulation on Graphics Processing Units

Timo Balz, Institute for Photogrammetry, University of Stuttgart, Germany

Abstract

SAR simulators usually apply the ray-tracing approach. Ray-tracing, which is also used for virtual image generation, is based on accurate physical models, but is unfortunately rather computational time intensive. Because of this, real-time applications, like interactive visualisation, in general use the rasterization method. Rasterization is less complex to calculate and is therefore faster. This paper presents the real-time SAR simulator SARViz, which uses rasterization and is implemented on programmable graphics processing units, which are nowadays included in most modern PCs. SARViz is able to visualise even complex scenes in real-time, using the tremendous development in computer graphics for the next generation SAR simulator.

1 Introduction

Useful applications of SAR simulations are hindered if the simulator needs minutes or hours to simulate a single scene. For image interpretation and object detection, databases containing simulated objects of interests are created in a pre-processing step. This is necessary because the response time of common SAR simulators is too long. A real-time SAR simulation supersedes these databases. Furthermore, it allows the modification of models or sensor properties in real-time. SAR mission planning also would benefit from real-time SAR simulations. The ability to simulate different sensor properties and immediately see the simulated results is an advantage, especially for the mission planning in urban areas. The optimal azimuth direction, looking and squint angle can be found easily by testing all reasonable options and immediately see the results. Image processing applications based on SAR simulated images also benefit from fast simulation results, due to the reduction of the overall computation time.

Modern graphic cards offer 3D hardware acceleration and programmable graphics processing units (GPU). These graphic cards are standard equipment for almost every new PC and offer a boost in computational power. Since these graphic cards are mostly used by computer games, the game industry is the driving force behind this development. Thanks to the economy of scale, these graphic cards are now widely available and inexpensive. In addition to gaming, the computational power of the GPU can be used for a variety of general computation purposes [1].

Realising a real-time SAR simulation, as aspired by our approach, opens new fields of applications. New algorithms for SAR image interpretation are applicable if the simulator is not the bottleneck anymore.

2 Visualisation Technologies

Real-time visualisation of real-word or artificial scenes is one of the main goals in computer graphics. Usually ray-tracing technology is being used to achieve a realistic visualisation. In ray-tracing the path of each “ray of light” is followed. Starting from the light source, each ray is traced until the ray reaches the eye. Each interaction with the respective object surfaces is simulated. To reduce the amount of rays to trace, the ray can also be traced backward, thus from the eye to the light source. The combination of both methods is reasonable [2].

The reflection of each ray at any object surfaces is simulated based on the respective material properties. This enables a high degree of realism. Ray-tracing is supposed to be based on accurate physical models and can be implemented easily. Unfortunately the ray-tracing approach is rather computational time consuming, because for each pixel of the simulated image one ray has to be traced. For this reason, this technique is normally not used for real-time applications.

Real-time visualisations are normally based on the rasterization approach. Triangulated 3D vector models are used as input data for the simulation. Each triangle is then, according to the visualisation properties, like the position and orientation of the virtual camera, transformed from the world to the screen or image coordinate system. Each triangle is rasterised and depicted at this transformed position. Triangles which are calculated first can be overwritten by triangles which are rasterised later, even if the pixel which is rasterized later is occluded. To avoid this, a so called z-buffer is used. In this z-buffer each depth value of an already rasterised pixel is saved. Every new pixel which has to be rasterised, must pass the z-test, thus must be closer to the virtual camera as the pixels already rasterised before. This technique is simple, effi-

cient and, less computational effort compared to ray-tracing is necessary. Modern graphic cards support the rasterization and provide hardware acceleration. This acceleration enables the real-time visualisation of even very complex scenes.
However, since the technique is not based on the physical light transport, it is not able to simulate scenes containing complex reflections. Still, as it is possible to create realistic scenes for computer games, it is also possible to simulate realistic SAR images using rasterization, adopting the methods developed in computer graphics as it is described and depicted in section 3 and 4.

3 GPU based SAR Simulation

The hardware accelerated visualisation is designed for visual light scene generation, but SAR images differ from such scenes in many ways. To visualise a SAR image using standard 3D hardware acceleration, these differences have to be implemented.
Modern graphic cards offer flexible and programmable graphics processing units (GPU). Using programmable GPUs allows the implementation of the SAR geometry and therefore real-time SAR simulation using standard graphic hardware. The real-time SAR simulator SARViz, which is described in this paper, is implemented using Microsoft's DirectX API [3] and the High-Level Shading Language (HLSL) for GPU programming. SARViz uses slightly modified DirectX meshes as input source. A variety of 3D CAD tools can export to this format. SARViz also offers import tools for other data sources, for example Arc-GIS datasets.

3.1 SAR rasterization

SARViz uses the programmable programming pipeline of a graphic card. Each triangle is visualised independent from the other triangles, which allows a massively parallel computing as it is realised by today's graphic cards. The vertex of each triangle is processed by so-called vertex shaders. The vertex processing treats the geometry, while the radiometry is calculated in the subsequent pixel processing.

3.1.1 Vertex processing

Each vertex or triangle point respectively, is processed in the vertex shader. Vertex shaders are highly specialized parts of a graphic card and are optimized for matrix calculations. Each point is transformed from the model coordinate system to world coordinates and then subsequently to screen or image coordinates. The position in range of each object in the SAR image depends on the distance of the object to the sensor, thus higher points, i.e. points with larger z-values, are closer to the sensor and are therefore mapped closer to near-range. The shift in range $\triangle x$ depends on the height above the ground-level z and the incidence angle σ :

$$\triangle x = z \cdot \tan(\sigma)$$

3.1.2 Pixel processing

Each triangle processed by the vertex shader is rasterized and the resulting pixels are processed by the pixel shader to compute the corresponding radiometry. Calculating the reflection intensity for SAR images is a relative complex task. Since the first programmable pixel shaders were limited to only a few calculations, complex illuminations could not be calculated. In contrast, modern pixel shaders (pixel shader version 3.0 or higher) support more than 512 machine-code commands, which is sufficient even for complex tasks.
For each pixel the corresponding face normal and the sensor position is known. Taking material properties, like the dielectric constant, and sensor properties into account, reflection strengths can be calculated in the pixel shader. Reflections from different object or object parts which are equidistant to the sensor in range direction will sum up in the SAR sensor. This effect can also be modelled by GPU based SAR simulations, e.g. by alpha blending. Alternatively already rendered information can be copied back from the render target, thus the simulated image, and added to the new calculated reflection value. Both methods have advantages and disadvantages. Accessing the render target during the rendering is not optimal, because of the heavily parallelized rendering process of the graphic card. Two pixel shaders accessing the same pixel at the same time can overwrite the result of each other. Alpha blending has at the moment no hardware support for 32-bit floating point textures. Furthermore it is disadvantageous, because in SAR processing the different reflections sum up, which is not as accurately simulated by alpha blending.
SAR images cover a high data range which cannot be represented by 8-bit images. To cover the complete dynamic range, 32-bit data is adequate. Modern graphic cards support high-dynamic range lighting with 32-bit floating point accuracy, which is sufficient for the simulation of SAR images. To use this ability, the pixel shader first renders to a 32-bit floating point texture. In the next step this texture is enhanced for 8-bit display and rendered to the screen, enabling the SARViz simulator to deliver both 8-bit data for visualisation, but also 32-bit floating point images for further processing.

3.2 Reflection properties

Usually surface reflections are a mix between the two basic reflection models, the Lambertian and the specular reflection. These reflections for example sum up in the Phong shading model. The SARViz simulator is able to handle different reflection properties for each

triangle of the simulated 3D model. According to the roughness of the material surface, the reflection is either more Lambertian or more specular and the resulting reflection is calculated accordingly. Using rasterization the reflected radar beams are not traced. Therefore the influence of specular reflections to surrounding objects is not modelled.

3.3 Speckle effect

Speckle is important for the realistic visualisation of SAR images. Both cases of speckling, i.e. with and without dominant scatterers, can be simulated. For each triangle in the model the value for the dominant scatterer can be set, whereas a value of zero stands for a speckling without any dominant scatterer. The final speckle value is calculated by the GPU, based on random values generated by the CPU.

3.4 SAR shadows

The computation of shadow areas is an integral part of ray-tracing. In the rasterization approach, the paths of the rays are not traced and every vertex and pixel is processed separately, therefore occlusions are not modelled.
By using shadow maps [6] both shadow and occluded areas can be modelled. For this purpose, the scene is rendered twice. First the scene is rendered from the position of the light source, in other words from the sensor position. Instead of reflection values, the distance of the rendered pixel to the sensor is copied to the so-called shadow map. SARViz directly simulates ground-range images to avoid the computational extensive transformation from slant-range to ground-range. Because of this, the scene is, in the final step, rendered looking from nadir direction. The distance of each pixel rendered in the nadir view is compared to the distance from the senor position. For this purpose the values stored in the shadow map are transformed to nadir. If the distance of a pixel to the sensor is longer as the value stored in the shadow map, the pixel is not visible and therefore it will not be rendered.
As described above, using rasterization for real-time SAR simulation is feasible and delivers promising results, which will be discussed in section 4.

4 Examples

The SARViz simulator is designed to simulate complex and extensive 3D models. In Figure 1, the simulation result of an E-3 Sentry (Awacs) model is depicted. The off-nadir angle of the sensor is 60° and the model surface is set to Lambertian reflection properties.

Figure 1. SARViz simulation of the E-3 Sentry (Lambertian reflection)

The simulation of the same model, but assuming metallic surface properties and a different azimuth angle, is depicted in Figure 2. The specular reflection is strong on edges, but the wings of the airplane reflect fewer amount of energy back to the sensor. The reflection from the material surface of the model in Figure 2 is not totally specular. Otherwise most parts of the wings would not reflect any energy back to the sensor.

Figure 2. SARViz simulation of the E-3 Sentry (less Lambertian more specular reflection)

Simulations of complex urban environments are depicted in Figure 3 and Figure 4. The city model, simulated in Figure 3 and Figure 4, includes 9950 buildings containing 548729 triangles and can be simulated in about 100 milliseconds, not including the time needed to copy the data to the video memory.

Figure 3. SARViz simulation of the 3D city model of Stuttgart (5m pixel resolution)

The simulation is not simplifying the models, i.e. all vertices are rendered. The use of simplified models could allow even higher frame rates during the low-resolution simulation of complex models.
In Figure 4, a subset of the model is simulated with 1m pixel resolution. SARViz is able to simulate and visualise complex urban areas in different resolutions. All buildings are assumed to have Lambertian reflection and to have the same material properties.
These results are feasible for real-time applications like SAR mission planning or object detection based on SAR simulation.

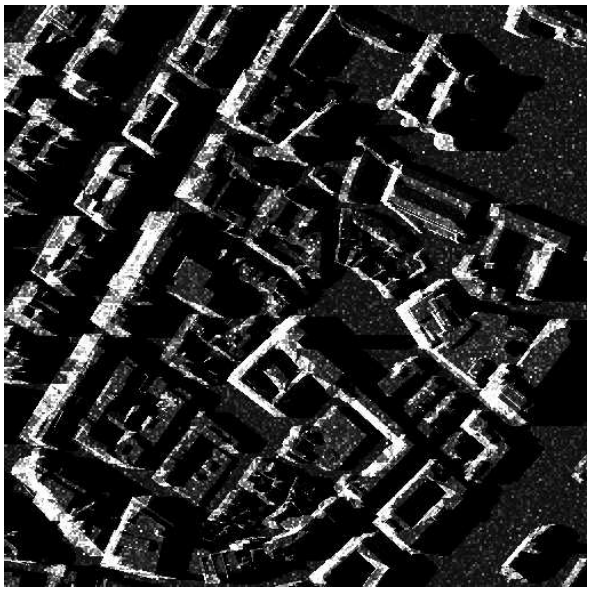

Figure 4. SARViz simulation of area surrounding the "Marktplatz" in Stuttgart (1m pixel resolution)

5 Conclusions

In our opinion hardware accelerated SAR simulation using techniques originally developed for computer games is a promising technology. The real-time simulation capability is amazing and applications based on SAR simulation profit from tremendous increase in simulation speed. In addition to consumer type graphic cards, no expensive hardware is required, i.e. real-time visualisation of SAR images is possible on standard PC hardware. On the other hand the simulation results are not based on as exact physical models as ray-tracing simulators. Especially the missing multi-bounce effect is a drawback. Double-bounce effects are realisable using environmental mapping, whereas multi-bounce requires ray-tracing. Ray-tracing on the GPU as proposed by [4] could be an alternative, as well as ray-tracing hardware acceleration [5].
However, for many applications fast results are more desirable than the exact physical modelling of multi-bouncing. For these applications the next generation SAR simulator SARViz provides high-quality results in real-time.

References

[1] Buck, I. and Purcell, T.: A Toolkit for Computation on GPU. In: Fernando, R.: GPU Gems. Programming Techniques, Tips, and Tricks for Real-Time Graphics. Boston: Addison-Wesley, 2004

[2] Heckbert, P.S.: Adaptive Radiosity Textures for Bidirectional Ray Tracing. Computer Graphics 24 (1990), No. 4, pp. 145-154

[3] Microsoft: DirectX 9.0c SDK, 2005 http://msdn.microsoft.com/directx

[4] Purcell, T.J., Buck, I., Mark, W.R., and Hanrahan, P.: Ray tracing on programmable graphics hardware. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio, Texas, 2002

[5] Schmittler, J., Wald, I., and Slusallek, P.: SaarCOR - A Hardware Architecture for Ray Tracing. In: Ertl, T., Heidrich, W., and Doggett, M.: Graphics Hardware, 2002

[6] Williams, L.: Casting curved shadows on curved surfaces. In: ACM SIGGRAPH Computer Graphics, Proceedings of the 5th annual conference on Computer graphics and interactive techniques SIGGRAPH '78, Atlanta Georgia, 1978