

REAL-TIME SAR SIMULATION OF COMPLEX SCENES USING PROGRAMMABLE GRAPHICS PROCESSING UNITS

Timo Balz

Institute for Photogrammetry (ifp), Universität Stuttgart, Germany
Geschwister-Scholl-Strasse 24D, D-70174 Stuttgart
timo.balz@ifp.uni-stuttgart.de

Commission VII

KEY WORDS: SAR, Radar, Real-Time, Simulation

ABSTRACT

The usability of SAR simulations is often limited by the long processing times of traditional SAR simulators which 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 therefore faster. The ability to simulate even complex SAR scenes in real-time is a new development in SAR simulation. In this paper we present 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

SAR images are difficult to understand. SAR simulators are key tools for SAR image understanding (Leberl & Bolter, 2001), because they can support the interpretation of SAR images. They are also useful for mission planning, especially in urban or mountainous areas. Furthermore, SAR simulations can be used for automatic image interpretation, by comparing SAR simulated images with real SAR images. All of these applications benefit from fast SAR simulation results.

The interpretation of SAR images by a human operator is an interactive process. In this process, the SAR simulation results should be provided fast, otherwise the simulation will not be regarded as a useful tool. SAR mission planning requires the testing of a variety of flight and sensor parameters to find the optimal ones. For these purposes complex environments, like city areas, have to be simulated fast, but a limited simulation quality is acceptable, as long as the results are delivered in real-time. This is also true for many automated image analysis methods based on SAR simulation, which are only useful if they can be computed fast.

To speed up the simulation, SAR simulators can apply the techniques used by computer graphics. Computer games nowadays offer amazing visualisation effects in real-time. This is possible, because of innovations in software and hardware design. Modern graphic cards provide 3D hardware acceleration support and programmable graphics processing units (GPU). These graphic cards are sold with almost every PC and provide a boost in computational power, which is mostly used for games. The game industry is the driving force behind this development and thanks to the economy of scale these graphic cards are now widely available and inexpensive. Beside game development, the computational power of the GPU can be used for a variety of general computation purposes (Owens et al, 2005) (Buck, 2005), like for example linear algebra, signal and image processing, audio processing and data mining.

As an introduction, the SAR principle will be explained briefly in section 2, followed by a short overview of the techniques used in computer graphics. In section 4 the real-time SAR simulator SARViz will be described. Example images will be depicted and discussed in section 5.

2. SAR PRINCIPLE

SAR systems are using different wavelengths as optical systems. Beside the radiometric differences, the image geometry is also different. SAR systems are active remote sensing systems. Due to their running-time or distance geometry, the position of a pixel in range direction depends on the distance between the sensor and the imaged object. The SAR system is emitting a radar beam and the radiometry of the SAR image is determined by the backscattered energy. Thus, the SAR image radiometry depends on the sensor as well as the roughness and the dielectric constant of the reflecting material.

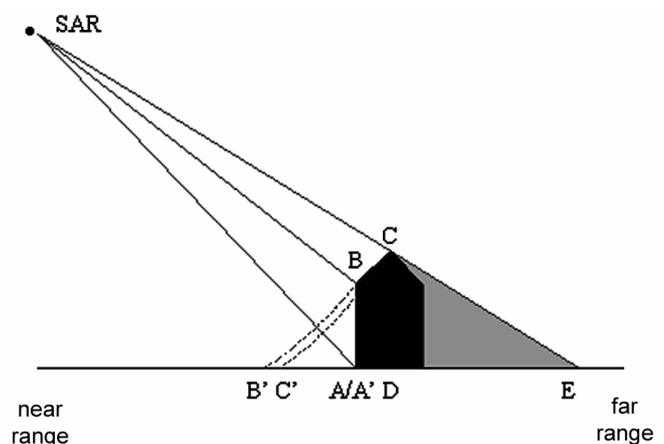


Figure 1. SAR geometry in range

In Figure 1, the SAR geometry is shown schematically in range direction, which is normally perpendicular to the flying direction of the sensor. Point A is imaged correctly in A', whereas the points B and C are imaged in B' and C', closer to near range, as their real position. This so-called layover-effect is caused by the running-time geometry of the SAR sensor. The range between A and E is the shadow area. Because of the displacement of point C in C' the shadow is starting at point A and not at point D.

Beside bi-static SAR, the SAR sensor emits and receives the electromagnetic wave. Only energy which is reflected back can be received by the sensor. In contrast to passive remote sensing systems, SAR systems control the electromagnetic energy used for image generation. The properties of the electromagnetic wave are well known and can be influenced, whereas this is not true for passive systems. For the simulation of SAR images, this is beneficial, because the electromagnetic wave is known and the behaviour of the wave can therefore be modelled more exactly. Still the material properties, like for example the dielectric constant, are unknown.

SAR images are different from images in the visible domain of the light and are therefore unusual for the human operator. SAR simulations can be a useful tool for SAR image understanding and interpretation.

3. VISUALISATION TECHNIQUES IN COMPUTER GRAPHICS AND SAR SIMULATION

Real-time visualisation of real-world or artificial scenes is one of the main goals in computer graphics. Ray tracing, which is based on the physical model of light, is usually used for realistic visualisations. There are many theories about the nature of light. For our purpose light can be described as energy emitted from light sources in the form of rays, which can be imagined as streams of photons. Each ray is travelling through space until it eventually hits the surface of an object in its way.

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 virtual camera. Each interaction with respective object surfaces is simulated. To reduce the amount of rays to trace, the rays can also be traced backward, thus from the virtual camera to the light source. The combination of both methods is reasonable (Heckbert, 1990).

In SAR simulation the sensor position matches the light source as well as the position of the virtual camera. Therefore, there are no differences between forward and backward ray tracing in the SAR case. Interactions with surfaces are simulated based on the respective material and light properties. Each ray or microwave beam can be absorbed, refracted or reflected, which enables a high-degree of realism in the visualised scene, as long as the material properties are known and the interaction between the surface and the microwave beam can be modelled accurately. Ray tracing is not only based on accurate physical models, it is also easy to implement, but is rather computational time consuming. For each pixel in the resulting image at least one ray has to be traced. Assuming reflections and multiple light sources, the number of rays to trace is increasing accordingly. For real-time applications, ray tracing is usually too slow.

A primitive-by-primitive approach is faster as a pixel-by-pixel approach like ray tracing. Rasterization is a primitive-by-primitive approach and is usually used for real-time visualisations and is using triangles as primitives. Therefore, 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 rasterized and depicted at this transformed position.

Triangles which are calculated first can be overwritten by triangles which are rasterized later, even if the pixel which is rasterized later is occluded in reality. To avoid this, z-buffering is used (Catmull, 1978). In the z-buffer each depth value of an already rasterized pixel is saved. Every new pixel which has to be rasterized, must pass the z-test, thus must be closer to the virtual camera as the pixels already rasterized. The z-buffer should be a 32-bit texture to achieve a more accurate z-test. 16-bit or even 8-bit textures can result in artefacts if two objects are too close together.

Z-buffering is simple and efficient. For rasterization and z-buffering less computational effort is necessary compared to ray tracing. Modern graphic cards support 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 4 and 5.

4. SARVIZ: HARDWARE ACCELERATED SAR-SIMULATION

3D graphic cards are specially designed for the fast rendering of 3D scenes. The hardware implementation of standard visualisation routines guarantees highest performance for these specialised tasks. The visualisation of SAR scenes is not possible using standard visualisation methods, due to the different image geometry. 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.

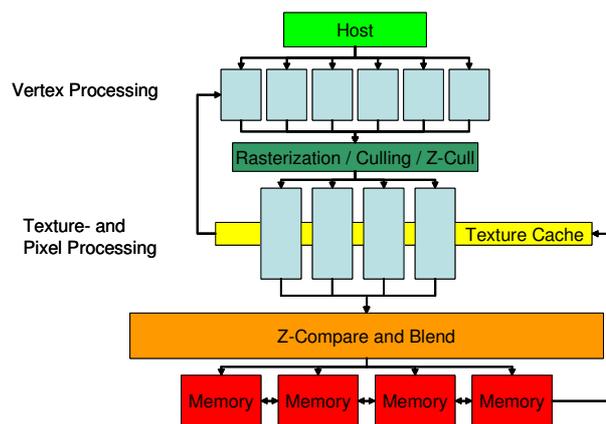


Figure 2. NVIDIA GeForce series 6 architecture (Kilgariff & Fernando, 2005)

In Figure 2, the architecture of NVIDIA's GeForce 6 series is depicted as block diagram. Programming the so-called vertex and pixel shaders, the visualisation can be customized. The real-time SAR simulator SARViz, which is described in this paper, is implemented using Microsoft's DirectX API (Microsoft, 2005). Starting with DirectX 8, the shaders could be programmed using assembler.

Since DirectX 9, the high-level shading language (HLSL) was introduced, which is used by SARViz for GPU programming. SARViz uses slightly modified DirectX meshes as input source. These meshes are directly supported by the DirectX API and a variety of 3D CAD tools can export to this format. The colour information of the meshes is adapted to meet the needs of SAR visualisation. Import tools for other data sources, like for example ArcGIS datasets, integrates the SARViz simulator to the GIS-world.

4.1 SAR rasterization

The GPU is a data-parallel streaming processor working in a single-instruction, multiple data (SIMD) fashion. The GPU allows a massively parallel computing. Each triangle or pixel is visualised independent from any other triangle or pixel. Each triangle point is processed by the so-called vertex shader, which treats the geometry. After the rasterization, the radiometry of each pixel is calculated by the so-called pixel shader.

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 geometry of a SAR image differs from the visible light image geometry. The running-time geometry of a SAR image needs to be implemented in the vertex shader. In SAR images the position in range of each object depends on the distance between the object and 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. This results in a shift in range direction Δx depending on the height above the ground level z and the incidence angle σ :

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

Each triangle processed by the vertex shader is afterwards rasterized. 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 and customized illuminations could not be calculated. In contrast, modern pixel shaders (pixel shader version 3.0 or higher) support branching and more than 512 machine-code commands, which is sufficient even for complex tasks like SAR simulation.

For each pixel the corresponding face normal is known. Taking material properties, like the dielectric constant, and sensor properties into account, reflection strengths can be calculated in the pixel shader.

4.2 Reflection properties

According to the Phong reflection model (Phong, 1975), three illumination elements (diffuse, specular and ambient) are combined for the resulting reflection. Ambient lightning is negligible for SAR simulation. The diffuse illumination element is equivalent to the Lambertian reflection element.

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.

4.3 Data range

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.

4.4 SAR illumination

In the SAR geometry, reflections from different objects 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 parallel pixel shaders accessing the same pixel at the same time can overwrite the results of each other. This could lead to image artefacts or wrong simulation results. Alpha blending avoids these problems. Alpha blending is a way of combining two colours in computer graphics. Alpha blending allows transparency effects, but at the moment the hardware support for 32-bit floating point textures is not sufficient. Furthermore, it is disadvantageous, because in SAR processing the different reflections sum up, which is not as accurately simulated by alpha blending. Two colours sum up fine using alpha blending and an alpha value of 0.5, but three or more equidistant reflections would sum up inconsistently.

4.5 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 in real-time. Because no random values, beside Perlin noise, can be generated on the GPU, the random values are generated by the CPU and stored in a texture. Accessing the random texture, the GPU can calculate speckle noise for pixels with and without dominant scatterers.

For each triangle in the model the value for the dominant scatterer can be set, whereas a value of zero represents a speckling without any dominant scatterer. The integration of different speckle behaviours allows the realistic simulation of different material behaviours related to scattering. This is very important for a realistic SAR simulation.

4.6 SAR shadow

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 (Williams, 1978) both shadows and occluded areas can be modelled.

A shadow map is generated in two steps. First, the scene is rendered from the position of the light source, which is in SAR simulation equivalent to the SAR sensor position. Instead of reflection values, the distance of every rendered pixel to the sensor is copied to the so-called shadow map, as it is depicted in Figure 3.

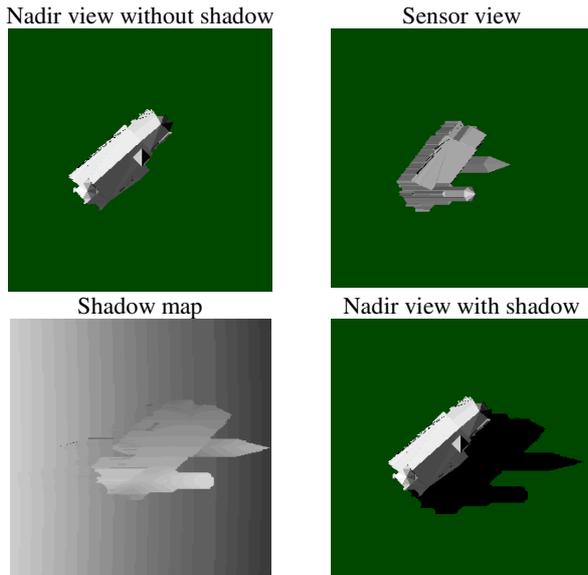


Figure 3. SAR shadow map generation

In the second step the scene is rendered from the position of the virtual camera. SARViz directly simulates ground-range images to avoid the computational intense transformation from slant-range to ground-range. Because of this, the scene is rendered looking from nadir direction. The distance of each pixel rendered in the nadir view is compared to the distance from the sensor 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. Shadow maps are fast and provide real-time shadows. They have problems if the virtual camera is inside a shadow area, or in other problematic constellation between virtual camera and light source. For SAR simulation, the sensor is identical to the light source and the virtual camera, therefore no such problems exist. Still, some precision and aliasing problems may occur while using shadow maps.

4.7 Simulation of irradiation stars

Strong reflecting objects like corner reflectors can cause typical blooming effects in SAR images. The blooming is caused by the high amount of energy reflected to the sensor. Analysing the simulation result, which is first rendered to a 32-bit floating point texture, high-reflecting points can be found. Depending on a threshold, these high-reflecting points are considered to cast irradiation stars, as depicted in Figure 4. Afterwards these stars are rendered to the simulation result.

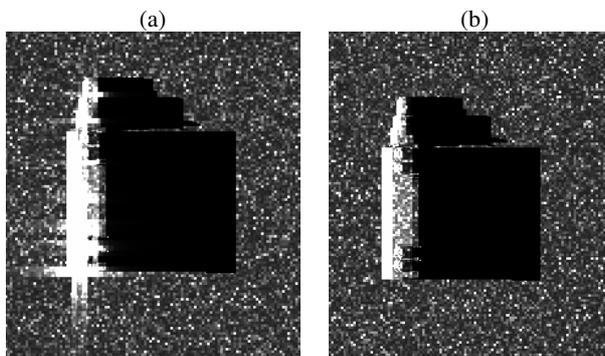


Figure 4. SARViz high-resolution simulation of a truck, with (a) and without (b) the star effect

4.8 Environment maps for double-bounce simulation

Environment maps are used in computer graphics to visualise reflecting surfaces, which nowadays is directly supported by the graphic hardware. For environmental mapping, which is also sometimes referred as cube mapping, a cube texture is created around the reflecting object. The cube contains the scene around the object for the visualisation of the reflections. For this purpose, the scene is rendered six times without the reflecting object, but from the position of the object and looking to different directions. The six sides of the cubes are filled with textures containing the results from this rendering step.

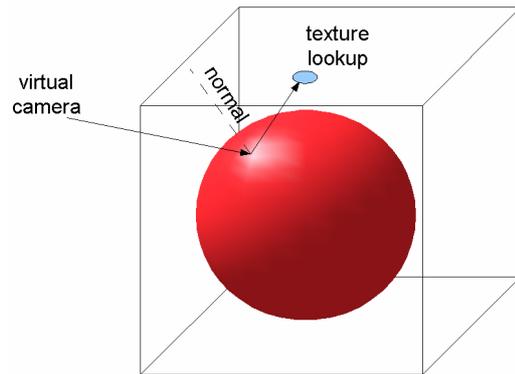


Figure 5. Environment mapping

Based on the environment stored in the cube texture, reflections from the surrounding area can be visualised. In the final rendering step, the cube texture lookup position of a specified pixel is determined by the position of the virtual camera and the normal direction of the object surface, as depicted in Figure 5. No complex multi-bounce reflections are visualised in this way, but reflections can be visualised in real-time.

For SAR simulation, this technique is also applicable, but only double-bounce reflections can be visualised that way. Even worse, only double-bounce reflections between different objects can be visualised. Intra-object reflections are not visualised. A reflection map for every triangle would be a possible way to overcome that problem. But a scene containing thousands of triangles would end up in thousands of rendering steps, which is far too much for real-time applications.

Double-bounce reflections from the ground, for example caused by asphalt surfaces, can be simulated using this technique. For this purpose, not even a real cube has to be simulated, because just one cube side is needed for the environment map, since only the asphalt surface is assumed to reflect.

To render the reflection texture, the scene is rendered with a virtual sensor position regarding the reflecting angle. The image geometry of the double-bounce texture differs from the single-bounce geometry and the different image geometry has to be implemented.

In the final step, the scene is rendered from the sensor position. The double-bounce reflection information is, regarding possible occlusions and shadows, added to the single-bounce reflection. Depending on the roughness of the ground, the intensity of the double-bounce reflection is adapted accordingly.

5. EXAMPLES

As described above, using rasterization for real-time SAR simulation is feasible and delivers promising results, which will be discussed in this section.

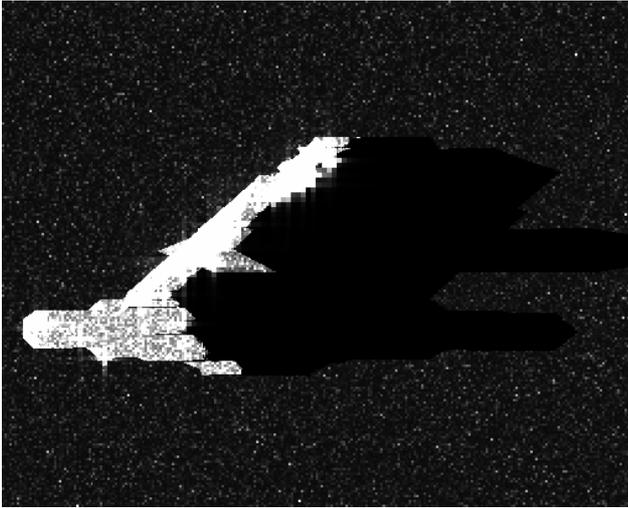


Figure 6: High-resolution SAR simulation of the "Stiftskirche" in Stuttgart (0.3m resolution)

In Figure 6, a simulated SAR image of the "Stiftskirche" in Stuttgart can be seen. The model is the same which has been used for Figure 3. The displacement of the spires due to the SAR image geometry is clearly visible. Simple building models are simulated in about 10 milliseconds. Single buildings are seldom of interest, whereas the simulation of complete urban areas is important for many applications.



Figure 7. SARViz simulation of the 3D city model of Stuttgart (5m pixel resolution)

Simulations of complex urban environments are depicted in Figure 7 and Figure 8. The city model, simulated in Figure 7 and Figure 8, includes 9950 buildings containing 548,729 triangles and can be simulated in about 100 milliseconds using a NVIDIA GeForce 6600 graphic card. This time does not include the time needed to copy the data to the video memory. 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.

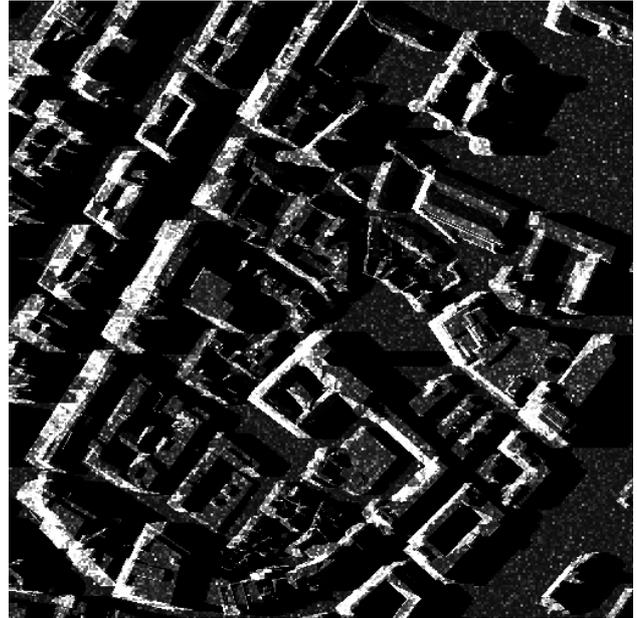


Figure 8. SARViz simulation of area surrounding the "Marktplatz" in Stuttgart (1m pixel resolution)

In Figure 8, 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.

Different material properties influence the SAR simulation results. In Figure 9, a Boeing 747-400 model is simulated two times. One time the model is supposed to have Lambertian reflection (a), for the next simulation the model is more specular (b).

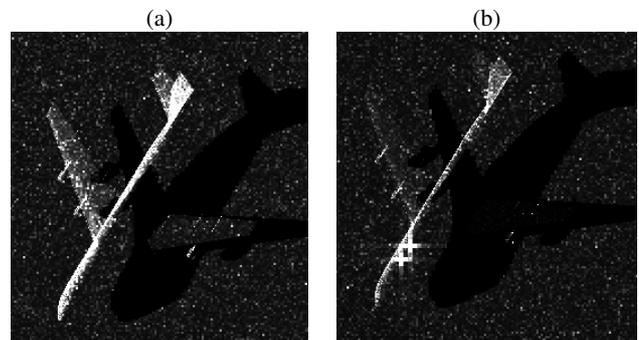


Figure 9. SAR simulation of a Boeing 747-400 model with more Lambertian (a) and more specular (b) reflection properties

The model in Figure 9 (b) is not totally specular, otherwise the wings would not be visible at all, because they would reflect the whole energy away from the sensor. The reflection on the edges is stronger in the specular model. The speckling properties also differ between the two simulations. The Lambertian model is supposed to speckle without a dominant scatterer, whereas the metallic model is simulated with a strong dominant scatterer. Objects also get influenced by surrounding objects. Object parts can get occluded, shadows and reflection may mix. In Figure 10, a high-resolution SAR simulation of such a scene is depicted. The same scene is also depicted in Figure 11 as visible light scene seen from nadir and from the sensor position.

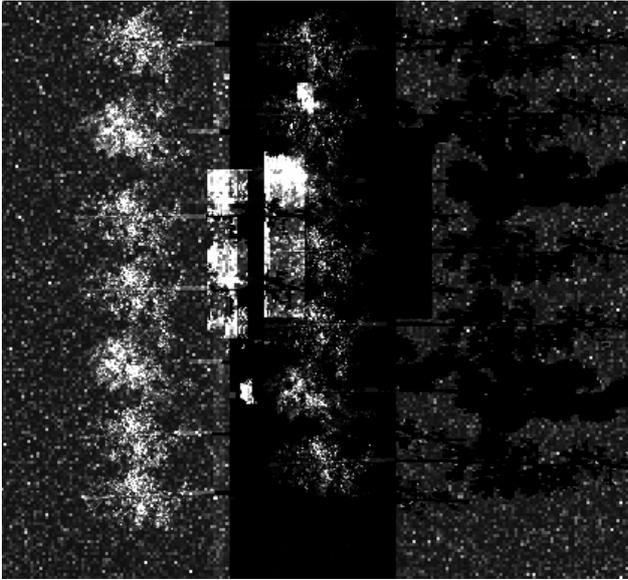


Figure 10. High-resolution SAR simulation of two trucks covered by trees

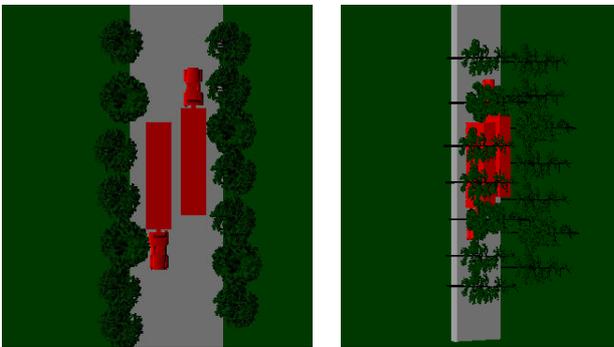


Figure 11. Nadir-view (left) and sensor-view (right) of the scene simulated in Figure 10

As can be seen in Figure 10, the complex interaction of the simulated objects is often not comparable to our everyday visible light experience. SAR systems “see” the world different and a SAR simulator helps understanding the world as it is “seen” by SAR. Using a real-time SAR simulation, the user is able to simulate a variety of different scenes and sensor properties with a few clicks. The real-time capability is important for the usability of SAR simulations for training purposes.

Real-time SAR simulation provides feasible results. Real-time applications like SAR mission planning or object detection based on real-time SAR simulation are possible.

6. CONCLUSION

In our opinion hardware accelerated SAR simulation using techniques originally developed for computer games is a promising technology. The simulation results are convincing and the real-time simulation capability is amazing. Applications based on SAR simulation profit from the tremendous increase in simulation speed, due to the rasterization technique and the hardware acceleration. 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 used by ray tracing simulators. Especially the missing multi-bounce effect is a drawback. Double-

bounce effects are realisable using environmental mapping, but these effects are still not comparable to the results from ray tracing simulations. Multi-bounce effects are not realisable and require ray tracing.

If multi-bounce effects are essential, ray tracing on the GPU as proposed by (Purcell et al, 2002) could be an alternative, as well as hardware accelerated ray tracing proposed by (Schmittler et al, 2002).

However, for many applications fast results and the ability to simulate extensive models including hundreds of thousands of triangles, 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.

REFERENCE

Buck, I., 2005. Taking the Plunge into GPU Computing. In: Pharr, M.: *GPU Gems 2. Programming High-Performance Graphics and General-Purpose Computation*. Boston, pp. 509-519

Catmull, E., 1978. A hidden-surface algorithm with anti-aliasing. In: *ACM SIGGRAPH Computer Graphics, Proceedings of the 5th annual conference on Computer graphics and interactive techniques SIGGRAPH '78*, Atlanta, Georgia

Heckbert, P.S., 1990. Adaptive Radiosity Textures for Bi-directional Ray Tracing. In: *Computer Graphics*, 24 (4), pp. 145-154

Kilgariff, E., Fernando, R., 2005. The GeForce 6 Series GPU Architecture. In: Pharr, M.: *GPU Gems 2. Programming Techniques for High-Performance Graphics and General-Purpose Computation*. Boston, pp. 471-491

Leberl, F.W., Bolter, R., 2001. Building reconstruction from Synthetic Aperture Radar images and interferometry. In: Baltsavias, E.P., Grün, A., Gool, L.v.: *Automatic Extraction of Man-Made Objects From Aerial and Space Images (III)*. Lisse, pp. 281-290

Microsoft, 2005. DirectX 9.0c Software Development Kit. <http://msdn.microsoft.com/directx>

Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E., Purcell, T.J., 2005. A Survey of General-Purpose Computation on Graphics Hardware. In: *Eurographics 2005, State of the Art Reports*, Dublin

Phong, B.T., 1975. Illumination for computer generated pictures. In: *Communications of the ACM*, 18 (6), pp. 311-317

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

Schmittler, J., Wald, I., Slusallek, P., 2002. SaarCOR - A Hardware Architecture for Ray Tracing. In: Ertl, T., Heidrich, W., Doggett, M.: *Graphics Hardware*. Saarbrücken, Germany

Williams, L., 1978. 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