Improved Real-Time SAR Simulation in Urban Areas

Timo Balz, Norbert Haala
Institute for Photogrammetry
Universität Stuttgart
Stuttgart, Germany
{firstname.lastname}@ifp.uni-stuttgart.de

Abstract— **Within the paper the applicability of modern graphic cards for SAR simulation is demonstrated. By these means SAR simulation, which is frequently used as a key tool for the analysis and interpretation of SAR scenes, is feasible in real-time even in complex urban environments. This is realised by the implementation of SAR geometry within standard graphics hardware, which offers 3D hardware acceleration and programmable graphics processing units (GPU).**

SAR;Simulation;Real-Time;Urban Area

I. INTRODUCTION

Frequently, the applicability of SAR simulations is hindered by the considerable amount of time, which is already required to generate a single scene. However, simulations to interactively support the interpretation of SAR images are only useful, if results are available almost in real-time. The same is true for SAR mission planning. Especially in urban areas a considerable number of test scenarios have to be checked in order to avoid or at least reduce occlusions by almost optimal flight and sensor parameters. Because the response time of common SAR simulators is too long, databases containing simulated objects of interests are created, to support object detection and SAR image interpretation. A real-time SAR simulation supersedes these databases.

Such real-time simulation is feasible based on modern graphic cards, which offer 3D hardware acceleration and programmable graphics processing units (GPU). These graphic cards are a standard equipment of 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]. Realizing 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.

II. VISUALISATION TECHNOLOGIES

Real-time visualization of complex 3D scenes is one of the main goals in computer graphics. Visual realism can be e.g. achieved by ray-tracing. There the path of each "ray of light" is traced from the light source to the virtual eye, while each reflection at object surfaces is simulated depending on the respective material properties. Ray-tracing applies accurate physical models and can be implemented easily. Unfortunately it is very time consuming, which prevents this technique from being used in real-time environments.

This type of application is usually based on the so-called rasterization approach. There, triangulated 3D vector models, which represent the scene to be visualized are mapped from world to the screen or image co-ordinate system. Afterwards, each triangle is rasterized and depicted at his image position. In order to handle occlusions a so called z-buffer is used. In this 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 drawn. This technique is simple, efficient and, less computational effort is necessary compared to ray-tracing. Rasterization is supported by hardware acceleration within modern graphic, thus real-time visualization of even complex scenes is feasible. However, since the technique does not really model the physical light transport, scenes with complex reflections can not be simulated accurately. Still, similar to the creation of realistic scenes for computer games, it is possible to simulate realistic SAR images using rasterization at sufficient quality. As it is described in the following section, for this purpose the methods originally developed in computer graphics have to be modified.

III. GPU BASED SAR-SIMULATION

Since hardware accelerated visualization was originally designed for visual light scene generation, the differences with respect to SAR imaging geometry have to be implemented within such a process. This can be realized by means of the programmable graphics processing units (GPU) offered by modern graphic cards. Our real-time SAR simulator SARViz [2] 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, which can be provided by a variety of 3D CAD tools. Alternatively, import tools for other data sources like Arc-GIS datasets are available.

A. SAR rasterization

The GPU is a data-parallel streaming processor working in a single-instruction, multiple data (SIMD) fashion. It allows a massively parallel computing since each triangle or pixel is visualized independent from any other triangle or pixel. The

geometric processing of each triangle or vertex point is provided by the so-called 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.

Since the geometry of a SAR image differs from the visible light it has to be implemented appropriately 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 $\Delta x = z \cdot \tan(\sigma)$ depending on the height above the ground level z and the incidence angle σ

After geometric processing each triangle is rasterized. The resulting pixels are processed by the fragment shader to compute the corresponding radiometry. Calculating the reflection intensity for SAR images is a relative complex task. Since the first programmable fragment shaders were limited to only a few calculations, complex and customized illuminations could not be calculated. In contrast, modern fragment shaders (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 available. Reflection strengths can then be computed in the fragment shader by additionally taking into account the respective material and the sensor properties.

B. *Reflection properties*

According to the Phong reflection model [4], three elements (diffuse, specular, and ambient) are combined for the resulting illumination. 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 modeled.

C. *SAR illumination*

Unlike the projective geometry used for the visualizations of visual light scenes, the SAR image geometry is a distance or running-time geometry in range direction. 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 modeled 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.

D. *Speckle effect*

Speckle is important for the realistic visualization 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 scatterer. 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 behaviors allows the realistic simulation of different material behaviors related to scattering. This is very important for a realistic SAR simulation.

E. *SAR shadow*

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 modeled. By using shadow maps [5] both shadow and occluded areas can be modeled. 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 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.

IV. CREATING 3D-MODELS FOR SAR-SIMULATION IN URBAN AREAS

SAR simulation within urban areas requires the availability of 3D city models as well as digital terrain models (DTM). For this purpose frequently dense 3D point clouds from airborne laser scanning are applied, which can also be used to estimate the roughness of the terrain.

A. *Automated 3D building reconstruction*

The development of tools for the efficient and area covering collection of 3D city models has been a topic of intense research for the past years. A good overview on the current state-of the-art is for example given in [6]. Frequently, these 3D models are used to generate photorealistic visualisations of the urban environment. Typically, this type of application is required in the context of urban planning, tourism or entertainment, like games based on real locations. Meanwhile, a number of algorithms are available, which are usually based on 3D measurements from airborne stereo imagery or LIDAR. Airborne photogrammetric data collection is suitable to efficiently provide a complete set of 3D building models, mainly representing the footprints and the roof shapes of all buildings at sufficient detail and accuracy. For this reason, the production of virtual city models is more and more becoming a standard task of photogrammetric data collection, resulting

in a growing availability of data sets, which include 3D representations of buildings.

B. GPU based LIDAR point triangulation

Alternatively to the explicit generation of 3D CAD models, point clouds from laser scanning can be directly used for geometric surface representation at least for some applications. However, in this scenario the construction of polygonal meshes from the LIDAR points is still required, which can be a relatively time consuming process. Usually the required surface meshing is based on Delaunay triangulation. According to Rong and Tan [7], the required Voronoi diagram can be approximated on the GPU, thus providing a basis for the GPU based Delaunay triangulation.

Another, simpler approach uses the ability of GPUs to visualize large amounts of triangles and provides hardware accelerated z-tests. This can be used for hardware supported triangulation of LIDAR point data, as depicted in Figure 1. The triangulated surface is iteratively compared to the rendered points. Differences between the point heights and the triangle surface heights which exceed a certain threshold, force the triangles to iteratively segment themselves. This triangulation can be calculated fast, especially if a considerable generalization of the surface is aspired. However, the resulting triangles are not shaped optimal, because no optimization criterion is used.

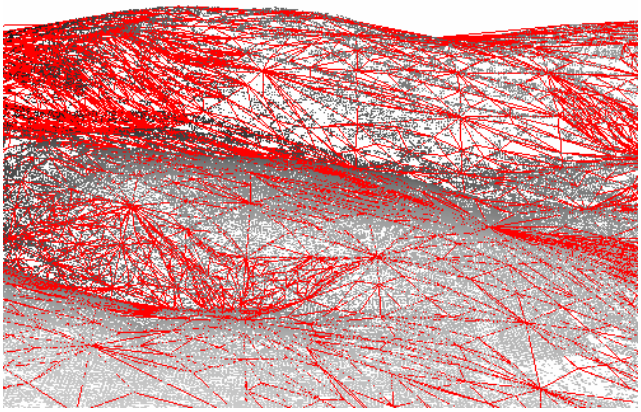

Figure 1. Triangulated LIDAR surface points

C. Estimating the roughness of surfaces using LIDAR

Surface roughness is an important parameter representing material properties as required for SAR simulation. In this context roughness is defined as height variation of the surface compared to the wavelength of the SAR sensor. In principle, this parameter can be derived from LIDAR measurement. Since available point density and accuracy from airborne data collection is limited, such a determination surface roughness should be supported by existing GIS-data. By these means, homogenous areas are already defined, which then are used for the estimation of one common roughness parameter. To allow for stable parameter estimation, the size of the respective areas should be large enough. Alternatively, to GIS-data, homogenous areas for the roughness analysis could be defined by an unsupervised or supervised classification of remote sensing data.

In order to determine the roughness, the LIDAR points should be normalized with respect to a Digital Terrain Model in order to avoid an influence of surface topography during computation of their standard deviation. For plane surfaces, like streets or sports fields, the computed standard deviation of LIDAR points is around 0.1m, in forest areas the value is considerable larger than 1m. Grassland, fields or bush areas can be found in between 0.2m – 1m. Since the LIDAR data contains a lot of noise, this approach is only useful for large wavelength systems like L-band (20 cm) or P-band (1m) systems. Another way to determine the roughness is the comparison between first and last pulse data. Differences between first and last pulse is an indication for vegetation in the area, the absolute difference between first and last pulse indicates, depending on the vegetation and the season, the respective height above ground. Depending on the type of the vegetation, different values for the roughness of the simulated area can be assumed, which can be used to further improve the realism of SAR simulations.

V. EXAMPLES

SAR simulation in urban areas requires high-quality ground-truth data. In principle, this information can be provided with sufficient accuracy from dense point data as they are collected by airborne LIDAR. However, best simulation results are expected using 3D city models as provided from 3D building reconstruction processes.

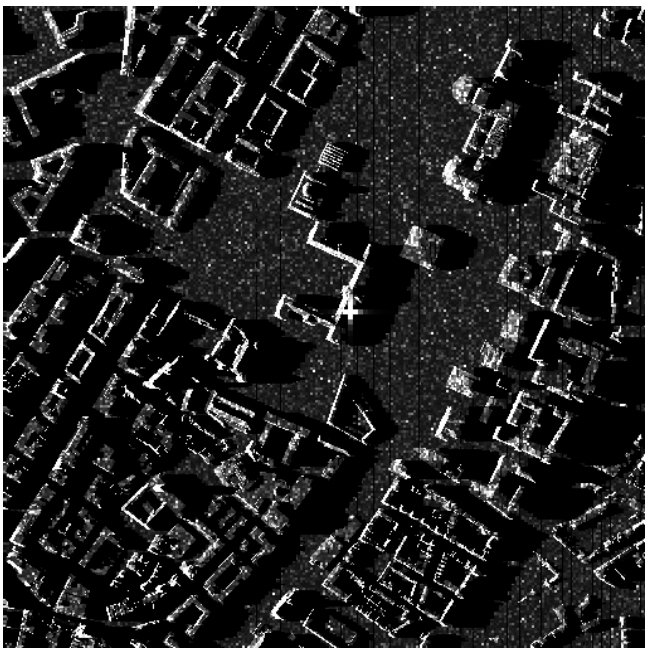

Figure 2. SARViz simulation of the Stuttgart city model around the “Neues Schloss”

In Figure 2 a SAR simulation by SARViz is depicted, which is based on a 3D city model for the city of Stuttgart, Germany. This model consists of 9950 buildings containing 548,729 triangles. The simulation was generated in about 100 milliseconds, not including the time needed to copy the data to

the video memory. Thus, the real-time capability of SARViz could be demonstrated.

The simulation of complex terrain models is also possible using SARViz, as can be seen in Figure 3. These terrain models are also produced from triangulated LIDAR point clouds.

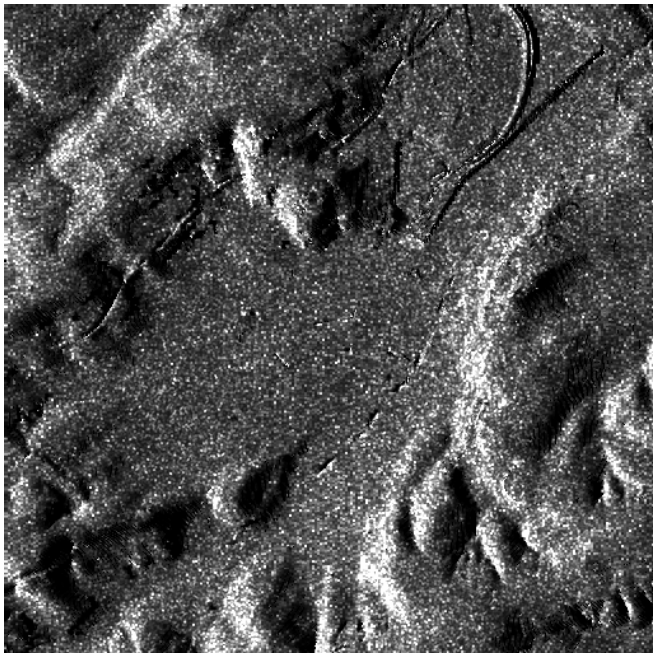

Figure 3. DTM of the valley of Stuttgart simulated using SARViz (10m resolution, 60° off-nadir angle, 4 looks)

All the SAR simulations depicted above always assume the same material properties. The differences in the reflected energy are caused by the topography, not by differences of the surface roughness or the dielectric constant of the materials. But these differences are important for a realistic simulation of SAR images. In Figure 4. a model of the E-3 sentry is simulated assuming different material properties.

Figure 4. SARViz simulation of a E-3 sentry model with Lambertian reflection (left) and more specular reflection properties (right)

The simulation of the model assuming Lambertian reflection properties on the left side is less realistic as the simulation assuming more specular reflection properties. The reflection is still assumed not to be totally specular, otherwise most parts of the wings would not reflect any energy back. Beside the reflection properties, the speckling also differs. The simulation on the left side is simulating the plane model assuming no dominant scatterer behavior, whereas reflection of the model on the right side is assumed to be dominated by one scatterer per resolution cell.

VI. CONCLUSIONS

Real-time SAR simulation using techniques originally developed for computer games is a promising technology. In addition to consumer type graphic cards, no expensive hardware is required, i.e. real-time visualization of SAR images is possible on standard PC hardware. The ease of use and the real-time capability opens up new fields of applications of the SAR simulation.

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. The result of the simulation always depends on the quality and amount of information provided by the input data. Usually, the available data is not perfect and SAR simulations suffer from the lack of information about the material properties as well as from erroneous or unavailable city models.

Using information acquired from LIDAR systems is a way to gather the required information. LIDAR data can be used for the reconstruction of city models, DTM generation and even for analyzing the surface roughness, regarding the reservations mentioned above.

SAR simulations in urban areas can be based on data acquired by LIDAR systems. The real-time SAR simulation is not only able to deliver fast results, furthermore the ability to simulate and visualize huge amounts of 3D data, makes it the best choice for simulating city models. Cities are huge, complex and their models consist of hundreds of thousands of triangles. Slow SAR simulators are therefore unusable in this complex environment, whereas the drawbacks of the rasterization based SAR simulations are acceptable for most real-world applications.

REFERENCES

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

[2] T. Balz: "Real Time SAR-Simulation on Graphics Processing Units", in: Proceedings of EUSAR 2006, Dresden, 2006

[3] Microsoft: "DirectX 9.0c Software Development Kit". 2005

[4] B.T. Phong: " Illumination for computer generated pictures". Communications of the ACM 18, No. 6, pp. 311-317, 1975

[5] L. Williams: "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

[6] Baltsavias, E., Grün, A. & van Gool, L.: *Automatic Extraction of Man-Made Objects From Aerial and Space Images (III)*. A.A. Balkema Publishers:, 2001

[7] G. Rong and T.-S. Tan: "Jump Flooding in GPU with Applications to Voronoi Diagram and Distance Transform", in: Symposium on Interactive 3D Graphics and Games (i3D 2006), Electronic Arts Campus, Redwood City, CA, 2006