AUTOMATIC POINT TARGET DETECTION FOR INTERACTIVE VISUAL ANALYSIS OF SAR IMAGES

Martin Lambers and Andreas Kolb

Institute for Vision and Graphics, University of Siegen, Germany
{lambers,kolb}@fb12.uni-siegen.de

ABSTRACT

Point target analysis is an important tool to analyze the quality of SAR images. To permit interactive visual analysis, visualization applications need to automatically detect point targets in a SAR image and estimate associated quality measurements such as the peak sidelobe ratio (PSLR). This task is computationally expensive.

In this paper, we propose methods for automatic point target detection that work on hierarchical data structures and process the image data on the graphics processing unit (GPU) to allow interactive use. For each detected point target in the currently visualized area of the image, the visualization application can then display color-coded quality measurements, thus providing the user with an overview of the point targets in the scene as well as an immediate impression of the SAR image quality. Detailed point target analysis results can be displayed on demand.

Index Terms— Synthetic aperture radar, Visualization, Image analysis

1. INTRODUCTION

Images from Synthetic Aperture Radar (SAR) systems consist of the amplitude values from single look complex (SLC) data files, which contain information of the combined reflectivity (amplitude and phase) in each resolution cell. Point scatterers in the scene result in characteristic patterns in such SAR images. See Fig. 1 for an example. The analysis of point target responses is an important tool to analyze the quality of the image and the SAR processing technique. Quality measurements associated with a point target response include the estimated width of the mainlobe and the peak sidelobe ratio (PSLR), which measures the ratio of the peak sidelobe amplitude to the peak mainlobe amplitude.

Interactive visualization applications should allow interactive visual analysis of SAR images. For this purpose, the application must automatically detect point targets in the currently visualized area of the image and compute associated quality measurements such as the PSLR. This allows to mark point targets in the image and provide color coded quality information at the same time, thus providing the user with an overview of the point targets in the scene as well as an immediate impression of the image quality. More detailed analysis results, such as those provided by the RAT radar tools [1], can then be displayed if the user clicks on a point target mark.

In this paper, we present methods for automatic point target detection that work on a hierarchical representation of the SAR image and process the image data on the graphics processing unit (GPU) to achieve the processing speed that is necessary for interactive use. The methods are integrated in our GPU-based framework for interactive visualization of SAR images [2].

2. FRAMEWORK

Our GPU-based framework for interactive visualization of SAR images uses a combination of hierarchical data structures and GPU-based data processing to achieve the processing speed that is necessary to allow interactive exploration of large SAR images and provide the user with immediate feedback on changes of visualization parameters [2].

The basic data structure is a tiling pyramid (see Fig. 2). To build a tiling pyramid, the SAR image is divided into tiles of fixed size at different resolution levels. Each higher pyramid level halves the resolution of the image.

To display a region of the SAR image in a view area of a given resolution, only a subset of the pyramid tiles from a single pyramid level is required. The application determines the required subset of tiles from the current view area resolution and the current region of interest (ROI), which is the region of the image that the user currently wants to display, and loads these tiles into graphics memory.

The data is then processed on the GPU according to the current visualization parameters, such as the despeckling method and the method to transfer amplitude values to gray levels [3].

Additional borders around each tile ensure that local neighborhoods (with a fixed maximum size) can be accessed for every relevant pixel of a tile without the need to fetch data from neighboring tiles.

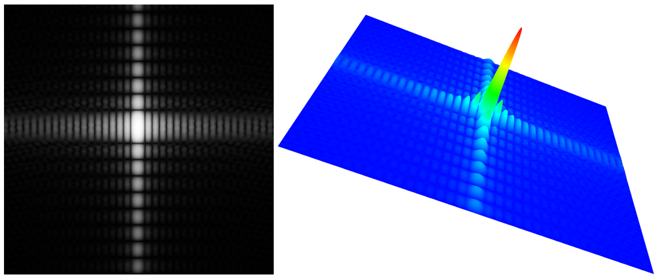


Fig. 1. A typical point target response (simulated).

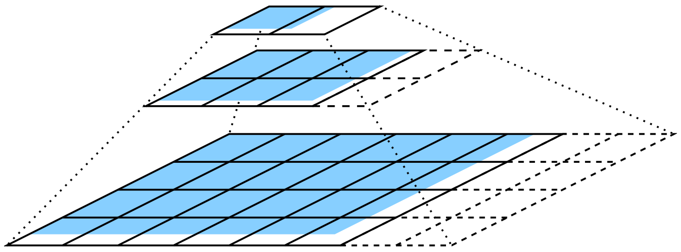


Fig. 2. A tiling pyramid for a 2D data set.

3. AUTOMATIC POINT TARGET DETECTION

The same subset of tiles that is used to display the current ROI is also used to detect point targets, in order to reduce the computational costs associated with point target detection. As a consequence, point target detection methods must be able to handle tiles at different resolution levels.

The required tiles are already in GPU memory, and the search for point targets is a highly parallelizable task that is well suited to be executed on the GPU. Point target detection methods must examine each pixel in an input tile together with its local neighborhood, and decide whether the pixel is the center of a point target or not. Using the GPU's fragment shader to make this decision, the results can be written to an output tile of the same dimensions [2]. Each pixel in this output map then indicates the presence or absence of a point target center at the respective position. The value zero means that there is no point target center. A value greater than zero can additionally store properties of the point target that were detected during its examination.

Often the sidelobes of a point target extend in azimuth and slant direction, orthogonal to each other, but this is no general rule, since the angle between them depends on the squint angle [4]. To reliably detect point targets, the directions in which the sidelobes extend have to be estimated.

For this purpose, we use 12 fixed masks of size $7{\times}7$, as shown in Fig. 3. The masks represent the angles $0°, \ldots, 165°$ in steps of $15°$. The masks that generate the two highest values likely correspond to the sidelobe directions. The masks are generic enough to work for point targets of different sizes and in different pyramid resolution levels. However, at high pyramid levels, when the ROI spans a large area of the original SAR image which is then displayed in a very low resolution, point targets may become undetectable. They will only appear once the user zooms in to specific regions.

For each pixel in a tile, the following series of tests is performed to determine whether it is the center of a point target:

1. If the amplitude value is lower than 15% of the maximum amplitude value in the SAR image, then it is considered too low. A zero is written to the output tile, and no further tests are performed.

2. If the amplitude value is not the maximum value in its $7{\times}7$ neighborhood, then it cannot be the center of a point target. A zero is written to the output tile, and no further tests are performed.

3. Mean values are computed using the 12 masks for the directions $0°, \ldots, 165°$. The two highest mean values whose directions differ by at least $45°$ are chosen. If one of these two values is lower than a threshold t multiplied with the arithmetic mean of all amplitude values in the $7{\times}7$ neighborhood, then the pixel is not considered to be a point target center, and a zero is written to the output tile. Otherwise, the two direction estimates (from $\{1, \ldots, 12\}$) are encoded in an 8bit value using 4 bits for each direction. This value is then written to the output tile, which therefore only needs to provide 8bit of information per pixel.

The result of this decision step is a map of point target locations for the currently processed tile. This map must then be transformed into a list of point targets for further processing.

One way to do this is to read the map back to main memory and scan it using the CPU. However, this approach is too slow to be used in interactive applications because large amounts of data have to be transferred and scanned.

In the context of general purpose computations on GPUs (GPGPU), the task of selecting a subset of elements from a data stream is called stream filtering [5]. Because the number and location of elements to be filtered is not known a priori, stream filtering was a complex problem on previous generations of GPUs and required multiple render passes [6].

The latest generation of GPUs includes a geometry shader unit. This unit can be used to perform stream filtering on the GPU in a straightforward and parallelizable manner [7]: the geometry shader scans the input data stream and, using OpenGL's transform feedback extension, writes one geometric primitive per filtered element to an output buffer. To parallelize the task, the geometry shader can be started multiple times, each time for a different subregion of the input data.

This technique allows to transform the point target map to a list of point targets directly on the GPU, without the need to transfer the map. The resulting reduced data set is then transferred to main memory for further examination.

1	1	1	1	1	1	1
2	2	2	2	2	2	2
4	4	4	4	4	4	4
2	2	2	2	2	2	2
1	1	1	1	1	1	1

					1	1
		1	1	1	2	2
1	1	2	2	2	4	4
2	2	4	4	4	2	2
4	4	2	2	2	1	1
2	2	1	1	1		
1	1					

				1	1	2
			1	2	2	4
	1	1	2	4	4	2
1	2	2	4	2	2	1
2	4	4	2	1	1	
4	2	2	1			
2	1	1				

				1	2	4
			1	2	4	2
		1	2	4	2	1
	1	2	4	2	1	
1	2	4	2	1		
2	4	2	1			
4	2	1				

Fig. 3. The detection masks for 0°, 15°, 30°, and 45°. The masks for the directions 60°, . . . , 165° are analogous.

4. POINT TARGET ANALYSIS

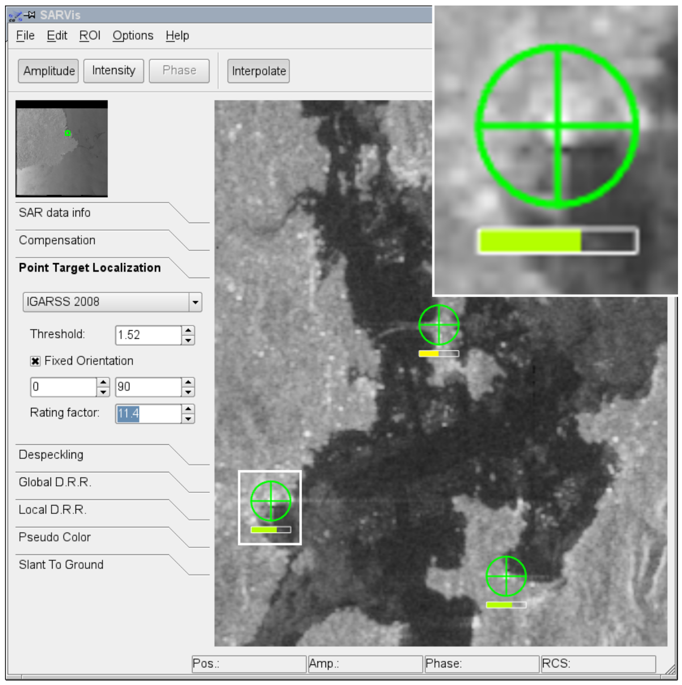
The result of the point target detection step is a list of point targets in the current ROI. Each point target in this list has to be examined in more detail in order to compute associated quality measurements. Since typically not too many point targets are visible in a scene at one time, this task can be performed on the CPU.

As a first step, the original amplitude data at the point target location is gathered from the lowest pyramid level, to have access to the full original information. This data is then upscaled for further processing, using a magnification factor of 10 and bicubic interpolation. This resampled neighborhood is then recentered around its maximum value, which does not necessarily lie in the middle of the neighborhood due to interpolation effects.

The sidelobe directions of the point target were estimated on the GPU in steps of 15°. These coarse estimates can now be used as starting points for a refinement process. Profiles through the data can be computed using angles from -8° to +8° around each coarse estimate, in steps of 1°. The amplitude values along these profiles are summed up, and the angle that is associated with the highest amplitude sum is taken as the refined direction. Alternative methods are described in [4].

Once profiles in the refined sidelobe directions are available, some points of interest are computed both in left and right direction for each profile:

- The first point where only half of the amplitude is left compared to the mainlobe peak. This point can be used to compute the -3db width of the mainlobe.
- The first local amplitude minimum. This point can be used to compute an alternative width value for the mainlobe.
- The first local amplitude maximum, which is the peak of the first sidelobe. This value is used to compute the PSLR.



IGARSS 2008
Threshold: 1.52
☒ Fixed Orientation
0 90
Rating factor: 11.4

Fig. 4. Screenshot of the application. The detected point targets are marked with a circle, a cross that indicates the profile directions (0° and 90°), and a color coded quality measurement. The upper right part of the image shows a closeup of the area enclosed in the white rectangle.

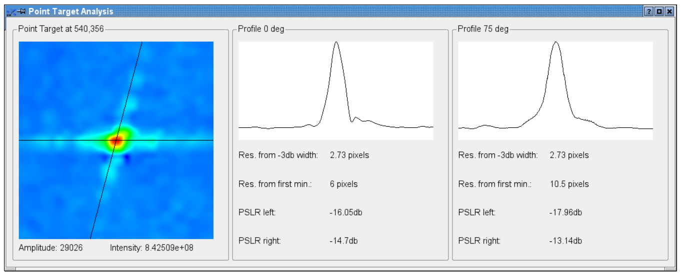
All of the computed data is stored in a cache so that it does not have to be recomputed when a point target re-enters the ROI at a later time.

5. INTERACTIVE VISUAL ANALYSIS

At this point, the application has a list of point targets in the current ROI, and it has access to detailed information about each of them.

With this information, it can mark the point targets and display color coded quality measurements for them. An example is shown in Fig. 4. Each point target is marked with a circle. The cross indicates the sidelobe directions, in this case 0° and 90°. The bar below the circle indicates the mean PSLR of the point target: a full green bar indicates a high PSLR, and a nearly empty red bar indicates a low PSLR.

This information provides the user with a quick overview of point targets in the ROI as well as their main characteristics.



Amplitude: 29026   Intensity: 8.42509e+08

Profile 0 deg

```
Res. from -3db width:   2.73 pixels
Res. from first min.:   6 pixels
PSLR left:              -16.05db
PSLR right:             -14.7db
```

Profile 75 deg

```
Res. from -3db width:   2.73 pixels
Res. from first min.:   10.5 pixels
PSLR left:              -17.96db
PSLR right:             -13.14db
```

Fig. 5. Example of a more detailed point target analysis, displayed on user demand. In this example, the sidelobe directions are 0° and 75°.

A click in a point target circle opens a window that displays the cached analysis results in more detail. The example shown in Fig. 5 displays the results for a point target with the profile directions 0° and 75°.

If the sidelobe directions of point targets in the SAR image are known in advance, this information can be used to further speed up the detection and analysis steps: only two 7×7 masks have to be used for detection, and the direction refinement step can be omitted from the analysis.

The threshold parameter t steers the sensitivity of the detection process. Higher values result in less detected point targets. In our tests, a value of $t = 1.5$ delivered good results.

6. CONCLUSION

Interactive visual point target analysis provides the user with an overview of the point targets in the visualized scene as well as an immediate impression of the SAR image quality and/or the quality of the SAR processing technique.

The techniques for automatic detection of point targets described in this paper are implemented in our GPU-based framework for interactive visualization of SAR data [2].

Results show that real-time detection and analysis of point targets is possible when using hierarchical data structures and the data processing features of the latest generation of GPUs.

Acknowledgement

This project is partially funded by grant KO-2960-3/1 from the German Research Foundation (DFG).

7. REFERENCES

[1] A. Reigber and O. Hellwich, "RAT (radar tools): A free SAR image analysis software package," in *Proceedings of EUSAR*, 2004, pp. 997–1000.

[2] M. Lambers, A. Kolb, H. Nies, and M. Kalkuhl, "GPU-based framework for interactive visualization of SAR data," in *Proc. Int. Geoscience and Remote Sensing Symposium (IGARSS) 2007*, July 2007.

[3] M. Lambers, H. Nies, and A. Kolb, "Interactive Dynamic Range Reduction for SAR Images," *Geoscience and Remote Sensing Letters*, 2008, accepted for publication.

[4] Z. Fan and H. Wen, "Analysis of squint angle in point target assessment," in *Int. Conf. on Radar 2006. CIE '06.*, 2006, pp. 1–4.

[5] John D. Owens, David Luebke, Naga Govindaraju, Mark Harris, Jens Krüger, Aaron E. Lefohn, and Timothy J. Purcell, "A survey of general-purpose computation on graphics hardware," *Computer Graphics Forum*, vol. 26, no. 1, pp. 80–113, 2007.

[6] D. Horn, *GPU Gems 2*, chapter Stream Reduction Operations for GPGPU Applications, pp. 573–589, Addison-Wesley, 2005.

[7] F. Diard, *GPU Gems 3*, chapter Using the Geometry Shader for Compact and Variable-Length GPU Feedback, pp. 891–907, Addison-Wesley, 2007.