GPU Accelerated Textons and Dense SIFT Features for Human Settlement Detection from High-Resolution Satellite Imagery

D. R. Patlolla[1], S. Voisin[1], H. Sridharan[1], A. M. Cheriyadat[1]

[1]Oak Ridge National Laboratory,
Email: {patlolladr, voisins, sridharanh, cheriyadatam}@ornl.gov

Abstract

Automated analysis of large-scale high-resolution satellite imagery requires computationally efficient image representation techniques that characterize the visual content of a scene. The computational process involved in feature descriptor generation is often expensive and its scalability to large image databases forms an important research problem. This paper presents an overview of our work on exploiting the Graphics Processing Unit architecture for careful implementation of two different feature representation techniques – (i) Textons and (ii) Dense Scale Invariant Feature Transform. We evaluate the performance of our implementation for human settlement detection on an image database consisting of high-resolution aerial scenes representing diverse settlements. The rapid computation and robust detection accuracy of our experiments suggest that this High Performance Computing based framework has unique capabilities for Peta-scale production of high fidelity human settlement maps.

Keywords: High Performance Computing, Settlement Mapping, High-Resolution Satellite Imagery.

1. Introduction

The profusion of high-resolution satellite imagery allows daily update of the Earth's surface providing an overwhelming amount of data to monitor changes of land-cover and land-use. In order to extract relevant information it is critical to automate the analysis of this large-scale high-resolution satellite imagery using computationally efficient image representation techniques that characterize the visual content of a scene. The computational process involved in feature descriptor generation is often expensive and its scalability to large image databases forms an important research problem. This paper presents an overview of our work on leveraging the Graphics Processing Unit **(**GPU**)** architecture for the implementation of two different feature representation techniques – (i) Textons and (ii) Dense Scale Invariant Feature Transform **(**DSIFT**)**. We evaluate the performance of our implementation for human settlement detection on an image database consisting of high-resolution aerial scenes representing diverse settlements.

2. Feature Descriptors

The general workflow of the settlement mapping process is provided in Figure 1. Interested readers are directed to refer Patlolla et al. (2012) for further details. As shown in the figure, generating robust feature descriptors is a key component of the framework.

In this work, we focus on two important texture-based feature representations, namely Textons and DSIFT. Textons are basic atomic elements of visual perception and are obtained by convolving images with a set of filter banks (Leung and Malik, 2001). DSIFT descriptors, on the other hand, are obtained by convolving images with a flat 2D gradient filter and measuring 8-directional histogram of the gradient orientation. Both are computationally expensive processes requiring an HPC based approach.

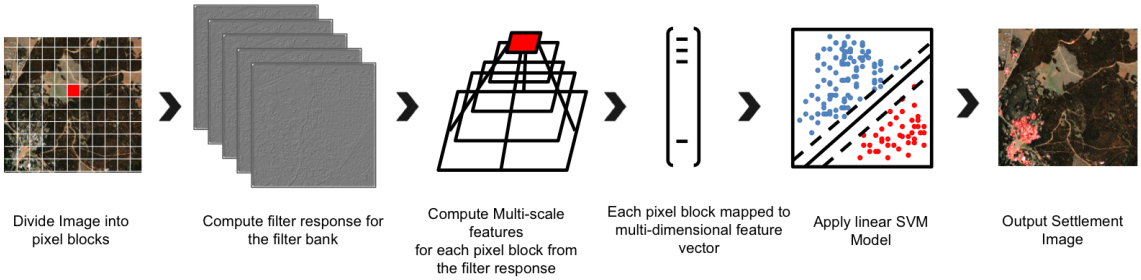

Figure 1. Overview of the Settlement Mapping Process

3. GPU Implementation

3.1 Textons

The most compute intensive part of the process involves the computation of filter-responses by applying the 48 L-M filters on the imagery. Today, commodity-GPUs provide hundreds of cores offering ample opportunities to accelerate algorithms. This section briefs our approaches to filter-response computation and their performance.

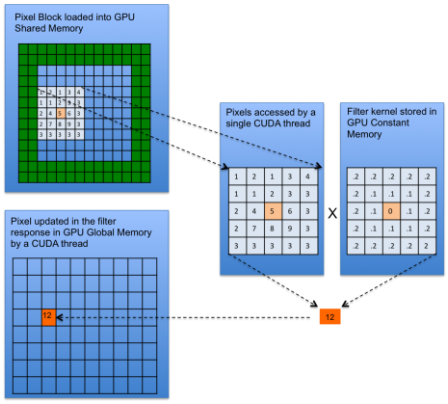

Figure 2. GPU implementation of Filter Convolution

A naïve approach of the convolution process on a GPU is depicted in Figure 2. With the image raster on the global memory, the subset window around a pixel in shared memory, a block of the Compute Unified Device Architecture (CUDA) threads can process a block of the image with each thread computing the element-wise multiplication of its corresponding pixel and the surrounding pixels with the filter. This requires the boundary pixels for the pixels that are on the edge of the block. One can utilize extra threads/block, to load the boundary pixels, but these threads could otherwise be used for

achieving better parallelism. From the overall perspective, a naïve approach to convolution can only provide limited speedups.

Separable Filters

If a filter kernel can be separable, it can be decomposed as a product of two vectors (one column vector and one row vector) (Rigamonti, 2013). The implementation of convolution using separable filters on the GPU provides us good parallelism and is divided into two parts (Podlozhnyuk, 2007). The first consists of applying the row vector with the pixel block along with the boundary pixels on the left and right, and the second consists of the column vector with the pixel block and the boundary pixels at top and bottom as shown in Figure 3. The reduction in redundant accesses of image data by each thread combined with modifications to the thread block size and the pixels processed by each thread resulted in 40x speedup.

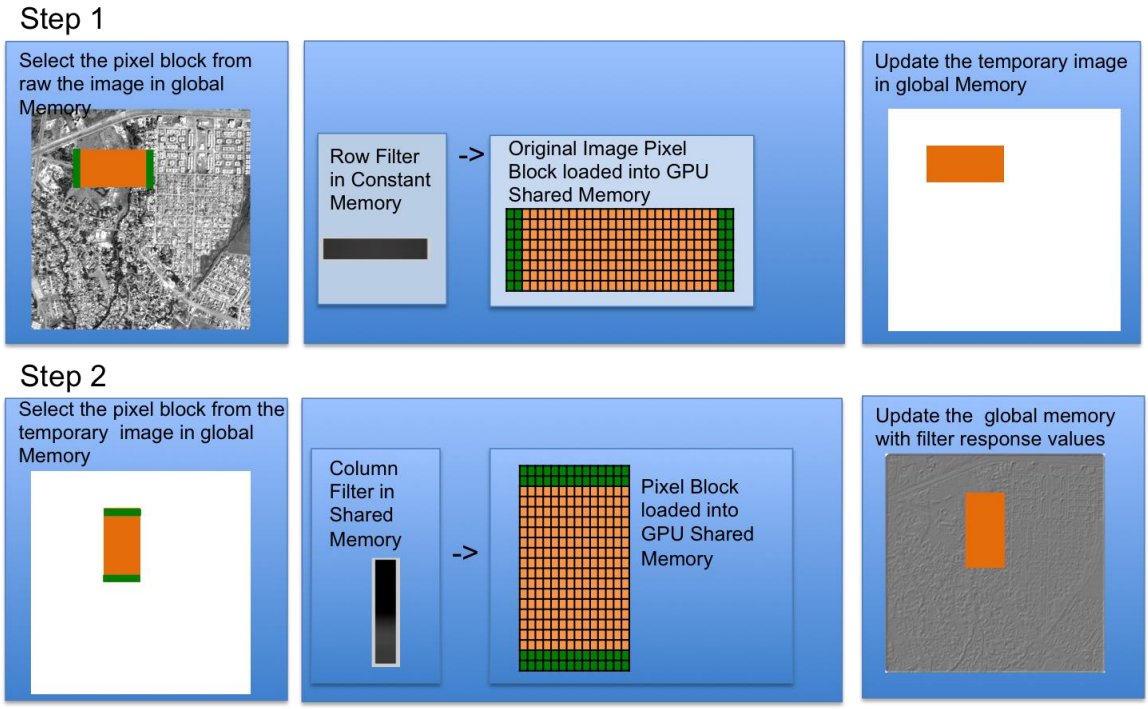

Figure 3. GPU Implementation of Separable Filters

Image Rotation

If the filter at 0 degree orientation is linearly separable, then the filter-response of the same filter at other orientations (which are non-separable) can be computed by rotating the image to as depicted in Figure 4. This achieves a 20x speedup, but is still not an ideal option considering the number of memory accesses required to access the original and re-oriented image in the memory during reorientation and convolution.

Experiments were conducted assuming that all the filters are linearly separable. This delivered 40x speedup without affecting the final accuracy. A final increase in speedup was achieved by utilizing CUDA Streams, which provide the opportunity to execute multiple CUDA operations simultaneously. Since, the computation of filter-response for

each of the filters is not interdependent, theoretically all the 48 filter-responses can be computed simultaneously, but are limited by CUDA resources (threads, shared memory etc.). We utilized CUDA Streams to compute filter-response kernels into different streams. This provided better speedups as shown in Table 1.

CUDA Streams	*1*	*2*	*3*	*4*
Speedup	-	1.7x	2.1x	2.1x
Time(ms)	280	165	130	130

Table 1. Filter Response Times and Speedup with different CUDA Streams

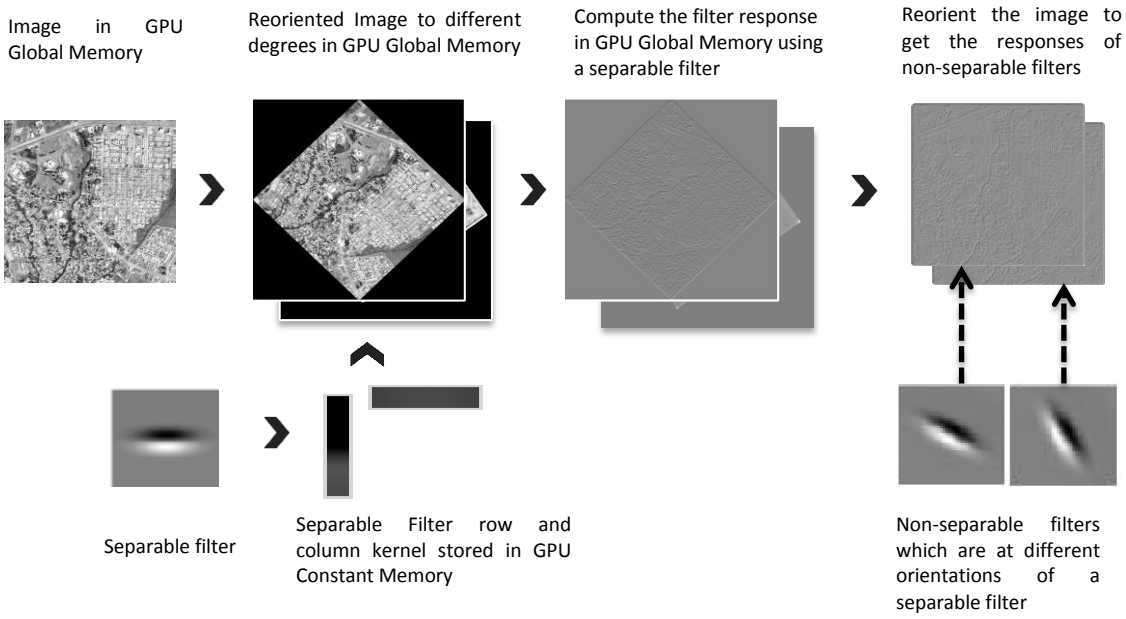

Figure 4. GPU Implementation of Image Rotation

The computation of multiscale Texton features is implemented on GPU using similar techniques presented in Patlolla et al. (2012).

3.2 DSIFT

The variant of the well-known SIFT features (Lowe 2004), the DSIFT has been demonstrated by Fei-Fei and Perona (2005) to work well for scene classification. The DSIFT descriptor computation requires 4 different steps (Figure 5) that have been implemented using CUDA. First, the angle and norm of the gradient for each pixel is computed using one thread per pixel. During this computation step, an 8-value vector (one value per orientation) is computed for each pixel and separately stored in global memory in 8 matrices (one per orientation). Note that the shared memory is used to compute the intermediate results: x- and y-gradient values, angle, norm, and the index of the first orientation. In the second step, the 8 matrices are convoluted with a separable 2D-filter, x- and y-basis are computed sequentially, similarly to the Textons computation.

The third step applies a weight to each of the 4x4 cells. Sixteen threads per block are used, with one per weight coefficient. The last step mass normalizes the descriptor values using shared memory common to 128 threads (one per value) to perform the summation using the reduction algorithm.

The cluster assignment is computed using shared memory and 64 threads for each descriptor. We selected 64 threads to balance active and idle status of the threads while computing the Euclidean distance (reduction algorithm). From this bag-of-words representation (Sivic 2003), feature vectors are created for each 16x16 pixel block. These feature vectors are the combination of 32-bin histograms for 5 scales.

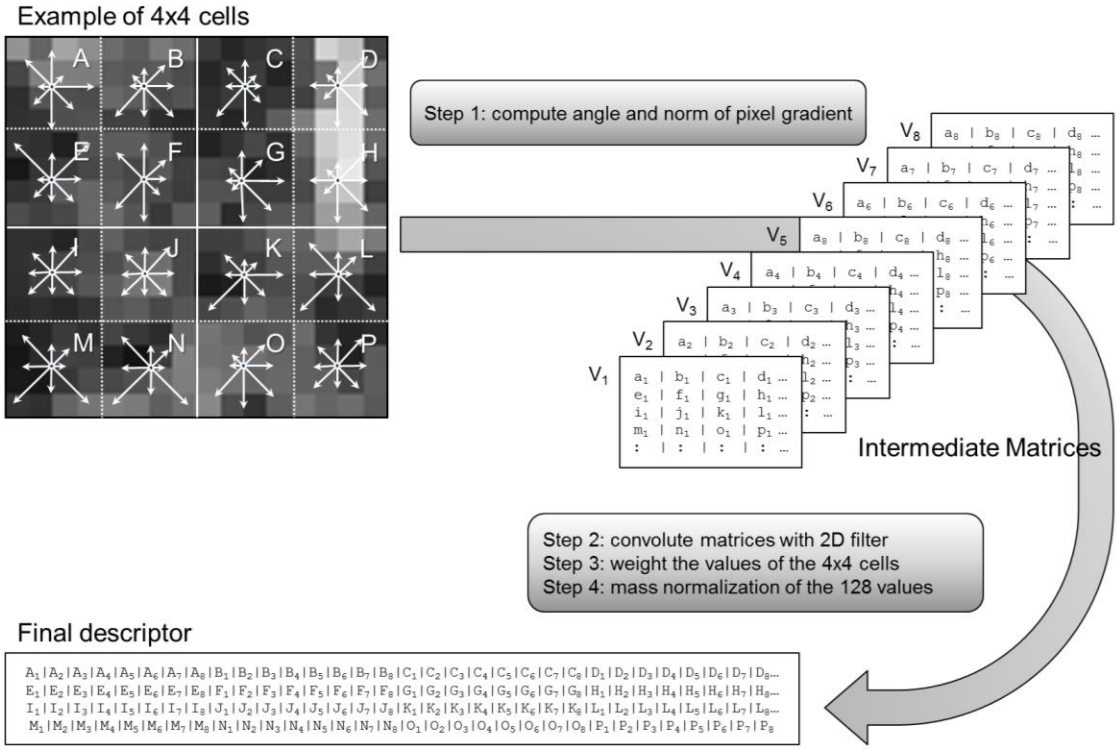

Figure 5: DSIFT Computation Steps

4. Performance

The performance speed and classification accuracy of our implementation was assessed on a benchmark dataset of 33 images of 0.5 m spatial resolution, each covering an area of 2.6 km^2, collected from various parts of Kandahar, Afghanistan. The average computational time for each component of the feature computation is provided in Table 2. For each image, a corresponding reference image is obtained manually. The dataset is split into two sets (set 1 and set 2) and a two-round cross-validation is performed. The average accuracy measures from the two rounds for each feature descriptor are presented in Table 3.

	Textons	DSIFT
Filter-Response (ms)	24.9	181.677
Cluster-Assignment (ms)	387.26	489.73
5 scale Feature-Computation (ms)	5.33	99.91
Overall-process (ms)	510	1200.44

Table 2: Breakdown of Computational Time

Feature	Overall Accuracy	User's Accuracy	Producer's Accuracy
Textons	92.68±1.32	73.38±3.81	81.57±9.68
DSIFT	90.88±1.96	68.41±1.81	78.61±0.24

Table 3: Performance of the Textons and DSIFT features in settlement detection

A typical settlement output from the designed framework is shown in Figure 6. A 0.6m spatial resolution image covering an area of 74 km^2 is processed in 85 seconds using Textons. The processing time includes input/output data transfer, image reading and writing.

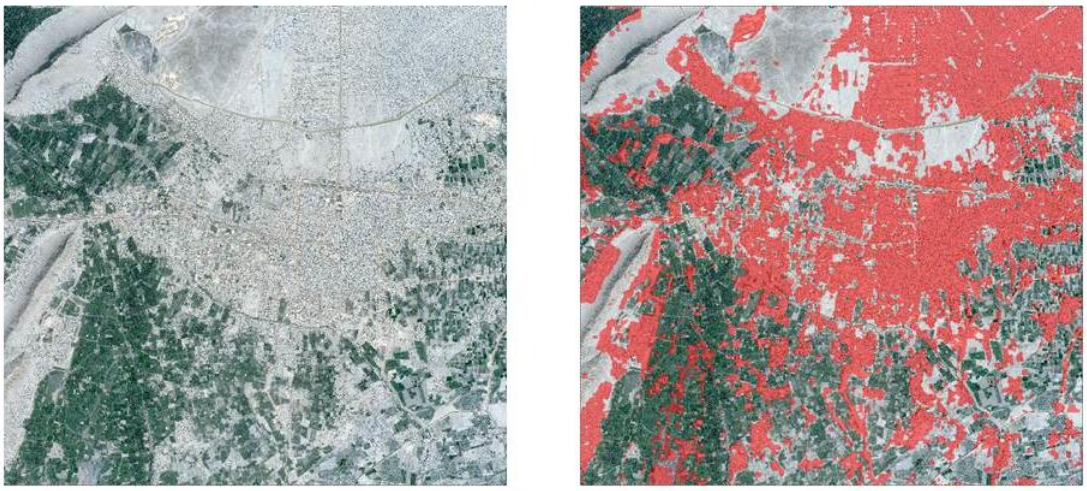

Figure 6. Settlement Detection using Texton Features.

5. Conclusion

With the rapid evolution of high-resolution remote sensing big data, there is a growing demand for high performance computing based approaches to various image processing tasks. In this research we have presented an overview of a GPU based implementation of two important texture features and demonstrated their usefulness in the context of large-scale human settlement mapping.

6. Acknowledgements

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

7. References

Fei-Fei L and Perona P, 2005, A Bayesian Hierarchical Model for Learning Natural Scene Categories. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, San Diego, CA, USA, 524-531.

Leung T. and Malik J., 2001, Representing and recognizing the visual appearance of material using three-dimensional Textons, *International Journal of Computer Vision*, 43(1), 29-44.

Lowe D, 2004, Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60(2):91-110.

Patlolla D. R., Bright E.A., Weaver, J.E. and Cheriyadat, A.M., 2012, Accelerating satellite image based large-scale settlement detection with GPU. *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, 43-51.

Podlozhnyuk V., 2007, Image convolution with CUDA. *NVIDIA Corporation white paper*, June 2007.

Rigamonti R., Sironi A., Lepetit V. and Fua P., 2013, Learning separable filters. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

J. Sivic and A. Zisserman, 2003 Video google: A text retrieval approach to object matching in videos. *IEEE International Conference in Computer Vision, ICCV*, Nice, France, 1470–1477