Parallel Implementation of a Hyperspectral Data Geometry-Based Estimation of Number of Endmembers Algorithm

Sergio Bernabe[1], Gabriel Martin[2], Guillermo Botella[1], Manuel Prieto-Matias[1] and Antonio Plaza[3]

[1]Complutense University, Madrid, Spain
[2]Instituto de Telecomunicações, Instituto Superior Técnico, Lisbon, Portugal
[3]Hyperspectral Computing Laboratory, University of Extremadura, Cáceres, Spain

ABSTRACT

In the last years, hyperspectral analysis have been applied in many remote sensing applications. In fact, hyperspectral unmixing has been a challenging task in hyperspectral data exploitation. This process consists of three stages: (i) estimation of the number of pure spectral signatures or endmembers, (ii) automatic identification of the estimated endmembers, and (iii) estimation of the fractional abundance of each endmember in each pixel of the scene. However, unmixing algorithms can be computationally very expensive, a fact that compromises their use in applications under real-time constraints. In recent years, several techniques have been proposed to solve the aforementioned problem but until now, most works have focused on the second and third stages. The execution cost of the first stage is usually lower than the other stages. Indeed, it can be optional if we known a priori this estimation. However, its acceleration on parallel architectures is still an interesting and open problem. In this paper we have addressed this issue focusing on the GENE algorithm, a promising geometry-based proposal introduced in.[1] We have evaluated our parallel implementation in terms of both accuracy and computational performance through Monte Carlo simulations for real and synthetic data experiments. Performance results on a modern GPU shows satisfactory 16x speedup factors, which allow us to expect that this method could meet real-time requirements on a fully operational unmixing chain.

Keywords: Hyperspectral imaging, estimation of number of endmembers, hyperspectral unmixing, high performance computing (HPC), commodity graphics processing units (GPUs)

1. INTRODUCTION

Many hyperspectral analysis methods have been proposed using high performance computing (HPC) techniques[2] in the last decade. Target detection and identification for military purposes, location of infected trees in forestry, monitoring of oil spills, wildfire tracking are just some examples of the application areas.

More concretely, many methods are focused on solving the main problem using hyperspectral imaging: "mixed pixels". To solve this problem, spectral unmixing[3] is one of the most challenging task explored by the community. This process comprises three stages: 1) estimation of the number of pure spectral signatures (endmembers); 2) automatic identification of the estimated endmembers, and (3) estimation of the fractional abundance of each endmember in each pixel of the scene. Very few methods have been proposed for the first stage, being virtual dimensionality (VD)[4] algorithm and the hyperspectral subspace identification algorithm (HySime)[5] two of the most popular methods. The acceleration of both methods on different architectures has been extensively studied by previous work.[6–8]

The geometry-based estimation of number of endmembers algorithm (GENE)[1] has been recently proposed as a promising alternative to VD and HySime. However, to the best of the authors' knowledge, its parallel implementation has not been studied yet. In this paper we have explored this performance optimization on CUDA-enable GPUs. Experimental results conducted on a heterogeneous platform equipped with an NVidia GeForce GTX 980 GPU and an Intel Xeon E3-1226 processor reveal satisfactory performance results despite the data transfers overheads between the GPU and the CPU memories.

Real-Time Image and Video Processing 2016, edited by Nasser Kehtarnavaz, Matthias F. Carlsohn, Proc. of SPIE Vol. 9897, 989708 · © 2016 SPIE
CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2227910

The rest of this paper is organized as follows: Section 2 details the GENE method. Section 3 describes the proposed parallel implementation. Section 4 presents experimental results of the proposed implementation in terms of both accuracy and parallel performance using real and synthetic data sets on different heterogeneous platforms. Finally, Section 5 presents a few concluding remarks and possible future research lines.

2. GENE METHOD

The GENE method makes use of the key geometric characteristics of the hyperspectral data. The original proposal[1] introduced two different methods denoted as GENE-CH and GENE-AH respectively. The former assumes the presence of pure pixels on the data whereas the later do not make this assumption. In this work we have focused on GENE-AH (see Algorithm 1) since it is more robust than GENE-CH against the absence of pure pixels in the data. The GENE method works with the dimension-reduced hyperspectral data, due to (i) under the LMM assumption it preserves the true information about the mixing process, (ii) it reduces the complexity of the unmixing process and (iii) it reduces the impact of the noise in the hyperspectral data. The dimensionality reduction proposed in[1] assumes that the statistics of the noise are known. In this work we have used the multiplied-regression-analysis-based noise covariance estimation method reported in[5] as suggested by the authors of GENE. The dimensionality reduction is similar to the computation of the principal components of the data but removing the noise statistics of the data from the covariance matrix. The GENE algorithm may be used in combination with any endmember extraction method, but the authors propose a geometry based endmember extraction algorithm called TRI-P. This algorithm may be used with any norm, but in this work we have implemented the TRI-P algorithm with the ℓ_2 norm. In this case TRI-P is similar to the well-known ATGP algorithm. Like ATGP, TRI-P identifies endmembers iteratively: on every iteration the algorithm identifies a potentially new endmember, which GENE tries to decompose into a linear combination of the previously identified endmembers. By using the residual error of the decomposition and the noise statistics of the data, the algorithm performs a Neyman-Pearson classifier rule to determine if the dimensionality of the new subspace is higher that the previous iteration, i.e. the output of the Neyman-Pearson classifier is used as GENE stopping criteria.

3. PARALLEL IMPLEMENTATION

Although the full GENE-AH method can be implemented on the GPU, we have opted to used a hybrid approach that only uses the GPU to accelerate the main parts of the algorithm. We have identified those stages with a profiling of an optimized serial GENE-AH implementation based on the BLAS library. These stages are highlighted in green in Algorithm 2. As described below, some of them can be efficiently implemented on CUDA-enabled GPUs using the cuBLAS library,[9] whereas others require handcrafted cuda kernel code.

3.1 CUDA implementation

Once the hyperspectral image $\mathbf{X}$ is uploaded into the GPU memory, the first stage computes the noise estimation. Algorithm 2) shows a pseudo-code of this step, highlighting in green the computations that are mapped onto the GPU using the cuBLAS matrix multiplication (*cublasDgemm*) kernel. The other parts of this step do not have enough data parallelism and are performed on the CPU with negligible performance degradation.

After noise estimation, an ad-hoc kernel implemented in CUDA computes the mean value of each band of $\mathbf{W}$ using a reduction process and subtract it to all the pixels in the same band. This step has been optimized using shared memory and coalesced memory accesses. The resulting matrix ($\mathbf{W}$ - $\bar{\mathbf{W}}$) is used to finalize the calculation of the covariance matrix $\mathbf{C}_w$ by means of a matrix multiplication operation $(\mathbf{W} - \bar{\mathbf{W}})\ (\mathbf{W} - \bar{\mathbf{W}})^T$ which is mapped on the GPU using the *cublasDgemm* function. The same strategy is performed later to compute the covariance matrix $\mathbf{C}_x$. The singular value decomposition (SVD) is performed very efficiently on the CPU, but the $\tilde{\mathbf{X}}$ matrix is computed on the GPU using the cuBLAS matrix multiplication. $\tilde{\mathbf{C}}_w$ is again computed on the CPU using the BLAS *dgemm* functions because we are operating with small data structures.

The next step of the algorithm uses the ATGP method. For the first endmember, we need to calculate the brightest pixel in the reduced image, $\tilde{\mathbf{X}}$. For this purpose, we have developed a kernel to compute the dot product between each pixel vector and its own transposed version and another kernel to retain the pixel that results in

Algorithm 1 : Pseudocode of the GENE AH algorithm

INPUT: $\mathbf{X}, \mathbf{W}, N > 3, P_f \geq 0$
% $\mathbf{X}$ is $L \times n_s$ matrix with the hyperspectral data set, where L is the number of spectral bands and n_s is the number of pixels.
% $\mathbf{W}$ is $L \times n_s$ matrix with the noise estimation from the scene.
% N and P_f are the maximum number of estimated pixels and false-alarm probability values.
OUTPUT: $\widehat{\mathbf{M}}, \widehat{p}$
% $\widehat{\mathbf{M}}$ is $L \times p$ matrix with the spectral signatures for each endmember.
% $\widehat{p}$ is the estimated number of endmembers.
$\mathbf{C}_w := (\mathbf{W} - \bar{\mathbf{W}})(\mathbf{W} - \bar{\mathbf{W}})^T / n_s$
% $\mathbf{C}_w$ is $L \times L$ matrix with the covariance from the noise estimation.
$\mathbf{C}_x := (\mathbf{X} - \bar{\mathbf{X}})(\mathbf{X} - \bar{\mathbf{X}})^T / n_s - \mathbf{C}_w$
% $\mathbf{C}_x$ is $L \times L$ matrix with the covariance from the hyperspectral data.
$\mathbf{U} := \mathbf{U}\Sigma\mathbf{U}^T \equiv \mathbf{C}_x$ {SVD decomposition}
$\tilde{\mathbf{X}} := \mathbf{U}(\mathbf{X} - \bar{\mathbf{X}})$ {Reduced image}
$\tilde{\mathbf{C}}_w := \mathbf{U}\mathbf{C}_w\mathbf{U}^T$
$k := 1$
$\widehat{\mathbf{M}}_k := \tilde{\mathbf{X}}_{l_k}$ for $l_k \in \arg\max_i \|\tilde{\mathbf{X}}_i\|_2$ {First iteration selects the pixel with higher ℓ_2 norm.}
$\mathbf{Q} := \widehat{\mathbf{M}}_k$
for $k = 2$ to N **do**
  $\widehat{\mathbf{M}}_k := \tilde{\mathbf{X}}_{l_k}$ for $l_k \in \arg\max_i \|\mathbf{P}^{\perp}_{\mathbf{Q}}\mathbf{X}_i\|_2$ {ATGP iteration}
  $\theta := \arg\min_{1^T_{k-1}\theta=1} \|\widehat{\mathbf{M}}_k - \mathbf{Q}\theta\|_2^2$ { SCLSU algorithm over the selected endmember}
  $\mathbf{e} := \widehat{\mathbf{M}}_k - \mathbf{Q}\theta$ {Compute the residual error}
  $\mathbf{r} := \mathbf{e}^T((1 + \theta^T\theta)\tilde{\mathbf{C}}_w)^{-1}\mathbf{e}$ {Neyman-Pearson classifier}
  $\psi := 1 - \frac{\gamma(r/2,(N-1)/2)}{\Gamma((N-1)/2)}$
  if $\psi > P_f$ **then**
    $\tilde{\mathbf{M}} := \mathbf{Q}$
    $\widehat{p} := k - 1$
    EXIT
  end if
  $\mathbf{Q} := [\mathbf{Q}, \widehat{\mathbf{M}}_k]$
end for

Algorithm 2 Pseudocode of noise estimation

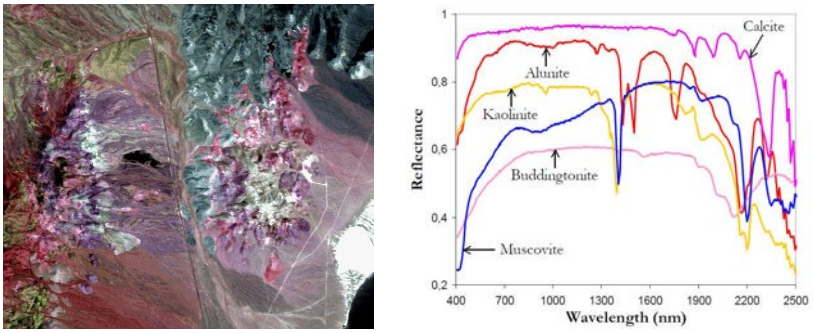
INPUTS: X
$\mathbf{Z} \leftarrow \mathbf{X}^T$, $\widehat{\mathbf{R}} \leftarrow \mathbf{Z}^T\mathbf{Z}$
$\mathbf{R}' \leftarrow \widehat{\mathbf{R}}^{-1}$
for $i = 1$ to L **do**
  $\widehat{\beta}_i \leftarrow ([\mathbf{R}']_{\alpha_i,\alpha_i} - ([\mathbf{R}']_{\alpha_i,i}[\mathbf{R}']_{i,\alpha_i}) / [\mathbf{R}']_{i,i})[\mathbf{R}']_{\alpha_i,i}$ % Note that α_i = [1, . . . , i-1, i+1, . . . , L]
end for
$\mathbf{W} \leftarrow \mathbf{Z} - \mathbf{Z}\widehat{\beta}$
OUTPUT: W
% $N \times L$ matrix with the estimated noise

the maximum projection value (for more details, see the parallel implementation described in [10]). Then, we need to compute the most orthogonal vector to those obtained thus far. For this part, we have used a kernel to calculate the projection for each pixel and another kernel to obtain the maximum of all them. Finally, we have observed that the remaining steps (i.e., SCLSU and Neyman-Pearson test for estimation of the number of endmembers) can be computed very fast in the CPU.

4. EXPERIMENTAL RESULTS

4.1 Real and Synthetic Data Sets

The first data set used in our experiments is the well-known AVIRIS Cuprite scene [see Fig. 1(a)], which has been widely used to study the accuracy of unmixing algorithms. This data set is available online in reflectance units after atmospheric correction. The portion used in our experiments corresponds to a 350 × 350 pixels subset with a spatial resolution of 20-m pixels, which comprises 188 spectral bands in the range from 0.4 to 2.5 μm and a total size of around 46 MB. The site is well understood mineralogically, and has several exposed mineral of interest including *alunite*, *buddingtonite*, *calcite*, *kaolinite* and *muscovite*. The GENE algorithm has been assessed in this work using the reference ground signatures of the above minerals displayed in Fig. 1(b), which are available in the United States Geological Survey (USGS) library *.



(a) (b)

Figure 1. (a) False color composition of the AVIRIS hyperspectral over the Cuprite mining district in Nevada. (b) U.S. Geological Survey mineral spectral signatures used for validation purposes.

We have also considered a synthetic data set using a set of 30 signatures from the USGS library and the procedure described in[11] to simulate natural spatial patterns, composed by a total size of 400 × 500 pixels and a total size of around 179.2 MB. Fig. 2 displays a color-scale composition and three examples of ground-truth abundance maps from the simulated image.

4.2 Work Environment

The performance evaluation has been obtained on a multicore heterogeneous system equipped with a 4-core Intel Xeon E3-1226 v3 at 3.30 GHz processor with 32GB of DDR3 RAM memory and an NVidia GeForce GTX 980† with features such as 2048 cores operating at 1.126 GHz and 4 GB of GDDR5 memory.

4.3 Accuracy Evaluation

For illustrative purposes, Table 1 provides the values of $\widehat{p}$ (number of endmembers) estimated by the GENE method for the two considered hyperspectral data sets using different values of the false-alarm probability (P_f) and N.

As shown by Table 1, the number of endmembers estimated by GENE is $\widehat{p}$=30 and $\widehat{p}$=27 for the Synthetic and the Cuprite scenes respectively. For Cuprite, this value can be compared with another algorithms, such as VD[4] ($\widehat{p}$=28 and P_f=10^{-2}) and HySime[5] ($\widehat{p}$=16). In real scenes, the estimated value is similar compared with the previous solutions. However for the Synthetic scene, the obtained value by GENE is accurate because this image has been generated using 30 endmembers.

*http://speclab.cr.usgs.gov/spectral-lib.html

†http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-980/specifications

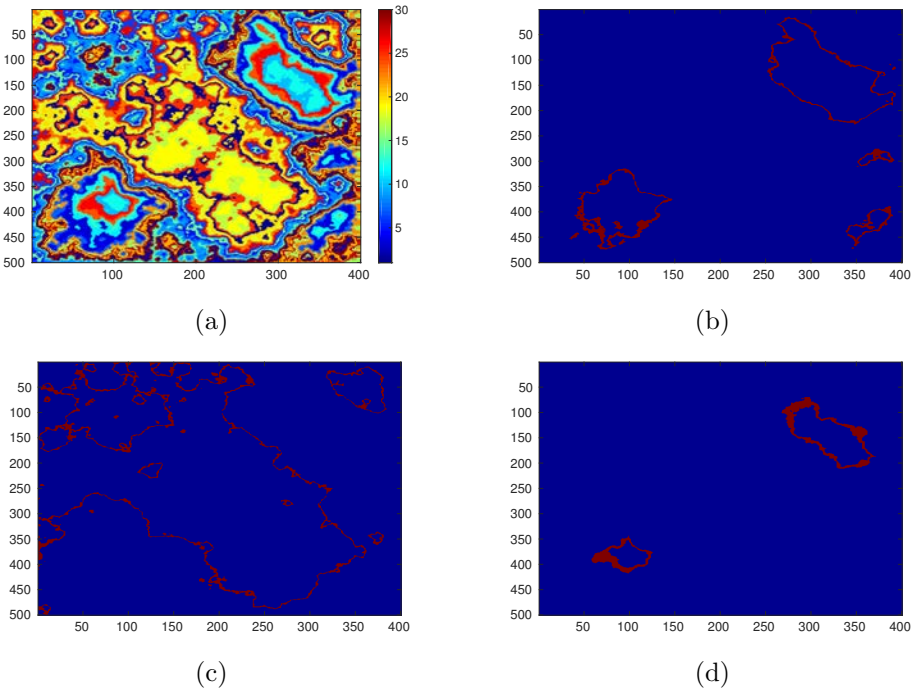


Figure 2. (a) Color-scale composition and three examples of ground-truth abundance maps of endmembers in the synthetic hyperspectral data. (b) Endmember #3. (c) Endmember #15. (d) Endmember #26.

Table 1. GENE estimates of $\widehat{p}$ on real and synthetic data sets for various false-alarm probabilities (P_f) and maximum number of endmembers, N.

P_f	N	AVIRIS Cuprite	Synthetic
10^{-1}	45	43	31
10^{-2}	45	41	31
10^{-3}, 10^{-4}	45	40	30
10^{-5}	45	40	29
10^{-6}	45	40	29
10^{-7}	45	39	29
10^{-7}	45	39	28
0	45	39	28
10^{-1}	43	40	**30**
10^{-2}	43	39	29
10^{-3}, 10^{-4}	43	38	29
10^{-5}, ..., 10^{-7}	43	38	28
10^{-8}	43	34	28
0	43	34	28
10^{-1}	40	37	29
10^{-2}, 10^{-3}	40	36	29
10^{-4}, ..., 10^{-8}	40	36	28
0	40	36	28
10^{-1}, ..., 10^{-3}	30	30	28
10^{-4}, ..., 10^{-8}	30	30	27
0	30	27	27
10^{-1}, ..., 10^{-3}	29	29	27
10^{-4}, ..., 10^{-8}	29	27	27
0	29	**27**	27
10^{-1}, ..., 10^{-8}	28	28	26
0	28	28	26

4.4 Performance Evaluation

Firstly, it is important to emphasize that our parallel version provides exactly the same results as our single-threaded BLAS optimized implementation. Both the parallel and the CPU implementations were compiled using the GNU `gcc`-4.9.2 compiler with the `-O3` optimization flag. Note that for the CPU implementation, only one of the available CPU cores was used. 10 runs were performed for each experiments and we have reported the mean

value (the execution times were very similar executions with differences on the order of a few milliseconds).

Table 2 shows a breakdown of the execution time. *Initialization* includes the overhead of loading the image and allocating memory, *Mem. Copies* accounts for data transfers overheads between the main memory and the GPU dedicated memory. The rest of the table shows the execution times of the different stages of GENE, whereas the *Total time* summarizes the execution time including the *Inizialization* overhead. Overall, the observed speedups factors are significant for both scenes reaching a satisfactory 16x acceleration for the synthetic case.

Table 2. Processing times (in seconds) and speedups achieved for the proposed implementations in different platforms based on the Algorithm 1 and tested with real and synthetic data sets.

	AVIRIS Cuprite		**Synthetic**	
	Serial CPU	**GPU**	**Serial CPU**	**GPU**
Initialization	0.0513	0.5828	0.1494	0.8691
Mem. copies	0.0275	0.1290	0.0530	0.2574
Lines 3-4	14.0584	0.3710	32.6711	0.8186
Lines 5-7	0.5086	0.0545	1.4617	0.1404
Lines 8-10	0.0029	0.0005	0.0156	0.0008
Line 12	1.6423	0.1292	6.7814	0.5163
Line 13	0.0027	0.0050	0.0039	0.0065
Lines 14-16	0.0001	0.0002	0.0003	0.0003
Lines 17-22	≈ 0.0000	≈ 0.0000	≈ 0.0000	≈ 0.0000
Total Time	16.2988	1.2722	41.1364	2.6094
Speedup	–	12.81x	–	15.76x

To conclude this section, we emphasize that our reported GENE processing times are strictly in real-time performance for both data sets. The cross-track line scan time in AVIRIS, a push-broom instrument, is quite fast (8.3 ms to collect 512 full-pixel vectors). This introduces the need to process the AVIRIS Cuprite scene (350×350 pixels and 188 spectral bands) in less than 1.98s and the Synthetic data set (400×500 pixels and 224 spectral bands) in less than 3.24s in order to achieve real-time performance. As noted in Table 2 both reported times are bellow 1.98 and 3.24 seconds to fully achieve real-time performance. These promising results allow us to expect that this method could meet real-time requirements on a fully operational unmixing chain, which is one of the subjects of our future work.

5. CONCLUSION AND FUTURE LINES

In this work, we have explored and evaluated the parallel implementation of a recent alternative to estimate the number of endmembers known as GENE that shows promising results compared to the popular VD and HySime methods. Our implementation has been written mixing ad-hoc CUDA kernels and the `cuBLAS` library. As expected the speedup factor grows with the image size but even for the smallest scene investigated, the speedup factor has been high-enough to meet real-time requirements. As future work, we plan to map GENE-AH into other computing platforms using the OpenCL standard and to generalize our evaluation with important issues such as power-performance ratios and performance portability issues.

6. ACKNOWLEDGEMENT

This work has been supported by the Formación Posdoctoral programme (FPDI-2013-16280), and by the Portuguese Science and Technology Foundation under Project SFRH/BPD/94160/2013. Funding from the Spanish Ministry of Economy and Competitiveness (MINECO) through the research contracts TIN2012-32180 and TIN2015-65277-R are also gratefully acknowledged.

REFERENCES

1. A. Ambikapathi, T.-H. Chan, C.-Y. Chi, and K. Keizer, "Hyperspectral data geometry-based estimation of number of endmembers using p-norm-based pure pixel identification algorithm," *IEEE Transactions on Geoscience and Remote Sensing* **51**(5), pp. 2753–2769, 2013.

2. A. Plaza, Q. Du, Y.-L. Chang, and R. King, "High performance computing for hyperspectral remote sensing," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **4**(3), pp. 528–544, 2011.
3. J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing: geometrical, statistical, and sparse regression-based approaches," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **5**(2), pp. 354–379, 2012.
4. C.-I. Chang and Q. Du, "Estimation of number of spectrally distinct signal sources in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.* **42**(3), pp. 608–619, 2004.
5. J. M. Bioucas-Dias and J. M. P. Nascimento, "Hyperspectral subspace identification," *IEEE Transactions on Geoscience and Remote Sensing* **46**(8), pp. 2435–2445, 2008.
6. S. Sanchez and A. Plaza, "Fast determination of the number of endmembers for real-time hyperspectral unmixing on GPUs," *Journal of Real-Time Image Processing* **9**, pp. 397–405, 2014.
7. C. Gonzalez, S. Lopez, D. Mozos, and R. Sarmiento, "A novel FPGA-based architecture for the estimation of the virtual dimensionality in remotely sensed hyperspectral images," *Journal of Real-Time Image Processing* , pp. 1–12, 2015. DOI 10.1007/s11554-014-0482-2.
8. C. Gonzalez, S. Lopez, D. Mozos, and R. Sarmiento, "FPGA implementation of the Hysime algorithm for the determination of the number of endmembers in hyperspectral data," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **8**(6), pp. 2870–2883, 2015.
9. NVidia, "CUBLAS Library User Guide." http://docs.nvidia.com/cublas/index.html, Oct. 2012.
10. S. Bernabe, S. Lopez, A. Plaza, and R. Sarmiento, "GPU implementation of an automatic target detection and classification algorithm for hyperspectral image analysis," *IEEE Geoscience and Remote Sensing Letters* **10**(2), pp. 221–225, 2013.
11. G. S. Miller, "The definition and rendering of terrain maps," in *ACM SIGGRAPH Computer Graphics*, **20**(4), pp. 39–48, ACM, 1986.