International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:12, 2014

Real-Time Visualization Using GPU-Accelerated Filtering of LiDAR Data

Sašo Pečnik, Borut Žalik

***Abstract*—**This paper presents a real-time visualization technique and filtering of classified LiDAR point clouds. The visualization is capable of displaying filtered information organized in layers by the classification attribute saved within LiDAR datasets. We explain the used data structure and data management, which enables real-time presentation of layered LiDAR data. Real-time visualization is achieved with LOD optimization based on the distance from the observer without loss of quality. The filtering process is done in two steps and is entirely executed on the GPU and implemented using programmable shaders.

***Keywords*—**Filtering, graphics, level-of-details, LiDAR, real-time visualization.

I. Introduction

ONE of the popular remote sensing technologies used to capture information about the Earth's surface is Light Detection and Ranging (LiDAR) [1]–[3]. LiDAR is an active remote sensing technology based on laser light. Laser pulses are emitted towards a target, and each interaction with an object reflects a portion of the initial pulse energy to the sensor. For every one of these reflections, the time between sending and receiving of the laser pulses is measured. Using the speed of light constant, the distance between the sensor and the point of reflection can be calculated. The LiDAR system can be attached to various vehicles such as cars, trains, or aircraft. In combination with the Differential Global Positioning System (DGPS), the locations of every discrete return can be georeferenced [4]. LiDAR provides a huge amount of 3-Dimensional (3D) point clouds without any topology.

LIDAR point clouds hold valuable information not only within the spatial distribution of the points but also in their various attributes. However, LIDAR data are rarely used directly for visual interpretation, since for most users the 3D point cloud is difficult to understand compared to optical imagery. The specifics of LiDAR technology inspire new visualization techniques, allowing interactive 3D interpretation but typically only one attribute at a time. This results in a large number of points that crowd the scene and often obscure details or important 3D cues.

This paper presents a real-time visualization technique, capable of presenting LiDAR data grouped in layers using GPU. This paper is organized as follows: a brief overview of the visualization techniques is presented in the next section. Section III explains our data organization using the GPU filtering technique and the rendering process that is able to visualize data in real-time. The results are presented in Section IV, whilst Section V concludes the paper.

II. Related Work

Most of the real-time visualization techniques use hierarchical space subdivision for efficient scene representation. The general idea of a hierarchical model was introduced by Clark back in 1976 [5]. Clark provided a meaningful way of varying the amount of detail within a scene in regard to the screened areas occupied by objects within a scene, and to the speed at which objects or the camera moves. Then Schachter [6] discussed the need for optimizing the number of graphic primitives representing a scene, and stated that it was common to display objects in less detail when they appeared to be further away. The first applications using such Level of Details (LOD) technique for optimizing graphic work-load were flight simulators.

Several Discrete LOD (DLOD) techniques have been proposed over past decades [7]–[10]. It is typical for DLOD techniques to simplify a 3D object into a number of objects with different detail levels. Then the proper LOD of the object is selected according to the calculated distance from the viewpoint. Falby et al. [11] used the DLOD technique for managing massive terrain datasets using a terrain-paging algorithm to manage the swapping of visible terrain tiles. They used three different dataset resolutions and displayed the terrain within a resolution, depending on the distance.

The concept of using points as rendering primitives for representing an object was introduced due to the pioneering work by Levoy and Whitted [12]. The same concept is used in a real-time visualization of LiDAR data, where two different LOD criteria were evaluated to find the best terrain representation in terms of visual perception [13].

Realistic rendering of massive LiDAR point clouds in real-time is a hard challenge. A two-pass point-based rendering technique that uses elliptical weighed average filtering for solving problems relating to aliasing was introduced in [14]. Recently, a web-based LiDAR visualization has been proposed using point-based rendering [15]. They used an efficient data organization within a data structure and data compression that enabled quick handling of range and LOD queries.

III. Visualization

The fundamental goal of visualization is to convert data into

Sašo Pečnik is with the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia (phone: +386 2 220-7439; fax: +386 2 220-7272; e-mail: saso.pecnik@um.si).
Borut Žalik is with the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia (e-mail: borut.zalik@um.si).

a graphical representation that enables the end-users to understand the data. The massive point clouds provided by LiDAR technology represents a significant technical challenge for visualizing, assessing, and interpreting these data. A typical LiDAR point cloud is saved within a widely used binary LIDAR data exchange format (LAS) [16]. Each point's record in the LAS file is attributed by intensity, color, classification, etc. but without any topological information among individual points. Due to the nature of the data we deal with each point as a graphical primitive instead of linking them with triangles or quads. Although that implementation using such display primitives is fast and robust, the amount of data can still exceed the graphical card's limitations. Indeed, recent graphical cards have been incapable of rendering such amounts of geometric primitives with high frame rates. Consequently, they cannot show a whole dataset within a real-time interaction. Therefore, an efficient data management is needed. An efficient visualization should also provide the mechanism for either querying or filtering irrelevant data.

In the continuation we describe an efficient data organization that arranges LiDAR data into layers, then a GPU filtering technique, and finally a rendering process is presented that allows real-time presentation of layered LiDAR data.

A. Data Organization

The basis for fast and effective visualization of 3D point clouds is hierarchical space partitioning, where data are divided into smaller segments. Although we are dealing with 3D point cloud data, a quadtree data structure has been applied, since LiDAR data describing terrain is considered as 2.5D data. The quadtree's root domain is aligned with the principal axes and covers the bounding box of the whole point cloud that is then halved by four children into equal quadrants. By inserting point after point into the corresponding node of the tree, nodes are recursively split starting at the root until the leaves do not contain more than a predetermined maximal number of points. The geometry is stored and randomly sampled within the graphic memory using vertex buffer objects (VBO), which speeds up the rendering process. This also reduces the usage of the main memory to only upkeep point indices and VBO references. The point quadtree is created and prepared for rendering during a preprocessing phase.

In order for better interpretation of LiDAR data, individual points are colored according to various attributes contained within point records of the LAS file. These color data are also stored in the VBO, besides the geometry, in order to speed-up the whole rendering process. Coloring individual points can significantly improve the user's perception of LiDAR data by showing clear boundaries between objects. Nevertheless, such visual interpretation is still a hard challenge [13]. Therefore, we organize points within the graphic memory using an additional attribute to achieve better user interpretation of LiDAR points. Data are organized into layers according to the classification attribute within the LAS point record.

The additional attribute is stored within the VBO to avoid extra time for copying data between the main and the graphic memory when the rendering process is running. In order to store this data into the graphic memory we exploit homogeneous coordinates, where a Cartesian point (x, y, z) is represented by (xw, yw, zy, w). The w component of the homogenous coordinates presents scaling and usually has the value 1. In this way, we can substitute the fourth coordinate (w) by an additional attribute of classification. Fig. 1 shows the described data organization, where the main memory stores the quadtree with the indexes and VBOs, the geometry is on the graphic card and the actual points with all attributes are stored in the LAS file on the hard drive.

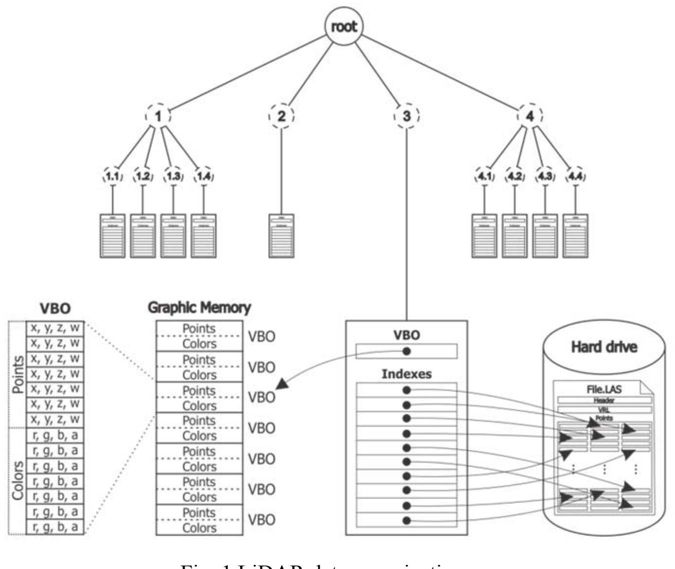

Fig. 1 LiDAR data organization

B. GPU-Accelerated Point-Filtering

Homogeneous coordinates are ubiquitous in computer graphics. Modern graphics processing unit (GPU) are built around a vector processor with 4-element registers and use single instruction, multiple data (SIMD) architecture to efficiently work with homogeneous coordinates [17]. OpenGL take advantage of this with programmable GPU rendering pipeline through shaders. Programmable shaders are simple programs that calculate rendering effects on the graphic hardware fast and with a high degree of flexibility.

Our technique is entirely GPU-based and implemented using vertex and fragment shaders. The vertex and fragment shaders are linked to a shader program that is individually performed on each graphic primitive. The shader program using both shaders performs the filtering, where those points that meet the condition are rendered, and others are discarded from the rendering pipeline.

First, the vertex shader is executed where the w coordinate (in the continuation referred to as a classification attribute) is passed to the fragment shader and the 2D screen coordinates are calculated with $w = 1$. Filtering is then performed in the fragment shader with the classification attribute. A traditional filtering approach on the GPU is ineffective because of the SIMD architecture. SIMD native branching is slow and needs a special strategy for emulating this kind of flow. Therefore, a masking technique is used, where a bitmask is applied to hold information of 32 layers according to the ASPRS LAS specifications [18]. The layers' bitmask is passed to the fragment shader from the rendering process (see next section). Every single bit in the layers bitmask corresponds to a single layer, where a set bit means a visible layer and vice versa. To compare or filter points, a point's classification attribute needs to be transformed from the classification value to the classification bit. The transformation is done with shifting value 1 left for the classification value. In this way, the corresponding classification bit is set and can be compared to the layers bitmask with a fast bitwise AND operation. At least, if the result is bigger than 0, the point is rendered otherwise it is discarded from rendering. In this way, we are able to filter point by the classification attribute very quickly on the GPU.

C. Rendering

Reduction of the graphic workload is needed in order to achieve real-time visualization. During the rendering process, we recursively traverse the quadtree from the root and check whether each node's domain intersects the current view frustum, and cull nodes from rendering that do not intersect. The described frustum culling is effective but has some drawbacks when the view covers the entire scene, where no workload is reduced. Therefore, points inside the viewing frustum have to be removed, too. For this purpose, a LOD technique is applied for simplifying the scene. LOD is realized by rendering a detailed geometry when the object is close to the viewer and coarser approximation when the object is distant or small. In this way, the graphic workload is significantly reduced and image quality is preserved. Our data organization with the geometry and colors stored on the graphic card ensures a simple and fast LOD implementation, where only the number of rendered points for each visible quadtree node needs to be calculated.

The rendering process works over two-passes: during the first pass the required information is obtained, whilst during the second pass the number of points is calculated and rendered. The first pass is conducted simultaneously with frustum calling and computes information about the average distance $\overline{D}$ of all visible quadtree nodes to the observer and the rendering ratio R. The rendering ratio is determined as the relationship between the maximum number of points that can be rendered in real-time and the number of points that are inside all visible quadtree nodes. In the second pass, the percentage of rendered points LOD_i for each visible quadtree node i is calculated as:

$$LOD_i = \frac{D_i \cdot (R-1)}{\overline{D}} + 1, \tag{1}$$

where D_i is the distance between the observer and the center of the visible quadtree node i. The result of the visualization can be seen in Fig. 2.

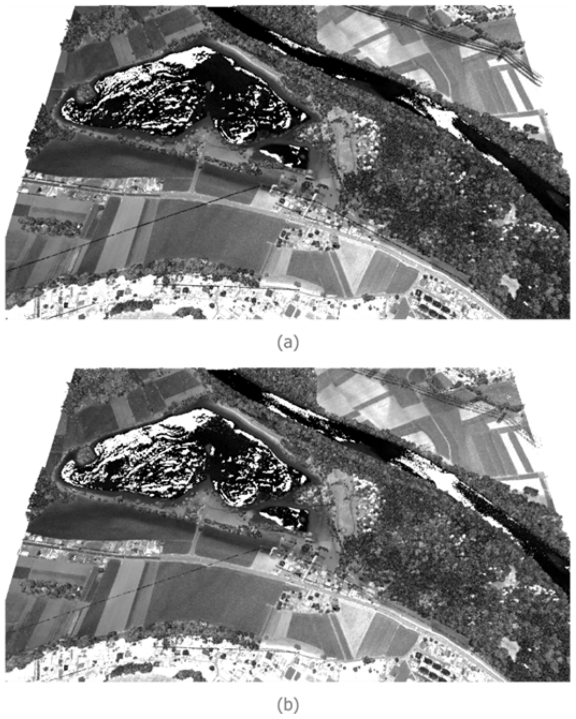

Fig. 2 LiDAR dataset (LAS file 4) visualization: (a) without LOD and (b) with LOD optimization

IV. Results

The presented methods have been implemented in C++ under Microsoft Foundation Class Library (MFC) and OpenGL 2.1 on Microsoft Windows 7 Professional. The measurements were obtained on a PC with 3.30GHz Intel Core i5, 8 GB of main memory, Western Digital Blue 1TB 7200RPM hard drive, and NVIDIA Ge-Force GTX 560 with 1

GB graphic memory.

Table I summarizes the performance in Frames per Second (FPS) and rendered points for eight different datasets, while the scene was rendered with and without LOD. The reduction of points (point ratio) and speed (FPS ratio) calculations were made in the forms of rendering relationships with and without LOD. The LOD optimization achieve real-time rendering in all cases and without loss of image quality. The image quality of the visualization can be seen in Fig. 2. The filtering process is as fast as the rendering process since it is integrated into the rendering process. The images with filtered data can be seen in Fig. 3.

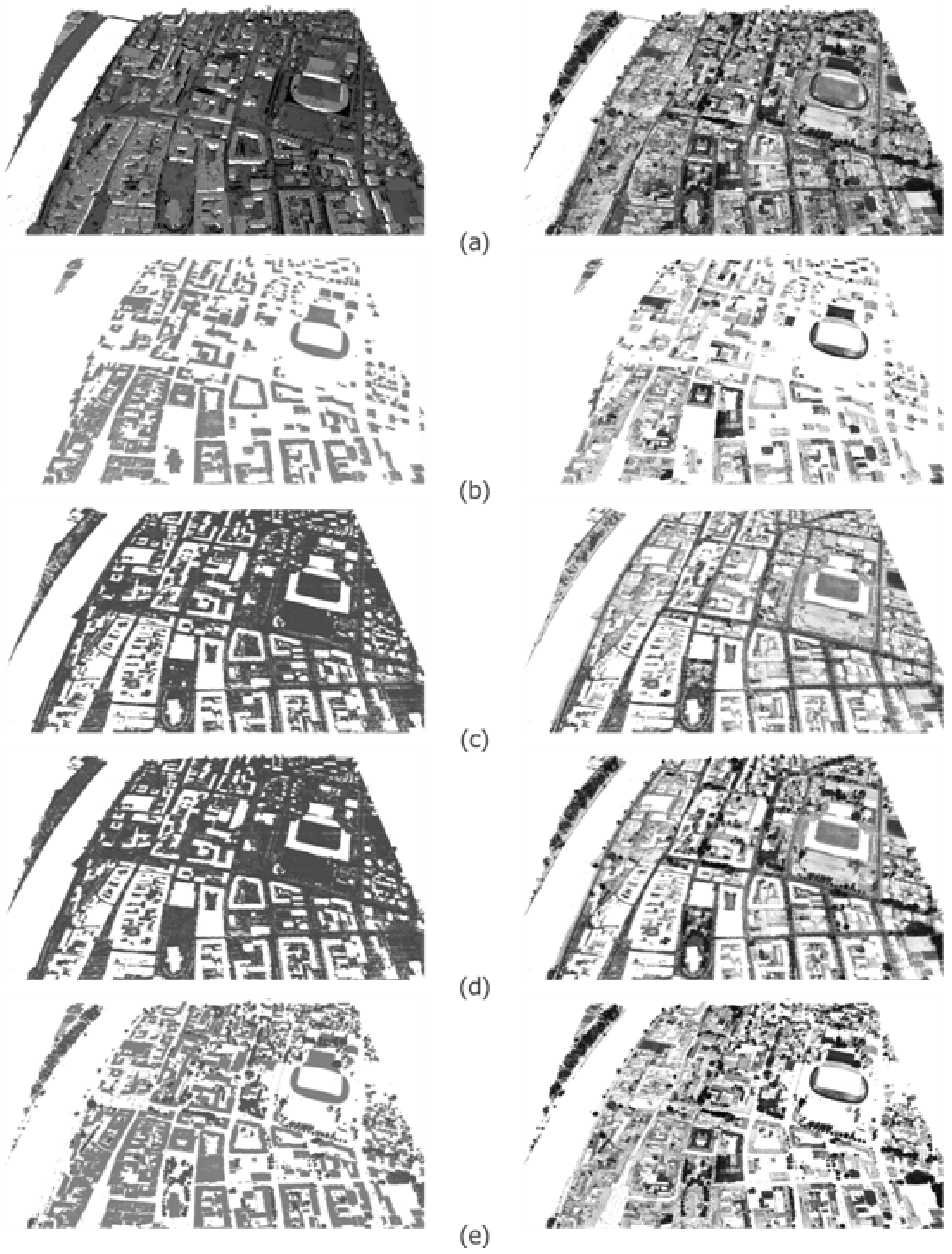

Fig. 3 LiDAR dataset (LAS file 4) filtering, points left colored according to classification attribute and right according to intensity attribute: (a) no data filtered, (b) filtering on building points, (c) filtering on ground points, (d) filtering on vegetation and ground points, (e) filtering on vegetation and building points

TABLE I
COMPARISON OF VISUALIZATION PERFORMANCE WITH AND WITHOUT LOD OPTIMIZATION

Dataset	Without LOD		With LOD			
	Points	FPS	Points	FPS	Points ratio	FPS ratio
LAS file 1	6190800	58	5051868	62	0.82	1.07
LAS file 2	15328038	29	5611506	50	0.37	1.72
LAS file 3	11884312	35	4828047	53	0.41	1.51
LAS file 4	18194184	26	4956370	52	0.27	2.00
LAS file 5	27355270	17	5323969	40	0.19	2.35
LAS file 6	20208650	23	5188756	42	0.26	1.83
LAS file 7	11667829	38	4992648	45	0.43	1.47
LAS file 8	11446218	38	4966357	55	0.43	1.45

V. CONCLUSION

The proposed visualization techniques is capable to clearly presenting objects contained within LiDAR point clouds in real-time, while overcoming the issues of massive point clouds and lack of topology. Further, we are able to present data in layers and filter them during the rendering process. For this, the method fully exploits the capabilities of actual GPU through programmable shaders.

ACKNOWLEDGMENT

This work was supported by the Slovenian Research Agency under Grants J2-5479, and P2-0041. This paper was produced within the framework of the operation entitled ‘‘Centre of Open innovation and Research UM’’. The operation is co-funded by the European Regional Development Fund and conducted within the framework of the Operational Programme for Strengthening Regional Development Potentials for the period 2007–2013, development priority 1: ‘‘Competitiveness of companies and research excellence’’, priority axis 1.1: ‘‘Encouraging competitive potential of enterprises and research excellence’’.

REFERENCES

[1] R. A. White, B. C. Dietterick, T. Mastin, and R. Strohman, “Forest Roads Mapped Using LiDAR in Steep Forested Terrain,” Remote Sensing, vol. 2. pp. 1120–1141, 2010.

[2] D. Mongus and B. Žalik, “Parameter-free ground filtering of LiDAR data for automatic DTM generation,” ISPRS J. Photogramm. Remote Sens., vol. 67, pp. 1–12, Jan. 2012.

[3] N. R. Vaughn, L. M. Moskal, and E. C. Turnblom, “Tree Species Detection Accuracies Using Discrete Point Lidar and Airborne Waveform Lidar,” Remote Sensing, vol. 4. pp. 377–403, 2012.

[4] A. Wehr and U. Lohr, “Airborne laser scanning—an introduction and overview,” ISPRS J. Photogramm. Remote Sens., vol. 54, no. 2–3, pp. 68–82, Jul. 1999.

[5] J. H. Clark, “Hierarchical geometric models for visible surface algorithms,” Commun. ACM, vol. 19, no. 10, pp. 547–554, 1976.

[6] B. J. Schachter, “Computer Image Generation for Flight Simulation,” Comput. Graph. Appl. IEEE, vol. 1, no. 4, pp. 29–68, 1981.

[7] M. Garland and P. Heckbert, “Simplification using Quadric Error Metrics,” in Proceedings of SIGGRAPH, 1997, pp. 209–216.

[8] R. Klein, G. Liebich, and W. Strasser, “Mesh reduction with error control,” in Proceedings of Visualization ’96., 1996, pp. 311–318.

[9] C. Erikson, D. Manocha, and W. V Baxter III, “HLODs for faster display of large static and dynamic environments,” in Proceedings of the 2001 symposium on Interactive 3D graphics, 2001, pp. 111–120.

[10] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright, “Simplification envelopes,” in Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996, pp. 119–128.

[11] J. S. Falby, M. J. Zyda, D. R. Pratt, and R. L. Mackey, “NPSNET: Hierarchical data structures for real-time three-dimensional visual simulation,” Comput. Graph., vol. 17, no. 1, pp. 65–69, Jan. 1993.

[12] M. Levoy and T. Whitted, “The Use of Points as a Display Primitive,” in Technical Report TR 85-022, University of North Carolina at Chapel Hill, 1985.

[13] S. Pečnik, D. Mongus, and B. Žalik, “Evaluation of Optimized Visualization of LiDAR Point Clouds, Based on Visual Perception,” in Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data SE - 35, vol. 7947, A. Holzinger and G. Pasi, Eds. Springer Berlin Heidelberg, 2013, pp. 366–385.

[14] B. Kovač and B. Žalik, “Visualization of LIDAR datasets using point-based rendering technique,” Comput. Geosci., vol. 36, no. 11, pp. 1443–1450, Nov. 2010.

[15] M. Kuder and B. Žalik, “Web-Based LiDAR Visualization with Point-Based Rendering,” in 2011 Seventh International Conference on Signal Image Technology & Internet-Based Systems, 2011, pp. 38–45.

[16] A. Samberg, “An Implemetation of the ASPRS LAS Standard,” Proc. ISPRS Work. “Laser Scanning 2007 SilviLaser 2007,” vol. 36, no. 3, pp. 363–372, 2007.

[17] A. R. Brodtkorb, T. R. Hagen, and M. L. Sætra, “Graphics processing unit (GPU) programming strategies and trends in GPU computing,” J. Parallel Distrib. Comput., vol. 73, no. 1, pp. 4–13, 2013.

[18] American Society for Photogrammetry and Remote Sensing (ASPRS), “LAS specification version 1.3,” The American Society for Photogrammetry & Remote Sensing, 2009. (Online). Available: www.asprs.org. (Accessed: 28-Jul-2014).

Sašo Pečnik received his BSc in Computer Science from University of Maribor, Maribor, Slovenia in 2009. He is currently employed as a researcher at the Faculty of electrical Engineering and Computer Science of University of Maribor and is working towards a PhD degree. His research interests include processing and visualization of LiDAR data, computer geometry, computer graphics, CAD, GIS, cloud computing and applications in Civil Engineering.

Borut Žalik is a full professor of computer science at the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. He received his BSc in electrical Engineering in 1985, MSc and PhD in computer science, both from the University of Maribor in 1989 and 1993, respectively. For 2 years, he had a position of a visiting senior research fellow at De Montfort University, UK. Žalik lead a laboratory for geometric modelling and multimedia algorithms. His research interests include computational geometry, geometric modelling, CAD, GIS and multimedia applications.