# Jornadas SARTECO

Actas de las

## Jornadas de Paralelismo - JP2015

23, 24 y 25 de septiembre de 2015

Córdoba

JP2015


# JORNADAS DE PARALELISMO

Actas
XXVI EDICIÓN DE LAS JORNADAS DE PARALELISMO (JP2015),
23, 24 y 25 de septiembre de 2015
Córdoba

Editores:
María Brox Jiménez
José María Castillo Secilla
Juan Carlos Gámez Granados
Andrés Gersnoviez Milla
Juan Gómez Luna
Ezequiel Herruzo Gómez
Fernando León García
Carlos Diego Moreno Moreno
Joaquín Olivares Bueno
José Manuel Palomares Muñoz
José Manuel Soto Hidalgo

ISBN:
978-84-16017-52-2

# VGLiDAR: Una herramienta de procesamiento de datos LiDAR en la GPU usando WebGL

David Deibe[1], Margarita Amor[1], Ramón Doallo[1], Rafael Crecente[2], David Miranda[2] y Miguel Cordero[2]

*Resumen*— **En este artículo se presenta una nueva herramienta web para la visualización, manipulación y realización de mediciones geoespaciales sobre datos LiDAR. El diseño de la herramienta ha estado enfocado en lograr tres objetivos fundamentales, el rendimiento en términos de interacción en tiempo real, la funcionalidad y la disponibilidad de datos LiDAR bajo demanda. Mediante el uso de WebGL la herramienta es capaz de aprovechar todo el potencial de las GPUs para manipular grandes volúmenes de datos. Todos los datos LiDAR son preprocesados y almacenados en una estructura de datos sin pérdida de calidad la cual minimiza los requisitos de transferencia y permite así ofrecer un servicio de datos LiDAR bajo demanda dentro de la herramienta.**

*Palabras clave*— **LiDAR, GPU, WebGL, mediciones geoespaciales, datos bajo demanda.**

## I. Introducción

La tecnología LiDAR (*Light Detection and Ranging*) proporciona información geoespacial de alta resolución y de gran utilidad que puede ser utilizada en un gran número de disciplinas tales como la agricultura, la arqueología, la biología, la geología o la ingeniería forestal. Las aplicaciones de LiDAR dentro de dichas disciplinas abarcan estudios sobre cambios en la morfología de las costas [1], mejoras en la comprensión de los procesos geomorfológicos [2], clasificación de superficies urbanas [3], análisis de corrimientos de tierra [4] o estudios sobre volcanes [5], entre otros muchos campos.

En la actualidad, la gran cantidad de información espacial que puede ser adquirida por la tecnología LiDAR implica un enorme desafío a la hora de desarrollar herramientas que puedan hacer uso de ella sin ningún tipo de restricción y sobre cualquier sistema operativo o dispositivo. La necesidad de implementar algoritmos para procesar y visualizar tal volumen de datos se convierte en un punto clave en el proceso de desarrollo. La interacción en tiempo real con los modelos 3D y las capacidades de almacenamiento necesarias para el manejo de este tipo de datos suponen un gran reto a la hora de desarrollar software que haga uso de ellos.

En [6] se presenta un método para la visualización en tiempo real de grandes conjuntos de datos LiDAR mediante una técnica basada en *point rendering*. En [7] se muestra una aplicación interactiva de gran calidad basada en un novedoso sistema jerárquico de *Level of detail* (LOD) para el manejo de conjuntos masivos de puntos. Con un enfoque centrado en la naturaleza 2.5D de los datos aéreos LiDAR se presenta en [8] una herramienta interactiva y visualmente completa para el *rendering* de nubes de puntos de ciudades, sin embargo, es necesaria la intervención por parte del usuario. En [9] se propone un sistema web para dispositivos móviles basado en un un modelo multiresolución comprimido que utiliza una eficiente representación de la malla. [7] y [10] muestran la visualización interactiva de nubes de puntos 3D cuyos tamaños exceden tanto la memoria disponible en el sistema como sus capacidades de *rendering*. Para lograrlo se hace uso de técnicas *out-of-core*. En [7], los autores logran un *render* masivo de puntos utilizando *geo-morphing* e interpolación de puntos. En [10] se presenta un *rendering* interactivo y *out-of-core* basado en un Kd-tree multiresolución por capas.

Todas las propuestas anteriores centran sus esfuerzos en obtener una sólida y foto realista representación 3D de los datos LiDAR logrando además eliminar huecos o falta de información en las superficies. Normalmente presentan los puntos como *splats* (discos planos), esferas o partículas. Sin embargo, un análisis detallado y el uso de conocimiento acerca de sus características y estructura es todavía un proceso complicado y en la mayoría de los casos una tarea manual. En contraste con las propuestas anteriores nosotros proporcionamos una solución alternativa basada en la utilización de los datos originales LiDAR sin pérdida o modificación de la información, uno de los principales requisitos exigidos dentro de las aplicaciones de las ciencias forestales o agrónomas.

En este trabajo presentamos VGLiDAR, una novedosa herramienta web para datos LiDAR basada en la tecnología WebGL[11]. Hemos desarrollado un sistema web escalable para la visualización y la exploración interactiva de datos LiDAR con capacidad para realizar mediciones directamente sobre las nubes de puntos y que permite obtener la información LiDAR bajo demanda. VGLiDAR puede ser utilizado en cualquier plataforma y con cualquier sistema operativo. El único requisito necesario para el uso de nuestra herramienta es la utilización de un navegador web compatible con WebGL.

## II. Tecnología LiDAR

Un gran número de aplicaciones LiDAR utilizan el formato de archivo LAS, un estándar en el campo de las soluciones LiDAR. Su especificación ha sido

[1]Grupo de Arquitectura de Computadores (GAC), Univ. de A Coruña, e-mails: david.deibe@udc.es, margarita.amor@udc.es y ramon.doallo@udc.es.

[2]Laboratorio del Territorio, Departamento de Ingeniería Agroforestal, Escuela Politécnica Superior, Univ. de Santiago de Compostela, e-mails: rafael.crecente@usc.es, david.miranda@usc.es y miguel.cordero@usc.es.

creada por la *American Society for Photogrammetry and Remote Sensing* (ASPRS)[1]. El objetivo del formato LAS es proporcionar un formato abierto para todo el hardware y software LiDAR.

Los archivos en formato LAS están pensados para almacenar información sobre nubes de puntos LiDAR o de cualquier otro tipo. La información es almacenada desde software especializado que combina información GPS con *Inertial Measurement Unit* (IMU) y pulsos láser para producir las coordenadas de punto $(x, y, z)$.

Específicamente, los archivos LAS contienen información binaria siguiendo una estructura bien definida consistente en: una cabecera inicial, cualquier número opcional de *variable length records* (VLRs), un gran bloque de datos con toda la información de cada punto y finalmente un número opcional de *extended variable length records* (EVLRs). La cabecera contiene información general sobre los datos almacenados o acerca del propio fichero, en ella puede encontrarse información como la cantidad de puntos que contiene, el byte dentro del fichero a partir del cual están almacenados o la fecha en la que fueron tomados. Tanto los VLRs como los EVLRs son secciones opcionales que pueden ser utilizadas por los diferentes softwares LiDAR para almacenar la información extra que consideren oportuna.

La Tabla I muestra las propiedades de un único punto almacenado dentro de un fichero LAS, concretamente un fichero que sigue el formato 0 (el formato LAS define varios tipos de formato de punto, cada uno con diferentes propiedades).

### A. Visualizadores LiDAR

El uso de la tecnología LiDAR ha experimentado un notable crecimiento en los últimos años y con ello el número de aplicaciones que manejan este tipo de datos, ya sean aplicaciones web o de escritorio, enfocadas únicamente a trabajar con LiDAR o como parte de un software más general. Entre las múltiples opciones disponibles hemos analizado algunas de las más conocidas y utilizadas.

*IDECanarias*[2]: Visor web bastante conocido en España que proporciona datos LiDAR bajo demanda de las Islas Canarias. Tiene unos tiempos de carga muy rápidos y es sencillo de utilizar. Su principal inconveniente es que está programado en Flash con lo que no logra una interacción fluida con muchos equipos, con 250.000 puntos se obtienen valores en torno a 36 FPS. La cantidad de puntos cargada en el visor está siempre entre los 100.000 y los 350.000, con lo que el detalle de la zona seleccionada por el usuario va a depender del tamaño de la misma. Zonas de unos pocos metros cuadrados se muestran con relativo buen detalle, sin embargo cuando se solicitan grandes zonas de varios kilómetros cuadrados los puntos utilizados son prácticamente los mismos haciendo que la imagen pierda mucho detalle. Tiene muy pocas opciones de visualización (intensidad, altura y mixto) y no tiene herramientas de medición geoespacial sobre la nube de puntos 3D.

*LiDAR Online*[3] *(visor 3D)*: Uno de los visores web más conocidos a nivel global basado en la tecnología *Dielmo 3D*. Permite solicitar datos bajo demanda de varias zonas del mundo. El visor 3D posee buenas opciones de visualización (clasificación, intensidad, altura y RGB) y posee un buen rendimiento gráfico, sin embargo la cantidad de puntos utilizada no es muy superior a la de *IDECanarias* y su sistema de carga de datos es muy parecido a este ya que la cantidad de puntos varía en función de la zona seleccionada haciendo que las zonas de gran extensión pierdan mucho detalle. Los tiempos de carga son muy lentos (más de un minuto para la carga de menos de un millón de puntos) y a veces el proceso se queda bloqueado. No tiene herramientas de medición geoespacial sobre la nube de puntos 3D.

*Visor Ayuntamiento de Leioa*[4]: Este visor web está basado en *Dielmo 3D Viewer* y utiliza JavaWS para funcionar. Según el manual del propio visor es posible visualizar los puntos en función de su intensidad, altura o clasificación y realizar algunas mediciones como varios tipos de distancias, pendientes y taludes pero solamente sobre secciones planas de puntos, esto es, mediciones 2D sobre un subconjunto de puntos.

*Lidarview*[5]: Visor web que trabaja con WebGL. Tiene un buen rendimiento y varias opciones de visualización (intensidad, clasificación, altura y RGB). Es sencillo de utilizar pero solo permite cargar los archivos locales de los que disponga el usuario (formato LAS o XYZ). Según las propias opciones de la aplicación tiene un límite de 10 millones de puntos. No tiene herramientas de medición geoespacial sobre la nube de puntos.

*LAStools*[6] *(visor 3D)*: Probablemente la aplicación de escritorio de datos LiDAR más conocida. Es gratuita, sencilla de utilizar y con múltiples herramientas de trabajo. Su visor 3D posee numerosas opciones de visualización tales como intensidad, altura, clasificación, retornos, RGB o ángulo de vuelo. Permite filtrar puntos por propiedades e incluso triangular todo el conjunto de puntos para generar un modelo 3D completo. A pesar de ofrecer múltiples posibilidades, el visor 3D tiene un bajo rendimiento, con 4.700.000 se obtienen valores inferiores a los 10 FPS Además, solo permite cargar los archivos locales de los que disponga el usuario y no tiene herramientas de medición geoespacial sobre la nube de puntos.

## III. Estructura general de VGLiDAR

La Figura 1 muestra la estructura general del sistema. La arquitectura de VGLiDAR puede ser considerada como una arquitectura cliente-servidor con dos partes claramente diferenciadas. La primera de

[1]ASPRS. Especificación LAS: http://www.asprs.org/Committee-General/LASer-LAS-File-Format-Exchange-Activities.html

[2]http://visor.grafcan.es/visorweb/

[3]http://www.lidar-online.com/tools/maps/

[4]http://dataserver.dielmo.com/leioa/

[5]http://lidarview.com/

[6]http://rapidlasso.com/lastools/

TABLA I
INFORMACIÓN ALMACENADA DENTRO DE CADA REGISTRO EN LOS ARCHIVOS LAS (FORMATO 0).

| Propiedad | Formato | Tamaño | Obligatorio |
|---|---|---|---|
| *X* | long | 4 bytes | * |
| *Y* | long | 4 bytes | * |
| *Z* | long | 4 bytes | * |
| *Intensity* | unsigned short | 2 bytes | |
| *Return Number* | 3 bits (bits 0 - 2) | 3 bits | * |
| *Number of Returns* | 3 bits (bits 3 - 5) | 3 bits | * |
| *Scan Direction Flag* | 1 bit (bit 6) | 1 bit | * |
| *Edge of Flight Line* | 1 bit (bit 7) | 1 bit | * |
| *Classification* | unsigned char | 1 bytes | * |
| *Scan Angle Rank* | char | 1 bytes | * |
| *User Data* | unsigned char | 1 bytes | |
| *Point Source ID* | unsigned short | 2 bytes | * |

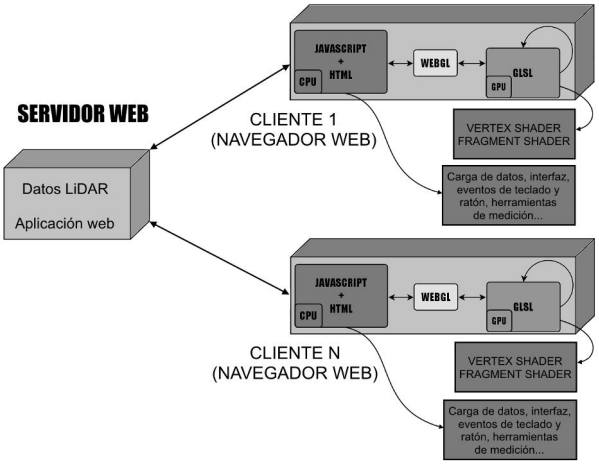



Fig. 1. Estructura general de VGLiDAR.

ellas consta de un servidor web tradicional, como Apache HTTP Server, el cual almacena todo el sistema y los datos LiDAR. La segunda parte engloba a cualquier navegador web, del lado del cliente, compatible con WebGL. Esto permite que múltiples usuarios puedan acceder al sistema a través de sus navegadores y generar el *render* de la escena final en su propia plataforma.

En el desarrollo de VGLiDAR se han utilizado tres lenguajes de programación diferentes además del API gráfica WebGL. Estos lenguajes son; HTML5, JavaScript y *OpenGL Shading Language* (GLSL[7]). Los dos primeros son ejecutados en la CPU y se encargan de realizar tareas tales como la creación de la interfaz de usuario, la gestión de los eventos de teclado y ratón, la comunicación con el servidor para la obtención de los datos LiDAR solicitados por el usuario o la realización de diferentes mediciones geospaciales. El tercer lenguaje, GLSL, es utilizado para programar los *shaders* de la GPU (*Vertex Shader* y *Fragment Shader*). Los *shaders* son programas que proporcionan una gran flexibilidad a la hora de implementar los algoritmos de *rendering* y computación en la GPU. En nuestro caso se utilizan en tareas tales como la generación de colores para la nube de puntos, la eliminación de puntos en escena o la selección de vértices mediante ratón que permite realizar las diferentes mediciones geoespaciales. Por último, WebGL, una API gráfica que a pesar de ser también código JavaScript, es una parte claramente diferenciada del resto del código de la aplicación. Esta parte funciona como una interfaz entre la CPU y la GPU, permitiendo la comunicación y el envío de información entre ambas.

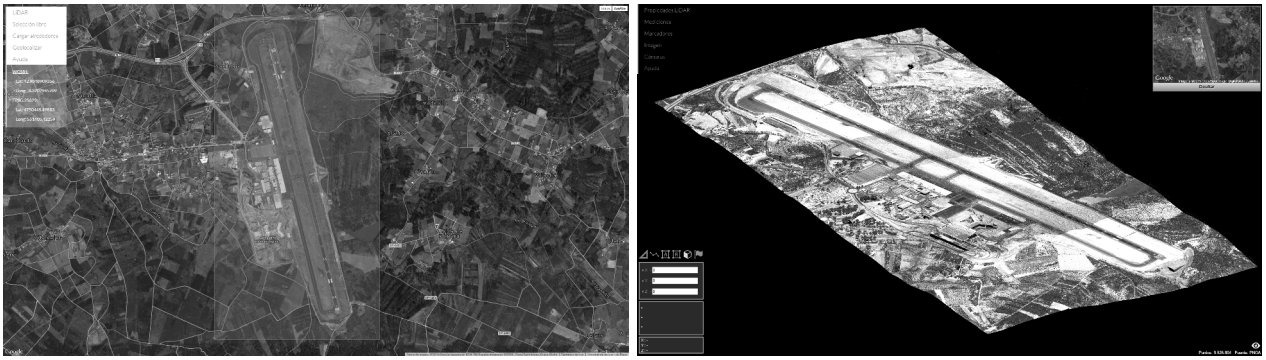
Desde el punto de vista de los usuarios la aplicación está compuesta por dos páginas web separadas. La página de *Selección* (ver Figura 2(a)) muestra un mapa mediante el uso de GoogleMaps API 3.0. Sobre este mapa los usuarios pueden obtener su posición geográfica o seleccionar una región de la cual deseen obtener sus datos LiDAR. Una vez que los usuarios ha seleccionado una determinada región la página de *Visualización* (ver Figura 2(b)) emerge mostrando los datos LiDAR disponibles para la misma. Los usuarios podrán cambiar el modo en el que la GPU produce el *render* de la nube de puntos atendiendo a diferentes propiedades LiDAR como la altitud, la intensidad, la clasificación o el número de retorno.

Una de las principales diferencias de VGLiDAR con el resto de herramientas de visualización es la incorporación de mediciones geoespaciales que pueden ser realizadas directamente sobre las nubes de puntos. Los usuarios podrán tomar distancias entre puntos, medir la altura de estructuras, generar áreas proyectadas, triangular puntos en un perímetro determinado, realizar medidas sobre fachadas de edificios o eliminar puntos de la escena. Por ejemplo, la Figura 3(a) muestra los datos LiDAR del estadio municipal de Riazor (A Coruña) en donde se está realizando una medición de atura, desde lo alto de la torre de maratón hasta el césped del terreno de juego. La Figura 3(b) muestra otro ejemplo de medición, en este caso se está creando un área proyectada que cubre diferentes partes de la superficie del río Oitaven (Pontevedra).

## IV. PREPROCESADO DE DATOS LIDAR

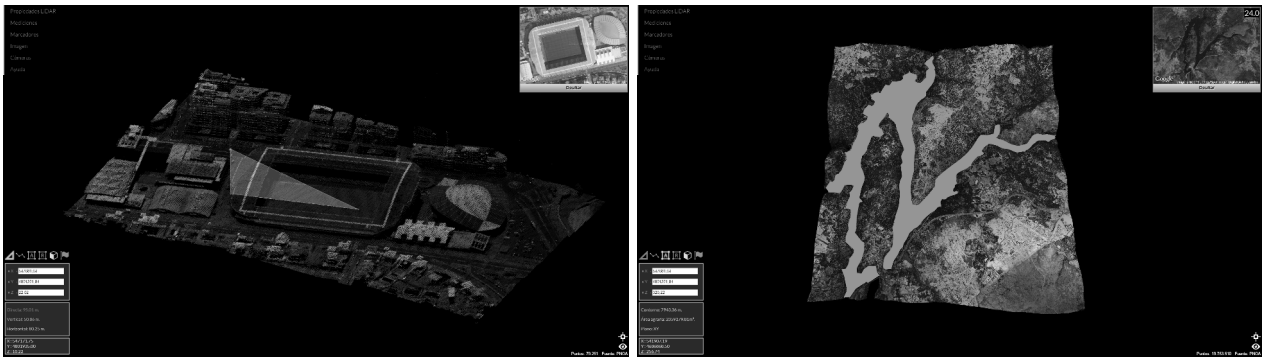
En esta sección explicaremos la reestructuración de los datos que permite a VGLiDAR obtener flexi-

[7]https://www.opengl.org/documentation/glsl/



(a) (b)

Fig. 2. Herramienta VGLiDAR. (a) Página de selección de VGLiDAR sobre el Aeropuerto Internacional de Santiago de Compostela. (b) Visualización de la nube de puntos.



(a) (b)

Fig. 3. Ejemplos de uso de la herramienta de mediciones geoespaciales sobre datos LiDAR 3D. (a) Altura. (b) Área proyecta.

bilidad, rendimiento y menores requisitos de almacenamiento.

Algunas de las propiedades almacenadas en los archivos LAS no son útiles desde el punto de vista de la visualización ni de las mediciones geoespaciales lo cual aumenta de manera innecesaria el volumen de datos que debe manejar una aplicación que haga uso de ellos. Mantener estos datos demanda gran capacidad de almacenamiento. Debido a ello, y al hecho de que en la actualidad las nubes de puntos pueden llegar a contener miles de millones de puntos, se hace casi obligada la implementación de algún tipo de compresión de los datos con el objetivo de reducir la cantidad de información que es almacenada o enviada por la red. Por ejemplo, todos los archivos LAS de Galicia obtenidos por el Plan Nacional de Ortofotografía Aérea (PNOA[8]) tienen un tamaño aproximado de 900 GB, sin embargo tras la eliminación o re estructuración de determinados elementos el tamaño puede ser reducido hasta 450 GB, lo que implica una reducción del 50%. Entre la información innecesaria desde el punto de vista de la visualización podemos encontrar: toda la cabecera del archivo LAS, todos los VLR, los EVLR y las propiedades de punto: *Scan Direction Flag*, *Edge of Flight Line*, *Scan Angle Rank*, *User Data* y *Point Source ID*. Además, algunos archivos LAS pueden contener datos con valores nulos o cero, lo que los convierte en información no útil. En el caso de los datos de PNOA todos los puntos tienen una propiedad de color ($RGB$) cuyos valores son siempre cero, esto supone 12 bytes de información no útil por cada punto, por ello, esta propiedad es eliminada. Parte del trabajo futuro de este proyecto se centra en la implantación de técnicas de compresión tales como las descritas en [12] que permitan seguir reduciendo los tamaños de los archivos procesados.

Además de los problemas descritos anteriormente, los archivos LiDAR de gran tamaño, incluso después de haber sido simplificados, hacen difícil la implementación de un sistema bajo demanda para la visualización. La selección de una pequeña sección de terreno para ser utilizada puede implicar la descarga de una cantidad de puntos mucho mayor que la estrictamente necesaria o razonable. Esto puede convertirse en un problema mucho mayor si, por ejemplo, los usuarios intentan visualizar una zona comprendida en la intersección de 4 archivos de gran tamaño. Para resolver estos inconvenientes todos los datos LiDAR son redistribuidos en archivos de un tamaño menor a 5 MB lo que reduce significativamente la cantidad de información solicitada al servidor cada vez que un usuario selecciona una zona para trabajar. La razón de haber elegido 5 MB es la de asegurar que todos los navegadores web puedan

[8]http://pnoa.ign.es/presentacion

ser capaces de almacenar en su caché la información LiDAR descargada consiguiendo con ello que las descargas solamente tengan que ser realizadas una vez y permitiendo que los datos obtenidos puedan ser reutilizados en las siguientes consultas.

Por otra parte, el formato de archivo LAS almacena toda la información relativa a un punto concreto de manera conjunta y cuando esta termina comienza la información del punto siguiente. Como se comentó al comienzo de este apartado no toda la información contenida en los archivos es útil desde el punto de vista de la visualización. De cada punto solamente serán utilizadas sus coordenadas ($x$, $y$, $z$) y sus valores de clasificación, retorno e intensidad. Es por ello que, de trabajar con los archivos LAS originales, tras su descarga sería necesario acceder a cada una de las propiedades útiles y descartar el resto. Para manipular archivos binarios puros en Javascript es necesario utilizar la función *Slice* y transformaciones de tipo de dato desde binario a *byte*, *short*, *integer* o *float* según sea necesario. La función *Slice* permite obtener datos desde un archivo binario, accediendo a un byte de inicio dentro del archivo y leyendo hasta un byte de finalización. Ya que los archivos LAS y los puntos que estos contienen siguen una estructura muy concreta y definida (ver Sección II) es posible saber exactamente en qué posiciones se puede encontrar la información. Basta pues con realizar un bucle que por cada punto acceda a posiciones concretas dentro del archivo para ir progresivamente obteniendo uno a uno los valores que se necesitan. Así pues, esta distribución de datos implica la necesidad de realizar, por cada punto, 4 llamadas a la función *Slice*. En la primera se extraen las coordenadas del punto, en la segunda su intensidad, a continuación el retorno y finalmente la clasificación. Tomando como referencia los archivo LAS de PNOA que contienen cada uno casi 3.000.000 de puntos es necesario realizar cerca de 12.000.000 de llamadas a *Slice* y realizar 12.000.000 de transformaciones de datos. Todas estas operaciones penalizan en gran medida el tiempo de carga de datos. Para solucionarlo, el preprocesado de datos genera un archivo donde las coordenadas de todos los puntos se encuentran almacenadas conjuntamente, a continuación se sitúan todas las intensidades y finalmente los números de retorno junto a los valores de clasificación. De este modo durante la lectura es posible obtener todo un bloque de datos (bloque de coordenadas, bloque de clasificación, ...) con cada operación *Slice* y realizar la transformación de datos sobre todo el conjunto. Este método consigue reducir los tiempos de carga y además permite enviar directamente los datos obtenidos a la GPU.

Los dos modos más comunes para el almacenamiento de vértices, o atributos de puntos, son el *array* de estructuras (AOS - array of structures) y la estructura de *arrays* (SOA - structure of arrays). La AOS es implementada en las APIs gráficas utilizando una configuración de un único *buffer* que almacena todos los atributos de los puntos conjuntamente. Por su parte, la SOA es implementada utilizando una configuración con múltiples *buffers*, cada *buffer* almacena en exclusiva un tipo de atributo. Los archivos LAS están formados siguiendo una estructura del tipo AOS, sin embargo, para el diseño de VGLiDAR se ha optado por programar los *shaders* de la GPU siguiendo una estructura SOA debido a que este tipo de estructura obtuvo mejores resultados en los test de rendimiento que fueron llevados a cabo para comparar ambas estructuras. Mantener esta discrepancia entre la estructura de los datos descargados y la configuración de los *shaders* requería leer individualmente las propiedades de cada punto para ir progresivamente reorganizándolas en las estructuras adecuadas que serán usadas por la GPU. El almacenamiento por bloques comentado con anterioridad sigue una estructura del tipo SOA con lo que permite enviar directamente a la GPU los datos descargados. La Tabla II muestra la estructura interna de los archivos preprocesados para un archivo de datos LiDAR de $N$ puntos.

TABLA II
DISTRIBUCIÓN DE BLOQUES DE PROPIEDADES DENTRO DE LOS ARCHIVOS LAS PREPROCESADOS.

| Propiedad | Tamaño | Tipo |
|---|---|---|
| $X, Y, Z$ | $3 \times 4 \times N$ | bytes |
| Intensidad | $4 \times N$ | bytes |
| Ret. + Clas. | $(3bits + 5bits) \times N$ | bytes |

Las coordenadas contenidas en los archivos LAS no son las coordenadas reales de cada punto. Cada una de ellas ha de ser ajustada según una escala y un desplazamiento indicado en la cabecera del archivo LAS al que pertenecen. El preprocesado almacena los valores de las coordenadas con este ajuste ya realizado con lo que se evitan numerosas operaciones que tendrían que ser realizadas durante la lectura de datos.

## V. Resultados experimentales

En esta sección se muestran diversas pruebas de rendimiento realizadas sobre VGLiDAR. Para realizar las pruebas se han elegido 3 diferentes GPUs intentando abarcar diferentes arquitecturas. Las GPU seleccionadas han sido: GeForce GTX 980 (arquitectura Nvidia Maxwell), GeForce GTX Titan (arquitectura Nvidia Kepler) y Radeon HD 6970 (arquitectura AMD Cayman XT), utilizando los *drivers* 350.12 para las Nvidida y los 14.12 Omega en la AMD. El equipo utilizado consta de una CPU Intel Core i7 4790, 32 GB de RAM DDR3 y el sistema operativo Windows 7 Professional (64 bits). Los test han sido realizados a una resolución de $2048 \times 1152$ bajo el navegador Chrome 42.0.2311.135 (64 bit). Se ha optado por el navegador Chrome debido a que mostró un mejor rendimiento y estabilidad durante las pruebas respecto a otros navegadores como Firefox o Internet Explorer. Hemos utilizado como base de datos la información LiDAR de PNOA de la comunidad autónoma de Galicia. Estos datos fueron obtenidos mediante sensores LiDAR aerotransporta-

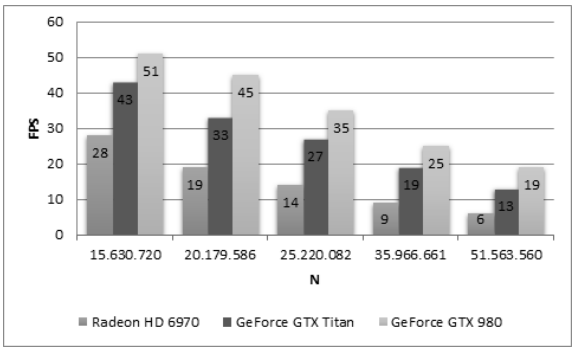



Fig. 4. Rendimiento en términos de FPS con diferentes arquitecturas de GPU.

dos, con una densidad de 0.5 puntos por metro cuadrado y una precisión altimétrica mejor que 20 centímetros RMSE $z$.

En la Figura 4 se muestran los resultados obtenidos para cada GPU en donde $N$ indica el número de puntos que son cargados en la página de visualización. Como se puede ver en la figura hemos logrado interacción en tiempo real hasta los 36 millones de puntos utilizando la GeForce GTX 980. A pesar de no lograr la misma interacción sobre 51 millones de puntos es posible trabajar de manera aceptable sobre esta cantidad de puntos tanto en una GeForce GTX 980 como en una GeForce GTX Titan. Finalmente destacar también que la Radeon HD 6970, a pesar de ser una GPU con una arquitectura antigua, logra una interacción en tiempo real con conjuntos de 15 millones de puntos.

En resumen, VGLiDAR permite visualizar 10,97 veces más puntos que LAStools multiplicando por 1,9 el rendimiento obtenido o 100 veces más puntos que IDECanarias a una tasa en torno a 36 FPS. Con respecto a Lidarview nuestra herramienta permite visualizar 5 veces más puntos. Además alcanza interacción en tiempo real con hasta casi 36 millones de puntos mientras que otras herramientas web como IDECanarias o Lidar-Online consiguen fluidez recortando el número de puntos que muestran al usuario.

## VI. Conclusiones y trabajo futuro

VGLiDAR es una aplicación web con una arquitectura cliente-servidor que permite disponer de datos LiDAR bajo demanda. Además, mientras que otras aplicaciones web obtienen buenos rendimientos con un conjunto pequeño de puntos (menos de un millón) VGLiDAR logra interacción en tiempo real manejando conjuntos de 36 millones de puntos e incluso es capaz de visualizar de manera aceptable conjuntos de más de 51 millones, lo que supera a algunas aplicaciones de escritorio como LAStools.

Con el propósito de seguir mejorando los 3 pilares fundamentas de la herramienta (interacción en tiempo real, funcionalidades y datos bajo demanda) se pretenden incorporar en el futuro técnicas de optimización gráfica como *View Frustum Culling* o LOD, aplicar técnicas de compresión, seguir incorporando nuevas funcionalidades y mejorar la herramienta de mediciones geoespaciales.

## Agradecimientos

Esta investigación ha sido apoyada por el Gobierno Gallego (Xunta de Galicia) bajo el Programa para la Consolidación de Grupos de Referencia Competitiva, cofinanciado con fondos FEDER de la UE (Ref. GRC2013 / 055); bajo el Programa para la Consolidación de Unidades de Investigación Competitiva, cofinanciado con fondos FEDER de la UE (Ref. R2014/049); y por el Ministerio de Economía y Competitividad de España y los fondos FEDER de la UE (Project TIN2013-42148-P). Los datos LiDAR utilizados en este paper pertenecen al repositorio de datos LiDAR-PNOA cedido por © Instituto Geográfico Nacional de España.

## Referencias

[1] A. H. Sallenger Jr., W. Krabill, J. Brock, R. Swift, M. Jansen, S. Manizade, B. Richmond, M. Hampton, and D. Eslinger, "Airborne laser study quantifies El Niño-induced coastal change," *Eos, Transactions, American Geophysical Union*, vol. 80, no. 8, pp. 89–92, 1999.

[2] P. Tarolli, "High-resolution topography for understanding earth surface processes: Opportunities and challenges," *Geomorphology*, vol. 216, pp. 295–312, 2014.

[3] W. Y. Yan, A. Shaker, and N. El-Ashmawy, "Urban land cover classification using airborne LiDAR data: A review," *Geomorphology*, vol. 158, pp. 295–310, 2015.

[4] G. Ventura, G. Vilardo, C.Terranova, and E. B. Sessa, "Tracking and evolution of complex active landslides by multi-temporal airborne LiDAR data: the Montaguto landslide (Southern Italy)," *Remote Sensing of Environment*, vol. 115, no. 12, pp. 3237–3248, 2011.

[5] H. Hasegawa, H. P. Sato, and J. Iwahashi, "Continuous caldera changes in miyakejima volcano after 2001," *Bulletin of Geospatial Information Authority of Japan*, vol. 54, pp. 60–64, 2007.

[6] B. Kovac and B. Zalik, "Visualization of LIDAR datasets using point-based rendering technique," *Computers and Geosciences*, vol. 36, no. 11, pp. 1443–1450, 2010.

[7] P. Goswami, F. Erol, R. Mukhi, R. Pajarola, and E. Gobbetti, "An efficient multi resolution framework for high quality interactive rendering of massive point clouds using multi-way kd-trees," *The Visual Computer*, vol. 29, no. 4, pp. 95:1–95:6, 2013.

[8] Z. Gao, L. Nocera, M. Wang, and U. Neumann, "Visualizing aerial LiDAR cities with hierarchical hybrid point-polygon," in *Proceedings of Graphics Interface*, 2014, pp. 137–144.

[9] M. B. Rodríguez, E. Gobbetti, F. Marton, and A. Tinti, "Coarse-grained multiresolution structures for mobile exploration of gigantic surface models," in *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*, 2013, pp. 4:1–4:6.

[10] R. Richter, S. Discher, and J. Döllner, "Out-of-core visualization of classified 3d point clouds," in *3D Geoinformation Science. Lecture Notes in Geoinformation and Cartography*, 2015, pp. 227–242.

[11] T. Parisi, *WebGL - Up and Running*, O'Reilly Media, 2012.

[12] D. Mongus and B. Žalik, "Efficient method for lossless LiDAR data compression," *International Journal of Remote Sensing*, vol. 32, no. 9, pp. 2507–2518, 2011.