MrSID: A Modern Geospatial Image Format

Over the past few years, the size and variety of geospatial data has increased at an astonishing pace. Multispectral imagery and LiDAR data are now being collected with better accuracy than ever before. Through it all, the LizardTech® MrSID® format has evolved to anticipate the needs of the GIS industry. Now in its fourth generation, the MrSID format has become the most advanced compressed image format on the market, with support for multispectral imagery, alpha bands, and even LiDAR point clouds.

In this white paper, LizardTech introduces you to the concept of compression, to the MrSID technology, and to the features that the MrSID format can bring to your applications and workflows.

Contents

I. The Need for Data Compression

Between the rise of mobile devices and the advent of cloud computing, it has never been easier to access and analyze geospatial data—likewise, the demand for good data has never been higher. To meet this demand, commercial satellites have become ever more powerful and unmanned aerial vehicles more popular. Unfortunately, the proliferation of geospatial imagery comes with its own set of problems. Organizations as diverse as web content providers and county governments now share the challenges of storing and accessing massive amounts of data.

Because digital image files are so large, maintaining imagery in its raw, uncompressed form requires immense physical storage resources, and accessing raw data requires high-bandwidth networks and large memory workstations. A typical alternative is to store imagery in compressed form, using file formats like JPEG. Although JPEG versions of the images may enable faster access to lower resolution image overviews, their quality is not suitable for analysis and exploitation work at high resolution. This often leads to the practice of storing multiple versions of each dataset, at different resolutions or compression ratios – one for browsing, one for analysis, and so on. The storage and maintenance problem gets worse, not better.

We know from working with customers such as the United States Geological Survey (USGS) and the National Geospatial-Intelligence Agency (NGA) that today's geospatial workflows regularly require support for files hundreds of gigabytes in size, storage of the imagery without significant quality loss, access to multiple resolutions or overviews quickly, efficient random access into the file to support arbitrary scene requests, and so on.

For two decades, the patented MrSID technology from LizardTech has been the GIS industry's leading solution to these problems. Our applications such as GeoExpress® and Express Server® are used by thousands of people every day to encode and deliver imagery. And thousands more use MrSID files in hundreds of applications including Esri ArcGIS, ERDAS Imagine, and Google Earth.

With the introduction of the MG4™ format, the latest version of the MrSID technology, LizardTech offers compression for new kinds of datasets like multispectral images and LiDAR point clouds. In this white paper, we will introduce you to MrSID technology: what features it offers, where it can be applied, and how it works.

II. MrSID Technology

What is Compression?

Compression just means making data more compact so it occupies less space, but the word sometimes has negative connotations in our industry: often compressing imagery resulted in degraded data quality. You could have high fidelity or you could save disk space – but not both.

Consider a typical aerial image that is in 8-bit color and is 6,000 pixels wide by 6,000 pixels high. This means the raw data would consist of about 100 megabytes (MB) on disk: 36 million pixels (6,000 x 6,000), with each pixel requiring three bytes (three color bands of one byte each). A common workflow might, in fact, have dozens or even hundreds of such images arranged as tiles to make up a complete mosaic, but let's just consider the one image.

When the pixels are stored in an *uncompressed* (raw) form like this, the image data is not compressed at all: the amount of disk space required is equal to the number of bytes needed to represent the pixels (plus some small amount for metadata such as the geospatial positions of the corner points). Geospatial file formats like GeoTIFF store data in this raw, or *uncompressed*, way: every single pixel is fully represented. Uncompressed formats have the advantage of representing each pixel's data exactly as it was originally recorded, and so we say the image data is *lossless*. The obvious disadvantage, of course, is all the disk space you need.

Through the use of various algorithms, however, we can often represent the pixel data in a more efficient form. Intuitively, you can think of it this way: if the data consisted of a sequence of five identical values such as 123 123 123 123 123, you might instead be able to code it using a sort of shorthand notation like 123[5] in which you store the value only once but add a "repeat count" to it. Techniques like this yield data that is *compressed* – which means less disk space is needed – but the image data is still lossless.

If we push our compression algorithms further, we can find techniques which do not store the data for each pixel exactly but instead store only approximations of the data. Again intuitively, a high precision number like 3.1415, which requires five digits, might be stored as just the single-digit number 3, an 80% reduction in digit space (a 5:1 *compression ratio*). The JPEG file format, commonly used for small images on the web, uses this sort of *lossy* compression. The penalty for such lossy compression schemes is that you typically lose image quality: edges might not be as sharp, colors might seem flatter, small artifacts might be introduced. For some less demanding geospatial workflows, this might be acceptable.

In summary, then, we have a spectrum of three kinds of compression and data loss: uncompressed and lossless; compressed and lossless; and compressed and lossy. The naive techniques just described to achieve these kinds of compression will work, but by no means would provide high compression ratios, high image quality, and high performance. To do compression well in the real world, the algorithms need to be designed for specific kinds of data, such as geospatial imagery, and specific kinds of workflows.

MrSID Technology: Quality and Performance

The compression techniques used by MrSID technology provide both high quality imagery and high performance while still meeting our industry's challenging workflows.

Image Quality

MrSID technology's lossless compression yields compression ratios of 2:1 for typical imagery. This means you need only half the storage space and yet still retain your numerically identical, original data.

For further storage savings, MrSID technology's lossy compression can yield typical ratios of up to 20:1 while still offering a level of image quality that makes the data indistinguishable to the eye for most workflows. We refer to this style of compression as being *visually lossless*: there is still data loss, but with respect to what the image is being used for, the amount of loss is imperceptible.

Higher compression ratios are, of course, possible. Depending on how much image quality you need to retain, and depending on the type of your original imagery, ratios of 40:1 and beyond can be used.

Encoding Performance

Geospatial datasets can grow to consume terabytes of disk space. Sometimes the dataset is just one large file, but often it consists of hundreds of smaller tiles that, taken together, form a large mosaic. Either way, with many applications the amount of memory required to process and compress such large datasets is prohibitive, requiring the use of server-class hardware or requiring the end user to work with their imagery "piecemeal", only a subset of the tiles at a time.

MrSID technology is designed to address this problem. With full support for input and output file sizes larger than 2 GB and with support for 64-bit processors, there is virtually no limit to the size of the imagery that can be compressed.

Viewing Performance

Even at a ratio of 20:1, compression of a 20 TB dataset still results in an extremely large (1 TB) MrSID file. Users are often nervous about working with files of that size because many GIS applications will attempt to ingest the

entire file, causing excessive and occasionally fatal demands on the CPU and memory. Two aspects of MrSID technology solve this problem.

First, the encoding technique we use creates *multiple resolutions* of the image within the generated MrSID file. This is similar in effect to the trick of creating image "pyramids", but the MrSID technique does not create extra files for all the pyramid levels nor incur the corresponding overhead in storage space. Instead, the resolution levels are inherent in the encoding scheme and fully contained within the single MrSID output file itself. These levels are scaled such that each level is one quarter of the previous level: for example, the full image at 1024x1024, then a quarter-scale image at 512x512, then a sixteenth-scale image at 256x256, and so on down to an "icon" or "thumbnail" of typically 32x32. Applications can be instructed to extract and process to only the level of detail required, without having to decode (and perhaps then manually down-sample) the entire full-resolution image.

In addition to enabling requests for only the resolution level desired, MrSID technology offers *selective decompression*, which allows applications to request and decode only the scene (geographic area) of interest from the file. Some other file formats and compression schemes, by contrast, require the whole image to be decoded even if only a small part is to be shown on the screen.

A Brief History of MrSID

The prototype for MrSID technology was developed at Los Alamos National Laboratory in 1992. LizardTech's subsequent commercial version of the technology was called the MrSID Generation 2 (MG2) format and was introduced in 1998. The next version, the MrSID Generation 3 (MG3) format, introduced in 2002, offered improved image quality and key features such as lossless encoding. The latest version of the MrSID file format, MrSID Generation 4 (MG4), was introduced in 2009. The MG4 format supports multispectral imagery and alpha bands.

Additionally, the MG4 format introduced support for LiDAR data. Users of MG4 LiDAR data get many of the same key features found in our raster compression technology, including lossless compression, visually lossless compression, and selective decoding.

MrSID technology is available in LizardTech's suite of applications and tools, ranging from full-featured encoding applications to image servers to lightweight viewers. In addition, through our integration partners, MrSID files are supported in hundreds of GIS applications. Whether you are using older MG2 files in your legacy applications, taking advantage of MG3's lossless files in your current workflows, or have a new need to efficiently encode multispectral raster or LiDAR point cloud datasets, MrSID compression technologies will support your needs.

III. MrSID Technology for Imagery

In this section we describe in more detail some of the features and capabilities of the MrSID technology for raster image data.

Data Types and Formats

The MrSID technology is agnostic with respect to the input file format, as long as the input pixel data meets certain datatype requirements. This means that MrSID files can be generated from a variety of data sources including GeoTIFF, Imagine, and ECW.

The MrSID technology supports most data types used in geospatial raster imagery today: up to 32 bits per sample (signed or unsigned). Raster image data is almost always represented using unsigned integers. Digital elevation models and file formats like DTED, however, often use a signed integer representation, and so to support situations

where our users want to compress these sorts of datasets, or perhaps use terrain models as base layers for their visualizations, MrSID supports signed integer data of up to 32 bits.

The MrSID technology also supports 1-band grayscale, 3-band RGB, and 1- to 255-band multispectral or hyperspectral imagery.

Image Quality

As discussed above, MrSID technology offers excellent image quality for a given file size target.

- *Numerically lossless*: This level of compression typically yields a 2:1 compression ratio, for a 50% reduction in storage space. Lossless compression should be used when it is critical that all bits of the original image be preserved. This is the case for archival storage, as well as for uncommon workflows where no possible loss of precision is ever acceptable. You may also wish to use lossless compression when you are generating a "master" image from which other derivative images will be made, as through the MrSID optimization process described below.
- *Visually lossless:* This level of compression is typically 20:1 for RGB and 10:1 for grayscale imagery. This is the most common level of compression quality used, as it preserves the appearance of the imagery for most workflows, including use of your imagery as a background layer and for many forms of visual analysis and exploitation.
- *Lossy:* Beyond 20:1, image degradation and artifacts can appear, although often not too significantly until ratios of 40:1 or 50:1. Such lossy quality may be acceptable when the imagery is used only as a background layer for appearance or when the image quality is less important than the storage size or speed, such as for informal visual inspections.

Performance

When considering performance, we usually consider the cost of running some process, such as compression or decompression, in terms of memory usage, CPU usage, and I/O bandwidth. The MrSID technology is designed with these concerns in mind.

- *Compression:* When dealing with very large images, many image processing algorithms first partition the image into tiles and then process each tile independently. This allows the computation to proceed without slowing down due to excessive paging of memory to disk. However, especially in the case of compression algorithms, such tiling can introduce artifacts in the resulting image because the algorithms cannot efficiently process cross-tile regions. MrSID technology is specifically designed to process imagery whose size is larger than the amount of RAM available on the machine without resorting to tiling schemes and therefore without introducing any tiling artifacts. Additionally, the compression process takes advantage of multi-threading to create images more quickly.
- *Decompression:* When decompressing imagery, the most common use case is for viewing, which means extracting out scenes – only some subsets or regions of the image are needed at any one time. With the multiresolution support inherent in the MrSID format, the viewing application may first decide the resolution level needed to display the scene at some physical screen resolution and then extract only the resolution levels needed; this significantly improves disk I/O time and lowers the amount of imagery the CPU must process. Additionally, the viewer need only request those portions of the file that correspond to the region of interest; the entire image (at the given level) need not be processed, again saving I/O bandwidth and processing time.

When decompressing the entire image is required, the performance of the decompression step is roughly comparable to that of the earlier compression step: again, MrSID technology is designed to run within reasonable amounts of

RAM, even for large datasets. If lossy compression was used, the decompression will be somewhat faster since there is correspondingly less data being read in and processed.

Metadata

Because MrSID is a geospatial data format, MrSID files also include geospatial referencing information such as the coordinate reference system (CRS), the geographic extents (corner points) of the image, and the pixel resolution.

This metadata is an inherent part of the MrSID file format and is based on the well-known GeoTIFF tag scheme. When performing a reprojection operation or one of the optimization steps described above, the metadata is updated to reflect the properties of the derived image: when performing scale reduction, for example, the resolution metadata is updated accordingly.

MrSID metadata also is used to record what operations may have been performed on your dataset. For example, you can determine if the file you have still corresponds to the lossless original data or if it has been modified in some way.

This native geographic metadata support allows you use a third-party application to import your MrSID imagery for use as a base map with other georeferenced datasets you might have.

Multispectral Support

For many years, some types of geospatial data have included more than just the usual three color (RGB) bands. Only recently, however, have these kinds of *multispectral* datasets started to be widely available to GIS users. For example, four-band imagery is now the standard for the USDA NAIP program. Additionally, in the commercial sector, Digital Globe deployed perhaps the most powerful commercial satellite to-date, WorldView 3, capturing imagery on 29 bands with a maximum resolution of 25 cm for panchromatic imagery. Other remote sensing platforms are now collecting *hyperspectral* datasets, typically one hundred or more narrow bands. All these additional bands are chosen for their abilities to improve feature classification and extraction by providing more discriminating information in areas such as vegetation cover, shallow-water bathymetry, and man-made features.

To support these new, richer datasets, the MrSID technology can compress images with up to 255 bands. The same key features are still available: lossless and lossy encoding, multiple resolution levels, and selective decoding.

As more data is being encoded and decoded, of course, more time will be required. Figure 1a shows the relative performance of encoding 5Kx5K pixel images with 1, 2, 4, 8, 16, and 32 bands of data: the time required scales linearly, when normalized to the number of bands. That is, if it takes 1 minute to encode a 1-banded image, it will take 10 minutes to encode a 10-banded image of the same width and height.

The time required to decode imagery with varying numbers of bands scales similarly. However, many users of multispectral imagery only view one or perhaps three of the bands at a time, mapping the bands into the familiar grayscale or RGB space. In the same way that the MrSID algorithms will perform selective decompression for viewing only the scene of interest, they will also decode only the bands of interest. Figure 1b shows the relative time it takes to decode 1-, 2-, and 4-band scenes from images with 1, 2, 4, 8, 16, and 32 bands of data: the time required does not depend on the number of bands. More concretely, if it takes 1 minute to extract a single band from 1-banded image, it will take only 1 minute to extract a single band from a 10-banded image of the same width/height.

Alpha Bands

In previous versions of the MrSID format, *no-data* regions were indicated by a sentinel pixel value, typically black. When mosaicking tiles together, no-data regions would be used to indicate how to “combine” one image on top of

another. Users who have worked with MrSID images in the past, however, may have noticed a problem with this. A black no-data pixel, represented by (0,0,0) might be slightly changed when subjected to lossy compression. The value (0,0,0) might change to (1,0,2) or (0,2,0) – by itself visually indistinguishable from black, but in a mosaicking context it is no longer the no-data sentinel value and so in the worst case this might have caused "speckling" artifacts to appear.

The MG4 format uses an *alpha band* instead of a single no-data pixel value to indicate which areas of the image do not have valid data. When encoding existing imagery, users indicate which pixel value corresponds to no-data and a mask is created corresponding to those values. Subsequent mosaicking operations then use that mask to determine how to combine tiles. Lossy compression no longer affects this process, because while the putative no-data pixels might get slightly changed, the alpha mask is always kept lossless and is always honored by the decoders.

The alpha band is treated just like the other bands in the image, such as the RGB bands, except that it is never subjected to any lossy compression. Because the alpha band contains relatively simple sequences of data – very long runs of ones or of zeros – it compresses losslessly extremely well and little or no overhead will be noticeable in your MrSID files.

Tiling and Composites

Many of our customers have a single MrSID file which covers a large geographic region. With the ability of the MrSID technology to *composite* multiple MrSID files together, you can have one MrSID file that is made up internally of dozens of MrSID files serving as image tiles. You can even combine older MG2 files in your MrSID composite image. MrSID composite images are created with a special overview tile, so even files consisting of hundreds of tiles can be quickly viewed at lower resolutions which span multiple tiles.

As new MrSID tiles are acquired – such as from a more recent flight, perhaps with higher accuracy data – these tiles can be easily added to the existing MrSID composite image. Because only MrSID files are involved, this process does not require any decompression or compression steps and so can be done very quickly. When displaying the data, the new tiles' data will correctly layer on top of the older data. Additionally, the overview tile is automatically updated to account for the new tiles.

Differences among the MG2, MG3, and MG4 Formats

As the MrSID technology has evolved over the years, the range of capabilities supported has evolved as well. Specifically:

- MG2 does not support lossless compression
- MG2 does not support composite images
- MG3 does not support 32-bit floating point data
- Only MG4 supports signed integer data
- Only MG4 supports alpha masking
- Only MG4 offers support for multispectral and hyperspectral imagery

While some applications may only write newer versions of the MrSID format, all applications that read MrSID files will always continue to support all versions of the format. These considerations are important to keep in mind, since there are so many older MG2 and MG3 files kept in long-term archives.

IV. MrSID Technology for LiDAR

After what seems like only a few years, LiDAR is now becoming a mainstream GIS technology. LiDAR data is already being used in a variety of areas, including utility corridor monitoring, construction of centimeter-accurate surface and elevation models, and vegetation or biomass measurement.

As with raster imagery, however, this data comes with a cost: file size. A typical day of LiDAR acquisition can result in a dataset hundreds of gigabytes in size and the amount of data collected will continue to grow as sensors improve in resolution and functionality. Large files mean storage hassles and hampered interactive workflows. But technologies like MrSID have been developed to address those problems for raster imagery – can the MrSID technology be used for LiDAR point clouds as well?

A typical LiDAR collection scenario consists of an aircraft with a GPS, IMU (inertial measurement unit), a laser that emits over one hundred thousand pulses per second, and a laser receiver. As each laser pulse is reflected off the ground, it is received by the sensor on the aircraft, which then stores data including the time of the pulse, the inferred (x,y,z) position, and the intensity of the return signal. In addition, we might later perform a classification analysis that identifies what each return represents – tree canopy, impervious surface, water, etc.

This makes LiDAR data fundamentally different from image data in three ways. First, and most obviously, we have a Z height component in LiDAR data. Second, we may also have a set of attributes associated with each point, such as intensity, classification label, or transmitted time of the laser pulse – more than just the pixel value (color) in images. And third, the LiDAR data do not lie on an implicit regular grid, as image data do: each LiDAR point must be stored explicitly, as we cannot infer position based on an implicit grid location.

Although the algorithms differ, the MrSID compression support for LiDAR gives still the same core features we find in the raster world.

> *Selective decompression* – As the point cloud data is encoded, a spatial index is created which keeps track of where the compressed points are located in geographic space, regardless of where they may be located in the MrSID file. This means that the data for a desired region can be accessed and decoded without reading the whole file as may be the case with LAS files.
>
> *Multiple resolutions* – Because the data is on an irregular grid, the multiresolution scheme described earlier for raster imagery is not as effective. Instead, the MrSID file is designed so that applications may request a certain density of points within the region of interest; as further requests are made, additional points may be retrieved. This allows an application to try to achieve a uniform point density across the entire dataset, rather than have an excess of points in some areas when the workflow does not require them.
>
> *Lossless and lossy compression* – The data points are encoded using, again, techniques similar to the MrSID algorithms for raster compression, allowing for typical lossless compression ratios of 4:1. Lossy compression ratios of 10:1 or higher are typical. However, lossy compression is achieved only by removing some points from the dataset, unlike the approximation technique intuitively described above. (Consider that if the decimal value of a point position was "rounded", then the point would actually move in space; this is a more significant issue for high-accuracy elevation workflows than for workflows using the intensity values of imagery.)

LiDAR data is often stored as LAS files, equivalent in spirit to GeoTIFF files in that the data is in an uncompressed, lossless format. It is interesting to compare the cost of extracting a scene from a large LAS file versus from a MrSID file. Figure 2 shows the time it takes to decode scenes from a point cloud encoded in the LAS file format and the MrSID file format. The scenes range from 1% of the total volume of the point cloud up to 64% of the total volume. The time to decode from the LAS file is constant, because extracting data from an unindexed

representation requires walking through every point in the cloud, regardless of scene size. The time required to read the MG4 file is significantly slower for large scenes (over a third of the file, in this case) due to the computational overhead of the decompression algorithms. For smaller scenes, more realistic of many viewing and analysis workflows, the built-in spatial index inherent in the MG4 format dramatically speeds up the data access compared to LAS.

V. LizardTech's Products and Integrations

LizardTech was formed as a spin-off from Los Alamos National Laboratory to commercialize their MrSID compression technology. Since its first commercial release in 1998, the MrSID file format has become an industry standard. Today LizardTech offers a number of products that create, manage, and distribute MrSID imagery.

GeoExpress

The GeoExpress application is LizardTech's flagship product for compressing imagery. It supports creation of MrSID files and includes tools for mosaicking, reprojecting, and color balancing imagery. The latest release, GeoExpress 9, adds support for multi-core processing and floating point data. GeoExpress also fully supports JPEG 2000, an ISO standard technology similar to MrSID technology.

LiDAR Compressor

The LiDAR Compressor application is LizardTech's first product to support LiDAR data with the new MG4 format. The software supports lossless compression up to 4:1 and offers both mosaicking and previewing. It supports reading both the LAS and ASCII point cloud file formats.

Express Server

The LizardTech Express Server system is the best solution for distributing MrSID imagery over the internet. A client connected to a website running the Express Server image server can download and view high-resolution imagery quickly using industry-standard protocols such as WMS. Client applications can include browser plug-ins for the desktop, applications on hand-held devices, or enterprise GIS applications.

The Express Server system can also be used in combination with ESRI's ArcGIS Server and ArcGIS Image Server products to speed up image delivery. You can also publish Express Server catalogs as layers in ArcGIS Server and ArcGIS Image Server, or you can connect an ArcMap client directly to Express Server host so you can work with datasets hundreds of gigabytes in size right on your desktop.

GeoGofer

LizardTech's GeoGofer software provides powerful search and filter features to find imagery by keyword, by projection, by file format and more. You can browse all of your imagery on a single map and open your imagery in other applications easily. Then, make it easy to find the images that you use most often with user-defined tags. Use tags to organize images by project, by recency or by anything that makes sense to you—but never again struggle to find your images.

GeoViewer and ExpressView

LizardTech offers two freely downloadable viewers for MrSID imagery.

GeoViewer is a standalone application for viewing raster imagery, vector overlays and LiDAR data. With GeoViewer you can combine, view and export visual layers from varied sources, such as local repositories, Express Server catalogs, and WMS servers.

The ExpressView Browser Plug-in enables you to view MrSID imagery in Internet Explorer or Firefox. Like GeoViewer, ExpressView enables you to save a portion of an image in a number of other image formats.

LizardTech MrSID Decode SDK

LizardTech offers an SDK to enable MrSID file format support in third-party applications. The SDK exposes C and C++ APIs that enables developers to open raster and LiDAR MrSID files, read the geospatial metadata, and extract scenes of arbitrary size and scale. The SDK is taken from the same code base LizardTech uses to build its own applications.

Our SDK may be downloaded and integrated into your application for free. Versions of the SDK are available for Windows, Linux, Solaris, and Macintosh.

Third Party Integration

Through our SDK program, hundreds of geospatial applications support the MrSID file format. ESRI's ArcMap, ERDAS' Imagine, and Google Earth all support importing MrSID imagery as fully georeferenced raster layers.

VI. Conclusion

LizardTech's MrSID technology is designed to be the most effective storage solution for geospatial imagery and elevation data on the market. We recognize that the quality of your data can be just as important as the cost of storing and accessing it, and we've developed the features of MrSID with that in mind:

- Data quality from lossless to visually lossless to lossy
- Compression ratios from 2:1 to 20:1 and beyond
- Multiple resolutions to obviate manually constructing and managing image pyramids
- Selective decompression for fast access to small scenes in big files
- Support for up to 255 spectral bands
- Support for irregular, 3D data sets like elevation data and point clouds

All these features are available in LizardTech's own suite of products and, through our partners, the MG2 and MG3 format is supported in all the other major GIS applications on the market today. Third-party adoption for the MG4 raster and LiDAR has already begun.

Every year, more airborne and satellite sensors are collecting more and more data, with new spectral bands and higher bit depths. Our software was developed to compress two-dimensional image data, and is now supporting three-dimensional elevation data. As we look to the future, we are starting to see some customer needs for compressing even more types of data such as sonar, SAR, and datasets with explicit time dimensions. LizardTech and our MrSID technology will continue to track the needs of our industry to make it easier to store, manage, and distribute geospatial data.

Appendix A. What is Compression?

Adapted from "Image Compression – Why Is It Now So Important?", Matt Fleagle and Michael P. Gerlek, GeoWorld, September 2008.

Compression makes files smaller while preserving all or most of the data they contain. In imaging, compression enables you to store an archive of many more image files on a disk than you could if they were uncompressed, and to reduce the bandwidth required to transmit them across a network, so you can send more of them faster.

Raster images (satellite images and aerial photographs, for example) are much larger on average than vector images (predominately line-based images like maps and drawings) and other types of files, which is why imaging and compression are such constant companions.

But compression is often misunderstood. Some people assume that compression necessarily entails image degradation, which is not true. Some confuse compression, which is reducing the size of an image file (as measured in bytes), with cropping, which is reducing the dimensions of an image by reducing its scope. There are also several kinds of compression, different algorithms used for different purposes. Finally, choices made in compressing image files have consequences far beyond the moment of compression.

What Does it Mean to Compress an Image?

An image is made up of a certain number of pixels according to its width and height. Each pixel is made up of a certain number of samples, determined by the number of bands (colors) the image has. A pixel from an RGB image has three samples: one each for the red, green and blue color bands. Each sample is in turn composed of a certain number of bits, typically between 8 and 16. Eight bits equal one byte.

The nominal size of an image (in bytes) is its width times its height times the number of samples times the number of bytes per sample. For example, the nominal size of an 8-bit RGB image measuring 1000 x 2000 is calculated as follows:

1000 (width) x 2000 (height) x 3 (samples) x 1 (byte) = 6,000,000 bytes = 6 MB

A compression ratio is the amount or degree of reduction in an image's file size, expressed as the ratio of its nominal size to the target size (the size we intend it to be after compression). Compressing an image at a ratio of 20:1 means that our target size is 5 percent of the nominal size.

Compressed at a ratio of 20:1, the size of the above image is:

6,000,000 / 20 = 300,000 = 300 KB

How Compression Works

Using Fewer Bytes

Compression at a practical level is reducing the number of bytes it takes to communicate the data. We can do this in several ways. As an example, let's consider an array of 16 numbers representing pixel sample values like those discussed above:

9 9 9 11 11 11 9 9 9 10 9 9 9 11 11 11

Alternatively, in an *averaging method*, we would take pairs of numbers starting at the beginning and average them together resulting in the following array of eight numbers

9 10 11 9 9 9 10 11

because the average of 9 and 9 is 9, the average of 9 and 11 is 10, and the average of 9 and 10 is 9 (rounding down to the nearest integer). If these values described a line, the line produced by this array would approximate that produced by the original, but we've reduced the amount of data by a factor of two.

In a *run-length encoding method*, we replace runs of identical values with just two numbers, the value itself and a repeat count. Using again our array of 16 numbers, we would get this:

9[3] 11[3] 9[3] 10[1] 9[3] 11[3]

With a resulting array of twelve items, the run-length encoding method here gives us a byte savings of almost a quarter.

Using a third approach, a *dictionary method*, we remark common sequences of values and replace them with special tokens, storing the sequences and the associated tokens in a dictionary at the top of the file.

Dictionary: A = 9 9 9
B = 11 11 11

Data: A B A 10 A B

Here we've reduced the actual data to just six items, but we have to store the dictionary too.

These are very simplistic examples using a short, unrepresentative input, but we can make some observations. First, it might take some time to encode or decode: think about the time involved in searching for and updating those common sequences, for example.

Second, you might lose data: in the "averaging" example we've discarded data in the rounding operation and can never get it back. Compressed image files can be described as either lossless or lossy. Lossless means that no pixel data has actually been discarded; even if the number of bits it takes to describe the data has been reduced, you can still reverse the process and arrive at the exact same data that you started with. The run-length encoding and dictionary methods above represent lossless compression. That is, using the run-length duets or the dictionary and six data values, you can reconstruct the original string of 16 numbers. The averaging method, however, is lossy. You can't infer the 16 original values from the eight averaged values. The precision lost by that encoding process is lost for good.

Image Quality and Artifacts

Two costs of compression are performance and artifacts. Compressed files may take longer to process (e.g. color balance, further compress) or to decode (e.g. decompress, view). And compression can introduce visible patterns, anomalies or distortions in the image that were not present in the original data.

You might see at least two kinds of artifacts when you look at an image. A mosaic (an image composed of two or more joined images or "tiles") might display seam lines as an artifact of the mosaicking process or tonal imbalances caused by differences in the weather conditions under which each tile was photographed. These are image-level artifacts not related to compression.

Compression artifacts are caused by the mathematical algorithms used to compress images, and different compression algorithms leave different artifacts. Common artifacts include *blocking* (appearance of the image being composed of small squares), *seam lines* (discontinuities along tile boundaries), *ringing* ("echoes" of sharp edges), and *flattening* (small gradients or color changes in otherwise constant areas are smoothed or even removed).

Appendix B. Compression with MrSID Technology

Adapted from "Compressing Lidar Data", Michael P. Gerlek, PE&RS, November 2009.

The MrSID technology relies on two key algorithms to encode data. First, a *wavelet* is used to change the data into a form that serves to represent the data at multiple resolutions. The wavelet transform is a mathematical transformation of the data that is, in general, lossless. The second step is to encode the data produced by the wavelet transform into a new representation that is significantly more compact than the original and yet still lossless; optionally, this step may also reduce the precision of the data for even more space savings at the cost of incurring some data loss.

Wavelet Encoding

Wavelet encoding can be thought of as nothing more than a weighted averaging and differencing scheme. Consider a list of 4 numbers: {2, 6, 14, 6}. By pairwise averaging, we can reduce this to a "second level" list of two numbers, {4, 10}, and a list of second level "differences", {+2, -4}. Performing the same operation again on the {4, 10} list, we get the third-level list {7} and the third level difference list {+3}. This is shown in Figure 3.

These three lists, and the two associated difference lists, are used to represent the original four-element list at three levels of resolution. Observe that only the final singleton list {7} and two difference lists of {+3} and {+2, -4} are required to perform the inverse operation and restore the original input list. With only these three lists, we can reconstruct the second level: the third level list {7} and the difference list {+3} yields {4, 10} (because 7-3=4 and 7+3=10). The second level list {4, 10} and the second level difference list {+2,-4} yields in turn the original list (4-2, 4+2, 10-(-4), 10+(-4)).

Although the details are somewhat more complicated in practice, the principle is the same for image data. Using a 1D wavelet across both dimensions of the 2D array of pixel data, we reduce the image to a pyramid of images, each level being half the width and height of the previous level and including the corresponding difference list. This is a mathematically lossless process. No significant compression has been achieved, but we have a better representation of the data because it inherently reflects the data at multiple resolution levels. This is similar to the well-known manual image pyramid scheme, which has power-of-two sequential reductions in resolution. These reduced resolution decodes fall directly out of the wavelet decomposition, making them easily accessible.

In the case of LiDAR data, the situation is somewhat different because raster datasets are regularly gridded while point cloud datasets are not. In the former case, we don't need to explicitly encode the x and y coordinates, just the (pixel) values at each grid point. In the latter case, having no implicit reference grid, we must encode the coordinates and any associated values (such as intensity). In a decompression at less than the full resolution of the data, a subset of the pyramid levels are used to compute the data to be returned; the resulting point cloud is effectively subsampled.

Arithmetic Encoding and Bit Planes

The points produced by the wavelet decomposition are grouped into spatially adjacent blocks that can now be encoded independently. Within each block, we consider the data as sets of significant bits: the most significant bit of each piece of data forms the first "bit plane", the second most significant bit of each item forms the second bit plane, and so on. We use an encoding technique known as arithmetic encoding to efficiently (and losslessly) compress these strings of bits. It can be thought of as using winzip or gzip on a small binary array, although it is more efficient in general.

If we wish, as part of this process we may choose to remove a number of least significant bits from each point. For example, a block of numbers such as {314, 572, 419, 845} nominally requires twelve (four times three) digits of

representation. However, if stored as {31_, 57_, 41_, 84_}, where _ represents an omitted digit, only eight (four times two) digits are required, and if stored as {3__, 5__, 4__, 8__}, only four digits are required. Clearly 310 is only an approximation of 314; such is the nature of lossy compression – space is traded off for precision.

Again, this description is only to provide some intuition into the actual process but the key properties are the same. We have reduced the storage requirements of our data, and can adjust the amount of loss we are willing to accept. Lossless means not throwing away any bits, lossy means we throw away least significant bits. For lossy compression one cannot simply remove arbitrary bits from arbitrary blocks, however, statistics are kept to determine the relative "importance" of each block (based on resolution level and potential error), and thus less important blocks are more likely to lose bits.

We also have processed our data in blocks, allowing for spatially adjacent points to be stored (and accessed) independently. This gives us our selective decompression property, as we only need to access (and decompress) a subset of the entire file if we are only interested in a subset of the entire spatial extent of the point cloud.

Putting It Together

The wavelet transform followed by the bit plane encoding gives us a representation of the data in the file format that offers some of the capabilities we have discussed in this white paper. For example, a MrSID file can be further compressed (optimized) by simply considering each already-encoded bit plane and truncating its precision by the degree necessary to achieve the smaller target file size. Selective decompression can be achieved by selecting only those blocks of bits relevant to the scene (and resolution scale) and then performing the *inverse* of the bit plane encoding and the wavelet transform on them. As long as no bit planes have been truncated (incurring data loss), these inverse transforms are numerically lossless, thus ensuring you will get back exactly the original data you started with.

Figures

Figure 1a: Time to encode an image with a varying numbers of bands

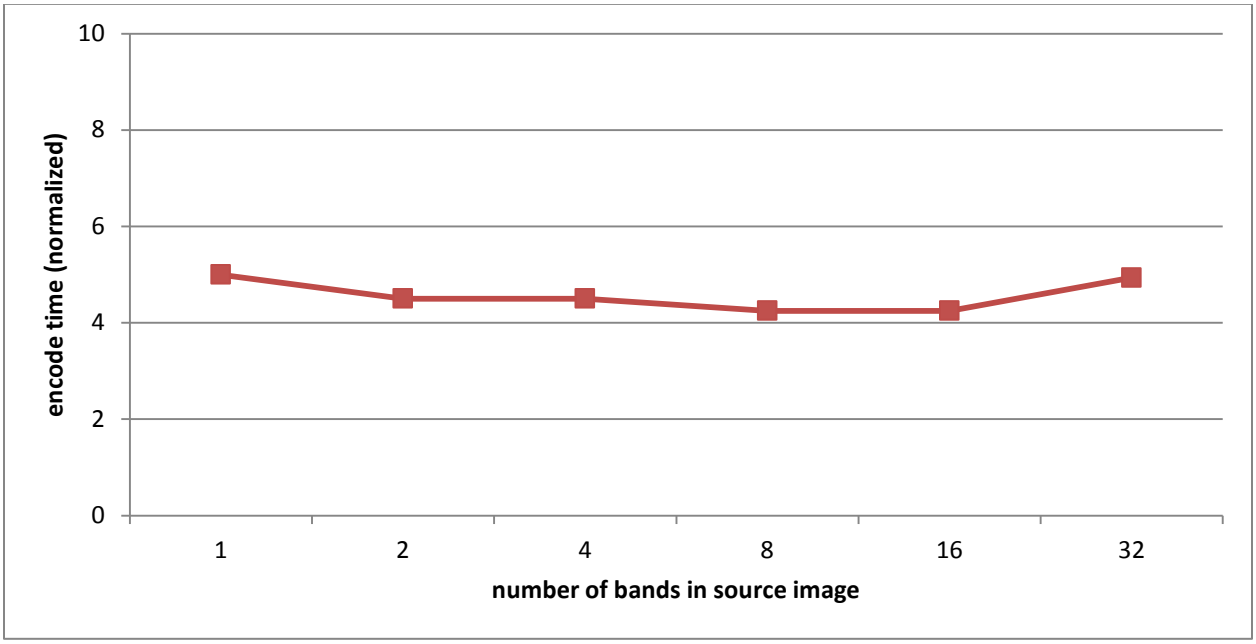


Figure 1b: Time to decode an image with a varying numbers of bands

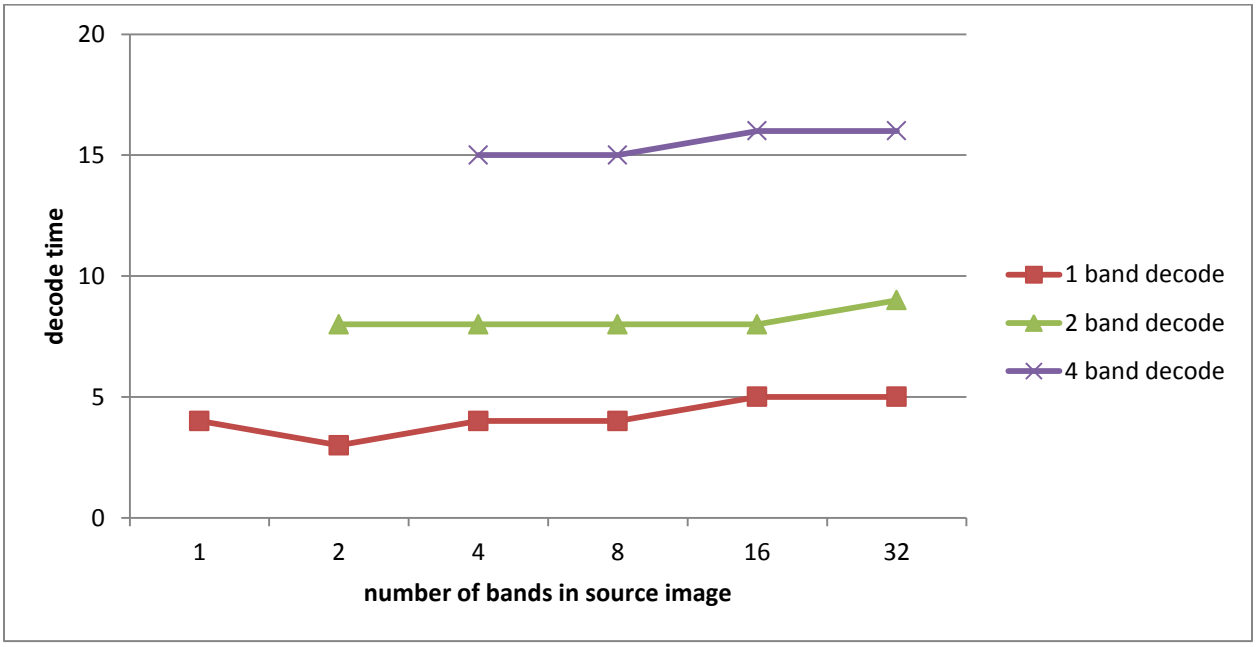


Figure 2: Time to decode various scene sizes from a LiDAR file, using LAS and MG4 formats

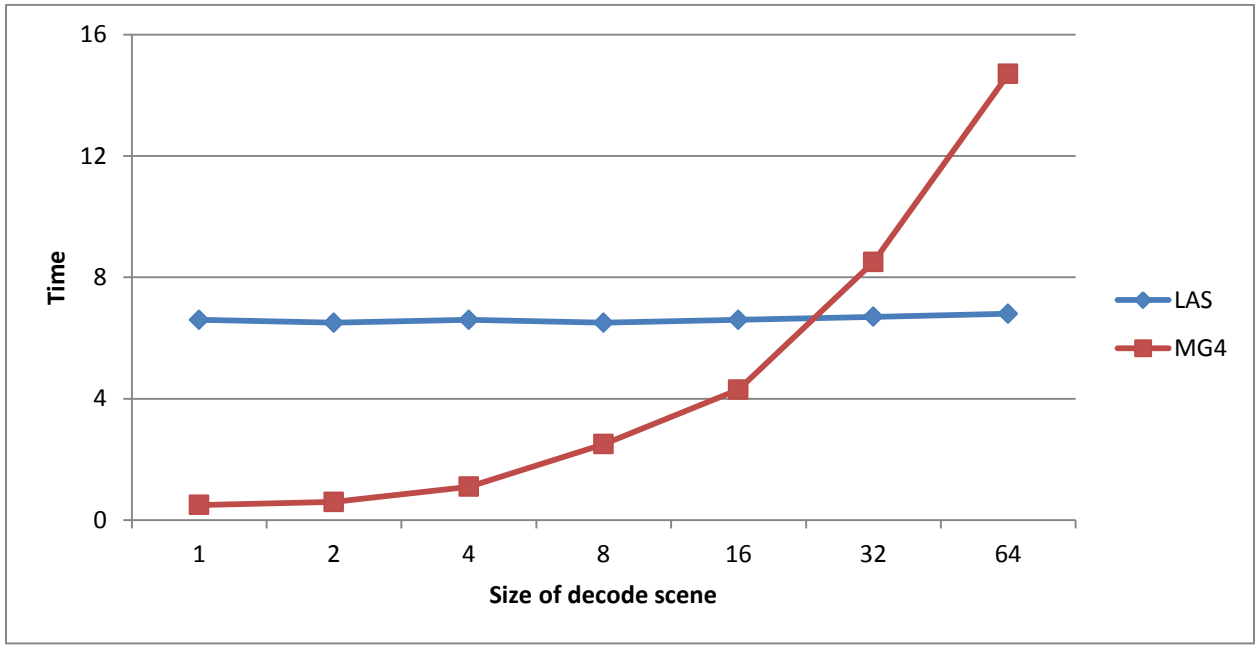
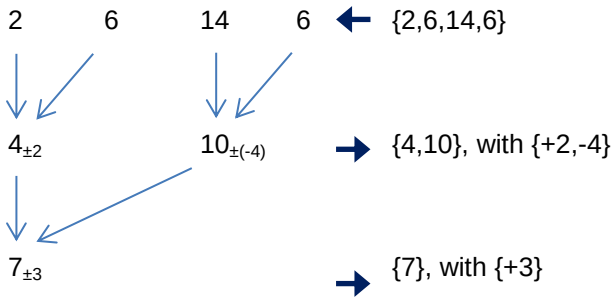


Figure 3



LizardTech
1008 Western Avenue, Suite 200
Seattle, Washington 98104
www.lizardtech.com