Aerial and Mobile Point Cloud Integration – An Industrial Perspective

LEWIS GRAHAM, Huntsville, AL, USA

Abstract

It is a great privilege to have been invited to deliver a paper at the 53rd Photogrammetric Week. My initial association with this event came with the merger of Intergraph's and Zeiss' photogrammetry divisions, forming Z/I Imaging in the second half of the 1990s. It is great to be back here once again, now as the manager of companies involved in geospatial workflows and laser scanning tools.

When Dr. Fritsch sent me the invitation to this event, he also included the suggested subject and even the title! This was good since titles, much less subjects, can be tricky. The subject is merging point cloud data but notice that Dr. Fritsch added the line "an industrial perspective." I thought about what this means and concluded that it lets me give a paper without any math. This is also good since I think I have forgotten all that I once knew (which was not all that much to begin with). I am really going to take advantage of this position and offer only approximate thoughts in this presentation. Of course, what he actually might have meant was that he was after a paper that would provide some insight into what is commonly being done within production shops, whether it is correct or not, and I hope that I address this adequately as well. Finally, I will only offer reference citations when I think it might actually be worth your while to review the referenced document.

INTRODUCTION

In a bit, I will review in some detail a point cloud. For now, it is sufficient to know that a point cloud is a set of data tuples in three spatial dimensions[1] that is not organized in a uniform pattern with respect to any of the spatial coordinates; hence the terminology "cloud." These point cloud organizations of data are not really new since they can be derived by correlating conjugate points in overlapping images. However, our discussion will focus on point clouds that originate from the post-processing of laser scanned data.

If we want to talk about merging clouds from one or more sources, we need to first ask for a motivation. We might think along the lines of:

1. Increasing the density of an elevation model ("thickening")
2. Increasing the accuracy of the model
3. Increasing the available perspective views of the model
4. Using one point cloud to fill in voids ("holes") in another point cloud
5. Extending the horizontal range of an elevation model (similar to 3 above but extending from an edge rather than filling in a hole)

Of the above, all are obvious except perhaps (3). This is the case where we might have an aerial system with a near-nadir scanner collecting vertical objects (for example, vertical signs in a railroad data collection application). A mobile scanner collects in a 360° field of view and therefore will impinge the vertical object with perpendicular rays. This difference is illustrated in Figure 1.

[1] I don't consider time to be another dimension but simply an overlay. This lets us have N time layers arranged in the same space – a more useful concept for geomatics than extending the dimensionality.

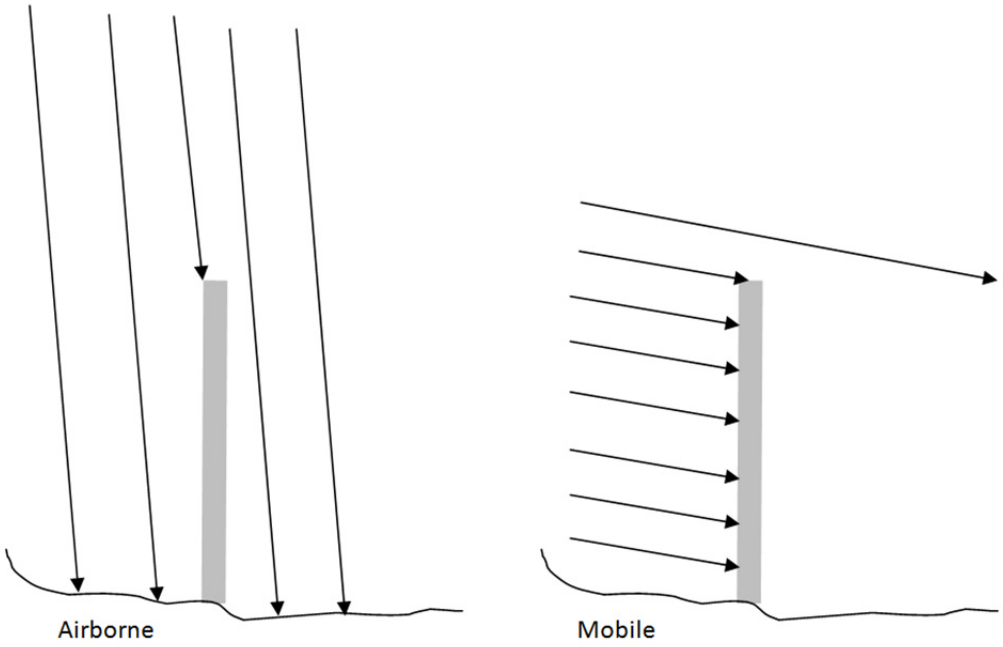

Figure 1: Airborne vs. Mobile Collection Perspective

POINT CLOUDS

Once when I was teaching a basic signal processing class, I skipped the development of the Euler relation (you know, the series expansion in the complex plane of the trigonometric functions as the base of the natural logarithms) because I was afraid that I would offend the students by assuming they did not have this basic knowledge. This turned out to be a big mistake since it left those a bit rusty on the topic at a distinct disadvantage when I started in to developments that relied on this prior knowledge. Therefore, I will provide some background information on laser scanned point clouds. If you know this already, skip on to the next section!

Figure 2: Airborne Laser Scanning

An airborne kinematic laser scanning system is illustrated in Figure 2. The collected data are range data collected in a near-nadir geometry (perhaps deviation of up to say 40° from the nadir).

Most systems function by firing a short duration laser pulse and collecting the return intensity from reflective surfaces. The detector generally integrates the return energy with respect to time and makes a "decision" as to where to declare (in time) the "return."

Most modern systems will perform this integration multiple times for the same outgoing pulse and thus can detect multiple returns per pulse (e.g. the initial return from the surface of a tree canopy, an additional return from a lamp post under the tree and a final return from the ground). This concept is illustrated in Figure 3.

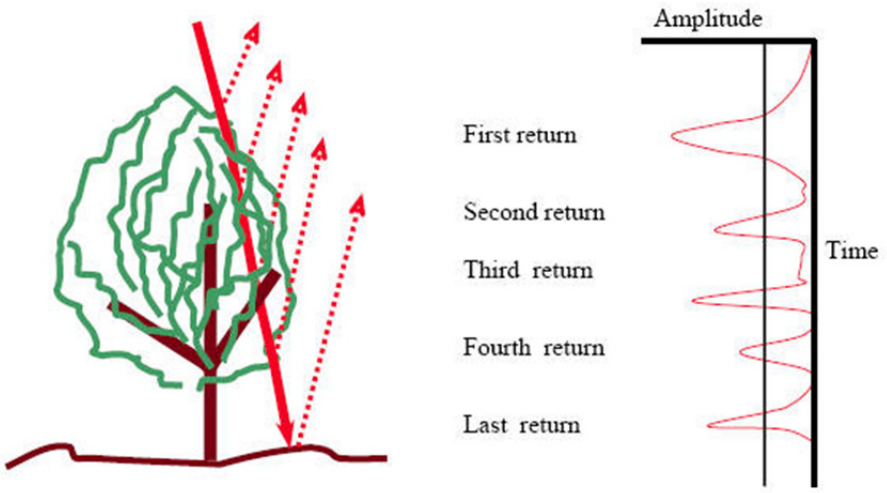

Figure 3: A multiple return laser scanner

The scanning patterns are usually either side-to-side using an oscillating mirror or in a helical pattern using a two axis scanning mirror. In both cases, the along-track density is grossly determined by the forward aircraft speed.

Figure 4: Typical Mobile Scanner

Mobile kinematic scanners operate in much the same way as airborne with the exception of the scanner sensors. Modern geomatics quality scanners are 360° rotating scanners with (commonly) two scanners on the platform. These scanners are mounted at 45° with respect to the direction of motion of the vehicle. This provides very dense coverage in a circular pattern centered on the vehicle track. A typical configuration is depicted in Figure 4.

A point cloud is generally 3 dimensional data tuples that are not organized on any particular uniform spacing. A close range point cloud is depicted in Figure 5 and Figure 6. I say data "tuple" since in most rich data environments, each point carries attributes in addition to the geometric location.

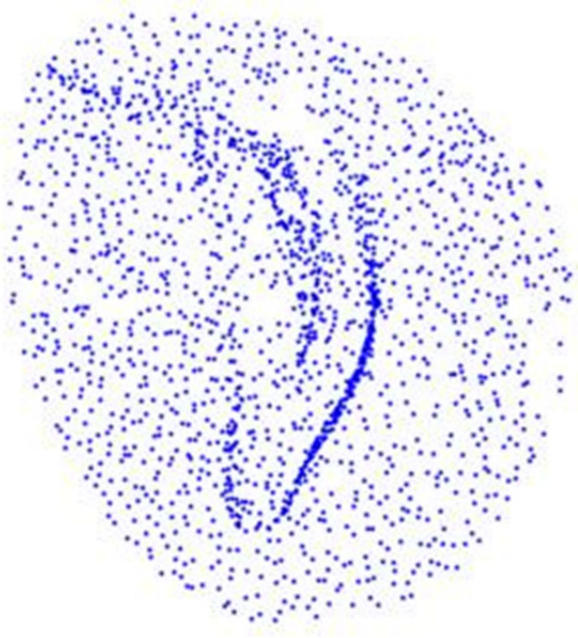

Figure 5: A 2D projection of a point cloud

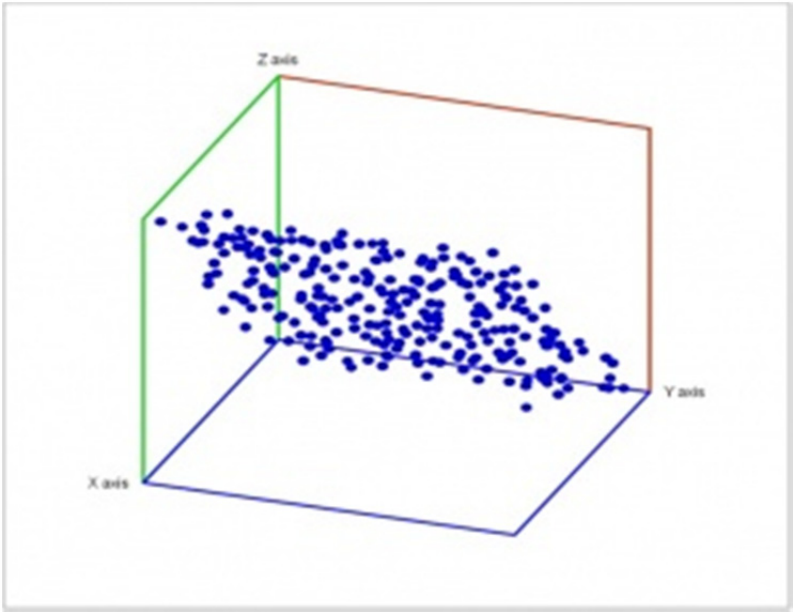

Figure 6: A 3D perspective view of a point cloud

An important (and complicating) aspect of a point cloud is that the data do not contain any inherent topology. That is, we generally do not know the relationship (if any) of a given point to its neighbors.

Now we sometimes read of a point cloud "represented as a Triangulated Irregular Network (TIN)"; see Figure 7. This statement is generally not correct since imposing a TIN on a point cloud is adding a constraint geometry that, in the real world, may not exist. Thus I think of a TIN as a data model that has been "generated" from a point cloud rather than as a representation of the cloud. This is a subtle and important difference. In fact, if I think of the physics of the process, a TIN decreases the entropy[2] of a point cloud, even if all of the points are consumed in the TIN.

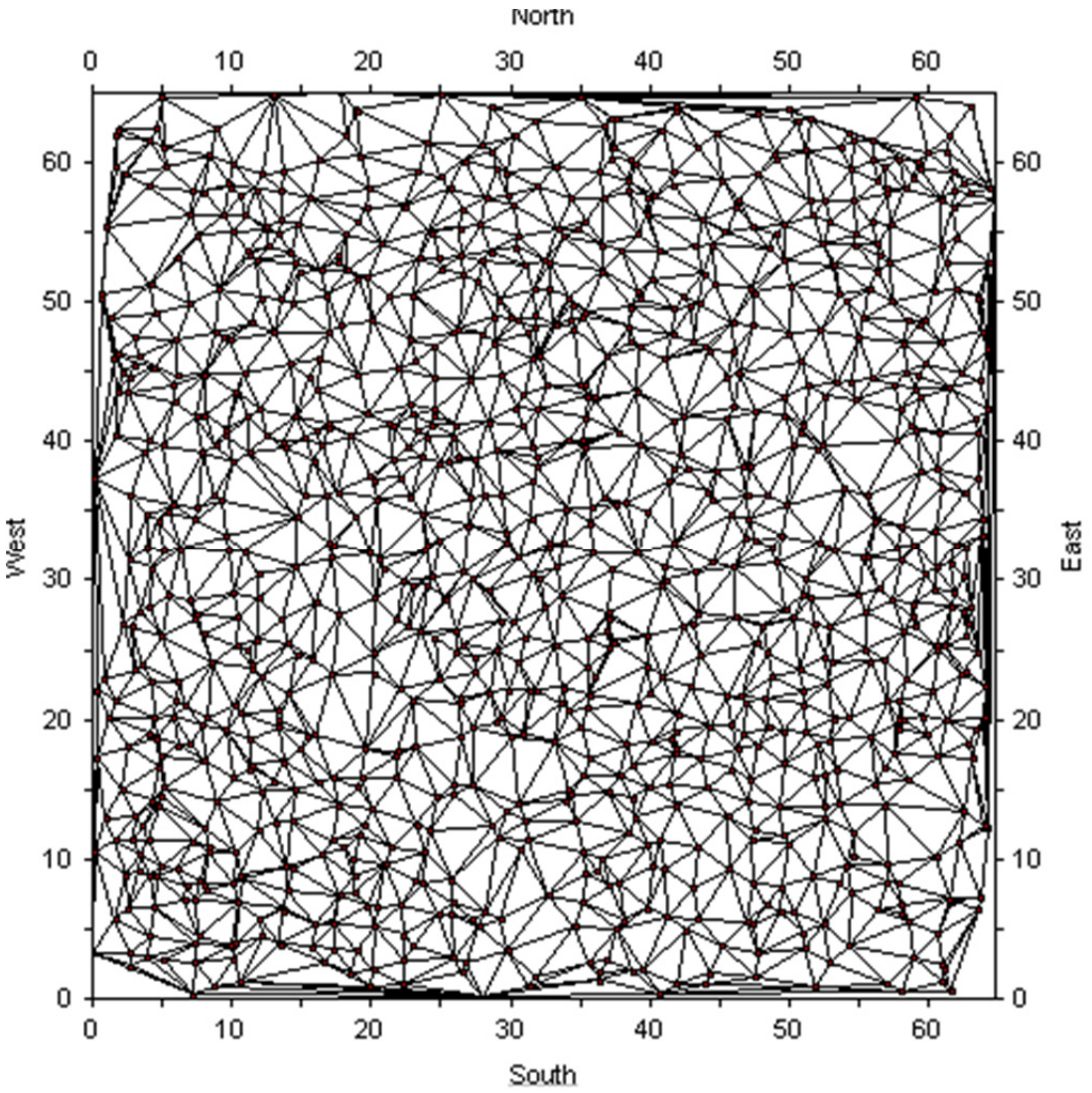

Figure 7: A Triangulated Irregular Network (TIN)

[2] Here I am using entropy as a measure of the randomness of the informational content of the data set as in *Shannon Entropy* (Shannon, 1948)..

ACCURACY OF KINEMATIC LASER SCANNER POINT CLOUDS

Points clouds that are derived from kinematic laser scanners are generally realized by combining laser range data, laser time-dependant internal positional information (i.e. the location of the scanning mirror), platform calibration information and platform positional information (generally derived from a dynamic position and orientation system or POS). Since there is generally no rigid geometric constraint from point to point in a laser scanner, these systems present a more difficult system to model (and hence subject to higher error) than do line scanners (which, in turn, have fewer constraints than framing imaging systems). As such, error propagation models are still somewhat in their infancy for these systems [citation needed].

For the sake of this paper (which, recall, is simply a high level overview without the mathematical rigor that would be necessary for actual use!), we might consider the total error of a single point within a cloud to comprise a number of components:

ε_R	Random Error	(1)
ε_C	Platform Calibration Errors	(2)
ε_T	Trajectory Errors	(3)

The random errors (ε_R) are components that we assume to be stationary[3] and usually Gaussian. A good example for a laser scanner is the range error of the laser. For a mobile system, this might be a zero mean, 5 mm one δ error, assumed Gaussian.

Platform calibration errors (ε_C), in this paper, are analogous to the traditional platform errors we would deal with in photogrammetry. That is, errors between a platform reference frame and the various components. We will consider our system be of sufficient quality that the calibration will only vary over a project's duration in a predictable way (meaning that it can be captured in a mathematical model). Thus they comprise things such as:

- Lever arm errors
- Errors regarding the internal model of the laser such as mirror position
- Stationary timing errors
- Errors that can be modeled as temperature dependent
- etc.

Finally, we have the trajectory error, ε_T. This represents errors between the platform reference system's true location within a global reference frame (e.g. International Terrestrial Reference Frame, ITFR) and the platform location derived from the POS. Of course, short of having some external measuring device, we cannot know this true location and hence must estimate from the POS data. Now the problem with this is that the POS errors are, in general, not constant and, even worse, they are non-stationary. Most high end POS's provide estimates of these errors in terms of variance tuples for each position/orientation tuple (note that since we are provided a variance tuple per every position/orientation tuple, the manufacturers are suggesting the non-stationary nature of the errors). I won't delve any further into this error source in this paper. Suffice it to say that it is the error source that really throws the wrench into the works when working with point clouds!

[3] By Stationary, I mean that its statistical characteristics (mean, variance, etc.) are the same across the data set and furthermore, that if I return to a point in space, our sample characteristics will obey the same model as they did on the first visit (note that I am not saying the errors would be the same).

Note that we cannot sum our error sources to produce a net error. Rather we simply note the total error as some function of the contributors:

$$\varepsilon_{Total}(i) = U\{ \varepsilon_R(i), \varepsilon_C(i), \varepsilon_T(i) \} \qquad \text{Total System Error at point i} \qquad (4)$$

Now the sad state of laser scanning today is that, even though we can nicely write an equation for the total error at a point, we do not have a developed math model to actually implement this error model!

In my opinion, we should at least put our place holders into the point cloud model as variances on each of the 3 location parameters (X, Y, Z) and throughout the following sections I will pretend that I have these values. It is important to note that these variances would probably not represent a Gaussian distribution, making it difficult to make statements about error centering and percent confidence about the location.

POINT CLOUD STORAGE AND TRANSFER FORMATS

There are two common transfer formats for LIDAR point clouds that are produced by kinematic scanning systems[4]; ASCII and LAS. Transfer formats such as extensible markup language (XML) are not practical due to the data volumes involved (typically millions of points per file).

When encoding laser scanner data in ASCII formats, any fields desired can be included so long as the sender and receiver agree, in advance, as to the exact format and layout of the files. To standardize the fields as well as compact the storage into a binary stream, the American Society for Photogrammetry and Remote Sensing (ASPRS) maintains an open point cloud format called LAS. Now in its fourth revision (LAS 1.3), LAS includes a number of file-wide metadata elements (coordinate system, scaling, number of points and so forth) as well as rich, per-point information. Among the per-point fields are:

- Absolute GPS time for this point
- X, Y, Z (the location of the point
- Intensity
- Return number n of m returns (e.g. Return 2 of 3, return 1 of 1, etc.)
- Mirror angle
- Edge of scan flag
- Classification
- etc.

Unfortunately (or fortunately, depending on your point of view), LAS contains no standard accuracy information at all. As LAS has developed (the author of this paper is the chairperson of the LAS Working Group), the idea of adding fields for per-point accuracy data has surfaced time and again. It has not yet made it in to the specification because the committee cannot reach a consensus on populating the fields! You can appreciate this dilemma based on the discussions of the error contribution section of this paper. As will become evident in a later section, we cannot really hope to accurately merge laser point clouds until these per-point parameters are included.

[4] There are an entirely different set for stationary scanners (tripod systems).

For the sake of discussion in this paper, I will pretend (I cannot assume since what I am pretending generally does not exist!) that we do have an error propagation model for the generator of the point cloud and that, for each point P_{we} we have the following data tuple:

X_i	The X coordinate of P_i	(5)
$\delta_{x,i}$	The variance of the x coordinate of P_i	(6)
Y_i	The Y coordinate of P_i	(7)
$\delta_{y,i}$	The variance of the y coordinate of P_i	(8)
Z_i	The Z coordinate of P_i	(9)
$\delta_{z,i}$	The variance of the z coordinate of P_i	(10)

It is critically important to note that we are not saying anything about the underlying distribution of the errors (that is, we are not saying that these errors obey a Gaussian distribution).

DIGITAL ELEVATION MODELS

Unfortunately, our industry has never converged on a specific meaning for the term Digital Elevation Model (DEM). About the only agreement seems to be that it is a regularly spaced grid of elevation (Z) values that represent some aspect of the underlying topology (surface, bare earth, etc.). Perhaps a better name might be Digital Elevation Matrix and you should think matrix each time I say DEM in this paper.

DEMs are used in representing a surface mainly to make processing "easy." Common operations such as interpolating an elevation at an X, Y location that is not the X,Y of a particular point in a model is quite straightforward with a DEM but can be rather involved when using a TIN and devolves almost to a point by point search when using a non-indexed point cloud. An additional reason for using a DEM is the compression that arises from quantizing the X, Y locations of representation. Since the DEM locations in X and Y are specified *a priori* (meaning we decide upon an X and a Y spacing and this spacing is uniform throughout the DEM), the X, Y values are not stored. Finally, matrices have been much more amenable to common signal processing techniques such as Fourier transforms.

MODELING DATA

It is very constructive to think about what we are trying to achieve when we set out to build a model of the 'real' surface of the earth. At the most basic level, we want a (hopefully compact!) way to specify any x, y location within our area of interest and have a way to determine, as accurately as possible, the elevation at that specified point.

$z = G(x, y)$	Our elevation lookup function	(11)

At this basic level, we give an x, y location to our little daemon; he pops into the elevation box and comes back with a z. What is contained in the box and how the daemon determines the elevation basically determines the model (what's in the box) and the interpolation approach (how he determines the z).

The most direct (and expensive and time consuming – you see these tradeoffs in design, of course!) is for the daemon to visit the site with surveying equipment and determine the Z at precisely the

desired X, Y location. This is, of course, the "infinite" model in that the entire surface to infinite resolution must be available to our daemon.

The above is simply a thought process, of course. The infinite model is not useful for general purposes so we must create a model that reduces the storage and access times. For example, we might overlay a gird (matrix) over our area of interest and assign some value to each grid location. It is evident then that the "fineness" of this grid will determine the resolution and accuracy of our model. So how do we build this model?

There is a remarkable result, probably first formalized by Joseph Fourier in 1822, that gives us a handle on this modeling problem. This says that a "signal" that obeys certain restrictions (the Dirichlet conditions) can be exactly reproduced by the integral of sine and cosine function. Now this is very useful from an analytic perspective but does not help with the data compression/representation problem. A better implementation for our purposes is the *discrete Fourier series.* This result says that for a *periodic*[5] *signal*:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos\frac{n\pi x}{L} + b_n \sin\frac{n\pi x}{L} \right) \tag{12}$$

Now this is a pretty remarkable result. It says that if we have a general function (here we have a function of one dimension but it extends in a separable way to x,y) we can represent this function by a series of sines and cosines, each multiples of a fundamental frequency (in our case, spatial frequency) with discrete multipliers (amplitudes). This means that we can simply store the a_{we} and b_{we} coefficients.

Now a big problem with our work thus far is that this series is infinite; not very useful! Beginning with work by Nyquist (Nyquist, 1924) and culminating in work by Claude Shannon and others (Shannon, 1949), it has been shown that, under some very specific conditions called the "Nyquist Sampling Criteria" (which I will review in a moment), the above series can be truncated. This sampling criteria says that if our original signal is "band limited" and we sample the signal at a rate that is at least twice the band limiting frequency, we can truncate the Fourier series representation and *perfectly* reconstruct the original signal. (17)

$$f(x) = a_0 + \sum_{n=1}^{N} \left(a_n \cos\frac{n\pi x}{L_0} + b_n \sin\frac{n\pi x}{L_0} \right) \tag{13}$$

This *still* is not all that useful to us because we need to know the function f(x) over the entire area of interest in order to compute the coefficients a_n and b_n. For example, to find b_n we have:

$$b_n = \frac{2}{L_0} \int_{-L_0/2}^{L_0/2} f(x)\cos(n\omega_0 x)dx, \qquad \omega_0 = \frac{\pi}{L_0} \tag{14}$$

[5] For finite signals (e. g. a finite area over which I am examining the terrain), I can ignore this requirement for periodicity by simply considering 1 period to be the extend of our area of interest.

So, if we regroup, we see that we would like to have:

- A reconstruction function that will generate Z(x,y)
- The reconstruction function created not from the continuous elevation surface but from some *samples* of the surface.

I won't go in to a lot of detail here but this process is really a digital sampling problem. The basic operation of sampling is depicted in Figure 8. Here I illustrate one dimension but the concept directly extends to multiple dimensions. This is really the same problem as that faced in Compact Disc music. We want to sample a real world piece of music, record the samples digitally and then be able to reconstruct the music from the digital samples. In our case, we are sampling a terrain surface using some point sample technique such as moving about with a Real-Time Kinematic (RTK) GPS data collector, LIDAR, photogrammetric stereo correlation or some other similar technique. The important point (pun intended!) is we have only these samples with which to work.

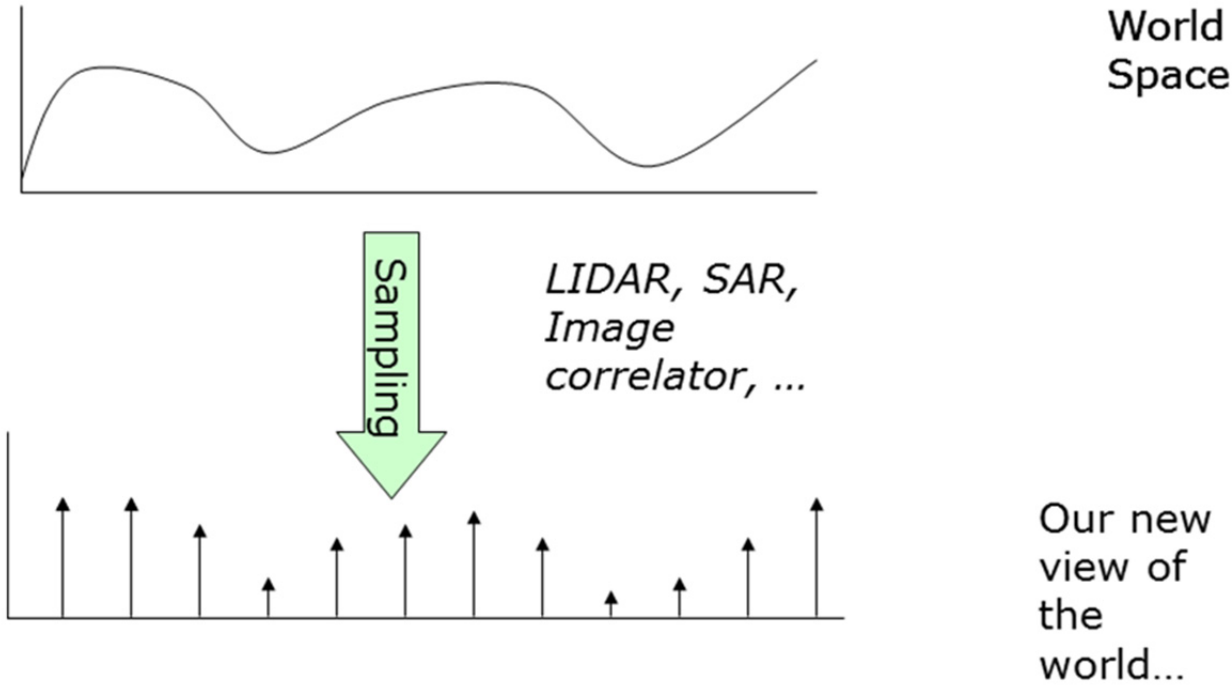

Figure 8: Digital Sampling

THE SAMPLING THEOREM

I will not go through the development (see, for example, Lathi, 1998) but the Fourier techniques of the previous section along with our sampling desires lead directly to the sampling theorem. I am going to express this result in the time domain but understand that I simply replace time with the dimension I desire (e.g. x or y) for spatial sampling.

Assume we have sampled our signal with a spacing of T between samples (for example, we have sampled every 10 milliseconds). This means our k^{th} sample was collected at time = kT. So if we were simulating this with a function f(t), we would have an ensemble of data points, f(kT) with k running from zero to the total number of samples minus one.

If we obey a criteria called the "Nyquist Rate" when selecting the rate at which we collect (we'll discuss this in a bit), then we have the truly remarkable result:

$$f(t) = \sum_k f(kT)sinc(2\pi Bt - k\pi), \qquad where\ \ sinc(x) = \frac{\sin x}{x} \tag{15}$$

and

$$B = \frac{1}{2T}, the\ bandwidth \tag{16}$$

It is very important to examine (15) in some detail. Note that while it is amazing that we can completely and exactly recover[6] a signal *f(t)* at *any* point *t* from a set of samples taken at discrete points *f(kT)*, we have to use <u>all</u> of the samples each time we reconstruct a point. For example, if we take 1,000 samples spaced 10 milliseconds apart, then T is 0.01. Suppose further that we start our sampling at t = 0. This means we have a sample at t = 0.0, t = 0.01, t = 0.02 and so forth. Now suppose we want to know the value of our signal at t = 0.234. Examining (15), we see that we merely substitute 0.234 for *t*. The discouraging news is that we will have to sum over all 1,000 samples to achieve a precisely correct result[7]! Obviously this is not going to be practical and thus we will have to make some approximations. The important thing here is that we can see that, in general, the fewer the terms we use in our reconstructor function (often referred to as a "filter kernel") the farther we will deviate from the true value of the original signal.

The sampling theorem is separable and extends in the natural way to two dimensions: (see, for example, Steeb, 2005)

$$f(x,y) = \sum_k f\left(kL_x, jL_y\right)sinc(2\pi B_x x - k\pi)sinc\left(2\pi B_y y - j\pi\right), \tag{17}$$

where

$$B_x = \frac{1}{2L_x}, B_y = \frac{1}{2L_y}\ \ the\ spatial\ bandwidth \tag{18}$$

[6] For this reason, this function is often referred to as the "interpolation function."
[7] Note that I am taking some big liberties here. I am ignoring effects caused by the fact that I have performed sampling over a finite interval and a few other details that are not material to our discussion.

Here we have *spatial* bandwidth rather than time domain bandwidth. Those from the photogrammetric camera discipline will recognize spatial bandwidth as, for example, line pairs per millimeter.

ACHIEVING THE NYQUIST RATE

In order to ensure that we meet the Nyquist sampling criteria, we must either know the maximum frequency contained in the signal *a priori,* or preprocess the signal prior to sampling. Limiting the bandwidth is accomplished by low pass filtering the signal prior to sample collection. The general case is illustrated in Figure 9. This pre-filtering will turn out to be critically important when we look at merging elevation data of different sample spacing.

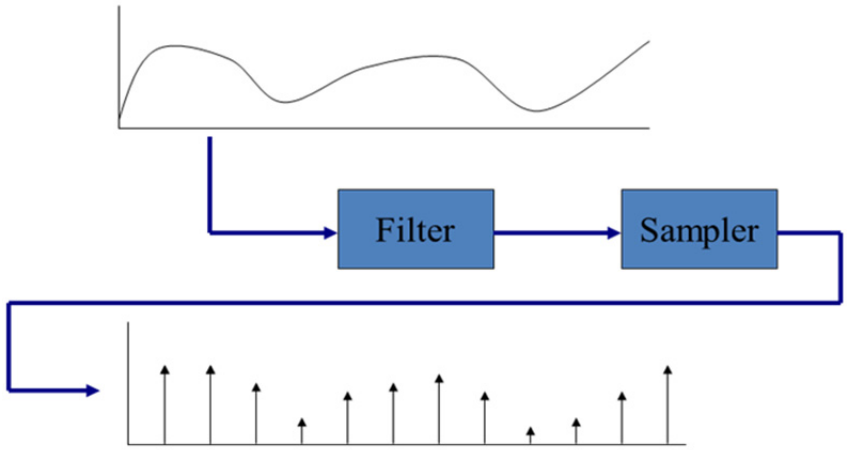

Figure 9: Filtering prior to Sampling

There is a wonderful and fairly simple proof based on the Fourier spectrum of the need for achieving the Nyquist rate in a sampled signal (see, for example, Shannon 1949). Basically it shows that if we do not sample at a sufficiently high rate (> 1/2B where B is the bandwidth), then high frequencies will masquerade as low frequencies in our samples. This phenomenon is referred to as frequency folding or *aliasing*. An example is depicted in Figure 10. On the left we constructed an image of varying spatial frequency. In a grey level image, spatial frequency is related to the rate at which the grey levels vary spatially. Examining the image on the right (the source image) from the lower right diagonally up to the upper left, we see that the image has very low frequency, jumps to a fairly high frequency in the middle and then to a mid frequency in the upper left. The aliased image on the right indicates that our sampling appears to have been high enough for the upper left but inadequate for the central section of the image. Thus we see a dramatically different image being generated by inadequate sampling.

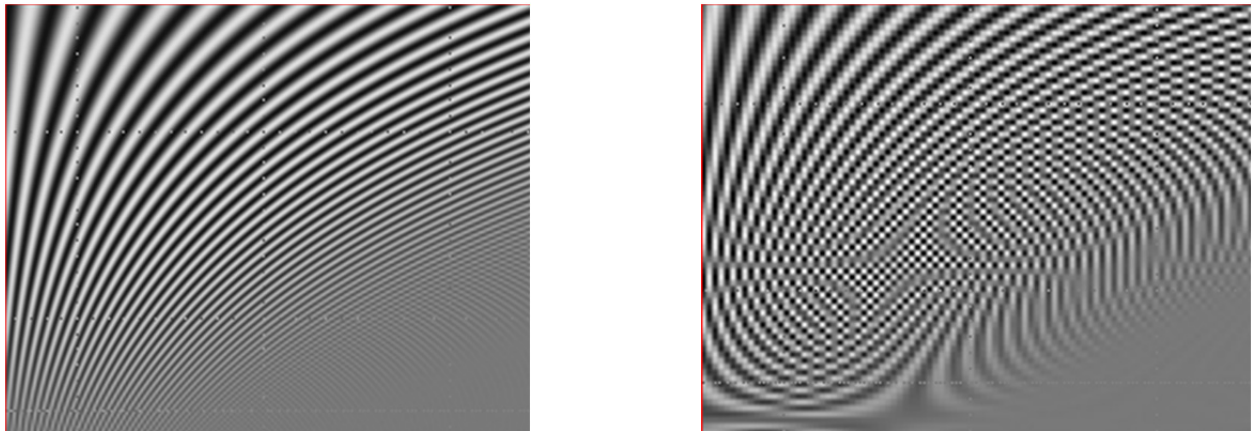

Figure 10: Original Image (left), Aliased Image (right)

We can look at this in a more intuitive fashion by examining the sampling of a dramatically varying (in the spatial frequency sense) cultural feature. In Figure 12 I depict a levy being sampled at such a low rate that only a single sample point has encountered the feature. In Figure 13, I depict the effect of increasing the sampling rate.

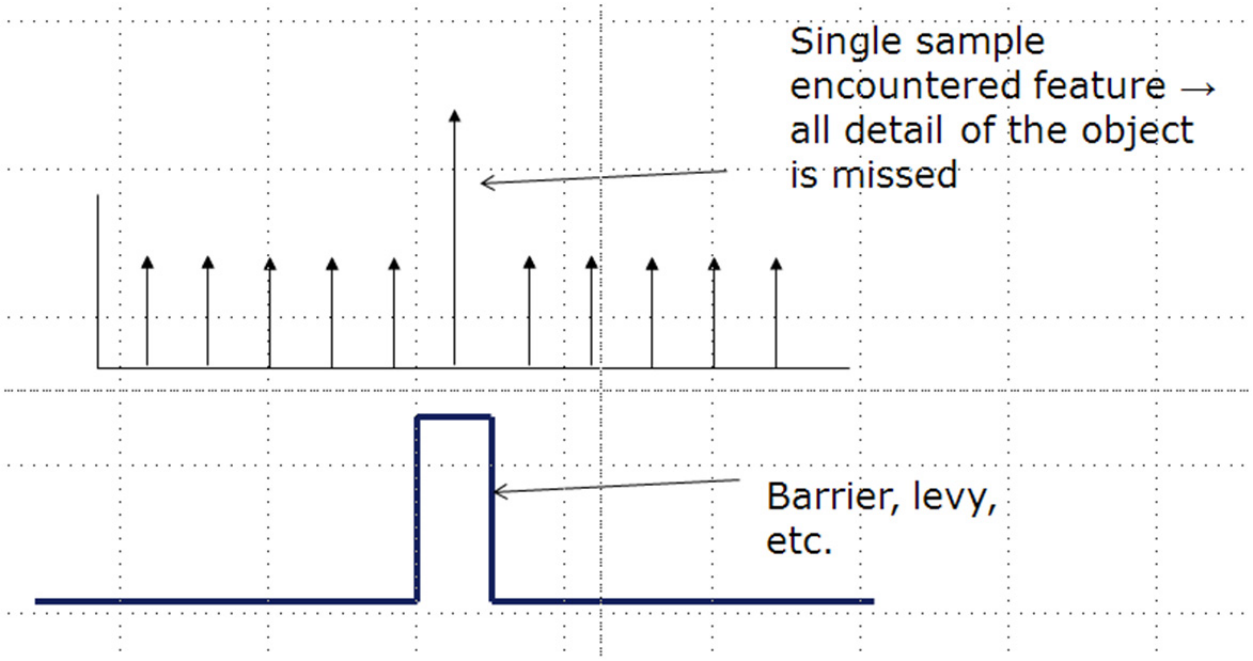

Figure 11: Sampling a levy at a low sampling rate

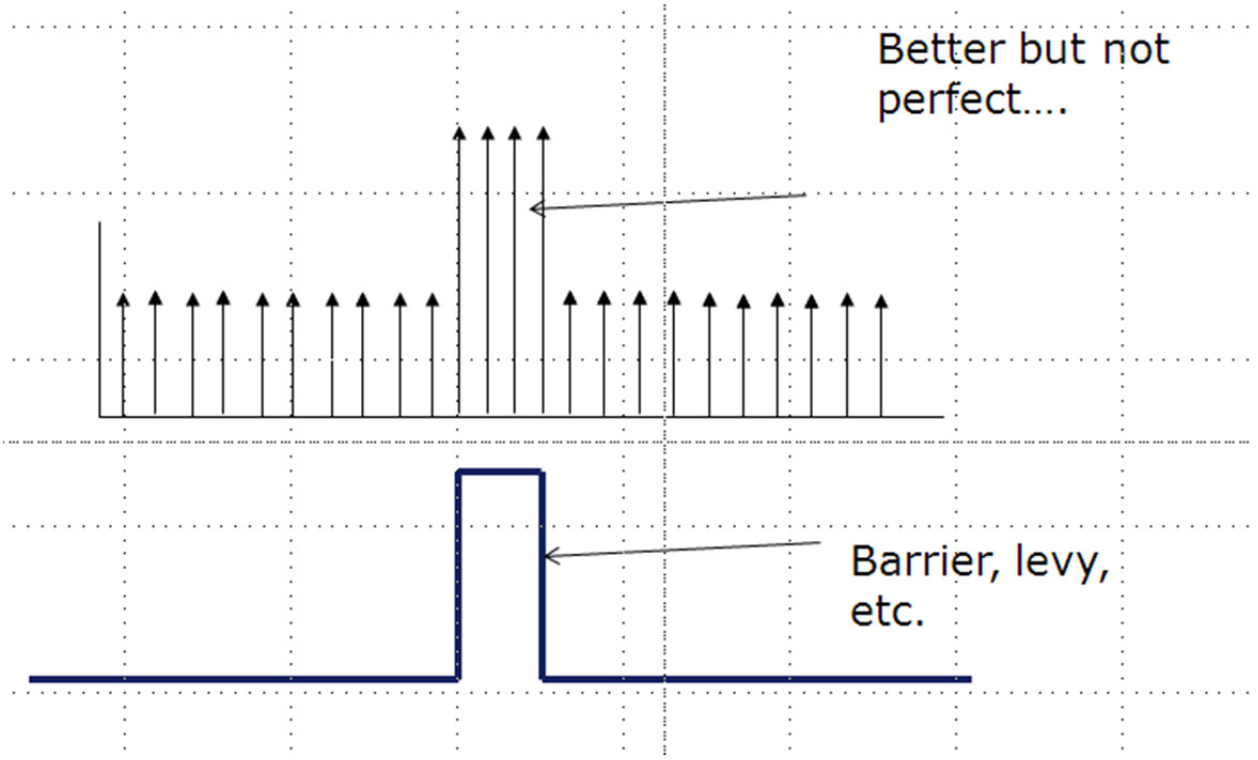

Figure 12: Increasing the sampling rate

It is very interesting to consider the original sampling that occurs when sampling a function for which we cannot directly apply a filter. For example, if we are sampling an analog music signal for the construction of a CD, we can simply insert a band-limiting analog filter prior to the sampling operation (which is essentially what I illustrated in Figure 9). However, what can I do if I am sampling a surface with a LIDAR? Low pass filtering is an averaging process. If our sensor contains "smear" at the point, then I am effectively averaging data into a lower frequency representation. An example of this for a LIDAR sensor is depicted in Figure 13.

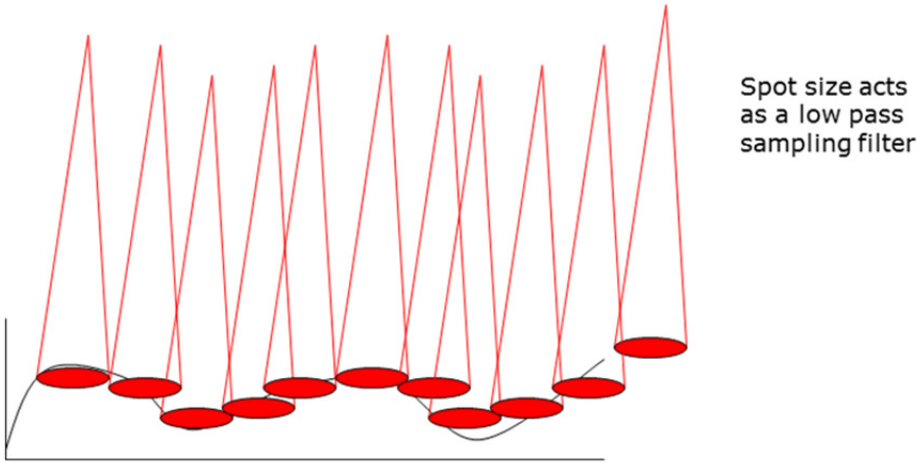

Figure 13: Low Pass Filtering via large "window" samples

INTERPOLATION

Now at the risk of belaboring the point, examine equation (17) with some care. This is the equation that gives us a recipe for interpolating a point from an elevation grid (or image or any other correctly sampled 2D signal). It tells us two important facts:

1. If the underlying data were sampled according to the Nyquist sampling criteria, we can interpolate any point between our samples (e.g. between grid posts in the DEM) *exactly.* This equation does not mean *closely,* it really means *exactly*!
2. Well, that was the good news. The bad news is that we will have to use every point in the grid to accomplish the interpolation! Thus if we have a grid of 1,000 x 1,000 posts, we will have to sum over all 1 million posts to do the interpolation of a single point!

Obviously approximations must be made if the interpolation equation is to be practical in the real world. The real value in this paper of examining this equation is to point out the tradeoff in resampling kernel size. The fewer the number of terms, the less accurate the interpolation.

Let us point out the severe error that can occur with small kernel interpolation. In Figure 14, I have created a synthetic surface (the red plot) by summing two sine functions, one with frequency 1 Hz and the second with frequency 2 Hz. The Nyquist sampling criteria tells us we must sample this signal at least 4 times per second (twice the highest frequency). The red plot is our synthetic function. I sampled this function at each black diamond. I then reconstructed the function (the dashed brown line) using the full interpolation function, (15). You can now see the impact of bilinear interpolation (the blue line). Note the gross errors that occur at the peaks and valleys. Of course, even worse would be nearest neighbor interpolation which would approximate the function as the sample points.

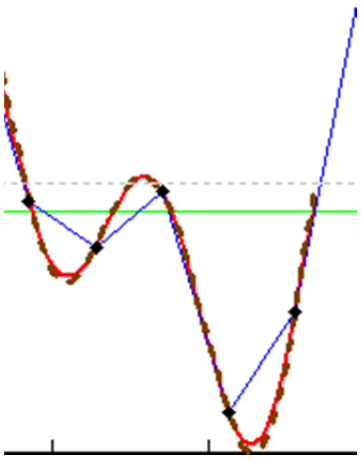

Figure 14: Original Function (red), Samples (black diamond), Full Reconstruction (brown) Linear Interpolation (blue)

The general process of creating (interpolating) a DEM from a TIN is illustrated in Figure 15. You will note that the interpolation is not quite as robust as a two dimensional interpolation! Thus as we approach the Nyquist limit in creating the point cloud, we will significantly degrade the data when interpolating the TIN.

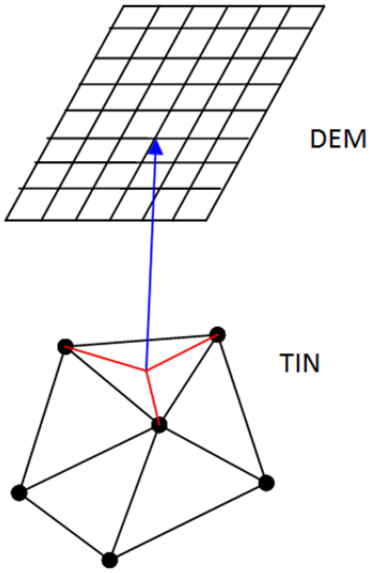

Figure 15: Interpolating a DEM

PHASE SHIFTING

Phase shifting is a term from optics and electrical engineering that means to shift a signal in time (for a time dependent signal) or space (for a spatial signal such as an elevation model) without changing the nature of the signal. Thus a phase shifted signal, f(t) would look like:

$$f(t + T_0) \tag{19}$$

where T_0 is the "phase shift." Phase shifting is a critical operation when one attempts to align geospatial data. Even if the data are on the same sample spacing, they still must be shifted to fit the output data scheme.

A subtle area where phase shifting arises is in the transformation of coordinate systems. For example, if we resample an elevation matrix from Universal Transverse Mercator (UTM) to geographic, we will move the post locations, even if we do not change the ground sample distance (GSD) of the posts.

The two dimensional reconstruction equation (17) tells us how we must go about this. We have to interpolate each point, using our entire ensemble of sampled data!

Merging Elevation Data - Clouds

So, I finally reach the subject of this paper! How should we merge elevation data, particularly when those data come from different resolution and accuracy sensors? I will now disappoint you by saying "I don't know!" I am, however, convinced that most methods in practice today are not correct.

First, let's consider the merge of point cloud data. The easiest way to accomplish this is to push the interpolation problem down the road! This means, given appropriate metadata per point, we can simply merge the data without changing any of the X, Y, Z locations. We leave it up to the interpolating algorithm to then extract the desired data (interpolate to a grid, interpolate an arbitrary point and so forth). This is illustrated in Figure 16. Of course while this is completely technically correct, it is really only saying we are storing the points in a segregated fashion. We have gained nothing from the merge until we try to extract a value.

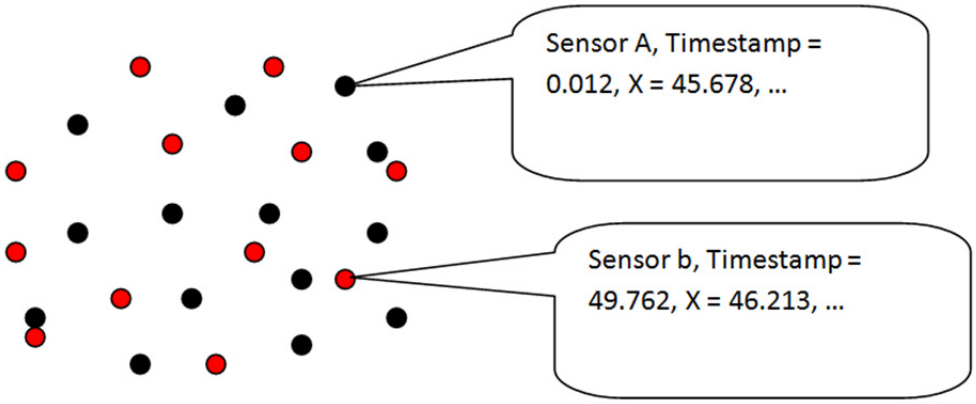

Figure 16: Merging by simply combining

As seen in the previous sections of the paper, we still have not arrived at a satisfactory method for interpolating information from a TIN. However, if the data are significantly oversampled, a simple TIN facet interpolation will introduce minimal error. This would lead us to simply interpolating from the nearest points (Figure 17).

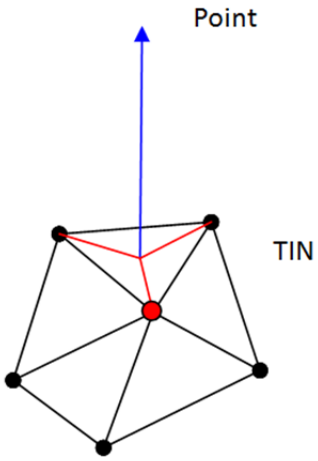

Figure 17: Interpolating from the nearest points

Suppose now, though, that we have accuracy information on a per point basis (that is, we know the variance in X, Y and Z for each point). We would have these data via an error propagation model for the collection system. For example, we might have estimates of the point variance based on the variance estimates from the Position and Orientation System (POS) of the collection platform.

If we consider one dimension only and assume that the error is normally distributed (a big assumption that is probably not true!) then we multiply the normal distributions to obtain the composite statistics (see, for example, Stroupe, et. al.). This gives us merged values for the mean and variance of the merged distributions (called 1 and 2):

$$\sigma_m = \sqrt{\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2+\sigma_2^2}} \tag{20}$$

and

$$\mu_m = \frac{\mu_1\sigma_2^2 + \mu_2\sigma_1^2}{\sigma_1^2+\sigma_2^2} \tag{21}$$

So now we separately interpolate from the two distinct TIN surfaces and then use (21) to compute a new, weighted average Z_m (see Figure 18).

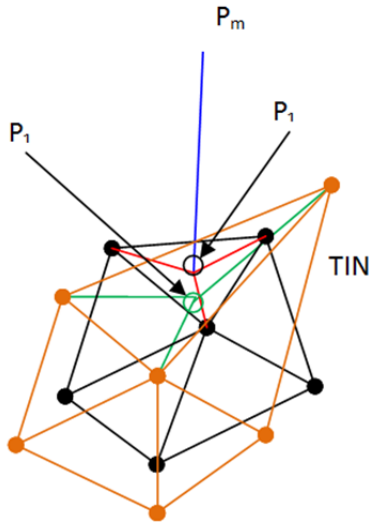

Figure 18: "Statistical" merge

MERGING ELEVATION DATA – GRIDS

Merging grids of elevation data is a bit easier to formulate, given the sampling theorem. The process, if we ignore error per point, proceeds as:

1. Define a common merge grid (G_m). This will be the destination grid. It can be identical to one of the source grids or it can change the resolution and/or "phase" of the sources.
2. Resample each source into separate merge grids. Note that if you are "up sampling" (meaning the resolution of G_m is the same as or higher than the source grid), simply use the

interpolation equation (17)[8] to populate G_m. If the resolution of G_m is lower than that of the source grid, you are "down-sampling." In this case, you must low pass filter the source data before sampling (remember the Nyquist criteria!). This low pass filter would be the same as used in generated reduced resolution data sets for images, for example a Lagrangian.

3. Average the data using the statistical weighting of (21).

CONCLUSIONS

I will readily admit that this paper is a bit of a "starter" document. When I was asked to address this topic, I thought it would be several paragraphs – just stuff the data together! However, as you can see from the development of digital signal processing, there are some critical considerations not only in the merge process but in simply changing the representation of the data.

The big hole in the presentation of this paper is phase shifting a point cloud. I simply have not addressed this issue. From Figure 14 it is obvious that, unless the data have been highly oversampled, a significant error is introduced by simply interpolating from the three closest (in X, Y) points using a triangle facet (TIN interpolation). Nothing new need be invented here – non-uniform sampling is well covered in the electrical engineering literature. I just did not find any references to using these techniques for interpolating elevation data. This is an area that would make quite a nice paper for a graduate thesis!

A useful exercise would be to assemble a block diagram of the entire multi-sensor collection and merge process. Each step of this process would then be modeled in terms of its representation, how to correctly transform to the next block (e.g. the sampling and interpolation requirements) and the error propagation model.

The truly amazing thing to me is how well our data work in spite of a lack of conformance to the precepts of signal processing! Be warned, however. As expectations for higher precision, more accurate data advance, we will have to hone our processes.

[8] In reality, a finite impulse response filter such as a bicubic interpolated will be used.

REFERENCES

C. E. Shannon (1948): “A Mathematical Theory of Communication”, The Bell System Technical Journal, Vol. 27, pp. 379-423, 623-656, July, October.

C. E. Shannon (1949): “Communications in the Presence of Noise”, Proceedings of the IRE, Vol. 37, No. 1, pp. 10-21.

H. Nyquist (1924): “Certain Factors Affecting Telegraph Speed”, Journal of the A.I.E.E, Vol. 43, p. 124.

B. P. Lathi (1998): “Signal Processing and Linear Systems”, Oxford University Press, pp. 319-325.

W. Steeb (2005), “Mathematical Tools in Signal Processing with C++ and Java Simulations”, World Scientific Publishing Company, Chapt 1, (September).

A.W. Stroupe, M.C. Martin, and T.R. Balch (2000), “Merging Gaussian Distributions for Object Localization in Multi-robot Systems”, in Proc. ISER, pp. 343-352.