

Chapter 2

Processing Coastal Lidar Time Series

In this chapter, we analyze time series of lidar data point clouds to assess the point density, gaps in coverage, spatial extent and accuracy. Based on this analysis and a given application, we select an appropriate resolution and interpolation method for computing raster-based digital elevation models (DEM). We explain a per raster-cell average approach and two splines-based approaches for computing DEMs. Finally, we discuss how to assess systematic error using geodetic benchmarks or other ground truth point data and how to correct any shifted DEMs to create a consistent DEM time series.

2.1 General Workflow

Time series of lidar point clouds include data from multiple surveys often acquired for different purpose by various types of lidar technology. To understand the properties of point clouds in the time series, we first analyze the data at a sequence of resolutions and then apply interpolation to compute a DEM at the selected resolution. The methodology, which can be applied to both first return or bare ground data can be summarized as follows:

- Integrate the point-cloud data acquired from various sources within a single coordinate system.
- Perform per-cell statistical analysis of point data at a hierarchical set of resolutions, and use the results to select a common DEM resolution.
- Derive the spatial extent of each survey and a mask for the study area from preliminary low-resolution DEMs computed using the mean elevation value for each cell.

Table 2.1 Characteristics of the lidar surveys based on the available metadata

Agency,* Dates	Lidar equipment	Published point density	Published Accuracy vertical/horizontal (m))
NOAA/NASA/USGS October 19, 1996 September 1 and 26, 1997 September 7, 1998; post-Bonnie† September 9, 1999; post-Dennis† September 18, 1999; post-Floyd† October 6, 1999	Airborne topographic Mapper II	1pt/3m	0.15/2.00
NCDENR/FEMA/NCFMP February 2001	Leica Geosystems aeroscan	1pt/3m	0.20/2.00
NASA/USGS September 18, 2003 pre-Isabel† September 21, 2003 post-Isabel†	EAARL	1pt/3m	0.15/2.00
JALBTCX August 28, 2004 September 28, 2005, post-Ophelia†	Compact hydrographic Airborne rapid total Survey (charts)	1pt/1m	0.3/1.4
NOAA March 27, 2008	IOCM	1pt/1m	0.3/1.4
NASA, USGS December 1, 2009, post-Nor'Ida†	EAARL	1pt/1m	0.2/0.75
NOAA August 8, 2011, post-Irene†		1pt/1m	0.3/1.4

*NASA = National Aeronautics and Space Administration, NOAA = National Oceanic and Atmospheric Administration, USGS = U.S. Geological Survey, NCDENR = North Carolina Department of Environment and Natural Resources, FEMA = Federal Emergency Management Agency, NCFMP = North Carolina Floodplain Mapping Program, JALBTCX = Joint Airborne Lidar Bathymetry Center of Expertise, EAARL = Experimental Advanced Airborne Research Lidar, IOCM = Integrated Ocean and Coastal Mapping

† Hurricane names

- Compute more detailed, smoothed DEMs for the masked study area using spatial interpolation.
- Compare the DEMs with high accuracy ground-based data to remove potential systematic errors and verify the accuracy of each DEM.

The result of this procedure is a consistent series of DEMs which have a common resolution and are clipped to a common spatial extent. To illustrate the workflow we use the provided series of lidar point clouds acquired along the coast of NC since 1996 (Table 2.1). The published horizontal accuracy of this data is 2 m, while the vertical accuracy is 0.12–0.20 m.

2.2 Analysis of Lidar Point Clouds

Data from lidar surveys acquired over the span of several years and for a wide range of applications have varied point densities, spatial extents, and accuracy. We use point per-cell statistics to map the distribution of point densities (as the number of points found in each raster cell) and the range of values in a raster cell. We use this to create a low resolution DEM by computing mean point elevation per cell. This information is helpful when selecting a common resolution for the entire series of DEMs.

To compute the per cell statistics we first set the resolution using `g.region` and then import the lidar points from a lidar text files. In the code below, we use the `r.in.xyz` GRASS command to compute the point count for each raster cell at a resolution 5 m. Then we use the `r.univar` module to calculate mean point count per raster cell.

```
# Purpose: Get mean cell statistics.
# Refer to Ch. 1 Sec. 1.2 for information on where to
# download the data.
# Start grass with location northcarolina_coast_spm
# and mapset NagsHead_series.
grass70

# Set region to Jockey's Ridge area and display
# provided DEM in 2D and 3D
g.region rast=NagsHead_series_1m -p
d.rast NH_2008_1m

# Compute number of lidar points per grid cell at 5m
# resolution for the 1999 survey
g.region res=5
r.in.xyz input=JR_19990909.txt output=JR_stats_n \
    method=n fs=', '
r.null map=JR_stats_n setnull=0
# Get the mean per cell count
r.univar -ge map=JR_stats_n
```

To perform this kind of analysis for a set of resolutions and series of lidar point clouds, we use Python code. In the code below, we compute the point count, elevation range, and mean elevation for each raster cell at the resolutions of 0.5, 2, 5, and 10 m. Then we calculate global statistics on the range raster maps for each resolution.

```
# Purpose: Get mean cell statistics.
import grass.script as grass
files = ['JR_19971002.txt', 'JR_19990909.txt',
        'JR_2001.txt', 'JR_20051126.txt', 'JR_20080327.txt'
        ]
```

```

resolutions=[ 0.5, 2, 5, 10 ]
grass.run_command('g.region',
    region='NagsHead_series_1m')
report = 'date\tres\tn\ttrange\n'
for f in files:
    report += f + '\n'
    for res in resolutions:
        report += '\t' + str(res) + '\t'
        # Set the resolution.
        grass.run_command('g.region', res=res)
        # Get the per cell count.
        grass.run_command('r.in.xyz', input=f,
            output='JR_stats_n', method='n', fs=',',
            overwrite=True)
        grass.run_command('r.null', map='JR_stats_n',
            setnull=0)
        # Get the mean per cell count.
        stats = grass.parse_command('r.univar',
            flags='ge', map='JR_stats_n')
        report += str(stats['mean']) + '\t'
        # Get the per cell range.
        grass.run_command('r.in.xyz', input=f,
            output='JR_stats_range', method='range',
            fs=',', overwrite=True)
        grass.run_command('r.mapcalc',
            expression='JR_stats_range_c=if(isnull(JR_
                stats_n),
            null(), JR_stats_range)', overwrite=True)
        # Get the mean per cell range.
        stats = grass.parse_command('r.univar',
            flags='ge', map='JR_stats_range_c')
        report += str(stats['mean']) + '\n'
grass.run_command('g.region',
    region='NagsHead_series_1m')
print( report )

```

The results of this lidar point cloud analysis at a hierarchy of resolutions for selected surveys are summarized in Table 2.2 and illustrated by Fig. 2.1. At a resolution of 10 m, the mean range of elevations within the raster cells exceeds 1 m for all surveys and 2 m for the last three surveys, indicating that important features may be lost at this resolution. At a resolution of 2 m, the mean range is between 0.08 and 0.65 m and the number of points per raster cell is less than one for older surveys, indicating the need for interpolation. At 0.5 m resolution, the within-cell mean range

Table 2.2 Mean per cell point count and elevation range at 0.5, 2 and 10 m resolution for selected lidar surveys of Nags Head

	Grid size (m)	Points per cell	Range (m)
1997	0.5	1.102	0.020
	2	2.559	0.249
	10	45.522	1.753
1999	0.5	1.113	0.023
	2	3.295	0.315
	10	60.012	1.822
2001	0.5	1.000	0.000
	2	1.006	0.011
	10	7.394	1.358
2005	0.5	1.267	0.034
	2	6.025	0.560
	10	145.361	2.669
2008	0.5	1.030	0.012
	2	3.589	0.444
	10	85.303	2.143

was less than the published data accuracy and interpolation is necessary for all surveys. To preserve the shape of the buildings, we select 0.5 m resolution and the time series of DEMs will be created by interpolation.

2.3 Computing DEMs

To compute a consistent series of DEMs we first derive masks of mapped areas for each survey, then we apply interpolation using the method most appropriate for our application.

2.3.1 Masking Surveyed Areas

Interpolating lidar point data to high resolution DEMs is only meaningful in regions with adequate point coverage. We can mask out low density point regions so that only high density regions are interpolated. Masking is also important because it can substantially reduce the processing time during data analysis. We identify regions to mask by first importing the lidar points at a resolution much greater than the lidar point space (The high resolution value is selected based on the point density analysis in Sec. 2.2). Then we set each cell in the resultant raster to 1 if the cell contains any lidar data points or a ‘no-data’ value if it does not (Fig. 2.2). The following GRASS code sets the resolution to 5 m and uses the `r.in.xyz` and `r.mapcalc` functions to perform these two steps and create a mask based on point density:

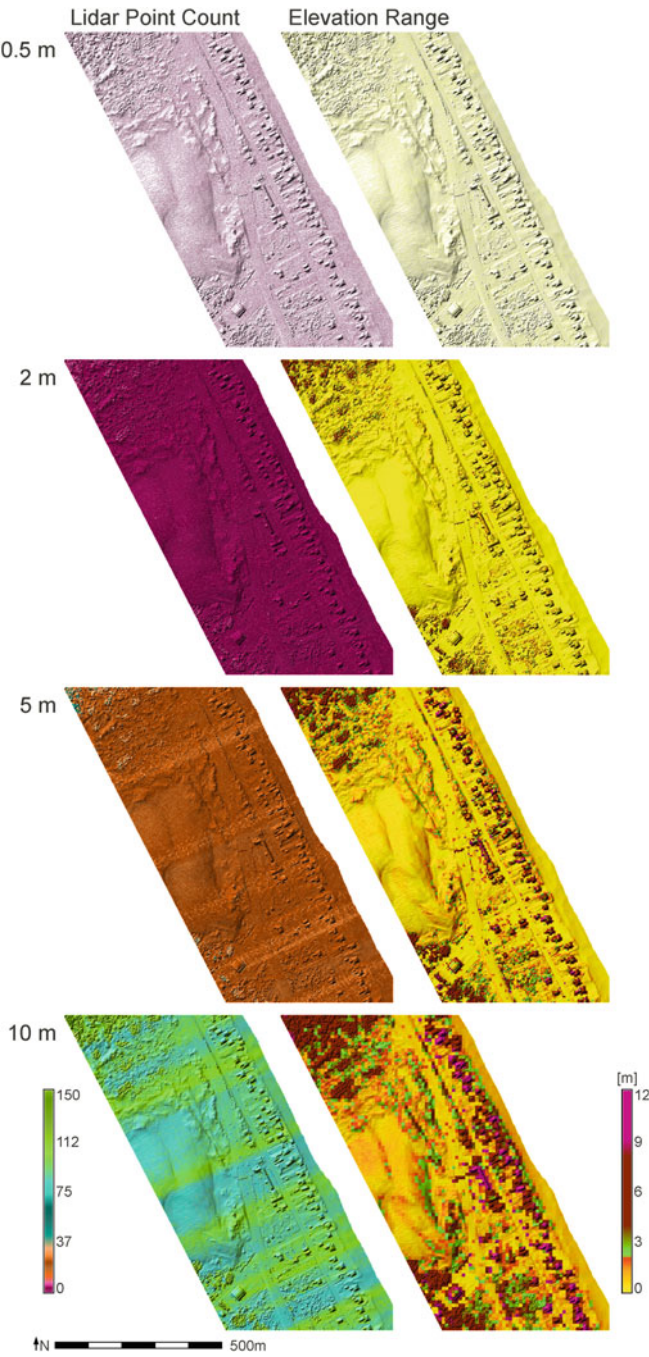


Fig. 2.1 Point density (lidar point count) and elevation range at different resolutions

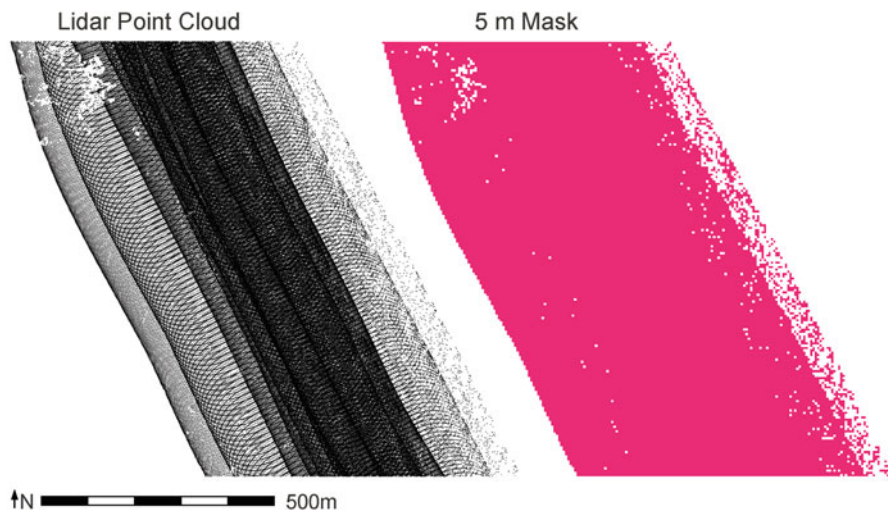


Fig. 2.2 Point cloud and a derived mask based on 1999 lidar

```
# Purpose: Create a masked survey area.
g.region NagsHead_series_1m res=5
r.in.xyz input=JR_1999.txt output=JR_1999_n_5m \
  method=n fs=',' --o
r.mapcalc expression='JR_1999_mask=if( JR_1999_n_5m \
  == 0, null(), 1 )' --o
```

Raster operations (including interpolation) can then be limited to the mask by running the following commands:

```
# Purpose: Limit raster operations with a mask.
g.region res=0.5
r.mask input=JR_1999_mask
```

Raster operations will continue to be limited to the mask area until the mask is removed by running `r.mask` with the `-r` flag:

```
# Purpose: Remove raster mask.
r.mask -r input=JR_1999_mask
```

With the mask set up we can now interpolate the DEMs using a method suitable for the given application.

2.3.2 Binning

When a lidar point cloud is available in an ASCII text format (such as x,y,z tuples) and has at least one point in each raster cell at a fixed resolution, a DEM surface can be generated directly from the lidar points using the `r.in.xyz` module. The

module computes a raster map where the value in each raster cell is a univariate statistic of the lidar data points contained in that cell. For this reason, the method is referred to as binning. The `method` parameter specifies the statistical measure, such as the maximum, minimum, or mean elevation value. We use the analysis of lidar point density outlined in the Sec. 2.2 to select the adequate resolution. For binning, a resolution of 2 m was chosen to ensure that most grid cells contained at least one lidar point.

DEMs are usually computed by setting `method` to `mean` (Fig. 2.3a).

```
# Purpose: Create DEM using raster statistics.
g.region region=NagsHead_series_1m res=2
r.in.xyz input=JR_20080327.txt \
  output=JR_20080327_binmean1m method=mean fs=','
```

2.3.3 Spline Interpolation

Continuous DEMs at resolutions higher than the average point spacing can be computed using spatial interpolation. GRASS7 provides two spline-based modules for bivariate interpolation: `v.surf.rst` and `v.surf.bspline`.

Detailed, smoothed sets of DEMs and topographic parameters (slope, aspect and curvatures) can be computed using the regularized spline with tension (RST) method (Mitasova et al. 2005). RST belongs to interpolation functions that minimize the deviations from the measured points and a smoothness seminorm (Mitas and Mitasova 1999). The RST smoothness seminorm includes derivatives of all orders with their weights decreasing with the increasing derivative order leading to the following function:

$$z(\mathbf{r}) = a_1 + \sum_{j=1}^N \lambda_j R(\varrho_j) \quad (2.1)$$

$$R(\varrho_j) = -[E_1(\varrho_j) + \ln(\varrho_j) + C_E] \quad (2.2)$$

where $z(\mathbf{r})$ is elevation at a point $\mathbf{r} = (x, y)$, a_1 is a trend, λ_j are coefficients, N is the number of given points, $R(\varrho_j)$ is a radial basis function, $\varrho_j = (\varphi r_j/2)^2$, φ is a generalized tension parameter, $r_j = |\mathbf{r} - \mathbf{r}_j|$ is a distance, $C_E = 0.577215$ is the Euler constant, and $E_1(\varrho_j)$ is the exponential integral function (Abramowitz and Stegun 1965; Mitášová and Mitáš 1993). The coefficients a_1 and $\{\lambda_j\}$ are obtained by solving the system of linear equations:

$$\sum_{j=1}^N \lambda_j = 0. \quad (2.3)$$

$$a_1 + \sum_{j=1}^N \lambda_j \left[R(\varrho_j) + \delta \frac{w_0}{w_j} \right] = z(\mathbf{r}_i), \quad i = 1, \dots, N \quad (2.4)$$

where w_0/w_j are positive weighting factors representing a smoothing parameter at each given point $\mathbf{r}_j = (x_j, y_j)$.

The method has both geostatistical and physical interpretation (Mitas and Mitasova 1999). It is formally equivalent to universal kriging with the choice of the covariance function determined by the smoothness seminorm. The intuitive physical interpretation of this method is a thin surface that can be tuned from a rigid plate to a rubber sheet by changing its tension (Fig. 2.3). The tension parameter φ controls the distance over which the given points influence the resulting surface while smoothing controls the vertical deviation of the surface from data points. By using an appropriate combination of tension and smoothing, it is possible to apply the function to various types of surfaces from smoothly changing topography to rough terrain, and select a level of detail represented by a DEM without changing the resolution. The optimal values of parameters can often be found by minimizing the cross validation error (Hofierka et al. 2002; Mitas and Mitasova 1999). The tension and smoothing parameters for each DEM computation can be optimized to reduce the noise and ensure a comparable level of detail in each DEM (see Mitasova et al. (2005), or Neteler and Mitasova (2008) for more details on RST implementation and optimization of its parameters for lidar data).

The RST interpolation for the entire DEM series along with computation of topographic parameters (slope, aspect, profile and tangential curvatures) can be carried out in GRASS by importing the lidar data points using the `v.in.ascii` function and then interpolating the points using the `v.surf.rst` function, as in the following Python script:

```
# Purpose: Import point clouds and interpolate using
# the RST method.
# Refer to Ch. 1 Sec. 1.2 for information on where to
# download the data.
import grass.script as grass

# Find and set region from point cloud.
grass.run_command( 'v.in.ascii',
    input='R_19961016_lidar.txt',
    output='temp', format='point', separator=',',
    skip=0, x=1, y=2, z=3)
grass.run_command( 'g.region', flags='pa',
    vect='temp', res=0.5 )

# Import and interpolate point clouds.
dates = [19961016, 19971002, 19980907, 19990909,
    19990918, 19991104, 2001, 20030916, 20030921,
```

```

    20040925, 20051126, 20080327]
ten = [1200, 1200, 500, 1000, 1000, 1000, 1500, 1000,
       1000, 1500, 2000, 2000]
for i in range( len(dates) ):
    fin = 'R_'+str(dates[i])+'_lidar.txt'
    vect = 'R_'+str(dates[i])
    # Import lidar points that fall within the current
    # region.
    grass.run_command( 'v.in.ascii', flags='tbr',
                       input=fin, output=vect, format='point',
                       fs=',', skip=0, x=1, y=2, z=3)
    rast = 'R_'+str(dates[i])+'_05mrst'
    # Interpolate using RST with scale dependent
    # tension.
    grass.run_command( 'v.surf.rst', flags='tz',
                       input=vect, elev=rast, slope=rast+'_slp',
                       pcurv=rast+'_pcurv', tcurv=rast+'_tcurv',
                       tension=ten[i], smooth=0.5, overwrite=True )

```

The value of the tension parameter is modified for each data set to account for the differences in point densities and level of detail. The script runs the RST interpolation with the -t flag, so that tension is not influenced by the data segmentation and normalization.

Another approach to generating smoothed sets of DEMs is bilinear or bicubic spline interpolation with Tykhonov regularization. In this approach each observation (or lidar data point) is interpreted as a linear combination of spline functions (Brovelli et al. 2004)

$$h_0(t_m) = \sum_{lk} a_{lk} s_{\Delta^g}(t_m - \tau_{lk}) + v_m \quad (2.5)$$

where $h_0(t_m)$ is the elevation of the m th lidar data point, t_m is the planimetric location of the lidar data point, a_{lk} is an unknown fitting parameter, s_{Δ^g} is an interpolation function with compact support (the range of which is described by Δ) and order g (e.g., $g = 1$ describes a bilinear function), τ_{lk} is the planimetric location of the spline interpolating function (which centers on raster cell lk), and v_m is an unobserved disturbance.

Equation (2.5) can be written in matrix form as

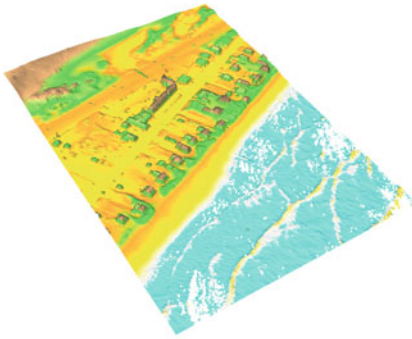
$$\underline{Y}_0 = \underline{A}\underline{a} + \underline{v} \quad (2.6)$$

where

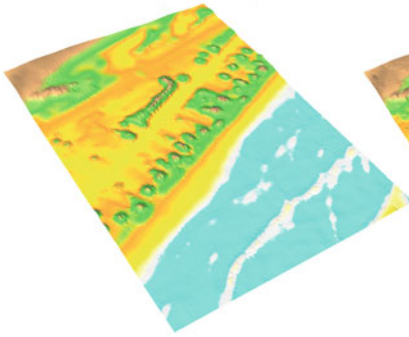
$$\underline{Y}_0 = [\dots h_0(t_m) \dots]^T \quad (2.7)$$

$$\underline{a} = [\dots a_{lk} \dots]^T \quad (2.8)$$

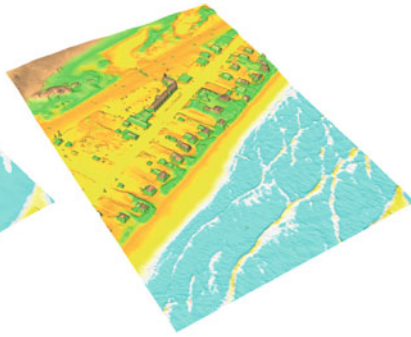
a Binning: resolution = 2 m



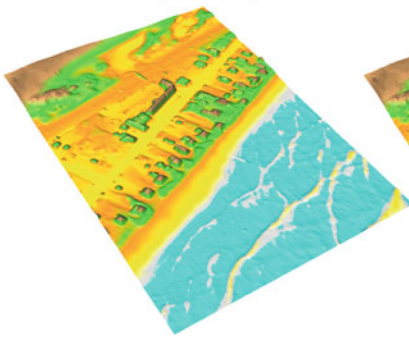
b RST: tension = 40
(default)



c RST: tension = 2000



d Bspline: $\lambda_i = 1$
(default)



e Bspline: $\lambda_i = 0.01$

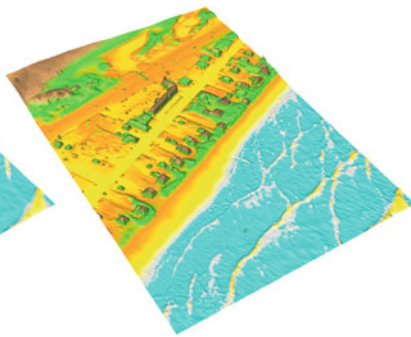


Fig. 2.3 DEM computed by (a) binning (b) `v.surf.rst` with low tension (c) `v.surf.rst` with high tension (d) `v.surf.bspline` with large Tykhonov regularization (e) `v.surf.bspline` with small Tykhonov regularization

and

$$A = \begin{bmatrix} \dots & \dots & \dots \\ \dots & \sum_{lk} a_{lk} s_{\Delta^g}(\underline{t}_m - \tau_{lk}) & \dots \\ \dots & \dots & \dots \end{bmatrix} \quad (2.9)$$

The estimated set of parameters, $\hat{\underline{a}}$ is obtained by minimizing the equation

$$\min \Psi(\underline{a}) = \min\{|\underline{Y}_0 - \hat{\underline{y}}|^2 + \lambda K(\underline{a})\} = \Psi(\hat{\underline{a}}) \quad (2.10)$$

where $|\underline{Y}_0 - \hat{\underline{y}}|^2$ is the least squares minimizing functional and $\lambda K(\underline{a})$ is a regularizing factor that avoids singularities in areas with no data. Regularization is done by minimizing the slope or curvature of the interpolating function. If λ is chosen to be small, the normal matrix is poorly conditioned in areas with little or no data. If λ is chosen to be large, a smoother surface is obtained.

Spline interpolation with Tykhonov regularization is achieved in GRASS using `v.surf.bspline`. The compact support of the weighting function (i.e., Δ) is controlled by the spline step (`sie` and `sin` in the EW and NS directions). Adequate values for `sie` and `sin` are likely to be close to twice the mean point spacing, which can be found by running `v.surf.bspline` with the `-e` flag. The degree of smoothing is controlled by the Tykhonov smoothing parameter `lambda_i`. Larger values of `lambda_i` result in a smoother map, and the optimal value can be determined with a leave-one-out cross validation procedure with the `-c` flag.

```
# Purpose: Import point clouds and interpolate using
# the bspline method.
# Refer to Ch. 1 Sec. 1.2 for information on where to
# download the data.
g.region NagsHead_series_1m
v.in.ascii -ztbr input=JR_20080327.txt \
    output=JR_20080327 fs=',' x=1 y=2 z=3
v.surf.bspline -ze input=JR_20080327 raster=temp
[...]
Estimated point density: 0.8537
Estimated mean distance between points: 1.082
[...]
# Choose sin and sie to be twice mean distance
# between points.
v.surf.bspline -z input=JR_20080327 \
    raster=NH_2008_1mbspl_lam1 sin=2 sie=2
r.colors map=NH_2008_1mbspl_lam1 \
    rules=color_elev_coast.txt
# Find an optimal lambda_i.
# Reduce region for computational efficiency
```

```

g.region n=s+40 e=w+40
v.surf.bspline -c input=JR_20080327 \
    raster_output=temp sin=2 sie=2
g.region n=250670 s=249730 w=913366 e=914342 res=1
v.surf.bspline - input=JR_20080327 \
    raster_output=JR_2008_1mbspl_lam001 sin=2 sie=2 \
    lambda_i=0.01 --o
r.colors map=JR_2008_1mbspl_lam001 \
    rast=NH_2008_1mbspl_lam1

```

To interpolate the entire series, use the Python code above for the RST method but replace `v.surf.rst` with the command `v.surf.bspline`. DEMs resulting from large and small values of `lamda_i` are shown in Figs. 2.3 d and e.

2.4 Eliminating Water Surface Features

For many applications, such as volume calculations or shoreline extraction, elevation data representing water surface features should be set to ‘no-data’ values. After the DEMs are generated this can be achieved by setting elevations that are lower than the mean high water (MHW) elevation to ‘no-data’ values. Any remaining data regions that have a smaller area than the largest one are presumed to represent wave crests and other spurious data, so these are also set to the ‘no-data’ value (Fig. 2.4).

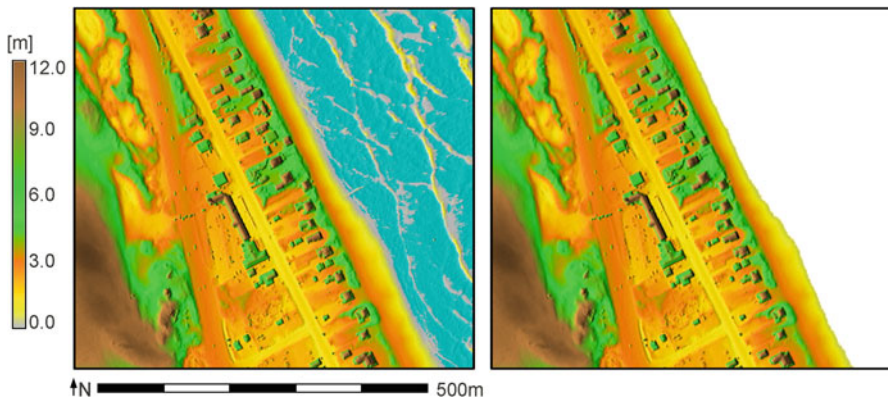


Fig. 2.4 Elimination of water surface features

```

# Purpose: Eliminate water surface features.
r.mapcalc \
expression='JR_20080327_05mbspl_ext_gt036=if
    (JR_20080327_05mbspl_ext>0.36,\

```

```

1, null())' --o
r.to.vect input=JR_20080327_05mbspl_ext_gt036 \
    output=JR_20080327_05mbspl_ext_gt036 type=area --o

# Find the unique, database-generated category of the
# largest area.
# In this case, the category is 1.
v.report -s map=JR_20080327_05mbspl_ext_gt036 \
    option=area

v.extract input=JR_20080327_05mbspl_ext_gt036 \
    output=JR_20080327_05mbspl_ext_mask cats=1 --o
v.to.rast input=JR_20080327_05mbspl_ext_mask \
    output=JR_20080327_05mbspl_ext_mask use=val \
    value=1 --o
r.mapcalc\
    expression='JR_20080327_05mbspl_ext_masked=
    JR_20080327_05mbspl_ext\
    * float(JR_20080327_05mbspl_ext_mask)' --o

g.remove rast="JR_20080327_05mbspl_ext_gt036,
    JR_20080327_05mbspl_ext_mask"
g.remove vect="JR_20080327_05mbspl_ext_gt036,
    JR_20080327_05mbspl_ext_mask"

```

2.5 Correcting Systematic Errors

Due to the registration errors, lidar data can be shifted and this shift needs to be identified and corrected if the data are used for assessment of topographic change.

Systematic errors can be identified by comparing the interpolated DEMs along stable features and geodetic benchmarks in open areas (Fig 2.5). Our sample data set was corrected using the centerline of highway NC-12 because this road was not modified during the study time period and thus had a time-invariant elevation (unlike the erodible terrain surface). If no high-accuracy altimetric data along the centerline is available and if the metrics that are to be derived from the DEM time-series are not datum dependent (e.g., change measurements or rates of change), then the DEMs can simply be referenced to each other using the stable features, such as roads. Alternatively, if high-accuracy altimetric data is available, then for each lidar dataset, elevation differences between the high-accuracy data and lidar can be computed. The median difference quantifies the systematic error. Although mean and median errors are often comparable, the median is chosen for its lower sensitivity to outliers. In the relatively flat coastal terrain, systematic error can be assumed to be spatially constant and can be corrected by shifting the lidar-based DEMs so that the median difference becomes zero. Although the median error is

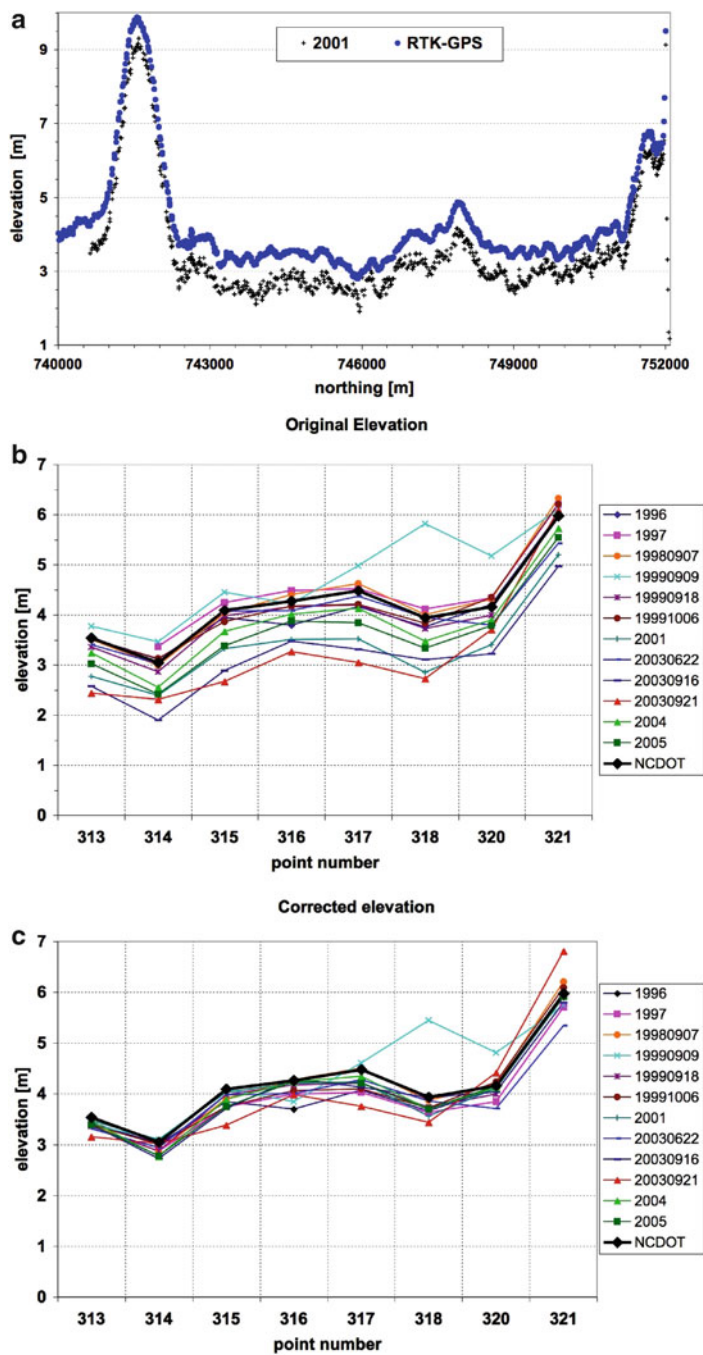


Fig. 2.5 (a) RTKGPS versus lidar profile along the road centerline. Elevation along the centerline of highway NC 12 from (b) uncorrected lidar and (c) lidar with corrected systematic error (Mitasova et al. 2009)

used because of its resistance to outliers, care should still be given to ensure that spurious features captured in the lidar (e.g., cars and overwash deposits) are not used to correct systematic error.

High-accuracy altimetric data along the centerline of highway NC-12 are available as high-resolution road lidar point clouds and as geodetic benchmarks measured by the NCDOT. In the following examples, `DARE_BE94zm3_01m_rstdm.txt` contains road lidar, whereas `road_centerline.txt` contains data points digitized from a DEM along the centerline of NC-12. Systematic error can be corrected using the NCDOT benchmarks by importing them as raster cells, computing the error using `r.mapcalc`, and finally finding the median error by running `r.univar` with the extended statistics flag `-e`:

```
# Purpose: Correct systematic error using road
# centerline using raster approach.
# Refer to Ch. 1 Sec. 1.2 for information on where to
# download the data.
import grass.script as grass
grass.run_command( 'r.in.xyz',
    input='road_centerline.txt',
    output='road_centerline', fs=',', x=1, y=2, z=3 )
grass.run_command( 'r.mapcalc',
    expression='temp_NH_2008_1m_error=road_centerline -
    NH_2008_1m' )
# load statistics into a python dictionary with
# parse_command
stats = grass.parse_command( 'r.univar', flags='ge',
    map='temp_NH_2008_1m_error' )
correction = stats['median']
grass.run_command( 'r.mapcalc',
    expression='NH_2008_1m_corrected=NH_2008_1m + ' +
    correction )
```

If the region is large, a vector approach may be more efficient. The benchmarks can be imported as vector points, and a data table can be populated with DEM elevations at benchmark locations using `v.what.rast`. After updating the database tables, the individual errors can be calculated and the median error can be found by running `v.db.univar` with the extended statistics `-e`:

```
# Purpose: Correct systematic error using road
# centerline using vector approach.
import grass.script as grass
grass.run_command( 'v.in.ascii',
    input='road_centerline.txt',
    output='road_centerline', fs=',', x=1, y=2,
    overwrite=True )
```

```

grass.run_command( 'v.db.renamecolumn',
    map='road_centerline', column='dbl_1,x' )
grass.run_command( 'v.db.renamecolumn',
    map='road_centerline', column='dbl_2,y' )
grass.run_command( 'v.db.renamecolumn',
    map='road_centerline', column='dbl_3,z' )

grass.run_command( 'v.db.addcolumn',
    map='road_centerline', layer=1, columns='elev
    DOUBLE PRECISION, error DOUBLE PRECISION' )
grass.run_command( 'v.what.rast',
    map='road_centerline', layer=1,
    raster='NH_2008_1m', column='elev' )
grass.run_command( 'v.db.update',
    map='road_centerline', col='error', qcol='z-elev' )
grass.run_command( 'v.db.select',
    map='road_centerline' )
stats = grass.parse_command( 'v.db.univar',
    flags='ge', table='road_centerline', column='error'
    )
correction = stats['median']
grass.run_command( 'r.mapcalc',
    expression='NH_2008_1m_corrected=NH_2008_1m + ' +
    correction )

```

The approach for correcting systematic error using road lidar data is analogous to using geodetic benchmarks, with the additional step of a centerline extraction. This can be achieved using a least cost path approach where the cost is a function of distance to the sides of the road.

```

import grass.script as grass
# Purpose: Correct systematic error using road surface
# lidar.
# Refer to Ch. 1 Sec. 1.2 for information on where to
# download the data.
grass.run_command( 'r.in.ascii',
    input='DARE_BE94zm3_01m_rstdm.txt',
    output='DARE_BE94zm3_01m_rstdm' )
grass.run_command( 'g.region', flags='pg',
    rast='DARE_BE94zm3_01m_rstdm' )
region = grass.region()
res = region['nsres']
# Generate start raster (side of road).
grass.run_command( 'r.mapcalc',
    expression='DARE_BE94zm3_01m_rstdm_inv=if(isnull
    (DARE_BE94zm3_01m_rstdm),

```

```

1, null())' )
grass.run_command( 'r.buffer',
    input='DARE_BE94zm3_01m_rstdm_inv',
    output='start_rast', distance=res )
grass.run_command( 'g.remove',
    rast='DARE_BE94zm3_01m_rstdm_inv' )
# Generate a map equal to resolution on the road
# and a map equal to distance from the side of road.
grass.run_command( 'r.mapcalc',
    expression='temp=if(isnull(DARE_BE94zm3_01m_rstdm),
    null(), ' + res + ' )' )
grass.run_command( 'r.cost', flags='k', input='temp',
    output='cost', start_rast='start_rast' )
# Extract centerline connecting two points
# that were digitized at opposite ends of the road.
pt1='913795,250598'
pt2='913992,250202'
grass.run_command( 'r.mapcalc',
    expression='cost=exp(-5*cost)', overwrite=True )
grass.run_command( 'r.cost', flags='k', input='cost',
    output='ccost', start_coordinates=pt1,
    stop_coordinates=pt2, overwrite=True )
grass.run_command( 'r.drain', flags='n',
    input='ccost', output='NC12_centerline',
    voutput='NC12_centerline', coordinate=pt2 )

```

Once the centerline is extracted, the error correction can be computed using `r.mapcalc` and `r.univar` as before:

```

# Purpose: Correct systematic error using
# lidar-extracted centerline.
import grass.script as grass
grass.run_command( 'r.mapcalc',
    expression='NH_2008_1m_error=if(isnull(NC12_
    centerline), null(),
    DARE_BE94zm3_01m_rstdm-NH_2008_1m)' )
stats = grass.parse_command( 'r.univar', flags='ge',
    map='NH_2008_1m_error' )
correction = stats['median']
grass.run_command( 'r.mapcalc',
    expression='NH_2008_1m_corrected=NH_2008_1m + ' +
    correction )

```

References

- Abramowitz, M. and Stegun, I. (1965). *Handbook of mathematical functions: with formulas, graphs, and mathematical tables*, volume 55. Dover publications.
- Brovelli, M. A., Cannata, M., and Longoni, U. M. (2004). LIDAR data filtering and DTM interpolation within GRASS. *Transactions in GIS*, 8(2):155–174.
- Hofierka, J., Parajka, J., Mitasova, H., and Mitas, L. (2002). Multivariate interpolation of precipitation using regularized spline with tension. *Transactions in GIS*, 6(2):135–150.
- Mitas, L. and Mitasova, H. (1999). Spatial interpolation. *Geographical Information Systems: Principles, Techniques, Management and Applications*, Wiley, 481.
- Mitášová, H. and Mitáš, L. (1993). Interpolation by regularized spline with tension: I. Theory and implementation. *Mathematical geology*, 25(6):641–655.
- Mitasova, H., Mitas, L., and Harmon, R. (2005). Simultaneous spline approximation and topographic analysis for lidar elevation data in open-source GIS. *IEEE Geoscience and Remote Sensing Letters*, 2:375–379. DOI: 10.1109/LGRS.2005.848533.
- Mitasova, H., Overton, M., Recalde, J., Bernstein, D., and Freeman, C. (2009). Raster-based analysis of coastal terrain dynamics from multitemporal lidar data. *Journal of Coastal Research*, 25:207–215. DOI: [10.2112/07-0976.1](https://doi.org/10.2112/07-0976.1).
- Neteler, M. and Mitasova, H. (2008). *Open source GIS: a GRASS GIS approach*. New York: Springer, third edition.

GIS-based Analysis of Coastal Lidar Time-Series

Hardin, E.; Mitsova, H.; Tateosian, L.G.; Overton, M.

2014, VI, 84 p. 35 illus., 34 illus. in color., Softcover

ISBN: 978-1-4939-1834-8