APPROXIMATING LIDAR GROUND ELEVATION DATA BY HIGHER-ORDER TENSOR-PRODUCT B-SPLINES

S. Mukherji

Centre of Studies in Resources Engineering, Indian Institute of Technology, Bombay,
P.O.: Powai - I.I.T., Mumbai - 400 076, INDIA –
shyamali@csre.iitb.ac.in

Commission III, WG III/3

KEY WORDS: DEM/DTM, LIDAR, Surface, Modelling, Reconstruction, GIS.

ABSTRACT:

We consider, in this paper, the problem of reconstructing the surface from LIDAR ground elevations. We reconstruct the surface by first rotating the LIDAR scan-lines so that they are parallel to the Y-axis, and then approximating the elevations by higher-order tensor-product B-splines using the least squared-error criterion. The resulting surface is both accurate and smooth..The approximating surface is a linear combination of tensor-product cubic B-splines. So, the coefficients of the tensor-product B-splines define the reconstructed surface. Since tensor-product cubic (quadratic) B-splines are non-zero only for four (three) knot-intervals in the x-direction and y-direction, the elevation at any point can be found in constant time and a grid DEM can be generated from the coefficients of the B-splines in time linear in the size of the grid.

1. INTRODUCTION

In this paper, we reconstruct the surface from LIDAR ground elevations by approximating the elevations by higher-order tensor-product B-splines using the least squared-error criterion.

Mitasova et al. approximated LIDAR elevations using regularized splines with tension (Mitasova, 2005). These splines are infinitely differentiable everywhere. Brovelli et al. approximated LIDAR elevation data by bilinear B-splines which are only C^0 -continuous (Brovelli, 2004).

Higher-order B-splines have continuous higher-order derivatives and, unlike regularized splines, B-splines have finite support whereby the coefficient matrix of the resulting system of linear equations is sparse. We propose using higher-order B-splines for approximating LIDAR elevations as they yield a smooth surface. Tensor-product B-splines are used for the approximation. In this paper, we propose rotating the LIDAR scan-lines so that they are parallel to the Y-axis, as this leads to a more accurate approximation by tensor-product B-splines.

The least squared-error approximation technique is described in Section 2. The results obtained by approximating the LIDAR elevations by higher-order tensor-product B-splines are given in Section 3. In Section 4, we show that the elevation of the reconstructed surface can be computed efficiently and also give an efficient method for computing the DEM. We conclude the paper in Section 5, with a summary of the advantages of this reconstruction.

2. LEAST SQUARED-ERROR APPROXIMATION BY TENSOR-PRODUCT CUBIC B-SPLINES

In this Section, we consider least squared-error approximation by cubic splines.

A cubic B-spline, $B(x)$, centered at xi , and with uniform knot-spacing Δ , is given by $h(|\, x - xi \,| / \Delta)$ where

$$h(p) = \begin{cases} 3p^3/6 - p^2 + 4/6 & 0 \le p < 1 \\ -p^3/6 + p^2 - 2p + 8/6 & 1 \le p < 2 \\ 0 & 2 \le p \end{cases}$$

A tensor-product cubic B-spline, centered at a point (x_1, y_1) , is the product of a cubic B-spline $B(x)$ centered at x_1 and a cubic B-spline $B(y)$ centered at y_1 .

Let $B_i(x)$ and $C_j(y)$ be cubic B-splines along the x and y-directions, respectively. Let us take n_x B-splines along the x-direction and n_y B-splines along the y-direction. Then, the spline surface $S(x, y)$ which is to be constructed is

$$S(x,y) = \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} c_{ij} B_i(x) C_j(y)$$

Let N be the number of points, p_k , at which the elevations, h_k , are known.

A smoothing term is added to the squared-error that is to be minimized (Floater, 2000) and the function to be minimized is

$$F(c) = \sum_{k=1}^{N} \{S(p_k) - h_k\}^2 + \lambda J(c), \qquad (1)$$

where c is the coefficient vector, λ, a positive constant and

$$J(c) = \int_{a_1}^{b_1}\int_{a_2}^{b_2} S_{xx}^2 + 2S_{xy}^2 + S_{yy}^2 dydx$$

where S_{xx}, S_{xy} and S_{yy} are the second-order partial derivatives of $S(x,y)$ and $[a_1,b_1]\times[a_2,b_2]$ is the domain of S.

This integral can be expressed as

$$\sum_{i=1}^{n_x}\sum_{j=1}^{n_y}\sum_{r=1}^{n_x}\sum_{s=1}^{n_y} E_{ijrs}c_{ij}c_{rs} ,$$

where

$$E_{ijrs} = A_{ijrs} + 2B_{ijrs} + C_{ijrs}$$

and

$$A_{ijrs} = \int_{a_1}^{b_1} B_i''(x)B_r''(x)dx \int_{a_2}^{b_2} C_j(y)C_s(y)dy ,$$

$$B_{ijrs} = \int_{a_1}^{b_1} B_i'(x)B_r'(x)dx \int_{a_2}^{b_2} C_j'(y)C_s'(y)dy ,$$

$$C_{ijrs} = \int_{a_1}^{b_1} B_i(x)B_r(x)dx \int_{a_2}^{b_2} C_j''(y)C_s''(y)dy .$$

A minimum of $F(c)$ must occur at a point c where all partial derivatives are zero.

Let

$$J(c) = c^T Ec ,$$

where E is a square matrix of dimension $n = n_x \times n_y$ whose elements are

$$E_{(j-1)n_x+i,(s-1)n_x+r} = E_{ijrs}$$

for $i,r = 1,...,n_x$ and $j,s = 1,...,n_y$.

By differentiating (1), we get

$$(B^T B + \lambda E)c = B^T h , \qquad (2)$$

where $h = (h_1,...,h_N)^T$ and B is the $N \times n$ matrix

$$\begin{pmatrix} B_1(x_1)C_1(y_1) & B_2(x_1)C_1(y_1) & ... & B_{n_x}(x_1)C_{n_y}(y_1) \\ . & . & . & . \\ . & . & . & . \\ B_1(x_N)C_1(y_N) & B_2(x_N)C_1(y_N) & ... & B_{n_x}(x_N)C_{n_y}(y_N) \end{pmatrix}$$

where $p_k = (x_k, y_k)$.

Then, the solution to (1) is the solution c to (2).

The matrix $G = B^T B + \lambda E$ is positive semi-definite. The matrix is strictly positive definite if the only solution to $c^T Gc = 0$ is $c = 0$.

First observe that

$$c^T Ec = J(c) = 0$$

implies that S must be a linear polynomial $a + bx + cy$. Second, observe that

$$c^T B^T Bc = \| Bc \|^2 = 0$$

implies that $S(p_k) = 0$ for all $k = 1,...,N$. Thus, we have that $c^T Gc = 0$ implies that S is a linear polynomial which is *0* at every point p_k. Clearly then, if there are at least 3 points p_k which do not lie on a straight line, S would have to be *0* and all the coefficients c_{ij} would have to be *0*. Since the points p_k can never all be collinear, we deduce that G is indeed non-singular and the minimizer c of (2) is unique.

3. RESULTS

As tensor-product B-splines are used for the approximation, rotating the LIDAR scan-lines so that they are parallel to the Y-axis, results in a better approximation. LIDAR elevation data (post-rotation) for a rural area scanned from an altitude of 1350 metres with a point spacing of 1.1 metres are shown in Fig. 1.

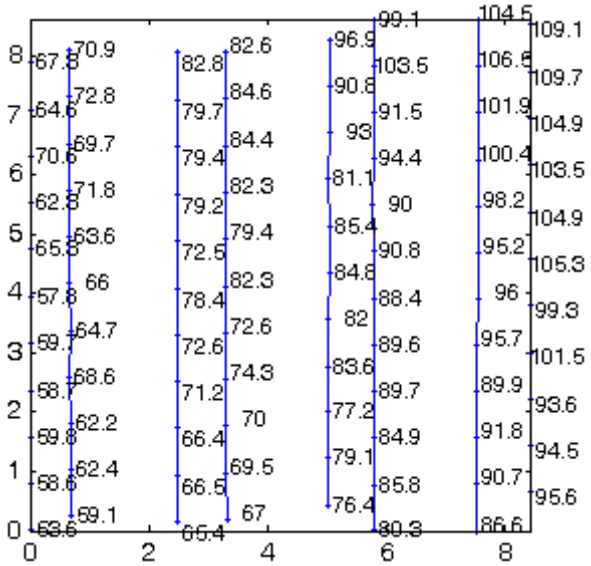

Figure 1: LIDAR elevations: The LIDAR scan-lines have been rotated so that they are parallel to the Y-axis. The first and last scan-lines in the figure are coincident with the Y-axis and the right-end edge of the bounding box of the

figure, respectively. The elevation values shown are in centimetres and are in excess of 115 metres.

The elevation contours of the surfaces obtained by approximating the LIDAR elevation data by bilinear and bi-quadratic splines without rotating the scan-lines and after rotating the scan-lines are shown in Figs. 2-4. The shaded reliefs are shown in Figs. 5-7 and the surfaces (scaled by a factor of 10 in the z-direction) are shown in Figs. 8-10. The approximation by bilinear splines after rotating the scan-lines is noisy, as is evident from the shaded relief. The approximation by bilinear splines, without rotating the scan-lines, has an r.m.s. deviation of 2 cm. from the LIDAR elevations. The surface obtained by approximating the elevations by 7 x 7 tensor-product quadratic B-splines (corresponding to 7 B-splines along the *x*-direction and 7 B-splines along the *y*-direction) has a lower r.m.s. deviation from the LIDAR elevations (1.8 cm.).

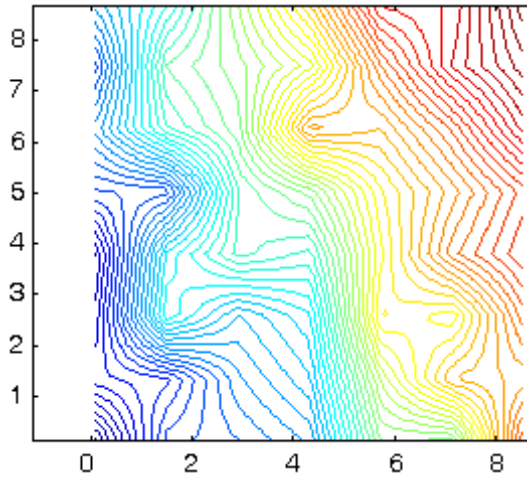

Figure 2: Bilinear spline approximation of the data after rotation.

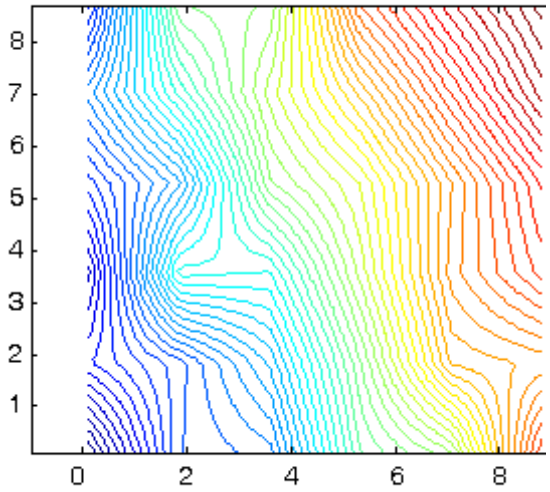

Figure 3: Bilinear spline approximation of the data without rotation.

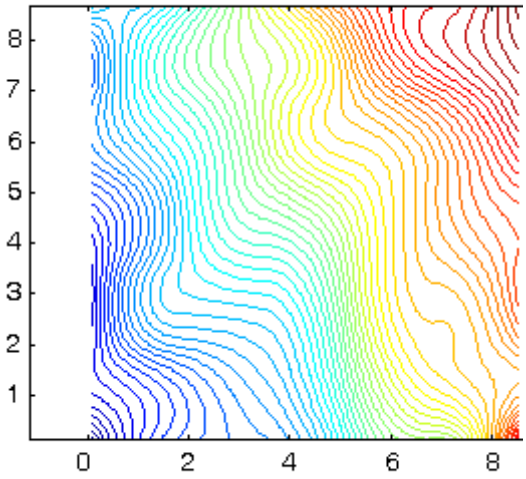

Figure 4: Bi-quadratic spline approximation of the data after rotation.

LIDAR elevation data with gaps for a rural area, again, are shown in Fig. 11. The elevations are approximated well by tensor-product cubic B-splines as is shown in Fig. 12. 4 x 4 tensor-product cubic B-splines and a value of 0.000001 for λ were used for the approximation and the r.m.s. deviation of the surface from the LIDAR elevations is 1.8 cm..

4. COMPUTING THE DEM

In this Section, we show that the elevation of the reconstructed surface at a point can be computed efficiently. We then describe an efficient way of generating the DEM with low memory requirements.

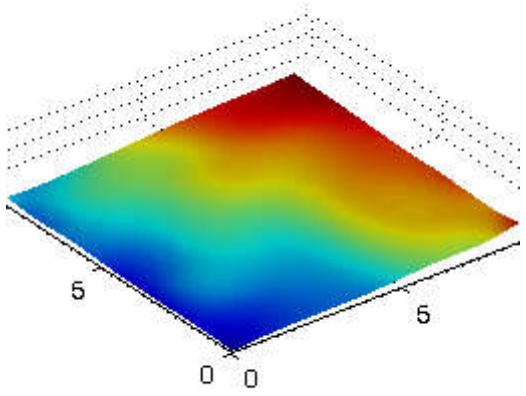

Figure 5: Bi-quadratic spline surface approximation of the data after rotation.

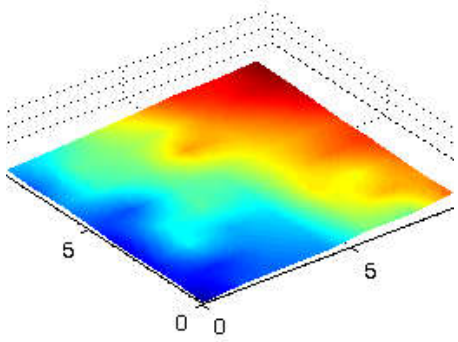

Figure 6: Approximation by bilinear splines after rotation.

We consider, here, the computation for cubic splines. The procedure for quadratic splines is similar.

Since tensor-product cubic B-splines are non-zero only for four knot-intervals in the *x*-direction and the *y*-direction, exactly 16 tensor-product cubic B-splines contribute to the elevation at a point. In contrast, when regularized splines with tension are used for the approximation, though points which are sufficiently distant from a point, do not significantly contribute to the elevation at the point, at least 100 points contribute significantly to the elevation at a point (Mitasova, 2005).

Let us assume that the B-splines that are non-zero at a point $(x1, y1)$ are $B_k(x)$, $B_{k+1}(x)$, $B_{k+2}(x)$, $B_{k+3}(x)$, $C_l(y)$, $C_{l+1}(y)$, $C_{l+2}(y)$ and $C_{l+3}(y)$. The elevation at $(x1, y1)$ is found by evaluating the right-hand side of Eqn. 2 at $(x1, y1)$ as follows:

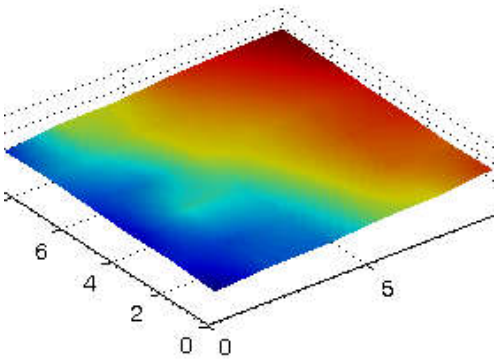

Figure 7: Approximation by bilinear splines without rotation.

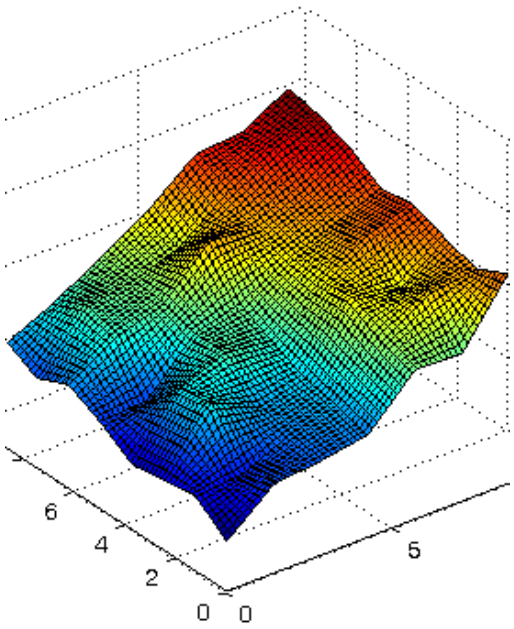

Figure 8: Approximation by bilinear splines after rotation.

We first evaluate the cubic polynomials $P_j(x) = \sum_{i=k}^{k+3} c_{ij} B_i(x)$, $j = l,...,l+3$ at $x = x1$. To do this, we first find $p = | x1 - x_{k+1} | / \Delta_x$ where x_{k+1} is the center of the B-spline $B_{k+1}(x)$ and Δ_x is the knot-interval for the $B_i(x)$. $P_j(x1)$ is, then,

$$c_{kj}(-p^3 + 3p^2 - 15p + 1) + c_{k+1,j}(3p^3 - 6p^2 + 4) + c_{k+2,j}(-3p^3 + 3p^2 + 3p + 1) + c_{k+3,j}p^3 \quad (3)$$

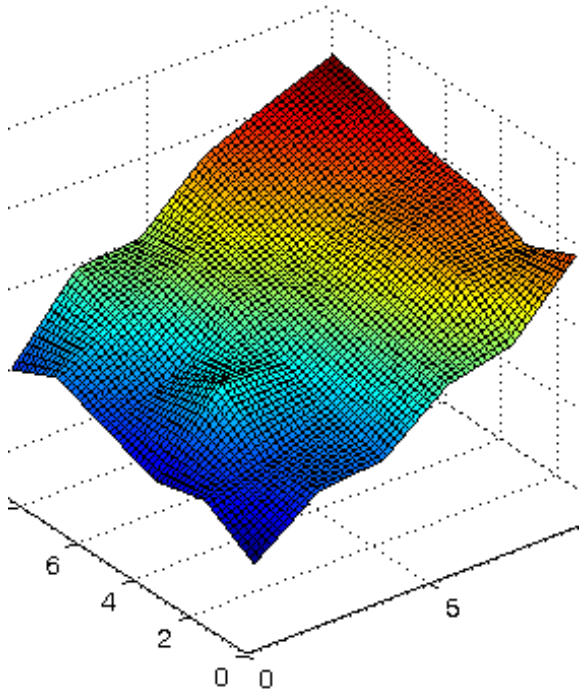

Figure 9: Approximation by bilinear splines without rotation.

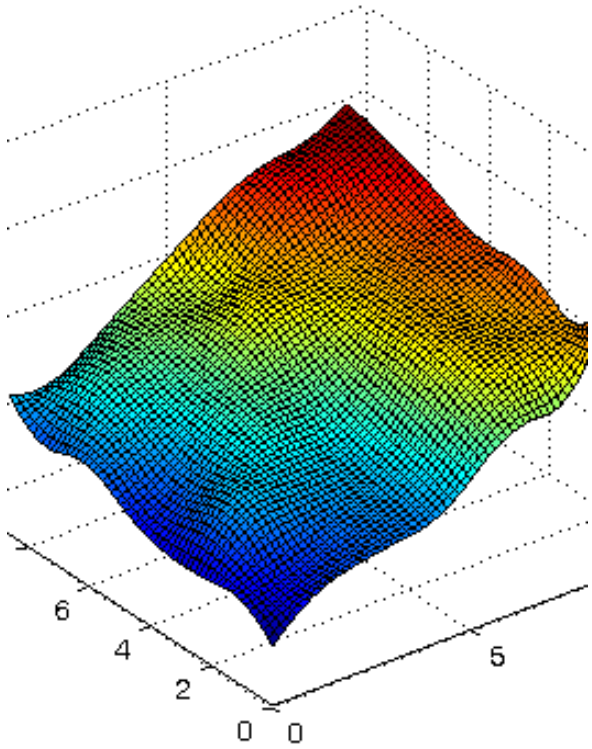

Figure 10: Approximation by bi-quadratic splines after rotation.

The quantities in the brackets require 17 operations {where an operation is either an addition (subtraction) or a multiplication}. So, finding the four $P_j(x1)$ requires 17 + 4 x 7 = 45 operations.

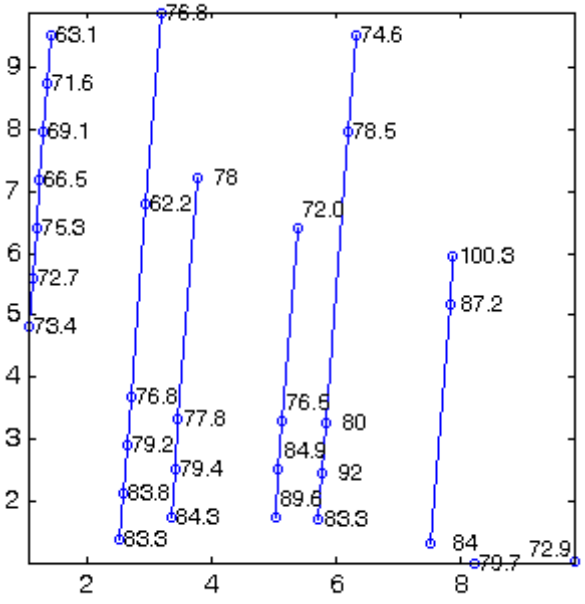

Figure 11: LIDAR elevations with gaps. (The elevation values are in centimetres and are in excess of 114 metres.)

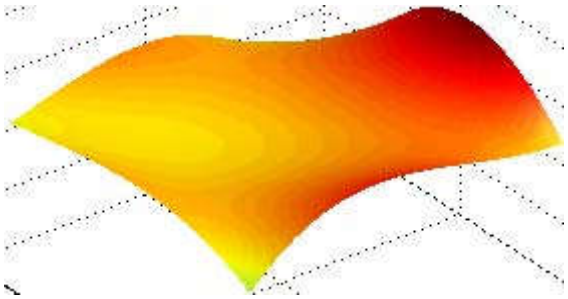

Figure 12: Surface approximating LIDAR data with gaps.

Then, we find $q = | y1 - y_{l+1} | / \Delta_y$ where y_{l+1} is the center of the B-spline $C_{l+1}(y)$ and Δ_y is the knot-interval for the $C_j(y)$. The elevation at $(x1, y1)$, which is $\sum_{j=l}^{l+3} P_j(x1) C_j(y1)$, can be expressed in a form that is analogous to the expression in (3) above. A rearrangement of the terms yields

$$q^3\{-P_l(x1) + 3(P_{l+1}(x1) - P_{l+2}(x1)) + P_{l+3}(x1)\}$$
$$+ 3q^2\{P_l(x1) + P_{l+2}(x1) - 2P_{l+1}(x1)\} + 3q\{-5P_l(x1) + P_{l+2}(x1)\}$$
$$+ P_l(x1) + P_{l+2}(x1) + 4P_{l+1}(x1) \qquad (4)$$

the computation of which requires 21 operations.

Thus, the elevation at any point can be found with 45 + 21 = 66 operations, or, in constant time.

When generating a DEM, we start with the distinct x co-ordinates, of the points of the DEM grid (Fig. 13), which lie

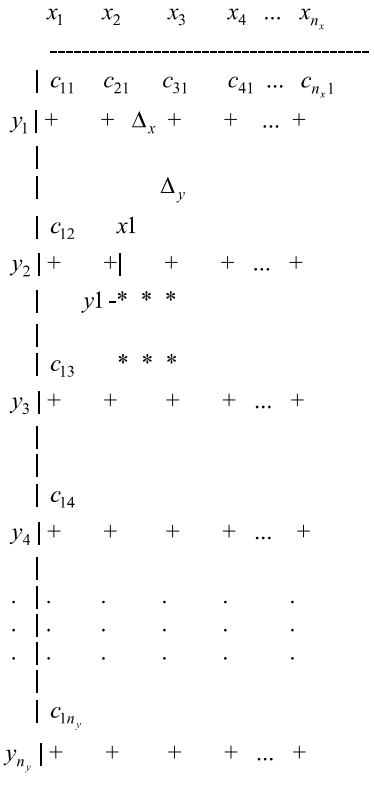

Figure 13: Computing the DEM

between x_2 and x_3 (x_1 and x_{n_x} lie at least Δ_x to the left of and Δ_x to the right of the first and last data points, respectively, in the horizontal direction; and there are no data points in the intervals $[x_1, x_2]$ and $[x_{n_x-1}, x_{n_x}]$), compute the respective $p = | x - x_2 | / \Delta_x$ and compute and store the quantities in the brackets in expression (3), for these x co-ordinates. This takes 17 N_x operations where N_x is the number of distinct x co-ordinates of the grid that lie between x_2 and x_3. We then compute and store the four $P_j(x1)$, $j = 1,...,4$ for each of these x co-ordinates. This takes N_x x 4 x 7 = 28 N_x operations.

We then take the y co-ordinates of the DEM grid that lie between y_2 and y_3 and find the corresponding $q = | y - y_2 | / \Delta_y$. The elevation at the grid-points, $(x1, y1)$, whose x co-ordinates lie between x_2 and x_3 and y co-ordinates lie between y_2 and y_3, is

$$\sum_{j=1}^{4} P_j(x1) C_j(y1) = P_1(x1)(-q^3 + 3q^2 - 15q + 1) +$$

$$P_2(x1)(3q^3 - 6q^2 + 4) + P_3(x1)(-3q^3 + 3q^2 + 3q + 1) + P_4(x1)q^3 \quad .$$

Thus, the computation of the elevations at these grid-points takes 17 N_y + N_y x N_x x 7 operations where N_y is the number of distinct y co-ordinates of the grid that lie between y_2 and y_3.

Next, we compute $P_5(x1)$ and use $P_j(x1)$, $j = 2,...,5$ to find the elevations at the grid-points, whose x co-ordinates lie between x_2 and x_3 and y co- ordinates lie between y_3 and y_4. We proceed down the grid in this manner, computing the elevations at the grid-points till we reach the bottom of the grid. Then we return to the top of the grid and repeat the entire procedure starting, this time, with the distinct x co-ordinates of the grid-points between x_3 and x_4. This process continues till the lower right corner of the grid is reached.

Thus, the computation of the elevations at the grid-points requires $17N$ +7 Nn_y +17 $M(n_x - 3)$ +7 MN = 7 MN +17 $M(n_x - 3)$ + 7 Nn_y + 17N operations for an M x N grid. n_x and n_y can be atmost N +3 and M+3. So, the computation time is linear in the size of the grid and the multiplying constant is small, viz., 31.

5. CONCLUSIONS

We have seen, in this paper, that rotating the LIDAR scan-lines so that they are parallel to the Y-axis results in a better approximation and higher-order tensor-product B-splines lead to a good reconstruction. The smoothness of the surface results from the inherently smooth nature of quadratic (cubic) B-splines, which are C^1 (C^2)-continuous.

Tensor-product quadratic (cubic) B-splines are non-zero only for three (four) knot-intervals in the x-direction and y-direction. Therefore, exactly 9 (16) quadratic (cubic) tensor-product B-splines contribute to the elevation of the surface at a point. So, the elevation at any point can be found in constant time and a grid DEM can be generated from the coefficients of the B-splines in time linear in the size of the grid.

ACKNOWLEDGEMENT

This work was supported by the Indian Space Research Organization.

REFERENCES

Brovelli, M. A., Cannata, M., Longoni, U. M., 2004. LIDAR data filtering and DTM interpolation within GRASS. *Transactions in GIS*, 8(2), pp. 155-174.

Floater, M. S. , 2000. Meshless parameterization and B-spline surface approximation. In: *The Mathematics of Surfaces IX*, Cipolla, R. and Martin, R. (eds.), Springer Verlag, pp. 1-18.

Mitasova, H. , Mitas, L. , Harmon, R. S. , 2005. Simultaneous spline approximation and topographic analysis for lidar elevtion data in open-source GIS. IEEE Geoscience and Remote *Sensing Letters*, 2(4), pp. 375-379.