A new machine learning paradigm for terrain reconstruction

Huang, Guang-Bin; Yeu, Thomas Chee Wee; Lim, Meng-Hiot; Agarwal, Amit; Ong, Yew Soon

2006

Yeu, C. W., Lim, M. H., Huang, G. B., Agarwal, A., & Ong, Y. S. (2006). A new machine learning paradigm for terrain reconstruction. *Geoscience and Remote Sensing Letters*. 3(3), 382-386.

<https://hdl.handle.net/10356/91543>

<https://doi.org/10.1109/LGRS.2006.873687>

© 2006 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder. <http://www.ieee.org/portal/site> This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.

A New Machine Learning Paradigm for Terrain Reconstruction

Chee-Wee Thomas Yeu, Meng-Hiot Lim, Guang-Bin Huang, *Senior Member, IEEE*,
Amit Agarwal, and Yew-Soon Ong, *Member, IEEE*

Abstract—Terrain models that permit multiresolution access are essential for model predictive control of unmanned aerial vehicles in low-level flights. The authors present the extreme learning machine (ELM), a recently proposed learning paradigm, as a mechanism for learning the stored digital elevation information to allow multiresolution access. We give results of simulations designed to compare the performance of our approach with two other approaches for multiresolution access, namely: 1) linear interpolation on Delaunay triangles of the sampled terrain data points and 2) terrain learning using support vector machines (SVMs). The results show that to achieve the same mean square error during access, the memory needed in our approach is significantly lower. Additionally, the offline training time for the ELM network is much less than that for the SVM.

Index Terms—Delaunay triangulation, extreme learning machine, radial basis function (RBF) networks, support vector machine (SVM), terrain mapping.

I. INTRODUCTION

THE PROBLEM of path planning for a field robot comprises (at least) two stages. The first stage involves computing an optimum set of waypoints in the free configuration space [1]. This is generally done assuming ideal sensor and vehicle control characteristics. Additionally, straight line travel or travel along simple nonholonomic paths [2] is assumed between consecutive waypoints. The second stage involves real-time update of the path between consecutive waypoints to ensure obstacle avoidance and close compliance between the actual and the planned path. This stage takes into account the dynamics constraints of the robot, errors due to inaccurate sensing and imperfect control, and response times of the actuators. The control loop (for example, see [3]) in the second stage can be computationally expensive in cases such as those involving low-level flight of an unmanned aerial vehicle (UAV) in urban zones or over high-relief terrain. This, in turn, can put severe restrictions on the speed at which the UAV can fly safely. One way to reduce the computational load is to incorporate terrain (a two-dimensional manifold) information.

High-resolution terrain maps are needed to support UAV navigation. However, due to large amount of memory needed for storing such maps, only low-frequency information is typically stored. Queries for elevation information of arbitrary points in the terrain are supported with the aid of a local (e.g.,

an interpolation on vertices of a cell of a mesh) or a global estimator [e.g., a neural network (NN) classifier].

In this letter, we propose the use of extreme learning machine (ELM) for multiresolution access of terrain height information. We compare it with the use of Delaunay triangulation (DT) and support vector machines (SVMs) for regression on the bases of accuracy of queries measured in terms of mean square error (mse), memory requirement, time needed to initialize appropriate data structures, and query response time (Section III). In the remaining part of the letter, we deemphasize discussions on the practical motivation for our study of the multiresolution terrain access problem in favor of a more thorough discussion on the application of the ELM to the problem and comparative simulation results.

II. TECHNIQUES FOR MULTIREOLUTION TERRAIN ACCESS

A. DT

The DT of a set of N planar points is the triangulation $DT(N)$ for which no point in N lies within the interior of circumcircle of any triangle in $DT(N)$. It can be computed in $O(N \log N)$ time and requires $O(N)$ storage. DT-based interpolation is widely used to estimate elevations of points in the terrain that are not in the set of sampled points because it minimizes the roughness of the resulting terrain over all possible data-independent triangulations [4]. To use this data structure for multiresolution access, a set of sample points is selected [5]–[7] from the raw dataset. The accuracy for queries can be greatly affected by the manner in which the sample points are picked. In this work, all sample points were chosen from a uniform grid. Next, the triangles are computed using (x, y) coordinate information of the sample points. Finally, a query for the height of any point (x, y) in the terrain that is not in the sampled set is interpolated using

$$z = ax + by + c. \quad (1)$$

In (1), a , b , and c are the coefficients that parameterize the plane defined by vertices of the smallest triangle in $DT(N)$ that encloses the query point.

B. SVM

SVMs have been extensively used for object classification and regression problems [8], [9]. It is particularly attractive for its good generalization ability. However, fine tuning of SVM kernel parameters is a time-intensive process. In our experiments, ELM-trained NN yielded much better results and trained

Manuscript received November 7, 2005; revised December 30, 2005.

The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: emhlim@ntu.edu.sg).

Digital Object Identifier 10.1109/LGRS.2006.873687

orders of magnitude faster than the SVM. We still report comparisons with SVM due to very limited public data on the use of SVM for multiresolution terrain access.

C. NN

The problem of terrain reconstruction is a problem in function approximation. This problem has been widely studied in the NN community. Supporting multiresolution terrain access is an important step in terrain reconstruction. In recent years, NN have been successfully applied to the problem of three-dimensional surface reconstruction [10], [11]. NN can compare favorably with DT for multiresolution access (especially in applications such as ours that allow offline training and in which the onboard memory is rather limited) because of the following reasons: 1) in standard DT-based techniques, the interpolation is linear, whereas NN inherently perform adaptive nonlinear interpolation; 2) NN, in contrast with DT, exploit the height information of the terrain points (at least indirectly in the feed-forward/feed-back loop); and 3) due to their nonlinear nature, NN inherently perform data compression, and thus, a trained network can yield a very compact representation of the terrain.

D. ELM

Despite the advantages of NN for multiresolution terrain access, NN training algorithms typically iteratively optimize the input and output weight vectors and the neuron biases. This can result in long training times and possibly make NN less attractive in comparison with Delaunay-based interpolation techniques in which both the time to compute map representations and the query time are much shorter.

ELM [12]–[15] is a recently proposed machine learning algorithm that can significantly reduce the amount of time needed to train an NN. The ELM training algorithm in contrast with traditional training methods for NN uses randomly initialized hidden neuron parameters (input weight vectors and neuron biases for additive hidden neurons and centers and impact factors for RBF hidden neurons) and only iteratively computes the output weight vector (see Section III). Our empirical results show superior mse performance of the ELM- over BP-trained NN, the DT-based technique described earlier, and the SVM.

The remaining part of this letter is organized as follows. Section III covers preliminaries of ELM. Section IV presents our simulation results. We conclude this letter in Section V.

III. REVIEW OF ELM

A. Function Approximation on Single Hidden Layer Networks (SLFN)

The output function of a standard SLFN with L hidden neurons is

$$f_L(\mathbf{x}) = \sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \quad \mathbf{x} \in \mathbf{R}^n, \mathbf{a}_i \in \mathbf{R}^n, \beta_i \in \mathbf{R}^m \quad (2)$$

where $G(\mathbf{a}_i, b_i, \mathbf{x})$ is the output of the i th hidden neuron corresponding to the input $\mathbf{x}$, and $\beta_i = [\beta_{i1}, \beta_{i2}, \dots, \beta_{im}]^T$ is the

weight vector connecting the i th hidden neuron and the output neurons.

For additive hidden neurons with activation function $g(x) : \mathbf{R} \rightarrow \mathbf{R}$, the output of the i th hidden neuron is

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i), \quad b_i \in \mathbf{R} \quad (3)$$

where $\mathbf{a}_i$ is the input weight vector connecting the input neurons and the i th hidden neuron, and b_i is the bias of the i th hidden neuron.

For RBF hidden neurons with activation function $g(x) : \mathbf{R} \rightarrow \mathbf{R}$, the output of the i th hidden neuron is

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|), \quad b_i \in \mathbf{R}^+ \quad (4)$$

where $\mathbf{a}_i$ and b_i are the center and impact factor of the i th radial basis function (RBF) hidden neuron. $\mathbf{R}^+$ is the set of all positive reals.

Given N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i)$; $\mathbf{x}_i \in \mathbf{R}^n$ are the inputs and $\mathbf{t}_i \in \mathbf{R}^m$ are the targeted outputs. That a standard SLFN with L hidden neurons can approximate N samples with zero error implies that for some set of values of β_i , $\mathbf{a}_i$, and b_i

$$\sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \dots, N. \quad (5)$$

The above N equations can be compactly written as

$$\mathbf{H}\beta = \mathbf{T} \quad (6)$$

where

$$\mathbf{H}(\mathbf{a}_1, \dots, \mathbf{a}_L, b_1, \dots, b_L, \mathbf{x}_1, \dots, \mathbf{x}_N) = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \dots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \dots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L} \quad (7)$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}, \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m}. \quad (8)$$

$\mathbf{H}$ defines the hidden layer output matrix of the SLFN with the i th column of $\mathbf{H}$ being the i th hidden neuron output with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$.

B. ELM Algorithm

In general, the number of hidden neurons is much less than the number of training samples, i.e., $L \ll N$. This means that $\mathbf{H}$ is a nonsquare matrix, and there may exist values for β_i , $\mathbf{a}_i$, and b_i that satisfy the equality $\mathbf{H}\beta = \mathbf{T}$. However, as discussed in [12]–[15], the parameters of hidden neurons need not be tuned and can be randomly generated permanently according to any continuous probability distribution. This special consideration on the network parameters translates the training of an SLFN to simply finding a least square solution for the network's output bias matrix $\hat{\beta}$ of the linear system $\mathbf{H}\beta = \mathbf{T}$. The unique smallest norm least squares solution of this linear system is

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}. \quad (9)$$

Fig. 1. Terrain A.

$\mathbf{H}^\dagger$ is the Moore–Penrose generalized inverse of the hidden layer output matrix $\mathbf{H}$ [16]. Thus, given a training set $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \dots, N\}$, activation function $g(x)$, and number of hidden neurons L , the ELM algorithm can be summarized in the following three steps:

- Step 1) Randomly assign the values for parameters $\mathbf{a}_i$ and b_i of the hidden neurons, $i = 1, \dots, L$.
- Step 2) Calculate the hidden layer output matrix $\mathbf{H}$.
- Step 3) Calculate the output weight β : $\beta = \mathbf{H}^\dagger \mathbf{T}$.

The universal approximation capability of ELM has been rigorously proved in an incremental method by Huang *et al.* [14]. In selecting the method for calculating Moore–Penrose inverse [17], singular value decomposition (SVD) is chosen for its capability in calculating the Moore–Penrose inverse for all matrices including singular cases.

IV. PERFORMANCE COMPARISON

Three sets of terrain models¹ are used in our simulations. They are shown in Figs. 1–3.

Models A and B comprise 4096 sets of x – y – z coordinates in a 64×64 square matrix, whereas model C is a 1000×1000 matrix. In all simulations, the x, y coordinate values and the elevation values of the terrain points are shifted and scaled so that $\{-1 \leq x, y \leq 1$ and $0 \leq z \leq 1\}$.

All simulations for the ELM² and DT algorithms are carried out in MATLAB 6.5 environment running on a Pentium 4 2.0-GHz central processing unit (CPU) and 512-MB double-data-rate random access memory (DDR RAM). The simulations for SVM are carried out using compiled C-coded SVM packages LIBSVM [18].

Both additive and RBF hidden neuron types of ELM have been tested in our simulations. ELM with sigmoid additive hidden neurons (the ELM is thus denoted as ELM-SIG in this case) has been compared with DT learning algorithm where the

Fig. 2. Terrain B.

Fig. 3. Terrain C.

output of the i th hidden neuron is

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = \frac{1}{1 + \exp(-(\mathbf{a}_i \cdot \mathbf{x} + b_i))}. \quad (10)$$

We also compare BP-RBF network and SVM with ELM-RBF. Gaussian activation function is used in all these three algorithms, and the output of the i th hidden neuron of ELM-RBF is

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = \exp(-b_i \|\mathbf{x} - \mathbf{a}_i\|^2). \quad (11)$$

For ELM, the hidden neurons parameters $(\mathbf{a}_i, b_i)$ are randomly generated from $(-1, 1)^n \times (0, 1)$ based on the uniform probability distribution.

In each case, ten trials were carried out. Our comparisons use the average of the results of these trials. Where applicable, the mean value of the absolute percentage errors is calculated as follows:

$$\text{Mean of absolute percentage error} = \frac{\sum_{i=1}^N \left| \frac{O_i - t_i}{t_i} \times 100\% \right|}{N}. \quad (12)$$

¹The digital elevation models (DEMs) used in this letter are courtesy of U.S. Geological Survey (USGS) Website. Model C is the DEM of Las Vegas, NV at 1:250 000 scale. Models A and B are cropped from model C. The raw DEM data can be downloaded from http://edcsgs9bb.cr.usgs.gov/glis/hyper/guide/1_dgr_demfig/states/NV.html.

²The source code of ELM can be found from <http://www.ntu.edu.sg/home/egbhuang/>.

TABLE I
MEAN OF ABSOLUTE PERCENTAGE ERROR OF ELM-SIG

		Map Size								
		100 by 100	200 by 200	300 by 300	400 by 400	500 by 500	600 by 600	700 by 700	800 by 800	900 by 900
Number of neurons	5	5.2866	6.113	7.0777	4.9347	6.1287	5.8024	7.3902	4.3153	6.8238
	10	3.035	3.7153	4.0781	3.2954	3.0531	3.2781	3.4845	3.4721	3.3083
	15	3.1086	2.6678	2.9333	2.7466	2.6502	2.9485	3.0723	3.0056	2.939
	20	2.6194	2.6383	2.9129	2.6702	2.5802	2.6823	2.8034	2.7024	2.5438
	25	2.4814	2.5139	2.5011	2.5543	2.5329	2.4538	2.516	2.4892	2.5338
	50	2.1032	2.1344	2.0855	2.1472	2.0746	2.1379	2.1954	2.0887	2.1356
	100	1.9521	1.9655	1.9229	1.905	1.9204	1.945	1.9463	1.9111	1.9074
	150	1.8689	1.8751	1.8953	1.8521	1.9028	1.9071	1.8436	1.9039	1.8485
	200	1.8626	1.8796	1.8761	1.8527	1.8828	1.863	1.8294	1.8466	1.8537

Fig. 4. Comparison of memory needed in DT versus ELM-SIG. The solid line represents minimum number of sample points, N , required for DT to match the accuracy of a 25-node ($L = 25$) ELM-SIG (shown as dashed line).

O_i is the real output of the network, t_i is the target output corresponding to the i th tested data point, and N is the number of tested data points.

A. ELM-SIG and DT

Raw data in terrain model C (Fig. 3) is downsampled to give nine terrain models with sizes 100×100 , 200×200 , ..., 900×900 . For each terrain model, 1000 sample points are randomly selected to form the sample set for training the ELM-SIG network over a set of 5–200 neurons. The remaining data points of the terrain model are used to test the network. The results are shown in Table I. For DT, we iteratively increase the number of sample points selected to form the triangulation for each terrain size and compare the accuracy performance with that of ELM-SIG in Table I.

Without loss of generality, we focus our attention on two cases, one involving 25 hidden neurons and the other with 150 hidden neurons. Memory requirement when DT-based interpolation is used is compared with memory needed when ELM-SIG is used in Figs. 4 and 5.

DT(N) requires a storage of $13.5N$ floating points [19]. In the case for ELM-SIG-trained NN, the N sample points are only required during the training of the network; thereafter, these sample points can be “forgotten.” The only data that the

Fig. 5. Comparison of memory needed in DT versus ELM-SIG. The solid line represents minimum number of sample points, N , required for DT to match the accuracy of a 150-node ($L = 150$) ELM-SIG (shown as dashed line).

network requires to be stored are the hidden node parameters, which depend on the dimension of the inputs and the number of hidden neurons implemented. In our terrain reconstruction case, the dimension of the inputs is 2, and the number of hidden neurons used is L , which is much lesser than the number of known sample points N . With L hidden neurons, we have a total of $2L$ interconnections between input and hidden layer, giving us $2L$ input weights and L hidden neuron biases. Together with L output weights, a total of $4L$ parameters will be needed for the network to describe the same terrain model.

A typical 50-node single-layer NN would need 200 parameters to model a terrain. For DT to maintain equivalent memory consumption, it would have resulted in a highly low-resolution model using only 15 samples.

In Fig. 4, DT, which requires 13.5 times the number of sample points, needs more floating point parameters than the 25-node ELM model using 100 floating point parameters. Fig. 5 shows the same results with the 150-node 600-parameter ELM model. It is clear that as the terrain size increases, DT will require much more data points to represent the terrain, whereas ELM-SIG maintains a very much lower neuron count, and hence, much smaller memory is required.

B. ELM-RBF and RBF NN

One thousand points are randomly selected from 4096 data points in terrain models A and B to form the training set for both ELM-RBF and normal RBF network trained by BP algorithm. The networks are trained over 50–200 neurons. The remaining data points are used to test the terrain models obtained from the two algorithms.

Tables II and III show the performance comparison between ELM-RBF with BP-trained RBF on the two different terrain sets. Again, without loss of generality, we present the results for 50 and 200 neurons. Both BP and ELM networks use the RBF activation function [cf. (11)].

The ELM-RBF surpasses the BP-RBF in the training time, although both networks use the same number of hidden neurons. ELM-RBF networks also achieve a much better accuracy

TABLE II
PERFORMANCE COMPARISON BETWEEN ELM-TRAINED AND BP-TRAINED
RBF NETWORK USING GAUSSIAN RBF KERNEL ON TERRAIN A

Algorithm	ELM-RBF		BP-RBF	
# Neurons	50	200	50	200
MSE	0.0074	0.00631	0.0257	0.0211
Mean of Absolute % Error	5.98	4.87	11.74	9.32
Training Time (s)	0.0547	0.515	11.40	75

TABLE III
PERFORMANCE COMPARISON BETWEEN ELM-TRAINED AND BP-TRAINED
RBF NETWORK USING GAUSSIAN RBF KERNEL ON TERRAIN B

Algorithm	ELM-RBF		BP-RBF	
# Neurons	50	200	50	200
MSE	0.01483	0.01269	0.0304	0.0235
Mean of Absolute % Error	9.67	8.16	14.18	12.31
Training Time (s)	0.0516	0.4922	11.43	75.8

TABLE IV
PERFORMANCE COMPARISON BETWEEN SVM
AND ELM-RBF ON TERRAIN A

Algorithm	SVM ($C = 2048$ $\gamma = 8$)	ELM-RBF	
# Neurons/SVs	295.3	50	100
MSE (testing data)	0.06751	0.00784	0.00655
% Improvement	NA	861.1	1030.7
Training Time	708.11	0.0547	0.2189
Testing Time	1.28	0.0719	0.1281

TABLE V
PERFORMANCE COMPARISON BETWEEN SVM
AND ELM-RBF ON TERRAIN B

Algorithm	SVM ($C = 4096$ $\gamma = 8$)	ELM-RBF	
# Neurons/SVs	252.8	50	100
MSE (testing data)	0.07296	0.0148	0.0131
% Improvement	NA	492.97	556.95
Training Time	1193.7	0.0516	0.201
Testing Time	2.31	0.0702	0.117

compared with BP-RBF networks. Inasmuch as both networks have the same topology and use the same activation function, they have the same testing or query time (query time is independent of the algorithm used for training).

C. ELM-RBF and SVM

Table IV shows the performance of ELM-RBF network and SVM on Terrain A. Out of 4096 data points, 1000 points are selected randomly to form the training set. The remaining points form the testing set. A total of ten trials were conducted, and the results were averaged and tabulated.

From the table, with the optimized parameters for SVM being $C = 2048$ and $\gamma = 8$, it takes more than 700 s to train SVM, whereas ELM-RBF takes less than a second to train, yet yielding a remarkable performance improvement of 861% with 50 neurons and 1030% with 100 neurons. The number of vectors for the SVM is also much higher than the number of neurons used in the ELM-RBF network, causing the recall and testing time to be very costly.

Using Terrain B, which has a more undulating terrain characteristic, we can see in Table V that SVM at $C = 4096$ and $\gamma = 8$ requires a much longer time to train and test, yielding an accuracy which ELM-RBF easily surpasses by at least five times with only 50 neurons.

V. CONCLUSION

This letter investigates the problem using NN for supporting multiresolution terrain access. This problem has been less studied in literature, probably due to the large time needed to train NN to achieve acceptable error rates during the query stage. We use the ELM training algorithm to dramatically speed up the rate at which the network learns a priori available maps. ELM achieves this by avoiding iterative optimization of the hidden neuron parameters (input weight vector and the biases for additive hidden neurons and the centers and impact factors for RBF hidden neurons).

When ELM is used for training, mse during the query stage is competitive or better than those obtained when BP-trained NN, interpolation using DT, or SVM are used.

Another advantage of our proposed technique for queries on large maps is that the ELM needs far less memory than that needed by DT to achieve the same levels of MSE errors.

REFERENCES

- [1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 2005, pp. 77–167.
- [2] L. E. Dubins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, no. 3, pp. 497–516, Jul. 1957.
- [3] F. Ruffier and N. Franceschini, "Optic flow regulation: The key to aircraft automatic guidance," *Robot. Auton. Syst.*, vol. 50, no. 4, pp. 177–194, Mar. 2005.
- [4] S. Rippa, "Minimal roughness property of the Delaunay triangulation," *J. Comput. Aided Geom. Des.*, vol. 7, no. 6, pp. 489–497, Nov. 1990.
- [5] R. J. Fowler and J. J. Little, "Automatic extraction of irregular network digital terrain models," in *Proc. Int. Conf. Comput. Graph. Interactive Tech.*, 1979, pp. 199–207.
- [6] J. Lee, "A comparison of existing methods for building irregular networks models of terrain from grid digital elevation models," *Int. J. Geog. Inf. Syst.*, vol. 5, no. 3, pp. 267–286, 1991.
- [7] M. Garland and P. Heckbert, "Fast polygonal approximation of terrains and height fields," Carnegie Mellon Univ., Pittsburgh, PA, Tech. Rep. CMU-CS-95-181, 1995.
- [8] H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Neural Information Processing Systems*, vol. 9, M. Mozer, J. Jordan, and T. Petsche, Eds. Cambridge, MA: MIT Press, 1997, pp. 155–161.
- [9] A. J. Smola and B. Scholkopf, "A tutorial on support vector regression. NeuroCOLT2," Univ. London, London, U.K., Tech. Rep. NC2-TR-1998-030, 1998.
- [10] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *Proc. ACM SIGGRAPH*, Dec. 2001, pp. 67–76.
- [11] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell, "Smooth surface reconstruction from noisy range data," in *Proc. ACM GRAPHITE*, Feb. 2003, pp. 119–126.
- [12] G.-B. Huang and C.-K. Siew, "Extreme learning machine: RBF network case," in *Proc. Int. Conf. Control Autom. Robot. Vis.*, Dec. 2004, pp. 1651–1663.
- [13] —, "Extreme learning machine with randomly assigned RBF kernels," *Int. J. Inf. Technol.*, vol. 11, no. 1, pp. 16–24, 2005.
- [14] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, 2005, to be published.
- [15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. Int. Joint Conf. Neural Networks*, Jul. 2004, pp. 985–990.
- [16] D. Serre, *Matrices: Theory and Applications*. New York: Springer-Verlag, 2002.
- [17] J. M. Ortega, *Matrix Theory*. New York: Plenum, 1987.
- [18] C.-C. Chang and C.-J. Lin, *LIBSVM—A Library for Support Vector Machines*. [Online]. Available: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- [19] R. J. Renka, "Algorithm 751; TRIPACK: A constrained two-dimensional Delaunay triangulation package," *ACM Trans. Math. Softw.*, vol. 22, no. 1, pp. 1–8, Mar. 1996.