Neural Comput & Applic (2013) 22:469–477
DOI 10.1007/s00521-012-0866-9

Extreme learning machine terrain-based navigation for unmanned aerial vehicles

Ee May Kan · Meng Hiot Lim · Yew Soon Ong · Ah Hwee Tan · Swee Ping Yeo

Received: 10 August 2011 / Accepted: 25 January 2012 / Published online: 9 February 2012

Abstract Unmanned aerial vehicles (UAVs) rely on global positioning system (GPS) information to ascertain its position for navigation during mission execution. In the absence of GPS information, the capability of a UAV to carry out its intended mission is hindered. In this paper, we learn alternative means for UAVs to derive real-time positional reference information so as to ensure the continuity of the mission. We present extreme learning machine as a mechanism for learning the stored digital elevation information so as to aid UAVs to navigate through terrain without the need for GPS. The proposed algorithm accommodates the need of the on-line implementation by supporting multi-resolution terrain access, thus capable of generating an immediate path with high accuracy within the allowable time scale. Numerical tests have demonstrated the potential benefits of the approach.

Keywords Unmanned aerial vehicles (UAVs) · Extreme learning machines (ELM) · Terrain-based navigation

E. M. Kan (✉) · M. H. Lim
Intelligent Systems Center,
Nanyang Technological University,
Singapore, Singapore
e-mail: ka0001ay@ntu.edu.sg

M. H. Lim
e-mail: emhlim@ntu.edu.sg

Y. S. Ong · A. H. Tan
School of Computer Engineering,
Nanyang Technological University,
Singapore, Singapore
e-mail: asysong@ntu.edu.sg

A. H. Tan
e-mail: asahtan@ntu.edu.sg

S. P. Yeo
Center for Microwave and RF,
National University of Singapore,
Singapore, Singapore
e-mail: eleyeosp@nus.edu.sg

1 Introduction

Unmanned aerial vehicles (UAVs) are useful for military and law enforcement operations [1–3] such as environmental surveillance, battlefield assessment, ordnance delivery, resource assessment, etc. This trend is likely to continue with UAVs poised to replace what used to be high-risk human-in-the-loop missions (particularly in situations that are hazardous for human operators). During mission execution, UAVs rely on global positioning system (GPS) information to ascertain its position for navigation. However, GPS devices rely on information from global navigation satellite system (GNSS) about global positional information, including the longitude, the latitude and the time when the signal was released from the satellites. When signals lapse, it can cause a breakdown of information, causing the GPS device to operate in an erroneous mode or possibly shutting down. Moreover, there are certain atmospheric indicators which can cause the inaccurate mapping of certain areas, such as vast waterways or terrain, large structures, electronic interference, or dense foliage. All these can affect signal reception resulting in positional errors or even false readings.

In the absence of GPS information, the capability of a UAV to carry out its intended mission is hindered. In this sense, the UAV is incapacitated and if the disruption is too serious, the success of the mission is jeopardized. It is

therefore necessary to find alternative means for UAVs to derive positional reference information to ensure the continuity of the mission. In recent years, there have been significant progresses made in terrain-based navigation [4, 5] due to more widespread availability of sensors that can directly sense terrain while the platform carrying the sensor is in motion. The technique of terrain based navigation has been utilized by cruise missiles and aircrafts over land, aiding the navigation system based on terrain information in order to increase the estimation accuracy.

In principle, Delaunay triangulation (DT) interpolation is directly applicable to estimate elevations of points in terrain modeling. However DT requires more floating point parameters to represent a terrain [3]. In the case of UAV where the amount of memory allocated for storage of terrain data is preferably small so that much of the memory resources can be dedicated to other components onboard the UAV. Not denying the fact that the current level of technology have resulted in cheap and readily available memory resources, a more compact and efficient representation of geographical and terrain data is still advantageous.

Our focus here is to explore a connectionist approach as a basis for deriving positional reference information based on the terrain data captured by the UAV onboard sensor. The objective is to estimate the state of the UAV (position and attitude) and a map of the surrounding environment simultaneously based on limited sensing capabilities. The sensed terrain information is correlated with existing reference data to derive the navigation estimate. We present a scheme based on extreme learning machine (ELM) [6–10] as a mechanism for learning the stored digital elevation information to perform the estimation or map building for UAV navigation without the need for GPS or other derive positioning information. In earlier work [3], ELM training algorithm was applied to dramatically speed up the rate at which the network learns a priori available maps. The results presented in [3] show superior mean square error (MSE) performance of the ELM over other approaches. The parameters of hidden neurons need not be tuned and can be randomly generated according to any continuous probability distribution. 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 radial basis function hidden neurons). The advantages of the ELM are that it has only one tunable parameter, specifically the number of neurons, and its training algorithm consists of only a single step. Thus the ELM needs far less memory than that needed by other approaches. Additionally, the ELM offers a fast decomposition of a function at different levels of resolution as shown by Yeu et al. [3]; therefore it can be implemented online to reduce the computational cost dramatically. In the remaining part of this paper, rather than focusing on discussions over the practical motivation of terrain-based navigation problem, a more thorough discussion on the application of ELM to the problem is presented.

2 Problem formulation

In this paper, a terrain-based navigation problem is defined as deriving positional reference information based on the state of the vehicle (position and attitude) and a map of the surrounding environment captured by onboard sensors. Various sensors (e.g., cameras, radars, laser scanners, satellite imagery) having different range and resolution characteristics are employed to collect information about the environment the vehicle operates in. A computationally efficient terrain modeling method, specifically adopted for on-line implementation, should therefore choose the expedient information from all these sensors, and use the on-board computational resources to aid UAVs navigate through terrain without the need of GPS. We provide a survey on various terrain modeling methods and compare their performances with the ELM training algorithm for terrain based navigation.

2.1 Delaunay triangulation (DT)

The Delaunay triangulation for a set N of points in the plane is the triangulation DT(N) of N such that no point in N is inside the circumcircle of any triangle in DT(N). DT has a time complexity of $O(N \log N)$ and interpolation algorithms based on DT are widely used in terrain elevation estimation because DT minimizes the coarseness of the estimated terrain over all possible data-independent triangulations [11]. The accuracy of the terrain model computed by DT can be significantly influenced by the selection scheme used for the sample points [12, 13]. In this paper, we adopt a uniform random selection scheme for choosing the sample points from the raw dataset. The sample points are triangulated using their (x, y) coordinates and the elevation information of any point not in the sample set can be interpolated using Eq. 1:

$$z = ax + by + c \tag{1}$$

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

2.2 Resilient back propagation (R-BP)

With most neural networks using sigmoid functions in their hidden layers, one of the side-effects of implementing BP

is that the gradient calculated may be too small in magnitude even when the current iteration is still far from the optimal solution. This is due to the nature of the gradient of sigmoid functions approaching to zero rapidly when the inputs get large. R-BP [14] addresses this issue by ignoring the magnitude of the gradient in the calculation of the update function for the next iteration, choosing to use just the direction (or sign) of the consecutive gradients to determine whether the update value should be increased or decreased in magnitude. Consecutive gradients in the same direction will have a boost in the update value while a switch in the direction will result in a reduction of the update factor.

2.3 Quasi-Newton back propagation (QN-BP)

QN-BP [15] follows the typical Newton's method but without the computation of the Hessian matrix, **H**, of the network at each iteration. QN-BP involves the generation of a sequence of matrices $\mathbf{F}^{(\mathbf{k})}$ that represent increasingly accurate approximation of $\mathbf{H}^{-1}$. QN-BP offers a fast alternative to conjugate gradient methods in optimizing a NN. In spite of it being a fast optimization algorithm for BP trained NN, QN-BP requires additional memory for retaining the approximate Hessian matrix. As such QN-BP is normally found in smaller networks with lower neuron count.

2.4 One-step secant back propagation (OSS-BP)

OSS-BP [16] improves upon QN-BP' memory requirement by not storing the complete Hessian matrix for every iteration. Instead, OSS-BP assumes that the previous Hessian matrix is an identity matrix. As such, a new search direction can be computed less expensively without involving matrix inversion.

2.5 Levenberg-Marquardt back propagation (LM-BP)

LM-BP [17] provides a middle ground between traditional GD-BP and QN-BP. LM-BP makes use of the approximation of the Hessian matrix as shown in Eq. 2:

$$H = J^T J \tag{2}$$

where **J** is the Jacobian matrix that comprises of the first derivative of the errors with respect to the network weights and biases. The corresponding gradient is given by Eq. 3:

$$g = J^T e \tag{3}$$

where e is the network error vector.

2.6 ELM

The architecture of ELM [6–9] is similar to that of a single-layer feed-forward neural network; the only difference is that there is no bias for the output neuron. Every neuron in the input layer is connected to all other neurons in the hidden layer. All hidden layer neurons are also provided with a bias. The activation function for the output neuron layer is linear, while that of the hidden neuron layer can be any piecewise continuous function. The extreme learning machine employs a completely different algorithm for calculating weights and biases that can significantly reduce the amount of time needed to train a neural network. In the case of extreme learning machines, the weights and biases between the hidden layer and input layer neurons are randomly assigned. For an extreme learning machine having "j" hidden layer neurons trained on a data set having "k" training cases and "i" input neurons, the activation of all hidden layer neurons is calculated for every training case using the following formula.

$$H_{jk} = g(\Sigma(W_{ji}X_{ik}) + B_j) \tag{4}$$

where $g(\cdot)$ is any nonlinear piecewise continuous activation function, W_ji is the weight between the ith input neuron and jth hidden layer neuron, B_j represents the bias for the jth hidden layer neuron, X_ik is the input at the ith input neuron of the kth training case, and H_{jk} is the matrix containing activation of the jth hidden layer neuron for the kth training case. The activation of all the hidden layer neurons for all training samples is represented by a matrix H, which has k rows and j columns. The H matrix is called the hidden layer output matrix of the neural network. The weights between hidden layer neurons and the output neuron are determined by performing a least-squares fit to the target values in the training set against the output of the hidden layer neurons for each training case. In mathematical notation, this is equivalent to solving the following linear system.

$$H_{k\times j}\beta_{j\times 1} = T_{k\times 1} \tag{5}$$

$$\beta = (\beta_1 \ldots \beta_j)_{j\times 1} \tag{6}$$

where β is a vector representing weights between hidden layer neurons and the output layer neuron.

$$\mathbf{T} = (T_1 \ldots T_k)_{k\times 1} \tag{7}$$

where **T** is the vector representing targets for all training cases. To obtain the weights, the above system is solved by multiplying the Moore-Penrose pseudo-inverse of the **H** matrix with **T**.

$$\beta = \mathbf{H}'\mathbf{T} \tag{8}$$

β = vector of weights between hidden layer and output layer neurons, $\mathbf{H}'$ = Moore-Penrose pseudoinverse of matrix $\mathbf{H}$, and $\mathbf{T}$ = vector containing targets for the training case. This completes the training of the network. Thus training of an extreme learning machine involves only two steps: (1) random assignment of weights and biases to hidden layer neurons and calculation of the hidden layer output matrix $\mathbf{H}$; (2) calculation of output weights using the Moore-Penrose pseudo-inverse of matrix $\mathbf{H}$ using the values of targets for training cases.

The training process is fast as it involves finding the Moore-Penrose inverse of the hidden layer matrix, which is done much faster than normal epoch based training algorithms such as Levenberg-Marquardt; also the training only relies on a closed-form solution and does not involve any kind of nonlinear optimization routines. Consequently, the training time is significantly reduced and the only parameter left to be tuned is the number of hidden layer neurons [6–9]. The extreme learning machine works by making use of a large number of random nonlinear projections of input space. Each neuron corresponds to a single projection. In the case of a conventional artificial neural network each of these projections is tuned. In the case of the extreme learning machine linear regression is performed on these projections; projections that match the curve get higher weights. This way a line is fitted to the training points in the space of hidden layer neurons. The universal approximation capability of ELM has been rigorously proven in an incremental method by Huang et al. [6–9].In selecting the method for calculating Moore-Penrose inverse [18], singular value decomposition (SVD) is chosen for its capability in calculating the Moore-Penrose inverse for all matrices including singular cases.

3 Implementation

3.1 Navigational terrain

In our simulations, we exploit the terrain digital elevation model (DEM) used in the earlier work [3] as a navigational space for UAV. The raw data set consists of 1,000 × 1,000 elevation data in a square uniform grid. The visualization of the terrain is as shown in Fig. 1. In all simulations, the x, y coordinates of the data points and the elevation value z are scaled and shifted such that $\{-1 \leq x,\ y \leq 1$ and $0 \leq z \leq 1\}$.

To illustrate the terrain based navigation scenario, consider a UAV navigating through the terrain (as shown in Fig. 1) based on limited sensing capabilities. The UAV updates the map of the surrounding environment simultaneously based on the state of the UAV (position and attitude). The objective of the UAV is to navigate through the terrain while circumventing the obstacles over a certain elevation threshold. Since the real-time terrain based navigation problem at the finest resolution is computationally prohibitive, the proposed approach accommodates the need of the on-line implementation by limiting the amount of time to process.

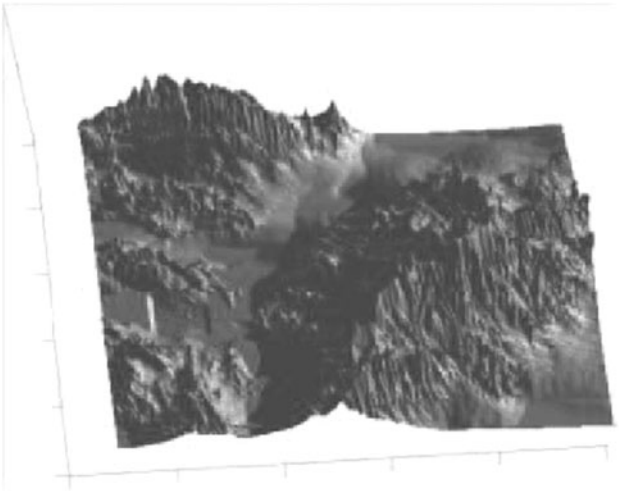

Fig. 1 Visualization of Las Vegas, NV DEM

3.2 ELM training algorithm

We make use of ELM training algorithm to achieve this remarkable speed improvement by using randomly initialized hidden neuron parameters and only iteratively compute the output weight vector. The algorithm greatly cuts down the training time of neural networks (NN), in contrast to the conventional training methods such as back-propagation method. It is therefore capable of generating a terrain model that offers data compression as well as reasonably good approximation of the actual terrain. Due to their nonlinear nature, ELM inherently performs data compression, and thus, a trained network yields a very compact representation of the terrain. Based on the representation of the terrain, an UAV is competent to navigate through the terrain that may comprise of different resolutions. Figure 2 shows an example of the multi-resolution approximation of the environment.

We employ the ELM training algorithm to perform the required multi-resolution terrain access. It is assumed that the UAV navigates over the terrain, while modeling the terrain with the elevation data gathered from a proximity sensor. The input data to the algorithm are given by the raw data of the terrain captured by the onboard sensor, which is then down sampled into a 100 × 100 map with 10K data points, a 200 × 200 map with 40k data points and a 300 × 300 map with 90k data points. For each sensory region, 10% of the data points are randomly selected to

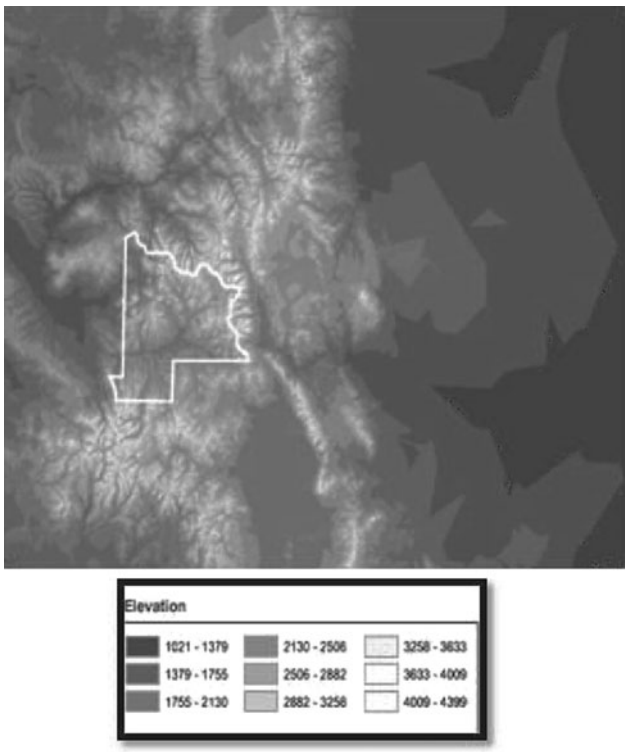

Fig. 2 The multi-resolution representation with altitude ranging from 1,000 to 4,000 m

form the triangle map for DT and the training set for the ELM with sigmoid additive hidden neurons (the ELM is thus denoted as ELM-SIG in this case) over a set of 5–200 neurons. The remaining data points are used to test the accuracies of the terrain model obtained from the algorithm. The training algorithm inherently performs adaptive nonlinear interpolation and exploits the height information of the terrain points. The idea is to employ high resolution close to the elevation threshold, and a coarse resolution at large distance from elevation threshold based on the current location of the UAV.

Both additive and RBF hidden neuron types of ELM have been tested in our simulations. ELM-SIG network has been compared with DT learning algorithm where the output of the ith hidden neuron is

$$G(a_i, b_i, x) = \frac{1}{1 + \exp(-(a_i \cdot x + b_i))} \tag{9}$$

We also compare BP-RBF network with ELM-RBF. Gaussian activation function is used in all these algorithms, and the output of the ith hidden neuron of ELM-RBF is as per Eq. 10

$$G(a_i, b_i, x) = \exp(-b_i ||x - a_i||^2) \tag{10}$$

For ELM, the hidden neurons parameters (a_i, b_i) are randomly generated from $(-1, 1)^n \times (0, 1)$ based on a uniform probability distribution. For BP and ELM, the number of hidden nodes is gradually increased by an interval of 5 and the nearly optimal number of nodes for BP and ELM are then selected based on cross-validation method. 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:

$$\text{Mean of absolute \% error} = \frac{\Sigma_{i=1}^{N} \left| \frac{O_i - t_i}{t_i} \times 100\% \right|}{N} \tag{11}$$

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

Figures 3, 4 and 5 show the relationship between the generalization performance of ELM and its network size for terrain modeling. As observed from these figures, the generalization performance of ELM is very stable on a wide range of number of hidden nodes. 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 the ELM. Without lost of generality, we focus our attention on 50 hidden neurons. DT (N) requires a storage of $13.5N$ floating points [19]. In the case for ELM-trained NN, the N sample points are only required during the training of the network; thereafter, these sample points can be discarded. The only data that the 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 modeling 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, i.e. $L \ll N$. 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. Yeu et al. [3] have conducted a good comparison of the memory requirement between DT-based interpolation and ELM. It is clear in [3] that as the terrain size increases, DT would require much more data points to represent the terrain, whereas ELM maintains a very much lower neuron count, and hence, much smaller memory is required. The generated terrain model (within a sensory region based on the current location of the UAV) can be used to build a network by considering the elevation threshold of the topography. Once this network is available, navigation routes can be determined for different routing objectives (i.e., shortest path; safest path; traveling salesman problem) to aid UAVs to

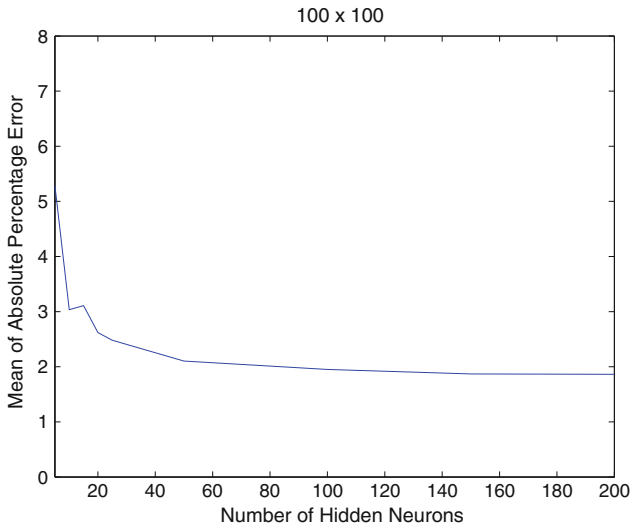

Fig. 3 Mean of absolute percentage error against number of hidden nodes for a 100 × 100 map of terrain

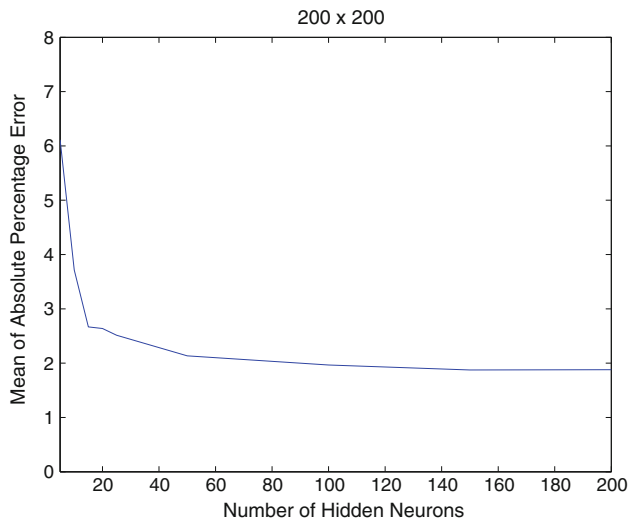

Fig. 4 Mean of absolute percentage error against number of hidden nodes for a 200 × 200 map of terrain

navigate through the terrain efficiently without the need of GPS. The following section provides a detailed description of the network generation procedure within the sensory region.

3.3 Generation of nodes in a trained network

To begin with, the density at which the nodes that will constitute the vertices of the network that are to be generated latitudinally and longitudinally is specified. The number of nodes will serve as a basis for constructing the neural networks for terrain segments. In principle, depending on the profile of the terrain that the network is trained to represent, different resolutions may be employed

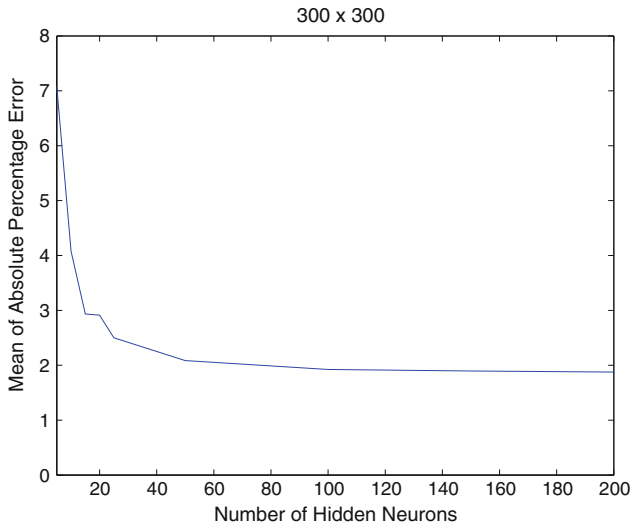

Fig. 5 Mean of absolute percentage error against number of hidden nodes for a 300 × 300 map of terrain

to enhance efficiency. Nodes below the elevation threshold are extracted and used for the generation of links. Thus, the nodes above the elevation threshold are discarded; given that at those elevations the UAVs could be in danger. The number of rows and columns determine the density of the nodes to capture equally likely the characteristics of the terrain structure obtained from the multi-resolution terrain modeling.

3.4 Generation of links in a trained network

The output from the node generation procedure is a grid with nodes placed at the center of the vertices which are below the elevation threshold. These nodes form a set of potential nodes for the UAV to navigate through. Two important parameters are identified for terrain based navigation: (1) the link length and (2) the maximum elevation on the link. These parameters are based on the operation of the UAVs; the purpose, logistics, and safety of the mission. A desirable maximum and minimum length need to be specified (i.e., MIN $\leq$ link length $\leq$ MAX) for the links. The maximum elevation of the links is considered a constraint ensuring that at any given time, the link connecting two nodes does not cross over an area with elevation greater than the elevation threshold. The output of this stage is a set of links connecting a subset of the nodes resulting from the node generation process. The elevation of the nodes in the center of each vertex has values below the elevation threshold. Those nodes characterized by low resolution are located in the low elevation areas. Given the flat areas that characterize low elevation areas, links can generally be connected in all directions without exceeding the elevation threshold. An illustration of the resulting

network based on the current location of UAV (star) is shown in Fig. 6.

Through the generated network, UAVs are able to estimate their own position in the environment and navigate to target location to accomplish their missions without the need of GPS. An outline of the ELM procedure for the proposed approach is listed in Fig. 7.

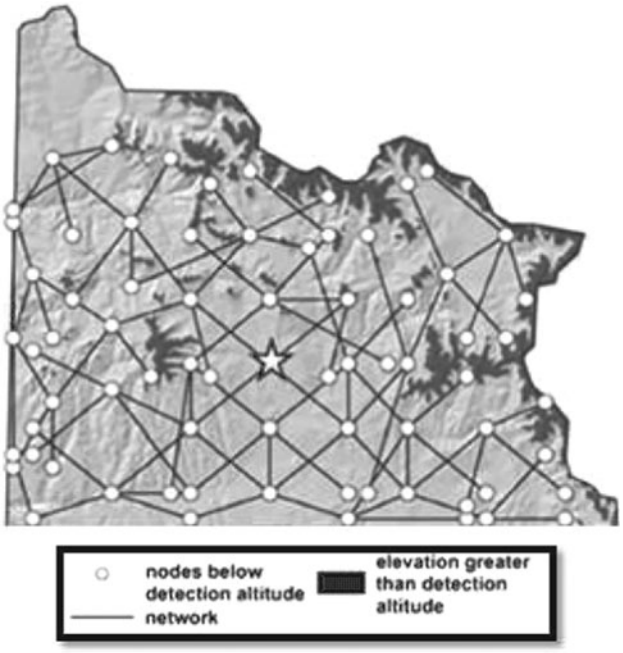

Fig. 6 Generated network within the sensory region

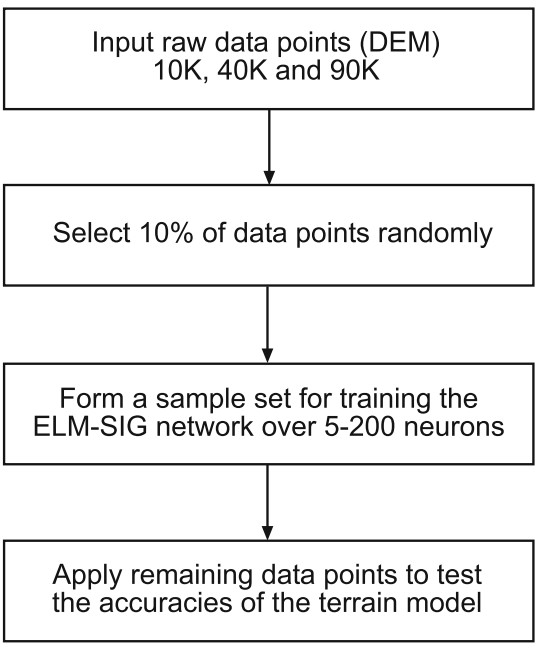

Fig. 7 Overview of the proposed approach

All simulations for the ELM algorithms, BP algorithms and DT are carried out in MATLAB 7 environment running on a Core 2 Quad 2.67 GHz central processing unit (CPU) and 2 GB double-data-rate random access memory (DDR RAM). By utilizing ELM training algorithm, the computational cost is significantly reduced. The corresponding performance accuracies in terms of mean square error (MSE), training times and testing times are shown in Tables 1, 2 and 3.

The gain in the tables refers to gain value in the activation function. Research studies [20, 21] show that gain of the activation function have a significant impact on training time. Thus higher values of gain can cause instability. In this case, the ELM surpasses all the other BP algorithms in the training time even though all networks use the same number of hidden neurons. ELM also achieves competitive accuracy compared to all the other BP networks. Since all

Table 1 Performance comparison on a 100 × 100 map of terrain

	Test MSE	Train time	Gain	Test time
ELM	0.04462	0.0414	0.00	0.0981
LM-BP	0.01105	0.4923	9.801	0.1404
QN-BP	0.04311	0.8883	18.414	0.1413
QSS-BP	0.04889	1.4913	31.518	0.1539
R-BP	0.04614	1.2942	27.324	0.1404
DT	0.00126[a]		0.0594	

[a] Requires larger memory consumption

Table 2 Performance comparison on a 200 × 200 map of terrain

	Test MSE	Train time	Gain	Test time
ELM	0.04462	0.0567	0.00	0.1539
LM-BP	0.01105	0.5625	8.028	0.2115
QN-BP	0.03735	1.2096	18.297	0.1971
QSS-BP	0.04113	2.475	38.385	0.1962
R-BP	0.04785	1.379	20.97	0.2106
DT	0.000787[a]		0.0954	

[a] Requires larger memory consumption

Table 3 Performance comparison on a 300 × 300 map of terrain

	Test MSE	Train time	Gain	Test time
ELM	0.0401	1.1583	0.00	2.25
LM-BP	0.0099	6.0192	3.78	2.9385
QN-BP	0.0305	15.9059	11.457	2.9673
QSS-BP	0.03816	29.007	21.636	2.925
R-BP	0.03573	15.4125	11.079	2.9394
DT	0.000122[a]		2.034	

[a] Requires larger memory consumption

Table 4 Computation cost of the proposed multi-resolution terrain access

Terrain model trained by ELM	0.154 s
Terrain model computed by DT	2.034 s
Terrain model trained by LM-BP	6.019 s
Average time of node generation	0.202 s
Average time of constructing the connectivity relationship in a network	0.232 s

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). As for the comparison with DT, ELM scores similar total time taken to complete the terrain modeling. It should be noted that at 10% training proportion, DT is storing about 13.5 times the number of training points in the generated terrain model. Hence, in terms of memory consumption, ELM and DT are considerably not equivalent. It has been shown in [3] that to achieve equivalent MSE performances with ELM, DT would have to exploit significantly larger number of parameters in its terrain model. As the terrain size increases, DT will require much more data points to represent the terrain, whereas ELM maintains a very much lower neuron count, and hence, much smaller memory is required. Therefore even though DT can provide a more accurate representation of the terrain, in situations where memory capacity is severely limited, ELM would have offered a better solution. Table 4 shows the computational cost of the proposed navigation approach. The algorithm computes the multi-resolution representation using the ELM algorithm and it takes 1.592 s for execution. On the other hand, DT and BP algorithms require much longer time for execution, compared to ELM. With the knowledge of the execution time of the proposed approach, autopilot manages not only to execute the basic tasks such as data acquisition and processing, inner loops control, and etc., but also to plan a route in a seamless manner within the allowable time scale. Thus, the proposed approach is scalable and can be tailored to the available computational resources.

4 Conclusion

In this paper, we derive positional reference information based on the terrain data captured by the UAV onboard sensors. The proposed procedure has the following advantages. First, it considers elevation threshold based on multi-resolution representation. Second, it incorporates elevation for the links in the network. Third, it creates a network based on the terrain model, which in turn allows this network to be used for routing a fleet of vehicles. And finally it uses ELM as a mechanism for learning the stored elevation data which introduces competitive solution for UAVs to navigate through terrain without the need of GPS. On the whole, this paper has shown how ELM can be used as representation of patterns or models. The further work is to consider training the input weights with R-ELM [22] and FIR-ELM [23] so as to improve the performance under noisy environment. In the context of memetic computing [24], an ELM can be perceived as a meme. By exploiting the recurrence or persistency of patterns [25, 26] spanning over a region described by its DEM, the efficacy of ELM can be further improved. Further enhancement can be achieved by means of reconfigurable and context independent hardware such as that presented in [27].

Acknowledgments The authors gratefully acknowledge the funding provided by Temasek Defence Systems Institute, Singapore.

References

1. Agarwal A, Lim MH, Er MJ, Nguyen TN (2007) Rectilinear workspace partitioning for parallel coverage using multiple UAVs. Adv Robot 21(1):105–120
2. Lim KK, Ong YS, Lim MH, Agarwal A (2008) Hybrid ant colony algorithm for path planning in sparse graphs. Soft Comput 12(10):981–994
3. Yeu CW, Lim MH, Huang G, Agarwal A, Ong YS (2006) A new machine learning paradigm for terrain reconstruction. IEEE Geosci Remote Sens Lett 3(3):382–386
4. Kim J, Sukkarieh S (2004) Autonomous airborne navigation in unknown terrain environments. IEEE Trans Aerospace Electron Syst 40:1031–1045
5. Kosecka J, Li F (2004) Vision based topological Markov localization. In: 23rd IEEE international conference on robotics and automation 2:1481–1486
6. Huang G-B, Siew C-K (2004) Extreme learning machine: RBF network case. Proc Int Conf Control Autom Robot Vis 2: 1029–1036
7. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. Proc Int Joint Conf Neural Netw 2:985–990
8. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
9. Huang G-B, Wang DH, Lan Y (2011) Extreme learning machines: a survey. Int J Mach Learn Cybern 2(2):107–122
10. Wu J, Wang S, Chung F-l (2011) Positive and negative fuzzy rule system, extreme learning machine and image classification. Int J Mach Learn Cybern 16(8):1408–1417
11. Rippa S (1990) Minimal roughness property of the Delaunay triangulation. J Comput Aided Geometr Des 7(6):489–497
12. Fowler RJ, Little JJ (1979) Automatic extraction of irregular network digital terrain models. Intl Conf Comput Graph Interact Tech 13(2):199–207
13. Lee J (1991) A comparison of existing methods for building irregular networks models of terrain from grid digital elevation models. Int J GIS 5:267–286
14. Riedmiller M, Braun H (1993) EA direct adaptive method for faster backpropagation learning: the RPROP algorithm. Proc IEEE Int Conf Neural Netw 1:586–591
15. Gill PE, Murray W, Wright MH (1981) Practical optimization. Academic Press, New York

16. Battiti R (1992) First and second order methods for learning: between steepest descent and Newton's method. Neural Computat 4(2):141–166
17. Gill PE, Murray W (1978) Algorithms for the solution of the nonlinear least-squares problem. SIAM J Numer Anal 15(5): 977–992
18. Ortega JM (1987) Matrix theory. Plenum, New York
19. Renka RJ (1996) TRIPACK: a constrained two-dimensional Delaunay triangulation package. ACM Trans Math Softw 22(1): 1–8
20. Holger RM, Graeme CD (1998) The effect of internal parameters and geometry on the performance of back-propagation neural networks. Environ Model Softw 13(1):193–209
21. Hollis PW, Harper JS, Paulos JJ (1990) The effects of precision constraints in a backpropagation learning network. Neural Computat 2(3):363–373
22. Deng W, Zheng Q, Chen L (2009) Regularized extreme learning machine. In: Proceedings of IEEE symposium on computational intelligence and data mining, pp 389–395
23. Man Z, Lee K, Wang D, Cao Z, Miao C (2011) A new robust training algorithm for a class of single hidden layer neural networks. Neurocomputing 74:2491–2501
24. Lim MH, Gustafson S, Krasnogor N, Ong YS (2009) Editorial to the first issue. Memetic Comput 1:1–2
25. Meuth R, Lim MH, Ong YS, Wunsh DC (2009) A proposition on memes and meta-memes in computing for higher-order learning. Memetic Comput 1(2):85–100
26. Ong YS, Lim MH, Chen X (2010) Memetic computing—an overview. Res Frontier Article IEEE Computat Intell Mag 5(2): 24–36
27. Lim MH, Cao Q, Li JH, Ng WL (2004) Evolvable hardware using context switchable fuzzy inference processor. IEE Proc Comput Digital Tech 151(4):301–311