7th International DAAAM Baltic Conference
"INDUSTRIAL ENGINEERING
22-24 April 2010, Tallinn, Estonia

# AERIAL IMAGERY TERRAIN CLASSIFICATION FOR LONG-RANGE AUTONOMOUS NAVIGATION

Robert Hudjakov, Mart Tamre

**Abstract:** *The paper presents a method of terrain classification and path planning for unmanned ground vehicles. The terrain classification is done on imagery that is acquired from UAV (Unmanned Aerial Vehicle) and is used for UGV (Unmanned Ground Vehicle) path planning thus introducing collaboration capabilities to the system of two. The system complements UGV on-board navigation system by increasing its perception distance and providing long-range path planning capability.*

*Key words: Optical terrain classification, UAV, UGV, path planning, convolutional neural networks.*

## 1. INTRODUCTION

The UAV and UGV are built in Estonia [1,2,3]; current efforts are focused in introducing collaboration capabilities into the system of two. The target is to reduce UGV energy consumption by fusing UGV on-board navigation system with a long-range path planner that relies on overhead imagery.

The perception distance of an UGV is usually limited to visibility range of its on-board sensors; it is seldom over 100 m [4] which is sufficient for obstacle avoidance but is insufficient for long-term path planning. For off-road navigation it is important to know what is behind a bush or a house ahead for cul-de-sac or rough terrain avoidance. Having fresh data about terrain behind horizon helps to reduce time/energy spent on wandering and to avoid potentially dangerous terrain that are hard to detect on time with on-board sensors (ditches and cliffs).

To increase UGV perception distance and overall performance we use aerial overhead imagery provided by UAV for analysing terrain ahead of the UGV and for generating path to target position. We detect a set of features on overhead imagery that should be preferred (such as roads, grass) or avoided (buildings, water) during navigation and feed the acquired information into A* path planner.

Effectiveness of fusing UGV on-board sensor data with overhead data has been previously demonstrated by Silver et al. [5]; in their experiments with overhead data they measured significant increase in UGV average speed and decrease in required human interventions. Their experiments, however, relied heavily on 3D point cloud acquired by LiDAR mounted to UAV but our system must be limited to passive sensors (cameras, gyro, GPS).

Suitability of convolutional neural networks for terrain classification was demonstrated by Sermanet et al. [6,7]; they built a robust UGV navigation system that relied solely on visual data. Their system uses two-level architecture: fast obstacle detection module with perception range of around 5 m and slower "long-range" vision module with perception range around 35m. The short range module was trained to avoid immediate obstacles and long range modules task was to find pathways ahead. Our described method can be seen as extension to this architecture as it is adding

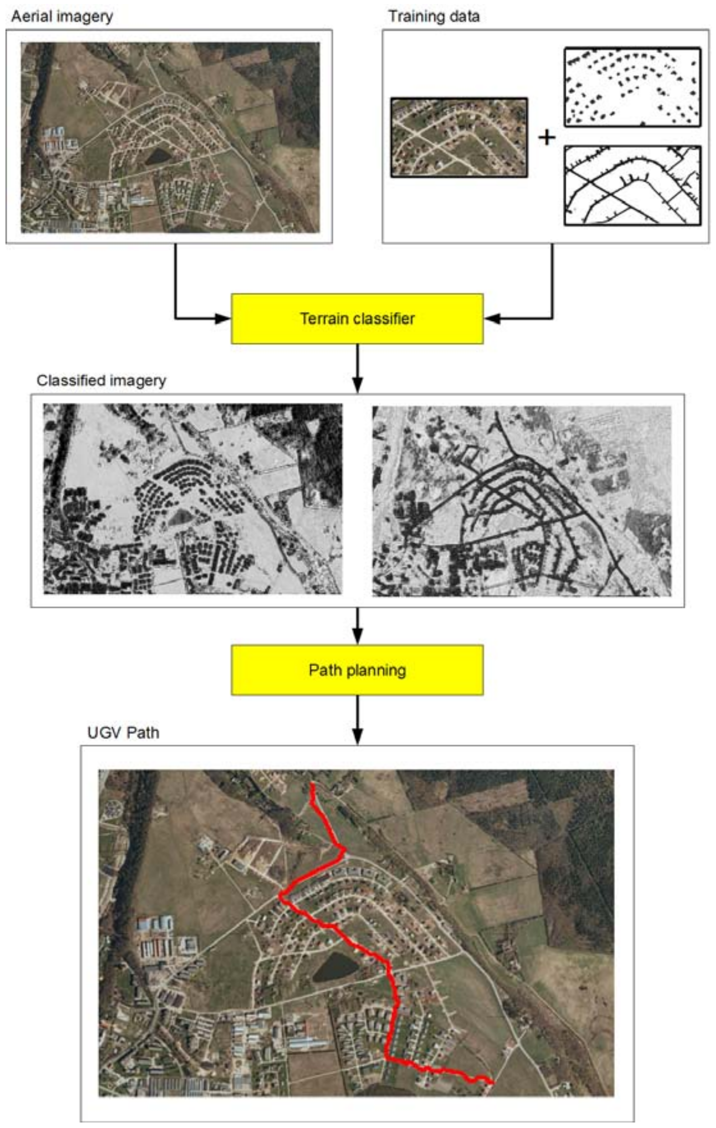

Fig. 1: Terrain classification and path planning

an additional module with even longer perception range but instead of using front facing cameras we rely on overhead imagery.
For aerial imagery terrain classification a multilayer convolutional artificial neural network is used. Based on terrain classification results a cost map is generated and fed into path planner. The path planner will calculate an optimal path to given target position from current UGV location (Fig. 1). The process can be repeated as UGV moves along the path learning about the terrain and as fresh aerial imagery becomes available.

Comparing to our previous article [8] we report significant increase in terrain classification capability of the convolutional neural network as the software that prepared imagery for the network has been improved and have added a path planner.

## 2. TERRAIN CLASSIFICATION

As terrain classifier we are using an artificial neural network (Fig. 2) with three hidden layers; the first two (convolutional) layers are feature extractors and third (fully connected) layer is linear classifier.

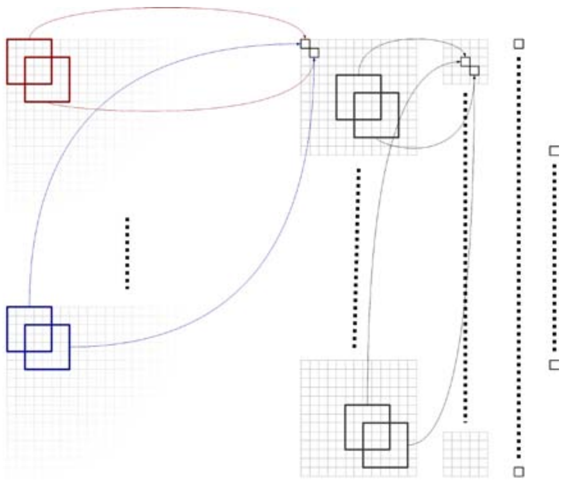

Fig. 2: Network structure. The input layer contains three or four 29x29 neuron 2D maps. Second layer contains six 13x13 and third layer contains fifty 5x5 feature maps. Fourth layer contains 100 fully connected neurons and output layer has an output per feature

The convolutional layers are well suited for feature extraction as they can extract spatially local features [9] and are invariant to shift rotation and scale transformations [9]. In addition the layers are connected so that they extract the features from gradient images instead of color intensity values achieving invariance to lighting conditions. When compared to fully connected layers the convolutional layers have smaller variable space and thus can be trained much faster and with smaller sample set.

The classifier layer is fully connected layer of 100 neurons. In output layer there is an independent output for each trained class/feature. Each output represents likelihood of feature being present in input pattern.

In input layer we have three 29x29 neuron grids; each grid is for a color channel of 29x29 pixel input pattern. Optionally we can accommodate an additional grid for infrared channel when imagery is available. Near-infrared imagery can effectively used to detect vegetation as chlorophyll absorbs red light and reflects in the near-infrared channel.

First hidden layer contains six 13x13 neuron feature maps that are connected to input layer by using 5x5 shared kernels. The weights are shared by all neurons on same feature map that connect to given input grid so there are only (5*5+1)*6*3=468 weights connecting the 29*29*3=2523 neurons in RGB input layer and 13*13*6=1014 neurons in feature maps (including bias connections).

Second hidden layer contains fifty 5x5 feature maps that are connected to previous layer the same way as first layer is connected to input layer. The layer contains 5*5*50=1250 neurons and uses (5*5+1)*50*6=7800 weights to connect them to previous layer.

Third hidden layer is a fully connected linear classifier that contains 100 neurons; each of them is connected to all neurons in previous layer using 100*(1250+1) =125100 weights in total.

Final output layer is also a fully connected layer that contains one neuron per feature category; in case of four categories it contains 4 neurons and 4*(100+1)=404 weights.

The pixel intensity values (in range [0 ... 255]) of input pattern are fed directly into network inputs; the scaling of network inputs into range [-1..1] is left to weights that connect network layers. The scaling is necessary because we use tanh (1.7159 * tanh( 2/3 x) [11] to be precise) function as neuron activation function and we want to keep the neuron input in sigmoid part of the tanh function. To reduce training time we initialize the network weights into random value in range [-0.004 ... +0.004] so the network input (in range [0..255]) multiplied by weight (in range [-0.004 ... +0.004]) will be approximately in range [-1 ...+1].

To shorten time required for training a second order method called "stochastic diagonal Levenberg Marquardt" is used. It should shorten the learning time by threefold without introducing significant computational overhead.

The optimal size (count of feature maps and neurons in classifier) of the network is still to be tested.

## 3. MAP GENERATION AND PATH PLANNING

We define map as grid of nodes; at each node we can predict likelihood of presence for all features. In short our map is grid of confidence vectors. To simplify the path planning task we use a threshold for every feature category (more about it later) to binarize the likelihood.

After the map is binarized (at each node a feature must be either be definitely present or definitely missing) the cost map can be easily generated. A weight factor can be assigned to each feature category depending if the feature is easy to trespass (roads, grass) or hard to trespass (houses, bushes). The easily trespass able feature classes get negative weight and hardly trespass able terrains get positive weight.

To calculate terrain trespass ability cost at each node we calculate a weighted sum and offset it by a positive constant that makes sure that the cost is always positive:

$$\text{cost} = \sum p_i w_i + \text{const} + C_{UGV}, \quad (1)$$

where **p** is the binarized confidence vector and **w** is the weight vector. For areas that are explored by UGV an additional augment $C_{UGV}$ can be added to incorporate UGV data about the terrain.

One thing to note here is that cost for unknown terrains (with no features detected the confidence vector is zero) is equal to the *const* that is higher than cost for easily trespassable terrain but remains lower for hardly trespassable terrain.

For path planning A* algorithm is used. As the map generated from aerial imagery is a 2D rectangular grid it is easy to define base cost of movement from one cell to another as the distance of the two nodes and multiply it by the cost defined earlier. Base cost is 1 for horizontal and vertical movements and sqrt(2) for diagonal movements.

## 4. TEST SETUP

### 4.1 Classification

To verify the performance of the terrain classifier and path planner two different areas were selected and corresponding aerial RGB photographs were loaded from Estonian Land Board database. The Estonian Land Board database was used because of the easy availability of aerial imagery and because the imagery is similar to the imagery acquired by UAV (up to 10 pixels per meter resolution).

The images were hand classified and then scaled down by 2x in order to reduce the scale of manual classification errors. From each image two sets of data were generated by randomly picking 29x29 pixel patterns: training and testing sets for classifier. The training set contains 30000 patterns and testing set 3000 patterns.

Additional care was taken to ensure that any of the patterns would not contain too few pixels from any of features. A pattern was discarded when a feature was presented only by few pixels in pattern: each feature had to be presented by at least 100 pixels or not be presented at all.

The network training was started with randomized network weights and learning rate of 0.1%. The whole testing set was introduced to the network multiple times (passes); the learning rate was multiplied by 0.9 after each pass. For training we used backpropagation algorithm but before each pass the averages of second order derivatives for stochastic diagonal Levenberg Marguardt method were evaluated by using 500 random patterns from training set.

For validating the network prediction capability we found a threshold for each output; when an output value is above the threshold we say that corresponding feature is definitely in input pattern and if it's below we say it's not.

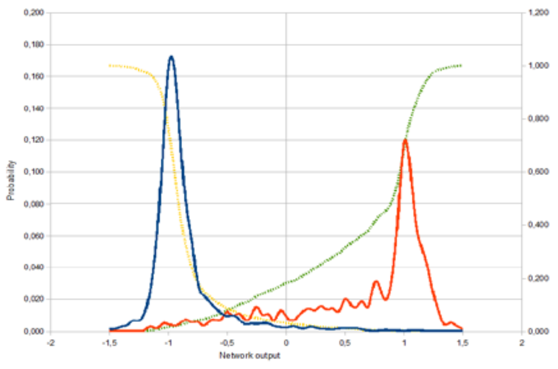

Fig. 3: Network output density functions (when feature is present = red and when feature is not present = blue) and corresponding cumulative distribution functions (dotted lines).

To find the threshold we fed the training set through classifier and for each network output plot out two probability density graphs (Fig. 3): one shows the output value when feature is present in the network input and the other when feature is not present in the network input. Next we plot out cumulative versions of the same graph and take the crossing point of the cumulative graphs as threshold for the output [8].

### 4.2 Map Generation

The easiest way to generate a map would be to divide a selected image to a grid of 29x29 pixel patterns and feed each pattern into the classifer. In order to increase the map resolution it is possible to divide the image so that the patterns overlap. The experiments were done by using 10 pixel steps increasing the map resolution threefold in both directions.

To increase the performance of map creation procedure it is possible to integrate the map generation and path planning procedure by evaluating the confidence vectors as needed by path planner. Each time the path planner needs a cost of unevaluated area the pattern of the area is requested and fed to artificial neural network. This avoids evaluating whole aerial photograph because only the nodes actually used by path planner are evaluated.

## 5. RESULTS

We used two distinct areas for testing that are described in more details in our previous article [8]. One area was from Tallinn outskirts and other was a fen near Tallinn. Since we updated terrain classifier input formatting we have new results to report.

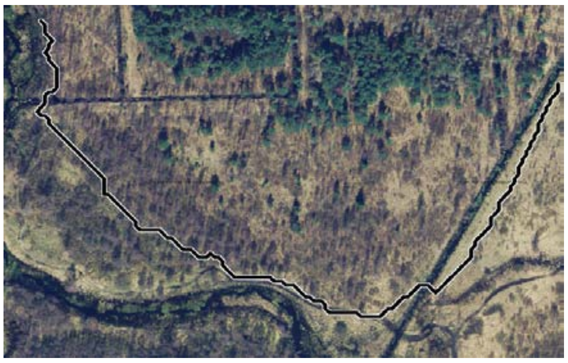

Fig. 4: Path planning through fen 1

In the outskirts area after 21 epochs of training the network output MSE (Mean Squared Error) converged to 0.066. The rates of good classification for houses, roads, grass and debris were 99.73%, 99.80%, 99.57% and 99.20% correspondingly. All features were classified correctly on 98.33% patterns (up from 74.9% from previous experiment).

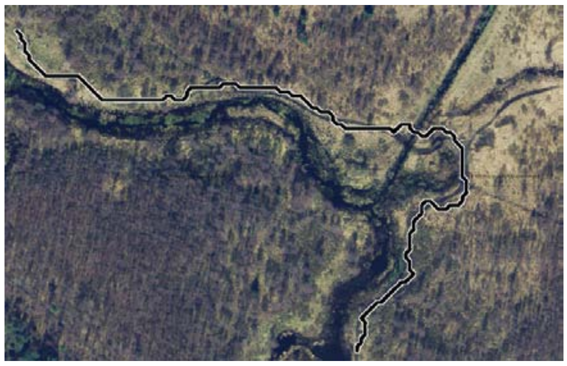

Fig. 5: Path planning through fen 2

In fen area it took 25 epochs of training before the network MSE converged to 0.24 and rates of good classification for water, forest and roads were 92.83%, 99.43% and 79.23% correspondingly. The "road" tracks in fen were full of water and even during hand labelling it was difficult to distinguish

them from creek so the detection rate was lower. All classes were correctly classified on 73.7% of the patterns; up from 55.5% from previous experiment on same area.
The results from classification experiments show that the test made were not exhaustive enough and further testing on higher quality training and testing set is needed.
To verify the path planning capability of the system several experiments were made by using the overhead imagery from Estonian Land Board database. The feature category weights were chosen so that the path planner prefers roads and avoids water, houses, trees and debris (Figs. 4-6).

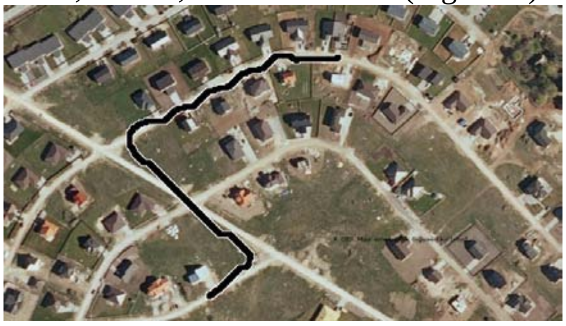
Fig. 6: Path planning in suburban area

## 6. SUMMARY AND FURTHER WORK

The experiments show that our system is capable of extracting distinctive features from aerial data and use them for path planning.
The work is important because estimating and minimizing energy consumption during navigation is vital in mission critical tasks for battery powered UGV´s. Estimating energy consumption allows us to predict if UGV is capable of completing given mission. Our system does not relie on LiDAR data allowing us to use smaller UAVs.
In future we plan to carry out additional experiments to find optimal size for the artificial neural network and to further test the capabilities of the classsifier. We also plan to integrate the system with UGV on-board navigation system and implement real world tests with automatic terrain labelling for network training.

## 7. ACKNOWLEDGMENTS

This work was supported by Doctoral School of Energy and Geotechnology II and Estonian Ministry of Education and Research Project SF0140113Bs08.

## 8. REFERENCES

1. P. Leomar, M. Tamre, T. Vaher, "Estonia Making Steps Towards Joining International UAV Community.", Proc. of Conf. EURO UAV, 2004,11-22
2. Tallinn University of Technology, Department of Mechatronics, "Universal Ground Vehicle", Research project L523, 2005-2008
3. M. Hiiemaa, M. Tamre, "Semi-autonomous Motion Control Layer for UGV-Type Robot", Recent Advances in Mechatronics, 2009,203 - 207
4. Pierre Sermanet, Raia Hadsell, Marco Scoffier, Urs Muller and Yann LeCun, "Mapping and Planning under Uncertainty in Mobile Robots with Long-Range Perception", Proc. Intelligent Robots and Systems , 2008
5. Max Bajracharya, Benyang Tang, Andrew Howard, Michael Turmon, Larry Matthies, "Learning Long-Range Terrain Classification for Autonomous Navigation", IEEE International Conference on Robotics and Automation, 2008
6. P Sermanet, Raia Hadsell, "A Multi-Range Architecture for Collision-Free Off-Road Robot Navigation", Journal of Field Robotics, 2009, 58-87
7. R Hadsell, A Erkan, P Sermanet, "A Multi-Range Vision Strategy For Autonomous Offroad Navigation", Proc. Robotics and Applications , 2007
8. Robert Hudjakov, Mart Tamre, "Aerial Imagery Terrain Classification for Long-Range Autonomous Navigation", Proc. of International Symposium on Optomechatronic , 2008
9. Christopher M. Bishop, "Pattern Recognition and Machine Learning", Springer, 2006