Estonian Journal of Engineering, 2011, **17**, 1, 17–27 doi: 10.3176/eng.2011.1.03

Ortophoto analysis for UGV long-range autonomous navigation

Robert Hudjakov and Mart Tamre

Department of Mechatronics, Tallinn University of Technology, Ehitajate tee 5, 19086 Tallinn, Estonia; robert.hudjakov@gmail.com, mart@staff.ttu.ee

Received 9 September 2010, in revised form 8 November 2010

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) or satellite and is used for UGV (unmanned ground vehicle) path planning thus introducing collaboration capabilities to the system of two. The system complements the 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

Our ultimate target is to build an UGV that is capable of driving independently to given GPS coordinates. 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 and energy spent on wandering and to avoid potentially dangerous terrain that is hard to detect on time with on-board sensors (ditches and cliffs). The perception distance of an UGV is usually limited to visibility range of its on-board sensors; it is rarely over 100 m [1], which is sufficient for obstacle avoidance but is insufficient for long-term path planning.

To increase UGV perception distance and overall performance we use aerial imagery provided by UAV for analysing terrain ahead of the UGV and for generating path to the target position. We detect a set of features on aerial

imagery that should be preferred (such as roads, grass) or avoided (buildings, water) during navigation and feed the acquired information into the path planner.

Effectiveness of fusing UGV on-board sensor data with aerial imagery has been previously demonstrated in [2]; in these experiments with aerial data significant increase in UGV average speed and decrease in required human interventions has been measured. These 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. [3,4]; 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 35 m. The short range module was trained to avoid immediate obstacles and long range module's task was to look ahead and find passages. Our described method can be seen as important addition to this system as it adds an additional module with far longer perception range.

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 a path planner. The path planner will calculate an optimal path to given target position from current UGV location (Fig. 1). As the UGV travels along the path it gathers fresh data about terrain and uses it to retrain the network and update the path. The cycle is repeated every few minutes to keep our system adaptive and capable of learning new terrain.

The UAV and UGV are built in Estonia [5–7]; 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 aerial imagery.

In comparison to our previous paper [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. In addition, we introduce a path planner and demonstrate that the classifier output can be used for navigation.

2. TERRAIN CLASSIFICATION

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

The convolutional layers are well suited for feature extraction as they can extract spatially local features 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 colour intensity values, achieving

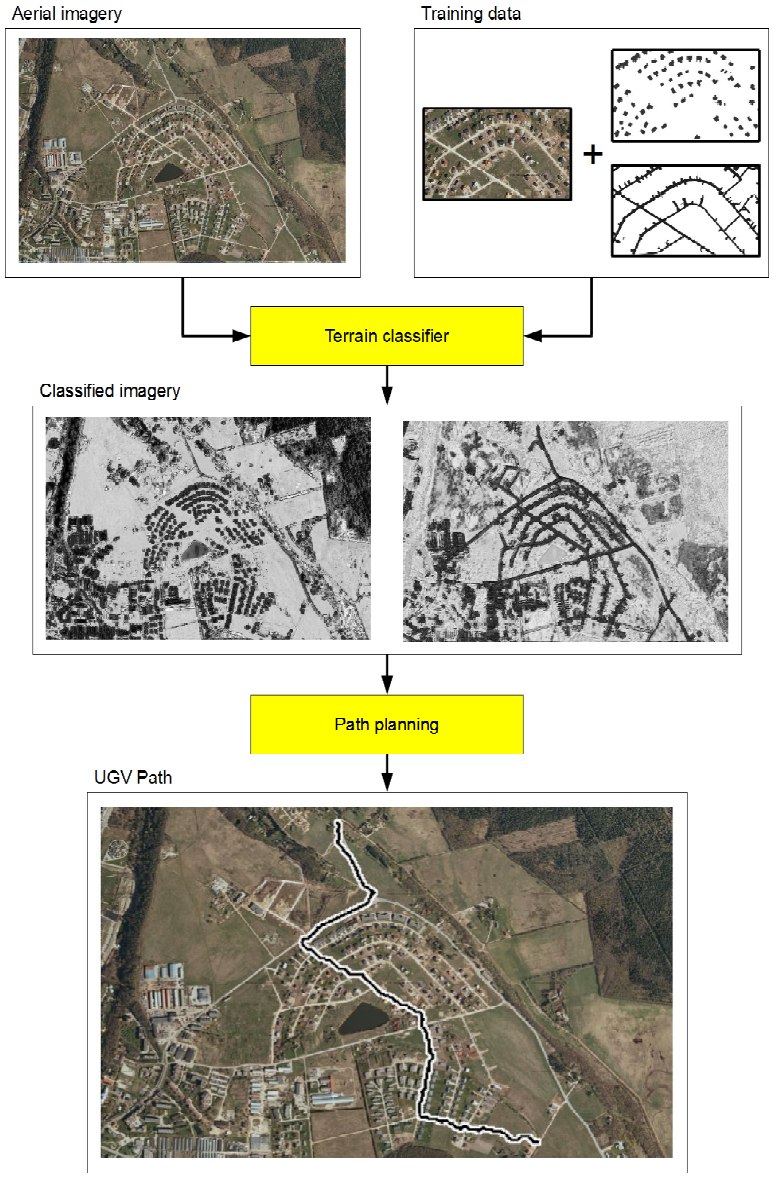

Fig. 1. Terrain classification and path planning.

invariance to lighting conditions [10]. 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 [11].

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

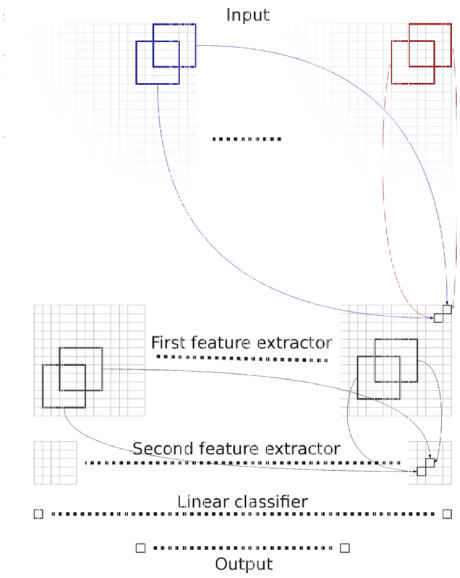

Fig. 2. Network structure.

In the input layer we have three 29×29 neuron grids; each grid is for a color channel of 29×29 pixel input pattern. Optionally we can accommodate an additional grid for infrared channel when imagery is available. Near-infrared imagery can effectively be used to detect vegetation as chlorophyll absorbs red light and reflects in the near-infrared channel [2].

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

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

Third hidden layer is a fully connected linear classifier that contains 100 neurons; each of them is connected to all neurons in the previous layer using $100 \times (1250 + 1) = 125\,100$ weights in total.

The 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 $4 \times (100 + 1) = 404$ weights.

The pixel intensity values (in the range of 0 to 255) of the input pattern are fed directly into network inputs; the scaling of network inputs into the range [–1…1] is left to weights that connect network layers. The scaling is necessary

because we use tanh $(1.7159 \times \tanh(2/3\ x))$ [11] 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 values in the range of -0.004 to $+0.004$, so the network input (in the range of 0 to 255), multiplied by weight (in the range of -0.004 to $+0.004$) will be approximately in the range of -1 to $+1$.

To train the network, the backpropagation method is used, but to shorten time, required for training, a second order method called "stochastic diagonal Levenberg Marquardt" [11] is used. The second order method allows picking different learning rate for each weight in the network and should shorten the learning time threefold without introducing significant computational overhead [11].

We have made a preliminary batch of tests to find optimal network size (the number of feature maps and neurons in the classifier); in short, we observed slight increase in classification capability when we increased the number of feature maps in the first hidden layer and dramatic decrease in training time when we halved the size of the second and third hidden layers. The experiments described in the current article, however, are executed with network configuration that is described above.

3. MAP GENERATION AND PATH PLANNING

We define *map* as a grid of nodes; at each node we can predict likelihood of the presence of all features. In short, our map is a grid of confidence vectors. To simplify the path planning task we use a threshold for every feature category to binarize the likelihood. The threshold is calculated from the training set after the network training cycle is completed (more on that in the following paragraph).

After the map is binarized (at each node a feature must 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 on whether the feature is easy to traverse (roads, grass) or impassable (houses, bushes). The easily traversable feature classes get negative weight and impassable terrains get positive weight.

To calculate terrain traverse 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:

$$\mathrm{cost} = \sum p_i w_i + const + C_{\mathrm{UGV}},$$

where $\mathbf{p}$ is the binarized confidence vector and $\mathbf{w}$ is the weight vector. For areas that are explored by UGV, an additional term 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 then the confidence vector is zero) is equal to the *const* that is higher

than the cost for easily traversable terrain but remains lower for impassable terrain.

For path planning the A* [12] algorithm is used. As the map, generated from aerial imagery, is a 2D rectangular grid, it is easy to define base cost of the movement from one cell to another as the distance of the two nodes, and to multiply it by the cost defined earlier. Base cost is 1 for horizontal and vertical movements and $\sqrt{2}$ for diagonal movements.

The generated path is too rough for short-range navigation; the UGV must rely on onboard sensors to avoid immediate obstacles. The path, however, is detailed enough to direct the UGV to a nearby road or to use pathways in the forest.

4. TEST SET-UP

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 size of one image (suburbs) is 25 Mpx and the size of the other image (marsh) is 12 Mpx. 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 manually classified and then scaled down by 2× in order to reduce the scale of manual classification errors. During manual classification it was difficult to find exact boundaries of features and by scaling the classified image we wanted to reduce the percentage of uncertain pixels in 29×29 input patterns. From each image, two sets of data were generated by randomly picking 29×29 pixel patterns: training and testing sets for the classifier.

The training set contains 30 000 patterns and the testing set 3000 patterns. We initially used 60 000 patterns for testing like Simard et al. [10] but halved the training set for convenience (smaller files, faster training passes). For testing set size we chose 10% of the training set. The patterns are overlapping but we do not see this as a problem because as long as the patterns are not matching they contain different combinations of features and thus are unique inputs for the classifier.

Extra care was taken to ensure that any of the patterns would not contain too few pixels from any of the features. A pattern was discarded when a feature was presented only by few pixels in the 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 training set was introduced to the network for several 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 the training set.

We trained the network outputs to be independent; each network output shows likelihood of the presence of the corresponding feature in the network input pattern. A network output is 1 if corresponding feature is present in the input pattern and –1 if not.

It is also possible to train network outputs to become dependent; in this case the network would classify the pixel that is in the middle of the input pattern; only one network output may be 1, all the others are –1. From preliminary experiments we observed that we lost in classification capability but won in spatial resolution of the generated map. Many of the features that were smeared out in case of independent outputs were pixel precise in case of dependent outputs. It is so because a feature that is represented by a single pixel will be detected on all patterns containing that pixel when independent outputs are used (hence the smearing) but will only be detected on one pattern when dependent outputs are used.

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 the input pattern and if it is below, we say it is not.

To find the threshold, we feed the training set through the 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 the 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].

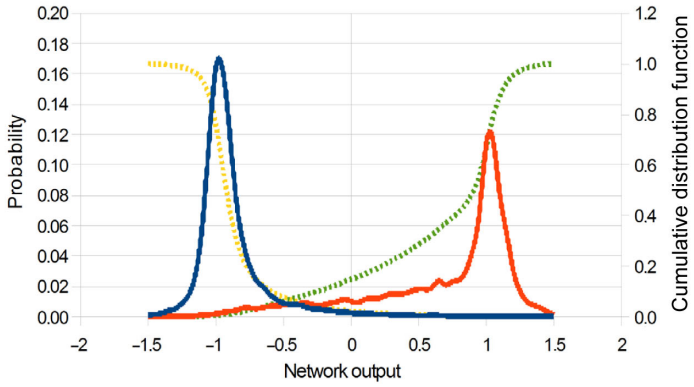

Fig. 3. Network output density functions (red, when feature is present and blue, when feature is not present) and corresponding cumulative distribution functions (dotted lines).

4.2. Map generation

The easiest way to generate a map would be to divide a selected image to a grid of 29×29 pixel patterns and feed each pattern into the classifier. 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 the map creation procedure it is possible to integrate the map generation and path planning procedure by evaluating the confidence vectors as needed by the path planner. Each time the path planner needs the cost of an unevaluated area, the pattern of the area is requested and fed to artificial neural network. This avoids evaluating the 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 detail in our previous article [[8]]. One area was from Tallinn outskirts and the other was a fen near Tallinn. Since we updated the terrain classifier input formatting, we have obtained new results.

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 the previous experiment).

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 the fen were full of water and even during hand labelling it was difficult to distinguish them from drainage canals 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).

To verify the path planning capability of the system, several experiments were made by using the aerial 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 to 6).

6. SUMMARY

The experiments show that our system is capable of extracting distinctive features from aerial data and of using them for path planning. The work is important because estimating and minimizing energy consumption during

Fig. 4. Path planning through fen 1.

Fig. 5. Path planning through fen 2.

Fig. 6. Path planning in suburban area.

navigation is vital in mission critical tasks for battery powered UGVs. Estimating energy consumption allows us to predict if UGV is capable of completing the given mission. Our system does not rely on LiDAR, data allowing us to use smaller UAVs with passive sensors.

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 classifier. We also plan to integrate the system with UGV on-board navigation system and to implement real world tests with automatic terrain labelling for network training.

REFERENCES

1. Sermanet, P., Hadsell, R., Scoffier, M., Muller, U. and LeCun, Y. Mapping and planning under uncertainty in mobile robots with long-range perception. In *Proc. International Conference on Robotics and Automation, IEEE*. Nice, 2008, 2525–2530.
2. Silver, D., Sofman, B., Vandapel, N., Bagnell, J. A. and Stentz, A. Experimental analysis of overhead data processing to support long range navigation. In *Proc. International Conference on Intelligent Robots and Systems, IEEE*. Beijing, 2006, 2443–2450.
3. Sermanet, P., Hadsell, R., Scoffier, M., Grimes, M., Ben, J., Erkan, A., Crudele, C., Miller, U. and LeCun, Y. A multirange architecture for collision-free off-road robot navigation. *J. Field Robotics*, 2009, **26**, 52–87.
4. Hadsell, R., Erkan, A., Sermanet, P., Ben, J., Kavukcuoglu, K., Muller, U. and LeCun, Y. A multi-range vision strategy for autonomous offroad navigation. In *Proc. 13th IASTED International Conference on Robotics and Applications*. ACTA Press, Anaheim, 2007, 457–463.

5. Leomar, P., Tamre, M. and Vaher, T. Estonia making steps towards joining international UAV community. In *Proc. Conference EURO UAV*. Paris, 2004, 11–22.
6. Universal Ground Vehicle, Research Project L523. Tallinn University of Technology, Department of Mechatronics, 2005–2008.
7. Hiiemaa, M. and Tamre, M. Semi-autonomous motion control layer for UGV-type robot. In *Recent Advances in Mechatronics* (Brezina, T. and Jablonski, R., eds). Springer, Berlin, 2009, 203–207.
8. Hudjakov, R. and Tamre, M. Aerial imagery terrain classification for long-range autonomous navigation. In *Proc. International Symposium on Optomechatronics, IEEE*. Istanbul, 2008, 88–91.
9. Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.
10. Simard, P. Y., Steinkraus, D. and Platt, J. C. Best practices for convolutional neural networks applied to visual document analysis. In *Proc. Seventh International Conference on Document Analysis and Recognition*. Edinburgh, 2003, vol. 2, 958.
11. LeCun, Y., Bottou, L., Orr, G. and Müller, K. *Efficient BackProp in Neural Networks: Tricks of the Trade* (Orr, G. and Müller, K., eds.). *Lecture Notes in Computer Science*, **1524**. Springer, Berlin, 1998, 546.
12. Hart, P. E., Nilsson, N. J. and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Sci. Cybernetics SSC4*, 1968, **4**, 100–107.

Ortofotode analüüs, toetamaks autonoomsete maastikuvalmidusega robotite navigeerimist

Robert Hudjakov ja Mart Tamre

On välja töötatud adaptiivne süsteem aerofotode klassifitseerimiseks ja selle põhjal autonoomse roboti teekonna planeerimiseks. Süsteemi treenimiseks saab kasutada nimetatud roboti kogutud infot, tsükliline ületreenimine värskelt kogutud andmetega tagab süsteemi kohanemisvõime muutuvates oludes. Töö võimaldab hinnata ja vähendada nimetatud roboti energiakulu missiooni vältel.