Using Neural Gas Networks in Traffic Navigation

Ján Vaščák

Centre for Intelligent Technologies, Technical University in Košice,
Letná 9, 042 00 Košice, Slovakia,
e-mail: jan.vascak@tuke.sk

Abstract: The quality of navigation methods for mobile means depends first of all on description accuracy of their movement in a given area. This paper deals with Neural Gas (NG) networks whose role is creating of topologies for complex objects as for instance road networks of municipal communications where it is necessary to determine relations among individual elements (in our case a network of mutually interconnected communication nodes), too. The proposed approach combines besides NG networks also tree search algorithms, namely A*, hereby enabling to consider also various restraints, e.g. traffic rules, too. The performance of the proposed algorithm is shown on the road network of the city Košice in Slovakia.

Keywords: navigation, Neural Gas networks, tree search algorithms, algorithm A^*

1. Introduction

At present there are three basic groups of navigation methods for mobile means: heuristic, grid and exact algorithms [1]. A typical representative of the first group there are Bug algorithms. They are simple but suitable only for avoiding only a smaller number of obstacles, mainly in tasks for preventing immediate collisions. Potential fields [15, 18] are the most known approaches of the grid algorithms. However, they require not only creating a potential field of the whole area in advance but in the case of some changes (e.g. a new obstacle) it will be necessary again newly to create the whole field and computational efforts are considerably high. The exact algorithms as visibility graphs or Voronoi diagrams enable to find the best (in our case the shortest) trajectory. If there is no solution they will be even able to terminate the computation and so to prevent timeless sub-cycling. Opposite to the potential fields in the case of changes it will be necessary to do modifications only in the given area. However, their drawback is that they require very accurate data about obstacles that means a serious problem in practical applications.

The NG networks are basically graphs, which enable modelling the form of given patterns, similarly as in Kohonen networks (Self-organizing Maps) but opposite to them NG networks do not have any definite topology of connections in the output layer. This fact seems to be very advantageous mainly in the case of non-homogenous areas. Their ability of accurate description for a given pattern was compared to other kinds of neural networks and confirmed e.g. in [6, 12, 21]. Therefore it seems to be very suitable to utilize them just for the description of road networks with various types and forms of communications. Besides, the use of some modified search algorithms enables us to incorporate into the description various restraints, e.g. traffic rules or traffic density, too. In addition, convenient graph structure of NG networks enables very efficient searching.

This paper deals with description of NG networks principles, followed by description of the traffic networks problem and modification of the A^* algorithm. In concluding parts some experiments will be analysed, which were done on a real road network of the city Košice in Slovakia.

2. Principles of NG Networks

The NG networks are basically derived from Kohonen networks where for the position change determination of output neurons also the neighbourhood principle is used. However, opposite to them the neighbourhood is changed in each adaptation step according to given input [9]. This enhanced adaptation measure enables a relatively free movement of neurons in the area, which should be described just by their suitable deployment and it represents analogy to gas spreading in a closed space. Learning of NG networks mutually combines two learning paradigms – *own NG learning* and *competitive Hebbian learning*.

Let us suppose that we have a predetermined number of points N with help of them we can describe a given space. They will be denoted as *reference vectors* w whose elements are in general space coordinates – in our case they will be two-dimensional. Reference vectors w divide the area (space) into N parts and in each of them they represent centres of these parts. Such an approach is known as *vector quantization* that represents certain form of coding with help of which we can describe given area and which is the own task of NG networks. The success of this task is dependent just on suitable deployment of reference vectors. Therefore, similarly as in Kohonen networks, in individual cycles we will select points ξ belonging to this area and with their help we will adapt positions of reference vectors, which define positions of neurons c_i of the network output layer A, i.e. $A=\{c_1, c_2, \ldots, c_N\}$.

The object of the *own learning* is adaptation of reference vectors, i.e. calculation of their change $\Delta w_i(t)$ in the time t. It is a kind of the multiple-winners learning where the most change is at the winner vector w_{s1} but in accordance with the *neighbourhood range* $h(t,k)$ also other k neighbours are changed although in a lower measure. The vector w_{s1} is assigned to such a point, which is the nearest to the input ξ, i.e. arg(min($\xi - w_{S1}$)). The ambition is to reach bigger changes in a broader range at the beginning of the learning process and continuing with time this trend would decrease till the adaptation end time t_{max}. From this reason parameters λ_p and λ_z are defined – starting and final where λ_p□λ_z. The parameter λ for the given time t is defined as:

$$\lambda(t) = \lambda_p (\lambda_z / \lambda_p)^{t/t_{\max}} \tag{1}$$

and the *neighbourhood range* $h(t, k)$ for the k^{th} neighbour is computed as:

$$h(t,k) = \exp\left(\frac{1-k}{\lambda(t)}\right). \tag{2}$$

The adaptation process is also influenced by the learning parameter γ whose value is time-dependent according to γ_p and γ_z similarly as for λ and the calculation is like in (1).

The entire adaptation process in individual adaptation steps till t_{max} is following:

1. Sequencing all reference vectors by their distance to ξ into series:

$$\|\xi - w_{S1}\| \leq \|\xi - w_{S2}\| \leq \ldots \leq \|\xi - w_{SN}\|. \tag{3}$$

2. Adapting all reference vectors by:

$$\Delta w_{Si}(t) = y(t) \cdot h(t,k) \cdot (\xi - w_{Si}). \tag{4}$$

3. Setting up the time t = t+1 and until $t < t_{max}$ return to the step 1 else finish the adaptation.

It is possible to prove that this kind of adaptation principally corresponds to the optimization of a cost function according to the gradient descent method [10], which does not exist in Kohonen networks. In other words, the adaptation in NG networks is usually quicker.

The *competitive Hebbian learning* [9] serves for constructing a topological structure among neurons of the output layer. This kind of learning comes from the basic idea of Hebbian learning, i.e. that the connections whose neurons are activated at the same time (synchronously) are strengthened. Their change corresponds to the product of the activation values of these neurons. However, at the same time a competition element is embedded, too. This means in one adaptation step only one connection is created, namely between the two closest network points to the input ξ, i.e. between the reference vectors w_{s1} and w_{s2} of the points *S1* and *S2* (neurons c_1 and c_2). The distance among points is usually calculated by Euclidian norm. In [11] it was shown that the topology created in such a manner corresponds to Delaunay triangulation. Consequently, after transforming to Voronoi regions, these regions ensure finding an optimal path.

The simplest way for learning NG networks is using a two-stage process where at first suitable deployment of a given number of points is created and then they are mutually interconnected using competitive Hebbian learning. However, this approach is possible only in the case of predefined t_{max}, which is a considerably limiting circumstance. The other possibility is based on parallel processing of both learning kinds. However, there is a danger that Delaunay triangulation will be damaged due to continued changes of reference vectors. From this reason it is necessary to incorporate a mechanism of removing obsolete connections, too. For this purpose a process of *aging connections* is used where an age $v(c_1, c_2)$ is assigned to each connection between the neurons c_1 and c_2. At the moment of creating a new connection its age is set up to zero as well as in the case if the algorithm tries again to create this connection (the so-called *connection rejuvenating*). The age will be incremented by 1 in all connections among all direct neighbours of c_1 if it becomes again the winner. If the age of a connection reaches the value $T(t)$ it will be removed. In such a way a risk of invalid connections will be minimized. As at the adaptation start bigger changes are done it is necessary for $T(t)$ to be changed in time from smaller to bigger values. It is determined in a similar way as in (1) where T_p □ T_z.

A learning process by adaptation steps for obtaining a description of a ring (grey surface) [4] is depicted in Fig. 1. It is possible to observe the influence of parameters ($\lambda_p = 10$, $\lambda_z = 0.01$, $\gamma_p = 0.5$, $\gamma_z = 0.005$, $t_{max} = 40000$, $T_p = 20$, $T_z = 200$) with advancing time, too. From intelligibility reasons in the depictions b – e the connections among neurons are missing.

In the first steps of learning a large neighborhood of the input ξ contains adapted points, which is characterised by a massive deployment of a large number of points in a form similar to the given space. Later, the influence of adaptation parameters *h(t, k)* and $\gamma(t)$ will be weakened and thereby the adaptation will be performed only in close surroundings of the input ξ, i.e. on a local level individual reference vectors try to cover the given space uniformly. Such a phenomenon is typical for various forms of described space [3].

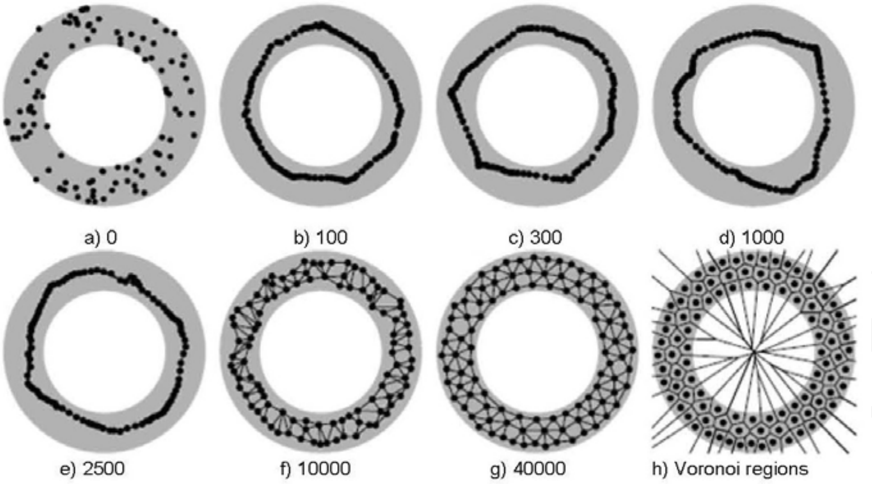
a) 0 b) 100 c) 300 d) 1000
e) 2500 f) 10000 g) 40000 h) Voronoi regions

Figure 1. Learning process of a NG network with competitive Hebbian learning on a ring indicating the number of steps

2.1. Growing NG Networks

A considerable limitation of the previous approach is based on a fact that it is necessary to determine a fix number of output neurons in advance whose estimation can be difficult (if at all possible). Besides, further required property could be embedding a quality criterion directly as an ending condition for learning. These requirements led to a design of modified NG networks with incremental learning named as *Growing NG (GNG) networks* [2] using already mentioned algorithms.

A GNG network starts learning with two starting neurons. For growing their number as well as their deployment it uses heuristics and the so-called *quantization error* E_c of individual neurons c_i where squared errors of the vector w_s relating to the input ξ will be accumulated, i.e.:

$$E_c = \sum \left\| \xi - w_S \right\|^2 \tag{5}$$

if this neuron is a winner, i.e. a neuron with the biggest error (further point *S1*). The learning task is to perform the space quantization in such a way to obtain the minimum total quantization error *E*, i.e. $\sum E_c$ for $i = 1, \ldots, N$. From experimental experience (heuristics) it can be mostly supposed that inserting further neurons will reduce the error *E* and the best result will be obtained if the new point *r* is 'close' to *S1*. For 'closeness' determination there is another heuristic based on using point *S2*, which is a direct neighbour having the biggest E_c comparing to another ones. A new reference vector w_r will be then placed between *S1* and *S2*:

$$w_r = \left(w_{S1} + w_{S2} \right) / 2. \tag{6}$$

Consequently, the connection between *S1* and *S2* will be removed and new two ones will be created between *S1* and *r* as well as *S2* and *r*. Simultaneously, the errors of points *S1* and *S2* will be reduced by values $\alpha.E_{c1}$ and $\alpha.E_{c2}$ where α is the quantization error reduction parameter. As it is clear the point *r* will not fully reduce the total quantization error analogically to (6) a certain amount of starting error will be assigned to *r*, i.e. $E_r = (E_{c1} + E_{c2})/2$.

Unlike NG networks only the reference vectors of *S1* and its direct neighbours are adapted. We can consider only local adaptation with uniform growing of output neurons as it is visible in Fig. 2 and it seems to be also a certain advantage from the point of view of possible simpler learning process analysis. As the adaptation of reference vectors is an iterative process inserting new points will be done only in adaptation steps being a natural product of the parameter τ. Further differences to NG networks are timely constant learning parameters for *S1* and its direct neighbours S_i, i.e. γ_{S1} and γ_{Si} as well as the maximum age of a connection *T*. It is supposed in each adaptation step certain improvement will occur and therefore the quantization error E_{ci} will be always reduced by the value $\beta.E_{ci}$.

The whole learning process of GNG networks is as follows:

1. During initialization two starting neurons are selected arbitrarily. The connection set is empty.
2. An input signal ξ enters the system and the point *S1* with the biggest error E_{c1} from all $c_i \in A$ will be obtained. Consequently, the point *S2* as a direct neighbour of *S1* will be determined. The points *S1* and *S2* will be connected and the connection age will be set up to zero. If the connection already exists then its age will be set up again to zero.

For c_1 the equation (5) will be used and reference vectors for *S1* as well as its direct neighbours will be adapted by:

$$\Delta w_{Si} = \gamma_i \cdot \left(\xi - w_{Si} \right) \tag{7}$$

for $i = 1, 2, \ldots$ and the age of their connections will be incremented.

3. If a connection reaches the age *T* it will be removed as well as all points without any connections.

4. If the adaptation step reaches a natural product of τ (if no then next step 6) a new point r (6) will be inserted and connections and errors for $S1$, $S2$ and r will be modified.
5. For each neuron c_i the quantization error E_{ci} is reduced by the value $\beta.E_{ci}$.
6. If the ending condition is not yet fulfilled (e.g. maximum network dimension or minimum error) then continue next adaptation step and go to the step 2.

In Fig. 2 there is depicted learning process of a GNG network by individual adaptation steps again on the same example of a ring (see Fig. 1) [4] ($\tau = 300$, $\gamma_{S1} = 0.05$, $\gamma_{Si} = 0.0006$, $\alpha = 0.5$, $\beta = 0.0005$, $T = 88$, $N = 100$). Comparing Fig. 1 and Fig. 2 it is possible to see differences of learning NG and GNG networks. However, the obtained results are almost identical (see Fig. h).

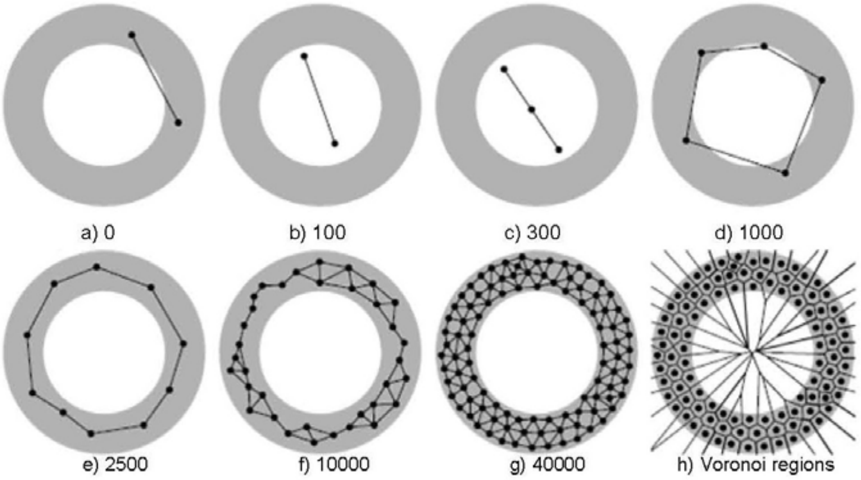

Figure 2. Learning process of a GNG network on a ring indicating the number of steps

3. Utilization and Modification of NG Networks for Purposes of Path Planning

Still nowadays data for traffic navigation systems are created by a complicated way with considerable portion of manual work either in preparing maps or directly on place in measuring orientation points using GPS. In our case it is possible to almost fully automate this preparation stage (besides inserting data about one-way roads and other entry restrictions). Using a colour filter only communications are extracted from the map (Fig. 3) and this kind of information will be a direct basis (training set) for learning. For our purposes we used a GNG network and experiments were done on the map of city Košice.

In Fig. 3 there is a part of a primarily learned network (blue colour) together with the real state of communications (black colour). We can see that this network partially:

- connects communications without any real connection, which is wrong,

- is redundant, i.e. it contains too many points, which can be omitted and thereby we can get a simpler network structure.

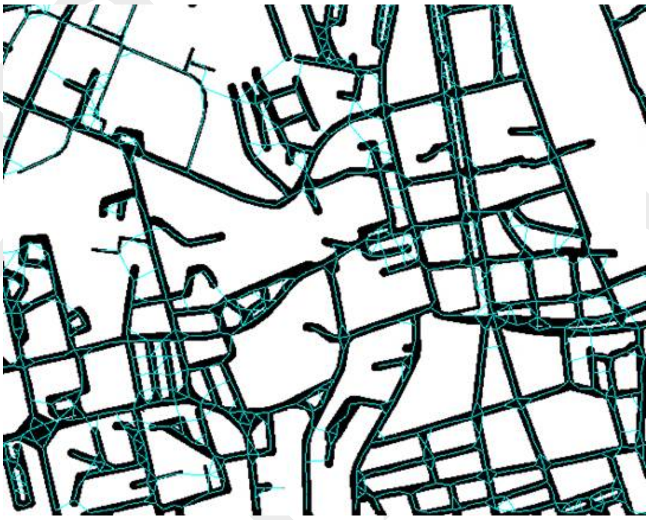

Figure 3. A part of a primarily learned GNG network

During removing wrong connections it is necessary to distinguish between really incorrect connections as e.g. the connections *a, b* in Fig. 4 and principally acceptable connections *c, d*. Acceptable connections represent a principal existence of a road only they are not able accurately to describe its form. From this reason additional points will be inserted better to form it. For separating these two cases a new modification of the Bug2 algorithm [8] was proposed [16] where a search oval is created with the radius ρ covering the investigated connection (see Fig. 4). If in this oval a continuous connection exists between the end points of the investigated connection then the connection is acceptable else it will be removed.

Further step is removing redundant connections. In Fig. 3 it can be seen straightforward roads are described not by only one connection but by several shorter connections, too. All intermediate connections are dispensable because they can be substituted by a longer one, which causes lower memory efforts and shortening the path search. Usually, using reduction mechanisms a considerably simpler network is obtained.

Since, in a traffic network there are both bidirectional and one-way roads the network from Fig. 3 will be doubled and connections will be given orientations. Consequently, for one-way communications (including rotaries) the prohibited directions will be removed. Only this stage requires manual activity. Finally, the connections will be given the information about their length, which is in other words the path cost. The last two kinds of information, i.e. orientation and path cost are fundamental for path search algorithms.

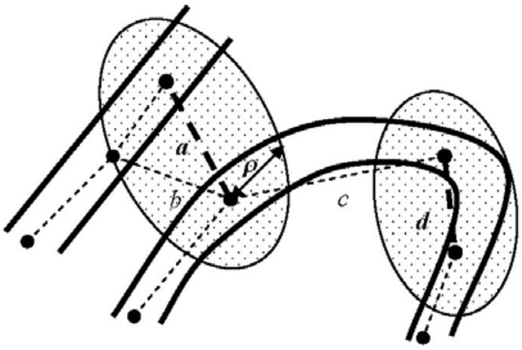

Figure 4. Identification of wrong connections

Creating a description of the road network in the form of a tree structure using NG networks represents only the first stage in the navigation task. Next stage is preformed by tree search algorithms utilizing the structure of a NG network as a convenient data source where they solve the task to find the best path between two points. For this reason the A^* algorithm was used being able to find the shortest path with a minimum number of browsing and in the case of its absence also being able to give a message [8].

However, it is necessary still to take some modifications of the A^* algorithm [19] to keep traffic rules, namely:

- prohibition of turning in a node,
- the so-called P-problem.

Since in the bidirectional communications there is each connection doubled with reverse orientations this would enable turning in next node (point), which means turning in a road. Therefore, being used the current connection its reverse orientation is temporarily cancelled.

As seen from Fig. 5 the so-called P-problem resembles to the character P. There is a problem of inability to find a path although it exists indeed. Let us suppose a car is on the connection between the points 1 and 2 and tries again to come back to the point 1. In such a case the algorithm will stop although there is a solution in the form 2→3→4→5→2→1 (principles of A^* as well as its modifications are more detailed described in [20]). To prevent this problem the coding of the whole NG network was changed in such a way the path will not be searched by the points but by their connections [19]. In other words, neurons of the NG network will not represent communication points (crossings and curves) but connections between these points. In our case for Fig. 5 the solution will look like 23→34→45→52→21.

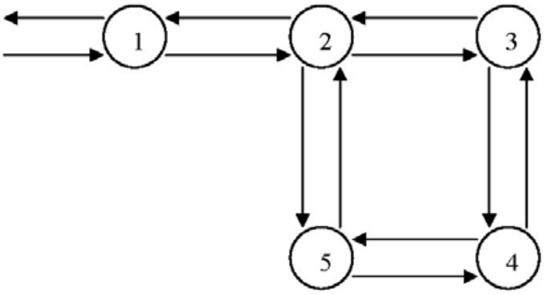

Figure 5. Description of the P-problem

4. Setting up Learning Parameters and Experiments

There are in total eight parameters whose setting up influences not only the learning process but also the entire quality of the resulting space description. In following we will summarize them and introduce some remarks concluded from experiments. Namely there are these parameters for GNG networks:

- number of adaptation steps given by the time t_{max},
- maximum number of neurons (points) of the output layer N,
- interval of inserting new points τ,
- maximum age of a connection T,
- learning parameters γ_{S1} and γ_{Si},
- parameters for reduction of the quantization error α and β.

By [7] the value for γ_{S1} is chosen considerably smaller than $0,3$ – in our case $\gamma_{S1} = 0,05$ and for γ_{Si} the value will be approximately one tenth, i.e. $\gamma_{Si} = 0,006$. Further, for our experiments values of remaining parameters were: $T = 100$, $\alpha = 0,5$, $\beta = 0,0005$.

The experiments were mainly focused on observing parameters t_{max}, N and τ because just these parameters influence most the quality of the created network and interact mutually. Combining their various values and comparing obtained results following outcomes can be confirmed:

1. The optimum number of output neurons is approximately one hundredth of the number of training points.
2. The number of adaptation steps should be from two up to three times bigger as the number of training points.

The interval of inserting new points should enable inserting all points during the first 2/3 of the adaptation (having enough time for deploying new points to a correct position), i.e.:

$$\tau \approx \frac{2t_{\max}}{3N}. \qquad (8)$$

During the experiments a roughly uniform deployment of training points was supposed. The bigger number of adaptation steps the more accurate the network but also the bigger computational efforts. Therefore, the proposed values for parameters express a compromise between descriptive precision and computational speed. Although the greatest influence on the position of a given neuron is caused by the initial step of inserting (6) but it will be influenced also by continuous adaptation of its position even if relating to the magnitudes of γ_{S1} and γ_{Si} in a considerably smaller measure. Therefore, it is necessary to enable a multiple adaptation of each neuron and from this reason t_{max} needs to have high values as well as an inserted neuron needs to have possibility of moderate position correction at least during the last third of the entire adaptation time t_{max}.

For needs of creating a GNG network describing the communication network of the city Košice a training set with *242 470* points was used and other parameters owned these values: $t_{max} = 900\,000$, $N = 2\,000$, $\tau = 300$, $T = 100$, $\alpha = 0{,}5$, $\beta = 0{,}0005$, $\gamma_{S1} = 0{,}05$, $\gamma_{Si} = 0{,}006$. The experiments showed the network was able to learn with the same quality on various types of road networks regardless the form and density as seen in the Fig. 6.

5. Conclusions

The proposed combination of NG networks and exact graph algorithms offers very advantageous properties for navigation either as an auxiliary for drivers [17] or as a direct means for navigation of mobile robots with the ability incrementally to modify space description. The absence of any definite topology of connections in the output layer (in comparison to Kohonen networks) enables NG networks to model whatever area of arbitrary complexity without any limitations regarding various restrains, e.g. forms of roads and traffic rules in the case of communications. This approach enables including further mechanisms like automatic removing of connections in the case of one-way roads or merging several independently created networks describing neighbourhood areas. It is possible to create an overview network with a less detailed description (like maps with different scales) for purposes of approximate navigation [13], too. Since the concept of NG networks is general for spaces with arbitrary dimensions it is also possible to incorporate for instance height data.

The advantage of this approach is based mainly on its two-stage processing. In the first stage a descriptive network is created although computationally demanding but necessary only ones, which will be later modified only occasionally using mentioned mechanisms. Anyway, also in this stage most of activities are automated (opposite to conventional approaches). In the second stage, which will be used many times, highly efficient algorithms are used for finding optimum paths whose time efforts do not exceed 3 seconds in the case of Košice.

The descriptive form of NG networks for contour manifold and heterogeneous spaces offers further utilization possibilities. It would be probably very perspective to use such a numerical knowledge representation form for knowledge extraction into rules using fuzzy logic (because of its nonlinear matter [14]) and its learning (adaptation) approaches [5] to obtain symbolic form of knowledge, which is necessary for more complex control and decision tasks.

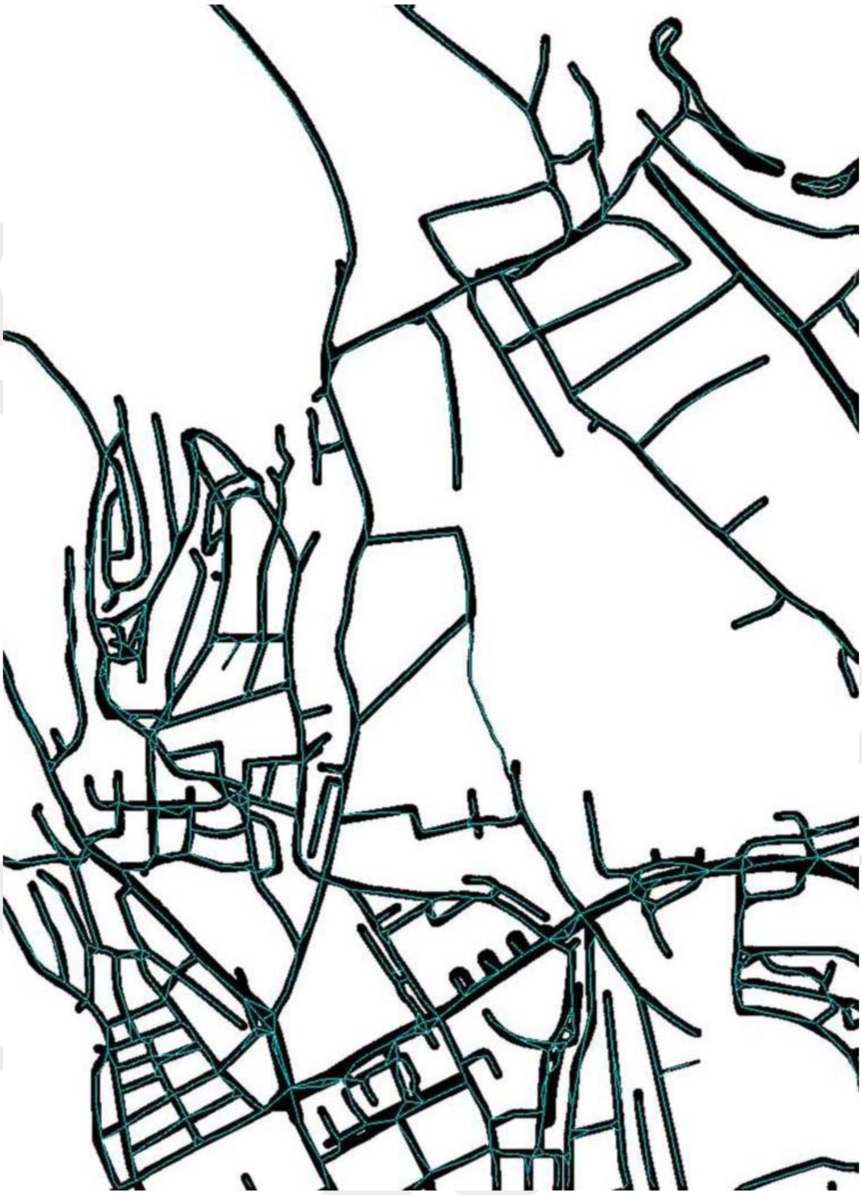

Figure 6. Description of the road network for urban part Košice – North

Acknowledgement

This publication is the result of the project implementation Centre of Information and Communication Technologies for Knowledge Systems (project number: 26220120020) supported by the Research & Development Operational Programme funded by the ERDF.

References

[1] Dudek, G., Jenkin M.: *Computational Principles of Mobile Robotics*, Cambridge University Press, Cambridge, (2000).

[2] Fritzke, B.: *A growing neural gas network learns topologies*, In: Advances in Neural Information Processing Systems 7; MIT Press Cambridge, USA, (1995), pp. 625-632.

[3] Fritzke, B.: *Some competitive learning methods*, (1997), pp. 45, <citeseer.ist.psu.edu/fritzke97some.html> [cit. 22.7. 2008].

[4] Fritzke, B.: *Vektorbasierte Neuronale Netze* (Prof. thesis in German), Shaker Verlag, (1998), pp. 157, <http://www.ki.inf.tu-dresden.de/~fritzke/> [cit. 22.7. 2008].

[5] Heinke, D., Hamker F.H.: *Comparing Neural Networks: A Benchmark on Growing Neural Gas*, *Growing Cell Structures, and Fuzzy ARTMAP*, IEEE Transactions on Neural Networks, Vol. 9, No. 6, (1998), pp. 1279-1291. [6]

[6] Holmström, J.: *Growing Neural Gas* (PhD. thesis), Uppsala University - Department of Information Technology, (2002), pp. 42. [7]

[7] Johanyák, Z. C., Kovács, Sz.: *A brief survey and comparison on various interpolation based fuzzy reasoning methods,* Acta Polytechnica Hungarica, Vol. 3, No. 1, (2006), pp. 91-105. [5]

[8] Lavalle, S. M.: *Planning algorithms*, Cambridge University, (2006), pp. 842, http://planning.cs.uiuc.edu/ [cit. 22.7. 2008].

[9] Martinetz, T. M., Schulte, K. J.: *A neural gas network learns topologies,* In: Artificial Neural Networks; North Holland Amsterdam, (1991), pp. 397-402.

[10] Martinetz, T. M.: *Competitive Hebbian learning rule forms perfectly topology preserving maps*, In: ICANN - International Conference on Artificial Neural Networks, Amsterdam, Springer, (1993), pp. 427-434. [11]

[11] Martinetz, T. M.: Selbstorganisierende neuronale Netzwerkmodelle zur Bewegungssteuerung (in German), Infix Verlag, (1992). [10]

[12] Milano, M., Koumoutsakos, P., Schmidhuber, J.: *Self-Organizing Nets for Optimization*, IEEE Transactions on Neural Networks, Vol. 15, No. 3, (2004), pp. 758-765.

[13] Mls. K.: *Implicit knowledge in Concept Maps and their Revealing by Spatial Analysis of Hand-Drawn Maps*, Proc. of the Second International Conference on Concept Mapping – Concept Maps: Theory, Methodology, Technology, Vol. 2, San José, Costa Rica, (2006).

[14] Oblak, S., Škrjanc, I., Blažič, S.: *If approximating nonlinear areas, then consider fuzzy systems,* IEEE Potentials, Vol. 25, No. 6, (2006), pp. 18-23.

[15] Pozna, C., Troester, F., Precup, R.-E., Tar, J. K., Preitl, St.: *On the Design of an Obstacle Avoiding Trajectory: Method and Simulation*, Mathematics and Computers in Simulation, Elsevier Science, Vol. 79, No. 7, (2009), pp. 2211-2226.

[16] Rutrich, M.: *Využitie sietí typu Neural Gas v navigácii* (MSc. thesis in Slovak), TU v Košiciach, (2007), pp. 67.

[17] Spalek, J., Dobrucký, B., Lusková, M., Pirník, R.: *Modeling of Traffic Flows of Suburban Agglomerations: Technical and Social Aspects, Applications*, The 2nd International Conference on Knowledge Generation, Communication and Management: KGCM 2008 ORLANDO, (2008).

[18] Vaščák, J., Rutrich, M.: *Path Planning in Dynamic Environment Using Fuzzy Cognitive Maps*, In: SAMI – 6th International Symposium on Applied Machine Intelligence and Informatics, Herľany, Slovakia, January 21-22 (2008), pp. 5-9. [19]

[19] Vaščák, J., Szászi, T.: *Navigácia mobilných robotov pomocou harmonických potenciálových polí (1) and (2)* (in Slovak), In: AT&P Journal, No. 2 and 3, (2007) pp. 58-60 and 74-75. [18]

[20] Vaščák, J.: *Fuzzy cognitive maps in path planning*, In: Acta Technica Jaurinensis: Series intelligentia computatorica, Vol. 1, No. 3, (2008), pp. 467-479.

[21] Yeh, M. F., Chang, K. Ch.: *A Self-Organizing CMAC Network With Gray Credit Assignment,* IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 36, No. 3, (2006), pp. 623-635.