**REPÜLÉSTUDOMÁNYI KÖZLEMÉNYEK**

32. évfolyam (2020) 2. szám 137–153. • DOI: 10.32560/rk.2020.2.10

Kővári Bálint,[1] Kolat Máté,[2] Haraszti Sándor,[3] Gyenis Tamás,[4] Vörös Dániel,[5] Rohács József,[6] Bicsák György[7]

# Area Scanning with Reinforcement Learning and MCTS in Smart City Applications

*This research project is focused on area scanning in the scope of smart city applications with unmanned aerial vehicles or UAVs. More powerful devices have been designed in terms of range, capacity and sensory capabilities in the recent years. This makes possible easier automation, thus suppressing the need for human resources. Some of the fields of applications include: traffic or pollution monitoring, land surveying, civil security control or natural disaster control and monitoring. With the increased number of UAV applications, the use and development of efficient algorithms is more and more essential. This paper investigates the possibility of using Monte-Carlo Tree Search (MCTS) and Reinforcement Learning (RL) in this area, which are already successful methods in other control tasks.*

**Keywords:** *area scanning, machine learning, Monte-Carlo method, smart city*

[1] MSc Student, Budapest University of Technology and Economics Faculty of Transportation Engineering and Vehicle Engineering, Department of Control for Transportation and Vehicle Systems, e-mail: kovari.balint1995@gmail.com, ORCID: https://orcid.org/0000-0003-2178-2921

[2] MSc Student, Budapest University of Technology and Economics Faculty of Transportation Engineering and Vehicle Engineering, Department of Control for Transportation and Vehicle Systems, e-mail: komate1995@gmail.com, ORCID: https://orcid.org/0000-0002-8434-6169

[3] MSc Student, Budapest University of Technology and Economics Faculty of Transportation Engineering and Vehicle Engineering, Department of Aeronautics, Naval Architecture and Railway Vehicles, e-mail: hs4ny1@gmail.com, ORCID: https://orcid.org/0000-0001-9458-4446

[4] MSc Student, Budapest University of Technology and Economics Faculty of Transportation Engineering and Vehicle Engineering, Department of Control for Transportation and Vehicle Systems, e-mail: tamgyen@gmail.com

[5] MSc Student, Budapest University of Technology and Economics Faculty of Transportation Engineering and Vehicle Engineering, Department of Control for Transportation and Vehicle Systems, e-mail: voros.dani@hotmail.com

[6] Professor, Budapest University of Technology and Economics Faculty of Transportation Engineering and Vehicle Engineering, Department of Aeronautics, Naval Architecture and Railway Vehicles, e-mail: jrohacs@vrht.bme.hu, ORCID: https://orcid.org/0000-0002-4607-9063

[7] Senior lecturer, Budapest University of Technology and Economics Faculty of Transportation Engineering and Vehicle Engineering, Department of Aeronautics, Naval Architecture and Railway Vehicles, e-mail: gybicsak@vrht.bme.hu, ORCID: https://orcid.org/0000-0002-3427-3918

## Területszkennelés megerősítéses tanulás és MCTS segítségével Smart city applikációkban

*Jelen kutatási projekt célja egy olyan területfeltérképező algoritmus kifejlesztése, amely jól használható okosvárosokban pilóta nélküli légi járműveket (UAV), azaz drónokat alkalmazva. Az elmúlt évek technológiai fejlődésének következtében olyan eszközöket fejlesztenek ki, amelyek egyre nagyobb hatótávolsággal, kapacitással és egyre jobb szenzoros képességekkel rendelkeznek. Ennek köszönhetően az automatizálás sokkal könnyebbé vált, és kevesebb szükség van humán erőforrásokra. Egyre több területen alkalmaznak drónokat például forgalom és környezetszennyezés felügyelete, földmérés, közbiztonság fenntartása, természeti katasztrófák kezelése. Mivel jelentősen nő a területek száma, ahol ezeket a technológiákat alkalmazzák, nagy igény van az ehhez szükséges algoritmusok fejlesztésére. Ez a cikk a Monte-Carlo Tree Search, illetve megerősítéses tanulás ezen területen való alkalmazhatóságát kutatja, amely algoritmusok más területeken már sikeresnek bizonyultak.*

**Kulcsszavak:** *területszkennelés, gépi tanulás, Monte-Carlo-módszer, Smart city*

## 1. Introduction

A smart city is defined as a city that engages its citizens and connects its infrastructure electronically. The goal of building a smart city is to improve the quality of life by using technology to improve the efficiency of services and meet residents' needs.[8] The interest in smart cities is increasing day by day, especially after the global financial recession. The world population is increasing and it is foreseeable that the population of the cities is going to be doubled by 2050.[9] Consequently, these expectations create new challenges and opportunities for cities and communities. Therefore, there is increased interest to focus on utilising information and communication technology (ICT) services and smart solutions in long-term smart city development.[10] Design of such a smart city requires huge and full integration of ICT and its trends. UAVs contribute to these goals. That is why UAVs are involved in a wide range of applications and functions in smart cities. These applications range from monitoring traffic flow to measuring and detecting floods and natural disasters by using wireless sensors. This development is based on technical reports published by technical institutions. Furthermore, many opportunities for UAVs and their applications in smart cities will continue to increase at a fast pace.[11]

A report prepared by McKinsey and Company shows that worldwide expenditure on construction and infrastructure is about 2 trillion US Dollars per year, and ICT expenditure is about 1.5 to 2% of that number. However, during the coming decade, the advances are expected to continue in areas of cloud computing, wireless sensors, networked unmanned systems, big

[8] Sam Musa, 'Smart City Roadmap,' 2016.

[9] Ibid.

[10] R. Shreih, 'Intelligent Systems for Smarter Cities,' King Abdullah University of Science & Technology, 26 August 2013.

[11] 'Smart Cities,' King Abdullah University of Science & Technology, 2013.

data, open data, and Internet of Things. In addition, billions of devices are going to be connected. Consequently, there will be a substantial opportunity for using UAVs in Smart Cities.[12]

UAVs are more powerful in terms of range, capacity and sensory capabilities. Furthermore, the advancement in information technology allows for the placement of increasingly powerful and compact on-board computers, opening possibility of deploying more and more complex algorithms. The power consumption and weight of the flight controller and/or the computer board still has a big impact though. Thus, researching and optimizing algorithms is an ever-going task for developers.

The paper is structed as follows: firstly, the real-world applications are introduced, then the formulation of the environment, after that the used algorithms are explained and finally the results and the possible ways of improvements are also discussed.

## 2. Application

Drones are utilised in various applications and by many industries, institutes and organisations all over the world. These days drones are helping people to create high definition, photogrammetric maps of large areas in a cost-effective manner. There are numerous utilisations which can be applied to smart city applications, greatly needed for effective and high-performance area scanning algorithms. In the next section some of these use cases will be introduced to support the importance of our research on which this paper is mainly focused.

### *2.1. Traffic and Crowd Management*

As population on Earth is growing and the huge cities overpopulate, in the future it will be inevitable for smart cities to build such a system which is capable of controlling the enormous crowds. This system should not only be able to control pedestrians but also traffic on the roads to minimise the emergence of traffic jams. In order to solve this problem, UAVs could be utilised to monitor the roads and supply instant, accurate information for the online traffic management system. A company called ideaForge is already developing such a system which is intended to solve the task.[13]

### *2.2. Civil Security Control*

Another aspect of next generation cities which must be taken into consideration is crime reduction. As the cities are evolving and growing in size, the number of felonies committed might also be on the rise in the future. It will be even harder to find and localise criminals, consequently there must be a competent system implemented which is able to support police activities in such situations. A possible solution is to build such a UAV system which is not only able to monitor the streets but can also detect any kind of irregularity which might imply a felony. These drones

[12] W. Elfrink, 'The Smart-City Solutions.' McKinsey & Co, 2012, ideaForge.

[13] See www.ideaforge.co.in/drone-uses/crowd-monitoring/ (18. 11. 2019.)

will be equipped with enhanced sensory systems, huge capacity computational resources and fast, wireless communication technologies, which are expected to be developed by the time smart cities appear; and which will allow establishing communication channels to police stations, providing them with instant information on the illegal activity; thus the necessary measures can be taken as fast as possible.[14]

## *2.3. Environmental Management*

In the future it is going to be even more vital to establish a healthy environment in which a large group of people can live their lives together. One aspect of this is clean air, which is closely related to green areas like forests and parks, since plants are the natural sources of fresh oxygen in the air. Planting and then watering the trees, bushes, and so on, are also such tasks which can be automated by applying drones, as they can carry the seeds and water tanks and then spot the intended place to deliver those there. Dendra Systems[15] has already introduced products which are capable to fulfill these tasks.

## *2.4. Construction*

The use cases presented so far show how well drones could be used for public services; on the other hand, they can also be utilised in various industrial fields. One industry which greatly benefits from drone mapping is the construction industry. Continuous monitoring would make possible to update on progress instantly on a regular basis. It would also be beneficial to analyse the terrain beforehand on aerial maps, which would help managers to obtain an overview of the site and to make strategic decisions. Furthermore, 3D models of the construction sites can also be generated, which can also help to resolve issues in many cases. Figure 1 shows the map and 3D model of a construction site made by DroneDeploy.

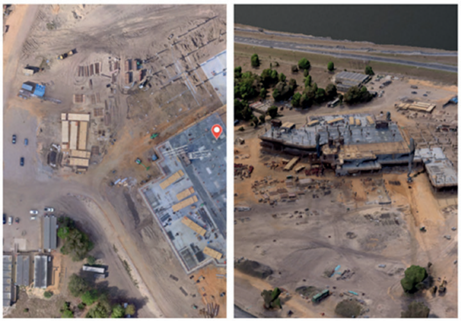

Figure 1

*Map and 3D model of a construction site.* Source: *'Drone Mapping in Construction,' DroneDeploy.*

[14] Farhan Mohammed, Ahmed Idries, Nader Mohamed, Jameela Al-Jaroodi and Imad Jawhar, 'UAVs forSmart Cities: Opportunities and Challenges,' Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), May 27-30, 2014, Orlando, FL, USA.

[15] See www.dendra.io/services (18. 11. 2019.)

### 2.5. *Land surveying*

In land surveying, precise measurements are essential to determine boundaries. During measurements, a huge amount of data must be collected about the given area, which supports the creation of maps, plots and documentation. In order to create highly accurate maps, drone photogrammetry can be used, which makes the whole process much faster. It also makes possible to generate 3D topographic maps and to collect the necessary contour data. A case study by DroneDeploy[16] shows the methodology of land surveying using drones.

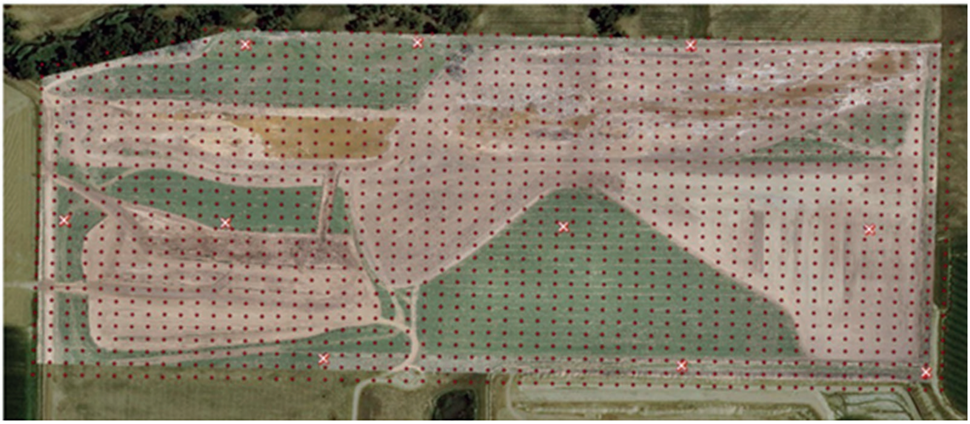

Figure 2
*Land surveying using drones.* Source: 'Mapping Drones.'

In all the above mentioned cases, monitoring a certain area is a key element. It is most efficient if the task is fulfilled in the shortest amount of time, for which a proper algorithm has to be developed. In the next sections our research on these algorithms will be introduced.

## 3. Environment

In order to simplify the area scanning task, the graph traversal abstraction is used, hence the goal is to visit each and every node of a graph. The graph-based realisation of the map is shown in Figure 3. As it can be seen, the indexing starts from the upper left edge of the map, which means the origin of it is the upper left cell. The device starts from a random graph point. The value of the graph-points can be three different numbers. In case of –1, the point is out of the boundary of a given map or is an obstacle. On these terms, the device would not make a movement if the next step happened at such a point. Moreover, the value of the point can be 0, which means that point has not been visited yet. Last but not least, the value can be 1 if the device has already visited that point. These values are stored in a matrix and updated after each step.

[16] 'Mapping Drones for Professional Surveyors,' DroneDeploy, 2015.

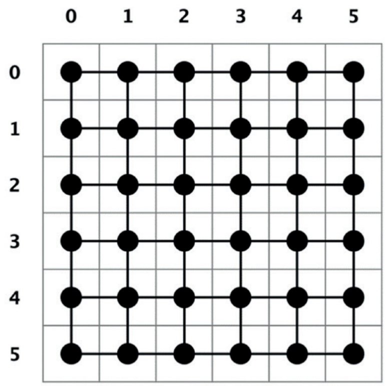

Figure 3
*The graph based realisation of the map.* Source: Created by the authors

### *3.1. The vehicle models*

A simple, four directional vehicle model has been used – which models the UAV's movement from one position to another in the graph – with the possible moves of up, down, left and right. The possible moves can be seen in Figure 4.

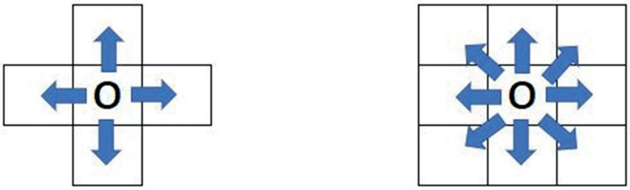

Figure 4
*The two vehicle models. Left is the actual four-move; right is the possible eight-move.* Source: created by the authors

This simple model can be further developed by an eight-move vehicle model, which is also a good solution, but for the actual problem solving, it is not needed to be developed. Also, it can be considered as a future development in the environment.

### *3.2. Map generation*

For the investigation of the performance of the two selected algorithms, several map generational rules need to be defined. A few examples of the possible maps were already introduced in Figure 3, but the main rules will be discussed in this part.

The structure of the map has been randomised, with the maximum value of obstacle (–1) points of the map being 10. In other words, the number of these graph points can range between 0 and 10. Secondly, these points are randomly divided into maximum three groups. This is the step, where the final shape of the obstacles is designed. Last but not least, these groups are placed on the map, which results in a full random map with obstacles. The random map generation causes flexibility and can be well parameterised.

### *3.3. Sensor information*

While MCTS does not need any sensor information – only the actual state of the map for generating the vertices –, RL needs a representative abstraction of the problem, which can be fulfilled by sensor information in all four directions. This sensor information should show how many steps can be made in one direction to reach the wall. The information is stored in a four-element vector with the sensor information in the four direction.

Instead of using the interval [0; 5], it is favorable to transform the information into the interval [–1; 1]. This can be explained by the activation function of the neural network, which can be seen in Figure 5. During the learning process, if the value is greater than 1, the derivative of the tangent hyperbolic activation function will be 0. Therefore, the learning algorithm backpropagates zeros, which means that the weights of the network will not change. This means that the activation function saturates the values above 1 and below –1, hence the neural network will not learn. Thereby, normalisation helps to accelerate solving the problem.

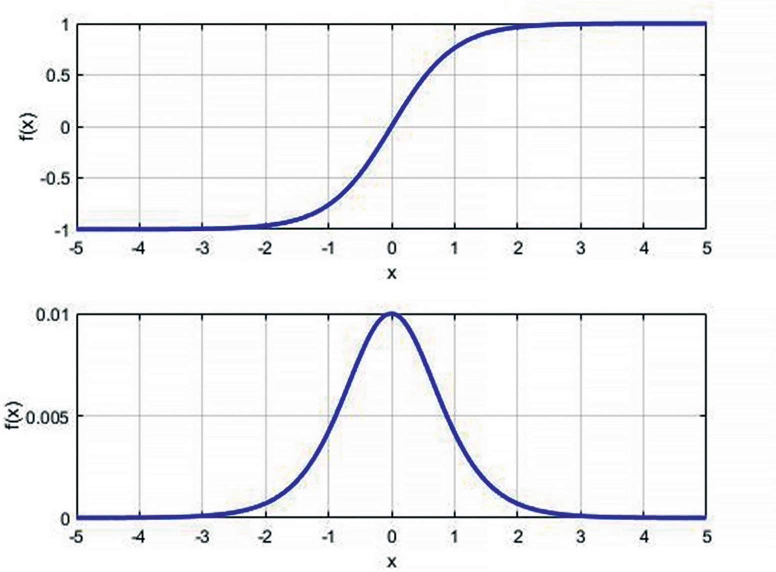

Figure 5
*Tanh activation function and its derivative.* Source: created by the authors

As it is mentioned above, a simple linear interpolation transformation is used between the possible steps and the interval with a step size of 0.4. The interpolated sensor information can be seen in Table 1.

Table 1

*Interpolation between possible steps in a direction and sensor information.* Source: created by the authors

| Possible steps in a direction | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Sensor information | 1 | 0.6 | 0.2 | –0.2 | –0.6 | –1 |

An example for the final form of the sensor vector – if we assume an empty map with the starting point in the top left corner (the directions in the vector in order: left, right, up, down) – is the following: [–1; 1; –1; 1].

Thanks to the new sensor information vector – which is based on the position and the relative distance from the wall –, the learning is implicitly accelerating, while the problem is becoming more general.

## 4. Algorithms

### *4.1. Machine Learning Based Solution*

Recently, many industries has been facing the problem of computational complexity, because there are a lot of problems that can be solved with classic algorithms or approaches, but the implementation of the needed computational resources is just not feasible, or the algorithm does not exist yet. Therefore, the interest in machine learning based methods is growing year by year, although this area also has a serious drawback, namely, such techniques cannot be validated, hence cannot be used solely in safety critical systems. Machine learning has three different areas: unsupervised, supervised and reinforcement learning. In unsupervised learning the hidden structure of the given dataset is explored, while in the supervised case the goal is to re-establish the connection between the input data and the output label. Currently the supervised learning methods are the most often used in several industries,[17] and it is also heavily researched in the aircraft domain.[18] Unfortunately, adversarial attack[19] arising from the special associative features of neural networks cause a lot of trouble in decision making, hence worsen the applicability of supervised learning.

[17] See, for example, Pierre Sermanet and Yann Lecun, 'Traffic sign recognition with multi-scale Convolutional Networks,' Proceedings of the 2011 International Joint Conference on Neural Networks, July 2011. p. 2809–2813.

[18] See, for instance, Fan Zhang, Bo Du, Liangpei Zhang and Miaozhong Xu, 'Weakly supervised learning based on coupled convolutional neural networks for aircraft detection,' *IEEE Transactions on Geoscience and Remote Sensing* 54, no 9 (2016), 5553–5563.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras and Adrian Vladu, 'Towards deep learning models resistant to adversarial attacks,' arXiv, Cornell University, 2017.

Reinforcement learning have just become popular thanks to the extraordinary results in video games,[20] robotics,[21] and other control tasks.[22] In contrast to supervised learning, it has serious advantages: it does not require a huge number of labeled training data, which is expensive to create, and in case of RL the reachable result is not limited.[23] Although RL also has some drawbacks, like struggling with convergence in the training phase, or lack of robustness and reliability.[24]

## *4.2. Reinforcement Learning*

RL uses a trial-error based technique to solve sequential decision-making problems, through which it generates the data for the learning process by interactions between the agent and the environment, as shown in Figure 6.

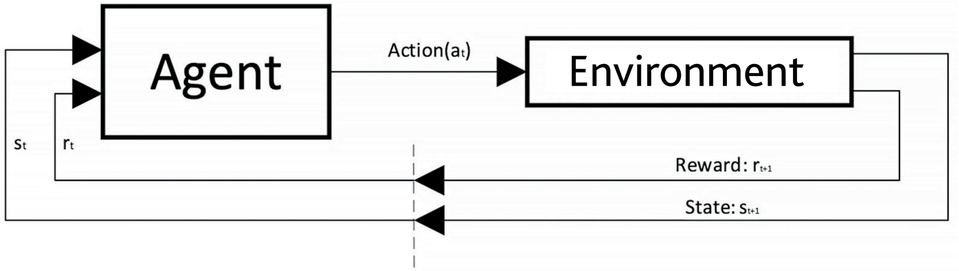

Figure 6
*The interaction between the agent and the environment.* Source: Created by the authors

The interactions between the agent and the environment are formalised as a Markov Decision Process (MDP) where the agent starts from an arbitrary state $s_0 \in S$ and then it chooses one from the possible actions $a_0 \in A$, then the environment executes the chosen actions which induce a state transition in the environment $P\,(s_t, a_t, s_{t+1})$, and finally, at the end of the step, the environment shows the immediate consequence of the execution of the chosen action with a scalar feedback called reward $r_{t+1} \in R$.

[20] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin Riedmiller, 'Playing atari with deep reinforcement learning,' arXiv, Cornell University, 2013.

[21] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver and Daan Wierstra, 'Continuous control with deep reinforcement learning,' arXiv, Cornell University, 2015.

[22] Tamás Bécsi, Szilárd Aradi, Ádám Szabó and Péter Gáspár, 'Policy gradient based Reinforcement learning control design of an electro-pneumatic gearbox actuator,' *IFAC-PapersOnLine* 51, no 22 (2018), 405–411; Árpád Fehér, Szilárd Aradi, Ferenc Hegedűs, Tamás Bécsi and Péter Gáspár, 'Hybrid DDPG Approach for Vehicle Motion Planning,' Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, 2019.

[23] See, for example, David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel and Demis Hassabis, 'Mastering the game of go without human knowledge,' Nature 550, no 7676 (2017), 354–359.

[24] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup and David Meger, 'Deep reinforcement learning that matters,' arXiv, Cornell University, 2018.

The goal of the agent is to maximize the cumulated reward for the process:

$$G = \sum_{i=1}^{T} r_t \tag{1}$$

Unfortunately, following the highest rewards may not direct the agent to this goal, because it only describes the immediate performance. Therefore the discount factor $\gamma$ ($0 \leq \gamma \leq 1$) has been introduced. The shows how the choices of the present determine the rewards of the future. If $\gamma = 0$, it means that the current decision does not influence the future rewards at all, while in the $\gamma = 1$ case a choice totally determines the future. The use of $\gamma$ allows us to further specify the goal of the agent:

$$G = \sum_{i=1}^{T} \gamma^t r_t \tag{2}$$

This is the maximisation of the cumulated discounted rewards.

If the environment has a terminal state which interrupts the training process, then the learning task is called episodic; if there is no terminal state in the environment, then the learning task is continuous. To solve the problem with RL the agent also has to solve the so-called exploration-exploitation trade-off. It basically means that at the beginning the agent does not know for sure which actions lead to the optimal behavior, so it has to choose randomly from the actions to explore the best path, but it may backfire with a punishment instead of a reward.

## *4.3. Policy Gradient*

RL has three main groups in case of learning algorithms. The first group is the value-based methods, where the neural network is trained to approximate action value function by applying the Bellmann equation:

$$Q(s_t, a_t) = r_t + \gamma max Q(s_{t+1}, a_t) \tag{3}$$

In this method the behavior – the policy – is determined indirectly, because the algorithm calculates the action values but does not recommend anything for how to exploit them. So, this method works as a situation interpretation. The drawback of this approach is that the convergence of the algorithm is not guaranteed.[25] The second group is called policy based RL, because in that case the agent is trained to directly approximate the policy, thus to determine which action to choose. The output of the neural network in every state is a probability distribution over the possible actions that sums up to 1. The biggest advantage of these methods is the guaranteed convergences to local optima.[26] The third algorithm is

[25] Dimitri P. Bertsekas, *Dynamic programming and optimal control* (Belmont, MA: Athena scientific, 1995).

[26] See Richard S. Sutton, David McAllester, Satinder Singh and Yishay Mansour, 'Policy gradient methods for reinforcement learning with function approximation,' *Advances in neural information processing systems* 12 (2000), 1057–1063.

the mixture of the introduced ones, and is called Actor–Critic. For this task the policy-based approach has been chosen because of its great convergence properties.

The Policy Gradient (PG) algorithm gives the actions by a stochastic policy $\pi_\theta\,(a_t, s_t)$, that is parametrised by the $\theta$ parameters of the neural network, which is used as a function approximator. Therefore, this is an optimisation problem where the goal is to find the weights – parameters of the neural network – that maximises the e $J(\theta) = J(\pi_\theta)$ function, which is the performance indicator of the problem:

$$J = \mathbf{E}\left[\sum_{t=1}^{T} \gamma^t r_t\right] \tag{4}$$

The PG follows the steepest ascent of the performance indicator function in order to reach local optima, hence the update rule of the $\theta$ parameters can be formalised as follows:

$$\Delta\theta = \alpha\nabla_\theta J(\theta) \tag{5}$$

where $\alpha$ is called learning rate, which is one of the most important hyperparameters of the training process, because this parameter determines the size of the change in the neural network weights.

The gradients are calculated by applying Sutton's theorem,[27] thus the final form of the gradient update rule is:

$$\theta \leftarrow \theta + \alpha\nabla log\pi(s_t, a_t)\sum_{t=1}^{T}\gamma^t r_t \tag{6}$$

## *4.4. Monte-Carlo Tree Search*

The greedy algorithms, which use heuristics and which contain environment specific information and strategies, are applied in several areas, but in general there are no guaranties for the performance, because of the nature of the heuristics. The opposite of these methods is the uninformed search algorithms like breadth first or depth first search. These methods are able to find the optimal solution of discrete problems if there is one. In the meantime, these methods cannot be used in real world applications, because the needed computational resources do not exist yet, or, in other cases, it is not feasible to implement this kind of equipment into devices that are mass produced. So, the problems are general applicability and computational expenses; these concerns are mitigated by the MCTS algorithm.

The MCTS algorithm builds a tree representation of a problem node by node with the help of the model of the environment. The algorithm has four steps, that is repeated during the building process:

- Selection: Starts from the current state's node R and recursively chooses the best child node until it reaches a leaf node;
- Expansion: If the leaf node is not a terminal node, then it populates its child nodes with the generative model of the environment, and chooses one C child node of them;

[27] Ibid.

- Simulation: Carries out a random rollout from the current node, which means choosing actions randomly until a terminal node occurs;
- Backpropagation: Updates the nodes information between the path of R and C with the result of the rollout.

MCTS uses the UCT (Upper Confidence bound applied for Trees) algorithm to choose the best node in the selection phase:

$$v_i + C\sqrt{\frac{lnN_i}{n_i}} \tag{7}$$

where the $v_i$ is the value of the current node and the $N_i$ is the number of visits of the parent node, while the $n_i$ is the number of visits of the current node and is the constant that controls the exploration-exploitation trade-off.

The node with the highest UCT value is chosen every time. The two sides of the UCT algorithm are the opposites of each other, because while the left side with the expression goes to the direction that seems the most valuable, the right side drives the attention into the less explored parts of the tree, hence through the $C$ constant the trade-off is domain specifically parametrisable. Thanks to that, it can be proved that the MCTS has an optimal solution if enough time is given.[28]

### *Customization of MCTS*

In order to further decrease the computational complexity of the problem, domain specific knowledge is implemented into the tree to carry out cut-offs. These techniques let the algorithm reduce the search space by ignoring the generation of specific nodes based on pre-implemented rules. In this case the rules are:

- Do not populate nodes that are created by actions that lead the algorithm to cells that are already visited.
- Do not populate nodes that are created by actions that lead the algorithm to obstacles or cells out of the map.

The search space for the algorithm is remarkably simplified by these rules, which also decrease the time needed for the decision-making process.

The calculation of the values in (7) are based on the reached state at the end of the rollout phase. If all the cells are visited ones – so the map is scanned –, then the given value is 1, otherwise it is –1.

[28] See Levente Kocsis and Csaba Szepesvári, 'Bandit based monte-carlo planning,' in *Machine Learning: ECML* 2006, ed. by J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Berlin – Heidelberg: Springer, 2006), 282–293.

# 5. Results

## *5.1. Reinforcement learning-based solution*

There are many parameters that affect the result of the training and do not change during the process at all; these parameters are called hyperparameters. Hyperparameters are the number of neurons, the number of hidden layers, the discount factor, the update frequency of the neural network parameters, or the used activation functions, and also the learning rate. The chosen values are shown in Table 2.

Table 2
*Hyperparameters.* Source: created by the authors

| Parameter | Value |
|---|---|
| Number of neurons per layer | 128 |
| Number of hidden layers | 4 |
| Update frequency of neural network parameters ($\xi$) | 25 |
| Discount factor ($\gamma$) | 0.3 |
| Activation function | TANH |
| Learning rate ($\alpha$) | 0.0001 |

It is also important to specify the terminal conditions and the rewards of the environment before training. The episode is over if the agent chooses an action that leads to a cell – or node – which is already visited, or an obstacle, or the wall; the episode also ends when the agent visits every node once. The final criterion is that all the episodes should last until a terminal state occurs. In that problem immediate rewards are not used, hence the agent gets the feedback at the end of the episode, which is 1 in case of success and –1 otherwise.

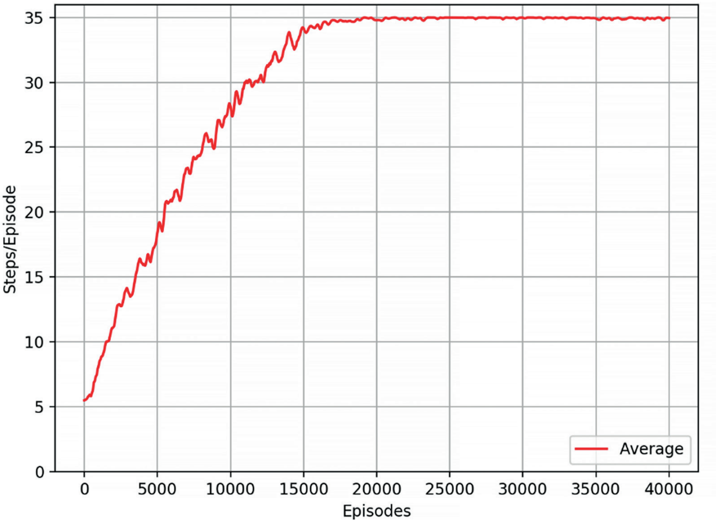

Figure 7
*The convergence of the learning algorithm.* Source: created by the authors

In the training phase the task is to scan a simple 6×6 map where the agent starts from random positions in all the episodes. The result of the training is shown in Figure 7.

Figure 7 shows the number of steps taken in each episodes; after 20 000 episodes the agent learned how to visit all the nodes in the graph – to scan all the cells in the map – without visiting any node more than once, which is considered as the optimal solution of this task.

After the evaluation of the trained agent the result showed that it learned to avoid steps which would lead to already visited cells, and it is able to solve a map which has the same shape but different size, hence the relative sensor information vector encouraged the agent to generalise the problem and to learn patterns instead of absolute scenarios, due to the fact that the agent developed some kind of scale invariance. Figure 8 displays strategies executed in maps with different sizes.

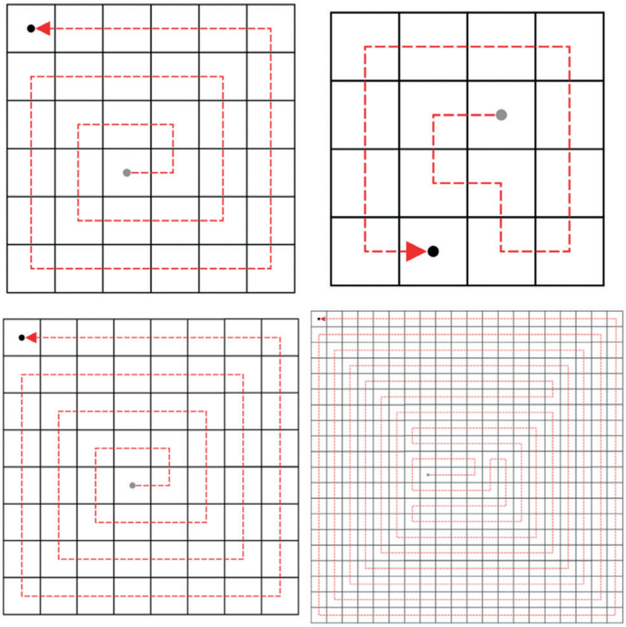

Figure 8

*Strategies on maps of different size.* Source: created by the authors

## 5.2. The search-based solution

The search-based method uses the MCTS as an online planner and operates as follows:

1. The algorithm starts from a random position in a randomly generated map;
2. Executes an MCTS with a fixed number of iterations;
3. Chooses an action based on the result of the MCTS;
4. Updates the state of the map by the execution of the chosen action;
5. Repeats until a terminal condition is fulfilled.

The terminal conditions are the same as in the RL based solution. The decision-making is made by the recommendation of the MCTS, which is a probability distribution over the possible actions in the given state. The probability distribution is created by the visit counts of the child nodes of the given state. In order to get probabilities from the visit counts, the softmax function is used:

$$\sigma(n_i) = \frac{e^{n_i}}{\sum_{j=1}^{K} e^{n_j}} \tag{8}$$

where $n_i$ is the number of visits of the current node and $K$ is the number of child nodes in the given state.

The action with the highest probability is chosen in all case. Figure 9 shows some strategies carried out by MCTS.

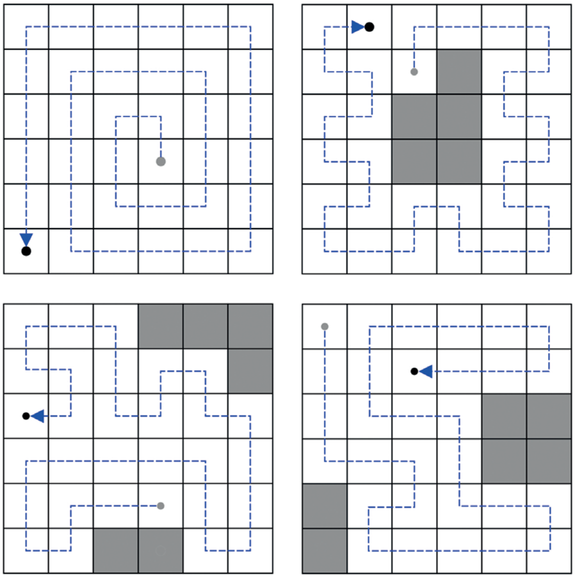

Figure 9
*Traversals carried out by the MCTS algorithm.* Source: created by the authors

The MCTS is able to solve any randomly generated map from any starting point. The performance of this method is limited by the size of the map, hence the computational expenses, because the bigger the map the more iteration is needed to find the optimal decision in the given state.

## 6. Conclusion

We introduced two approaches that solve the simplified form of the area scanning problem in different ways. The MCTS is able to carry out a traversal strategy for both convex and concave shapes from any starting point in the area, while real-time applicability in smaller domains is maintained. Unfortunately, as the number of cells which should be visited increases, it still becomes too complex in terms of computation; to challenge this issue in our further work, we propose to investigate the possibility of decomposing the large areas into smaller, thereby manageable ones which can be solved faster.

The RL based solution is reliably able to provide real-time applicability, and through the use of relative sensor information a scale invariancy in similar shapes has been learned, so the agent can solve the same shapes with different sizes from different starting points. Unfortunately, in that case the agent is only trained on one shape, hence it is not able to solve any kind of shapes; so it is less robust then MCTS.

The next steps of this research will be the development of the environment to be more realistic and to train more robust agents by using different shapes and scales in the training process; we also see enormous potential in the synergy of the introduced methods, because the absolute robustness of the MCTS and the real-time applicability of the RL are the two most important properties of an algorithm that might be able to solve such problems. Therefore, we propose to investigate the applicability of planning agent methods that are already successfully used in several domains.[29]

## Acknowledgement

**The results are used in the 'Small aircraft hybrid propulsion system development' project, supported by Hungarian national EFOP-3.6.1-16-2016-00014 project titled 'Investigation and development of the disruptive technologies for e-mobility and their integration into the engineering education.'**

### Bibliography

Bécsi, Tamás – Aradi, Szilárd –Szabó, Ádám – Gáspár, Péter: 'Policy gradient based Reinforcement learning control design of an electro-pneumatic gearbox actuator.' *IFAC-PapersOnLine* 51, no 22 (2018), 405–411. DOI: https://doi.org/10.1016/j.ifacol.2018.11.577

Bertsekas, Dimitri P.: *Dynamic programming and optimal control*. Belmont, MA, Athena scientific, 1995.

Dendra systems, url: www.dendra.io/services (18. 11. 2019.)

Elfrink, W.: 'The Smart-City Solutions.' McKinsey & Co, 2012, ideaForge. Available: www.ideaforge.co.in/drone-uses/crowd-monitoring/ (18. 11. 2019.)

'Drone Mapping in Construction.' DroneDeploy. Available: www.dronedeploy.com/resources/ebooks/drone-mapping-construction-earthwork/#form (01. 05. 2019.)

Fehér, Árpád – Aradi, Szilárd – Hegedűs, Ferenc – Bécsi, Tamás – Gáspár, Péter: 'Hybrid DDPG Approach for Vehicle Motion Planning.' Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics, 2019. DOI: https://doi.org/10.5220/0007955504220429

[29] Silver et al, 'Mastering the game of go,' 354.

Henderson, Peter – Islam, Riashat – Bachman, Philip – Pineau, Joelle – Precup, Doina – Meger, David: 'Deep reinforcement learning that matters.' arXiv, Cornell University, 2018. Available: https://arxiv.org/abs/1709.06560 (18. 11 2020.)

Kocsis, Levente – Szepesvári, Csaba: 'Bandit based monte-carlo planning.' In: *Machine Learning: ECML* 2006, ed. by Fürnkranz, J. – Scheffer, T. – Spiliopoulou, M. Berlin – Heidelberg, Springer, 2006. 282–293. DOI: https://doi.org/10.1007/11871842_29

Lillicrap, Timothy P. – Hunt, Jonathan J. – Pritzel, Alexander – Heess, Nicolas – Erez, Tom – Tassa, Yuval – Silver, David – Wierstra, Daan: 'Continuous control with deep reinforcement learning.' arXiv, Cornell University, 2015. Available: https://arxiv.org/abs/1509.02971 (10. 11 2020.)

Madry, Aleksander – Makelov, Aleksandar – Schmidt, Ludwig – Tsipras, Dimitris – Vladu, Adrian: 'Towards deep learning models resistant to adversarial attacks.' arXiv, Cornell University, 2017. Available: https://arxiv.org/abs/1706.06083 (10. 11 2020.)

'Mapping Drones for Professional Surveyors.' DroneDeploy, 2015. Available: https://blog.dronedeploy.com/case-study-830cfc23db55 (01. 05. 2019.)

Mnih, Volodymyr – Kavukcuoglu, Koray – Silver, David – Graves, Alex – Antonoglou, Ioannis – Wierstra, Daan – Riedmiller, Martin: 'Playing atari with deep reinforcement learning.' arXiv, Cornell University, 2013. Available: https://arxiv.org/abs/1312.5602 (10. 11 2020.)

Mohammed, Farhan – Idries, Ahmed – Mohamed, Nader – Al-Jaroodi, Jameela – Jawhar, Imad: 'UAVs for Smart Cities: Opportunities and Challenges.' Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), May 27-30, 2014, Orlando, FL, USA. DOI: https://doi.org/10.1109/icuas.2014.6842265

Musa, Sam: 'Smart City Roadmap.' 2016. Available: www.academia.edu/21181336/Smart_City_Roadmap (10. 11 2020.)

Sermanet, Pierre – Lecun, Yann: 'Traffic sign recognition with multi-scale Convolutional Networks.' Proceedings of the 2011 International Joint Conference on Neural Networks, July 2011. p. 2809–2813. DOI: https://doi.org/10.1109/ijcnn.2011.6033589

Shreih, R.: 'Intelligent Systems for Smarter Cities.' King Abdullah University Of Science & Technology, 26 August 2013. Available: https://innovation.kaust.edu.sa/intelligent -systems -for – smarter -cities/ (10. 11 2020.)

Silver, David – Schrittwieser, Julian – Simonyan, Karen – Antonoglou, Ioannis – Huang, Aja – Guez, Arthur – Hubert, Thomas – Baker, Lucas – Lai, Matthew – Bolton, Adrian – Chen, Yutian – Lillicrap, Timothy – Hui, Fan – Sifre, Laurent – Driessche, George van den – Graepel, Thore – Hassabis, Demis: 'Mastering the game of go without human knowledge.' Nature 550, no 7676 (2017), 354–359. DOI: https://doi.org/10.1038/nature24270

'Smart Cities.' King Abdullah University of Science & Technology, 2013. Available: https://innovation.kaust.edu.sa/industry/brows e-technology/smart-cities/ (10. 11 2020.)

Sutton, Richard S. – McAllester, David – Singh, Satinder – Mansour, Yishay: 'Policy gradient methods for reinforcement learning with function approximation.' *Advances in neural information processing systems* 12 (2000), 1057–1063.

Zhang, Fan – Du, Bo – Zhang, Liangpei – Xu, Miaozhong: 'Weakly supervised learning based on coupled convolutional neural networks for aircraft detection.' *IEEE Transactions on Geoscience and Remote Sensing* 54, no 9 (2016), 5553–5563. DOI: https://doi.org/10.1109/tgrs.2016.2569141