Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 58-4 (2024) 228–233

Safe vehicle motion design with learning for moving in environment with uncertainties ⋆

Dénes Tompos *,**, **Balázs Németh** *,**, **Tamás Hegedűs** *, **Vu Van Tan** ***, **Péter Gáspár** *,**

* *Institute for Computer Science and Control (SZTAKI), Hungarian Research Network (HUN-REN), Kende u. 13-17., 1111 Budapest, Hungary (e-mail: [denes.tompos, balazs.nemeth, tamas.hegedus, peter.gaspar]@sztaki.hun-ren.hu).*

** *Department of Control for Transportation and Vehicle Systems, Budapest University of Technology and Economics, Stoczek u. 2., 1111 Budapest, Hungary*

*** *Department of Automotive Mechanical Engineering, Faculty of Mechanical Engineering, University of Transport and Communications, 3 Cau Giay Street, 100000 Hanoi, Vietnam*

Abstract: In this paper a motion profile design for unmanned aerial vehicles is proposed which method is able to guarantee safe collision-free motion. The motivations of the work are provided by the uncertainties of covered areas by the vehicles, and also the need of high performance fast vehicle motion. The uncertain information on the environment for detecting conflict areas is processed through clustering and Mahalanobis-distance-based filtering methods. The resulted conflict areas are involved in the motion design method, which is facilitated through reinforcement learning. This paper shows the application of the method on a drone that moves together with a mobile robot in the same environment. The safe and high performance motion of the drone is illustrated through simulation example.

Keywords: Unmanned aerial vehicles, safe motion, clustering point clouds.

1. INTRODUCTION AND MOTIVATION

This paper proposes a safe motion planning algorithm with reinforcement learning (RL) for collision-free motion of indoor unmanned drones. The development of the algorithm is motivated by automated manufacturing systems that contain various unmanned robots and drones. The motion of the drones must be safe for avoiding collision to objects in their surroundings. Moreover, their motion must be effective, i.e., velocity profile of each drone must be improved for supporting continuous production in the manufacturing system. This paper focuses on the design of drone motion profile, such as velocity planning.

In the field of trajectory and velocity planning, various methods have also been developed in the area of trajectory and velocity planning. For example, in the paper Szmuk et al. (2017) a lossless convexification method has been proposed. The main advantage of this algorithm is that it can be used for real-time onboard applications on quadcopters. Using the convexification technique, non-convex constraints are considered during convex optimal control to avoid obstacles. The algorithm is tested in a real implementation for an agile flip maneuver. In addition, a collision-free velocity profile design method can be found in Sabetghadam et al. (2022). This research aims to design a feasible velocity profile with gaps smaller than the diameter of the UAV. For this problem, it is sufficient to consider the orientation of the UAV. The UAV and the obstacles are modeled as convex sets and the Bézier curve is used for collision-free trajectory planning.

Furthermore, paper of Lupascu et al. (2019) proposes a method for generating trajectories using 3D rectangular cuboid representation. The paper of Yang et al. (2020) investigates designing optimal trajectories for multiple UAVs while constraining curvatures to meet feasibility requirements. Although traditional approaches are still relevant in velocity planning, machine learning-based methods are emerging as viable alternatives. The study described in Hu et al. (2021) outlines a learning-based strategy that concentrates on designing trajectory and velocity profiles for fleets of UAVs. Significantly, the fusion of game theory and the estimation of Nash equilibria is essential for tackling interactions among various UAVs, see Spica et al. (2020).

A new trend in velocity planning is the incorporation of data-driven and learning-based methods. Using these techniques, it may be possible to guarantee performance that is difficult to formalize. For example, based on visual and inertial sensors, the UAV is able to navigate itself in an unknown environment using the potential field approach Mac et al. (2016). In an indoor environment, for visual-based navigation purposes, deep neural network-based solutions provide a powerful tool to effectively solve the problems Chhikara et al. (2021). In addition, neural network-based solutions are also suitable for UAV control

⋆ The paper was partially funded by the National Research, Development and Innovation Office (NKFIH) under OTKA Grant Agreement No. K 135512. The research was also supported by the National Research, Development and Innovation Office through the project "Cooperative emergency trajectory design for connected autonomous vehicles" (NKFIH: 2019-2.1.12-TÉT VN).

10.1016/j.ifacol.2024.07.222

in an indoor corridor environment. In the paper Padhy et al. (2018) a method has been introduced that uses the output of the monocular camera. The Convolutional Neural Network (CNN) is used to classify the sides of the corridor, which is used to navigate the drone through an initially unknown environment. Furthermore, a UAV flight controller has been presented in Bui et al. (2022). The proposed method also uses camera information and a CNN to explore and safely navigate the UAV through the environment. In addition, the proposed method can correct the position and direction of the UAV.

Although data-driven methods are also suitable for UAV control purposes, but the main advantage of these methods is the reduced online computational capacity of the algorithm. In the paper of Salzmann et al. (2023) a so-called real-time neural model predictive control (MPC) is presented, which aims to overcome the problem of computation time. In the proposed solution, the neural network is used to create an accurate dynamic model of the system, and then the online optimization is performed. The test results show that the real-time implementation requirements can be met and the control performances are significantly improved due to the obtained accurate UAV model.

The proposed solution of this paper is based on a reinforcement learning (RL) based solution which is extended with a supervisor for guaranteeing collision-free motion. This paper improves the preliminary solution which is found in Tompos and Németh (2023). The main contribution is the providing of safe motion profile under uncertain size of the moving obstacles. Although the routes of the obstacles in the manufacturing system are considered to be known, their real covered route suffers under the uncertainty of the moving obstacle size. In this paper an enhanced clustering based method is proposed with which the conflict points of the drone and the moving obstacles based on noisy point cloud information can be determined. The result of the clustering is built in the environment of the RL-based training process and also in the supervisor.

The paper is organized as follows. Section 2 the selection process of conflict points is presented. Moreover, the operation of the clustering algorithm is illustrated through a simulation example. The design process of the safe velocity is found in Section 3. That section presents the also illustration on the effectiveness of the entire algorithm. Finally, the paper is concluded in Section 4.

2. CONFLICT POINT SELECTION METHOD

This section presents the conflict point selection method, i.e., the clustering algorithms for noisy point clouds. First the algorithm is presented and second, an example on the operation can be found.

2.1 Clustering method for noisy conflict point clouds

The goal of conflict area selection is to find points in the environment where the routes of vehicles cross, i.e., more than one vehicle covers the points. The selection has three steps: the selection of conflicting point cloud, the clustering of point cloud and the filtering of the resulted clusters.

For each point along the path $\mathcal{P}$ in the proposed algorithm, corresponding points in the graph $\mathcal{G}$, see the review of Yang et al. (2023). In this paper the goal is to explore routes between arbitrarily chosen positions, which results in a densely generated graph within the environment. The generated graph gives the basis of the trajectory planning for the drone, see Hegedűs et al. (2023). The points of the graph are assigned at a certain safe distance s_{safe} determined by the vehicle. This specific safe distance is related to the physical dimensions of the vehicles, which are modeled as spheres and centered at P_0. Then, each graph point P is chosen in such a way that the vehicle can practically occupy it, with the definition $\forall P \in \mathcal{G}, P_0 \in \mathcal{P} || P_0(x,y,z) - P(x,y,z)|| < s_{safe}$, as explained in the algorithm 1. This method, despite its computational intensity, offers advantages for simulation purposes. The graph calculations are done outside the simulation environment, which adds to its advantage. Sparse coordinates marking the path allow not all graph points to be selected. The result of the

Algorithm 1. Clustering method for conflict point selection

conflict_points $= \{P_1, P_2, \ldots, P_n\}$
while conflict_points $\neq \emptyset$ **do**
  cluster$_k \leftarrow P_1$
  for $\forall P_c \in$ cluster$_k$ **do**
    if $\forall P \in$ conflict_points, $P_c \in$ cluster and $||P_c(x,y,z) - P(x,y,z)|| < s_{\text{safe}}$ **then**
      cluster $\leftarrow P$
    end if
  end for
end while

selection process is a comprehensive list comprising data for all conflict points. Segmentation is necessary using a clustering method, which relies on the distance between points in the list. A cluster is a subset of intersecting points where there is no secure point between any pair of members, thereby enabling the controlled vehicle to halt firmly. Therefore, clusters are formed by selecting points located within a safe distance of each other. The proposed method entails selecting the starting point from the list and checking for any additional points within the safe distance, denoted as s_{safe}. If there are any, their immediate vicinity is analyzed for any further points within the safe distance. This process is repeated until no more points are identified within the safe distance, thereby segregating one cluster from the list of conflict points. The iterative process continues until all the points are successfully partitioned into clusters.

In the next step the resulted point cloud is separated using clustering algorithm K-means method, see Hartigan and Wong (1979). The algorithm separates elements of a group based on certain similarities thus it will separate the points of our list based on their positions in space. K-means does that by creating k number of means in the space and grouping the points to the closest mean to them:

$$clusters = C_1, ..., C_k, C_k = P_1, ...P_n \tag{1a}$$

$$Inertia = \sum_{i=1}^{k} \sum_{j \in C_i} ||P_{ij} - \mu_i||^2 \tag{1b}$$

The algorithm calculates the inertia of each group, which in this case is the physical distances from the means. The algorithm then groups the points to the closest mean to them. The algorithm will attempt to optimize the inertia of the clusters, which is the sum of distances between the points and their means. It does it with updating the positions of the means by recalculating them from the created group. Then it redistributes the points to the closest mean, thus repeating this two steps until a stopping criteria is met. The stopping criteria for the K-means algorithm are the position of the means remains consistent or the points in the clusters does not change

Algorithm 2. K-means Clustering
1: Select k random means
2: **while** means$_{t-1} \not\approx$ means$_t$ **do**
3: **for all** $P \in$ conflict points, $\forall \mu_i, i = 1, \ldots, k$ **do**
4: Calculate distance $||P_n - \mu_i||$
5: Group all P to its smallest distance group C_i
6: Update means $\mu_i = \frac{\sum_{j \in C_i} P_j}{n}$
7: **end for**
8: **end while**

or the maximum number of iterations is reached. The algorithmic formulation of the applied K-means clustering method can be found at Algorithm 2.

Remark that K-means requires the number of clusters predefined in order to separate the conflict point list into clusters. The standard solution used to solve this problem is the Elbow method, see Kodinariya et al. (2013). It works by plotting the inertia for all possible k number of clusters on a curve. The inertia of the system drops fast with the increase of the number of clusters, until the optimum number of clusters is reached. Visually it forms a curve with an 'elbow' at the optimal number of clusters.

Finally, the resulted clusters must be filtered for avoiding noisy points in the point cloud. Due to measuring error it is possible to have points in the conflict point list, which are not part of any conflict areas. To decide if a cluster point belongs to its cluster, its Mahalanobis distance M can be calculated, see Xiang et al. (2008). Its advantage compared to Euclidean distance is that the shape of the cluster can be considered by this metric. The algorithm operates by weighing the distance from the mean of the cluster μ with the correlation of the points P_l in the cluster. It means that if the point the distance is measured from a direction where the point of the cluster is strongly correlated, the clustering algorithm divides the distance with a large weight, thus indicating that the point is closer to the cluster itself:

$$M_i^2 = (P_l - \mu_i)^T * Cov_i^{-1} * (P_l - \mu_i). \quad (2)$$

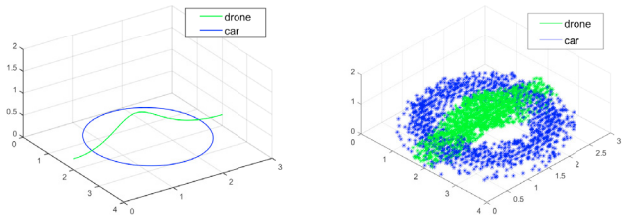
The correlation of the points in the cluster i is represented with the covariance matrix Cov_i in the (2). The covariance matrix is an indicator for detecting the shape of a cluster, since it results in increased values if the different components of a cluster elements dependent on each other. The result of the Mahalanobis distance computation is not an exact distance from the cluster in Euclidean metric, instead it is a probability-like metric that a given point is involved in the cluster.

The result of the filtering process is a set of clusters. The mean of each cluster i is applied to compute the S_j distance of vehicle j from the conflict point. Moreover, the resulted set of points in cluster i are applied to compute the safe distance for vehicle j, such as $s_{safe,j}$. The safe distance $s_{safe,j}$ represents an Euclidean distance, which must be kept by vehicle j, if colliding vehicle k is inside of conflict area i, see Németh and Gáspár (2023). In case of two vehicles, such as $j = 1$ and $k = 2$, each vehicle has its own safe distance, such as $s_{safe,1}$ or $s_{safe,2}$. Moreover, their physical distances from the conflict point i are S_1, S_2. These distance information are transferred to the optimization process of the supervisor.

2.2 Example on the conflict point selection process

To train and evaluate the algorithm, a simulation environment is created. In the environment ground vehicles, i.e., mobile robot obstacles are moved that emulate the configuration of a physical demonstration laboratory. Due to spatial limitations of the real laboratory, a simulation environment with dimensions of 4 meters in length, 3 meters in width, and 2 meters in height is selected. Within this simulation, a scenario is presented, in which a mobile robot acts as an obstacle and follows an elliptical trajectory. In the example the noise of the vehicle trajectories is represented by χ^2 distribution with a degree of freedom value 2.

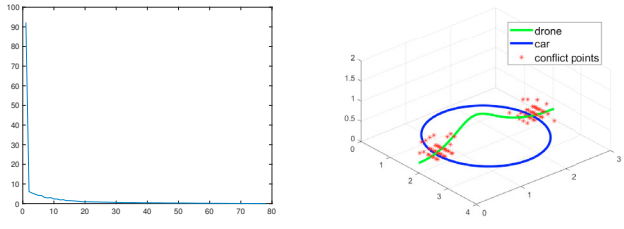
Additionally, a drone moves parallel to the y-axis and intersects the trajectory of the mobile robot. In the example the motion of the drone is controlled. In the simulation a static wall positioned with the height of 1 meter parallel to the x-axis. The traveling height of the drone is preferred to be as close to the ground as possible and the avoidance of collisions with walls is achieved by the vertical elevation of the drone. On the other hand, preventing collisions when working with mobile robots requires adjusting the velocity of the drone. An example of the environmental model is depicted in Figure 1(a), illustrating the path of the drone in green and the path of the mobile robot in blue. Figure 1(b) shows the paths of the objects and the covered area around these paths. It can be seen that due to the uncertainties of the sizes of the drone and the mobile robot, the occupied areas are noisy from the viewpoint of clustering.



(a) Routes of the drone and the mobile robot (b) Occupied areas

Fig. 1. Illustration of the simulation example.

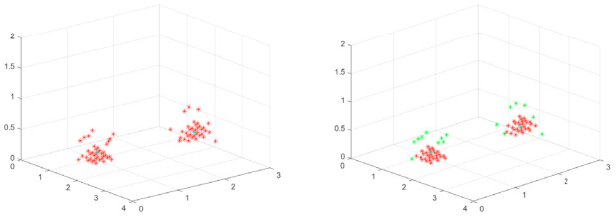
The illustration on the effectiveness of the proposed method is based on this example. First, the results of applying Algorithm 1 in Figure 2 are found. In the case of the simulation created for the control the designed pathways have two separate space where collision is possible. The elbow curve also shows the optimal number of clusters to be two, see Figure 2(a). Figure 2(b) illustrates for the clusters of conflict points around the conflict situation in the given example. The path of the drone is marked with green and the path of the car is marked with blue and the conflict point are red.



(a) Elbow curve of the k-means clustering algorithm. (b) Clusters for conflict point selection

Fig. 2. Selection of clusters for conflict point determination.

Secondly, results of the Mahalanobis distance-based filtering process is shown. A critical value has to be determined which informs if the point is inside the cluster or not. The significance level is chosen to be 0.1, which means the given point is 10% or less likely to be out side of the of the cluster. At this significance level the critical number is 4.605, which means that if the Mahalanobis distance is under this number, that point is considered to be the member of the cluster. The result of the filtering process is illustrated in Figure 3.



(a) Noisy clusters. (b) Filtered points (green) with Mahalanobis distance algorithm.

Fig. 3. Illustrating the result of filtering through Mahalanobis distance computation process.

2.3 Computation of vehicle distances from the mean of the cluster

The results of the filtered clusters are applied to the computation of S_j and $s_{safe,j}$. Figure 4 illustrates an example with three vehicles around a clustered set of points. The blue colored points represent the cluster point members, green lines connect each of the vehicles to the mean of the cluster and red points represent the furthest points of the cluster from the mean, which are closest the green lines. The distance of a red point from the mean represents $s_{safe,j}$. Similarly, the length of a green line represents S_j.

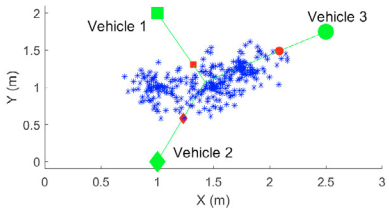


Fig. 4. Illustration of distance computation for three vehicles.

Since the conflict ares are given by points, the computation of an $s_{safe,j}$ value leads to an iteration process in practice. The selected points of the corresponding green line are used for testing the P_l points of the cluster to find the red point, i.e., the edge of the cluster. The distance of the selected red edge point P_l to μ_i is considered to be $s_{safe,j}$. if a selected point of the corresponding green line is close to μ_i, i.e., the selected point is inside of the cluster, then the distance r_{min} of the selected point from the closest P_l point has low value. But, if a selected point of the corresponding green line is far from μ_i, then the distance $r_{min,j}$ of the selected point from the closest P_l point has high value. Consequently, the edge point of the cluster for vehicle j can be selected based on the analysis of $r_{min,j}$ distances, i.e., distances between the points of the green line and the corresponding closest P_l point. The idea selection process is close to the elbow principle, i.e., the characteristics of r_{min} shows a spike in value when it leaves the cluster. Figure 5 illustrates the computed $r_{min,j}$ value for each vehicle. For example, in case of Vehicle 2 the selection of $s_{safe,2} = 0.52m$ is recommended due to the corresponding $r_{min,2} = 0.02m$ value, see the green circle in Figure 5.

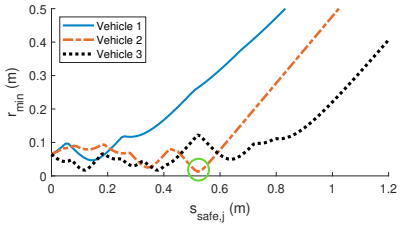


Fig. 5. Illustration of $r_{min,j}$ for finding $s_{safe,j}$.

3. DESIGN OF SAFE MOTION PROFILE WITH LEARNING FEATURE

The motion of the drone is performed with a control algorithm, which has been presented in a previous work, see Tompos and Németh (2023). The aim of the method is to consider interactions of vehicles when crossing each others routes with assuming both vehicles move straight. One of the unmanned vehicle must adapt to the motion of the other vehicle in such a way that a minimum safe distance from it is kept at any time of the interaction. The goal of this control is to maximize the velocity of the controlled vehicle (i.e., the drone in the example), while simultaneously to avoid collision to the surrounding objects (i.e., to the mobile robot).

The structure of the motion profile design is illustrated in Figure 6. It contains a learning-based agent and a supervisor. The supervisor computes a $u(k) = u_L(k) + \Delta(k)$ acceleration input for the drone in each k time instance, with which collision to the surrounding objects can be avoided. The learning-based agents provide a candidate acceleration signal $u_L(k)$ to the supervisor. In this work it is designed using a reinforcement-learning-based algorithm, which attempts to optimize the energy consumption of the system via minimizing the acceleration and deceleration of the drone. Thus, the learning-based agent and the supervisors work together: the learning-based agent facilitates energy and time performance requirements and the supervisor guarantees collision avoidance. This latter property is achieved through the computation of peak-bounded $\Delta(k)$ for guaranteeing collision free motion. Moreover, in Figure 6 the elements of the selection process are involved, see Section 2.1. The preprocessing module results the conflict point clouds. These are calculated offline through the clustering algorithm and the Mahalanobis distance based filtering. The conflict point selection block involves the algorithms needed to be executed in each time interval (online). It contains an algorithm selecting the one conflict point cluster to work with. It also calculates the requested $s_{safe,j}$ value.

In the proposed structure the energy and time optimal motion through an RL process is achieved, via the appropriate definition of reward. During the training process the agent receives a reward from the environment, which reward reflects to the selected candidate acceleration $u_L(k)$. The methodology utilized in this paper is a training method that incorporates the model-free, online, and off-policy

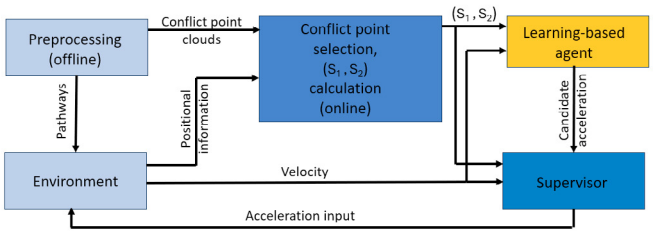


Fig. 6. Structure of the proposed control system.

deep deterministic policy gradient (DDPG) outlined in Gu et al. (2016). The reason of its selection is that is is easy to tune and low training time can be achieved. It also provides an appropriate basis to tune reward function for applying more complex learning methods in the future. The resulted agent is derived by a structured actor-critic relationship with approximators. Throughout the training process, the system calculates an optimal policy to maximize long-term rewards, equating the discovery of such a policy with the acquisition of knowledge of the Bellman equation. Actor and critic approximators possess observations. The aim of the actor approximator is to identify an action $u_L(k)$, that optimizes the long-term reward. Simultaneously, the critic endeavors to estimate the expected value of the long-term reward. The agent must learn to move the controlled object forward and optimize it energy consumption, i.e., it is rewarded for gaining distance with minimum amount is acceleration. Moreover, reward is lost if the two conflicting object enters the avoidable circle the same time. The reward function for Vehicle j is defined as:

$$r(k) = S_j(k) - Q|u(k)| - \Theta(k). \tag{3}$$

The magnitude of acceleration input $u(k)$ is applied in the reward with a constant penalty weight $Q > 0$. Another penalty $\Theta(k)$ is in relation with the entering of the drone into the conflict area, when another vehicle is also in it. Thus, its value is 0.1 if Vehicle j is in the area together with an another vehicle, otherwise 0. During each episode the reward is accumulated, which means that $\sum_{k=0}^{N} r(k)$ is used for the evaluation of each episode.

The formulation of the algorithm in the supervisor is based on a constrained quadratic optimization process, see Németh and Gáspár (2023); Tompos and Németh (2023). The goal of the supervisor is to guarantee safe drone motion and thus, e.g., Vehicle 1 (drone in the example) and Vehicle 2 (mobile robot) cannot be in the conflict area $\mathcal{R}(s_{safe,1}, s_{safe,2}$ in the same time. It forms the safety constraint in the algorithm of the supervisor: $(S_1(k+1), S_2(k+1)) \notin \mathcal{R}(s_{safe,1}, s_{safe,2})$, where $(S_1(k+1), S_2(k+1))$ are predicted distances at $k+1$ time step. If this constraint can be guaranteed by $u_L(k)$, then $u(k) = u_L(k)$ can be applied. But, if this constraint is violated by $u_L(k)$, then $\Delta(k) \neq 0$ must be selected within the range of Δ_{bound} and thus, $u(k)$ is able to guarantee safe motion. Consequently, the supervisory algorithm is formed as:

$$\min_{\Delta(k)} \ \Delta(k)^2 \tag{4a}$$

subject to

$$\Delta(k) \in \Delta_{bound} \tag{4b}$$

$$(S_1(k+1), S_2(k+1)) \notin \mathcal{R}(s_{safe,1}, s_{safe,2}). \tag{4c}$$

Finally, two remarks on (4) must be taken. First, if (4) is infeasible, i.e., $\Delta(k)$ cannot be found for guaranteeing constraints, the drone is stopped for temporary safety reasons. Second, the mathematical representation of the avoidable conflict area $\mathcal{R}(s_{safe,1}, s_{safe,2})$ forms an ellipsoid. The axes of the ellipsoid are the computed values of $s_{safe,1}, s_{safe,2}$. Thus, the guaranteeing of the collision avoidance can be evaluated through a graphical representation on the state-space S_1, S_2, see an illustration on Figure 7.

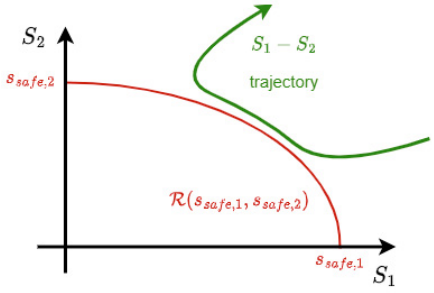


Fig. 7. Illustration on the collision avoidance constraint.

Example on the safe motion profile design

The effectiveness of the motion profile design through the simulation example in Section 2 is illustrated. The RL-based training process is performed on the formulated model of the environment. The achieved cumulative reward value in each episode is illustrated in Figure 8. The illustration shows that the maximization of the reward can be achieved, which is the main goal of the applied DDPG algorithm.

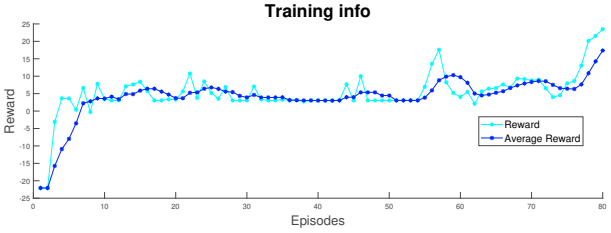
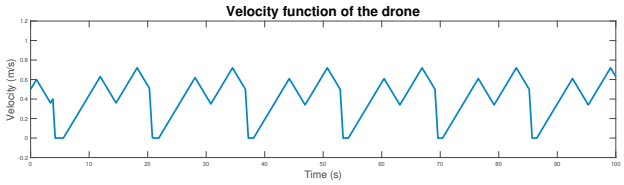


Fig. 8. Reward gains of the learning algorithm in training.

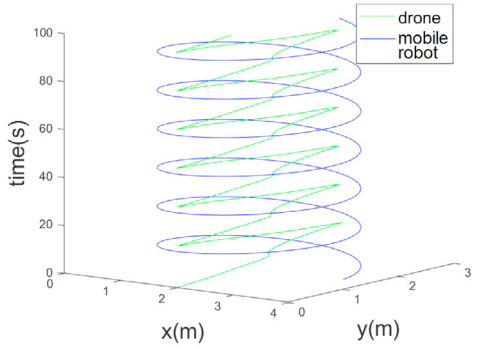
The simulation results on the example with the drone and the mobile robot can be seen in Figure 9. In the example the drone continuously moves along its path forward and backward, while the mobile robot moves on its circle-shaped route, see Figure 1(a). The velocity profile of the drone in in Figure 9(a) shows that the drone has to stop sequentially to avoid collision to the mobile robot. The trajectories of the drone and the mobile robot during the simulation can be found in Figure 9(b), i.e., the collision is avoided. It is also confirmed through the illustration on 9(c), which shows the $S_1 - S_2$ trajectory to one of the selected conflict area.

4. CONCLUSIONS

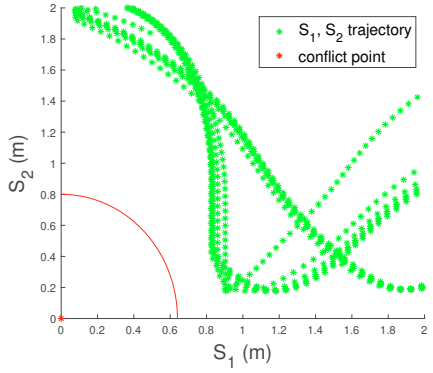
The presented results on the simulation-based evaluation of the method show that the safe motion of the drone within an environment with moving mobile robot can be guaranteed. The drone velocity profile is continuously adapted to the actual motion of the mobile robot in the conflict areas, i.e., arriving to one conflict area the drone



(a) Velocity of the drone



(b) Drone and mobile robot trajectories



(c) Illustration on $S_1 - S_2$ state-space

Fig. 9. Drone motion in conflict situations.

is stopped, but to the another one the drone velocity is only reduced. Thus, the drone is not stopped unnecessarily that leads to safe and high performance motion. The safe motion is guaranteed in spite of the noisy information of the covered areas, due to the proposed Mahalanobis-distance-based filtering algorithm.

REFERENCES

Bui, V.D., Shirakawa, T., and Sato, H. (2022). Autonomous unmanned aerial vehicle flight control using multi-task deep neural network for exploring indoor environments. *SICE Journal of Control, Measurement, and System Integration*, 15(2), 130–144.

Chhikara, P., Tekchandani, R., Kumar, N., Chamola, V., and Guizani, M. (2021). DCNN-GA: A Deep Neural Net Architecture for Navigation of UAV in Indoor Environment. *IEEE Internet of Things Journal*, 8(6), 4448–4460.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 2829–2838. JMLR.org.

Hartigan, J.A. and Wong, M.A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. Series C*, 28(1), 100–108.

Hegedűs, T., Fényes, D., Németh, B., and Gáspár, P. (2023). Cooperation strategy for optimal motion of aerial and ground vehicles. In *2023 31st Mediterranean Conference on Control and Automation (MED)*, 19–24.

Hu, Y., Chen, M., Saad, W., Poor, H.V., and Cui, S. (2021). Distributed multi-agent meta learning for trajectory design in wireless drone networks. *IEEE Journal on Selected Areas in Communications*, 39(10), 3177–3192.

Kodinariya, T.M., Makwana, P.R., et al. (2013). Review on determining number of cluster in k-means clustering. *International Journal*, 1(6), 90–95.

Lupascu, M., Hustiu, S., Burlacu, A., and Kloetzer, M. (2019). Path planning for autonomous drones using 3D rectangular cuboid decomposition. In *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, 119–124.

Mac, T.T., Copot, C., Hernandez, A., and De Keyser, R. (2016). Improved potential field method for unknown obstacle avoidance using uav in indoor environment. In *2016 IEEE 14th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 345–350.

Németh, B. and Gáspár, P. (2023). Hierarchical motion control strategies for handling interactions of automated vehicles. *Control Engineering Practice*, 136, 105523.

Padhy, R.P., Verma, S., Ahmad, S., Choudhury, S.K., and Sa, P.K. (2018). Deep neural network for autonomous uav navigation in indoor corridor environments. *Procedia Computer Science*, 133, 643–650.

Sabetghadam, B., Cunha, R., and Pascoal, A. (2022). Trajectory generation for drones in confined spaces using an ellipsoid model of the body. *IEEE Control Systems Letters*, 6, 1022–1027.

Salzmann, T., Kaufmann, E., Arrizabalaga, J., Pavone, M., Scaramuzza, D., and Ryll, M. (2023). Real-time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms. *IEEE Robotics and Automation Letters*, 1–8.

Spica, R., Cristofalo, E., Wang, Z., Montijano, E., and Schwager, M. (2020). A real-time game theoretic planner for autonomous two-player drone racing. *IEEE Transactions on Robotics*, 36(5), 1389–1403.

Szmuk, M., Pascucci, C.A., Dueri, D., and Açikmeşe, B. (2017). Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4862–4868.

Tompos, D. and Németh, B. (2023). Safe trajectory design for indoor drones using reinforcement-learning-based methods. In *2023 IEEE 17th Int. Symposium on Applied Computational Intelligence and Informatics (SACI)*, 27–32.

Xiang, S., Nie, F., and Zhang, C. (2008). Learning a mahalanobis distance metric for data clustering and classification. *Pattern Recognition*, 41(12), 3600–3612.

Yang, M., Jeon, S.W., and Kim, D.K. (2020). Optimal Trajectory for Curvature-Constrained UAV Mobile Base Stations. *IEEE Wireless Communications Letters*, 9(7), 1056–1059.

Yang, Y., Xiong, X., and Yan, Y. (2023). UAV formation trajectory planning algorithms: A review. *Drones*, 7(1).