Unsupervised Learning based Flexible Framework for Surveillance Planning with Aerial Vehicles

Jan Faigl
Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University
166 27 Prague, Technická 2, Czech Republic
faiglj@fel.cvut.cz

Petr Váňa
Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University
166 27 Prague, Technická 2, Czech Republic
vanapet1@fel.cvut.cz

Robert Pěnička
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University
166 27 Prague, Technická 2, Czech Republic
penicrob@fel.cvut.cz

Martin Saska
Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University
166 27 Prague, Technická 2, Czech Republic
saskam1@fel.cvut.cz

Abstract

The herein studied problem is motivated by practical needs of our participation in the Mohamed Bin Zayed International Robotics Challenge (MBZIRC) 2017 in which a team of Unmanned Aerial Vehicles (UAVs) is requested to collect objects in the given area as quickly as possible and score according to the rewards associated with the objects. The mission time is limited, and the most time-consuming operation is the collection of the objects themselves. Therefore, we address the problem to quickly identify the most valuable objects as surveillance planning with curvature-constrained trajectories. The problem is formulated as a multi-vehicle variant of the Dubins Traveling Salesman Problem with Neighborhoods (DTSPN). Based on the evaluation of existing approaches to the DTSPN, we propose to use unsupervised learning to find satisfiable solutions with low computational requirements. Moreover, the flexibility of unsupervised learning allows considering trajectory parametrization that better fits the motion constraints of the utilized hexacopters that are not limited by the minimal turning radius as the Dubins vehicle. We propose to use Bézier curves to exploit the maximal vehicle velocity and acceleration limits. Besides, we further generalize the proposed approach to 3D surveillance planning. We report on evaluation results of the developed algorithms and experimental verification of the planned trajectories using the real UAVs utilized in our participation in MBZIRC 2017.

1 Introduction

The surveillance planning problem studied in this paper is motivated by practical needs of our participation in the *Mohamed Bin Zayed International Robotics Challenge* (MBZIRC) 2017 (MBZIRC, 2017; Saska, 2017). In particular, in our effort towards the Challenge 3, where a team of *Unmanned Aerial Vehicles* (UAVs) is requested to search and collect objects of interest located in a specified arena. Placement of the objects is not known a priori, and therefore, a quick scan of the whole area is performed at a high altitude to provide

a rough estimation of the possible object locations with particular preference of false positives rather than false negatives. Then, a group of up to three UAVs is requested to verify the objects and identify the reward associated with them to prefer collecting the most rewarding objects for achieving a high total score, see Fig. 1 with snapshots from our preparation experiments. The particular problem addressed in this paper is the trajectory planning to identify the objects of interest that is considered as the surveillance planning with known target locations provided from the first overview scans of the area.

Figure 1: A snapshot of three UAVs following the planned trajectories in a validation of the objects of interest (left) and detail of the used object of interest (right) in the preparation phase towards the Challenge 3

The time for the whole mission is limited, and the most time-consuming part is the pickup and delivery of the objects; hence, the UAVs have to quickly visit the expected locations of the objects and confirm the object location and its reward or reject false positive estimates. Hence, it is desirable to spend as little time as possible in this verification part of the mission. Moreover, regarding the size of the arena, which is in tens of meters, and velocity of the UAVs that fly up to 5 $\mathrm{m.s}^{-1}$, it is preferable to do not spend too much time by planning the trajectories for objects identification as the UAV can travel a significant distance in any additional second spent in planning. Therefore, it has been requested to develop a surveillance planning algorithm with low computational requirements while still be able to provide solutions of satisfiable quality. Thus, our initial intention was to provide a cost-efficient solution in less than one second using a single core of a conventional computer with a CPU of the iCore7 class running at the frequency around 3.4 GHz, i.e., computational resources available at our UAVs (Spurný et al., 2018).

Surveillance planning as finding a cost-efficient trajectory to visit a set of locations can be addressed as a solution of the *Traveling Salesman Problem* (TSP) which is a well-studied problem of combinatorial optimization, for which several computationally efficient heuristic algorithms have been developed (Applegate et al., 2007; Helsgaun, 2000). Regarding trajectory planning for a team of vehicles, such that the total time required to validate all possible object locations is minimized, it is necessary to consider the m-TSP approaches that directly minimize the longest tour length, i.e., the *minmax* variant of the m-TSP (Bektas, 2006). Notice, the problem where the sum of the lengths (*minsum*) of all tours is minimized can be addressed by a transformation of the m-TSP to the single vehicle TSP using (Bellmore and Hong, 1974); however, such solutions are of poor quality as they can contain degenerative solutions with zero tour lengths for particular vehicles. Therefore, it is necessary to address the *minmax* m-TSP directly.

Moreover, when planning trajectories for UAVs it is suitable to provide smooth trajectories even for our hexacopter UAVs utilized in MBZIRC 2017 because the low-level trajectory following controller can more precisely navigate the vehicle along the planned path (Báča et al., 2016). An example of the trajectory following performance is shown in Fig. 2. Therefore a curvature-constrained path is desirable to enable fast motion with the maximal forward velocity and precise trajectory following rather than paths with sharp turns that can be found as a solution of the regular Euclidean TSP.

A suitable kinematic model widely used for the UAVs is the Dubins vehicle for which the curvature-constrained TSP becomes the Dubins TSP (DTSP) (Savla et al., 2005) and we further call the multi-vehicle

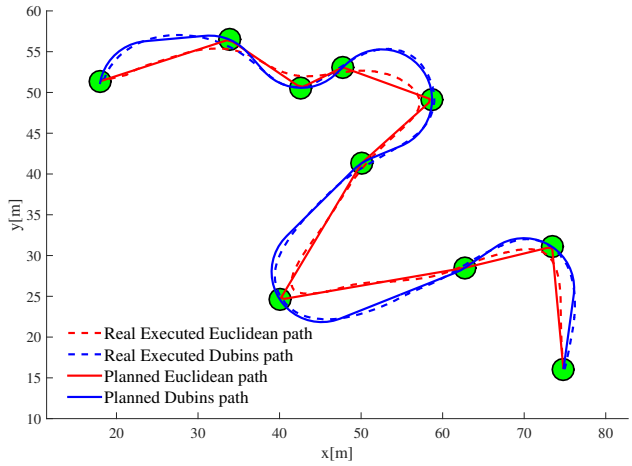


Figure 2: An example of the planned paths and their real execution by the used MPC-based controller for trajectory following (Báča et al., 2016)

problem for m vehicles as the m-DTSP. In addition, it is sufficient to visit proximity of the expected object location to capture the object by a camera sensor with a particular field of view, and thus it is sufficient to reach the object location at the specific sensing range δ to reliably detect the object of interest. Hence, the problem can be formulated as the DTSP with Neighborhoods (DTSPN) (Obermeyer, 2009; Oberlin et al., 2010; Isaacs et al., 2011) and its multi-vehicle variant is denoted the m-DTSPN (Macharet et al., 2013).

For the Dubins vehicle model with the minimal turning radius ρ, the forward velocity is assumed to be constant, and thus the required time to complete the surveillance mission is proportional to the longest tour. Besides, we can consider smaller ρ which requires lower velocity in turning parts of the path, but the vehicle can accelerate and then decelerate on straight line segments to achieve the required velocity in turns. The vehicle can eventually finish the mission sooner than for a high but a constant forward velocity and longer ρ. In general, smooth trajectories can be parametrized, e.g., by B-Splines (Neubauer and Müller, 2015) or Bézier curves (Yang and Sukkarieh, 2010), and the trajectory curvature can be then utilized with the maximal vehicle velocity and acceleration to determine velocity profile along the trajectory from which the *Travel Time Estimation* (TTE) can be computed. Thus, the herein addressed problem is to determine m trajectories to visit the given set of n object locations such that the longest time to travel the particular trajectory is minimized, and it is allowed to visit the location in δ distance, i.e., the problem is formulated as a variant of the m-DTSP for $\delta = 0$ and as the m-DTSPN for $\delta > 0$.

1.1 Focus of the Proposed Approaches and Contributions

The motivation and practical needs of the surveillance planning deployed in the robotic competition steered our effort towards a suitable solution of the m-DTSPN instances arising from MBZIRC 2017. Therefore, we focused on the development of surveillance planning framework that is capable of providing a feasible solution for a typical scenario of MBZIRC 2017 with up to three vehicles and around twenty object locations relatively sparsely placed in the arena around 80 m $\times$ 60 m large. In addition, the required computational time of the planning should be significantly shorter than the time to travel across the arena, and at best, it should be around one second, and it should not exceed 60 seconds. Thus, heuristic algorithms providing solutions of satisfiable quality are preferred than a computationally demanding optimal solution of the DTSP, which is known to be NP-hard (Le Ny et al., 2012).

Due to these requirements, the studied and proposed approaches have been evaluated in the scenario called *mbzirc22* with 22 targets with additional up to three starting locations, one for each of three UAVs, to obtain a realistic estimation of the real performance in MBZIRC 2017, see Fig. 3. Although efficient solutions for

Figure 3: Motivational scenario called *mbzirc22* on top of the test field site (about 80 m × 60 m large) used for real experiments

such a relatively small problem may not scale well with the number of vehicles or the number of targets, the practical deployment, and real experimental verification provide realistic validation of the real and time-critical deployment as it is the participation in a robotics competition.

Regarding the particular approaches to the m-DTSP(N), we consider a purely combinatorial optimization approaches already proposed in the literature to address the m-DTSPN and *minmax* variant of the m-TSP. We also consider our previous effort towards surveillance planning with UAVs based on unsupervised learning of the *Self-Organizing Map* (SOM) firstly deployed in a solution of the DTSP in (Faigl and Váňa, 2016) and later generalized for the m-DTSPN in (Faigl and Váňa, 2017). Following the sampling-based approaches of the continuous optimization problem of Dubins planning (Oberlin et al., 2010; Obermeyer et al., 2012), the *Variable Neighborhood Search* (VNS) metaheuristic (Soylu, 2015) is also considered for a direct solution of the *minmax* m-DTSP and its generalization to the m-DTSPN. Besides, an evolutionary-based Memetic algorithm (Zhang et al., 2014) has been selected for a comparison with the proposed solutions. The promising results and very low computational requirements of the SOM-based solution motivate us to further generalize the unsupervised learning for 3D surveillance planning using Bézier curves (Jolly et al., 2009; Yang and Sukkarieh, 2010) and computation of the velocity profile along the planned trajectory using the vehicle velocity and acceleration limits.

Even though the presented work is built on the previous approaches published in the literature, i.e., the VNS for the m-TSP (Soylu, 2015) and SOM-based unsupervised learning for the m-DTSPN (Faigl and Váňa, 2017), they have been further developed to address the m-DTSPN by the VNS-based approach and the SOM-based approach has been generalized to 3D surveillance planning. Therefore, we consider the main contributions of the paper with respect to the existing approaches as follows:

- Deployment of the VNS-based m-TSP solver in the m-DTSPN.
- Fast and efficient initialization for the VNS-based optimization in the m-DTSPN.
- Comprehensive evaluation of the proposed VNS-based solver and the existing Memetic and SOM-based approaches in the *mbzirc22* scenarios of the m-DTSPN with varying number of vehicles.
- Experimental verification of the found trajectories using real UAVs utilized in MBZIRC 2017.

- Generalization of the SOM-based solver to 3D surveillance planning.
- Verification of the feasibility of the found 3D trajectories using real aerial vehicles.
- Since the developed unsupervised learning based solver allows a straightforward extension from the Dubins vehicle model to a Bézier curve or any similar model (e.g., Dubins-Helix model (Wang et al., 2015b)) while the main principles are the same, we consider the proposed planner as a suitable flexible framework for surveillance planning with aerial vehicles.
- Unsupervised learning framework for surveillance planning addressing the m-DTSPN but also the multi-vehicle planning problem where it is requested to quickly find surveillance trajectories considering the maximal vehicle velocities and acceleration limits that better fit the real motion capabilities of multi-rotor UAVs than Dubins vehicle describing curvature-constrained trajectories suitable for fixed-wing vehicles.

The paper is organized as follows. An overview of the related work is presented in the next section. A formal definition of the addressed problems with a brief overview of the Dubins vehicle model is presented in Section 3. Necessary background on the related *Dubins Touring Problem* (DTP) (Faigl et al., 2017) and 3D smooth trajectory parametrization based on Bézier curve is described in Section 4. The proposed VNS-based m-DTSPN solver is introduced in Section 5 and the generalized SOM-based planner to the 3D surveillance planning is presented in Section 6. Reports on empirical evaluation and experimental deployment are presented in Section 7. Conclusion is dedicated to Section 8.

2 Related work

Surveillance planning for an aerial vehicle is usually closely related to the curvature-constrained path planning for which the fundamental work is (Dubins, 1957) where the problem of the optimal planning for a vehicle with the minimal turning radius ρ is studied. In 1957, Dubins showed that the optimal path connecting two states $q_i, q_j \in SE(2)$ (representing the vehicle configurations as two points in the special Euclidean group $SE(2)$) is one of six possible maneuvers that consist of a straight line segment and a part of a circle with the radius ρ. Although a closed-form expression of the optimal path for the Dubins vehicle between two states exists, it is not sufficient to directly solve surveillance planning where a vehicle is requested to collect information from the given set of target locations. It is due to the initially unknown optimal sequence of visits to the targets, and also the particular headings at the target locations are not known. Therefore it is necessary to determine both the sequence and the headings, which can be formulated as the *Dubins Traveling Salesman Problem* (DTSP).

The DTSP can be considered as an extension of the regular TSP for the Dubins vehicle, and thus the path connecting the particular locations are the Dubins maneuvers respecting the minimal turning radius ρ. Similarly to the regular TSP, also the DTSP stands to determine the optimal sequence of visits to the targets, which is a discrete combinatorial problem. However, the DTSP also includes a continuous optimization part in finding the optimal heading of the vehicle at each target location. Each particular heading value can be selected from the interval $[0, 2\pi)$ and every change of a single heading may significantly change the Dubins tour connecting the locations. Therefore, the DTSP can be considered as a more challenging problem than a discrete optimization of the regular TSP, albeit both problems are NP-hard (Le Ny et al., 2012) as the DTSP becomes the regular Euclidean TSP (ETSP) for $\rho = 0$.

Moreover, in the *Dubins Traveling Salesman Problem with Neighborhoods* (DTSPN), it is also required to determine the most suitable waypoint locations from which information about the targets is collected such that the waypoint is at a distance equal or shorter than the given sensing range δ (Isaacs and Hespanha, 2013). Hence, the DTSPN contains two continuous optimization parts in addition to the determination of the sequence of visits to the targets. The first continuous part is the determination of the optimal heading at the waypoints and the second is the determination of the waypoint locations themselves. Therefore

finding optimal solutions of the DTSP or the DTSPN is computationally challenging and approximation algorithms (Oberlin et al., 2010; Ny et al., 2012; Yu, 2015), heuristics (Savla et al., 2005; Váňa and Faigl, 2015; Isaiah and Shima, 2015), and evolutionary (Yu and Hung, 2012) approaches have been proposed.

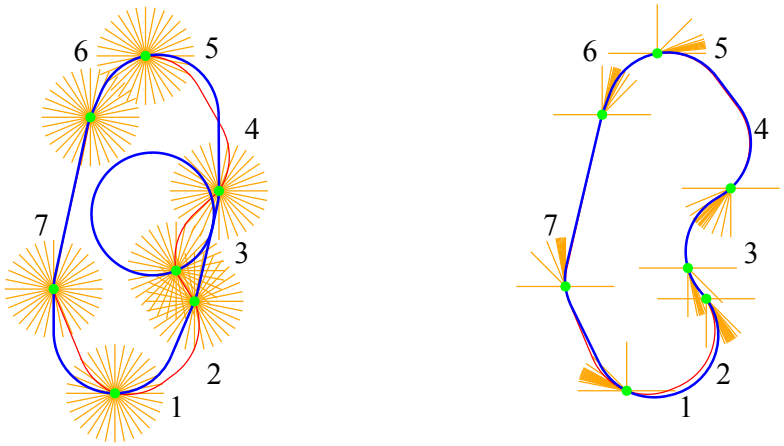
The existing approaches to the DTSP and DTSPN can be categorized into four main classes. The first class represents decoupled approaches where the sequence of the visits to the targets is determined independently on the determination of the headings. The second class is sampling-based methods where a finite discrete set of possible heading values or/and waypoint locations are sampled, and the problem is then transformed into a discrete optimization problem, e.g., the Asymmetric TSP, that can be solved by existing optimal solvers such as Concorde (Applegate et al., 2003) or heuristic algorithms such as LKH (Helsgaun, 2000). The third class of approaches is evolutionary methods that can provide high-quality solutions but are usually computationally very demanding. Finally, the fourth class is the recently proposed unsupervised learning which combines a solution of the sequencing part of the problem with the on-line sampling of the suitable heading values (Faigl and Váňa, 2016) and for the DTSPN also the waypoint locations (Faigl and Váňa, 2017). Selected approaches of the particular classes are briefly described in the rest of this section to support our selection of the considered methods in our effort towards a suitable solution for a practical deployment motivated by MBZIRC 2017.

One of the simplest approaches, that is also computationally very efficient, is the decoupled approach called the *Alternating Algorithm* (AA) proposed by (Savla et al., 2005). The sequence of visits to the targets is determined by a solution of the Euclidean TSP without considering the curvature-constrained path. After that, headings at the waypoints are established in such a way that even edges are connected by straight line segments which prescribe all the headings, and thus odd edges are connected by the optimal Dubins maneuvers that can be computed analytically (Dubins, 1957). The AA has been improved by a randomized adaptive search in (Macharet et al., 2011) and by considering a distance between two consecutive waypoints in the sequence (Macharet and Campos, 2014). Because only two consecutive waypoints in the sequence are considered in these approaches, determination of the headings is computationally very efficient, and for n targets, the computational complexity can be bounded by $O(n)$.

Following the idea of the AA, a receding horizon technique has been utilized in the look-ahead approach proposed in (Ma and Castanon, 2006), where the heading at the next waypoint in the sequence is determined according to the three waypoint locations and heading at the previous waypoint. The reported results are better than for the AA which is also reported in (Isaiah and Shima, 2015), where a combination of the k-look-ahead technique is accompanied by a local improvement based on the 2-Opt heuristic (Croes, 1958). However, the authors do not report on the required computational times.

Another promising decoupled approach called the *Local Iterative Optimization* (LOI) algorithm has been proposed in (Váňa and Faigl, 2015) to address the computationally challenging DTSPN. In particular, the proposed approach is focused on problem instances where a distance of the waypoints in the sequence is longer than 4ρ, i.e., the so-called D_4 instances of the DTSP(N). The initial sequence of the visits to the targets is determined as a solution of the ETSP for target locations in their respective neighborhoods. Then, the problem is addressed as a continuous optimization of two variables for each target. The first variable is the waypoint heading and the second variable is for the waypoint location which is considered as a single variable denoting its position on the boundary of the target's neighborhood. Both variables are then iteratively optimized until the solution is not improving. According to the reported results, the LIO algorithm provides almost about 10 % better solutions than the AA while the computational requirements are still around tens or hundreds of milliseconds using a single core of a conventional CPU. LIO has been proposed for the D_4 instances, but it can also be utilized for solving any instance of the DTSP and DTSPN; however, the quality of found solutions depends on the sequence determined as the ETSP, which can be inadequate for dense and mutually close target locations.

The problem of determining the optimal headings at the waypoints for a given sequence of visits to the targets is called the *Dubins Touring Problem* (DTP) in (Faigl et al., 2017) and it has been addressed by several approaches. An optimal solution of the D_4 instances of the DTP has been proposed in (Goaoc et al.,



(a) Uniform sampling – $N = 224$, $\mathcal{L} = 19.8$, $\mathcal{L}_U = 13.8$, $t = 128$ ms

(b) Guided sampling (Faigl et al., 2017) – $N = 128$, $\mathcal{L} = 14.4$, $\mathcal{L}_U = 14.2$, $t = 76$ ms

Figure 4: A solution of the DTSP for a given sequence of the targets (the green disks) with the total number of samples N, final path length $\mathcal{L}$, and lower bound $\mathcal{L}_U$. The found solution is the blue curve, and the red curve is its lower bound determined as a solution of the *Dubins Interval Problem* (DIP) with the cost $\mathcal{L}_U$ (Manyam and Rathinam, 2015). The uniform sampling utilizes 32 heading values per each target. The required computational time is denoted t.

2013). The solution is based on solving a family of n-dimensional convex optimization sub-problems, where n is the number of waypoints in the sequence. The number of sub-problems can be bounded by 2^{2n-2}, which regarding the computational complexity of the whole algorithm is relatively high in comparison to simple heuristics such as the AA (Savla et al., 2005) or LIO (Váňa and Faigl, 2015). Notice, a solution of the DTSP with a given sequence of visits to the targets can be easily found as a solution of the DTP for a discrete set of possible heading values at each waypoint, see Fig. 4a and a description of the forward search procedure in Section 4.1.

A very important result on the tight-lower bound of the DTP has been proposed in (Manyam and Rathinam, 2015) which has been further evaluated in (Manyam et al., 2016), but unfortunately without reporting the computational requirements. The computation of the tight-lower bound is based on the solution of the so-called the *Dubins Interval Problem* (DIP) introduced in (Manyam et al., 2015). DIP is a variant of the Dubins planning between two waypoints, i.e., locations with the prescribed headings. In DIP, the heading at the waypoint is not a single value but an interval. Thus, for the interval of the full range 2π, the solution of DIP is a straight line segment connecting the locations. The tight-lower bound (Manyam and Rathinam, 2015) has been utilized to guide sampling of the possible heading intervals and the heading values themselves in (Faigl et al., 2017), where the authors show improved results over a uniform sampling of the heading, see Fig. 4 for an example of the DTSP solution based on the DTP and DIP.

Transformation (or also sampling-based) methods represent the second class of the approaches to the DTSP(N). Similarly to the aforementioned discretization of the headings in the DTP, these methods consider a finite set of discrete heading values at each waypoint location or a set of possible locations in the case of the DTSPN. Then, the optimal Dubins maneuvers between all possible pairs of the waypoint locations are computed to build a complete graph representing the original problem, which can be solved by combinatorial graph-based solvers.

One of the first sampling-based and resolution complete approaches to the DTSPN has been proposed in (Obermeyer et al., 2010). In this approach, the DTSPN is transformed into the Generalized TSP (GTSP) where the targets with their neighborhoods are represented by mutually exclusive finite sets of nodes. The

GTSP is then transformed into the Asymmetric TSP (ATSP) because the optimal Dubins maneuvers between two states depend on the path direction. Such a transformed problem is solved by the LKH algorithm (Helsgaun, 2000). Even though the LKH algorithm is one of the most powerful heuristics for the TSP, due to the samples and transformation, the final problem has many nodes. The reported computational times for problems with 20 targets and 1500 random samples are several hundreds of seconds (Obermeyer et al., 2010), which is reported to be faster than the genetic algorithm for the DTSPN proposed by the same authors in (Obermeyer, 2009), but still far from our needs and expectations.

A comparison of the DTSPN approaches is provided in (Macharet et al., 2012) where significant improvement of the solution quality is reported for evolutionary techniques. A memetic algorithm for the DTSPN with the disk-shaped neighborhoods and relaxed terminal heading is proposed in (Zhang et al., 2014). The superior solution quality is reported for the memetic algorithm in the DTSPN instances with ten targets. The reported computational times are 8.3 seconds for ten targets and 45.5 seconds for problems with 17 targets. A genetic algorithm for the DTSPN with polygonal goals has been proposed in (Obermeyer, 2009) but the real computational requirements are not reported. However, the same authors report that their sampling-based approach proposed in (Obermeyer et al., 2010) requiring hundreds of seconds is faster than the genetic algorithm (Obermeyer, 2009).

The recently proposed unsupervised learning method for the DTSP (Faigl and Váňa, 2016) is based on an evolution of the growing *Self-Organizing Map* (SOM) for the TSP (Faigl, 2018). The input layer of the two-layered neural network servers for presenting the input signals which are the target locations. The neurons' weights represent locations in the input space, and the output layer is an array of nodes representing the waypoints. Since the output layer is one dimensional and the nodes are organized in an array, it forms a ring of neurons that directly represents a TSP tour. In (Faigl and Váňa, 2016), a possible heading value at the target is determined in the selection of the best matching neuron to the target location presented to the network. Besides, additional heading values are associated with the winner neuron which is adapted towards the presented target, i.e., its weights are moved towards the target location in the input space. The adaptation of the network is performed in learning epochs in which all targets are presented to the network. The weight of the adaptation is decreased after each epoch according to a cooling schedule, and the network is stabilized in hundreds of epochs. However, a solution of the DTSP is determined as a solution of the DTP represented by the winner neurons of the current epoch, where the ring of nodes prescribes the sequence of visits to the targets and the particular headings are the associated headings to the neurons. Thus, a solution is available after each learning epoch, and the final solution is found as the best-found solution among all the learning epochs. The reported results are better than the solutions provided by the Memetic algorithm (Zhang et al., 2014) with the computational time restricted to 100 seconds while the SOM needs less than 30 seconds for problems with up to 100 targets.

The SOM-based algorithm (Faigl and Váňa, 2016) has been significantly improved in (Faigl and Váňa, 2017), where the reported required computational time for scenarios (motivated by MBZIRC 2017) with 22 targets is found in less than 600 milliseconds, while the solutions are better than those provided by the Memetic algorithm (Zhang et al., 2014) with the computational time restricted to 10 seconds. Moreover, the SOM-based approach has been generalized for the DTSPN, where the particular waypoint locations are determined during the winner selection together with the expected heading at the waypoint. In addition, the m-DTSPN is addressed by creating an individual neural network for each vehicle, and during the winner selection, neurons from the network which represents a shorter tour are preferred to address the *minmax* variant of the m-TSP (Somhom et al., 1999).

Regarding approaches for the m-DTSPN, they are similar to the m-TSP in many ways (Bektas, 2006; Oberlin et al., 2010), especially the transformation/sampling-based solvers, but only a few approaches directly address the challenges of the *minmax* variant of the m-DTSPN. One of them is the memetic algorithm (Zhang et al., 2014), which has been compared with the additional direct approach based on SOM in (Faigl and Váňa, 2017). Another evolutionary based approach to the *minmax* variant of the m-DTSPN has been proposed in (Macharet et al., 2013), but the authors do not report on the computational requirements, which also hold for the improved version presented in (Macharet et al., 2016).

Having a transformed problem with a graph representation, graph-based m-TSP approaches may be considered. The *minmax* variant of the m-TSP has been addressed by (França et al., 1995) where exact algorithms are proposed. In (Kulich et al., 2004), the authors compare genetic algorithm, ant colony optimization, and SOM-based solver in m-TSP scenarios arising from rescue missions, where the superior results are provided by SOM. In addition to the soft-computing techniques, a general metaheuristic called the *Variable Neighborhood Search* (VNS) proposed by (Hansen and Mladenović, 2001) has been applied to the *minmax* m-TSP in (Soylu, 2015).

Regarding the presented overview of the existing methods for the DTSPN and more specifically to the m-DTSPN. We consider the Memetic algorithm (Zhang et al., 2014) and SOM-based approach (Faigl and Váňa, 2017) as the most promising because the Memetic algorithm is capable of providing a high-quality solution, and thus it may represent a suitable reference approach. On the other hand, the SOM-based approach has the computational requirements lower than the desired one second while it also provides better solutions than the simple heuristics AA and LIO (Faigl and Váňa, 2016). Besides, solutions of the m-DTSPN are reported for both the Memetic and SOM-based algorithms and both approaches are any-time as they provide the first solution very quickly, which is also desirable property for a practical deployment under real-time constraints.

In addition, we also included sampling-based approach in our evaluation to cover purely combinatorial optimization approaches which work on some finite discrete set of possible heading values and waypoint locations. In this case, we consider the VNS method (Soylu, 2015) as a particularly interesting method. First, it directly addresses the *minmax* m-TSP, and it improves the initial solution if more computational time is available. Besides, the VNS metaheuristic has been recently successfully deployed in a solution of the closely related problem of the surveillance planning called the *Dubins Orienteering Problem* (DOP) (Pěnička et al., 2017) which has been further generalized to the DOP with Neighborhoods in (Pěnička et al., 2017). Therefore, we consider VNS as a promising method for the sampling-based approach to the herein addressed m-DTSPN. However, an initial solution of the m-DTSPN is needed for the VNS-based optimization which is addressed by a newly proposed procedure described in Section 5.

In addition to the Dubins vehicle model, which is a suitable model for rotary vehicles because it provides smooth trajectories with a constant speed, we are interested also in other types of trajectory parametrization because the motion of the rotary UAV is limited mainly by the maximum speed and acceleration, and the minimal turning radius is not defined. Various types of curves such as splines (Lepetič et al., 2003), polynomial functions (Papadopoulos et al., 2005), and Bézier curves (Jolly et al., 2009) can be utilized for continuous and smooth path generation (Wang et al., 2015a) for which the final trajectory with the velocity profile is computed according to the maximum possible velocity and acceleration of the vehicle. Moreover, we are also interested in the generalization of the surveillance planning with curvature-constrained paths from the 2D environment representation to 3D. An extension of the Dubins vehicle for the 3D is possible using Dubins-Helix method (Wang et al., 2015b) or using the so-called *Dubins Airplane model* proposed in (Chitsaz and LaValle, 2007) to address the bounded curvature and also limited pitch angle of real UAVs, especially fixed-wing vehicles. The Dubins Airplane model has been used for solving the 3D-DTSPN (Váňa et al., 2018) with fixed-wing vehicle. However, hexacopters are used in our motivational problem, and therefore, we consider Bézier curves (Yang and Sukkarieh, 2010) that can describe trajectories of arbitrary curvature and are specified only by four control points. Thus trajectory parametrization based on Bézier curves is selected as a suitable generalization of the proposed surveillance planning framework to directly find smooth trajectories for a team of UAVs in 2D but also in 3D scenarios.

The SOM-based approach (Faigl and Váňa, 2017) has been selected as a suitable optimization framework for the generalized surveillance planning with Bézier curves because of two main reasons. The first reason is related to the expected increased computational requirements related to the optimization of Bézier curves that is more demanding than the analytical solution of Dubins maneuvers, and regarding the reported results, SOM is computationally efficient. Besides, the unsupervised learning principles used in SOM are flexible to relatively straightforwardly utilize different parametrization of the curves. Therefore, the proposed unsupervised learning based 3D surveillance planning framework for the m-DTSPN is presented in Section 6.

Table 1: Summary of the Existing Methods for Surveillance Planning with Aerial Vehicles.

Approach	**Method type**	**Sampling**	**Trajectory optimization**	**Any-time**	**Neighborhoods**	**Multiple vehicles**	**3D Trajectory**	**Any-curvature**	**Computational requirements*
(Savla et al., 2005)	Decoupled								low
(Ma and Castanon, 2006)	Decoupled								low
(Obermeyer, 2009)	Evolutionary			√	√				moderate
(Obermeyer et al., 2010)	Transformation	√			√				moderate
(Oberlin et al., 2010)	Transformation	√			√	√			high
(Macharet et al., 2011)	Decoupled		√						low
(Macharet et al., 2012)	Evolutionary			√	√				high
(Le Ny et al., 2012)	Decoupled								low
(Yu and Hung, 2012)	Evolutionary			√					high
(Macharet et al., 2013)	Evolutionary			√	√	√			high
(Zhang et al., 2014)	Evolutionary		√	√	√	√			high
(Macharet and Campos, 2014)	Decoupled		√						low
(Váňa and Faigl, 2015)	Decoupled		√		√				low
(Isaiah and Shima, 2015)	Decoupled								low
(Manyam et al., 2015)	Transformation	√			√				moderate
(Macharet et al., 2016)	Transformation	√			√	√			moderate
(Faigl and Váňa, 2016)	Unsupervised learning			√					low
(Faigl and Váňa, 2017)	Unsupervised learning			√	√				low
(Váňa et al., 2018)	Decoupled		√		√		√		low
Proposed methods									
VNS – Section 5, 2018	Transformation	√		√	√	√			low
SOM Dubins – Section 6.1, 2018	Unsupervised learning		√	√	√	√			low
SOM Bézier – Section 6.2, 2018	Unsupervised learning		√		√	√	√	√	moderate

*Computational requirements are considered *low* if a satisfiable solution (for $n \approx 20$, i.e., *mbzirc22* scenario, see Section 7) is found in less than 1 second and *moderate* in less than 60 seconds using conventional computational resources; otherwise, the requirements are considered *high*.

A summary and evolution of the existing approaches together with the herein proposed methods for solving variants of the DTSP is presented in Table 1 with an indication of their particular properties and capabilities. Besides, we further distinguish if the approach performs continuous trajectory optimization, which may further improve the solution. The transformation methods perform sampling, and thus they transform the problem to the combinatorial optimization. On the other hand, the decoupled approaches firstly determine the sequence of visits and then generate the requested trajectories where the recent approaches employ continuous optimization of the headings and possibly also the locations of the waypoints. The unsupervised learning is similar to the decoupled approaches in the trajectory optimization. However, the continuous trajectory optimization is performed during the solution of the sequencing part that is the main difference to decoupled and transformation approaches and makes it similar to evolutionary methods, but the convergence of the learning is much faster than finding satisfiable solutions by, e.g., memetic algorithms.

3 Problem Statement

The studied surveillance planning problem is motivated by the MBZIRC 2017 competition where it is needed to identify possible objects of interest as quickly as possible by three aerial vehicles. The problem is considered as surveillance planning where a team of m vehicles is requested to take a camera snapshot of the objects using non-zero sensing range δ to save the travel cost. Moreover, due to the employed *Model Predictive Control* (MPC) trajectory following (Báča et al., 2016), the surveillance trajectories have to fit the vehicle motion constraints to allowing fast and precise motion of the vehicle along the trajectory. Thus, the problem is to determine a sequence of visits to the object locations for each vehicle together with the corresponding trajectories connecting the waypoints from which objects are captured such that all the objects are identified as quickly as possible, and the vehicles return to their initial locations. The expected computational requirements for the surveillance planning and the specific setup of the MBZIRC 2017 deployment allow to relax the collision avoidance in the planning part, and it is addressed by the reactive collision avoidance implemented in the employed MPC-based trajectory following controller (Spurný et al., 2018; Báča et al., 2018). Therefore, an explicit finding of collision-free trajectories is not considered in the following problem formulations.

First, the problem is formulated as the m-DTSPN in which m curvature-constrained paths (one path for each vehicle) for the Dubins vehicle with the minimal turning radius ρ are found such that each of the given n target locations is visited by at least one of the planned path in the distance not exceeding the sensing range δ and the length of the longest path is minimized. In addition to ρ, the utilized Dubins vehicle model (Dubins, 1957) assumes the constant forward velocity v and the state q of the Dubins vehicle is described as a triplet $q = (x, y, \theta)$, where $p = (x, y)$ is the vehicle position in the plane $p \in \mathbb{R}^2$ and θ is the vehicle heading at p and $\theta \in \mathbb{S}^1$, i.e., $q \in SE(2)$. The motion of the vehicle is described as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = v \begin{bmatrix} \cos\theta \\ \sin\theta \\ u \cdot \rho^{-1} \end{bmatrix}, \quad |u| \leq 1, \tag{1}$$

where u is the control input.

The team of UAVs consists of m identical vehicles with the same ρ allowing the constant, maximal, and safe forward velocity while the error of the trajectory following is acceptable to capture the object of interest at the target location from the determined waypoint location within the δ sensing range from the target location. In fact, the real field of view of the utilized camera is wider than δ used for planning such that the employed MPC-based controller (Báča et al., 2016) follows the trajectory with the error less than the difference of the real field of view and δ, and thus it is assured that the object of interest can be identified from the snapshot taken at the particular waypoint location.

Each vehicle (denoted r) starts at its individual initial location $p_d^r \in \mathbb{R}^2$ (further denoted as depot) and the requested path for the r-th vehicle terminates at the same location p_d^r, i.e., we are searching for m closed trajectories. The trajectories consist of a sequence of Dubins maneuvers connecting the determined waypoints. Thus, two consecutive waypoints in the sequence q_i and q_{i+1} both from $SE(2)$ are connected by one of the six Dubins maneuvers respecting the kinematic constraints of the Dubins vehicle (1).

In the DTSPN with a single vehicle, the goal is to find the shortest trajectory to take a snapshot of all n objects of interest $\mathcal{O} = \{o_1, \ldots, o_n\}$. For simplicity and readability, we consider o_i be the target location of the object i, i.e., $o_i \in \mathbb{R}^2$. Since it is allowed to collect information about o_i within δ distance, we need to determine for each o_i a waypoint location p_i such that $|(p_i, o_i)| \leq \delta$. Besides, for each waypoint location p_i we need to determine the heading θ_i and for the all waypoints, we search for a sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ of the waypoints $q_i = (p_i, \theta_i)$ such that the sum of the lengths of the Dubins maneuvers connecting the waypoints in the sequence Σ is minimal.

Problem 3.1 (DTSPN)

$$
\begin{aligned}
&\textit{minimize}_{\ P,\Theta,\Sigma} && \mathcal{L}(Q,\mathcal{O}) = \sum_{i=1}^{n} \mathcal{L}(q_{\sigma_{i-1}}, q_{\sigma_i}) + \mathcal{L}(q_{\sigma_n}, q_{\sigma_0}) \\
&\textit{subject to} && Q = (q_{\sigma_1}, \ldots, q_{\sigma_n}),\ q_{\sigma_i} = (p_{\sigma_i}, \theta_{\sigma_i}),\ q_{\sigma_i} \in SE(2) \\
& && P = (p_{\sigma_1}, \ldots, p_{\sigma_n}),\ p_{\sigma_i} \in \mathbb{R}^2 \text{ and } |(p_{\sigma_i}, o_i)| \leq \delta \text{ for } o_i \in \mathcal{O}\ , \\
& && \Theta = (\theta_{\sigma_1}, \ldots, \theta_{\sigma_n}),\ 0 \leq \theta_{\sigma_i} < 2\pi \\
& && \Sigma = (\sigma_1, \ldots, \sigma_n),\ 1 \leq \sigma_i \leq n \text{ and } \sigma_i \neq \sigma_j \text{ for } i \neq j \\
& && q_{\sigma_0} \in SE(2) \text{ and } \mathcal{P}(q_d) = p_d \text{ is the vehicle depot}
\end{aligned}
\tag{2}
$$

where $\mathcal{L}(q_i, q_j)$ is the length of the shortest Dubins maneuver between q_i and q_j computed analytically according to (Dubins, 1957) and $\mathcal{P}(q) = p$ is a projection of the waypoint $q = (p, \theta)$ to $\mathbb{R}^2$, i.e., $p \in \mathbb{R}^2$. Notice, we may further distinguish a single depot location p_d or a depot with a neighborhood defined by the sensing range δ as for other target locations. Since the practical, motivational deployment is for specified initial locations of the vehicles, we focus on depots without the neighborhoods, and $\delta > 0$ for depots is further discussed in the description of the particular methods and empirical evaluation.

For the m-DTSPN, it is requested to find m trajectories $\{Q^1, \ldots, Q^m\}$ satisfying the limited curvature of the Dubins vehicles (1), one for each of m vehicles, such that the length of the longest trajectory is minimal, i.e., the *minmax* variant of the m-DTSPN. An individual trajectory for the r-th vehicle can be considered as a solution Q^r of the DTSPN formulated as Problem 3.1 for a subset of objects of interest $\mathcal{O}^r \subseteq \mathcal{O}$ that are covered along the trajectory Q^r with the length $\mathcal{L}(Q^r, \mathcal{O}^r)$. Besides, the initial location of the vehicle is prescribed by p_d^r. Since a solution of the DTSPN is a closed and continuous trajectory, it is sufficient that p_d^r is a part of Dubins tour; however, for this special waypoint location, the sensing range is individually set to zero. Thus, the m-DTSPN can be formulated as a problem to determine a subset of n^r locations $\mathcal{O}^r$ for each vehicle r, $1 \leq r \leq m$ such that all objects $\mathcal{O}$ are covered, and the length of the longest trajectory is minimal.

Problem 3.2 (m-DTSPN)

$$
\begin{aligned}
&\textit{minimize}_{(Q^r,\mathcal{O}^r) \text{ for } r\in\{1,\ldots,m\}} && \max_{r\in\{1,\ldots,m\}} \mathcal{L}(Q^r, \mathcal{O}^r \bigcup \{p_d^r\}) \\
&\textit{subject to} && Q^r \text{ is a solution of Problem 3.1 for the subset } \mathcal{O}^r \text{ and } p_d^r \\
& && \mathcal{O} = \bigcup_{r=1}^{m} \mathcal{O}^r \text{ and for each } o \in \mathcal{O} \text{ there is} \\
& && q \in \bigcup_{r=1}^{m} Q^r \text{ such that } |(\mathcal{P}(q), o)| \leq \delta
\end{aligned}
\tag{3}
$$

In addition to the Dubins vehicle model (1), the addressed surveillance planning is also considered for a general 3D trajectory satisfying constraints of the utilized hexacopters, i.e., the maximal velocity and acceleration. Such a problem formulation is formally identical to Problem 3.2 except Q^r which needs to be substituted by the parametrization of the trajectory $\mathcal{X}^r$ and the length of the trajectory $\mathcal{L}(Q^r, \mathcal{O}^r)$ needs to be replaced by the *Travel Time Estimation* (TTE) of the trajectory $\mathcal{T}(\mathcal{X}^r, \mathcal{O}^r)$. For simplicity and w.l.o.g., we assume that each object of interest is covered from some point x on the determined trajectories $\mathcal{X}^1, \ldots, \mathcal{X}^r$ and x can be $x \in \mathbb{R}^2$ or $x \in \mathbb{R}^3$ in the case of the 3D trajectory.

Problem 3.3 (Surveillance Planning with a 3D Smooth Trajectory)

$$
\begin{aligned}
&\textit{minimize}_{(\mathcal{X}^r,\mathcal{O}^r) \text{ for } r\in\{1,\ldots,m\}} && \max_{r\in\{1,\ldots,m\}} \mathcal{T}(\mathcal{X}^r, \mathcal{O}^r) \\
&\textit{subject to} && \mathcal{O} = \bigcup_{r=1}^{m} \mathcal{O}^r \text{ and for each } o \in \mathcal{O} \text{ there is} \\
& && q \in \bigcup_{r=1}^{m} \mathcal{X}^r \text{ such that } |(\mathcal{P}(q), o)| \leq \delta.
\end{aligned}
\tag{4}
$$

4 Background

4.1 Dubins Touring Problem (DTP)

An important part of the sampling-based approaches for the DTSP is a solution of the *Dubins Touring Problem* (DTP) (Faigl et al., 2017). The DTP stands to determine the optimal heading values for a given sequence of the waypoint locations, and it can be formally defined as follows. Let the given sequence of n waypoint locations be $P = (p_1, \ldots, p_n)$ and it is requested the vehicle returns to the initial location because of the context of solving the DTSP. The problem is to find the particular heading value at each target location, i.e., the headings $T = (\theta_1, \ldots, \theta_n)$ such that the optimal Dubins maneuvers (Dubins, 1957) connecting the targets in the sequence form a Dubins tour with the minimal length. Thus, the cost function is piecewise continuous, and the DTP is a continuous optimization problem.

Problem 4.1 (DTP)

$$\begin{aligned} &\textit{minimize}_{\,T} && \mathcal{L}(T,P) = \sum_{i=1}^{n-1} \mathcal{L}(q_i, q_{i+1}) + \mathcal{L}(q_n, q_1) \\ &\textit{subject to} && q_i = (p_i, \theta_i), \quad p_i \in P, \quad \theta_i \in T, \quad 0 \leq \theta_i < 2\pi, \quad i = 1, \ldots, n \end{aligned} \quad , \tag{5}$$

where $\mathcal{L}(q_i, q_j)$ is the length of the shortest Dubins maneuver between q_i and q_j which can be computed by a closed-form expression (Dubins, 1957).

Sampling-based Solution of the DTP

Having a discrete finite set of possible heading values per each target location in the sequence P, e.g., h heading values for each target, we can construct a graph where nodes represent particular vehicle states and edges represents an optimal Dubins maneuver connecting the states. For a sequence of n targets, the graph has n layers and each layer has up to h nodes, see Fig. 5. Then, the optimal solution of the DTP

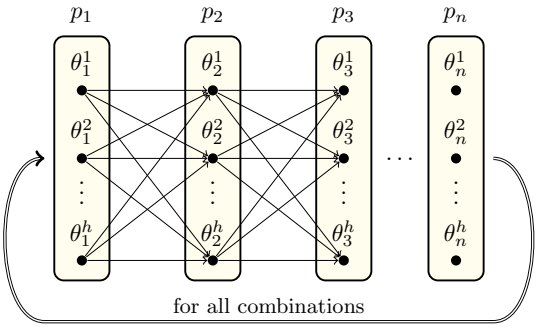


Figure 5: A search graph where each layer corresponds to one target location $p_i \in P$ with particular heading values $\Theta_i = \{\theta_i^1, \ldots \theta_i^h\}$. Two neighboring layers are fully connected by the oriented edges representing the optimal Dubins maneuver between the states.

for the given discretization of the headings can be found by a forward search of the graph. Since we need a closed tour, the graph has to be searched for h initial/termination headings, and thus the overall time complexity of the search procedure can be bounded by $O(nh^3)$.

4.2 3D Smooth Trajectory based on Bézier Curve

The utilized parametrization of the 3D smooth trajectory is based on the cubic Bézier curve that is defined by four control points. The first two control points define the end locations and direction of the curve directly, and two additional points define the departure and terminal tangents. Therefore, Bézier curves can be easily connected into a smooth path from multiple segments. A general Bézier curve of the d-th degree can be parametrized by

$$\mathbf{X}(\tau) = \sum_{i=0}^{d} \mathbf{B}_i J_{d,i}(\tau), \quad 0 \leq \tau \leq 1, \tag{6}$$

where $\mathbf{B}_i$ stands for the control points and $J_{d,i}(\tau)$ is the Bézier polygon of the d-th degree which prescribes weights for the control points $\mathbf{B}_i$ (Bézier, 1973). Since the Bézier polygon is given by

$$J_{d,i} = \binom{d}{i} \tau^i (1-\tau)^i, \tag{7}$$

the parametrization of the utilized cubic Bézier curve in the expanded form can be expressed as

$$\mathbf{X}(\tau) = \mathbf{B}_0(1-\tau)^3 + 3\mathbf{B}_1\tau(1-\tau)^2 + 3\mathbf{B}_2\tau^2(1-\tau) + \mathbf{B}_3\tau^3. \tag{8}$$

Notice, the Bézier curve can be used for a path parametrization in 2D and 3D, the only difference is in the dimension of the control points, i.e., $\mathbf{B}_i \in \mathbb{R}^2$ and $\mathbf{B}_i \in \mathbb{R}^3$, respectively.

Travel Time Estimation (TTE)

Having a parametrization of the trajectory as a sequence of Bézier curves described by (8), the travel time of the vehicle along the trajectory can be computed from the velocity profile for the trajectory. The maximal velocity and acceleration of the utilized vehicles are limited individually for the horizontal movements by the maximal speed v_{horiz} and the maximal acceleration a_{horiz}. Similarly, the maximal speed v_{vert} and the maximal acceleration a_{vert} limit the vertical movements. Regarding these limitations, the vehicle velocity along the given path is adjusted to minimize the travel time of the trajectory, which is further referred as the *Travel Time Estimation* (TTE). The maximal achievable velocity is determined concerning the path curvature and the acceleration limits as follows.

The profile for the vertical velocity is directly computed from the altitude differences along the curve, and thus the first and second derivatives along the $z-$axis are utilized, and the vertical velocity is limited by v_{vert} and a_{vert}. In the horizontal plane, two different acceleration components are affecting the vehicle simultaneously. The first one is the tangent acceleration a_{tan} which is responsible for the speed changes. The second component is the radial acceleration a_{rad} which is caused by the path curvature, but it does not directly influence the vehicle velocity. The tangent and radial accelerations are always perpendicular, and their combined value cannot exceed a_{horiz} which can be expressed as

$$a_{tan}^2 + a_{rad}^2 \leq a_{horiz}^2. \tag{9}$$

The radial acceleration a_{rad} in the horizontal plane is given by

$$a_{rad} = v^2 \, \kappa_h, \tag{10}$$

where κ_h stands for the horizontal curvature of the trajectory, e.g., computed as

$$\kappa_h = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{\frac{3}{2}}}. \tag{11}$$

Notice that for Bézier curves, the curvature has a closed-form expression. From the curvature, the maximal possible velocity v_{pos} of the vehicle along the trajectory can be computed from

$$v_{pos} = \min\left(v_{horiz}, \sqrt{\frac{a_{horiz}}{\kappa_h}}\right). \tag{12}$$

The maximal possible tangent acceleration a_{tan} is then defined by

$$a_{tan}^2 = a_{horiz}^2 - a_{rad}^2 = a_{horiz}^2 - v_{pos}^4 \kappa_h^2. \tag{13}$$

The right side of (13) is always positive because of (12).

Based on these preliminaries, the velocity profile, and thus the TTE can be numerically determined in the following six steps.

1. Sample the parametrized path into a finite set of uniformly sampled points and compute the horizontal curvature of the trajectory (11) at each sample using the first and second derivatives expressed from (8).
2. Set the initial and final vehicle velocity to zero.
3. Determine derivatives along the z-axis and limit the vertical velocity according to the v_{vert} and a_{vert}.
4. Compute v_{pos} for every sampled point of the trajectory (12).
5. Iterate over the samples forward and limit the vehicle velocity and acceleration by the maximum possible tangent acceleration (13), i.e., adjust the travel time between the respective samples.
6. Iterate over the samples backward and limit the vehicle velocity and acceleration by the maximum possible tangent acceleration (13), similarly as in the previous step.

5 Variable Neighborhood Search for the m-DTSPN

The proposed VNS-based solution of the m-DTSPN is based on the existing deployment of the VNS metaheuristic to the m-TSP (Soylu, 2015). The expected locations of the objects of interest $\mathcal{O}$ are considered as target locations in the m-DTSPN and the extension towards the minimal turning radius ρ of the Dubins vehicle model and non-zero sensing range δ is based on sampling possible heading values and waypoint locations. In particular, s locations are uniformly sampled for the neighborhood of each target $o \in \mathcal{O}$ on the circle with the radius δ centered at o. Then, h possible heading values are uniformly sampled for each such a sampled location. Besides, an individual starting location for each vehicle is considered, which better corresponds with the practical deployment in the surveillance planning contrary to a common depot utilized in (Soylu, 2015). Therefore a modified initialization of the VNS-based solver is proposed to support the individual starting locations.

The VNS metaheuristic consists of two main procedures: the *shake* and *local search*. The *shake* procedure is used to get the currently best incumbent solution x from possible local optima by changing it randomly to a solution x' within the neighborhoods $\{N_1, \ldots, N_{k_{max}}\}$. On the other hand, the *local search* procedure searches fully specific neighborhoods of a solution x' using l_{max} predefined operators to find a possibly better incumbent solution, which in the addressed *minmax* variant of the m-DTSPN, is the one with a smaller the longest tour. The utilized procedures are detailed below, and a summary of the VNS-based solver for the m-DTSPN is in Algorithm 1.

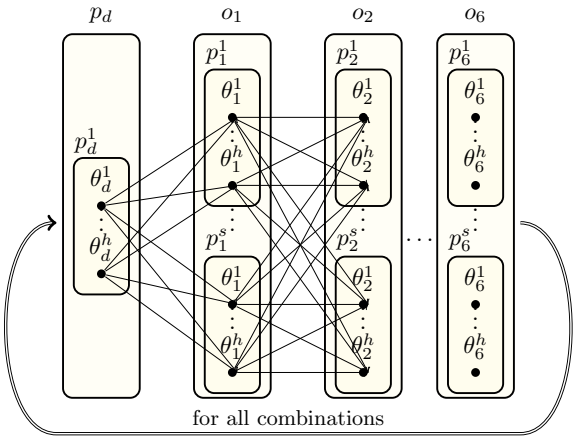
The VNS-based solution to the m-DTSPN uses static sampling. Therefore all possible Dubins maneuvers are precomputed, and all the lengths are stored in a distance matrix to reduce the computational burden

Algorithm 1: VNS-based solver for the m-DTSPN

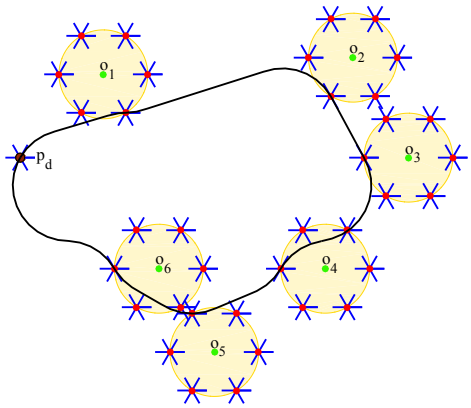
Input : $\mathcal{O}$ – A given set of objects of interest
Input : $(p_d^1, \ldots, p_d^m)$ – The requested initial locations (depots) for the m vehicles
Parameter : ρ – The minimal turning radius
Parameter : δ – The sensing range
Parameter : h – The number of heading samples
Parameter : s – The number of samples of the waypoint locations
Parameter : l_{max} – The number of local search neighborhood operators
Parameter : k_{max} – The number of shaking neighborhoods
Output : $x = \{Q^1, \ldots, Q^m\}$ – The found Dubins tours for the m vehicles

$\mathcal{O}' \leftarrow$ sampleWaypoints$(\mathcal{O}, (p_d^1, \ldots, p_d^m), s, h, \delta, \rho, m)$
$x \leftarrow$ initialization$(\mathcal{O}')$
while $m > 1$ and *stopping condition is not met* **do**
  $k \leftarrow 1$
  while $k \leq k_{max}$ **do**
    $x' \leftarrow$ shake$(x, k, \mathcal{O}')$
    $x'' \leftarrow$ localSearch$(x', l_{max}, \mathcal{O}')$
    if $\max_{Q^r \in x} \mathcal{L}(Q^r) < \max_{Q^r \in x''} \mathcal{L}(Q^r)$ **then**
      $x \leftarrow x''$
      $k \leftarrow 1$
    else
      $k \leftarrow k + 1$
return x

during the VNS optimization. Thus, a solution of a single vehicle for the prescribed visits to the targets can be determined in a similar way as finding a Dubins tour in the DTP, just instead of h possible states per each target, sh states are considered, see the extended search graph in Fig. 6.



(a) A search graph with h heading samples per each neighborhood sample $p_1^i, \ldots, p_s^i$ for each target location o_i. In this case, a fixed initial vehicle location (depot) p_d is considered without the neighborhood.



(b) An example of a solution of the DTSPN with disk-shaped neighborhood (yellow) around each target location o_i with uniform sampling of $s = 6$ waypoint locations (red) inside the neighborhood and $h = 6$ heading samples at each possible waypoint location visualized as blue segments.

Figure 6: A search graph utilized in the proposed VNS-based approach to the m-DTSPN with an example of the found solution

Besides, a dynamic programming technique is utilized for storing the particular distances from the tour start in the forward direction and also from the tour end in the backward direction. For each target location

o_i and the corresponding sample of the waypoint location p_i^j and sample of the heading value θ_i^k in the current solution, the shortest distances from the starting samples (i.e., all samples corresponding to the starting target location) and from the ending samples together with the respective sequence of the particular samples are stored. Then, the evaluation of the resulting path length for a simple target location removal or addition require significantly less computational time because all paths are precomputed and stored. Only the calculation of the shortest connection from the previous and to the following target location samples in the target location sequence is required without the need to find the shortest path in the whole search graph shown in Fig. 6a. However, after each change to the sequence, the stored shortest paths to particular samples have to be updated. Notice, the first waypoint is the same as the final waypoint because the tours are closed in the m-DTSPN. Therefore also the particular heading values and the waypoint locations (for depots with neighborhood) must be the same for both of these waypoints. When computing the shortest tour over a given sequence of sampled waypoints (as shown in Fig. 6b), the shortest tour has to be evaluated for each sampled heading value at the depot to keep the tour closed and minimal. In the case of the depot with the neighborhood, it has to be also evaluated for every possible waypoint location and the heading, which is naturally more demanding.

A tour in the VNS optimization represents a sequence of the targets for which the most suitable heading and waypoint location is determined from the sampled values. Therefore, a tour for the r-th vehicle is denoted Q^r and it is a sequence of waypoints

$$Q^r = (q_0^r, q_1^r, \ldots, q_{n_r}^r, q_0^r), \tag{14}$$

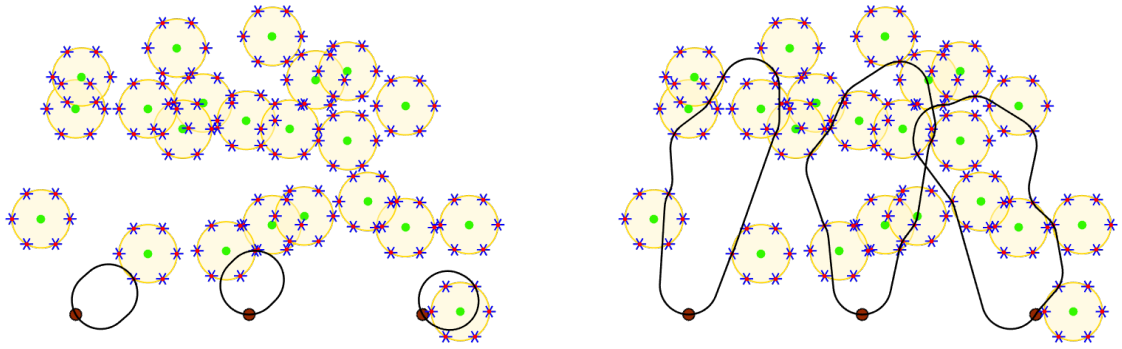
where q_0^r is the waypoint corresponding to the requested initial and terminal location of the r-th vehicle (i.e., the depot p_d^r) and n_r is the number of objects of interest visited by the r-th vehicle except q_0^r. The waypoints q_u^r for $u > 0$ are alternated during the VNS optimization (while q_0^r are fixed), but each q_u^r in all tours always corresponds to a unique object of interest $o \in \mathcal{O}$ and all objects are visited by the tours, i.e., $n = \sum_{r=1}^{m} n_r$. For better readability, we consider the subscript u of q_u^r as an index of the particular object and its waypoint in the respective sequence of the waypoints Q^r.

The most time-consuming part of the initialization is the computation of all Dubins maneuvers between all possible waypoints in the `sampleWaypoints()` function, which are saved for further usage in the VNS optimization. Regarding the particular numbers for the considered *mbzirc22* scenario with 22 targets locations and up to $m = 3$ vehicles, we consider $s = 6$ and $h = 12$ which gives up to 1620 waypoints. For such a small number of waypoints, the initialization is fast, and it is done in tens of milliseconds using conventional computational resources; however, the precomputation becomes quickly computationally more demanding with increasing n and the number of samples, see empirical evaluation in Section 7.

5.1 Initialization Procedure

The initialization heuristic utilized in the VNS-based algorithm for the m-TSP in (Soylu, 2015) is based on the competitive rule initially proposed to address the *minmax* variant of the m-TSP by SOM in (Somhom et al., 1999) to favor shorter tours and rather do not extend the longest one. The initialization starts with sorting all the target locations $o_i \in \mathcal{O}$ according to its minimal distance d_{min}^i to any of the m starting locations using the Euclidean distance. Then, a small tour for each vehicle $1 \leq r \leq m$ is created by adding one waypoint location such that the Dubins tour connecting the initial location of the r-th vehicle with the added location has the minimal tour length (see Fig. 7a). Thus, each Q^r has the form $Q^r = (q_0^r, q_1^r, q_0^r)$ and it represents a Dubins tour consisting of two Dubins maneuvers from q_0^r to q_1^r and from q_1^r to q_0^r with the lengths $\mathcal{L}(q_0^r, q_1^r)$ and $\mathcal{L}(q_1^r, q_0^r)$, respectively. The length of the Dubins tour represented by Q^r is further denoted $\mathcal{L}(Q^r)$ for brevity.

After the creation of the first tours, particular waypoints for all not assigned objects are sequentially inserted to the tours in the order defined by the increasing distance d_{min}^i of the target location to the initial location. The respective waypoint $q(o_i)$ (together with its heading and location) is selected during the determination



(a) Initial small tours each with one target location

(b) Final solution created by the Initialization procedure

Figure 7: Tours for all $m = 3$ vehicles created by the proposed Initialization procedure of the VNS based m-DTSPN

of the most suitable tour r^* and the particular position j^* in the tour using the competitive rule

$$r^*, j^* = \underset{1 \leq r \leq m, 1 \leq j \leq n_r}{\operatorname{argmin}} \left(\mathcal{L}(q_j^r, q(o_i)) + \mathcal{L}(q(o_i), q_{j+1}^r) \right) \mathcal{W}(m, r), \tag{15}$$

where $q(o_i)$ represents the most suitable location from up to s samples around o_i with the best heading of the h heading samples. The term $\mathcal{W}(m, r)$ represents the competitive weight introduced in (Somhom et al., 1999) to address the *minmax* variant of the m-TSP. It is computed as

$$\mathcal{W}(m, r) = 1 + \frac{\mathcal{L}(Q^r) + \mathcal{L}_{avg}^m}{\mathcal{L}_{avg}^m}, \tag{16}$$

where $\mathcal{L}_{avg}^m$ is the average length of the Dubins tours $Q^1, \ldots, Q^m$

$$\mathcal{L}_{avg}^m = \frac{1}{m} \sum_{r=1}^{m} \mathcal{L}(Q^r). \tag{17}$$

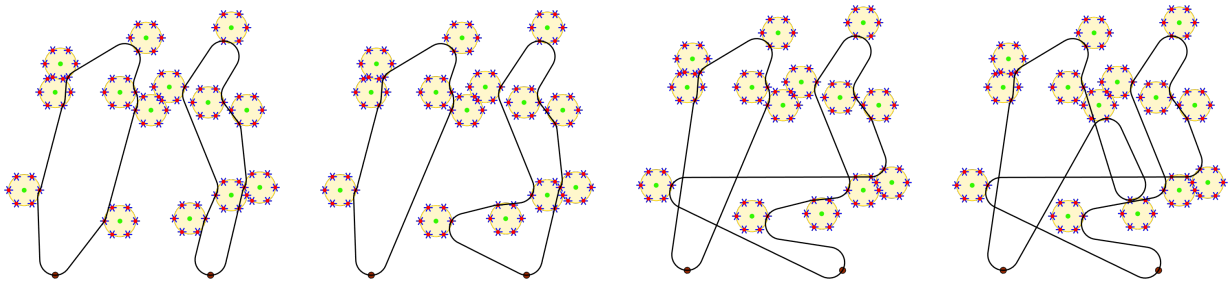
An example of the solution created by the proposed Initialization procedure is shown in Fig. 7b.

5.2 Shake Procedure

The *shake* procedure is utilized to get the currently best incumbent solution x from possible local optima by using up to k_{max} consecutive simple one point moves. Each such a single move starts with a random selection of two distinct tours $i, j \in \{1, \ldots, r\}, i \neq j$ and one target location in each tour $u \in Q^i, v \in Q^j$, where u and v are the position indexes of the particular selected target locations in the i-th and j-th tours, respectively. Then, the corresponding object associated with q_u^i is moved from Q^i to Q^j where it is placed after the v-th position, such that the tours after the operation become $Q^i = (q_0^i, \ldots, q_{u-1}^i, q_{u+1}^i, \ldots, q_0^i)$ and $Q^j = (q_0^j, \ldots, q_v^j, q_u^i, q_{v+1}^j, \ldots, q_0^j)$. By using the one point move operation for $k = 1, \ldots, k_{max}$ times, the *shake* procedure creates a random solution x' within N_k neighborhood of the original solution (see Fig. 8). The particular number of the performed operations for the results presented in this paper is $k_{max} = 5$.

5.3 Local Search Procedure

The *local search* procedure uses a randomly created solution produced by the *shake* procedure and systematically tries to find a better solution. The employed variant of the procedure is called the sequential local search, which indicates the fact that all the neighborhood operators are tested in a sequence according to



(a) Incumbent solution x (b) First random move $k = 1$ (c) Second random move $k = 2$ (d) Third random move $k = 3$

Figure 8: An example of the shake procedure sequence with $k = 1 \dots 3$ random moves for $m = 2$ vehicles

the value l of the six operators defined below (i.e., $l_{max} = 6$). Once the solution quality is improved, the *local search* is started again to optimize the improved solution. Notice that the ordering of the operators in the *local search* procedure can significantly influence the final solution quality. To improve the efficiency of the VNS search, only operators that can decrease the length of the longest tour are considered. The *local search* operators (Soylu, 2015) adopted for the m-DTSP(N) are as follows:

- **One point move** ($l = 1$) operator uses the smallest neighborhood possible to move only a single target location to a different tour.
- **Or-opt2 move** ($l = 2$) operator moves two adjacent target locations to a different tour.
- **Two point move** ($l = 3$) operator exchanges two points (target locations).
- **Or-opt3 move** ($l = 4$) operator moves three adjacent target locations to a different tour.
- **Three point move** ($l = 5$) operator exchanges two adjacent target locations from the longest tour with one target location in a different tour.
- **2-Opt move** ($l = 6$) operator (Croes, 1958) selects two target locations in a tour and swaps the sub-tour between the targets which tries to improve all individual tours separately. The operator is repeatedly performed until it improves the tour length. Notice, the original idea of this heuristic is to remove unnecessary self-crosses in a solution of the Euclidean TSP.

6 Unsupervised Learning for m-DTSPN and 3D Bézier Curve-based Multi-Vehicle Surveillance Planning

A solution of the m-DTSPN based on unsupervised learning has been introduced in (Faigl and Váňa, 2017), and therefore, an overview of the method is presented in this section to provide the necessary details for the proposed generalization to trajectory planning using Bézier curves. The unsupervised learning framework is based on the growing *Self-Organizing Map* (SOM) for the TSPN proposed in (Faigl, 2018) which differs from a regular SOM (i.e., usually a 2D lattice (Kohonen et al., 2001)) in the organization of the output layer and incremental adding new neurons to the network. In general, SOM for the TSP is a two-layered neural network in which the input layer serves for presenting the target locations $\mathcal{O}$ and the output layer is organized into an array of neurons (Angéniol et al., 1988; Fort, 1988), which defines a sequence of visits to the targets. The neuron weights are in the same space as the input signals (target locations) and the connected neuron weights form a ring in the input space $\mathbb{R}^2$, and thus represent a closed path in $\mathbb{R}^2$, see Fig. 9a, i.e., the weights are considered as the neuron locations. SOM can be represented as a sequence of neurons $\mathcal{N} = (\nu_1, \dots, \nu_M)$, where M is usually more than two times the number of target locations (Somhom et al., 1997).

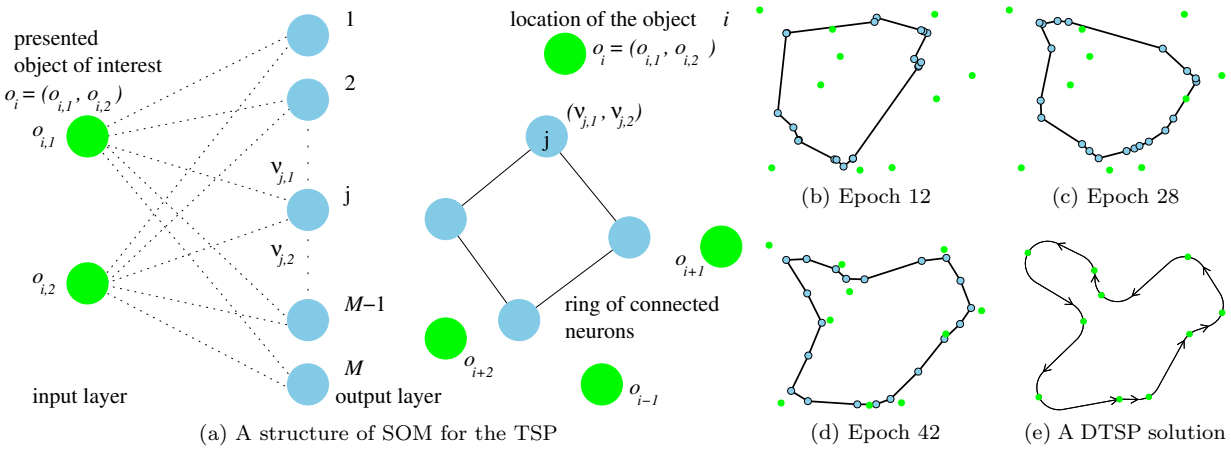


Figure 9: A structure of the SOM for the TSP and visualization of the ring evolution during the learning. The green disks are the target locations to be visited by the tour, and blue disks represent the neuron weights in the input space $\mathbb{R}^2$. The connections between the input and output layers represent that the best matching neuron is computed using its distance to the input signal (location). For the DTSP (Faigl and Váňa, 2017), each neuron is in addition to the neuron weights (locations) as a point in $\mathbb{R}^2$ also associated with the particular target (or waypoint) location and with h heading values, and thus a solution of the DTSP can be determined from the ring of neurons after each learning epoch by solving the related DTP, e.g., using the forward search method described in Section 4.1.

The unsupervised learning of the network is realized by an iterative procedure in which all the target locations are presented to the network and for each such a presented location $o \in \mathcal{O}$, the best matching neuron is selected in the winner selection procedure, i.e., the neuron with the closest weights to o. Then, the winner neuron is adapted towards the presented input together with the neighboring neurons to the winner neuron with decreasing power of the adaptation defined by the neighboring function. In a single learning epoch, all targets are presented to the network, and a solution of the TSP can be retrieved after each epoch by traversing the ring, i.e., the tour is constructed from the targets associated with their winner neurons into a sequence of targets defined by the position of the winner neurons in the ring. During the adaptation, the winner neurons are getting closer to the targets, and the network is stabilized in tens or hundreds of epochs because of cooling schedule of the power of the adaptation. An evolution of SOM in solving an instance of the TSP is shown in Fig. 9. For the DTSP, the neurons are associated not only with a location in $\mathbb{R}^2$ but also with up to h heading values (Faigl and Váňa, 2016). Therefore a solution of the DTSP can be determined after each learning epoch by a solution of the DTP with the sequence of visits to the targets defined by the order of the winner neurons in the SOM output layer (ring), e.g., using the feed-forward search method presented in Section 4.1.

In addition to the headings associated with the neurons, the main part of the unsupervised learning for the DTSPN is the winner selection in which expected heading of the vehicle at the waypoint location is determined together with the waypoint location itself. The idea of the winner selection is visualized in Fig. 10. The range of the neighboring neurons that are adapted together with the winner neuron is restricted by the neurons ν_{prev} and ν_{next} such that the expected length of the Dubins path (see the red curve in Fig. 10) to visit the target location o is minimized:

$$\mathcal{L}_g = \mathcal{L}(\nu_{prev}, (o, \theta)) + \mathcal{L}((o, \theta), \nu_{next}), \tag{18}$$

where θ is one of the h heading values associated with the winner neuron. Nevertheless, the search for ν_{prev} and ν_{next} is limited to the range $0.2M$ around the winner, where M is the current number of neurons in the ring, as in other SOM-based TSP solvers, e.g., (Somhom et al., 1997). The neurons adapted with the winner neuron ν^* are in the range of ν_{prev} and ν_{next} for which a value of the neighbouring function (19) is above a threshold that is empirically set to 10^{-5}. The neighbouring function is defined for the active neuron in a

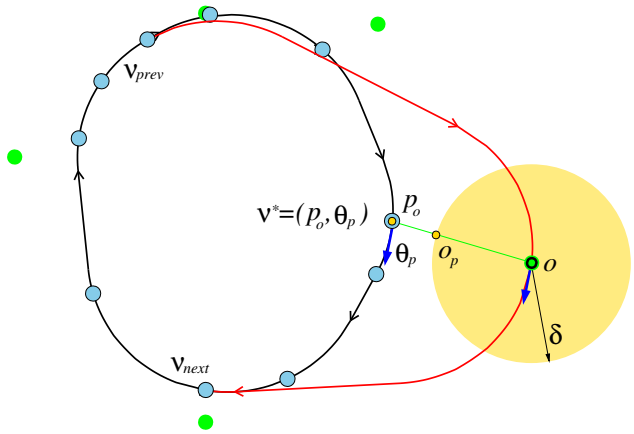


Figure 10: A selection of the winner neuron for the presented location o in unsupervised learning for the DTSPN. The current ring of neurons represents the Dubins path showed as the black curve connecting the blue neurons. The closest point p_o of the Dubins path to o is used as the neuron weights for the winner neuron. The point o_p corresponds to the alternate target location towards which the network is adapted because o can be covered within δ sensing range from the target location. The shortest possible path connecting ν_{prev} and ν_{next} through the point o using the vehicle heading θ_p is in red.

similar way as in a regular SOM for the TSP (Somhom et al., 1997; Cochrane and Beasley, 2003):

$$f(\sigma, d) = \begin{cases} e^{\frac{-d^2}{\sigma^2}} & \text{for neurons around } \nu^* \text{ in the range defined by } \nu_{prev} \text{ and } \nu_{next} \\ 0 & \text{otherwise} \end{cases}, \tag{19}$$

where σ is the learning gain and d is a distance of the neuron from ν^* in the number of neurons in the ring.

The schema of the unsupervised learning is depicted in Algorithm 2. Due to the non-monotonicity of the length of the Dubins maneuvers, the ring may contain unnecessary loops and crossings, and therefore, the simple 2-Opt heuristic (Croes, 1958) is employed to improve the solution similarly as in other SOM-based TSP solvers (Ahmad and Kim, 2015). The 2-Opt heuristic is computationally inexpensive procedure $O(n^3)$ which can improve the solution about few percentage points. In addition, the final trajectory is determined by the high-quality DTP solver (Faigl et al., 2017) which utilizes a tight-lower bound (Manyam and Rathinam, 2015) to stop the refinement of the heading samples when the maximal number of samples 1024 is reached or when the ratio of the trajectory length to the lower bound solution is less than 1.01. Notice, the sensing range δ can be easily individualized for each particular object of interest by a simple usage of the particular range in the winner selection. Besides, in the case of the fixed starting location, the range can be set to zero and the target location o is directly used as the alternate target location o_p similarly to the solution of the DTSP (Faigl and Váňa, 2016; Faigl and Váňa, 2017).

Based on the empirical evaluation, the parameters of the learning $\mu = 0.6$, $\alpha = 0.1$, and the initial value of $\sigma = 10$ can be considered as fixed and they have been selected as a trade-off between the computational requirements and quality of the found solutions, albeit they can be further tuned for specific scenarios. Thus, the only parameters of the learning procedure are the number of additional heading values h per each neuron and the maximal number of learning epochs i_{max}. Regarding the results presented in (Faigl and Váňa, 2017), values $h > 3$ only increase the computational burden and do not significantly improve the solution quality, therefore $h = 3$ is used for all the results presented in this paper. The network is usually stabilized in around 130 learning epochs, and thus the maximal number of learning epochs i_{max} is set to $i_{max} = 150$. A further discussion of the network convergence can be found in (Faigl and Hollinger, 2018). Nevertheless, a solution is available after each learning epoch using the waypoint locations associated with the winner neurons.

The computational complexity of a single learning epoch depends on the number of targets n presented to the network and the number of neurons M in the ring, which does not exceed $2n$ because of the ring

Algorithm 2: Unsupervised learning algorithm for solving the DTSPN.

Input : $\mathcal{O}$ – A given set of objects of interest
Input : p_d – The requested initial location (depot) of the vehicle
Output : Q – Determined Dubins path covering $\mathcal{O}$

▷ **Initialization:**

1. For n target locations $\mathcal{O}$, create a ring $\mathcal{N}$ with one neuron with the weights set to the starting location p_d.
2. Set the learning gain $G = 10$, the learning rate $\mu = 0.6$, the gain decreasing rate $\alpha = 0.1$, and the epoch counter $i = 1$.

▷ **Learning Epoch:**

3. For each target o in the randomized set $o \in \Pi(\mathcal{O} \bigcup \{p_d\})$
   (a) *Winner selection*: determine the point p_o together with the expected heading θ_p and waypoint location o_p as in Fig. 10. Create a new neuron with the weights p_o and add it as a new winner ν^* to the ring.
   (b) *Adapt* ν^* and its neighbouring neurons (defined by ν_{prev} and ν_{next} (18)) to o_p using the neighbouring function (19). The weights of each adapted neuron ν are set to a new location $\nu' = \nu + \mu f(\sigma, d)(o_p - \nu)$.

▷ **Update:**

4. *Ring regeneration*: remove all non-winner neurons. Use the sequence of the winner neurons defined by the ring together with the headings and waypoint locations associated with the neurons to solve the DTSPN as a solution Q_i of the related DTP (using the sampling-based algorithm described in Section 4.1 with the length $\mathcal{L}(Q_i, \mathcal{O})$.
5. *Update learning parameters:* $\sigma = \sigma(1 - i\alpha)$, $i = i + 1$.
6. *Termination condition:* If $i \geq i_{max}$ or *winner neurons are negligibly close to the waypoint locations* (e.g., less than 10^{-3}) `Stop` the adaptation. Otherwise go to Step 3.

▷ **Final Tour Construction:**

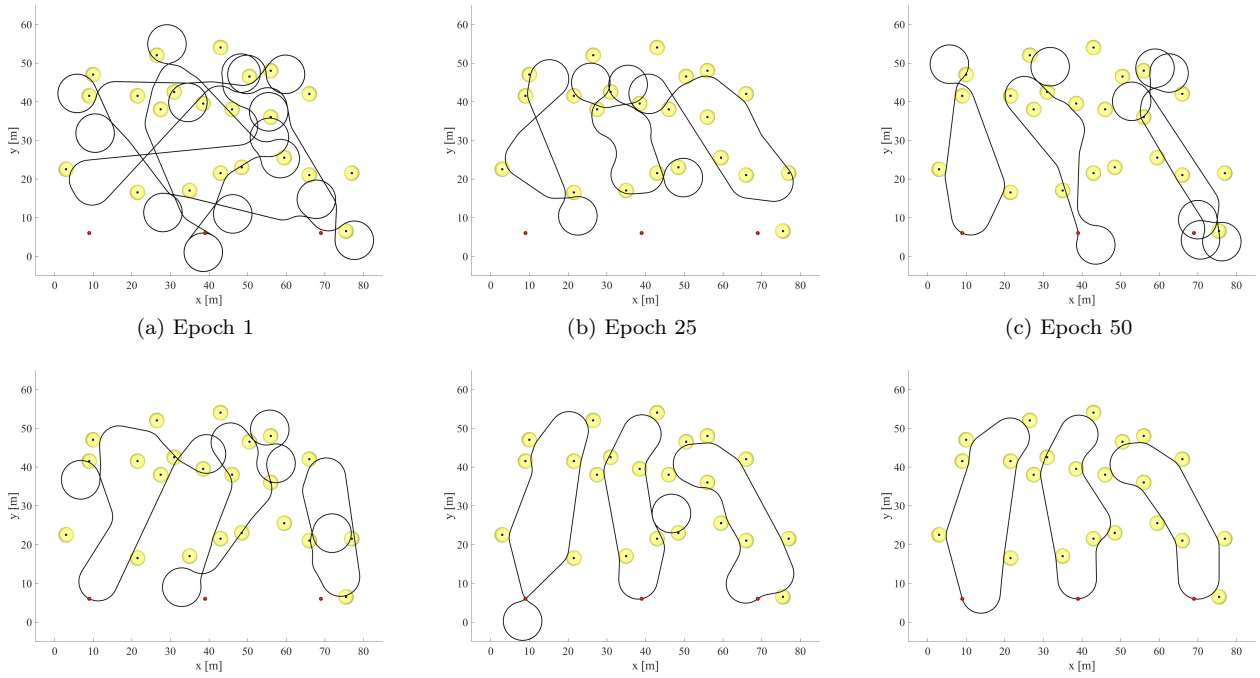
7. *Improve the solution* Q using 2-Opt heuristic (Croes, 1958).
8. *Return* the final trajectory as a solution of the DTP found by the guided sampling with up to 1024 samples or the approximation factor 1.01 by the algorithm (Faigl et al., 2017).

regeneration (Faigl and Váňa, 2017), and thus it can be bounded by $O(n^2)$. The number of learning epochs is constant ($i_{max} = 150$), and thus the computational complexity depends on the 2-Opt improvement that can be bounded by $O(n^3)$ and a solution of the DTP (Faigl et al., 2017), which depends on the iterative forward search procedure described in Section 4.1 with $O(nh^3)$. For a fixed $h = 3$, the total computational complexity can be bounded by $O(n^3)$ because of 2-Opt. Nevertheless, the real required computational time for the *mbzirc22* scenario is in hundreds of milliseconds as it is reported in Section 7, which perfectly fits our expectation about the computational requirements.

6.1 Learning for a Team of Vehicles

The described learning procedure for a single vehicle can be straightforwardly applied to a team of vehicles by creating an individual ring of neurons for each vehicle as $\mathcal{N}^r = (\nu_1^r, \ldots, \nu_{M^r}^r)$, where M^r is the number of the neurons in the r-th ring. The application follows existing extension of the SOM-based solution for the TSP to the *minmax* variant of the m-TSP (Somhom et al., 1999; Faigl, 2016) where a winner neuron is preferably selected from the ring which represents the shortest tour, which is motivated to minimize the longest tour (Somhom et al., 1999). In the winner neuron determination, the distance $|(p_o, o)|$ of the point p_o on the Dubins tour represented by the current ring $\mathcal{N}^r$ and the target location o is weighted according to the difference of the length $\mathcal{L}(\mathcal{N}^r)$ of the Dubins tour represented by $\mathcal{N}^r$ and the average length of the tours represented by the rings. The winner neuron ν^* is selected from the ring r for which the respective point $|(p_o, o)|$ used as the weights of ν^* has the minimal weighted distance:

$$r = \underset{r \in \{1,\ldots,m\}}{\operatorname{argmin}} \mathcal{W}(m, r)|(p_o^r, o)|, \tag{20}$$



(a) Epoch 1 (b) Epoch 25 (c) Epoch 50

(d) Epoch 75 (e) Epoch 118 – SOM solution (f) Final tours

Figure 11: Evolution of SOM solving the *mbzirc22* m-DTSPN scenario with $n = 22$ target locations that are shown as small black disks. The sensing range $\delta = 2$ m is visualized by yellow disks around the target locations. The shown Dubins paths connect the neuron locations using the determined headings. The minimal turning radius of the Dubins vehicle is $\rho = 5$ m, and the initial locations of the vehicles are highlighted by the red disks. A solution is available after each learning epoch using the waypoints associated with neurons (not shown), and the network converges (the neuron locations match their waypoint locations) in 118 learning epochs. The final solution is then improved by a solution of the related DTP.

where $\mathcal{W}(m, r)$ is the competitive weight (16) with the trajectory length $\mathcal{L}(Q^r)$ computed as the length of the trajectory represented by the ring $\mathcal{N}^r$, i.e., $\mathcal{L}(\mathcal{N}^r)$ is used instead of $\mathcal{L}(Q^r)$ in (16) and (17).

A straightforward usage of the learning procedure depicted in Algorithm 2 in multi-robot planning would provide a set of m independent patrolling routes. Therefore in the case the initial locations of the vehicles are prescribed by the depots $p_d^1, \ldots, p_d^m$, each ring $\mathcal{N}^r$ is individually adapted towards p_d^r without the competition among the rings prior a regular adaptation of the rings to the targets $\mathcal{O}$ without the depots, which ensures each ring will be connected with the respective initial location p_d^r. Moreover, for such a case it is suitable to consider the initial location without the neighborhood, and thus, for the depots, $\delta = 0$ is considered in the selection of winner neurons and adaptions. An example of the SOM evolution is visualized in Fig. 11.

The computational complexity of the unsupervised learning in solving the m-DTSPN does not significantly increase because the learning depends on the number of neurons that are distributed into the particular rings. Therefore, the complexity grows only with the additional m locations that are the individual depots of the vehicles. Hence the computational complexity can be bounded by $O((n + m)^3)$ which for $m \ll n$ can be bounded by $O(n^3)$, and thus it is independent on the number of vehicles, see (Best et al., 2018) for a detail discussion.

6.2 Surveillance Planning with Bézier Curves

The SOM-based solution of the m-DTSPN can be easily generalized to a different parametrization of the trajectory. Even though the Dubins vehicle is used in the above-described procedure, the unsupervised learning does not rely on the Dubins vehicle model. In fact, the Dubins maneuvers can be substituted by any curve parametrization, and in this work, we consider Bézier curves briefly introduced in Section 4.2. Since Bézier curve can be used for the parametrization of the 3D path, we do not use the kinematic model of the Dubins vehicle (1). Instead of that, we consider the hexacopters can generally follow any 3D path, and therefore, we consider the position of the UAVs along the 3D path described as a point $p = (x, y, z) \in \mathbb{R}^3$. The velocity $\mathbf{v}$ is defined by the turning angle θ and the climb/dive angle of the trajectory ψ at the position p

$$\mathbf{v} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = v \begin{bmatrix} \cos\theta\cos\psi \\ \sin\theta\cos\psi \\ \sin\psi \end{bmatrix}. \tag{21}$$

Notice, here, we do not model the orientation of the vehicle; however, the parameters of the Bézier curves are adjusted during the unsupervised learning to provide fast execution of the determined path by a real vehicle. Finally, a trajectory of the final solution is constructed from the determined Bézier curves with respect to the vehicle motion constraints using the computation of the velocity profile and the TTE described in Section 4.2.

There are two main parts of the learning procedure where additional computations related to Bézier curves have to be included: 1. the winner selection and adaptation; and 2. the determination of the trajectory represented by the ring instead of a solution of the DTP with heading values sampled during the learning. In the winner selection, the point p_o is determined in a similar way as for the DTSPN, just a sequence of the Bézier curves, each defined by four control points, (8) is utilized. However, it is requested that the final trajectory is smooth and continuous, and therefore, the following conditions have to be satisfied after the adaptation of neurons to satisfy this requirement.

Let $\mathcal{B}^i$ and $\mathcal{B}^j$ be two consecutive Bézier curves (i.e., $j = i + 1$) with the control points ($\mathbf{B}_0^i$, $\mathbf{B}_1^i$, $\mathbf{B}_2^i$, $\mathbf{B}_3^i$) and ($\mathbf{B}_0^j$, $\mathbf{B}_1^j$, $\mathbf{B}_2^j$, $\mathbf{B}_3^j$), respectively. The last control point $\mathbf{B}_3^i$ of $\mathcal{B}^i$ and the first control point $\mathbf{B}_0^j$ need to be identical to keep the trajectory continuous

$$\mathbf{B}_3^i = \mathbf{B}_0^j. \tag{22}$$

Besides, the tangents of the Bézier curve have to point to the same direction to support traveling of the vehicle along the final trajectory. The tangents can be defined as

$$\mathbf{t}_a^i = \mathbf{B}_1^i - \mathbf{B}_0^i, \quad \mathbf{t}_b^i = \mathbf{B}_3^i - \mathbf{B}_2^i \tag{23}$$

with the length of the particular tangent vector

$$l_a^i = \left|\mathbf{t}_a^i\right|, \quad l_b^i = \left|\mathbf{t}_b^i\right|. \tag{24}$$

This requirement can be satisfied by the condition ensuring the smooth trajectory

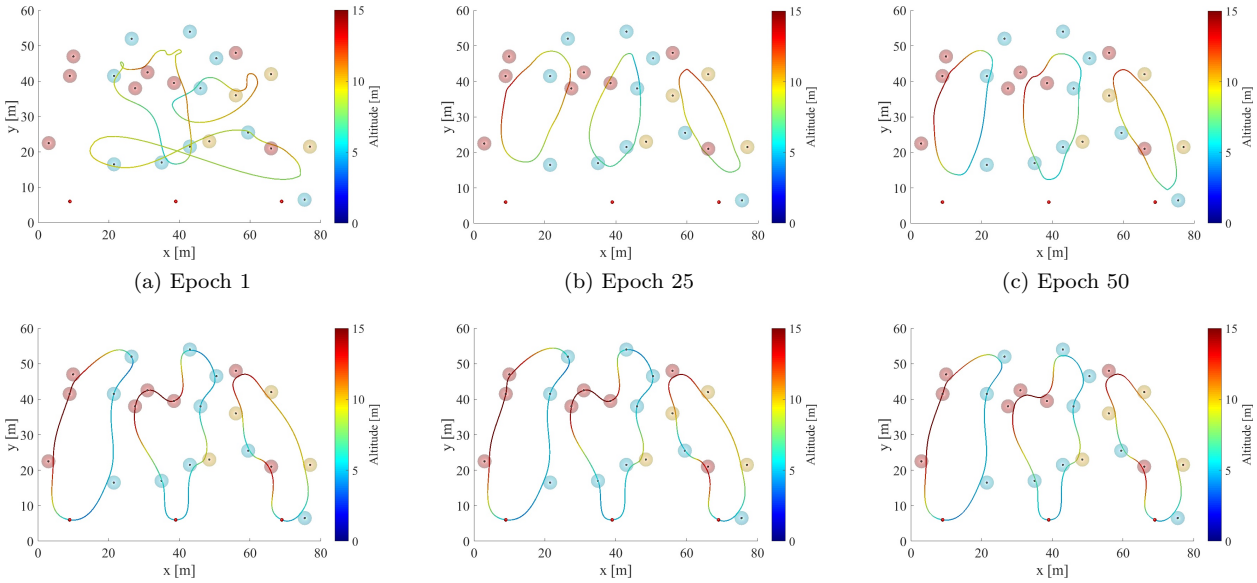
$$l_a^j\, \mathbf{t}_b^i = l_b^i\, \mathbf{t}_a^j. \tag{25}$$

The waypoint locations and headings are not sufficient to define a Bézier curve represented by two neighboring neurons, which is further called maneuver. Therefore, each neuron is associated with the tangent vectors for the unique characterization of the maneuver. Each Bézier maneuver is defined by two tangent vectors separately, and thus a continuity of the velocity is ensured by (25). Hence, each neuron ν_i is associated with the heading angle θ_i, pitch angle ψ_i, and the lengths of the tangent vectors l_a^i and l_b^i. The tangent vectors for the two maneuvers, for which ν_i is incident with, can be expressed as

$$\mathbf{t}_b^{i-1} = l_b^i \frac{\mathbf{v}}{|\mathbf{v}|}, \quad \mathbf{t}_a^i = l_a^i \frac{\mathbf{v}}{|\mathbf{v}|}. \tag{26}$$

Notice that the tangent vector $\mathbf{t}_b^{i-1}$ corresponds with the Bézier curve $\mathcal{B}_{i-1}$ which terminates at ν_i. Conversely $\mathbf{t}_a^i$ defines the initial part of $\mathcal{B}_i$, and thus the indexes of these two tangent vectors that are related to the same neuron differ.

In addition to the constraints on the consecutive Bézier curves, a local optimization of the trajectory is performed after the ring regeneration because the neurons that are not winners are removed from the ring at the end of each learning epoch. For the multi-vehicle planning, each ring is treated independently as an optimization problem of minimizing the TTE along the trajectory as follows.



(a) Epoch 1 (b) Epoch 25 (c) Epoch 50

(d) Epoch 75 (e) Epoch 77 – SOM solution (f) Final tours

Figure 12: An evolution of the proposed SOM-based 3D surveillance planning using Bézier curves in solving 3D instance of the *mbzirc22* scenario with target locations at different altitudes. The target locations are visualized as small disks surrounded by a spherical neighborhood for sensing range $\delta = 2$ meters. The altitude of the targets and paths is indicated by the color (from a low altitude in the blue color to the highest altitude in the red).

The whole trajectory is described by the sequence of the neurons $\mathcal{N}$, where each neuron $\nu_i \in \mathcal{N}$ represents the particular parameters of the Bézier curve. A single change of one neuron influences the two incident Bézier curves, and it can also influence the velocity profile of the whole trajectory. However, based on our empirical observations, the changes are mostly local, and therefore, the optimization of the whole trajectory is performed locally and the values of θ_i, ψ_i associated with ν_i are numerically optimized with respect to the velocity profile of the trajectory defined by the three consecutive neurons in the ring ν_{i-1}, ν_i, and ν_{i+1}. Notice, the ring is closed, and therefore, the subscripts of the neurons are closed to the modulo of the number of neurons in the ring. An example of the evolution of the proposed SOM-based solution for the 3D surveillance planning with Bézier curves is visualized in Fig. 12.

Beside these local optimizations, we employed the idea of LIO (Váňa and Faigl, 2015), and the individual local optimizations of all neurons in the ring are performed in multiple iterations of the whole ring. In particular, three iterations of the whole ring are performed, and each neuron is locally optimized in each iteration. The local numerical optimization uses a step 0.5% of the variable range, and thus the step for θ and ψ is 0.01π. On the other hand, the 2-Opt heuristics (Line 7 in the unsupervised learning Algorithm 2) is not utilized, because any change would require optimization of the control points. The modified learning procedure is summarized in Algorithm 3. Each Bézier curve is defined by the control points associated with the neurons including the locations of the neurons, and thus a feasible solution is not available after each learning epoch unless the waypoints associated with the neurons are used, and a new trajectory is determined.

Algorithm 3: Unsupervised learning algorithm for surveillance planning with Bézier curves.

Input : $\mathcal{O}$ – A given set of objects of interest
Input : p_d – The requested initial location (depot) of the vehicle
Output : $\mathcal{X}, \mathcal{T}$ – Determined surveillance trajectory that covers $\mathcal{O}$ and the computed TTE (using velocity profile)

▷ **Initialization:**

1. For n target locations $\mathcal{O}$, create a ring $\mathcal{N}$ with n neurons with the weights set such that the connected weights form a closed path around the center of the target locations.
2. Set the learning gain $G = 10$, the learning rate $\mu = 0.6$, the gain decreasing rate $\alpha = 0.1$, and the epoch counter $i = 1$.

▷ **Learning Epoch:**

3. For each target o in the randomized set $o \in \Pi(\mathcal{O} \bigcup \{p_d\})$
   (a) *Winner selection*: determine the point p_o and waypoint location o_p similarly as in Fig. 10 but p_o is the closest point of the sequence of Bézier curves to the location of o. The particular Bézier curve on which p_o is located is constructed from the respective two consecutive neurons and the associated control points, i.e., the tangents (directions) of the curves, see Section 4.2. Create a new neuron ν^* with the weights according to p_o (for the location) and the tangents according to the Bézier curves to split it into two parts, and add ν^* to the ring.
   (b) *Adapt* ν^* and its neighbouring neurons to o_p using the neighbouring function (19) but with the neighborhood defined as $0.2M$, where M is the current number of neurons in the ring. The weights of each adapted neuron ν are set to a new location $\nu' = \nu + \mu f(\sigma, d)(o_p - \nu)$, i.e., only the control points corresponding to the neuron location are modified and the tangents remain the same.

▷ **Update:**

4. *Ring regeneration*: remove all non-winner neurons. Use the sequence of the winner neurons defined by the ring together with the control points and waypoint locations associated with the neurons to optimize the sequence of Bézier curves using LIO (Váňa and Faigl, 2015).
5. *Update learning parameters:* $\sigma = \sigma(1 - i\alpha)$, $i = i + 1$.
6. *Termination condition:* If $i \geq i_{max}$ or *winner neurons are negligibly close to the waypoint locations* (e.g., less than 10^{-3}) `Stop` the adaptation. Otherwise go to Step 3.

▷ **Final Tour Construction:**

7. *Return* the final trajectory as a sequence of Bézier curves for which the velocity profile is determined using the procedure described in Section 4.2.

However it is one of the most computationally demanding parts, especially for completely changed locations, and therefore, we do not consider the learning procedure with Bézier curves as the any-time algorithm. After the network convergence, the velocity profile for the Bézier curve is calculated numerically using 200 uniformly distributed samples for the range $\tau \in [0, 1]$ according to (8). The real computational requirements are reported in Section 7.

7 Results

An empirical evaluation of the proposed VNS-based and SOM-based solvers for the m-DTSPN consists of four main parts. First, the algorithms' performance is studied in the *mbzirc22*[1] scenario because of our motivation for the addressed problem. After that, the proposed generalization of the SOM-based solver for surveillance planning using Bézier curves is studied in 2D problems first, and we compare trajectories

[1]The *mbzirc22* scenario contains 22 objects of interest positioned at the target locations (in meters): (27.5, 47.0), (10.0, 36.5), (51.5, 41.5), (32.0, 37.5), (67.0, 16.0), (44.0, 49.0), (44.0, 16.5), (49.5, 18.0), (60.5, 20.5), (39.5, 34.5), (78.0, 16.5), (67.0, 37.0), (76.5, 1.5), (28.5, 33.0), (22.5, 11.5), (57.0, 31.0), (47.0, 33.0), (4.0, 17.5), (36.0, 12.0), (57.0, 43.0), (22.5, 36.5), (11.0, 42.0). The scenario is visualized in Fig. 3 where the initial locations (depots) of the vehicles are (10,1) for the first vehicle, (40,1) for the second vehicle, and (70,1) for the third vehicle.

consisting of Dubins maneuvers with Bézier curves in the second part of the evaluation. Then, the proposed unsupervised learning based 3D surveillance planning with Bézier curves is studied in 3D scenarios. Finally, a brief evaluation of the algorithms' performance in larger problems is presented in the fourth part of the herein reported results to provide an overview of the expected performance of the evaluated algorithms in different scenarios.

In addition to the proposed algorithms, the Memetic algorithm (Zhang et al., 2014) is included in the evaluation as it demonstrates high-quality solution in (Faigl and Váňa, 2017), albeit it is computational demanding. Regarding the motivation, the computational time for the VNS and Memetic solvers has been limited to 1, 5, 10, and 60 seconds, because of our initial intention to have a solution in less than one second. The SOM provides a solution of the addressed problem in less than one second, and therefore, the computational time is not explicitly limited, but the maximal number of the learning epochs is set to $i_{max} = 150$. The particular values of VNS solver parameters and also learning parameters of the SOM are used as they are reported in Section 5 and Section 6, respectively. The parameters of the Memetic algorithm are selected as in (Faigl and Váňa, 2017) according to the recommendation of (Zhang et al., 2014), i.e., the population size is set to $20n$, where n is the number of target locations of the solved problem.

All the evaluated algorithms are randomized; therefore 20 trials are computed for every problem instance by each of the evaluated algorithms, and the reported performance indicators are computed as the average values accompanied by the standard deviations and the best-found solution from the solved trials. The indicators of the solution quality are computed as the length of the longest Dubins tour among the tours for the vehicles in the team. Besides, the TTE is used in the case of Bézier curve and velocity profiles computed for the Dubins tours. In addition to average values of the length of the longest tour L_{avg}, and its standard deviation L_{std}, the quality of the best solution among the trials is reported as L_{best}.

The computational requirements are measured as the real required computational time. All the algorithms have been implemented in C++, and they use the same implementation for computing Dubins maneuvers and solution of the related DTP. All implementations are compiled by the same compiler Clang 4.0 and executed within the identical computational environment using a single core of the iCore7 processor running at 4 GHz. Therefore, all the reported computational times represent realistic requirements and can be directly compared.

The particular evaluated algorithms and their variants with the restricted computational time are denoted: Memetic 1 s, Memetic 5 s, Memetic 10 s, Memetic 60 s, VNS 1 s, VNS 5 s, VNS 10 s, VNS 60 s, SOM (Dubins) and SOM (Bézier). The problems being solved are parametrized by the number of vehicles $m \in \{1, 2, 3\}$, the sensing range δ limited to $0 \leq \delta \leq 5$ m, and the minimal turning radius ρ, which has the default value $\rho = 5$ m. For the comparison of the Dubins maneuvers with the Bézier curves the value of ρ is selected from the set $\rho \in \{5, 6, 7, 8, 9, 10, 11, 12, 12.5\}$ in meters and the velocity profile is computed for $v_{horiz} = 5$ m.s^{-1} and the maximal vehicle acceleration $a_{horiz} = 2$ m.s^{-2}, which are also used for velocity profiles along the trajectories consisting of Bézier curves.

7.1 Performance Evaluation in m-DTSP and m-DTSPN

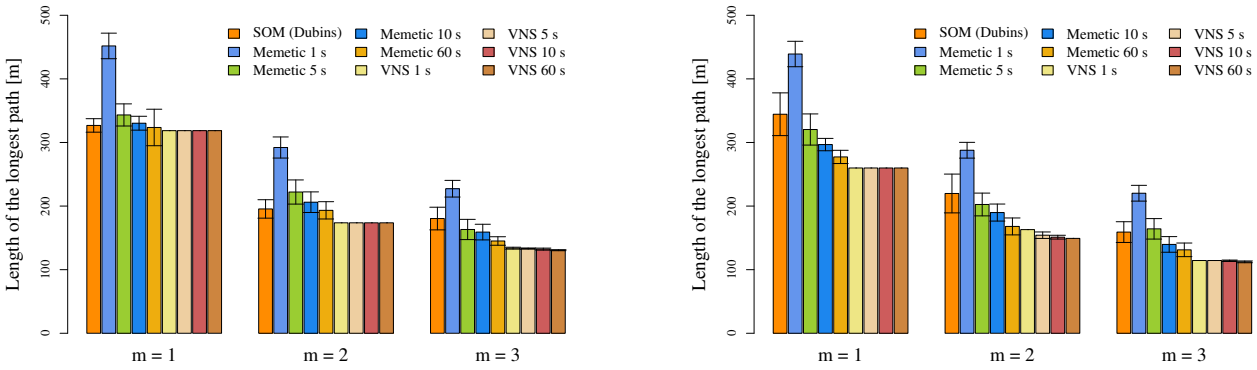
The m-DTSP formulation represents the basic surveillance planning and the lengths found by the evaluated algorithms for m vehicles are reported in Table 2, where the best results found under less than one second are highlighted in bold, while the shortest solution regardless the computational requirements are underlined. For a single vehicle, the SOM-based approach provides the best solution in less than 250 ms, which is a bit more demanding than the initialization part of the VNS (further denoted as the VNS Init), see computational requirements depicted in Table 3. For the relatively small problem *mbzirc22* and $m = 1$, the VNS initialization is very fast, and a solution is provided in less than 100 ms. Besides, the standard deviation for SOM is about 10 meters, and therefore, the most suitable algorithms seem to be the SOM-based planning framework and VNS-based optimization. However, for a team of UAVs, the best solutions found in less than one second are provided by the proposed VNS solver, and they are found with very low standard deviations

because they are mostly based on the initial solutions.

Table 2: Average and Best Found Solutions of the $m-$DTSP *mbzirc22* Scenarios

Method	$m = 1$			$m = 2$			$m = 3$		
	L_{best}	L_{avg}	L_{std}	L_{best}	L_{avg}	L_{std}	L_{best}	L_{avg}	L_{std}
Memetic 1 s	402.8	451.8	20.1	258.4	292.1	16.6	194.1	227.3	13.1
Memetic 5 s	318.5	343.3	17.4	194.4	222.1	19.0	139.6	163.2	15.9
Memetic 10 s	310.7	330.3	10.9	180.4	206.1	16.2	134.3	159.2	12.3
Memetic 60 s	306.4	323.6	28.7	170.7	193.2	13.6	131.0	145.1	6.7
VNS 1 s	318.6	**318.6**	0.0	173.7	**173.7**	0.0	**130.5**	**133.8**	1.5
VNS 5 s	318.6	318.6	0.0	173.7	173.7	0.0	130.5	133.2	0.9
VNS 10 s	318.6	318.6	0.0	173.7	173.7	0.0	130.0	132.3	1.6
VNS 60 s	318.6	318.6	0.0	173.7	173.7	0.0	130.0	130.6	0.8
SOM	**311.2**	326.8	10.6	**170.5**	195.5	14.5	136.3	156.1	13.0

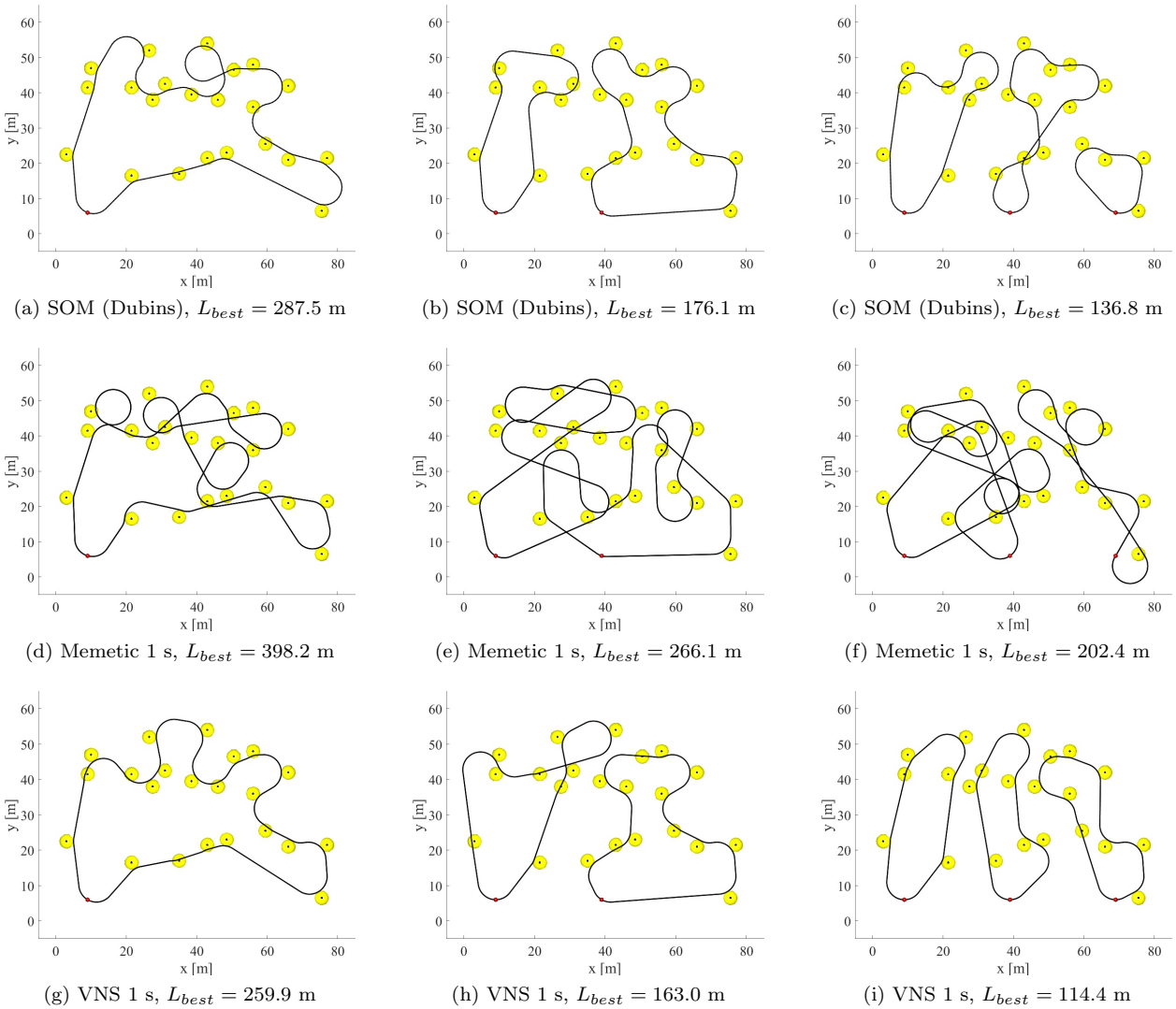
The Memetic algorithm is capable of providing high-quality solutions, but as it has been reported in other studies mentioned in the related work, it is computationally demanding. Even though only the single *mbzirc22* scenario is evaluated, the results indicate the VNS probably scales better with the number of vehicles than the Memetic algorithm. On the other, the solution improvement for increasing computational time is more evident for the Memetic algorithm than for the VNS which is highly related to the proposed initialization of the VNS. Therefore, the main observation from the results is that the proposed Initialization procedure (Section 5.1) performed prior the VNS optimization perfectly fits the properties of the *mbzirc22* scenario and our practical deployment.



(a) m-DTSP, $\delta = 0.0$ m (b) m-DTSPN, $\delta = 2.0$ m

Figure 13: Average lengths of the longest path found by the evaluated algorithms in the *mbzirc22* m-DTSPN scenario with the sensing range δ and m vehicles. The shown lengths are average values computed from 20 trials, and the standard deviations are shown as error bars, and very low values are not visible.

The performance of the algorithms in the m-DTSPN instances with sensing range $\delta = \{0.0, 2.0\}$ meters is depicted in Fig. 13 and the best solutions found by the selected algorithms are visualized in Fig. 14. Most of the algorithms provide slightly shorter solutions for increasing δ; however, two and three vehicles are more beneficial, and the longest tour is shortened more significantly than for a longer δ. Even though SOM provides better results than the Memetic 1 s (as it is reported in (Faigl and Váňa, 2017)), it can be noticed that the SOM-based approach provides a bit worse results for $\delta > 0$ than for $\delta = 0$, which is especially noticeable for $m = 2$. It is probably caused by marking the neurons within the δ distance from the target as



(a) SOM (Dubins), $L_{best} = 287.5$ m (b) SOM (Dubins), $L_{best} = 176.1$ m (c) SOM (Dubins), $L_{best} = 136.8$ m

(d) Memetic 1 s, $L_{best} = 398.2$ m (e) Memetic 1 s, $L_{best} = 266.1$ m (f) Memetic 1 s, $L_{best} = 202.4$ m

(g) VNS 1 s, $L_{best} = 259.9$ m (h) VNS 1 s, $L_{best} = 163.0$ m (i) VNS 1 s, $L_{best} = 114.4$ m

Figure 14: Selected best found solutions of the *mbzirc22* m-DTSPN scenario $\delta = 2$ m and with one (left), two (middle), and three (right) vehicles found by the evaluated algorithms with the computational time limited to one second

the winner without the adaptation as the neuron already covers the target. Besides, it can be related to the non-monotonicity of the length of Dubins maneuvers as such behavior is not observed in (Faigl and Hollinger, 2018) nor for Bézier curves, see Fig. 17. However, this phenomenon needs to be further investigated.

Table 3: CPU Time – SOM and Initialization of the VNS in m-DTSPN Scenarios

Problem	SOM – T_{CPU} [ms]			VNS Init – T_{CPU} [ms]		
	$m = 1$	$m = 2$	$m = 3$	$m = 1$	$m = 2$	$m = 3$
mbzirc22, $\delta = 0.0$	224.2	195.5	173.9	94.5	50.5	39.9
mbzirc22, $\delta = 0.5$	165.9	157.5	159.7	1 108.8	1 232.8	1 206.8
mbzirc22, $\delta = 1.0$	152.7	151.1	151.6	1 312.2	1 160.1	1 100.0
mbzirc22, $\delta = 2.0$	134.4	135.2	142.0	1 978.8	1 085.9	983.7

The required computational times of the SOM and the proposed Initialization procedure for the VNS are presented in Table 3. The SOM-based solver scales well with increasing m and δ. It is partially because the network converges in less number of learning epochs, but mostly because of saving the adaptation once a winner neuron is in the neighborhood of the particular target location. Decreasing computational requirements with m can be surprising, as algorithms are usually more demanding for multi-robot problems. However, the SOM benefits from the spatial allocation of the neurons and the total number of neurons is almost the same as for the single robot instances. Therefore, a selection of ν_{prev} and ν_{next} in the minimization of (18) is quicker because fewer neurons are in the ring and the most time-consuming operation is the computation of Dubins maneuvers. Notice, in SOM, Dubins maneuvers are computed on demand, and none of the precomputed distances are utilized because sampling of the heading and waypoint locations is performed on-line during the learning.

The SOM is far the fastest solver from the evaluated algorithms, especially for the m-DTSPN instances where the VNS initialization suffers from the sampled waypoint locations ($s = 6$) which make the construction of the initial tours demanding. For none zero sensing range δ, the initialization takes more than 1 second, and therefore, VNS 1 s does not satisfy the limit on the computational time. Also, less time is available for the VNS optimization for the higher limit of the computational time because of the demanding initialization. Besides, the optimization itself is also demanding because of the evaluation of the possible waypoint locations, and thus only a few iterations are performed, and the solution is not improving within the given time limit up to 60 seconds.

The Memetic algorithm provides worse results than SOM for one second limit, but it is capable of improving the solution if more computational time is available. However, it starts with a relatively poor solution, and even in 60 seconds, the solution is improved to be only close to the solution provided by the VNS solver.

Based on the reported results, it can be summarized that the proposed SOM-based approach for the m-DTSPN can be preferred whenever the computational requirements matters. It is a far way the fastest approach providing solutions in hundreds of milliseconds, and thus it perfectly fits real-time requirements. On the other hand, if the computational requirements are not limited, the proposed VNS-based approach is capable of providing best solutions. However, in the case of exploiting non-zero sensing range δ, depending on the selected number of samples of possible waypoint locations and heading values, the VNS-based algorithm can be quickly computationally demanding.

Real Deployment

Verification that the planned paths are feasible for the real vehicles has been performed in real experiments with three vehicles, and thus the setup of the experiment corresponds to the evaluated *mbzirc22* scenario with $m = 3$. Since the mutual trajectory collisions are not explicitly addressed in the m-DTSPN formulation, a solution without mutually crossing trajectories is selected from the found trajectories. It is not a big issue for the *mbzirc22* instances because the initial locations of the vehicles support splitting the field that is approximately 60 m $\times$ 80 m large, see Fig. 3. Besides, in our early results on the SOM-based planner (Faigl and Váňa, 2017), we further consider initial positions of the vehicles not only with different coordinates along the x-axis but also along the y-axis, see the small red disks denoting depots in Fig. 15. In addition, the SOM-based solver tends to find mutually non-crossing paths because of SOM property to preserve the topology of the input space (Faigl, 2016). The mutually non-crossing tours are not guaranteed, and this can be further addressed by adjusting velocity profiles which is out of the scope of the herein presented approach. Nevertheless, empirical results provide sufficient solutions that have been deployed in the field testing.

A snapshot of the planned and real trajectories is visualized in Fig. 15. The UAVs have been operating at the altitude of 7 meters with the trajectory following provided by the model predictive controller (MPC) (Báča et al., 2016). The RTK GPS with precision less than 2 cm has been utilized for controlling the UAVs and recording the real trajectories. The particular value of sensing range according to the camera field of view is 4 meters, but δ has been set to $\delta = 2$ m for the trajectory planning because of noise and

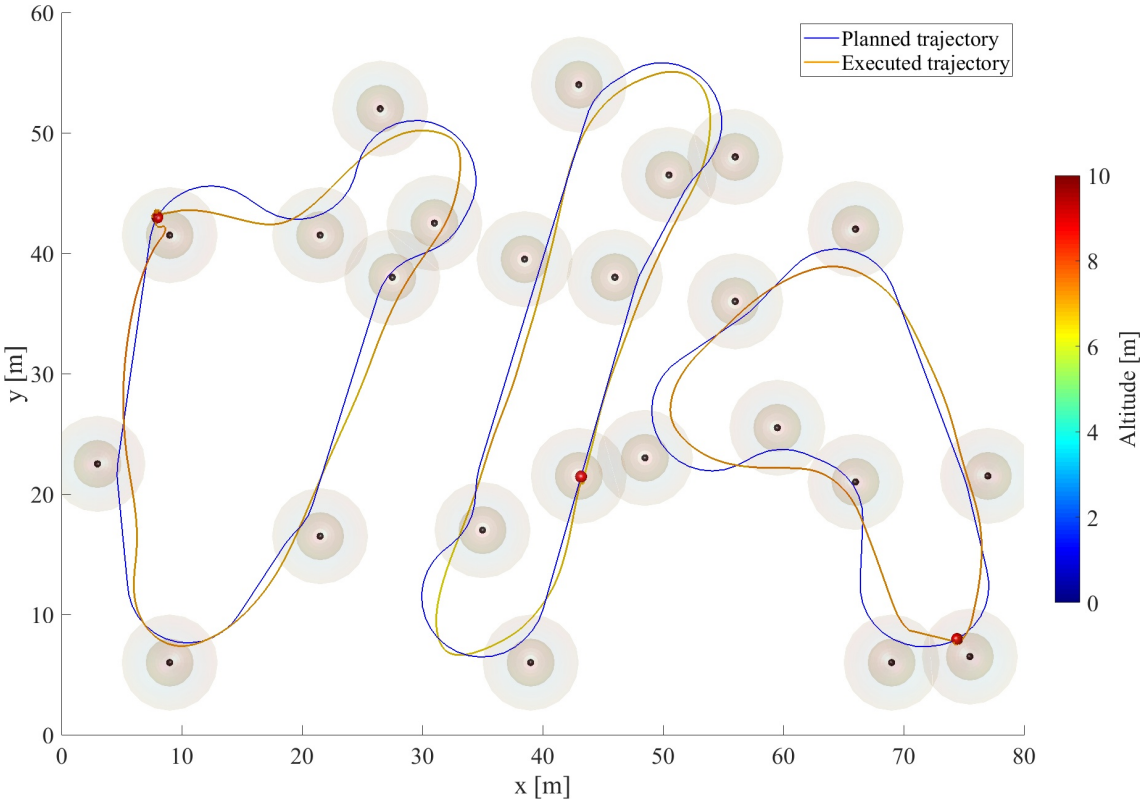


Figure 15: Planned and real executed trajectories by UAVs, results adopted from (Faigl and Váňa, 2017)

imperfections in the trajectory following, which can be observed in the recorded real trajectories. Even though the trajectory following is not perfect, a sufficient vicinity of the object of interest has been achieved. Besides, the real deployment of the proposed Dubins-based planning approach has been thoroughly validated during participation of the CTU team in MBZIRC 2017 (MBZIRC, 2017). Thus, a further step in the proposed approach for surveillance planning with UAVs is the utilization of Bézier curves as the trajectory parametrization.

7.2 Surveillance Planning with Dubins Maneuvers and Bézier Curves

The proposed extension of the unsupervised learning to the utilization of Bézier curves is compared with the Dubins vehicle model utilized in the SOM, Memetic, and VNS algorithms in a single vehicle scenario. The planning with Bézier curves directly optimizes the TTE using the velocity profiles computed according to the algorithm presented in Section 4.2. On the other hand, the Dubins vehicle model assumes a constant forward velocity v, and therefore, the TTE can be computed from the path length and v. However, the velocity depends on the allowed radial acceleration when the vehicle follows the circular path with the minimal turning radius ρ. We consider the maximal allowed horizontal acceleration $a_{horiz} = 2$ m.s^{-2} for which the forward velocity is computed as $v = \sqrt{\rho a_{horiz}}$. Hence, the vehicle can travel at the maximal horizontal velocity $v_{horiz} = 5$ m.s^{-1} along Dubins maneuvers with $\rho = 12.5$ m, which may provide longer paths. Shorter paths can be determined for shorter ρ, but the vehicle needs to travel with a lower velocity. Therefore, we consider $5.0 \leq \rho \leq 12.5$ and determine the most suitable ρ for the *mbzirc22* instance of the DTSP regarding the TTE.

In addition to such a simple computation of the TTE along the Dubins path with a constant forward velocity, we determine a faster trajectory considering the motion of the hexacopter is limited by the maximal

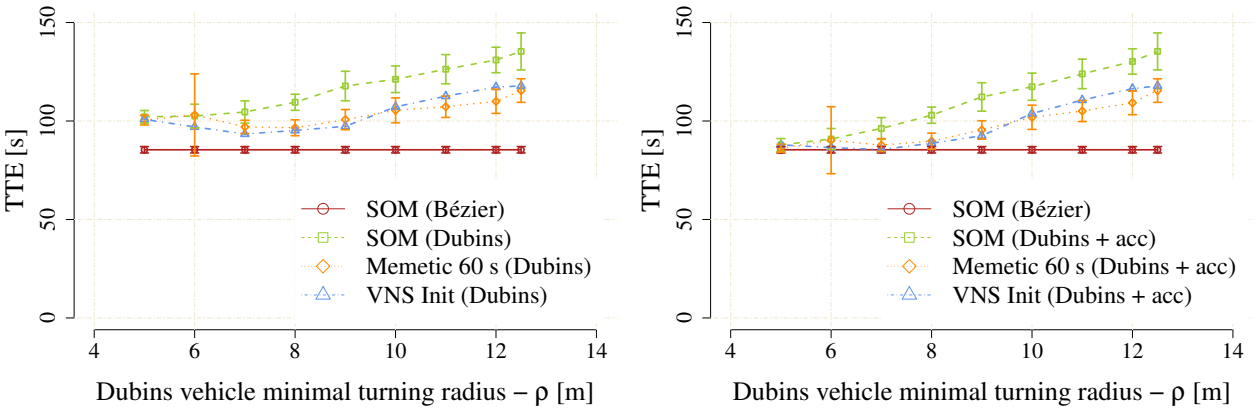
acceleration and not by the minimal turning radius. Therefore, the hexacopter can accelerate on the straight segments of the Dubins path, and a lower velocity v_{turn} can be computed for the turning segments according to the maximal allowed horizontal acceleration a_{horiz} as

$$v_{turn} = \min\left(v_{horiz}, \frac{a_{horiz}}{\kappa_h}\right), \tag{27}$$

where κ_h is the horizontal curvature (11) of the trajectory. For the turning radius ρ, (27) can be expressed as

$$v_{turn} = \min\left(v_{horiz}, a_{horiz}\rho\right). \tag{28}$$

The average values of the TTE for different ρ are depicted in Fig. 16, where the simple computation of the TTE based on constant forward velocity is denoted (Dubins), and the results for the acceleration on straight segments are denoted (Dubins + acc). Notice that for the VNS, only the initialization part is performed because of $m = 1$.



(a) A constant forward velocity

(b) Acceleration/deceleration up to v_{horiz}

Figure 16: Average values of the TTE along the planned trajectories for the Dubins vehicle with different minimal turning radius ρ. The SOM-based framework allows usage of the Dubins vehicle model denoted as the SOM (Dubins) and Bézier curves denoted as the SOM (Bézier).

The fastest trajectories are provided by the SOM-based planning with Bézier curves. The results further indicate that for the considered *mbzirc22* scenario, the fastest Dubins paths are provided by the proposed initialization of the VNS-based algorithm with $\rho = 7$ m, for which the TTE is a bit shorter (85.5 s) than the found trajectory consisting of Bézier curves with TTE = 86.6 s. However, suitability of the particular value of ρ depends on the configuration of the target locations as it is indicated by shorter ρ and trajectories determined by the other solvers with the acceleration of the vehicle on the straight line segments. Therefore Bézier curves seem to be a more suitable option than the Dubins vehicle model. Here, it is worth noting that the proposed SOM-based planning with Bézier curves converges better than with the Dubins maneuvers with non-monotonicity of the maneuver length, which can also be the reason for better solutions found by the unsupervised learning than the simple heuristic used in the VNS Init.

The surveillance planning with Bézier curves is further evaluated in instances of the *mbzirc22* scenario with increasing sensing range δ and m vehicles. The average value of the TTE together with the real required computational times are depicted in Fig. 17. Also, in this case, adding more vehicles to the team decreases TTE more significantly than increasing δ, but an improvement for a longer sensing range is noticeable, see Fig. 18. The computational requirements of the unsupervised learning are almost about two orders of magnitude higher than using the Dubins vehicle model. It is mainly because Dubins maneuvers are determined analytically while a numerical optimization is utilized for the optimization of Bézier curves. Nevertheless, solutions are provided in less than 15 seconds using the conventional computational resources.

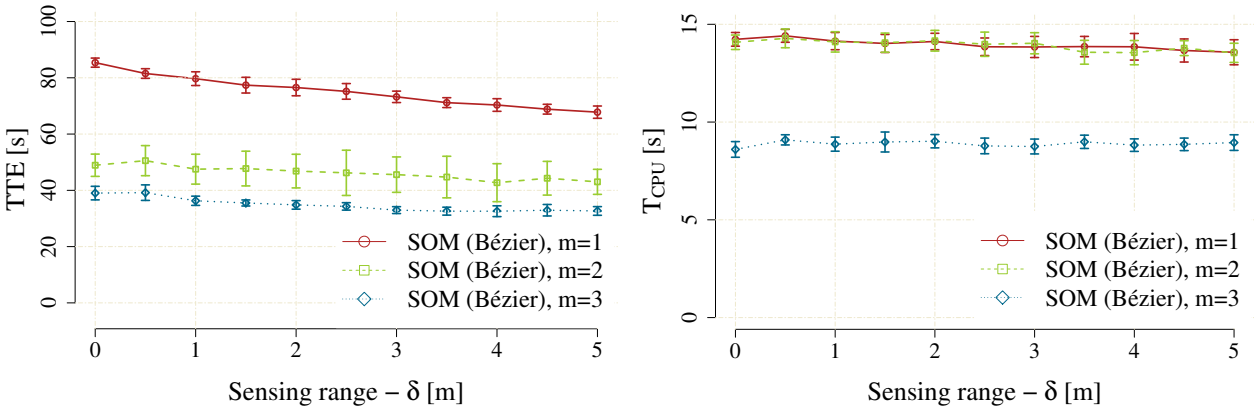
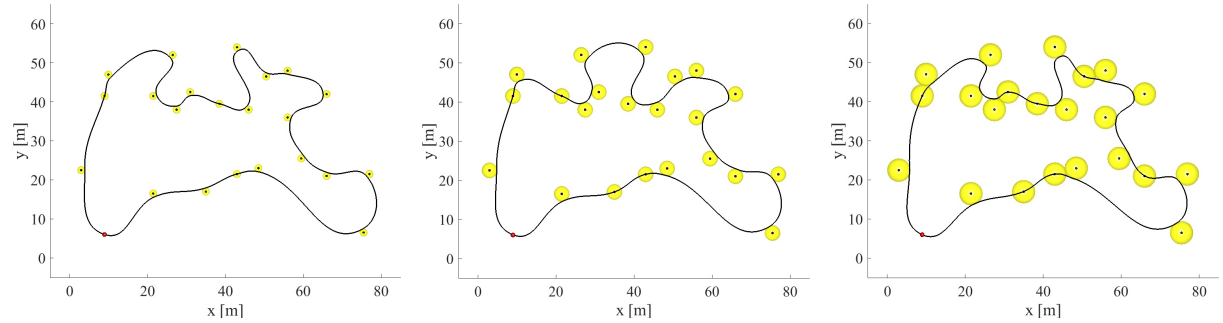


Figure 17: Average values of the TTE (left) and required computational times (right) for the proposed SOM-based surveillance planning with Bézier curves



(a) δ=1 m, TTE= 77 s (b) δ=2 m, TTE= 70.8 s (c) δ=3 m, TTE= 69.6 s

Figure 18: Selected found solutions for the proposed SOM-based surveillance planning with Bézier curves for a single vehicle ($m = 1$) and various sensing ranges

Real experimental verification is not performed for the 2D Bézier curves and velocity profiles of the Dubins tours because of the utilized MPC-based controller (Báča et al., 2016) which guarantees the trajectories are finished at the desired time. Therefore, Bézier curves are further evaluated in the 3D surveillance scenarios in the next section.

7.3 Performance Evaluation in 3D Surveillance Scenarios

The usage of Bézier curves provides a great advantage in a direct deployment of the proposed unsupervised learning based planning in 3D surveillance scenarios. The testing scenario *mbzirc22* has been extended to 3D by adding altitudes to the particular target locations. It can be expected that a high variance in the target altitudes would need a longer trajectory than the 2D scenario. Therefore we consider two scenarios with different ranges of the altitude changes to study limits of the maximum horizontal velocity v_{horiz} and the maximal vertical velocity as v_{vert}. For low altitude changes in the range of [5, 10] meters, the vehicle mostly needs to travel horizontally, and thus it is expected the vehicle velocity will be saturated at v_{horiz} more frequently than for high altitude changes of the target locations in the range of [5, 20] meters, where the vehicle needs to change the altitude, and thus it is limited by v_{vert}. The considered horizontal limits are the same as for the 2D planning, i.e., $v_{horiz} = 5$ m.s^{-1} and $a_{horiz} = 2$ m.s^{-2}. The vertical limits correspond to the capabilities of the used real UAVs and are set to $v_{vert} = 1$ m.s^{-1} and $a_{vert} = 1$ m.s^{-2}.

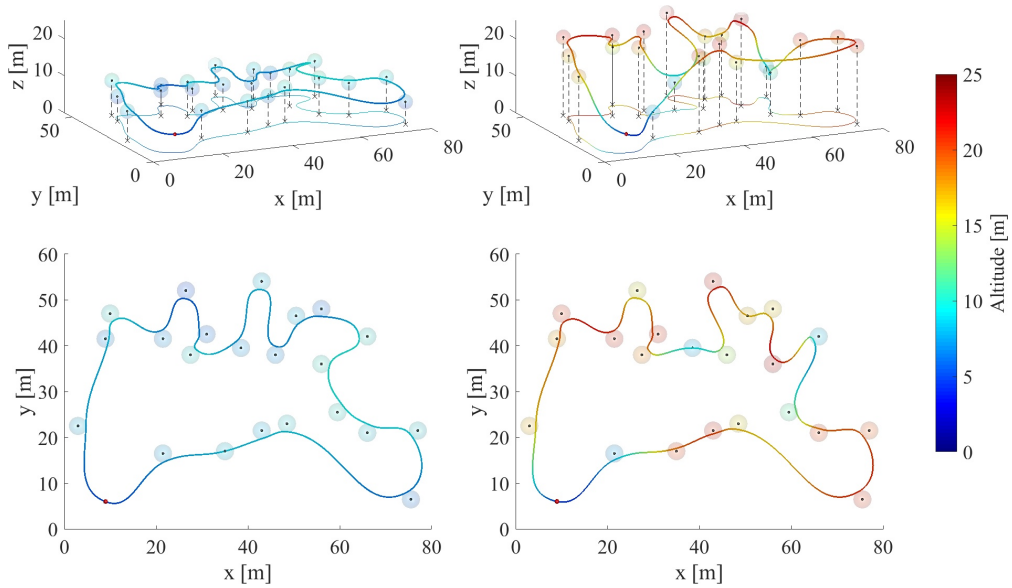
An example of the 3D surveillance scenarios with low and high altitude changes in the target locations created

from the *mbzirc22* scenario is depicted in Fig. 19 together with the horizontal position and altitude along the trajectories and the corresponding velocity profiles. As expected, increasing altitude changes increases the time needed to travel along the trajectory, and therefore, we validated the trajectories in a real experimental deployment.

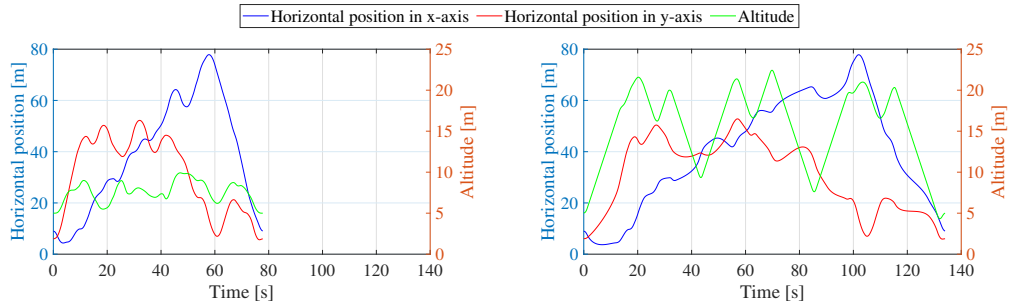
Experimental Validation of the 3D Surveillance Planning

Feasibility of 3D trajectories planned by the proposed SOM-based framework with Bézier curves has been validated in a real experiment with three vehicles and modified *mbzirc22* scenario with the target locations at different altitudes. The same hardware and GPS-based localization as in the validation of the Dubins tours have been utilized. A snapshot from the field experiment is depicted in Fig. 21.

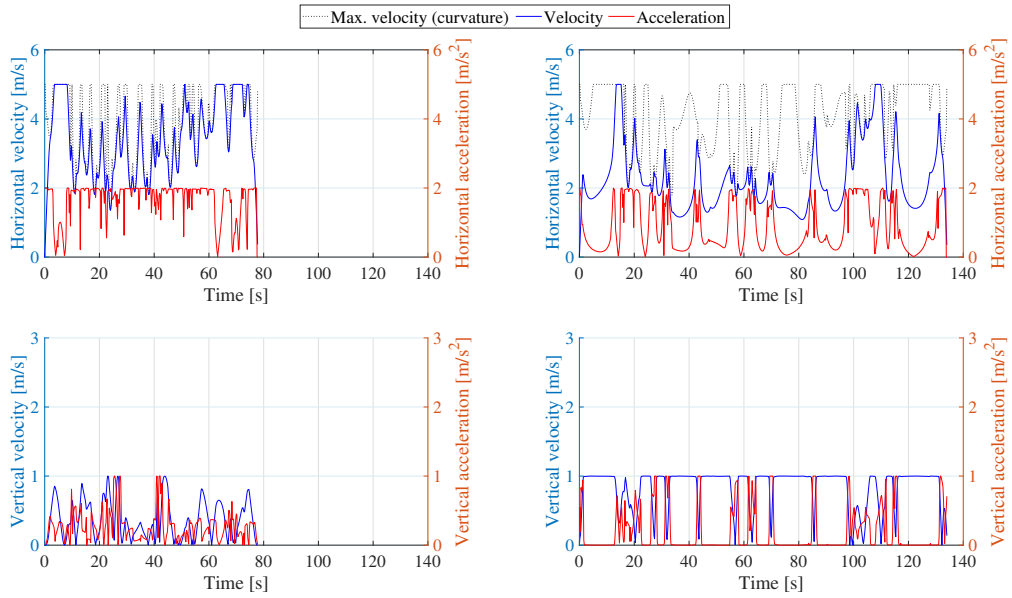
The evaluated scenario with the sensing range $\delta = 2$ m together with the planned and real trajectories of the vehicles from two trials are depicted in Fig. 20 with the corresponding planned and real velocity profiles presented in Fig. 22. Also in this real deployment, the MPC-based trajectory following controller (Báča et al., 2016) has been utilized, and all the planned trajectories have been found feasible. Besides, all the target locations have been visited within the requested distance, which is δ+2 meters because of the identified behavior of the MPC controller as in the previous case. The vehicles reached their velocity limits in both direction, i.e., v_{horiz} and v_{vert}, which is indicated in Fig. 22.



(a) 3D Surveillance scenarios with low (left) and high (right) altitude changes and found solutions



(b) A position of the vehicle along the trajectories



(c) Velocity and acceleration profiles for vehicles traveling along the determined trajectories

Figure 19: An example of the 3D surveillance problems with low (left) and high (right) altitude changes in the target locations of the *mbzirc22* scenario and solutions found by the proposed SOM-based algorithm with Bézier curves (top), particular positions of the vehicles along the found paths (middle), and velocity and acceleration profiles (bottom)

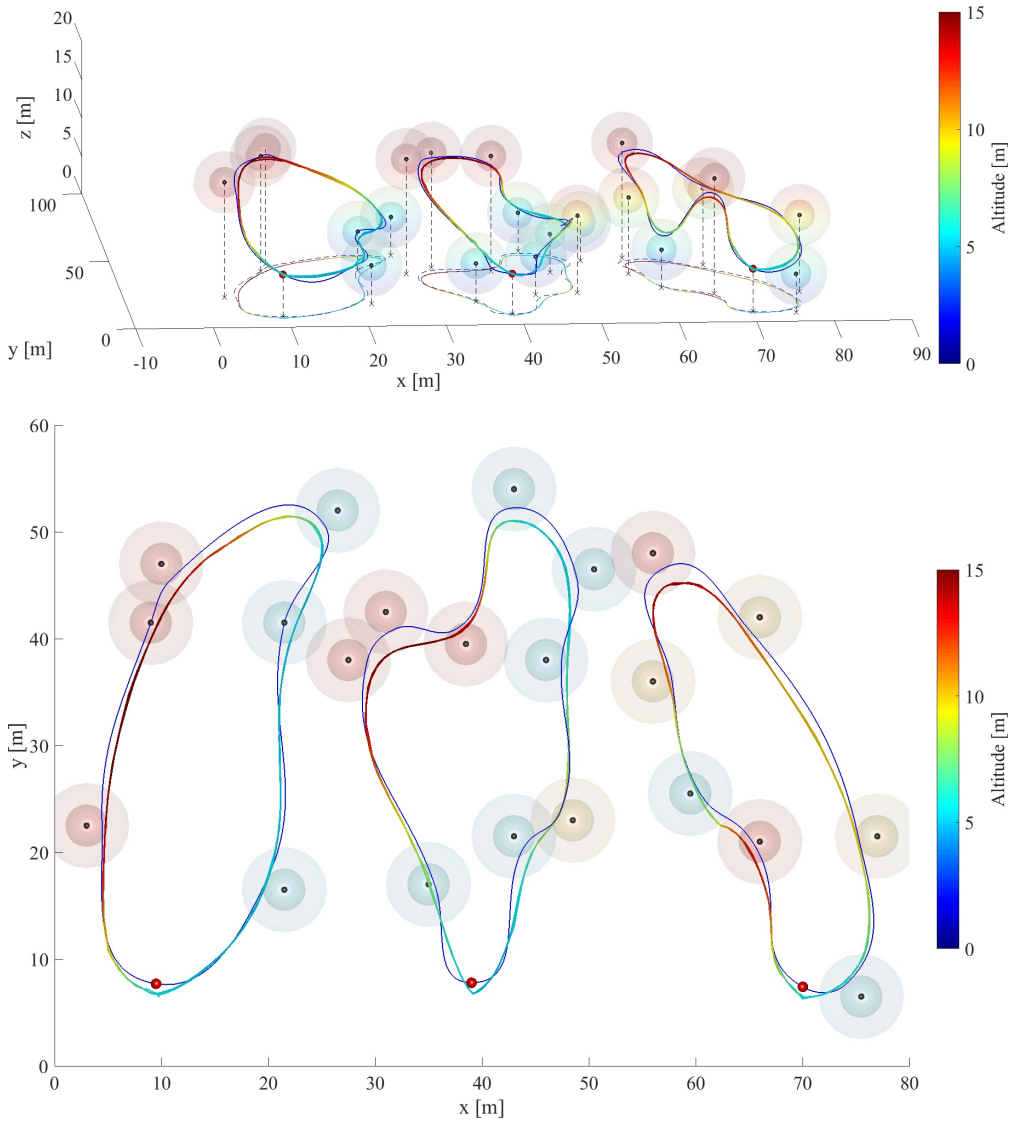


Figure 20: Planned and real trajectories executed simultaneously on three hexacopters. The target locations of the *mbzirc22* scenario are depicted as small black spheres each surrounded by its spherical neighborhood $\delta = 2$ m. The three additional initial locations of the vehicles are positioned at the altitude of 5 m, and they are visualized as small red spheres. The planed trajectories are shown by blue curves, and the trajectories from two experimental trials are visualized by curves with the color based on the altitude (from blue to red for increasing altitude).

Figure 21: A snapshot of the three UAVs deployed in the experimental verification of following the planned 3D trajectories

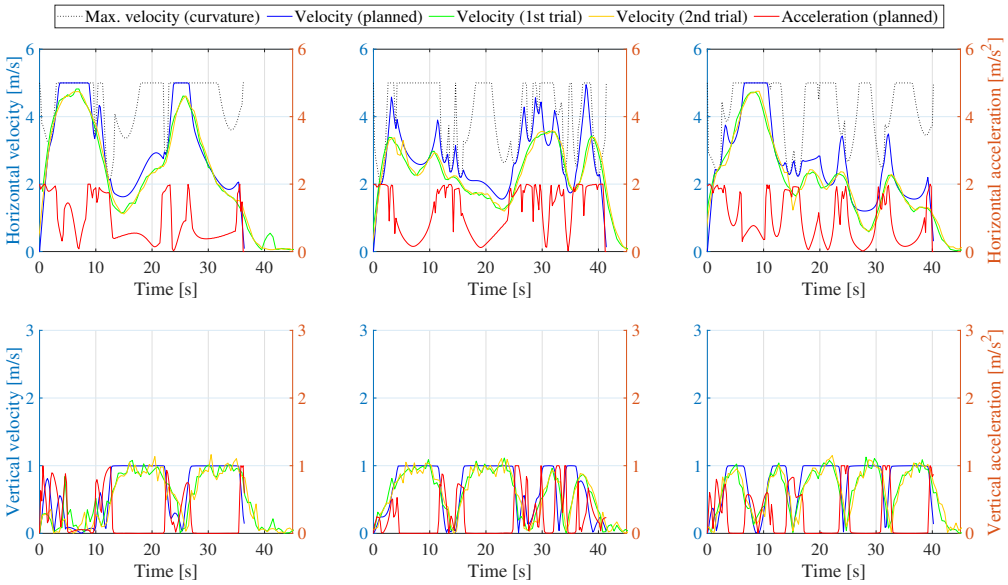


Figure 22: Planned and real horizontal (top) and vertical (bottom) velocity profiles from two experimental trials for each of the vehicle. Each column corresponds to one vehicle according to Fig. 20 (from left to right).

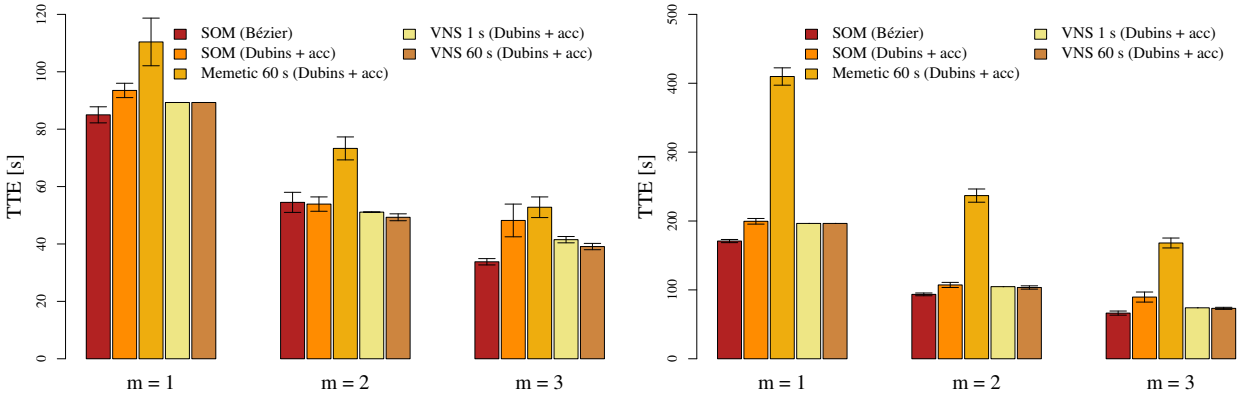
7.4 Performance Evaluation in Complex Instances

In this part of the presented results, we report on an overview of the performance of the evaluated algorithms in instances that are beyond the motivational *mbzirc22* scenario with the aim to show particular properties that may not be directly visible from the previous results. First, we focus on the proposed initialization of the VNS algorithm, which seems to be fast and powerful in the m-DTSP instances but it becomes quite demanding in m-DTSPN with $\delta > 0$ as it is shown in Table 3. Moreover, it is even more demanding in the instances where the initial vehicle position is not a single waypoint location, but it is considered with the same neighborhood δ as the other target locations. The required computational times for the *mbzirc22* scenario with $\delta = 2$ m and m vehicles is depicted in Table 4. SOM-based solvers are not influenced by $\delta > 0$, but the VNS initialization is several times more demanding when the neighborhood is considered for the initial locations of the vehicles, i.e., $\delta_d = \delta$, and thus the solution is not provided in less than the desired one second.

Table 4: Influence of Vehicle Initial Locations with Neighborhood in *mbzirc22* and $\delta = 2.0$ m

Depot	SOM (Bézier) – T_{CPU} [s]			SOM (Dubins) – T_{CPU} [s]			VNS Init – T_{CPU} [s]		
δ_d [m]	$m = 1$	$m = 2$	$m = 3$	$m = 1$	$m = 2$	$m = 3$	$m = 1$	$m = 2$	$m = 3$
$\delta_d = 0$	8.5	8.7	8.9	0.1	0.1	0.1	2.0	1.1	1.0
$\delta_d = 2$	8.3	9.3	8.9	0.2	0.2	0.1	5.3	6.2	5.5

An additional evaluation is focused on the solution of large problems since the *mbzirc22* scenario is relatively small. Therefore the solvers have been deployed in two random instances with 50 and 100 relatively dense target locations. In this case, the minimal turning radius for the Dubins vehicle is set to $\rho = 1$ m, but all other parameters are the same as in Section 7.2.

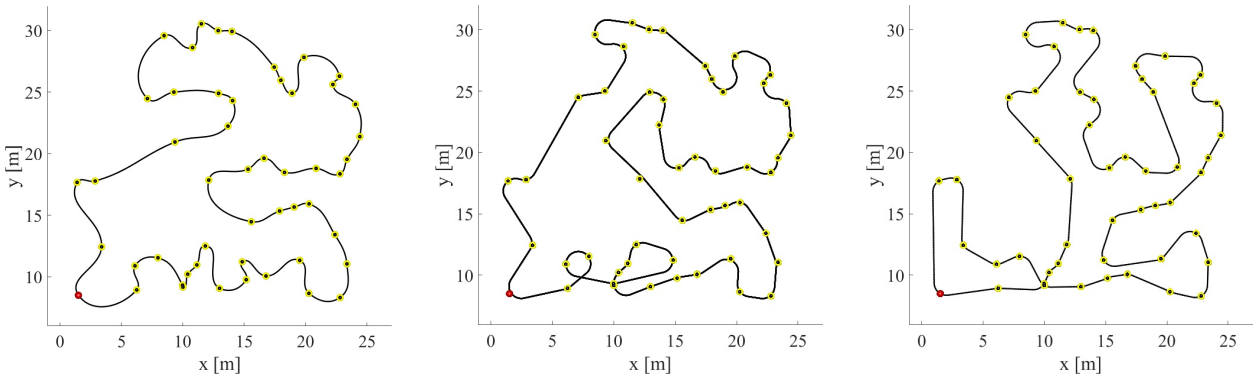


(a) m-DTSP instance with $n = 50$ (b) m-DTSP instance with $n = 100$

Figure 23: Average values of the TTE for large problems with n target locations

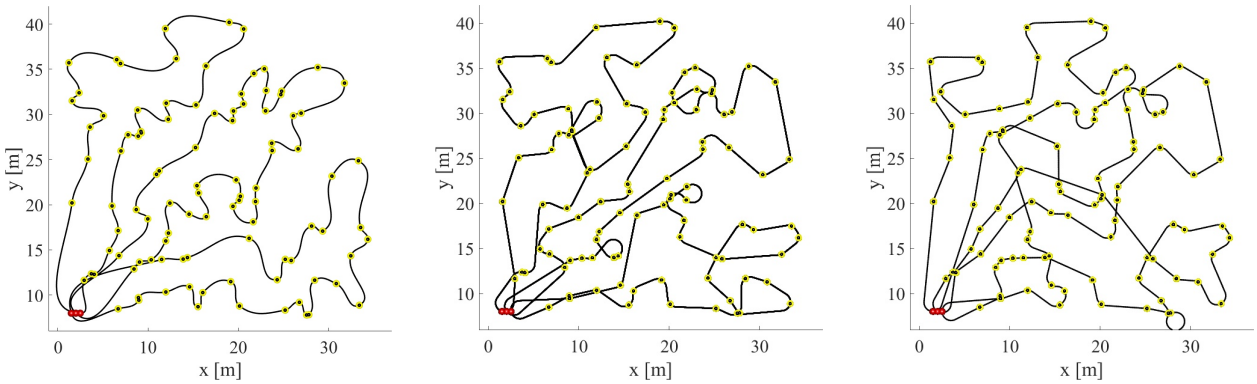
The average values of the TTE using acceleration/deceleration for the Dubins maneuvers are shown in Fig. 23 and selected solutions are depicted in Fig. 24 for $n = 50$ and in Fig. 25 for $n = 100$ target locations. The results indicate that the fastest trajectories are provided by the SOM solver with Bézier curves. The Memetic algorithm is not competitive with the proposed SOM nor the VNS algorithm. Notice, for large instances, the initialization of the VNS can be more demanding, and thus more than the dedicated computational time can be spent on the creation of the first feasible solution, see Fig. 26.

Regarding the computational requirements, the best trade-off between the solution quality and computational time is provided by the VNS-based solver or more precisely by the Initialization procedure proposed in



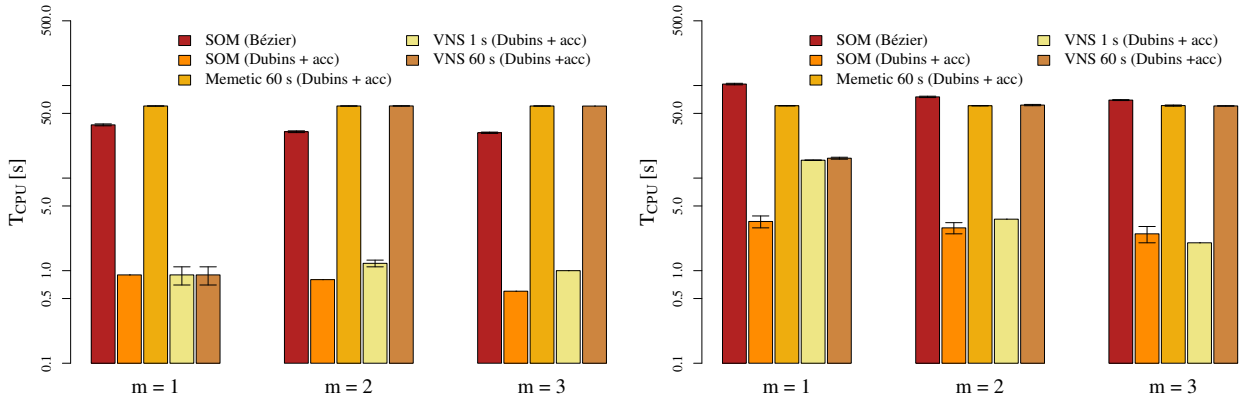
(a) SOM (Bézier), $TTE = 80.2$ s (b) SOM (Dubins), $TTE = 87.6$ s (c) VNS 60s (Dubins), $TTE = 89.3$ s

Figure 24: Selected found solutions for the problem with $n = 50$ target locations and one vehicle ($m = 1$)



(a) SOM (Bézier), $TTE = 55.2$ s (b) SOM (Dubins), $TTE = 72.3$ s (c) VNS 60 s (Dubins), $TTE = 70.8$ s

Figure 25: Selected best found solutions for the problem with $n = 100$ target locations and three vehicles ($m = 3$)



(a) m-DTSP instance with $n = 50$ (b) m-DTSP instance with $n = 100$

Figure 26: Average required computational time for large problems with n target locations

Section 5.1. For the larger problem with $n = 100$, the VNS initialization takes about 15 seconds while SOM with Dubins maneuvers takes only about three seconds for $m = 1$. However, for more vehicles, the VNS initialization is less demanding, and it is competitive to the SOM with the Dubins vehicle model. The

heuristic initialization of the VNS works faster with more vehicles because the evaluation of all possible insertions is faster for tours with fewer targets.

7.5 Discussion

Based on the presented results, the SOM-based algorithm scales better in the problems with non-zero sensing range, but for the m-DTSP, the superior results are provided by the proposed heuristic initialization employed for the initial construction of a feasible solution prior a further improvement by the VNS optimization. A great benefit of the SOM solver is its flexibility to utilize Bézier curves that allow for exploiting motion capabilities of the used hexacopters which are not limited by a minimal turning radius ρ as the Dubins vehicle. Comparing parametrization of the requested surveillance trajectories using Dubins vehicle model and the proposed sequence of Bézier curves, the main advantage of Dubins maneuvers is the closed form solution for two waypoints with prescribed headings, which leads to lower computational requirements. On the other hand, the Bézier curves better fit the real limitations of the multi-rotor vehicles that are not limited by minimal turning radius, but by the maximal vehicle velocity and acceleration limits.

In small scenarios such as *mbzirc22*, it can be possible to find the best performing radius ρ for which Dubins maneuvers provide similar TTE as the Bézier curves, but only if velocity profiles are determined with the allowed acceleration/deceleration up to v_{horiz}. For larger instances, it may not be beneficial because of computational requirements for solving the m-DTSPN for several values of ρ would be similar or higher than a solution using the Bézier curves. Finally, the main advantage of the Bézier curves is a direct generalization of the surveillance planning to the 3D, which is not directly possible for the Dubins vehicle. The Dubins Airplane model (Chitsaz and LaValle, 2007) can be used for 3D trajectory planning; however, shorter and faster trajectories will be found using the proposed Bézier curves for multi-rotor vehicles that do not need to use the additional spiral to gain the requested altitude, as these vehicles can directly flight almost in any direction. Therefore, if the configuration of the planning problem is known in advance and enough time to compute several solutions is available, which is not the case of the robotic competition, it can be suitable to consider the Dubins vehicle model. In other cases and especially 3D planning with $\delta > 0$, the proposed SOM-based unsupervised learning framework with Bézier curves is a suitable choice.

The reported evaluation results are only for problem instances with up to three vehicles because of our motivation arising from the MBZIRC 2017 competition, where we deployed only three vehicles. All the proposed and evaluated algorithms for m-DTSPN including the SOM-based method for surveillance planning with Bézier curves can solve problems with a higher number of vehicles; however, we do not consider such scenarios because of the scope of this paper and challenging experimental verification, e.g., with 10 vehicles, that needs significantly larger and more demanding experimental setup. Regarding scalability of the employed SOM-based unsupervised learning, it is worth mentioning that it can be considered as independent on the number of vehicles if the number of target locations n is significantly higher than the number of vehicles m, i.e., $n \gg m$, see the analysis in (Best et al., 2018).

For multi-robot deployment, an important part of the surveillance planning is collision avoidance. In the presented approach, we do not include the collision avoidance explicitly in the planning part because it is addressed in MPC-based controller employed in the trajectory following which is considered to be out of the scope of this paper. The reactive collision avoidance based on the MPC predictions uses a slight alteration of the desired trajectory altitude if the MPC predictions contain collisions between the vehicles. The employed MPC-based collision avoidance is partially described in (Spurný et al., 2018) and it is presented in (Báča et al., 2018). Besides, the found solutions and especially those found by the proposed unsupervised learning are such that the found trajectories are mutually non-crossing, and thus collision-free, see a discussion on that in (Faigl, 2016). Although non-crossing trajectories are not guaranteed; such solutions are found with a high probability in the considered scenarios also because of the selection of the vehicle depots that have to respect safety zones around each vehicle.

Regarding future work related to the proposed solvers, there are several open questions in addition to the

explicit consideration of collision-free trajectories. One of them is that the proposed VNS initialization exhibits surprisingly good results and since it seems the VNS optimization scales poorly with increasing computational time, such an initial solution can be fed to the Memetic algorithm for further improvement. For large instances with tens, hundreds, and more target locations, the unsupervised learning with Bézier curves seems to scale better than the VNS, and there are two ways how the computation can be further speeded up. The first is to improve the local optimization. The second is to exploit parallelization of the unsupervised learning of SOM, which has been already reported in the literature including SOM for the TSP.

The promising results of the Dubins vehicle model with various ρ accompanied with the computation of the velocity profile for hexacopters provide a source of motivation for generalization of the proposed approaches to consider multiple radii simultaneously during the optimization. In addition, a further generalization of the Dubins vehicle model employed in the proposed solvers towards the 3D Airplane model or Dubins-Helix model is also a possible subject for the future work.

8 Conclusion

In this paper, we address surveillance planning problem motivated by our participation in the robotic competition MBZIRC 2017. Because of the motivation, we aim to quickly find a solution to the planning problem with satisfiable quality, and thus we focus on the heuristic solution rather than optimal algorithms. The problem is firstly tackled as a variant of the m-DTSPN with the Dubins vehicle model for satisfying curvature-constrained trajectories that fit properties of the utilized trajectory follower. The m-DTSPN has been addressed by the proposed VNS-based and SOM-based algorithms that are significantly less demanding than the existing Memetic algorithm, and both proposed algorithms provide better solutions in less computational time. However, Dubins vehicle model is suitable for vehicles with the limited turning radius that is not necessarily the case of the used hexacopters whose motion is constrained by the maximal velocity and acceleration limits. Enabled by the flexibility of the employed unsupervised learning, we propose to consider a more general trajectory parametrization based on Bézier curves, which enable to better exploit motion capabilities of the used vehicles. Moreover, it also allows solving 3D surveillance planning missions and finding 3D smooth trajectories for a team of our hexacopters. The solutions found by the proposed algorithms have been numerically evaluated in several realistic problem instances. Besides, the solutions have also been experimentally verified by a real multi-robotic system, where all the provided trajectories have been found feasible, and they fit the properties of the utilized trajectory following controller of the used hexacopters.

Acknowledgments

The authors would like to thank to Vojtěch Spurný and Tomáš Báča for their assistance with the experimental deployment of the presented method with the real UAVs in the field and also to organizers of the MBZIRC competition.

The presented work was supported by the Czech Science Foundation (GAČR) under research project No. 16-24206S. The authors acknowledge the support of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics”.

References

Ahmad, R. and Kim, D. (2015). An Extended Self-Organizing Map based on 2-opt algorithm for solving symmetrical Traveling Salesperson Problem. *Neural Computing and Applications*, 26(4):987–994.

Angéniol, B., de la C. Vaubois, G., and Texier., J.-Y. L. (1988). Self-organizing feature maps and the travelling salesman problem. *Neural Networks*, 1:289–293.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2003). Concorde TSP Solver. `http://www.tsp.gatech.edu/concorde.html`, (Accessed 4 Aug 2017).

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA.

Báča, T., Hert, D., Loianno, G., Saska, M., and Kumar, V. (2018). Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. In *IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*. `http://mrs.felk.cvut.cz/data/papers/baca-mpc-tracker.pdf`, (accepted).

Báča, T., Loianno, G., and Saska, M. (2016). Embedded model predictive control of unmanned micro aerial vehicles. In *21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 992–997.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219.

Bellmore, M. and Hong, S. (1974). Transformation of Multisalesman Problem to the Standard Traveling Salesman Problem. *Journal of the ACM*, 21(3):500–504.

Best, G., Faigl, J., and Fitch, R. (2018). Online planning for multi-robot active perception with self-organising maps. *Autonomous Robots*, 42(4):715–736.

Bézier, P. (1973). Numerical control: Mathematics and applications. *International Journal for Numerical Methods in Engineering*, 6(3):456–456.

Chitsaz, H. and LaValle, S. M. (2007). Time-optimal paths for a dubins airplane. In *46th IEEE Conference on Decision and Control (CDC)*, pages 2379–2384.

Cochrane, E. M. and Beasley, J. E. (2003). The co-adaptive neural network approach to the Euclidean travelling salesman problem. *Neural Networks*, 16(10):1499–1525.

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, 6(6):791–812.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, pages 497–516.

Faigl, J. (2016). An application of self-organizing map for multirobot multigoal path planning with minmax objective. *Computational Intelligence and Neuroscience*, pages 2720630:1–2720630:15.

Faigl, J. (2018). GSOA: growing self-organizing array - unsupervised learning for the close-enough traveling salesman problem and other routing problems. *Neurocomputing*, 312:120–134.

Faigl, J. and Hollinger, G. A. (2018). Autonomous data collection using a self-organizing map. *IEEE Transactions on Neural Networks and Learning Systems*, 29(5):1703–1715.

Faigl, J. and Váňa, P. (2016). Self-organizing map for the curvature-constrained traveling salesman problem. In *International Conference on Artificial Neural Networks*, pages 497–505. Springer International Publishing.

Faigl, J. and Váňa, P. (2017). Unsupervised learning for surveillance planning with team of aerial vehicles. In *International Joint Conference on Neural Networks (IJCNN)*, pages 4340–4347.

Faigl, J., Váňa, P., Saska, M., Báča, T., and Spurný, V. (2017). On solution of the dubins touring problem. In *European Conference on Mobile Robots (ECMR)*.

Fort, J. C. (1988). Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem. *Biological Cybernetics*, 59(1):33–40.

França, P. M., Gendreau, M., Laporte, G., and Müller, F. M. (1995). The m-Traveling Salesman Problem with Minmax Objective. *Transportation Science*, 29(3):267–275.

Goaoc, X., Kim, H.-S., and Lazard, S. (2013). Bounded-curvature shortest paths through a sequence of points using convex optimization. *SIAM Journal on Computing*, 42(2):662–684.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467.

Helsgaun, K. (2000). An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic. *European Journal of Operational Research*, 126(1):106–130.

Isaacs, J. T. and Hespanha, J. P. (2013). Dubins traveling salesman problem with neighborhoods: a graph-based approach. *Algorithms*, 6(1):84–99.

Isaacs, J. T., Klein, D. J., and Hespanha, J. P. (2011). Algorithms for the Traveling Salesman Problem with Neighborhoods Involving a Dubins Vehicle. In *American Control Conference*, pages 1704–1709.

Isaiah, P. and Shima, T. (2015). Motion planning algorithms for the Dubins Travelling Salesperson Problem. *Automatica*, 53:247–255.

Jolly, K., Sreerama Kumar, R., and Vijayakumar, R. (2009). A Bezier curve based path planning in a multi-agent robot soccer system without violating the acceleration limits. *Robotics and Autonomous Systems*, 57(1):23–33.

Kohonen, T., Schroeder, M. R., and Huang, T. S., editors (2001). *Self-Organizing Maps*. Springer-Verlag New York, Inc., 3rd edition.

Kulich, M., Faigl, J., Kléma, J., and Kubalík, J. (2004). Rescue operation planning by soft computing techniques. In *IEEE 4th International Conference on Intelligent Systems Design and Application*, pages 103–108.

Le Ny, J., Feron, E., and Frazzoli, E. (2012). On the Dubins Traveling Salesman Problem. *IEEE Transactions on Automatic Control*, 57(1):265–270.

Lepetič, M., Klančar, G., Škrjanc, I., Matko, D., and Potočnik, B. (2003). Time optimal path planning considering acceleration limits. *Robotics and Autonomous Systems*, 45(3):199–210.

Ma, X. and Castanon, D. A. (2006). Receding horizon planning for Dubins traveling salesman problems. In *45th IEEE Conference on Decision and Control*, pages 5453–5458.

Macharet, D. G. and Campos, M. M. (2014). An orientation assignment heuristic to the dubins traveling salesman problem. In *Advances in Artificial Intelligence–IBERAMIA 2014*, pages 457–468. Springer.

Macharet, D. G., Monteiro, J. W., Mateus, G. R., and Campos, M. M. (2016). Time-Optimized Routing Problem for Vehicles with Bounded Curvature. In *XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium*, pages 145–150. IEEE.

Macharet, D. G., Neto, A. A., da Camara Neto, V. F., and Campos, M. M. (2011). Nonholonomic path planning optimization for dubins' vehicles. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4208–4213.

Macharet, D. G., Neto, A. A., da Camara Neto, V. F., and Campos, M. M. (2012). An evolutionary approach for the Dubins' traveling salesman problem with neighborhoods. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pages 377–384. ACM.

Macharet, D. G., Neto, A. A., da Camara Neto, V. F., and Campos, M. M. (2013). Efficient target visiting path planning for multiple vehicles with bounded curvature. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3830–3836.

Manyam, S. and Rathinam, S. (2015). A Tight Lower Bounding Procedure for the Dubins Traveling Salesman Problem. *arXiv preprint arXiv:1506.08752v2*.

Manyam, S., Rathinam, S., and Casbeer, D. (2016). Dubins paths through a sequence of points: Lower and upper bounds. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 284–291.

Manyam, S., Rathinam, S., Casbeer, D., and Garcia, E. (2015). Shortest Paths of Bounded Curvature for the Dubins Interval Problem. *arXiv preprint arXiv:1507.06980*.

MBZIRC (2017). Mohamed Bin Zayed International Robotics Challenge (MBZIRC). `http://www.mbzirc.com`, (Accessed 28 July 2017).

Neubauer, M. and Müller, A. (2015). Smooth orientation path planning with quaternions using B-splines. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2087–2092.

Ny, J., Feron, E., and Frazzoli, E. (2012). On the dubins traveling salesman problem. *IEEE Transactions on Automatic Control*, 57(1):265–270.

Oberlin, P., Rathinam, S., and Darbha, S. (2010). Today's traveling salesman problem. *Robotics & Automation Magazine, IEEE*, 17(4):70–77.

Obermeyer, K. (2009). Path planning for a uav performing reconnaissance of static ground targets in terrain. In *AIAA Guidance, Navigation, and Control Conference*, pages 10–13.

Obermeyer, K., Oberlin, P., and Darbha, S. (2010). Sampling-Based Roadmap Methods for a Visual Reconnaissance UAV*. In *AIAA Guidance, Navigation, and Control Conference*.

Obermeyer, K., Oberlin, P., and Darbha, S. (2012). Sampling-based path planning for a visual reconnaissance unmanned air vehicle. *Journal of Guidance, Control, and Dynamics*, 35(2):619–631.

Papadopoulos, E., Papadimitriou, I., and Poulakakis, I. (2005). Polynomial-based obstacle avoidance techniques for nonholonomic mobile manipulator systems. *Robotics and Autonomous Systems*, 51(4):229–247.

Pěnička, R., Faigl, J., Váňa, P., and Saska, M. (2017). Dubins orienteering problem. *IEEE Robotics and Automation Letters*, 2(2):1210–1217.

Pěnička, R., Faigl, J., Váňa, P., and Saska, M. (2017). Dubins orienteering problem with neighborhoods. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1555–1562.

Saska, M. (2017). MBZIRC team of the Czech Technical University. `http://mrs.felk.cvut.cz/projects/mbzirc`, (Accessed 28 July 2017).

Savla, K., Frazzoli, E., and Bullo, F. (2005). On the point-to-point and traveling salesperson problems for Dubins' vehicle. In *Proceedings of the American Control Conference*, pages 786–791. IEEE.

Somhom, S., Modares, A., and Enkawa, T. (1997). A self-organising model for the travelling salesman problem. *Journal of the Operational Research Society*, pages 919–928.

Somhom, S., Modares, A., and Enkawa, T. (1999). Competition-based neural network for the multiple travelling salesmen problem with minmax objective. *Computers & Operations Research*, 26(4):395–407.

Soylu, B. (2015). A general variable neighborhood search heuristic for multiple traveling salesmen problem. *Computers & Industrial Engineering*, 90:390–401.

Spurný, V., Báča, T., Saska, M., Pěnička, R., Thomas, J., Thakur, D., Loianno, G., and Kumar, V. (2018). Cooperative autonomous search, grasping and delivering in a treasure hunt scenario by a team of uavs. *Journal of Field Robotics*. `http://mrs.felk.cvut.cz/data/papers/mbzirc-treasure-hunt2017.pdf`, (accepted).

Váňa, P. and Faigl, J. (2015). On the dubins traveling salesman problem with neighborhoods. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4029–4034.

Váňa, P., Sláma, J., and Faigl, J. (2018). The Dubins Traveling Salesman Problem with Neighborhoods in the Three-Dimensional Space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 374–379.

Wang, Y., Wang, S., and Tan, M. (2015a). Path Generation of Autonomous Approach to a Moving Ship for Unmanned Vehicles. *IEEE Transactions on Industrial Electronics*, 62(9):5619–5629.

Wang, Y., Wang, S., Tan, M., Zhou, C., and Wei, Q. (2015b). Real-time dynamic Dubins-Helix method for 3-D trajectory smoothing. *IEEE Transactions on Control Systems Technology*, 23(2):730–736.

Yang, K. and Sukkarieh, S. (2010). An Analytical Continuous-Curvature Path-Smoothing Algorithm. *IEEE Transactions on Robotics*, 26(3):561–568.

Yu, X. (2015). *Optimization Approaches for a Dubins Vehicle in Coverage Planning Problem and Traveling Salesman Problems*. PhD thesis, Auburn University.

Yu, X. and Hung, J. (2012). A genetic algorithm for the dubins traveling salesman problem. In *IEEE International Symposium on Industrial Electronics*, pages 1256–1261.

Zhang, X., Chen, J., Xin, B., and Peng, Z. (2014). A memetic algorithm for path planning of curvature-constrained uavs performing surveillance of multiple ground targets. *Chinese Journal of Aeronautics*, 27(3):622–633.