JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 31, No. 5, September–October 2008

# Unmanned Aerial Vehicles Cooperative Tracking of Moving Ground Target in Urban Environments

Vitaly Shaferman* and Tal Shima†
*Israel Institute of Technology, 32000 Haifa, Israel*

DOI: 10.2514/1.33721

**The problem of autonomous tracking of a ground moving target in an urban terrain is studied. In the investigated scenario, the target is tracked from multiple unmanned aerial vehicles using gimballed or body-fixed sensors under the constraints of terrain occlusions and airspace limitations. Information regarding the occlusions may be available a priori from a database or may be provided to the system by the operator based on his understanding of the environment. To ensure flyable trajectories, the unmanned aerial vehicles' dynamic constrains must be taken into account. A methodology is proposed for solving in real time a general class of such problems by casting the tracking task as a cooperative motion planning problem. Because of the computational complexity of the problem, a stochastic search method (genetic algorithm) is proposed for finding in real time monotonically improving solutions. An important attribute of the proposed solution approach is its scalability and, consequently, applicability to large-sized problems. For testing the algorithm, it was implemented in a high-fidelity simulation test bed using a visual database of an actual city. The viability of using the algorithm is shown using a Monte Carlo study. It is envisioned that automating this part of a ground moving target tracking problem will considerably reduce operators' workload and dramatically improve mission performance in such real-life problems.**

## I. Introduction

THE use of unmanned aerial vehicles (UAVs) has increased dramatically in the last two decades, both in military and civilian applications. Civilian applications include geological surveying, fire monitoring, and search and rescue missions. Military applications include reconnaissance and tracking, providing a communication relay, and conducting search and destroy missions. Their extensive use was mainly promoted to reduce the risk to human life, lower cost, and increase operational capabilities.

In early years, UAVs were totally controlled by human operators from the ground. As technology progressed and mission requirement increased, some of the basic tasks, such as flying and trajectory planning from waypoint to waypoint, were automatized and allocated to the UAV, reducing the operators' workload, thus enabling their human operators to focus on other tasks such as target tracking and identification. In the future, UAVs are expected to have a high level of autonomy and operate in groups [1], resulting in requirements for further reduction in operator involvement and transforming operators into supervisors. These requirements sprang an intensive research effort in recent years to develop decision and control methods for various missions involving one or more UAVs.

Several UAV applications require target tracking. Tracking becomes an especially challenging problem when the target is moving in an urban environment and the onboard UAV's sensor line of sight (LOS) to the target might be occluded by the presence of buildings. In this scenario, the operator has to track the target using the UAV's sensor and plan a trajectory that can be followed by the UAV and not result in target occlusion or a violation of airspace limitations. The operator's workload preforming the described mission would, therefore, be extremely high, even for a single UAV, and would amount to the mere impossible for a multi-UAV scenario.

Received 26 July 2007; revision received 1 February 2008; accepted for publication 2 February 2008. 

*Graduate Student, Department of Aerospace Engineering, Technion; vitalysh@tx.technion.ac.il.

†Senior Lecturer, Department of Aerospace Engineering, Technion; tal.shima@technion.ac.il. Senior Member AIAA.

In [2], a two-dimensional scenario of a single observer with omnidirectional dynamic constraints tracking a target with occlusions was addressed. The general problem was divided into two categories on the basis of whether the target is predictable or not. For the predictable case, the acceptable observer locations were marked using a computed visibility polygon, and the dynamic programming approach was used to solve the optimization. The main drawback of the dynamic programming approach is its computational complexity. For the partially predictable target scenario, two online algorithms were presented, one that maximizes the probability that the target will remain in view in the next step and the other maximizes the minimum time in which the target could escape the visibility region in the next time step. The main disadvantage of both approaches was that they only considered a one-step horizon and assumed a very simple target model. The maximization of the minimum escape time in a two-dimensional scenario was further expanded in [3,4].

In scenarios in which target tracking cannot be maintained/ensured by a single UAV, two or more UAVs might be required for the task. Multiple-UAV cooperation can be categorized into centralized and distributed approaches. In the centralized approach, a central control agent (one of the UAVs or a control center) allocates the missions to each UAV. For example, in the tracking problem of a single target, a centralized approach would be to plan each of the UAVs' trajectories by the central control agent or at least divide the time frame in which each UAV is responsible for tracking the target, all based on the available global information. In the distributed approach, each UAV chooses its own mission based only on its local information or that communicated by neighboring UAVs, but not based on the global information available from all UAVs.

An example of a centralized approach was recently proposed in [5] for the combat wide area search and destroy mission in which the task assignment and motion-planning problem for classification, attack, and verification of multiple targets by multiple UAVs is addressed. Suboptimal flight paths were generated by solving a combinatorial optimization problem with a genetic algorithm (GA) using a piecewise optimal Dubins [6] car model to generate the paths and evaluate the cost of each assignment. This approach ensured both piecewise optimal and dynamically possible UAV trajectories. The same problem was solved in [7] using a tree search algorithm. In another application [8], a centralized optimization solution was proposed for assigning micro UAVs to visit multiple targets in an urban terrain. Alternatively, a distributed approach was employed in

[9] to achieve cooperative timing among teams of UAVs flying from different initial points to the same endpoint in a threatened area. Each approach, centralized and distributed, has its respective pros and cons. The centralized approach usually generates better solutions due to the use of global information, whereas distributed algorithms are usually more computationally parsimonious, scalable, and robust.

The main objective of this paper is to present a comprehensive methodology enabling one to automate the cooperative tracking mission of a ground target in an urban environment with airspace limitations, thus reducing the associated operators' workload and improving overall mission performance. For this task, we propose a centralized GA-based motion-planning algorithm that accounts for the UAVs' dynamic constraints. It enables the tracking to be performed with multiple UAVs, carrying gimballed or body-fixed sensors, having preferably different look angles at the target. The remainder of this paper is organized as follows: In the next section, some background on motion planning and evolutionary algorithms is given. Then, the solution methodology is described followed by the mathematical formulation. In Sec. V, a GA multi-UAV tracking planner (GAMUTP) is derived. A performance analysis is presented in Sec. VI, followed by concluding remarks.

## II. Background

In this paper, an evolutionary algorithm is derived to solve the real-time motion-planning problem for a group of UAVs. In this section, first some background on motion planning, mainly for ground robots, is provided. Then, we address the use of evolutionary algorithms for this task.

### A. Motion Planning

Basic path planning has its roots in the field of robotics and attempts to answer the question of how to control a robot from an initial state to a final one in an obstacle-cluttered environment without coming in contact with any of the obstacles. Planning a path for a UAV requires accounting for its dynamic constraints, like minimal turn radius and minimal speed, which is often referred to as motion planning [10]. Probably the best-formulated discipline for solving the motion-planning problem is optimal control [11]. In optimal control, the dynamic constraints are directly implemented into the formulation, and, therefore, the UAV's dynamics can be fully accounted for. In addition, the trajectory is optimized for some required criterion. Optimal control tools include Pontryagin's minimum principle [12] and Bellman's dynamic programming approach [13]. The main drawback of these approaches is that an analytical solution is only possible for very simple problems with a very small number, if any, of obstacles. The dynamic programming approach can numerically solve almost any obstacle configuration and is often used in motion-planning problems [14] but tends to be computationally intractable for real time and high dimensional applications due to the curse of dimensionality.

To reduce the computational burden, some random sampling based methods were proposed. The rapidly exploring random trees [15] method is such an algorithm, that online builds a tree of feasible trajectories by extending branches toward randomly generated target points. The algorithm is initiated at the start point, and a target point is randomly selected at each iteration. The closest (Euclidian) vertex in the tree is extended toward the target point using a constant input for a constant time period. An extension to this approach was proposed in [11] by assuming that an obstacle-free guidance loop is available, which enables one to steer the system from any initial state to any desired one at rest. The distance metric in the extended approach was the collision free cost to go.

When full information on the obstacle location is not available, local path planners are frequently used. Such planners usually lose path optimality but enable planning a path based on partial information about the environment, supplemented by some sensory data collected online as the robot moves toward the goal. Different approaches have been proposed for local planner formulation. A potential field approach was proposed in [16], whereas a two-stage decision mode and trace mode was advocated in [17]. Conversely, in this paper, we will assume full information in the calculated horizon, resulting in a global path planner for the considered horizon.

### B. Evolutionary Algorithms

Another stochastic approach, which is inspired by the mutation-selection process witnessed in nature, are evolutionary algorithms (EAs). EAs are well-known optimization tools used in many disciplines [18], enabling efficient search for feasible solutions. These methods involve iteratively manipulating a population called chromosomes, which encode candidate solutions, to obtain a population that includes better solutions. The manipulation is performed by applying evolutionary operators like selection, crossover, and mutation. Candidate solution selection is performed by evaluating the fitness of each chromosome in the population. Historically, there were three main types of EAs: GA, evolutionary strategies, and evolutionary programming, with GA being the most popular one. In recent years, the difference among the three blurred as each method borrowed ideas from the others [18].

EAs have been previously advocated for motion-planning problems. In [19], an adaptive evolutionary planner/navigator for mobile robots was proposed. The planner used trajectory nodes coding, and unified off-line planning and online planning/navigation in the same evolutionary algorithm. The main drawback of this approach is that the proposed coding results in trajectories composed of straight lines and does not take into account the dynamic constrains of the system. Because of their minimal turn radius and minimum speed constraints, UAVs can not adequately follow such a trajectory.

In [20], routing of autonomous underwater vehicles (AUVs) through evolutionary programming was considered. The authors used an instruction list of changes in the AUV's speed and heading coding to plan a trajectory that passes through required points at specified times, while avoiding fixed threats. A similar coding was employed in [21] for the motion-planning problem with obstacles, by adopting a variable length command list. This type of coding can directly account for some of the UAV's dynamic constraints by limiting the maximal heading and speed change in every time step.

## III. Solution Methodology

We seek a guidance type solution that will enable the UAVs to track, from gimballed or body-fixed sensors, the ground target while accounting for the UAVs' autopilot performance and kinematic constraints. In planning the path for each UAV, the urban occlusions must be taken into account as well as possible obstacles, sensor performance, masking, and other flight restrictions. The planning should be performed cooperatively by the group in real time.

The first step in the solution process is to formulate the autopilot dynamics and problem kinematics and to make simplifying assumptions such as point mass representation for the target. We also assume that perfect information is available regarding all of the UAVs' states.

The next stage in the solution will be to find the visibility regions from which an occlusion-free LOS to the target exists during the scenario. These regions will be calculated based on all available information on the urban terrain and the target future location in the chosen time horizon. The information can be perfect for a totally predictable target trajectory and a full database on the urban environment, or imperfect, like predicted target trajectory and visibility regions that are based on partial information. The prediction of the target trajectory can be obtained in a receding horizon manner, based on statistical methods or operator inputs. Visibility regions can also be provided by the operator, based on his understanding of the scenario, terrain information like maps or photos of the area, and some type of data management system that stores and fuses all available information.

We will then calculate the gimballed or body-fixed sensors' coverage regions from which the UAVs can sense the target and which account for the sensors' effective range, footprint, and masking. This will be followed by the definition of restricted regions, in which the UAVs cannot fly due to the presence of buildings or

other airspace limitations. Collision avoidance between the UAVs in the group must also be taken care of.

Using the previously described method, we will cast the cooperative tracking task as a centralized optimization motion planning problem. The cost function will be associated with the location of the UAVs relative to the visibility and restricted regions and the location of the target relative to the sensors' coverage regions. The cost function will then be minimized under the constraints of the UAVs' autopilot dynamics and problem kinematics. The final step will be to solve this computationally intensive optimization problem using some stochastic approach, allowing real-time implementation. In this study, the chosen approach, presented in Sec. V, is GA.

## IV. Mathematical Formulation

The mathematical formulation of the problem in which multiple UAVs track a ground moving target in an urban terrain is presented in this section. First, the UAVs' dynamics and kinematics are discussed. This is followed by a representation of the urban environment occlusions, sensor coverage regions, and restricted regions. Finally, the cost function for the problem is presented.

### A. Nonlinear Unmanned Aerial Vehicle Kinematics and Dynamics

Let

$$U = \{1, 2, \ldots, N_u\} \tag{1}$$

be the set of UAVs performing the cooperative tracking. In Fig. 1, a schematic planar view of the tracking geometry is shown, where $X^I$, $Y^I$, $Z^I$ is a Cartesian inertial reference frame with the $Z^I$ axis pointing outward. We denote the UAVs and target by the subscripts $i \in U$ and $T$, respectively. $X_i^B$, $Y_i^B$, $Z_i^B$ is a body-fixed Cartesian reference frame attached to the $i$th UAV with the origin at the UAV's center of gravity (cg), $Y_i^B$ axis pointing forward, $X_i^B$ axis passing through the right wing and the $Z_i^B$ axis pointing upward according to the right-hand rule. The speed, normal acceleration, and heading of the UAVs are denoted by $V_i$, $a_i$, and $\psi_i$, respectively. Assuming that the UAVs are flying at a constant altitude and speed, and neglecting wind effects, the UAVs' kinematics relative to the inertial reference frame, are

$$\dot{x}_i = V_i \sin(\psi_i) \tag{2a}$$

$$\dot{y}_i = V_i \cos(\psi_i) \tag{2b}$$

$$\dot{z}_i = 0 \tag{2c}$$

$$\dot{\psi}_i = a_i / V_i \tag{2d}$$

In addition, we assume that the UAVs are equipped with an autopilot system for which the lateral command $u_{\dot{\psi}i}; i \in U$ is the only required input, and it is bounded $u_{\dot{\psi}i} \in [-u_{\dot{\psi}i}^{\max}, u_{\dot{\psi}i}^{\max}]$. Thus, the UAVs' dynamics can be represented by the following arbitrary order nonlinear equation set:

$$\dot{\mathbf{x}}_{Di} = f(\mathbf{x}_{Di}, u_{\dot{\psi}i}, t) \tag{3}$$

where

$$a_i = g(\mathbf{x}_{Di}, u_{\dot{\psi}i}, t) \tag{4}$$

and $\mathbf{x}_{Di}$ is the vector of UAV $i \in U$ internal state variables with $\dim(\mathbf{x}_{Di}) = n$.

We define the state vector of UAV $i \in U$ as

$$\mathbf{x}_i = [\, x_i \quad y_i \quad z_i \quad \psi_i \quad \mathbf{x}_{Di} \,]^T \tag{5}$$

and

$$\bar{\mathbf{x}}_i = [\, \mathbf{x}_i^T \quad \mathbf{x}_T^T \,]^T \tag{6}$$

is the state vector associated with a single UAV and the target, where $\mathbf{x}_T = [x_T, y_T, z_T]^T$ is the target's position relative to the inertial reference frame.

The state vector of the entire problem, which includes the states of all the UAVs and that of the target, is, therefore

$$\mathbf{x} = [\, \mathbf{x}_1^T \quad \mathbf{x}_2^T \quad \cdots \quad \mathbf{x}_{N_u}^T \quad \mathbf{x}_T^T \,]^T \tag{7}$$

### B. Urban Environment Occlusions and Restricted Regions

We will now mathematically define the concepts of occlusion, visibility region, visibility polygon, sensor coverage region, and restricted region.

#### 1. Visibility Region

Let

$$O = \{1, 2, \ldots, N_o\} \tag{8}$$

be the set of buildings or other objects representing the urban environment. Each element $o \in O$ is represented by a polyhedron's body $B^o$, therefore

$$B = \bigcup_{o=1}^{N_o} B^o \tag{9}$$

represents all the objects in the modeled urban environment. Let

$$\mathbf{X}_i = \{(x_i, y_i, z_i) \in \mathbb{R}^3 | z_i \geq 0\} \tag{10}$$

be the space of all possible position states of the $i$th UAV relative to the inertial Cartesian reference frame presented in Fig. 1. Therefore, for every possible target location $\mathbf{x}_T$ and a given urban environment $B$, the target visibility region $\mathbf{X}_i^V(\mathbf{x}_T) \subset \mathbf{X}_i$ is the subspace for which a LOS exists to the target location $\mathbf{x}_T$

$$\mathbf{X}_i^V(\mathbf{x}_T) = \{(x_i, y_i, z_i) \in \mathbf{X}_i | \tilde{\mathbf{X}}_{i,T} \cap B = \{\phi\} \ \forall\, \tilde{x} \in [x_{iT}^-, x_{iT}^+]\} \tag{11}$$

where

$$\begin{aligned} \tilde{\mathbf{X}}_{i,T} &= \{(\tilde{x}, \tilde{y}, \tilde{z}) \in \mathbb{R}^3 | \alpha\tilde{x} + \beta\tilde{y} + \gamma\tilde{z} + \delta \\ &= \alpha x_T + \beta y_T + \gamma z_T + \delta = \alpha x_i + \beta y_i + \gamma z_i + \delta = 0\} \end{aligned} \tag{12}$$

are the coordinates of a straight line passing through the target location $\mathbf{x}_T$ and the $i$th UAV's position, and

$$x_{iT}^- = \min(x_i, x_T); \qquad x_{iT}^+ = \max(x_i, x_T) \tag{13}$$

Note that $\mathbf{X}_i^V(\mathbf{x}_T)$ is a function of only the target position and the modeled urban environment $B$. It is, therefore, identical for all UAVs and is time dependent for a moving target.

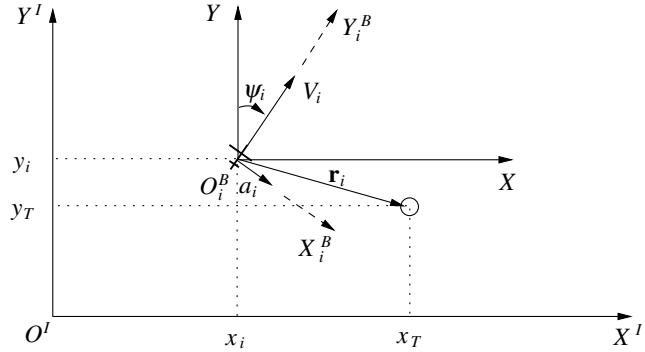


**Fig. 1 Planar view of the tracking geometry.**

In the previous section, we have assumed that the UAVs are flying at constant altitudes. We denote the altitude of the $i$th UAV as $h_i$. Let

$$\mathbf{X}_i^{h_i} = \{(x_i, y_i, z_i) \in \mathbf{X}_i | z_i = h_i\} \tag{14}$$

be the plane defined using this altitude. Therefore, for every possible target location $\mathbf{x}_T$ and a given urban environment $B$, $\mathbf{X}_i^{h_i V}(\mathbf{x}_T) \subset \mathbf{X}_i^{h_i}$ is the subspace for which a LOS exists to the target location $\mathbf{x}_T$ at the $i$th UAV's altitude $h_i$.

$$\mathbf{X}_i^{h_i V}(\mathbf{x}_T) = \mathbf{X}_i^{h_i} \cap \mathbf{X}_i^{V}(\mathbf{x}_T) \tag{15}$$

We denote $\mathbf{X}_i^{h_i V}(\mathbf{x}_T)$ as the target visibility region at altitude $h_i$. The $i$th UAV is within the target visibility region at a given time $t$ if $\mathbf{D}\mathbf{x}_i(t) \in \mathbf{X}_i^{h_i V}(\mathbf{x}_T)$, where $\mathbf{D}$ is a constant matrix

$$\mathbf{D} = [\,\mathbf{I}_{3\times 3} \quad [0]_{3\times(1+n)}\,] \tag{16}$$

with $\mathbf{I}$ representing the identity matrix and [0] a matrix of zeros.

Figure 2 presents a schematic example of the visibility region of a single target at altitude $h_i$. In this figure, UAV2 is in the visibility region; therefore, $\mathbf{D}\mathbf{x}_2(t) \in \mathbf{X}_2^{h_i V}(\mathbf{x}_T)$. UAV1 is outside the visibility region; therefore, $\mathbf{D}\mathbf{x}_1(t) \notin \mathbf{X}_1^{h_i V}(\mathbf{x}_T)$. For simplification, in the rest of the derivation we will not carry the notation for the dependency of $\mathbf{X}_i^{h_i V}$ on $\mathbf{x}_T$ (and consequently possibly on time as well).

For given target location, urban terrain occlusion map, and UAV/sensor altitude, $\mathbf{X}_i^{h_i V}$ can be calculated using a sweep algorithm. Using such an algorithm will result in a visibility polygon consisting of $N_p$ nodes for every calculated time $t$, where $N_p$ is the number of discrete angles sampled in the sweep algorithm. To allow real-time implementation, we compute the visibility polygons at discrete times with intervals of $\Delta T$. Note that not restricting each UAV to a constant altitude, as done in this paper, results with visibility polyhedrons, which are 3-D visibility polytopes.

*Remark*: Instead of calculating visibility polygons, a different approach to examine if a UAV is within the visibility region can be to verify an occlusion free LOS exists for every candidate UAV position. Such an approach has a number of disadvantages: 1) It will require more calculations if the number of candidate UAV positions checked is higher than the number of polygon nodes. In our investigated problem, the number of locations is approximately 2 orders of magnitude larger than the number of nodes; therefore, this approach will be substantially less time efficient. 2) The visibility polygons can be precalculated, whereas this approach cannot. 3) The visibility polygon is calculated once for every target location and does not depend on the number of UAVs in the team.

### 2. *Sensor Coverage Region*

The target visibility region accounts only for urban environment occlusions. To enable target tracking, we would also need to verify that the target is within the sensor's effective range, footprint, and that it is not masked by the UAV. These questions will be addressed next.


**Fig. 2 Visibility region schematic example.**

The relative position $\mathbf{r}_i$, between the $i$th UAV and the target, represented in the inertial reference frame of Fig. 1, is

$$\mathbf{r}_i = \mathbf{x}_T - \mathbf{D}\mathbf{x}_i \tag{17}$$

It can be transformed to the body frame by

$$\mathbf{r}_i^B = \mathbf{D}_{Bi}^I \mathbf{r}_i \tag{18}$$

where $\mathbf{D}_{Bi}^I$ is the direction cosine matrix (DCM) from the inertial reference frame to the $i$th UAV's body frame. Note that $\mathbf{x}_i$ and $\mathbf{x}_T$ are time dependent; therefore, $\mathbf{r}_i$ and $\mathbf{r}_i^B$ are also time dependent. Let us define the $i$th UAV's body-fixed spherical sensor coordinate system $(\rho, \varphi, \theta)_i$ with its origin located at the sensor's position, which is aligned with the UAV's body frame. Assuming that the distance between the UAV's cg location and the sensor's location is negligible relative to the range to the target $\varphi_{iT}$, the target's elevation is the angle between the LOS and the positive $Z_i^B$ axis; $\theta_{iT}$, the target's azimuth, is the angle between the projection of the LOS on the $X_i^B Y_i^B$ plane and the $Y_i^B$ axis; and $\rho_{iT}$ is the range between the UAV and the target (see Fig. 3).

Then, $\mathbf{r}_i^S$, which is the sensor spherical coordinates representation of $\mathbf{r}_i^B$, can be obtained by

$$\rho_{iT} = \|\mathbf{r}_i^B\|_2 \tag{19a}$$

$$\varphi_{iT} = \arccos\left(\frac{r_{iz}^B}{\|\mathbf{r}_i^B\|_2}\right) \tag{19b}$$

$$\theta_{iT} = \arctan\left(\frac{r_{ix}^B}{r_{iy}^B}\right) \tag{19c}$$

where

$$\mathbf{r}_i^B = [r_{ix}^B, r_{iy}^B, r_{iz}^B]^T$$

and

$$\mathbf{r}_i^S = [\rho_{iT}, \varphi_{iT}, \theta_{iT}]^T$$

Note that if the assumption regarding the distance between the UAV's cg and the sensor's location does not hold, Eq. (19) can be adjusted using the level arm between the two positions.

Let us define $\mathbf{X}_i^T$ to be the space of all possible target positions relative to the body-fixed reference frame of the $i$th UAV presented in the spherical sensor coordinate system

$$\mathbf{X}_i^T = \{(\rho_{iT}, \varphi_{iT}, \theta_{iT}) \in \mathbb{R}^3 | 0 \le \rho_{iT} < \infty, 0 \le \varphi_{iT} < \pi, 0 \le \theta_{iT} < 2\pi\} \tag{20}$$

Then, the sensor coverage region $\mathbf{X}_i^{\text{sensor}} \subset \mathbf{X}_i^T$ is a subspace from which the target could be tracked by the $i$th UAV's sensor if the urban terrain occlusions were not present. Therefore, the target will be considered tracked by the $i$th UAV at a given time $t$ if the UAV's

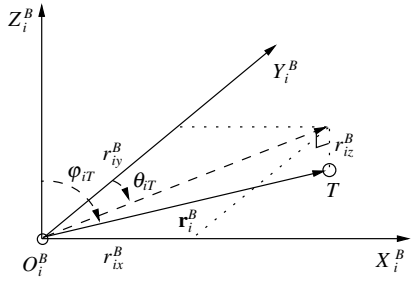


**Fig. 3 Spherical sensor coordinates.**

position states are within the target visibility region $\mathbf{Dx}_i \in \mathbf{X}_i^{h_i V}(\mathbf{x}_T)$ and the target is within the sensor's coverage region $\mathbf{r}_i^S \in \mathbf{X}_i^{\text{sensor}}$.

UAV sensors can be roughly divided into two categories: gimballed and body fixed. Gimballed sensors are usually more complex and enable pointing the sensor to a desired position, with usually minimal effect by the UAV's state. Body-fixed sensors are usually much simpler and cheaper, but their footprint is determined by the UAV's states like pitch and roll angles. Figure 4 presents a schematic example of the footprints of gimballed and body-fixed sensors. In the figure, UAV1 is carrying a gimballed sensor, which can be moved within a larger possible footprint. Target 1 (T1) is enclosed by this larger possible footprint, but the sensor is currently pointing to a different direction. UAV2 is carrying a body-fixed sensor, and target 3 (T3) is within its footprint, but its tracking will not be assured if the UAV rolls or pitches. Target 2 (T2) is outside the footprints of both UAVs.

If the UAV carries a gimballed sensor, and its dynamics are much faster than the UAV's dynamics, then $\mathbf{X}_i^{\text{sensor}}$ can be represented as a Cartesian product $\mathbf{X}_i^{\text{sensor}} = R_i \times \mathbf{M}_i$, where $R_i$ can receive any value between the minimal and maximal sensor effective range, and $\mathbf{M}_i$ is a subspace representing all the angular positions the sensor's footprint can attain without being masked by the UAV's body.

$$R_i = \{\rho_i \in \mathbb{R}^+ | \rho_i^{\min} \le \rho_i \le \rho_i^{\max}\} \tag{21a}$$

$$\mathbf{M}_i = \{(\varphi_i, \theta_i) \in \mathbb{R}^2 | 0 \le \varphi_i < \pi, 0 \le \theta_i < 2\pi, \ \text{noUAVmask}\} \tag{21b}$$

For given sensor performance, installation location, and UAV geometry, $\mathbf{M}_i$ can be represented by a polygon in $(\varphi_i, \theta_i)$. $\mathbf{M}_i$ can be obtained during the UAV's design stage, based on solid models or after production by measuring the installed UAV. The target is within the sensor's coverage region if $\mathbf{r}_i^S \in \mathbf{X}_i^{\text{sensor}}$. Because $\mathbf{X}_i^{\text{sensor}}$ is a Cartesian product, this requirement can be separated into two necessary conditions:

$$\mathbf{E}_1 \mathbf{r}_i^S \in R_i \tag{22a}$$

$$\mathbf{E}_2 \mathbf{r}_i^S \in \mathbf{M}_i \tag{22b}$$

where $\mathbf{E}_1$ and $\mathbf{E}_2$ are constant matrixes

$$\mathbf{E}_1 = [1 \quad 0 \quad 0], \qquad \mathbf{E}_2 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{23}$$

When the gimballed sensor is positioned underneath the UAV, it has full coverage of the lower hemisphere. In such a case, if the UAV's bank and pitch angles are small, no masking from the UAV's body occurs, resulting in the satisfaction of Eq. (22b). This leaves only the necessary condition presented in Eq. (22a). Note that this is usually the case for most gimballed UAV sensors.

If the UAV carries a body-fixed sensor, the same formulation can be used, where $\mathbf{M}_i$ is a subspace representing all the angular positions $(\varphi_i, \theta_i)$ the sensor's field of view (FOV) covers, not masked by the UAV's body. Unlike gimballed sensors, for body-fixed sensors, the necessary condition of Eq. (22b) usually has to be checked, because the sensor's FOV typically does not cover the entire lower hemisphere, and these UAVs tend to be smaller, fly lower, and consequently tend to generate higher roll and pitch angles.

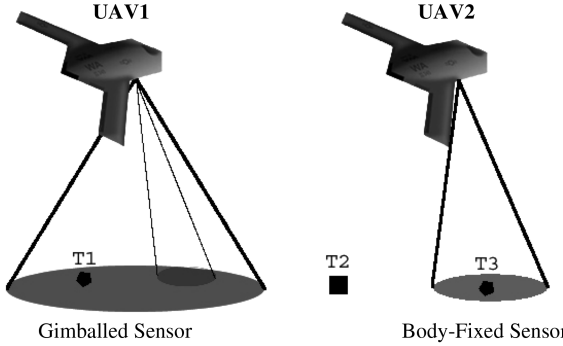


**Fig. 4 Sensor footprint schematic examples.**

*3. Collision Avoidance and Restricted Region*

In the operation of aerial vehicles, altitude deconfliction can be used to avoid collision between the UAVs. Thus, in contrast to planning problems for multiple ground robots, the trajectory of each UAV can be obtained independently. This reduces considerably the coupling between the trajectories of the UAVs in the group. However, different altitudes define different visibility polygons. The derivation accounted for this difference in visibility polygons, but for simplicity we use, in our implementation, the same visibility polygon for all UAVs. This can be justified by assuming that the difference in altitude is small, rendering the difference in visibility polygons negligible. Another possibility is to use for the planning of the entire group the polygon obtained for the UAV flying at the lowest altitude, as it is a subset of all the others $\mathbf{X}_i^{h_i V} \subseteq \mathbf{X}_j^{h_j V} \ \forall\, j \in U | h_i \le h_j$.

We will now define the concept of restricted region. Let

$$L = \{1, 2, \ldots, N_l\} \tag{24}$$

be the set of airspace limitations imposed on the UAVs. Each element $l \in L$ is represented by a polyhedron's body $R^l$; therefore

$$R = \bigcup_{l=1}^{N_l} R^l \tag{25}$$

represents all the airspace limitations imposed on the UAVs. Then, the restricted region of the $i$th UAV, given the urban terrain $B$ with the airspace limitations $R$, is

$$\mathbf{X}_i^R = \mathbf{X}_i \cap (B \cup R) \tag{26}$$

And by confining the $i$th UAV to the altitude $h_i$, we obtain $\mathbf{X}_i^{h_i R}$, which is the UAV's restricted region in the given altitude

$$\mathbf{X}_i^{h_i R} = \mathbf{X}_i^{h_i} \cap \mathbf{X}_i^R \tag{27}$$

accounting for both the urban environment and airspace limitations.

### C. Cost Function

We will now define the cost function for the tracking problem. We will first define the cost function for a single UAV, which we will later expand to the multi-UAV scenario.

As discussed in the preceding subsection, we compute the visibility polygons in intervals of $\Delta T$. Our main objectives are to maintain the $(x_i, y_i)$ position of the UAV within the visibility polygons and maintain the relative position $\mathbf{r}_i^S$ within the sensor coverage region, while avoiding the restricted region, all for the entire calculated trajectory. Therefore, the cost function chosen in this paper is

$$J_i = a \cdot l_f[\bar{\mathbf{x}}_i(t_M)] + \sum_{k=0}^{M-1} \{l_k[\bar{\mathbf{x}}_i(t_k), u_{\dot{\psi}i}(t_k)] + b \cdot l_{Uk}[u_{\dot{\psi}i}(t_k), u_{\dot{\psi}i}(t_{k+1})]\} \tag{28}$$

where $t_M = M\Delta T$ is the calculated horizon, and $a$, $b$ are positive constants. $l_f[\bar{\mathbf{x}}(t_M)]$ is the loss associated with the final state,

$$l_k[\bar{\mathbf{x}}_i(t_k), u_{\dot{\psi}i}(t_k)]$$

is the loss associated with the state $\bar{\mathbf{x}}_i$ and the control command $u_{\dot{\psi}i}$ at time $t_k = k\Delta T$, and

$$l_{Uk}[u_{\dot{\psi}i}(t_k), u_{\dot{\psi}i}(t_{k+1})]$$

is the loss associated with the change in the control command of the $i$th UAV between $t_k$ and $t_{k+1}$.

By choosing $l_f = 0$, $l_{Uk} = 0$, and

$$l_k[\bar{\mathbf{x}}_i(t_k), u_{\dot{\psi}i}(t_k)] = \begin{cases} 0 & \bar{\mathbf{D}}\bar{\mathbf{x}}_i(t_k) \in \mathbf{X}_i^{h_i V}, \mathbf{r}_i^S(t_k) \in \mathbf{X}_i^{\text{sensor}}, \bar{\mathbf{D}}\bar{\mathbf{x}}_i(t_k) \notin \mathbf{X}_i^{h_i R} \\ 1 & \bar{\mathbf{D}}\bar{\mathbf{x}}_i(t_k) \notin \mathbf{X}_i^{h_i V} \quad \text{or} \quad \mathbf{r}_i^S(t_k) \notin \mathbf{X}_i^{\text{sensor}}, \bar{\mathbf{D}}\bar{\mathbf{x}}_i(t_k) \notin \mathbf{X}_i^{h_i R} \\ W_R & \bar{\mathbf{D}}\bar{\mathbf{x}}_i(t_k) \in \mathbf{X}_i^{h_i R} \end{cases} \tag{29}$$

where $\bar{\mathbf{D}}$ is a constant matrix

$$\bar{\mathbf{D}} = [\, \mathbf{I}_{3\times 3} \quad [0]_{3\times(n+4)} \,] \tag{30}$$

and $\mathbf{X}_i^{h_i V}$ is calculated for $t_k$, we impose a cost function that measures the number of time frames for which the target is not tracked by the $i$th UAV, and the number of time frames weighted by $W_R$ for which the $i$th UAV is within the restricted region. By choosing $W_R \to \infty$, we can impose a solution that does not penetrate the restricted regions.

By defining $l_f$ in a similar manner, we can impose a final location on the UAV while avoiding the restricted region. Choosing

$$l_{Uk}[u_{\dot{\psi}i}(t_k), u_{\dot{\psi}i}(t_{k+1})] = |u_{\dot{\psi}i}(t_k) - u_{\dot{\psi}i}(t_{k+1})| \tag{31}$$

can be used to penalize jerky solutions, resulting in smoother UAV trajectories.

Let us now expand the cost function proposed in Eq. (28) to a multi-UAV scenario. In such a setting, we would like the target to be tracked by at least one UAV, whereas tracking from multiple UAVs is preferred to increase the solution robustness and to verify that none of the UAVs diverges even when good tracking performance of the other UAVs is attained. Let

$$T_k[\mathbf{x}(t_k)] \in \{0, 1\}$$

be a binary decision variable that is 0 if at least one UAV is tracking the target at time $t_k = k\Delta T$ and 1 otherwise

$$T_k[\mathbf{x}(t_k)] = \begin{cases} 0 & \exists i | \bar{\mathbf{D}}\bar{\mathbf{x}}_i(t_k) \in \mathbf{X}_i^{h_i V}, \mathbf{r}_i^S(t_k) \in \mathbf{X}_i^{\text{sensor}} \\ 1 & \text{else} \end{cases} \tag{32}$$

We can now expand Eq. (28) in the following manner:

$$J = \sum_{k=0}^{M-1} T_k + cT_M + d\sum_{i=1}^{N_u} J_i \tag{33}$$

This cost has to be minimized by the group under the kinematic constraints of the UAVs given in Eq. (2). Note that the individual cost for every UAV is weighted by a positive number $d$, and $c$ is a positive weight on $T_k$ at the final time $t_k = M\Delta T$. Increasing $d$ will result in solutions in which the individual tracking performance of each UAV will improve on the expense of global tracking performance and vice versa.

## V. Genetic Algorithm Multiple Unmanned Aerial Vehicle Tracking Planner

Multi-UAV tracking in an urban environment was mathematically formulated in the previous section as a very complex optimization problem. The need to maintain the UAVs' position states in the visibility polygons and the relative positions within the sensors' coverage regions while avoiding restricted regions and accounting for the dynamic constraints of the UAVs results with a large number of constraints. This makes any attempt to analytically solve the problem using optimal control tools impractical. Because of the need for real-time implementation, the planning algorithm should have the following desirable attributes: quick feasible solution and monotonically improving solution quality over runtime. A deterministic search algorithm ($A^*$ or Dijkstra, for example) might be suggested for solving the optimization, but such an approach would be computationally intractable due to the large search space (see Appendix); therefore, a stochastic search method should be employed. As presented in Sec. II, EAs are well-known optimization tools and have been shown to be suitable for solving motion-planning problems. In this section, we will present a GA multi-UAV tracking planner, which has the desirable characteristic stated previously. Note that other stochastic optimization methods, like the cross entropy, simulated annealing, and random search methods, may also be considered for solving the optimization problem at hand.

### A. Encoding

An important part of solving such a problem using a GA is to select the appropriate chromosome encoding. Because of the need to account for the dynamic constraints of the UAVs, the candidate solution for each UAV is represented by a string of discrete commands, of the form presented in Fig. 5, in which each command $u_{\dot{\psi}i}^k$ represents a constant input to the UAV's autopilot system for the time interval of $t \in [t_k, t_{k+1})$. Each such chromosome representation of the solution is composed of an ordered set $G_M$ of $M + 1$ genes where

$$G_M = \{g_0, g_1, \ldots, g_M\} \tag{34}$$

By limiting $u_{\dot{\psi}i}^k$ to the closed set

$$u_{\dot{\psi}i}^k \in [-u_{\dot{\psi}i}^{\max}, u_{\dot{\psi}i}^{\max}]$$

we can guarantee that any candidate solution will uphold the dynamic constraints of the UAV as presented in Sec. IV.A. The possible values of $u_{\dot{\psi}i}^k$ can either be any continuous value in the closed set

$$u_{\dot{\psi}i}^k \in [-u_{\dot{\psi}i}^{\max}, u_{\dot{\psi}i}^{\max}]$$

or can be chosen from a subset of those values

$$u_{\dot{\psi}i}^k \in \{(u_{\dot{\psi}i})^1, (u_{\dot{\psi}i})^2, \ldots, (u_{\dot{\psi}i})^q\} \subset [-u_{\dot{\psi}i}^{\max}, u_{\dot{\psi}i}^{\max}]$$

In this paper, we have chosen to use the former option.

Thus far, we have considered the solution encoding of a single UAV. Let us now expand the proposed encoding to a multi-UAV setting. We propose to combine the encoding for all UAVs into one large chromosome and perform the genetic operators on that chromosome, resulting in a single population containing the solution for the entire team. The chromosome has a matrix shape in which every row represents the solution for one UAV, and every column represents the same time frame for all UAVs. Figure 5 schematically presents this structure.

### B. Candidate Solutions Evaluation

The fitness $f$ of each of the solutions, coded in the chromosomes presented in the previous section, will be based on computing the cost function of Eq. (33), where

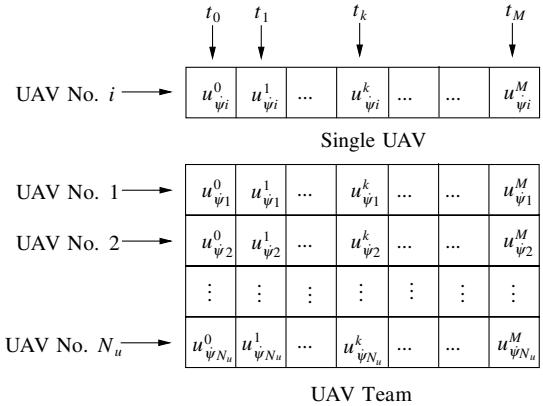


**Fig. 5 Control input encoding.**

$$f = 1/J \tag{35}$$

Calculating the value of Eq. (33) for each chromosome requires the values of each UAV's states $\mathbf{x}_i$ in the calculated time horizon $M\Delta T$, which can be obtained by integrating Eqs. (2–4) using the commands coded in the chromosomes. Such an integration can be performed numerically for any UAV dynamics using numerical ordinary differential equation solvers, like the Runge–Kutta method. Note that al Eqs. (2–4) are continuous, the values of the UAV states $\mathbf{x}_i$ are only needed at discrete times $t_k = k\Delta T$. To simplify the solution, an ideal dynamics Dubins [6] car model type UAV was considered in this paper, for which Eq. (4) degenerates to

$$a_i = g(\mathbf{x}_{Di}, u_{\dot{\psi}i}, t) = u_{\dot{\psi}i} \tag{36}$$

Once $\mathbf{x}_i$ has been calculated, the cost function of Eq. (33) can be evaluated by examining whether, for each time frame, the following conditions hold: 1) $\mathbf{Dx}_i \in \mathbf{X}_i^{h_i V}$; 2) $\mathbf{Dx}_i \in \mathbf{X}_i^{h_i R}$; 3) $\mathbf{E}_2\mathbf{r}_i^S \in \mathbf{M}_i$; 4) $\mathbf{E}_1\mathbf{r}_i^S \in R_i$. Conditions 1–3 require checking whether a point lies within a polygon. The point in a polygon (PIP) is a well-known problem in computational geometry [22]. The problem is to find out whether a given point in the plane lies inside a polygon. Many algorithms for this problem appear in the literature, like the winding number or crossing number algorithms [22]. In the crossing number algorithm, for example, the number of times a ray starting from the point intersects the edges of the polygon is counted. The point is inside the polygon if and only if the number is odd. The algorithm complexity is $\mathcal{O}(N_p)$, where $N_p$ is the number of polygon edges.

*Remarks*: 1) The visibility polygons for an entire scenario can be calculated in $\mathcal{O}(N_p M)$ time, where $M$ is the number of time frames in the calculated horizon. These polygons are calculated once for the target trajectory before the GA is run. One option to reduce the calculation time is to compute visibility polygons off line for some sampled target position resolution and store them in a database. Extracting the polygons from the database can be done in real time based on the expected target location. 2) Unlike the visibility polygon, the restricted region might be composed of a few polygons, each emanating from a different airspace limitation. To improve computation time, a grid-based database should be constructed containing information on which airspace limitation applies for each sector on the grid. Based on the UAV's position on the grid, only the relevant airspace limitations will have to be checked. The restricted region is not a function of the target's position and, therefore, the proposed database can be precalculated.

### C. Reproduction Process

Our algorithm is initialized with $N_s$ randomly generated chromosomes, and we maintain a fixed sized population throughout the process. Off line, reproducing new generations of solutions can be repeated until some stopping criterion is met, for example, the fitness of the best chromosome has not improved more than 5% from the previous iteration. For an online implementation, the algorithm can be run only for a given allotted time. In this study, the population has been progressed for a given number of generations, denoted $N_g$. For creating a new population of solutions, we employ the genetic operators: selection, crossover, mutation, and elitism.

#### 1. *Selection*

In this stage, two parent chromosomes are selected based on their fitness. For the selection we employ the roulette wheel method.

Let

$$C = \{c_1, c_2, \ldots, c_{N_s}\} \tag{37}$$

be the set of chromosomes. Using the mapping of Eq. (33) and (35), let

$$F = \{f_1, f_2, \ldots, f_{N_s}\} \tag{38}$$

be the set of corresponding fitness of the chromosomes. This set does not have to be ordered. We now calculate

$$Fs = \sum_{\chi=1}^{N_s} f_\chi \tag{39}$$

and define the set

$$Fc = \{Fc_1, Fc_2, \ldots, Fc_{N_s}\} \tag{40}$$

where

$$Fc_\xi = \sum_{\chi=1}^{\xi} f_\chi; \qquad \xi = 1, 2, \ldots, N_s \tag{41}$$

Note that $Fc_1 = f_1$ and $Fc_{N_s} = Fs$.

We then generate a uniformly distributed random number $\mu \sim U(0, Fs)$ and select as the first parent the chromosome that satisfies

$$\text{parent} = \underset{c_\xi \in C}{\arg\min}(Fc_\xi - \mu \geq 0) \tag{42}$$

and repeat the process for the second parent.

#### 2. *Crossover*

We apply the crossover operators with a probability denoted as $p_c$. The probability that the offsprings will be exact replicas of their parents is bounded from below by $(1 - p_c)$ because there is a nonzero probability that identical parents will be chosen. In this study, three types of single-point crossover methods have been chosen, and the methods are schematically illustrated in Fig. 6.

In the first method, applied with probability $p_{cf}$, and denoted full row crossover, a column $i$ is selected randomly based on a uniform distribution. When applying this operator, the first child solution consists of the first $g_s = iN_u$ genes ($i$ genes from each row) from the first parent and $(M + 1)N_u - g_s$ genes from the second parent, and vice versa for the second child.

In the second method, applied with probability $p_t$ and denoted turn crossover, a column $i$, a row $j$, and an acceleration command $u_{\dot{\psi}i} \in [-u_{\dot{\psi}i}^{\max}, u_{\dot{\psi}i}^{\max}]$ are selected randomly from uniform distributions. When applying this operator, the genes in all rows except the $j$th row are passed directly from the first parent to the first child and similarly for the second child. The last $g_s = N_u - i$ genes of the first child and the first $g_s = i$ genes of the second child in the $j$th row are replaced with a constant acceleration command $u_{\dot{\psi}i}$; the last $g_s = N_u - i$ genes of the second child are taken from the first parent. This operator manipulates a single UAV in both parents and is a blend of a crossover and a mutation operator.

In the last method, applied with probability $(1 - p_{cf} - p_t)$ and denoted point crossover, both a column $i$ and a row $j$ are selected randomly based on uniform distributions. Applying this operator is equivalent to joining all the UAVs' coding of each parent to a string in which the first child solution consists of the first $g_s = (j - 1)(M + 1) + i$ genes from the first parent and $N_u(M + 1) - g_s$ genes from the second parent strings, and vice versa for the second child.

Importantly, all crossover operators produce feasible solutions, and there is no need of chromosome repairs. Note that the methods have very different physical meanings. The full row crossover operator replaces the time history of all the UAVs in the candidate child solution from the $(i + 1)$th time frame; the turn crossover operator changes the time history of a single UAV and leaves the rest of the UAVs unchanged. The point crossover operator changes the time history of one UAV while possibly taking complete trajectories of other UAVs from other candidate solutions. Another interesting observation regarding the crossover operators is that for the full row and point operators, generally speaking, a crossover at one of the first genes corresponds to a larger perturbation than in one of the last genes because it has a higher probability of affecting the rest of the path and, consequently, the group plan.

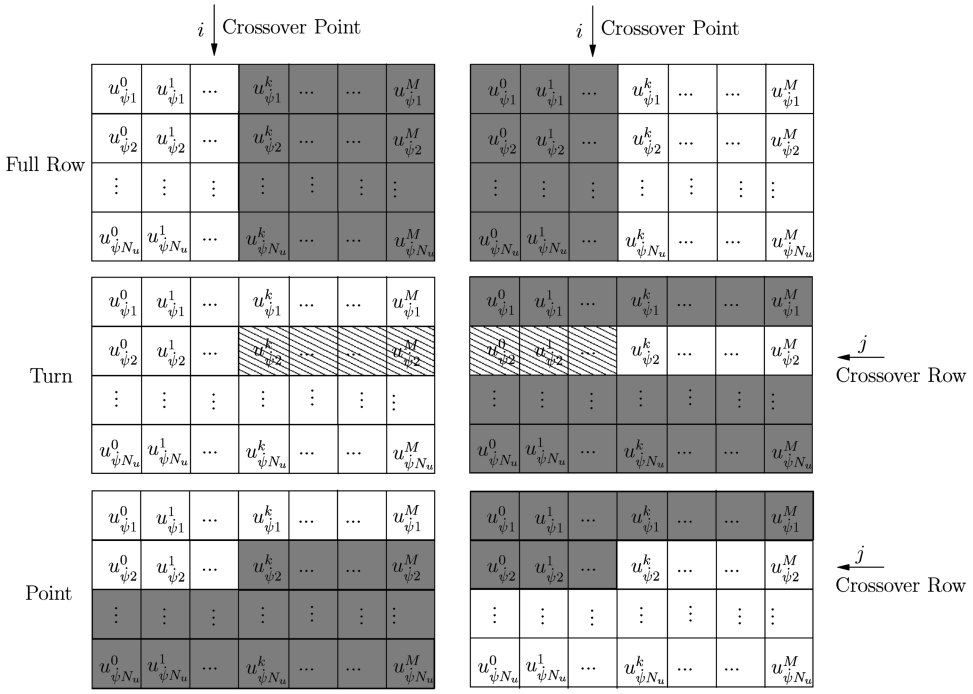


**Fig. 6 Crossover operators.**

*3. Mutation*

The mutation operator is applied with a probability $p_m$ to each gene of the chromosome. We mutate the gene by randomly selecting a new acceleration command value $u^k_{\dot{\psi}i} \in [-u^{\max}_{\dot{\psi}i}, u^{\max}_{\dot{\psi}i}]$. The application of this operator prevents the algorithm from getting trapped in a local minimum of the problem, and from one solution to dominate the entire population. Generally speaking, as for the full row and point crossover operators, mutating the first genes of each UAV has a larger effect on $J$ than mutating the last genes.

*4. Elitism*

We apply this operator to keep the best solutions from each generation of solutions. We keep the best $N_e$ chromosomes from the previous generation, and the rest of the new chromosomes $(N_s - N_e)$ are obtained by repeatedly producing children, by the methods described previously, until the entire population is filled.

*Remarks*: 1) Note that GA solutions have been shown to converge [18] in probability to the optimal solution, when the elitism operator is applied and the mutation rate is $0 < p_m < 1$. 2) One of the time-consuming parts of the algorithm is calculating $\mathbf{x}_i \; \forall \; i \in U$. Therefore, whenever the first section of each row is identical to one of the parents, then there is no need for recomputing $\mathbf{x}_i$ and the associated part of the cost $J_i$ for this section. This can considerably reduce the computation time.

### D. Algorithm Initialization

Initialization is an important part of many optimization and motion-planning algorithms. The problem's cost function was chosen so as to maintain the UAVs within the visibility polygons and the target within the sensors' coverage regions, and not to bring the UAVs to the visibility region and the target to the sensors' coverage regions in minimal time. Therefore, if the tracking scenario is initiated when one or more of the UAVs are far from the visibility polygons, or the target is far from the sensors' coverage regions, a supplementary algorithm should be employed to bring the UAVs to adequate initial conditions. For a stationary or a very slow target and no restricted regions, a Dubins [6] trajectory to the predicted target location can be used. Such a trajectory ensures a minimal length path to a fixed target point, with a terminal heading constraint, for a UAV with ideal dynamics. When the target is not stationary, the Dubins trajectory is no longer optimal, and a guidance type approach should be used. In a recent paper by the authors, an optimal control-based guidance law and a linear quadratic game-based guidance law were proposed [23], which enable intercepting a target with a terminal angle constraint. These guidance laws can be used with the actual target location or the center of the appropriate visibility polygon. Moreover, for maximizing the group performance, different final headings should be enforced on the UAV group, for example

$$\psi_i = i2\pi/N_u; \qquad i \in U \tag{43}$$

## VI. Performance Analysis

The performance of the proposed GAMUTP is investigated in this section via simulation using a high-fidelity graphics software. We will first present the simulation environment, followed by a presentation of the test scenarios. We will then present sample runs and analyze the performance of the proposed planner in each test scenario using a Monte Carlo study.

### A. Simulation Environment

We used the graphics capabilities of a software named Tri-Malat. This C++ application combines sensory images from three sensors (payloads) and includes three video channels and a map/ortho view. By default, all sensor/camera payloads are coupled so that commanding one of them to look at a point or move will result in moving all of them. The application is based on VegaPrime, which is a commercial, off-the-shelf tool available for the creation and deployment of visual simulations. For the terrain, it uses the data base of the city of Tel Aviv, Israel. The computer used was an Intel 2.0 GHZ dual core 2 with 2 GB RAM using an NVIDEA GeForce Go 7600 graphic card.

In this application, situation awareness of the supervisory operator is maintained by tightly connecting the command and control (C2) display and the payload displays. An example of the display is shown in Fig. 7.

In the upper right square of the display (C2), the location of the three UAVs is plotted on top of a map of the terrain. The payloads, directions of gaze, and the ground footprints are also displayed. In each payload display, lines are drawn indicating the LOS from the other payloads to the target. Each payload is identified by a different color (yellow, green, red).

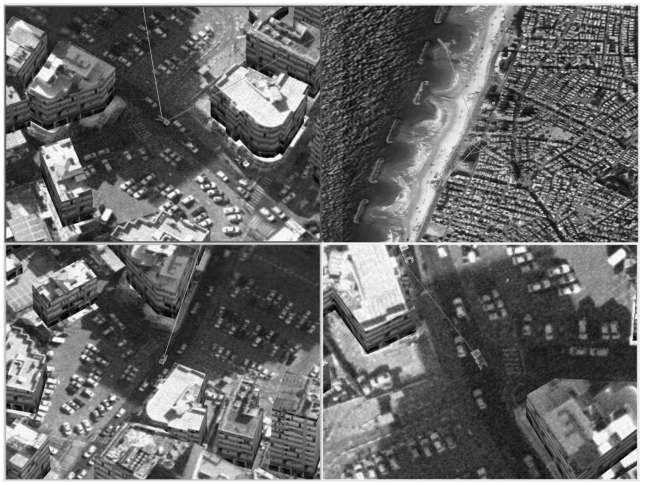

**Fig. 7 Tri-Malat display.**

The GAMUTP was implemented in MATLAB® and compiled to a dll using the MATLAB compiler toolbox. The UAVs' motion planning using the GAMUTP is initiated by the operator in the Tri-Malat environment. Based on the operator inputs, the Tri-Malat software calculates the predicted visibility polygons in the calculated horizon and passes the polygons and UAVs' initial states to the planner. The planner returns the planned trajectory, autopilot commands, tracking strategy, and predicted tracking performance for all the tasked UAVs. The motion plan is then implemented in the Tri-Malat environment, freeing the operator from the path-planning mission.

### B. Test Scenarios

Two 120-s-long scenarios consisting of a target moving at approximately 15 m/s were considered. Figure 8 presents the target trajectory on an orthophoto in which the dots represent 10 s intervals. The target is tracked by multiple homogeneous UAVs flying at a constant speed of $V_i$ and a constant altitude of $h_i$ and carrying gimballed sensors situated underneath the UAV. The sensors' maximal range is 3000 m, and they do not experience any masking. The UAVs' positions are initialized at a distance $R_0$ in each axis from the initial target location. The UAVs are flying above the urban environment; therefore, $h_i > h_B^{\max}\ \forall\ i$, where $h_B^{\max}$ is the highest point in the urban environment.

In the first scenario, no airspace limitations exist, whereas in the second scenario, three square $400 \times 400$ m airspace limitations inhibit the UAVs at their flight altitude with centers situated at

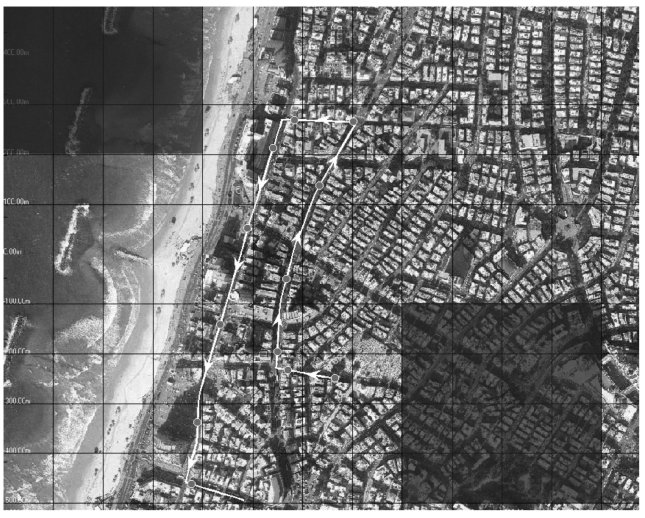

**Fig. 8 A map of the test scenario.**

**Table 1 Simulation parameters**

| GA | Scenario |
|---|---|
| $N_s \in \{10, (20), 40, 80\}$ | $V_i = 30$ m/s |
| $N_e = 4$ | $h_i = 800$ m |
| $p_c = 0.9$ | $R_{\min} \in \{100, (200), 400\}$ m |
| $p_{cf} \in \{(0.7), 1\}$ | $N_u \in \{1, (2), 3, 4\}$ |
| $p_t \in \{0, (0.2)\}$ | $R_o \in [0, 1500]$ m |
| $p_m = 0.01$ | $\Delta T = 1$ s |
| $N_g = 100$ | $N_p = 18$ |

$(-300, -300)$, $(-1100, 400)$ and $(-1100, -800)$, two of which can be seen in Fig. 8 marked as darker areas on the orthophoto.

The visibility polygons are computed at intervals of $\Delta T$, and each polygon consists of $N_p$ nodes. The maximal acceleration command can be derived from $u_{\dot{\psi} i}^{\max} = V_i^2 / R_{\min}$. The weights of Eqs. (28) and (33) were chosen as $a = c = d = 1$, $b = 0$, $W_R = 10$, and $l_f = l_k$ and equals to Eq. (32). The parameters used for the simulation and the GA are summarized in Table 1, in which the values marked with brackets are the default ones.

Figures 9 and 10 present the planned trajectories of two UAVs and a ground moving target in the described scenario without and with airspace limitations, respectively. The circles and squares on the UAVs' trajectories indicate loss of LOS of UAV1 and UAV2, respectively, signifying that the UAVs were outside the visibility polygons at the marked locations. The circles and squares are also marked on the target trajectory, indicating the target's location when the loss on LOS transpired. In the sample run presented in Fig. 9, the first UAV has lost LOS to the target in 7 time frames out of 120, whereas the second UAV maintained LOS throughout the entire scenario. Note that the first UAV has attained good tracking performance notwithstanding being initiated at a diverging direction from the target, corrected by a hard left turn at the beginning of the scenario.

In the sample run presented in Fig. 10, the first UAV has lost LOS to the target in 10 time frames out of 120, mainly due to the initiation in a diverging direction far from the target, whereas the second UAV lost LOS to the target in 5 time frames. The loss of LOS of the two UAVs transpired at different time frames, resulting in perfect team tracking performance while avoiding the restricted regions.

### C. Monte Carlo Simulation

A Monte Carlo study consisting of sets of 200 runs is used in this section to evaluate the performance of the GAMUTP. For every run, the position and heading of the UAVs were selected from a uniform distribution, and only positions that were within the first visibility

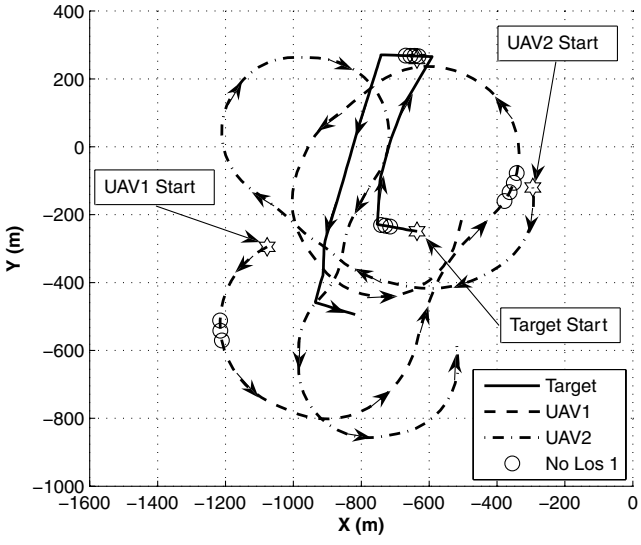


**Fig. 9 Test run of two UAVs tracking a target without airspace limitations.**

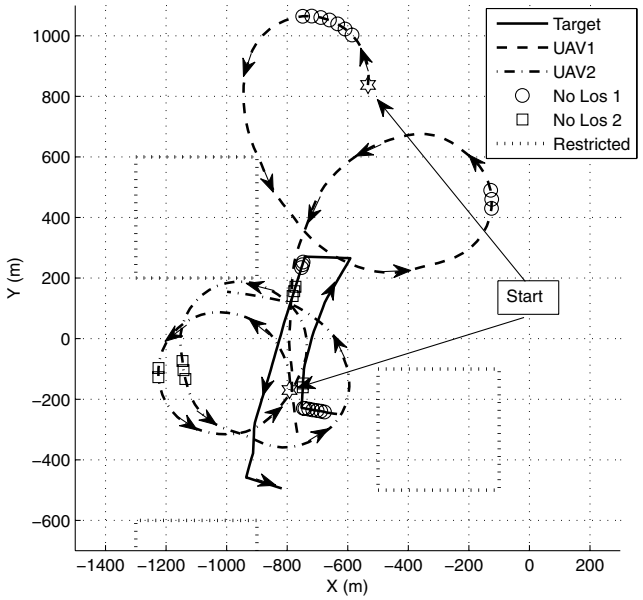


**Fig. 10 Test run of two UAVs tracking a target with airspace limitations.**

polygon and at a distance greater than $R_{\min}$ from the restricted regions, when present, were selected. The algorithm was then run for the test scenarios presented in the preceding section. Most of the results are presented as a function of solutions evaluated to enable comparisons to population and nonpopulation-based methods. Each figure in the study presents the ensemble averages of the total cost [Eq. (33)] and the team's tracking cost [Eq. (44)].

$$[\text{team tracking }(\%)] = \frac{100}{M+1}\sum_{k=0}^{M} T_k \tag{44}$$

The simulation results for the scenario without airspace limitations will be presented first followed by the results for the scenario with airspace limitations.

### 1. *Scenario Without Airspace Limitations*

Figure 11 compares between the GAMUTP with two possible values of $p_t$ and a random search algorithm using a similar encoding. The random search algorithm generated acceleration commands drawn from a uniform distribution for each candidate solution and maintained the best solution. The random search algorithm performed poorly compared with the GAMUTP with an expected cost of $E(J) = 146$ after 2000 evaluated solutions, versus $E(J) = 32$ for the GAMUTP, with $p_t = 0.2$. It is also evident that $p_t = 0$ yields substantially higher total and team tracking costs when compared with $p_t = 0.2$ indicating that the turn crossover operator has a vital effect on the planner's performance. The expected value of the team's tracking cost is approximately 2.5% for these parameters.

The expected value of the cost converged to within 5% of its final value after approximately 850 evaluated solutions, which is approximately 43 iterations, for a 20 chromosome population. And the expected value of the team's tracking cost converged to within 5% of its final value after approximately 450 evaluated solutions.

Figure 12 presents the GAMUTP performance for a multi-UAV scenario. As expected, the total cost increases with the number of UAVs, due to the summation of all individual UAV costs. On the other hand, the expected value of the team's tracking cost decreases dramatically to approximately 1% for a three-UAV tracking team. It is evident that the benefit of adding a fourth UAV to the team, for the tracking performance alone, is negligible, al other benefits like robustness can still be gained. Note that al the Tri-Malat simulation can only currently accommodate three UAVs, a four-UAV scenario was run in this Monte Carlo simulation in the planner alone, which is not limited by this restriction.

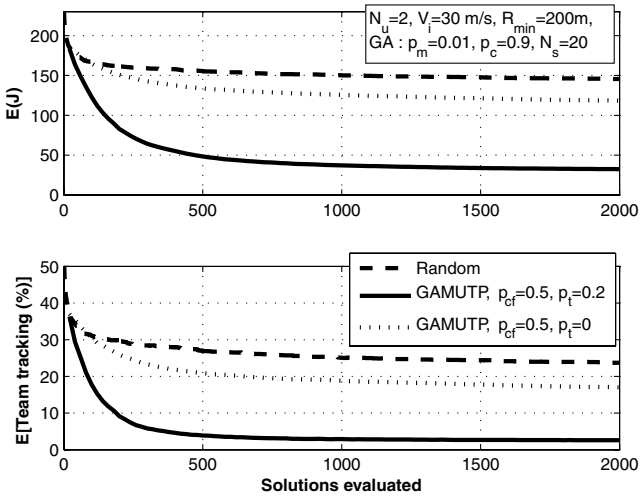


**Fig. 11 GAMUTP and random performance in the scenario without airspace limitations.**

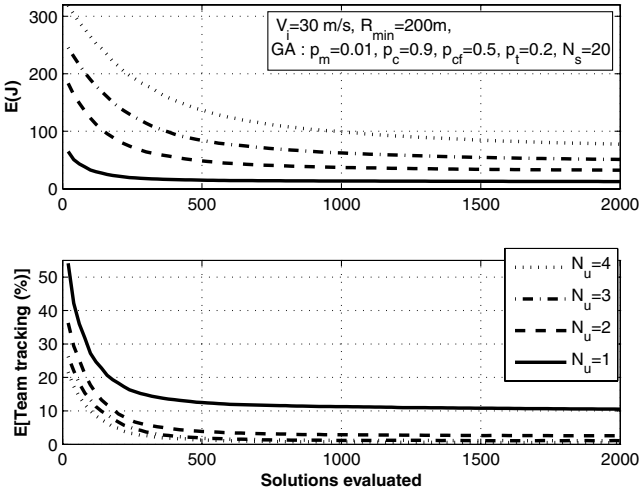


**Fig. 12 Performance for multiple UAVs in the airspace limitations free test scenario.**

A major concern in a multi-UAV scenario is system stability. In all Monte Carlo runs performed, none of the UAVs in the tracking teams diverged in any of the test runs. The stability of the UAV team is attained mainly due to the selected cost function, which is a summation of the individual UAV costs and the total team tracking performance. Therefore, solutions in which one or more of the team members diverge will result in high costs and, consequently, low fitness, reducing their probability of being selected in the selection process. The turn crossover operator also improves system stability by allowing a divergent UAV to converge back to the target by adding a turn component to its control input string.

Table 2 presents the number of evaluated solutions and computation time for the expected value of the cost to converge to within 5% of its final value. The algorithm maintains similar convergence properties for a multi UAV scenario with a moderate increase in the number of needed evaluations, due to the increased search space. This scalability property is very important for real-life implementation of such an algorithm. Note that the computational time is linearly dependent on the number of UAVs ($N_u$) and the number of evaluated solutions. The computation times presented in Table 2 were attained in a MATLAB implementation of the algorithm. Al these computation times may constitute real-time performance even when implemented in MATLAB (because the scenario duration is 120 s), it is expected that a C/C++ implementation will further reduce the computational time by at least an order of magnitude.

**Table 2 Computation time for multiple UAVs in the test scenario with no airspace limitations**

| UAVs | Evaluations | Time (s) |
|---|---|---|
| 1 | 500 | 0.97 |
| 2 | 850 | 2.85 |
| 3 | 1050 | 4.95 |
| 4 | 1300 | 8.15 |

Figure 13 presents the algorithm's sensitivity to the population size $N_s$ for a two-UAV tracking team as a function of generations evaluated. For the same number of generations, the expected value cost decreases with the population size as expected, as it covers more of the search space. The difference between a population of $N_s = 40$ and $N_s = 80$ is very small, and it is, therefore, not recommended to use a population size of more than $N_s = 40$ in this case, due to the low gain in performance compared with the increase in computation time. For the test scenario at hand, a population size of $N_s = 40$ outperforms $N_s = 20$ when the number of evaluated solutions exceeds 1600; therefore, due to computation time requirements, a population size of $N_s = 20$ was used in most simulations in this scenario. Note that this population size is somewhat small relative to the control input vector. The main reasons that such a small population size is sufficient in this case are 1) the UAVs' visibility constraints are "soft," meaning that the solution is still feasible even when the UAV is outside the visibility polygons in some of the time frames; 2) the cost function for the problem has only an integral cost, unlike many other motion-planning problems in which some goal has to be reached (not requiring the UAV to reach a final location enables a smaller population size and faster convergence); 3) the crossover operators used in the solution have been found to be very efficient, especially the turn crossover operator, which adds turn segments to the population, preventing population stagnation, and, when combined with the other crossover operators, enables fast exploration of the search space. Note that when "hard" constraints like restricted regions exist, a larger population size is usually needed (see VI.C.2).

*2. Scenario with Airspace Limitations*

Figure 14 compares the GAMUTP with two possible values of $p_t$ and the random search algorithm. The same trends seen in the scenario without airspace limitations appear in the analyzed scenario. The GAMUTP outperformed the random algorithm, and applying the turn crossover operator with probability $p_t = 0.2$ yielded substantially better performance than for $p_t = 0$. The expected values of the cost and the team's tracking cost converge to approximately $E(J) = 45$ and 4.5%, respectively, which is as expected slightly degraded performance when compared with the scenario without airspace limitations.

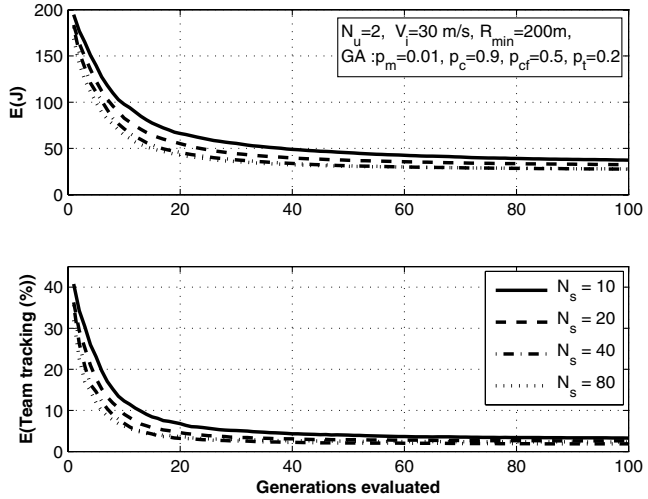


**Fig. 13 Sensitivity to population size in the scenario without airspace limitations.**

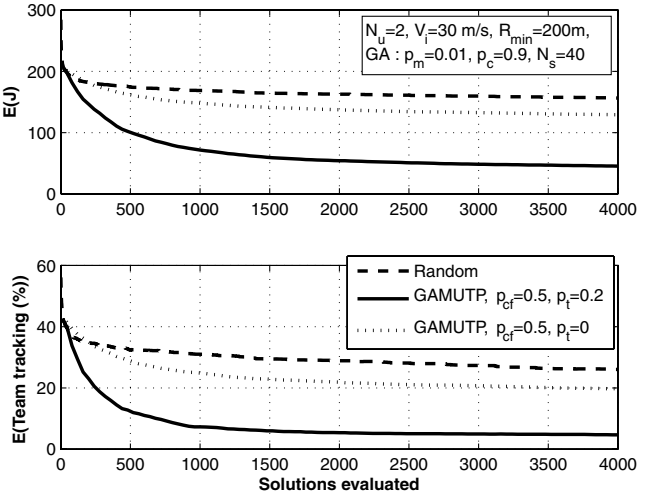


**Fig. 14 GAMUTP and random performance with airspace limitation.**

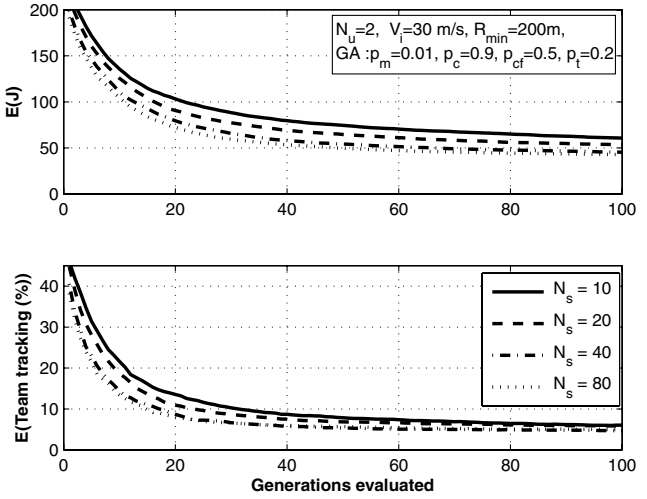


**Fig. 15 Sensitivity to population size with airspace limitation.**

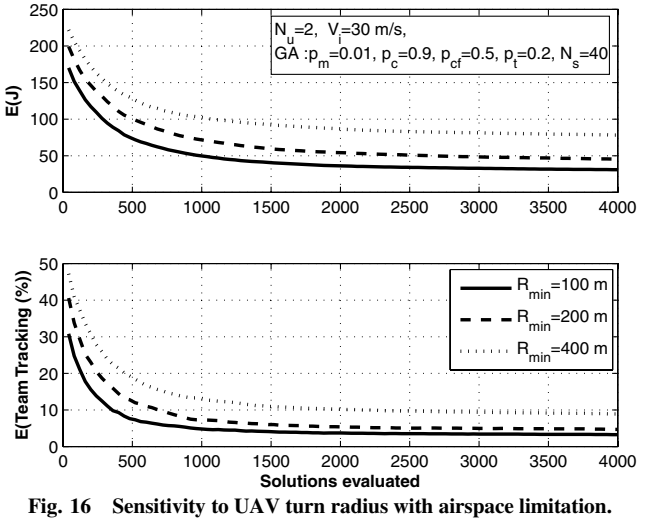


**Fig. 16 Sensitivity to UAV turn radius with airspace limitation.**

The expected value of the cost converged to within 5% of its final value after approximately 2100 evaluated solutions, which is approximately 53 iterations, for a 40 chromosome population. And the expected value of the team's tracking cost converged to within 5% of its final value after approximately 1200 evaluated solutions. The increase in the number of evaluated solutions needed is also expected due to the increased scenario complexity.

Figure 15 presents the algorithm's sensitivity to the population size for a two-UAV tracking team. The effect of increasing the population size on the expected cost for this scenario is higher than for the scenario without airspace limitations. This can be clearly seen for the increased distance between the constant $N_s$ curves in Figs. 13 and 15. For the test scenario at hand, a population size of $N_s = 40$ outperforms $N_s = 20$ when the number of evaluated solutions exceeds 1900; therefore, a population size of $N_s = 40$ was used in most simulations of this scenario.

Figure 16 analyzes the sensitivity to the UAVs' minimal turn radius. As expected, the team tracking performance degrades as the minimal turn radius increases, but the performance is still good even when the minimal turn radius is increased to $R_{\min} = 400$ m. The team cooperation compensates for the higher minimal turn radius in this case.

## VII. Conclusions

The tracking problem of a ground moving target in an urban terrain, with and without airspace limitations, was studied. In the investigated problem multiple UAVs cooperate in performing the task. It was shown that this task can be automated in real time, thus reducing considerably operators' workload in such real-life problems.

Occlusion maps of known urban terrains can be transformed into visibility polygons, in a given altitude, in which a UAV must remain to be able to continuously track the target. Because the target might be moving, and the UAV has a minimal turn radius, it may not be able to remain in a given set of visibility polygons. Cooperation between the UAVs can significantly minimize the duration of time during which the target is occluded by buildings.

To ensure flyable trajectories, the path planning for each UAV must account for its dynamic constraints. The problem is computationally intractable for real-time solution using methods such as dynamic programming. To provide an implementable solution, stochastic search methods such as genetic algorithms can be used. Such an algorithm was derived and implemented in a high fidelity simulation test bed using a visual database of an actual city. It was shown that the algorithm is scalable and can be applied to large-sized problems.

## Appendix: Deterministic Search Algorithm Complexity

Deterministic search algorithms ($A^*$ and Dijkstra for example) might also be used for solving the optimization problem at hand. Their main drawback in this case is their computational complexity. To implement a deterministic search algorithm, each UAVs' control input in the time interval $k$ has to be taken from a closed set

$$u_{\dot{\psi}i}^k \in \{(u_{\dot{\psi}i})^1, (u_{\dot{\psi}i})^2, \ldots, (u_{\dot{\psi}i})^q\} \subset [-u_{\dot{\psi}i}^{\max}, u_{\dot{\psi}i}^{\max}]$$

Using a string of discrete acceleration commands similarly to Sec. V.A for each UAV yields a feasible solution search space of $N_{fs} = q^{(M \cdot N_u)}$, where $M$ is the control input string length for each UAV, $q$ is the number of possible discrete acceleration commands for each UAV at any time interval, and $N_u$ is the number of UAVs. Solving a similar scenario to the ones presented in Sec. VI, even with only $q = 3$, would, therefore, result in an enormous search space, which cannot be solved in real time using deterministic search algorithms.

## Acknowledgments

This work was supported in part by the Israeli Government and a Horev fellowship through the Taub Foundation. The authors would like to thank Jacob Silbiger and his team at Synergy, Ltd., for helpful discussions and help in integrating our algorithm with their Tri-Malat simulation environment.

## References

[1] "Unmanned Aerial Vehicles Roadmap 2002–2027," Office of the Secretary of Defense: United States of America, Dec. 2002.

[2] La Valle, M. S., Gonzalez Banos, H. H., Becker, C., and Latombe, C. J., "Motion Strategies for Maintaining Visibility of a Moving Target," *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE Press, Piscataway, NJ, 1997, pp. 731–736.

[3] Cid, M. R., Gonzalez Banos, H. H., and Tover, B., "A Reactive Motion Planner to Maintain Visibility of Unpredictable Targets," *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE Press, Piscataway, NJ, 2002, pp. 4242–4248.

[4] Gonzalez Banos, H. H., Lee, Y. C., and Latombe, C. J., "Real-time Combinatorial Tracking of a Target Moving Unpredictably Among Obstacles," *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE Press, Piscataway, NJ, 2002, pp. 1683–1690.

[5] Shima, T., Rasmussen, J. S., Sparks, G. A., and Passino, M. K., "Multiple Task Assignments for Cooperating Uninhabited Aerial Vehicles Using Genetic Algorithms," *Computers and Operations Research*, Vol. 33, No. 11, 2006, pp. 3252–3269. doi:10.1016/j.cor.2005.02.039

[6] Dubins, L., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, Vol. 79, No. 3, 1957, pp. 497–516. doi:10.2307/2372560

[7] Rasmussen, J. S., and Shima, T., "Tree Search for Assigning Cooperating UAVs to Multiple Tasks," *International Journal of Robust and Nonlinear Control*, Vol. 18, No. 2, 2008, pp. 135–153. doi:10.1002/rnc.1257

[8] Shima, T., Rasmussen, S., and Gross, D., "Assigning Micro UAVs to Task Tours in an Urban Terrain," *IEEE Transactions on Control System Technology*, Vol. 15, No. 4, 2007, pp. 601–612. doi:10.1109/TCST.2007.899154

[9] McLain, W. T., and Beard, W. R., "Coordination Variables, Coordination Functions, and Cooperative Timing Missions," *Journal of Guidance, Control, and Dynamics*, Vol. 28, No. 1, 2005, pp. 150–161. doi:10.2514/1.5791

[10] La Valle, M. S., "Robot Motion Planning: A Game-Theoretic Foundation," *Algorithmica*, Vol. 26, Nos. 3–4, 2000, pp. 430–465. doi:10.1007/s004539910020

[11] Frazzoli, E., Dahleh, A. M., and Feron, E., "Real-Time Motion Planning for Agile Autonomous Vehicles," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, 2002, pp. 116–129.

[12] Bryson, A. E., and Ho, Y. C., *Applied Optimal Control*, Prentice–Hall, Hemisphere, New York, 1975.

[13] Kirk, E. D., *Optimal Control Theory*, Prentice–Hall, Englewood Cliffs, New Jersey, 1970.

[14] La Valle, M. S., *Planning Algorithms*, Cambridge Univ. Press, New York, 2006.

[15] La Valle, S., and Kuffner, J., "Randomized Kinodynamic Planning," *Proceedings of the IEEE International Conference on Robotics and Automation*, IEEE Publications, Piscataway, NJ, 1999, pp. 473–479.

[16] Hwang, Y. K., and Ahuja, N., "A Potential Field Approach to Path Planning," *IEEE Transactions on Robotics and Automation*, Vol. 8, No. 1, 1992, pp. 23–32. doi:10.1109/70.127236

[17] Sasiadek, Z. J., and Duleba, I., "3D Local Trajectory Planner for UAV," *Journal of Intelligent and Robotic Systems: Theory and Applications*, Vol. 29, No. 2, 2000, pp. 191–210. doi:10.1023/A:1008108910932

[18] Spall, C. J., *Introduction to Stochastic Search and Optimization*, Wiley-Interscience, Hoboken, NJ, 2003.

[19] Xiao, J., Michalewicz, Z., Zhang, L., and Trojanowski, K., "Adaptive Evolutionary Planner/Navigator for Mobile Robots," *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, 1997, pp. 18–28. doi:10.1109/4235.585889

[20] Fogel, B. D., and Fogel, J. L., "Optimal Routing of Multiple Autonomous Under water Vehicles through Evolutionary Programming," *Proceedings of the 1990 Symposium on Autonomous Underwater Vehicle Technology*, IEEE Press, Piscataway, NJ, 1990, pp. 44–47.

[21] Capozzi, J. B., "Evolution-Based Path Planning and Management for Autonomous Vehicles," Ph.D. Dissertation, University of Washington, Seattle, WA, 2001.

[22] Goodman, E. J., and O'Rourke, J., *Handbook of Discrete and Computational Geometry*, CRC Press, Boca Raton, NY, 1997.

[23] Shaferman, V., and Shima, T., "Linear Quadratic Guidance Laws for Imposing a Terminal Intercept Angle," *Journal of Guidance, Control, and Dynamics*, Vol. 31, No. 5, 2008, pp. 1400–1412.