Unmanned Systems, Vol. 1, No. 1 (2013) 121−142

DOI: 10.1142/S2301385013500064

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us

# Cooperative Search and Exploration in Robotic Networks

Jinwen Hu[*,§], Jun Xu[†,¶], Lihua Xie[‡,‖]

[*]*Singapore Institute of Manufacturing Technology, Singapore 638075*

[†]*Western Digital Technologies, Singapore 118261*

[‡]*Nanyang Technological University, Singapore 639798*



Great potentials of robotic networks have been found in numerous applications such as environmental monitoring, battlefield surveillance, target search and rescue, oil and gas exploration, etc. A networked multi-robot system allows cooperative actions among robots and can achieve much beyond the summed capabilities of each individual robot. However, it also poses new research and technical challenges including novel methods for multi-agent data fusion, topology control and cooperative path planning, etc. In this paper, we review recent developments in cooperative control of robotic networks with focus on search and exploration. We shall first present a general formulation of the search and exploration problem, and then divide the overall search strategy into different modules based on their functions. Methods and algorithms are illustrated and compared following the classification of the modules. Moreover, a 3D simulator developed in our laboratory is introduced and its application is demonstrated by experiments. Finally, challenges and future research in this area are provided.

*Keywords*: Search and exploration; cooperative control; networked control; unmanned system; simulation.

US

## 1. Introduction

Due to the continuous development of advanced mechatronic technologies, robotic networks become possible in many applications including environmental monitoring, battlefield surveillance, target search and rescue, intruder detection and interception, and exploration of unsecured regions, etc., see e.g., Refs. 1−8. A robotic network is formed by a group of autonomous robots (or agents), each of which is equipped with sensing devices to monitor a variety of environment conditions such as obstacles, temperature, pressure, humidity, acoustic intensity, acceleration, magnetism, lighting conditions, and so on. Each agent also possesses a communication module by which its sensing results and decision information can be propagated to some other agents. The behaviors of agents are controlled by a human-accessible central station that can communicate with all of them (so-called centralized control) or separately by a computation unit installed on each one of them (so-called decentralized control). Decentralized control methods have been shown to be more suitable for large-scale networks with robots of low communication and computation capabilities.[2,9,10]

In recent years, great attentions have been drawn by the applications of robotic networks in search and exploration such as target search and rescue, intruder detection and interception, terrain and mineral exploration, forest fire detection, etc.[8,11−15] Compared to the search and exploration by a single robot that was addressed by some early works,[16,17] it is a much more complicated task for multi-robot networks which requires novel methods for dynamic data fusion, topology control, task allocation, coverage control and path planning. In addition, centralized control and optimization may not be feasible in practice and confines the scalability of the network.[18] Thus, decentralized or distributed control has nearly become a must for large scale networks, though only a suboptimal solution can be obtained most of time. Other challenges include resource

Received 29 January 2013; Revised 16 March 2013; Accepted 30 March 2013; Published 20 May 2013. This paper was recommended for publication in revised form by the Editors-in-Chief.
Email Addresses: §*jwhu@SIMTech.a-star.edu.sg*, ¶*jun.xu@wdc.com*, ‖*elhxie@ntu.edu.sg*

limitations (e.g., limited sensing region, communication region, energy), switching communication topologies, network constraints (e.g., limited bandwidth, delays and package drops), time constraints, unknown environment, incomplete or uncertain location information, computational complexity, obstacle and collision avoidance, nonholonomic system dynamics, etc.

The purpose of this paper is to provide a review, to the best of our knowledge, on recent developments and ongoing research focused on cooperative control of robotic networks in search and exploration. However, due to the space limitation, it is impossible for us to review all relevant works of the area. Readers may refer to[1,2] for a broader view of research directions in robotic networks such as formation, rendezvous and self-configuration, etc. The remainder of this paper is organized as follows. Terminology and problem formulation are presented in Sec. 2. An overview of cooperative search and exploration methods is provided in Sec. 3, which is followed by the demonstration of simulator and experiment implemented on our laboratory platform in Sec. 4. Challenges and future research are discussed in Sec. 5. Conclusions are drawn in Sec. 6.



## 2. Problem Formulation

On the start of reviewing methods and results, we need to clarify the terminology and problem formulation for search and exploration. Consider a set of $N$ agents with synchronous timing which are deployed to search or explore some desired information about a geometrically bounded surface region $\mathbb{O}$ ($\mathbb{O} \in \mathbb{R}^2$ or $\mathbb{R}^3$). Each agent takes observations of environment and communicates with other agents at discrete time steps denoted by positive integer $k$, i.e., during the $k$th information processing cycle of time interval $[kT, (k+1)T]$. Note that communication and observation of an agent may not be carried out synchronously at each time $k$. Each agent is assumed to have limited sensing range and communication range. Thus, agent $i$ at time step $k$ can only directly communicate with a subset of all agents, i.e., $\mathcal{N}_i(k)$, and observe a limited area $O_i(k) \subset \mathbb{O}$. The agents in $\mathcal{N}_i(k)$ are called neighbors of agent $i$ at time $k$ and $\mathcal{N}_i(k)$ may be time-varying due to the movement of agents, i.e., the network topology may be switching from time to time. An application example of unmanned aerial vehicle networks in ground target search and terrain exploration is shown in Fig. 1.

During search and exploration, agent $i$ may keep a knowledge map $Q_i(q) : \mathbb{O} \to \mathbb{S}$ in storage registering the environmental information at position $q$ in $\mathbb{O}$, where $\mathbb{S} \subset \mathbb{R}$ with $\mathbb{R}$ being the set of real numbers and $Q_i(q)$ denotes the value of a concerned environmental quantity evaluated at point $q$ within $\mathbb{O}$. In the following, we denote by $Q_i$ the

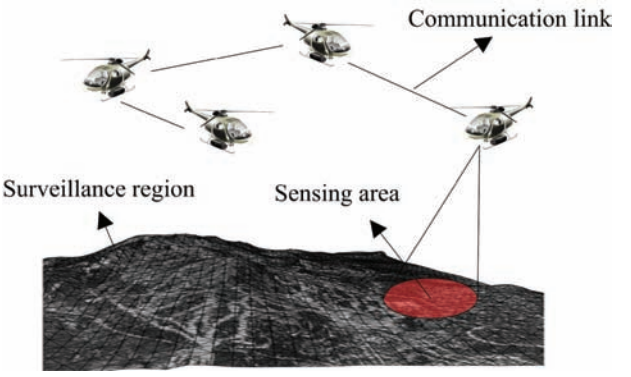



Fig. 1. Cooperative search and exploration in UAV networks.

entire map of agent $i$ across all points in $\mathbb{O}$ if $q$ is dropped, i.e., $Q_i = \{\{Q_i(q), q\} : q \in \mathbb{O}\}$. The set $\mathbb{S}$ is related to the problem under consideration. For target search, $\mathbb{S} = [0, 1]$ and $Q_i$, often denoted by a probability, represents the confidence or belief of agent $i$ in that a target is located at point $p$. In exploration, $\mathbb{S}$ can be any subset of $\mathbb{R}$ and $Q_i$ represents the estimated value of a certain environmental quantity. Note that in a centralized control strategy, there is only one map kept in the central station, and it can be treated as the special case that $Q_i = Q_j \ \forall i \neq j$, $i, j = 1, \ldots, N$. By taking observations from the environment, the map can be iteratively updated by detection results and information sharing with other agents, which is described by

$$Q_i(k+1) = g\left(\bigcup_{j \in \mathcal{N}_i(k) \cup \{i\}} \{Q_j(k), , Z_j(k)\}\right), \qquad (1)$$

where $Z_i(k) = \{z_i(k, q), \forall q \in O_i(k)\}$ denotes the set of detection results $z_i(k, q)$ obtained by agent $i$ at time step $k$. Function $g$ includes the fusion of measurements and individual knowledge maps of different agents.

According to the knowledge map $Q_i$, or due to any control command sent from the central station, agent $i$ makes a decision in a real-time manner on what reactions should be taken such as alarming and attacking, and, more frequently, on future path planning for complete search of the entire region. The searching path of each agent can be preplanned off-line or dynamically tuned on-line through coordinated efforts. These control decisions must be from a predefined decision set $\mathcal{A}$ which includes all admissible actions associated with an event set $\mathcal{B}$ that includes all events of possible occurrence during search and exploration. For example, when an agent detects an obstacle during search which is an event defined in $\mathcal{B}$, it first looks up the decision in $\mathcal{A}$ associated to that event, say stopping search and avoiding the obstacle, and then executes the decision.

The general control strategy for target search and exploration can be formulated as the following optimization problem:

$$\begin{aligned} &\min_{r(0:t)} \mathcal{H}(\mathrm{Q}(t), r(0:t), t), \\ \text{s.t.}\quad &L_i(r_i(0:t)) \le \bar{L}_i, \forall\, i = 1, \ldots, N, \\ &D_i(r_i(t), u(t)) = 0, \forall\, i = 1, \ldots, N, \\ &u_i(t) \in U_i(t), \end{aligned} \tag{2}$$

where $\mathcal{H}$ denotes some desired performance index such as the uncertainty of the knowledge map of an agent, the time taken by the agents to discover all existing targets, the total energy consumption to finish the search, etc. $\mathcal{H}$ is evaluated after searching for a time length of $t$ and is related to the knowledge maps of agents. To avoid confusion with the discrete-time step $k$, we denote by $t$ the continuous-time index and assume $k = \lfloor t/T \rfloor$. The position of agent $i$ at time $t$ is denoted by $r_i(t)$ and its trajectory from time 0 to $t$ by $r_i(0:t)$. We define $\mathrm{Q}(t) = \{Q_1(t), \ldots, Q_N(t)\}$ and $r(0:t) = \{r_1(0:t), \ldots, r_N(0:t)\}$. In the constraints, $L_i(r_i(0:t))$ denotes the length of the trajectory of agent $i$ up to time $t$ which may be bounded by $\bar{L}_i$ due to limited energy; $D_i(r_i(t), u(t)) = 0$ denotes the dynamic model of agent $i$ which may consist of some differential equations; $u_i(t)$ denotes the control law of each agent which is usually constrained within a set $U_i(t)$ due to the system limitations or for the purposes of connectivity maintenance and obstacle avoidance. There might be some other constraints dependent on specific application conditions. For example, sensor's performance may be affected by environmental conditions and may not be identical everywhere. Sound propagation is susceptible to change with respect to water salinity and temperature. Angle-of-arrival-based sensor generally is line-of-sight only. Note that such an optimization problem is usually NP-hard and one has to simplify the problem by, for example, predefining the trajectory pattern, or using some distributed methods to get a suboptimal solution to this problem.

## 3. Overview of Search Algorithms

Based on the way of making decisions by each agent, two classes of control strategies can be defined. If the control decision of each agent is determined by a central station without which no agent can work properly, the control strategy is said to be centralized, otherwise, it is decentralized. Specifically, if the control decision of each agent is only dependent upon the information from itself and its neighbors, the control strategy is said to be distributed. The relations between the classes of control strategies are shown in Fig. 2. In general, distributed control strategies

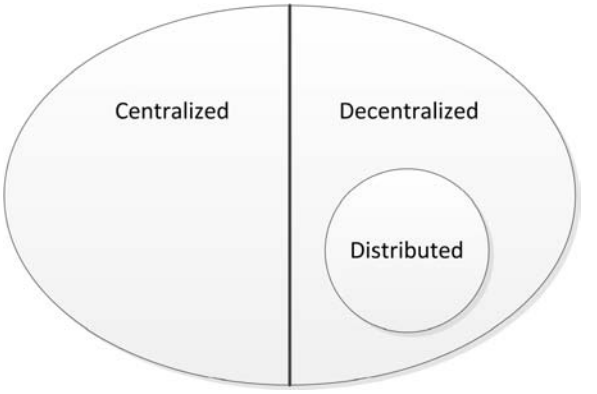



Fig. 2. Classification of control strategies.

have a lot of advantages[19] such as higher scalability (the computation load of the whole network is a polynomial function of the number of agents), easier to reconfigure (the control decision of each agent does not depend on a fixed communication topology) and more robust against unexpected change (the rest of agents will not work improperly when some agents break down). The closed-loop system diagram of distributed control strategies is shown in Fig. 3.

Basically, the information flow of all control strategies must go through the following modules in sequence: sensing, communication, decision and control (the sequence of the communication and decision modules can be changed according to requirement). Such classification of the modules is based on the functions that the whole system needs to accomplish in order to fulfill the overall search task, and it is the way of functioning of each module that differentiates one control strategy from another. Although the four modules are relatively independent in terms of functions, the functioning schemes of different strategies at a specific module may not be exchangeable. This is because the modules are correlated based on information running through them. For example, the information fusion and decision making at the decision module is highly related to the type of detection results obtained at the sensing module. The way of dealing with an image taken by a camera sensor must be different from the one with a signal strength taken by an acoustic sensor. However, in most cases, if the strategies are not limited to any specific type of sensor or application condition, their functioning schemes at each module are exchangeable based on different function performances to be achieved. For example, at the decision module, agents may need to do on-line path planning which is always formulated as an optimization problem and they can select different optimization criteria according to different requirement of performance.

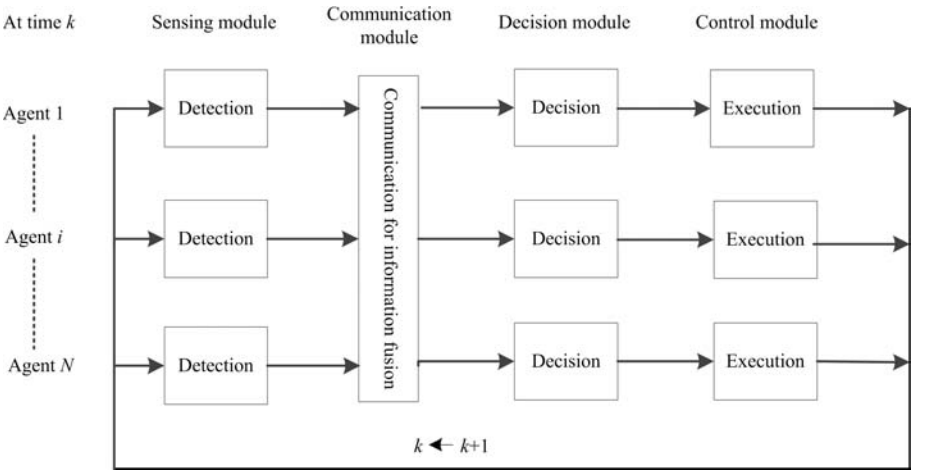



Fig. 3. Closed-loop diagram of distributed control strategies.



Being aware of such exchangeability and similarity of functioning schemes of different strategies, we will introduce and summarize the existing control strategies following the division of modules, and classify them by their similarities in functioning schemes at each module as shown in Table 1, where some application conditions are also included such as the target characteristics.

### 3.1. *Sensing module*

Sensing module directly deals with the raw detection data obtained by sensors. In this module, one needs to establish a sensing model and carry out signal noise filtering, raw data processing and fusion, etc. The sensing model is the key function of this module and basically gives a translation from

Table 1. Comparison between some typical search methods.

| Reference | Sensing model | Target characteristics | Connectivity maintenance | Info. sharing | Agent model | Online path optimization | If distributed |
|---|---|---|---|---|---|---|---|
| 20 | Deterministic | Multiple & stationary | No | No | Constant speed | No | No |
| 21 | Deterministic | Single & stationary | No | No | Physical model | Yes | No |
| 22 | Deterministic | Single & mobile | No | No | Constant speed | No | No |
| 23 | Probabilistic | Multiple & stationary | No | Yes | Constant speed | Yes | No |
| 24 | Deterministic | No | No | Yes | Way points | Yes | Yes |
| 25 | Probabilistic | Multiple & stationary | No | Yes | Way points | Yes | Yes |
| 26 | Probabilistic | Multiple & stationary | No | Yes | Way points | Yes | No |
| 27 | Deterministic | No | No | Yes | Physical model | Yes | No |
| 28 | Probabilistic | Single & mobile | No | Yes | Way points | Yes | No |
| 23 | Deterministic | Multiple & mobile | No | No | Constant speed | No | Yes |
| 29 | Probabilistic | No | No | Yes | Way points | Yes | Yes |
| 30 | Deterministic | No | No | Yes | Physical model | Yes | Yes |
| 31 | Probabilistic | No | No | Yes | Way points | Yes | No |
| 14 | Probabilistic | Multiple & stationary | No | Yes | Way points | Yes | Yes |
| 32 | Deterministic | No | No | Yes | Physical model | Yes | Yes |
| 33 | Probabilistic | No | Yes | Yes | Way points | Yes | Yes |
| 34 | Probabilistic | No | No | Yes | Physical model | Yes | Yes |
| 35 | Probabilistic | Multiple & stationary | No | Yes | Way points | Yes | Yes |
| 36 | Probabilistic | Multiple & stationary | Yes | Yes | Way points | Yes | Yes |
| 37 | Probabilistic | Multiple & stationary | Yes | Yes | Way points | Yes | Yes |
| 38 | Probabilistic | Single & mobile | No | Yes | Way points | Yes | No |

the raw detection data to some environmental information needed and a confidence or belief on the correctness of each detection result. For example, in vision-based target search, the sensing model gives not only detection results about where the targets are located from the image but also the associated probability of target detection and probability of false alarm. Two types of sensing models can be classified: deterministic and probabilistic, according to the belief of a detection result. A sensing model is said to be deterministic if the detection result is not a random variable given the true environment condition, otherwise it is probabilistic. In comparison, the methods assuming probabilistic detections are more general for universal applications and more suitable for stochastic environments in the real world.

#### 3.1.1. *Deterministic sensing model*

The target search problem has been studied in Refs. 21−23, where it is assumed that a target can be detected deterministically by an agent within its sensing region. In Ref. 21, the search problem for a stationary or slow moving target is addressed which is to be identified as a decoy or friendly asset. When an agent detects a target, another agent will be assigned to make a second detection from perpendicular direction to identify the target. On the other hand, the environment exploration problem is considered in Refs. 27 and 30. In Ref. 30, an uncertain density is used to model the lack of information about the surveillance region. The uncertainty associated with a point in the region is reduced by certain amount due to sensed information acquired by the agent closest to the point. The authors further extend the results by considering the limit on sensing range for each agent in Ref. 39. In Ref. 32, agents are deployed to learn a parameterized model of a constant environment field while seeking an optimal coverage configuration. The measurements are assumed to be the true values of the field parameters at sampling positions of agents.

#### 3.1.2. *Probabilistic sensing model*

Probabilistic detections have been considered in Refs. 25, 26, 35−38, 40−42, where the whole surveillance region is divided into a grid of cells and each cell is associated with a probability of target existence within the cell. Each agent can detect targets in each cell within its sensing region independently in a certain probability. In these methods, the binary detection model is assumed for each observation over $q \in O_i(k)$, i.e.,

$$z_i(k,q) = \begin{cases} 1, \text{ if sensor reports : that a} \\ \qquad \text{target is present at q;} \\ 0; \text{ otherwise,} \end{cases} \tag{3}$$

where $q$ is treated as a cell rather than a point. $z_i(k,q)$ is a random variable occurring with two key parameters, detection probability $\alpha_i(k,q) = P(z_i(k,q) = 1 | \theta(k,q) = 1)$ and false alarm probability $\beta_i(k,q) = P(z_i(k,q) = 1 | \theta(k,q) = 0)$, where $\theta(k,q) = 1$ (or 0) represents a target is present (or not present respectively) within cell $q$ at time step $k$. In this case, the knowledge map of agent $i$ is given by

$$Q_i(k,q) = P\left(\theta(k,q) = 1 | W_i(k), \bigcup_{l=1:k} Z_i(l), Q_i(0)\right),$$

where

$$W_i(k) = \bigcup_{l=1:k} \bigcup_{j \in \mathcal{N}_i(l)} \{Q_j(l), Z_j(l)\}.$$

The environment exploration problem is addressed in Refs. 27−29, 31, 33 and 34. In Refs. 29 and 31, agents are deployed to estimate the environment model, where each agent measures the unknown model parameters by a stochastic observation model with white Gaussian noises. The same sensing model is assumed in Ref. 33 for seeking peaks of an unknown scalar environment field. A similar gradient climbing problem is considered in Ref. 27, where each agent detects the field with a single sensor and the detection result is the value of the field at the sensing position contaminated by noise. Sensor placement and motion coordination strategies are studied in Ref. 28 for target tracking with range sensors in robotic networks. In Ref. 34, a dynamic stochastic environment field is detected and estimated by a group of agents, where the measurements of the agents are spatially and temporally correlated. White Gaussian process noises are also assumed for measurements. In these methods, the following sensing model is assumed:

$$z_i(k, r_i(k)) = h_i(r_i, x(k)) + v_i(k), \tag{4}$$

where $h_i(r_i, x(k))$ may be a nonlinear function related to the sensing position $r_i$, $x(k)$ is the state vector of environment field parameters and $v_i(k)$ is the measurement noise. In this case, the knowledge map of agent $i$ is given by

$$Q_i(k,q) = h_i(q, \hat{x}_i(k)),$$

where $\hat{x}_i(k)$ is the estimate of $x(k)$ at time step $k$ by agent $i$.

### 3.2. *Communication module*

The communication module is responsible for establishing necessary communication links between agents for information sharing. In this part, we do not want to discuss in detail about the physical level problems such as power setting, communication reliability, routing protocols, channel management, etc, as these topics are out of the authors' expertise. Readers may find more discussions and results on these problems in literature relating to network communications such as Refs. 43−45. Our focus is on the application level

problems such as what information should be communicated and which agents should be linked. As illustrated in Sec. 2, it is the communication scheme that defines the property of the overall control strategy, i.e., whether it is distributed, decentralized or centralized.In this part, we discuss two basic issues related to the communication module: (i) determination on information to be communicated; (ii) connectivity maintenance.

#### 3.2.1. *Determination of communication information*

To save energy consumption in communications, one should determine the least necessary information to be communicated based on application needs. Generally, translated detection results are preferred to raw sensing data since detection results are abstracted information taking up less storage space and requiring lower bandwidth for transmission. For example, in Refs. 35−37, the raw sensing data are translated to two different detection results, 0 (no target detected) or 1 (target detected). In some cases, only communicating the detection results is enough to get the desired performance. For example, in centralized control strategies,[26,40] detection results of each agent are sent to the central station for information fusion, through which the central station can promptly get knowledge of the places in the surveillance region detected by all agents. However, it may not be enough to achieve the desired performance in some distributed control strategies such as Ref. 37, where each agent also communicates their accumulated knowledge map about the environment in order to make each agent benefit from the information acquired by other agents in addition to its neighbors. Besides the sensing information, agents also need to communicate their positions which will be utilized in the control module for cooperative motion control.

#### 3.2.2. *Connectivity maintenance*

Connectivity maintenance of the entire robotic network is critical for efficient information sharing and cooperative control. A network with high degree of connectivity is more robust against accidental failures of some agents. In stationary networks, connectivity problem is more of a routing problem, i.e., to find a path that can connect one agent with any other in the network. However, in robotic networks, connectivity maintenance is always formulated as some constraints on the control of each agent, i.e., to constrain the movement of each agent within some limited space such that certain degree of network connectivity is guaranteed.[37,46] For example, in Ref. 46, based on the current positions $r_i(k)$ and $r_j(k)$ of two neighboring agents $i$ and $j$, a pairwise constrained set for the motion control of the two agents is defined as

$$\chi_{ij}(k) = \left\{ q : \left\| q - \frac{r_i(k) + r_j(k)}{2} \right\| \le R_c, q \in \mathbb{R}^2 \right\},$$

where $R_c$ is the communication range of each agent. Thus, to maintain the connectivity between each agent and all its neighbors at each time step, one should constrain the control of each agent by

$$u_i(k) \in \bigcap_{j \in \mathcal{N}_i(k)} \chi_{i,j}(k), \quad \forall i = 1, \ldots, N,$$

where $u_i(k) = r_i(k+1) - r_i(k)$. If the movement of an agent exceeds the constraint, there might be a chance for the agent to lose connections with some agents and thus to break the network connectivity. Unfortunately, not many control strategies in the existing literatures have considered this problem.

### 3.3. *Decision module*

In this module, the functions of information fusion and decision making are realized. The function of information fusion mainly deals with the fusion of detection results and knowledge map sent from other agents into its own accumulated knowledge map about the environment. The function of decision making refers to the selection of a control decision at each time from the given decision set $\mathcal{A}$ based on the accumulated knowledge. Decision making is a more application oriented function, where one needs to define some rules beforehand that map each event to a decision in $\mathcal{A}$ according to application background. When the occurrence of a certain event is confirmed according to the accumulated knowledge, a control decision must be made following the rule and leave the control execution problem for task accomplishment to the control module. For example, when an enemy is confirmed to be present at certain place by detection, an agent may be assigned to track or destroy the enemy. The more challenging issue is how to realize the fusion of information coming from different agents since it may involve detection results from heterogeneous sensors, correlated information from different agents, and its coupled influence on the control module. Moreover, in distributed control strategies, the information fusion schemes are also distributed, which means each agent can only get local information but is required to estimate the state of the overall environment. This makes the distributed schemes much more complicated to design.

Fusion of the detection results into the knowledge map is a challenge only when the sensing model is probabilistic. For the cell-based sensing model (3), a commonly used

method for fusion of detection results is the Bayesian method[25,26,35–38,41,42]:

$$Q_i(k+1,q) = \begin{cases} \dfrac{\alpha_i(k,q)Q_i(k,q)}{\alpha_i(k,q)Q_i(k,q)+\beta_i(k,q)(1-Q_i(k,q))}, & \text{if } z_i(k,q)=1; \\ \dfrac{(1-\alpha_i(k,q))Q_i(k,q)}{(1-\alpha_i(k,q))Q_i(k,q)+(1-\beta_i(k,q))(1-Q_i(k,q))}, & \text{if } z_i(k,q)=0; \\ Q_i(k,q), & \text{otherwise.} \end{cases}$$

For the sensing model (4), many conventional filtering methods can be used such as Bayesian filtering, Kalman filtering, Particle filtering, etc.[27,29,31,32,34]

Fusion of the knowledge maps of different agents is more complex since the correlation between individual maps is usually hard to compute especially in distributed strategies. Without the correlation information, the fusion is mostly done by a weighted average combination of individual maps.[34,35,37] For example, in Ref. 37, the maps of neighboring agents are fused by a consensus-based protocol:

$$Q_i(k+1,q) = Q_i(k,q) + \sum_{j\in\mathcal{N}_i(k)} \frac{1}{N}(Q_j(k,q) - Q_i(k,q)).$$

Such protocol will not only help each agent fuse information from other agents, but also make their maps converge to the same. Sometimes, agents do not communicate the maps, but some parameters constitute the maps. For example, in Ref. 29, the estimates of the environment state vector are communicated by agents instead of the whole environment field map. In such case, there have been many distributed filtering methods for fusing correlated estimates which is a very large scope that cannot be enclosed in this paper.[47–54]

In Ref. 25, a target within a cell can only be detected with some probability and the probabilities of target existence within all cells form a probability map. The detection results of different agents are assumed to be independent and the detection results of neighboring agents are fused by the Bayesian rule at each time step in a distributed manner. The same cell-based parablastic method for fusion update by detection results is used for datafusion in Refs. 26, 35–38, 41 and 42 except that a centralized probability map at the central station is required to be accessible by all agents in Refs. 26 and 41. In Ref. 38, a single framework for searching and tracking for a single mobile target is established based on nonlinear representations of the target position probability density function, where particle filtering method is utilized for target tracking and fusion of detection results. In Ref. 29, the local measurement data of neighboring agents are fused using a proportional-integral average consensus estimator by information sharing, and then filtered by Kalman filtering to estimate the parameters. A similar consensus-based information fusion method is used in Ref. 32 for agents to achieve the same parameter estimate. In Ref. 31, the environment model parameter is estimated based on Bayesian filtering and bounded error assumption. In Ref. 27, the range-based detection results with noises of all agents are collected by the central station for estimation of the target position by extended Kalman filtering. In Ref. 33, each agent maintains its own local estimate of the environment field and updates the estimate using collective measurements from itself and neighboring agents by recursive least square method. In Ref. 42, detection results are exchanged between neighboring agents for information fusion, where the number of historical observations that need to be diffused to neighbors from an agent at each time is related to its information divergence which is defined as the change of its probability map. In Ref. 35, after the probability map update by measurement, each agent transmits its probability map to its neighbors for map fusion, which is fully distributed. However, in a network which is not fully connected, each agent can only obtain information directly from its neighboring agents. The lack of information correlation makes the map fusion difficult and only an intuitive fusion method is given in Ref. 35. In Ref. 36, each agent only communicates the latest observations to neighbors for information sharing, but limited field-of-view constraint is considered for target detection. In Ref. 37, the nonlinear Bayesian update of the probability map is transformed to be a linear update of a new information map, which saves the computation load. A consensus-like distributed information sharing scheme is proposed to fuse the probability maps of different agents by ignoring the correlation information. In Ref. 34, the estimation of a spatio-temporal stochastic field is formulated as Bayesian universal Kriging and a distributed algorithm is designed to compute weighted least squares estimates, which combines the Jacobi overrelaxation method with dynamic average consensus algorithms. Then, a so-called distributed Kriged Kalman filter is designed for predictive inference of the spatial field and its gradient.

### 3.4. *Control module*

The function of this module is to realize the control decision made by the decision module, where a common aspect in the context of search and exploration is motion control and path planning. Generally, motion control refers to the control of an agent to move from current position to a desired position, which may involve the agent's physical dynamics,

while path planning aims to find out a path for each agent to follow which goes across a sequence of desired waypoints in a given time horizon, where the agent's physical dynamics may be ignored. When a motion control scheme is provided, path planning is often not necessary. Depending on the assumptions, a control strategy may or may not put the true dynamics of agents into consideration for the ease of theoretic analysis. However, this is an innegligible issue in the real applications of a control strategy, and thus is listed as an index for comparing different control strategies as shown in Table 1.

#### 3.4.1. *Motion control model*

Motion control models can be in continuous-time formulation or discrete-time formulation. When given in continuous-time formulation, a motion model of agent is normally coherent with its dynamic model. When given in discrete-time formulation, it can be seen as a model for one step ahead path planning. The physical dynamic model of an agent is usually described by a set of differential equations subject to limited velocity and acceleration. In Ref. 21, a two-dimensional dynamic model is assumed for each agent with constant speed and controllable heading maneuvering in a fixed horizontal plane. A control law based on dynamic inversion is utilized to make vehicles track the assigned lanes while considering the maximum turn radius constraint. The search task of all robots is stopped if a target is found. In Refs. 22 and 23, agents move at a constant speed according to some predefined parallel line patterns without considering the physical dynamic model of agents. However, system parameters can be easily evaluated off-line in well-defined patterns such as the minimum number of agents required for task completion. In Refs. 24, 25, 35, 37 and 40, the motion of each agent is described by a movement from one cell to another. In Refs. 14, 28, 29, 31, 33 and 38, a discrete-time motion model is assumed for the position control of each agent which provides the waypoints along a path. In Refs. 27, 30, 32 and 34, first-order integrator dynamics are assumed for each agent.

#### 3.4.2. *Path planning schemes*

Path planning has been widely studied in robotic networks for different application backgrounds such as formation,[55–57] flocking,[58–60] swarming,[61] gradient climbing,[27] deployment,[62–65] rendezvous,[66–69] etc. Tens of different path planning algorithms have been reported in the literature. Some researchers divide them into two classes depending on whether an algorithm considers the differential constraints of the robots, and some classify them based on whether the information about the search environment is completely known (or static), and some generally sort into deterministic and probabilistic approaches, and some simply consider whether the approach is local or global.[55–57,70]

Generally speaking, two main steps are involved in a path planning scheme. The first step compiles the available information into an effective and appropriate configuration space $\mathcal{O}$, and the second step uses a search algorithm to find the optimal/sub-optimal path in $\mathcal{O}$ based on the pre-defined criteria or constraints, e.g., system dynamics, path distance, obstacles, and so on. Several main categories of the configuration spaces are widely applied in search and exploration, e.g., cell decomposition, roadmaps, and potential fields. Cell decomposition-based approaches, including grid-based method, use a set of representative areas to represent $\mathcal{O}$, such as convex polygons, and then describe the characteristics of each cell. Grid-based approaches are usually employed for low-dimensional problems and often need to search repeatedly. The roadmap-based approaches describe $\mathcal{O}$ in terms of how to get from one key location to another, and the cost of getting between them. The potential fields-based approaches represent $\mathcal{O}$ under the influence of virtual force generated by goals (attraction) and obstacles (repulsion), which have advantages in less computation and disadvantages in being easily trapped into a local minimum of the potential field. Many techniques are proposed to over come the problem of getting stuck in a local minimum, e.g., including techniques for escaping minima or with the definition of potential field which has no or few minima.[71] Note that these approaches may be mixed, e.g., a Voronoi graph can be configured with potential fields.[72]

Once the information abstraction is built, a search algorithm is applied. When the world is represented by graphs, less sophisticated algorithms e.g., Djikstra's algorithm and Depth-First Search, and complicated algorithms, e.g., $A^*$ algorithm, and $D^*$ algorithm, are widely used.

When optimization-based algorithm is considered, again, we obtain a formulation similar to (2) stated in Sec. 2. Hence, the traditional control strategies, such as dynamic programming[40] and receding horizon optimization,[35] can be applied too. As mentioned in Sec. 2, the optimization problem (2) is generally NP-hard. Therefore, one must make some simplifications or use distributed methods to get suboptimal solutions of the trajectories $r(0:t)$. In some search and exploration strategies, agents move according to predefined patterns like parallel straight line search which are determined off-line and can guarantee network connectivity and collision avoidance so that path planning is not a problem during search such as in Refs. 20, 22 and 73–75. In other strategies, agents must dynamically adjust and optimize their path planning based on the real-time updated knowledge, which may involve dynamic programming, topology control and collision avoidance such as in Refs. 25, 28–30, 34 and 37. In such dynamic path planning,

a distributed design for solving the optimization problem (2) is necessary for fast and efficient responses. A widely used method is the gradient-based control method to find a local optimum solution.[28,32,37,62,76] Take the one step-ahead planning scheme as an example. By ignoring the constraints, the dynamic path planning problem at a given time step $k$ can be formulated as the following optimization problem:

$$\min_{r(k+1)} \mathcal{H}(\mathrm{Q}(k+1), r(k+1), k+1), \tag{5}$$

i.e., to find the optimal places for the agents to move to at time step $k+1$. The cost function $\mathcal{H}$ should be properly designed such that $\frac{\partial \mathcal{H}(\mathrm{Q}(k+1), r(k+1), k+1)}{\partial r_i(k+1)}$ for each agent $i$ is not related to the information from any other agent $j \notin \mathcal{N}_i(k)$. Thus, a distributed gradient-based one-step ahead planning can be given as

$$r_i(k+1) = r_i(k) + G_i(k) \frac{\partial \mathcal{H}(\mathrm{Q}(k+1), r(k+1), k+1)}{\partial r_i(k+1)},$$

where $G_i(k)$ is a positive gain parameter which is critical to guarantee that $r_i(k+1)$ can be reached by agent $i$ at time step $k+1$ when putting the constraints in (2) into consideration. To further alleviate the computational load, sampling-based algorithms are explored. These algorithms represent the sampled configuration space $\mathcal{O}$. Typical techniques include rapidly exploring random trees and state-space navigation function with interpolation.[70]

For the cell decomposition approaches, many works have been reported. In Ref. 21, the whole surveillance region is divided into lanes and each robot is assigned to exhaustively search one lane for targets. Once an agent completes the search within its current lane, its task assignment is re-determined using dynamic programming. In Ref. 24, based on a map of uncertainty within each cell in the surveillance region, an information reward is defined for each agent by visiting a cell. A distributed path planning algorithm is proposed which gives a 2-step ahead path planning for each agent that maximizes its predicted information reward along the path. In Ref. 25, the target search problem is considered by using a similar method. Each agent keeps a probability map of target existence within each cell, based on which the information reward is defined. Neural networks are used to learn and predict the number of agents occupying a cell for the prediction of the information reward by visiting the cell in future steps. In Ref. 40, a software architecture is developed for cooperative control of UAV networks, where the threat of environment, probabilistic existence of targets and the cooperative path planning are processed at a central station. The dynamic programming approach is used for path planning with considering the information gain as well as the threats. In Ref. 42, a co-evolutionary approach of search path planning is proposed under constrained information sharing for multi-agent systems. The communication bandwidth constraint is considered and a distributed path planning scheme is developed aiming at minimizing the team entropy over target existence within each cell. In Ref. 36, the proposed path planning algorithm aims to maximize the joint detection probability that a target is detected by at least one agent. The algorithm is distributed and incorporates the connectivity preservation for the network. In Ref. 37, the path planning is formulated as a coverage control problem, i.e., control the agents to maximize the collective sensing coverage area measured by a time-varying density function, which is related to the probability maps of agents and is updated in real time. The probability maps of all agents converge to be a same one that reflects the true target existence or nonexistence within each cell. Topology control is included to maintain network connectivity as a constraint on the former coverage control.

Roadmap-based path planning approaches have been considered in many papers, e.g., Refs. 30, 32, 35 and 39. In Ref. 30, agents are controlled to maximize a single-step uncertainty reduction integrated over the whole surveillance region. The uncertainty function which denotes the lack of information of each point in the region is updated in real time. Each agent performs search within its Voronoi cell and the deployment of agents finally converges to a centroidal Voronoi configuration, where each agent is located at the centroid of its Voronoi cell with perceived uncertainty density. The control strategy is further modified in the presence of constraints on robot speed and limit on sensor range in Ref. 39. In Ref. 35, each agent chooses a path, some number of time steps into the future that maximizes the likelihood of the agent finding a target. The receding horizon optimization is performed using the portion of the probability map that lies in the agent's Voronoi partition. In Ref. 32, the robots are controlled to move towards the estimated centroids of their Voronoi regions, respectively, while also make their estimates of the sensory distribution improve over time.

Potential field approaches are also widely applied. In Ref. 30, the optimal trajectories of agents are selected to maximize the expected value of the search by a fraction of the uncertainty in the uncertain posterior distribution, which is drawn from the uncertainty in the centralized probabilistic state transition matrix of targets. In Ref. 33, a distributed cooperative control method is proposed to let robots reach peaks of an unknown scalar field. Each agent is controlled to move towards a peak of the field using the gradient of its estimated environment field while avoiding collision and maintaining communication connectivity. In Ref. 34, a similar path planning scheme is applied for gradient climbing in a spatio-temporal stochastic field to estimate a spatio-temporal field. In Ref. 27, a stable control

strategy is proposed for groups of agents to move and reconfigure cooperatively in response to a sensed environment, which aims to seek out local maxima or minima in the environment field. The underlying coordination framework uses virtual bodies and artificial potentials. In Ref. 29, the problem of environmental modeling is addressed by measuring and estimating the model parameters at each time step. The motions of agents are controlled to maximize their sensory information relative to current uncertainties in the model. In Ref. 31, the path planning is formulated as a sequential optimization problem, where each agent sequentially determines its next sampling position by maximizing the determinant of the new estimation error covariance matrix.



## 4. Simulator and Experiment Development

### 4.1. *Background*

From the aforementioned discussion, it is generally difficult to integrate many techniques into a unified physical platform. However, simulation technologies have the potential to overcome the difficulty and accelerate processing procedures in a cost-effective way. For multi-agent networks, simulation technologies are especially important due to the additional complexity involved in collaboration, networking and communication.

2D simulation techniques have dominated the society for many years, e.g., player/stage. However, 3D techniques are attracting more and more attentions owning to the rapid developments on GPU technologies. Some recent works on robot have demonstrated the potential advantages of 3D simulation, e.g., Virtual Robot Experimentation Platform (V-REP),[77] Modular OpenRobots Simulation Engine (MORSE) ,[78] AGI, Xplane, OpenSim,[79] Delta3D,[80,81] Microsoft robotics studio and Gazebo[82] (see Ref. 83 for details). In addition, the new developments on serious games for the main purpose to train, investigate, or advertise,[84] also show the benefits of simulation techniques. For example, VBS2 (Virtual Battlespace 2)[85] is a realistic battlefield simulation system mainly for military training purpose. It not only offers the ability to operate land, sea, and air vehicles, but also creates new scenarios of different perspectives. DARWARS Ambush![86] is another PC-based, networked, multiplayer training simulator. It provides military training on the basis of practitioners' experiences, and includes capabilities to ensure the capture and dissemination of lessons learned.

Recently, USARSim (Unified system for automation and robot simulation),[87] which was initially developed for urban search and rescue simulation, attracts more attention nowadays, as it is a high fidelity simulation tool based on Unreal technology[88,89] with user-friendly API and some validated models. It accurately represents robot automation and behavior, and renders user interface elements in both local and remote environments. It has been applied in both education[90] (e.g., RoboCup rescue virtual robot competition[91,92] and IEEE Virtual Manufacturing Automation Competition VMAC[93]) and research (e.g., sensor and motion models of robots[94] and localization[95,96]).

Based on USARSim, a hybrid multi-purpose simulation platform for networked MAS (multi-agent system) under different scenarios is developed in Nanyang Technological University (NTU). The low-cost hybrid framework based on a hierarchical structure integrates some mission packages so that it can perform multiple tasks, e.g., formation flight, collaborative search and exploration (especially, vision-based approaches), collaborative tracking, and dynamic task assignment, in various 3D environments, e.g., urban, rural, foliage and indoor environments. Meanwhile, based on its detailed modeled system dynamics with actuators and sensors, the platform should enable the interaction between real world and virtual world so that hardware-in-the-loop simulation becomes possible. In addition, it can operate in wireless network environments with several different choices. Three implications of the hybrid simulation system are the interaction between the real world and virtual environment systems, the combination of hardware and software, and the hybrid algorithms used in MAS control.

### 4.2. *Simulator platform and design methodologies*

#### 4.2.1. *System architecture*

Figure 4 provides the system architecture, which is roughly divided into six portions, namely Real World, Transmission, Control Console, Control Algorithms, and Virtual Network. Virtual world is emulated by Unreal Engine,[88] created by Unreal Editor and further edited by Unreal Script. Unreal Editor enables game developers to build their own levels and create virtual environments which replicates real world with the aid of other 3D editors, e.g., 3DS Max.[97]

We have implemented three different programming interfaces: LabView, Matlab and OMNet++, whose control flows are indicated in different colors in Fig. 4.

- The first interface LabView,[98] acting as the console center for both hardware and software, is the interface to Unreal Engine and virtual environment. In fact, LabView communicates with the Unreal server to feedback sensor information, and sends commands to USARSim to control the motions of vehicles. The Unreal Engine supports multi-vehicle simulation, and each vehicle occupies one socket channel, which is created by the controller (LabView). Although some simple algorithms (including the

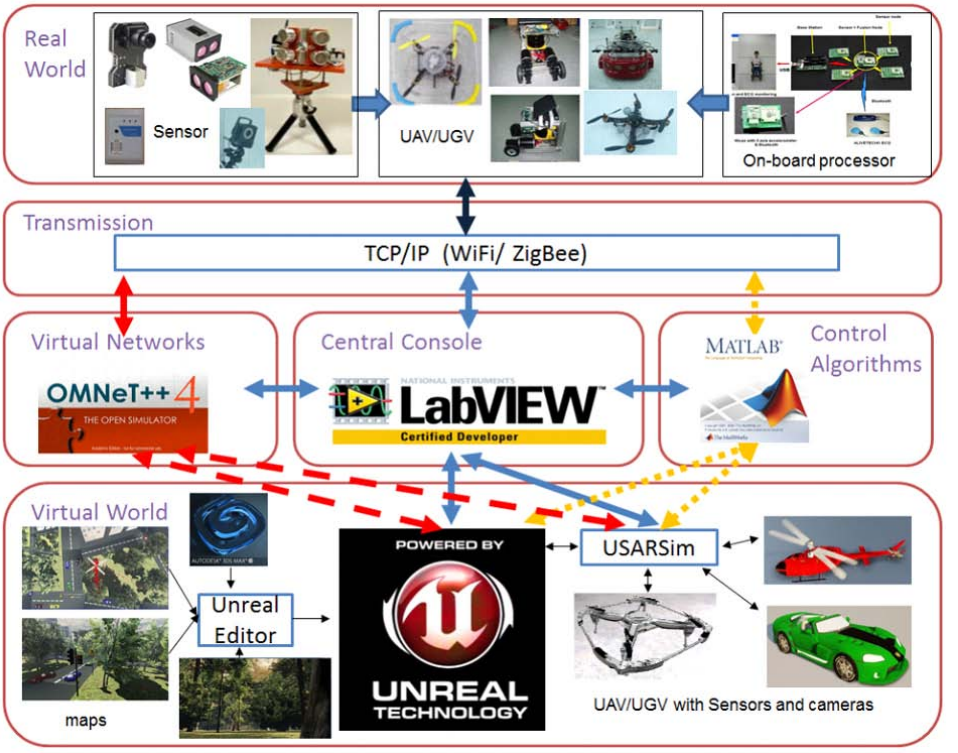



Fig. 4. Design architecture of hybrid simulator.

communication and system dynamic control) can be implemented in LabView directly (e.g., Mathscript), more complex control algorithms need to be implemented using Matlab, i.e., run Matlab Scripts in LabView via Matlab server through the ActiveX technique (Windows platform only). Other techniques, e.g., DLL (dynamic link library), COM and DDE (dynamic data exchange), can be also used to exchange data and send/receive commands between LabView and Matlab.

- The second interface Matlab directly communicates the Unreal through TCP/IP command from the Instrument Toolbox, which also provides the hardware interface. Note that we can also call java[99] or Winsock2 to perform the same functionality. By Using Simulink together with

Fig. 5. Hardware for HIL simulation.

Fig. 6. Simulated Qud-rotor UAV and Helicopter.

Real-time Toolbox, we can run the simulation in a real-time manner.

- The third interface OMNet++ connects to Unreal server using socket programming. MiXiM provides TCP/IP socket related functions for communication between OMNet++ and Unreal engine. This implementation provides more flexibility in studying the behavior of networks.

To enable hardware-in-the-loop (HIL) simulation, some hardware should be configured in the lab environments. For localization in real world, ultra-wideband (UWB)-based tool (e.g., Unisense system[a]), WiFi-based tool and vision-based tool (e.g., Vicon system[b]) are used in indoor environments, and Globe Position System (GPS) together with camera/IMU is used in outdoor environments. In the virtual world, range scanner sensor (sonar or IR) is used to measure the distance between vehicles and obstacles. Camera is used in both real and virtual worlds for tracking and navigation. Robots communicate betweeneach other through WiFi 802.11b, and connect with ground station through Zigbee in Real World, and OMNet++ is used to simulate the corresponding virtual networks, which will be discussed in Sec. 4.3.

#### 4.2.2. *Robot models*

USARSim[87] constructs a robot model by using chassis, joints, parts, and attached items. We have developed several new UAV models in the similar way, e.g., GAUI 330X Quad Flyer UAV[100] and Xcell-60 helicopter. The simulated Quad Flyer and the helicopter in UDK are shown in Fig. 6. Besides the rotary-wing UAVs, some fixed-wing and flapping-wing UAVs are also under development.[101]

As stated in Ref. 83, it is complicated to implement the full dynamics of a physical UAV in UDK, although Unreal Script provides the possibility. Hence, we can realize the inner-loop control (rigid-body dynamics and internal stability) in Matlab while reflects the outer-loop control (Kinematic model) in UDK. In other words, the inner-loop guarantees the asymptotic stability and disturbance-rejection performance of the UAV, while the outer-loop responds to the behavior request to achieve a certain flight trajectory with a desired orientation. Figure 7 gives an example of the hybrid configuration in UDK and Matlab/LabView. In addition, UDK provides simple embedded AI for the moving objects, e.g., obstacle avoidance, edge detection, predefined path following/tracking. We can use those properties to define the moving cars on the road and pedestrians on the cross, as well as the traffic control. Furthermore, the properties of PhysX Engine shall be considered.

#### 4.2.3. *Sensor and vision*

Search and exploration algorithms cannot work without the aid of sensing technologies. There are many classification methods of sensors. The most commonly used is to classify the main types of sensors into two domains: passive and active. Passive sensors can only work when the natural energy is available, e.g., visible and infrared spectrum camera, sound detector, odometer, IMU (inertial measurement unit, combination of accelerometer and gyroscope). Active sensors use their own energy, e.g., radar (radio detection and ranging), lidar/ladar (light/laser detection and ranging) and sonar (sound navigation and ranging). Some other classifications include proprioceptive sensors (e.g., IMU and GPS) versus exteroceptive sensors (e.g., camera, laser scanner, radar), contact sensors (e.g., touch probe) versus noncontact sensors (most of sensors), 1D sensors (distance sensors), 2D sensors (e.g., single camera, 2D radar) versus 3D sensors (e.g., stereo camera, 3D laser scanner and 3D flash lidar). Over 10 different sensors are provided in USARSim and the hybrid platform, e.g., INS, GPS and Range Scanner sensors (see Fig. 8), whose sensing data can be easily obtained from the return messages from Unreal.

[a] http://www.ubisense.net/
[b] http://www.vicon.com

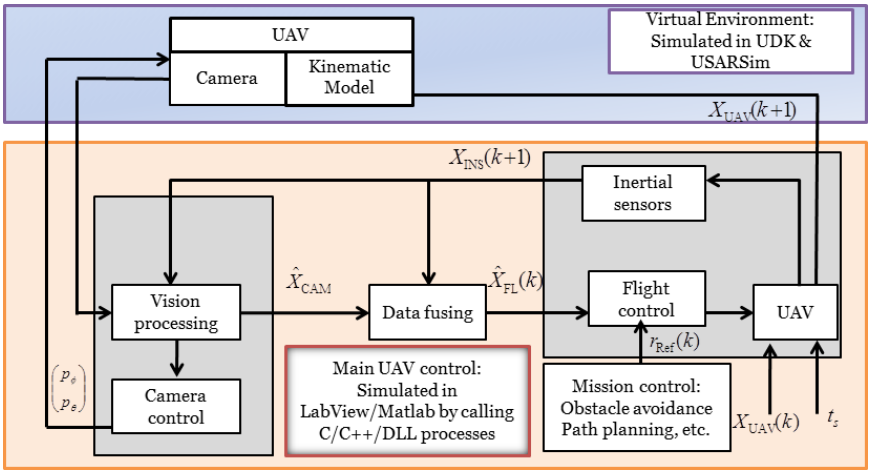



Fig. 7. Hybrid closed-loop control in a simulated UAV.

Among these sensors, vision-based sensors have attracted more attentions, e.g., (stereo/omni) camera. In LabView, we can simulate camera feedback by embedding the Unreal Client into the user interface (via calling Window API/ ActiveX component), directly capturing Unreal Client, or using the image server (UPIS) to get camera pictures. The communication mechanism between image server and LabView is similar to what is introduced in Sec. 2.1 of Ref. 83 with a different port. For Matlab implementation, the Machine Vision Toolbox[102] together with the Camera Calibration Toolbox is used.[103] Based on the latest OpenCV,[104] we can implement the efficient algorithm with different view angle. We can also extend the current 2D SLAM to 3D SLAM using VSLAM, PTAM or RGBD-SLAM algorithms through ROS (robot operating system).[105] Using our recently developed depth sensor, we can simulate the behavior of Kinect. Currently, we are actively developing the radar, ladar/lidar and multi-mode camera (e.g., night mode).

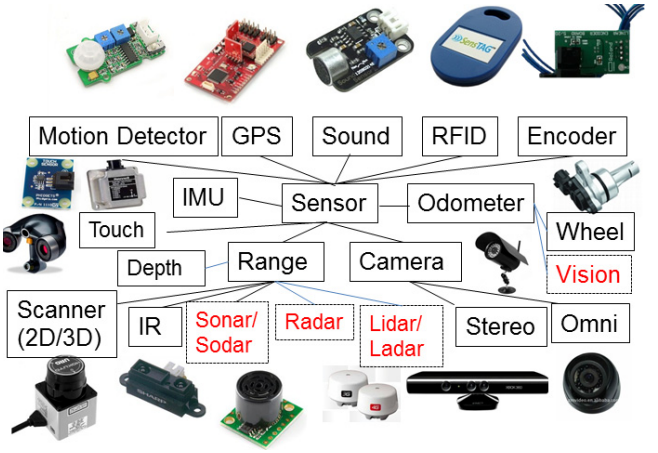



Fig. 8. The sensor structure.

#### 4.2.4. *Environment maps*

Both indoor and outdoor environments are created to satisfy different requirements. Besides the maps of NTU Sensor Network Lab (indoor), NTU Student Recreation Centre and the sketch map of NTU,[83] we also build several maps in UDK, e.g., the urban and rural environments, as shown in Fig. 9.

Note that it is not easy to develop a high quality and large scale virtual world in Unreal, although there exist some tools, e.g., the one based on OSG & OpenFlight[106] and the World Generator tool.[107] To build vivid environments, we can further apply Google SketchUp,[108] CityEngine[109] or Urban Pad,[110] and create some scripts to export t3d files such that some building models can be imported into UDK via the aid of 3DS Max. Meanwhile, the GIS data from OpenStreetMap.org is also useful for a fast prototype of the real environments.

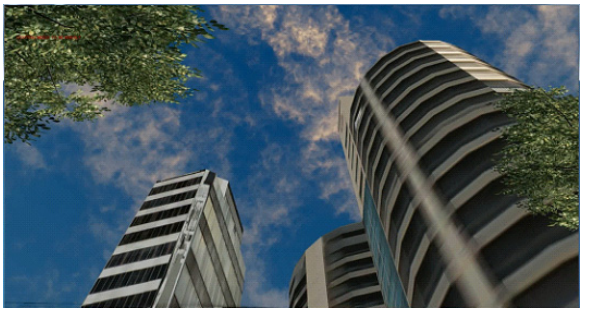
### 4.3. *Network simulation*

USARSim's simple wireless simulator named Wireless Simulation Server (WSS) captures some constraints of wireless communication, e.g., limited range and multi-hop routing,[111] however, it is not suitable if the network performance is a critical research objective. In our current implementation with LabView, we have coded some parallel wireless modules based on some mathematical formulations.

In order to model the complex behaviors of the networks, we apply OMNet++[112] together with MiXiM, which provides the infrastructure of wireless communication network. We develop two different approaches to fully use the functionality of OMNet++ so that some critical performance indexes, such as the package drop rate, transmission

Road intersection

Straight road



City view

City street

Small park

Woods

Fig. 9. Environment simulation.



delay and data rate, can be analyzed. One is to use the mixed socket programming between Labview/Matlab and OMNet++. The other uses C++ socket programming in OMNet++ to control the Unreal directly. The most important consideration is the synchronization of the various independent modules into one coherent unit which can function efficiently. This is achieved by the synchronization timer and the blocking nature of the sockets. Note that for Matlab/Simulink interface, an alternative option is the Simulink-based toolbox TrueTime[113] or PiccSim.[114] For example, if using TrueTime, we can create duplicated agents and run them simultaneously with UDK. There are two key issues to be solved. Besides synchronization, we also need to extend the tool from 2D to 3D.

#### 4.3.1. *Mixed approach with socket synchronization*

In this implementation, both the client and the server are implemented in C++, and LabView acts as the mediator. Figure 10 illustrates the main idea, where UAVs communicate with each other through wireless networks. The main components of UAVs in OMNet++ sides are the communication modules. Upon initialization, UAV clients in LabView send requests to establish connection with the relevant UAV servers in OMNet++. Once this connection is established data starts flowing between the two systems. Whenever a UAV needs its neighbors' information, a command is sent to OMNet++. The corresponding UAV in OMNet++ then communicates with the neighbor UAVs through the simulated

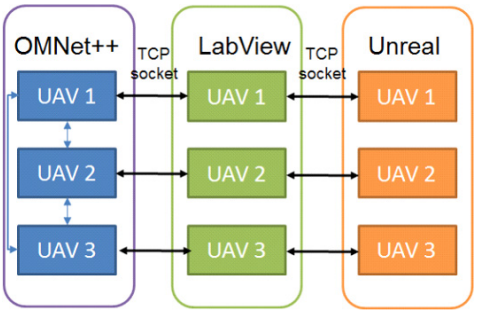



Fig. 10. Network structure of the simulator.

network and gets the required information. With this information in hand, a connection request is now sent by the OMNet++ client to connect the LabView server. Once accepted, the required information is transferred to LabView through the socket connection. Meanwhile, if the state information in LabView is updated, an "update" command is also sent to OMNet++ to update the corresponding UAV's state in OMNet++. Note that the design of OMNet++ socket and the detailed flowchart for message exchanges can be found in Ref. 101. In the simulation, we can maintain the socket connection throughout the duration of the execution, which partially removes the overheads due to multiple connection cycles as well as the possible de-synchronization problem. However, such a connection may result in energy waste and security problem for the physical systems.

#### 4.3.2. *OMNet++ socket control*

In this implementation, Omnet++ acts as the client connected to Unreal server directly to establish a channel through which these socket commands are sent to control the motion of the UAVs. The significance of this implementation lies in the fact that it eliminates LabView as the mediator and hence reduces overheads. However, the control algorithm should be implemented in C++ too. Most work is carried out in the application layer. The mobility layer was also used in order to simulate the movements of the UAVs. The timeline of each node is divided into time zones which are used for synchronization. A workflow on how to implement formation control is shown in Ref. 101.

### 4.4. *Applications in cooperative control*

The simulator has been applied in several projects since 2009.[83] Besides the applications in Refs. 83 and 101, several examples in cooperative control are presented to illustrate the usage of the simulator.

#### 4.4.1. *Cooperative formation control and coverage control*

Formation control and coverage control are widely applied in outdoor search and exploration for improved performance. We have implemented several formation scenarios,[83,101] where the formation algorithm introduced in Ref. 115 is applied. Some simple obstacleavoidance algorithms are also integrated in order to avoid the collision with the buildings as shown in Fig. 11.

Distributed coverage control using multiple UAVs often involves problems of three types: target (point) coverage, barrier (curve) coverage and area (surface) coverage. In the example displayed in Fig. 11 we show an area coverage scenario within a 500 m $\times$ 500 m area using the generalized locational optimization approach.[72]

#### 4.4.2. *Cooperative search and exploration*

We give two examples here. One uses a modified cooperative $A^*$ algorithm and the other applies the distributed search algorithm introduced in Ref. 37. In the first example, two UAVs are dynamically assigned to search targets at several possible target positions and they are required to find the actual target positions. They have range scanner sensors to measure the distance between UAVs and obstacles. Figure 12 shows the actual map of the indoor environment and one scenario that a UAV finds the target. Figure 13 provides the UAV trajectories at different stages/time.

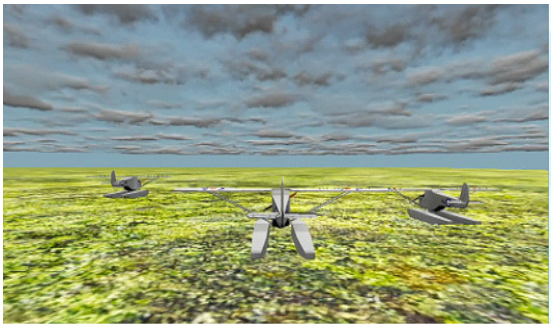
For the second example, two UAVs cooperatively search two targets in a noisy environment with cars, trees, roads and buildings. The target positions are unknown to UAVs, although the UAVs have rough information about the environment, i.e., the rough size of the map. Each UAV is equipped with a look-down camera with FOV of 45° and hence has a partial view of the road as they fly around 10 m high only. They also have a 180° scanner sensor. Once the UAV detects a target through imaging processing, it is required to hover over and track the target. Figures 14(b)–14(d) shows three different situations when the position of Target 1 is different. In Fig. 14(b), the target is on the way of UAV1's first round flying path. So the UAV1 can detect it very quickly. In Fig. 14(c), UAV1 spends more time to look around. In Fig. 14(d), Target 1 is actually not on the road. Hence the UAV1 will search the whole road and try to locate the target.

## 5. Future Challenges and Remaining Work

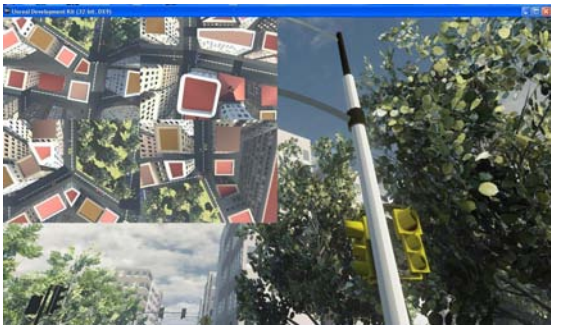
Although lots of research efforts have been made in addressing the issues in this area, fully implementable out-door

formation control: flying over buildings

formation control with OA: flying among buildings



formation control with constraints: fixed-wing UAVs



coverage control in an urban environment

Fig. 11. Formation control and coverage control.

robotic systems are seldom evidenced at current stage. Some of the major barriers obstructing the application of multi-robot networks are limited wireless communication bandwidth, unpredictable control interferences and low positioning accuracy for fast moving agents, etc. Great challenges remain and a lot more efforts are to be made in tackling these issues.

### 5.1. *Topology control*

Topology control should be incorporated into the collaborative control for efficient and robust communication, after a basic connectivity is maintained.

- *Connection Maintenance.*[72] In order to get an efficient connectivity maintenance algorithm, we can integrate a certain distributed topology control algorithm with necessary modifications, e.g., LINT (local information no topology), LILT (local information link-state topology) or CBTC (cone-based topology control),[116,117] into the search and exploration algorithm, which could be much less conservative than existing approaches. Note that the static topology control algorithms, such as local minimal spanning tree and R&M,[116,117] cannot be applied here due to the huge communication load and constructive

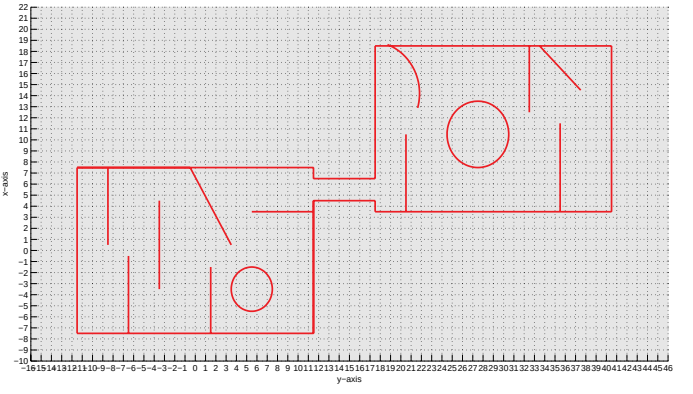
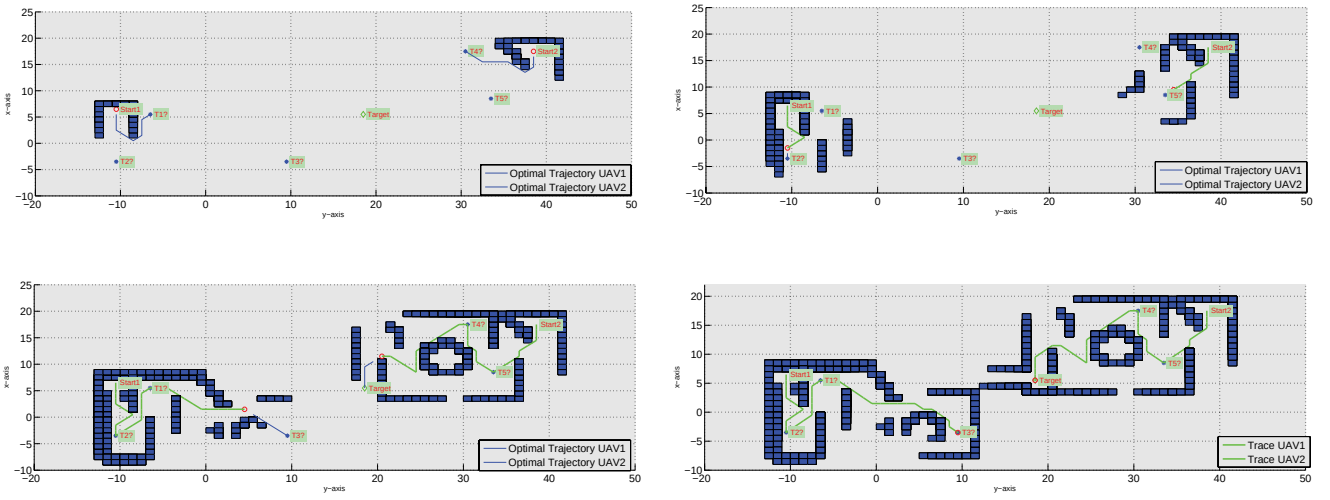
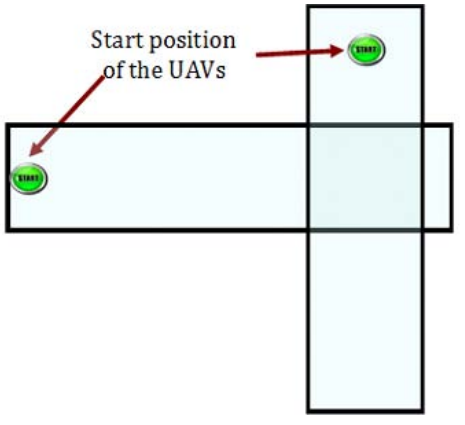



Fig. 12. UDK scenario and actual map.

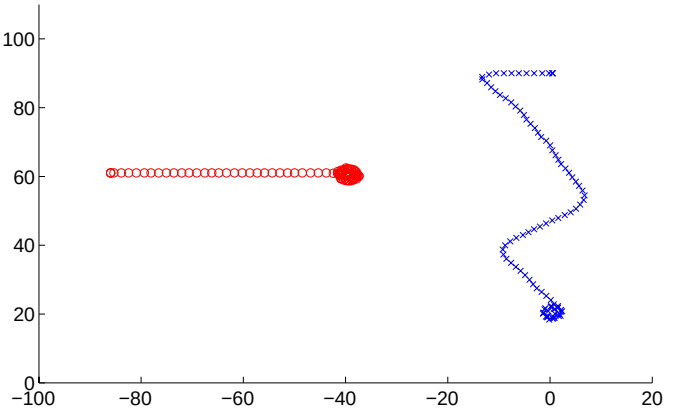



(a) (b)

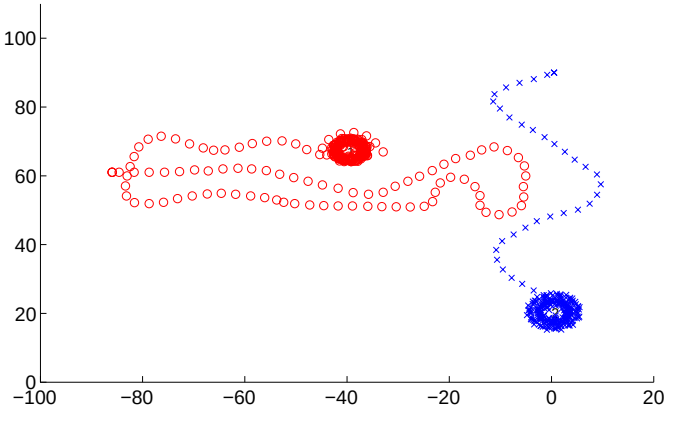
Fig. 13. The UAV trajectories and map at different stages.



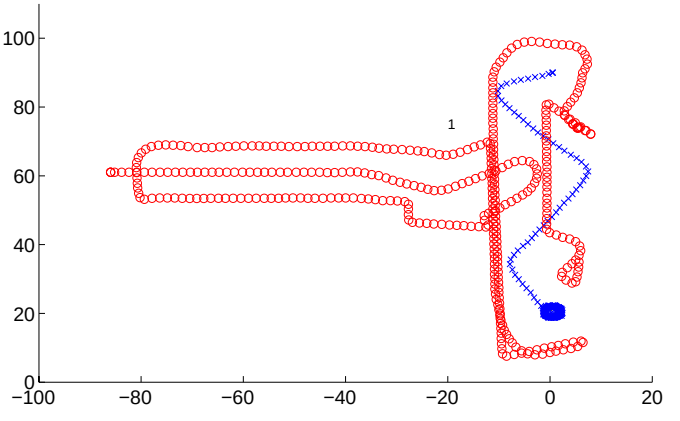

(a) Road map



(b) One round search



(c) Several rounds search



(d) Complete Search

Fig. 14. Road map and search trajectories.

inefficiency. Regarding switching topologies and other events (e.g., failure of UAVs, human interference), the discrete-event system approach[118] and triggered control technique can be applied. In addition, we can solve the so-called complete coverage problem by using a switched control mechanism. Regarding the real-time collision avoidance, the augmented Lagrange multiplier method may perform better than the penalty function approach. Furthermore, the topology control algorithms can be used to relax potential design conservatism with multi-hop consideration.

- *Efficient Routing*. It is reasonable to use a joint algorithmic and queueing approach to design the collaborative search algorithm to optimize the service time (e.g., response time and wait time).[119] It is not a simple extension of traveling salesman problem (TSP) for graph theory. Instead, it is a combination of queueing theory, combination algorithm and stochastic geometry.
- *K-connectivity and Robustness.* The communication robustness is a critical issue for robotic networks, while *K*-connectivity is a good approach to address the problem. As pointed out in Sec. 3.2, most cooperative control strategies lack the consideration of this issue, which, however, degrades their applicability in a real network. Different from the conventional topology control methods that satisfy the *K*-connectivity requirement for a stationary network, it is much helpful to develop a method that can be adapted to a network with switching topologies.

### 5.2. *3D Environment*

It is not easy to use one map to describe the 3D environment and targets, although multiple dimension probability may be useful for multiple target problems. Due to the effect of LOS (line-of-sight), the coverage problem under 3D building environments is significantly different from 2D case. Currently, a lot of methods have not considered the effect of LOS, which can in fact substantially influence the sensing performance. An instinctive approach for an agent is to fly around the buildings.[120,121] However, the search task would be quite challenging for coordination of multiple agents. For example, the agents located around an obstacle need to not only find the optimal position for detection, but also tune their views properly such that the shadow area blocked by the obstacle is minimized. In addition, the obstacle avoidance (OA) algorithm is more complex. In our view, OA should be developed as an independent mode running in a middleware at a relatively high frequency to the other outer-loop control algorithm. For a nonconvex region, existing methods generally lack of computational efficiency. A recent approach is based on Lloyd algorithm and TangentBug (a local path planner with obstacle avoidance behavior).[122] An extremum seeking scheme may be useful for unknown environments with an embedded terrain analysis algorithm.

### 5.3. *Communication cost saving*

Due to the limited wireless communication bandwidth, only the necessary data should be communicated and the "talk" frequency should be as low as possible. Hence, generally only updated information is sent to neighbors. However, because of the communication delay, the map information may not be correctly updated or estimated. Therefore, efficient and robust algorithm should be designed to satisfy this requirement.

### 5.4. *Concurrent control*

Most existing results assume that the individual systems are executed serially, i.e., sequentially with no overlap in time. Hence, no concurrent update exists. However, if concurrent updates with interleaving operations are allowed in an uncontrolled manner, some unexpected, undesirable results may occur. Some typical examples can be found in practical systems. For instance, the system A updates its state at time $k$, while using its neighbor B's state at time $k-1$ (the dirty/delay read problem). However, at that time, B's state actually changes, which can potentially affect the connectivity (the lost update problem). Most time-critical systems need to run transactions concurrently to meet their performance requirements. Thus, without concurrency control such systems can neither provide correct results nor maintain their state consistently.

### 5.5. *Time constraint*

The time issue is often indirectly studied by considering the traveling distance problem. Basically, we need to minimize the time cost for the search task. Without the full global information, the optimal solution seems to be impossible. Some results, e.g., Refs. 123–125, use the time as an objective function directly. It may be feasible to consider the multiple objective optimization. A relevant problem is the finite-time coverage problem, i.e., the convergence of the gradient flow is finite time.[126]

### 5.6. *Nonholonomic system*

Most existing results are based on holonomic systems, i.e., the number of independent control inputs is the same as that of the controlled independent state variables, although

many physical systems are actually nonholonomic. There are several methods to solve this problem including

- *Trajectory generation method.* One of the best references may be Ref. 127. This approach requires some way points first and then generates the corresponding robot trajectories along these points under the dynamics constraints of robot. It is relatively easy to be applied with limited capability.
- *Adaptive control.* The main idea is to find a proper Lyapunov function to fit the robot dynamics.[128] Although it is generally arduous to find such a Lyapunov function for a complex system, there are some patterns for Dubin-car-like systems. When considering the reaction to the environment, game-theoretic framework may be applied too.[129]
- *Linearization transformation with additional constraints.* This approach uses some nonlinear transformation to convert a nonholonomic system into a holonomic-like system with additional input/state constraints.

### 5.7. *Simultaneous localization and tracking*

In reality, a localization technique is often required to obtain the target's position. When the "global" positioning information is unavailable, even the robot should estimate its own position. The source used for localization may be vision, bearing/strength signal, etc. The question is how to incorporate the localization techniques into search and tracking problem seamlessly. Under the assumption of full connectivity/communication, it is possible to freely choose different formation of UAVs for better localization of the target. Another interesting problem is to localize the robots themselves by exchanging bearing/distance information (it would be difficult if only bearing information is available), thus they may form a certain shape during flying.

## 6. Conclusion

This paper presented a review on recent research developments in the field of cooperative control of robotic networks for environmental search and exploration. Four fundamental modules in the closed-loop control strategies were defined based on their functions which constitute the overall searching process. Then, different methods and algorithms were introduced and compared following the classification of the modules. Simulator and experiment implementations on our laboratory platform were demonstrated. Challenges and interesting directions for future research and development of robotic networks in this area were also highlighted.

## References


[1] Y. U. Cao, A. S. Fukunaga and A. Kahng, Cooperative mobile robotics: Antecedents and directions, *Auton. Robots* **4**(1) (1997) 7–27.

[2] T. Arai, E. Pagello and L. E. Parker, Editorial: Advances in multi-robot systems, *IEEE Trans. Robot. Autom.* **18**(5) (2002) 655–661.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, Wireless sensor networks: A survey, *Comput. Netw.* **38**(4) (2002) 393–422.

[4] D. Li, K. Wong, Y. Hu and A. Sayeed, Detection, classification and tracking of targets in distributed sensor networks, *IEEE Signal Process. Mag.* **19**(2) (2002) 17–29.

[5] J. Lin, W. Xiao, F. Lewis and L. Xie, Energy-efficient distributed adaptive multisensor scheduling for target tracking in wireless sensor networks, *IEEE Trans. Instrum. Meas.* **58**(6) (2009) 1886–1896.

[6] Y. Zhai, M. Yeary, J. Havlicek and G. Fan, A new centralized sensor fusion-tracking methodology based on particle filtering for power-aware systems, *IEEE Trans. Instrum. Meas.* **57**(10) (2008) 2377–2387.

[7] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, Energy-efficient signal classification in *ad hoc* wireless sensor networks, *IEEE Trans. Instrum. Meas.* **57**(1) (2008) 190–196.

[8] J. R. Riehl, G. E. Collins and J. P. Hespanha, Cooperative search by uav teams: A model predictive approach using dynamic graphs, *IEEE Trans. Aerosp. Electron. Syst.* **47**(4) (2011) 2637–2656.

[9] R. M. Murray, Recent research in cooperative control of multivehicle systems, *J. Dyn. Syst. Meas. Control* **129**(5) (2007), p. 571.

[10] L. E. Parker, Current state of the art in distributed autonomous mobile robotics, in *Distributed Autonomous Robotic Systems*, Vol. 4 (Springer, 2000), pp. 3–12.

[11] J. George and J. Sousa, Search strategies for multiple uav search and destroy missions, *J. Intell. Robot. Syst.* **61**(1) (2011) 355–367.

[12] S. K. Gan and S. Sukkarieh, Multi-uav target search using explicit decentralized gradient-based negotiation, *IEEE Int. Conf. Robotics and Automation (ICRA) (IEEE, 2011)*, pp. 751–756.

[13] J. Happe and J. Berger, Couav: A multi-uav cooperative search path planning simulation environment, in *Proc. Summer Computer Simulation Conf., SCSC '10* (Society for Computer Simulation International, San Diego, CA, USA, 2010), pp. 86–93.

[14] R. W. Beard and T. W. McLain, Multiple UAV cooperative search under collision avoidance and limited range communication constraints, in *Proc. 42nd IEEE Conf. Decision and Control*, Vol. 1 *(IEEE, 2003)*, pp. 25–30.

[15] M. Bernard, K. Kondak, I. Maza and A. Ollero, Autonomous transportation and deployment with aerial robots for search and rescue missions, *J. Field Robot.* **28**(6) (2011) 914–931.

[16] S. J. Benkoski, M. G. Monticino and J. R. Weisinger, A survey of the search theory literature, *Nav. Res. Logist.* **38**(4) (1991) 469–494.

[17] L. D. Stone, Search theory: A mathematical theory for finding lost objects, *Math. Mag.* **50**(5) (1977) 248–256.

[18] K. Trummel and J. Weisinger, Technical note the complexity of the optimal searcher path problem, *Oper. Res.* **34**(2) (1986) 324–327.

[19] A. Howard, M. J. Mataric and G. S. Sukhatme, Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem, in *Proc. 6th Int. Symp. Distributed Autonomous Robotics Systems*, Citeseer (2002), pp. 299–308.

[20] D. G. Rajnarayan and D. Ghose, Multiple agent team theoretic decision-making for searching unknown environments, in *Proc. 42nd IEEE Conf. Decision and Control*, Vol. 3 (IEEE, 2003), pp. 2543–2548.

[21] D. Enns, D. Bugajski and S. Pratt, Guidance and control for cooperative search, in *Proc. American Control Conf.*, Vol. 3 (2002), pp. 1923–1929.

[22] P. Vincent and I. Rubin, A framework and analysis for cooperative search using uav swarms, in *Proc. ACM Symp. Applied Computing* (ACM, 2004), pp. 79–86.

[23] Y. Altshuler, V. Yanovsky, I. A. Wagner and A. M. Bruckstein, Efficient cooperative search of smart targets using UAV swarms, *Robotica* **26**(4) (2008) 551–557.

[24] Y. Yang, A. A. Minai and M. M. Polycarpou, Analysis of opportunistic method for cooperative search by mobile agents, in *Proc. 41st IEEE Conf. Decision and Control*, Vol. 1 (IEEE, 2002), pp. 576–577.

[25] Y. Yang, A. Minai and M. Polycarpou, Decentralized cooperative search in UAVs using opportunistic learning, in *Proc. AIAA Guidance, Navigation, and Control Conf. Exhibit* (2002).

[26] Y. Jin, A. Minai and M. Polycarpou, Cooperative real-time search and task allocation in UAV teams, in *Proc. IEEE Conf. Decision and Control*, Vol. 1 (2003), pp. 7–12.

[27] P. Ogren, E. Fiorelli and N. Leonard, Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment, *IEEE Trans. Autom. Control* **49**(8) (2004) 1292–1302.

[28] S. Martínez and F. Bullo, Optimal sensor placement and motion coordination for target tracking, *Automatica* **42**(4) (2006) 661–668.

[29] K. Lynch, I. Schwartz, P. Y. and R. Freeman, Decentralized environmental modeling by mobile sensor networks, *IEEE Trans. Robot.* **24**(3) (2008) 710–724.

[30] K. Guruprasad and D. Ghose, Automated multi-agent search using centroidal voronoi configuration, *IEEE Trans. Autom. Sci. Eng.* **8**(2) (2011) 420–423.

[31] K. Huguenin and J. Rendas, An architecture for distributed collaborative adaptive sensing, in *Proc. Int. Symp. Unmanned Untethered Submersibles Technology* (2007).

[32] M. Schwager, D. Rus and J. Slotine, Decentralized, adaptive coverage control for networked robots, *Int. J. Robot. Res.* **28**(3) (2009) 357–375.

[33] J. Choi, S. Oh and R. Horowitz, Distributed learning and cooperative control for multi-agent systems, *Automatica* **45**(12) (2009) 2802–2814.

[34] J. Cortés, Distributed kriged Kalman filter for spatial estimation, *IEEE Trans. Autom. Control* **54**(12) (2009) 2816–2827.

[35] P. Millet, D. Casbeer, T. Mercker and J. Bishop, Multi-agent decentralized search of a probability map with communication constraints, in *Proc. AIAA Guidance, Navigation, and Control Conf.*, Toronto, Ontario Canada, August 2010, pp. AIAA2010–8424.

[36] M. Zhong and C. G. Cassandras, Distributed coverage control and data collection with mobile sensor networks, *IEEE Trans. Autom. Control* **56**(10) (2011) 2445–2455.

[37] J. Hu, L. Xie, K.-Y. Lum and J. Xu, Multi-agent information fusion and cooperative control in target search, *IEEE Trans. Control Syst. Technol.* (2012), doi: 10.1109/TCST.2012.2198650.

[38] J. Tisdale, A. Ryan, Z. Kim, D. Tornqvist and J. K. Hedrick, A multiple UAV system for vision-based search and localization, *American Control Conf.* (IEEE, 2008), pp. 1985–1990.

[39] K. Guruprasad and D. Ghose, Performance of a class of multi-robot deploy and search strategies based on centroidal voronoi configurations, *Int. J. Syst. Sci.* **44**(4) (2013) 680–699.

[40] M. Flint, E. Fernandez-Gaucherand and M. Polycarpou, Cooperative control for UAV's searching risky environments for targets, in *Proc. 42nd IEEE Conf. Decision and Control*, Vol. 4 (IEEE, 2003), pp. 3567–3572.

[41] L. Bertuccelli and J. How, UAV search for dynamic targets with uncertain motion models, in *Proc. IEEE Conf. Decision and Control* (2007), pp. 5941–5946.

[42] J. Berger and J. Happe, Co-evolutionary search path planning under constrained information-sharing for a cooperative unmanned aerial vehicle team, in *Proc. IEEE Congress on Evolutionary Computation* (2010), pp. 1–8.

[43] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma and S. Lim, A survey and comparison of peer-to-peer overlay network schemes, *IEEE Commun. Surveys Tut.* **7**(2) (2005) 72–93.

[44] K. Akkaya and M. Younis, A survey on routing protocols for wireless sensor networks, *Ad hoc Netw.* **3**(3) (2005) 325–349.

[45] C. E. Jones, K. M. Sivalingam, P. Agrawal and J. C. Chen, A survey of energy efficient network protocols for wireless networks, *Wireless Netw.* **7**(4) (2001) 343–358.

[46] F. Bullo, J. Cortés and S. Martinez, *Distributed Control of Robotic Networks* (Princeton University Press, 2009).

[47] C. Chong, S. Mori and K. Chang, Adaptive distributed estimation, in *Proc. IEEE Conf. Decision and Control* (1987), pp. 2233–2238.

[48] K. Chang, R. Saha and Y. Bar-Shalom, On optimal track-to-track fusion, *IEEE Trans. Aerosp. Electron. Syst.* **33**(4) (1997) 1271–1276.

[49] X. Li, K. Zhang, J. Zhao and Y. Zhu, Optimal linear estimation fusion. part v. relationships, in *Proc. Fifth Int. Conf. Information Fusion*, Vol. 1 (2002), pp. 497–504.

[50] S. Mori, W. Barker, C. Chong and K. Chang, Track association and track fusion with nondeterministic target dynamics, *IEEE Trans. Aerosp. Electron. Syst.* **38**(2) (2002) 659–668.

[51] Y. Zhu, E. Song, Z. J. and Z. You, Optimal dimensionality reduction of sensor data in multisensor estimation fusion, *IEEE Trans. Signal Process.* **53**(5) (2005) 1631–1639.

[52] S. Marano, V. Matta and P. Willett, Distributed estimation in large wireless sensor networks via a locally optimum approach, *IEEE Trans. Signal Process.* **56**(2) (2008) 748–756.

[53] I. Schizas, G. Giannakis and Z. Luo, Distributed estimation using reduced-dimensionality sensor observations, *IEEE Trans. Signal Process.* **55**(8) (2007) 4284–4299.

[54] F. Cattivelli, C. Lopes and A. Sayed, Diffusion recursive least-squares for distributed estimation over adaptive networks, *IEEE Trans. Signal Process.* **56**(5) (2008) 1865–1877.

[55] I. Suzuki and M. Yamashita, Distributed anonymous mobile robots: Formation of geometric patterns, *SIAM J. Comput.* **28** (1999), p. 1347.

[56] C. Belta and V. Kumar, Abstraction and control for groups of robots, *IEEE Trans. Robot.* **20**(5) (2004) 865–875.

[57] E. Justh and P. Krishnaprasad, Equilibria and steering laws for planar formations* 1, *Syst. Contr. Lett.* **52**(1) (2004) 25–38.

[58] A. Jadbabaie, J. Lin and A. Morse, Coordination of groups of mobile autonomous agents using nearest neighbor rules, *IEEE Trans. Autom. Control* **48**(6) (2003) 988–1001.

[59] H. Tanner, A. Jadbabaie and G. Pappas, Flocking in fixed and switching networks, *IEEE Trans. Autom. Control* **52**(5) (2007) 863–868.

[60] R. Olfati-Saber, Flocking for multi-agent dynamic systems: Algorithms and theory, *IEEE Trans. Autom. Control* **51**(3) (2006) 401–420.

[61] V. Gazi and K. Passino, Stability analysis of social foraging swarms: Combined effects of attractant/repellent profiles, in *Proc. IEEE Conf. Decision and Control*, Vol. 3 (2002), pp. 2848–2853.

[62] J. Ĉortés, S. Martinez, T. Karatas and F. Bullo, Coverage control for mobile sensing networks, *IEEE Trans. Robot. Autom.* **20**(2) (2004) 243–255.

[63] J. Cortés, S. Martínez and F. Bullo, Spatially-distributed coverage optimization and control with limited-range interactions, *Control, Optim. Calc. Var.* **11**(4) (2005) 691–719.

[64] J. Cortés and F. Bullo, Coordination and geometric optimization via distributed dynamical systems, *SIAM J. Contr. Optim.* **44**(5) (2005) 1543–1574.

[65] E. Frazzoli and F. Bullo, Decentralized algorithms for vehicle routing in a stochastic time-varying environment, in *Proc. 43rd IEEE Conf. Decision and Control*, Vol. 4 (2004).

[66] H. Ando, Y. Oasa, I. Suzuki and M. Yamashita, Distributed memoryless point convergence algorithm for mobilerobots with limited visibility, *IEEE Trans. Robot. Autom.* **15**(5) (1999) 818−828.

[67] J. Lin, A. Morse and B. Anderson, The multi-agent rendezvous problem, in *Proc. 42nd IEEE Conference on Decision and Control*, Vol. 2 (2003), pp. 1508−1513.

[68] Z. Lin, M. Broucke and B. Francis, Local control strategies for groups of mobile autonomous agents, *IEEE Trans. Autom. Control* **49**(4) (2004) 622−629.

[69] J. Çortés, S. Martinez and F. Bullo, Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions, *IEEE Trans. Autom. Control* **51**(8) (2006) 1289−1298.

[70] C. Goerzen, Z. Kong and B. Mettler, A survey of motion planning algorithms from the perspective of autonomous uav guidance, *J. Intell. Robot. Syst.* **57** (2010) 65100.

[71] J. Giesbrecht, Global path planning for unmanned ground vehicles, Tech. Rep., Defence R&D Canada, Suffield, December, 2004.

[72] J. Xu, J. Hu, L. Xie and K. Y. Lum, Distributed coverage control under generalized locational optimization framework, *The 31th Chinese Control Conf.*, Hefei, China, July 2012.

[73] S. Spires and S. Goldsmith, Exhaustive geographic search with mobile robots along pace-filling curves, in *Collective Robotics*, eds. A. Drogoul, M. Tambe and T. Fukuda, Lecture Notes in Computer Science, Vol. 1456 (Springer, Berlin Heidelberg, 1998), pp. 1−12.

[74] V. Ablavsky and M. Snorrason, Optimal search for a moving target: A geometric approach, *AIAA Guidance, Navigation, and Control Conf. Exhibit*, Citeseer (2000).

[75] Z. Tang and U. Ozguner, On non-escape search for a moving target by multiple mobile sensor agents, *American Control Conf.* (IEEE, 2006), p. 6.

[76] M. Schwager, B. Julian, M. Angermann and D. Rus, Eyes in the sky: Decentralized control for the deployment of robotic camera networks, in *Proc. IEEE* **99**(9) (2011) 1541−1561.

[77] V-REP, Virtual robot experimentation platform, https://www.v-rep.eu/.

[78] LAAS-CNRS, MORSE, https://www.openrobots.org/wiki/morse/.

[79] D. Jung, OpenSim, https://opensimulator. sourceforge.net.

[80] Delta3D (2011), www.delta3d.org.

[81] P. McDowell *et al.* Delta3d: A complete open source game and simulation engine for building military training systems, *J. Def. Model. Simul.: Appl., Meth., Technol.* **3** (2006) 143−154.

[82] A. H. Nathan Koenig, Design and use paradigms for gazebo, an open-source multi-robot simulator, *IEEE/RSJ Intelligent Robots and Systems*, Sendai, Japan (2004), pp. 2149−2154.

[83] J. Xu, L. Xie, T. G. Toh and Y. K. Toh, Hnmsim: A 3d multi-purpose hybrid networked multi-agent simulator, *The 31th Chinese Control Conf.*, Hefei, China, July 2012.

[84] C. Aldrich, *The Complete Guide to Simulations and Serious Games* (Pfeiffer, 2009).

[85] Bisimulations, VBS2, 2nd edn. (2010), https://www.bistudio.com.

[86] BBN and Technologies, Darwars, https://ww34.ambush.darwars.net.

[87] J. Wang and S. Balakirsky, USARSim, NIST, University of Pittsburgh, Pittsburgh, PA, USA, 3.1.3 edn.

[88] U. Unreal Technology (2011), www.unreal.com, www.udk.com, www.unrealengine.com/.

[89] M. Lewis, J. Wang and S. Hughes, Usarsim: Simulation for the study of human-robot interaction, *J. Cognit. Eng. Decis. Making* **1**(1) (2007) 98−120.

[90] M. N. R. Smith, M. Shaker, S. Yue and T. Duckett, A thin middleware for simulated robot vision applications, in *Proc. Int. Conf. Computer Science and Information Technology*, Chengdu, China (2011).

[91] R. rescue virtual robot competition, RoboCupRescue Simulation League, www.robocuprescue.org.

[92] T. Schmits and A. Visser, An Omnidirectional Camera Simulation for the USARSim World, in *RoboCup 2008: Robot Soccer World Cup XII*, Lecture Notes on Artificial Intelligence series, Vol. 5339 (Springer, Berlin Heidelberg New York, 2009), pp. 296−307.

[93] VMAC, Virtual Manufacturing Automation Competition, https://vma-competition.com/.

[94] A. Visser, N. Dijkshoorn, M. van der Veen and R. Jurriaans, Closing the gap between simulation and reality in the sensor and motion models of an autonomous ar.drone, in *Proc. Int. Micro Air Vehicle Conference and Flight Competition (IMAV11)*, Harde, The Netherlands, September, 2011.

[95] P. Robbel, D. Demirdjian and C. Breazeal, Simultaneous localization and mapping with people, in *Proc. Advanced Reasoning with Depth Cameras Workshop, Robotics Science and Systems Conf.*, Ca, USA, June 2011.

[96] M. A. Sharbafi, S. Taleghani, E. Esmaeili and O. A. A. T. Haghighat, Ice matching, a novel approach for localization problem, in *Proc. IEEE Int. Conf. Control, Automation and Systems*, October 2010, pp. 1904−1907.

[97] Autodesk, *3DS Max*. 2010 edn. (2010), http://usa.autodesk.com/3ds-max/.

[98] G. Johnson, *LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control* (McGraw-Hill School Education Group, 1997).

[99] M. A. Hsieh *et al.*, MATLAB USARSim Toolbox (2010), https://robotics.mem.drexel.edu/USAR/.

[100] GAUI, 330X-S quad-flyer instruction manual, http://www.gaui.com.tw/html/download.asp.

[101] J. Xu, L. Xie, N. Khanna and W. Chee, Cooperative control in hnmsim — a 3d hybrid networked mas simulator, *The 12th Int. Conf. Control, Automation, Robotics and Vision*, Guangzhou, China, December 2012.

[102] P. Corke, Machine Vision Toolbox, https://www.petercorke.com/Machine-Vision-Toolbox.html.

[103] P. Corker, *Robotics, Vision and Control: Fundanmental Algorithms in Matlab* (Springer, 2011).

[104] W. Garage, Intel and etc., OpenCV, https://code.opencv.org.

[105] ROS, Robot Operating System, https://www.ros.org.

[106] R. Rathnam, M. Pfingsthorn and A. Birk, Incorporating large scale ssrr scenarios into high fidelity simulator usarsim, in *Proc. Int. Workshop on Safety, Security and Resuce Robotics* (2009).

[107] T. Brent, S. Carlson and J. Dutko, USARSim World Generator tool, June 2011, https://usarsim.sourceforge.net/wiki/index.php/WorldGenerator.

[108] G. Sketchup, 2011, sketchup.google.com.

[109] CityEngine, *Esri*, https:// www.esri.com/software/cityengine.

[110] Gamr7, Urban Pad, https://www.gamr7.com.

[111] M. Pfingsthorn, Robocup virtual robots — wireless simulation server documetation, Tech. Rep., Jacobs University, Germany, October 2008.

[112] A. Varga, The OMNeT++ discrete event simulation system, in *Proc. European Simulation Multiconf.*, Vol. 9 (2001).

[113] A. Cervin and M. O. D. Henriksson, TrueTime 2.0 beta 5 — Reference Manual, Department of Automatic Control, Lund University, Sweden, June 2010.

[114] M. Pohjola and S. Nethi, PiccSIM Manual, Helsinki University of Technology, http://autsys.aalto.fi/en/attach/Control/PiccSIM Manual.pdf, 0.81 edn., June 2009.

[115] K. You and L. Xie, Network topology and communication data rate for consensusability of discrete-time multi-agent systems, *IEEE Trans. Autom. Control* **56**(10) (2011) 2262−2275.

[116] P. Santi, *Topology Control in Wireless Ad Hoc and Sensor Networks* (John Wiley & Sons, England, 2006).

[117] M. A. Labrador and P. M. Wightman, *Topology Control in Wireless Sensor Networks* (Springer Science + Bussiness Media, 2009).

[118] R. Su, J. H. van Schuppen and J. E. Rooda, Aggregative synthesis of distributed supervisors based on automaton abstraction, *IEEE Trans. Autom. Control* **55**(7) (2010) 1627−1640.

[119] F. Bullo, E. Frazzoli, M. Pavone, K. Savla and L. Smith, Dynamic vehicle routing for robotic systems, *in Proc. IEEE* **99**(9) (2011) 1482−1504.

[120] P. Cheng and V. Kumar, Constant-factor optimal control for 3-d coverage by uavs, GRASP Laboratory, University of Pennsylvania, USA, March 2008.

[121] P. Cheng, J. Keller and V. Kumar, Time-optimal uav trajectory planning for 3d urban structure coverage, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Nice, France, September 2008, pp. 2750−2757.

[122] A. Breitenmoser, M. Schwager, J. C. Metzger, R. Siegwart and D. Rus, Voronoi coverage of non-convex environments with a group of networked robots, in *Proc. Int. Conf. Robot. Automation (ICRA 10)*, May 2010, pp. 4982−4989.

[123] E. Xargay, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, N. Hovakimyan and C. Cao, Time-critical cooperative control of multiple autonomous vehicles, *IEEE Control Syst. Mag.* **32**(5) (2012) 49−73.

[124] A. Dippold, L. Ma and N. Hovakimyan, Time-critical coordination of multiple vehicles with uni-directional communication constraints, in *Proc. American Control Conf.*, Baltimore MD, USA, June 2010, pp. 1617−1622.

[125] A. Ahmadzadeh, J. Keller, G. J. Pappas, A. Jadbabaie and V. Kumar, An optimization-based approach to time critical cooperative surveillance and coverage with unmanned aerial vehicles, *The 10th Int. Symp. Experimental Robotics* (2008), pp. 491−500.

[126] J. Cortés, Finite-time convergent gradient flows with applications to network consensus, *Automatica* **42** (2006) 1993−2000.

[127] D. Mellinger, Trajectory generation and control for quadrotors, Ph.D. thesis, University of Pennsylvania (2012).

[128] J.-M. Luna, R. Fierro, C. Abdallah and J. Wood, An adaptive coverage control algorithm for deployment of nonholonomic mobile sensors, in *Proc. Conf. Decision and Control*, Atlanta, GA, USA, December 2010, pp. 1250−1255.

[129] H.-B. Dürr, M. S. Sankovií and K. H. Johansson, Distributed positioning of autonoumous mobile sensors with application to coverage control, in *Proc. American Control Conf.*, San Francisco, CA, USA, July 2011, pp. 4822−4827.