Hindawi
Scientific Programming
Volume 2020, Article ID 8530763, 14 pages
https://doi.org/10.1155/2020/8530763

Research Article

Online Path Generation and Navigation for Swarms of UAVs

Adnan Ashraf,[1] **Amin Majd**,[1] **and Elena Troubitsyna**[1,2]

[1]*Faculty of Science and Engineering, Åbo Akademi University, Turku, Finland*
[2]*Department of Theoretical Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden*

Correspondence should be addressed to Adnan Ashraf; aashraf@abo.fi

Received 17 January 2019; Revised 25 October 2019; Accepted 19 December 2019; Published 11 January 2020

Academic Editor: Can Özturan

With the growing popularity of unmanned aerial vehicles (UAVs) for consumer applications, the number of accidents involving UAVs is also increasing rapidly. Therefore, motion safety of UAVs has become a prime concern for UAV operators. For a swarm of UAVs, a safe operation cannot be guaranteed without preventing the UAVs from colliding with one another and with static and dynamically appearing, moving obstacles in the flying zone. In this paper, we present an online, collision-free path generation and navigation system for swarms of UAVs. The proposed system uses geographical locations of the UAVs and of the successfully detected, static, and moving obstacles to predict and avoid the following: (1) UAV-to-UAV collisions, (2) UAV-to-static-obstacle collisions, and (3) UAV-to-moving-obstacle collisions. Our collision prediction approach leverages efficient runtime monitoring and complex event processing (CEP) to make timely predictions. A distinctive feature of the proposed system is its ability to foresee potential collisions and proactively find best ways to avoid predicted collisions in order to ensure safety of the entire swarm. We also present a simulation-based implementation of the proposed system along with an experimental evaluation involving a series of experiments and compare our results with the results of four existing approaches. The results show that the proposed system successfully predicts and avoids all three kinds of collisions in an online manner. Moreover, it generates safe and efficient UAV routes, efficiently scales to large-sized problem instances, and is suitable for cluttered flying zones and for scenarios involving high risks of UAV collisions.

1. Introduction

An unmanned aerial vehicle (UAV) or drone is a semiautonomous aircraft that can be controlled and operated remotely by using a computer along with a radio link [1]. UAVs can be classified into different types based on their design, size, and flying mechanism. Among the existing types, the quadrotors or quadrocopters are particularly popular because of their simple design, small size, low cost, greater maneuverability, and the ability to hover in place. A quadrotor uses two pairs of identical, vertically oriented propellers of which one pair spins clockwise and the other spins counterclockwise. Commercially available quadrotors are increasingly been used in a variety of applications such as monitoring and surveillance, search and rescue operations, geographic mapping, photography and filming, wildlife research and management, media coverage of public events, remote sensing for agricultural applications, and aerial package delivery [2–5]. Efficient and scalable solutions for these applications require an online path generation and navigation system for multiple UAVs.

UAVs are becoming increasingly popular. In the United States, the Federal Aviation Administration (FAA) has projected that the number of small hobbyist drones is set to increase from an estimated 1.1 million in 2017 to 2.4 million by 2022 (https://www.faa.gov/news/updates/?newsId=89870). With the growing popularity and use of UAVs for consumer applications, the number of accidents involving drones is also increasing dramatically. The FAA receives more than 100 reports every month of unauthorized and potentially hazardous UAV activity reported by pilots, citizens, and law enforcement (https://www.faa.gov/uas/resources/public_records/uas_sightings_report/). In a recent incident (https://www.bbc.com/news/uk-england-sussex-46623754) that took place in the United Kingdom, the runway at the London Gatwick Airport was shut down for more than a day because two drones were spotted flying repeatedly over the airfield. The disruption affected about

110,000 passengers on 760 flights as no flights were able to take off or land. Such incidents on one hand show the importance of educational and training programmes for drone operators and stricter legislation for offenders, but on the other hand, they also motivate the need for a collision-free path generation and navigation system for UAVs. Ensuring a hazard-free, safe UAV flight is also equally important for indoor applications. Therefore, motion safety of UAVs has become a prime concern for UAV operators. It refers to the ability of the UAVs to detect and avoid collisions with static and moving obstacles in the environment. The static obstacles include buildings, trees, and other similar stationary items, while movable items (for example, birds) are considered as moving obstacles.

Some of the commercially available quadrotors are capable of detecting and avoiding some obstacles. For example, DJI's Phantom 4 Pro (https://www.dji.com/phantom-4-pro) uses five-directional sensors to provide obstacle detection or sensing in five directions with a front and rear sensing range of up to 30 meters and up to 7 meters for left and right sides. However, its obstacle avoidance mechanism does not work in all kinds of scenarios. In this work, we assume that each UAV is equipped with an adequate obstacle detection capability and can successfully detect all static and dynamically appearing, moving obstacles in its surroundings. Therefore, the emphasis of this work is not on obstacle detection. Instead, we focus on collision prediction and avoidance.

Multiple UAVs working in a cooperative manner can be used to provide powerful capabilities that a single UAV cannot offer [3]. Therefore, for larger and highly complex applications and tasks which are either beyond the capabilities of a single UAV or cannot be performed efficiently if only a single UAV is used, multiple UAVs can be used together in the form of a swarm or a fleet. In such scenarios, a safe operation cannot be guaranteed without preventing the UAVs from colliding with one another and with static and dynamically appearing, moving obstacles in the flying zone. Therefore, in the context of UAV swarms, ensuring motion safety entails devising and implementing an online motion path planning, coordination, and navigation system for multiple UAVs with an integrated support for collision prediction and avoidance.

The problem of motion safety of UAVs is currently attracting significant research attention. Some comprehensive literature reviews on motion planning algorithms for UAVs can be found in [6, 7]. The main focus of these approaches is on an offline motion planning phase to plan and produce UAV paths or trajectories before the start of the mission. Augugliaro et al. [8] also presented a planned approach that generates feasible paths ahead of time. LaValle [9] and Karaman and Frazzoli [10] presented sampling-based path planning algorithms. Silva Arantes et al. [11] proposed a path planning approach for critical situations requiring an emergency landing of a UAV. Dong et al. [3] presented a software platform for cooperative control of multiple UAVs. Bürkle et al. [12] proposed a multiagent system architecture for team collaboration in a swarm of drones. Ivanovas et al. [4] proposed an obstacle detection approach for UAVs. de Souza [13] and de Souza and Endler [14] presented an approach for movement coordination of swarms of drones using smart phones and mobile communication networks. Their work focuses on the internal communication of the swarm and does not provide a solution for collision-free path generation.

In this paper, we present an online, collision-free path generation and navigation system for swarms of UAVs. The proposed system uses geographical locations of the UAVs and of the successfully detected, static and dynamically appearing, moving obstacles to predict and avoid the following: (1) UAV-to-UAV collisions, (2) UAV-to-static-obstacle collisions, and (3) UAV-to-moving-obstacle collisions. It comprises three main components: (1) a complex event processing (CEP) and collision prediction module, (2) a mutually exclusive locking mechanism, and (3) a collision avoidance mechanism. The CEP and collision prediction module leverages efficient runtime monitoring and CEP to make timely predictions. The mutually exclusive locking mechanism prevents multiple UAVs from attempting to fly to the same location at the same time. The collision avoidance mechanism tries to find best ways to prevent the UAVs from colliding into one another with the successfully detected static and moving obstacles in the flying zone. A distinctive feature of the proposed system is its ability to foresee potential collisions and proactively find best ways to avoid the predicted collisions in order to ensure safety of the entire swarm. In contrast to the existing works [3, 4, 6–16], our proposed system does not depend on a planning phase and produces efficient, collision-free paths in an online manner. We focus on collision prediction and avoidance and online path generation and navigation for swarms of UAVs.

We also present a simulation-based implementation of the proposed system along with an experimental evaluation involving a series of experiments and compare our results with the results of four existing approaches [9–11, 17]. The results show that the proposed system successfully predicts and avoids all three kinds of collisions in an online manner. Moreover, it generates safe and efficient UAV routes, efficiently scales to large-sized problem instances, and is suitable for cluttered flying zones and for scenarios involving high risks of UAV collisions. Our proposed navigation system, its implementation, experiments, and results are not based on or limited to a particular application of UAV swarms. Instead, they are generic enough to be applicable to a wide range of applications. The work presented in this paper extends our preliminary approach and results reported in [2].

We proceed as follows: Section 2 sets up the terminology and context. The proposed online, collision-free path generation and navigation system for UAV swarms is presented in Section 3. In Section 4, we illustrate the main steps of our proposed approach on a small example. Section 5 presents some important implementation details along with the experimental evaluation. Section 6 reviews important related works. Finally, we present our conclusions in Section 7.

2. Preliminaries

The proposed system not only provides support for online collision prediction and avoidance, it also generates

complete routes for all UAVs in the swarm. Unlike traditional motion path planning approaches that require that all obstacles and their precise locations must be known before the start of the mission, the proposed approach does not assume any a priori knowledge of the obstacles. In other words, we assume that the terrain of the flying zone is not known beforehand. Therefore, the proposed system does not make any assumptions on the number and locations of the static and dynamically appearing, moving obstacles. It does not require a preliminary, offline motion planning phase to produce efficient routes for the UAVs. In our approach, the drones take off from their start locations and fly uninterruptedly towards their destinations until the proposed system predicts a collision and triggers our collision avoidance mechanism to prevent the predicted collision. Since the proposed system uses geographical locations of the UAVs to generate their paths and predict and avoid collisions, it requires correct and precise location information of all UAVs in the fleet. Imprecise and incorrect information can result in longer paths, and in the worst case, some UAVs can collide with other UAVs or with some static or moving obstacles.

Let the mission flying zone be represented by a finite set of locations $\text{AREA} = \{l_1, l_2, l_3, \ldots, l_M\}$, where each location l_i is represented as a point in a three-dimensional space (x, y, z). In an outdoor mission, the dimensions x, y, and z may correspond to latitude, longitude, and altitude or elevation, respectively. To ensure a suitable formation of the swarm, it is assumed that the distance between any two consecutive locations in AREA is less than or equal to the sensing range sen_r of the UAVs and greater than or equal to the safe distance dis_s for the UAVs. For example, the front and rear sensing range sen_r of Phantom 4 Pro UAV is up to 30 meters. Therefore, if the swarm comprises Phantom 4 Pro UAVs, the maximum distance between any two consecutive locations $l_i, l_j \in \text{AREA} | i \neq j$ should be less than or equal to 30 meters. The safe distance dis_s for UAVs depends on their maximum speed Sp, obstacle detection and processing time Pt, and wireless communication latency Cl [13]. For example, if two UAVs are found heading towards each other at a maximum speed Sp of 5 meter per second each and with an obstacle detection and processing time Pt of 0.5 seconds and a wireless communication latency Cl of 0.2 seconds, the safe distance dis_s can be estimated as

$$\text{dis}_s = 2 \cdot \text{Sp}\,(2 \cdot \text{Cl} + \text{Pt}), \tag{1}$$

which yields 9 meters. Therefore, in this example, the minimum distance between any two consecutive locations $l_i, l_j \in \text{AREA} \,|\, i \neq j$ should be greater than or equal to 9 meters. As a simplification to the problem, we assume that all consecutive locations in AREA are a uniform, fixed distance apart from one another denoted as dis such that $\text{dis}_s \leq \text{dis} \leq \text{sen}_r$. Hence, the flying zone AREA can be viewed as a three-dimensional grid. This simplification allows faster generation, comparison, and evaluation of solutions or UAV paths. For clarity, important terminology and notation used in this paper are summarized in Table 1.

Table 1: Terminology and notation.

Notation	Description
AREA	Three-dimensional flying zone
Cl	Wireless communication latency
dis	Distance between two consecutive UAVs
dis_s	Safe distance for the UAVs
SWARM	Swarm of drones
l_{fin}	Final or destination location of a UAV
l_i	A location in AREA
l_{in}	Initial or start location of a UAV
MOV_OBS	Set of moving obstacles
Pt	Obstacle detection and processing time
route_i	A UAV route
sen_r	Sensing range of the UAVs
Sp	Maximum speed of the UAVs
STA_OBS	Set of static obstacles

Furthermore, let $\text{SWARM} = \{d_1, d_2, \ldots, d_N\}$ be a set of drones or UAVs in the swarm. The static obstacles are denoted as $\text{STA\_OBS} = \{\text{so}_1, \text{so}_2, \ldots, \text{so}_O\}$. Similarly the dynamically appearing, moving obstacles are represented by the set $\text{MOV\_OBS} = \{\text{mo}_1, \text{mo}_2, \ldots, \text{mo}_P\}$. Each drone occupies a certain location in AREA. The drones take off from their start locations and fly towards their destination locations. A drone route or path is a sequence of locations from drone's start location to drone's destination location. For a drone d_i, $\text{route}_i = \langle l_{\text{in}}, \ldots, l_{\text{fin}} \rangle$ such that $\text{ran}\,(\text{route}_i) \subseteq \text{AREA}$ and where l_{in} is the initial or start location and l_{fin} is the final or destination location of d_i. Similarly, each static and moving obstacle occupies a certain location in AREA. Moreover, the moving obstacles keep on moving arbitrarily until they leave the flying zone.

We formulate three basic safety requirements (SRs) for a swarm of drones:

SR1: $\forall d_i \in \text{SWARM}, \forall \text{so}_j \in \text{STA\_OBS}$, d_i does not collide with so_j

SR2: $\forall d_i, d_j \in \text{SWARM} | i \neq j$, d_i and d_j do not collide with each other

SR3: $\forall d_i \in \text{SWARM}, \forall \text{mo}_j \in \text{MOV\_OBS}$, d_i does not collide with mo_j

Since the proposed system does not assume any a priori knowledge on the numbers and locations of the static and moving obstacles and does not depend on a preliminary, offline motion planning phase, none of the SRs can be verified before the start of the mission. For *SR1* which concerns static obstacles, it is necessary that the drones do not fly into a location where a static obstacle is situated. Our proposed system helps the drones to avoid all successfully detected static obstacles in an online manner by providing efficient collision prediction and collision avoidance mechanisms. Similarly, for *SR2* which concerns collisions with other drones, it is required that at any given time t, each location is occupied by at most one drone. The proposed system stops the drones from flying into other drones in the vicinity. The proposed mutually exclusive locking and collision avoidance mechanisms prevent the drones from flying into any locations occupied by other drones at time t. For *SR3* which concerns collisions with

dynamically appearing, moving obstacles, the proposed system provides a similar approach as *SR1* that helps the drones to avoid all successfully detected moving obstacles in an online manner.

3. Collision-Free Path Generation and Navigation

Figure 1 presents a high-level system architecture and overview of the proposed online, collision-free path generation and navigation system for swarms of UAVs. The main components of the proposed system include the following: (1) a CEP and collision prediction module, (2) a mutually exclusive locking mechanism, and (3) a collision avoidance mechanism. The inputs to the system are the UAV location updates, static obstacle detections, and moving obstacle detections. Based on these three inputs, the CEP and collision prediction module predicts: (1) UAV-to-UAV collisions, (2) UAV-to-static-obstacle collisions, and (3) UAV-to-moving-obstacle collisions. Our collision avoidance mechanism tries to find best ways to avoid or bypass collisions and computes collision-free routes for UAVs in an online manner. In a densely populated and cluttered flying zone, it might not be possible to immediately compute a bypass route for all drones. In such scenarios, the proposed system might put some of the drones into the hover-in-place mode until the situation improves and the routes clear. Additionally, it may also let some UAVs to temporarily retreat or backtrack to find more suitable, collision-free routes.

The proposed system implements a safety-first approach. Therefore, a hazard-free, safe operation of the swarm takes precedence overall of the other objectives including lengths of the UAV routes, timely arrival of the UAVs to their destinations, and achievement of any other mission-specific goals. As a consequence, we do not formulate the problem as an optimization problem. Instead, we implement a stochastic, greedy approach that tries to find safe and efficient routes. The UAVs take off from their start locations and fly uninterruptedly towards their destinations until the CEP and collision prediction module predicts a collision, in which case our collision avoidance mechanism is invoked to avoid the collision. In addition, our mutually exclusive locking mechanism prevents multiple UAVs from attempting to fly to the same location at the same time. At each step, the proposed approach makes a stochastic, greedy decision for each UAV. It tries to find a next location for each UAV which is not only safe but also reduces the distance from the destination. The main components of the proposed system are described in the following subsections.

3.1. Complex Event Processing and Collision Prediction. Complex event processing (CEP) is a technique for real-time, fast processing of a large number of events from one or more event streams to derive and identify important complex events and patterns in the event streams. CEP has been successfully used in a variety of business domains including retail management, health care, and cloud computing [18, 19]. For example, in retail management, CEP can be used to detect shoplifting and out-of-stock events. The basic or primitive events in CEP are processed into complex or composite events by means of event processing queries, which are written in a Structured Query Language-(SQL-) like language. Therefore, CEP provides a similar functionality for real-time event streams that a relational database management system provides for persistent data.

One of the most widely used CEP tools is the Esper CEP engine (http://www.espertech.com/esper/), in which the event processing queries are written in the Event Processing Language (EPL). There are three main steps for using the Esper CEP engine:

(1) In the first step, event types and sources of events are registered with the CEP engine. An event class in Esper is written as a Plain Old Java Object.

(2) The second step requires event processing queries to be written in EPL.

(3) Finally, in the third step, event sinks are implemented which can be used to perform some suitable control and repair actions.

The CEP and collision prediction module in our proposed system uses a CEP engine to monitor and keep track of the current location of the UAVs and of the successfully detected static and moving obstacles. Table 2 presents the three types of events from the proposed system along with their properties. The UAVs generate and send location update events on regular intervals, for example, every 50 milliseconds. A drone location event (*DroneLocEvent*) contains the drone name of the concerned drone $d_i \in$ SWARM, drone location l_i in the three-dimensional flying zone AREA, and the event time t. The CEP engine receives and processes these events to predict possible UAV-to-UAV collisions in the swarm. Similarly, for each successfully detected static obstacle, a static obstacle event (SObsEvent) is generated and sent to the CEP engine. A static obstacle detection event contains the obstacle name of the static obstacle $so_i \in$ STA_OBS and the location $l_i \in$ AREA of the static obstacle. The CEP engine processes all UAV location update events and static obstacle detection events to predict UAV-to-static-obstacle collisions. Finally, for successfully detected moving obstacles, moving obstacle events (MObsEvents) are generated and sent to the CEP engine. A moving obstacle detection event contains obstacle name of the moving obstacle $mo_i \in$ MOV_OBS, the location $l_i \in$ AREA of the moving obstacle, and the event time t. The CEP engine processes UAV location update events and moving obstacle detection events to predict UAV-to-moving-obstacle collisions.

The proposed system implements three EPL queries to process the three types of events and determine if a drone is flying in a close proximity of another drone or a static or moving obstacle. Listing 1 presents the first query. It uses DroneLocEvents to check if two drones are in a close proximity of each other. If a match is found, the CEP engine triggers the concerned event sink, which may predict a UAV-to-UAV collision and then invoke the collision

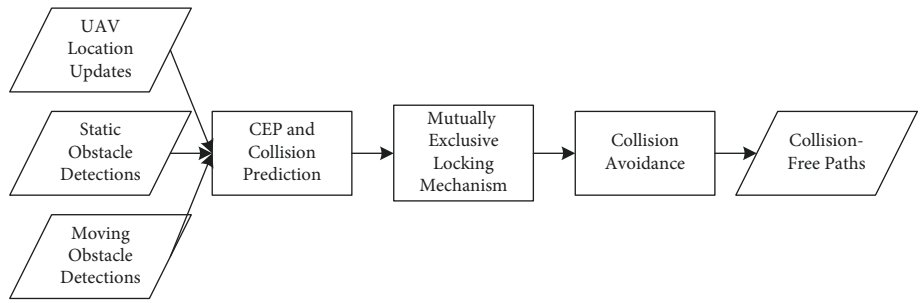


Figure 1: Overview of the proposed online, collision-free path generation and navigation system for swarms of UAVs.

Table 2: Three types of CEP events from the proposed system.

Event type	Properties
DroneLocEvent	Drone name d_i, drone location l_i, event time t
SObsEvent	Static obstacle name so_i, obstacle location l_i
MObsEvent	Moving obstacle name mo_i, obstacle location l_i, event time t

avoidance mechanism to prevent the UAVs from colliding into each other.

The second query in Listing 2 uses DroneLocEvents and SObsEvents to determine if a drone is in close proximity of a static obstacle. Similarly, the third query in Listing 3 uses DroneLocEvents and MObsEvents to determine if a drone is in close proximity of a moving obstacle. In each case, the relevant event sink is triggered, which may predict a collision and invoke the collision avoidance mechanism.

To predict the three different kinds of collisions, the event sinks use various parameters and rules. The parameters include the (current) locations of the drones and of the static and moving obstacles and the desired next locations of the drones. The collision prediction rules are presented in Algorithm 1. Rule 1 states that a UAV-to-UAV collision is predicted when the desired next location of a drone $d_i \in \text{SWARM}$ is the same as the current or the desired next location of another drone $d_j \in \text{SWARM} \mid i \neq j$. Similarly, Rule 2 is used for predicting UAV-to-static-obstacle collisions, which can occur if a drone $d_i \in \text{SWARM}$ attempts to fly to a location occupied by a static obstacle $so_j \in \text{STA\_OBS}$. Finally, Rule 3 states that a UAV-to-moving-obstacle collision is predicted when the desired next location of a drone $d_i \in \text{SWARM}$ is the same as the current location of a moving obstacle $mo_j \in \text{MOV\_OBS}$.

3.2. Mutually Exclusive Locking Mechanism. The CEP and collision prediction module described in the previous section covers most of the scenarios that can lead to a UAV collision. However, since the UAVs may move fast and arbitrarily, some failures and collisions can still occur. For instance, during a mission, a location $l_i \in \text{AREA}$ is free and two UAVs $d_i, d_j \in \text{SWARM} \mid i \neq j$ concurrently decide to move to l_i. If the CEP and collision prediction module takes slightly longer to predict the UAV-to-UAV collision, the collision avoidance mechanism might not be left with enough time to prevent the collision. However, if the CEP and collision prediction module quickly and correctly predicts the collision, the collision avoidance module can save the UAVs d_i and d_j by allowing only one of them to continue flying to l_i. The other UAV will either be redirected to another location or will fail to move in the current iteration. To prevent such scenarios and failures, we augment our collision prediction approach with a mutually exclusive locking mechanism.

The proposed system uses mutually exclusive locks on the current and the immediate next locations of UAVs to prevent multiple UAVs from attempting to move to the same location at the same time. The lock state of each location $l_i \in \text{AREA}$ can be either *locked* or *unlocked*. The current location of each UAV is always considered locked for all other UAVs. Moreover, as soon as a UAV decides its next move, the system puts a mutually exclusive lock on the immediate next location of the UAV so that the other UAVs do not attempt to move to the same location. Similarly, while deciding about their next moves, the UAVs first check the lock state of the possible next locations and only attempt to move to some of the unlocked locations. Moreover, if multiple UAVs $d_1, d_2, \ldots, d_N \in \text{SWARM}$ concurrently attempt to lock the same location, only one of them acquires the lock. For scenarios involving very short time intervals, the mutually exclusive locks may be acquired in one time interval and the moves may be performed in the next time interval. The UAVs release the locks of their previous locations as soon as they fly to their next locations.

Figure 2 illustrates the proposed locking mechanism. It shows that a UAV always keeps a mutually exclusive lock for its current location. Moreover, when deciding about a next move, it first checks the lock state of all possible next locations. It then attempts to lock one of the unlocked locations. After successful locking of the next location, it moves to the next location. Finally, it releases the lock of the previous location. In this way, two or more UAVs never attempt to move to the same location at the same time.

3.3. Collision Avoidance Mechanism. Whenever the CEP and collision prediction module predicts a collision, it invokes our collision avoidance mechanism which tries to find best ways to avoid the predicted collisions and computes collision-free routes for UAVs in an online manner. Based on the severity of the predicted collision, its surroundings, and the overall situation of the SWARM and of the successfully

```
select A.droneName as aName, A.x as aX, A.y as aY, A.z as aZ,
B.droneName as bName, B.x as bX, B.y as bY, B.z as bZ, from
DroneLocEvent.win:time(1 sec) A, DroneLocEvent.win : time(1 sec)
B where A.droneName != B.droneName and A.x in [B.x − 2 : B.x + 2]
and A.y in [B.y − 2 : B.y + 2] and A.z in [B.z − 2 : B.z + 2]
and (A.x = B.x or A.y = B.y or A.z = B.z)
```

Listing 1: EPL query to determine if two drones are in a close proximity of each other.

```
select A.droneName as aName, A.x as aX, A.y as aY, A.z as aZ,
O.obstacleName as oName, O.x as oX, O.y as oY, O.z as oZ, from
DroneLocEvent.win : time(1 sec) A, SObsEvent.win : time(1 hour) O
where A.x in [O.x − 1 : O.x + 1] and A.y in [O.y − 1 : O.y + 1] and A.z in
[O.z − 1 : O.z + 1] and (A.x = O.x or A.y = O.y or A.z = O.z)
```

Listing 2: EPL query to determine if a drone is in close proximity of a static obstacle.

```
select A.droneName as aName, A.x as aX, A.y as aY, A.z as aZ,
O.obstacleName as oName, O.x as oX, O.y as oY, O.z as oZ, from
DroneLocEvent.win : time(1 sec) A, MObsEvent.win : time(1 sec) O
where A.x in [O.x − 2 : O.x + 2] and A.y in [O.y − 2 : O.y + 2] and A.z in
[O.z − 2 : O.z + 2] and (A.x = O.x or A.y = O.y or A.z = O.z)
```

Listing 3: EPL query to determine if a drone is in close proximity of a moving obstacle.

(1) {**Rule 1**}
(2) $\forall d_i, d_j \in \text{SWARM} \mid i \neq j$, let $l_i, l_m \in \text{AREA}$ be the current and the desired next locations of d_i and similarly $l_j, l_n \in \text{AREA}$ be the current and the desired next locations of d_j
(3) **if** $l_m = l_j \vee l_m = l_n \vee l_n = l_i$ **then**
(4) predict a UAV-to-UAV collision
(5) **end if**
(6) {**Rule 2**}
(7) $\forall d_i \in \text{SWARM}, \forall \text{so}_j \in \text{STA\_OBS}$, let $l_m \in \text{AREA}$ be the desired next location of d_i and $l_j \in \text{AREA}$ be the location of so_j
(8) **if** $l_m = l_j$ **then**
(9) predict a UAV-to-static-obstacle collision
(10) **end if**
(11) {**Rule 3**}
(12) $\forall d_i \in \text{SWARM}, \forall \text{mo}_j \in \text{MOV\_OBS}$, let $l_m \in \text{AREA}$ be the desired next location of d_i and $l_j \in \text{AREA}$ be the current location of mo_j
(13) **if** $l_m = l_j$**then**
(14) predict a UAV-to-moving-obstacle collision
(15) **end if**

Algorithm 1: Collision prediction rules.

detected static and moving obstacles (STA_OBS and MOV_OBS) in AREA, our collision avoidance mechanism uses one of the three collision avoidance techniques in the following order: (1) redirecting the UAV into another direction, (2) putting the UAV into the hover-in-place mode until the route is cleared, and (3) temporarily retreating or backtracking the UAV to explore some alternate collision-free routes. The pseudocode of the proposed collision avoidance mechanism is given in Algorithm 2. Our backtracking approach is described in the following section.

The first collision avoidance technique, namely, redirecting the UAV into another direction, means changing the

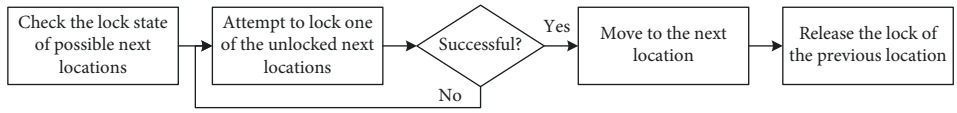


Figure 2: The mutually exclusive locking mechanism.

(1) redirect the UAV into another direction
(2) **if** not successful **then**
(3) activate the hover-in-place mode until the UAV route is cleared
(4) **end if**
(5) **if** the UAV hovers for too long or if it takes too long to find a suitable collision-free route **then**
(6) temporarily backtrack the UAV to explore some alternate collision-free routes
(7) **end if**

Algorithm 2: Collision avoidance mechanism.

flying direction of the UAV. For example, if a UAV is flying in the x dimension of AREA, but the CEP and collision prediction module predicts a collision due to the presence of an obstacle or another UAV on the path, then the UAV cannot continue a hazard-free flight in the x dimension any more. Therefore, the collision avoidance mechanism redirects the UAV to fly in the y or z dimension so the UAV may be able to avoid the collision. However, in a densely populated and cluttered flying zone, the collision avoidance mechanism might not be able to immediately compute a bypass route for all drones. Therefore, in such scenarios, the proposed collision avoidance mechanism activates the hover-in-place mode for some of the UAVs until the situation improves and the routes clear. Additionally and as a last resort, it temporarily backtracks some UAVs to explore some alternate collision-free routes. It should be noted that all three collision avoidance techniques incur some overhead, which might extend the routes and increase the flight durations for some of the UAVs. However, as explained previously, this is inevitable for a safety-first approach.

3.4. Backtracking Approach. The proposed backtracking approach temporarily retreats or backtracks a UAV, so it may explore some alternate collision-free routes. As shown in Algorithm 2, the proposed backtracking algorithm is triggered in two situations: (1) if a UAV hovers for too long or (2) if a UAV keeps on moving but it does not find a suitable collision-free route to reach to its destination. In densely populated, cluttered flying zones, such situations are not unprecedented. Sometimes, a UAV keeps on hovering or moving near its destination, but it does not find a collision-free route to reach to the destination because some other UAVs or obstacles reside between the UAV and its destination location and thus obstructs the UAV's routes. In such situations, it is important to allow the UAV to temporarily retreat or backtrack, so it may be able to explore some alternate routes to reach to its destination.

The pseudocode of the proposed backtracking algorithm is presented in Algorithm 3. The algorithm iterates until the required number of backtrack steps is successfully completed or the maximum number of backtrack attempts is reached (line 9). In each iteration, it attempts to move the UAV in the opposite direction of the UAV's destination. It randomly chooses one of the three dimensions (x, y, z) and tries to move the UAV so that the distance from the destination is increased. The backtrack algorithm does not disable the CEP and collision prediction module and the mutually exclusive locking mechanism. Thus, in each iteration, the UAV either backtracks one step or if a collision hazard is found or the UAV fails to lock the required location, then it hovers at its current location. When the UAV returns to the normal flight mode (line 10), it explores some alternate collision-free routes to reach to its destination.

4. An Illustrative Example

In this section, we present a small example to illustrate the main components and steps of the proposed online, collision-free path generation and navigation system. Although the proposed system works for a realistic, three-dimensional flying zone, it is difficult to illustrate and demonstrate a three-dimensional flying zone on a paper. Therefore, we use a two-dimensional flying zone for a simpler illustration.

Figure 3 presents an illustrative example with four UAVs, two static obstacles, and four moving obstacles in a two-dimensional flying zone. The flying zone in our example is shown as a 7×7 grid, where all consecutive locations are a uniform, fixed distance apart from one another. The start and destination locations of each drone are also highlighted. The goal is to route the drones from their start locations to their destination locations while avoiding collisions with static and moving obstacles and with the other drones in the swarm.

It should be noted that the knowledge of the precise locations of the obstacles in this example is only for illustration purposes. As described previously, the proposed system does not make any assumptions on the number and locations of the static and moving obstacles in the flying zone. Similarly, although Figure 3(a) shows that all moving obstacles are present in the flying zone before the start of the

```
(1) while the UAV is in the backtrack mode do
(2)    randomly choose one of the three dimensions (x, y, z)
(3)    attempt to move the UAV in the opposite direction of its destination
(4)    if a collision-hazard is found or the required location is locked by another UAV then
(5)       hover in place
(6)    else
(7)       backtrack the UAV by moving it one step away from its destination
(8)    end if
(9)    if the required number of backtrack steps is successfully completed or the maximum number of backtrack attempts is reached
       then
(10)      deactivate the backtrack mode and return to the normal flight mode
(11)   end if
(12) end while
```

Algorithm 3: Backtracking.

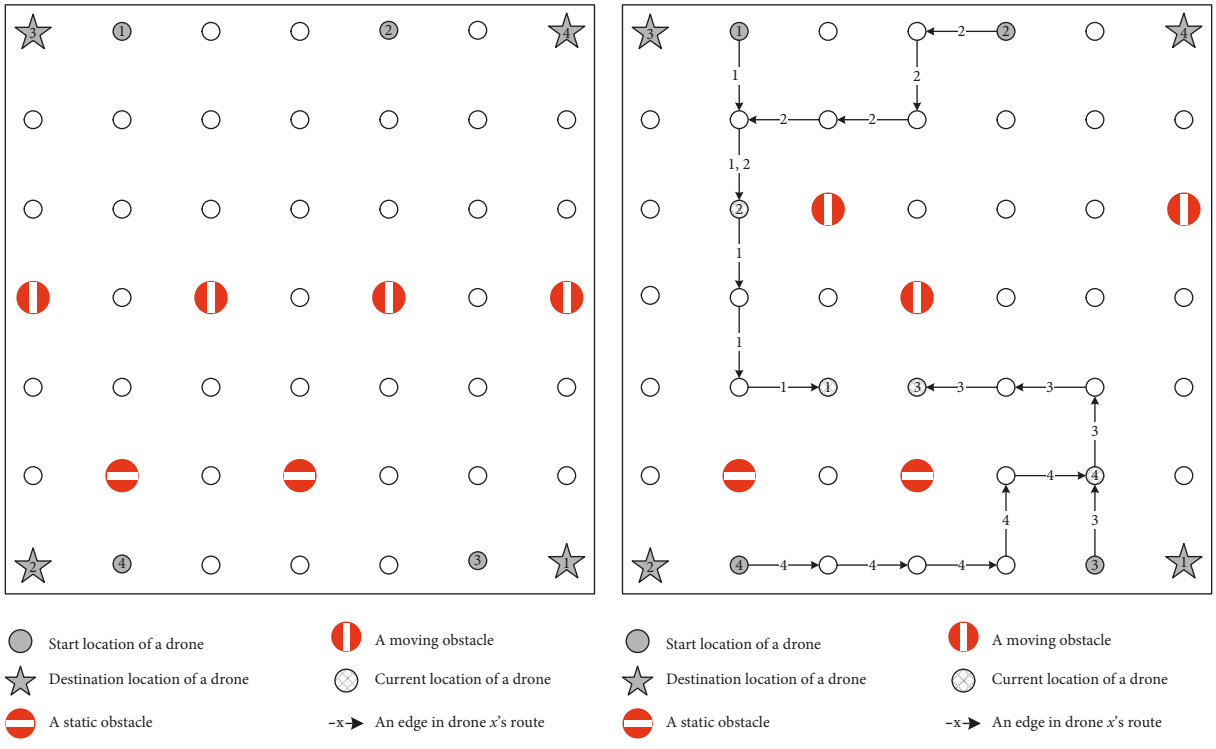
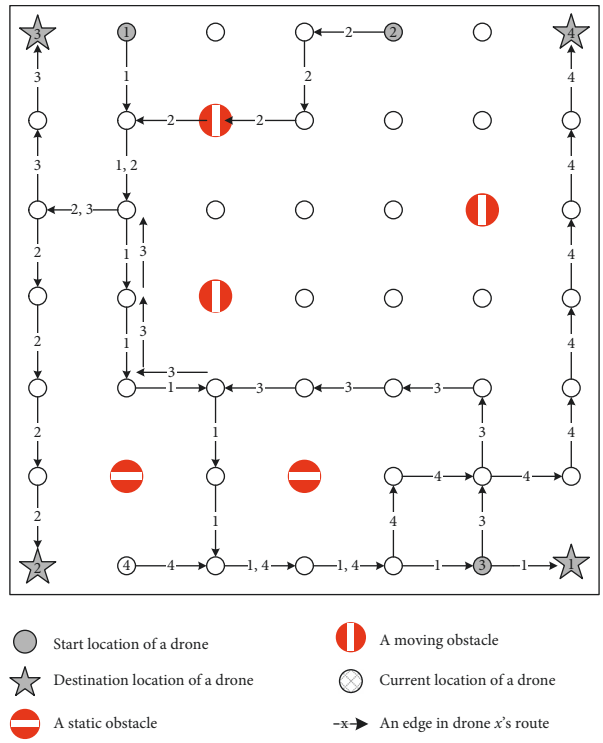


Figure 3: Continued.



(c)

Figure 3: A simple illustrative example with four UAVs, two static obstacles, and four moving obstacles in a two-dimensional flying zone. (a) Before the start of the mission. (b) After five time intervals. (c) After the completion of the mission.

mission, in a realistic scenario, some moving obstacles (for example, birds) may dynamically appear in the flying zone during the execution of the mission.

Figure 3(b) presents a snapshot of the flying zone after five time intervals have elapsed since the start of the mission. It shows that each UAV started flying from its start location and flew towards its destination location while randomly choosing to fly in the horizontal or vertical dimension in each time interval. Figure 3(b) also shows that the left most moving obstacle from Figure 3(b) left the flying zone during the execution of the mission and that the remaining moving obstacles moved to some new arbitrary locations within the flying zone. Although the moving obstacles moved in an arbitrary fashion either horizontally or vertically, in five time intervals each moving obstacle moved only one step, that is, only to a next consecutive location in the flying zone. Therefore, the moving obstacles moved slower than the drones. This is a reasonable assumption because if the moving obstacles move faster than the drones, even the most advanced and fastest collision detection, prediction, and avoidance mechanisms will not be able to avoid UAV-to-moving-obstacle collisions.

The labelled, directional edges in Figure 3(b) show the collision-free UAV routes generated by the proposed system in an online manner. For example, in the top left corner of Figure 3(b), the first downward edge labelled 1 means that UAV 1 flew in the downward direction. Similarly, the next edge in the same direction labelled 1, 2 shows that UAVs 1 and 2 used the same edge. However, two UAVs using the same edge does not mean a UAV-to-UAV collision. A UAV-to-UAV collision on an edge can happen when two UAVs fly at the same edge at the same time. In this example, UAV 1 and UAV 2 flew on the same edge but in different time intervals. UAV 1 left the edge before UAV 2 arrived there, and hence, there was no collision hazard between the two UAVs. Figure 3(b) also shows the current locations of the UAVs after five time intervals. It can be seen that all UAVs except UAV 3 flew five steps. UAV 3 flew four steps and then hovered in the fifth time interval because the system could not find a collision-free move for UAV 3.

The UAVs in Figure 3 used the proposed mutually exclusive locking mechanism at each step. As described in Section 3.2, each UAV first attempted to lock one of the unlocked next locations. After successful locking of their next locations, the UAVs moved to their next locations and released the locks of their previous locations. Therefore, at any time, two or more UAVs did not attempt to move to the same location in the flying zone.

UAV 1 in Figure 3(b) started flying vertically in the downward direction and continued towards its destination until it detected a static obstacle. At this stage, our CEP and collision prediction module predicted a UAV-to-static-obstacle collision and invoked our collision avoidance mechanism, which redirected the UAV into the horizontal, rightward direction, so the drone could continue flying towards its destination. However, in the same time interval, UAV 3 tried to fly into the same location where UAV 1 was

headed. The two UAVs detected each other, and the UAV and collision prediction module predicted a UAV-to-UAV collision. As a result, our collision avoidance mechanism was invoked, which tried to redirect UAV 3 in the vertical, upward direction, but the UAV detected a moving obstacle at that location, and the CEP and collision prediction module predicted a UAV-to-moving-obstacle collision. Therefore, the collision avoidance mechanism activated the hover-in-place mode for UAV 3, but let UAV 1 to lock and then move to the next location. Hence, UAV 3 flew only four steps in five time intervals. In this example, UAVs 2 and 4 did not encounter a collision hazard and flew normally towards their destinations. Moreover, none of the UAVs hovered for too long or took too long to find a suitable, collision-free route. Therefore, our backtracking algorithm was not invoked.

Figure 3(c) shows a snapshot of the flying zone after the completion of the mission. It shows that how each drone found its way to its destination while avoiding obstacles and other drones on its way. Once again, the remaining three moving obstacles moved to some new arbitrary locations within the flying zone. In the sixth time interval, UAV 1 was redirected in the downward direction to avoid a collision with UAV 3. Similarly, after flying downwards for two time intervals, UAV 1 reached the end of the flying zone and was once again redirected to the horizontal, rightward direction. Finally, after flying for a few more intervals in the rightward direction, UAV 1 reached its destination. As can be seen in Figure 3(c), all other UAVs found their ways in similar ways.

5. Implementation and Experimental Evaluation

To demonstrate and evaluate our proposed system, we have developed a software simulator. This section briefly describes some important implementation details along with an experimental evaluation involving a series of experiments. We also compare our results with the results of the following four approaches:

(1) *Particle Swarm Optimization- (PSO-) Based Approach*. Sujit and Beard's [17] PSO-based path planning approach generates paths for a swarm of drones.

(2) *Greedy Heuristics and Genetic Algorithms (GAs) Approach*. Silva Arantes et al.'s [11] approach uses greedy heuristics and GAs to generate and optimize paths for a UAV under critical situations.

(3) *Rapidly Exploring Random Trees (RRTs)*. A sampling-based path planning algorithm proposed by LaValle [9].

(4) *RRT**. An extension of RRT developed by Karaman and Frazzoli [10] to plan optimal paths.

5.1. Implementation Details. The implementation of the first main component of the proposed system called the CEP and collision prediction module is based on the Esper CEP engine and Algorithm 1. The second component, called the mutually exclusive locking mechanism, implements the proposed locking mechanism presented in Figure 2. Similarly, the collision avoidance component implements Algorithms 2 and 3.

We have implemented a simple, controlled simulation platform that does not take into account complex physical phenomena and uncontrolled environment variables such as gravity and wind. The objective is to test and evaluate the proposed system in an ideal scenario while ignoring and minimizing the effects of external uncontrolled factors. Therefore, it is easier to analyze and interpret the results. The implementation assumes that all drones fly at the same speed and that there were no internal drone failures during the execution of the mission. We also assume that at least one feasible path exists for each UAV.

5.2. Experiment Design and Setup. Table 3 presents the experiment design. The experimental evaluation comprises four experiments. In each experiment, we ran our proposed approach, Sujit and Beard's [17] PSO-based approach, Silva Arantes et al.'s [11] greedy heuristics and GA-based approach, LaValle's [9] RRT algorithm, and Karaman and Frazzoli's [10] RRT* algorithm 10 times and used a random seed every time. All results reported in this section are averaged over 10 runs. The experiments were run on an Intel Core i7-4790 processor with 16 gigabytes of memory. The length of the time interval used in the software simulator was 50 milliseconds. We measured the following dependent variables:

(i) *Average Route Length (ARL)*. The average UAV route length measured as the number of UAV moves in the discretized flying zone. A UAV route is a sequence of moves or steps from UAV's start location to UAV's destination location. To be minimized to generate shorter routes.

(ii) *Length of the Longest Route (LLR)*. The total number of steps in the longest generated route. To be minimized to generate shorter routes.

(iii) *Number of Collisions (NC)*. The number of UAV collisions. To be minimized to generate safer routes.

(iv) *Computation Time (T)*. Computation time of the algorithm in milliseconds (ms). It is the time that the algorithm takes to run and produce the results. To be minimized to reduce the computation overhead.

Experiment 1 was designed to simulate a small problem instance. The main objective was to evaluate the collision prediction and avoidance capabilities of the proposed system in a simpler scenario. The experiment used a $10 \times 10 \times 10$ flying zone with 20 drones, 20 static obstacles, and 20 moving obstacles. Experiment 2 used a large problem instance involving a larger flying zone and a higher number of drones and obstacles and was designed to evaluate the proposed system for a larger problem instance. The

Table 3: Experiment design.

Experiment	1	2	3	4
Experiment type	Small	Large	Cluttered flying zone	High risk of UAV collisions
Flying zone	$10 \times 10 \times 10$	$20 \times 20 \times 20$	$10 \times 10 \times 10$	$20 \times 20 \times 20$
Number of UAVs	20	50	20	100
Static obstacles	20	50	40	50
Moving obstacles	20	50	40	50

experiment used a $20 \times 20 \times 20$ flying zone with 50 drones, 50 static obstacles, and 50 moving obstacles.

The third experiment evaluated the proposed system for a densely populated, cluttered environment involving a large number of static and moving obstacles. Experiment 3 used a similar experiment design as Experiment 1, but with twice as many static and moving obstacles. Finally, the objective of Experiment 4 was to evaluate the proposed system in a scenario involving high risks of drone collisions. The experiment used a similar experiment design as Experiment 2 but with twice as many drones.

All drones and obstacles were placed randomly. However, to ensure that the drones do not collide during take off, unique start locations were used and no obstacles were placed at the drone start locations. Similarly, the destination locations for the drones were also chosen randomly, but it was ensured that all destination locations are unique and that no obstacles were present at the destination locations.

5.3. Results and Analysis. The results are presented in Table 4. The best results in the table are highlighted in **bold** font. The results in the ARL and LLR columns show that the RRT* algorithm [10] produced the shortest drone routes in all experiments, while RRT [9] generated the second shortest routes. The NC column shows that the proposed approach produced the safest routes in all experiments, while the greedy heuristics and GA-based approach [11] produced the second safest routes. In terms of T, RRT performed the best, while the proposed approach performed second best in Experiments 2 and 3, and the greedy and GA-based approach performed second best in Experiments 1 and 4. The PSO-based approach [17] did not perform best or second best with respect to any dependent variable.

In the first experiment, the ARL for the proposed, PSO-based, greedy and GA-based, RRT, and RRT* algorithms was 17, 25, 26, 16, and 15, respectively. Similarly, the LLR for the proposed, PSO-based, greedy and GA-based, RRT, and RRT* algorithms was 36, 49, 47, 36, and 29, respectively. The five approaches also produced similar results in Experiments 2 to 4. Therefore, the RRT* algorithm produced the shortest routes in all experiments.

Although the proposed approach did not produce the shortest routes, it produced the safest routes in all four experiments. The NC column in Table 4 shows the number of collisions or crashes for the proposed, PSO-based, greedy and GA-based, RRT, and RRT* algorithms. The total number of crashes in all experiments was 0, 14, 9, 18, and 29, respectively. As stated in Section 3, the proposed system provides a safety-first approach in which a hazard-free, safe operation of the UAV swarm takes precedence over any other objectives including the route length. This safety-first aspect of the proposed approach is evident in all experiments. The proposed approach generated slightly longer routes by trading route length for UAV safety. As a result, all UAV-to-UAV, UAV-to-static-obstacle, and UAV-to-moving-obstacle collisions were avoided, and all drones successfully completed their maneuvers.

Table 4: Comparison of the results with a UAV-based approach [17], a greedy heuristics and GA-based approach [11], and two sampling-based path planning algorithms called RRT [9], and RRT* [10].

Approach	Experiment	RRT	LLR	NC	T (ms)
Proposed	1	17	36	**0**	670
	2	34	62	**0**	683
	3	20	47	**0**	642
	4	36	97	**0**	637
PSO	1	25	49	3	949
	2	54	71	1	894
	3	35	53	5	932
	4	34	91	5	793
Greedy and GA	1	26	47	1	600
	2	53	69	**0**	715
	3	36	56	4	714
	4	34	94	4	627
RRT	1	16	36	2	**481**
	2	29	**58**	4	**598**
	3	20	44	7	**610**
	4	34	**86**	5	**599**
RRT*	1	**15**	**29**	3	612
	2	**27**	**58**	4	721
	3	**18**	**37**	9	738
	4	**30**	89	13	688

The last column in Table 4 shows the T results of the five algorithms in milliseconds. The results show that the RRT algorithm took the least amount of time to run and produce the results in all experiments, while the proposed approach performed second fastest in Experiments 2 and 3, and the greedy and GA-based algorithm performed second fastest in Experiments 1 and 4. Therefore, the results show that the proposed algorithm has a low computation overhead and it generates safe and efficient routes in a reasonable amount of time.

Experiment 1 results show that the proposed system is suitable for smaller problem instances. The performance and scalability of the proposed system are further demonstrated in the results of Experiment 2 which produced drone routes for a larger problem instance. In Experiment 3, the drones encountered more obstacles on their ways because the flying

zone was cluttered with static and moving obstacles. It forced them to take longer routes to their destinations, but the proposed approach managed to successfully avoid all obstacles and collisions and routed all drones to their destinations. It shows that the proposed system is also suitable for densely populated, cluttered flying zones. Finally, Experiment 4 results show that the proposed approach is also suitable for complex problem instances involving high risks of drone collisions.

6. Related Work

The problem of motion safety of semiautonomous robotic systems is currently attracting significant research attention. A comprehensive overview of the problems associated with autonomous mobile robots is given in [20]. The analysis carried out in [21] shows that the most prominent routing schemes do not guarantee motion safety. Our approach resolves this issue and ensures not only safety but also provides efficient, online routing.

Macek et al. [22] proposed a layered architectural solution for robot navigation. They focused on the problem of safe navigation of a vehicle in an urban environment. They also distinguished between global route planning and collision avoidance control. However, in their work, they focused on the safety issues associated with the navigation of a single vehicle and did not consider the problem of collision-free path generation and navigation in the context of fleets or swarms of robots. Aniculaesei et al. [23] presented a formal approach that employs formal verification to ensure motion safety. They used UPPAAL model checker (http://www.uppaal.org/) to verify that a moving robot engages brakes and safely stops upon detection of an obstacle. Since our proposed system does not assume any a priori knowledge on the numbers and locations of the static and moving obstacles and does not depend on a preliminary, offline motion planning phase, the safety requirements cannot be verified before the start of the mission. Therefore, we did not employ formal verification. The solution proposed in our work is fast and flexible as it dynamically generates and recomputes the drone routes in an online manner and avoids unnecessary stopping of the drones.

Petti and Fraichard [24] proposed an approach that relies on partial motion planning to ensure safety. They state that calculation of an entire route is such a complex and compute-intensive problem that the only viable solution is a computation of the next safe states and navigation within them. Their solution supports navigation of a single vehicle. In our work, we have discretized the flying zone and have developed a highly efficient system that computes the next safe states for an entire swarm and provides a mechanism for online path generation and collision avoidance.

A comprehensive literature review on motion planning algorithms for UAVs can be found in [6]. The approaches reviewed in [6] are applicable to a preliminary, offline motion planning phase to plan and produce an efficient path or trajectory for a UAV before the start of the mission. Our proposed system does not depend on a planning phase and produces efficient, collision-free paths for an entire swarm in an online manner. A more recent survey on motion planning of UAVs can be found in [7].

Augugliaro et al. [8] presented an algorithm for generating collision-free trajectories for a quadrotor fleet. They focused on a planned approach that generates feasible paths ahead of time. LaValle [9] and Karaman and Frazzoli [10] presented sampling-based path planning algorithms called RRT and RRT*, respectively. RRT was designed to efficiently explore high-dimensional spaces by incrementally building a tree. RRT* is an extension of RRT. It was designed to plan optimal paths.

Majd et al. [15, 16] proposed a path planning and navigation approach for swarms of drones. They combined offline path planning with an online navigation approach and used machine learning and evolutionary algorithms to generate efficient paths while maximizing safety of the drones in the swarm. They also used collision prediction and drone reflexes to prevent collisions with unforeseen obstacles. In comparison, this paper presents an online, collision-free path generation and navigation approach, which does not need offline path planning.

Dong et al. [3] presented a software platform for online cooperative control of multiple UAVs. Their work focuses on monitoring and control of multiple UAVs from a ground control station. The approach does not generate paths for the UAVs. Instead, the complete flight information (including the UAVs paths) is provided to the ground control station that sends control commands to the UAV fleet. de Souza [13] and de Souza and Endler [14] presented an approach for movement coordination of swarms of drones using smart phones and mobile communication networks. They used CEP but only to analyze and evaluate the formation accuracy of the swarm. Moreover, their work focuses on the internal communication of the swarm and does not provide a solution for collision-free path generation. Bürkle et al. [12] proposed a multiagent system architecture for team collaboration in a swarm of drones. They also developed a simulation platform for patrolling or surveillance drones which monitor a protected area against potential intrusions. However, they did not address path planning and collision avoidance for the swarm.

Ivanovas et al. [4] proposed an obstacle detection and avoidance approach for a UAV. Their approach uses computer vision techniques for detecting static obstacles in stereo camera images. The main focus of their approach is on how some block matching algorithms can be used for obstacle detection. They did not present a path planning and collision avoidance approach for multiple UAVs. Barry and Tedrake [25] proposed an obstacle detection algorithm for UAVs that allows detect and avoid collisions in an online manner. Similarly, Lin [26] presented an online path planner for UAVs that detects and avoids moving obstacles. These approaches are only applicable for individual UAVs and do not provide support for a swarm of UAVs. In our work, we assumed that each UAV is equipped with an adequate obstacle sensing and detection capability and does not require any additional support for obstacle detection. Therefore, we focused on collision prediction and avoidance and online path generation and navigation for swarms of UAVs.

Sujit and Beard [17] proposed a PSO-based path planning algorithm for swarms of drones. In their approach, whenever a drone detects a moving obstacle, the PSO-based algorithm generates a new path for the drone depending on the time allowed to compute a new path before the collision can occur. Silva Arantes et al. [11] presented a UAV path planning approach for critical situations requiring an emergency landing of the UAV. Their approach uses greedy heuristics and UAVs to generate and optimize feasible paths under different types of critical situations caused by equipment failures.

In Section 5, we have presented a comparison of the results of our proposed approach with Sujit and Beard's [17] PSO-based approach, Silva Arantes et al.'s [11] greedy heuristics and GA-based approach, LaValle's [9] RRT algorithm, and Karaman and Frazzoli's [10] RRT* algorithm. The results show that our proposed approach produced the safest routes in all four experiments. Therefore, the proposed approach outperformed the PSO-based, greedy heuristics and GA-based, RRT, and RRT* approaches with respect to drone safety.

7. Conclusions

In this paper, we presented an online, collision-free path generation and navigation system for swarms of UAVs. The proposed system uses geographical locations of the UAVs and of the successfully detected, static and dynamically appearing, moving obstacles to predict and avoid the following: (1) UAV-to-UAV collisions, (2) UAV-to-static-obstacle collisions, and (3) UAV-to-moving-obstacle collisions. It comprises three main components: (1) a CEP and collision prediction module, (2) a mutually exclusive locking mechanism, and (3) a collision avoidance mechanism. The CEP and collision prediction module leverages efficient runtime monitoring and CEP to make timely predictions. The mutually exclusive locking mechanism prevents multiple UAVs from attempting to fly to the same location at the same time. The collision avoidance mechanism tries to find best ways to prevent the UAVs from colliding into one another with the successfully detected static and moving obstacles in the flying zone. Therefore, a distinctive feature of the proposed system is its ability to foresee risks of collisions in an online manner and proactively find best ways to avoid the predicted collisions in order to ensure safety of the entire swarm.

We also presented a simulation-based implementation of the proposed system along with an experimental evaluation involving a series of experiments and compared our results with the results of four existing approaches. The results showed that the proposed system successfully predicts and avoids all three kinds of collisions in an online manner. Moreover, it generates safe and efficient UAV routes, efficiently scales to large-sized problem instances involving dozens of UAVs and obstacles, and is suitable for densely populated, cluttered flying zones and for scenarios involving high risks of UAV collisions.

As part of our future work, we plan to implement the proposed system in a more realistic simulation environment that allows take into account complex physical phenomena and uncontrolled environment variables. Moreover, we want to test and evaluate our system for heterogeneous drones that may have diverse capabilities and fly at different speeds. Finally, an adequate support and online mechanisms to handle and control the situations arising from imprecise information of UAV locations and internal drone failures during mission execution are also planned as future works.

Data Availability

All important data are included in the manuscript (Table 4).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Academy of Finland projects. OpenCPS: Open Integrated Framework for Accelerating Development of Resilient CPS and CoRA: Continuous Resilience Assurance of Complex Software-Intensive Systems.

References

[1] R. Austin, "Unmanned aircraft systems: UAVS design, development and deployment," in *Aerospace Series*, Wiley, Hoboken, NJ, USA, 2010.

[2] A. Ashraf, A. Majd, and E. Troubitsyna, "Towards a realtime, collision-free motion coordination and navigation system for a UAV fleet," in *Proceedings of the Fifth European Conference on the Engineering of Computer-Based Systems, ECBS '17*, pp. 1–9, ACM, Larnaca, Cyprus, August 2017.

[3] X. Dong, B. M. Chen, G. Cai, H. Lin, and T. H. Lee, "Development of a comprehensive software system for implementing cooperative control of multiple unmanned aerial vehicles," in *Proceedings of the IEEE International Conference on Control and Automation*, pp. 1629–1634, Christchurch, New Zealand, December 2009.

[4] A. Ivanovas, A. Ostreika, R. Maskeliūnas, R. Damaševičius, D. Połap, and M. Woźniak, "Block matching based obstacle avoidance for unmanned aerial vehicle," in *Artificial Intelligence and Soft Computing*, L. Rutkowski, R. Scherer, M. Korytkowski, W. Pedrycz et al., Eds., Springer International Publishing, Salmon Tower Building, NY, USA, pp. 58–69, 2018.

[5] G. Pajares, "Overview and current status of remote sensing applications based on unmanned aerial vehicles (UAVs)," *Photogrammetric Engineering & Remote Sensing*, vol. 81, no. 4, pp. 281–330, 2015.

[6] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous UAV guidance," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1–4, pp. 65–100, 2010.

[7] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[8] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: a sequential convex programming approach," in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent*

Robots and Systems, pp. 1917–1922, Algarve, Portugal, October 2012.

[9] S. M. LaValle, "Rapidly-exploring random trees: a new tool for path planning," *Computer Science Deptartment*, Technical Report TR 98-11, Iowa State University, Ames, IO, USA, 1998.

[10] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[11] J. D. Silva Arantes, M. D. Silva Arantes, C. F. Motta Toledo, O. T. Júnior, and B. C. Williams, "Heuristic and genetic algorithm approaches for UAV path planning under critical situation," *International Journal on Artificial Intelligence Tools*, vol. 26, no. 1, Article ID 1760008, 2017.

[12] A. Bürkle, F. Segor, and M. Kollmann, "Towards autonomous micro UAV swarms," *Journal of Intelligent & Robotic Systems*, vol. 61, no. 1–4, pp. 339–353, 2011.

[13] B. J. O. de Souza, *An Approach for Movement Coordination of Swarms of Unmanned Aerial Vehicles Using Mobile Networks*, Master's thesis, Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, 2015.

[14] B. J. O. de Souza and M. Endler, "Coordinating movement within swarms of UAVs through mobile networks," in *Proceedings of the 2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pp. 154–159, Seattle, WA, USA, March 2015.

[15] A. Majd, A. Ashraf, E. Troubitsyna, and M. Daneshtalab, "Integrating learning, optimization, and prediction for efficient navigation of swarms of drones," in *Proceedings of the 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, pp. 101–108, Cambridge, UK, March 2018.

[16] A. Majd, A. Ashraf, E. Troubitsyna, and M. Daneshtalab, "Using optimization, learning, and drone reflexes to maximize safety of swarms of drones," in *Proceedings of the 2018 IEEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Brazil, January 2018.

[17] P. B. Sujit and R. Beard, "Multiple UAV path planning using anytime algorithms," in *Proceedings of the 2009 American Control Conference*, pp. 2978–2983, Saint Louis, MO, USA, June 2009.

[18] A. Mdhaffar, R. B. Halima, M. Jmaiel, and B. Freisleben, "Reactive performance monitoring of cloud computing environments," *Cluster Computing*, vol. 20, no. 3, pp. 2465–2477, Sep 2017.

[19] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD '06*, pp. 407–418, ACM, New York, NY, USA, June 2006.

[20] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots: Intelligent Robotics and Autonomous Agents*, MIT Press, Cambridge, MA, USA, 2011.

[21] T. Fraichard, "A short paper about motion safety," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pp. 1140–1145, Rome, Italy, April 2007.

[22] K. Macek, D. Vasquez, T. Fraichard, and R. Siegwart, "Safe vehicle navigation in dynamic urban scenarios," in *Proceedings of the 2008 11th International IEEE Conference on Intelligent Transportation Systems*, pp. 482–489, Beijing, China, October 2008.

[23] A. Aniculaesei, D. Arnsberger, F. Howar, and A. Rausch, "Towards the verification of safety-critical autonomous systems in dynamic environments," in *Proceedings of the First Workshop on Verification and Validation of Cyber-Physical Systems*, pp. 79–90, Reykjavík, Iceland, June 2016.

[24] S. Petti and T. Fraichard, "Partial motion planning framework for reactive planning within dynamic environments," in *Proceedings of the IFAC/AAAI International Conference on Informatics in Control, Automation and Robotics*, Exeter, UK, 2005.

[25] A. J. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3046–3052, Seattle, WA, USA, May 2015.

[26] Y. Lin, *Moving Obstacle Avoidance for Unmanned Aerial Vehicles*, Ph.D. thesis, Arizona State University, Tempe, AZ, USA, 2015.