LSwarm: Efficient Collision Avoidance for Large Swarms with Coverage Constraints in Complex Urban Scenes

Senthil Hariharan Arul, Adarsh Jagan Sathyamoorthy, Shivang Patel, Michael Otte, Huan Xu, Ming C Lin, and Dinesh Manocha

Abstract—In this paper, we address the problem of collision avoidance for a swarm of UAVs used for continuous surveillance of an urban environment. Our method, LSwarm, efficiently avoids collisions with static obstacles, dynamic obstacles and other agents in 3-D urban environments while considering coverage constraints. LSwarm computes collision avoiding velocities that (i) maximize the conformity of an agent to an optimal path given by a global coverage strategy and (ii) ensure sufficient resolution of the coverage data collected by each agent. Our algorithm is formulated based on ORCA (Optimal Reciprocal Collision Avoidance) and is scalable with respect to the size of the swarm. We evaluate the coverage performance of LSwarm in realistic simulations of a swarm of quadrotors in complex urban models. In practice, our approach can generate global trajectories in a few seconds and can compute collision avoiding velocities for a swarm composed of tens to hundreds of agents in a few milliseconds on dense urban scenes consisting of tens of buildings.

I. INTRODUCTION

Recent advances in multi-rotor UAVs have created new areas of application, including surveillance, search and rescue, and monitoring. Continuous surveillance involves gathering sensory data (such as from a camera) from all regions in an area or volume of interest by traversing a path optimized for maximum coverage while accounting for static and dynamic obstacles [1]. For applications like surveillance, using a single UAV/agent may not be effective, when time, battery capacity, reliability, and coverage performance are considered.

An obvious solution to this problem is to use a swarm of agents and have each agent follow an optimal path provided by a global coverage strategy that maximizes the covered area or the information gathered from the environment. We use “optimal coverage path” or “global coverage path” to refer to this global path in this paper. Many techniques have been proposed that have formulated this optimal coverage path for multi-agent systems [2], [3], [4], [5], [6].

However, some of the existing swarm algorithms ignore collisions between the swarm agents and dynamic obstacles in the environment [2], [4], while others do not provide solutions that can be scaled to hundreds of agents. In addition, making an agent hover to avoid collision would not work when multiple dynamic obstacles approach head-on. When an agent encounters large numbers of dynamic obstacles, a haphazard maneuver to avoid collisions will again lead to loss of valuable data on the ground. In addition, the resolution of the data gathered is important to ensure its usefulness and it is an important factor to consider while performing collision avoidance maneuvers.

Objective: In this paper, we address the complex problem of collision avoidance in a swarm of quadrotors during continuous surveillance of an urban environment. We impose constraints on coverage area and data resolution such that the quadrotors lose minimal ground coverage information while deviating to avoid collisions.

Main Contributions: We present LSwarm, a local collision avoidance method for quadrotor swarms performing continuous surveillance of large, complex 3-D urban environment (see Fig.1). LSwarm builds on ORCA [7] (a Velocity Obstacle based collision avoidance method), and to the best of our knowledge, it is the first implementation that considers all of the following:

(a) Collision avoidance with agents & dynamic obstacles,
(b) Collision avoidance with static obstacles,
(c) Scalability to large number of agents,
(d) Acceleration Limits (Dynamics constraint),
(e) Constraints on coverage,
(f) Uncertainty in position and velocity.

Our coverage constraints consider the following:
(i) Each agent conforms maximally to an optimal path given by a global coverage strategy during collision avoidance. In other words, the calculated collision avoidance velocities for each agent also minimizes the coverage area loss (defined in Section III.A). (ii) The coverage data obtained by the agents’ sensors have sufficient resolution (defined by Ground Sampling Distance (GSD)).

LSwarm employs a precomputed look-up table containing coverage area overlaps for various collision avoiding veloc-
ities (Section IV.G) so that collision avoidance under coverage constraints can be executed in real-time. Our collision avoidance method is compatible with any global coverage strategy that provides waypoints. We use a simple lawn mower strategy to provide the global coverage path in this paper.

We evaluate LSwarm’s performance with respect to ORCA in urban environments which contain a plethora of static obstacles such as buildings of various heights, and dynamic obstacles in the form of birds, helicopters and other swarm agents. We perform accurate simulations of the quadrotor swarm in four different 3-D urban scenes. Depending on the model, the number of buildings vary between 10 to 80, and the height of buildings vary between 5 to 30 meters. Our algorithm takes milliseconds to compute collision-free velocities for 50 quadrotor agents on eight cores. We observe that LSwarm provides an improvement of up to a factor of two in coverage area over the use of ORCA, as the number of dynamic obstacles increases.

We first survey the previous work in coverage strategies and collision avoidance in Section II. We formally define our coverage constraints and give a description of the global lawnmower strategy in Section III, and present our local collision avoidance algorithm in Section IV. Section V presents the experiments and results of our implementation. Section VI summarizes the paper and suggests possible directions for future work.

II. PREVIOUS WORK

Swarm agents such as UAVs have been an active area of research, due to their flexibility and a wide range of applicability in surveillance, search and rescue operations, and other information-gathering scenarios. A recent survey on control, planning, and design of aerial swarm robots can be found in [8].

A. Swarm Control Strategies for Coverage Optimization

In this section, we give a brief overview of prior work on swarm-based coverage optimization, and collision avoidance strategies.

Coverage Optimization: Urban area surveillance with added localization constraints during trajectory generation and optimization of the final swarm distribution to cover high priority surveillance regions is presented in [4]. Julian et al. present a scalable consensus-based approximation method to generate trajectories to maximize the information gathered by a swarm in [2]. Dames et al. [9], introduce a new formulation to track an unknown number of mobile targets using a team of robots based on random finite sets. Their method employs the Probability Hypothesis Density (PHD) filter to simultaneously estimate the number of targets and their positions after which the robots greedily choose actions to maximize certain control objectives. In [10] a distributed formulation of the PHD filter is used to search for and track the targets, which is combined with Lloyd’s algorithm to control the robots’ motions. Velagapudi et al. [6] present a distributed planner to explore moderately constrained environments that also accounts for communication constraints in large swarms in 2-D. A receding horizon planning strategy for multi-robot coverage in partially known environments is presented by Das et al. in [11].

Collision Avoidance through precomputed trajectories: Saha et al. [3] demonstrate a scalable incremental motion planning method to find the order in which the robots should be dispatched to their destination in constricted environments. Although this approach is useful for planning in narrow spaces in an urban setting, the time taken to precompute trajectories is high for even a group of tens of robots. Turpin et al. [5] present a centralized trajectory generation algorithm with optimality guarantees along with a scalable decentralized case. However, they assume the absence of static obstacles. A centralized feedback control law for uniform and ergodic domain coverage using multi-agent systems is shown in [12]. Trajectory generation for hundreds of quadrotors in dense continuous environments is demonstrated in [13]. This approach generates an inter-robot conflict annotated roadmap, then solves a generalized Multi-Agent Path Finding (MAPF) problem to avoid conflicts in the roadmap, while continuously smoothening the trajectory. A parallelization formulation for centralized trajectory generation methods for swarms is shown in [14].

In contrast to previous work, our method combines a global coverage strategy (based on the lawnmower strategy) with a local collision avoidance scheme (based on Velocity Obstacle) which avoids collisions on the fly. The global coverage strategy in our case accounts only for avoiding static obstacles and not the inter-agent collisions in the swarm. The decentralized local collision avoidance formulation can be scaled up to a large number of swarm agents, avoids having a single point of failure and also accounts for the dynamic nature of the environment such as dealing with obstacles not known a priori.

B. Space Filling Curves

Multiple works have investigated applying and simplifying the implementation of space-filling curves such as the lawnmower and Hilbert curves for global coverage. The first formal work on sweep search that used a probabilistic lawnmower sweep method is presented in [15]. In [16], the boustrophedon cellular decomposition of the free space is discussed. This work drastically simplifies the number of cellular decompositions compared to previous methods and allows a robot to cover each cell with back and forth lawnmower motions. Kong et al. [17] build on this work by allocating multiple cooperating and communicating robots to cover different areas in the environment in a distributed fashion. Vincent and Rubin [18] present a cooperative search using a swarm of UAVs that follows lawnmower trajectories to detect an evading target. The use of Hilbert space-filling curves for a geographic search using robot swarms is explored in [19]. Our method adopts the lawnmower strategy [16] to plan for optimal coverage paths, which are then used as a constraint for performing local collision avoidance that
minimizes deviation from the optimal coverage using a team of aerial robots.

C. VO-based collision avoidance

Collision avoidance with dynamic obstacles has been investigated extensively in [20], [21], [22], [23]. These approaches estimate the future positions of obstacles using their observed velocities. Velocity Obstacle (VO) [24] methods transform the collision avoidance problem in the workspace to an equivalent formulation in the velocity space. Reciprocal Velocity Obstacles (RVO) [25] improved on VO to tackle robot-robot collision avoidance, where each agent chooses its own velocity based on the velocity of other agents. Optimal Reciprocal Collision Avoidance (ORCA) [7] further built upon RVO by providing a sufficient condition for multiple robots to avoid collisions among one another, thus guaranteeing collision-free navigation. ORCA was extended to 3-D in [26] and to include a proximity based static obstacle avoidance in [27]. Proximal Policy Optimization (PPO), a policy gradient based reinforcement learning algorithm for decentralized collision avoidance is shown in [28]. Agents that use PPO have slightly better collision avoidance success rates, reach their goals sooner and have higher average speeds. Breitenmoser and Martinoli [29] combine multi-robot coverage and reciprocal collision avoidance and evaluate the new method’s benefits in several scenarios.

Although ORCA has several advantages since it is decentralized, and scalable, the algorithm assumes perfect state information in terms of position and velocity of each agent and does not model uncertainty.

D. Modeling Uncertainties

Many techniques have been proposed that extend the VO based algorithms to be more practical by considering the uncertainty in localization of the agents and sensing of nearby obstacles. Hennes et al. [30] provide an Adaptive Monte Carlo localization-based solution (AMCL) for modeling the pose of an agent and basic communication between agents to share the positions and velocities of nearby agents. Although this method works well for ground robots (2-D scenarios), the dependence of AMCL on odometry and laser data makes it hard to use it for UAVs. Recent advances in sensor technologies such as GPS-RTK (Real Time Kinematics) have improved the localization accuracy of the agents to centimeter levels. Thus, communicating such information between nearby swarm agents could result in a better collision avoidance scheme. Even in GPS-denied environments, works such as PIXHAWK [31] have shown the feasibility of vision-based localization and the use of stereo cameras to detect nearby obstacles with all computations being performed on an on-board computer. Such techniques can be used to detect and avoid unreactive dynamic obstacles such as birds. Other techniques take into account sensing and actuation uncertainties. Gopalakrishnan et al. [32] address this problem by presenting a tractable approximation of chance constraints along with the velocity constraints of RVO for 2D agents [25]. Snape et al. [33] and Kamel et al. [34] present Kalman filtering based methods to vary the safe distance between the agents based on position and velocity uncertainties. We make use of a similar Kalman filtering based method to handle uncertainties among agents and obstacles in LSwarm.

**Simulation:** OpenUAV, the state of the art in UAV simulation is presented in [35]. It is a cloud-enabled testbed for UAVs using ROS, Gazebo and PX4 simulation packages. The current implementation of OpenUAV is capable of simulating 9 UAVs with cameras and around 30 UAVs without simulated cameras. This number can improve as OpenUAV moves to a more scalable cloud computing platform.

While constraints to avoid collisions between swarm members and static obstacles are included in most trajectory generation methods, issues like the absence of dynamic obstacle avoidance, ability to adapt to changing scenes, and issues in scalability are still prevalent in most of these methods. ORCA is insufficient for collision avoidance in urban scenarios since it does not have a suitable static collision avoidance method for dense 3-D urban environments. LSwarms overcomes such limitations of ORCA and retains its advantages such as scalability and compatibility with global planners.

III. SWARM COVERAGE IN URBAN SCENES

In this section, we formally define our constraints on coverage loss and resolution of the collected data. We state the assumptions we make for each swarm agent and then describe the global lawnmower strategy that provides the optimal coverage path. LSwarms is compatible with any global method that generates waypoints for each agent, and in this paper we use a lawnmower strategy due to its simplicity.

A. Coverage Constraint Definition

As stated earlier, LSwarms coverage constraints ensure that collision avoidance velocities are calculated such that coverage area loss is minimized and the resolution of collected data is satisfactory.

1) Coverage Area Loss: Each swarm agent’s primary objective is to follow the optimal coverage path as closely as possible. The optimal coverage path provides the maximum coverage of the environment, and any deviation from this path would result in loss of coverage area. The following definitions apply to each quadrotor in the swarm.

Let \( \text{area}_{\text{pref}} \) be the preferred area that would have been covered if a quadrotor followed the optimal coverage path perfectly for a given period of time. Let \( \text{area}_{\text{actual}} \) be the actual area covered by the quadrotor. When the quadrotor does not face any obstacle while on the optimal coverage path, \( \text{area}_{\text{actual}} = \text{area}_{\text{pref}} \). In cases where the quadrotor faces obstacles, \( \text{area}_{\text{actual}} \cap \text{area}_{\text{pref}} < \text{area}_{\text{pref}} \). Let \( \text{area}_{\text{overlap}} \) denote the overlap area between \( \text{area}_{\text{pref}} \) and \( \text{area}_{\text{actual}} \). Then,

\[
\text{area}_{\text{overlap}} = \text{area}_{\text{pref}} \cap \text{area}_{\text{actual}}
\]

The overlap ratio is given by,

\[
\text{overlap ratio} = \frac{\text{area}_{\text{overlap}}}{\text{area}_{\text{pref}}}
\]
The coverage loss is then defined as $(1 - \text{overlap ratio})$. When overlap ratio = 1, (optimal path is perfectly followed), the coverage loss is 0.

2) Sensor Resolution: We assume that each quadrotor or the agent in our swarm has a sensor (a camera in our case) to gather information from the environment. The sensor is modeled to have a conical view of the environment with a fixed apex angle. The radius $r$ of the base circle of this view-cone would depend on the altitude $h$ at which the quadrotor is flying as:

$$ r = h \cdot \tan \theta $$

where $\theta$ is half of the apex angle of the cone.

This means that the sensor covers more area when the quadrotor is flying at a higher altitude, but the resolution of the sensor output at this higher altitude may not be satisfactory. In some collision avoidance scenarios, the quadrotor might change its altitude which again could result in loss of resolution.

To ensure that there are no detrimental effects to resolution, we use a measure called Ground Sampling Distance (GSD) and use it in our coverage constraints. GSD is the length on the ground denoted by one pixel on the image given by our camera. GSD increases with altitude and the greater the value of GSD, the poorer the resolution (see Fig.2). GSD is chosen as a metric for resolution because it provides a linear relationship between the flying altitude of the quadrotor and the features that can easily be covered on the ground from that altitude. For example, if our objective is to view the people on the ground, we can calculate the optimal height at which the quadrotor has to fly to make at least two pixels represent a person. Since GSD is calculated for a rectangular image, we take the largest area square within the base circle of the sensor cone for our calculation of GSD. The area of this square (of side $s$) is considered to be the area covered by a quadrotor at one instant of time. The GSD for a rectangular image is generally calculated as,

$$GSD_h = K_h \cdot h$$
$$GSD_w = K_w \cdot h$$

where $h$ is the flying altitude and $K$ is a constant calculated as:

$$K_h = \frac{\text{Sensor Height}}{f \cdot \text{Image Height}}$$
$$K_w = \frac{\text{Sensor width}}{f \cdot \text{Image Width}}$$

Where the sensor height and sensor width are the dimensions of the camera lens and $f$ is the focal length of the lens. Since we are considering a square camera footprint, $GSD_h$ is equal to $GSD_w$.

The relationship between the side of the largest area base square and the altitude from (1) is given as:

$$s = \frac{h \cdot \tan \theta}{\sqrt{2}}$$

![Fig. 2: Change in area of the coverage grid for different flying altitudes of the quadrotor is shown. $GSD_h$ and $GSD_w$ are a measure of the height and width of the pixels in the coverage grid. As the grid becomes larger, the finer details on the ground are missed.](image)

B. Assumptions on Swarm Agents

We assume that the sensor attached to the agent always faces downwards irrespective of the orientation of the agent. This can be achieved by attaching the sensor to a gimbal. We also assume that each agent knows its own position and velocity, and can sense the positions and velocities of other agents around it. All of these values could have some level of uncertainty in them. Section IV.E discusses our Kalman filtering method to handle this uncertainty.

C. Global Coverage Strategy

We use a simple lawn mower sweep to generate our optimal coverage path. Lawn mower sweep, which is also known as Boustrophedon coverage, is a simple yet widely used strategy that guarantees 100% coverage of a search space [16], [36]. Some implementations [16] segregate the obstacle and the non-obstacle region and perform a separate search sweep in each connected non-obstacle region. Similar ideas have also been used in unknown terrain; for example, [36] uses a combination of boustrophedon and $A^*$ to, respectively, (a) perform a search sweep in unknown terrain, and (b) optimally traverse the known terrain.

Using lawn mower sweep in urban environments provides a unique set of challenges. Covering occluded areas where buildings are densely constructed is one challenge that is not typically addressed by the standard two-dimensional implementations of the approach. The occlusion problem can be mitigated by having agents adjust their height to fly over the buildings that create such occlusion. In other words, connections between different regions of ground-level search are possible by having agents perform a temporary change in altitude by flying over buildings.
1) Lawn mower sweep: formalization and problem statement: This subsection will formulate Lawn Mower algorithm mathematically.

Given, a space $\chi$. Defining $\chi_{obs}$ and $\chi_{free}$, as obstacle space and free space respectively, such that $\chi_{obs}, \chi_{free} \subset \chi$. A path $\rho$ is a function $\rho : [0, 1] \rightarrow \chi_{free}$ which maps the path to a point in the free space. Therefore we can define

$$\rho = \bigcup_{i=1}^{n} \{\rho_i\}$$

where $n$ is number of agents. Now let $\chi_{search}$ be the search space such that $\chi_{search} \subseteq (\chi_{free} \cap \text{groundplane})$. Assuming agent’s camera sensor always facing downwards, the volume of space that is sensed by the sensor forms a conical shape below the agent. Let $K(X, d_{\text{max}})$ is the set of points in a downward facing camera at $X$ and extended below $X$ till $d_{\text{max}}$ forming a 3D conical space. Here $d_{\text{max}}$ might be inferred as the valid sensing distance. The multi-robot coverage problem is formally defined as follows:

Multi-robot coverage problem:
Find $\rho$ such that $\chi_{\text{search}} \setminus \chi_{\text{swept}} = \emptyset$, where $\chi_{\text{swept}}$ is given by:

$$\chi_{\text{swept}} = \{X | X \in \bigcup_{\in \rho} K(x, d_{\text{max}})\}$$

and where $\chi_{\text{search}} \subseteq \chi_{\text{swept}}$.

2) Global Solution: Our global lawnmower solution experiments take the quadrorts’ sensor model and a resolution measure for the sensor output into account and precompute the waypoints (as seen in Fig 3) over the environment. We use a simple approach that extends a standard lawn mower sweep from two-dimensions to three dimensions by determining the optimal flying altitude to obtain a good resolution in the output and the necessary altitude changes to avoid collisions with building along the sweep path, and then augmenting the height components of the path accordingly. It is possible to use more sophisticated methods, which calculate a sweep path that minimizes the number of altitude changes required over the entire search. We choose a simple method so that we can easily compute such waypoints for complex urban scenes with a large number of buildings.

IV. LOCAL COLLISION AVOIDANCE WITH COVERAGE CONSTRAINTS

In this section, we provide a brief introduction to ORCA and then define our local collision avoidance method with coverage constraints.

A. Symbols and Notations

A list of symbols and their definitions used in the formulation of ORCA [7] is shown in Table I. A brief explanation for ORCA using these symbols is provided in Section IV.B.

B. Optimal Reciprocal Collision Avoidance

1) Velocity Obstacles: Consider a pair of agents A and B. The velocity obstacle (VO) for the pair is defined as the set of all relative velocities of A with respect to B that will result in a collision between A and B before time $\tau$. It is formally defined as,

$$VO_A^B = \{v | \exists t \in [0, \tau] : tv \in D(p_B - p_A, r_A + r_B)\}$$

where $D(p, r)$ is a disk centered at $p$ with radius $r$ which is defined as:

$$D(p, r) = \{q | ||q - p|| < r\}.$$  

In 2-D, ORCA represents all agents as discs which is extended to a sphere representation in 3-D. The velocity obstacle for such a case can be geometrically interpreted as a truncated cone with its apex at the origin of the velocity space and its legs tangent to the disc of radius $r_A + r_B$, centered at $p_B - p_A$. We can infer from the formulation of VO that for any velocity $v_A \in V_B$, if $v_A \not\in VO_A^B \oplus V_B$, then A and B are guaranteed to avoid collision for at least time $\tau$. The ORCA algorithm calculates optimal collision avoidance velocity sets from which new $v_A$ and $v_B$ can be chosen.
2) Optimizing the Velocity Computation: ORCA tries to optimize the new chosen velocities to be close to the agents’ optimal velocities. An agent’s optimal velocity can be one of three values: zero, preferred velocity $v_{A}^{\text{pref}}$, or the agent’s current velocity. The objective is to calculate the velocity sets $\text{ORCA}_{A,B}^\tau$ for A and $\text{ORCA}_{B,A}^\tau$ for B. These sets contain more velocities close to $v_{A}^{\text{opt}}$ and $v_{B}^{\text{opt}}$, respectively, than any other pair of sets of reciprocally collision-avoiding velocities.

If suppose $v_{A}^{\text{opt}} - v_{B}^{\text{opt}} \in VO_{A/B}^\tau$, then the two agents may collide before time $\tau$. To achieve collision avoidance with the least amount of effort, ORCA finds a relative velocity on the boundary of $VO_{A/B}^\tau$ that is closest to $v_{A}^{\text{opt}} - v_{B}^{\text{opt}}$.

Reactive Obstacles: ORCA lets each agent take half of the responsibility for changing their velocities such that their relative velocity lies outside $VO_{A/B}^\tau$. Based on this formulation, the ORCA velocity set for A is given as:

$$\text{ORCA}_{A,B}^\tau = \left\{ v \mid \left( v - \left( v_{A}^{\text{opt}} + \frac{1}{2} u \right) \right) \cdot \hat{n} \geq 0 \right\},$$

where $u$ is the vector from $v_{A}^{\text{opt}} - v_{B}^{\text{opt}}$ to the VO boundary point and $\hat{n}$ is the outward normal at point $(v_{A}^{\text{opt}} - v_{B}^{\text{opt}})$ + $u$ on the boundary of $VO_{A/B}^\tau$.

Non-reactive Obstacles: The dynamic obstacles that cannot change their velocities to avoid a collision are referred to as non-reactive dynamic obstacles. Since such obstacles take no responsibility to avoid a collision, the ORCA velocity set for an agent in such cases is given by:

$$\text{ORCA}_{A,B}^\tau = \left\{ v \mid \left( v - v_{A}^{\text{opt}} + u \right) \cdot \hat{n} \geq 0 \right\},$$

The same expression also applies to static obstacles.

As an extension for a swarm of ‘n’ agents, the permitted velocity set for agent A denoted as $\text{ORCA}_{A}^\tau$ is the intersection of all half-planes induced by all agent B’s. It is defined as:

$$\text{ORCA}_{A}^\tau = \bigcap_{B \neq A} \text{ORCA}_{A,B}^\tau$$

The new velocity that agent A chooses would be the one that is closest to its preferred velocity $v_{A}^{\text{pref}}$ from $\text{ORCA}_{A}^\tau$.

$$v_{A}^{\text{new}} = \arg\min_{v \in \text{ORCA}_{A}^\tau} || v - v_{A}^{\text{pref}} ||$$

The new velocity is computed using linear programming. ORCA is a simple algorithm with low computational overhead. Therefore, it can be run online on each agent and is scalable to a large number of swarm agents. With its potential to be implemented on a large swarm, ORCA needs to be augmented with additional constraints to handle the coverage loss that arises from agents encountering a large number of dynamic obstacles, and a suitable static obstacle avoidance scheme.

C. Collision Avoidance with Static Obstacles

ORCA models static obstacles using line segments in 2-D and using planes in 3-D. Such a model is inefficient and time consuming when used in dense 3-D urban environments. The lawn mower strategy computes the global coverage path while accounting for the static obstacles in the environment. In dense urban scenarios, the quadrotors might be required to maneuver close to buildings and any deviation from the global path during dynamic obstacle avoidance may cause them to collide with the buildings (scenario represented in Fig. 5). To prevent such collisions, and to overcome ORCA’s limitation, our method accounts for the static obstacles in the ORCA formulation (described in section IV.B) using proximity queries as suggest in [27].

Each quadrotor continuously monitors its surroundings and computes its proximity to static obstacles ($d$ in Fig[2], like the buildings in an urban scene. The quadrotor’s collision avoidance model is taken as a sphere with radius $r$. The closest point on the static obstacle is considered as a point obstacle and the Minkowski sum is calculated considering a small positive value for the radius of the closest point ($\varepsilon$) and the quadrotor sphere’s radius ($r$).

$$R = r \oplus \varepsilon$$

We compute the Velocity Obstacle using the Minkowski sum as shown in Figure 4. We use the ORCA formulation for non-reactive obstacles as described in Section IV.B to select a new velocity for the agent that is closest to its optimal velocity and lies outside the VO. Since this method considers only the closest obstacle point, it can be easily incorporated in a physical quadrotor system using simple depth sensors.

Selecting Preferred Velocity: In ORCA, the agent’s preferred velocity $v_{A}^{\text{pref}}$ is directed towards the next waypoint. In certain cases, with the deviation caused by collision avoidance, this $v_{A}^{\text{pref}}$ can be directed through the buildings. The static obstacle avoidance described in Section IV.C will prevent collision with the buildings. However, since we only consider the closest points as obstacles, the agent may reach a deadlock trying to follow this preferred velocity to the next waypoint. To prevent such deadlocks, we update the $v_{A}^{\text{pref}}$ as being directed to the closest point along the line segment connecting the waypoint that the agent visited before the deviation, and next waypoint, the waypoint that the agent would have visited if not for the deviation.

D. Dynamics Constraints

ORCA assumes that agents can modify their velocities instantaneously. Hence, some fundamental dynamics constraints need to be considered before calculating new velocities for quadrotors. These dynamics constraints can be limits on the maximum velocity and acceleration of a quadrotor, which in turn translates to constraints on the roll, pitch and yaw angles. Quadrotors in the swarm may destabilize when there is a large change in their velocities, which results in large pitch or roll angles that might topple the quadrotors.
To prevent such scenarios, we fix the maximum acceleration of the quadrotors and eliminate any velocity that does not obey the following constraint:

\[ v_{\text{new}} < v_{\text{current}} + a_{\text{max}} \cdot t \quad (14) \]

This formulation helps keep the collision-free velocity space convex, thus ensuring fast computation time. The search for appropriate velocities can be performed using acceleration velocity obstacles [37].

**E. Modeling Uncertainty**

We overcome ORCA’s requirement for perfect sensing by using Kalman filtering to handle noise in the position and velocity of the swarm agents and obstacles. We use a Kalman filter to estimate the mean and covariance matrix of the positions and velocities of all moving entities. We then use the square root of the largest eigenvalue of an entity’s covariance matrix as a measure to increase the radius of the bounding sphere of that entity. Thus, as the uncertainty in sensing increases, each agent takes a more conservative approach to avoid collisions. We observed that when noise was introduced in the state parameters of the agents and dynamic obstacles, the inclusion of Kalman filtering resulted in collision-free trajectories for the agents.

**F. Dynamic Collision Avoidance with Coverage Constraints**

We first derive an objective function that needs to be maximized based on the overlap area that was described in section III.A.

In the absence of dynamic obstacles, all quadrotors would fly between one waypoint to the next with their preferred velocities at the optimal altitude used by the lawnmower strategy. As mentioned previously, we consider that each quadrotor covers a square area at any instant of time. Therefore, the area covered by a quadrotor at any time instant \( t \in [0, \tau] \) can be given as:

\[
\text{area}_{\text{pref}} = \text{Area}(x, y, s, t) = \text{Area of square of side } s \text{ centered at } (x, y) \quad (15)
\]

where, \( \tau \) is the time horizon over which we calculate the area, and

\[
x = v_{\text{pref}x} \cdot t \quad (16)
\]

\[
y = v_{\text{pref}y} \cdot t \quad (17)
\]

\[
s = (h - v_{\text{pref}z} \cdot t) \cdot \frac{\tan \theta}{\sqrt{2}} \quad (18)
\]

where \( h \) is the initial height at which the quadrotor was flying at \( t = 0 \).

The area covered over a time horizon \( \tau \) for which an agent’s velocity is constant can be computed as:

\[
\bigcup_{\tau} \text{Area}(x, y, s, t) \quad (19)
\]

We sample the time horizon \( \tau \) into smaller time steps \( t \) and forward simulate the velocity \( v_{\text{new}} \) to calculate the area that would be covered if \( v_{\text{new}} \) is chosen as the new velocity.

When dynamic obstacles are present, the quadrotors choose a new velocity \( v_{\text{new}} \) according to (12), and \( x, y \) and \( s \) in equation (15) would be with respect to \( v_{\text{new}} \).

\[
\text{area}_{\text{new}} = \bigcup_{\tau} \text{Area}(v_{\text{new}x} \cdot t, v_{\text{new}y} \cdot t,
\left((h - v_{\text{new}z} \cdot t) \cdot \frac{\tan \theta}{\sqrt{2}}\right) \quad (20)
\]

\]
The above equation is obtained as follows. The X and Y coordinates of the center of coverage squares are obtained from the product of the X and Y components of \( v_{\text{new}} \) and \( t \). The product of \( v_{\text{new}} \) and \( t \) gives us the change in the flying height or altitude of the quadrotor.

\[
\Delta h = v_{\text{new}} \cdot t \\
h_{\text{new}} = h - \Delta h = h - v_{\text{new}} \cdot t \\
s_{\text{new}} = h_{\text{new}} \cdot \tan \theta \sqrt{2}
\]

The conformity to the global optimal coverage path automatically results in minimizing the loss of the coverage area. Conformity increases with increase in overlap between the areas in equations (15) and (20), while also limiting the GSD below an upper threshold.

Therefore, our objective function becomes:

\[
\max_{GSD \leq GSD_{\text{max}}} \left( \text{area}_{\text{overlap}} \right) = \max_{GSD \leq GSD_{\text{max}}} \left( \text{area}_{\text{pref}} \cap \text{area}_{\text{new}} \right) (21)
\]

A collision avoidance scenario is shown in Fig. 6, and the corresponding coverage area for different states of the quadrotor are shown in Fig. 7.

Overlap area can be computed by methods such as clipping, but performing clipping in each time-step for each agent would be computationally expensive. We avoid the above issue using a look up table with pre-computed values for overlap area as explained in the next section.

1) Constructing the Look Up Table: Let us consider the unit vector \((1,0,0)\) and call it \( v_{\text{unit}} \). This unit vector corresponds to the forward direction for the quadrotor in the quadrotor’s frame of reference and can be transformed to any preferred velocity vector by using the standard 3x3 rotational transformation matrix \( (R) \) as,

\[
v_{\text{pref}} = R_{\text{trans}} \cdot v_{\text{unit}}
\]

For a time horizon \( \tau \), we calculate \( \text{area}_{\text{unit}} \) corresponding to \( v_{\text{unit}} \) as given in (15) assuming a flying altitude of 5m. We rotate \( v_{\text{unit}} \) (about \( Y \) and \( Z \) axes) in all possible forward directions to evaluate how such rotations affect the overlap area with \( \text{area}_{\text{unit}} \).

We first apply a rotation to \( v_{\text{unit}} \) about the \( Z \)-axis as,

\[
v_{\alpha} = R_{Z\alpha} \cdot v_{\text{unit}},
\]

where \( R_{Z\alpha} \) is the rotation matrix denoting a rotation by angle \( \alpha \) about \( Z \)-axis which is given as:
| $\alpha$ (Yaw) | $\beta$ (pitch) | $v_{\alpha\beta}$ | $v_{\text{unit}} - v_{\alpha\beta}$ | area$_{\alpha\beta}$ | area$_{\text{unit}} \setminus$ area$_{\alpha\beta}$ |
|---------------|---------------|-----------------|-----------------------------|-----------------|----------------------------------|

TABLE II: The structure of the Look-Up Table.

\[
\begin{vmatrix}
\cos \alpha & -\sin \alpha & 0 \\
\sin \alpha & \cos \alpha & 0 \\
0 & 0 & 1
\end{vmatrix},
\]

$\alpha$ is the angle which governs whether the quadrotor will move right or left. We limit $\alpha$ to the range $[-90^\circ, 90^\circ]$. For each angle of rotation $\alpha$ about the Z-axis, we again apply a rotation about the Y-axis as,

\[
v_{\alpha\beta} = R_{\beta \beta} \cdot v_{\alpha} \tag{24}
\]

where $\beta$ is the angle which governs whether the quadrotor will move upward or downward, and $\beta \in [-90^\circ, 90^\circ]$. The matrix $R_{\beta \beta}$ is defined as,

\[
\begin{vmatrix}
\cos \beta & 0 & \sin \beta \\
0 & 1 & 0 \\
-\sin \beta & 0 & \cos \beta
\end{vmatrix}.
\]

Let us denote the coverage area for $v_{\alpha\beta}$ over time $\tau$ as area$_{\alpha\beta}$. The rotation of $v_{\text{unit}}$ is continued until all values of $\alpha$, and $\beta$ within their ranges are reached by increments of $1^\circ$, and $v_{\alpha\beta}$ and the overlap between area$_{\alpha\beta}$ and area$_{\text{unit}}$ are recorded into the LUT. The fully constructed LUT has 32,761 x 5 entries. The structure of the LUT is shown in Table II.

2) Selecting New Velocity from LUT: When the quadrotor has to change its velocity to avoid an obstacle, we first use ORCA to compute a new velocity $v_{\text{new}}$ as given in (12). The difference $\delta$ between $v_{\text{new}}$ and $v_{\text{pref}}$ is given as,

\[
\delta = |v_{\text{pref}} - v_{\text{new}}|. \tag{25}
\]

Next, we search the LUT for $v_{\alpha\beta}$ such that

\[
|v_{\text{unit}} - v_{\alpha\beta}| \geq \delta + \varepsilon \tag{26}
\]

where $\varepsilon$ is a small positive value. All the velocities which satisfy (26) are ranked based on their corresponding overlap area with area$_{\alpha\beta}$. The Z-component of $v_{\alpha\beta}$ is used to compute the altitude change in the time horizon, and the velocities whose altitude change fail the GSD constraint are neglected. The rotation $R_{\text{trans}}$ is applied to each shortlisted $v_{\alpha\beta}$ to transform it correspond to the preferred velocity orientation. Let us denote the transformed velocity as $v_{\alpha\beta}^{\text{trans}}$. It is computed as:

\[
v_{\alpha\beta}^{\text{trans}} = R_{\text{trans}} \cdot v_{\alpha\beta} \tag{27}
\]

Note that there will always be a velocity that is equal to or closest to $v_{\text{new}}$ given by ORCA in the list of $v_{\alpha\beta}^{\text{trans}}$. We check if each $v_{\alpha\beta}^{\text{trans}}$ is in ORCA for each agent considering the position and velocity of its neighbors. The collision avoiding velocity thus computed is the closest distance away from the preferred velocity. In contrast, LSwarm replaces the closest distance constraint in order to identify a velocity that gives better coverage overlap from the collision free velocity region. The LUT provides a pre-computed sample of all velocities and their coverage overlap to facilitate choosing the new velocity quickly rather than performing the overlap area computation in each iteration.

Algorithm 1 LSwarm

1: procedure LSwarm
2: for all quadrotors do
3: Get position and velocity of Agent A
4: Compute nearest neighbors set NN for A
5: for all agent B in NN do
6: Get position and velocity for B
7: Compute Velocity Obstacle
8: Construct ORCA Planes
9: Construct Collision Free Velocity Space
10: if A lies in the optimal path = True then
11: $V_{A}^{\text{pref}} \leftarrow$ normalize($WP_{A}^{\text{New}} - P_{A}$)
12: else
13: $V_{A}^{\text{pref}} \leftarrow$ normalize($CP_{A} - P_{A}$)
14: Compute $V_{A}^{\text{New}}$ using ORCA
15: Search LUT for velocities $\geq V_{A}^{\text{pref}} - V_{A}^{\text{New}}$
16: remove velocities with with fail GSD constraint
17: for every velocity $V$ in FV do
18: if $V$ satisfies ORCA planes then
19: $V_{A}^{\text{New}} \leftarrow V$
20: Publish Velocity to Agent

V. RESULTS AND PERFORMANCE

In this section, we describe our implementation and highlight LSwarm’s performance on different urban scenarios.

A. Implementation

LSwarm was implemented on an Intel Xeon w-2123 octocore processor at 3.6 GHz with 32 GB memory and GeForce GTX 1080 GPU. For simulating the swarm of quadrotors, we use ROS Kinetic, Gazebo 7.14.0, along with the PX4 Software In The Loop firmware.

1) LawnMower Strategy: LSwarm can handle any arbitrary 3D model. X-Y values of the global waypoints are computed using an occupancy grid constructed from the 2D footprint of the environment, while the Z values vary based on the building heights (Section III.C).
just acts as a placeholder. Any global coverage strategy that urban scenes. In our implementation, the lawnmower strategy that the lawnmower strategy can easily be scaled to larger 2 \sim \text{compute waypoints is proportional to the model size (i.e. area).}

C. Performance Evaluation

Suburban situations.
Sparse model (0.003 buildings/meter.sq) represents rural or represents the dense metropolitan scenario, and the Low scenarios that the aerial surveillance system may face. For example, the High Dense model (0.022 buildings/meter.sq) building densities (buildings/sq.m) capture the real world situations. Hence, we test ORCA with the lawnmower strategy (which accounts for static obstacles) to test if it would be adequate to avoid collision with buildings.

B. Benchmarks

Table III shows the dimensions and complexity of the different urban models used to evaluate our method, and the time taken to compute the global lawnmower waypoints. Three urban models are shown in Fig. 2. Fig. 1 shows the High-Dense urban model. The first word in the name of the models corresponds to the average height of the buildings (high or low) and the second word in the name signifies the density of buildings in the model.

The 4 benchmarks with varying building heights and building densities (buildings/sq.m) capture the real world scenarios that the aerial surveillance system may face. For example, the High Dense model (0.022 buildings/meter.sq) represents the dense metropolitan scenario, and the Low Sparse model (0.003 buildings/meter.sq) represents rural or suburban situations.

C. Performance Evaluation

1) Lawnmower Path Generation: The time taken to compute waypoints is proportional to the model size (i.e. area). For a model of area \sim 6000 \text{ m}^2, the lawnmower strategy took 2.5 seconds to generate the waypoints. From Fig. 8 we infer that the lawnmower strategy can easily be scaled to larger urban scenes. In our implementation, the lawnmower strategy just acts as a placeholder. Any global coverage strategy that provides a set of waypoints for each agent, and has similar computation time as the lawnmower strategy, can be used instead.

2) Local Collision Avoidance: ORCA is implemented using the RVO2-3D [26] library. The RVO2-3D library makes use of multiple cores for decentralized collision avoidance among multiple agents. In LSwarm, we compute the Euclidean separation distance for static collision avoidance using the Proximity Query Package (PQP) library [38]. We do not include any static obstacle avoidance as a part of ORCA in our comparisons. ORCA models static obstacles as line segments or planes, which is impractical for real world scenarios. Hence, we test ORCA with the lawnmower strategy (which accounts for static obstacles) to test if it would be adequate to avoid collision with buildings.

B. Benchmarks

Table III shows the dimensions and complexity of the different urban models used to evaluate our method, and the time taken to compute the global lawnmower waypoints. Three urban models are shown in Fig. 2. Fig. 1 shows the High-Dense urban model. The first word in the name of the models corresponds to the average height of the buildings (high or low) and the second word in the name signifies the density of buildings in the model.

The 4 benchmarks with varying building heights and building densities (buildings/sq.m) capture the real world scenarios that the aerial surveillance system may face. For example, the High Dense model (0.022 buildings/meter.sq) represents the dense metropolitan scenario, and the Low Sparse model (0.003 buildings/meter.sq) represents rural or suburban situations.

C. Performance Evaluation

1) Lawnmower Path Generation: The time taken to compute waypoints is proportional to the model size (i.e. area). For a model of area \sim 6000 \text{ m}^2, the lawnmower strategy took 2.5 seconds to generate the waypoints. From Fig. 8 we infer that the lawnmower strategy can easily be scaled to larger urban scenes. In our implementation, the lawnmower strategy just acts as a placeholder. Any global coverage strategy that provides a set of waypoints for each agent, and has similar computation time as the lawnmower strategy, can be used instead.

2) Local Collision Avoidance: ORCA is implemented using the RVO2-3D [26] library. The RVO2-3D library makes use of multiple cores for decentralized collision avoidance among multiple agents. In LSwarm, we compute the Euclidean separation distance for static collision avoidance using the Proximity Query Package (PQP) library [38]. We do not include any static obstacle avoidance as a part of ORCA in our comparisons. ORCA models static obstacles as line segments or planes, which is impractical for real world scenarios. Hence, we test ORCA with the lawnmower strategy (which accounts for static obstacles) to test if it would be adequate to avoid collision with buildings.

B. Benchmarks

Table III shows the dimensions and complexity of the different urban models used to evaluate our method, and the time taken to compute the global lawnmower waypoints. Three urban models are shown in Fig. 2. Fig. 1 shows the High-Dense urban model. The first word in the name of the models corresponds to the average height of the buildings (high or low) and the second word in the name signifies the density of buildings in the model.

The 4 benchmarks with varying building heights and building densities (buildings/sq.m) capture the real world scenarios that the aerial surveillance system may face. For example, the High Dense model (0.022 buildings/meter.sq) represents the dense metropolitan scenario, and the Low Sparse model (0.003 buildings/meter.sq) represents rural or suburban situations.

C. Performance Evaluation

1) Lawnmower Path Generation: The time taken to compute waypoints is proportional to the model size (i.e. area). For a model of area \sim 6000 \text{ m}^2, the lawnmower strategy took 2.5 seconds to generate the waypoints. From Fig. 8 we infer that the lawnmower strategy can easily be scaled to larger urban scenes. In our implementation, the lawnmower strategy just acts as a placeholder. Any global coverage strategy that provides a set of waypoints for each agent, and has similar computation time as the lawnmower strategy, can be used instead.

2) Local Collision Avoidance: ORCA is implemented using the RVO2-3D [26] library. The RVO2-3D library makes use of multiple cores for decentralized collision avoidance among multiple agents. In LSwarm, we compute the Euclidean separation distance for static collision avoidance using the Proximity Query Package (PQP) library [38]. We do not include any static obstacle avoidance as a part of ORCA in our comparisons. ORCA models static obstacles as line segments or planes, which is impractical for real world scenarios. Hence, we test ORCA with the lawnmower strategy (which accounts for static obstacles) to test if it would be adequate to avoid collision with buildings.

B. Benchmarks

Table III shows the dimensions and complexity of the different urban models used to evaluate our method, and the time taken to compute the global lawnmower waypoints. Three urban models are shown in Fig. 2. Fig. 1 shows the High-Dense urban model. The first word in the name of the models corresponds to the average height of the buildings (high or low) and the second word in the name signifies the density of buildings in the model.

The 4 benchmarks with varying building heights and building densities (buildings/sq.m) capture the real world scenarios that the aerial surveillance system may face. For example, the High Dense model (0.022 buildings/meter.sq) represents the dense metropolitan scenario, and the Low Sparse model (0.003 buildings/meter.sq) represents rural or suburban situations.

C. Performance Evaluation

1) Lawnmower Path Generation: The time taken to compute waypoints is proportional to the model size (i.e. area). For a model of area \sim 6000 \text{ m}^2, the lawnmower strategy took 2.5 seconds to generate the waypoints. From Fig. 8 we infer that the lawnmower strategy can easily be scaled to larger urban scenes. In our implementation, the lawnmower strategy just acts as a placeholder. Any global coverage strategy that provides a set of waypoints for each agent, and has similar computation time as the lawnmower strategy, can be used instead.

2) Local Collision Avoidance: ORCA is implemented using the RVO2-3D [26] library. The RVO2-3D library makes use of multiple cores for decentralized collision avoidance among multiple agents. In LSwarm, we compute the Euclidean separation distance for static collision avoidance using the Proximity Query Package (PQP) library [38]. We do not include any static obstacle avoidance as a part of ORCA in our comparisons. ORCA models static obstacles as line segments or planes, which is impractical for real world scenarios. Hence, we test ORCA with the lawnmower strategy (which accounts for static obstacles) to test if it would be adequate to avoid collision with buildings.

B. Benchmarks

Table III shows the dimensions and complexity of the different urban models used to evaluate our method, and the time taken to compute the global lawnmower waypoints. Three urban models are shown in Fig. 2. Fig. 1 shows the High-Dense urban model. The first word in the name of the models corresponds to the average height of the buildings (high or low) and the second word in the name signifies the density of buildings in the model.

The 4 benchmarks with varying building heights and building densities (buildings/sq.m) capture the real world scenarios that the aerial surveillance system may face. For example, the High Dense model (0.022 buildings/meter.sq) represents the dense metropolitan scenario, and the Low Sparse model (0.003 buildings/meter.sq) represents rural or suburban situations.

C. Performance Evaluation

1) Lawnmower Path Generation: The time taken to compute waypoints is proportional to the model size (i.e. area). For a model of area \sim 6000 \text{ m}^2, the lawnmower strategy took 2.5 seconds to generate the waypoints. From Fig. 8 we infer that the lawnmower strategy can easily be scaled to larger urban scenes. In our implementation, the lawnmower strategy just acts as a placeholder. Any global coverage strategy that provides a set of waypoints for each agent, and has similar computation time as the lawnmower strategy, can be used instead.
(a) **High Sparse**: Sparsely populated urban model with 16 skyscrapers

(b) **Low Dense**: Sparsely populated model with 23 small buildings

(c) **Low Dense**: Densely packed model with 79 buildings of low height

Fig. 9: Benchmark urban environments

| Environment Model      | Dimension (in meters) | Number of Buildings | Global Path Calculation Time (ms) | Collisions with buildings |
|------------------------|-----------------------|---------------------|----------------------------------|--------------------------|
| High Dense             | 50.96 x 39.33 x 29.50 | 27                  | 753                              | 4 nil                    |
| High Sparse            | 56.25 x 53.03 x 14.25 | 16                  | 1027                             | 2 nil                    |
| Low Dense              | 64.26 x 53.80 x 12.5  | 79                  | 1186                             | nil nil                  |
| Low Sparse             | 96.67 x 62.92 x 7.2   | 23                  | 2585                             | nil nil                  |

**TABLE III**: We present the complexity of different urban models used to evaluate our algorithm. Moreover, we highlight the time taken to compute the lawnmower waypoints and the number of collision with buildings when using ORCA and LSwarm.

| Number of Dynamic Obstacles | ORCA | LSwarm |
|-----------------------------|------|--------|
|                             | Overlap Ratio (with required Resolution) | Overlap Ratio (with required Resolution) |
|                             | 1m Coverage Radius | 3m Coverage Radius | 1m Coverage Radius | 3m Coverage Radius |
| **Approaching from all directions** | | | | |
| 10                          | 0.7210 | 0.9235 | 0.8977 | 0.9355 |
| 25                          | 0.6594 | 0.7040 | 0.8715 | 0.9197 |
| 40                          | 0.5916 | 0.6331 | 0.6084 | 0.8875 |
| **Approaching from Left to Right** | | | | |
| 10                          | 0.7762 | 0.8879 | 0.9308 | 0.9448 |
| 20                          | 0.3201 | 0.5374 | 0.8923 | 0.9379 |

**TABLE IV**: We present the complexity of different urban models used to evaluate our algorithm. Moreover, we highlight the time taken to compute the lawnmower waypoints using our global solution.

VI. CONCLUSION, LIMITATIONS, AND FUTURE WORK

We present an efficient method for local collision avoidance under coverage constraints for a large number of agents in a swarm. We pose the problem using an optimization formulation based on coverage constraints. In order to solve the non-linear optimization problem, we use a precomputed lookup table to compute a solution that provides guarantees on collision-free trajectories as well as the coverage area. The running time of our method scales linearly with the number of swarm agents and takes a few milliseconds for tens of swarms covering a large urban scene.

Our approach has some limitations. Currently, we use a simple Kalman filter to address the uncertainties in localization of agents/obstacles whereas uncertainty in actuation is not considered. Furthermore, the lawnmower algorithm assumes an exact representation of the urban environment. Our solution to the optimization formulation is approximate and does not provide guarantees with respect to global optimality.

There are many avenues for future work. In addition to addressing these limitations, we would like to further evaluate our approach on complex urban models by coupling with other coverage strategies. It would be useful to develop more robust methods that can handle noise in the sensor measurements and account for uncertainty. Finally, we would like to accurately model the dynamics constraints of swarm agents (e.g. quadrotors).

REFERENCES

[1] Galceran, E., & Carreras, M. (2013). "A survey on coverage path planning for robotics". Robotics and Autonomous Systems, 61, 1258-1276.

[2] Julian, B. J., Angermann, M., Schwager, M., & Rus, D. (2012). "Distributed robotic sensor networks: An information-theoretic approach". The International Journal of Robotics Research, 31(10), 11341154.

[3] I. Saha, R. Ramaithitima, V. Kumar, G. J. Pappas and S. A. Seshia, "Implan: Scalable Incremental Motion Planning for Multi-Robot Systems," 2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS), Vienna, 2016, pp. 1-10.

[4] Saska, M., Vonsek, V., Chudoba, J. et al. J Intell Robot Syst (2016) 84: 469. https://doi.org/10.1007/s10846-016-0338-z

[5] Turpin, M., Michael, N., & Kumar, V. (2014). "Capt: Concurrent assignment and planning of trajectories for multiple robots". The International Journal of Robotics Research, 33(1), 98112.
Fig. 10: Trajectories followed by agents using LSwarm and ORCA in a scenario with 10 dynamic obstacles moving perpendicular to the motion of the agent. The red arrow shows the increase in an agent’s altitude when ORCA was used for dynamic obstacle avoidance.

Fig. 11: Running time of LSwarm algorithm and ORCA vs. number of agents on a multi CPU-core. We observe nearly linear growth for LSwarm indicating scalability with the number of agents. As expected, ORCA takes less time to compute collision avoidance velocities.

Fig. 12: This graph shows the ratio of overlap area with the preferred area vs the number of dynamic obstacles while considering GSD constraints. LSwarm provides a better overlap for a large number of dynamic obstacles over ORCA.

[6] P. Velagapudi, K. Sycara and P. Scerri, "Decentralized prioritized planning in large multirobot teams," 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, 2010, pp. 4603-4609.

[7] van den Berg J., Guy S.J., Lin M., Manocha D. (2011) Reciprocal n-Body Collision Avoidance. In: Pradalier C., Siegwart R., Hirzinger G. (eds) Robotics Research. Springer Tracts in Advanced Robotics, vol 70. Springer, Berlin, Heidelberg

[8] S. Chung, A. A. Paranjape, P. Dames, S. Shen and V. Kumar, "A Survey on Aerial Swarm Robotics," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 837-855, Aug. 2018.

[9] P. Dames, P. Tokekar, and V. Kumar, "Detecting, Localizing, and Tracking an Unknown Number of Moving Targets Using a Team of Mobile Robots, Springer Proceedings in Advanced Robotics Robotics Research, pp. 513-529, 2017.

[10] P. Dames, "Distributed multi-target search and tracking using the PHD filter," 2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), Los Angeles, CA, 2017, pp. 1-8. doi: 10.1109/MRS.2017.8250924

[11] S. N. Das and I. Saha, "Rhocop: Receding Horizon Multi-Robot Coverage," 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), Porto, 2018, pp. 174-185. doi: 10.1109/IC-CPS.2018.00025

[12] Mathewa, G., & Mezic, I. (2010). “Metrics for ergodicity and design of ergodic dynamics for multi-agent systems”.

[13] W. Haug, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme and N. Ayanian, "Trajectory Planning for Quadrotor Swarms," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 856-869, Aug. 2018.

[14] M. Hamer, L. Widmer and R. Dandrea, "Fast Generation of Collision-Free Trajectories for Robot Swarms Using GPU Acceleration," in IEEE Access, vol. 7, pp. 6679-6690, 2019.

[15] B. O. Koopman. 1956. “The Theory of Search. II. Target Detection”. Oper. Res. 4, 5 (October 1956), 503-531. DOI=http://dx.doi.org/10.1287/opre.4.5.503

[16] Choset H., Pignon P. (1998) Coverage Path Planning: The Boustrophedon Cellular Decomposition. In: Zelinsky A. (eds) Field and Service Robotics. Springer, London.

[17] Chan Sze Kong, New Ai Peng and I. Rekleitis, "Distributed coverage with multi-robot system," Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., Orlando, FL, 2006, pp. 2423-2429.

[18] Patrick Vincent and Izhak Rubin. 2004. A framework and analysis for cooperative search using UAV swarms. In Proceedings of the 2004 ACM symposium on Applied computing (SAC ’04). ACM, New York, NY, USA, 79-86. DOI: https://doi.org/10.1145/967900.967919

[19] Spires S.V., Goldsmith S.Y. (1998) Exhaustive geographic search with mobile robots along space-filling curves. In: Drogoiu A., Tambe M., Fukuda T. (eds) Collective Robotics. CRW 1998. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 1456. Springer, Berlin, Heidelberg

[20] de Lamadrid, J. G. (1994) "Avoidance of Obstacles With Unknown Trajectories: Locally Optimal Paths and Periodic Sensor Readings", The International Journal of Robotics Research, 13(6), pp. 496-507.

[21] C. Fulgenzi, A. Spalanzani and C. Laugier, "Dynamic Obstacle Avoidance in uncertain environment combining PVOs and Occupancy Grid," Proceedings 2007 IEEE International Conference on Robotics and Automation, Roma, 2007, pp. 1610-1616.

[22] Hsu, D., Kindel, R., Latombe, J.-C., & Rock, S. (2002). "Randomized Kinodynamic Motion Planning with Moving Obstacles". The International Journal of Robotics Research, 21(3), 233255.

[23] L. Martinez-Gomez and T. Fraichard, "Collision avoidance in dynamic environments: An ICS-based solution and its comparative evaluation," 2009 IEEE International Conference on Robotics and Automation, Kobe, 2009, pp. 100-105.

[24] Fiorini, P., & Shiller, Z. (1998). "Motion Planning in Dynamic Environments Using Velocity Obstacles". The International Journal of Robotics Research, 17(7), 760772.

[25] J. van den Berg, Ming Lin and D. Manocha, "Reciprocal Velocity Obstacles for real-time multi-agent navigation," 2008 IEEE International Conference on Robotics and Automation, Pasadena, CA, 2008, pp. 1928-1935.

[26] J. van der Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha.
2008. RVO2 Library: Reciprocal Collision Avoidance for Real-Time Multi-Agent Simulation. (May2008). http://gamma.cs.unc.edu/RVO2/

[27] D. Alejo, J. A. Cobano, G. Heredia and A. Ollero, "Optimal Reciprocal Collision Avoidance with mobile and static obstacles for multi-UAV systems," 2014 International Conference on Unmanned Aircraft Systems (ICUAS), Orlando, FL, 2014, pp. 1259-1266.

[28] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 6252-6259. doi: 10.1109/ICRA.2018.8461113

[29] Breitenmoser A., Martinoli A. (2016) On Combining Multi-robot Coverage and Reciprocal Collision Avoidance. In: Chong NY., Cho YJ. (eds) Distributed Autonomous Robotic Systems. Springer Tracts in Advanced Robotics, vol 112. Springer, Tokyo.

[30] Daniel Hennes, Daniel Claes, Wim Meeusen, and Karl Tuyls. Multi-robot collision avoidance with localization uncertainty. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, pages 147154. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[31] L. Meier, P. Tanskanen, F. Fraudorfer and M. Pollefeys, "PIXHAWK: A system for autonomous flight using onboard computer vision," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 2992-2997. doi: 10.1109/ICRA.2011.5980229

[32] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna and D. Manocha, "PRVO: Probabilistic Reciprocal Velocity Obstacle for multi robot navigation under uncertainty," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 1089-1096. doi: 10.1109/IROS.2017.8202279

[33] J. Snape, J. v. d. Berg, S. J. Guy and D. Manocha, "The Hybrid Reciprocal Velocity Obstacle," in IEEE Transactions on Robotics, vol. 27, no. 4, pp. 696-706, Aug. 2011. doi: 10.1109/TRO.2011.2120810

[34] Y. Kamel, J. Alonso-Mora, R. Siegwart and J. Nieto, "Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control," 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, 2017, pp. 236-243. doi: 10.1109/IROS.2017.8202163

[35] M. Schmittle et al., "Open UAV: A UAV Testbed for the CPS and Robotics Community," 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPSS), Porto, 2018, pp. 130-139.

[36] A. Niewmennikizaba, Hoang Huu Viet and TaeChoong Chung, "An online complete coverage algorithm for cleaning robots based on boustrophedon motions and A* search," 2012 8th International Conference on Information Science and Digital Content Technology (ICIDT2012), Jeju, 2012, pp. 401-405.

[37] J. van den Berg, J. Snape, S. J. Guy and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," 2011 IEEE International Conference on Robotics and Automation, Shanghai, 2011, pp. 3475-3482.

[38] E. Larsen, S. Gottschalk, M. C. Lin and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), San Francisco, CA, USA, 2000, pp. 3719-3726 vol.4. doi: 10.1109/ROBOT.2000.845311