Visual Coverage Optimization in Aerial Swarms with Local Sensing

Alexandre Hébert
Masters Student
alexandre.hebert@epfl.ch

Benjamin Jarvis
Supervisor
EPFL Laboratory of Intelligent Systems
benjamin.jarvis@epfl.ch

Dario Floreano
Director
EPFL Laboratory of Intelligent Systems
dario.floreano@epfl.ch

***Abstract*— Swarms are widely recognized as the next step in Unmanned Aerial Vehicle (UAV) technology, offering significant advantages in fields such as search and rescue and industrial plant inspection. These areas still benefit from having a human operator to supervise or control the swarm, especially for tasks like fault recognition during inspections. However, gathering and combining video streams from all drones to create an intuitive interface for swarm control is a complex task. To address this, we explore various methods for calculating the target viewing direction for each drone in the swarm, aiming to provide the best mapping for a pilot's point of view. Our approach relies solely on local sensing, using neighboring drones, to ensure scalability for high-density swarms and suitability for real-world applications. We developed a Python simulator to evaluate the proposed algorithms against factors such as swarm size, number of neighbors, and sensing noise. Finally, we demonstrated our method using a commercial simulator (Webots) and validated it with real hardware (Crazyflies).**

***Index terms*—Swarm, Drones, Viewing alogrithm, Coverage, FPV**

I. Introduction

As swarming in robotics becomes a popular approach for many different fields such as search and rescue [1], [2], inspection of industrial plants [3], [4] and mapping of inaccessible areas [5]. However, in a setting where swarms need to be controlled by a human pilot, we aim to provide a 360 degree viewing coverage to enhance the pilot's sensation of embodiment. Such coverage is possible by utilizing the distributed sensing capabilities of a swarm. Assuming no perfect knowledge of the swarm and utilizing only the distributed and local sensing capabilities of its members, finding such viewing directions for each drones to obtain maximum coverage is not a trivial task.

(a) (b) (c)

Figure 1: Pipeline overview: 9 drones swarm with (a) Simulator, (b) Webots and (c) Crazyflies

Furthermore, this specific field of research has not been tackled yet by the scientific community, which makes it even more difficult to find a starting point. However, estimating the pose of robots within a swarm (position + orientation) has already been done by others [6]–[9]. Indeed, relative localization is often used because of the non availability and/or the lack of accuracy compared to the inter-agent distance of global sensors such as GNSS positioning. Using inter-agent communication channels, such as UWB [10], or local state estimation, such as decentralized particle filter (DPF) [11] is often a reliable approach for localization. However, such communication between agents is not always possible. Vision based techniques with no communication mechanism [7] have also shown good performances with drone swarm in the presence of occlusions.

As a starting point, a geometric method is employed to develop basic algorithms for determining viewing directions (average and Outer). More advanced algorithms are derived by analyzing 3D point clouds, utilizing corresponding techniques to estimate the surface normals based on a subset of nearby points (convex hull and tangent plane) [12]–[15].

Results show that the proposed convex hull approach with adjacent edges offer the most stable and the best performance in terms of coverage against factors such as number of neighbors and sensing noise.

II. METHODS

In order to compare the four proposed viewing strategies, a consistent simulation environment is developed in python. This allows us to quickly integrate the defined neighborhood selection scheme as well as the viewing direction algorithms in the simulator. Then, multiple tests with varying parameters are executed to assess the performance and robustness of the different viewing strategies.

A. Python Simulation

A python simulator (GUI application) was developed in order to assess the different performances of the viewing metrics implemented. It was decided not to use the existing simulator *Swarmlab* [16] developed by the *LIS* lab because it was developed in Matlab, which is not directly compatible with the *Webots* and the *Crazyflie* environments, which are both mostly based on python/C++ and ROS, and it has limited access to external librairies.

This simulator was built with simplicity in mind: drones in the swarm are considered as point mass objects (fig. 2 (a) and (b)). Furthermore, the mass is set to 1 so that the acceleration becomes simply the resulting forces. A simple forward Euler integration scheme is used to update velocities and positions based on the acceleration.

This simulator is also built with scalability in mind: features can be added easily to perform other type of experiments. Thus, such parameters can be easily accessed and changed directly through the front end (fig. 2c). For this experiment, only the Olfati-Saber swarming algorithm is used (see [17] for in depth explanation), but other algorithms, such as Reynolds, can be easily added if needed. Another panel, fig. 2d, is customized for the proposed experiment on viewing metrics for drone swarms. It contains the choice of metric to use (see Section II.C) and the calculated statistics for the selected drone in the simulation.

The last feature of this simulator is the *autorun*. Provided with a *json* formatted configuration file (*sim_autorun.json*), the application can run, record and export automatically simulations defined in the file. Every app variable can be set with its respective name, making everything accessible to perform experiments with different sets of parameters.

B. Neighbor selection

As proposed in [7], four different neighbor selection algorithms are considered in this experiment: euclidian and topological (distance based), visual line of sight (vision based) and voronoi. These methods are well explained in [7] and thus will not be covered here.

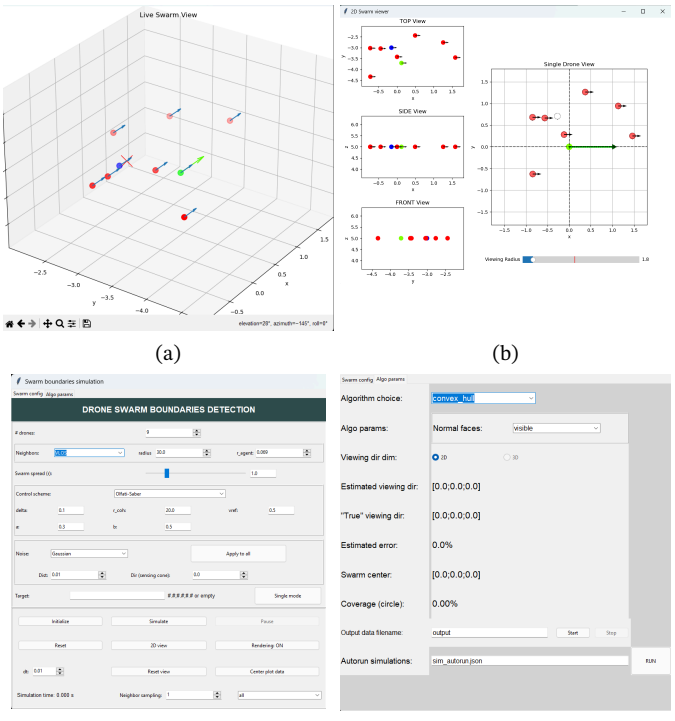


Figure 2: Main panels of the simulator: 3D and 2D views of the drone swarm (a & b). Simulation and algorithm parameters (c & d)

C. Viewing metric

Here, four different algorithm are proposed to estimate a suitable viewing (pointing) direction of a drone based on local sensing. These are the foundation of our experiment and different evaluation metrics will be used to compare their performances and find the ones that work the best.

Some definitions: A viewing direction is calculated for the currently selected drone i. The set of detected neighbors of drone i is defined by $\mathcal{N}$.

1) *Average:*

This is the simplest approach: drone i should point to the opposite of its neighbors centroid (fig. 3a).

$$o_i = \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}} p_j \tag{1}$$

$$\vec{d} = -\frac{p_i - o_i}{\|p_i - o_i\|} \tag{2}$$

(1) calculates the centroid of the neighborhood by averaging the estimated positions, p_j, of each neighbor. Subsequently, (2) provides a normalized viewing direction by pointing away from the centroid o_i.

2) *Tangent plane:*

Inspired from [12], the main idea is as follow: drone i should point away from the tangent plane expressing the most variance of its neighbors. This is similar to taking the normal of the surface spanning the manifold of its neighbors, but only limited to a 2D plane. As shown in [12], this is

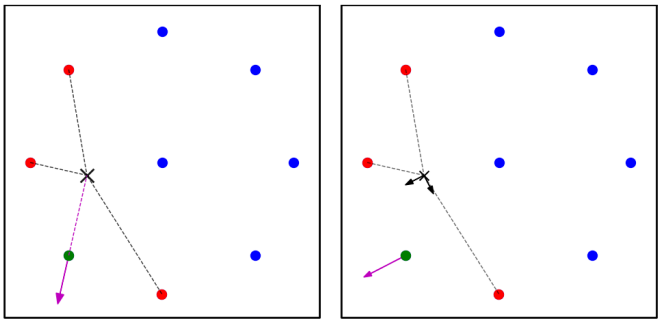
done by using the eigenvector expressing the least variance. Using (1) again, the centroid o_i is found.

$$\begin{aligned} & C = V\Lambda V^T \\ \text{with} \quad & C = \sum_{y \in \mathcal{N}(i)} (y - o_i) \otimes (y - o_i) \end{aligned} \tag{3}$$

Then (3) performs PCA decomposition to find eigenvectors and eigenvalues of the covariance matrix.

$$\begin{aligned} & \vec{d} = -\text{sign}((p_i - o_i) \cdot \vec{e_3})\vec{e_3} \\ & \vec{e_3} : \text{eigenvector with smallest } \lambda \end{aligned} \tag{4}$$

Finally, (4) makes sure drone i is pointing towards the outside by using the sign of the dot product w.r.t. the vector $(p_i - o_i)$.



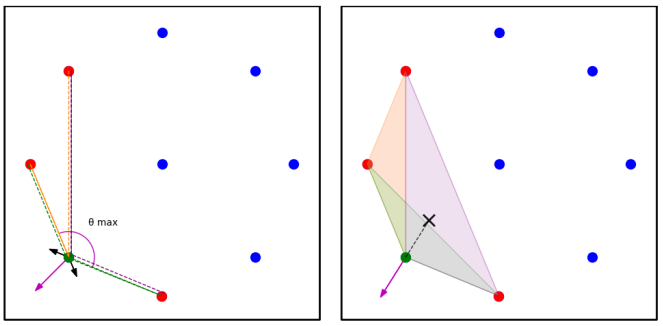
(a) (b)

Figure 3: 9 drones 2D swarm example for viewing metric
(a) pointing away from the average position of the neighbors (centroid). As one can see, this method is highly dependent on the neighborhood $\mathcal{N}$ of drone i. (b) Pointing in the direction that expresses the least variance (tangent plane)

3) *Outer:*

This method relies on creating different subsets of the neighborhood. An additional parameter, s, is used to specify the size of the subsets. Three possible cases are treated (all using neighbors in subset $\mathcal{S}$ and drone i):

1. $s = 2$: creating all combinations of 2 within $\mathcal{N}$, the viewing direction is given by the mean of the two adjacent edges spanning the biggest angle from drone i. See fig. 4a for detailed schematic.
2. $s = 3$: same principle, creating all combinations of 3 within $\mathcal{N}$, the triangle, formed by the subset $\mathcal{S}$, having the biggest area is used. The viewing direction is the opposite direction of the centroid of this triangle. Again, see fig. 4b for schematic.
3. $s \geq 4$: same thing as $s = 3$, but the convex hull is used to calculate the area or volume of the resulting shape. The viewing direction is given by pointing away from the centroid of the biggest shape.



(a) (b)

Figure 4: Viewing direction estimation with
(a) Outer 2: max angle between combinations of 2 neighbors
(b) Outer 3: pointing away from the centroid of the biggest triangle formed by combinations of 3 points within neighborhood

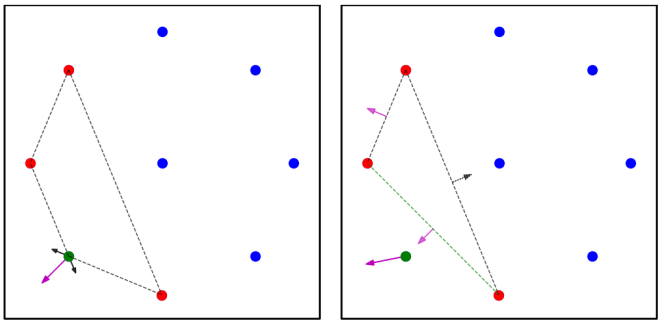
4) *Convex hull:*

Two variants are proposed, but they both use the convex hull of $\mathcal{N}$.

Adjacent: Including drone i in the convex set, the viewing direction is the mean of the normals of the adjacent simplices (edges or faces) of drone i.

Visible: Not including drone i in the convex set, only the normals of the visible simplices are retained to be used as a viewing direction. *Visible* is defined as follow:

$$v = \text{sign}\left(\hat{n}_{\text{closest}} \cdot \hat{n}_j\right) > 0 \quad \text{for } j \in \mathcal{E} \tag{5}$$



(a) (b)

Figure 5: Viewing direction estimation with convex hull
(a) using the mean of the normals of adjacent edges
(b) using the mean of all visible edges (arrows in purple) from i

As for the sensing noise, the model presented in [7] was used. It is composed of two components: the range noise, which is proportional to distance, and the bearing noise, which is constant over the field of view. Both follow a normal distribution with zero mean. The relative localization of a neighbor is thus:

$$\begin{aligned} & \hat{\boldsymbol{r}}_{\text{ij}} = \begin{pmatrix} \hat{d}_{\text{ij}} \cos\left(\hat{\beta}_{\text{ij}}\right) \\ \hat{d}_{\text{ij}} \sin\left(\hat{\beta}_{\text{ij}}\right) \end{pmatrix} \\ \text{with:} \quad & \hat{d}_{\text{ij}} : \text{noisy distance to neighbor j} \\ & \hat{\beta}_{\text{ij}} : \text{noisy heading to neighbor j} \end{aligned} \tag{6}$$

Refer to [7] for further details.

III. Results

The goal of this experiment was to compare the performance of the proposed viewing strategies against a set of parameters: the number of drones in the swarm, the number of neighbors, the neighbors selection and the sensing noise of the drone. For the majority of the experiments, a 2D swarm composed of 9 drones is used because of its simplicity to analyze and its shorter simulation time. To limit the number of runs needed, which grows rapidly due to the amount of parameters to verify, a first comparison between the Outer and the convex hull metric is done.

During each simulation, the swarm follows an infinity loop shaped trajectory with a $v_{\text{ref}} = 0.25$ m/s (fig. 13). All the fixed parameters across runs are presented in *Table 1*.

Table 1: Olfati-Saber parameters used for all simulations

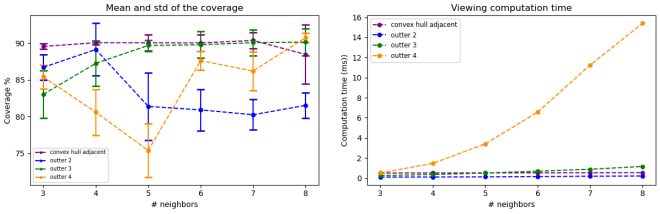
Parameter	Value
Drone count	9
Swarm spread	1.0 m
v_{ref}	0.25 m/s
r_{coh}	20 m
δ	0.1
a	0.3
b	0.5
c	$\frac{b-a}{2\sqrt{ab}} \approx 0.26$
Δt	0.01 s

Qualitatively, an optimal configuration for maximum area coverage is achieved when the drones' headings are equally distributed around a circle (considering a fixed camera mounted to a drone, thus only adjustable through the drone's yaw). Therefore, we defined a coverage metric based on the percentage of the field of view (FOV) of all drones projected onto an infinite radius circle or sphere. This approach simplifies the calculations by considering the swarm as the center of the circle or sphere, allowing us to focus on the FOV of each drone individually. Additionally, this method makes it easier to detect and account for overlaps between drones, which impacts the overall coverage values. All drones have equal FOVs, adjusted so that the best-case scenario results in 100% coverage.

The computational time required to calculate the desired viewing directions for each drone in the swarm (per iteration) is also used to assess the performance of the viewing metrics.

A. Outer vs Convex Hull metrics

First, the Outer and convex hull viewing metrics were compared due to their similarities. To compare the different algorithms, a topological neighbor selection scheme was used to have fixed amount of neighbors throughout the runs. The number of neighbors started at 3 because it is the minium amount required for the convex hull viewing metric. Only the convex hull and Outer 3 metrics were retained. Although Outer 2 showed good performance with a low number of neighbors, its standard deviation was higher than that of the convex hull and Outer 3 metrics (2-5% variation across runs), and its performance significantly declined with more than four neighbors (fig. 7a). Outer 4, on the other hand, not only performed poorly in terms of coverage but also exhibited exponentially increasing computational time as the number of neighbors grew (fig. 7b).



(a) (b)

Figure 6: Averaging over one loop of a figure 8 trajectory (a) Average coverage with standard deviation: Outer 2 is decreasing w.r.t neighbors. Outer 4 is varying a lot with the number of neighbors. Outer 3 shows stabilizes around 5 neighbors. Convex hull adjacent is stable across the range of neighbors. (b) Average computation time of the viewing metrics. Outer 4 is the only one increasing exponentially with the number of neighbors. It is thus considered as computationally intensive

B. Effects of swarm size

To analyze the effect of the number of drones in the swarm, six different swarm sizes were tested using the five selected viewing direction metrics: average, tangent plane, convex hull adjacent, convex hull visible, and Outer 3. Additionally, three neighbor selection schemes—visual line of sight (VLOS), Voronoi, and topological (with 3 neighbors)—were chosen to compare the effects of neighbor choice and count.

First, the number of detected neighbors using the Voronoi scheme remains almost constant at 5 for swarms larger than 10 drones, starting at 3 for a small swarm of 5 drones. In contrast, VLOS shows an increasing number of neighbors up to a swarm size of 20 drones, where it stabilizes at around 10 neighbors (fig. 9f). Topological remains at 3, as expected.

Second, the computational time required to calculate the viewing directions for all drones in the swarm increases linearly with the number of drones for each viewing metric (fig. 8). The average viewing metric scales not as expected with the number of drones, increasing rapidly with the topological selection, which has the lowest number of neighbors and should be the quickest to compute (fig. 8a). Running these simulation in parallels could potentially explain the cause, as the work load of the processors varies during each

trial, Further testing is needed to have a clear conclusion. The Outer 3 metric's computational time increases more rapidly (steeper slope) with both the number of drones and the number of detected neighbors, reaching up to 4 ms for VLOS compared to only 0.5 ms for the topological scheme in a swarm of 25 drones. This rapid increase is due to the need to compute all possible combinations of 3 neighbors. For a swarm of 30 drones, the convex hull adjacent metric requires double the computation time of the tangent plane, while Outer 3 requires five times more.

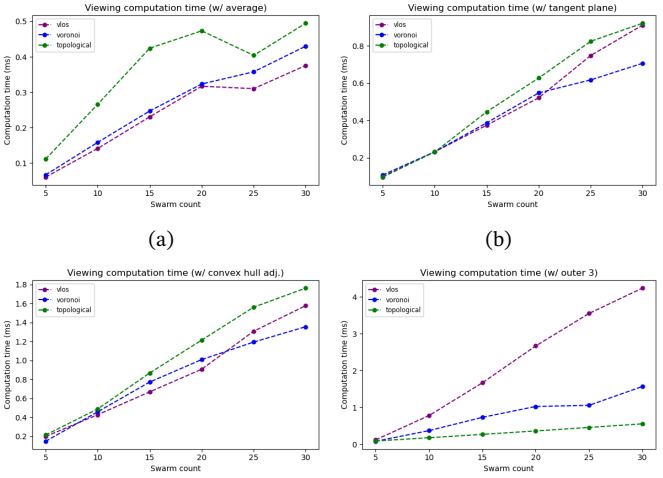


Figure 7: Effects of the swarm size on the computational time of the 5 different viewing metrics. (a) Average: fastest to calculate among all of the metrics. Shows unexpected behaviors for topological with 3 neighbors as it should be the fastest. Could be linked to computer resources usage during testing. (b and c) Tangent plane as well as both convex hull (adjacent + visible) show similar behaviors for the 3 selection algortihms, increasing linearly with number of drones. (d) Outer 3 is the most computational intensive algorithm due to the need of testing all combinations to find the triangle with the maximum area. It still grows linearly, but with much higher slope, reaching up to 4 ms (almost half the simulation step used).

In terms of coverage, all viewing metrics show a decrease from around 90% to 65-70% as the swarm size increases. The average metric with VLOS outperforms the voronoi and topological selection scheme (reaching over 90% with small variation with a 10 drones swarm), mostly due to the higher number of neighbors, since the gap is decreasing as the number of neighbors stabilizes (fig. 9a). The tangent plane metric experiences a significant drop in coverage (from 93% to 70%) when the swarm size increases from 5 to 10 drones, before stabilizing for the remainder of the trials (fig. 9b). The convex hull adjacent metric exhibits the smoothest decrease as the number of drones increases, with similar results across VLOS, Voronoi, and topological neighbor selection schemes, indicating its independence from the number of neighbors (fig. 9c). The convex hull visible metric shows similar trends, except for a performance drop (from 90% to 70%) when the swarm size increases from 10 to 15 drones with VLOS (fig. 9d). Finally, the Outer 3 metric displays behavior similar to the tangent plane metric (fig. 9e).

Based on these results, for the subsequent parts (Section II-C and Section III.D), the topological neighbor selection is used. This decision is due to the viewing metrics' reliance on the number of neighbors rather than their specific selection, and the simplicity of specifying a fixed number of neighbors throughout a run.

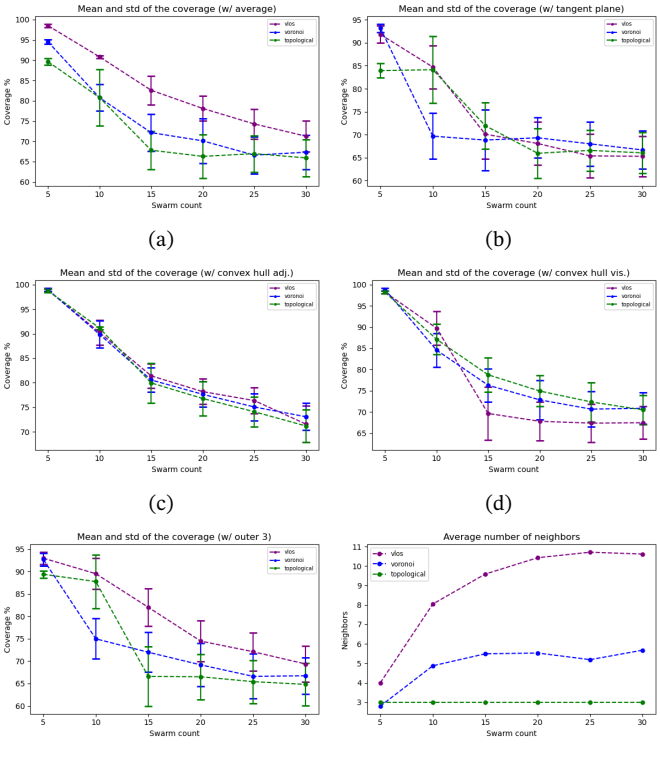
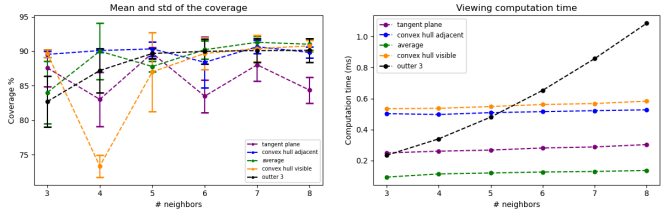


Figure 8: Effects of the swarm size on the performance of the 5 viewing metrics for VLOS, voronoi and topological neighbors selection. Neighbors count for voronoi is stable at around 5 for swarm bigger than 10 drones and VLOS stabilizes at around 10 neighbors after 15 drones (f). Convex Hull adjacent (c) shows the tightest and the best performance across the number of drones in the swarm as well as the neighbors selection method. Tangent plane and Outer 3 have similar behaviors (b and e). Average using VLOS outperforms topological and voronoi due to the higher number of neighbors (a).

C. *Effects of neighbors count*

To analyze the effect of the neighbors count on viewing metrics performance, topological neighbors selection was used again for the same reasons explained in Section III.A. The effect of the number of neighbors was tested against 5 different viewing metrics: convex hull adjacent and visible, tangent plane, average and Outer 3.



(a) (b)

Figure 9: Effects of the number of neighbors with topological selection on the performance of the 5 chosen viewing metrics. (a)

Interestingly enough, the computational time of all the viewing metrics except the Outer 3 is not affected by the neighbors growth (fig. 10b). The average metric has the least computation time, taking less than 0.1 ms per iteration. The tangent plane follows with more than double the time, and the convex hull reaches around 0.5 ms. The Outer 3 metric's computation time grows almost quadratically with respect to the number of neighbors, which can pose problems for real-time applications if needed (e.g., from 3 to 6 neighbors, the time increases from 0.2 to 0.7 ms).

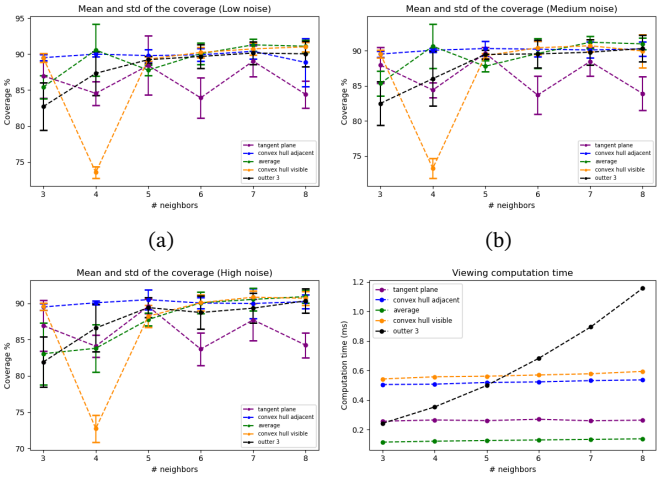
In terms of coverage, the convex hull adjacent metric shows excellent performance and remains one of the most stable across the entire range of neighbor counts, with an average of 90% coverage (fig. 10a). This is near the maximum possible coverage, given that the central drone does not significantly contribute. Indeed, if the eight boundary drones have perfectly distributed viewing directions, the central drone's field of view largely overlaps with theirs, adding little additional coverage. Additionally, it generally exhibits lower standard deviation than the other metrics, except when running with 6 neighbors. The convex hull visible metric is more sensitive to 'outlier' drones, showing a significant drop from 90% to less than 75% when there are 4 neighbors. This drop occurs because the center drone becomes part of the neighborhood, adding a visible edge that affects the viewing direction. The tangent plane metric varies significantly, with almost a +/- 10% variation each time a neighbor is added, ranging from 83% to 89%. The Outer 3 metric shows a smooth improvement with an increasing number of neighbors, reaching around 90% coverage at 5 neighbors. Finally, the average metric benefits significantly from including the center drone in its neighborhood, with an improvement of over 6% when increasing from 3 to 4 neighbors. All metrics exhibit similar performance with reasonable standard deviations at 5 neighbors, averaging around 89% coverage. However, the convex hull visible metric still shows larger variations than the others in this case.

D. Adding sensing noise

Sensing noise values were taken from [7] considering visual detection of neighbors. They were adjusted to create three different noise level: low (L), medium (M) and high (H) (Table 2).

Table 2: Values for the three noise profiles used in the simulation

Profile	σ_d (mm)	σ_β (rad)
Low (L)	10	0.01
Medium (M)	25	0.05
High (H)	50	0.10



(a) (b)

(c) (d)

Figure 10: Effects of noise on the performance of the five viewing metrics were evaluated. The noise does not significantly affect the performance of the viewing metrics, as all metrics exhibit similar coverage values across the runs. Notably, the convex hull visible metric shows an increase in variation (standard deviation) from 1% to 2% when transitioning from a low to high noise environment (a and c). However, this increase in variation is not observed in any other metrics. This stability can be attributed to the averaging performed across the figure-8 trajectory and the proportional controller for yaw, which acts as a low-pass filter.

Regarding computational time, the three noise levels exhibit the same behavior as the no-noise case with an increasing number of neighbors (fig. 12b). Thus, noise has no impact on the computational time of the viewing direction with any of the five algorithms.

In terms of coverage, the increasing amount of sensing noise does not significantly affect the performance of the metrics. However, the standard deviation of the convex hull visible metric increases with the amount of noise, rising from 1% to nearly 3% for a neighbor count of 5. The average metric shows the most variation when moving from low to high noise levels (from 90% down to 83% with 4 neighbors). It no longer shows a sharp improvement with the addition of the center drone, instead displaying a smoother improvement over the neighbor counts and lower performance with fewer neighbors, indicating higher sensitivity to positioning (82.5% in high noise environment compared to 86% with low and medium noise).

These results can be explained as follows: the drones use a proportional controller to adjust the yaw angle, which also functions as a low-pass filter. Since the sensing noise is sampled at every simulation step (0.01s), it can be modeled as high frequency and thus filtered by the proportional controller. Furthermore, the coverage is averaged over the figure-8 trajectory, which is also smoothing out the effect of noise.

IV. Conclusion

Five different viewing direction metrics were tested during this experiment: a simple average, a tangent plane normal estimation, the max area of a triangle formed by combinations of three neighbors (Outer 3), the adjacent edges of a convex hull, and the visible edges of a convex hull. Among these, the convex hull adjacent algorithm demonstrated the best coverage performance, achieving 90% even with a low number of neighbors and high noise levels. It also has a relatively low computational time that remains constant across varying neighbor counts and grows linearly with swarm size, making it suitable for real-time applications involving a large number of drones.

The conducted experiments represent only the beginning. Additional viewing metrics should be developed to address more complex swarm shapes and environments, such as the presence of obstacles. One potential approach is to use a concave hull of the swarm to identify significant drones that are not part of the previously used convex hull. Adding a tolerance factor could help exclude drones that are too close to the center and have occluded vision.

This work is the beginning of developing advanced viewing metrics and techniques to optimize swarm control and enhance immersive experiences, including the use of VR headsets for providing a high-end immersive experience for pilots. Achieving good vision coverage would significantly improve the pilot's experience and ease of controlling the swarm.

References

[1] J. Horyna *et al.*, "Decentralized swarms of unmanned aerial vehicles for search and rescue operations without explicit communication," *Auton Robot*, vol. 47, no. 1, pp. 77–93, Jan. 2023, doi: 10.1007/s10514-022-10066-5.

[2] J. R. Cooper, M. P. Vaughan, and B. N. Kelley, "Multi-Agent Search and Rescue Applied to a Swarm of Ground Vehicles," *AIAA SCITECH 2024 Forum*. American Institute of Aeronautics, Astronautics. doi: 10.2514/6.2024-2892.

[3] R. H. Jacobsen *et al.*, "Design of an Autonomous Cooperative Drone Swarm for Inspections of Safety Critical Infrastructure," *Applied Sciences*, vol. 13, no. 3, 2023, doi: 10.3390/app13031256.

[4] P. Tosato, D. Facinelli, M. Prada, L. Gemma, M. Rossi, and D. Brunelli, "An Autonomous Swarm of Drones for Industrial Gas Sensing Applications," in *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2019, pp. 1–6. doi: 10.1109/WoWMoM.2019.8793043.

[5] M. G. Miiller *et al.*, "Robust Visual-Inertial State Estimation with Multiple Odometries and Efficient Mapping on an MAV with Ultra-Wide FOV Stereo Vision," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3701–3708. doi: 10.1109/IROS.2018.8594117.

[6] H. Xu, L. Wang, Y. Zhang, K. Qiu, and S. Shen, "Decentralized Visual-Inertial-UWB Fusion for Relative State Estimation of Aerial Swarm," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8776–8782. doi: 10.1109/ICRA40945.2020.9196944.

[7] F. Schilling, E. Soria, and D. Floreano, "On the Scalability of Vision-Based Drone Swarms in the Presence of Occlusions," *IEEE Access*, vol. 10, pp. 28133–28146, 2022, doi: 10.1109/ACCESS.2022.3158758.

[8] F. Schilling, F. Schiano, and D. Floreano, "Vision-Based Drone Flocking in Outdoor Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2954–2961, 2021, doi: 10.1109/LRA.2021.3062298.

[9] M. Pavliv, F. Schiano, C. Reardon, D. Floreano, and G. Loianno, "Tracking and Relative Localization of Drone Swarms With a Vision-Based Headset," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1455–1462, 2021, doi: 10.1109/LRA.2021.3051565.

[10] S. Chen, D. Yin, and Y. Niu, "A Survey of Robot Swarms' Relative Localization Method," *Sensors (Basel)*, vol. 22, no. 12, p. 4424–4425, Jun. 2022, doi: 10.3390/s22124424.

[11] S. Zhang, K. Cokona, R. Pöhlmann, E. Staudinger, T. Wiedemann, and A. Dammann, "Cooperative Pose Estimation in a Robotic Swarm: Framework, Simulation and Experimental Results," in *2022 30th European Signal Processing Conference (EUSIPCO)*, Aug. 2022, pp. 987–991. doi: 10.23919/EUSIPCO55093.2022.9909666.

[12] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th annual conference*

on Computer graphics and interactive techniques, ACM, Jul. 1992, pp. 71–78. doi: 10.1145/133994.134011.

[13] L. Yang, Y. Li, X. Li, Z. Meng, and H. Luo, "Efficient plane extraction using normal estimation and RANSAC from 3D point cloud," *Computer Standards & Interfaces*, vol. 82, p. 103608–103609, Aug. 2022, doi: 10.1016/j.csi.2021.103608.

[14] C. Mineo, S. G. Pierce, and R. Summan, "Novel algorithms for 3D surface point cloud boundary detection and edge reconstruction," *Journal of Computational Design and Engineering*, vol. 6, no. 1, pp. 81–91, Jan. 2019, doi: 10.1016/j.jcde.2018.02.001.

[15] J. Calder, S. Park, and D. Slepčev, "Boundary Estimation from Point Clouds: Algorithms, Guarantees and Applications," *J Sci Comput*, vol. 92, no. 2, p. 56–57, Jul. 2022, doi: 10.1007/s10915-022-01894-9.

[16] E. Soria, F. Schiano, and D. Floreano, "SwarmLab: a Matlab Drone Swarm Simulator," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 8005–8011. doi: 10.1109/IROS45743.2020.9340854.

[17] E. Soria, "Model Predictive Control of Aerial Swarms," 2022. doi: 10.5075/epfl-thesis-9542.

V. Appendix

A. Figure-8 swarm trajectory

For all runs, the swarm was following 50 waypoints defined over a figure-8 shape (fig. 13).

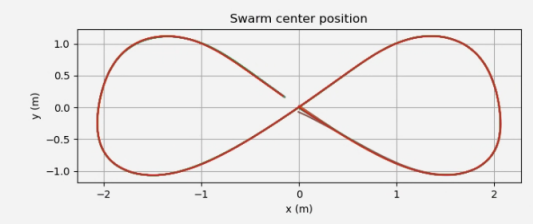
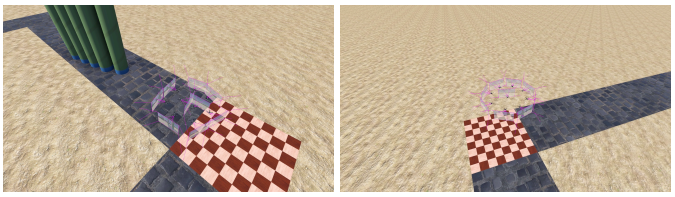


Figure 11: Infinity loop trajectory for running swarm simulations. This was obtained with 24 runs for comparing Outer with convex hull metrics

B. Webots simulation

Simulations were performed in the *Webots* simulator with more realistic drone dynamics, using exactly the algorithms presented in this report for estimating the best viewing direction for each drone based on the detected neighbors. A nice feature of *Webots* is the ability to show the camera frame, which makes it easier to qualitatively assess the performance of the different metrics. As a bonus, some experiments were done with added obstacles to see the behaviors of the viewing metrics with more complex swarm shapes.

1) *Without obstacles:*

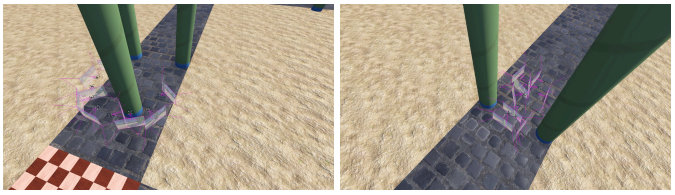


(a) (b)

Figure 12: (a) 9 drones swarm with 2 in the middle, using the average viewing direction metric and 2 neighbors with topological selection. Corners are not well covered because it is detecting one of the center drones as the second neighbor. (b) Convex hull visible metric applied to a 9 drones swarm, with one in the middle. A more uniform distribution of viewing direction is obtained.

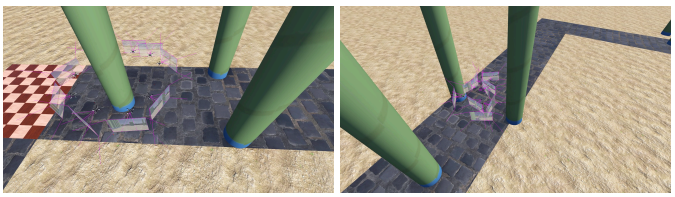
2) *With obstacles:*

Using the best performing viewing metric, convex hull with adjacent edges, two different scenarios were tested. First, the swarm surrounding an obstacle and second, the swarm passing in between two obstacles with narrow gap. The flaw observed with every run is that none of the drones has tendency to point towards obstacles, especially when the swarm is in a complex shape that requires something more than only the convex hull to provide better viewing direction for drones inside the convex hull, but still close to the edges of the swarm. That is why another approach, combining a concave hull to identify meaningful drones inside the convex hull, should be the next step for these experiments.



(a) (b)

Figure 13: 3 neighbors with topological selection method. (a) Around the obstacle, all the drone point towards the outside as there where no obstacles. (b) In between the obstacles, the swarm is more squished on a line. Drones are looking in every direction, covering a lot of the essentials such as the obstacles. However, the distribution of the viewing angles throughout the swarm is not great.



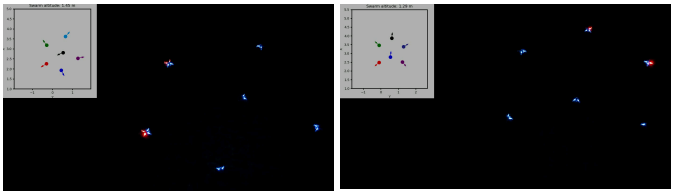
(a) (b)

Figure 14: Neighbor selection is done with VLOS. (a) Same behavior as fig. 15a. (b) In between obstacles, still pointing in every direction, but the distribution of viewing angles between neighboring drones could be improved.

C. Crazyflie experiment

The best performing metric, convex hull adjacent, and the simplest one, average, were tested on a swarm composed of 6 *Crazyflie* drones. Positions of all drones within the swarm were obtained with a *Mokap* camera system. No noise was

added during the experiment to the neighbors pose. Both tests used topological neighbors selection with 2 neighbors for the average metric and 3 neighbors for the convex hull (fig. 17). As expected, convex hull adjacent shows the best performance with a uniform distribution of viewing direction across drones at the boundaries. Average still offer good performances, but is very susceptible to the choice of neighbors.



(a) (b)

Figure 15: Swarm of 6 Crazyflie drones with topological neighbors selection. (a) With the average metric, decent results but having only two neighbors increase sensitivity with the chosen neighbors for small swarms. (b) Convex hull adjacent is performing better, having a more uniform distribution of drone's viewing direction along the boundaries of the swarm.

Three videos of the conducted experiments with the crazyflie hardware are available on *Youtube* [See playlist here]:

- Average viewing direction metric with two neighbors (topological)
- Average viewing direction metric with three neighbors (topological)
- Convex hull adjacent viewing direction metric with three neighbors (topological)

D. Simulation code

The source code of the developed python simulator is available on a Github repository.