SwarmRaft: Leveraging Consensus for Robust Drone Swarm Coordination in GNSS-Degraded Environments

Kapel Dev, Yash Madhwal, Member, IEEE Sofia Shevelo, Pavel Osinenko, Yury Yanovich, Member, IEEE

Abstract-Unmanned aerial vehicle (UAV) swarms are increasingly used in critical applications such as aerial mapping, environmental monitoring, and autonomous delivery. However, the reliability of these systems is highly dependent on uninterrupted access to the Global Navigation Satellite Systems (GNSS) signals, which can be disrupted in real-world scenarios due to interference, environmental conditions, or adversarial attacks, causing disorientation, collision risks, and mission failure. This paper proposes SwarmRaft, a blockchain-inspired positioning and consensus framework for maintaining coordination and data integrity in UAV swarms operating under GNSS-denied conditions. SwarmRaft leverages the Raft consensus algorithm to enable distributed drones (nodes) to agree on state updates such as location and heading, even in the absence of GNSS signals for one or more nodes. In our prototype, each node uses GNSS and local sensing, and communicates over WiFi in a simulated swarm. Upon signal loss, consensus is used to reconstruct or verify the position of the failed node based on its last known state and trajectory. Our system demonstrates robustness in maintaining swarm coherence and fault tolerance through a lightweight, scalable communication model. This work offers a practical and secure foundation for decentralized drone operation in unpredictable environments.

Index Terms—Blockchain, UAV swarms, Raft consensus, GNSS-denied environments, Fault tolerance

I. Introduction

Rapid proliferation of autonomous swarms, including unmanned aerial vehicles (UAVs), ground robots, and maritime systems, has enabled transformative applications in domains such as logistics, infrastructure monitoring, agriculture, and disaster response [1, 2, 3]. These distributed systems rely heavily on precise coordination, predominantly facilitated by Global Navigation Satellite Systems (GNSS) such as GPS, GLONASS, Galileo, and BeiDou [4]. However, GNSS signals are often unreliable in real-world conditions, such as urban canyons, dense vegetation, or indoor and subterranean environments. Even with recent advances in noise-resilient GNSS acquisition techniques [5], the technology remains vulnerable to spoofing, jamming, and environmental interference. To mitigate these limitations, systems frequently integrate Inertial Navigation Systems (INS) to improve robustness and continuity. This inherent limitation of standalone GNSS or INS systems motivates the need for collaborative, consensus-driven approaches, such as SwarmRaft, that combine GNSS, INS, and peer-to-peer fusion to achieve robust and fault-tolerant state estimation in UAV swarms.

All the authors are from Skolkovo Institute of Science and Technology. Manuscript received xx, 2025; revised xx, 2025.

As swarm deployments scale in size and mission complexity, achieving reliable consensus among agents becomes a critical challenge, one that draws direct parallels to problems in distributed computing and blockchain technologies. In this context, swarms must maintain accurate and synchronized state information (e.g., position, velocity, heading) despite partial failures and environmental interference [6].

Several mission-critical scenarios exemplify the need for robust consensus mechanisms. In urban package delivery, GNSS reflections from buildings can compromise altitude coordination, and a single altimeter failure can lead to collisions [7]. During bridge inspections, complete GNSS denial requires drones to maintain spatial awareness solely through local sensing and swarm coordination. Agricultural spray missions present additional complexities, as drones must adjust for dynamic weight changes during chemical dispersion in rural settings, often degraded by GNSS.

These operational constraints echo classic challenges of distributed systems, yet swarm robotics introduces distinct requirements [8, 9]. Unlike traditional consensus algorithms that emphasize Byzantine fault tolerance, such as those used in permissioned or permissionless blockchain systems, swarm consensus must prioritize low-latency agreement, operate under stringent resource constraints, and address crash-fault-dominated failure models [10, 11, 12]. The core principles of consensus remain: ensuring *termination* (all non-faulty agents reach a decision), *agreement* (consistency across decisions), and *integrity* (decisions are based on valid inputs), even in the presence of partial system failures [13].

UAV swarms operate under unique constraints, including limited compute and energy resources, intermittent connectivity, and bounded-delay communication networks. Consequently, traditional blockchain consensus protocols such as Proof-of-Work and Proof-of-Stake are ill-suited due to their computational and energy overhead [14, 15]. Similarly, Byzantine Fault-Tolerant (BFT) protocols, e.g., Practical Byzantine Fault Tolerance (PBFT), Tendermint, Exonum [16, 17, 18], assume adversarial behavior and incur unnecessary complexity for typical crash-fault scenarios in swarms. This creates a compelling opportunity to adapt crash-tolerant consensus protocols, such as Raft [19], which are designed for environments where nodes (the drones) may fail by crashing, but not by acting maliciously. Raft's lightweight leader-based architecture is well suited to the resource-constrained and real-time nature of swarm robotics.

To address these challenges, we propose SwarmRaft, a

2

consensus-driven positioning and crash-tolerant system designed for drone swarms. Our approach leverages the Raft consensus algorithm to enable drones to communicate and synchronize over a distributed network, even in partially GNSS-denied environments. SwarmRaft integrates GNSS and INS data to enable drones to exchange critical state information such as position and heading. In the event of GNSS loss or sensor malfunction, the swarm uses consensus to reconstruct or verify the location and trajectory of affected nodes based on shared data and prior motion. This consensus-driven estimation ensures that the swarm remains cohesive and continues its mission safely, even when individual drones experience degraded sensing.

Our work makes the following key contributions:

- Adaptation of Raft for Swarm Environments: We adapt the Raft consensus protocol to support real-time coordination in UAV swarms, prioritizing low-latency decision-making and resilience under crash-fault conditions typical in GNSS-compromised settings.
- Sensor Fusion for Robust State Estimation: Swarm-Raft combines GNSS and INS data using a distributed consensus framework to produce robust, fault-tolerant state estimates. This fusion mitigates the effects of signal intermittency, spoofing, and sensor drift, enhancing the swarm's overall situational awareness and operational reliability.
- Empirical Validation in Realistic Scenarios: We evaluate SwarmRaft through a comprehensive set of experiments and simulations, demonstrating its effectiveness in preserving swarm cohesion and operational continuity across urban and natural terrains.

Motivated by the real-world challenges UAV swarms face, such as GNSS spoofing, jamming, and sensor failure, we develop a lightweight, fault-tolerant coordination framework that maintains collective situational awareness even when individual agents are compromised. Unlike centralized or resource-intensive approaches, SwarmRaft leverages crashtolerant consensus and sensor fusion to operate efficiently in contested or degraded environments. This work bridges swarm robotics and distributed consensus, demonstrating that established protocols like Raft can be adapted for real-time UAV coordination. Our design balances theoretical soundness with practical deployability, yielding substantial improvements in localization accuracy and swarm resilience across simulated and real-world scenarios. SwarmRaft demonstrates how lightweight consensus protocols can enhance resilience and coordination in UAV swarms operating under GNSS-degraded or adversarial conditions.

II. RELATED WORK

Decentralized coordination mechanisms for UAV swarms have gained increasing attention, particularly through consensus-based and blockchain-backed approaches. While leader-follower architectures have shown success in formation control [20], their reliance on central nodes introduces vulnerabilities to single points of failure. In contrast, distributed strategies offer greater resilience. Tariverdi et

al. [21] present a formation control framework tailored to UAV dynamics, and Jia et al. [22] demonstrate decentralized estimation and coordination in quadcopter swarms. Similarly, Zuo et al. [23] propose a voting-based scheme for leader election in lead–follow UAV swarms under constrained communication ranges, representing an earlier Raft-inspired approach tailored to leader selection rather than distributed fault-tolerant localization.

In the context of Industrial IoT (IIoT), Seo et al. [24] propose a co-design framework that jointly optimizes communication scheduling and consensus execution to meet strict latency requirements. Although their setting involves fixed infrastructure, their insights into delay-aware consensus coordination are applicable to mobile multi-agent systems. Similarly, Ilić et al. [25] introduce adaptive asynchronous gossip algorithms for heterogeneous sensor networks, offering resilience and scalability through weighted message exchange without requiring global synchronization. While their work focuses on distributed estimation in sensor-rich environments, the underlying principles of asynchronous fault-tolerant consensus motivate swarm-scale extensions.

Blockchain-based protocols add guarantees of trust and tamper resistance in decentralized coordination. P-Raft [26] optimizes consensus in consortium settings, and Dynamic Trust Practical Byzantine Fault Tolerance (DTPBFT) [27] dynamically adjusts trust levels in UAV networks. Yazdinejad et al. [28] propose a zone-based drone authentication framework using a Delegated Proof-of-Stake (DDPOS) mechanism to enable secure, low-latency inter-zone handovers in smart city deployments, and fairness-oriented multi-UAV communication frameworks for urgent tasks [29].

Comparative analyses of consensus algorithms, such as Raft, PBFT, and Proof-of-Work (PoW) have focused on trade-offs in scalability, fault tolerance, and energy use [30, 31, 32, 33], and communication efficiency optimizations in UAV ad hoc networks [34]. Surveys such as [35] highlight the challenges of adapting these mechanisms to real-time robotics, particularly under energy and processing constraints.

Research on fault-tolerant swarm behavior includes predictive failure mitigation [36], self-healing topologies [37], and GNSS spoofing resilience [38], as well as UAV-aided localization via RF characteristics [39]. Robust navigation under sensor failures [40] and adaptive topology control [41] further emphasize the need for distributed, resilient solutions.

Real-world deployments, such as outdoor flocking and coordinated flight demonstrations [42], underline the feasibility of translating algorithmic designs to physical swarm systems.

Despite these advances, many existing frameworks either impose significant communication or computational overhead, or fail to address the specific fault models and coordination challenges faced by UAV swarms operating under GNSS degradation and adversarial interference. In contrast, SwarmRaft introduces a lightweight, crash-tolerant localization framework that combines peer-based voting with distributed fault recovery, offering robust performance in dynamic and partially compromised environments.

III. PROPOSED SOLUTION

A. Overview of the Consensus Scheme

In modern UAV swarms, individual GNSS or INS readings alone cannot guarantee resilience against sensor faults or deliberate spoofing. Our consensus scheme, SwarmRaft, leverages peer-to-peer distance measurements, crash fault-tolerant communication consensus, and a Byzantine-resilient evaluation mechanism to detect and correct malicious or *faulty* position reports. At each time step k, the swarm determines the position of each UAV (node). To determine the positions, nodes communicate via Raft consensus and elect a leader for the step. The leader collects every node's position estimate (derived from GNSS when available and maintained by INS otherwise), tests if it is consistent with others, recovers it if required, and returns the results to the nodes.

This design achieves fault tolerance under the assumption that up to f nodes' sensors may be corrupted, so long as $n \geq 2f+1$, where n is the total number of nodes in the swarm. Honest nodes never collude to mislead, and initial positions are trusted. The result is a distributed protocol that detects spoofing and significant sensor drift, locally filters out unreliable data, and recovers accurate positions through consensus based on neighbor estimates.

A high-level workflow of SwarmRaft in the case of a determined leader at a single step k is as follows:

- Sense: Each UAV measures its position(GNSS/INS) and internode distances.
- 2) **Inform**: Measurement data is sent to the leader (as a client's transaction).
- 3) **Estimate**: The leader recomputes each UAV's position from range constraints.
- 4) **Evaluate**: The leader determines if the data of each sensor is *faulty* or *honest* based on residuals.
- 5) **Recover**: If a sensor is determined to be *faulty*, the leader replaces its data with the position computed with peer estimates.
- 6) **Finalize**: The leader sends the final results to all nodes.

B. System & Threat Model

Each UAV in our scheme is modeled as a fully-capable sensing and communication node. At each discrete time step k, node i collects an absolute position measurement $\mathbf{z}_{i,k}^{\mathrm{GNSS}}$ from its onboard GNSS receiver; because GNSS alone can drift or be spoofed, the UAV also carries an INS that produces short-term motion increments $\Delta \mathbf{u}_{i,k}$ through accelerometers and gyroscopes. To bind these two modalities and detect inconsistent readings, every UAV is further equipped with a ranging sensor, for example, ultra-wideband or Received Signal Strength Indicator (RSSI)-based) that yields noisy internode distances $d_{ij,k}$ to each neighbor j. Finally, nodes share all measurements and voting messages over an authenticated broadcast channel protected by pre-distributed cryptographic keys, so that neither message forgery nor tampering can occur undetected.

We assume the communication graph among the n UAVs is fully connected and synchronous: in each protocol round every node can exchange messages with every other node, and

timing is sufficiently well aligned that messages from step k are received before step k+1 begins. This idealization lets us focus on sensor faults rather than network delays or partitions. To tolerate up to f arbitrarily faulty or malicious sensors, we require the bound $n \geq 2f+1$. Under this constraint, even if f nodes report completely corrupted GNSS or ranging data (or collude to bias their votes), the remaining honest nodes still form a majority that can detect and override any bad information.

3

Assumption 1: Nodes' compute and communication modules are *honest*, authorized, and synchronous.

Assumption 2: Only GNSS sensors can be Byzantine, while INS are reliable. Up to f nodes' GNSS sensors out of n may be arbitrarily corrupted, where $n \ge 2f + 1$.

Assumption 3: True positions at time k=0 are securely known.

We model three classes of adversarial behavior against which our consensus scheme must defend:

Attack scenario 1 (GNSS Spoofing Attack): An adversary corrupts the satellite positioning signals received by up to f UAVs, causing each compromised node to report position measurements $\mathbf{z}_{i,k}^{\text{GNSS}}$ that are offset by an arbitrary, potentially time-varying bias. Such spoofing can be orchestrated remotely and may remain undetected by the individual UAV's receiver.

However, our inter-node consistency checks and consensus mechanism aim to reveal any GNSS readings that deviate significantly from the peer-estimated location.

Attack scenario 2 (Ranging-Tampering Scenario): The adversary injects errors into the inter-UAV distance measurements $d_{ij,k}$. This could be accomplished by jamming or forging ultra-wideband pulses (or manipulating RSSI readings), causing one or more *honest* nodes to compute incorrect range constraints.

Since our fused position estimates rely on the integrity of these distance measurements, the protocol's voting and median-based recovery step are designed to mitigate the impact of up to f such corrupted range reports.

Attack scenario 3 (Collusion Among Faulty Sensors): We allow for collusion among up to f faulty sensors across different UAVs. In this worst-case scenario, compromised nodes coordinate both their reported measurements and their votes in the SwarmRaft protocol in an attempt to sway the swarm's decision about a particular node's status.

By enforcing the requirement $n \geq 2f+1$ and using majority thresholds, the scheme guarantees that *honest* votes outnumber colluding malicious votes, ensuring that no coalition of size $\leq f$ can force an incorrect global decision.

C. Position Estimation

At each time step $k \in \mathbb{N}$, the UAV with index $i \in \{1, \ldots, n\}$ obtains two complementary sources of information about measurements of its state. First, the GNSS receiver produces a direct but noisy readout of the true position $\mathbf{x}_{i,k}$:

$$\mathbf{z}_{i,k}^{\text{GNSS}} \ = \ \mathbf{x}_{i,k} \ + \ \mathbf{v}_{i,k}^{\text{GNSS}}, \quad \mathbf{v}_{i,k}^{\text{GNSS}} \sim \mathcal{N}(0, R_{\text{GNSS}}),$$

where $z_{i,k}^{\text{GNSS}}$ is the GNSS measurement, $v_{i,k}^{\text{GNSS}} \in \mathbb{R}^3$ is the GNSS measurement noise, and $R^{\text{GNSS}} \in \mathbb{R}^{3 \times 3}$ is its covari-

TABLE I: Summary of Notation

Symbol	Description	
$x_{i,k}$ $z_{i,k}^{GNSS}$	True position of drone i at time k	
$z_{i,k}^{\text{CINSS}}$ $v_{i,k}^{\text{CNSS}}$	GNSS position measurement of drone <i>i</i> GNSS measurement noise	
$x_{i,k}^{INS}$	Position estimated from INS	
$v_{i,k}^{INS}$	INS sensor noise or drift	
$d_{ij,k} \ \eta_{ij,k}$	Measured distance from drone i to j Noise in range measurement $d_{ij,k}$	
R_{GNSS}, R_{INS}	Covariance matrices for GNSS and INS noise	
$\sigma_d^2 \ N$	Variance of range noise Number of drones in the swarm	
f	Number of compromised/faulty drones	

ance matrix. Here, $\mathcal{N}(\mu, \Sigma)$ denotes a multivariate Gaussian distribution with mean μ and covariance Σ .

Second, in the event of a GNSS outage, the UAV can fall back on its inertial navigation system. The INS predicts the next position $\mathbf{x}_{i,k}^{INS} \in \mathbb{R}^3$ based on the previous estimate $x_{i,k-1}$ and the inertial measurements $\Delta u_{i,k}$, which represent increments derived from accelerometer and gyroscope readings over the interval [k-1,k]:

$$x_{i,k}^{\text{INS}} = f(x_{i,k-1}, \, \Delta u_{i,k}) + v_{i,k}^{\text{INS}}, \quad v_{i,k}^{\text{INS}} \sim \mathcal{N}(0, R^{\text{INS}}),$$

where $f(\cdot)$ denotes the INS state-propagation model, $v_{i,k}^{\text{INS}} \in \mathbb{R}^3$ is the accumulated INS error over one step, and $R^{\text{INS}} \in \mathbb{R}^{3 \times 3}$ characterizes the associated covariance.

Thus, the INS provides high-rate relative motion at the cost of accumulating drift, while GNSS provides drift-free absolute position with higher instantaneous noise.

To cross-check these two sensor streams, each UAV also measures its distance to every other UAV. If UAV j measures the range to UAV i, the reading satisfies

$$d_{ij,k} = \left\| \mathbf{x}_{i,k} - \mathbf{x}_{j,k} \right\| + \eta_{ij,k}, \quad \eta_{ij,k} \sim \mathcal{N}(0, \sigma_d^2).$$

where $d_{ij,k}$ is the measured distance from UAV j to UAV i at time step k, $x_{i,k}$ and $x_{j,k}$ are the true positions of UAVs i and j, respectively, $\eta_{ij,k}$ is the measurement noise, and σ_d^2 is its variance. This geometric constraint links the true positions of nodes i and j and can reveal inconsistencies when one node's GNSS add/or INS has been spoofed or drifted excessively.

D. Fused Estimation through Probabilistic Localization

Each node i measures its distances to all neighboring nodes and pairs this information with its own GNSS-derived position. Both the GNSS estimate and the set of computed inter-node distances are then transmitted to the leader node. Similarly, every other node in the network reports its GNSS position together with its locally computed distances to neighbors. Upon receiving the report from node i, the leader uses the GNSS location of i and its distance measurements to approximate the possible locations of neighboring nodes. Since a single distance measurement corresponds to a circle (in 2D) or a sphere (in 3D) centered at node i, each neighbor's location is constrained to lie within such a probabilistic region. By aggregating the distance-based regions provided by all nodes,

the leader constructs an intersection of feasible regions that captures the most likely positions of every node. This process can be interpreted as a form of SwarmRaft's fused estimation: node i's GNSS information defines its own anchor position, while the distance measurements extend this knowledge to constrain neighboring nodes. The leader's role is to integrate these partial constraints from all reporting nodes, producing a consistent, network-wide probabilistic localization of the fleet.

E. Fault Detection

In this phase, the leader evaluates the consistency of a node's reported position by estimating its location from internode distances and the GNSS-based positions of its neighbors. Large discrepancies between a node's self-reported GNSS position and the probabilistic regions derived from neighbors' distance measurements indicate that either the reporting node's sensors or the peers' range measurements may be *faulty*.

To make this precise, we define a residual, apply a statistically justified threshold, and then cast a binary vote.

Specifically, for node i, the leader constructs feasible regions of its location from all non-faulty neighbor reports (as spheres/circles defined by measured distances). The reported GNSS position $\mathbf{z}_{i,k}$ is then compared against this feasible region. We define the residual as the minimum Euclidean distance between $\mathbf{z}_{i,k}$ and the feasible region boundary:

$$e_{i,k} = \min_{\mathbf{x} \in \mathcal{R}_{i,k}} \|\mathbf{x} - \mathbf{z}_{i,k}\|,$$

where $\mathcal{R}_{i,k}$ denotes the feasible region of node *i*'s position at time *k* derived from verified neighbors.

Under nominal (non-attacked) conditions, this residual is primarily driven by sensor noise in GNSS and inter-drone ranging, so it remains small with high probability. To detect abnormal residuals, we define a threshold

$$T = \mu_e + 3\sigma_e,$$

where μ_e and σ_e are obtained via offline calibration under honest sensor operation. By Gaussian tail bounds, the probability that an honest residual exceeds T is less than 0.01. Any residual $e_{i,k} > T$ is thus treated as strong evidence of a fault or spoofing attempt.

1) Local Vote Broadcast: Based on the threshold test, the leader assigns a binary decision for each node i:

$$v_{i,k} = \begin{cases} +1, & e_{i,k} \le T, \\ -1, & e_{i,k} > T. \end{cases}$$

A decision of +1 indicates that node i's reported position is *consistent* with the probabilistic regions derived from neighbor constraints, while -1 signals a potential *fault*. These decisions are then broadcast to all nodes, forming the input for the subsequent SwarmRaft stage.

- If $S_i \geq 0$, decide i is honest.
- If $S_i < 0$, decide i is faulty.

F. Position Correction & Recovery

Once a node i is flagged as faulty, its position is recomputed using only the non-faulty neighbors' information. Let $\mathcal{N}_i^{\text{good}}$ denote the set of neighbors of node i that are not flagged as faulty. Each neighbor $j \in \mathcal{N}_i^{\text{good}}$ provides a reported position $\mathbf{x}_{j,k}$ and the measured distance $d_{i,j}$ to node i.

We model the feasible location of the faulty node i as the intersection of spheres centered at each non-faulty neighbor with radius equal to the measured distance:

$$S_j = \{ \mathbf{x} \in \mathbb{R}^3 \mid ||\mathbf{x} - \mathbf{x}_{j,k}|| = d_{i,j} \}.$$

The intersection of these spheres defines the feasible region for the node's true position.

To compute a single position estimate $\tilde{\mathbf{x}}_{i,k}$, we solve a nonlinear least-squares problem that minimizes the squared deviations from all spheres:

$$ilde{\mathbf{x}}_{i,k} = rg \min_{\mathbf{x} \in \mathbb{R}^3} \sum_{j \in \mathcal{N}_i^{\mathsf{good}}} (\|\mathbf{x} - \mathbf{x}_{j,k}\| - d_{i,j})^2.$$

This formulation finds the point that best fits all range measurements from non-faulty neighbors, effectively performing a multilateration.

In practice, we initialize the optimization at the centroid of the non-faulty neighbors:

$$\mathbf{x}_{i,k}^{(0)} = rac{1}{|\mathcal{N}_i^{ ext{good}}|} \sum_{j \in \mathcal{N}_i^{ ext{good}}} \mathbf{x}_{j,k},$$

and refine $\tilde{\mathbf{x}}_{i,k}$ via iterative least-squares (e.g., using a soft- L_1 loss to reduce sensitivity to residual errors). Once obtained, the node resets its INS state to $\tilde{\mathbf{x}}_{i,k}$.

Note: If too few non-faulty neighbors remain (e.g., due to false positives), we skip multilateration and rely solely on the node's INS propagation. This ensures that the recovery procedure remains robust even under adversarial conditions.

Note: In rare cases, multiple honest sensors may be falsely flagged as faulty, potentially leaving too few verified neighbors for multilateration. When this occurs, we skip median/multilateration recovery and rely solely on the node's INS propagation to update its state.

G. Security & Fault-Tolerance Analysis

The security and fault-tolerance guarantees of SwarmRaft arise from its algorithmic design, the assumptions of the threat model, and the underlying Raft consensus protocol.

Safety: SwarmRaft ensures safety through majority voting by enforcing the condition $n \geq 2f+1$, the system guarantees that *honest* nodes always outnumber any coalition of up to f faulty or malicious nodes. This majority threshold prevents incorrect decisions from being adopted, even in the presence of adversarial behavior.

Liveness: In a synchronous network setting, the protocol guarantees liveness as long as a majority of nodes remain honest and responsive. The Raft algorithm enables reliable leader election and, once a leader is in place, timely progress is maintained throughout the swarm through efficient rounds of position estimation, fault detection, and recovery.

Integrity: All inter-node communications are authenticated using pre-distributed cryptographic keys, preventing forgery and tampering. Consensus-based validation ensures the trust-worthiness of the reported data. Although rare statistical anomalies can incorrectly flag an *honest* sensor as *faulty*, such false positives are mitigated by majority voting. In edge cases where too many nodes are flagged, the system falls back to inertial (INS) data for continuity, preserving mission integrity at the cost of reduced accuracy.

5

Communication Efficiency: SwarmRaft is designed to scale efficiently. The leader incurs a communication and computation cost of O(2(n-1)) per round, while non-leader nodes communicate only with the leader at O(2) cost. This model minimizes bandwidth usage and processing load as the swarm scales. Raft log replication further ensures consistency across all nodes.

Fault Tolerance: The protocol tolerates up to f Byzantine faults, including GNSS spoofing, range manipulation, and collusion. By requiring $n \geq 2f+1$, SwarmRaft ensures that honest node votes dominate and that consensus decisions remain trustworthy even under adversarial conditions. The coordination of the leader of the Raft and the replicated logs reinforces fault-tolerant behavior during ongoing operations.

Recovery: When faults are detected, SwarmRaft employs a robust median-based recovery strategy that replaces outlier or corrupted readings with consensus estimates derived from peer data. This allows the swarm to maintain stable formation and control even under localized sensor failures. Recovery decisions are consistently propagated via Raft's replication mechanism, preserving swarm-wide state alignment.

IV. PROTOTYPE IMPLEMENTATION

To evaluate SwarmRaft under adversarial localization scenarios, we developed a modular simulation framework in Python that models both the dynamics and vulnerabilities of drone swarms. The framework simulates each drone's ground-truth motion, inertial navigation estimates, GNSS observations (including spoofed signals), and noisy inter-drone ranging data. A dedicated leader node module performs position verification, fault detection through voting, and recovery via iterative multilateration. To support performance benchmarking, the simulator includes tools for running large-scale Monte Carlo experiments that vary swarm size and attacker configuration, producing statistical metrics including mean absolute error (MAE). Additionally, the framework supports real-time visualization and post-simulation plotting to analyze error evolution and swarm behavior.

A. Algorithmic Procedure

The leader-based verification algorithm operates in two main stages. First, each drone's reported position is cross-validated against inter-drone distance measurements. The leader tallies votes from all pairwise checks, where consistent neighbors contribute positive votes and inconsistent ones contribute negative votes. Any drone that accumulates a majority of negative votes is flagged as potentially faulty.

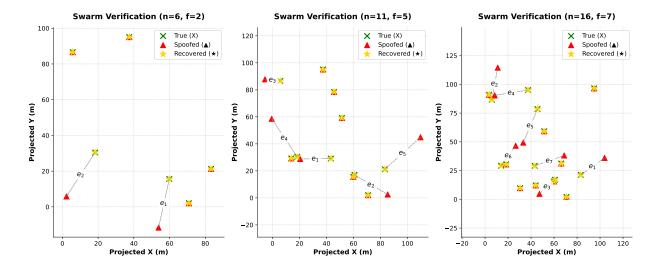


Fig. 1: Visualization of SwarmRaft recovery under different swarm sizes (N=6,11,16). Yellow stars represent recovered positions using consensus and range-based fusion.

In the second stage, the leader refines the positions of flagged drones using multilateration. This involves solving a least-squares problem with respect to verified neighbors, iteratively adjusting the estimated position until convergence or until a maximum number of iterations is reached. If the corrected estimate deviates significantly from the originally reported position, the corrected value replaces the spoofed report; otherwise, the drone is reclassified as valid. This process guarantees that up to f malicious or faulty drones can be corrected in a swarm of size $n \geq 2f + 1$.

The prototype accommodates configurable attack injection, including GNSS spoofing, range distortion, and randomized attacker behavior. The implementation serves as the experimental foundation for the visual and quantitative evaluations discussed in subsequent sections. The source code for Swarm-Raft is publicly available on GitHub [43].

V. NUMERICAL EXPERIMENTS AND RESULTS

A. Qualitative Results for Position Recovery

Figure 1 illustrates the SwarmRaft recovery mechanism across varying swarm sizes. The three subplots correspond to $(n, f) \in \{(6, 2), (11, 5), (16, 7)\}$, i.e., swarms with two, five, and seven faulty drones, respectively. Each visualization demonstrates that SwarmRaft is able to correctly identify spoofed drones and restore their reported positions to match ground truth via multilateration. For spoofed drones, the recovered position (yellow star) aligns with the true position (green cross), while the spoofed report (red triangle) is visibly displaced. Arrows further highlight the deviation between true and spoofed reports, making the correction visually evident. SwarmRaft's ability to detect and isolate faulty agents, even in small swarms with limited redundancy. The recovered positions align closely with the truth of the ground, highlighting the effectiveness of peer triangulation and consistency-based voting. Each subfigure demonstrates how noisy or spoofed measurements are corrected through distributed voting and recovery. This visualization not only showcases concrete recovery behavior but also reveals the spatial structure of localization errors in both small- and large-scale swarms, complementing the accompanying statistical evaluation.

Each drone D_i is annotated with:

- Green cross (imes): the ground-truth position $\mathbf{x}_i^{\mathrm{true}}$.
- **Red triangle** (\triangle): the spoofed reported position $\mathbf{x}_i^{\text{rep}}$.
- Yellow star (\star): the recovered position $\hat{\mathbf{x}}_i$ computed via multilateration.

The SwarmRaft protocol verifies positions through two stages. First, consistency voting checks whether reported inter-drone distances match measured distances; drones that fail majority voting are flagged as faulty. Second, flagged drones undergo multilateration using non-faulty neighbors as anchors. If the corrected estimate significantly deviates from the reported position, the spoofed report is replaced with the multilateration-based recovery. This ensures that up to f compromised drones can be tolerated in a swarm of size $n \geq 2f + 1$.

B. Quantitative Evaluation Across Swarm Sizes and Attacks

To complement the qualitative visualizations, we conduct a large-scale statistical evaluation of SwarmRaft under varying swarm sizes and adversarial conditions. For each swarm size $n \in \{3,5,7,9,11,13,15,17\}$ and each number of attacked drones $f \in \{1,2,\ldots,8\}$, we run Monte Carlo trials where f drones are randomly selected as spoofed. Spoofed drones report displaced positions and manipulated ranges, while the remaining drones report correctly.

Each trial produces two error metrics:

- **Baseline Error:** The MAE of reported (potentially spoofed) positions relative to ground truth.
- **Recovered Error:** The MAE of the recovered and multilaterated positions $\hat{\mathbf{x}}_i$ produced by SwarmRaft.

Algorithm 1 Leader-based Neighbor Verification with Voting and Multilateration

```
Require: Reported positions \mathbf{x}_i^{\text{rep}} from all UAVs i = 1, \dots, n
      Inter-UAV distance matrix D = [d_{ij}] measured from true
      geometry Base tolerance \epsilon > 0, maximum iterations k_{\rm max}
Ensure: Verified positions \mathbf{x}_i^{\text{ver}} and fault flags for all UAVs
 1: Leader collects \mathbf{x}_{1:n}^{\text{rep}} and distance matrix D
 2: for each UAV A = 1 to n do
                                                                  Initialize vote counter v_A \leftarrow 0
 3:
 4:
           for each neighbor B \neq A do
                 Compute distance from reports: \hat{d}_{AB} = \|\mathbf{x}_{A}^{\text{rep}} - \mathbf{x}_{A}^{\text{rep}}\|
 5:
     \mathbf{x}_{B}^{\mathrm{rep}} \|
                 if |\hat{d}_{AB} - d_{AB}| < \tau then
                                                              6:
                      v_A \leftarrow v_A + 1
 7:
 8:
                      v_A \leftarrow v_A - 1
 9:
                 end if
10:
           end for
11:
           Flag A as faulty if v_A < 0
12:
           Initialize \mathbf{x}_A^{\mathrm{ver}} \leftarrow \mathbf{x}_A^{\mathrm{rep}}
13:
14: end for
15: for each UAV A flagged as faulty do
                                                                             ⊳ Stage 2:
      Multilateration refinement
           Select non-faulty neighbors \mathcal{N}_A
16:
           if |\mathcal{N}_A| < 3 then
17:
                 \mathcal{N}_A \leftarrow \text{all other UAVs} \quad \triangleright \text{ fallback if insufficient}
18:
      anchors
19:
           end if
           Initialize estimate \mathbf{p} \leftarrow \mathbf{x}_{\Delta}^{\text{rep}}
20:
           for k=1 to k_{\max} do
21:
                 Update p by solving least-squares residuals:
22:
                  \mathbf{p} \leftarrow \arg\min_{\mathbf{q} \in \mathbb{R}^3} \sum_{j \in \mathcal{N}_A} \left( \|\mathbf{q} - \mathbf{x}_j^{\text{rep}}\| - d_{jA} \right)^2
                 if converged then
23:
                      break
24:
                 end if
25:
26:
           Compute deviation \Delta_A = \|\mathbf{x}_A^{\text{rep}} - \mathbf{p}\|
27:
28:
           if \Delta_A > \epsilon then
                 \mathbf{x}_A^{\mathrm{ver}} \leftarrow \mathbf{p}

⊳ replace spoofed report

29:
30:
                 \mathbf{x}_A^{\mathrm{ver}} \leftarrow \mathbf{x}_A^{\mathrm{rep}}
31:
32:
                 Mark A as non-faulty
           end if
33:
34: end for
35: return Verified positions \mathbf{x}_{1:n}^{\text{ver}} and fault flags
```

Figure 2 shows the scaling performance of SwarmRaft as swarm size grows. For a given number of faulty drones f, we consider the minimal swarm size n=2f+1, which guarantees that the majority of drones remain non-faulty. The plot reports the mean recovery error on a logarithmic scale, averaged over 10,000 Monte Carlo simulations. As the size of the swarm increases relative to f, the recovery accuracy improves dramatically, with a mean error dropping from approximately

19 m to 0.28 m.

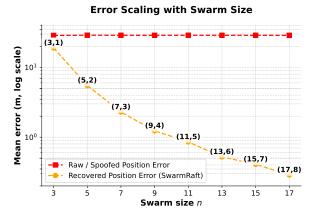


Fig. 2: Scaling performance of SwarmRaft as swarm size increases. Mean recovery error (log scale) decreases significantly with larger swarms, from 19 m to 0.28 m.

We observe that:

- Reported (spoofed) positions quickly diverge from ground truth, reflecting the impact of adversarial manipulation.
- SwarmRaft consistently reduces error compared to the baseline, even when nearly half of the swarm is compromised.
- 3) Recovery accuracy improves with swarm size, demonstrating scalability and resilience of the protocol.

Figure 2 further compares error distributions across trials. Baseline errors show high spread, large medians, and many outliers, confirming vulnerability to spoofing. In contrast, SwarmRaft-corrected positions exhibit low medians, tight interquartile ranges, and few outliers, reflecting robustness under adversarial conditions.

C. Computational Complexity

To provide a fair comparison, we summarize the computational complexity of the proposed verification algorithm against conventional GNSS and INS solutions. For GNSS, position computation after signal tracking reduces to solving a set of pseudorange equations, typically involving $m\approx 6$ –10 satellites. This least-squares solution requires a matrix inversion of size $m\times m$, with complexity $\mathcal{O}(m^3)$, which is effectively constant in practice. INS propagation involves simple state integration (position, velocity, attitude), with complexity $\mathcal{O}(n)$ for a state dimension $n\approx 9$ –15. When GNSS and INS are fused using a Kalman filter, the update step has complexity $\mathcal{O}(n^3)$ due to covariance updates, where $n\approx 15$ states, yielding on the order of 3,000 arithmetic operations per update.

In contrast, in our proposed distributed swarm verification scheme, each regular node computes distances to all other nodes, yielding $\mathcal{O}(N)$ operations per update for swarm size N. The leader node, in the worst-case solvable scenario $(f \approx (N-1)/2$ faulty nodes), must iteratively correct f faulty positions, resulting in $\mathcal{O}(fN) \approx \mathcal{O}(N^2)$. For typical values

System	Relevant dimension	Complexity per update	Typical scale
GNSS (LS solution)	$m \approx 610$ satellites	$\mathcal{O}(m^3)$ (constant)	Few 10 ² ops
INS propagation	$n \approx 915$ states	$\mathcal{O}(n)$	Few 10^2 ops
GNSS+INS (Kalman)	$n \approx 15$ states	$\mathcal{O}(n^3)$	\sim 3,000 ops
Proposed algorithm	N drones	Regular: $\mathcal{O}(N)$, Leader: $\mathcal{O}(N^2)$	$200-6,000 \text{ ops (for } N \le 17)$

TABLE II: Computational complexity comparison of GNSS, INS, and the proposed verification algorithm.

 $(N \leq 17)$, this corresponds to fewer than 6,000 floating-point operations, which is of the same order as a GNSS+INS Kalman update and easily handled by modern embedded processors. A summary of these computational complexities, together with the proposed SwarmRaft verification algorithm, is provided in Table II for ease of comparison.

VI. CONCLUSION

This paper introduced SwarmRaft, a novel consensus-based protocol for resilient and decentralized UAV swarm localization in GNSS-degraded and adversarial environments. Unlike prior approaches that rely on centralized control or computationally intensive BFT protocols, SwarmRaft adapted the Raft consensus algorithm to support lightweight, crash-tolerant decision-making within resource-constrained aerial swarms. The system fused inertial navigation data with inter-drone ranging and applied distributed voting to detect and recover from sensor faults or spoofed position reports. Through extensive simulations under varying swarm sizes and attack intensities, SwarmRaft demonstrated significant improvements in both mean and median localization accuracy compared to GNSS-only baselines, particularly in scenarios with partial compromise. The protocol's ability to maintain collective situational awareness without a central authority highlights its practical deployability for time-sensitive missions such as autonomous logistics, disaster response, and infrastructure inspection. Looking ahead, future research will focus on incorporating confidence-weighted fusion, dynamic trust scores, and asynchronous consensus strategies, as well as validating the system on physical UAV swarms in real-world settings. These advancements aim to establish SwarmRaft as a foundational architecture for fault-resilient coordination in next-generation autonomous multi-agent systems.

ACKNOWLEDGMENT

We acknowledge the use of ChatGPT and DeepSeek in enhancing the readability and clarity of this manuscript. These tools were employed to assist in refining language and improving the overall presentation of the content. However, the authors retain full responsibility for the integrity, accuracy, and intellectual contributions of the research at all stages.

REFERENCES

- [1] A. A. Laghari, A. K. Jumani, R. A. Laghari, and H. Nawaz, "Unmanned aerial vehicles: A review," *Cognitive Robotics*, vol. 3, pp. 8–22, 2023.
- [2] M. AlMarshoud, M. S. Kiraz, and A. H. Al-Bayatti, "Security, privacy, and decentralized trust management in vanets: A review of current research and future directions," ACM Computing Surveys, vol. 56, pp. 1–39, 10 2024.
- [3] I. Bae and J. Hong, "Survey on the developments of unmanned marine vehicles: Intelligence and cooperation," Sensors, vol. 23, p. 4643, 5 2023.

- [4] B. Bhatta, Global Navigation Satellite Systems, 2nd ed. CRC Press, 2021.
- [5] H. Zhang, Y. Xu, R. Luo, and Y. Mao, "Fast gnss acquisition algorithm based on sfft with high noise immunity," *China Communications*, vol. 20, no. 5, p. 70–83, May 2023. [Online]. Available: http://dx.doi.org/10.23919/JCC.2023.00.006
- [6] M.-T. O. Hoang, K. A. R. Grøntved, N. van Berkel, M. B. Skov, A. L. Christensen, and T. Merritt, *Drone Swarms to Support Search and Rescue Operations: Opportunities and Challenges*, 2023, pp. 163–176.
- [7] K. Kuru, D. Ansell, W. Khan, and H. Yetgin, "Analysis and optimization of unmanned aerial vehicle swarms in logistics: An intelligent delivery platform," *IEEE Access*, vol. 7, pp. 15804–15831, 2019.
- [8] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Sok: Consensus in the age of blockchains," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*. ACM, 10 2019, pp. 183–198.
- [9] M. Raikwar, N. Polyanskii, and S. Müller, "Sok: Dag-based consensus protocols," in 2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2024, pp. 1–18. [Online]. Available: https://ieeexplore.ieee.org/document/10634358/
- [10] V. Buterin, "On public and private blockchains ethereum blog," 2015. [Online]. Available: https://blog.ethereum.org/2015/08/ 07/on-public-and-private-blockchains/
- [11] B. Group and J. Garzik, "Public versus private blockchains. part 1: Permissioned blockchains," bitfury.com, pp. 1–23, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt1-1.pdf
- [12] —, "Public versus private blockchains part 2: Permissionless blockchains," bitfury.com, pp. 1–20, 2015. [Online]. Available: http://bitfury.com/content/5-white-papers-research/public-vs-private-pt2-1.pdf
- [13] C. Dwork, N. Lynch, and L. Stockmeyer, "Consensus in the presence of partial synchrony," *Journal of the ACM*, vol. 35, pp. 288–323, 4 1988. [Online]. Available: http://portal.acm.org/citation.cfm?doid= 42282.42283
- [14] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," www.bitcoin.org, pp. 1–9, 2008. [Online]. Available: https://bitcoin.org/ bitcoin.pdf
- [15] A. Kiayias, A. Russell, B. David, and R. Oliynykov, *Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol*. Springer, Cham, 8 2017, vol. 10401 LNCS, pp. 357–388. [Online]. Available: http://link.springer.com/10.1007/978-3-319-63688-7_12
- [16] M. Castro and B. Liskov, "Practical byzantine fault tolerance," Proceedings of the Third Symposium on Operating Systems Design OSDI '99, pp. 173–186, 1999. [Online]. Available: http://pmg.csail.mit. edu/papers/osdi99.pdf
- [17] J. Kwon, "Tendermint: Consensus without mining," pp. 1–10, 2014. [Online]. Available: tendermint.com/docs/tendermint.pdf
- [18] Y. Yanovich, I. Ivashchenko, A. Ostrovsky, A. Shevchenko, and A. Sidorov, "Exonum: Byzantine fault tolerant protocol for blockchains," bitfury.com, pp. 1–36, 2018. [Online]. Available: https://bitfury.com/ content/downloads/wp-consensus-181227.pdf
- [19] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Annual Technical Conference, USENIX ATC 2014*, 2014.
- [20] R. Rafifandi, D. L. Asri, E. Ekawati, and E. M. Budi, "Leader-follower formation control of two quadrotor uavs," SN Applied Sciences, vol. 1, pp. 1–12, 2019.
- [21] A. Tariverdi and J. Torresen, "Rafting towards consensus: Formation control of distributed dynamical systems," 2023. [Online]. Available: https://arxiv.org/abs/2308.10097
- [22] Z. Jia, M. Hamer, and R. D'Andrea, "Distributed estimation, control and coordination of quadcopter swarm robots," arXiv preprint arXiv:2102.07107, 2021.
- [23] Y. Zuo, W. Yao, Q. Chang, X. Zhu, J. Gui, and J. Qin, "Voting-based scheme for leader election in lead-follow uav swarm with constrained

- communication," *Electronics*, vol. 11, no. 14, p. 2143, Jul. 2022. [Online]. Available: http://dx.doi.org/10.3390/electronics11142143
- [24] H. Seo, J. Park, M. Bennis, and W. Choi, "Communication and consensus co-design for distributed, low-latency, and reliable wireless systems," *IEEE Internet of Things Journal*, vol. 8, pp. 129–143, 1 2021.
- [25] N. Ilić, M. Vučetić, A. Makarov, R. Petrović, and M. Punt, "Adaptive asynchronous gossip algorithms for consensus in heterogeneous sensor networks," *IEEE Internet of Things Journal*, vol. 12, pp. 25516–25532, 7 2025.
- [26] S. Lu, X. Zhang, R. Zhao, L. Chen, J. Li, and G. Yang, "P-raft: an efficient and robust consensus mechanism for consortium blockchains," *Electronics*, vol. 12, no. 10, p. 2271, 2023.
- [27] P. Han, X. Wu, and A. Sui, "Dtpbft: A dynamic and highly trusted blockchain consensus algorithm for uav swarm," *Computer Networks*, vol. 250, p. 110602, 2024.
- [28] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, H. Karimipour, G. Srivastava, and M. Aledhari, "Enabling drones in the internet of things with decentralized blockchain-based security," *IEEE Internet of Things Journal*, vol. 8, pp. 6406–6415, 4 2021.
- [29] F. Xu, B. Duo, Y. Xie, G. Pan, Y. Yang, L. Zhang, Y. Ye, T. Bao, T. A. Gulliver, and Y. Wang, "Multi-uav assisted mixed fso/rf communication network for urgent tasks: Fairness oriented design with drl," *IEEE Transactions on Vehicular Technology*, vol. 74, no. 1, p. 1736–1741, Jan. 2025. [Online]. Available: http://dx.doi.org/10.1109/TVT.2024.3453333
- [30] V. Sharma and N. Lal, "A novel comparison of consensus algorithms in blockchain," *Advances and Applications in Mathematical Sciences*, vol. 20, no. 1, pp. 1–13, 2020.
- [31] "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*. ACM, 4 2018, pp. 1–15. [Online]. Available: https://dl.acm.org/doi/10. 1145/3190508.3190538
- [32] P. Kostyuk, S. Kudryashov, Y. Madhwal, I. Maslov, V. Tkachenko, and Y. Yanovich, "Blockchain-based solution to prevent plastic pipes fraud," in 2020 Seventh International Conference on Software Defined Systems (SDS). IEEE, 4 2020, pp. 208–213. [Online]. Available: https://ieeexplore.ieee.org/document/9143879/
- [33] B. Borzdov, M. Minchenok, and Y. Yanovich, "Practical vulnerabilities in byzantine fault-tolerant blockchain consensus protocols," in 2023 XVIII International Symposium Problems of Redundancy in Information and Control Systems (REDUNDANCY). IEEE, 10 2023, pp. 94–99.
- [34] J. Chen, J. Wang, J. Wang, and L. Bai, "Joint fairness and efficiency optimization for csma/ca-based multi-user mimo uav ad hoc networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 18, no. 7, p. 1311–1323, Oct. 2024. [Online]. Available: http://dx.doi.org/10.1109/JSTSP.2024.3435348
- [35] U. S. Aditya, R. Singh, P. K. Singh, and A. Kalla, "A survey on blockchain in robotics: Issues, opportunities, challenges and future directions," *Journal of Network and Computer Applications*, vol. 196, p. 103245, 2021.
- [36] J. O'Keeffe and A. G. Millard, "Predictive fault tolerance for autonomous robot swarms," arXiv preprint arXiv:2309.09309, 2023.
- [37] V. S. Varadharajan, D. St-Onge, B. Adams, and G. Beltrame, "Swarm relays: Distributed self-healing ground-and-air connectivity chains," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5347–5354, 2020.
- [38] A. RANGANATHAN, A. BELFKI, and P. CLOSAS, "The impact of gnss spoofing on unmanned aerial vehicle swarms."
- [39] M. Kato, T. Koketsu Rodrigues, T. Abe, and T. Suganuma, "Exploiting radio frequency characteristics with a support unmanned aerial vehicle to improve wireless sensor location estimation accuracy," *IEEE Internet* of Things Journal, vol. 11, no. 24, pp. 39570–39578, 2024.
- [40] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Fault-tolerant cooperative navigation of networked uav swarms for forest fire monitoring," *Aerospace Science and Technology*, vol. 123, p. 107494, 2022.
- [41] G. Wang, H. Luo, X. Hu, H. Ma, and S. Yang, "Fault-tolerant communication topology management based on minimum cost arborescence for leader-follower uav formation under communication faults," *International Journal of Advanced Robotic Systems*, vol. 14, no. 2, p. 1729881417693965, 2017.
- [42] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz, and T. Vicsek, "Outdoor flocking and formation flight with autonomous aerial robots," in 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2014, pp. 3866–3873.
- [43] K. Dev, "Swarmraft," https://github.com/kapeldev/SwarmRaft, 2025, gitHub repository.



Kapel Dev is a wireless communication researcher with expertise in secure positioning, signal processing, and blockchain applications for next-generation networks. He holds an M.Sc. in Information Technology from the Skolkovo Institute of Science and Technology (Skoltech), where his thesis addressed security vulnerabilities in IEEE 802.11az ranging protocols. His work includes international research collaborations on physical-layer security, SDR-based wireless testbeds, and a co-invented U.S. patent for secure key exchange. His current research explores

blockchain-integrated solutions to enhance trust and resilience in positioning systems.



Yash Madhwal (Member, IEEE) is a Research Scientist at the Skolkovo Institute of Science and Technology (Skoltech), where he specializes in applying blockchain technology to solve supply chain challenges. He earned his Ph.D. in 2024, with a focus on blockchain applications in supply chain systems. Yash has authored several scientific papers in which he developed prototypes of blockchain-based decentralized applications (DApps) designed to address real-world industrial problems. He serves as a teaching assistant for the course "Introduction"

to Blockchain" and regularly conducts technical seminars that demonstrate practical methods for building blockchain applications. Additionally, he serves as a guest lecturer at various universities, delivering introductory lectures on blockchain technology and its potential applications.



Sofia Shevelo is a M.Sc. graduate in Engineering Systems from the Skolkovo Institute of Science and Technology (Skoltech), Moscow, Russia. Her research focuses on human–robot interaction (HRI), haptics, robotics, machine learning, and computer vision. Her research explores gesture-based interfaces to improve the intuitiveness and user experience of drone operating systems. She collaborates with the Artificial Intelligence in Dynamic Actions team at Skoltech and contributes to projects involving multimodal user interfaces for robotics platforms.



Pavel Osinenko is an Associate Professor at the Skolkovo Institute of Science and Technology (Skoltech), where he leads the AIDA (AI in Dynamic Action) research group. He earned his Doctor habilitatus degree in 2025 from Technische Universität Chemnitz, Germany, focusing on reinforcement learning (RL) with formal guarantees for safety, stability, and robustness. His research interests center on the intersection of RL and control theory, particularly the development of trustworthy AI for dynamical systems. He has authored numerous peer-

reviewed publications and proposed several RL algorithms, including the CALF (Critic as Lyapunov Function) framework. At Skoltech, he teaches graduate courses and supervises research in machine learning, RL, and control systems. He also conducts technical seminars and contributes to interdisciplinary projects aimed at building reliable, provable, and safe AI systems for real-world applications.



Yury Yanovich (Member, IEEE) received his Bachelor's (Honours) and Master's (Honours) degrees in Applied Physics and Mathematics from the Moscow Institute of Physics and Technology, Moscow, Russia, in 2010 and 2012, respectively. He received his Ph.D. in Probability Theory and Mathematical Statistics from the Institute for Information Transmission Problems, Moscow, Russia, in 2017. He is currently an Assistant Professor at the Skolkovo Institute of Science and Technology, Moscow, Russia. Yury is the author of the Exonum consensus protocol

and has been a lecturer in the Introduction to Blockchain course at top Russian universities since 2017. His research interests include blockchain, consensus protocols, privacy, and their applications.