

ACCELERATED FORMATION-RECOVERY LAYER FOR SWARM CONTROL WITH A SINGLE GLOBAL COMMAND

Konstantin Amelin*

St. Petersburg State University
St. Petersburg, Russia
k.amelin@spbu.ru

Andrey Chernov*

St. Petersburg State University
St. Petersburg, Russia
a.o.chernov@spbu.ru

Oleg Granichin*

St. Petersburg State University
St. Petersburg, Russia
o.granichin@spbu.ru

Article history:

Received 25.07.2026, Accepted 20.09.2026

Abstract

The paper proposes an accelerated local formation-recovery layer for robot swarms driven by a single global command. The global command specifies the motion of the group, while each robot uses only neighbor-to-neighbor communication to restore formation consistency. The local correction problem is formulated as a regularized strongly convex consensus objective and solved using the Accelerated Local Voting Protocol (ALVP), an accelerated variant of the standard Local Voting Protocol (LVP). In the considered aerial-swarm implementation, the same recovery principle is applied to heading, altitude, and horizontal position alignment. The approach is evaluated in a high-fidelity multirotor software-in-the-loop (SITL) environment with ArduPilot SITL, emulated inter-robot communication, packet losses, and bounded control inputs. In parameter-wise recovery experiments, ALVP reduces the mean convergence time by 37.2%, 39.8%, and 38.6% for heading, altitude, and horizontal position, respectively, compared with LVP. In stationary formation assembly, ALVP remains faster for all tested swarm sizes and shows lower variance for larger swarms. Commanded-motion experiments further indicate that local recovery is necessary under global motion and that ALVP improves post-maneuver recovery in turning and circular trajectories. These results suggest that accelerated local voting is a promising mechanism for improving transient formation recovery in single-interface swarm control.

Key words

Robot swarms, formation control, single-command control, accelerated consensus, Local Voting Protocol, multirotor SITL.

1 Introduction

Unmanned robotic swarms are attracting growing interest due to their advantages in coverage, redundancy, scalability, and robustness to individual robot failures [Chen et al., 2020]. Collectives of aerial, ground, and surface robots enable applications in agriculture, search-and-rescue, environmental monitoring, mapping, infrastructure inspection, and delivery services [Debie et al., 2023; Skorobogatov et al., 2020]. In many of these missions, robots are not required merely to move independently. Instead, they must preserve a structured relative configuration while the group moves through the environment [Tang et al., 2023]. Maintaining such a formation under sensing noise, actuation uncertainty, communication losses, and external motion commands remains a central problem in swarm control.

Two classical coordination paradigms are centralized and decentralized control. In centralized control, a base station collects global state information and computes individual commands for all robots. This can simplify planning and constraint handling, but it scales poorly with the number of agents and is sensitive to communication bottlenecks and single-point failures. In decentralized control, each robot acts using locally exchanged information. This improves scalability and resilience, and has motivated a broad class of flocking, consensus, and formation control methods. However, a fully decentralized architecture is not always the most natural interface for practical robotic missions. Mission objectives are often specified externally by a human operator, a supervisory planner, or a base station, especially in safety-critical scenarios such as disaster response, inspection, and precision agriculture.

This motivates a hybrid global–local architecture. A low-dimensional global command specifies the motion

*This work was supported by St. Petersburg State University, research project 121061000159-6.

of the group as a whole, for example a centroid velocity, a joystick input, or a reference generated by a planner. At the same time, distributed local feedback maintains the relative formation using neighbor-to-neighbor communication. Such an architecture combines a simple external command interface with local autonomy: the global command moves the swarm, while the local layer compensates for formation distortions caused by disturbances, actuator heterogeneity, packet losses, and maneuver-induced transients.

The main difficulty is transient recovery speed. Standard consensus-based protocols are attractive because they are distributed, scalable, and robust to local communication constraints. The Local Voting Protocol (LVP) [Vergados et al., 2017], in particular, provides a simple neighbor-based mechanism for local balancing and alignment. Nevertheless, in real-time swarm operation, slow convergence may lead to large formation errors after turns, changes of direction, random initial placements, or temporary communication degradation. Therefore, an effective local layer should not only be distributed and robust, but also recover the desired formation rapidly enough for responsive operator-driven or planner-driven motion.

In this paper, we develop an accelerated local formation-recovery layer for swarms driven by a single global command. The proposed layer is based on the Accelerated Local Voting Protocol (ALVP) [Erofeeva et al., 2024]. The local alignment objective is regularized to obtain a strongly convex surrogate suitable for the ALVP recursion. In the aerial-swarm implementation considered in this work, the local layer acts on heading, altitude, and horizontal position. These quantities are handled through the same recovery principle, but the main motivation is practical: to preserve formation quality during single-interface swarm motion without requiring centralized individual commands for all robots.

1.1 Related Work

Human–swarm interaction studies emphasize the importance of compact interfaces and high-level supervisory commands for controlling large robot groups [Kolling et al., 2015; Kolling et al., 2012]. Semi-autonomous human–robot cooperation has also been demonstrated in mission-oriented settings such as search and rescue [Doroodgar et al., 2010]. These works motivate the use of global or supervisory commands, but they are primarily focused on interaction paradigms rather than on accelerating distributed formation recovery.

Formation control has been studied extensively through flocking, leader–follower, and consensus-based methods. Classical works analyze stable flocking under fixed topologies [Tanner et al., 2003], decentralized shape generation [Hsieh et al., 2008], and flocking with preserved network connectivity [Zavlanos et al., 2007]. Consensus-based formation control provides distributed mechanisms for maintaining inter-robot configurations using local information [Ren, 2006; Zhu et al., 2019].

More recent studies address robust formation coordination under nonlinear dynamics and unknown disturbances [Hu et al., 2022] and dynamic consensus-based formation under communication constraints [Li et al., 2025]. These approaches provide scalable local coordination mechanisms, but their transient recovery speed can be limiting in time-critical global-motion scenarios.

A complementary line of work studies stochastic-approximation and randomized-input control methods under weak statistical assumptions on disturbances, including multiagent load-balancing and adaptive-control problems with arbitrary external noise [Granichin, 2015; Amelin and Granichin, 2016].

Hybrid global–local architectures are closely related to the setting of this paper. In [Zhang et al., 2023], centralized trajectory guidance is combined with local consensus-based estimation and distributed observers. Human-in-the-loop consensus control is considered in [Ma et al., 2023], where a human supervises a leader and followers maintain formation through local consensus. A two-level architecture with supervisory velocity references and local model predictive consensus control is proposed in [Yang et al., 2023]. These works support the idea that global commands and local consensus can be combined, but they do not focus on accelerating the local recovery mechanism itself.

Partially centralized single-console swarm-control architectures have also been studied in earlier work, where a common operator command is combined with local inter-robot communication and LVP-based formation maintenance [Amelin et al., 2025]. A preliminary ALVP-based version for a group controlled by a single remote was later evaluated under sharp maneuvers and packet losses [Arkhipov et al., 2025]. The present paper extends this line by formulating channel-wise strongly convex recovery objectives and evaluating heading, altitude, and horizontal-position recovery in a multirotor SITL stack.

Another relevant direction studies systems controlled by shared global inputs. Multiple magnetic robots actuated by a global magnetic field are considered in [Asadi and Hurmuzlu, 2023], while [Shahrokhi et al., 2017] controls large microrobot swarms through global light-based inputs and swarm statistics. These works demonstrate that global-input strategies can be effective, but the actuation and coordination models differ substantially from mobile robot swarms with neighbor-based local formation recovery.

Compared with the above literature, our work focuses on the local recovery bottleneck in single-interface swarm control. We do not propose a new human–swarm interface or a full centralized planner. Instead, we adapt ALVP to a real-time robotic formation-recovery problem in which the swarm is driven by a single global command, while each robot restores local formation consistency using only neighbor messages. This requires defining pose-compatible local recovery objectives for alti-

tude, heading, and horizontal position, embedding them into a global–local command architecture, and evaluating the resulting layer in a multirotor SITL stack with autopilot dynamics, bounded low-level commands, packet losses, and emulated inter-robot communication.

1.2 Contributions

The main contributions of this work are as follows:

1. We adapt the Accelerated Local Voting Protocol (ALVP) to a real-time swarm formation-recovery layer operating under a single global command. In the proposed architecture, the global command drives the group motion, while the ALVP-based local layer restores formation consistency using only neighbor-to-neighbor communication.
2. We define channel-wise local recovery objectives compatible with robotic pose variables. The altitude, heading, and horizontal position components are formulated as coordinate-wise recovery channels with neighbor-predicted targets, allowing the same ALVP recursion to be applied across the formation quantities used in the aerial-swarm implementation.
3. We regularize each local recovery objective by a control-effort term, obtaining a strongly convex surrogate suitable for the ALVP estimating-sequence update. This connects the acceleration mechanism to physically meaningful local corrections rather than to an abstract consensus variable.
4. We validate the resulting global–local architecture in a high-fidelity multirotor software-in-the-loop (SITL) stack including ArduPilot SITL, bounded low-level commands, packet losses, and emulated inter-robot communication. The evaluation compares ALVP with standard LVP and no-local-correction baselines in parameter-wise recovery, stationary formation assembly, and commanded-motion scenarios.

2 Problem Formulation

We consider a swarm of N mobile robots that receive a common global command while maintaining a prescribed relative formation through distributed local corrections. The global command specifies the motion of the group as a whole, whereas the local layer compensates for formation deviations caused by communication delays, sensing noise, actuator heterogeneity, and limited onboard resources.

In the main simulation study, this architecture is instantiated for aerial robots. The corresponding formation quantities are heading, altitude, and horizontal position. These quantities will be used below to define the local recovery objectives and performance metrics.

The robots exchange information over a time-varying undirected communication graph

$$\mathcal{G}(t) = \{\mathcal{N}, \mathcal{E}(t)\}, \quad \mathcal{N} = \{1, \dots, N\},$$

where $\mathcal{E}(t)$ is the set of available communication links at time t . The neighbor set of robot i is denoted by $\mathcal{N}_i(t)$.

Assumption 1. There exists an integer $T_c > 0$ such that, for every t , the union graph over the interval $[t, t + T_c - 1]$ is connected.

This standard bounded-window connectivity assumption allows temporary packet losses and switching neighbor sets, but excludes permanent decomposition of the swarm into disconnected components. Similar delayed neighbor-exchange models arise in operating-system-level coordination primitives for edge multiagent systems, where agreement mechanisms must tolerate unsynchronized messaging, bounded delays, and peer failures [Krokhalev et al., 2026; Arkhipov et al., 2026].

Let $\mathbf{x}_i(t) \in \mathbb{R}^m$ be the state of robot i , and let

$$\mathbf{X}(t) = [\mathbf{x}_1^\top(t), \dots, \mathbf{x}_N^\top(t)]^\top \in \mathbb{R}^{mN}$$

be the stacked swarm state. A reference centralized formulation can be written in the discrete-time linear-quadratic form

$$\mathbf{x}_i(t+1) = A\mathbf{x}_i(t) + B\mathbf{u}_i(t) + \mathbf{w}_i(t), \quad (1)$$

where A , B , and $\mathbf{w}_i(t)$ denote the state matrix, input matrix, and disturbance, respectively.

Define the tracking error $\mathbf{e}(t) = \mathbf{X}(t) - \mathbf{X}^*(t)$. The corresponding finite-horizon centralized objective is

$$J_0 = \sum_{t=0}^{T-1} \left[\mathbf{e}^\top(t) Q \mathbf{e}(t) + \sum_{i=1}^N \mathbf{u}_i^\top(t) R \mathbf{u}_i(t) \right] + \mathbf{e}^\top(T) Q \mathbf{e}(T), \quad (2)$$

where $Q \succ 0$, $R \succ 0$, and admissible controls satisfy $\|\mathbf{u}_i(t)\| \leq u_{\max}$.

This centralized formulation is used only as a reference point. Solving such a problem over the full state $\mathbf{X}(t) \in \mathbb{R}^{mN}$ is generally undesirable for real-time swarm operation because the optimization dimension grows with N . Therefore, we use a hybrid global–local structure: a low-dimensional global input, generated for example by a human operator, a supervisory planner, or a reduced-order Model Predictive Control (MPC) layer, drives the group motion, while distributed local corrections restore the relative formation using neighbor information.

The present paper does not claim to solve the full centralized finite-horizon optimal control problem. Instead, it focuses on the local formation-recovery layer executed after the global command has been issued. The heading, altitude, and horizontal position components are treated as coordinate-wise instantiations of a common accelerated recovery template. In each channel, robots build a local disagreement objective from neighbor messages, add a control-effort regularizer to obtain a strongly convex surrogate problem, and apply the same Accelerated Local Voting Protocol (ALVP), an accelerated variant of the standard Local Voting Protocol (LVP), to compute the local correction.

3 Main Result

This section presents the proposed global–local control architecture and the accelerated local formation-recovery layer. The global layer provides a low-dimensional command for the motion of the group, while the local layer restores the relative formation using only neighbor information. We first describe this decomposition and the platform-level command map. We then formulate altitude, heading, and horizontal-position recovery as coordinate-wise instantiations of the same local recovery template and apply the Accelerated Local Voting Protocol (ALVP) to this template.

3.1 Global–Local Control Architecture

The full swarm state has dimension mN . Therefore, a centralized controller that computes an individual command for every robot at every time step generally scales with the swarm size. Even in a linear–quadratic setting, solving a centralized Model Predictive Control (MPC) problem over the complete state may become unsuitable for real-time operation as N grows [Erofeeva et al., 2025].

To avoid this dependence at the command-generation level, we use a two-layer architecture. Each robot receives a global command $\mathbf{u}^g(t)$ and combines it with a distributed local correction $\mathbf{u}_i^l(t)$:

$$\bar{\mathbf{u}}_i(t) = \delta_i(t)H_g\mathbf{u}^g(t) + \mathbf{u}_i^l(t), \quad (3)$$

where H_g maps the global command to the reduced-command coordinates of the local correction, and $\delta_i(t) \in \{0, 1\}$ indicates whether robot i has received the global command at time t . The global command may be generated by a human operator, a supervisory planner, a reduced-order MPC controller, or a Linear Quadratic Regulator (LQR). It is responsible for group-level motion, such as translation of the swarm centroid or a common heading command.

The local correction $\mathbf{u}_i^l(t)$ is computed by each robot using only its own state and the information received from its neighbors in $\mathcal{N}_i(t)$. Its purpose is not to solve the full centralized optimal-control problem, but to compensate for formation errors caused by disturbances, communication losses, actuator heterogeneity, and commanded maneuvers.

The reduced command $\bar{\mathbf{u}}_i(t)$ is translated into platform-level inputs by a low-level command map

$$\mathbf{u}_i(t) = \mathcal{R}(\bar{\mathbf{u}}_i(t)), \quad (4)$$

where $\mathcal{R}$ is platform-specific. For an aerial robot, $\mathcal{R}$ produces attitude, throttle, and yaw commands; for a planar ground robot, it may produce wheel commands or body-frame velocity commands. In this paper, $\mathcal{R}$ is assumed to be available at the firmware or low-level control layer.

For local recovery synthesis, we use the local linear approximation

$$\mathcal{R}(\bar{\mathbf{u}}_i(t)) = M\bar{\mathbf{u}}_i(t) + \boldsymbol{\mu}_i(t), \quad (5)$$

where M is a constant input map and $\boldsymbol{\mu}_i(t)$ collects unmodeled nonlinearities and low-level tracking errors. Substituting (5) into the robot dynamics (1) gives

$$\mathbf{x}_i(t+1) = A\mathbf{x}_i(t) + BM\bar{\mathbf{u}}_i(t) + \bar{\mathbf{w}}_i(t), \quad (6)$$

where

$$\bar{\mathbf{w}}_i(t) = \mathbf{w}_i(t) + B\boldsymbol{\mu}_i(t)$$

is treated as a bounded effective disturbance. This reduced model is used only to motivate the local correction design; the closed-loop performance of the resulting architecture is evaluated in simulation.

3.2 Coordinate-Wise Local Recovery Template

The local correction is applied to several formation quantities. In the aerial implementation considered in the simulation study, these quantities are altitude, heading, and horizontal position. Instead of treating them as unrelated controllers, we describe them as coordinate-wise instantiations of the same accelerated recovery template.

Let r denote a recovery channel. In this paper, $r \in \{z, \text{yaw}, xy\}$, where z denotes altitude, yaw denotes heading, and xy denotes horizontal position. For robot i , let $\mathbf{y}_i^{(r)}(t) \in \mathbb{R}^{m_r}$ be the local variable associated with channel r , and let $\hat{\mathbf{u}}_{i,t}^{(r)} \in \mathbb{R}^{m_r}$ be the current estimate of the corresponding local correction. The one-step channel prediction is written as

$$\mathbf{y}_i^{(r)}(t+1) \approx \mathbf{y}_i^{(r)}(t) + G_r\hat{\mathbf{u}}_{i,t}^{(r)}, \quad (7)$$

where G_r is the effective input matrix of the channel.

For each neighbor $j \in \mathcal{N}_i(t)$, let $\hat{\mathbf{y}}_{j \rightarrow i}^{(r)}(t)$ be the value that neighbor j expects robot i to have in channel r . For altitude and heading, this value is obtained from the desired relative offset. For horizontal position, it is the grid-based neighbor-expected position.

For compactness, define the predicted neighbor residual

$$\mathbf{r}_{ij}^{(r)}(\mathbf{u}) = \mathbf{y}_i^{(r)}(t) + G_r\mathbf{u} - \hat{\mathbf{y}}_{j \rightarrow i}^{(r)}(t). \quad (8)$$

For angular variables, such as heading, this residual is evaluated after wrapping the angular difference to the principal interval. Thus, the Euclidean norm used below is interpreted channel-wise: standard Euclidean differences are used for altitude and horizontal position, whereas wrapped angular differences are used for heading.

The local regularized recovery objective of robot i in channel r is

$$\phi_{i,t}^{(r)}(\mathbf{u}) = \frac{1}{2} \sum_{j \in \mathcal{N}_i(t)} p^{ij} \left\| \mathbf{r}_{ij}^{(r)}(\mathbf{u}) \right\|^2 + \frac{\lambda_r}{2} \|\mathbf{u}\|_{W_r}^2, \quad (9)$$

where $p^{ij} > 0$ are communication weights, $\lambda_r > 0$, $W_r \succ 0$, and $\|\mathbf{u}\|_{W_r}^2 = \mathbf{u}^\top W_r \mathbf{u}$. The first term penalizes predicted neighbor disagreement, while the second term penalizes excessive local control effort.

The gradient of (9) is

$$\nabla \phi_{i,t}^{(r)}(\mathbf{u}) = G_r^\top \sum_{j \in \mathcal{N}_i(t)} p^{ij} \mathbf{r}_{ij}^{(r)}(\mathbf{u}) + \lambda_r W_r \mathbf{u}. \quad (10)$$

All quantities in (10) are available locally to robot i .

The regularization makes the local surrogate strongly convex. Indeed, the Hessian of (9) is

$$\nabla^2 \phi_{i,t}^{(r)} = \left(\sum_{j \in \mathcal{N}_i(t)} p^{ij} \right) G_r^\top G_r + \lambda_r W_r \succeq \lambda_r W_r \succ 0. \quad (11)$$

Thus, the local channel objective is strongly convex with a bound

$$\mu_r \geq \lambda_r \sigma_{\min}(W_r),$$

and its gradient is Lipschitz continuous with, for example, the bound

$$L_r \leq \left(\sum_{j \in \mathcal{N}_i(t)} p^{ij} \right) \|G_r^\top G_r\|_2 + \lambda_r \|W_r\|_2.$$

These constants are used to select the parameters of ALVP in each channel.

3.3 Local Voting and Accelerated-LVP Update

We now describe the update used in each recovery channel. The standard Local Voting Protocol (LVP) is a distributed gradient-like consensus rule: each robot compares its current channel value with the values predicted by its neighbors and applies a local correction proportional to the corresponding disagreement.

For channel r , the non-accelerated LVP update can be interpreted as the gradient-like step

$$\hat{\mathbf{u}}_{i,t+1}^{(r)} = \hat{\mathbf{u}}_{i,t}^{(r)} - h_r \mathbf{Y}_{i,t}^{(r)}, \quad (12)$$

where $h_r > 0$ is the channel stepsize and $\mathbf{Y}_{i,t}^{(r)}$ is the local gradient estimate computed from neighbor messages. In the deterministic notation of (10), this quantity is

$$\mathbf{Y}_{i,t}^{(r)} = \nabla \phi_{i,t}^{(r)}(\hat{\mathbf{u}}_{i,t}^{(r)}),$$

whereas in implementation it may include measurement and communication errors.

To accelerate this local voting step, we use the ALVP recursion from [Erofeeva et al., 2024]. Unlike a fixed-momentum update, ALVP uses an estimating-sequence parameterization. For each channel r , define the internal parameter set

$$\mathcal{P}_r = \{h_r, L_r, \mu_r, \eta_r, \alpha_{x,r}, \gamma_{0,r}\},$$

where L_r and μ_r are the Lipschitz and strong convexity parameters of the local surrogate, $\eta_r \in (0, \mu_r)$, and $\alpha_{x,r} \in (0, 1)$. Set

$$H_r = h_r - \frac{h_r^2 L_r}{2}. \quad (13)$$

At every time instant, robot i maintains two ALVP variables in each channel: the correction estimate $\hat{\mathbf{u}}_{i,t}^{(r)}$ and the auxiliary variable $\mathbf{v}_{i,t}^{(r)}$. For compactness, define

$$c_r = 2 \left(H_r - \frac{\alpha_{x,r}}{2} \right), \quad q_r = \mu_r - \eta_r.$$

The scalar coefficient $\alpha_{i,t}^{(r)}$ is chosen as the admissible positive solution of

$$\left(\alpha_{i,t}^{(r)} \right)^2 = c_r \left(\gamma_{i,t}^{(r)} + \alpha_{i,t}^{(r)} \left(q_r - \gamma_{i,t}^{(r)} \right) \right). \quad (14)$$

Then

$$\gamma_{i,t+1}^{(r)} = \left(1 - \alpha_{i,t}^{(r)} \right) \gamma_{i,t}^{(r)} + \alpha_{i,t}^{(r)} (\mu_r - \eta_r). \quad (15)$$

The extrapolated point is computed as

$$\mathbf{s}_{i,t}^{(r)} = \frac{\alpha_{i,t}^{(r)} \gamma_{i,t}^{(r)} \mathbf{v}_{i,t}^{(r)} + \gamma_{i,t+1}^{(r)} \hat{\mathbf{u}}_{i,t}^{(r)}}{\gamma_{i,t}^{(r)} + \alpha_{i,t}^{(r)} (\mu_r - \eta_r)}. \quad (16)$$

Robot i then computes the local gradient estimate at this extrapolated point:

$$\mathbf{Y}_{i,t}^{(r)} = \nabla \phi_{i,t}^{(r)}(\mathbf{s}_{i,t}^{(r)}), \quad (17)$$

where the gradient is evaluated using the locally available neighbor messages as in (10). The correction estimate is updated by

$$\hat{\mathbf{u}}_{i,t+1}^{(r)} = \mathbf{s}_{i,t}^{(r)} - h_r \mathbf{Y}_{i,t}^{(r)}. \quad (18)$$

Finally, the auxiliary variable is updated as

$$\mathbf{v}_{i,t+1}^{(r)} = \left(1 - \alpha_{i,t}^{(r)} \right) \mathbf{v}_{i,t}^{(r)} + \frac{\alpha_{i,t}^{(r)}}{\gamma_{i,t}^{(r)}} \left(q_r \mathbf{s}_{i,t}^{(r)} - \mathbf{Y}_{i,t}^{(r)} \right). \quad (19)$$

Thus, ALVP preserves the neighbor-based structure of LVP but replaces the direct local voting step with an estimating-sequence accelerated update. The channel-specific quantities are $\mathbf{y}_i^{(r)}$, the neighbor-expected values $\hat{\mathbf{y}}_{j \rightarrow i}^{(r)}$, the local gradient estimate $\mathbf{Y}_{i,t}^{(r)}$, and the parameters $\mathcal{P}_r$. The internal ALVP recursion itself is the same for altitude, heading, and horizontal-position recovery.

3.4 Channel Instantiations

The three recovery channels used in the experiments differ only in the choice of the channel variable, the construction of $\hat{\mathbf{y}}_{j \rightarrow i}^{(r)}(t)$, and the platform-level output.

3.4.1 Altitude channel. For altitude alignment,

$$\mathbf{y}_i^{(z)}(t) = z_i(t) \in \mathbb{R}.$$

Typically, the desired relative offset is zero, so

$$\hat{y}_{j \rightarrow i}^{(z)}(t) = z_j(t).$$

The resulting channel correction $\hat{u}_{i,t+1}^{(z)}$ is mapped to the throttle correction.

3.4.2 Heading channel. For heading alignment,

$$\mathbf{y}_i^{(\text{yaw})}(t) = \theta_i(t) \in \mathbb{S}^1.$$

Since heading is angular, the channel residual is computed using the wrapped error

$$\text{wrap}(\theta_i(t) - \theta_j(t) - d_{ij}^\theta).$$

In the experiments, $d_{ij}^\theta = 0$, which corresponds to common heading alignment. The resulting correction $\hat{u}_{i,t+1}^{(\text{yaw})}$ is mapped to the yaw command.

3.4.3 Horizontal-position channel. For horizontal formation recovery, let

$$\boldsymbol{\xi}_i(t) = [x_i(t), y_i(t)]^\top$$

be the Cartesian position of robot i . Each robot computes, for every neighbor j , a neighbor-expected position $\hat{\boldsymbol{\xi}}_{j \rightarrow i}(t)$, i.e., the position where neighbor j expects robot i to be according to the desired grid spacing and the current heading of j .

Let $d > 0$ be the grid spacing and define the set of admissible planar offsets

$$\mathcal{S} = \{(kd, \ell d)^\top \mid k, \ell \in \{-1, 0, 1\}, (k, \ell) \neq (0, 0)\}.$$

For every neighbor j , robot i selects

$$\mathbf{s}_{j \rightarrow i}^*(t) \in \arg \min_{\mathbf{s} \in \mathcal{S}} \|\boldsymbol{\xi}_i(t) - (\boldsymbol{\xi}_j(t) + R(\theta_j(t))\mathbf{s})\|, \quad (20)$$

where $R(\theta_j)$ is the planar rotation matrix defined by the heading of robot j . Then

$$\hat{\boldsymbol{\xi}}_{j \rightarrow i}(t) = \boldsymbol{\xi}_j(t) + R(\theta_j(t))\mathbf{s}_{j \rightarrow i}^*(t). \quad (21)$$

The horizontal channel applies the same ALVP update to move $\boldsymbol{\xi}_i$ toward these neighbor-predicted positions. The resulting planar correction $\hat{\mathbf{u}}_{i,t+1}^{(xy)}$ is converted into roll and pitch commands for aerial robots.

3.5 Generic Algorithm

Algorithm 1 summarizes one accelerated recovery step for a single channel r . The same algorithm is executed independently for altitude, heading, and horizontal position with channel-specific variables and parameters.

Algorithm 1 Accelerated Local Recovery Channel via ALVP

Require: Channel variable $\mathbf{y}_i^{(r)}(t)$, neighbor messages $\{\mathbf{y}_j^{(r)}(t)\}_{j \in \mathcal{N}_i(t)}$, previous ALVP variables $\hat{\mathbf{u}}_{i,t}^{(r)}$, $\mathbf{v}_{i,t}^{(r)}$, $\gamma_{i,t}^{(r)}$, channel parameters $\mathcal{P}_r = \{h_r, L_r, \mu_r, \eta_r, \alpha_{x,r}\}$.

Ensure: Channel correction $\hat{\mathbf{u}}_{i,t+1}^{(r)}$ for the next actuation cycle.

- 1: Exchange $\mathbf{y}_i^{(r)}(t)$ with neighbors $j \in \mathcal{N}_i(t)$.
- 2: Construct neighbor-expected values $\hat{\mathbf{y}}_{j \rightarrow i}^{(r)}(t)$ for all $j \in \mathcal{N}_i(t)$.
- 3: Compute $\alpha_{i,t}^{(r)}$ from (14).
- 4: Update $\gamma_{i,t+1}^{(r)}$ using (15).
- 5: Compute the extrapolated point $\mathbf{s}_{i,t}^{(r)}$ using (16).
- 6: Compute the local gradient estimate

$$\mathbf{Y}_{i,t}^{(r)} = \nabla \phi_{i,t}^{(r)}(\mathbf{s}_{i,t}^{(r)})$$

from neighbor data.

- 7: Update the correction estimate:

$$\hat{\mathbf{u}}_{i,t+1}^{(r)} = \mathbf{s}_{i,t}^{(r)} - h_r \mathbf{Y}_{i,t}^{(r)}.$$

- 8: Update the auxiliary variable $\mathbf{v}_{i,t+1}^{(r)}$ using (19).

- 9: **return** $\hat{\mathbf{u}}_{i,t+1}^{(r)}$.
-

The output of Algorithm 1 computed at cycle t is applied as the local correction for the next actuation cycle. The channel corrections are assembled as

$$\mathbf{u}_i^l(t+1) = [(\hat{\mathbf{u}}_{i,t+1}^{(xy)})^\top, \hat{u}_{i,t+1}^{(z)}, \hat{u}_{i,t+1}^{(\text{yaw})}]^\top.$$

At the next cycle, this correction is combined with the global command according to (3) and passed to the low-level command map $\mathcal{R}$.

Consequently, altitude, heading, and horizontal position are not three independent algorithms. They are three coordinate-wise instantiations of the same ALVP-based local recovery mechanism. The channel-dependent elements are the variable $\mathbf{y}_i^{(r)}$, the construction of the neighbor-expected value $\hat{\mathbf{y}}_{j \rightarrow i}^{(r)}$, the input map G_r , and the parameters $\mathcal{P}_r$.

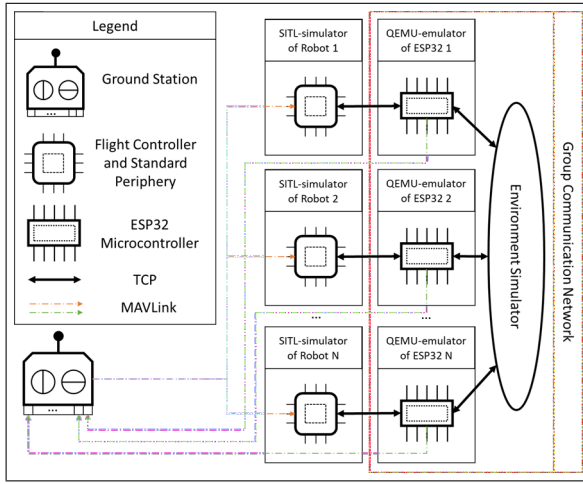


Figure 1. High-fidelity SITL architecture used in the experiments.

4 Simulation Experiments

This section evaluates the proposed Accelerated Local Voting Protocol (ALVP) layer in a high-fidelity multirotor software-in-the-loop (SITL) environment and compares it with the standard Local Voting Protocol (LVP) and, where applicable, with the no-local-correction baseline. The experiments are designed to answer three questions: (i) whether ALVP accelerates individual recovery channels, (ii) whether the full formation can be assembled from random initial conditions, and (iii) whether local recovery remains useful when the swarm is driven by an external global command.

4.1 Simulation Setup

All experiments use a physics-based multirotor simulation with identical onboard controllers, bounded control inputs, state-estimation noise, and synchronized discrete-time updates. Communication is local and determined by a fixed neighborhood radius. Packet losses are enabled in the convergence and assembly experiments: at each communication cycle, an agent misses an incoming neighbor message with probability 0.1, and fails to transmit its own message with probability 0.05.

The SITL stack consists of a Go-based environment simulator, one ArduPilot SITL instance per drone, MAVProxy for global coordination, and an ESP32 firmware running in a QEMU-based simulator for emulated inter-drone communication. Unlike a purely kinematic simulation, this stack includes the vehicle autopilot loop, bounded low-level commands, communication timing, and message losses. Therefore, the experiments evaluate the formation-recovery layer under implementation constraints that are closer to deployment than idealized point-mass simulations. The overall simulation stack is shown in Fig. 1.

All drones send neighbor messages at 5 Hz and compute local corrections at 10 Hz. The desired planar formation is a grid with spacing $d = 10$ m. Unless otherwise stated, initial positions, headings, and altitudes are

sampled independently across runs. The same controller and algorithm parameters are used within each experimental set.

The ALVP parameters used in the simulations are summarized in Table 1. The LVP gains are 0.3, 0.8, and 0.03 for heading, altitude, and horizontal position, respectively. For each method and recovery channel, parameters were tuned empirically on pilot runs and then kept fixed for all reported runs.

Table 1. ALVP parameters used in simulation.

Ch.	α_x	γ_0	h	L	μ	η
Head.	0.75	0.50	0.30	10	0.60	0.050
Alt.	0.40	0.80	0.30	10	0.60	0.100
Pos.	0.10	0.10	0.05	5	0.09	0.005

Although the quantitative evaluation in this paper is performed in SITL, supplementary videos show a preliminary hardware run with several quadrotors executing the same global-command/local-recovery principle: <https://disk.yandex.ru/d/sKljuLKvenlCdA>.

4.2 Experimental Design and Metrics

The first experiment evaluates the three recovery channels separately: heading, altitude, and horizontal position. Each channel is tested over 10 runs for LVP and 10 runs for ALVP with $N = 4$ drones and a duration of approximately 30 s. The remaining channels are kept inactive so that each test isolates one formation quantity.

The second experiment evaluates stationary formation assembly. The swarm starts from random positions and headings, no global motion command is applied, and the local layer assembles the desired formation. The swarm size is varied as

$$N \in \{4, 6, 9, 12\}.$$

Each run lasts approximately 60 s.

The third experiment evaluates formation maintenance under global motion. A swarm of $N = 9$ drones starts from a correct formation and is driven by a uniform external command along three trajectories: straight motion, motion with a turn, and circular motion. The local recovery layer is responsible for compensating formation errors induced by the commanded maneuver.

We use four metrics. Heading error is the aggregate mean absolute yaw deviation from the group heading. Altitude error is the aggregate mean absolute altitude deviation from the group altitude. Horizontal formation error is the aggregate deviation from the desired grid-relative positions. Convergence time is the first time instant at which the corresponding error remains below a prescribed threshold for 1 s. The thresholds are 1° for

heading, 25 cm for altitude, and 0.5 m for horizontal position.

4.3 Results

Table 2 summarizes the repeated convergence experiments for two complementary settings. The first group of rows isolates individual recovery channels. This experiment evaluates the effect of ALVP on one formation quantity at a time and therefore directly tests whether acceleration improves the corresponding local recovery mechanism. The second group evaluates stationary full-formation assembly, where altitude, heading, and horizontal-position corrections act together while the swarm starts from randomized initial conditions.

In the parameter-wise tests, ALVP consistently reduces the mean convergence time relative to LVP: by 37.2% for heading, 39.8% for altitude, and 38.6% for horizontal position. The variance is also lower or comparable in all three channels. This indicates that the accelerated update improves not only the average transient speed, but also the repeatability of convergence under randomized initial conditions and packet losses.

The stationary full-formation assembly experiment shows that the same trend persists when all recovery channels are active simultaneously. ALVP remains faster for all tested swarm sizes $N \in \{4, 6, 9, 12\}$. As N increases, the relative improvement decreases from 38.7% at $N = 4$ to 21.7% at $N = 12$, which suggests that larger formations introduce additional coupling and communication-induced transients. Nevertheless, the absolute time gap between LVP and ALVP remains substantial. Moreover, for $N \geq 6$, ALVP shows a much smaller variance than LVP, indicating more predictable assembly times for larger swarms.

Fig. 2 shows horizontal formation errors during commanded motion. The no-local-correction baseline produces substantially larger errors, confirming that local recovery is necessary when the group is driven by a common external command. The comparison between LVP and ALVP is scenario-dependent. In straight-line motion, LVP achieves a lower steady error for part of the run. However, in the turn and circular-motion cases, ALVP reduces the post-maneuver error more effectively and reaches a lower final error. This suggests that ALVP is most beneficial when the global command creates stronger transient formation distortions.

The heading and altitude channels were aligned before commanded motion started. During the motion, both quantities remained within admissible bounds even without additional dedicated alignment: below 25 cm for altitude and below 1° for heading. Therefore, the dominant formation degradation in these tests is horizontal, which is the most sensitive channel under translational and turning commands.

Fig. 3 shows representative ALVP trajectories for the same commanded-motion scenarios. The trajectories confirm that the reported error dynamics correspond to

coherent group motion rather than isolated pointwise corrections. In the straight-line case, the formation remains visually stable; in the turn and circular cases, transient distortions appear during the maneuver and are reduced by the local recovery layer.

Overall, the simulation results support the main claim of this paper: the proposed ALVP layer accelerates local formation recovery in convergence-oriented scenarios and improves post-maneuver recovery under non-stationary global motion. The strongest quantitative evidence comes from the repeated parameter-wise and stationary assembly experiments, whereas the commanded-motion experiments demonstrate representative behavior of the global-local architecture under external motion commands.

5 Conclusion

This paper presented an accelerated local formation-recovery layer for robot swarms driven by a single global command. The proposed architecture separates group-level motion from local formation maintenance: the global command specifies the desired motion of the swarm, while the distributed local layer compensates for formation errors using only neighbor information. The local recovery problem was formulated as a regularized strongly convex consensus objective, which enables the use of the Accelerated Local Voting Protocol (ALVP) for heading, altitude, and horizontal-position recovery.

The proposed approach was evaluated in a high-fidelity multirotor software-in-the-loop environment and compared with the standard Local Voting Protocol (LVP). In parameter-wise experiments, ALVP reduced the mean convergence time by 37.2%, 39.8%, and 38.6% for heading, altitude, and horizontal position, respectively. In stationary formation assembly, ALVP was faster for all tested swarm sizes $N \in \{4, 6, 9, 12\}$ and showed lower variance for larger swarms. In commanded-motion experiments, local recovery was necessary to keep the formation bounded under a global motion command, and ALVP improved post-maneuver recovery in turning and circular trajectories.

These results suggest that accelerated local voting is a promising mechanism for improving transient formation recovery in single-interface swarm control. The current study focuses on the local recovery layer and uses SITL validation rather than physical multi-UAV experiments. Future work will include hardware validation, systematic delay and packet-loss sweeps, integration with collision avoidance, and joint tuning of the global command generator and the accelerated local recovery layer.

References

- Amelin, K., Arkhipov, I., Granichin, O., Kiselev, V., and Chernov, A. (2025). A system for robot swarm control with a single remote controller. *Mekhatronika, Avtomatizatsiya, Upravlenie*, **26** (9), pp. 471–479.

Table 2. Convergence summary for parameter-wise recovery and stationary formation assembly.

Experiment	Case	LVP, s	ALVP, s	Improvement	LVP var.	ALVP var.
<i>Parameter-wise recovery, $N = 4$, 10 runs</i>						
Heading	yaw alignment	28.41	17.83	37.2%	1.0409	0.8081
Altitude	altitude alignment	11.66	7.02	39.8%	1.6944	0.4836
Position	horizontal recovery	19.77	12.13	38.6%	3.3001	2.9801
<i>Stationary full-formation assembly, varying N</i>						
Assembly	$N = 4$	28.38	17.40	38.7%	2.1019	2.1150
Assembly	$N = 6$	45.63	32.90	27.9%	13.4756	0.7250
Assembly	$N = 9$	77.20	57.67	25.3%	14.9400	0.7022
Assembly	$N = 12$	101.35	79.32	21.7%	16.0840	0.9510

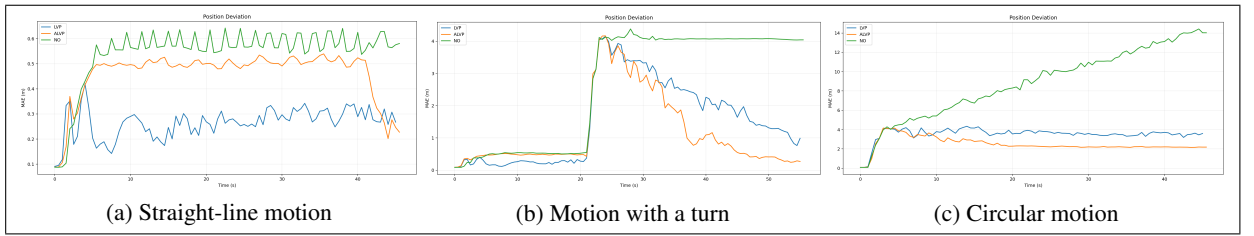


Figure 2. Horizontal formation error during commanded motion.

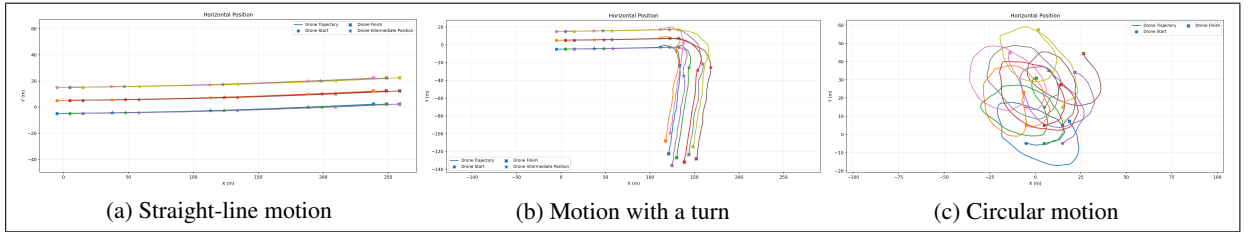


Figure 3. Representative ALVP trajectories during commanded motion.

Amelin, K. and Granichin, O. (2016). Randomized control strategies under arbitrary external noise. *IEEE Transactions on Automatic Control*, **61** (5), pp. 1328–1333.

Arkhipov, I., Erofeeva, V., Granichin, O., Kiselev, V., and Chernov, A. (2025). Accelerated local voting protocol for single remote controller robot swarm system. *Informatics and Automation*, **24** (4), pp. 1029–1058.

Arkhipov, I., Krokhaliev, E., Tarasova, E., and Granichin, O. (2026). Monotonic logical time for cooperative tracking in decentralized multi-agent networks. *IEEE Access*, **14**, pp. 140323–140358.

Asadi, F. and Hurmuzlu, Y. (2023). Motion planning for multiple heterogeneous magnetic robots under global input. *IEEE Transactions on Robotics*, **40**, pp. 697–713.

Chen, W., Liu, J., Guo, H., and Kato, N. (2020). Toward robust and intelligent drone swarm: Challenges and future directions. *IEEE Network*, **34** (4), pp. 278–283.

Debie, E., Kasmarik, K., and Garratt, M. (2023). Swarm

robotics: A survey from a multi-tasking perspective. *ACM Computing Surveys*, **56** (2), pp. 1–38.

Doroodgar, B., Ficocelli, M., Mobedi, B., and Nejat, G. (2010). The search for survivors: Cooperative human-robot interaction in search and rescue environments using semi-autonomous robots. In *2010 IEEE International Conference on Robotics and Automation*, pp. 2858–2863.

Erofeeva, V., Granichin, O., and Uzhva, D. (2025). Meso-scale coalitional control in large-scale networks. *Automatica*, **177**, pp. 112276.

Erofeeva, V., Granichin, O., and Volodina, E. (2024). Accelerated decentralized load balancing in multi-agent networks. *IEEE Access*, **12**, pp. 161954–161967.

Granichin, O. N. (2015). Stochastic approximation search algorithms with randomization at the input. *Automation and Remote Control*, **76**, pp. 762–775.

Hsieh, M. A., Kumar, V., and Chaimowicz, L. (2008). Decentralized controllers for shape generation with robotic swarms. *Robotica*, **26** (5), pp. 691–701.

- Hu, J., Turgut, A. E., Lennox, B., and Arvin, F. (2022). Robust formation coordination of robot swarms with nonlinear dynamics and unknown disturbances: Design and experiments. *IEEE Transactions on Circuits and Systems II: Express Briefs*, **69**(1), pp. 114–118.
- Kolling, A., Nunnally, S., and Lewis, M. (2012). Towards human control of robot swarms. In *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*, pp. 89–96.
- Kolling, A., Walker, P., Chakraborty, N., Sycara, K., and Lewis, M. (2015). Human interaction with robot swarms: A survey. *IEEE Transactions on Human-Machine Systems*, **46**(1), pp. 9–26.
- Krokhalev, E., Arkhipov, I., Granichin, O., and Tarasova, E. (2026). Consensus algorithms for a multiagent operating system. *Cybernetics and Physics*, **15**, pp. 38–55.
- Li, S., Ren, S., Hu, Q., and Li, D. (2025). Dynamic consensus-based formation control with output mask privacy for multi-robot systems. *International Journal of Robust and Nonlinear Control*.
- Ma, L., Zhu, F., and Zhao, X. (2023). Human-in-the-loop consensus control for multiagent systems with external disturbances. *IEEE Transactions on Neural Networks and Learning Systems*.
- Ren, W. (2006). Consensus based formation control strategies for multi-vehicle systems. In *Proceedings of the 2006 American Control Conference*, IEEE, pp. 6–11.
- Shahrokhi, S., Lin, L., Ertel, C., Wan, M., and Becker, A. T. (2017). Steering a swarm of particles using global inputs and swarm statistics. *IEEE Transactions on Robotics*, **34**(1), pp. 207–219.
- Skorobogatov, G., Barrado, C., and Salamí, E. (2020). Multiple uav systems: A survey. *Unmanned Systems*, **8**(02), pp. 149–169.
- Tang, J., Duan, H., and Lao, S. (2023). Swarm intelligence algorithms for multiple unmanned aerial vehicles collaboration: A comprehensive review. *Artificial Intelligence Review*, **56**(5), pp. 4295–4327.
- Tanner, H. G., Jadbabaie, A., and Pappas, G. J. (2003). Stable flocking of mobile agents, part i: Fixed topology. In *Proceedings of the 42nd IEEE International Conference on Decision and Control (CDC)*, vol. 2, pp. 2010–2015.
- Vergados, D. J., Amelina, N., Jiang, Y., Kravlevska, K., and Granichin, O. (2017). Toward optimal distributed node scheduling in a multihop wireless network through local voting. *IEEE Transactions on Wireless Communications*, **17**(1), pp. 400–414.
- Yang, W., Shen, T., Pan, T., Hu, G., and Xu, D. (2023). Distributed model predictive consensus control of unmanned surface vehicles with post-verification. *Drones*, **7**(1), pp. 42.
- Zavlanos, M. M., Jadbabaie, A., and Pappas, G. J. (2007). Flocking while preserving network connectivity. In *Proceedings of the 46th IEEE Conference on Decision and Control (CDC)*, pp. 2919–2924.
- Zhang, Y., Wang, S., Heinrich, M. K., Wang, X., and Dorigo, M. (2023). 3d hybrid formation control of an underwater robot swarm: Switching topologies, unmeasurable velocities, and system constraints. *ISA Transactions*, **136**, pp. 345–360.
- Zhu, H., Juhl, J., Ferranti, L., and Alonso-Mora, J. (2019). Distributed multi-robot formation splitting and merging in dynamic environments. In *2019 international conference on robotics and automation (ICRA)*, IEEE, pp. 9080–9086.