IDS Project: A distributed multi-robot framework for automated warehouse management

Michele Daniel, *Student ID:249817*
Gianluca Lona, *Student ID:248735*

***Abstract*— With the growing adoption of autonomous systems in warehouse automation, multi-robot coordination offers an effective and scalable alternative to centralized logistics. This project presents a distributed control framework for a swarm of mobile robots designed to detect, encircle, and transport packages from an inbound towards outbound zones. The proposed system integrates an Extended Kalman Filter for state estimation, Voronoi-based partitioning for area allocation and target assignment, and a linear consensus algorithm for cooperative decision-making. A formation control strategy maintains a predefined geometric configuration during collective motion without explicitly modeling inter-robot forces, while an obstacle avoidance mechanism ensures safe navigation in dynamic environments. Simulation results demonstrate effective package localization, stable formation maintenance, robust cooperative navigation, and reliable collision avoidance under sensing and communication constraints.**

I. INTRODUCTION

Automation in modern warehouses increasingly relies on autonomous mobile robots (AMRs) to improve efficiency, flexibility, and safety in logistics operations. While many existing systems depend on centralized control, such approaches often face scalability and robustness limitations as the number of robots grows. Distributed control strategies offer an attractive alternative, enabling robots to cooperate through local sensing and communication without a central coordinator.

This project presents an intelligent distributed control framework for a swarm of mobile robots that detect, align around, and transport packages toward an outbound zone. The system combines an Extended Kalman Filter (EKF) for state estimation, Voronoi-based partitioning for area coverage and task allocation, linear consensus for coordination, and formation control for maintaining predefined geometric configurations. Although inter-robot forces are not modeled, the proposed approach achieves stable formation maintenance and cooperative transport behavior in a simulated warehouse environment.

II. PROBLEM FORMULATION

A. Overview

This work addresses the problem of distributed cooperative control for a group of autonomous mobile robots (AMRs) operating as warehouse agents in a two-dimensional environment. The robots are fully connected in terms of communication and are designed to perform tasks autonomously, regardless of the number of agents present in the workspace. A key challenge lies in maintaining a semi-rigid formation that mimics the physical coupling effect that would occur when multiple robots jointly transport a package.

To achieve this, distinct control schemes are employed for robots directly involved in the transport and for those that are not. In addition, each agent must ensure collision avoidance both with other robots and with randomly placed obstacles within the environment, increasing the realism and complexity of the scenario.

The main objective of this project is to integrate a robust distributed control framework combining a Voronoi–Lloyd partitioning scheme for spatial coverage, an Extended Kalman Filter (EKF) for optimal state estimation, and a Linear Consensus algorithm for decentralized estimation and sharing of package locations. The combination of these techniques enables scalable, adaptive, and coordinated transport within a fully distributed robotic swarm.

B. Structure and workflow

The topology of the problem models the map of an ideal warehouse, in which one can recognize the following elements:

- one inbound zone (blue)
- three outbound zones (red)
- two types of obstacles (black): circular or squared
- agents (light blue dots)
- package (gray square)

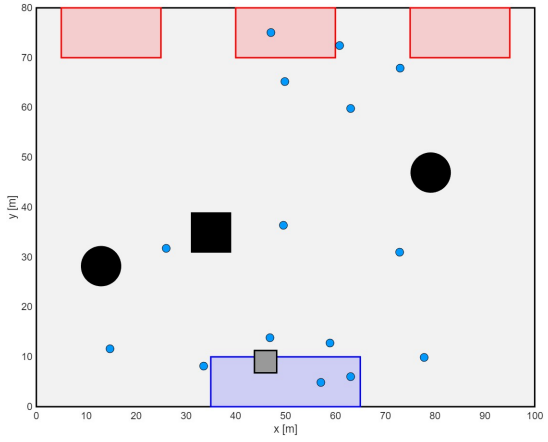

Fig. 1. Warehouse topology

The number of elements is user-defined except for the outbound/inbound zones. The problem's workflow is instead defined by these steps:

1) spawn: an arbitrary number of agents and obstacles spawn inside the map in random positions. The packages spawn randomly inside the inbound zone and their drop-off points in the outbound zones.
2) package localization: the robots that can detect the package proceed to estimate its position via a linear consensus algorithm, sharing the information to the others via the network.
3) ring line-up: the robots decide which and how many agents will carry the package, and proceed to encircle it avoiding collisions with other robots and with obstacles.
4) delivery: when ready, the ring formation picks the package and departs pointing to the designated drop-off point in one of the three outbound zones.
5) drop-off: once the formation has reached the target point, the package is dropped and the simulation starts over again from point 2.

Each of these steps will be thoroughly described in its details in the next sections of the report.

III. System modeling

A. Robots Model

Ideally, an accurate representation of a mobile robot moving in a 2D environment requires a second-order dynamic model with a six-dimensional state vector. Such a formulation captures both the translational and rotational dynamics and typically includes:

- position components $[x, y]$
- linear velocity components $[\dot{x}, \dot{y}]$
- orientation component θ
- angular velocity component ω

Although this level of detail provides a highly realistic representation of the robot's motion, it comes with significant drawbacks. A full second-order model results in a set of nonlinear differential equations that must be integrated at each timestep. This inevitably introduces higher computational cost, and a potentially higher risk of numerical instability, especially when simulating multiple agents or performing real-time control. For the purposes of this project, the emphasis lies on scalability, and real-time simulation capability. Therefore, instead of the full dynamic representation, we adopt a simplified unicycle-style kinematic model.
The state used for the dynamic of the robots is reduced to the following three components:

$$\mathbf{s} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \tag{1}$$

This abstraction still captures the non-holonomic nature of a differential-drive wheeled robot, while avoiding its intrinsic complexity of the physical dynamics.
With this approach joint torques and motors characteristics are completely neglected, instead, velocities are treated directly as control inputs. These include the linear and angular components, as shown below.

$$\mathbf{u} = \begin{bmatrix} v_{lin} \\ \omega \end{bmatrix} \tag{2}$$

Where $v_{lin} = \sqrt{\dot{x} + \dot{y}}$.
The discrete-time equations of motion for the agents, can be written as:

$$s_{k+1} = f(s_k, u_k) \tag{3}$$

Where s_k and s_{k+1} are respectively the robot's states at the current and next time instants, while u_k is the velocity control input.
With the aim of increasing the empirical realism of the system, it is possible to introduce some form of disturbance inside the equations of motion. In the simplified unicycle dynamics, uncertainty is typically introduced through input noise affecting the commanded linear and angular velocities. This choice is motivated by the fact that, in real differential-drive robot, most deviations from the ideal kinematic model originate at the actuation level, keeping in account in particular:

- wheel-slippage and surface irregularities
- imperfect actuation (motors not delivering exactly the commanded velocity)
- general unmodeled forces

In this case the discrete-time dynamics becomes:

$$s_{k+1} = f(s_k, u_k, \nu_k) \tag{4}$$

Where ν_k is defined as a Gaussian noise, with $\nu_k \sim \mathcal{N}(0, Q)$. Matrix Q, namely *Process Noise Covariance Matrix* is chosen as a (2×2) diagonal matrix representing independent noise contributions to the translational and rotational motion. The diagonal entries are specified as:

$$Q = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_\omega^2 \end{bmatrix}, \qquad \sigma_v = 0.2, \;\; \sigma_\omega = 0.1. \tag{5}$$

The extended version of the system dynamics is written as:

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \end{bmatrix} \Delta t \left(\begin{bmatrix} v_{\text{lin},k} \\ \omega_k \end{bmatrix} + \nu_k \right). \tag{6}$$

B. Communication System

In order for the different control strategies to work correctly, it is assumed, in general that the agents located inside the warehouse, are able to communicated through a fully connected network topology. This enables for example to share their estimated state for both a correct formation control and Voronoi partitioning. A scheme of the communication system is represented in Fig.2.

However, assuming that the robots can communicate and share their states at every time instant is not realistic in practical scenarios. Therefore, the following sections describe methods to limit this ideal communication capability, creating more challenging conditions from a control perspective. In particular, communication between any two robots at a given time instant occurs only with a certain probability, thereby simulating possible information loss.

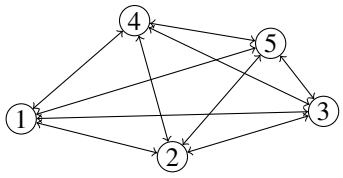

Fig. 2. Fully connected graph with 5 nodes

C. Sensing Systems

For the objective of the project, each robot is thought as equipped with three different measurement systems:

- UWB sensor (for spatial localization): Used to measure the relative distance between the agent and each of the fixed anchors. The measurements are affected by Gaussian noise:

$$d_{meas} = d + \epsilon_{UWB}, \quad \epsilon_{UWB} \sim \mathcal{N}(0, {\sigma_{UWB}}^2), \quad \sigma_{UWB} = 0.2m$$

- LiDAR sensor: Used to retreive measurements of distance between the agents and the surrounding objects (obstacles or packages). The measurement is affected by Gaussian noise:

$$d_{meas} = d + \epsilon_{LiD}, \quad \epsilon_{LiD} \sim \mathcal{N}(0, {\sigma_{LiD}}^2), \quad \sigma_{LiD} = 0.1m$$

- Digital Compass sensor: Used to measure the robot's absolute orientation w.r.t the earth's magnetic field. These types of sensors usually incorporate a magnetometer and a 3-axis accelerometer for tilt compensation. The measure is affected by Gaussian noise:

$$\theta_{meas} = \theta + \epsilon_\theta, \quad \epsilon_\theta \sim \mathcal{N}(0, {\sigma_\theta}^2), \quad \sigma_\theta = 3$$

Different sensors provide measurements at different update rates. In particular, for the sensors directly responsible for estimating the robot's state—the UWB and Digital Compass—the following sampling frequencies are chosen:

- $f_{UWB} = 10Hz$
- $f_\theta = 5Hz$

Although these sensors are typically capable of much higher sampling rates, using lower frequencies creates a more challenging scenario and requires high controller stability.

IV. PROBLEM SOLUTION

This section describes the adopted methods to solve the main tasks of the proposed problem, which can be formalized as follow:

- Robots' state estimation
- Distributed control algorithm with collision avoidance
- Linear consensus for package localization
- Formation control

A. Initialization of the State Estimate

Before an Extended Kalman Filter (EKF) can operate, each robot requires an initial state estimate and its associated covariance. Since no prior information is available at mission start, the initialization relies only on the first sensor readings: (i) noisy ranges to a set of fixed anchors and (ii) a noisy absolute heading from the digital compass. These measurements are combined using a weighted least–squares formulation,

$$\hat{s}_0 = \begin{bmatrix} \hat{x}_0 & \hat{y}_0 & \hat{\theta}_0 \end{bmatrix}^\top = (H^\top C^{-1} H)^{-1} H^\top C^{-1} z,$$
$$P_0 = (H^\top C^{-1} H)^{-1}.$$

where H is the measurement matrix, z is the stacked measurement vector, and C is the measurement noise covariance.

1) Position Estimation via Trilateration: Consider n fixed anchors at known positions $a_i = (x_i, y_i)$. The robot measures the ranges

$$r_i = \|p - a_i\| + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma_r^2),$$

where $p = (x, y)$ is the unknown robot position. The squared–range relations,

$$r_i^2 = (x - x_i)^2 + (y - y_i)^2$$

are linearized by subtracting consecutive equations. For $i = 1, \ldots, n-1$,

$$2((x_{i+1} - x_i)x + (y_{i+1} - y_i)y) = x_{i+1}^2 - x_i^2 + y_{i+1}^2 - y_i^2 - (r_{i+1}^2 - r_i^2).$$

Stacking these relations yields the linear model

$$z_{xy} = H_{xy} \begin{bmatrix} x \\ y \end{bmatrix} + v_{xy}.$$

Since $z_{xy}(i)$ depends on r_i and r_{i+1}, first–order noise propagation gives

$$\mathrm{Var}(z_{xy}(i)) \approx 4\sigma_r^2 (r_i^2 + r_{i+1}^2)$$
$$\mathrm{Cov}(z_{xy}(i), z_{xy}(i+1)) \approx -4\sigma_r^2 r_{i+1}^2.$$

All remaining covariances vanish, producing a tridiagonal matrix C_{xy}.
The weighted least–squares solution for (x, y) is

$$\hat{s}_{xy} = (H_{xy}^\top C_{xy}^{-1} H_{xy})^{-1} H_{xy}^\top C_{xy}^{-1} z_{xy}$$
$$P_{xy} = (H_{xy}^\top C_{xy}^{-1} H_{xy})^{-1}.$$

2) Heading Measurement: The digital compass provides a direct measurement of the robot's heading:

$$z_\theta = \theta + v_\theta, \qquad v_\theta \sim \mathcal{N}(0, \sigma_\theta^2).$$

This yields the scalar model

$$H_\theta = [0\ 0\ 1], \qquad C_\theta = \sigma_\theta^2.$$

3) Combined Initialization: The full initialization stacks the position and heading components:

$$H = \begin{bmatrix} H_{xy} & 0 \\ H_\theta \end{bmatrix}, \qquad z = \begin{bmatrix} z_{xy} \\ z_\theta \end{bmatrix}, \qquad C = \begin{bmatrix} C_{xy} & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}.$$

Solving the weighted least–squares system yields the initial state estimate $\hat{s}_0 = [\, \hat{x}_0,\ \hat{y}_0,\ \hat{\theta}_0 \,]^\top$ and covariance P_0.

B. Extended Kalman Filter (EKF) for state estimation

In order to implement an effective and robust control scheme, it's necessary that all agents are capable of estimating precisely their state. To handle the system's non-linear dynamics and fuse together the different sensors readings, an Extended Kalman Filter (EKF) was developed as the central state-estimation module. This in particular provides a framework for combining informations from multiple sensing modalities (UWB and Digital Compass), while accounting for both model and measurement uncertainties.
The algorithm runs in two main phases at each iteration: a *prediction* and a *update* step.

1) Prediction Step: In this phase, the algorithm exploits the knowledge of the current state estimate, and the applied control input to forecast the state at the next time step using the system non-linear dynamics. We have

$$\hat{s}_{k+1}^{-} = f(\hat{s}_k, u_k), \tag{7}$$

where f is the model dynamics, $\hat{s}_k$ is the current state estimate and u_k is the applied control input.
Moreover, the covariance matrix is also propagated. Since the system model is non-linear, linearization around the current estimate is required:

$$P_{k+1}^{-} = A_k P_k A_k^T + G_k Q G_k^T, \tag{8}$$

with A_k and G_k denoting the Jacobians of the dynamics with respect to the state and the process noise, respectively.

$$\begin{aligned} A_k &= \left.\frac{\partial f(s,u,\nu)}{\partial s}\right|_{s=\hat{s}_k,\, u=u_k,\, \nu=0}, \\ G_k &= \left.\frac{\partial f(s,u,\nu)}{\partial \nu}\right|_{s=\hat{s}_k,\, u=u_k,\, \nu=0}. \end{aligned} \tag{9}$$

Computing explicitly the Jacobians of eq.9 on the unicycle model, we obtain:

$$A_k = \begin{bmatrix} 1 & 0 & -v_k \sin(\hat{\theta}_k)\,\Delta t \\ 0 & 1 & v_k \cos(\hat{\theta}_k)\,\Delta t \\ 0 & 0 & 1 \end{bmatrix}, G_k = \begin{bmatrix} \Delta t \cos(\hat{\theta}_k) & 0 \\ \Delta t \sin(\hat{\theta}_k) & 0 \\ 0 & \Delta t \end{bmatrix}.$$

2) Update Step: Following the prediction phase, the EKF integrates the sensor measurements available at the current time step to refine the state estimate. In this framework, two sensing modalities are considered: a UWB-based trilateration and a digital compass. Since these sensors operate at different effective sampling rates, the update step is structured so that each measurement type is processed independently and only when a new observation is available.

Let $h(\cdot)$ denote the nonlinear measurement function associated with the sensing process. For the UWB subsystem, the corresponding measurement matrix H_{xy} is obtained by linearizing the trilateration model (as shown in detail in IV-A.1). For the digital compass, the orientation measurement is modeled as a direct observation of the system heading, which yields the linear measurement matrix

$$H_\theta = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}.$$

Whenever a measurement becomes available at instant k, the EKF first computes the innovation

$$y_k = z_k - h(\hat{s}_k^{-}), \tag{10}$$

which quantifies the discrepancy between the received measurement z_k and its predicted value. Position measurements satisfy $z_k = z_k^{xy} \in R^{N-1}$ (with N the number of fixed anchors), whereas the compass provides a scalar reading $z_k = z_k^{\theta} \in R$.

The covariance of the innovation is computed as

$$S_k = H_k P_k^{-} H_k^T + R_k, \tag{11}$$

where H_k is either H_{xy} or H_θ, depending on the sensing modality, and R_k denotes the corresponding measurement noise covariance.

The Kalman gain, is evaluated as

$$K_k = P_k^{-} H_k^T S_k^{-1}. \tag{12}$$

The corrected state estimate is subsequently obtained through

$$\hat{s}_k = \hat{s}_k^{-} + K_k y_k, \tag{13}$$

followed by the covariance update:

$$P_k = (I - K_k H_k) P_k^{-}. \tag{14}$$

A key feature of this update structure is that position and orientation corrections remain decoupled. When a UWB measurement is processed, the associated matrix H_{xy} exhibits no sensitivity with respect to the heading variable, ensuring that the update affects only the (x, y) components of the state while leaving the orientation unchanged. Conversely, a compass update employs the scalar matrix H_θ, which directly selects the heading component and prevents unintended corrections to the estimated position.

C. Voronoi-LLoyd based partitioning and control

The prerequisite of this project is to provide a distributed control logic that works reliably and effectively in a rather crowded and narrow environment, using the available suite of sensors for the robots. For this purpose, a Lloyd-based navigation solution was adopted [2], along with Voronoi for space partitioning.

1) Voronoi partitioning: given a discretized map of the environment, the algorithm assigns a portion of it to each robot: let $Q \in R^2$ be the mission space (discretized map), q a generic point in the Q space and p_r the robot position. Given the set of agents and their positions $P = p_1, ..., p_n$ the i-th Voronoi cell can be computed as follows:

$$V_i = \{q \in Q |\, ||q - p_i|| \leq ||q - p_j||, \forall j \neq i\} \tag{15}$$

The equation above assigns each robot located at p_i a portion of the map: its Voronoi cell V_i. This region contains all points that are closer to robot i than to all other robots.

2) Lloyd algorithm: Having computed the Voronoi cell for each agent, a target has to be assigned to control the robots; a possible solution is to use pdf functions $\phi(q)$ in order to provide the robot a desirable location or target inside the map [2]. The way this is achieved is by computing an hypothetical mass associated with each robot's Voronoi cell:

$$M_{V_i} = \int_{V_i} \phi(q)\, dq \tag{16}$$

The weighted centroid of the cell is then computed:

$$C_{V_i} = \frac{1}{M_{V_i}} \int_{V_i} q\, \phi(q)\, dq \tag{17}$$

The position of this centroid can be used as a reference in a control law for the robots, which in the case of this project

is a simple proportional (P) controller that guides the robot towards the centroid:

$$u_i = k_p(C_{V_i} - p_i) \tag{18}$$

This way each agent will converge asymptotically to the position of its centroid. The choice for the expression of $\phi(q)$ to assign to each robot depends on its working state, which is chosen between the following:

- 'p' point: the pdf is an isotropic bivariate Gaussian, described by the following expression:

$$\rho(x,y) = \frac{1}{2\pi\sigma_p^2}\exp\left(-\frac{(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma_p^2}\right) \tag{19}$$

where $\sigma_x^2 = \sigma_y^2 = \sigma_p^2$ is the covariance and (μ_x, μ_y) is the location of the pdf. This pdf is used to guide a robot to a specific point of the map, or to navigate it towards a desired location.

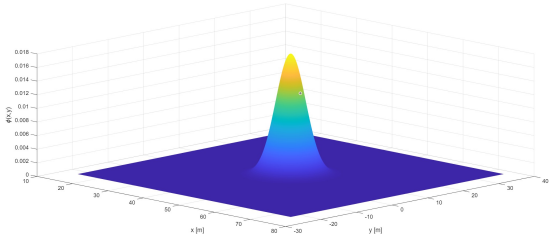

Fig. 3. Isotropic bivariate gaussian pdf

- 'f' free: the pdf is uniform over the map. This provides optimal coverage for the swarm.
- 'r' ring: the ring-shaped pdf (figure IV-C.2) forms a circular 'ridge' that guides the robots during the line-up phase. The expression for ϕ is:

$$\phi(x,y) = \exp\left(-\frac{(r-R)^2}{2\,\sigma_r^2}\right) \tag{20}$$

with $r = \sqrt{(x-\mu_1)^2+(y-\mu_2)^2}$, and R being computed from the package surface area, assuming circular shape. Since all points on the upper edge of the ring share the same value of ϕ, the robots have no preference among them and will therefore spread uniformly along the ring. This working state allows not to give explicit target points to each robot, which would typically imply using a centralized type of commands and communication: the agents will only need to know the position of the package, which is already available from the first step "package localization". Using this information, each robot will build the same pdf and they will autonomously create a ring formation around the package.
- 't' transportation: during package delivery this working state stops the calculation of the centroid for the selected robots, switching off the Voronoi-Lloyd-based control in favour of the formation control algorithm. Choosing this approach means giving the robots in the formation a "higher priority" with respect to the "free" ones in terms of navigation, meaning that the formation will be blind to other robots (but not to static obstacles). The collisions between all agents are still avoided, since all other robots will move out of the formation's way. This is achieved by keeping switched on the computation of the Voronoi cell of all robots, formation included, thus providing the correct indications for the free ones.

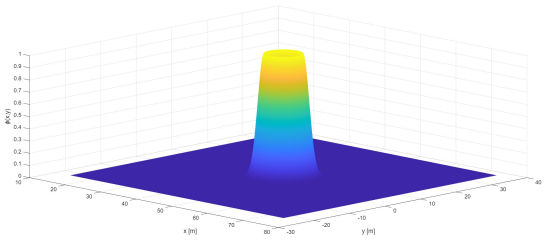

Fig. 4. Ring pdf

- 'i' inbound attraction: the pdf is again a bivariate Gaussian located at the centroid of the inbound zone. The robots with this working state will be attracted to the inbound zone, in order to allow for reliable localization of the package at step 2.

The choice of the value of σ for the working states 'p' and 'r' influences the behaviour of the robots by a large margin. Too little values lead to high precision at the expense of longer travel times and rather numerically unstable centroid computations. For this reason in the code this parameter is chosen adaptively, increasing it when the robot is far from its target so that the pdf becomes wider and less "peaky". This also keeps the mass of the Voronoi cell from becoming too small when the robot is far away, making the centroid computation numerically safer and more reliable. As the robot travels towards the target, the sigma is lowered to provide more precise positioning. The logic for the computation of σ for the 'p' working state is the following:

$$\sigma = \sigma_{0,p}\left(1 + \frac{d}{R}\right) \tag{21}$$

whereas for the 'r' working state is:

$$\sigma = \sigma_{0,r}\left(1 + \frac{||p_r - \mu|| - R}{R}\right) \tag{22}$$

where d is the distance of the robot from the target in the case of 'p', p_r is the robot poistion, R is a reference radius value (used for normalization) and σ_0 is the standard deviation value of the pdf at $d = 0$ which is distinctively chosen on the basis of the working state of each agent.

3) Reactive control for static obstacles: the Voronoi partitioning algorithm alone does not allow for any collision avoidance with the static obstacles present in the map: to enforce this behaviour, a reactive control for static obstacles was adopted [2]. This approach first constrains the Voronoi cell of each robot on its visibility set and then further reduces the cell imposing a sensing range. We first reduce the Voronoi cell by checking the line-of-sight of the robot:

$$S = \{q \in V | p_r + a(q - p_r) \notin O\}, \forall a \in [0, 1] \tag{23}$$

where $O \in Q$ is the obstacle set, i.e. the points in the map assigned to a static obstacle. At this point, the resulting visibility set S is intersected with the Voronoi cell, reducing it to the set $W = V \cap S$. The reduced cell W is then "trimmed" by the sensing radius of the robot r_s:

$$W_{rs} = \{q \in W | \, ||q - p_r|| \leq r_s\} \tag{24}$$

These two operations are practically performed as one, since the sensing radius limits the visibility of the robot by itself. The resulting reduced Voronoi cell W_{rs} (purple in figure IV-C.3) is generally safe for the purpose of robot navigation, due to the Matlab implementation of this algorithm: as soon as the line-of-sight is interrupted by an obstacle (grey in figure IV-C.3), the scanning operation on the line stops and the cell is cut at that point. More in details, the scanning operation is performed using a "ray-casting" approach [3], of which the pseudo-code is the following.

foreach θ *in* $[-\pi, \pi]$ **do**
  for $r \leftarrow 0$ ***to*** r_s ***by*** d_s **do**
    $p \leftarrow [\, x_r + r\cos\theta,\ y_r + r\sin\theta\,]$;
    if $obs(p) = 1$ **then**
      break
    end
    else
      `visible`$(p) \leftarrow$ `true`;
    end
  end
end
return `visible`;

Where r_s is the sensing radius of the robot, d_s is the step increment along the ray, $[x_r, y_r]$ is the current robot position. In practical terms, the robots scans the environment tracing the path of rays originating from its position on the map. Whenever a ray hits an object ($obs(p) = 1$ in the pseudo-code) the visibility is stopped and the cell is cut.

However, the cell may not be, and typically isn't, geometrically convex in general. This implies that in some pathological cases the computed centroid C_{rs} (red cross in figure IV-C.3) could lie on an obstacle, outside the W_{rs}. A solution to this problem [2] is to project the centroid position to the closest safe point to the centroid that belongs to W_{rs}, such that C_{rs} lies inside W_{rs}. With this additional operation, the controls provided to the robot will be safer and more reliable. In the simulations however, the pathological case in which the computed centroid lies outside the cell nearly never occured: this is probably due to a combination of factors, such as the shape of the pdf, the relatively low sensing radius of the robot and the shape of the obstacles.

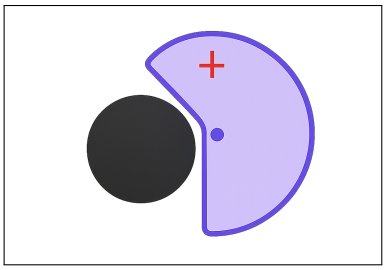

Fig. 5. Reduced Voronoi cell W_{rs}

D. Linear consensus

When a new package appears in the inbound zone, the robots must detect it and estimate its position before coordinating the encirclement and transportation phases. Each robot is equipped with a LiDAR sensor that provides the relative position of the package with respect to its own pose, provided that the package lies within the sensor's nominal range. From this measurement, the robot reconstructs an estimate of the absolute package position by combining its own estimated state with the relative measurement.
In the following discussion, we focus on the subset of N robots that are able to sense the package. Only these robots participate in the estimation and consensus process.

1) Communication assumption: If a classical linear consensus algorithm with Metropolis–Hastings weights were applied over a fully connected communication graph, the system would converge to the average of the initial estimates in a single step. This is because each robot would have degree $d_i = N-1$, leading to a update matrix Q in which all entries are equal to 1/N.
To make the estimation problem more realistic and challenging, we instead assume a time-varying communication topology. Specifically, at each time step only one robot broadcasts its current estimate to the others. This communication pattern requires a different consensus mechanism, namely the *Cyclic Broadcast Gossip* algorithm [1].

2) Algorithm: Suppose that at time instant k, node i is designated to share its estimate with the other agents. In this case, the remaining $N - 1$ robots, assuming no communication losses, receive the broadcasted value $x_i(k)$ and update their estimate as

$$x_j(k+1) = \alpha x_i(k) + (1-\alpha)x_j(k), \quad j \neq i, \tag{25}$$

while the broadcaster keeps its current estimate:

$$x_i(k+1) = x_i(k). \tag{26}$$

These equations can be written compactly in vector form as

$$x(k+1) = Q(k)\, x(k), \tag{27}$$

where the entries of the update matrix $Q^{(i)}$, corresponding

to a broadcast by node i, are given by

$$Q_{jk}^{(i)} = \begin{cases} 1, & k = j = i, \\ 1-\alpha, & k = j \neq i, \\ \alpha, & k = i, k \neq j, \\ 0, & \text{otherwise.} \end{cases} \tag{28}$$

For example, in the 3-node case, the matrix for a broadcast from node 1 would be

$$Q^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ \alpha & 1-\alpha & 0 \\ \alpha & 0 & 1-\alpha \end{bmatrix} \tag{29}$$

In order for the consensus to converge to the average, the update matrices must satisfy the following properties:

- $Q^{(i)}$ doubly stochastic,
- all entries are non-negative.

As shown in Eq. (29), the matrix $Q^{(i)}$ defined above clearly does not satisfy these properties. To enforce them, we modify the i-th row as

$$Q_{ik}^{(i)} = \begin{cases} 1-\alpha(N-1), & k = i, \\ \alpha, & k \neq i. \end{cases} \tag{30}$$

Choosing $\alpha \leq \frac{1}{N-1}$ ensures all entries remain non-negative. In particular, we set $\alpha = \frac{1}{N}$, which, while not providing the fastest convergence, guarantees that all diagonal elements are non-zero. An alternative approach using optimization-based techniques to compute $Q^{(i)}$ matrices was also developed (see *project appendix*). Although this method shows promising convergence rates, it was not selected as the preferred approach for the aims of this project.

E. Control for Transport in Formation

We now describe the control strategy that allows the robots to maintain a predefined formation during the transport phase. The main requirements of the controller are:

- **Distributed control:** There is no leader or centralized control. Each robot relies only on the package destination and information shared with the other robots.
- **Formation stability:** The formation shape must remain stable, even in critical phases, such as near obstacles.
- **Robustness:** The controller must remain effective despite potential communication losses between robots.

For the controller to operate, each robot must know the desired final position for the package delivery in one of the three outbound zones. Additionally, each robot needs, at every iteration (when communication is successful), the estimated position of the other robots involved in the transport and also informations about their proximity to obstacles. To allow for a more challenging scenario, communication is probabilistic: the i-th robot can communicate with the j-th one only with probability *prob_comm*. In all simulations, we chose *prob_comm* $= 0.8$ to test the reliability of the control.

The control law combines three main objectives, each weighted by a parameter reflecting its relative importance:

1) Formation keeping
2) Obstacle avoidance
3) Target position

These three tasks are investigated below.

1) Formation Keeping: The goal of formation control is to maintain a circular configuration of robots around the package, independent of the number of agents. Let p_i and p_j denote the positions of the i-th and j-th robots, and $d_{ij} = \|p_i - p_j\|$ their distance. The desired distance between robots is computed as

$$d_{ij}^{\text{des}} = 2R_{\text{form}} \sin\left(\frac{\pi k_{ij}}{N}\right),$$

where R_{form} is the formation radius, N the number of robots, and k_{ij} the separation index along the circular formation. The formation control term is then

$$u_{\text{form}} = -\sum_{j \neq i} (d_{ij}^2 - {d_{ij}^{\text{des}}}^2)(p_i - p_j),$$

which generates attractive or repulsive inputs between a couple of robots to maintain the correct spacing for the formation.

2) Obstacle Avoidance: Obstacle avoidance ensures that the formation does not collide with static obstacles during transportation. Each robot evaluates obstacles within a sensing range s_r and computes a repulsive control vector

$$u_{\text{obs}} = \sum_{o \in \mathcal{O}} \phi(d_{io})\, \hat{d}_{io} + \phi_t\, \hat{t}_{io},$$

where:

- d_{io} is the distance between robot i and obstacle o,
- $\hat{d}_{io} = \frac{p_i - p_o}{\|p_i - p_o\|}$ is the normalized direction away from the obstacle,
- $\hat{t}_{io}$ is a tangential unit vector, oriented perpendicular to $\hat{d}_{io}$ in the direction of the target to prevent motion stops,
- $\phi(d)$ is a repulsive potential function defined as

$$\phi(d) = \phi_0 \exp\Big(-\frac{(d - d_{\text{rep}}/2)^2}{2\sigma_{\text{rep}}^2}\Big),$$

where ϕ_0 is the maximum repulsive intensity, and d_{rep}, σ_{rep} are parameters that can be tuned.

The tangential component is scaled by a factor $\alpha < 1$ to ensure smooth lateral motion around the obstacle:

$$\phi_t = \alpha\, \phi(d).$$

It has to be noticed that robots not involved in the transport phase can be regarded, from a physical perspective, as moving obstacles. However, the transport formation has priority over surrounding agents, and these individual robots are self responsible for avoiding the moving formation using their Voronoi-based collision-avoidance strategy. This ensures that the formation is able to maintain its shape and path towards the target without being hindered by other agents in the environment.

3) Target Position: Finally, the formation is attracted towards the delivery target. The attraction component is defined as

$$u_{\text{att}} = \frac{p_{\text{target}} - p_c}{\|p_{\text{target}} - p_c\|},$$

where p_c is the current estimated center of the formation. If communication with other robots fails, a robot can instead use its own position relative to the target.

4) Total Control Input: The three contributions are combined using their respective weights:

$$u = w_{\text{form}} u_{\text{form}} + w_{\text{obs}} u_{\text{obs}} + w_{\text{att}} u_{\text{att}}.$$

The desired next position for each robot is computed as

$$p_{\text{des}} = p_i + \frac{\Delta t}{u_{\max}} u,$$

where Δt is the control time step and $u_{\max}$ the maximum allowable linear velocity. This reference target position is then passed to another function , namely *ROB_control*, which computes the unicycle inputs in terms of linear and angular velocities.

V. SIMULATION OVERVIEW

In this section of the report we highlight in details the actual Matlab implementation of the previously discussed techniques in a typical simulation of the system, referencing to what is already described in section II for what concerns the workflow.

A. Spawn

First it is defined a map, an inbound zone and three outbound zones as polygons of the proper dimensions. The number of anchors is chosen and their location is randomly generated inside the map, assuming no positional constraints since we consider that these objects could in practice be hanging from the ceiling. A procedure then generates a fixed number of static obstacles inside the map, at a minimum distance from one another, choosing randomly between square and circular shape. The position of these obstacles is constrained to the horizontal mid section of the map. A list of an arbitrary number of packages is then generated inside the inbound zone in random positions. Finally, the robots are spawned close to the inbound zone, in order for the next step to work properly. The system's state after the spawn phase is depicted in figure V-A, in which only the first package is shown.

B. Package localization

If the package falls within the sensing range of a robot, that robot can detect it and contributes to estimate its position. All robots that can sense the package then cooperate through the consensus algorithm to refine and agree on a common estimate of its position. In figure V-B it is shown how the estimate of the position of the package is refined at every iteration of the consensus algorithm for all robots that can see the package; the end result is registered and then broadcasted in the agents' network so that the system can proceed with the next phase.

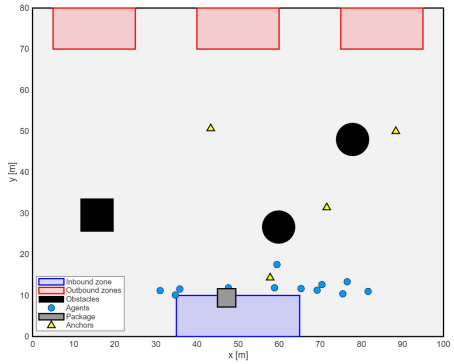

Fig. 6. Spawn

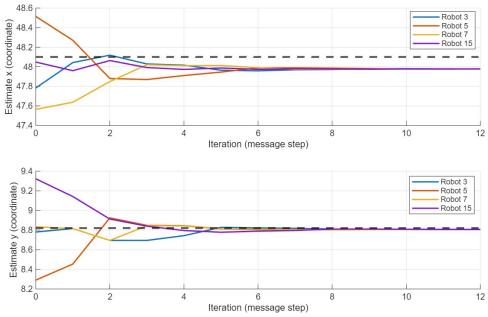

Fig. 7. Package position estimate

C. Ring line-up

In this stage, the actual dynamics of the system begins to run: the Voronoi tasselation is computed, and each robot is assigned its target based on its position. The N_r robots that are the closest to the package, with N_r being the number of requested robots, depending on the package's surface area, begin advancing towards it (figure V-C) attracted by their "ring-shaped" pdf. At each time instant, the system is checked and the positions of the line-up robots are controlled with respect to the package: once the line-up condition is satisfied (within a tolerance), these robots switch their working state to 't' and their Voronoi-based control is turned off in favor of the formation control. In figure V-C the grey lines (trajectories) and dots (initial and final positions) refer to the "free" agents, while the orange ones are the line-up robots.

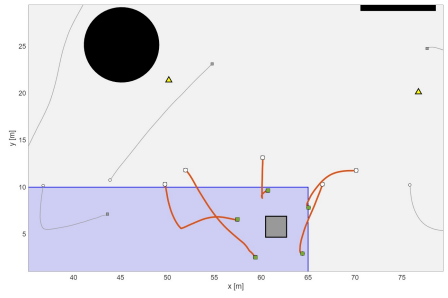

Fig. 8. Line-up phase

D. Delivery

Once the line-up robots are aligned with respect to the initial ring, the delivery phase can start. The robots in the formation adopt the formation control algorithm, switching off the computation of the centroid of their cell while keeping active the Voronoi partitioning in order to provide other robots information that is useful for implementing collision avoidance (Lloyd-Voronoi). In this phase, the formation has priority over all other robots, in the sense that their motion is not restricted by the presence of other agents, and the collision avoidance is guaranteed by the "free" robots. The delivery proceeds as the formation escorts the package to the drop-off target point, avoiding static obstacles via the "formation control" algorithm. In figure V-D, the green lines and dots refer to the trajectories and positions of the formation agents.

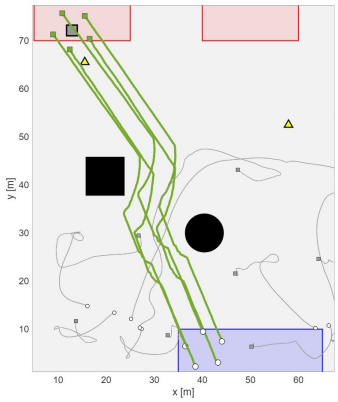

Fig. 9. Delivery phase

E. Drop-off

The delivery phase ends when the package has reached the target: this condition is checked at every time instant during the simulation. At this point, the formation splits apart, dropping the package, and returning to working state 'i': this way, they will get closer to the inbound zone, ready to proceed with the localization of the newly available package.

VI. Results

In this section of the report the results of the simulations are analysed, in terms of performance of the estimation algorithms and robustness of the controls.

A. Package localization

Concerning the package localization, the linear consensus algorithm gives reliable results provided that a sufficient number of robots sees the package. The results in table VI-A were obtained by fixing the sensing radius of the robots while navigating (thus creating a similar scenario for all simulations) and imposing a progressively larger sensing radius in the phase of package localization for every simulation, in order to analyse the effect of the number of robots that can see the package, provided a sufficient number of iterations for the algorithm in order to guarantee convergence. The error is the euclidean norm between the actual position of the package and the estimated one.

	PKG 1 [m]	PKG 2 [m]	PKG 3 [m]	MEAN [m]
2 robots	0.35	0.38	0.46	0.39
3 robots	0.29	0.21	0.38	0.29
4 robots	0.37	0.28	0.19	0.27
5 robots	0.09	0.14	0.26	0.16
6 robots	0.15	0.09	0.1	0.11
All robots	0.02	0.08	0.05	0.05

TABLE I
PACKAGE LOCALIZATION ERROR VS NUMBER OF ROBOTS PARTICIPATING IN CONSENSUS

It is evident that the number of robots participating in the consensus algorithm influences the quality of the package estimated position.

B. Robot's state estimation

For the purpose of analysing the performance of the implemented WLS and EKF algorithms, the Matlab code registers a log file per each simulation containing useful information on the state estimate. Figure VI-B highlights the trend in time (iterations) of the state estimation error in terms of position error (above) and orientation error (below).

Fig. 10. Robot's state error

In figure VI-B instead, the trace of the state's covariance matrix is reported. The trace of P was chosen as a significant indicator of the overall estimation algorithm performance as it's the sum of the variances $tr(P) = \sigma_x^2 + \sigma_y^2 + \sigma_\theta^2$, thus it truly indicates the amount of uncertainty on the estimated state of a robot.

The graphical analysis of this plot highlights many interesting aspects:

- **Trend**: the overall trend is that of a decreasing uncertainty. As expected, at first, the WLS algorithm gives an initial, rather "rough" estimate of the robot's position, then the EKF rapidly refines it further on in the simulation.
- **"Zig-zag" pattern**: with reference to the red box in figure VI-B, the trace of P periodically raises and

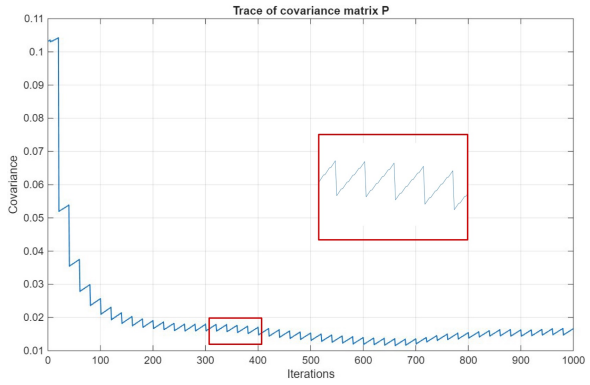

Fig. 11. Trace of state's covariance matrix

decreases in time as a consequence of the fact that the update step of the EKF algorithm can only be computed by the time a new set of measurements comes from the sensors, whereas the prediction step can be computed at each instant. This is coherent with theory, since the prediction steps increases the overall uncertainty on the estimate, whereas the update decreases it. The resulting "steady state" value of $tr(P)$ is a sort of average between the effect of the two steps.

In table II it is reported the mean error on position, computed as the euclidean norm between the estimate and real position, and the mean error on the orientation of the robot for five simulations.

	Position error [m]	Orientation error [deg]
SIM 1	0.07	6.30
SIM 2	0.06	6.88
SIM 3	0.11	2.27
SIM 4	0.11	2.72
SIM 5	0.07	3.60
MEAN	0.08	4.36

TABLE II

STATE ERROR METRICS OVER 5 SIMULATIONS

C. Formation control performances

To evaluate how well the formation control strategy performs (provided safe navigation), we computed a formation error that represents how far the actual formation deviates from the desired one during the initial phase of the transport. For this analysis, we considered the first 1000 iterations of the simulation.

The metric used at each time step is based on the absolute distance error between every pair of robots, defined as:

$$form_{err} = \frac{1}{N_{pairs}} \sum_{i \neq j} |d_{ij} - d_{ij}^{des}| \tag{31}$$

Where N_{pairs} is the total number of robot pairs.

Several configurations were tested to allow comparisons across different scenarios. Specifically, we kept the formation radius fixed at $2.5m$ and varied the number of robots involved in the transport task. The simulations showed that the formation error varied significantly depending on the trajectory, with the largest deviations occurring when the formation passed close to obstacles.

To ensure a fair comparison, we ran 10 simulations for each configuration and calculated the mean formation error across these runs. The results are reported in table VI-C.

Number of robots	Average formation error [m]
3	0.1254
4	0.1017
5	0.1069
6	0.0864
7	0.1003

TABLE III

AVERAGE FORMATION ERROR FOR DIFFERENT NUMBERS OF ROBOTS

Interestingly, no clear correlation emerges between the number of robots and the resulting formation error. This suggests that simply increasing or decreasing the team size does not directly improve or degrade formation accuracy under the tested conditions. It should also be noted that all simulations were performed using the same default controller weight values. In practice, these parameters should be tuned for each specific configuration to achieve optimal performance.

Nevertheless, the results highlight a positive aspect of the proposed controller: they prove good robustness and scalability. Despite the lack of parameter tuning, the controller maintains stable behaviour and acceptable error levels across all tested team sizes.

VII. CONCLUSIONS

In this work, a distributed control framework for a swarm of mobile robots was presented. A Weighted Least Squares and an Extended Kalman Filter were adopted for state estimation, a Voronoi-Lloyd based partitioning and control algorithm was implemented for navigation and a linear consensus algorithm was chosen for package localization. Moreover, a formation control algorithm was adopted for maintaining a predefined geometric configuration during the transportation phase. The conducted simulations produced convincing results under sensing and communication constraints, with robust behaviour with respect to the imposed changes in the environment (obstacle spawn) and initial conditions (robots and package spawn).

A. Possible improvements

The system proved to be relatively robust, although some key aspects could be more deeply investigated and improved, for example:

- **Dynamics**: the model does not consider any type of force acting on the elements. In the real world, robots exchange forces with the environment and with the package. The choice of not including this phenomenon was made since the focus of this project was investigating and providing a solid initial framework for distributed control, navigation and scheduling of a

swarm of robots for the goal of warehouse management; nevertheless, this aspect could be modelled for more reliable and realistic results.

- **Formation control**: the current algorithm was fine-tuned for the specific role of transportation and obstacle avoidance in the presence of only two types of static obstacles. This approach performs well under rather specific conditions, but more advanced solutions, such as MPC, could provide better performance and should be investigated as possible future improvements.
- **Voronoi-Lloyd based control**: for the purpose of this project of investigating control algorithms for distributed systems, the Voronoi-Lloyd based control was implemented with limited fidelity with a real-world application, in the sense that the Matlab code does not model the scanning and ray-casting processes with realism, thus the real-world implementation of this algorithm should in practice be managed using a different approach. Nevertheless, this work serves as a proof of concept for possible future applications, therefore realism was not considered as a constraint to the development of the simulations.
- **Workflow and logistics**: in the simulation, the packages are delivered one at a time, in a "sequential" fashion. From a logistical perspective, it would make sense that the robots which are not delivering a package (if available in number) could begin the delivery of the next one, identifying its position in the inbound zone, lining up and picking up the package. This however was not implemented since the focus was put on providing a solid framework from a distributed control standpoint, considering logistics as of secondary importance. In a real implementation however this aspect may be of more paramount importance.

REFERENCES

[1] Tuncer Can Aysal, Mehmet Ercan Yildiz, Anand D Sarwate, and Anna Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal processing*, 57(7):2748–2761, 2009.

[2] Manuel Boldrer, Luigi Palopoli, and Daniele Fontanelli. Lloyd-based approach for robots navigation in human-shared environments. pages 6982–6989, 10 2020.

[3] Colin Sauzé and Mark Neal. A raycast approach to collision avoidance in sailing robots. 06 2010.