agriculture

MDPI

Article

Satellite-Enabled Two-Tier UAV Vineyard Inspection with Multispectral Smart Sampling and Adaptive Path Planning

Konstantinos Konstantoudakis [1], Kyriaki Christaki [1], Tomaso de Cola [2,*], Roshith Sebastian [2] and Gayathri Guruvayoorappan [2]

[1] KyberKore P.C., 54624 Thessaloniki, Greece; kostis@kyberkore.com (K.K.); kiki@kyberkore.com (K.C.)
[2] Institute of Communications and Navigation, Deutsches Zentrum für Luft- und Raumfahrt (DLR) Oberpfaffenhofen, 82234 Wessling, Germany; roshith.sebastian@dlr.de (R.S.); gayathri.guruvayoorappan@dlr.de (G.G.)
* Correspondence: tomaso.decola@dlr.de

Abstract

Vineyard monitoring requires efficient methods for detecting plant stress and disease while limiting flight time, data volume, and labour effort. This paper presents a satellite-enabled two-tier UAV workflow for semi-automated vineyard inspection. The proposed approach combines high-altitude multispectral scanning, NDVI-based point-of-interest identification, adaptive flight path planning, low-altitude RGB inspection, and downstream vision-based disease analysis. Processing tasks are offloaded to a remote server accessed through an emulated Low Earth Orbit satellite communication environment, allowing the UAV-side system to remain lightweight while receiving multispectral analysis results during the mission. A simulation framework was developed to evaluate mission behaviour under controlled and repeatable conditions, using both pseudo-random point generation and real multispectral vineyard images processed through the satellite emulation testbed. A flight with a real drone was also conducted to validate adaptive flight optimisation. Experimental results focus on the impact of path-adaptation strategies and communication bandwidth on mission efficiency. The results show that route optimisation can reduce mission time by up to 15% when new low-altitude waypoints emerge, while bandwidth bottlenecks affect performance once image transmission can no longer keep pace with acquisition. The findings highlight the need to consider sensing, communication, and mission planning jointly in adaptive UAV-based crop monitoring.

Keywords: precision viticulture; unmanned aerial vehicles; satellite communications; multispectral imaging; NDVI; adaptive path planning; computer vision; plant disease detection

Academic Editor: Lijun Wang

Received: 1 July 2026
Revised: 3 August 2026
Accepted: 13 August 2026
Published: 15 August 2026

1. Introduction

Viticulture is a high-value agricultural sector with both economic and cultural significance in many regions. Vineyards are exposed to a range of pressures, including disease, water stress, and climatic variability. Efficient monitoring is essential for informing timely interventions and management decisions, while keeping labour costs and effort low. Automated and data-driven monitoring approaches offer an increasingly practical path to vineyard monitoring, combining sensing, communication, and analytics for decision support. Relevant technologies include satellite imagery and communications, drone mapping, multispectral vegetation indices, and computer vision.

Satellite technologies can support crop monitoring both directly, through Earth observation, and indirectly, through communication. Satellite imagery can provide repeated,

wide-area observations of vineyard condition and has been used to support precision viticulture by mapping vegetation vigour, water status, spatial variability, and other relevant indicators for improved sustainability and operational efficiency [1]. As data-driven approaches gain traction in agriculture, satellite communication is also becoming increasingly relevant, especially for remote regions where terrestrial connectivity may be unreliable or unavailable [2].

Drones provide a flexible and high-resolution sensing platform for precision agriculture, complementing the wider coverage of satellite imagery with detailed observations for crop mapping, plant-health assessment, irrigation support, yield estimation, spraying, and other monitoring and management tasks [3]. Drone-based multispectral mapping can provide vegetation indices, such as NDVI, which are widely used to quantify vegetation condition, growth, and stress [4]. Previous vineyard-focused research has examined such data for specific applications including water-status variability mapping and disease progression monitoring [5,6]. Nevertheless, UAV monitoring involves an inherent trade-off between coverage and detail: high-altitude flights can map larger areas efficiently, while low-altitude inspection can provide more detailed visual information but requires additional flight time and produces larger data volumes. At a broader monitoring scale, drone imagery is therefore often considered alongside satellite imagery, combining wide-area observation with finer spatial detail [7].

Computer vision and machine learning further expand the role of image-based crop monitoring by enabling automated interpretation of RGB, multispectral, and other data. In agriculture, these methods have been applied to a broad range of tasks, including plant and canopy segmentation, stress detection, disease identification, and yield estimation. For disease monitoring specifically, recent reviews highlight the growing use of UAV imagery, deep learning, and remote or proximal sensing tools for early detection and field-scale assessment [8–10]. In grapevines, recent work includes both leaf-level disease classification using convolutional neural networks and field-level disease monitoring using UAV multispectral data [6,11].

Building on these developments, this work presents a satellite-enabled, two-tier UAV workflow for semi-automated vineyard inspection. The proposed approach combines high-altitude multispectral scanning, identification of points of interest from vegetation index analysis, live flight path adaptation, low-altitude RGB inspection, and vision-based disease detection, classification, and severity estimation. Processing tasks are offloaded to a remote server reached through an emulated satellite communication environment, allowing the UAV-side system to remain lightweight while receiving multispectral analysis results during the mission. These results are used to adapt the flight path and guide selective low-altitude inspection of areas that may indicate plant stress or disease.

Unlike existing approaches that typically address multispectral crop monitoring, close-range disease inspection, communication constraints, or UAV path planning separately, this work integrates these elements into a single adaptive inspection workflow. The main contributions of this paper are as follows:

- A two-tier UAV inspection architecture combining high-altitude multispectral sensing, NDVI-based smart sampling, low-altitude RGB inspection, and remote image processing.
- On-the-fly path adaptation for inserting newly identified waypoints into an ongoing UAV mission, together with a comparative evaluation of alternative insertion and route-optimisation strategies.
- An experimental analysis of how satellite-link bandwidth affects image-transfer latency, waypoint availability, and total mission duration during adaptive inspection.

The experimental evaluation in Section 3 focuses on the effect of different path-adaptation strategies and communication bottlenecks on mission efficiency. A detailed eval-

uation of point-of-interest identification as a smart sampling mechanism for low-altitude image acquisition is beyond the scope of this work and will be addressed in future research. The disease-detection component is included as a downstream analysis module based primarily on established computer vision approaches, and is therefore not benchmarked as a separate contribution in this paper.

2. Materials and Methods

2.1. System Architecture

The overall system architecture and information flow are shown in Figure 1. The drone captures the images from the vineyard and sends them to the remote server for processing. The system incorporates high-altitude multispectral scanning, dynamic mission adaptation, and computer vision-based disease assessment. It is deployed as two interacting parts:

1. A *locally deployed application* that interfaces with—and controls—the drone: For missions with a real drone, this is an Android app leveraging an SDK to communicate with the drone and deployed on the drone remote controller; for the tests presented in this work, a simulator was used instead, to facilitate repeat testing, as described in Section 2.6.
2. A *remote server* that handles image processing and computer vision tasks: In this work, a server hosted in a VM within the satellite emulation testbed presented in Section 2.2 was used.

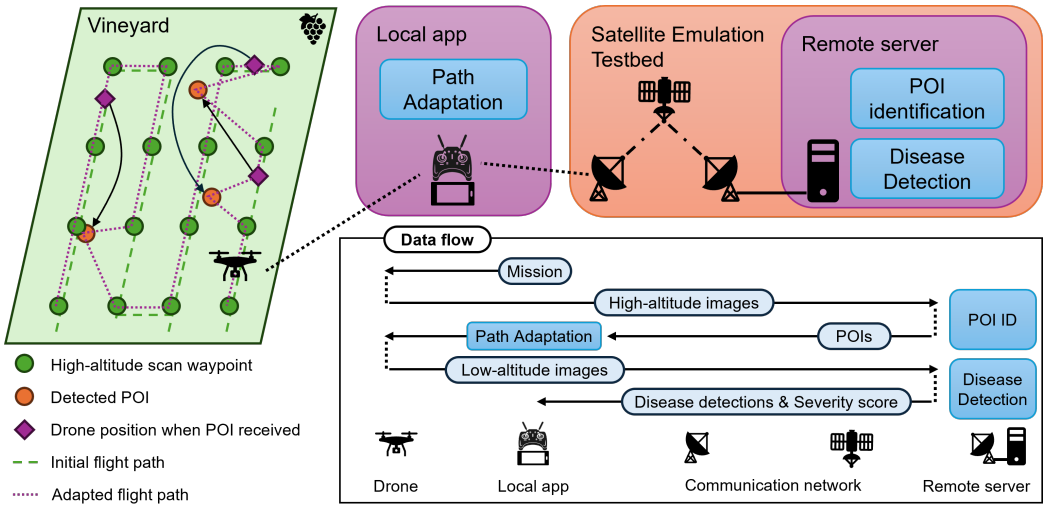


Figure 1. System Architecture and data flow.

Furthermore, the system comprises three main software modules, each with a distinct input and output, as shown in Table 1.

Table 1. System software modules.

Module	Deployment	Input	Output(s)
POI identification	Remote	Red and NIR multispectral bands	POI coordinates
Path adaptation	Local	List of new waypoints	Updated drone path
Disease detection	Remote	RGB image of grapevine	Severity score

The flow of information between the modules, during the course of a mission, is as follows:

Initially, a mission is created according to user input, comprising only high-altitude waypoints, placed evenly on a grid so as to cover the whole vineyard. At each high-altitude waypoint, the drone captures a multispectral image. The local application sends the pair of Red and NIR bands to the remote server. There, the *POI identification* module (Section 2.3) receives and processes them, and sends back the coordinates of any POIs identified. The *Path adaptation* module (Section 2.4) inserts these POIs into the existing path, adapting and optimizing it to minimise its total length. When a POI is visited, the drone captures a close-up RGB photo of the grapevine. All RGB close-ups are sent to the remote server after the mission for *Disease detection* (Section 2.5).

The satellite constellation is transparent and serves the purpose of providing connectivity between the drone and the ground node hosting the remote server.

2.2. Satellite Emulation Testbed

The satellite emulation testbed is capable of emulating a configurable number of satellites in an orbital plane. In this work, it is used to reproduce the behaviour of a Low Earth Orbit (LEO) satellite constellation with Inter-Satellite Links (ISLs), providing the communication environment between the drone-side system and the remote processing server.

The testbed was developed by DLR for research purposes using Mininet [12], a network emulator that can create networks of virtual hosts, switches, controllers, and links. In contrast to purely analytical or discrete-event simulators, Mininet networks run real code, including standard Unix/Linux network applications and the Linux kernel network stack. As a result, software developed and tested in Mininet can be transferred to real systems with minimal changes, making it suitable for network virtualisation, performance evaluation, and deployment-oriented testing.

In the emulated satellite network, ground nodes connect to the nearest visible satellite. Depending on the constellation configuration, communication between two ground nodes may traverse more than one satellite, making ISLs important for end-to-end connectivity. The performance of the communication path depends on the links between ground nodes and satellites, as well as on the ISLs between satellites. For this reason, the testbed supports multiple configurable parameters, including the following:

- The height of the orbital plane, which affects both the delay between a ground node and the nearest satellite and the duration of satellite visibility with respect to the ground node;
- The number of satellites between the source and destination ground nodes;
- Ground-to-satellite link parameters, such as bandwidth and packet loss rate;
- ISL parameters, including delay, bandwidth, and packet loss rate.

Although the testbed can support two orbital planes, only one orbital plane was considered in this work for simplicity. The testbed can also emulate orbits other than LEO, provided that the relevant parameters are selected carefully to avoid unrealistic scenarios. Communication between nodes is based on Internet Protocol (IP) routing, and both Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) are supported.

The experimental setup is shown in Figure 2. The emulated satellite network runs inside a Virtual Machine (VM) on Ubuntu 22.04, which is accessed through a Virtual Private Network (VPN) connection. The drone-side system connects to this VPN and sends data to the VM. Upon receiving the data packets, the VM forwards them to the satellite emulation framework. Packets are then forwarded through the emulated satellite path, from the nearest satellite toward the gateway or destination ground node. In this work, end-to-end communication was carried out using TCP, and four satellites were modelled along the path between the drone and the ground node. The ground node hosts the Docker container

running the remote processing service. After the images are processed, the results are returned through the same emulated communication path in the reverse direction.

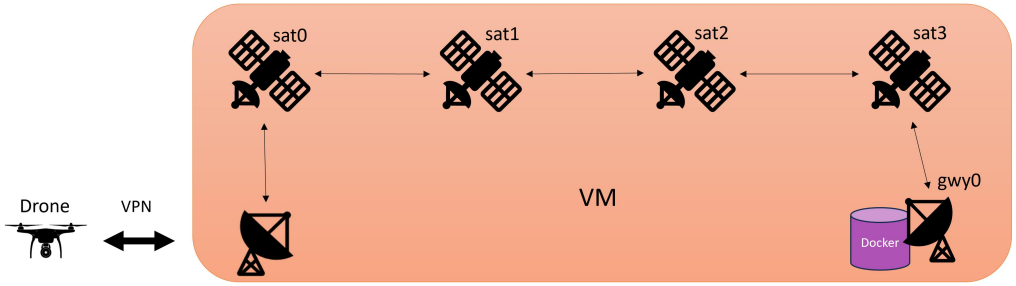


Figure 2. Experimental testbed configuration.

Satellite emulation is considering a Starlink-like constellation. Hence, the ground nodes have continuous connectivity since at least one satellite is visible. This ensures that the ground nodes can always communicate without the need of a storage mechanism on the ground nodes. Considering that TCP is employed in the transport layer it is essential that the end-to-end connectivity is always present. Otherwise, the docker present on the other end of the constellation will not be readily reachable, which can adversely affect the overall performance.

The links between satellites and ground nodes, namely the drone and gateway, are modelled based on the orbital height of the satellite constellation and the relative movement between satellites and ground nodes. Link delay is highest when the satellite appears at the horizon of the ground node. It then decreases as the satellite moves toward the point above the ground node, before increasing again until the satellite is no longer visible. Satellite visibility duration also depends on orbital height, with higher orbital planes resulting in longer visibility duration. In the scenario considered in this work, the orbital height was set to 1221 km.

2.3. POI Identification

2.3.1. POI Identification Rationale and Objective

Scanning an area, or a vineyard in the present case, is a trade-off between flight altitude and effective ground resolution. The ground area captured by a single photo (also known as the *footprint*) depends on the altitude of flight, as well as the camera's field of view (FOV). For simplicity and uniformity, in this paper we are considering only photos taken at nadir orientation (i.e., the camera is pointing straight down).

For a camera with $FOV_H \times FOV_V$ field of view and the drone flying at altitude A, the footprint is a rectangle of dimensions $X \times Y$:

$$\begin{aligned} X &= 2A \tan(FOV_H/2) \\ Y &= 2A \tan(FOV_V/2) \end{aligned} \tag{1}$$

The ground sampling distance (GSD) describes the distance between adjacent pixels and is equal to the footprint size divided by the camera resolution. For a camera with a $W \times H$ resolution:

$$\begin{aligned} GSD_x &= 2A\frac{\tan(FOV_H/2)}{W} \\ GSD_y &= 2A\frac{\tan(FOV_V/2)}{H} \end{aligned} \tag{2}$$

For a given drone model, the camera specifications are constant and GSD is reduced to a function of altitude, which can be derived from either the horizontal or the vertical GSD and resolution, equivalently. For the camera of a modern commercial drone (in this work, experiments were conducted using the DJI Mavic 3M (DJI, Shenzhen, China)), with a vertical resolution of 3956 pixels and a corresponding vertical FOV of 48°, the altitude required to achieve a desired GSD is:

$$A = \frac{1}{2}\frac{\tan(48/2)}{3956} GSD \approx 4443 \cdot GSD \quad (3)$$

Grapevine leaves average about 20 cm in size, while discolouration spots that can point to disease are often about 1 cm. In disease detection datasets such as [13–15], discolouration spots that signify disease are usually 7–20 pixels wide. Hence, targeting a similar resolution and assuming that spots would be clearly identifiable if they are at least 10 pixels wide, at a GSD of no more than 0.001 m per pixel would be required, pointing at a maximum altitude of about 4.4 m for disease detection. This GSD is also in line with relevant research on unhealthy leaf detection in the wild [16]. This corresponds to a footprint roughly 5.2 m × 3.9 m, or an area of 20 m^2, meaning that even a relatively small vineyard of 3–4 hectares would require thousands of photos to scan exhaustively.

Non-exhaustive scans, or sampling, can provide insights regarding disease presence while keeping the number of images, the drone flight time, and the energy expenditure manageable. POI identification's goal is to maximise the efficiency of low-altitude image sampling by identifying the most interesting, from a plant-health perspective, points in the vineyard.

2.3.2. Approach

Previous research has shown that NDVI bears a strong correlation with plant health and can be used to identify relevant risks [6,17]. Crucially, NDVI-based insights do not rely on specific details but on the average value of a plant or vineyard area, and hence can be acquired with a greater GSD (e.g., 0.02–0.05 m per pixel), at higher altitudes. Accounting for the comparatively lower resolution of multispectral cameras, this corresponds to an altitude of 45–110 m.

Hence, NDVI images constructed from high-altitude multispectral photos can provide insights regarding plant health. In this work, such photos are used to drive a smart sampling scheme, in which spots are identified as points of interest (POIs) based on their NDVI value, and are marked for low-altitude image acquisition.

2.3.3. NDVI Calculation and Correction

The NDVI is computed from the Red and Near Infrared (NIR) bands:

$$NDVI = \frac{NIR - Red}{NIR + Red} \quad (4)$$

While the NIR and Red bands are acquired as part of the multispectral photo, they each have slightly different fields of view, due to the placement of the sensors on the drone. In addition, both images have slight vignetting distortion. Before computing the NDVI, band images must be undistorted and aligned:

To compensate for vignetting distortion, the vignetting polynomial coefficients are extracted from each image's metadata and used. For alignment, Oriented FAST and Rotated BRIEF (ORB) [18] is used to detect keypoints and compute compact binary descriptors in the Red and NIR images. ORB was selected because it combines low computational cost with robustness to the small rotational differences between the two bands, making it suitable for repeated processing of high-resolution multispectral images, while its binary

descriptors can be matched efficiently using Hamming distance. A brute-force matcher compares the descriptors between the two bands, with cross-checking applied to retain only mutual best matches. The matches are sorted by distance, and the best 50 correspondences are used to estimate an affine transformation that aligns the NIR band to the Red band.

2.3.4. Grapevine Row Identification and Segmentation

From each NDVI image depicting a patch of the vineyard, individual segments of plants, corresponding to potential POIs, must be identified and the mean NDVI of each must be calculated:

Firstly, the NDVI images show rows of grapevines as well as soil. To separate the grapevines from the background, a binary mask is applied to NDVI with a threshold of 0.35. This threshold is in line with previous research, which suggests thresholds between 0.35 and 0.4, depending on plant development [19]. Grapevine rows are identified as continuous areas in the binary mask. Finally, each row is divided into smaller segments. Segmentation is performed along each row's long axis, by lines perpendicular to it. The mean NDVI of each segment is then computed. Figure 3a–e shows the above process, from NDVI to binary mask to segments.

In Figure 3e, one can observe a smooth radial intensity pattern, with segments further away from the centre exhibiting lower NDVI values. The same pattern appears in both the Red and NIR band images, although it is less discernible. This is further apparent in the binary mask itself (Figure 3b), which fails to recognise grapevines in the image corners because of reduced NDVI values.

This effect is likely due to lens shading (vignetting) and other residual radiometric non-uniformities in the imaging chain, which cause a gradual drop in measured intensity away from the centre, as off-axis rays reach the sensor at steeper angles and with lower effective irradiance. To compensate for this, a radial polynomial model was defined using the average radial distortion in the acquired NDVI images and applied, following a well-established approach for correcting image vignetting [20], including on DJI Mavic 3M multispectral imagery [21]. This radial compensation results in a corrected NDVI calculation for segments (Figure 3f), which also correctly includes plants in the corners.

2.3.5. POI Identification Logic

Following the process described above, each patch of the vineyard captured by a single set of multispectral images is divided into small segments and the mean NDVI value for each is calculated. POIs can be identified based on anomalies in these values.

In general, two types of anomalies can be defined: NDVI lower or higher than expected. A lower NDVI points to drier leaves, which is often a sign of disease. Higher NDVI points to an area of increased humidity. Although not a sign of disease in itself, high-humidity areas can be more susceptible to fungal infections. In the present work, we focus on low-NDVI segments. Two checks are performed: one global (i.e., for the whole patch captured in the multispectral photo), and one local (i.e., comparing each segment with its immediate neighbours).

For *global POI identification*, the algorithm computes the mean NDVI of all segments and the deviation of each segment from that mean. A neutral zone NZ around the mean is defined (default 0.05). Any segments with $NDVI_{segment} < NDVI_{mean} - NZ$ are flagged as potential global POIs and ranked according to their deviation from the mean.

Local POI identification is defined similarly but only considers each segment's deviation from the mean of segments in its vicinity. For each segment, neighbouring segments are identified based on the distance of their centroids to the centroid of the segment in question.

If a segment NDVI is lower than that of all its neighbours (including a 5% margin), it is flagged as a potential local POI: $NDVI_{segment} < 0.95 \cdot min([NDVI_{neighbours}])$.

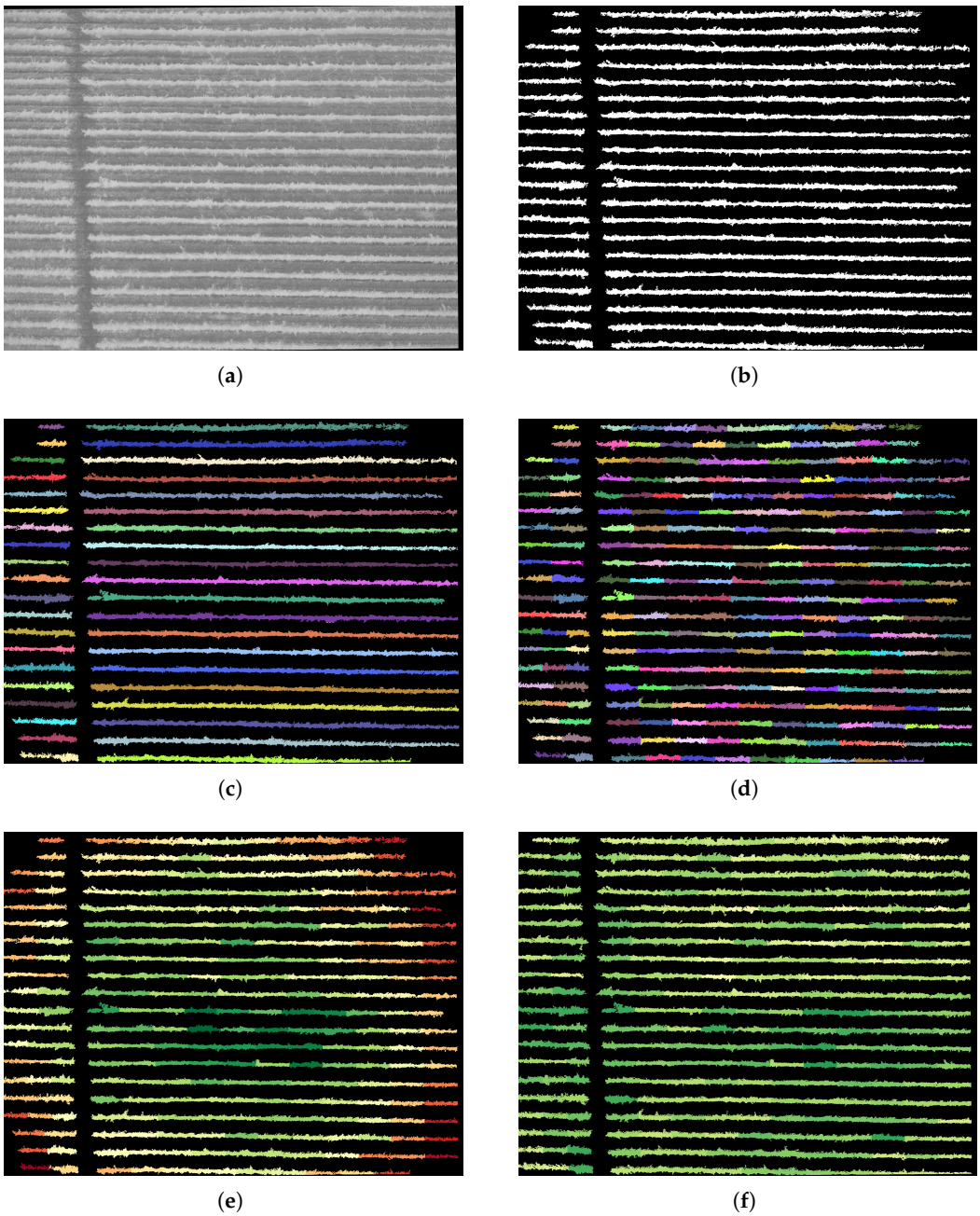


Figure 3. NDVI image segmentation: (**a**) The NDVI image of a patch of vineyard. (**b**) Binary mask separating grapevines from the background and identifying rows. (**c**) Identification in rows as continuous segments of the mask, each randomly colourised. (**d**) Division of each row into smaller segments, each randomly colourised. (**e**) Mean NDVI of each segment, with radial error apparent. (**f**) Mean NDVI of each segment after radial error compensation. In (**e**,**f**), mean NDVI is mapped to a red-to-green colour scale, with red indicating lower NDVI values and green indicating higher values.

Finally, two mechanisms are implemented to maximise POI impact: (1) the total number of POIs for each patch is capped at 3; and (2) potential POIs that are close to already selected POIs are excluded, to avoid dense sampling.

The final POIs for each patch are selected by alternating between selecting the highest-ranked candidate global and local segments, until the maximum number of POIs has been reached or no segments fulfil the criteria. The algorithm can be summarised as below:

1. Sort segments into two sorted lists:
   - *Global list* is sorted by NDVI, lowest first.
   - *Local list* is sorted by the difference $0.95 \cdot min([NDVI_{neighbours}]) - NDVI_{segment}$, highest first.
2. Pick the first segment from the *global list*. If $NDVI_{segment} < NDVI_{mean} - NZ$, select it as POI.
   - Remove the selected POI and all its neighbours from both lists.
   - If max number of POIs reached, stop algorithm.
3. Pick the first segment from the *local list*. If $NDVI_{segment} < 0.95 \cdot min([NDVI_{neighbours}])$, select it as POI.
   - Remove the selected POI and all its neighbours from both lists.
   - If max number of POIs reached, stop algorithm.
4. If step 2 or step 3 returned a POI, go to step 2.

Hence, each multispectral image set will generate 0–3 POIs for sampling, ensuring a minimum distance between them.

2.4. Path Planning and Adaptation

In order to conserve time and energy, it is important to optimise the flight path the drone will follow to cover its objectives.

2.4.1. Waypoint Type Definition

Drone missions are defined as an ordered list of waypoints (WPs). Each WP is defined by its 3D coordinates (latitude, longitude, and altitude). In addition, one or more actions, such as "take photo", may be associated with a WP, to be taken once the drone reaches it.

In terms of drone mission flight path, this two-tier vineyard inspection pipeline corresponds to two primary types of waypoints:

- *High-altitude WPs (HAWPs)* are the WPs necessary to scan the whole vineyard at high altitude (typically 50–100 m). Each HAWP is associated with an action to take a multispectral set of photos at nadir (straight-down) camera orientation, which is sent to the POI identification module to identify POIs in the area covered by this HAWP. All HAWPs are already known at mission start, as they depend on set parameters: the vineyard shape and size, the drone's multispectral camera's field of view, and the defined altitude. Assuming a rectangular-shaped vineyard, HAWPs will form an evenly spaced grid.
- *Low-altitude WPs (LAWPs)* correspond to POIs and are situated at very low altitude (3–5 m). LAWPs are associated with an action to take an RGB photo, which will be used for disease detection and classification. LAWPs are not known at mission start; they emerge following the POI identification process on a multispectral image set acquired at a HAWP. Hence, each LAWP corresponds to a "parent" HAWP and will only emerge some time after the drone has visited its parent. Each LAWP will be within the area covered by its parent, and hence it will be closer to its parent than to any other HAWP.

In addition to HAWPs and LAWPs, each mission must also include a *Start* and *Finish* WP. These are typically identical and are situated at a preset altitude above the drone's home point—its take-off and landing pad.

2.4.2. Problem Definition and Parameters

Assuming a constant flight speed and energy expenditure per unit distance, path optimisation reduces to minimizing the total travelled distance. It is a routing problem,

closely related to the Travelling Salesman Problem (TSP) family, which, in its simplest form, optimises the visitation order of a list of cities (WPs) so as to minimise the distance travelled by a salesman (drone) that needs to visit them all. It is worth mentioning that in the case of drone flight and waypoints at different altitudes, distance must be considered in three dimensions.

Since not all WPs are known from the beginning, the present problem most closely resembles Online TSP, in which new cities are revealed over time. However, two-tiered vineyard scanning diverges from Online TSP in that it is highly structured and constrained, limiting the solution space significantly:

- Grid: HAWPs are arranged in a grid.
- Spatial: all emerging LAWPs will be within the vineyard.
- Time: each LAWP will emerge some time after visiting its parent HAWP
- Exclusivity: once POI identification returns its results for a given HAWP, no additional LAWPs will emerge in that area.
- Predictability: stemming from the two points above, LAWP emergence is partially predictable at any given point during the mission: new LAWPs may only emerge near HAWPs that *have been visited but have not yet yielded results*.

Hence, rather than approaching it as a generic Online TSP, this work's approach is guided by these constraints. Three major parameters were considered for path optimisation and adaptation: the initial flight path, consisting only of HAWPs, leverages the grid and spatial constraint while also facilitating the revisitability of previously scanned areas; the WP insertion logic defines where new LAWPs are inserted in the existing path; and a selection of additional path optimisation processes optimise the path following each insertion.

Each parameter is described below, while its impact on the path length is evaluated in Section 3.

2.4.3. Initial Path Pattern

Before any LAWPs emerge, the initial path, consisting solely of HAWPs, resembles that of a mapping mission. It is well-established that the optimal path for this is a Boustrophedon path [22], with the drone moving in rows of alternately opposite direction. Assuming a rectangular vineyard, aligning these rows parallel to the longer axis of the vineyard is optimal, minimizing the number of turns needed. Regarding drone orientation, it is optimal to align the longer (typically, the horizontal) camera axis perpendicular to the Boustrophedon rows, maximizing the spacing between them, hence allowing coverage of the vineyard with a minimum number of scanning rows.

Anticipating the emergence of LAWPs, an important consideration of the initial HAWP path is area revisitability, i.e., the proximity of the future WPs of the path to its previously visited HAWPs—and hence to possible emerging LAWPs. Higher area revisitability can allow for the easier (less costly in terms of additional distance) insertion of future LAWPs.

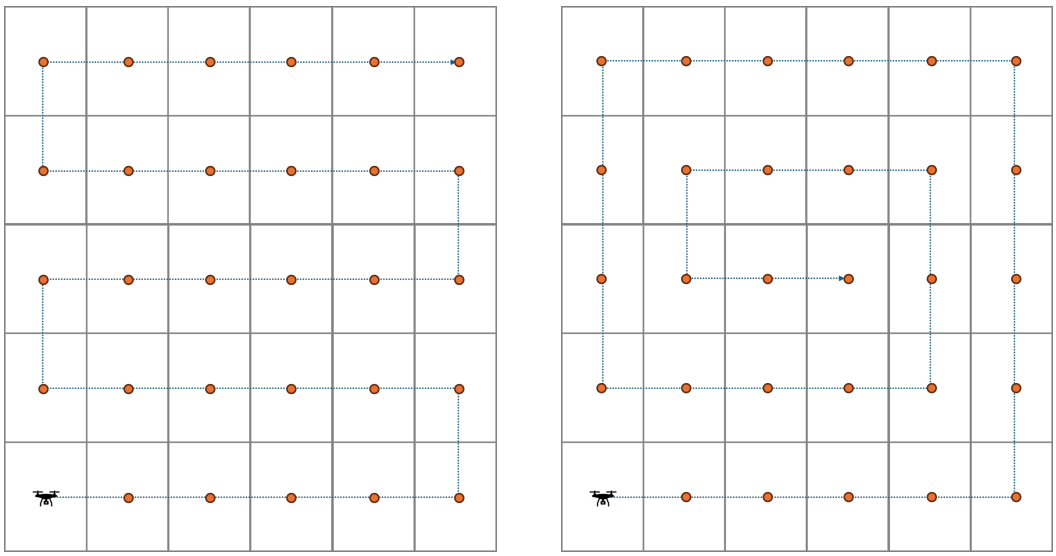
Importantly, the time lag between HAWP visitation and POI result reception depends primarily on the connection bandwidth to the device performing POI identification, and can range from 2 s to over a minute. This means that the drone will typically have time to move significantly away from a HAWP, continuing on its route, before the emergence of LAWPs necessitates the visitation of the same area again. Hence, when considering revisitability, it is important to exclude WPs in the drone's immediate future and only include WPs that are expected to be visited after the estimated POI result time lag.

The Boustrophedon path already has some revisitability potential, although the alternating direction of the path along parallel rows makes for an uneven distribution regarding time: the HAWP in the beginning of row 1 will be approached again at the end of row 2,

with a time difference corresponding to travelling two full rows. On the contrary, although the HAWP at the end of row 1 is close to the beginning of row 2, the time difference is short and perhaps insufficient in many scenarios. The next approach will be at the end of row 3, at a longer distance.

In comparison to the classic Boustrophedon pattern, a Spiral pattern was also examined. As this executes more "sideways" moves than the Boustrophedon, it is not optimal in distance travelled. However, it has better revisitability properties, with the spiral path approaching all quadrants of the vineyard periodically, with more consistency and longer delay.

Both Boustrophedon and Spiral patterns are shown in Figure 4.



(**a**) (**b**)

Figure 4. Initial flight path patterns: (**a**) Classic Boustrophedon pattern minimises distance, with inconsistent revisitability delay. (**b**) Spiral pattern has a longer path, with more consistent area revisitability with a longer delay.

2.4.4. Waypoint Insertion Logic

New LAWPs must be inserted into the flight path in a way that minimises its total length. In the broader literature on dynamic routing [23,24], a range of different strategies have been applied, including periodic re-optimisation of the full route, incremental insertion heuristics that integrate newly revealed waypoints into the current plan, rolling-horizon schemes that re-plan only a limited future segment of the route, and, more recently, reinforcement learning [25]. The present work follows an incremental insertion approach, reducing the significant changes in initial path pattern and leveraging its structure and revisitability potential. Two main approaches are considered:

In *end insertion*, LAWPs are always inserted immediately after the final HAWP, resulting in two distinct "passes". The forward pass covers all HAWPs, following the initial pattern selected without altering it. In the return pass, LAWPs are visited in reverse order relative to their parent HAWPs, i.e., LAWPs derived from the last HAWP will be visited first (or as soon as they become available), while those derived from the first HAWP will be visited last. The return path will therefore resemble a reverse, less-ordered version of the forward path.

End insertion has three main advantages: (a) zero computational requirements; (b) predictability; and (c) WPs are grouped by altitude, which minimises altitude changes, reducing the overall path length. Disadvantages include a lack of optimisation, non-use of

revisitability potential, and low robustness to POI result delays, especially near the end of the forward pass.

In *minimum-detour insertion*, each new LAWP is inserted between any two consecutive existing WPs, so as to minimise the detour needed. Hence, in an ordered list of existing WPs, a new LAWP N will be inserted after the i-th WP, with i selected to minimise the detour:

$$\begin{aligned} \Delta_i(N) &= D(WP_i, N) + D(N, WP_{i+1}) - D(WP_i, WP_{i+1}) \\ i &= \arg\min_i \Delta_i(N) \end{aligned} \tag{5}$$

where $D(A, B)$ is the distance between points A and B and $\Delta_i(N)$ is the detour (added path length) caused by insertion of N immediately after WP_i.

In practice, in a typical minimum-detour insertion mission, some LAWPs will be inserted between HAWPs and covered in an extended forward pass, while others, either arriving late or not placed conveniently, will be covered in a shorter return pass. Minimum-detour insertion can capitalise better on path revisitability and adapt to POI emergence unpredictability. However, early LAWPs will often be placed between HAWPs, resulting in frequent altitude changes, which is especially prominent when LAWPs are sparse and therefore less likely to be grouped together.

2.4.5. Path Optimisation

For post-insertion path optimisation, *1-opt and 2-opt* [26,27] are well-established node relocation and edge exchange moves, respectively, that can be used to shorten a path and correct inconvenient order placement. 1-opt operates on single WPs, while 2-opt operates on edges, i.e., the segments between two consecutive WPs in the path. In the present work, these are used as follows:

- 1-opt: for each unvisited WP in the path, remove it from the path and reinsert it following the shortest detour insertion logic.
- 2-opt: for each pair of unvisited edges $(i, i+1)$, $(j, j+1)$, $j \geq i+2$, examine if the length of the path will be shortened if the inner WPs are exchanged between the pair:
  $$D(WP_i, WP_j) + D(WP_{i+1}, WP_{j+1}) < D(WP_i, WP_{i+1}) + D(WP_j, WP_{j+1})$$
  If so, reverse the visitation order of all WPs between $i+1$ and j.

1-opt and 2-opt can be used separately or in sequence and are applicable to both insertion logics. Since both moves operate on a single WP or edge at a time while disregarding the other parts of the path, they are performed iteratively, until no changes are made. Hence, after each insertion operation of new LAWPs, 1-opt and 2-opt can be performed as follows:

1. For each unvisited WP, perform 1-opt;
2. For each pair of unvisited edges, perform 2-opt;
3. If either step 1 or step 2 resulted in a change in the path, go to step 1. Otherwise, stop.

Both 1-opt and 2-opt are $O(n^2)$ complex: given n unvisited WPs, 1-opt considers the placement of each WP in n candidate positions, while 2-opt considers the exchange of any one of $(n-1)$ edges with any other $(n-3)$. Including a number of iterations needed until the optimisation process stabilises, this can incur a non-negligible computational cost, especially for larger missions.

An experimental comparative analysis of each of the described path planning, adaptation, and optimisation options can be found in Section 3.

2.5. Disease Detection and Classification

2.5.1. Problem Definition

The objective of this final part of the pipeline is to detect signs of disease in grapevine leaves. For this purpose, both classification and detection approaches can be pursued:

Disease classification concerns the classification of a given leaf in one of multiple categories, corresponding to different grapevine diseases as well as healthy leaves. Although several relevant datasets exist, they include images of extreme close-ups of leaves [13], often removed from the plant and placed on a neutral background [14,15], while established classifiers work well on individual leaves [28,29], they are constrained by the necessity of needing individual leaves as input, matching the format of the training data.

Disease detection, without classification, is a much simpler problem that leverages object detection models to detect unhealthy leaves in a larger image [30]. YOLO detectors have been able to detect individual unhealthy leaves in in-the-wild images of whole grapevine plants, each containing hundreds of leaves. In [16], mAP@0.5 metric results range from 7% to 31%, depending on the image resolution. Although in general object detection these values seem low, in the disease detection they are adequate, as not every single unhealthy leaf needs to be detected, and an unhealthy plant will include multiple unhealthy leaves.

2.5.2. Approach and Models

This work combines both detection and classification approaches. It performs detection of unhealthy leaves first and then crops and resizes each detection before feeding it into a classification model.

Disease detection follows the approach of [16], using a YOLOv8 medium detector pretrained on the COCO dataset [31] and fine-tuning it on the VinEye dataset [30] for unhealthy leaf detection.

Aiming for a lightweight architecture, classification uses a unified model based on MobileNetV3-Small [32] using a multi-head sigmoid architecture. Instead of a single Softmax layer, the network outputs independent probabilities for: Healthy leaves, Generic disease (any), and specific disease classes present across the training datasets. This formulation avoids conflicts between datasets that use different taxonomies, enables the model to represent uncertainty, and allows detection of cases not belonging to any known disease class (via low-confidence outputs). The model was trained on a merged dataset combining IDADP [13], NGLD [15], Grape_Disease [33], and VinEye leaf crops.

2.5.3. Performance

The disease classification model was evaluated on 15,309 labelled leaf crops. It achieved a macro-averaged precision of 0.9564, recall of 0.9233, and F1-score of 0.9285 across the 13 healthy and disease classes. Performance was strong overall, although lower scores were observed for some less-represented classes, indicating that classification reliability may vary with class prevalence and visual similarity. These results concern the classification of detected leaf crops and do not directly affect the area-based severity score, which is derived from the preceding object-detection stage.

The disease detector was evaluated on a test set of 804 images of the merged dataset that were not used during training, using an input image size of 640 pixels and several operational confidence thresholds. Table 2 reports the resulting precision, recall, and F1-score, while the mAP values were calculated using the standard validation procedure with a confidence floor of 0.001 and are common to all operating thresholds.

The detector demonstrates moderate performance, which is reasonable given the complexity of the low-altitude, in-the-wild images, which may contain hundreds of overlapping leaves at different scales, together with occlusions, illumination variation, and background

clutter. In the present application, precision is arguably more important than leaf-level recall. False-positive detections may cause a healthy image and its corresponding POI to be prioritised unnecessarily, whereas an image showing substantial disease is likely to contain multiple affected leaves and may therefore still be identified when only a subset is detected. Lower confidence thresholds provide higher recall and the highest overall F1-score, but retain more false-positive detections. Increasing the threshold to 0.40 or 0.50 substantially improves precision, producing more conservative alerts, but also increases the likelihood of missing images containing only a small number of diseased leaves or early-stage symptoms. The confidence threshold can therefore be selected according to the intended operational balance between avoiding false alarms and detecting sparse disease evidence. This selection directly impacts the calculation of a disease severity score.

Table 2. Object-detection performance on the held-out evaluation set at different confidence thresholds.

Image Size	Confidence	Precision	Recall	F1-Score	mAP@0.5	mAP@0.5:0.95
640	0.20	0.5301	0.4568	0.4907	0.4627	0.2175
640	0.25	0.5863	0.4035	0.4780		
640	0.40	0.7257	0.2699	0.3935		
640	0.50	0.8032	0.1850	0.3007		

2.5.4. Severity Score

The result of the above pipeline is a list of detections (crops) and the classification for each. Although these can be presented to the user for perusal, it was also deemed useful to output a disease severity score for the whole of each low-altitude photo (and hence, each POI). In this way, the end-user can have a single scalar metric for each POI, rather than a list of detections, classifications, and confidence scores. This can provide a simpler and more intuitive understanding of high-risk vineyard patches to prioritise in-person inspections or other mitigation actions.

Previous research has often used the ratio of infected vs. healthy area, per leaf, as a measurement of severity [34]. As detection and classification results are at the whole-leaf level, and leaf resolution is insufficient to accurately measure infected area, we generalise from this approach and use the percentage of the image area that is covered by diseased leaves. This percentage can serve as a disease severity score, in line with previous research [35,36].

$$Severity_L = \frac{D}{T} \tag{6}$$

where:

D is the number of pixels covered by diseased leaves bounding boxes;
T is the total number of pixels in the image.

A second relevant metric is the total number of confirmed detections in an image. Although this will generally be correlated with the detected area, it can capture cases where several diseased leaves occupy only a small fraction of the image, for example because of their distance from the camera or partial occlusion.

To combine these two complementary indicators into a single user-facing metric, we additionally define a sigmoid severity score based on both the detected-area ratio and the number of confirmed detections. The sigmoid function maps their combined contribution onto a scale bounded to $(0, 1)$, providing a gradual transition between low- and high-severity cases and allowing for a smooth binary classification, adjustable by its parameters.

The resulting score is intended primarily to support the rapid ranking and prioritisation of POIs. Images with limited detection evidence receive values near the lower end

of the scale, the score becomes more sensitive as the combined evidence increases, and images containing numerous or spatially extensive detections approach the upper end of the scale. The individual detections, classifications, confidence values, and area-based severity measure remain available to the user for more detailed inspection.

$$Severity_S = 0.5\left(\frac{1}{1+e^{-k_S L(Severity_L - Severity_L 0)}} + \frac{1}{1+e^{-k_C(C-C_0)}}\right) \quad (7)$$

where:

$Severity_L$ is the linear severity (Equation (6)).
$Severity_L 0$ is the linear severity midpoint (default = 0.55%).
$k_S L$ is the linear severity slope (default = 8).
C is the count of confirmed detections.
C_0 is the detection count midpoint (default = 2).
k_C is the detection count slope (default = 2.2).

Midpoint and slope default values were assigned based on manual inspection of sample results. However, in future work, these can be further calibrated or be user-definable. Similarly, the weight of the two metrics (area and detection count) need not be equal.

Severity score ranges from 0 to 1, and can be expressed as a percentage of 0–100%. By way of example, an image with signs of disease on a single leaf covering 0.1% of the pixel area will have a severity score of 4.6%, which is very low and would not warrant further investigation. A single leaf could be a false positive of the computer vision models, or possibly a leaf that is damaged by non-disease causes.

In contrast, a POI with six confirmed detections covering 0.55% of the image area would score 75%. It is unlikely that all six detections would be false positives, so further inspection is needed, either by perusing the close-up photo at that POI, or by physically visiting that patch of the vineyard.

2.6. Simulation Tool

To facilitate system testing under controlled, repeatable conditions without the need for a large number of drone flights, a simulation tool was developed. Simulation enables controlled and repeatable comparisons, allowing variables to be kept constant across runs, including random seeds governing the sequence, timing, and location of emerging waypoints. It also makes it feasible to conduct the large number of missions required to examine multiple parameter combinations and average metrics for statistical robustness. Finally, it isolates the effects of path planning and communication parameters from external factors such as weather, lighting, positioning errors, and other sources of variability between physical flights. Two versions of the tool were implemented in the Python (v3.13.2) programming language: one focusing on path optimisation and adaptation, and the other expanding on the first to add real-time communication with the testbed and, through it, remote image processing and POI identification. Simulation tests were conducted between September 2025 and March 2026.

2.6.1. Simulation for Path Planning and Adaptation

The first version of the simulation tool focuses on testing path planning and adaptation. It implements all options discussed in Section 2.4, including options for Boustrophedon and spiral initial patterns, adaptive and non-adaptive insertion logic, and path optimisation. The tool simulates drone flight, path planning, and waypoint insertion and includes a simulated abstraction of POI identification with randomly generated results.

Inputs include vineyard size, flight altitude, drone and camera specifications (camera FOV and average speed), and POI identification simulation parameters (time needed for response, probability of POI appearance, maximum number of POIs per HAWP). A user-selectable random seed drives all random values in the simulation, allowing for repeated runs to compare different parameters with the exact same pseudo-random events.

Outputs include time and distance travelled as well as an interactive step-by-step visualisation of the mission, which shows how new LAWPs generated by POIs are inserted into the path. Figure 5 shows a screenshot of the simulator visualisation.

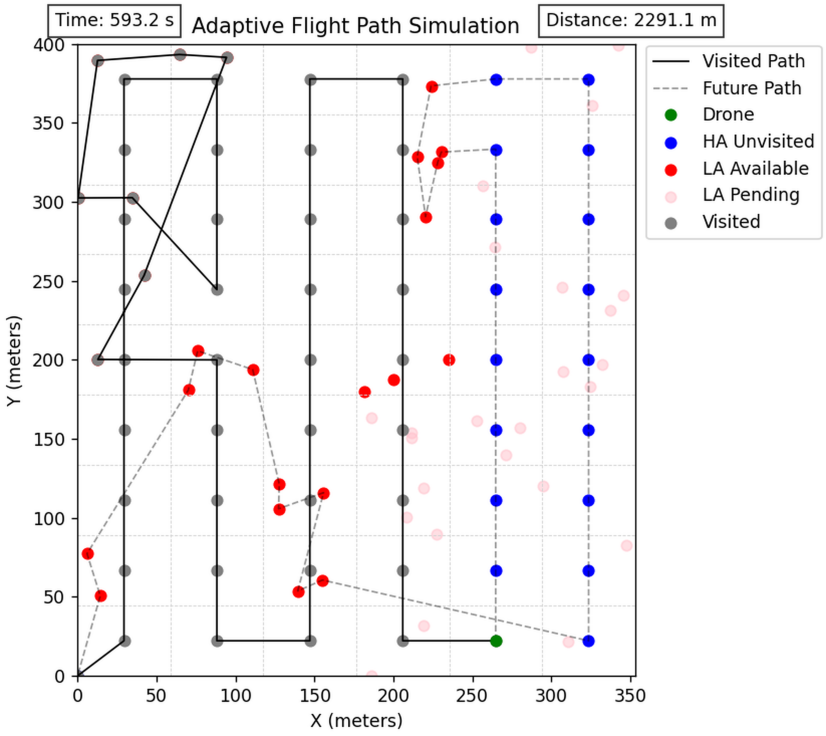


Figure 5. A screenshot of the simulator visualisation, showing a Boustrophedon adaptive path. The green point represents the current drone position. The solid line and grey dots show the past drone path and WPs already visited, respectively. The dashed line shows the future path in its current form. Blue points represent as-yet-unvisited HAWPs. Red points show emerged LAWPs, some of which are already included into the future path, while the newest ones will be included in the next step. Pink dots show LAWPs that have not yet emerged. At the top of the graph, the current time and distance travelled are displayed.

2.6.2. Real-Time Simulation for Testbed-Deployed POI Identification

In order to test the systems integration and communication with a POI identification module deployed on the testbed, the initial simulation tool was adapted and expanded. All path planning and adaptation features were kept, while POI identification was switched from pseudo-random POIs to POIs computed remotely on the testbed. The latter includes connectivity over a VPN, the transmission of real multispectral image sets, and reception of results in real time, and an adjustable connection bandwidth influencing the time lag of transmission and reception.

The simulation tool has access to a folder of real multispectral images of vineyards, with each HAWP linked to a different multispectral image pair of Red and NIR bands. As the virtual drone visits each HAWP, the corresponding image pair commences transmission to the testbed, while the drone continues on its course.

On the testbed side, the POI identification algorithm, as described in Section 2.3.5, runs on reception of each image pair and generates a list of 0–3 POIs as pairs of latitude and longitude. These are sent back to the simulation tool, driving the generation of new LAWPs.

Hence, in this version of the simulation tool, only the flight is simulated, while multispectral input and the communication process are real. The simulation runs in real time, to most faithfully simulate a real drone flight, network conditions, and processing delays. The simulation does not encompass disease detection from low-altitude RGB images, as these do not influence the flight path and have no real-time execution restrictions.

In terms of POIs, result repeatability is guaranteed by using the same multispectral image files as input, as POI identification has no random component and will always yield the same results for the same input. In terms of timing, some fluctuation resulting from the connection does differentiate the results of even identical runs.

2.7. Software and Hardware Configurations

The emulated satellite network runs inside a VM on UBuntu 22.04. The hardware details are listed in Table 3. The routing of the packets is based on IP routing, employing TCP at the transport level. The emulation of the testbed is based on Mininet. The modelled satellite constellation is at orbital height of 1221 km.

Table 3. Testbed specifications.

Parameter	Value
OS	Ubuntu 22.04
CPU	x86-64-v2-AES (16 cores)
RAM	32 GB
Emulation framework	Mininet
Transport protocol	TCP
Orbital Height	1221 km
No. of satellites	4

The pipelines for path adaptation, POI identification, and disease detection, as well as the simulator, were implemented in Python. Table 4 summarises the Python version and major libraries used.

Table 4. Python software environment and principal dependencies.

Function	Software and Version
Python runtime	Python 3.13.2
Numerical and scientific processing	NumPy 2.2.6; SciPy 1.16.3; scikit-learn 1.7.2
Image processing	OpenCV 4.12.0.88; Pillow 12.0.0
Image metadata handling	PyExifTool 0.5.6
Communication and web services	Flask 3.1.2; Flask-SocketIO 5.5.1; Requests 2.32.5; FastAPI 0.120.0; Uvicorn 0.38.0
Visualisation	Matplotlib 3.10.7
Machine learning	PyTorch 2.0.1+cpu; Torchvision 0.15.2+cpu; Ultralytics 8.0.166

Multispectral and RGB aerial images used in testing and validation were captured with a DJI Mavic 3M UAV in July 2025.

3. Results

3.1. Path Adaptation Results

A series of experiments were performed using the simulation tool (see Section 2.6.1) to compare the impact of different options on initial pattern, insertion logic, and path optimisation. Tests on each parameter combination were conducted 20 times, using 20 fixed random seeds, thus ensuring that randomised elements within each test (mainly the emergence and placement of LAWPs) are identical for all parameter combinations.

The following parameters remain constant across all tests:

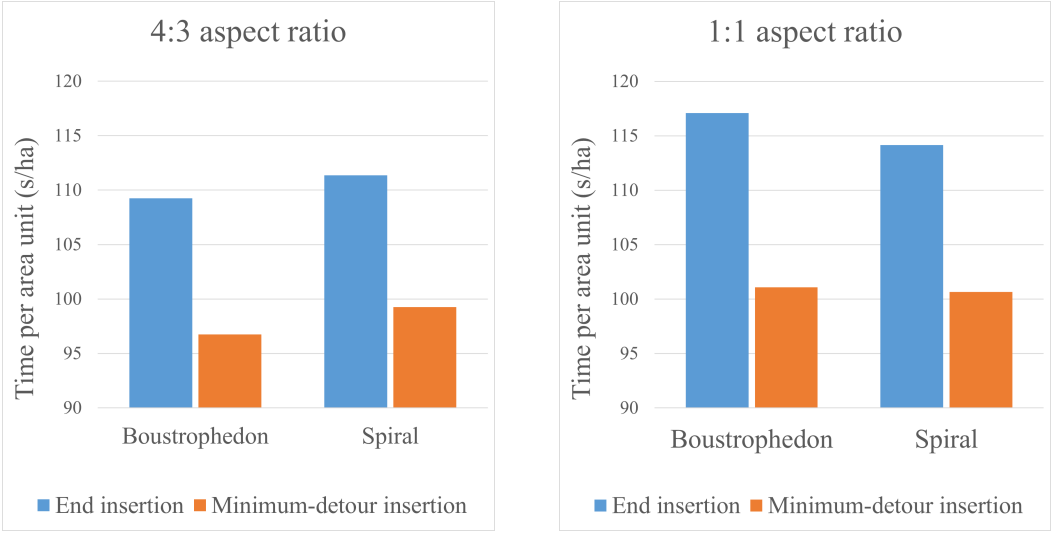
- HAWP altitude: 50 m.
- LAWP altitude: 5 m.
- Drone speed: 5 m/s.
- Additional delay at each WP: 3 s.
- Multispectral camera FOV: 61°× 48° (except where otherwise noted).
- Vineyard shape: rectangular at 530 m × 410 m (except where otherwise noted).

A delay of 3 s was added to each WP to account for deceleration, image acquisition, and acceleration. The camera FOV was modelled after the multispectral camera onboard the DJI Mavic 3M. The vineyard size was selected as the largest that allows mission completion within the time limit imposed by the drone autonomy (45 min).

For each test, the overall distance flown and time needed for mission completion was calculated. Given a constant speed, the two are equivalent. These were then normalised for vineyard size, yielding distance and time per area unit, respectively. In the following presentation, time per area unit, in seconds per hectare, is used for comparison.

3.1.1. Initial Path Pattern and Insertion Logic

The first series of tests compares the two initial patterns, Boustrophedon and Spiral (as described in Section 2.4.3), and the effectiveness of adaptive vs. non-adaptive WP insertion (Section 2.4.4). As illustrated in Figure 6a, adaptive WP insertion provides a more than 10% improvement in time use, while the Spiral pattern is slightly slower than the Boustrophedon.



(**a**) (**b**)

Figure 6. Results for Boustrophedon and Spiral patterns with a non-adaptive and adaptive WP insertion logic: (**a**) Using a standard 4:3 camera aspect ratio, the Boustrophedon pattern is slightly faster. (**b**) With a 1:1 aspect ratio, the Spiral pattern is more advantageous. In both cases, adaptive WP insertion provides a significant reduction in mission time.

As noted in Section 2.4.3, this is likely due to the Spiral path executing more "sideways" moves—moves along the horizontal footprint axis, which correspond to greater distances. To test this hypothesis, the test was repeated with a square camera aspect ratio. A FOV of 54.235° × 54.235° was used, which yields a square footprint of equal area. The results are shown in Figure 6b, in which the Spiral pattern is more advantageous than the Boustrophedon. Although in practice 1:1 aspect ratios are not commonly used, this result highlights the potential of a Spiral pattern to adapt better to include emerging WPs.

3.1.2. Path Optimisation Options

As described in Section 2.4.5, 1-opt and 2-opt were evaluated for path optimisation, both individually and in combination. Optimisation was performed either over the entire remaining path or within rolling windows restricted to a fixed-length look-ahead of upcoming waypoints. For non-adaptive insertion, optimisation options were applied only during the second pass (i.e., after all HAWPs have been visited), in order to keep the initial pattern intact.

1-opt performed best over the entire path during both non-adaptive and adaptive insertion. For 2-opt, however, results differed between the two insertion logics. Figure 7 shows the performance for various look-ahead lengths in each case. For non-adaptive insertion, performance increased with the look-ahead length, while for the adaptive insertion, a smaller window performed better. This is likely because long-horizon 2-opt significantly rearranges the placement of new WPs, defeating the purpose of adaptive insertion, whereas a smaller window acts as a local refinement that remains robust to ongoing changes.

Using both 1-opt and 2-opt in combination, it bears noting that there was a marked increase in performance when 2-opt was performed first, followed by 1-opt. This behaviour is expected, as 2-opt performs larger changes, impacting whole segments of the path, while 1-opt moves single WPs at a time. Hence, in combination, 2-opt will effect larger-scale optimisation while 1-opt will fine-tune with smaller changes.

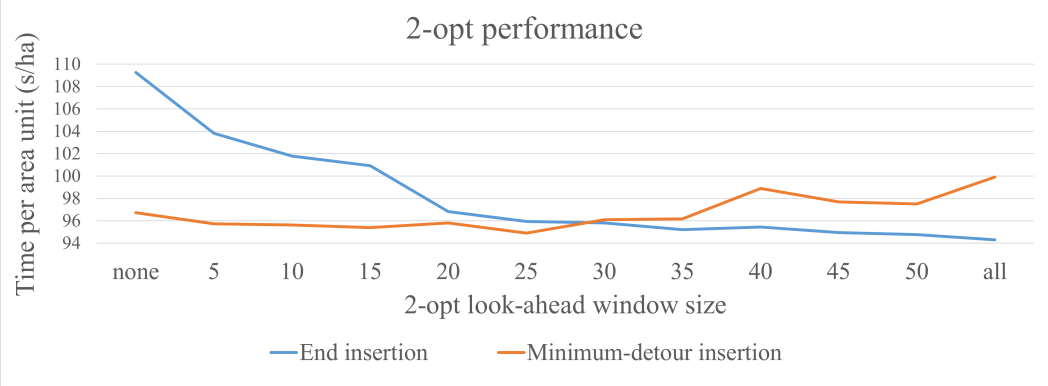


Figure 7. The impact of limiting the window of 2-opt, while non-adaptive insertion benefits most when 2-opt considers the full future path, for adaptive optimisation, it is more advantageous to limit 2-opt to consider only the next few WPs rather than the whole path.

Figure 8 compares the use of 1-opt and 2-opt for both WP insertion logics, using the optimal parameters for each. While the adaptive insertion begins with a significant advantage, optimisation gains are minimal. In contrast, non-adaptive insertion benefits greatly from both 1-opt and 2-opt, as well as their combination, ultimately resulting in a 15% performance gain and surpassing adaptive insertion.

In conclusion, these results highlight the merit of keeping the initial flight pattern intact on the forward pass and adapting the path only on the return pass. This groups all

HAWPs in the forward pass and all LAWPs in the return pass, minimising altitude changes and resulting in an optimal flight path.

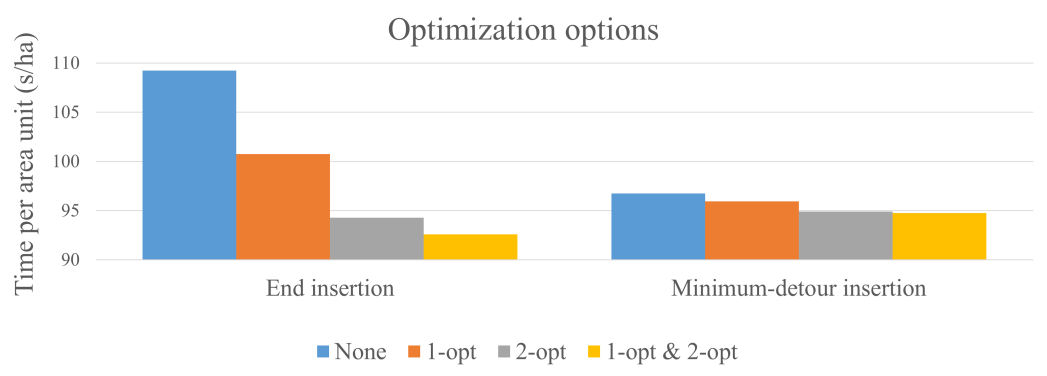


Figure 8. The impact of optimisation options on the path. Both 1-opt and 2-opt cause significant improvements to the non-adaptive insertion logic, while for adaptive logic benefits are more modest. With full optimisation, non-adaptive WP insertion performs better.

3.2. Real-Time Simulation Results

Once the optimal waypoint insertion and path adaptation options were identified, real-time, real-input tests were performed using the setup described in Section 2.6.2. The best-performing option set (end insertion followed by 1-opt and 2-opt path optimisation) was contrasted with the base, non-adaptive option (simple end insertion).

In these tests, the independent variable is the bandwidth allocated to the communication between the (simulated) drone and the server hosting the POI identification module, while the measured dependent variables are response latency and mission time. To minimise the effects of connectivity fluctuations, each test was performed five times and the results averaged.

Figure 9 shows the impact of bandwidth on the latency of the POI identification response, i.e., the time between the capturing of a multispectral image set and delivery of any POIs identified in that set. Please note that the vertical axis is in logarithmic scale. As also stated earlier, the significant part of the latency corresponds to the transmission of the multispectral images, which is affected by the bandwidth; their processing and POI identification takes roughly 1.5 s, and is of course not affected by the bandwidth.

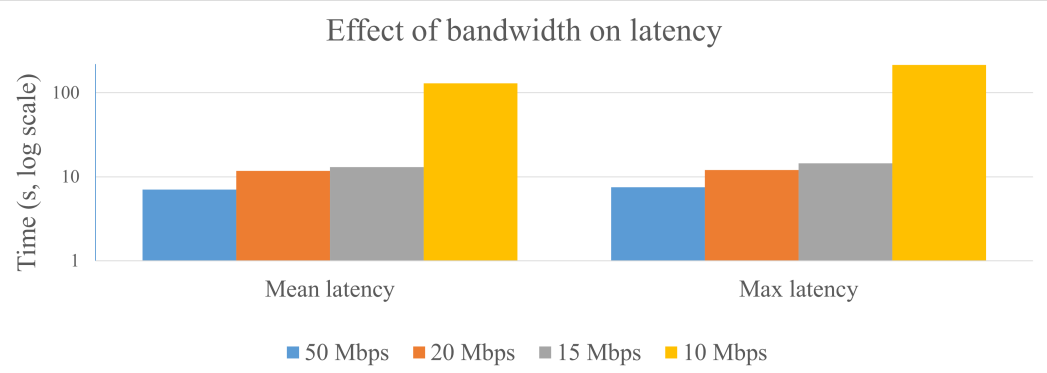


Figure 9. Effect of bandwidth limitations on POI identification latency. At bandwidths below the real-time transmission threshold, the transmission queue builds up resulting in significant latency increases. Vertical axis in log scale.

For bandwidth values above 15 Mbps, latency is, predictably, inversely proportional to the bandwidth. However, at 10 Mbps there is a sharp increase. This corresponds to a

transmission time greater than the average time between HAWP visits, resulting in the build-up of a queue.

This can be corroborated theoretically: using the drone speed, altitude, and camera FOV detailed in Section 2.3, the drone will visit consecutive HAWPs every 11.8 s. Each multispectral band image has a size of 9.62 MB (uncompressed TIFF), and two are needed: the Red and the NIR. Hence, a bandwidth of $9.62 \times 8 \times 2/11.8 = 13$ Mbps is needed for real-time transmission.

Figure 10 shows how this affects mission time. At bandwidths adequate for real-time transmission, it does not. At 10 Mbps, it has a very significant effect, but only in the case of end insertion with no further optimisation. In that insertion logic, all HAWPs are visited first, and therefore all image transmission tasks are generated in the first half of the mission (the forward pass). In contrast, 1-opt and 2-opt optimisation reposition some POIs between HAWPs, effectively allowing the queue to drain and spreading HAWP visits and image transmission tasks more evenly across the whole mission.

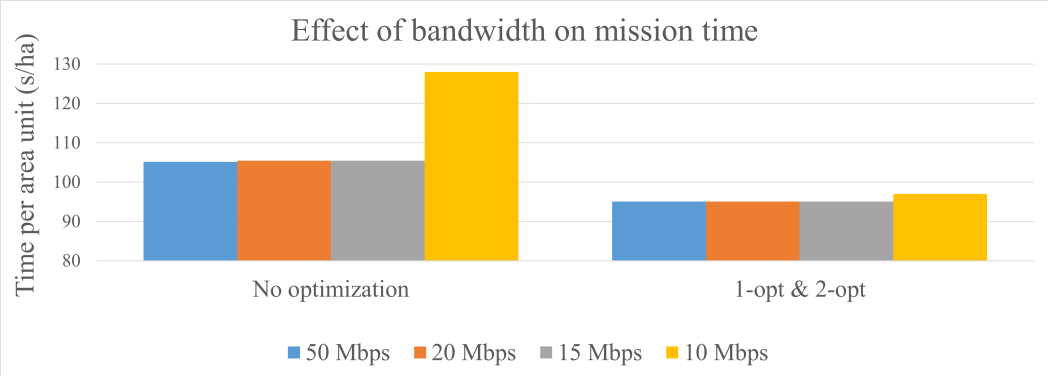


Figure 10. The effect of bandwidth on mission time is negligible above the real-time transmission threshold. Below the threshold, waypoint reordering through 1-opt and 2-opt results in better utilisation of the available bandwidth, spreading the transmission tasks throughout the whole mission.

In conclusion, these real-time tests show that allowing on-the-fly waypoint reordering not only reduces the path length but also makes better use of the available bandwidth.

3.3. Real-World Path Optimisation Testing

As an initial real-world feasibility test, optimised and non-optimised flight paths were executed using a DJI Mavic 3M UAV. Both paths contained the same set of high- and low-altitude waypoints and differed only in their visitation order. For each path, the final waypoint sequence generated by the simulator was translated into local coordinates, packaged as a KMZ waypoint mission, and uploaded to the UAV. The elapsed time from take-off to landing was then recorded.

The test missions covered a rectangular area of 250 m $\times$ 240 m, selected to maintain visual line of sight and remain within the permitted flight area, with a target flight speed of 5 m/s. The optimised mission was completed in 515 s, compared with 552 s for the non-optimised mission, corresponding to a reduction of 6.7%. Both missions were slightly faster than the corresponding simulations because the UAV did not pause at each waypoint. The observed gain was lower than the maximum simulated reduction of 15%, which is expected given the smaller test area, the lower number of emerging waypoints, and the consequently fewer opportunities for route optimisation. Although this simplified experiment did not evaluate live in-flight waypoint insertion, it provides an initial indication that the optimised route is executable by a real UAV and that its relative advantage is retained under physical flight conditions.

4. Discussion

4.1. Summary and Interpretation of Findings

A core takeaway of the described work is the feasibility of two-tier drone scanning, combining higher-altitude multispectral imaging with targeted close-up inspection at selected points. This approach combines the benefits of each altitude and modality. The higher-altitude tier can scan the whole vineyard efficiently and identify candidate sampling points for the low-altitude tier, while the low-altitude tier benefits from increased resolution without incurring the long mission times associated with exhaustive close-up scanning.

With new low-altitude waypoints emerging mid-flight, path adaptation can shorten the overall mission time significantly. This is especially relevant in larger vineyards requiring multiple missions to cover, where a 15% reduction in mission time can translate to substantial savings in effort and operational cost, while frequent altitude changes can lengthen the flight path, strict grouping of waypoints by altitude does not necessarily yield the shortest mission. In the experiments presented in this work, initially inserting new waypoints at the end of the path and then iteratively optimising the remaining route was found to be the most beneficial strategy. This approach uses altitude grouping as a starting point, but retains enough flexibility to reorganise the path when beneficial.

Although direct numerical comparison is difficult because the existing studies address different settings, they provide a useful performance benchmark. Recent studies [37–39] report reductions of up to 10–16.3% in route length or mission time, which is in line with the optimisation gains reported in the current work. However, the current problem formulation, including a grid-following high-altitude tier and low-altitude waypoints emerging in the vicinity of the former some time after they are visited, is unique to this study. Hence, the above should be interpreted as path optimisation context rather than direct comparison.

With local drone-side processing power often limited, offloading processing tasks to a remote server is useful, but the system must account for available bandwidth. The transmission of high-resolution drone images can require significant bandwidth, especially in multispectral missions where multiple band images, often uncompressed, must be transmitted. On-site and off-site processing must therefore strike a balance between available bandwidth and local processing power, while prioritising time-critical computations such as point-of-interest identification.

Communication bottlenecks become relevant when bandwidth drops below the real-time transmission threshold, allowing queues to build up. At low bandwidths, adapted flight paths that mix high-altitude and low-altitude waypoints can help reduce this effect, as low-altitude waypoints do not trigger real-time multispectral image transmissions and therefore allow previously accrued queues to decrease during the mission.

Practical deployment of the proposed workflow would require some operational considerations. The UAV platform must support in-flight waypoint updates while maintaining core functionalities such as terrain clearance, obstacle avoidance, and return-to-home, which could require a custom UAV controller application. Moreover, safeguards should be implemented to allow the incorporation of additional waypoints only when the new path remains safely feasible with the available battery. Communication must be robust to connectivity status, allowing local storage and delayed transmission when necessary. Finally, and including the above, an appropriate user interface must be developed to allow the practical definition of inspection area, flight parameters, and user interventions.

Overall, in this type of two-tier workflow with emergent waypoints, mission planning and communication cannot be approached separately. Routing choices affect not only flight distance but also when data are generated, when processing results become available, and how communication queues evolve.

4.2. Remaining Gaps and Future Work

In this work, the evaluation was designed to isolate the effects of path adaptation and communication bandwidth under controlled and repeatable conditions. With the exception of the feasibility test described in Section 3.3, drone motion was simulated, while the POI-identification pipeline operated on real multispectral vineyard images and exchanged data with the satellite emulation testbed in real time. The experiments therefore serve as proof of concept for adaptive flight path and two-tier inspection, but do not constitute validation in realistic conditions. Future work must test and validate the presented workflow using real drone flights and operational communication links, evaluating navigation accuracy, mission duration, and the benefits of in-flight path adaptation under field conditions.

The experiments also adopted deliberately simplified baseline conditions, including a rectangular vineyard shape and flat terrain. These assumptions primarily simplify the generation of the initial high-altitude coverage path rather than the subsequent waypoint-insertion and route-optimisation methods. Boustrophedon coverage can be applied to non-rectangular areas by clipping the sweep lines to the vineyard boundary or decomposing more complex regions into simpler cells. The resulting waypoint set can then be processed using the same adaptation and optimisation procedures. Nevertheless, irregular boundaries may introduce unequal row lengths, additional turns and transfer segments, and disconnected subregions, potentially affecting both absolute mission duration and the relative benefit of optimisation.

Terrain elevation, although not considered in these experiments, is fully supported by the proposed approach, as waypoint distances, and hence optimisation, are calculated using three-dimensional distances. For gently varying terrain, the additional vertical distance is expected to be small compared with the altitude difference between the two inspection tiers. However, the present simulation did not model terrain-following waypoint generation, climb- and descent-specific flight characteristics, energy consumption, or changes in ground clearance and image footprint. These effects may become significant in steep terrain, particularly during the low-altitude inspection tier, and should therefore be examined in future work using terrain models and field experiments.

This study examines single satellite architecture designed to satisfy the communication requirements of the evaluated scenario. It is worthy of note that the performance of the system is primarily influenced by link parameters such as packet loss rate and stability of the link. This impact is more prominent on ground–satellite links than on ISLs. This is because ISLs are fairly constant and do not vary much other than during the handover which happens between the satellites in inter-satellite orbits. The latency between the ground link and the satellite scales linearly with the height of the orbital plane. Given the speed of light, each 300 km increase in altitude can introduce approximately 1 ms of one-way delay. Hence, loss rate and stability dominate the link performance. A higher packet loss rate necessitates frequent re-transmissions which adversely effects the TCP performance due to its congestion control mechanisms. Although the end-to-end latency increases in this case, TCP's reliability mechanisms help to maintain connectivity. In addition to this, severe weather conditions can negatively impact the performance of the links; but these conditions restrict drone operations eliminating communication requirements during these time frames.

Another important direction for future research is the agronomic validation of multispectral-derived points of interest as smart sampling points. While this work identifies POIs as local or global minima in NDVI, further research should evaluate how well these points correspond to actual plant stress, disease presence, or other agronomically relevant conditions. This may also require calibrating the POI selection logic for different vineyard varieties, growth stages, environmental conditions, and management objectives.

The disease-detection and severity-estimation components were included as downstream modules in the overall workflow, but were not benchmarked as separate contributions in this paper. Future work can further evaluate these components using vineyard-specific ground truth, compare alternative detection and classification models, and calibrate severity scores against expert assessment or field observations.

Lastly, although the present work focused on vineyard monitoring, significant parts of the proposed architecture may be applicable to other crops. The general idea of using broad-area sensing to guide selective close-up inspection can transfer to other agricultural domains, but the sensing modalities, vegetation indices, point-of-interest logic, and inspection strategy would need to be adapted and validated for each specific use case.

Author Contributions: Conceptualisation, K.K., K.C., and T.d.C.; methodology, K.K., K.C., R.S., and G.G.; software, K.K., K.C., R.S., and G.G.; validation, K.K., K.C., R.S., and G.G.; investigation, K.K., K.C., T.d.C., R.S., and G.G.; data curation, K.K.; writing—original draft preparation, K.K., K.C., T.d.C., R.S., and G.G.; writing—review and editing, K.K., K.C., T.d.C., R.S., and G.G.; visualisation, K.C.; supervision, K.K. and T.d.C.; project administration, K.K. and T.d.C.; funding acquisition, T.d.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research is funded by the Horizon Europe project AGRARIAN (Grant Agreement No. 101134128), supporting the sub-project VINEA-Drone via Financial Support to Third Parties (FSTP). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author to conserve the privacy of vineyard owners contributing relevant imagery.

Conflicts of Interest: The authors declare that this study received funding from the European Union's Horizon Europe programme. The funder was not involved in the study design, data collection, data analysis, data interpretation, the writing of this article, or the decision to submit it for publication. Konstantinos Konstantoudakis and Kyriaki Christaki were employed by the company KyberKore P.C. However, the research and work described in this paper are not connected to any commercial solution or financial benefit. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

1-opt	One-node relocation local-search move
2-opt	Two-edge exchange local-search move
COCO	Common Objects in Context
DLR	German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt)
FOV	Field of View
GSD	Ground Sampling Distance
HAWP	High-Altitude Waypoint
IP	Internet Protocol
ISL	Inter-Satellite Link
LAWP	Low-Altitude Waypoint
LEO	Low Earth Orbit
mAP	mean Average Precision
MB	Megabyte
Mbps	Megabits per second
NDVI	Normalized Difference Vegetation Index
NIR	Near Infrared

ORB	Oriented FAST and Rotated BRIEF
POI	Point of Interest
RGB	Red–Green–Blue
SDK	Software Development Kit
TCP	Transmission Control Protocol
TIFF	Tagged Image File Format
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
VM	Virtual Machine
VPN	Virtual Private Network
WP	Waypoint
YOLO	You Only Look Once

References

1. Mucalo, A.; Matić, D.; Morić-Špranić, A.; Čagalj, M. Satellite solutions for precision viticulture: Enhancing sustainability and efficiency in vineyard management. *Agronomy* **2024**, *14*, 1862. [CrossRef]
2. Ramírez-Arroyo, A.; López, M.; Rodríguez, I.; Damsgaard, S.B.; Mogensen, P. Multi-connectivity solutions for rural areas: Integrating terrestrial 5G and satellite networks to support innovative IoT use cases. *Smart Agric. Technol.* **2025**, *12*, 101260. [CrossRef]
3. Guebsi, R.; Mami, S.; Chokmani, K. Drones in precision agriculture: A comprehensive review of applications, technologies, and challenges. *Drones* **2024**, *8*, 686. [CrossRef]
4. Radočaj, D.; Šiljeg, A.; Marinović, R.; Jurišić, M. State of major vegetation indices in precision agriculture studies indexed in web of science: A review. *Agriculture* **2023**, *13*, 707. [CrossRef]
5. Berry, A.; Vivier, M.; Poblete-Echeverría, C. Evaluation of canopy fraction-based vegetation indices, derived from multispectral UAV imagery, to map water status variability in a commercial vineyard. *Irrig. Sci.* **2025**, *43*, 135–153. [CrossRef]
6. Portela, F.; Sousa, J.J.; Araújo-Paredes, C.; Peres, E.; Morais, R.; Pádua, L. Monitoring the Progression of Downy Mildew on Vineyards Using Multi-Temporal Unmanned Aerial Vehicle Multispectral Data. *Agronomy* **2025**, *15*, 934. [CrossRef]
7. Campos, J.; García-Ruíz, F.; Gil, E. Assessment of vineyard canopy characteristics from vigour maps obtained using UAV and satellite imagery. *Sensors* **2021**, *21*, 2363. [CrossRef] [PubMed]
8. Abbas, A.; Zhang, Z.; Zheng, H.; Alami, M.M.; Alrefaei, A.F.; Abbas, Q.; Naqvi, S.A.H.; Rao, M.J.; Mosa, W.F.; Abbas, Q.; et al. Drones in plant disease assessment, efficient monitoring, and detection: A way forward to smart agriculture. *Agronomy* **2023**, *13*, 1524. [CrossRef]
9. Chin, R.; Catal, C.; Kassahun, A. Plant disease detection using drones in precision agriculture. *Precis. Agric.* **2023**, *24*, 1663–1682. [CrossRef]
10. Portela, F.; Sousa, J.J.; Araújo-Paredes, C.; Peres, E.; Morais, R.; Pádua, L. A systematic review on the advancements in remote sensing and proximity tools for grapevine disease detection. *Sensors* **2024**, *24*, 8172. [CrossRef] [PubMed]
11. Mousavi, S.; Farahani, G. A novel enhanced VGG16 model to tackle grapevine leaves diseases with automatic method. *IEEE Access* **2022**, *10*, 111564–111578. [CrossRef]
12. Mininet. 2021. Available online: https://mininet.org/ (accessed on 1 July 2026).
13. Yuan, Y.; Chen, L. An image dataset for IDADP-grape disease identification. *China Sci. Data* **2022**, *7*, 86–90. [CrossRef]
14. Geetharamani, G.; Pandian, A. Identification of plant leaf diseases using a nine-layer deep convolutional neural network. *Comput. Electr. Eng.* **2019**, *76*, 323–338. [CrossRef]
15. Dharrao, M.; Zade, N.; Kamatchi, R.; Sonawane, R.; Henry, R.; Dharrao, D. Grapes leaf disease dataset for precision agriculture. *Data Brief* **2025**, *61*, 111716. [CrossRef] [PubMed]
16. Székely, D.E.; Dobra, D.; Dobre, A.E.; Domşa, V.; Drăghici, B.G.; Ileni, T.A.; Konievic, R.; Molnár, S.; Sucala, P.; Zah, E.; et al. Bacterial-fungicidal vine disease detection with proximal aerial images. *Heliyon* **2024**, *10*, e34017. [CrossRef] [PubMed]
17. Pañitrur-De la Fuente, C.; Valdés-Gómez, H.; Roudet, J.; Verdugo-Vásquez, N.; Mirabal, Y.; Laurie, V.F.; Goutouly, J.P.; Opazo, C.A.; Fermaud, M. Vigor thresholded NDVI is a key early risk indicator of Botrytis bunch rot in vineyards. *Oeno ONE* **2020**, *54*, 279–297. [CrossRef]
18. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In *Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011*; IEEE: Piscataway, NJ, USA, 2011; pp. 2564–2571.

19. Vera-Esmeraldas, A.; Galleguillos, M.; Labbé, M.; Cáceres-Mella, A.; Rojo, F.; Salazar, F. UAV NDVI-Based Vigor Zoning Predicts PR-Protein Accumulation and Protein Instability in Chardonnay and Sauvignon Blanc Wines. *Plants* **2026**, *15*, 243. [CrossRef] [PubMed]
20. Bal, A.; Palus, H. Image vignetting correction using a deformable radial polynomial model. *Sensors* **2023**, *23*, 1157. [CrossRef] [PubMed]
21. Ahn, H.; Kim, C.; Lim, S.; Jin, C.; Kim, J.; Choi, C. Minimizing seam lines in UAV multispectral image mosaics utilizing irradiance, vignette, and BRDF. *Remote Sens.* **2025**, *17*, 151. [CrossRef]
22. Coombes, M.; Fletcher, T.; Chen, W.H.; Liu, C. Optimal polygon decomposition for UAV survey coverage path planning in wind. *Sensors* **2018**, *18*, 2132. [CrossRef] [PubMed]
23. Rios, B.H.O.; Xavier, E.C.; Miyazawa, F.K.; Amorim, P.; Curcio, E.; Santos, M.J. Recent dynamic vehicle routing problems: A survey. *Comput. Ind. Eng.* **2021**, *160*, 107604. [CrossRef]
24. Zhang, J.; Van Woensel, T. Dynamic vehicle routing with random requests: A literature review. *Int. J. Prod. Econ.* **2023**, *256*, 108751. [CrossRef]
25. Nazari, M.; Oroojlooy, A.; Snyder, L.; Takác, M. Reinforcement learning for solving the vehicle routing problem. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; Volume 31.
26. Ma, Y.; Cao, Z.; Chee, Y.M. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 49555–49578. [CrossRef]
27. Yang, T.; Wang, W.; Wu, Q. Fuzzy demand vehicle routing problem with soft time windows. *Sustainability* **2022**, *14*, 5658. [CrossRef]
28. Karim, M.J.; Goni, M.O.F.; Nahiduzzaman, M.; Ahsan, M.; Haider, J.; Kowalski, M. Enhancing agriculture through real-time grape leaf disease classification via an edge device with a lightweight CNN architecture and Grad-CAM. *Sci. Rep.* **2024**, *14*, 16022. [CrossRef] [PubMed]
29. Kunduracioglu, I.; Pacal, I. Advancements in deep learning for accurate classification of grape leaves and diagnosis of grape diseases. *J. Plant Dis. Prot.* **2024**, *131*, 1061–1080. [CrossRef]
30. Molnár, S.; Tamas, L. Close proximity aerial image for precision viticulture. A review. *J. Plant Dis. Prot.* **2025**, *132*, 57. [CrossRef]
31. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 740–755.
32. Howard, A.G.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for MobileNetV3. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019.
33. Mandal, R.; Gouda, M.K. Vineyard vigilance: Harnessing deep learning for grapevine disease detection. *J. Emerg. Investig.* **2024**, *7*, 1–9.
34. Patil, S.B.; Bodhe, S.K. Leaf disease severity measurement using image processing. *Int. J. Eng. Technol.* **2011**, *3*, 297–301.
35. Bedi, P.; Gole, P.; Marwaha, S. PDSE-Lite: Lightweight framework for plant disease severity estimation based on Convolutional Autoencoder and Few-Shot Learning. *Front. Plant Sci.* **2024**, *14*, 1319894. [CrossRef] [PubMed]
36. Zhao, C.; Li, C.; Wang, X.; Wu, X.; Du, Y.; Chai, H.; Cai, T.; Xiang, H.; Jiao, Y. Plant disease segmentation networks for fast automatic severity estimation under natural field scenarios. *Agriculture* **2025**, *15*, 583. [CrossRef]
37. Majd, A.; Loni, M.; Sahebi, G.; Daneshtalab, M. Improving motion safety and efficiency of intelligent autonomous swarm of drones. *Drones* **2020**, *4*, 48. [CrossRef]
38. Airlangga, G.; Sukwadi, R.; Basuki, W.W.; Sugianto, L.F.; Nugroho, O.I.A.; Kristian, Y.; Rahmananta, R. Adaptive path planning for multi-UAV systems in dynamic 3D environments: A multi-objective framework. *Designs* **2024**, *8*, 136. [CrossRef]
39. Yang, Q.; Zhang, Y.; Shao, S. Dynamic siting and coordinated routing for UAV inspection via hierarchical reinforcement learning. *Machines* **2025**, *13*, 861. [CrossRef]