Expert Systems With Applications 324 (2026) 132506

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A recommendation system for requirements tuning of BVLoS drones

Soumik Das [a,b,*], Punyasha Chatterjee [b], Agostino Cortesi [a]

[a] DAIS, Università Ca' Foscari Venezia, Venice, Italy
[b] School of Mobile Computing and Communication, Jadavpur University, Kolkata, India

ARTICLE INFO

Keywords:
Drones
UAVs
BVLoS
Requirement interdependencies
Parameter taxonomy
Dependency graph
Recommendation system

ABSTRACT

The purposes for which Unmanned Aerial Vehicles (UAVs) are used today are innumerable and their role is continuously expanding. In particular, BVLoS (Beyond Visual Line of Sight) drones, that enable large-scale and long-distance missions, pose complex challenges such as managing dynamic obstacles, connectivity issues and energy constraints. Drone performance heavily depends on interdependent parameters such as altitude, speed and battery usage where tuning one affects others, but such interdependencies have not been adequately addressed in the existing drone software. This paper introduces a generic approach for BVLoS mission planning through an interactive recommendation system that considers both parameter dependencies and operational requirements. The framework includes parameter taxonomy creation, requirement classification as functional and non-functional ones, dependency graph modeling considering both static and dynamic parameters, and an optimization algorithm to generate tuning recommendations. Extensive simulations and implementation results demonstrate improved mission efficiency, lower computational complexity, and higher reliability, providing a structured, scalable solution for BVLoS operations under diverse environmental and mission-specific scenarios.

1. Introduction

Unmanned Aerial Vehicles (UAVs), commonly known as drones, are aircraft systems that operate without an onboard pilot, that are controlled either remotely by an operator or autonomously through pre-programmed instructions and onboard systems (Sorbelli et al., 2024c). UAVs are popularly used for aerial photography and videography. In addition to that, these are used in various fields such as agriculture, logistics, disaster management, etc (Idries et al., 2015). In the agricultural sector, crop monitoring, pesticide spraying, and soil analysis are being carried out with their help (Liu et al., 2023). In logistics, companies use UAVs for quick package deliveries, especially in remote areas (Hu et al., 2025; Kuru et al., 2019). They are also vital in disaster management, providing real-time surveillance and aiding rescue missions (Laghari et al., 2023; Zhang et al., 2023)[1].

Based on the operator's ability to maintain visual contact, drone operations are divided into three categories, namely Visual Line of Sight (VLoS), Extended Visual Line of Sight (EVLoS), and Beyond Visual Line of Sight (BVLoS) (Matalonga et al., 2022; Sorbelli et al., 2024c). In VLoS, the operator can see the drone at all times i.e. the drone remains in the visual range of the operator. These drones are used for short-range tasks such as aerial photography, infrastructure inspection, small-scale agriculture monitoring, etc. In EVLoS, the drone is flown beyond the operator's direct line of sight but with the aid of a visual observer who keeps the drone in sight (Sorbelli et al., 2024c). The observer communicates with the operator, extending the operational range while maintaining visual awareness of the drone. EVLoS enables applications such as larger infrastructure inspections or broader search and rescue operations. But the most promising application is BVLoS (Matalonga et al., 2022), where drones can perform tasks beyond the operator's direct line of sight. This makes it perfect for applications such as extensive surveying, environmental monitoring, industrial inspections and long-distance drone delivery services. The ranges of VLoS, EVLoS and BVLoS are shown in Fig. 1.

To enable a wide range of autonomous functions, from flight control to real-time data collection, processing, and analysis, drone software plays a crucial role, for navigation (Patrik et al., 2019), delivery (Hu et al., 2025), obstacle detection and collision avoidance (Wang & Qian, 2026), GPS-based positioning (Kwak & Sung, 2018), automated take-off and landing, return-to-home functionality etc. Compliance with aviation regulations and adaptability to various environmental conditions further enhance its effectiveness, making drone software indispensable for reliable UAV operations. Among the existing software systems we can

* Corresponding author.
E-mail addresses: soumikd.smcc.pg@jadavpuruniversity.in (S. Das), punyasha.chatterjee@jadavpuruniversity.in (P. Chatterjee), cortesi@unive.it (A. Cortesi).

[1] The authors obviously deplore the use of drones for military purposes.

https://doi.org/10.1016/j.eswa.2026.132506
Received 24 October 2025; Received in revised form 16 March 2026; Accepted 15 April 2026
Available online 22 April 2026

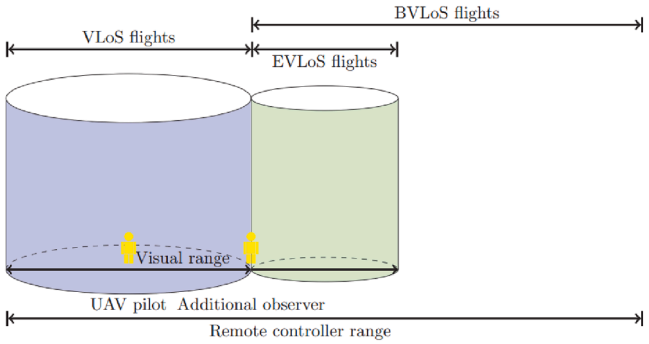


Fig. 1. Ranges of VLoS, EVLoS and BVLoS (Sorbelli et al., 2024c).

mention PX4[2], ArduPilot[3], DroneKit[4], DroneUp[5], Skydio Autonomy[6], DJI Terra[7], PIX4D[8]. However, most of these software systems do not support drones' BVLoS missions. This is because BVLoS drone operations are constrained by strict regulations and need fully autonomous functionalities to address some demanding challenges while flying, including dynamic obstacle avoidance (Park et al., 2008), risk assessment (Sorbelli et al., 2023, 2024a), maintenance of stable connections over long distances (Sorbelli et al., 2023, 2024a), and optimal use of battery energy.

Recent research has also focused on improving UAV architectures and control mechanisms to enhance operational robustness and adaptability. For example, reconfigurable drone platforms have been investigated to allow UAV systems to adapt to different mission requirements and operational conditions. In Derrouaoui et al. (2022) authors provide a comprehensive review of reconfigurable drones, discussing their classification, structural characteristics, design principles, and control technologies that enable flexible UAV configurations. Similarly, in Salmi et al. (2024) a fault-tolerant control strategy is proposed for reconfigurable quadrotors that allows stable flight even in the presence of actuator failure. These studies highlight the increasing complexity and sophistication of modern UAV systems and emphasize the importance of intelligent tools that assist operators in configuring mission parameters effectively, particularly in challenging operational scenarios such as BVLoS missions.

It is important to notice that different functionalities supported by the software are dependent on multiple operational parameters. For example, navigation depends on altitude, speed, flight mode, PID (Proportional-Integral-Derivative) controller, rate controller, etc. If we increase the speed of a drone to reduce latency, faster adjustment of altitude is necessary, as the drone might face aerodynamic forces or external factors (like sudden obstacles). But if the rate-controller or the PID controller for altitude adjustment can't react quickly, the drone may find it difficult to maintain stable flight and run the risk of overshooting or undershooting in height. Another example could be battery management in drones. If we increase the speed or payload capacity to achieve better performance for the drone's delivery mission, this could lead to faster battery depletion, which affects the drone's range and endurance, impacting its overall performance. Requirements and parameters corresponding to different requirements have interdependencies: adjusting one parameter to achieve better functionality may affect other parameters, deteriorating another functionality (Das et al., 2024). As far as we know, there is no tool that solves this important problem in the specific context of BVLoS drones.

In this paper, we introduce a generic methodological approach that analyses the interdependencies among requirements and the parameters to build a recommendation system for BVLoS drone missions. The novel contribution of the paper can be summarized as follows:

- We create a general taxonomy covering parameters and constraints associated to the different functionalities related to drone's BVLoS mission.
- We introduce an algorithm that given a Software Requirement Specification (SRS) provided by the user generates the dependency graph representing the relationships between different parameters.
- On top of the dependency graph and taxonomy, we develop a recommendation system which identifies the parameters to be tuned to achieve an optimal outcome for a specific mission while respecting the system constraints.

The rest of the paper is organized as follows: Section 2 discusses the related work, Section 3 introduces the proposed methodology, Section 4 describes the experimental evaluation of the system and Section 5 concludes.

2. Related work

In this section, we perform a comparative analysis of existing software, considering the basic requirements and features relevant for BVLoS drone missions such as navigation, vehicle state/position tracking, mission planning, geospatial and regulatory, safety, battery management. Navigation enables precise handling and stabilization of the drone and ensures smooth operations (Kuroki et al., 2010; Patrik et al., 2019). It also encompasses altitude, velocity, flight mode, autonomy, terrain mapping, and surface tracking. Vehicle State/Position Tracking provides real-time updates on the drone's location and orientation, along with position control and real-time mapping using GPS (Kwak & Sung, 2018). These are crucial for tasks like surveying or inspections. Mission planning allows users to design, schedule, and execute drone operations with predefined routes and objectives using various path planning algorithms that ensure efficient and safe completion of the mission (Leary et al., 2011). It is required for product delivery within the available flight time. Geospatial and regulatory compliance ensure local airspace constraints such as no-fly zones, obstacles, etc (Leary et al., 2011). Safety features protect the drone and environment by incorporating fail-safe mechanisms (for preventing accidents) that relies on obstacle detection to navigate safely and also address actions for lost radio control, low battery, and geofence breaches (Alamouri et al., 2023). Battery Management optimizes energy usage and provides alerts for low power to avoid mission interruptions, including battery level/status, battery capacity monitoring, and power consumption rate (Jiao et al., 2023).

Recent research has also focused on improving different aspects of UAV mission planning, system autonomy and operational evaluation through advanced simulation frameworks and intelligent coordination strategies. For instance, in Cofield et al. (2026) authors proposed a scalable end-to-end multi agent simulation environment designed to support the development and evaluation of cooperative UAV systems. The framework integrates realistic communication constraints and flexible mission configurations, enabling researchers to study distributed coordination strategies and communication aware UAV operations within large scale simulation environments. Such platforms provide valuable tools for analyzing collaborative UAV behaviors and testing mission strategies before real world deployment. Similarly, Chen et al. (2026) proposes an advanced navigation framework that leverages vision language models to enable UAVs to interpret high level human commands and autonomously navigate through complex environments. This approach integrates perception modules with reinforcement learning techniques, allowing the drone to process visual inputs and language based

[2] PX4- https://docs.px4.io/main/en/
[3] ArduPilot-https://ardupilot.org/
[4] DroneKit-https://dronekit.io/
[5] DroneUp-https://www.droneup.com/
[6] Skydio Autonomy-https://www.skydio.com/skydio-autonomy
[7] DJI Terra-https://enterprise.dji.com/dji-terra
[8] PIX4D-https://www.pix4d.com/

instructions simultaneously. This method improves navigation reliability by enabling drones to adapt their trajectory planning based on environmental context and mission objectives. The proposed framework demonstrates how recent developments in artificial intelligence can significantly enhance the level of autonomy in UAV systems, particularly in scenarios where traditional waypoint based navigation may be insufficient. Another recent contribution (Yang et al., 2025) focuses on control and optimization strategies for UAV formations operating in uncertain environments. The authors introduce an adaptive predefined time path following control method designed to ensure accurate trajectory tracking and robust coordination among multiple UAVs. This approach incorporates adaptive control techniques that compensate for system uncertainties and external disturbances while maintaining synchronization among UAV agents. The study highlights the importance of advanced control strategies for maintaining stability, performance and safety in multi UAV missions particularly in applications such as surveillance and cooperative sensing.

Collectively, these studies demonstrate the increasing complexity of modern UAV operations where autonomy, navigation reliability and coordinated control play critical roles in mission success. However, while these approaches significantly advance specific aspects of UAV systems such as autonomous navigation, intelligent architectures and coordinated control they primarily focus on algorithmic improvements or system-level control strategies. They do not address the problem of assisting users in selecting appropriate UAV software tools based on mission specific operational requirements. This limitation highlights the need for intelligent decision support systems capable of analyzing mission parameters and recommending suitable software configurations for UAV operations particularly in BVLoS scenarios where operational constraints are more complex.

The comparison of the software solutions mentioned above with respect to these requirements is shown in Table 1. Table 1 compares several widely used UAV software platforms with respect to key operational requirements relevant for BVLoS missions including navigation, vehicle state tracking, mission planning, geospatial/regulatory compliance, safety mechanisms, battery management and BVLoS support. From the comparison, it can be observed that PX4[9] and ArduPilot[10] provide the most comprehensive support across these requirements. Both platforms offer integrated functionalities for navigation, real time state monitoring, mission planning and battery management making them suitable for complex autonomous UAV operations. In contrast, DroneKit[11] provides only partial support for several features as it mainly operates as an API layer that interacts with autopilot systems rather than providing a full operational stack. As a result, functionalities such as safety monitoring and battery management are only partially implemented or rely on external modules. Platforms such as DroneUp[12] and Skydio Autonomy[13] provide certain advanced capabilities particularly in navigation and mission execution but they offer limited BVLoS support due to restrictions related to operational range and system extensibility. Finally, DJI Terra[14] and PIX4D[15] are primarily designed for mapping and photogrammetry applications which explains the absence of several operational features such as vehicle state monitoring or battery management. Although they provide strong geospatial processing capabilities they are not designed as complete mission control frameworks. Overall, the comparison highlights that no existing platform simultaneously addresses all operational requirements together with explicit support for parameter interdependencies which motivates the need for the recommendation based framework proposed in this work.

Table 2 compares the same software solutions based on some global features such as open source, SDK/API (Software Development Kit/Application Program Interface), multi-platform support, simulator support, data privacy, customization, and extensibility which are required for BVLoS simulation. Open source provides transparency to alter or add features to the software. SDK/API support facilitates integration with other systems, empowering developers to build custom applications or control drones programmatically. Multi-platform support ensures the program to work on a range of devices and operating systems, which improves its usability. Before drone missions are ever deployed, users can test and refine them in a simulated setting with simulator support. Data privacy protects private user information and guarantees legal compliance. Last but not least, users can add more plugins or features to the software to increase its usefulness or customize it to meet their own needs. Table 2 indicates that PX4[16] and ArduPilot[17] stand out due to their open-source nature and high level of extensibility allowing researchers and developers to customize algorithms integrate new modules and adapt the systems to different mission scenarios. These platforms also support simulators which enables safe testing of complex flight behaviors before real world deployment. On the other hand, commercial solutions such as DroneUp[18], Skydio Autonomy[19], DJI Terra[20] and PIX4D[21] are mostly closed-source platforms which limits their adaptability for research and experimentation. Although some of them provide APIs for integration, their internal architecture is not fully accessible, restricting deeper system customization. Another important feature is simulator support, which is available in PX4[22] and ArduPilot[23] but limited or absent in most other platforms. Simulation capabilities are particularly important for BVLoS missions where testing complex scenarios in a safe environment is critical. These observations indicate that while some platforms provide strong individual capabilities, none of them simultaneously offers full transparency, extensibility and support for analyzing parameter dependencies across mission requirements. This limitation further motivates the need for intelligent recommendation tools capable of assisting UAV operators in tuning mission parameters effectively. A detailed presentation of each of the software mentioned above is available in GitHub (2025c).

We have seen that most of the software is not open source, which is a problem from a research perspective. Also, simulation support is not present, and software is not customizable in most cases. Finally, the functionalities/requirements supported by the software can be interdependent, as they rely on multiple cross-domain operational parameters. So, tuning one parameter to get better functionality in one aspect may impact other parameters, which deteriorates other functionality. According to the best of our knowledge, there is no such work that addresses this important issue in the BVLoS missions specific context.

In the rest of this work, we analyze the interdependencies among the requirements and parameters mentioned above both qualitatively and quantitatively, so that we can fine-tune them and generate a recommendation system for the success of drone BVLoS missions.

3. Proposed methodology

In this Section, we introduce a generic framework to model a recommendation system for a drone's successful BVLoS mission. It has two main components: *Taxonomy creation* and *Recommendation System Design*. The procedure is detailed below:

Taxonomy creation: At first, based on a literature survey, we identify the parameters that are usually associated with the requirements

[9] 2
[10] 3
[11] 4
[12] 5
[13] 6
[14] 7
[15] 8
[16] 2
[17] 3
[18] 5
[19] 6
[20] 7
[21] 8
[22] 2
[23] 3

Table 1
Comparative analysis of the software based on requirements.

Requirements	PX4	Ardu Pilot	DroneKit	DroneUp	Skydio Autonomy	DJI Terra	PIX4D
Navigation	Yes	Yes	Partially (API)	Yes	Yes (short range)	No	No
Vehicle State/position tracking	Yes	Yes	Limited	Yes	Yes	No	No
Mission Planning	Yes	Yes	Yes	Yes	Yes (short range)	Yes	No
Geospatial and Regulatory	Yes	Yes	Limited	Yes	Limited	Limited	Yes
Safety	Yes	Yes	Limited	Yes	Yes	No	No
Battery Management	Yes	Yes	Limited	Yes	Limited	No	No
BVLoS	Yes	Yes	Limited	No	Limited	No	No

Table 2
Comparative analysis of the software based on features.

Features	PX4	Ardu Pilot	DroneKit	DroneUp	Skydio Autonomy	DJI Terra	PIX4D
Open Source	Yes	Yes	Yes	No	No	No	No
SDK/API	Yes	Yes	Yes	Yes	Yes	No	No
Multi-Platform Support	Yes	Yes	Yes	Yes	Limited	Yes	Yes
Simulator Support	Yes	Yes	No	Limited	Yes	No	No
Data Privacy	Yes	Yes	Limited	Yes	Yes	Yes	Yes
Customization and Extensibility	High	High	Moderate	Limited	Limited	Limited	No

needed for the BVLoS missions. Then the parameters are classified based on whether they can be dynamically tuned by the software or are dependent on static hardware, known as *dynamic parameters* and *static parameters* respectively. We also find out the *admissible* and *desirable range of values* for each of the parameters. This leads to a generalized *taxonomy* of requirements and parameters for a drone's BVLoS mission. This *taxonomy* will provide the qualitative and quantitative information for individual parameters and requirements that are useful to the user for any specific drone mission. This *taxonomy* is created once, however it is upgradable in an agile manner.

Recommendation System Design: When a user provides a specific *Software Requirement Specification (SRS)* document, the system can identify the *functional* and *non-functional requirements*. Then with the help of the *taxonomy*, the dependencies among the requirements and parameters are analysed and a visual representation is created by constructing weighted *dependency graphs*. Finally, the system recommends values of the parameters to the user to get optimal results.

The workflow of our proposal is shown in Fig. 2. In the next subsections, we illustrate each step with more detail.

3.1. Taxonomy creation

Algorithm 1 defines the formation of the *taxonomy* from the literature survey. It has three main phases: parameter identification for individual requirements, segregating the parameters into *static* and *dynamic*, and identifying *admissible* and *desirable ranges* of each parameter. In the next subsections we will describe each with examples.

3.1.1. Parameter identification

To create a generalized *taxonomy*, at first, we need to find out the requirements for drone's BVLoS operation, and the detail information about the parameters that affect each of the requirements. By exploring the research papers (journals, conferences), websites of drone software, or any other regulatory information for drones, we can get the information about the requirements and parameters. By linking each parameter to its corresponding requirement, a clear understanding of the system's dependencies is established, ensuring accurate tracking, evaluation, and optimization during development and implementation.

For example, as illustrated in Table 1, different requirements for the user can be navigation, mission planning, safety, energy management etc. However, some *non-functional requirements* also exist such as link reliability, latency etc. The corresponding parameters of a few requirements are described below:

Algorithm 1 Taxonomy formation.

Input: L: Literature survey (journals, conferences, drone software websites, regulatory information)
Output: Taxonomy $\mathcal{T}$
Initialize: Taxonomy $\mathcal{T} = \{\}$
for each requirement $r_i \in L$ **do**
  \\Identify parameters for requirements
  Identify set of parameters $P_i \leftarrow L$ \\referred in Section 3.1.1
  \\Segregate parameters and assign range of values
  Static parameters $Stat_i = \{\}$
  Dynamic parameters $Dyna_i = \{\}$
  for each $p \in P_i$ **do**
    $Value\_ranges_p = \{\}$
    Identify equation eq_p to compute $p \leftarrow L$ \\referred in Section 3.1.1
    if p is Hardware Dependent **then** \\referred in Section 3.1.2
      $Stat_i = Stat_i \cup \{p\}$
    else
      $Dyna_i = Dyna_i \cup \{p\}$
    end if
    $Value\_ranges_p = \langle lower\_lim, upper\_lim \rangle \leftarrow L$ \\referred to Section 3.1.3
  end for
  Create tuple $T_i = \langle r_i, P_i, Stat_i, Dyna_i, \cup_{p \in P_i} Value\_ranges_p, \cup_{p \in P_i} \{eq_p\} \rangle$
end for
$\mathcal{T} = \mathcal{T} \cup \{T_i\}$
return $\mathcal{T}$

Navigation (N). Navigation (Patrik et al., 2019) depends upon the velocity (u), bearing angle (θ_w) and heading of drone (H). Bearing angle is the directional angle from the drone's current position to the target location. Heading of drone is the direction the drone, currently facing, which is measured as an angle relative to a fixed reference frame (North)

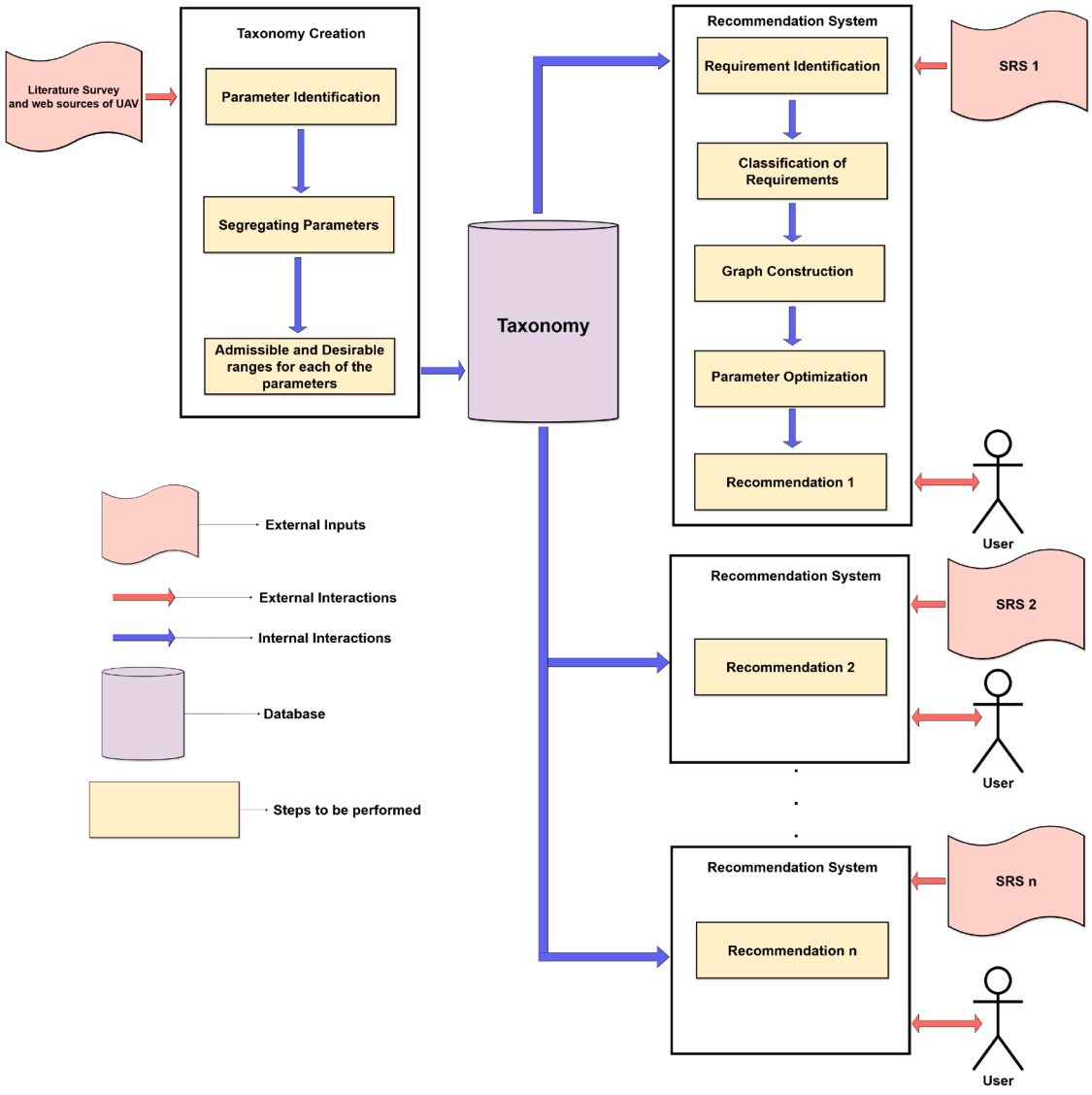


Fig. 2. Overall architecture of the system.

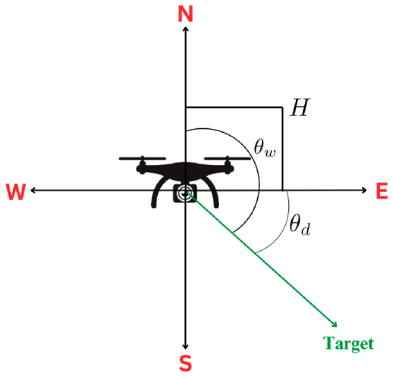


Fig. 3. Drone angle, bearing angle, and the heading of drone.

and drone adjustment angle (θ_d) as shown in Fig. 3. The corresponding equation is represented as $\theta_d = \theta_w - H$ (Patrik et al., 2019).

Safeness/Risk (S). There can be two types of safeness, ground safeness (GS) (Sorbelli et al., 2023, 2024a,c) and air safeness (AS) (Sorbelli et al., 2024b). Ground safeness represents the frequency of casualty occurring due to a drone fall and is also referred to as the probability per hour of a lethal accident (Sorbelli et al., 2024a). On the other hand, air safeness represents the risk due to collision with other UAVs, birds or air obstacles.

The probability of ground safeness $\mathcal{P}_{GS}$ depends upon the probability of casualty $\mathcal{P}_{casualty}$ due to a drone crash (Primatesta et al., 2020a).

The $\mathcal{P}_{casualty}$ depends upon the probability of a crashing event ($\mathcal{P}_{event}$), the probability to impact a person ($\mathcal{P}_{impact}$) and probability of causing fatal injuries ($\mathcal{P}_{fatality}$). $\mathcal{P}_{impact}$ depends on the population density at a given location ρ, impact angle of drone ψ towards the ground after a crash and the area exposed to the crash $A_{\exp}(\psi)$. $\mathcal{P}_{fatality}$ depends upon the kinetic energy at impact, and sheltering factor. The corresponding equations to calculate each parameter are listed below:

where r_p and h_p are the average radius and height of a person respectively, r_{uav} is the radius of the drone, E_{imp} is the kinetic energy at impact, SF is the sheltering factor, α, β are the impact energy for a fatality, v_{imp} is the impact velocity, δ is the drone's mass, v_{init} is initial vertical velocity, g is the acceleration due to gravity, and ζ is the height from which the drone is falling.

The probability of air safeness ($\mathcal{P}_{AS}$) mostly depends upon the number of air obstacles or drones (Sorbelli et al., 2024b) in a drone swarm.

Energy consumption (E). Energy consumption (E) depends on path-length (p_L) and halting time (H_T). As path-length increases the flight duration of the drone, it enhances the energy consumption. Likewise halting time significantly impacts energy use, as drones require continuous thrust to hover at the halting points (for example for data collection) which consumes battery power even when stationary.

Link reliability (LR). For BVLoS, the UAV has to maintain seamless connection with the ground control station (GCS) via wireless links. This long distance communication is established either by satellite (SATCOM) (Ghamari et al., 2022) or cellular networks (Azari et al., 2020; Fotouhi et al., 2019). As SATCOM is costly and technically complex (Ghamari et al., 2022), the existing cellular networks can be used for wide applications. The probability of wireless link reliability $\mathcal{P}_{LR}$ depends on the probability of line of sight of drone with a cellular tower and received signal power from tower $\mathcal{P}_W$ (Sorbelli et al., 2023, 2024a). The corresponding equations (Sorbelli et al., 2023, 2024a) are :

$$\mathcal{P}_{LR} = \mathcal{P}_{LOS} \cdot P_W$$

$$\mathcal{P}_{LOS} = \frac{1}{1 + \tau e^{-\omega\left(\arctan\frac{\zeta}{d_G}\right)-\tau}}$$

$$P_W(x,y,t) = P_W(t)\, G(t)\, G(x,y)\left(\frac{\lambda}{4\pi d_s}\right)^2$$

$$d_G = \sqrt{\left(x_{(x,y)} - x_t\right)^2 + \left(y_{(x,y)} - y_t\right)^2}$$

$$d_S = \|(x,y) - t\|_2$$

where d_G is the ground distance between the drone and the tower; ζ is the altitude of drone; ω, τ are the S-curved parameters; d_S is the Euclidean slant distance that separates the antennas between drone and the tower; G is the antenna gains of the transmitting and receiving devices; and λ is the wavelength representing the effective aperture area of the receiving antenna.

Latency/Delay (D). Latency/Delay (D) depends on path length (p_L) of drone's trajectory, velocity (u) of the drone, number of halting points in between a path (H_P), where the drone temporarily stops for data collection or doing some activity (e.g. spraying pesticides in fields), and halting time (H_T) (Sorbelli et al., 2024a).

Table 3 summarizes the mentioned requirements with associated parameters.

Table 3
Requirements with parameters.

Requirements	Parameters	Symbols used
Navigation (N)	Velocity	u
	Drone adjustment angle	θ_d
	Bearing angle	θ_w
	Heading of drone	H
Safeness (S)	Probability of ground safeness	$\mathcal{P}_{GS}$
	Probability of air safeness	$\mathcal{P}_{AS}$
	Probability of casualty	$\mathcal{P}_{casualty}$
	Probability of event	$\mathcal{P}_{event}$
	Probability of impact	$\mathcal{P}_{impact}$
	Probability of fatality	$\mathcal{P}_{fatality}$
	Population density	ρ
	Area of exposure	$A_{\exp}$
	Radius of person	r_p
	Height of person	h_p
	Radius of drone	r_{uav}
	Impact angle	ψ
	Event type	E_{type}
	Impact of Kinetic energy	E_{imp}
	Mass of drone	δ
	Impact velocity	v_{imp}
	Initial vertical velocity	v_{init}
	Altitude of drone	ζ
	Acceleration due to gravity	g
	Sheltering factor	SF
	Number of drones	n
	Impact energy for fatality	α, β
Link Reliability (LR)	Probability of line of sight	$\mathcal{P}_{LOS}$
	Received signal power	P_W
	Altitude of drone	ζ
	Antenna gain of transmitter (tower)	$G(t)$
	Antenna gain of receiver (drone)	$G(x, y)$
	Wavelength of receiving antenna	λ
	Euclidean distance between drone and tower	d_S
	Ground distance between drone and tower	d_G
	S-curved Parameters	τ, ω
Latency/Delay (D)	Path length	p_L
	Velocity	u
	No. of halting points	H_P
	Halting time	H_T
Energy Consumption (E)	Path length	p_L
	Halting time	H_T

3.1.2. Segregating parameters

After identifying the parameters, we have to identify which parameters can be tuned by software dynamically during the mission and which parameters we cannot. A few parameters, like the velocity of the drone or the altitude at which the drone flies, can be changed through onboard programming. These are called *dynamic parameters*. On the other hand, the physical components of the system or the static environmental conditions, such as mass or size of the drone, population density, and obstacles, remain static throughout the mission. These are called *static parameters*. For example, based on the parameter list in Table 3, we classify the *static* and *dynamic parameters* as shown in Table 4 .

3.1.3. Admissible and desirable ranges

Once the parameters have been established, it is necessary to identify the *admissible* and *desirable ranges* for each of them. The *admissible range* refers to the minimum and maximum limits within which a parameter can function without causing failure or instability, which defines the extreme limits that a system can tolerate. Whereas the *desirable range* is the optimal subset within the *admissible range* where performance, efficiency, and safety are maximized for stable and reliable operation. In the proposed recommendation system, we try to limit the values within the *desirable ranges* for optimized performance.

These values are elicited through relevant literature survey and regulatory information regarding drones. For instance, in terms of altitude, a drone may have an *admissible range* of $10m$ to $15,000m$, but a review of the relevant literature (T-DRONES, 2025a) suggests a *desirable range* of $30m$ to $120m$ for stable operation and regulatory compliance. Unlike *dynamic parameters* (e.g., velocity or altitude) where the *admissible range* indicates the extreme operational limits and the *desirable range* denotes the optimal subset for efficiency, *static parameters* such as the mass of the drone, radius of the drone, height or radius of a person, population density, or antenna gain are fixed by design specifications, physical laws, or environmental conditions. Hence, for *static parameters*, the *admissible* and *desirable ranges* are the same during the mission. We identified the *admissible* and *desirable ranges* of a few *static* and *dynamic parameters* as listed in Table 5 and 6.

The architectural diagram of our proposed method for *taxonomy* creation is depicted in Fig. 4. The *taxonomy* is further used to analyze interdependencies among requirements and parameters, to suggest optimized values for the parameters towards a recommendation system for drone's BVLoS mission.

3.2. Recommendation system design

After creating the *taxonomy*, the proposed recommendation system reads the specific SRS, given by the user, and provides the optimized range of values for mission specific parameters for a drone's BVLoS

Table 4
Static and *dynamic* parameters.

Static		Dynamic	
Parameter	Symbol	Parameter	Symbol
Mass of drone	δ	Path length	p_L
Radius of drone	r_{uav}	No. of halting points	H_P
Radius of person	r_p	Halting time	H_T
Height of person	h_p	Velocity	u
Impact energy for fatality	α, β	Impact angle	ψ
Antenna gain of transmitter	$G(t)$	Initial vertical velocity	v_{init}
Antenna gain of receiver	$G(x, y)$	Impact velocity	v_{imp}
No. of drones	n	Bearing angle	θ_w
Sheltering Factor	SF	Heading of drone	H
Population Density	ρ	Event type	E_{type}
Wavelength of receiving antenna	λ	Altitude of drone	ζ
Acceleration due to gravity	g	Drone angle	θ_d
		Euclidean distance between drone and tower	d_S
		Ground distance between drone and tower	d_G

Table 5
Admissible and *desirable ranges* of *static parameters*.

Parameters	Admissible / Desirable range	Ref.
Mass of drone (δ)	0.5 kg to 3.75 kg	Primatesta et al. (2020a)
Radius of drone (r_{uav})	0.2 to 0.88 m	Primatesta et al. (2020a)
Radius of person (r_p)	0.2025 to 0.23 m	First in Architecture (2025)
Height of person (h_p)	1.65 to 1.74 m	First in Architecture (2025)
α	100 J	Primatesta et al. (2020a)
β	34 J	Primatesta et al. (2020a)
Sheltering Factor (SF)	0 to 10	Primatesta et al. (2020a)
Population Density (ρ)	R*- 141 to 172 p/km^2, S*- 866 to 1287 p/km^2, U*- 2135 to 7430 p/km^2	Sorbelli et al. (2024a)

* → different cities from Italy, R→Rural, S→Suburban, U→Urban, $p/km^2 \rightarrow people/km^2$

Table 6
Admissible and *desirable ranges* of *dynamic parameters*.

Parameters	Admissible	Desirable	Ref.
Path length (p_L)	More than 644 km	0.9144 - 30 km	T-DRONES (2025b)
Velocity (u)	16 to 2173 km/h	48 to 145 km/h	JOUAV (2025)
Impact angle (ψ)	50° - 70°	≤ 60°	EASA (2025)
Event type (E_{type})	Ballistic descent, Uncontrolled Glide, Parachute descent, Flyaway	Parachute descent, Flyaway	Primatesta et al. (2020a)
Altitude (ζ)	10 to 15000 m	30 to 122 m	T-DRONES (2025a)

mission. The recommendation system consists of four phases: Requirement identification, Requirement classification, Dependency graph construction, and Parameter optimization. In the next subsections, we will describe each step in details.

3.2.1. Requirement identification

In order to determine the set of requirements for a specific mission, we first need to consider and analyze the information provided on the software requirement specification (SRS). The SRS generally includes the software's capabilities, behaviors, and rules that must be followed to meet user needs. In more detail, SRS frequently outlines the precise functions, interactions, and results that the system must provide. They also cover topics including the system's usability, performance, and interoperability.

In the rest of the paper we refer without loss of generality to a generic SRS document (GitHub, 2025b) that is the comprehensive blueprint for developing and maintaining the software. Based on that, we prepare the set of requirements such as Navigation(N), GPS Based Positioning(GPS), Delivery(DV), Obstacle Detection(OD), Safeness/Risk(S), Link Reliability(LR), Latency/Delay(D), Energy Consumption(E).

3.2.2. Classification of requirements

A clear grasp of the requirements, parameters, and expectations of the system is ensured by segregating requirements into *functional* and *non-functional* categories as defined below (Bag et al., 2025).

Functional Requirement (FR): Functional Requirement (FR) describes the system functionality, indicating what the system should do to meet user needs and achieve its goals (Kurtanović & Maalej, 2017).

Non-functional Requirement (NFR): Non-functional Requirement (NFR) describes the system properties and constraints, such as performance, security, and usability, which define how the system operates (Kurtanović & Maalej, 2017).

For example, based on the SRS in GitHub (2025b) the *functional requirements* are Navigation (N), GPS-based positioning (GPS), Delivery (DV) and Obstacle Detection (OD). The *non-functional requirements* are Safeness/Risk (S), Link reliability(LR), Latency/Delay (D) and Energy consumption (E). Hence, $FR = \{N, GPS, DV, OD\}$ and $NFR = \{S, LR, D, E\}$.

3.2.3. Dependency graph construction

After classifying the requirements, these are used to build a graph structure that visually maps out how different parameters for each

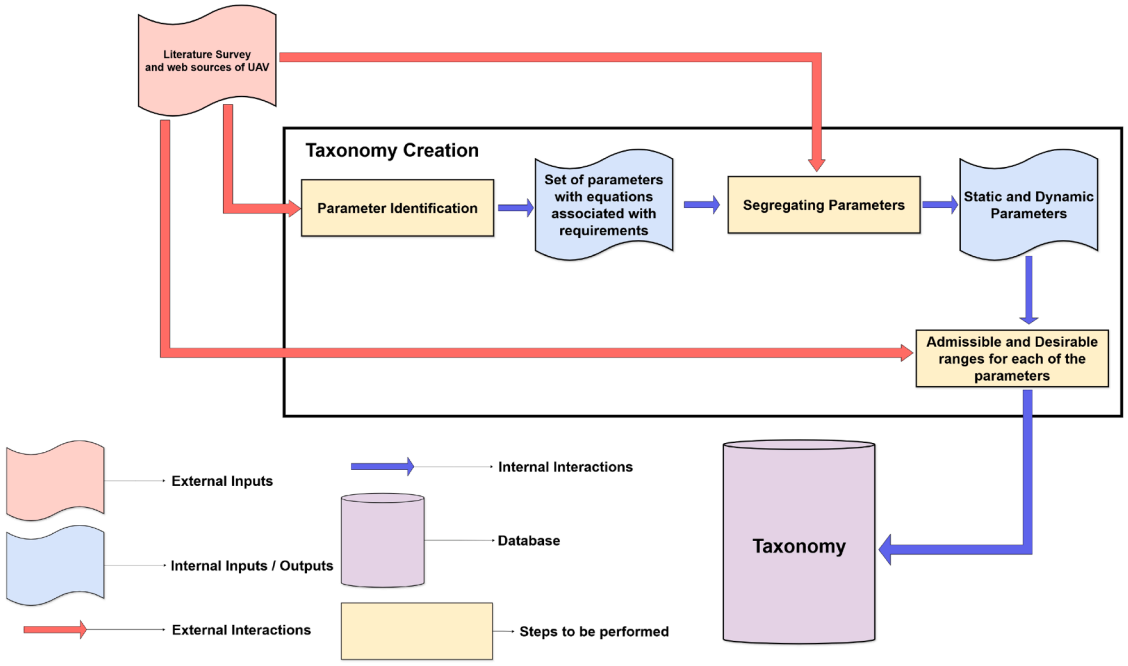


Fig. 4. Architectural diagram of *taxonomy* creation.

requirement relate and depend on each other. This leads to the creation of a weighted *dependency graph*, which shows the connections among the parameters for each requirement, providing a visual structure for further analysis and optimization. The weights are put on the edges based on *desirable ranges* as it is the optimal subset within the *admissible range* where performance, efficiency, and safety are maximized for stable and reliable operation. A few definitions related to the graph construction are given below.

Dependency Graphs: A dependency graph is defined as a formal representation of system requirements, parameters and their dependencies. It consists of vertices representing *functional requirements (FRs)*, *non-functional requirements (NFRs)*, and the associated parameters, along with edges that capture dependencies among them (Roy et al., 2021).

Basic Clusters: Basic clusters focus on the direct dependencies among parameters within individual requirements, providing a granular view of their connections (Roy et al., 2021).

The algorithm for step 3.2.1 to 3.2.3 is shown in Algorithm 2. It starts by initializing an empty graph $\mathcal{G}$ and empty sets for *functional* (FR) and *non-functional* (NFR) *requirements*. The algorithm then identifies the set of requirements from the SRS and classifies each as either *functional* or *non-functional*. For each requirement, it calls a subroutine 3 to create a *basic cluster graph*, which maps the parameters and dependencies associated with that requirement based on the *taxonomy*. Algorithm 3 begins by initializing empty sets for nodes (V_i) and edges (E_i). For each parameter p associated with the requirement, a corresponding node v_p is created and added to the node set. Then, for each equation eq_p related to a parameter, it identifies other nodes (parameters) involved in the computation and creates edges between them and v_p. Each edge is assigned a weight based on the *desirable range* of the connected parameter, representing the level of dependency. The algorithm ultimately returns a *basic cluster graph* G_i representing the internal parameter relationships of the given requirement. These individual *basic cluster graphs* are then added to the main graph $\mathcal{G}$, which is ultimately returned as the output, representing the system's requirement-parameter dependency structure.

For example, Fig. 5a shows a *basic cluster* for the requirement *Risk*. The corresponding color codes are referred to Fig. 5(b). A graph may contain *macro cluster* (Roy et al., 2021) as well, which combines multiple basic clusters to illustrate the broader inter dependencies among

Algorithm 2 Requirement identification and graph construction.

Input: $\mathcal{S}$: SRS, $\mathcal{T}$: $Taxonomy$
Output: Dependency graph $\mathcal{G}$
Initialize: $\mathcal{G} = \{\}$, Set of functional requirements $FR = \{\}$, Set of non-functional requirements $NFR = \{\}$
\\Requirement Identification
Identify set of requirements $R \leftarrow \mathcal{S}$ \\referred to Section 3.2.1
\\Requirement Classification
for each $r_i \in R$ **do**
  if r_i defines system functionality **then**
    $FR = FR \cup \{r_i\}$ \\referred to Section 3.2.2
  else
    $NFR = NFR \cup \{r_i\}$ \\referred to Section 3.2.2
  end if
  \\Basic Cluster Construction
  G_i = Create_basic_cluster(r_i, $\mathcal{T}$) \\referred to Algorithm 3
  $\mathcal{G} = \mathcal{G} \cup \{G_i\}$
end for
return $\mathcal{G}$

requirements, offering a comprehensive perspective on how different aspects of the system interact. Analysing inter requirement dependencies is beyond the scope of this paper.

3.2.4. Parameter optimization:

Using the *dependency graph* and SRS, the system identifies which parameters need to be tuned to achieve the best possible outcomes for the UAV system according to the identified requirements. This can be formulated as an optimization problem as given below :

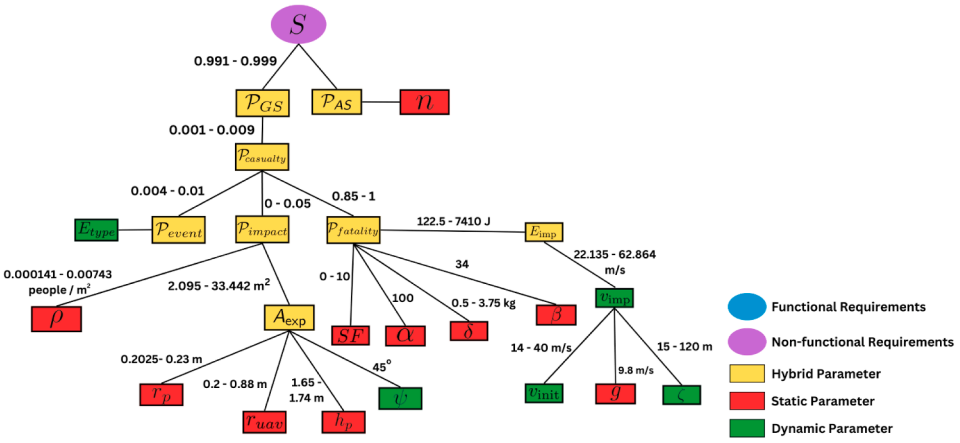
$$\textbf{Maximize:} \quad Z = \sum_{r \in R} Req(r) \tag{1}$$

Subject to:

$$L_i \leq p_i \leq U_i, \quad \forall p_i \in P \tag{2}$$

where:

- R is the set of requirements, with $R = FR \cup NFR$.



(a) Basic Cluster for Safeness/Risk(S) (b) Color Code

Fig. 5. Basic cluster.

Algorithm 3 Create_basic_cluster(r_i, $\mathcal{T}$).

Input: r_i: Requirement , $\mathcal{T}$: Taxonomy
Output: cluster G_i
Initialize: Node set $V_i = \{\}$, Edge set $E_i = \{\}$
\\Create node set
for each parameter $p \in P_i \leftarrow \mathcal{T}$ **do** \\$P_i$: Set of parameters for r_i
  Create node v_p
  $V_i = V_i \cup \{v_p\}$
end for
\\Create edge set
for each equation $eq_p \in T_i$ **do**
  Identify nodes $U_i \in V_i$ to compute eq_p
  for each node $u \in U_i - v_p$ **do**
    Create edge e between v_p and u
    Assign edge weight
    $E_i = E_i \cup \{e\}$
  end for
end for
return cluster G_i

- P is the set of parameters.
- L_i, U_i denote the desirable lower and upper bounds of each parameter p_i.

An optimization Algorithm 4 is applied on the *dependency graphs* to find the best values for these parameters while respecting system constraints.

It starts by reading the user inputs provided in the SRS. Each parameter is checked: if it has no children (i.e., nothing else depends on it), it is treated as a leaf node. The algorithm then begins solving these leaf parameters first, using the equations defined in the *taxonomy* and the values already known from the SRS. As values are computed, the algorithm moves upward in the graph, updating parent parameters step by step until all dependencies are resolved. This bottom up approach ensures that every parameter is calculated only after its prerequisites are available. Finally, the output of the algorithm yields a recommendation system that provides the user with clear, actionable suggestions on how to adjust parameters to maximize performance and meet requirements efficiently.

The architecture diagram of the recommendation system is depicted in Fig. 6. Considering different scenarios, we have demonstrated different use cases where based on the inputs by the user (from SRS), the Algorithm 4 recommends optimized values for individual parameters for a specific requirement. Those are described below.

Algorithm 4 Optimization.

Input: $\mathcal{G}$: Cluster graph, $\mathcal{S}$: SRS
Output: Optimized values for parameters
Initialize: Create stack $S = \{\}$
$Input\_values\{\langle p_1, p_1.val\rangle, \langle p_2, p_2.val\rangle, \ldots.\} \leftarrow \mathcal{S}$
\\input user-defined values for parameters
for each cluster $G \in \mathcal{G}$ **do**
  \\push all nodes
  for each $v \in G$ **do**
    $v.visited = false$
  \\mark leaf-nodes
    if $child(v)$ == NULL **then** \\$child(v)$: set of children of node v
      $v.leaf = true$
    else
      $v.leaf = false$
    end if
    $S.push(v)$
  end for
  \\pop nodes to compute
  while S ! = NULL **do**
    $v_p = S.pop()$
  \\compute if leaf
    if $v_p.leaf == true$ **then**
      $v_p.val \leftarrow$ solution of $\{e_w(v_p, u) : u \in child(v_p)\}$
        using eq_p on $Input\_values \cup \{v_q.val : q \neq p\}$
      print $\langle p, v_p.val\rangle$
      $v_p.visited = true$ \\Flag visited nodes
    else \\mark as leaf, if all children are computed
      if $\forall u \in child(v_p) : u.visited = true$ **then**
        $v_p.leaf = true$
        $S.push(v_p)$
      end if
    end if
  end while
end for

Use Case 1

We have considered Probability of Ground Safeness ($\mathcal{P}_{GS}$) as a requirement, considering an urban region (Kolkata (22.654899, 88.439464)) having population density (ρ) = 0.024 $people/m^2$, where a Tylon aircraft (Primatesta et al., 2020a) is used. Some values of the parameters are provided in the SRS, and the rest of the values are taken from the taxonomy as given below:

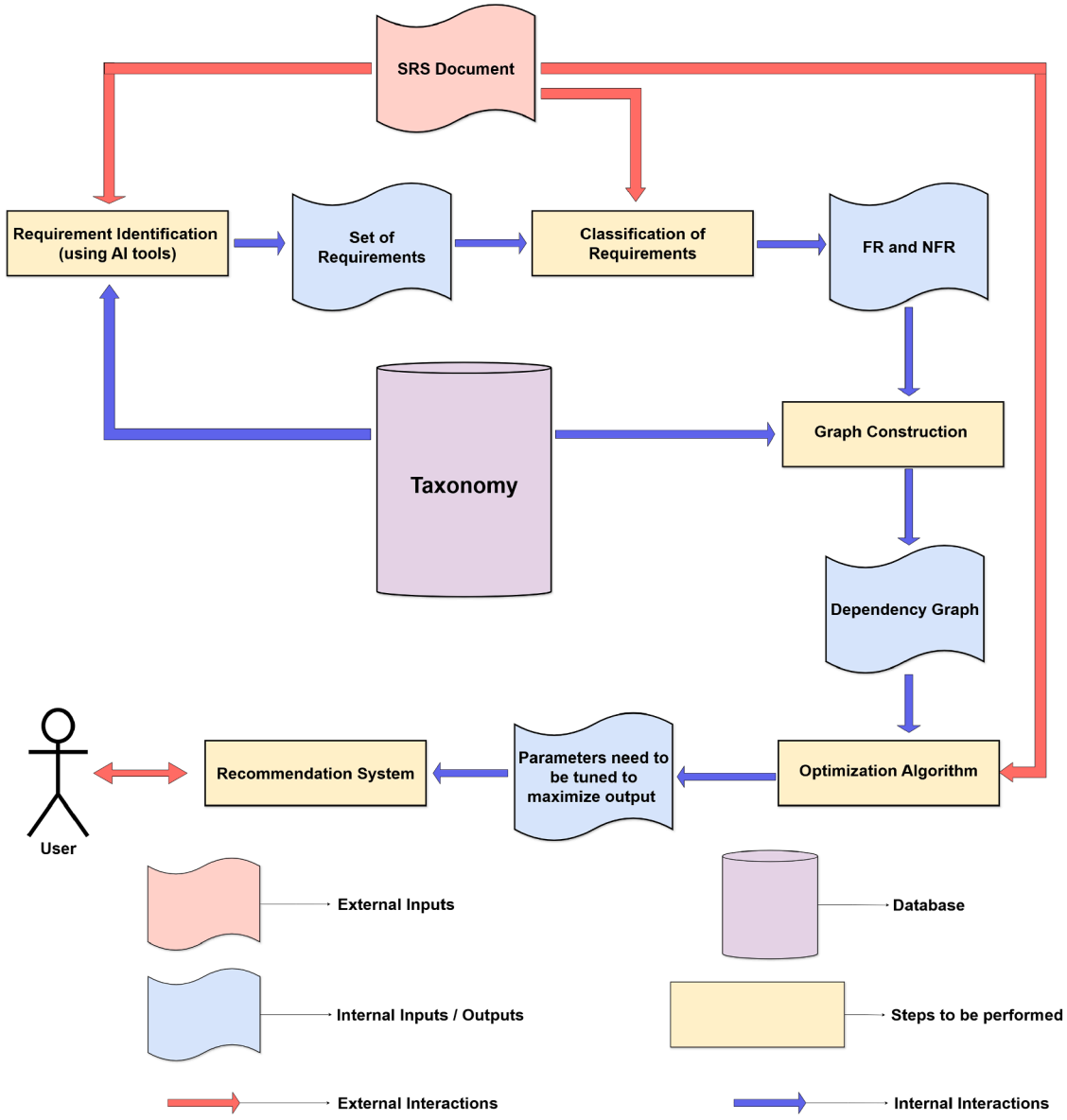


Fig. 6. Architectural diagram of the recommendation system.

- **Values from SRS:** Mass of drone (δ) = 3.75 kg, radius of drone (r_{uav}) = 0.88 m, altitude (ζ) = 120 m, initial velocity (u) = 40 m/s
- **Values from Taxonomy:** Radius of person (r_p) = 0.2032 m, height of person (h_p) = 1.65 m, impact angle (ψ) = 45°, Sheltering Factor (SF) = 7.5 and $\mathcal{P}_{event}$ = 0.005

Based on these values, the recommendation system gives the results as: impact velocity (v_{imp}) = 62.864935 m/s, impact of kinetic energy (E_{imp}) = 7410 J, $\mathcal{P}_{impact}$ = 0.174211, $\mathcal{P}_{fatality}$ = 0.914184, $\mathcal{P}_{casualty}$ = 0.000796, $\mathcal{P}_{GS}$ = 0.999204

Use Case 2

Now consider a suburban region (Barasat(22.724610, 88.484729)) having population density(ρ) = 0.0082 $people/m^2$, where a Parrot Disco aircraft (Primatesta et al., 2020b) is used for simulating Probability of Ground Safeness ($\mathcal{P}_{GS}$) as a requirement.

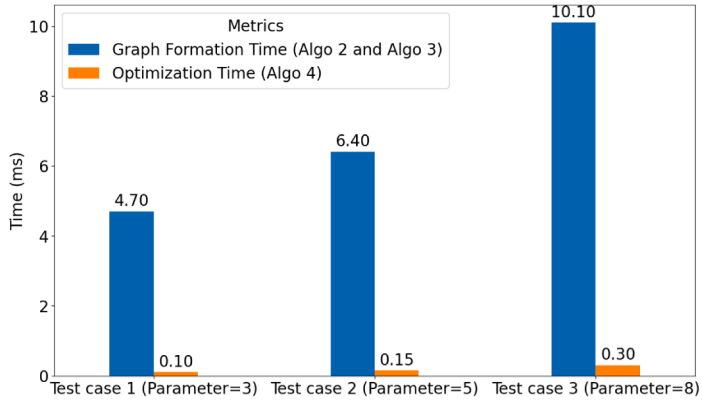
- **Values from SRS:** Mass of drone (δ) = 0.75 kg, radius of drone(r_{uav}) = 0.575 m, altitude(ζ) = 120 m, initial velocity(u) = 30 m/s.
- **Values from Taxonomy:** Radius of person(r_p) = 0.2032 m, height of person(h_p) = 1.65 m, impact angle(ψ) = 45°, Sheltering Factor(SF) = 5(Considering sub urban region Primatesta et al., 2020a) and $\mathcal{P}_{event}$ = 0.005(Considering E_{type} = ballistic descent Primatesta et al., 2020a)

Hence the respective outputs are impact velocity(v_{imp}) = 57.02631 m/s, impact of kinetic energy(E_{imp}) = 1219.5 J, $\mathcal{P}_{impact}$ = 0.036651, $\mathcal{P}_{fatality}$ = 0.913668, $\mathcal{P}_{casualty}$ = 0.000167, $\mathcal{P}_{GS}$ = 0.0.999833

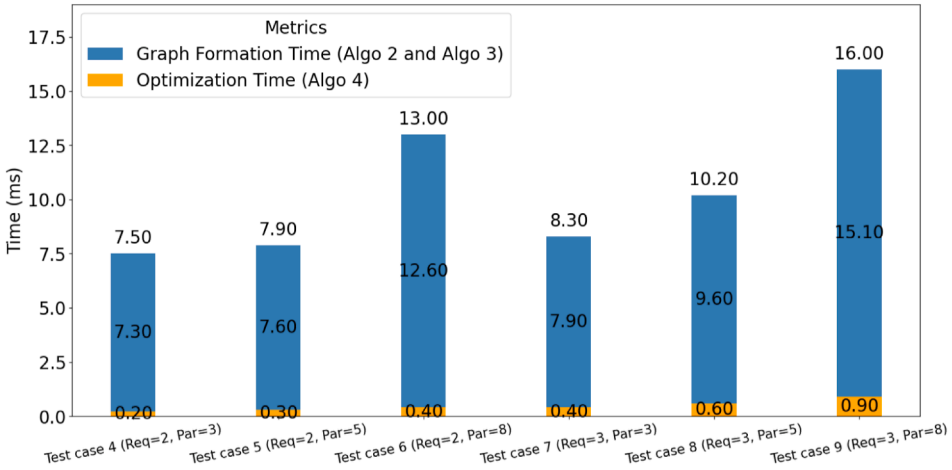
Use Case 3

Finally, consider a Rural region (Anandapur(22.738762, 87.738036)) having population density(ρ) = 0.000604 $people/m^2$, where a DJI Mavic aircraft (Primatesta et al., 2020a) is used simulating Probability of Ground Safeness ($\mathcal{P}_{GS}$) as a requirement.

- **Values from SRS:** Mass of drone (δ) = 0.7 kg, radius of drone(r_{uav}) = 0.2 m, altitude(ζ) = 120 m, initial velocity(u) = 20 m/s.
- **Values from Taxonomy:** Radius of person(r_p) = 0.2032 m, height of person(h_p) = 1.65 m, impact angle(ψ) = 45°, Sheltering Factor(SF) = 2.5(Considering rural region Primatesta et al., 2020a)



(a) Varying no. of parameters for single requirement



(b) Varying no. of parameters for multiple requirements

Fig. 7. Computational time analysis.

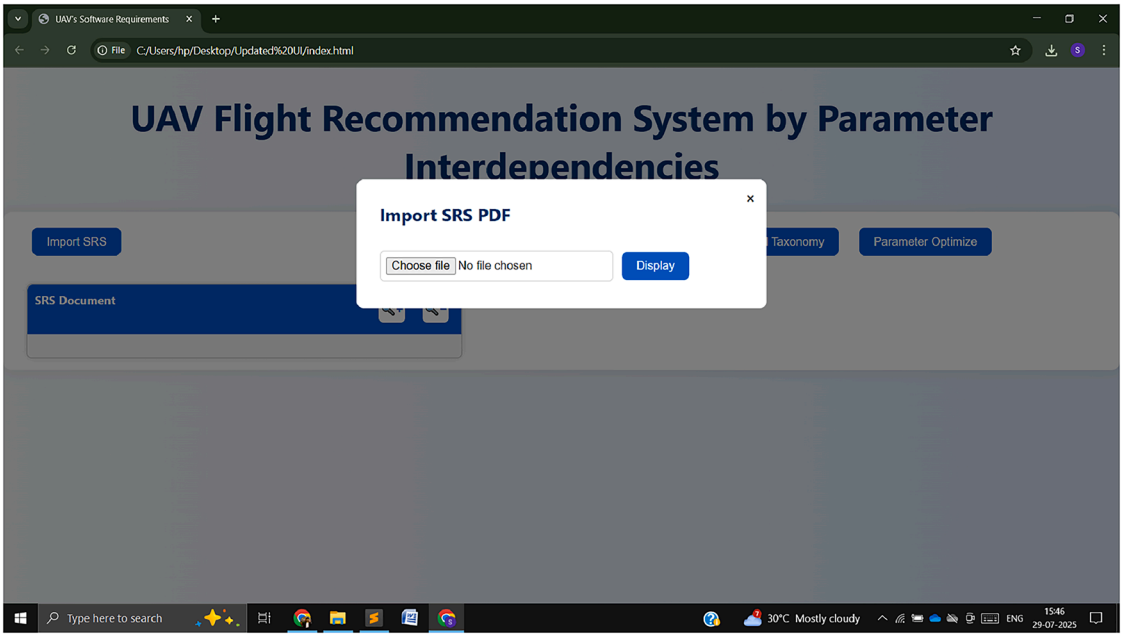


Fig. 8. Importing SRS document.

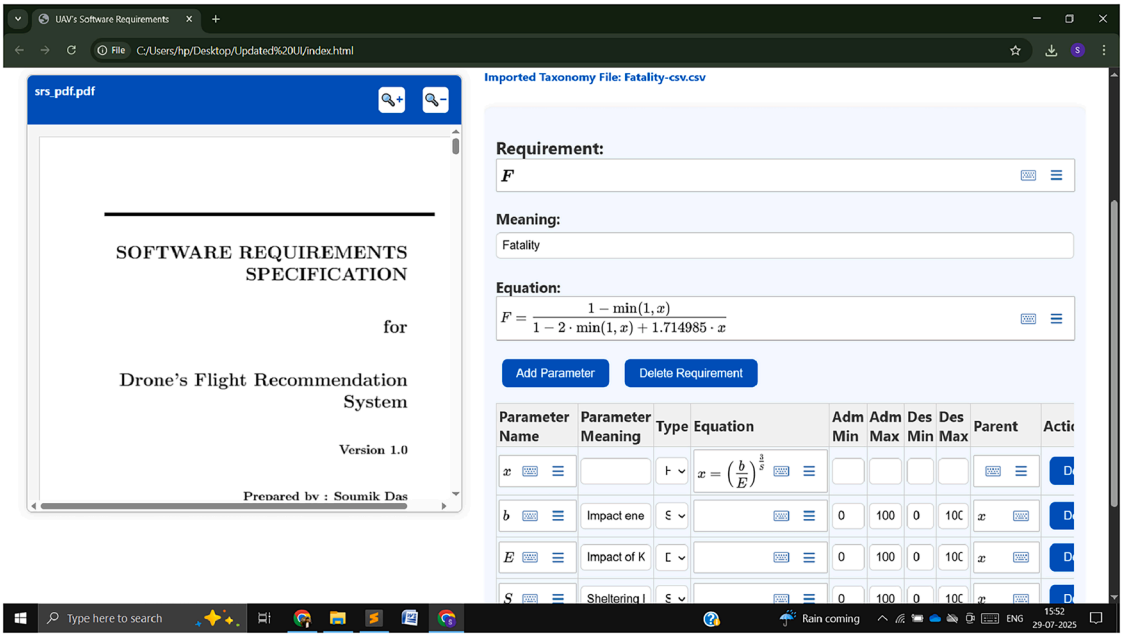


Fig. 9. Displaying the SRS and taxonomy file.

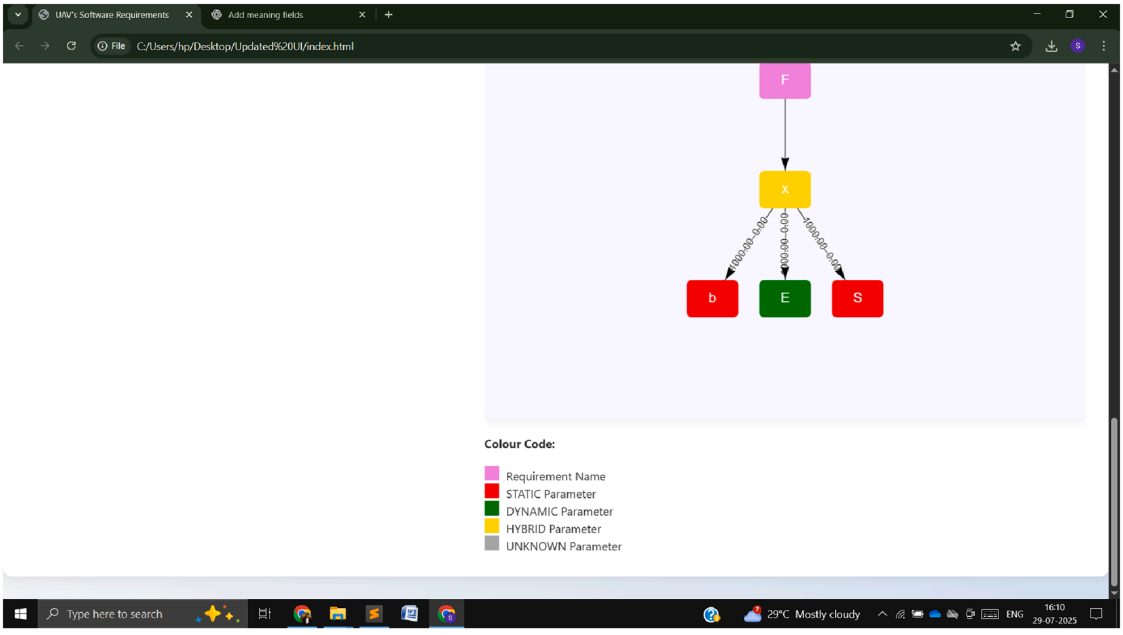


Fig. 10. Displaying basic cluster.

and $\mathcal{P}_{event} = 0.005$(Considering $E_{type} =$ ballistic descent Primatesta et al., 2020a)

Hence the respective outputs are impact velocity(v_{imp}) = 52.459508 m/s, impact of kinetic energy(E_{imp}) = 963.2 J, $\mathcal{P}_{impact} = 0.001112$, $\mathcal{P}_{fatality} = 0.987003$, $\mathcal{P}_{casualty} = 0.000005$, $\mathcal{P}_{GS} = 0.999995$

Table 7 summarizes all the mentioned use-cases.

4. Experimental evaluation

To validate the proposed recommendation system, we perform a set of simulations to demonstrate how parameter interdependencies and optimizations impact UAV BVLoS missions. This allows us to assess how well the recommendation system adapts to varying conditions while ensuring compliance with operational constraints.

4.1. Simulation environment

We perform the simulation on a system having Windows 10 operating system with an Intel Core i5 processor (2.9 GHz, 8 GB RAM). The user interface for the system is built using standard web technologies (HTML, CSS, and JavaScript), which is interactive and lightweight. To handle mathematical input and equations, we integrate the MathLive equation builder. The entire implementation code can be found in GitHub (2025a).

We consider once more the SRS in GitHub (2025b) and we analyze the expected performance of the system based on the execution time,

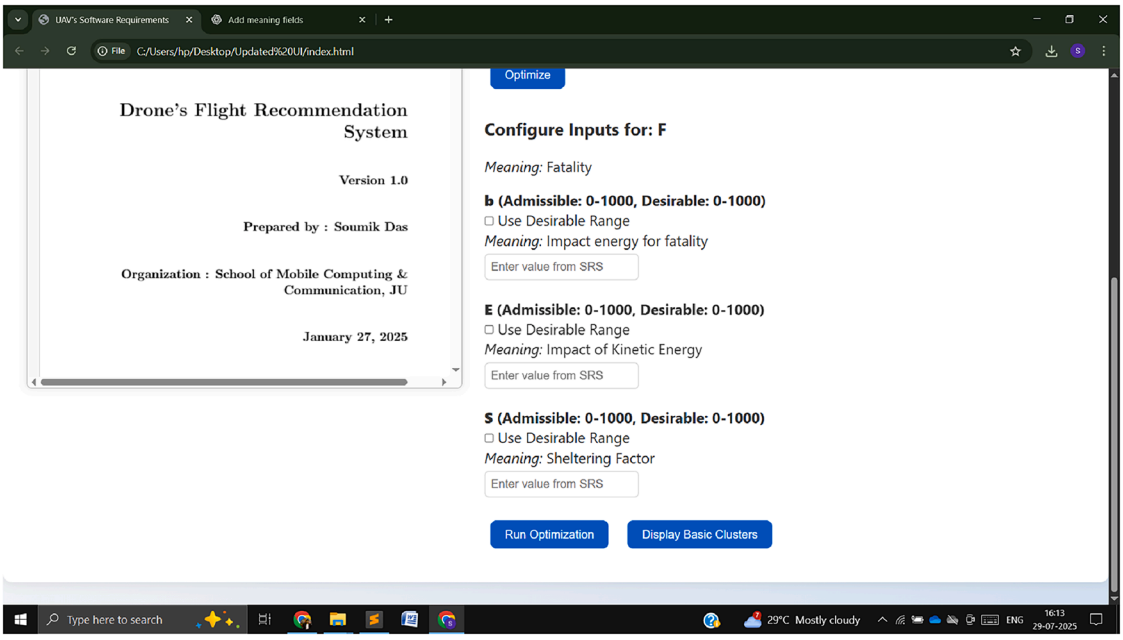


Fig. 11. Configure inputs for the parameters.

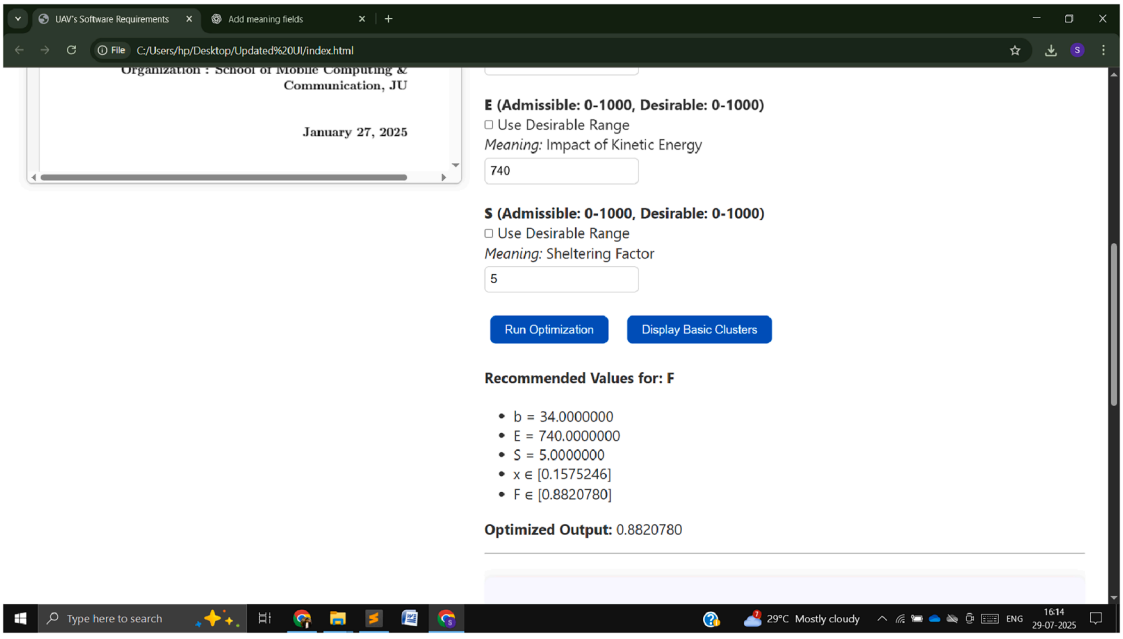


Fig. 12. Recommended values for the parameters and requirement.

Table 7
Use cases.

Input Par	Case 1	Case 2	Case 3	Output Par	Case 1	Case 2	Case 3
u	$40m/s$	$30m/s$	$20m/s$	v_I	$62.864935m/s$	$57.02631m/s$	$52.459508m/s$
ζ	$120m$	$120m$	$120m$	E_I	$7410J$	$1219.5J$	$963.2J$
SF	7.5	5	2.5	P_I	0.174211	0.036651	0.001112
δ	$3.75kg$	$0.75kg$	$0.7kg$	P_F	0.914184	0.913668	0.987003
P_E	0.005	0.005	0.005	P_C	0.000796	0.000167	0.000005
ρ	$0.024p/m^2$	$0.0082p/m^2$	$0.000604p/m^2$	P_{GS}	0.999204	0.999833	0.999995
r_p	$0.2032m$	$0.2032m$	$0.2032m$				
r_u	$0.88m$	$0.575m$	$0.2m$				
h_p	$1.65m$	$1.65m$	$1.65m$				
ψ	45°	45°	45°				

Par→Parameter, $P_E \rightarrow P_{event}$, $P_I \rightarrow P_{impact}$, $P_F \rightarrow P_{fatality}$, $P_C \rightarrow P_{casualty}$, $r_u \rightarrow r_{uav}$, $v_I \rightarrow v_{imp}$, $E_I \rightarrow E_{imp}$, $p/m^2 \rightarrow people/m^2$

varying the inputs (parameters and requirements) to measure the scalability and responsiveness of the system. We measure this separately for different algorithms.

4.2. Results

To systematically evaluate the system, we began by analysing how changes in the number of requirements and parameters influence the overall computational performance. This evaluation helps in assessing both the scalability and responsiveness of the framework when applied to UAV mission planning. Also, we put a few implementation snapshots of the proposed system.

4.2.1. Execution time

To understand how well our framework performs in practice, we carry out a computational time analysis focusing on the two most important algorithms of the system: graph construction (Algorithm 2) and optimization (Algorithm 4). These are the stages that handle most of the processing, as these are executed for each SRS. So their computational efficiency directly impacts the performance of the system.

We consider nine test cases, where number of requirements varies from 1 to 3 and number of parameters varies from 3 to 8. The results for the test cases are shown in Fig. 7. Overall, we can see that the graph construction algorithm has higher computational complexity compared to the optimization algorithm. Graph construction needs to build the entire dependency structure from scratch for every requirement. For each requirement, it first generates nodes for all parameters and then examines the equations associated with each parameter to identify their dependencies. Once the dependencies are known, it creates edges between the relevant nodes and assigns appropriate weights based on the desirable parameter ranges. This process involves checking every parameter pair and performing repeated lookups to ensure that all relationships are captured, which becomes time-consuming as the number of requirements and parameters increases. On the other hand, the optimization algorithm works on the already constructed graph and only evaluates the parameters that are required for the final result. It computes the values of leaf parameters first and then moves upward through the graph in a single pass, which requires far fewer operations.

In Fig. 7(a) we consider one requirement varying the number of parameters. Here, rendering the graph becomes slower while increasing the number of parameters, while the optimization time barely changed. Further, we extended the study by considering multiple requirements, as illustrated in Fig. 7(b). Obviously the computational time increases with the higher number of requirements and parameters. However, it is in the order of 10^{-2} to 10^{-3} seconds, as the graph rendering time ranges from roughly 7 to 15 milliseconds and the optimization time stays between 0.2 and 0.9 milliseconds even when the number of requirements and parameters increases.

Even though we tested with small input sizes, the trend clearly shows that the system is computationally efficient and can comfortably scale to more complex UAV mission scenarios.

4.2.2. Simulation snapshots

The snapshots of different stages of the recommendation system are illustrated below. Initially, in the landing page, we import the SRS (Fig. 8) and then similarly import the Taxonomy file and display these (Fig. 9). After that we can select any of the requirement and display the corresponding graph (Fig. 10). Then we can select any of the requirement for parameter optimization and configure the inputs based on the SRS as well as taxonomy (Fig. 11). Finally we are able to get the recommended values for the requirement and the parameters (Fig. 12).

5. Conclusion

In this work, we introduce a structured framework to study how different requirements and parameters in UAV BVLoS missions are connected and influence one another. By creating a generalized *taxonomy* and using weighted *dependency graphs*, we show that how tuning one parameter can affect multiple aspects of a mission. The proposed recommendation system further demonstrate its ability to suggest optimal parameter settings efficiently, as confirmed through computational time analysis. Altogether, this approach offers UAV operators a practical tool to make mission planning safer, more reliable, and better informed.

Despite these promising results, some limitations of the current work should be acknowledged. First, the proposed framework relies on predefined relationships among parameters derived from domain knowledge and existing documentation. While this allows the system to capture important interdependencies, these relationships may evolve as UAV technologies, regulatory frameworks, and operational requirements change. Second, the current taxonomy focuses on a specific set of parameters relevant to BVLoS missions and may not yet capture all environmental, hardware, or regulatory factors that could influence drone operations. Finally, the evaluation has been performed in a controlled experimental setting, and further validation with large-scale real-world UAV mission data would provide deeper insight into the robustness and practical applicability of the proposed system.

Looking ahead, we plan to expand this framework by introducing macro clusters which depicts the interdependencies among multiple requirements, making it easier to analyze complex missions. This higher-level view will also help in spotting broader interdependencies that may otherwise go unnoticed. In addition, we aim to test the system with real-world UAV datasets and explore adaptive optimization methods that can adjust to changing mission conditions. These improvements will make the recommendation system more scalable, flexible, and resilient for future BVLoS operations.

CRediT authorship contribution statement

Soumik Das: Conceptualization, Software, Investigation, Writing – original draft; **Punyasha Chatterjee:** Conceptualization, Validation, Supervision, Writing – review & editing; **Agostino Cortesi:** Conceptualization, Formal analysis, Supervision, Writing – review & editing.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability

The implementation code is available at the github repository https://github.com/soumik1065/AIRRUB/tree/main/Implementation

All the code with respect to the recommendation system are available on a GitHub Link: https://github.com/soumik1065/AIRRUB

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Alamouri, A., Lampert, A., & Gerke, M. (2023). Impact of drone regulations on drone use in geospatial applications and research: Focus on visual range conditions, geofencing and privacy considerations. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, *91*(5), 381–389. https://doi.org/10.1007/s41064-023-00246-y

Azari, A., Ghavimi, F., Ozger, M., Jantti, R., & Cavdar, C. (2020). Machine learning assisted handover and resource management for cellular connected drones. In *2020 IEEE 91st vehicular technology conference (VTC2020-spring)* (pp. 1–7). IEEE. https://doi.org/10.1109/VTC2020-Spring48590.2020.9129453

Bag, R., Roy, M., Cortesi, A., & Chaki, N. (2025). Eliciting context-oriented NFR constraints and conflicts in robotic systems. *Innovations in Systems and Software Engineering*, *21*(3), 863–876. https://doi.org/10.1007/s11334-023-00545-y.

Chen, Q., Gao, N., Huang, S., Low, J., Chen, T., Sun, J., & Schwager, M. (2026). Gradnav++: Vision-language model enabled visual drone navigation with gaussian radiance fields and differentiable dynamics. *IEEE Robotics and Automation Letters*, *11*(2), 1418–1425. https://doi.org/10.1109/LRA.2025.3643290

Cofield, J., Siddique, U., & Cao, Y. (2026). Modifly: A scalable end-to-end multi-agent simulation for unmanned aerial vehicles. In A. Vidler, & S. Swarup (Eds.), *Multi-agent-based simulation XXVI* (pp. 128–139). Cham: Springer Nature Switzerland.

Das, S., Deb, N., Chaki, N., & Cortesi, A. (2024). Minimising conflicts among run-time non-functional requirements within devops. *Systems Engineering*, *27*(1), 177–198. https://doi.org/10.1002/sys.21715.

Derrouaoui, S. H., Bouzid, Y., Guiatni, M., & Dib, I. (2022). A comprehensive review on reconfigurable drones: Classification, characteristics, design and control technologies. *Unmanned Systems*, *10*(01), 3–29. https://doi.org/10.1142/S2301385022300013

EASA, Guidelines for the assessment of the critical area of an unmanned aircraft, 2025. https://www.easa.europa.eu/en/downloads/139781/en.

First in Architecture (2025). Average dimensions of person standing, . https://www.firstinarchitecture.co.uk/metric-data-01-average-dimensions-of-person-standing/ .

Fotouhi, A., Qiang, H., Ding, M., Hassan, M., Giordano, L. G., Garcia-Rodriguez, A., & Yuan, J. (2019). Survey on UAV cellular communications: Practical aspects, standardization advancements, regulation, and security challenges. *IEEE Communications Surveys & Tutorials*, *21*(4), 3417–3442. https://doi.org/10.1109/COMST.2019.2906228

Ghamari, M., Rangel, P., Mehrubeoglu, M., Tewolde, G. S., & Sherratt, R. S. (2022). Unmanned aerial vehicle communications for civil applications: A review. *IEEE Access*, *10*, 102492–102531. https://doi.org/10.1109/ACCESS.2022.3208571

GitHub (2025a). Implementation code. https://github.com/soumik1065/AIRRUB/tree/3556d4300fb5e5522d48c0eddadc116105f3d509/Implementation%20Code.

GitHub (2025b). SRS Doc. https://github.com/soumik1065/AIRRUB/tree/5c902fa68891b0eb65d750d5d4a662444c01c949/SRS%20Doc.

GitHub (2025c). Technical report. https://github.com/soumik1065/AIRRUB/tree/main/Technical%20Report.

Hu, Z., Cao, Y., Li, X., Zhang, X., Zhang, C., & Liu, Z. (2025). A real-time UAV delivery system considering dock selection and spatial conflict. *Expert Systems with Applications*, *281*, 127498. https://www.sciencedirect.com/science/article/pii/S0957417425011200. https://doi.org/10.1016/j.eswa.2025.127498

Idries, A., Mohamed, N., Jawhar, I., Mohamed, F., & Al-Jaroodi, J. (2015). Challenges of developing UAV applications: A project management view. In *2015 International conference on industrial engineering and operations management (IEOM)* (pp. 1–10). https://doi.org/10.1109/IEOM.2015.7093730

Jiao, S., Zhang, G., Zhou, M., & Li, G. (2023). A comprehensive review of research hotspots on battery management systems for UAVs. *IEEE Access*, *11*, 84636–84650. https://doi.org/10.1109/ACCESS.2023.3301989

JOUAV (2025). How fast can a drone fly? top speeds of various types. https://www.jouav.com/blog/how-fast-can-a-drone-fly.html#:~:text=These%20drones%20can%20achieve%20speeds,(2%2C173%20km%2Fh).

Kuroki, Y., Young, G. S., & Haupt, S. E. (2010). Uav navigation by an expert system for contaminant mapping with a genetic algorithm. *Expert Systems with Applications*, *37*(6), 4687–4697. https://www.sciencedirect.com/science/article/pii/S0957417409010859. https://doi.org/10.1016/j.eswa.2009.12.039

Kurtanović, Z., & Maalej, W. (2017). Automatically classifying functional and non-functional requirements using supervised machine learning. In *2017 IEEE 25th international requirements engineering conference (RE)* (pp. 490–495). https://doi.org/10.1109/RE.2017.82

Kuru, K., Ansell, D., Khan, W., & Yetgin, H. (2019). Analysis and optimization of unmanned aerial vehicle swarms in logistics: An intelligent delivery platform. *IEEE Access*, *7*, 15804–15831. https://doi.org/10.1109/ACCESS.2019.2892716

Kwak, J., & Sung, Y. (2018). Autonomous UAV flight control for GPS-based navigation. *IEEE Access*, *6*, 37947–37955. https://doi.org/10.1109/ACCESS.2018.2854712

Laghari, A. A., Jumani, A. K., Laghari, R. A., & Nawaz, H. (2023). Unmanned aerial vehicles: A review. *Cognitive Robotics*, *3*, 8–22. https://www.sciencedirect.com/science/article/pii/S2667241322000258. https://doi.org/10.1016/j.cogr.2022.12.004

Leary, S., Deittert, M., & Bookless, J. (2011). Constrained UAV mission planning: A comparison of approaches. In *2011 IEEE International conference on computer vision workshops (ICCV workshops)* (pp. 2002–2009). https://doi.org/10.1109/ICCVW.2011.6130494

Liu, X., Li, G., Yang, H., Zhang, N., Wang, L., & Shao, P. (2023). Agricultural UAV trajectory planning by incorporating multi-mechanism improved grey wolf optimization algorithm. *Expert Systems with Applications*, *233*, 120946. https://www.sciencedirect.com/science/article/pii/S0957417423014483. https://doi.org/10.1016/j.eswa.2023.120946

Matalonga, S., White, S., Hartmann, J., & Riordan, J. (2022). A review of the legal, regulatory and practical aspects needed to unlock autonomous beyond visual line of sight unmanned aircraft systems operations. *Journal of Intelligent & Robotic Systems*, *106*(1), 10. https://doi.org/10.1007/s10846-022-01682-5

Park, J.-W., Oh, H.-D., & Tahk, M.-J. (2008). Uav collision avoidance based on geometric approach. In *2008 SICE Annual conference* (pp. 2122–2126). https://doi.org/10.1109/SICE.2008.4655013

Patrik, A., Utama, G., Gunawan, A. A. S., Chowanda, A., Suroso, J. S., Shofiyanti, R., & Budiharto, W. (2019). Gnss-based navigation systems of autonomous drone for delivering items. *SpringerOpen-Journal of Big Data*, *6*, 1–14. https://doi.org/10.1186/s40537-019-0214-3

Primatesta, S., Rizzo, A., & la Cour-Harbo, A. (2020a). Ground risk map for unmanned aircraft in urban environments. *Journal of Intelligent & Robotic Systems*, *97*(3), 489–509. https://doi.org/10.13140/RG.2.2.34331.77602

Primatesta, S., Rizzo, A., & Cour-Harbo, A. l. (2020b). Ground risk map for unmanned aircraft in urban environments. *The Journal of Intelligent and Robotic Systems*, *97*(3), 489–509. https://doi.org/10.1007/S10846-019-01015-Z

Roy, M., Deb, N., Cortesi, A., Chaki, R., & Chaki, N. (2021). Nfr-aware prioritization of software requirements. *Systems Engineering*, *24*(3), 158–176. https://doi.org/10.1002/sys.21572

Salmi, A., Guiatni, M., Bouzid, Y., Derrouaoui, S. H., & Boudjema, F. (2024). Fault tolerant control based on thau observer of a reconfigurable quadrotor with total loss of actuator. *Unmanned Systems*, *12*(04), 667–683. https://doi.org/10.1142/S2301385024500146

Sorbelli, F. B., Chatterjee, P., Coro, F., Palazzetti, L., & Pinotti, C. M. (2023). A novel multi-layer framework for BVLos drone operation: A preliminary study. In *IEEE infocom 2023 - IEEE conference on computer communications workshops (infocom wkshps)* (pp. 1–6). https://doi.org/10.1109/INFOCOMWKSHPS57453.2023.10225806

Sorbelli, F. B., Chatterjee, P., Corò, F., Ghobadi, S., Palazzetti, L., & Pinotti, C. M. (2024a). A novel graph-based multi-layer framework for managing drone BVLos operations. *IEEE Transactions on Network and Service Management*, (pp. 1–1). https://doi.org/10.1109/TNSM.2024.3401175

Sorbelli, F. B., Chatterjee, P., Corò, F., Ghobadi, S., & Pinotti, C. M. (2024b). Scheduling of multiple UAVs in BVLos operations along unidirectional and bidirectional paths. In *2024 IEEE 49th conference on local computer networks (LCN)* (pp. 1–7). https://doi.org/10.1109/LCN60385.2024.10639622

Sorbelli, F. B., Chatterjee, P., Das, P., & Pinotti, C. M. (2024c). Risk assessment in BVLos operations for UAVs: Challenges and solutions. In *2024 20th international conference on distributed computing in smart systems and the internet of things (DCOSS-iot)* (pp. 300–307). https://doi.org/10.1109/DCOSS-IoT61029.2024.00053

T-DRONES (2025a). Drones fly in different countries, . https://www.t-drones.com/blog/how-high-can-a-drone-fly.html#:~:text=While%20a%20common%20altitude%20limit,prevent%20conflicts%20with%20manned%20aircraft.

T-DRONES (2025b). Unlock the secrets: How far can your drone fly, . https://www.t-drones.com/blog/how-far-can-a-drone-fly.html.

Wang, W., & Qian, H. (2026). A hierarchical ORCA framework for multi-UAV navigation in unstructured environments with velocity optimization and local minima avoidance. *Expert Systems with Applications*, *296*, 129205. https://www.sciencedirect.com/science/article/pii/S0957417425028210. https://doi.org/10.1016/j.eswa.2025.129205

Yang, Z., Luo, D., Guo, Z., & Xu, Y. (2025). Adaptive predefined-time path-following control for fixed-wing UAV formations with prescribed performance. *Applied mathematical modelling*, *147*, 116173. https://www.sciencedirect.com/science/article/pii/S0307904X25002483. https://doi.org/10.1016/j.apm.2025.116173

Zhang, C., Zhou, W., Qin, W., & Tang, W. (2023). A novel UAV path planning approach: Heuristic crossing search and rescue optimization algorithm. *Expert Systems with Applications*, *215*, 119243. https://www.sciencedirect.com/science/article/pii/S0957417422022618. https://doi.org/10.1016/j.eswa.2022.119243