BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2026
http://excel.fit.vutbr.cz

User Interface for Drone Missions: Automated 3D Planning

Václav Sovák

Abstract
Manual drone flights for building inspections are incredibly stressful, prone to human error, and current planning tools are often cluttered 2D nightmares. To solve this, I developed a 3D extension for the DroCo system in Unity that allows operators to interact directly with real-world map data and automatically generates collision-free scan routes. The system calculates optimal camera overlaps and replaces tedious waypoint clicking with intuitive 3D spatial editing tools. Planning time rapidly decreases, thus reducing pilot cognitive load and guaranteeing high-quality, consistent data collection for precise photogrammetry.

*xsovakv00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

[Motivation] Imagine flying a drone worth thousands of dollars inches away from a building to capture a perfect 3D model. As a pilot, you probably already know how much data there is to look after. It is a cognitive nightmare. Research indicates that high operator workload significantly degrades flight safety and mission effectiveness [1]. Automation is clearly the answer, but the current software market is incredibly frustrating. Professional planners often look like outdated CAD systems packed with confusing spreadsheets and endless menus. Worse, many tools force pilots to plan complex building inspections on flat 2D maps, relying on pure guesswork for vertical height. Even when 3D environments are supported, the buildings are usually just passive graphics, leaving the pilot to manually calculate distances and overlaps. Fighting with an incomprehensible, limiting interface before you even take off completely defeats the purpose of automation.

[Problem definition] For a proper photogrammetric output, the drone must maintain a precise distance from the facade and a consistent image overlap. These strict parameters are essential for successful 3D reconstruction and minimizing artifacts in photogrammetric software [2]. Manual flying practically guarantees human error, leading to inconsistent data or crashed drones. The problem is how to automate this trajectory generation over 3D structures without trapping the user in an unintuitive, spreadsheet-like UI.

[Existing solutions] The drone software market is crowded and complicated. Professional tools like *UgCS* are incredibly powerful, but their UI rather resembles a CAD system, intimidating and cluttered with modal menus. Others, like *Litchi*, are very user-friendly but primarily 2D focused, forcing the pilot to guess building heights blindly. Existing 3D tools treat buildings as passive obstacles rather than interactive targets, which is a common limitation in current structural inspection path planning [3].

[Our solution] I designed and implemented an interactive and user-friendly 3D mission planner built in the Unity engine as an extension of the already existing *DroCo* system [4]. It transforms *OSM* data from a mere list of GPS coordinates into an interactive anchor. You click the building, and the app calculates the rest.

[Contributions] My work completely eliminates the need for manual waypoint-by-waypoint construction. It features:

- A clean, dark mode interactive User Interface for mission planning.
- Auto-generation of horizontal and vertical inspection paths with optimal overlap.
- Intuitive mission editor with 3D gizmos for rapid visual adjustments.
- Executing autonomous missions on supported drones.

2. The Inspection Route

The poster visually guides you through the exact workflow a pilot experiences when using the application. Let's walk through it.

First, the operator navigates through the 3D map environment, which is populated with real-world buildings as a mesh. As shown in Fig 2, the target building can be interactively selected. The system recognizes its boundaries and dimensions. The process of building reconstruction before being selected can be seen in Fig 1.

Instead of clicking waypoints into the world scene, the operator simply confirms the target, and the algorithm takes over (Fig 3). It automatically wraps a 3D trajectory around the building, calculating the exact segments and vertical steps required to achieve the desired camera overlap for optimal photogrammetry.

However, no algorithm is perfect for every unpredictable real-world scenario. That is why the app is not a dead end here. As seen in Fig 4, the operator retains full control. Using intuitive 3D gizmos directly in the scene, the user can grab entire columns or walls of waypoints and shift them dynamically to avoid an unmapped tree or chimney. The UI panels on the right side provide instant feedback on mission parameters.

Once the route is verified by the user, the operator connects the drone via the local DroCo server (Fig 5). This local architecture is crucial because internet coverage near remote inspection sites is often unstable. The operator gets a clear overview of telemetry, signal strength, and potential warnings (wind, GPS loss) before takeoff.

Finally, the operator hits start, and the drone autonomously executes the mission (Fig 6). The UI minimizes distractions, hiding unnecessary tools and displaying a live camera stream, allowing the pilot to simply supervise the safe collection of perfect data.

3. Design and UX

The core philosophy was *Zero Cognitive Overload*. During the prototyping phase, user testing revealed that putting too many menus or unknown icons on the screen caused confusion. This aligns with cognitive load theory in user interface design, which emphasizes minimizing unnecessary visual elements [5].

Therefore, the final UI relies on a cleaner canvas approach. The left panel houses only fast and immediate actions (like emergency stop or camera controls), while the right panel handles complex settings within collapsible sections. To prevent user errors during route editing, an Undo stack (Ctrl+Z) was implemented, following established usability heuristics for user control and error recovery [6].

4. Implementation and Testing

The application is built within the Unity engine, leveraging its rendering pipeline to visualize real-world building meshes generated from OpenStreetMap data. The user interface was developed using Unity's native UI system to ensure responsive scaling across different tablet devices, which are commonly used in the field by drone operators. Communication with the underlying DroCo server is handled asynchronously to prevent any UI freezing during critical flight operations.

Before deploying to real hardware, the system's trajectory generation algorithms were iteratively tested using a custom simulation script. This script simulated the drone's flight path around the virtual model of the building, allowing for safe verification of the generated routes, distance constraints, and camera overlaps without risking actual drone crashes. Following this virtual verification, the system was successfully validated during real-world testing with a physical drone. Preliminary user feedback indicates that the interactive 3D visual approach significantly reduces the time required to plan a mission compared to traditional 2D tools. To formally measure this UX improvement, future tests will be evaluated using standardized quantitative metrics, such as the System Usability Scale (SUS) [7].

5. Conclusions

Planning an inspection flight does not have to feel like rocket science. By leveraging the power of Unity and treating 3D buildings as interactive objects, this work bridges the gap between complex photogrammetry math and user-friendly design.

The resulting tool reduces mission preparation time, minimizes the operator expertise required, and ensures the collected data is optimal for 3D reconstruction. In the future, this application could be expanded to support multiple drones dividing a single mission to save time, and to integrate automatic detection of other subtle obstacles, such as power lines.

Acknowledgements

I would like to thank my supervisor Ing. Daniel Bambušek for his invaluable feedback and guidance throughout the development of this user interface and the underlying application logic.

References

[1] Mary L Cummings and Paul J Mitchell. Predicting controller capacity in supervisory control of multiple uavs. *IEEE Transactions on Systems, Man, and*

Cybernetics-Part A: Systems and Humans, 38(2):451–460, 2008.

[2] Francesco Nex and Fabio Remondino. Uav for 3d mapping applications: a review. *Applied geomatics*, 6(1):1–15, 2014.

[3] Manh Duong Phung, Cong Hoang Quach, Thien Hoang Dinh, and Quang Phan Ha. Structure-aware coverage path planning for uavs. *IFAC-PapersOnLine*, 50(1):11646–11651, 2017.

[4] Daniel Bambušek et al. Droco: Drone co-pilot system for teleoperation and navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2020.

[5] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.

[6] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

[7] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.