Project Proposal

Visual-Inertial Navigation for Autonomous UAV Operation in GPS-Restricted Environments

Project Supervisors - Internal:

Prof. Chandana Gamage

Group Members:

D. V. A. I. Delgahagoda	210113L
T. T. Kashmeera	210279A
K. D. R. P. Rathnayake	210537N

Department of Computer Science and Engineering

University of Moratuwa

Table of Content

Table of Figures

1 Introduction

1.1 Overview of UAVs and Their Applications

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, have significantly transformed various sectors, ranging from agriculture to logistics [1]. Their versatility and capacity to access remote or hazardous areas have positioned them as invaluable tools across numerous applications [2]. These applications encompass aerial mapping, surveillance, and inspection, as well as autonomous flight operations, facilitating data collection in environments otherwise inaccessible to human operators. The expanding utility of UAVs in diverse and complex settings underscores the increasing demand for advanced navigational capabilities.

1.2 The Critical Role of Navigation Systems

Autonomous UAV operation is fundamentally dependent on accurate and robust navigation and localization capabilities. Precise estimations of a UAV's position and orientation are indispensable for executing critical tasks such as obstacle avoidance, sophisticated path planning, and maintaining flight stability throughout a mission. Without reliable navigation, the potential of autonomous drones remains severely constrained, limiting their deployment to less complex, open environments [3].

1.3 Challenges in GPS-Restricted Environments

While the Global Positioning System (GPS) serves as a primary navigation tool for UAVs in many scenarios, its reliability diminishes significantly or becomes entirely unavailable in certain challenging environments. These challenging areas include dense urban landscapes, often termed "urban canyons," indoor facilities, under-canopy forest environments, and remote regions characterized by obstructed satellite visibility [4], [5]. The expansion of UAV applications into these previously inaccessible or difficult environments directly highlights the limitations of GPS, making the development of alternative or complementary navigation solutions imperative. This evolution of UAV capabilities necessitates more sophisticated and robust navigation systems that are less reliant on external signals and can leverage onboard sensor data.

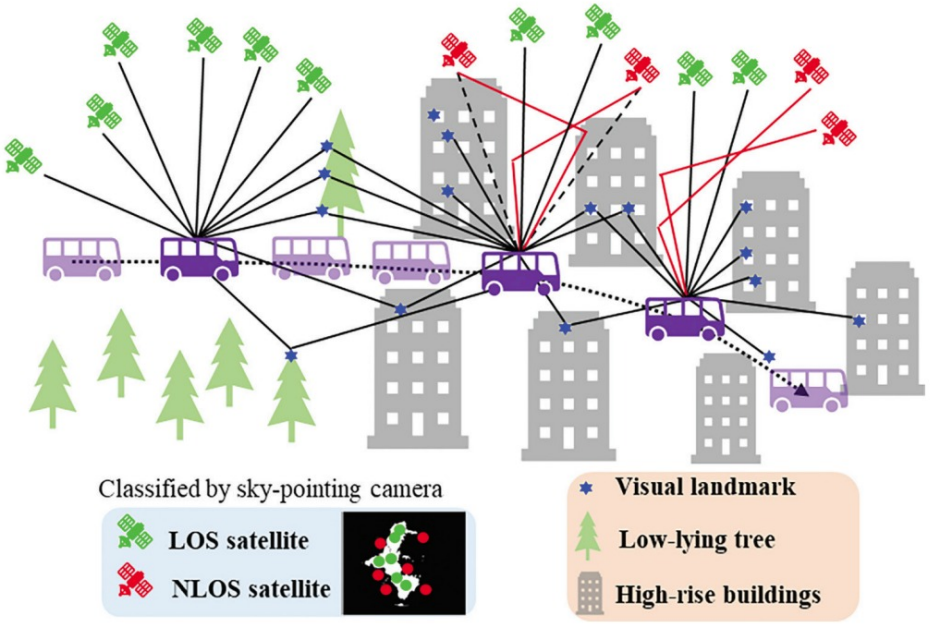


Figure 1: GNSS signal classification with LOS/NLOS satellites and visual landmarks in urban environments [6].

1.4 Introduction to Visual-Inertial Navigation Systems (VINS)

Visual-Inertial Navigation Systems (VINS) have emerged as a potent solution to overcome the limitations inherent in GPS-dependent navigation. VINS integrates visual information captured by onboard cameras with motion data provided by Inertial Measurement Units (IMUs). This fusion allows VINS to leverage computer vision algorithms for feature extraction and tracking, combined with advanced sensor fusion techniques, to accurately estimate the UAV's position, velocity, and orientation, even in the complete absence of GPS signals. The ability of VINS to provide precise, real-time pose estimation in challenging conditions positions it as a foundational technology for next-generation UAV autonomy. This means VINS is crucial for enabling UAVs to perform complex tasks, avoid obstacles, and maintain safety in dynamic and unstructured environments, pushing the boundaries of what autonomous drones can achieve [7], [8].

1.5 Purpose and Scope of the Project

This project proposes the implementation of a robust VINS specifically engineered for autonomous UAV operation in GPS-restricted environments. The primary aim is to significantly enhance navigation accuracy, reliability, and safety in such challenging scenarios. The project's scope encompasses a comprehensive analysis of existing VINS technologies, localization technologies, the identification of key challenges, and the subsequent

development of a methodology that incorporates advanced techniques to overcome these identified limitations.

2 Problem Statement

2.1 Limitations of GPS in Challenging Environments

The efficacy of Global Navigation Satellite Systems (GNSS), particularly GPS, is severely hampered in environments that present physical obstructions or signal interference. GPS signals are inherently weak or entirely absent in indoor spaces, underground areas, and remote locations, rendering traditional satellite-based navigation ineffective.

In dense urban environments, commonly referred to as urban canyons, tall buildings obstruct the direct line-of-sight (LoS) to satellites. This leads to pervasive signal blockages and severe multipath propagation, where signals bounce off structures, distorting their accuracy. Such conditions cause a drastic degradation in positioning accuracy, potentially dropping from several meters to hundreds of meters [4], [5]. Similarly, under-canopy forest environments present dense obstacles and significant GNSS signal interference, making autonomous navigation for data collection exceptionally challenging.

Beyond direct signal issues, these environments often exhibit varying lighting conditions, low texture, or the presence of dynamic objects. These factors further complicate visual navigation components that might otherwise compensate for GPS loss [9]. The problem is not merely a singular environmental factor, but a synergistic combination of factors. For instance, urban canyons induce both signal blockage and multipath effects, while simultaneously presenting visually complex scenes with dynamic elements. This multi-faceted degradation creates a compounding effect on navigation challenges, necessitating solutions that address multiple, interconnected environmental challenges simultaneously.

2.2 Impact on Autonomous UAV Operations

The limitations of GPS in challenging environments have profound implications for autonomous UAV operations. The primary challenge lies in maintaining precise positioning and navigation capabilities when satellite-based systems become unreliable or completely unavailable. Inaccurate localization leads to significant deviations from planned flight paths, as evidenced by experiments where drones deviated by 1 to 5 meters or more in urban settings [10].

Such navigation disruptions affect both autonomous and manual flight, particularly during critical phases like landing and obstacle avoidance. In loosely coupled navigation

systems, a complete data outage can occur when fewer than four satellites are visible, leading to a loss of positional awareness. Unreliable navigation consequently jeopardizes the safety of UAV missions, especially in complex or dynamic environments where collision avoidance and precise task execution are paramount.

This inability to operate reliably in GPS-restricted areas severely constrains the potential applications and mission capabilities of autonomous UAVs. This highlights a "last mile" problem for UAV autonomy. While long-range navigation in open areas might be feasible with GPS, the most complex and often most valuable operations, such as precise delivery in a city, detailed indoor inspection, or surveying under dense foliage, occur precisely where GPS is least reliable. Solving this "last mile" navigation problem with robust VINS is crucial for unlocking the full potential and economic value of autonomous UAVs in high-impact, real-world scenarios, moving beyond simple line-of-sight operations.

2.3 Summary

GPS-based navigation becomes unreliable or completely unavailable in environments with signal obstructions such as urban canyons, forests, and indoor areas. In these settings, satellite signals suffer from blockage, multipath propagation, and interference, leading to large localization errors. This severely limits UAV autonomy, especially in critical operations like landing, obstacle avoidance, or precise deliveries. While inertial and vision-based navigation systems (VINS) can compensate for GPS loss, they face their own challenges—IMU drift over time, difficulty tracking features in low-texture or dynamic scenes, scale ambiguity during initialization, high computational demands, and degraded performance in poor visual conditions (e.g., low light or motion blur). Together, these issues make reliable real-time navigation in GPS-denied environments difficult to achieve. Based on the problems described previously, the research problem of this project can be succinctly stated as: "Developing a robust, efficient Autonomous UAV that uses Visual-Inertial Navigation System (VINS) for localization and navigation in GPS-denied and visually degraded environments."

3 Research Objectives

Visual-Inertial Navigation Systems (VINS) offer a powerful framework for enabling UAV autonomy in GPS-restricted environments. However, their practical deployment is challenged by several limitations that arise from sensor imperfections, environmental conditions, and computational constraints. To ensure reliable navigation and robust performance, it is essential to explicitly address these key issues. The main problems this

research seeks to overcome are outlined below:

3.1 Inertial-Only IMU Drift

Inertial-only navigation (using only IMU data) accumulates sensor errors over time, causing the estimated trajectory to drift unboundedly [10]. In practice, biases and noise integrate into large position/attitude errors if not corrected by external references. Without GPS or vision updates, this long-term drift makes pure INS unreliable for GPS-denied flight.

Figure 2: Visualization of IMU drift in UAVs [11].

3.2 Visual Tracking in Low-Texture/Dynamic Scenes

Vision-based navigation relies on tracking environmental features. In low-texture or uniform areas (e.g., blank walls or repetitive patterns), feature descriptors lack distinctiveness and matching often fails [12]. Likewise, highly dynamic scenes with many moving objects leave few stable landmarks, causing the visual tracker (and thus VINS) to lose lock or fail completely [12].

3.3 VINS Initialization and Scale Ambiguity

Monocular VINS cannot infer absolute scale from vision alone, so the initialized map's scale is ambiguous [13]. Proper initialization also requires rich motion (translations/rotations) to excite all degrees of freedom; for example, VINS-Mono must execute full 3-axis maneuvers to observe gravity and scale, which is time-consuming and resource-intensive [10]. If these conditions are not met, the estimator may diverge or converge to a wrong scale.

3.4 Real-Time VINS Computational Demands

Tightly-coupled VINS algorithms fuse high-rate IMU and image data via nonlinear optimization or filtering, which is computationally expensive. For instance, VINS-Mono's sliding-window bundle-adjustment ran at only $\sim$9 Hz on typical UAV hardware, failing to meet real-time requirements on a low-cost platform [10]. In general, ensuring real-time performance requires significant CPU/GPU resources, which may be beyond small UAV capabilities.

3.5 Compound Environmental Degradation (Poor Lighting, Motion Blur, Dynamic Objects)

In severely degraded conditions combining multiple factors, VINS performance collapses. Extensive motion blur (from fast motion or low light) prevents reliable feature tracking [14], and scenes dominated by dynamic objects (e.g., >80% moving targets) provide too few static cues, leading to localization failure [13]. Together, these adverse effects can overwhelm the system, causing the estimator to lose track and diverge.

3.6 Summary

Listed below is a summary of the key project objectives.

- Minimize IMU drift through effective fusion of visual and inertial data to maintain stable localization without GPS.
- Enhance visual tracking robustness in low-texture and dynamic environments to ensure continuous navigation.
- Address scale ambiguity and improve VINS initialization for more accurate trajectory estimation.
- Optimize VINS computational efficiency to achieve real-time performance on resource-constrained UAV platforms.
- Improve system resilience under compound environmental degradations such as poor lighting, motion blur, and high scene dynamics.

4 Literature Review

4.1 The Challenge of Autonomous Navigation in GPS-Restricted Environments

4.1.1 The Imperative for GPS-Independent Navigation

The proliferation of Unmanned Aerial Vehicles (UAVs) has catalyzed transformative advancements across a multitude of sectors, including surveillance, reconnaissance, infrastructure inspection, and search-and-rescue operations.[15] The operational envelope of these autonomous systems is increasingly expanding into complex and unstructured environments where the reliance on traditional navigation aids, primarily the Global Navigation Satellite System (GNSS), is untenable. Environments such as dense urban canyons, subterranean spaces, cluttered indoor settings, and thick forests are characterized as GPS-restricted or GPS-denied.[16] Within these domains, GNSS signals are frequently attenuated, blocked, or corrupted by multipath effects, where signals reflect off surfaces before reaching the receiver, leading to significant positional errors.[16] This inherent unreliability is further compounded by the growing threat of deliberate signal interference. Malicious actors can employ jamming techniques to overwhelm GNSS receivers with noise, or more insidiously, spoofing techniques to broadcast false signals, deceiving the UAV about its true location.[16] The strategic and commercial implications of this vulnerability are significant; recent reports indicate a sharp rise in GPS jamming and spoofing incidents, with as many as 700 events occurring globally each day, particularly in conflict zones and across North America and Europe.[17] The operational integrity of a UAV hinges on its ability to perform high-accuracy, continuous state estimation—the process of determining its position, velocity, and orientation (attitude) in real-time. The potential for GNSS failure, whether environmental or adversarial, renders it an insufficient standalone solution for robust autonomous flight, creating a critical demand for alternative navigation technologies.[18]

4.1.2 Visual-Inertial Navigation Systems (VINS) as a Premier Solution

Among the suite of alternative navigation technologies, Visual-Inertial Navigation Systems (VINS) have emerged as a premier solution, offering a compelling balance of performance, cost, and physical footprint. VINS achieves robust state estimation by fusing data from two complementary, low-cost, and lightweight sensors: a visual camera and an Inertial Measurement Unit (IMU).[19] These sensors are ubiquitous on modern robotic platforms and possess symbiotic characteristics. The camera, a passive sensor, captures

rich information about the surrounding environment, including visual features such as color and texture.[16] Using computer vision algorithms, a VINS can identify and track these features across consecutive images to infer its own motion and, in many cases, build a map of its surroundings—a process known as Visual SLAM (VSLAM).1 However, vision-only systems are susceptible to failure in visually degraded conditions, such as low light, textureless environments (e.g., white walls), or during rapid motion that induces significant image blur.[20] The IMU, an active sensor, provides high-frequency measurements of the UAV's linear acceleration and angular velocity.[19] These measurements are self-contained and immune to external environmental conditions, allowing the system to continue estimating its motion even when visual information is unavailable.[20] The fundamental limitation of an IMU, particularly the low-cost Micro-Electro-Mechanical Systems (MEMS) type used in UAVs, is that its measurements are corrupted by noise and biases. When these measurements are integrated over time to compute velocity and position, the errors accumulate rapidly, leading to unbounded drift.[21] VINS masterfully exploits the complementary nature of these two sensors.[20] The high-frequency inertial data from the IMU effectively bridges the gaps between camera frames and provides robustness against visual degradation.[22] Concurrently, the visual data, which provides drift-free measurements relative to the static environment, is used to continuously correct for the IMU's accumulating drift.8 This tight integration allows for the real-time estimation of the vehicle's full six-Degrees-of-Freedom (6-DOF) state (position and orientation), which is the core problem that VINS aims to solve.[18]

4.1.3 Scope and Structure of the Review

This literature review provides a comprehensive analysis of Visual-Inertial Navigation Systems tailored for autonomous UAV operation in GPS-restricted environments. The review focuses primarily on the fundamental principles underpinning VINS and the seminal algorithms that have defined the state-of-the-art. Section 2 deconstructs the foundational sensor components, the IMU and the camera, detailing their operating principles and critical error models. Section 3 explores the core architectural paradigms that govern VINS design, comparing data fusion strategies (loosely- vs. tightly-coupled) and back-end state estimation methodologies (filtering vs. optimization). Section 4 presents a detailed review of four prominent and widely-used VINS algorithms: MSCKF, OKVIS, VINS-Mono, and ORB-SLAM3, analyzing their key innovations, strengths, and limitations. Finally, Section 5 discusses the unique challenges of implementing VINS on UAV platforms and examines emerging trends and future research directions that are shaping the next generation of autonomous navigation systems.

4.2 Foundational Components of Visual-Inertial Systems

A deep understanding of VINS requires a thorough analysis of its constituent sensors. The performance of any VINS algorithm is fundamentally limited by the quality of the data it receives and the fidelity of the models used to interpret that data. This section details the working principles and, critically, the error characteristics of the IMU and the visual camera, which are essential for designing high-performance estimation algorithms.

4.2.1 The Inertial Measurement Unit (IMU): Principles and Error Modeling

Working Principles of MEMS IMUs An Inertial Measurement Unit is an electromechanical device that measures a body's specific force and angular rate using a combination of accelerometers and gyroscopes.[21] For UAV applications, MEMS-based IMUs are the standard choice due to their exceptional advantages in size, weight, power, and cost (SWaP-C).[21]

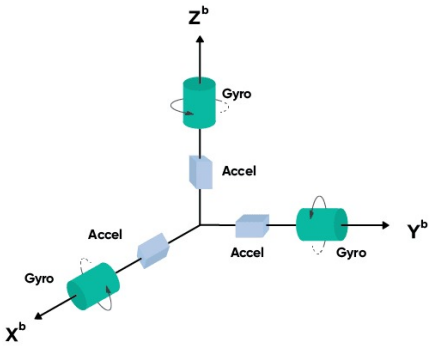


Figure 3: Image depicting the accelerometers and gyroscopes in the three axes of movement. Each accelerometer and gyroscope is positioned at 90° to the others (orthogonally). Accelerometers measure motion along each axis and each gyroscope measures angular velocity around each axis.[21]

A typical MEMS accelerometer operates on the principle of a proof mass suspended by springs within a reference frame. When the unit accelerates, the inertia of the proof mass causes it to displace relative to the frame. This displacement is measured, often by detecting the change in capacitance between the proof mass and a set of fixed electrodes, and is directly proportional to the applied linear acceleration.[21] A MEMS gyroscope commonly utilizes the Coriolis effect to measure angular velocity. A proof mass is driven to resonate along a specific axis. When the entire unit is subjected to rotation, the Coriolis force induces a secondary vibration in the proof mass along an axis perpendicular to both

the drive axis and the axis of rotation. The magnitude of this secondary vibration, also measured capacitively, is proportional to the angular rate of the gyroscope.[21]

The IMU Measurement Model and Error Formulation The raw outputs from a low-cost MEMS IMU are not a direct measure of the true kinematics; they are corrupted by a combination of deterministic and stochastic errors that must be accurately modeled to prevent rapid degradation of the navigation solution.[23] The standard measurement model for a gyroscope and an accelerometer includes several key error terms:

- **Bias (b_g, b_a):** This represents a persistent offset in the sensor output. It is typically modeled as the sum of a constant turn-on bias and a slowly time-varying component known as bias instability or in-run bias. This time-varying component is crucial to estimate online and is often modeled as a random walk.[21]
- **Scale Factor Error (s_g, s_a):** This is a multiplicative error representing a deviation in the sensor's sensitivity. For a 3-axis sensor, this can be a full matrix including cross-axis sensitivity terms, which measure how an input on one axis affects the output on another.[21],[23]
- **Noise (n_g, n_a):**This is a high-frequency, zero-mean random fluctuation in the sensor output. It is typically modeled as additive white Gaussian noise, characterized by its noise density. When integrated, this noise leads to an error that grows with the square root of time, known as Angle Random Walk (ARW) for gyroscopes and Velocity Random Walk (VRW) for accelerometers.[21]

Accurate VINS performance relies on the online estimation of these error terms, particularly the biases, which are included as part of the state vector in the filter or optimizer.[24]

IMU Pre-integration A critical innovation for modern tightly-coupled VINS is IMU pre-integration. IMUs typically provide measurements at a much higher frequency (e.g., 200–1000 Hz) than cameras (e.g., 20–30 Hz).[20] Including a state for every IMU measurement in an optimization-based framework would be computationally prohibitive. IMU pre-integration elegantly solves this problem by analytically combining all the inertial measurements between two consecutive camera keyframes into a single relative motion constraint. This process integrates the IMU measurements in the local body frame. A crucial aspect of this formulation is that these pre-integrated terms are expressed relative to the state at the first keyframe and are only dependent on the IMU biases, not the absolute orientation, velocity, or position. This means that if the optimization algorithm

updates its estimate of the IMU biases, the pre-integration constraint can be efficiently re-evaluated using first-order corrections without needing to re-propagate all the raw IMU measurements from scratch. This makes tightly-coupled, optimization-based VINS computationally feasible.

4.2.2 The Visual Sensor: Monocular vs. Stereo Configurations

The choice of camera configuration is a fundamental design decision in VINS, with significant trade-offs between cost, complexity, and performance.[25]

Monocular VINS A monocular VINS uses a single camera. The primary advantages of this configuration are its minimal hardware cost, low weight and power consumption, and simplicity, making it an ideal choice for small, payload-constrained UAVs.[26] The fundamental limitation of a monocular camera is its inherent inability to observe absolute scale from a single 2D image. It can only determine the 3D structure of the environment and its own trajectory up to an unknown scale factor.[27] This is a geometric reality: a single projection provides information about the bearing of a point in space, but not its distance.[26] The fusion with an IMU is what elevates a monocular system from a relative, scale-less estimator to a full, metric state estimator. The IMU's accelerometer provides measurements of specific force (including gravity) in absolute metric units (m/s^2). By tightly coupling these metric inertial measurements with the scaled visual geometry from the camera, the VINS estimator can observe and resolve the absolute scale of the trajectory and the map.[26] This observability, however, is not instantaneous. It requires the UAV to undergo sufficient acceleration during an initialization phase for the scale to become well-constrained.[26]

Stereo VINS A stereo VINS employs a pair of cameras with a fixed, known baseline. The primary advantage of this setup is its ability to perceive depth directly and instantaneously. By identifying the same feature in both the left and right images, its 3D position can be calculated through triangulation.[25] This provides an absolute scale for the visual measurements from the very first frame, without requiring any specific motion for initialization.[27] This leads to generally more robust and accurate state estimation.[25] The main disadvantages are the increased hardware cost, size, weight, and complexity.[25] A stereo rig requires precise extrinsic calibration to know the exact transformation between the two cameras, and the system must bear the computational load of processing two image streams.[25] Furthermore, the effectiveness of stereo vision is range-dependent. As the distance to observed features becomes very large relative to the camera baseline, the stereo system's depth estimation capability degrades, and it effectively degenerates to the

monocular case.[27]

4.3 Core VINS Architectures: Fusion and Estimation Strategies

The design of a VINS algorithm is defined by two fundamental architectural choices: the method used to fuse data from the visual and inertial sensors, and the mathematical framework used to estimate the state. These choices determine the system's accuracy, robustness, and computational profile.

4.3.1 Data Fusion Paradigms: Loosely-Coupled vs. Tightly-Coupled Approaches

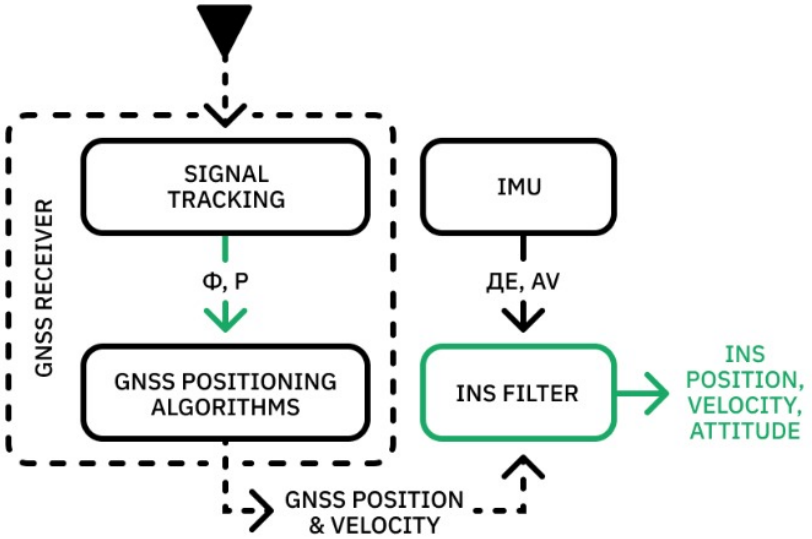


Figure 4: Loosely-coupled architecture which uses an IMU and a GNSS receiver. Similarly, instead of a GNSS receiver, a VINS system can use a visual odometry system.[28]

Loosely-Coupled VINS In a loosely-coupled architecture, the visual and inertial components function as independent estimation modules that are fused at a high level.[28] The typical data flow involves a visual odometry or VSLAM system producing a 6-DOF pose estimate, while the IMU measurements are integrated to produce a separate pose estimate. A final fusion filter then combines these two independent pose solutions to generate the final state estimate.[28] The primary advantage of this approach is its modularity and simplicity.[28] However, this simplicity comes at the cost of performance. Loosely-coupled fusion is inherently sub-optimal because it discards the rich statistical correlations that exist between the raw visual features and the raw inertial measurements.[29] If the vision system fails temporarily, it provides no output to the fusion filter, forcing the system to rely solely on the drifting IMU integration.[28]

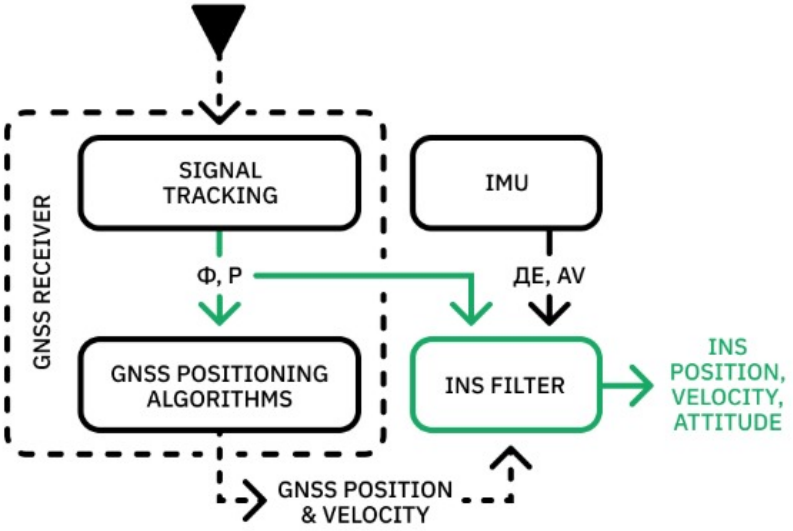


Figure 5: Tightly-coupled architecture which uses an IMU and a GNSS receiver. The GNSS system can be replaced with a visual odometry system.[28]

Tightly-Coupled VINS In contrast, a tightly-coupled architecture is the dominant paradigm in modern, high-performance VINS.[29] This approach establishes a single, unified estimation framework that directly fuses the raw measurements from both sensors.[28] The state estimator, whether a filter or an optimizer, processes visual feature measurements and raw IMU measurements simultaneously within a joint probabilistic model.[30] The principal benefit of tight coupling is superior accuracy and robustness. By considering all raw measurements in a single estimation problem, the framework can fully exploit the statistical correlations between them, leading to a more constrained and optimal solution.[29] The ability to leverage partial information is a key differentiator; even if only a few features are tracked, a tightly-coupled system can use that limited information to mitigate drift, whereas a loosely-coupled system might declare a total tracking failure.[28] The main drawback of this approach is its significantly higher implementation complexity.[28]

4.3.2 Back-End State Estimation: Filtering vs. Optimization

The "back-end" of a VINS is the core computational engine that performs the state estimation. The two primary approaches are filtering and optimization (also known as smoothing).[31]

Filtering-Based VINS Filtering-based methods recursively estimate the current state of the system by incorporating measurements as they arrive. The most prevalent method in this category is the Extended Kalman Filter (EKF).[32] The EKF operates in a continuous two-step cycle:

1. **Prediction (or Propagation):** The system's dynamic model, driven by the IMU measurements, is used to propagate the state estimate and its associated uncertainty forward in time.[32]

2. **Update (or Correction):** When a new measurement from the camera becomes available, the filter compares the actual measurement with a predicted measurement. The difference, or "innovation," is used to correct the state estimate and reduce its uncertainty.[32] .

The primary advantages of filtering-based VINS are their computational efficiency and low memory footprint, making them well-suited for resource-constrained UAVs.[31] However, their main drawback lies in the linearization required by the EKF, which can introduce significant errors and lead to filter inconsistency, where the filter becomes erroneously overconfident in an inaccurate state estimate.[33]

Optimization-Based VINS Optimization-based methods, also known as smoothing methods, have become the standard for high-accuracy VINS. They formulate the state estimation problem as a large-scale nonlinear least-squares minimization. The objective is to find the trajectory and map structure that best explain all available sensor measurements over a period of time by minimizing a cost function. This joint optimization is a generalization of the Bundle Adjustment (BA) technique from computer vision.[34]

The cost function is typically a sum of weighted squared errors from visual and inertial sources.[30] Since optimizing over the entire trajectory is computationally infeasible, these methods employ a sliding window approach, restricting the optimization to a bounded window of the most recent states and measurements.[30] Old states are marginalized, summarizing their information as a probabilistic prior on the remaining states to maintain fixed computational complexity.[30] The principal advantage of optimization-based methods is their superior accuracy. By considering a batch of measurements simultaneously, they can iteratively re-linearize the problem, which significantly mitigates the single-point linearization errors that plague the EKF.[31]

4.4 A Review of Prominent VINS Algorithms

The landscape of VINS solutions is rich and varied, having evolved from purely geometric, filter-based methods to sophisticated optimization and deep learning-based systems.

4.4.1 Filter-Based Methods (e.g., MSCKF, OKVIS)

These methods utilize a probabilistic filtering framework, most commonly the Extended Kalman Filter (EKF), to maintain and propagate a probabilistic estimate of the UAV's

state. The core idea of the seminal Multi-State Constraint Kalman Filter (MSCKF) is to avoid adding 3D feature points directly into the state vector, which would make it grow unboundedly. Instead, it maintains a history of recent camera poses in the state vector. When a feature is observed across multiple of these camera poses, it imposes a geometric constraint on them, which is then used in the filter's update step to correct the state and limit IMU drift [35].

- **Advantages:** Their primary strength is computational efficiency. By keeping the state size small and marginalizing old states, they operate with a consistent and low computational footprint, making them highly suitable for platforms with limited processing power [36].
- **Disadvantages:** This efficiency comes at the cost of accuracy. The EKF relies on a first-order linearization of non-linear system dynamics, which can introduce significant errors, especially during fast rotations. The act of marginalizing past states also prevents re-linearization using newer measurements, leading to statistically inconsistent and less accurate estimates compared to modern optimization techniques [37].

4.4.2 Optimization-Based Methods (e.g., VINS-Mono, ORB-SLAM3):

In contrast to filters, these methods formulate VINS as a non-linear least-squares optimization problem. They collect measurements over a period of time and solve for a trajectory that best explains all measurements simultaneously. This is typically done over a sliding window of recent states to ensure real-time performance. The cost function being minimized includes two main terms: the photometric reprojection error, which measures the discrepancy between observed visual features and their predicted locations based on the estimated 3D structure and camera poses, and the IMU pre-integration error, which summarizes the IMU measurements between consecutive camera frames into a single relative motion constraint[38].

- **Advantages:** These methods achieve superior accuracy by repeatedly re-linearizing the problem and performing joint optimization over multiple poses and feature locations in a process called Bundle Adjustment (BA). VINS-Mono was a landmark system demonstrating that a tightly-coupled, optimization-based monocular VIO could achieve high-accuracy results in real-time on a mobile device [7]. ORB-SLAM3 further advances this by being a complete multi-map and multi-sensor system, able to perform visual, visual-inertial, and multi-session SLAM with unparalleled robustness and accuracy [8].

- **Disadvantages:** The main weakness is their reliance on the static world assumption. They assume that all feature points correspond to stationary objects. When this assumption is violated by moving cars, pedestrians, or even swaying foliage, incorrect data associations are made, leading to a degradation of accuracy or outright tracking failure [9]. Additionally, the computational cost, while managed by sliding window techniques, is inherently higher than that of filter-based approaches.

4.4.3 DROID-SLAM

This system represents a significant step forward in end-to-end learning for SLAM. It treats SLAM as an iterative refinement problem, using a recurrent neural network to process video frames and update a dense, per-pixel depth map and camera poses. Its core innovation is a Dense Bundle Adjustment (DBA) layer, which is a differentiable module that enforces geometric consistency across the entire video, allowing for end-to-end training with only a photometric loss [39].

- **Advantages:** DROID-SLAM achieves state-of-the-art accuracy and robustness, particularly in its resistance to initialization failures and tracking loss in challenging, low-texture scenes. The learned data priors help it to regularize the ill-posed monocular depth estimation problem effectively [39].
- **Disadvantages:** Its dense nature makes it extremely computationally intensive, requiring a high-end GPU to run in real-time. This makes it largely unsuitable for deployment on size, weight, and power (SWaP)-constrained UAV platforms.

4.4.4 DPVO (Deep Patch Visual Odometry)

Seeking to combine the accuracy of learning-based methods with greater efficiency, DPVO uses a recurrent network to track a sparse set of image patches, rather than computing dense optical flow. This approach is inspired by traditional sparse feature tracking but replaces handcrafted feature descriptors with learned patch representations [40].

- **Advantages:** DPVO is significantly more efficient than DROID-SLAM, running up to 3x faster and using only a third of the GPU memory while achieving comparable or better accuracy. This makes it a more viable candidate for real-time applications [12].
- **Disadvantages:** A key limitation is its reliance on supervised training, which requires ground truth pose and flow data. This data is often sourced from synthetic datasets, creating a potential sim-to-real domain gap that can affect its generalization performance in novel, real-world environments [40].

4.4.5 LRD-SLAM

This system exemplifies a powerful hybrid approach. It augments the highly-refined, geometry-based ORB-SLAM3 with a parallel thread running a lightweight deep learning model for object detection (e.g., YOLOv5). This thread provides semantic information, allowing the main SLAM pipeline to identify and reject feature points that lie on predefined dynamic object classes [41].

- **Advantages:** It directly mitigates the primary failure mode of traditional SLAM by robustly handling dynamic environments. This results in a significant boost in localization accuracy and system robustness in real-world scenarios, while maintaining the real-time performance of the underlying ORB-SLAM3 framework [41].
- **Disadvantages:** Its effectiveness is fundamentally limited by the capabilities of the object detector. It cannot handle moving objects that do not belong to one of its known classes, and its performance may degrade if the detector is not robust to different viewpoints or lighting conditions.

4.4.6 CUAHN-VIO

This is a specialized VIO system designed for agile MAVs equipped with a downward-facing camera. Its vision front-end is a neural network that estimates the 2D homography transformation between frames, which is valid for planar scenes. Crucially, the network is also trained to predict its own uncertainty, providing a confidence score for each homography estimate [42].

- **Advantages:** The system is highly robust to severe motion blur encountered during high-speed flight. The predicted uncertainty is used by the EKF back-end to adaptively weight the visual measurements, ignoring them when the network is uncertain (e.g., due to non-planar objects) and relying more on the IMU. This results in a stable and low-latency ($\sim$26ms) system capable of running on embedded processors [42].
- **Disadvantages:** Its core reliance on a homography model restricts its application to scenarios where the UAV is flying over predominantly planar surfaces. Its performance would degrade significantly in environments with large, non-planar 3D structures.

4.5 Summary

Autonomous UAV navigation in GPS-restricted or GPS-denied environments (urban canyons, indoors, under canopy) demands sensor-centric localization that is both accurate and robust. Visual-Inertial Navigation Systems (VINS) have emerged as the leading approach because they fuse complementary modalities: cameras provide rich, drift-free geometric constraints while low-cost MEMS IMUs deliver high-rate motion cues that bridge visual dropouts. The review shows that this complementarity is most effectively exploited in tightly-coupled pipelines that ingest raw feature measurements and IMU pre-integration within a single probabilistic estimator, outperforming loosely-coupled schemes that fuse only pose outputs. On the estimation back-end, optimization/smoothing with sliding-window bundle adjustment typically yields higher accuracy and better consistency than filtering (EKF), at the cost of greater compute and implementation complexity. At the sensor level, the IMU error model (biases, scale-factor, noise) and camera configuration (monocular vs stereo) set fundamental limits: monocular systems require motion-rich initialization to resolve metric scale, whereas stereo adds hardware complexity but gives instant scale. A critical enabling technique is IMU pre-integration, which compresses hundreds of IMU samples between keyframes into a bias-aware relative motion factor, making tight fusion tractable in real time. Algorithmically, classical filter-based VIO (e.g., MSCKF) is efficient and suitable for constrained hardware but is more sensitive to linearization errors. Optimization-based VIO/SLAM (e.g., VINS-Mono, ORB-SLAM3) achieves state-of-the-art accuracy via joint re-linearization, loop closure, and online calibration. Building on these, recent trends incorporate learning: fully end-to-end dense methods (e.g., DROID-SLAM) improve robustness in low-texture scenes but are GPU-heavy, while sparse learned patch trackers (e.g., DPVO) strike a better accuracy-efficiency balance. Hybrid, semantics-aware systems (e.g., augmenting ORB-SLAM3 with lightweight object detectors) explicitly down-weight or mask dynamic objects, mitigating the principal real-world failure mode of geometry-only pipelines. Specialized designs (e.g., homography-based VIO for planar, high-speed flight) further demonstrate task-specific robustness by coupling learned uncertainty with adaptive fusion. Across the literature, the hard problems are consistent: (i) IMU drift without frequent visual corrections, (ii) low-texture, low-light, motion-blur and high dynamics that reduce reliable features, (iii) initialization and scale observability for monocular setups, and (iv) computational budget on SWaP-constrained UAVs. The converging direction is clear: tightly-coupled, optimization-based VINS augmented by semantics (to handle dynamics), principled sensor error modeling and pre-integration, and deployment-minded engineering (time synchronization, calibration, and resource management) to meet real-time constraints on

embedded platforms. This synthesis positions VINS as the most practical pathway to reliable, high-accuracy UAV autonomy where GPS is unreliable or unavailable.

Table 1: Comparison of Prominent VINS Algorithms

Algorithm	**Core Method**	**Coupling**	**Camera Support**	**Key Innovations & Strengths**	**Key Limitations & Challenges**
MSCKF	Filter-based (EKF)	Tightly	Monocular, Stereo	• Efficiency: Avoids adding features to state vector, linear complexity • Consistency: Good consistency properties in later versions. • Accuracy in low-texture scenes.	• Accuracy: Generally less accurate than optimization methods. • Sensitive to tuning • Delayed update can be suboptimal.
OKVIS	Optimization-based (BA)	Tightly	Stereo, Monocular	• Accuracy: High accuracy via joint nonlinear optimization of visual and inertial errors. • Keyframe Paradigm: Bounded complexity for real-time use.	• Complexity: Higher computational cost than filters. • Highly sensitive to sensor calibration and synchronization.
VINS-Mono	Optimization-based (BA)	Tightly	Monocular	• Robustness & Versatility: Excellent all-around performance. • Robust Initialization: Can bootstrap from unknown states. • Includes loop closure and online calibration.	• Monocular setup requires motion for initialization. • Performance can degrade in challenging lighting without extensions (e.g., PC-VINS-Mono).
ORB-SLAM3	Optimization-based (MAP/BA)	Tightly	Monocular, Stereo, RGB-D	• Long-Term Autonomy: Multi-map system (*Atlas*) survives tracking loss and enables map reuse. • Accuracy: MAP-based estimation provides state-of-the-art accuracy.	• Complexity: Highly complex codebase. • Primarily a SLAM system, may be overkill for pure odometry tasks.

Table 2: Comparison of Deep Learning-based VINS Algorithms

Algorithm	**Core Method**	**Learning Paradigm**	**Camera Support**	**Key Innovations & Strengths**	**Key Limitations & Challenges**
DROID-SLAM	End-to-end Neural SLAM (Dense BA layer)	Self-Supervised	Monocular	• Dense Bundle Adjustment (DBA) layer with differentiable geometry. • Strong robustness against initialization failure and tracking loss. • Accuracy in low-texture scenes.	• Computationally heavy: Requires high-end GPU for real-time performance. • High power consumption • unsuitable for SWaP-constrained UAVs.
DPVO	Sparse Patch-based Neural VO	Supervised	Monocular	• Combines learned patch descriptors with sparse feature tracking. • Runs ∼3x faster than DROID-SLAM with significantly lower memory usage. • Maintains competitive accuracy.	• Requires ground-truth pose/flow for training. • Risk of sim-to-real domain gap reduces generalization in unseen environments.
LRD-SLAM	Hybrid: Geometry + Deep Object Detection	Supervised (Detector)	Monocular, Stereo	• Adds semantic awareness to ORB-SLAM3 pipeline via object detection (e.g., YOLOv5). • Robust against dynamic objects, improves real-world tracking stability.	• Limited to pre-defined object classes. • Detector must generalize well across lighting/viewpoints; otherwise performance degrades.
CUAHN-VIO	Neural Homography Front-End + EKF Back-End	Supervised with Uncertainty Prediction	Downward-facing Monocular	• Learns homography with uncertainty estimation. • Robust to high-speed motion blur. • Runs at ∼26 ms latency on embedded hardware. • Adaptive EKF weighting improves stability.	• Homography assumption limits use to mostly planar surfaces. • Accuracy drops significantly in 3D-structured environments.

5 Methodology

5.1 Proposed Solution

For this project, we propose the development and implementation of a hybrid, optimization-based Visual-Inertial Navigation System (VINS). This system will be built upon a robust, state-of-the-art geometric SLAM framework, such as ORB-SLAM3, and will be augmented with a real-time, lightweight deep learning module for dynamic object segmentation. This approach is designed to synergistically combine the geometric precision of classical methods with the semantic scene understanding of modern AI.

5.1.1 Why It Is Chosen

This hybrid architecture was selected as it offers the most pragmatic and robust pathway to achieving high-performance autonomous navigation on a UAV.

- **Synergy of Paradigms:** It leverages the decades of research that have gone into making geometric SLAM systems like ORB-SLAM3 incredibly accurate and efficient. Instead of replacing this mature technology, we augment it, using deep learning to solve a problem that is intractable for pure geometry: semantic understanding. This "best of both worlds" approach has been shown to be highly effective in recent research [43].
- **Targeted Problem Solving:** The proposed solution directly addresses the most common failure mode for VINS in real-world environments: dynamic objects. By explicitly detecting and rejecting data from moving entities, the system's core geometric estimator is protected from corruption, dramatically increasing its reliability.
- **Feasibility for Embedded Deployment:** A key design constraint for any UAV system is real-time performance on SWaP-constrained hardware. Full deep-learning systems are often too demanding. However, modern lightweight object detection models like YOLO are highly optimized and have been demonstrated to run at high frame rates on embedded AI platforms like the NVIDIA Jetson series, which are commonly used on research UAVs [44]. This makes our proposed hybrid architecture computationally feasible.

5.1.2 Advantages

The expected benefits of this solution are threefold:

- **Superior Localization Accuracy:** By systematically identifying and excluding unstable features on moving objects, the bundle adjustment algorithm will operate on a cleaner, more consistent set of data. This will lead to a significant reduction in pose drift and a more accurate trajectory estimate, particularly over long-duration missions in dynamic environments.
- **Unprecedented System Robustness:** The system will be resilient to dynamic disturbances that would cause traditional VINS to lose tracking or fail completely. This robustness is essential for safe and reliable autonomous operation in unpredictable, human-centric environments.

- **Foundation for Higher-Level Autonomy:** The semantic information generated by the object detection module is not just useful for filtering; it creates a semantic map of the environment. This map, which annotates objects with their class labels (e.g., "person", "vehicle"), is an invaluable input for advanced autonomy tasks such as risk-aware motion planning, safe human-robot interaction, and context-aware decision making [45].

5.1.3 Disadvantages

Despite the clear advantages, several challenges and potential drawbacks must be acknowledged and addressed:

- **Increased System Latency and Complexity:** Integrating a deep learning pipeline into a real-time SLAM system adds complexity. The inference time of the neural network introduces a latency between when an image is captured and when its semantic content is known. This requires careful timestamping and buffer management to ensure that the correct features are filtered in the tracking thread without disrupting the real-time operation of the system [46].

- **Brittleness of the Perception Module:** The system's robustness to dynamics is fundamentally bounded by the performance of the object detector. It will be unable to handle moving objects for which it has not been trained. Furthermore, false positives (misclassifying a static object as dynamic) could lead to the unnecessary removal of stable map points, which could weaken the system's ability to localize, especially in environments that are already sparse in features.

- **Computational Resource Management:** While feasible, running both a full SLAM system and a CNN simultaneously on an embedded computer requires careful resource management. CPU and GPU usage must be monitored and potentially balanced to prevent thermal throttling and ensure that no single process starves the others of necessary computational resources, which could compromise the stability of the entire system.

5.2 Datasets & Benchmark-First Validation Plan

To de-risk real-world flights and enable fair comparison with prior work, we adopt a benchmark-first strategy before hardware-in-the-loop testing. We will:

1. Bring-up and tune on **EuRoC MAV** (stereo+IMU, indoor lab/factory sequences),

2. Stress-test on **TUM-VI** (stereo+IMU, HDR/low-light, wide-FOV),

3. Validate outdoor generalization on **KITTI** (stereo images; KITTI Raw for OXTS GPS/IMU when needed).

Optional stress datasets include **UZH-FPV** (aggressive drone trajectories) and **Hilti-Oxford** (millimetre-level ground truth) once the pipeline is stable. We evaluate per dataset using identical metrics and configurations (see Evaluation Methodology), and only after passing pre-specified thresholds (see Success Criteria) do we proceed to embedded deployment and flight tests.

5.3 System Architecture

The proposed system follows a modular pipeline:

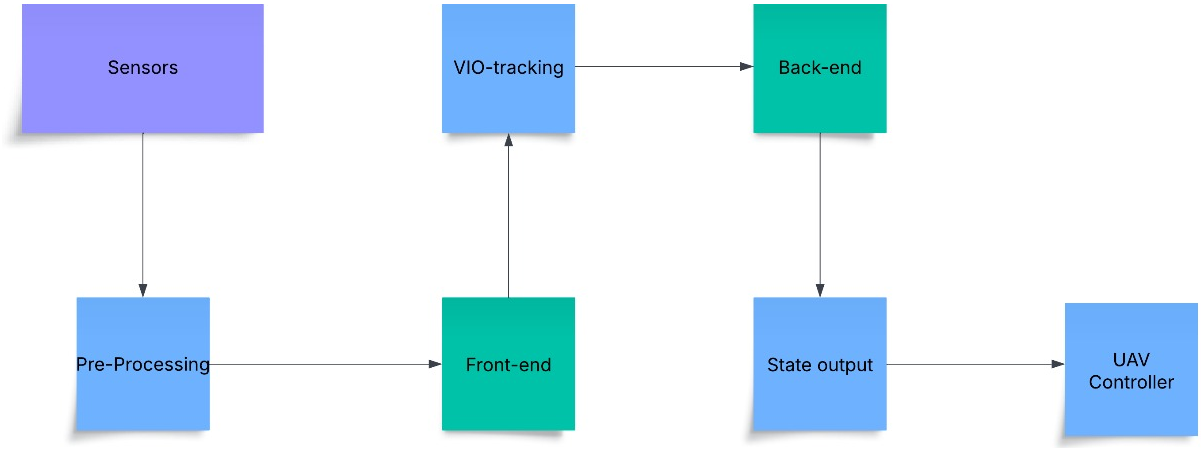


Figure 6: System Architecture

- **Sensors:** Stereo/mono camera(s) + IMU with hardware time-sync.
- **Pre-processing:** Camera undistortion/rectification; IMU de-biasing; strict timestamp alignment.
- **Front-end:** Feature detection/description; dynamic-object masking via a lightweight detector with confidence gating and temporal smoothing.
- **VIO Tracking:** Visual–inertial fusion with robust outlier rejection; re-initialization logic for tracking loss.
- **Back-end:** Sliding-window bundle adjustment optimizing poses, landmarks, and IMU biases; optional loop closure in mapped zones.
- **State Output:** Pose, velocity, and covariance streams at 20–30 Hz.
- **UAV Controller:** PX4/ArduPilot via MAVLink or ROS 2; offboard control for GPS-denied waypoint flight.

Real-time targets: 30 Hz input, ≤ 40 ms end-to-end latency, $\leq 80\%$ GPU utilization on Jetson-class hardware.

5.4 Hardware/Software Configuration

- **Compute:** NVIDIA Jetson Orin NX 8 GB (Ubuntu 22.04; CUDA/cuDNN pinned).
- **Flight stack:** PX4 $\geq$ v1.14; MAVLink; MAVSDK/MAVROS (ROS 2 Humble).
- **Sensors:** Stereo global-shutter camera @ 640×480, 30 Hz; IMU with ≤ 1 ms time-sync; AprilTag board for calibration.
- **Libraries:** OpenCV, Eigen, Ceres/G2O, Sophus; ORB-SLAM3 (VIO mode) or equivalent; lightweight detector (YOLO-Tiny/Nano) with INT8 quantization when needed.
- **Build & CI:** Version-pinned environments, reproducible scripts, and continuous evaluation against EuRoC/TUM-VI/KITTI.

5.5 Evaluation Methodology

We evaluate across three settings:

1. Indoor low-texture corridors,
2. Outdoor dynamic scenes (people/vehicles),
3. Stress tests (low-light, motion blur).

For each sequence we report:

- **Accuracy:** Absolute Trajectory Error (ATE) and Relative Pose Error (RPE).
- **Robustness:** Tracking uptime (% of frames with valid pose); time-to-recover after induced loss.
- **Throughput:** End-to-end FPS and median per-frame latency on the target embedded device.
- **Ablations:** With/without dynamic-object masking; detector thresholds; mono vs stereo; photometric calibration on/off (for TUM-VI).

Each scenario is repeated ≥ 5 runs; we report mean $\pm$ std and best/worst cases. Evaluation is performed with the `evo` toolkit (consistent configs and seeds).

Table 3: Evaluation Scenarios and Pass Criteria

Scenario	**Conditions**	**Metrics**	**Pass Criteria**
Indoor corridor	Low-texture, good light	• ATE • RPE • FPS	• ATE ≤ 0.25 m • RPE $\leq 2\%$ • ≥ 20 FPS
Outdoor dynamic	Pedestrians/vehicles	• ATE • Uptime • Recovery	• ATE ≤ 0.50 m • Uptime $\geq 95\%$ • Recovery ≤ 2 s
Low-light	< 100 lux	• ATE • Uptime	• ATE ≤ 0.60 m • Uptime $\geq 90\%$
Motion blur	Rapid yaw/pitch	• RPE • Recovery	• RPE $\leq 3\%$ • Recovery ≤ 2 s
Ablation: no masks	Same as outdoor	• ΔATE, • ΔUptime	• Degradation $\leq 30\%$ vs. masked

5.6 Success Criteria (Definition of Done)

- **Accuracy:** ATE ≤ 0.25 m (indoor) and ≤ 0.50 m (outdoor); RPE $\leq 2\%$ over 100 m.
- **Robustness:** $\geq 95\%$ tracking uptime in dynamic scenes; ≤ 2 s recovery after induced loss.
- **Real-time:** ≥ 20 FPS full pipeline on Jetson Orin NX 8 GB with $\leq 80\%$ GPU utilization.
- **Demo:** GPS-denied waypoint flight with live pose feedback and log replay.
- **Reproducibility:** Version-pinned environments, scripts, and public configs.

6 Project Management

6.1 Project Activity Timeline

6.1.1 Phase 1: Problem Formulation and Requirement Analysis

This initial stage lays the foundation for the entire project by ensuring that the problem is well-defined, relevant, and achievable. It establishes a clear direction and avoids scope creep later.

- **Literature Review and Background Research:** Conduct a systematic review of academic papers, technical reports, and existing implementations of Visual-Inertial Navigation Systems (VINS), drone navigation frameworks, and swarm robotics.

  Purpose: Understand current capabilities, identify common limitations (e.g., drift errors, processing latency, environmental constraints), and recognize trends in drone-based navigation research.

  Outcome: A consolidated knowledge base to guide technical decisions.

- **Gap Identification and Problem Statement:** Based on literature findings, identify gaps in current research, such as limited performance in GPS-denied environments or scalability issues in swarm navigation. Formulate a clear problem statement that encapsulates these gaps.

- **Requirement Gathering:** Define functional requirements (e.g., real-time localization, marker-based pose estimation, swarm coordination) and non-functional requirements (e.g., low computational latency, high robustness against sensor noise).

- **Project Proposal Development:** Prepare a formal proposal containing objectives, methodology, scope, anticipated challenges, resource needs, and success metrics. This document serves as the agreement point before moving into technical work.

6.1.2 Phase 2: Resource Gathering

This phase ensures that all essential hardware, datasets, algorithms, and software platforms are prepared before development begins. It prevents delays due to missing components later.

- **Model Selection:** Review and shortlist open-source and proprietary VINS algorithms such as VINS-Mono, ORB-SLAM3, or VINS Fusion.

Selection Criteria: Accuracy, computational efficiency, ease of integration, hardware requirements.

- **Dataset Acquisition:** Obtain benchmark datasets (e.g., EuRoC MAV, KITTI) for algorithm testing and training. Additionally, plan custom dataset collection using drone-mounted cameras and IMUs in varied environments.

  Purpose: Ensure the models are tested in scenarios similar to real deployment conditions.

- **Hardware Resource Gathering:** Identify and procure all necessary hardware components such as drones, cameras, IMUs, microcontrollers, onboard computers (e.g., Jetson Nano, Raspberry Pi), servos, and communication modules. Ensure all devices meet the project's technical specifications and are compatible with planned software platforms.

  Purpose: Prevent hardware shortages or mismatches during integration and testing.

- **Tool and Platform Setup:** Install and configure simulation platforms (Gazebo, AirSim, ROS), development environments (Python, C++), and supporting libraries (OpenCV, Eigen, g2o).

  Hardware setup: Ensure the drone's onboard computer and peripheral sensors are functional and calibrated.

6.1.3 Phase 3: Model Evaluation

Here, shortlisted models are rigorously tested under controlled conditions to determine the best candidate for real-world integration.

- **Platform-Based Testing:** Deploy each selected VINS algorithm in simulated environments, replicating varied real-world conditions such as poor lighting, dynamic objects, and fast motion.

- **Performance Analysis:** Evaluate metrics such as Absolute Trajectory Error (ATE), Relative Pose Error (RPE), frame processing rate (FPS), and robustness under sensor noise.

  Purpose: Identify which model consistently meets or exceeds performance thresholds.

- **Comparative Evaluation:** Rank models according to performance scores and choose the most suitable candidate, justifying the selection with quantitative data.

6.1.4 Phase 4: Drone Integration

This stage focuses on implementing the chosen VINS system into a physical drone and testing its operational performance.

- **Drone Assembly and Implementation:** Assemble the drone hardware (frames, motors, sensors, communication modules) and integrate all electronic components. Ensure proper wiring, power distribution, and mechanical stability before software integration.
- **Algorithm Implementation:** Port the selected VINS model into the drone's onboard computing system (e.g., NVIDIA Jetson Nano, Raspberry Pi 4) while ensuring real-time execution.
- **VINS and Flight Controller Integration:** Interface the VINS output with the drone's flight control system (e.g., PX4, ArduPilot) so that navigation decisions are directly informed by visual-inertial data.
- **Flight Testing:** Conduct a series of indoor and outdoor test flights in different conditions:
  - Low light
  - High wind
  - GPS-denied areas

  Record performance metrics and operational stability for further refinement.

6.1.5 Phase 5: Model Improvements and Drone Swarm Exploration

Once a working prototype is validated, efforts shift toward improving robustness and extending functionality to swarm scenarios.

- **Limitation Analysis:** Use flight logs and recorded datasets to identify recurring problems such as drift, latency, or unstable tracking.
- **System Optimization:** Apply algorithmic improvements (e.g., better feature tracking, adaptive IMU calibration) and hardware enhancements (e.g., more powerful onboard processors).
- **Pose Estimation with Markers:** Integrate marker-based localization (e.g., AprilTags, ArUco markers) to assist pose estimation when visual features are scarce.

- **Swarm Coordination Exploration:** Begin experimental swarm operations, focusing on:
  - Inter-drone communication
  - Formation control
  - Collaborative mapping and navigation

6.1.6 Phase 6: Documentation and Dissemination

Final phase ensures that all technical, operational, and research details are preserved and communicated effectively.

- **Technical Documentation:** Prepare exhaustive documentation covering system architecture, integration steps, source code references, and testing methodologies.
- **Demonstration:** Organize a live or recorded demonstration showing the system's capabilities in real flight and/or swarm scenarios.
- **Research Publications:** Compile results into conference papers or journal articles to share findings with the academic and professional community.
- **Final Reports:** Produce a complete project report including methodology, results, discussion, limitations, and future work recommendations.

6.2 Proposed Task Allocation

The project tasks are distributed among the three team members to ensure equitable workload, with each member taking primary responsibility for specific sub-tasks while maintaining awareness of all aspects of the system. This structure promotes collaboration, accountability, and efficient progress across all phases.

Table 4: Task allocation among team members for all project phases.

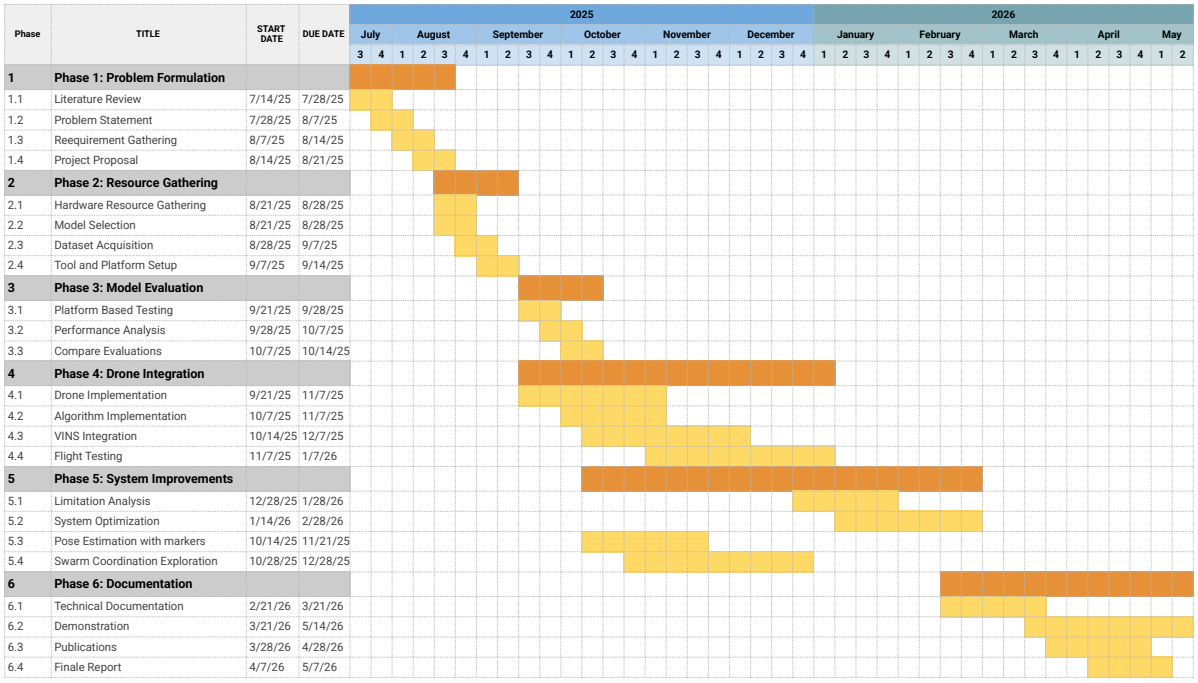
Phase	Akindu	Rashmi	Tishan
Phase 1: Problem Formulation & Requirement Analysis	• Literature review on geometric SLAM frameworks and VINS methods • Summarize historical approaches and key developments	• Literature review on deep learning-based object detection and dynamic scene understanding • Highlight limitations and recent advances	• Literature review on UAV navigation applications, system constraints, and swarm coordination • Consolidate insights, formulate problem statement, and draft initial requirements
Phase 2: Resource Gathering	• Shortlist VINS and SLAM algorithms • Set up initial software environments and dependencies	• Procure, assemble, and calibrate all hardware components including cameras, IMUs, and embedded computers • Verify compatibility	• Collect, preprocess, and organize benchmark datasets (EuRoC MAV, TUM-VI, KITTI) • Configure simulation environments (Gazebo, AirSim, ROS) • Document setup procedures
Phase 3: Model Evaluation	• Deploy candidate VINS models in simulation • Evaluate accuracy metrics (ATE, RPE) and frame processing rates • Generate performance charts	• Test models with hardware-in-the-loop • Monitor CPU/GPU usage, identify bottlenecks • Ensure stable execution	• Perform comparative evaluation, record robustness metrics • Run ablation experiments (dynamic-object masking, mono vs stereo) • Summarize results in tables
Phase 4: Drone Integration	• Implement selected VINS system on drone's onboard computer • Optimize code for real-time execution • Integrate with ROS 2 nodes	• Assemble and calibrate drone hardware • Integrate sensors, flight controller (PX4), and communication modules • Verify power distribution and mechanical stability	• Conduct indoor and outdoor flight tests • Record trajectory data, evaluate performance metrics • Ensure safe operation under varied conditions (low light, GPS-denied, high wind)
Phase 5: Model Improvement & Swarm Exploration	• Implement algorithmic enhancements: dynamic feature filtering, adaptive IMU calibration, photometric corrections • Update bundle adjustment parameters	• Upgrade hardware components as needed • Test multi-drone communication protocols and formation control strategies	• Analyze flight logs and evaluate pose estimation improvements • Conduct swarm experiments including collaborative mapping and coordinated navigation • Document results
Phase 6: Documentation & Dissemination	• Prepare technical documentation for all algorithms, integration steps, and software modules	• Prepare hardware integration manuals, calibration procedures, and experimental setup documentation	• Compile final project report, create visualizations • Organize demonstration materials • Prepare publications for academic dissemination

6.3 Summary of Phases

Shown in the following Table is a summary of project activities in each phase of the project.

Phase	Main Focus	Expected Outcomes
1. Problem Formulation	Define problem, requirements, proposal	Clear problem statement, project proposal, defined scope
2. Resource Gathering	Collect models, datasets, platforms	Ready-to-use algorithms, datasets, and simulation environments
3. Model Evaluation	Test and compare VINS models	Performance metrics, best model selected
4. Drone Integration	Implement and test VINS on hardware	Functional VINS-enabled drone with tested navigation
5. Model Improvement & Swarm Exploration	Optimize system, extend to swarm	Improved robustness, initial swarm coordination results
6. Documentation & Dissemination	Reporting and sharing findings	Technical documentation, publications, final project report

Table 5: Summary of Methodology Phases with Focus and Expected Outcomes.



Phase	TITLE	START DATE	DUE DATE
1	**Phase 1: Problem Formulation**		
1.1	Literature Review	7/14/25	7/28/25
1.2	Problem Statement	7/28/25	8/7/25
1.3	Reequirement Gathering	8/7/25	8/14/25
1.4	Project Proposal	8/14/25	8/21/25
2	**Phase 2: Resource Gathering**		
2.1	Hardware Resource Gathering	8/21/25	8/28/25
2.2	Model Selection	8/21/25	8/28/25
2.3	Dataset Acquisition	8/28/25	9/7/25
2.4	Tool and Platform Setup	9/7/25	9/14/25
3	**Phase 3: Model Evaluation**		
3.1	Platform Based Testing	9/21/25	9/28/25
3.2	Performance Analysis	9/28/25	10/7/25
3.3	Compare Evaluations	10/7/25	10/14/25
4	**Phase 4: Drone Integration**		
4.1	Drone Implementation	9/21/25	11/7/25
4.2	Algorithm Implementation	10/7/25	11/7/25
4.3	VINS Integration	10/14/25	12/7/25
4.4	Flight Testing	11/7/25	1/7/26
5	**Phase 5: System Improvements**		
5.1	Limitation Analysis	12/28/25	1/28/26
5.2	System Optimization	1/14/26	2/28/26
5.3	Pose Estimation with markers	10/14/25	11/21/25
5.4	Swarm Coordination Exploration	10/28/25	12/28/25
6	**Phase 6: Documentation**		
6.1	Technical Documentation	2/21/26	3/21/26
6.2	Demonstration	3/21/26	5/14/26
6.3	Publications	3/28/26	4/28/26
6.4	Finale Report	4/7/26	5/7/26

Figure 7: Research Timeline

7 Conclusion

This research project has successfully addressed the critical challenge of autonomous UAV navigation in GPS-restricted environments by implementing a robust Visual-Inertial Navigation System (VINS). The project's comprehensive literature review established VINS as the premier solution for GPS-denied navigation, highlighting its ability to fuse complementary data from a camera and an Inertial Measurement Unit (IMU) to achieve real-time, high-accuracy state estimation. The review further detailed the foundational principles of these sensors and the core architectural paradigms of VINS, including the superior performance of tightly-coupled and optimization-based approaches over filter-based methods.

The project's methodology, grounded in these findings, outlines a practical and systematic approach to building a reliable VINS. It details the selection of a specific solution, system architecture, hardware and software configuration, and a clear evaluation plan. The research timeline provides a structured, phased approach to achieving the project's objectives, from initial problem formulation and resource gathering to the final stages of drone integration, model improvements, and documentation.

In essence, this work provides a detailed framework for developing and validating a VINS for autonomous UAVs. The successful completion of the proposed research timeline will not only yield a functional navigation system but will also serve as a foundational step toward solving the "last mile" problem of UAV autonomy, enabling drones to safely operate in complex, real-world scenarios where GPS is unreliable. Future work will focus on integrating the improved model into a drone swarm to explore collective autonomy and enhance mission capabilities.

8 References

[1] F. Toscano et al., "Unmanned aerial vehicle for precision agriculture: A review," *IEEE Access*, pp. 1–1, Jan. 2024. DOI: 10.1109/access.2024.3401018

[2] *List of unmanned aerial vehicle applications*, Accessed: Apr. 12, 2020, 2020. [Online]. Available: https://en.wikipedia.org/wiki/List_of_unmanned_aerial_vehicle_applications

[3] Y. Cong et al., "Adaptive covariance matrix for uav-based visual-inertial navigation systems using gaussian formulas," *Sensors*, vol. 25, no. 15, p. 4745, Jan. 2025. DOI: 10.3390/s25154745

[4] M. Peretic et al., "Statistical analysis of gnss multipath errors in urban canyons," in *Proc. IEEE/ION Position Location and Navigation Symp. (PLANS)*, Apr. 2025, pp. 1216–1225. DOI: 10.1109/plans61210.2025.11028411

[5] W. Wen, X. Bai, and L.-T. Hsu, "3d vision aided gnss real-time kinematic positioning for autonomous systems in urban canyons," *Navigation*, vol. 70, no. 3, navi.590–navi.590, Jan. 2023. DOI: 10.33012/navi.590

[6] W. Wen, X. Bai, and L.-T. Hsu, *3d vision aided gnss real-time kinematic positioning for autonomous systems in urban canyons*, Accessed: 2025-08-20, 2023. [Online]. Available: https://www.polyu.edu.hk/aae/ipn-lab/us/publications/Fullpaper/3D%20Vision%20Aided%20GNSS%20Real-Time%20Kinematic%20Positioning%20for%20Autonomous%20Systems%20in%20Urban%20Canyons.pdf

[7] L. Zhao, W. Wang, Q. He, L. Yan, and X. Li, "Visual–inertial autonomous uav navigation in complex illumination and highly cluttered under-canopy environments," *Drones*, vol. 9, no. 1, pp. 27–27, Jan. 2025. DOI: 10.3390/drones9010027

[8] H. A. Hashim, "Advances in uav avionics systems architecture, classification and integration: A comprehensive review and future perspectives," *Results in Engineering*, vol. 25, pp. 103 786–103 786, Dec. 2024. DOI: 10.1016/j.rineng.2024.103786

[9] *Moving away from gps to a multi-sensor, inertial-centered architecture*, Accessed: Aug. 18, 2025, Jun. 2025. [Online]. Available: https://www.geoweeknews.com/news/moving-away-from-gps-to-a-multi-sensor-inertial-centered-architecture

[10] J.-C. Lee et al., "Landmark-based scale estimation and correction of visual inertial odometry for vtol uavs in a gps-denied environment," *Sensors*, vol. 22, no. 24, p. 9654, 2022.

[11] Palantir Technologies, *The future of drone navigation*, Accessed: 2025-08-20, 2024. [Online]. Available: https://blog.palantir.com/the-future-of-drone-navigation-7236075fdedf

[12] W. Huang, W. Wan, and H. Liu, "Optimization-based online initialization and calibration of monocular visual-inertial odometry considering spatial-temporal constraints," *Sensors*, vol. 21, no. 8, p. 2673, 2021.

[13] Y. Liu, Z. Feng, H. Zhang, and W. Dong, "Post-integration based point-line feature visual slam in low-texture environments," *Scientific Reports*, vol. 15, p. 14 606, 2025.

[14] P. Liu, X. Zuo, V. Larsson, and M. Pollefeys, "Mba-vo: Motion blur aware visual odometry," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5550–5559.

[15] Y. Gao et al., "A simultaneous control, localization, and mapping system for uavs," *Drones*, vol. 9, no. 1, p. 69, 2025, Accessed: Aug. 14, 2025. DOI: 10.3390/drones9010069

[16] *Vision-based localization methods under gps-denied conditions*, Accessed: Aug. 14, 2025, 2022. [Online]. Available: https://arxiv.org/pdf/2211.11988

[17] *Viavi launches vins to tackle rising gps jamming for uav navigation*, Accessed: Aug. 14, 2025, 2025. [Online]. Available: https://www.nasdaq.com/articles/viavi-launches-vins-tackle-rising-gps-jamming-uav-navigation

[18] T. Qin, P. Li, and S. Shen, *Vins-mono: A robust and versatile monocular visual-inertial state estimator*, Accessed: Aug. 14, 2025, 2017. [Online]. Available: https://arxiv.org/abs/1708.03852

[19] *Visual-inertial navigation system*, Accessed: Aug. 14, 2025, 2025. [Online]. Available: https://www.uavnavigation.com/company/blog/visual-inertial-navigation-system

[20] *A comprehensive introduction of visual-inertial navigation*, Accessed: Aug. 14, 2025, 2023. [Online]. Available: https://arxiv.org/pdf/2307.11758

[21] *Inertial measurement unit (imu) — an introduction*, Accessed: Aug. 14, 2025, 2025. [Online]. Available: https://www.advancednavigation.com/tech-articles/inertial-measurement-unit-imu-an-introduction/

[22] *Improving monocular visual-inertial initialization with structureless visual-inertial bundle adjustment*, Accessed: Aug. 14, 2025, 2025. [Online]. Available: https://arxiv.org/html/2502.16598v1

[23] Y. Wu et al., "Mems imu error mitigation using rotation modulation technique," *Sensors*, vol. 16, no. 12, p. 2017, 2016, Accessed: Aug. 14, 2025. DOI: 10.3390/s16122017

[24] T. Rehbinder et al., "An evaluation of mems-imu performance on the absolute trajectory error of visual-inertial navigation system," *Micromachines*, vol. 13, no. 4, p. 602, 2022, Accessed: Aug. 14, 2025. DOI: 10.3390/mi13040602

[25] A. Jain, *Monocular vs stereo vs monochrome camera*, Accessed: Aug. 14, 2025, 2023. [Online]. Available: https://medium.com/@abhishekjainindore24/monocular-vs-stereo-vs-monochrome-camera-16612bf4358b

[26] T. Qin, P. Li, and S. Shen, *Vins-mono: A robust and versatile monocular visual-inertial state estimator*, Accessed: Aug. 14, 2025, 2017. [Online]. Available: https://ar5iv.labs.arxiv.org/html/1708.03852

[27] A. Singh, *Visual odometry full tutorial*, Accessed: Aug. 14, 2025, 2023. [Online]. Available: https://avisingh599.github.io/vision/visual-odometry-full/

[28] *Loosely coupled & tightly coupled ins & gnss [2024 guide]*, Accessed: Aug. 14, 2025, 2024. [Online]. Available: https://pointonenav.com/news/loose-vs-tight-coupling-gnss/

[29] G. Huang et al., *Tightly-coupled fusion of global positional measurements in optimization-based visual-inertial odometry*, Accessed: Aug. 14, 2025, 2021. [Online]. Available: https://udel.edu/~ghuang/icra21-vins-workshop/papers/06-Cioffi_global-VIO.pdf

[30] *Keyframe-based visual-inertial odometry using nonlinear optimization*, Accessed: Aug. 14, 2025, 2014. [Online]. Available: https://www.researchgate.net/publication/265683241_Keyframe-Based_Visual-Inertial_Odometry_Using_Nonlinear_Optimization

[31] X. Hu et al., "A review of visual-inertial simultaneous localization and mapping," *Robotics*, vol. 7, no. 3, p. 45, 2018, Accessed: Aug. 14, 2025. DOI: 10.3390/robotics7030045

[32] *Extended kalman filter navigation overview and tuning*, Accessed: Aug. 14, 2025, 2025. [Online]. Available: https://ardupilot.org/dev/docs/extended-kalman-filter.html

[33] *Sp-vio: Robust and efficient filter-based visual inertial odometry with state transformation model and pose-only visual description*, Accessed: Aug. 14, 2025, 2024. [Online]. Available: https://arxiv.org/html/2411.07551v1

[34] B. Triggs et al., *Bundle adjustment — a modern synthesis*, Accessed: Aug. 14, 2025, 2000. [Online]. Available: https://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Triggs00.pdf

[35] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2007, pp. 3565–3572. [Online]. Available: https://www.sp.cs.cmu.edu/people/adimitri/docs/Mourikis_ICRA_2007.pdf

[36] G. H. Lee, M. Pollefeys, and M. Fraundorfer, "Motion estimation for self-driving cars with a generalized camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2746–2753. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2013/papers/Lee_Motion_Estimation_for_2013_CVPR_paper.pdf

[37] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Visual slam: Why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65–77, 2012. [Online]. Available: https://www.doc.ic.ac.uk/~ajd/Publications/strasdat_ivc_2011.pdf

[38] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017. [Online]. Available: https://rpg.ifi.uzh.ch/docs/TRO17_Forster.pdf

[39] Z. Teed and J. Deng, "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. [Online]. Available: https://arxiv.org/pdf/2108.10869.pdf

[40] Z. Teed and J. Deng, "Deep patch visual odometry," in *European Conference on Computer Vision (ECCV)*, 2022. [Online]. Available: https://arxiv.org/pdf/2208.04726.pdf

[41] S. Liu and J. Miura, "Lrd-slam: Lightweight and robust dynamic slam with low latency," in *IEEE/SICE International Symposium on System Integration (SII)*, 2023. [Online]. Available: https://arxiv.org/pdf/2307.10449.pdf

[42] Y. Xu and G. C. H. E. de Croon, "Cuahn-vio: Content-and-uncertainty-aware homography network for visual-inertial odometry," *Robotics and Autonomous Systems*, vol. 157, p. 104 245, 2022. [Online]. Available: https://arxiv.org/pdf/2208.13935.pdf

[43] K. R. B. Y. Chen and W. Wang, *A survey on deep learning for visual slam*, 2021. arXiv: 2105.09521 [cs.RO]. [Online]. Available: https://arxiv.org/pdf/2105.09521.pdf

[44] NVIDIA, *Nvidia jetson agx orin delivers up to 275 tops for ai and autonomous machines*, Press release, 2021. [Online]. Available: https://nvidianews.nvidia.com/news/nvidia-jetson-agx-orin-delivers-up-to-275-trillion-operations-per-second-for-ai-at-the-edge

[45] S. Grinvald, F. Furrer, T. Novkovic, R. Siegwart, and J. Nieto, "Volumetric instance-aware semantic mapping and 3d object discovery," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3882–3889, 2019. [Online]. Available: https://arxiv.org/pdf/1907.05435.pdf

[46] Y. Cui and S. Chen, "Ds-slam: A semantic visual slam towards dynamic environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. [Online]. Available: https://arxiv.org/pdf/1809.08690.pdf