Transformer-based visual positioning and optimisation framework for GPS-denied high-altitude navigation

Aerial ROS

Thursday, 19 June 2025

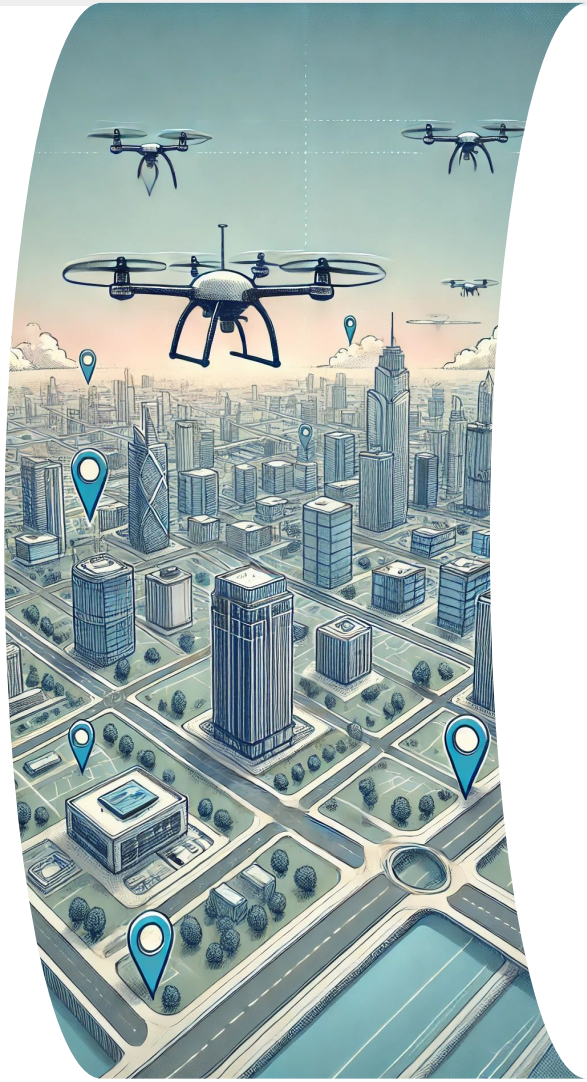
Sanket Sharma

About Me

Mechatronics graduate from India(2024)
GSOC contributor twice (2022 & 2024)
Tech Lead, Summon

Introduction

Develop a reliable vision positioning system for UAVs to operate at high altitudes in absence of GPS positioning.



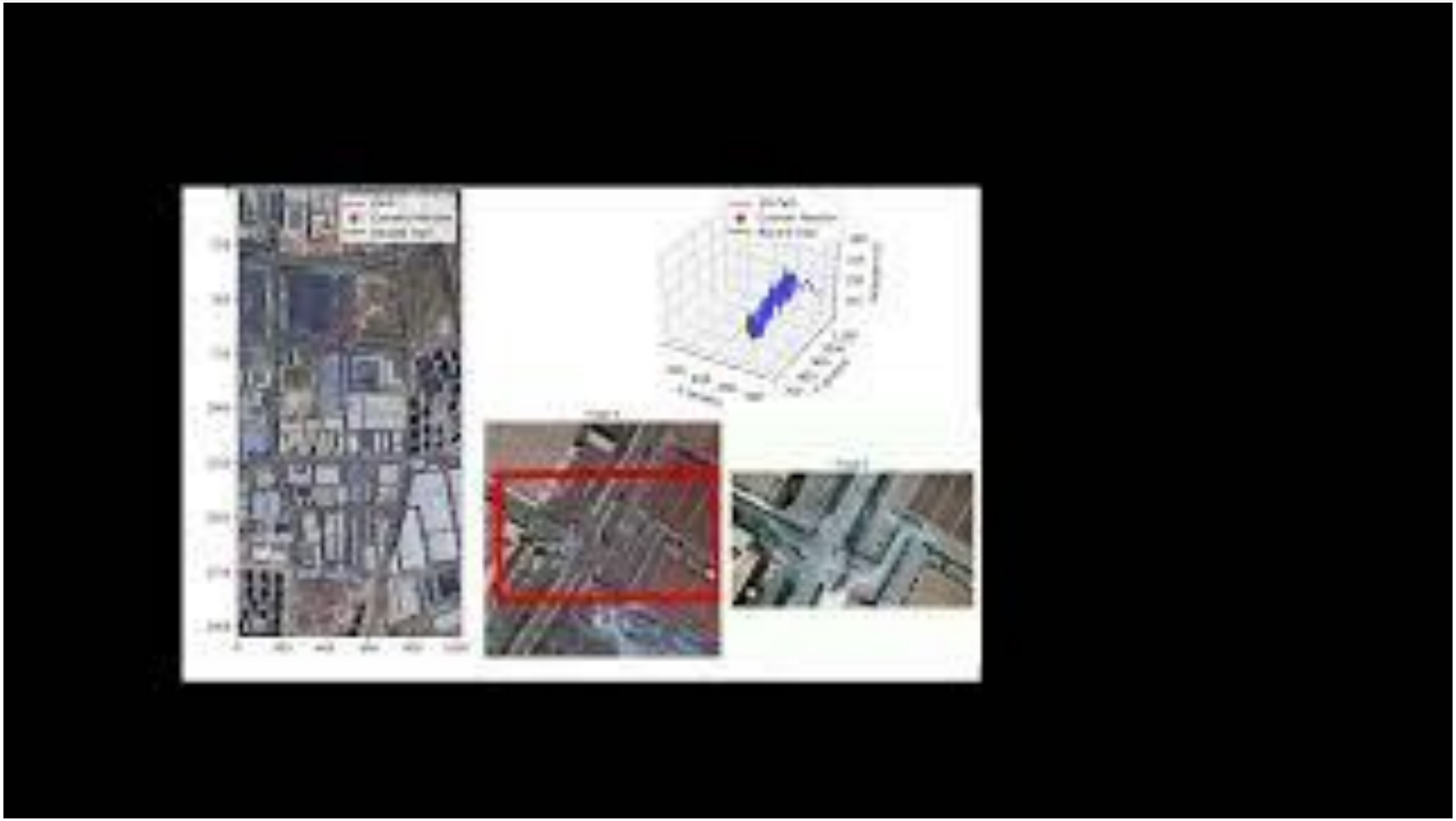
Why Non-GPS?

- GPS failures at high altitude and in interference zones
- Swarm and sentinel drones in strategic environments
- Need for Return-To-Launch (RTL) even without GNSS
- Real-time safety & fallback navigation

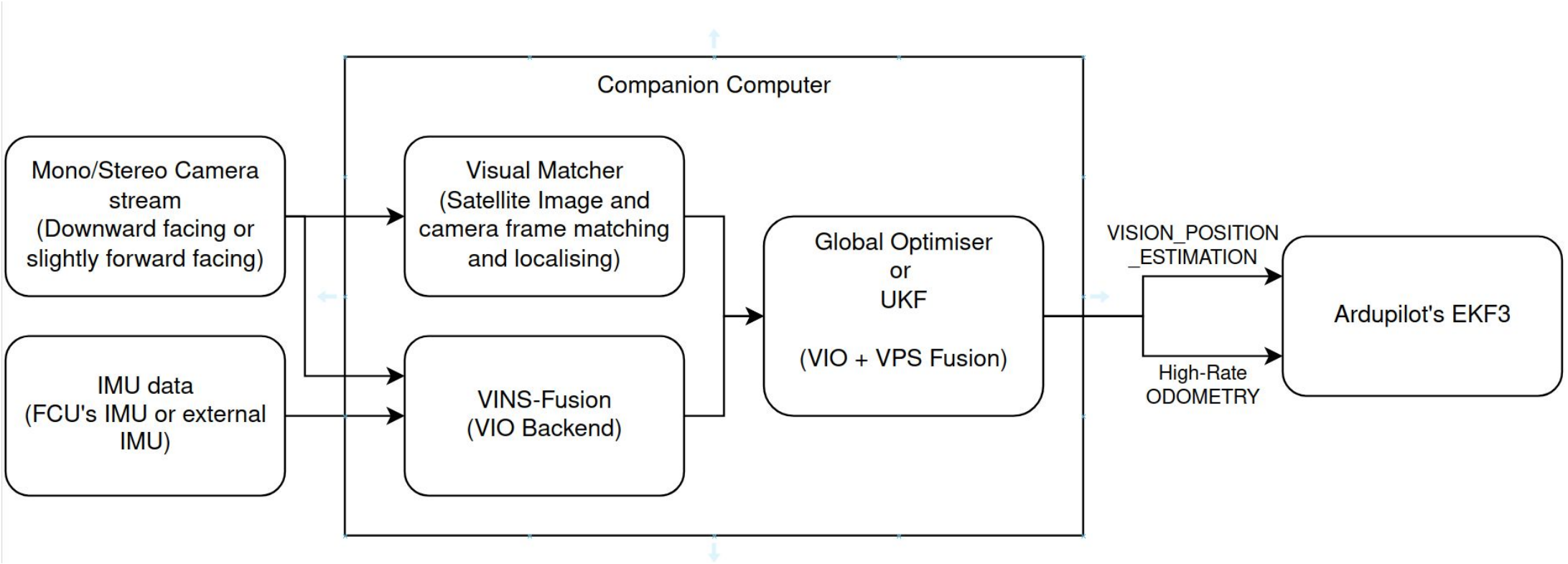
Implementation

Feature matching and Inertial Odometry based VPS

Inference

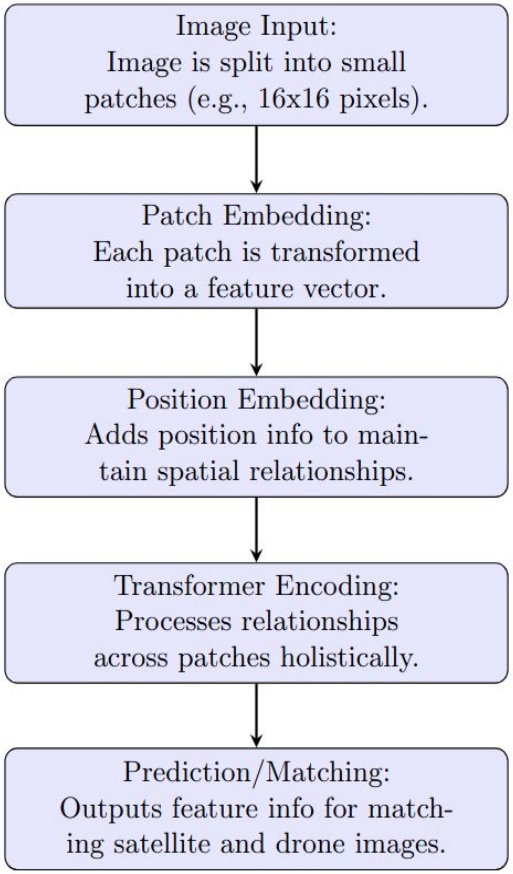


System Overview



Vision Transformers

- A Vision Transformer (ViT) processes images by breaking them down into small patches and treating each patch as a sequence of inputs, similar to how words are treated in natural language processing.



Concept:

The Attention Mechanism in transformers helps the model focus on the most relevant parts of the image when trying to match features between two different frames (e.g., satellite and drone images).

How Attention Helps in Feature Matching:

1. Global View
2. Focusing on Important Features
3. Robust Matching

Satellite Image Matching

A transformer-based architecture to match features between images for pose estimation.

Feature Matching:

- Uses Transformer-like attention mechanism
- Detects relevant regions even in low-texture/cloudy areas
- Keypoint descriptor based on a CNN backend (SuperPoint) and transformer based matcher architecture.

Implementation

➢ Real-time keypoint detection + match to satellite map (stored features and descriptors)

➢ Dynamic scaling of matching kernel based on drone's position and matching confidence

➢ Uses pruning, confidence filtering and dense matching

➢ Outputs global position (if using tiff) or in local NED frame (if using just the image of satellite map)

Visual-Inertial Odometry via VINS-Fusion

VINS-Fusion:

- GPU-enabled VIO runs at high frequency (IMU preintegration + Visual Reprojection + Marginalization)
- Optimizer updated to also accept VPS inputs
- Switched loss from Huber to Tukey for better outlier suppression (Still testing)

Advantage:

➢ Smooth high-rate local odometry

➢ Fallback when VPS update is unavailable

Why GPU-accelerated?

Massive Parallelism: Offloading per-frame feature detection & optical flow tracking (`cv::cuda::CornersDetector`, `cv::cuda::SparsePyrLKOpticalFlow`) to GPU

Reduced CPU Load: CPU fee to focus on optimization and other processes.

Higher Throughput: In GPU-accelerated, >10 Hz update rates on Jetson NX with 15% GPU usage, while CPU-only processing(RPi) provided well under 10 Hz with almost 100% usage.

Fusing VIO with VPS: Global Optimizer

- A separate modified VINS-Fusion's Global optimizer runs in ROS2
- Periodically corrects VIO drift using global VPS by optimising a loosely coupled pose-graph.
- Provides hybrid fusion of local and absolute positioning

Fusion Mode:

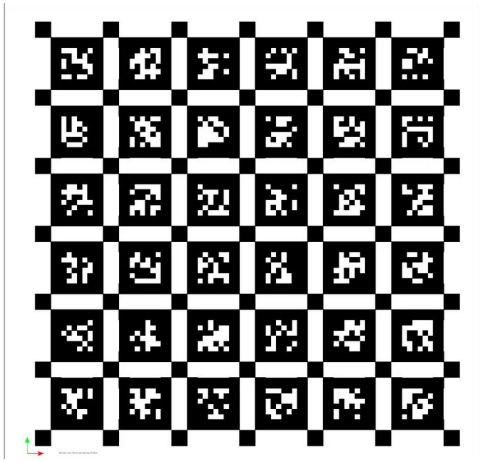
- VPS as low-rate, high variance absolute observation
- VIO as high-frequency prediction
- Global correction loop runs with VIO and VPS as independent factors
- Uses VIO inputs from frontend VINS-Fusion and positioning from satellite-matcher node

Camera-IMU Calibration

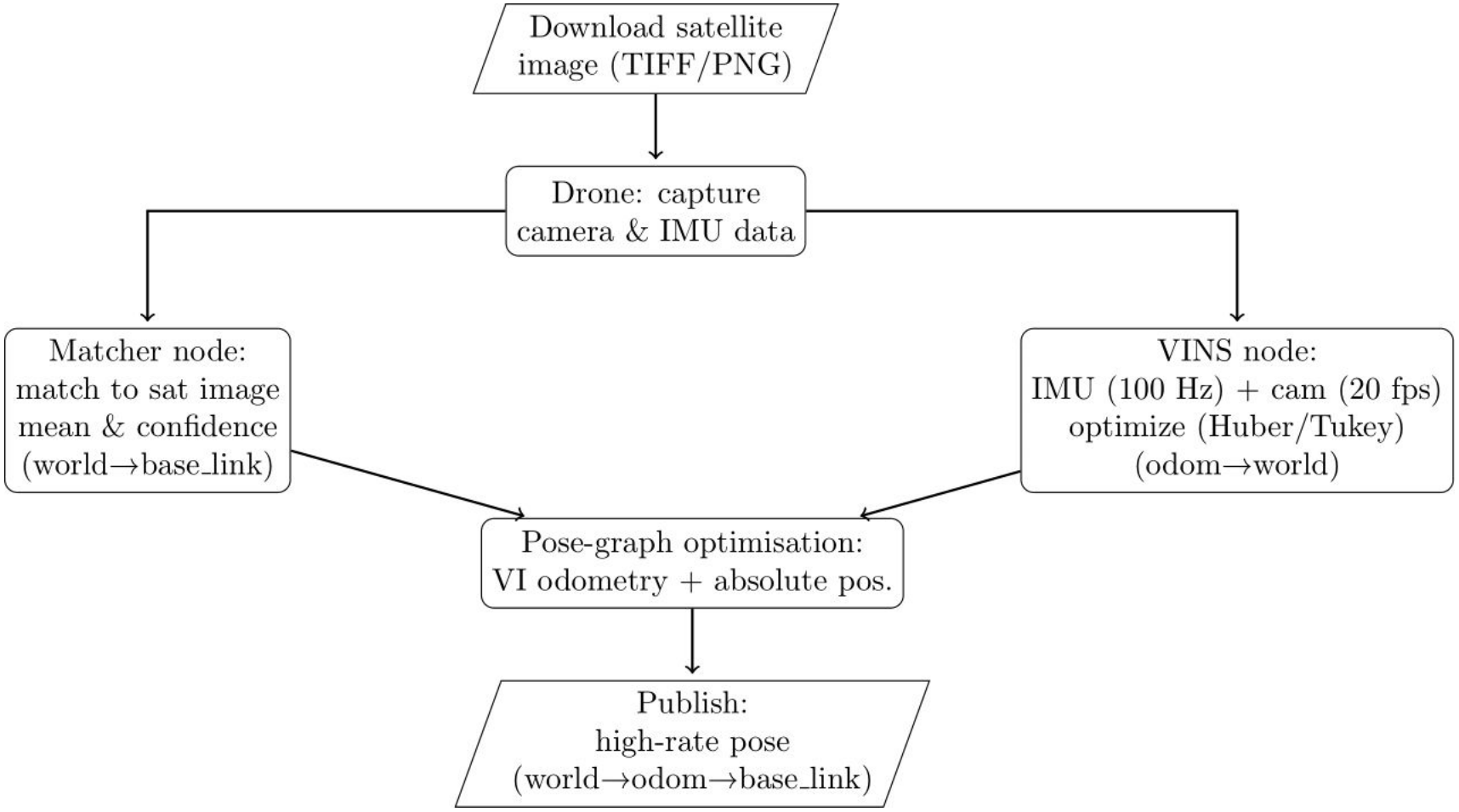
- **Mono Camera Calibration – Intrinsic:**
  Used Kalibr inside Docker (ROS 1) to calibrate mono camera. Kept the camera stationary and moved an AprilGrid.
  Outcome: Camera intrinsic parameters.

- **IMU Noise Characterization:**
  Used allan_ros2 to calculate allan parameters by recording rosbag and keeping it stationary for 2 hours. Estimated white noise and bias instability using Allan deviation.
  Outcome: IMU noise model for accelerometer and gyroscope.

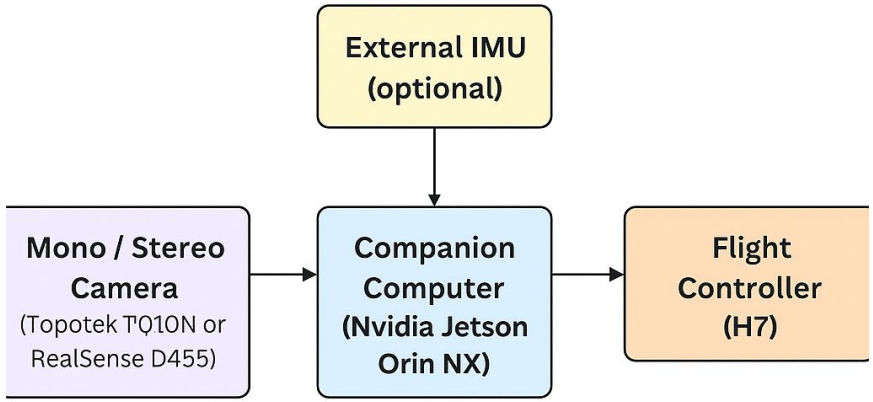
- **IMU–Camera Calibration – Extrinsic:**
  Ran Kalibr in Docker with rosbag containing data collected by keeping AprilGrid fixed and moved sensor module to excite all axes.
  Outcome: Extrinsics from IMU to camera frame.



Feature Matching Implementation



Hardware setup and ROS2 Stack



Real-Time Performance & Challenges

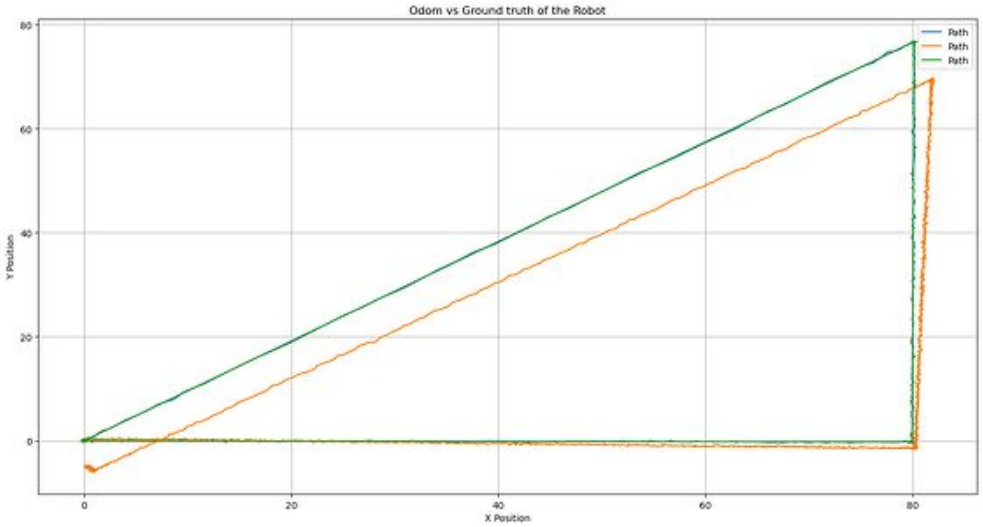
Achievements:

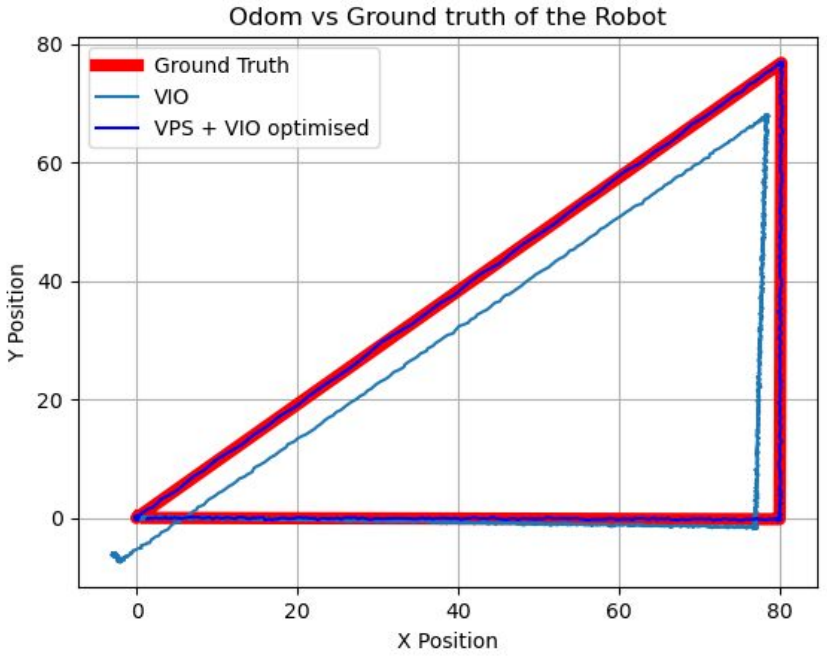
- Runs real-time (VPS @ ~1-2Hz, VIO @ ~10Hz, Fused @ ~10Hz)
- Accurate positioning even in sparse-texture areas
- Drift corrected VPS

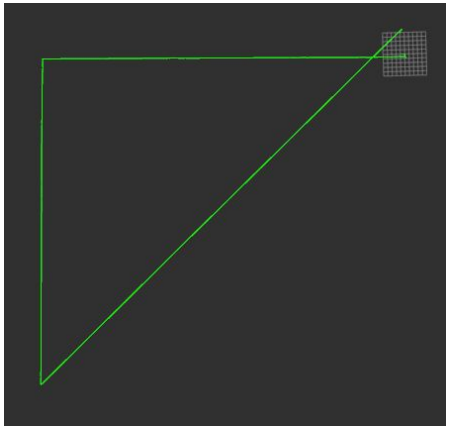
Challenges:

- Raspberry Pi 5 for just VIO (Damped Least Squares with optimised steps and caps) and occasional feature matching(FAST) for correction + EKF, but low accuracy and prone to drifts
- High compute (transformer attention = heavy for VPS+VIO at high rates)
- Clouds, water can cause false matches if too much noise.
- VPS dependent on location specific satellite image availability and manual pose initialisation

Results







Simulation results: ground truth vs. raw VPS vs. VPS fused with VIO. A 3-point triangle trajectory simulated in gazebo

Estimated Localization Performance – 150 m Altitude, 2 m/s

Method	ATE (m)	Remarks
Mono VIO	~20-22 m	High drift due to no loop closure
VPS + VIO	~4–6 m	VPS corrects drift with satellite matching

Open-Source & Security Considerations

Why not open all right away?

- Unintended exploitation in defense/surveillance
- Potential misuse

What will be public initially?

- VIO backend (Modified VINS-Fusion)
- ROS2 pipeline
- Integration tools (MAVROS bridge, scripts)
- Code will be available here: https://github.com/snktshrma/ngps_ap_flight

References

A. George, N. Koivumäki, T. Hakala, J. Suomalainen, and E. Honkavaara, “Visual-Inertial Odometry Using High Flying Altitude Drone Datasets,” Drones, vol. 7, no. 1, p. 36, 2023. https://doi.org/10.3390/drones7010036

S. Han, F. Deng, T. Li, and H. Pei, “Tightly Coupled Optimization-based GPS-Visual-Inertial Odometry with Online Calibration and Initialization,” arXiv preprint, arXiv:2203.02677, 2022. https://doi.org/10.48550/arXiv.2203.02677

J. Park et al., “Run Your Visual-Inertial Odometry on NVIDIA Jetson,” arXiv preprint, arXiv:2103.01655, 2021. https://arxiv.org/abs/2103.01655

A. Kashi, “Visual-Inertial Odometry on the MIT Racecars,” MIT AeroAstro Racecars Final Paper, 2020. https://github.com/mit-racecar/racecar-vio

Thank you!!

Questions?