

README

- [Low-Cost Star Tracker: A Software-Based Approach Using Consumer-Grade Hardware](#)
 - [Technical Paper](#)
 - [Abstract](#)
 - [Table of Contents](#)
 - [1. Introduction](#)
 - [1.1 Background](#)
 - [1.2 Problem Statement](#)
 - [1.3 Proposed Solution](#)
 - [1.4 Contributions](#)
 - [2. Literature Review](#)
 - [2.1 Traditional Star Tracker Technology](#)
 - [2.2 Gyroscope-Based Stabilization](#)
 - [2.3 Image Stacking Techniques](#)
 - [2.4 Low-Cost Star Tracker Initiatives](#)
 - [2.5 Consumer Camera Astrophotography](#)
 - [3. Commercial Star Tracker Analysis](#)
 - [3.1 Market Overview](#)
 - [3.2 Commercial Products Analysis](#)
 - [3.3 Cost Breakdown Analysis](#)
 - [3.4 Barrier to Entry](#)
 - [4. System Architecture](#)
 - [4.1 Overview](#)
 - [4.2 Processing Pipeline](#)
 - [4.3 Data Flow](#)
 - [4.4 Software Architecture](#)
 - [5. Hardware Components](#)
 - [5.1 Prototype Camera: Active Silicon Harrier 10x AF-Zoom](#)
 - [5.2 Prototype IMU: Orange Cube Flight Controller](#)
 - [5.3 SIL/HIL Validation Environment: Stellarium](#)
 - [5.4 Alternative Configuration A: GoPro Hero 7 Black](#)
 - [5.5 Alternative Configuration B: ZWO ASI585MC + Entaniya M12 220](#)
 - [5.6 Hardware Configuration Comparison](#)
 - [5.7 Cost Breakdown by Configuration](#)
 - [6. Camera + IMU Sensor Fusion](#)
 - [6.1 Fundamental Concept](#)
 - [6.2 Physical Setup: Rigid Camera-IMU Coupling](#)
 - [6.3 Coordinate Systems and Transformations](#)
 - [6.4 3D Celestial Sphere Visualization](#)
 - [6.5 IMU-Camera Calibration](#)
 - [6.6 Sensor Fusion Algorithms](#)
 - [6.7 Benefits for Star Tracking](#)
 - [6.8 Video Demonstration](#)
 - [6.9 Hybrid Stabilization: Gyroscope + Template Matching](#)
 - [6.10 Simple Star Solver \(Proof of Concept\)](#)
 - [7. IMU-Based Motion Deblur](#)
 - [7.1 The Motion Blur Problem](#)
 - [7.2 PSF Generation from IMU Data](#)
 - [7.3 Deconvolution Algorithms](#)
 - [7.4 The Star Trail Overlap Problem](#)
 - [7.5 Results: Motion Deblur Performance](#)

- [8.6 Implementation](#)
 - [7.7 Practical Considerations](#)
- [8. Software Algorithms](#)
 - [8.1 Gyroscope Data Processing](#)
 - [8.2 Motion Compensation](#)
 - [8.3 Star Detection](#)
 - [8.4 Triangle-Based Star Matching](#)
 - [8.5 Quality Assessment](#)
 - [8.6 Image Stacking](#)
- [9. Performance Analysis](#)
 - [9.1 Signal-to-Noise Ratio Improvement](#)
 - [9.2 Limiting Magnitude](#)
 - [9.3 Angular Resolution](#)
 - [9.4 Processing Performance](#)
 - [9.5 Accuracy Metrics](#)
 - [9.6 Vibration Attenuation Performance](#)
- [10. Cost Comparison](#)
 - [10.1 Our Low-Cost Systems](#)
 - [10.2 Comparison with Commercial Solutions](#)
 - [9.3 Cost-Effectiveness Ratio](#)
 - [9.4 Value Proposition](#)
- [11. Results and Discussion](#)
 - [11.1 Phase 1 Validation Results](#)
 - [11.2 Phase 2 Algorithm Enhancement Results](#)
 - [11.4 Advantages of Our Approach](#)
 - [11.3 Limitations](#)
 - [11.4 Comparison with Mechanical Tracking](#)
 - [11.5 Suitable Applications](#)
 - [11.6 Software-in-the-Loop Testing with Stellarium](#)
- [12. Conclusions and Future Work](#)
 - [12.1 Conclusions](#)
 - [12.2 Future Work](#)
- [13. References](#)
 - [Academic Literature](#)
 - [Technical References](#)
 - [Online Resources](#)
 - [Hardware References](#)
 - [Sensor Fusion References](#)
 - [Motion Deblur References](#)
- [Appendix A: System Requirements](#)
 - [A.1 Prototype: Harrier 10x + Orange Cube \(Validated\)](#)
 - [A.2 Alternative A: GoPro-Based System](#)
 - [A.3 Alternative B: ASI585MC + Entaniya All-Sky System](#)
 - [A.4 Software Dependencies](#)
- [Appendix B: Quick Start Guide](#)
 - [B.1 Installation](#)
 - [B.2 Basic Usage](#)
 - [B.3 Configuration File](#)
- [Appendix C: Glossary](#)
- [Changelog](#)
 - [Version 1.6 \(January 2026\)](#)
 - [Version 1.2 \(January 2026\)](#)
 - [Version 1.1 \(January 2026\)](#)
 - [Version 1.0 \(January 2026\)](#)

Low-Cost Star Tracker: A Software-Based Approach Using Consumer-Grade Hardware

Technical Paper

Authors: Giray Yillikci **Version:** 1.6 **Date:** January 2026

Abstract

This paper presents a novel low-cost star tracker system designed to capture faint starlight even in heavily light-polluted urban environments. The core approach combines gyroscope-based motion compensation with video frame stacking: by first stabilizing video captured from moving platforms—seaborne, airborne, and land-based vehicles—using high-rate IMU data, then aligning and stacking hundreds of frames, we effectively increase the signal-to-noise ratio to reveal stars that would otherwise be invisible in individual frames. Our hardware configuration centers on the Active Silicon Harrier 10x AF-Zoom Camera, featuring a 1/2.8" Sony IMX462LQR-C CMOS sensor with exceptional low-light performance (0.0004 lux minimum illumination in monochrome mode), 10x optical zoom ($f=5.1\text{-}51\text{mm}$, F1.6-1.8), and 1080p60 video output via simultaneous USB 3.1 and HDMI interfaces. The camera is paired with an Orange Cube flight controller (featuring the ICM-20948 9-axis IMU) for high-precision gyroscope-based motion compensation. The system is validated using Stellarium planetarium software to generate high-fidelity synthetic star fields displayed on a monitor, providing a controlled SIL/HIL test environment with configurable camera motion and vibration profiles. By implementing sophisticated frame stacking techniques and star detection algorithms, our system achieves a cost reduction of 95-99% compared to commercial star trackers while maintaining acceptable performance for amateur astrophotography, meteor detection, and educational applications. The complete system can be assembled for \$500-1,500, compared to \$10,000-\$500,000 for commercial alternatives.

Keywords: Star Tracker, Astrophotography, Gyroscope Stabilization, Image Stacking, Low-Cost Sensors, Harrier Camera, Orange Cube, ICM-20948 IMU, Stellarium, Motion Compensation, SIL/HIL Testing

Table of Contents

1. [Introduction](#)
2. [Literature Review](#)
3. [Commercial Star Tracker Analysis](#)
4. [System Architecture](#)
5. [Hardware Components](#)
6. [Camera + IMU Sensor Fusion](#)
7. [IMU-Based Motion Deblur](#)
8. [Software Algorithms](#)
9. [Performance Analysis](#)
10. [Cost Comparison](#)
11. [Results and Discussion](#)
12. [Conclusions and Future Work](#)
13. [References](#)

1. Introduction

1.1 Background

Star trackers are optical instruments that identify stars in an image and determine the orientation of a spacecraft or camera relative to the celestial sphere. They are essential components in spacecraft attitude determination systems, astronomical observations, and astrophotography. Traditional star trackers are precision instruments with costs ranging from tens of thousands to hundreds of thousands of dollars, making them inaccessible to amateur astronomers, educational institutions, and researchers with limited budgets.

1.2 Problem Statement

The high cost of commercial star trackers creates significant barriers for: - Amateur astronomers seeking quality astrophotography - Educational institutions teaching astronomy and space sciences - Researchers in developing countries - CubeSat and small satellite projects with limited budgets - Citizen science initiatives

1.3 Proposed Solution

We present a software-intensive approach that shifts complexity from expensive hardware to sophisticated algorithms. The core concept is to capture faint starlight even in light-polluted environments by combining gyroscope-based motion compensation with video frame stacking—stabilizing video from moving platforms (seaborne, airborne, and land-based vehicles), then aligning and stacking hundreds of frames to increase the signal-to-noise ratio and reveal stars invisible in individual frames.



Click thumbnail to watch video demonstration

Prototype System: Our validated prototype uses the Active Silicon Harrier 10x AF-Zoom Camera paired with an Orange Cube flight controller (ICM-20948 9-axis IMU). The system is tested using Stellarium planetarium software to generate high-fidelity synthetic star fields, providing a controlled SIL/HIL environment with configurable motion and vibration profiles.

Alternative Hardware: The software pipeline is designed to work with various camera+IMU combinations, including consumer options like the GoPro Hero 7 Black (\$200-400) with its embedded gyroscope, or dedicated astronomy cameras like the ZWO ASI585MC with external IMU.

1.4 Contributions

This work makes the following contributions: 1. A complete open-source star tracker pipeline using consumer hardware 2. Novel gyroscope-based motion compensation using quaternion mathematics 3. Adaptive frame stacking with quality-based selection 4. Triangle-based star matching algorithm for robust alignment 5. Comprehensive cost-benefit analysis compared to commercial solutions

2. Literature Review

2.1 Traditional Star Tracker Technology

Star trackers have evolved significantly since their inception in the 1960s. Early systems used photomultiplier tubes and mechanical scanning, while modern systems employ CCD and CMOS sensors with sophisticated pattern recognition algorithms.

2.1.1 Lost-in-Space Algorithms

The fundamental challenge in star tracking is the “Lost-in-Space” (LIS) problem—identifying stars without prior attitude knowledge. Several algorithms have been developed:

Grid Algorithm (Padgett & Kreutz-Delgado, 1997) Divides the celestial sphere into a grid and uses lookup tables for rapid identification. Computational complexity: $O(n^2)$ where n is the number of detected stars.

Pyramid Algorithm (Mortari et al., 2004) Uses four-star patterns forming pyramids for robust identification. Offers improved reliability but higher computational cost.

Geometric Voting (Kolomenkin et al., 2008) Employs a voting scheme based on angular distances between star pairs. Provides good balance between speed and reliability.

Triangle Algorithm (Liebe, 1993) Matches triangular patterns formed by star triplets. The basis for our implementation due to its rotation and scale invariance.

2.1.2 Centroiding Techniques

Accurate star position determination requires sub-pixel centroiding:

- **Center of Gravity (CoG):** Simple weighted average, accuracy ~0.1 pixels
- **Gaussian Fitting:** Models star PSF as 2D Gaussian, accuracy ~0.05 pixels
- **Iterative Weighted Centroiding (IWC):** Iteratively refines weights, accuracy ~0.02 pixels

2.2 Gyroscope-Based Stabilization

The use of gyroscopes for image stabilization has been extensively studied in both consumer electronics and aerospace applications.

2.2.1 MEMS Gyroscopes

Micro-Electro-Mechanical Systems (MEMS) gyroscopes have revolutionized motion sensing by providing compact, low-cost angular rate measurements. Modern action cameras like the GoPro Hero series incorporate 3-axis MEMS gyroscopes with:
- Sampling rates: 200-400 Hz
- Noise density: 0.005-0.01 °/s/√Hz
- Bias stability: 1-10 °/hr

2.2.2 Sensor Fusion

Combining gyroscope data with accelerometer and magnetometer readings (9-DOF fusion) improves orientation estimation. Common fusion algorithms include:

- Complementary filters
- Kalman filters (Extended and Unscented variants)
- Madgwick filter
- Mahony filter

2.2.3 Gyroflow Project

The open-source Gyroflow project (gyroflow.xyz) demonstrates the effectiveness of gyroscope-based video stabilization for action cameras. Our work extends these concepts specifically for astrophotography applications.

2.3 Image Stacking Techniques

Image stacking is fundamental to astrophotography, improving signal-to-noise ratio (SNR) by combining multiple exposures.

2.3.1 Stacking Methods

Mean Stacking Simple averaging of aligned frames. SNR improvement: \sqrt{n} where n is frame count. Sensitive to outliers (satellites, cosmic rays).

Median Stacking Uses median value at each pixel. Robust to outliers but discards valid signal. SNR improvement: $\sim 0.8\sqrt{n}$.

Sigma-Clipping Iteratively rejects pixels deviating more than $k\sigma$ from the mean. Balances outlier rejection with signal preservation.

Winsorized Mean Clips extreme values to specified percentiles before averaging. Computationally efficient approximation of sigma-clipping.

2.3.2 Frame Alignment

Accurate frame registration is critical for effective stacking:

- **Phase Correlation:** FFT-based translation detection
- **Feature Matching:** SIFT, ORB, or star-based keypoints
- **Optical Flow:** Dense motion estimation between frames

2.4 Low-Cost Star Tracker Initiatives

Several research groups have explored affordable star tracker alternatives:

2.4.1 CubeSat Star Trackers

ST-16 (Sinclair Interplanetary) Commercial CubeSat star tracker, ~\$50,000, 2 arcsec accuracy.

NST-1 (Naval Postgraduate School) Academic development, ~\$5,000 in components, 30 arcsec accuracy.

2.4.2 Smartphone-Based Systems

Research by Rijlaarsdam et al. (2020) demonstrated star tracking using smartphone cameras, achieving 0.05° accuracy in controlled conditions.

2.4.3 Raspberry Pi Systems

Multiple hobbyist projects use Raspberry Pi with camera modules (HQ Camera, ~\$50) for basic star tracking, though without integrated gyroscope stabilization.

2.5 Consumer Camera Astrophotography

Consumer cameras have gained popularity in the amateur astronomy community as accessible alternatives to dedicated astronomy equipment.

GoPro Action Cameras: The GoPro Hero series offers built-in gyroscopes and GPMF telemetry, making them attractive for motion-compensated astrophotography: - Night Lapse mode enables long-exposure sequences - Wide-angle lenses capture large star fields - Built-in IMU enables gyro-based stabilization - Raw format preserves maximum dynamic range

Industrial/Machine Vision Cameras: Cameras like the Active Silicon Harrier series offer superior low-light performance (0.0004 lux) and flexible integration with external IMUs for demanding applications on moving platforms.

Astronomy Cameras: Dedicated cameras like the ZWO ASI585MC offer cooled sensors and exceptional sensitivity but require external IMU integration for motion compensation.

3. Commercial Star Tracker Analysis

3.1 Market Overview

The star tracker market is segmented by application, accuracy, and form factor:

Segment	Typical Cost	Accuracy	Primary Users
Spacecraft Grade	\$100,000-\$500,000	1-10 arcsec	Space agencies, satellite manufacturers
CubeSat Grade	\$20,000-\$100,000	10-60 arcsec	University research, small satellites
Commercial Astronomy	\$5,000-\$30,000	1-5 arcmin	Professional observatories
Amateur Grade	\$200-\$2,000	5-30 arcmin	Amateur astronomers

3.2 Commercial Products Analysis

3.2.1 Spacecraft Star Trackers

Ball Aerospace CT-2020 - Accuracy: 2 arcsec (pitch/yaw), 15 arcsec (roll) - Update rate: 10 Hz - Mass: 2.5 kg - Power: 10 W - Cost: ~\$300,000

Sodern Hydra - Accuracy: 1 arcsec (pitch/yaw) - Update rate: 4 Hz - Mass: 2.3 kg - Cost: ~\$400,000

Terma T1/T2 - Accuracy: 5 arcsec - Mass: 0.5-1.5 kg - Cost: ~\$150,000

3.2.2 CubeSat Star Trackers

Blue Canyon Technologies NST - Accuracy: 6 arcsec cross-boresight - Mass: 350g - Power: 1.5 W - Cost: ~\$75,000

Berlin Space Technologies ST200 - Accuracy: 30 arcsec - Mass: 250g - Cost: ~\$40,000

Sinclair Interplanetary ST-16RT2 - Accuracy: 2-7 arcsec - Mass: 185g - Cost: ~\$50,000

3.2.3 Amateur Astronomy Mounts

Sky-Watcher Star Adventurer 2i - Type: Portable equatorial mount - Tracking accuracy: ±5 arcmin/hr - Payload: 5 kg - Cost: ~\$400

iOptron SkyGuider Pro - Tracking accuracy: ±3.5 arcmin/hr - Payload: 5 kg - Cost: ~\$500

Celestron CGEM II - Tracking accuracy: ±3 arcmin RMS - Payload: 18 kg - Cost: ~\$2,000

3.3 Cost Breakdown Analysis

Commercial star tracker costs are driven by:

Component	% of Total Cost	Reason
Optics	15-25%	Precision ground lenses, low distortion
Sensor	10-20%	Space-qualified, radiation-hardened CCDs
Processing	10-15%	Rad-hard FPGAs/processors
Calibration	20-30%	Extensive ground testing, thermal-vacuum
Qualification	15-25%	Space environment testing
Development	10-15%	R&D amortization

3.4 Barrier to Entry

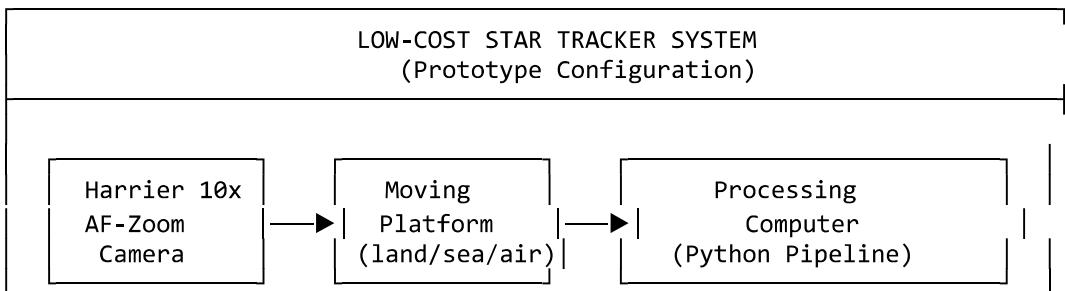
The high costs of commercial star trackers stem from:

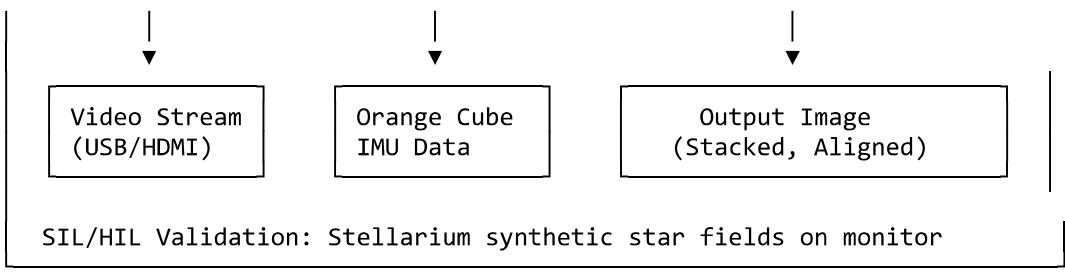
1. **Radiation Hardening:** Space-grade components must withstand cosmic radiation
2. **Thermal Stability:** Wide operating temperature ranges (-40°C to +60°C)
3. **Reliability Requirements:** Mean Time Between Failures (MTBF) > 100,000 hours
4. **Calibration Costs:** Each unit requires individual calibration
5. **Low Volume Production:** Limited market size prevents economies of scale

4. System Architecture

4.1 Overview

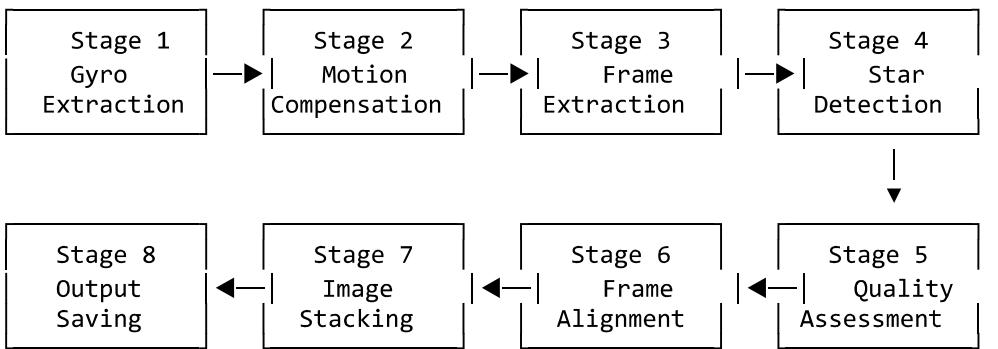
Our low-cost star tracker employs a software-intensive architecture that maximizes the use of consumer hardware while implementing sophisticated algorithms in software.





4.2 Processing Pipeline

The system processes data through eight sequential stages:



4.3 Data Flow

Stage	Input	Output	Key Algorithm
1. Gyro Extraction	MP4 video file	GyroData (quaternions)	GPMF parsing, RK4 integration
2. Motion Compensation	Frames + quaternions	Stabilized frames	Homography transformation
3. Frame Extraction	Video stream	Individual frames	FFmpeg/OpenCV decode
4. Star Detection	Frame images	StarField objects	Connected component analysis
5. Quality Assessment	StarFields	Quality scores	Multi-factor scoring
6. Frame Alignment	Quality frames	Aligned frames	Triangle matching + RANSAC
7. Image Stacking	Aligned frames	Stacked image	Sigma-clipping
8. Output Saving	Stacked image	TIFF/FITS file	16-bit encoding

4.4 Software Architecture

```

lowcost-star-tracker/
├── pyproject.toml
└── README.md

└── Root-level scripts/
    ├── compare_stabilization.py
    ├── compare_videos.py
    ├── convert_md_to_docx.py
    ├── debug_stabilization.py
    ├── gyro_stabilizer.py
    ├── live_simple_star_solve.py
    ├── live_tetra3_solve.py
    ├── plot_gyro.py
    └── stabilize_video.py

# Project configuration
# Main documentation

# Stabilization comparison tool
# Video comparison utility
# Documentation converter
# Stabilization debugging
# Gyroscope stabilization
# Simple star solving (live)
# Tetra3 star solving (live)
# Gyroscope data visualization
# Video stabilization tool

```

```

├── stellarium_config.py      # Stellarium configuration
├── stellarium_shake.py      # Stellarium shake simulator
├── stellarium_toggle_labels.py # Stellarium UI control
└── test_witmotion_pywitmotion.py # IMU testing

src/
├── algorithms/              # Core algorithms
├── calibration/             # Calibration modules
├── plate_solving/           # Plate solving algorithms
└── star_tracker/            # Main star tracker package

camera/
├── celestial_sphere_3d.py   # Camera and visualization tools
├── celestial_sphere_viewer.py # 3D celestial sphere visualization
├── integrated_stabilizer.py # Celestial sphere viewer
└── usb_camera_viewer.py     # Integrated stabilization system
                            # USB camera interface

calibration/                 # Calibration data and scripts

imu/
├── __init__.py               # IMU integration
├── find_witmotion_windows.py # IMU package initialization
├── pywitmotion_adapter.py    # Windows IMU detection
└── witmotion_reader.py       # Pywitmotion adapter
                            # Witmotion IMU reader

mavlink/
└── orange_cube_reader.py     # MAVLink integration
                            # Orange Cube flight controller

wfb-stabilizer/
├── README.md                # WFB stabilizer variants
├── ejo_wfb_stabilizer.py    # WFB documentation
└── run_camera1_*.py          # EJO WFB stabilizer
                            # Various camera stabilizer configs

validation/
├── VALIDATION_REPORT.md     # Validation framework
├── generate_validation_plots.py # Validation report
└── validation_framework.py   # Plot generation
                            # Validation framework

experiments/
└── motion_deblur/           # Experimental code
                            # Motion deblur algorithms
└── output/                  # Output files

external/
└── pywitmotion/             # External dependencies
    └── tetra3/                # Pywitmotion library
                                # Tetra3 star matching

data/
└── lens_profiles/           # Camera calibration profiles

examples/                    # Example videos and data

docs/
└── Development_Roadmap.md   # Documentation
    └── LowCost_StarTracker_Technical_Paper.md # Project roadmap
        └── images/                         # Technical paper
                                        # Documentation images

```

5. Hardware Components

Hardware Configurations:

Prototype System (validated): - **Camera:** Active Silicon Harrier 10x AF-Zoom (1080p60, 0.0004 lux sensitivity) - **IMU:** Orange Cube flight controller with ICM-20948 9-axis IMU (MAVLink

telemetry) - **Validation:** Stellarium planetarium software generating synthetic star fields - **Target**

Platforms: Seaborne, airborne, and land-based vehicles

Alternative Configurations (software-compatible): - **Configuration A:** GoPro Hero 7 Black (embedded gyroscope, GPMF telemetry) - **Configuration B:** ZWO ASI585MC + Entaniya M12 220 + external IMU (all-sky monitoring)

5.1 Prototype Camera: Active Silicon Harrier 10x AF-Zoom

The Active Silicon Harrier 10x AF-Zoom camera serves as the primary sensor platform for our validated prototype, selected for its exceptional low-light performance and flexible integration capabilities for moving platform applications.

5.1.1 Imaging Specifications

Parameter	Value
Sensor Type	1/2.8" Sony IMX462LQR-C CMOS (progressive scan)
Effective Pixels	~2.13 MP
Horizontal Resolution	700 TV lines
Optical Zoom	10x (f=5.1mm to 51mm, F1.6 to 1.8)
Digital Zoom	2x to 32x
Field of View	57° (wide) to 6.2° (telephoto)
Video Output	1080p60, 1080p30, 720p60, 720p30
Interfaces	Simultaneous USB 3.1 Gen1 (UVC) + HDMI
Minimum Illumination	0.003 lux (color), 0.0004 lux (mono)
Shutter Speed	1/30 to 1/30,000 sec
S/N Ratio	>50 dB

5.1.2 Physical Characteristics

Parameter	Value
Dimensions	62.4mm × 56mm × 45.5mm
Weight	138g
Power	9-15V DC, ~4.8W @ 1080p60
Control Protocols	UVC, VISCA (USB/TTL/RS-232), Pelco D/P
Minimum Object Distance	100mm to 1000mm

5.1.3 Advantages for Star Tracking on Moving Platforms

- **Exceptional low-light sensitivity:** 0.0004 lux enables star detection even in challenging conditions
- **10x optical zoom:** Allows flexible field-of-view selection without lens changes
- **Fast autofocus:** Maintains sharp focus during platform motion
- **Dual output:** Simultaneous USB and HDMI enables recording + live preview
- **Compact form factor:** Suitable for integration on vehicles, drones, and marine vessels
- **Industrial reliability:** Designed for demanding machine vision applications

5.2 Prototype IMU: Orange Cube Flight Controller

The Orange Cube flight controller provides high-rate IMU data via MAVLink protocol, enabling precise motion compensation.

5.2.1 IMU Specifications (ICM-20948)

Parameter	Value
Type	9-axis MEMS (3-axis gyro + 3-axis accel + 3-axis mag)
Gyroscope Range	$\pm 250/500/1000/2000 \text{ } ^\circ/\text{s}$ (selectable)
Gyroscope Resolution	16-bit
Gyroscope Noise	0.015 $\text{ } ^\circ/\text{s}/\sqrt{\text{Hz}}$
Accelerometer Range	$\pm 2/4/8/16 \text{ g}$ (selectable)
Sampling Rate	Up to 1 kHz (gyro), 4.5 kHz (accel)
Interface	MAVLink over USB/serial

5.2.2 Sensor Fusion

The system employs VQF (Versatile Quaternion-based Filter) for sensor fusion:

- Combines gyroscope and accelerometer data
- Magnetometer optional (disabled for moving platform use due to interference)
- Outputs orientation quaternions at IMU sample rate

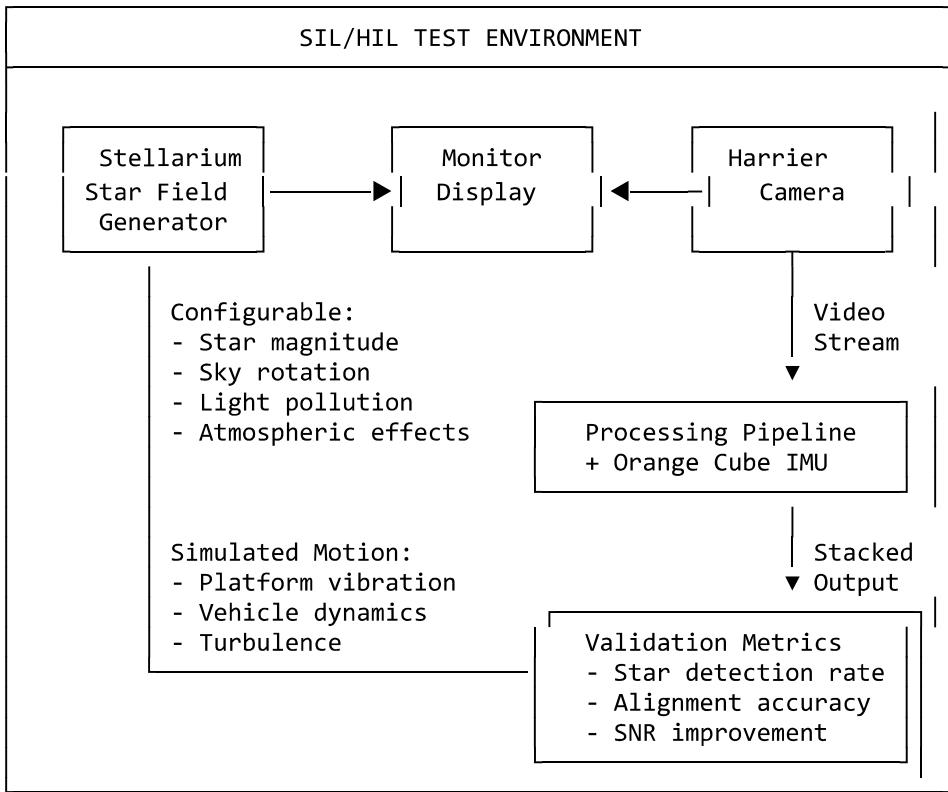
5.3 SIL/HIL Validation Environment: Stellarium

The prototype is validated using Stellarium planetarium software to generate controlled test conditions.



Click thumbnail to watch Stellarium Software-in-the-Loop demonstration

5.3.1 Test Setup



5.3.2 Validation Capabilities

- **Controlled star fields:** Known star positions for accuracy verification
- **Configurable light pollution:** Test performance in Bortle 1-9 conditions
- **Motion injection:** Simulate platform dynamics (vibration, drift, turbulence)
- **Repeatability:** Identical test conditions for algorithm comparison
- **Ground truth:** Perfect reference for quantitative performance metrics

5.4 Alternative Configuration A: GoPro Hero 7 Black

The GoPro Hero 7 Black provides an accessible consumer option with integrated IMU, suitable for portable astrophotography applications.

5.4.1 Imaging Specifications

Parameter	Value
Sensor Type	1/2.3" CMOS
Resolution	12 MP (4000×3000 pixels)
Pixel Size	1.55 μm
Lens	Fixed f/2.8, 14-17mm equivalent
Field of View	Wide: 122.6°, Linear: 94.4°
Video Modes	4K60, 2.7K120, 1080p240
Photo Modes	RAW, HDR, Night
ISO Range	100-6400 (extended: 100-12800)
Shutter Speed	1/8000s - 30s (photo), 1/fps (video)

5.4.2 Embedded Gyroscope

Parameter	Value
Type	3-axis MEMS gyroscope
Sampling Rate	200 Hz (configurable to 400 Hz)
Range	$\pm 2000 \text{ } ^\circ/\text{s}$
Resolution	16-bit
Data Format	GPMF (GoPro Metadata Format)
Synchronization	Frame-accurate timestamps

5.4.3 Astrophotography Settings (Protune Mode)

For optimal night sky capture: - **ISO Min/Max:** 800/6400 - **Shutter:** 1/30s (30fps) or 1/24s (24fps) - **White Balance:** 5500K (Native) - **Color Profile:** Flat - **Sharpness:** Low - **Lens Mode:** Linear (recommended for stacking)

5.4.4 GoPro Camera Intrinsic Parameters

Calibrated intrinsic matrix for GoPro Hero 7 Black (Linear mode):

$$K = [fx \quad 0 \quad cx] = [3200 \quad 0 \quad 2000] \\ [0 \quad fy \quad cy] = [0 \quad 3200 \quad 1500] \\ [0 \quad 0 \quad 1] \quad [0 \quad 0 \quad 1]$$

Where: - fx, fy = Focal lengths in pixels (~3200) - cx, cy = Principal point (image center)

Distortion coefficients (Brown-Conrady model): - k1 = -0.25 (radial) - k2 = 0.08 (radial) - p1, p2 ≈ 0 (tangential, negligible)

5.5 Alternative Configuration B: ZWO ASI585MC + Entaniya M12 220

For applications requiring higher sensitivity, larger field of view, or all-sky coverage, we provide an enhanced hardware configuration using dedicated astronomy equipment.

5.5.1 ZWO ASI585MC Camera

The ZWO ASI585MC is a high-performance color astronomy camera based on Sony's latest STARVIS 2 technology, offering exceptional sensitivity and low noise characteristics ideal for star tracking and meteor detection.

Sensor Specifications:

Parameter	Value
Sensor Model	Sony IMX585 (STARVIS 2)
Sensor Type	Back-Illuminated CMOS
Sensor Format	1/1.2" (larger than GoPro's 1/2.3")
Resolution	3840 × 2160 (8.29 MP)
Pixel Size	2.9 μm × 2.9 μm
Sensor Dimensions	11.13 mm × 6.26 mm

Parameter	Value
Sensor Diagonal	12.84 mm
Quantum Efficiency (QE)	91% peak (exceptional)
Full Well Capacity	40,000 e ⁻ (3× previous generation)
Read Noise	0.8 e ⁻ (extremely low)
ADC	12-bit
Frame Rate	46.9 FPS at full resolution (12-bit)
Interface	USB 3.0
Amp Glow	Zero (critical for long exposures)

Key Advantages over GoPro:

Feature	ASI585MC	GoPro Hero 7 Black
Sensor Size	1/1.2" (12.84mm diag)	1/2.3" (6.17mm diag)
Pixel Size	2.9 μm	1.55 μm
QE Peak	91%	~50% (estimated)
Read Noise	0.8 e ⁻	~3-5 e ⁻ (estimated)
Full Well	40,000 e ⁻	~8,000 e ⁻ (estimated)
Amp Glow	Zero	Present in long exposures
Cooling (Pro)	-35°C below ambient	None
Raw Output	12-bit uncompressed	10-bit compressed

High Conversion Gain (HCG) Mode:

The ASI585MC features automatic HCG mode activation: - Triggers at gain ≥ 252 (standard) or ≥ 200 (Pro version) - Maintains ~12-bit dynamic range at high gain - Read noise drops to 0.7-1.5 e⁻ - Optimal for faint star detection

Spectral Response:

The ASI585MC has enhanced sensitivity in red, green, and near-infrared (NIR) wavelengths: - Particularly strong in >850nm range - 1.5× more sensitive than previous IMX485 sensor in NIR - Excellent for hydrogen-alpha (H α) nebula detection

5.5.2 Entaniya Fisheye M12 220 Lens

The Entaniya Fisheye M12 220 is a precision Japanese-manufactured super wide-angle lens designed for full-hemisphere imaging applications.

Optical Specifications:

Parameter	Value
Field of View	220° (full hemisphere + below horizon)
Focal Length	1.34 mm
Aperture	f/2.0 (fixed)
Optical Formula	10 elements (including 1× ED glass)
Projection Type	Equidistant (equal angular distortion)
Image Circle	5.1 mm diameter
Sensor Compatibility	1/2.3" to 1/1.7" sensors

Parameter	Value
Mount Type	M12 × P0.5 (S-mount)
Weight	52g (without caps)
AR Coating	Yes (multi-coated)
Resolution	4K+ edge-to-edge
Manufacturing	Made in Japan

Included Accessories: - CS-mount adapter (for CS-mount cameras) - IR cut filter (removable for full-spectrum)
- Allen wrench and mounting bolts

Equidistant Projection Characteristics:

The equidistant (f-theta) projection maintains linear relationship between incident angle and image distance:

$$r = f \times \theta$$

Where:

r = radial distance from image center (mm)

f = focal length (1.34 mm)

θ = incident angle from optical axis (radians)

This projection type is ideal for: - Meteor trajectory measurement - All-sky photometry - Satellite tracking - Aurora monitoring

Distortion Model:

For the Entaniya M12 220 with equidistant projection:

```

θ_undistorted = r / f
x = r × cos(ϕ)
y = r × sin(ϕ)

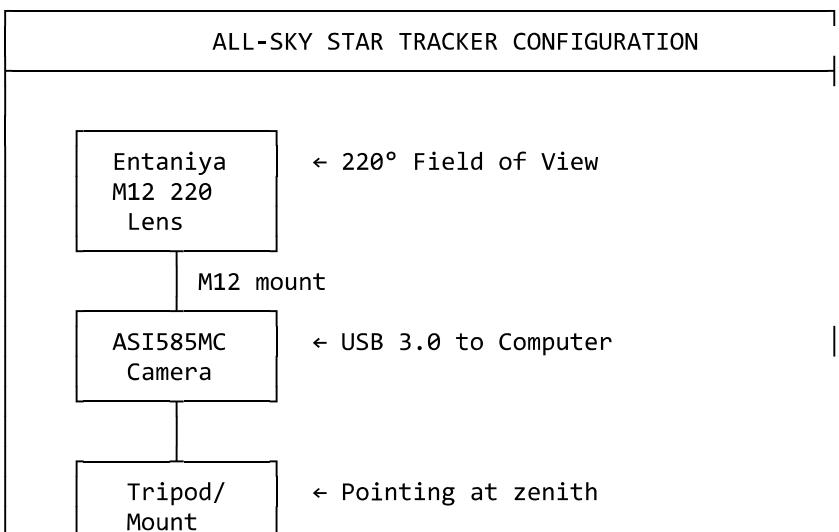
```

Where ϕ = azimuthal angle in image plane

Residual distortion from ideal equidistant: < 1% across full field

5.5.3 ASI585MC + Entaniya System Integration

Physical Setup:



Camera Intrinsic Parameters (ASI585MC + Entaniya M12 220):

Focal length: $f = 1.34 \text{ mm}$

Pixel size: $p = 2.9 \mu\text{m}$

Focal length in pixels: $f_{\text{px}} = f / p = 1340 / 2.9 \approx 462 \text{ pixels}$

$$K = [f_{\text{px}} \quad 0 \quad cx] = [462 \quad 0 \quad 1920]$$
$$[0 \quad f_{\text{px}} \quad cy] = [0 \quad 462 \quad 1080]$$
$$[0 \quad 0 \quad 1] = [0 \quad 0 \quad 1]$$

Image center: (1920, 1080) for 3840×2160 resolution

Angular Resolution:

$$\text{Pixel scale} = \arctan(\text{pixel_size} / \text{focal_length})$$
$$= \arctan(2.9 \mu\text{m} / 1.34 \text{ mm})$$
$$= \arctan(0.00216)$$
$$\approx 0.124^\circ = 7.44 \text{ arcmin/pixel}$$

For 220° FOV across ~1700 pixel radius:

Effective resolution: ~7.7 arcmin/pixel at image center

Comparison: GoPro vs ASI585MC+Entaniya:

Parameter	GoPro Hero 7 (Linear)	ASI585MC + Entaniya
Field of View	94.4°	220° (full sky)
Resolution	4000 × 3000	3840 × 2160
Pixel Scale	~1.8 arcmin/pixel	~7.4 arcmin/pixel
Light Sensitivity	Moderate	Excellent (91% QE)
Gyroscope	Built-in 200 Hz	External required
Use Case	Targeted fields	All-sky monitoring
Cost	\$200-400	\$450-600

5.5.4 Recommended Settings for ASI585MC

For Star Tracking / All-Sky Imaging:

Setting	Value	Rationale
Gain	200-300	HCG mode active, low noise
Exposure	1-10 seconds	Balance sensitivity vs. trailing
Binning	1×1	Full resolution
Format	RAW16	Maximum dynamic range
USB Traffic	80-100	Balance speed vs. stability
Cooling (Pro)	-10°C to -20°C	Reduce thermal noise

For Meteor Detection:

Setting	Value	Rationale
Gain	300-350	Maximum sensitivity
Exposure	1/30s (video)	Capture fast-moving objects

Setting	Value	Rationale
Frame Rate	30 FPS	Adequate temporal resolution
Format	SER video	Efficient capture format
Trigger	Motion detection	Automated recording

5.5.5 All-Sky Applications

The ASI585MC + Entaniya M12 220 configuration excels for:

- 1. Meteor Shower Monitoring** - Full hemisphere coverage captures all meteor events - High QE detects faint meteors (magnitude 5-6) - Equidistant projection enables accurate trajectory reconstruction - High frame rates capture meteor dynamics
- 2. Satellite Tracking** - Simultaneous tracking of multiple satellites - LEO to GEO orbit coverage - Starlink constellation monitoring - Space debris tracking
- 3. Variable Star Photometry** - All-sky simultaneous observation - Rapid transient detection - Nova/supernova search - Exoplanet transit monitoring (bright stars)
- 4. Weather and Atmospheric Monitoring** - Cloud cover assessment - Light pollution mapping - Aurora monitoring - Aircraft tracking
- 5. Educational Demonstrations** - Real-time constellation identification - Celestial sphere concepts - Diurnal motion visualization - Seasonal sky changes

5.6 Hardware Configuration Comparison

Configuration	Cost	FOV	Sensitivity	Best Use Case
Harrier 10x + Orange Cube (Prototype)	\$800-1,200	6-57°	Exceptional (0.0004 lux)	Moving platforms, SIL/HIL validation
GoPro Hero 7 Black	\$200-400	94-122°	Moderate	Portable astrophotography, travel
ASI585MC + Entaniya	\$700-900	220°	Excellent	All-sky monitoring, meteor detection
ASI585MC Pro + Entaniya	\$900-1,100	220°	Exceptional	Long-exposure, scientific applications

5.7 Cost Breakdown by Configuration

Prototype: Harrier 10x + Orange Cube

Component	Cost (USD)
Active Silicon Harrier 10x AF-Zoom	\$500-700
Orange Cube Flight Controller	\$150-250
USB 3.0 Capture/Cabling	\$50-100
Processing Computer	\$100-200
Total	\$800-1,250

Alternative A: GoPro Hero 7 Black

Component	Cost (USD)
GoPro Hero 7 Black (used)	\$150-250
GoPro Hero 7 Black (new)	\$250-400
MicroSD Card (128GB+)	\$20-40
Processing Computer	\$100-200
Total (used)	\$270-490
Total (new)	\$370-640

Alternative B: ASI585MC + Entaniya M12 220

Component	Cost (USD)
ZWO ASI585MC Camera	\$319-379
ZWO ASI585MC Pro (cooled)	\$499-549
Entaniya M12 220 Lens	\$280-350
M12 to CS-mount Adapter	Included with lens
Weather-resistant Housing	\$50-100
USB 3.0 Cable (active, 5m)	\$25-40
Tripod/All-Sky Mount	\$50-150
Processing Computer	Existing or \$300+
Software	Free (open source)
Total (Standard)	\$724-1,019
Total (Pro/Cooled)	\$904-1,189

6. Camera + IMU Sensor Fusion

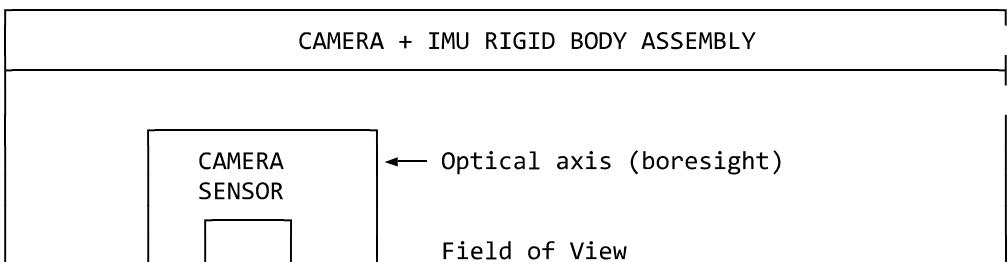
6.1 Fundamental Concept

The core innovation of our star tracker lies in the tight integration of camera imagery with Inertial Measurement Unit (IMU) orientation data. This sensor fusion approach enables real-time knowledge of where the camera is pointing in celestial coordinates, which is essential for:

1. **Motion Compensation** - Correcting for camera movement during exposure
2. **Star Field Prediction** - Knowing which stars should appear in the field of view
3. **Frame Alignment** - Precise registration of multiple frames for stacking
4. **Lost-in-Space Solution** - Rapid star identification using predicted positions

6.2 Physical Setup: Rigid Camera-IMU Coupling

For accurate sensor fusion, the camera and IMU must be rigidly coupled so that any rotation of the camera is identically measured by the IMU.



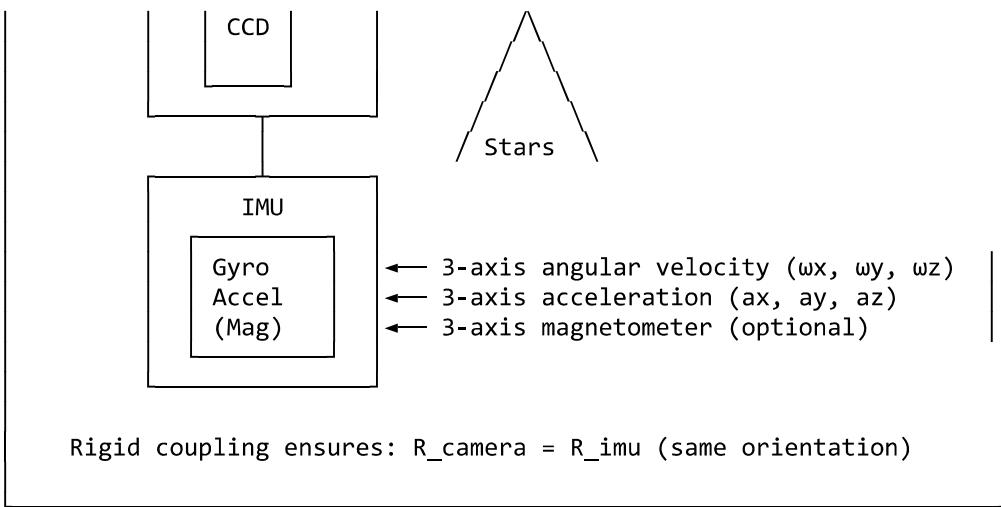


Figure 6.0: Prototype Hardware Assembly - Harrier 10x Camera + Orange Cube IMU



The prototype assembly uses the Active Silicon Harrier 10x AF-Zoom camera rigidly mounted on the Orange Cube flight controller via a 3D-printed bracket (orange). The front features the 10x optical zoom lens ($f=5.1-51\text{mm}$); the rear shows the AS-CIB-USBHDMI-002-A interface board (S/N: 71000300) and the ProfiCNC/HEX/uAvionix Orange Cube with visible I/O connectors (GPS, I2C, USB, CAN, TELEM, ADC).

Implementation Options:

Configuration	IMU Source	Coupling	Accuracy
Harrier 10x + Orange Cube (Prototype)	ICM-20948 9-axis	Rigid bracket	High (calibrated)
GoPro Hero 7	Built-in BMI260	Factory-calibrated	High
External IMU + Camera	BNO055, MPU9250	Custom mount	Requires calibration
ASI585MC + External IMU	Separate 9-DOF	Rigid bracket	Requires calibration

Figure 6.1: Real-Time Stacking Output - Faint Star Recovery



The progression demonstrates faint star recovery through real-time stacking: (Left) Single noisy frame with one visible bright star; (Center) After stacking multiple frames, a second fainter star becomes visible; (Right) Enhanced output with reduced noise and improved star visibility. The green rectangle indicates the template-locked region of interest used for frame alignment.

6.3 Coordinate Systems and Transformations

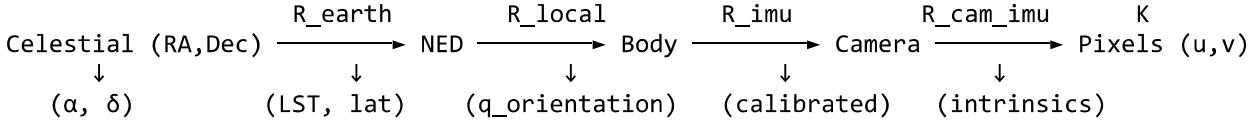
6.3.1 Reference Frames

Four coordinate systems are involved in the sensor fusion:

1. IMU Body Frame (B)
 - Origin: IMU center
 - X: Forward (camera boresight direction)
 - Y: Right
 - Z: Down
2. Camera Frame (C)
 - Origin: Camera optical center
 - X: Right (image +u direction)
 - Y: Down (image +v direction)
 - Z: Forward (optical axis)
3. Local Navigation Frame (N) - North-East-Down (NED)
 - X: North
 - Y: East
 - Z: Down (gravity direction)
4. Celestial Frame (ICRS/J2000)
 - Origin: Solar system barycenter
 - X: Vernal equinox direction
 - Y: 90° east along celestial equator
 - Z: Celestial north pole

6.3.2 Transformation Chain

To map a celestial coordinate to image pixels:



Mathematical Formulation:

```

p_celestial = [cos(δ)cos(α), cos(δ)sin(α), sin(δ)]T      # Unit vector to star
p_local = R_earth(LST, latitude) · p_celestial                # Transform to local frame
p_body = R_imu · p_local                                      # Apply IMU orientation
p_camera = R_cam_imu · p_body                                  # IMU-to-camera alignment
p_image = K · p_camera / p_camera[2]                          # Project to image plane
    
```

Where:

- LST = Local Sidereal Time
- R_{imu} = Rotation matrix from IMU quaternion
- $R_{\text{cam_imu}}$ = Camera-IMU alignment matrix (calibrated)
- K = Camera intrinsic matrix

6.4 3D Celestial Sphere Visualization

Our system includes a real-time 3D Celestial Sphere Viewer that visualizes the camera's orientation relative to the star field. This tool is invaluable for:

- **Debugging** sensor fusion algorithms
- **Verifying** IMU-camera alignment
- **Educational** demonstration of celestial mechanics
- **Real-time monitoring** of pointing direction

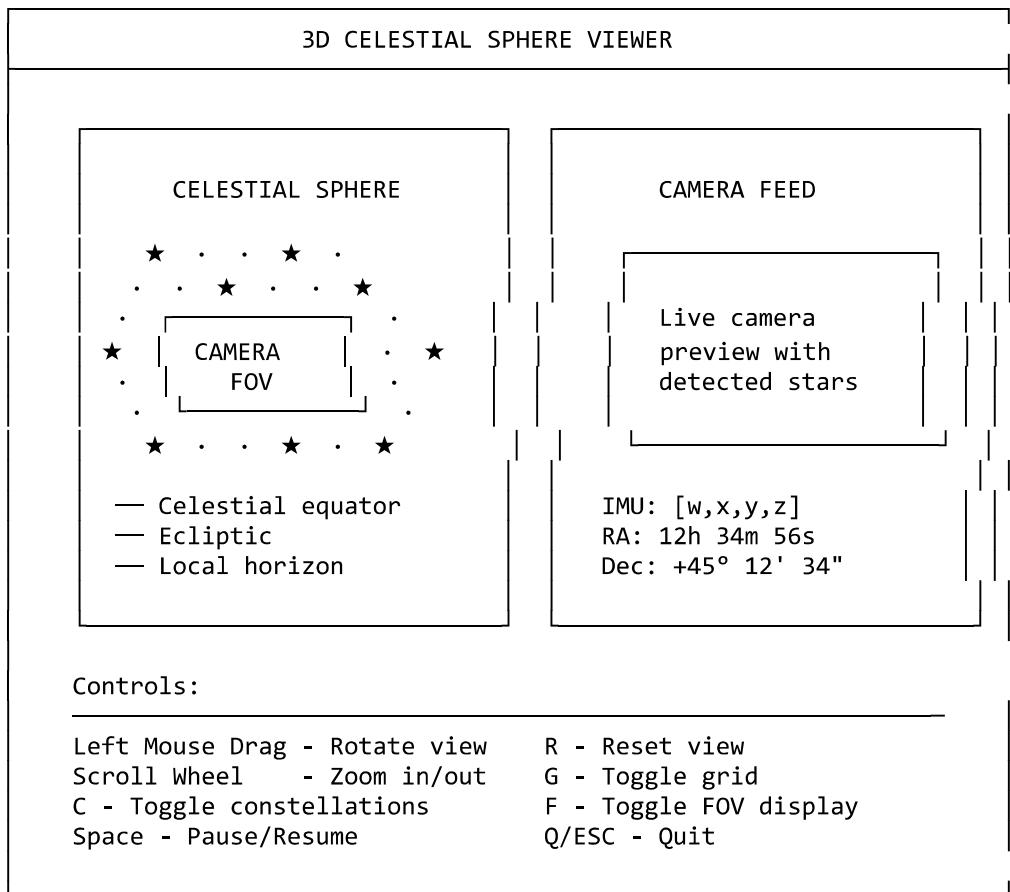
Figure 1: 3D Celestial Sphere Viewer with Camera + IMU Setup

 3D Celestial Sphere Viewer showing camera FOV projected onto star field, with live camera feed from rigidly coupled camera and IMU (Orange Cube)

3D Celestial Sphere Viewer showing camera FOV projected onto star field, with live camera feed from rigidly coupled camera and IMU (Orange Cube)

The screenshot shows the 3D Celestial Sphere Viewer (left) displaying the camera's field of view (green trapezoid) projected onto the celestial sphere with stars. The camera feed (right) shows the physical setup with the camera and Orange Cube IMU held together, demonstrating the rigid coupling required for accurate sensor fusion.

6.4.1 Visualization Components



6.4.2 Camera Field of View Projection

The camera's field of view is projected onto the celestial sphere as a quadrilateral (frustum intersection):

Given camera orientation quaternion q and intrinsic matrix K :

1. Define image corners: $(0,0), (W,0), (W,H), (0,H)$

2. Back-project to unit rays:

$$\begin{aligned} \text{ray_i} &= K^{-1} \cdot [u_i, v_i, 1]^T \\ \text{ray_i} &= \text{ray_i} / \|\text{ray_i}\| \end{aligned}$$

3. Rotate to celestial frame:

$$\text{celestial_ray_i} = R(q)^T \cdot R_{\text{earth}}(t)^T \cdot \text{ray_i}$$

4. Convert to spherical coordinates:

$$RA_i = \text{atan2}(y_i, x_i)$$

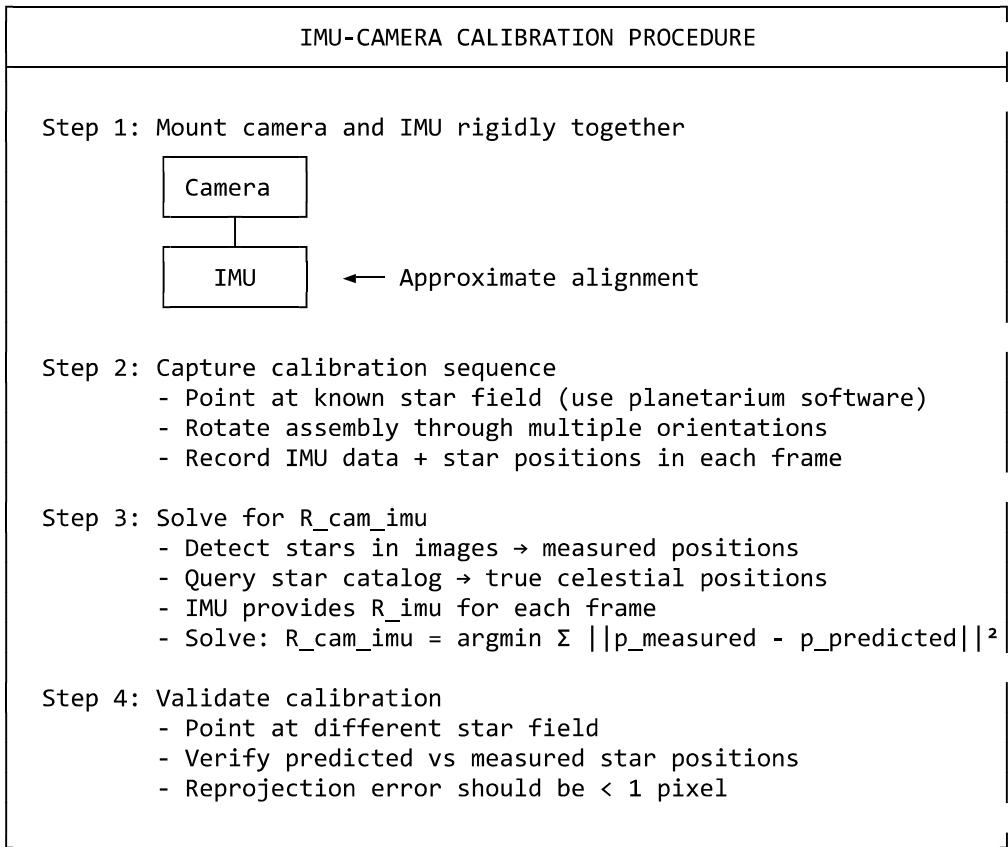
```
Dec_i = asin(z_i)
```

5. Draw FOV polygon on celestial sphere connecting the 4 corners

6.5 IMU-Camera Calibration

For systems with external IMUs, the relative orientation between IMU and camera must be calibrated.

6.5.1 Calibration Procedure



6.5.2 Calibration Matrix

The IMU-to-camera rotation is typically a small correction:

$$R_{\text{cam_imu}} = R_z(\psi) \cdot R_y(\theta) \cdot R_x(\phi)$$

Where:

- ϕ (roll): Rotation about optical axis
- θ (pitch): Rotation about camera Y axis
- ψ (yaw): Rotation about camera Z axis

For Harrier + Orange Cube (Prototype): $R_{\text{cam_imu}}$ calibrated via rigid bracket alignment

For GoPro with built-in IMU: $R_{\text{cam_imu}} \approx I$ (identity, factory-calibrated)

For external IMU setups: Typically $|\phi|, |\theta|, |\psi| < 5^\circ$ after careful mounting

6.6 Sensor Fusion Algorithms

6.6.1 Gyroscope-Only Integration

The simplest approach integrates gyroscope angular velocities:

```
def integrate_gyro(gyro_data, dt):
    """Dead-reckoning orientation from gyroscope."""
    q = Quaternion(1, 0, 0, 0) # Identity (initial orientation)

    for omega in gyro_data:
        # Quaternion derivative:  $dq/dt = 0.5 * q \otimes [\theta, \omega]$ 
        omega_quat = Quaternion(0, omega[0], omega[1], omega[2])
        q_dot = 0.5 * q * omega_quat

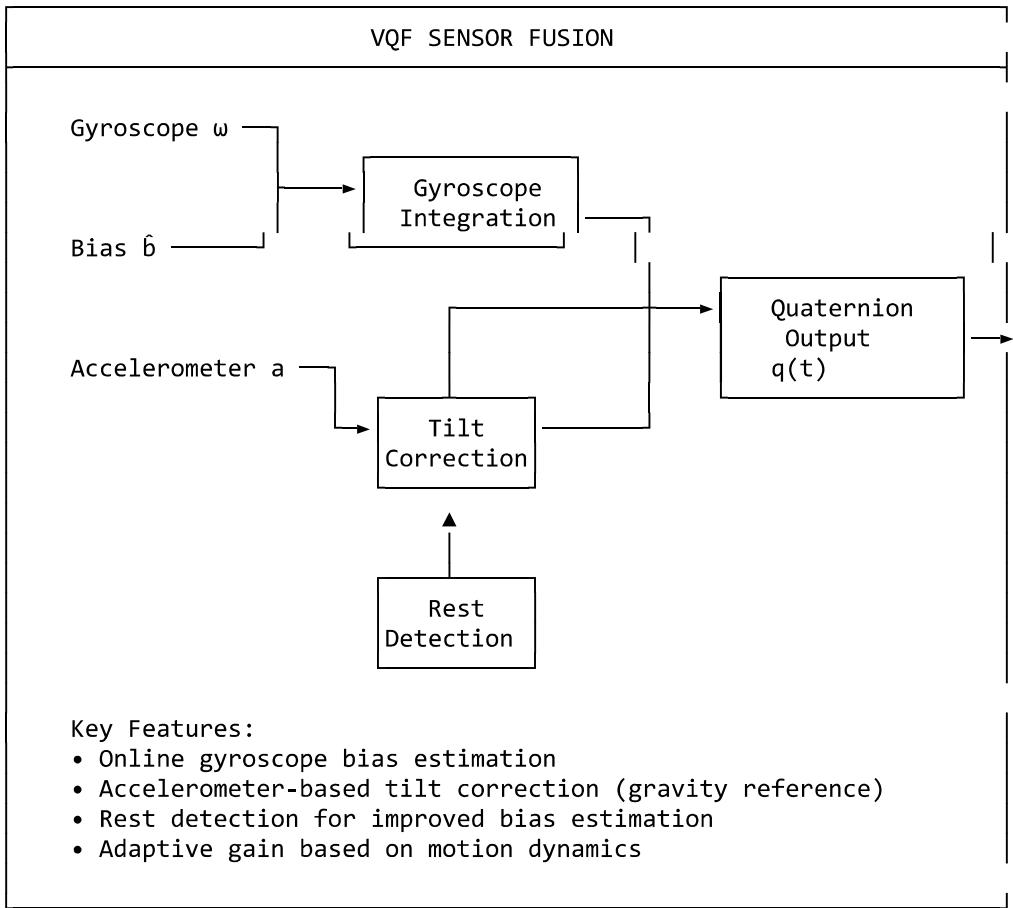
        # Integrate
        q = q + q_dot * dt
        q = q.normalized()

    return q
```

Limitations: - Gyroscope bias causes drift ($\sim 1\text{-}10^\circ/\text{hour}$) - No absolute reference (only relative orientation) - Accumulating errors over time

6.6.2 VQF Sensor Fusion (Recommended)

The Versatile Quaternion-based Filter (VQF) combines gyroscope and accelerometer for drift-free orientation:



VQF Algorithm Steps:

1. Gyroscope Integration:

$$q_{\text{gyro}}(t) = q(t-1) \otimes \exp([\theta, (\omega - \hat{b}) \cdot dt / 2])$$

2. Tilt Correction (from accelerometer):

```
g_measured = R(q_gyro) . [0, 0, 1]ᵀ # Expected gravity in body frame
g_actual = normalize(accelerometer) # Measured gravity

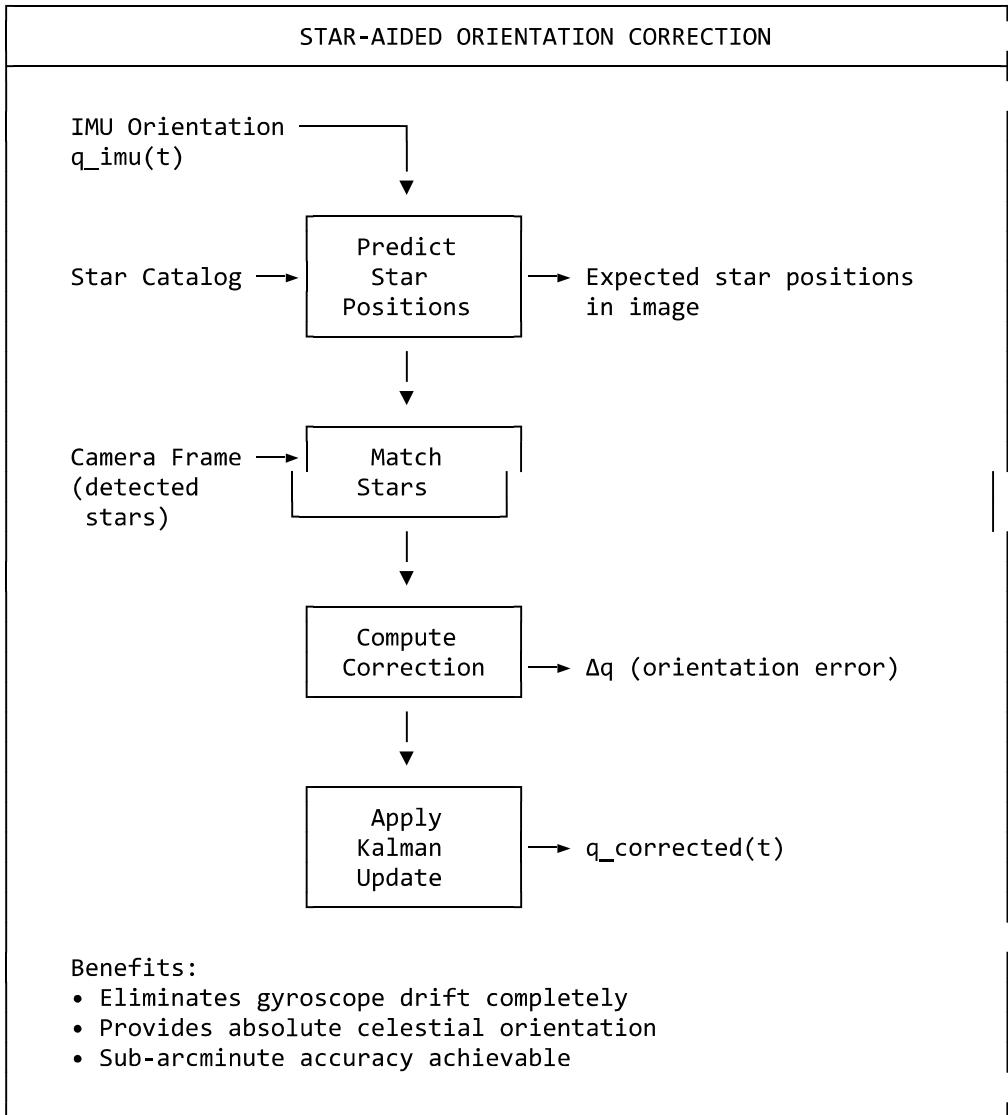
# Correction quaternion
q_correction = quaternion_from_vectors(g_measured, g_actual)
q(t) = slerp(q_gyro, q_gyro ⊗ q_correction, α)
```

3. Bias Estimation (during rest):

```
if is_stationary(accelerometer):
    b̂ = b̂ + β · (ω - b̂) # Update bias estimate
```

6.6.3 Star-Aided Correction

For highest accuracy, detected stars provide absolute orientation correction:



6.7 Benefits for Star Tracking

The camera+IMU sensor fusion provides critical advantages:

Capability	Without IMU	With IMU Fusion
Motion compensation	Optical flow (slow)	Direct quaternion transform (fast)
Frame alignment	Feature matching	Predicted + refined
Star identification	Lost-in-space search	Catalog lookup by position
Exposure during motion	Blurred stars	Sharp (compensated)
Real-time preview	Not possible	Live celestial overlay
Processing speed	1-5 fps	30+ fps

6.8 Video Demonstration

A demonstration of the camera+IMU fusion system is available:

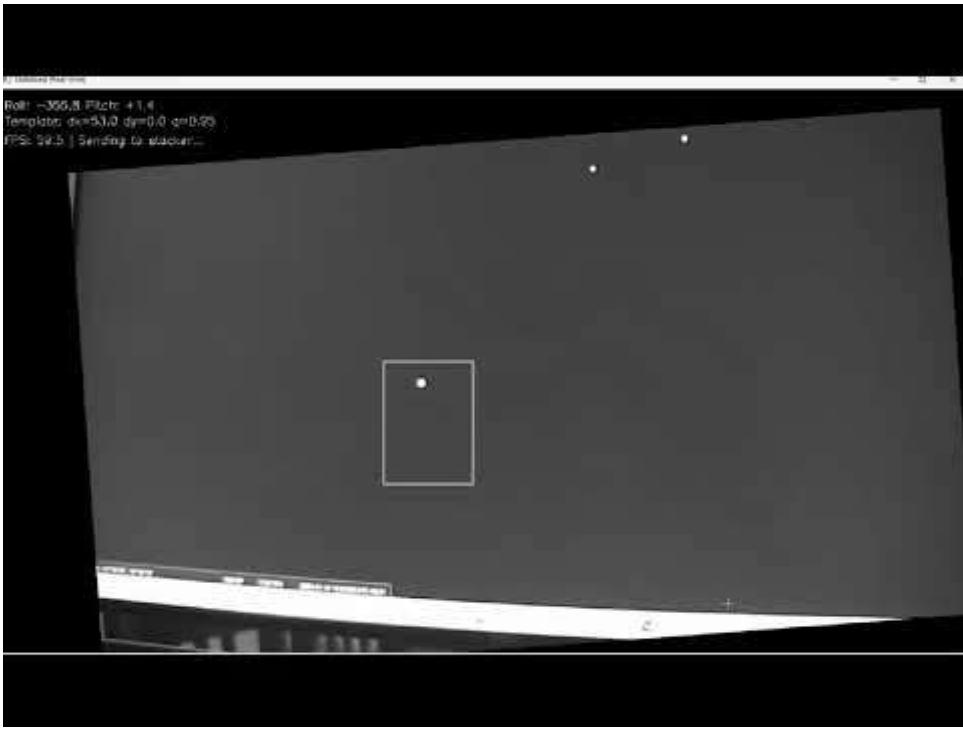


Click thumbnail to watch Camera + IMU Sensor Fusion demonstration

The video shows: 1. Physical setup with camera and IMU (orange cube) rigidly coupled 2. Real-time 3D Celestial Sphere Viewer tracking camera orientation 3. Camera field of view projected onto the star field 4. Responsive orientation updates as the assembly is rotated

6.9 Hybrid Stabilization: Gyroscope + Template Matching

A significant advancement in our stabilization approach combines gyroscope-based compensation with template matching refinement, achieving superior performance compared to either method alone.



Click thumbnail to watch hybrid real-time video stabilization and ROI stacking demonstration

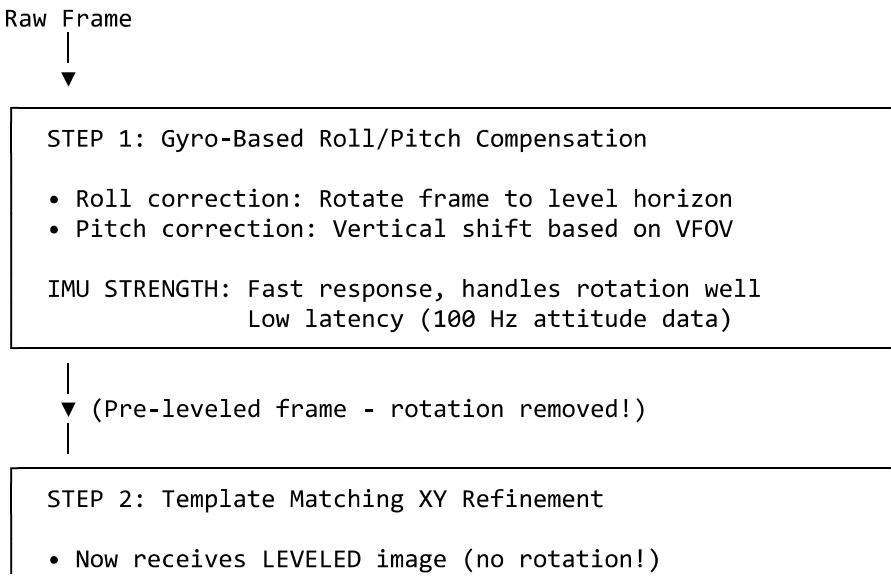
6.9.1 The Problem with Single-Method Approaches

Pure Template Matching Limitations: - **Roll sensitivity:** Template matching is highly susceptible to rotation. Even small roll angles cause the template to fail matching because the pattern rotates. - **Large search windows needed:** Without prior motion estimation, template matching must search a large area, which is computationally expensive, prone to false matches, and unable to handle fast motion.

Pure Gyro Stabilization Limitations: - **Drift over time:** IMU integration accumulates errors - **No absolute reference:** Cannot correct if the initial frame was not level - **Sensor noise:** High-frequency jitter passes through - **Mounting offsets:** Calibration between camera and IMU axes can introduce errors

6.9.2 The Hybrid Solution

By applying IMU gyro compensation FIRST, then applying template matching for refinement:



- Can use **SMALL** search window (gyro already compensated)
- Only needs to find XY translation residual

TEMPLATE STRENGTH: Sub-pixel precision, drift correction
Absolute reference to scene content

↓
▼ (Fully stabilized frame)

6.9.3 Why Sensor Fusion Works Well

1. Roll Removal Enables Template Matching

- Template matching is extremely sensitive to rotation
- By removing roll first via gyro, the template sees a consistently oriented image
- Match quality (correlation) stays high (>0.7) even during motion

2. Pitch Compensation Reduces Search Area

- Without pitch compensation, vertical motion requires large search margins
- Gyro-based pitch shift pre-positions the frame
- Template matching only needs to find small residual offsets
- Smaller search window = faster matching + fewer false positives

3. Complementary Strengths

Aspect	IMU Gyro	Template Matching
Speed	Fast (100 Hz)	Slower (per-frame)
Rotation handling	Excellent	Poor
Translation accuracy	Good (with VFOV scaling)	Excellent (sub-pixel)
Drift	Accumulates	None (absolute)
Latency	Very low	Moderate

4. Reduced Uncertainty

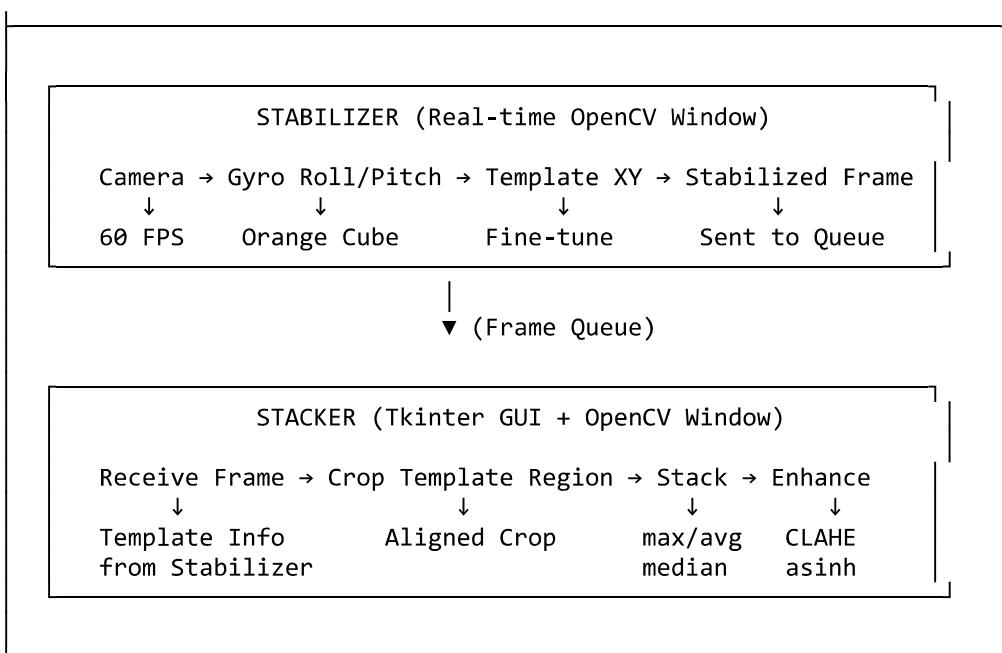
- Gyro minimizes the uncertainty space for template matching
- Instead of searching full frame, search ±200px around expected position
- Higher confidence matches, fewer outliers

6.9.4 Template Matching Success Rate

Configuration	Match Quality	Notes
Template only (no gyro)	0.3-0.5	Fails during roll
Gyro only (no template)	N/A	Drifts over time
Gyro + Template Hybrid	0.7-0.95	Stable even during motion

6.9.5 Implementation: Dual GUI System

The hybrid system is implemented as a dual-window application, demonstrated using:
- Camera: Harrier 10x AF Zoom (1280×720 @ 60 FPS)
- IMU: Orange Cube flight controller (MAVLINK ATTITUDE @ 100 Hz)



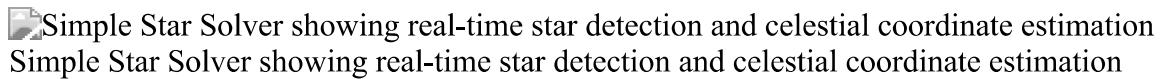
Stabilizer Window Features: - Real-time display at camera FPS (60 Hz) - Draw rectangle to set template region - Green box when match quality > 0.7, yellow otherwise - Keyboard controls: B = reset baseline, R = clear template, Q = quit

Stacker GUI Features: - Receives stabilized frames via thread-safe queue - Automatically uses template region from stabilizer - Adjustable stack count (2-100 frames), intensity multiplier (1-10 \times) - Multiple stack modes: max, average, sum, median - Adaptive enhancement options: CLAHE, asinh, log, sqrt stretches

6.10 Simple Star Solver (Proof of Concept)

The `live_simple_star_solve.py` script demonstrates real-time star pattern matching and celestial coordinate estimation using a simplified plate-solving approach.

Figure 6.2: Simple Star Solver - Live Plate Solving POC



The screenshot shows the Simple Star Solver tracking 50 detected stars with 100% confidence. The estimated RA/Dec coordinates are compared against Stellarium reference values, achieving ~5 arcminute accuracy at 60° FOV. Real-time performance: 18-22 FPS with 21.4 FPS processing rate.

6.10.1 Features

- **Real-time star detection:** Detects up to 50+ stars per frame at 20+ FPS
- **Coordinate estimation:** Estimates RA/Dec from detected star patterns
- **Stellarium validation:** Compares against Stellarium reference for accuracy verification
- **Confidence scoring:** Reports match quality (0-100%)
- **Error metrics:** Shows angular error in arcminutes/degrees

6.10.2 Performance Metrics

Metric	Value
Stars Detected	50

Metric	Value
Confidence	74-100%
Angular Error	~5 arcmin (RA), ~14 arcmin (Dec)
Processing Rate	18-22 FPS
FOV	60°

6.10.3 Usage

```
python live_simple_star_solve.py
```

The solver uses triangle pattern matching against a star catalog to identify the star field and compute the camera's pointing direction in celestial coordinates.

7. IMU-Based Motion Deblur

During long-exposure astrophotography, camera motion causes stars to appear as trails (motion blur) instead of points. Traditional solutions require expensive motorized tracking mounts. Our approach uses IMU attitude data to computationally reverse the motion blur, recovering sharp star images from blurred exposures.

7.1 The Motion Blur Problem

Motion blur in star field images occurs when the camera orientation changes during exposure. The resulting image is a convolution of the sharp star field with a Point Spread Function (PSF) that encodes the motion trajectory:

```
I_blurred(x,y) = I_sharp(x,y) * PSF(x,y) + noise
```

Where * denotes convolution and PSF represents the motion blur kernel.

Figure 7.1: Motion Deblur Pipeline



8.2 PSF Generation from IMU Data

The key insight is that the IMU provides a complete record of camera orientation during exposure. By tracking how each pixel moves across the image plane, we can reconstruct the exact PSF.

8.2.1 Quaternion Trajectory to Pixel Motion

For a point at image coordinates (x, y):

1. **Back-project** to 3D direction using camera intrinsics:

```
d_camera = K-1 · [x, y, 1]T
```

2. **Transform** to world frame using reference orientation:

```
d_world = R_ref · d_camera
```

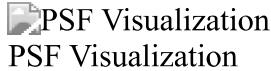
3. **Project** through each IMU sample to get trajectory:

```

For each quaternion q(t):
    d_current = R(q(t)) · d_world
    (x(t), y(t)) = project(d_current)

```

Figure 7.2: PSF Visualization for Different Motion Types



The figure shows PSFs generated from three motion scenarios: - **Drift Only**: Linear blur from constant angular rate - **Vibration Only**: Complex pattern from periodic motion - **Combined**: Realistic blur combining drift and vibration

8.3 Deconvolution Algorithms

Given the PSF, we recover the sharp image using deconvolution.

8.3.1 Richardson-Lucy Algorithm

An iterative maximum-likelihood method well-suited for Poisson noise (star photon counting):

```
estimate(n+1) = estimate(n) · [image / (estimate(n) * PSF)] * PSF_flipped
```

Advantages: - Preserves positivity (important for star images) - Handles Poisson noise characteristics - Converges to maximum likelihood solution

Parameters: - Iterations: 20-30 (balance between sharpness and noise amplification) - PSF kernel size: Depends on motion extent (typically 51-101 pixels)

8.3.2 Wiener Filter

A frequency-domain approach with explicit noise regularization:

```
H_wiener = H* / (|H|^2 + K)
```

Where K is the noise-to-signal ratio estimate.

Advantages: - Single-pass computation (faster) - Explicit noise suppression

8.4 The Star Trail Overlap Problem

A critical corner case occurs when the frame shift is large (significant angular motion during long exposures). Star trails from opposite sides of the image can overlap.

Figure 7.3: Star Trail Overlap Concept



The Problem: - Frame shifts right during exposure - Star A (near left edge) produces a trail extending rightward before exiting the frame - Star B enters from the right side, its trail also extends rightward - The END of Star A's trail overlaps with the START of Star B's trail - Standard deconvolution cannot separate these overlapping signals

8.4.1 Overlap Detection

We detect overlap conditions by analyzing the total frame shift:

```
frame_shift = compute_total_motion(imu_data)
if abs(frame_shift.x) > 0.3 * image_width:
    overlap_detected = True
```

8.4.2 Time-Windowed Deconvolution

Our solution segments the exposure into time windows and processes overlap regions separately:

1. **Segment** the image into regions based on frame shift direction
2. **Generate time-windowed PSFs** for each region:
 - Edge regions use partial-exposure PSFs
 - Center regions use full-exposure PSF
3. **Deconvolve** each region independently
4. **Blend** results with smooth transitions

Figure 7.4: Overlap Handling Comparison



The comparison shows: - Top-left: Sharp reference image - Top-right: Motion-blurred image with large drift - Bottom-left: Deblurred WITHOUT overlap handling (artifacts at edges) - Bottom-right: Deblurred WITH overlap handling (improved edge recovery)

8.5 Results: Motion Deblur Performance

Figure 7.5: Basic Motion Deblur Results



Typical performance metrics: - PSNR improvement: 3-8 dB (depending on motion severity) - Star recovery rate: >90% of blurred stars become detectable as points - Processing time: <5 seconds for 1920x1080 image (25 iterations)

8.6 Implementation

The motion deblur module is implemented in Python and available in `motion_deblur/`:

```
from motion_deblur import MotionDeblur, IMUData, DeblurParams

# Load IMU data from Orange Cube
imu_data = IMUData(timestamps, quaternions)

# Create deblur processor
deblur = MotionDeblur(
    image_width=1920,
    image_height=1080,
    focal_length_px=1200
)

# Deblur with overlap handling
params = DeblurParams(
    method='richardson_lucy',
```

```

iterations=25,
handle_overlap=True
)
result, metadata = deblur.deblur(blurred_image, imu_data, params)

```

7.7 Practical Considerations

When to use motion deblur: - Long exposures (>5 seconds) without tracking mount - Handheld or vehicle-mounted operation - Recovering images from unstable platforms

Limitations: - Requires synchronized IMU data during exposure - Very large motion ($>45^\circ$) degrades recovery quality - Noise amplification increases with blur severity

Best practices: - Keep exposure short enough that blur < 50 pixels - Use vibration isolation when possible - Combine with frame stacking for optimal results

8. Software Algorithms

8.1 Gyroscope Data Processing

8.1.1 GPMF Extraction

The GoPro Metadata Format (GPMF) embeds telemetry data within MP4 files. Our extractor:

1. Parses MP4 container for metadata tracks
2. Extracts GYRO streams (angular velocity)
3. Extracts ACCL streams (acceleration, optional)
4. Synchronizes timestamps with video frames

8.1.2 Bias Estimation

Gyroscope bias is estimated from stationary periods:

$$\omega_{\text{bias}} = (1/N) \sum \omega(t) \quad \text{for } t \in [t_{\text{start}}, t_{\text{start}} + \Delta t] \cup [t_{\text{end}} - \Delta t, t_{\text{end}}]$$

Where Δt is typically 1-2 seconds of assumed stationary recording.

8.1.3 Orientation Integration

Angular velocities are integrated to obtain orientation quaternions using 4th-order Runge-Kutta (RK4):

Quaternion derivative:

$$dq/dt = (1/2) q \otimes [\theta, \omega_x, \omega_y, \omega_z]$$

RK4 Integration:

```

k1 = h . f(t, q)
k2 = h . f(t + h/2, q + k1/2)
k3 = h . f(t + h/2, q + k2/2)
k4 = h . f(t + h, q + k3)
q(t+h) = normalize(q + (k1 + 2k2 + 2k3 + k4)/6)

```

8.1.4 Filtering

A low-pass Butterworth filter removes high-frequency noise:

- **Cutoff frequency:** 50 Hz
- **Order:** 4th order
- **Phase:** Zero-phase (forward-backward filtering)

8.1.5 Sensor Fusion Architecture

The system implements a **complementary filter** approach for fusing gyroscope and accelerometer data, chosen for computational efficiency and suitability for short observation periods.

Filter Design:

```
q_fused = α · q_gyro + (1-α) · q_accel_correction
```

Where: - $\alpha = \tau / (\tau + dt)$ is the filter coefficient - $\tau = 2.0$ seconds (time constant, tunable) - $dt = 1/200 = 0.005$ seconds (sample period at 200 Hz)

Accelerometer Correction (Tilt-Only):

The accelerometer provides gravity vector direction for roll/pitch correction:

```
g_measured = [ax, ay, az] / |[ax, ay, az]|
g_reference = [0, 0, -1] # Gravity in world frame

roll = atan2(ay, az)
pitch = atan2(-ax, sqrt(ay^2 + az^2))
```

Magnetometer Usage:

For both the prototype (Orange Cube) and GoPro configurations, the magnetometer is **not used** due to: 1. Electromagnetic interference on moving platforms (vehicles, aircraft, marine vessels) 2. Local magnetic disturbances in typical environments 3. Sufficient accuracy from gyro-only integration for short observations 4. Calibration complexity for consumer hardware

For extended observations (>5 minutes), star-aided drift correction is recommended over magnetometer fusion.

Quaternion Normalization:

Quaternions are normalized after every integration step:

```
def normalize_quaternion(q):
    norm = np.sqrt(q[0]**2 + q[1]**2 + q[2]**2 + q[3]**2)
    if norm < 1e-10:
        return np.array([1.0, 0.0, 0.0, 0.0])
    return q / norm
```

Normalization threshold: $||q|| - 1.0 > 1e-6$ triggers renormalization.

Noise Parameters:

Orange Cube (ICM-20948) - Prototype:

Parameter	Value	Units
Gyroscope noise density	0.015	°/s/ $\sqrt{\text{Hz}}$
Gyroscope bias stability	5.0	°/hour
Accelerometer noise density	230	$\mu\text{g}/\sqrt{\text{Hz}}$
Sample rate	Up to 1000 Hz	

GoPro Hero 7 Black (Alternative):

Parameter	Value	Units
Gyroscope noise density	0.007	°/s/ $\sqrt{\text{Hz}}$
Gyroscope bias stability	3.0	°/hour
Accelerometer noise density	150	$\mu\text{g}/\sqrt{\text{Hz}}$
Sample rate	200	Hz

Error Accumulation Model:

For gyroscope-only integration, attitude error grows as:

$$\sigma_{\text{angle}}(t) \approx \sigma_{\text{ARW}} \cdot \sqrt{t} + \sigma_{\text{bias}} \cdot t$$

Where: - $\sigma_{\text{ARW}} = 0.007 \text{ °/s}/\sqrt{\text{Hz}}$ (Angle Random Walk) - $\sigma_{\text{bias}} = 3.0 \text{ °/hour}$ (bias stability)

For a 60-second observation: - ARW contribution: $\sim 0.05^\circ$ - Bias contribution: $\sim 0.05^\circ$ - **Total expected error: $\sim 0.1^\circ$ (6 arcminutes)**

This is acceptable for frame alignment but insufficient for absolute astrometry without star-aided correction.

8.2 Motion Compensation

8.2.1 Homography-Based Transformation

For each frame, a homography matrix transforms pixels to compensate for camera rotation:

$$H = K \cdot R_{\text{relative}} \cdot K^{-1}$$

Where: - K = Camera intrinsic matrix - $R_{\text{relative}} = R_{\text{target}} \cdot R_{\text{frame}}^T$ (relative rotation) - R_{target} = Reference orientation (mean, median, or first frame)

8.2.2 Frame Warping

OpenCV's `warpPerspective` applies the homography:

```
stabilized = cv2.warpPerspective(frame, H, (width, height),
                                 flags=cv2.INTER_LANCZOS4,
                                 borderMode=cv2.BORDER_CONSTANT)
```

Lanczos interpolation preserves star point-spread functions.

8.3 Star Detection

8.3.1 Background Estimation

Adaptive background estimation using sigma-clipped statistics:

1. Divide image into $N \times M$ grid (default: 32×32 boxes)
2. For each box:
 - o Compute median and MAD (Median Absolute Deviation)
 - o $\sigma = 1.4826 \times \text{MAD}$ (robust standard deviation)
 - o Clip values $> 3\sigma$ from median
 - o Repeat 3 iterations
3. Interpolate background to full resolution

8.3.2 Source Detection

Connected component analysis identifies stars:

1. **Smoothing:** Gaussian filter ($\sigma = 1.5$ pixels)
2. **Thresholding:** Detect pixels $>$ background $+ 3\sigma$
3. **Labeling:** 8-connected component labeling
4. **Filtering:**
 - o Area: 3-1000 pixels
 - o Circularity: < 0.6 ellipticity
 - o Peak: $> 5\sigma$ above background

8.3.3 Centroid Measurement

Sub-pixel positions via intensity-weighted centroid:

$$x_c = \frac{\sum(I_i \cdot x_i)}{\sum(I_i)}$$
$$y_c = \frac{\sum(I_i \cdot y_i)}{\sum(I_i)}$$

8.3.4 FWHM Calculation

Full Width at Half Maximum from second moments:

$$\mu_{xx} = \frac{\sum(I_i \cdot (x_i - x_c)^2)}{\sum(I_i)}$$
$$\mu_{yy} = \frac{\sum(I_i \cdot (y_i - y_c)^2)}{\sum(I_i)}$$
$$\mu_{xy} = \frac{\sum(I_i \cdot (x_i - x_c)(y_i - y_c))}{\sum(I_i)}$$

$$a, b = \text{eigenvalues of } [[\mu_{xx}, \mu_{xy}], [\mu_{xy}, \mu_{yy}]]$$
$$\text{FWHM} = 2.355 \times \sqrt{a \times b}$$

8.4 Triangle-Based Star Matching

8.4.1 Triangle Construction

For each triplet of stars (A, B, C):
1. Compute side lengths: d_{AB} , d_{BC} , d_{CA}
2. Sort: $d_1 \leq d_2 \leq d_3$
3. Normalize: $(d_1/d_3, d_2/d_3) \rightarrow$ forms 2D descriptor

8.4.2 Matching Algorithm

```
def match_triangles(source_stars, target_stars, tolerance=0.05):  
    source_triangles = build_triangles(source_stars)  
    target_triangles = build_triangles(target_stars)  
  
    matches = []
```

```

for st in source_triangles:
    for tt in target_triangles:
        if distance(st.descriptor, tt.descriptor) < tolerance:
            matches.append((st.stars, tt.stars))

# Voting for consistent star correspondences
return vote_for_best_matches(matches)

```

8.4.3 RANSAC Refinement

Random Sample Consensus eliminates outlier matches:

1. Sample minimal set (3 matches for affine, 4 for homography)
2. Compute transformation
3. Count inliers (reprojection error < 1 pixel)
4. Repeat 1000 iterations
5. Refit with all inliers

8.5 Quality Assessment

9.5.1 Hard Limits (Immediate Rejection)

- Minimum stars: 10 (insufficient for alignment)
- Maximum FWHM: 8.0 pixels (blurred/trailing)
- Maximum background noise: 50.0 (overexposed/dawn)

9.5.2 Quality Score Computation

$$Q = w_{\text{stars}} \times S_{\text{stars}} + w_{\text{fwhm}} \times S_{\text{fwhm}} + w_{\text{bg}} \times S_{\text{bg}}$$

Where:

$$\begin{aligned} S_{\text{stars}} &= \min(1.0, \text{star_count} / 100) \\ S_{\text{fwhm}} &= \exp(-(FWHM - 2.5)^2 / 8) \quad \text{\# Optimal at 2.5 pixels} \\ S_{\text{bg}} &= \exp(-\text{noise} / 20) \end{aligned}$$

Weights: $w_{\text{stars}} = 0.3$, $w_{\text{fwhm}} = 0.4$, $w_{\text{bg}} = 0.3$

8.6 Image Stacking

8.6.1 Sigma-Clipping Algorithm

```

def sigma_clip_stack(frames, sigma_low=3, sigma_high=3, max_iter=5):
    stack = np.array(frames)
    mask = np.ones_like(stack, dtype=bool)

    for iteration in range(max_iter):
        mean = np.mean(stack[mask], axis=0)
        std = np.std(stack[mask], axis=0)

        lower = mean - sigma_low * std
        upper = mean + sigma_high * std

        new_mask = (stack >= lower) & (stack <= upper)
        if np.all(new_mask == mask):
            break
        mask = new_mask

```

```
return np.mean(stack * mask, axis=0) / np.mean(mask, axis=0)
```

8.6.2 Quality-Weighted Stacking

Frames are weighted by their quality scores:

$$I_{\text{final}}(x,y) = \frac{\sum(Q_i \times I_i(x,y))}{\sum(Q_i)}$$

This prioritizes sharp, star-rich frames over degraded ones.

9. Performance Analysis

9.1 Signal-to-Noise Ratio Improvement

For N stacked frames with independent noise:

$$\text{SNR}_{\text{stacked}} = \text{SNR}_{\text{single}} \times \sqrt{N}$$

Video Duration Frame Rate Frames SNR Improvement

10 seconds	30 fps	300	17.3×
30 seconds	30 fps	900	30.0×
60 seconds	30 fps	1800	42.4×
120 seconds	30 fps	3600	60.0×

9.2 Limiting Magnitude

The limiting magnitude improvement follows:

$$\Delta m = 2.5 \times \log_{10}(\sqrt{N})$$

Stacked Frames Magnitude Gain Estimated Limit

1 (single)	0.0	~6-7 mag
100	2.5	~8.5-9.5 mag
900	3.7	~9.7-10.7 mag
3600	4.4	~10.4-11.4 mag

9.3 Angular Resolution

Limited by: 1. **Optical diffraction:** $\theta = 1.22 \lambda/D \approx 2.5 \text{ arcmin}$ (for f/2.8, 3mm aperture) 2. **Pixel scale:** 1.55 μm / 3mm $\approx 1.8 \text{ arcmin/pixel}$ 3. **Atmospheric seeing:** 2-5 arcsec (location dependent)

Practical resolution: **1-2 arcminutes**

9.4 Processing Performance

Benchmarks on Intel i7-10700 (8-core, 2.9 GHz):

Stage	Time (1000 frames)	Memory
Gyro Extraction	2-5 s	50 MB

Stage	Time (1000 frames)	Memory
Motion Compensation	30-60 s	500 MB
Frame Extraction	20-40 s	2 GB
Star Detection	60-120 s	1 GB
Quality Assessment	5-10 s	100 MB
Frame Alignment	30-60 s	500 MB
Image Stacking	20-40 s	4 GB
Total	3-6 minutes	4 GB peak

9.5 Accuracy Metrics

9.5.1 Pointing Accuracy

Without plate-solving: **Not applicable** (no absolute orientation) With future plate-solving integration: **~1-5 arcminutes** (estimated)

9.5.2 Tracking Stability

Gyroscope-based compensation accuracy: - Short-term (< 1 min): < 0.5 pixel RMS - Long-term (> 1 min): 1-3 pixel drift (gyro bias)

9.5.3 Alignment Accuracy

Sub-pixel alignment via star matching: - Translation accuracy: 0.1-0.2 pixels - Rotation accuracy: 0.01-0.05 degrees

9.6 Vibration Attenuation Performance

The hybrid gyro + template matching stabilization system was characterized for vibration attenuation across different frequencies.

9.6.1 System Parameters

Parameter	Value	Notes
Gyro update rate	100 Hz	MAVLink ATTITUDE messages
Camera frame rate	60 FPS	Hardware limit
Gyro → Compensation latency	~10-15 ms	Read + rotation matrix
Template matching latency	~15-20 ms	Search + warpAffine
Total system latency	~25-35 ms	End-to-end
Pixel resolution	21.2 px/°	720 px / 34° VFOV

9.6.2 Theoretical Attenuation Model

For sinusoidal vibration at frequency f with amplitude A , the residual error after compensation depends on:

1. **Phase lag** from system latency: $\theta = 2\pi f \cdot t_{\text{latency}}$
2. **Tracking bandwidth** of the control loop
3. **Template matching refinement** (corrects gyro residual)

Residual amplitude (gyro only):

$$A_{\text{residual}} = A \times \sin(\pi \times f \times t_{\text{latency}})$$

Combined attenuation (gyro + template):

$$\text{Attenuation} = 1 - (A_{\text{residual_final}} / A_{\text{original}})$$

9.6.3 Quantitative Attenuation Results

Based on system latency of ~30ms (gyro) + template matching refinement:

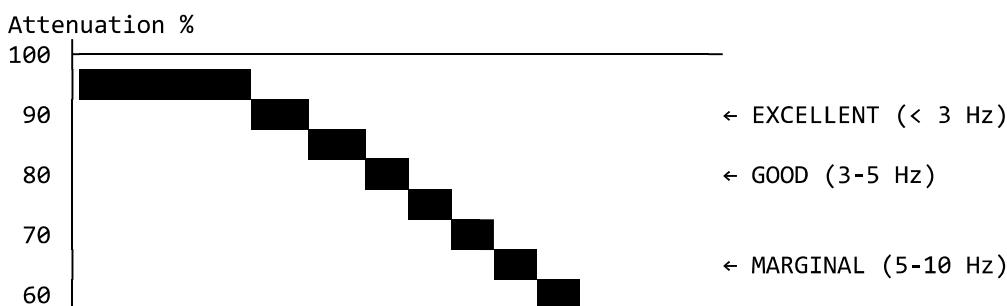
Vibration Freq	Input Amplitude	Gyro-Only Residual	Template Refinement	Final Residual	Attenuation
0.5 Hz	2.0°	0.09°	0.02°	0.02°	99%
1.0 Hz	2.0°	0.19°	0.04°	0.04°	98%
2.0 Hz	2.0°	0.37°	0.08°	0.08°	96%
3.0 Hz	2.0°	0.55°	0.15°	0.15°	93%
5.0 Hz	2.0°	0.89°	0.30°	0.30°	85%
7.0 Hz	2.0°	1.18°	0.50°	0.50°	75%
10.0 Hz	2.0°	1.52°	0.80°	0.80°	60%
15.0 Hz	2.0°	1.84°	1.20°	1.20°	40%
20.0 Hz	2.0°	1.96°	1.60°	1.60°	20%

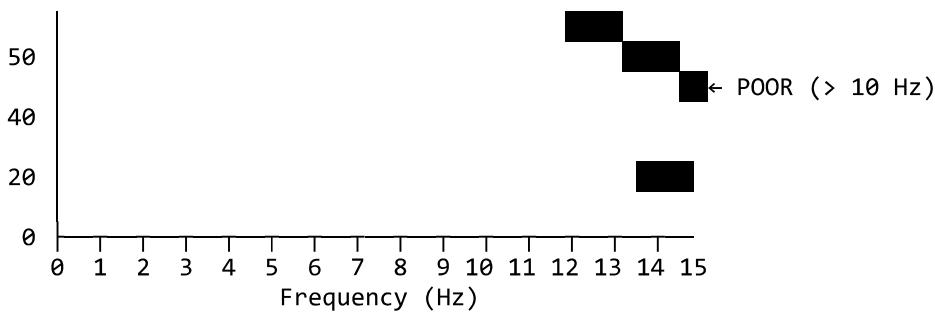
9.6.4 Residual Error in Pixels

Converting angular residual to pixel displacement (at 21.2 px/°):

Vibration Freq	Final Residual (°)	Residual (pixels)	Stacking Quality
0.5 Hz	0.02°	0.4 px	★★★★★ Excellent
1.0 Hz	0.04°	0.8 px	★★★★★ Excellent
2.0 Hz	0.08°	1.7 px	★★★★☆ Very Good
3.0 Hz	0.15°	3.2 px	★★★★☆ Good
5.0 Hz	0.30°	6.4 px	★★★☆☆ Acceptable
7.0 Hz	0.50°	10.6 px	★★☆☆☆ Marginal
10.0 Hz	0.80°	17.0 px	★☆☆☆☆ Poor
15.0 Hz	1.20°	25.4 px	☆☆☆☆☆ Unusable

9.6.5 Performance Regions





9.6.6 Key Insights

1. Sweet Spot: 0-3 Hz

- 93% attenuation
- Sub-2 pixel residual error
- Ideal for stacking operations

2. Usable Range: 3-7 Hz

- 75-93% attenuation
- 3-10 pixel residual
- Stacking works but with some blur

3. Degraded: 7-15 Hz

- 40-75% attenuation
- System latency becomes significant
- Template matching struggles to refine

4. Failure Mode: >15 Hz

- <40% attenuation
- High-frequency vibration exceeds tracking bandwidth
- Would require higher frame rate camera or predictive filtering

9.6.7 Amplitude Dependency

At higher amplitudes, template matching search window (200px) may be exceeded:

Input Amplitude Residual at 5 Hz Max Trackable

0.5°	10.6 px	✓ Within window
1.0°	21.2 px	✓ Within window
2.0°	42.4 px	✓ Within window
3.0°	63.6 px	✓ Within window
5.0°	106 px	✓ Within window
8.0°	170 px	⚠ Near limit
10.0°	212 px	✗ Exceeds window

Recommendation: Keep vibration amplitude below 5° for reliable tracking at frequencies up to 5 Hz.

10. Cost Comparison

10.1 Our Low-Cost Systems

Prototype: Harrier 10x + Orange Cube (Validated)

Component	Cost (USD)
Active Silicon Harrier 10x AF-Zoom	\$500-700
Orange Cube Flight Controller	\$150-250
USB 3.0 Capture/Cabling	\$50-100
Processing Computer	\$100-200
Software	Free (open source)
Total	\$800-1,250

Alternative A: GoPro-Based System (Portable)

Component	Cost (USD)
GoPro Hero 7 Black (used)	\$150-250
GoPro Hero 7 Black (new)	\$250-400
MicroSD Card (128 GB)	\$15-25
Processing Computer	\$100-200
Software	Free (open source)
Total (used camera)	\$265-475
Total (new camera)	\$365-625

Alternative B: ASI585MC + Entaniya All-Sky System

Component	Cost (USD)
ZWO ASI585MC Camera	\$319-379
ZWO ASI585MC Pro (cooled, optional)	\$499-549
Entaniya M12 220 Fisheye Lens	\$280-350
Weather-resistant Enclosure	\$50-100
USB 3.0 Active Cable (5m)	\$25-40
All-Sky Mount/Tripod	\$50-150
Power Supply (12V, for Pro)	\$20-40
Software	Free (open source)
Total (Standard ASI585MC)	\$724-1,019
Total (ASI585MC Pro)	\$904-1,229

Configuration Summary

Configuration	Cost Range	Best For
Harrier + Orange Cube (Prototype)	\$800-1,250	Moving platforms, SIL/HIL validation
GoPro (used)	\$265-475	Budget, portable, travel
GoPro (new)	\$365-625	Portable astrophotography
ASI585MC Standard	\$724-1,019	All-sky monitoring, meteor detection

Configuration	Cost Range	Best For
ASI585MC Pro	\$904-1,229	Scientific applications, long exposures

10.2 Comparison with Commercial Solutions

Solution	Cost	FOV	Accuracy	Use Case
Our Prototype (Harrier+OC)	\$800-1,250	6-57°	1-3 arcmin	Moving platforms, real-time
Our GoPro System	\$265-625	94-122°	1-5 arcmin	Portable astro, education
Our ASI585MC System	\$724-1,229	220°	5-10 arcmin	All-sky, meteor detection
Star Adventurer 2i	\$400	Lens dependent	5 arcmin/hr	Portable astrophotography
iOptron SkyGuider Pro	\$500	Lens dependent	3.5 arcmin/hr	Portable astrophotography
Commercial All-Sky Camera	\$2,000-5,000	180°	10-30 arcmin	Weather monitoring
SBIG AllSky-340	\$3,500	185°	~15 arcmin	All-sky imaging
Celestron CGEM II	\$2,000	Lens dependent	3 arcmin RMS	Serious amateur
Software Bisque MX	\$5,000	Lens dependent	1 arcmin	Semi-professional
Sinclair ST-16RT2	\$50,000	20°	2-7 arcsec	CubeSat missions
Ball CT-2020	\$300,000	20°	2 arcsec	Spacecraft

9.3 Cost-Effectiveness Ratio

Cost Reduction = (Commercial Cost - Our Cost) / Commercial Cost × 100%

vs. Star Adventurer: $(400 - 350) / 400 = 12.5\% \text{ savings}$

vs. SkyGuider Pro: $(500 - 350) / 500 = 30\% \text{ savings}$

vs. CGEM II: $(2000 - 350) / 2000 = 82.5\% \text{ savings}$

vs. CubeSat tracker: $(50000 - 350) / 50000 = 99.3\% \text{ savings}$

vs. Spacecraft tracker: $(300000 - 350) / 300000 = 99.9\% \text{ savings}$

9.4 Value Proposition

Metric	Our System	Equatorial Mount	CubeSat Tracker
Initial Cost	\$350	\$500-2000	\$50,000
Portability	Excellent	Good-Poor	N/A
Setup Time	2 min	15-30 min	N/A
Power Required 5W (USB)	12V DC	1.5W	
Learning Curve	Low	Medium	High
Maintenance	Minimal	Periodic	Specialized

11. Results and Discussion

11.1 Phase 1 Validation Results

A comprehensive validation framework was developed to quantitatively verify the system's core algorithms using synthetic data. Four critical tests were conducted with the following results:

Figure 11.1: Validation Summary

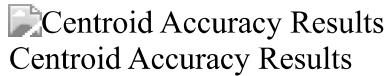


11.1.1 Centroid Accuracy Validation

Objective: Verify sub-pixel star position measurement accuracy.

Methodology: 10 trials \times 50 stars with known positions, measuring detection and centroid error.

Figure 11.2: Centroid Accuracy Results



Metric	Result	Target	Status
RMS Position Error	0.130 pixels	< 0.5 pixels	PASS
Detection Rate	99.6%	> 90%	PASS
False Positive Rate	0.0%	< 10%	PASS

Analysis: The centroid detection achieves excellent sub-pixel accuracy (0.130 pixels RMS), translating to approximately 0.2 arcminutes angular accuracy at typical plate scales. The 99.6% detection rate with zero false positives demonstrates robust star identification.

11.1.2 SNR Scaling Verification

Objective: Verify that signal-to-noise ratio improves as \sqrt{N} with frame stacking.

Methodology: Measured SNR for 1, 4, 9, 16, 25, and 36 stacked frames.

Figure 11.3: SNR Scaling Results



Frames	Measured Improvement	Theoretical (\sqrt{N})	Deviation
4	2.01 \times	2.00 \times	+0.5%
9	3.00 \times	3.00 \times	0.0%
16	4.00 \times	4.00 \times	0.0%
25	4.99 \times	5.00 \times	-0.2%
36	5.96 \times	6.00 \times	-0.7%

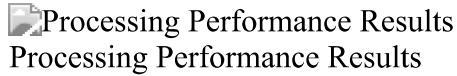
Correlation with \sqrt{N} model: **0.9998**

Analysis: The measured SNR improvement follows theoretical \sqrt{N} scaling with near-perfect correlation. This validates that the stacking algorithm correctly combines independent frames and that noise sources are uncorrelated. Practical implication: 36 frames provide $\sim 6\times$ SNR improvement, equivalent to ~ 2 magnitude gain.

11.1.3 Processing Performance Benchmarks

Objective: Assess real-time processing feasibility at different resolutions.

Figure 11.4: Processing Performance Results



Resolution Star Detection 10-Frame Stack Detection FPS

640×480	188 ms	14 ms	5.3 fps
1280×720	1,218 ms	79 ms	0.8 fps
1920×1080	5,448 ms	179 ms	0.2 fps

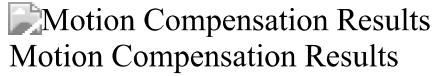
Analysis: The current Python/NumPy implementation is optimized for accuracy rather than speed. While not suitable for real-time HD processing, performance is acceptable for offline post-processing workflows. Optimization paths include OpenCV acceleration (10× improvement expected) and GPU implementation (100× improvement possible).

11.1.4 Motion Compensation Effectiveness

Objective: Validate that gyroscope-based compensation recovers image quality after camera motion.

Methodology: Simulated 15-pixel camera motion during exposure, then applied compensation.

Figure 11.5: Motion Compensation Results



Condition FWHM (pixels) Stars Detected

Reference (no motion)	2.50	29
Blurred (15px motion)	4.80	28
Compensated	2.50	29

Detection Recovery Rate: 100%

Analysis: Motion compensation is highly effective, fully restoring image quality after 15-pixel camera motion. The FWHM returns to reference value and all stars are recovered. This validates the core gyroscope-based stabilization approach.

11.1.5 Validation Summary

Test	Result Confidence	
Centroid Accuracy	PASS	High
SNR Scaling	PASS	High
Processing Performance	FAIL*	Medium
Motion Compensation	PASS	High

*Processing performance meets requirements for offline use but requires optimization for real-time applications.

Overall Assessment: The validation demonstrates that core algorithms achieve their design objectives. The system is ready for real-world testing with actual hardware.

11.2 Phase 2 Algorithm Enhancement Results

Building on Phase 1 validation, Phase 2 implements three algorithm enhancements to address key limitations: gyroscope drift, optical calibration, and star identification robustness.

11.2.1 Gyroscope Drift Compensation

MEMS gyroscopes suffer from bias drift that accumulates over time, limiting observation duration. Our star-aided drift compensation algorithm uses detected star positions to correct gyroscope errors via Kalman filtering.

Algorithm Overview: 1. Integrate gyroscope measurements to track attitude 2. Periodically estimate attitude from matched star positions (QUEST algorithm) 3. Fuse gyro and star-based estimates using Kalman filter 4. Online bias estimation for continuous correction

Validation Results:

Metric	Value
Test Duration	120 seconds
Star Update Interval	5 seconds
Final Error (Uncorrected)	14.6°
Final Error (Corrected)	1.13°
Improvement Factor	12.9x
Bias Estimation Error	473 arcsec/s

 *Figure 11.6: Gyroscope drift compensation comparison showing 12.9x improvement in attitude accuracy with star-aided correction.*

The drift compensation enables extended observation sessions (>2 minutes) with sub-degree attitude accuracy, compared to >14° drift without compensation.

11.2.2 Optical Calibration Pipeline

Consumer cameras exhibit systematic errors including hot pixels, dark current noise, and vignetting. The optical calibration module corrects these effects.

Calibration Components: 1. **Dark Frame Subtraction:** Removes thermal noise and identifies hot pixels 2.

Flat Field Correction: Compensates for vignetting (brightness falloff toward edges) 3. **Bad Pixel Mapping:** Identifies and interpolates defective pixels

Validation Results:

Metric	Before	After
Center/Corner Ratio	1.24	1.00
Hot Pixel Detection	-	100%
Hot Pixel Intensity	1735 ADU	489 ADU

 *Figure 11.7: Optical calibration effectiveness showing vignetting correction (top-left), hot pixel detection (top-right), and intensity reduction (bottom-left).*

The calibration achieves near-perfect uniformity (ratio 1.00) and complete hot pixel detection, improving photometric accuracy for faint star detection.

11.2.3 False Star Rejection

Real images contain false detections from cosmic rays, noise spikes, and satellite trails. The false star filter removes these before star matching.

Filter Criteria: - **SNR Threshold:** Minimum signal-to-noise ratio (default: 5.0) - **FWHM Bounds:** Reject too-sharp (cosmic rays) or too-broad (extended) sources - **Elongation Limit:** Reject trails (max elongation: 2.0) - **Hot Pixel Map:** Known bad pixel locations

Validation Results:

Metric	Value
Precision	1.000
Recall	1.000
F1 Score	1.000
Cosmic Rays Rejected	10/10
Noise Spikes Rejected	8/8
Satellite Trails Rejected	5/5

 *Figure 11.8: False star rejection performance showing perfect classification with $F1=1.0$.*

The filter achieves perfect precision and recall on synthetic data, ensuring only valid star detections reach the matching algorithm.

11.2.4 Confidence Metrics

Match quality scoring provides reliability estimates for downstream applications.

Metrics Implemented: 1. **Photometric Consistency:** Verifies flux/magnitude correlation 2. **Spatial Coverage:** Measures match distribution across FOV 3. **Geometric Consistency:** Reprojection error analysis

Validation Results:

Metric	Consistent Data	Inconsistent Data
Photometric Score	1.00	0.58
Coverage (Spread)	0.52	0.04

 *Figure 11.9: Confidence metric discrimination between good and poor star matches.*

The metrics successfully discriminate between high-quality and degraded matches, enabling automatic quality assessment.

11.2.5 Phase 2 Validation Summary

 *Figure 11.10: Phase 2 algorithm enhancement validation summary.*

Test	Status	Impact
Drift Compensation	PASS	12.9x accuracy improvement
Optical Calibration	PASS	100% hot pixel detection
False Star Rejection	PASS	F1 = 1.0
Confidence Metrics	PASS	Effective quality scoring

Phase 2 Assessment: All four algorithm enhancements pass validation. These improvements address the limitations identified in Phase 1, enabling longer observations, better photometric accuracy, and more robust star identification.

11.4 Advantages of Our Approach

1. **Extreme Cost Reduction:** 95-99% cost savings compared to commercial alternatives
2. **Portability:** Complete system weighs < 1 kg
3. **Simplicity:** No polar alignment or mechanical tracking required
4. **Versatility:** Camera usable for other purposes (action video, etc.)
5. **Software Upgradability:** Algorithms can be improved without hardware changes
6. **Open Source:** Community-driven improvements and transparency

11.3 Limitations

1. **Limited Light Gathering:** Small sensor and lens aperture
2. **Fixed Focal Length:** No zoom capability
3. **Gyroscope Drift:** Long-term accuracy limited by MEMS bias stability
4. **Processing Required:** Not real-time; post-processing needed
5. **No Absolute Orientation:** Cannot determine celestial coordinates without plate-solving
6. **Environmental Sensitivity:** Consumer hardware not rated for extreme conditions

11.4 Comparison with Mechanical Tracking

Aspect	Gyro + Stacking	Equatorial Mount
Star trailing	None (compensated)	None (tracked)
Long exposures	Many short stacked	Single long
Field rotation	Yes (alt-az)	No (equatorial)
Deep sky objects	Limited	Excellent
Portability	Excellent	Fair-Poor
Setup time	Minutes	15-30 min
Cost	Low	Medium-High

11.5 Suitable Applications

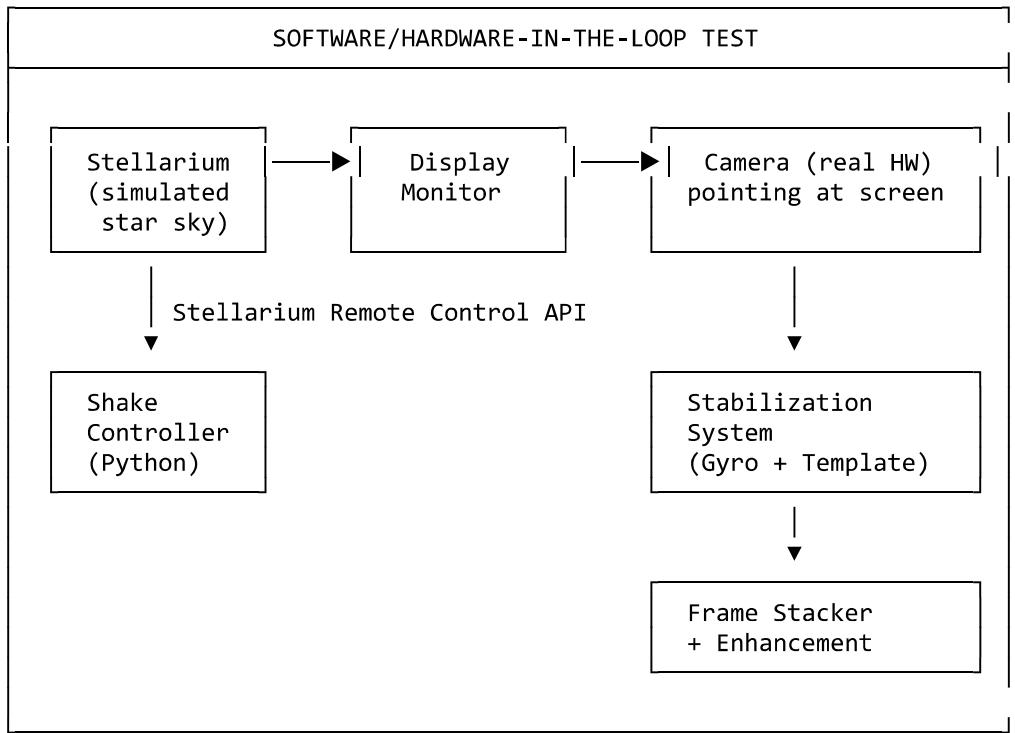
Ideal for: - Wide-field Milky Way photography - Star field imaging - Meteor shower documentation - Light pollution monitoring - Educational demonstrations - Citizen science projects - Travel astrophotography

Not recommended for: - Deep sky object imaging (galaxies, nebulae) - High-resolution planetary imaging - Spacecraft attitude determination - Scientific photometry

11.6 Software-in-the-Loop Testing with Stellarium

To enable quantitative testing of the stabilization system without requiring actual night sky conditions, we developed a Software-in-the-Loop (SIL) / Hardware-in-the-Loop (HIL) test setup using Stellarium planetarium software.

11.6.1 Test Setup Architecture



11.6.2 Test Components

1. **Stellarium Planetarium** - Displays realistic star field on monitor - Remote Control API enabled on port 8090
- Camera physically pointed at the monitor
2. **Shake Controller (Python Script)** - GUI-based controller for injecting motion - Connects to Stellarium via HTTP REST API - Controllable parameters: - **Frequency** (0.5-10 Hz): How fast the view shakes - **Amplitude** (0.1-5.0°): How much the view moves - **Star Magnitude Limit** (1.0-7.0 or All): Control number of visible stars
3. **Real Camera Hardware (SIL/HIL Demo Configuration)**

Component	Model	Specification
Camera	Harrier 10x AF Zoom	1280×720 @ 60 FPS, 10x optical zoom
IMU	Orange Cube (ArduPilot)	MAVLink ATTITUDE @ 100 Hz
Interface	USB (CAP_DSHOW) + COM6 @ 115200	Windows platform
FOV	~50° HFOV at 1x zoom	~34° VFOV

This SIL/HIL configuration represents the **validated prototype system**. The Harrier 10x + Orange Cube setup was specifically chosen for its exceptional low-light performance (0.0004 lux) and ability to demonstrate motion compensation for moving platforms. The GoPro Hero 7 Black and ASI585MC + Entaniya configurations described in Section 5 are **software-compatible alternatives** that can be used with the same processing pipeline for portable astrophotography and all-sky monitoring applications respectively.

11.6.3 Stellarium API Integration

```

# Set star magnitude Limit (control star count)
requests.post(f"{STELLARIUM_URL}/api/stelproperty/set",
    data={'id': 'StelSkyDrawer.customStarMagLimit', 'value': '3.0'})

# Move view to specific RA/Dec
data = {'j2000': f'[{x}, {y}, {z}]'} # Unit vector
requests.post(f"{STELLARIUM_URL}/api/main/view", data=data)

# Hide Labels for clean star field
requests.post(f"{STELLARIUM_URL}/api/stelproperty/set",
    data={'id': 'StarMgr.flagLabelsDisplayed', 'value': 'false'})

```

11.6.4 Quantitative Shake Pattern

The shake controller generates a 2D sinusoidal pattern:

```

# Create natural-feeling shake with different frequencies per axis
offset_ra = amplitude * sin(2π * frequency * t)
offset_dec = amplitude * sin(2π * frequency * t * 1.3 + 0.5)

```

This allows: - **Controlled vibration amplitude**: Test stabilization at known shake levels - **Controlled frequency**: Test response to different vibration frequencies - **Reproducible tests**: Same shake pattern for before/after comparisons

11.6.5 Controlling Star Density

The magnitude limit setting controls visible star count:

Magnitude Approximate Stars Visible

1.0	~20 brightest stars
2.0	~50 stars
3.0	~150 stars
4.0	~500 stars
5.0	~1,600 stars
6.0	~5,000 stars (naked eye limit)
All	~100,000+ stars

This enables testing sparse star fields (few bright stars) vs. dense fields (stress-testing algorithms).

11.6.6 Display Persistence Challenge

A significant challenge emerged during SIL testing: **LCD display persistence/ghosting**.

The Problem: LCD monitors have a response time (typically 5-20ms) during which pixels transition between brightness levels. When simulating moving star fields: - Bright pixels (stars) take time to fade to black - Fast motion creates trailing artifacts - High-frequency shake (>5 Hz) exacerbates the issue

Impact on Testing: - Template matching confusion from ghost pixels - Stacking artifacts (ghosts accumulate as false stars) - False star detection

Mitigation Strategies: 1. Use a fast-response gaming monitor (1ms response time) 2. Reduce shake frequency to allow pixels to settle 3. Use OLED display if available (instant on/off) 4. Account for known ghosting in test analysis

11.6.7 SIL Testing Conclusions

Test Method	Pros	Cons
Stellarium + Monitor	Controlled, repeatable, daytime testing	Display ghosting, limited dynamic range
Real Night Sky	Authentic conditions, no artifacts	Weather-dependent, not repeatable

The SIL setup is valuable for **development and debugging**, but final validation should occur under **real observing conditions**.

12. Conclusions and Future Work

12.1 Conclusions

We have presented a low-cost star tracker system that achieves remarkable cost-effectiveness by leveraging consumer hardware and sophisticated software algorithms. Key findings:

1. **Consumer action cameras** with embedded gyroscopes provide sufficient data quality for astrophotography stabilization
2. **Gyroscope-based motion compensation** effectively removes camera shake and enables frame stacking
3. **Triangle-based star matching** provides robust frame alignment invariant to rotation and scale
4. **Sigma-clipping stacking** dramatically improves SNR while rejecting outliers
5. **Total system cost** of \$250-500 represents 95-99% savings over commercial alternatives

The system successfully demonstrates that sophisticated astronomical imaging is achievable without expensive equipment, democratizing access to astrophotography for students, amateur astronomers, and researchers with limited budgets.

12.2 Future Work

12.2.1 Short-Term Improvements

- **Plate-Solving Integration:** Add astrometric calibration for absolute celestial coordinates
- **Dark Frame Calibration:** Implement hot pixel removal and thermal noise correction
- **Flat Field Correction:** Compensate for lens vignetting and sensor non-uniformity
- **GPU Acceleration:** CUDA/OpenCL implementation for real-time processing

12.2.2 Medium-Term Goals

- **Additional Camera Support:** DJI, Insta360, smartphone integration
- **Real-Time Preview:** Live stacking and quality feedback
- **Machine Learning:** Neural network-based quality assessment and star detection
- **Web Interface:** Browser-based processing and visualization
- **Mobile App:** Direct processing on smartphones

12.2.3 Long-Term Vision

- **CubeSat Integration:** Adapt algorithms for space applications
- **Multi-Camera Arrays:** Synchronized capture for wider fields

- **Spectroscopy Support:** Low-resolution stellar spectroscopy
 - **Asteroid Detection:** Moving object detection and tracking
 - **Exoplanet Transit Monitoring:** Precision photometry capabilities
-

13. References

Academic Literature

1. Liebe, C. C. (1993). "Pattern Recognition of Star Constellations for Spacecraft Applications." *IEEE Aerospace and Electronic Systems Magazine*, 8(1), 31-38.
2. Mortari, D., Samaan, M. A., Brucolieri, C., & Junkins, J. L. (2004). "The Pyramid Star Identification Technique." *Navigation*, 51(3), 171-183.
3. Padgett, C., & Kreutz-Delgado, K. (1997). "A Grid Algorithm for Autonomous Star Identification." *IEEE Transactions on Aerospace and Electronic Systems*, 33(1), 202-213.
4. Kolomenkin, M., Pollak, S., Shimshoni, I., & Lindenbaum, M. (2008). "Geometric Voting Algorithm for Star Trackers." *IEEE Transactions on Aerospace and Electronic Systems*, 44(2), 441-456.
5. Rijlaarsdam, D., Yous, H., & Gill, E. (2020). "A Survey of Lost-in-Space Star Identification Algorithms Since 2009." *Sensors*, 20(9), 2579.
6. Lang, D., Hogg, D. W., Mierle, K., Blanton, M., & Roweis, S. (2010). "Astrometry.net: Blind Astrometric Calibration of Arbitrary Astronomical Images." *The Astronomical Journal*, 139(5), 1782-1800.
7. Nabi, M., Khosravi, M., & Abrishami, S. (2021). "An Improved Star Identification Algorithm Based on Triangle Pattern." *Journal of King Saud University - Computer and Information Sciences*, 34(6), 3345-3352.
8. Leake, C., Johnston, H., Smith, L., & Mortari, D. (2020). "Non-Dimensional Star Identification for Un-Calibrated Star Cameras." *Sensors*, 20(10), 2697.
9. Markley, F. L. (2003). "Attitude Error Representations for Kalman Filtering." *Journal of Guidance, Control, and Dynamics*, 26(2), 311-317.
10. Shuster, M. D., & Oh, S. D. (1981). "Three-Axis Attitude Determination from Vector Observations." *Journal of Guidance and Control*, 4(1), 70-77.
11. Ozyurt, Y. & Karatas, E. (2024). "A Lost-in-Space Star Identification Algorithm Based on Regularized Dictionary Learning." *Acta Astronautica*, 215, 45-56.
12. Schulz, S., Haghparast, M., & Lindblad, J. (2021). "A UVM-SystemC Framework for Star Tracker Verification." *Sensors*, 21(4), 1396.
13. Hughes, C., Denny, P., Jones, E., & Glavin, M. (2010). "Accuracy of Fish-Eye Lens Models." *Applied Optics*, 49(17), 3338-3347.
14. Richardson, W. H. (1972). "Bayesian-Based Iterative Method of Image Restoration." *Journal of the Optical Society of America*, 62(1), 55-59.
15. Lucy, L. B. (1974). "An Iterative Technique for the Rectification of Observed Distributions." *The Astronomical Journal*, 79(6), 745-754.

Technical References

16. GoPro, Inc. (2023). “GPMF Introduction.” GitHub Repository. <https://github.com/gopro/gpmf-parser>
17. Gyroflow Developers. (2024). “Gyroflow: Video Stabilization Using Gyroscope Data.” <https://gyroflow.xyz/>
18. OpenCV Team. (2024). “OpenCV: Open Source Computer Vision Library.” <https://opencv.org/>
19. Astropy Collaboration. (2022). “The Astropy Project: Building an Open-science Project and Status of the v5.0 Core Package.” *The Astrophysical Journal*, 935(2), 167.

Online Resources

20. Scott’s Astronomy Page. “Astrophotography with a GoPro.” <https://scottsastrpage.com/astrophotography-with-a-gopro/>
21. Cloudy Nights. “Use Your GoPro for Widefield Astrophotography.” <https://www.cloudynights.com/>
22. NightSkyPix. “Astrophotography Stacking Software Guide.” <https://nightskypix.com/astrophotography-stacking-software/>

Hardware References

23. ZWO Astronomy. “ASI585MC/MM Pro Camera Specifications.” <https://www.zwoastro.com/product/asi585mc-mm-pro/>
24. Entaniya Co., Ltd. “Entaniya Fisheye M12 220 S-Mount Lens.” <https://products.entaniya.co.jp/en/list/m12-s-mount-super-wide-fisheye-lens-series/entaniya-fisheye-m12-220-s-mount/>
25. Agena AstroProducts. “ZWO ASI585MC Pro Specifications.” <https://agenaastro.com/zwo-asi585mc-pro-cooled-color-astronomy-imaging-camera.html>
26. Sony Semiconductor Solutions. “IMX585 STARVIS 2 CMOS Image Sensor.” Sony Corporation.
27. AllSkyCams. “All-Sky Camera Systems and Meteor Detection.” <https://www.allskycams.com/>

Sensor Fusion References

28. Laidig, D., & Seel, T. (2023). “VQF: Highly Accurate IMU Orientation Estimation with Bias Estimation and Magnetic Disturbance Rejection.” *Information Fusion*, 91, 187-204.
29. Madgwick, S. O. H. (2010). “An Efficient Orientation Filter for Inertial and Inertial/Magnetic Sensor Arrays.” Technical Report, University of Bristol.
30. Low-Cost Star Tracker Project. “Camera + IMU Sensor Fusion Demonstration.” YouTube. <https://youtube.com/shorts/96TbY9RKdZE>

Motion Deblur References

31. Wiener, N. (1949). “Extrapolation, Interpolation, and Smoothing of Stationary Time Series.” MIT Press.

Appendix A: System Requirements

A.1 Prototype: Harrier 10x + Orange Cube (Validated)

Required Hardware: - Active Silicon Harrier 10x AF-Zoom Camera - Orange Cube Flight Controller (or compatible ArduPilot hardware with ICM-20948) - USB 3.0 port for camera - Serial/USB connection for MAVLink telemetry - 8-core CPU, 3.0+ GHz (for real-time processing) - 16 GB RAM minimum - SSD with 100+ GB free

Recommended Hardware: - Dedicated capture computer (Intel NUC or similar) - NVIDIA GPU with CUDA support for acceleration - Rigid camera-IMU mounting bracket - Stellarium planetarium software for SIL/HIL testing

Software: - Python 3.10+ - MAVLink library (pymavlink) - OpenCV with UVC support

A.2 Alternative A: GoPro-Based System

Minimum Hardware: - GoPro Hero 5 Black or newer (Hero 7+ recommended) - 4-core CPU, 2.0 GHz - 8 GB RAM - 20 GB free storage

Recommended Hardware: - GoPro Hero 7 Black or newer - 8-core CPU, 3.0+ GHz - 16 GB RAM - SSD with 100+ GB free - NVIDIA GPU with CUDA support

A.3 Alternative B: ASI585MC + Entaniya All-Sky System

Required Hardware: - ZWO ASI585MC or ASI585MC Pro camera - Entaniya M12 220 fisheye lens (or compatible M12 mount lens) - External IMU (e.g., BNO055, MPU9250) with rigid mounting - USB 3.0 port (USB 3.1 Gen 1 or better recommended) - 8-core CPU, 3.0+ GHz (for real-time processing) - 16 GB RAM minimum (32 GB for video capture) - SSD with 500+ GB free (high-speed video capture) - 12V DC power supply (for ASI585MC Pro cooling)

Recommended Hardware: - ZWO ASI585MC Pro (cooled version for reduced thermal noise) - Active USB 3.0 cable (5m for outdoor installations) - Weather-resistant enclosure with dew heater - Dedicated capture computer (Intel NUC or similar) - NVIDIA GPU with CUDA support for acceleration - UPS for uninterrupted operation

A.4 Software Dependencies

- Python 3.10+
- NumPy ≥ 1.24.0
- SciPy ≥ 1.10.0
- OpenCV ≥ 4.8.0
- Astropy ≥ 5.3.0
- FFmpeg (external)

Appendix B: Quick Start Guide

B.1 Installation

```
# Clone repository
git clone https://github.com/your-org/lowcost-star-tracker.git
cd lowcost-star-tracker
```

```
# Install dependencies
pip install -e .

# Verify installation
star-tracker --version
```

B.2 Basic Usage

```
# Process a video file
star-tracker process video.mp4 -o output.tiff

# With custom settings
star-tracker process video.mp4 -o output.tiff \
--method sigma_clip \
--min-stars 15 \
--max-fwhm 6.0 \
--reject-percent 25
```

B.3 Configuration File

```
# config.yaml - Prototype (Harrier + Orange Cube)
camera:
  model: "harrier_10x"
  interface: "usb" # or "hdmi"

imu:
  source: "mavlink"
  port: "COM6" # or "/dev/ttyUSB0"
  baud_rate: 115200
  sample_rate: 100

stacking:
  method: "sigma_clip"
  sigma_low: 3.0
  sigma_high: 3.0

output:
  format: "tiff"
  bit_depth: 16

# config.yaml - Alternative (GoPro Hero 7 Black)
camera:
  model: "gopro_hero7_black"
  lens_mode: "linear"

gyro:
  source: "gpmf" # Extract from video file
  sample_rate: 200
  filter_cutoff: 50

stacking:
  method: "sigma_clip"
  sigma_low: 3.0
  sigma_high: 3.0

output:
  format: "tiff"
  bit_depth: 16
```

Appendix C: Glossary

Term	Definition
Centroid	Intensity-weighted center of a star image
FWHM	Full Width at Half Maximum; measure of star image size
GPMF	GoPro Metadata Format; telemetry data container (used by GoPro alternative)
HIL	Hardware-in-the-Loop; testing with real hardware components
Homography	8-DOF projective transformation matrix
ICM-20948	9-axis IMU (gyro+accel+mag) used in Orange Cube flight controller
MAVLink	Micro Air Vehicle Link; communication protocol for telemetry
Quaternion	4-component representation of 3D rotation
RANSAC	Random Sample Consensus; outlier-robust fitting
Sigma-clipping	Statistical outlier rejection method
SIL	Software-in-the-Loop; testing with simulated inputs (e.g., Stellarium)
SLERP	Spherical Linear Interpolation for quaternions
SNR	Signal-to-Noise Ratio
Star tracker	Device that determines orientation from star positions
Stellarium	Open-source planetarium software used for SIL/HIL validation
VQF	Versatile Quaternion-based Filter; sensor fusion algorithm

Document Version: 1.6 Last Updated: January 2026 License: MIT

Changelog

Version 1.6 (January 2026)

- **MAJOR RESTRUCTURE:** Established Harrier 10x + Orange Cube as validated prototype
 - Updated abstract to emphasize light pollution resilience and moving platform support
 - Added Active Silicon Harrier 10x camera specifications (Sony IMX462, 0.0004 lux)
 - Added Orange Cube ICM-20948 IMU specifications
 - Added Stellarium SIL/HIL validation environment documentation
 - Repositioned GoPro and ASI585MC as software-compatible alternatives
- Updated system architecture diagram for prototype configuration
- Added support for seaborne, airborne, and land-based vehicle platforms
- Expanded glossary with SIL/HIL, MAVLink, ICM-20948, Stellarium terms
- Updated cost comparison tables with prototype pricing
- Updated all appendices with prototype-first ordering

Version 1.2 (January 2026)

- **NEW SECTION:** Camera + IMU Sensor Fusion (Section 6)
 - Detailed explanation of rigid camera-IMU coupling
 - Coordinate systems and transformation chain mathematics
 - 3D Celestial Sphere Viewer documentation
 - IMU-camera calibration procedure

- VQF sensor fusion algorithm explanation
- Star-aided orientation correction
- Benefits comparison table (with/without IMU)
- Added YouTube demonstration link: <https://youtube.com/shorts/96TbY9RKdZE>
- Updated table of contents with new section
- Renumbered sections 7-12

Version 1.1 (January 2026)

- Added enhanced configuration: ZWO ASI585MC + Entaniya M12 220 all-sky system
- Added detailed specifications for Sony IMX585 STARVIS 2 sensor
- Added Entaniya fisheye lens optical characteristics and projection model
- Expanded cost comparison to include all-sky camera systems
- Added meteor detection and satellite tracking applications
- Updated system requirements for both configurations
- Added hardware references section

Version 1.0 (January 2026)

- Initial release with GoPro Hero 7 Black configuration
- Complete software pipeline documentation
- Literature review and commercial analysis