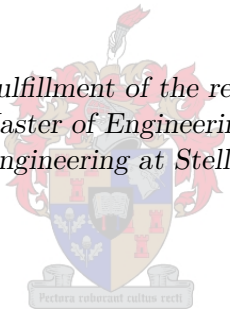
# Augmented Stellar Sensor for a Small Spacecraft

by

Gabriël Johannes Roux




*Thesis presented in partial fulfillment of the requirements for the degree of*
*Master of Engineering*
*in the Faculty of Engineering at Stellenbosch University*

Supervisor:

Prof. W.H. Steyn

Department Electrical and Electronic Engineering

April 2019

# Plagiarism Declaration

1. I have read and understand the Stellenbosch University Policy on Plagiarism and the definitions of plagiarism and self-plagiarism contained in the Policy [Plagiarism: The use of the ideas or material of others without acknowledgement, or the re-use of one's own previously evaluated or published material without acknowledgement or indication thereof (self-plagiarism or text-recycling)].
2. I also understand that direct translations are plagiarism.
3. Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism.
4. I declare that the work contained in this assignment is my own work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

| | |
|---|---|
| ***G. J. Roux*** | ***April 2019*** |
| Initials and Surname | Date |

# Abstract

With the maturity of the CubeSat industry and advancements in commercial off-the-shelf components, CubeSat-based projects have become an attractive option for advanced outer space missions. This increase in mission complexity has given rise to the necessity of a new generation of accurate attitude determination subsystems.

The purpose of this work, therefore, entailed the design and development of an augmented stellar sensor. The focus was not only on the development of a suitable high-performance, low-power hardware platform, but also on the identification, implementation, and development of suitable software techniques as well as the simulation, integration and testing of the augmented platform. This developed sensor delivers accurate attitude and rate estimates, whilst conforming to the small satellite power and size requirements. The augmented system uses inertial rate sensor data, with error compensation performed by use of matched vector measurements obtained from a star sensor. Measurements are combined in an Extended Kalman filter, providing both high rate attitude propagation and bias drift compensation. The designed system features a robust tracking mode as well as a stellar gyro algorithm to deliver accurate, low-frequency rate estimates independent of host dynamics.

To prove overall system functionality, the sensor has undergone verification during simulated conditions, testing in an in-house developed star emulation environment, as well as testing under night sky conditions. During these tests, it was exposed to conditions typically experienced by satellites throughout their mission lifetimes. These conditions range from low-rate tumbling, to fine pointing.

Initial testing shows that the system offers a robust response regardless of satellite rate and orientation whilst simultaneously adhering to CubeSat standards. IMU bias compensation worked successfully, and estimated results show that the average $3\sigma$ stellar gyro rate accuracies were in the order of $0.01\,^{\circ}/\mathrm{s}$ whilst the cross-axis $3\sigma$ orientation accuracy was close to $0.01^{\circ}$ during low rates.

# Uittreksel

Met die volwassewording van die CubeSat-industrie en vooruitgang van kommersieël beskikbare elektroniese komponente, het die CubeSat-platform 'n aantreklike keuse geword vir ruimtevaartsendings. Hierdie belangstelling in die CubeSat-platform het tot 'n vermeerdering van sendingskompleksiteit gelei wat die behoefte vir 'n nuwe generasie akkurate oriëntasiebeheer-substelsels geskep het.

Gevolglik was die doel van hierdie werk die ontwerp en ontwikkeling van 'n uitgebreide stersensor. Die fokus was egter nie net om 'n gepaste hardewarestelsel te ontwerp nie, maar ook om geskikte sagtewaretegnieke en algoritmes te identifiseer, te ontwikkel, en toe te pas. Die ontwikkelde stelsel lewer akkurate oriëntasie- en hoeksnelheidafskattings, terwyl dit geskik vir gebruik in 'n nanosatelliet is. Hierdie uitgebreide stelsel gebruik inersiële sensormetings waarop foutkorrigering, soos afgeskat deur middel van vektorinligting vanaf 'n sterkamera, toegepas is. Die sensormetings word gekombineer in 'n uitgebreide Kalman filter, wat beide hoë-frekwensie oriëntasie-afskattings kan verskaf, sowel as om die inersiële sensor foutkorrigering te beheer. Die ontwerpte stelsel bevat verder 'n robuuste stervolgmodus om die mikroverwerker se berekeninge te verminder, sowel as 'n hoeksnelheid-afskattingsalgoritme om baie akkurate lae-frekwensie afskattings te bied. Die laasgenoemde algoritme kan onafhanklik van 'n dinamiese model funksioneer.

Om die oorhoofse stelsel se werking te bevestig, is die sensor tydens gesimuleerde, geëmuleerde, en praktiese omstandighede getoets. Gedurende hierdie toetse is die stelsel blootgestel aan tipiese gebruikstoestande soos lae-snelheid tuimel en fyn oriëntasiebeheer.

Aanvanklike toetse wys dat die stelsel goed werk ongeag die hoeksnelheidstoestande waaraan dit blootgestel word. Inersiële sensor hoeksnelheidsmetings kon suksesvol gekorrigeer word. Afgeskatte resultate toon daarop dat 'n stervektor se hoeksnelheid oor die kruisas akkuraat tot $0.01\,^\circ/\mathrm{s}$, op die $3\sigma$-vlak, afgeskat kon word. Resultate aangaande oriëntasie-akkuraatheid was in die orde van $0.01^\circ$, $3\sigma$, oor die kruisas tydens 'n lae hoeksnelheid.

# Scientific Contributions

A preliminary account of this work was presented as an oral presentation at the following international conference:

Roux G.J, Steyn W.H., "A Novel High-Performance Nanosatellite Attitude and Rate Sensor", 69th International Astronautical Congress, Bremen, Germany, 1-5 October 2018.

# Acknowledgements

I would like to thank the following;

- Professor Steyn, for his invaluable guidance;
- Riccardo Swanepoel, for his superior language and editing skills;
- My family and friends for their constant support and motivation;
- My ESL colleagues, as well as Denise Crowley and Beryl Sameuls, for the constant, much-needed distractions;
- Johan Arendse, for saving my skin more than once with a well-aimed wire-mod;
- Wessel Croukamp, for his help involving everything technical;
- The CubeSpace staff members for all their help, especially Christo Groenewald, without whose assistance and constant motivation I would not have gotten very far;
- Dr Willem Jordaan, for the invaluable conversations and introducing me to the world of satellites;
- SANRAL, for their funding during both undergraduate and postgraduate studies.

# Contents

# List of Figures

# List of Tables

# Nomenclature

### Abbreviations and Acronyms

| | |
|---|---|
| ADC | Analog-to-Digital Converter |
| ADCS | Attitude Determination and Control System |
| APS | Active Pixel Sensor |
| ARW | Angle Random Walk |
| BI | Bias Instability |
| BSV | Boresight Vector |
| CAN | Controller Area Network |
| CASC | Closest Approximate Sample Centre |
| CCD | Charge-Coupled Device |
| DCM | Direction Cosine Matrix |
| DOF | Degrees of Freedom |
| DSP | Digital Signal Processing |
| ECI | Earth-Centered Inertial |
| EKF | Extended Kalman Filter |
| EMI | External Memory Interface |
| ESL | Electronic Systems Laboratory |
| FOG | Fibre-Optic Gyro |
| FOV | Field of View |
| FPGA | Field Programmable Gate Array |
| FPU | Floating Point Unit |
| I2C | Inert-Integrated Circuit |
| IMU | Inertial Measurement Unit |
| ISC | Inertial Stellar Compass |
| LEO | Lower-Earth Orbit |
| LIS | Lost-in-Space |
| LSQ | Least Squares |
| MCU | Microcontroller Unit |
| MEMS | Micro Electromechanical System |

**Abbreviations and Acronyms (Continued)**

| | |
|---|---|
| OBC | On-Board Computer |
| PCB | Printed Circuit Board |
| PSF | Point-Spread Function |
| ROI | Region of Interest |
| RRW | Rate Random Walk |
| SPI | Serial Peripheral Interface |
| SRAM | Static Random Access Memory |
| STEVE | Star Tracker Evaluation Environment |
| UART | Universal Asynchronous Receiver Transmitter |

**Greek Letters**

| | |
|---|---|
| $\alpha$ | Generic angle |
| $\delta, \Delta$ | Incremental value operator |
| $\eta$ | Variable relating to a random process |
| $\epsilon$ | Distortion function |
| $\lambda$ | Eigenvalue |
| $\mathbf{\Xi}$ | Matrix quantity defined as in Equation (2.4.14) |
| $\xi$ | Maximum permissible interstellar angular error |
| $\sigma$ | Standard deviation |
| $\tau$ | Emulated sampling period used in Allan Variance |
| $\mathbf{\Psi}$ | Transition Matrix |
| $\omega$ | Angular Rate |

**Lowercase Letters**

| | |
|---|---|
| $\mathbf{b}$ | Vector containing raw gyro measurements $b_x, b_y, b_z$ |
| $c$ | Numerical constants |
| $d$ | Interstellar distance in radians |
| $f$ | Camera focal length |
| $i, j, k$ | Indices |
| $n$ | Integer signifying group size in Allan Variance |
| $\mathbf{q}$ | Represents a quaternion defined as in Equation (2.4.7) |
| $r_\square$ | Radial distance with origin denoted by subscript |
| $t$ | Time |
| $\mathbf{u}$ | Vector containing raw gyro measurements $u_x, u_y, u_z$ |
| $\mathbf{v}$ | Star vector containing components $v_x, v_y, v_z$ |

### Uppercase Letters

| | |
|---|---|
| $\mathbf{A}$ | Attitude matrix |
| $I$ | Star intensity |
| $\mathbf{I}$ | Identity matrix |
| $N$ | Number |
| $R$ | Geometric voting uncertainty region |

### Subscripts

| | |
|---|---|
| $\omega$ | Pure imaginary rate quaternion |
| $\mathcal{B}$ | Body reference frame |
| $\mathcal{G}$ | Gyro reference frame |
| $\mathcal{I}$ | Inertial reference frame |
| $\mathcal{T}$ | An intermediate reference frame |
| $bsv$ | Represents a star located at the image plane centre |
| $c$ | Star centroid |
| $o$ | Reference frame origin |
| $r$ | Radial direction |
| $t$ | Tangential direction |
| $v$ | Vector |

### Syntax and Style

| | |
|---|---|
| $X, Y, Z$ | Axes of the inertial reference frame |
| $x, y, z$ | Axes of the body reference frame |
| $u, v$ | Axes of the pixel frame |
| $\psi, \phi, \theta$ | Euler angles: yaw, pitch, roll |
| $\mathbf{K}, \mathbf{H}, \mathbf{P}, \mathbf{Q}, \mathbf{R}$ | Kalman filter matrices |
| $\mathbf{x}, \mathbf{e}$ | Kalman filter vectors |
| $K, P$ | Distortion parameters |
| $s, \mathbf{z}, \mathbf{B}, \mathbf{C}$ | Quantities related to QUEST |
| $\mathbf{r}, \mathbf{s}, \mathbf{w}$ | Vectors related to the TRIAD method |

# Chapter 1

# Introduction

## 1.1 Background

In recent years the CubeSat industry has evolved from a small area of research primarily focused on education, to that of a full-blown commercial industry. With over 850 CubeSat missions launched since the year 2000, a definite trend can be seen in the increase of mission complexity. Despite its increased popularity, however, the standard enforced by the CubeSat launching system constrains the size and available power of these nanosatellite missions and, more importantly, limits the hardware that can be flown. The necessity of increased innovation and miniaturisation in the field of satellite Attitude Determination and Control Systems (ADCSs) is therefore evident.

With the advent of embedded electronic miniaturisation, it has for the first time become possible to develop small, autonomous sensors that can perform functions akin to those of fully fledged ADCSs. As discrete component power requirements decrease, so to do the limits of sensor integration.

The ADCS community is by no means new to the integration of multiple sensors, as single-sensor attitude determination is rarely found during satellite missions. The fusing of sensor data is prominent, primarily because it allows for the mitigation of inherent sensor flaws. An example of sensors often used together is the star tracker and gyroscope. Although both technologies have served as cornerstones in solving the problem of attitude determination and propagation, each introduces its own complexity in system integration, and can only be used under certain system conditions.

In the case of star trackers, unique attitude solutions can only be computed when bright objects are not in the Field of View (FOV), and when host slew rates are low enough that individual stars can be observed. Although gyros can then help by providing state propagation during these invalid star tracker conditions, gyro measurements tend to be inaccurate and lead to a random walk process in the estimated attitude, causing a loss in attitude accuracy.

## 1.2 Research Aim

To solve this problem, specifically on a nanosatellite scale, the existing CubeStar nano star tracker platform will be extended so that a preintegrated nanosatellite attitude and rate determination system is developed. This project therefore not only entails the physical integration and augmentation of the current CubeStar platform with a rate-measurement device, but also treats the investigation of methods of successful low-level system integration, as well as algorithm development for use on a star tracker-based attitude and rate system.

The aim of this project can therefore be translated into the following technical requirements, such that a system is designed where:

1. The system delivers both attitude and rate measurements;
2. The rate measurements are free of biases;

3. It operates efficiently by use of a reduced-search tracking mode;
4. It exhibits low power usage, preferably under 0.5 W;
5. Stellar gyro functionality is added to provide high-accuracy rate measurements when star tracker data are available; and
6. It provides a preintegrated state determination system, delivering attitude estimates, bias-compensated rate data from an Inertial Measurement Unit (IMU), and high-accuracy rate estimates from a stellar gyro estimation.

These technical requirements will be achieved by completing the following objectives:

1. Designing a suitable hardware platform;
2. Investigating software techniques pertaining to a star tracker as well as attitude, rate, and bias estimation;
3. Implementing, testing, and verifying the identified software techniques in a simulation environment;
4. Integrating and verifying the hardware and software interfaces; and
5. Testing and validating the system under true night sky conditions

## 1.3 Thesis Roadmap

This thesis is organised according to the objectives, as presented in the following chapters:

### Literature Study

Theoretical concepts with regards to reference frames, attitude representations and some basic astronomy concepts are given. Thereafter, a brief review of current state-of-the-art technology is given in context of stellar, and inertial navigation as well as stellar gyro systems.

### Hardware Design

The initial hardware design of the CubeStar–IMU interface is treated, along with design decisions regarding the replacement of the original CubeStar Microcontroller Unit (MCU) and hardware system integration.

### Software and Algorithms

A brief discussion of a commonly used star model is given, after which some key algorithms pertaining to image processing, and star matching during Lost-in-Space (LIS) and tracking modes are identified. Finally, some commonly used attitude and rate estimation algorithms and an initial system software implementation are discussed.

### Simulation

The attitude and bias Extended Kalman Filter (EKF) simulation and initial observations are given. Thereafter, simulation results of the complete designed system are treated, with some main software design decisions and a discussion of further system software integration following.

### Practical Testing and System Integration

A star emulation environment is developed that is used during initial testing and algorithm verification. Sensor calibration and the final system hardware and software integration are then treated.

### Results and Discussion

Practical testing during earth-fixed and dynamic system conditions are conducted, results are given, and some discussions and observations are considered to serve as functional system validation.

### Conclusions and Recommendations

Concluding remarks with regards to the developed system are given. Afterwards, some recommendations related to future work are made.

# Chapter 2

# Literature Study

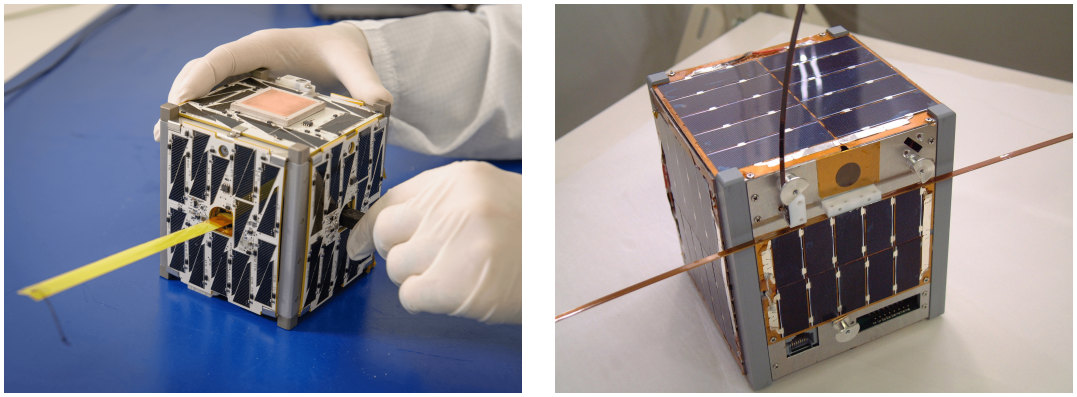
In this chapter, the basic literature pertaining to the theoretical background and state of the art are given. This section starts with important background concepts in Sections 2.1 to 2.5 after which state-of-the-art research in the field of stellar gyro systems is treated in Section 2.6

## 2.1 CubeSats

With the development of high-performance microelectronic systems, the nanosatellite industry has become a prominent contender in the commercial space market. One of the best known nanosatellite architectures is the CubeSat. This architecture standard was originally developed by Professors Bob Twiggs and Jordi Puig-Suari to enable hands-on satellite mission design and operation experience in universities.

This standard defines a one unit (1U) CubeSat as a 10 cm cube weighing a maximum of 1.33 kg [1]. Key to this technological standard, is the P-POD, or Poly Picosatellite Orbital Deployer mechanism, developed at California Polytechnic State University. This deployer system consists of a square tube, delivering minimal deployment spin, with a linear velocity of around 0.3 m/s, whilst offering an inexpensive, modular solution [1].

Since the original development of CubeSats, the standard has grown to include various other sizes of nanosatellites, including, but not limited to 1.5U, 2U, 3U, and 6U [2]. Figure 2.1 shows two 1U CubeSats: PhoneSat 2.5, which was launched in 2013, and OSCAR CO-57.



**(a)** PhoneSat 2.5 [3] **(b)** OSCAR CO-57 [4]

**Figure 2.1:** CubeSats

Similar to their larger counterparts, CubeSats also consist of similar standard satellite subsystems, such as: ADCSs, propulsion, power, and communications, amongst others. Owing to the short development cycle, high flexibility and relative inexpensiveness due to the use of commercial

off-the-shelf components, they are often used as a means of new technology demonstration and testing.

Over the past twenty years of development, the usage of CubeSats for Lower-Earth Orbit (LEO) missions have shown an exponential increase [5]. Despite the original CubeSat purpose, functionality and mission complexities planned for future, and even present CubeSat missions now far extend that of only education and training.

An example of this is seen in the recent deployment of the Mars Cube One (MarCO) satellite [6]. This nanosatellite mission, launched in May of 2018, is to demonstrate the usage of a small spacecraft for an interplanetary mission, and will function as a communications relay service for the InSight Mars lander.

MarCO is not the only planned deep-space CubeSat mission, however, as Exploration Mission-1, scheduled for launch in June of 2020, also aims to demonstrate the increased reliance on the CubeSat platform, in this case, towards the exploration of the lunar environment. This mission, as secondary payload, contains thirteen low-cost CubeSats to be used for various scientific and exploration purposes [7], [8].

With this increase in mission complexity and reliance on the CubeSat platform, the overall system requirements have reached a critical point in the design of subsystems. As this satellite architecture is therefore reaching maturity, and their usage extend that of LEO, so too does the payload pointing requirement, and in effect the ADCS.

To achieve these pointing requirements, high-accuracy vector measurements are required to relate a satellite to some inertial frame of reference. The most accurate of these vector measurements are obtained from observing stars. Some basic knowledge of astronomy is therefore required.

## 2.2 Astronomy Concepts

To ensure accurate ADCS pointing capability, knowledge of the celestial sphere, stars, and star catalogues are required.

### 2.2.1 Star Catalogues

Star catalogues are lists of stars that have been observed, and contain information about the position, movement, and observable properties thereof. The positions of some of the brightest stars are shown in Figure 2.2.

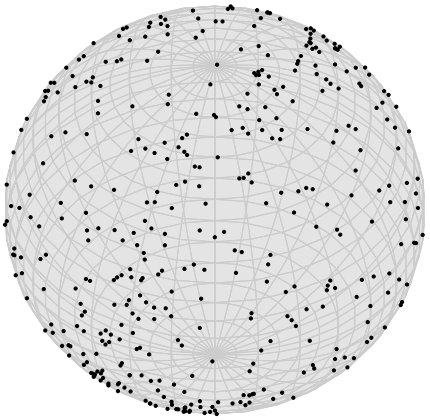


**Figure 2.2:** Celestial Sphere Showing Stars with Visual Magnitudes Greater than $m_v = 3.8$

Shown here are all the stars brighter than a star with a visual magnitude of 3.8, as points on the celestial sphere. The star locations shown here come from the Hipparcos catalogue [9]. This

catalogue was originally generated from observations made during the Hipparcos mission between 1989 and the first quarter of 1993 [10]. The generated catalogue had an epoch of J1991.25 and contained around 118,218 entries to a positional precision of 1 milli-arcsec. The Hipparcos catalogue was later extended with data obtained from the star mapper hardware on board the Hipparcos satellite. This extended catalogue was called the Tycho catalogue, and although slightly less accurate than Hipparcos, contained over one million objects. Both the Hipparcos and Tycho catalogues contain accurate astrometry and calibrated photometry data, as well as star positions, parallaxes, and proper motions. A common way of classifying the stars contained in these catalogues, is according to their magnitudes.

### 2.2.2 Star Magnitude Scale

Star intensity is measured on a stellar magnitude scale. Although various magnitude scales, based on the photometric band of the observer, exist, the most common classifications are based on the visual magnitude or the absolute magnitude.

#### Visual Magnitude Scale

In this scale, the visual magnitude of a star is defined in terms of some reference magnitude, given as.

$$m_v = -2.5 \log_{10} \left( \frac{I_p}{I_0} \right) \tag{2.2.1}$$

In this equation, $I_p$ is the observed light intensity, described in terms of flux density, and $I_0$ the reference or zero point. The apparent magnitude scale is an inverse logarithmic scale, defined such that a star with visual magnitude of $m_v = 1$ is about five times brighter than stars with magnitudes of $m_v = 5$. This convention was chosen to correspond with historical data [11].

#### Absolute Magnitude Scale

The visual brightness scale takes into account only the perceived brightness of a celestial object and not the brightness intrinsic to the star. Subsequently, as the perceived brightness is linked to the square of the distance to the source, astronomers developed the absolute magnitude scale. This scale takes into account not only the visual magnitude, but also the distance to the objects. The absolute magnitude scale is defined as the apparent magnitude a star would exhibit, if seen at ten parsecs away, under zero diffraction conditions.

## 2.3 Coordinate Systems

Also of importance in the interpretation of star catalogues is the definition of a frame of reference in which the orientation of a star and the observer can be defined. In this work there are two reference frames that feature regularly, namely the spacecraft-centred inertial reference frame, and sensor-body reference frame. Although other coordinate systems, such as satellite body or orbit-referenced frames, are also used in spacecraft navigation, they are not necessary for the understanding of the following concepts and are treated elsewhere [12].

### 2.3.1 Spacecraft-Centred Inertial Coordinates

As described in Wertz et al. [12], the spacecraft-centred inertial, or inertial for short, coordinate system is defined relative to Earth's rotation axis, with the positive $Z$-direction towards the north celestial pole. Here, the $X$-axis of this reference frame points in the direction to where the ecliptic crosses the equatorial plane on the first day of spring. This point is also known as the vernal equinox. The $Y$-axis is then chosen to complete the right-handed axis system. Figure 2.3 shows the spacecraft-centred inertial frame.

Owing to the precession of the Earth's axis around the ecliptic pole, this inertial frame is not fixed relative to the mean position of the visible stars [12]. Because of this phenomenon, fixed

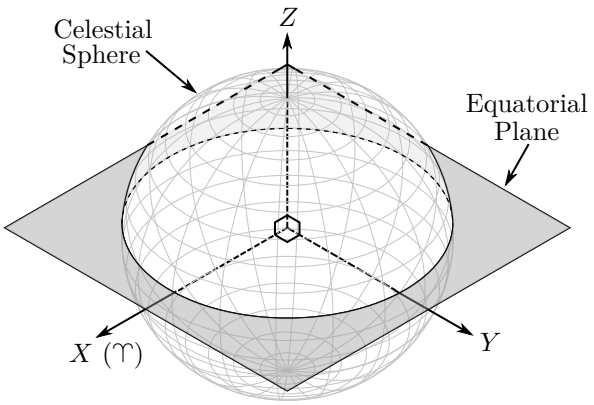



**Figure 2.3:** Spacecraft-Centred Inertial Coordinates

points on the celestial sphere move. To compensate for this precession, an epoch is usually coupled to the quasi-inertial frame to ensure accuracy during attitude determination [13].

In the spacecraft-centred inertial coordinate system, landmark positions are usually specified in the form of Cartesian or spherical vectors. To simplify the observation of celestial bodies, all stars in the region of the sun are assumed to be distributed across a unit sphere, centred around the spacecraft.

### 2.3.2 Sensor-Body Coordinates

In the case of an optical sensor, the sensor-body reference frame is chosen to be centred on the sensing element, with the positive $z$-axis pointing in the direction the sensor is facing. The $x$-axis is then chosen to correspond with that of the sensing element's $x$-axis, with the sensor-body $y$-axis completing the right-handed axis system. Figure 2.4 shows a star camera mounted to a satellite. The star camera's body reference frame is shown to be at the centre of the sensor.

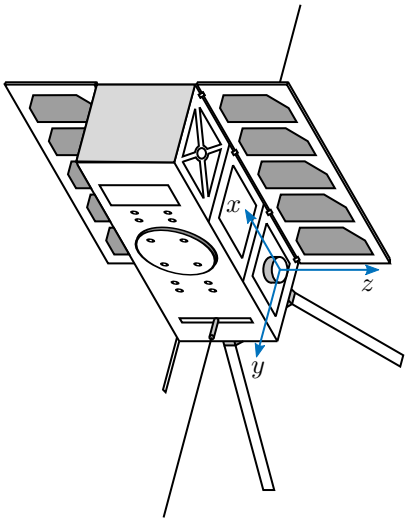



**Figure 2.4:** 3U CubeSat with Star Tracker Demonstrating Sensor-Body Coordinates

With these coordinate systems, it is possible to define the attitude of an observer, relative to that of the inertial reference frame.

## 2.4 Attitude Representations

The attitude of a rigid body can be represented in numerous ways, of which three are of importance in this work. The three important parameterisations are: rotation matrices, Euler angles, and quaternions. Each of these attitude representations has its own preferred applications, advantages and disadvantages [14].

### 2.4.1 Rotation Matrices

A rotation, or attitude, matrix is an orthogonal $3 \times 3$ matrix that defines the relative rotation between two frames of reference. In this case, a body vector, $\mathbf{v}^{\mathcal{B}}$, is related to its inertial counterpart, $\mathbf{v}^{\mathcal{I}}$, by:

$$\mathbf{v}^{\mathcal{B}} = \mathbf{A}_{\mathcal{I}}^{\mathcal{B}} \mathbf{v}^{\mathcal{I}} \tag{2.4.1}$$

where $\mathbf{A}_{\mathcal{I}}^{\mathcal{B}}$ defines the inertial to body attitude matrix. It should, however, be noted that the vector norm stays constant such that:

$$||\mathbf{v}^{\mathcal{B}}|| = ||\mathbf{v}^{\mathcal{I}}|| \tag{2.4.2}$$

A rigid body can also be subject to multiple rotations:

$$\mathbf{v}^{\mathcal{I}} = \mathbf{A}_{\mathcal{O}}^{\mathcal{I}} \mathbf{A}_{\mathcal{B}}^{\mathcal{O}} \mathbf{v}^{\mathcal{B}} = \mathbf{A}_{\mathcal{O}}^{\mathcal{I}} \mathbf{v}^{\mathcal{O}} \tag{2.4.3}$$

Operations regarding multiple rotation matrices are not commutative, however, as rotations have to be applied in a certain order to achieve the required attitude representation. The matrices that describe a rotation of $\alpha$ degrees around each axis are given in Equation (2.4.4)

$$A_1(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}, \quad A_2(\alpha) = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix}$$

$$A_3(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4.4}$$

Here, the enumeration 1, 2, and 3 describes the $x$- , $y$-, and $z$-axes, respectively. An attitude matrix further also encodes the reverse of the rotation as

$$\left(\mathbf{A}_{\mathcal{B}}^{\mathcal{I}}\right)^{-1} = \left(\mathbf{A}_{\mathcal{B}}^{\mathcal{I}}\right)^{T} = \mathbf{A}_{\mathcal{I}}^{\mathcal{B}} \tag{2.4.5}$$

Despite the simple, intuitive parameterisation that attitude matrices offer, they require nine variables and have an orthogonality constraint which is difficult to obey during estimation as numerical rounding errors can accumulate.

### 2.4.2 Euler Angles

A different method in describing the attitude of a rigid body is by use of Euler angles. Euler angles, classically known as the pitch, roll, and yaw of a vehicle, describe orientation by use of only three angles, thus leading to a much more efficient parameterisation method. However, this description contains mathematical singularities, making it undesirable for use in applications where operation close to singular regions are possible. Although there are twelve different Euler representations, these can all be subdivided into two main types:

#### Type 1

These angles are also called Cardan angles and involve a rotation around each of the distinct axes. An example of this is the 3-2-1, or $z$-$y$-$x$ sequence. The singularity for this attitude description occurs at a pitch angle of $\theta = n\pi$ for $n \in \mathbb{Z}$. This is illustrated in Figure 2.5.

**Type 2**

Repeated axis parameterisations involve multiple rotations around a single axis, of which one of the most common sequences is the 3-1-3 or $z''$-$y'$-$z$ sequence [12]. The singularity for this type of Euler angle occurs at $\theta = \frac{\pi}{2} + n\pi$ for $n \in \mathbb{Z}$ [14].

Figure 2.5 shows the three different Euler rotations, namely a rotation around the $x$-axis (Figure 2.5a), the $y$-axis (Figure 2.5b), and the $z$-axis (Figure 2.5c).

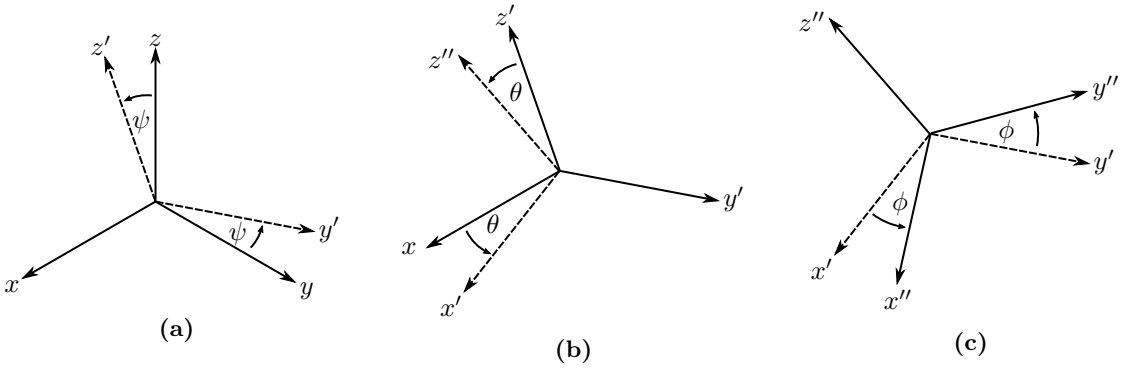



**Figure 2.5:** Euler Sequence 3-2-1

An attitude parameterised with pitch, roll and, yaw angles, ($\phi$, $\theta$, and $\psi$, respectively) can also be represented as a rotation matrix constructed from the three matrices described in Equation (2.4.4) such that

$$\mathbf{A}_{321}(\phi, \theta, \psi) = \mathbf{A}_3(\phi)\mathbf{A}_2(\theta)\mathbf{A}_1(\psi) \tag{2.4.6}$$

Despite its convenience, Euler angles cause a condition known as Gimbal Lock [14]. This occurs when a system able to rotate with three degrees of freedom enters an orientation where it is forced into a two-dimensional space. In this locked configuration, changes to two or more of the gimbals will apply the same rotation to the rotating body causing an indeterminable attitude.

### 2.4.3 Quaternions

An alternative method of describing attitude was proposed by William Hamilton in 1843. His solution, the quaternion, involves a four-element attitude parameterisation representing the relationship between two vectors in three-dimensional space. The Hamilton quaternion, $\mathbf{q}$, defined as,

$$\mathbf{q} = q_4 + q_1 i + q_2 j + q_3 k \quad \Leftrightarrow \quad \mathbf{q} = q_4 + \mathbf{q_v} \tag{2.4.7}$$

is traditionally described as the sum of a pure imaginary vector and scalar weighting factor. For the sake of convenience, this text uses the Jet Propulsion Laboratory (JPL) vector notation. This notation allows quaternion operations to be simplified to matrix and vector operations. A JPL quaternion is defined as

$$\mathbf{q} \triangleq \begin{bmatrix} \mathbf{q}_v \\ q_4 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \tag{2.4.8}$$

Figure 2.6 shows a graphical approach to understanding the quaternion convention. Here, the rotation from $\mathbf{v}^{\mathcal{I}}$ to $\mathbf{v}^{\mathcal{B}}$ is parametrised by a quaternion that $q_4 = \cos\left(\frac{\alpha}{2}\right)$ and $\mathbf{q}_v = \sin\left(\frac{\alpha}{2}\right)\hat{\mathbf{u}}$. In this case, the rotation applied to the vector, $\mathbf{v}^{\mathcal{I}}$, can be described by

$$\mathbf{v}^{\mathcal{B}} = \mathbf{q}\mathbf{v}^{\mathcal{I}}\mathbf{q}^* = \mathbf{A}_{\mathcal{I}}^{\mathcal{B}}\mathbf{v}^{\mathcal{I}} \tag{2.4.9}$$

where the corresponding attitude matrix can be calculated from the quaternion as

$$\mathbf{A}(\mathbf{q}) = (|q_4|^2 - |\mathbf{q}_v|^2)\mathbf{I}_{3\times 3} + 2\mathbf{q}_v\mathbf{q}_v^T + 2q_4[\mathbf{q}_v]_\times^T \tag{2.4.10}$$

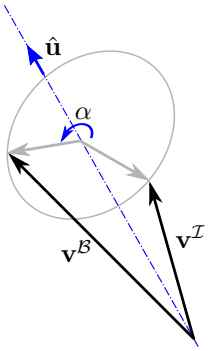



**Figure 2.6:** Rotation Described by a Quaternion

and is equal to

$$\mathbf{A} = \begin{bmatrix} -q_3^2 - q_2^2 + q_1^2 + q_4^2 & 2q_1q_2 - 2q_3q_4 & 2q_1q_3 + 2q_2q_4 \\ 2q_3q_4 + 2q_1q_2 & -q_3^2 + q_2^2 - q_1^2 + q_4^2 & 2q_2q_3 - 2q_1q_4 \\ 2q_1q_3 - 2q_2q_4 & 2q_2q_3 + 2q_1q_4 & q_3^2 - q_2^2 - q_1^2 + q_4^2 \end{bmatrix} \tag{2.4.11}$$

Furthermore, it should also be noted that, similar to rotation matrices, quaternions also encode the reverse of the rotations they describe:

$$\mathbf{q}_{\mathcal{I}}^{\mathcal{B}} = \left(\mathbf{q}_{\mathcal{B}}^{\mathcal{I}}\right)^* = \begin{bmatrix} -\mathbf{q}_v \\ q_4 \end{bmatrix} \tag{2.4.12}$$

**Some Useful Quaternion Definitions**

In this text, the following definitions derived from quaternion mathematics are used. Sequential rotations, defined by compound quaternions, are given as:

$$\mathbf{A}(\mathbf{q}')\mathbf{A}(\mathbf{q}) = \mathbf{A}(\mathbf{q}' \otimes \mathbf{q}) \tag{2.4.13}$$

Also useful, especially for the understanding of the Kalman filter as described in Section 4.7, is the matrix quantity, $\mathbf{\Xi}(\mathbf{q})$, defined as [15]:

$$\mathbf{\Xi}(\mathbf{q}) = \begin{bmatrix} q_4 & -q_3 & q_2 \\ q_3 & q_4 & -q_1 \\ -q_2 & q_1 & q_4 \\ -q_1 & -q_2 & -q_3 \end{bmatrix} \tag{2.4.14}$$

Lastly, the four-element pure imaginary rate quaternion is given as:

$$\mathbf{q}_\omega = \begin{bmatrix} \boldsymbol{\omega} \\ 0 \end{bmatrix} \tag{2.4.15}$$

By using these attitude definitions, it is possible to measure and control the attitude of a satellite by use of an ADCS.

## 2.5 Attitude Determination and Control Systems

The goal of an ADCS is twofold: firstly it involves the *estimation* of a spacecraft's orientation relative to an inertial frame or point of interest [13]; and secondly, it involves the *control* of the spacecraft's orientation through means of actuation.

The simplest form of ADCSs is passive. These systems implement attitude control through spinning, or by use of the earth's magnetic or gravity fields. Although these methods are sufficient in low-performance systems, they do not offer the high pointing accuracy required in more complex applications. In high-performance ADCSs, active orientation control through means of orientation sensing is required [13].

By definition, high-performance attitude control is directly linked to the accuracy of attitude estimation, which is impacted by the reference vectors used in the attitude sensing process. Important to note is the that direction of these reference vectors is of key importance, lesser so the vector magnitude [12].

The main reference vectors used in ADCSs, along with their advantages and disadvantages, as identified by Wertz et al. [12] is shown in Table 2.1.

**Table 2.1:** Reference Vectors Used in Attitude Determination [12]

| Reference | Advantages | Disadvantages |
|---|---|---|
| Sun | Bright; unambiguous; low power; low weight | Not always visible; accuracy limit of 0.016° |
| Earth | Simple analysis; always available; largely unambigious | Prone to sun interference; resolution limit of 0.1° owing to horizon definition |
| Magnetic Field | Economical; low power; always available at low attitudes | Accuracy in the order of 0.5°; dependent on spacecraft magnetism; bias sensitive |
| Stars | Very accurate; available almost everywhere in sky | Complex and expensive; power intensive; requires sun shading |

Typically, in low-power, low-performance satellites, magnetic sensing and magnetic control are sufficient. However, to ascertain very high-performance attitude estimation, more accurate sensors are needed. From this table, it is shown that star trackers offer the highest accuracy, at the cost of power and capital. These star trackers are treated in the next subsection.

### 2.5.1 Star Trackers

A star tracker is a spacecraft attitude sensor, consisting of an optical assembly coupled with external electronics, typically used in spacecraft navigation. Traditionally, star trackers can be subdivided into three types: star scanners, gimballed star trackers, and fixed-head star trackers [12]. With initial advancements in digital electronics, first-generation fixed-head star trackers have become much more powerful, showing observed accuracies in the order of arcseconds [16]. The general system description of a fixed-head star tracker is shown in Figure 2.7.

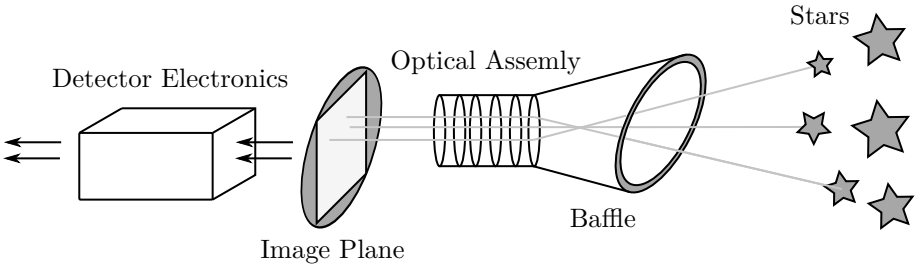



**Figure 2.7:** Basic Star Tracker Components, Adapted from [12]

The first generation of these fixed-head star trackers, developed in the early 1970s at NASA JPL, were of Charge-Coupled Device (CCD) technology. These star trackers were able to track only a few stars, and were completely dependent on external processing for star acquisition, distortion compensation, and data processing [17]. With regards to performance, a maximum of six stars could be detected in each frame [17], with the instrument only being able to output CCD coordinates corresponding to the location of the star spots. These data could then be processed on

board, or later on the ground. Not only did the first fixed-head trackers have limited processing ability, but they were also heavy and rather power intensive [17].

Star tracker technology has since come a long way, especially with the increase in microprocessor capability and the development of Active Pixel Sensor (APS) sensing elements. Owing to this capability increase, second-generation star trackers are able to feature pattern recognition algorithms as well as large internal catalogues, allowing increased autonomy and On-Board Computer (OBC) offloading through means of distributed processing. These star trackers have further seen a significant decrease in mass and power usage, with some of the early second-generation star trackers weighing as little as 1 kg whilst using 7 W only [18]. In recent years, star tracker technology has also been further extended to the CubeSat industry, albeit with limited performance.

### 2.5.2 Current CubeSat Star Tracker Capabilities

Table 2.2 shows a nonexhaustive list of some of the star tracker sensors prominently available.

**Table 2.2:** Current Stance of CubeSat Star Tracker Technology

| | **MAI-SS** [19] | **NST-4** [20] | **ST200** [21] | **CubeStar** [22] | **BCTNST** [23], [24] |
|---|---|---|---|---|---|
| **Company** | Maryland | TY-Space | Berlin Space | CubeSpace | BCT |
| **Accuracy** (arcsec) | | | | | |
| Cross-axis | 4 | 5 | 30 | 77.4 | 9 |
| Roll-axis | 27 | 70 | 200 | 219.6 | 60 |
| **Slew** (°/s) | $> 2$ | 2 | 0.3 | — | $> 2$ |
| **Physical** | | | | | |
| Size ($cm^3$) | 117.5 | 107.5 | 32 | 96.25 | 275 |
| Mass (g) | 170 | 107 | 42 | 55 | 350 |
| **Electrical** | | | | | |
| Voltage (V) | 5 | 5 | 3.7 to 5 | 3.3 | 5 |
| Power (W) | 1.5 | 0.6 | 0.65 | 0.142 | 0.75 |
| Interface | UART/I2C | RS422/CAN | I2C/UART | I2C | RS-422 |
| **Sensor** | | | | | |
| Update (Hz) | 4 | 10 | 5 | 1 | 5 |
| Sensitivity (Mv) | 6 | 5.8 | — | 3.8 | 7.5 |
| Acquisition (s) | 0.13 | 2 | — | 1 | 4 |
| **Environmental** | | | | | |
| Thermal (°C) | -40 to 80 | -40 to 40 | -20 to 40 | -10 to 60 | — |
| Radiation (kRad) | 75 | 30 | 9 | TBD | 11-16 |
| Sun Exclusion (°) | 45 | $> 25$ | 30/45 | | 45 |

From the five identified star tracker modules, it can be noted that there are some definite technological trends. Overall, the average star tracker power usage tends to be quite high, with requirements upwards of 600 mW. In stark contrast to this, stands the CubeStar star tracker module. Although very low power, it does not hold its own in terms of accuracy or sensitivity. It is therefore evident that a star tracker module's power usage is strongly coupled with its attitude determination accuracy and the update rate of attitude solutions.

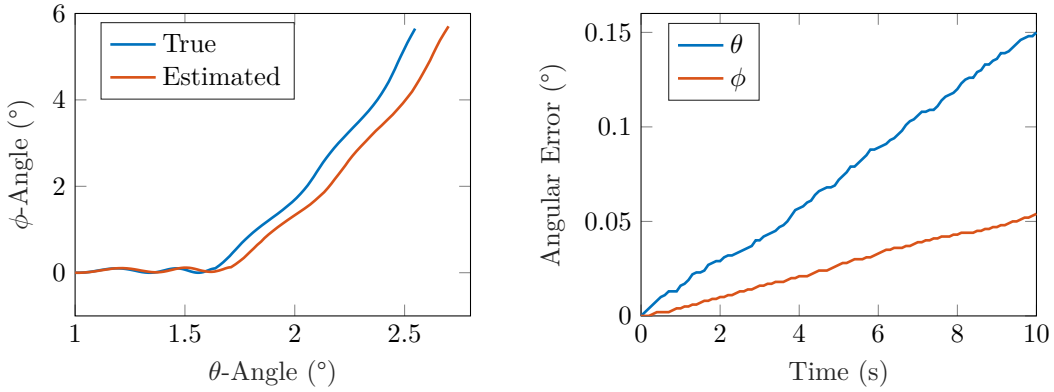
One very clear shortcoming shown in all the identified CubeSat star tracker technologies, is the maximum slew rate at which they can operate. This is not only a problem with low-power star trackers, however, as most star trackers do not show successful functioning during high slew rates. Although star trackers can offer excellent attitude accuracy for the nanosatellite industry, the sensor technology itself is limited by deficiencies such as high power usage and general system inoperability when within range of high-intensity celestial bodies and at high rates. Because of this, these sensors can rarely function as a sole attitude determination solution. To compensate for this, a common method of star tracker functional extension is augmenting a star tracker ADCS with gyros to enable inertial navigation.

### 2.5.3 Inertial Navigation

Inertial navigation is described as the determination of a vehicle's attitude or position through the process of dead reckoning. This technique uses the integration of angular rate and velocity data [25] in the estimation of the current vehicle position and orientation, relative to its last known position and orientation [26].

The key parts of an inertial navigation system are: the absolute position or orientation sensors, providing intermittent system state knowledge; and velocity sensors, providing a means of system state propagation.

Data originating from velocity measurement devices tend to show slight inaccuracies in the form of measurement noise and sensor biases. The use of these data in state propagation therefore leads to a runaway effect called drift caused by the accumulation of integrated errors. This process is illustrated in Figure 2.8, where a spacecraft's orientation is determined by use of noisy angular velocity measurements. These data are shown in contrast to the true system orientation, illustrating the drift as a function of the sensor noise power.




**(a)** True, and Dead-Reckoned Position

**(b)** Dead-Reckoned Position Error

**Figure 2.8:** Effect of Noise on Dead-Reckoning Systems

The mitigation of this drift is directly impacted by the velocity sensing accuracy. Although an increase in accuracy shows better ADCS performance, increasing sensitivity leads to an increase in system mass and power requirements. In the case of orientation propagation, the sensing field has not yet delivered commercially available, highly accurate, low-power rate-sensing solutions. As large inertial sensors can therefore not be justified for a marginal increase in ADCS performance capabilities, and drift can only be mitigated this way, more expensive sensors are not seen to be a complete solution. The only way of successfully compensating for this system drift, therefore involves absolute, bias-free orientation measurements to reset intermittently the accumulated errors, subsequently leading to an overall increase in estimation accuracy.

The compensation for this type of sensor deficiency in heading and orientation systems has been a subject of frequent research [27], [28], with the Kalman filter regularly applied as a method of optimal measurement fusion.

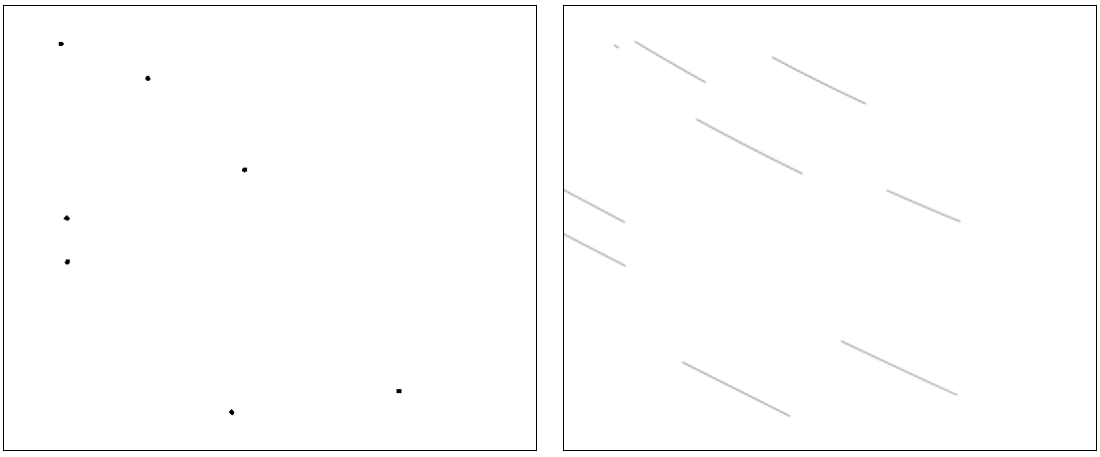
Although the fusion of sensor measurements leads to a much higher accuracy, the overall performance is still linked to the deficiencies of the most accurate sensor used in the fusion scheme, and once again leads to an increase in ADCS complexity.

## 2.6 Integrated Star Tracker-Based ADCSs

These systems involve either the estimation of rate through star observations, or the integration of gyros and star trackers into discrete units.

### 2.6.1 Gyroless Platforms

One of the proposed solutions to the multisensor ADCS complexity problem has involved the use of star tracker systems in the performance of multisensor functions. In the case of work done by Liebe et al. [29], a system was proposed to replace the traditional usage of discrete star trackers and rate gyro combinations. This proposed system planned to reduce the typical ADCS weight and power usage, as well as to simplify the required ADCS integration effort. Liebe et al. [29] proposed a single star tracker for usage during both high and low body rates. Owing to the differences in the observed features during these two rate conditions, a double algorithm was proposed. This algorithm is split into cases where the attitude can be determined, and cases where the slew rate experienced by the spacecraft is high, and attitude cannot be determined. These conditions are shown in Figure 2.9.



**(a)** Simulated Image During Low Slew Rates **(b)** Simulated Image During High Slew Rates

**Figure 2.9:** Inverted Simulated Star Images During Low and High Slew Rates

During low slew rates, Liebe et al. [29] propose that a high-performance APS sensor be used at a high update frequency by selectively reading out regions of interest around estimated star locations. In this mode, as distinct stars are assumed to be visible, enough star information will be available to determine unique attitude solutions. The system will therefore be able to function in an attitude determination mode.

During high slew rates, however, Liebe et al. [29] noted that the relative star magnitude will be low and constellations in the FOV will be indistinguishable due to their streaky nature. In this case, the FOV will only show one or two bright streaks corresponding to the path of the brightest of stars across the view cone of the imager. Here, Liebe et al. [29] suggest that the stellar gyro system be used for spacecraft de-tumbling, and claim that this system will be able to function at a slew rate up to 420 °/s. During this mode the star sensor integration period will be increased to ensure star streak visibility. Although this leads to a slower sensor update rate, it also causes an increase in the signal-to-noise ratio, and allows rate determination through means of a circle fitting method, to within an accuracy of 5 %. This does, however, cause a discrepancy with regards to update rate, during system state transition from low to high slew rates. The host ADCS would then have to be able to transition flawlessly between both modes, subsequently complicating the overall ADCS integration.

The work of Liebe et al. [29] work does not take into account the performance of a low-cost, low-power APS imager, however, as a sensor read-out scheme in the order of 198 Hz is assumed. No proof of functional system design nor testing is given. Although this might be plausible for implementation on a very high-performance, high-power APS chip, when compared to CubeSat star tracker trends, Liebe and coworkers' technology is not yet feasible for implementation on a nanosatellite platform.

The work by Liebe et al. [29] by no mean constitutes the full extent of stellar gyro research, however, as other researchers have also conducted experiments regarding the extension of star tracker-based systems with stellar gyro technology. Crassidis et al. [30] developed a Least Squares (LSQ) method of determining angular velocity by use of vector measurements obtained directly from a star tracker. This research has been the cornerstone of previous work done by Calitz [31], with practical test data achieved on the CubeStar [32] platform. Although this work did not undergo functional night sky testing, proof-of-concept results were obtained in an emulated environment. These results showed $3\sigma$ rate estimation accuracy in the order of $0.0379\,^{\circ}/s$ around the roll-axis and $0.0258\,^{\circ}/s$ around the cross-axis, for input angular rates of $1\,^{\circ}/s$.

Alternative methods for star tracker-based rate estimation have also been developed, with Singla et al. [33] demonstrating angular rate estimation by means of star tracker data, for the explicit case of gyro failure. Singla et al. [33] compared two separate approaches: firstly, spacecraft attitude and Kalman filtering were used by modelling external disturbance torques as a random walk process; and secondly, as with work done by Crassidis [30], two finite difference approaches were used in conjunction with a LSQ formulation. The main conclusions from this research were that the attitude-independent approach of the LSQ method showed a better response as it was decoupled from attitude estimation errors. Singla et al. [33] further concluded that a second-order LSQ approach showed better functioning at higher angular rates, whilst a first-order LSQ approach showed good results during low angular rates.

Further investigation into the subject has also been done recently by Pal et al. [34] and Fasano et al. [35] who, by use of optic flow techniques, enabled star sensor rate estimation without star identification. In the case of the work done by Pal et al. [34], a Kalman filter was used to decrease the overall estimation noise; however, no physical results were obtained. In contrast, the optical flow technique employed by Fasano et al. [35] showed proof of concept, despite potentially inaccurate operation. It was noted that their method for estimating satellite rates of $5\,^{\circ}/s$ showed increased errors owing to the lower signal-to-noise ratio of the observed images. Despite this, gyroless rate estimation functionality was demonstrated.

The largest part of the stellar gyro research has been relevant only towards high-performance applications, with much less focus on that of the low-power CubeSat star trackers, using lower dynamic-range APS sensors. Although all of these methods provide some measure of rate output, they are still limited to the update rate, dynamic range, and integration period of the APS sensor, which in the case of some low-power sensor systems, is a severely limiting factor. Therefore, although these systems can provide some measure of functional redundancy in terms of rate measurement, they offer little with regards to increasing sensor output bandwidth or extending functionality past the $2\,^{\circ}/s$ rate limitation. Although various well-researched methods for gyroless rate estimation does therefore exist, the use of these methods has distinct problems with regards to update rate, power usage and feasibility on a nanosatellite platform. Dedicated gyroless platforms therefore still require some improvement before they are fully implementable.

### 2.6.2 Gyro-Based System Integration

In stark contrast to the gyroless implementation of using only a star tracker-based system, more traditional sensor integration methods have also been seen in the development of a stellar gyro system. In these cases, gyros do not function as discrete sensing units, however, as they are closely integrated into a single attitude determination package. One specific system is the Inertial Stellar Compass (ISC) designed by Brady et al. [36]. The basic system submodules, as developed by Draper Laboratories, are shown in Figure 2.10.

As shown in this figure, the ISC, consisted of a closely coupled gyro–imager assembly, with measurements fused by use of a 27-state square root Kalman filter. Not only were proof-of-concept ground validation results achieved for this sensor, but the system also underwent actual orbital operation validation.

During the TacSat-2 [38] mission, the ISC demonstrated integrated star tracker and gyro functioning as an individual product. The ISC achieved $1\sigma$ accuracy in the order of $0.1°$, reaching specification. As identified by Brady et al. [36] some key features with regards to low-level sensor measurement integration are:

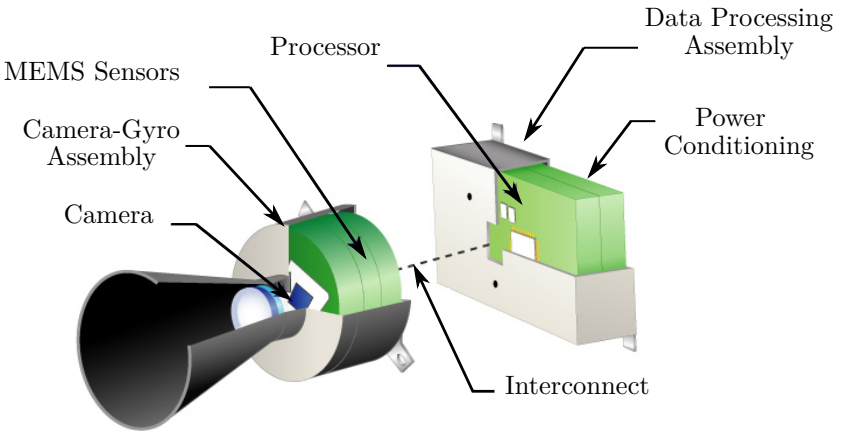



**Figure 2.10:** Inertial Stellar Compass, Adapted from [37]

- Attitude updates are available even when the star camera measurements are inconclusive due to high rates;
- System functionality is retained, even when the camera FOV is obstructed;
- The system usage is much broader than just fine pointing and attitude determination as the rate gyro data can be used during initial satellite detumbling; and
- Overall system ADCS integration is much easier as there is no necessity for external gyro error compensation.

Another more commercial application of a system similar to that designed by Brady et al. [36], is the ASTROgyro [39], developed by Jena Optronik. This system consists of two star trackers and two gyroscopes, providing an integrated, redundant system with $1\sigma$ random errors quoted in the arcsecond range. As this technology has only very recently started to become available on larger satellites, no integrated solution for a small satellite exists yet. The ASTROgyro system is shown in Figure 2.11. This system has a mass of about 6 kg and power consumption in the order of 21 W to 54 W.

**Figure 2.11:** Jena Optronik ASTROgyro [40]

One attempt at a low-power stellar gyro-assisted system for nanosatellites is that of work done by Rawashdeh et al. [41], [42]. In their work, an inexpensive stellar gyro system measuring only body rates was developed to reset periodically gyro bias drift. This work showed that drift compensation every fifteen seconds led to attitude knowledge better than 1° of accuracy. The

developed system did not consist of a complete integrated solution, but demonstrated the usage of star sensing optical systems for the compensation of rate gyros deficiencies.

Although most work mentioned so far was only concerned with development of a high update rate, high-functioning stellar system, other authors have also researched the use of rate gyro data in the augmentation of star tracker algorithms. An example of this is the development of a hybrid star tracking algorithm by use of sensor fusion as done by Lu et al. [43]. In their work, they attempted to address star tracking at high slew rates, as it was identified that tracking accuracy is dependent on the implementation of the star tracker tracking mode. To solve this problem, they developed an innovative tracking mode based on gyro measurements. Further research in the integration of gyros and star trackers was done by Sun et al. [44], who developed a method of star spot reconstruction during high-exposure conditions, based on gyro measurements. During their work, Sun et al. [44] also provided an innovative method of determining gyro bias drift, subsequently compensating for both sensor deficiencies at a very low level.

## 2.7 Chapter Outcomes

In this chapter, an introduction to the most important background information regarding star tracker systems was given. Examples of integrated ADCSs are used to highlight the need for integrated and autonomous sensor solutions with a focus on star tracker systems. Subsequent chapters in this thesis will utilise concepts of astronomy, reference frames, and attitude representations treated in this chapter. These concepts were given as background to star tracker-based attitude determination and will be used in the development of suitable software and algorithms. A brief overview of star trackers and inertial navigation systems presented in this chapter was given along with currently available nanosatellite star tracker systems to determine suitable star tracker specifications. This information is most important when considering the hardware design of an integrated star tracker platform, as is considered in the next chapter.

# Chapter 3

# Hardware Design

The following chapter deals with the design of the star tracker gyro assembly used throughout this work, in the fulfilment of the first objective. This chapter starts with an in-depth description of the CubeStar nano star tracker. Thereafter, the proposed hardware changes are described including notable design decisions made in this work. Finally, this chapter concludes with the system hardware integration.

## 3.1 The CubeStar Platform

CubeStar, as designed by Erlank [45], is a nano star tracker developed with a modular design approach. The first hardware iteration consisted of three Printed Circuit Boards (PCBs) interconnected through means of 40-pin headers in a stacked fashion. Since the completion of the first hardware revision, development has been taken over by CubeSpace, a local small satellite ADCS manufacturer. Although CubeStar has gone through many developmental phases, the overall system structure has stayed roughly the same, with major changes only affecting the optical sensor and image data flow. For any more information on the original CubeStar project, the reader is referred to Erlank [32]. CubeStar revision 4.2 as of February 2017 is shown in Figure 3.1.

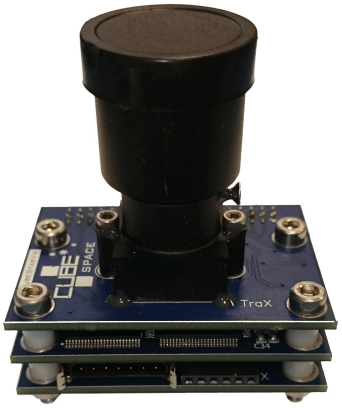



**Figure 3.1:** CubeStar Revision 4.2

Despite the hardware consisting of a variety of electronic components, the overall structure can be divided into two main subsystems: that of i) the optics, and ii) the data-handling and processing system.

### 3.1.1 Optical Hardware

In the case of the optical hardware, an image sensor-lens assembly is used to enable star detection. The assembly was designed to ensure visibility of at least three stars across the entire celestial sphere, enabling full three-axis attitude determination with 99.9% sky coverage. In this application,

the optical exposure time is limited such that only stars with visible magnitudes up to 3.8 Mv can be observed allowing for a sensor update rate of at least 1 Hz. The CubeStar sensing and optics subsystem is shown in Figure 3.2

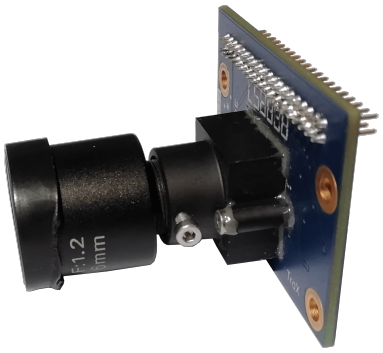



**Figure 3.2:** Optical PCB of CubeStar Version 4.2

The specifications of the image sensor and lens follow.

#### Image Sensor

The image sensor used in CubeStar is the EV76C560 developed by Teledyne e2v [46]. This image sensor features 1.3 million pixels in a $1280 \times 1024$ array. The pixels used in this sensor have dimensions of $5.3 \times 5.3\,\mu m$ and feature microlens technology to enable improved optical efficiency. Pixel data width is selectable between 8 and 10 bits. To decrease overall image size, however, a data width of 8 bits is used. This sensor further features a selectable shutter type to provide improved flexibility. In the case of CubeStar, the global shutter mode is used to limit the effect of slew-induced distortion.

Image download is enabled through a parallel interface, with an added Serial Peripheral Interface (SPI) bus for command and telemetry transfer. Sensor control is further enabled through the trigger and reset pins. This image sensor provides a relatively high sensitivity at low light, as well as a power consumption of only 200 mW, making it an ideal candidate for a low-power star tracker.

#### Lens

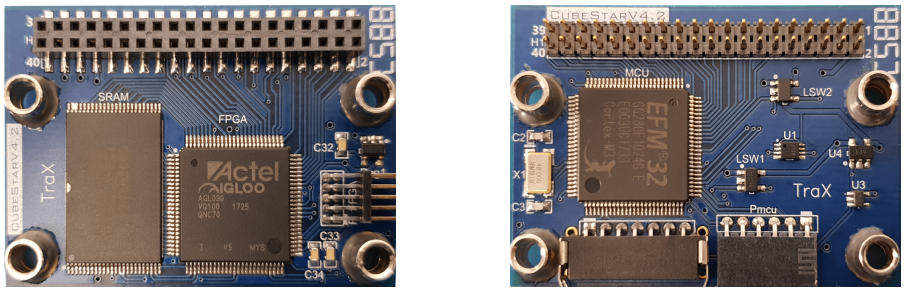
Although the image sensor shows above average sensitivity, exposure time is limited to 500 ms to allow enough time for image processing and attitude determination. A wider FOV is therefore required to ensure full sky coverage. The FOV is determined by a Lensation BL6012 S-mount lens. This lens has a horizontal FOV specified as 44.5° with an aperture of F1.2 [47]. With a total weight of only 20 g, it is ideal for usage in a light, small form factor application.

### 3.1.2 Data Handling and Control

To enable image processing and housekeeping functionality, the star tracker assembly further consists of a data handling and control subsystem as shown in Figure 3.3. This subsystem consists of two PCBs containing an MCU, Field Programmable Gate Array (FPGA), and Static Random Access Memory (SRAM), as well as supporting circuitry.

#### Microcontroller

Considering the sizeable software requirements of a star tracker, CubeStar was originally designed with a 32-bit ARM Cortex-M3 MCU operating at 48 MHz. The MCU chosen was the EFM32GG280F1024 in LQFP100 package as developed by Energy Micro. This processor was originally used owing to its Electronic Systems Laboratory (ESL) heritage dating back to the design of CubeComputer, a nanosatellite OBC. It thereby also enabled faster low-level hardware and software design decreasing the overall project timeline. The EFM32 processor features 1024 kB of flash




**(a)** External Memory and Control Board **(b)** Microcontroller and Interface Board

**Figure 3.3:** The CubeStar Data Handling and Control Subsystem

and 128 kB of SRAM. This MCU further includes a rich set of communication peripherals such as Universal Synchronous/Asynchronous Receiver Transmitter (USART), SPI, and Inter-Integrated Circuit (I2C) as well as a 12-bit Analog-to-Digital Converter (ADC) peripheral for accurate analog sampling [48].

#### Field Programmable Gate Array

CubeStar further consists of an FPGA enabling high-speed sensor-to-memory image transfer. The FPGA used in this application is the Microsemi Igloo Nano AGLN030-VQ100. This component was initially chosen for the CubeStar application as it has a long heritage in the ESL, thereby simplifying system design. In this application, the FPGA is used to pipe data efficiently from the image sensor to external SRAM and act as a memory controller by handling SRAM access.

#### External Memory

As captured images are too large for internal SRAM, additional external memory is added. This SRAM chip consists of 2 MB of low-power, asynchronous SRAM in an easy-to-solder TSop package. The SRAM is specified as having an access time of 55 ns allowing a maximum image transfer rate of 18 MHz.

### 3.1.3 Data Flow and Operation

The subsystems as described in Sections 3.1.1 and 3.1.2 are interconnected as shown in Figure 3.4.

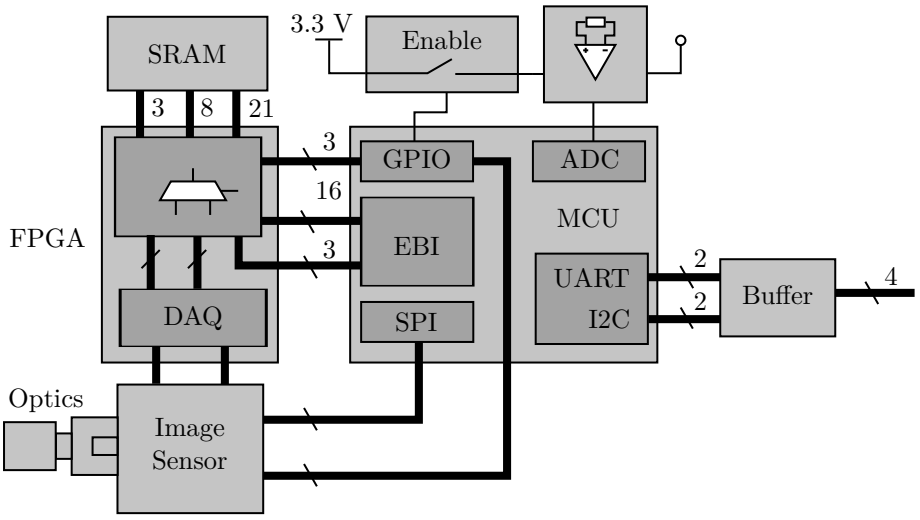



**Figure 3.4:** CubeStar Hardware Interconnection

Image sensor operation is controlled by the MCU through the SPI-bus. The process of image capture works as follows. Firstly, image capture is triggered by a rising edge on the trigger pin. After image integration is completed, the MCU is notified. SRAM access is then handed over to the image sensor to allow high-speed image transfer. Once this download is completed, the FPGA notifies the MCU and memory control is transferred back to the MCU so that processing can commence.

### 3.1.4 Additional CubeStar Specifications

Additional CubeStar specifications are given in Table 3.1.

**Table 3.1:** CubeStar Specifications

| | **Specification** | **CubeStar** |
|---|---|---|
| **Accuracy** | $(3\sigma)$ Cross-axis | 0.0215° |
| | $(3\sigma)$ Round-axis | 0.061° |
| | Catalogue Size | 410 |
| | Sensitivity | Up to 3.8 Mv |
| **Physical** | Mass | 55 g |
| | Size | $50 \times 35 \times 55$ mm |
| **Power Supply** | Supply voltage | 3.3 V |
| | Average Power Usage | 142 mW |
| | Peak Power Usage | 264 mW |

## 3.2 Hardware Design Requirements and Initial Design Decisions

Similar to the original design of CubeStar, the decision was made to reuse as much of the previously generated hardware as possible. The proposed hardware would therefore be expected to consist of:

- An image sensor with peripheral circuitry, lens, and lens mount;
- A FPGA and SRAM with peripheral circuitry;
- A microcontroller, power management, and external interface headers; and
- An angular rate-measurement device.

At the earliest design stage it was decided to keep the stacked, modular interface so that existing subsystems could be reused. This changed the problem of system design to that of system augmentation, subsequently decreasing the expected hardware design cycle timeline. If the choice of rate-measurement device would permit it, a three board design would be opted for, allowing for a smaller total system volume.

Owing to the augmentative nature of this project, it was decided that the FPGA–imager interface of CubeStar V4.2 would feature as the centre point of the new hardware design.

## 3.3 Microcontroller Replacement

From the CubeStar Interface Control Document [49], it was noted that the EFM32GG chip offered a maximum of 20 ms idle processing time during each 1 s cycle. Owing to the necessity of more computationally intensive estimation techniques this was deemed insufficient. An MCU with increased performance was therefore required.

With current technological trends and the subsequent increase of processing power in low-power embedded chips, a wide variety of new ARM MCUs has become available for use in low-power applications. Many different options for replacing the CubeStar Cortex-M3 therefore exist.

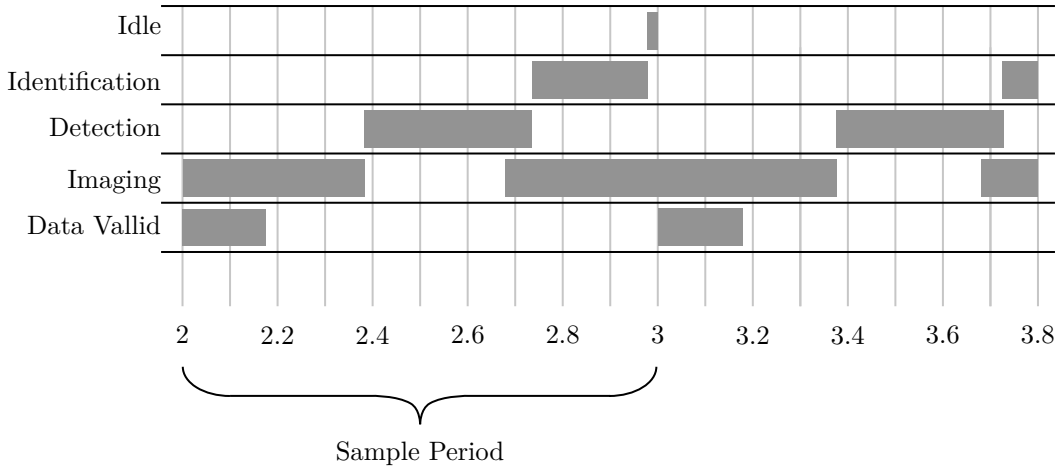



**Figure 3.5:** CubeStar V4.2 Timing

Three possible candidate architectures of the ARM Cortex-M family were subsequently identified, namely the M3, M4, and M7 series. In this family, the M3 and M4 series showed very similar power usage and performance characteristics whilst the M7 series had a much higher computational ability. Despite the advantages seen in Cortex-M7 processing capability, these MCUs showed high power usage and were therefore not investigated as a suitable replacement [50]. A brief description of the M3 and M4 MCUs follows.

### 3.3.1 ARM Cortex-M Microcontroller Description

**Cortex-M3:**

The ARM Cortex-M3 has been described as an industry-leading 32-bit processor. This processor family was specifically developed for high-performance, low-cost platforms. The Cortex-M3 MCUs use the ARMv7-M Harvard architecture with a three-stage pipeline and complete Thumb / Thumb-2 instruction set. This microcontroller family further offers hardware divide circuitry using 2-12 clock cycles as well as single cycle, 32-bit width multiplication operations, with operating frequencies as high as 120 MHz [51].

**Cortex-M4:**

In many instances the Cortex-M4 family can be seen as close to identical to the Cortex-M3 family. One of the key differences is, however, the availability of a Digital Signal Processing (DSP) instruction set, as well as an optional Floating Point Unit (FPU) for increased computational ability. The Cortex-M4 chips are further based on the newer ARM7E-M Harvard architecture, supporting the same Thumb /Thumb-2 instruction sets [52].

Overall, owing to the family similarities, controllers from either series could be good replacements, as long as a chip is chosen with a higher clock frequency than the currently used 48 MHz. Ideally, however, because of the optional FPU, preference would be given to MCUs from the Cortex-M4 series.

### 3.3.2 Design Considerations

Following identification of replacement MCUs, three possible design routes were identified. Firstly, the current hardware design could have stayed exactly the same, in the hope that enough code optimization could be implemented to ensure real-time functioning. Although this would have been the easiest from a platform design and integration point of view, it offered little leeway in terms of more creative processing techniques. It was decided that this option would only be a last resort.

Secondly, a different chip from the same Silicon Labs family could have been used. This would have been the most optimal solution, owing to the documented code migration techniques silicon

vendors tend to offer. Many vendors also promise pin-to-pin microcontroller compatibility, enabling drop-in replacement. If a suitable controller from this range could have been found, it would offer the greatest boost in execution efficiency with the least amount of design effort. In this case, software redesign could have been assumed minimal as most of the low level firmware interfaces should be transferable.

Lastly, a chip from the same family, but different silicon vendor could have been used. Owing to the relatively standard design guidelines given by ARM, it can be assumed that most chips based on the M3/M4 architecture have strong similarities in functionality. Redesigning the current CubeStar hardware to fit silicon from a different vendor should have therefore, in theory at least, not be much more effort than choosing a chip from the same vendor. Following this route does, however, lead to much more low-level software redesign, as there is no guarantee that hardware abstraction layers are available cross platform.

### 3.3.3 Microcontroller Comparison

To help with the MCU design and compare the MCU specifications, the following rubric was constructed. It was decided that the MCU should at least be comparable to the original Giant Gecko MCU, not only in hardware functionality, but also in support documentation and development tools. The most crucial hardware requirements can therefore be summarised as:

- Availability of design documentation;
- Low power requirements;
- Availability of an External Memory Interface (EMI);
- Internal flash and SRAM of at least 1024 kB and 128 kB, respectively;
- Easy-to-solder packages, preferably LQFP or QFPN; and
- A clock speed of no less than 48 MHz.

At the time of hardware design, three additional chips were identified as possible replacements. These MCUs are briefly compared in Table 3.2 with the Giant Gecko given as benchmark.

**Table 3.2:** Microcontroller Comparison

| **Chip** | **Power** (µA/MHz) | **EMI** | **Package** | **Freq** (MHz) | **FLASH** (kB) | **SRAM** (kB) |
|---|---|---|---|---|---|---|
| EFM32GG [48] | 219 | EBI | LQFP100 | 48 | 1024 | 128 |
| NXP K24 [53] | 250 | Flexi Bus | QFP100 | 120 | 1024 | 256 |
| STM32L476 [54] | 100 | FMC | LQFP100 | 80 | 1024 | 128 |
| ATSAMG54 [55] | 102 | NONE | LQFP100 | 120 | 512 | 96 |
| EFM32PG [56] | 64 | NONE | BGA125 | 40 | 1024 | 256 |

During the component identification, the EFM32 microcontroller range offered a sparse selection of Cortex-M4 MCUs, offering only the Pearl Gecko range. Despite the excellent low-power capabilities of this chip, it offered no external memory peripheral and a maximum clock frequency of only 40 MHz [56]. This was deemed as insufficient, as external memory was a crucial prerequisite and the FPU optimisation would not compensate for a reduced clock speed in an already burdened application.

The ATSAMG54 range by Atmel was also identified as a potential replacement. However, owing to the relatively small amount of SRAM and flash memory, as well as the lack of an EMI, it did not fulfil the requirements as stipulated at the beginning of this subsection [55].

The only two viable options were therefore the NXP K24 and STM32L4 chips. Both offered similar peripheral options, with the NXP K24 going one step further with the higher clock frequency and larger on-board SRAM. The NXP K24 did, however, offer worse power performance and higher development tool price [53], [54].

### 3.3.4 Microcontroller Design Choice

Owing to the superior power performance, rich peripheral options, and inexpensive development support it was decided to opt for the STM32L476. With development boards costing as little as USD 26 and external target programming support, it further offered the best features to price ratio.

Not only did it offer good hardware development support, but also excellent software support as well as an active community forum. The STM32 development environments were also numerous, free, and available for both Linux and Windows.

## 3.4 Inclusion of Rate-Measurement Device

As this project also relies on measuring sensor rates, a rate-measurement device is also required. A variety of rate sensing sensor technologies are available, with sensors based on the Sagnac and Coriolis effect commonly used in the astronautical field [57]–[60].

Some of the key contributions to rate measurement-technology are the mechanical gyro, the optical gyro, and the MEMS gyro [61], [62]. Although progress has been made in this measurement field, only certain types of implementations are relevant to the nanosatellite industry owing to the strenuous constraints on power and size requirements enforced thereon [63]. Subsequently, because mechanical gyros are expensive, large, and power intensive, they lose relevance.

The alternative angular rate-sensing methods, that is the fibre optic as well as MEMS gyros, are deemed to be much more suited for small satellite application and are investigated further.

### 3.4.1 Current State of Technology

#### Fibre-Optic Gyroscopes

FOGs use the Sagnac effect [64] to sense the rotational rate of a body. Similar to other optical gyro technologies, the measurement depends on the phase shift observed in counterpropagating light sources [65]. The basic structure of a FOG is shown in Figure 3.6.

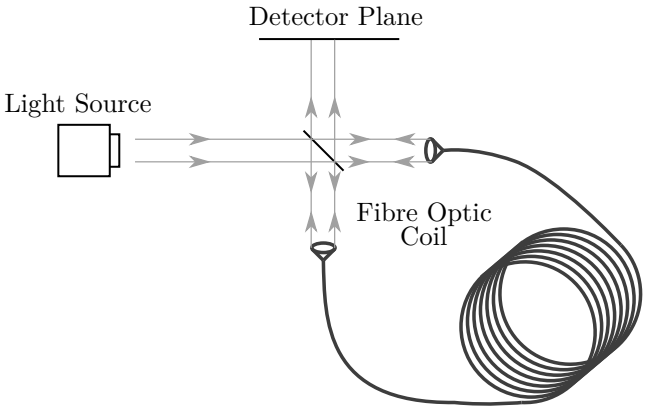



**Figure 3.6:** Structure of a FOG

Generally a FOG consists of a i) laser source, ii) single mode fibre, and iii) detector, combined to form a Sagnac interferometer **Gialorenzi1982a**. The biggest factor in FOG technology accuracy is the optical fibre length. In some cases a FOG can consist of a few kilometres of optical fibre, depending on the specific attenuation, laser intensity, and accuracy required [66].

#### MEMS Gyroscopes

An alternative to FOG technology is that of the MEMS gyros and IMUs. These sensors are based on a vibrating reference motion enabling conservation of momentum as seen in the Foucault pendulum problem [67].

When an external rotation is applied to a MEMS gyro, a Coriolis force with frequency equalling that of the reference motion is generated [68]. This force can be measured, leading to an output gyro rate. The current group of MEMS sensors consists of low-performance devices designed as electronically driven resonators usually made from quartz or silicon. The structure of a MEMS gyro is shown in Figure 3.7.

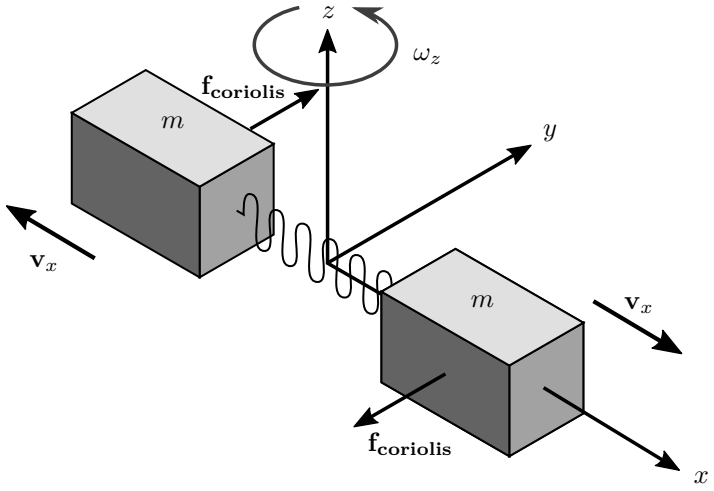



**Figure 3.7:** Structure of MEMS Gyro

### Technology Comparison

A comparison of the rate-measurement technology specifications, as done by KVH Industries [69], is given in Table 3.3.

**Table 3.3:** Gyro Technology Comparison [69]

| Parameter | FOG | MEMS |
|---|---|---|
| **Gyros** | | |
| Bias Instability (°/h) | $\leq 0.05$ | $\leq 1$ |
| Angular Random Walk (°/$\sqrt{\text{h}}$) | $\leq 0.7$ | $\leq 9$ |
| Dynamic Range (°/s) | $\pm 490$ | $\pm 400$ |
| **Electrical** | | |
| Power Consumption (W) | $\leq 8$ | $\leq 2$ |
| **Physical** | | |
| Mass (kg) | $\leq 0.7$ | $\leq 0.05$ |
| Dimensions ( mm) | 88.9 Dia × 73.7 h | 45.7 × 38.1 × 20.32 |

In comparison, FOG technology is at least an order of magnitude more accurate than that of MEMS. Despite this, MEMS gyros tend to have a significantly smaller form factor and lower power usage than standard FOG technology. Therefore using MEMS gyros would lead to much more efficient use of available satellite resources, making the application thereof much more attractive. Commercial MEMS sensors are also much easier to procure and offer a more reasonable price.

Taking into account all of these factors, it was decided to use a MEMS gyro.

## 3.4.2 Design Considerations

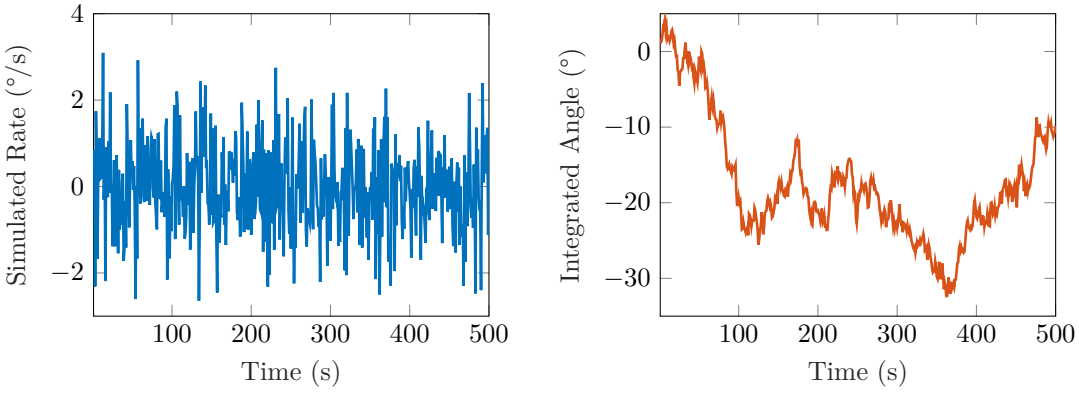
All gyros have a variety of specifications that govern the performance metrics. Most important are Angle Random Walk (ARW) and bias instability. What follows is a description of the most important sensor metrics to be taken into account in the design process.

### Angle Random Walk

In practical ADCSs, rate sensor data are regularly used to determine relative angular change by means of numerical integration, specifically when direct attitude measurements are not available. In most rate-measurement devices, measured data show high-frequency angular rate variations independent of the true rate experienced by the system. These variations can be attributed to normally distributed sensor noise.

During the numerical integration of this noisy data, random changes, similar to that shown in Figure 3.8, will be observed. This phenomenon is called ARW and it is generally measured in $°/\sqrt{\text{h}}$.

Sensors with low noise measurements will therefore exhibit less ARW leading to decreased uncertainty during attitude propagation. Less frequent correction updates will therefore be necessary to achieve certain ADCS accuracy margins.




**(a)** Simulated Noisy Rate Measurement

**(b)** Integrated Rate Data

**Figure 3.8:** Angle Random Walk and the Effect of Sensor Noise

### Bias Instability

Sensor bias is defined as a nonzero offset in measured output data. In rate-measurement devices such as gyros, this bias is not necessarily a constant offset but can change dramatically over the course of system operations. The magnitude of this bias change at a constant temperature is known as a gyros bias instability and it is generally measured in °/h for high-performance sensors.

According to Kirkko et al. [70], bias instability is a nonstationary error process with a one over frequency power density and can be seen as a source of flicker noise. This noise source dominates at low frequencies due to the slow fluctuations observed. Low bias instability is crucial for application where autonomous navigation or long term accuracy is required [71].

### Power Usage and Volume

The final factor that is of importance during the consideration of an IMU for a nanosatellite application, is the physical and electrical specifications of the sensor; in this case: size, mass, required voltage, and power requirement.

Due to the nature of the application, the size and mass of the total system should be as little as possible, with the maximum corresponding to the size of 1 U. This is the absolute maximum, however, whereas the total system volume should ideally be as small as possible.

With regards to the electrical specifications, a required IMU bus voltage of 3.3 V would be preferable as no additional regulation would be required. In terms of total power usage, the designed system is required to use as little as possible.

### 3.4.3 IMU Comparison

In the subsequent search for suitable IMUs, the sensors' bias instability, ARW, and form factor were therefore among the top priorities. Subsequently, three possible IMUs were identified as plausible components for the integrated star sensor: the Epson-G364, Sensonor STIM202, and Analog Devices ADIS16460. These sensors are shown in Figures 3.9a to 3.9c.


**(a)** Epson G364 [72] **(b)** Sensonor STIM202 [73] **(c)** AD ADIS16460 [74]

**Figure 3.9:** Potential Inertial Measurement Sensors

Unfortunately no small-package IMU with good specifications could be found that would easily fit on the floorspace available on a three-board design. A list of the most notable sensor specifications is given in Table 3.4.

**Table 3.4:** Comparison of Investigated IMU Specifications

| | **ADIS16460** | **M-G364PD** | **STIM202** |
|---|---|---|---|
| **Physical Specifications** | | | |
| Voltage (V) | 3.3 | 3.3 | 5 |
| Current Requirement (mA) | 44 to 55 | 18 | 200 |
| Size (mm) | $22.4 \times 22.4 \times 9$ | $24 \times 24 \times 10$ | $44.75 \times 28.6 \times 10$ |
| Mass (g) | 15 | 10 | 55 |
| Operating Range (°C) | -25 to 85 | -40 to 85 | -40 to 85 |
| **Sensing Specifications ($1\sigma$)** | | | |
| Bias Instability (°/h) | 8 | 2.2 | 0.3 |
| Angular Random Walk ($°/\sqrt{h}$) | 0.12 | 0.09 | 0.2 |
| **Average Price** (USD) | 272.96 | 1043.12 | 5080.00 |

Despite the flight heritage of the STIM202 [75], it was found to be very large with a nominal power requirement of 1 W [76]. In contrast to its superior bias instability and high price, it also showed the worst ARW of all the sensors and was therefore not deemed feasible for use in this application.

Of all the investigated sensors, the Epson M-G364PD showed the best ratio of physical size to sensor capability, with a current usage of only 18 mA and accuracy specifications slightly better than that of the ADIS16460 [77], [78]. Although the ADIS16460 had worse specifications across the board, the M-G364PD showed to be much more difficult to procure.

### 3.4.4 IMU Design Choice

Owing to the local availability, low price, and ease of procurement, it was therefore decided to use the ADIS16460. The ADIS16460 is a six-Degrees of Freedom (DOF) inertial sensor consisting of a three-axis gyroscope and accelerometer. The sensor is by far the most inexpensive in its functionality class with a price of only USD 272.96. This sensor also has widespread availability with a lead time of only three weeks from Mouser Electronics. Specification-wise, the ADIS16460 offers

a relatively small, low-power rate-measurement solution with a simple communication interface implemented by use of a full-duplex SPI bus. The physical interface of the sensor offers external trigger support for direct sample control together with data-ready functionality to allow readout synchronisation.

A wide variety of software options is also offered with easy-to-use Bartlett window FIR and decimation filtering to allow for an overall better noise response. A more complete list of sensor specifications is available in Table 3.5.

**Table 3.5:** Additional ADIS16460 Specifications

| | **Typical** |
|---|---|
| **Gyros** | |
| Bias Instability (°/h ) | 8 |
| Bias Temperature Coefficient (°/s/°C) | ± 0.007 |
| Effect of Linear Acceleration (°/s/g) | ± 0.01 |
| Angular Random Walk (°/$\sqrt{\text{h}}$) | 0.12 |
| Dynamic Range (°/s) | ± 100 |
| **Power Supply** | |
| Voltage (V ) | 3.3 |
| Current Requirement (mA) | 44 to 55 |

As the sensor is only rated as industrial grade, a risk does exist when using it in space application as there is no proof of the IMUs radiation hardness. The sensor further does not have a wide temperature operating band although it can be compensated for by stricter thermal control.

## 3.5 System Integration and Data Flow

Following component selection, a hardware layout was completed which consists of four stacked PCBs. An artistic representation of this hardware is shown in Figure 3.10.

The overall structure was kept very similar to CubeStar V4.2 with two of the original boards having been kept identical.

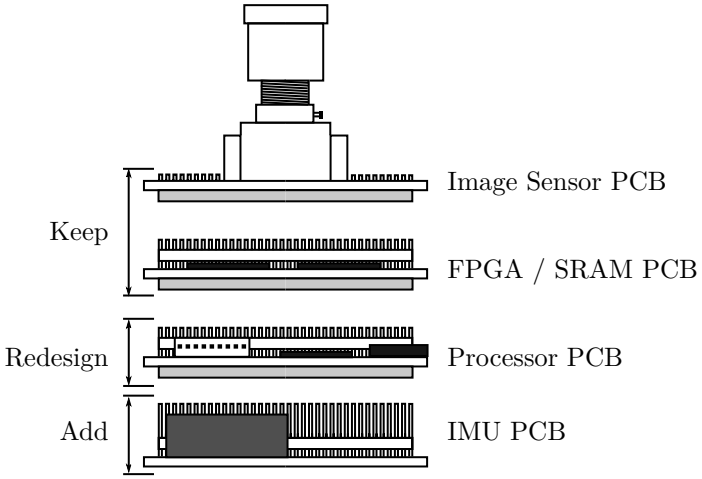



**Figure 3.10:** CubeStar Artist Representation

### 3.5.1 Designed Data Flow

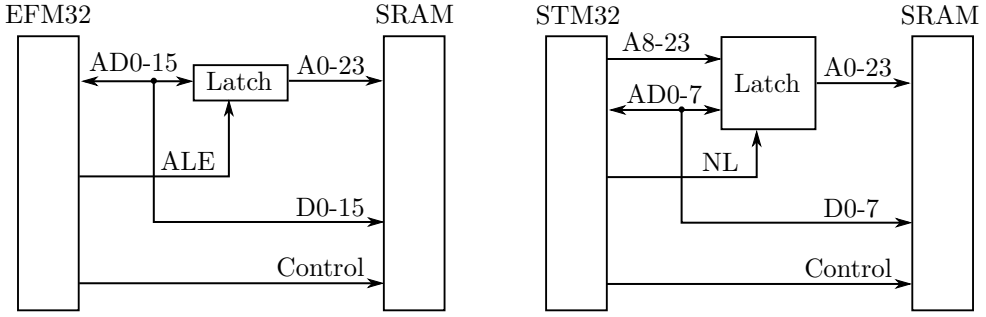
The designed data flow between the microcontroller, FPGA, and image sensor stayed practically identical to that of V4.2. This allowed for simple subsystem replacement in the case of catastrophic failure, as well as decreased debugging effort. The only addition to this hardware was added support for an IMU, as well as an extension of the main header for Controller Area Network (CAN) support, along with the already-used I2C and Universal Asynchronous Receiver Transmitter (UART).

In this case, the IMU was connected to the MCU by use of an SPI bus, external trigger, and data-ready pins. The IMU trigger pin was connected to a timer peripheral to enable automatic trigger pulse generation. Subsequently, the data-ready pin was then connected to an interrupt line to allow complete independent functioning of the IMU. The reasoning behind this was due to the time sensitivity of the IMU data. In conclusion of the data flow, the IMU and image sensor were connected to different SPI buses to ensure reduced interference and allow parallel communication, if necessary.

With initial hardware testing, most redesigned interfaces seemed to be working correctly. It was, however, noted that the MCU–SRAM interface was not functioning as was originally required; the main reason being the difference in operation of the Flexible Memory Controller (FMC) and the External Bus Interface (EBI) when applied to multiplexed asynchronous SRAM. A bus redesign was therefore required.

### 3.5.2 External Memory Bus Redesign

It was discovered that the current MCU–SRAM interface was not compatible with the new ARMv7 processor owing to fundamental differences in the implementation of the Silicon Labs EBI and STMicroelectronics FMC. This difference is shown in Figure 3.11




**(a)** EFM32 EBI SRAM Connection **(b)** STM32 FMC SRAM Connection

**Figure 3.11:** External Memory Controller Implementation Difference

In this case, the EBI supported 8-bit data read/write to a 24-bit address through an interface of only 16 bits. This process functioned by first putting the most significant sixteen address bits on the address lines; the ALE pin would then be pulled low to latch these address bits. In the next clock cycle, the address LSB would be multiplexed with the 8-bit data. Although the FMC supports multiplexed SRAM access, it does not support the same external latch functionality as implemented in the EFM chips.

The bus connecting the processor and FPGA therefore had to be modified to support multiplexed SRAM access with an address bus width of at least 21 bits. The resulting changes led to a hardware reconfiguration, as shown in Figure 3.11b. As these hardware changes were only implemented after the initial hardware layout had been completed, the changes had to be implemented in the form of wire-mods, secured by epoxy to ensure longevity. Minor changes were also brought on to the FPGA firmware to ensure correct functioning with the updated interface. The final CubeStar connection digram is hence shown in Figure 3.12

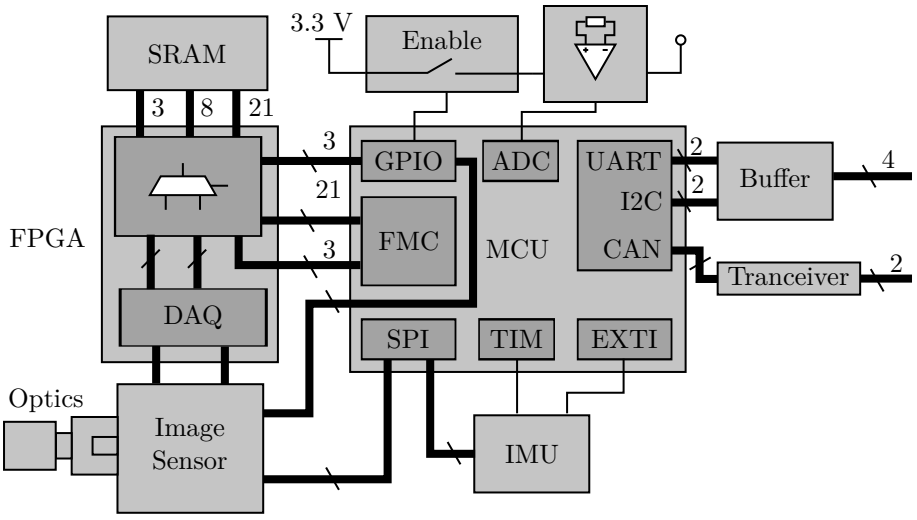



**Figure 3.12:** Final Hardware Interconnection Diagram

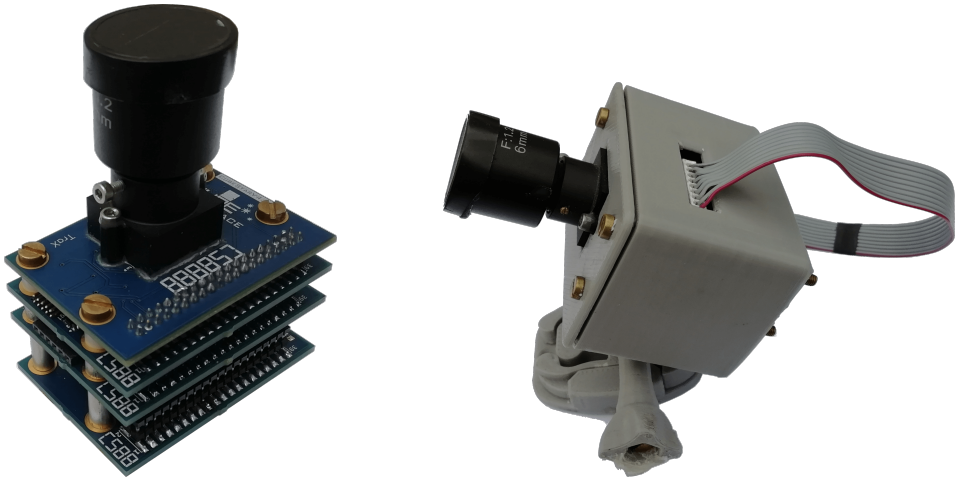
## 3.6 Chapter Summary

In this chapter, the most prominent design decisions made during the development of the augmented CubeStar hardware are discussed in fulfilment of the platform design objective. Of these design decisions, the most notable two are that of the MCU and IMU.

With regards to the MCU, the STM32L476 was chosen to function as MCU replacement, giving a much needed boost in computational efficiency. A rate-measurement device in the form of a MEMS IMU was then added, to augment further the CubeStar platform functionality.

Although an attempt was made to keep as much of the previous CubeStar design as possible, it was found that the two MCU families had irreconcilable differences in implementation, showing the necessity of a bus redesign. Only the CubeStar camera module was therefore used as is, with minor adjustments made to the FPGA–MCU interface to ensure successful image transfer from SRAM.

The final engineering model, as used in subsequent tests, is shown in Figure 3.13a. The hardware is encased in a 3D-printed case, as shown in Figure 3.13b.



**(a)** Hardware **(b)** Encased Hardware

**Figure 3.13:** Designed Hardware

# Chapter 4

# Software and Algorithms

As a suitable platform was developed with which measurements can be gathered, the second objective, that of the identification of suitable software techniques, is addressed in this chapter.

A star tracker system consists not only of a hardware assembly, but also depends on various algorithms to ensure i) correct functioning, ii) star pattern matching, and iii) accurate attitude determination. Although the system designed in this work does not function purely as a star tracker, but rather as an augmented stellar sensor for the accurate and robust detection of satellite attitude and rates, star tracker algorithms still apply. The overall software interconnect to achieve these tracking and detection functions is shown in Figure 4.1.

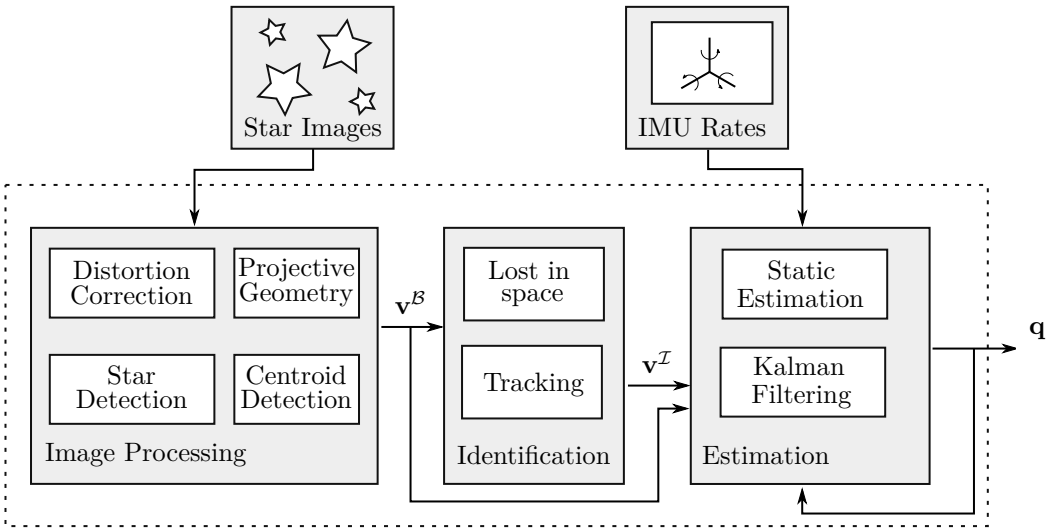



**Figure 4.1:** Software Overview

The figure shows the three main functional blocks constituting: i) image processing, ii) star identification, and iii) attitude estimation. This figure further shows the system dependence on two different sets of input data; namely star images, and IMU rates. In the case the IMU data, the interpretation thereof is simple, as raw measurements represent a linear scaling of interpretable data. Star images, however, require more processing in the conversion of raw sensor data to usable attitude measurements.

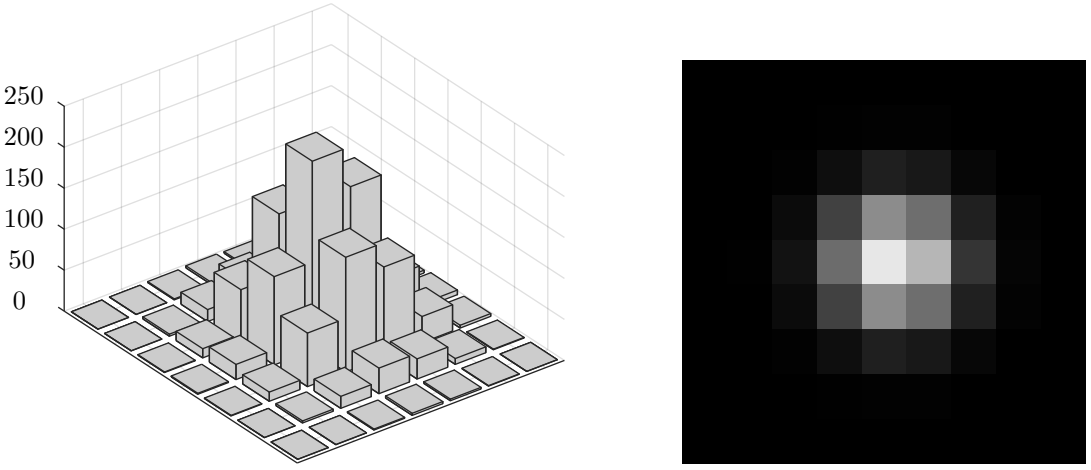
The following chapter gives a brief overview of some of the methods employed to generate an attitude from star images with attitude propagation enabled by high-speed IMU measurements. This chapter begins with a simplified star model used during this work, followed by image processing techniques as required by a second-generation fixed-head star tracker. A brief review is then given of star identification methods during LIS and tracking modes. Finally, some useful attitude and rate estimation techniques are presented. The overall chapter layout follows that of Figure 4.1.

## 4.1 Star Model

Optic defocusing and diffraction make stars, which are point light sources, appear not as discrete illuminated pixels, but as areas point spread in nature. Contrary to intuition and the Nyquist theorem, this enables much higher star detection accuracy than perfectly focused optics. In this case, the a priori knowledge of the star distribution pattern in an image is used to enable hyperacuity, or superresolution. Previous work [31], [32] has therefore found it sufficient to model an ideal star as a symmetric normal distribution that is a function of the radial distance from the star centroid, $r_c$, such that

$$I(r_c) = I(0)e^{\frac{-r_c^2}{2\sigma}} \tag{4.1.1}$$

In this equation, the peak star intensity is represented by $I(0)$, with $\sigma$ influencing the star spread width on the image plane. This star model is shown as a sampled two-dimensional distribution alongside its appearance in an image in Figure 4.2.




**(a)** Ideal Star Distribution **(b)** Ideal Star Image

**Figure 4.2:** Star Model as 3D Distribution and Appearance on Image Plane

Although this figure represents only an ideal star observed whilst stationary relative to the inertial frame, it has been found to provide sufficient accuracy during star camera operation [32].

## 4.2 Image Processing Techniques

With insight into the appearance of stars in images, it is therefore possible to investigate appropriate star detection methods.

As shown in Figure 4.1, the star tracker application uses image processing as a method of extracting body-referenced vector measurements from star images. This extraction process can be reduced to four main steps: i) the detection of stars; ii) the determination of star centroids; iii) compensation for sensor irregularities and distortion, and iv) optic modelling and the use of projective geometry to determine the position of features relative to the observer.

### 4.2.1 Star Detection

The first step in the determination of body-referenced measurements involves the detection of stars in an image. To simplify matters, images are assumed to be planes of low-intensity values with regions corresponding to stars shown as high-intensity areas. The detection of stars can therefore be enabled by stepping through each pixel and comparing the value to a noise cut-off threshold. If the current pixel in question is found to have a sufficiently high intensity, it is used for further processing.

A graphical representation of the star identification method employed by previous researchers [31], [32] is shown in Figure 4.3:

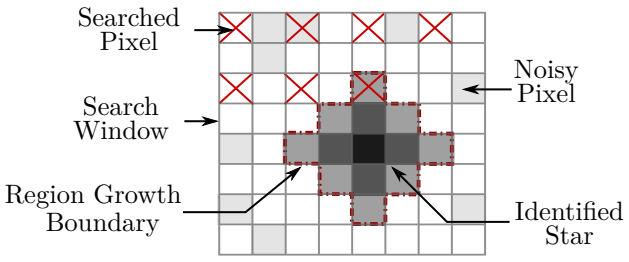



**Figure 4.3:** Image Plane Search

Several optimisations of the algorithm exist. One frequently applied during previous work is that of only searching every other pixel, which subsequently reduces the total computational time.

After the detection of a pixel that is assumed to be part of a star, the pixel is then used as the seed location for a recursive region growing search as described in Erlank et al. [45]. After the star detection step has been completed, a list of pixelgroups and locations will be available.

### 4.2.2 Centroid Detection

Owing to the optical defocusing technique used, the stars are represented as pixel groups. To convert these groups to single bodyvectors, the best estimate star centroid is required.

#### Point Spread Function-Fit

A possible method of determining the star centroid is by fitting a normal distribution to the points and determining the peak, as described by Fosu et al. [79]. This is done by linearising Equation (4.1.1) such that:

$$\ln I(r_c) = \ln I(0) - \frac{1}{2\sigma} r_c^2 \tag{4.2.1}$$

Then, with the expansion of the radial distance to the star in terms of pixel coordinates, Equation (4.2.1) can be rewritten as

$$\ln I(r_c) = \ln I(0) - \frac{1}{2\sigma} \left[ (u_i - u_c)^2 + (v_i - v_c)^2 \right] \tag{4.2.2}$$

where the $u_c$ and $v_c$ coordinates represent the star centroid location in the pixel frame in the horizontal and vertical directions, respectively. Subscripts denoted by a $c$ represent the centre location, whilst subscripts denoted by an $i$ represent the $i$th pixel. This equation can then further be simplified as

$$E_i = c_0 + c_1 u_c + c_2 v_c + c_3 u_c^2 + c_4 v_c^2 \tag{4.2.3}$$

In Equation (4.2.3), $E_i$ represents a compound term used in the simplification of the equation and is described by

$$E_i = -2\sigma(\ln I(r_c) - \ln I(0)) \tag{4.2.4}$$

Parameters $c_0$ to $c_4$ can then be estimated in a LSQ fashion with the final estimated $u_c$ and $v_c$ positions calculated by determining the function maxima

$$u_c = -\frac{c_1}{2c_3} \quad , \quad v_c = -\frac{c_2}{2c_4} \tag{4.2.5}$$

Although the Point-Spread Function (PSF)-fit method promises subpixel centroid determination accuracy of 0.1 pixels [79], the overall implementation uses a computationally expensive $5 \times 5$ matrix inversion. As some image frames can contain more than twenty stars, the process will be time consuming owing to the high computational cost. The PSF-fit method is therefore not deemed applicable for a low-power embedded platform.

#### Centre of Mass

An alternative method is that of Erlank [45] and Calitz [31]. This method relies on the calculation of the centre of mass and shows only a small decrease in centroiding accuracy over that of the PSF-fit method. By using the centre of mass method, the star centroid can be related to the pixel groups with

$$(u_c, v_c) = \left( \frac{\sum_{ij} u_{ij} I_{ij}}{\sum_{ij} I_{ij}}, \frac{\sum_{ij} v_{ij} I_{ij}}{\sum_{ij} I_{ij}} \right) \tag{4.2.6}$$

In this equation, $I_{ij}$ represents the pixel intensity of the pixel in the $i$th row and $j$th column.

### 4.2.3 Projective Geometry

Once all star centroids have been determined, all centroid locations, referenced relative to the location of the first pixel, have to be transformed to a different reference frame as the pixel coordinate frame origin does not coincide with that of the body reference frame.

This process employs the pinhole camera model [80], as well as projective geometry in the conversion of the two-dimensional image plane coordinates to three-dimensional sensor relative body vectors. A brief description of the process follows, with a graphical description of the problem given in Figure 4.4.

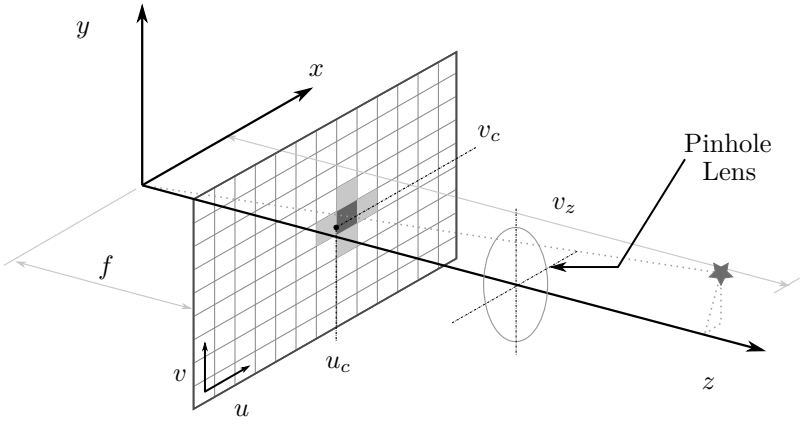



**Figure 4.4:** Image Plane Coordinates

This figure shows a description of the coordinate transform problem. Centroids are measured in the pixel frame $(u_c, v_c)$, but are required in the sensor-body frame. To transform this into meaningful data, two main steps are required.

#### Projection

During this step, a coordinate transform, from two-dimensional pixel coordinates, $(u, v)$, to three-dimensional body coordinates, $(x, y, z)$, is required and can be determined with:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_c - u_o \\ v_c - v_o \\ f \end{bmatrix} \tag{4.2.7}$$

In this equation, $f$ represents the calibrated focal length and $(u_o, v_o)$, the point of intersection of the image plane and boresight in pixels. $(S_x, S_y)$ describe the image sensor dimensions in mm per pixel and are obtained from the datasheet. From these equations, it can be noted that the calculated body coordinates assume that the image plane is situated a distance $f$ away from the focal point in the $z$-direction.

### Coordinate Transform

In the case of the star tracker, vector measurements are assumed to have a unit norm. By using this unit norm constraint, the transform can be determined as

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \frac{v_z}{f} \tag{4.2.8}$$

where the $v_z$-component is given by

$$v_z = \sqrt{\frac{f^2}{f^2 + x_c^2 + y_c^2}} \tag{4.2.9}$$

For a more complete description of the projective geometry process, the reader is referred to Erlank [32].

## 4.3 Distortion Correction

If ideal rectilinear optics are used, such that an image describes a perfect mapping of a real world scene on a tangent plane, the determination of star locations relative to the observing sensor would only involve the steps described in Section 4.2.3. Because of the nonidealities in practical observation systems, however, distortion is introduced into measurements. The main cause of the aforementioned optical distortion is due to irregular magnification of lenses and alignment errors brought on by the manufacturing process of the optics.

Because of this, the true projection of a star centroid on an image plane is described by its true position with an added distortion term.

### 4.3.1 Distortion Types

There are two main types of geometric distortion that impact optical accuracy, namely radial and tangential distortion. The effect of these distortion types on feature locations is shown in Figure 4.5.

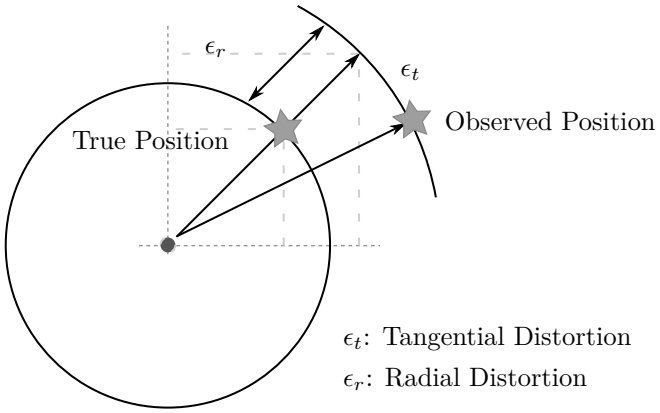
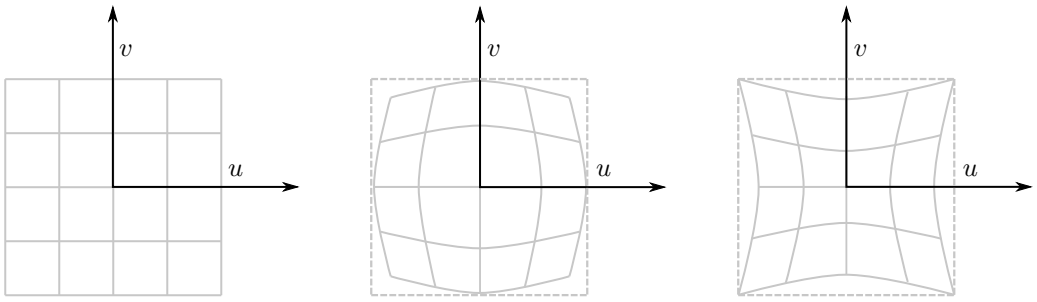



**Figure 4.5:** Effect of Distortion on Observed Feature Position

Radial distortion can further be subdivided into two commonly observed types: negative radial displacement, more commonly known as barrel distortion, and positive radial displacement or pincushion distortion [81]. These radial distortion types are shown in Figure 4.6.

Although a variety of different models for compensating for optical distortion exists, only two will be discussed. One of the most common distortion models is that of Brown [82]. As this model was extensively treated by Calitz [31] and Erlank [32] in previous work as related to CubeStar,




**(a)** Distortionless Image **(b)** Barrel Distortion **(c)** Pincushion Distortion

**Figure 4.6:** Common Optical Distortion Types

only a brief overview will be given. In these two models, the radial distance to a feature, $r$, is defined as the Pythagorean distance from the optical centre.

### 4.3.2 Brown's Distortion Model

The reduced-order Brown's distortion model is given in Equations (4.3.1) and (4.3.2). Important to note here is that the distorted locations, $x'_c$ and $y'_c$, are denoted by primes.

$$x_c \approx x'_c(1 + K_1 r_o^2 + K_2 r_o^4) + P_2(r_o^2 + 2x_c'^2) + 2P_1 x'_c y'_c \tag{4.3.1}$$

$$y_c \approx y'_c(1 + K_1 r_o^2 + K_2 r_o^4) + P_2(r_o^2 + 2y_c'^2) + 2P_1 x'_c y'_c \tag{4.3.2}$$

This model attempts to solve for the radial distortion parameters $K_1, K_2$, and tangential components $P_1$ and $P_2$ by using the film coordinates of features observed on an image plane.

### 4.3.3 CubeSpace Radial Distortion Model

Recently CubeSpace has opted for a custom distortion model as it is coupled with a simplified, more accurate calibration process. In this model, the distortion is described as:

$$\epsilon_r(u_c, v_c) = K_1 + r_o K_2 + r_o^2 K_3 \tag{4.3.3}$$

The image distortion correction can then be done with

$$\begin{aligned} u_c &= u'_c + u'_c \epsilon_r(u'_c, v'_c) \\ v_c &= v'_c + v'_c \epsilon_r(u'_c, v'_c) \end{aligned} \tag{4.3.4}$$

It is evident from these equations that this distortion model only compensates for the radial distortion.

## 4.4 Lost-In-Space Star Matching

For successful attitude determination, observed body vectors should be matched with inertial counterparts. This matching process can be completed in a variety of ways, depending on whether prior attitude knowledge is available. When no prior attitude knowledge is available the sensor is said to be in a LIS mode.

The following section handles the lost-in-space star matching method, as used in the original development of CubeStar [45]. Any alternative suitable method of star identification can be used as long as it is robust against false detection, and adheres to the processing requirements of a low-power embedded architecture.

The lost-in-space method used is that as developed by Kolomenkin [83]. This algorithm depends on two main parts, mainly: i) offline catalogue generation, and ii) online star matching. A brief description of the implementation of this method follows.

### 4.4.1 Star Catalogue

The offline processing part used in this method depends on the construction of a list of visible stars, as well as a catalogue, or lookup table. The star list, in this case, represents a list of all visible stars, with their positions as Earth-Centered Inertial (ECI) vectors. The second component, the catalogue, relies on a lookup table used during star identification. This table contains star pairs as well as the distance between them. An example of the structure of the catalogue is shown in Table 4.1.

**Table 4.1:** Star Catalogue

| **Entry** | **Star 1** | **Star 2** | **Distance** (mrad) |
|---|---|---|---|
| 0 | 54 | 53 | 6.75 |
| 1 | 53 | 50 | 10.39 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 409 | 47 | 106 | 7.332 |

#### Star List

The star list used on-board CubeStar currently consists of a subset of the Hipparcos catalogue [9] limited to a minimum brightness of 3.8 Mv. This star list contains the positions of 410 of the brightest stars, with an accuracy better than 0.001 arc seconds. As shown in Table 4.2, the star list consists of the inertial positions as vectors, coupled with an identification number.

**Table 4.2:** Star List

| **Star ID** | $X$ | $Y$ | $Z$ |
|---|---|---|---|
| 0 | 0.873 | 0.320 | 0.486 |
| 1 | 0.512 | 0.021 | 0.858 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 409 | 0.213 | -0.019 | 0.977 |

#### Catalogue Generation Process

This process occurs offline. For every star, the angular distance to every other star is determined. If this calculated distance is larger than a certain margin, in this case chosen as the sensor FOV, it is ignored; otherwise both the star pair and interstellar distance in radians are saved.

Once all star pairs have been determined, they are organised according to increasing interstellar distances. This data structure is then saved for use as a lookup table during star identification.

#### Catalogue Size Requirements

Entries to both the star list and catalogue are stored in on-board nonvolatile flash memory. At sensor startup, this catalogue is then read into onboard SRAM to reduce the total memory access time. The star list is stored as three 32-bit floating point arrays, corresponding with the $X$-, $Y$-, and $Z$-coordinates, as well as one 16-bit unsigned integer array holding the star ID. Each star list entry therefore has a byte width of 14 bytes amounting to a total star list memory requirement of 5.6 kB.

A star list of 410 entries and FOV of around 45° corresponds to a catalogue containing 11,740 entries. Each entry in this lookup table consists of two unsigned 16-bit integers, holding the corresponding star ID and one 32-bit floating point value representing the interstellar distance

in radians. Each catalogue entry therefore amounts to 8 bytes, subsequently leading to a total accumulated catalogue memory requirement of 92 kB.

Subsequently, the total accumulated memory requirements of both the star catalogue and star list is approximately 98 kB.

### 4.4.2 Description of Matching Technique

During online matching, two voting stages are then used to match body vectors to their corresponding inertial positions. These are known as i) initial voting, and ii) verification voting.

#### Initial Voting

The initial voting process functions as follows:

1. Compute interstellar angular distance for each star as $d_{ij} = \arccos(\mathbf{v}_i^{\mathcal{B}} \cdot \mathbf{v}_j^{\mathcal{B}})$;
2. Compute the uncertainty region $R_{ij}$ where $R_{ij} = [d_{ij} - \xi_{ij}, d_{ij} + \xi_{ij}]$, where $\xi_{ij}$ represents the maximum permissible distance error between two stars;
3. Locate the catalogue entries that correspond to the uncertainty region $R_{ij}$. This process is shown in Figure 4.7, where the light shaded region shows all catalogue entries included in an uncertainty region of width 2 mrad, for a calculated interstellar distance of 14.31 mrad; and
4. Each star in the star pair then receives a vote for all IDs identified in the previous step.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Star 1: | 53 | 50 | 308 | 72 | 229 | 91 | 157 | 305 | 45 | 50 | 251 |
| Star 2: | 54 | 53 | 311 | 74 | 230 | 93 | 159 | 306 | 46 | 54 | 252 |
| Distance: | 6.75 | 10.39 | 10.43 | 13.54 | 13.72 | 14.44 | 14.73 | 14.80 | 15.93 | 17.11 | 18.01 |

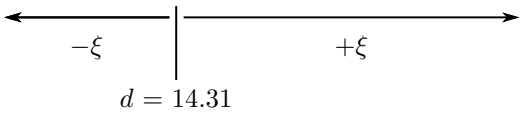



**Figure 4.7:** Geometric Voting Uncertainty Region for $\xi = 2$

This process is then repeated for each star on the image plane. The IDs that received the most votes are then assigned to the stars as an initial match. A second verification step is then implemented to ensure robustness against false matches.

#### Verification Voting

This process comprises two steps:

1. As a validation step, the distance between each star pair in the inertial and body frame is compared: if these vectors show good agreement, both stars receive a validation vote; and
2. After the validation voting stage, stars with the most votes clustered together are matched: stars that have too few matches are discarded.

## 4.5 Tracking Mode

A star tracker functioning only in lost-in-space mode is undesirable, however, as the overall execution speed and computational requirements of lost-in-space matching makes it unfavourable for continuous use. To solve this problem, most star trackers also implement a tracking mode. During tracking mode, a predicted location based on a priori system state knowledge can be used to limit the star search bounds. Star matching can therefore be done by matching detected stars to predicted locations, leading to increased efficiency, higher update rates, and lower power usage.

One of the key components of a tracking mode is that of limiting the star search locations to encompass only those pixels surrounding the predicted star locations. In some cases, rather than predicting a star location, the stars' previous locations can be used. An example of search region limiting is shown in Figure 4.8.

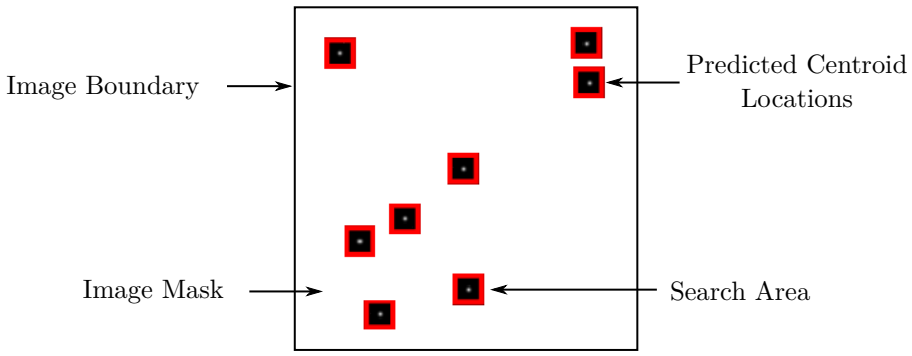



**Figure 4.8:** Example of Reduced Region of Interest Image Plane Search

Here, the red bounding boxes show the search bounds. As seen, the software takes into account only the pixels surrounded by these bounding boxes, not processing any of the masked pixels. If any stars are masked, however, they will not be detected.

The original CubeStar tracking implementation relied on the computation of a reduced star catalogue depending on the initial estimated sensor Boresight Vector (BSV). This method of tracking implementation does not entail a true tracking mode, however, as no search region limiting was implemented.

During this work, an additional algorithm was therefore developed to ensure efficient usage of onboard functionality by enabling a gyro-assisted tracking mode dependent on an orientation EKF.

### 4.5.1 Gyro-Assisted Tracking

Despite the merits of matching previously detected stars during tracking mode, an obvious problem arises when stars enter and leave the current FOV.

If no attempt at constantly identifying new stars entering the FOV is made, intermittent state transitions to LIS will be seen as the number of tracked stars will decrease owing to stars leaving the FOV. Although stars leaving the FOV prove to show little to no increase in complexity, as stars not found can simply be discarded, the efficient identification and verification of new stars prove slightly more difficult.

Three methods were identified and investigated, with which an attitude, predicted by use of a gyro, could be used to enable fast star identification.

### 4.5.2 Identification Methods

#### Full Catalogue Search and Projection

This method is the simplest and promises the best results. However, the overall computational requirements are quite high as each star in the catalogue is compared to the current boresight vector. This method consists of three main steps:

1. Calculate BSV using the current estimated attitude;
2. Compare all the stars in the star list with the inertially referenced BSV; and
3. If the angle between the star and BSV is smaller than that of half of the image sensor FOV, and has not previously been identified, add it to the list of new stars for which to search.

Despite the simplicity of this method, it becomes less feasible for usage as the star list size increases. Although the current CubeStar catalogue consists of only 410 stars, a significant computational effort will have to be expended to determine new stars during tracking mode.

#### Unique Random Neighbour

Another technique of star detection during tracking involves a slight star list augmentation. In this approach, every star in the star list is linked to the star closest to it, as shown in Figure 4.9. Each star in the list is only augmented with a single ID and is only referenced once. The output of this method is a giant linked list structure which can then be traversed from previously identified entities. All stars in the approximate vicinity of those already identified can therefore be found.

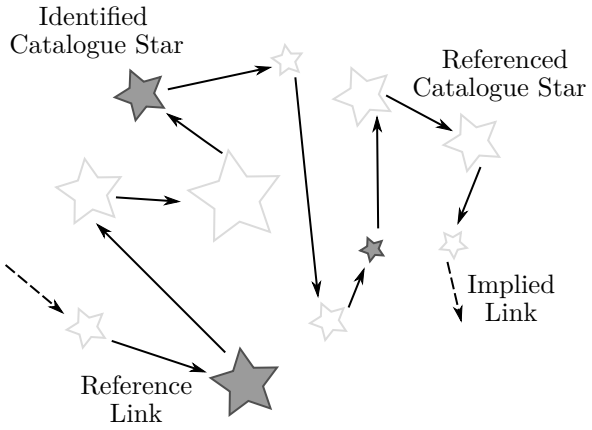



**Figure 4.9:** Graphical Representation of Random Nearest Neighbour Matching

The main concern during this approach subsequently involves the generation of a unique neighbour linked to each star. Numerous methods of generating these reference links were identified, some involving randomly assigning stars to each other, whilst others involved exhaustively assigning the closest stars. It should, however, be noted that the brightest 410 stars are not distributed evenly over the celestial sphere, sparse regions therefore exist. Owing to the movement of the FOV across the celestial sphere, identified stars therefore tend to cluster together, leading to poor identification.

#### Lookup Table Approach

This method relies on projecting the celestial sphere onto a cylinder and grouping stars according to their celestial coordinates. During tracking mode, the estimated sensor boresight can be used as an index in a lookup table with the intent of finding a subcatalogue of stars possibly visible in the FOV. As this method shows a compromise between the high computational cost and the low memory requirements of the previous methods, it was identified as being the most feasible method

for star identification during tracking mode. This method is therefore discussed in more detail in the following subsection.

### 4.5.3 Lookup Table Matching

Although this method promises to be much faster than a complete catalogue search and much more robust than that of the nearest neighbour approach, additional memory is required in storing the lookup table. This was, however, an acceptable compromise, given the increase in computational efficiency. This algorithm relies on the following key processes:

1. Determining a suitable lookup table;
2. Calculating the current predicted boresight location;
3. Searching for relevant new stars;
4. Finding seed locations to use in the reduced Region of Interest (ROI) window; and
5. Verifying identified stellar entities.

A full description of each of these steps follows.

**Preprocessing: Lookup Table Construction** This step can either be performed as a startup procedure, or, to save computational effort, during offline processing. Given a vector from the star catalogue with coordinates $(X, Y, Z)$, the corresponding bin index, starting at (0,0) can be calculated as follows:

$$RA_{ind} = \text{floor}\left(\frac{\arctan\left(\frac{v_y}{v_z}\right) + 180^\circ}{N_{RA}}\right) \tag{4.5.1}$$

$$DEC_{ind} = \text{floor}\left(\frac{\arctan\frac{v_z}{\sqrt{v_x^2+v_y^2}} + 90^\circ}{N_{DEC}}\right) \tag{4.5.2}$$

These equations calculate the celestial coordinates of the stellar body. In this case, the first bin index at position (0,0) corresponds to the celestial coordinates of $-180^\circ$ right ascension and $-90^\circ$ in declination, and the variables $N_{RA}$ and $N_{DEC}$ correspond to the bin widths in degrees. The graphical representation of this lookup table is given in Figure 4.10. The table used is divided into 18 bins in both right ascension and declination such that each bin consists of an area of sky $20^\circ \times 10^\circ$ degrees in size.

The full catalogue is generated by use of the following algorithm:

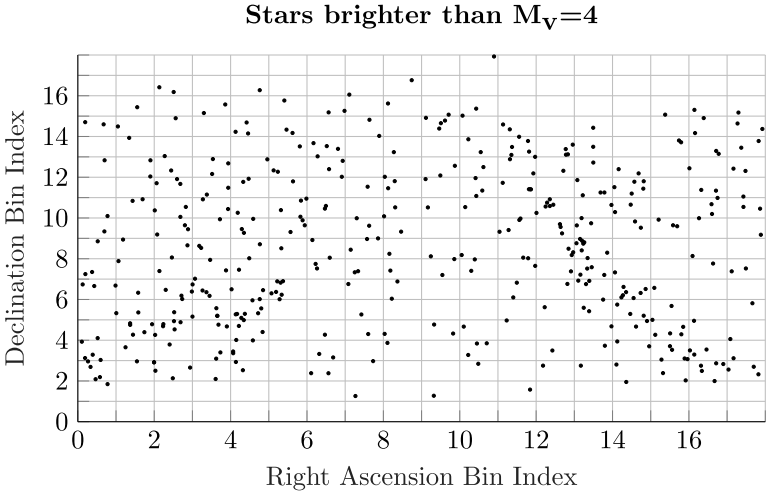



**Figure 4.10:** Stellar Lookup Table

**Algorithm 4.5.1** Lookup Table Construction

```
for i ← 0 to length( starIDlist ) - 1 do
    Determine RA_ind
    Determine DEC_ind
    if DEC_ind > 16 then                                          ▷ Group top pole
        NorthPoleList[ 0 ] ← NorthPoleList[ 0 ] + 1
        PoleIndex ← NorthPoleList[ 0 ]
        NorthPoleList[ PoleIndex ] ← starIDlist[ i ]
    else
        if DEC_ind < 2 then                                       ▷ Group Bottom Pole
            SouthPoleList[ 0 ] ← SouthPoleList[ 0 ] + 1
            PoleIndex ← SouthPoleList[ 0 ]
            SouthPoleList[ PoleIndex ] ← starIDlist[ i ]
        else                                                      ▷ Add to main Table
            LookupTable[ RA_ind ][ DEC_ind ][ 0 ] ← LookupTable[ RA_ind ][ DEC_ind ][ 0 ] + 1
            index ← LookupTable[ RA_ind ][ DEC_ind ][ 0 ]
            LookupTable[ RA_ind ][ DEC_ind ][ index ] ← starIDlist[ i ]
        end if
    end if
end for
```

This algorithm further implements two slight optimisations. Firstly, the table size is optimised by grouping stars close to the poles together as these bins are relatively sparse. A further simplification is to store the number of stars contained in the bin as the first array element.

### Step 1: Boresight Location Prediction

Given that this method is only to be used during tracking mode, it is assumed that the predicted attitude would have stabilised such that attitude transients are at a minimum. Attitude estimates would therefore be sufficiently accurate for correct star position prediction. Firstly the calculated quaternion is used to determine the current attitude matrix by use of Equation (2.4.11). As the sensor boresight vector coincides with the $z$-axis of the body coordinate frame, the boresight vector is then calculated with:

$$\hat{\mathbf{v}}_{bsv}^{\mathcal{I}} = \mathbf{A}_{\mathcal{B}}^{\mathcal{I}} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{4.5.3}$$

after which the corresponding bin index can be calculated as described in Equations (4.5.1) and (4.5.2). In this equation, $\hat{\mathbf{v}}_{bsv}^{\mathcal{I}}$, represents a star in-line with the estimated boresight vector.

**Step 2: Finding Identities of Interest**

Once the boresight index is located, the surrounding bins are searched to determine a subset of identities that can be matched. The angular distance between the boresight and each identity is then calculated with

$$d_j = \arccos\left(\hat{\mathbf{v}}_{bsv}^{\mathcal{I}} \cdot \mathbf{v}_j^{\mathcal{I}}\right) \tag{4.5.4}$$

If the resulting angle is smaller than half the expected sensor FOV and is not actively being tracked, it is added to the newly identified list comprising of stars to be searched for. This process is shown in Figure 4.11.

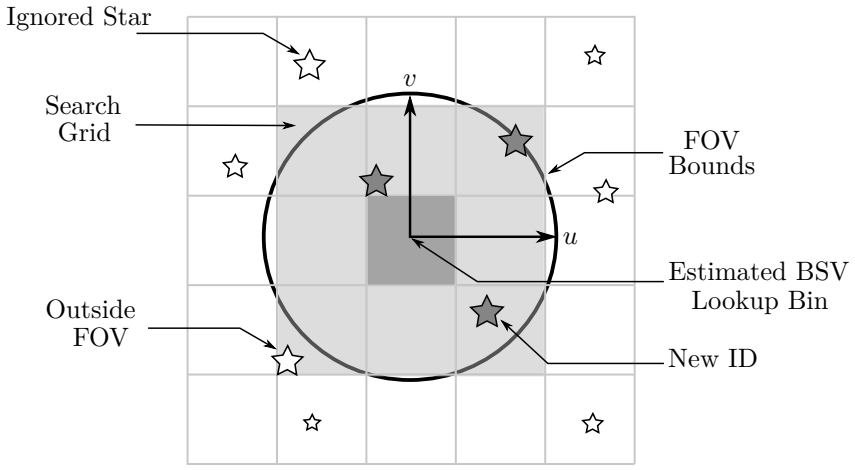



**Figure 4.11:** Graphical Description of Tracking Mode Lookup Process

**Step 3: Reduced ROI Image Plane Search**

The output of Step 2 is therefore a list of stars that might possibly be in the FOV. By using the predicted attitude, the reverse of the steps in the image processing section is then performed so that image plane seed locations can be found. The identified inertially referenced star vectors are therefore:

1. Transformed to the body reference frame by using the predicted inertial to body attitude matrix;
2. Distorted, to ensure a more accurate search location;
3. Projected to image plane coordinates by use of the calibrated focal length and image plane parameters; and
4. Transformed to pixel coordinates.

Once this list of search locations is found, a small ROI is searched around the predicted location. If no star is found, the identity is thrown away; if a star is found, however, the algorithm proceeds to Step 4.

**Step 4: Verification**

Once all predicted identities have been found and matched, a verification voting algorithm, similar to that used in the lost-in-space mode, is used to verify the matched identities. During this verification mode, the measured distance between all identified catalogue and measured body vectors are compared. If the difference is smaller than the calibrated tracking margin, both stars receive a verification vote. If an identity has less than 70 percent of the maximum votes, the identity is assumed to be a false match and is discarded.

Although this algorithm should be relatively robust against false identification, a trade-off still exists between finding the maximum number of stars in the image field of view, and having increased certainty about those vectors that are available to use.

After all stars have been identified and matched to their inertial counterparts, the system can use these measured vectors in attitude determination.

## 4.6 Attitude Determination

Many methods of attitude determination have been developed over the course of the past century, with methods such as the TRIaxial Attitude Determination (TRIAD), QUaternion Estimator (QUEST), EStimator of the Optimal Quaternion (ESOQ), and the Fast Optimal Attitude Matrix (FOAM) being prominent.

The following section describes some of the important attitude determination algorithms used in this work. The section begins with the TRIAD algorithm, after which a brief description of Whaba's problem is given. The Whaba problem is then followed by some important optimal attitude estimation techniques such as the q-method and QUEST.

### 4.6.1 TRIAD

The TRIAD method is a popular attitude determination method developed by Black in the early 1960s [84]. This method determines the attitude Direction Cosine Matrix (DCM) by using two vector measurements in the construction of an intermediate reference frame as

$$\mathbf{A}_{\mathcal{I}}^{\mathcal{B}} = \mathbf{A}_{\mathcal{T}}^{\mathcal{B}} \mathbf{A}_{\mathcal{I}}^{\mathcal{T}} \tag{4.6.1}$$

Over the years, the TRIAD algorithm has been thoroughly investigated with numerous optimal, and suboptimal solutions to the attitude determination problem. This work is concerned only with the symmetric and asymetric TRIAD method, as described by Markley [84]. Firstly, one of the observed vectors is chosen as a basis vector for an intermediate frame. In the asymmetrical case, this vector is usually the one with the least uncertainty, such that.

$$\mathbf{w}_1 \equiv \mathbf{s}^{\mathcal{I}} \quad , \quad \mathbf{r}_1 \equiv \hat{\mathbf{s}}^{\mathcal{B}} \tag{4.6.2}$$

In both reference frames, a new vector is then found that is orthogonal to the two measurements such that

$$\mathbf{w}_2 \equiv \frac{\mathbf{s}^{\mathcal{I}} \times \mathbf{v}^{\mathcal{I}}}{||\mathbf{s}^{\mathcal{I}} \times \mathbf{v}^{\mathcal{I}}||} \quad , \quad \mathbf{r}_2 \equiv \frac{\hat{\mathbf{s}}^{\mathcal{B}} \times \hat{\mathbf{v}}^{\mathcal{B}}}{||\hat{\mathbf{s}}^{\mathcal{B}} \times \hat{\mathbf{v}}^{\mathcal{B}}||} \tag{4.6.3}$$

Given that $\mathbf{r}_1$ and $\mathbf{r}_2$ are then found to be orthogonal, the last basis vector can then be calculated as

$$\mathbf{w}_3 = \mathbf{w}_1 \times \mathbf{w}_2 \quad , \quad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \tag{4.6.4}$$

From this it follows that the attitude matrices describing the orientation relative to the intermediate frame can be constructed from these three vector sets such that

$$\mathbf{A}_{\mathcal{T}}^{\mathcal{B}} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix} \quad , \quad \mathbf{A}_{\mathcal{T}}^{\mathcal{I}} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \mathbf{w}_3 \end{bmatrix}$$

Markley [84], however, further shows that the asymmetric estimator proves much more reliable for attitude estimation by use of sun vector and magnetometer measurements. In the case of star

trackers, Markley claims that the symmetrical triad shows much better performance. This form of the TRIAD algorithm can be constructed by defining the following two unit vectors:

$$\mathbf{r}_1 \equiv \frac{(\mathbf{s}^{\mathcal{B}} + \mathbf{v}^{\mathcal{B}})}{||\mathbf{s}^{\mathcal{B}} + \mathbf{v}^{\mathcal{B}}||} \quad , \quad \mathbf{r}_2 \equiv \frac{(\mathbf{s}^{\mathcal{B}} - \mathbf{v}^{\mathcal{B}})}{||\mathbf{s}^{\mathcal{B}} - \mathbf{v}^{\mathcal{B}}||} \tag{4.6.5}$$

It then directly follows that the set $\{\mathbf{r}_1, \mathbf{r}_2, (\mathbf{r}_1 \times \mathbf{r}_2)\}$ forms a basis of orthogonal vectors with which the attitude DCM can be determined. The classic TRIAD method relies on only two vectors for full three-axis attitude determination, as long as measurements are not colinear. In the case of modern attitude determination, single sensors are apt to output multiple vector measurements. To address this, suboptimal methods were developed with which a vector average can be used instead of the individual vector measurements [84].

### 4.6.2 Wahba's Problem

A better-known problem with regards to attitude determination by use of multiple vector observations was originally posed by Wahba [85]. This problem is expressed as

$$J = \frac{1}{2}\sum_{k=1}^{N} a_k \left|\left|\hat{\mathbf{v}}_k^{\mathcal{B}} - \mathbf{A}_{\mathcal{I}}^{\mathcal{B}}\mathbf{v}_k^{\mathcal{I}}\right|\right|^2 \tag{4.6.6}$$

By expanding this loss function, it can be rewritten as

$$J = \lambda_0 - \text{tr}(\mathbf{A}_{\mathcal{I}}^{\mathcal{B}}\mathbf{B}^T) \tag{4.6.7}$$

where

$$\lambda_0 = \sum_{k=1}^{N} a_k \quad \text{and} \quad \mathbf{B} = \sum_{k=1}^{N} a_k \left(\hat{\mathbf{v}}_k^{\mathcal{B}}\mathbf{v}_k^{\mathcal{I}}\right) \tag{4.6.8}$$

Over the years, a variety of solutions have been proposed to the Wahba problem [86], [87], among some of the most frequently referenced are that of Davenport's q-method and QUEST.

### 4.6.3 Davenports Q-Method

One of the solutions to this optimal attitude estimation problem involves the parametrisation of an attitude matrix $\mathbf{A}_{\mathcal{I}}^{\mathcal{B}}$ by use of a quaternion. This paramaterisation is described as

$$\mathbf{A}_{\mathcal{I}}^{\mathcal{B}} = (q_4^2 - |\mathbf{q}_v|)\mathbf{I} + 2\mathbf{q}_v\mathbf{q}_v^T - 2q_4[\mathbf{q}_v]_\times \tag{4.6.9}$$

From the properties of this attitude representation, the trace from Equation (4.6.7) can be expanded such that

$$\text{tr}(\mathbf{A}_{\mathcal{I}}^{\mathbf{B}}\mathbf{B}^T) = \mathbf{q}\mathbf{K}\mathbf{q} \tag{4.6.10}$$

with $\mathbf{C}$ defined as:

$$\mathbf{C} \equiv \begin{bmatrix} \mathbf{S} - \mathbf{I}s & \mathbf{z} \\ \mathbf{z} & s \end{bmatrix} \tag{4.6.11}$$

and the elements of $\mathbf{C}$ is given as

$$\mathbf{S} \equiv \mathbf{B} + \mathbf{B}^T \tag{4.6.12}$$

$$\mathbf{z} \equiv \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{bmatrix} = \sum_k a_k \hat{\mathbf{v}}_k^{\mathcal{B}} \times \mathbf{v}_k^{\mathcal{I}} \tag{4.6.13}$$

$$s = \text{tr}\,(\mathbf{B}) \tag{4.6.14}$$

The quaternion that solves this optimisation problem can then be found by determining the eigenvector corresponding to the largest eigenvalue, $\lambda_{max}$ of the matrix $\mathbf{C}$ such that

$$\mathbf{C}\mathbf{q} \equiv \lambda_{max}\mathbf{q} \tag{4.6.15}$$

Solving the eigenproblem is, however, a computationally expensive process; even more so on an embedded platform.

### 4.6.4 QUEST

To decrease the computational requirements of the q-method, the optimal eigenvalue, $\lambda_{opt}$, can be approximated as

$$\lambda_{opt} \approx \lambda_0 \tag{4.6.16}$$

This approximation is known as QUEST, or the QUaternion ESTimator. Solving for the attitude is then simply solving for a vector of Rodriques parameters [88], $\mathbf{p}$, such that

$$\hat{\mathbf{q}} = \frac{1}{\sqrt{1+\mathbf{p}^T\mathbf{p}}}\begin{bmatrix}\mathbf{p}\\1\end{bmatrix} \tag{4.6.17}$$

Where the vector $\mathbf{p}$ is calculated as

$$\mathbf{p} = \left[(\lambda_0 + s)\,\mathbf{I}_{3\times3} - \mathbf{S}\right]^{-1}\mathbf{z} \tag{4.6.18}$$

This process can subsequently reduce the number of floating point operations of the q-method by a factor of two to five [86]. Although a $3\times3$ matrix inverse can also be computationally expensive, the process can further be simplified by using Gaussian elimination.

Although these methods offer a good solution when only estimating the orientation, they do not necessarily grant the option of fusing numerous data types such as rate measurements and star vectors. A solution to this is the usage of an EKF.

## 4.7 State Estimation and Kalman Filtering

All previously mentioned attitude estimation techniques only relied on vector measurements. However, in the scope of this project, rate data are also available. A method of combining the star tracker and IMU measurements, is by use of an EKF. An EKF is a recursive algorithm used in the computation of a system state by use of successive linearisation through means of partial derivatives.

The general form of the EKF is formulated such that

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), t) + \mathbf{g}(\mathbf{x}(t), t)\mathbf{w}(t) \tag{4.7.1}$$

where $\mathbf{x}(t)$ describes system states, and $\mathbf{f}$ and $\mathbf{g}$ represent nonlinear functions. The EKF then attempts to estimate the system states from nonlinear measurements, $\mathbf{z}_k$, related to the system states such that

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}(t_k)) + \mathbf{m}_k \tag{4.7.2}$$

In the above equations, $\mathbf{w}(t)$ represents zero-mean, normally distributed noise with a spectral density given as $\mathbf{Q}_k$. Similarly, the measurement uncertainty, $\mathbf{m}_k$, represents a number sequence distributed such that $\mathbf{m}_k \sim \mathcal{N}(0, \mathbf{R}_k)$ [89]. The minimum variance state estimate is then found as the conditional mean of the state vector as a function of both time and the accumulated measurement data.

The following section addresses the orientation EKF as described by Lefferts et al. [15]. This filter employs an indirect filtering method based on the error state to enable orientation and gyro bias estimation. This section comprises of the following parts: i) attitude kinematics in the continuous and discrete time; ii) models of the star and rate sensors; iii) governing state equations; and iv) measurement model. The section concludes with a summary of the EKF algorithm.

### 4.7.1 System Model

The attitude kinematics for a system with orientation quaternion $\mathbf{q}$ and angular rate $\boldsymbol{\omega}$ is given as

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega})\mathbf{q} \tag{4.7.3}$$

where the matrix, $\boldsymbol{\Omega}(\boldsymbol{\omega})$, is given as.

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix} \tag{4.7.4}$$

**Discretisation**

By assuming that the rate sampling frequency is high enough so that the system rate stays constant over $\Delta t$, it can be proven that Equation (4.7.3) becomes [12]:

$$\mathbf{q}(t+\Delta t) = \left[\cos\left(\frac{||\boldsymbol{\omega}||\Delta t}{2}\right)\mathbf{I}_{4\times 4} + \frac{1}{||\boldsymbol{\omega}||}\sin\left(\frac{||\boldsymbol{\omega}||\Delta t}{2}\right)\boldsymbol{\Omega}(\boldsymbol{\omega})\right]\mathbf{q}(t) \tag{4.7.5}$$

### 4.7.2 Sensor Model

In this work, the gyro model used is the one proposed by Lefferts et al. [15]. This model relates the true rate vector, $\boldsymbol{\omega}$, to the measured rate vector, $\mathbf{u}$, such that

$$\boldsymbol{\omega} = \mathbf{u} - \mathbf{b} - \boldsymbol{\eta}_1 \tag{4.7.6}$$

where the bias, $\mathbf{b}$, is a function of a stationary white noise process such that

$$\dot{\mathbf{b}} = \boldsymbol{\eta}_2 \tag{4.7.7}$$

In this model both $\boldsymbol{\eta}_1$ and $\boldsymbol{\eta}_2$ are zero-mean white noise processes. For the star tracker model, the location of the $j$th body vector, $\mathbf{v}_j^{\mathcal{B}}$, measured relative to the sensor, is related to its corresponding catalogue vector, $\mathbf{w}_j^{\mathcal{I}}$, by use of the true orientation such that

$$\mathbf{v}_j^{\mathcal{B}} = \mathbf{A}_{\mathcal{I}}^{\mathcal{B}}(\mathbf{q})\mathbf{w}_j^{\mathcal{I}} \tag{4.7.8}$$

### 4.7.3 State Equations

From Equations (4.7.3) and (4.7.6) it can then be shown that:

$$\dot{\mathbf{q}} = \frac{1}{2}(\mathbf{u}-\mathbf{b})\otimes\mathbf{q} - \frac{1}{2}\boldsymbol{\eta}_1\otimes\mathbf{q} \tag{4.7.9}$$

which can be expanded to the following nonlinear state equations

$$\dot{\mathbf{q}} = \frac{1}{2}\boldsymbol{\Omega}(\mathbf{u}-\mathbf{b})\mathbf{q} - \frac{1}{2}\boldsymbol{\Xi}(\mathbf{q})\boldsymbol{\eta}_1 \tag{4.7.10}$$

$$\dot{\mathbf{b}} = \boldsymbol{\eta}_2 \tag{4.7.11}$$

As no knowledge of host spacecraft inertia is available, the IMU will function as a replacement for the system dynamic model, subsequently leading to the necessity of an error-state EKF [90]. The perturbation states are defined as:

$$\delta\mathbf{q} = \mathbf{q}\otimes\hat{\mathbf{q}}^{-1} \tag{4.7.12}$$

$$\Delta\mathbf{b} = \mathbf{b} - \hat{\mathbf{b}} \tag{4.7.13}$$

It then directly follows from Equations (4.7.3) and (4.7.12) that:

$$\delta\dot{\mathbf{q}} = \frac{1}{2}[\mathbf{q}_\omega\otimes\delta\mathbf{q} - \delta\hat{\mathbf{q}}\otimes\hat{\mathbf{q}}_\omega] \tag{4.7.14}$$

from which the linearised system model in state-space form can then be written as

$$\begin{bmatrix}\delta\dot{\mathbf{q}}\\ \Delta\dot{\mathbf{b}}\end{bmatrix} = \begin{bmatrix}-[\hat{\boldsymbol{\omega}}]_\times & -\frac{1}{2}\mathbf{I}_{3\times 3}\\ \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3}\end{bmatrix}\begin{bmatrix}\delta\mathbf{q}\\ \Delta\mathbf{b}\end{bmatrix} + \begin{bmatrix}\frac{1}{2}\boldsymbol{\eta}_1\\ \boldsymbol{\eta}_2\end{bmatrix} \tag{4.7.15}$$

Owing to the singularity of the state covariance matrix brought on by the quaternion unity constraint, the necessity of truncating the perturbation state vector arises [15]. This reduced perturbation state vector is then given as

$$\delta\mathbf{x} = \begin{bmatrix}\delta\mathbf{q}_v & \Delta\mathbf{b}\end{bmatrix}^T \tag{4.7.16}$$

By assuming that the attitude perturbation is small, the four-element perturbation quaternion can then be reconstructed by use of the unity constraint such that:

$$\delta q_4 = \sqrt{1 - ||\delta\mathbf{q}_v||} \tag{4.7.17}$$

### 4.7.4 Measurement Model

The preceding section describes the system equations with no means of inputting observations from a star tracker. For this, the system innovation, $\mathbf{e}_j$ equations are required. In this case, the innovation is defined as the cross product of the observed body vector, with its modelled position in the body reference frame. This modelled position is defined as the inertial catalogue vector transformed to the body frame by use of the estimated attitude quaternion, and is calculated as:

$$\mathbf{e}_j = \mathbf{v}_j^{\mathcal{B}} \times \mathbf{A}_{\mathcal{I}}^{\mathcal{B}}(\hat{\mathbf{q}})\mathbf{v}_j^{\mathcal{I}} \tag{4.7.18}$$

By comparing Equations (2.4.3) and (4.7.12), it can be shown that

$$\mathbf{A}(\mathbf{q}) = \mathbf{A}(\delta\mathbf{q})\mathbf{A}(\hat{\mathbf{q}}) \tag{4.7.19}$$

The innovation can then be written as a function of only the modelled body vector and perturbation vector such that:

$$\mathbf{e}_j = \mathbf{A}(\delta\mathbf{q})\hat{\mathbf{v}}_j \times \hat{\mathbf{v}}_j \tag{4.7.20}$$

By applying Equation (2.4.11) to the perturbation quaternion, it can be shown that:

$$\mathbf{A}(\delta\mathbf{q}) = \begin{bmatrix} 1 & 2\delta q_3 & -2\delta q_2 \\ -2\delta q_3 & 1 & 2\delta q_1 \\ 2\delta q_2 & -2\delta q_1 & 1 \end{bmatrix} + \mathcal{O}(\delta\mathbf{q}) \tag{4.7.21}$$

If the perturbation is assumed to be small, which in most cases will be true, the high-order terms can the neglected. The perturbation rotation matrix can then be defined by only the low-order terms as shown in Equation (4.7.21). Subsequently, the following simplification can be applied

$$\begin{aligned} \mathbf{e} &= [\hat{\mathbf{v}}]_{\times}^{T}\mathbf{A}(\delta\mathbf{q})\hat{\mathbf{v}} + \mathcal{O}(\delta\mathbf{q}) \\ &= 2\begin{bmatrix} \hat{v}_z^2 + \hat{v}_y^2 & -\hat{v}_x\hat{v}_y & -\hat{v}_x\hat{v}_z \\ -\hat{v}_y\hat{v}_x & \hat{v}_z^2 + \hat{v}_x^2 & -\hat{v}_y\hat{v}_z \\ -\hat{v}_z\hat{v}_x & -\hat{v}_z\hat{v}_y & \hat{v}_x^2 + \hat{v}_y^2 \end{bmatrix}\begin{bmatrix} \delta q_1 \\ \delta q_2 \\ \delta q_3 \end{bmatrix} + \mathcal{O}(\delta\mathbf{q}) \end{aligned} \tag{4.7.22}$$

which can be written in the form

$$\mathbf{e} = \mathbf{H}(\hat{\mathbf{v}})\delta\mathbf{q}_v + \mathbf{m} \tag{4.7.23}$$

with $\mathbf{m}$ the combined measurement and model noise, distributed such that $\mathbf{m} \sim \mathcal{N}(0, \mathbf{R})$. For a complete derivation of the EKF as well as the determination of the covariance of the process noise, the reader is referred to Lefferts et al. [15], with similar work described in work done by Ahmadi et al. [91], Markley [92] and Sola [93].

#### Noise Covariance

To propagate the Kalman filter uncertainty, an estimate of the process noise covariance is further required. In this case, this was determined as:

$$\mathbf{Q} = \begin{bmatrix} \left(\frac{\Delta t}{4} + \frac{\Delta t^2}{32}\right)\sigma_{\eta_1}^2\mathbf{I} & -\frac{\Delta t}{16}\sigma_{\eta_1}^2\mathbf{I} \\ -\frac{\Delta t}{4}\sigma_{\eta_2}^2\mathbf{I} & \sigma_{\eta_2}^2\mathbf{I} \end{bmatrix} \tag{4.7.24}$$

Here $\sigma_{\eta_1}^2$ represents the gyro noise covariance, and $\sigma_{\eta_2}^2$ the covariance of the process responsible for the bias drift.

### 4.7.5 EKF Algorithm

The Kalman filter, as implemented, consists of three functional states: i) initialisation, ii) prediction, and iii) correction. A brief overview of each of these states follows. This process is explained in Figure 4.12.

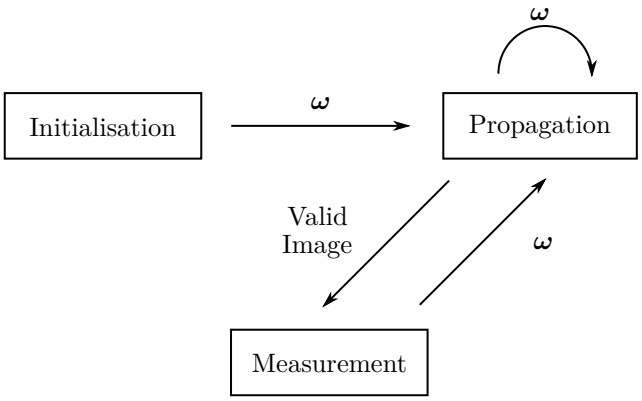



**Figure 4.12:** EKF State Flow Diagram

### Initialisation

During EKF initialisation, an initial attitude estimate is generated by means of an alternate, non-model-based method, such as QUEST or TRIAD. This ensures a faster filter transient response, leading to less overshoot.

### Prediction

Once a valid attitude has been estimated, rate information can be used to enable attitude propagation when star tracker vector measurements are not available. The order of operations during the prediction phase is as follows:

1. Determine the bias corrected gyro measurements with

$$\hat{\boldsymbol{\omega}}_k = \mathbf{A}_{\mathcal{G}}^{\mathcal{B}}\mathbf{u}_k - \mathbf{b}_{k/k} \tag{4.7.25}$$

   where the matrix $\mathbf{A}_{\mathcal{G}}^{\mathcal{B}}$ signifies the relative rotation between the gyro and camera axis system. If the IMU and star tracker axes are on top of each other, this can be replaced by the $3 \times 3$ unit matrix $\mathbf{I}_{3\times 3}$;

2. Determine the propagated attitude, $\hat{\mathbf{q}}_{k+1/k}$, by use of the discretised kinematic equation from Equation (4.7.5);

3. Determine the current state transition matrix $\boldsymbol{\Phi}_{k+1/k}$ by discretisation of Equation (4.7.15)

4. Propagate the state covariance matrix such that

$$\mathbf{P}_{k+1/k} = \boldsymbol{\Phi}_{k+1/k}\mathbf{P}_{k/k}\boldsymbol{\Phi}_{k+1/k}^T + \mathbf{Q}_{k+1} \tag{4.7.26}$$

### Measurement

Owing to rate-measurement noise and propagation error, the predicted attitude drifts with time. Once star tracker measurements become available, these predictions can be corrected and the current least-variance EKF states can be estimated. As each star tracker frame can contain multiple matched stars, the following steps can be executed multiple times per image; once for each matched star vector pair.

1. Determine the current sensitivity matrix, $\mathbf{H}_{k+1/k}$, by using the estimated vector position as described in Equation (4.7.22);

2. Determine the feedback gain vector with

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1/k}\mathbf{H}_{k+1/k}^T[\mathbf{H}_{k+1/k}\mathbf{P}_{k+1/k}\mathbf{H}_{k+1/k}^T + \mathbf{R}]^{-1} \tag{4.7.27}$$

3. Determine the system innovation, $\mathbf{e}_{k+1}$, by using Equation (4.7.18);

4. Update the current six-element perturbation state with

$$\begin{bmatrix} \delta\mathbf{q}_{v,k+1} \\ \Delta\mathbf{b}_{k+1} \end{bmatrix} = \mathbf{K}_{k+1}\mathbf{e}_{k+1} \tag{4.7.28}$$

5. Calculate the four-element perturbation quaternion, $\delta\mathbf{q}_{k+1}$ by using Equation (4.7.17);

6. Determine the estimated states such that

$$\hat{\mathbf{q}}_{k+1/k+1} = \delta\mathbf{q}_{k+1} \otimes \hat{\mathbf{q}}_{k+1/k} \tag{4.7.29}$$

$$\hat{\mathbf{b}}_{k+1/k+1} = \Delta\mathbf{b}_k + \Delta\mathbf{b}_{k+1} \tag{4.7.30}$$

7. Recompute the sensitivity matrix, $\mathbf{H}_{k+1/k+1}$, by using the newly estimated attitude, $\hat{\mathbf{q}}_{k+1/k+1}$; and

8. Update the state covariance matrix

$$\mathbf{P}_{k+1/k+1} = \left[\mathbf{1}_{6\times6} - \mathbf{K}_{k+1}\mathbf{H}_{k+1/k+1}\right]\mathbf{P}_{k+1/k}\left[\mathbf{1}_{6\times6} - \mathbf{K}_{k+1}\mathbf{H}_{k+1/k+1}\right]^T + \mathbf{K}_{k+1}\mathbf{R}\mathbf{K}_{k+1}^T \tag{4.7.31}$$

## 4.8 Rate Estimation

Previous research by Calitz [31] investigated the design of a stellar gyro algorithm for the determination of sensor rates from star tracker vector measurements. The estimation technique employed by Calitz was developed by Crassidis [30]. Consider a system state description such that

$$\mathbf{v}_i^{\mathcal{B}}(k) = \mathbf{A}_{\mathcal{I}}^{\mathcal{B}}(k)\mathbf{v}_i^{\mathcal{I}} + \boldsymbol{\eta}_i(k) \tag{4.8.1}$$

where $\mathbf{v}_i^{\mathcal{B}}(k)$ describes an observed star in the body frame; $\mathbf{A}_{\mathcal{I}}^{\mathcal{B}}$, is a proper orthogonal matrix describing the transformation from an inertial to the current body frame; $\mathbf{v}_i^{\mathcal{I}}(k)$, is the observed vectors coordinates in the inertial frame; and $\boldsymbol{\eta}_i(k)$ is the measurement noise. Given this system state description, the least squares estimate of the sensor rates can be expressed as

$$\hat{\boldsymbol{\omega}}^{\mathcal{B}} = \frac{1}{\Delta t}\left\{\sum_{i=1}^{n}\bar{\sigma}_i^{-2}[\mathbf{v}_i^{\mathcal{B}}(k)\times]^T[\mathbf{v}_i^{\mathcal{B}}(k)\times]\right\}^{-1}\sum_{i=1}^{n}\bar{\sigma}_i^{-2}[\mathbf{v}_i^{\mathcal{B}}(k)\times]^T\mathbf{v}_i^{\mathcal{B}}(k+1) \tag{4.8.2}$$

where $[\mathbf{v}_i^{\mathcal{B}}(k)\times]$ signifies the cross product matrix obtained from the $i$th body vector measurement during time step $k$. The full derivation of this method can be found in Crassidis [30].

## 4.9 Chapter Summary

In this chapter, the most notable software techniques used on a star tracker were investigated and explained in the fulfilment of the second main research objective. Firstly, a brief overview of the image processing methods was given, so that a list of body vectors can be calculated form a given star image. In this work, a similar approach was used as in Erlank [32] and Calitz [31], as is concluded that their methods have previously been applied with great success.

Work involving the matching of body-referenced star vectors to their inertial counterparts was then discussed. It was concluded that, although Kolomenkins [83] geometric voting algorithm proved sufficient during the development of CubeStar, it will only be used as a LIS matching method in this work. Various tracking modes were then investigated and compared. It was finally decided to use a lookup table-based method, based on the currently estimated sensor boresight vector.

Some methods of attitude and rate determination were then investigated, TRIAD, Davenport's q-method, and QUEST were treated, with an alternative of an EKF, as given by Markley et al. [15]. The EKF was chosen to be the centre point of the developed system.

Finally, the vector-based rate estimation scheme, as explained by Crassidis [30], was given. This rate scheme was used primarily as it has previously been implemented on the CubeStar platform with great success [31].

Before these algorithms can be implemented in an embedded application, all software methods should be verified during simulation.

# Chapter 5

# Simulation

Owing to the intricacies of implementation of complex systems in real world applications, as well as the lack of reliable ground truth data, software and algorithms must first undergo proof-of-concept performance testing. All algorithms and software, as described in the previous section were therefore designed and implemented as subsystems in a simulation environment. This not only served as a proof-of-concept system design, but also functioned as a method of algorithm performance analysis.

This chapter therefore comprises three main sections in fulfilment of the third project objective: firstly, a simulation procedure used during functional Kalman filter testing is designed and some initial results and conclusions are given. Thereafter, a complete system simulation with initial results for the proof-of-concept system is given. Finally, the software is rewritten for the embedded environment and stimulated with simulated inputs during hardware-in-loop tests so that algorithm timing could be determined.

## 5.1 Kalman Filter Verification

As the Kalman filter is used as the basis for the tracking mode, and is one of the most important algorithms used in this work, initial simulation and testing are required. This simulation is done with the Kalman filter isolated under controlled simulation conditions so that filter performance can be verified.

### 5.1.1 Simulation Procedure

During this simulation, filter inputs were generated by use of Simulink and implemented as a MATLAB script. The input data generation procedure is shown in Figure 5.1.

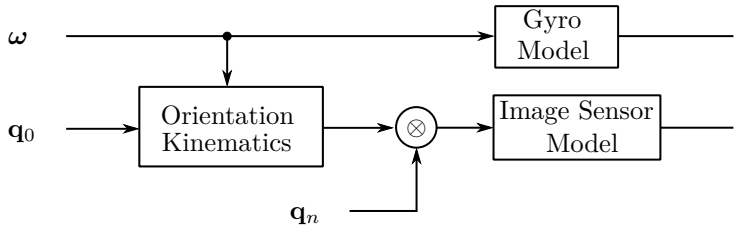



**Figure 5.1:** Simplified Description of Simulation Procedure

True body rates, $\boldsymbol{\omega}$, were generated along with an initial quaternion, $\mathbf{q}_0$, with which the numerical integration process could be initialised. These data were then fed into the quaternion kinematic equation as given by Equation (4.7.3). To emulate a real-world scenario more closely, a noise quaternion was added to represent orientation error. This quaternion is described by $\mathbf{q}_n$, and represents a relative rotation with standard deviation of 0.01°. To emulate sensor measurements, initial sensor models were then constructed as described below.

### 5.1.2 Gyro Model

The gyro model used during the simulation is similar to that used by suggested by Lefferts et al. [15]. In this case the IMU measurements consist of four main parts:

1. Initial bias offset, which is the constant offset measured at IMU start up;
2. Random walk, or bias instability component, of IMU measurements that cause low-frequency random changes in IMU bias;
3. True rate; and
4. A noise component that causes random walk during rate integration.

The gyro model diagram is shown in Figure 5.2.

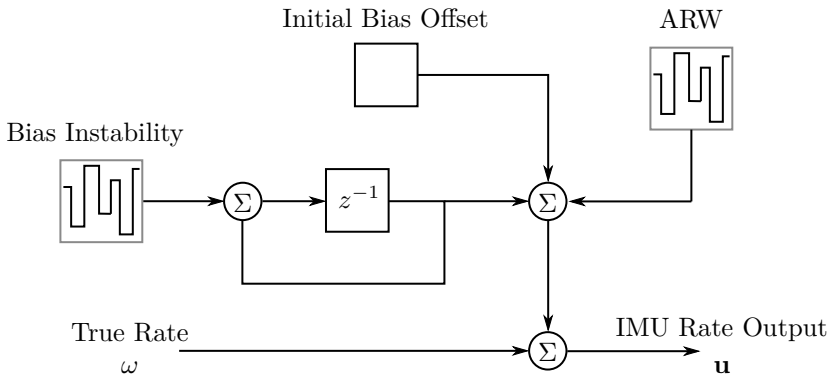
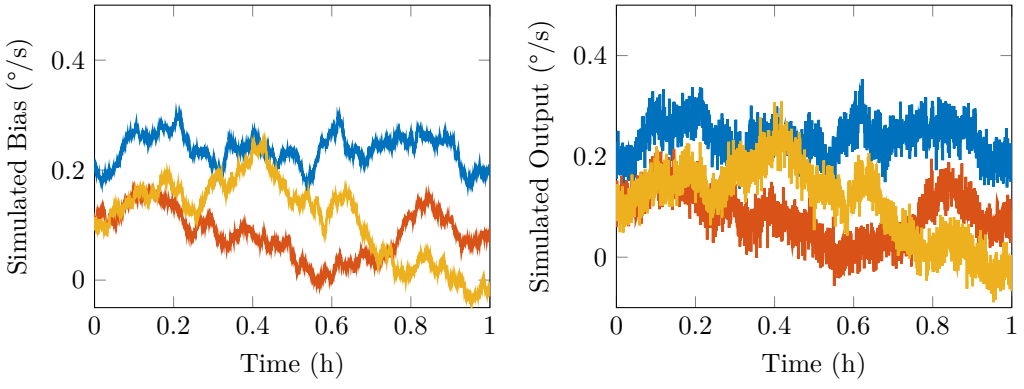



**Figure 5.2:** Gyro Model Used in Simulation

In this figure the blocks representing the bias instability and ARW signify stationary normally distributed processes. To emulate the bias instability, three normally distributed number sequences are generated and integrated over time, giving rise to a random walk. The output angular rate measurement is then obtained by summing the offset, instability, true rate, and ARW. An example of the simulated true bias is shown in Figure 5.3a, with simulated rate measurements shown in Figure 5.3b.




**(a)** Simulated Gyro Bias

**(b)** Simulated Gyro Output

**Figure 5.3:** Simulated IMU Measurements

### 5.1.3 Star Tracker Model

The star tracker output consists of matched inertial and body vector pairs. As the actual vectors are not as important as the rotation between them, it was decided to emulate the star tracker measurements by using randomly generated unit vectors. The generation process for each measurement therefore relied on generating two random numbers used as boresight relative, azimuth, and elevation angles. By assuming a unit radial distance from the observer, a three-element Cartesian vector is then generated. As the true attitude is available, the inertial vector corresponding to each body vector can be obtained.

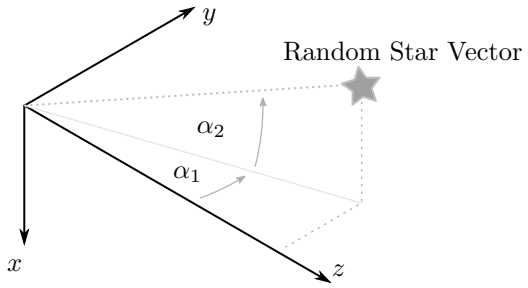



**Figure 5.4:** Random Star Generation

To emulate vector measurement error, three random angles are generated to represent a roll, yaw, and pitch error in each measurement. The originally generated body vector is then rotated by this compound rotation matrix leading to the output body vector. For clarity, this process is shown in Algorithm 5.1.1.

**Algorithm 5.1.1** Star Tracker Measurement Generation

$numVecs \leftarrow$ rand(3,15) ▷ Number stars detected between 3 and 15
**for** $i \leftarrow 1$ to $numsVecs$ **do** ▷ Generate random body vectors
  $[\alpha_1, \alpha_2] \leftarrow$ Generate random relative star angle between -0.5 and 0.5
  $[\alpha_1, \alpha_2] \leftarrow [\alpha_1, \alpha_2] \times fov$
  $\mathbf{v}_i^{\mathcal{B}} \leftarrow$ sph2cart( $\alpha_1$ , $\alpha_2$ , 1 )
**end for**
**for** $i \leftarrow 1$ to $numsVecs$ **do** ▷ Generate corresponding inertial vectors
  $\mathbf{v}_i^{\mathcal{I}} \leftarrow \mathbf{A}_{\mathcal{B}}^{\mathcal{I}}(\mathbf{q})\mathbf{v}_i^{\mathcal{B}}$
**end for**
**for** $i \leftarrow 1$ to $numsVecs$ **do** ▷ Add Measurement Error
  $[\alpha_3, \alpha_4, \alpha_5] \leftarrow$ Generate 3 by 1 random noise angles
  $\hat{\mathbf{v}}_i^{\mathcal{B}} \leftarrow \mathbf{A}(\alpha_3)\mathbf{A}(\alpha_4)\mathbf{A}(\alpha_5)\mathbf{v}_i^{\mathcal{B}}$
**end for**

### 5.1.4 Verification and Results

By using these sensor models, as described in the previous section, a dataset spanning fifteen minutes was then generated. This dataset consisted of:

1. True attitude quaternions sampled at 10 Hz;
2. True rate data sampled at 10 Hz;
3. The number of detected stars sampled at 1 Hz; and
4. A list of corresponding inertial and body vectors also sampled at 1 Hz.

During this simulation, the simulated system experienced a rate corresponding to

$$\boldsymbol{\omega} = \begin{bmatrix} 0.5\sin\left(\frac{t\pi}{100}\right) & 0.1 & -0.2 \end{bmatrix} ^\circ/\text{s} \tag{5.1.1}$$

After exactly five minutes of simulation time, the Kalman filter was enabled. Initial simulation results are shown in Figure 5.5. These plots show the estimated attitude quaternion and IMU bias.

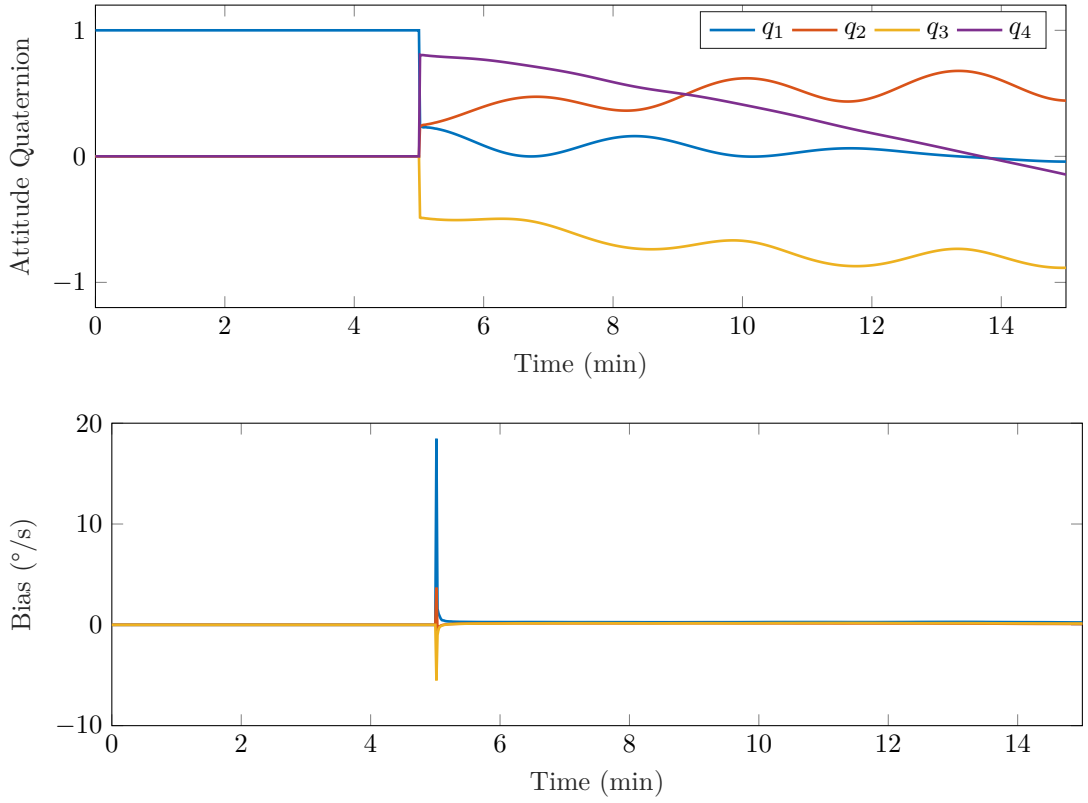



**Figure 5.5:** Simulated Kalman Filter Output Without QUEST Initialisation

Although the filter shows good attitude and bias estimation, a large bias overshoot is observed, owing to errors in the initial state. Although this effect is partly due to the filter covariance initialisation, it is predominantly affected by the initial filter state values. In this case, the lack of prior state knowledge severely impacts the filter settling time. During this simulation the Kalman filter covariance matrix, $\mathbf{P}$, was initialised with values such that

$$\mathbf{P} = \mathbf{I}_{6\times 6} \tag{5.1.2}$$

and the initial system state vector was chosen as

$$\mathbf{x} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & b_x & b_y & b_z \end{bmatrix}^T = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T \tag{5.1.3}$$

During this simulation, the initial filter covariance matrix was varied in an attempt to decrease the bias overshoot. Although changing the filter covariance affected the attitude overshoot, very little changed with regards to the bias overshoot. This is due to the indirect filtering method used during the estimation process. In the case of this filter, any seemingly unexplained error in the attitude estimation is manifested as an estimated IMU bias, subsequently leading to a large state overshoot after filter initialisation.

### 5.1.5 Effect of Initial Attitude Error on Filter Overshoot

Subsequently, the effect of the initial attitude error was investigated so that the impact on bias overshoot could be analysed. During this analysis, a sensitivity analysis was implemented, where the Kalman filter attitude state was initialised with an attitude of increasing error.

In this simulation, the filter was run numerous times during the same conditions as described in the previous subsection. For each run, the initial attitude error along either the roll or yaw-axis was altered by 0.25° until a maximum attitude error of 90° across both axes was incurred. The peak bias state for each individual dataset was then recorded. Results of this simulation were capped at a maximum of 8 °/s, and are shown in Figure 5.6. The reason for limiting the bias overshoot error, was to improve observations of the low-error trends, as, in certain conditions, overshoots in the order of 20 to 100 °/s could be observed. The representation of low variations in bias overshoot were more difficult to observe.

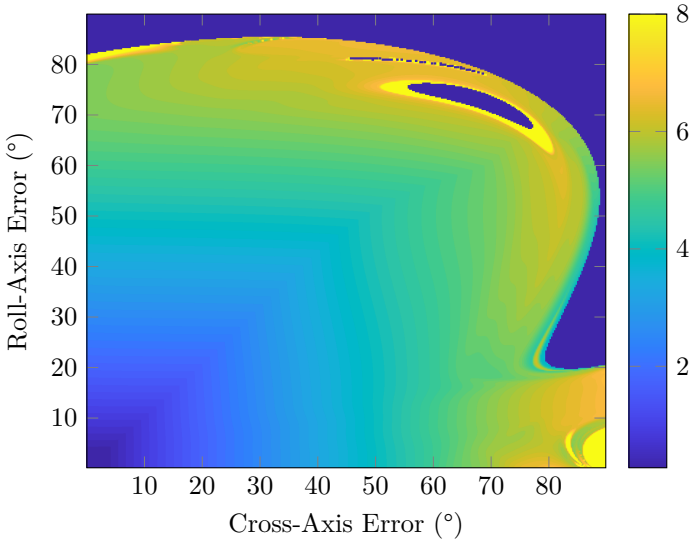



**Figure 5.6:** Effect of Initial Attitude Error on Bias Overshoot

This figure shows that the bias overshoot was largely dependent on the initial attitude error. In this case, roll error has a slightly dominant affect at higher input angular errors. This figure further shows a dark band at the edges close to the 90° error mark. Although these edges appear to be low bias overshoot, it actually represents an invalid starting position that leads to attitude estimation failing. At these starting angular errors, the Kalman filter could not estimate a valid attitude because the estimated $\delta\mathbf{q}_v$, had a norm larger than unity. The determination of scalar part of the quaternion ,$q_4$, therefore became imaginary, leading to unexpected filter behaviour. Subsequently, to ensure correct initialisation, Kalman filter initialisation was determined to be imperative.

Although the initial attitude can be determined by use of any of the attitude determination algorithms given in Section 4.4.1 it was decided to use QUEST. The key reason was because the QUEST algorithm is much more immune to single measurement errors than TRIAD, therefore allowing for a small attitude error and subsequently reducing lengthy filter transient times. QUEST was also successfully used during the original design of CubeStar [32].

### 5.1.6 Effect of Filter Initialisation

With the implementation of Kalman filter initialisation by use of QUEST, the simulation described previously was rerun to analyse the improved filter response. The results obtained are shown in Figure 5.7.

Overall, results relating to bias overshoot showed a significant improvement, as the overshoot was reduced to the order less than 0.1 °/s. Unlike to the uninitialized case, these results show near

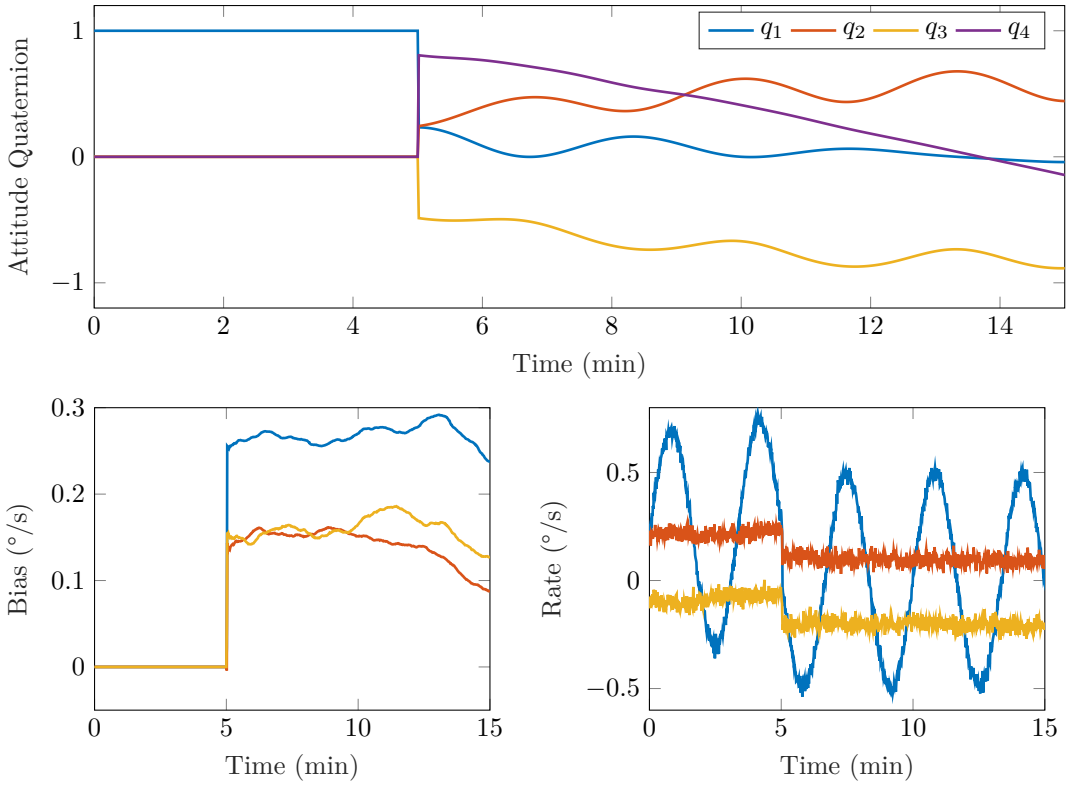



**Figure 5.7:** Simulated Kalman Filter with Filter Initialisation

immediate bias tracking. As in the uninitialized case, attitude estimates show good tracking with little error, albeit slightly improved.

In conclusion to the EKF analyses, it was noted that the algorithm showed successful proof-of-concept functionality with the given system parameters. The final estimation accuracy results were not analysed, however, as the system's accuracy is largely dependent on the simulated measurements which depends on individual sensor characteristics. It was however noted that the Kalman filter showed a similar error response as that obtained by use of QUEST. It is therefore expected to operate with a similar accuracy to the current CubeStar model.

## 5.2 System Simulation

In the context of this work, the EKF represents only a single subsystem in a much greater assembly. To analyse the algorithm interaction, the complete system software was implemented in simulation. The complete system relied on image processing as a means of extracting vector measurements from star data. An augmentation to the current EKF simulation was therefore required, as it relied on randomly generated matched vectors rather than simulated star images.

### 5.2.1 Image Generation and Camera Model

Valid images therefore had to be generated from quaternions. During this process, the CubeStar catalogue of 410 of the brightest stars were used, along with the focal length of an old CubeStar engineering model, given as 6.213 mm.

#### Image Generation Procedure

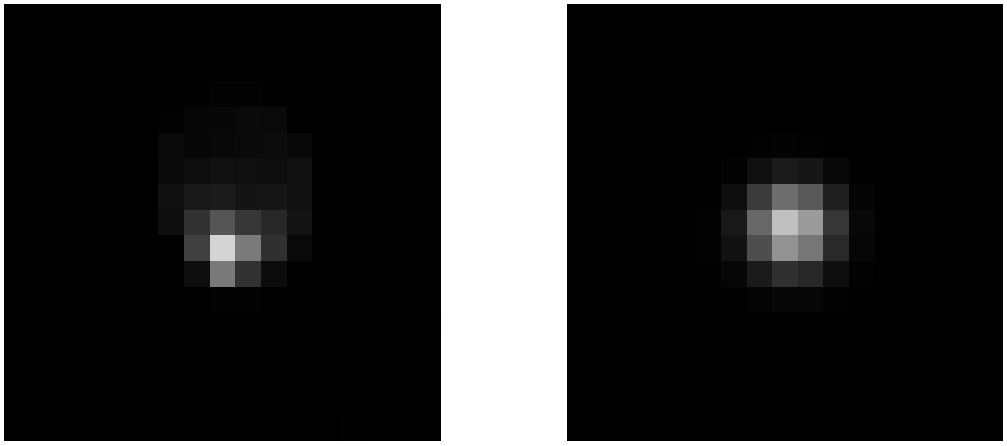
Given a sensor with attitude described by an attitude quaternion, $\mathbf{q}$, the process for generating an image was as follows:

1. The current sensor BSV in the inertial reference system was calculated;

2. A list of stars in range of the boresight was determined;
3. The corresponding image plane star location in pixel coordinates was found by first transforming the stars to sensor-body coordinates, then to image plane coordinates, and finally to pixel coordinates; and
4. Each star PSF was overlayed on a blank image of $1280 \times 1038$ pixels. This size was slightly larger than that described in the hardware design section, as it also takes into account the fourteen dark image columns of the image sensor.

### Star Model Validity

Figure 5.8 shows a comparison between simulated and real star data. These images show good similarities, although subtle differences can be observed. In the case of the simulated star, the overall star variance is slightly smaller than that of the real star, leading to a star with smaller diameter. The simulated star is also slightly dimmer than the real star. This is mainly because of the magnitude differences in the true observed stars, as real stars have varying brightnesses. Thus, some detected stars will be much fainter than the simulated stars, whilst others might be bright enough to cause sensor saturation.



**(a)** Real Star **(b)** Simulated Star

**Figure 5.8:** Star Comparison

This model represents only ideal images, however, as neither image plane noise nor camera distortion was considered during the image generation process. Despite this, actual captured images are still expected to appear quite similar to simulated images during low sensor rotation rates. During high rates, however, this model will be less valid, as long integration times lead to stars appearing as streaks. In this case, the star power will be spread across multiple pixels. As the incident photons become averaged over a larger area, streaks of low intensity will be observed, potentially causing stars to fall below the noise floor. An inverted, simulated example of such a streaky image is shown in Figure 5.9.

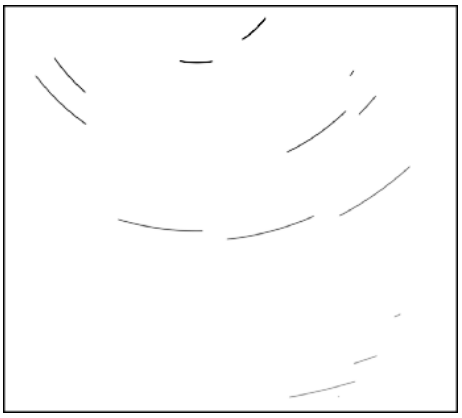


**Figure 5.9:** Inverted Simulated Star Streaks

Although this model would be much closer to a real world situation, it requires an increase in simulation complexity whilst only contributing a small amount to real world accuracy. It was therefore omitted from the image generation process.

### 5.2.2 Simulation Methodology

The EKF system simulation structure was augmented with the emulated star image generation, and the full system implemented in software. The total simulation procedure could be divided into two main sections: i) the generation of system inputs and test measurements, and ii) feeding the test measurements to the simulated system. As in the previous section, the system was implemented in MATLAB and Simulink. The overall simulation structure is shown in Figure 5.10.

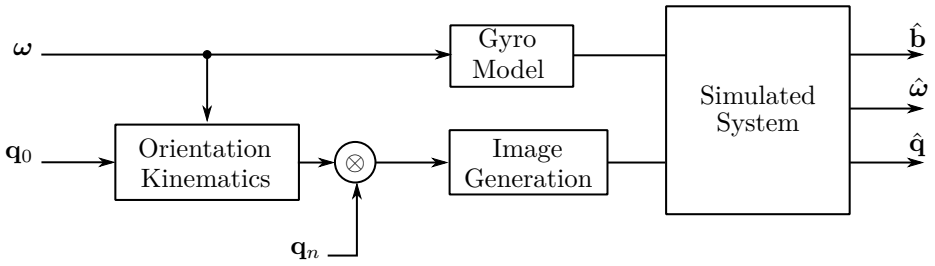



**Figure 5.10:** Simulation Structure

In this case, Simulink was used to simulate both attitude kinematics and the gyro model. The output of this Simulink model was a list of sensor attitudes sampled at 10 Hz, together with simulated gyro data, sampled at the same frequency.

Once an array of attitudes and rates was generated, a list of subsequent images could then be generated. These images, generated at 1 Hz, were then fed into the simulated system at the corresponding simulation time and results were recorded.

### 5.2.3 Simulated System Software Flow

The designed system software flow is shown in Figure 5.11. As can be seen here, the implemented system consists of three main states: lost in space, verification, and tracking.

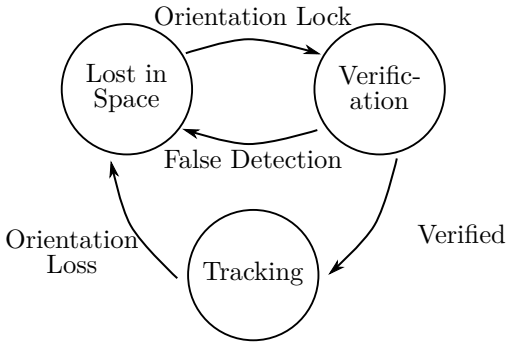



**Figure 5.11:** Implemented System State Flow

During LIS mode, there is no attitude knowledge. A full image plane search is therefore conducted, and all detected stars are fed to the geometric voting algorithm for identification. Thereafter, an initial attitude estimate is obtained by using QUEST, and the Kalman filter is initialised. If no attitude can be estimated, the sensor remains in LIS mode. Once a valid attitude has been determined, the software enters a verification mode.

When the sensor functions in verification mode, the estimated location is not yet used to predict star locations as the software first waits for estimation transients to die down. This perio was chosen as roughly 5 s. During this functional mode, stars are therefore still identified by use of a full image plane search and geometric voting.

Once the attitude has been verified, the sensor enters tracking mode. During tracking mode, the attitude predicted by using IMU measurements ais then used to decrease the overall computational load, as described in Section 4.5.1.

### 5.2.4 Simulation Results

At each possible sampling instant of the simulation, the following data were requested so that results could be analysed: i) sensor attitude, ii) stellar gyro rate estimates, iii) estimated bias and IMU rates, and iv) the number of stars tracked during the tracking mode. These results are analysed below.

#### Attitude Estimation

Figure 5.12 shows the estimated attitude quaternion obtained during the simulation. In contrast to previous simulations, this system was initialised on the first sample.

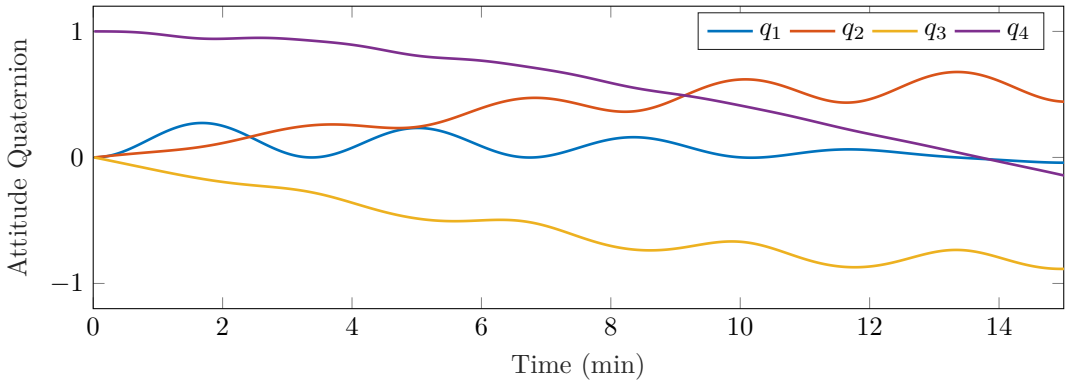



**Figure 5.12:** Attitude Estimation Results

As in the previous cases, attitude estimation was successful and worked as expected, showing good attitude estimation. As mentioned in previous sections, the Kalman filter start up procedure

involved initialisation with the QUEST algorithm. The attitude quaternion shows the same trend as in the previous simulation example, with no discontinuities or loss of attitude lock.

### Stellar Gyro Estimation

For the given input rate, as described in Section 5.1.4, the estimated stellar gyro rates are given in Figure 5.13.

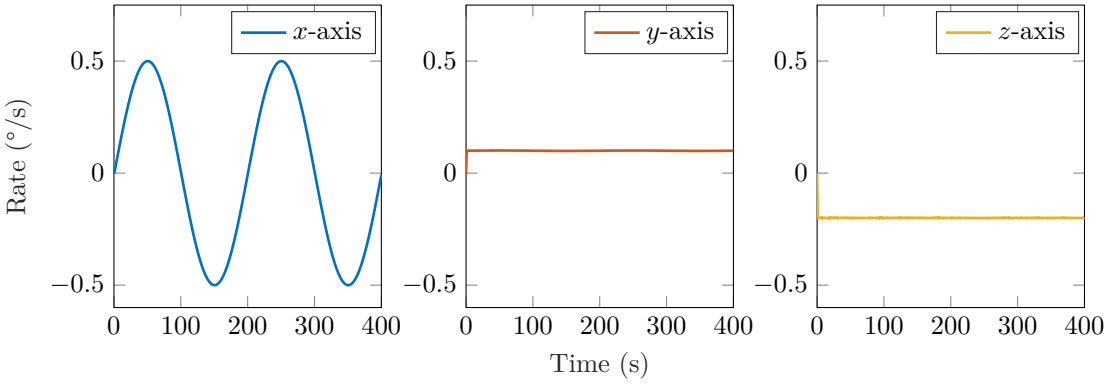



**Figure 5.13:** LSQ Rate Estimation Results

As expected, these estimated rates show excellent rate tracking, with almost no visible sensor noise. To analyse these data further, the residual between the true rate and estimated rates was also determined and is shown in Figure 5.14.

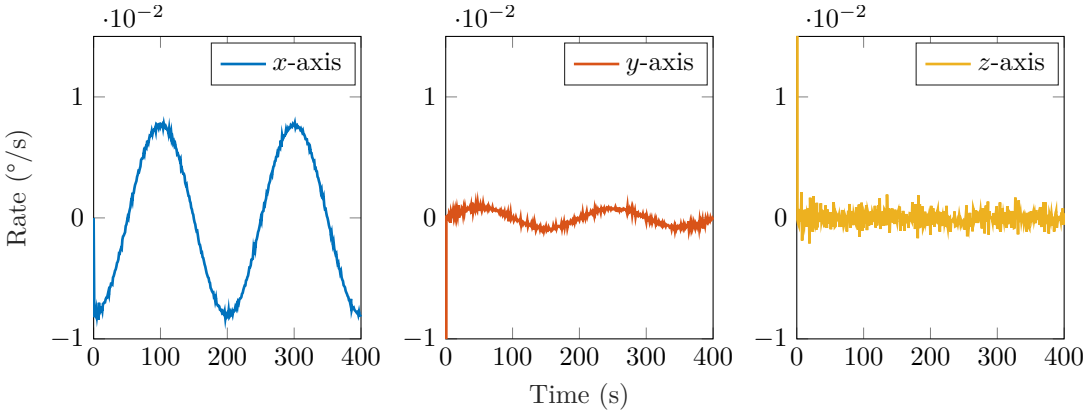



**Figure 5.14:** LSQ Rate Estimation Error

From the Figure 5.14, it can be noted that the residual rates around the $z$-axis show the largest random component. This phenomenon is prominent due to the difficulties in estimating changes around the out-of-plane axis. Larger estimation errors are therefore expected.

In these results, all three of the sensor axes show an initial error spike. This spike exists as the algorithm relies on at least two images such that sets of matched body vectors, one sample apart are available. During the first sample, as only a single image has been processed, no stellar gyro output is available.

A prominent behaviour can be seen in the first plot pf Figure 5.14, around the $x$-axis. In this figure it can be seen that a relatively large sinusoidal residual exists. The main reason for this sinusoidal response is due to the input rate sinusoid and stellar gyro phase lag: as the stellar gyro algorithm has access to only two star images consisting of a time average of the sensor response over the integration period, sinusoidal errors are expected in the sensor output. The residual reaches

a peak when the input rate is close to 0 °/s, as the angular acceleration is at a maximum here. This effect is further exacerbated by the low sampling rate of the sensor, as high-frequency rate dynamics is unobservable.

Although this can be a problem during high dynamic rates, this algorithm is only to be used during periods of slow dynamic spacecraft changes or low sensor rates. Overall, the system shows promising rate estimation results with a high accuracy, albeit slightly delayed response.

### IMU Rate and Bias Estimation

During this simulation, the effects of the bias estimation and IMU rates were also analysed. To serve as a benchmark, the raw IMU measurements are given in Figure 5.15, along with the true rate experienced on each axis as dashed lines.

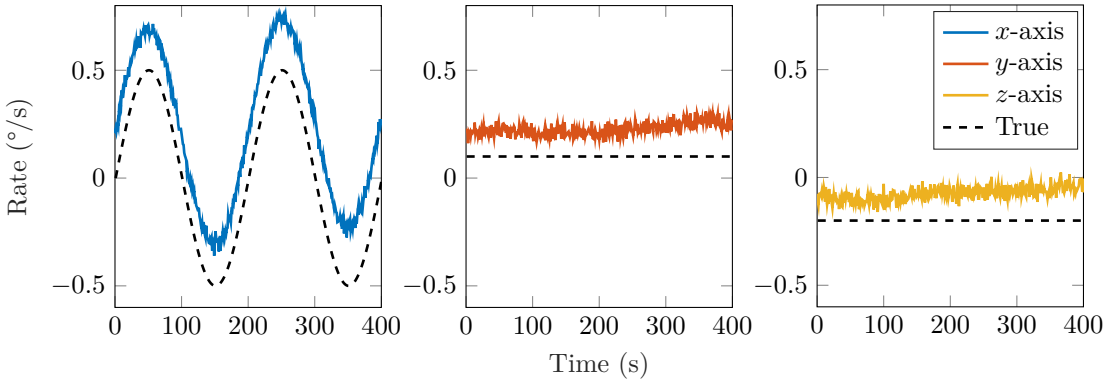



**Figure 5.15:** Simulated IMU Measurements

These figures clearly show the bias drift as a function of time, along with the initial bias offset error. In comparison to these raw measurements, the bias-compensated IMU rates are shown in Figure 5.16.

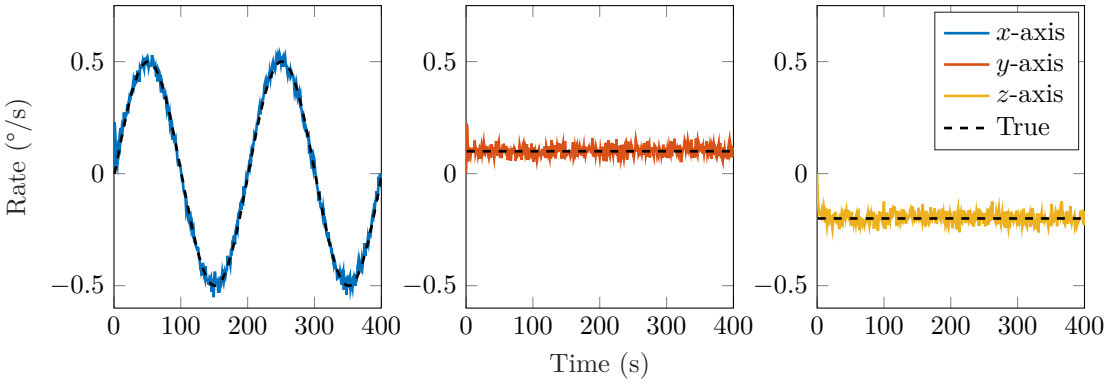



**Figure 5.16:** Bias-Corrected IMU Measurements

The measurement residual in terms of estimated bias error is given in Figure 5.17. This error is computed by subtracting the true bias from the estimated bias, and then plotting the residual.

Overall, the bias estimation shows successful operation with transient behaviour dying down quickly, and peak residual being at least an order of magnitude lower than the actual sensor bias. Bias estimation therefore shows good performance during both dynamic changing and constant rates.

In comparison to the stellar gyro rate estimates, however, the corrected IMU rates show far more coarse measurements, owing to the high ARW of the IMU. This is unfortunately not something

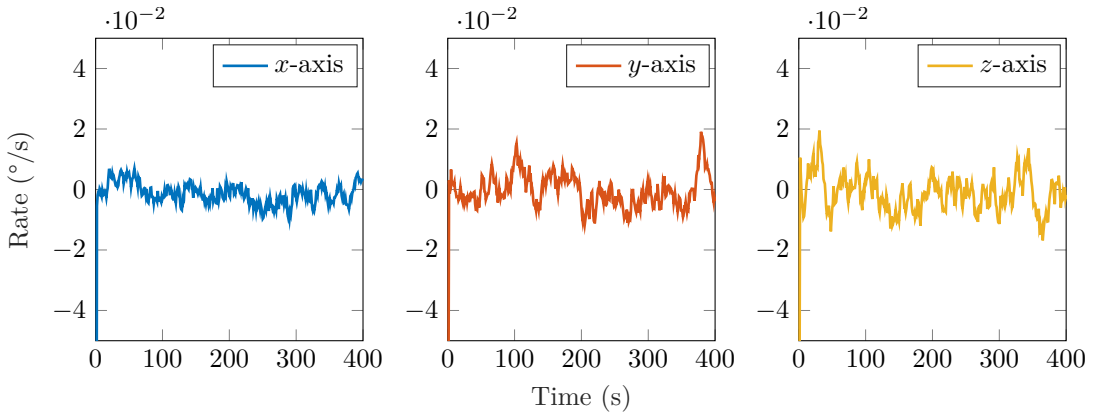



**Figure 5.17:** Estimated Bias Error

that can be fixed without decreasing the ARW. Although simple in simulation, changes like these in real-world applications require the use of expensive, large, and resource-intensive sensors, which are not necessarily feasible.

### Star Identification and Tracking

The final sensor functionality to be tested was that of the improved star tracking mode. As the tracking mode uses a ROI search, only image snippets are available for search, directly impacting the number of stars that can be found in the FOV. The following datasets show the effect of varying the number of bins used during the lookup table search

The tracking mode was tested under two cases. Firstly, only a $3 \times 3$ grid was searched around the boresight bin index, whereafter the search width was changed to a $5 \times 5$ grid. An illustration of the search bin width is shown in Figure 5.18. During these simulations, each bin comprised of an area of 20° in right ascension by 10° declination.

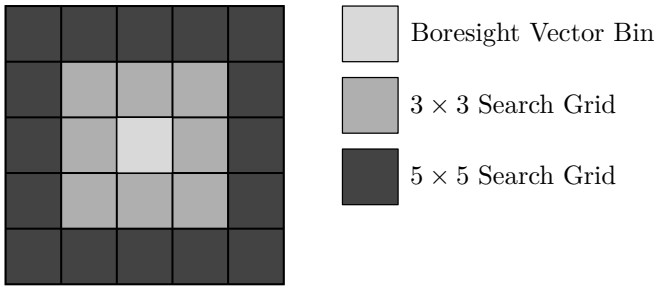
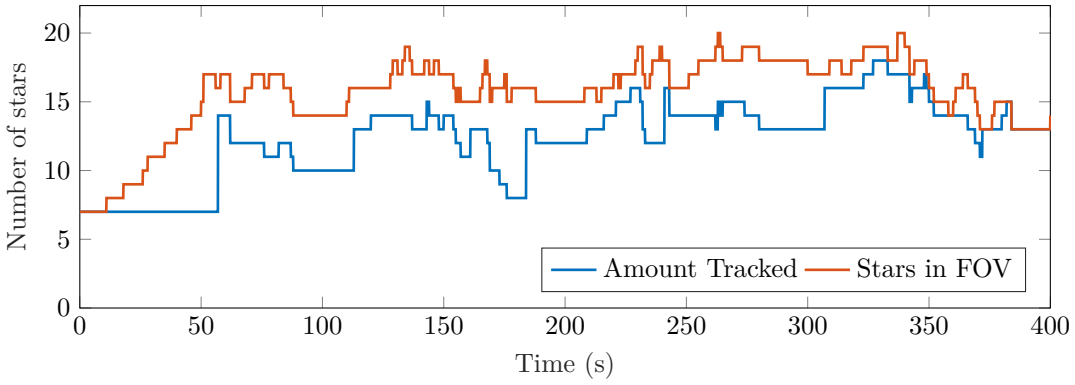



**Figure 5.18:** Graphical Interpretation of Varying Bin Search Size

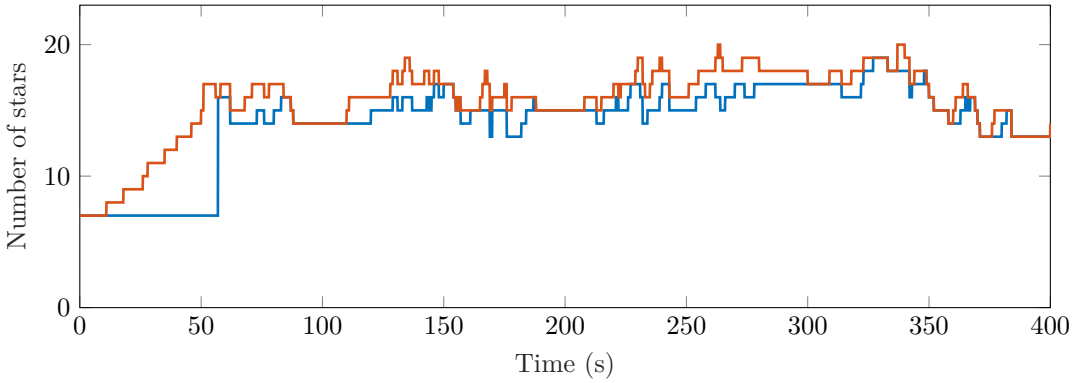
The simulation was then used to determine the effect of using different search grids. These results are shown in Figure 5.19. Once again, the same starting conditions as in the previous simulations were used. In this case, Figure 5.19a shows the number of stars tracked by using a $3 \times 3$ search grid, whilst Figure 5.19b shows the number of stars tracked by using the larger $5 \times 5$ search grid.

Overall, both simulations showed good results in terms of the number of stars tracked, with the tracking mode never dropping below seven stars. During both simulations, the number of stars tracked were less than those in the FOV, with the $3 \times 3$ grid showing worse performance than that of the $5 \times 5$ grid.

During both simulations, identification performance during the first few seconds showed worse results than expected, with a large spike in detection at about 57 seconds. This discontinuity is repeated often throughout the dataset, however, and is part of the algorithm disadvantages. This




**(a)** Lookup Table Identification Results for $3 \times 3$ Grid




**(b)** Lookup Table Identification Results for $5 \times 5$ Grid

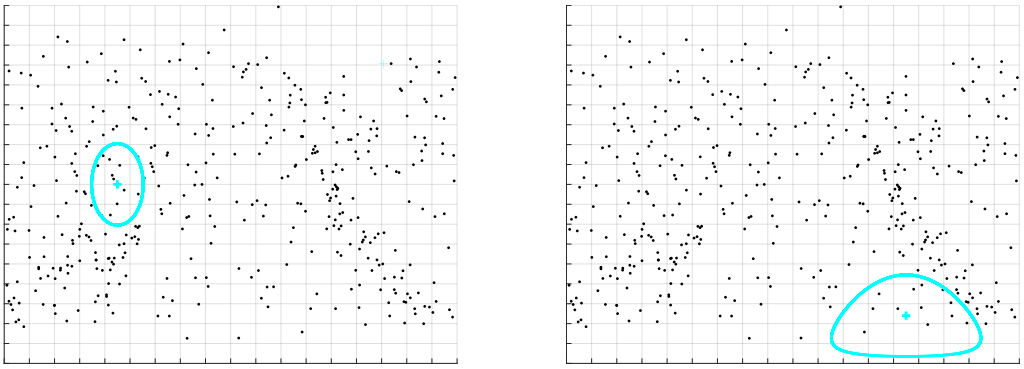
**Figure 5.19:** Number of Stars Tracked During Simulated Conditions

happens at points where the estimated BSV crosses from one bin to another, as new searchable identities suddenly become available.

As shown in both these figures, the discontinuity is worse at the poles, as the BSV is initialised at

$$\mathbf{v}_{bsv}^{\mathcal{I}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

This BSV corresponds with the north celestial pole. The reason for the delay in detection at the pole is due to the effect of the projective distortion of a circular FOV on a sphere. Close to the poles, each of the bins consists of a smaller area. The FOV projected thereon will therefore encroach on more bins. Using only a simple symmetrical search grid throughout the celestial sphere will therefore decrease detection performance at certain points on the celestial sphere. This effect is illustrated in Figure 5.20



**(a)** Projected FOV Close to Equator **(b)** Projected FOV close to Pole

**Figure 5.20:** Effect of FOV Distortion

As can be seen here, when the boresight is located closer to the celestial pole, the stars included in the FOV are spread over more bins, subsequently leading to worse performance. Although this might impact overall sensor performance, star identification still shows good results with at least five stars detected during each of the simulations.

### 5.2.5 Hardware-In-Loop Testing

Although the simulation results promise good state estimation, there are no real guarantees that a small, low-power application can effectively implement the estimation algorithms at an update rate of 1 Hz. All algorithms were therefore implemented in the C programming language and uploaded to the hardware to serve as a hardware-in-loop simulation.

In this case, however, as software ran on the microcontroller, a UART interface was also developed to upload input measurements, control system states, and download measurement results. This interface not only assisted in algorithm verification, but also enabled algorithm execution timing measurement capabilities.

The developed environment consisted of the three main parts: i) input data generation or preprocessing, ii) data upload, and iii) data request and storage. This process is shown in the diagram in Figure 5.21.

As the image upload process took at least 30 seconds, the hardware-in-loop testing was not done during real time. The results, as obtained during this simulation, are given in the next section.

#### Algorithm Execution Comparison

During hardware-in-loop tests, the main algorithms that were tested included the LIS versus tracking algorithm execution speed, and the execution speed of the EKF versus that of the QUEST algorithm. Figure 5.22 shows the timing results of the two search and matching algorithms.

As shown here, the reduced region of interest search had a substantial impact on star search and tracking performance when compared to the LIS algorithm. The total search and matching performance was at least two orders of magnitude faster than that of the LIS mode, with the longest search and matching during LIS mode taking 197 ms. On the other hand, the maximum execution time of search and matching during tracking mode was found to be only 7 ms.

This clearly shows the performance increase a region of interest tracking mode can deliver. Of the time required to execute the LIS mode, around 25 % was due to the geometric voting algorithm.

The total execution time of the attitude estimation algorithms was also measured and is shown in Figure 5.23.

Clearly in this case, it can be seen that the EKF requires much more computational effort than QUEST. The difference in execution time can be attributed to the costly matrix inversions required by the EKF. Overall, however, owing to the relatively low execution requirement of both

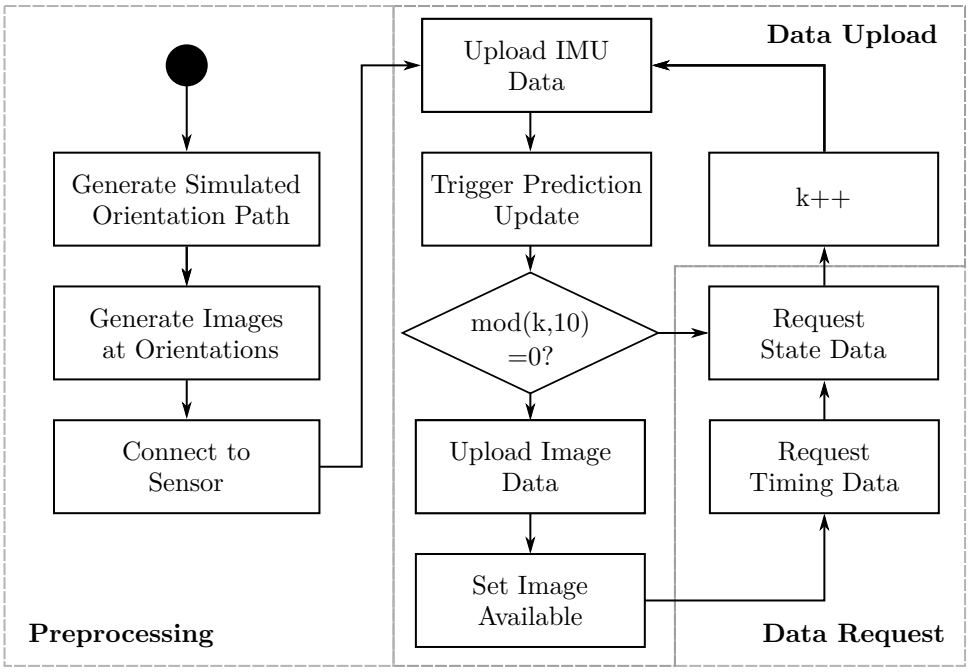



**Figure 5.21:** Hardware-In-Loop Testing Procedure

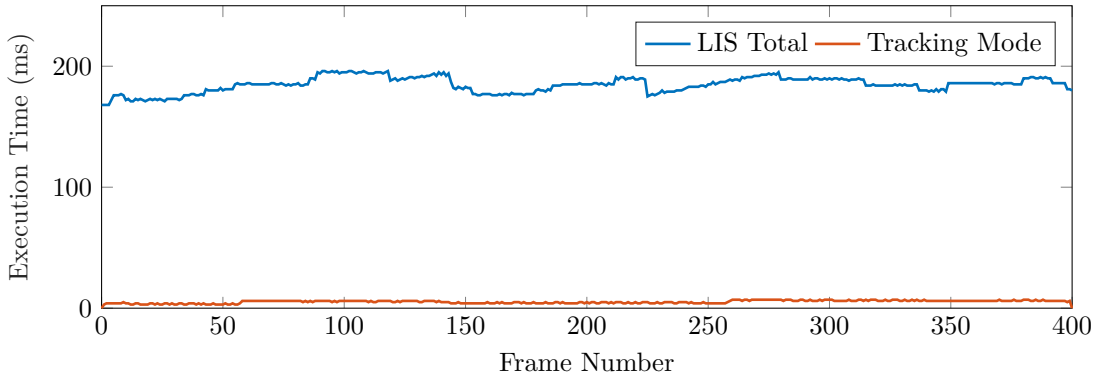



**Figure 5.22:** Star Identification Timing Comparison

algorithms, both can be used during nominal sensor operation without causing a reduction in overall sensor update rate.

## 5.3 Chapter Summary

During this chapter, the initial algorithm implementation was conducted, thereby proving the validity of the previously identified algorithms, thereby achieving objective three of this work. This chapter started with the investigation and verification of the Kalman filter behaviour. Firstly, the system model was developed. This simulated system consisted of the quaternion kinematics, as well as sensor models for both the IMU and star tracker. Although the modelled IMU showed a slightly higher bias drift than expected from initial sensor verification results, it still proved to be a useful method of verifying the Kalman filter under high bias drift conditions.

During the Kalman filter simulations, it was noted that the initial system bias overshoot and numerical algorithm stability is strongly dependent on the choice of initial system states. To compensate for this, it was concluded that it will be necessary to initialise the system states with an accurate attitude estimate obtained from an algorithm such as QUEST. This showed a significant improvement in overall filter response, with the filter immediately tracking the estimated

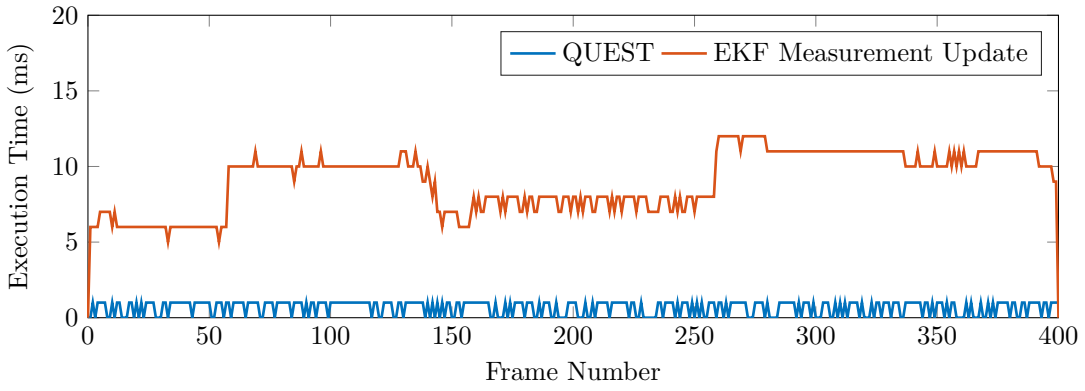



**Figure 5.23:** Orientation Estimation Timing Comparison

bias correctly.

A more extensive system model was then developed, based on image generation. Not only did this model allow for the implementation of the tracking mode developed in Section 4.5.1, but it also allowed a means of testing the various software interfaces. Although this simulation did not take a precise sensor noise model or camera distortion into account, it provides good insights into system performance, and could successfully function during simulated dynamic conditions. During the full system simulation, the effect of tracking search grid size on the number of identified and tracked stars were also investigated. For the two grid sizes investigated, the system showed successful functioning, with the $5 \times 5$ grid being able to identify and track more stars than the $3 \times 3$ grid. Because the overall performance was not hindered by a smaller grid, it was decided that the $3 \times 3$ grid would be sufficient for use during online operation.

As the system was deemed to function correctly, the algorithms were then implemented in the C language and deployed on the STM32L476 MCU. Hardware-in-loop tests were then conducted to analyse the timing performance of the LIS and tracking modes. It was concluded that the tracking mode had a far superior performance when compared to the LIS mode, showing an improvement of around two orders of magnitude in execution time, clearly illustrating the necessity of implementing a reduced ROI search during tracking. During the hardware-in-loop simulations, it was further determined that the EKF measurement update speed is strongly dependent on the number of stars tracked, and is much more computationally expensive than attitude determination through use of only QUEST. The overall execution speed did not jeopardise the requirement of a 1 Hz update rate, however, and should function successfully during actual practical tests, as considered in the next chapter.

# Chapter 6

# Practical Testing and System Integration

Although the simulated system showed initial proof-of-concept system integration results, algorithms have to be tested online, sensors have to be calibrated, and online system integration has to be completed, as specified in the fourth main objective.

Notwithstanding the inherent functional uncertainties in sensor design, star tracker-based systems validation is further heavily impacted by atmospheric conditions such as light pollution, visibility of celestial bodies, and weather. Although work-arounds can be implemented so that these factors are compensated for, the overall time required so that a favourable window for system testing is available may be in the order of weeks to months. Subsequently, to decrease the dependence of system verification on uncontrollable atmospheric conditions, a form of night sky emulation was required to verify final system integration is prior to actual night sky testing.

This chapter therefore presents a simple-to-implement, low-cost alternative to night sky testing, specifically for use during initial star tracker-based dead-reckoning system verification. This chapter begins with the initial design, implementation, and calibration of the Star Tracker Evaluation Environment (STEVE), after which some initial results and conclusions are given. Finally, in preparation for final system testing, subsystem calibration and integration, as required during actual night-time testing, are discussed.

## 6.1 Star Tracker Evaluation Environment

In the case of the attitude estimation system described in this work, an attitude relative to some inertial frame is important. The design of systems similar to that by Calitz [31] would therefore not be sufficient, as the author's experiments required only matched body vector pairs, therefore removing the necessity of inertial knowledge.

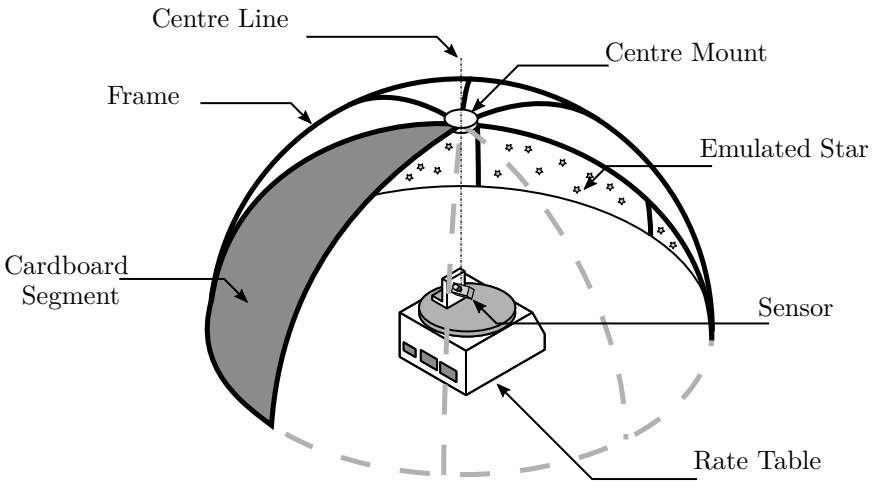



**Figure 6.1:** Star Tracker Evaluation Environment

Considering these observations, the following specifications with regards to the STEVE system were chosen: i) low cost, ii) easy to assemble, iii) simple to map, and iv) functional.

Because it is important that features observed in the emulation environment are static relative to one another, it is crucial that the depth of field of the environment is uniform. This can be achieved by either ensuring that features are so far away from the observer that any relative error becomes negligible, or that the distance to each feature is uniform. As this project was on a strict size budget of three meters, and an environment was required where a full 360° degree rotation would be favoured, it was decided to design the emulation environment such that it was in the shape of a dome. An artistic representation of the initial emulation environment design is shown in Figure 6.1.

Shown in this figure is the dome, as constructed from a frame structure covered with cardboard segments. The sensor was mounted to a rate table in the centre of the structure, such that system could be excited with an input and a response measured. To ensure the low-cost requirement was met, the frame was constructed from sixteen lengths of polyvinyl chloride (PVC) piping, covered with eight cardboard segments. On the inside of the dome, the structure was painted matte black to emulate a pitch-black sky. The dome radius was chosen as 1.5 m as it was constrained by the containing room.

Once the overall structure was completed, LED assemblies, as shown in Figure 6.2, were developed to serve as light points representing stars.

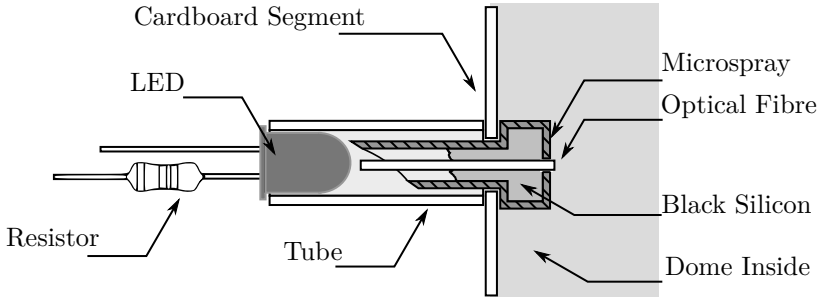



**Figure 6.2:** Emulated STEVE Star

The optical fibre was encased in a black microspray with a hole drilled at the centre. To ensure the fibre is held in place and that no light leaks through, black silicon was inserted around in the microspray. These stars were mounted in the dome by punching small holes in the cardboard and inserting the fibre assembly from the inside. Afterwards, to ensure stability, a small piece of tubing was placed over the spray backside with the LEDs inserted into the tube from the back. All LEDs were then wired to a communal power source so that star brightness could be varied uniformly.

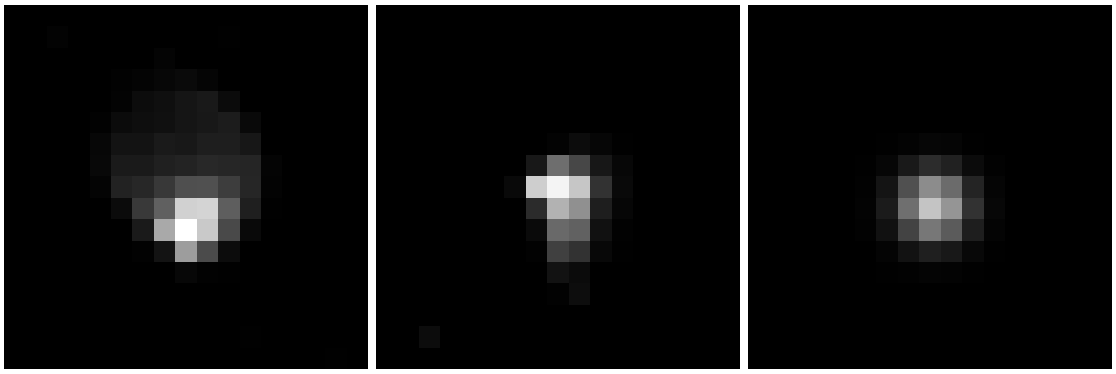
### 6.1.1 Star Comparison

After construction, images were captured from inside the STEVE to compare the emulated stars with real stars. This comparison is shown in Figure 6.3, along with that of a simulated star as reference.

From this figure it can be seen that the real and emulated stars show good agreement with only slight variations between the shown examples. The real star, however shows a slightly larger spread than that of the emulated star. This is expected, however, as the real star light has to pass through the atmosphere, causing some diffraction. The emulated stars were, however, sufficient.

### 6.1.2 STEVE Mapping and Catalogue Generation

As the STEVE star locations were randomly chosen, no knowledge of inertially referenced star positions, as used in attitude determination, were available. It was therefore imperative to map the emulation environment, so that a suitable star list and catalogue could be generated.



**(a)** Real Star **(b)** Emulated Star **(c)** Simulated Star

**Figure 6.3:** Star Comparison

Although physically measuring the angular distance between the STEVE origin and each star seemed to be the obvious method, it was deemed as being cumbersome and inaccurate with regards to true star mapping. An alternative mapping method was therefore identified.

In this method, the designed star tracker hardware would be employed so that relative star vector positions could be determined at known orientations. The mapping process would therefore consist of taking measurements and determining the inertial positions by use of the true attitude. Although near-perfect inertial positioning was an initial prerequisite, it was accepted that the usage of the dome would be more for functional testing of embedded software algorithms owing to errors in the dome design process. During testing, star detection accuracy margins could therefore simply be adjusted to allow matching in the emulated environment.

The identified mapping process was therefore seen to be the most optimal mapping method and consisted of the following steps:

1. Calibrate the star tracker hardware for the working distance;
2. Capture star images at known orientations;
3. Determine body-referenced star centroid measurements;
4. Transform the body-referenced star vectors back to the inertial frame by using the known orientation; and
5. Determine a suitable star catalogue that can be used for matching and attitude estimation

The mapping problem therefore became a much simpler attitude measurement problem. Before the catalogue could be determined, camera calibration was required.

### Sensor Calibration

Because the working distance in the STEVE is much shorter than that in the night sky, a different calibration than that used by CubeSpace had to be employed. Instead of the radial model, the Brown distortion model was therefore implemented as MATLAB's camera calibration toolbox provided a simple calibration interface.

The calibration process involved two main steps: lens focusing, and distortion modelling. Firstly, the camera was focused at 1.5 m by continuously capturing images of a chequerboard and manually adjusting the focus until a clear distinction between small features could be observed. After the lens was focused, a set of 77 images was then taken of a chequerboard with blocks of 100 mm in size. After each subsequent image was saved, the chequerboard was moved slightly so that the FOV was entirely covered. The results were then postprocessed by use of MATLAB's calibration toolbox. During this calibration, 60 of the 77 images showed complete visibility of

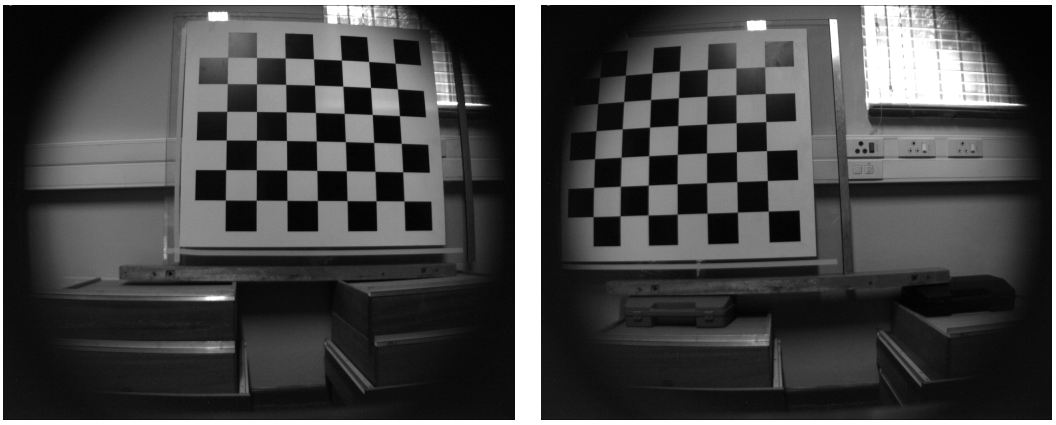
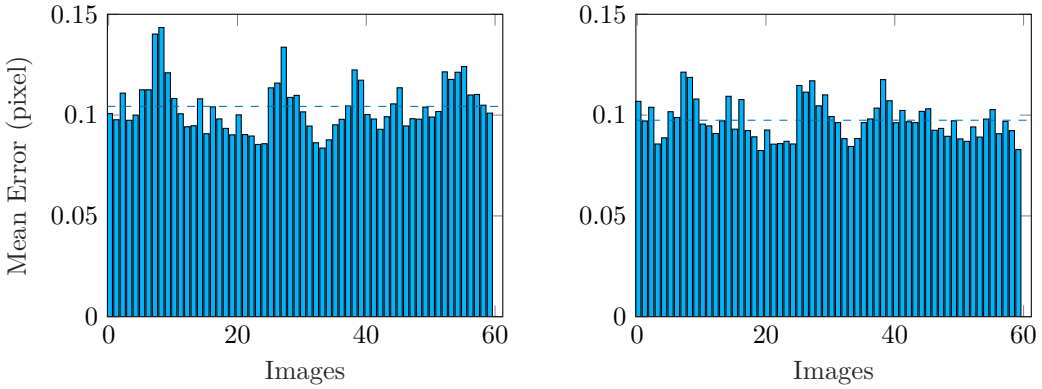


**Figure 6.4:** Dome Calibration Captures

the chequerboard and were used during final camera parameter identifications. A subset of the calibration images is shown in Figure 6.4

As the toolbox offered a simple way of investigating the effect of the different distortion types, as described in Section 4.3.1, the effect of compensating for only radial distortion (two-parameter calibration) and that of compensating for both radial and tangential distortion (four-parameter calibration) could further be analysed.

By use of the calibration toolbox, a suitable distortion model was then determined, and the reprojection errors for each image used were calculated. The reprojection errors for the two- and four-parameter calibration are shown in Figure 6.5.




**(a)** Two-Parameter Calibration **(b)** Four-Parameter Calibration

**Figure 6.5:** Pixel Reprojection Errors

From this figure, it can be noted that the overall peaks in image reprojection error were slightly lower in the case of the four-parameter calibration than that of the two-parameter calibration. The overall performance was very similar, however, as it was found that the use of a two-parameter radial model lead to a reprojection error of 0.1043 pixels, whilst using a four-parameter calibration that included tangential distortion lead to a slightly decreased mean projection error of 0.0975 pixels. Although the two-parameter calibration showed slightly higher errors, it was found that radial distortion dominated the overall lens performance. As the accuracy increase was close to negligible, it was concluded that compensating for only radial distortion was sufficient for accurate attitude estimates.

From the two parameter calibration sessions, the following distortion parameters were estimated:

$$K_1 = -0.317, \qquad K_2 = 0.170, \qquad P_1 = 0, \qquad P_2 = 0$$

with the camera intrinsic parameters determined as:

$$f = 6.242\,\text{mm} \qquad u_o = 638.11\,\text{pixels} \qquad v_o = 497.32\,\text{pixels}$$

After distortion calibration was completed, the actual mapping process involving the determination of inertial star positions could then be investigated.

### Mapping Hardware Setup

Ideally, hardware offering complete three-axis attitude setpoints should have been used, as perfectly known roll, pitch, and yaw angles could then be induced. Nevertheless, a simplified method by use of known azimuth and elevation angle was determined to provide enough degrees of freedom for elementary mapping.

To enable these required two degrees of freedom, a mapping hardware setup based on an Ideal Aersomith 1270Vs rate table, as well as a single-axis rotary stage was used. In this setup the rate table allowed for single-axis azimuth measurements up to resolution of 0.001° relative to some starting position. The elevation angle was manually controlled with an Edmund Optics 60 mm manual single-axis rotary stage. Unfortunately, this rotary stage only offered an accuracy of up to 1°. The final hardware setup, as used during the mapping process, is shown in Figure 6.6.

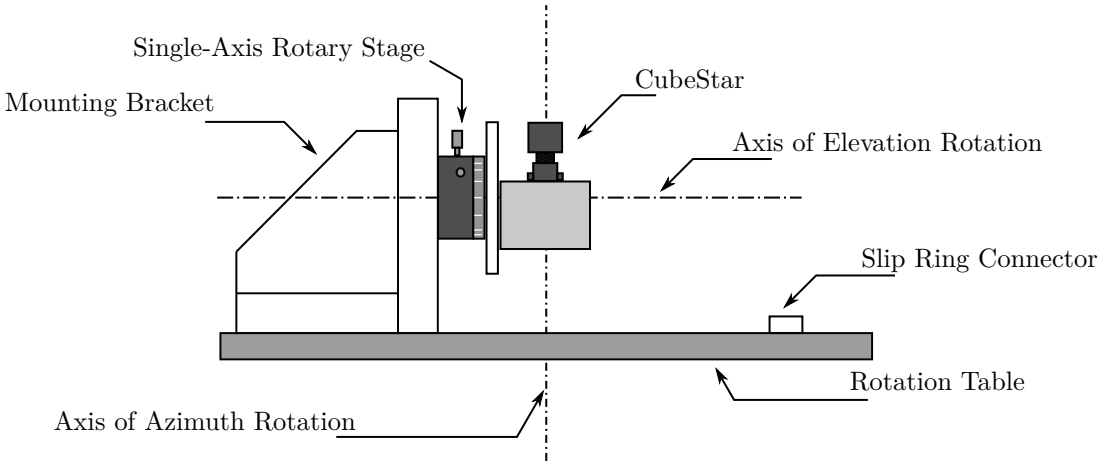



**Figure 6.6:** STEVE Mapping Hardware Setup

As shown in this figure, the star tracker was mounted such that the image plane centre coincided with the axes of rotation. To ensure the setup could rotate freely, the main MCU UART connector was connected to the Aerosmith slip rings.

This CubeStar rotary stage assembly was then mounted within the star emulation dome such that the image plane centre was approximately at the dome centre. After assembly, the rotation stage was levelled to ensure rotations around the true azimuth and elevation axes were decoupled.

### Data Acquisition

During the data acquisition stage of the mapping process, the elevation angle was fixed, and photos were taken at roughly 10° azimuth increments. In this case, the table home position was chosen as the 0° azimuth angle. Once a photo was captured, the rate table was set to rotate for ten degrees, after which the previous image was downloaded and saved. Figure 6.7 shows an inverted image captured during the mapping process, at an elevation angle of 90°. Here, identified stars are highlighted by blue circles.

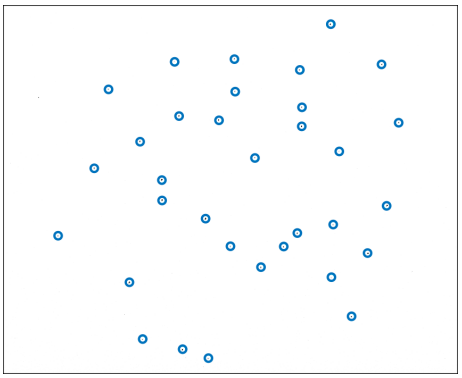


**Figure 6.7:** Capture at 0.000° Azimuth Angle

### Offline Image Processing

Once the data acquisition phase was completed, all results could be processed offline. During offline processing, the sensor attitude at each captured image was reconstructed by use of two Euler angles; one representing a rotation around the inertial $z$-axis, as measured by the rate table, and another signifying a rotation around the inertial $y$-axis, as manually set up by use of the single-axis rotary stage. For each image, star centroids were then detected, undistorted, and the measured vectors were transformed to the body-referenced system. As the attitude was also known, the estimated inertial position could then be determined.

However, owing to errors in the physical setup and irregularities of dome radius, inertial star positions for the data sets did not coincide perfectly. This effect was amplified at small elevation angles, when the sensor boresight was closer to the dome base, where the dome radius became more irregular. Only the data acquired at the 90° elevation angle could therefore be used.

At this 90° elevation angle, the inertial vectors determined for 36 images, led to small vector clusters, as shown in Figure 6.8. Owing to measurement resolution in elevation and minor setup errors, this behaviour was not unexpected.

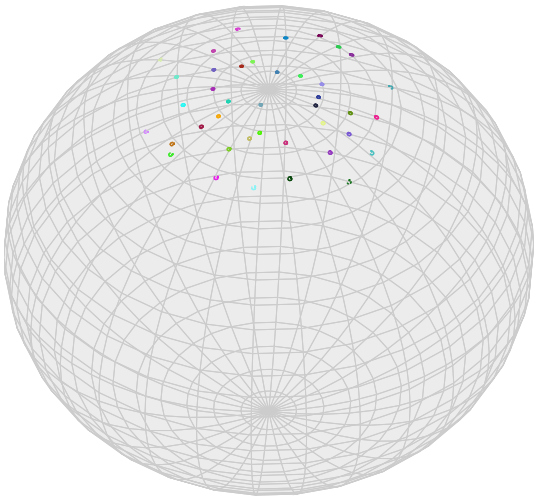


**Figure 6.8:** Observed STEVE Star Positions

For each of the clusters shown in this figure, the best estimate inertial position was then determined by use of singular value decomposition [94]. These inertial positions were then stored and used as a star list as described in Section 4.4, after which a star catalogue as required by

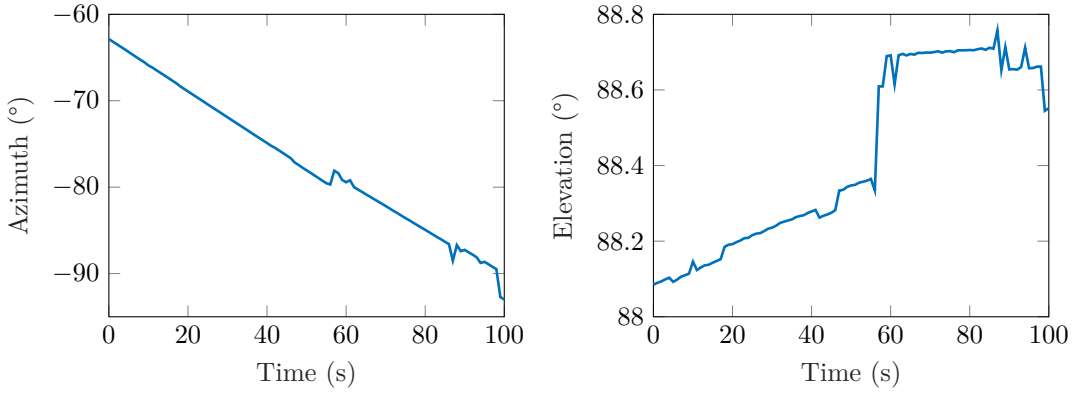
the geometric voting algorithm could be generated. The final star list contained 40 stars, with a catalogue size of 780 entries.

## 6.2 Initial System Feasibility Testing

Once the STEVE catalogue was generated, enough information was available so that online algorithm testing could be investigated. The initial algorithms could therefore be tested on the new hardware platform, not only to analyse the overall system feasibility, but also investigated the algorithm interaction and initial system integration. The first STEVE tests were therefore devised to test the functioning of each of the separate sensor technologies.

The sensor was once again set up in the star mapping position with the azimuth angle set to the home position and the elevation angle close to the STEVE north pole. The hardware was then initialised, and data were requested every second.

The data shown in Figure 6.9 were collected over a 100 s period. During this period, all processing was completed on the CubeStar hardware, with attitude and rate data requested via a MATLAB UART link. Figure 6.9a shows the estimated sensor boresight azimuth angle, whilst Figure 6.9b and elevation angle.




**(a)** Measured Azimuth **(b)** Measured Elevation

**Figure 6.9:** Measured Boresight Position

These figures emphasize the shortcomings of the emulation environment. Here, the attitude estimation starts off quite stable during the first half of the data set, with small variations in the order of 0.1° observed. These observations stem from error with regards to the observed depth of field difference in stars, as well as errors in physical setup and the mapping process.

At around 58 s, a discontinuity of about 1.5 degrees in the azimuth angle can be seen. Although this can be caused by a variety of factors, owing to the magnitude of the error as well as the length thereof, it can only be caused by a false match. As absolute inertial vector locations are much less accurate than those of true stars, the geometric voting algorithm error margin, $\xi$, had to be adapted. Although this enabled the validation of detected dome stars so that attitude could be determined, it also increased the probability of false matches.

The attitude is shown to stabilise to the original attitude trend close to the end of the dataset. A possible method of decreasing the effect of false matching is by removing some of the stars from the emulation environment so that false matches are less likely to occur. Despite these factors, it was proven that the original CubeStar algorithms could function effectively in the dome environment, with much less attitude accuracy than during previously conducted night sky tests.

During the initial testing, the functioning of the rate estimation algorithm as developed by Calitz [31], as well as the integration of the IMU was further investigated. Both of these datasets, are shown in Figure 6.10.

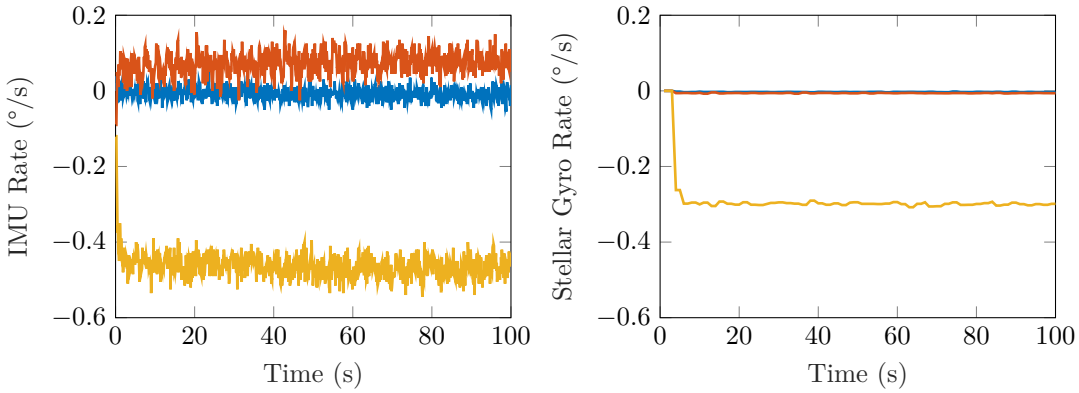



**Figure 6.10:** Measured IMU and Stellar Gyro Rate

From these figures it is shown that the rate estimation and gyro measurement worked as expected. Notable is, however, the effect of IMU bias and noise, standing in stark contrast to the stable measurements as estimated by the LSQ formulation. This clearly emphasizes the necessity of gyro augmentation during low angular rates.

Although the evaluation environment was deemed to be success, subsequent tests yielded results which were much more erroneous. The main reason was because of changes in the physical setup. Owing to the 1.5 m working distance, the catalogue had to be regenerated every time the emulation environment changed as small errors in setup were common.

Owing to the errors in inertial positions, it also proved to be much more difficult to test tracking mode algorithms, as high accuracy attitude knowledge was required to ensure successful star detection. The STEVE was therefore only used as a method of testing the initial hardware and software functionality, and enable embedded software debugging.

As proof-of-concept algorithm and functional hardware testing was completed, the final system integration could be performed.

## 6.3 Sensor Calibration

Although the STEVE offered good insight into the functioning and shortcomings of the developed algorithms, the hardware was still to be verified under actual night sky conditions, and tracking algorithms were still to be tested. Final system integration also involved the identification of characteristics of the sensor measurement noise and lens irregularities as it was either ignored or known during simulation procedures. Further, software time integration had to be performed to ensure processing deadlines are met. Hence, to integrate the software, hardware, and algorithms completely, the following steps had to be performed:

1. Lens distortion model for use under night sky conditions had to be determined;
2. IMU noise sources had to be characterised for use in the EKF; and
3. Software interfaces had to be integrated for correct system timing.

### 6.3.1 Camera Calibration

One of the specific issues raised by previous research [32] was that the chequerboard calibration method was not accurate enough in the determination of an adequate distortion model for calibrating sensors. This is mainly because the camera had to be focused at close to hyperfocal distance. It was therefore difficult to find a chequerboard of adequate size so that the lens could be correctly characterised.

With the development of CubeStar over the past few iterations, a new method of star camera calibration has been adopted. This calibration process works by capturing a multitude of star

images of known constellations. Some of these stars are then manually assigned their true inertial vector to enable initial distortion model determination. Once enough stars have been manually matched, the initial parameters are estimated. This process is repeated for a number of stars, until the the software's auto-match functionality can then be used to detect and identify all other images automatically. As this method utilizes the actual star positions during the calibration process, it is preferred above the chequerboard method as it leads to much higher accuracy during night sky conditions.

#### Calibration Process

To calibrate the optics, the star camera was manually focused under the night sky such that the faintest detectable stars were visible. Thereafter, a set of 200 star images of known constellations was captured and saved.

A subset of the captured images is given in Figure 6.11, with the identified star locations highlighted with red circles. As shown here, the constellations of Orion and the southern cross were used extensively during this calibration procedure due to their distinct shapes.

**(a)** The Constellation of Orion **(b)** The Southern Cross

**Figure 6.11:** Night Sky Calibration Captures

These camera calibration images were then fed into the calibration software. During night-time calibration, the following constants pertaining to the radial distortion model, as described in Section 4.3.3, were determined

$$K_1 = -0.0110036300, \qquad K_2 = -0.0000223042, \qquad K_3 = 0.0000002710, \qquad K_4 = 0$$

and the following intrinsic camera parameters where also found

$$f = 6.182\,\text{mm}, \qquad u_o = 635.70\,\text{pixels}, \qquad v_o = 522.3\,\text{pixels}$$

### 6.3.2 Redistortion Modelling

Searching for stars at estimated locations on an image plane further involves knowing how to determine a distorted star position, given the knowledge of the true image plane position. This process either requires a closed-form solution based on a reverse model or by using expensive error-reduction calculation methods. In this work, a much simpler approach was taken by use of the following approximation:

$$u_c \approx u'_c \tag{6.3.1}$$
$$v_c \approx v'_c \tag{6.3.2}$$

At small distortions these approximations will be valid, as the effect of the radial distance to the boresight will be close to negligible. The model mentioned in Equation (4.3.4) will then be

directly invertible such that

$$u'_c \approx u_c - u_c\epsilon_r(u_c, v_c) \tag{6.3.3}$$
$$v'_c \approx v_c - v_c\epsilon_r(u_c, v_c) \tag{6.3.4}$$

Although this assumption is valid close to the boresight, it will lead to increased errors when attempting to search for stars that are closer to the edges of the image plane. It is therefore imperative to analyse the effect of the redistortion error on the effective tracking FOV. To do this, a grid of points in the distorted axis system was generated, undistorted by use of the previously identified model, and then redistorted. A reduced version of these grids is shown in Figure 6.12.

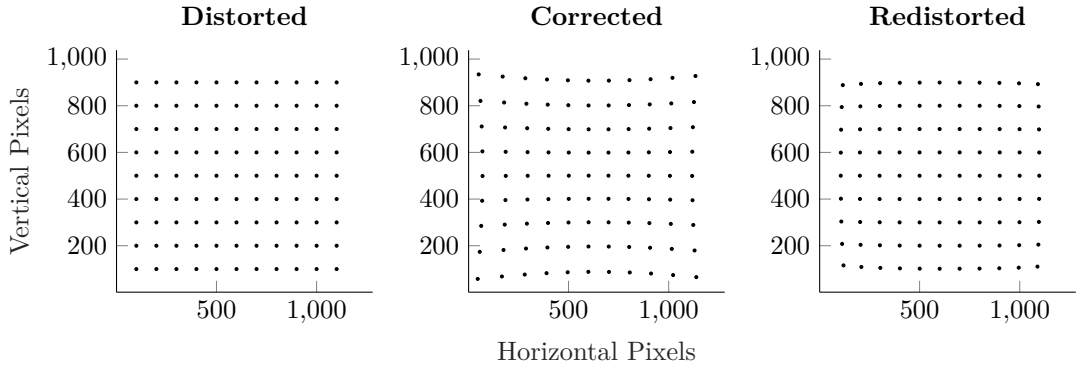



**Figure 6.12:** Visual Representation of Distorted Points

In the comparison of the distorted and redistorted versions of the image, it is evident that the redistortion model seems to provide sufficient accuracy across the largest part of the FOV. The four corners do, however, show the largest reprojection error. To analyse this, the total radial error between the true-distorted and estimated-distorted positions, as pixels, were then determined, after which the total radial distance from the boresight to the farthest detectable star for each case could be determined. The results of this are shown in Figure 6.13

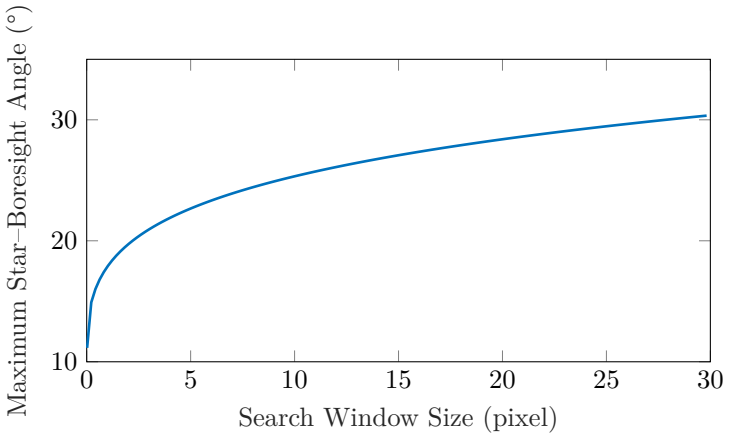



**Figure 6.13:** Maximum Detectable Distance to Star as Function of Search Window Size

The values shown in this figure are based only on the search position error induced by the redistortion model and does not include attitude prediction error. From this figure it can be deduced that the expected equivalent circular FOV will be smaller than 45° only if the search window width is smaller than 5 pixels. To compensate for possible prediction errors, it was decided to use a search window of around 8 pixels, leading to a subsequent effective tracking FOV in the

order of 49°. The FOV should therefore not be affected by the tracking mode, but rather limited by the hardware itself.

### 6.3.3 Gyro Characterisation

As seen in Section 4.7, the Kalman system covariance matrix, $\mathbf{Q}$, used in the covariance propagation is dependent on the statistical properties of the gyro rate noise, as well as that of the underlying process governing the change in bias over time. Although values for some of these parameters are quoted in the ADIS16460 datasheet [78], sensor variations exist. It is therefore imperative to characterise these processes to verify that the datasheet reflect the IMU characteristics and determine the covaraince used in the Kalman filter calculations.

The two suggested methods for use during gyro calibration are that of the spectral density [95], and that of Allan variance [96]. In this work, the Allan variance method is used as it is both well documented and regularly applied in the literature.

#### Overlapped variable $\tau$ estimator

The Allan variance is a statistical tool used to determine various noise sources present in data. This variance can be calculated by subdividing a dataset into varying cluster lengths corresponding to an emulated sampling period, $\tau$, where $\tau$ is an integer multiple of the sampling period at which the dataset was originally acquired, $\tau_0$:

$$\tau = n\tau_0 \tag{6.3.5}$$

In this case $n$ represents the data cluster size. For this calculation, the overlapped variable $\tau$ estimator was used as it promised superior performance. A graphical representation of the process used is shown in Figure 6.14, along with that of the nonoverlapped variable $\tau$ estimator.

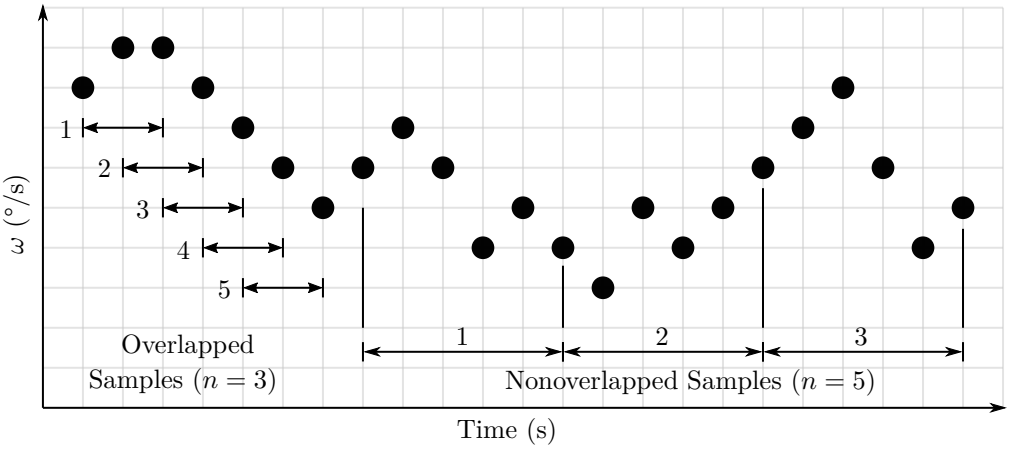



**Figure 6.14:** Visual Interpretation of Allan Variance Calculation

In this calculation, the times series, defined as

$$\theta(t) = \int_0^t \omega(t)dt \tag{6.3.6}$$

is used where $\theta(t)$ represents the angular displacement. From this time series, the overlapped Allan variance can then be calculated as [96]

$$\sigma_w^2(n\tau_0, N) = \frac{1}{2n^2\tau_0^2(N-2n)} \sum_{i=0}^{N-2n-1} (\theta_{i+2n} - 2\theta_{i+n} + \theta_i)^2 \tag{6.3.7}$$

In this equation, $N$, is defined as one more than the number of samples in the dataset.

The Allan variance signifies a general method of noise characterisation and can be used to determine quantization noise, ARW, Bias Instability (BI) and Rate Random Walk (RRW), among others. In MEMS gyro technologies, however, only the following noise sources were found to be dominant:

- Angle Random Walk – This can be determined by fitting a tangent line with a slope of $-\frac{1}{2}$ to the log-log Allan deviation plot on the high-frequency side. The ARW value, $Q$, can be read off on this line at $\tau = 1\,\text{s}$. This constant is then related to the gyro noise deviation by [96]

$$\sigma_{\eta_1}(\tau) = \frac{Q}{\sqrt{\tau}} \tag{6.3.8}$$

  where $\sigma_{\eta_1}(\tau)$ represents the deviation of the noise on the gyro measurements as a function of the gyro sample period, $\tau$. In this equation $\sigma_{\eta_1}(\tau)$ is measured in °/s.

- Bias instability – This value, denoted by $B$, can be determined from the lowest point on the Allan deviation plot as:

$$\sigma(\tau) = 0.664B \tag{6.3.9}$$

- Rate Random Walk – To determine this value, a tangent line with slope of $+\frac{1}{2}$ can be fitted to the log-log Allan deviation plot at the low-frequency side. The RRW value, $K$, can be read off on this line at $\tau = 3$ s [96]. This is then related to noise process governing the bias drift over time by [96].

$$\sigma_{\eta_2}(\tau) = \frac{K\sqrt{\tau}}{\sqrt{3}} \tag{6.3.10}$$

  where $\sigma_{\eta_2}(\tau)$ represents the deviation of the process governing the bias drift as a function of sample period, $\tau$. In this equation $\sigma_{\eta_2}(\tau)$ is measured in °/s$^2$.

A graphical example of the ARW, BI, and RRW determination is shown in Figure 6.15.

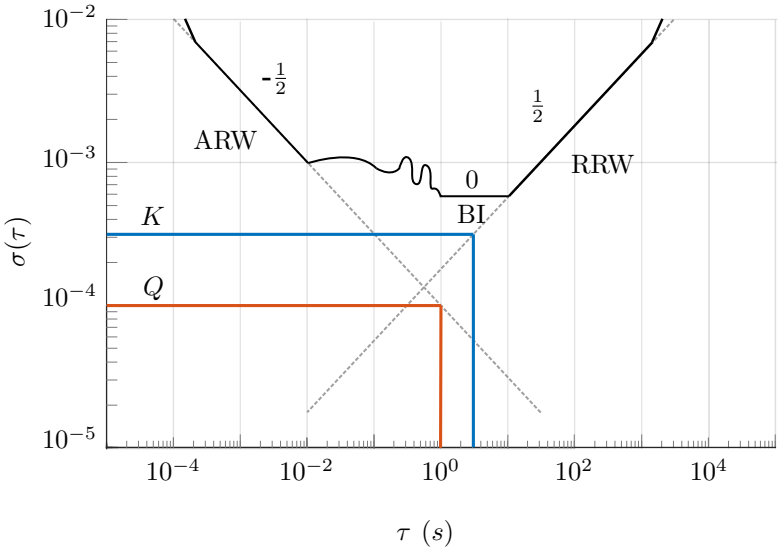


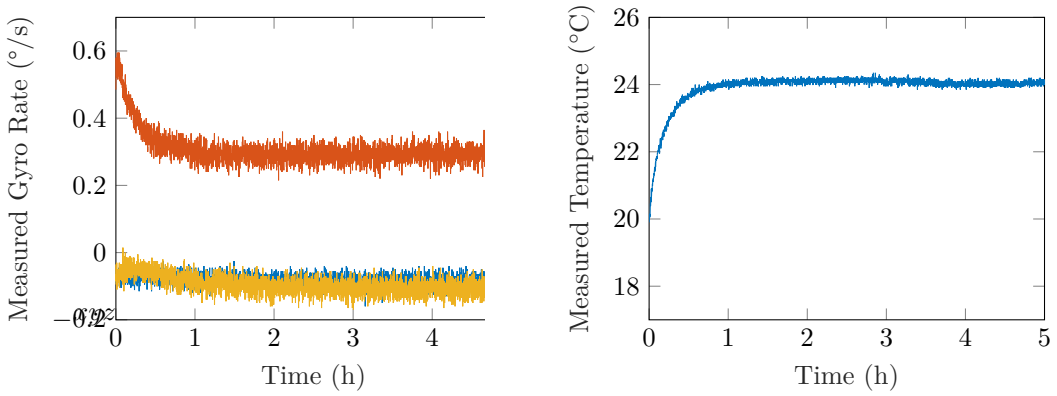

**Figure 6.15:** Graphical Representation of Allan Deviation Interpretation

### Experimental Setup and Methodolgy

To determine the gyro calibration factors, the hardware was set up so that the star tracker gyro assembly was left unperturbed by external vibrations. IMU data were then collected over the span of five hours.

### Characterisation Results

Figure 6.16a shows the IMU data as obtained during one of the experiments. Significant initial bias drift can be seen in this figure, with measurements finally settling to a relatively steady position after about two hours. According to the IEEE [97], this period is described as the gyro warm-up period, and should be discarded during the calculation of the Allan variance.




**(a)** Measured IMU Response **(b)** Measured IMU Temperature

**Figure 6.16:** Measured IMU Dataset

Five of these datasets were collected, so that the variation between datasets could also be analysed. A complete set of the calculated Allan variance for each axis is shown in Figure 6.17.

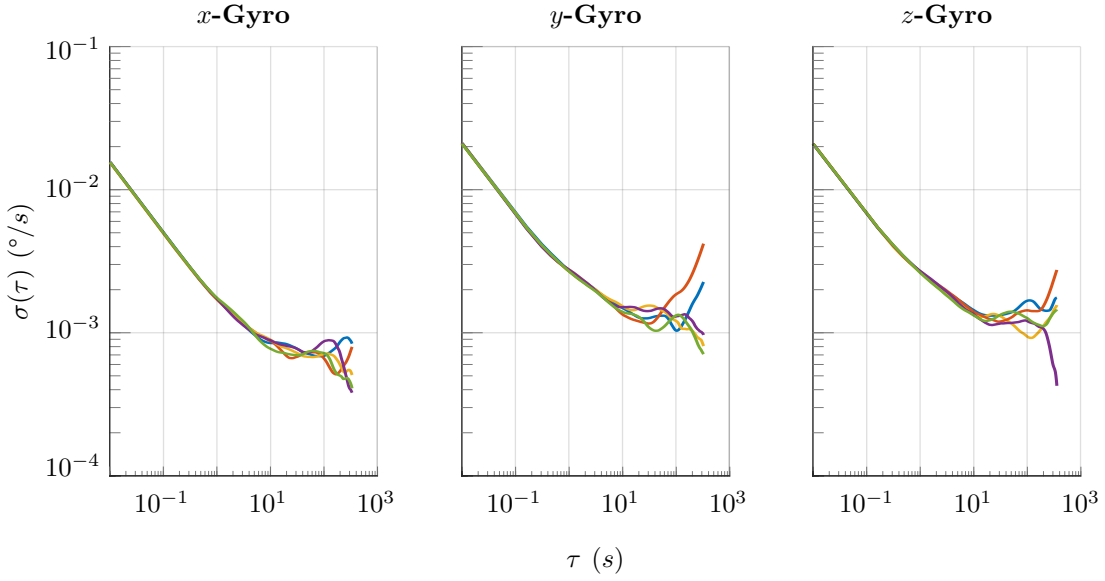



**Figure 6.17:** Allan Deviation of Three-Axis MEMS Gyro

These figures show the Allan variance for sample sizes ranging between 0.01 s and 327.68 s. From these data, it can be noted that the high-frequency part stayed consistent across all three axes, whilst the low-frequency data showed much higher variations with sample period increase. This can mainly be attributed to temperature fluctuations as well as population size used in the Allan variance calculation.

As some of the datasets were inconclusive, only those datasets that showed an Allan variance response consistent with literature were taken into account. Only two of the datasets were therefore

valid. To analyse the average parameters for these two datasets, the pooled standard deviation was used. This is described as:

$$\bar{\sigma}(\tau) = \sqrt{\frac{\sum_1^k \sigma_k(\tau)^2}{k}} \tag{6.3.11}$$

Where $k$ represents the number of sample sets, $\sigma$, the deviation for each sample set, and $\bar{\sigma}$ the pooled standard deviation. Important to note here is that this equation is valid only if each sample set contains the same number of observations.

The calculated pooled Allan deviation is shown in Figure 6.18. In these graphs, the yellow lines represent the BI, red lines ARW, and black lines RRW.

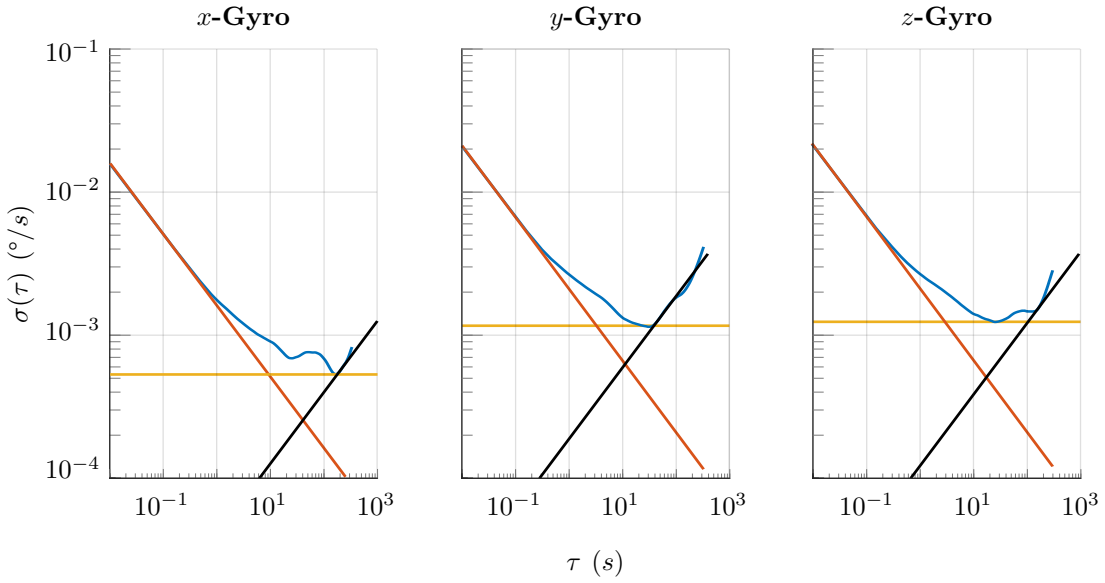



**Figure 6.18:** Pooled Allan Deviation of Three-Axis MEMS Gyro

From these plots the values for the ARW, RRW and BI were determined and are given in Table 6.1.

**Table 6.1:** IMU Characterisation Results

| | **x-axis** | **y-axis** | **z-axis** |
|---|---|---|---|
| Bias Instability (°/s) | 0.00075 | 0.0016 | 0.0018 |
| Rate Random Walk (°/s$^2$/$\sqrt{\text{s}}$) | $2.1\times10^{-4}$ | $1.0 \times 10^{-3}$ | $6.8 \times10^{-4}$ |
| Angle Random Walk (°/s/$\sqrt{\text{s}}$) | 0.0016 | 0.0021 | 0.0021 |

In this table, it is shown that the measured values correspond well to those of the datasheet provided by Analog Devices [78] who quotes the BI as 0.0022 °/s. In this case, although the values estimated in this work were slightly lower than what the datasheet reports, Analog Devices mentions that the quoted BI is a compound term also taking into account the RRW. Differences are therefore expected. In terms of ARW, the estimated values correspond well with that of the datasheet as these were specified as 0.0020 °/$\sqrt{\text{s}}$ around the $x$-axis and 0.0028 °/$\sqrt{\text{s}}$ around the $y$- and $z$-axes, and component-to-component variation is expected.

Although these values can be used to determine the sensor noise variances used in the Kalman filter, concerns with regards to the effect of unmodelled processes on the perceived sensor variances were noted. To ensure therefore that the Kalman filter will be robust against these unmodelled processes, the calculated gyro variances were only used to determine the minimum variances. From these minimum variances, an iterative approach was used to determine the final sensor variances

used in the Kalman filter. Although this method was found to be robust, it is suggested that a more in-depth characterisation and gyro modelling method be used in future work. Values used in the Kalman filter were determined as:

$$\boldsymbol{\sigma}^2_{\eta_1} = \begin{bmatrix} 0.276 \\ 0.377 \\ 0.381 \end{bmatrix} \text{m(rad/s)}^2, \qquad \boldsymbol{\sigma}^2_{\eta_2} = \begin{bmatrix} 3.231 \\ 3.531 \\ 3.531 \end{bmatrix} \mu\text{(rad/s}^2\text{)}^2$$

Given this information, the sensors were characterised sufficiently to enable final system integration.

## 6.4 Sensor Fusion

In this work, as the IMU was used to observe higher-frequency dynamics than offered by the star camera, a more intricate timing scheme had to be developed than what has been used previously. The main reason for this was because two different update-rate sensor measurements had to be fused. The IMU was set up to sample the sensor body-rates at a frequency of 10 Hz to allow for high-frequency orientation propagation, whilst the star camera provided 1 Hz measurements enabling attitude corrections. The key problem here was, however, that star camera measurements did not become available when they were required as the integration and image download to SRAM took around 700 ms. This problem is shown graphically in Figure 6.19.

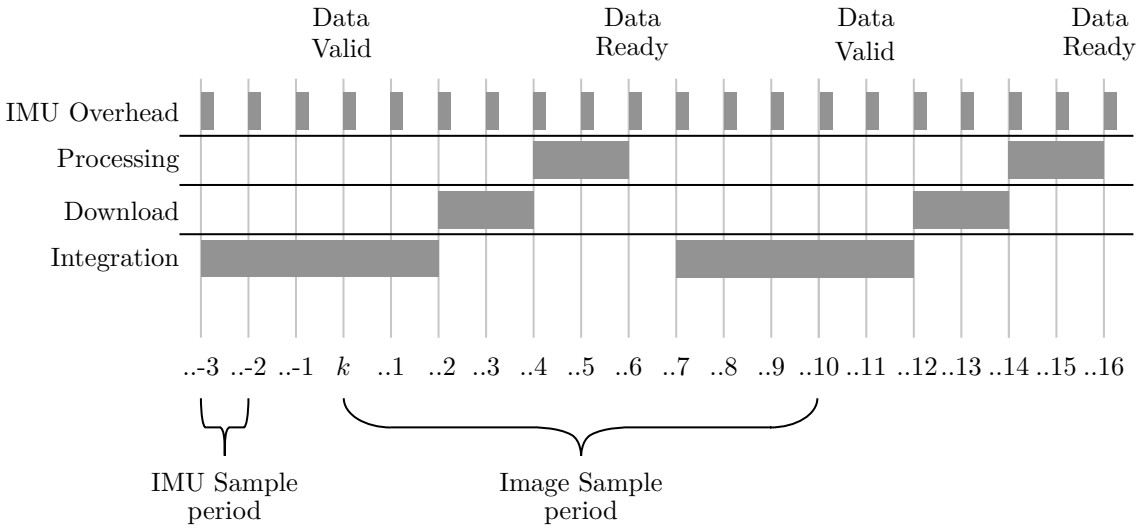



**Figure 6.19:** Processing Delay Problem

In this figure, each vertical line representes 100 ms, and $k$ represents the point where image data are supposed to be fused into the Kalman filter. The reason why this timestep is of importance, is because the exposure of the star image is approximated to happen instantaneously at this point in time, which falls approximately in the middle of the sensor exposure time. This simplification is necessary as it decreases the complexity of the data integration. Owing to the capture and download delay, vector data used in the measurement update process only become available at timestep $k+6$. As the EKF attitude is still propogated during timesteps $k$ to $k+6$, the Kalman filter would have advanced during this period. To incorporate these vector measurements successfully, the data therefore have to be incorporated retrospectively. To handle this situation, it was decided that when star tracker data become available at timestep $k+6$, the EKF state vector and covariance matrix be reset to that as determined at timestep $k$. Once the filter has been reset, the image is processed by use of the tracking mode. All vector measurements are then fused into the Kalman filter and the sensor attitude is repropogated to the current timestep by using timestamped and buffered gyro data. This ensures the EKF is always at the most recent attitude.

## 6.5 Chapter Summary

The purpose of this chapter was twofold. Firstly, an emulation environment was developed, and initial subsystem testing was completed, after which sensor calibration and characterisation was done. Finally, the system could be completely integrated, as required by the fourth project objective.

Although it was found that the low-cost star emulation environment was sufficient for initial algorithm testing, it proved to be a cumbersome, difficult-to-calibrate method of star emulation. The main problem with this evaluation environment was found to be the potential errors in star matching, owing to mapping and setup errors.

This chapter also treated the calibration process of both the star camera and the gyros. With regards to the star camera, the effect of tangential distortion was evaluated, and found to be close to negligible, as a reprojection accuracy increase of only 0.007 pixels could be seen in images for which tangential distortion was calibrated. The star camera was subsequently also calibrated for use during night sky tests, and the effect of the chosen redistortion model was investigated.

Further calibration pertaining to the gyros was also done. Five five-hour long datasets were collected and processed by use of the Allan variance method. The Allan variance method showed that the sensor was within manufacturers' specifications with regards to BI and ARW. It was however concluded that the gyro covariance modelling required a much more in-depth analyses as the values used in the Kalman filter had to be approximated experimentally.

Finally, the complete system integration, as used during subsequent night sky tests were explained. The final system integration highlights some timing and synchronisation issues noted during the full system implementation. To solve this, a sensor synchronisation scheme based on a EKF state reset was implemented. The sensor developement was therefore satisfactory, and final night sky tests could be completed.

# Chapter 7

# Results and Discussion

To validate the system capabilities during autonomous operation, two different testing situations were devised to achieve the final project objective. Firstly, sensor performance during near-stationary conditions was analysed. During these experiments the sensor was left earth fixed whilst exposed to night-time conditions. Although this proved sufficient for initial system validation, a more realistic test case was then implemented to test the system response during variable rate excitations.

## 7.1 Earth-Fixed Testing

Due to the nature of the ECI coordinate frame, the boresight position of an earth-fixed star tracker on the celestial sphere is expected to stay relatively constant in declination whilst showing an almost linear response in right ascension due to the earth's rotation rate. Although this nonstationary nature of the coordinate frame does make it difficult to measure the sensor output stability, it does allow the testing of bias and rate estimation algorithms at very low rotation rates.

During the earth-fixed tests, the developed sensor was left pointing at a populated patch of sky for about one hour. A full set of rate, bias, and orientation data was then requested by use of a serial interface that was connected to a PC. Ground support software was developed in MATLAB to enable serial communication, with a further custom power supply/serial converter assembly developed to enable power measurement and communication.

Further, to show experiment repeatability, the testing procedure was repeated on three separate occasions. Data were requested every second over a period of one hour. Communications were limited to one request per second as a higher request rate caused more frequent packet errors on the PC side of the setup.

An illustration of the hardware setup is shown in Figure 7.1

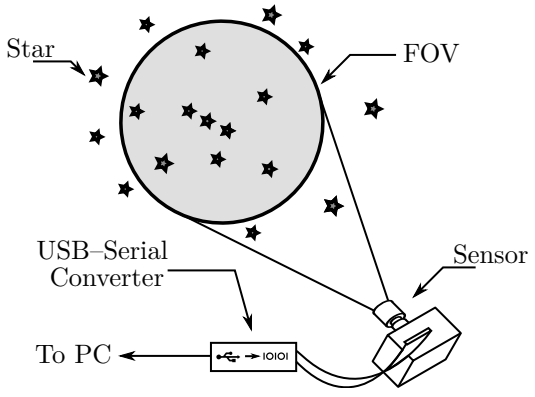



**Figure 7.1:** Earth-Fixed Testing Setup

During these experiments, the functioning of orientation, bias correction, and rate estimation, as well as the performance of the gyro-assisted tracking mode, was observed. The performance of the sensing algorithms in each case is treated in the following subsections.

### 7.1.1 Star Identification and Tracking

Figure 7.2 shows the number of stars tracked, as well as a rough estimate of the number of stars that should have been in the FOV. This estimated value is based on the postprocessed boresight vectors, as well the assumed FOV.

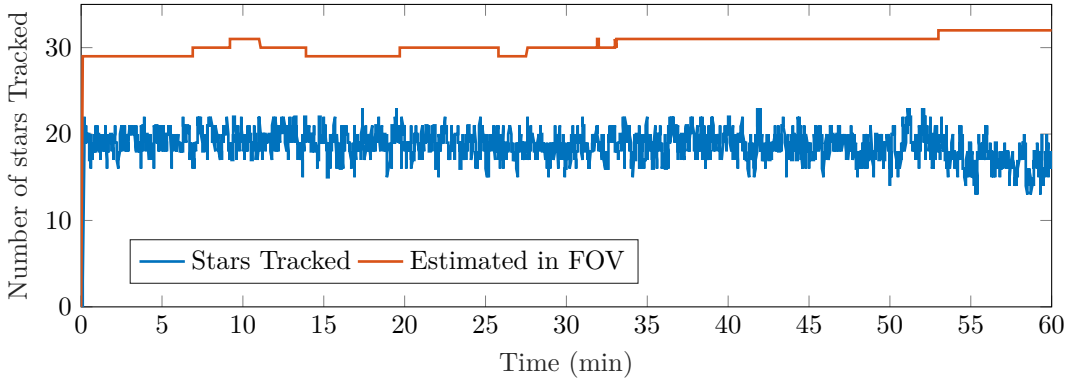



**Figure 7.2:** Number of Stars Tracked Over One Hour

Overall, the results shown are promising, with the tracking mode showing on average nineteen stars identified and tracked. In these data the number of stars tracked are shown to be less than those estimated to be in the FOV. This difference is attributed to the effect of diffraction in the atmosphere, as stars will appear dimmer than expected, this also causes the variations in the number of stars tracked. Overall, new stars could successfully be identified, tracked, and discarded once no longer in the FOV.

### 7.1.2 Orientation Estimation Results

In the case of the orientation data, only the inertially referenced quaternions were requested. All attitude quaternions were then postprocessed to determine the corresponding attitude matrices and therefore, by extent, the sensor boresight vectors. These vectors were then transformed to right ascension, declination, and roll angles. These orientation angle estimates are shown in Figure 7.3.

As expected, the lines themselves show definite trends, with the right ascension showing an almost linear increase over time, whilst the declination and roll angles show a slight increase lower than 0.5° over the course of an hour. These variations over long periods exist as the sensor is not perfectly alligned with the earth's rotation axis. Overall, however, this figure shows a good, consistent response. It can also clearly be seen that the round-boresight data are much coarser than that of the cross-boresight values, as these estimated data are not in the observable plane [16].

Although these results give a relatively good indication of relative noise, no intuitive conclusions can be made with regards to the actual measured accuracy of the designed system. To solve this problem, a linear regression, as proposed by Liebe et al. [16] is performed. This relative attitude drift caused by the earth's rotation is then subtracted. The remaining zero-mean data can then be analysed statistically to determine some intuitive measure of system accuracy. The determined distributions for this dataset are shown in Figure 7.4:

These data show unbiased, normally distributed errors for all three datasets. As expected, and as seen in previous research by Erlank [32], the error in declination shows the least variance, whilst the round-boresight error shows the greatest variation. For these datasets, a summary of the statistical properties, related to determined noise variance, is given in Table 7.1.

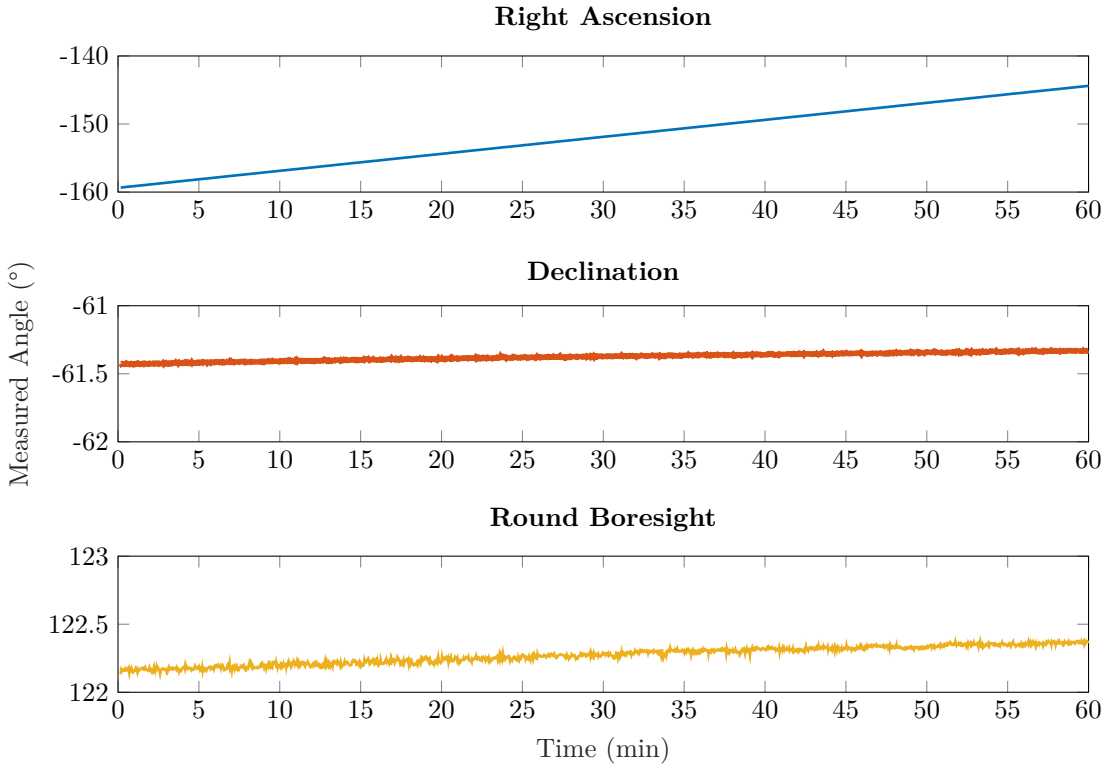



**Figure 7.3:** Estimated Boresight Location

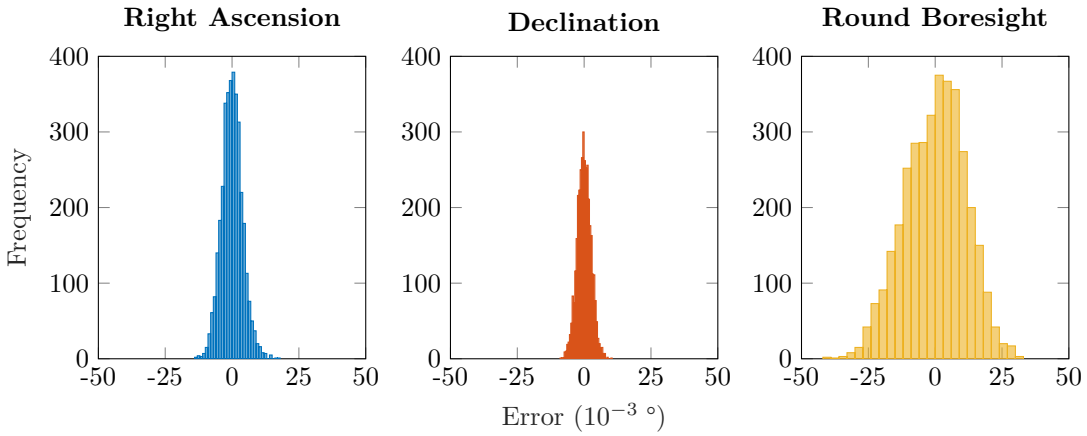



**Figure 7.4:** Cross-Axis Error Distribution

In conclusion of the orientation estimation results, the sensor showed good repeatability with major statistical differences attributed to inconsistency with regards to weather. Overall, the sensor delivered reliable autonomous operation for at least an hour at a time during constant, low rates.

### 7.1.3 Rate Estimation and Bias Correction

During earth-fixed testing, rate data, as measured by the gyro, compensated by the EKF, and estimated by the stellar gyro algorithm were also requested. These data are shown in Figure 7.5.

This figure shows the comparison between the uncompensated IMU, compensated IMU and estimated body rates. As can be seen from these data, bias rejection with regards to EKF filtering worked well, as there is considerably less error on the output rate data. In comparison with the estimated stellar rates, however, the IMU rates show a much coarser response, clearly indicating

**Table 7.1:** Static Sensor Measurement Results ($3\sigma$)

| | $1\sigma$ **Error** (°) | $3\sigma$ **Error** (°) |
|---|---|---|
| Right Ascension | 0.0039 | 0.0117 |
| Declination | 0.0025 | 0.0078 |
| Round Boresight | 0.0114 | 0.0342 |

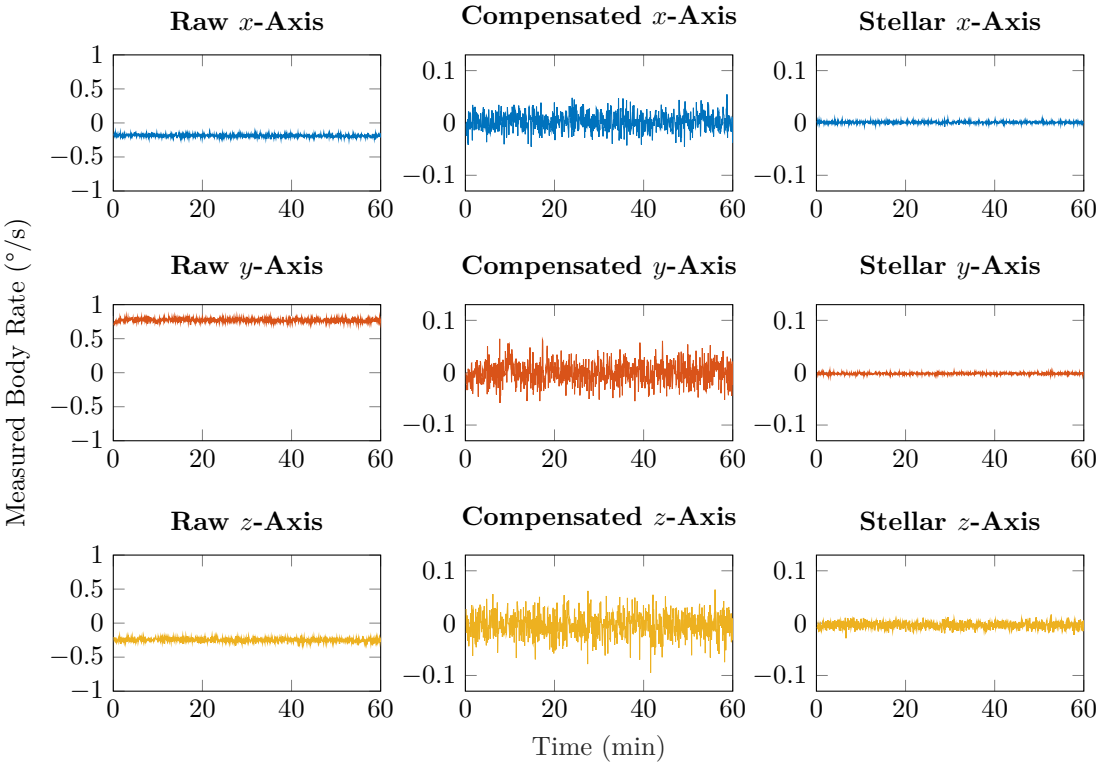



**Figure 7.5:** Measured Sensor Rates During Earth-Fixed Testing

the advantages of augmenting the system with the vector-calculated values at low rates.

For both the compensated and estimated stellar rates, the mean, and variance were then calculated during postprocessing. Owing to the rotation of the earth, it is expected to see the data distributed not around zero, but around some mean value representing the earth rate. As the axis of the sensor is not aligned with that of the earth's rotation, the rate is expected to be divided across the three sensor axes. The statistical characteristics of these measured rate signals are shown in the following table

**Table 7.2:** Earth-Fixed Sensor Rate-Measurement Mean

| | **Mean** (°/s) | | | $3\sigma$ **Deviation** (°/s) | | |
|---|---|---|---|---|---|---|
| | $x$ | $y$ | $z$ | $x$ | $y$ | $z$ |
| Raw Measurements | −0.187 | 0.770 | −0.248 | 0.050 | 0.077 | 0.068 |
| Bias Compensated | 0.002 | −0.001 | −0.004 | 0.049 | 0.086 | 0.067 |
| Stellar Gyro | 0.001 | −0.001 | −0.003 | 0.005 | 0.004 | 0.017 |

From this table, it is evident that the stellar gyro algorithm has a substantially less noisy response than that of the IMU. As shown here, the mean of the norms of the estimated rate vectors, is nonzero. This is expected, however, as the earth itself is not static relative to inertial space. Instead, it rotates at around 0.004 178 °/s (the equivalent of 360° every sidereal day). By

taking the norm of the mean of the stellar gyro data sets, the estimated rate of the earth's rotation was calculated as 0.004 184 °/s, showing an error of around 22 milli-arcsecond/s.

### 7.1.4 Experiment Repeatability

To ensure robust and successful functioning of the sensor, datasets were collected on three separate occasions. During these three tests, weather conditions were relatively similar. Although skies were clear during each test, differences in environmental conditions such as humidity and ambient light contributed to slight variations in measured accuracies. A summary of the most notable results is given in Table 7.3

**Table 7.3:** Static Sensor Measurement Results ($3\sigma$)

| | **Test 1** | **Test 2** | **Test 3** |
|---|---|---|---|
| **Orientation** (°) | | | |
| Right Ascension | 0.0092 | 0.0158 | 0.0117 |
| Declination | 0.0106 | 0.0109 | 0.0078 |
| Round Boresight | 0.0546 | 0.0561 | 0.0342 |
| **Stellar Rates** (°/s) | | | |
| $x$-Axis | 0.0047 | 0.0059 | 0.0052 |
| $y$-Axis | 0.0035 | 0.0056 | 0.0040 |
| $z$-Axis | 0.0144 | 0.0277 | 0.0173 |
| **IMU Rate** (°/s) | | | |
| $x$-Axis | 0.0491 | 0.0494 | 0.0498 |
| $y$-Axis | 0.0849 | 0.0796 | 0.0862 |
| $z$-Axis | 0.0663 | 0.0667 | 0.0670 |

From these results, estimated sensor errors show good agreement. It can therefore be concluded that this sensor functions reliably during static low-rate conditions.

### 7.1.5 Discussion

During the practical system testing, it was determined that the sensor showed similar responses to that of the simulated measurements shown in Chapter 5. As expected, the attitude and bias Kalman filter responds similarly to previous results. During this test, it was, however, determined that the sensor shows much less gyro drift than the modelled system showed. This is, however, expected, as the noise profile of the modelled and actual sensors did not correspond perfectly. The estimated and compensated rates further also responded quite similarly to the simulated data, although the measurements from the simulated system are slightly less noisy. Overall, however, the integrated system worked correctly, with all subsystems functioning as expected during online algorithm execution.

## 7.2 Dynamic Testing

Satellite orientations are rarely completely stationary but tend to experience some form of dynamic condition. To ensure correct functioning of the system during these dynamic conditions, proper testing should be performed. An experimental setup, such shown in Figure 7.6 was therefore devised.

Similar to tests in the STEVE, an Ideal Aerosmith 1270VS rate table was used. This rate table was set up under clear night sky conditions. The rate table was then controlled through the on-board serial interface to induce a sinusoidal slew rate with peak of 1 °/s and period of 400 s. The input waveform was updated every two seconds and is given by:

$$\boldsymbol{\omega} = \begin{bmatrix} 0 & 0 & \sin\left(\frac{k\pi\Delta t}{200}\right) \end{bmatrix} °/\text{s} \tag{7.2.1}$$

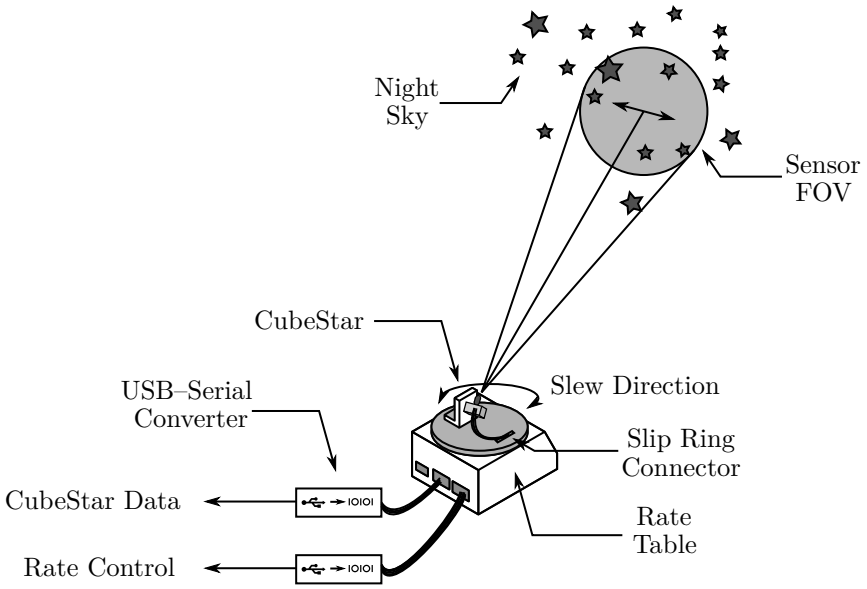



**Figure 7.6:** Slew Test Setup

The sensor was then set up at an angle of about 58° to ensure the excitation of multiple sensor axes. During these tests, the same sensor data as in the earth-fixed tests were requested. As before, orientation quaternions were postprocessed to determine the path of the boresight vector. Owing to the physical setup of the sensor, a maximum slew angle of 60° was allowed as any larger deviation would lead to the sensor FOV being obstructed. Results obtained during these tests follow.

### 7.2.1 Star Identification and Tracking

Firstly, the performance of the gyro-augmented tracking mode was analysed to determine its performance. During these tests, as in the previous case, the more computationally efficient $3 \times 3$ search grid was used. Figure 7.7 shows the number of stars tracked, along with the estimated number of stars in the FOV.

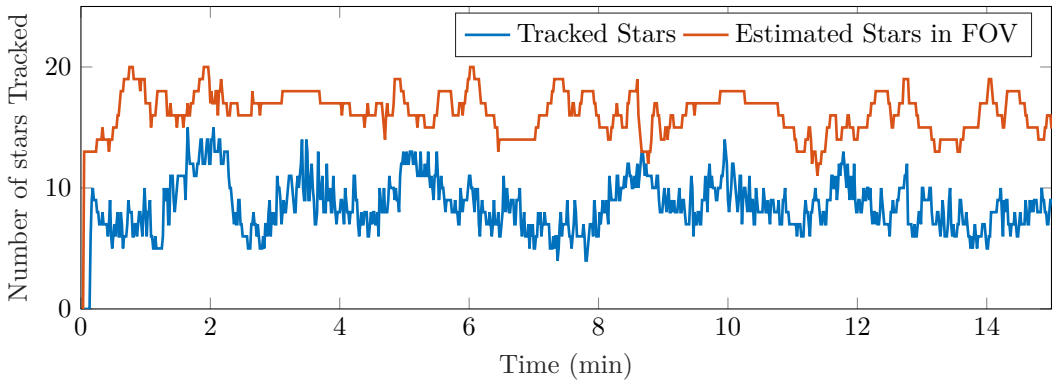



**Figure 7.7:** Stars Tracked During Dynamic Conditions

Overall, the algorithm performance was satisfactory with more than three stars detected even during rates as high as 1 °/s. In this plot, it is clearly shown that the number of stars estimated to be in the FOV are considerably more than those actually tracked. As in the earth-fixed test case, this is mainly attributed to the effect of the earth's atmosphere.

In this case, however, it should also be noted that the number of stars tracked is not necessarily equivalent to the number of stars detected in the current frame. This is mainly due to stars only being counted as tracked once they have been identified in more than one image. During these dynamic tests, star will also appear as low-intensity light streaks. As the total incident photons are then spread over a strip of pixels, the total star intensity will be much lower, causing stars in the FOV to fall under the noise threshold. Owing to the decrease in star centroiding accuracy with regards to star streaks, the tracking validation margin also decreases the total number of stars tracked.

### 7.2.2 Orientation Estimation

Figure 7.8 shows the orientation estimation results obtained during the dynamic system testing. Both the right ascension, declination, and roll angle behaved as expected. Determining the actual accuracy during dynamic testing proved to be much more difficult, however, as the orientation not only follows a severely nonlinear trend, but no ground truth measurement of sufficient accuracy was available. The fact that the sensor could function in tracking mode shows that enough accuracy was obtained, however, as the estimated star locations were completely dependent on the previous attitude estimate.

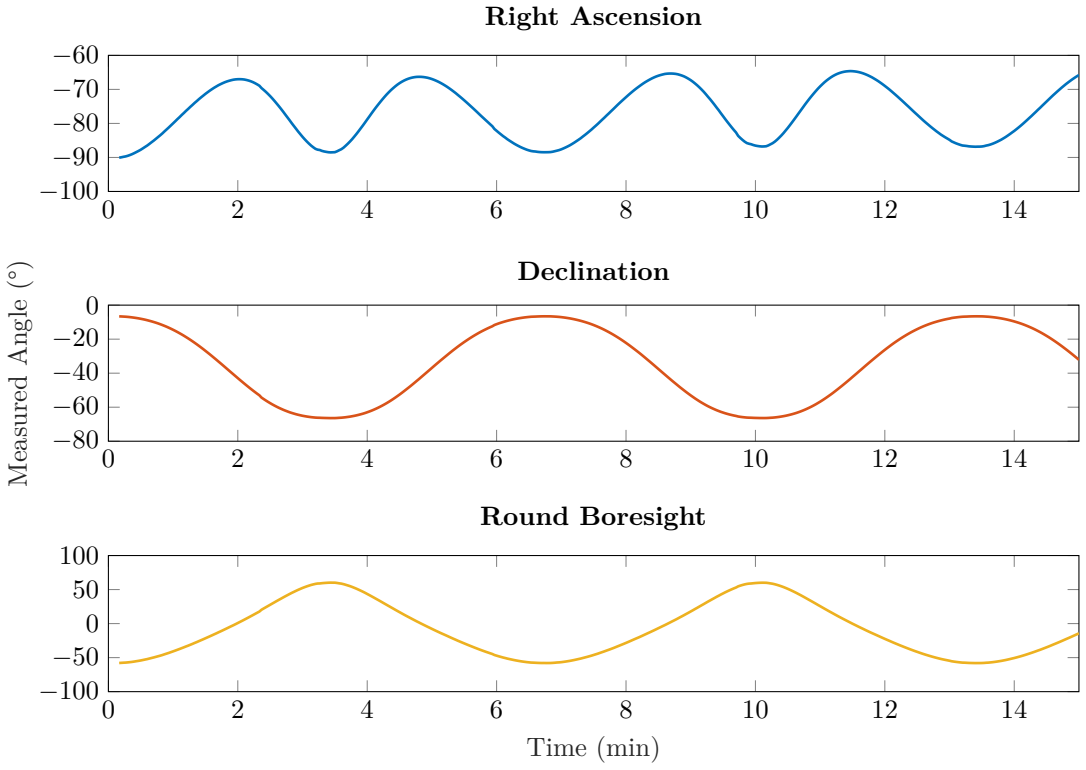



**Figure 7.8:** Estimated Orientation During Dynamic Testing

This figure does, however, show relatively smooth transitions with only some slight discontinuities at the rate transition spots. Owing to the nature of the required sinusoidal excitation, the rate table had to transition between gears, leading to the necessity of enabling the clutch, and subsequently causing a minor stall during transition periods. This stall is better demonstrated by looking at the numerical derivative of the right ascension data as shown in Figure 7.9.

From this figure, the periodic spike in right ascension rate data can clearly be seen. Another interesting phenomenon that can be observed is the correlation between the smoothness of the rate of right ascension change, number of stars observed and experienced system rate. This illustrates the dependence of the estimation stability, and therefore accuracy, on the number of stars tracked

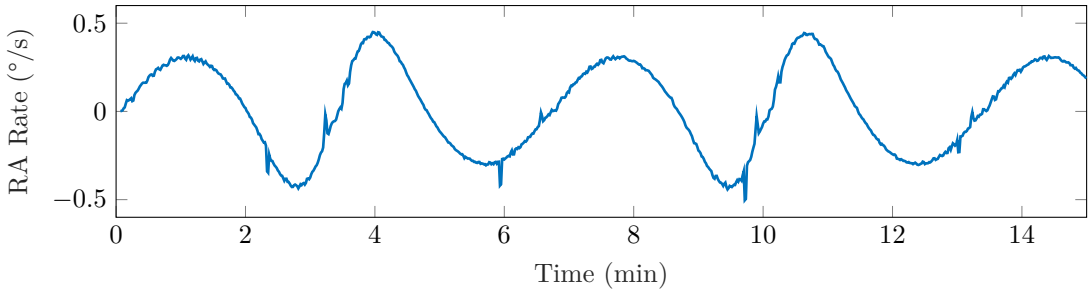



**Figure 7.9:** Rate of Right Ascension Change

and size of the rate excitation. Although not shown here, this phenomenon also affects both the declination and round-boresight data.

### 7.2.3 Rate Estimation and Bias Correction

As in the previous section, measured sensor rates were also obtained during these dynamic test conditions. These measured rates are shown in Figure 7.10.

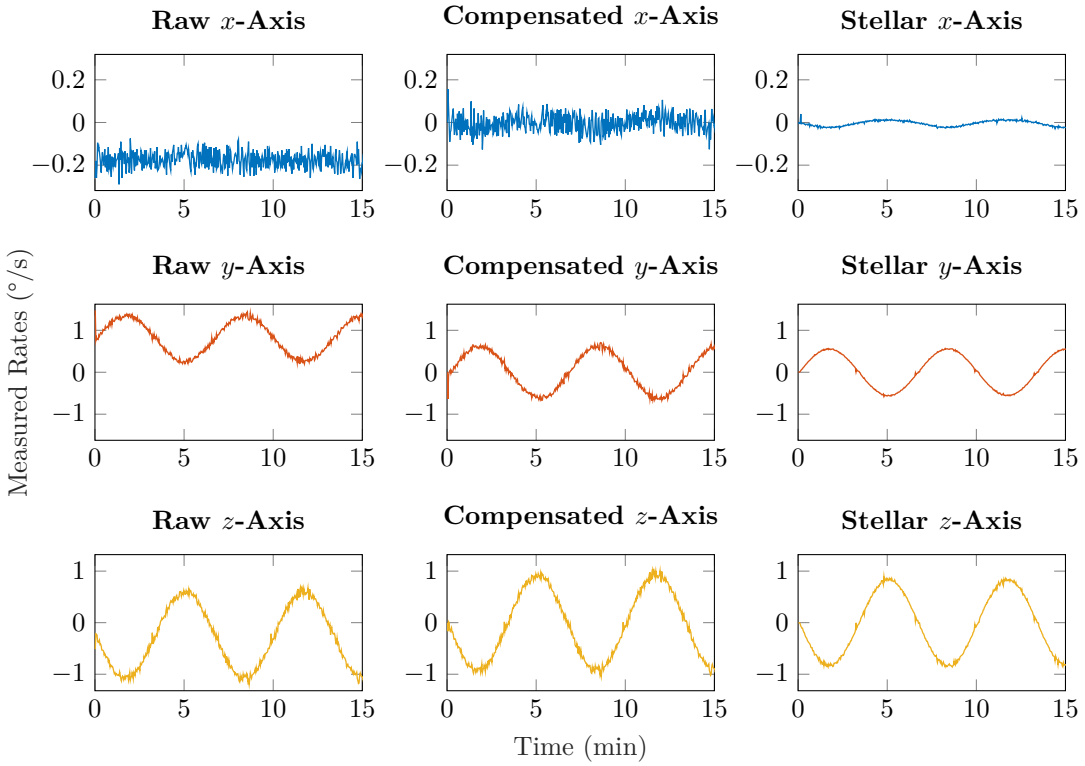



**Figure 7.10:** Measured Rates During Dynamic Conditions

This figure shows a comparison between the raw measured IMU rates, the IMU rates compensated by means of the estimated bias, and the stellar gyro rates around each axis. Like the test results obtained during the earth-fixed tests, the round-boresight rate estimation shows to be much less accurate than across the cross-boresight axes. Although no explicit rate was induced on the sensor $x$-axis, some cross coupling can be seen and is expected due to slight sensor misalignments. Overall, sensor cross coupling was very low, however. As in the orientation estimation results, it can be noted, however, that the rate estimation algorithm performance degrades at higher body rates. Similar to the previous dataset shown, this is mainly caused by the expected decrease in centroiding accuracy owing to the dispersion of the observed star spots.

### 7.2.4 Rate Residual Analysis

To determine the accuracy of the system with regards to the bias-compensated and stellar gyro rates, the ground truth input rate measurements can be used.

For each of the cases, the total experienced system rate was calculated by taking the norm of each angular rate measurement, the sign was then obtained by taking the measured sign of the $y$-gyro measurements. This process was repeated for the whole dataset for both the stellar gyro and compensated IMU rates. The measured stellar gyro rates along with true rates and residual are shown in Figure 7.11

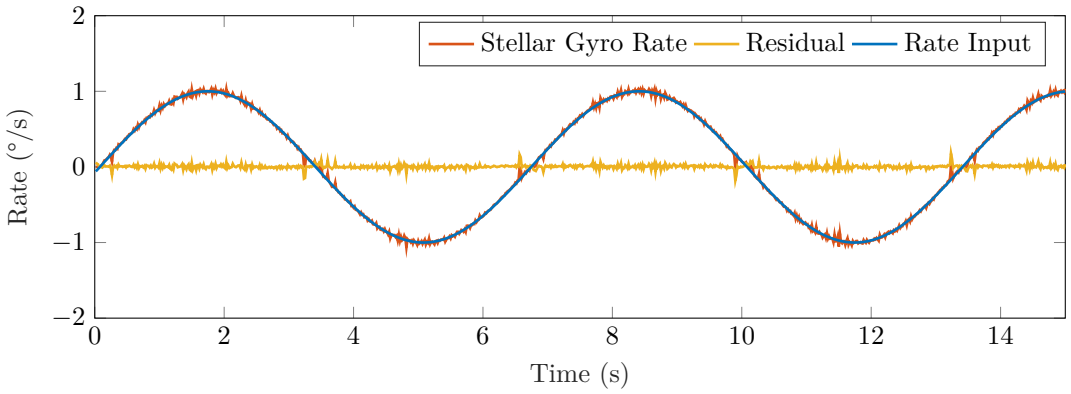



**Figure 7.11:** Stellar Gyro Rate Performance

This figure shows the relatively consistent measured rate noise determined by the stellar gyro algorithm. Also shown is the sudden rate discontinuities close to the rate crossing points. Overall, the stellar gyro estimates show satisfactory results with low noise amplitudes. The noise does however increase as rates increase, and reaches a maximum of 0.06 °/s. Important to note here, however, is that the rate residual is calculated without perfect knowledge of phase variations between the measured stellar gyro rates and the rate table setpoint. This is because subsecond synchronisation accuracy could not be achieved between inducing a rate excitation and requesting the estimated sensor rates.

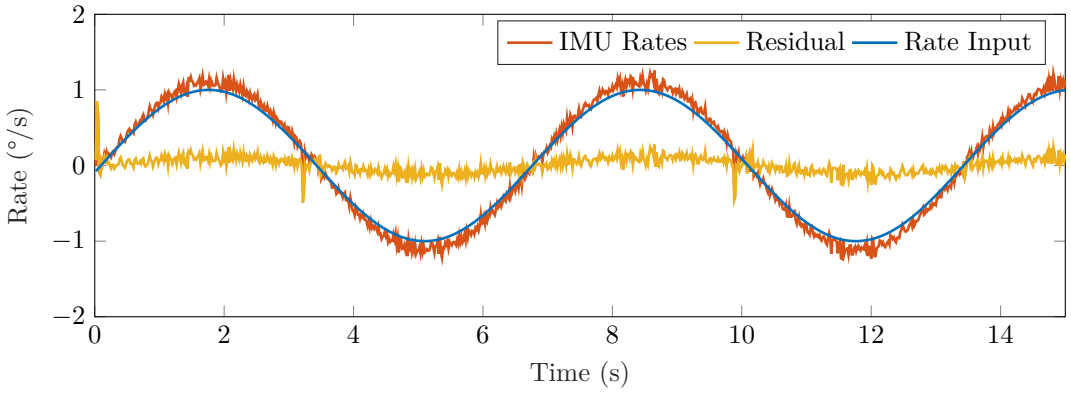



**Figure 7.12:** Bias-Corrected Rate Performance

The bias corrected results performed worse than that of the stellar gyro rates, as the bias was slightly overestimated due to the inaccurate star centroid determination during high slew-rates. This was further influenced by the number of stars tracked, as fewer visible stars meant that less corrections were available. Overall, however, the bias corrected performance was good in agreement with the raw measurements, and seen as functioning successful.

### 7.2.5 Discussion

As in the previous tests, results requested from the sensor showed good correspondence to those obtained during both earth-fixed testing and simulation. Although the number of stars was slightly lower than previous cases, this is expected as the other tests did not have the star streaking problem observed during these tests. The estimated attitude and rate, as well as the compensated rates worked as expected. In this case, no estimate of the system orientation accuracy is given as there was no ground truth measurement. Both the LSQ rates and the bias-corrected rates showed a greatly improved response above the raw rate measurements. In this case, the LSQ rate estimation formulation showed a much higher accuracy than that of the bias corrected values. Overall, the system showed a robust attitude and rate estimation response up to 1 °/s.

## 7.3 Estimated Average Power Consumption

In conclusion, to analyse the sensor's power capability, an estimate of the actual power consumption of the sensor is required. The current measurement output of the support hardware was connected to a Tektronix TDS 1012B oscilloscope. The current was measured and finally multiplied by the estimated supply voltage. Figure 7.13 shows this measured power usage. Like the first CubeStar

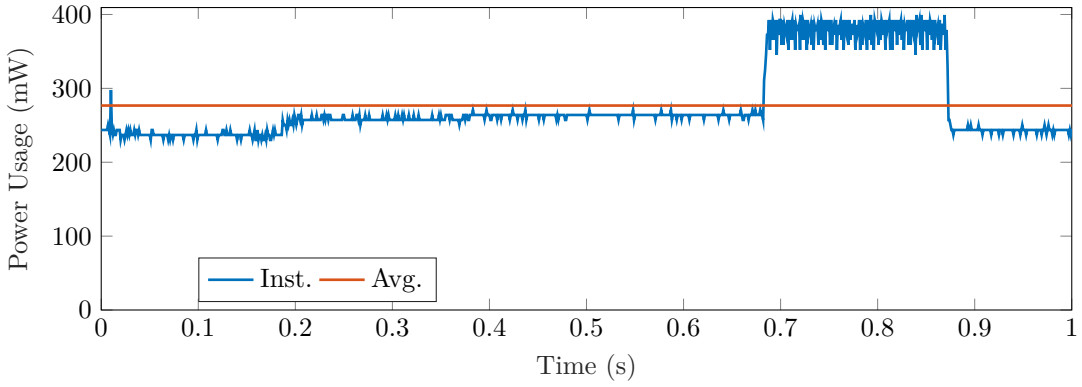



**Figure 7.13:** Measured Instantaneous Power Usage Over One Period

iteration, the power usage followed some distinct operational phases. In this case, these phases are i) processing, ii) integration, and iii) image download. A summary of these results is shown in the Table 7.4.

**Table 7.4:** Sensor Power Measurement Results

| **Usage Phase** | **Usage** (mW) | **Duration** (ms) |
|---|---|---|
| Processing | 240 | 0.3 |
| Integration | 270 | 0.5 |
| Download | 400 | 0.2 |
| Average | 277 | |

This measured power usage was, however, found to be significantly higher than that of CubeStar V4.2. This can mostly be attributed to the power consumption of the IMU. As the specified rate sensor current requirement was given as 44 to 55 mA, the expected IMU power consumption should range between 132 and 165 mA.

## 7.4 Chapter Summary

In this chapter, the final system validation testing was completed. These tests involved observing sensor functioning during both earth-fixed and slew conditions. These experiments were seen as being successful as attitude and rate estimation, as well as IMU bias compensation worked succesfully, thereby achieving the final project objective.

During both of these test cases, it was found that the sensor performed as expected, with the tracking mode able to track a sufficient number of stars during operation. Both orientation and rate estimation, as well as bias correction further showed promising results.

With the analysis of the attitude determination accuracy, it was found that the sensor could repeatably provide estimates up to an accuracy of 0.01° around the cross-axes, and 0.03° to 0.05° around the round-boresight axis during earth-fixed tests.

During slewed tests, no orientation accuracy estimates are given owing to the lack of available ground truth data.

In terms of measured rates, the stellar gyro rate estimates showed good performance. These measurements were unfortunately affected by the effect of the rate table gear transitions. Overall, however, the system showed successful functioning and could track the rates well with a good signal-to-noise ratio.

The system power was also analysed to determine the overall system performance. Although it was found to be higher than CubeStar V4.2, it was not unexpected as the IMU used is quite a power-intensive device. Overall, however, the system still functions within the required specifications.

Overall, the system showed successful function, delivering proof-of-concept results, as well as relatively robust operation.

# Chapter 8

# Conclusions and Recommendations

## 8.1 Summary

The overarching aim of this research project, developing a high-performance attitude and rate system, was therefore achieved and the project was deemed as successful. The technical requirements were met by completing the five objectives as defined in Chapter 1.

### Designing a Suitable Hardware Platform

An augmented stellar sensor based on an IMU and star tracker was designed by extending the CubeStar hardware platform. This augmented platform featured an improved, low-power, 32-bit MCU from the STM32L4 family showing increased computational capabilities, with a higher clock frequency than the original EFM32 MCU, as well as an FPU. The CubeStar platform was further extended by adding an ADIS16460 IMU, allowing high-frequency rate updates. Although the redesigned platform was not perfectly compliant with the interface of CubeStar V4.2, the hardware functioned as required and is deemed to be a success.

### Investigating Software Techniques

During this work, numerous software techniques as required by the integrated system were identified. It was decided to reuse most of the image processing algorithms as identified by Erlank [32] and Calitz [31]. With regards to LIS matching, the geometric voting star matching method scheme was identified to allow for initial star matching, as it was used successfully during the initial development of CubeStar. Numerous tracking methods were also identified for usage; however, it was decided to develop a gyro-aided tracking mode based on star location propagation as this enabled the most efficient usage of available resources. With regards to attitude determination, three alternatives were identified: TRIAD, QUEST, and an attitude and bias EKF. The EKF was chosen as it provided most of the required functionality. Finally, the rate estimation scheme, as developed by Crassidis et al. [30] was investigated, as it allows high accuracy, gyroless rate estimates.

### Implementing the system In Simulation

The EKF was then implemented in a simulated environment for initial proof-of-concept testing and validations. During these simulations, it was noted that leaving the EKF uninitialised led to sporadic, unexpected results such as high bias overshoots and algorithm instability. To compensate for this, it was decided to use the QUEST algorithm to provide the initial EKF state vector. A simulated system was then implemented to verify the interoperation of all required software techniques. All algorithms showed successful operation. In this case, the simulated IMU performed slightly worse than the LSQ rate estimation technique, as expected. Overall, however, the system worked as expected. Although the overall tracking mode performance was proven to perform better with a larger search grid size, using only a $3 \times 3$ grid was found to prove sufficient performance for initial algorithm validation. During system simulations and hardware-in-loop testing, it was further noted that the tracking mode showed a great performance increase when compared to that of the LIS mode. In this case, although the EKF was further found to require more computation

than QUEST, it was found to be a convenient formulation that allowed both attitude estimation and bias compensation.

### Integrating the Hardware and Software Interface

To ensure hardware and software integration, an emulation environment was developed with which initial functional testing could be performed. Although the emulation environment allowed for initial system testing, it was proven to be of insufficient accuracy for complete online algorithm validation. The emulated environment was therefore only used to enable proof-of-concept results. The final hardware calibration and characterisation was then performed, along with the last software integration, to ensure a fully functional attitude and rate estimation system.

### Validating Sensor Operation During Night Sky Tests

To analyse system performance, the developed sensor was tested under clear night sky conditions. This system was exposed to both earth-fixed and dynamic conditions. During both cases, the sensor showed successful functioning with regards to attitude and rate estimation, as well as IMU bias correction. The most notable sensor results are shown in Table 8.1.

**Table 8.1:** Improved CubeStar Specifications

| | | |
|---|---|---|
| **Attitude** ($3\sigma$) | Cross-axis | 0.0117° |
| | Round-axis | 0.0342° |
| **Rate** ($3\sigma$) | Cross-axis | 0.005 °/s |
| | Round-axis | 0.017 °/s |
| **IMU** ($3\sigma$) | Cross-axis | 0.086 °/s |
| | Round-axis | 0.067 °/s |
| | Catalogue Size | 410 |
| | Sensitivity | Up To 3.8 Mv |
| **Physical** | Mass | 87 g |
| | Size | $50 \times 35 \times 70$ mm |
| **Power Supply** | Supply voltage | 3.3 V |
| | Average Power Usage | 277 mW |
| | Peak Power Usage | 400 mW |

In these tests, the gyro-aided tracking mode showed excellent functioning during both high- and low-dynamic rates. Although the number of stars detected was found to be lower than those expected to be in the FOV, it is attributed to environmental conditions and diffraction in the earth's atmosphere, as well as the low signal to noise ratios observed during high slew conditions. The system was tested up to a rate of 1 °/s. The system further showed accurate IMU bias estimation results during low slew rates, although slight overestimations did occur during high rates, owing to inaccurate star-streak centroiding. Overall, however, the bias estimation was found to be successful, and was proven to mitigate per-axis gyro drift. In conclusion, the sensor solution proved to function successfully during the test conditions it was exposed to, showing attainable attitude accuracy in the order of 0.01° around the cross-boresight, and 0.03° around the round-boresight axes during earth-fixed testing. The sensor could deliver an update frequency of at least 1 Hz, with attitude propagates available at a higher frequency, if required. The system was found to use, on average, 277 mW.

The technical requirements were therefore fulfilled by:

- Implementing a star tracker-based system that delivers both rate and attitude data;
- Implementing, testing, and proving the functioning of an orientation and bias EKF in the estimation of IMU bias.

- Implementing a high-performance gyro-assisted star tracker tracking mode, based on a reduced ROI search and lookup-table, to reduce overall sensor computation and power;
- Developing a system that requires a maximum power consuption of around 400 mW;
- Implementing a high-accuracy rate estimation scheme for usage during low update rates; and
- Providing an integrated star tracker–gyro solution, functioning both accurately and autonomously.

## 8.2 Recommendations and Future Work

### 8.2.1 Better Night Sky Testing

Although night sky testing showed good results, no ground truth measurements are available to aid in the determination of sensor accuracy. It would therefore imperative to test the sensor in conditions where ground truth data is available to determine the actual accuracy of the developed system.

### 8.2.2 High-Dynamic Rate Centroid Reconstruction

One of the key problems identified during this project was the decrease in accuracy obtained from measurements when the sensor was subject to high body rates. This was caused by the streaking of stars across the image plane, brought on by high image sensor integration times. To compensate for this, it is suggested that investigation be done in the viability of a gyro-aided star-streak reconstruction algorithm. This work should be similar to the work of Sun et al. [44]; however, it should specifically be applied for online usage on a low-power platform.

Not only will this increase the signal to noise ratio of the star tracker, but it will also allow for much higher centroid detection performance during high slew conditions, as reconstructed stars will much closer resemble the desired point-spread star nature, enabling higher accuracy.

### 8.2.3 On-Orbit Recalibration Techniques

During this project, it was noted that the star tracker accuracy was very dependent on the environmental conditions. Not only does this severely impact the overall accuracy measurement of the star tracker, but it also impacted the calibration process. In this case, the calibrated parameters could fluctuate, dependent on the time of calibration. The accuracy of the star tracker system would therefore be questionable during actual on-orbit usage.

The estimated calibration parameters are further also quite sensitive to factors such as temperature and barometric pressure, as well as vibrations experienced during launch. It is therefore likely that lens characteristics might change between calibration and usage. Because of this, it is suggested that an in-orbit calibration scheme be developed to allow for automatic recalibration.

Not only will this be of use during on-orbit conditions, but it will also streamline the calibration process, by removing the need of downloading large image datasets. Only coarse calibration would therefore be necessary such that initial star matching is possible, where after autonomous calibration would be enabled.

### 8.2.4 Hardware Refinement

It was further noted that the hardware usage is suboptimal. A few hardware and data-flow optimisations can therefore be implemented.

#### Investigation of Alternate Microcontroller Technologies

With the global increase of the internet-of-things market, miniature, low-power devices have become prominent. Currently, some MCU families offer peripherals that can interface with image sensor devices. These interfaces are connected to separate Direct Memory Access peripherals and

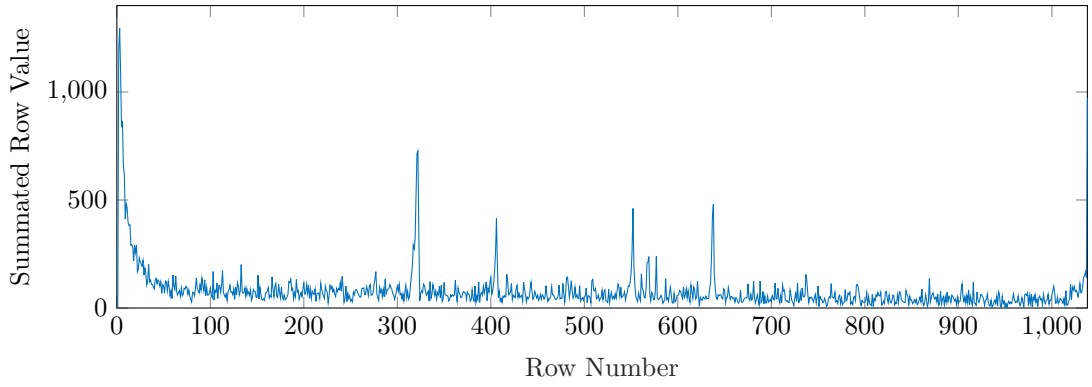



**Figure 8.1:** Image Spectrum

can autonomously, and at high speeds, transfer image data from a camera sensor to external SRAM. An example of a low-power MCU with such a peripheral, is the STM32L4+ series.

By utilising this MCU family, the FPGA would become redundant, subsequently making for a much smaller, more efficient star tracker assembly. The overall module production cost and power usage will also be decreased, leading to a much more elegant solution.

### Increasing FPGA Utilisation

Other optimisations depend on increasing the utilisation of the FPGA. An example of this entails developing an inline centroid detection technique. This will greatly decrease the required computation, as the image plane search procedure is the most computationaly expensive process in the CubeStar software stack. Although this does require extensive FPGA development, the determination of star centroids, as images are clocked from the image sensor, will therefore improve hardware utilisation and overall computational requirements.

Another method of decreasing computational cost during LIS mode, would be to use the FPGA in the calculation of the total value of each image row. If an image row contains a star, it will be easy to identify, as it will show up as a peak. An example of this is shown in Figure 8.1, where such an image spectrum was calculated. Here, the peaks of stars are clearly visible in the centre of the image plane. This process is not only simple to implement but can also lead to a great decrease in necessary computation during initial attitude acquisition, subsequently decreasing power usage and computational effort.

### Design of a High-Speed Calibration Device

Another problem identified during the development of the sensor, was that the camera calibration process is a long, tedious process that can span days. Not only is this highly dependent on weather, but also on the position and visibility of the moon. Optimal usage of clear, unobscured night-time conditions is therefore necessary. Currently, the download of calibration images over the serial link is a tedious process, taking up to a few hours to gather enough data for successful calibration.

To optimize this process, the design and implementation of a plug-in USB device that can attach to the image sensor assembly is suggested. This will allow for fast image transfer, decreasing required lens focusing and calibration time, as well as enabling real-time, offline algorithm testing and refinement.

# References


[1] H. Heidt, J. Puig-Suari, A. S. Moore, S. Nakasuka and R. Twiggs, "CubeSat: A new Generation of Picosatellite for Education and Industry Low-Cost Space Experimentation", in *AIAA/USU Conference on Small Satellites*, Logan, Utah, 2000, pp. 1–19.

[2] J. Chin, R. Coelho, J. Foley, A. Johnstone, R. Nugent, D. Pignatelli, S. Pignatelli, N. Powell, J. Puig-Suari, W. Atkinson, J. Dorsey, S. Higginbotham, M. Krienke, K. Nelson, B. Poffenberger, C. Raffington, G. Skrobot, J. Treptow, A. Sweet, J. Crusan, C. Galica, W. Horne, C. Norton and A. Robinson, "CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers", NASA, Tech. Rep. October, 2017, p. 96. [Online]. Available: `https://www.nasa.gov/sites/default/files/atoms/files/nasa%7B%5C_%7Dcsli%7B%5C_%7Dcubesat%7B%5C_%7D101%7B%5C_%7D508.pdf`.

[3] R. Garner, *NASA's Science Mission Directorate Cubesat Initiative*, 2017. [Online]. Available: `https://www.nasa.gov/content/goddard/nasas-science-mission-directorate-cubesat-initiative` (visited on 19/11/2018).

[4] K. Dallas, *Introduction to CubeSats*, 2015. [Online]. Available: `http://dallaskasaboski.blogspot.com/2015/07/introduction-to-cubesats.html` (visited on 19/11/2018).

[5] M. Swartwout, "You say "Picosat", I say "CubeSat": Developing a better taxonomy for secondary spacecraft", in *IEEE Aerospace Conference Proceedings*, vol. 2018-March, Big Sky, MT, USA: IEEE, 2018, pp. 1–17.

[6] A. Klesh and J. Krajewski, "MarCO: Mars Cube One - Lessons Learned from Readying the First Interplanetary Cubesats for Flight", in *49th Lunar and Planetary Science Conference*, The Woodlands, TX: Lunar and Planetary Institute, 2018.

[7] B. A. Cohen, P. O. Hayne, B. T. Greenhagen, D. A. Paige, J. M. Camacho, G. Sellar, K. Crabtree and C. Paine, "Payload Developments on the Lunar Flashlight Mission", in *48th Lunar and Planetary Science Conference*, The Woodlands, Tx, 2017.

[8] J. Hernando-Ayuso, S. Takahashi, S. Campagnola, T. Ikenaga, T. Yamaguchi, T. Hashimoto, C. H. Yam and B. Sarli, "Trajectory Design for the JAXA Moon Nano-Lander OMOTENASHI", in *31st Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, 2017, pp. 1–16.

[9] ESA, "The HIPPARCOS and TYCHO catalogues. Astrometric and photometric star catalogues derived from the ESA HIPPARCOS Space Astrometry Mission", *European Space Agency, (Special Publication) ESA SP*, vol. 1200, 1997.

[10] M. A. C. Perryman, L. Lindegren, J. Kovalevsky, E. Hog, U. Bastian, P. L. Bernacca, M. Creze, J. Donatowicz, M. Grenon, M. Grewing, F. van Leeuwen, H. van der Marel, F. Mignard, C. A. Murray, R. S. Le Poole, H. Schrijver, C. Turon, F. Arenou, M. Froeschle, C. S. Peterson, E. Hoeg and J.-F. Donati, "Letter to the Editor The Hipparcos Catalogue", *Astronomy & Astrophysics*, vol. 323, no. May 2014, pp. L49–L52, 1997.

[11] D. Kilkenny, "Basics - II. Time, Magnitudes and Spectral types", National Astrophysics and Space Science Programme, Tech. Rep., 2014, pp. 11–14. [Online]. Available: `https://www.star.ac.za/sites/default/files/downloads/basics2.pdf`.

[12] J. R. Wertz, *Spacecraft Attitude Determination and Control*, ser. Astrophysics and Space Science Library - Volume 73. Dordrecht, Netherlands: Kluwer Academic, 1978, p. 858.

[13] *Space mission engineering : the new SMAD*, ser. Space technology library ; v. 28. Hawthorne, CA: Microcosm Press, 2011, p. 412.

[14] J. Diebel, "Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors", Stanford University, Tech. Rep., 2006, pp. 1–35. [Online]. Available: https://www.astro.rug.nl/software/kapteyn/%7B%5C_%7Ddownloads/attitude.pdf.

[15] E. Lefferts, F. Markley and M. Shuster, "Kalman filtering for Spacecraft Attitude Estimation", *Journal of Guidance, Control, and Dynamics*, vol. 5, no. 5, pp. 417–429, 1982.

[16] C. C. Liebe, "Accuracy performance of star trackers - A tutorial", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 2, pp. 587–599, 2002.

[17] A. Read Eisenman and C. Liebe, "The Advancing State-of-the-art in Second Generation Star Trackers", in *1998 IEEE Aerospace Conference Proceedings (Cat. No.98TH8339)*, vol. 1, Snowmass at Aspen, CO, 1998, pp. 111–118.

[18] C. C. Liebe, "Star trackers for attitude determination", *IEEE Aerospace and Electronic Systems Magazine*, vol. 10, no. 6, pp. 10–16, 1995.

[19] AdcoleMA, "MAI-SS Space Sextant Datasheet", AdcoleMA, Crafton, MD, Tech. Rep., 2015, pp. 1–3. [Online]. Available: https://www.cubesatshop.com/wp-content/uploads/2016/06/MAI-SS-Specification-10-11-17.pdf.

[20] TY-Space, "NST04 Datasheet", TY-Space, Beijing, China, Tech. Rep., 2018, pp. 1–2. [Online]. Available: http://www.ty-space.com/en/uploadfile/files/201801/201801152203028373.pdf.

[21] Hyperion Technologies, "ST200 Datasheet", Hyperion Technologies, Berlin, Tech. Rep., 2015, pp. 1–2. [Online]. Available: https://hyperiontechnologies.nl/wp-content/uploads/2018/07/HTBST-ST200-V1.01%7B%5C_%7DFlyer.pdf.

[22] CubeSpace, *CubeStar Datasheet*, [Online]. Available: https://cubespace.co.za/. [Accessed: 09 Sep 2018], 2017. (visited on 14/09/2018).

[23] D. Hegel, *FlexBus - A 6U CubeSat Platform for Any Mission*, 2016. [Online]. Available: http://mstl.atl.calpoly.edu/%7B~%7Dbklofas/Presentations/DevelopersWorkshop2016/3%7B%5C_%7DDanielHegel.pdf (visited on 19/11/2018).

[24] F. Irom and G. R. Allen, *CubeSat Star Tracker - Radiation*, 2016. [Online]. Available: https://nepp.nasa.gov/workshops/etw2016/talks/14TUE/20160614-1100-Irom-ETWPresentationJune2016new1gra-CL16-3400.pdf (visited on 19/11/2018).

[25] W. Kao, "Integration of GPS and dead-reckoning navigation systems", in *Vehicle Navigation and Information Systems Conference, 1991*, vol. 2, Troy, MI: IEEE, 1991, pp. 635–643.

[26] O. J. Woodman, "An Introduction To Inertial Navigation", University of Cambridge, Cambridge, UK, Tech. Rep. no. 696, 2007, pp. 1–37.

[27] S. K. Hong and S. Park, "Minimal-drift heading measurement using a MEMS gyro for indoor mobile robots", *Sensors*, vol. 8, no. 11, pp. 7287–7299, 2008.

[28] A. M. Sabatini, "Kalman-filter-based orientation determination using inertial/magnetic sensors: Observability analysis and performance evaluation", *Sensors (Basel, Switzerland)*, vol. 11, no. 10, pp. 9182–9206, 2011.

[29] C. C. Liebe, K. Gromov and D. M. Meller, "Toward a Stellar Gyroscope for Spacecraft Attitude Determination", *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 1, pp. 91–99, 2004.

[30] J. L. Crassidis, "Angular Velocity Determination Directly from Star Tracker Measurements", *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 6, pp. 1165–1168, 2002.

[31] N. Calitz, "The Design and Implementation of a Stellar Gyroscope for Accurate Angular Rate Estimation on CubeSats", MEng, University of Stellenbosch, Stellenbosch, 2015.

[32] A. O. Erlank, "Development of CubeStar: A CubeSat-Compatible Star Tracker", MEng, Stellenbosch University, 2013.

[33] P. Singla, J. L. Crassidis and J. L. Junkins, "Spacecraft Angular Rate Estimation Algorithms For Star Tracker-Based Attitude Determination", *Advances in the Astronautical Sciences*, vol. 114, no. SUPPL. Pp. 1297–1314, 2003.

[34] M. Pal and M. S. Bhat, "Star sensor based spacecraft angular rate estimation independent of attitude determination", in *Proceedings of the IEEE International Conference on Control Applications*, Hyderabad, India, 2013, pp. 580–585.

[35] G. Fasano, G. Rufino, D. Accardo and M. Grassi, "Satellite angular velocity estimation based on star images and optical flow techniques", *Sensors (Switzerland)*, vol. 13, no. 10, pp. 12 771–12 793, 2013.

[36] T. M. Brady, C. E. Tillier, R. A. Brown, R. Antonio, A. S. Kourepenis and Draper Labs, "The Inertial Stellar Compass : A New Direction in Spacecraft Attitude Determination", in *16th Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, 2002, pp. 18–23.

[37] NASA JPL, *Inertial Stellar Compass*. [Online]. Available: `https://www.jpl.nasa.gov/missions/web/compass%7B%5C_%7Dpackage.jpg` (visited on 25/11/2018).

[38] C. J. Finley and N. Peck, "TacSat-2: A Story of Survival", in *21st Annual AIAA/USU Conference on Small Satellites*, Logan, Utah, 2007, pp. 1–16.

[39] Jena Optronik, "ASTROgyro Datasheet", Jenna Optronik, Jena, Germany, Tech. Rep., 2016, pp. 1–2. [Online]. Available: `https://www.jena-optronik.de/products/attitude-and-orbit-control-systems/astrogyro.html?file=tl%7B%5C_%7Dfiles/pdf/Data%20Sheet%20ASTROgyro.pdf`.

[40] ——, *ASTROGYRO Qualification*. [Online]. Available: `https://artes.esa.int/projects/astrogyro` (visited on 19/11/2018).

[41] S. A. Rawashdeh, W. C. Danhauer and J. E. Lumpp, "Design of a Stellar Gyroscope for visual attitude propagation for small satellites", in *IEEE Aerospace Conference Proceedings*, Big Sky, MT, USA, 2012, pp. 1–9.

[42] S. A. Rawashdeh, J. E. Lumpp, J. Barrington-Brown and M. Pastena, "A Stellar Gyroscope for Small Satellite Attitude Determination", in *26th AIAA/USU Conference on Small Satellites*, Logan, Utah, 2012, pp. 1–9.

[43] J. Lu, L. Yang and H. Zhang, "A hybrid method for accurate star tracking using star sensor and gyros", *Review of Scientific Instruments*, vol. 88, no. 10, p. 105 004, 2017.

[44] T. Sun, F. Xing, Z. You, X. Wang and B. Li, "Deep coupling of star tracker and MEMS-gyro data under highly dynamic and long exposure conditions", *Measurement Science and Technology*, vol. 25, no. 8, 2014.

[45] A. O. Erlank and W. H. Steyn, "Arcminute attitude estimation for CubeSats with a novel nano star tracker", in *19th World Congress of The International Federation of Automatic Control*, vol. 19, Capte Town, South Africa: IFAC, 2014, pp. 9679–9684.

[46] E2v, "EV76C560 Datasheet", e2v, Tech. Rep., 2011, pp. 1–107. [Online]. Available: `http://www.e2v.com/resources/account/download-datasheet/3600`.

[47] Lensation, "Lensagon BL6012 Datasheet", Lensation, Tech. Rep., 2017, p. 1.

[48] Sillicon Labs, "EFM32 Giant Gecko Family Datasheet", Silicon Labs, Austin, TX, Tech. Rep., 2018, p. 432. [Online]. Available: `https://www.silabs.com/documents/public/data-sheets/efm32gg-datasheet.pdf`.

[49] CubeSpace, *CubeStar Interface Control Document*, 2017.

[50] J. Yiu, "ARM White Paper - ARM Cortex-M for Beginners", ARM, Tech. Rep. September, 2016, pp. 1–25.

[51] S. Sadasivan, "An Introduction to the ARM Cortex-M3 Processor", Tech. Rep., 2006, pp. 1–17. [Online]. Available: `https://www.arm.com/ja/files/pdf/IntroToCortex-M3.pdf`.

[52] Sillicon Labs, "ARM Cortex-M3 and M4 MCU Architecture", Sillicon Labs, Tech. Rep., pp. 1–7. [Online]. Available: https://www.silabs.com/documents/public/white-papers/arm%7B%5C_%7Dcortex%7B%5C_%7Dm3%7B%5C_%7Dand%7B%5C_%7Dm4%7B%5C_%7Dmcu%7B%5C_%7Darchitecture.pdf.

[53] Freescale Semiconductor, "Kinetis K24F Sub-Family Data Sheet", NXP, Tech. Rep., 2014, pp. 1–80. [Online]. Available: https://www.nxp.com/docs/en/data-sheet/K24P144M120SF5.pdf.

[54] STMicroelectronics, *STM32L476xx Datasheet*, 2018. [Online]. Available: https://www.st.com/en/microcontrollers/stm32l476vg.html (visited on 16/09/2018).

[55] Atmel, "Atmel SMART SAM G54 Datasheet", Atmel, Tech. Rep., 2014. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11266-32-bit%20Cortex-M4-Microcontroller-SAM-G54%7B%5C_%7DDatasheet.pdf.

[56] Sillicon Labs, "EFM32 Pearl GeckoData Sheet", Sillicon Labs, Tech. Rep., 2018. [Online]. Available: https://www.silabs.com/documents/public/data-sheets/efm32pg12-datasheet.pdf.

[57] N. de Rooij, S. Gautsch, D. Briand, C. Marxer, G. Mileti, W. Noell, H. Shea, U. Staufer and van der Schoot.B, "MEMS FOR SPACE", in *15th International Conference on Solid-State Sensors, Actuators and Microsystems*, Denver, CO, 2009, pp. 17–24.

[58] H. R. Shea, "MEMS for pico- to micro-satellites", in *SPIE 7208, MOEMS and Miniaturized Systems VIII*, vol. 7208, 2009, pp. 1–8.

[59] W. S. Otaguro, E. Udd and R. F. Cahill, "Fiber-Optic Gyro For Space Applications", in *SPIE 0616, Optical Technologies for Communication Satellite Applications,*, vol. 0616, 1986, pp. 616–619.

[60] W. Wang, Q.-S. Yang and X.-F. Wang, "Application of fiber-optic gyro in space and key technology", *Hongwai yu Jiguang Gongcheng/Infrared and Laser Engineering*, vol. 35, no. 5, pp. 509–512, 2006.

[61] N. Barbour and G. Schmidt, "Inertial sensor technology trends", *IEEE Sensors Journal*, vol. 1, no. 4, pp. 332–339, 2001.

[62] V. M. N. Passaro, A. Cuccovillo, L. Vaiani, M. De Carlo and C. E. Campanella, "Gyroscope Technology and Applications: A Review in the Industrial Perspective", in *Sensors*, vol. 17, 2017, p. 2284.

[63] A. Chin, R. Coelho, R. Nugent, R. Munakata and J. Puig-Suari, "CubeSat: The Pico-Satellite Standard for Research and Education", in *AIAA SPACE 2008 Conference & Exposition*, San Dieco GA, 2008, pp. 1–11.

[64] G. Pascoli, "The Sagnac effect and its interpretation by Paul Langevin", *Comptes Rendus Physique*, vol. 18, no. 9-10, pp. 563–569, Nov. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1631070517300907.

[65] T. Giallorenzi, J. Bucaro, A. Dandridge, G. Sigel, J. Cole, S. Rashleigh and R. Priest, "Optical fiber sensor technology", *IEEE Journal of Quantum Electronics*, vol. 18, no. 4, pp. 626–665, Apr. 1982. [Online]. Available: http://ieeexplore.ieee.org/document/1071566/.

[66] A. Burg, A. Meruani, B. Sandheinreich and M. Wiickmann, "MEMS Gyrocopes and Their Applications", Tech. Rep., 1992, pp. 389–445.

[67] J. Sommeria, "Foucault and the rotation of the Earth", *Comptes Rendus Physique*, vol. 18, no. 9-10, pp. 520–525, Nov. 2017. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1631070517301019.

[68] J. Söderkvist, "Micromachined gyroscopes", *Sensors and Actuators: A.*, vol. 43, no. 1-3, pp. 65–71, 1994.

[69] KVH Inc., "Guide to comparing gyro and IMU technologies Micro-electro-mechanical systems and fiber optic gyros", 2014, [Online]. Available: http://webnews.kvh.com/LP-WP-Guide-to-Comparing-Gyro--IMU-Technologies%7B%5C_%7DBanner.html.

[70] M. Kirkko-Jaakkola, J. Collin and J. Takala, "Bias prediction for MEMS gyroscopes", *IEEE Sensors Journal*, vol. 12, no. 6, pp. 2157–2163, 2012.

[71] Freescale Semiconductor, "Allan Variance: Noise Analysis for Gyroscopes", *Freescale white paper*, p. 9, 2015. [Online]. Available: http://cache.freescale.com/files/sensors/doc/app%7B%5C_%7Dnote/AN5087.pdf.

[72] EeNewsEurope, *24x24x10mm IMUS optimized for autonomous applications*, Digital, 2018. [Online]. Available: http://www.eenewsanalog.com/node/104532 (visited on 23/11/2018).

[73] Sensonor, *STIM202*, 2018. [Online]. Available: https://www.sensonor.com/products/gyro-modules/stim202/ (visited on 23/11/2018).

[74] Digikey, *ADIS16460AMLZ-ND*. [Online]. Available: https://www.digikey.com/product-detail/en/analog-devices-inc/ADIS16460AMLZ/ADIS16460AMLZ-ND/5957823 (visited on 23/11/2018).

[75] S. W. Janson and R. P. Welle, "The NASA Optical Communication and Sensor Demonstration Program", *Annual AIAA/USU Conference on Small Satellites*, pp. 1–10, 2013.

[76] Sensonor, "STIM202 Multi-axis Gyro Module Datasheet", Sensonor, Tech. Rep. March, 2010, pp. 1–41. [Online]. Available: https://www.sensonor.com/media/1074/datasheet-stim202-ts1439-r6.pdf.

[77] Epson, "M-G364PD Datasheet", Epson, Tech. Rep., 2018, pp. 2–4. [Online]. Available: https://global.epson.com/products%7B%5C_%7Dand%7B%5C_%7Ddrivers/sensing%7B%5C_%7Dsystem/assets/pdf/m-g364%7B%5C_%7Dbriefsheet%7B%5C_%7De%7B%5C_%7Drev20180228.pdf.

[78] Analog Devices, "ADIS16460 Datasheet", Tech. Rep., 2012. [Online]. Available: https://www.analog.com/media/en/technical-documentation/data-sheets/adis16460.pdf.

[79] C. Fosu, G. W. Hein and B. Eissfeller, "Determination of Centroid of Ccd Star Images", in *35th International congress for photogrammetry and remote sensing;*, vol. 1, 2004, pp. 612–617.

[80] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. New York, NY, USA: Cambridge University Press, 2003.

[81] P. Drap and J. Lefèvre, "An exact formula for calculating inverse radial lens distortions", *Sensors (Switzerland)*, vol. 16, no. 6, pp. 1–18, 2016.

[82] D. Brown, "Decentering Distortion of Lenses - The Prism Effect Encountered in Metric Cameras can be Overcome Through Analytic Calibration", *Photometric Engineering*, vol. 32, no. 3, pp. 444–462, 1966.

[83] M. Kolomenkin, S. Pollak, I. Shimshoni and M. Lindenbaum, "Geometric voting algorithm for star trackers", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 2, pp. 441–456, 2008.

[84] F. Markley, "Attitude Determination Using Two Vector Measurements", NASA Goddard Space Flight Center, Greenbelt MD, Tech. Rep., 1998, pp. 1–13.

[85] G. Wahba, "A Least Squares Estimate of Satellite Attitude", *SIAM Review*, vol. 7, no. 3, pp. 409–409, Jul. 1965. [Online]. Available: http://epubs.siam.org/doi/10.1137/1007077.

[86] F. L. Markley and D. Mortari, "How to estimate attitude from vector observations", *Advances in the Astronautical Sciences*, vol. 103, no. PART III, pp. 1979–1996, 2000.

[87] D. Mortari, F. Markley and P. Singla, "An Optimal Linear Attitude Estimator", *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 6, pp. 1619–1627, 2007.

[88] W. Kühlbrandt, "Two-dimensional crystallization of membrane proteins.", *Quarterly reviews of biophysics*, vol. 25, no. 1, pp. 1–49, Feb. 1992. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0094114X15000415%20http://www.ncbi.nlm.nih.gov/pubmed/1589568.

[89] A. Gelb, J. F. Kaser, R. A. Nash, A. Price and C. F. Sutherland, *Applied optimal estimation*, A. Gelb, Ed. M.I.T. Press, 1974.

[90] S. Roumeliotis, G. Sukhatme and G. Bekey, "Circumventing dynamic modeling: evaluation of the error-state Kalman filter applied to mobile robot localization", *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, no. May, pp. 1656–1663, 1999.

[91] M. Ahmadi, A. Khayatian and P. Karimaghaee, "Orientation estimation by error-state extended Kalman filter in quaternion vector space", *Proceedings of the SICE Annual Conference*, pp. 60–67, 2007.

[92] F. L. Markley, "Attitude estimation or quaternion estimation?", *Journal of the Astronautical Sciences*, vol. 52, no. 1, pp. 221–238, 2004.

[93] J. Solà, *Quaternion kinematics for the error-state KF Quaternion kinematics for the error-state KF*, 2017. [Online]. Available: `http://www.iri.upc.edu/people/jsola/JoanSola/objectes/notes/kinematics.pdf`.

[94] G. Golub and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions", *Numerische mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[95] Z. Diao, H. Quan, L. Lan and Y. Han, "Analysis and compensation of MEMS gyroscope drift", in *2013 Seventh International Conference on Sensing Technology (ICST)*, Wellington, New Zealand: IEEE, Dec. 2013, pp. 592–596. [Online]. Available: `http://ieeexplore.ieee.org/document/6727722/`.

[96] N. El-Sheimy, H. Hou and X. Niu, "Analysis and Modeling of Inertial Sensors Using Allan Variance", *IEEE Transactions on Instrumentation and Measurement*, vol. 57, no. 1, pp. 140–149, Jan. 2008. [Online]. Available: `http://ieeexplore.ieee.org/document/4404126/`.

[97] IEEE, "IEEE Standard Specification Format Guide and Test Procedure for Single -Axis Interferometric Fiber Optic Gyros", *IEEE Std 952-1997*, pp. 1–84, 1998. [Online]. Available: `http://dx.doi.org/10.1109/IEEESTD.1998.86153`.