drones

Article

Imagery Synthesis for Drone Celestial Navigation Simulation

Samuel Teague [1,*] and Javaan Chahl [1,2]

[1] School of Engineering, University of South Australia, Mawson Lakes, SA 5095, Australia
[2] Defence Science and Technology Group, Joint and Operations Analysis Division, Melbourne, VIC 3207, Australia
* Correspondence: samuel.teague@mymail.unisa.edu.au

Abstract: Simulation plays a critical role in the development of UAV navigation systems. In the context of celestial navigation, the ability to simulate celestial imagery is particularly important, due to the logistical and legal constraints of conducting UAV flight trials after dusk. We present a method for simulating night-sky star field imagery captured from a rigidly mounted 'strapdown' UAV camera system, with reference to a single static reference image captured on the ground. Using fast attitude updates and spherical linear interpolation, images are superimposed to produce a finite-exposure image that accurately captures motion blur due to aircraft actuation and aerodynamic turbulence. The simulation images are validated against a real data set, showing similarity in both star trail path and magnitude. The outcomes of this work provide a simulation test environment for the development of celestial navigation algorithms.

Keywords: celestial; stellar; simulation; strapdown; imagery

check for updates

Citation: Teague, S.; Chahl, J. Imagery Synthesis for Drone Celestial Navigation Simulation. *Drones* **2022**, *6*, 207. https://doi.org/10.3390/drones6080207

Academic Editors: Andrzej Łukaszewicz, Wojciech Giernacki, Zbigniew Kulesza, Jaroslaw Pytka and Andriy Holovatyy

Received: 19 July 2022
Accepted: 11 August 2022
Published: 15 August 2022

1. Introduction

Celestial navigation in uncrewed aerial vehicles (UAV) has been a topic of interest for over half a century (see, for example, [1]). The significance of this mode of navigation has been overshadowed, however, by the ubiquity of global navigation satellite systems and the integration of compact micro-electromechanical attitude sensors into aviation platforms. Nonetheless, celestial navigation has unique advantages due to its independence from critical infrastructure and robustness to external interference. We see recent works, such as [2,3] integrating celestial imaging into their navigation solutions. Modern UAVs must typically conform to size, weight and power constraints and, to this end, benefit from a strapdown celestial implementation, as opposed to an actively stabilized alternative. In a strapdown configuration, the imaging sensor has no control authority over the vehicle, and therefore requires a larger field of view, and longer exposure intervals, to track stars during motion. We propose here a method for simulating the imagery captured from such a strapdown celestial system.

Celestial imagery is commonly used in spacecraft to obtain a highly accurate attitude reference. This technique is less commonly used, however, in low altitude aircraft navigation. Aircraft are subjected to many sources of noise that spacecraft are not, such as light pollution, cloud cover, atmospheric diffraction, aerodynamic turbulence, engine vibration and control/actuation, which all impact the signal strength of a celestial image obtained from an aircraft. These effects are exacerbated by the need for long-exposure imagery when operating at low-altitude (less than 500 m). The standard approach to this problem is to use a stabilized viewing platform with a telescopic lens [1], which limits the aforementioned attenuation. Such an approach is costly and adds significant weight to an airframe. The design of this simulation has arisen from the desire for a low cost, low weight, "strapdown" [4] celestial navigation solution.

As with all avionic navigation solutions, simulation plays an important role in the system design and precedes the implementation. The intent of this work is to provide a means of simulating imagery from a camera with a wide-angle lens, rigidly mounted to a fixed wing airframe with no active stabilization. Preliminary testing indicated that despite the increased motion blur, longer exposure images are consistently able to capture stars at lower levels of illumination. Consequently, the need arises for a simulation that can replicate the effects of motion blur due to these longer exposure images. In addition, there is benefit to tuning the simulation based on a reference image captured from the ground. This provides a quick solution for users to encapsulate their camera sensor and lens characteristics in the simulation environment, without the logistical constraints of night flying.

The use of star field simulation is most commonly seen in star identification research and development [5]. In this field, simulation is used to obtain baseline performance metrics for newly designed algorithms. An example of star field simulation for this purpose can be seen in [6], in which, rather than rendering stars, their position and magnitude are generated directly, with the addition of Gaussian noise. Other works tend to follow a similar design, seen in [7,8]. These simulations are intended to replicate imagery captured from spacecraft, which is not typically affected by rotational motion, nor atmospheric attenuation. Recent work considers the effects of star smearing [9] and the effect this has on the observability of stars. This work leverages from the simulation concepts presented in [10], later followed by [11]. These studies assume the angular velocity of the camera to be constant; however, we can see in Section 3 that for aerial vehicles, this assumption is invalid. Advancements were made in [12], highlighting the importance of modeling in star sensor design and calibration. In each of these cases, testing was conducted using a turntable, and as indicated in [10], this approach is not capable of running in real-time due to the large number of integral calculations involved.

This work offers two significant contributions to this field of research. We present a simple and effective framework for the real-time simulation of long-exposure images from non-stabilized UAV-mounted hardware using spherical linear interpolation, and a method for calibrating the simulation based on a ground reference image. Concepts from this work may be extended to aid in simulation design for spacecraft in highly dynamic situations.

2. Simulation Architecture

The position of stars and planets in the sky are represented in the International Celestial Reference Frame (ICRF). The location of a celestial body is expressed in terms of right ascension, α, and declination, δ, as seen in Figure 1. Stars are assumed to be infinitely far away, lying on the celestial sphere. Consequently, translational motion has no effect on the apparent position of the stars.

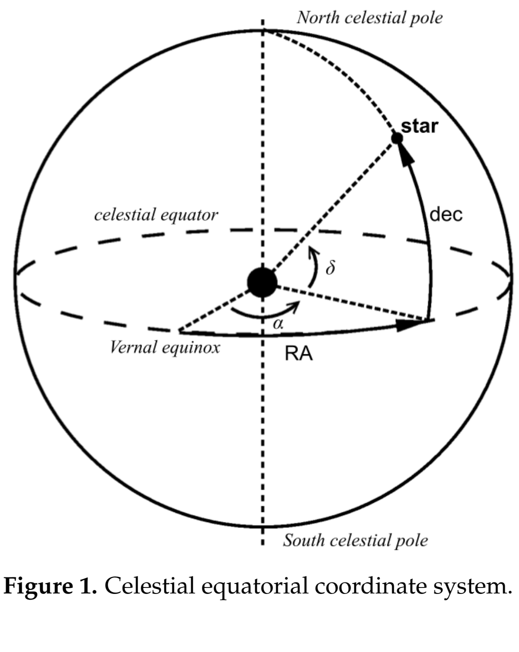


Figure 1. Celestial equatorial coordinate system.

The aircraft position is represented in the Earth-centred, Earth-fixed WGS84 system. This is the standard reference frame for GPS positional data. Aircraft attitude is represented in the local North-East-Down (NED) coordinate system. The camera is assumed to be mounted to the aircraft, sharing a location with the vehicle and differing in orientation by a single rotational transformation. The camera coordinate system is oriented with z positive in the optical axis, x positive towards the right of the image plane and y positive towards the bottom of the image plane.

A star catalogue must be used as a reference for the location of stars. While there are many star catalogues available, we selected the Yale Bright Star Catalogue (BSC) due to its minimal size. The BSC contains records of stars down to magnitude 6.5, totalling 9110 stars. This magnitude threshold is sufficient for most aircraft camera systems (including stabilized systems) [1]. For ease of implementation, the ASCII-format catalogue was converted into an SQLite database. Indices were created for magnitude, right ascension and declination, including additional composite indices to allow fast querying of the database.

This simulation initially corrects for celestial phenomena which perturb the right ascension and declination in the star catalogue, before entering a simulation loop. The simulation loop performs the following steps:

1. Update position, time and attitude of simulation;
2. Calculate star homogeneous coordinates in camera frame of reference;
3. Project each star onto the image plane;
4. Render the star.

2.1. Initial Corrections

On initialization, adjustments are made for the right ascension and declination of stars due to annual proper motion (the apparent motion of stars), precession (changes in the Earth's rotational axis over time), nutation (axial changes due to the Moon's gravitational pull) and aberration (due to the velocity of the Earth's orbit).

Proper motion is provided in star catalogues and is compensated by computing the change in right ascension and declination since the given epoch, such that:

$$\begin{aligned}\hat{\alpha} &= \alpha_0 + \dot{\alpha}T \\ \hat{\delta} &= \delta_0 + \dot{\delta}T\end{aligned} \tag{1}$$

where α_0 and δ_0 are the right ascension and declination at the epoch. $\dot{\alpha}$ and $\dot{\delta}$ are the annual proper motion, typically expressed in arcseconds per year, and T is the time (in years) since the epoch (J2000 in our case). Subsequently, right ascension and declination are corrected due to precession, following the method outlined in [13]. That is, we use the polynomial approximation for the precession angles ζ, z and γ:

$$\begin{aligned}\zeta &= 2306.2181t + 0.30188t^2 + 0.017998t^3 \\ z &= 2306.2181t + 1.09468t^2 + 0.018203t^3 \\ \gamma &= 2004.3109t - 0.42665t^2 - 0.041833t^3\end{aligned} \tag{2}$$

where t is the number of centuries since the J2000 epoch. Right ascension α and declination δ are then found given:

$$\begin{aligned}\tan(\alpha - z) &= \frac{\sin(\hat{\alpha} + \zeta)}{\cos\gamma\ \cos(\hat{\alpha} + \zeta) - \sin\hat{\gamma}\ \tan\hat{\delta}} \\ \sin(\delta) &= \sin\gamma\ \cos\hat{\delta}\ \cos(\hat{\alpha} + \zeta) + \cos\gamma\ \sin\hat{\delta}\end{aligned} \tag{3}$$

Following [13], we correct for the effects of nutation. Nutation is comprised of nutation in longitude, $\Delta\lambda_n$, and nutation in obliquity, $\Delta\epsilon$. These quantities can be approximated to within 0.5 arcseconds by the following equations (expressed in arcseconds):

$$\Delta\lambda_n = -17.20\sin\Omega + 1.32\sin 2L - 0.23\sin 2L' + 0.21\sin 2\Omega \tag{4}$$
$$\Delta\epsilon = 9.2\cos\Omega + 0.57\cos 2L + 0.10\cos 2L' - 0.09\cos 2\Omega \tag{5}$$

where Ω, the longitude of the ascending node of the Moon's mean orbit on the ecliptic, expressed in degrees, is approximated as:

$$\Omega = 125.04452 - 1934.136261t \tag{6}$$

where t is expressed in Julian centuries, as above. L and L', the mean longitudes of the Sun and Moon, respectively, expressed in degrees, are given by:

$$\begin{aligned} L &= 280.4665 + 36000.7698t \\ L' &= 218.3165 + 481267.8813t \end{aligned} \tag{7}$$

The mean obliquity of the ecliptic can be found given the following equation, expressed in degrees:

$$\epsilon_0 = 23.439291 - 13.004166\, t \times 10^{-3} - 0.1639\, t^2 \times 10^{-6} + 0.50361\, t^3 \times 10^{-6} \tag{8}$$

Subsequently, the true obliquity of the ecliptic ϵ is given by:

$$\epsilon = \epsilon_0 + \Delta\epsilon \tag{9}$$

The resulting corrections due to nutation, $\Delta\alpha_n, \Delta\delta_n$ for the star's right ascension and declination, respectively, are given by:

$$\Delta\alpha_n = (\cos\epsilon + \sin\epsilon\ \sin\alpha\ \tan\delta)\Delta\lambda_n - (\cos\alpha\ \tan\delta)\Delta\epsilon \tag{10}$$
$$\Delta\delta_n = (\sin\epsilon\ \cos\alpha)\Delta\lambda_n + (\sin\alpha)\Delta\epsilon \tag{11}$$

We also consider the effects of aberration, as presented in [13]. Given the constant of aberration, $\kappa = 20.29552$ arcseconds, the true longitude of the Sun λ_s, eccentricity of the Earth's orbit e, and the longitude of the perihelion, ρ, we can compute the corrections $\Delta\alpha_a, \Delta\delta_a$ for the star's right ascension and declination with the following equations:

$$\Delta\alpha_a = -\kappa\left(\frac{\cos\alpha\ \cos\lambda_s\ \cos\epsilon + \sin\alpha\ \sin\lambda_s}{\cos\delta}\right) + e\kappa\left(\frac{\cos\alpha\ \cos\rho\ \cos\epsilon + \sin\alpha\ \sin\rho}{\cos\delta}\right) \tag{12}$$
$$\begin{aligned} \Delta\delta_a = &-\kappa\left[\cos\lambda_s\ \cos\epsilon(\tan\epsilon\ \cos\delta - \sin\alpha\ \sin\delta) + \cos\alpha\ \sin\delta\ \sin\lambda_s\right] \\ &+ e\kappa\left[\cos\rho\ \cos\epsilon(tan\epsilon\ \cos\delta\ - \sin\alpha\ \sin\delta) + \cos\alpha\ \sin\delta\ \sin\rho\right] \end{aligned} \tag{13}$$

where:

$$e = 16.708634 \times 10^{-3} - 42.037t \times 10^{-6} - 0.1267t^2 \times 10^{-6} \tag{14}$$
$$\rho = 102.9375 + 1.71946t + 0.46t^2 \times 10^{-3} \tag{15}$$

and the true longitude of the Sun is calculated as:

$$\lambda_s = L_0 + C \tag{16}$$

where the mean longitude of the Sun, L_0, is given by:

$$L_0 = 280.46646 + 36000.76983t + 0.3032t^2 \times 10^{-3} \tag{17}$$

and the Sun's equation of the center, C, is given by:

$$C = \big[(1914.602 - 4.817t - 0.014t^2)\sin M + (19.993 - 0.101t)\sin 2M + 0.289\sin 3M\big] \times 10^{-3} \tag{18}$$

and the mean anomaly of the Sun, M, is given by:

$$M = 357.52911 + 35999.05029t - 0.1537t^2 \times 10^{-3} \tag{19}$$

The correction terms, $\Delta\alpha_a$, $\Delta\delta_a$, $\Delta\alpha_n$ and $\Delta\delta_n$ are added to α and δ, yielding the corrected celestial coordinates of the star.

2.2. Updating Position and Time

Conversion from terrestrial time (as measured with the Gregorian calendar) to celestial time (typically expressed in Julian days) is a necessary step in simulating celestial bodies. The BSC represents the location of stars with respect to the J2000 epoch. The conversion from Gregorian date to Julian day is detailed in [13]. The basic steps are shown in Algorithm 1.

Algorithm 1 Conversion from Gregorian date to Julian day

```
let Y be the current year                                   ▷ Integer, Gregorian
let M be the current month                                  ▷ Integer [1..12], Gregorian
let D be the current day                                    ▷ Decimal, Gregorian
if $M \leq 2$ then
    $Y = Y - 1$
    $M = M + 12$
end if
let A = $int\left(\frac{Y}{100}\right)$
let B =$2 - A + int\left(\frac{A}{4}\right)$
J = $int(365.25(Y + 4716)) + int(30.6001(M + 1)) + D + B - 1524.5$     ▷ Julian Days
```

This conversion expresses the current time with respect to the number of Julian days that have elapsed since 4716 BC. Furthermore, the J2000 epoch is expressed in relation to the Gregorian date 1 January 2000 and can be found by subtracting the Julian day at that time from the current Julian date:

$$J2000 = J_D - 2451545.0 \tag{20}$$

Sidereal local time, analogous to celestial longitude, can be calculated given WGS84 longitude and the current time of day expressed in decimal hours. Given longitude λ, Julian date J_D expressed in the J2000 epoch, and decimal hours H_D, the local sidereal time LST is given by [13]:

$$LST = 100.46 + (0.985647 J_D) + \lambda + 15 H_D \tag{21}$$

LST is typically limited to the range $[0, 360]$ degrees. The hour-angle ω of a celestial body can then be simply calculated from LST and right ascension α as:

$$\omega = LST - \alpha \tag{22}$$

The hour-angle of a body can be used to compute its azimuth and elevation, expressed in local NED coordinates. The local elevation θ of a celestial body given hour angle ω, declination δ and latitude ϕ is given by:

$$\theta = \operatorname{asin}(\sin\delta\ \sin\phi\ +\ \cos\delta\ \cos\phi\ \cos\omega) \tag{23}$$

Subsequently, using elevation from Equation (23), the local azimuth ψ is given by the following equation:

$$\psi = \operatorname{atan2}(\sin\omega\ ,\ \cos\omega\ \sin\phi\ - \tan\delta\ \cos\phi) + \pi \tag{24}$$

where the addition of π radians converts azimuth to a representation that is positive East of North [13].

The observed elevation of a celestial body is altered due to the effects of atmospheric refraction. Consequently, objects in the sky appear at a greater elevation than they would without the atmospheric effects. This effect is exaggerated at lower elevations (closer to the horizon), which leads to an angular displacement of up to 0.5°. For cameras in the visible light spectrum, the refractive distance R (expressed in arcminutes) can be approximated to within 4 arcseconds [14], given the following formula:

$$R = \frac{1.02}{\tan\left(\theta + \dfrac{10.3}{\theta + 5.11}\right)} \tag{25}$$

If a greater level of accuracy is required, alternative methods such as that seen in [15] can be used. The apparent elevation θ' is then given by:

$$\theta' = \theta + R \tag{26}$$

The inverse of this formula, Equation (27) [16], allows for the correction of observations:

$$R = \frac{1}{\tan\left(\theta' + \dfrac{7.31}{\theta' + 4.4}\right)} \tag{27}$$

Formulas (25) and (27) assume an atmospheric pressure of 1010 millibars, and an air temperature of 10 °C. According to [13], an approximate scale factor may be applied given pressure P at the Earth's surface, and air temperature T °C, given by the following formula:

$$\frac{P}{1010} \times \frac{283}{273 + T} \tag{28}$$

Finally, we map the celestial sphere onto a unit sphere for the purposes of image plane projection. Given the azimuth and elevation of a star, the corresponding unit vector in local NED coordinates is given by:

$$\begin{aligned} x &= \cos\psi \cos\theta' \\ y &= \sin\psi \cos\theta' \\ z &= -\sin\theta' \end{aligned} \tag{29}$$

For resource constrained systems, the rate at which these unit vectors are updated should be chosen relative to the precision required by the simulation. For reference, a geostationary camera with an update rate of 1 Hz will be accurate to within ±15 arcseconds (4.17×10^{-3} degrees). For an aircraft at a latitude of ±45 °, travelling East/West at mach 1, an update rate of 1Hz will be accurate to within ±30 arcseconds ($8.37^\circ \times 10^{-3}$).

2.3. Updating Attitude

Representing celestial bodies in the local NED frame of reference simplifies the transformation from aircraft attitude to camera attitude. Aircraft attitude should come either directly from the onboard attitude reference or from a simulator. The work presented here uses the open source Ardupilot toolchain. Attitude log data collected from real flights were used for this research, so as to correlate real images with their simulated counterparts. The roll p_a, pitch q_a and yaw r_a Euler angles of the aircraft in local NED coordinates were logged at 30 Hz. These Euler angles can be represented as a rotation matrix, through a yaw–pitch–roll rotation sequence. The rotation matrix, $\boldsymbol{C_{a/l}}$, transforms objects in the local NED frame to the aircraft body frame, where $\boldsymbol{C_{a/l}}$ is given by:

$$\begin{bmatrix} \mathrm{c}(q_a)\mathrm{c}(r_a) & \mathrm{c}(q_a)\mathrm{s}(r_a) & -\mathrm{s}(q_a) \\ -\mathrm{c}(p_a)\mathrm{s}(r_a) + \mathrm{s}(p_a)\mathrm{s}(q_a)\mathrm{c}(r_a) & \mathrm{c}(p_a)\mathrm{c}(r_a) + \mathrm{s}(p_a)\mathrm{s}(q_a)\mathrm{s}(r_a) & \mathrm{s}(p_a)\mathrm{c}(q_a) \\ \mathrm{s}(p_a)\mathrm{s}(r_a) + \mathrm{c}(p_a)\mathrm{s}(q_a)\mathrm{c}(r_a) & -\mathrm{s}(p_a)\mathrm{c}(r_a) + \mathrm{c}(p_a)\mathrm{s}(q_a)\mathrm{s}(r_a) & \mathrm{c}(p_a)\mathrm{c}(q_a) \end{bmatrix} \tag{30}$$

where $\mathrm{c}(x)$ and $\mathrm{s}(x)$ represent $\cos x$ and $\sin x$, respectively. The camera is mounted to the aircraft, with roll p_c, pitch q_c and yaw r_c expressed in the aircraft body frame. The z axis of the camera is parallel to the optical axis, positive in the direction of the image plane. The y and x axes are orthogonal and directed towards the bottom and the right of the image plane, respectively. The rotation matrix, $\boldsymbol{C_{c/a}}$, transforms objects in the aircraft body frame to the camera frame, where $\boldsymbol{C_{c/a}}$ is given by:

$$\begin{bmatrix} \mathrm{c}(q_c)\mathrm{c}(r_c) & \mathrm{c}(q_c)\mathrm{s}(r_c) & -\mathrm{s}(q_c) \\ -\mathrm{c}(p_c)\mathrm{s}(r_c) + \mathrm{s}(p_c)\mathrm{s}(q_c)\mathrm{c}(r_c) & \mathrm{c}(p_c)\mathrm{c}(r_c) + \mathrm{s}(p_c)\mathrm{s}(q_c)\mathrm{s}(r_c) & \mathrm{s}(p_c)\mathrm{c}(q_c) \\ \mathrm{s}(p_c)\mathrm{s}(r_c) + \mathrm{c}(p_c)\mathrm{s}(q_c)\mathrm{c}(r_c) & -\mathrm{s}(p_c)\mathrm{c}(r_c) + \mathrm{c}(p_c)\mathrm{s}(q_c)\mathrm{s}(r_c) & \mathrm{c}(p_c)\mathrm{c}(q_c) \end{bmatrix} \tag{31}$$

Given rotation matrices $\boldsymbol{C_{a/l}}$ and $\boldsymbol{C_{c/a}}$, the transformation of a unit vector, $\boldsymbol{u}$, from the local NED frame to the camera frame, can be computed as:

$$\boldsymbol{u^c} = \boldsymbol{C_{c/a}}\boldsymbol{C_{a/l}}\boldsymbol{u} \tag{32}$$

Equation (32) allows the unit vectors of the celestial bodies computed in Section 2.2 to be represented in the camera frame of reference.

2.4. Projection

We assume that the camera intrinsic matrix, $\boldsymbol{K}$, is known. The intrinsic properties of a camera can be found through a calibration method such as that described in [17], yielding matrix:

$$\boldsymbol{K} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{33}$$

where f_x and f_y are the x and y focal lengths in pixel units, x_0 and y_0 are the x and y pixel locations of the principal point, respectively, and s describes the sensor skewness (typically 0 for digital sensors).

We first convert each celestial body unit vector into homogeneous coordinates. For components x, y and z of unit vector $\boldsymbol{u^c}$, the homogeneous point $\boldsymbol{P}$ in the camera frame of reference is calculated as:

$$\boldsymbol{P} = \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix} \tag{34}$$

Objects whose depth value, z, is less than 0 are ignored due to being positioned behind the camera object. Finally, given no translational component to our camera, the pixel location, $\boldsymbol{x}$, of the object is found as:

$$\boldsymbol{x} = \boldsymbol{KP} \tag{35}$$

Lens distortion may also be included in the model. Lens distortion models are typically non-linear, and expressed as a function of displacement from the principal point. Various models exist for lens distortion, and should be chosen according to the level of precision required [18]. If using a lens distortion model, this should be applied after the star is rendered, so as to capture the resultant eccentricities from the distortion. We do not model lens distortion in our simulation; instead, we rectify all images prior to analysis, such that any residual distortion is negligible.

2.5. Calibration

The apparent pixel intensity of a star is determined by the apparent star magnitude, atmospheric conditions, lens properties and sensor properties. We present here a method which precludes the need for detailed modeling, through a single-image calibration process. We use a reference image captured from a stationary aircraft on the ground and fit an exponential curve to define the relationship between star magnitude and pixel intensity.

The relationship between observed brightness and apparent star magnitude is given by the equation:

$$m_x - m_r = -2.5\ \log_{10}\left(\frac{B_x}{B_r}\right) \tag{36}$$

where m_x is the observed star magnitude, m_r is the reference star magnitude, B_x is the observed star brightness and B_r is the reference star brightness. This magnitude scale is designed such that a magnitude difference of -5 correlates with a brightness factor of 100. That is to say, a magnitude 1 star is 100 times brighter than a magnitude 6 star. For this work, we take the brightness of a star to be its maximum pixel value. Images are converted from the blue–green–red (BGR) colour space to the hue–saturation–value (HSV) colour space, and the value channel is used as a greyscale image.

By rearranging Equation (36) we can compute the brightness of an observed star, given that we have a reference brightness B_r, reference magnitude m_r and observed magnitude m_x:

$$B_x = B_r\, 10^{\frac{m_x - m_r}{2.5}} \tag{37}$$

The choice of reference star is an important factor, as the magnitude of stars are typically considered to be unreliable [19]. Many factors can cause the apparent magnitude of a star to differ from the catalogue, including spectral attenuation caused by the atmosphere, camera characteristics, atmospheric refraction, as well as the luminescent characteristics of the star itself. We assume that the magnitude error follows a zero-mean Gaussian distribution. Following this assumption, we select the star with magnitude and brightness that minimizes the Frobenius norm of the difference between observed and calculated brightness:

$$min\left([\sum_i abs(B_i' - B_i)^2]^{1/2}\right) \tag{38}$$

where B_i' is a vector containing all calculated star brightnesses, and B_i is a vector containing all measured star brightnesses. The vector B_i' is computed for each star in the reference image by choosing B_r and m_r from the reference star, and recomputing B_x for all stars in the image using Equation (37). Stars whose brightness is saturated (i.e., have a maximum pixel value of 255) are excluded from this process. Figure 2 shows the output from this calibration process, conducted on an image captured from a grounded aircraft, using a PiCamHQ with 500 ms exposure interval. We note that this procedure is most effective with a larger number of visible stars, such that the sample better represents the population.

In this example, stars were automatically detected and matched to the database using a star identification algorithm. As an alternative to this automated process, one could manually label each star in the reference image.

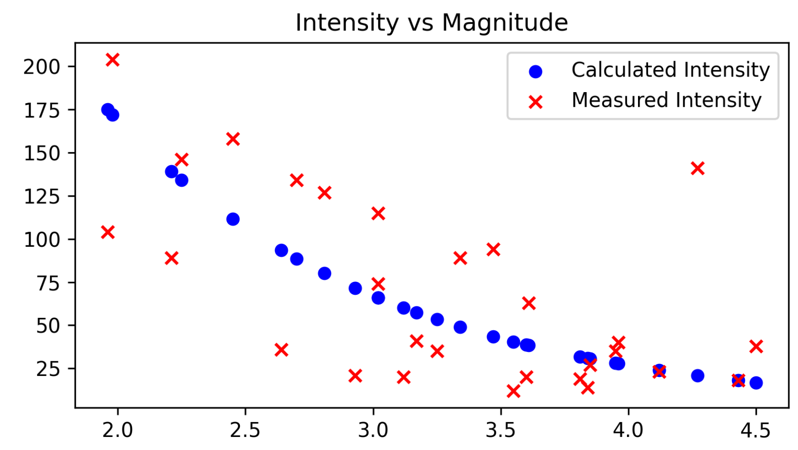


Figure 2. Calibration curve of image brightness using a PiCamHQ, 500 ms exposure time.

Stars are effectively point light sources. The camera lens will tend to defocus a star, such that it appears larger than one pixel. The apparent size of a star is affected by magnitude only to the extent that otherwise-undetectable pixels exceed the noise floor. Consequently, we can treat each star as having constant size, and changes to the star's brightness will yield a larger or smaller effective area. We assume that the pixel intensity of a star follows a two-dimensional Gaussian point spread function, with diagonal covariance matrix whose elements are equal. That is, a star image is circularly symmetric about its center. In practice, this may not be the case, and we note that such methods might be used for calibrating lens and sensor distortion (this is, however, out of scope for this work). Following this assumption, we measure the standard deviation across the x and y axes of each normalized star detection in the reference image and use the median value across all stars as the reference size for rendering in the simulation. We normalize by scaling the peak pixel value to 1. The standard deviation σ_B is calculated as:

$$\sigma_B = \frac{1}{n}\sum_n \operatorname{std}\left(\mathbf{B}_n \odot \frac{1}{\max(\mathbf{B}_n)}\right) \tag{39}$$

where $\mathbf{B}_n$ is the histogram of intensities with respect to pixel position, $\operatorname{std}(x)$ is the standard deviation of sample set x, the symbol $\odot$ represents element-wise multiplication, and $\max(\mathbf{B}_n)$ is a scalar value equal to the maximum-valued element of $\mathbf{B}_n$.

A graph showing the standard deviation in pixel intensity vs. star magnitude can be seen in Figure 3, demonstrating the approximate uniformity of apparent projected star size across various magnitudes. Figure 4 shows an example of a Gaussian star render, reconstructed from the standard deviation and star brightness.

Simulation noise levels are calibrated from the reference image. Sources such as moonlight and atmospheric light pollution contribute to Gaussian noise observed in an image and consequently reduce the signal-to-noise ratio and observability of stars. The mean and standard deviation is measured in the value channel of the image. We ignore sections of the image in which stars have been detected so as to avoid bias introduced by the stars themselves. Thus, we have the noise function:

$$X \sim N(\mu, \sigma^2) \tag{40}$$

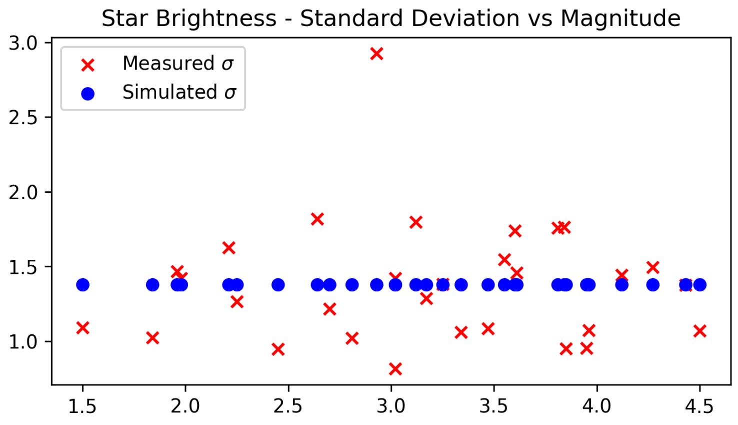


Figure 3. Calibration of standard deviation in star brightness using a PiCamHQ, 500 ms exposure time.

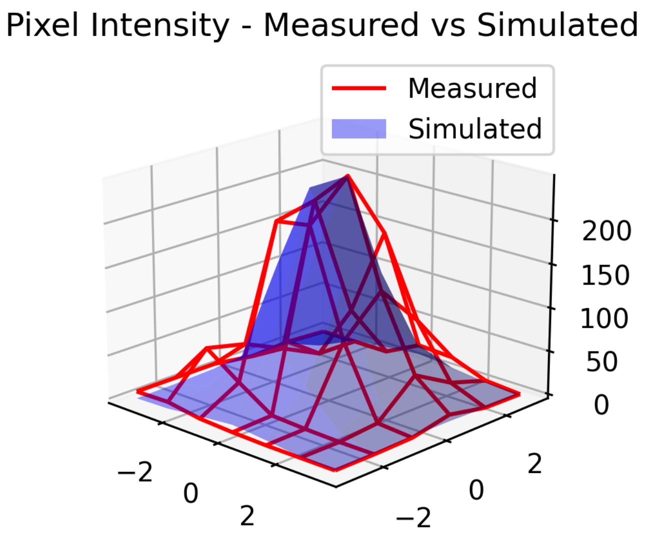


Figure 4. Pixel intensity of a measured star (red wireframe), with the simulated pixel intensity overlaid (blue solid).

2.6. Rendering

A long exposure image can be generated by superimposing multiple short-exposure images. Doing so requires a fast attitude update so as to reproduce the motion of the aircraft throughout the exposure period. The required temporal resolution is affected by the aircraft dynamics; however we found that a 5 ms (200 Hz) attitude update results in the contiguous rendering of stars for wide-angled lenses, even during aggressive maneuvers. A 200 Hz attitude measurement is not typically available on low-cost hardware, so we interpolate attitude measurements using the spherical linear interpolation algorithm described in [20] to achieve the desired rate. This method requires orientations to be represented as quaternions. The details are omitted here for brevity; however, most computer graphics libraries support this operation, transforming the direction cosine matrix in Equation (32) to quaternion format. We can retrieve any arbitrary orientation between two quaternions with the following equation:

$$\mathrm{Slerp}(q_1, q_2, u) = q_1 \left(q_1^{-1} q_2\right)^u, \{u \in \mathbb{R} : 0 \le u \le 1\} \tag{41}$$

where Slerp is the spherical linear interpolation function; a value of $u = 0$ returns q_1, a value of $u = 1$ returns q_2 and intermediate values of u provide interpolation along the shortest path between q_1 and q_2 on the unit sphere. Interpolation is performed such that attitudes are captured at 5 ms intervals from the beginning of the camera exposure interval, to the end, yielding a total of $n = 200\Delta t$ attitude references, where Δt is the camera exposure time in seconds.

The long-exposure image is constructed by superimposing n floating point images. For each attitude a_i, the pixel location $\mathbf{x}$ of each star S_j in the database is found from Equation (35). A discrete Gaussian kernel, $\boldsymbol{G}$ is constructed using the standard deviation found in Equation (39). The kernel is programmatically generated (see [21]) such that the value G_i of element i is given by:

$$G_i = \alpha \exp\left(\frac{-(i - \frac{(k-1)}{2})^2}{(2\sigma^2)}\right), \{i \in \mathbb{Z} : 0 \leq i < k\} \tag{42}$$

where the kernel size $k = \lceil 6\sigma^2 \rceil$ is odd and is selected to contain a minimum of 99.7% of the total star energy, and α is selected such that $\sum_i G_i = 1$. The kernel is subsequently scaled, such that the maximum element at index $i = \frac{k-1}{2}$ is equal to the peak pixel value, B_x, calculated from Equation (37) using the magnitude m_x of star S_j, as well as the reference magnitude and intensities, m_r and B_r respectively. Finally, the kernel is scaled down by a factor of n. Assuming the photon flux density is constant across the exposure window, a single short-exposure window contains a fraction $\frac{1}{n}$ of the total star energy. This scaling of kernel G generated in Equation (42) is given by:

$$G_i = G_i\left(\frac{B_x}{n * G_{max}}\right), \{i \in \mathbb{Z} : 0 \leq i < k\} \tag{43}$$

Note that $\boldsymbol{G}$ is stored as a floating point array. The short-exposure star is drawn to the image canvas by centering the kernel $\boldsymbol{G}$ on pixel $\boldsymbol{x}$ and rotating 180° about the centre, adding the values of $\boldsymbol{G}$ to the canvas, so as to render the star symmetrically. This process is repeated for each star in the database S_j at the current attitude.

Once each of the short-exposure images have been rendered, the Gaussian noise defined in Equation (40) is added to the canvas. Finally, the image is converted from a single-channel floating point image to an 8-bit single-channel image. The resulting image contains stars rendered with the motion blur caused by camera movement throughout the exposure interval.

3. Results

We compare here the simulation output with images captured during a flight to evaluate the performance of the simulator. Attitude logs from the flight test were recovered, and used to generate these simulation images. We followed the procedure outlined in Section 2 for image generation. We used a Phantom FX-61 airframe (Figure 5), with a PixHawk v2 autopilot, a Raspberry Pi 4 companion computer for image capture and storage and a Raspberry Pi High Quality Camera sensor, mounted with the official 6mm wide-angle lens. The camera was rigidly mounted to the autopilot, so as to mechanically couple sources of vibration and deflection. An approximate transform from aircraft body frame to camera frame is used for this test, at a yaw angle of −90°, pitch angle of 90° and roll angle of 0° (given a yaw-pitch-roll Euler sequence).

A flight was conducted capturing images every 3 s, with an exposure interval of 500 ms and ISO set to 800. Ground images were captured prior to launching the aircraft. The aircraft climbed to an altitude of 150 m above ground level, and loitered for several minutes before landing. A total of 130 images were captured for analysis. Attitudes throughout each exposure window were recorded as image metadata at a rate of 30 Hz, and retrieved from the flash logs after the flight. A ground image was selected for performing the simulation calibration outlined in Section 2.5. Simulation images were subsequently generated using these calibration parameters.

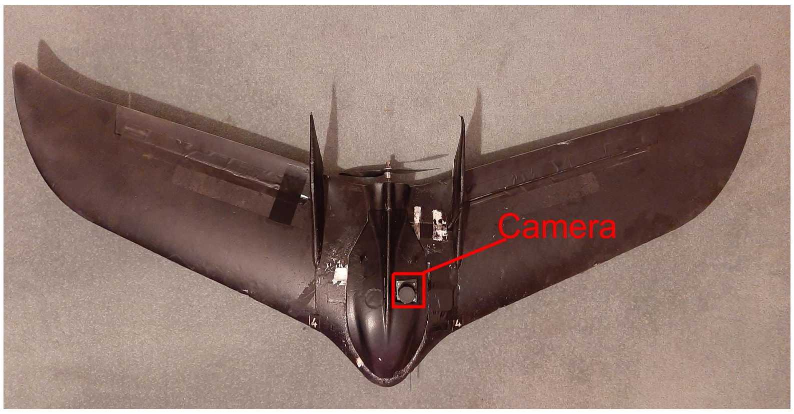


Figure 5. Phantom FX-61 with camera mounted in fuselage.

A visual comparison of simulation output against in-flight images can be seen in Figures 6 and 7. By observation, we can see that the shape and trajectory of the simulated light trail closely matches reality. Figure 6 shows a cluster of stars, captured while the aircraft was yawing at a rate of 8° per s. The difference in star trail direction between stars is due to alignment of the yaw axis, approximately directed toward the centre of the image. Figure 7 pictures the brightest visible stars in the image, highlighting the effectiveness of the spherical linear interpolation, and its ability to reproduce trajectories with visual magnitude similar to what is observed in reality.

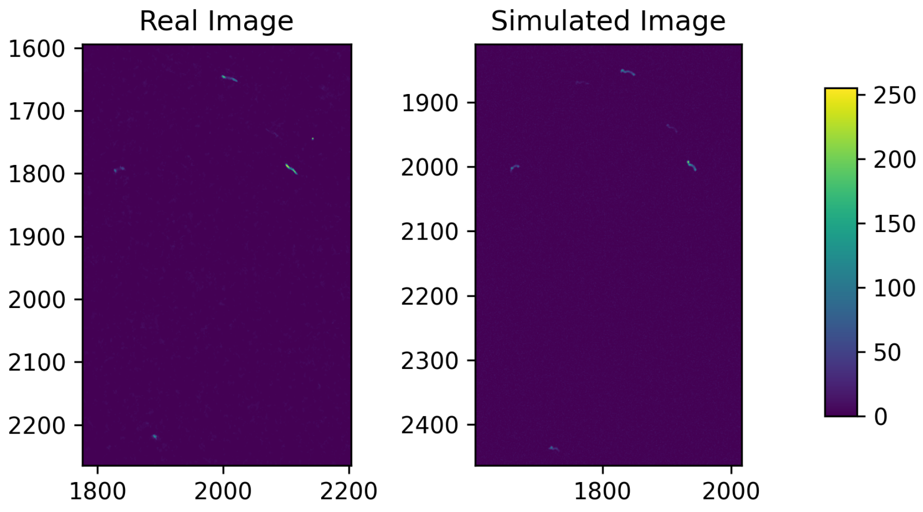
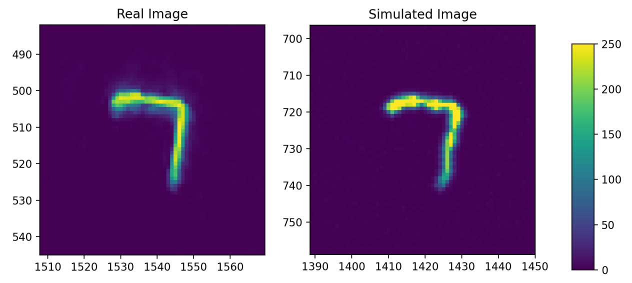
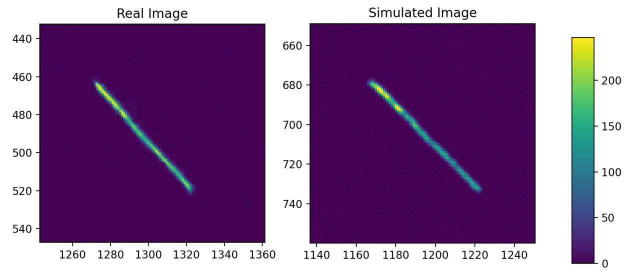


Figure 6. Lower magnitude stars captured in-flight (4× increased brightness for display purposes).

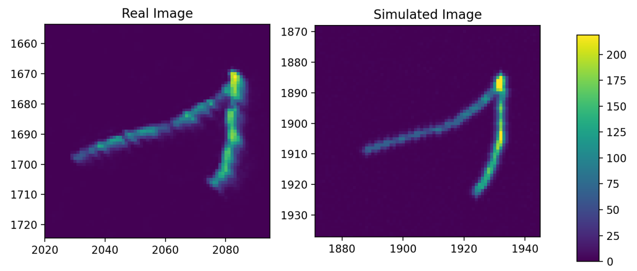
We evaluate the performance of the simulator based on its ability to replicate in-flight star intensities, given a ground calibration image. Figure 8 shows individual stars which were identified across multiple images in-flight. The intensity of the simulated star was plotted against the intensity of the observed star, such that points lying on the line $y = x$ represent a perfect match between reality and simulation. Three stars are shown here; these stars were identified frequently in both reality and simulation. We can see that, in practice, there tends to be error in the pixel intensity. The mean error is near-zero with a value of 3.38 pixels. The mean absolute percentage error in pixel intensity is measured at 47.4% (this is an average of per-star absolute percentage errors), which is similar to the mean absolute percentage error of 51% in the brightness ground calibration, as expected. Using Equation (36), a 47.4% error in star magnitude correlates to an absolute magnitude error of 0.42. This is within comparable range to the noise level simulated in other works, such as [7,8], which artificially add magnitude noise with standard deviation in the range of 0.3–0.9 to their simulation.



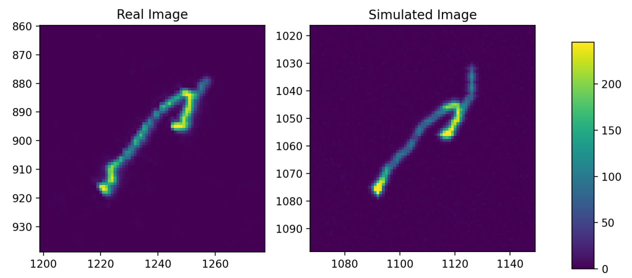
(**a**)



(**b**)

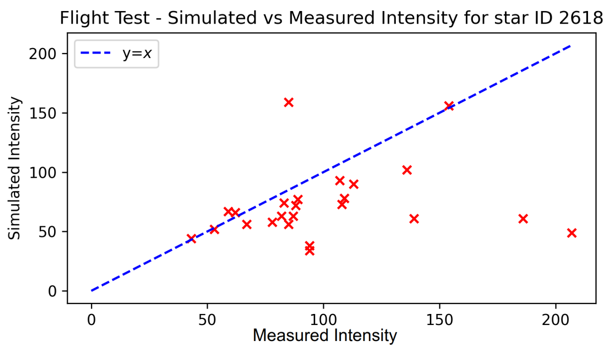


(**c**)

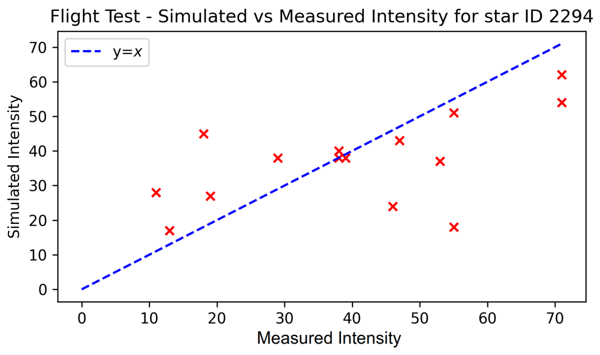


(**d**)

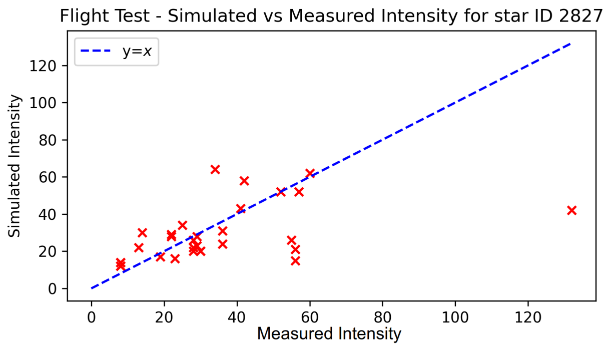
Figure 7. Close-up of bright stars captured in motion. Real images (left) are paired with their simulated counterpart (right). (**a**) Image captured during aerial manoeuvre; (**b**) Image captured during constant-rate turn; (**c**) Image captured with high pitch-rate; (**d**) Image captured during aerial manoeuvre.



(**a**)



(**b**)



(**c**)

Figure 8. Individual stars over multiple images, comparing measured intensity to simulated intensity. Red crosses indicate the peak pixel intensity for a given observation. (**a**) Visual magnitude: 1.5, mean error: −23.6, mean percentage error: 37.5%; (**b**) Visual magnitude: 2.25, mean error: 12.2, mean percentage error: 82.3%; (**c**) Visual magnitude: 2.45, mean error: −6.3, mean percentage error: 38.0%.

Furthermore, we consider the difference between the centroids of stars detected in both simulation and reality. A region of interest (ROI), $\boldsymbol{R}$, is chosen for each star such that $\boldsymbol{R}$ is the smallest grayscale image that contains the star. We compute the weighted centre D_x, D_y for both real and simulated images, expressed with respect to the centre of the ROI:

$$\begin{bmatrix} D_x \\ D_y \end{bmatrix} = \begin{bmatrix} \frac{w}{2} \\ \frac{h}{2} \end{bmatrix} - \frac{1}{M} \sum_{x,y} R_{x,y} \begin{bmatrix} x \\ y \end{bmatrix} \tag{44}$$

$$M = \sum_{x,y} R_{x,y} \tag{45}$$

where w and h are the width and height of the region of interest, respectively, x and y are the column and row indices of image R and scalar $R_{x,y}$ is the pixel intensity of $\boldsymbol{R}$ at pixel (x, y).

We compute the L2 norm of the difference between simulation centroid and real centroid to find the distance L:

$$L = \sqrt{(D_x^s - D_x^r)^2 + (D_y^s - D_y^r)^2} \tag{46}$$

for simulation centroid D_x^s, D_y^s and real centroid D_x^r, D_y^r. A histogram containing the computed centroid errors can be seen in Figure 9. For reference, we also compute a baseline estimate, which assumes the centroid is located at the centre of the ROI (analagous to simulating stars as a straight line with uniform intensity). The mean simulation error is measured to be 0.92 pixels, with a median error of 0.68 pixels. By comparison, the mean baseline error is measured at 1.23 pixels, with a median error of 0.93 pixels. This corresponds to a 25.2% reduction in mean centroid error.

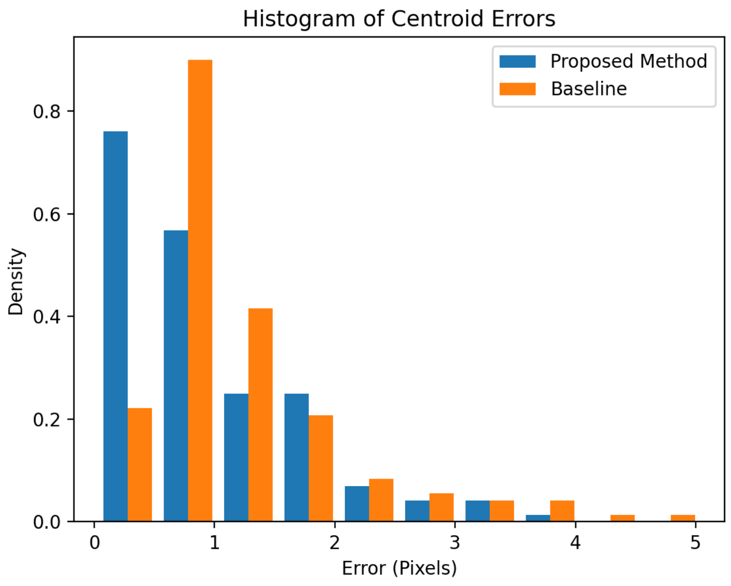


Figure 9. Histogram of absolute differences in star centroids between real and simulated images.

4. Discussion

The temporal correlation between the camera and attitude sensor, as well as the attitude sensor's accuracy and resolution, and individual differences between the database and observed star magnitudes, pose limitations to the accuracy with which simulation images can be generated. The tendency towards a low mean error, however, is an indication that there is little systematic error propagating from the simulation architecture, and that large sample sets will provide a statistically accurate representation of star intensity.

We can see in Figure 8 that despite a low mean error across the sample set, individual stars tend to be subjected to biases in intensity. Future work could make use of multiple ground images to map the intensity of individual stars, so as to reduce bias within individual stars. Furthermore, it is apparent that brighter stars will tend to be detected more frequently than dimmer stars. One may be able to determine an appropriate magnitude threshold for the observability of stars in-flight and bias the calibration towards the brighter stars, which are more likely to be detected.

We note that the ground calibration process is effective only for low-altitude applications. The simulation does not account for changes in atmospheric attenuation due to altitude. Higher-altitude flight will also result in changes to atmospheric refraction. Furthermore, the ground calibration process is subjected to light pollution, which again is a function of altitude. The simulation of higher altitude flight should make use of atmospheric models to account for these disparities between ground and high-altitude observations. The level of image noise due to moonlight is assumed to be constant here; however, in practice there is some degree of variation as the viewing angle changes with respect to the position of the moon. This is most noticeable within a 5° viewing angle [22], but less significant at greater angles.

Validation of this simulation was conducted with a fixed wing aircraft; however, the simulation architecture is applicable for any airframe which is capable of reporting its attitude. This might also be used for simulating motion artefacts from two-axis gimbals. While the Raspberry Pi Camera HQ is fit with a rolling shutter, long exposure images are achieved by a series of shorter exposures, similar to the process followed in this simulation. The intra-frame motion is not captured by this simulation; however, this effect appears to be negligible. If the characteristics of a rolling shutter are known, one could replicate this effect by interpolating at a faster rate and selecting an appropriate attitude given the time at which the shutter exposes the star. It is common, however, for charged-couple device (CCD) cameras to be used for celestial imaging. These cameras utilize a global shutter, which exposes all pixels simultaneously and hence are not subjected to the rolling shutter effect.

5. Conclusions

The intent of this work was to design and validate a simulation architecture to support the development of strapdown celestial navigation solutions in lightweight, low-altitude aircraft. An architecture for replicating the effects of long-exposure imagery was designed and implemented by superimposing multiple short-exposure images from aircraft attitude data. Additionally, a method for augmenting low-rate attitude data was proposed and validated. Simulation calibration was achieved through a single ground reference image, producing results which match reality within reasonable tolerance. The simulation architecture provides a tool for baseline testing star detection and identification algorithms. Future work will extend the capability of this simulator from low-altitude to high altitude through atmospheric modeling.

Author Contributions: Conceptualization, S.T. and J.C.; methodology, S.T.; software, S.T.; validation, S.T.; formal analysis, S.T.; investigation, S.T.; resources, J.C.; data curation, S.T and J.C.; writing—original draft preparation, S.T.; writing—review and editing, J.C.; visualization, S.T.; supervision, J.C.; project administration, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by Scope Global Pty Ltd. under the Commonwealth Scholarships Program, and the Commonwealth of South Australia under the Australian Government Research Training Program.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kayton, M.; Fried, W.R. *Avionics Navigation Systems*; John Wiley & Sons: Hoboken, NJ, USA, 1997.
2. Gao, Z.; Wang, H.; Wang, W.; Xu, Y. SIMU/Triple star sensors integrated navigation method of HALE UAV based on atmospheric refraction correction. *J. Navig.* **2022**, *75*, 1–23. [CrossRef]
3. Ting, F.; Xiaoming, H. Inertial/celestial integrated navigation algorithm for long endurance unmanned aerial vehicle. *Acta Tech. CSAV (Ceskoslov. Akad. Ved.)* **2017**, *62*, 205–217.

4. Titterton, D.; Weston, J.L.; Weston, J. *Strapdown Inertial Navigation Technology*; IET: London, UK, 2004; Volume 17.
5. Rijlaarsdam, D.; Yous, H.; Byrne, J.; Oddenino, D.; Furano, G.; Moloney, D. A survey of lost-in-space star identification algorithms since 2009. *Sensors* **2020**, *20*, 2579. [CrossRef] [PubMed]
6. Padgett, C.; Kreutz-Delgado, K.; Udomkesmalee, S. Evaluation of star identification techniques. *J. Guid. Control. Dyn.* **1997**, *20*, 259–267. [CrossRef]
7. Xu, L.; Jiang, J.; Liu, L. RPNet: A representation learning-based star identification algorithm. *IEEE Access* **2019**, *7*, 92193–92202. [CrossRef]
8. Wei, X.; Wen, D.; Song, Z.; Xi, J.; Zhang, W.; Liu, G.; Li, Z. A star identification algorithm based on radial and dynamic cyclic features of star pattern. *Adv. Space Res.* **2019**, *63*, 2245–2259. [CrossRef]
9. Han, J.; Yang, X.; Xu, T.; Fu, Z.; Chang, L.; Yang, C.; Jin, G. An End-to-End Identification Algorithm for Smearing Star Image. *Remote Sens.* **2021**, *13*, 4541. [CrossRef]
10. Haiyong, W.; Wenrui, Z.; Cheng, X.; Haoyu, L. Image smearing modeling and verification for strapdown star sensor. *Chin. J. Aeronaut.* **2012**, *25*, 115–123.
11. Yan, J.; Jiang, J.; Zhang, G. Dynamic imaging model and parameter optimization for a star tracker. *Opt. Express* **2016**, *24*, 5961–5983. [CrossRef] [PubMed]
12. Wang, Z.; Jiang, J.; Zhang, G. Global field-of-view imaging model and parameter optimization for high dynamic star tracker. *Opt. Express* **2018**, *26*, 33314–33332. [CrossRef] [PubMed]
13. Meeus, J. *Astronomical Algorithms*; Richmond: Richmond, VA, USA, 1991.
14. Saemundsson, T. Atmospheric refraction. *Sky Telesc.* **1986**, *72*, 70.
15. Wang, Z.; Jiang, J. Refraction Surface-Based Stellar Atmospheric Refraction Correction and Error Estimation for Terrestrial Star Tracker. *IEEE Sens. J.* **2022**, *22*, 9685–9696. [CrossRef]
16. Bennett, G. The calculation of astronomical refraction in marine navigation. *J. Navig.* **1982**, *35*, 255–259. [CrossRef]
17. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
18. Tang, Z.; Von Gioi, R.G.; Monasse, P.; Morel, J.M. A precision analysis of camera distortion models. *IEEED* **2017**, *26*, 2694–2704. [CrossRef] [PubMed]
19. Zhang, G. *Star Identification: Methods, Techniques and Algorithms*; Springer: Berlin/Heidelberg, Germany, 2016.
20. Shoemake, K. Animating rotation with quaternion curves. In Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques, San Francisco, CA, USA, 22–26 July 1985; pp. 245–254.
21. Itseez. The OpenCV Reference Manual. Release 2.4.9.0. 2014. Available online: https://docs.opencv.org/2.4.9 (accessed on 15 June 2022).
22. Krisciunas, K.; Schaefer, B.E. A model of the brightness of moonlight. *Publ. Astron. Soc. Pac.* **1991**, *103*, 1033. [CrossRef]