A HIGH-LEVEL MOTION CONTROL SCHEME FOR SATELLITE TRACKING USING OPTICAL FEEDBACK

Silas Fiore, Alessandro Vananti, and Thomas Schildknecht

Astronomical Institute University of Bern (AIUB), Sidlerstrasse 5, 3012 Bern, Switzerland,
Email: silas.fiore@unibe.ch, alessandro.vananti@unibe.ch, thomas.schildknecht@unibe.ch

ABSTRACT

The attitude state of defunct resident space objects (RSO) is becoming of increasing importance due to advances in active debris removal (ADR) capabilities. Photometric observations can be used to assess the attitude mode of suitable targets for ADR. The measured intensity of the sunlight reflected by the object can contribute to detecting periodicity and inferring the orientation of highly reflective surfaces. To resolve specular glints we want to use a single photon avalanche diode (SPAD) for these measurements. This sensor requires the observed object to be centered in the field of view (FOV) of the telescope. To this end, an optical tracking framework is developed that uses a CMOS camera for precise tracking of RSOs. A high-level controller is used to generate velocity inputs to the telescope axes based on ephemerides and optical feedback. A comprehensive overview of the key algorithms and models used is given and preliminary results are analyzed.

Keywords: active tracking, photometry.

1. INTRODUCTION

Photometric observations of RSOs can be used to characterize or fully determine their attitude state. There are several methods to acquire these measurements that depend on the orbital regime of the objects, the FOV and aperture of the telescope and the imaging sensor used. A fundamental distinction can be made between methods that compensate for the relative motion of the object in the image processing and methods that require the telescope to be guided along the path of the object in the sky. The latter type has been used by our institute for more than 10 years to acquire light-curves of space debris. These observations require smooth tracking but small pointing errors along-track and cross-track of the orbit are still compensated for in the image reduction. For sensors such as photomultiplier tubes or SPADs which measure the photon flux incident on a single pixel this is not an option and all pointing errors have to be compensated by the controller that guides the telescope along the trajectory of the observed object. To measure the pointing error a guiding camera is used in parallel to the sensor and corrections are computed in real-time and combined with the ephemeris of the object to control the telescope. In this paper we give an overview of a system that achieves this using a standard interface available on many modern telescopes. A C program was developed that controls our Andor Zyla 5.5 sCMOS camera and sends velocity commands to the telescope. The telescope used for this development is the 80 cm ZimMain telescope at the Swiss Optical Ground Station and Geodynamics Observatory in Zimmerwald. The CMOS sensor is attached to the 5600 mm focal station at one of the two nasmyth ports of the Ritchey-Chrétien altitude-azimuth telescope. In this configuration it covers a field of view (FOV) of approximately 10.2’$\times$ 8.6’. The program features a graphical user interface (GUI) that displays the frames as they are being acquired, allowing an observer to identify and select a target. Two threads control the camera and the telescope independently at first until an optical feedback mode is enabled. In this mode the observed deviation of the object from the center of the FOV is used instead of encoder positions to guide the telescope. The remainder of this paper is structured as follows: First, the generation of ephemerides and the necessary coordinate systems are introduced. Then the high-level control scheme for an alt-az mount is explained. System identification of the two axes using sinusoid velocity inputs is addressed, as it forms the basis for the tuning of the controller, simulation, and testing. Next the real-time image processing is adressed, which consists of tracking the region of interest (ROI) within the frame and detecting when the object is lost. The two topics of telescope control and image processing are then combined to achieve active tracking, which stands for the correction of the telescope trajectory during tracking based on the observed error with respect to the ephemeris. This includes a description of the operating modes of the camera and the framework developed to couple camera and telescope. Finally, some preliminary results are presented and the drawbacks and advantages of the method are discussed.

2. EPHEMERIS-ONLY TRACKING

To observe an RSO an ephemeris that predicts the position and velocity as a function of time is required. We use Two Line Element Sets in combination with the Simplified General Perturbation (SGP) propagator which enables us to compute the trajectory in real-time. Both the SGP library and a catalogue of TLEs are available online [1],[2]. The propagated position and velocity vectors obtained by this model are in a true equator mean equinox (TEME) frame. They are transformed to a topocentric spherical coordinate system that is aligned with the axes of the telescope. The TEME vectors are transformed to an earth centered earth fixed frame (ECEF) by applying two transformations that account for the rotation of the earth and for the polar motion. An intermediate so-called pseudo earth fixed frame (PEF) is reached when only the Earth rotation is taken into account. The transformation from ECEF to PEF

$$\mathbf{r}_{\text{pef}} = \mathbf{ST}^{\text{T}} \cdot \mathbf{r}_{\text{teme}} \tag{1}$$

$$\mathbf{v}_{\text{pef}} = \mathbf{ST}^{\text{T}} \cdot \mathbf{v}_{\text{teme}} - \boldsymbol{\omega} \times \mathbf{r}_{\text{pef}} \tag{2}$$

makes use of the sidereal time matrix $\mathbf{ST}$ as well as the angular velocity of the PEF frame with respect to the TEME frame $\boldsymbol{\omega}$.

$$\mathbf{ST} = \begin{bmatrix} \cos(\theta_{\text{gmst}}) & -\sin(\theta_{\text{gmst}}) & 0 \\ \sin(\theta_{\text{gmst}}) & \cos(\theta_{\text{gmst}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \boldsymbol{\omega} = \begin{bmatrix} 0 \\ 0 \\ \omega_{\oplus} \end{bmatrix}$$

The Greenwich mean sidereal time in radians θ_{gmst} can be computed from the universal time $UT1$ which is corrected for polar motion. The angular velocity of earth $\omega_{\oplus}$ is defined by the World Geodetic System 1984 and is corrected by the Length of Day (LOD) parameter. The earth orientation parameters x_p and y_p define the polar motion matrix

$$\mathbf{PM} = \begin{bmatrix} \cos(x_p) & 0 & -\sin(x_p) \\ \sin(x_p)\sin(y_p) & \cos(y_p) & \cos(x_p)\sin(y_p) \\ \sin(x_p)\cos(y_p) & -\sin(y_p) & \cos(x_p)\cos(y_p) \end{bmatrix}$$

which maps the PEF frame to the ECEF frame.

$$\mathbf{r}_{\text{ecef}} = \mathbf{PM}^{\text{T}} \cdot \mathbf{r}_{\text{pef}} \tag{3}$$

$$\mathbf{v}_{\text{ecef}} = \mathbf{PM}^{\text{T}} \cdot \mathbf{v}_{\text{pef}} \tag{4}$$

The coordinates of the object are transformed to the topocentric frame by subtracting the position of the observer. Then they are rotated to the north-east-down (NED) frame using the rotation matrix

$$\mathbf{R} = \begin{bmatrix} -\sin(\Phi)\cos(\Lambda) & -\sin(\Phi)\sin(\Lambda) & \cos(\Phi) \\ -\sin(\Lambda) & \cos(\Lambda) & 0 \\ -\cos(\Phi)\cos(\Lambda) & -\cos(\Phi)\sin(\Lambda) & -\sin(\Phi) \end{bmatrix}$$

which is a function of the geodetic latitude Φ and longitude Λ of the observer. The NED coordinates

$$\mathbf{r}_{\text{ned}} = \mathbf{R} \cdot \left(\mathbf{r}_{\text{ecef}} - \mathbf{r}_{\text{obs}}\right) \tag{5}$$

$$\mathbf{v}_{\text{ned}} = \mathbf{R} \cdot \mathbf{v}_{\text{ecef}} \tag{6}$$

are used to compute the azimuth ϕ and elevation λ of the RSO.

$$\phi = \text{atan2}\left(y_{\text{ned}}, x_{\text{ned}}\right) \tag{7}$$

$$\lambda = \text{atan2}\left(-z_{\text{ned}}, \sqrt{x_{\text{ned}}^2 + y_{\text{ned}}^2}\right) \tag{8}$$

The earth orientation parameters x_p, y_p, LOD and $(UT1 - UTC)$ are also made available on Celestrak.org [1].

2.1. Telescope Control using Velocity Inputs

The telescope features an Astronomy Common Object Model (ASCOM) interface. Using the ASCOM Alpaca API this interface can be used to control the telescope via TCP/IP. In this work all communication with the telescope happens through this interface but the concepts also apply for a more direct connection with the mount such as a serial interface with the motor controllers. In our case a program running on the dedicated PC that controls the mount communicates with the motor controllers through a serial interface and hosts the Alpaca API on the local network. For the work presented in this section only three methods of the interface are used. To read the encoders the *get azimuth* and *get altitude* methods are used. Optionally a mount model is applied and the elevation is corrected for refraction. To send velocity commands to both axes the *moveaxis* method is used which gives a velocity command in degrees per second to the specified axis. A proportional-integral (PI) controller is used to compute the velocity inputs for both axes from the errors between the target coordinates and the encoder angles. The controller is suited for this application since only the position is measured and adding a filtered derivative term typically amplifies the high-frequency gain of the sensitivity to measurement noise by an order of magnitude [4]. When the steady state angular velocity is added as a feedforward term to the output of the PI controller a constant velocity signal can be tracked with zero steady state error [5]. The high-level control law for both axes is given by equation 9. The proportional and integral gains are denoted by K_p and K_i, respectively. The angle θ_d and the angular velocity $\dot{\theta}_d$ are the desired values and θ is the measured angle. The angular rate $\dot{\theta}$ is the feedback control input to the axis.

$$\dot{\theta} = \dot{\theta}_d + K_p \cdot (\theta_d - \theta) + K_i \int_0^t (\theta_d - \theta) d\tau \tag{9}$$

For the integral term back-calculation is used to prevent windup in case the actuators saturate. Special care has to be taken to avoid hitting the axes limits at great speed or exceeding them. If the target angle θ_d is outside the allowed range it is clamped to the boundaries $[\theta_{min}, \theta_{max}]$. Near these limits, the reference angular rate $\dot{\theta}_d$ is modified to reach zero at the boundary. The desired behavior is a linear increase in breaking torque when a limit is approached at a given velocity. The maximum rate of

change of the acceleration J_{max} and the maximum velocity of the actuator $\dot{\theta}_{max}$ dictate the required braking distance.

$$\triangle T_{break} = \sqrt{\frac{2\dot{\theta}_{max}}{J_{max}}} \tag{10}$$

$$\triangle \theta_{break} = \dot{\theta}_{max} \cdot \triangle T_{break} - J_{max}\frac{\triangle T_{break}^3}{6} \tag{11}$$

Therefore, to start breaking early enough the upper and lower bound

$$\theta_{ub} = (\theta_{max} - \triangle\theta_{break}) \quad \theta_{lb} = (\theta_{min} + \triangle\theta_{break})$$

of the regime in which the axis can move at the rate $\dot{\theta}_{max}$ are defined. Beyond this regime, the timing law

$$\alpha(\tau) = -4\tau^3 + 3\tau \quad \tau \in [-0.5, 0.5] \tag{12}$$

is used to compute the velocity limit as a function of θ. The value of α is set to the fraction of the breaking distance that has been traversed i.e.

$$\alpha = \begin{cases} \frac{\theta-\theta_{ub}}{\triangle\theta_{break}} & \theta > \theta_{ub} \\ \frac{\theta-\theta_{lb}}{\triangle\theta_{break}} & \theta < \theta_{lb} \\ 0 & \text{otherwise .} \end{cases} \tag{13}$$

The fraction has a negative sign at the lower limit and ranges from -1 to 1. Equation 12 is then solved for τ in the allowed range. The normalized derivative of 12 then gives the admissible velocity range

$$\dot{\theta} \in \begin{cases} \left[-\dot{\theta}_{max}, \dot{\theta}_{max}(1-4\tau^2)\right] & \theta > \theta_{ub} \\ \left[-\dot{\theta}_{max}(1-4\tau^2), \dot{\theta}_{max}\right] & \theta < \theta_{lb} \\ \left[-\dot{\theta}_{max}, \dot{\theta}_{max}\right] & \text{otherwise .} \end{cases} \tag{14}$$

This lookup procedure is illustrated in figure 1. In the

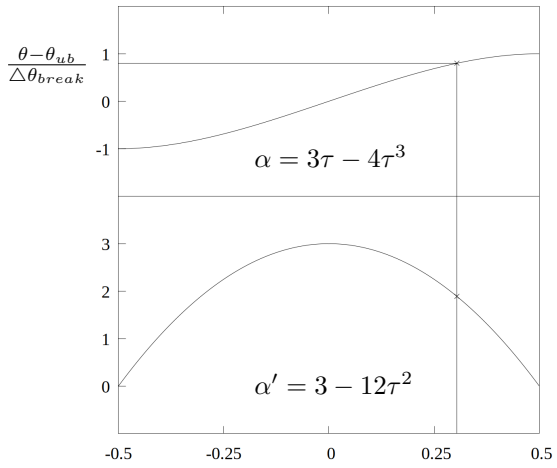

Figure 1: Graphical lookup of the timing law.

top part equation 12 is shown. The value of θ is greater than θ_{ub} and the fraction in 13 takes the value 0.8. This corresponds to a value of τ of approximately 0.3. In the lower part the derivative is shown which takes the value 1.89. Thus the angular velocity needs to be in the range $[-\dot{\theta}_{max}, 0.63 \cdot \dot{\theta}_{max}]$. Limiting the feed-forward velocity term $\dot{\theta}_d$ in this way leads to a smooth breaking when approaching the limits.

No slewing is implemented, so the control law for both axes simply tries to close the gap between the current pointing and the current location of the object. For large separations this can be slightly inefficient and it requires handling passes through zero azimuth. As alternative to that, the errors between the desired and current angle in equation 9 for the azimuth and elevation axes

$$\triangle\phi = \phi_{\text{target}} - \phi_{\text{telescope}}$$
$$\triangle\lambda = \lambda_{\text{target}} - \lambda_{\text{telescope}}$$

can be transformed to follow the geodesic path on the hemisphere. The haversine formula

$$a = \sin(\frac{\triangle\lambda}{2})^2 + \cos(\lambda_{\text{target}}) \cdot \cos(\lambda_{\text{telescope}}) \cdot \sin(\frac{\triangle\phi}{2})^2$$
$$\theta_{gc} = 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \tag{15}$$

gives the central angle θ_{gc} that separates the viewing direction and the target direction. The direction of the of the geodesic error is given by the terms

$$s_B = \cos(\lambda_{\text{target}}) \cdot \sin(\triangle\phi)$$
$$c_B = \cos(\lambda_{\text{telescope}}) \cdot \sin(\lambda_{\text{target}}) - \sin(\lambda_{\text{telescope}}) \cdot \cos(\lambda_{\text{target}}) \cdot \cos(\triangle\phi)$$

derived in [3]. The transformed errors $\triangle\tilde{\phi}$ and $\triangle\tilde{\lambda}$ point along the geodesic and have a magnitude proportional to θ_{gc}.

$$\triangle\tilde{\phi} = \frac{s_B}{\sqrt{s_B^2 + c_B^2}} \cdot \frac{1}{\cos(\lambda_{\text{telescope}})} \cdot \theta_{gc} \tag{16}$$

$$\triangle\tilde{\lambda} = \frac{c_B}{\sqrt{s_B^2 + c_B^2}} \cdot \theta_{gc} \tag{17}$$

Using these transformed errors only results in the telescope moving along the geodesic if the closed-loop response time of both axes is the same.

The high-level tracking control loop using only the ephemeris is shown in figure 2. The four purple blocks represent blocking *HTTP GET* or *PUT* requests that communicate with the mount controller. The orange blocks represent functions that are executed in sequence by the tracking process which runs on another machine (using a different OS) than the one hosting the Alpaca API. The encoder angles returned by the *get azimuth* and *get elevation* calls do not feature a timestamp so the system time is used instead. By getting the timestamp between the two calls for the encoder readings they don't have a

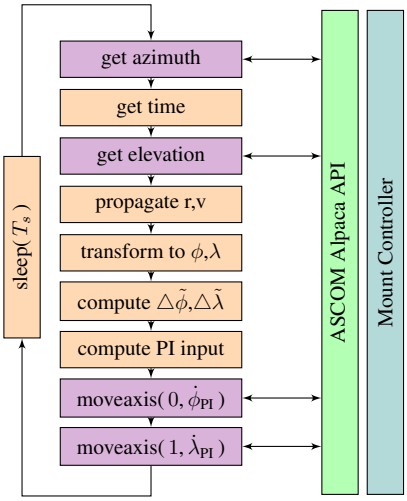

Figure 2: Ephemeris-only tracking loop using the ASCOM Alpaca API

compound delay. Because of the variable delay between the timestamp and the encoder readout the angles seen by the tracking loop feature a correlated error proportional to the velocity of the axis and the total delay. An additional problem that may be particular to our system are erroneous angle readings. Sometimes the API returns the same reading twice on subsequent calls. To address this issue the increments between the past 24 angle readings are kept in memory. The median and the median absolute deviation (MAD) of the array are computed. If the magnitude of the current increment deviates from the median by more than two MAD the reading is discarded and replaced by the previous value plus the median increment. This type of outlier filtering has proven to be robust in a similar application [8]. This filter introduces small errors when the tracking velocity is suddenly changing which happens while slewing to a target. During tracking however, it prevents wrong angle readings from further deteriorating the tracking performance. Figure 3 shows the tracking errors $\triangle\phi$ and $\triangle\lambda$ for a Starlink satellite pass culminating at an elevation of 47 degrees. The RMS for the azimuth and elevation error seen by the control loop is 57.81" and 20.82", respectively. The fact that the magnitude of the azimuth rate is on average 2.9 times greater than the magnitude of the elevation rate for this pass supports the hypothesis that these errors are mainly due to the delay (or lead) of the time stamp. For example, a 25ms variation of the delay between the reading of an encoder value and the use of the value in the control loop for an axis that moves at 0.5 degrees per second leads to a jump of 45 arcseconds (or 0.2 milliradians). The controller trying to correct for the apparent error exacerbates the issue. Since these errors are proportional to the angular rates they are especially problematic when tracking RSOs in LEO.

3. SYSTEM IDENTIFICATION, SIMULATION AND CONTROLLER DESIGN

To simulate the dynamic response of the telescope and design and test a controller it can be useful to model both axes of the telescope using a transfer function. The procedure to fit such a model [9] is described here for completeness. A series of experiments is performed in which a sinusoid input u_e is applied to one telescope axis. The amplitude and phase shift of the sinusoid response can then be used to fit a discrete-time transfer function. Each experiment consists of $(N + N_T)$ input values given at the same fixed sample time T_s and the measured output. The first N_T measurements are discarded because they contain a transient response. The discrete frequencies Ω_l range from constant input to the Nyquist frequency.

$$\Omega_l = \frac{2\pi l}{N} \quad l \in \{0, \ldots, N/2\} \tag{18}$$

The amplitude A of the test input is in the order of 0.01 rad/s which is representative of the corrections that are made by the controller. The test input at timestep k for one frequency Ω_l is given by equation 19.

$$\begin{aligned} u_e[k] &= A \cdot cos(k\,\Omega_l\,T_s) \\ k &\in \{0, \ldots, N + N_T - 1\} \end{aligned} \tag{19}$$

The measured output y_m consists of a scaled and phase-shifted sinusoid, a transient response that decays to zero for increasing k, and some noise.

$$\begin{aligned} y_m[k] &= scale(\Omega_l) \cdot A \cdot cos(k\,\Omega_l\,T_s - phase(\Omega_l)) \\ k &\in \{N_T, \ldots, N + N_T - 1\} \end{aligned} \tag{20}$$

The estimate of the complex value of the transfer function $H(z)$ at frequency Ω_l is then given by the ratio of the discrete Fourier transforms of the input and the output.

$$Y_m = \sum_{k=N_T}^{N+N_T-1} y_m[k]\, e^{-j\cdot\Omega_l\cdot k} \tag{21a}$$

$$U_e = \sum_{k=N_T}^{N+N_T-1} u_e[k]\, e^{-j\cdot\Omega_l\cdot k} \tag{21b}$$

$$\hat{H}(\Omega_l) = \frac{Y_m}{U_e} \tag{21c}$$

Using all measured frequencies Ω_l the coefficients of a discrete transfer function can be obtained by a least-squares fit. The transfer function $H(z)$ is parametrized by the coefficients b_i and a_i of the numerator and denominator polynomials. In the following, a transfer function of order 2 is considered to facilitate notation.

$$H(z) = \frac{b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{22}$$

$$= z^{-1} \frac{b_1 + b_2 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{23}$$

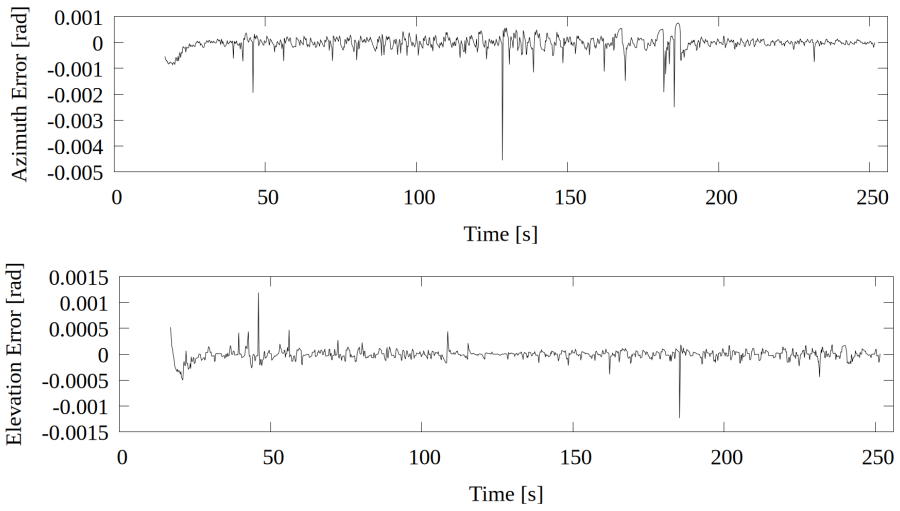

Figure 3: Tracking errors seen by the control loop for a Starlink satellite pass culminating at 47 degrees of elevation. The actual tracking error is not known since this is a daytime pass that was tracked with a closed dome for testing purposes.

Note that the input delay factored out in equation 23 ensures that the output can only depend on inputs up to the previous timestep of the discrete system. For an integrating plant the transfer function can be constrained to have a zero at -1 and a pole at $+1$. This corresponds to rewriting the equation as

$$H(z) = \frac{\tilde{b}_1 z^{-1}}{1 + \tilde{a}_1 z^{-1}} \cdot \frac{T_s}{2} \frac{1 + z^{-1}}{1 - z^{-1}}. \tag{24}$$

To fit the system to the measured frequency responses the operator z^{-k} is replaced by $exp(-j \cdot \Omega_l \cdot k)$ for each of the measured frequencies Ω_l.

$$\begin{aligned}(1 + a_1 e^{-j\Omega_l} + a_2 e^{-j2\Omega_l}) \cdot \hat{H}(\Omega_l) \\ = b_1 e^{-j\Omega_l} + b_2 e^{-j2\Omega_l}\end{aligned} \tag{25}$$

Rewriting equation 25 in terms of the unknown coefficients of the transfer function gives the complex linear equation corresponding to the frequency Ω_l.

$$\begin{aligned}\left[e^{-j\Omega_l}\ e^{-j2\Omega_l}\ \left(-e^{-j\Omega_l}\hat{H}(\Omega_l)\right)\ \left(-e^{-j2\Omega_l}\hat{H}(\Omega_l)\right)\right] \\ \cdot \begin{pmatrix} b_1 \\ b_2 \\ a_1 \\ a_2 \end{pmatrix} = \left[\hat{H}(\Omega_l)\right]\end{aligned} \tag{26}$$

Combining the equations of all measured frequencies gives an overdetermined system of equations. The complex-valued least-squares solution is obtained by solving the system below.

$$(\mathbf{A}^* \mathbf{A}) \cdot \Theta = \mathbf{A}^* \mathbf{B}$$

$$\mathbf{A} = \begin{bmatrix} e^{-j\Omega_l}\ e^{-j2\Omega_l}\ \left(-e^{-j\Omega_l}\hat{H}(\Omega_l)\right)\ \left(-e^{-j2\Omega_l}\hat{H}(\Omega_l)\right) \\ \vdots \end{bmatrix}$$

$$\Theta = \begin{pmatrix} b_1 \\ b_2 \\ a_1 \\ a_2 \end{pmatrix} \qquad \mathbf{B} = \begin{bmatrix} \hat{H}(\Omega_l) \\ \vdots \end{bmatrix} \tag{27}$$

To ensure the coefficient vector Θ is real-valued, the real and imaginary parts of equation 27 are stacked.

$$\mathbf{A}_\mathbb{R} \cdot \Theta = \mathbf{B}_\mathbb{R}$$

$$\begin{aligned}\mathbf{A}_\mathbb{R} &= \begin{bmatrix} Re(\mathbf{A}^*\mathbf{A}) \\ Im(\mathbf{A}^*\mathbf{A}) \end{bmatrix} \\ \mathbf{B}_\mathbb{R} &= \begin{bmatrix} Re(\mathbf{A}^*\mathbf{B}) \\ Im(\mathbf{A}^*\mathbf{B}) \end{bmatrix}\end{aligned} \tag{28}$$

The integrator constraint shown in equation 24 can be expressed by an additional set of equality constraints.

$$\mathbf{C} \cdot \Theta = \mathbf{D}$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 \end{bmatrix} \qquad \mathbf{D} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} \tag{29}$$

The constrained problem is solved using the null space method [6]. First a particular solution Θ_p is determined that fulfills equation 29. Then the nullspace of $\mathbf{C}$ is computed (i.e. a matrix which has all linearly independent solutions of $\mathbf{C} \cdot x = 0$ as columns). The matrices $\bar{\mathbf{A}}_\mathbb{R}$ and $\bar{\mathbf{B}}_\mathbb{R}$ are computed using the nullspace of $\mathbf{C}$ and the particular solution. The projection of the least squares problem in equation 28 is then solved for Θ_h. Finally, the solution to the constrained least squares problem is given by equation 30e.

$$\mathbf{N} = \text{Null}(\mathbf{C}) \tag{30a}$$

$$\bar{\mathbf{A}}_{\mathbb{R}} = \mathbf{A}_{\mathbb{R}}\mathbf{N} \tag{30b}$$

$$\bar{\mathbf{B}}_{\mathbb{R}} = \mathbf{B}_{\mathbb{R}} - \mathbf{A}_{\mathbb{R}} \cdot \Theta_p \tag{30c}$$

$$\bar{\mathbf{A}}_{\mathbb{R}} \cdot \Theta_h = \bar{\mathbf{B}}_{\mathbb{R}} \tag{30d}$$

$$\Theta = \Theta_p + N \cdot \Theta_h \tag{30e}$$

A MATLAB script that fits such a model to a measured system response is included in appendix A. Since the azimuth inertia is a function of the elevation it makes sense to perform the system identification experiment at an elevation of $45°$ to capture a mean model [7]. A third order transfer function (i.e. a second order system integrated once) fits the data well and is used to model the response of the two axes. The transfer functions are used to simulate the response of both axes in MATLAB Simulink. This model is useful to test the implementation of particular features of the controller such as the back-calculation and the behavior at the limits. The same code that is used to control the telescope can be safely tested in this environment by compiling it as a C *S-function*. The proportional and integral gains for the azimuth and elevation axes are obtained using the *pidtune* function of the MATLAB Control System Toolbox. A target *cross-over frequency* of 1 rad/s is specified for both axes to achieve an acceptable settling time.

4. REAL-TIME IMAGE PROCESSING

The goal of the algorithms described in this section is to keep track of the observed object within the frame and to detect when the object is lost. The optical tracking should yield a contiguous set of image coordinates corresponding to the tracked object. It should be robust to disturbances such as a variation in the intensity of the object or bright streaks from stars. The two assumptions made here are that the object only moves by a moderate number of pixels from one frame to the next and that its SNR remains above a threshold.

4.1. Optical tracking

The first step consists of recursively moving a small ROI across the frame, such that the object remains in its center. The second step is performing aperture photometry on the ROI to obtain centroid coordinates with subpixel precision. The first step makes use of a variation of the Kanade-Lucas-Tomasi (KLT) feature tracker [11] called the inverse compositional algorithm [13]. In its simplest form, this algorithm minimizes the residual sum of squares between the intensities of a template and a shifted image. The algorithm is initialized by copying a subframe of the image to a template. The subframe is parametrized by the offset (dx, dy) from the origin of the image and its dimensions Ω. For each subsequent image an increment of the offset $(\triangle dx, \triangle dy)$ is computed that minimizes the least squares pixel distance 31 between the shifted image and the template.

$$\sum_{x,y\in\Omega} \left[T(x+\triangle dx, y+\triangle dy) - I(x+dx, y+dy)\right]^2 \tag{31}$$

For small increments this expression is well approximated by the first order Taylor expansion given by 32.

$$\sum_{x,y\in\Omega} \left[T(x,y) + \frac{\partial T}{\partial x}\triangle dx + \frac{\partial T}{\partial y}\triangle dy - I(x+dx, y+dy)\right]^2 \tag{32}$$

The minimum of this expression is given by the linear system of equations 33.

$$\begin{bmatrix} \sum_\Omega \frac{\partial T}{\partial x}\cdot\frac{\partial T}{\partial x} & \sum_\Omega \frac{\partial T}{\partial x}\cdot\frac{\partial T}{\partial y} \\ \sum_\Omega \frac{\partial T}{\partial y}\cdot\frac{\partial T}{\partial x} & \sum_\Omega \frac{\partial T}{\partial y}\cdot\frac{\partial T}{\partial y} \end{bmatrix} \begin{bmatrix} \triangle dx \\ \triangle dy \end{bmatrix} = \begin{bmatrix} \sum_\Omega \frac{\partial T}{\partial x}\cdot[I(x+dx, y+dy) - T(x,y)] \\ \sum_\Omega \frac{\partial T}{\partial y}\cdot[I(x+dx, y+dy) - T(x,y)] \end{bmatrix} \tag{33}$$

The hessian matrix on the left-hand side consists of four scalars that only need to be computed once when setting the template. The right-hand side is computed for each frame by subtracting the template from the image at the offset (dx, dy) and computing the inner product of the difference and the gradients of the template which are kept in memory along with the template itself. The offset is then updated using equation 34.

$$\begin{bmatrix} dx \\ dy \end{bmatrix}_{k+1} = \begin{bmatrix} dx \\ dy \end{bmatrix}_k + \begin{bmatrix} \triangle dx \\ \triangle dy \end{bmatrix} \tag{34}$$

In general, the offset computed in this way is a floating point number which means that the sub-frame at $I(x+dx, y+dy)$ is obtained by interpolating between the pixels of the image. After a few iterations this algorithm converges to the coordinates of the best match of the template in the image. This best match its not a global minimum of the error between the template and the image since the right hand side of equation 33 only considers a sub-frame of size Ω located at the offset (dx, dy). As the offset is incrementally changed the subframe is displaced to a region of the image which "looks more like the template". The algorithm acts like a dynamic aperture that is automatically displaced such that the object remains in its center. The image in our application is an integer-valued array in which each integer represents the Analog-to-Digital Units accumulated by the corresponding pixel on the sensor. Since the KLT tracker is just used to keep the object in the subframe the offset (dx, dy) doesn't have to be accurate to the single pixel. Thus, an integer formulation of the algorithm is used. Since the gradients of the template are not integer-valued both sides of equation 33 are multiplied by four. The scaled gradients of the template are then computed by convolving it with the kernel $[-1, 0, 1]$ along the respective axis. This convolution reduces the size of the image by the first and last row or column. To end up with a gradient of the same size as the template the array is padded with one extrapolated row or

column at each end, before the convolution. Since the solution of equation 33 is still rational only its sign is used to move the offset (dx, dy) by one pixel along the x axis and one pixel along the y axis. This algorithm can only converge to a neighborhood with manhattan-distance 1 to the pixel containing the coordinates that minimize the least squares distance. The Hessian determinant of the convex minimization problem is non-negative, thus only the numerator of the solution to equation 33 needs to be considered to determine the sign.

Algorithm 1 Integer KLT Tracker

$dx \leftarrow x_0 - w/2$
$dy \leftarrow y_0 - h/2$
$T \leftarrow \text{SUBFRAME}(I_0, dx, dy, w, h)$
$G_x \leftarrow [-1, 0, 1] * T$ // $*$ is the convolution operator
$G_y \leftarrow [-1, 0, 1]^T * T$
$h_{xx} \leftarrow \sum_\Omega G_x \cdot G_x$ // $\Omega \hat{=} \{x \in [1, w], y \in [1, h]\}$
$h_{xy} \leftarrow \sum_\Omega G_x \cdot G_y$
$h_{yy} \leftarrow \sum_\Omega G_y \cdot G_y$
for $k = 1, \ldots, N_{\text{images}}$ **do**
    for $i = 1, \ldots, 10$ **do**
        $M \leftarrow \text{SUBFRAME}(I_k, dx, dy, w, h)$
        $b_x \leftarrow \sum_\Omega G_x \cdot (M - T)$
        $b_y \leftarrow \sum_\Omega G_y \cdot (M - T)$
        if $(h_{yy} \cdot b_x > h_{xy} \cdot b_y)$ **then**
            $dx \leftarrow dx - 1$
        else
            $dx \leftarrow dx + 1$
        end if
        if $(h_{xx} \cdot b_y > h_{xy} \cdot b_x)$ **then**
            $dy \leftarrow dy - 1$
        else
            $dy \leftarrow dy + 1$
        end if
    end for
end for
procedure SUBFRAME(I, dx, dy, w, h)
    $S \leftarrow \mathbf{0}$
    for $x = 1, \ldots, w$ **do**
        for $y = 1, \ldots, h$ **do**
            if $(x + dx, y + dy)$ **is inside** I **then**
                $S(x, y) \leftarrow I(x + dx, y + dy)$
            end if
        end for
    end for
    return S
end procedure

Algorithm 1 has three parameters, the first two being the height and width of the subframe. The third parameter is the number of iterations after which the tracker is assumed to have converged. Selecting a number close to $w/2$ makes sense because the accumulated offsets $\triangle dx$ and $\triangle dy$ should remain smaller than that. Note that if the tracker loses the object the number of iterations also dictates how fast the algorithm gets lost in the image. If the object is non-resolved the template could be generated analytically using a point spread function. This would be especially useful when applying the algorithm in parallel to track multiple similar point sources. If the subframe does not contain an object the tracker is lost and the entire next frame needs to be scanned for a good feature to track. The hessian matrix seen on the left hand side of equation 33 needs to have two large eigenvalues that do not differ by several orders of magnitude [12]. To find the best subframe within the image, which hopefully contains the tracked object, the gradients of the entire frame are computed using the same functions as in algorithm 1. The three components of the symmetric hessian are then computed for cells that measure one fourth of the subframe size. For every combination of four neighboring cells the hessian is then computed by summing each of the three components over the four cells. The eigenvalues are then given by the roots of the characteristic polynomial as expressed in equation 35. The subframe with the largest minor eigenvalue is then selected and the object is centered in the frame using algorithm 2. This works well enough if there is a single bright target in the frame and no substantial disturbances such as bright stars in the background. If this is not the case a more elaborate resetting of the tracker or intervention by a human observer is required.

$$\lambda_{1,2} = \frac{(h_{xx} + h_{yy}) \pm \sqrt{(h_{xx} - h_{yy})^2 + 4 \cdot h_{xy}^2}}{2} \tag{35}$$

4.2. Object Detection

The second step takes the best match of the template in the current image obtained by applying algorithm 1 and computes the centroid of the object. The background is estimated by the median intensity within an annulus of fixed size centered in the subframe M. The median is an integer pixel value obtained by applying the *select* algorithm from [10]. Using the median as a threshold a bitmap of the ROI is generated encoding all pixels with values above the median as 1 and all others as 0. Then an erosion procedure consisting of bit-wise operations is applied to the bitmap five times. The erosion procedure counts how many bits are set within a 3x3 neighborhood around each pixel. If the sum exceeds an integer k the pixel is left unchanged otherwise it is set to 0. The integer k is incremented in each pass of the erosion procedure and takes the values $\{4, 5, 6, 7, 8\}$. This removes everything but large patches that are above the noise level as shown in figure 4. If the entire ROI is filled with Poisson noise with an expected value of λ a pixel has a value greater than λ with the probability p_+.

$$p_+ = 1 - \sum_{k=0}^{\lambda} \frac{\lambda^k e^{-\lambda}}{k!} \tag{36}$$

The probability that any κ of the 8 neighboring pixels are also greater than λ is given by the binomial distribution

$$\binom{8}{\kappa} p_+^\kappa (1 - p_+)^{8-\kappa}.$$

Summing the probabilities that zero, one, or two neighboring pixels have a value greater than λ and taking the complement

$$1 - \sum_{\kappa=0}^{2} \binom{8}{\kappa} p_+^{\kappa} (1 - p_+)^{8-\kappa}$$

gives the probability that there are 3 or more neighboring pixels around the central pixel that have a value greater than λ. Thus, the probability that a pixel and more than three of its neighbors are above the expected value is then given by equation 37.

$$p_4 = p_+ \cdot (1 - \sum_{\kappa=0}^{2} \binom{8}{\kappa} p_+^{\kappa} (1 - p_+)^{(8-\kappa)}) \tag{37}$$

Algorithm 2 Object detection

$M \leftarrow \text{SUBFRAME}(I_0, dx^*, dy^*, w, h)$
$tmp \leftarrow \text{COPYRING}(M, r_i, r_o)$
$med \leftarrow \text{MEDIAN}(tmp)$
$B \leftarrow (M > med)$
for $k = 4, \ldots, 8$ **do**
    $B \leftarrow \text{ERODE}(B, k)$
end for
for $k = 3, \ldots, 0$ **do**
    $B \leftarrow \text{DILATE}(B, k)$
end for
$m_x \leftarrow 0$
$m_y \leftarrow 0$
$s \leftarrow 0$
for $x = 1, \ldots, w$ **do**
    for $y = 1, \ldots, h$ **do**
        if $B(x, y)$ **then**
            $m_x \leftarrow m_x + x \cdot M(x, y)$
            $m_y \leftarrow m_y + y \cdot M(x, y)$
            $s \leftarrow s + M(x, y)$
        end if
    end for
end for
$x_c \leftarrow dx^* + \frac{m_x}{s} - 0.5$
$y_c \leftarrow dy^* + \frac{m_y}{s} - 0.5$
procedure ERODE(B, k)
    return $B \cap ((B * \mathbb{1}_{3\times3}) > k)$
end procedure
procedure DILATE(B, k)
    return $((B * \mathbb{1}_{3\times3}) > k)$
end procedure

This corresponds to the probability that a pixel survives a pass of the erosion procedure with $k = 4$. The probability that it survives all 5 passes with increasing values of k is in the order of 10^{-5} for realistic values of λ. Depending on the size of the ROI this value can be used to set a threshold number of pixels left after the erosion below which the tracker is considered lost. Since the erosion procedure also shrinks the patch containing the object a similar procedure referred to as dilation is applied four times. The dilation procedure sets all bits that have more than k neighbors regardless of their previous values. To ensure the patch does in fact grow k takes the values $\{3, 2, 1, 0\}$. As can be seen in figure 4 the dilation passes start by filling holes and concavities of the signal patch for $k = 3$ and finish by adding a one pixel offset around the patch for $k = 0$. If the number of set bits in the bitmap indicate that there is an object in the subframe the centroid is computed using the bitmap as a mask. Finally, the coordinates of the subframe dx and dy are adjusted to the nearest integer values that center the object in the subframe.

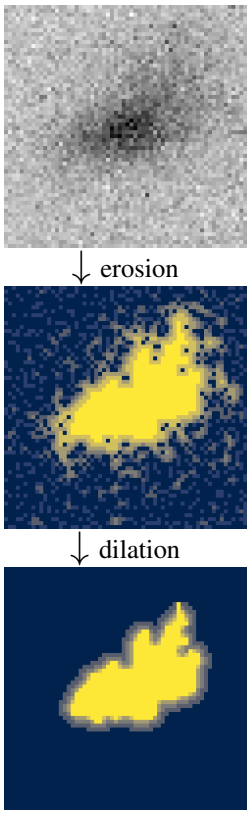

Figure 4: Inverted subframe containing object partially occluded by clouds (top), Eroded bitmap of the subframe, the brightest shade indicates the pixels left after five iterations (middle), Bitmap after dilation, the brightest shade corresponds to the bitmap before dilation (bottom)

5. ACTIVE TRACKING USING OPTICAL FEEDBACK

Some challenges arise when the ephemeris-only tracking is combined with the optically measured tracking errors. The tracking loop shown in figure 2 does not have a constant runtime due to the delays of the blocking function calls. Waiting a full timestep T_s after giving the velocity commands is necessary in order for these inputs to be able to take effect before the next error measurement is made. The CMOS Camera on the other hand acquires frames at a precise sampling rate and the implementation of the real-time processing runs orders of magnitude faster. The

tracking performance of the ephemeris-only tracking illustrated in figure 3 is not good enough for measurements because of the varying delay, the missing timestamps, and the occasional false readings. To address both issues only the optically measured error is used for tracking, once the object has been acquired. The computer that is connected to the camera also runs the high-level controller that generates the telescope inputs.

5.1. Camera operating states

The operating state of the camera dictates the operating state of the entire system. The camera is controlled by a main loop that can switch between three operating states. These are called idle, stare, and chase in reference to a similar system developed for laser-ranging [8]. In the idle mode, the camera does nothing. The purpose of the stare mode is for a user to see the entire FOV of the sensor in order to identify and select a target. In the stare mode the camera is acquiring full frames at the highest framerate for a fixed exposure time, but not faster than two fps. In addition, a binning of 2x2 is used to reduce the image size. Both of these measures ensure that the GUI which uses the Xlib runs smoothly even when the display is being forwarded through SSH. In the chase mode each frame is processed in real-time and written to a fits file. A 240 by 240 pixel subframe centered in the FOV is captured without binning. The framerate is set to the highest possible value for the exposure time but not faster than 20 fps. Switching between the states is achieved by a single character input buffer that is updated in every iteration of the main loop. The characters 'i','s', and 'c' are the commands for the corresponding mode and they are input by the user through the GUI. While in chase mode the two algorithms described in the previous section are used to track a 32x32 pixel ROI containing the object and to compute its centroid. The coordinates of the center of the frame are subtracted from the centroid coordinates to obtain deviations in image coordinates. These errors then need to be transformed to normal coordinates ξ,η. Since the CMOS camera is attached to a Nasmyth port without derotator this transformation is a function of the elevation and takes the form shown in equation 38 for our configuration. The angle β is the sum of a constant offset and the elevation of the telescope. The scaling is given by the ratio of the pixel size w_p and the focal length of the telescope f.

$$\begin{bmatrix}\xi\\ \eta\end{bmatrix} = \frac{w_p}{f}\begin{bmatrix}\cos(\beta) & \sin(\beta)\\ -\sin(\beta) & \cos(\beta)\end{bmatrix}\cdot\begin{bmatrix}dx\\ dy\end{bmatrix} \qquad (38)$$

5.2. Telescope Control with optical feedback

Combining the described telescope control and the optical tracking of the object in the frame enables the system to track using an optical error.

First, the ephemeris-only tracking is used to acquire the object in the image using the stare mode of the camera. The object is selected by the observer clicking on it on the GUI. The user input is transformed to image coordinates and then to the local coordinates ξ,η. These are added to the ephemeris position which centers the object in the frame. When the object is centered in the full frame the observer gives the command to switch to the chase mode. The control loop switches to the optical feedback mode which entirely relies on the ephemeris of the object and the observed tracking error. The errors $\triangle\phi$,$\triangle\lambda$ are replaced by the local coordinates ξ,η. This eliminates the need for the two API calls getting the encoder readings. It also ties the sampling rate of the control loop to the acquisition framerate of the camera. To ensure the two remain synchronous the camera is used as trigger for the control loop. Whenever a new frame has been acquired, the real-time image processing is performed. After the centroid has been computed the thread running the camera state machine raises a condition which signals the control loop of the telescope to compute new velocity inputs and send them to the axes. Since the processing of the frame incurs an almost constant delay the variation of the timestamp is now much lower and so should be the tracking RMS. This is indeed the case for the pass of Ajisai shown in figure 6 since the image errors dx and dy have an RMS of 0.81" and 0.94", respectively.

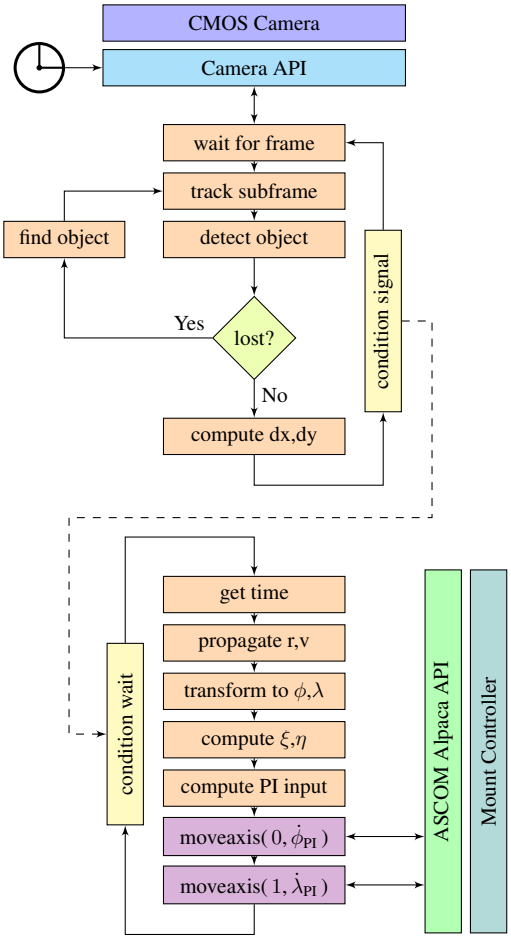

Figure 5: Optical feedback tracking using the ASCOM Alpaca API

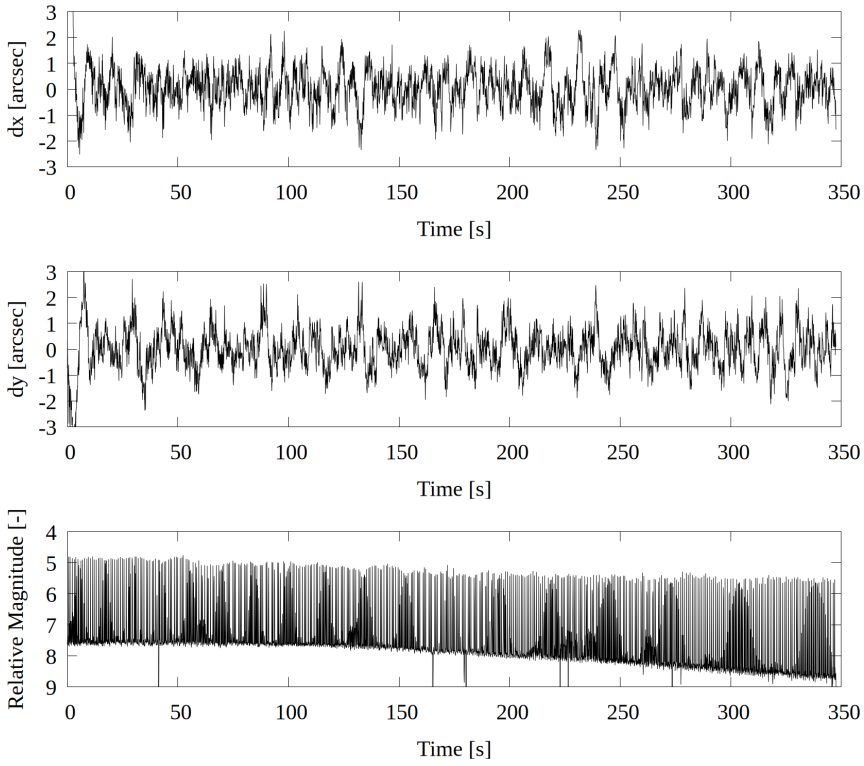

Figure 6: Tracking errors along the image axes for a zenith pass of Ajisai (top and center) and the light curve generated from the tracking frames (bottom).

6. RESULTS AND DISCUSSION

The 6 minute observation of Ajisai shown in figure 6 is the second half of a zenith pass and has a tracking rms on the sensor of 1.24”. An uninterrupted 11 minute observation of LCS 1 acquired 20 minutes earlier with a culmination at 47 degrees has a tracking rms of 1.06”. A 1 minute observation of a starlink satellite that entered the earth shadow at an elevation of 55 degrees after a culmination at 57 degrees has a tracking rms of 4.46”. This is in part due to the brightness of the object which led to the frames being overexposed. A circle with a radius of 10 pixels (corresponding to 2.4”) is saturated which makes the centroid computed with algorithm 2 become the geometric center of the signal patch. Another artifact that can be observed for the bright flashes of the Ajisai pass is that in a few frames the entire annulus used for the background estimation is covered by the bright signal. In these frames the computed median is much higher than the true background so the object detection throws a false negative. This does not disrupt the tracking since no correction to the angular rates is made when the optical tracker is lost. In the subsequent frame the specular flash is over and the target is automatically reacquired. The relative magnitude shown in figure 6 illustrates the robustness of the real-time image processing to specular flashes. The observation of the starlink satellite entering the earth shadow helps the understanding of the behavior of the optical tracking at low SNR. If the template of the object features strong gradients with a bright spot in the center algorithm 1 doesn’t lose the object even if it becomes very faint. Algorithm 2 however with the fixed number of iterations and pre-defined values of k soon indicates that the object has been lost. This triggers the reset procedure that looks for the best hessian in the subframe which will result in a new template with much smaller gradients. Thus in order to track low SNR targets the following measures can be taken: An artificial template that is much brighter than the actual signal and has strong gradients can be used to ensure algorithm 1 keeps track of the object. The threshold number of pixels left after the erosion procedure can be lowered in order to reduce the number of false negatives of the object detection. The number of iterations and the number of neighbors for both the erosion and dilation can be adapted.

One of the main limitations of the method is the ephemeris-only tracking performance illustrated in figure 2. Especially for RSOs in LEO below 1000km altitude a noticeable wobble can be observed when manually selecting the target in stare mode. If the Alpaca API would return timestamps with the encoder readings this performance could be improved. A much more severe limitation is that the sampling time of the discrete controller is directly tied to the exposure time of the camera. For many targets the assumption can be made that the angular velocities remain constant during one full exposure time, for the LEO observations an exposure time of a 0.1 seconds and a frame rate of 9.87 fps was used. Objects in

MEO and GEO usually need exposure times of 1 second or more but since their angular rates are much smaller the reduced sampling time is not an issue and even improves stability. However tracking fast-moving low-SNR objects or targets that are emitting optical signals such as LED-SAT may prove to be challenging with this method. For our application, the tracking performance of this system is sufficient. The use of the ASCOM Alpaca API and just the most basic functionalities of the mount should make this method easy to generalize to other telescopes. The camera being integrated with the PC that is running the high-level controller should not be an issue, since modern CMOS cameras and industrial PCs are portable and can easily be shipped whereas telescopes with an aperture of 80 cm are not. Also the sensor that is being used in parallel to the tracking CMOS camera is completely independent from the system. Our application is SPAD photometry but e.g. spectroscopic measurements of RSOs may have very similar requirements. The tasks of the operator i.e. finding and selecting the target and switching the operating mode should be easy to automate. Doing so would enable the system to acquire light curves autonomously and even if some have to be discarded the productivity of the sensor would likely exceed what can be achieved by human observers. A beneficial side-effect would be that a GUI that displays the footage as it is acquired would then be redundant, which would simplify the software implementation. Finally, a more incidental feature of the system is that it does not rely on high-precision encoders. Any encoder precise enough to keep the object in the FOV in the stare mode will do.

7. SUMMARY

For non-resolving optical sensors tracking errors result in the observed RSO leaving the FOV of the sensor. To deal with this issue, either highly accurate ephemerides and a well-calibrated telescope or active tracking is required. A framework that is based on a TCP/IP interface to the mount controller of the telescope and a custom program for a CMOS camera is presented. The solution should be applicable to other telescopes since only a few parameters need to be determined in a structured way. Some of the lag that is inherent to network communication can be avoided by closing the control loop through optical feedback. This requires robust algorithms that track and detect the object. Our solution is able to track calibration targets in LEO with an RMS in the order of 1". The presented method could be further developed for autonomous acquisition of light curves and may lower the requirements for the encoders of SST sensors.

ACKNOWLEDGMENTS

The first author would like to thank the Swiss National Science Foundation for providing the funds for the current study as well as the Astronomical Institute of the University of Bern for providing the infrastructure. He would also like to thank the staff at the SwissOGS Observatory in Zimmerwald for their support and fruitful discussions which contributed to the presented work.

REFERENCES

1. Celestrak https://celestrak.org/software/
2. Space-Track, United States Space Command, https://www.space-track.org
3. The Math Forum, https://web.archive.org /web/20171219235009/http://mathforum.org/library /drmath/view/55417.html
4. Astrom, K. J. and Hagglund, T. (1995) *PID Controllers: Theory, Design and Tuning*, Instrument Society of America, ed. 2
5. Lynch, K. M. and Park, F. C.(2017). *Modern Robotics: Mechanics, Planning, and Control*, Cambridge University Press
6. Lawson, C. L., Hanson, R. J. (1995). *Solving least squares problems*, Society for Industrial and Applied Mathematics
7. Gawronski, W. (2008). *Modeling and Control of Antennas and Telescopes*, Springer
8. Rodriguez-Villamizar, J., Cordelli, E., and Schildknecht, T. (2021). The stare and chase observation strategy at the Swiss Optical Ground Station and Geodynamics Observatory Zimmerwald: From concept to implementation, *Acta astronautica*, **189**
9. Signals and Systems Lecture, Institute for Dynamic Systems and Control ETHZ, (2017). https://ethz.ch/content/dam/ethz/special-interest/mavt/dynamic-systems-n-control/idsc-dam/Lectures/Signals-and-Systems/2019/notes/lecture11.pdf, accessed 07. Feb 2025
10. Teukolsky, S. A., Flannery, B. P., Press, W., and Vetterling, W. (1992). *Numerical recipes in C*. SMR, 693(1), 59-70.
11. Lucas, B. D., and Kanade, T., (1981). An iterative image registration technique with an application to stereo vision, *IJCAI'81: 7th international joint conference on Artificial intelligence.* , Vol. 2
12. Tomasi, C. and Kanade, T. (1991). Detection and tracking of point features. Carnegie Mellon Univ. Technical Report.
13. Baker, S. and Matthews, I. (2001). Equivalence and efficiency of image alignment algorithms, *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.*,Vol. 1

A. SYSTEM IDENTIFICATION SCRIPT

```
%% system identification script
% revolute joint controlled by velocity inputs
clear all; close all; clc;
sps = 10;                                   % sampling rate
N_T = 10 * sps;                             % skip 10s transient response
N = 256+1;
l = [1,2,3,4,6,8,12,16,24,32,48,64].';
omega = 2*pi*l./N;                          % subset of discrete frequencies
A = 0.01;                                   % input amplitude = 0.01 rad / s
i = [0:(N_T + N-2)];
% input matrix uexp: each row corresponds to an experiment
% consisting of N_T + N inputs to the axis.
uexp = A * cos(omega * i);
% output matrix ymeas: each row corresponds to the measured output
ymeas = zeros(length(omega),length(i));
% --- fill ymeas with "measured" outputs --- %
b = conv([0.0 0.03 0.002],[1, 1])
a = conv([1 -1.2 0.25],[1,-1])
noise = 0.001*randn(size(uexp));
ymeas = filter(b, a, uexp.').' + noise;
% --- remove bias --- %
ymeas = ymeas - mean(ymeas(:,N_T:(N_T + N-1)),2)*ones(1,(N_T + N-1));
% --- fit transfer function --- %
mask = [zeros(size(l,1),N_T),ones(size(l,1),N-1)];
Ym = sum( ymeas .* exp(-j*omega*i) .* mask ,2);
Ue = sum( uexp  .* exp(-j*omega*i) .* mask ,2);
Hest = Ym./Ue;
% --- order of transfer function --- %
na = 4;                                 % degree of denominator N is (na-1)
nb = 3;                                 % degree of numerator M is (nc+nb-1)
nc = 1;                                 % system delay
lhs = exp(-j*omega*[0:(na-1)]).*Hest;
rhs = exp(-j*omega*[nc:(nb + nc -1)]);
B = lhs(:,1);
A = [rhs, -lhs(:,2:end)];
ACA = A'*A;
ACB = A'*B;
AR = [real(ACA);imag(ACA)];
BR = [real(ACB);imag(ACB)];
% --- integrating plant constraint --- %
C = [zeros(1,nb+nc),ones(1,na-1);
    2*mod([1:nb+nc],2) - ones(1,nb+nc),zeros(1,na-1)];
D = [-1;0];
C = C(:,(1+nc):end);
theta_p = C\D;
N = null(C);
ARbar = AR*N;
BRbar = BR - AR*theta_p;
theta_h = lsqr(ARbar,BRbar);
theta = theta_p + N*theta_h;
% --- unconstrained solution --- %
%theta = lsqr(AR,BR);
b = [zeros(1,nc),theta(1:nb).']
a = [1,theta(nb+1:end).']
sysfit = tf(b,a,1/sps);
```

```
% --- bode plot --- %
wmeas = omega*sps;
fmeas = wmeas/(2*pi);
[mag,phas,wout] = bode(sysfit);
fout = wout/(2*pi);
magfit = mag2db(squeeze(mag));
phasfit = squeeze(phas);
if any(phasfit > 0)
    phasfit = phasfit - 360;
end
tiledlayout(2,1)
nexttile
scatter(fmeas,mag2db(abs(Hest)),'+k')
hold on
semilogx(fout,magfit,'k')
ylim([-50,50])
yticks([-40:20:40])
ylabel("[dB]")
xline(sps/2)
hold off
set(gca,'xscale','log')
grid on
box on
title('magnitude')
nexttile
scatter(fmeas,phase(Hest)/pi*180,'+k')
hold on
semilogx(fout,phasfit,'k')
ylim([-300,0])
ylabel("[deg]")
xlabel("[Hz]")
xline(sps/2)
hold off
set(gca,'xscale','log')
grid on
box on
title('phase')
```