A rigorous algorithm for telescope pointing

Patrick Wallace

Rutherford Appleton Laboratory,
Chilton, Didcot, Oxon OX11 0QX, United Kingdom

ABSTRACT

A typical modern telescope control system points by first calculating the direction of the target in nominal mount coordinates and then applying small corrections to the demanded mount angles. The pointing refers to the rotation axis of the instrument mount, and rotator demands are calculated *via* parallactic angle. This simple approach works well enough when the corrections are small and the accuracy objectives are modest. However, a more rigorous approach can pay off, in the form of improved pointing, more accurate guide probe predictions, reduced residual field rotation and reliable world coordinate system information even when the detector is off-axis. In this paper I propose a rigorous vector/matrix algorithm for generating pointing predictions on an imperfect telescope, with support for autoguiding, field stabilization and world coordinate systems even in difficult cases such as Nasmyth and coudé.

Keywords: telescope control, world coordinates, pointing, astrometry

1. INTRODUCTION

The phrase "telescope pointing" means choosing demand angles for the mount axes so that the specified celestial target is seen. Most users, of most telescopes, will find little wrong with that definition. But in fact there are already two important details missing. Where in the focal plane do you want the image to appear? And which way up do you want the picture to be on the instrument? There is also autoguiding to think about, as well as how to deal with corrections for pointing errors.

In this paper I look at the pointing problem for a conventional two-axis "gimbal" mount, discuss the limitations of existing practices, and introduce a new algorithm that avoids these limitations. The algorithm is at the heart of the proprietary TCSpk pointing-kernel, which will be used on the SOAR 4-meter telescope in Chile and the 10.4-meter Gran Telescopio Canarias.

2. EXISTING DESIGNS

Precise automatic control has for several decades been *de rigeur* for all large groundbased telescopes, whether equatorially mounted or altazimuth. At the other end of the scale, even mass-produced small telescopes these days often have some form of "Go To" capability; more expensive models include control facilities which rival professional telescopes in terms of pointing control and usually offer *much* more sophisticated user-interface facilities. Compensation of telescope defects such as misalignments, run-outs and flexures is commonplace, and most major observatories offer sufficiently accurate pointing that the use of "finding charts" is frowned upon. Evidently existing pointing algorithms are doing a satisfactory job; where is the scope for further improvement?

2.1. The simple approach

For both equatorials and altazimuths, the first step is to transform the supplied target position—typically a right ascension and declination, $[\alpha, \delta]$, from a star catalog—into an "observed place". This involves precession, nutation and aberration, then sidereal time, and finally refraction. All of this is described in positional-astronomy textbooks, and supported by subroutine libraries such as SLALIB[1] and NOVAS[2]; it will not be further discussed here. The pointing model, if any, is implemented as a set of semi-empirical expressions that are fitted to observations of standard stars. The model generates, as a function of direction in the sky, small adjustments to be added to the target $[\alpha, \delta]$.

E-mail: ptw@star.rl.ac.uk

Given the observed $[\alpha, \delta]$ plus any telescope-specific pointing corrections, controlling an equatorial telescope is relatively straightforward. Once the target has been acquired and centered on the instrument, sidereal tracking in hour angle alone will tend to keep the image fixed in both position and orientation. Consequently, many telescopes dispense with tracking corrections and, if there is an instrument rotator, it will usually stay fixed during observations. Altazimuth telescopes are less forgiving. An image rotator, continuously driven to follow the parallactic angle, is of course essential if gross field rotation effects are to be avoided. And the fact that both mount axes move to track the star means that the pointing corrections change more, and in a more complicated way, than for the equatorial.

If the telescope has an autoguider, a guide star can be found by trial and error, or the probe $[x, y]$ can be predicted from the science target and guide star positions using the text-book formulas for "standard coordinates" $[\xi, \eta]$ and multiplying by the focal length. And, when the guide star image drifts away from that $[x, y]$, you can adjust the $[\alpha, \delta]$ that you are tracking to bring the field back to the right place.

2.2. Complications

The simple approach just described can work quite well, and probably represents how most modern automated telescopes are controlled. But there are a few potential limitations:

1. If there is an instrument rotator, and the instrument is not accurately centered on the rotator axis, pointing errors will be introduced whenever the rotator angle is changed. Moreover, the instrument may have more then one place onto which targets are to be imaged. Rather than moving the image around by changing the target $[\alpha, \delta]$, and having to make fresh adjustments whenever the rotator is moved, it is better if new targets can be acquired straight onto the nominated "pointing origin" (*i.e.* the desired $[x, y]$), directly from the slew or after preliminary acquisition using a viewing device. Logged "World Coordinate System" (WCS) information may be inaccurate if the location of the detector in the focal plane is not being taken into account, or if the target $[\alpha, \delta]$ has been adjusted to achieve some guiding or offsetting effect.

2. The required rotator angle may not follow the parallactic angle: there may be unwanted field rotation effects. This happens near the zenith for an altazimuth, near the pole for an equatorial, in the presence of certain types of pointing corrections. The problem is especially acute when autoguiding; the guide star will remain fixed while the science target trails along an arc centered on the guide star. Instrument rotators at Nasmyth and coudé introduce additional complications.

3. When autoguiding, differential refraction and atmospheric dispersion will change the $[x, y]$ at which the guide star image is expected. If these effects are not allowed for, the target image will be trailed.

4. The pointing model may contain many terms; a telescope with multiple top-ends or focal stations will have more than one such model; and the model may evolve as experience grows. The control software must therefore be flexible enough to accept a model that is dynamic, both in form and in terms of coefficient values, preferably without the need to recompile code. Every term in the pointing model has the potential to cause numerical or geometrical difficulties at the zenith (or pole). Arithmetic faults, such as divide by zero or square roots of negative numbers, may occur. Pointing-correction formulas are often approximate and become progressively less accurate as the zenith (or pole) is approached. Whereas every telescope orientation corresponds to a well-defined direction in the sky, there may be places on the sky, near the zenith or pole, that are inaccessible; these blind-spots have to be handled in some orderly way. In addition, some mounts have a "below the pole" or "beyond the zenith" configuration, creating ambiguous zones that the mount can reach in two different ways; a further complication is that certain pointing corrections flip sign between the two states.

5. Subtle incompatibilities with the arrangements for pointing-test analysis can easily creep in. The order in which the pointing terms are calculated can affect the result to second-order, and if the pointing tests have to be done well off axis (with a guider probe for example) the usual approximate formulas may not be adequate. A particular area of difficulty is where different terms interact with each other and cannot simply be calculated separately and added up.

6. Any lack of rigor may produce disproportionate effects in cases where the chain of transformations is traversed in different directions and the results are brought together. In general, there is a need for complete "pointing integrity" that pervades the entire system and ensures precise agreement between all the separate parts—tracking, guiding, offsetting, chopping, nodding and so on.

Although some of the above complications are addressed in existing telescope control systems, such as those on Keck, WHT, UKIRT, TNG, VLT and Gemini, the new algorithm solves them all.

3. A NEW ALGORITHM

3.1. The Virtual Telescope

In common with a number of existing designs, the central concept in the new algorithm is the "virtual telescope" (VT). This term was coined by the implementors of the AAT control system[3] in the 1970s, to mean a software object that behaves like a perfect telescope, with the messy and irrelevant details of the pointing transformations and corrections hidden. This "information hiding" applied not only to the user-interface but also to how application programs controlled the telescope. A program that wanted to offset the telescope by a precise $[\Delta\alpha, \Delta\delta]$ could simply present those numbers and the telescope would go there, automatically allowing for differential refraction and pointing errors *etc.*

The concept was later extended, by this author and others,[4] beyond simply pointing and tracking the main telescope. In particular, it was recognized that autoguiders could be interpreted as separate VTs running in parallel, all perforce sharing the same mount but differing markedly in other respects, such as the target star and the image location in the focal plane. The Gemini telescope control system (TCS) went even further, using 13 VTs to control the tip/tilt secondary (including chopping) plus three autoguiders.

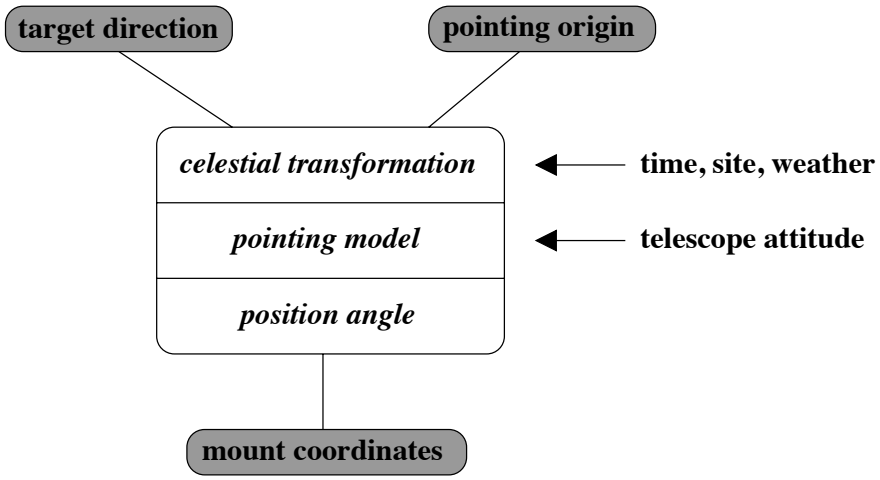

Figure 1. The Virtual Telescope.

The "Virtual Telescope" is a software object that hides the complicated details of all the pointing transformations and corrections and presents an idealized telescope to the user and to application programs. Three separate functions are supported, supporting three end-points: target direction in the sky, mount encoder readings and pointing-origin (image $[x, y]$); knowing any two of these, plus additional information such as the rotator angle, the third can be calculated. For example, tracking consists of predicting, for a given instant of time, the mount attitude that will cause an image of the specified $[\alpha, \delta]$ to fall on the specified $[x, y]$.

Figure 1 shows the VT concept as used in the present paper. A time- and attitude-dependent transformation links the target direction, the pointing origin and the mount coordinates; from any two of these the third can be found. The most important application is tracking, where a fixed $[\alpha, \delta]$ is imaged on a fixed $[x, y]$ by setting the mount to the appropriate $[Az, El]$ or $[h, \delta]$; time is steadily changing, and hence the mount coordinates move from one iteration to the next, so tracking the target. But, equally well, the transformation can predict the $[\alpha, \delta]$ for a given $[x, y]$ and *vice versa*, for a given mount attitude. All three of these modes are rigorously supported by the pointing algorithm described here.

3.2. Coping with Different Sorts of Telescope

It is still usual to write the control system for a new telescope *ab initio*, and so concepts such as "altazimuth mount" tend to be built into the design. However, successive designs of pointing kernel have tried to isolate these factors, relegating them to the status of system parameters. The main reason for generalizing in this way is to facilitate software re-use. A less obvious benefit is that it requires greater rigor and hence avoids some of the dangers listed in Section 2.2.

The new algorithm fully generalizes the two-axis "gimbal" mount. In the rest of this paper, "hour angle, declination" and "azimuth, elevation" are given the generic names "roll, pitch". Specifically, roll and pitch correspond to $[-h, \delta]$ for an equatorial and $[\pi - Az, El]$ for an altazimuth. These two types of mount are, in fact special cases: the algorithm supports any mount attitude. The mount attitude defines a frame (the "mount frame") whose z-axis is the mount roll axis and whose x-axis is at right-angles to the pitch axis for roll=0; the y-axis is at right-angles to the other two. (Note that the roll/pitch non-perpendicularity is not involved. Indeed, the new algorithm allows any angle between the axes, not merely a nominal 90°.)

A further respect in which the new algorithm has been generalized is that the location of the instrument rotator is selectable, depending upon which part of the structure the rotator is mounted:

- *OTA:* the instrument mount is fixed to the telescope tube. This option embraces Cassegrain, bent Cassegrain, prime focus, Newtonian and so on. (OTA stands for "Optical Telescope Assembly". The term is used here in a completely generic sense: it could equally well be the dish of a radio-telescope.)
- *Nasmyth:* the instrument mount moves with roll but not pitch.
- *Coudé:* the instrument mount is fixed to the ground.

It should be understood, however, that any additional moving parts (extra optics *etc.*), over and above mount roll and pitch, are not allowed for.

4. THE TRANSFORMATION CHAIN

Figure 2 shows the chain of transformations that implement the virtual telescope. Previous diagrams of this sort, for instance Figure 1 in reference [5], start with the target $[\alpha, \delta]$ and, after a series of transformations and corrections, finish with the demanded mount angles. This is the obvious approach, and a perhaps surprising feature of the new algorithm is that the mount angles appear in the *middle* of the chain. The reason for this layout is that the problem has been broken down into three independent pieces:

- The transformation between the target coordinates and the direction of the incoming light in the mount frame (see 3.2). The latter is called the **aim** vector.
- The direction of the pointing direction in a frame fixed to the pitch axis. This is called the **boresight** vector and is usually close to $[1, 0, 0]$.
- The "posture matrix", a rotation matrix that links **aim** and **boresight** by two large rotations (the mount roll/pitch angles) and one usually small rotation (the non-perpendicularity between the two axes).

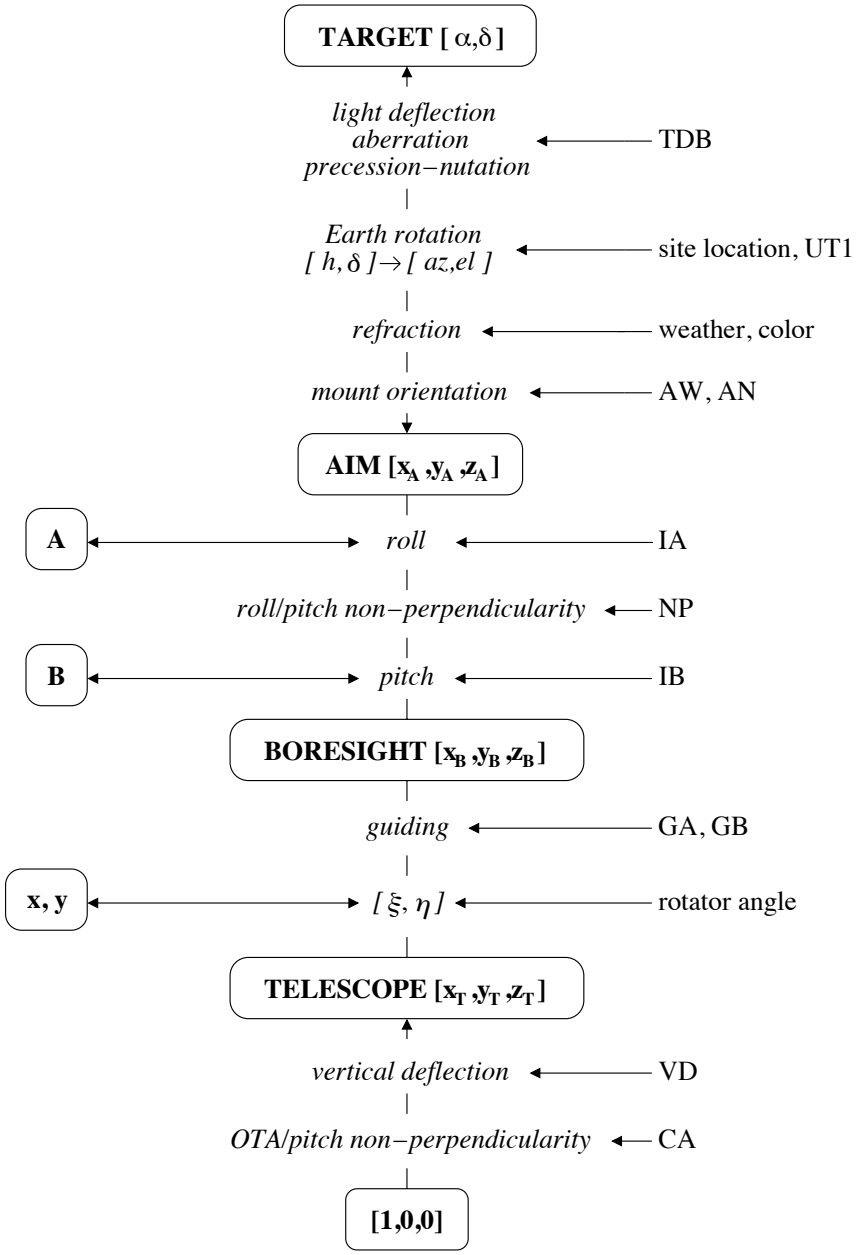

Figure 2. The chain of pointing transformations.

The diagram shows the sequence of transformations and adjustments that make up the telescope pointing algorithm. The entities in boxes are vectors representing directions in various frames, while the items in italics are the operations that link the vectors. The virtual-telescope facilities that implement this sequence support three end-points: target direction in the sky $[\,\alpha, \delta\,]$ (usually), mount encoder readings $[\,a, b\,]$ and pointing-origin $[\,x, y\,]$; knowing any two of these, plus additional information such as the rotator angle, the third can be calculated. The telescope's pointing model, typically 10-20 terms, is compressed before use into a 7-term summary, the terms being IA, IB, VD, CA, NP, AW and AN.

4.1. The Pointing Model

Note that the pointing model used in the transformation chain is an idealized one, consisting of only seven terms: two give mount attitude, two are non-perpendicularities, one is a vertical displacement and two are zero-point errors. How are they related to the real pointing model, which may have a large number of terms in addition to (or even instead of) those described so far?

4.1.1. The traditional approach

The pointing model consists of perhaps 10–20 mathematical expressions, each associated with a numerical coefficient, and each generating corrections $[\,\Delta Az, \Delta El\,]$ or $[\,\Delta h, \Delta\delta\,]$. The terms are summed and then added to the ideal $[\,Az, El\,]$ or $[\,h, \delta\,]$ to give the mount demands. As a rule, the expressions themselves are hard-coded into the control software, but the coefficients are downloaded from a file when the system starts.

The scheme is in most respects satisfactory. Recalibrated coefficients can immediately be introduced following a pointing test, or even at run-time, and multiple models can be implemented simply by setting the coefficients of unused terms to zero. However, several of the problems listed in Section 2.2 remain: (i) revising the form of the model involves code changes, (ii) every pointing term has to be defensively coded to avoid arithmetic problems, (iii) interactions between the terms are not accounted for, and (iv) incompatibilities with the pointing analysis phase may creep in.

4.1.2. The new algorithm

In the new approach, only the seven terms already described are implemented in the pointing kernel, with each individual term in the full pointing model contributing to whichever of the seven basic terms seems most natural. Whereas the traditional approach (above) forces each pointing effect to be expressed as if it were a contribution $[\Delta a, \Delta b]$ to the index errors, the new approach additionally offers the other five basic terms as candidates for the role. This simplifies the mathematics and promotes rigor.

The new scheme minimizes spurious interaction between the pointing terms by separately accumulating the various combinations and only then evaluating the net effect of the seven-term "summary". In the traditional approach, there is a particular problem with the so-called "collimation" terms, all of which have the effect of moving the place in the focal plane on which the image should fall. These include the two non-perpendicularities as well as the pointing origin $[\,x, y\,]$ itself. If the various components are separately computed, each with complicated pole or zenith handling, the results are inaccurate and may even be unstable (given the changing attitude of the mount and rotator). In the new algorithm, the two non-perpendicularities and the pointing origin all play distinct roles in an integrated solution.

Compatibility between the pointing algorithm and offline pointing-analysis tools can be guaranteed by sharing the relevant code, a technique pioneered on the ARC telescope.[6] This not only guarantees complete consistency of mathematical treatment, but also enables currently unused terms to be present in the control system ready to be activated if and when required, without code changes. The TCSpk pointing kernel offers the option of sharing code with the well-known TPOINT analysis tool, and simulated pointing observations obtained with TCSpk can be analyzed with TPOINT to recover the original pointing model to milliarcsecond accuracy.

4.1.3. The seven terms

All seven generic terms in the new algorithm correspond to well-known terms in existing pointing models. For example, the TPOINT pointing analysis tool has terms for equatorial and altazimuth telescopes that match the generic terms as follows:

term	*generic*	*equatorial*	*altazimuth*
roll index error	IA	−IH	IA
pitch index error	IB	ID	IE
vertical deflection	VD	FLOP	FLOP
OTA/pitch nonperp	CA	−CH	CA
roll/pitch nonperp	NP	−NP	NPAE
roll axis...	AW	−MA	AW
...misalignment	AN	ME	AN

Most terms in the TPOINT repertoire do not of course have a 1:1 correspondence with the generic terms. In each such case, a choice of generic term has been made to receive the correction, based on the likely mechanical causes. For example, the TPOINT "tube flexure" term TF adds an increment of $\mathrm{TF}\sin\zeta$ to the vertical deflection VD.

Three of the generic terms, IA, IB and NP, have individual roles in the pointing algorithm. Another two, the roll axis misalignment terms AW and AN, are used to form the "mount attitude matrix" (see 4.2.2). The final two terms, CA and VD, are used to form the **telescope** vector (see 4.2.4 and 4.2.14). The translation is, as a rule, telescope-attitude-dependent and hence time-dependent. It is up to the telescope control application to decide when to refresh the translation; typically this is done every few seconds.

4.1.4. Guiding

Guiding means making *ad hoc* adjustments to the telescope tracking. It is a somewhat problematical aspect in any pointing scheme because it addresses something that is not supposed to be there: a discrepancy between the scheme and reality. The traditional remedy is to apply small changes to the target $[\alpha, \delta]$. However, this is wrong in principle if the target coordinates are, in fact, accurately known. Some advanced telescope control systems can also offer the option of adjusting the desired image $[x, y]$; again, this may not correspond to reality.

The approach taken in the new algorithm is that guiding adjustments $[GA, GB]$ are small changes to the telescope pointing model, specifically changes to the position of the $[x, y]$ origin in the focal plane. Further details are given in Section 4.2.8. It is important to understand that all three methods of moving the telescope—$[\alpha, \delta]$, $[x, y]$ and $[GA, GB]$—are needed in order to achieve different effects, and that it is the operator's responsibility to make the right choice. Adroit user-interface design is essential here.

4.2. The Transformations in Detail

Because the algorithm supports multiple functions, in particular the three offered by the virtual telescope concept, it would be misleading to set out here a single-pass mathematical development with a single starting point and a final result. Instead, the algorithm is best seen as a set of components that can be strung together in different ways. At any one time, a telescope control application may be taking several different paths through the algorithm, to implement tracking, guiding and rotator control for example, and this will involve different sets of components applied in different orders. Each of the following sections describes one such component. Defensive coding measures (*e.g.* protection against divide-by-zero) are omitted from the descriptions.

Note that, internally, $[Az, El]$ vectors are in a right-handed frame with its x-axis pointing south. Some care is needed to keep this convention separate from that used for all external purposes, namely left-handed with zero azimuth in the north.

4.2.1. Nasmyth and coudé rotator angles

The pointing algorithm deals principally with the case where the instrument rotator is fixed to the OTA, for instance at the Cassegrain focus. Writing θ for the Cass-equivalent rotator angle that the algorithm requires, the following table shows how this angle is obtained from θ_M, the actual mechanical angle (corrected for any index error):

location	θ
OTA	θ_M
Nasmyth left	$\theta_M - b$
Nasmyth right	$\theta_M + b$
coudé right	$\theta_M + a - b$
coudé left	$\theta_M + a + b$

where a and b are the mount roll and pitch angles. The sign convention for both θ and θ_M is that a positive change makes the projection of the focal plane on the sky rotate counter-clockwise.

4.2.2. Formation of the mount attitude matrix

The mount attitude matrix is the rotation matrix that transforms $[\,Az, El\,]$ to the "mount frame" (see 3.2). It comprises a 3-D rotation into the nominal mount frame (*e.g.* equatorial or altazimuth), followed by adjustments to account for misalignment of the mount's roll axis. The nominal mount attitude matrix is:

$$\mathbf{M_0} = R_1(x)R_2(y)R_3(z)$$

where the three Euler angles x, y, z depend on mount type. For an altazimuth mount they are all zero, for an equatorial mount at latitude ϕ the y rotation is $\phi - 90^\circ$ with the other two angles zero, and for a generalized gimbal the three angles can be anything. The two adjustments AN and AW, which may include time- and attitude-dependent contributions, are applied, forming the final mount attitude matrix:

$$\mathbf{M} = R_1(+\mathrm{AW})R_2(-\mathrm{AN})\mathbf{M_0}$$

4.2.3. Relationship between target coordinates and AIM vector

The mount attitude matrix, above, is part of the transformation between the target coordinates and the **aim** vector. The other parts allow for the standard positional-astronomy effects, including precession-nutation, aberration, light deflection, Earth rotation and refraction. The precise way this is done is not specific to the new pointing algorithm. The TCSpk kernel in fact adopts the "osculating transformation matrix" approach described in reference [5], which has certain advantages including computational efficiency and rigorous inversion. We will from now on concentrate on the transformations between the **aim** vector, which is the line of sight in the mount frame, the mount angles (*i.e.* encoder readings) and the image position.

4.2.4. The TELESCOPE frame

The **telescope** vector describes the pointing change arising from OTA/pitch non-perpendicularity CA and vertical deflection VD. The **boresight** vector additionally takes into account the pointing-origin and guiding adjustments. Both are expressed in a frame that is fixed to the pitch axis of the mount in such a way that when the pitch angle is zero the x-axis, which represents the nominal OTA, is at right angles to both the roll and pitch axes and the y-axis points along the pitch axis. Consequently, a telescope that has small collimation error and vertical deflection has a **telescope** vector close to [1,0,0].

4.2.5. Composition of the posture matrix

The posture matrix rotates the **boresight** vector so as to align it to the **aim** vector:

$$\mathbf{P} = R_3(-a)R_1(+\mathrm{NP})R_2(+b)$$

where a and b are the roll and pitch angles and NP is the roll/pitch non-perpendicularity.

4.2.6. Focal-plane coordinates to/from standard coordinates

Coordinates $[\,x, y\,]$ in the rotating focal plane can be transformed into "standard coordinates" as follows:

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \frac{1}{F} \begin{bmatrix} -\cos\theta - \sin\theta \\ +\sin\theta - \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

where F is the focal length in the same units as x and y. The transformation inverts trivially.

4.2.7. Index errors

The encoder index errors $[\,\mathrm{IA}, \mathrm{IB}\,]$ relate the true mount angles $[\,a, b\,]$ to the encoder readings $[\,a_{_E}, b_{_E}\,]$:

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a_{_E} \\ b_{_E} \end{bmatrix} + \begin{bmatrix} \mathrm{IA} \\ \mathrm{IB} \end{bmatrix}$$

4.2.8. Guiding corrections

Section 4.1.4 discussed how guiding corrections are introduced into the pointing model as adjustments to the $[\,x, y\,]$ zero point. The corrections $[\,\mathrm{GA}, \mathrm{GB}\,]$ are applied to the (non-rotating) standard coordinates $[\,\xi, \eta\,]$ as follows:

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} \xi_0 \\ \eta_0 \end{bmatrix} - \begin{bmatrix} \mathrm{GA} \\ \mathrm{GB} \end{bmatrix}$$

where $[\,\xi_0, \eta_0,\,]$ are the unadjusted standard coordinates.

4.2.9. TELESCOPE to BORESIGHT

This transformation allows for pointing origin—the $[\,x, y\,]$ coordinates of the place in the focal plane at which the image is located—and guiding corrections $[\,GA, GB\,]$. The first step is to use procedure 4.2.6 to transform $[\,x, y\,]$ to standard coordinates $[\,\xi_0, \eta_0\,]$ followed by procedure 4.2.8 to apply the guiding adjustments. Finally, $[\,\xi, \eta\,]$ is combined with the **telescope** vector to form the **boresight** vector:

$$\begin{bmatrix} x_{_B} \\ y_{_B} \\ z_{_B} \end{bmatrix} = \begin{bmatrix} x_{_T} - (\,\xi\, y_{_T} + \eta\, z_{_T}\, x_{_T})/r \\ y_{_T} + (\,\xi\, x_{_T} - \eta\, z_{_T}\, y_{_T})/r \\ z_{_T} + \eta r \end{bmatrix} / \,(1 + \xi^2 + \eta^2)^{1/2}$$

where

$$r = (x_{_T}^2 + y_{_T}^2)^{1/2}$$

4.2.10. Decomposition of the posture matrix

The **aim** and **boresight** vectors $\vec{A}$ and $\vec{B}$ are related through the posture matrix $\mathbf{P}$:

$$\vec{A} = \mathbf{P}\vec{B}$$

Section 4.2.5 defined $\mathbf{P}$ in terms of three Euler angles: the roll a, the roll/pitch non-perpendicularity NP, and the pitch b. Given $\vec{A}$, $\vec{B}$ and NP, the pitch can be determined by decomposing $\mathbf{P}$ as follows:

$$b = \arctan\left(\frac{z_{_A} + \sin \mathrm{NP}\; y_{_B}}{\pm\,(x_{_A}^2 + y_{_A}^2 - y_{_B}(\,2\, z_{_A}\, \sin \mathrm{NP} + y_{_B}) - \sin^2 \mathrm{NP}\,)^{1/2}}\right) - \arctan\left(\frac{z_{_B}}{x_{_B}}\right)$$

Note that there are up to two solutions. These correspond to such cases as east and west of the pier on a "German" or cross-axis equatorial mount, or "beyond the zenith" on certain altazimuth designs. For each pitch solution, there is a matching roll:

$$a = \arctan\left(\frac{y_{_A} x - x_{_A} y}{x_{_A} x + y_{_A} y}\right)$$

where

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \; R_1(\mathrm{NP}) R_2(\,b\,) \begin{bmatrix} x_{_B} \\ y_{_B} \\ z_{_B} \end{bmatrix}$$

Near the roll axis there may be no solutions, producing blind spots.

4.2.11. AIM vector to standard coordinates

One of the three VT functions is to determine where in the focal plane the image of a specified target will fall. The core transformation is from **aim** vector to standard coordinates $[\,\xi, \eta\,]$. The first step is to obtain the posture matrix **P** using 4.2.5 and use it to transform the **aim** vector $\vec{A}$ to the **boresight** vector $\vec{B}$:

$$\vec{B} = \mathbf{P}^T \vec{A}$$

The standard coordinates are then obtained by using the **telescope** vector $\vec{T}$:

$$\begin{bmatrix} \xi \\ \eta \end{bmatrix} = \begin{bmatrix} (y_B\, x_T - x_B\, y_T) \\ (z_B(x_T^2 + y_T^2) - z_T(x_B x_T + y_B y_T) \end{bmatrix} / \,(\vec{B} \cdot \vec{T})$$

4.2.12. Encoder readings to AIM vector

Another VT function is to determine where in the sky the telescope is pointing. The core transformation in this case is encoder readings $[\,a_E, b_E\,]$ to **aim** vector given a **boresight** vector. The first step is to eliminate the encoder index errors using procedure 4.2.7. The resulting mount angles $[\,a, b\,]$ and the roll/pitch non-perpendicularity NP allow the posture matrix to be constructed using procedure 4.2.5. The **aim** vector is then obtained from the **boresight** vector $\vec{B}$ using:

$$\vec{A} = \mathbf{P}\vec{B}$$

4.2.13. AIM and TELESCOPE to pointing-origin

The remaining VT function is to predict where in the focal plane the image of a specified target will appear, given the encoder readings. The preliminary steps are to correct for index errors and to obtain the **aim** vector (see 4.2.7 and 4.2.3). Procedure 4.2.11 then yields the standard coordinates of the image, which can be freed from any guiding corrections using procedure 4.2.8. Finally, the $[\,\xi, \eta\,]$ is scaled and rotated into $[\,x, y\,]$ (see 4.2.6).

4.2.14. Formation of the TELESCOPE vector

The **telescope** vector captures the combined effect of the OTA/pitch non-perpendicularity CA and the vertical deflection VD.

Dealing with VD requires an estimated $[\,Az, El\,]$ for the OTA, so that the mapping of the vertical onto mount coordinates can be established. The result does not depend critically on the precise $[\,Az, El\,]$ over most of the sky, but problems can arise near the pole of the mounting, where there is a rapid change in the orientation of the vertical. In fact the problem is not confined to VD; similar chicken-and-egg difficulties arise when dealing with any operational pointing term that is a function of pointing direction. However, at the pole of the mounting there are so many other difficulties that some sort of avoiding action is inevitably needed anyway. Moreover, on altazimuth mounts, the most common case for large telescopes, the VD problem is extenuated because the contributing pointing terms (TF for instance) tend to be small at the zenith. In any case, once a **telescope** vector has been adopted, the pointing algorithm functions rigorously; the problems, such as they are, are confined to where the 7-term summary is refreshed (see 4.1.3).

The first step is to obtain the $[\,Az, El\,]$ coordinates for the undeflected OTA:

$$\begin{bmatrix} A_0 \\ E_0 \end{bmatrix} = \begin{bmatrix} A \\ E \end{bmatrix} + \begin{bmatrix} 0 \\ \mathrm{VD} \end{bmatrix}$$

where $[\,A, E\,]$ is the supplied $[\,Az, El\,]$ for the deflected OTA. The deflected $[\,Az, El\,]$ is then transformed into a unit vector:

$$\vec{V} = \begin{bmatrix} \cos A\, \cos E \\ \sin A\, \cos E \\ \sin E \end{bmatrix}$$

The undeflected vector $\vec{V}_0$ is obtained in a similar way. The two vectors are then transformed into **aim** vectors by using the mount attitude matrix **M**:

$$\vec{A} = \mathbf{M}\vec{V}$$

and similarly for $\vec{A}_0$.

The **telescope** vector for the undeflected OTA depends only on the OTA/pitch non-perpendicularity CA and is formed thus:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos \mathrm{CA} \\ \sin \mathrm{CA} \\ 0 \end{bmatrix}$$

Procedure 4.2.10 is then used to obtain the mount roll/pitch angles that relate this undeflected **telescope** vector to the undeflected **aim** vector $\vec{A}_0$, given the roll/pitch non-perpendicularity NP, for an on-axis pointing origin and in the absence of encoder index errors and guiding corrections. This allows procedure 4.2.10 to be used to express the displacement of $\vec{A}$ with respect to $\vec{A}_0$ as a point $[\xi, \eta]$ in the non-rotating focal plane. This $[\xi, \eta]$ is used to displace the undeflected **telescope** vector to form the final **telescope** vector:

$$\begin{bmatrix} x_T \\ y_T \\ z_T \end{bmatrix} = \begin{bmatrix} x - \xi\, y - \eta\, z\, x \\ y + \xi\, x - \eta\, z\, y \\ z + \eta \end{bmatrix} / \left(1 + \xi^2 + \eta^2\right)^{1/2}$$

4.3. Controlling the Mount and Rotator

The procedures set out above can be interconnected to provide pointing integrity throughout a complicated system that can perform complex tasks such as autoguiding while chopping on a solar-system target. But the best examples of the kernel in action are the more basic functions of controlling the principal moving parts: the two axes of the mount, and the instrument rotator.

Generating mount demands is extremely straightforward. The target coordinates are transformed, for some time in the near future, into an **aim** vector, using 4.2.3, which is then transformed into encoder demands using 4.2.10.

Many otherwise competent altazimuth control systems are let down by their parallactic-angle-based rotator predictions, which fail to take account of field rotation effects near the zenith due to pointing corrections in azimuth. A much better method, first used in the Gemini TCS, is to take two trial points either side of the pointing-origin, along a specified line, and, knowing the predicted mount angles for the time in question, projecting them onto the sky (see 4.2.12 and 4.2.3). The resulting coordinates yield the position-angle of the specified line on the sky, which is then compared with that desired and the current rotator demand adjusted accordingly. The position angle θ from unit vector $\vec{V}_0$ to vector $\vec{V}_1$ can be obtained from:

$$\theta = \arctan\left(\frac{y_1\, x_0 - x_1\, y_0}{z_1\, (x_0^2 + y_0^2) - z_0\, (x_1\, x_0 + y_1\, y_0)}\right)$$

4.4. Further Considerations

The algorithm described here is only a small part of a TCS, albeit one that may have a disproportionate effect on the character and performance of the final system. The entire control system is much too large a topic to be discussed here, but it is worth at least listing some of the issues that TCS developers have to address:

- The building blocks described here have to be suitably interconnected to form a full-blown TCS kernel supporting multiple guiders, steerable optics, several focal stations *etc.* as required. The system developed for the SOAR telescope[7] demonstrates one approach, using Tcl and C++. The system needs a real-time architecture —how the clock is handled, how the different parts of the calculation are scheduled, and whether interpolation schemes are to be used to enhance efficiency.
- A user-interface must be provided. It must support the three types of guide paddle (target, instrument, pointing), multiple hot-spots and all the other kernel capabilities.
- The algorithm assumes "pinhole camera" optics. Any departures from tangent-plane projection will require additional steps.

- Ancillary optics such as atmospheric dispersion compensators and image rotators will need special treatment. A related issue is direct support of instrument focal-plane coordinates (pixels for instance).
- There must be provision for pointing calibrations, both full-scale and start-of-night.
- The TCS may need to support chopping, nodding and other features that involve special pointing behaviour. The Gemini TCS design, for examples, uses software "filters" to feed information from one VT to another and thereby achieve special effects such as slow offsetting. Facilities for offsetting, dithering, scanning, sequencing and so on may also be required.
- Arrangements for accepting inputs from weather sensors are needed for the refraction calculations. These may involve averaging, glitch prevention, manual intervention and so on.
- The range of supported coordinate systems must be decided. Most future TCSs need provide control only in ICRS $[\,\alpha, \delta\,]$ and topocentric $[\,Az, El\,]$.
- The pointing kernel must be able to export WCS information, from which data headers can be generated.
- The interface between the TCS kernel and the mount (and rotator) servos must be carefully planned, including the precise arrangements for interpolation/extrapolation and the provision of any required "signal shaping" (trajectory control).
- The kernel includes no handling of cable wraps. It is left to the application to decide when to "unwrap".
- On some mounts, management of alternative postures is an essential element. The most common case is the German equatorial, which spends equal time on either side of the pier.
- Facilities for tracking solar-system objects may be needed. This may consist simply of providing differential tracking rates or (better) direct naming of objects and/or use of orbital elements.
- Some zenith/pole handling is needed. TCSpk offers a "radius of avoidance" feature, overriding the target coordinates to execute a track around rather than through the difficult region. Other schemes are possible. Some telescopes may call for Sun or Moon avoidance when slewing.

ACKNOWLEDGMENTS

I am grateful to the SOAR telescope project for supporting the presentation of this paper.

REFERENCES

1. P. T. Wallace, "The SLALIB library," in *Astronomical Data Analysis Software and Systems III, Volume 61*, D. R. Crabtree, R. J. Hanisch, and J. Barnes, eds., pp. 481–484, Astronomical Society of the Pacific, 1994.
2. G. H. Kaplan, J. A. Hughes, P. K. Seidelmann, C. A. Smith, and B. D. Yallop, "Mean and apparent place computations in the new IAU system. III – apparent, topocentric, and astrometric places of planets and stars," *Astron. Journ.* **97**, p. 1197, 1989.
3. J. O. Straede and P. T. Wallace, "The Anglo-Australian 3.9 meter telescope: software controlled slewing, setting and tracking," *Publ. Astr. Soc. Pacific* **88**, pp. 525–535, 1976.
4. J. Bailey, "The portable telescope control system project," in *Telescope Control Systems II, Volume 3112*, H. Lewis, ed., *Proceedings of SPIE*, pp. 124–131, SPIE, 1997.
5. P. T. Wallace, "Pointing and tracking algorithms for the Keck 10-meter telescope," in *Instrumentation for Ground-Based Optical Astronomy*, L. B. Robinson, ed., *The Ninth Santa Cruz Summer Workshop in Astronomy and Astrophysics*, pp. 691–706, Springer-Verlag, New York, 1987.
6. R. Owen, W. Siegmund, and C. Hull, "The control system for the Apache point 3.5-meter telescope," in *Instrumentation for Ground-Based Optical Astronomy*, L. B. Robinson, ed., *The Ninth Santa Cruz Summer Workshop in Astronomy and Astrophysics*, pp. 686–690, Springer-Verlag, New York, 1987.
7. D. L. Terrett, "Tcl as a software environment for a TCS," in *Advanced Telescope and Instrumentation Control Software II*, H. Lewis, ed., SPIE, Washington, 2002.