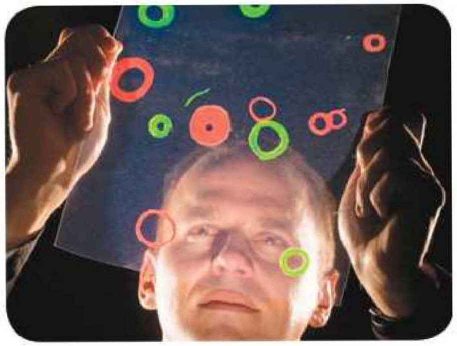
SAND2017-11281PE

[redacted]

[redacted]



Tonopah Test Range Optical Tracking TSPI Uncertainty Quantification

TJ Williams

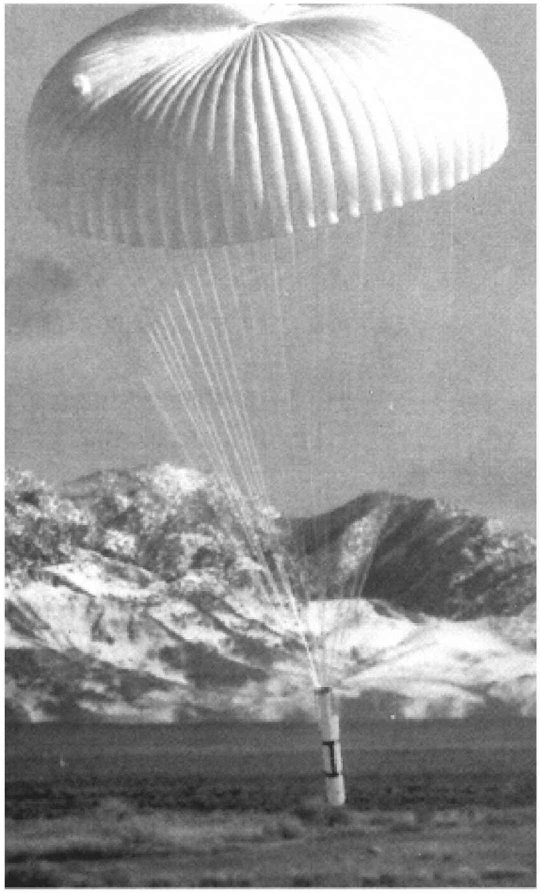
Tim Miller

TTR Target Tracking Data

- Testing at TTR often involves drops of various test articles.
- TTR uses a system of optical trackers and software triangulation to develop Time-Space-Position Information (TSPI) of the flight path of the test article.
- Seven cinetheodolite mounts, spread across a series of locations throughout the range.
- Custom code to do the calibration and triangulation.

How Good is the TSPI Data?

- TTR has generally said their data is accurate to 3 feet in every direction for normal geometries.
- This has been tested since the 1980s by tracking fixed target lights on poles at very well-known locations.
  - Issues:
    - All near zero elevation angle
    - At the same altitude as the mounts
    - Not moving
    - Point sources
- Need a better way to estimate error.
  - Just looking at the software doesn't do it, you need to understand the az-el errors coming from the mounts.



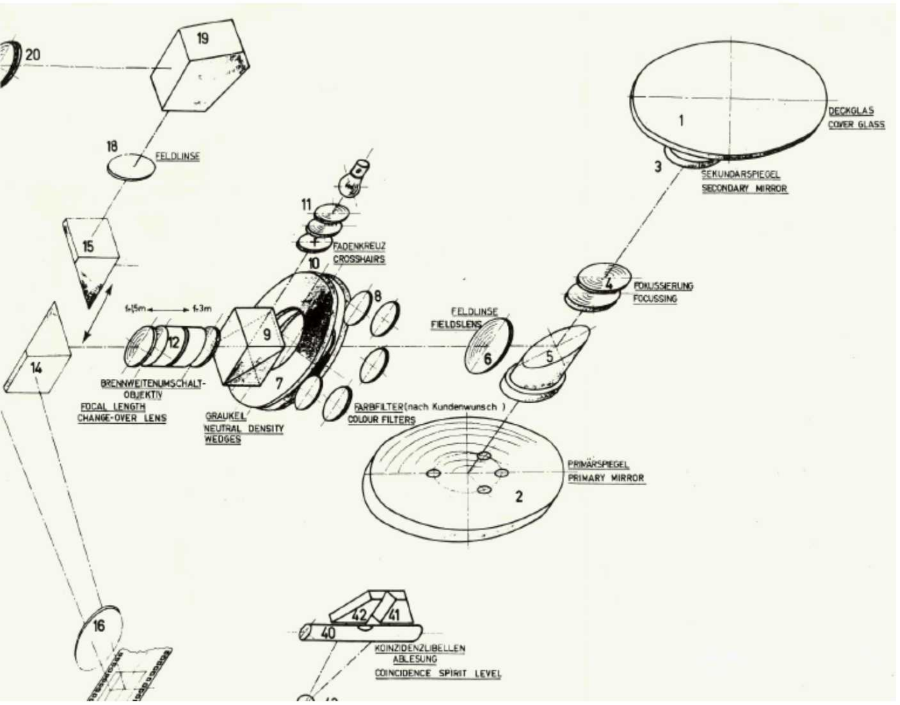
The Current Approach Estimates the Error with a Simulation

- Model-based analysis.
  - Implement all of the TTR software algorithms
  - Create a mathematical model of a cinetheodolite and its environment with randomized parameters.
  - Combine the hardware model and the software in a way that reflect TTR’s system and run the resulting simulation in Monte Carlo fashion to create statistical data.
    - Imitate the full operation of the system, including pre and post-test observations of calibration targets.
- Extremely complex multi-disciplinary problem
  - Software engineering, mechanical engineering, optics, astronomy, image processing, atmosphere modeling, statistics, etc.
- If both the software implementation and the model are *validated* than we can have confidence that the statistical errors produced by the model reflect the uncertainty in the actual TTR TSPI solutions.

TTR Software was Implemented in Matlab

- Four primary programs for calibration and triangulation
  - All written in FORTRAN in the mid-1980s.
  - I/O through ASCII text files.
  - Run from command line.
- Scope of the Effort
  - 161 functions and subroutines.
  - ~8000 lines of code.
- I manually turned this into a set of 95 Matlab functions.
  - In slightly simplified form: the original is set up to do batches of frames. I just needed one at a time.
- Note: The function Optxyz (both in FORTRAN and Matlab forms) uses a least squares algorithm to solve the triangulation problem, and thus also estimates the error volume from the linear fit (giving an estimate of the size of all non-correlated errors). This will be important later when it comes to validating the entire approach.

The Cinetheodolite Model Has Significant Detail



DECKGLAS COVER GLASS
SEKUNDARSPIEGEL SECONDARY MIRROR
FELDLINSE
FADENKREUZ CROSSHAIRS
FOKUSSIERUNG FOCUSSING
FELDLINSE FIELDSLENS
BRENNWEITENUMSCHALT-OBJEKTIV FOCAL LENGTH CHANGE-OVER LENS
GRAUKEIL NEUTRAL DENSITY WEDGES
FARBFILTER (nach Kundenwunsch) COLOUR FILTERS
PRIMÄRSPIEGEL PRIMARY MIRROR
KOINZIDENZLIBELLEN ABLESUNG COINCIDENCE SPIRIT LEVEL

- 48 model parameters
  - 40 of these are error parameters
- Basic Parts
  - Rotations via direction cosine matrices for the gimbal angles.
  - Errors in the mount's positioning and leveling.
  - Other angle rotations for errors from: atmospherics and mechanical errors.
  - In small angle space: errors in FPA mounting, track algorithms, and timing.
- Two types of random draws:
  - Those characterizing what is *stable* in a particular instantiation of the mount model.
  - Those characterizing what changes look to look, frame to frame (I call these "jitter").
- Basic model was *verified* through peer review
  - Karl Schrader of Dept. 5783
  - Dave Denning of Dept. 5782

Model Parameters in Detail

- Position errors: x, y, z
- Rotation errors: roll, pitch, yaw
- Pointing errors: Az, El
- Atmospherics
- Telescope sag error
- Telescope Alignment errors: Az, El
- Gimbal Bearing Errors: 1 Az Bearing, 2 El Bearing terms
- Axis nonperpendicularity error
- Encoder random error: Az, El
- Encoder scalefactor errors: Az, El
- FPA Rotation error
- FPA IFOV errors: x, y
- Track algorithm errors:
  - Along-track bias, along-track std, along-track jitter, cross-track bias, cross-track std, cross-track jitter.
- Timing errors
  - FPA timing bias, FPA error std, FPA timing jitter, encoder sample time std, encoder sample time jitter
- Residual un-modeled errors
  - Az and El std, Az and El jitter
- Non-error parameters
  - Mount height above base
  - Atmospheric pressure and temperature
  - Telescope aperture and F-number
  - FPA pixel pitch, integration time, and center wavelength

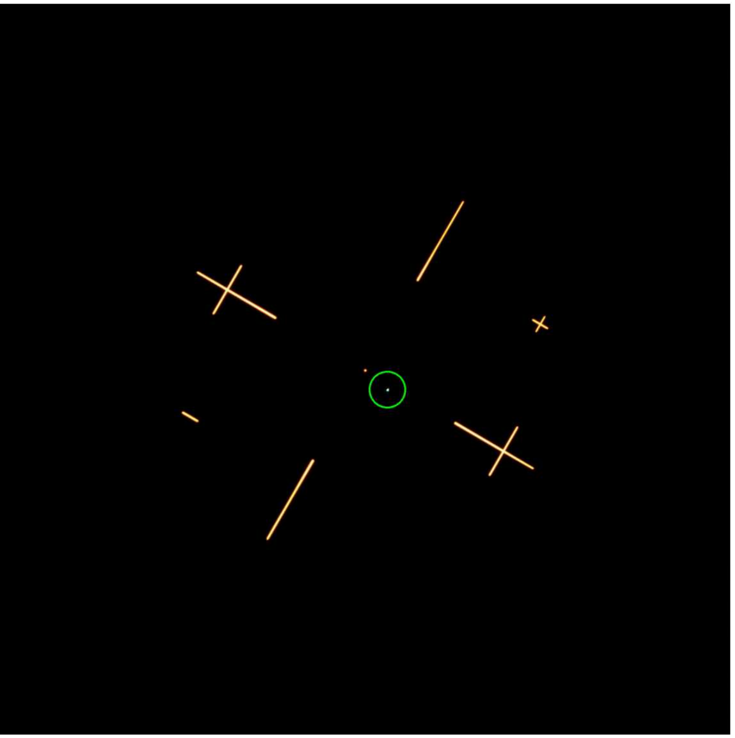
Populating the Error Parameters and Validating the Model is Problematic

- Model validation for this sort of mount is difficult, as discussed above on slide 4.
  - Need targets with extremely well-known positions that match the characteristics of the test targets.
    - Elevation angle, altitude, speed, visual character, etc.
  - Can't get all of those in one place, not in the necessary quantities.
- So, break the problem into pieces
  - Create a validated mount model from a star calibration.
  - Measure atmospheric scintillation.
  - Model altitude/refraction effects from validated science.
  - Model target shape/tracking effects from validated science.
  - Model speed effects with known parameters of the timing systems.
  - Implement all of these into the model and then verify the results.

Star Calibration: Taking the Data

- Trip to TTR in February 2015
- Used a Python script running the PyEphem package to predict true star positions.
- Assisted the TTR staff in recording observations of 40 different stars using Mount 3 at Station 28.
  - Down to a visual magnitude of 2.8. Stopped because of time, not because the system couldn't see even dimmer stars.
  - Well spread-out in the sky.
  - Also did several looks at R&L targets and at planets.
- Saved the imagery, recorded az/el encoder angles, and IRIG time ~2000 frames of data.
- PyEphem gave us the truth positions, including estimated atmospheric refraction.

Star Calibration: Processing the Images



- Have to find the star/planet/light in thousands of images to create corrections to the encoder angles.
- Permanent reticle etched into the glass of the telescope – present in all the images.
- Wrote an image processing algorithm to mask out the reticle, and then centroid track the target.
- Issues with some images:
  - Sometimes the target left the image.
  - When looking at the ground targets there is often other lights present in the image.
  - Sometimes the reticle smears.
  - So…ignore those images and move on.
    - Left with 1462 frames to work with from 49 different "points"
- For those images it worked on, find the star, measure its location, de-rotate the coude, apply the offsets to the encoder angles, and subtract truth to find the pointing error.

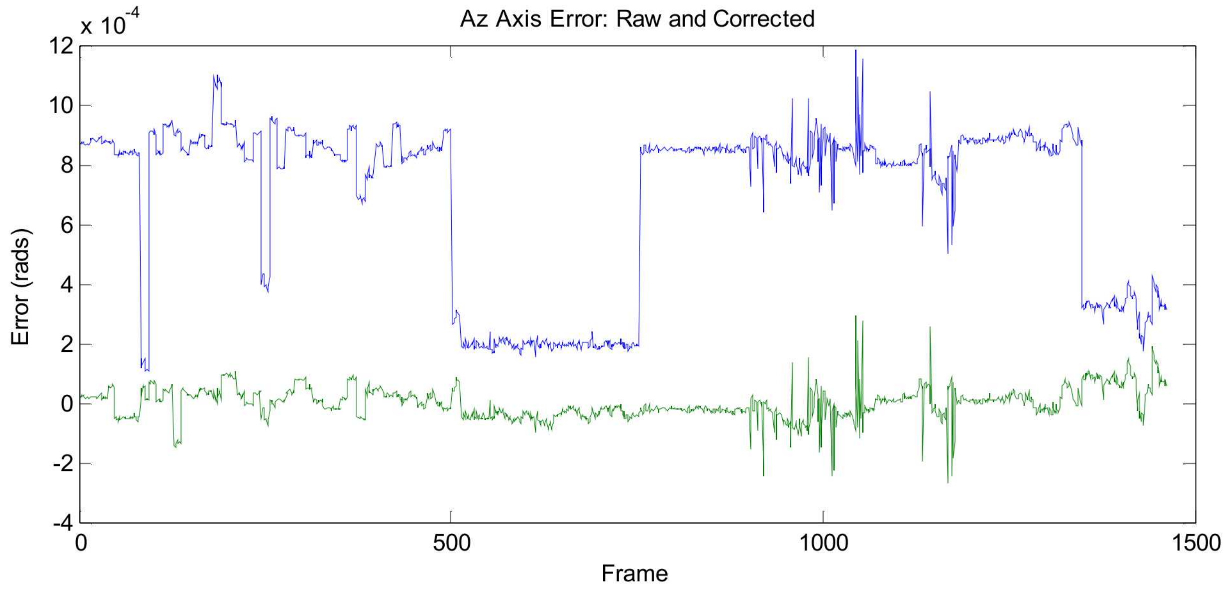
Star Calibration: Creating the Mount Model

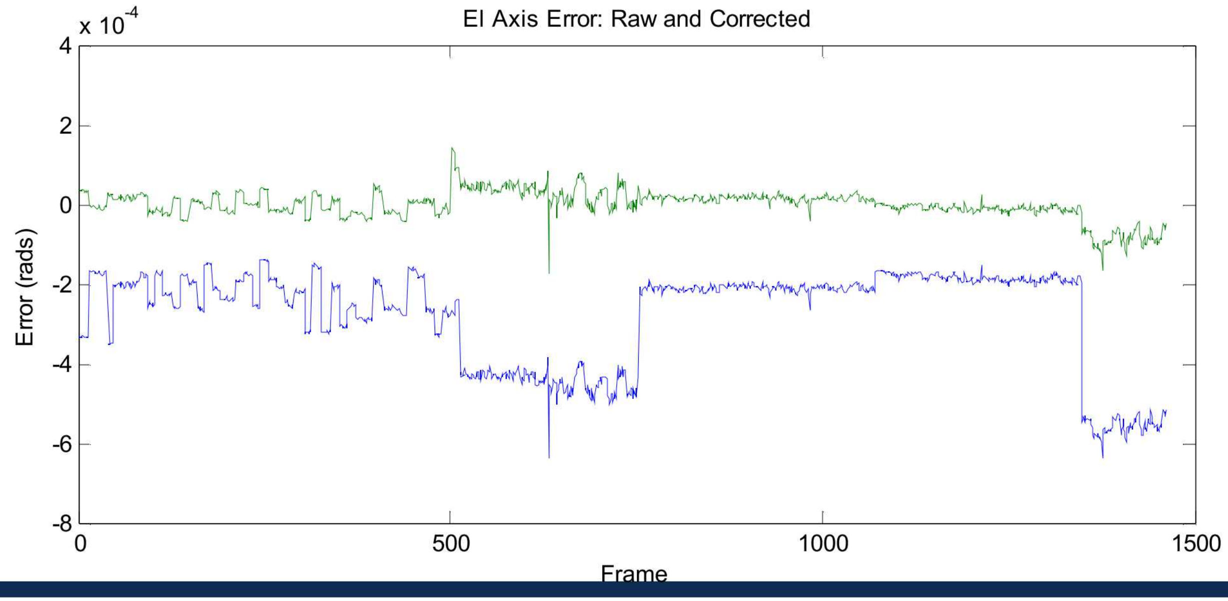
- Following the procedure outlined in Meeks [4], a linear model was created based on 16 error parameters
  - Went through several iterations to figure out which parameters were necessary and well-behaved.
  - Example: There was a problem with initial approach because in the Azimuth axis, several of the terms appear to not be independent.
    - If varied independently, the error explodes.
    - Typical telescope design methods would make them dependent on each other.
  - Solution: Add a term into the model to account for their shared component, other terms for what was not shared in each.
- The data from the star data analysis provided the inputs to a matrix least-squares algorithm to find these parameters.
  - Find the errors in each of 1462 frames: ΔP_A, ΔP_E
  - Calculate X_A and X_E based on the effect each parameter has on each axis, and the appropriate values for each frame.
  - Find the coefficient vector, β, by solving $\begin{bmatrix} \Delta P_A \\ \Delta P_E \end{bmatrix}_i = \begin{bmatrix} \mathbf{X_A} \\ \mathbf{X_E} \end{bmatrix}_i [\boldsymbol{\beta}] + \boldsymbol{\varepsilon}_i$ through OLS.
  - Also calculate residual, un-modeled, errors.

Star Calibration: Mount Model Details

With Lens Test 4						
Error	Symbol	Axis	Factor	Azimuth Effect	Elevation Effect	Calculated Value
Az Correlation Parameter				1 - .76*sin E + 1.064*tan E - 1.085*sec E + .1*cos E	0	0.013635199
El Encoder Offset	E0	y	1	0	1	-0.00094683
Az Encoder Scale	A'	z	A	A	0	-1.1068E-05
El Encoder Scale	E'	y	E	0	E	0.000440357
Base Roll (ytilt)	R	x	1	sin A tan E	cos A	2.64043E-05
Base Pitch (-xtilt)	P	y	1	cos A tan E	-sin A	-7.09606E-05
Azimuth bearing, EW term	ξB4	y	cos A	cos A tan E cos A	-sin A cos A	1.94632E-05
Elevation bearing, cos term	ξB2	z	cos E	cos E	0	-1.85327E-05
Elevation bearing, sin term	ξB1	z	sin E	sin E	0	-5.79433E-05
Axis Nonperpendicularity	η	x	1	tan E	0	-1.05753E-05
Tube Flexure	δ	y	cos E	0	cos E	0.000551905
Optical axis misalignment, z	ɣz	z	1	sec E	0	-7.79307E-06
Delta IFOV x				-x0*sinE*secE	x0*cosE	3.50437E-09
Delta IFOV y				-Y0*cosE*secE	-y0*sinE	1.74374E-08
FPA Rotation Error				C*y*tan E - C*x	-C*x*sin E - C*y*cos E	-0.001815763

Star Calibration: Performance





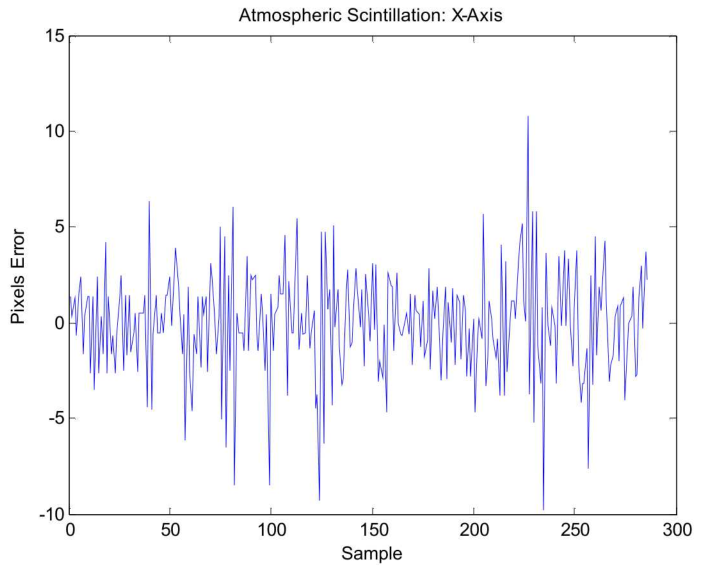
Pre cal delta RMS	8.06E-04
Post cal delta RMS	6.35E-05
Percent Post to Pre	7.87%

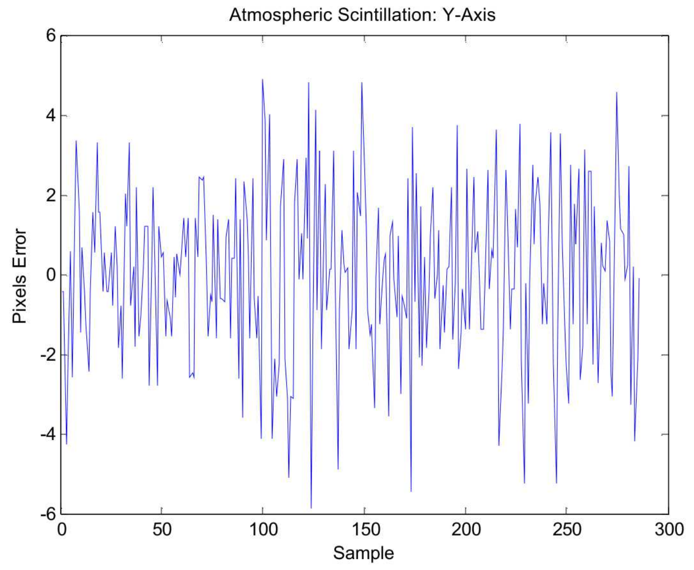
Atmospheric Scintillation Measurement

- While at Tonopah, attempted to perform an Intrinsic Optical System Calibration using an 8’ x 8’ check-patterned target board.
  - Had to set it up at 1 km distance.
  - Couldn’t get the data to work because of the extent of atmospheric scintillation.
  - But, that lets us measure the scintillation.
- Used the software package Open CV to find the corners of the checkers.
- Measured how the corners varied within the frame and also frame to frame.
- Used that to size the atmospheric scintillation.

Atmospheric Scintillation Estimation

- Std X = 2.78 pixels ; Std Y = 2.07 pixels
- This ought to be worst case, at 1 km.
  - It was at a near zero elevation angle, well after dawn.
- In general, the target is at farther range, but much higher elevation angle, and earlier.
  - Chose to use 10 pixels as the standard deviation in the model.
  - Any higher and the scintillation begins to dominate everything else and pull the linear model errors higher than what TTR measures.





Parallactic Refraction Model

- When observing things through changes in altitude, the air will refract the light, causing an error in measured zenith angle.
- Nearly all the literature is on *astronomical* refraction – looking at objects outside the atmosphere at infinite distance.
- Looking at something at near distance – especially in the atmosphere – requires analyzing *parallactic* refraction, which is much harder.
  - Path dependent.
  - I found two good papers from 40-50 years ago: [5] and [6].
  - In general it requires an iterative solution.

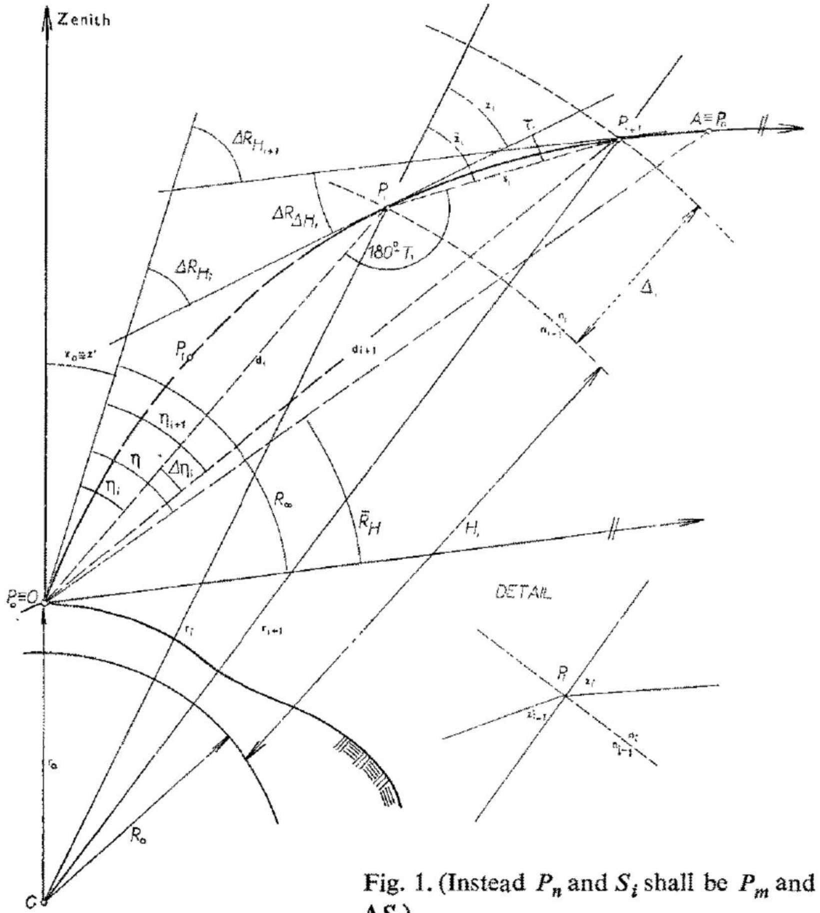


Fig. 1. (Instead P_n and S_i shall be P_m and ΔS_i).

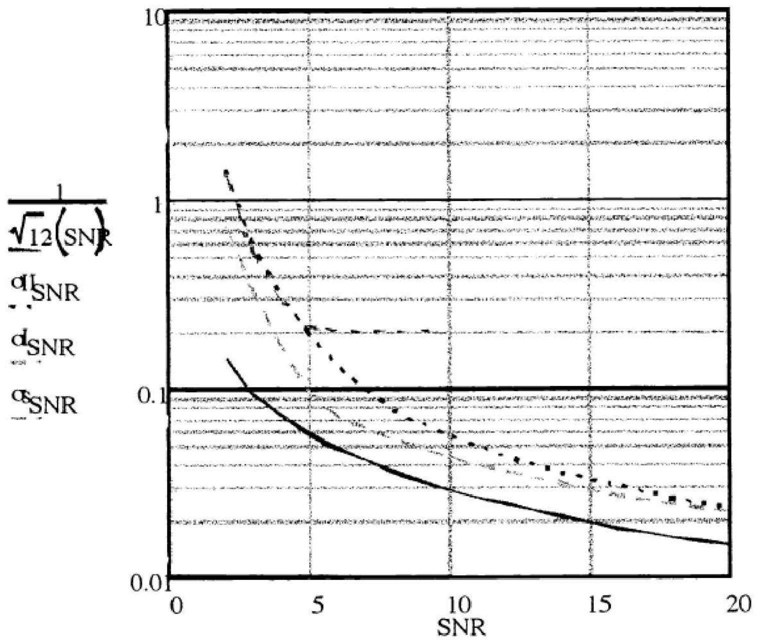
Parallactic Refraction Model Accuracy

- Can only *verify* the model, based on the information in the papers cited.
  - Fortunately, Kabelac's second paper [6], quotes the results from several other researchers.
  - I've compared my answers to these.

	10 km Altitude			20 km Altitude		
Zenith Angle:	60°	70°	80°	60°	70°	80°
Oterma	58.3"	91.8"	183.4"	37.1"	58.4"	117"
Baldini	94.6"	147.4"	273.6"	47.6"	74.1"	137.6"
Karsky	72.3"	115"	236.3"	36.4"	57.9"	118.9"
Kabelac	61.4"	96.8"	191"	39.3"	62.0"	123.7"
Stand, Earth	55.8"	87.1"	164.4"	34.9"	54.4"	102.8"
Schmid	78.7"	124.3"	250.7"	39.4"	62.3"	125.9"
Veis	55.3"	86.8"	168.4"	34.9"	54.5"	104.1"
My Model	60.7"	97.2"	212"	40.7"	65.2"	144.1"

  - Elevated numbers at 20km stem from a simplification I made in how to estimate the air density vs altitude.
    - The points of interest for this model are all at less than 12 km altitude.

Nose Track NEA Model



- TTR selects the track point in the imagery by hand.
  - Giving an error estimate of the process is problematic
- BUT, there exist good error models for standard implied edge track algorithms.
  - Noise Equivalent Angle (NEA) models.
  - See reference [7]
- If you know your Signal-to-Noise Ratio, the track jitter can be estimated.
  - Assume at worst case, SNR ~5
  - So, a jitter of 0.2 pixels.

TSPI Uncertainty Quantification

- Verified combined model by looking at the error estimates from the Triangulation Linear Regression.

Run	Mean Sdx (m)	Mean Sdy (m)	Mean Sdz (m)	Mean Errvol (m^3)
Results from Actual Test	0.38	0.33	0.34	0.04
Model at Test Point	0.38	0.34	0.35	0.05

- Run a Monte Carlo series against a point from a TTR test where knowledge of the TSPI error is desired.

Run	Mean X Error (m)	Mean Y Error (m)	Mean Z Error (m)		Std X Error (m)	Std Y Error (m)	Std Z Error (m)		Mean Sdx (m)	Mean Sdy (m)	Mean Sdz (m)	Mean Errvol (m^3)
Test at Point 1									0.94	1.95	1.89	1.67
Model at Point 1	-0.02	-0.13	1.79		0.97	1.43	1.17		0.90	1.29	1.48	1.87
Test at Point 2									0.85	1.31	1.52	1.06
Model at Point 2	-0.05	-0.12	1.68		0.98	1.30	1.14		0.83	1.06	1.37	1.39
Test at Point 3									0.61	0.88	1.10	0.44
Model at Point 3	-0.03	-0.09	1.61		0.99	1.26	1.12		0.80	0.97	1.33	1.24

- Mean Error is X and Y are near zero; mean error in Z is relatively large – atmospheric refraction
- Standard deviations are between 1 and 1.5 meters.

Bibliography

1. Leland Johnson, “Tonopah Test Range: Outpost of Sandia National Laboratories”, SAND96-0375, March 1996
2. Marcus Bunting and Ray Zazworsky, “Investigation of a Tonopah Test Range Target Position Estimation Problem”
3. “Desktop Procedures for Optical Data Reduction at the Tonopah Test Range (TTR)”, May 2000
4. R.L. Meeks, “Sources of uncertainty in telescope pointing models”, *Proc. Of SPIE* Vol. 5497, pg. 140-148
5. Josef Kabelac, “Atmospheric Models and Astronomical and Parallactic Refraction, *Studia Geophysica et Geodaetica*, Vol 11, No. 1, 1967, pg. 1-20
6. Josef Kabelac, “The Parallactic Refraction Determined with the Aid of Meteorological Balloon Data”, *Studia Geophysica et Geodaetica,* Vol 20, 1976
7. Steven L. Chodos, “NEA Model for Implied Edge Nose Tracking Algorithms”, *Acquisition, Tracking, and Pointing XV,* Proceedings of SPIE, Vol 4365, 2001, pg. 74-87

Backup: Verification and Validation

The Software Implementation Has Been Validated

- The code was validated by running through the inputs from a set of test cases provided by TTR and checking the data at all intermediate steps.
  - Code has the same outputs to the precision limits of the data.
- Six calibration points and three track points - each run through seven different mounts.
  - Checked their results at four different stages.
    - R&L target azimuth and elevation after con_calib.
    - All eight calibration parameters from con_camco.
    - Track target azimuth and elevation from con_calib_track.
    - Corrected azimuth and elevation after con_correct.
- Then the three track points from the seven mounts were combined in optxyz
  - Checked the resulting X,Y,Z estimates, as well as the estimated SDX, SDY, SDZ, ERRVol, and 14 skew data variables..

Mount Model Implementation Verification

- Set the hardware model parameters to the coefficients from the star cal with no additional error sources (and no random draws).
- Run through all the different star points and measure the calculated error
  - No jitter on this one because I'm just verifying the linear parameters.
- Compare the resulting error with what the linear parameters alone - $\begin{bmatrix} \mathbf{X}_A \\ \mathbf{X}_E \end{bmatrix}_i [\beta]$ - accounted for.
  - Both mean and median errors (across the 49 different star/target points) were accurate to within 0.05%

	Az % Diff	El % Diff
Mean	0.02%	0.00%
Median	0.05%	0.01%

Random Mount Model Verification

- Do each mount model parameter one at a time.
- For each parameter do a random draw from a normal distribution with standard deviation set to the value from the star calibration.
- Run several instances for each star point and then compute the mean error and compare with what the linear model gave for each parameter.
- Rescale the standard deviation of the draw until the mean and std match what was seen was expected – in general this meant multiply by 1.25.
  - That Az Correlation parameter was the exception – had to divide by 18.
- Then do the same comparison to the linear data for all the parameters at once.
  - My correlation parameter in azimuth worked well – all the errors superimpose.
  - The errors in Elevation did not superimpose - there must also some sort of correlation in parameters, but I could never get the star cal matrix to be non-singular for such a parameter.
  - Rescale the primary Elevation-axis parameters to get the total error correct.

	Model/Linear Az	Model/Linear El
Mean	140.32%	100.62%
Median	101.34%	96.12%

- This *verifies* that the model implements the linear parameters from the star cal correctly.

Mount Model Validation

- Prep work
  - Each star “point” involved multiple frames (10 to 200) at the same star, taken by the cinetheodolite at 10 Hz.
  - So, for each point, determine the mean and standard deviation of the error from the recorded (not simulated or linearized) data.
- Mount Model Validation
  - Add in the random error sources and residual errors (as random draws) that would have been present during the star observations.
  - Run 10000 runs per star point (100 mount instantiations x 100 jittered frames.
  - Calculate the mean and standard deviation of each point and compare with the actual measured test data, as reduced above.

	Mean Az	Std Az	Mean El	Std El
Mean	134.01%	130.74%	101.33%	102.11%
Median	100.52%	119.14%	96.34%	97.42%

Simulation Verification at Test Point

- Implement all these models and values in the simulation.
  - But how to know that this simulation is accurate?
- Now go back to the test case from the software validation.
  - Set the mounts up in identical positions, with the same R&L calibration targets.
  - Put the target at the x,y,z point in space that TTR estimated.
  - Monte Carlo the model: 10000 runs.
- Remember: the TTR triangulation software uses a least-squares algorithm to do the estimate, and this also estimates the error volume – this characterizes the (uncorrelated) noise on the measurements.
  - Find the average of these error terms as estimated by the triangulation algorithm across my 10000 runs.
  - If the input errors are of a similar nature, these terms ought to correspond – roughly – to what the code found in the actual test (averaged across the first 21 frames).

Run	Mean Sdx (m)	Mean Sdy (m)	Mean Sdz (m)	Mean Errvol (m^3)
Results from Actual Test	0.38	0.33	0.34	0.04
Model at Test Point	0.38	0.34	0.35	0.05