Ralph Geometric Distortion and Boresights

by Allen Lunsford
January 20, 2007

Timing

The start time (MET) of the first exposure of all Ralph observations is stored in the FITS keyword STARTMET. The SOC computes this value by adjusting the time of the observation command by the appropriate offset, depending on instrument and observation mode. The exposure time for each frame of all Ralph observations is stored in the FITS keyword RALPHEXP (seconds).

The time of the start of each subsequent frame in a Ralph observations can be calculated from the STARTMET and RALPHEXP values, depending on instrument and observation mode (Table 1).

Instrument-Mode	Start time (MET) for frame F
MVIC-TDI	STARTMET + F * RALPHEXP / 32
MVIC-FRAME	STARTMET + F * (RALPHEXP + 3.1535)
LEISA-SUB	STARTMET + F * RALPHEXP
LEISA-RAW	STARTMET + F * RALPHEXP

Table 1: Exposure start time for Ralph frames

Instrument Mode

The instrument and mode for a Ralph observation can be determined by the SOC filename prefix, or by keyword values in the FITS header (Table 2). For convenience, the TDI-Index value is introduced. This is the arrangement of the TDI channels in a spatial sense, with Index 0 (Near IR) being the channel at the leading edge of a TDI scan.

				FITS Keywords			
Instrument-Mode	Channel	SOC Prefix	TDI-Index	MODE	CPLANE	FILTER	PANCCD
MVIC-TDI	Near IR	mc2	0	2	2	NIR	
MVIC-TDI	Methane	mc3	1	2	3	CH4	
MVIC-TDI	Red	mc0	2	2	0	RED	
MVIC-TDI	Blue	mc1	3	2	1	BLUE	
MVIC-TDI	Pan 1	mp1	4	3	0		1
MVIC-TDI	Pan 2	mp2	5	4	0		2
MVIC-FRAME		mpf		1	0		
LEISA-SUB		lsb		7			
LEISA-RAW		lrw		8			

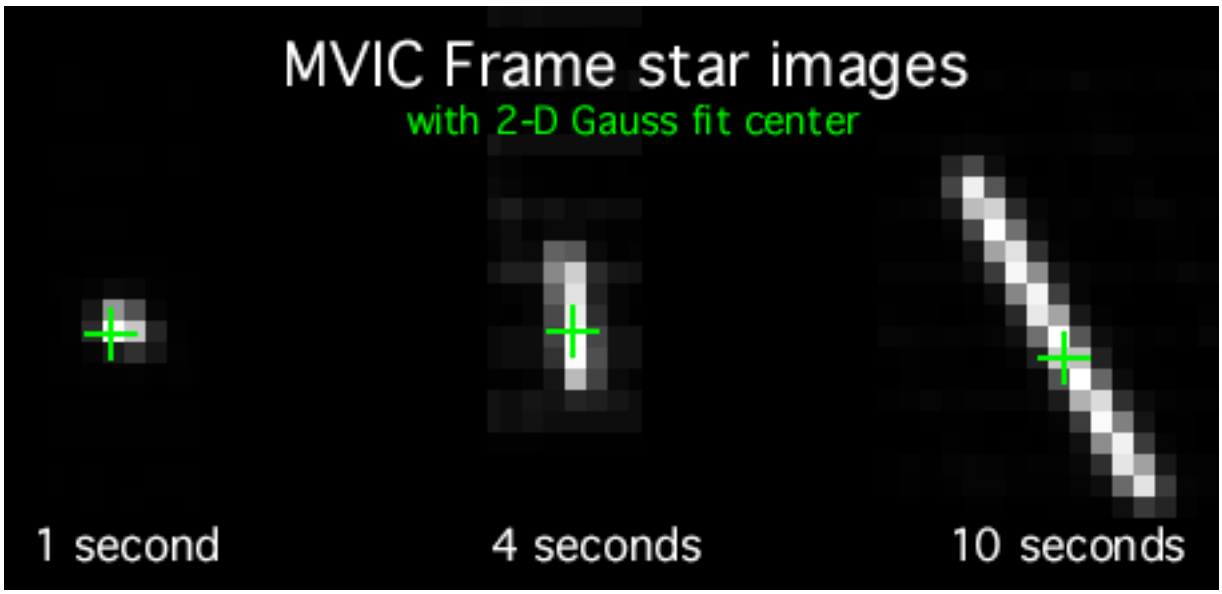
Table 2: FITS keywords for Identifying Ralph data types

Method

Data taken during the Ralph-006 (Observing M6, M7, and Procyon),Ralph-007 (Observing Jupiter), Ralph-008 (Observing M6 for Ralph/Lorri Co-alignment), Ralph-009 (Observing Uranus), and Ralph-017 (Observing asteroid 2002-JF56) were used to analyze the pointing performance of the MVIC and LEISA instruments. Boresight offsets and optical distortion information for each instrument and channel were derived from this analysis.

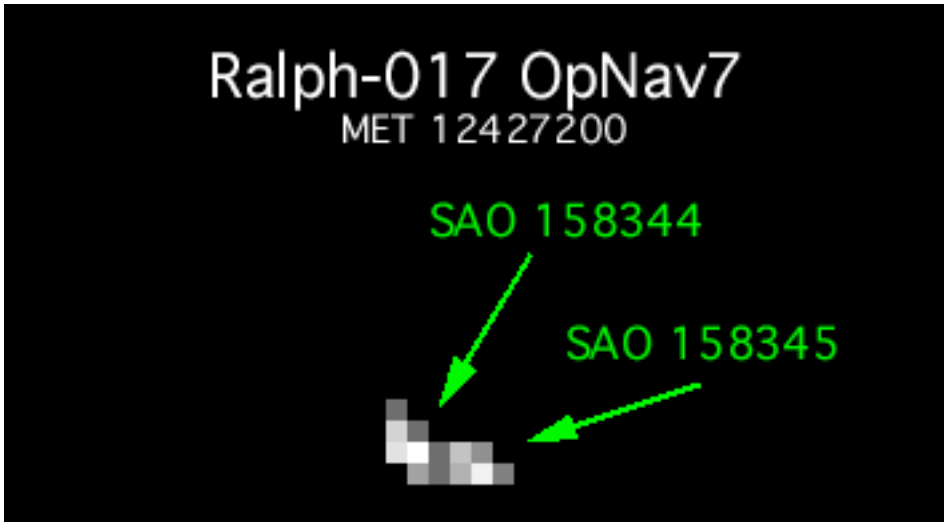
MVIC Pan Frame

For each observation frame, a predicted star centroid (and Jupiter in the case of LEISA) was calculated using the Spacecraft pointing quaternion provided by SPICE. The pixels surrounding the star were extracted from the image and a 2-d Gaussian was fit to the subframe. The difference between the center of the gauss fit and the predicted image position is interpreted as an error in the column or row direction of the image pointing (a delta row, or delta column).



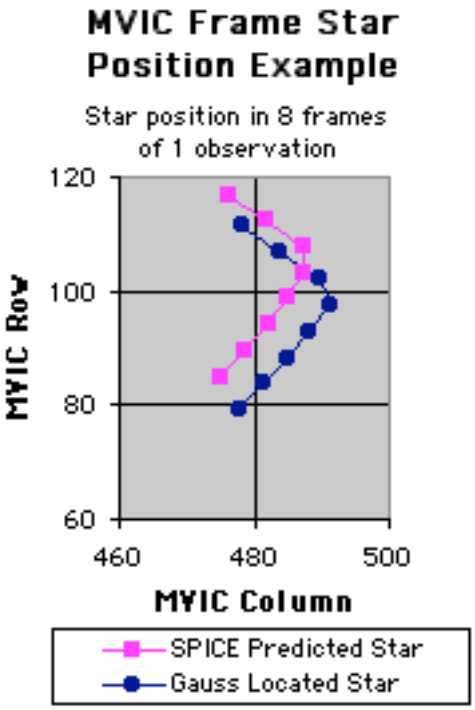
The MVIC Framing mode produces multiple images of nearly the same point, but only a relatively small sliver of coverage. The spacecraft attitude wanders within a pointing deadband and the position of each star moves about in the MVIC image. The movement of the spacecraft causes stars to be streaked for longer integration times. The gauss fit still does a good job determining the center of the star image.

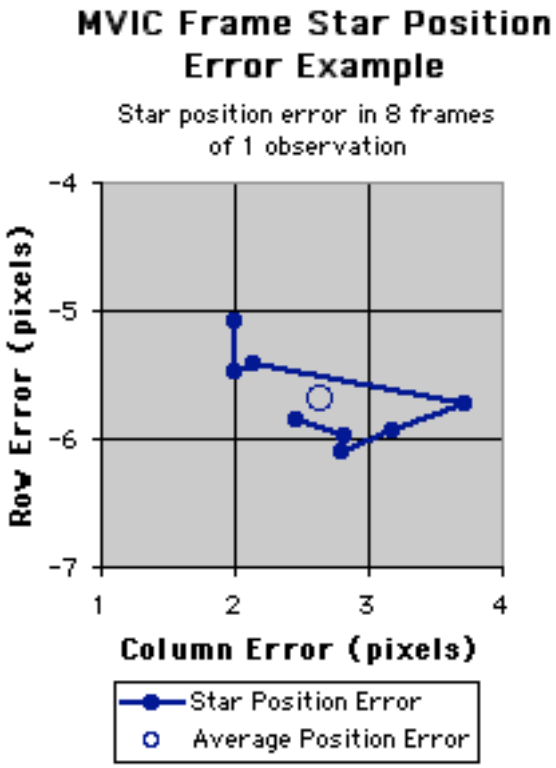
Noise bursts, or multiple stars in the sub-image would cause the 2-d Gauss function to produce a false result. Efforts were made to eliminate these anomalies from the analysis, but it was not possible to review every star image.



As a star moves about in the MVIC image, the difference between the predicted position and the Gauss position changes. There are many effects that may contribute to this difference: interpolation of spacecraft pointing to the image time, uncertainty in the spacecraft quaternion (attitude and rate), image noise.

Looking at a single star reveals a good correlation between the predicted and located position, with some offset. The plot of Row vs. Column position error suggests that the position error wanders around this offset.



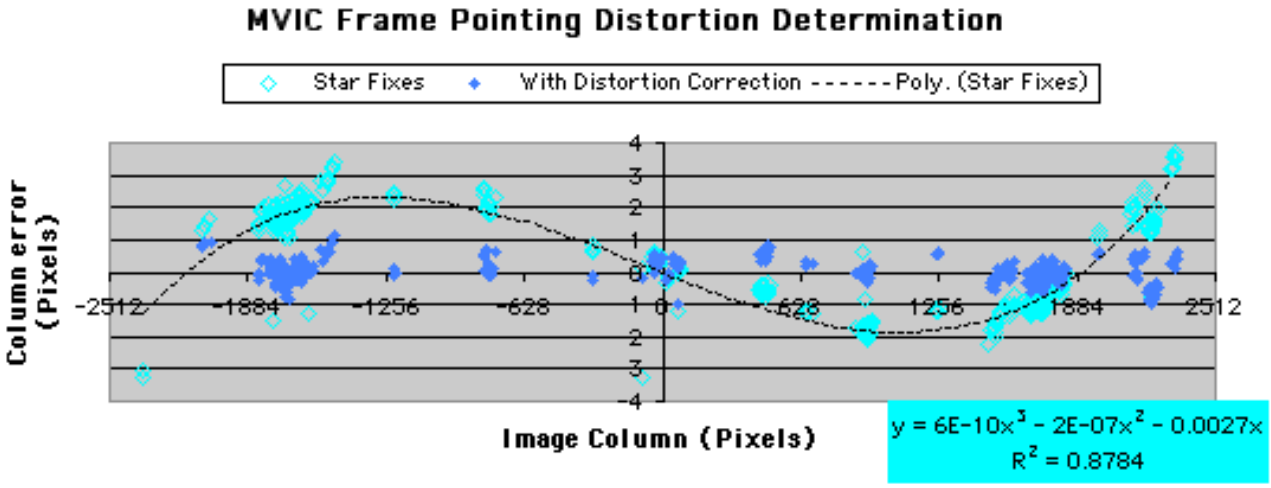


The boresight offset for the MVIC-Frame observation mode was determined from ground testing. This boresight definition does a good job of predicting pointing position near the middle of the image.

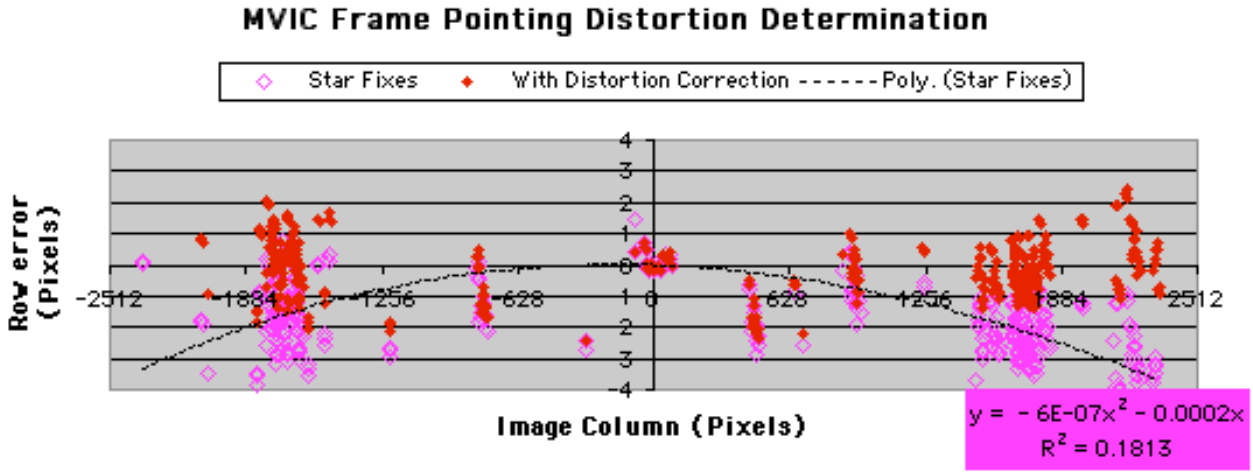
For the initial distortion model analysis, all the Pan-Frame observations from Ralph-006 (targeting M6 & M7) with integration time of one second or more were used. The gauss determined positions of each star in each frame in each observations were compiled and analyzed. There were a total of 38 stars that returned reliable positions. With the multiple frames and observations, 354 star fixes were used in the initial calculation of the distortion model.

The difference between the predicted positions and the gauss determined positions in the row and column direction were fit to polynomial functions verses column. The MVIC Frame image has only 128 rows. The row and column error as a function of row was to small to derive.

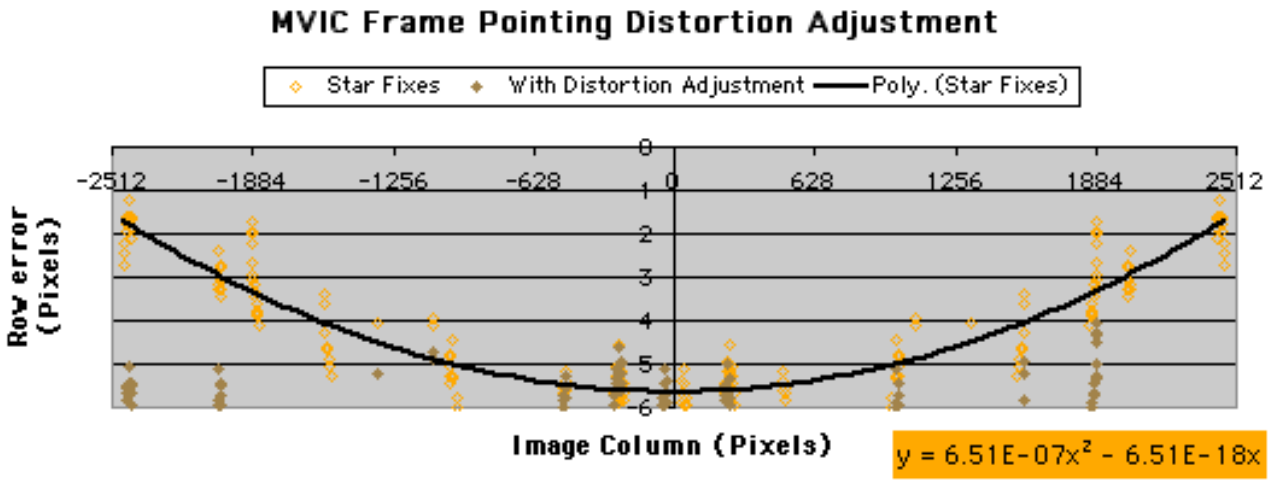
The column error as the star fix moves out from the middle of the image in the column direction follows an approximate 3rd degree polynomial:



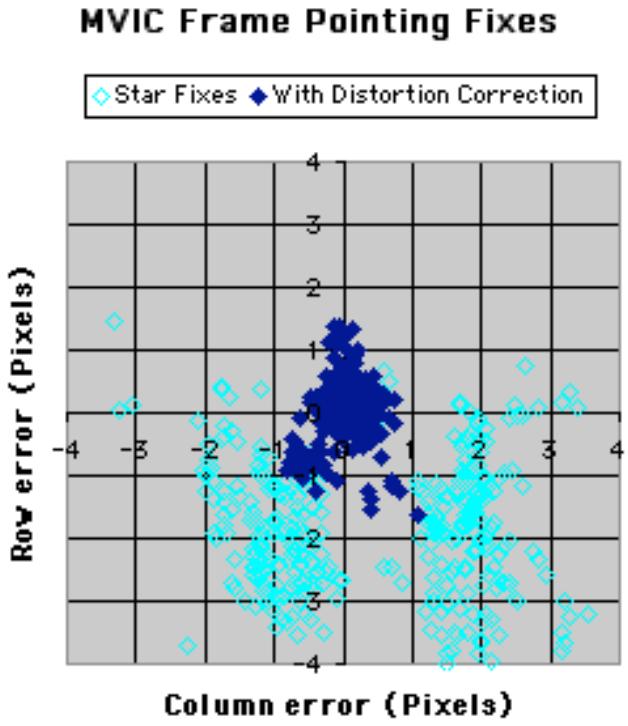
The row error as the star fix moves out from the middle of the image in the column direction follows an approximate 2rd degree polynomial:



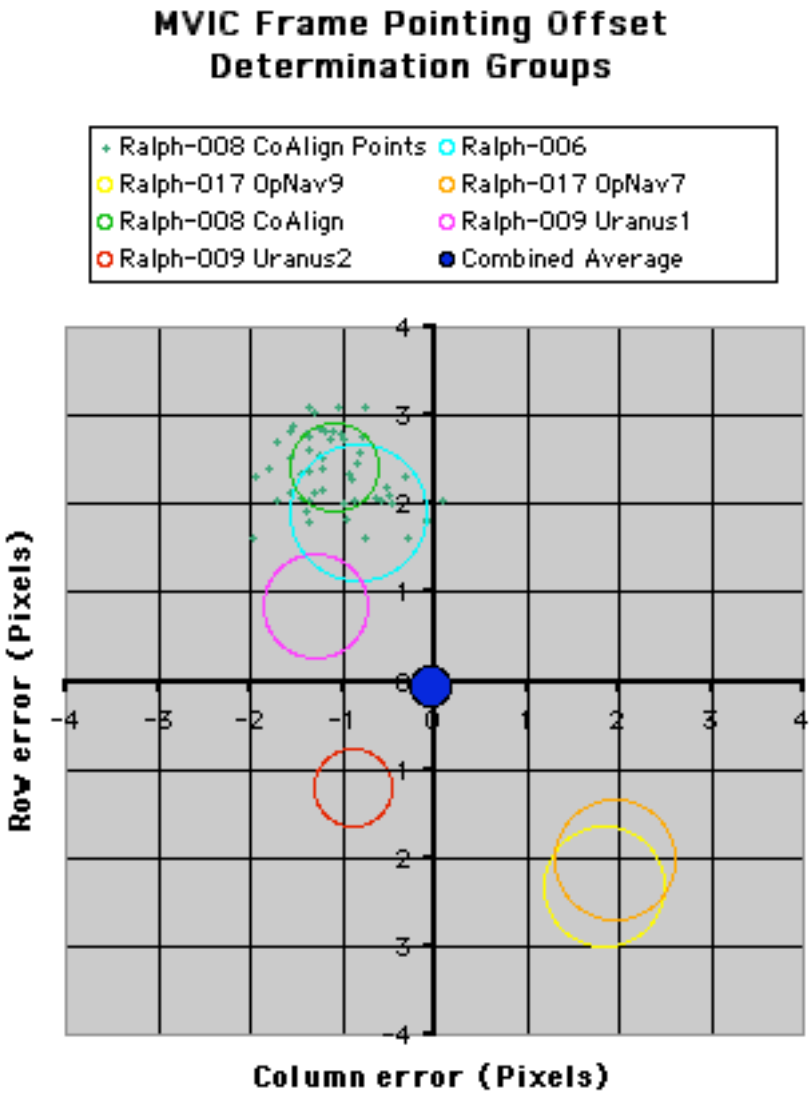
When the distortion correction was then applied to observation data from Ralph-017 observations, the correction for the raw error verses column was too high. The distortion model was adjusted to fit this data:



Without the distortion model, the average error in pointing is 2.51 pixels (stdev 1.03). Using the distortion model the average error in pointing is .55 pixels (stdev .38). This refers only to the distortion model. It does not include the overall error in pointing for the observation, which is an addition offset that is analyzed next.



Between activities and between observations there is a varying offset in the knowledge of the spacecraft pointing. The observations were separated into 6 groups of all the observations from a slew activity.. Applying the distortion model to each observation yields an average offset for each group. Averaging the offsets for each group give an overall offset that is applied to the determination of the instrument boresight definition. The boresight adjustment minimizes the overall pixel error offset for each observation group:



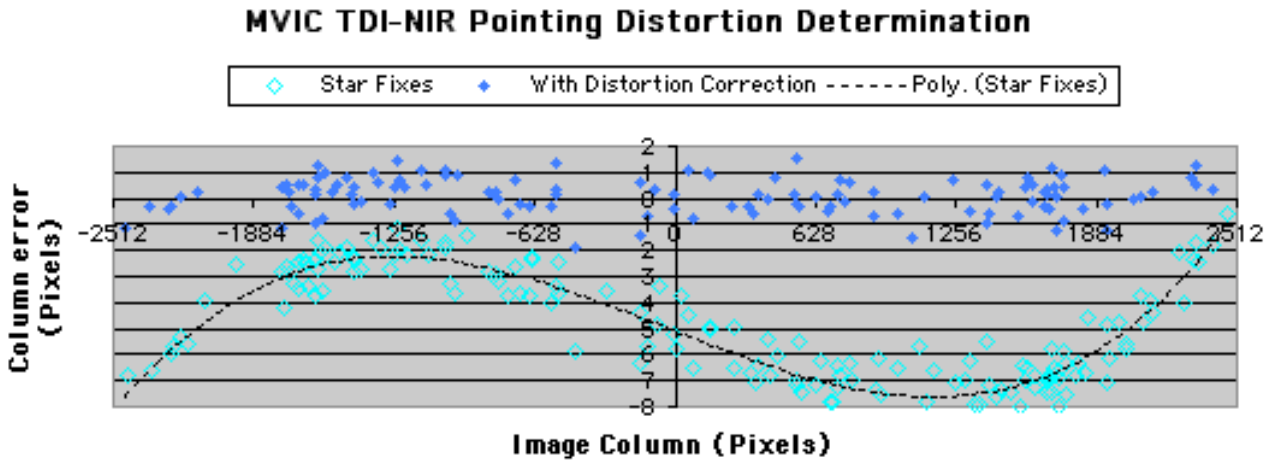
MVIC TDI

The boresight offset for the MVIC-TDI observation mode was approximated prior to this analysis based on the layout of the CCD arrays. Offsets in the star centers will be used to adjust the boresight definition. The initial fit to the TDI image plane was done using the Near IR channel since it is the leading edge of the TDI scan.

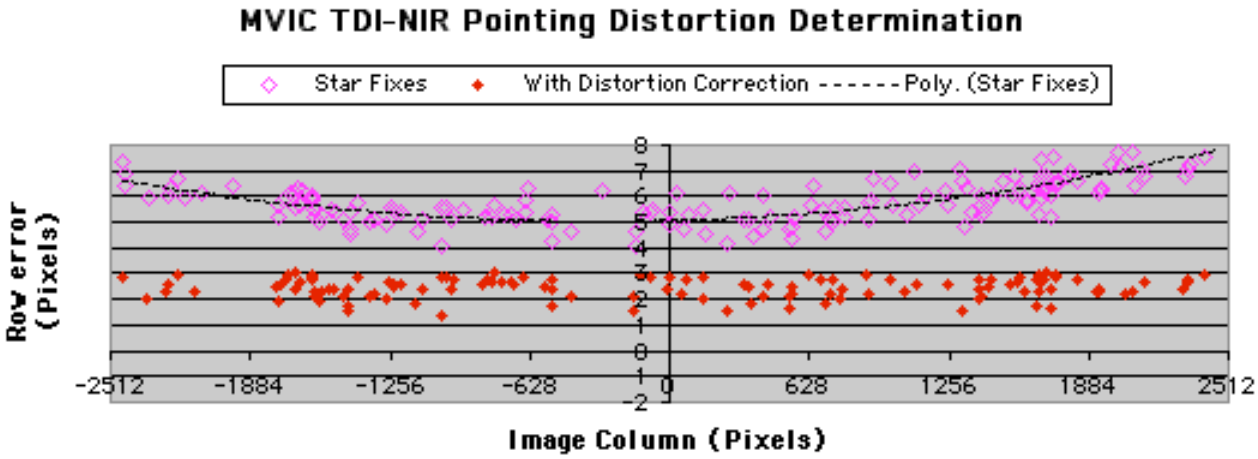
For the initial distortion model analysis, the TDI color scan from Ralph-006 (targeting M6 & M7) with integration time of 628 milliseconds was used. In the TDI mode, the target passes through 32 CCD pixels as it is scanned and the total signal is returned. The star location was calculated at the time when the star crossed center of the middle line of the TTI array (the middle of the integration time). The gauss determined positions of each star scanned were compiled and analyzed. There were a total of 123 stars that returned reliable positions.

The difference between the predicted positions and the gauss determined positions in the row and column direction were fit to polynomial functions verses column. The row and column error as a function of row cannot be derived for the TDI mode.

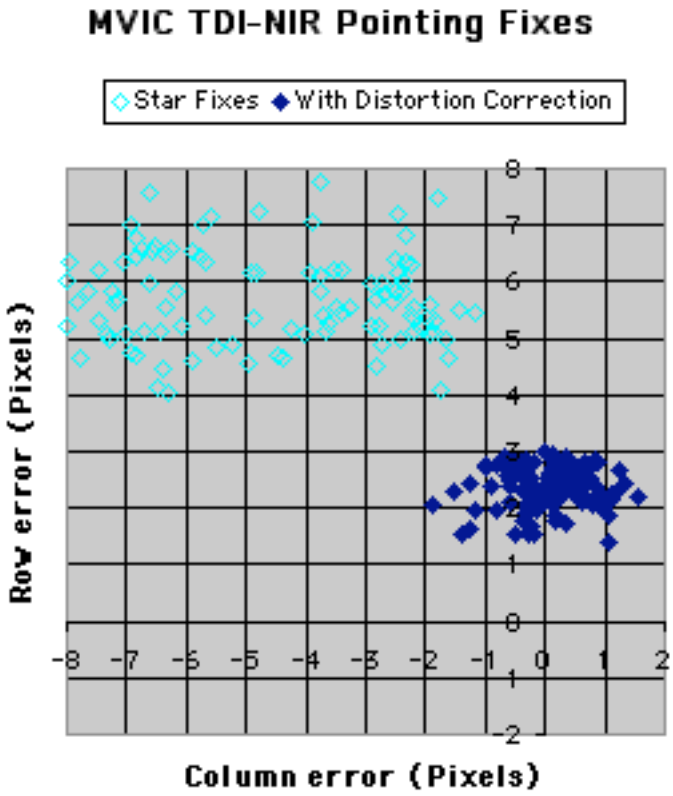
The column error as the star position moves out from the middle of the image in the column direction follows an approximate 3rd degree polynomial. The constant term was used to adjust the boresight definition for the TDI channels:



The row error as the star fix moves out from the middle of the image in the column direction follows an approximate 2rd degree polynomial:

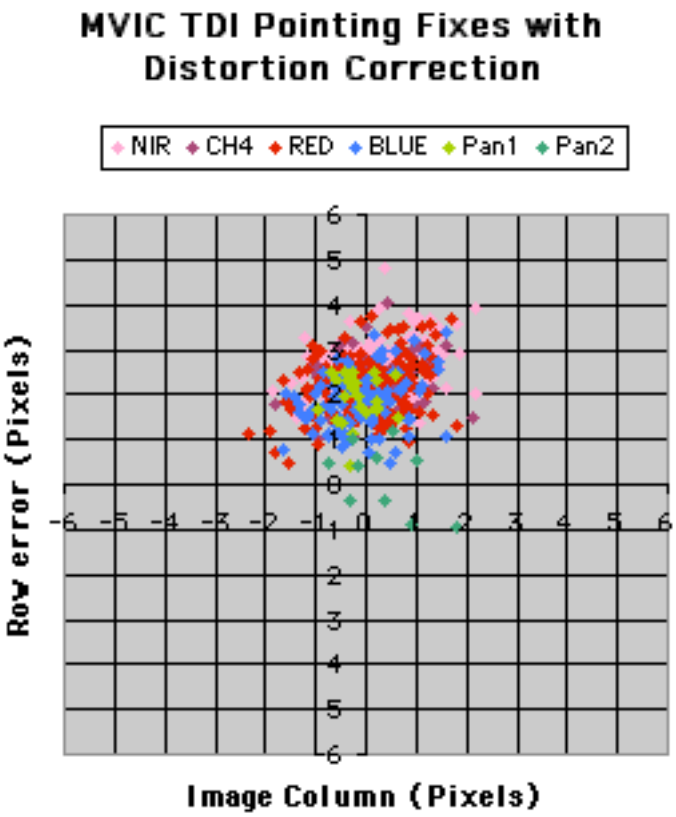


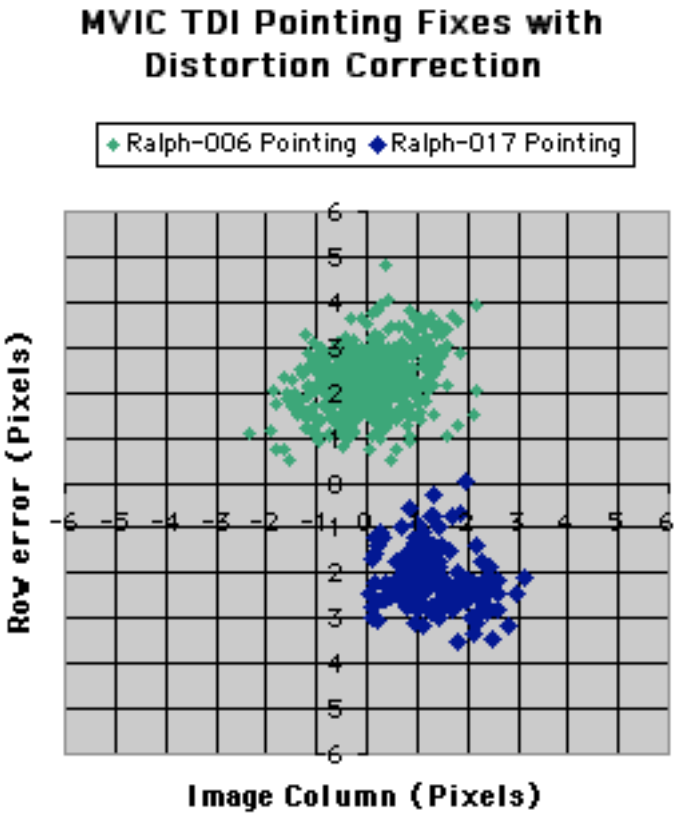
Using the average error to remove the remaining boresight offset. Without the distortion model, the average error in pointing is 5.13 pixels (stdev 2.1). Using the distortion model the average error in pointing is .70 pixels (stdev .39). The average error with the distortion model without compensating for the remaining boresight offset is 2.55 pixels (stdev .37).



The remaining 5 TDI channels had similar polynomial fits for the row and column pointing errors. The same row and column error corrections can be applied to all channels. The overall offset in the pointing error from the TDI channels was used to determine the boresight offset for each.

There is a residual ~2.5 pixel offset in the scan direction that could be removed by changing the boresight offset for the channel. However, data from the Ralph-017 observations yield a different offset. The average offset for the two long TDI scans taken so far were used to define the overall boresight offset.

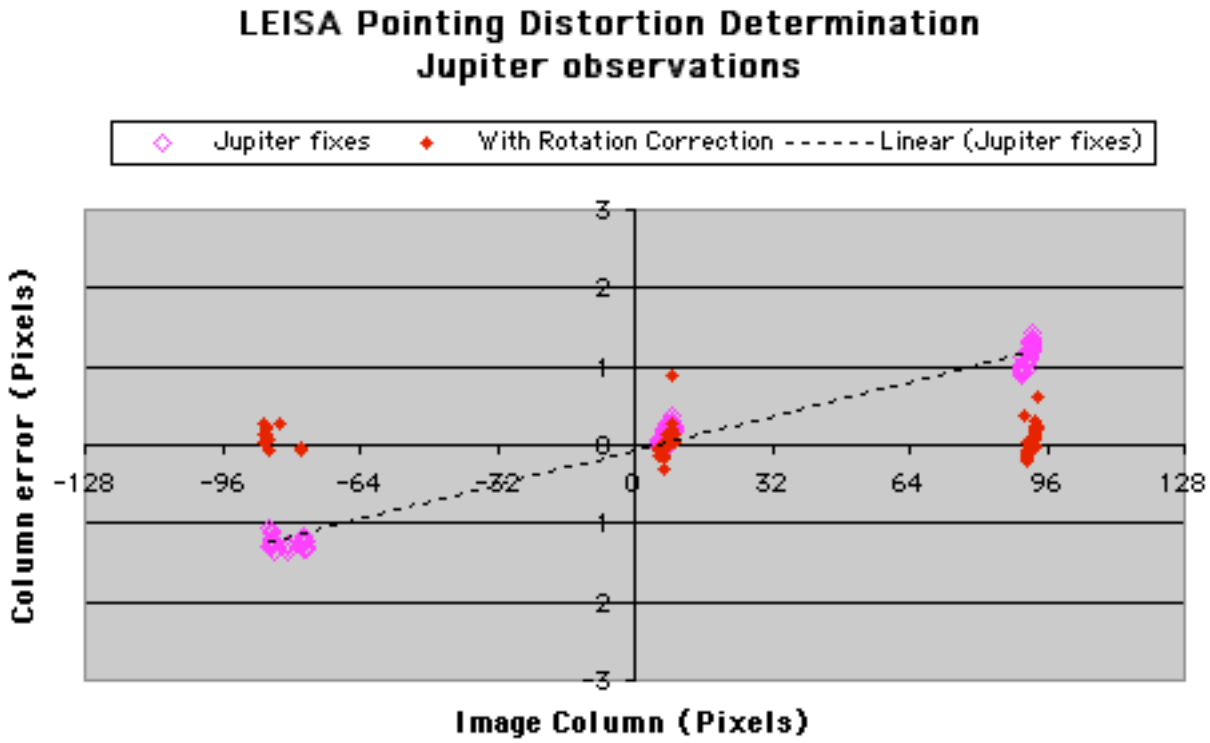




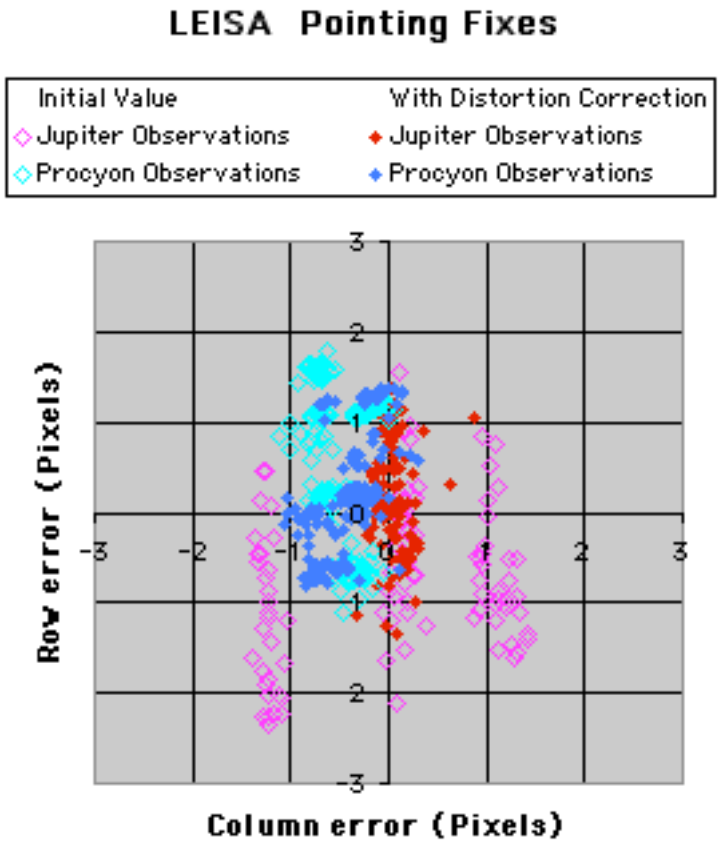
LEISA

The boresight offset for the LEISA observation mode was approximated prior to this analysis. Offsets in the star fixes (Procyon) were used to adjust the boresight definition. The Jupiter scans provided the first data with targets away from the center columns of the array.

The data shows a similar symmetric distortion, but the LEISA FOV is only about 15% as wide as MVIC. Since this does not provide data fit the 3rd degree polynomial, a linear fit was used and then the residual error was removed with a 2nd degree fit.

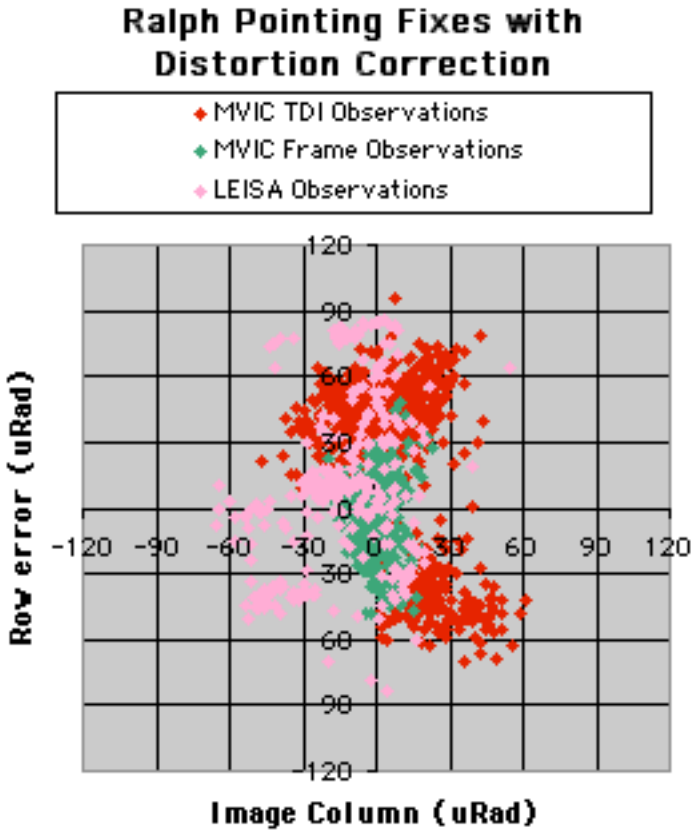


Without the distortion model, the average error in pointing is 1.15 pixels (stdev 0.51). Using the distortion model the average error in pointing is .69 pixels (stdev .40).



MVIC and LEISA combined

Converting the pixel pointing errors to angles allows the data from the different instruments to be displayed together. The pixels size for MVIC is 19.800 microradians. The pixel size for LEISA is 62.0065 microradians.



	LEISA	MVIC-Frame	MVIC-TDI
Mean pointing error (μrad)	42.83	17.05	49.44
Standard deviation	24.05	10.87	12.28

Boresight and Distortion Values

The simplest way to relay the values for the boresight offsets and distortion parameters is in the form of programming source code. Routines written in C to return arrays filled with the values are listed below, along with usage examples. Some of the example routines rely on C-SPICE routines. In these routines the prefix "Mvi" is used to indicate MVIC-Frame, the prefix "Tdi" is MVIC-TDI, and lei is the LEISA instrument.

Boresight Values and Vectors

The quaternion that represents the spacecraft pointing in J2000 terms can be converted to a rotation matrix. The inverse of this matrix rotates vectors from J2000 into the spacecraft frame. The rotation matrix that defines the instrument boresight will rotate this vector into the instrument frame.

An example for an MVIC-Frame takes the star location at RA,DEC and converts that to a vector in the MVIC-Frame:

```
pxform_c("J2000",
          "NH_SPACECRAFT",
           etime, toSpcMtx);          // get the spacecraft J2000 rotation matrix
mtx_SpcToMvi(trMtx);                  // get the instrument boresight rotation matrix
radrec_c(1., RA, DEC, jVect);         // convert the star RA,DEC into a J2000 vector
mxv_c(toSpcMtx, jVect, sVect);        // convert the J2000 vector into the spacecraft frame
mxv_c(trMtx, sVect, mVect);           // convert the vector into the MVIC-Pan frame
```

A similar process can be applied to the MVIC-TDI and LEISA instruments. Note that the TDI matrix routine requires the TDI-Index values to make the rotation of the TDI boresight for the appropriate color channel. The C routine to rotate the matrix is included.

```
void mtx_SpcToMvi(double mtx[3][3])
{
 mtx[0][0] = 0.999915384493112L;
 mtx[1][0] = 0.012962551750366L;
 mtx[2][0] = -0.001251478128812L;
 mtx[0][1] = -0.012957621237463L;
 mtx[1][1] = 0.999908274152575L;
 mtx[2][1] = 0.003858107811314L;
 mtx[0][2] = 0.001364218471732L;
 mtx[1][2] = -0.003840453099600L;
 mtx[2][2] = 0.999990771414394L;
}
void mtx_SpcToTdi(int tdiChannel, double mtx[3][3])
{
 mtx[0][0] = 0.999655483709946L;
 mtx[1][0] = 0.026223117930647L;
 mtx[2][0] = -0.001277523395108L;
 mtx[0][1] = -0.026218262378526L;
 mtx[1][1] = 0.999649201897941L;
```

```
 mtx[2][1] = 0.003662994056523L;
 mtx[0][2] = 0.001435954285879L;
 mtx[1][2] = -0.003626292333940L;
 mtx[2][2] = 0.999991470526226L;
 mRotZ(mtx, -(0.002008*(double)tdiChannel));  // Rotate for Channel
}
void mtx_SpcToLei(double mtx[3][3])
{
 mtx[0][0] = 0.999833614245650L;
 mtx[1][0] = 0.018161563736818L;
 mtx[2][0] = -0.001805825344462L;
 mtx[0][1] = -0.018154671968107L;
 mtx[1][1] = 0.999817822153710L;
 mtx[2][1] = 0.003751814849954L;
 mtx[0][2] = 0.001936450057155L;
 mtx[1][2] = -0.003715956943165L;
 mtx[2][2] = 0.999990289479017L;
}
void mRotZ(double tMtx[3][3], double angle)
{
double mtx[3][3];
double m[3][3];
  mtx[0][0] = cos(angle);  mtx[1][0] = sin(angle); mtx[2][0] = 0.0;
  mtx[0][1] = -sin(angle);  mtx[1][1] = cos(angle);  mtx[2][1] = 0.0;
  mtx[0][2] = 0.0;  mtx[1][2] = 0.0;  mtx[2][2] = 1.0;
  mxm_c( mtx, tMtx, m);
  tMtx[0][0] = m[0][0];  tMtx[1][0] = m[1][0];  tMtx[2][0] = m[2][0];
  tMtx[0][1] = m[0][1];  tMtx[1][1] = m[1][1];  tMtx[2][1] = m[2][1];
  tMtx[0][2] = m[0][2];  tMtx[1][2] = m[1][2];  tMtx[2][2] = m[2][2];
  return;
}
```

```
void coef_MviDistortion(double colCoeff[4], double rowCoeff[4])
{
  colCoeff[0] = 0.0;
  colCoeff[1] = -2.65160717584E-03;
  colCoeff[2] = -2.00030853786E-07;
  colCoeff[3] = 6.48442797916E-10;

  rowCoeff[0] = 0.0;
  rowCoeff[1] = -5.67345760626E-20;
  rowCoeff[2] = -6.46840932054E-7+6.50509199208E-07;
  rowCoeff[3] = 0.0;
}
void coef_TdiDistortion(double colCoeff[4], double rowCoeff[4])
{
  colCoeff[0] = 0.0;
  colCoeff[1] = -3.211651167000000e-03;
  colCoeff[2] = -1.794373500000000e-07;
  colCoeff[3] = 7.157154600000000e-10;

  rowCoeff[0] = 0.0;
  rowCoeff[1] = -1.264654000000000e-04;
  rowCoeff[2] = 2.994285500000000e-07;
  rowCoeff[3] = 0.0;
}
void coef_LeiDistortion(double colCoeff[4], double rowCoeff[4])
{
  colCoeff[0] = 0.0;
  colCoeff[1] = 1.4350043918383E-02;
  colCoeff[2] = -2.40026415852E-05;
  colCoeff[3] = 0.0;

  rowCoeff[0] = 0.0;
  rowCoeff[1] = 0.001155744;
  rowCoeff[2] = 0.00012576;
  rowCoeff[3] = 0.0;
}
```

Instrument Pixel Size

A key parameter in the calculation of pixels from vectors is the pixels size. Pre-flight test determined the nominal pixels size of all MVIC pixels to be 19.8 microradians and all LEISA pixels to be 61.0 microradians. It is expected that the optical distortion would make the pixels appear non-square. LEISA is the only instrument with significant angular FOV in two dimensions and this effect was seen. It was compensated for automatically by the distortion model, but the nominal pixels size was adjusted to remove a linear offset in one dimension. As a result the pixel size used in the application of these calculations has been adjusted:

	LEISA	MVIC-Frame	MVIC-TDI
Apparent pixel size (μrad)	62.065	19.8	19.8

Instrument Vectors and Rows and Columns

The instrument frame follows the spacecraft convention with -X being the boresight. Referening to the SOC data files used in this analysis, Y increases with increasing rows, and Z increases with decreasing columns. Internal to the Row,Column processing code the pointing vector is sometimes converted to align with the SPICE defined reference vector which converts the [Boresight,Column,Row] from [-X,+Z,+Y] to [+X, +Y, +Z].

The example for an MVIC-Frame takes the star location at RA,DEC and converts that to a vector in the MVIC-Frame, then to instrument column and row (this refers to number of pixels from the center of the array), then to image column and row:

```
pxform_c("J2000",
          "NH_SPACECRAFT",
           etime, toSpcMtx);              // get the spacecraft J2000 rotation matrix
mtx_SpcToMvi(trMtx);                      // get the instrument boresight rotation matrix
radrec_c(1., RA, DEC, jVect);             // convert the star RA,DEC into a J2000 vector
mxv_c(toSpcMtx, jVect, sVect);            // convert the J2000 vector into the spacecraft frame
mxv_c(trMtx, sVect, mVect);               // convert the vector into the MVIC-Pan frame
pxl_MviVect(1, mVect, &mCol, &mRow);   //  Compute the MVIC row and column
imageCol = 2511.5 + mCol;          // Compute the image pixels column
imageRow = 63.5 + mRow;           // Compute the image pixel row

// This routine is partially represented for illustration
 void pxl_MviVect(
 int distortionFlag,        // Input: compute distrtion if set
 double mVect[3],          // Input:  equivalent vector in instrument frame
 double *nCol,             // Output:  pixel offset up from the boresight  Column
 double *nRow )            // Output:  pixel offset to the right from the boresight  Row
{ . . .
  colAngle = atan2(mVect[12], -mVect[0]);
  rowAngle = atan2(mVect[1], -mVect[0]);
  *nCol = colAngle / pxlSize;
  *nRow = rowAngle / pxlSize;
```

```
  if (distortionFlag)  distort_MviPxl(0, nCol, nRow);
  return;
}
```

Distortion models

The distortion model is applied to the instrument row and column by adding the result of a 2nd or 3rd degree polynomial computed on instrument column. The polynomial coefficients are returned in the routines below as a four element array.

The column distortion is a function of column, so to reverse the process you must solve the cubic equation. The row distortion is a function of column so you can evaluate the polynomial and subtract.

```
void coef_TdiDistortion(double colCoeff[4], double rowCoeff[4])
{
  colCoeff[0] = 0.0;                                  //  Offset
  colCoeff[1] = -3.211651167000000e-03;       //   * column
  colCoeff[2] = -1.794373500000000e-07;       //   * column^2
  colCoeff[3] = 7.157154600000000e-10;         //  * column^3

  rowCoeff[0] = 0.0;
  rowCoeff[1] = -1.264654000000000e-04;
  rowCoeff[2] = 2.994285500000000e-07;
  rowCoeff[3] = 0.0;
}
void coef_MviDistortion(double colCoeff[4], double rowCoeff[4])
{
  colCoeff[0] = 0.0;
  colCoeff[1] = -2.65160717584E-03;
  colCoeff[2] = -2.00030853786E-07;
  colCoeff[3] = 6.48442797916E-10;

  rowCoeff[0] = 0.0;
  rowCoeff[1] = -5.67345760626E-20;
  rowCoeff[2] = -6.46840932054E-7+6.50509199208E-07;
  rowCoeff[3] = 0.0;
}
void coef_LeiDistortion(double colCoeff[4], double rowCoeff[4])
{
  colCoeff[0] = 0.0;
  colCoeff[1] = 1.4350043918383E-02;
  colCoeff[2] = -2.40026415852E-05;
  colCoeff[3] = 0.0;

  rowCoeff[0] = 0.0;
  rowCoeff[1] = 0.001155744;
  rowCoeff[2] = 0.00012576;
  rowCoeff[3] = 0.0;
}
```

SPICE Kernels

Using the boresight offset values, a SPICE kernels was developed with all instrument frames and is listed below. For some reason, which I cannot explain, these kernels returned results that were 62 microradians off around the Y axis for LEISA and MVIC-Frame, and 73 microradians off in the Y direction for MVIC-TDI. The matrix values were adjusted to make SPICE return the correct values (there is some residual error). Hopefully the reason can be resolved and consistent values used in the future.

The SPICE kernels simplify the calculation by allowing you to transform a vector from J2000 directly to the instrument frame coordinate system.

```
pxform_c("J2000",
          mviSpiceName[0],              // use the SPICE frame name
          etime, toInsMtx);             // get the instrument J2000 rotation matrix
radrec_c(1., RA, DEC, jVect);           // convert the star RA,DEC into a J2000 vector
mxv_c(toInsMtx, jVect, mVect);          // convert the J2000 vector into the instrument frame
pxl_MviVect(1, mVect, &mCol, &mRow);    //  Compute the MVIC row and column
imageCol = 2511.5 + mCol;          // Compute the image pixels column
imageRow = 63.5 + mRow;           // Compute the image pixel row
```

The text below is a SPICE kernel. It can be copied into a text file and added to the SPICE pool with FURNSH function or into a metakernel. It must be loaded after the "nh_allinstruments_vXXX.ti" kernel in order to override the frame definitions. Note that the instrument reference frames are all defined with respect to the NH_SPACECRAFT frame.

```
\begintext

Basic boresight for the instrument

\begindata

FRAME_NH_RALPH_MVIC      = -98200
FRAME_-98200_NAME        = 'NH_RALPH_MVIC'
FRAME_-98200_CLASS       = 4
FRAME_-98200_CLASS_ID    = -98200
FRAME_-98200_CENTER      = -98
TKFRAME_-98200_SPEC      = 'MATRIX'
TKFRAME_-98200_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98200_MATRIX    = ( 1.0, 0.0, 0.0,
                             0.0, 1.0, 0.0,
                             0.0, 0.0, 1.0 )


FRAME_NH_MVI           = -98298
FRAME_-98298_NAME        = 'NH_MVI'
FRAME_-98298_CLASS       = 4
FRAME_-98298_CLASS_ID    = -98298
```

```
FRAME_-98298_CENTER      = -98
TKFRAME_-98298_SPEC      = 'MATRIX'
TKFRAME_-98298_RELATIVE  = 'NH_RALPH_MVIC'
TKFRAME_-98298_MATRIX    = ( 1.0, 0.0, 0.0,
                             0.0, 1.0, 0.0,
                             0.0, 0.0, 1.0 )


\begintext
```

MVIC Pan Frame Transfer Array

```
\begindata

FRAME_NH_RALPH_MVIC_FT   = -98203
FRAME_-98203_NAME        = 'NH_RALPH_MVIC_FT'
FRAME_-98203_CLASS       = 4
FRAME_-98203_CLASS_ID    = -98203
FRAME_-98203_CENTER      = -98
TKFRAME_-98203_SPEC      = 'MATRIX'
TKFRAME_-98203_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98203_MATRIX    = (
        0.999915461106432,
        -0.012957863475922,
        0.001301425571368,
        0.012962551750366,
        0.999908274152575,
        -0.003840453099600,
        -0.001188689962478,
        0.003857294149614,
        0.999990855106924
                )

\begintext
```

TDI Arrays

```
\begindata

FRAME_NH_RALPH_MVIC_NIR     = -98209
FRAME_-98209_NAME        = 'NH_RALPH_MVIC_NIR'
FRAME_-98209_CLASS       = 4
FRAME_-98209_CLASS_ID    = -98209
FRAME_-98209_CENTER      = -98
TKFRAME_-98209_SPEC      = 'MATRIX'
TKFRAME_-98209_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98209_MATRIX    =  (
        0.999655573953295,
        -0.026218528636659,
        0.001363247322305,
```

```
        0.026223117930647,
        0.999649201897941,
        -0.003626292333940,
        -0.001204840860741,
        0.003661087780433,
        0.999991572287813
                )
FRAME_NH_RALPH_MVIC_METHANE    = -98208
FRAME_-98208_NAME       = 'NH_RALPH_MVIC_METHANE'
FRAME_-98208_CLASS      = 4
FRAME_-98208_CLASS_ID   = -98208
FRAME_-98208_CENTER     = -98
TKFRAME_-98208_SPEC     = 'MATRIX'
TKFRAME_-98208_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98208_MATRIX   =  (
        0.999706214601813,
        -0.024211181536640,
        0.001355962837282,
        0.024215758201663,
        0.999699832809316,
        -0.003629168417489,
        -0.001204837178782,
        0.003661233729783,
        0.999991571758176
                )
FRAME_NH_RALPH_MVIC_BLUE    = -98207
FRAME_-98207_NAME       = 'NH_RALPH_MVIC_BLUE'
FRAME_-98207_CLASS      = 4
FRAME_-98207_CLASS_ID   = -98207
FRAME_-98207_CENTER     = -98
TKFRAME_-98207_SPEC     = 'MATRIX'
TKFRAME_-98207_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98207_MATRIX   =  (
        0.999795403077788,
        -0.020196202570963,
        0.001341376615173,
        0.020200753919525,
        0.999789001887114,
        -0.003634876673948,
        -0.001204830694104,
        0.003661525649188,
        0.999991570697647

                )
FRAME_NH_RALPH_MVIC_RED    = -98206
FRAME_-98206_NAME       = 'NH_RALPH_MVIC_RED'
FRAME_-98206_CLASS      = 4
FRAME_-98206_CLASS_ID   = -98206
FRAME_-98206_CENTER     = -98
TKFRAME_-98206_SPEC     = 'MATRIX'
```

```
TKFRAME_-98206_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98206_MATRIX    =  (
        0.999752824372622,
        -0.022203736816693,
        0.001348672591773,
        0.022208300833224,
        0.999746432868338,
        -0.003632029868004,
        -0.001204833789898,
        0.003661379686231,
        0.999991571228120
                )
FRAME_NH_RALPH_MVIC_PAN1    = -98205
FRAME_-98205_NAME       = 'NH_RALPH_MVIC_PAN1'
FRAME_-98205_CLASS      = 4
FRAME_-98205_CLASS_ID   = -98205
FRAME_-98205_CENTER     = -98
TKFRAME_-98205_SPEC     = 'MATRIX'
TKFRAME_-98205_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98205_MATRIX    =  (
        0.999833950545631,
        -0.018188586893952,
        0.001334074936899,
        0.018193125555121,
        0.999827539694002,
        -0.003637708823840,
        -0.001204827891411,
        0.003661671618066,
        0.999991570166760
                )
FRAME_NH_RALPH_MVIC_PAN2    = -98204
FRAME_-98204_NAME       = 'NH_RALPH_MVIC_PAN2'
FRAME_-98204_CLASS      = 4
FRAME_-98204_CLASS_ID   = -98204
FRAME_-98204_CENTER     = -98
TKFRAME_-98204_SPEC     = 'MATRIX'
TKFRAME_-98204_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98204_MATRIX    =  (
        0.999868466620725,
        -0.016180897880494,
        0.001326767586392,
        0.016185423834895,
        0.999862046133615,
        -0.003640526306263,
        -0.001204825381830,
        0.003661817592276,
        0.999991569635460
                )

\begintext
```

LEISA Array

```
\begindata

FRAME_NH_RALPH_LEISA     = -98201
FRAME_-98201_NAME        = 'NH_RALPH_LEISA'
FRAME_-98201_CLASS       = 4
FRAME_-98201_CLASS_ID    = -98201
FRAME_-98201_CENTER      = -98
TKFRAME_-98201_SPEC      = 'MATRIX'
TKFRAME_-98201_RELATIVE  = 'NH_SPACECRAFT'
TKFRAME_-98201_MATRIX    =  (
        0.999833725668520,
        -0.018154907521815,
        0.001873657185926,
        0.018161563736818,
        0.999817822153710,
        -0.003715956943165,
        -0.001743042311673,
        0.003750674847578,
        0.999990409103958
                )

FRAME_NH_LEI             = -98299
FRAME_-98299_NAME        = 'NH_LEI'
FRAME_-98299_CLASS       = 4
FRAME_-98299_CLASS_ID    = -98299
FRAME_-98299_CENTER      = -98
TKFRAME_-98299_SPEC      = 'MATRIX'
TKFRAME_-98299_RELATIVE  = 'NH_RALPH_LEISA'
TKFRAME_-98299_MATRIX    = ( 1.0,
                     0.0,
                     0.0,
                     0.0,
                     1.0,
                     0.0,
                     0.0,
                     0.0,
                     1.0 )
```