The NGWFC Control Algorithms
KAON 517

November 28, 2007
Erik Johansson, Marcos van Dam

Abstract

This note provides a brief summary of the real-time algorithms of the NGWFC.

1. System Overview

A context diagram illustrating the major functional blocks of the NGWFC is shown below in Figure 1. To minimize the impact on existing systems, the NGWFC design reused as many of the existing WFC subsystems and external systems as possible. These components are shown in yellow, and their interfaces are shown with dashed lines. The remaining components in the figure either replaced existing functionality or, in the case of the Telemetry Recorder/Server (TRS), provided new capabilities. The Real-Time Controller portion of the NGWFC is shown with a light blue background. The RTC was provided by our collaborator, Microgate. The system components provided by Microgate are shown in blue. The remaining component, the wavefront sensor camera, was provided by WMKO, and is shown in violet.

The NGWFC is comprised of a wavefront sensor (WFS) camera, a wavefront processor (WFP), two tip-tilt controllers (DTT, UTT), a telemetry recorder/server (TRS), a wavefront controller interface (WIF), and a wavefront controller command processor (WCP). These components are described briefly below:

- WFS camera: The wavefront sensor camera is the detector portion of a Shack-Hartmann wavefront sensor. It detects the incoming wavefront of light and sends pixel data to the wavefront processor for real-time processing. A lenslet array, reducer optics and associated mechanical hardware and motion control software are part of the WFS as well.
- WFP: The wavefront processor receives the pixel data from the WFS camera, computes the locations or centroids of the Shack-Hartmann subaperture spots, computes a matrix-vector-multiply of the centroid vector with the reconstruction matrix, applies a servo control law to the result, and outputs the deformable mirror actuator command. It sends tip-tilt error data to the downlink tip-tilt controller (DTT) and uplink tip-tilt controllers (UTT) and also provides streams of telemetry data to the telemetry server. The WFP is the heart of the wavefront controller.
- DTT: The downlink-tip-tilt controller receives centroid error inputs from the WFP (originating either from the WFS or STRAP (the System for Tip-tilt Removal with Avalanche Photo-diodes)) and moves the DTT mirror in real-time to minimize these errors. Its purpose is to stabilize the guide star on either the WFS camera or the STRAP detector, depending on whether the system is being used in NGS or LGS mode. The DTT controller consists of two separate controllers: a tip-tilt mirror controller that corrects for the tilt disturbance caused by the atmosphere and a closed-loop mirror positioning system that moves the mirror to the position commanded by the atmospheric controller.
- UTT: The uplink-tip-tilt controller receives centroid error inputs from the WFP (originating from the WFS) and moves the UTT mirror in real-time to minimize these errors. Its purpose is to stabilize the laser guide star on the WFS camera. The UTT controller consists of two separate controllers: a tip-tilt mirror controller that corrects for any tilt disturbance measured in the laser spot and a closed-loop mirror positioning system that moves the mirror to the position commanded by the atmospheric controller.
- TRS. The telemetry recorder/server receives streams of data from the wavefront processor and makes them available to outside systems via keywords, the observatory-wide data interchange mechanism. It also provides a means to archive real-time data as diagnostics files and supports data retrieval queries. Finally, it supports the recording of telemetry from the NGWFC tip-tilt subsystems (DTT, UTT, and STRAP).
- WIF: The wavefront controller interface is a software interface that allows the wavefront command processor to exchange control, parameter, and status information with the WFS camera, WFP, and TRS.

- WCP: The wavefront controller command processor is the main external command and parameter interface for the NGWFC. The WCP is the interface between the NGWFC and the distributed control system (EPICS) used throughout the observatory. With the exception of telemetry queries made directly to the TRS, all external systems communicate with the NGWFC through the WCP.

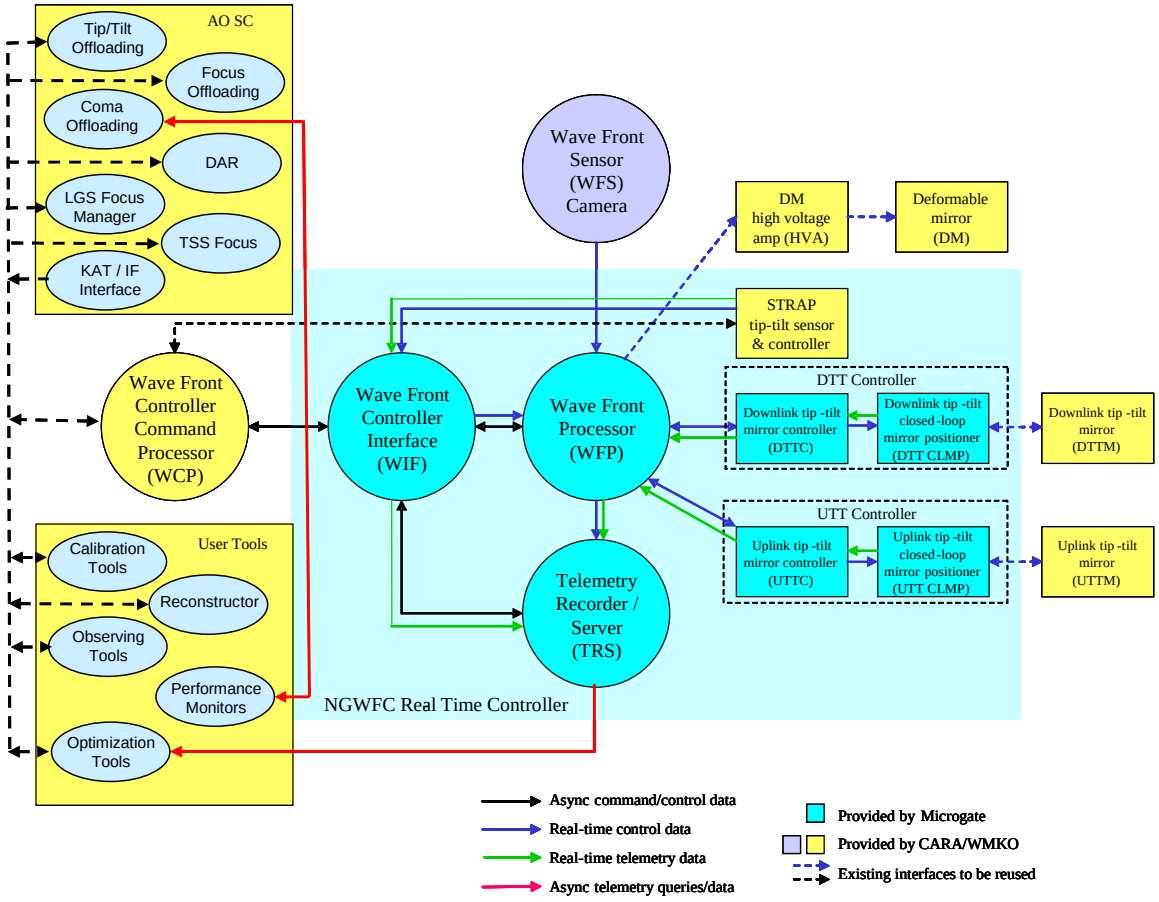

Figure 1: The main components of the Keck AO Next-Generation Wavefront Controller.

2. WFP Data flow overview

As mentioned above, the WFP is the heart of the RTC. A dataflow diagram for the WFP is shown in Figure 2 below.

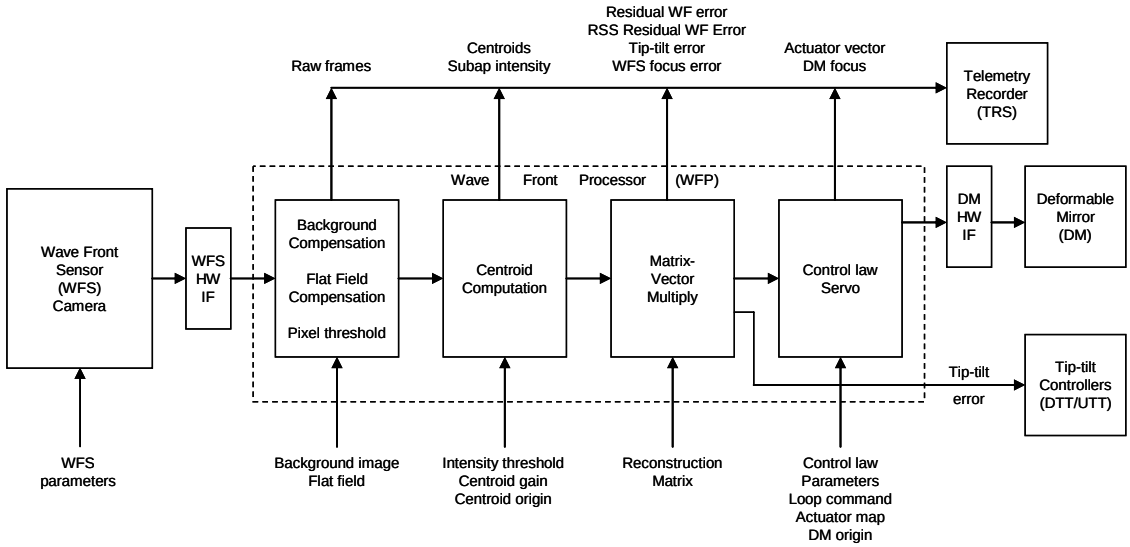

Figure 2: A data-flow diagram of the wavefront processor.

The data processing sequence is as follows:

- The WFS camera outputs a frame of pixel data. Each frame consists of an array of spots resulting from light incident on an array of lenslets focused on the CCD.
- The WFS camera pixel data is routed to the WFP by a hardware interface. Any pixel reformatting or rerouting required by the WFP is done in this interface. Raw pixel data are output as a telemetry stream.
- The pixels are compensated for gain (flat-field) and offset (bias) variations, then thresholded to zero.
- The centroids are computed over the pixels in the subaperture map as follows:
    - The raw centroid is computed using a center-of-mass algorithm. The incident subaperture flux is computed and output as telemetry during this step.
    - The centroids are converted from pixel units to angle-of-arrival in arc-seconds.
    - The centroid origin is subtracted from the result to give the offset centroids.
    - The offset centroid vector is output as a telemetry stream.
- The offset centroid vector is multiplied by a reconstruction matrix, forming the residual wavefront error. The residual wavefront error is output as a telemetry stream.
- A modal decomposition of the residual wavefront error into tip, tilt, and focus, is computed as a by-product of the matrix-vector multiply and is also output as a telemetry stream.
- The tip and tilt residual wavefront errors are written to the uplink- and downlink-tip-tilt controllers.
- The residual wavefront error vector is passed through the control law servo to generate the raw DM actuator vector. The raw DM actuator vector is compensated by adding the DM origin (flat) vector and applying actuator limits, yielding the final DM actuator vector. The RMS residual wavefront error is also computed during this step. The actuator vector and RMS residual wavefront error are output as telemetry streams.
- Finally, the actuator vector is remapped from actuator space to the appropriate channels of the DM high voltage amplifier (HVA) and sent to the HVA controller via a hardware interface to move the

deformable mirror. The hardware interface converts the format of the actuator vector to the appropriate protocol required by the HVA.

3. Algorithm description

This section describes the computational algorithm implemented in the WFP.

3.1 Raw Pixel Processing

Each pixel to be used in a subaperture centroid calculation is compensated for gain and offset using flat-field and background images. The equation describing this process is

$$i_c[k] = i_f[k](i[k] - i_b[k]) \tag{1}$$

where $i[k]$ is the incident flux at pixel k, $i_b[k]$ is the background level at pixel k, $i_f[k]$ is the normalized flat-field gain response for pixel k, and $i_c[k]$ is the resulting compensated pixel. The incident flux (raw pixel) data is injected into the telemetry stream at this point.

3.2 Centroid Processing

After the compensation described above, centroids are computed for all subapertures according to the following algorithm:

- Each pixel is set to a minimum threshold value of zero.
- The total incident flux (the sum of all the pixels) in the subaperture is computed.
- The sum is set to a user-defined minimum threshold value (the minimum subaperture flux). This result is S_T, the thresholded subaperture flux.
- The raw centroids are computed using these thresholded values according to

$$C_{Rx} = \frac{\sum_x x\left(\sum_y i_T(x,y)\right)}{S_T} \tag{2}$$

and

$$C_{Ry} = \frac{\sum_y y\left(\sum_x i_T(x,y)\right)}{S_T} \tag{3}$$

where S_T is thresholded subaperture flux, and $i_T(x,y)$ is the compensated and thresholded flux at pixel position (x,y) in the subaperture. C_{Rx} and C_{Ry} are the raw x- and y-centroids. The x and y values used for the weights in the multiplication are {-1, 1} for 2x2 pixel subapertures and {-3, -1, 1, 3} for 4x4 pixel subapertures. The x and y centroid values are then interleaved to form a vector of raw centroids over all the subapertures (304 subapertures, 608 centroid values).

- The raw centroids are multiplied by a conversion factor to scale from pixel units to arc-sec. This conversion factor may vary from subaperture to subaperture, so the parameter is a vector:

$$C_C = C_S C_R \tag{4}$$

where C_S is the vector of scale coefficients, C_R is the vector of raw centroids computed above, and C_C is the vector of converted centroid values in arc-sec.

- The centroid origin vector, C_O, is subtracted to give the offset centroid vector C:

$$C = C_C - C_O \tag{5}$$

- The offset centroid values are injected into the telemetry stream.

3.3 Denominator-Free Centroiding

In the future, we are interested in experimenting with denominator-free centroiding, where each subaperture has a user-defined average total flux value, which is constant over time (a constant value for S_T computed above). This has the potential to improve system performance on faint stars, when the signal-to-noise ratio on the WFS camera CCD is small. Each subaperture has its own average flux value, so the user parameter defining these values is a vector over all the subapertures. Although it is not routinely used, denominator-free centroiding has been implemented in the NGWFC as a vector of $1/d_i$ values, where d_i is the desired denominator value for subaperture i. Having the user specify the reciprocal allows for a very efficient multiplicative implementation in the centroid computation. The use of denominator-free centroiding is software selectable, so that we may proceed with normal centroiding for now and experiment with the new approach as desired.

3.4 Residual Wavefront Computation

Next the residual wavefront is computed by multiplying the vector of centroids computed above by a reconstruction matrix. The reconstruction matrix has been pre-computed in the IDL user tools and loaded into the RTC at the appropriate time for the current operating conditions (the derivation and calculation of the reconstruction matrix is outside the scope of this document, see KAON 356, Wavefront reconstruction for the NGWFC, for details). The computation to be performed is given by

$$W_R = RC \tag{6}$$

where W_R is the residual wavefront vector, R is the reconstruction matrix, and C is the centroid vector. The reconstruction matrix is quite large, and the majority of the real-time computation is spent computing this matrix-vector product.

There are 304 subapertures in the WFS, and hence 608 associated x and y centroid values. The deformable mirror has 349 actuators. Therefore, the size of the reconstruction matrix is 608 columns by 349 rows. To facilitate computation of tip, tilt, and focus modal errors, 3 additional rows are appended onto the reconstruction matrix, making it 608 x 352. Upon completion of the matrix-vector multiply, the 3 modal values are stripped off the results vector and injected into the telemetry stream as the modal residual wavefront error. The residual wavefront vector is also injected into the telemetry stream at this point. The tip and tilt residual wavefront errors are written to the uplink- and downlink-tip-tilt controllers. The residual sum of squares (RSS) of the wavefront error vector, $W_R^T W_R$, is also computed and inserted into the telemetry stream.

3.5 Control Law Servo Computation

The residual wavefront vector is next passed through a 3rd order control-law servo to compute the appropriate vector of actuator values for the deformable mirror to compensate the distortions in the incoming wavefront. The algorithm is of the form:

$$Y(z) = H(z)U(z) \tag{7}$$

in the Z-transform domain, where $Y(z)$ is the output and $U(z)$ is the input. $H(z)$ is the transfer function, and is defined as:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}{1 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}} \tag{8}$$

In the discrete domain, the output, $y[n]$, at time index n, is related to the input, $u[n]$ by:

$$\begin{aligned} y[n] = &-b_1 y[n-1] - b_2 y[n-2] - b_3 y[n-3] + \\ &a_0 u[n] + a_1 u[n-1] + a_2 u[n-2] + a_3 u[n-3] \end{aligned} \tag{9}$$

Note that the u and y variables above are actually vectors. The input, $u[n]$, is the residual wavefront vector computed in the previous step, and the output, $y[n]$, is the output vector for the deformable mirror. The a and b values are the control law coefficients, and z is the Z-Transform domain variable, where z^{-1} represents

a unit time delay. The indices *n, n*-1, *n*-2, and *n*-3 refer the current value and the three previous values of each vector.

Next, the result is checked for values that exceed the dynamic range of the mirror actuators, and clipped appropriately. If the control loop is closed, this is the new position that will be sent to the mirror. If the control loop is open, only the DM origin vector (the nominal flat position) will be sent to the mirror. The open-loop or closed-loop mirror output is normalized and converted to the appropriate 16-bit DAC values and sent to the DM High Voltage Amplifier controller. The DM Zernike focus is computed using the new mirror position. Finally, the actuator vector, and DM Zernike focus are injected into the telemetry stream.

3.6 Tip-Tilt Mirror Controller

A dataflow diagram of the tip-tilt mirror controller is shown below in Figure 3. This basic controller unit is used to implement both the Down-Tip-Tilt (DTT) and Up-Tip-Tilt (UTT) controllers.

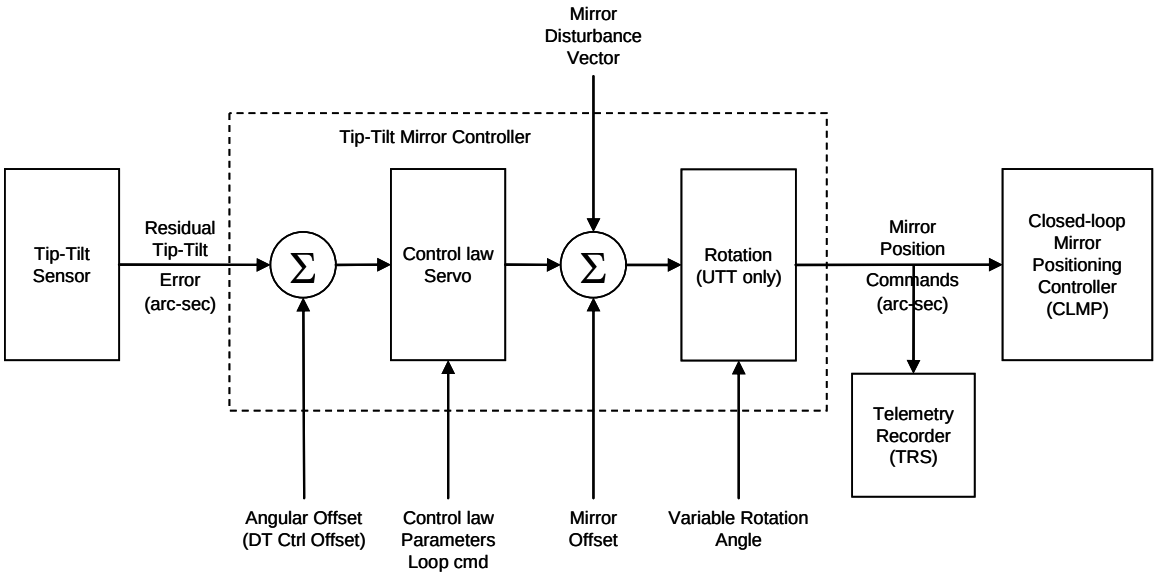

Figure 3: A dataflow diagram of the tip-tilt mirror controller.

The control algorithm is as follows:

- The residual tip-tilt error is received from the appropriate tip-tilt sensor via the WFP as an input *x*-*y* pair, (C_{ix}, C_{iy}), already scaled appropriately to be in units of arc-sec.
- When STRAP is selected as the tip-tilt sensor, the real-time APD quad-cell flux values are used in a centroid computation that is the similar to the steps described above in Eqs (1) – (5) with two exceptions: a) there is no flat field compensation, and b) the resulting centroids are multiplied by the STRAP system matrix to convert from quad-cell units to arc-sec and to convert for any rotational difference between the STRAP sensor and the (*x, y*) coordinate system of the tip-tilt controller:

$$\begin{bmatrix} C_{ix} \\ C_{iy} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix} \begin{bmatrix} C_{Sx} \\ C_{Sy} \end{bmatrix} \tag{10}$$

  where (C_{Sx}, C_{Sy}) are the STRAP centroids. The STRAP system matrix values are determined offline using an IDL calibration tool.
- An angular offset is subtracted from the input tip-tilt error:

$$C_x = C_{ix} - C_{ox} \tag{11}$$

and

$$C_y = C_{iy} - C_{oy} \tag{12}$$

where C_{ox} and C_{oy} are the angular offsets and C_x and C_y are the final tip-tilt errors to be processed. The angular offsets are used to adjust the null point of the controller. This control point offset is used to offload tip-tilt from the Keck Angle Tracker, for differential atmospheric refraction (DAR) compensation, and for general calibration purposes.

- The tip-tilt errors are passed through a 3rd order control law servo to compute the mirror position commands:

$$Y(z) = H(z)U(z) \tag{13}$$

in the Z-transform domain, where $U(z)$ are the inputs (the final tip-tilt errors, C) and $Y(z)$ are the outputs (the mirror position commands, M). $H(z)$ is the transfer function, and is defined as:

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}}{1 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}} \tag{14}$$

In the discrete domain, the outputs, $y[n]$, at time index n, are related to the inputs, $u[n]$ by:

$$\begin{aligned} y[n] = &-b_1 y[n-1] - b_2 y[n-2] - b_3 y[n-3] + \\ &a_0 u[n] + a_1 u[n-1] + a_2 u[n-2] + a_3 u[n-3]. \end{aligned} \tag{15}$$

The u and y variables above are vectors. The inputs, $u[n]$, are the final tip-tilt errors and the outputs, $y[n]$, are the mirror position commands. The a and b variables are the control law coefficients, and z is the Z-Transform domain variable, where z^{-1} represents a unit time delay. The indices n, n-1, n-2, and n-3 refer the current value and the three previous values of each vector.

- If the servo loop is open, the mirror position commands are set to zero (mid-range of the mirror). If the servo loop is closed, the mirror position commands are the output of the control law.
- A variable mirror offset is added to the mirror position. This allows for moving the mirror in open-loop mode.
- If the disturbance option is enabled, values from a disturbance vector are added to the offset mirror position commands:

$$M_f = M + M_d[i] \tag{16}$$

where M_f are the final mirror position commands, M are the offset mirror position commands, and $M_d[i]$ is the i^{th} entry into the disturbance vector. The disturbance vector is a circular buffer containing (x, y) mirror position offset pairs. The index into the disturbance vector is incremented in a circular fashion at the end of each loop cycle. This allows for the simulation of atmospheric turbulence and dithering of the mirror at frequencies beyond the cutoff of the digital controller to calculate the centroid gain due to variable spot size.

- For UTT only, a rotation by a variable angle, θ, is applied:

$$\begin{bmatrix} M_{rx} \\ M_{ry} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} M_{fx} \\ M_{fy} \end{bmatrix} \tag{17}$$

where M_{rx} and M_{ry} are the rotated mirror commands. If no rotation is performed, M_{rx} and M_{ry} are equal to M_{fx} and M_{fy}, the final mirror commands. The rotation is required for UTT because the tip-tilt mirror is mounted on the telescope and rotates in azimuth with respect to the wavefront sensor depending on the telescope and rotator positions.

- The final mirror commands are sent to the TRS as telemetry.

- The final mirror position commands (rotated, in the case of UTT) are sent to the CLMP to move the mirror to the desired position.

3.7 CLMP

A block diagram of the closed-loop mirror positioning controller (CLMP) is shown below in Figure 4.

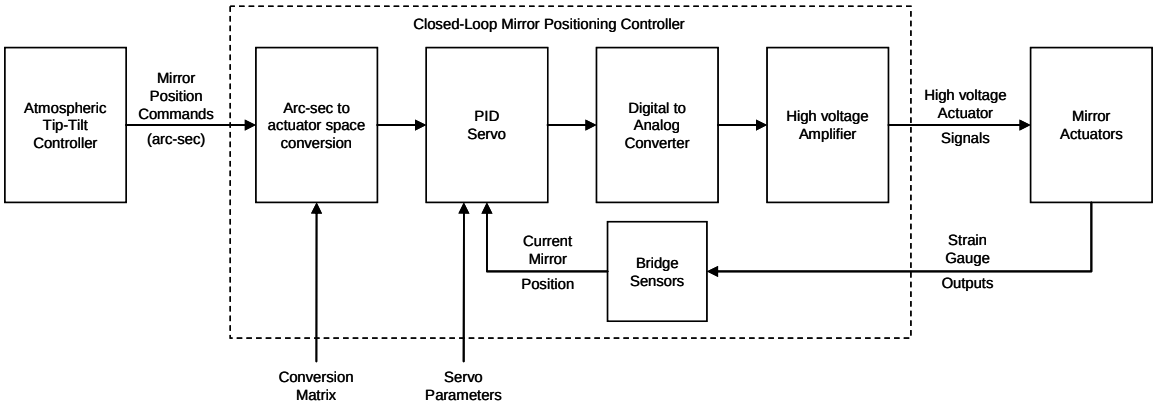

Figure 4: A block diagram of the closed-loop mirror positioning controller.

The CLMP functions as follows:

- The mirror position commands (arc-sec) are received from the tip-tilt mirror controller.
- The mirror position commands are converted from angle space (arc-sec) to actuator space (volts or digital numbers for input to a digital-to-analog converter (DAC)):

$$M_a = RM \tag{18}$$

  Where M is the vector of mirror position commands, R is the reconstruction matrix, and M_a are the mirror commands in actuator space. The values of R have been previously determined by moving the mirror in its independent modes to compute the influence matrix and computing the pseudo-inverse of the system matrix. In the case of DTT, this includes the conversion from (x, y) space to 3-actuator space.
- The mirror actuator commands are checked for values that exceed the range of the mirror and clipped appropriately.
- The mirror actuator commands are passed through a high-bandwidth 4th order servo controller to move the mirror to the desired position.
- The digital mirror actuator commands are sent to the DAC and converted to low-voltage analog signals.
- The low voltage analog signals are sent to a high voltage amplifier to generate the high voltage actuator commands that are used to drive the mirror actuators.
- The strain gauges are read using a bridge sensor to determine the current mirror position and feed it back into the controller.
- The controller uses the strain gauge feedback in the servo loop to drive the mirror to the desired position.