Use “Evaluation > Evaluate Notebook” in *Mathematica* menu on the top, before using the notebook.

Mount Models

Mark Cornell, Walter Moreira, Tom Rafferty

Monday, June 20, 2011

Quick Introduction (start here)

This notebook describes the linear transformations between some of the coordinate frames in the HET telescope.

The important sections to play with and modify are:

- Parameters: you can change some of the physical parameters of the telescope.
- Telescope commanding the SIRP: ideal telescope (no mount models) visualization by setting the SIRP coordinates
- Adding and Using Mount Models: how to define functions representing the several mount models
- Trajectories: visual display of trajectories with a toy mount model of rail sagging

Setting up

Helper function to compute projections of a vector onto the XZ and YZ planes:

```
angleProjXZ[v_] :=
 Module[
  {vxz},
  vxz = v - Projection[v, {0, 1, 0}];
  -Sign[vxz[[1]]] VectorAngle[vxz, {0, 0, 1}]
 ]
```

```
angleProjYZ[v_] :=
 Module[
  {vyz},
  vyz = v - Projection[v, {1, 0, 0}];
  -Sign[vyz[[2]]] VectorAngle[vyz, {0, 0, 1}]
 ]
```

Helper function to compute rotation of transformed Y axis with intersection of original YZ plane and plane orthogonal to w:

```
rotationY[y_, w_] :=
 Module[
  {yp, wp},
  yp = Cross[w, {1, 0, 0}];
  wp = Cross[y, yp];
  If[Abs[VectorAngle[wp, w]] ≥ 1, 1, -1] VectorAngle[yp, y]
 ]
```

Rotations

We write rotations as in the document “Observations in Pointing the HET”:

```
Rx[alpha_] := {{1, 0, 0, 0}, {0, Cos[alpha], Sin[alpha], 0},
  {0, -Sin[alpha], Cos[alpha], 0}, {0, 0, 0, 1}}
```

```
Ry[alpha_] := {{Cos[alpha], 0, -Sin[alpha], 0},
  {0, 1, 0, 0}, {Sin[alpha], 0, Cos[alpha], 0}, {0, 0, 0, 1}}
```

```
Rz[alpha_] := {{Cos[alpha], -Sin[alpha], 0, 0},
  {Sin[alpha], Cos[alpha], 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}
```

The following matrix T transform a vertical vector `v=(0,0,1)` such that the projection of the T(v) onto XZ and YZ form angles phi and theta with the vertical, respectively, and such that the transformed of the Y axis forms an angle rho with the intersection of the plane perpendicular to T(v) and the YZ plane.

The ranges for the angles are:

$-\pi \le \theta \le \pi,\ -\pi \le \phi \le \pi,\ -\pi \le \rho \le \pi$

```
tilde[theta_, phi_] :=
  Sign[phi] ArcCos[Cos[phi] / Sqrt[Cos[theta]^2 Sin[phi]^2 + Cos[phi]^2]];
```

```
R[theta_, phi_, rho_] := Module[{phitilde},
  phitilde = tilde[theta, phi];
  Dot[Rx[-theta], Ry[phitilde], Rz[rho]]
 ]
```

Checking correctness of R:

```
t = R[Pi / 3, Pi / 5, Pi / 6];
```

`w` is the transformation of the Z axis:

```
w = Dot[t, {0, 0, 1, 1}][[1 ;; 3]]
```

$$\left\{-\sqrt{1-\frac{\left(1+\sqrt{5}\right)^2}{16\left(\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2\right)}},\ -\frac{\sqrt{3}\left(1+\sqrt{5}\right)}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}},\ \frac{1+\sqrt{5}}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}\right\}$$

wxz is the projection of **w** onto the XZ plane:

```
wxz = w - Projection[w, {0, 1, 0}]
```

$$\left\{-\sqrt{1-\frac{\left(1+\sqrt{5}\right)^2}{16\left(\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2\right)}},\ \frac{1}{8}\left(1+\sqrt{5}\right)\sqrt{\frac{3}{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}-\frac{\sqrt{3}\left(1+\sqrt{5}\right)}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}},\ \frac{1+\sqrt{5}}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}\right\}$$

The following two must coincide:

```
{VectorAngle[wxz, {0, 0, 1}] // N, angleProjXZ[wxz] // N}
```

```
{0.628319, 0.628319}
```

```
Pi / 5 // N
```

```
0.628319
```

wyz is the projection of **w** onto the YZ plane:

```
wyz = w - Projection[w, {1, 0, 0}]
```

$$\left\{0, -\frac{\sqrt{3}\ \left(1+\sqrt{5}\right)}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}, \frac{1+\sqrt{5}}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}\right\}$$

The following two must coincide:

```
{VectorAngle[wyz, {0, 0, 1}] // N, angleProjYZ[wyz] // N}
```

```
{1.0472, 1.0472}
```

```
Pi / 3 // N
```

```
1.0472
```

yt is the transformation of the Y axis

```
yt = Dot[t, {0, 1, 0, 1}][[1 ;; 3]]
```

$$\left\{-\frac{1+\sqrt{5}}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}, \frac{\sqrt{3}}{4}+\frac{1}{4}\sqrt{3\left(1-\frac{\left(1+\sqrt{5}\right)^2}{16\left(\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2\right)}\right)}, \right.$$

$$\left. \frac{3}{4}-\frac{1}{4}\sqrt{1-\frac{\left(1+\sqrt{5}\right)^2}{16\left(\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2\right)}}\right\}$$

u is the intersection of the plane perpendicular to **w** and the YZ plane:

```
u = Cross[w, {1, 0, 0}]
```

$$\left\{0,\ \frac{1}{8\sqrt{\frac{5}{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}}+\frac{1}{8\sqrt{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}},\right.$$

$$\left.\frac{1}{8\sqrt{\frac{3}{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}}+\frac{1}{8\sqrt{\frac{15}{\frac{1}{4}\left(\frac{5}{8}-\frac{\sqrt{5}}{8}\right)+\frac{1}{16}\left(1+\sqrt{5}\right)^2}}}\right\}$$

The following two must coincide:

```
{VectorAngle[u, yt] // N, rotationY[yt, w] // N}
```

```
{0.523599, 0.523599}
```

```
Pi / 6 // N
```

```
0.523599
```

Translations

```
T[x_, y_, z_] := {{1, 0, 0, x}, {0, 1, 0, y}, {0, 0, 1, z}, {0, 0, 0, 1}}
```

Drawing

Reference frame

```
axesReference[OptionsPattern[{refLength → 10}]] :=
 Graphics3D[{Thin, GrayLevel[.5], Dashed, Arrow[{{0, 0, 0}, #}]}] & /@
  {{OptionValue[refLength], 0, 0},
   {0, OptionValue[refLength], 0}, {0, 0, OptionValue[refLength]}}
```

Coordinate Frame

Draw a coordinate frame given by a matrix of change of base.

x is Red, **y** is Green, **z** is Blue

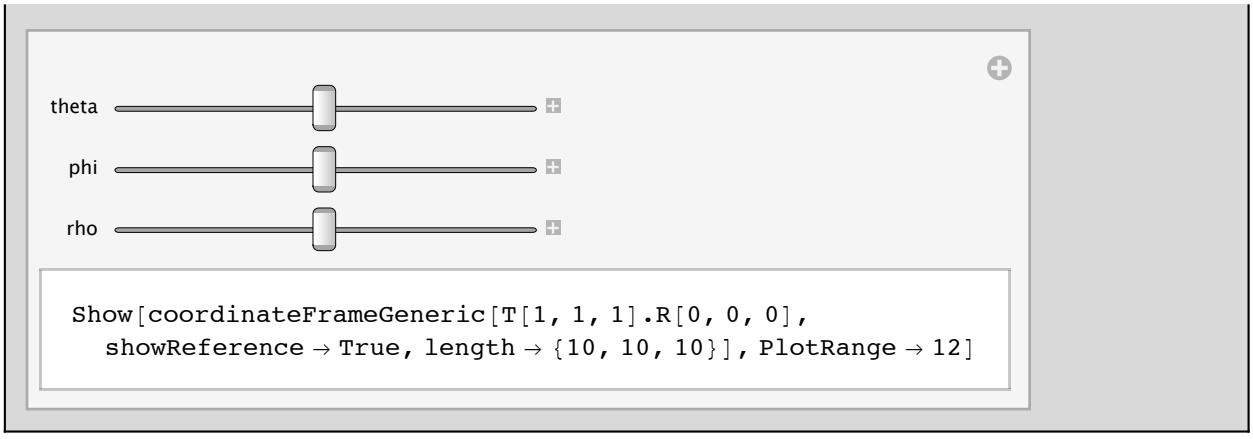
Optional arguments are:

- **showReference**: display reference frame
- **length**: list of 3 numbers to use as length for the axis

```
coordinateFrameGeneric[t_,
  OptionsPattern[{showReference → False, length → {10, 10, 10}}]] :=
 Module[
  {lengthX, lengthY, lengthZ,
   origin,
   ox, oy, oz, T,
   xaxis, yaxis, zaxis,
   axesRef, axes},
  {lengthX, lengthY, lengthZ} = OptionValue[length];
  ox = {lengthX, 0, 0, 1};
  oy = {0, lengthY, 0, 1};
  oz = {0, 0, lengthZ, 1};
  origin = (t.{0, 0, 0, 1})[[1 ;; 3]];
  xaxis =
   Graphics3D[{Red, Arrowheads[0.02], Arrow[ {origin, Dot[t, ox][[1 ;; 3]]}]}];
  yaxis = Graphics3D[{Green, Arrowheads[0.02],
     Arrow[{ origin, Dot[t, oy][[1 ;; 3]]}]}];
  zaxis = Graphics3D[{Blue, Arrowheads[0.02],
     Arrow[{ origin, Dot[t, oz][[1 ;; 3]]}]}];
  axesRef = axesReference[refLength → lengthX];
  axes =
   Join[If[OptionValue[showReference], axesRef, {}], {xaxis, yaxis, zaxis}];
  Show[axes, Boxed → False]
 ]
```

Example:

```
Manipulate[
 Show[coordinateFrameGeneric[T[1, 1, 1].R[theta Degree, phi Degree, rho Degree],
   showReference → True, length → {10, 10, 10}], PlotRange → 12],
 {{theta, 0}, -180, 180}, {{phi, 0}, -180, 180}, {{rho, 0}, -180, 180}]
```



Utility to display a panel with three graphics and titles:

```
panel[g_, g1_, g2_, title1_, title2_] :=
 Panel[
  Column[{
    Show[Graphics[g]],
    Row[{
      Column[{
        Text[Style[title1, "Subsection"]],
        Graphics[g1, BaselinePosition → Top]
       },
       Alignment → {Center, Top},
       Spacings → 1,
       BaselinePosition → Top
      ],
      Text["          "],
      Column[{
        Text[Style[title2, "Subsection"]],
        Graphics[g2, BaselinePosition → Top]
       },
       Alignment → Center,
       Spacings → 1,
       BaselinePosition → Top
      ]
     },
     BaselinePosition → Top
    ]},
   Alignment → {Center, Center},
   Spacings → 2
  ]
 ]
```

Parameters

A list of all the static parameters:

```
Clear[HET]
```

```
(* Origin of TWF in ITF coordinates *)
HET[X_TWF] = 0;
HET[Y_TWF] = 0;
HET[Z_TWF] = 400;

(* Orientation of the Z axis of TWF in ITF *)
HET[Theta_TWF] = 0;
HET[Phi_TWF] = 0;
HET[Rho_TWF] = 0;
```

```
(* Distance in Z of from the base of the hexapod to the top hexagon *)
HET[Z_HLF] = 400;

(* distance from the control point to the top of the hexapod *)
(* computed as height of hexapod
 minus distance from CP to bottom of hexapod *)
HET[D_CP] = 1039.7807 - 756.99549;

(* Origin of CF in HUF coordinates *)
HET[X_CF] = 0;
HET[Y_CF] = 0;
HET[Z_CF] = -50;

(* Orientation of the Z axis of CF in HUF *)
HET[Theta_CF] = 0;
HET[Phi_CF] = 0;
HET[Rho_CF] = 0;

(* Origin of FPAF in CF coordinates *)
HET[X_FPAF] = 0;
HET[Y_FPAF] = 0;
HET[Z_FPAF] = 80;

(* Orientation of the Z axis of FPAF in CF *)
HET[Theta_FPAF] = 0;
HET[Phi_FPAF] = 0;
HET[Rho_FPAF] = 0;

(* Distance from the origin of CF to the SIRP *)
(* computed as distance of SIRP from bottom of hexapod +
 height of hexapod - distance of corrector from top of the hexapod *)
HET[D_SIRP] := 665.5327 + 1039.7807 + HET[Z_CF];

(* Dimensions of the bridge *)
HET[Bridge_X] = 500;
HET[Bridge_Y] = 2 * 1850;

(* Range of movement for the bridge *)
HET[BridgeRange_X] = 1850;
HET[BridgeRange_Y] = 1850;

(* Range of movement for the SIRP in Z *)
HET[SIRPMin_Z] = -84;
HET[SIRPMax_Z] = 357;

(* Range of movement for the angles (in degrees) *)
HET[ThetaRange] = 9.4;
HET[PhiRange] = 9.4;
HET[RhoRange] = 24;
```

From ITF to TWF

See definition of ITF in “Observations in Pointing the HET”.

TWF (tracker working frame) lies somewhere on the top of the structure. The origin is a point with coordinates in ITF: (x_{TWF}, y_{TWF}, z_{TWF}). The z axis of TWF is oriented according to (θ, ϕ) in ITF, and the y axis of TWF is rotated an angle ρ from the y axis of ITF.

```
TWFtoITF[dXTwf_, dYTwf_, dZTwf_, dThetaTwf_, dPhiTwf_, dRhoTwf_][
  xL_, xU_, x_, y_] :=
 Module[
  {rx, ry, rz, rtheta, rphi, rrho},
  rx = HET[X_TWF] + dXTwf[xL, xU, x, y];
  ry = HET[Y_TWF] + dYTwf[xL, xU, x, y];
  rz = HET[Z_TWF] + dZTwf[xL, xU, x, y];
  rtheta = HET[Theta_TWF] + dThetaTwf[xL, xU, x, y];
  rphi = HET[Phi_TWF] + dPhiTwf[xL, xU, x, y];
  rrho = HET[Rho_TWF] + dRhoTwf[xL, xU, x, y];
  T[rx, ry, rz].R[rtheta, rphi, rrho]
 ]
```

From TWF to HLF

HLF (hexapod lower frame) is a coordinate frame attached to the base of the hexapod (lower strut joints). It translates from the TWF as the carriage moves in X and Y. The origin is in the center of the base, below the control point. Ideally, its axis are parallel to the TWF, but we allow for errors depending on X_L, X_U, X, Y (X_L is X position of the lower part of the bridge, and X_U is the X position of the upper part of the bridge).

```
HLFtoTWF[dXHlf_, dYHlf_, dZHlf_, dThetaHlf_, dPhiHlf_, dRhoHlf_][
  xL_, xU_, x_, y_] :=
 Module[
  {rx, ry, rz, rtheta, rphi, rrho},
  rx = x + dXHlf[xL, xU, x, y];
  ry = y + dYHlf[xL, xU, x, y];
  rz = HET[Z_HLF] + dZHlf[xL, xU, x, y];
  rtheta = dThetaHlf[xL, xU, x, y];
  rphi = dPhiHlf[xL, xU, x, y];
  rrho = dRhoHlf[xL, xU, x, y];
  T[rx, ry, rz].R[rtheta, rphi, rrho]
 ]
```

From HLF to HUF

The HUF (hexapod upper frame) is a reference frame attached to the upper strut joints. It tilts to orient the hexapod, and it also translates to keep the control point right above the center of the base of the hexapod. The static parameter `d` is the distance from the top of the hexapod to the control point.

z is the coordinate in HLF of the origin of HUF. θ, ϕ is the orientation of the z axis of HUF in HLF.

```
HUFtoHLF[dXHuf_, dYHuf_, dZHuf_, dThetaHuf_, dPhiHuf_, dRhoHuf_][
  xL_, xU_, x_, y_, z_, theta_, phi_] :=
 Module[
  {rx, ry, rz, rtheta, rphi, rrho, rot, tr},
  rx = dXHuf[xL, xU, x, y, z, theta, phi];
  ry = dYHuf[xL, xU, x, y, z, theta, phi];
  rz = z + dZHuf[xL, xU, x, y, z, theta, phi];
  rtheta = theta + dThetaHuf[xL, xU, x, y, z, theta, phi];
  rphi = phi + dPhiHuf[xL, xU, x, y, z, theta, phi];
  rrho = dRhoHuf[xL, xU, x, y, z, theta, phi];
  rot = R[rtheta, rphi, rrho];
  w = rot.{0, 0, HET[D_CP], 0};
  tr = T @@ ({rx, ry, rz} + {w[[1]], w[[2]], 0});
  tr.rot
 ]
```

From HUF to CF

The CF (corrector frame) is a frame attached to the corrector. The static parameters are the location and orientation of the corrector with respect to HUF:

$$(x_{CF},\ y_{CF},\ z_{CF},\ \Theta_{CF},\ \phi_{CF},\ \rho_{CF})$$

The z axis of this frame is w. The SIRP has coordinates $(0, 0, -D_{SIRP})$ in this frame.

```
CFtoHUF[dXCf_, dYCf_, dZCf_, dThetaCf_, dPhiCf_, dRhoCf_][
  xL_, xU_, x_, y_, z_, theta_, phi_] :=
 Module[
  {rx, ry, rz, rtheta, rphi, rrho},
  rx = HET[X_CF] + dXCf[xL, xU, x, y, z, theta, phi];
  ry = HET[Y_CF] + dYCf[xL, xU, x, y, z, theta, phi];
  rz = HET[Z_CF] + dZCf[xL, xU, x, y, z, theta, phi];
  rtheta = HET[Theta_CF] + dThetaCf[xL, xU, x, y, z, theta, phi];
  rphi = HET[Phi_CF] + dPhiCf[xL, xU, x, y, z, theta, phi];
  rrho = HET[Rho_CF] + dRhoCf[xL, xU, x, y, z, theta, phi];
  T[rx, ry, rz].R[rtheta, rphi, rrho]
 ]
```

From CF to FPAF

The FPAF (focal plane assembly frame) is attached to the focal plane. The focal plane can rotate with respect to CF.

```
FPAFtoCF[dXFpaf_, dYFpaf_, dZFpaf_, dThetaFpaf_, dPhiFpaf_, dRhoFpaf_][
  xL_, xU_, x_, y_, z_, theta_, phi_, rho_] :=
 Module[
  {rx, ry, rz, rtheta, rphi, rrho},
  rx = HET[X_FPAF] + dXFpaf[xL, xU, x, y, z, theta, phi, rho];
  ry = HET[Y_FPAF] + dYFpaf[xL, xU, x, y, z, theta, phi, rho];
  rz = HET[Z_FPAF] + dZFpaf[xL, xU, x, y, z, theta, phi, rho];
  rtheta = HET[Theta_FPAF] + dThetaFpaf[xL, xU, x, y, z, theta, phi, rho];
  rphi = HET[Phi_FPAF] + dPhiFpaf[xL, xU, x, y, z, theta, phi, rho];
  rrho = rho + HET[Rho_FPAF] + dRhoFpaf[xL, xU, x, y, z, theta, phi, rho];
  T[rx, ry, rz].R[rtheta, rphi, rrho]
 ]
```

Ideal Transformations

These are the transformations in absence of any mount model (but things can still be misaligned). We use it invert it and get the values for $(x_L, x_U, x, y, z, \theta, \phi, \rho)$ in terms of the position of the SIRP.

```
CFtoITFideal[x_, y_, z_, theta_, phi_] :=
 TWFtoITF[0 &, 0 &, 0 &, 0 &, 0 &, 0 &][x, x, x, y].
  HLFtoTWF[0 &, 0 &, 0 &, 0 &, 0 &, 0 &][x, x, x, y].
  HUFtoHLF[0 &, 0 &, 0 &, 0 &, 0 &, 0 &][x, x, x, y, z, theta, phi].
  CFtoHUF[0 &, 0 &, 0 &, 0 &, 0 &, 0 &][x, x, x, y, z, theta, phi]
```

For a given translation of the bridge on (x, y) in TWF, a vertical displacement of the origin of HUF by z in TWF, and an orientation of the top of the hexapod by (θ, ϕ) in TWF, we get a position of the SIRP in ITF with the following function:

```
SIRPideal[x_, y_, z_, theta_, phi_] :=
 Module[
  {t},
  t = CFtoITFideal[x, y, z, theta, phi];
  (t.{0, 0, -HET[D_SIRP], 1})[[1 ;; 3]]
 ]
```

```
SIRPideal[0, 0, 0, 0, Pi / 4]
```

```
{1005.88, 0., -405.839}
```

```
wideal[x_, y_, z_, theta_, phi_] :=
 Module[
  {t, w},
  t = CFtoITFideal[x, y, z, theta, phi];
  w = (t.{0, 0, 1, 0})[[1 ;; 3]];
  {angleProjYZ[w], angleProjXZ[w]}
 ]
```

Given an orientation of w by the angles (θ, ϕ) in ITF, find the angles we have to orient the HUF so that the corrector's axis coincides with w:

```
orientation[thetaSIRP_, phiSIRP_] :=
 Module[
  {t, w},
  t = Inverse[R[HET[Theta_TWF], HET[Phi_TWF], HET[Rho_TWF]]].
    R[thetaSIRP, phiSIRP, 0].Inverse[R[HET[Theta_CF], HET[Phi_CF], HET[Rho_CF]]];
  w = (t.{0, 0, 1, 0})[[1 ;; 3]];
  {angleProjYZ[w], angleProjXZ[w]}
 ]
```

Given the full position of the SIRP and orientation of w, find the position of the bridge and the height of the upper hexapod frame, so that the SIRP lies in the given point and the corrector's axis coincides with w:

```
position[xSIRP_, ySIRP_, zSIRP_, thetaSIRP_, phiSIRP_] :=
 Module[
  {theta, phi, v, w},
  {theta, phi} = -orientation[thetaSIRP, phiSIRP];
  v = R[theta, phi, 0].{0, 0, HET[D_CP], 0};
  w = R[theta, phi, 0].T[HET[X_CF], HET[Y_CF], HET[Z_CF]].
    R[HET[Theta_CF], HET[Phi_CF], HET[Rho_CF]].{0, 0, -HET[D_SIRP], 1};
  {xSIRP - HET[X_TWF] + w[[1]] + v[[1]],
   ySIRP - HET[Y_TWF] + w[[2]] + v[[2]],
   -w[[3]] + zSIRP - HET[Z_TWF] - HET[Z_HLF]}
 ]
```

```
SIRPideal[0, 0, 0, Pi / 3, Pi / 4]
```

```
{636.174, 1101.89, 37.3607}
```

```
position[636.1739465499472`,
 1101.885597876116`, -562.6393370682579`, Pi / 3, Pi / 4]
```

$$\{-1.13687 \times 10^{-13}, 0., -600.\}$$

Visual Model

Full Telescope

```
telescope[
  mountTWF_, (* mount model for TWF wrt ITF *)
  mountHLF_, (* mount model for HLF wrt TWF *)
  mountHUF_, (* mount model for HUF wrt HLF *)
```

```
 mountCF_, (* mount model for CF wrt HUF *)
 mountFPAF_, (* mount model for FPAF wrt CF *)
 xL_, xU_, x_, y_, z_, theta_, phi_, rho_,
 OptionsPattern[{
   ITF → True, HLF → True, TWF → True, HUF → True, CF → True, FPAF → True,
   showBridge → True, showControlPoint → True, showUpperHexapod → True,
   showCorrector → True, showFPA → True, showSIRP → True}]] :=
Module[
 {twftoitf, hlftotwf, huftohlf, cftohuf, fpaftocf,
  xITF, yITF, zITF, xLITF, xUITF, yLITF, yUITF, zLITF, zUITF, dummy,
  bridge, upperHexapod, corrector, fpa,
  sirpPoint, len},
 twftoitf = (TWFtoITF @@ mountTWF)[xL, xU, x, y];
 hlftotwf = (HLFtoTWF @@ mountHLF)[xL, xU, x, y];
 huftohlf = (HUFtoHLF @@ mountHUF)[xL, xU, x, y, z, theta, phi];
 cftohuf = (CFtoHUF @@ mountCF)[xL, xU, x, y, z, theta, phi];
 fpaftocf = (FPAFtoCF @@ mountFPAF)[xL, xU, x, y, z, theta, phi, rho];
 {xITF, yITF, zITF, dummy} = hlftotwf.twftoitf.{0, 0, 0, 1};
 {xLITF, yLITF, zLITF, dummy} = twftoitf.{xL, -HET[Bridge_Y] / 2, HET[Z_HLF], 1};
 {xUITF, yUITF, zUITF, dummy} = twftoitf.{xU, HET[Bridge_Y] / 2, HET[Z_HLF], 1};
 bridge = Module[
   {hexBase, bridgePlatform},
   hexBase = Graphics3D[
     Polygon[Map[
       {xITF + #[[1]] HET[Bridge_X] / 4, yITF + #[[2]] HET[Bridge_X] / 4, zITF} &,
       {{-1, -1}, {1, -1}, {1, 1}, {-1, 1}}]]];
   bridgePlatform = Graphics3D[
     Polygon[{
       {xLITF - HET[Bridge_X] / 2, yLITF, zLITF},
       {xLITF + HET[Bridge_X] / 2, yLITF, zLITF},
       {xUITF + HET[Bridge_X] / 2, yUITF, zUITF},
       {xUITF - HET[Bridge_X] / 2, yUITF, zUITF}}]];
   Show[{hexBase, bridgePlatform}, Boxed → False]
  ];
 upperHexapod = Module[
   {t, hexTop, controlPoint, line, cp},
   t = twftoitf.hlftotwf.huftohlf;
   hexTop = Graphics3D[
     Polygon[Map[
       (t.({HET[Bridge_X] / 4 #[[1]], HET[Bridge_X] / 4 #[[2]], 0, 1}))[[1 ;; 3]] &,
       {{-1, -1}, {1, -1}, {1, 1}, {-1, 1}}]]];
   cp = (t.{0, 0, -HET[D_CP], 1})[[1 ;; 3]];
   controlPoint = Graphics3D[{
      PointSize[Large], Point[cp]}];
   line = Graphics3D[{
      Thin, GrayLevel[.5], Dashed,
      Line[{(t.{0, 0, 0, 1})[[1 ;; 3]], cp}]}];
   Show[Join[
     {hexTop},
     If[OptionValue[showControlPoint], {controlPoint, line}, {}]]
   ]
```

```
 ];
corrector = Module[
  {t, corr, sirp, line},
  t = twftoitf.hlftotwf.huftohlf.cftohuf;
  corr = Graphics3D[
    Cylinder[{
      (t.{0, 0, -HET[Bridge_X] / 8, 1})[[1 ;; 3]],
      (t.{0, 0, HET[Bridge_X] / 8, 1})[[1 ;; 3]]},
     HET[Bridge_X] / 8]];
  sirpPoint = (t.{0, 0, -HET[D_SIRP], 1})[[1 ;; 3]];
  sirp = Graphics3D[{
     Thin, GrayLevel[.5], Dashed, Line[{(t.{0, 0, 0, 1})[[1 ;; 3]], sirpPoint}],
     PointSize[Large], Red, Point[sirpPoint]}];
  Show[Join[
    {corr},
    If[OptionValue[showSIRP], {sirp}, {}]]]
 ];
fpa = Module[
  {t, fp, line},
  t = twftoitf.hlftotwf.huftohlf.cftohuf.fpaftocf;
  fp = Graphics3D[
    Polygon[Map[
      (t.{#[[1]] HET[Bridge_X] / 8, #[[2]] HET[Bridge_X] / 8, 0, 1})[[1 ;; 3]] &,
      {{-1, -1}, {1, -1}, {1, 1}, {-1, 1}}]]];
  line = Graphics3D[{
     Thin, GrayLevel[0.5], Dashed,
     Line[{
       (t.{0, 0, 0, 1})[[1 ;; 3]],
       (t.{0, 0, -HET[D_SIRP] - HET[Z_FPAF], 1})[[1 ;; 3]]}]}];
  Show[{fp, line}]
 ];
len = HET[Bridge_Y] / 2;
{Show[Join[
   {},
   If[OptionValue[ITF], {axesReference[refLength → len]}, {}],
   If[OptionValue[TWF],
    {coordinateFrameGeneric[twftoitf, length → {len, len, len}]}, {}],
   If[OptionValue[HLF], {coordinateFrameGeneric[
      twftoitf.hlftotwf, length → {len, len, len}]}, {}],
   If[OptionValue[HUF], {coordinateFrameGeneric[
      twftoitf.hlftotwf.huftohlf, length → {len, len, len}]}, {}],
   If[OptionValue[CF], {coordinateFrameGeneric[
      twftoitf.hlftotwf.huftohlf.cftohuf, length → {len, len, len}]}, {}],
   If[OptionValue[FPAF], {coordinateFrameGeneric[twftoitf.hlftotwf.
       huftohlf.cftohuf.fpaftocf, length → {len, len, len}]}, {}],
   If[OptionValue[showBridge], {bridge}, {}],
   If[OptionValue[showUpperHexapod], {upperHexapod}, {}],
   If[OptionValue[showCorrector], {corrector}, {}],
   If[OptionValue[showFPA], {fpa}, {}]
  ], Boxed → False],
 sirpPoint,
```

```
    twftoitf.hlftotwf.huftohlf.cftohuf.fpaftocf,
    twftoitf.hlftotwf.huftohlf.cftohuf
   }
 ]
```

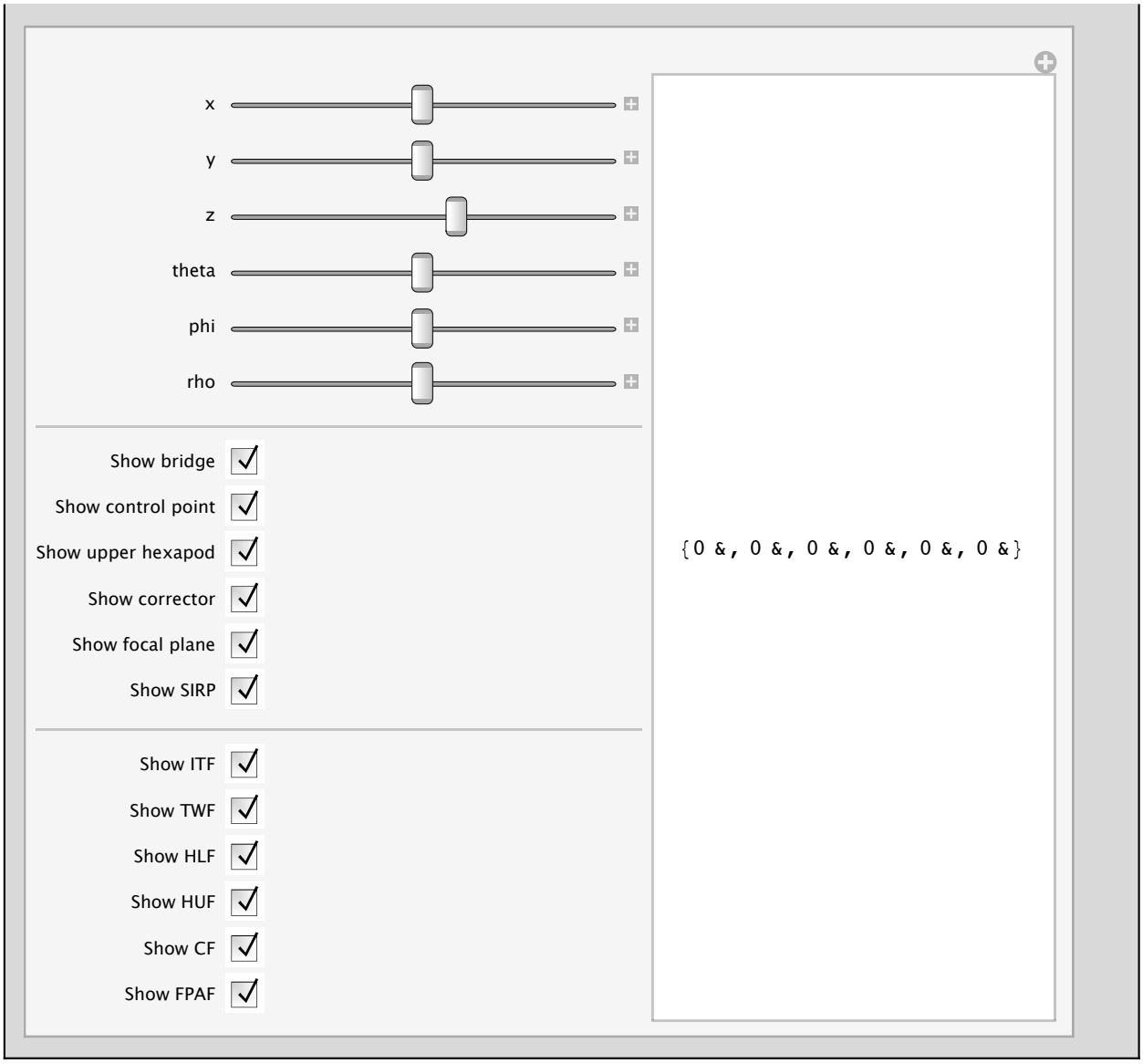
Telescope in commanded position

In this visualization, the variables are:

- (x, y) are the coordinates in TWF of the position of the base of the hexapod,
- z is the third coordinate in HLF of the center of the top of the hexapod,
- (θ, ϕ) are the angles which define the orientation of the perpendicular to the top of the hexapod in HLF,
- ρ is the angle of the FPAF with respect to CF

These are not ITF coordinates

```
Manipulate[telescope[
    {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
    {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
    {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
    {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
    {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
    x, x, x, y, z, theta, phi, rho,
    showBridge → bridge, showControlPoint → cp,
    showUpperHexapod → uh, showCorrector → c, showFPA → fpa, showSIRP → sirp,
    ITF → itf, TWF → twf, HLF → hlf, TWF → twf, HUF → huf,
    CF → cf, FPAF → fpaf][[1]],
 {{x, 0}, -HET[BridgeRange_X], HET[BridgeRange_X]},
 {{y, 0}, -HET[BridgeRange_Y], HET[BridgeRange_Y]}, {{z, 300}, 0, 500},
 {{theta, 0}, -Pi / 4, Pi / 4}, {{phi, 0}, -Pi / 4, Pi / 4}, {{rho, 0}, -Pi / 4, Pi / 4},
 Delimiter,
 {{bridge, True, "Show bridge"}, {True, False}},
 {{cp, True, "Show control point"}, {True, False}},
 {{uh, True, "Show upper hexapod"}, {True, False}},
 {{c, True, "Show corrector"}, {True, False}},
 {{fpa, True, "Show focal plane"}, {True, False}},
 {{sirp, True, "Show SIRP"}, {True, False}},
 Delimiter,
 {{itf, True, "Show ITF"}, {True, False}},
 {{twf, True, "Show TWF"}, {True, False}},
 {{hlf, True, "Show HLF"}, {True, False}},
 {{huf, True, "Show HUF"}, {True, False}},
 {{cf, True, "Show CF"}, {True, False}},
 {{fpaf, True, "Show FPAF"}, {True, False}}
]
```



Telescope commanding the SIRP

Display a projection of two points onto the XY plane of the ITF:

```
projITF[idealSIRP_, realSIRP_] :=
 Module[
  {},
  Show[{
    ListPlot[{{realSIRP[[1 ;; 2]]}, {idealSIRP[[1 ;; 2]]}},
     PlotStyle → {Directive[PointSize[Large], Red],
       Directive[Black, PointSize[Large]]}],
    ParametricPlot[HET[BridgeRange_x] {Sin[u], Cos[u]}, {u, 0, 2 Pi}]
   },
   AspectRatio → 1
  ]
 ]
```

Display a projection on the focal plane:

```
projFPA[idealSIRP_, realSIRP_, t_] :=
 Module[
  {tinv, ideal, real, c, corg},
  tinv = Inverse[t];
  ideal = (tinv.(Join[idealSIRP, {1}]))[[1 ;; 2]];
  real = (tinv.(Join[realSIRP, {1}]))[[1 ;; 2]];
  c = (tinv.{0, 1, 0, 0})[[1 ;; 2]];
  corg = {-HET[Bridge_x] / 2, -HET[Bridge_x] / 2};
  Show[{
    Graphics[{Arrow[{corg, corg + HET[Bridge_x] / 4 c}],
      Arrow[{corg, corg + HET[Bridge_x] / 4 {-c[[2]], c[[1]]}}]}],
    ListPlot[{{real}, {ideal}}, PlotStyle → {Directive[PointSize[Large], Red],
       Directive[PointSize[Large], Black]}],
    ParametricPlot[HET[Bridge_x] {Sin[u], Cos[u]}, {u, 0, 2 Pi}]
   },
   AspectRatio → 1, Axes → True
  ]
 ]
```

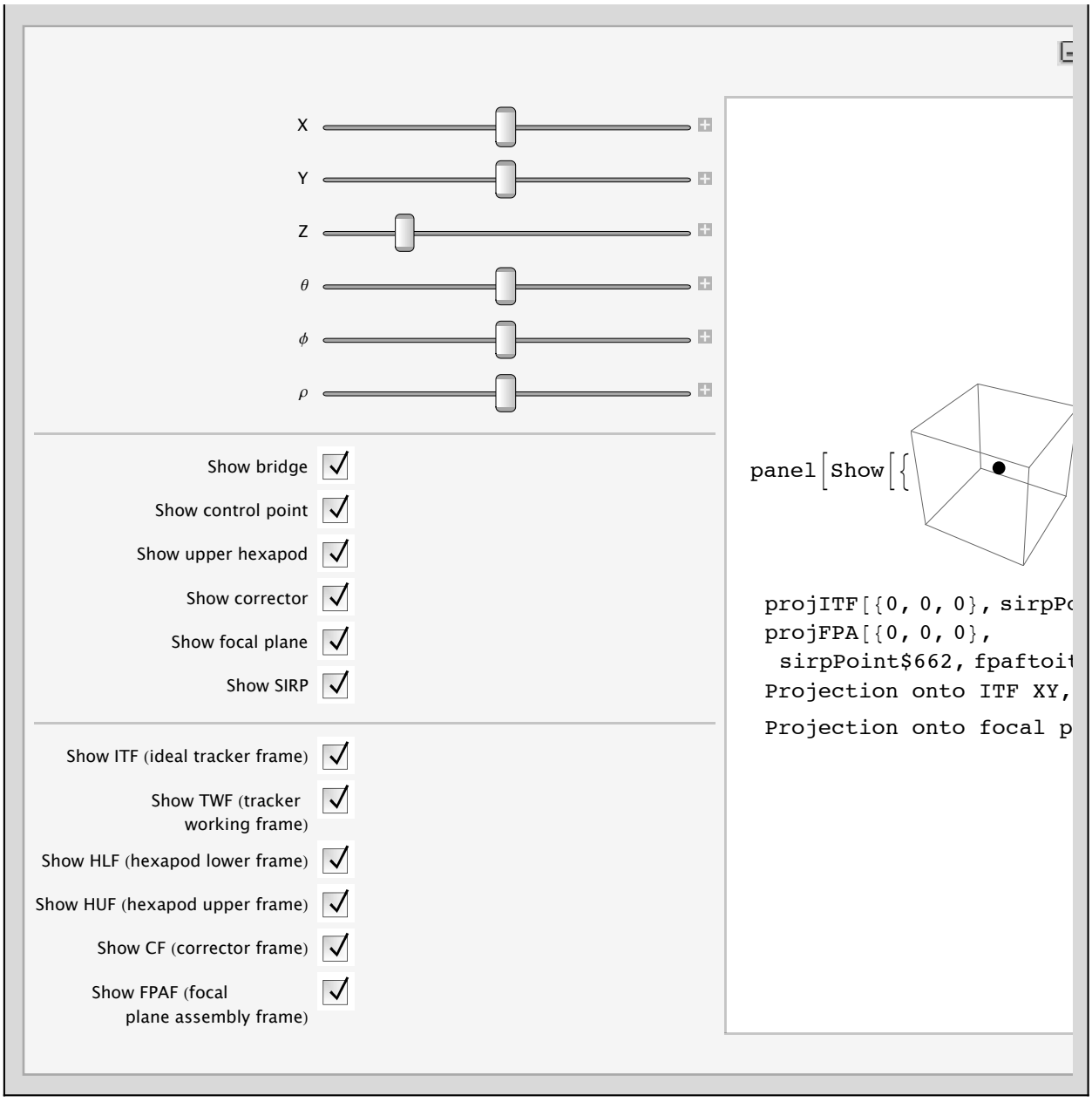
Main function. Use this for experimentation.

```
manipulateTelescope[
  mountTWF_, (* mount model for TWF wrt ITF *)
  mountHLF_, (* mount model for HLF wrt TWF *)
  mountHUF_, (* mount model for HUF wrt HLF *)
  mountCF_, (* mount model for CF wrt HUF *)
  mountFPAF_ (* mount model for FPAF wrt CF *)] :=
 Manipulate[
  Module[
   {x, y, z, theta, phi, sirpPoint, tele, proj, projfpa,
    fpaftoitf, cftoitf, teleModel, thetaSIRP, phiSIRP, rhoSIRP},
   {thetaSIRP, phiSIRP, rhoSIRP} =
    {thetaSIRPDegree, phiSIRPDegree, rhoSIRPDegree} Degree;
   {x, y, z} = position[xSIRP, ySIRP, zSIRP, thetaSIRP, phiSIRP];
```

```
  {theta, phi} = orientation[thetaSIRP, phiSIRP];
  {tele, sirpPoint, fpaftoitf, cftoitf} = telescope[
    mountTWF,
    mountHLF,
    mountHUF,
    mountCF,
    mountFPAF,
    x, x, x, y, z, theta, phi, rhoSIRP,
    showBridge → bridge, showControlPoint → cp, showUpperHexapod → uh,
    showCorrector → c, showFPA → fpa, showSIRP → sirp,
    ITF → itf, TWF → twf, HLF → hlf, HUF → huf, CF → cf, FPAF → fpaf];
  proj = projITF[{xSIRP, ySIRP, zSIRP}, sirpPoint];
  projfpa = projFPA[{xSIRP, ySIRP, zSIRP}, sirpPoint, fpaftoitf];
  teleModel = Show[{Graphics3D[
      {PointSize[Large], Black, Point[{xSIRP, ySIRP, zSIRP}]}], tele}];
  panel[teleModel, proj, projfpa, "Projection onto ITF XY",
   "Projection onto focal plane"]
  (* Grid[{
    {Show[teleModel],SpanFromLeft},
    {Text["Projection onto ITF XY"],Text["Projection onto focal plane"]},
    {Show[proj,ImageSize→Tiny],Show[projfpa,ImageSize→Tiny]}},
   Frame→All]*)],
 {{xSIRP, 0, "X"}, -HET[BridgeRange_X], HET[BridgeRange_X]},
 {{ySIRP, 0, "Y"}, -HET[BridgeRange_Y], HET[BridgeRange_Y]},
 {{zSIRP, 0, "Z"}, HET[SIRPMin_Z], HET[SIRPMax_Z]},
 {{thetaSIRPDegree, 0, "θ"}, -HET[ThetaRange], HET[ThetaRange]},
 {{phiSIRPDegree, 0, "ϕ"}, -HET[PhiRange], HET[PhiRange]},
 {{rhoSIRPDegree, 0, "ρ"}, -HET[RhoRange], HET[RhoRange]},
 Delimiter,
 {{bridge, True, "Show bridge"}, {True, False}},
 {{cp, True, "Show control point"}, {True, False}},
 {{uh, True, "Show upper hexapod"}, {True, False}},
 {{c, True, "Show corrector"}, {True, False}},
 {{fpa, True, "Show focal plane"}, {True, False}},
 {{sirp, True, "Show SIRP"}, {True, False}},
 Delimiter,
 {{itf, True, "Show ITF (ideal tracker frame)"}, {True, False}},
 {{twf, True, "Show TWF (tracker working frame)"}, {True, False}},
 {{hlf, True, "Show HLF (hexapod lower frame)"}, {True, False}},
 {{huf, True, "Show HUF (hexapod upper frame)"}, {True, False}},
 {{cf, True, "Show CF (corrector frame)"}, {True, False}},
 {{fpaf, True, "Show FPAF (focal plane assembly frame)"}, {True, False}},
 AppearanceElements → All
]
```

An example: a telescope with no “dynamic” mount models: (all the functions are constant and equal to 0)

```
manipulateTelescope[
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &},
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}]
```



Adding and using mount models

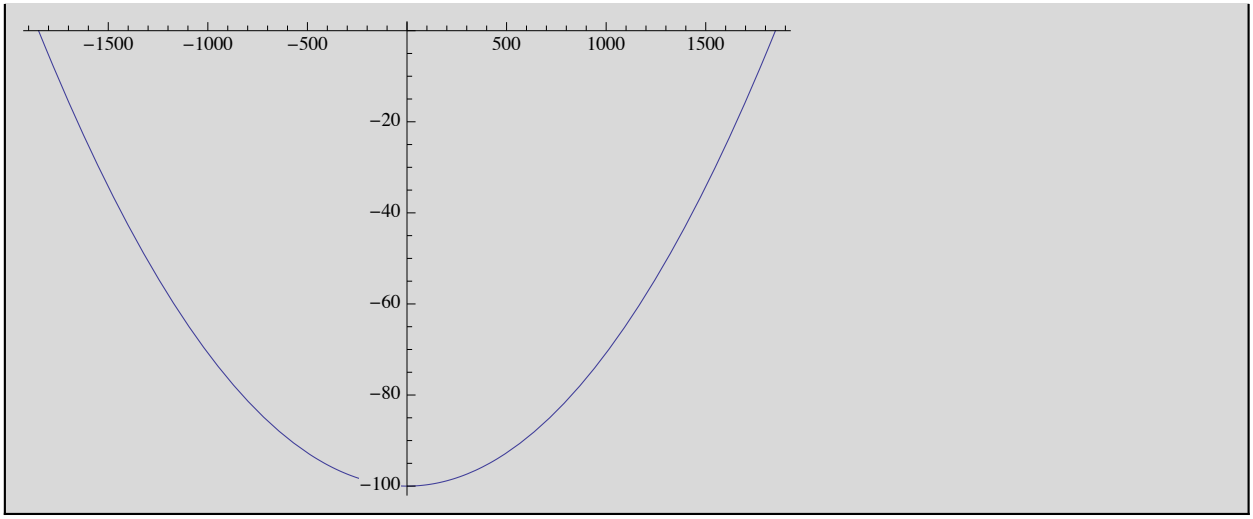
We separate the mount model effects into two kinds:

- static effects: are given in the parameter section, by setting the values for **HET[**parameter name**]**. For example, the coordinates of the corrector frame with respect to the hexapod upper frame
- dynamic effects: are given by functions depending on the position of the bridge, orientation of the hexapod, and rotation of the focal plane

An example of dynamic effect. Let's assume that the Y axis sags as a parabola, and that the base of the hexapod keeps tangent to such parabola. Then, we need to define a dynamic effect for Z and Theta, depending only on y:

```
saggingZ[xL_, xU_, x_, y_] := 100 (y^2 - 1850^2) / 1850^2
```

```
Plot[saggingZ[0, 0, 0, y], {y, -1850, 1850}]
```




```
saggingTheta[xL_, xU_, x_, y_] := D[saggingZ[0, 0, 0, Y], Y] /. Y → y;
```

The parameters to **manipulateTelescope** are 5 rows of 6 elements each. Each element is a function. For example:

```
manipulateTelescope[
  { x1, y1, z1, theta1, phi1, rho1},
  { x2, y2, z2, theta2, phi2, rho2},
  ...]
```

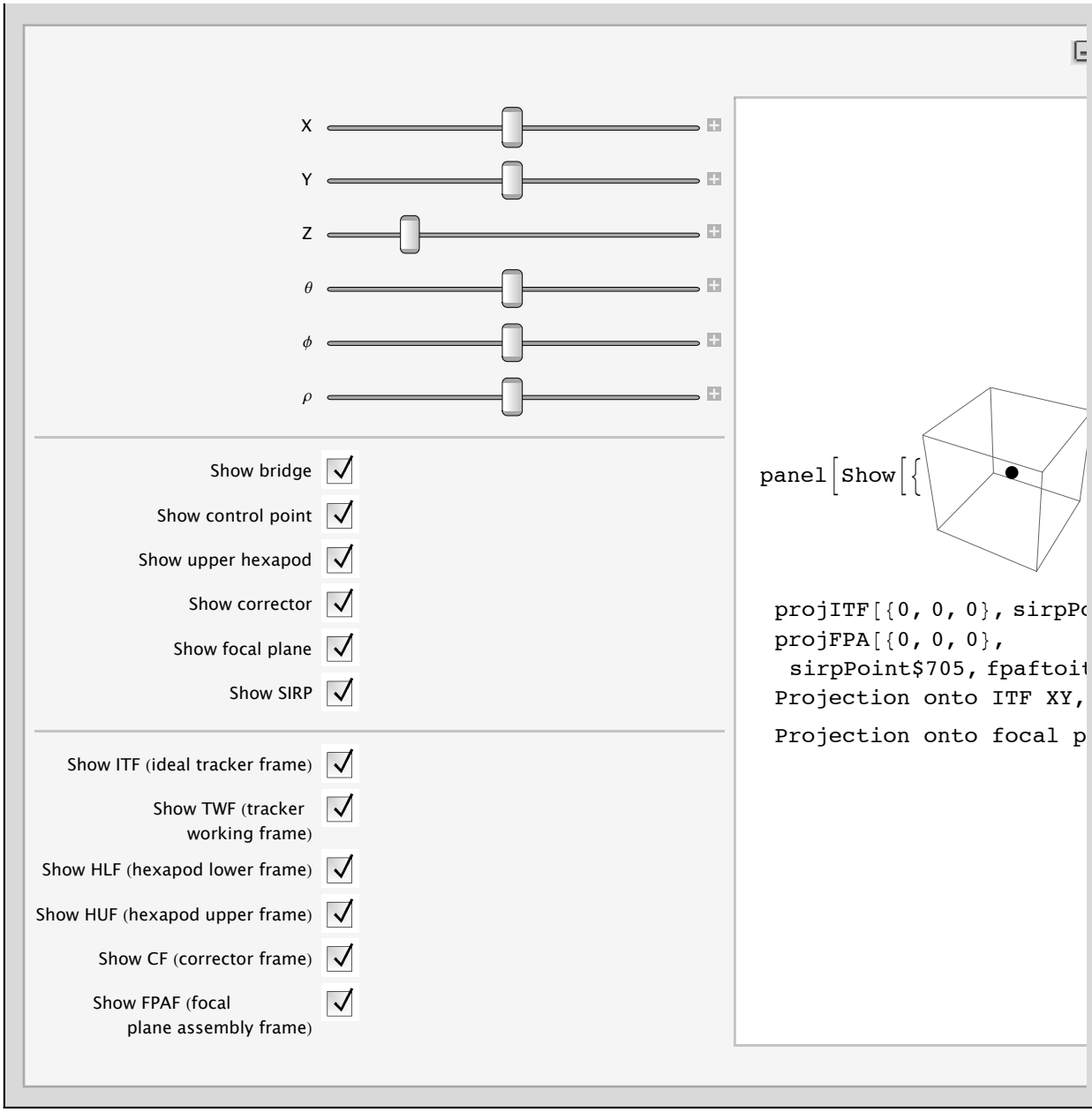
The function x_1 depends on 4 parameters: xL, xU, x, y, where xL is the position of the lower x drive, xU is the position of the upper x drive, x is the position to assign to the tracker (ideally, xL=xU=x, if there is no curling), and y is the position of the y drive. The definition for such a function would be, for example:

```
f[xL_, xU_, x_, y_] := x^2 + y
```

The first two rows (mount models: ITF->TWF and TWF->HLF) depend on xL, xU, x, y.
The third and four row (mount models: HLF->HUF and HUF->CF) depend on xL, xU, x, y, theta, phi
The last row (mount model: CF->FPAF) depend on xL, xU, x, y, theta, phi, rho

For example, to use **saggingTheta** and **saggingZ** (dynamic effects for the mount model TWF->HLF) we do:

```
manipulateTelescope[
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}, (* mount model for TWF wrt ITF *)
 {0 &, 0 &, saggingZ, saggingTheta, 0 &, 0 &},
 (* mount model for HLF wrt TWF *)
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}, (* mount model for HUF wrt HLF *)
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}, (* mount model for CF wrt HUF *)
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &} (* mount model for FPAF wrt CF *)
]
```



Trajectories

Code

```
manipulateTrajectory[
  mountTWF_, (* mount model for TWF wrt ITF *)
  mountHLF_, (* mount model for HLF wrt TWF *)
  mountHUF_, (* mount model for HUF wrt HLF *)
  mountCF_, (* mount model for CF wrt HUF *)
  mountFPAF_ , (* mount model for FPAF wrt CF *)
  trajectory_
 ] :=
 Module[
  {traj, allPoints, allPoints2D, allSIRP, allSIRP2D, f},
  traj = Import[trajectory][[6 ;;, 1 ;; 7]];
  allPoints = traj[[ ;; , 2 ;; 4]];
  allPoints2D = allPoints[[ ;; , 1 ;; 2]];
  f[pt_] :=
   Module[
    {x, y, z, theta, phi, rho,
     xSIRP, ySIRP, zSIRP, thetaSIRPDegree, phiSIRPDegree, rhoSIRPDegree},
    {xSIRP, ySIRP, zSIRP, thetaSIRPDegree, phiSIRPDegree, rhoSIRPDegree} = pt;
    {x, y, z} = position[xSIRP, ySIRP,
      zSIRP, thetaSIRPDegree Degree, phiSIRPDegree Degree];
    {theta, phi} = orientation[thetaSIRPDegree Degree, phiSIRPDegree Degree];
    rho = pt[[6]];
    telescope[mountTWF, mountHLF, mountHUF,
      mountCF, mountFPAF, x, x, x, y, z, theta , phi , rho ][[2]]
   ];
  allSIRP = f /@ traj[[ ;; , 2 ;; 7]];
  allSIRP2D = allSIRP[[ ;; , 1 ;; 2]];
  Manipulate[
   Module[
    {x, y, z, theta, phi, sirpPoint, tele, proj, projfpa,
     fpaftoitf, dummy, teleModel, thetaSIRP, phiSIRP, rhoSIRP},
    {xSIRP, ySIRP, zSIRP} = {traj[[t]][[2]], traj[[t]][[3]], traj[[t]][[4]]};
    {thetaSIRPDegree, phiSIRPDegree, rhoSIRPDegree} =
     {traj[[t]][[5]], traj[[t]][[6]], traj[[t]][[7]]};
    {thetaSIRP, phiSIRP, rhoSIRP} =
     {thetaSIRPDegree, phiSIRPDegree, rhoSIRPDegree} Degree;
    {x, y, z} = position[xSIRP, ySIRP, zSIRP, thetaSIRP, phiSIRP];
    {theta, phi} = orientation[thetaSIRP, phiSIRP];
    {tele, sirpPoint, fpaftoitf, dummy} = telescope[
      mountTWF,
      mountHLF,
      mountHUF,
      mountCF,
      mountFPAF,
      x, x, x, y, z, theta, phi, rhoSIRP,
```

```
      showBridge → bridge, showControlPoint → cp, showUpperHexapod → uh,
      showCorrector → c, showFPA → fpa, showSIRP → sirp,
      ITF → itf, TWF → twf, HLF → hlf, HUF → huf, CF → cf, FPAF → fpaf];
    proj = Show[{projITF[{xSIRP, ySIRP, zSIRP}, sirpPoint],
       Graphics[{PointSize[Small], Point /@ allPoints2D}],
       Graphics[{PointSize[Small], Red, Point /@ allSIRP2D}]}];
    projfpa = projFPA[{xSIRP, ySIRP, zSIRP}, sirpPoint, fpaftoitf];
    teleModel = Show[{Graphics3D[
        {PointSize[Large], Black, Point[{xSIRP, ySIRP, zSIRP}]}], tele,
       Graphics3D[{PointSize[Small], Point /@ allPoints}],
       Graphics3D[{PointSize[Small], Red, Point /@ allSIRP}]
      }];
    panel[teleModel, proj, projfpa,
     "Projection onto ITF XY", "Projection onto focal plane"]
   ],
   {{t, 1, "Trajectory points"}, 1, Length[traj], 1},
   Delimiter,
   {{bridge, True, "Show bridge"}, {True, False}},
   {{cp, True, "Show control point"}, {True, False}},
   {{uh, True, "Show upper hexapod"}, {True, False}},
   {{c, True, "Show corrector"}, {True, False}},
   {{fpa, True, "Show focal plane"}, {True, False}},
   {{sirp, True, "Show SIRP"}, {True, False}},
   Delimiter,
   {{itf, True, "Show ITF (ideal tracker frame)"}, {True, False}},
   {{twf, True, "Show TWF (tracker working frame)"}, {True, False}},
   {{hlf, True, "Show HLF (hexapod lower frame)"}, {True, False}},
   {{huf, True, "Show HUF (hexapod upper frame)"}, {True, False}},
   {{cf, True, "Show CF (corrector frame)"}, {True, False}},
   {{fpaf, True, "Show FPAF (focal plane assembly frame)"}, {True, False}},
   AppearanceElements → All
  ]
 ]
```

Visualization

In this visualization, the **red dot** is the SIRP of the optical system (i.e., a point lying in the optical axis of the corrector, at a distance **HET[D_{SIRP}]** from the origin of the corrector frame). Since the SIRP (red dot) is in the optical axis of the corrector, the projection onto the focal plane (which in this case is perfectly perpendicular to the optical axis of the corrector --though it may include a mount model too--), the SIRP is always in the center of the plot "Projection onto focal plane". The **black dot** near the red dot is the point where we wish to be. For example, if we want to be at the origin of the ITF (x=y=z=0, and theta=phi=rho=0), then the black dot will be exactly at that postion, but the red dot will be (in this case) slightly below the origin, because of the sagging term in the mount models. The goal is to make bring the red to coincide with the black dot.

To load a trajectory, substitute the name **"trajectory+10.dat"** for another file (write the full path to the file). It needs to have the same format as the output of Mark's Point application.

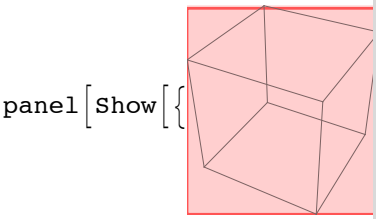
To change the mount models, see the section Adding and Using Mount Models.

```
manipulateTrajectory[
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}, (* mount model for TWF wrt ITF *)
 {0 &, 0 &, saggingZ, saggingTheta, 0 &, 0 &},
 (* mount model for HLF wrt TWF *)
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}, (* mount model for HUF wrt HLF *)
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}, (* mount model for CF wrt HUF *)
 {0 &, 0 &, 0 &, 0 &, 0 &, 0 &}, (* mount model for FPAF wrt CF *)
 "/Users/moreira/MyData/HETDEX/InHouse/math/MountModels/trajectory+10.dat"]
```

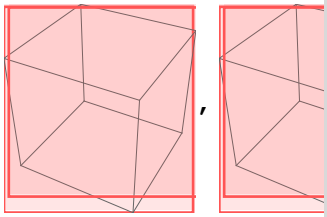
Trajectory points

Show bridge
Show control point
Show upper hexapod
Show corrector
Show focal plane
Show SIRP

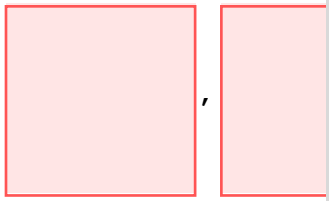
Show ITF (ideal tracker frame)
Show TWF (tracker working frame)
Show HLF (hexapod lower frame)
Show HUF (hexapod upper frame)
Show CF (corrector frame)
Show FPAF (focal plane assembly frame)



```
panel[Show[{
```



```
,
Show[{projITF[{1, traj$9
    traj$918[[1]][[4]]}, si
```



```
,
projFPA[{1, traj$918[[1]][
  traj$918[[1]][[4]]},
 sirpPoint$568, fpaftoit
Projection onto ITF XY,
Projection onto focal p
```