Analysis of Three Lens Optical System with CMOS sensor

by jph

The constants and functions compiled in this document apply to the camera payload of a nano-satellite built at NTU.

Curvature of the three lenses

The two opposing surfaces of each lens are spheres. The cross section of such a surface is a partial circle. The location of each circle is determined by a radius and midpoint.

The optical system we discuss here consists of 3 lenses. That means their boundaries are determined by 6 circles. The midpoints of the 6 circles are all on a line. In the following we state the 6 radii and midpoints (as distances on the z-axis):

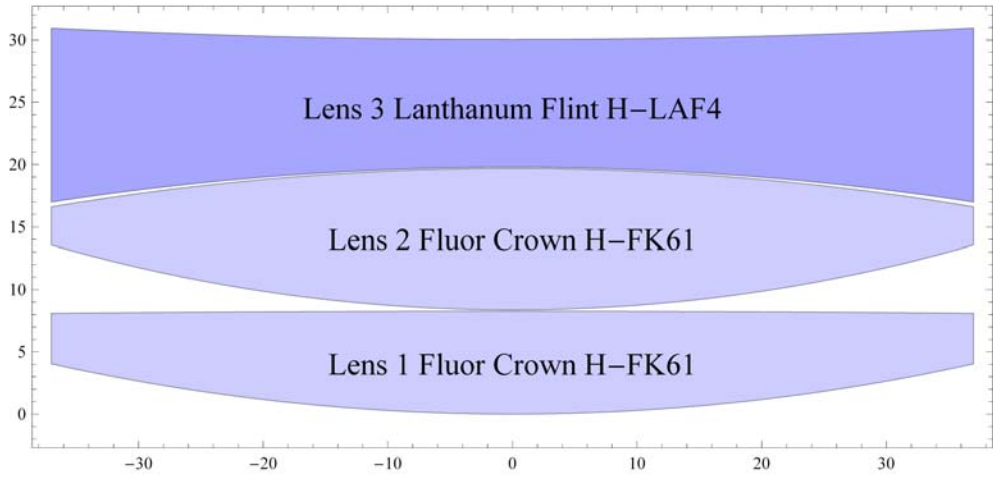
```
In[1]:= s1r = 171.545;
        s1z = 0 + s1r;
        s2r = 3900.000;
        s2z = 8.270 - s2r;
        s3r = 134.600;
        s3z = 8.270 + 0.1 + s3r;
        s4r = 223.475;
        s4z = 8.270 + 0.1 + 11.320 - s4r;
        s5r = 245.360;
        s5z = 8.270 + 0.1 + 11.320 + 0.1 - s5r;
        s6r = 760.000;
        s6z = 8.270 + 0.1 + 11.320 + 0.1 + 10.260 + s6r;
```

Visualization of the three lenses

The first two lenses are made from Fluor Crown H-FK61, the third lens is made from Lanthanum Flint H-LAF4. The crowns cause a dispersion of the different wavelengths. The flint counter acts this effect as much as possible.

```
In[13]:= cl1 = RGBColor[0.8, 0.8, 1];
         cl2 = RGBColor[0.65, 0.65, 1];
         re1 = RegionPlot[x^2 + (z - s1z)^2 < s1r^2 && x^2 + (z - s2z)^2 < s2r^2,
             {x, -37, 37}, {z, -2, 32}, PlotStyle → cl1];
         re2 = RegionPlot[x^2 + (z - s3z)^2 < s3r^2 && x^2 + (z - s4z)^2 < s4r^2,
             {x, -37, 37}, {z, -2, 32}, PlotStyle → cl1];
         re3 = RegionPlot[x^2 + (z - s5z)^2 > s5r^2 && x^2 + (z - s6z)^2 > s6r^2,
             {x, -37, 37}, {z, -2, 32}, PlotStyle → cl2];
         img = Show[{re1, re2, re3, Graphics[
              {Text[Style["Lens 1 Fluor Crown H-FK61", FontSize → 16], {0, 4}],
               Text[Style["Lens 2 Fluor Crown H-FK61", FontSize → 16], {0, 14}],
               Text[Style["Lens 3 Lanthanum Flint H-LAF4", FontSize → 16], {0, 24.5}]}
             ]}, {AspectRatio → Automatic, ImageSize → {256 × 2, 256}}]
```

Out[18]=

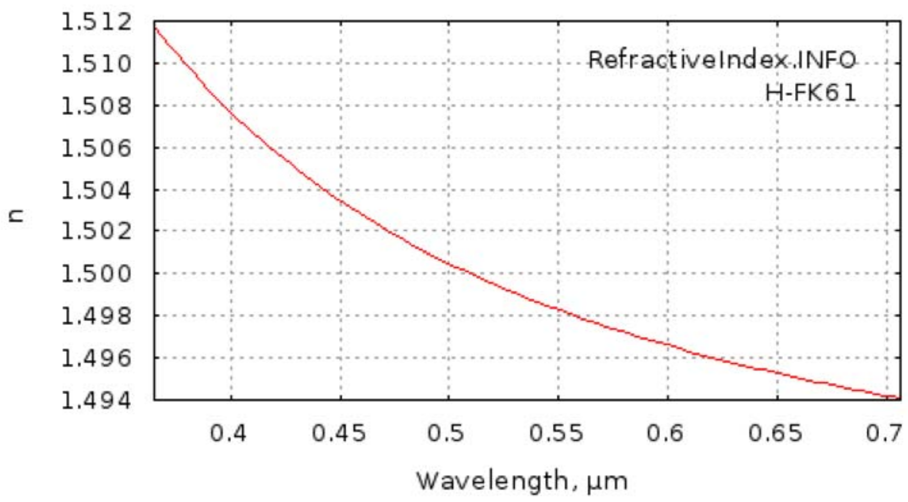

Lens 3 Lanthanum Flint H−LAF4
Lens 2 Fluor Crown H−FK61
Lens 1 Fluor Crown H−FK61
0
5
10
15
20
25
30
−30
−20
−10
0
10
20
30

The light enters the midpoint of the surface of the first lens at coordinate (0, 0).

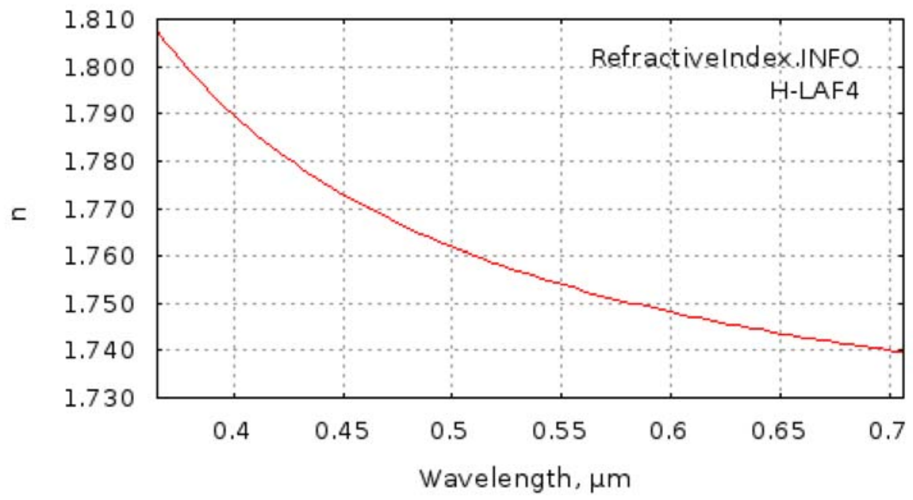
- **Refractivity of the material at different wavelengths**

The refractive index depends on the material and the wavelength of the light.

The refractive index for Fluor Crown H-FK61 is shown here:



The refractive index for Lanthanum Flint H-LAF4 is shown here:



From the plots we extract the refractive index for a selection of specific wavelengths:

Refractivity at *blue*, 0.460 μm:

```
In[19]:= rfkB = 1.50279;
         rlaB = 1.77047;
```

Refractivity at *green*, 0.500 μm

```
In[21]:= rfkG = 1.5005;
         rlaG = 1.76193;
```

Refractivity at *skin color*, 0.5876 μm

```
In[23]:= rfkS = 1.497;
         rlaS = 1.7495;
```

Refractivity at *red*, 0.640 μm

```
In[25]:= rfkR = 1.49553;
         rlaR = 1.7445;
```

The vector equation to refract a ray at a surface is according to Snell’s law

```
In[27]:= Refract[nrm_, dir_, rho_] := Module[{ct1 = nrm.dir, out},
           out = rho dir - (rho ct1 + √(1 - rho rho (1 - ct1 ct1))) nrm;
           out / Norm[out]]
```

- **Equations to trace a ray of light**

We solve the intersection of a circle with a ray as

```
In[28]:= crc = cx^2 + (cz - ch)^2 == cr^2;
         ray = {px, pz} + {dx, dz} v;
         sol = FullSimplify[Solve[crc /. {cx → ray⟦1⟧, cz → ray⟦2⟧}, {v}], {0 < cr}]
```

Out[30]=

$$\left\{\left\{v \to -\frac{-\text{ch}\,\text{dz} + \text{dx}\,\text{px} + \text{dz}\,\text{pz} + \sqrt{-\text{ch}^2\,\text{dx}^2 + \text{cr}^2\left(\text{dx}^2 + \text{dz}^2\right) - (\text{dz}\,\text{px} - \text{dx}\,\text{pz})^2 + 2\,\text{ch}\,\text{dx}\,(-\text{dz}\,\text{px} + \text{dx}\,\text{pz})}}{\text{dx}^2 + \text{dz}^2}\right\},\right.$$
$$\left.\left\{v \to \frac{\text{ch}\,\text{dz} - \text{dx}\,\text{px} - \text{dz}\,\text{pz} + \sqrt{-\text{ch}^2\,\text{dx}^2 + \text{cr}^2\left(\text{dx}^2 + \text{dz}^2\right) - (\text{dz}\,\text{px} - \text{dx}\,\text{pz})^2 + 2\,\text{ch}\,\text{dx}\,(-\text{dz}\,\text{px} + \text{dx}\,\text{pz})}}{\text{dx}^2 + \text{dz}^2}\right\}\right\}$$

The function “shoot” computes the refracted ray when a beam enters a medium with a different refractive index. In particular, the function applies to rays entering a lens as to exiting rays alike.

```
In[31]:= Shoot[{pos_, dir_}, rad_, hei_, rho_] := Module[{ndr, sub, tip, nrm, ct1, ct2, out, res},
           ndr = dir / Norm[dir];
           sub = {px → pos⟦1⟧, pz → pos⟦2⟧, dx → ndr⟦1⟧, dz → ndr⟦2⟧, cr → rad, ch → hei};
           tip = ray /. sub /. v → Min[Select[v /. sol /. sub, 0 < # &]];
           nrm = tip - {0, hei};
           nrm = nrm / Norm[nrm];
           nrm = If[0 < nrm.ndr, -nrm, nrm];
           {tip, Refract[nrm, ndr, rho]}]
```

The function “shootpath” traces a ray that enters the first lens all the way to the CMOS sensor and consists of 6 calls to the function “shoot”.

```
In[32]:= ShootPath[{pos_, dir_}, rho1_, rho2_] := Module[{t0c, t1c, t2c, t3c, t4c, t5c, t6c},
           t0c = {pos - 10 dir, dir};
           t1c = Shoot[t0c, s1r, s1z, 1 / rho1];
           t2c = Shoot[t1c, s2r, s2z, rho1];
           t3c = Shoot[t2c, s3r, s3z, 1 / rho1];
           t4c = Shoot[t3c, s4r, s4z, rho1];
           t5c = Shoot[t4c, s5r, s5z, 1 / rho2];
           t6c = Shoot[t5c, s6r, s6z, rho2];
           {t0c⟦1⟧, t1c⟦1⟧, t2c⟦1⟧, t3c⟦1⟧, t4c⟦1⟧, t5c⟦1⟧, t6c⟦1⟧, t6c⟦1⟧ + 210 t6c⟦2⟧}]
```

■ Location of CMOS sensor

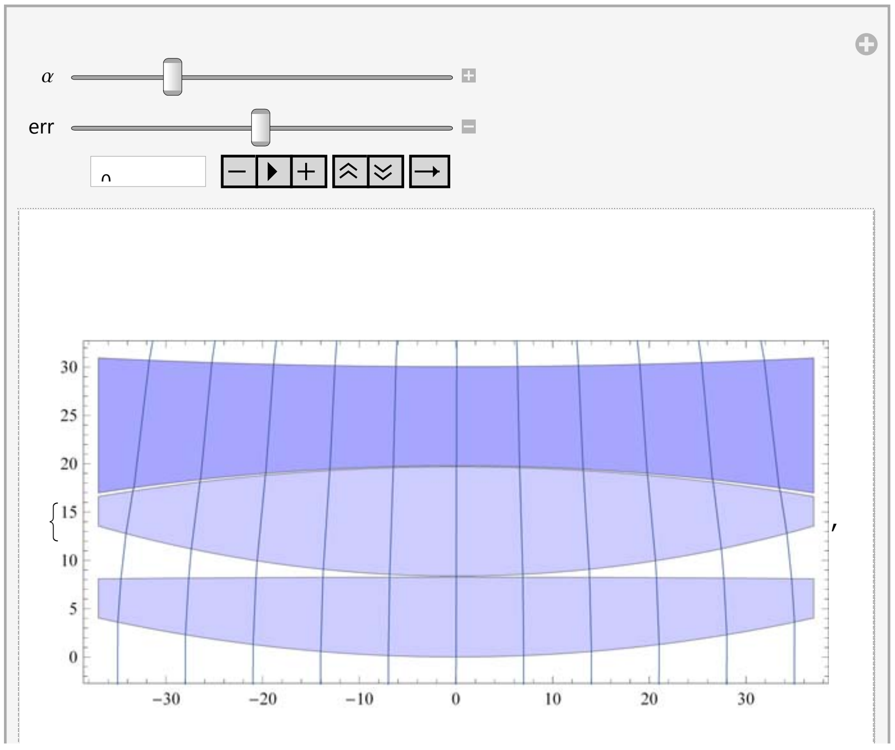
The function “pixelHit” establishes where a ray ends up on the CMOS sensor (in mm).

```
In[33]:= PixelHit[pth_, ahe_] := Module[{t, p1x, p1z, p2x, p2z, pix, loc, sub, eqs, lst},
           loc = {p1x, p1z} + t {p2x - p1x, p2z - p1z} - {pix, ahe};
           lst = Length[pth];
           sub = {p1x → pth⟦lst - 1, 1⟧, p1z → pth⟦lst - 1, 2⟧, p2x → pth⟦lst, 1⟧, p2z → pth⟦lst, 2⟧};
           eqs = # == 0 & /@ loc /. sub;
           pix /. Solve[eqs]⟦1⟧]
```

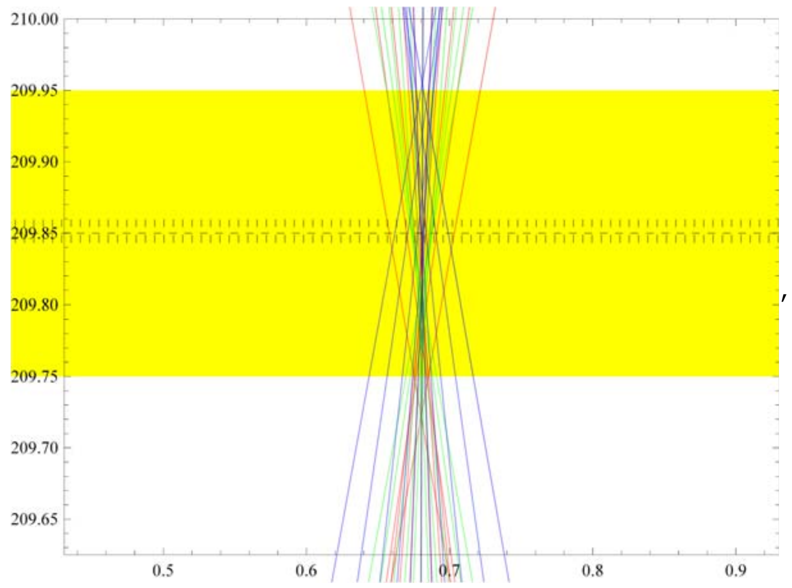
The distance between CMOS sensor array and the lens surface closest to the array is 179.7 mm. The distance between the midpoint of the surface of the first lens at coordinate (0, 0) to the surface of the CMOS sensor is 209.75.

- **Animation for different angles**

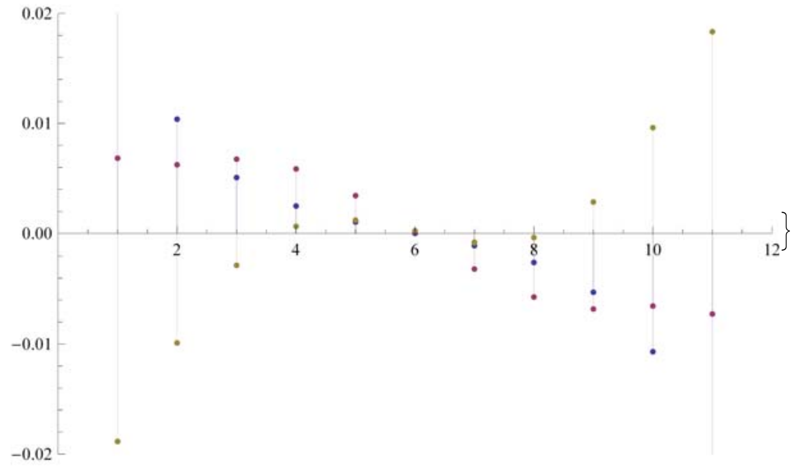
```
In[34]:= ColorR = RGBColor[1, 0, 0, 0.5];
         ColorG = RGBColor[0, 1, 0, 0.5];
         ColorB = RGBColor[0, 0, 1, 0.5];
         ani = Manipulate[
           Module[
            {ahg, ahe, lnsR, lnsG, lnsB, pixR, pixG, pixB, mid, cms = 3, tol = 0.15 + 0.1, pxw = 0.0065},
            ahg = s6z - s6r + 179.7 + err;
            ahe = ahg + 0.1;
            lnsR = Table[ShootPath[{{k, 0}, {Sin[α], Cos[α]}}, rfkR, rlaR], {k, -35, 35, 7}];
            lnsG = Table[ShootPath[{{k, 0}, {Sin[α], Cos[α]}}, rfkG, rlaG], {k, -35, 35, 7}];
            lnsB = Table[ShootPath[{{k, 0}, {Sin[α], Cos[α]}}, rfkB, rlaB], {k, -35, 35, 7}];
            pixR = PixelHit[#, ahe] & /@ lnsR;
            pixG = PixelHit[#, ahe] & /@ lnsG;
            pixB = PixelHit[#, ahe] & /@ lnsB;
            mid = pixR[[(Length[pixR] + 1) / 2]];
            {Show[{re1, re2, re3,
                Graphics[{ColorR, Line /@ lnsR, ColorG, Line /@ lnsG, ColorB, Line /@ lnsB}]},
              {AspectRatio → Automatic, ImageSize → {400, 300}}],
             Show[Graphics[{
                 Yellow, Rectangle[{-cms, ahg}, {cms, ahg + .2}],
                 ColorR, Line /@ lnsR, ColorG, Line /@ lnsG, ColorB, Line /@ lnsB,
                 Black, Dashed, Line[{{-cms, ahe}, {cms, ahe}}],
                 Table[Line[{{k pxw, ahe - pxw}, {k pxw, ahe + 2 pxw}}], {k, -768 / 2, 768 / 2}]
                }], {AspectRatio → Automatic, PlotRange → {{mid - tol, mid + tol},
                  {ahg - tol / 2, ahg + tol}}, Frame → True, Axes → False, ImageSize → {400, 300}}],
             ListPlot[{pixR, pixG, pixB} - mid, {PlotRange → {{0, Length[pixR] + 1}, {-0.02, 0.02}},
                Filling → Axis, ImageSize → {400, 300}}]
            }], {α, 0, 0.8 π / 180}, {err, -1, 1}]
```



Out[37]=



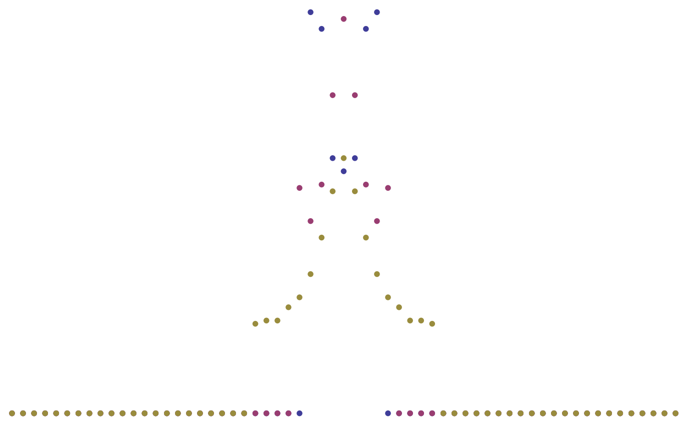
,



}

■ **Quantification of blur**

In[38]:=
```
err = -0.2;
pxw = 0.0065;
ahg = s6z - s6r + 179.7 + err;
ahe = ahg + 0.1;
stp = 1 / 10;
dir = {0, 1};
smp = {-30.5 pxw, 30.5 pxw, pxw};
lnsR = Table[ShootPath[{{k, 0}, dir}, rfkR, rlaR], {k, -35, 35, stp}];
lnsG = Table[ShootPath[{{k, 0}, dir}, rfkG, rlaG], {k, -35, 35, stp}];
lnsB = Table[ShootPath[{{k, 0}, dir}, rfkB, rlaB], {k, -35, 35, stp}];
pixR = PixelHit[#, ahe] & /@ lnsR;
pixG = PixelHit[#, ahe] & /@ lnsG;
pixB = PixelHit[#, ahe] & /@ lnsB;
pixR = PixelHit[#, ahe] & /@ lnsR;
binR = BinCounts[pixR, smp];
binG = BinCounts[pixG, smp];
binB = BinCounts[pixB, smp];
bina = {binR, binG, binB};
ListPlot[bina, {PlotRange → All, Axes → False}]
```

Out[56]=


■ **Details about the CMOS sensor**

The CMOS sensor has a resolution of 768 x 512. The satellite is estimated to be 650 km.

```
pixeldim = {768, 512};
pixelwid = 6.5 × 10^-6; (* unit: meter *)
lensdist = 0.2;
altitude = 650 000;
```

Thus, one pixels corresponds to 21 m on ground.

$$\text{gndwidth} = x \;/.\; \text{Solve}\left[\frac{\text{pixelwid}}{\text{lensdist}} == \frac{x}{\text{altitude}}\right][[1]]$$

```
21.125
```

The field of view in degree is 1.43 degree along the width of the array and 0.95 degree along the height of the array.

$$\frac{180}{\pi}\, 2\, \text{ArcTan}\left[\frac{1}{2}\, \text{pixeldim gndwidth} / \text{altitude}\right]$$

```
{1.43003, 0.95338}
```

```
xlen = 0.026;
lensdist = 0.317 - 0.153
```

```
0.164
```

lensdiam = $\sqrt{2\,(2\,\text{xlen})^2}$

```
0.0735391
```

```
fnumber = lensdist / lensdiam
```

```
2.23011
```

Optics on ground WRONG!

```
lensdiamG = 0.015;
lensdistG = 0.0015;
fnumberG = lensdistG / lensdiamG
```

```
0.1
```

human eye: f/8.3 brightly lit places
f/2.1 in dark places (or f/3.2)

Vacuum !!! different focal length!

```
pixelwid pixeldim 1000
```

```
{4.992, 3.328}
```