Find the field of view imaged by the Macro Camera Tripod with the Canon EOS 5D Mark IV camera.

P. McGill

27Aug19 - Initial release
17Apr20 - Corrected direction vector calculations
20Apr20 - Added calculations for Barry camera

First define some constants, such as the camera height in meters and tilt in radians, the size of the image sensor in mm, and the lens focal length in mm.

```
In[•]:= camTilt = 45 Degree;
camHeight = 0.75;
hSensor = 36;
vSensor = 24;
lensFL = 50;
```

Now we can find the field of view in water. For refractive index of seawater, assume 2 degC, 4000 m depth, 30 ppt salinity.

```
In[•]:= seawaterRI = 1.3375 + (400 × 0.000016) + 0.00577
Out[•]= 1.34967

In[•]:= airRI = 1.000293;
```

Now we can find the nominal field of view in water, not accounting for pincushion distortion from the flat-plate viewport.

```
In[•]:= hFOV = 2 ArcTan[hSensor/2 / lensFL] airRI / seawaterRI;

In[•]:= hFOV / Degree // N
Out[•]= 29.3474

In[•]:= vFOV = 2 ArcTan[vSensor/2 / lensFL] airRI / seawaterRI;

In[•]:= vFOV / Degree // N
Out[•]= 20.0044
```

Repeat the calculations for a 100 mm lens.

```
In[•]:= hFOV100 = 2 ArcTan[hSensor/2 / 100] airRI / seawaterRI;
```

```
In[•]:= hFOV100 / Degree // N
Out[•]= 15.1251

In[•]:= vFOV100 = 2 ArcTan[(vSensor / 2) / 100] airRI / seawaterRI;

In[•]:= vFOV100 / Degree // N
Out[•]= 10.1429
```

Define a function to find the intersection of a line and a plane (http://geomalgorithms.com/a05-_intersect-1.html). The line is given as any two points on the line, and the plane is defined by any three points in the plane. Returns a point in the plane where the line intersects.

```
In[•]:= linePlaneIntersection[{lp1_, lp2_}, {pp1_, pp2_, pp3_}] := Module[
  {norm, u, w, d, n, sI},
  norm = Cross[(pp1 - pp2), (pp3 - pp2)];
  (* find normal to plane from three points *)
  u = lp2 - lp1; (* find direction vector of line *)
  w = lp1 - pp1; (* find offset vector of line *)
  d = norm.u;
  n = -(norm.w);
  sI = n / d;
  Return[lp1 + sI * u] (* return point of intersection *)
 ] (* end module*)
```

Define the location of the camera viewport above the origin.

```
In[•]:= c = {0, 0, camHeight};
```

Define the seafloor with three points that lie upon it.

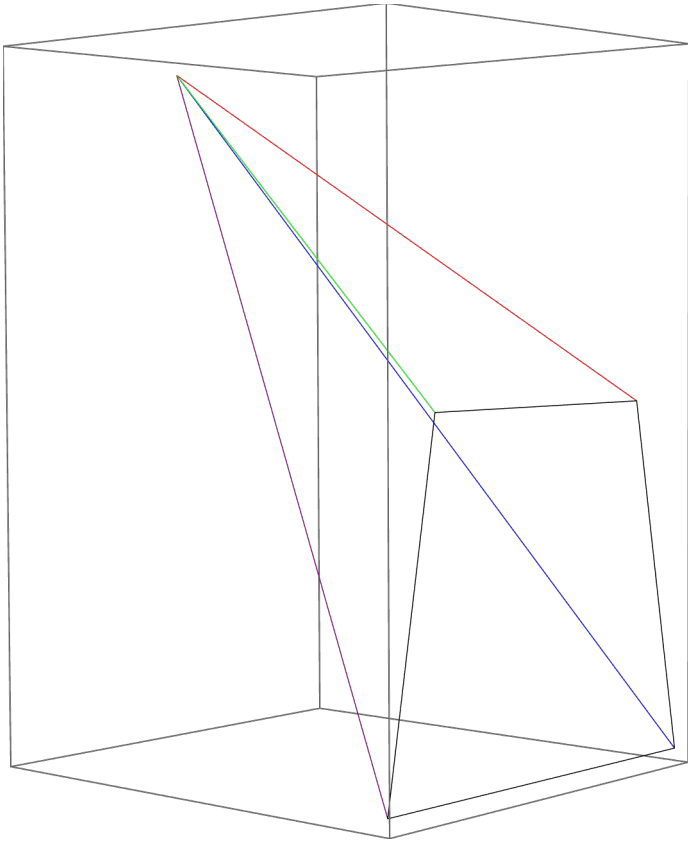
```
In[•]:= seafloor = {{-1, 0, 0}, {1, 0, 0}, {0, 1, 0}};
```

Define four direction vectors (i.e. lines), one for each corner of the field of view.

```
In[•]:= lul = {c,
    {-Tan[hFOV / 2] / Cos[camTilt - vFOV / 2], 1, camHeight - (Tan[camTilt - vFOV / 2])}};
lur =
  {c, {Tan[hFOV / 2] / Cos[camTilt - vFOV / 2], 1, camHeight - (Tan[camTilt - vFOV / 2])}};
lll = {c,
    {-Tan[hFOV / 2] / Cos[camTilt + vFOV / 2], 1, camHeight - (Tan[camTilt + vFOV / 2])}};
llr =
  {c, {Tan[hFOV / 2] / Cos[camTilt + vFOV / 2], 1, camHeight - (Tan[camTilt + vFOV / 2])}};
```

```
In[•]:= Graphics3D[{Red, Line[lul], Green, Line[lur], Blue, Line[lll], Purple, Line[llr],
          Black, Line[{Last[lul], Last[lur], Last[llr], Last[lll], Last[lul]}]}]
```

Out[•]=

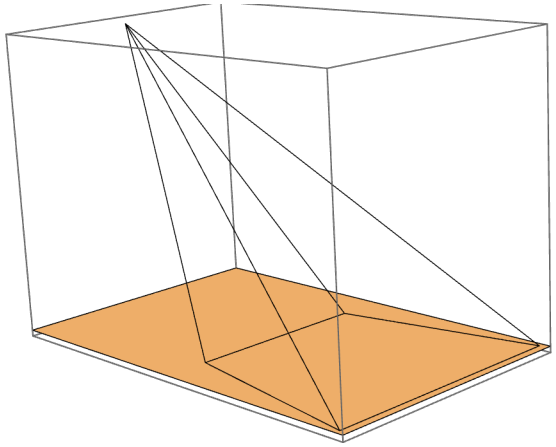


Find the four intersection points where these field of view lines intersect the seafloor.

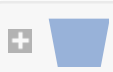
```
In[•]:= iul = linePlaneIntersection[lul, seafloor]
Out[•]= {-0.342416, 1.0712, 0.}

In[•]:= iur = linePlaneIntersection[lur, seafloor]
Out[•]= {0.342416, 1.0712, 0.}

In[•]:= ill = linePlaneIntersection[lll, seafloor]
Out[•]= {-0.239743, 0.525112, 0.}

In[•]:= ilr = linePlaneIntersection[llr, seafloor]
Out[•]= {0.239743, 0.525112, 0.}
```

In[•]:=
```
Graphics3D[{InfinitePlane[seafloor], Line[{c, iul}], Line[{c, iur}],
  Line[{c, ill}], Line[{c, ilr}], Line[{iul, iur, ilr, ill, iul}]}]
```

Out[•]=

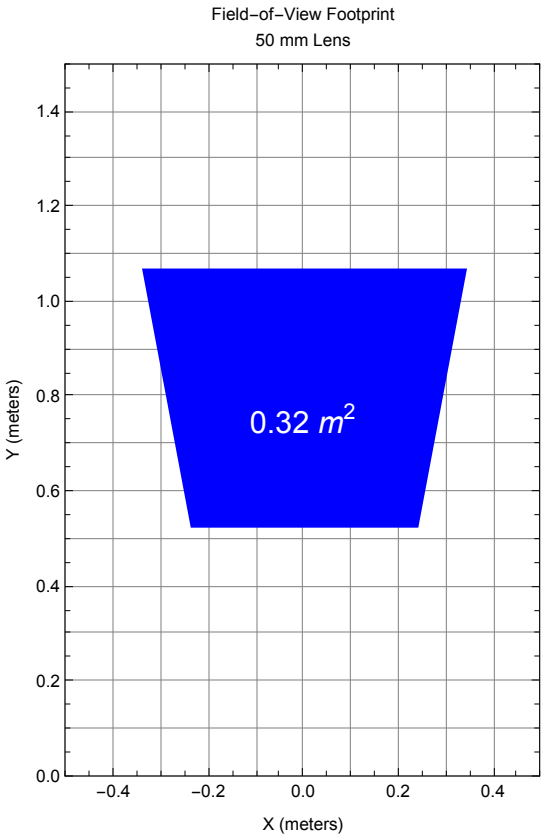


In[•]:=
```
footprint = Polygon[Take[#, 2] & /@ {iul, iur, ilr, ill}]
```

Out[•]=
```
Polygon[ 
Number of points: 4
Embedding dimension: 2 ]
```

In[ ]:=
```
footPlot = Graphics[{Blue, footprint},
  GridLines → {Range[-0.5, 0.5, 0.1], Range[0, 1.5, 0.1]}, AspectRatio → Automatic,
  PlotRange → {{-0.5, 0.5}, {0, 1.5}}, Frame → True, FrameLabel → {{"Y (meters)", ""},
    {"X (meters)", "Field-of-View Footprint\r50 mm Lens"}}, Epilog → Inset[
    Style[ToString[NumberForm[Area@footprint, 2]] <> " m²", White, 16], {0, 0.75}]]
```

Out[ ]=



In[ ]:=
```
Area[footprint]
```

Out[ ]=
```
0.317909
```

Was 0.228265 m^2 before corrected direction vectors were calculated.

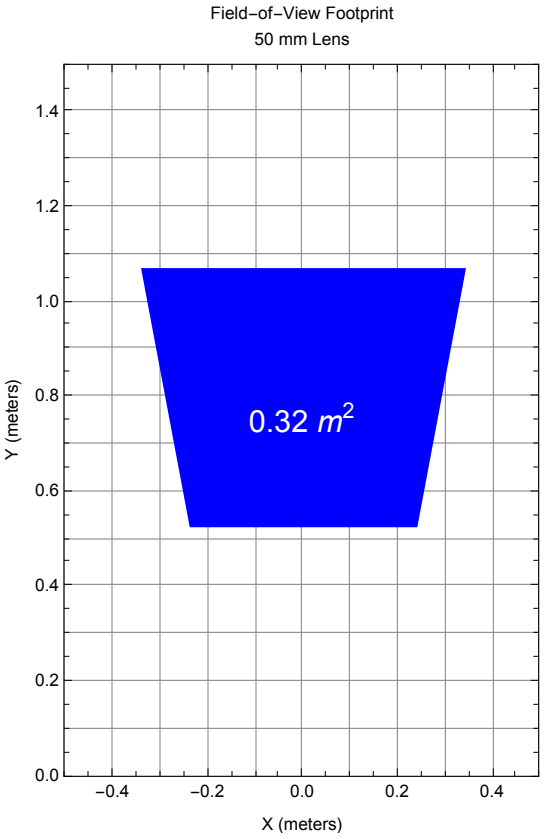
Write a function to plot the footprint as a function of the lens focal length, with all other parameters as defined above.

In[ ]:=
```
footPrintPlot[lensFL_] :=
 Module[{hFOV, vFOV, lul, lur, lll, llr, iul, iur, ill, ilr, footprint},
  hFOV = 2 ArcTan[(hSensor/2)/lensFL] airRI/seawaterRI;
  vFOV = 2 ArcTan[(vSensor/2)/lensFL] airRI/seawaterRI;
  lul = {c,
    {-Tan[hFOV / 2] / Cos[camTilt - vFOV / 2], 1, camHeight - (Tan[camTilt - vFOV / 2])}};
  lur = {c,
    {Tan[hFOV / 2] / Cos[camTilt - vFOV / 2], 1, camHeight - (Tan[camTilt - vFOV / 2])}};
  lll = {c,
    {-Tan[hFOV / 2] / Cos[camTilt + vFOV / 2], 1, camHeight - (Tan[camTilt + vFOV / 2])}};
  llr =
   {c, {Tan[hFOV / 2] / Cos[camTilt + vFOV / 2], 1, camHeight - (Tan[camTilt + vFOV / 2])}};
  iul = linePlaneIntersection[lul, seafloor];
  iur = linePlaneIntersection[lur, seafloor];
  ill = linePlaneIntersection[lll, seafloor];
  ilr = linePlaneIntersection[llr, seafloor];
  footprint = Polygon[Take[#, 2] & /@ {iul, iur, ilr, ill}];
  Graphics[{Blue, footprint},
   GridLines → {Range[-0.5, 0.5, 0.1], Range[0, 1.5, 0.1]},
   AspectRatio → Automatic, PlotRange → {{-0.5, 0.5}, {0, 1.5}},
   Frame → True, FrameLabel → {{"Y (meters)", ""},
     {"X (meters)", "Field-of-View Footprint\r" <> ToString[lensFL] <> " mm Lens"}},
   Epilog → Inset[Style[ToString[NumberForm[Area@footprint, 2]] <> " m^2",
      White, 16], {0, 0.75}]]
 ] (* end Module *)
```

In[ ]:=

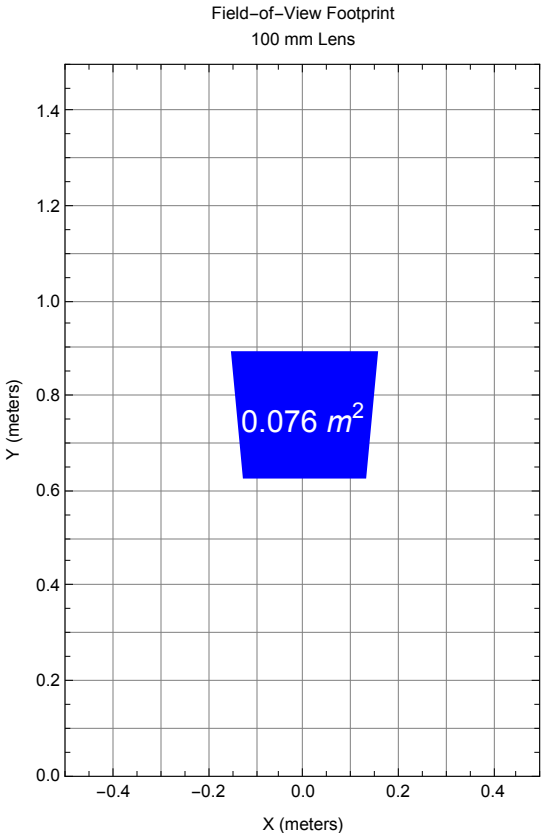
In[ ]:= **footPrintPlot[50]**

Out[ ]=



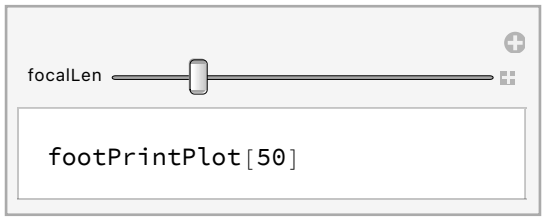
In[ ]:=
```
footPrintPlot[100]
```

Out[ ]=



In[ ]:=
```
Manipulate[footPrintPlot[focalLen], {focalLen, 25, 150, 5}]
```

Out[ ]=



Coral Observatory Macro Camera

The Coral Observatory Macro Camera won't be aimed at the seafloor, but instead be aimed horizontally at a vertical rock surface covered with coral. We need to find the size of the imaged area for various target distances.

Define a function that returns the height of the field of view given the lens focal length (in mm) and target distance (in m).

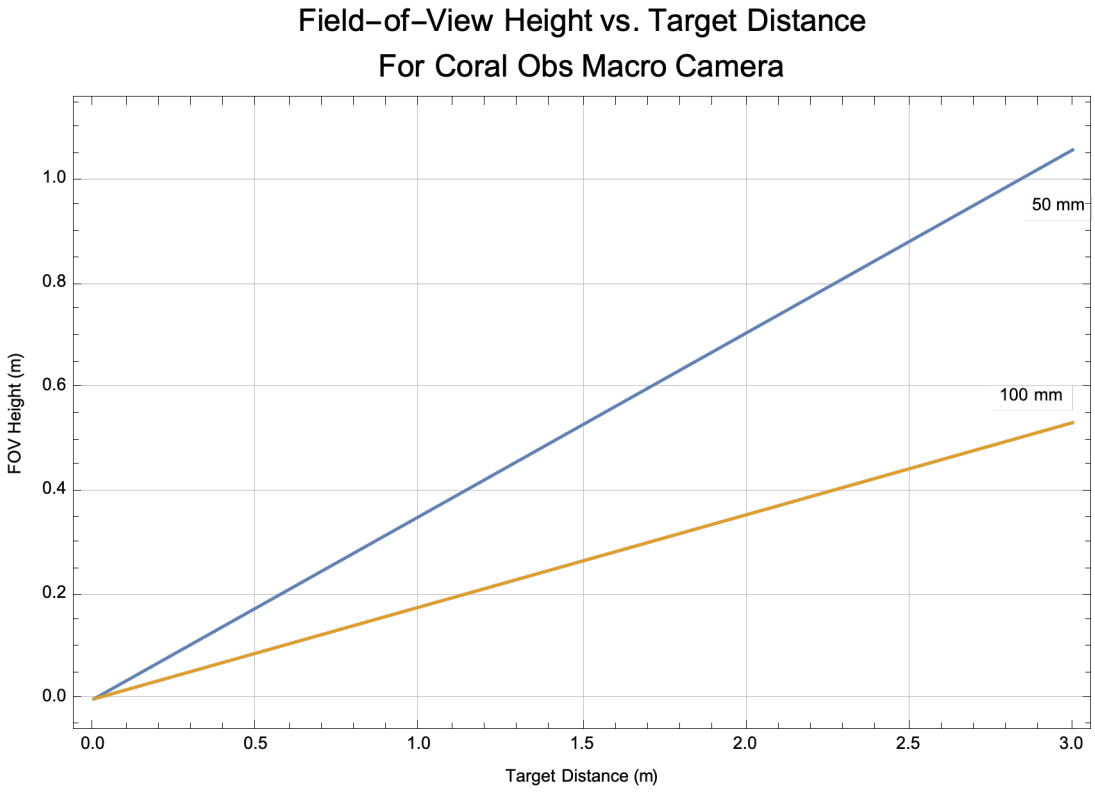
```
In[•]:= height[lensFL_, d_] := Module[{vFOV},
          vFOV = 2 ArcTan[(vSensor/2)/lensFL] airRI/seawaterRI;
          2 Tan[vFOV / 2] d
         ]
```

```
In[•]:= height[100, 1.41]
Out[•]= 0.250262
```

```
In[•]:= height[100, 1.5]
Out[•]= 0.266236
```

```
In[•]:= Plot[{height[50, d], height[100, d]}, {d, 0, 3}, GridLines → Automatic,
         Frame → True, FrameLabel → {{"FOV Height (m)", ""}, {"Target Distance (m)",
            Style["Field-of-View Height vs. Target Distance\rFor Coral Obs Macro Camera",
             16]}}, PlotLabels → Placed[{"50 mm", "100 mm"}, {Scaled[1], Right}]]
```

Out[•]=




```
In[•]:= width[lensFL_, d_] := Module[{hFOV},
          hFOV = 2 ArcTan[(hSensor/2)/lensFL] airRI/seawaterRI;
          2 Tan[hFOV / 2] d
         ]
```

In[•]:=
```
width[100, 1.41]
```

Out[•]=
```
0.374393
```

In[•]:=
```
width[100, 1.5]
```

Out[•]=
```
0.398291
```

In[•]:=
```
Plot[{width[50, d], width[100, d]}, {d, 0, 3}, GridLines → Automatic, Frame → True,
 FrameLabel → {{"FOV Width (m)", ""}, {"Target Distance (m)", Style[
     "Field-of-View Width vs. Target Distance\rFor Coral Obs Macro Camera", 16]}},
 PlotLabels → Placed[{"50 mm", "100 mm"}, {Scaled[1], Right}]]
```

Out[•]=