
Sargasso Camera Tripod Photogrammetric Calibration

P. McGill

v1.0.0

Introduction

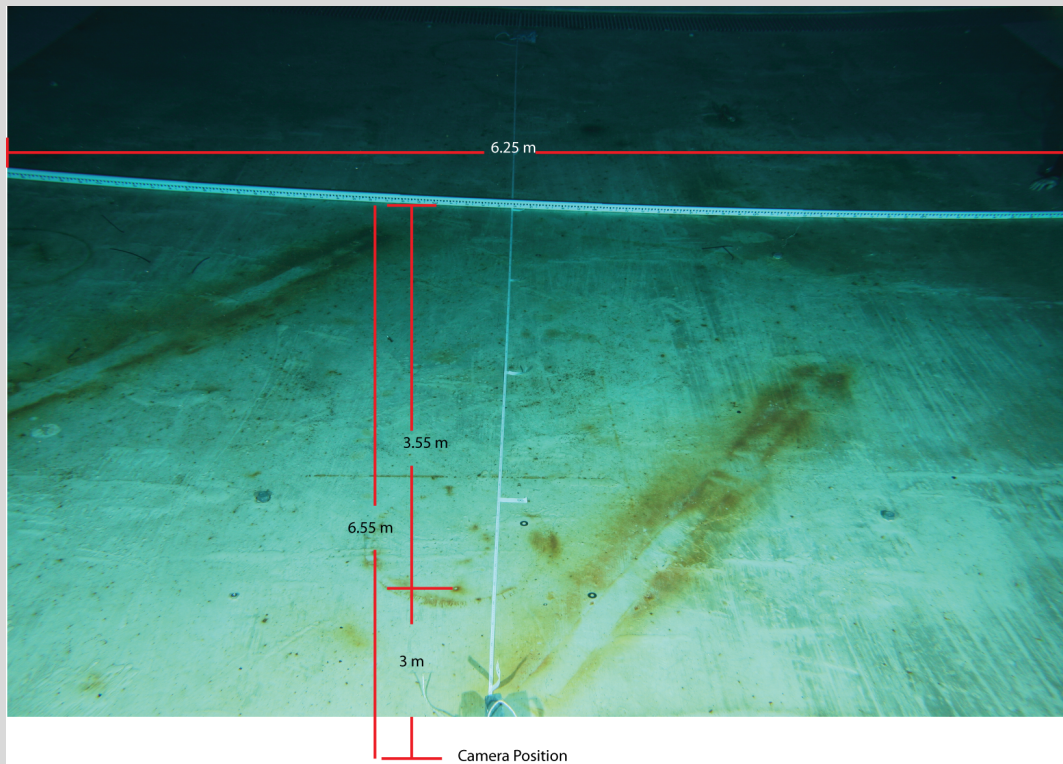
The Camera Tripod takes hourly images of the seafloor at abyssal depths. This notebook develops a set of routines that converts pixel locations in the Camera Tripod images into locations and distances in the plane of the seafloor. These routines assume that the seafloor is flat and do not account for barrel distortion in the camera lens.

Viewing Calibration Images

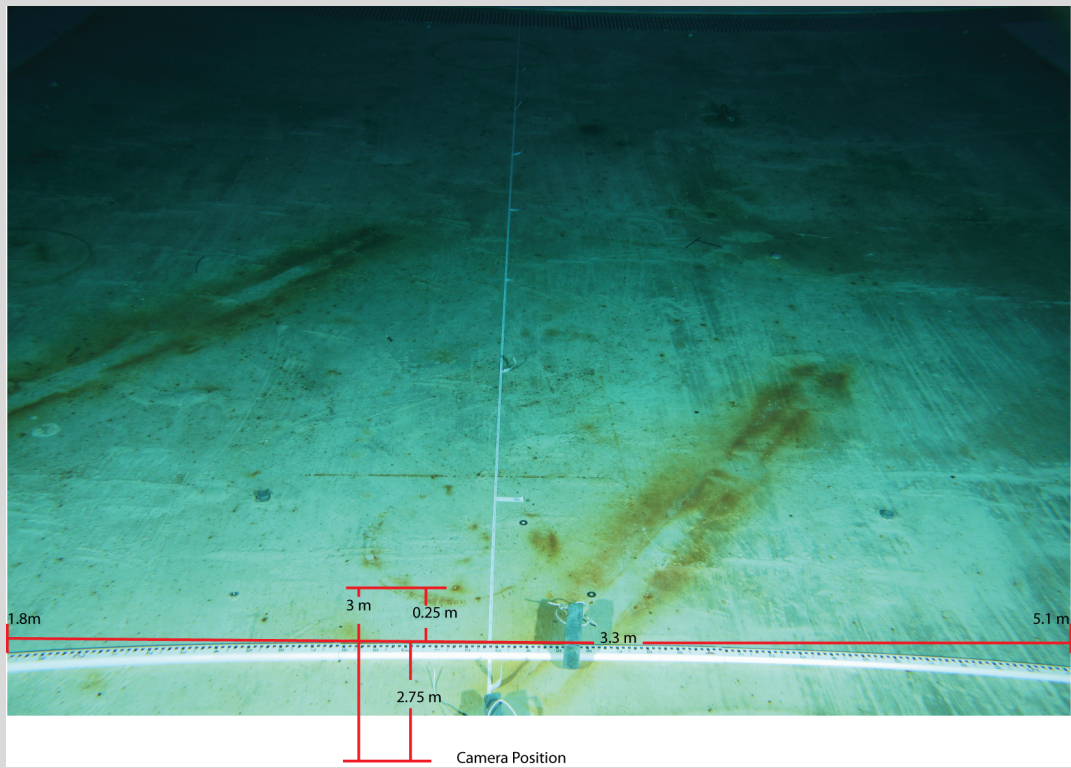
We have three images taken in the Test Tank with scales. We also have annotated versions of those images showing the distances.

First import the annotated images.

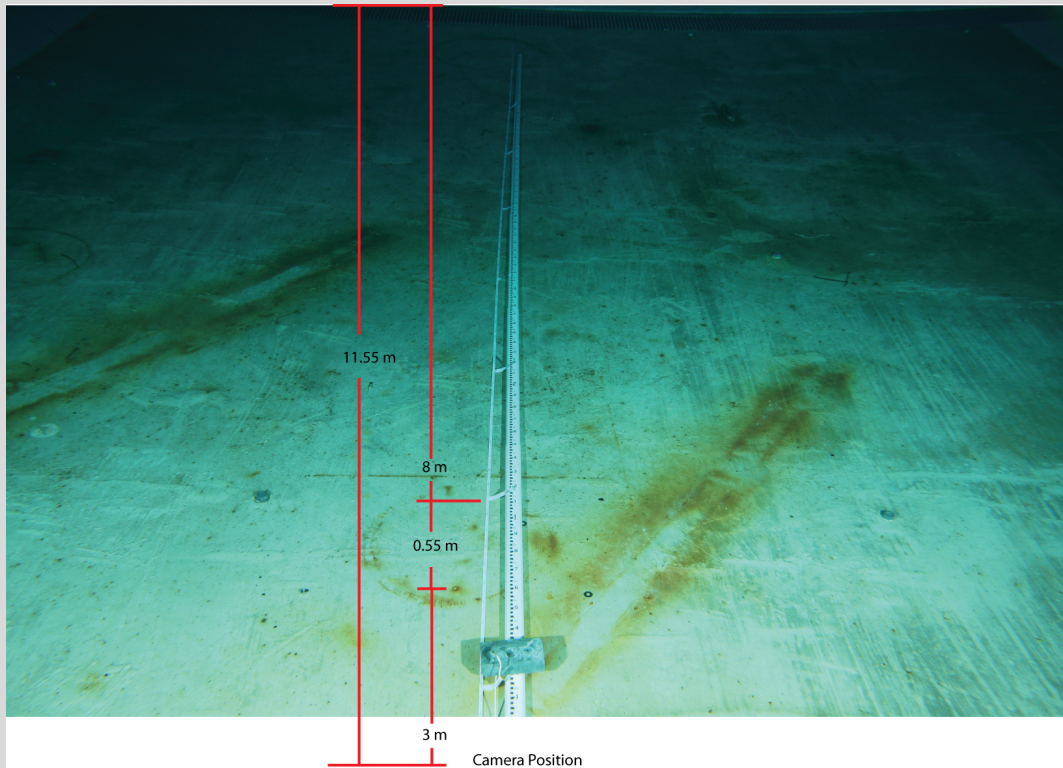

```
anno1 = Import["/Volumes/ProjectLibrary/900622.PelagicBenthic/CameraTripod/  
TankTestSargassoTripod/2013_02_12/AnnotatedImage1.png"]
```



```
anno2 = Import["/Volumes/ProjectLibrary/900622.PelagicBenthic/CameraTripod/  
TankTestSargassoTripod/2013_02_12/AnnotatedImage2.png"]
```



```
anno3 = Import["/Volumes/ProjectLibrary/900622.PelagicBenthic/CameraTripod/
TankTestSargassoTripod/2013_02_12/AnnotatedImage3.png"]
```



Import the original images, which are much higher resolution.

```
orig1 = Import["/Volumes/ProjectLibrary/900622.PelagicBenthic/CameraTripod/
TankTestSargassoTripod/2013_02_12/Pulse_20130212_00034.JPG"];

```

```
orig2 = Import["/Volumes/ProjectLibrary/900622.PelagicBenthic/CameraTripod/
TankTestSargassoTripod/2013_02_12/Pulse_20130212_00031.JPG"];

```

```
orig3 = Import["/Volumes/ProjectLibrary/900622.PelagicBenthic/CameraTripod/
TankTestSargassoTripod/2013_02_12/Pulse_20130212_00028.JPG"];

```

```
imageSize = ImageDimensions[orig1]
```

```
{4368, 2912}
```

```
hRes = imageSize[[1]]; vRes = imageSize[[2]];

```

Development of Camera Geometry Equations

We need to define the geometric characteristics of the camera and the tripod that holds it. These include the angular tilt and the height of the camera, the size of the Canon EOS 5D camera image sensor in millimeters, and the focal length of the lens in millimeters. Camera image sensor specifications are taken from <http://photo.net/equipment/canon/5D/specs>.

```
camTilt = 31 Degree;
```

```
camHeight = 2.6;
```

Size of image sensor and focal length of lens, all in mm.

```
hSensor = 35.8;
```

```
vSensor = 23.9;
```

```
lensFL = 28;
```

For refractive index of seawater, assume 2 degC, 4000 m depth, 30 ppt salinity.

```
seawaterRI = 1.3375 + (400 × 0.000016) + 0.00577
```

```
1.34967
```

```
airRI = 1.000293;
```

Now we can find the calculated field of view in water.

$$hFOV = 2 \operatorname{ArcTan} \left[\frac{\frac{hSensor}{2}}{lensFL} \right] \frac{airRI}{seawaterRI};$$

```
hFOV  
Degree // N
```

```
48.3077
```

$$vFOV = 2 \operatorname{ArcTan} \left[\frac{\frac{vSensor}{2}}{lensFL} \right] \frac{airRI}{seawaterRI};$$

```
vFOV  
Degree // N
```

```
34.2586
```

Find the horizontal angle from the point of interest in the image to the vertical centerline of the image. Left of the centerline is negative and right is positive. The pixel locations are defined with {0,0} at the bottom left corner of the image.

$$\text{hAngle[hPixLoc_]} := \text{ArcTan}\left[\frac{\left(\text{hPixLoc} - \frac{\text{hRes}}{2}\right) \text{Tan}\left[\frac{\text{hFOV}}{2}\right]}{\frac{\text{hRes}}{2}}\right]$$

$$\text{hAngle}\{0, \text{hRes} / 2, \text{hRes}\} / \text{Degree} // \text{N}$$

$$\{-24.1539, 0., 24.1539\}$$

Find the vertical angle from the point of interest to the horizontal centerline of the image. Below the centerline is negative and above is positive.

$$\text{vAngle[vPixLoc_]} := \text{ArcTan}\left[\frac{\left(\text{vPixLoc} - \frac{\text{vRes}}{2}\right) \text{Tan}\left[\frac{\text{vFOV}}{2}\right]}{\frac{\text{vRes}}{2}}\right]$$

$$\text{vAngle}\{0, \text{vRes} / 2, \text{vRes}\} / \text{Degree} // \text{N}$$

$$\{-17.1293, 0., 17.1293\}$$

Calculate how much the point of interest is offset in meters in the horizontal direction from the vertical centerline of the image. The measurement is in the plane of the seafloor; left of the centerline is negative, and right is positive. Note that this depends on both the x and y locations in the image, since a given horizontal pixel offset in the image represents a smaller distance for points closer to the camera.

$$\text{hPlanarOffset}\{\text{hPixLoc_}, \text{vPixLoc_}\} := \text{Tan}[\text{hAngle}[\text{hPixLoc}]] \frac{\text{camHeight}}{\text{Tan}[\text{camTilt} - \text{vAngle}[\text{vPixLoc}]]}$$

Points in the centerline of the image at the bottom, middle, and top of the image should have a horizontal offset of zero.

$$\text{hPlanarOffset}\{ \# \} \& / @ \{ \{ \text{hRes} / 2, 0 \}, \{ \text{hRes} / 2, 2912 / 2 \}, \{ \text{hRes} / 2, 2912 \} \}$$

$$\{0., 0., 0.\}$$

Find the width of the image at the bottom, middle, and top of the image.

$$2 \text{ hPlanarOffset}\{ \# \} \& / @ \{ \{ \text{hRes}, 0 \}, \{ \text{hRes}, \text{vRes} / 2 \}, \{ \text{hRes}, \text{vRes} \} \}$$

$$\{2.09018, 3.881, 9.44363\}$$

Calculate the distance between the point of interest and a horizontal line on the seafloor lying directly beneath the camera and orthogonal to the view direction. This distance depends only on the vertical pixel location in the image.

```
vPlanarDist[vPixLoc_] :=
  
$$\frac{\text{camHeight}}{\tan[\text{camTilt} - \text{vAngle}[\text{vPixLoc}]]}$$

```

Find the planar distance to a point on the centerline at the bottom, middle, and top of the image.

```
vPlanarDist[#] & /@ {0, vRes / 2, vRes}
{2.33045, 4.32713, 10.5292}
```

Define a function that calculates the planar coordinates of any pixel in the image. The coordinates are relative to a point on the seafloor lying directly beneath the camera, called the nadir point. The horizontal coordinate is left (negative) or right (positive) of the centerline, and the vertical coordinate is always a positive distance away from the camera.

```
planarCoords[{hPixLoc_, vPixLoc_}] :=
  {hPlanarOffset[{hPixLoc, vPixLoc}], vPlanarDist[vPixLoc]}
```

Find the planar coordinates of the point in the middle of the image.

```
planarCoords[{hRes / 2, vRes / 2}]
{0., 4.32713}
```

Define a function that calculates the distance in the plane of the seafloor between any two points in the image.

```
planarDist[{{hPixLoc1_, vPixLoc1_}, {hPixLoc2_, vPixLoc2_}}] :=
  Module[{x1, y1, x2, y2},
    {x1, y1} = planarCoords[{hPixLoc1, vPixLoc1}];
    {x2, y2} = planarCoords[{hPixLoc2, vPixLoc2}];
    Sqrt[(Max[{x1, x2}] - Min[{x1, x2}])^2 + (Max[{y1, y2}] - Min[{y1, y2}])^2]
  ]
```

Find the distance between the bottom left corner of the image and the bottom right corner. This should match the width of the bottom of the image as calculated above.

```
planarDist[{{0, 0}, {hRes, 0}}]
2.09018
```

Analysis of Uncorrected Test Images

See if the two test images measuring the horizontal field of view are consistent, and compare them to the calculated value of 48.3 deg.

First we must find the direct linear, not planar, distance between the camera and the scale.

```
farScaleDist = Norm@{6.55, camHeight} // N
```

```
7.04716
```

```
nearScaleDist = Norm@{2.72, camHeight} // N
```

```
3.76276
```

Knowing the distance to the scales and how much of them can be seen, we can find the field of view.

$$\text{farHFOV} = 2 \frac{\text{ArcTan}\left[\frac{\left(\frac{6.28}{2}\right)}{\text{farScaleDist}}\right]}{\text{Degree}}$$

```
48.0326
```

$$\text{nearHFOV} = 2 \frac{\text{ArcTan}\left[\frac{\left(\frac{3.31}{2}\right)}{\text{nearScaleDist}}\right]}{\text{Degree}}$$

```
47.4833
```

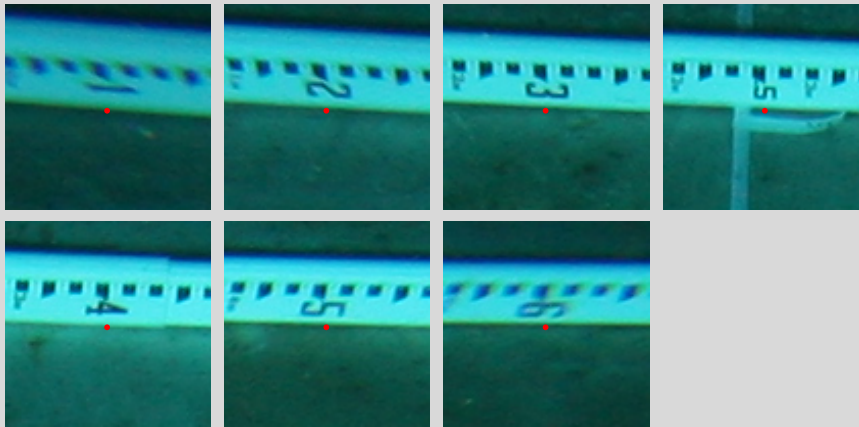
The two measurements are reasonable and consistent.

Define a function that, given an image and a point on that image, will show a zoomed view of the region around the chosen point. This will aid in picking calibration points more accurately. The calibration points are selected by using a *Mathematica* feature which allows you to read the pixel locations of selected points in an image by right-clicking on the image and selecting “Get Coordinates”. Of course this feature won’t work in the PDF version of this document.

```
zoomPick[image_Image, span_, {cph_, cpv_}] := Show[
  ImageTake[
    image, {vRes - (cpv + span), vRes - (cpv - span)}, {cph - span, cph + span}],
  Graphics[
    {Red, Point[{span, span}]}
  ]
]
```



```
GraphicsGrid[
  {{zoomPick[orig1, 50, cpf1 = {374, 2177}],
    zoomPick[orig1, 50, cpf2 = {1086, 2131}],
    zoomPick[orig1, 50, cpf3 = {1755, 2097}],
    zoomPick[orig1, 50, cpf3p5 = {2084, 2083}]}},
  {zoomPick[orig1, 50, cpf4 = {2411, 2070}],
    zoomPick[orig1, 50, cpf5 = {3073, 2050}],
    zoomPick[orig1, 50, cpf6 = {3760, 2040}]}}
]
```




```
Show[orig1, Graphics[{Red, Point[cpfh1], Point[cpfh2], Point[cpfh3],  
Point[cpfh3p5], Point[cpfh4], Point[cpfh5], Point[cpfh6]}], ImageSize → Large]
```



Calculate the distance from the nadir point of the camera to the calibration point nearest the vertical tape, and compare it to the measured distance of 6.55 m.

```
vPlanarDist[cpfh3p5[[2]]]
```

```
5.9968
```

Calculate the width of the image at the calibration point, and compare it to the measured width of 6.25 m.

```
2 hPlanarOffset[{hRes, cpfh3p5[[2]]}]
```

```
5.37853
```

Calculate the distance between adjacent calibration points, known to be exactly 1.00 m apart.

```
pdfhd12 = planarDist[{cpfh1, cpfh2}]
```

```
0.978127
```

```
pdfhd23 = planarDist[{cpfh2, cpfh3}]
```

```
0.865634
```

```
pdfhd34 = planarDist[{cpfh3, cpfh4}]
```

```
0.815193
```

```
pdfhd45 = planarDist[{cpfh4, cpfh5}]
```

```
0.799734
```

```
pdfhd56 = planarDist[{cpfh5, cpfh6}]
```

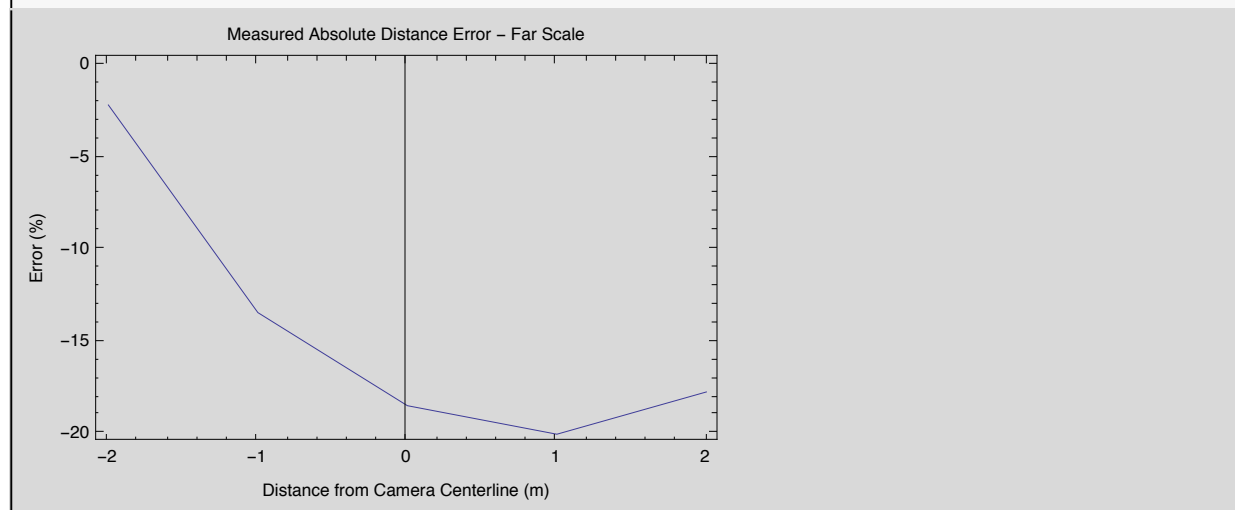
```
0.822637
```

Plot the error as a function of distance from the centerline of the camera view.

```
farCalData = {{-2, (pdfhd12 - 1) 100}, {-1, (pdfhd23 - 1) 100},  
             {0, (pdfhd34 - 1) 100}, {1, (pdfhd45 - 1) 100}, {2, (pdfhd56 - 1) 100}}
```

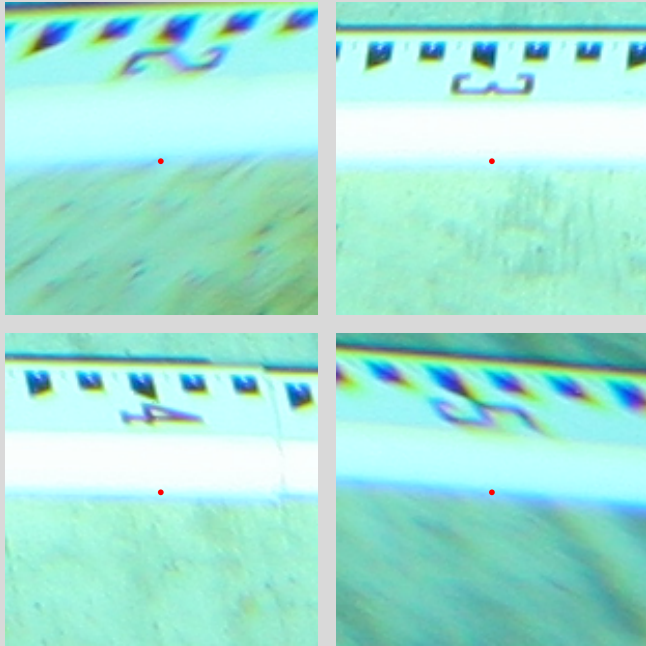
```
{{-2, -2.18726}, {-1, -13.4366}, {0, -18.4807}, {1, -20.0266}, {2, -17.7363}}
```

```
ListPlot[farCalData, {Frame → True, Joined → True,  
                    FrameLabel → {"Distance from Camera Centerline (m)",  
                                   "Error (%)", "Measured Absolute Distance Error - Far Scale"}}]
```

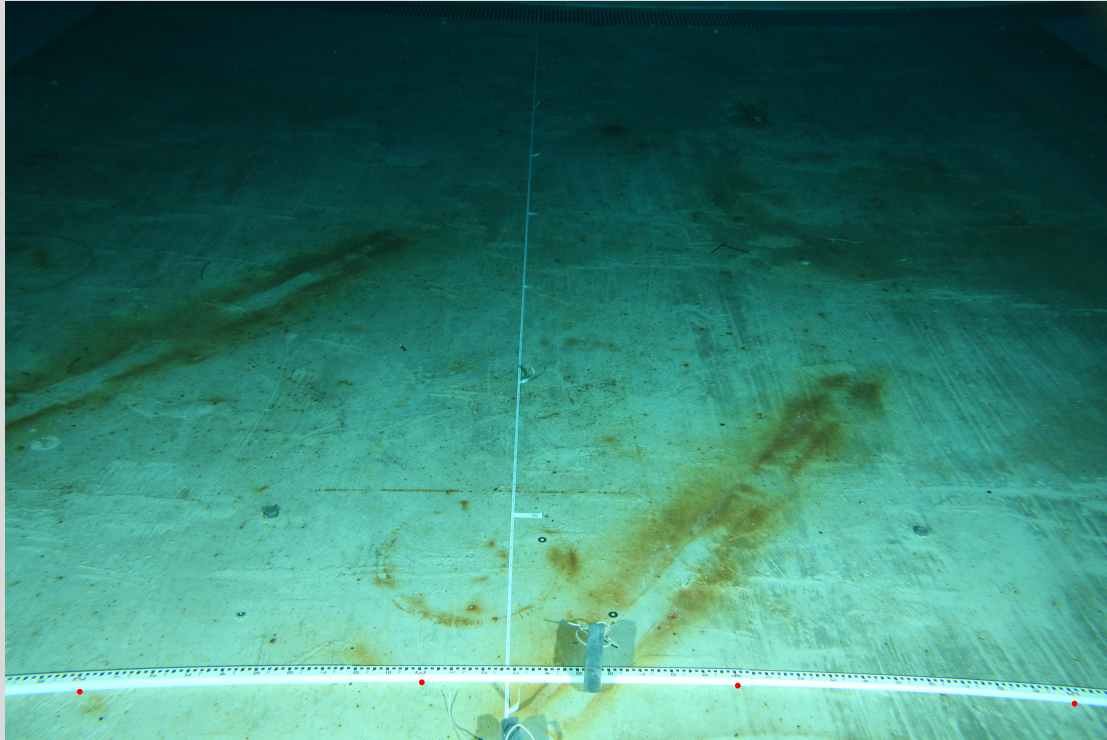


Find the error in the measurements for the near scale and see how they compare to errors with the far scale.

```
GraphicsGrid[  
  {{zoomPick[orig2, 75, cph2 = {289, 197}],  
    zoomPick[orig2, 75, cph3 = {1640, 234}]},  
  {{zoomPick[orig2, 75, cph4 = {2888, 221}],  
    zoomPick[orig2, 75, cph5 = {4218, 150}]}}  
]
```



```
Show[orig2,
Graphics[{Red, Point[cpnh2], Point[cpnh3], Point[cpnh4], Point[cpnh5]}],
ImageSize → Large]
```



Calculate the distance between adjacent calibration points, known to be exactly 1.00 m apart.

```
pdnhd23 = planarDist[{cpnh2, cpnh3}]
```

```
0.695241
```

```
pdnhd34 = planarDist[{cpnh3, cpnh4}]
```

```
0.652913
```

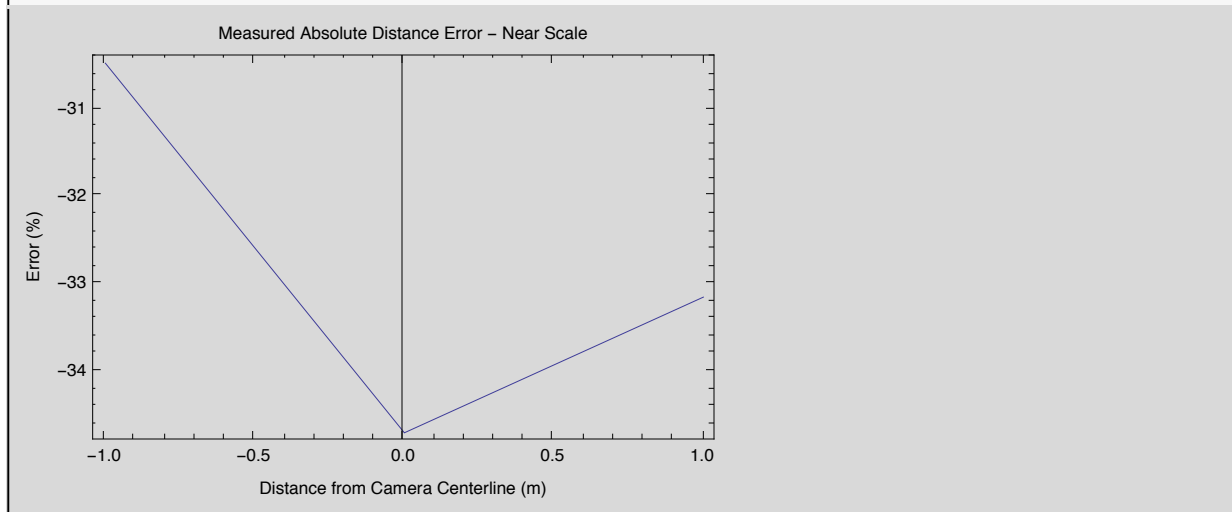
```
pdnhd45 = planarDist[{cpnh4, cpnh5}]
```

```
0.668453
```

```
nearCalData = {{-1, (pdnhd23 - 1) 100}, {0, (pdnhd34 - 1) 100}, {1, (pdnhd45 - 1) 100}}
```

```
{{-1, -30.4759}, {0, -34.7087}, {1, -33.1547}}
```

```
ListPlot[nearCalData, {Frame → True, Joined → True,
  FrameLabel → {"Distance from Camera Centerline (m)",
    "Error (%)", "Measured Absolute Distance Error - Near Scale"}}]
```

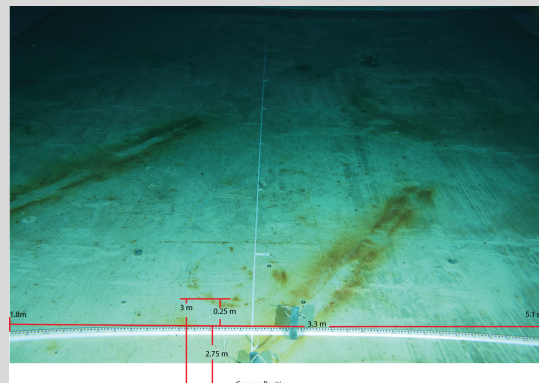
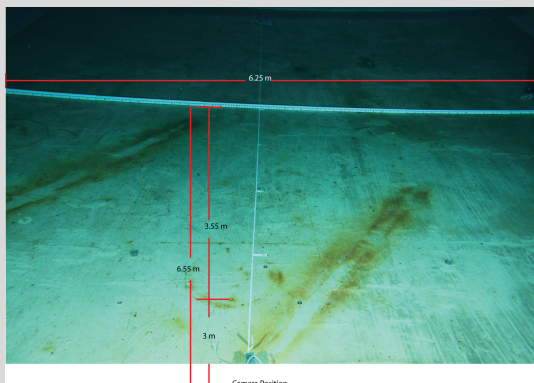


The near-scale errors are even worse than the far-scale errors, and all errors are unacceptably large.

Correcting for Pincushion Distortion

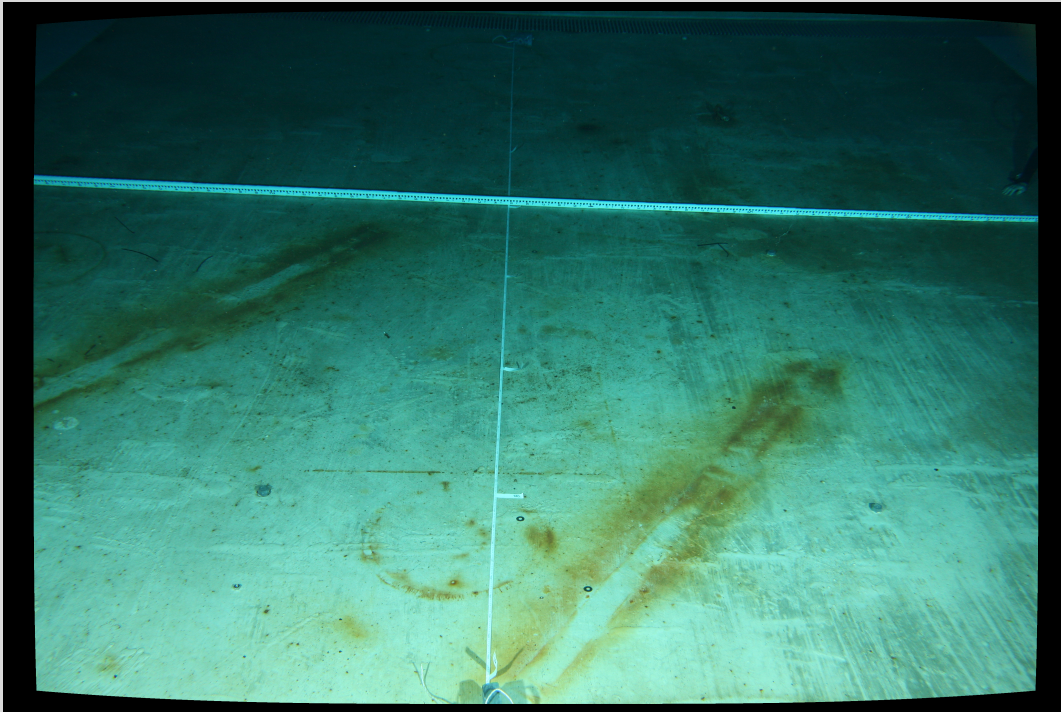
The camera housing's flat-plate viewport causes pincushion distortion, which accounts for the apparent curvature of the straight scale; the scale ends appear to bend up in the far field and down in the near field.

```
GraphicsGrid[{{anno1, anno2}}, ImageSize → Large]
```

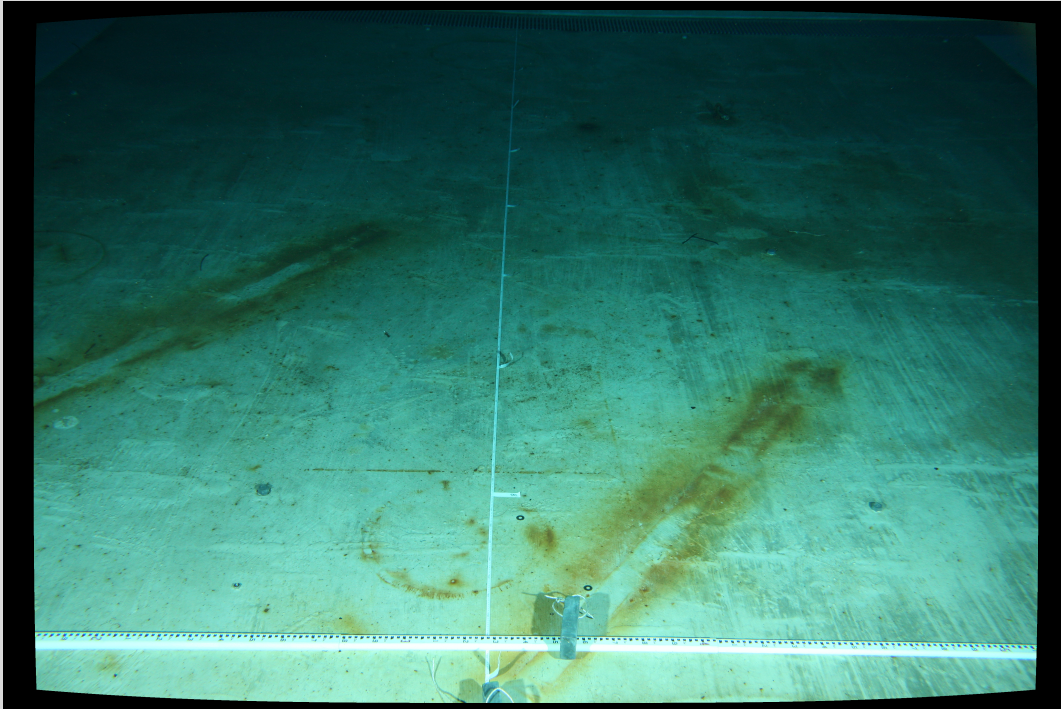


The distortion must be accounted for. Assume monotonic pincushion distortion for now, although we probably have a combination of barrel and pincushion distortion, sometimes called mustache distortion.

```
corr1 = ImageForwardTransformation[orig1, (rd = # - { $\frac{1}{2}$ ,  $\frac{1}{3}$ };  
rd (1 + -0.23 Norm@rd2) + { $\frac{1}{2}$ ,  $\frac{1}{3}$ }) &]
```



```
corr2 = ImageForwardTransformation[orig2, {rd = # - { $\frac{1}{2}$ ,  $\frac{1}{3}$ };
rd (1 + -0.23 Norm@rd2) + { $\frac{1}{2}$ ,  $\frac{1}{3}$ }} &]
```

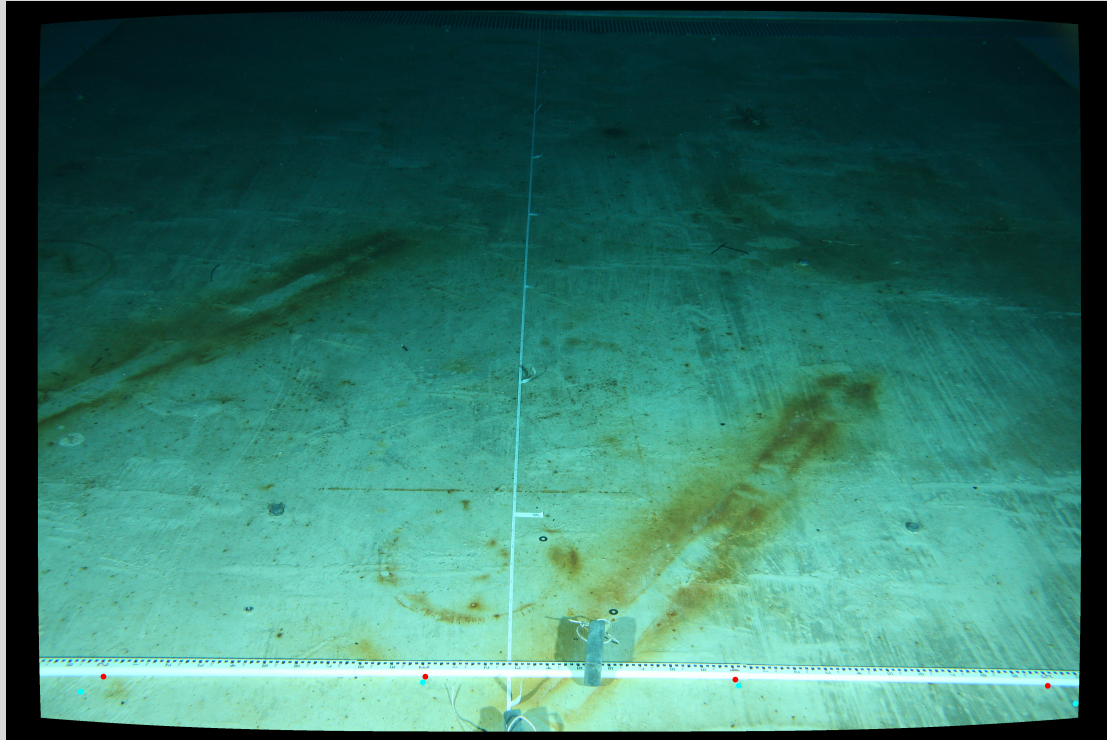


Write a function that accepts points chosen in the distorted image and remaps them to where they would be in an undistorted image.

```
distCorr[{x_, y_}] := Module[{rd},
  rd =  $\frac{\{x, y\} - \frac{\text{imageSize}}{2}}{\text{hRes}}$ ;
  (rd (1 + -0.23 Norm@rd2) hRes) +  $\frac{\text{imageSize}}{2}$ 
]
```

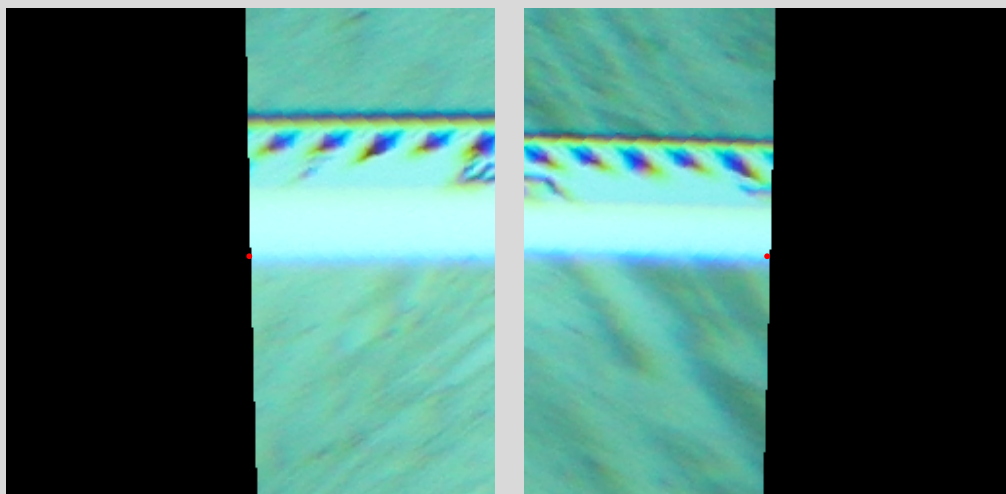
Check the function by plotting corrected and uncorrected calibration points onto a corrected image.

```
Show[corr2, Graphics[{Red, Point[distCorr@cpnh2], Point[distCorr@cpnh3],
  Point[distCorr@cpnh4], Point[distCorr@cpnh5], Cyan, Point[cpnh2],
  Point[cpnh3], Point[cpnh4], Point[cpnh5]}], ImageSize → Large]
```

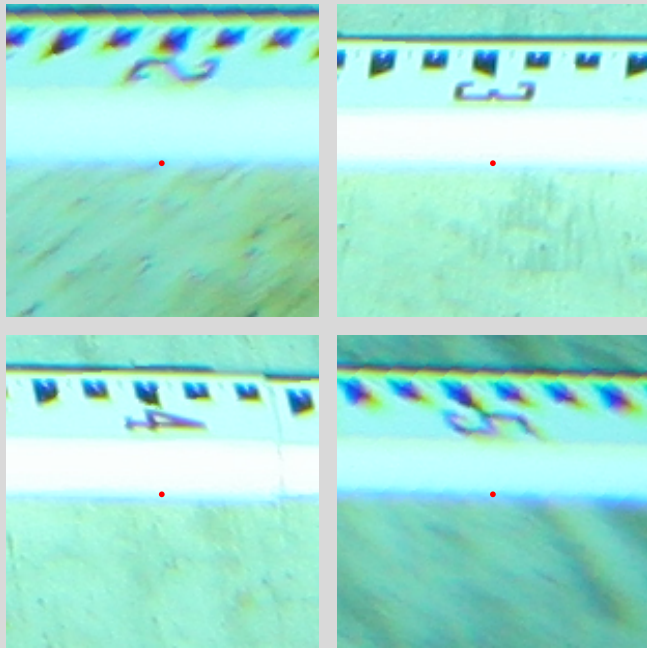


Zoom in on the corrected points to ensure they are reasonable.

```
GraphicsGrid[
  {{zoomPick[corr2, 125, {133, 259}], zoomPick[corr2, 125, {4234, 219}]}}]
```




```
GraphicsGrid[
  {{zoomPick[corr2, 75, distCorr@cph2],
    zoomPick[corr2, 75, distCorr@cph3]},
   {zoomPick[corr2, 75, distCorr@cph4],
    zoomPick[corr2, 75, distCorr@cph5]}}
]
```



Now that the image is linear, we must correct for the fact that the edges of the image have moved. Angles are calculated based on a given pixel's position relative to the edges of the image (i.e. the field of view), so changes to the position of either the pixel or the edges must be taken into account. All of the previous camera geometry functions must be rewritten.

```
hAngle[{hPixLoc_, vPixLoc_}] :=
  ArcTan[ $\left( \left( \text{distCorr}[\{h\text{PixLoc}, v\text{PixLoc}\}][[1]] - \frac{h\text{Res}}{2} \right) \right.$ 
    Tan[ $\frac{1}{h\text{Res}} h\text{FOV} \left( \text{distCorr}[\{h\text{Res}, v\text{PixLoc}\}][[1]] - \frac{h\text{Res}}{2} \right)$ 
  ] /
  ( $\text{distCorr}[\{h\text{Res}, v\text{PixLoc}\}][[1]] - \frac{h\text{Res}}{2}$ )]
```

Find the horizontal field of view at the bottom edge, one-quarter up, and halfway up the image.

```
2 hAngle[#] / Degree & /@ {{hRes, 0}, {hRes, vRes / 4}, {hRes, vRes / 2}}
```

```
{45.2681, 45.5129, 45.53}
```

```
vAngle[{hPixLoc_, vPixLoc_}] :=
  ArcTan[(((distCorr[{hPixLoc, vPixLoc}][[2]] -  $\frac{vRes}{2}$ )
    Tan[ $\frac{1}{vRes}$ vFOV (distCorr[{hPixLoc, vRes}][[2]] -  $\frac{vRes}{2}$ ))] /
    (distCorr[{hPixLoc, vRes}][[2]] -  $\frac{vRes}{2}$ ))]
```

Find the vertical field of view at the left edge, one-quarter across, and halfway across the image.

```
2 vAngle[#] / Degree & /@ {{0, vRes}, {hRes / 4, vRes}, {hRes / 2, vRes}}
```

```
{32.1029, 33.2541, 33.3831}
```

```
hPlanarOffset[{hPixLoc_, vPixLoc_}] :=
  Tan[hAngle[{hPixLoc, vPixLoc}]]
  (camHeight / Tan[camTilt - vAngle[{hPixLoc, vPixLoc}]])
```

Points in the centerline of the image at the bottom, middle, and top of the image should have a horizontal offset of zero.

```
hPlanarOffset[#] & /@ {{hRes / 2, 0}, {hRes / 2, 2912 / 2}, {hRes / 2, 2912}}
```

```
{0., 0., 0.}
```

```
vPlanarDist[{hPixLoc_, vPixLoc_}] :=
  camHeight / Tan[camTilt - vAngle[{hPixLoc, vPixLoc}]]
```

Find the planar distance to a point on the centerline at the bottom, middle, and top of the image.

```
vPlanarDist[#] & /@ {{hRes / 2, 0}, {hRes / 2, vRes / 2}, {hRes / 2, vRes}}
```

```
{2.36652, 4.32713, 10.1939}
```

Now we can recalculate some of the previous distances using corrected calibration points.

Calculate the distance from the nadir point of the camera to the calibration point nearest the vertical tape in the far scale, and compare it to the measured distance of 6.55 m.

```
vPlanarDist[cpfh3p5]
```

```
5.98332
```

Calculate the width of the image at the calibration point, and compare it to the measured width of 6.25 m.

```
2 hPlanarOffset[{hRes, cpf3p5[[2]]}]
```

```
4.92361
```

Calculate the distance between adjacent calibration points, known to be exactly 1.00 m apart.

```
pdnhcd23 = planarDist[{cpnh2, cpnh3}]
```

```
0.670059
```

```
pdnhcd34 = planarDist[{distCorr@cpnh3, distCorr@cpnh4}]
```

```
0.646485
```

```
pdnhcd45 = planarDist[{distCorr@cpnh4, distCorr@cpnh5}]
```

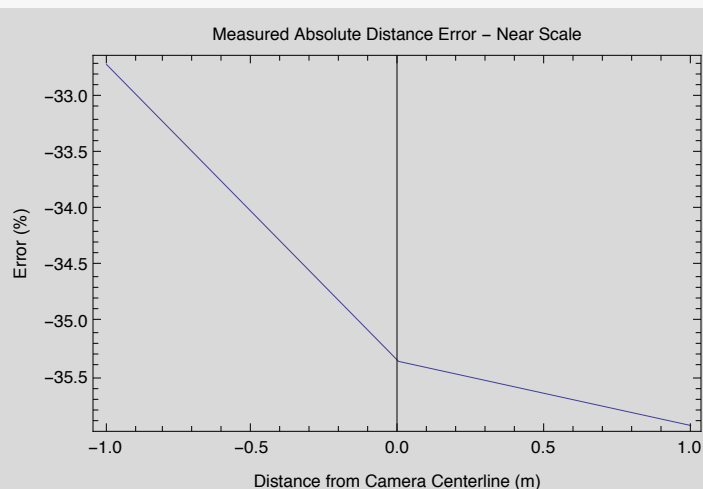
```
0.640777
```

```
nearCalDataCorr =
```

```
{{-1, (pdnhcd23 - 1) 100}, {0, (pdnhcd34 - 1) 100}, {1, (pdnhcd45 - 1) 100}}
```

```
{{-1, -32.7065}, {0, -35.3515}, {1, -35.9223}}
```

```
ListPlot[nearCalData, {Frame → True, Joined → True,
  FrameLabel → {"Distance from Camera Centerline (m)",
    "Error (%)", "Measured Absolute Distance Error - Near Scale"}}]
```



```
{StandardDeviation[nearCalData][[2]], StandardDeviation[nearCalDataCorr][[2]]}
```

```
{2.14119, 1.71575}
```

The distortion correction has reduced the standard deviation of the errors, but the the magnitude of

the errors is much too large.

Correcting the Field of View