

Rover Fluorescence Imaging

P. McGill

2 Feb23

This notebook imports fluorescence images from Rover and quantifies how much phytodetritus is present on the seafloor.

First get all the file names from a directory of images from Pulse 72.

```
fnames = FileNames["StaM_*.tiff",  
    "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images"];
```

```
Length[fnames]
```

164

Check that we're getting the files we want.

```
fnames[[;;3]]
```

```
{/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_191019_20_04_38.  
    tiff,  
/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_191021_22_42_34.  
    tiff,  
/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_191024_00_20_16.  
    tiff}
```

```
fnames[[-3;;]]
```

```
{/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_201022_16_29_24.  
    tiff,  
/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_201025_05_13_33.  
    tiff,  
/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_201027_11_33_31.  
    tiff}
```

Look at a sample image.

```
fnames[[100]]
```

```
/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-2019.10  
.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_200601_11_50_58.tiff
```

```
image = Import[fnames[[100]]]
```

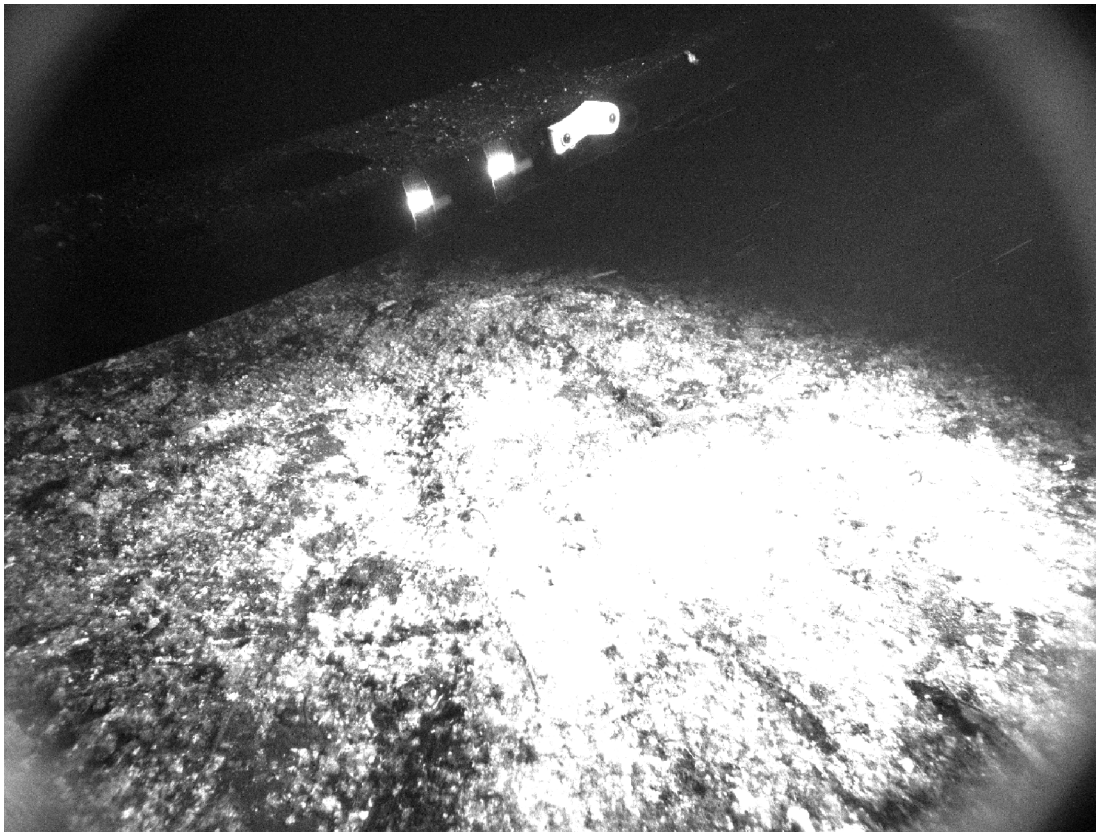


```
{imageW, imageH} = ImageDimensions@image
```

```
{1360, 1024}
```

Increase the brightness to see where the lens and housing port are actually imaging.

```
ImageAdjust[image, {0, 5}]
```



Full Field of View

Find the field of view. 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.

```
camTilt = 30 Degree;
camHeight = 0.42;
pixelSize = 6.45 × 10-6;
hSensor = 1360 pixelSize;
vSensor = 1024 pixelSize;
lensFL = 0.005;
```

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

```
(*seawaterRI=1.3375+(400 0.000016)+0.00577*)
seawaterRI = 1.34;

airRI = 1.00;
```

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

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

$$\frac{hFOV}{Degree} // N$$

$$61.578$$

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

$$\frac{vFOV}{Degree} // N$$

$$49.9164$$

Calculate the intersections of the lines formed between the camera and the corners of the field of view with the seafloor directly using simple trigonometry. “illf” means “intersection lower left, full field of view.” Note that all the Z components are zero because they’re on the seafloor.

$$illf = \left\{ -\operatorname{Tan} \left[\frac{hFOV}{2} \right] \frac{camHeight}{\operatorname{Sin} \left[camTilt + \frac{vFOV}{2} \right]}, \frac{camHeight}{\operatorname{Tan} \left[camTilt + \frac{vFOV}{2} \right]}, 0 \right\};$$

$$ilrf = \left\{ \operatorname{Tan} \left[\frac{hFOV}{2} \right] \frac{camHeight}{\operatorname{Sin} \left[camTilt + \frac{vFOV}{2} \right]}, \frac{camHeight}{\operatorname{Tan} \left[camTilt + \frac{vFOV}{2} \right]}, 0 \right\};$$

$$iulf = \left\{ -\operatorname{Tan} \left[\frac{hFOV}{2} \right] \frac{camHeight}{\operatorname{Sin} \left[camTilt - \frac{vFOV}{2} \right]}, \frac{camHeight}{\operatorname{Tan} \left[camTilt - \frac{vFOV}{2} \right]}, 0 \right\};$$

$$iurf = \left\{ \operatorname{Tan} \left[\frac{hFOV}{2} \right] \frac{camHeight}{\operatorname{Sin} \left[camTilt - \frac{vFOV}{2} \right]}, \frac{camHeight}{\operatorname{Tan} \left[camTilt - \frac{vFOV}{2} \right]}, 0 \right\};$$

Find the footprint of the full field of view. Note that since the sides of the footprint are curved and not straight, we can’t just find the four corners and draw straight lines between them. Instead, we’ll find the left and right edges of the field of view as we step the pitch angle down from the top to the bottom.


```

footprintFull = Polygon[
  Join[
    Table[{{-Tan[ $\frac{hFOV}{2}$ ]  $\frac{camHeight}{\sin[camTilt + \theta]}$ ,  $\frac{camHeight}{\tan[camTilt + \theta]}$ },
      { $\theta$ ,  $\frac{-vFOV}{2}$ ,  $\frac{vFOV}{2}$ ,  $\frac{vFOV}{10}$ }},
    Table[{{Tan[ $\frac{hFOV}{2}$ ]  $\frac{camHeight}{\sin[camTilt + \theta]}$ ,  $\frac{camHeight}{\tan[camTilt + \theta]}$ },
      { $\theta$ ,  $\frac{vFOV}{2}$ ,  $\frac{-vFOV}{2}$ ,  $\frac{-vFOV}{10}$ }}]
  ] (* end Join *)
] (* end Polygon *)

```

```

Polygon[  Number of points: 22  

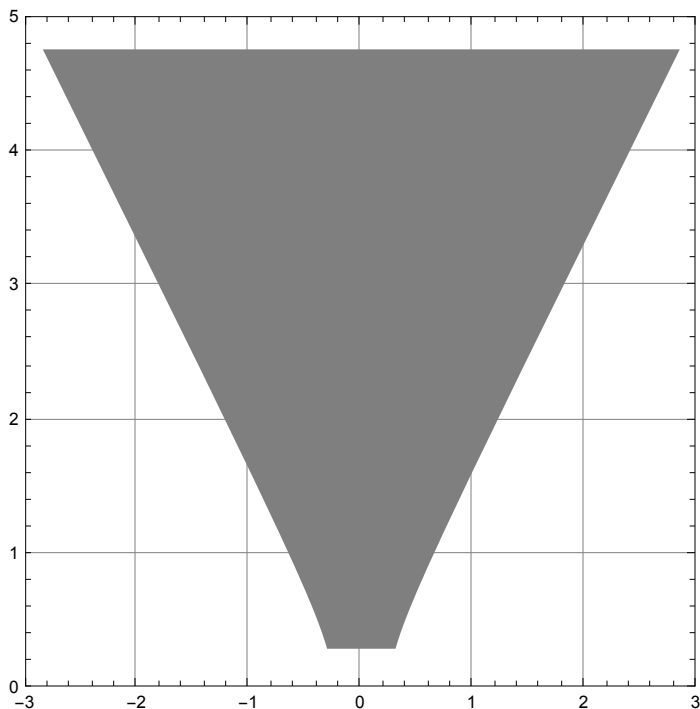
Embedding dimension: 2 ]

```

```

footPlotFull =
Graphics[{Gray, footprintFull}, GridLines -> {Range[-2, 2], Range[1, 4]},
  AspectRatio -> 1, PlotRange -> {{-3, 3}, {0, 5}}, Frame -> True]

```



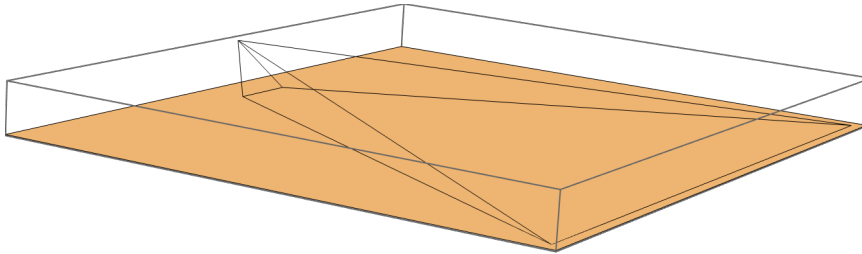
Define the location of the camera viewport above the origin.

```
c = {0, 0, camHeight};
```

Define the seafloor with three points that lie upon it.

```
seafloor = {{-1, 0, 0}, {1, 0, 0}, {0, 1, 0}};
```

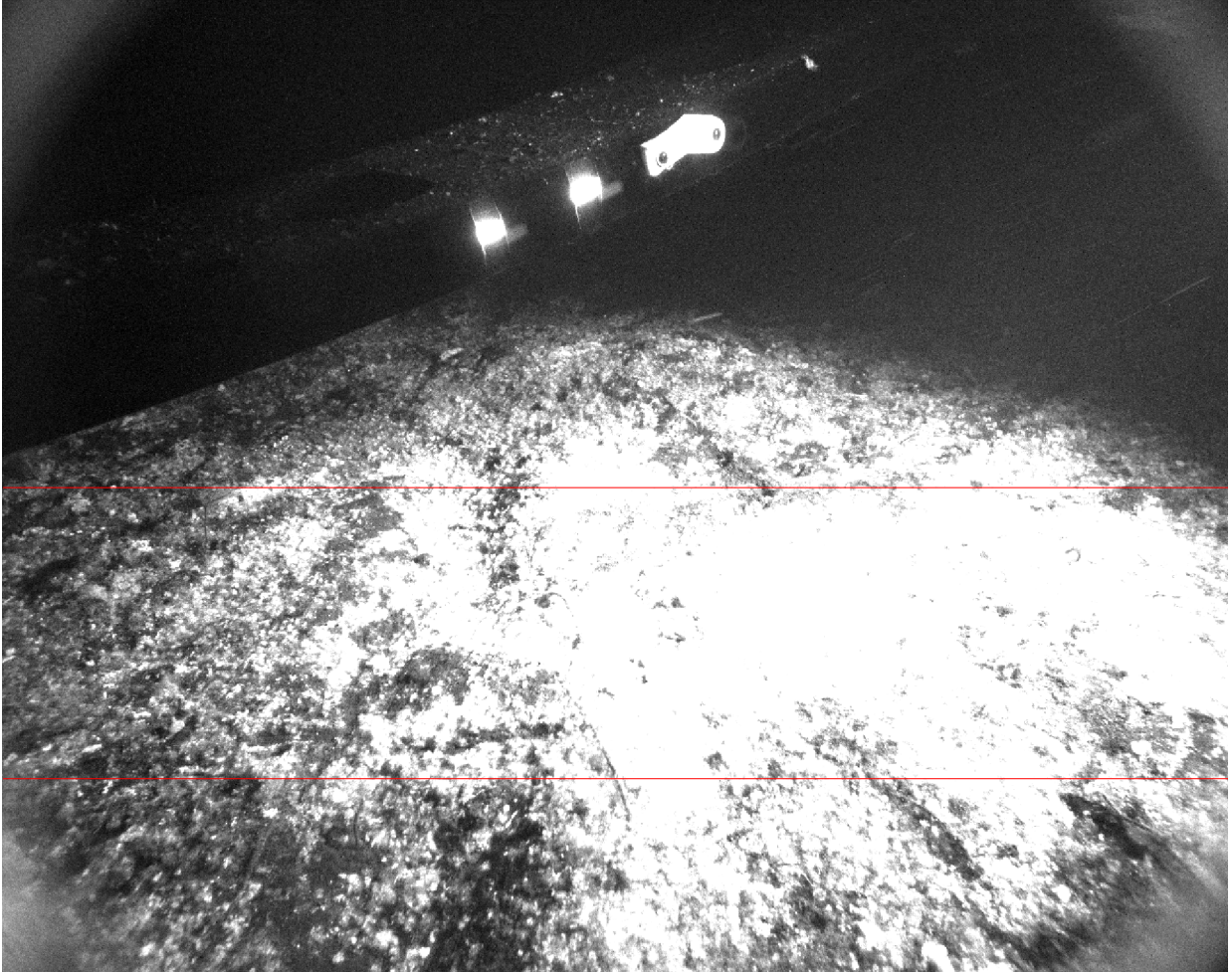
```
Graphics3D[{InfinitePlane[seafloor], Line[{c, iulf}], Line[{c, iurf}],  
  Line[{c, illf}], Line[{c, ilrf}], Line[{iulf, iurf, ilrf, illf, iulf}]}
```



Truncated Field of View

Not all of the field of view is usable due to vignetting by the camera housing, parts of Rover obscuring the view, and uneven lighting. After increasing the brightness to see the camera vignetting, designate a section of the image that is mostly free of these artifacts by eliminating the bottom 20% and the top 50% of the image.

```
Show[ImageAdjust[image, {0, 5}],
Graphics[{Red, Line[{{0, imageH / 5}, {imageW, imageH / 5}}],
Line[{{0, imageH / 2}, {imageW, imageH / 2}}]}]]]
```



Calculate the intersections of the lines formed between the camera and the corners of the truncated field of view with the seafloor directly using simple trigonometry. “illt” means “intersection lower left, truncated field of view.” Note that all the Z components are zero because they’re on the seafloor.

$$illt = \left\{ -\tan\left[\frac{hFOV}{2}\right] \frac{camHeight}{\sin[camTilt + 0.6 \frac{vFOV}{2}]}, \frac{camHeight}{\tan[camTilt + 0.6 \frac{vFOV}{2}]}, 0 \right\};$$

$$ilrt = \left\{ \tan\left[\frac{hFOV}{2}\right] \frac{camHeight}{\sin[camTilt + 0.6 \frac{vFOV}{2}]}, \frac{camHeight}{\tan[camTilt + 0.6 \frac{vFOV}{2}]}, 0 \right\};$$

$$iult = \left\{ -\tan\left[\frac{hFOV}{2}\right] \frac{camHeight}{\sin[camTilt]}, \frac{camHeight}{\tan[camTilt]}, 0 \right\};$$

$$iurt = \left\{ \tan\left[\frac{hFOV}{2}\right] \frac{camHeight}{\sin[camTilt]}, \frac{camHeight}{\tan[camTilt]}, 0 \right\};$$

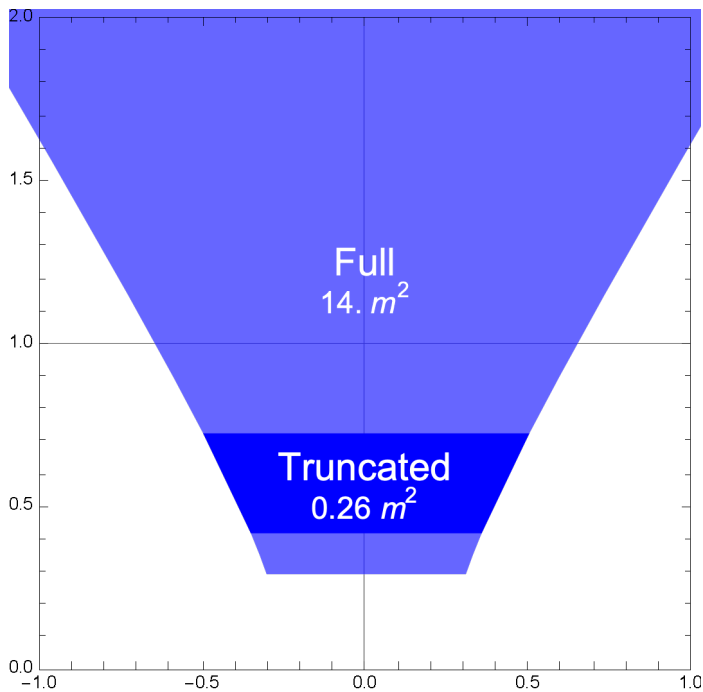
To find the footprint we’ll ignore the curved sides and just take the four corners of the truncated field

of view.

```
footprintTrun = Polygon[Take[#, 2] & /@ {iult, iurt, ilrt, illt}]
```

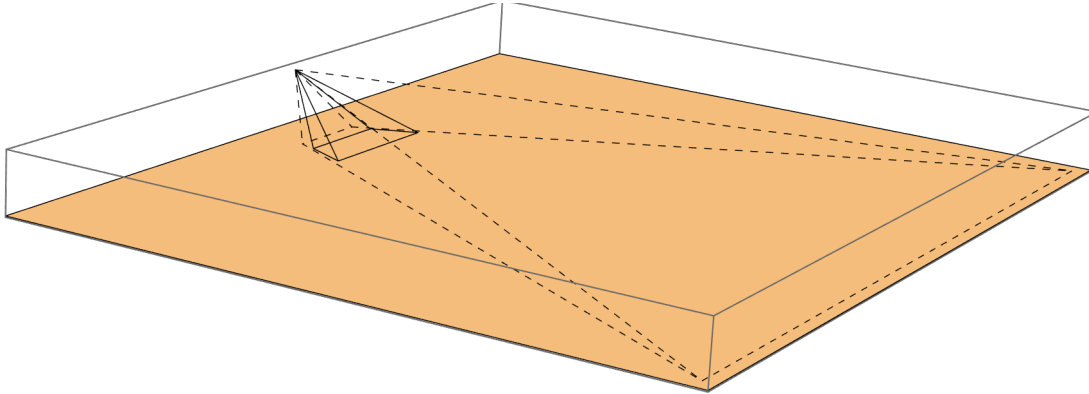
```
Polygon[ Number of points: 4  
Embedding dimension: 2]
```

```
footPlotBoth = Graphics[{Opacity[0.6], Blue, footprintFull,  
  Opacity[1], White, Text[Style["Full", FontSize → 20], {0, 1.25}], Blue,  
  footprintTrun, White, Text[Style["Truncated", FontSize → 20], {0, 0.62}]}],  
GridLines → {Range[-1, 1], Range[0, 2]}, AspectRatio → 1,  
PlotRange → {{-1, 1}, {0, 2}}, Frame → True,  
Epilog → {Inset[Style[ToString[NumberForm[Area@footprintFull, 2]] <> " m²",  
  White, 16], {0, 1.13}], Inset[Style[  
  ToString[NumberForm[Area@footprintTrun, 2]] <> " m²", White, 16], {0, 0.5}]}]}
```



Plot the full (dashed) and truncated (solid) views.

```
Graphics3D[{InfinitePlane[seafloor], Dashed, Line[{c, iulf}], Line[{c, iurf}],
  Line[{c, illf}], Line[{c, ilrf}], Line[{iulf, iurf, ilrf, illf, iulf}],
  Dashing[None], Line[{c, iult}], Line[{c, iurt}], Line[{c, illt}],
  Line[{c, ilrt}], Line[{iult, iurt, ilrt, illt, iult}]}]
```



Perspective Transformation

Find the geometric transformation that converts between the original and overhead points of view of the truncated area. First define the points in the image that designate the truncated field of view in the order lower left, lower right, upper left, upper right.

```
imagePts = {{0, 1024 / 5}, {1360, 1024 / 5}, {0, 1024 / 2}, {1360, 1024 / 2}}
{{0,  $\frac{1024}{5}$ }, {1360,  $\frac{1024}{5}$ }, {0, 512}, {1360, 512}}
```

Then define the points on the seafloor corresponding to the truncated field of view, taking only the x and y coordinates.

```
floorPts = Take[#, 2] & /@ {illt, ilrt, iult, iurt}
{{-0.354077, 0.420368}, {0.354077, 0.420368},
 {-0.500522, 0.727461}, {0.500522, 0.727461}}
```

Find the width in meters of the horizontal line at the 50% and 20% points of the image.

```
width50pct = 2 floorPts[[4, 1]]
1.00104
```

```
width20pct = 2 floorPts[[2, 1]]
0.708155
```

```
ratio = width20pct / width50pct
0.707417
```

```
dist20to50pct = floorPts[[3, 2]] - floorPts[[2, 2]]
0.307093
```

Find the number of pixels per meter in the image at the 50% line.

```
pxlsPerMtr = 1360 / width50pct
1358.58
```

Lastly define a set of points in the pixel coordinate system that correspond to a view of the image from above. Assume that the horizontal line across the 50% point of the image remains unchanged, and that the horizontal line across the 20% point of the image gets compressed. The end points of the 20% line will move in both x and y according to the scale (pixels/meter) established by the 50% line. Specify the points in the order {lower left, lower right, upper left, upper right}.

```
xformPts = {{ {  $\frac{\text{imageW}}{2} - \frac{\text{width20pct pxlsPerMtr}}{2}$ ,  $\frac{\text{imageH}}{2} - \text{dist20to50pct pxlsPerMtr}$  },
  {  $\frac{\text{imageW}}{2} + \frac{\text{width20pct pxlsPerMtr}}{2}$ ,  $\frac{\text{imageH}}{2} - \text{dist20to50pct pxlsPerMtr}$  },
  { 0,  $\frac{\text{imageH}}{2}$  }, { imageW,  $\frac{\text{imageH}}{2}$  } }
{{ {198.957, 94.7883}, {1161.04, 94.7883}, {0, 512}, {1360, 512} }}
```

Find the transform that moves the points specified in imagePts to those specified in xformPts.

```
xform = FindGeometricTransform[xformPts, imagePts][[2]]
TransformationFunction[ $\left( \begin{array}{cc|c} 0.591953 & -0.541938 & 277.472 \\ -9.76259 \times 10^{-15} & 0.728395 & -69.8581 \\ -1.11022 \times 10^{-16} & -0.000796967 & 1. \end{array} \right)$ ]
```

image



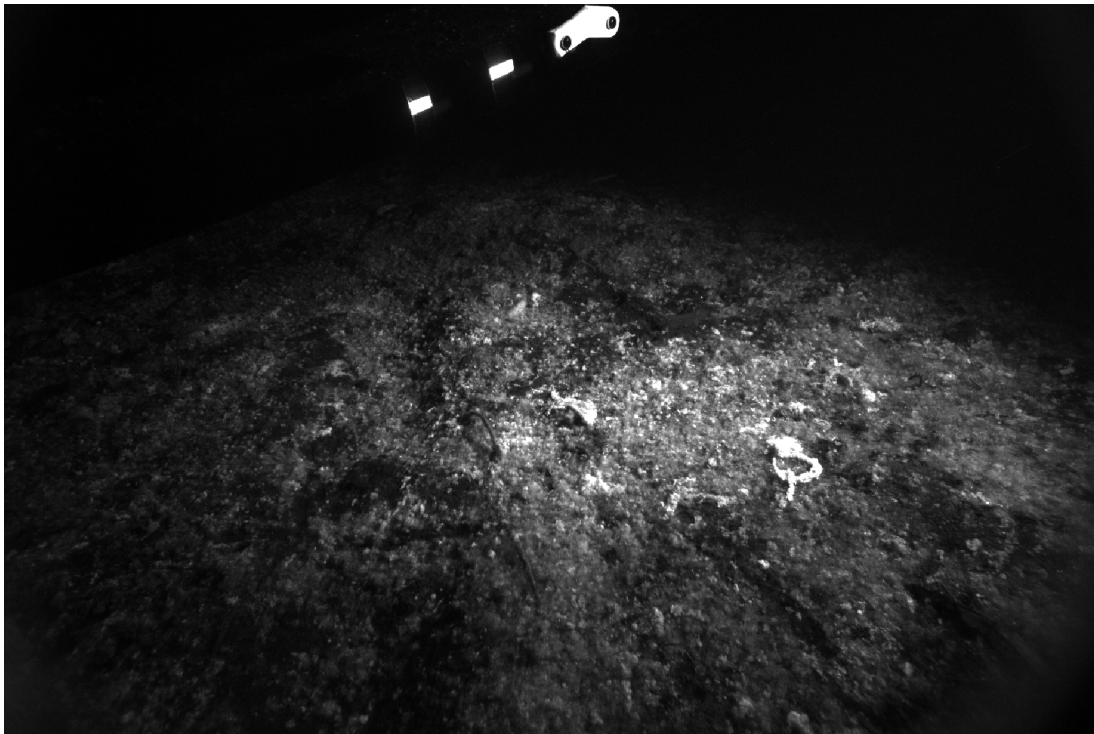
ImageDimensions@image

{1360, 1024}

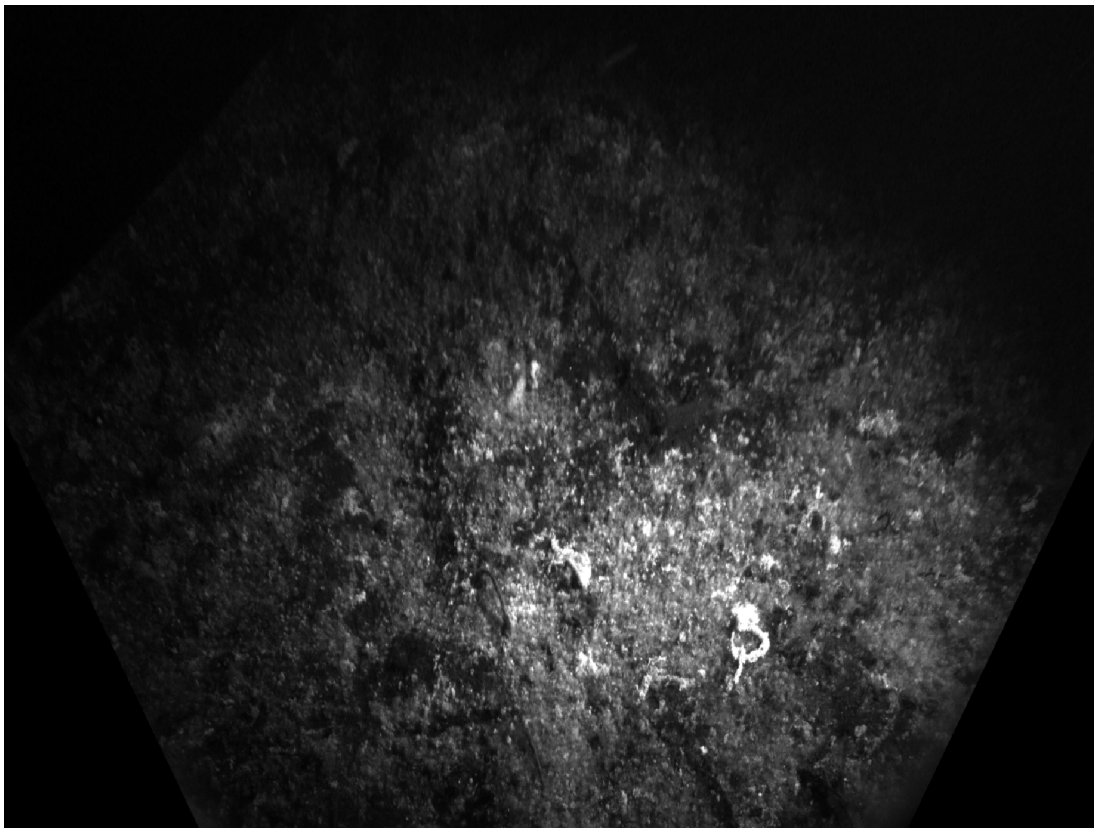
(527 / 596) 1024 // N

905.45

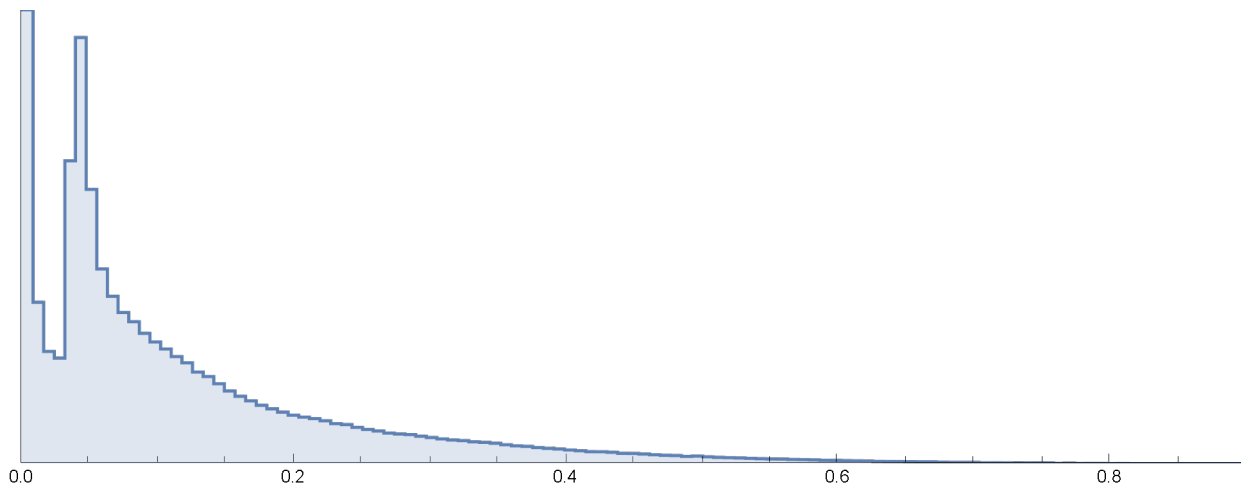

```
ImageTake[image, -905]
```



```
imagex = ImagePerspectiveTransformation[image, xform, PlotRange -> Full]
```



ImageHistogram@imagex



ImageDimensions@imagex

{1360, 1024}

Viewing the Excitation Illumination

The illumination is brighter in the middle and dim on the edges. Even when no fluorescent material is present, some of the blue excitation light leaks past the red long-pass filter on the camera.

```
bkgnd = Import[fnames[[52]]]
```



Transform and crop the background image.

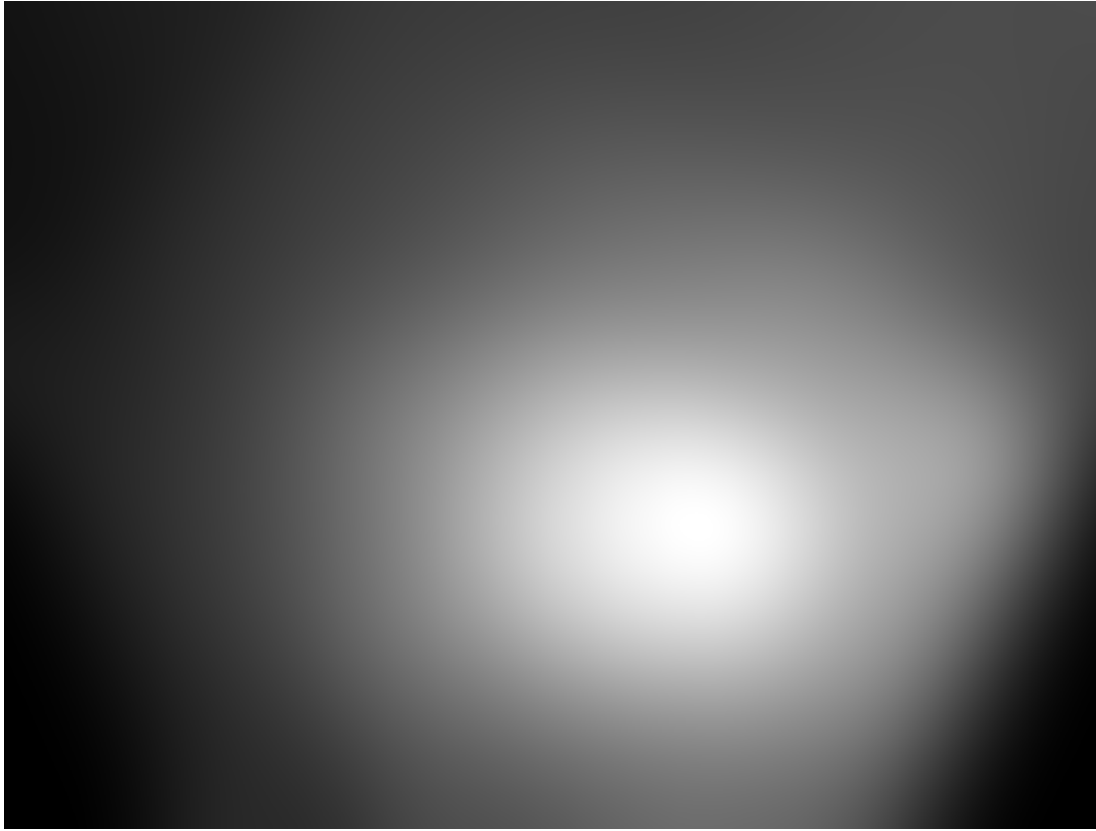
```
bkgndx = ImagePerspectiveTransformation[bkgnd, xform, PlotRange -> Full];
```

Blur the background image with a Gaussian filter of radius 100. This should show the illumination pattern of the excitation light source.

```
bkgndxBlr = GaussianFilter[bkgndx, 150]
```



ImageAdjust@bkgndxBlr

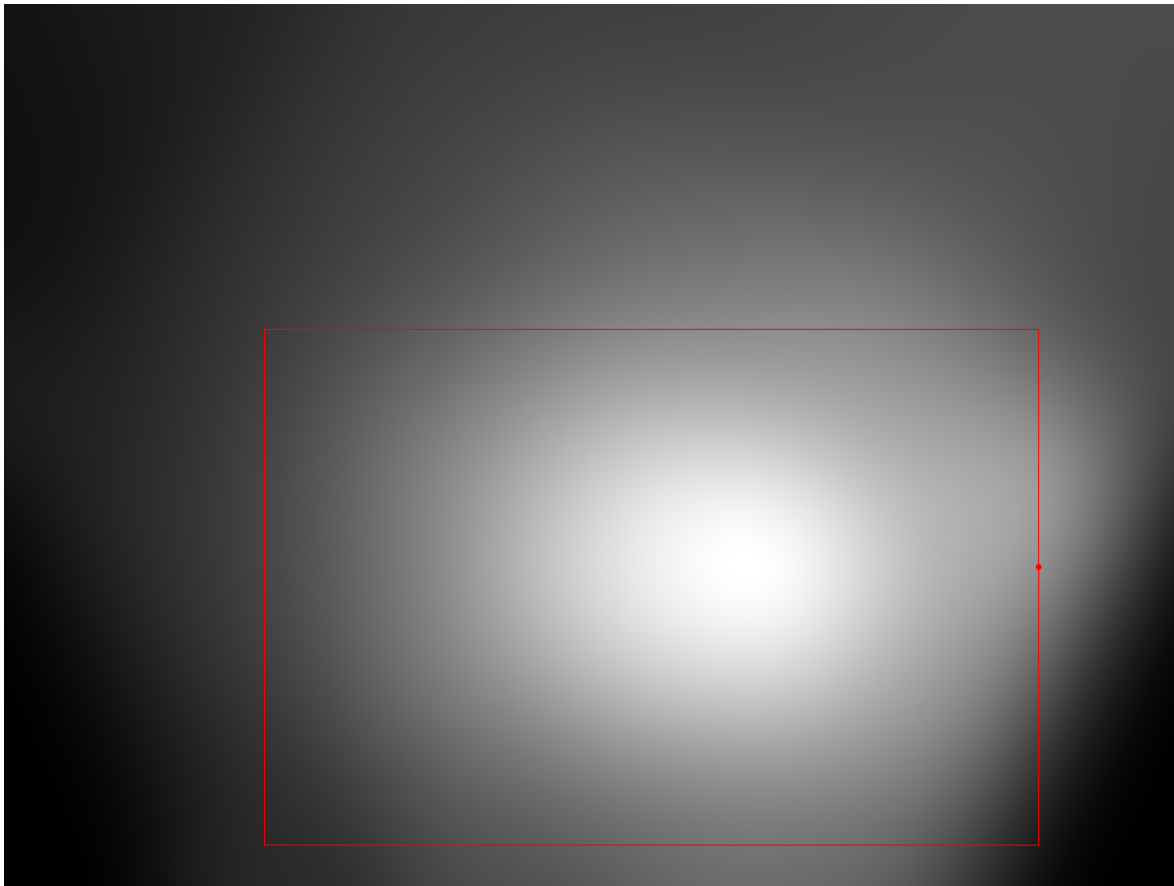


```
{ymax, xmax} = First@Position[ImageData@bkgndxBlr, Max[ImageData@bkgndxBlr]]  
{650, 858}
```

Define a rectangular selection of the transformed image. Define corners in the order {upper left, upper right, lower right, lower left}.

```
xmin = 301; xmax = 1200; ymin = 51; ymax = 650;
```

```
Show[ImageAdjust@bkgndxBlr, Graphics[{Red, Point[{xmax, imageH - ymax}],
  Line[{xmin, ymax}, {xmax, ymax}, {xmax, ymin}, {xmin, ymin}, {xmin, ymax}]}]]
```



Find the dimensions of the selected area in meters.

```
selDims = {900, 600} / pxlsPerMtr
{0.662455, 0.441637}
```

```
selArea = Apply[Times, selDims, {0}]
0.292565
```

```
ImageDimensions@bkgndxBlr
{1360, 1024}
```

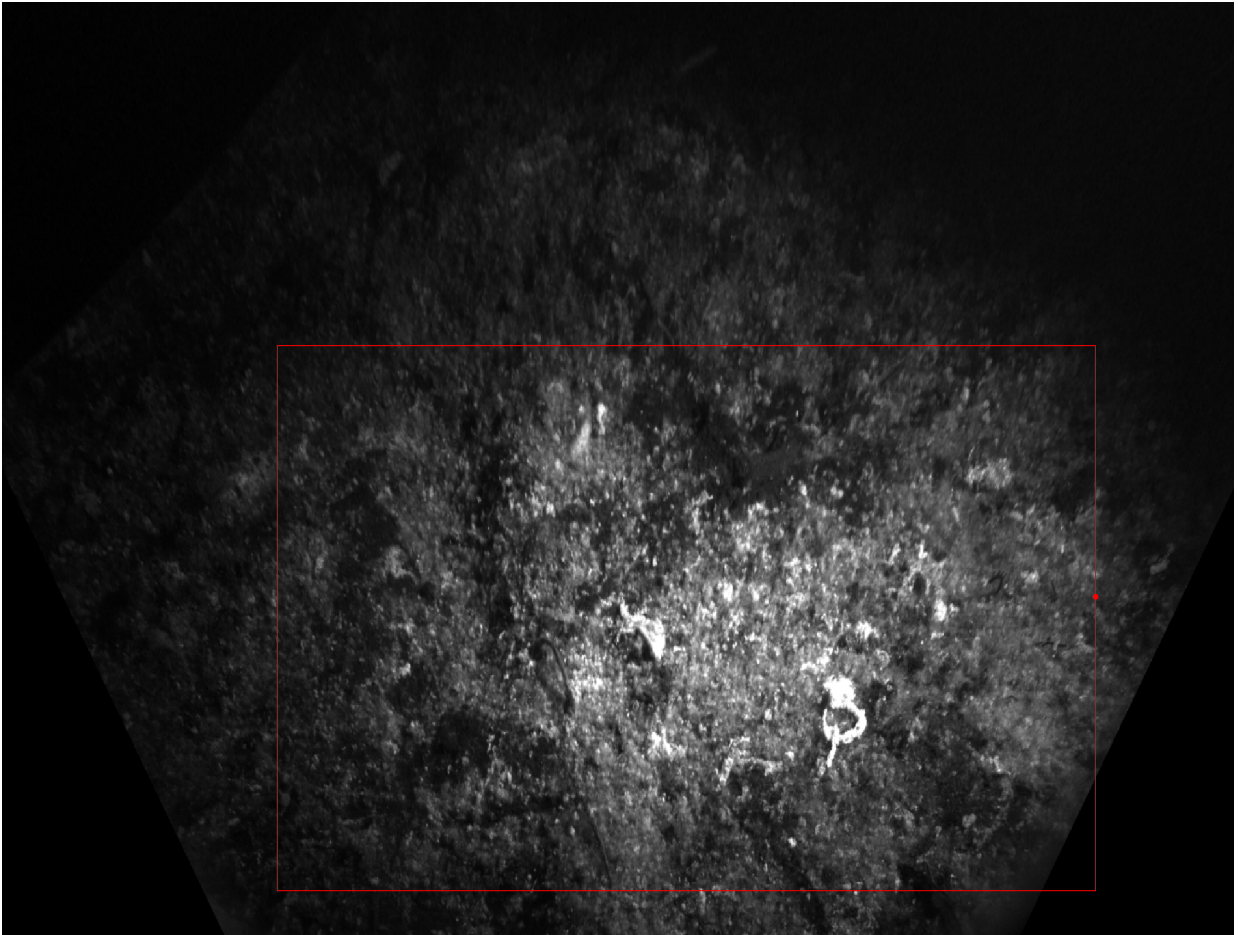
```
brtmsk = ImageTake[bkgndxBlr, 1024 - {ymin, ymax}, {xmin, xmax}]
```



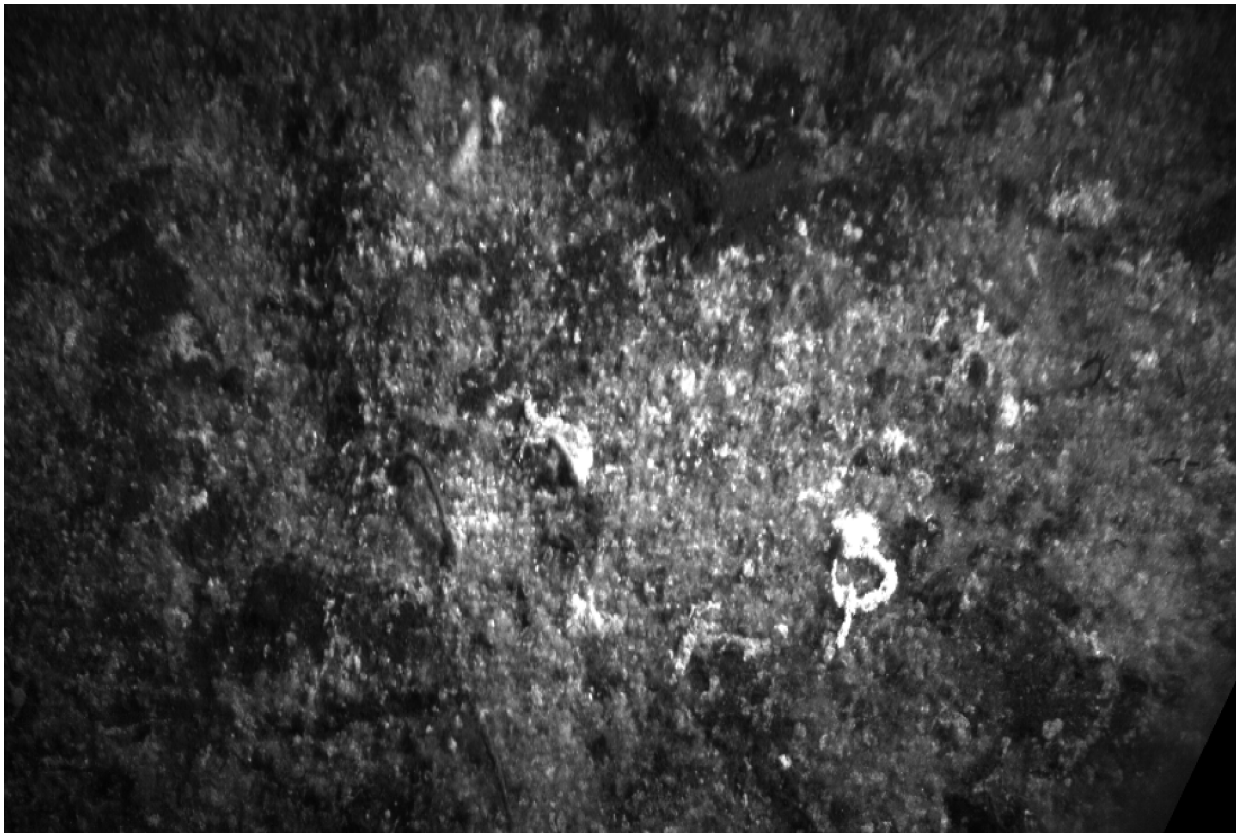
Quantifying the Phytodetritus

Select the area of study on a transformed image.

```
Show[imagex, Graphics[{Red, Point[{xmax, imageH - ymax}],  
  Line[{xmin, ymax}, {xmax, ymax}, {xmax, ymin}, {xmin, ymin}, {xmin, ymax}]}]]
```




```
imageXSel = ImageTake[imageX, {imageH - ymax, imageH - ymin}, {xmin, xmax}]
```



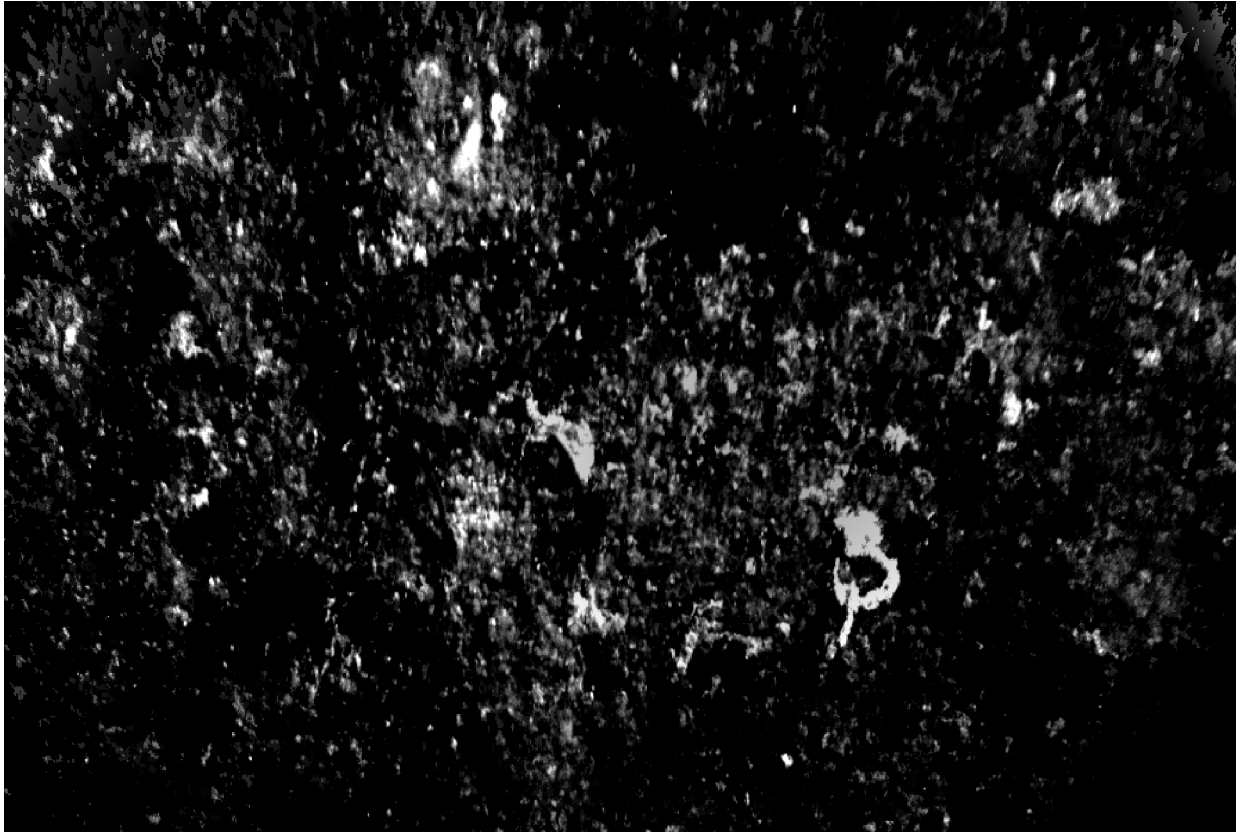
Attempt to equalize the brightness across the image.

```
BrightnessEqualize[imageSel, brtmsk]
```



Equalization seems to kill the image contrast, so bump the contrast up a bit.

```
ImageAdjust[BrightnessEqualize[imageSel, brtmsk], 1.2]
```



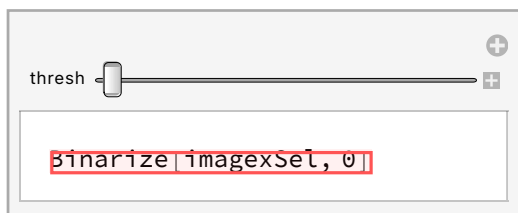
This seems to make things worse, so don't do brightness equalization for now.

```
ImageDimensions@imageSel
```

```
{900, 600}
```

Change all pixels below a threshold to 0, and all above to 1.

```
Manipulate[Binarize[imageSel, thresh], {thresh, 0, 1}]
```



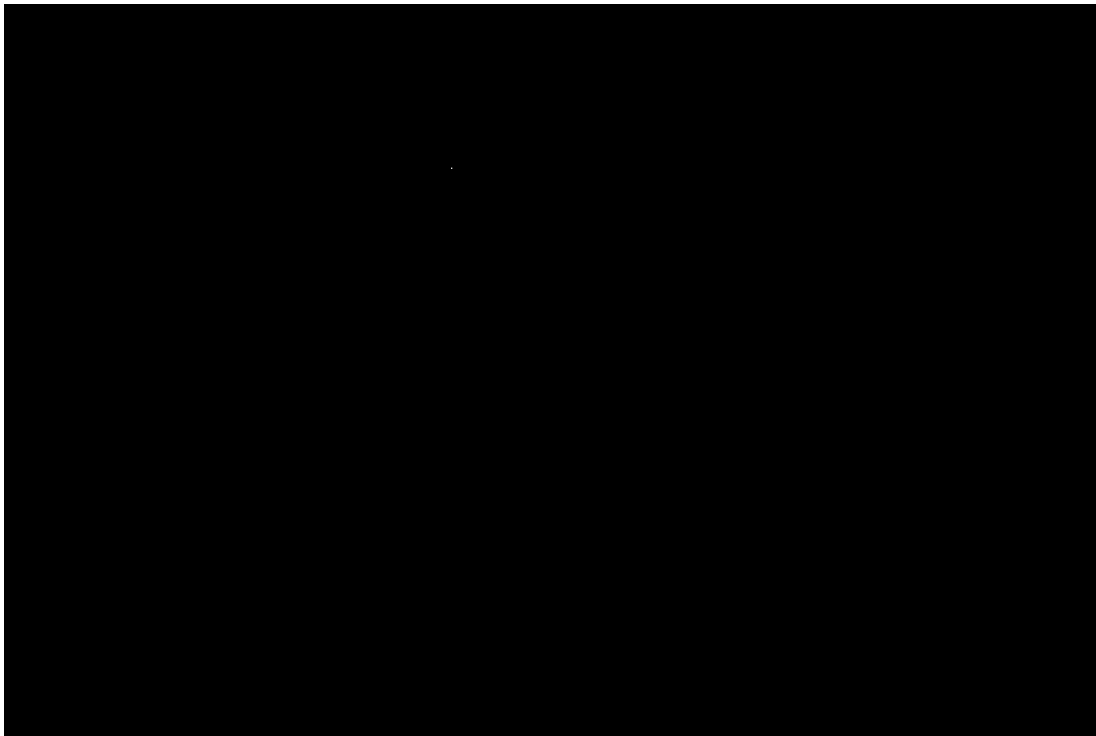
Binarize::imginv: Expecting an image or graphics instead of imageSel.

Make sure that the background light levels don't exceed the threshold of 0.3.

```
ImageTake[bkgndx, {imageH - ymax, imageH - ymin}, {xmin, xmax}]
```



```
Binarize[ImageTake[bkgndx, {imageH - ymax, imageH - ymin}, {xmin, xmax}], 0.3]
```

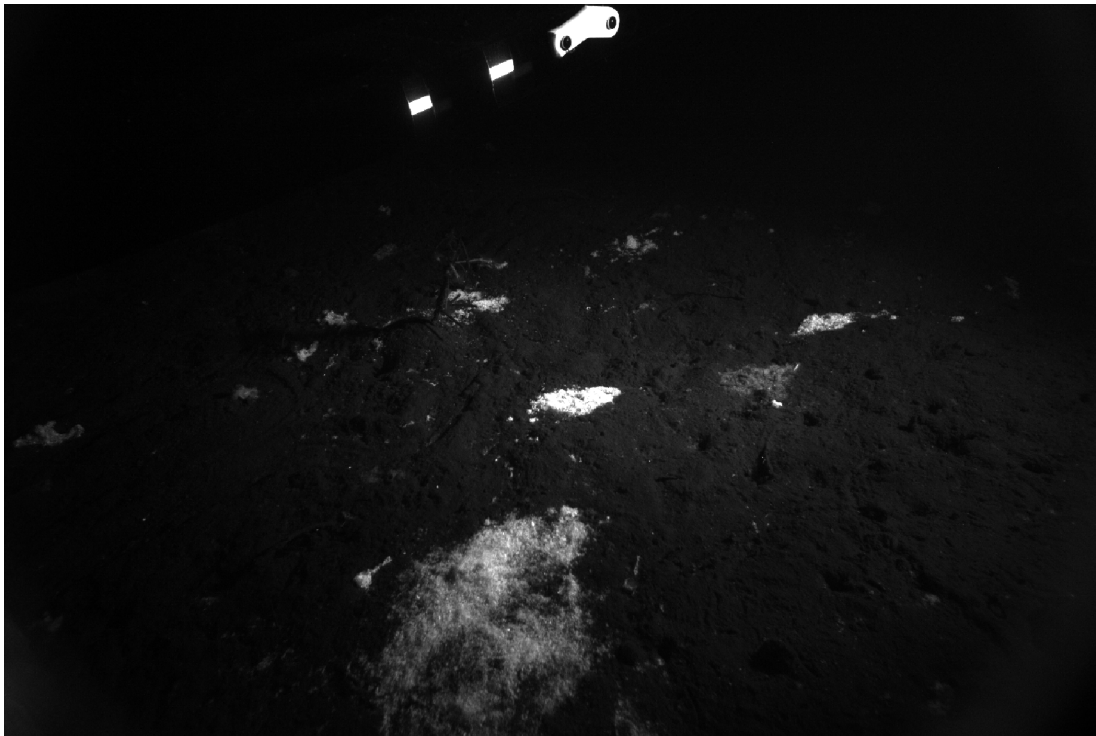


Analyze a new image.

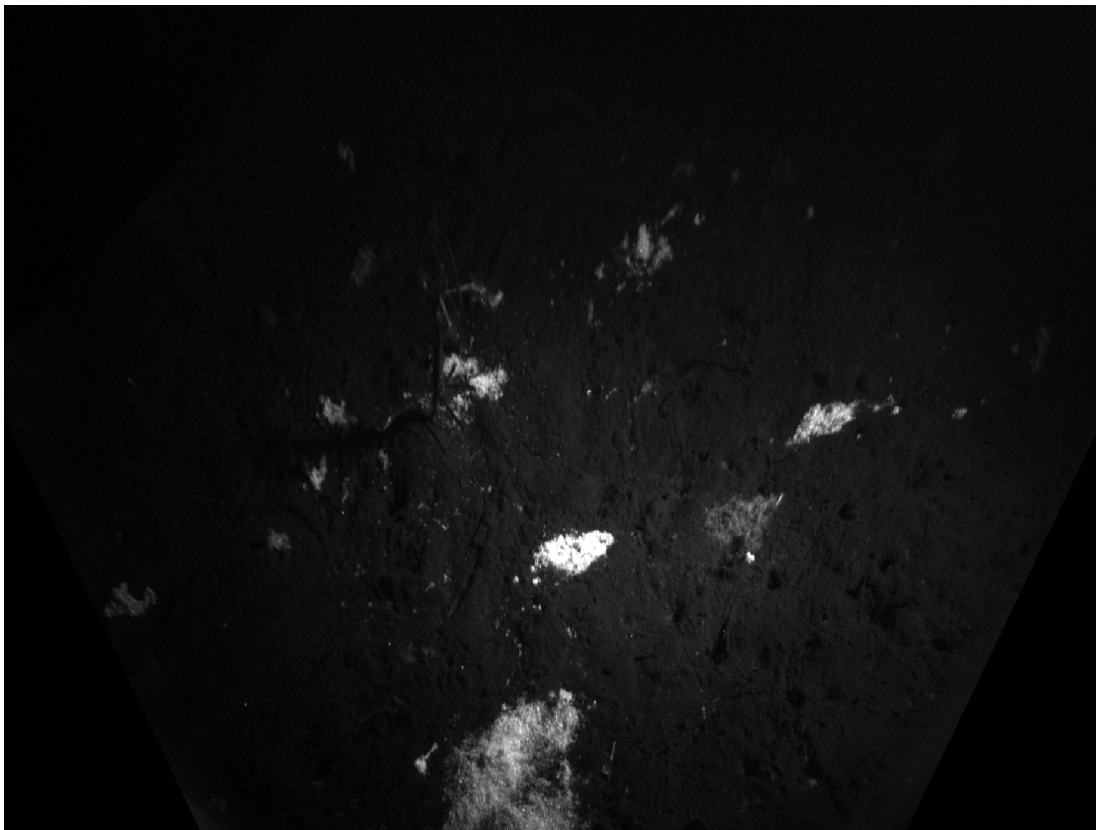
```
image2 = Import[  
  "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-  
    2019.10.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_200325_05_19_  
    22.tiff"]
```



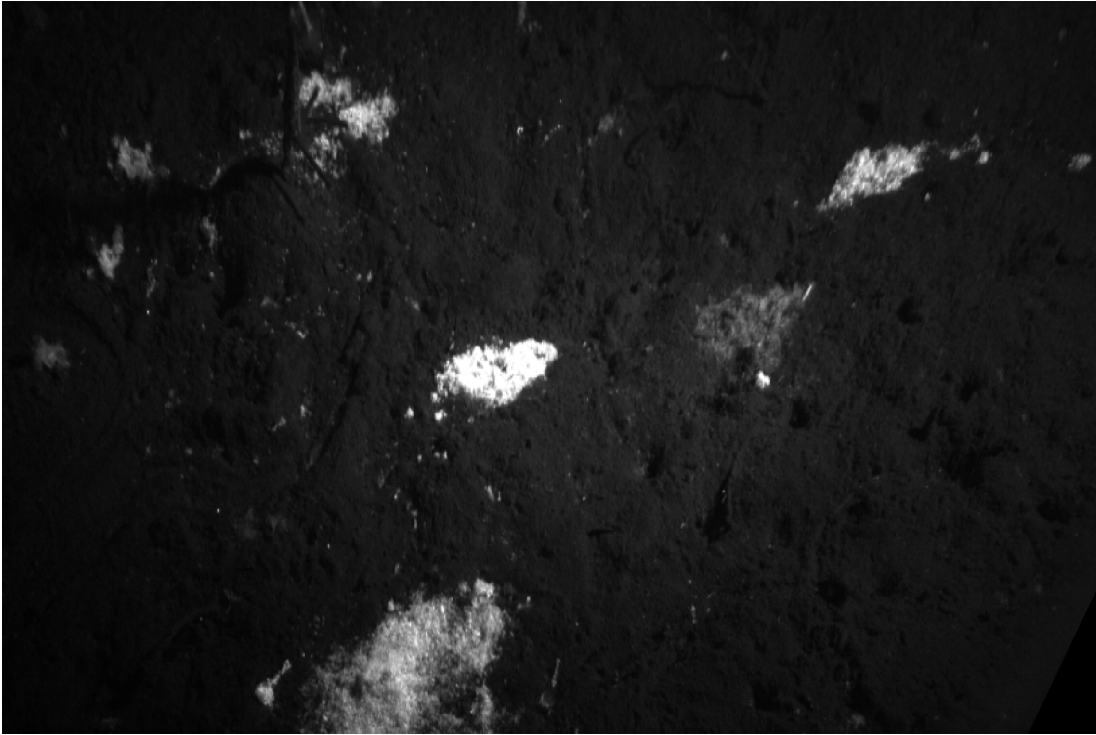
```
ImageTake[image2, -905]
```



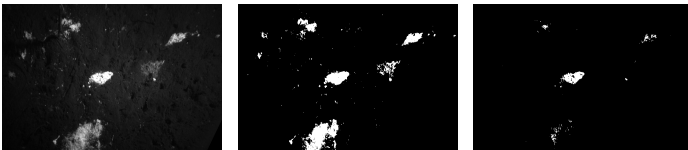
```
image2x = ImagePerspectiveTransformation[image2, xform, PlotRange -> Full]
```



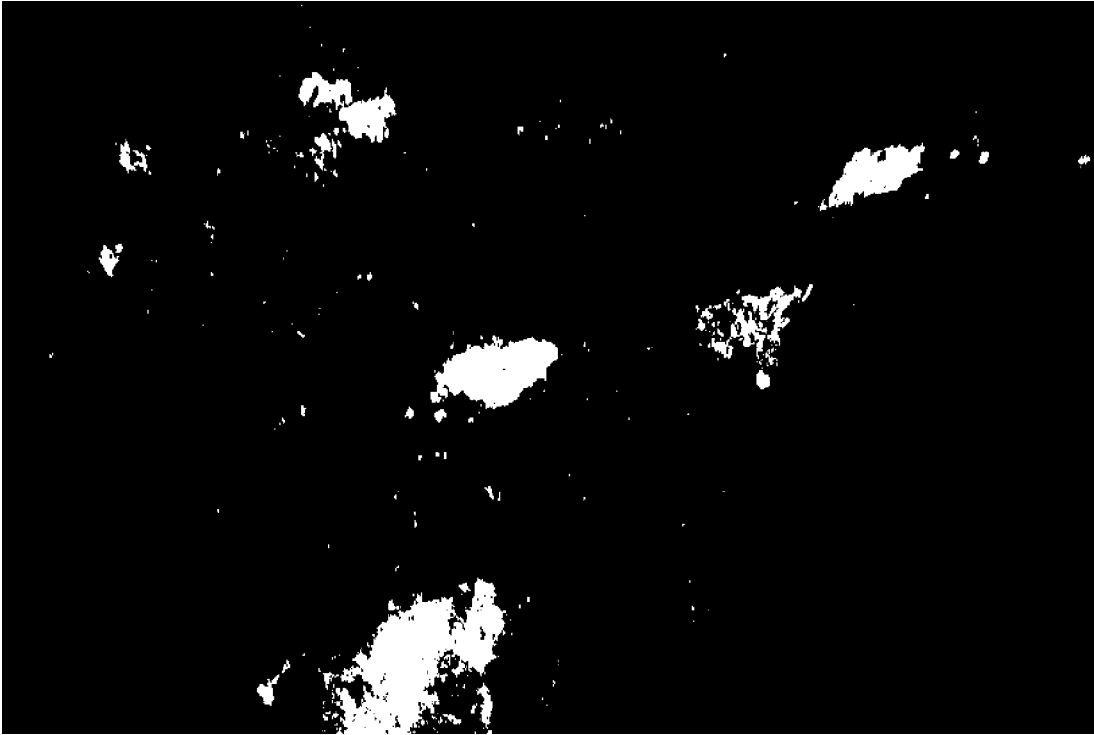
```
image2xSel = ImageTake[image2x, {imageH - ymax, imageH - ymin}, {xmin, xmax}]
```



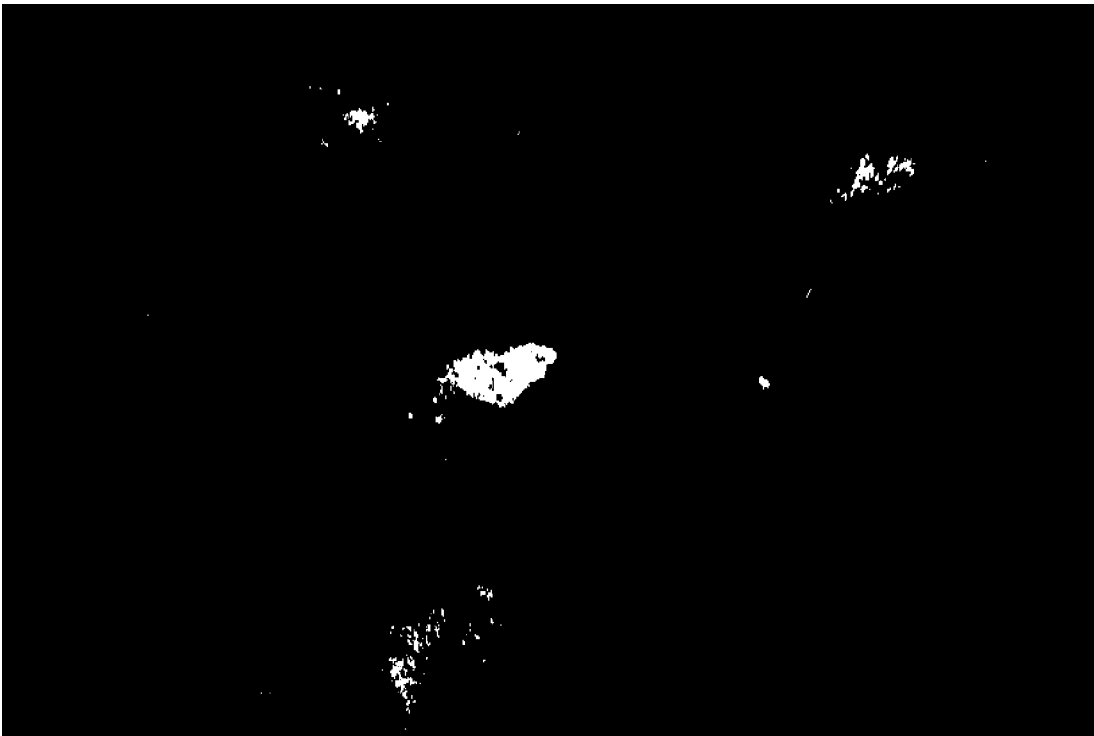
```
GraphicsGrid[{{image2xSel, Binarize[image2xSel, 0.3], Binarize[image2xSel, 0.7]}},]
```



`Binarize[image2xSel, 0.3]`



`Binarize[image2xSel, 0.7]`




```
phytCnt = Total[Flatten[ImageData[Binarize[image2xSel, 0.3]]]]
20831
```

```
frshCnt = Total[Flatten[ImageData[Binarize[image2xSel, 0.7]]]]
4098
```

```
pxlCnt = Apply[Times, ImageDimensions[image2xSel], {0}]
540000
```

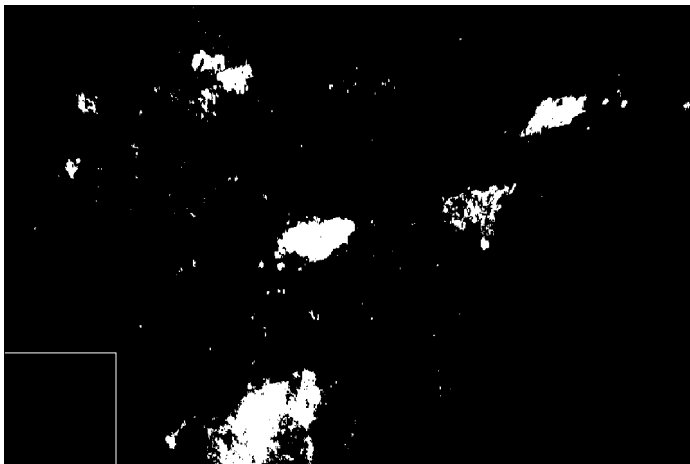
```
pxlCnt = Total[Flatten[ImageData[Binarize[image2xSel, 0.01]]]]
536129
```

```
phytFrac = phytCnt / pxlCnt // N
0.0388545
```

```
frshFrac = frshCnt / pxlCnt // N
0.00764368
```

Phytodetrital coverage looks like more than 3.8%, so overlay a 4% rectangle as a sanity check.

```
Show[{Binarize[image2xSel, 0.3],
  Graphics[{White, Line[{{0, 150}, {180 - 36, 150}, {180 - 36, 0}}]}]}
```



Automate the Analysis

Write a collection of functions to analyze a directory of fluorescence images and produce a time-series plot of phytodetrital coverage over time.

This extracts the date from a filename in the newer VARS format (YYMMDD_HH_MM_SS)

```

newExtractDate[filename_] := Module[{ymdstr, hmsstr, ymd, hms},
  ymdstr = StringTake[filename, {-20, -15}];
  hmsstr = StringTake[filename, {-13, -6}];
  ymd = DateList[ymdstr];
  hms = DateList[{hmsstr, {"Hour", "Minute", "Second"}}];
  DateList[
    {DateValue[ymd, "Year"], DateValue[ymd, "Month"], DateValue[ymd, "Day"],
     DateValue[hms, "Hour"], DateValue[hms, "Minute"], DateValue[hms, "Second"]}
  ] (* end newExtractDate[] *)

fnames[[1]]
/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-2019.10
.19-09-pulse-72/fluoro_images/StaM_7206_RoverFluoro_191019_20_04_38.tiff

newExtractDate[fnames[[1]]]
{2019, 10, 19, 20, 4, 38.}

```

This processes a single Rover fluorescence image and returns the areal fraction of both total and fresh phytodetritus.

```

processFluorImage[image_, xform_] := Module[{imagex, imagexSel,
  xmin = 301, xmax = 1200, ymin = 51,
  ymax = 650, imageH = 1024, phytCnt, frshCnt, pxlCnt},

  imagex = ImagePerspectiveTransformation[image, xform, PlotRange → Full];
  imagexSel = ImageTake[imagex, {imageH - ymax, imageH - ymin}, {xmin, xmax}];
  phytCnt = Total[Flatten[ImageData[Binarize[imagexSel, 0.3]]]];
  frshCnt = Total[Flatten[ImageData[Binarize[imagexSel, 0.7]]]];
  pxlCnt = Total[Flatten[ImageData[Binarize[imagexSel, 0.01]]]];
  {phytCnt, frshCnt}
  
$$\frac{\text{pxlCnt}}{\text{pxlCnt}} // N$$

] (* end processFluorImage[] *)

processFluorImage[image2, xform]
{0.0388545, 0.00764368}

```

This processes a directory of Rover fluorescence images and returns date-stamped measurements of both total and fresh phytodetritus.

```

processFluorDir[imageDir_] := Module[{fNames, xform,
  imagePts = {{0, 1024 / 5}, {1360, 1024 / 5}, {0, 1024 / 2}, {1360, 1024 / 2}},
  xformPts = {{150.51, 67.2134}, {1209.49, 67.2134}, {0, 512}, {1360, 512}},
  dateTime, i, datList = {}, phytFrac, frshFrac},
fNames = FileNames["StaM_*.tiff", imageDir];
xform = FindGeometricTransform[xformPts, imagePts][[2]];
Monitor[
  For[i = 1, i ≤ Length[fNames], i++,
    (* get date and time from file name *)
    dateTime = newExtractDate[fNames[[i]]];
    (* process an image *)
    {phytFrac, frshFrac} = processFluorImage[Import[fNames[[i]]], xform];
    (* add the data to the data list *)
    AppendTo[datList, {dateTime, {phytFrac, frshFrac}}];
  ], (* end For *)
  Framed@ProgressIndicator[i, {1, Length[fNames]}]
]; (* end Monitor *)
(* output the results *)
datList
] (* end processFluorDir *)

```

Process all the available Rover fluorescence images. The images themselves only have to be processed the first time and the results can be saved to dat files. For subsequent analyses the dat files can be imported directly, which is much faster than processing the images again. Comment/uncomment the relevant code section below.

```

(* p66dat=processFluorDir[
  "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_66/
  RoverPulse66/data/dep-2015.11.08-04-pulse-66/fluoro_images"];
p67dat=processFluorDir[
  "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_67/data/dep-
  -2016.11.12-05-pulse-67/fluoro_images"];
p69dat=processFluorDir[
  "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_69/dep-
  2017.11.11-02-pulse-69/fluoro_images"];
p71dat=processFluorDir[
  "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_71/logs/dep-
  -2018.10.19-03-pulse-71/fluoro_images"];
p72dat=processFluorDir[
  "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_72/dep-
  2019.10.19-09-pulse-72/fluoro_images"];
p75dat=processFluorDir[
  "/Volumes/ProjectLibrary/900502.BenthicRover/Rover.Deployment/Pulse_75/
  RoverPulse75/dep-2022.02.03-05-pulse-75/fluoro_images"]; *)

```

Length@p75dat

106

Save the results of each image analysis to a dat file for faster processing next time.

```
(* Save["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\ Data\
  Analysis/Fluorescence\ Imaging/Rover\ Fluor Pulse75.dat",p75dat] *)
```

Load the dat files directly instead of processing the images again.

```
Get["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ Fluor Pulse66.dat"];
Get["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ Fluor Pulse67.dat"];
Get["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ Fluor Pulse69.dat"];
Get["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ Fluor Pulse71.dat"];
Get["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ Fluor Pulse72.dat"];
Get["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ Fluor Pulse75.dat"];
```

```
fdat = p75dat;
```

```
dates = fdat[[All, 1]];
```

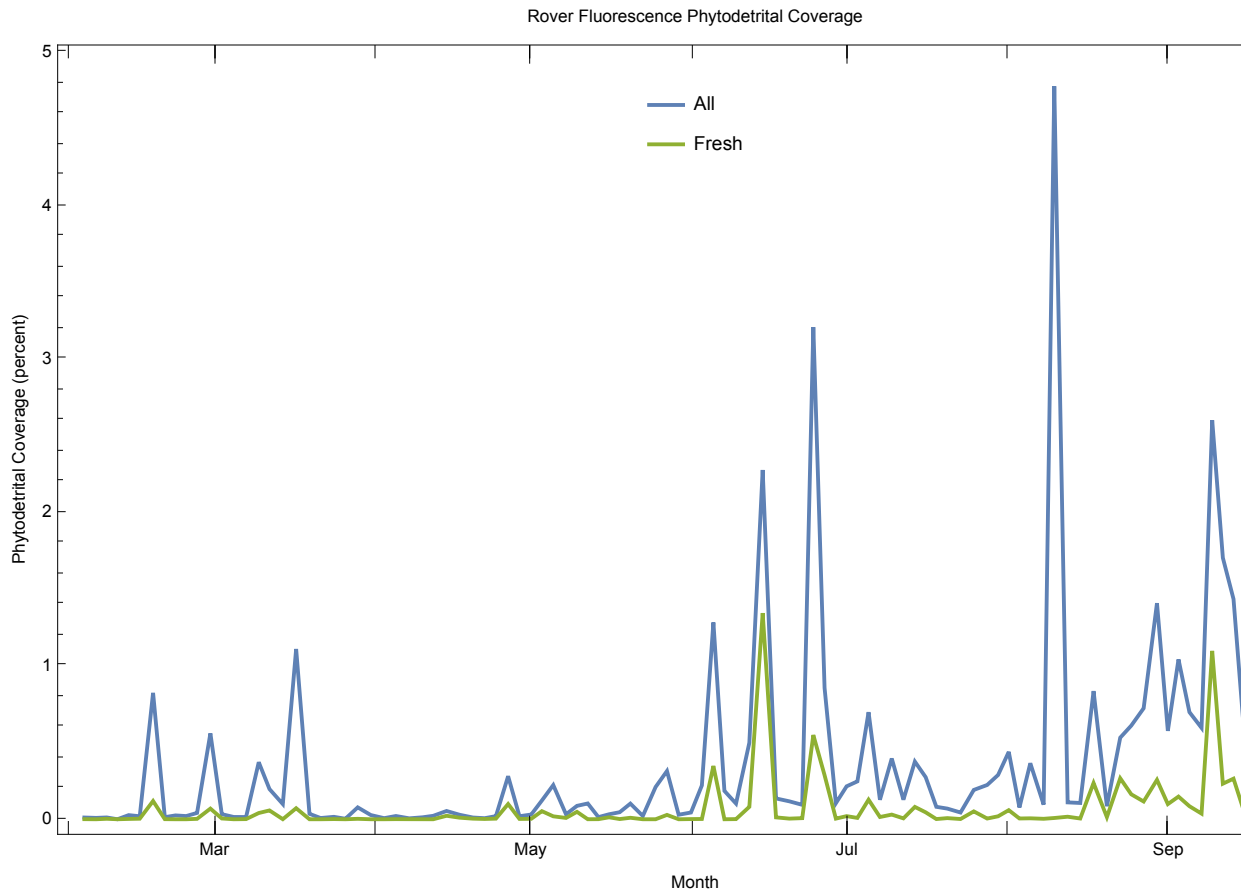
```
phyt = fdat[[All, 2, 1]] 100;
```

```
frsh = fdat[[All, 2, 2]] 100;
```

```
phytDat = Drop[Partition[Riffle[dates, phyt], 2], -1];
```

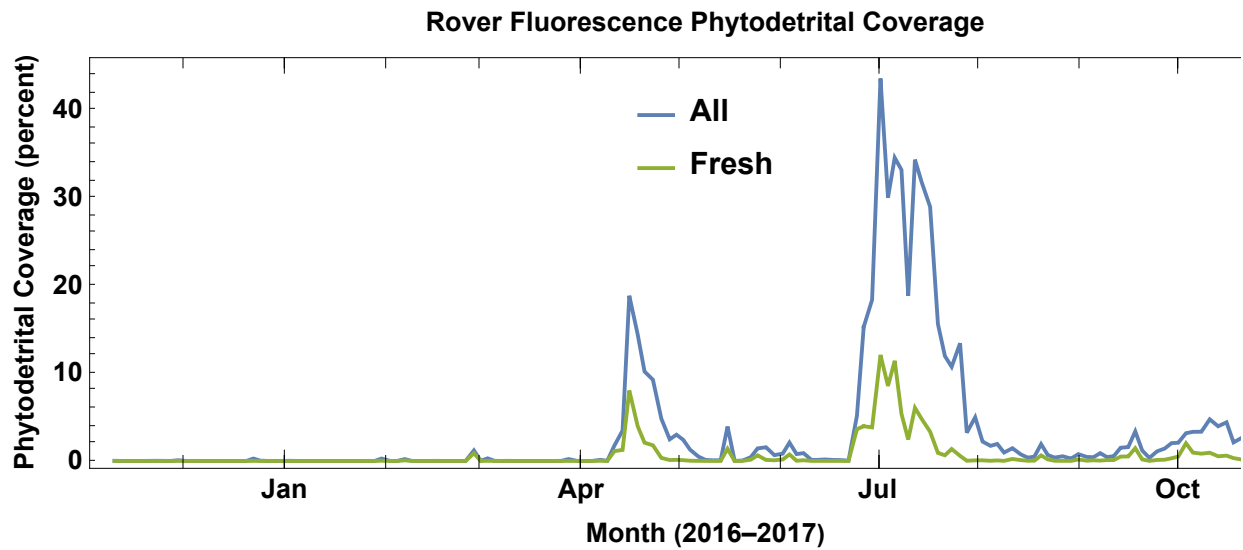
```
frshDat = Drop[Partition[Riffle[dates, frsh], 2], -1];
```

```
DateListPlot[{phytDat, frshDat}, PlotRange → All,
  FrameLabel → {"Phytodetrital Coverage (percent)", ""},
  {"Month", "Rover Fluorescence Phytodetrital Coverage"}},
  PlotLegends → Placed[{"All", "Fresh"}, {0.5, 0.9}],
  PlotStyle → {ColorData[97, "ColorList"][[1]], ColorData[97, "ColorList"][[3]]}]
```



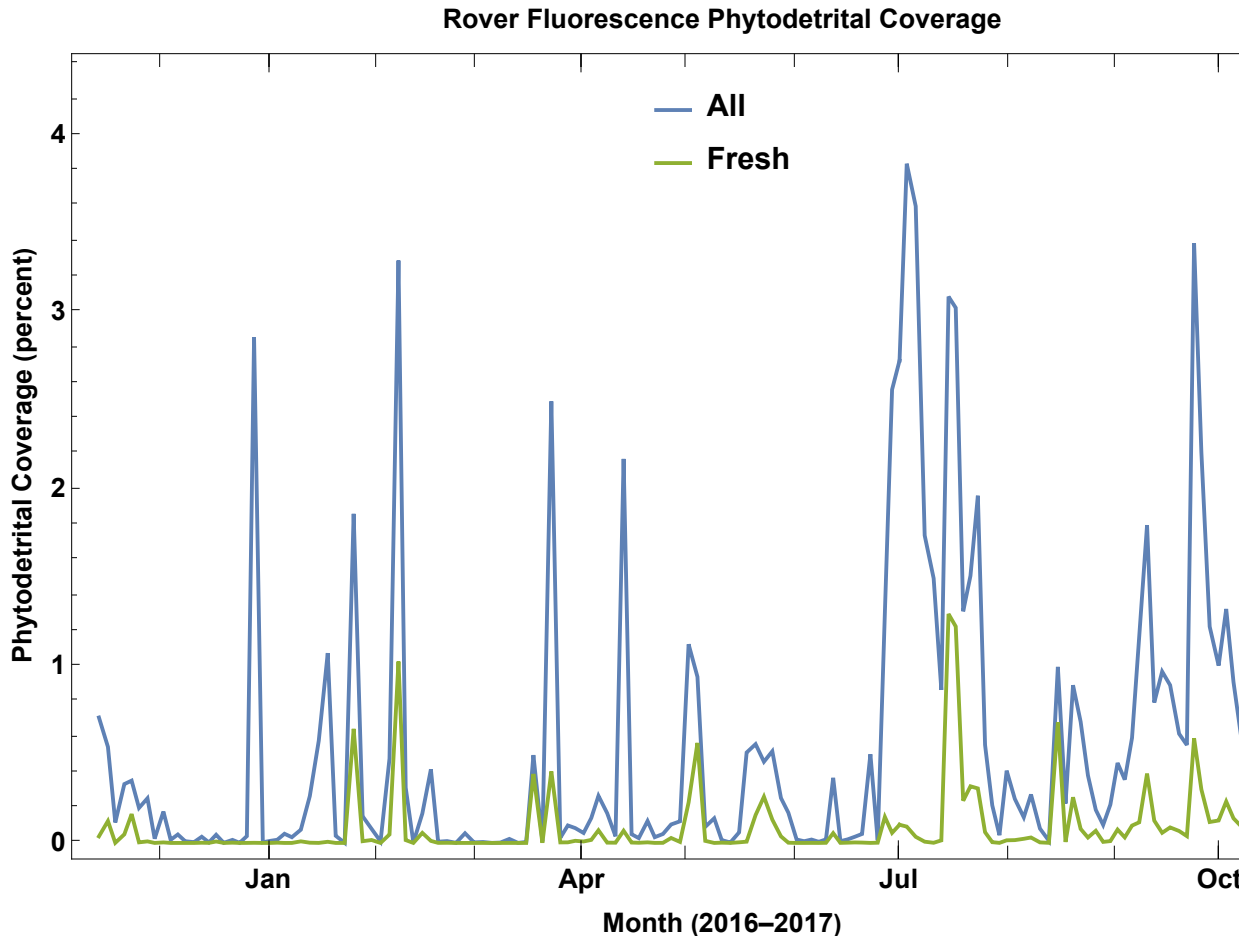
```
fdat = p66dat;
dates = fdat[[All, 1]];
phyt = fdat[[All, 2, 1]] 100;
frsh = fdat[[All, 2, 2]] 100;
phytDat = Drop[Partition[Riffle[dates, phyt], 2], -1];
frshDat = Drop[Partition[Riffle[dates, frsh], 2], -1];
```

```
DateListPlot[{phytDat, frshDat}, PlotRange → All,
  BaseStyle → {FontWeight → "Bold", FontSize → 14},
  FrameLabel → {"Phytodetrital Coverage (percent)", ""},
  {"Month (2016–2017)", "Rover Fluorescence Phytodetrital Coverage"}},
  PlotLegends → Placed[{"All", "Fresh"}, {0.5, 0.8}],
  PlotStyle → {{Thick, ColorData[97, "ColorList"][[1]]},
    {Thick, ColorData[97, "ColorList"][[3]]}}, AspectRatio → 1 / 3]
```



```
fdat = p67dat;
dates = fdat[[All, 1]];
phyt = fdat[[All, 2, 1]] 100;
frsh = fdat[[All, 2, 2]] 100;
phytDat = Drop[Partition[Riffle[dates, phyt], 2], -1];
frshDat = Drop[Partition[Riffle[dates, frsh], 2], -1];
```

```
DateListPlot[{phytDat, frshDat}, PlotRange → All,
  BaseStyle → {FontWeight → "Bold", FontSize → 14},
  FrameLabel → {{{"Phytodetrital Coverage (percent)", ""},
    {"Month (2016–2017)", "Rover Fluorescence Phytodetrital Coverage"}},
  PlotLegends → Placed[{"All", "Fresh"}, {0.5, 0.9}], PlotStyle →
    {{Thick, ColorData[97, "ColorList"][[1]]}, {Thick, ColorData[97, "ColorList"][[3]]}}
```



```
fdat = Join[p66dat, p67dat, p69dat, p71dat, p72dat];
dates = fdat[[All, 1]];
phyt = fdat[[All, 2, 1]] 100;
frsh = fdat[[All, 2, 2]] 100;
phytDat = Drop[Partition[Riffle[dates, phyt], 2], -1];
frshDat = Drop[Partition[Riffle[dates, frsh], 2], -1];
```

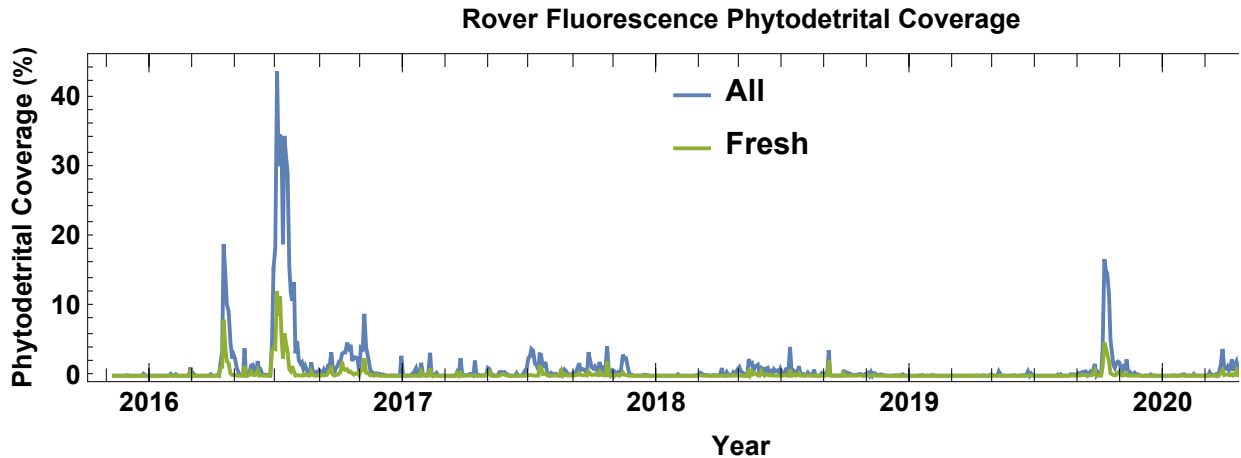
Save all the joined data to a CSV file for Crissy, who doesn't use Mathematica.

```
headers = {"DateTime", "AllPhytoCover", "FreshPhytoCover"};

Export["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ Fluor\ Data.csv",
  Join[headers, Transpose[{DateString[#, "ISODateTime"] & /@ dates, phyt, frsh}]]];
```

Plot all the data on a single plot.

```
DateListPlot[{phytDat, frshDat}, PlotRange → All,
  BaseStyle → {FontWeight → "Bold", FontSize → 14},
  FrameLabel → {"Phytodetrital Coverage (%)", ""},
  {"Year", "Rover Fluorescence Phytodetrital Coverage"}},
  PlotLegends → Placed[{"All", "Fresh"}, {0.5, 0.8}],
  PlotStyle → {{Thick, ColorData[97, "ColorList"][[1]]},
  {Thick, ColorData[97, "ColorList"][[3]]}}, AspectRatio → 1 / 4]
```



```
ColorData[97, "ColorList"]
```

```
{Blue, Orange, Green, Red, Purple, Brown, LightBlue, Yellow, Magenta, Olive, DarkRed, DarkBlue, DarkOrange, DarkMagenta, DarkGreen}
```

Compare With Other Station M Data

Find the dates covered by the Rover fluorescence data.

```
First@p66dat
```

```
{{2015, 11, 9, 10, 59, 35.}, {0.000118819, 0.0000482703}}
```

```
Last@p72dat
```

```
{{2020, 10, 27, 11, 33, 31.}, {0.00527847, 0.000790845}}
```

Import a trimmed dataset containing only that time period.

```
sdat = Import["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\
  Data\ Analysis/Fluorescence\ Imaging/Rover\ SCOC.csv", "CSV"];
```

```
First@sdat
```

```
{685, 11/8/2015, 11/9/2015, 2, 10.6131, 11/8/2015}
```

```
Last@sdat
```

```
{2178, 10/25/2020, 10/26/2020, 2, 12.185, 10/26/2020}
```



```
sdatdateval = sdat[[All, {6, 5}]];
```

```
scoc[[1]]
```

```
{{2015, 11, 8, 0, 0, 0.}, 10.6131}
```

```
scoc = {DateList[#[[1]], #[[2]]] & /@ sdatdateval;
```

```
... DateList: Warning: the interpretation of the string 11/8/2015 as a date is ambiguous.
```

```
... DateList: Warning: the interpretation of the string 11/9/2015 as a date is ambiguous.
```

```
... DateList: Warning: the interpretation of the string 11/12/2015 as a date is ambiguous.
```

```
... General: Further output of DateList::ambig will be suppressed during this calculation. ⓘ
```

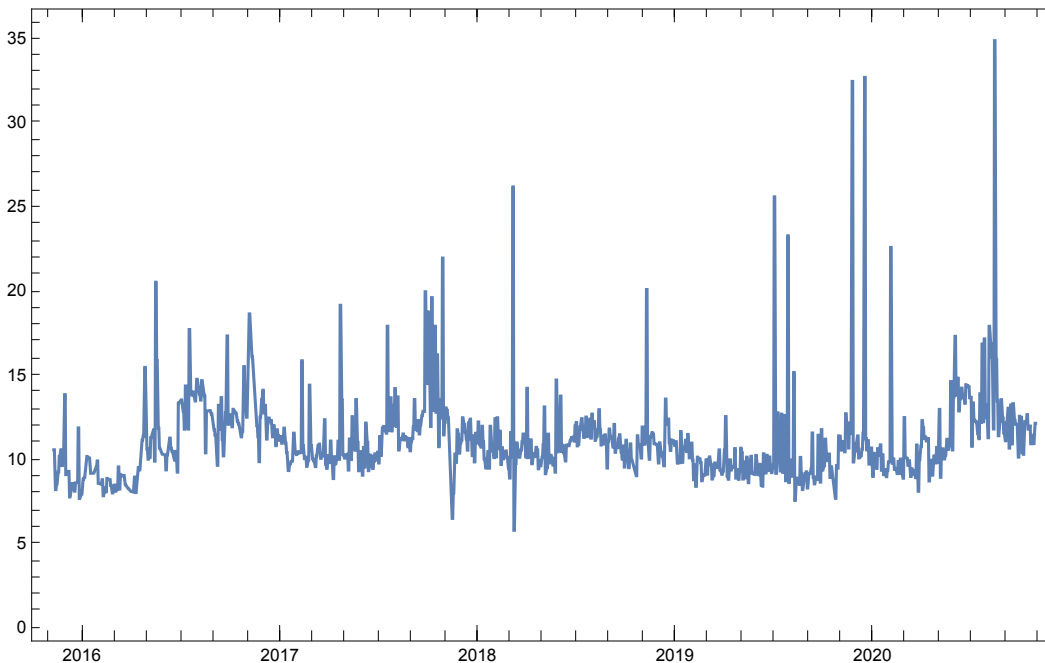
```
scoc[[1 ;; 10]]
```

```
{{{2015, 11, 8, 0, 0, 0.}, 10.6131}, {{2015, 11, 9, 0, 0, 0.}, 10.6131},  
{{2015, 11, 11, 0, 0, 0.}, 8.25054}, {{2015, 11, 12, 0, 0, 0.}, 8.25054},  
{{2015, 11, 13, 0, 0, 0.}, 8.58849}, {{2015, 11, 14, 0, 0, 0.}, 8.58849},  
{{2015, 11, 16, 0, 0, 0.}, 9.31399}, {{2015, 11, 17, 0, 0, 0.}, 9.31399},  
{{2015, 11, 18, 0, 0, 0.}, 10.2109}, {{2015, 11, 19, 0, 0, 0.}, 10.2109}}
```

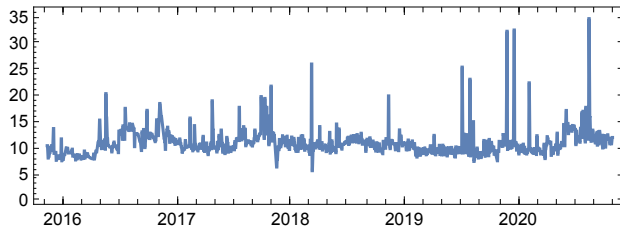
```
DateList[scoc[[1, 1]]
```

```
{2015, 11, 8, 0, 0, 0.}
```

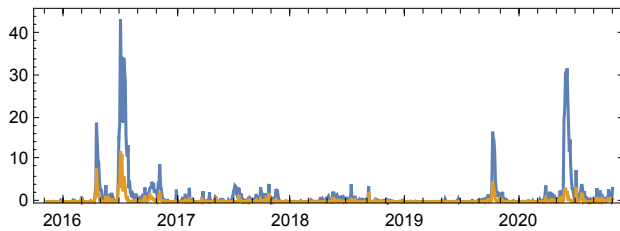
```
DateListPlot[scoc, PlotRange -> All]
```



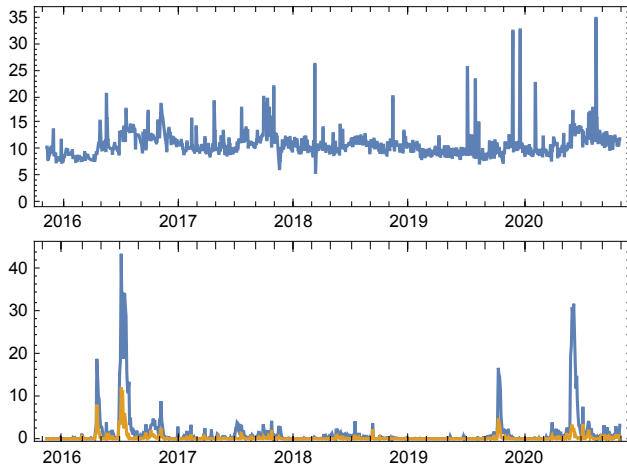
```
scoapl = DateListPlot[scoc, PlotRange → All, AspectRatio → 1 / 3, ImageSize → 320]
```



```
phypl = DateListPlot[{phytDat, frshDat},  
  PlotRange → All, AspectRatio → 1 / 3, ImageSize → 320]
```



```
GraphicsColumn[{scoapl, phypl}, Spacings → Scaled[0], ImageSize → 400]
```



Not much correlation between the data sets. Try POC.

```
carbdat = Drop[  
  Flatten[Import["/Users/mcgill/Documents/My\ Projects/Benthic\ Rover/Rover\  
    Data\ Analysis/Fluorescence\ Imaging/Station\  
    M\ Data/Rover\ Carbon.xlsx"], 1], 1];
```

```
First@carbdat
```

```
{ Tue 10 Nov 2015 00:00:00 GMT-7 , 21.8314, 4.50353 }
```

Last@carbdat

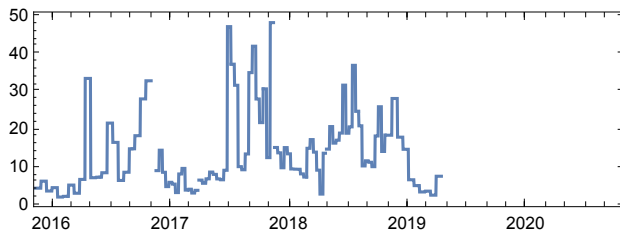
```
{ Sat 13 Apr 2019 00:00:00 GMT-7 , 34.8194, 7.6604 }
```

```
pocdat = carbdat[[All, {1, 3}]];
```

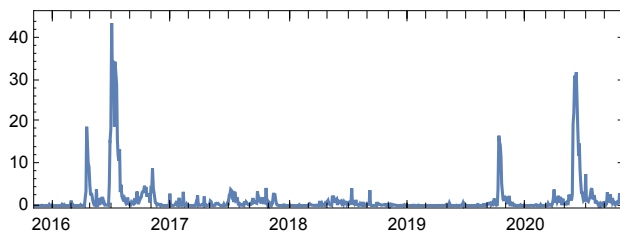
First@pocdat

```
{ Tue 10 Nov 2015 00:00:00 GMT-7 , 4.50353 }
```

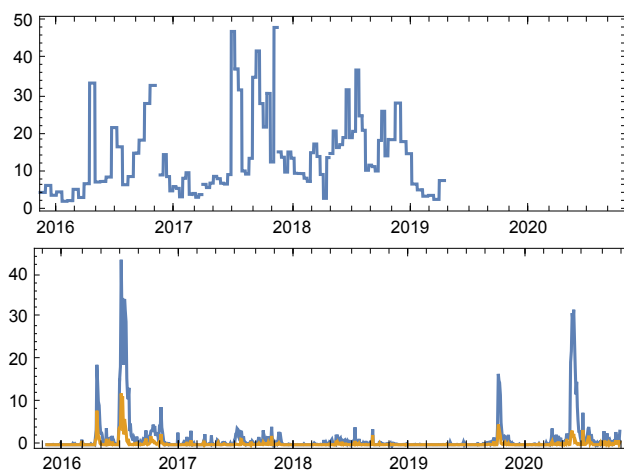
```
pocpl = DateListPlot[pocdat, PlotRange → {{p66dat[[1, 1]], p72dat[[-1, 1]]}, All},  
  AspectRatio → 1 / 3, ImageSize → 320]
```



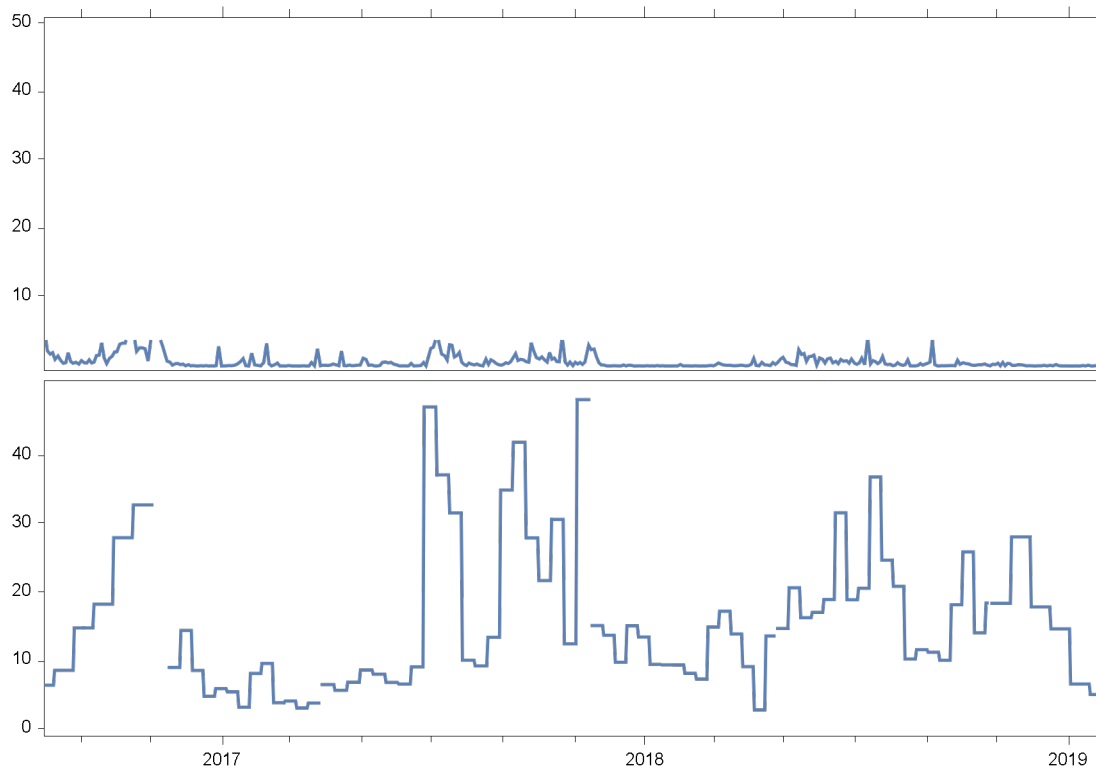
```
phylogpl = DateListPlot[phytDat, PlotRange → {{p66dat[[1, 1]], p72dat[[-1, 1]]}, All},  
  AspectRatio → 1 / 3, ImageSize → 320, ScalingFunctions → "Linear"]
```



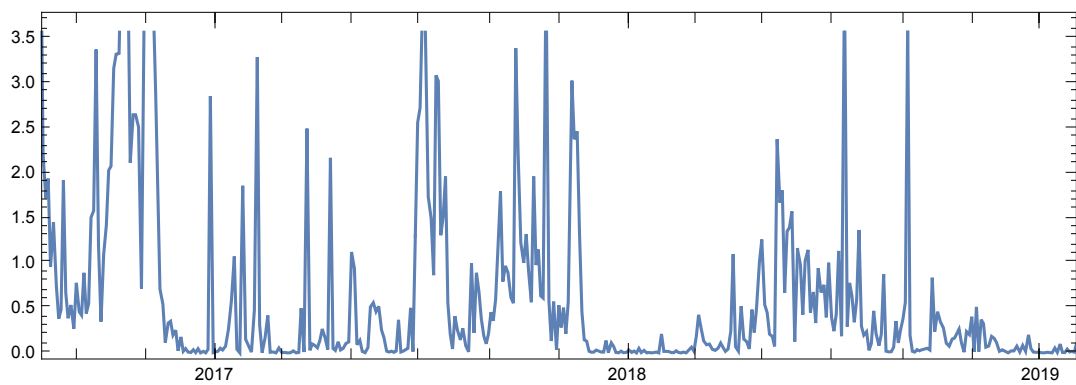
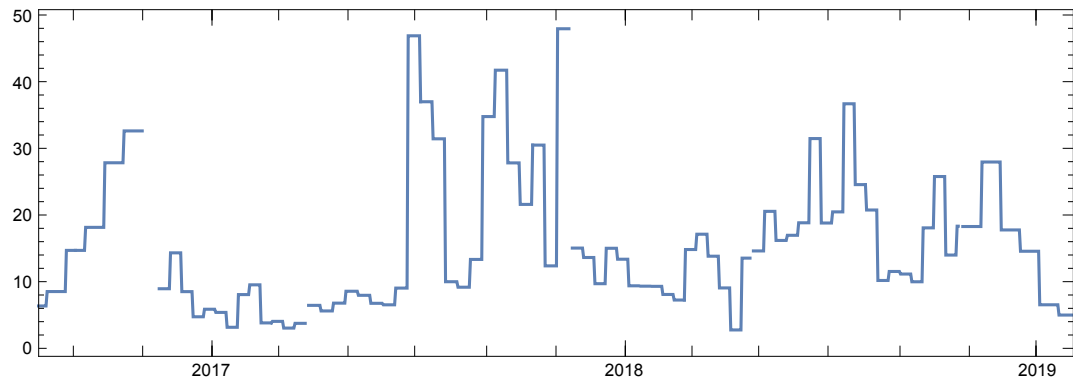
```
GraphicsColumn[{pocpl, phypl}, Spacings → Scaled[0], ImageSize → 400]
```



```
DateListPlot[{phytDat, pocdat},  
  PlotRange → {{{2016, 8, 1}, {2019, 2, 1}}, Automatic}, AspectRatio → 1 / 3,  
  PlotLayout → "Column", ScalingFunctions → {"Linear", "Linear"}, ImageSize → 600]
```

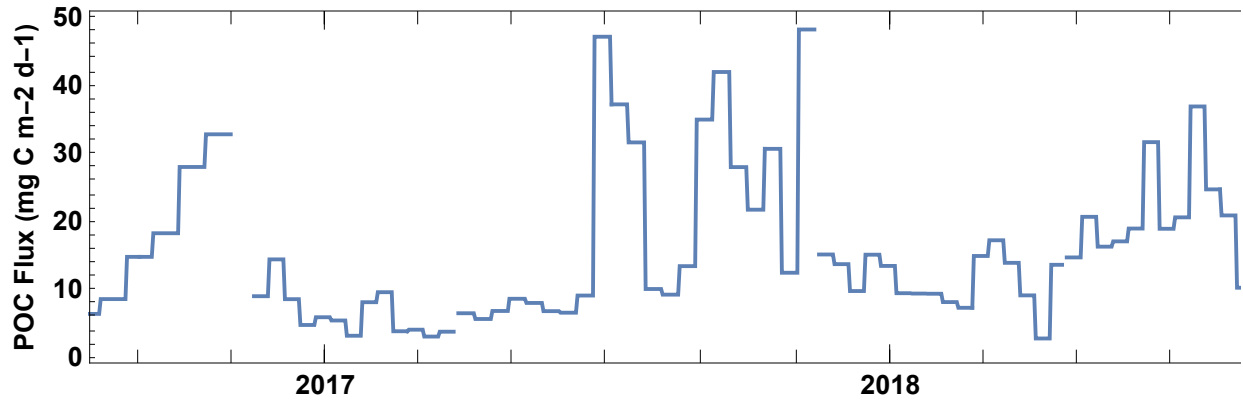


```
GraphicsColumn[  
  {DateListPlot[pocdat, PlotRange → {{{2016, 8, 1}}, {2019, 2, 1}}, Automatic},  
    AspectRatio → 1 / 3], DateListPlot[phytDat,  
    PlotRange → {{{2016, 8, 1}}, {2019, 2, 1}}, Automatic}, AspectRatio → 1 / 3]]
```

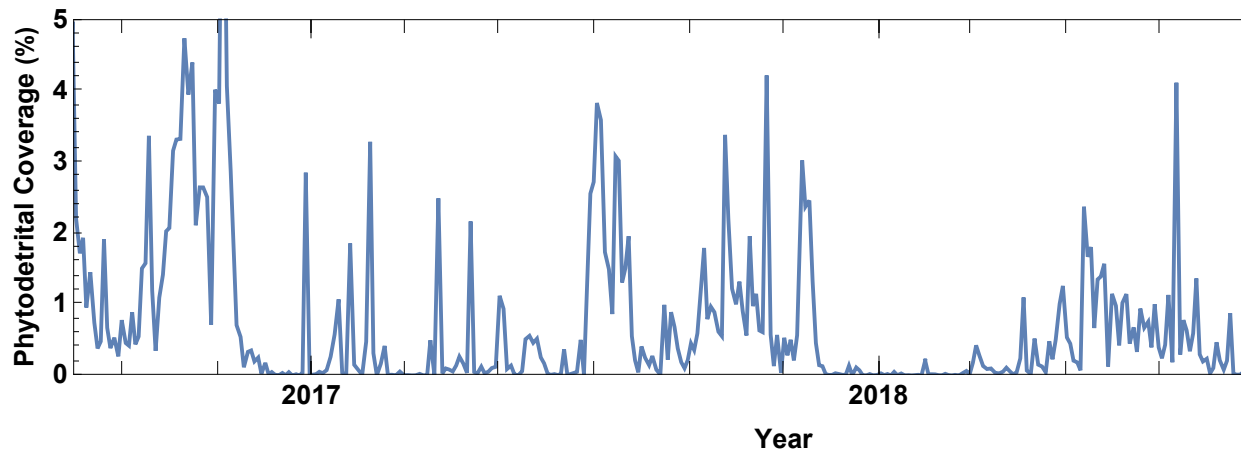


```
pocpl = DateListPlot[pocdat, PlotRange → {{{2016, 8, 1}, {2019, 2, 1}}, Automatic},
  AspectRatio → 1 / 4, BaseStyle → {FontWeight → "Bold", FontSize → 14},
  FrameLabel → {"POC Flux (mg C m-2 d-1)", ""},
  {"", Style["Particulate Organic Carbon Flux and Phytodetrital Coverage",
    Bold, 16]}}, ImageSize → 800, PlotStyle → Thick]
```

Particulate Organic Carbon Flux and Phytodetrital Coverage

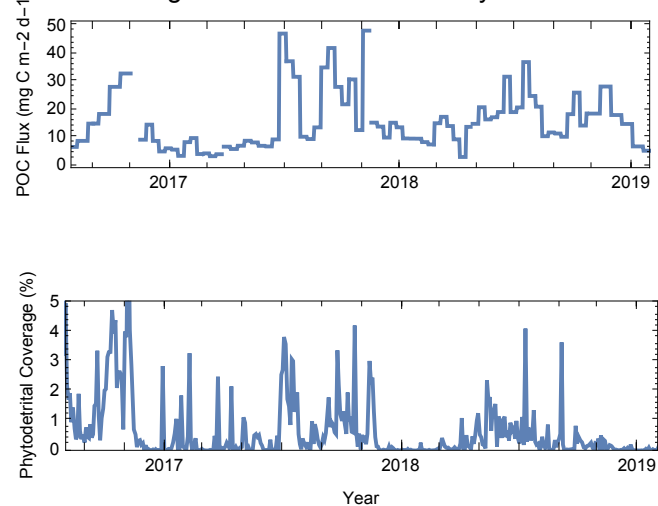


```
phytpl = DateListPlot[phytDat, PlotRange → {{{2016, 8, 1}, {2019, 2, 1}}, {0, 5}},
  AspectRatio → 1 / 4, BaseStyle → {FontWeight → "Bold", FontSize → 14},
  FrameLabel → {"Phytodetrital Coverage (%)", ""}, {"Year", ""},
  ImageSize → 795, PlotStyle → Thick]
```



```
GraphicsColumn[{pocpl, phypl}, Spacings → Scaled[0.1], ImageSize → 600]
```

Particulate Organic Carbon Flux and Phytodetrital Coverage



```
GraphicsColumn[
  {DateListPlot[pocdat, PlotRange → {{{2016, 8, 1}}, {2019, 2, 1}}, Automatic},
    AspectRatio → 1 / 4, FrameLabel → {{"POC Flux (mg C m-2 d-1)", ""}, {"", Style[
      "Particulate Organic Carbon and Phytodetrital Coverage", Bold, 16]}}],
  DateListPlot[phytDat, PlotRange → {{{2016, 8, 1}}, {2019, 2, 1}}, {0, 5}},
    AspectRatio → 1 / 4,
    FrameLabel → {"Phytodetrital Coverage (%)", ""}, {"Year", ""}], Spacings → 0]
```

Particulate Organic Carbon and Phytodetrital Coverage

