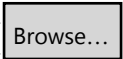

Initialize

```
$HistoryLength = 1;  
ClearAll["Global`*"];  
Off[General::spell, General::spell1]  
  
Needs["ErrorBarPlots`"];
```

Functions and PSF

Analysis

Reading in Maximum Projection Tif Images

```
{FileNameSetter[Dynamic[mymovie]], Dynamic[mymovie]}  
  
{, mymovie}
```

This will partition a three color movie, if the movie is not three colors then an error will occur

```
mymovie = If[StringTake[FileName[mymovie], 4] == "AVG_",  
  StringReplace[mymovie, "AVG_" -> ""], mymovie];  
myColMovie = Import[mymovie] // Partition[#, 3] & // Transpose;  
If[DirectoryQ[DirectoryName[mymovie] <> "Analysis\\"] == False,  
  CreateDirectory[DirectoryName[mymovie] <> "Analysis\\"]];  
analysisFolder = DirectoryName[mymovie] <> "Analysis\\" <>  
  StringDrop[StringDelete[mymovie, DirectoryName[mymovie]], -4];
```

This is making an average image of the movie to help make a mask

```
If[FileExistsQ[analysisFolder <> "_avg.tif"] == True,  
  avgImg = Import[analysisFolder <> "_avg.tif"];,  
  avgImg = Mean /@ myColMovie;  
  Export[(analysisFolder <> "_avg.tif"), avgImg];
```

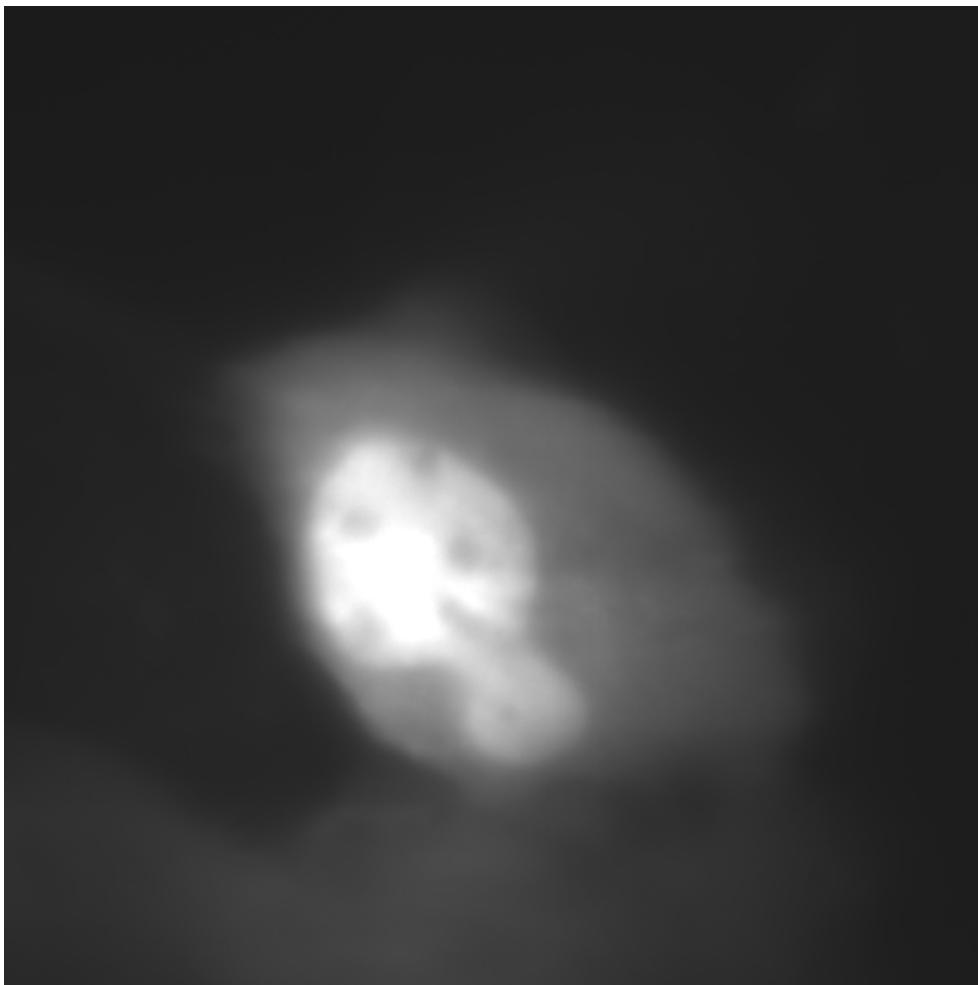
Make Masks of Cytoplasm and Nucleus

Choose a nice threshold first such that you can make out the shape of the cell


```
Manipulate[avgImg[2]] // ImageAdjust[#, {0, 0, 1}, {threshold1, threshold2}] &,  
  Button["Choose threshold such that the shape of the cell can be visualized and CLICK",  
    thtemp = {threshold1, threshold2}],  
  {threshold1, 0, 0.2, 0.001}, {{threshold2, 0.1}, 0, 0.2, 0.001}]
```

Run this next line to see your selected average image with the desired threshold. Click on the image and select the mask tool to create and copy the mask image below into the "mymask"

```
ImageAdjust[avgImg[[2]], {0, 0, 1}, thtemp]
```



This will export out the mask image to the analysis folder

```
mymask = ;
```

```
Export[analysisFolder <> "_mask.tif", mymask];
```

Creating Transformation Function (need to only run this once per imaging day to get a transformation function for that imaging day)

```
dirBeads = DirectoryName[mymovie] <> "Beads";
```

```

myFiles = FileNames["*.tif", dirBeads];
myBeadsImgs = Table[Import@myFiles[[i]], {i, 1, Length[myFiles], 1}];
myDims = ImageDimensions@myBeadsImgs[[1, 1]];
Manipulate[ColorCombine[
  {myBeadsImgs[[data, 1]], myBeadsImgs[[data, 2]], ConstantImage[0, myDims]}] // ImageAdjust,
  Button["Choose an image to calculate the transformation function and CLICK",
    nRefImg = data],
  {data, 1, Length[myFiles], 1}]

myRefBeadsRedImg = myBeadsImgs[[nRefImg, 1]];
myRefBeadsGreenImg = myBeadsImgs[[nRefImg, 2]];
Manipulate[GraphicsRow[{myRefBeadsRedImg // Binarize[#, thresholdRed] &,
  myRefBeadsGreenImg // Binarize[#, thresholdGreen] &}, ImageSize → 500],
  Button["Choose thresholds and CLICK", {thRed = thresholdRed, thGreen = thresholdGreen}],
  {thresholdRed, 0.01, 0.5, 0.005}, {thresholdGreen, 0.01, 0.5, 0.001}]

myRedPosGuess = ComponentMeasurements[ImageMultiply[myRefBeadsRedImg,
  Binarize[myRefBeadsRedImg, thRed]], {"IntensityCentroid"}][[All, 2, 1]];
myGreenPosGuess = ComponentMeasurements[ImageMultiply[myRefBeadsGreenImg,
  Binarize[myRefBeadsGreenImg, thGreen]], {"IntensityCentroid"}][[All, 2, 1]];
StringForm["  `` red particles and `` green particles were found.",
  Length@myRedPosGuess, Length@myGreenPosGuess]

20 red particles and 19 green particles were found.

```

```

myRedAbsPos = BeadsFitting[myRefBeadsRedImg, myRedPosGuess, 7];
myGreenAbsPos = BeadsFitting[myRefBeadsGreenImg, myGreenPosGuess, 7];
{myTfFuncError, myTfFunc, distanceRaw, distanceCorrected} =
  BeadsTransform[myRedAbsPos, myGreenAbsPos, 5];
{myTfFuncError, myTfFunc, Insert[{distanceRaw, distanceCorrected} // Transpose,
  {"Distance Before Correction", "Distance After Correction"}, 1] // TableForm}

```

General: $\frac{1.75426 \times 10^{-307}}{109.5}$ is too small to represent as a normalized machine number; precision may be lost.

General: $0.0013736^{109.5}$ is too small to represent as a normalized machine number; precision may be lost.

General: $\frac{5.06029 \times 10^{-307}}{109.5}$ is too small to represent as a normalized machine number; precision may be lost.

General: Further output of General::munfl will be suppressed during this calculation.

{0.107421, TransformationFunction[$\left(\begin{array}{cc|c} 1.00192 & -0.00474023 & 2.93625 \\ 0.00317464 & 1.00359 & -1.52968 \\ \hline 1.01473 \times 10^{-6} & 6.17355 \times 10^{-8} & 1. \end{array} \right)$],

Distance Before Correction	Distance After Correction
1.56028	0.15183
1.36684	0.124857
1.63938	0.020619
0.967872	0.0388495
1.35921	0.134548
1.42057	0.130699
1.01275	0.178449
1.41729	0.255172
1.82806	0.0858955
1.67689	0.0542119
1.80461	0.172144
2.41014	0.0343045
2.14091	0.213774
2.26445	0.0896564
2.95558	0.0135786
2.78519	0.0843578
3.21532	0.10184
3.1102	0.12743
3.13285	0.0287852

```
Export[DirectoryName[mymovie] <> "TransformationFunction.m", myTfFunc];
```

```
myTfFuncInv = InverseFunction@myTfFunc;
```

Running z-average (use this if you want to make a running average of the movie; I usually do not do a rolling average so I leave the avgFrameN = 1)

```
avgFrameN = 1;
myMaxN = Length[myColMovie[[1]]] - avgFrameN + 1;
myColMovieRunAv =
  Table[ImageMultiply[ImageAdd[myColMovie[[i, j ;; j + avgFrameN - 1]], 1 / avgFrameN],
    {i, 1, 3, 1}, {j, 1, myMaxN, 1}];
Manipulate[myColMovieRunAv[[channel, frame]] // ImageAdjust[#, {0, 0, 1}, {0, threshold}] &,
  {channel, 1, Length[myColMovieRunAv], 1},
  {frame, 1, myMaxN, 1}, {threshold, 0, 0.2, 0.001}];
```

Checking Transformation Function

Reading in Parameters (only if parameters exist)

Find Red Particles from Running z - averaged Movie (Red laser)

(* redRunAvp is list of {{x1,y1},{x2,y2}...{xn,yn}} in each frame *)

Adjust threshold to see spots nicely

```
Manipulate[
  myColMovieRunAv[[1, frame]] // ImageAdjust[#, {0, 0, 1}, {threshold1, threshold2}] &,
  Button["Choose threshold such that the dots appear nicely and CLICK",
    thred = {threshold1, threshold2}],
  {frame, 1, Length[myColMovieRunAv[[1]], 1}, {{threshold1, 0.001}, 0, 0.4, 0.001},
  {{threshold2, 0.03}, 0, 0.4, 0.001}]
```

Choose a threshold setting to where you can see nice well defined spots that match the left image

```
Manipulate[
  max =
    BandPass[ImageMultiply[myColMovieRunAv[[1, 1]] // ImageAdjust[#, {0, 0, 1}, thred] &,
      mymask], {lowpass, hipass}] // ImageData // Max;
  imgb = BandPass[myColMovieRunAv[[1, frame]] // ImageAdjust[#, {0, 0, 1}, thred] &,
    {lowpass, hipass}];
  GraphicsRow[{myColMovieRunAv[[1, frame]] // ImageAdjust[#, {0, 0, 1}, thred] &,
    SelectComponents[ImageMultiply[Binarize[imgb, max threshold], mymask],
      "Count", lo < # < hi &]], ImageSize -> 500],
  Button["Choose parameters and CLICK", {mymaxred = max,
    bandpassred = {lowpass, hipass}, thred2 = threshold, dotsizedred = {lo, hi}}],
  {frame, 1, myMaxN, 1}, {{threshold, 0.1}, 0, 0.4, 0.01}, {{lowpass, 1}, 0, 10, 1},
  {{hipass, 7}, 0, 10, 1}, {{lo, 5}, 0, 20, 1}, {{hi, 100}, 50, 200, 1}]

redRunAvp =
  Table[FindParticlesWeighted[myColMovieRunAv[[1, i]] // ImageAdjust[#, {0, 0, 1}, thred] &,
    bandpassred, mymaxred, thred2, mymask, dotsizedred], {i, 1, myMaxN, 1}];
StringForm[" `` red particles were found.", redRunAvp // Flatten[#, 1] & // Length]

66417 red particles were found.
```

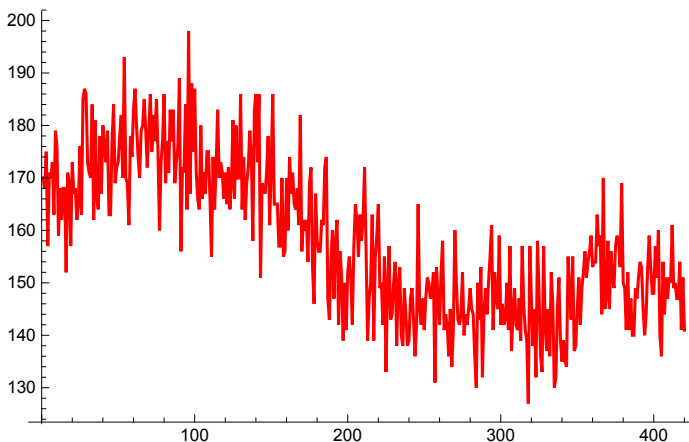
```
redComponentMask = Table[ImageTransformation[FindParticlesWeightedMask[
  myColMovieRunAv[[1, i]] // ImageAdjust[#, {0, 0, 1}, thred] &, bandpassred, mymaxred,
  thred2, mymask, dotsizedred], myTfFuncInv, DataRange → Full], {i, 1, myMaxN, 1}];
```

This will show you the red spots that are being found in each frame (this has not linked them up yet)

```
Manipulate[Show[myColMovieRunAv[[1, frame]] // ImageAdjust[#, {0, 0, 1}, thred] &,
  Graphics[{Red, Circle[#, 7] & /@ (redRunAvp[[frame]])}],
  Button["CLICK if you want to save this dataset",
    redRunAvp >> analysisFolder <> "_redspots.dat";
    Export[analysisFolder <> "_redComponentMask.tif", redComponentMask];],
  {frame, 1, myMaxN, 1}]
```

Just a plot to visualize the number of spots per frame

```
rSpotNPlot = Length /@ redRunAvp // ListLinePlot[#, PlotStyle → Red] &
```



Linking red tracks

This will link the tracks; set the pixel jump size that you will allow to link red spots; you can get more artifactual linked tracks if the maxJumpDistRed is higher than 5; having it too low will cause you to have partial/fragmented tracks

```
maxJumpDistRed = 5;
my488and561Interval = 2;

myredtracks = LinkTracks2[redRunAvp, maxJumpDistRed];
StringForm["`` red tracks were found by allowing
  `` pixel jump, and the average length of the tracks is ``",
  myredtracks // Length, maxJumpDistRed, (Length /@ myredtracks) // Mean // N]
4661 red tracks were found by allowing 5 pixel
  jump, and the average length of the tracks is 7.715726239004505`

myredtracks >> analysisFolder <> "_redtracks.dat";
```

This will filter out shorter tracks, I find that anything >40 helps to ensure you are tracking real RNA and not spot noise

```

shortestTrackRed = 41;

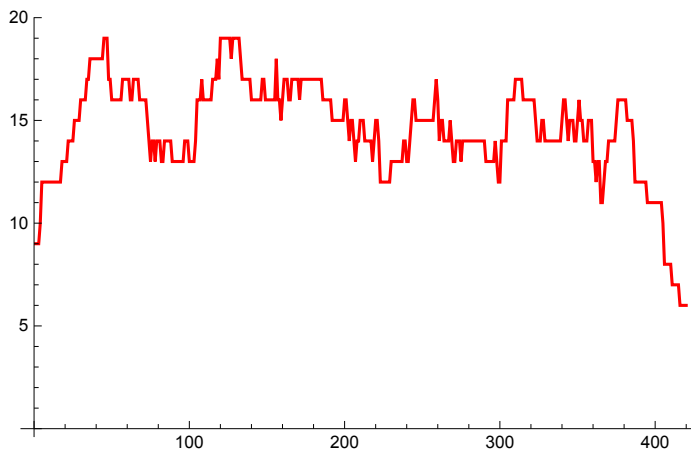
mylongredtracks = myredtracks // DeleteCases[#, x_ /; Length[x] < shortestTrackRed] &;
StringForm[
  "  `` red tracks are longer than ``, and the average length of the long tracks is ``",
  mylongredtracks // Length, shortestTrackRed, (Length /@mylongredtracks) // Mean // N]

  90 red tracks are longer than 41, and
  the average length of the long tracks is 67.91111111111111`

mylongredtracksf =
  mylongredtracks[[All, All, 2]] // Flatten[#, 1] & // Sort // SplitBy[#, First] &;
Manipulate[Show[myColMovieRunAv[[1, frame]] // ImageAdjust[#, {0, 0, 1}, thred] &, Graphics[
  {Red, Circle[#, 7] & /@ (mylongredtracksf[[frame, All, 2 ;; 3]])}], {frame, 1, myMaxN, 1}];

rSpotNPlot = Length /@mylongredtracksf // ListLinePlot[#, PlotStyle -> Red] &

```



Check all tracks (and double-check transformation function)

```

mylongredtracksColReg = mylongredtracks;
mylongredtracksColReg[[All, All, 2, 2 ;;]] =
  myTfFunc /@ mylongredtracksColReg[[All, All, 2, 2 ;;]];
Clear[mylongredtracksColRegf]
mylongredtracksColRegf =
  mylongredtracksColReg[[All, All, 2]] // Flatten[#, 1] & // Sort // SplitBy[#, First] &;

```

Linking red, blue, and green tracks

```

tracksr = Map[Append[#, {1, 0, 0}] &, mylongredtracksColReg, {2}];
tracksr[[All, All, 1]] = Range@Length@tracksr;

```

Fitting Red Tracks with Gaussian and Making Parameter File

Setting Parameters

```
pad = 5;
myTrimSize = 2 * pad + 1;
```

Fitting red tracks with Gaussian

This is doing an initial gaussian fit of the red spots and their positions; I use this information for the positions of the spots below

```
{myRedAllParameters, myRedPos, myRedData} =
  ParticleFittingFixedSigma[myColMovie, 1, tracksr, pad, myTrimSize, 1.5, avgFrameN];
```

... NonlinearModelFit: Failed to converge to the requested accuracy or precision within 100 iterations.

... NonlinearModelFit: Failed to converge to the requested accuracy or precision within 100 iterations.

... NonlinearModelFit: Failed to converge to the requested accuracy or precision within 100 iterations.

... General: Further output of NonlinearModelFit::cvmit will be suppressed during this calculation.

```
{myRedAllParameters, myRedPos, myRedData} >> analysisFolder <> "_redfits.dat";
```

```
{myRedAllParameters, myRedPos, myRedData} = << (analysisFolder <> "_redfits.dat");
```

```
myRedAllParametersColReg = myRedAllParameters;
```

```
(*myRedAllParametersColReg[[All,3;;4]] = myTfFunc / @myRedAllParameters[[All,3;;4]]);
```

```
mylabel = {"TrackN", "FrameN", "X (130 nm pixels)", "Y (130 nm pixels)",
  "Peak intensity", "Background intensity", "SigmaX (130 nm pixels)",
  "SigmaY (130 nm pixels)", "X min 90% confidence interval (130 nm pixels)",
  "X max 90% confidence interval (130 nm pixels)",
  "Y min 90% confidence interval (130 nm pixels)",
  "Y max 90% confidence interval (130 nm pixels)",
  "Peak intensity min 90% confidence interval",
  "Peak intensity max 90% confidence interval",
  "Background intensity min 90% confidence interval",
  "Background max 90% confidence interval",
  "SigmaX min 90% confidence interval (130 nm pixels)",
  "SigmaX max 90% confidence interval (130 nm pixels)",
  "SigmaY min 90% confidence interval (130 nm pixels)",
  "SigmaY max 90% confidence interval (130 nm pixels)", "R", "G", "B"};
```

```
myRedAllParametersColRegLabeled = Insert[myRedAllParametersColReg, mylabel, 1];
```

```
Export[analysisFolder <> "_FixedSigmaRedAllParameters.xls",
  myRedAllParametersColRegLabeled];
```

```
myRedAllParametersColReg >> (analysisFolder <> "_FixedSigmaRedAllParameters.m");
```

Making RGB Trims from Red Tracks

This reads in the movie again if you need to; if it is already defined from running the code above then you do not need to run this

```
{FileNameSetter[Dynamic[mymovie]], Dynamic[mymovie]}
```

```
{Browse..., mymovie}
```

This will read in the movie and partition it as it does above; doesn't need to be done if you have already defined it from above

```
mymovie = If[StringTake[FileName[mymovie], 4] == "AVG_",
  StringReplace[mymovie, "AVG_" -> ""], mymovie];

myColMovie = Import[mymovie] // Partition[#, 3] & // Transpose;
If[DirectoryQ[DirectoryName[mymovie] <> "Analysis\\"] == False,
  CreateDirectory[DirectoryName[mymovie] <> "Analysis\\"]];
analysisFolder = DirectoryName[mymovie] <> "Analysis\\" <>
  StringDrop[StringDelete[mymovie, DirectoryName[mymovie]], -4];
```

Reading in all saved parameters

Reading in parameters from the above cell

```
analysisFolder = DirectoryName[mymovie] <> "Analysis\\" <>
  StringDrop[StringDelete[mymovie, DirectoryName[mymovie]], -4];
If[FileExistsQ[StringReplace[mymovie, "MAX" -> "AVG_MAX"]] == True,
  avgImg = Import[StringReplace[mymovie, "MAX" -> "AVG_MAX"]];,
  avgImg = Mean /@ myColMovie];;
myavgImg = StringReplace[mymovie, "MAX" -> "AVG_MAX"];

mymask = Import[analysisFolder <> "_mymask.dat"];
myavgImg = StringReplace[mymovie, "MAX" -> "AVG_MAX"];
avgFrameN = IntegerPart[avgFrameN];
myTfFunc = Import[DirectoryName[mymovie] <> "TransformationFunction.m"];
myTfFuncRev = InverseFunction@myTfFunc;
myMaxN = Length[myColMovie[[1]]] - avgFrameN + 1;
myColMovieRunAv =
  Table[ImageMultiply[ImageAdd[myColMovie[[i, j ;; j + avgFrameN - 1]], 1 / avgFrameN],
    {i, 1, 3, 1}, {j, 1, myMaxN, 1}];
{myRedAllParameters, myRedPos, myRedData} = << (analysisFolder <> "_redfits.dat");
myRedAllParametersColReg = myRedAllParameters;
myRedAllParametersColReg[[All, 3 ;; 4]] =
  Table[myRedAllParameters[[i, 3 ;; 4]], {i, 1, Length[myRedAllParameters], 1}];
myRedParameterColRegSplit = myRedAllParametersColReg // SplitBy[#, First] &;
```

```

avgImg = Mean /@ myColMovie;
avgColImg = ColorCombine[
  ImageAdjust /@ {avgImg[[1]], avgImg[[2]], ConstantImage[0, ImageDimensions@avgImg[[1]]]};
myRedParameterSplit = myRedAllParameters // SplitBy[#, First] &;
(*Grouping tracks together from Parameter list*)
RedCentroid = Thread[{Range[Length[myRedParameterSplit]], Table[
  myRedParameterSplit[[i, All, 3 ;; 4]] // Mean, {i, 1, Length[myRedParameterSplit], 1}]]];
(*Taking the mean position of each grouped tracks*)
posRedLongTracks = Graphics[{Red, Circle[#, 7] & /@ (RedCentroid[[All, 2]]),
  Table[Inset[RedCentroid[[i, 1]], RedCentroid[[i, 2]], {i, 1, Length[RedCentroid], 1}]]];
(*Making dumb little circles around the average position of each tracks*)
(*Manipulate[Show[myColMovieRunAv[[1, start-1+frame]]//ImageAdjust[#, {0,0,1}, thred]&,
  posRedLongTracks], {frame, start, end, 1}]*

```

Making Red Track Trims with either Blue and Green aligned to center of pad

```

myRedParameterSplitColReg = myRedAllParametersColReg // SplitBy[#, First] &;
myRedParameterSplitColReg >> analysisFolder <> "_myRedParameterSplitColReg.m"

pad2 = 7;
ClearAll[myRedParameterSplitColReg, TrimRed, TrimGreen, TrimBlue, Trim3Colors]
myRedParameterSplitColReg = myRedAllParametersColReg // SplitBy[#, First] &;
{TrimRed, TrimGreen, TrimBlue, Trim3Colors} = Table[Table[
  tempTime = myRedParameterSplit[[myi, i, 2]];
  tempCenter = Ceiling[(myRedParameterSplit[[myi, i, 3 ;; 4]])];
  tempCenterColReg = Ceiling[myTffunc@myRedParameterSplitColReg[[myi, i, 3 ;; 4]]];
  {tempRed, tempGreen, tempBlue} = {
    ImageTrim[myColMovie[[1, tempTime + Ceiling[avgFrameN/2] - 1]],
    {tempCenter - pad2, tempCenter + pad2 - 0.1}],
    ImageTrim[myColMovie[[2, tempTime + Ceiling[avgFrameN/2] - 1]],
    {tempCenterColReg - pad2, tempCenterColReg + pad2 - 0.1}],
    ImageTrim[myColMovie[[3, tempTime + Ceiling[avgFrameN/2] - 1]],
    {tempCenterColReg - pad2, tempCenterColReg + pad2 - 0.1}]];
  {tempRed, tempGreen, tempBlue, ColorCombine[{ImageAdjust@tempRed,
    ImageAdjust@tempGreen, ImageAdjust@tempBlue}]},
  {i, 1, Length[myRedParameterSplit[[myi]], 1]} // Transpose,
  {myi, 1, 1 Length[myRedParameterSplit], 1}] // Transpose;

Thread[{TrimRed, TrimGreen, TrimBlue}] >> analysisFolder <> "_myRedTrackRGBTrims.m";

```

Hand Checking all red tracks

Define all the myDirectories below here, only put the directories that have the same construct and the same imaging conditions that you want to handcheck below. Always change the my

```
myDirectories =
  {("P:\\20190920_u2os_multiplex\\smFLAG-KDM5B-GBInterval_2\\Analysis" <> "\\")};
(*Input all analysis folder pathways here that already have the RedTrackRGBTrims
from above and that are the same construct and same imaging intervals,
make sure to add a \\ after the Analysis folder when inputting new folders*)
```

Showing the number of myRedTrackRGBTrims (cells) you have in this directory

```
Flatten@Table[FileNames[myDirectories[[directory]] <> "*_myRedTrackRGBTrims.m"],
  {directory, 1, Length@myDirectories, 1}]
{P:\\20190920_u2os_multiplex\\smFLAG-KDM5B-GBInterval_2\\Analysis\\MAX_Cell01
_myRedTrackRGBTrims.m,
 P:\\20190920_u2os_multiplex\\smFLAG-KDM5B-GBInterval_2\\Analysis\\MAX_Cell02
_myRedTrackRGBTrims.m}
```

Getting the file names of each cell in the directory and then uploading the RedRGBParameters made above

```
myRedRGBTrimsAllCellsFNs =
  Flatten@Table[FileNames[myDirectories[[directory]] <> "*_myRedTrackRGBTrims.m"],
    {directory, 1, Length@myDirectories, 1}];
myRedRGBParameterAllCellsFNs = Table[StringTake[myRedRGBTrimsAllCellsFNs[[cell]],
  (StringPosition[myRedRGBTrimsAllCellsFNs[[cell]], "my"] // Min) - 2] <>
  "_myRedParameterSplitColReg.m", {cell, 1, Length@myRedRGBTrimsAllCellsFNs, 1}];
myRedParameterSplitColReg = Get[#] & /@ myRedRGBParameterAllCellsFNs;
```

Counting the number of myRedParametersSplitColReg files, should be the same number as there are cells

```
Length@myRedParameterSplitColReg
```

```
2
```

Looking at the length of each track, each number is the length of the track for each spot (should all be over >40 in length)

```
Length /@ myRedParameterSplitColReg[[1]]
{239, 62, 126, 47, 99, 61, 73, 47, 108, 85, 68, 45, 50, 46, 49, 48, 46, 46, 88, 42, 44, 76, 89,
 43, 42, 54, 86, 60, 70, 166, 54, 118, 42, 94, 56, 68, 44, 136, 71, 132, 66, 51, 50, 43, 42,
 52, 56, 53, 50, 46, 218, 57, 53, 59, 72, 44, 48, 45, 91, 57, 101, 51, 42, 65, 84, 133, 56,
 43, 89, 84, 92, 55, 53, 64, 60, 43, 50, 55, 81, 46, 41, 56, 65, 48, 43, 54, 43, 51, 46, 45}
```

Reading in the trims made above from each cell in this directory

```
myRedTrackRGBTrimsAllCells = Get[#] & /@ myRedRGBTrimsAllCellsFNs;
```

```
Length /@ myRedTrackRGBTrimsAllCells[[1, All, 1]] (*Is in the same order as up top
need to make these both compatible with reading in multiple directories!*)
```

```
{239, 62, 66, 47, 99, 43, 40, 47, 49, 85, 45, 46, 43, 53, 53, 49, 90, 65, 43, 42,
 59, 67, 54, 86, 41, 68, 73, 58, 42, 166, 127, 43, 118, 57, 68, 49, 44, 155, 66,
 54, 51, 51, 50, 50, 57, 56, 46, 40, 176, 58, 53, 59, 41, 53, 48, 45, 91, 57, 101,
 92, 42, 73, 61, 59, 69, 43, 56, 49, 45, 78, 49, 57, 40, 48, 47, 58, 56, 44, 41}
```

Counting the number of myRedTrackRGBTrimsAllCells files, should be the same number as there are cells

```
Length@myRedTrackRGBTrimsAllCells
```

```
2
```

```
(*my488and561Interval=2;*)
(*myRedTrackRGBGaussIntAllCells=Table[
  If[cell>2,my488and561Interval=3,my488and561Interval=2];
  GetTrimPeakIntFixedSigma95Conf[#,1.5]&/@myRedTrackRGBTrimsAllCells[[cell,spot,All,
    1;;All;;my488and561Interval]],{cell,1,Length@myRedTrackRGBTrimsAllCells,1},
  {spot,1,Length@myRedTrackRGBTrimsAllCells[[cell]],1}];*)
(*Can use this if I have some cells with different 488 and 561 intervals*)

myRedRGBGaussIntAllCellsFNs = Table[StringTake[myRedRGBTrimsAllCellsFNs[[cell]],
  (StringPosition[myRedRGBTrimsAllCellsFNs[[cell]], "my"] // Min) - 2] <>
  "_myRedTrackRGBGaussIntAllCells.m", {cell, 1, Length@myRedRGBTrimsAllCellsFNs, 1}];
```

Fitting the gaussian intensity of only frames that all fluorescences were being imaged; make sure to put the correct imaging interval below! (should not have to run this again and can just skip down to the uploading if you have already done this step!)

```
my488and561Interval = 2;
myRedTrackRGBGaussIntAllCells = Table[GetTrimPeakIntFixedSigma95Conf[#, 1.5] & /@
  myRedTrackRGBTrimsAllCells[[cell, spot, All, 1 ;; All ;; my488and561Interval]],
  {cell, 1, Length@myRedTrackRGBTrimsAllCells, 1},
  {spot, 1, Length@myRedTrackRGBTrimsAllCells[[cell]], 1}];
```

Making file names for the gaussian fit information

Exporting the gaussian fit information because it takes a little while to run (should not have to run this again and can just skip down to the uploading if you have already done this step!)

```
Table[Put[myRedTrackRGBGaussIntAllCells[[cell]], myRedRGBGaussIntAllCellsFNs[[cell]],
  {cell, 1, Length@myRedRGBGaussIntAllCellsFNs, 1}];
(*Only use if you generated more Gaussian Fits from the line above*)
```

```
myRedTrackRGBGaussIntAllCells = Get[#,] & /@myRedRGBGaussIntAllCellsFNs;
```

Counting the number of myRedTrackRGBGaussIntAllCells files, should be the same number as there are cells

```
Length@myRedTrackRGBGaussIntAllCells
```

```
2
```

Looking at the length of each gaussian fit, should be the track number above divided by the 488 and 561 interval, each number is the length of the gaussian fits for each spot

```
Length /@ myRedTrackRGBGaussIntAllCells[[1, All, 1]]
```

```
{120, 31, 33, 24, 50, 22, 20, 24, 25, 43, 23, 23, 22, 27, 27, 25, 45, 33, 22,
 21, 30, 34, 27, 43, 21, 34, 37, 29, 21, 83, 64, 22, 59, 29, 34, 25, 22, 78, 33,
 27, 26, 26, 25, 25, 29, 28, 23, 20, 88, 29, 27, 30, 21, 27, 24, 23, 46, 29, 51,
 46, 21, 37, 31, 30, 35, 22, 28, 25, 23, 39, 25, 29, 20, 24, 24, 29, 28, 22, 21}
```

This is the handtracking code, you can either click each spot one at a time for the downstream analyses OR you can take all the spots from a particular cell not both.

```
goodredspotlist = {};
ClearAll[lastsaved, numbersaved];
Manipulate[GraphicsRow[
  {ImageAdjust@myRedTrackRGBTrimsAllCells[[cell, spot, 1, 1 ;;]] // Mean},
  ImageAdjust@myRedTrackRGBTrimsAllCells[[cell, spot, 2, 1 ;;]] // Mean},
  ImageAdjust@myRedTrackRGBTrimsAllCells[[cell, spot, 3, 1 ;;]] // Mean},
  {"Last Saved", lastsaved}, {"Number Saved", numbersaved}}, ImageSize -> 500],
{spot, 1, Length@myRedTrackRGBTrimsAllCells[[cell]], 1, Appearance -> {"Open"}},
{cell, 1, Length@myRedTrackRGBTrimsAllCells, 1, Appearance -> {"Open"}},
Button["Click To Save Spot", {lastsaved = {cell, spot},
  mytempgoodspot = {myRedTrackRGBTrimsAllCells[[cell, spot]]},
  mytemptracknumber = {myRedParameterSplitColReg[[cell, spot, 1, 1]]};
  mytempgoodspotGaussInt = {myRedTrackRGBGaussIntAllCells[[cell, spot]]};
  goodredspotlist = Append[goodredspotlist, {cell, myRedRGBTrimsAllCellsFNs[[cell]],
    mytemptracknumber, mytempgoodspot, mytempgoodspotGaussInt}},
  numbersaved = {"Cells_" <> ToString[Length@goodredspotlist],
    "Spots_" <> ToString[Length@goodredspotlist[[cell, 3, All, 1]]]}},
Button["Click Here If All Look Good (Take All Spots From This Cell!)",
{lastsaved =
  {Length@myRedTrackRGBTrimsAllCells, Length@myRedTrackRGBTrimsAllCells[[cell]]},
  mytempgoodspot = {myRedTrackRGBTrimsAllCells[[cell, All]]},
  mytempgoodspotGaussInt = {myRedTrackRGBGaussIntAllCells[[cell, All]]};
  goodredspotlist = Append[goodredspotlist, {cell, myRedRGBTrimsAllCellsFNs[[cell]],
    Thread@{myRedParameterSplitColReg[[cell, All, 1, 1]], myRedTrackRGBTrimsAllCells[[
      cell, All]], myRedTrackRGBGaussIntAllCells[[cell, All]]}}},
  numbersaved = {"Cells_" <> ToString[Length@goodredspotlist],
    "Spot_" <> ToString[Length@Flatten@goodredspotlist[[All, 3, All, 3, 1]], 1]}},
FrameLabel -> {{Style["Red", Large, Red], Style["Green", Large, Green],
  Style["Blue", Large, Blue]}}
```

```
myDataFileName = "20190920_smFLAG-KDM5B";
```

(*Change this name first!*)

```
HCRRedTracksAllCells = goodredspotlist; (*Need to now export it out in an excel format*)
analysisFolder = "P:\\FCS_data\\";
```

```
HCRRedTracksAllCells >>
```

```
(analysisFolder <> "\\\" <> myDataFileName <> "_HCRRedTracksAllCells.m");
```

(*Change the name of the construct above manually!*)

Categorizing Species (Do this one if all data sets are taken at the same imaging

intervals!)

I usually make this folder separately and copy and paste the HCRedTracksAllCells.m into this directory that I manually make, usually the date and the construct(s) name that was used for the experiment

```
analysisFolder = "P:\\FCS_data\\20190920_smFLAG-KDM5B";
HCRedTracksAllCellsFNs = FileNames[analysisFolder <> "\\*_HCRedTracksAllCells.m"];
HCRedTracksAllCellsFNs
{P:\\FCS_data\\20190920_smFLAG-KDM5B\\20190920_smFLAG-KDM5B_HCRedTracksAllCells.m}
HCRedTracksAllCells = Get[HCRedTracksAllCellsFNs[[1]]];
HCRedTracksAllCells // Dimensions
{2, 3}
```

This is the spot detector or “species” definer function; it will define 4 types of species; all species must have a red signal (JF646, RNA signal); the species are categorized as: Red tracks, Red + Green tracks, Red + Green + Blue tracks, and Red + Blue tracks; these colors are based on the light of excitation!

```
myHCRedTracksAllCellsSpecies = SpeciesPlotDataFN[HCRedTracksAllCells, 5, 2];
Length /@ myHCRedTracksAllCellsSpecies
{104, 74, 0, 0}
```

Categorizing Species (Do this one if you have two data sets that are taken with different imaging intervals! Do not need to use otherwise)

Exporting Out Excel With Intensities

This is making the labels for the excel sheet here

```
mylabel = {"Cell_#", "Spot_#", "Time_(sec)", "R_GaussianInt", "R_95Conf",
           "G_GaussianInt", "G_95Conf", "B_GaussianInt", "B_95Conf", "Predicted_Species"};
```

This sorts the cells and spots into an excel format

```

myXLSdata =
  Flatten[Table[{ConstantArray[myHCRedTracksAllCellsSpecies[[species, track, 1]],
    Length@myHCRedTracksAllCellsSpecies[[species, track, 4]]],
    ConstantArray[myHCRedTracksAllCellsSpecies[[species, track, 2]],
    Length@myHCRedTracksAllCellsSpecies[[species, track, 4]]],
    myHCRedTracksAllCellsSpecies[[species, track, 4]],
    myHCRedTracksAllCellsSpecies[[species, track, 5, 1, All, 1]],
    myHCRedTracksAllCellsSpecies[[species, track, 5, 1, All, 2]],
    myHCRedTracksAllCellsSpecies[[species, track, 5, 2, All, 1]],
    myHCRedTracksAllCellsSpecies[[species, track, 5, 2, All, 2]],
    myHCRedTracksAllCellsSpecies[[species, track, 5, 3, All, 1]],
    myHCRedTracksAllCellsSpecies[[species, track, 5, 3, All, 2]],
    ConstantArray[species,
      Length@myHCRedTracksAllCellsSpecies[[species, track, 4]]]],
    {species, 1, Length@myHCRedTracksAllCellsSpecies, 1},
    {track, 1, Length@myHCRedTracksAllCellsSpecies[[species]], 1}], 1];

myXLSdataThread = Thread @@ # & /@ myXLSdata;

mylabelandXLSdata = Join[{mylabel}], Flatten[myXLSdataThread, 1]];

mylabelandXLSdata // Length

6489

mylabelandXLSdata[[1 ;; 10]] // TableForm

```

Cell_#	Spot_#	Time_(sec)	R_GaussianInt	R_95Conf	G_GaussianInt	G_95Conf
1	1	2	0.013752	0.00429472	0.101251	0.0278364
1	1	12	0.012891	0.00379142	0.107346	0.0287541
1	1	22	0.0112601	0.00399341	0.0982463	0.0288326
1	1	32	0.0115934	0.0029731	0.0922034	0.0264748
1	1	42	0.011047	0.0027691	0.105562	0.0309518
1	1	52	0.0108015	0.00309043	0.105013	0.0276387
1	1	62	0.0125057	0.00315215	0.0953914	0.0317191
1	1	72	0.0155269	0.00291933	0.109387	0.0358723
1	1	82	0.0138254	0.00313099	0.121826	0.0325318

```

myDataFileName = "20190920_smFLAG-KDM5B" ;
(*Change this name first!*)

Export["D:\\Dropbox\\Work Folder\\CSU Graduate Work\\Stasevich Lab\\Paper
  Work\\Frameshift\\Figures\\_Mathematica\\munsky_data\\Multiplex_FCS" <>
  "\\\" <> myDataFileName <> ".xls", mylabelandXLSdata, "XLS"]
(*Need to name this correctly before export!*)

D:\\Dropbox\\Work Folder\\CSU Graduate Work\\Stasevich Lab\\Paper
  Work\\Frameshift\\Figures\\_Mathematica\\munsky_data\\Multiplex_FCS\\20190920_smFLAG-KDM5B.
  xls

```

Making Average Trims From (*May need to do this later after hand categorizing*)