Part #1: Measuring Spatial Covariance in Imaging System

1. Cap on, collect several hundred frames in 100x100 area of interest
   1. Compute spatial covariance function
      1. Eqn in class notes: x is some center pixel, y is all the other pixel; scalar and image multiplied together
         1. Y expected should be a mean image, one loop!! X is scalar, y is image
         2. Returns image
         3. Can do a running mean
         4. \*\*\*\*\*
         5. Get 300 noise images, each y - y mean
         6. X - x mean is center column
         7. Multiply together
         8. Mean of those 300 → image
         9. Double checks: center value has to be highest, covariance of x with self is = variance
   2. Compute noise powers spectrum (e.g. the Fourier transform of the covariance)

np.cov(array) returns covariance matrix

1. Should have a map for the covariance
2. Only pick the area of interest and calculate the covariance for that area not for all(difficult to calculate)
3. Cap off, Set up “good” imaging mode (with exposure ~25 ms) on uniform 100x100 AOI, collect several hundred frames in focus, and several hundred frames out of focus
   1. Compute spatial covariance function for in-focus and defocused images
   2. Compute noise power spectra for in-focus and defocused images

Part #2: Measuring Spatial Resolution Properties \*\*Don’t take over couple different angles\*\*

\*\*New physical set up with XY positioner as backlit target, aperture 16, pixelclock 7\*\*

1. Keeping focus the same (sharpest image possible seeing part that covers “2.0 lp/mm down to 10 lp/mm”), collect line pair, star, and knife images (averaging over a few to get rid of noise if necessary; can take two or three then see if needed)
   1. For star, says we can use specific AOI, again can take sample image and see. Should take full field image of knife though
2. Repeat A but out of focus, “so that the 4 lp/mm target is completely blurred”, but not more blurry than necessary
3. Repeat A but out of focus, “so that the 8 lp/mm target is completely blurred”, but not more blurry than necessary

A, B, and C all for MTF (spatial frequency response of an imaging system) calculations!

Calculating MTF: Edge Spread Function (from knife test) –derivative→ Line Spread Function –fourier transform→ MTF

MTF Calc:

* Pixel grid with y axis, left of y axis (slightly tilted line) is cross hatching = ?
* Want all samples on D axis (x axis)
  + Samples = distance between vertical line and the blue dot, that’s the x coordinate
* Y-axis: line of best fit between points
  + <https://www.researchgate.net/publication/259898919_Spatial_Quality_Assessment_of_Pan-Sharpened_High_Resolution_Satellite_Imagery_Based_on_an_Automatically_Estimated_Edge_Based_Metric#pf7> - figure 2 seems to have process on how to find this
* X Y pixel coordinates to D coordinates (eqn for distance between point and line)
  + Can loop through pixels ], don’t do that
* Image: I(x,y) (intensity function) X Y I → D (distance, sorted) I (should be between 0 and 1)
* Last, for loop to bin D vs I values so its uniformly sampled

<https://stackoverflow.com/questions/61135813/python-script-for-mtf>

<https://www.researchgate.net/post/Is_this_right_way_to_get_Modulation_Transfer_FunctionMTF>

Part #3: Obtain sharpest image possible of star pattern

* Bonus: collect line pair target using same methodology