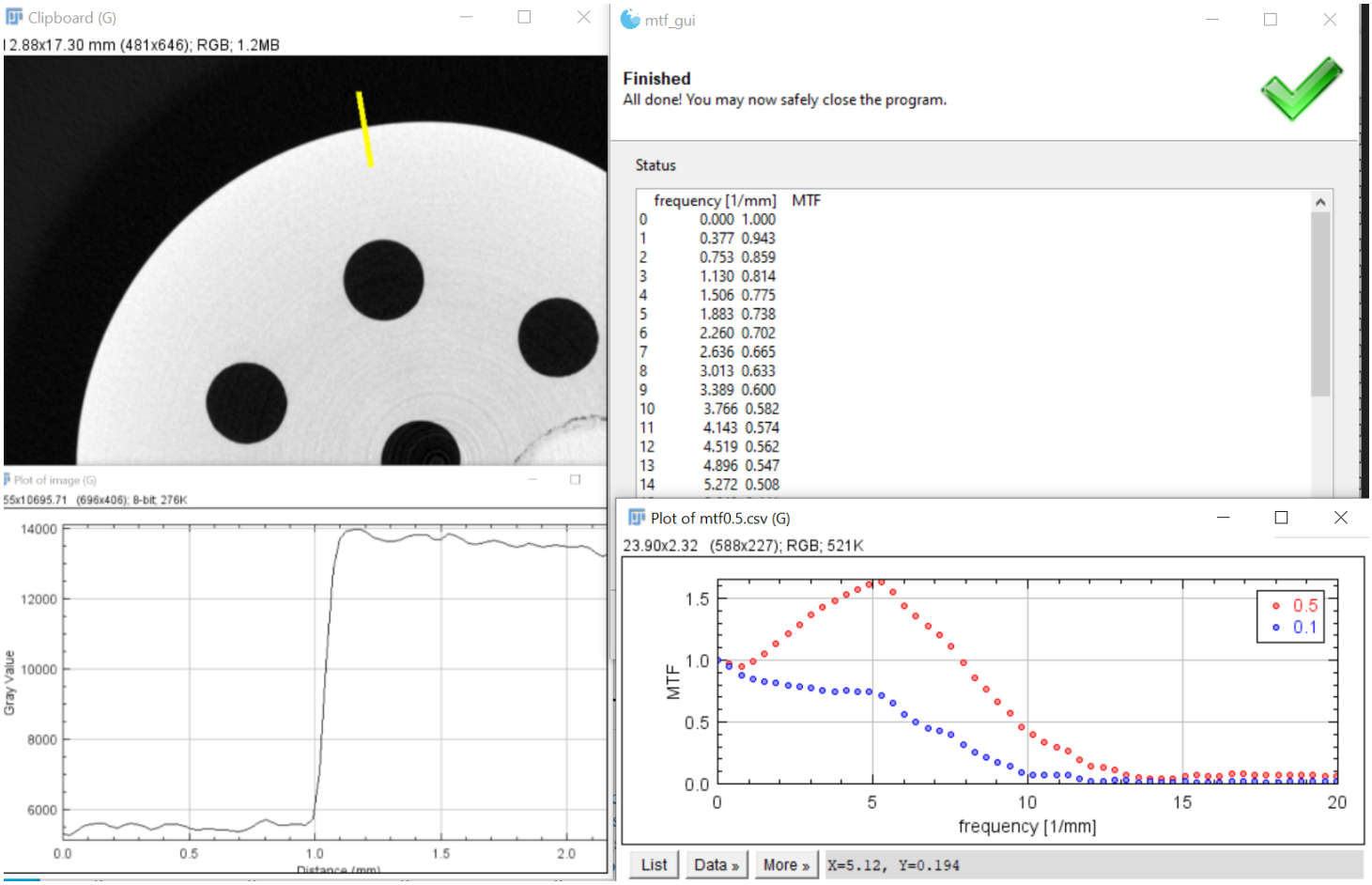
XRI MTF tool

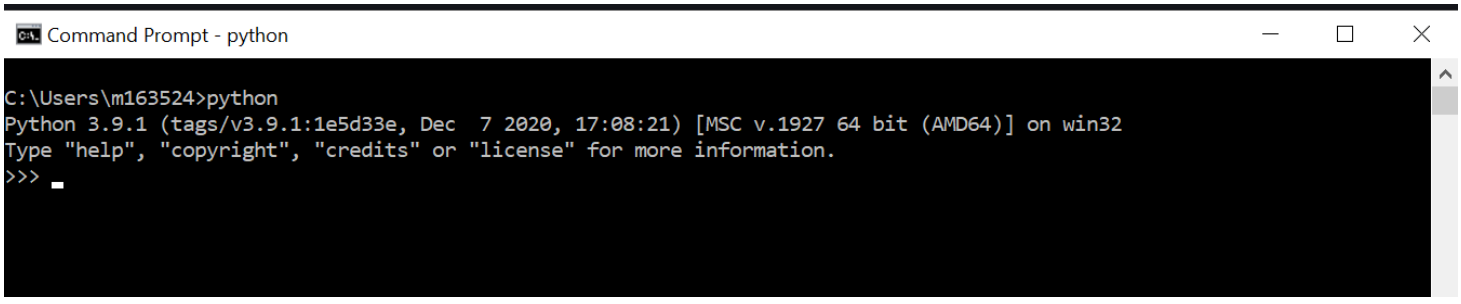
The XRI MTF tool is a simple MTF measurement tool/protocal using a small python script and leaning heavily on existing functionality in ImageJ



Installation

The following assumes installations of python and ImageJ

To check if you have a version of python installed, open a command line and type `python`



If you do not get something like above, first download python from: https://www.python.org/downloads/

Likewise ImageJ can be downloaded here (Fiji is just ImageJ and is the recommended fully-loaded ImageJ): https://imagej.net/software/fiji/downloads

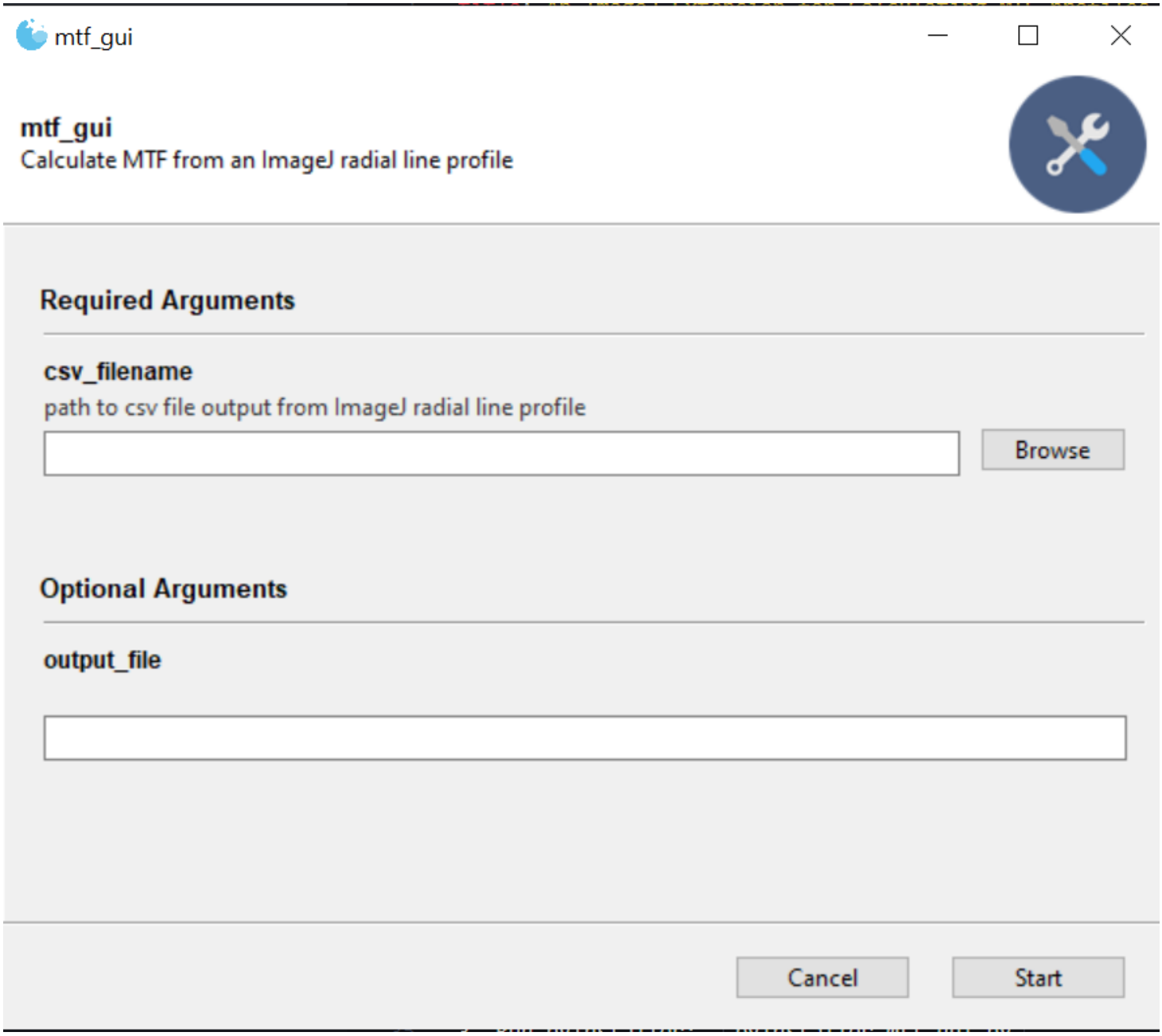
Install instructions for basic command line usage

1. Copy the files to your local computer:
    - **Preferred:** Using git:

      ```
      git clone https://rohaslrailgit.mayo.edu/ctcic/mct-developement/xri-mtf.git
      ```
    - If you prefer not to use git or don't have it installed you can also code to the code site: https://rohaslrailgit.mayo.edu/ctcic/mct-developement/xri-mtf and find the download button to download everything as a zip folder. Once you unzip it you can proceed with the rest of the install.

2. change the working directory to be in the program folder
    - `cd xri-mtf`
3. install dependencies
    - `pip install .`
4. test run:
    - `python mtf_gui.py`

The following window should appear:



(optional) If you want an executable shortcut

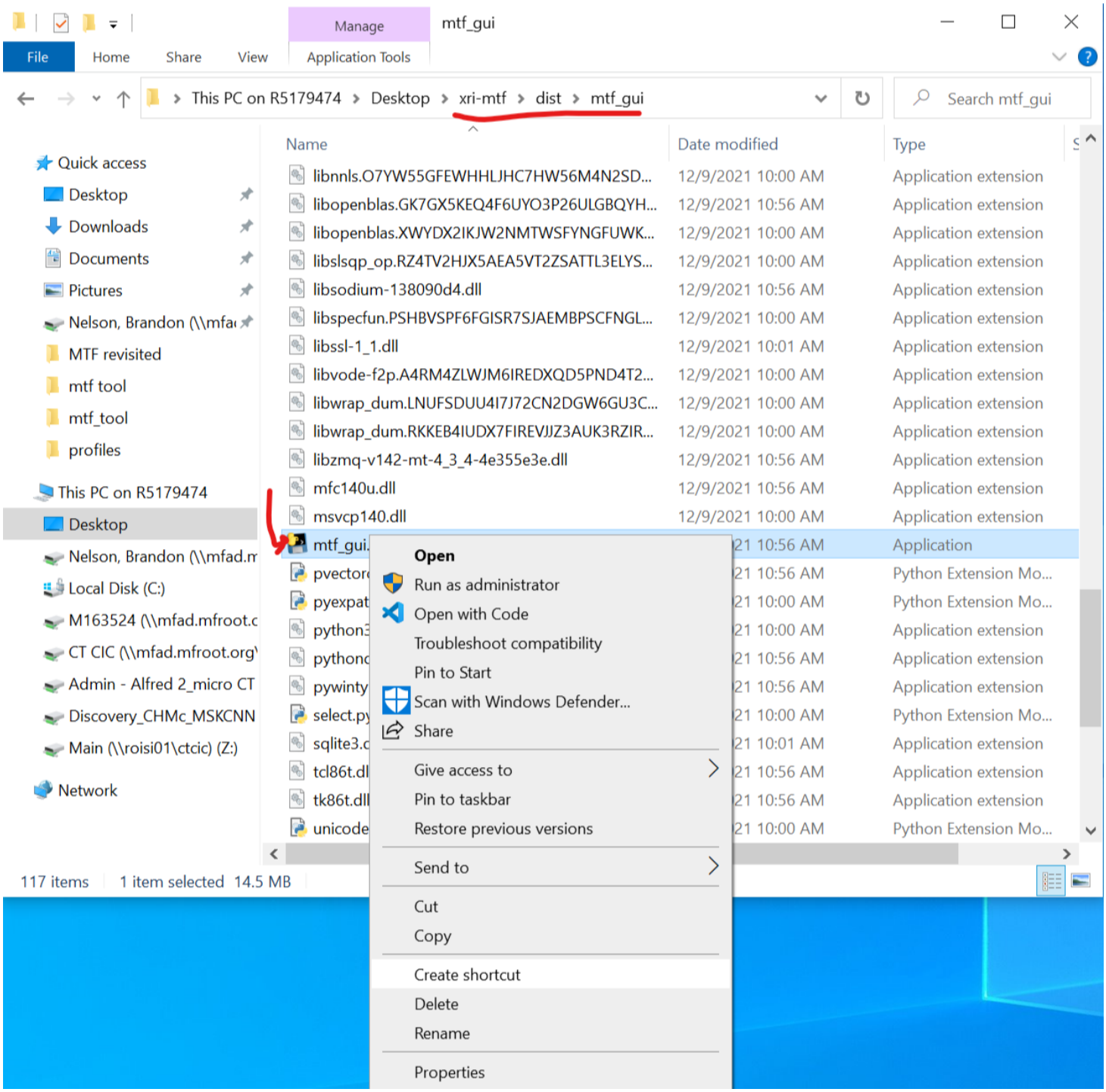
3. Run pyinstaller: `pyinstaller mtf_gui.py`

This should take 1-2 minutes to complete....

4. Create a shortcut to the executable

Once pyinstaller finishes, in your file explorer, inside the program folder ("xri-mtf") navigate to `/dist/mtf_gui/` and inside you will find `mtf_gui.exe`

right click on it and select `create shortcut` , you can then move this shortcut to your desktop or any convenient location to call the program from.



Usage

The program assumes `csv` files in the format produced by ImageJ which is a 2 columns CSV, where the first column contains the distances and the second contains gray values from the line profile.

Sample profiles have been provided in the `profiles` directory to experiment with.

Calibrating Pixel Scale

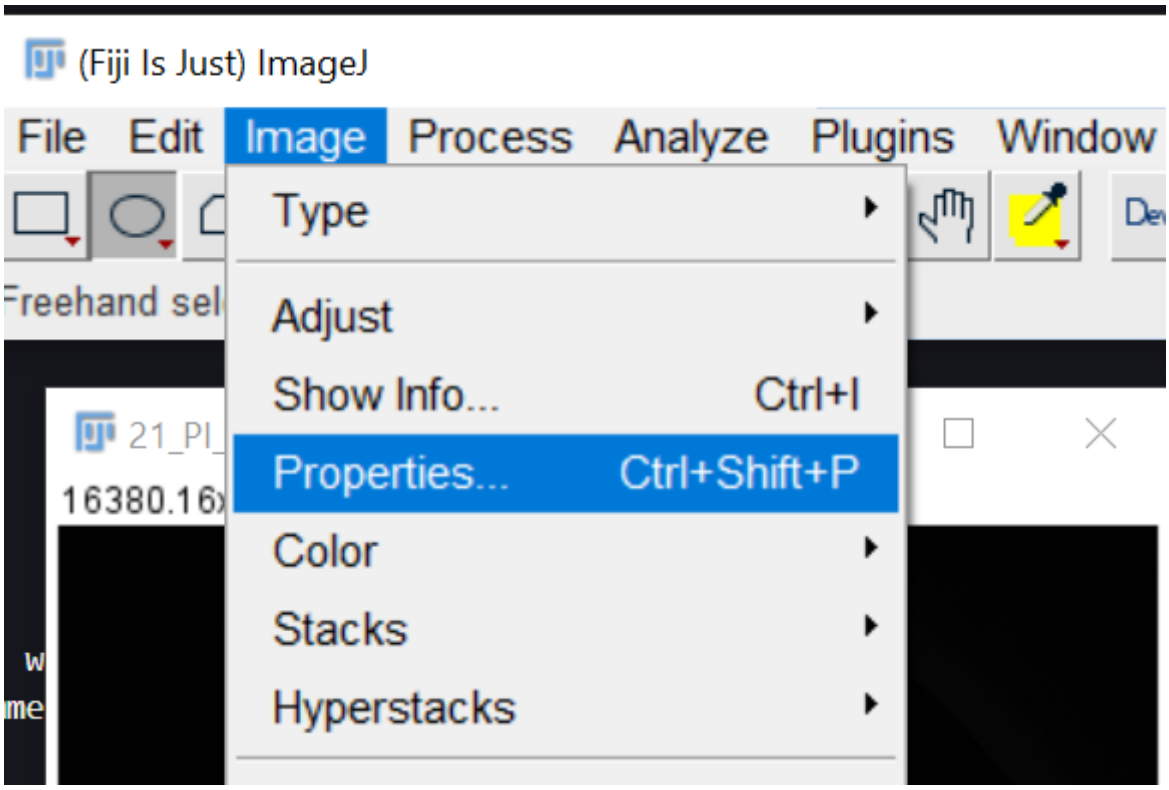
If you want the MTF results in proper units the image must have a spatial calibration, i.e the pixel sizes are known. (Note this is done automatically in the Bruker and NSI recons). However if the spatial scale is not set you'll need to do this manually. There are lots of simple tutorials online for how to do this, here's one: https://microscopy.berkeley.edu/courses/dib/sections/04IPIII/IJsetscale.html

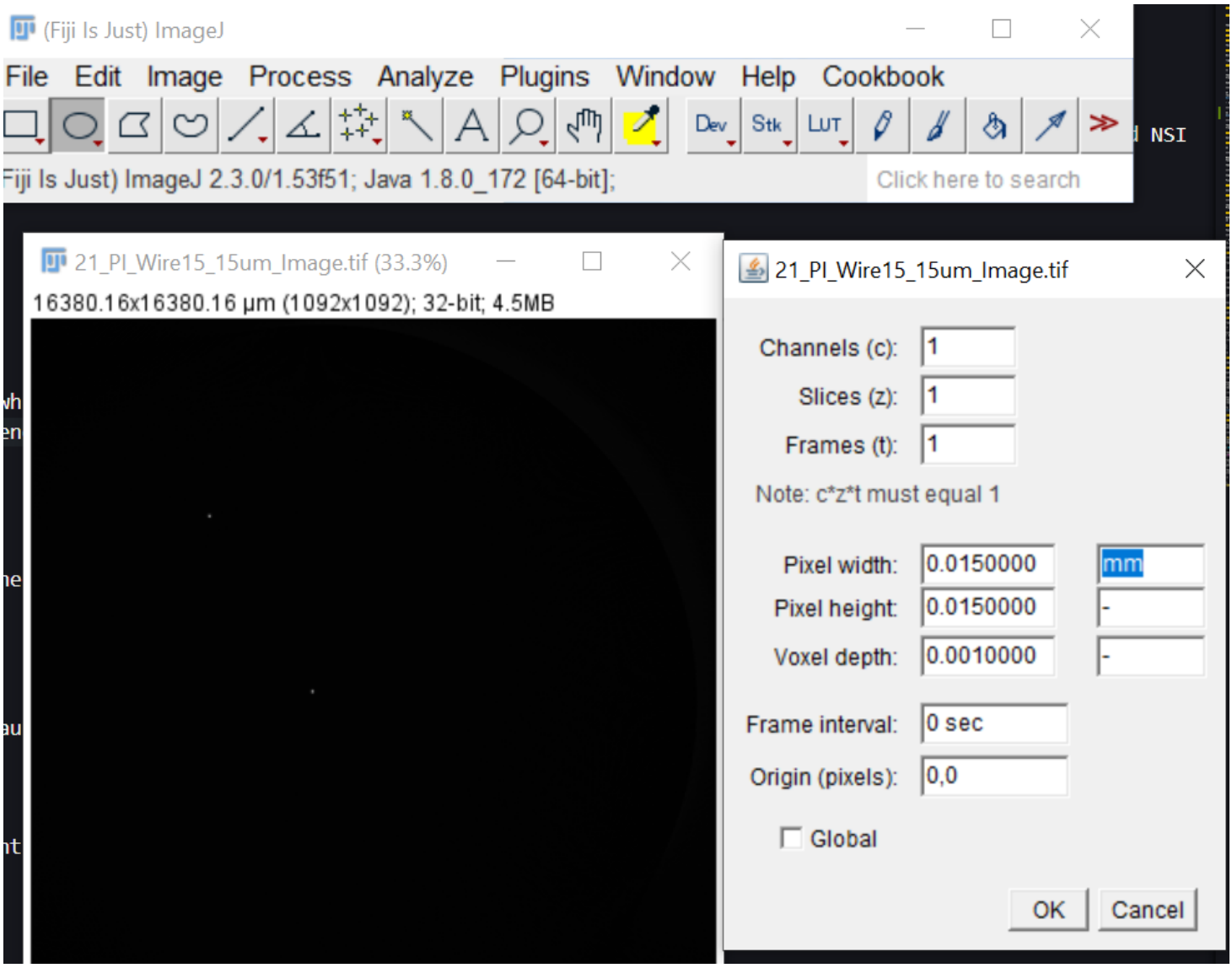
- If pixels are already calibrated you can skip this step

Spatial units

In clinical CT most MTFs are a function of frequencies in units of 1/cm, while in micro-CT the units are commonly 1/mm. While um are commonly used to described distances in micro-CT they will result in spatial frequencies less than 1, so I recommend converting pixel units to `mm` in ImageJ if they are in microns.

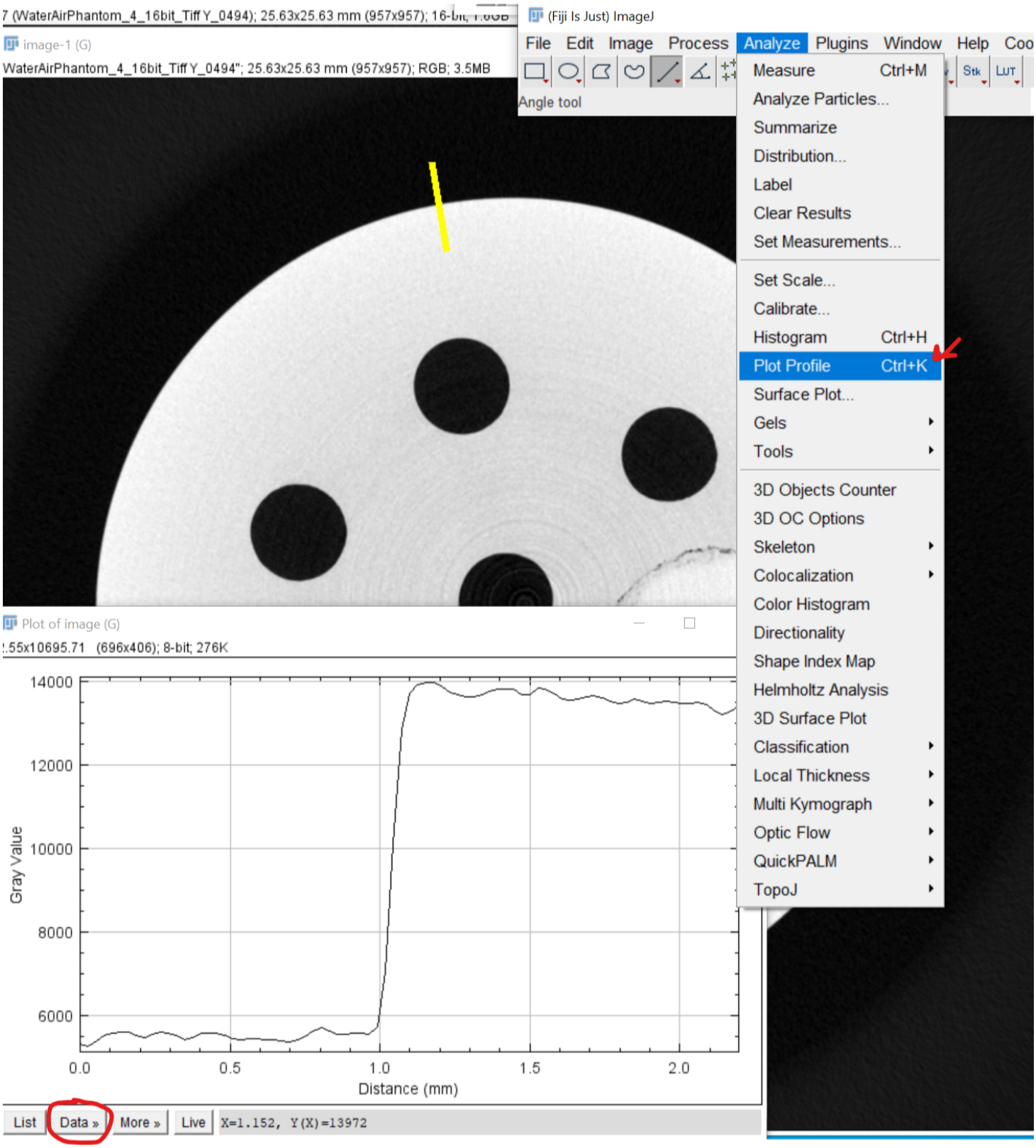
This is done in `Image-->Properties`



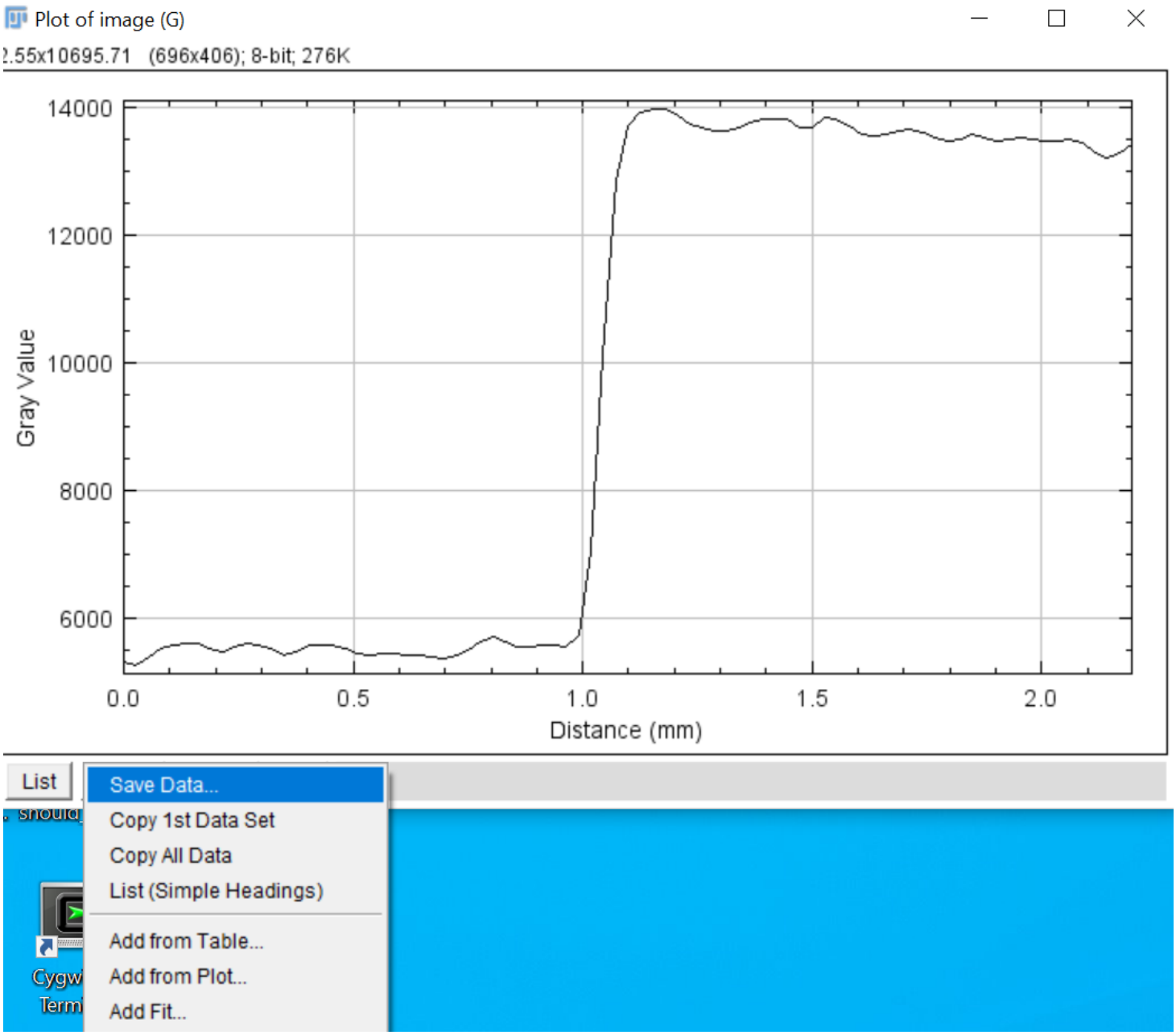


Measuring Edge Spread Profiles

The simplest MTF measurement starts with an edge profile from a single line ROI as shown below (note the <ctrl>+<k> keyboard shortcut to quickly grab a line profile from a line ROI). This will pull up a plot of the profile.



Save the profile using the `Data --> Save Data...` window, note the default output is csv.



In this example I saved out a file called `edge_profile.csv` to the current directory.

Measuring Radial Averages

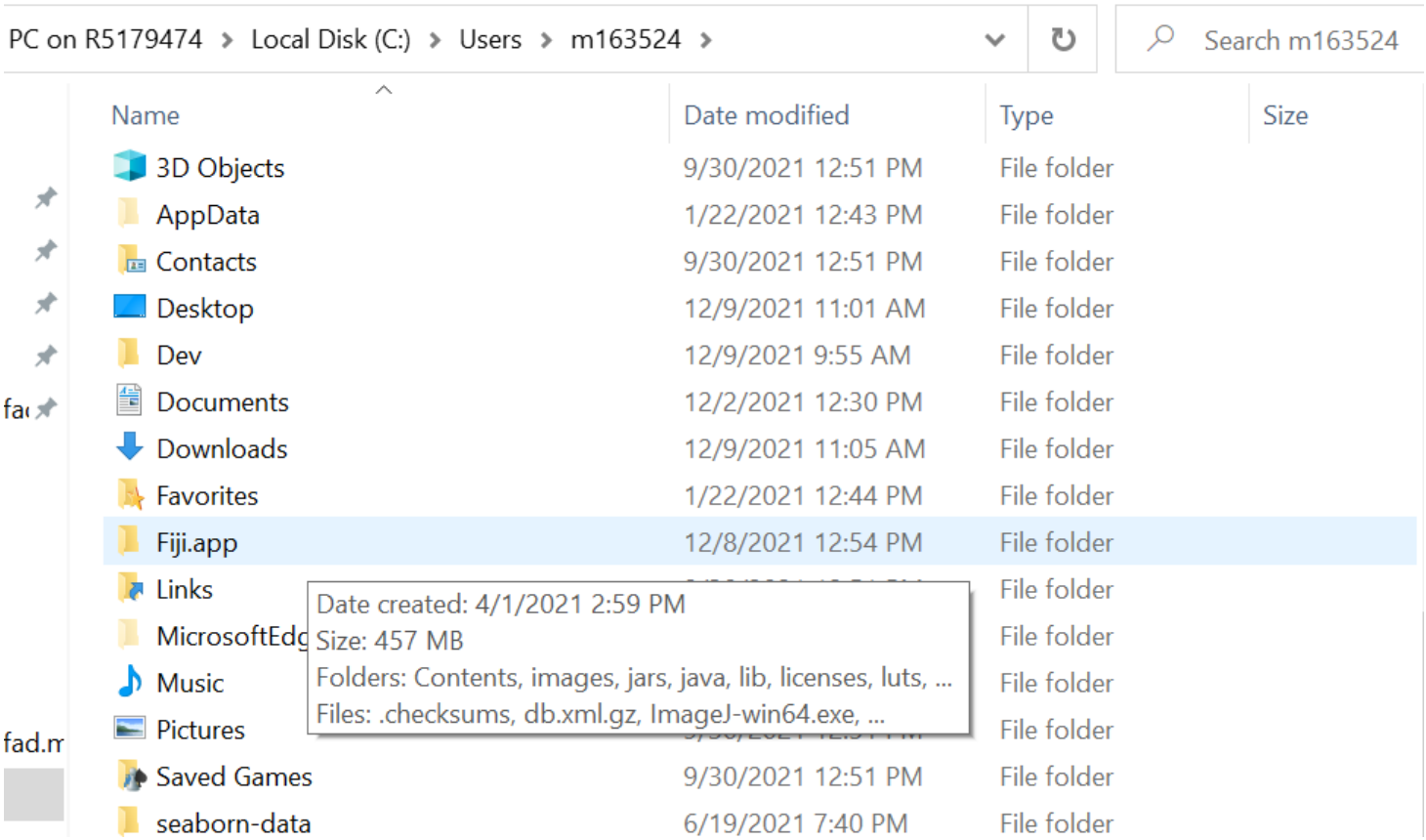
When measuring noise data or wire phantoms that are small and round, edge profiles may give noisey results. For this we make use of the **ImageJ radial profile plugin**, found here: https://imagej.nih.gov/ij/plugins/radial-profile.html

The plugin is called `Radial_Profile.class` and is already included inside `install files`

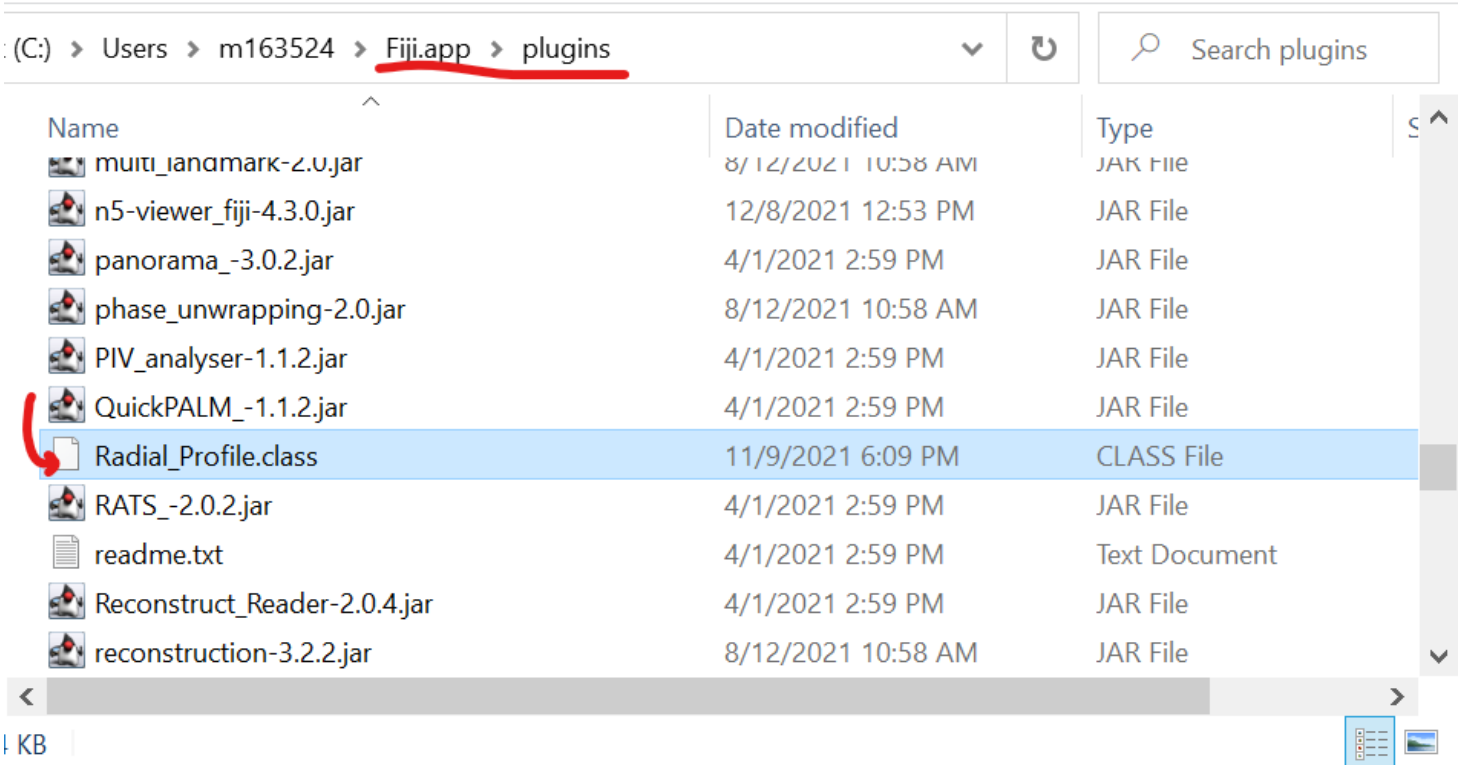
Installing the radial profile plugin

Copied from the plugin website: Move `Radial_Profile.class` to the **ImageJ plugins folder** and **restart ImageJ**

Fiji/ImageJ is typically installed by default in your home directory, e.g. `c:/Users/LanID/Fiji.app`

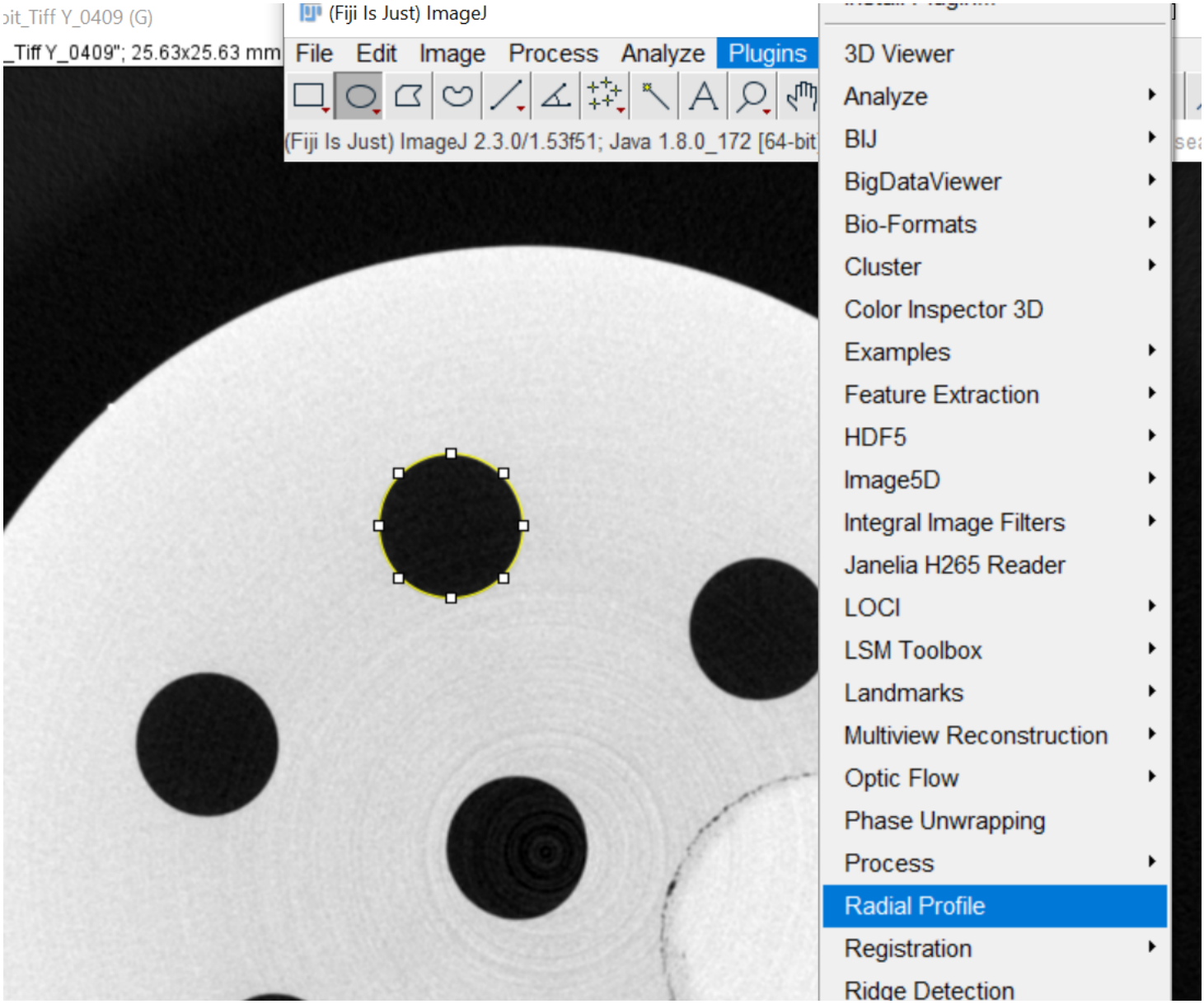


Inside the Fiji.app folder is the plugins folder and in there is where you will move `Radial_Profile.class` like shown below:

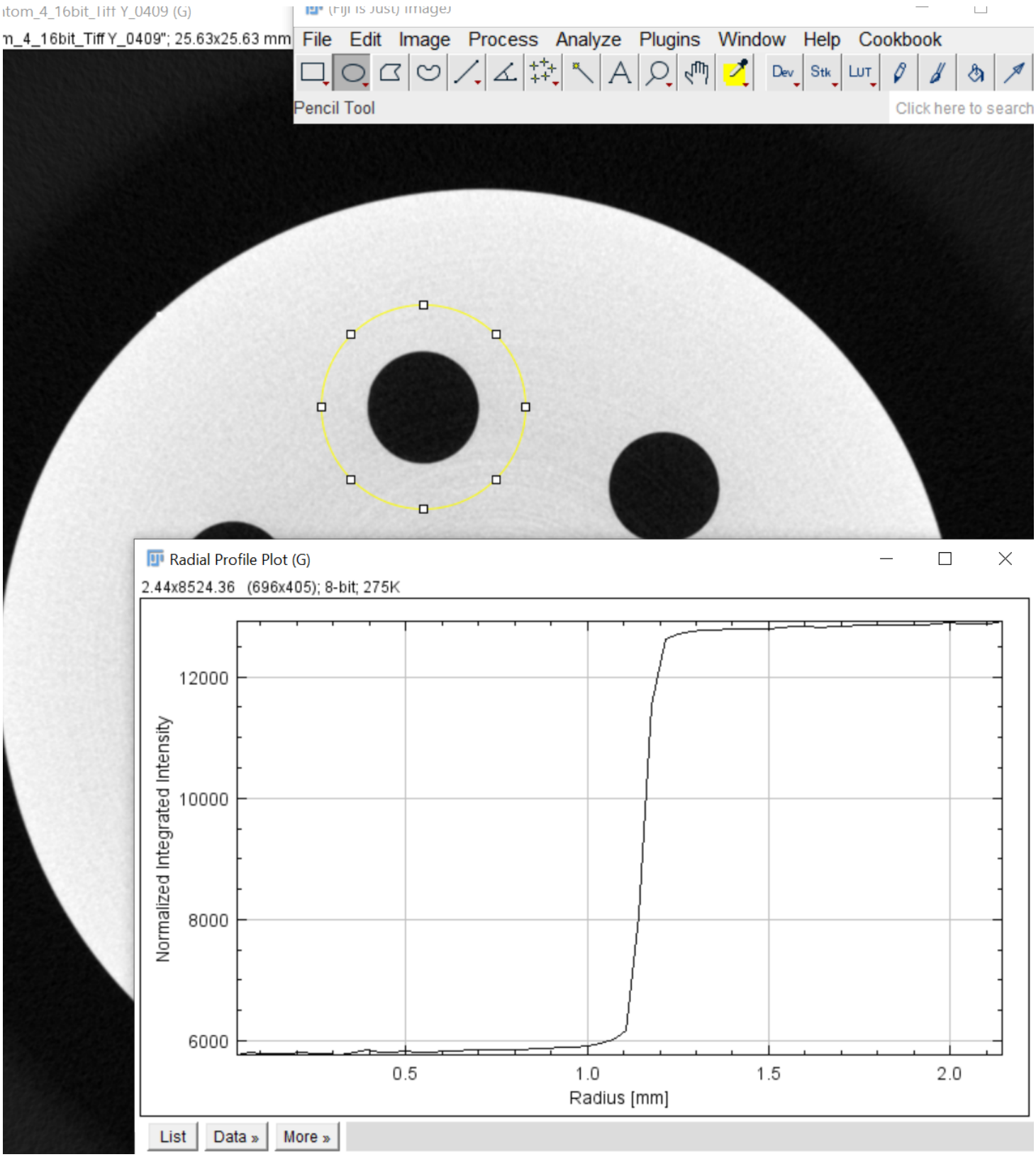


Using the radial profile plugin

the fastest way to use the plugin is to draw a circle ROI like so (plugin will draw equiangular lines from the center to the perimeter then average them)the select `Radial Profile` from the plugins folder:



Before running it will give you the option to change the (x, y) center and the radius, I typically increase the radius by a factor of 2 such that the edge is approximately in the center, the select ok.



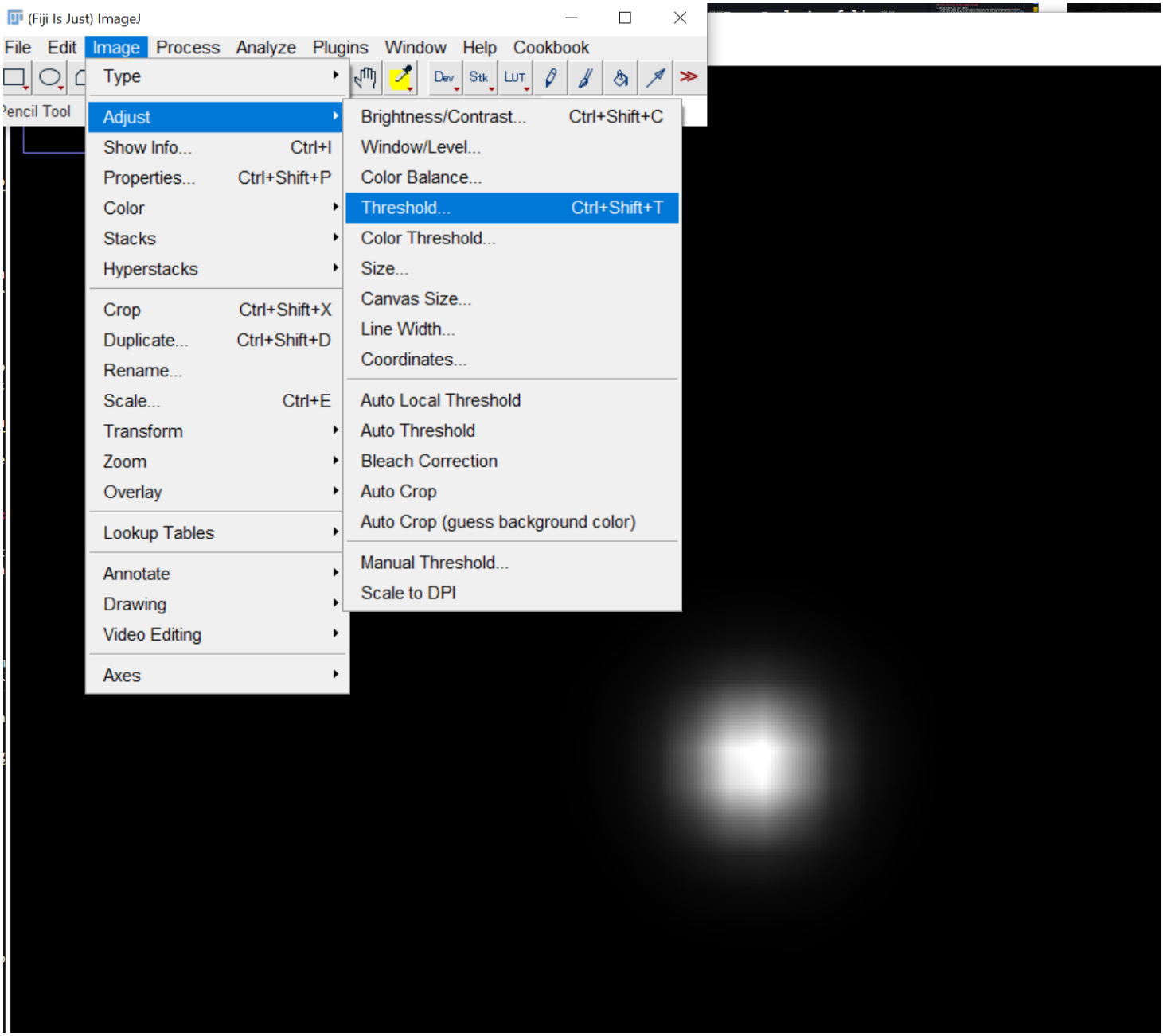
You can then save out the line profile the usual way described above.

Robust determination of circle center

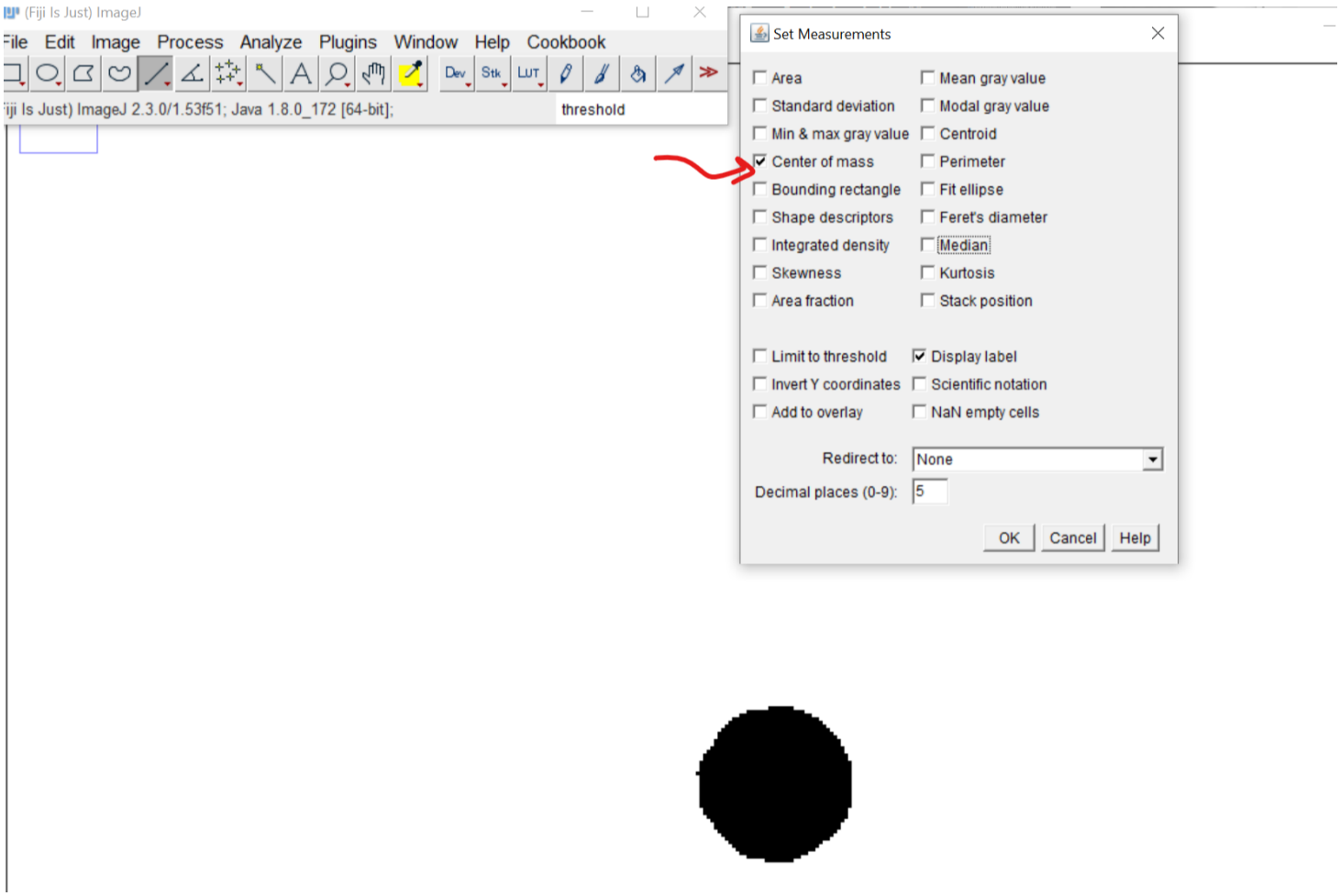
Generally hand drawn ROIs work plenty fine for getting a radial averaged point spread function. But if you want to be most accurate you can use the "center of mass" method to measure the absolute

center and uee that to update the x, y pixel locations when calculating the radial average.

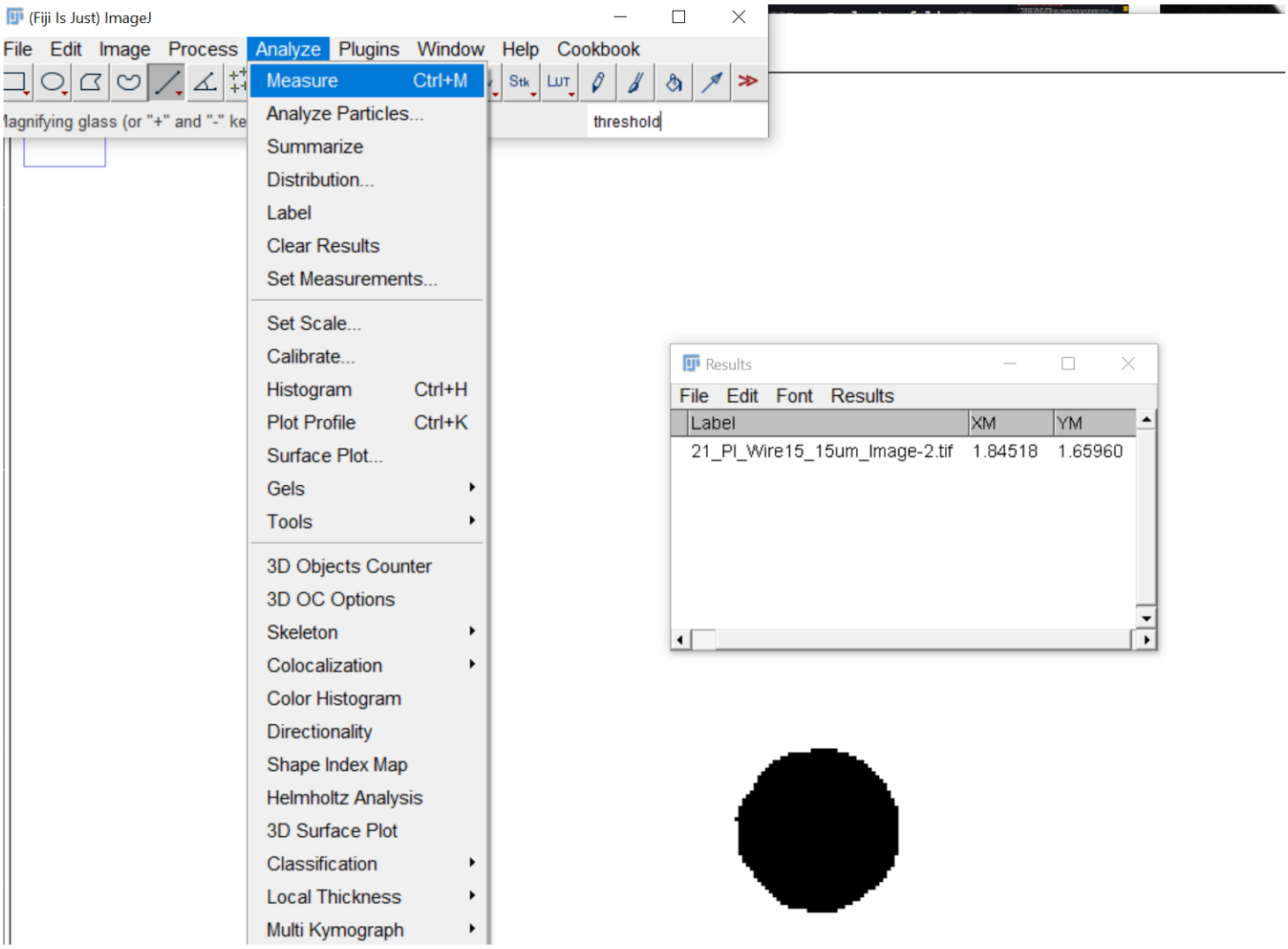
First threshold the image to get a binary image of the wire, click `auto` and `apply`



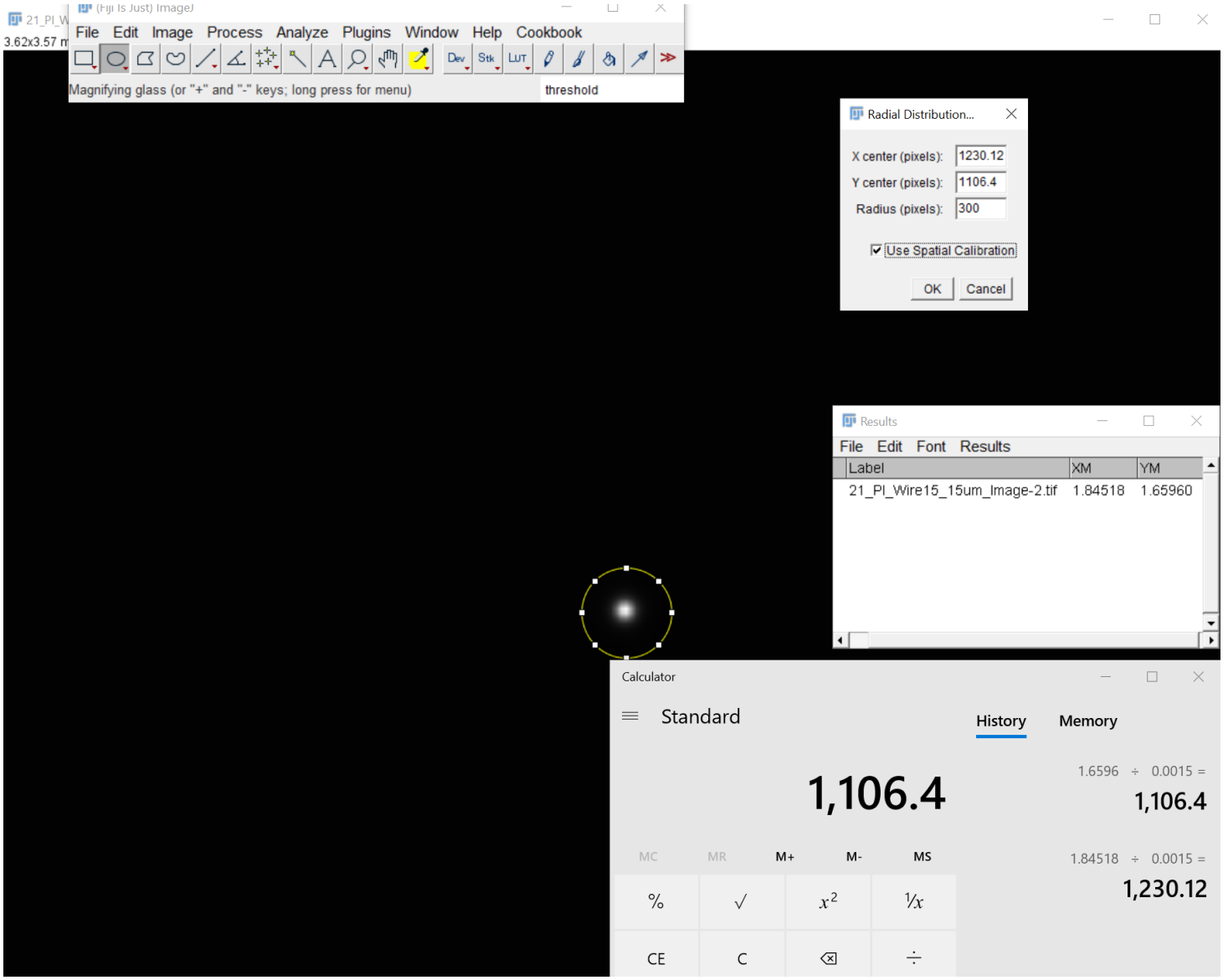
Then in `Analyze --> Set Measurements...` , make sure `Center of Mass` is ticked



Then select `Analyze --> Measure` which will display the X, Y coordinates of the center (annoyingly in pixel units)



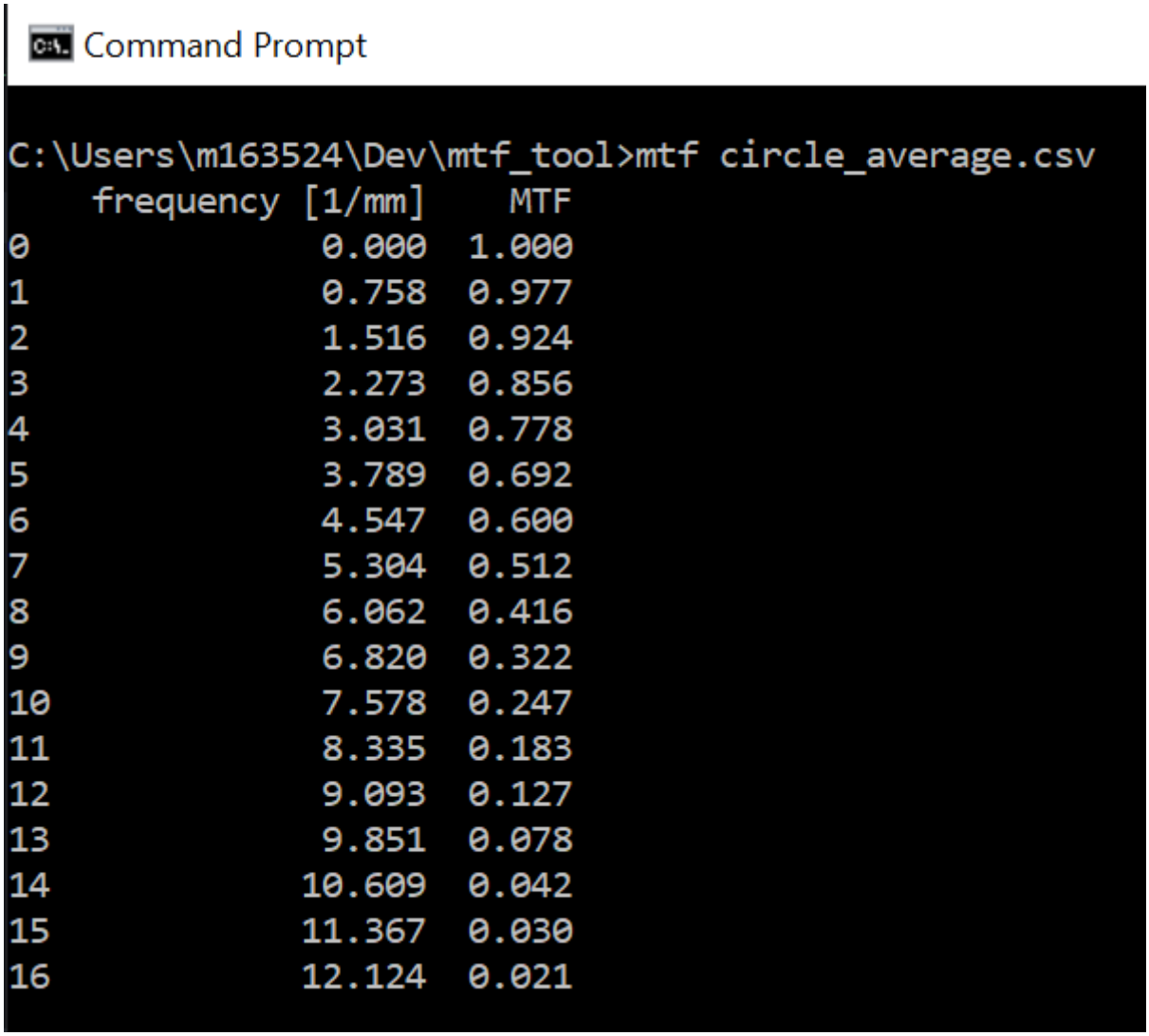
To correct the location of the center of wire you'll need to divide the measured x, y center by the pixel size to get the x, y pixel indices required by the radial profile plugin as demonstrated below.



Command line usage

- The MTF tool works both from the command line as well as from a graphical program. The command line program works as follows: Starting in the program directory run:
  `mtf my_edge_profile.csv` , where you can replace the csv filename with your own.

Without any other arguments the MTF values will be output straight to the terminal



Saving to csv file

You can also save the results to csv format with the following:

```
python mtf.py my_edge_profile.csv -o mtf.csv
```

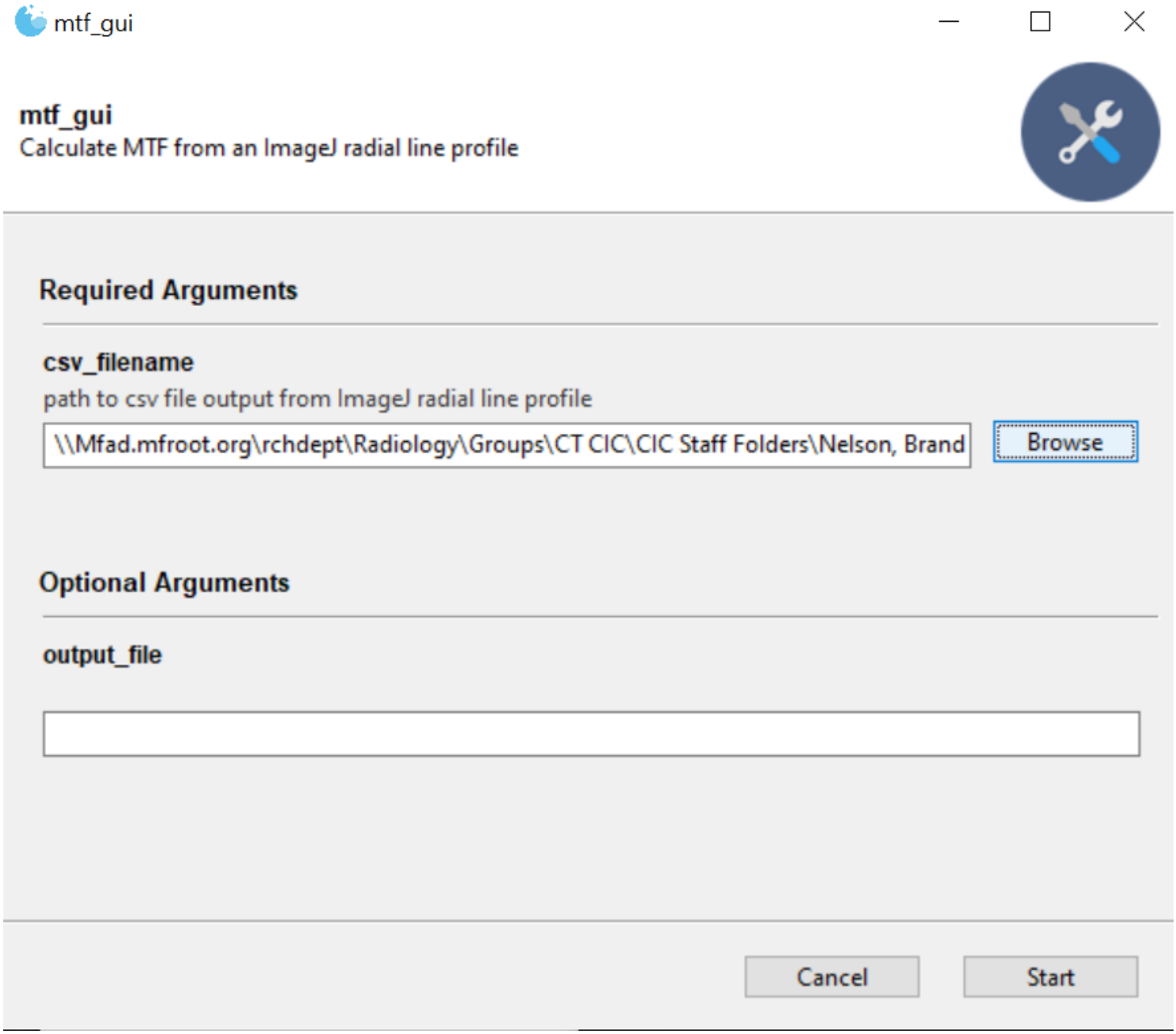
Note the `-o` flag is short for "output filename"

Graphical user interface

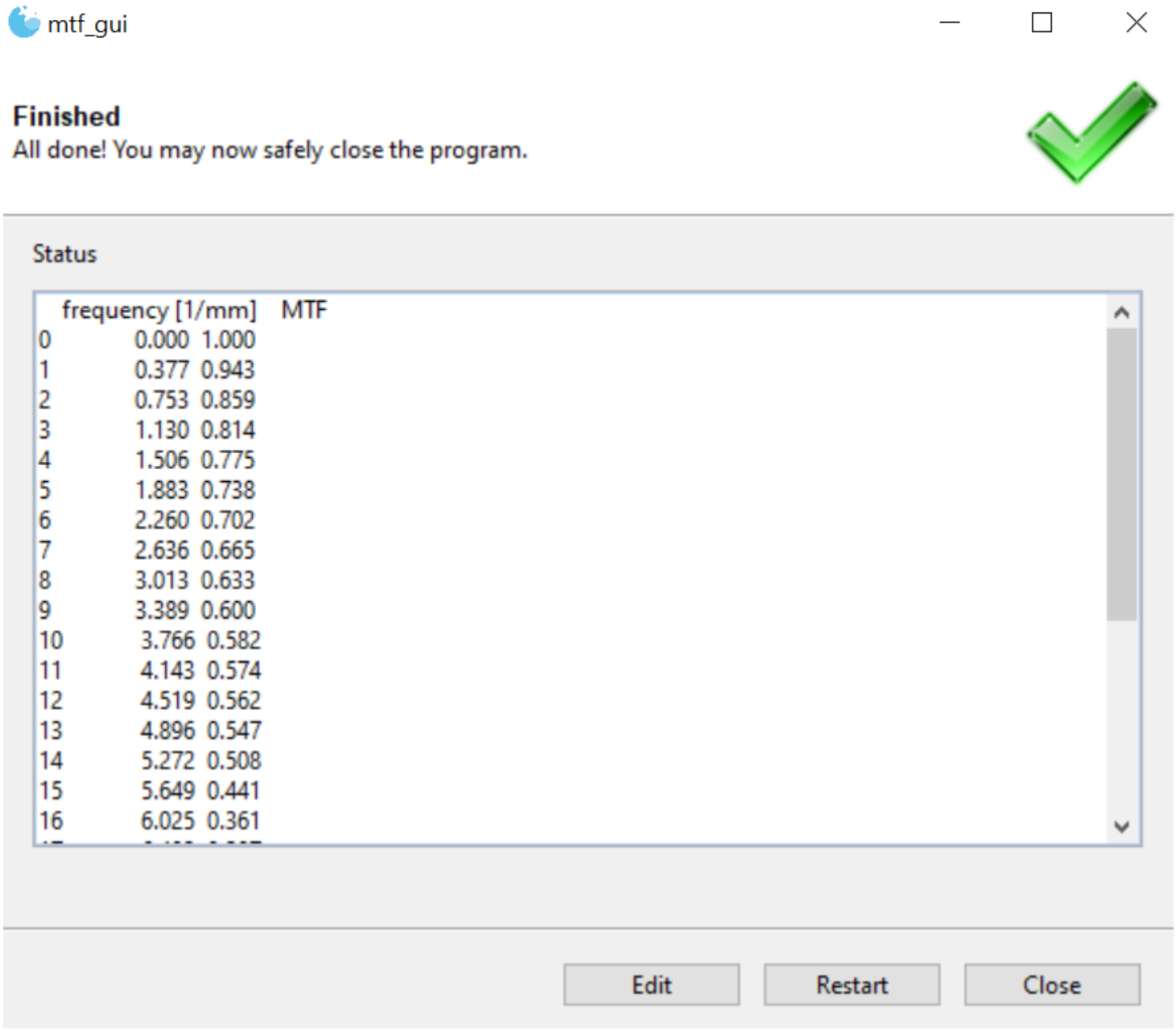
The graphical program can be started either by double-clicking on the executable shortcut or opened from the command line via:

```
python mtf_gui.py
```

Once the program is opened you can select the `Browse` button to navigate to and select a csv file containing an edge profile:



Selecting start will then run the program and output the results to the program window:



You can also specify an output filename (don't forget to have a `.csv` ending) to save the output to.

Visualizing MTF results

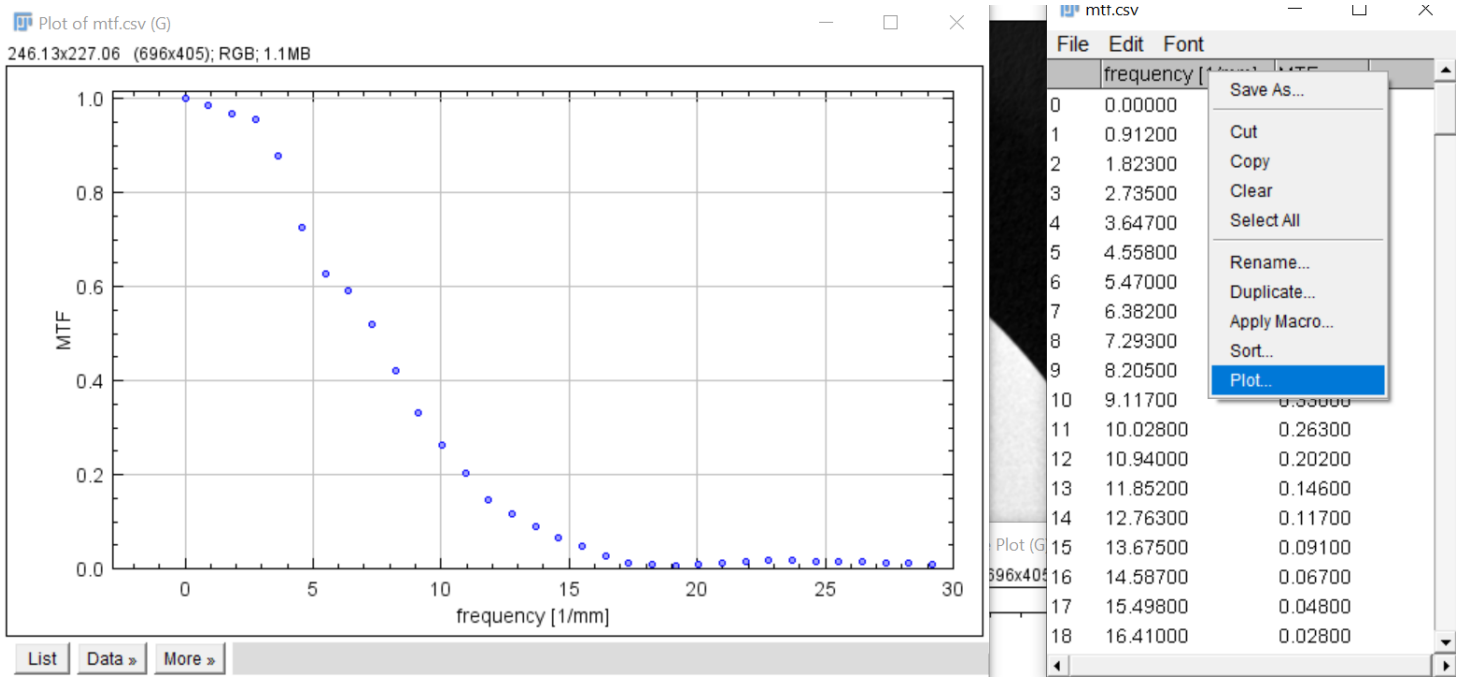
With the MTF csv output you can visualize the result in a variet of ways such as in Excel or even in ImageJ (since you likely still have it open). Since this my preference I will demonstrate ImageJ usage below

with ImageJ plotting

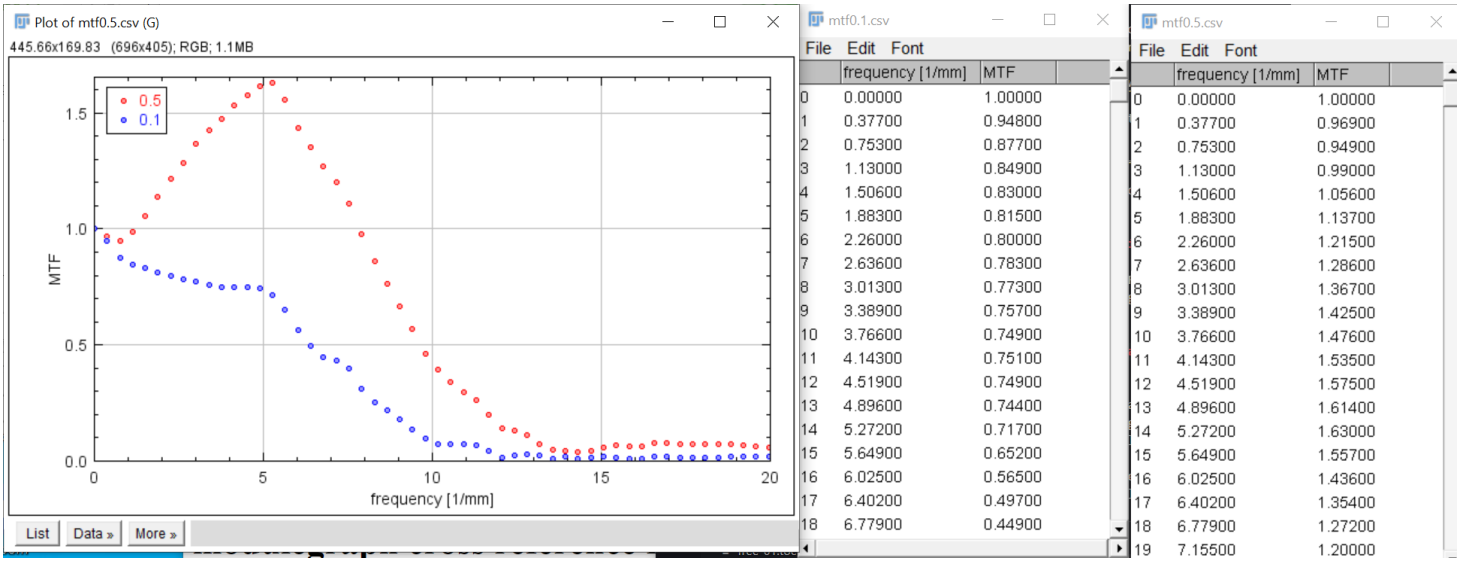
1. Find the output csv file in your file directory then drag and drop it into the ImageJ bar, which will automatically pull it up as a table.

2. Next, right click on the dark gray table header and select `Plot...`, which will automatically pull up the interactive plot shown on the bottom left.

In the ImageJ plotting window you can highlight individual points and adjust the axis limits interactively to better visualize the results.



ImageJ has nice plotting capabilities if you want to compare MTF curves directly from different systems rather than just comparing their 10% cutoff value.

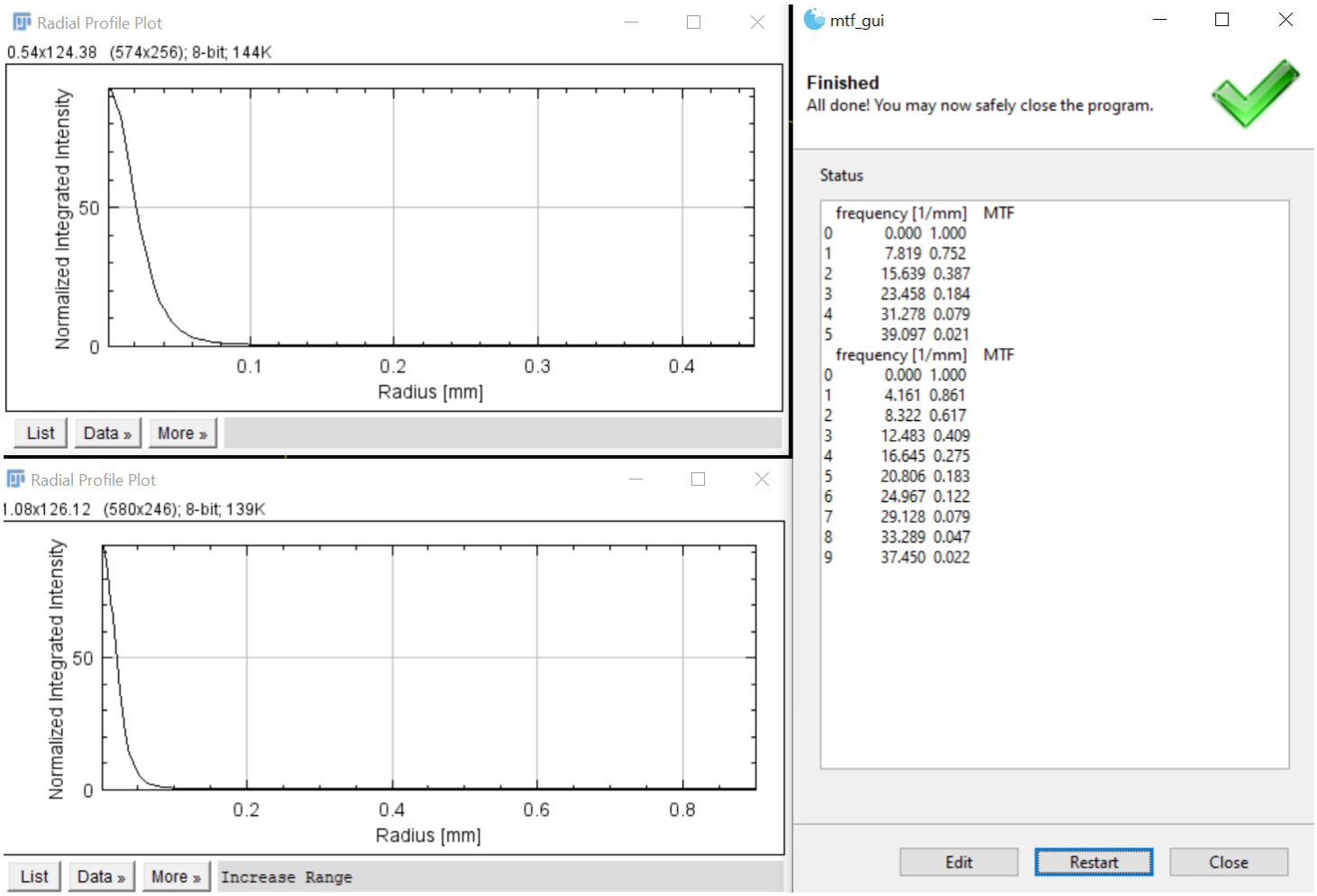


Note: the red curve is edge-enhanced so it has an MTF > 1 at a spatial frequency of around 5 lp/mmm.

Interpreting the MTF curve results

Modulation transfer function (MTF) indicates how well contrast (dark to bright or vice versa) can change for a given spatial frequency where a value of 1.0 is complete transfer (ideal). Using the plot above as an example the low frequencies maintain high MTF, so a bar pattern with a bar pattern frequency of 5 bars per mm (i.e. 5 [1/mm] on the frequency axis) will maintain about 60% of its true contrast while the remainder is lost to blur.

Spatial frequency sampling is determined by profile length



Summarizing a system's spatial resolution: the 10% cutoff

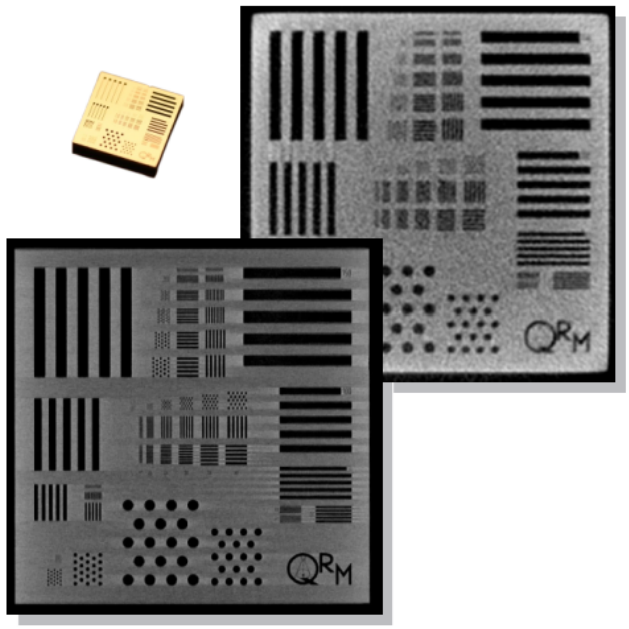
While the full curve contains the full spatial response of the imaging system sometimes it is convenient to be able to report a single number as the spatial resolution of the system. For this the 10% MTF cutoff is considered a good approximation of the finest detail that can be reasonably resolved. For the results above you can check either the table readout or the plot to find the spatial frequency where the MTF first dips below 10%, here is ~ 13 [1/mm] or 13 [lp/mm].

To see how this result translates to the smallest resolvable bar pattern we can refer to the QRM bar pattern table:

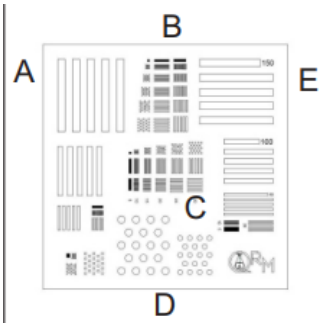
The different structures on the chip are arranged in such a way over the chip, that spatial resolution can be evaluated in the center as well as in the periphery of the image/chip in a single measurement.

linewidth [µm]	linepairs / mm
5	100
10	50
15	33.3
20	25
25	20
30	16.6
50	10
100	5
150	3.3

Bar / line pattern on the silicon chip



Micro-CT scans in air (left) with 5.5 µm voxel size and in resin (right) with 40 µm voxel size



Block	linewidth (µm)	linepairs per pattern	points (µm)	points per pattern
A	5, 10, 25, 50, 100, 150	5		
B	5, 10, 15, 20, 25, 30	5	5, 10, 15, 20, 25, 30	18
C	5, 10, 15, 20, 25, 30	5	5, 10, 15, 20, 25, 30	18
D			5, 10, 25, 50, 100, 150	18
E	5, 10, 25, 50, 100, 150	5		

Thus $MTF_{0.1} = 13$ [1/mm] works out to be somewhat better than the 50 um linewidth bar patterns. Thus in the A block the first 3 patterns should be clearly visible, while smaller patterns should be harder to resolve.

Other interpretations: Nyquist's theorem

Nyquist's theorem theorem is commonly used in signal processing to relate the system's sampling rate f_{sample}, inversely related to sampling period (e.g. voxel size in CT $\Delta d = 1/f_{sample}$) to the smallest structures it can reasonably detect, Δx, the size of the structure with max frequency $B = 1/\Delta x$. These are all related by the following:

$$\Delta x > 2/f_{sample} = 2 * \Delta d$$

The scan that I grabbed that image from had a voxel size of $\Delta d = 25 \mu m$, thus using this relation we could estimate the smallest resolvable structures to be about $50 \mu m$.

However we can compare this to our 10% MTF results which empirically tell us what the *actual* smallest structure we can resolve is, by $\Delta x \approx \frac{1}{MTF_{0.1}} = 77\mu m$. This is close to but not exactly our voxel size which gives us our theoretical max resolution. Other factors from the system such as focal spot size, geometry misalignment, reconstruction kernel could cause this additional blurring.

Thus the MTF is a valuable tool to give you real measurements of a system's spatial resolution to compare amongst its theoretical value as well as between systems. It can also be used to compare reconstruction kernels or look for specific frequencies other than $MTF_{0.1}$ that are enhanced.

More on MTF

From Kishore:

Hi Brandon,

Thanks for looping me in. You are right that 1/mm is same as lp/mm. The plot you show uses 1/cm or lp/cm unit in the x axis.

The true smallest resolvable object is identified using the cutoff frequency (i.e. the lp/cm or lp/mm when MTF becomes zero). For a 0% MTF = 40 lp/cm, the resolution in um is calculated as 0.5/40 (note it should be 0.5 in the numerator and not 1.0) since the unit is line "pair", so each line in a pair, if resolved, is half of the pair (0.5), which I think you referred to as the blur factor. The limiting resolution is therefore 125 um for 0% at 40 lp/cm. Trivia- this is actually the resolution of Alpha with the sharpest available kernel.

There is more to this, but without confusing you any further, please remember this detail - I have often seen the misconception (in uCT world) that the detector pixel size or the recon voxel size is the system's resolution. For instance, on Alpha we can get voxels as small 48 um (x-y direction), the detector pixel size is 151 um (isocenter), and the actual final image resolution is >= 125 um depending on the recon kernel. Voxel size and detector size only indirectly play a role in bringing out the best system resolution, but their individual sizes are not the final resolution of the system.

References

1. Richard S, Husarik DB, Yadava G, Murphy SN, Samei E. Towards task-based assessment of CT performance: system and object MTF across different reconstruction algorithms. Med Phys. 2012;39(7):4115-4122. https://doi.org/10.1118/1.4725171