Prof. Raghuveer Parthasarathy
University of Oregon; Fall 2024

Homework 4: Additional Notes

2 A high-resolution PSF.

We want to make a function that calculates the PSF:

```
def psf2D(N, scale = 0.1, NA = 0.9, lam = 0.53):
```

As noted in the suggestion, we'll first create a 2D array of distances from the center, ***r***, in "real" units (microns). Before we do this, we'll want a 2D array of distances from the center, in pixels; let's call this ***d***. We can create this using `numpy.meshgrid` to make 2D arrays of x and y distances from the center, from which we calculate *d*.

If $N = 5$, for example, the "x" pixel distances are:

```
[[-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]
 [-2. -1.  0.  1.  2.]]
```

the "d" distances are

```
[[2.82842712 2.23606798 2.         2.23606798 2.82842712]
 [2.23606798 1.41421356 1.         1.41421356 2.23606798]
 [2.         1.         0.         1.         2.        ]
 [2.23606798 1.41421356 1.         1.41421356 2.23606798]
 [2.82842712 2.23606798 2.         2.23606798 2.82842712]]
```

and the "r" array is `r = d*scale`, since `scale` is the microns per pixel. Make sure your code reproduces this.

The v values are then simply given by

```
v = (2*np.pi/lam)*NA*r,
```

since that's what v is, by definition.

Calculating the PSF uses a Bessel function:

```
psf = 4*(scipy_special.j1(v)/v)**2
```

However, there's one part to be careful of: where v==0, we'll get a divide by zero error. We therefore have to manually set the psf to 1 where v==0. (See note #2)

Finally, though it's not required, you should normalize the array so that the sum of all the elements is 1. This is simple:

```
psf = psf/np.sum(psf)
```

(Think about this.)

When running this, note that setting *N* and requiring that the array spans about 1 micron sets the value of `scale`. For example, if $N = 101$, `scale` is about 0.01 microns/px, or 10 nm/px. Actually it's 0.0099, but it will be convenient later if the value is easily divisible into 100 nm/px, so we'll use 0.01 microns/px and an overall size of N x 0.01 = 1.01 microns.

4 Simulated point sources, Part 1. (...

(a) ...Now create a grid corresponding to actual, typical camera pixel ...

If, for example, we used 0.01 microns/px in #2, and now we're pixelating with 0.1 um/px, we sum each 10x10 block of the "original" psf into a pixel.
Element [0,0] of the pixelated psf is therefore

```
numpy.sum(psf[0:10, 0:10])
```

Element [1, 0] is

```
numpy.sum(psf[11:20, 0:10])
```

and so on...