Qbio 2015
SPT and SR Microscopy

Keith Lidke (kalidke@unm.edu)
Mark Olah (mjo@cs.unm.edu)

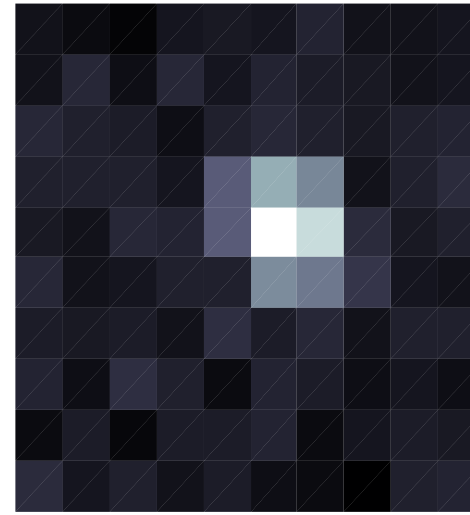
Estimate the center of the PSF from the detection on the camera

- Approaches:
  - Use center of mass from the intensity
  - Use a model based fitting approach

- Image formation process:
  1. Photons arrive at camera
  2. Photons converted to electrons
  3. Electrons are readout and amplified

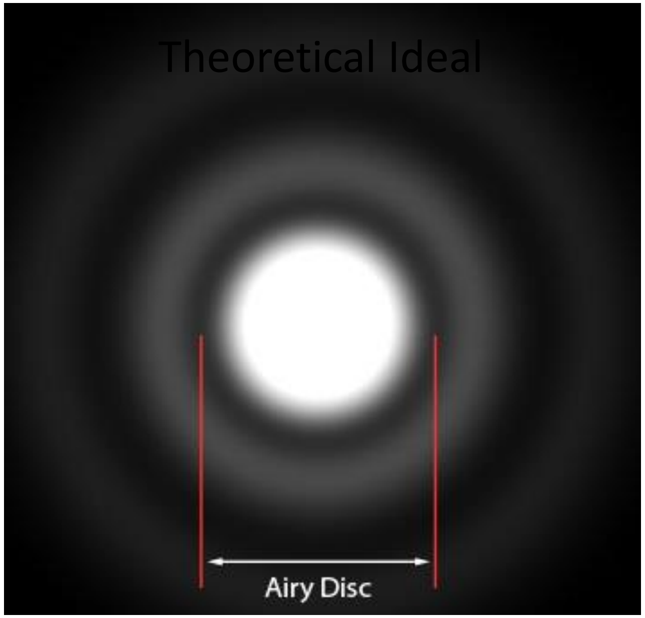
  - Poisson noise from photons & Gaussian noise from read-out electronics. Need to use lowest possible readout noise => EM-CCD

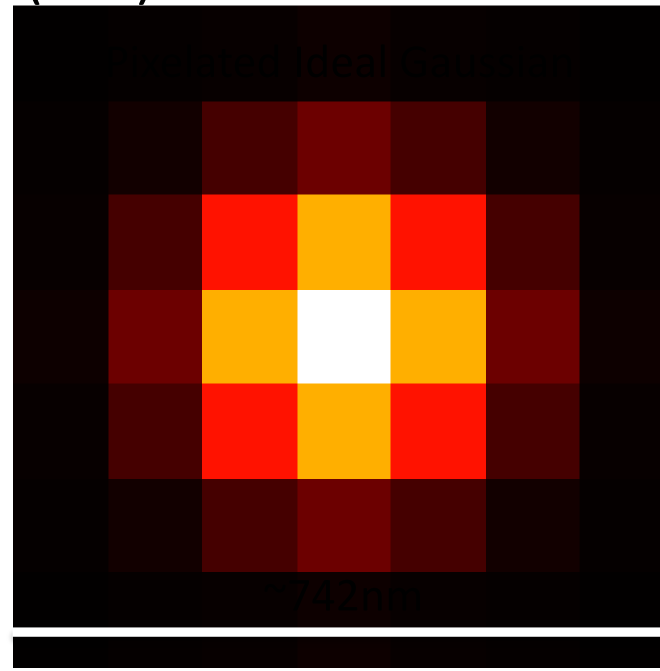
  - Poisson noise is dominant part in practice!

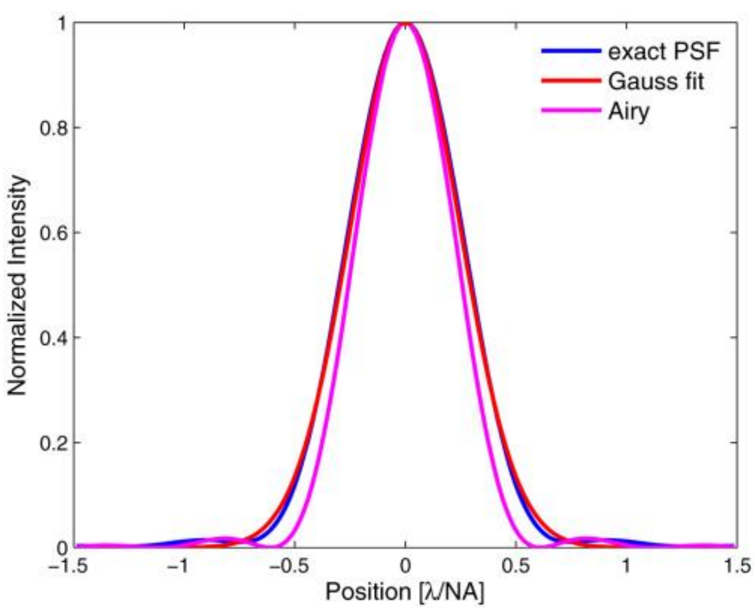


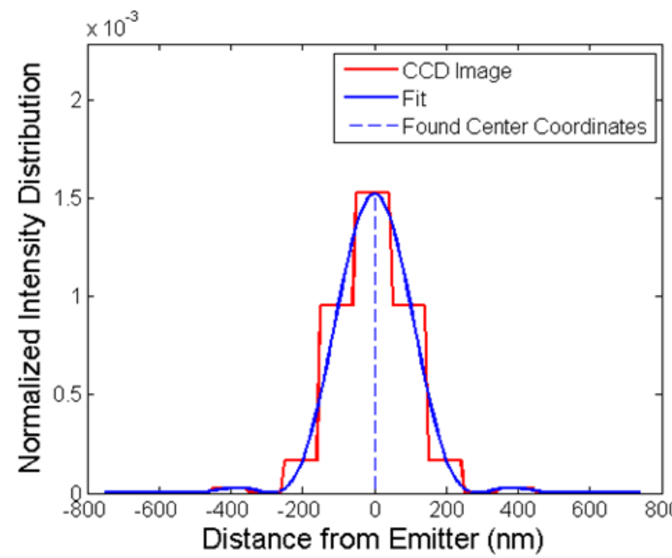
Single Point Emitters Appear as Diffraction-Limited Blobs

Point Spread Function (PSF)









Fitting of Diffraction Limited Blobs

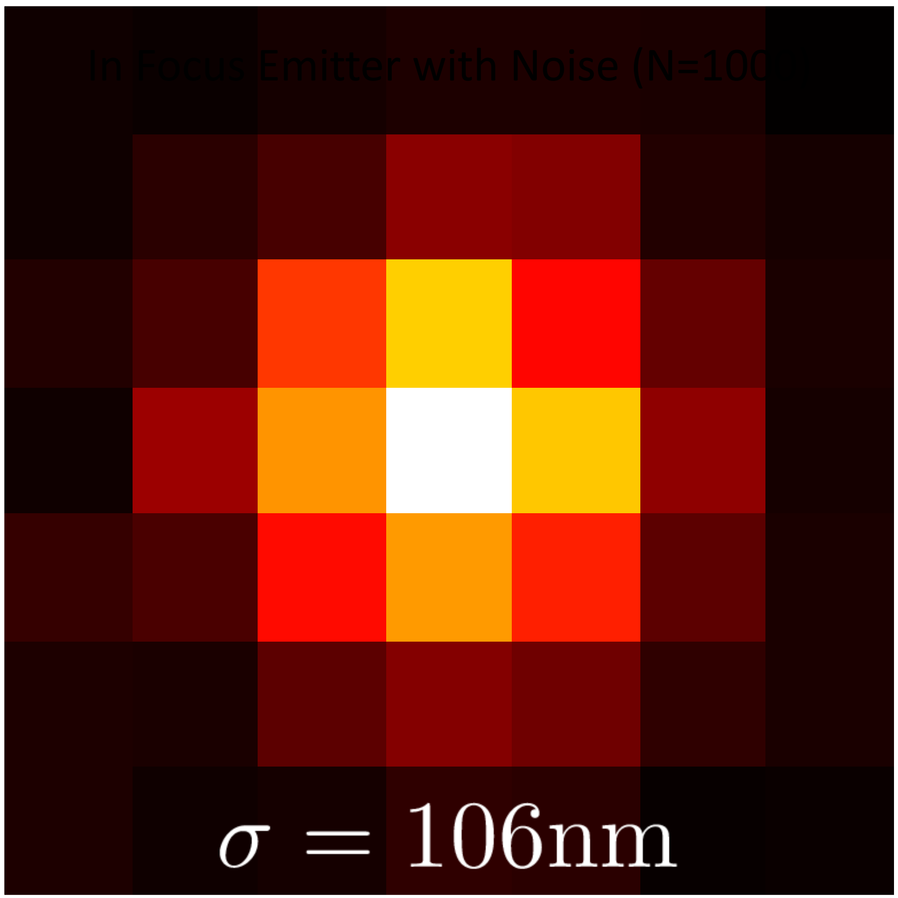
Numerical Aperture: $n_a \leq 1.4$

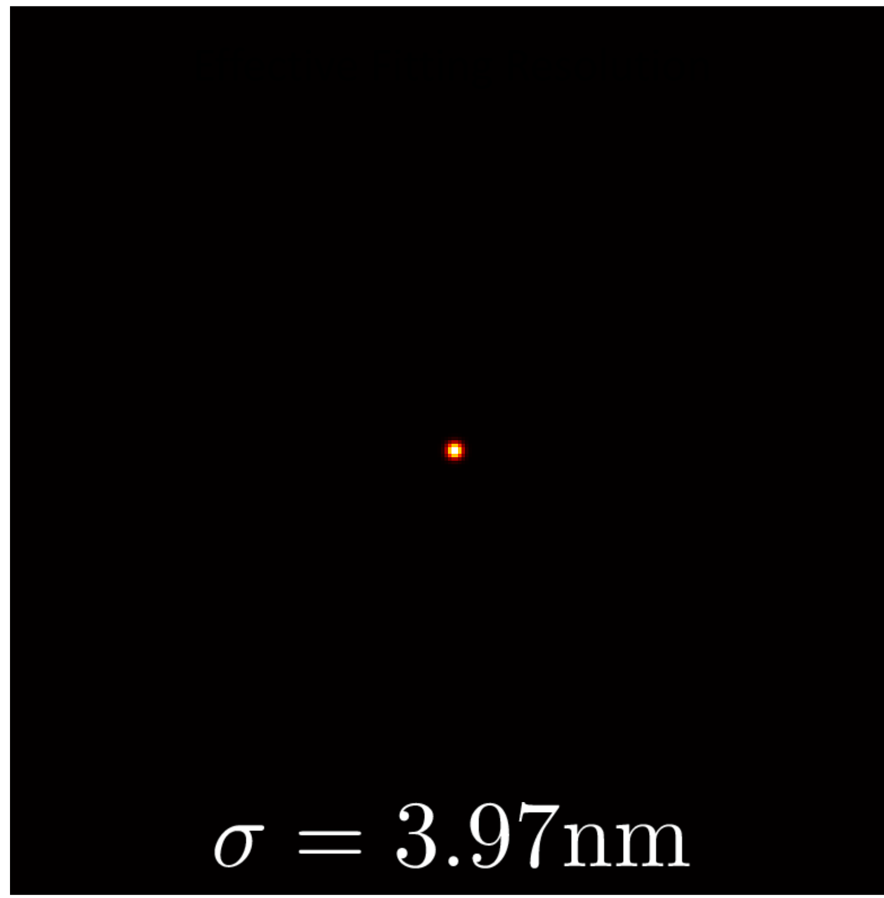
Emission Wavelength: $480 \leq \lambda_{em} \leq 750$

Number of photons: N

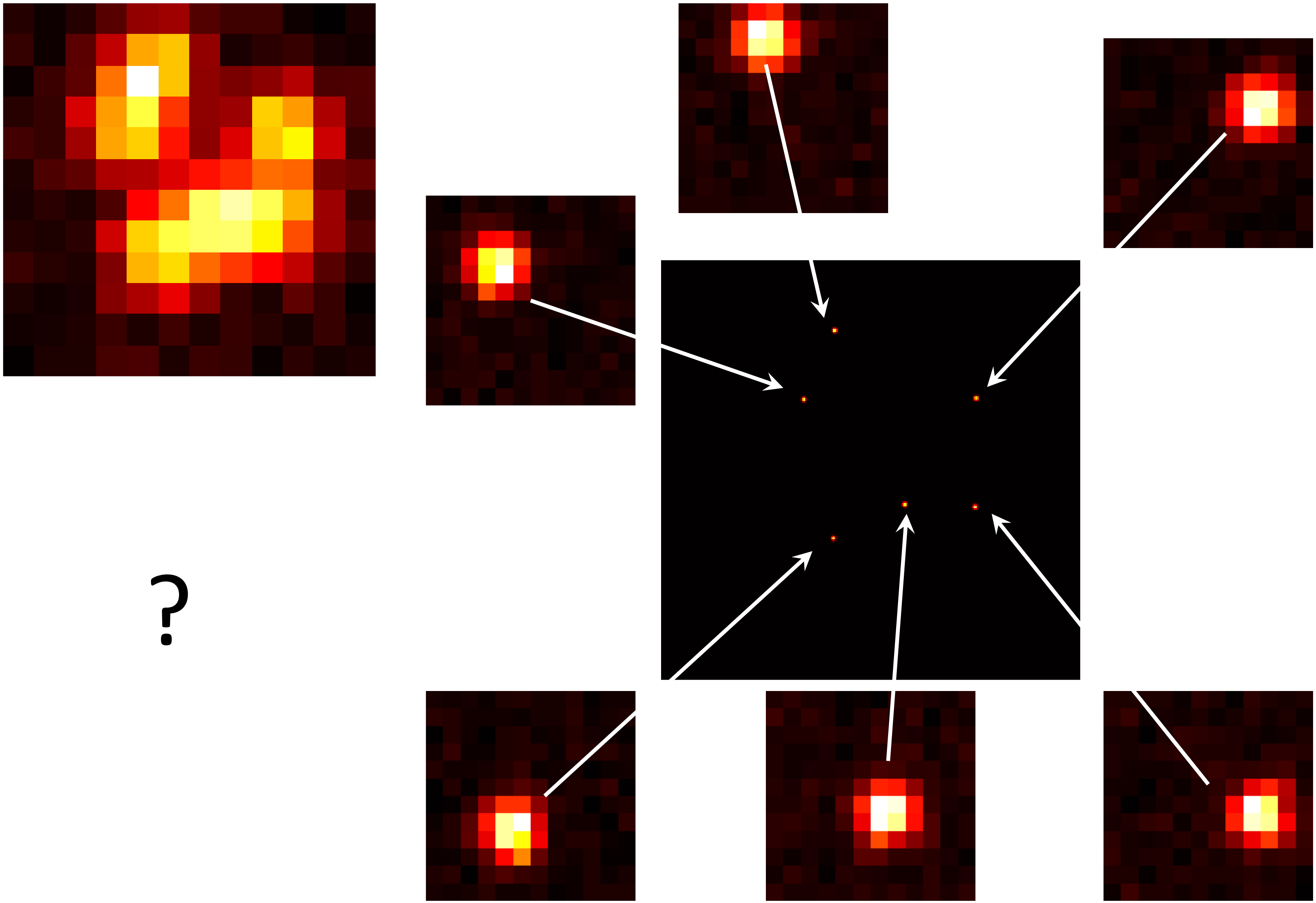
~742nm

Localization precision:



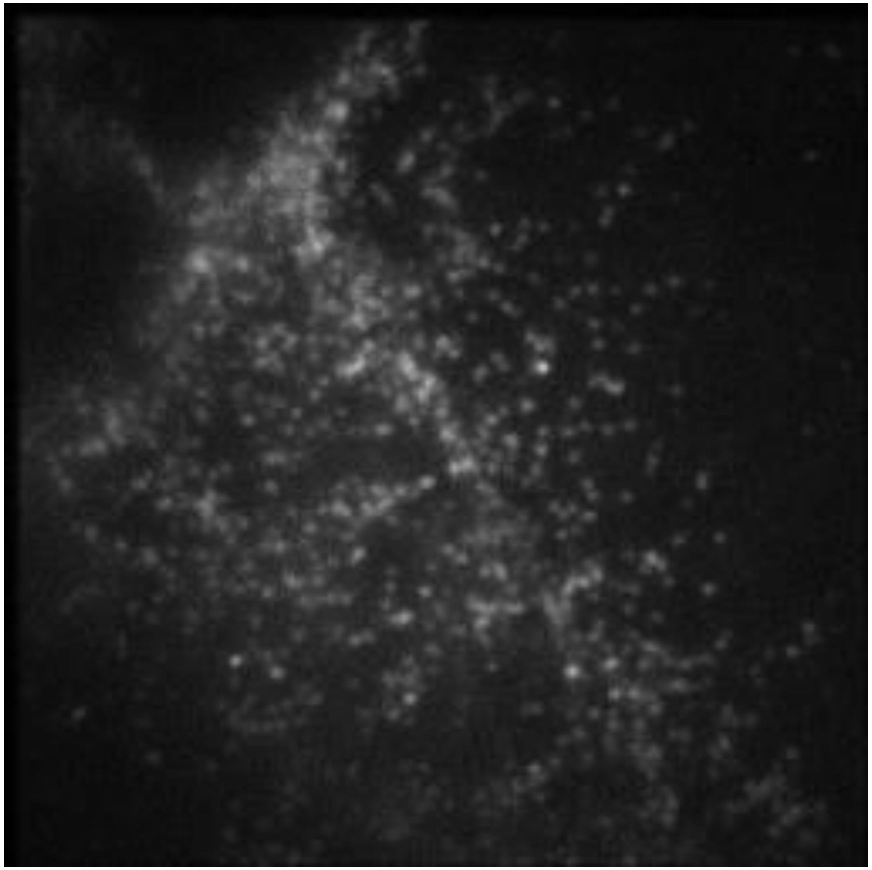


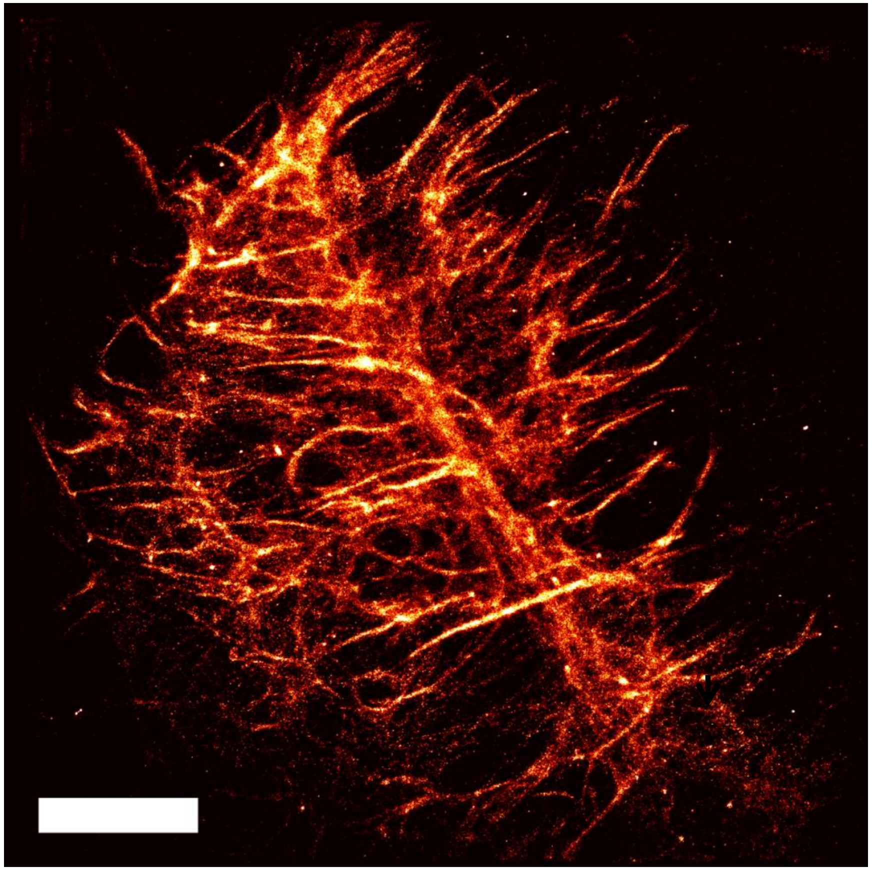
Problem: This only works when each emitter is spatially isolated



What Does a Super-Resolution Image Look Like?

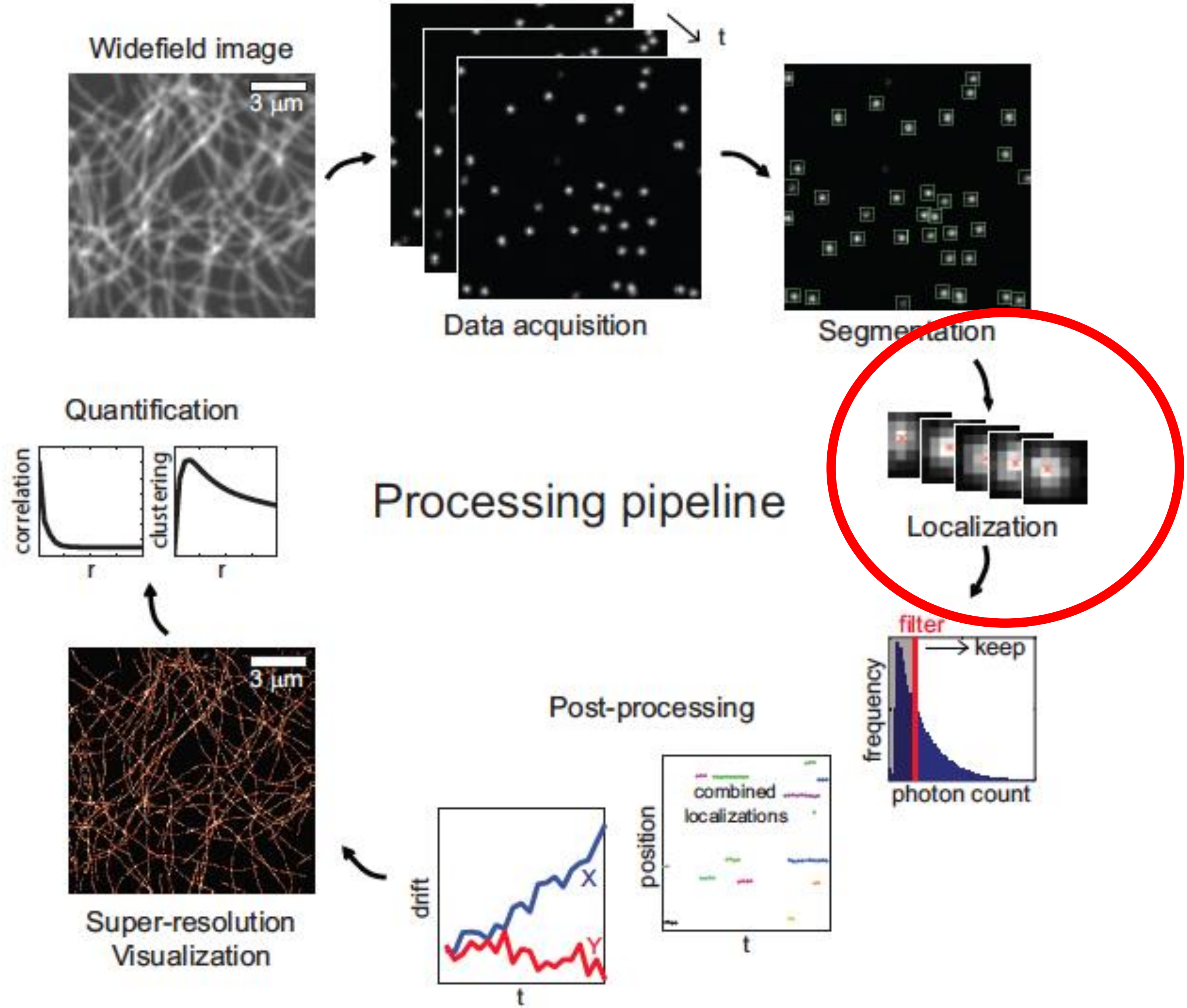
Blinking Fluorophores





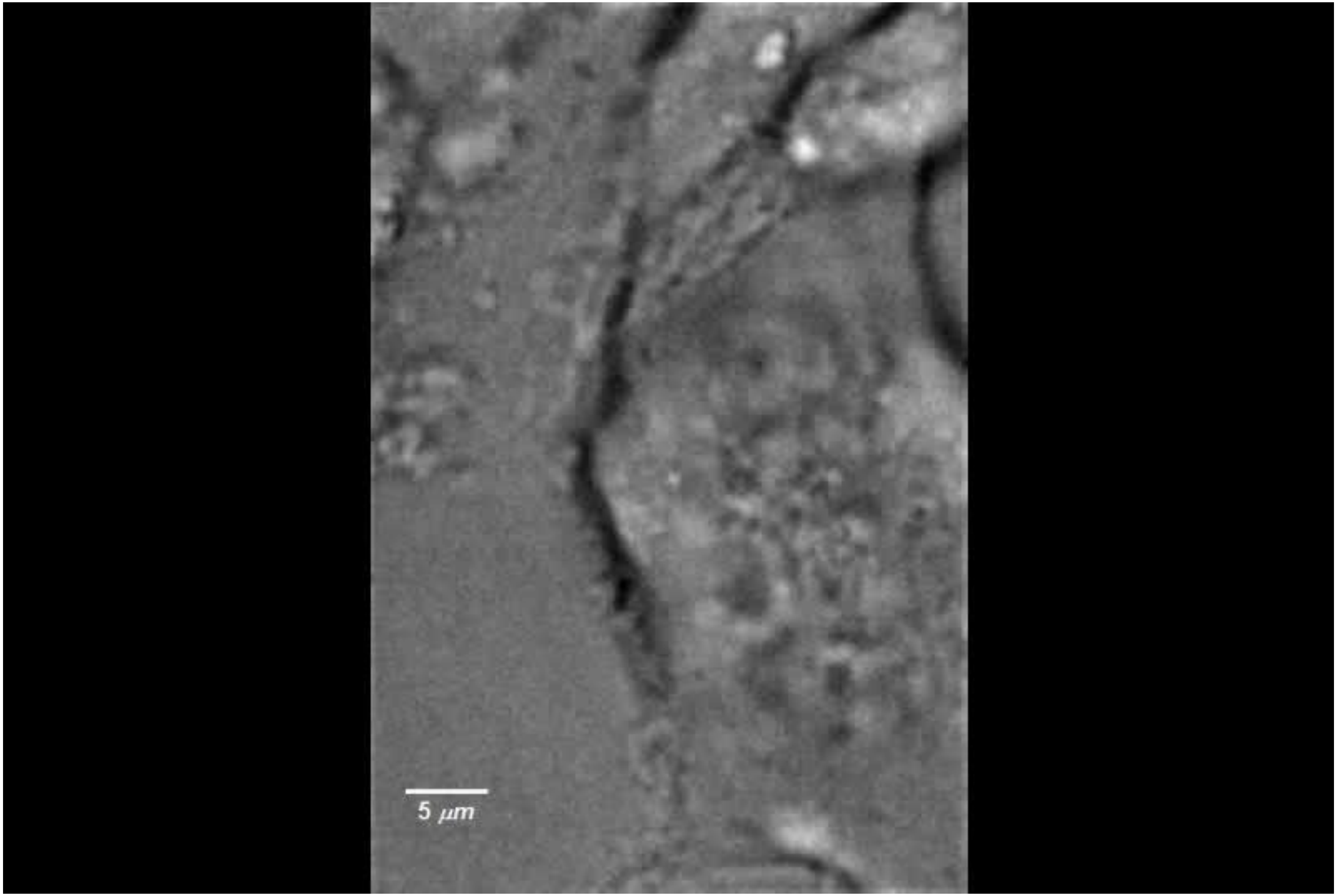
5 μm

Huang *et al*, Biomedical Optics Express 2011



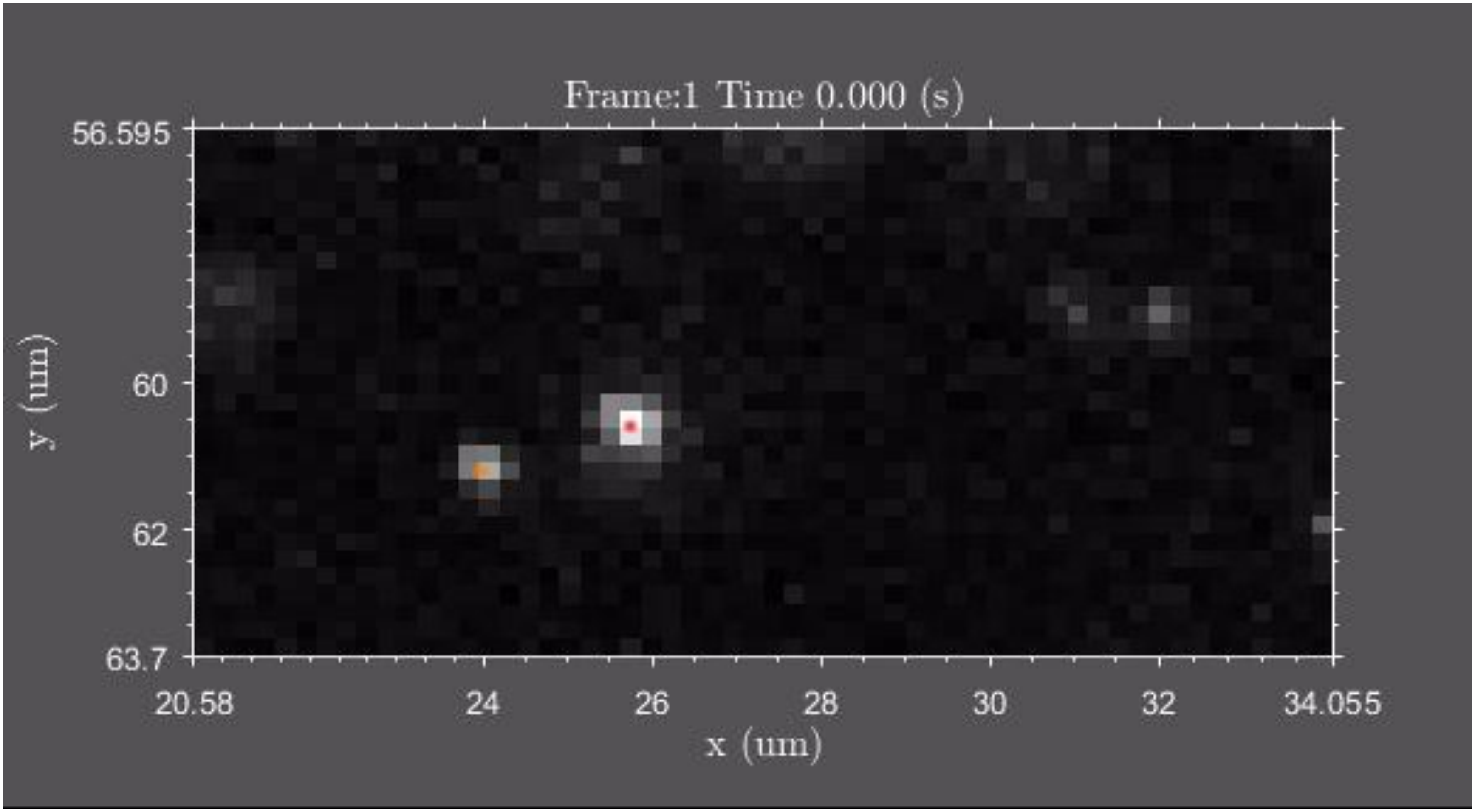
Widefield image
3 µm
Data acquisition
t
Segmentation
Localization
filter
keep
frequency
photon count
Post-processing
combined
localizations
position
t
drift
X
Y
t
Super-resolution
Visualization
3 µm
Quantification
correlation
r
clustering
r
Processing pipeline

Idea: Single Particle Tracking In Live Cells



Single Particle Tracking

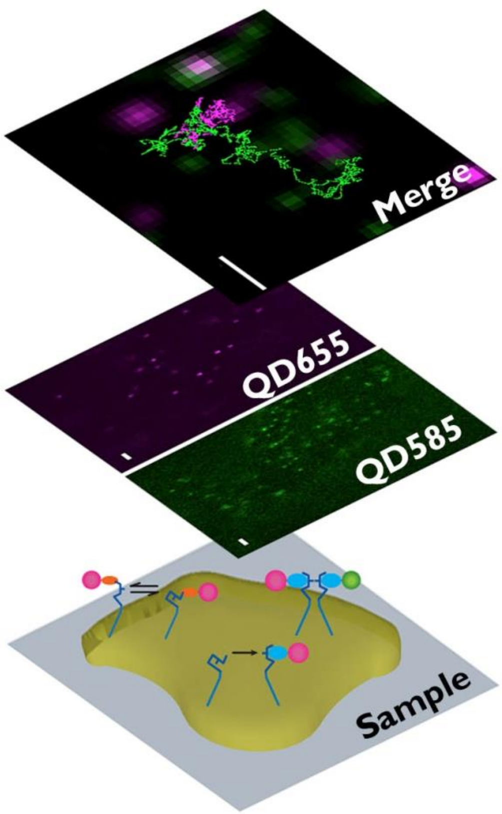
- Try to keep density of fluorophores low so they are spatially separated
- Track individual molecules over course of movie
- Still will have problems whenever trajectories intersect

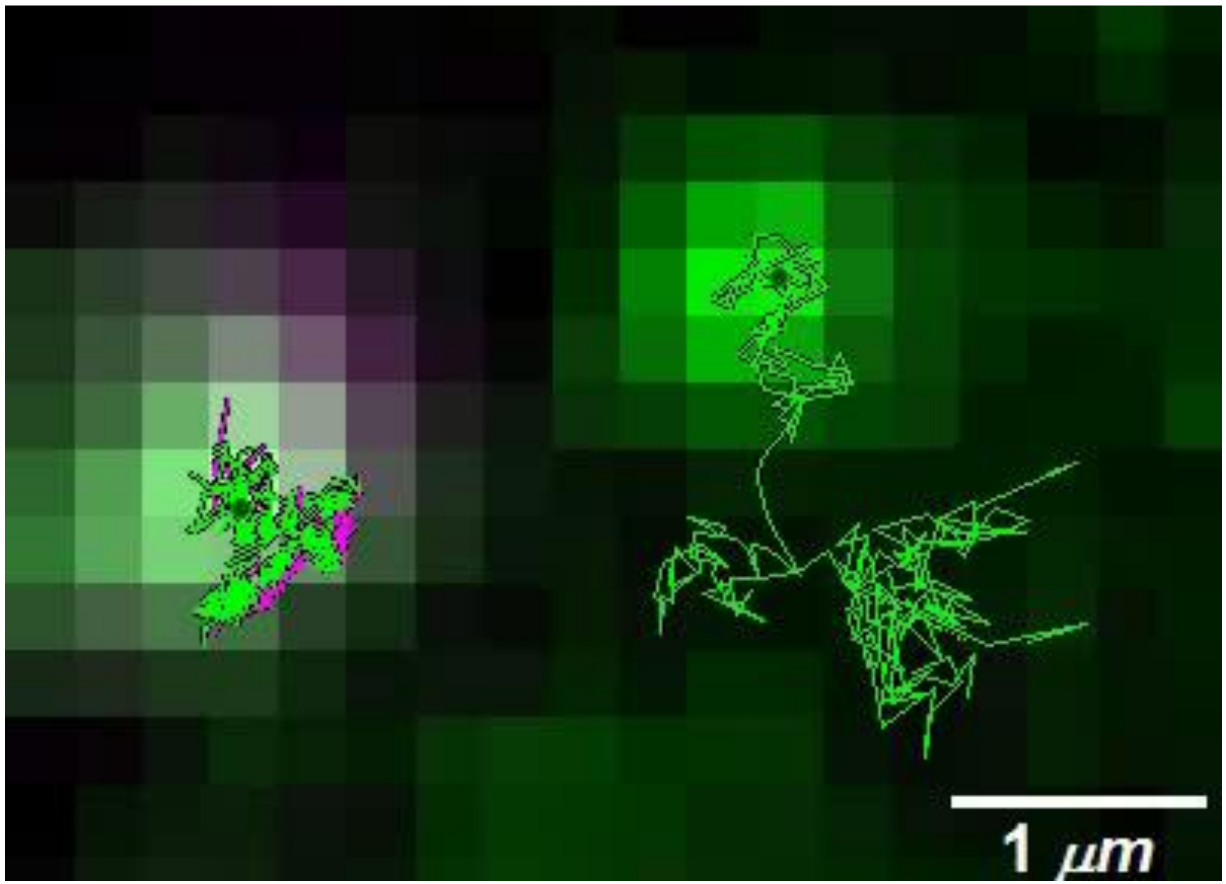


Two-Color Single Particle Tracking

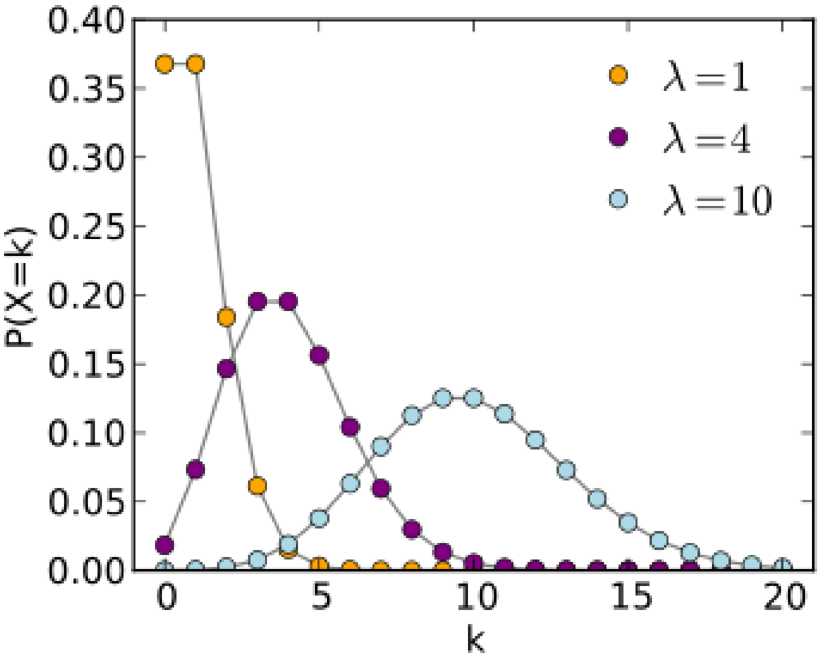
Idea: If we had more than one color we could distinguish overlapping trajectories

- EMCCD camera is Black-and-White
    - Color of capture is controlled with a filter
- Use 2 different colors of emitters
- Split CCD in half and allow for 2 different light-paths that have separate filter colors





Poisson Distribution



Discrete distribution --- Gives probability of observing k events over a fixed time period if mean emission rate over that period is lambda.

Examples:

- Number of packets arriving on a network connection
- Number of radioactive decay events in a sample
- Number of photons emitted from a fluorophore

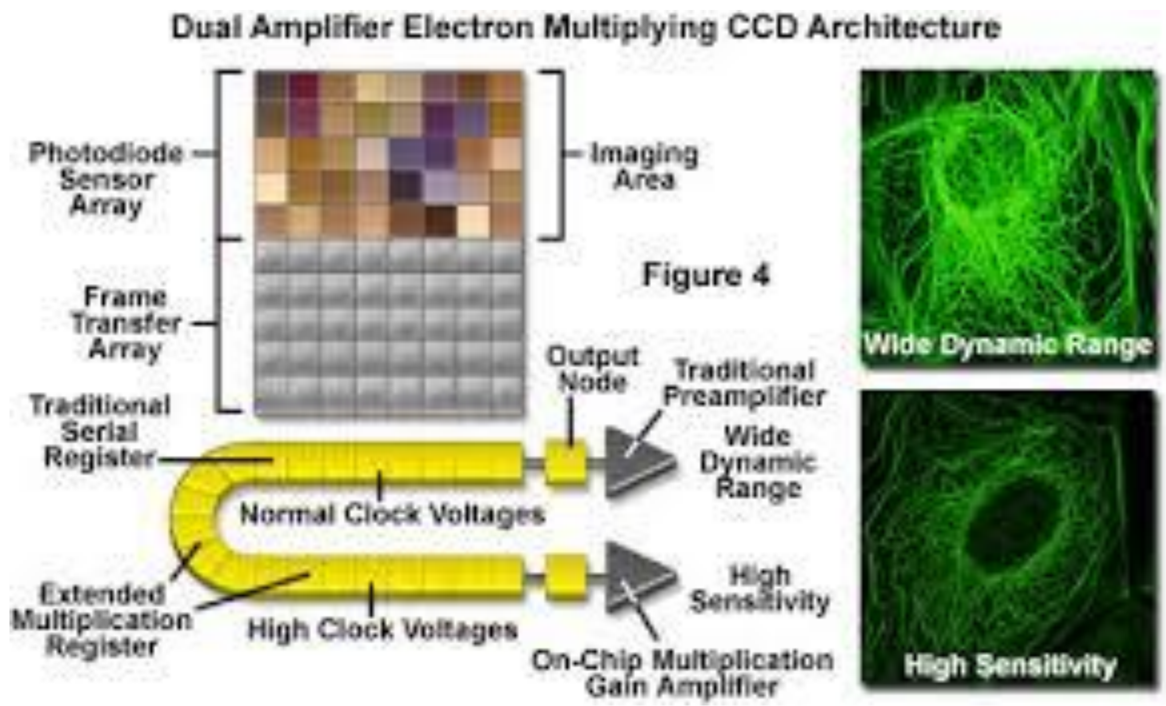
$$\mathrm{P}(X = k|\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$

$$\mathrm{E}(X) = \mathrm{Var}(X) = \lambda$$

[Step 1] EMCCD Gain Calibration

EMCCD cameras used in fluorescence microscopy are designed to be extremely sensitive to low intensity signals.

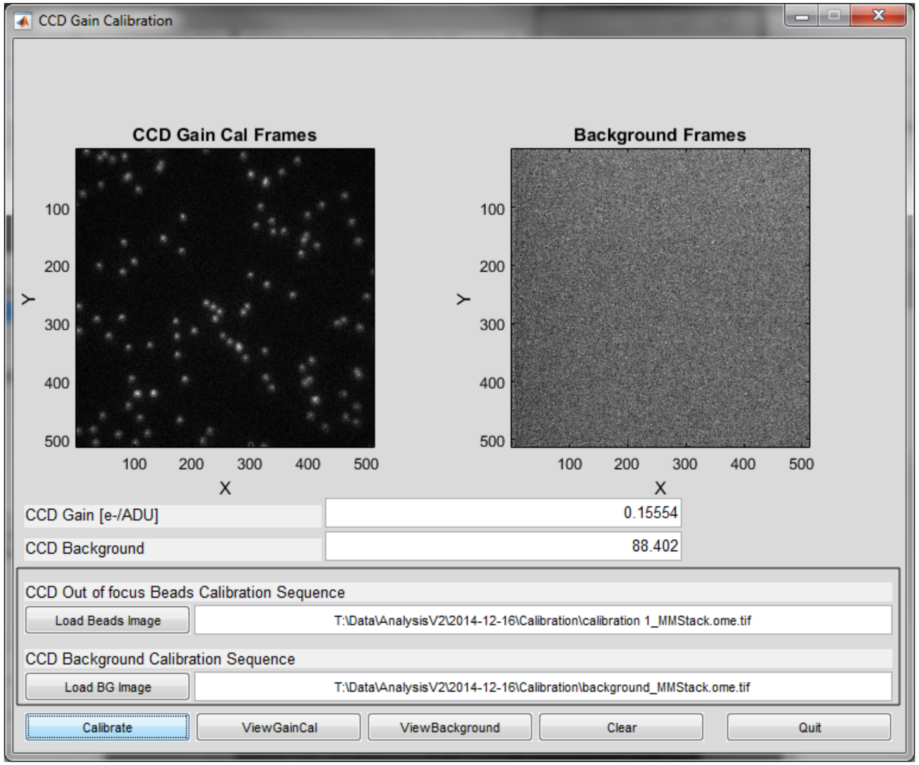
- Operate at -90°C to reduce sensor noise
- Every individual photon is important information!
- Effectively black-and-white
- Output is in 16-bit unsigned ADU (analog-to-digital-units)
- Gain calibration converts: ADUs →photon counts

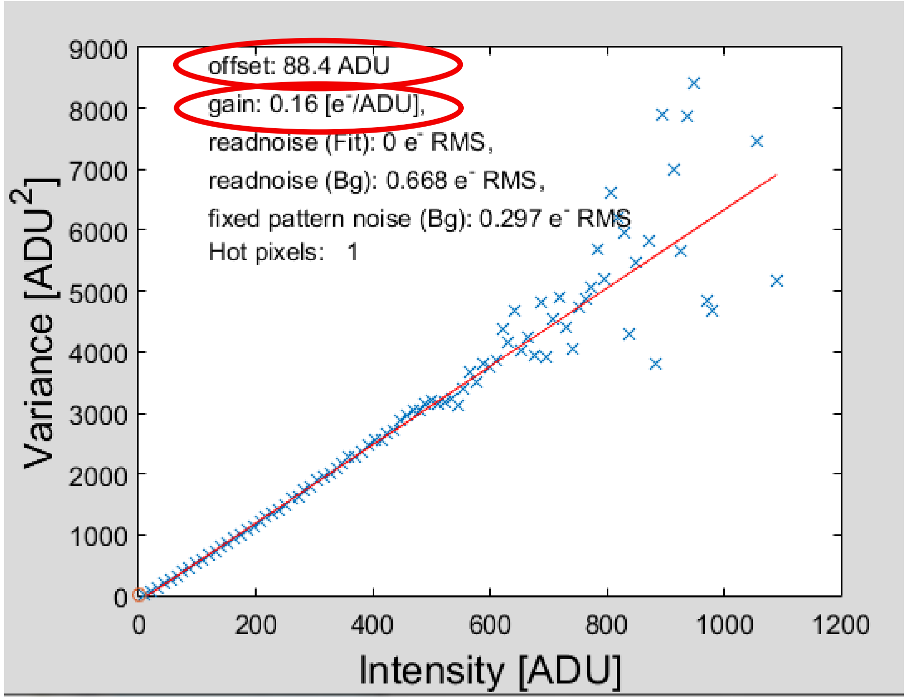


[Step 1] EMCCD Gain Calibration

EMCCD cameras used in fluorescence microscopy are designed to be extremely sensitive to low intensity signals.

- Operate at -90°C to reduce sensor noise
- Every individual photon is important information!
- Effectively black-and-white
- Output is in 16-bit unsigned ADU (analog-to-digital-units)
- **Gain calibration converts: ADUs → photon counts**



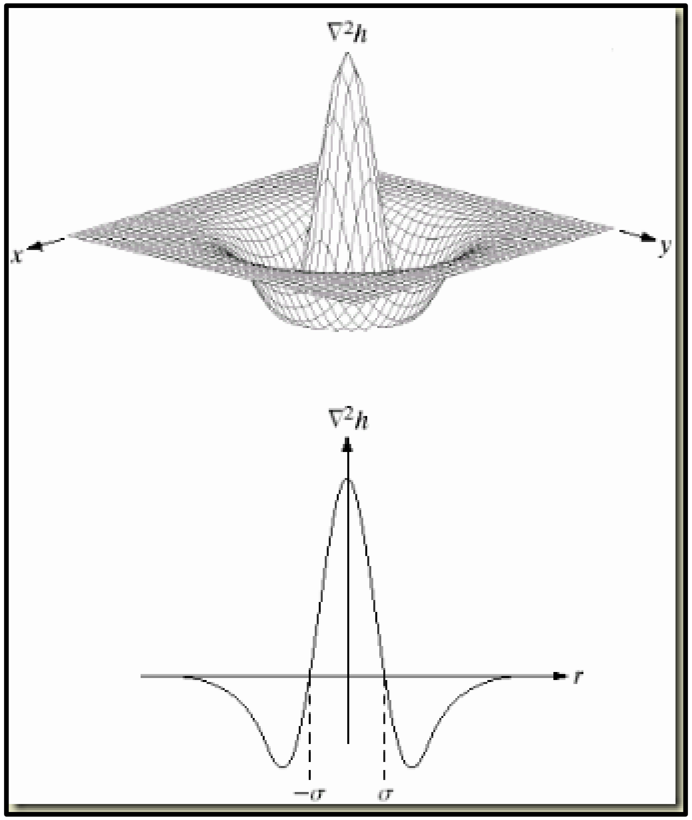


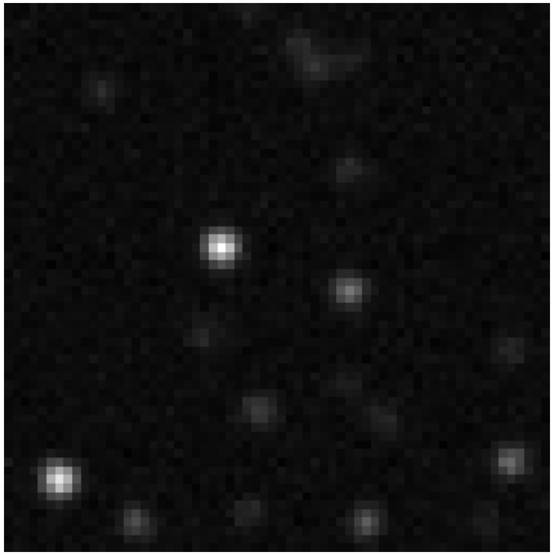
[Step 2] Identify Likely Single Emitters Sub-regions

Image Processing – Feature Detection

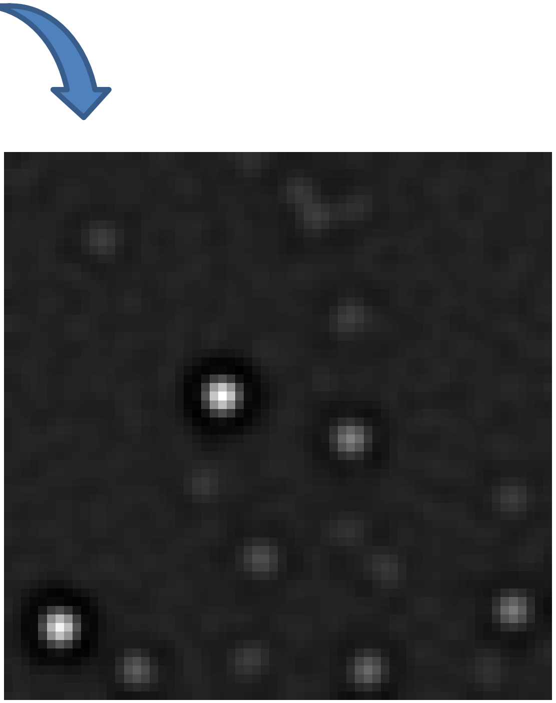
- Spot or Blob detection
- Apply filter of expected feature size to enhance desired features in image
- Ideal filter is Laplacian of Gaussian (LoG)

Laplacian of Gaussian (LoG) Filter





Original

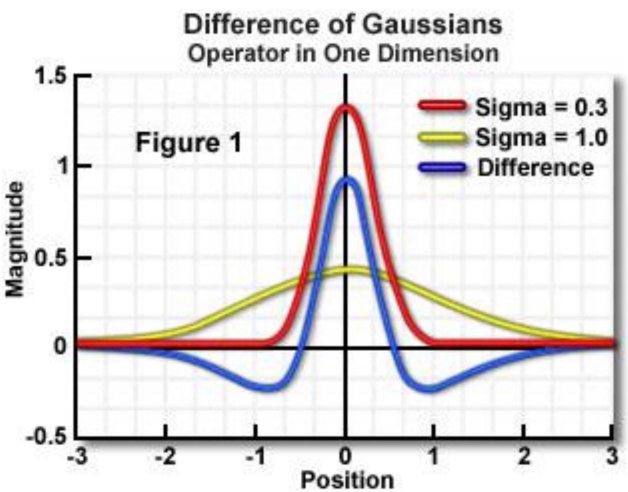


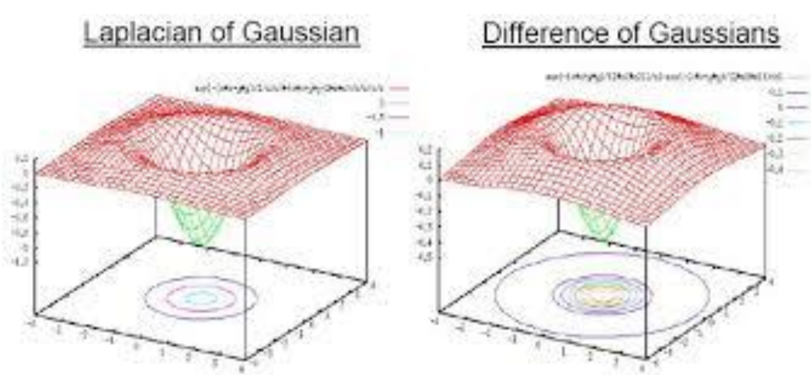
LoG Filtered

[Step 2] Identify Likely Single Emitters Sub-regions

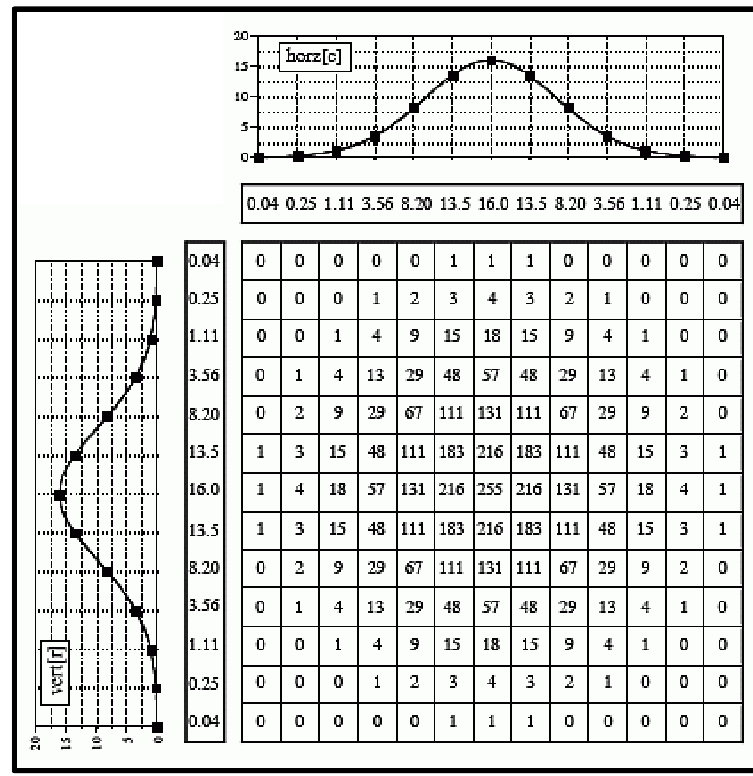
Image Processing – Feature Detection

- LoG can be approximated by Difference of Gaussians (DoG)
- DoG is more computationally efficient as it is separable across dimensions
- Separability is important for hyperspectral (3D) data!
- Note: There is a separable LoG implementation with 4-passes in 2D and 9-passes in 3D





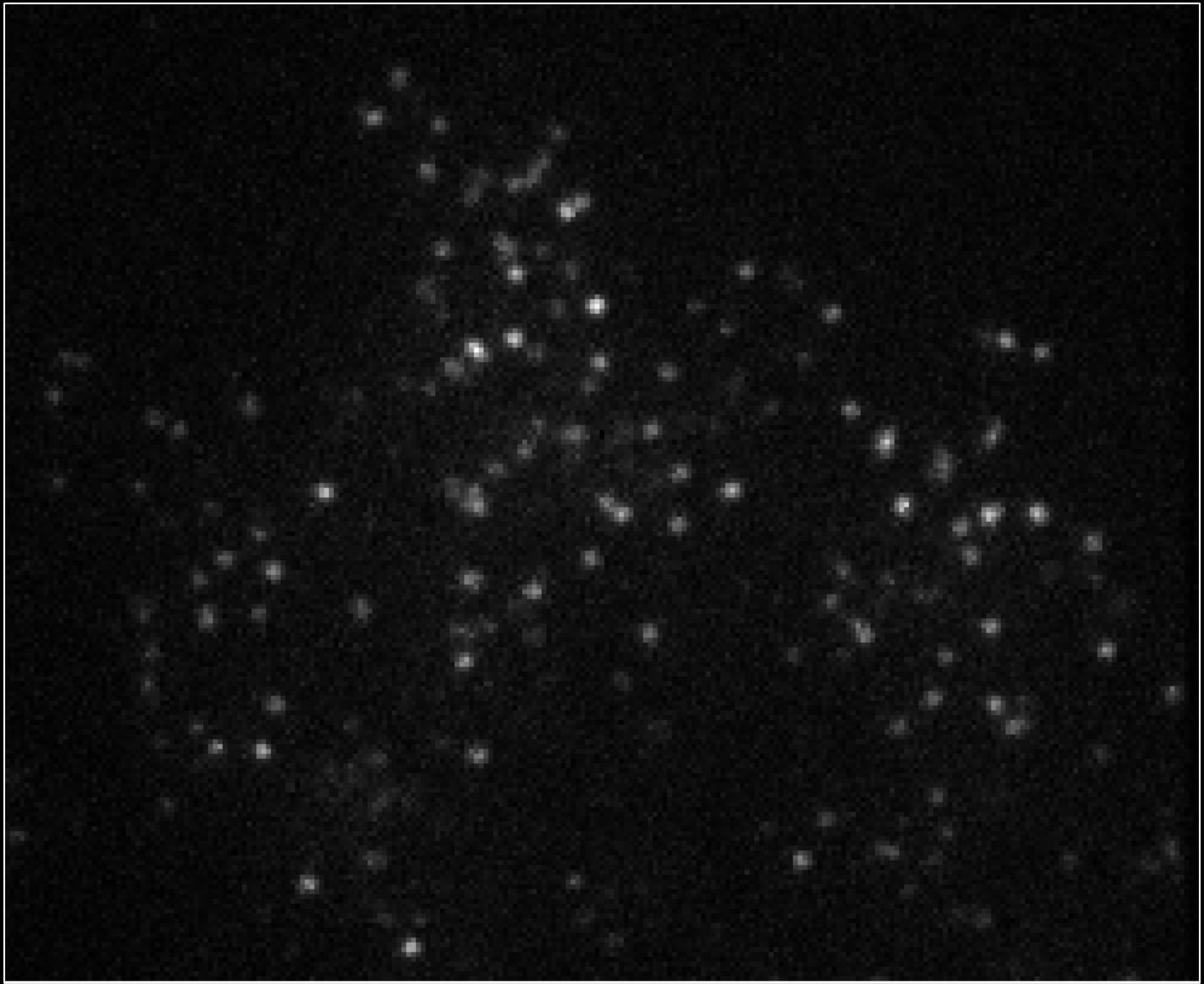
Gaussian is Separable



	0.04	0.25	1.11	3.56	8.20	13.5	16.0	13.5	8.20	3.56	1.11	0.25	0.04
0.04	0	0	0	0	0	1	1	1	0	0	0	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
16.0	1	4	18	57	131	216	255	216	131	57	18	4	1
13.5	1	3	15	48	111	183	216	183	111	48	15	3	1
8.20	0	2	9	29	67	111	131	111	67	29	9	2	0
3.56	0	1	4	13	29	48	57	48	29	13	4	1	0
1.11	0	0	1	4	9	15	18	15	9	4	1	0	0
0.25	0	0	0	1	2	3	4	3	2	1	0	0	0
0.04	0	0	0	0	0	1	1	1	0	0	0	0	0

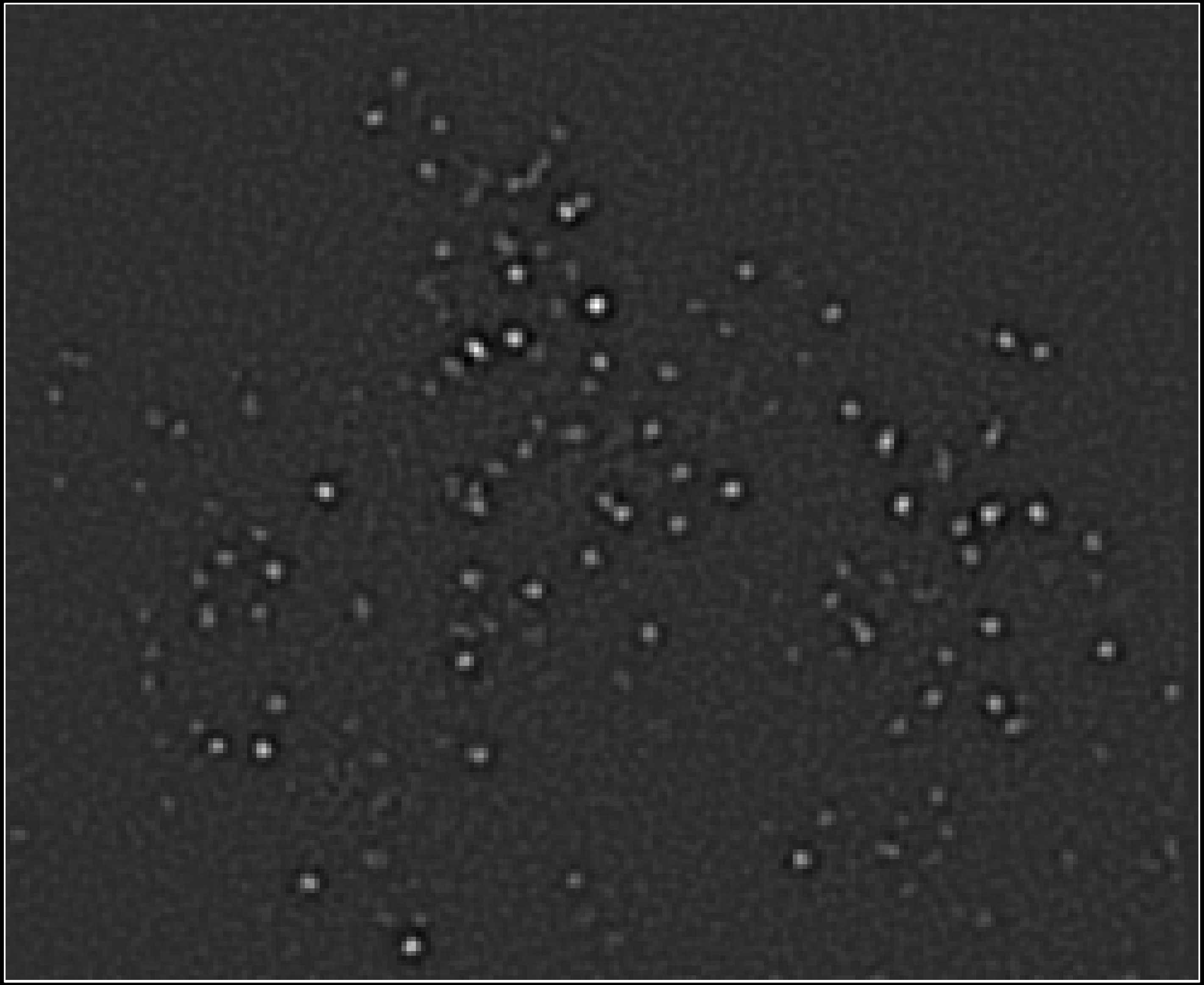
[Step 2] Identify Likely Single Emitters Sub-regions

Gain Corrected Image



[Step 2] Identify Likely Single Emitters Sub-regions

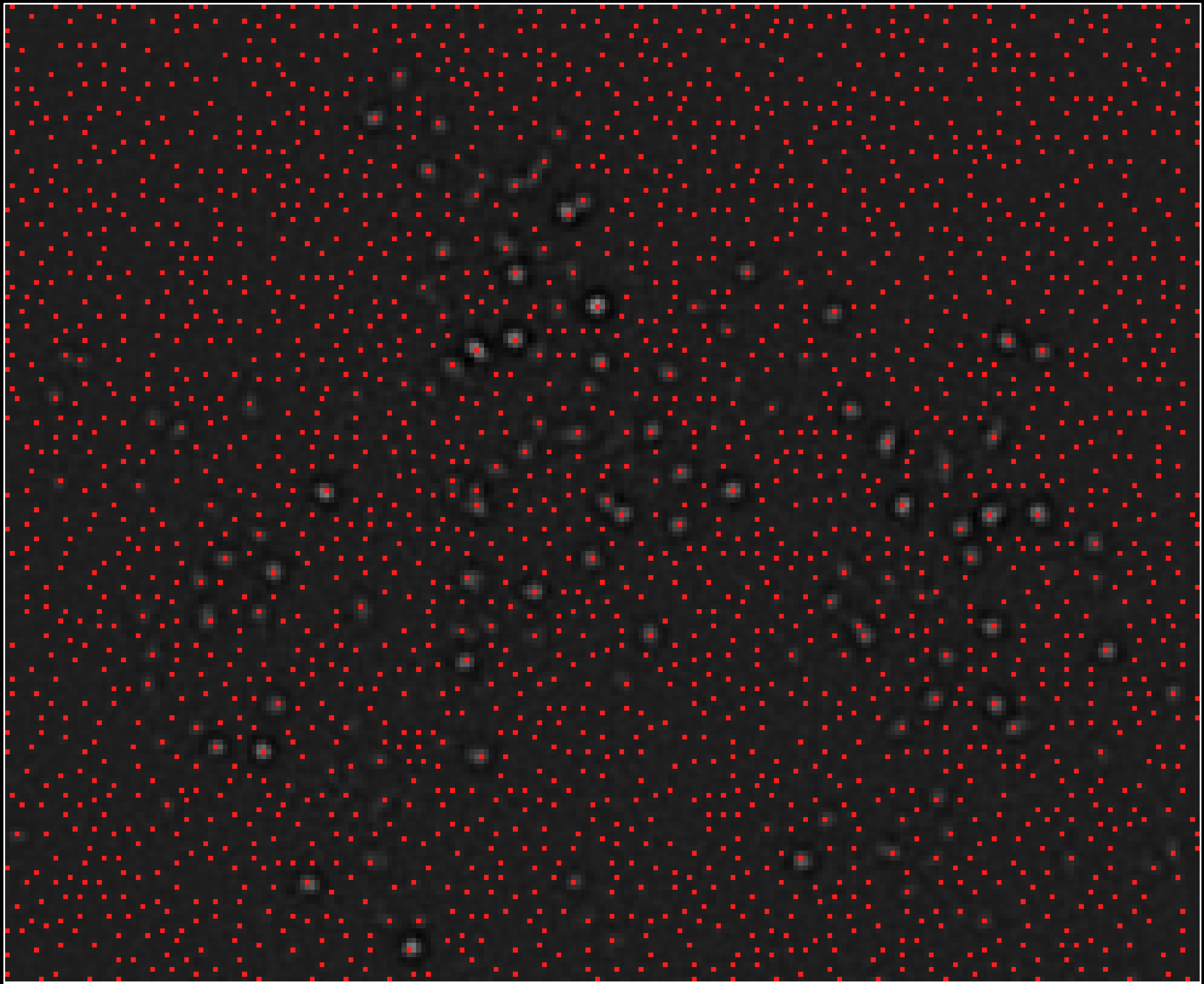
DoG Filtered Image



[Step 2] Identify Likely Single Emitters Sub-regions

Identify Local Maxima

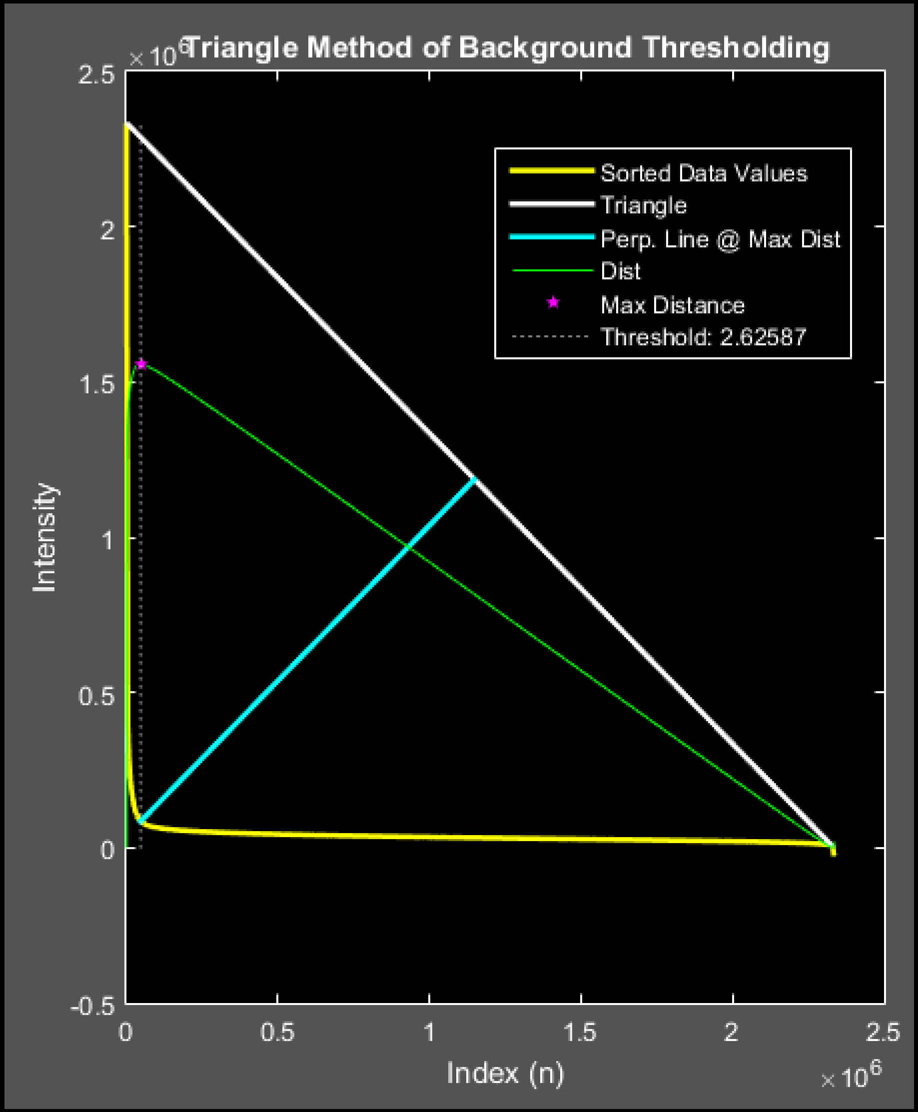
(Note: can be done with amortized 2 comparisons per pixel)



[Step 2] Identify Likely Single Emitters Sub-regions

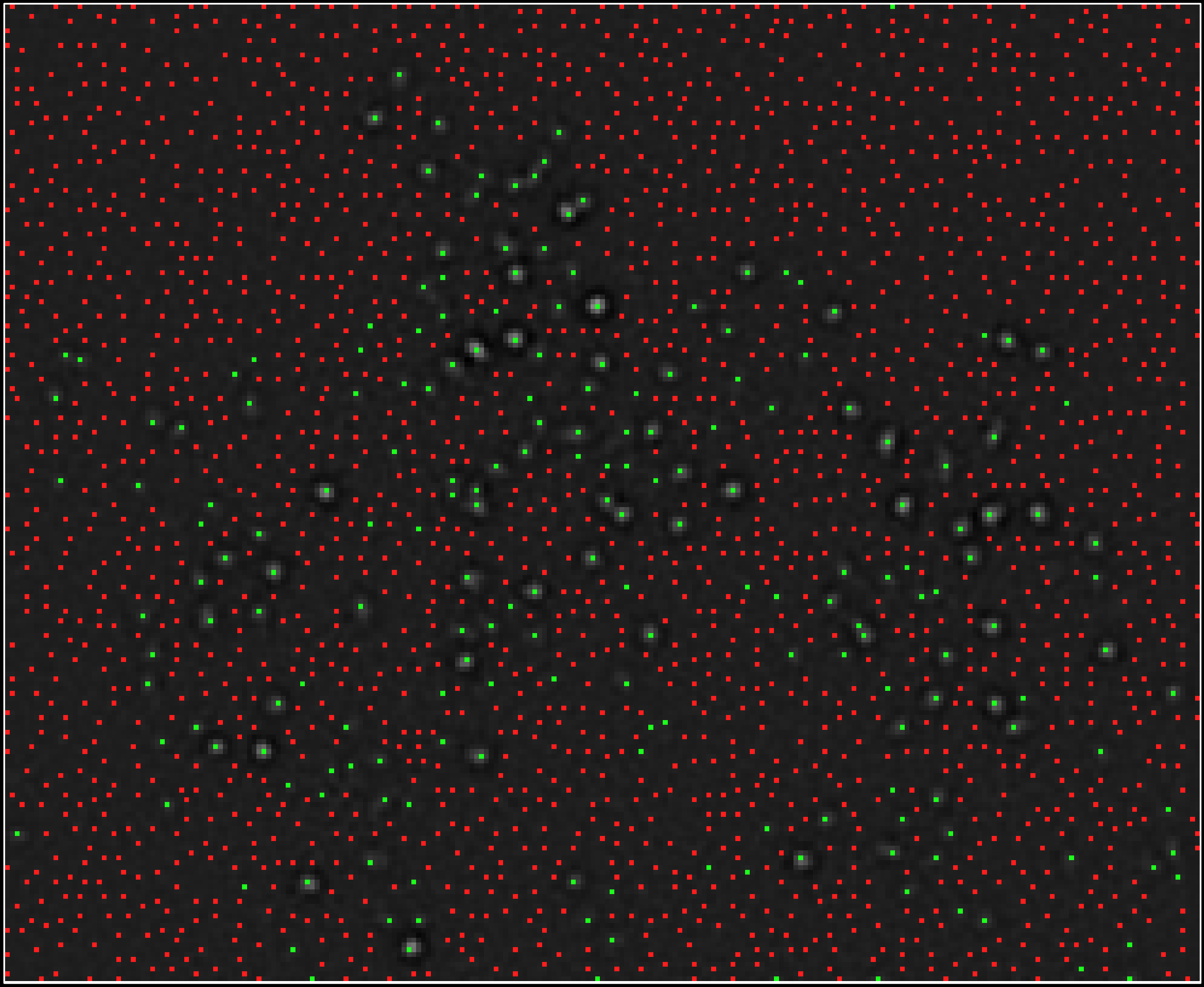
Separate Signal from Background

(Look for an “elbow” in the distribution of intensities of local maxima of filtered image)



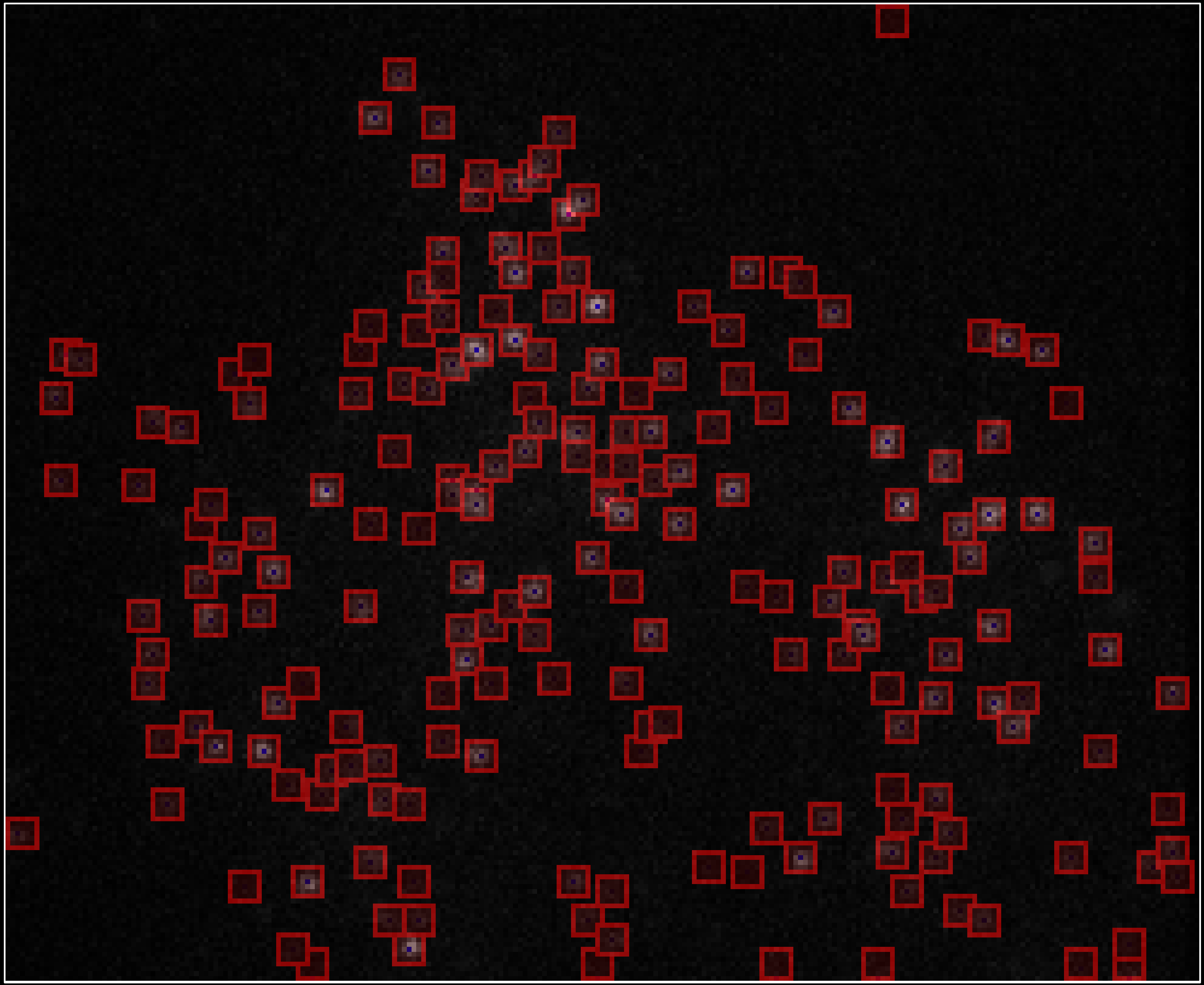
[Step 2] Identify Likely Single Emitters Sub-regions

Selected Maxima (Green=Signal Red=Noise)

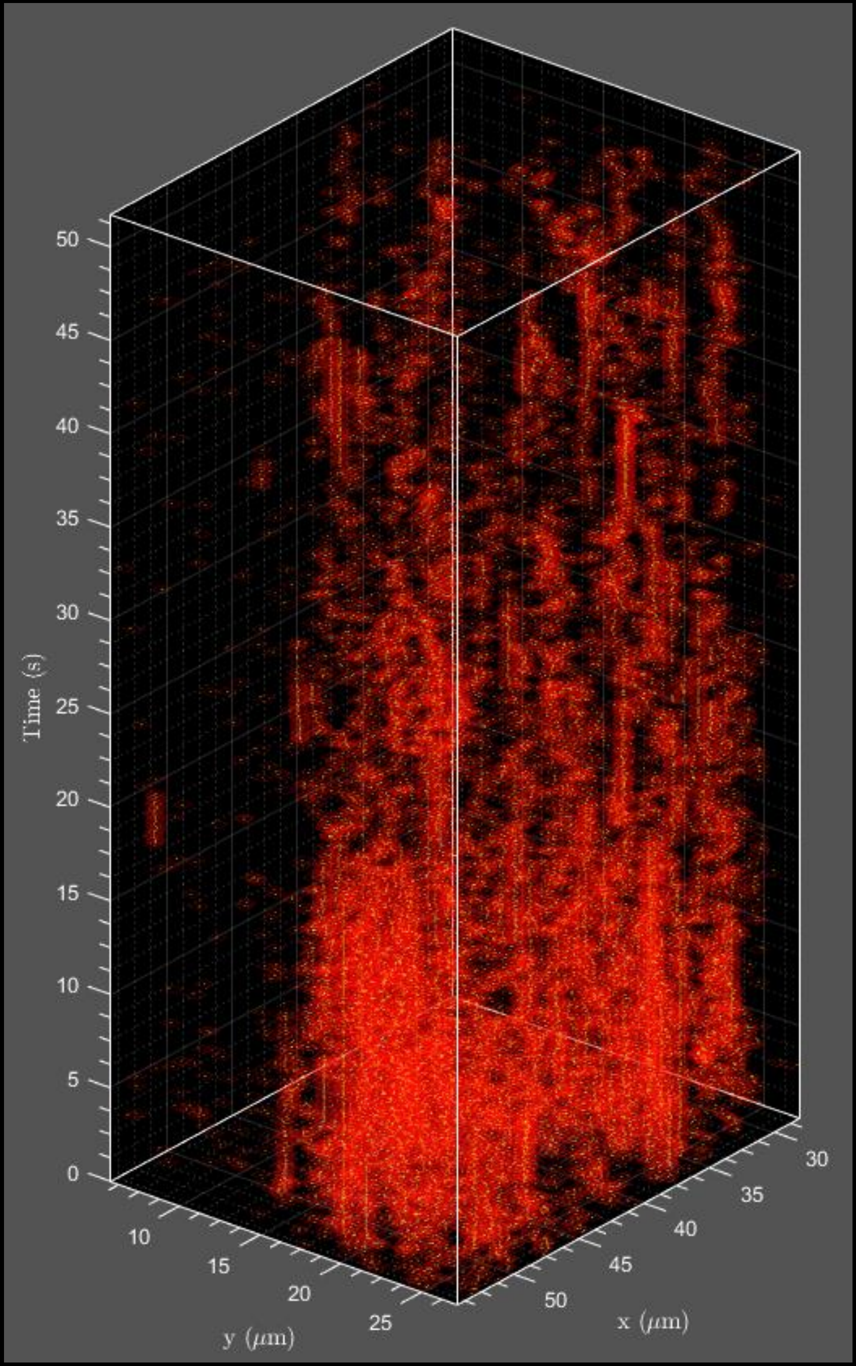


[Step 2] Identify Likely Single Emitters Sub-regions

Draw Boxes to Identify Fitting Regions



3D representation of all boxes over all frames for a 2D data set

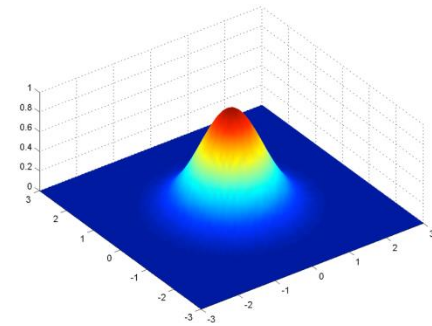


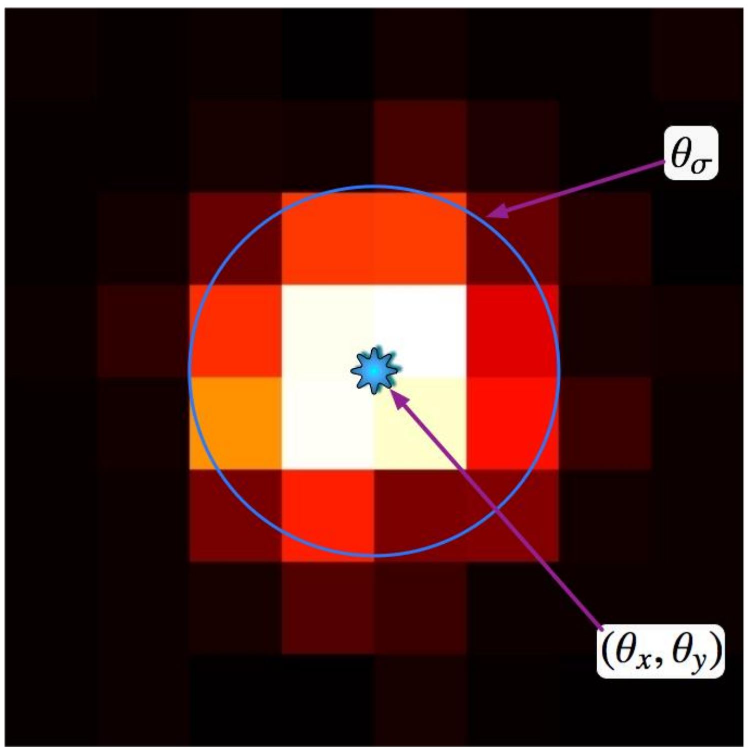
[Step 3] Point Emitter Localization

Model Parameters:

(s_x, s_y) ‣ box size

σ_{PSF} ‣ Point spread function width (pixels)



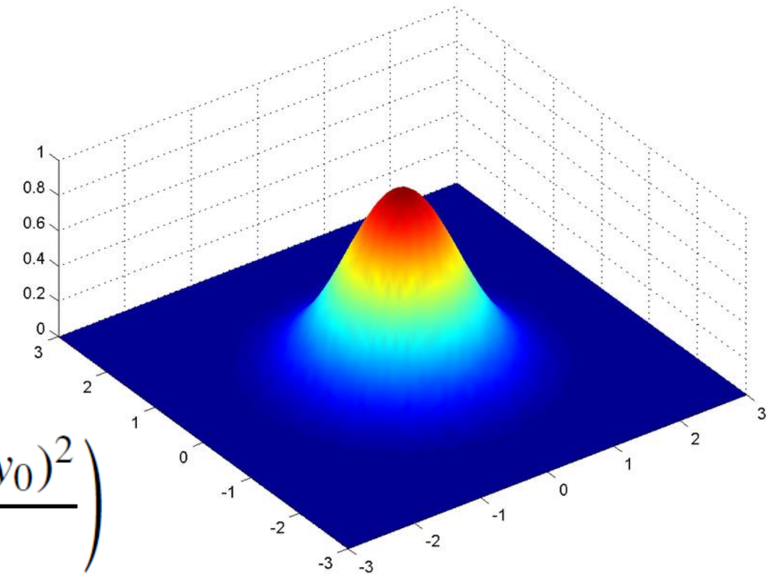


Goal: Given image, predict:

- θ_x ‣ x-position (pixels)
- θ_y ‣ y-position (pixels)
- θ_I ‣ Intensity (photons)
- θ_{bg} ‣ Background (photons/pixel)
- θ_σ ‣ Apparent Gaussian sigma

$$\boldsymbol{\theta}^{2D\sigma} := \boldsymbol{\theta} = (\theta_x, \theta_y, \theta_I, \theta_{bg}, \theta_\sigma)$$

2D Gaussian Point Spread Function



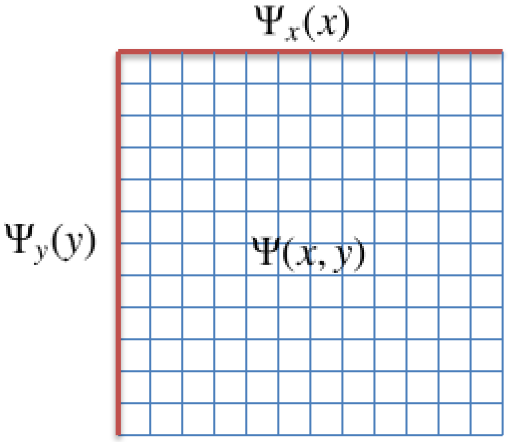
$$\Psi(x, y;\ x_0, y_0, \sigma) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}\right)$$

Can be separated into 2 1D Gaussian Point Spread Functions

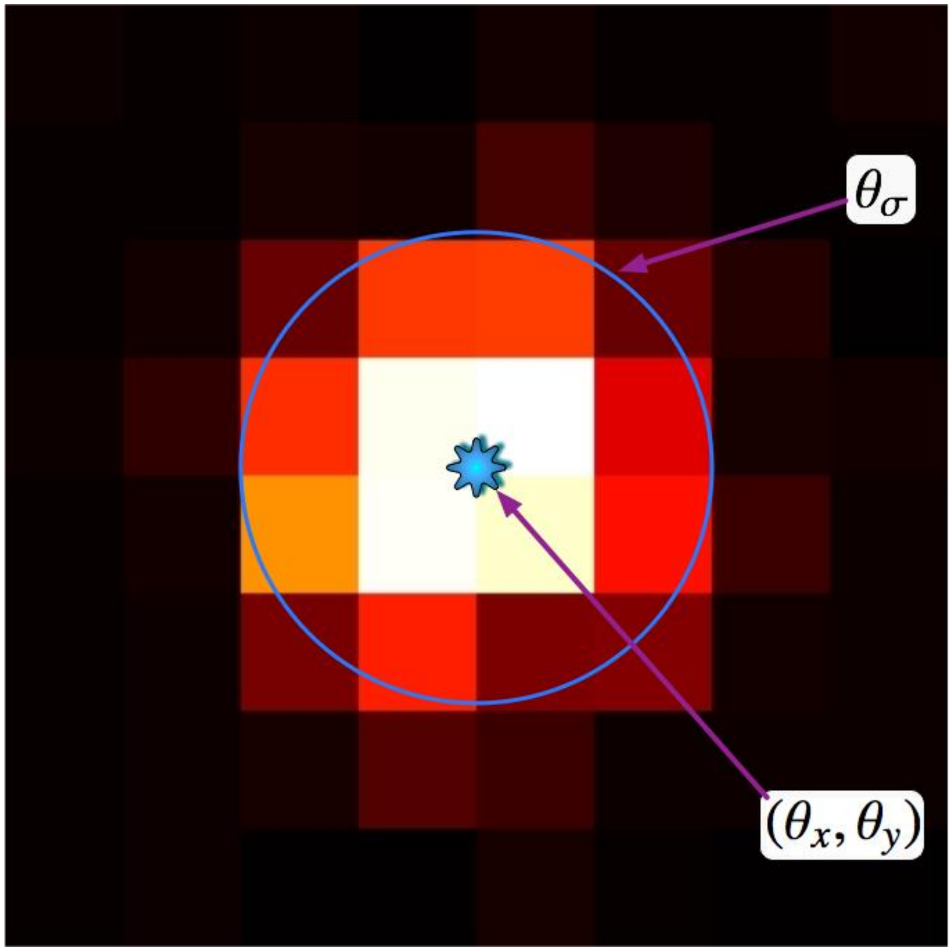
$$\Psi(x, y;\ x_0, y_0, \sigma) = \Psi_x(x;\ x_0, \sigma)\Psi_y(y;\ y_0, \sigma),$$

$$\Psi_x(x;\ x_0, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-x_0)^2}{2\sigma^2}\right)$$

$$\Psi_y(y;\ y_0, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(y-y_0)^2}{2\sigma^2}\right)$$



[Step 3] Point Emitter Localization



Data: pixel photon counts

$$\mathbf{c} = \{c_{ij}\}_{0 \le i, j \le s-1}$$

$$c_{ij} \sim \text{Poisson}(\mu_{ij}(\boldsymbol{\theta}))$$

Model: expected pixel photon counts

$$\mu_{ij}(\boldsymbol{\theta}) = \theta_{\text{bg}} + \theta_I X_i Y_j$$

$$X_i(\theta_x, \theta_\sigma) = \int_i^{i+1} \Psi_x(x;\ \theta_x, \theta_\sigma)\mathrm{d}x$$

$$Y_j(\theta_y, \theta_\sigma) = \int_j^{j+1} \Psi_y(y;\ \theta_y, \theta_\sigma)\mathrm{d}y$$

Maximum Likelihood Estimation

$$c_{ij} \sim \text{Poisson}\left(\mu_{ij}(\theta)\right) \qquad \mathcal{L}(c \mid \theta) = \prod_{i,j} \Pr\left[c_{ij} \mid \mu_{ij}(\theta)\right] = \prod_{i,j} \frac{\mu_{ij}(\theta)^{c_{ij}} e^{-\mu_{ij}(\theta)}}{c_{ij}!}$$

$$\theta_{\text{MLE}} = \operatorname*{argmax}_{\theta} \mathcal{L}(c \mid \theta)$$

$$\begin{aligned}
\Lambda(\theta) = \ln \mathcal{L}(c \mid \theta) &= \ln\left(\prod_{i,j} \frac{\mu_{ij}(\theta)^{c_{ij}} e^{-\mu_{ij}(\theta)}}{c_{ij}!}\right) \\
&= \sum_{i,j} \ln\left(\mu_{ij}(\theta)^{c_{ij}}\right) + \ln\left(e^{-\mu_{ij}(\theta)}\right) - \ln\left(c_{ij}!\right) \\
&= \sum_{i,j} c_{ij} \ln \mu_{ij}(\theta) - \mu_{ij}(\theta) - \ln\left(c_{ij}!\right) \\
&\approx \sum_{i,j} c_{ij} \ln \mu_{ij}(\theta) - \mu_{ij}(\theta) - c_{ij} \ln c_{ij} + c_{ij} - \frac{\ln\left(2\pi c_{ij}\right)}{2}
\end{aligned}$$

$$\Lambda^{\star}(\theta) = \sum_{i,j} c_{ij} \ln \mu_{ij}(\theta) - \mu_{ij}(\theta)$$

$$\theta_{\text{MLE}} = \operatorname*{argmax}_{\theta} \Lambda(\theta) = \operatorname*{argmax}_{\theta} \Lambda^{\star}(\theta)$$

Parameter Estimation using Probability Models

Bayes Estimate (Posterior mean)

$$\hat{\theta}_{\mathrm{BAYES}} = \hat{\theta}_{\mathrm{mean}} = \mathrm{E}[\boldsymbol{\theta}] = \int \boldsymbol{\theta}\, p(\boldsymbol{\theta} \mid \boldsymbol{C})\, \mathrm{d}\boldsymbol{\theta}$$

MAP Estimate

$$\hat{\theta}_{\mathrm{MAP}} = \hat{\theta}_{\mathrm{mode}} = \underset{\theta}{\mathrm{argmax}}\; p(\boldsymbol{\theta} \mid \boldsymbol{C})$$

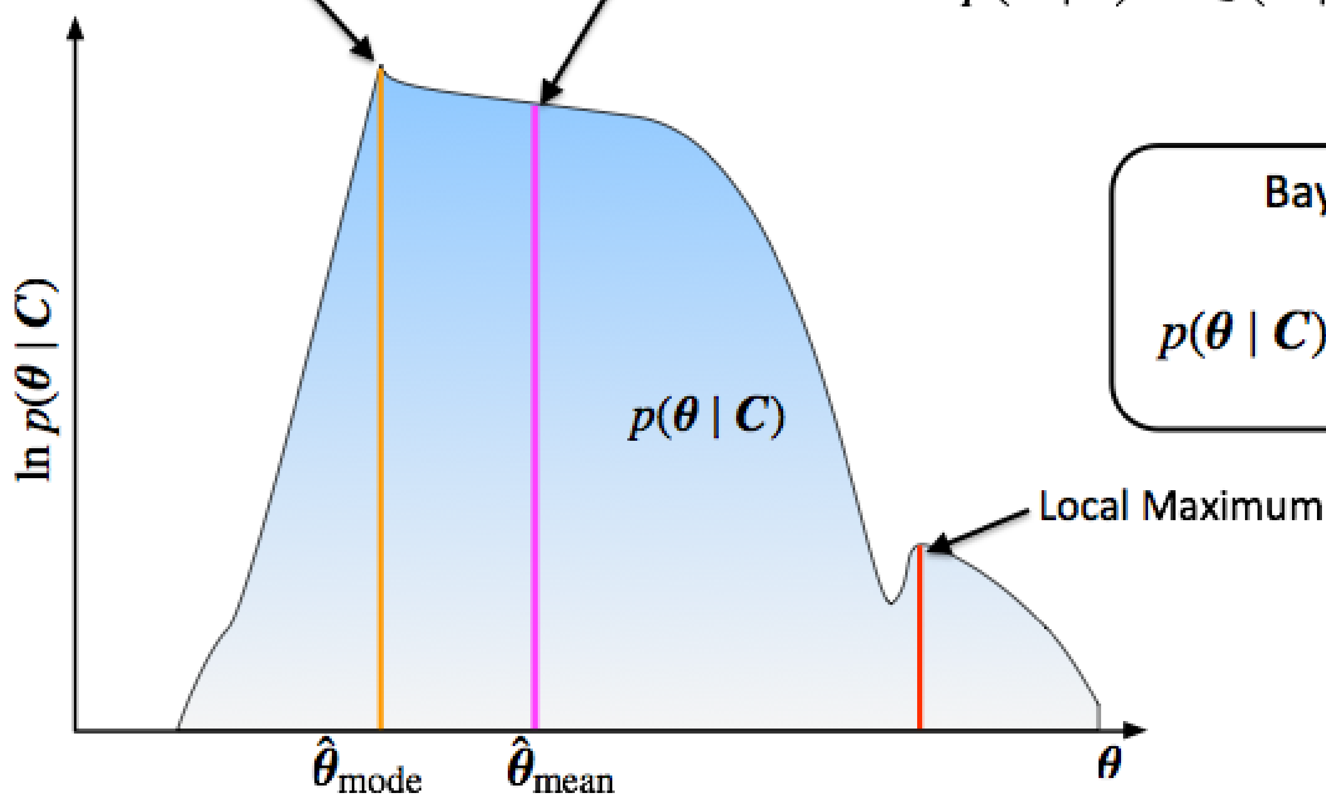
$\boldsymbol{\theta}$ - Parameters

$\boldsymbol{C}$ - Data

$p(\boldsymbol{\theta})$ - Prior

$p(\boldsymbol{\theta} \mid \boldsymbol{C})$ - Posterior

$p(\boldsymbol{C} \mid \boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{C} \mid \boldsymbol{\theta})$ - Likelihood

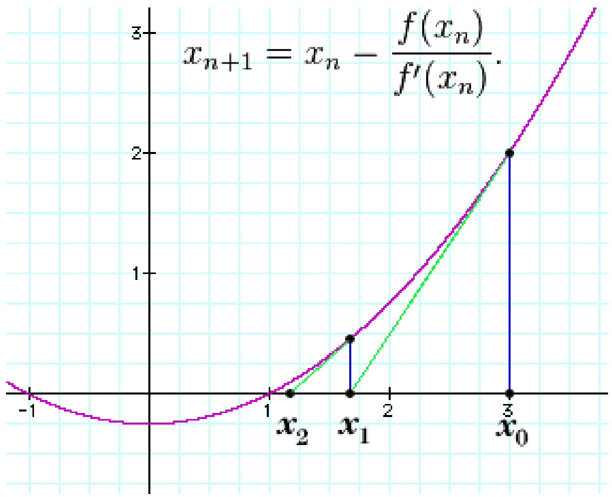


Bayes' Theorem

$$p(\boldsymbol{\theta} \mid \boldsymbol{C}) = \frac{p(\boldsymbol{C} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\boldsymbol{C})}$$

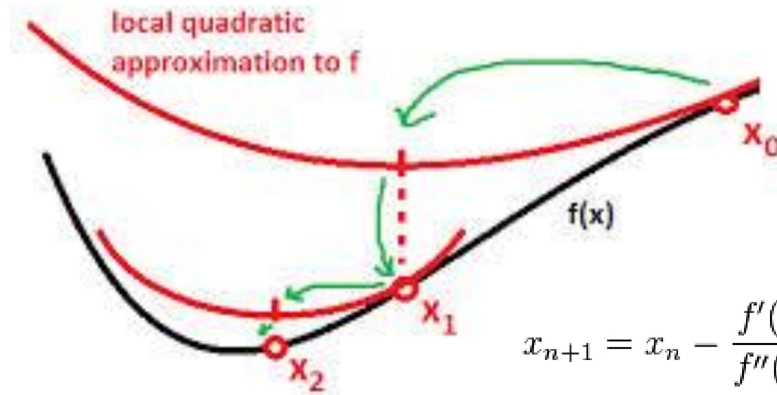
Newton's Method For Optimization (MAP Estimation)

1D Root-finding



$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

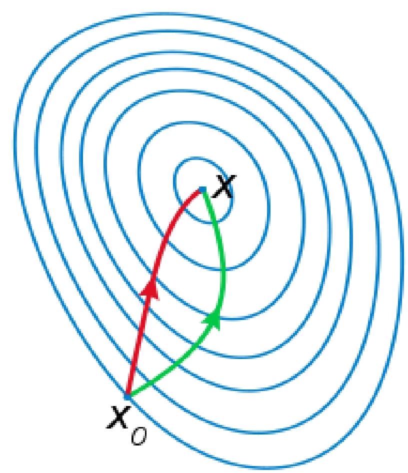
1D Optimization via Root finding for f′(x)



$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}.$$

nD Optimization requires computing the Hessian (2nd Derivative)

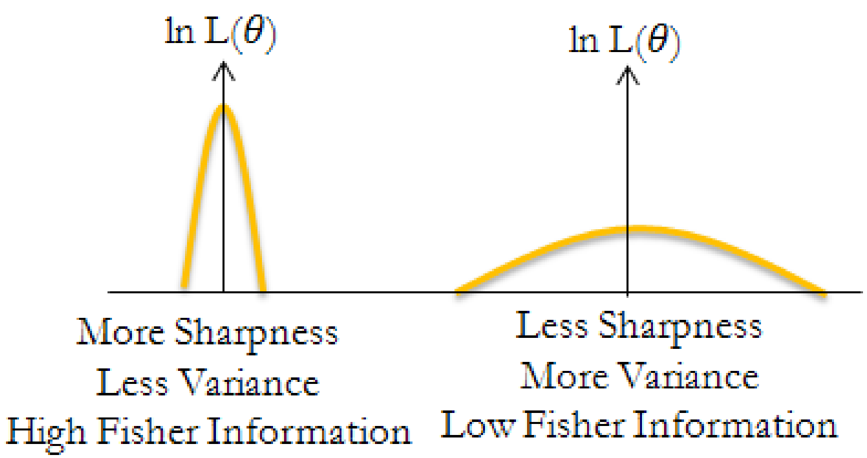
$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

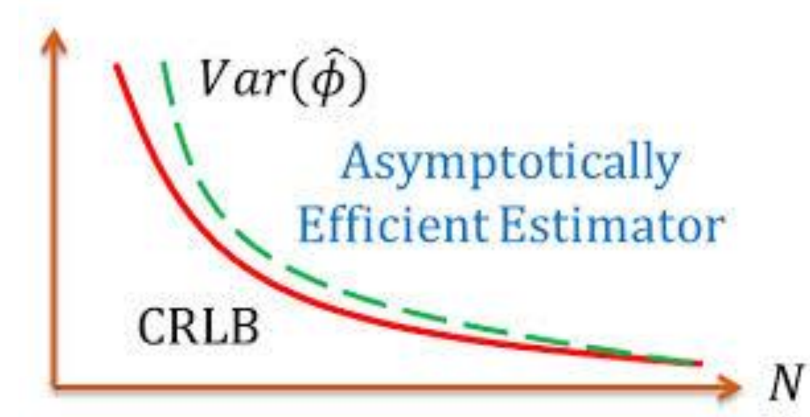


$$\mathbf{x}_{n+1} = \mathbf{x}_n - [Hf(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n),\ n \geq 0.$$

Cramer-Rao Lower Bound and Fisher Information

$$\text{Curvature} = -\frac{\partial^2}{\partial \theta^2}[\ln L(\theta)]$$





Fisher Information: Measures **average** curvature. A function of only the parameter value and not the data as we average over all possible data X.

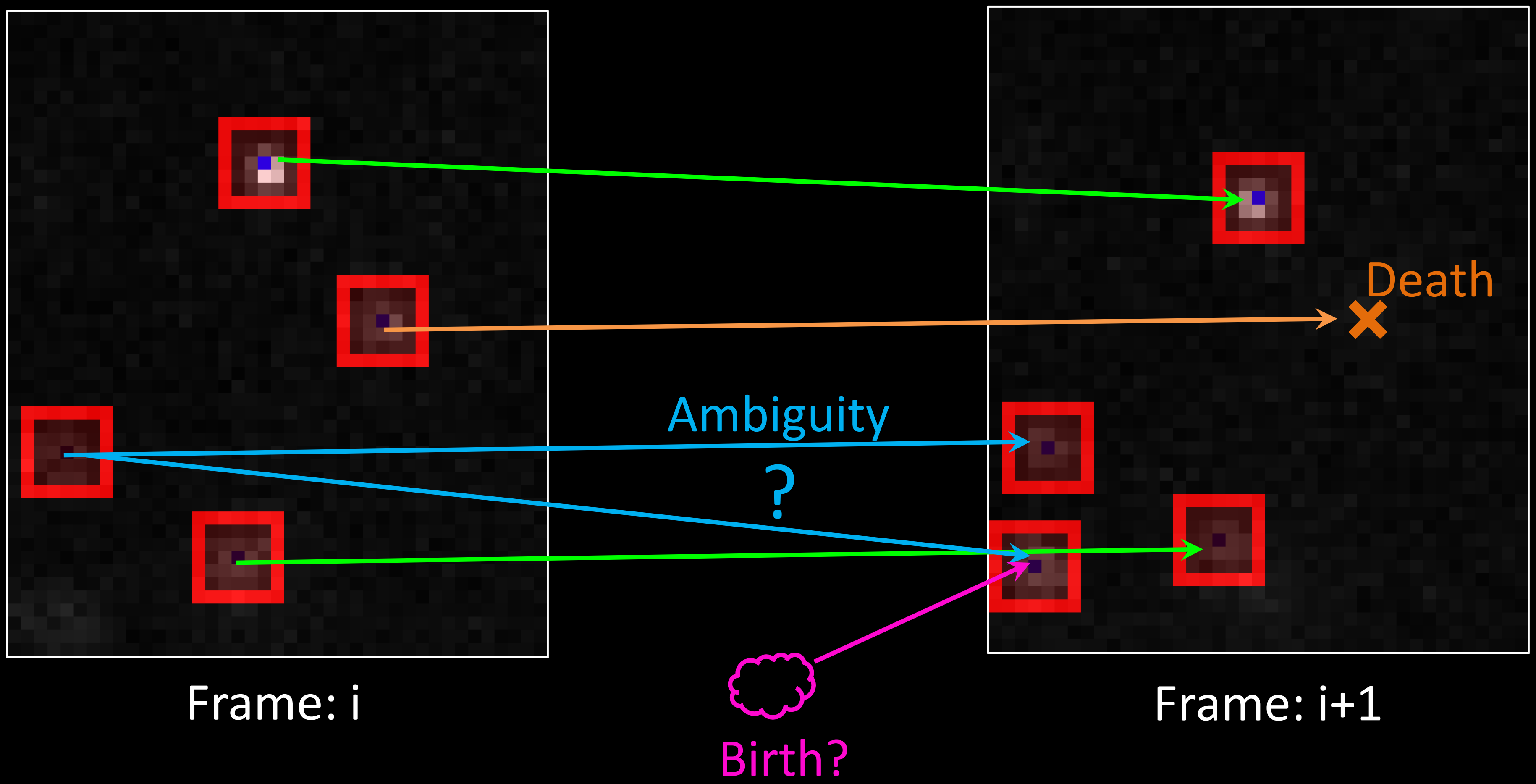
$$\mathcal{I}(\theta) = -\operatorname{E}\left[\left.\frac{\partial^2}{\partial\theta^2}\log f(X;\theta)\right|\theta\right],$$

Cramer-Rao Lower Bound

$$\operatorname{Var}(\hat{\theta}) \geq \frac{1}{\mathcal{I}(\theta)}$$

[Step 4] Tracking: Connecting Localizations into trajectories

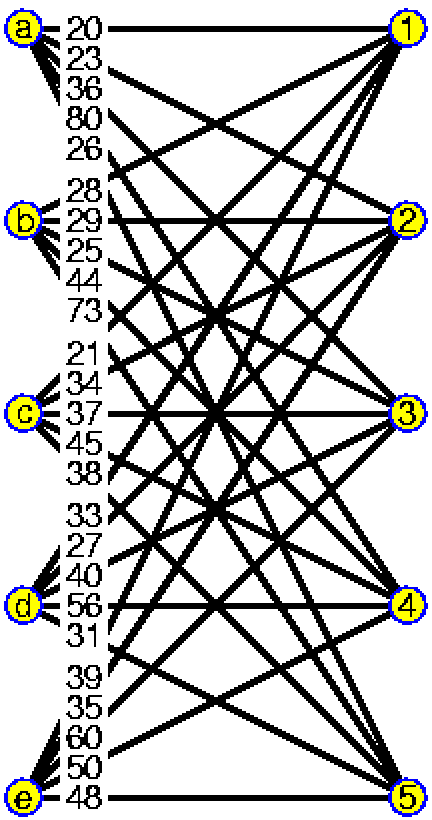
Tracking can be formulated as a combinatorial optimization problem
First Step is Frame-to-Frame connection



Linear Assignment Problem

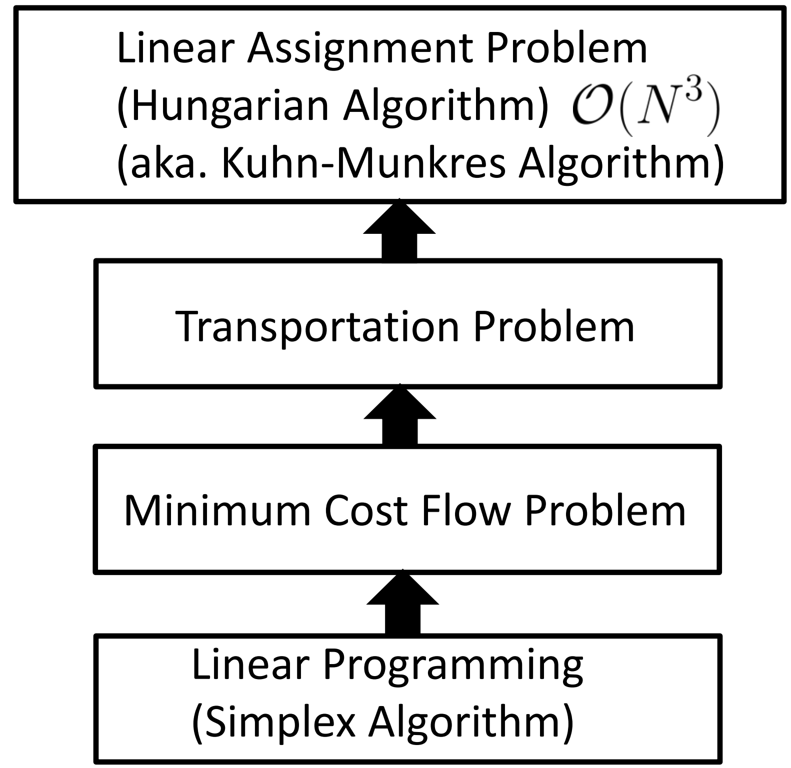
WEIGHTED BIPARTITE MATCHING

Bipartite Graph



Cost Matrix

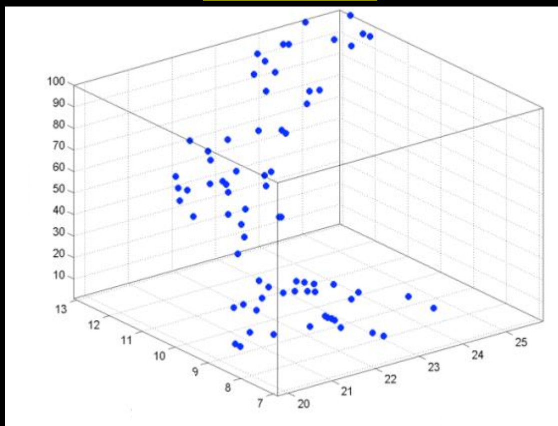
	1	2	3	4	5
a	20	23	36	80	26
b	28	29	25	44	73
c	21	34	37	45	38
d	33	27	40	56	31
e	39	35	60	50	48



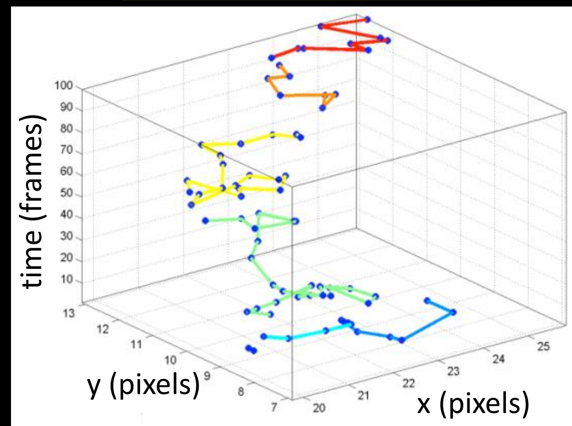
Related: Stable Marriage Problem $\mathcal{O}(N^2)$

[Step 4] Tracking: Connecting Localizations into trajectories

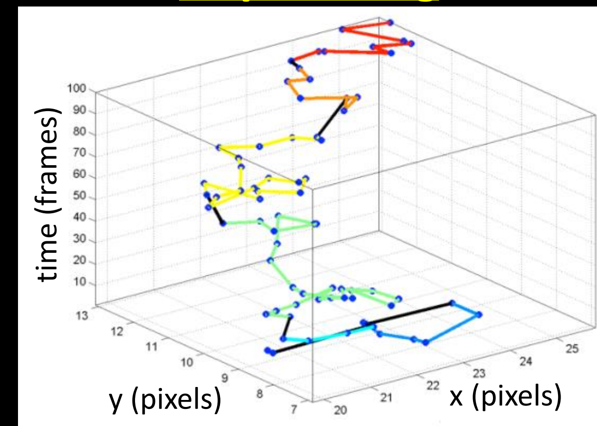
Localize

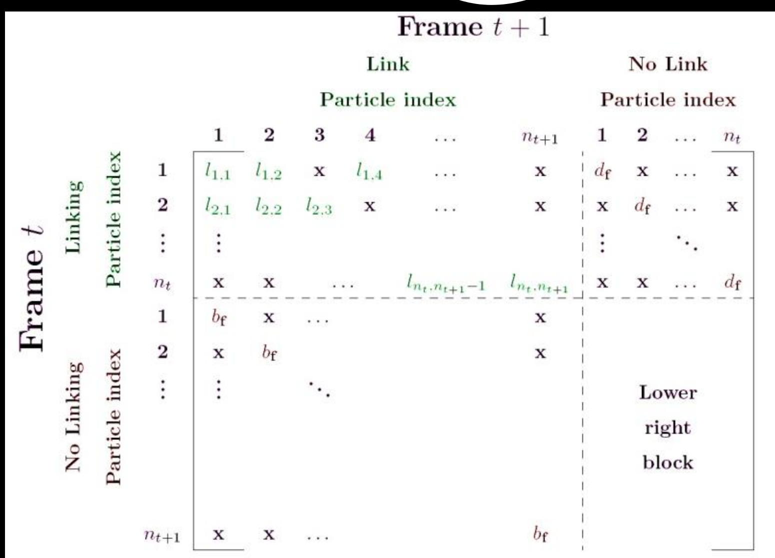


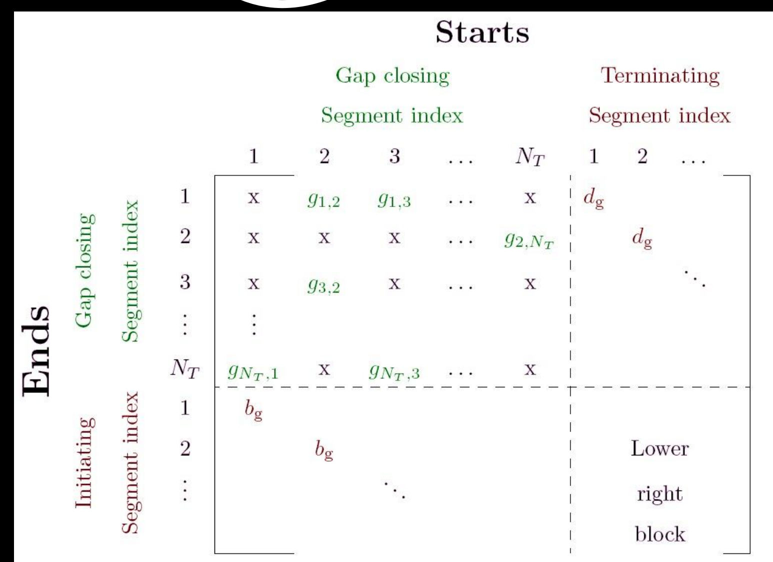
Frame-to-Frame

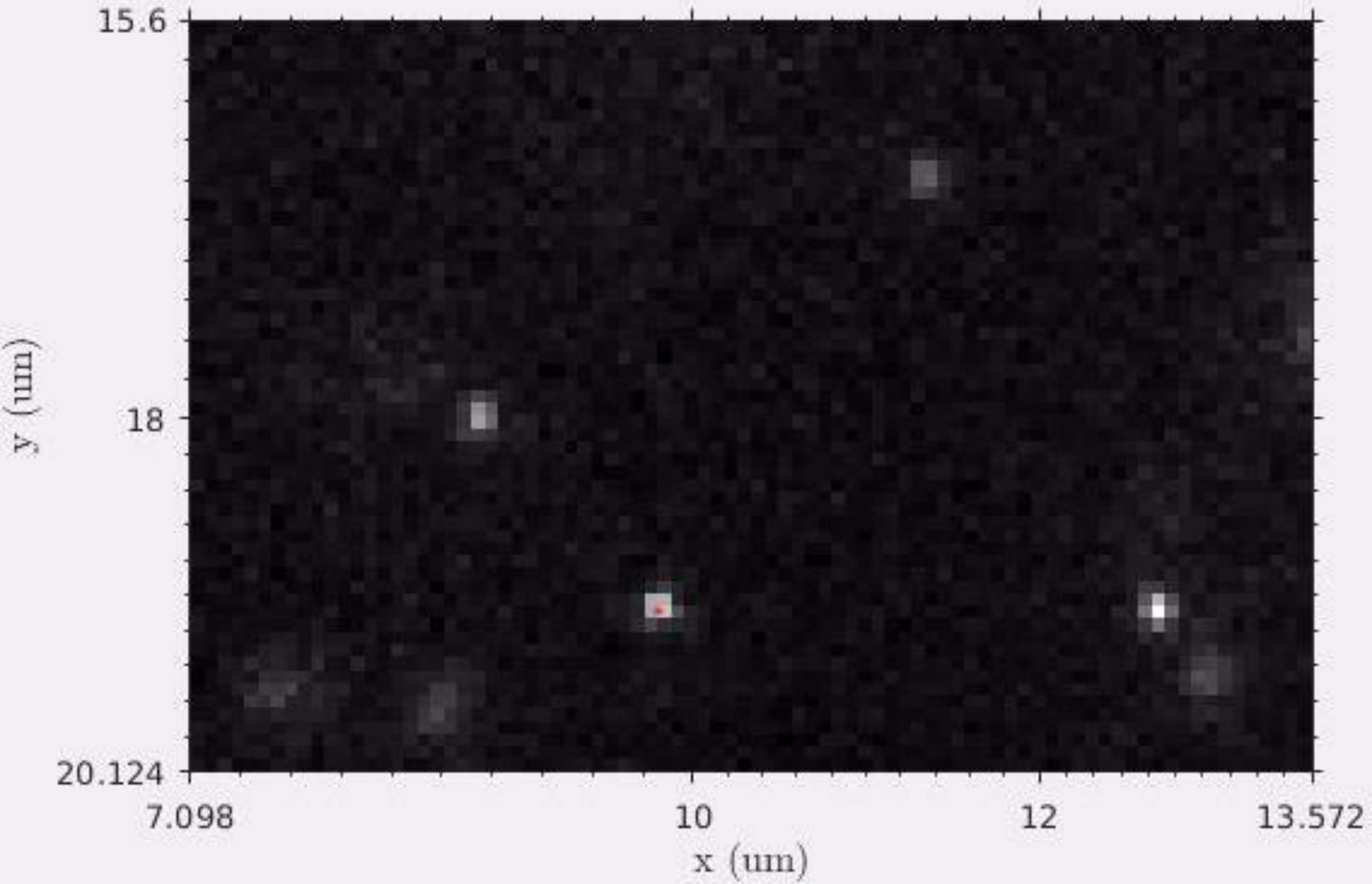


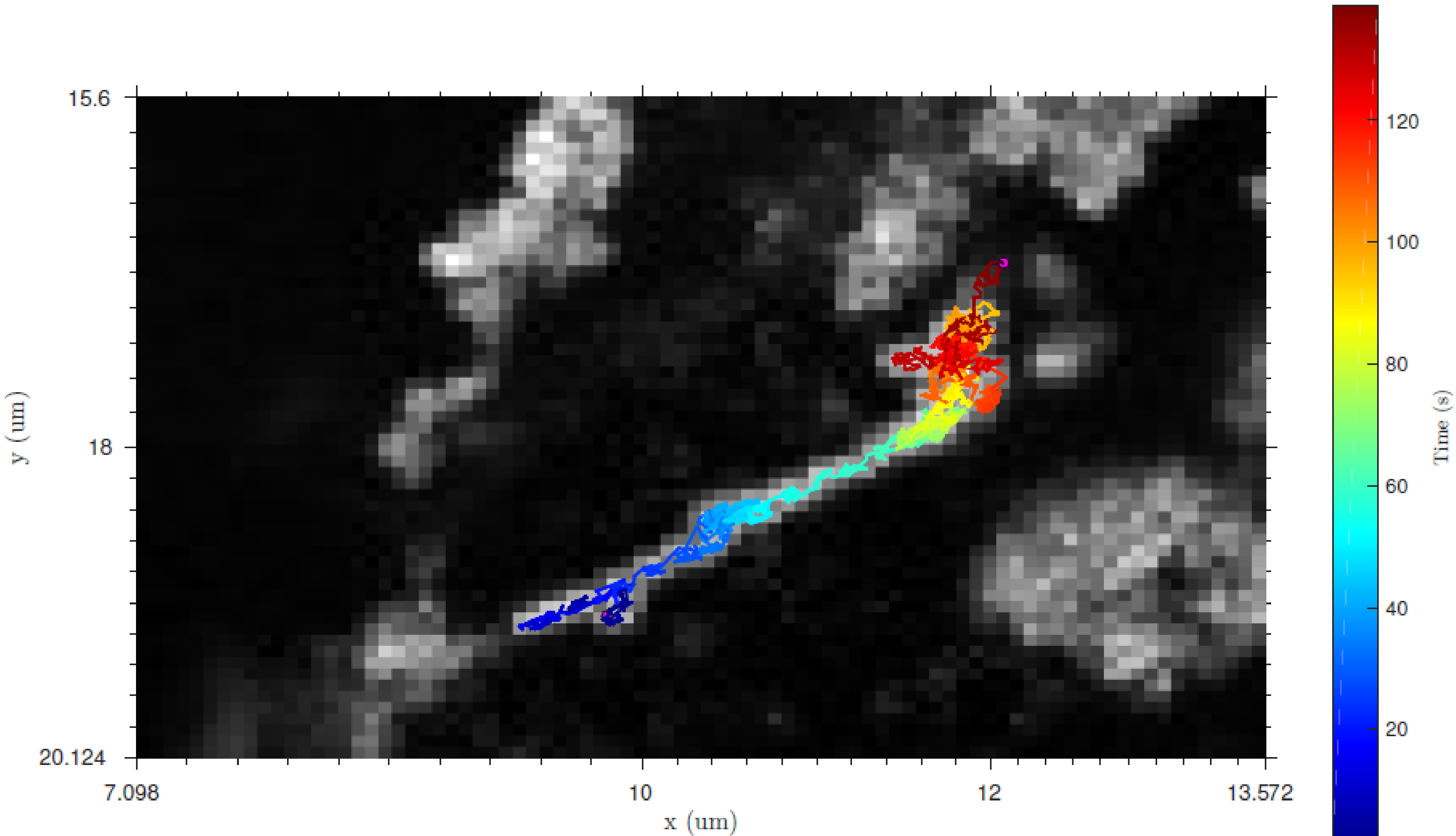
Gap Closing








15.6
18
20.124
y (um)
7.098
10
12
13.572
x (um)



15.6
18
20.124
y (um)
7.098
10
12
13.572
x (um)
120
100
80
60
40
20
Time (s)

Point Emitter Localization with MAPPEL

MAPPEL: Maximum A-Posteriori Point Emitter Localization

Mark J. Olah (mjo@cs.unm.edu)
Department of Physics and Astronomy, University of New Mexico

- MAPPEL is an object-oriented Matlab/C++ package for point emitter localization in 2D and 3D using a Gaussian PSF model and a Poisson noise model.
- MAPPEL can be used directly from C++ or through a Matlab interface which presents an object-oriented class structure where each class works with a different model variant.
- Designed to be extremely fast, numerically accurate, and robust for use with low S/N data.
- Cross-platform [Win64, Linux, OSX (soon)].
- Uses OpenMP to parallelize large computations. Does not currently rely on CUDA, so should run on any x86-64 machine without special hardware.
- Uses Cmake for cross-platform building, and Mingw64 for windows cross-compiling from Linux.
- Provides multiple maximization methods for MAP modes
  - Newton, Newton-Raphson, Quasi-Newton, etc,…
- Provides full posterior estimation with MCMC Metropolis-Hastings algorithm.
- Each MAPPEL object has methods that allow for the simulation of images computation of LLH and CRLB as well as emitter localization with MAP or Posterior methods.
- Requirements: DIPImage (optional but useful). Tested on Matlab 2013b+. Note that DIPImage 2.7 is required for 2014b+ support.

MAPPEL Model Classes

Basic Model Classes (probably you want one of these)

- Gauss2DMLE - 2D Model. Maximum Likelihood Estimation: theta = [x y I bg]
- Gauss2DsMLE - 2D Model. Maximum Likelihood Estimation: theta = [x y I bg sigma]
- Gauss2DMAP - 2D Model. Maximum a-posteriori Estimation: theta = [x y I bg]
- Gauss2DsMAP - 2D Model. Maximum a-posteriori Estimation: theta = [x y I bg sigma]

Other Model Classes

- Blink2DsMAP - 2D Model. MAP Estimation for Line-scanning microscopy.
- GaussHSMAP - Hyperspectral Model. MAP Estimation: theta = [x y L I bg]
- GaussHSsMAP - Hyperspectral Model. MAP Estimation: theta = [x y L I bg sigma sigmaL]
- BlinkHSsMAP - Hyperspectral Line-scanning Model. theta = [x y L I bg sigma sigmaL]

Useage:

```
g2d = Gauss2DMAP( imsize, psfsigma)
[or]
g2d = Gauss2DsMAP( imsize, psfsigma)
```

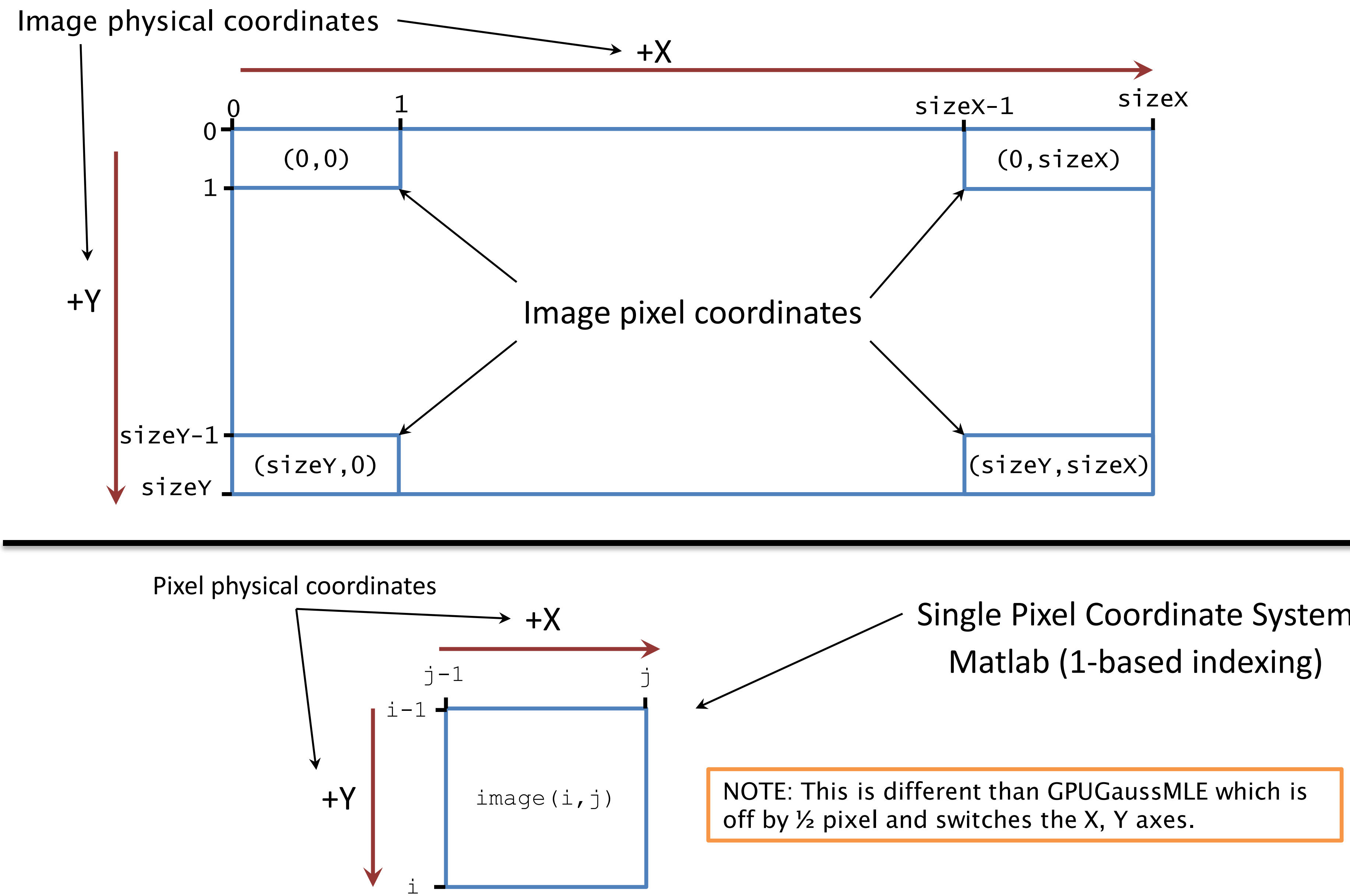
[ARGS]

imsize: [X Y] in pixels. See slides on coordinate systems

psfsigma: [X Y] in pixels. (or scalar value for symmetric PSF)

NOTE: for the ‘s’ models that also estimate the apparent psf sigma as an extra theta parameter, the sigma value returned in parameter theta is a unit-less scaling that corresponds to this input psfsigma constant. (i.e. the estimated sigma returned will be 1 if the emitter is exactly in focus and >1 if out-of-focus)

IMPORTANT: Image Sizes and Coordinate System Reference



NOTE: This is different than GPUGaussMLE which is off by ½ pixel and switches the X, Y axes.

Example: Make a Gauss2DMAP model object that works on images with

- sizeX=7; sizeY=15
- psfSigmaX=1; psfSigmaY=2

```
>> g2d = Gauss2DMAP([7 15], [1,2])
g2d =
  Gauss2DMAP with properties:

                Name: 'Gauss2DMAP'
             nParams: 4
          ParamNames: {'x'  'y'  'I'  'bg'}
          ParamUnits: {'pixels'  'pixels'  '#'  '#'}
    ParamDescription: {'x-position'  'y-position'  'Intensity'  'background'}
        nHyperParams: 5
     HyperParamNames: {'Beta_pos'  'Mean_I'  'Kappa_I'  'Mean_bg'  'Kappa_bg'}
             MinSize: 4
   EstimationMethods: {1x7 cell}
              imsize: [7 15]
           psf_sigma: [1 2]
```

```
Note: imsize and psf_sigma always use the [X Y]
format.  Only the images appear in the opposite
order.
```

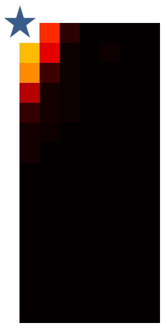
```
theta = [x-pos, y-pos, Intensity, bg]
```

x-pos (pixels), y-pos (pixels), Intensity (photons), bg (photons/pixel)

g2d.imsize = [7, 15] % [sizeX, sizeY]

Ex: Emitter at true coordinates X=0 Y=0

```
>> g2d.simulateDipImage([0, 0, 1000, 0])
```

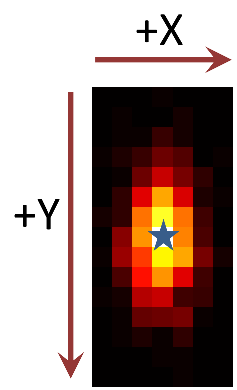


Ex: Emitter at true coordinates X=7 Y=0

```
>> g2d.simulateDipImage([7, 0, 1000, 0])
```

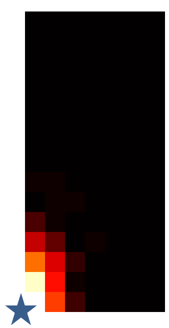
Ex: Emitter at true coordinates X=3.5 Y=7.5

```
>> g2d.simulateDipImage([3.5, 7.5, 1000, 0])
```



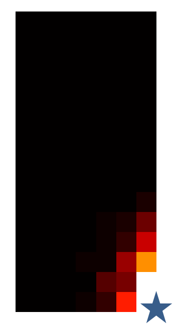
Ex: Emitter at true coordinates X=0 Y=15

```
>> g2d.simulateDipImage([0, 15, 1000, 0])
```



Ex: Emitter at true coordinates X=7 Y=15

```
>> g2d.simulateDipImage([7, 15, 1000, 0])
```



Example: Working with MAPPEL objects

- Create a new model

```
>> imsize=[8 8]; psfsigma=[1 1];
>> g2d=Gauss2DsMAP(imsize, psfsigma)
g2d =
  Gauss2DsMAP with properties:

              nParams: 5
         nHyperParams: 6
                 Name: 'Gauss2DsMAP'
           ParamNames: {'x'  'y'  'I'  'bg'  'sigma'}
      HyperParamNames: {'Beta_pos'  'Mean_I'  'Kappa_I'  'Mean_bg'  'Kappa_bg'  'alpha_sigma'}
           ParamUnits: {'pixels'  'pixels'  '#'  '#'}
     ParamDescription: {1x5 cell}
              MinSize: 4
    EstimationMethods: {1x7 cell}
               imsize: [8 8]
            psf_sigma: [1 1]
```

- Sample 1000 thetas from the prior distribution

```
>> thetas=g2d.samplePrior(1000);
```

- Generate the model images for each theta, which give the expected photon count.

```
>> model_ims = g2d.modelImage(thetas)
```

or

```
>> model_dip_ims = g2d.modelDipImage(thetas)
```

- Simulate images for each theta, which sample from the Poisson distribution with mean given by the model images

```
>> sim_ims = g2d.simulateImage(thetas)
```

or

```
>> sim_dip_ims = g2d.simulateDipImage(thetas)
```

Example: Working with MAPPEL objects

- Get the CRLB at each of the thetas

```
>> crlb = g2d.CRLB(thetas);
```

- Get the LLH of each a stack of images at their respective theta value

```
>> llh = g2d.LLH(ims, theta);
```

- Estimate the theta value for a stack of images using MAP with Newton's method as default maximizer

```
>> [etheta, crlb, llh, stats] = g2d.estimateMAP(ims);
```

- Estimate the theta value for a stack of images using Posterior MCMC sampling

```
>> [etheta, covariance] = g2d.estimatePosterior(ims);
```

Plus lots of other methods for more detailed work with the model space and information measures. See MappelBase.m for more documentation on other methods.

Estimator Methods

- MAPPEL provides the ability to do ordinary Maximum Likelihood estimation, or the Maximum a-posteriori estimation which is very similar operationally, but includes the influence of a prior distribution.
  - The Gauss2DMLE and Gauss2DsMLE classes will do ordinary maximum likelihood
  - The Gauss2DMAP and Gauss2DsMAP will do the Maximum a-posteriori estimation
  - All of the classes use the method `estimateMAP` to do the estimation, but the MLE classes will ignore the prior settings, despite the MAP in the method name.
- The estimateMAP method takes as an optional second argument a string giving a method for doing the estimation. The most important methods are:
  - "Newton" – Uses the full Hessian (this is the default)
  - "NewtonRaphson" – A newton's method that uses only the diagonal of the hessian matrix to compute the curvature of the objective function. This is also a good choice. It is slightly faster and sometimes gives better results for the "s" models that estimate sigma
  - "CGauss" – Uses the code from *Smith et. al. Nature Methods. 2010.* The output is converted to the MAPPEL coordinate system. This code has limitations and is slower than the other methods. This is included for comparison.

Examples:

```
>> [etheta, crlb, llh, stats] = g2d.estimateMAP(sim_ims,'Newton');
>> [etheta, crlb, llh, stats] = g2d.estimateMAP(sim_ims,'NewtonRaphson');
>> [etheta, crlb, llh, stats] = g2d.estimateMAP(sim_ims,'CGauss');
```

Converting from existing code that uses CGaussMLE

- MAPPEL can be viewed as a reimplementation of the mathematical description from *Smith et. al. Nature Methods. 2010.*, as well as an extension to other estimation methods and models.
- For users of the CGaussMLE or GPUGaussMLE codes from the Smith et.al. paper, a few points should be noted:
  - In CGaussMLE the different models are expressed as different “fit-types”. There are modes to fit where the effective sigma is not fit (fit-type 1) and where it is fit (fit-type 2). In MAPPEL the different fit types are different classes. The Gauss2DMAP and Gauss2DMLE classes do the fit-type=1 fitting where theta=[x y I bg]. The Gauss2DsMAP and Gauss2DsMLE do the fit-type=2 fitting where the apparent Gaussian sigma is estimated and theta=[x y I bg sigma]. Note that for MAPPEL, the sigma returned is a multiple of the PSF sigma, so when sigma=1 the emitter image Gaussian fit matches the PSF and the emitter is in focus. When sigma>1 the emitter image Gaussian is bigger than the PSF.
  - The image coordinate systems are different. See the slides on coordinate systems for details. The net effect is that X an Y are interchanged and the origin is shifted by (0.5,0.5) pixels.
  - The CRLB estimates will be potentially different because we use a more numerically stable matrix inversion algorithm.
  - The LLH values will be slightly different because MAPPEL includes constant correction terms to sterling’s approximation that will more accurately approximate the mathematically true log-likelihood values.
- NOTE: In MAPPEL-1.1 GPUGaussMLE is a built-in estimation method. Use estimator name ‘GPUGauss’. This method of calling through obj.estimateMAP automatically corrects for the image coordinate systems, and the results are consistent with the other estimators.