Imaging
June 17, 2008

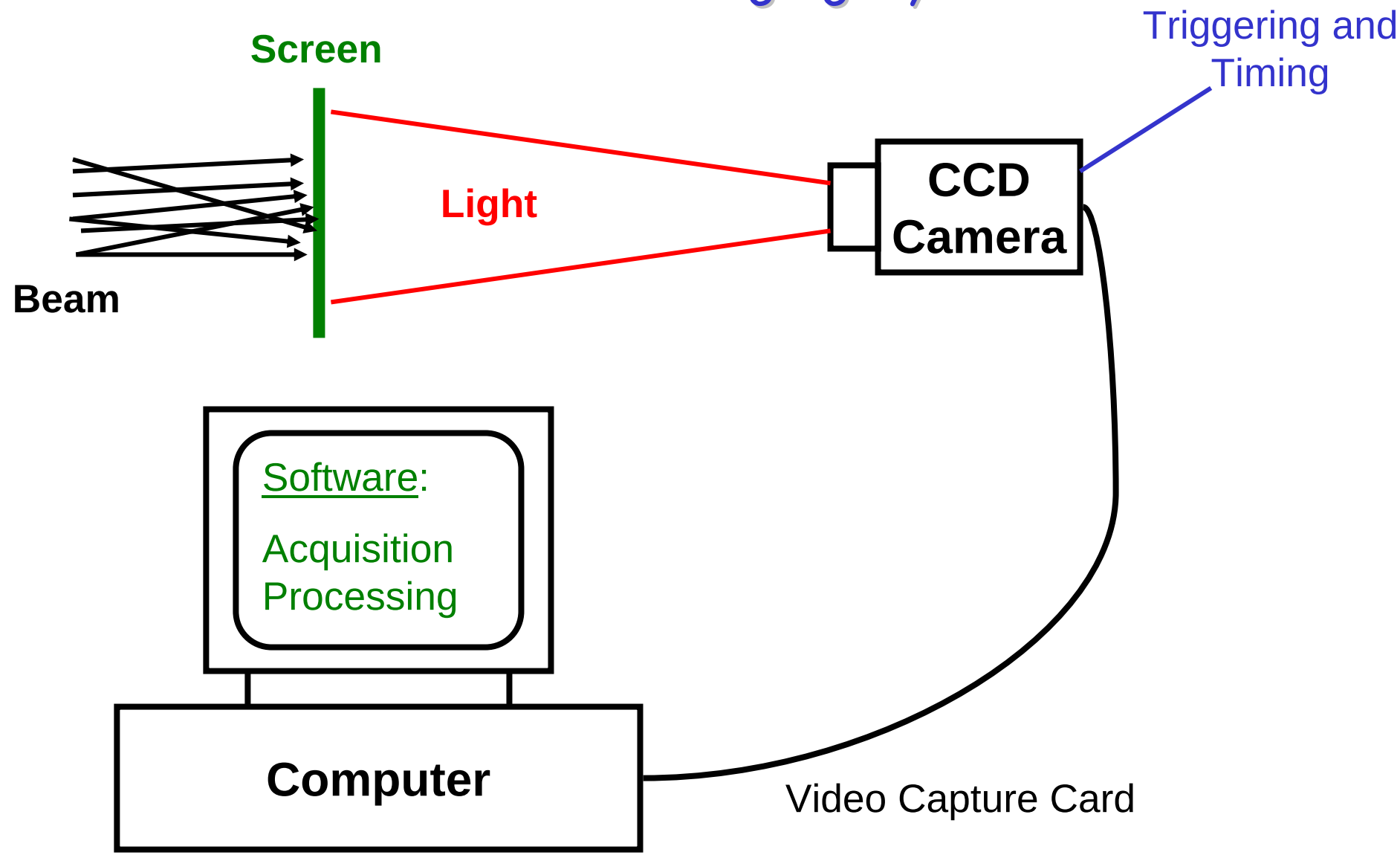
USPAS - Summer 2008

Rami Kishek

Motivation

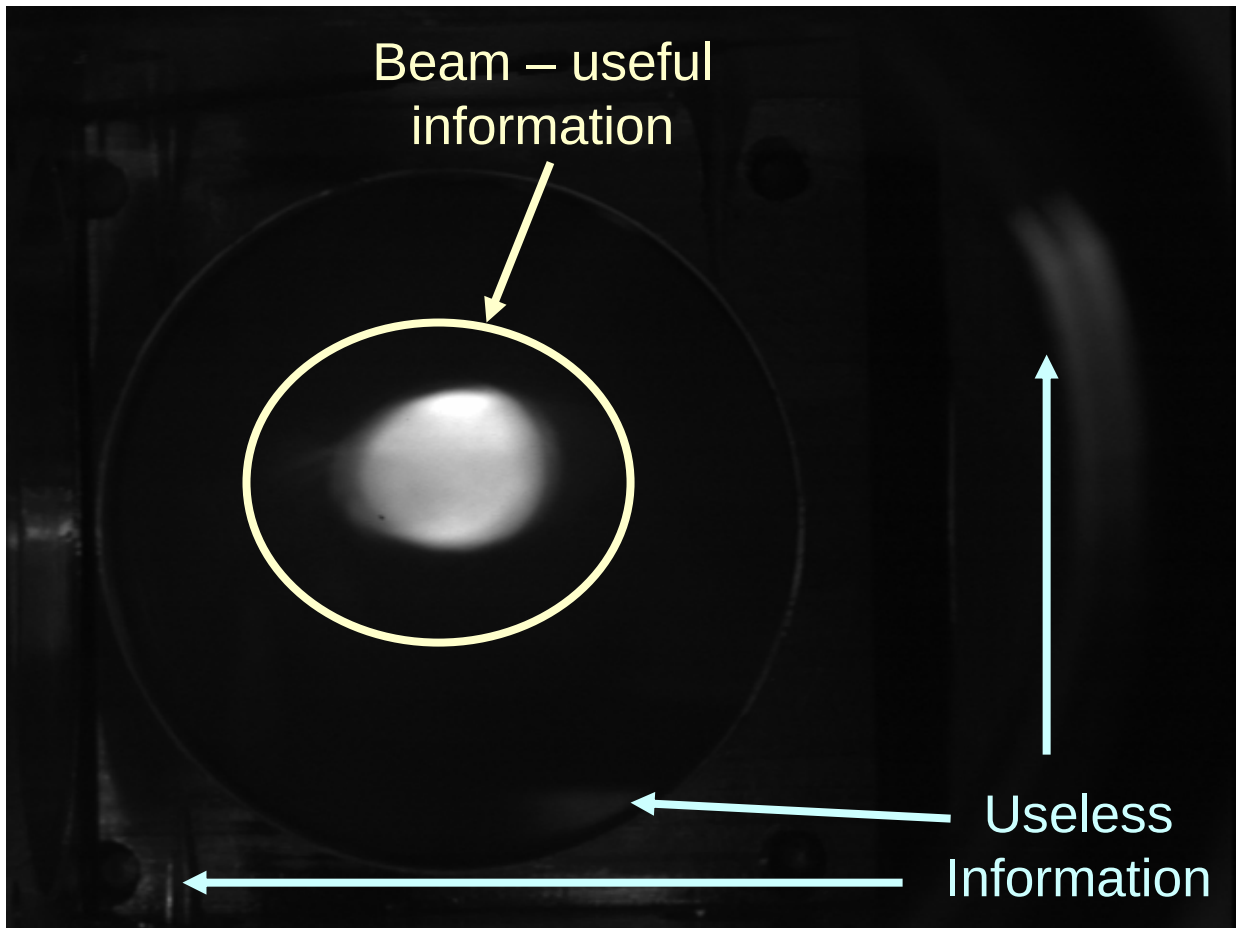
- Beam Density Distribution and Profile
- Obtain Moment Information:
  - Beam centroid position $\Rightarrow$ Steering
  - Beam radius $\Rightarrow$ Matching
  - Beam rotation angle $\Rightarrow$ Skew Correction
- Phase-space mapping devices:
  - Quad-scan
  - Pepper-pot
  - Tomography
- Beam Halo Detection

Cartoon of Imaging System



Goals of Imaging System

- Linear Acquisition: Light Intensity $\propto$ Beam Density
- Have sufficient spatial and temporal resolution
- Eliminate light not from beam
- Extract quantitative information about beam: center, size, rotation angle



Objectives of today's experiment

- Acquire familiarity with an imaging system.
- Learn to process beam photos for different beam distributions including beams with halo
- Use a pinhole measurement to estimate emittance
- Use beam optics to image an aperture with the beam

Outline

1. Imaging Hardware
   - Screens
   - Cameras and Light Collection Systems
   - Camera Control and Capture

Wide variety:
focus on UMER lab's

2. Digital Representation of Images – File Formats
3. Image Processing

More General

4. Outline of Experiment

Screens: Converting Beam Density to Light

- Phosphor-coated glass screens:
  - P43 (Gd_2O_2S:Tb) decay < 1.5 ms
  - GK31 (ZnO:Ga) decay < 5 ns
  - coating with Aluminum (50-100 nm) prevents charging and protects Phosphor
- Scintillating crystals
  - YAG ($Y_3Al_5O_{12}$:Ce) decay < 70 ns
  - YAP ($YAlO_3$:Ce) decay < 25 ns
- Optical Transition Radiation (Friday)

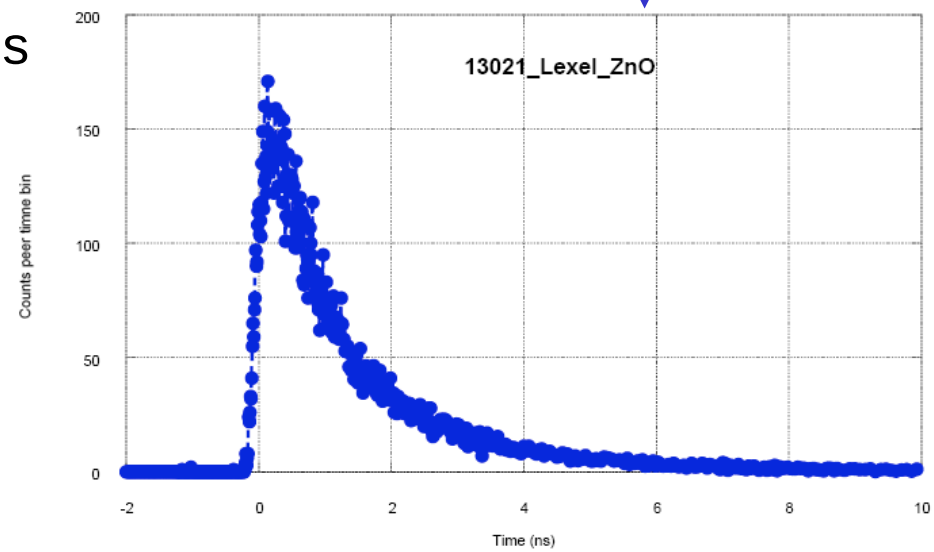


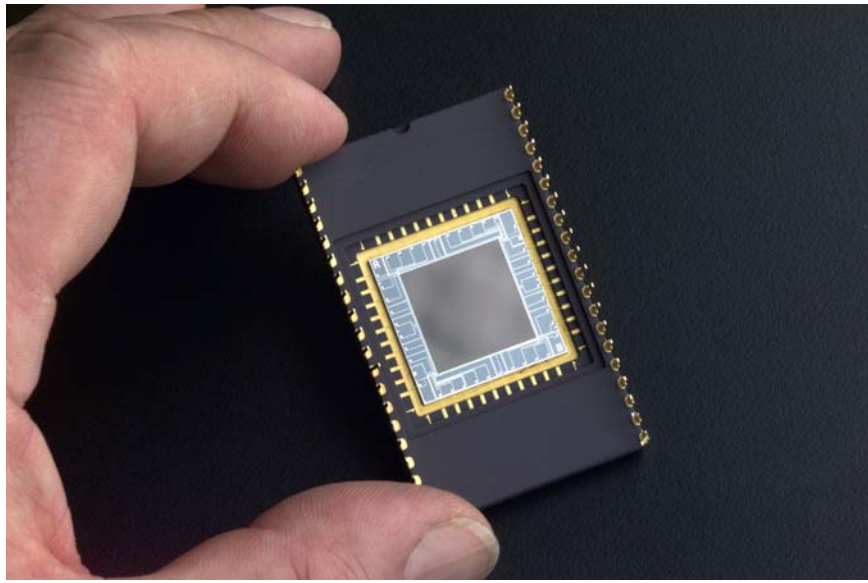
Figure credits: Lexel, Inc.

Light Intensity vs. Response Time

Credits: Ralph Fiorito

Cameras

- Charge-Coupled Device (CCD)
  - Converts light to charge on a 2-D bank of capacitors
  - Data passed sequentially from pixel to pixel (so involves reading time)



- Active Pixel Sensor (CMOS)
  - Each pixel has own amplifier
  - More scalable (faster for high resolution), cheaper, but noisier

Photo credits: Wikipedia

Example: UMER Camera

IMPERX Camera Specs

Sensor	CCD
Resolution	1000x1000 pixels
Bit Depth	Up to 12 bit
Area Scan	Progressive
Repetition Rate	30 Hz
Shutter Speed	Up to 1/50,000

Spatial Resolution enables detection of fine structures and smaller beams

Bit depth = dynamic range, want high for detecting faint halos

Most other video cameras are "interlaced", *i.e.,* skip every other line

Faster Rep Rate means faster data collection

Synchronization: want shutter open only when beam is on – minimize bg

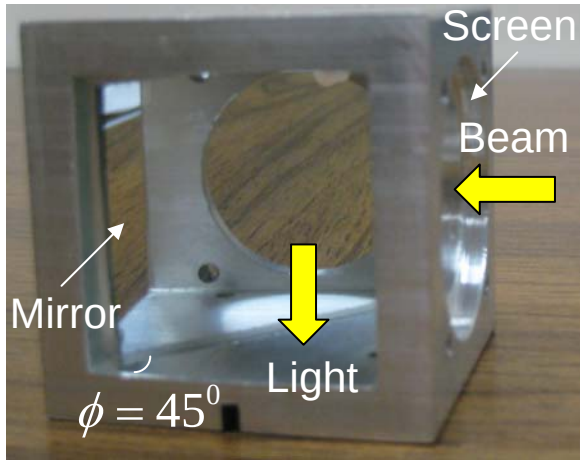
For fast screen, we use a 16-bit PIMAX Gated, Intensified CCD

Cooled CCD – lower noise

Light Collection Systems

Options

- Aperture (f/#): How much light is collected (want wide: low f/#, but without saturation)
- Magnification: Ability to project beam to largest area of CCD sensor for resolution
- Focus: must be able to focus at the given lens to screen distance



Frame Integration

- for faint light output (e.g., from fast screen)
- assumes beam stable from shot to shot
- integrate light over multiple beam pulses

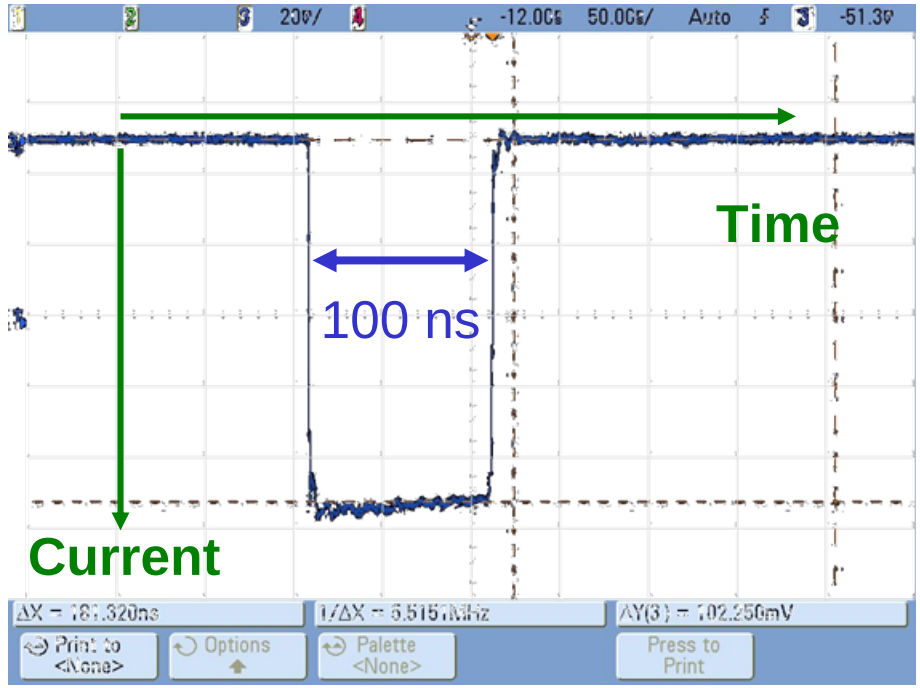
Nikkor Macro lens

F = 60 mm
Aperture: 2.8 -32

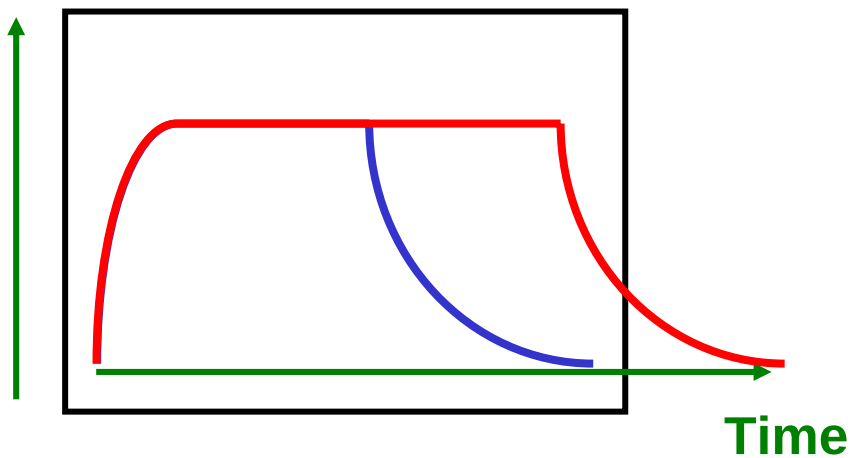
Photo credits: Diktys Stratakis

Time-Dependent Beam Distributions

- P43 screen integrates over 1.5 ms >> beam pulse
- Photo includes effect of particles in beam ends

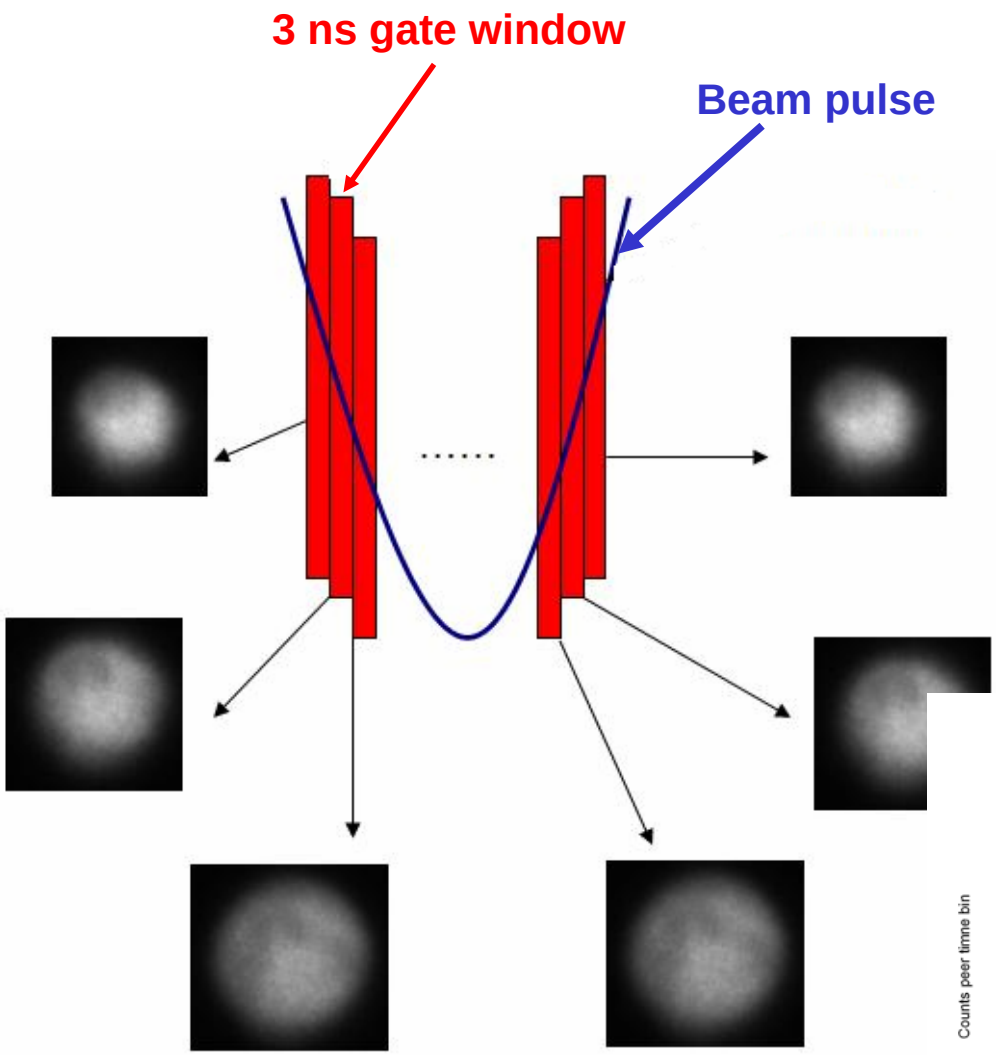


- Can asses effects of beam ends by changing pulse length

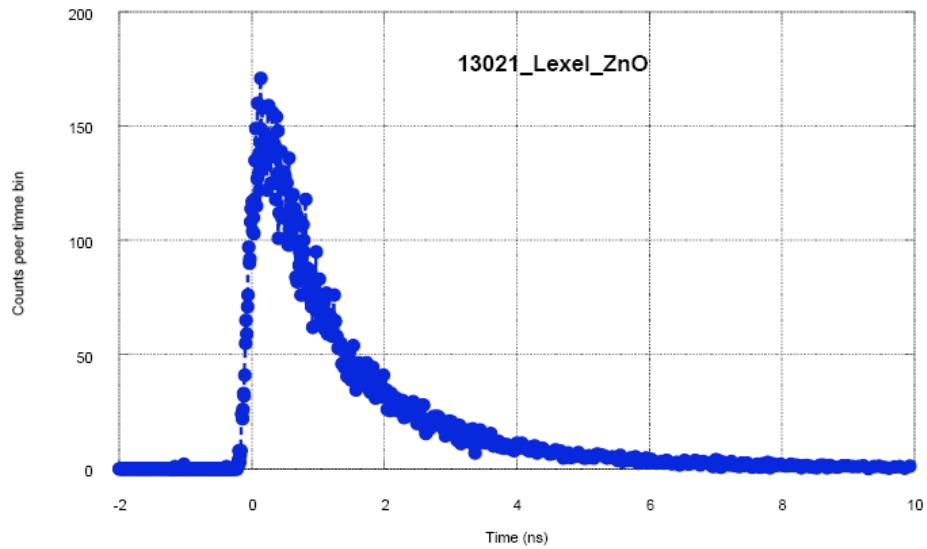


Current trace credits: Brian Beaudoin and Santiago Bernal, LSE

Time-Resolved Imaging

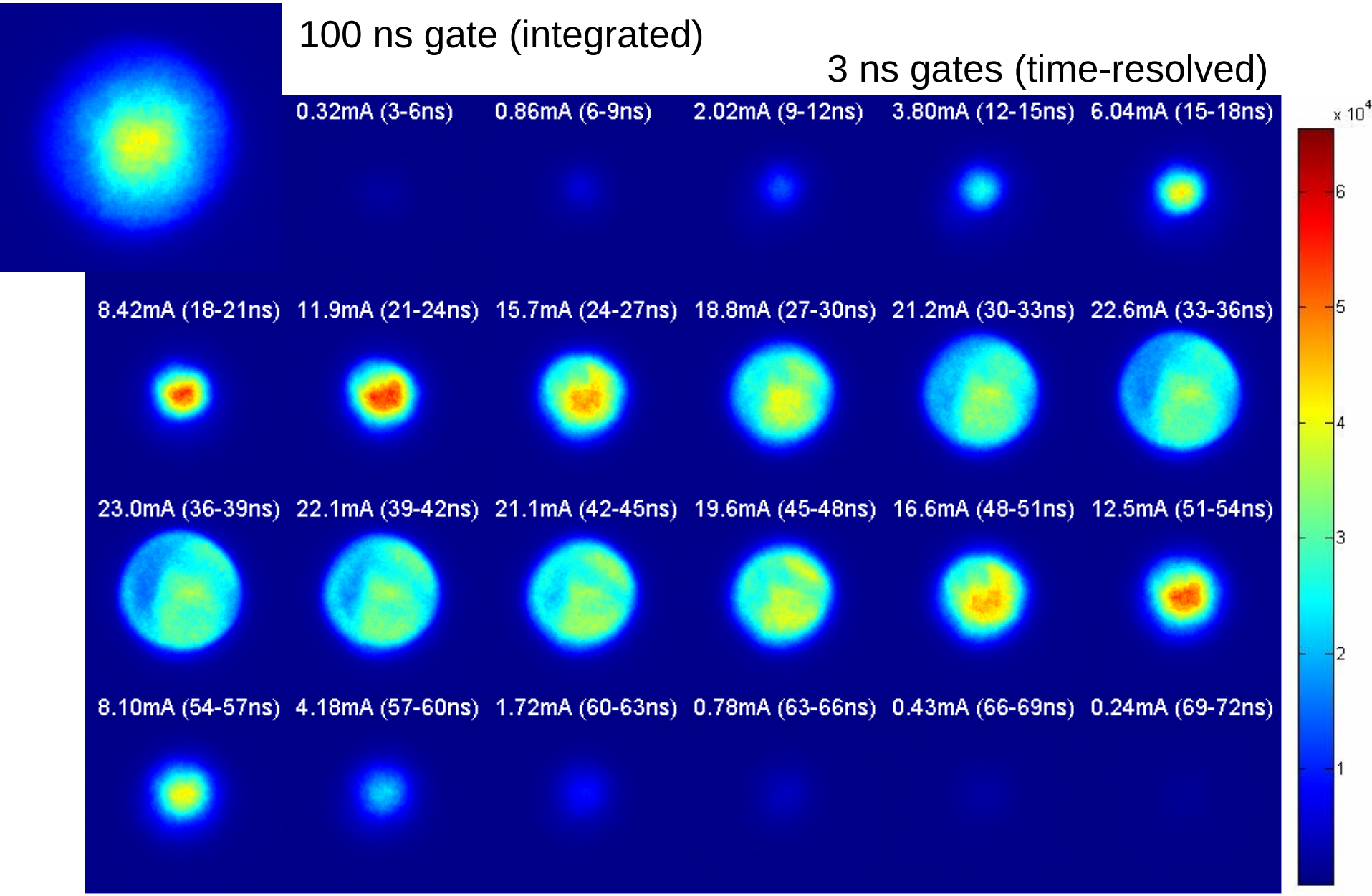


- ZnO:Ga Fast Screen
- PIMAX ICCD Gated Camera
- ~100 frames integrations = ~2 s
- 3 ns “gate” on camera



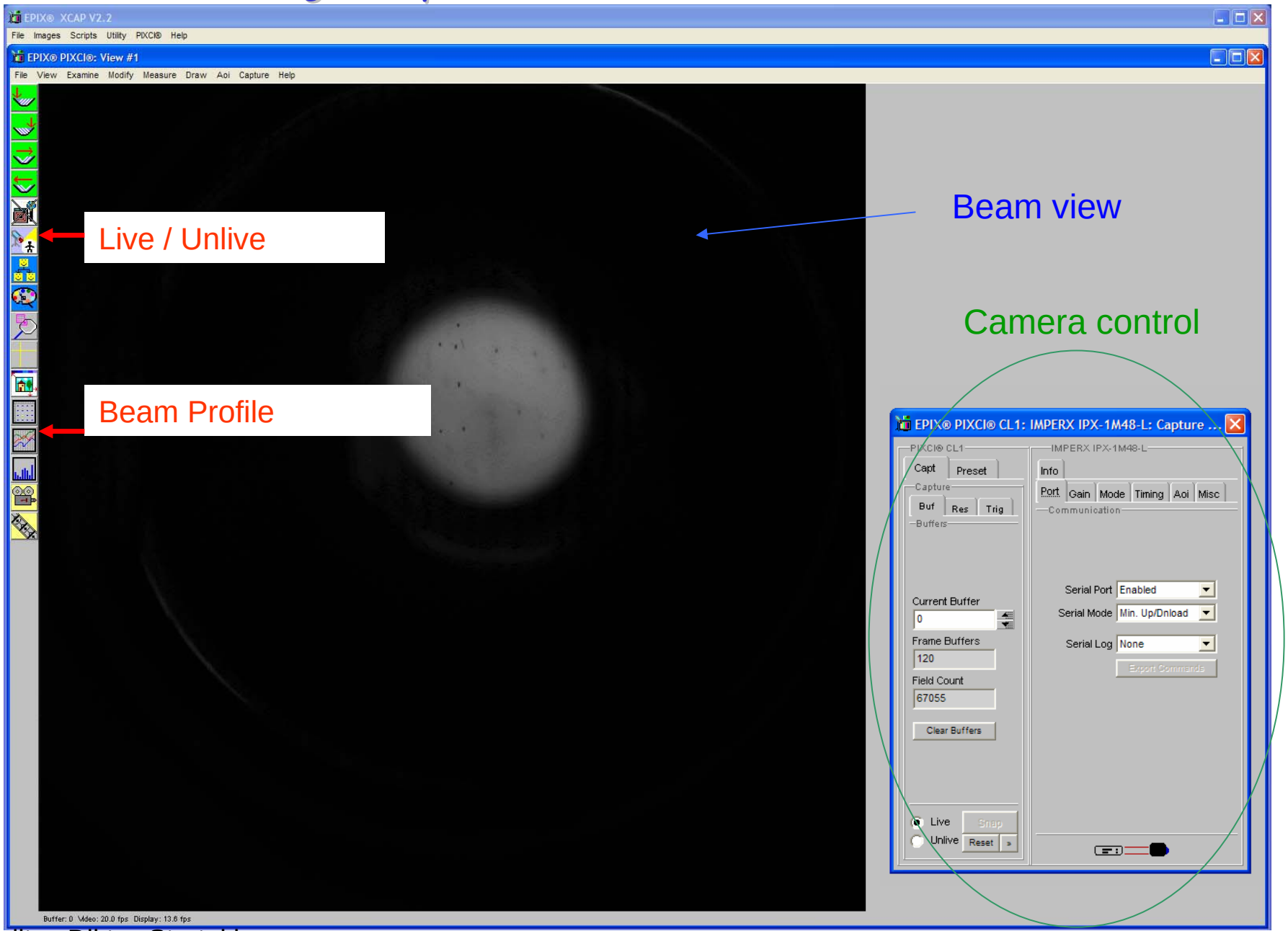
Credits: Kai Tian, LSE

Example: Time-sliced images of parabolic beam



*Tian, Kishek, O'Shea, Fiorito, Feldman, and Reiser, Phys. Plasmas **15**, 056707 (2008).*

Image Capture: EPIX XCAP Software



Credits: Diktys Stratakis

Acquisition Settings

Mode

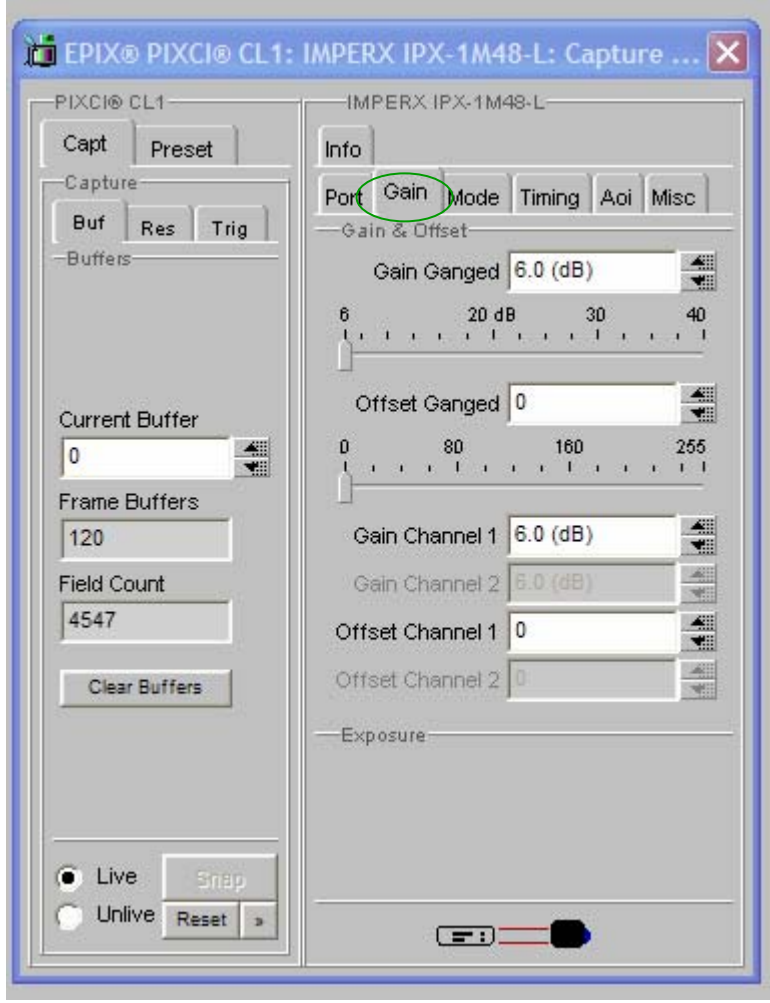
- External Trigger / Fast

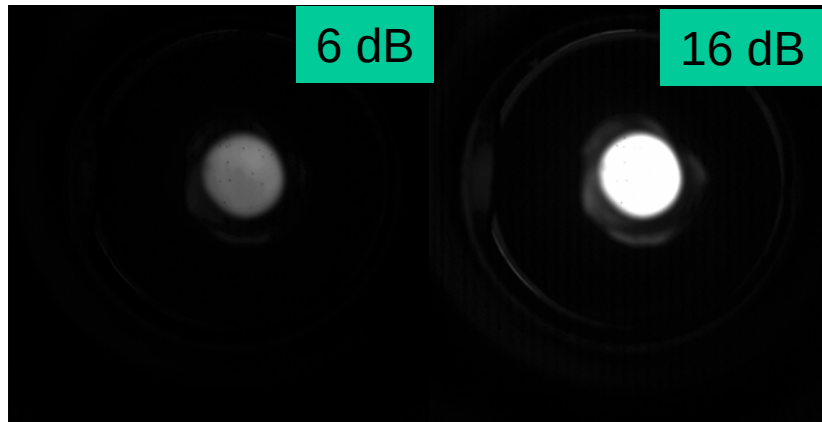
Timing

- single-tap / 12-bit depth

Gain

- Increase gain to enhance photo
  - gain also affects noise
  - be careful not to saturate image
- Offset subtracts background
  - better keep at zero, do it in post-processing





Credits: Diktys Stratakis

Image Processing

- Extract and preserve useful information

- Minimize or Eliminate useless information.

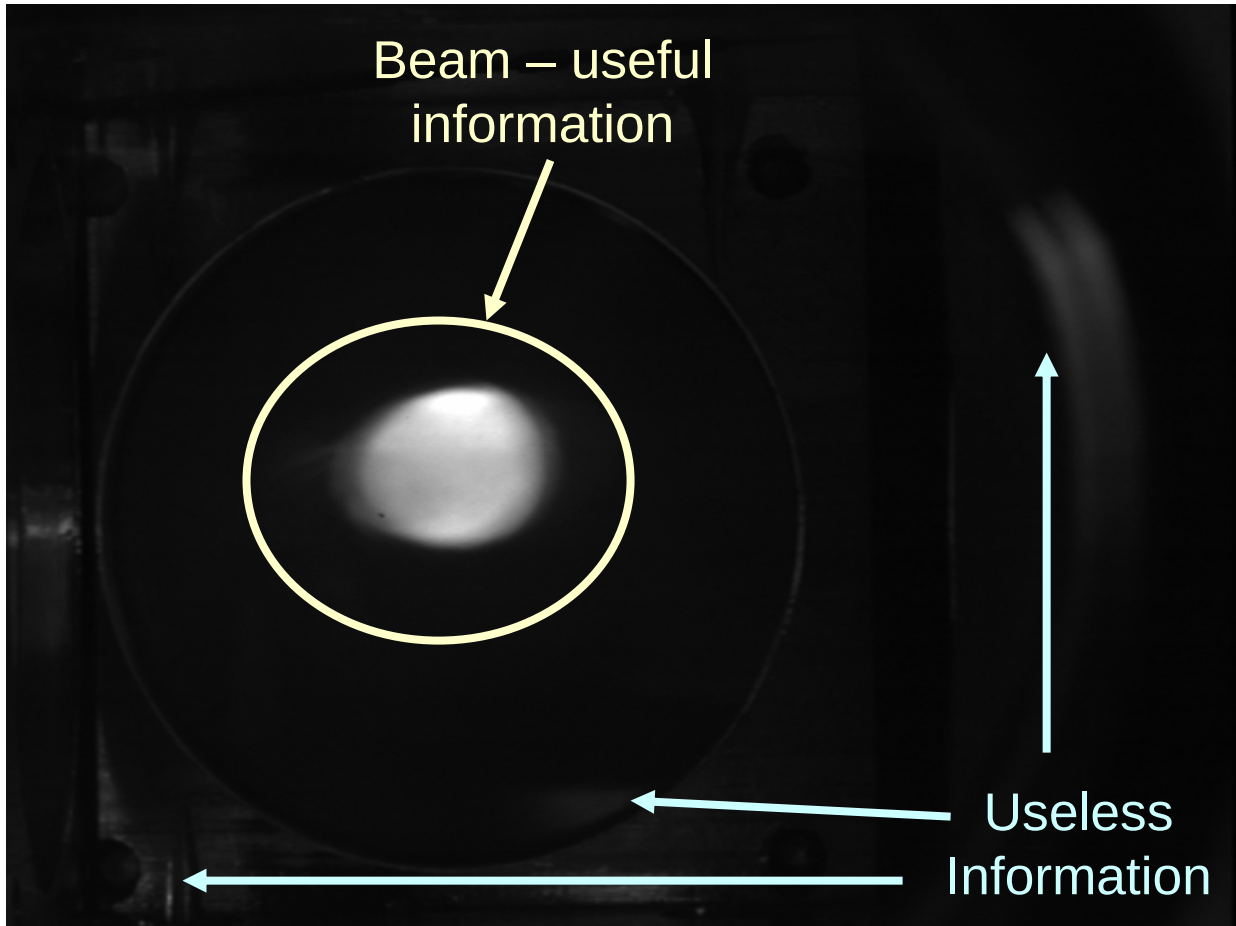


Image Enhancement: Acquisition vs. Post-Processing

Original photo should faithfully represent image on screen

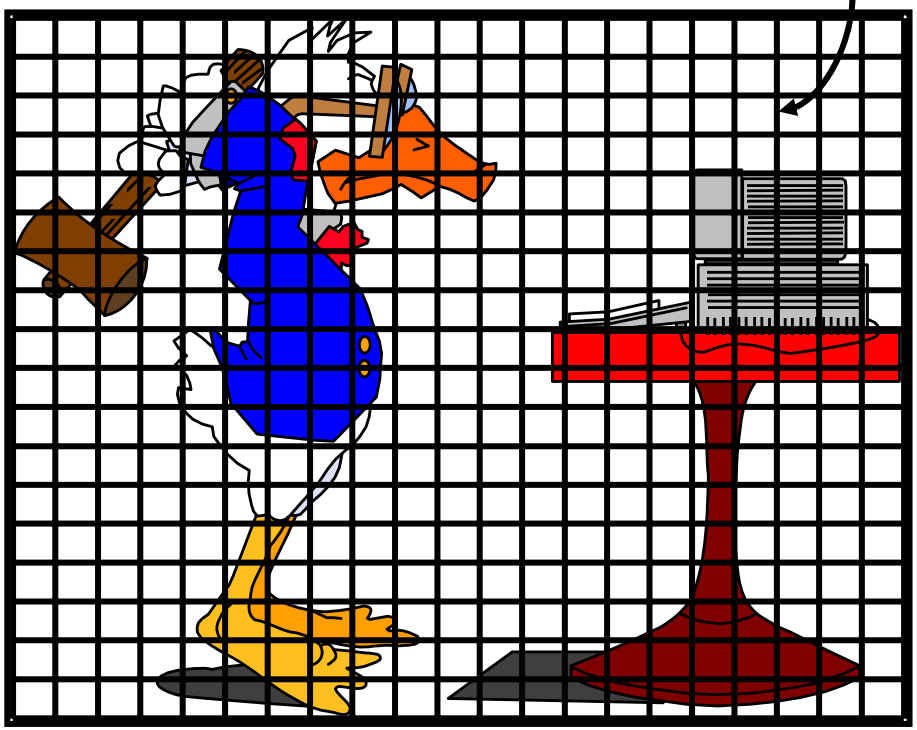
1. Control and optimize experimental conditions
   - synchronize with beam pulse
   - cover camera
   - turn off room lights
2. Avoid artificial enhancement during capture

   (offset / filters / color-coding)
3. Save original photo in uncompressed format (tif or bmp), with at least as many bits as the camera is capable of
4. Reversible adjustments: Post-process copy of image, so can go back

Do not lose useful info – else have to go back to lab

Digital Representation of Images – File Formats

An Image = Color + Intensity for each “point” in space



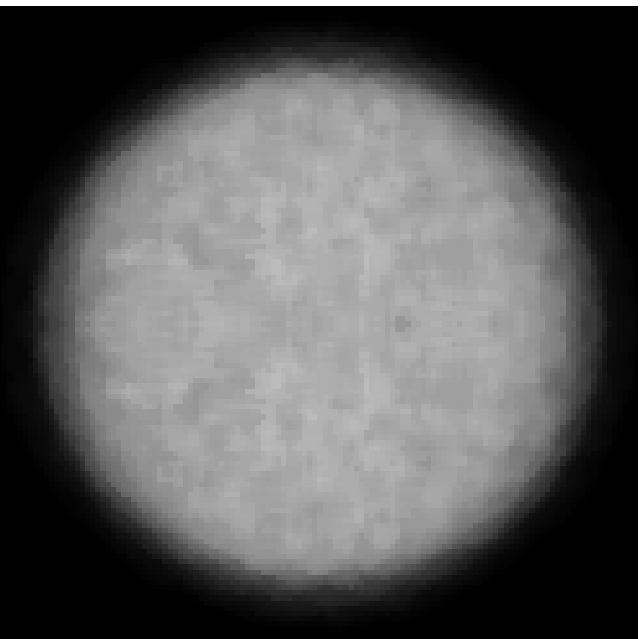
pixel picture element

Approximation:
Area inside each pixel with uniform color and intensity

Number of pixels determines resolution

Grayscale (intensity) Images (tif)

- Simplest (ignore color information)
- Represent intensity of a pixel as a number from 0 (black) to 1 (white).
- Vertical (intensity) resolution: # of gradations used (**“depth”, “bit depth”** or **“color depth”**)
  - e.g., 1 byte = 8 bits = 256 gradations
- Higher depth means higher dynamic range, but more storage / pixel.



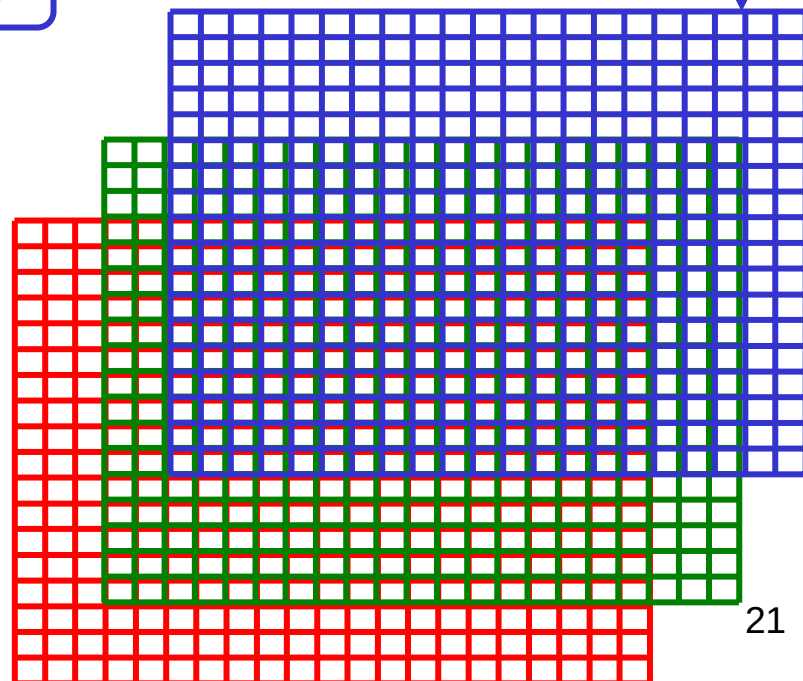
Small Header + Matrix

1/256	2	7	0	198	178	227	

Hexadecimal, 0,1,2,…,9,A,B,C,D,E,F
00 = darkest; FF = brightest

Color Images

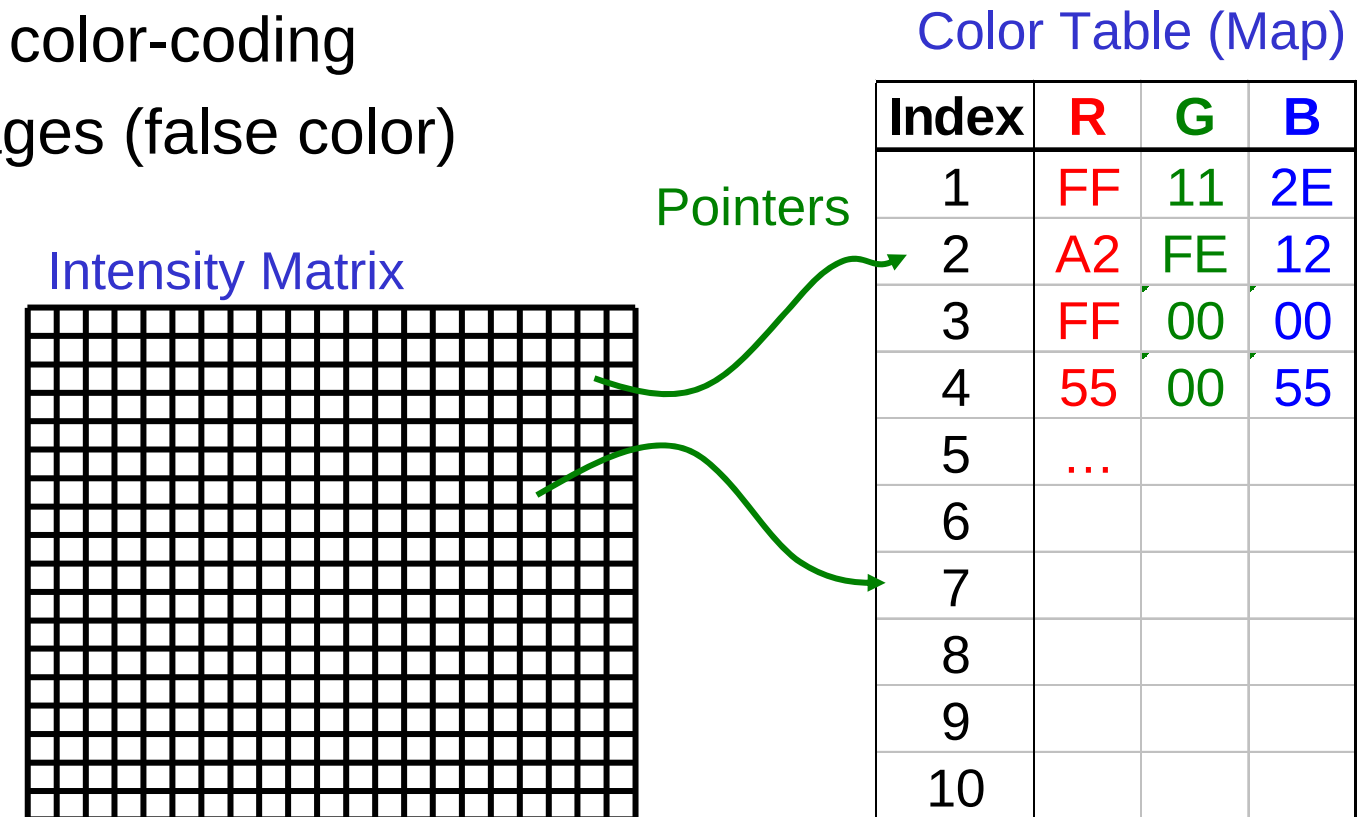
- Can compose any color by mixing the 3 Primary Colors: Red, Green, and Blue
- Several schemes for representing that information:
  - RGB (Red, Green, Blue) Images - “Truecolor” (*tif*)
    For each pixel, provide intensity of each of the primary colors.
  - Indexed (Colormap or bitmap, e.g. *bmp*)
  - Compressed RGB (e.g., *tif, jpeg, GIF*) used to reduce file size
  - CMYK (Cyan-Magenta-Yellow-Black)
  - HSL (Hue-Saturation-Luminance)



RGB too inefficient for large pictures.

Indexed Images (Colormap or Bitmap)

- Certain combinations of colors are more likely to appear.
- Intensity Matrix + Color Table (colormap)
  - e.g., 256 color - have 256 elements in colormap.
  - Can have greater depth in colormap without adding more bytes to every pixel.
- Great way for color-coding grayscale images (false color)



ImageProcess.m – Application to beams

- MATLAB script to process one or more beam images, based on MATLAB Image Processing Toolbox
- Automated – can handle large number of photos

For more Info on MATLAB commands:

```
>> help images        % lists image commands
>> help command
```

Contributors:

- *–Rami Kishek: originator, since 1997*
- *–Santiago Bernal: feedback and testing*
- *–Ksenia Danylevich: helped enhance original version*
- *–Marcel Pruessner: wrote menu-driven user interface*
- *–Han-Chih Shieh: added rotation angle moment*
- *–Matt Holland: extended to process pepper-pot data*
- *–Hui Li: fixed bug in rotation angle, rewrote in C++*
- *–Diktys Stratakis*

Operations

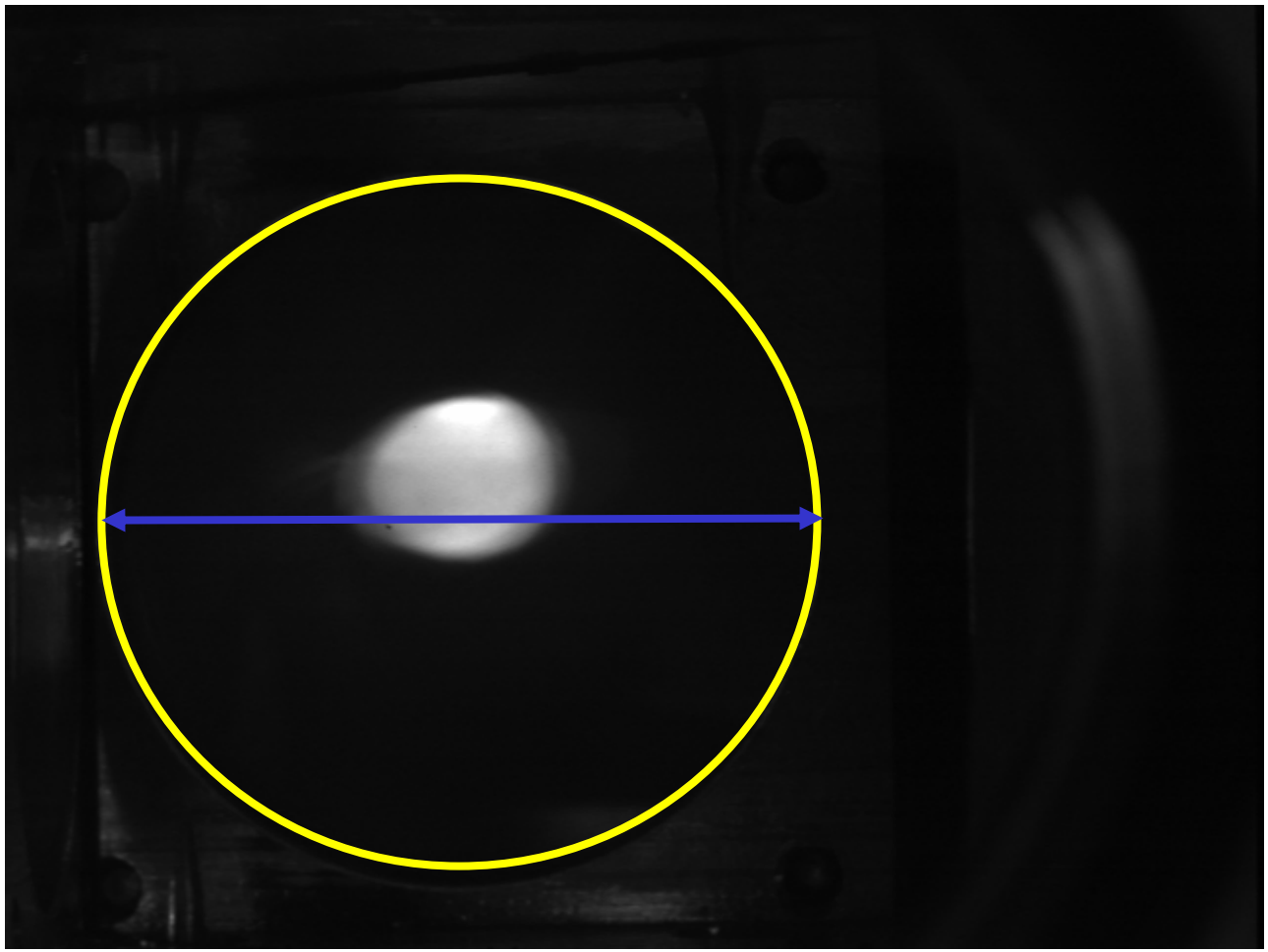
1. Calibration
2. Resizing and scaling
3. Circle Cropping
4. Median Filtering
5. Threshholding
6. Moment Calculations
7. Output
8. Color-coding
9. Logarithmic Scaling

Operations on the intensity matrix (irreversible)

Operations on the colormap (reversible)

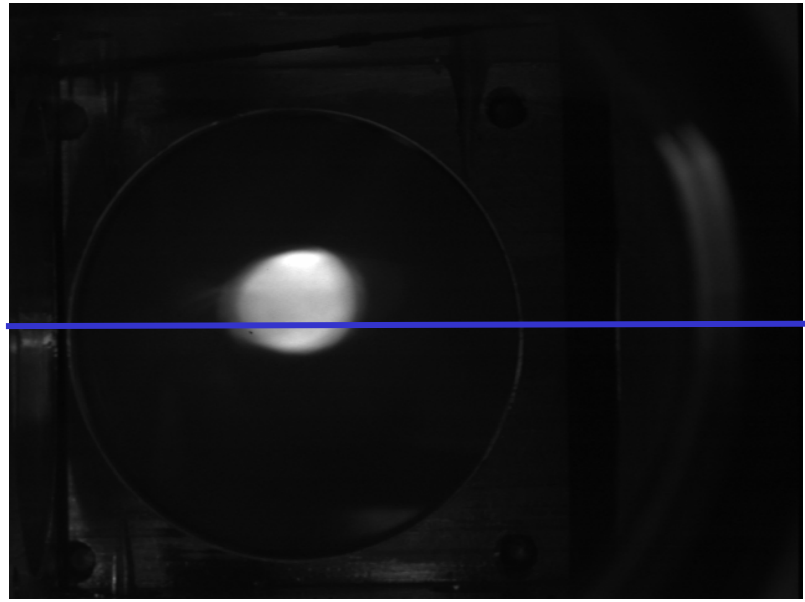
Calibration – using screen edge as a marker

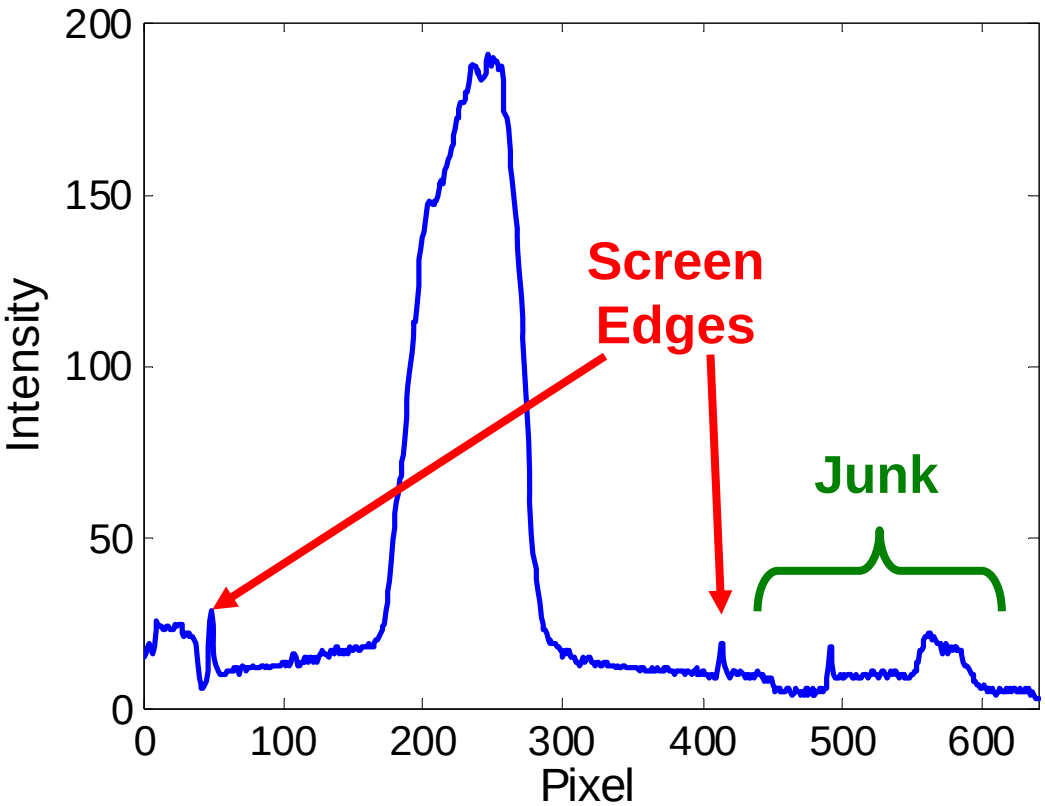
$$\text{mm\_per\_pixel} = \frac{\text{\# pixels across object}}{\text{known size of object}}$$



Beam Profile and Circle Cropping

```
>> U = imread('UMERPhoto.bmp');
>> imshow(U)
>> size(U)
   480   640

>> plot(U(260, :))
```





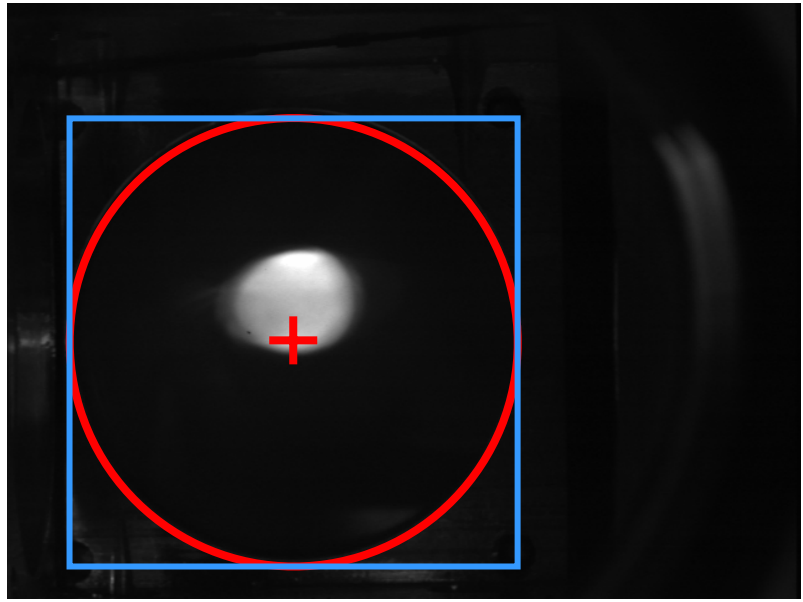
Circle Cropping

Specify:

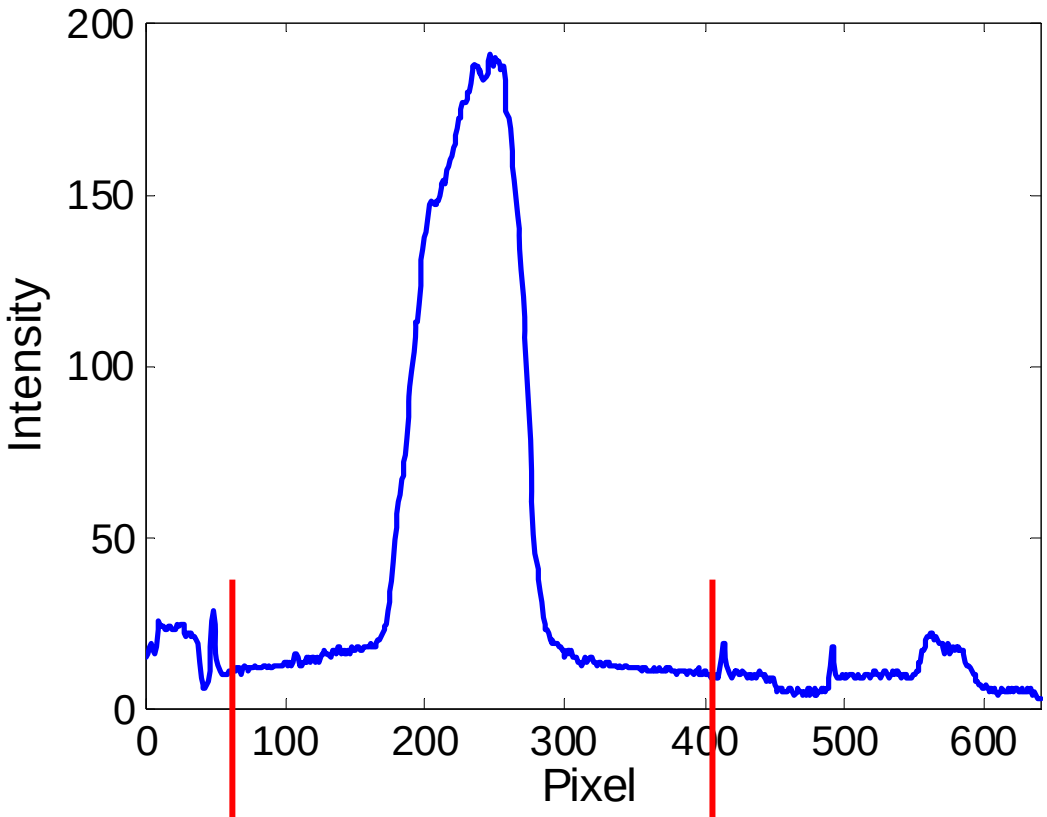
Pixel coordinates of screen center

Number of pixels across inner diameter of screen

```
>> cent=[230,270];
>> rad = 179;
```



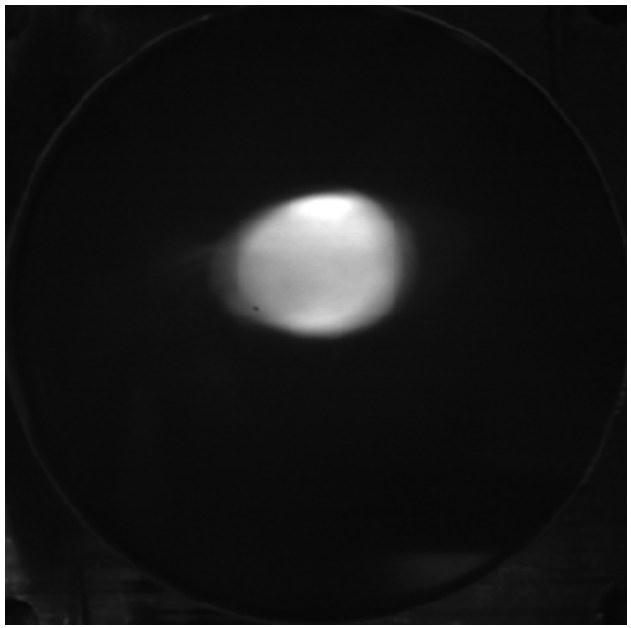
Crop square outside screen



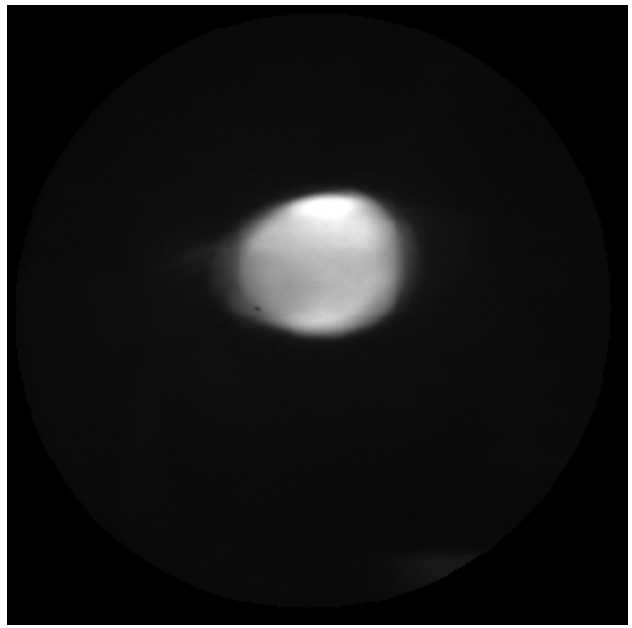
Iterate for each pixel:

if distance of pixel from center greater than screen radius, set its intensity to zero

Cropping Example



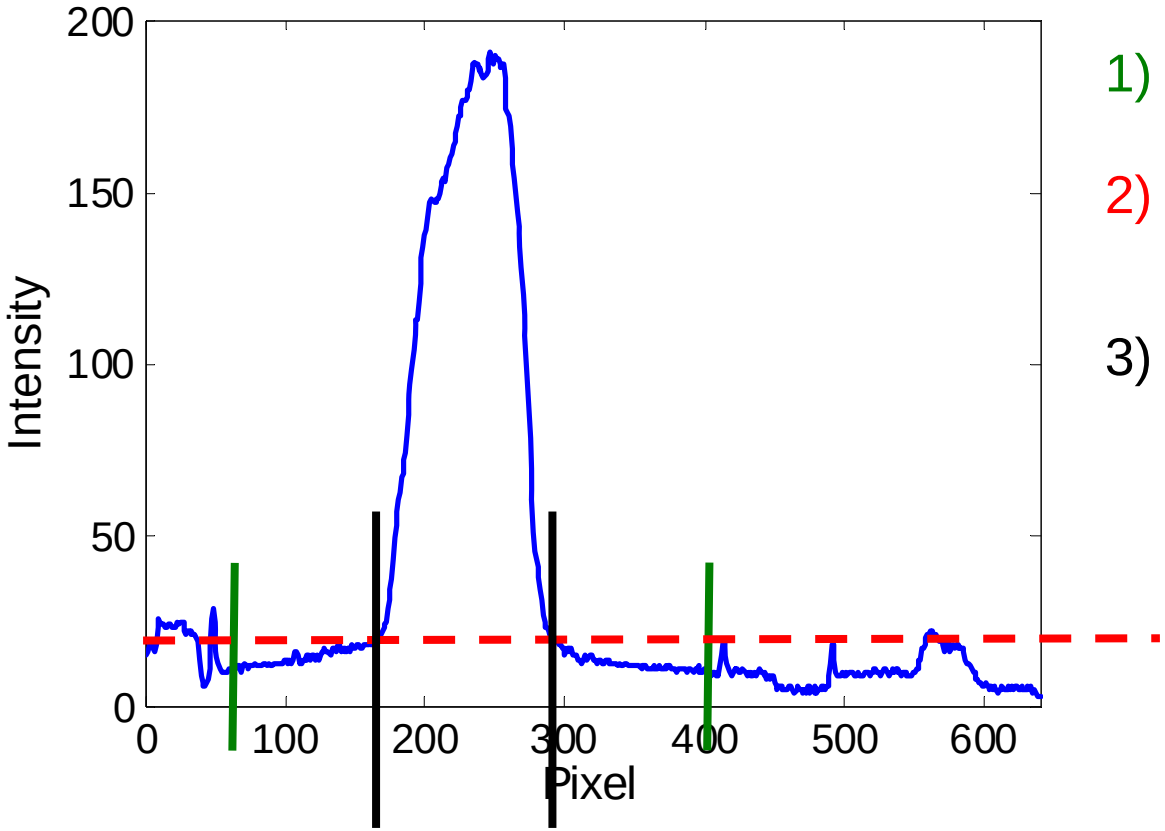
1) Center Square on Screen



2) Crop Outside Inner Screen Edge

Thresholding

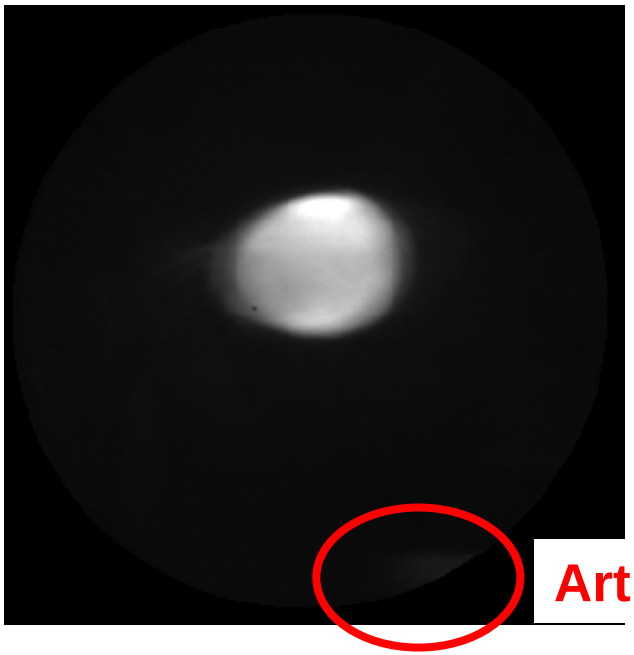
- Background affects moment calculation, so eliminate



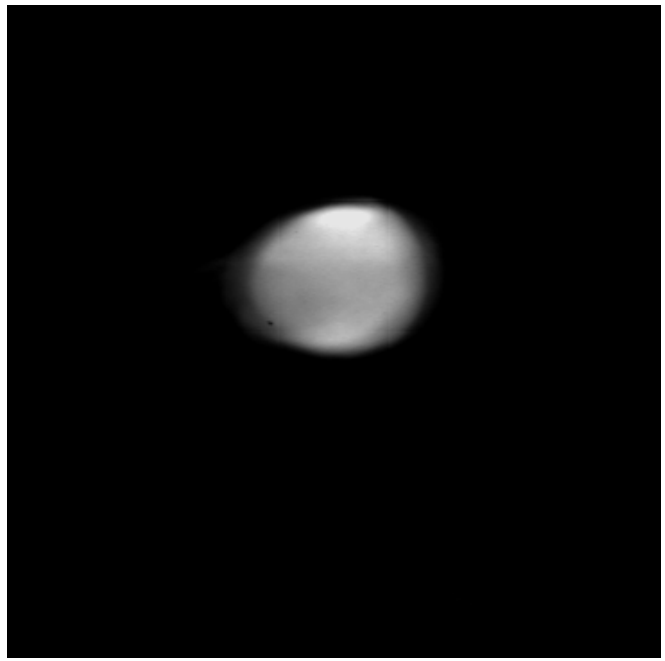
1) Crop screen as before
2) Impose threshold at a certain level
3) Find pixels with intensity below threshold
   i. set intensity to zero, or,
   ii. subtract threshold

- Use lowest possible threshold, or will lose beam information

Thresholding Example



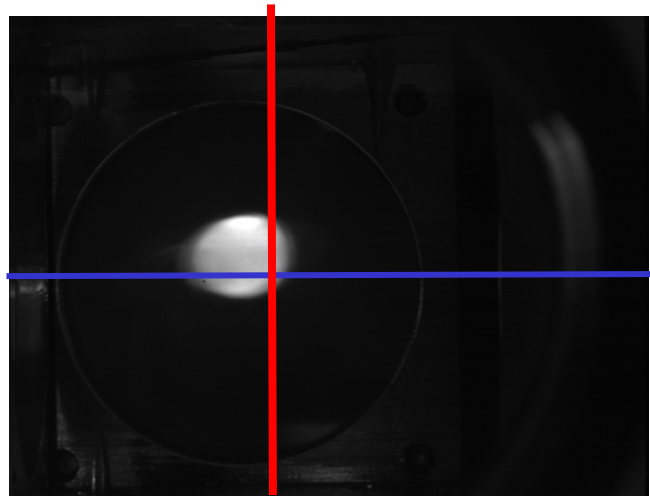
1) Before Threshold

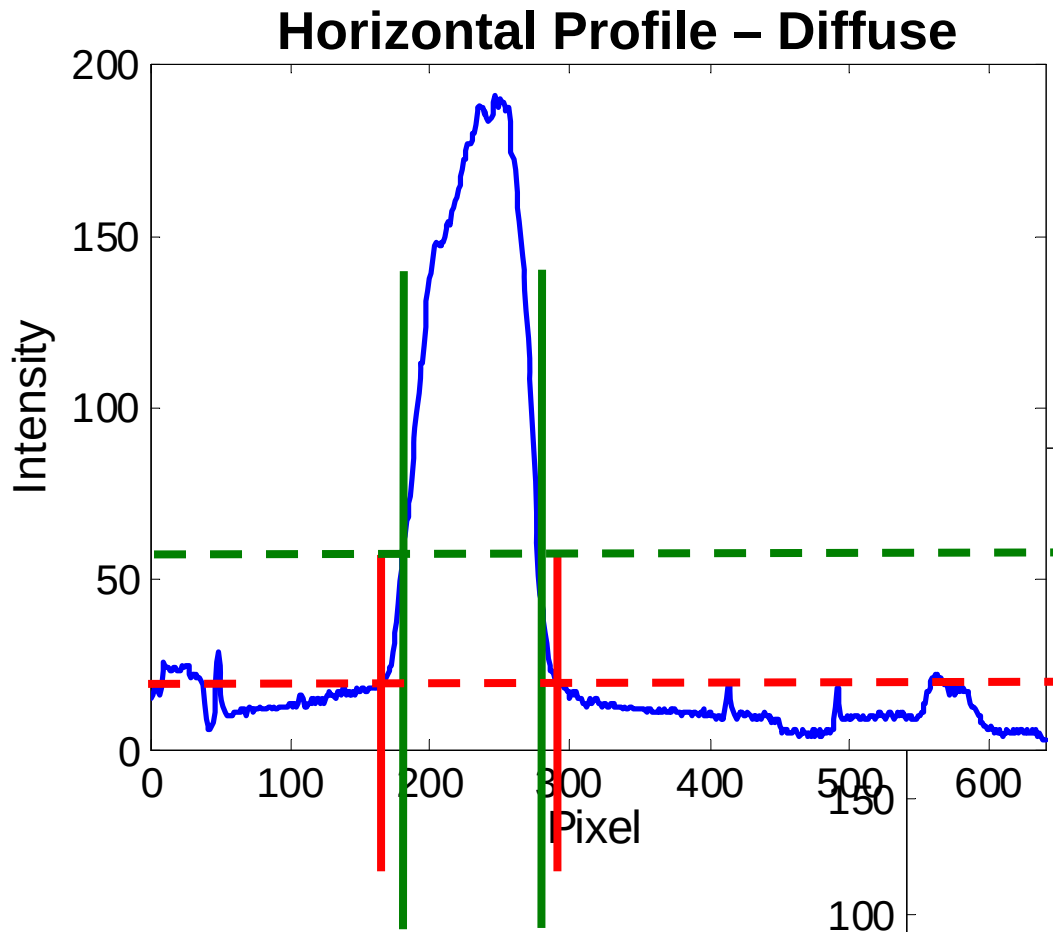


2) After Threshold (25th level)

```
>> K = J - 25;
>> K(find(K<0)) = 0;
```

Thresholding considerations





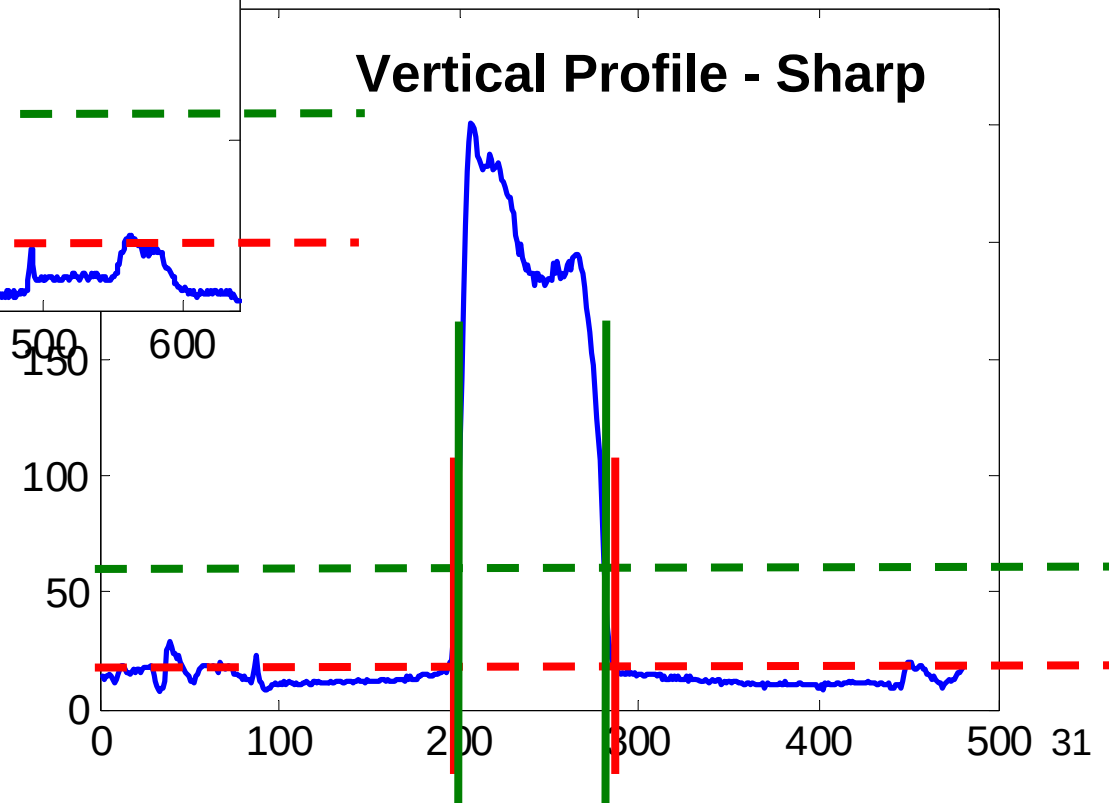
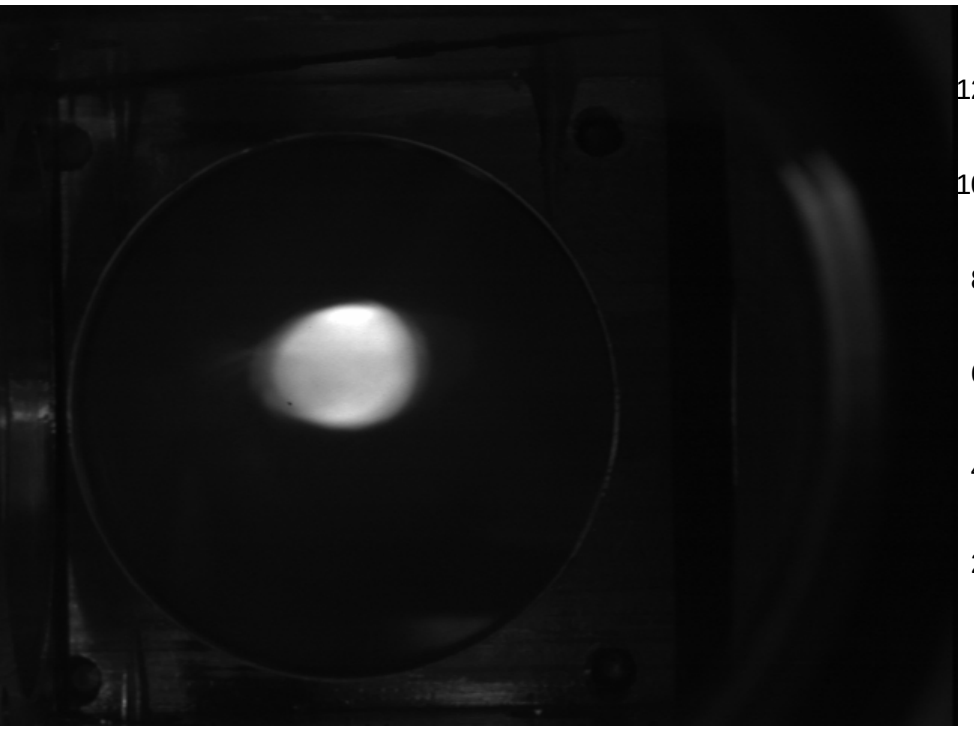
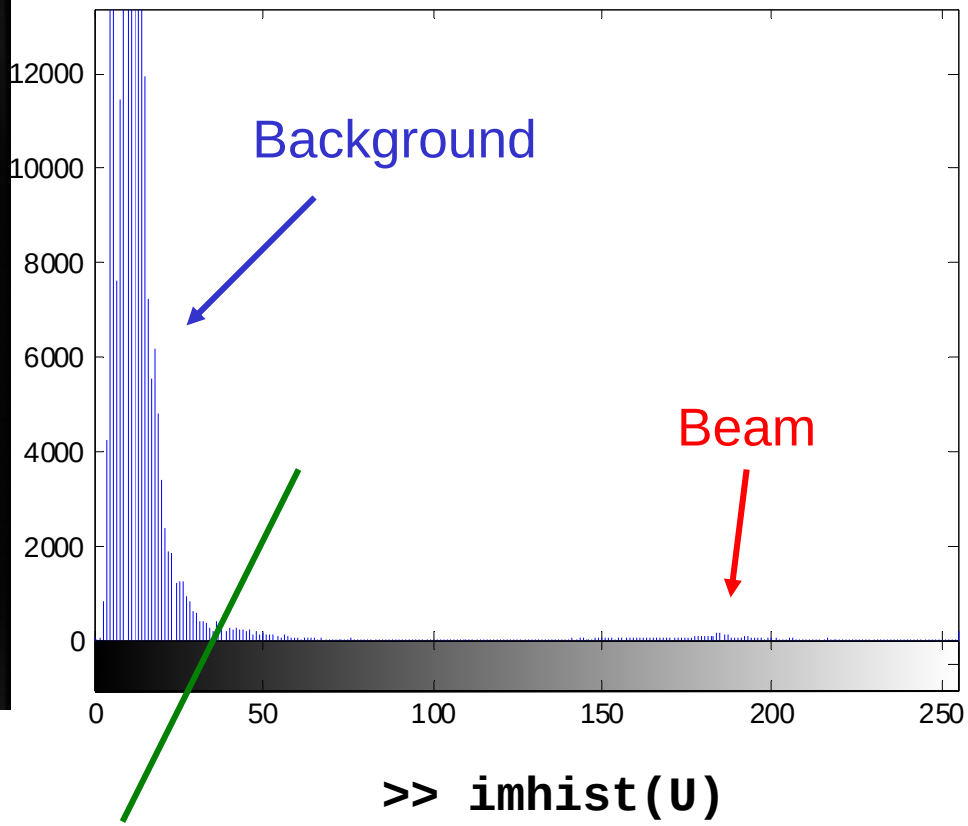


Image Histogram



Distribution of pixels by intensity

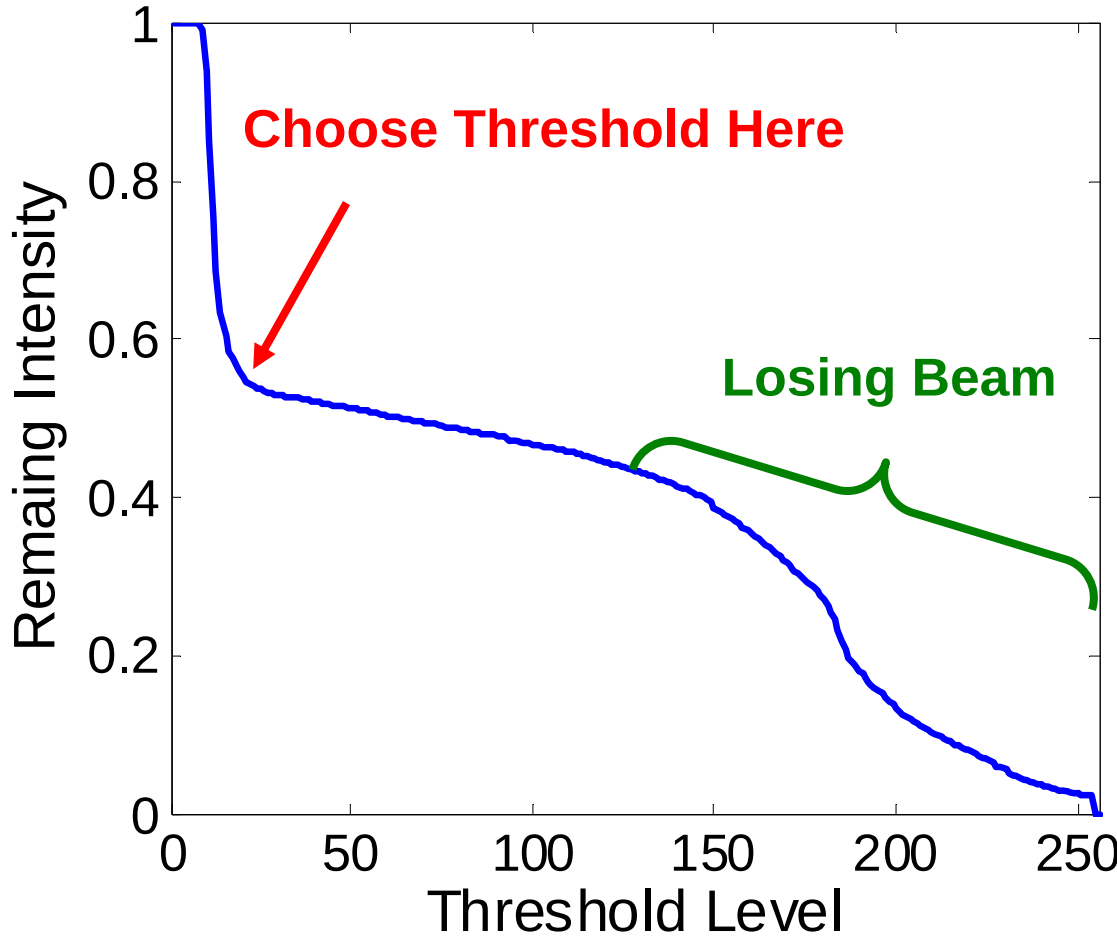



```
>> imhist(U)
```

How can we quantify thresholding effect?

Add up total pixels remaining after applying threshold

Total Intensity



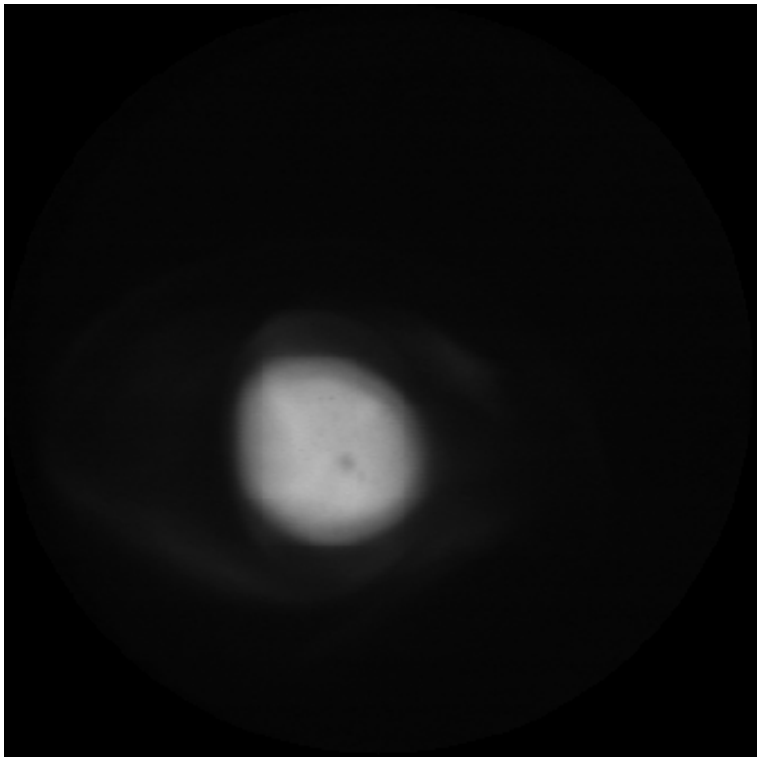

```
MATLAB
>> I=zeros([1,256]);
>> for t = 1:256
     I(t) = sum(J(find(J>t)));
   end

% Normalize
>> I = I/sum(sum(J));
>> plot (I)
```

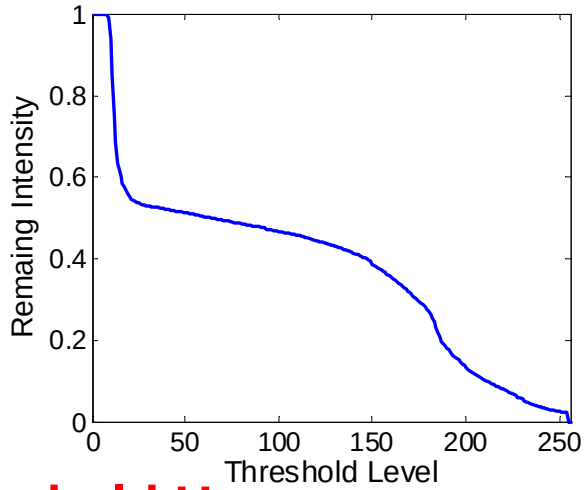
Related to Histogram

$$I(t) = 1 - \int_0^t i * H(i) di$$

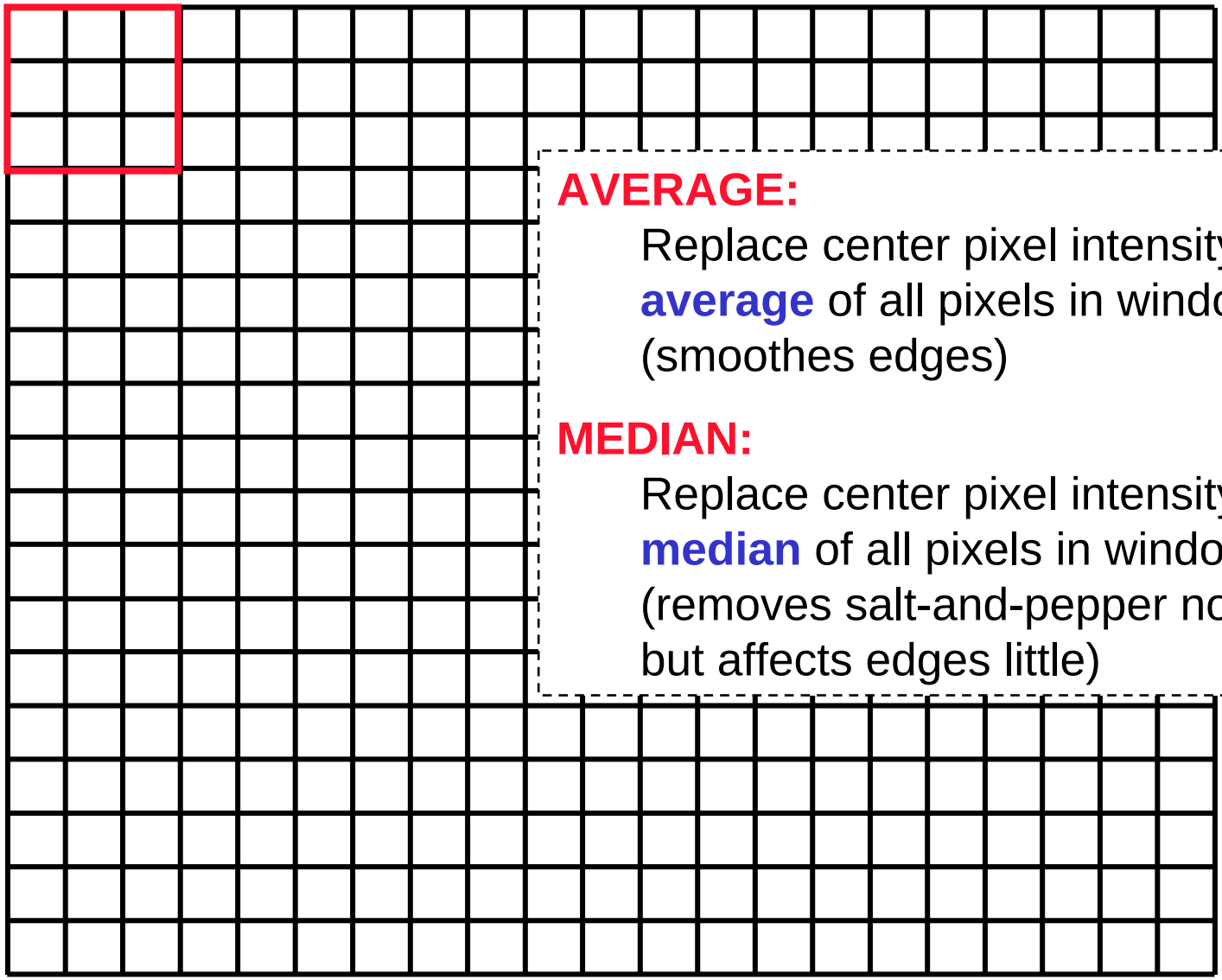
Caveat – What if there's a halo?

Different Photo





Filtering – Moving Window



AVERAGE:

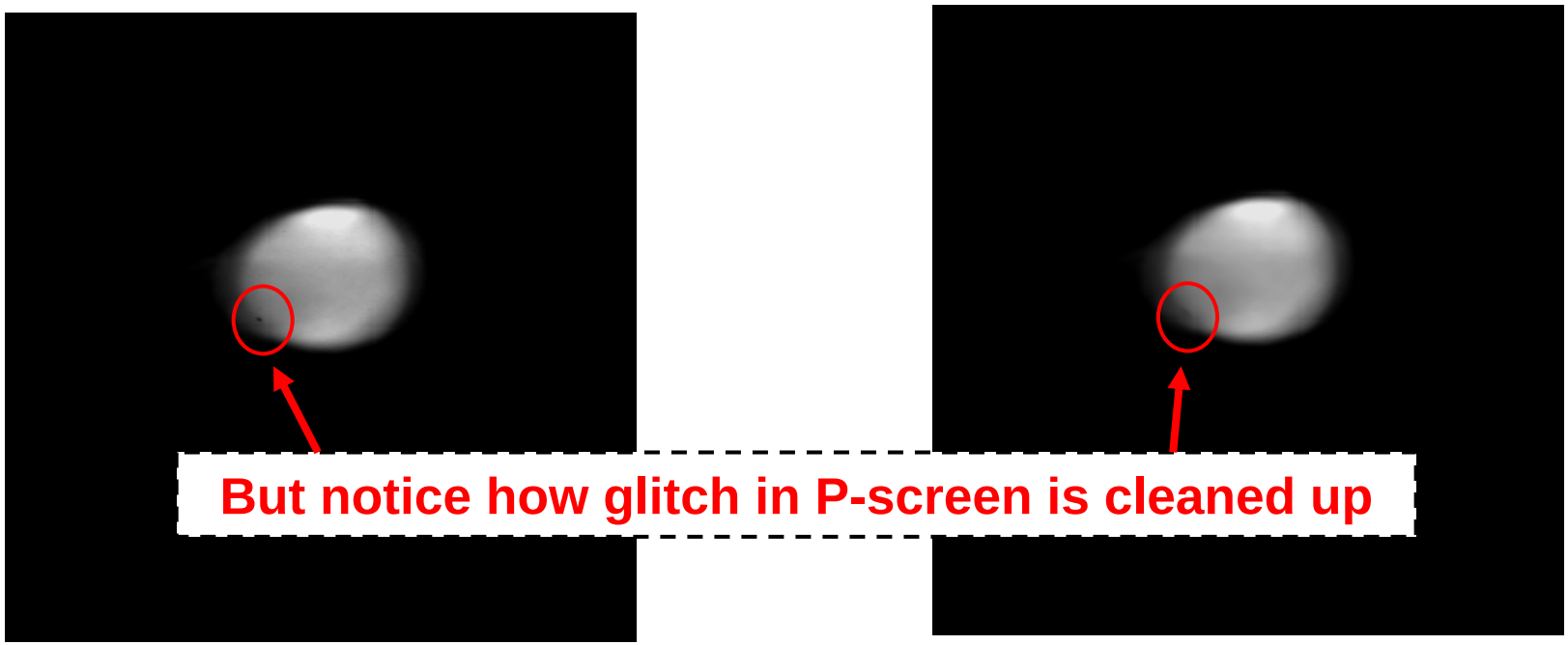
Replace center pixel intensity with **average** of all pixels in window (smoothes edges)

MEDIAN:

Replace center pixel intensity with **median** of all pixels in window (removes salt-and-pepper noise, but affects edges little)

Median Filtering

Preserves most features, as desired, including sharp beam edges



No Filtering

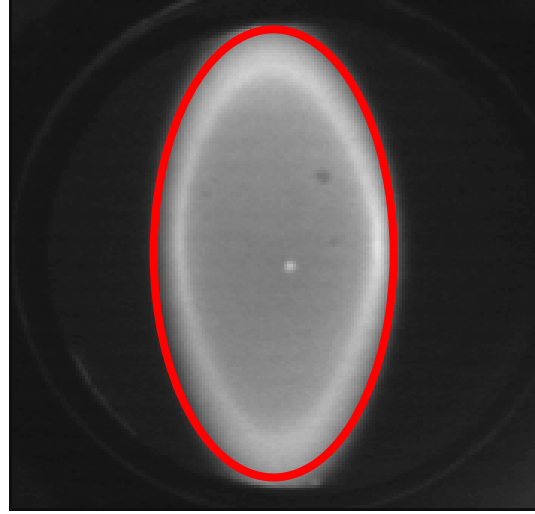
5x5 Median Filter

```
>> M2 = medfilt2(K, [5 5]);
>> imshow(M2)
```

Filtering helpful in cleaning up photos from simulations,
which are usually noisy due to the use of a discrete grid and few particles

Review Moments: Centroid and rms Radius

$$n(x,y) = \iint f(x,y,x',y')dx'dy'$$



Beam Ellipse

beam centroid

$$x_c = \langle x \rangle = \frac{\iint x n(x,y)dxdy}{\iint n(x,y)dxdy}$$

rms beam radius definition

$$x_{rms} = \tilde{x} = \sqrt{\langle x^2 \rangle} = \sqrt{\frac{\iint x^2 n(x,y)dxdy}{\iint n(x,y)dxdy}}$$

Similarly for y_c and y_{rms} …

For a uniform beam, a = 2 x_{rms} , b = 2 y_{rms}

Moment Calculations: Centroid, Size, and Rotation Angle

- Discrete grid, replace integrals with sums
- Apply calibration factor, *p* [mm/pixels], to convert results to mm
- For an n x m image:

$$T = \sum_i \sum_j I_{ij}$$

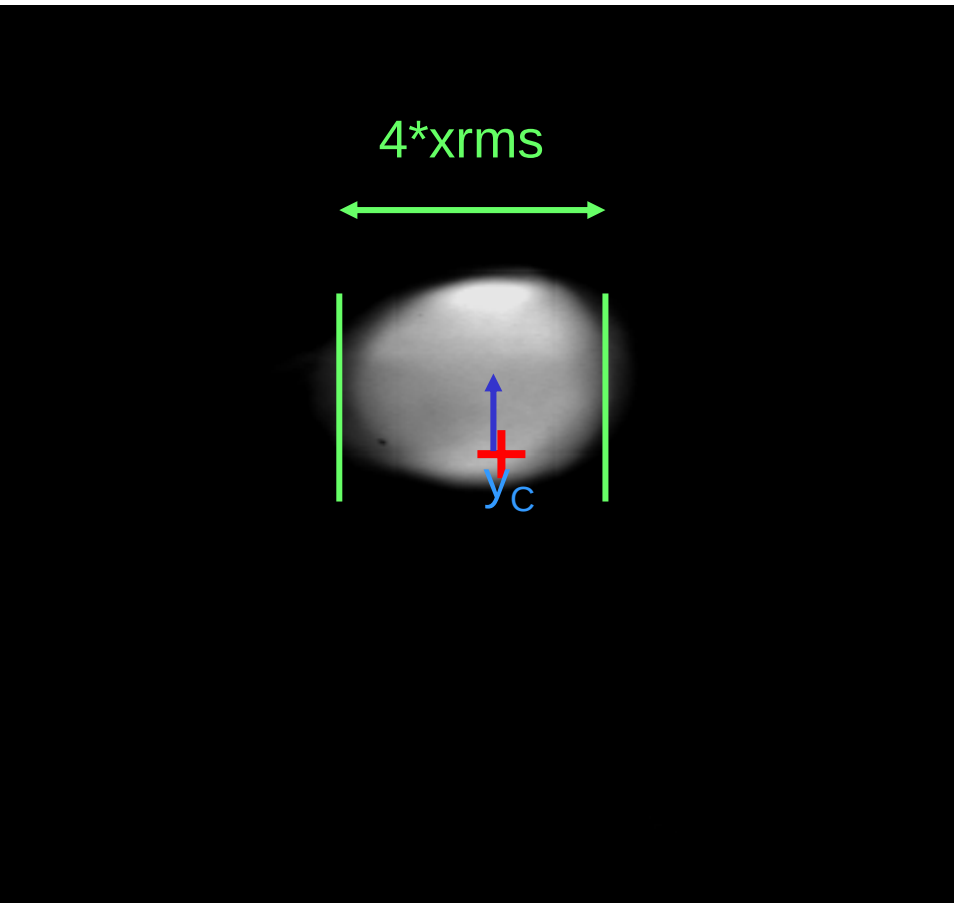
where I_{ij} is the (cleaned up) intensity matrix (M2 or K)

$$\langle x \rangle = \frac{p}{T} \sum_i \sum_j (i - m/2) I_{ij} \qquad \langle x^2 \rangle = \frac{p}{T} \sum_i \sum_j (i - m/2)^2 I_{ij}$$

$$\langle y \rangle = \frac{p}{T} \sum_i \sum_j (j - n/2) I_{ij} \qquad \langle y^2 \rangle = \frac{p}{T} \sum_i \sum_j (j - n/2)^2 I_{ij}$$

$$\langle xy \rangle = \frac{p}{T} \sum_i \sum_j (i - m/2)(j - n/2) I_{ij}$$

Illustration



E.g., Assuming
mm_p_pix = 0.088

```
xc   =   0.7189
yc   =  -2.3509
xrms =   4.2240
yrms =   3.7334
rot  = -13.9983
```

Serious Rotation

Output

Plot profiles

```
>> plot( K(iy,:) )
```

Display photos on screen

```
>> imshow(K, [0, 255])
```

Save processed photos to disk

```
>> imwrite(K, 'photo.tif')
```

Waterfall plots (overlaid profiles from different photos)

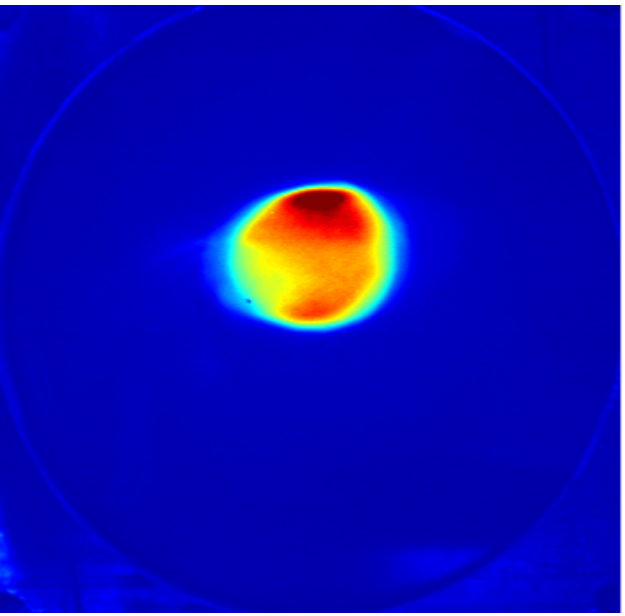
Montages (collage of different photos)

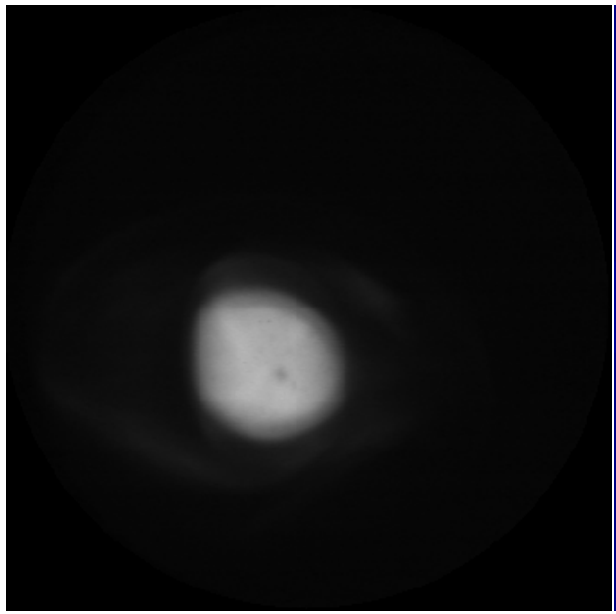
Movies (time sequence of different photos)

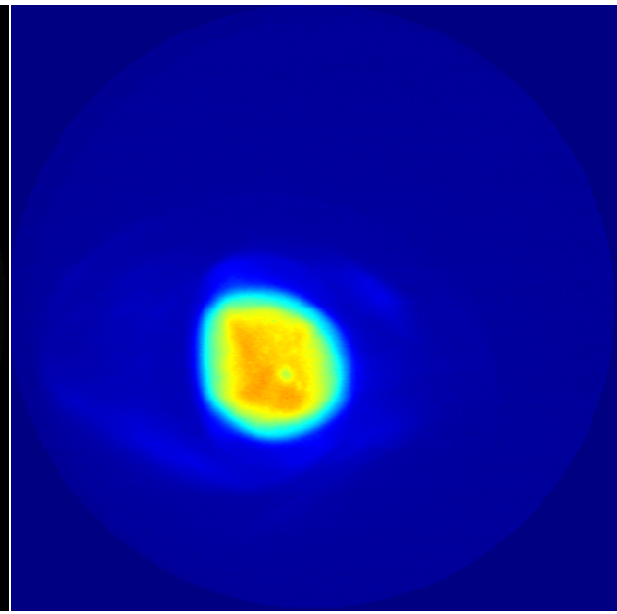
Contrast Enhancement – it's all in the map!

Indexed images:

- intensity matrix – preserves original grayscale picture
- colormap – change to artificially enhance photo







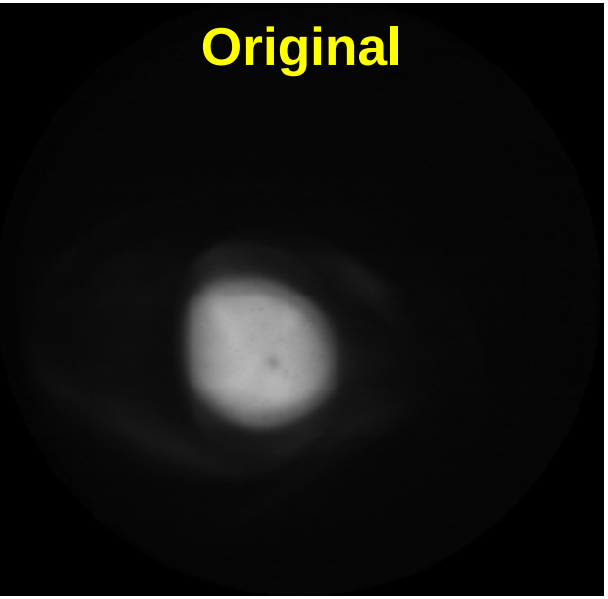
```
>> colormap('jet')
```

```
>> imshow(X, jet)
```

```
>> colormapeditor
```

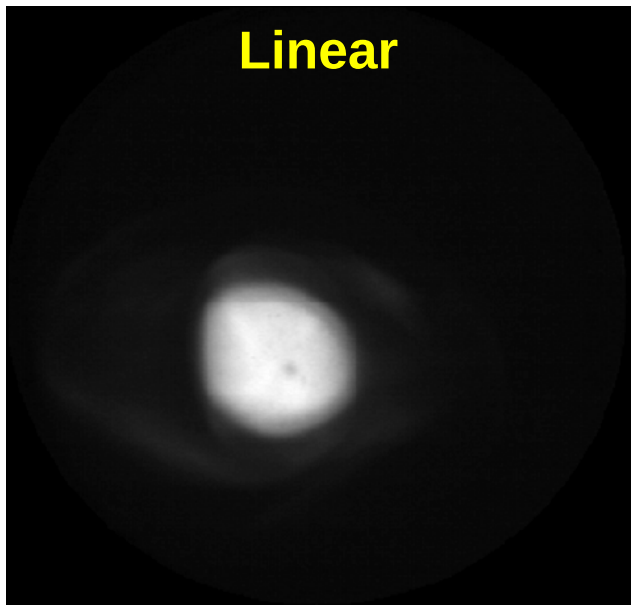
Can be misleading: Relationship between intensity and color nonlinear

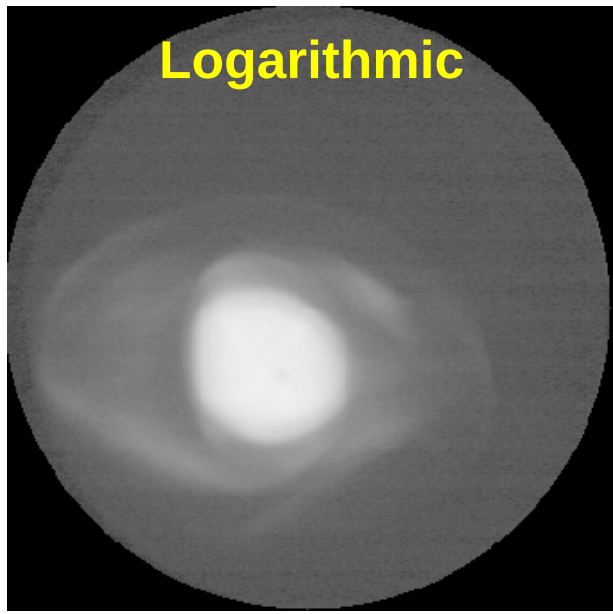
Linear vs. Nonlinear Contrast Enhancement

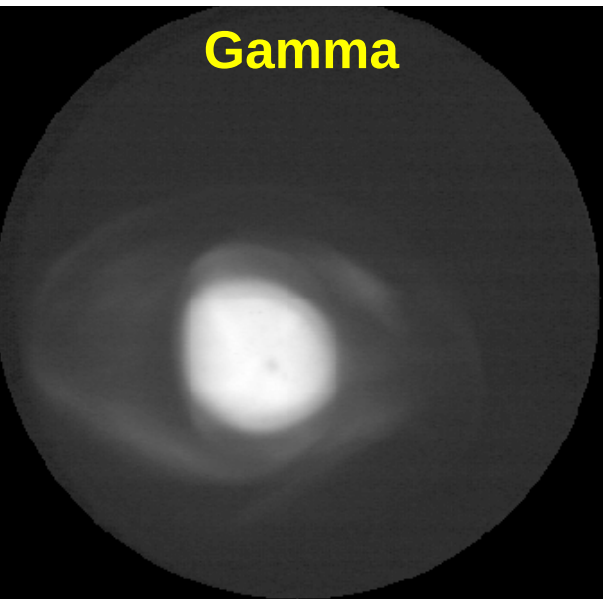


Save as bmp to preserve original intensities;
once saved as tif, changes are permanent

```
>> map2 = imadjust(map,
        [0,185/256],[0,1.0]);
```



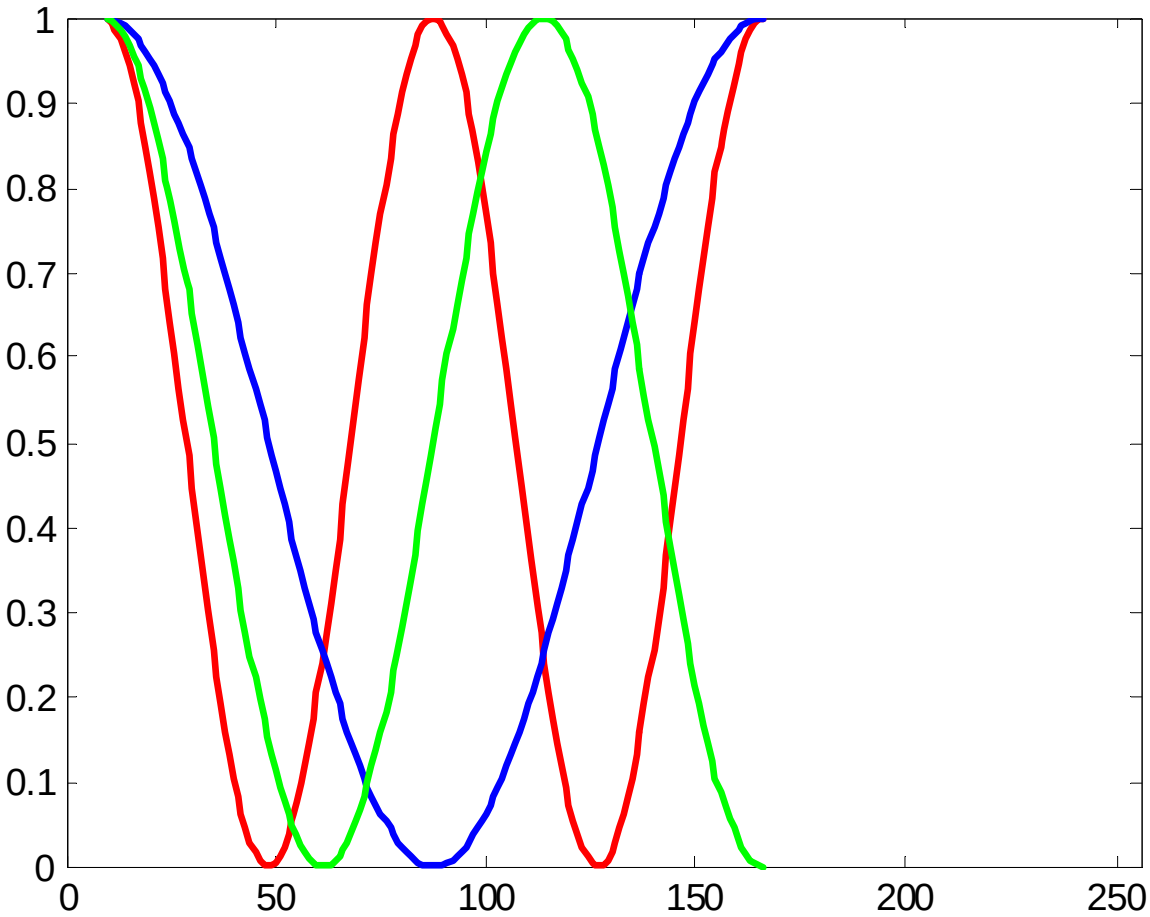





```
>> map3 = log(map+1/256);
>> n = min(min(map3));
>> x = max(max(map3));
>> logmap = (map3-n)/(x-n);
```

```
>> gammamap = imadjust(map,
        [0,185/255],[0,1], 0.5);
```

High-Contrast Colormap

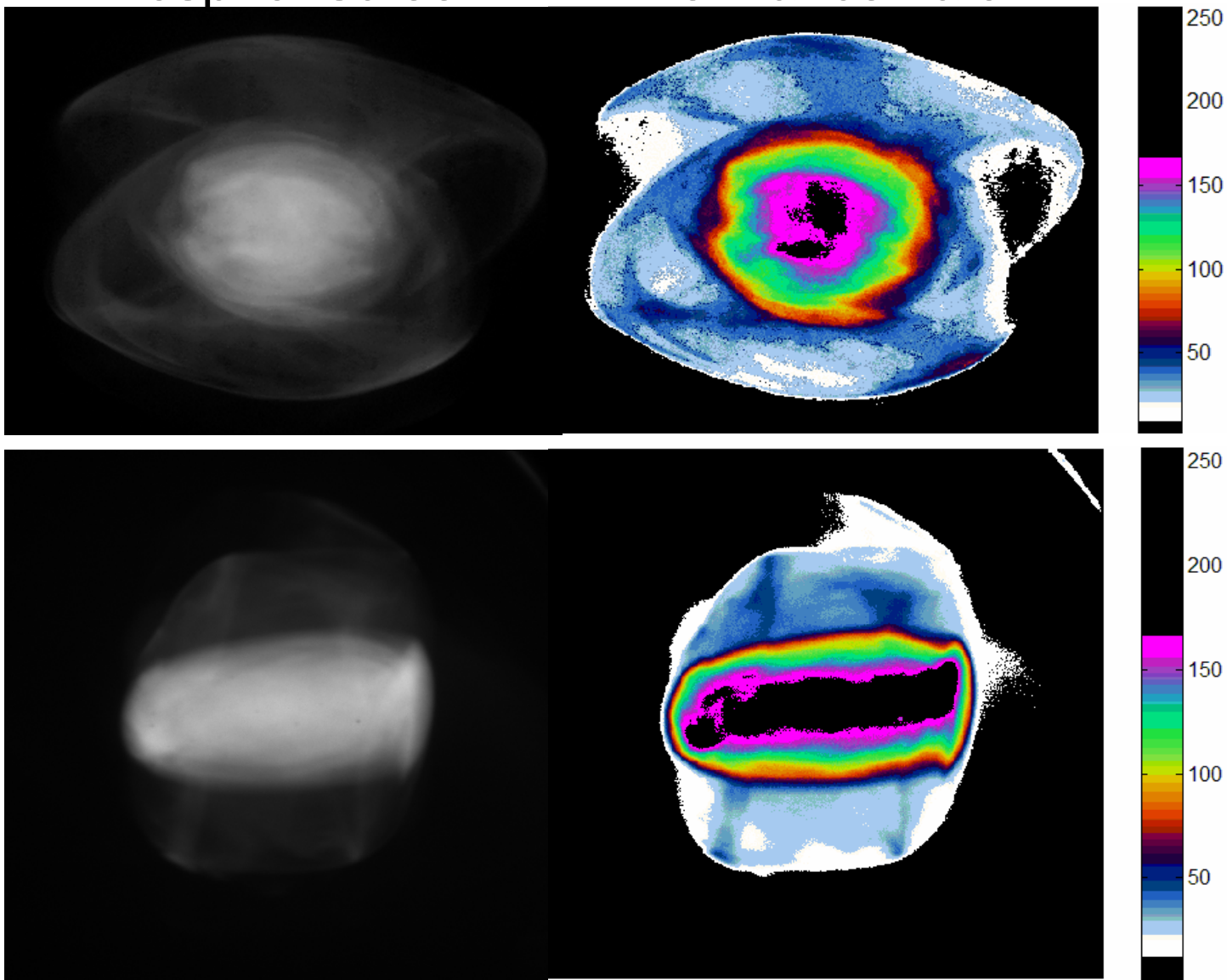



```
>> red   = (1+cos(4*pi*i/N))/2;
>> green = (1+cos(3*pi*i/N))/2;
>> blue  = (1+cos(2*pi*i/N))/2;
```

High-Contrast Colormap Examples

Beam Images from Phosphor screen

Color-coded to enhance halo



Outline of Experiment

1. Familiarize yourself with hardware, software, and settings
2. Cycle through different beam apertures and take photos
3. For each aperture, change the distribution and record photos
   - Triode guns: Change bias voltage, can form halo
   - Diode gun: Adjust synchronization with cathode heater circuit to change emittance
4. Pinhole aperture
   - like pepper-pot: image velocity space, so can estimate emittance
5. 5-beamlet aperture
   - Adjust solenoid setting to get perfect beam image of aperture on screen\
6. Process and analyze photos (offline)

Extras