Performance Improvements in NIF Optics Inspection Software

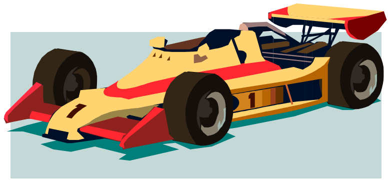
Philip Fong,

Steve Glenn, Judy Liebman, Laura Kegelmeyer

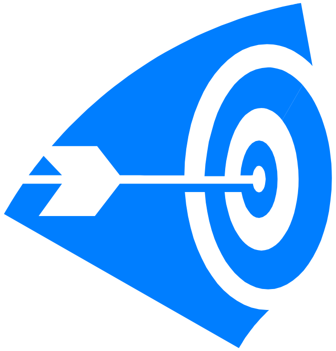
CASIS Signal and Imaging Workshop
November 16-17, 2006
UCRL-PRES-226125
This work was performed under the auspices of the U.S. Department of Energy by University of California, Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

Performance improvements in speed and accuracy

1. Speed: 23x speed increase in image filtering

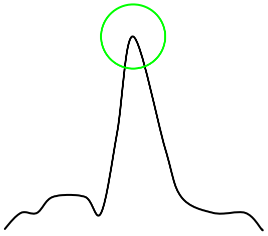


2. Accuracy: Improved estimation of object extent



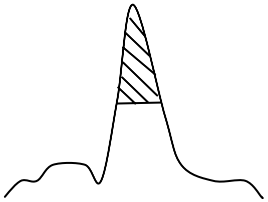
Images processing framework in Matlab

Detection:
Find pixels which are part of defects (seed pixels)



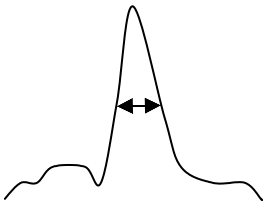
↓

Filling:
Determine extent of defects by considering neighbors of seed pixels (grow seeds)



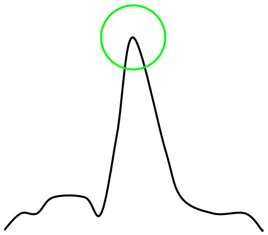
↓

Measurements:
Estimate properties like size of defect



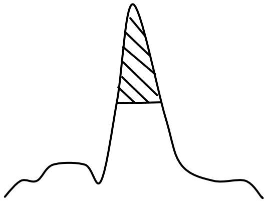
Images processing framework in Matlab

Detection:
Find pixels which are part of defects (seed pixels)



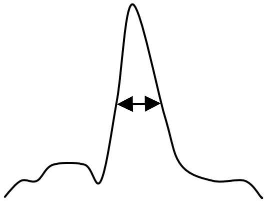
↓

Filling:
Determine extent of defects by considering neighbors of seed pixels (grow seeds)



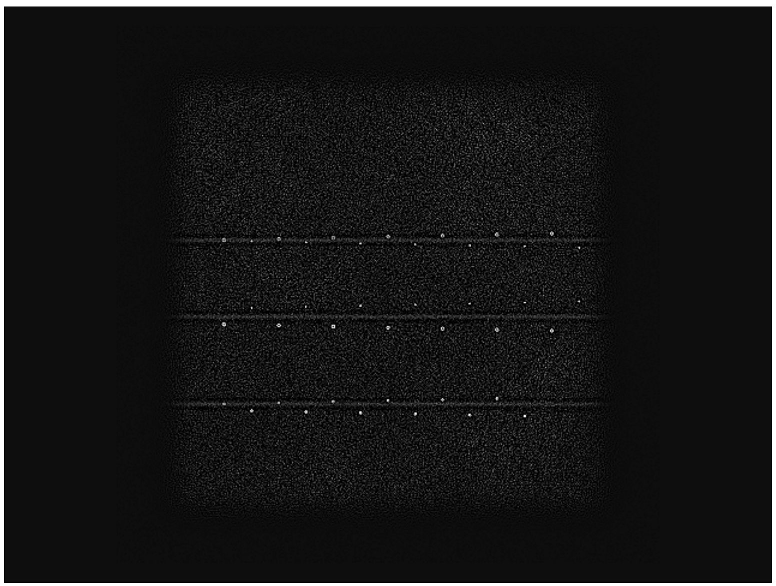
↓

Measurements:
Estimate properties like size of defect



Detection algorithm based on local signal to noise ratio

- Presented at CASIS 2005, 2004
- Defect sites are areas with high local SNR



÷

Signal: Signal at each pixel is estimated by subtracting local background from site

Noise: Local variance at each pixel is estimated averaging (local background)^2

`imfilter` is a bottleneck in detection speed

- Averaging was done using `imfilter` with a separable Gaussian kernel
  - Performed at multiple image scales
- For large images, 98% of detection time is spent on filtering
- `imfilter` has bad cache behavior
  - Applying the horizontal kernel is much slower than vertical kernel (466s vs 13s)
  - Matrices are stored column major in Matlab

Developed and compared `imfilter` alternatives

1. Transpose the image before and after horizontal filtering
2. Use an optimized image processing library
3. Convolve using fast fourier transforms (FFTs)
4. Combine using FFTs with transposing the image

Alternative 1: Transpose for horizontal filtering

- Horizontal filtering becomes vertical filtering
- Pros:
  - Simple to implement

```
out = imfilter(in,h{2},'symmetric','same','conv')';
out = imfilter(out,h{1},'symmetric','same','conv')';
```

- Cons:
  - Not the fastest. About 2x slower than fastest solution

Alternative 2: Use an optimized library

- DIPImage (http://www.ph.tn.tudelft.nl/DIPlib/) is an optimized image processing library for Matlab
- Pros:
  - Simple to implement

```
    out = single(gaussf(in,sigma));
```

  - Fast
- Cons:
  - Dependence on third party library
    - Possible license restrictions
    - No source code

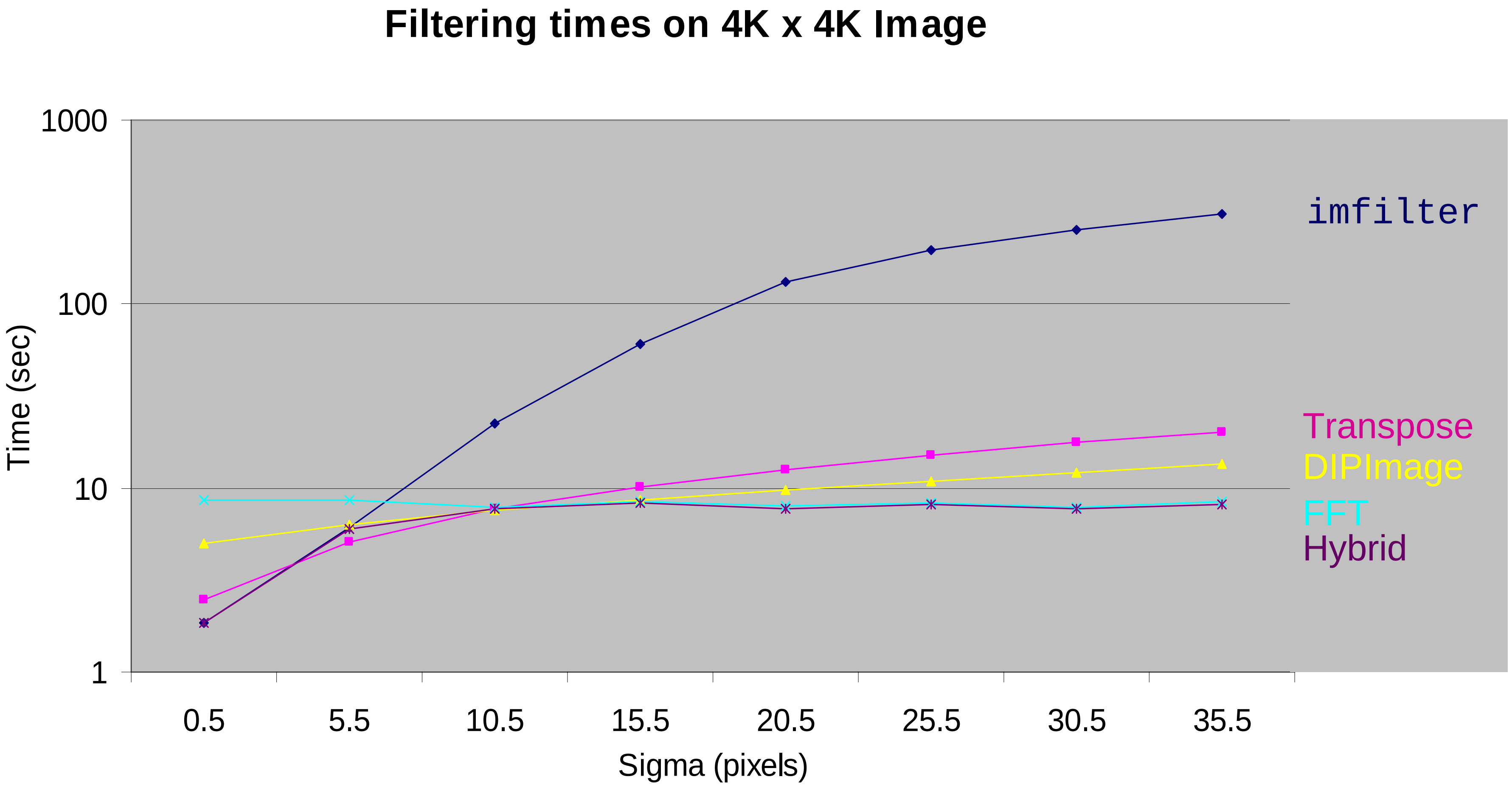
Alternative 3: Implement convolution with FFTs

- Convolution theorem: $x * y \leftrightarrow X \cdot Y$
- For convolution of length L sequence with length M kernel is: $O(L*M)$ in spatial domain
- $O((L+M)\log(L+M))$ with FFT
  - Win for long kernels
- FFTs in Matlab are highly optimized
- Pros:
  - Fast, runtime is almost independent of kernel length when kernel size << image size
- Cons:
  - Relatively complex to implement
  - Slower for small kernels and images

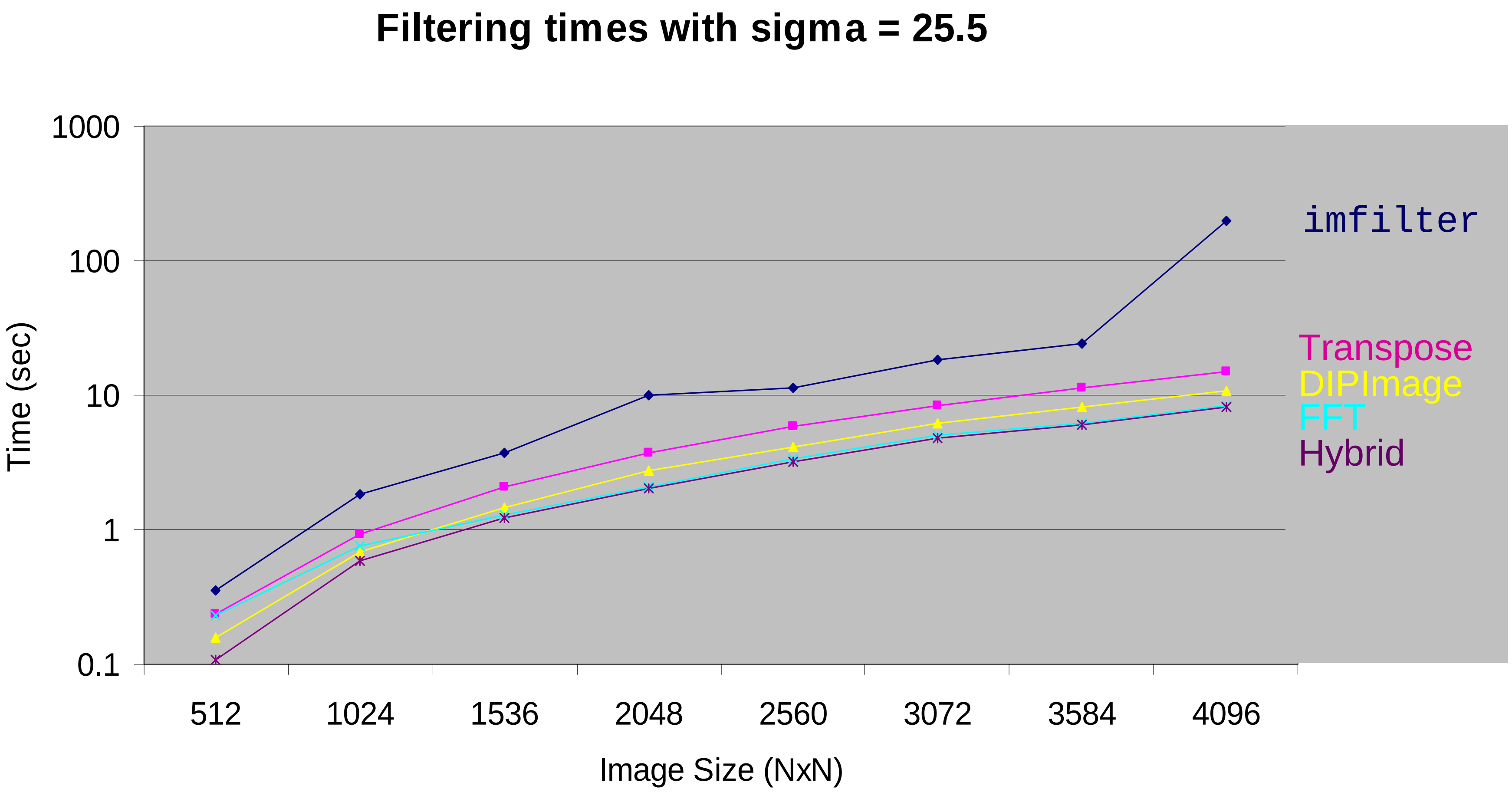
Alternative 4: Hybrid: Combine FFTs and transposing

- Use transposing technique for smaller images and kernels
  - Avoids overhead of FFT
- Pros:
  - Fast for all image and kernel sizes
- Cons:
  - Even more complex than just FFTs. Additional logic needed to select FFT vs transposing

Hybrid method is fastest for most kernel sizes

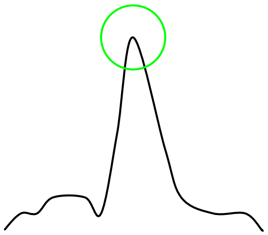


Hybrid method is fastest for all image sizes



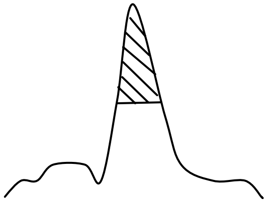
Images processing framework in Matlab

Detection:
Find pixels which are part of defects (seed pixels)



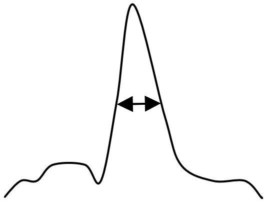
↓

Filling:
Determine extent of defects by considering neighbors of seed pixels (grow seeds)



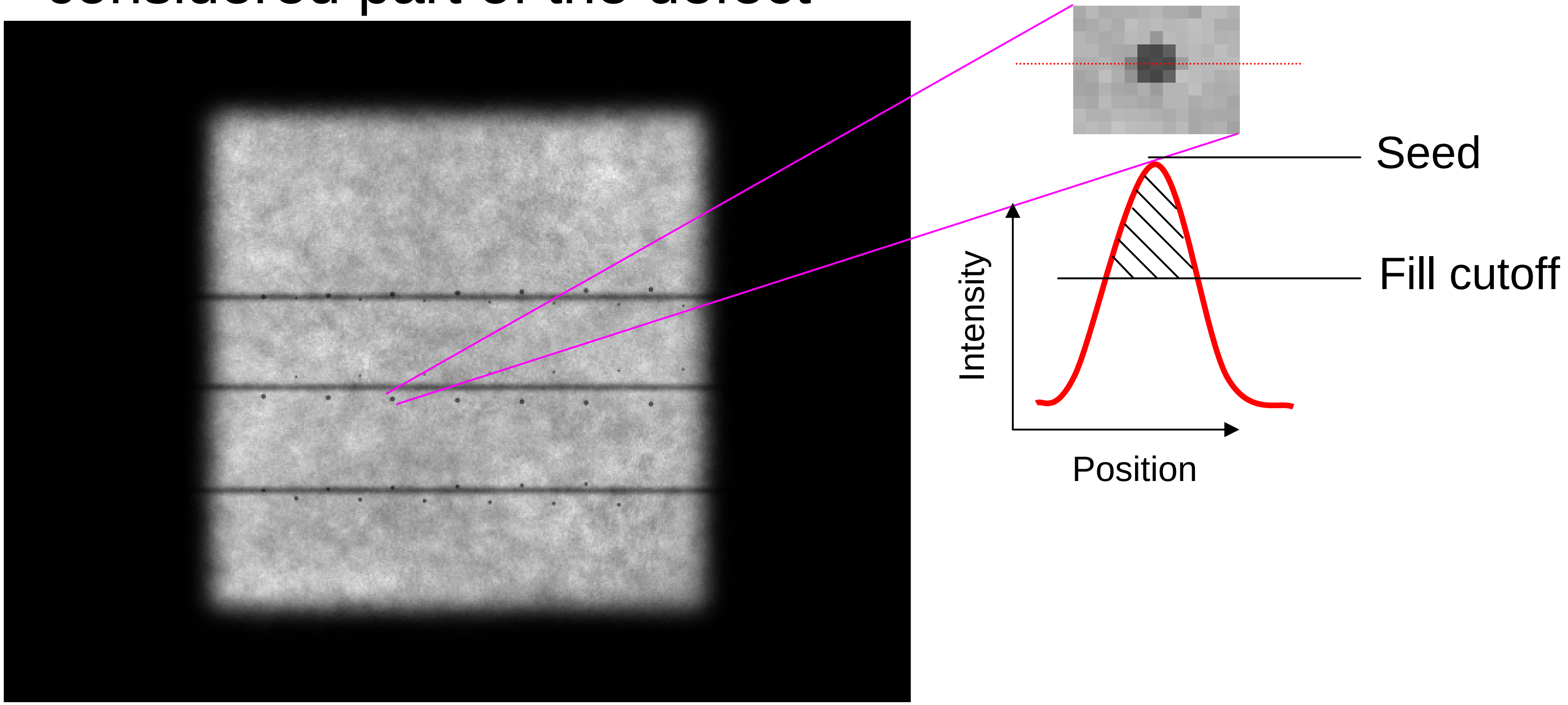
↓

Measurements:
Estimate properties like size of defect



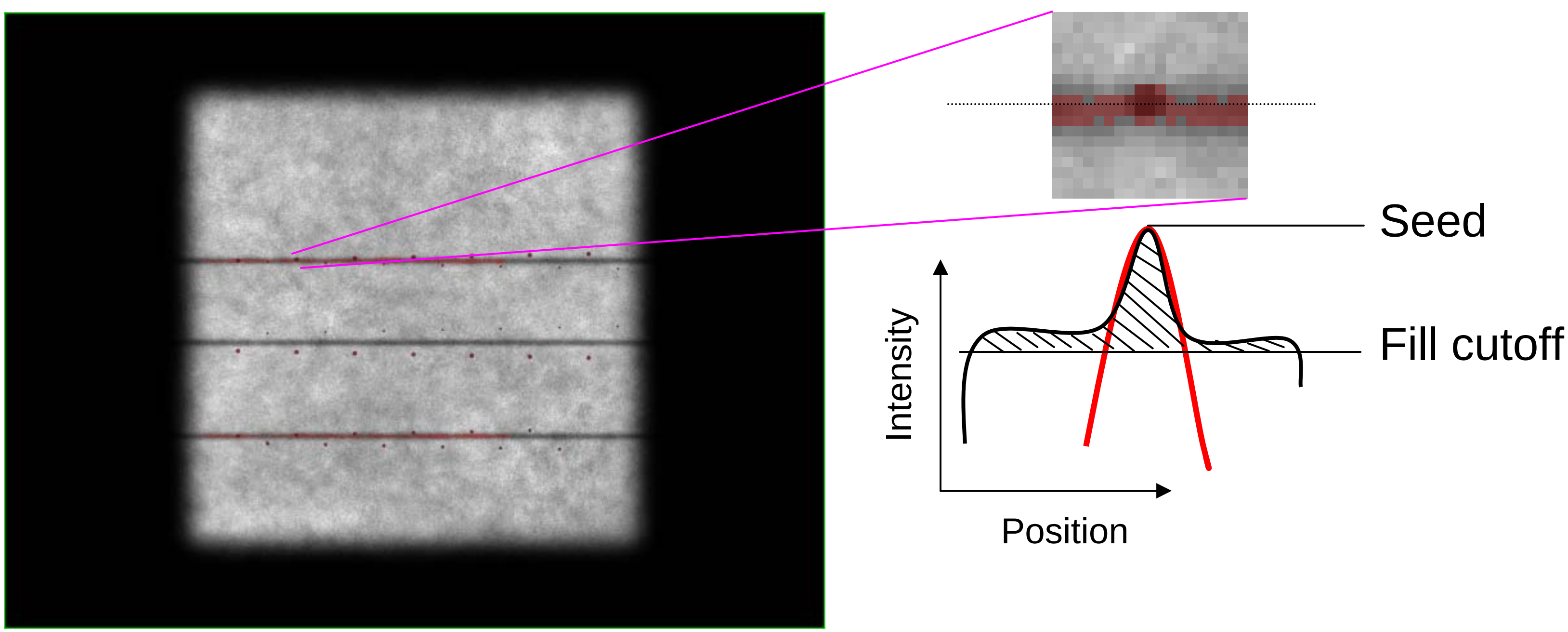
Filling determines a defect's extent

- Detection phase finds seed pixels in peaks
- Neighboring pixels with intensities above a pre-determined fraction of the seed pixel intensity are considered part of the defect



Previous fixed cutoff method can overfill

- When a defect is on a background feature with elevated intensity, non-defect pixels will be incorrectly labeled as defect pixels

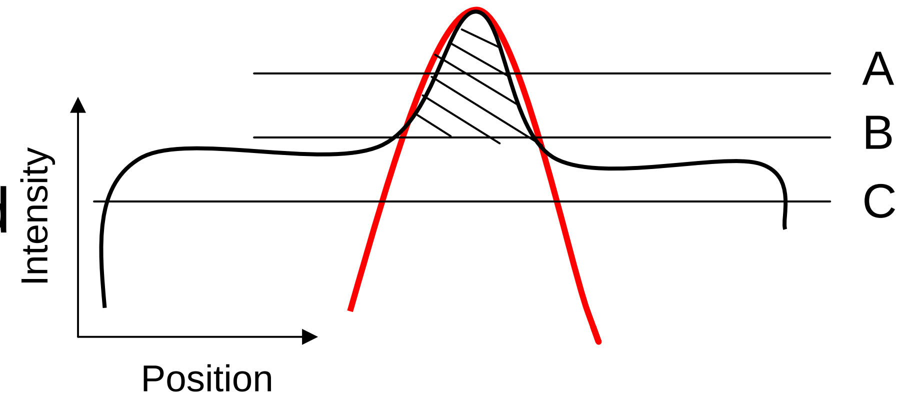


Determine the cutoff adaptively

- Fill pixels in decreasing order of intensity

- Track number of pixels filled over a sliding window of fraction of seed pixel intensity

- Stop when ratio of number of pixels filled in current window to number of pixels in previous window exceeds a threshold

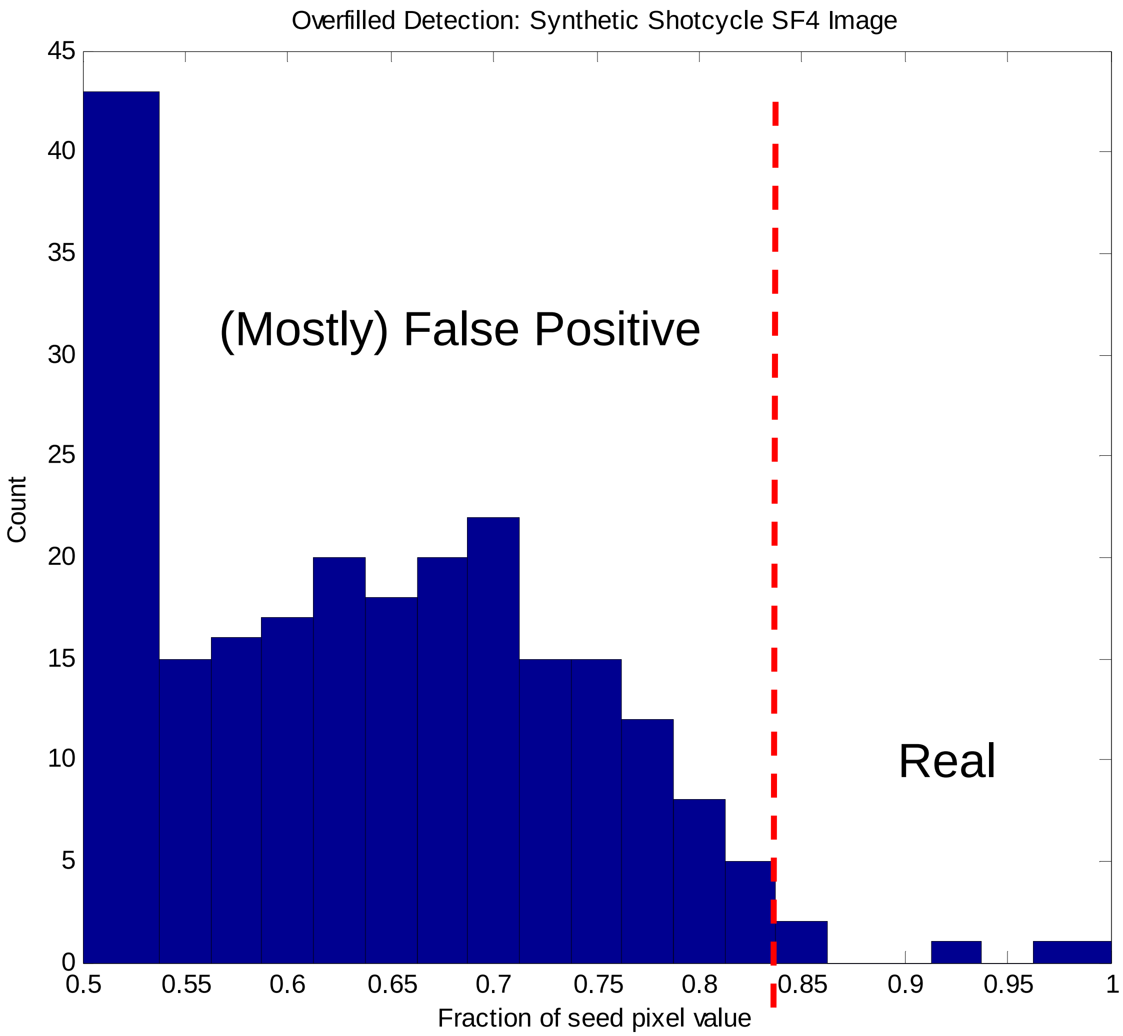


Number of pixels between A and B is much smaller than between B and C

Stop at B

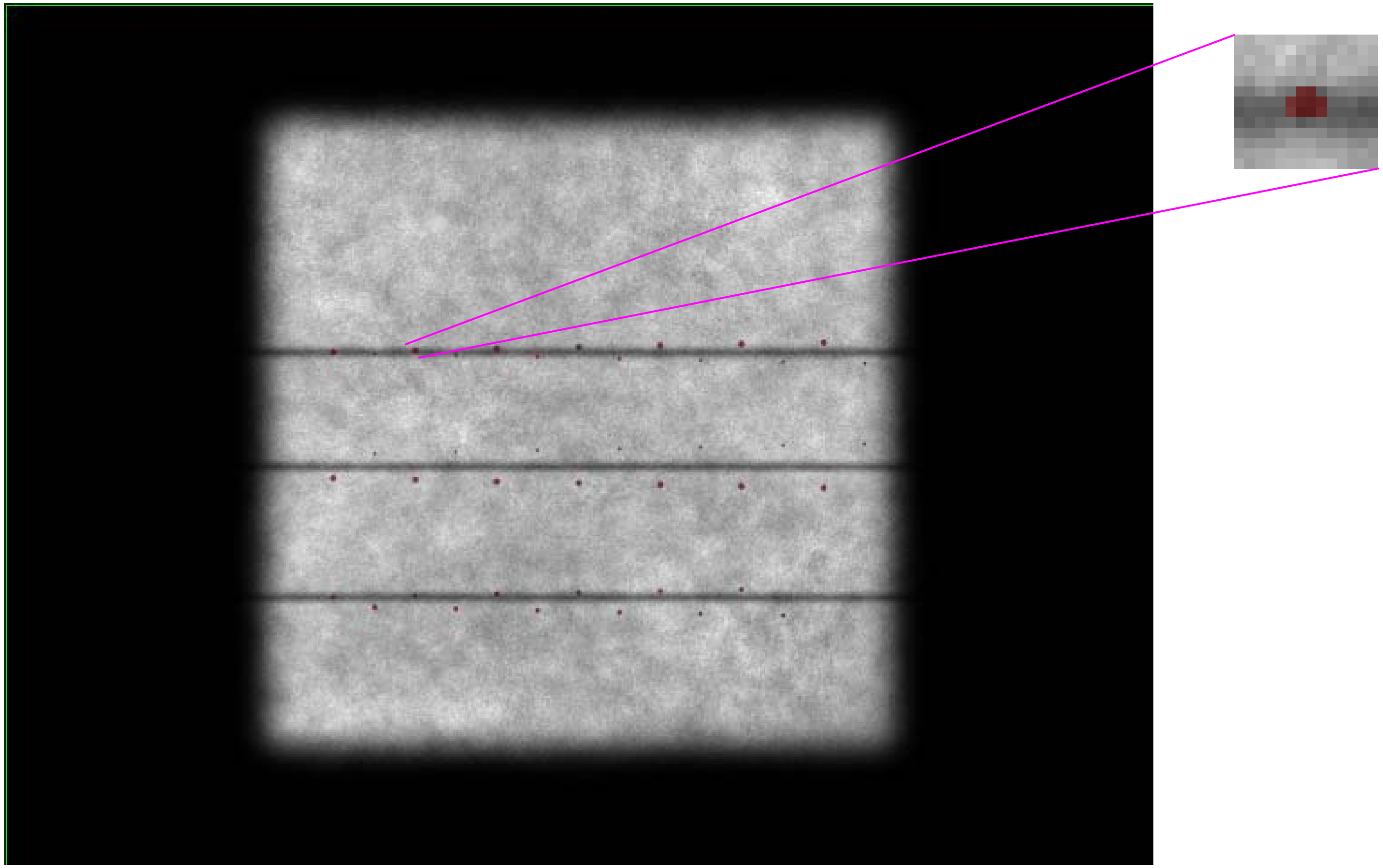
False positives increase quickly as cutoff is lowered

- Number of pixels per bin increases for false positives



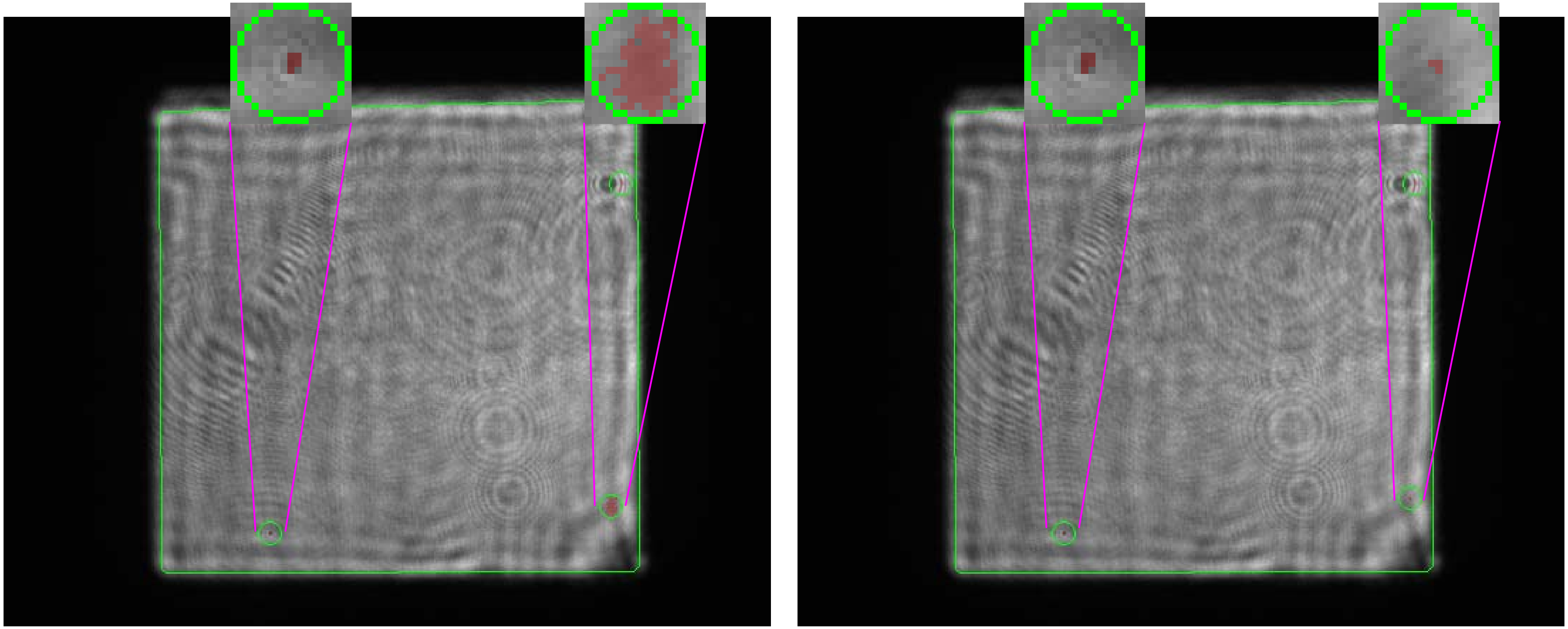
Adaptive filling almost eliminates false positives on synthetic image

- Simulated image of defects varying distances from reflectivity lines



Adaptive filling reduces false positives on real images

- Real detections are unchanged
- Fewer pixels are assigned to a false detection



Fixed cutoff SHOT_N050712-001-001B Adaptive cutoff

New algorithms improve speed and accuracy of NIF Optics Inspection

- Image processing time for NIF Final Optics is cut in half
- Speed and accuracy improved by reducing false positives by order of magnitude in many cases