CSCI 4974 / 6974
Hardware Reverse Engineering

Lecture 20: Automated RE / Machine Vision

Today's lecture

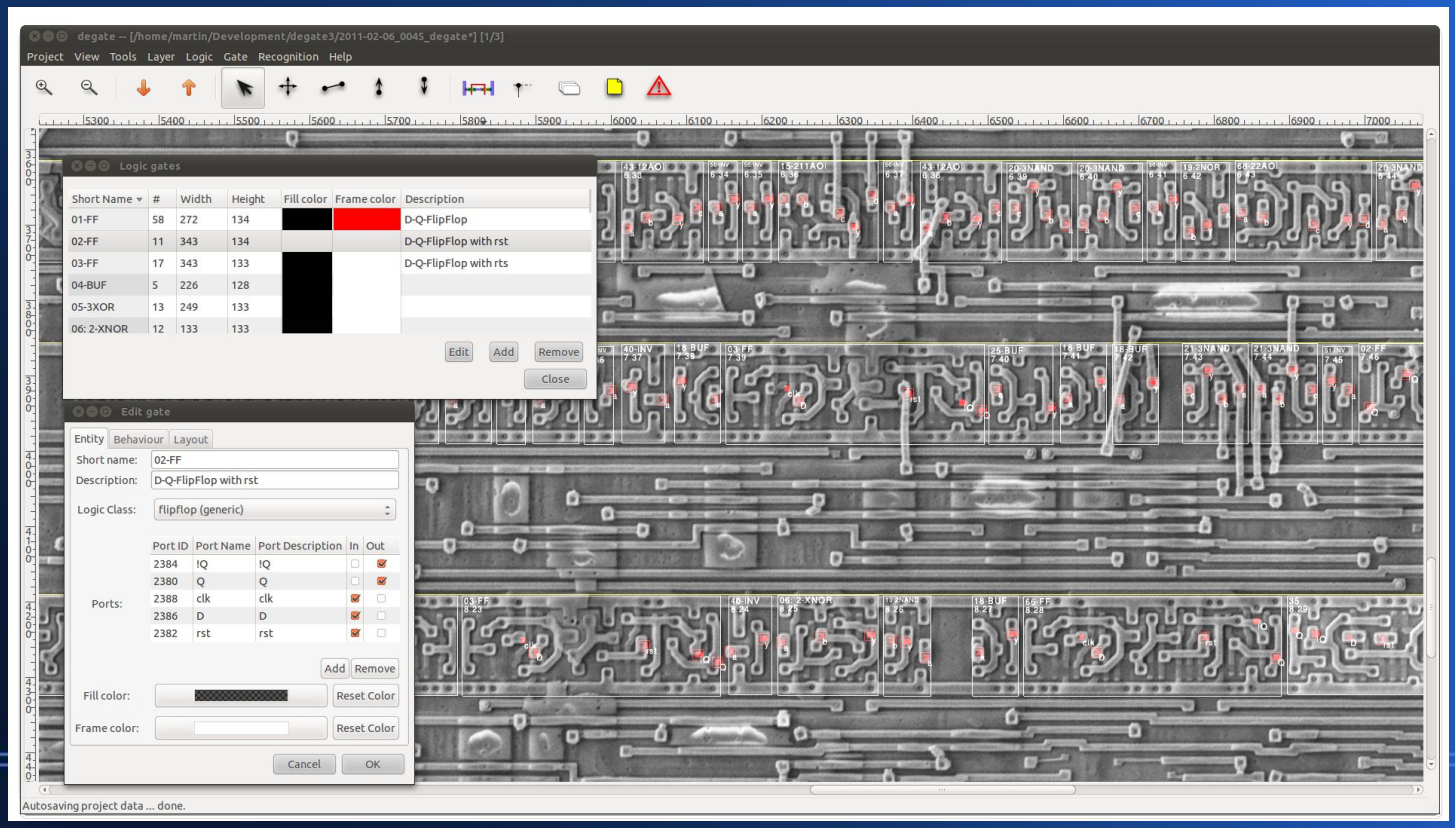
- Last lecture of the semester!
- Existing tools for reverse engineering of ICs
  - Sadly, not too many of these exist :(
- Introduction to machine vision
  - Writing your own tools is the way to go for now

Existing tools

- degate (automatic standard cell recognition)
- rompar (mask ROM extraction)

Degate

- http://www.degate.org/
- Semi-automated standard cell RE



Degate

- Recognizes copies of standard cells given a manually-identified master image.
  - Requires textbook standard cell design: one transistor layer, cells on M1, routing on M2+
  - Was not useful on the CPLD because there's a lot of custom logic with poly-level routing etc
- Can trace wires automatically from images
  - I have so far not been able to get this to work :(
  - Manual tracing isn't much better than Inkscape

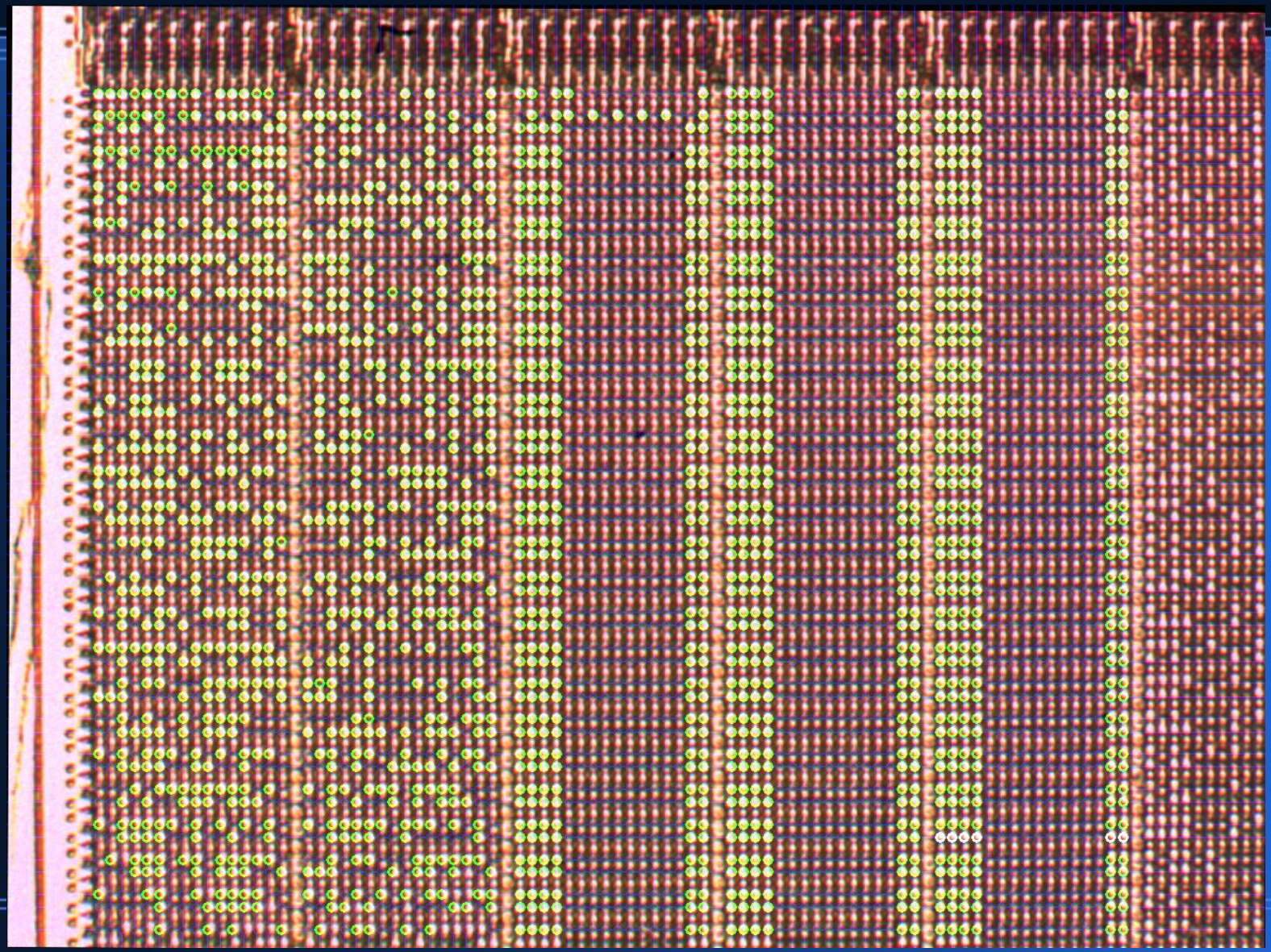
Degate

- Can export Verilog or XML netlists for analysis
- Has stability issues
  - Last time I tried a year or so ago it was completely unusable
  - The latest Git build seems to segfault less

Rompar

- https://github.com/ApertureLabsLtd/rompar
- Semi-automatic decoding of mask ROM
  - Seems to be designed for via type only
- User draws grid and selects threshold
- Tool IDs bright/dark spots

Rompar



General workflow

- Semi-automatic analysis
  - Images are noisy, may have particles, etc
  - Fully automated analysis is hard to impossible
  - Computer is guided by, but not replacing, the human engineer

So what is machine vision?

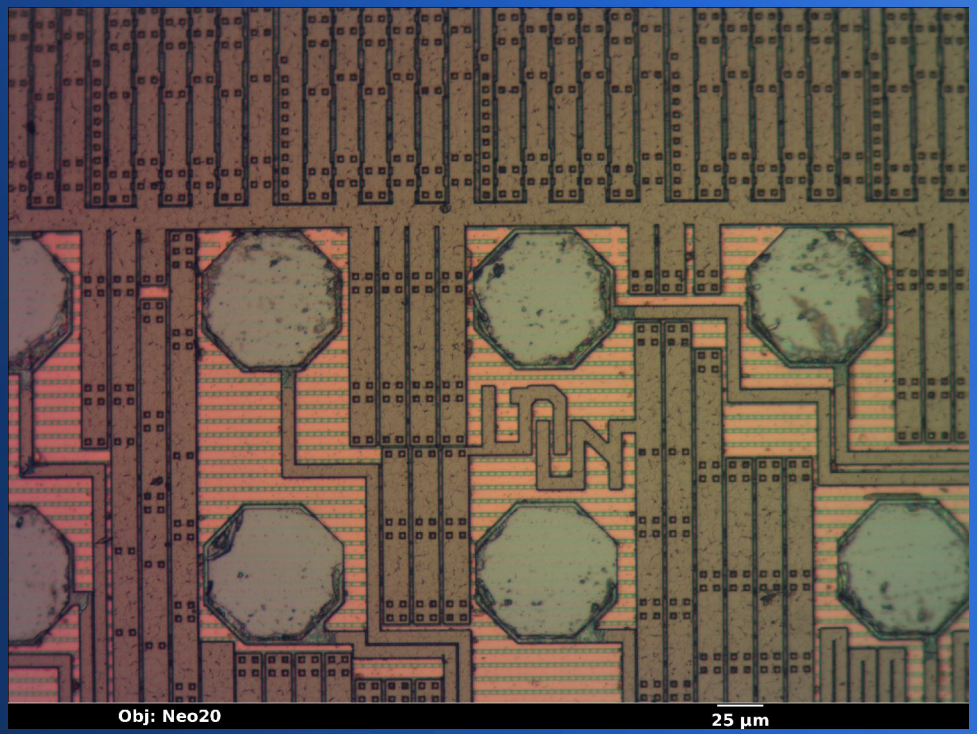
- Treat an N-d image as a function of N inputs
- Function may be vector or scalar valued
- Try to find some structure in the function that maps to a useful structure in the real world
- Today's lecture will focus on 2D scalars
  - 2D grayscale images
  - But much of the material generalizes well

Two classes of vision operations

- Pixel filters
  - Create a new image from one or more inputs
  - Each pixel is a function of the corresponding pixel in the input(s) and possibly its neighbors
  - Say hello to our friend multivariable calculus ;)
- Higher level analysis
  - Make lists of interesting features in filtered images (lines, corners, dots, etc)
  - Further processing on high-level datasets

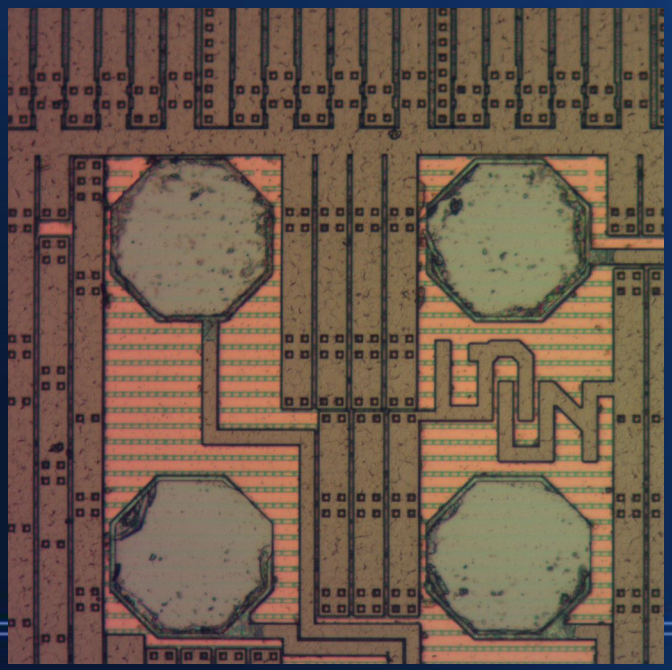
Our example image

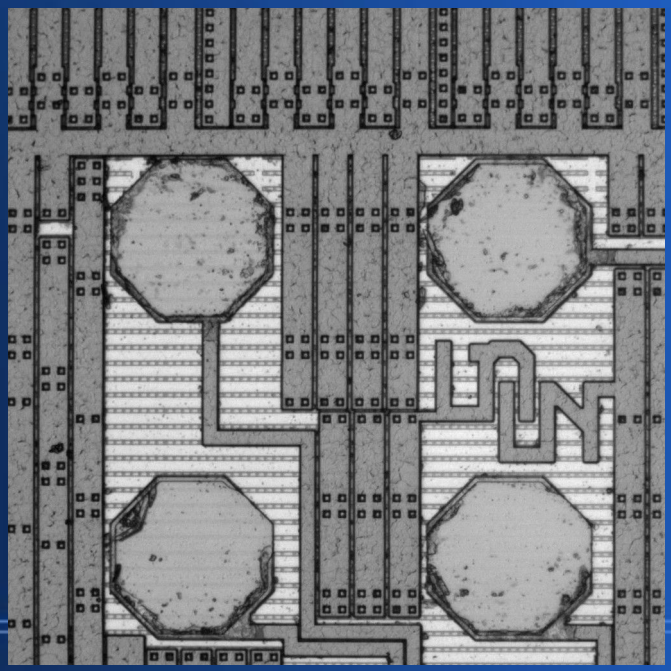
- Xilinx XC3S50A (Spartan-3A, 90 nm)
- 200x optical image of bond pads and M8 power routing



Simple filter: RGB to grayscale

- gray[x][y] = F(red[x][y], green[x][y], blue[x][y])
- Typically weight green higher (ex: NTSC)
  - Y = 0.299R + 0.587G + 0.114B





Discrete convolution

- Very common operator for processing pixels in a neighborhood around a point
- Analogous to continous-domain convolution
- Pairwise multiply filter kernel, centered at each pixel, with image, then sum results
- c[x] = f[0]i[x-1] + f[1]i[x] + f[2]i[x+1] etc

1D derivatives

- The derivative of an image along one axis can be represented as $\partial I/\partial x$ or $\partial I/\partial y$.
- Highlights changing intensity
- In the discrete domain, the simplest case comes out to pairwise subtraction!
  - Convolve with [-1 0 1]
  - dx[y][x] = img[y][x+1] - img[y][x-1]
  - Naive [-1 1] will cause image shift

Representing derivatives

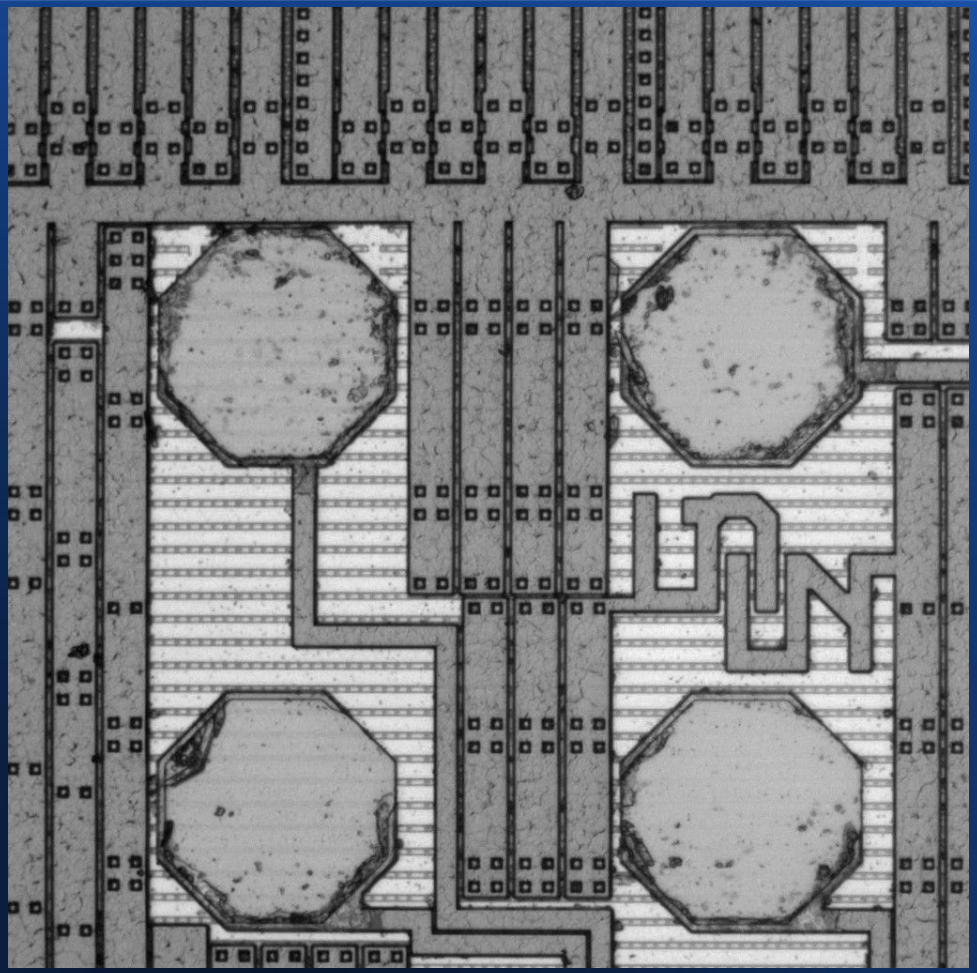
- Grayscale images are unsigned but gradients are signed (value can range from -255 to +255)
- Common mapping for display is (X/2) + 128
  - Dark = negative
  - Medium gray = 0
  - Light = positive

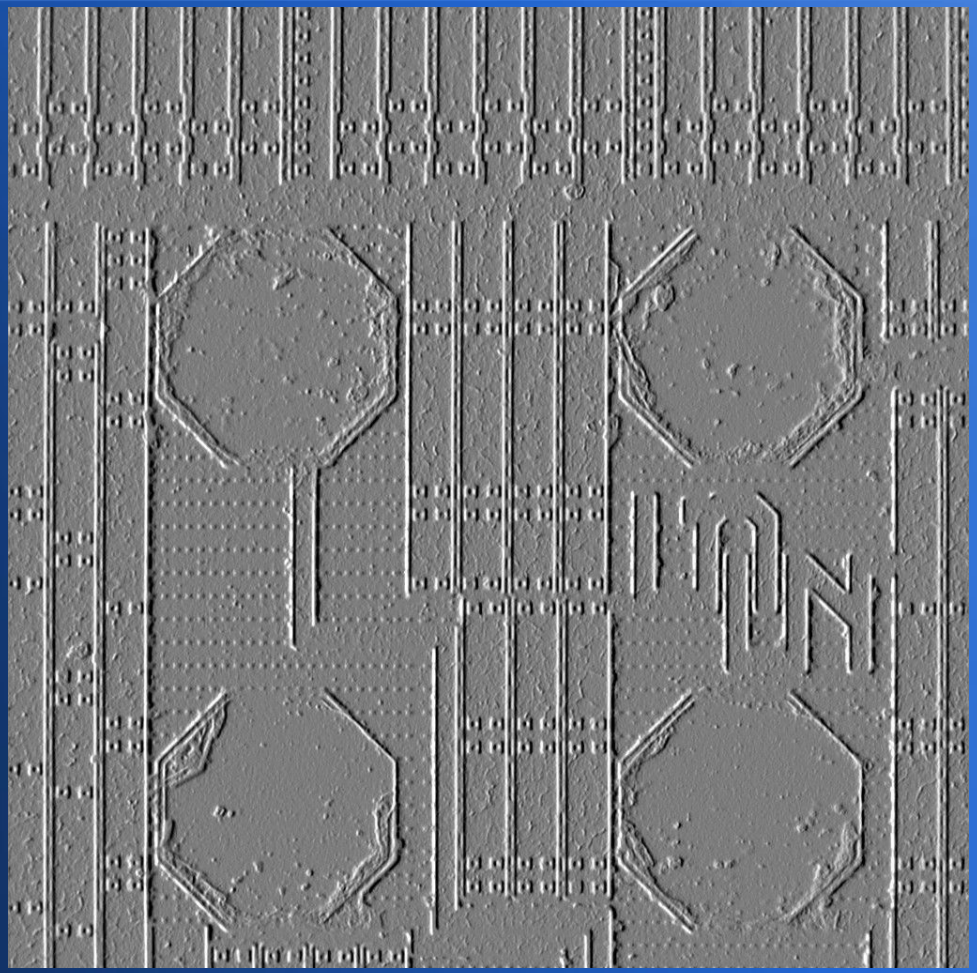
More complex gradients

- Most common is the Sobel filter
- Allows a single 3x3 neighborhood to be used for both X and Y gradients
- Applies a small amount of perpendicular smoothing

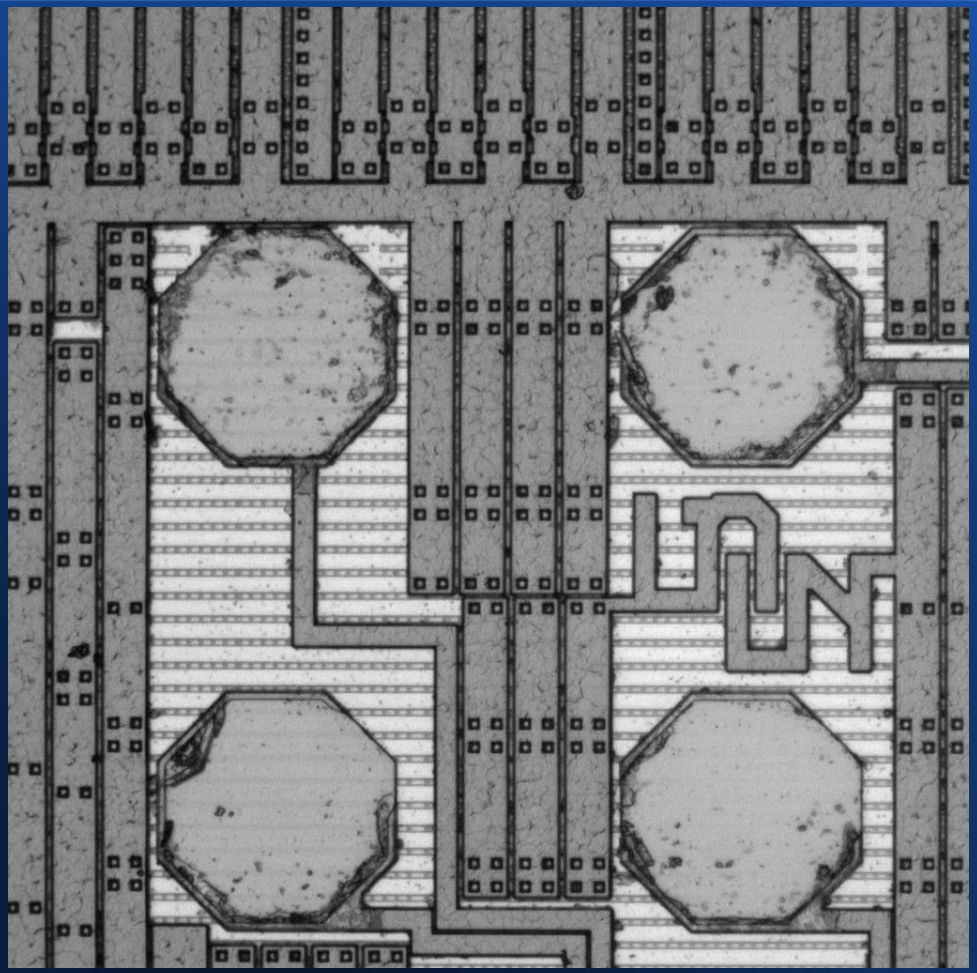
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

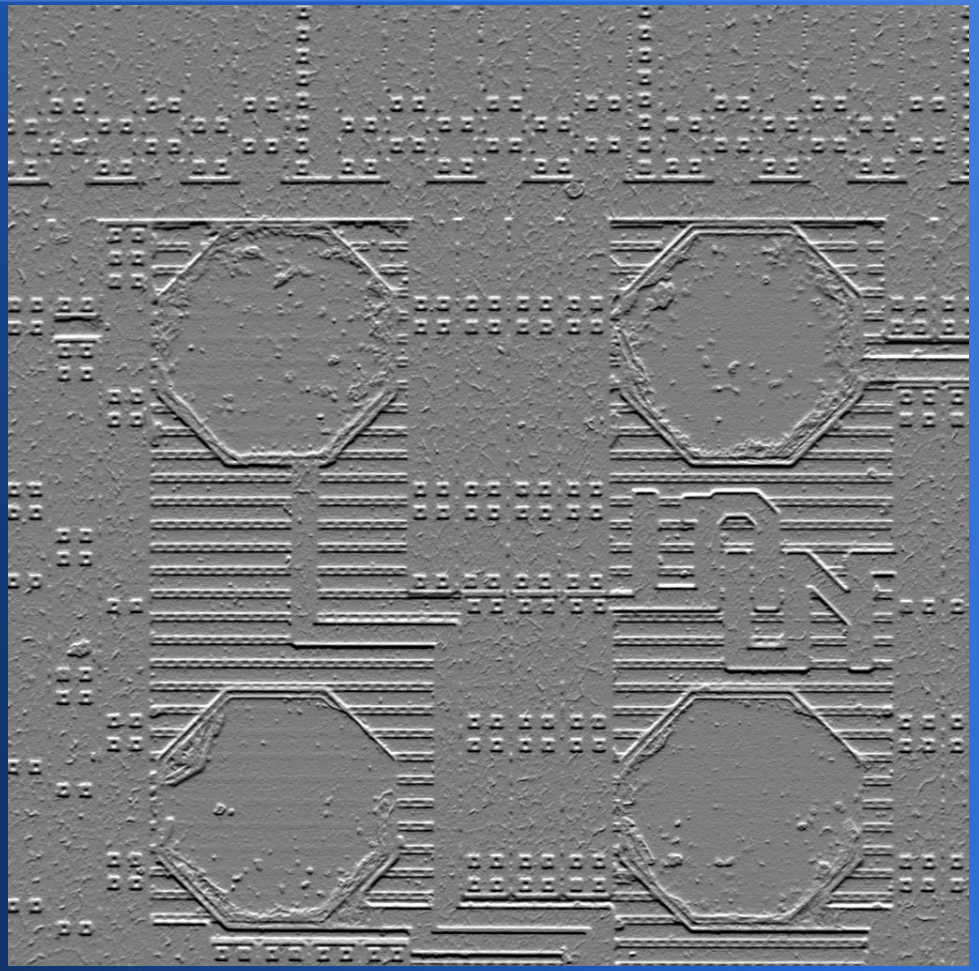
Sobel in X





Sobel in Y

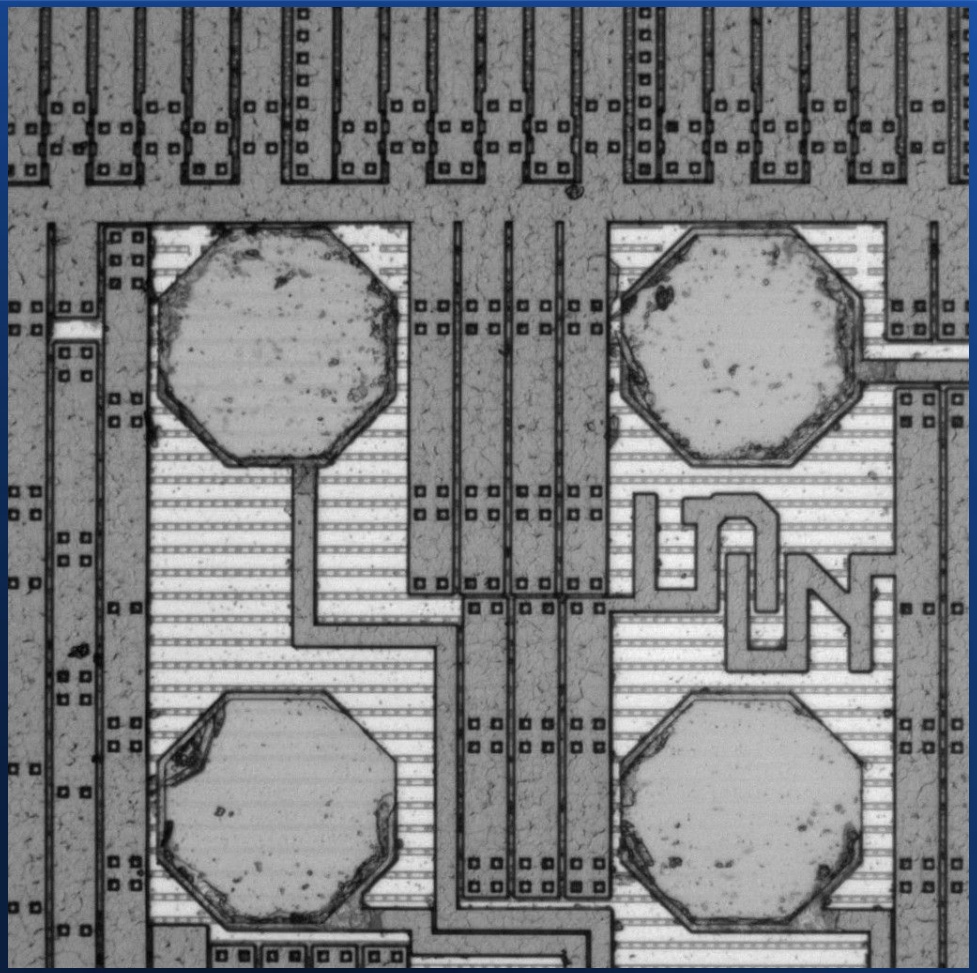


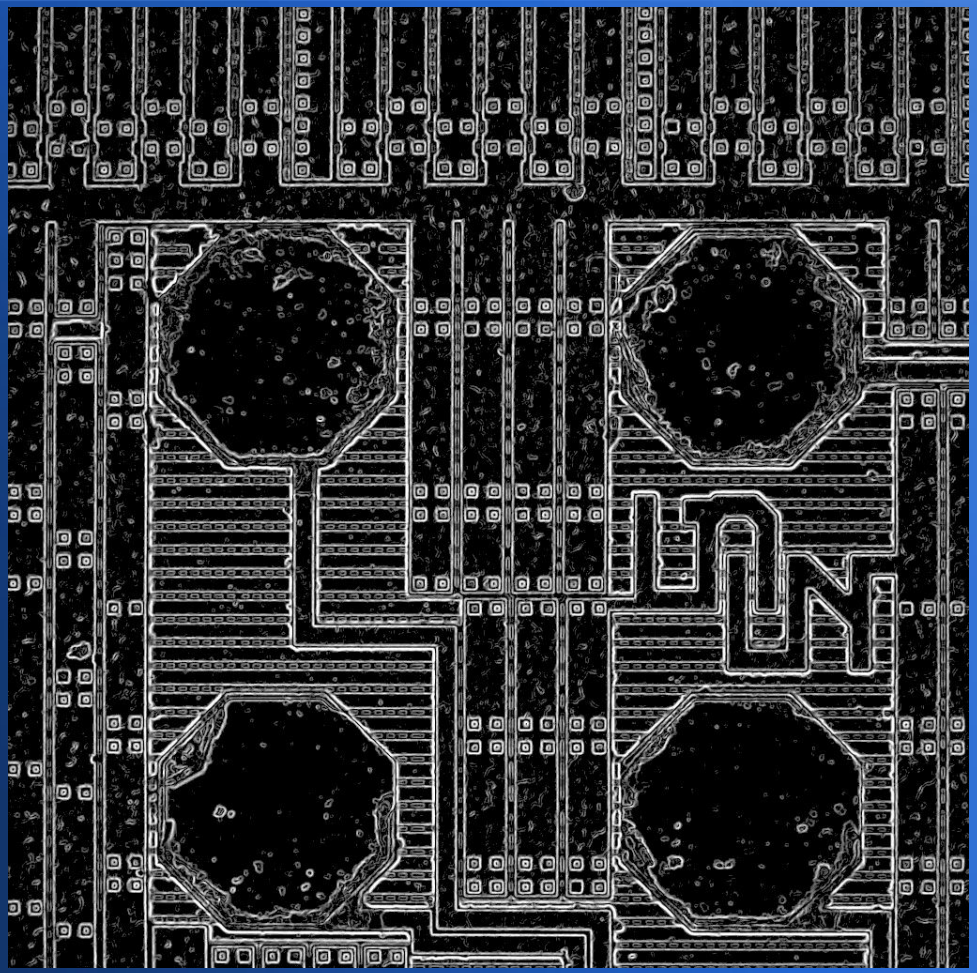


Gradient magnitude

- Compute X/Y gradients using method of choice
- Treat gradient for each pixel as a 2-vector
- Compute magnitude as usual
- $grad[x][y] = sqrt(dx[x][y]^2 + dy[x][y]^2)$

Sobel gradient magnitude

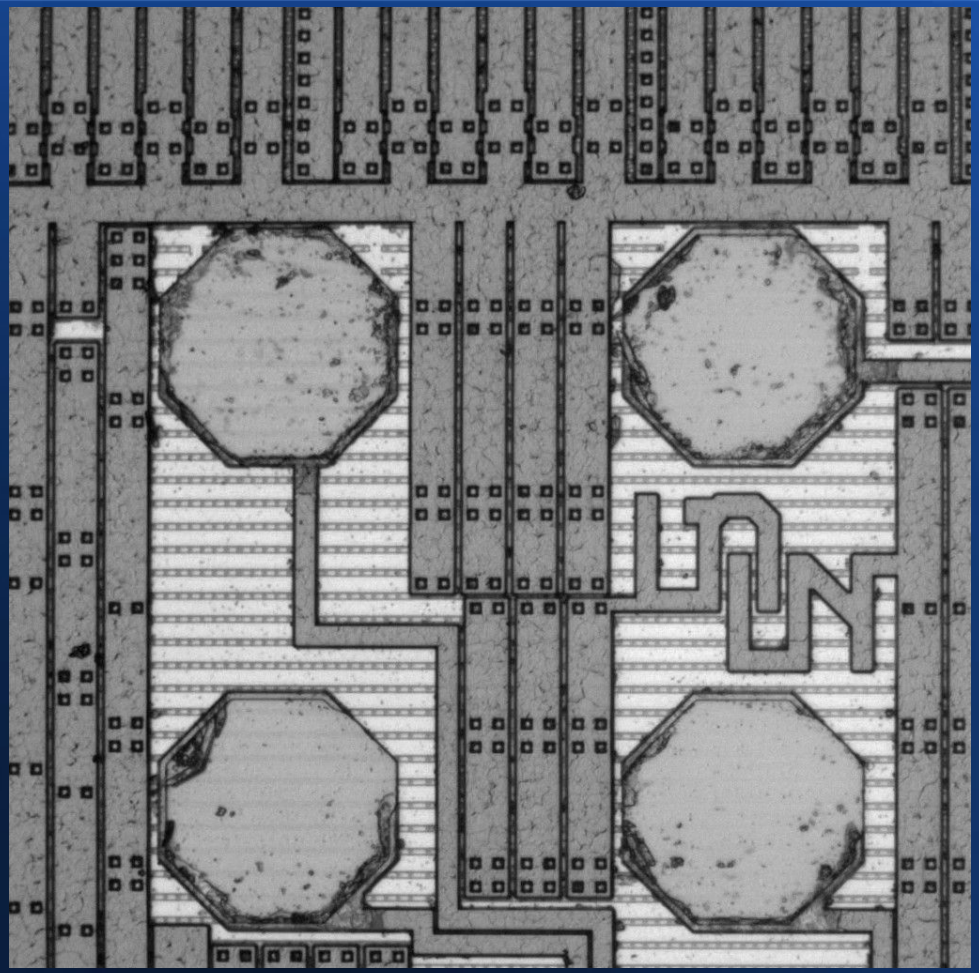


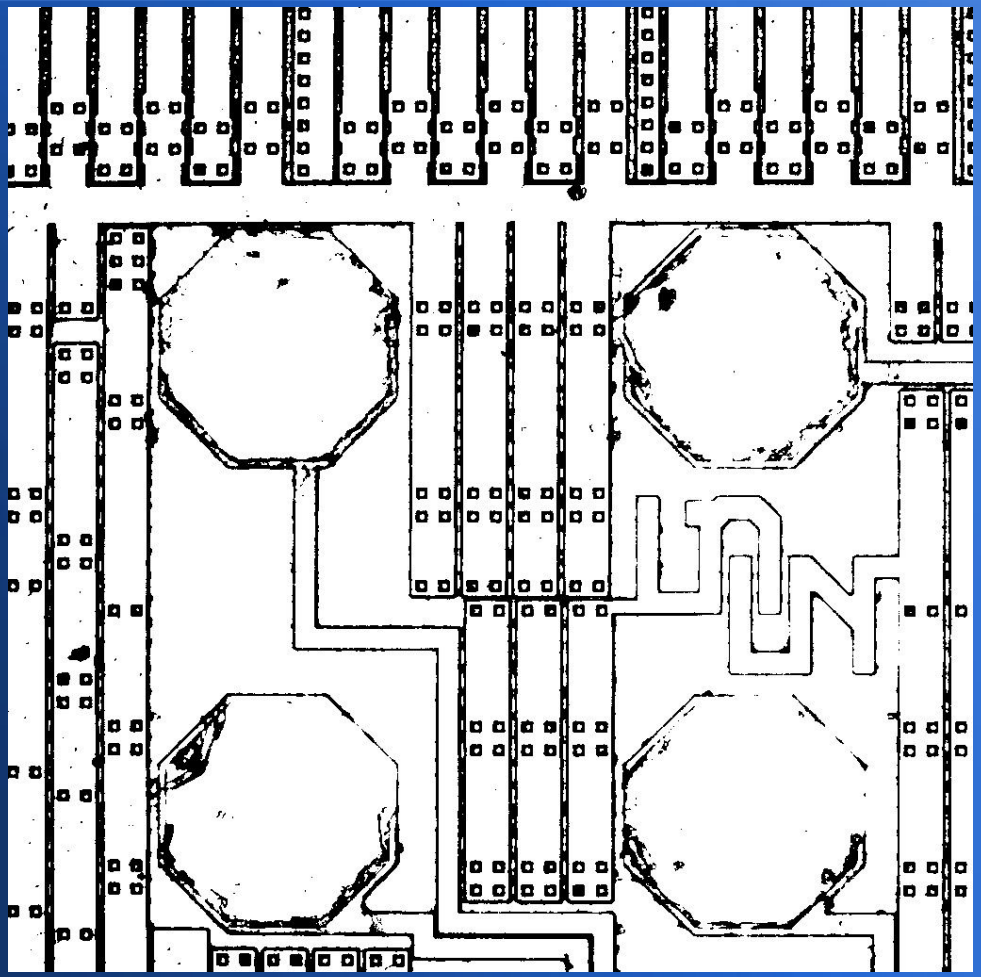


Thresholding

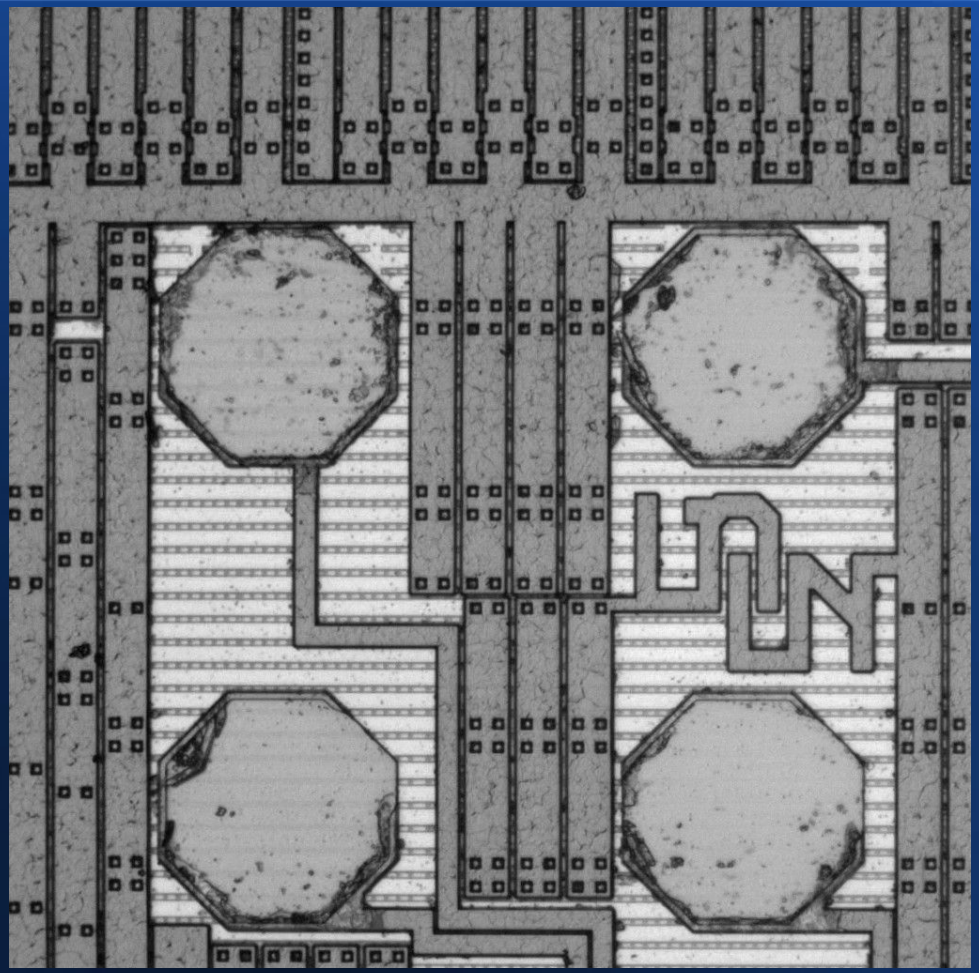
- Turn a grayscale image into a Boolean image
- thresh[x][y] = gray[x][y] > T ? 1 : 0
- How to choose constant T?
- Use same T globally or vary across image?

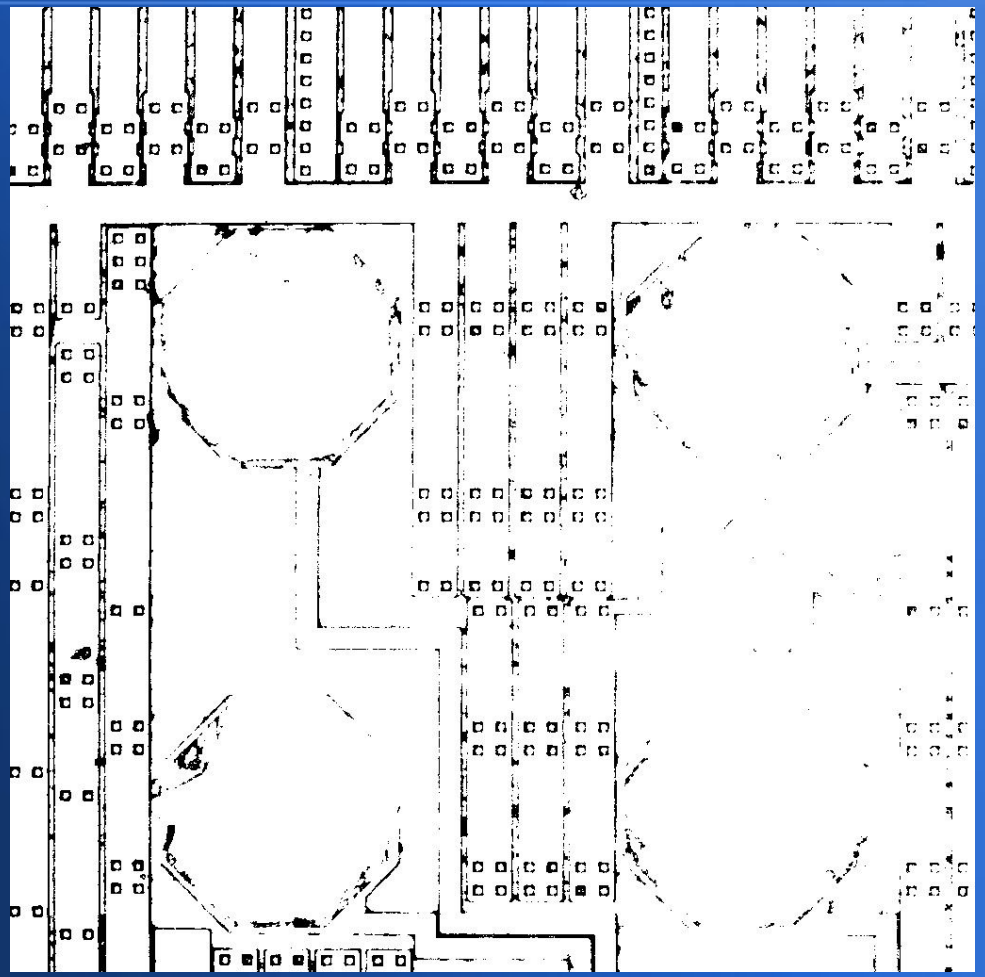
Constant thresholding



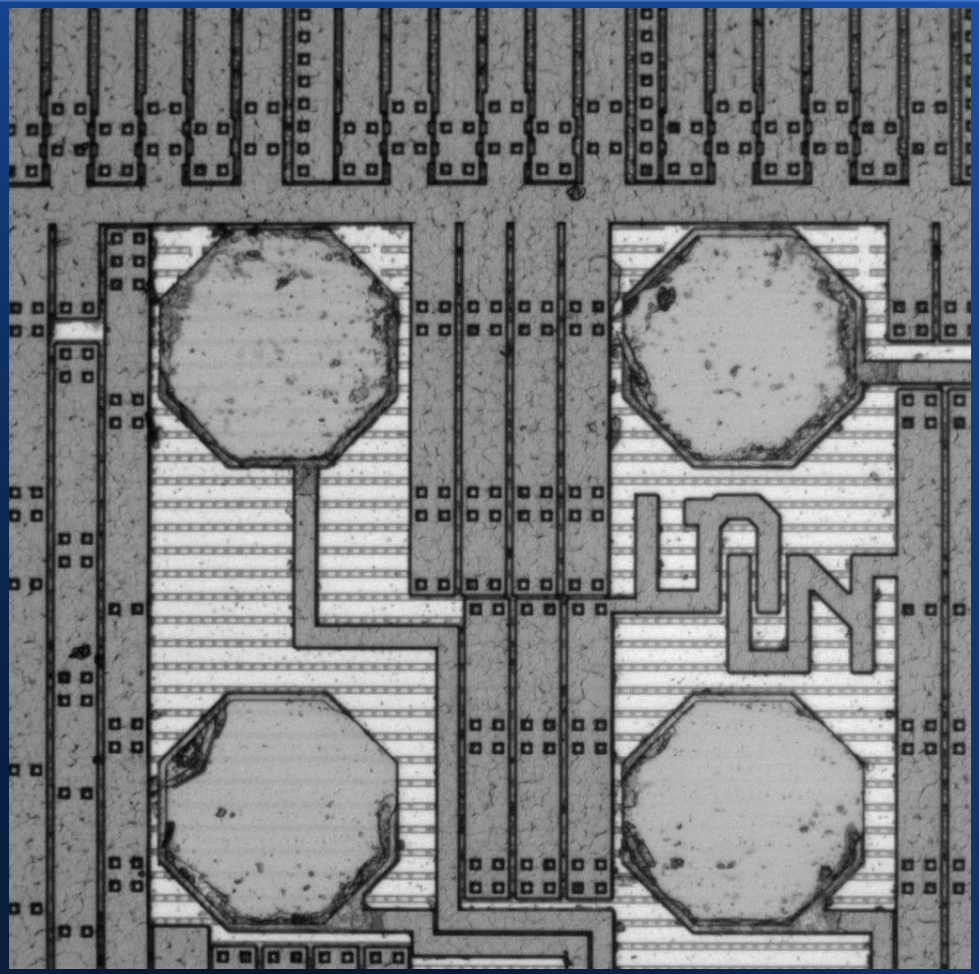


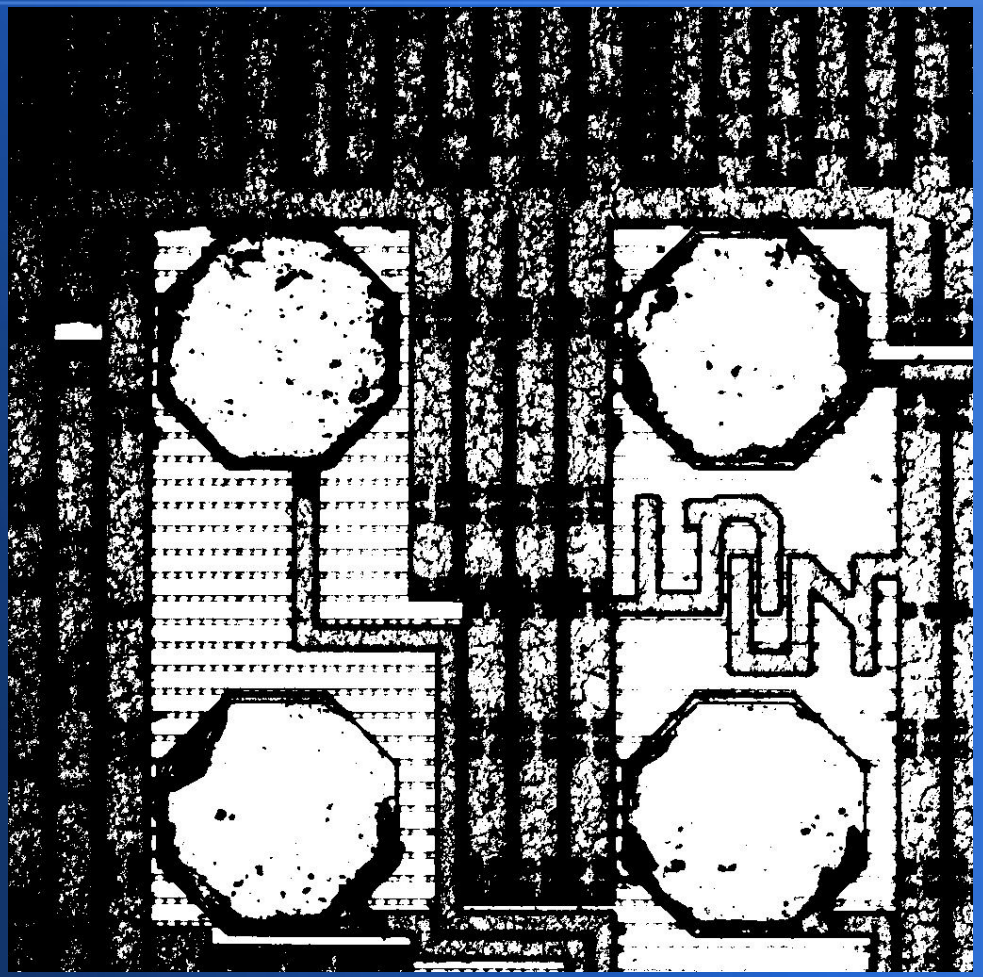
Threshold too low





Threshold too high





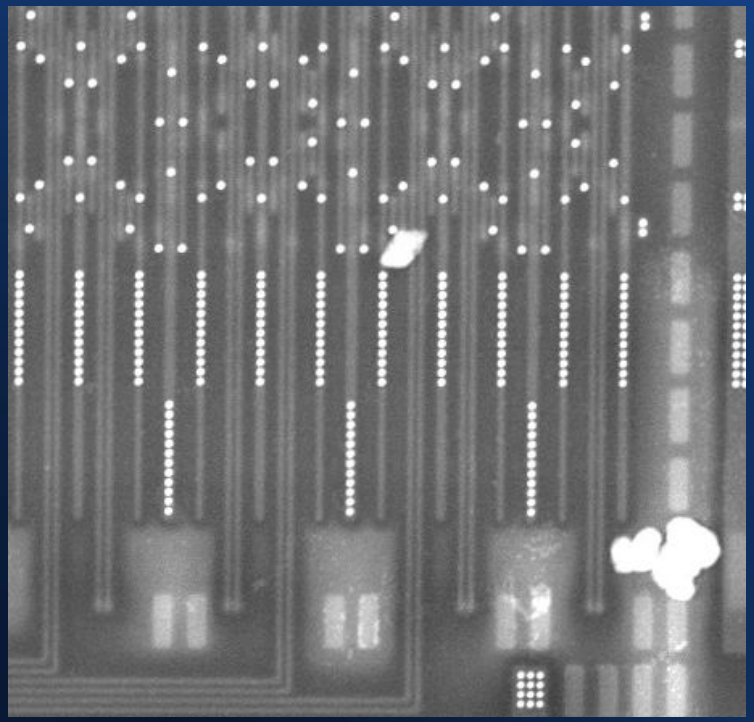
RATS

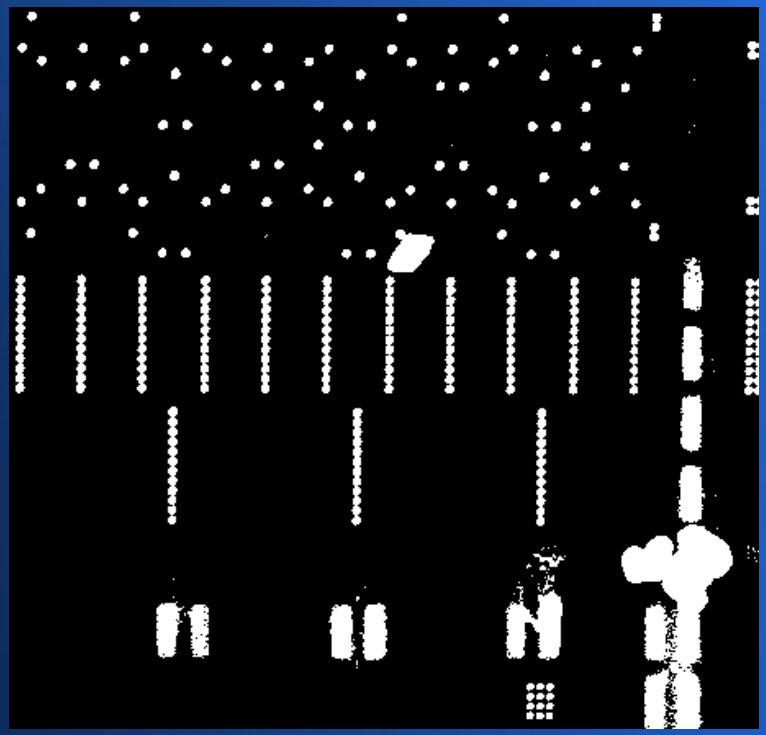
- Robust Automated Threshold Selection
- One of many algorithms for choosing threshold
- Set threshold at the highest gradient in the image, weighted by the image area
- Heuristic: Sharply changing regions of the image are probably important edges we want to preserve

$$T = \frac{\sum_{p \text{ in } D} G(p) \cdot I(p)}{\sum_{p \text{ in } D} G(p)}$$

Thresholding failures

- What if brightness alone isn't enough to identify our target feature?

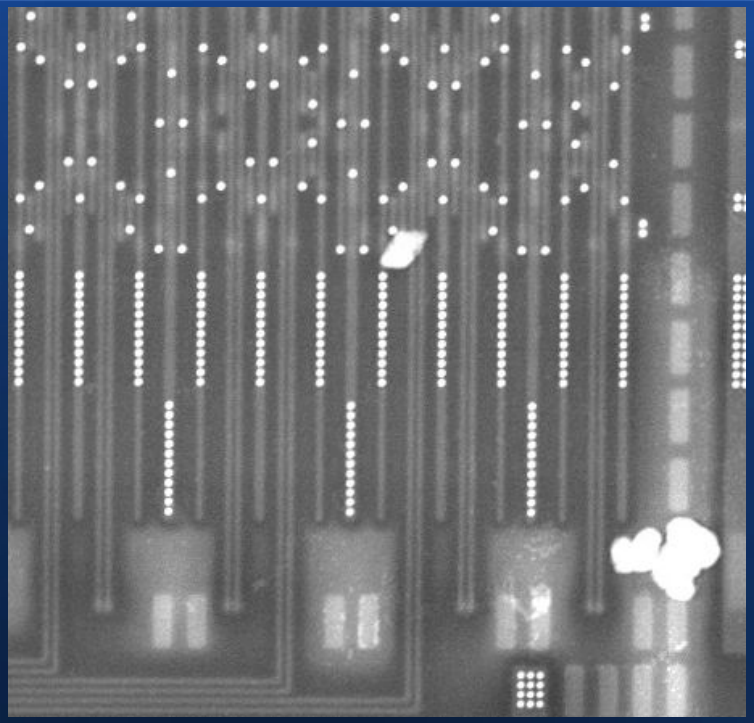


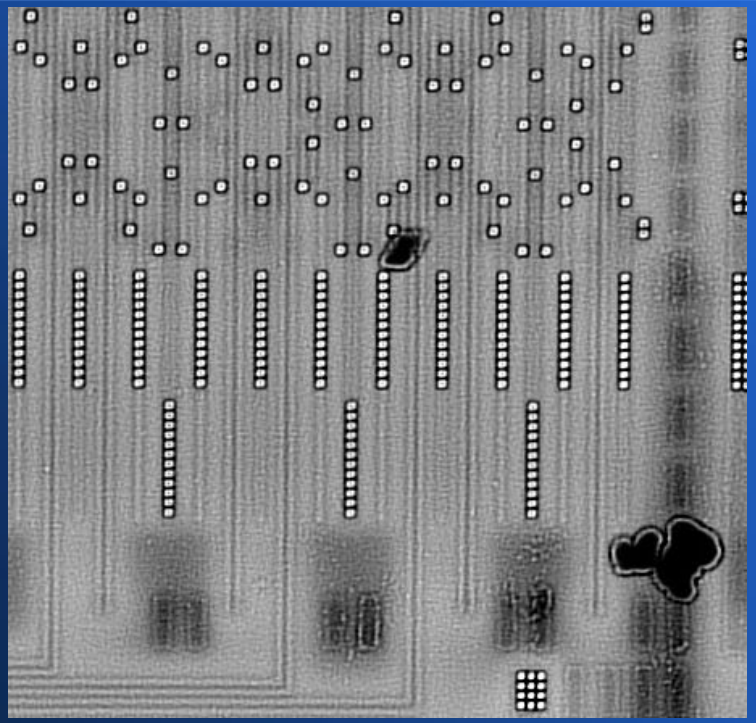


Cross-correlation filter

- Works well for detecting features of known size/shape
- Convolve image with a mask shaped like the feature (high values to match, low values to discard)

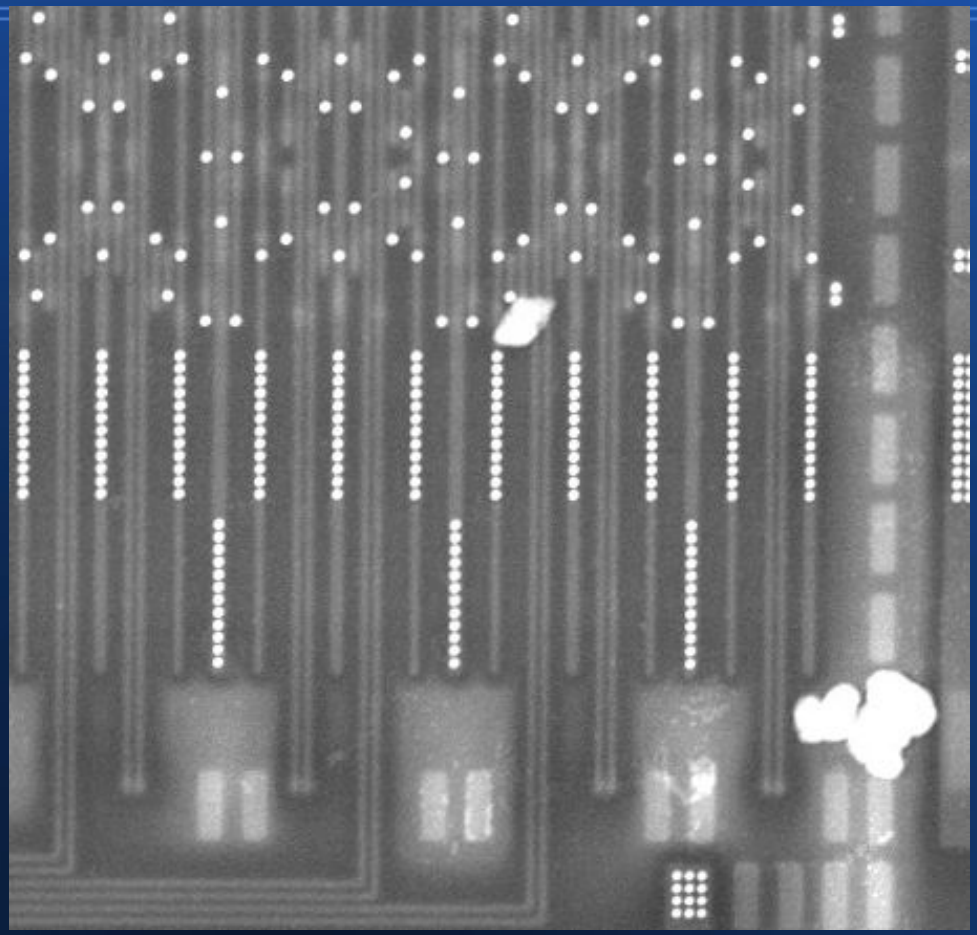
Via-finding filter

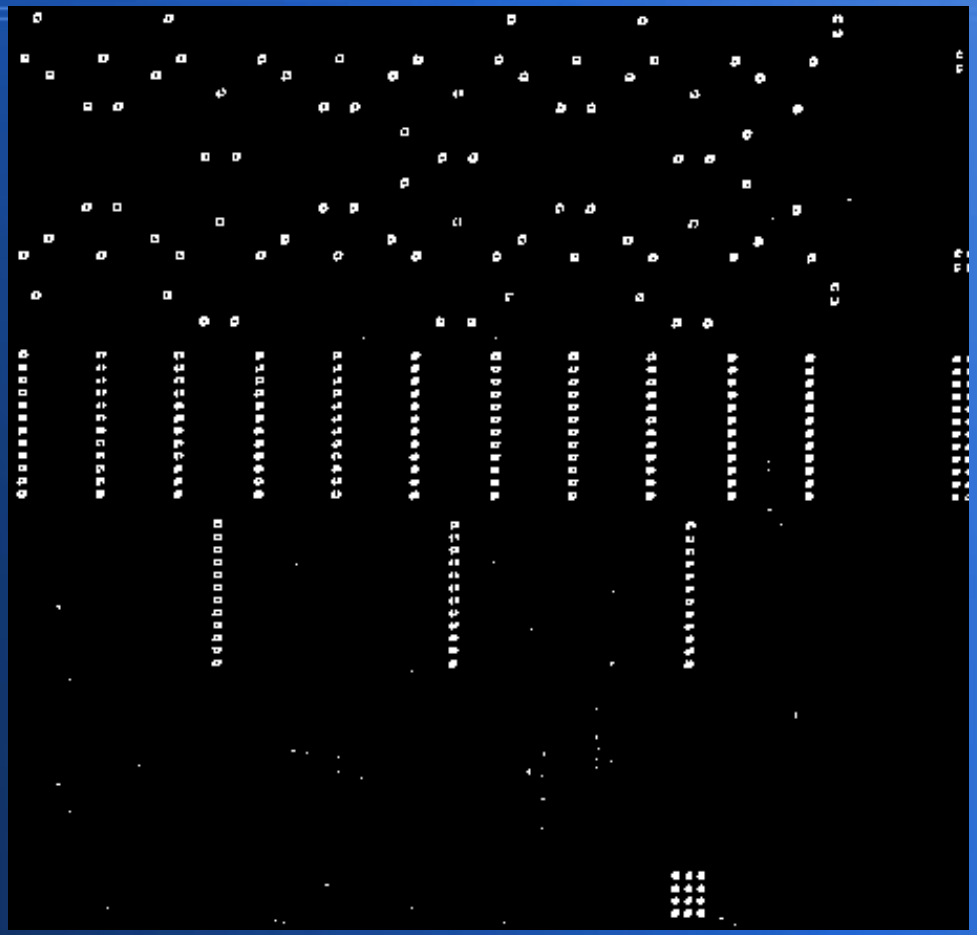




$$\begin{bmatrix} -2 & -2 & 0 & -2 & -2 \\ -2 & 0 & 3 & 0 & -2 \\ 0 & 3 & 6 & 3 & 0 \\ -2 & 0 & 3 & 0 & -2 \\ -2 & -2 & 0 & -2 & -2 \end{bmatrix}$$

After thresholding





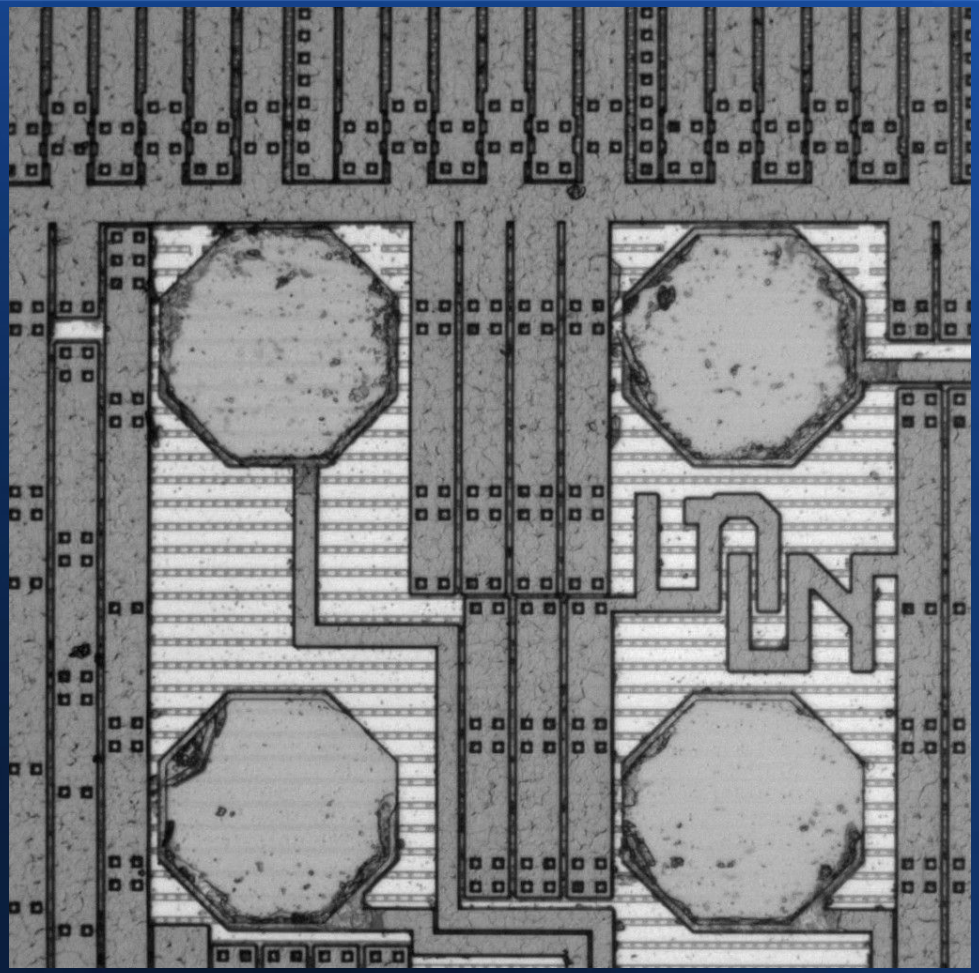
Blurs

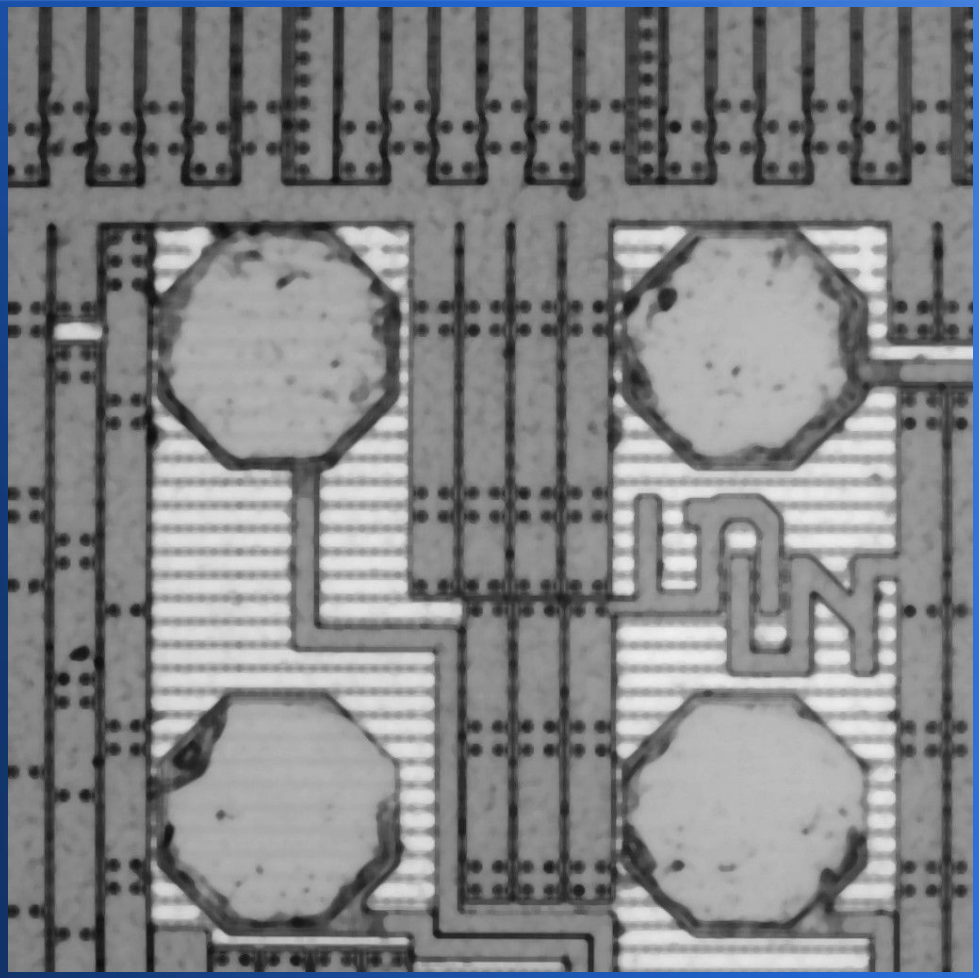
- A blur is essentially a low-pass filter
  - Remove high-frequency components from the image
- Remove noise or focus on larger features

Median filter

- Nonlinear filter for noise removal
- Sample neighborhood around each pixel
- Sort pixels by value and pick the middle
- Small features like speckle noise are removed while larger features and edges are preserved

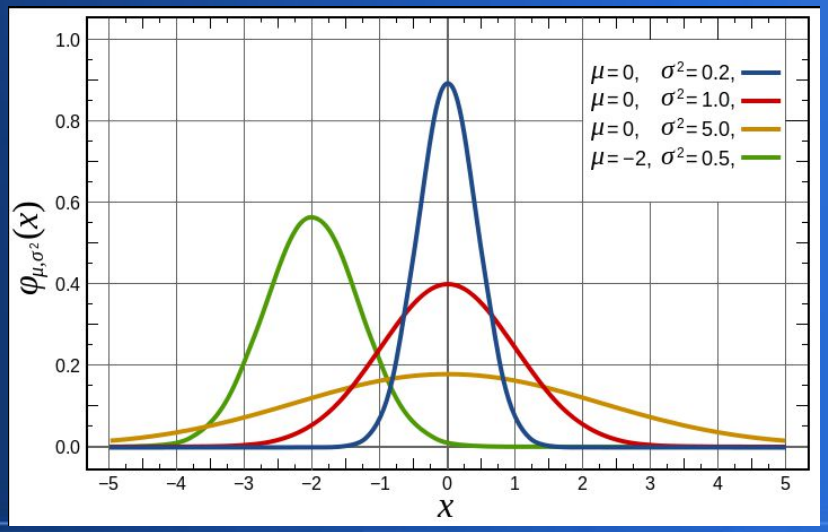
Median filter



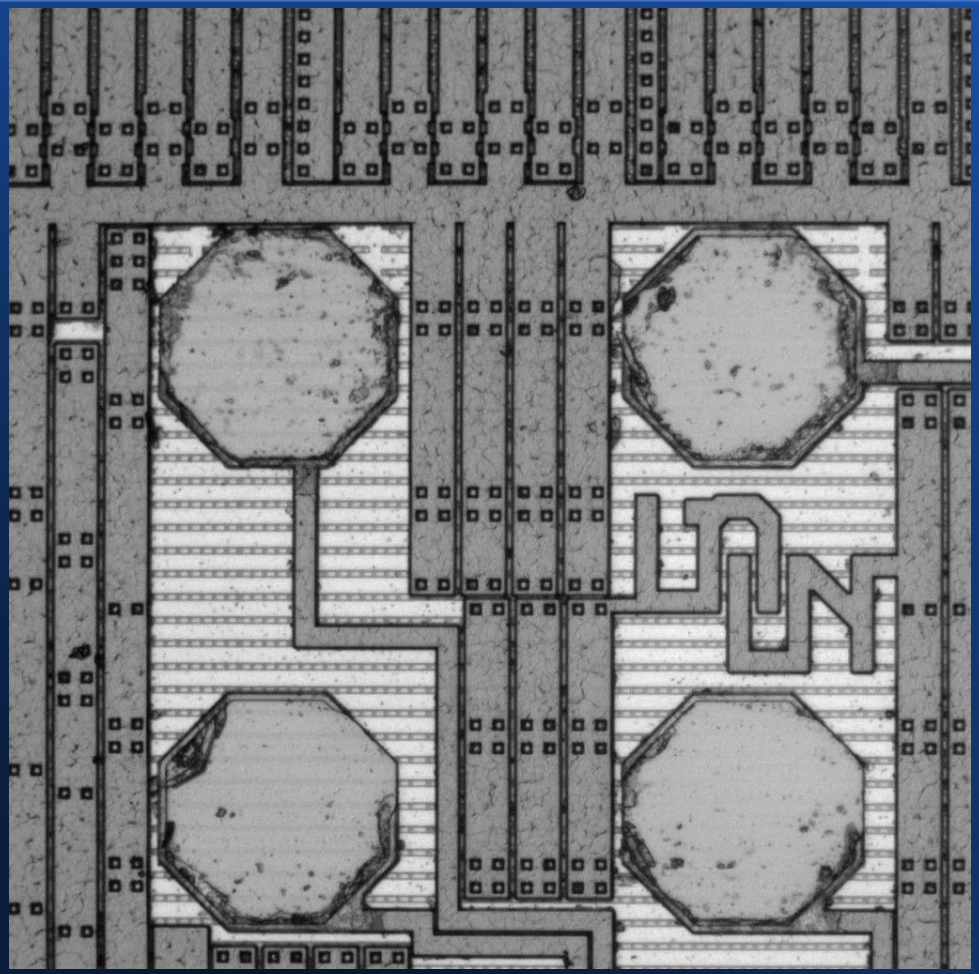


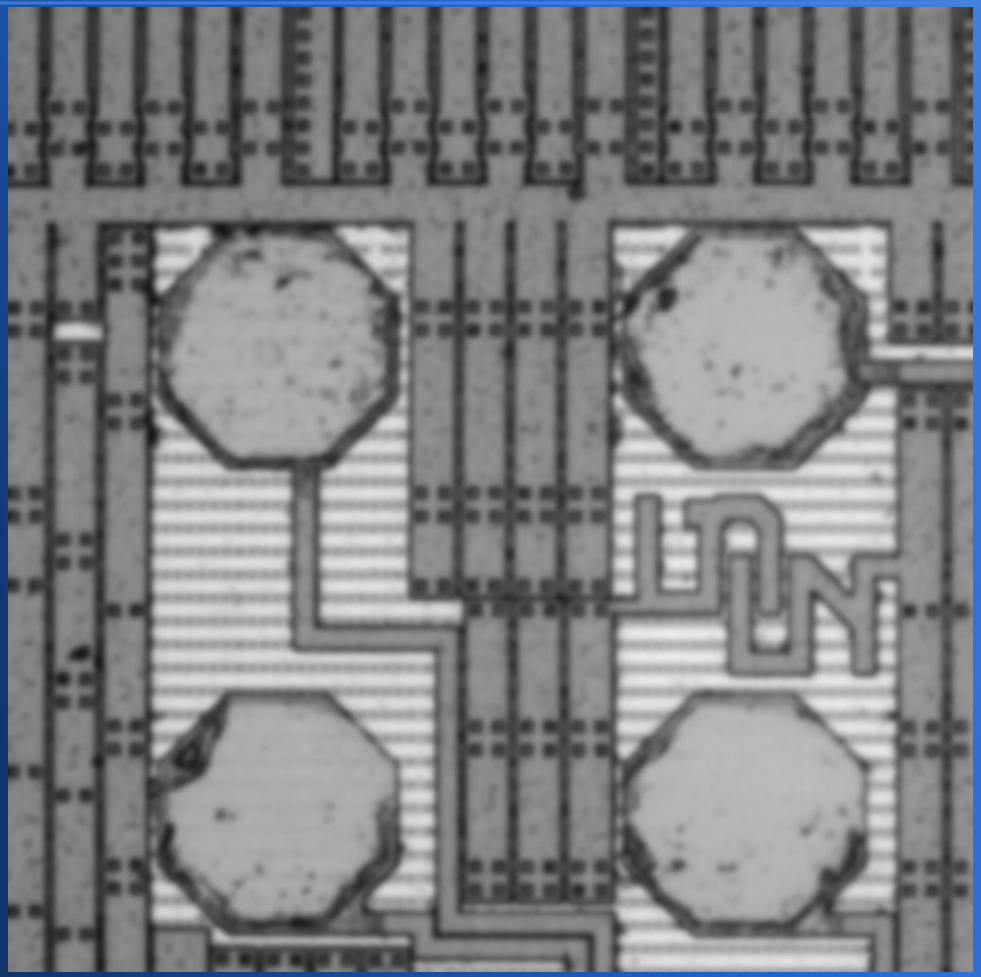
Gaussian blur

- Convolve image with a Gaussian function
- Gaussian function is infinite, wider window for same σ reduces artifacts. 3σ is common
- Changing σ affects cutoff frequency



Gaussian blur





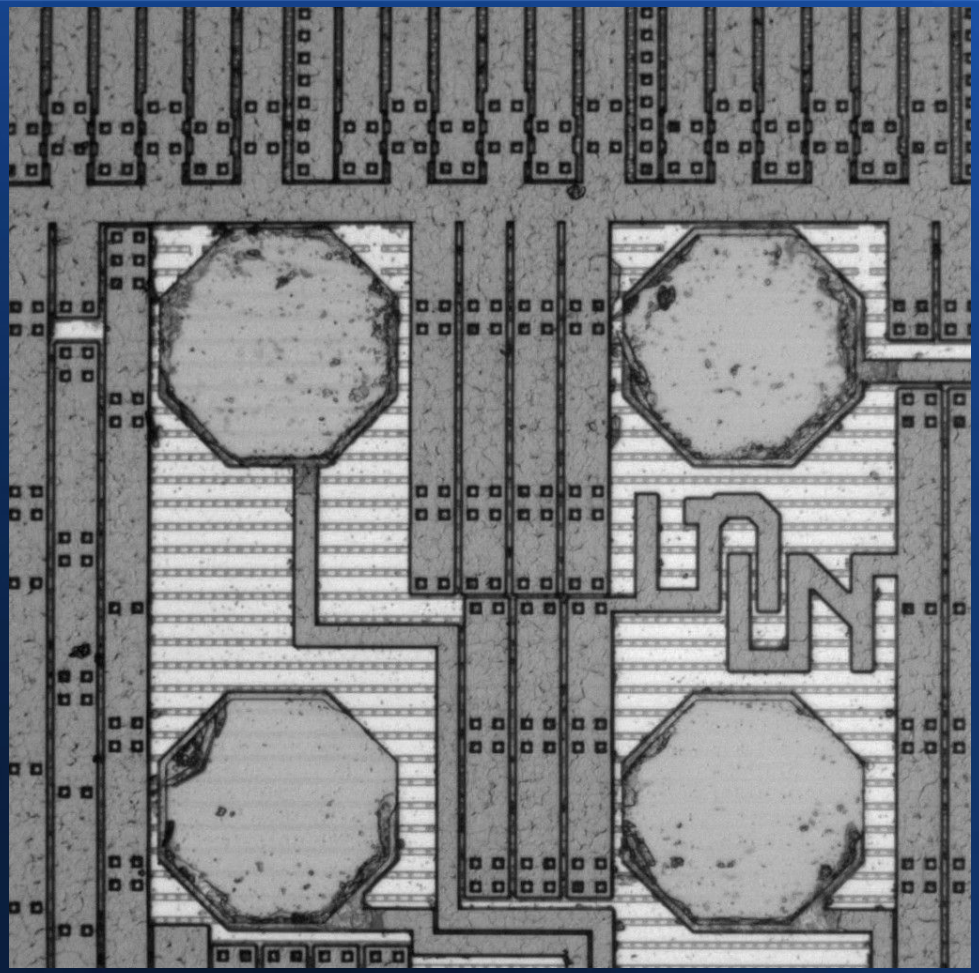
Bandpass filters

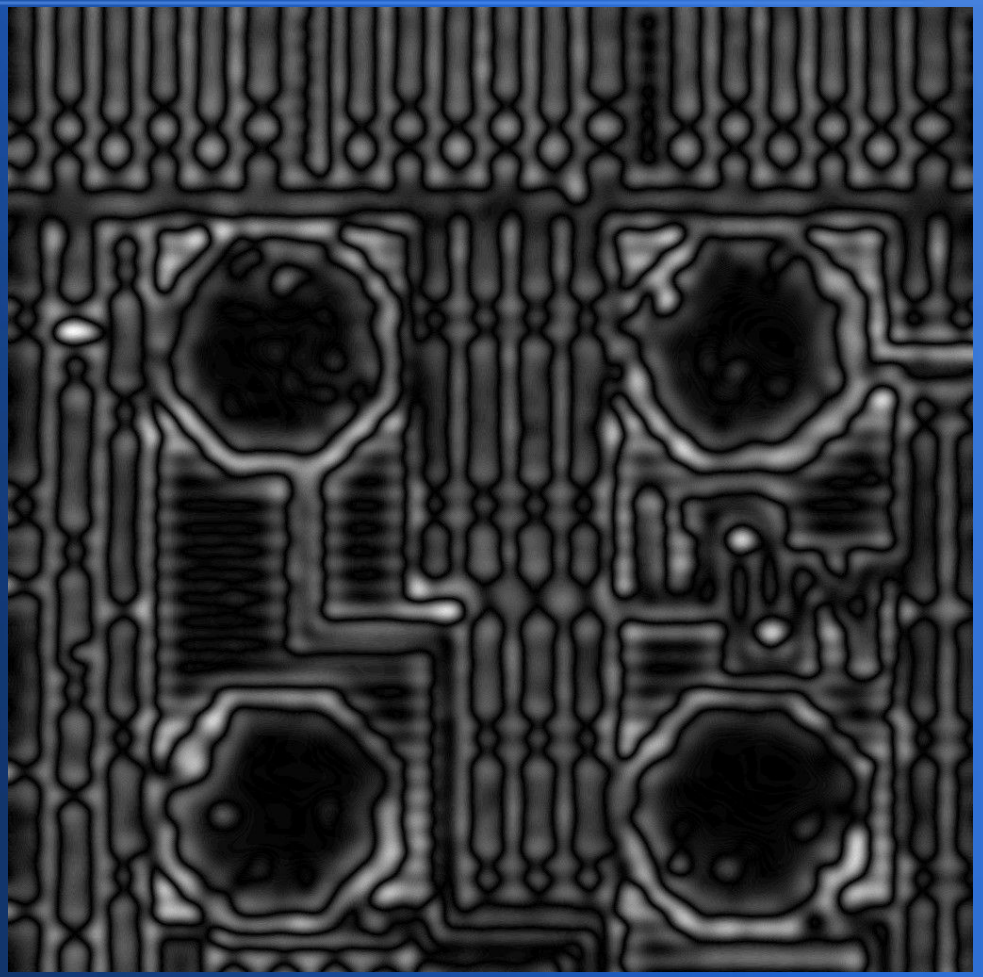
- Highlight features between two size ranges
- Used in scale-space systems to focus on one feature size at a time

Difference of Gaussians

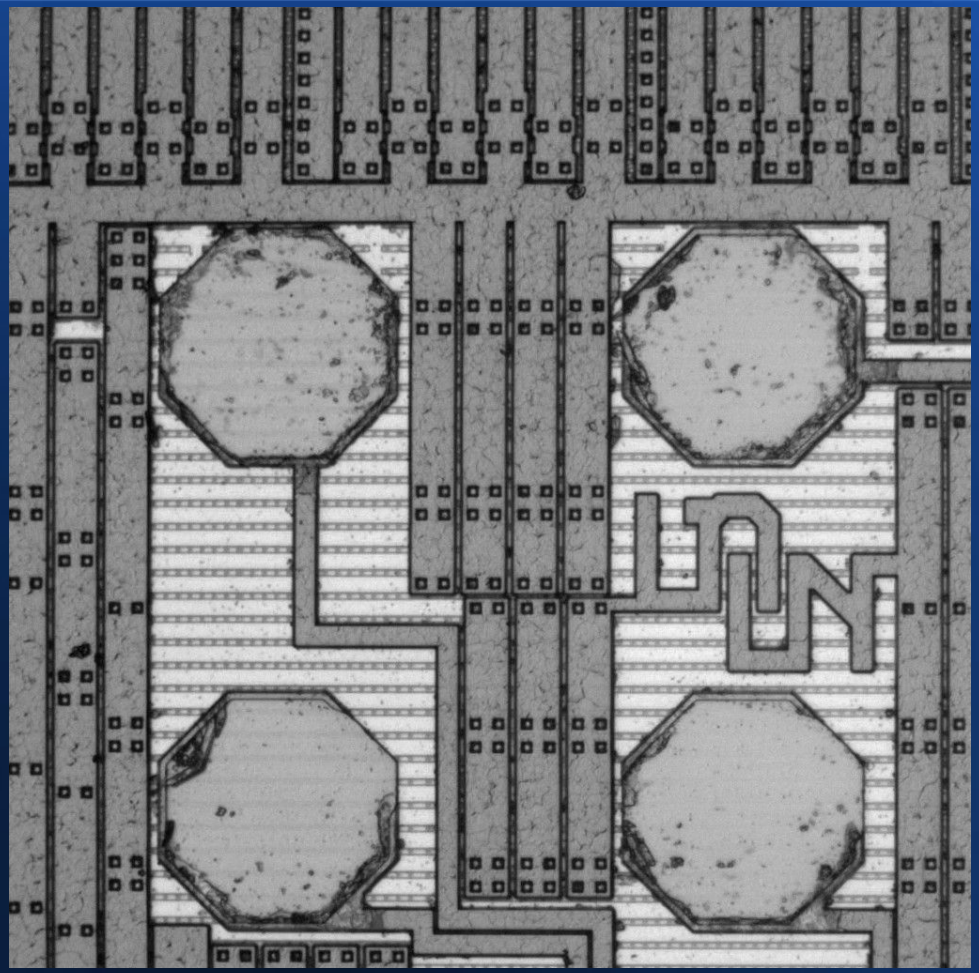
- Bandpass filter made from two Gaussian filters
- Blur with a small σ, then a large σ, and subtract
  - Small features are removed by both filters
  - Large features are passed by both
  - Medium-sized features pass one but not the other and show up in the difference

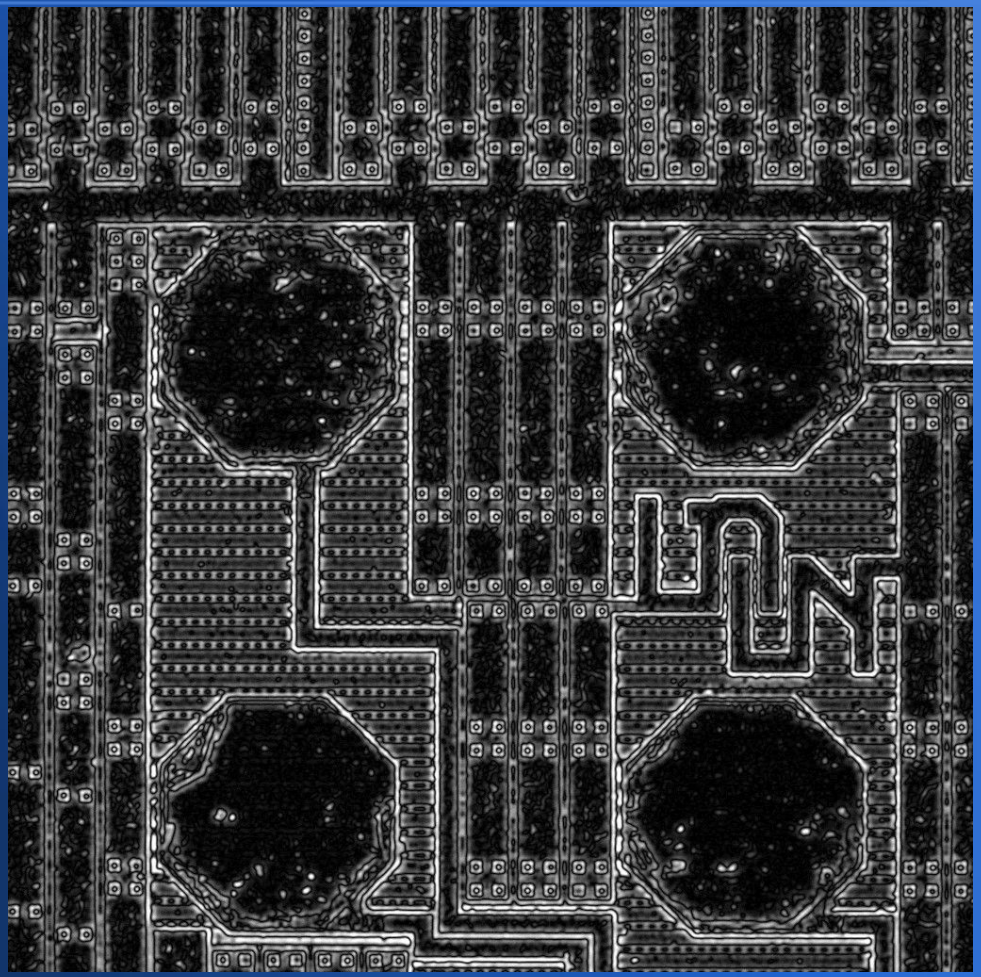
Difference of Gaussians





Difference of Gaussians





Scale invariance

- Results returned by many of these algorithms are highly dependent on the size of the image!
- Common solution to this problem is to take a series of bandpass-filtered images (using DoG etc) and look for features in each one

Keypoint matching

- Common algorithms include SIFT and SURF
- Details vary but the basic flow is similar
  - Create scale-space sequence from image
  - Find interesting areas (keypoints)
  - Compute some function of the image about each keypoint (descriptors)
  - Search for similar descriptors to match similar objects, find the same feature in multiple images, etc

Open source libraries

- OpenCV (http://opencv.org/) - BSD
  - General purpose machine-vision library
- ITK (http://www.itk.org/) - Apache, was BSD
  - Focus on segmentation and registration but has lots of basic filters as well.
  - Originally developed for medical images
  - Developed by Kitware (located in Clifton Park, strong ties to RPI research groups)

Closing notes

- This is our last lecture of the semester! I hope you've all enjoyed the class as much as I have.
- Final project presentations are next Tuesday!

Acknowledgements

- This class could not have happened without help from a lot of people behind the scenes doing training, lab setup, and providing interesting chips for us to study.
- RPI: Prof. Dan Lewis (MatSci), Ray Dove (EM lab), Bryant Colwill and David Frey (cleanroom)
- The siliconpr0n.org team: John McMaster, marshallh, balrog, Lord_Nightmare, and anyone else I forgot.

Questions?

- TA: Andrew Zonenberg <azonenberg@drawersteak.com>
- Image credit: Some images CC-BY from:
  - John McMaster <JohnDMcMaster@gmail.com>