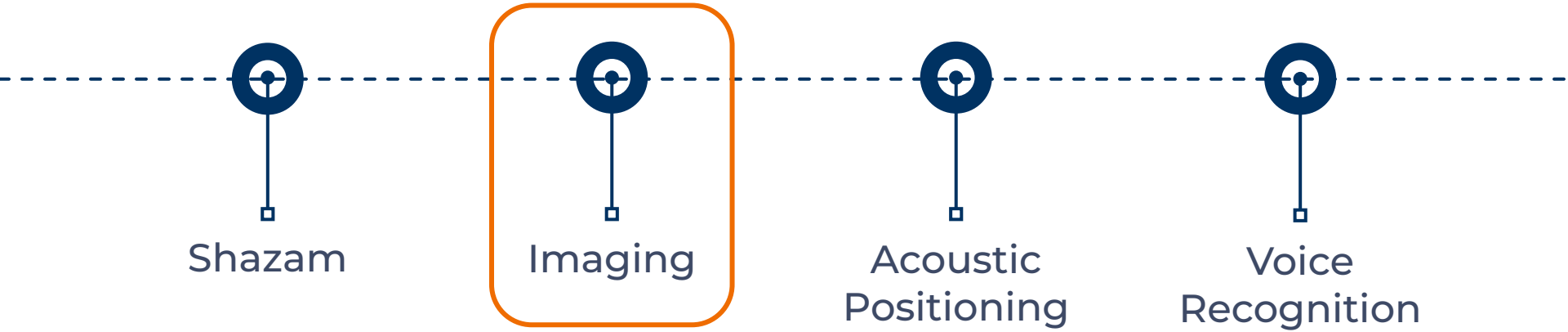
EECS 16A Imaging 2

We will start at Berkeley Time!

Working In Pairs

- Complete the lab in PAIRS, do ONE setup and notebook per groups.
- Speak to the staff if you do not have a lab partner.

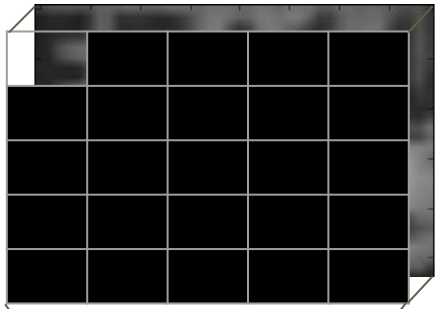
Semester Outline



Agenda

- Images as matrices and vectors
- Pixel-by-pixel scanning
- Reconstructing scans as images

Our Imaging System



Reflected Light

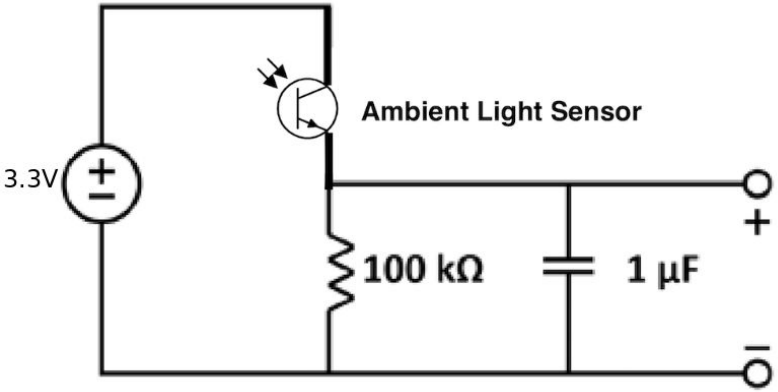
Light Sensor

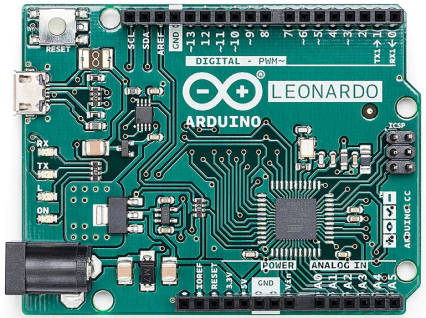
Reconstruction

Project masks onto object

Projector

Light Sensor?





PC

- This is the circuit that senses our reflected light
- For our purposes, it's a black box that turns light levels into voltage values, a signal that computers can work with

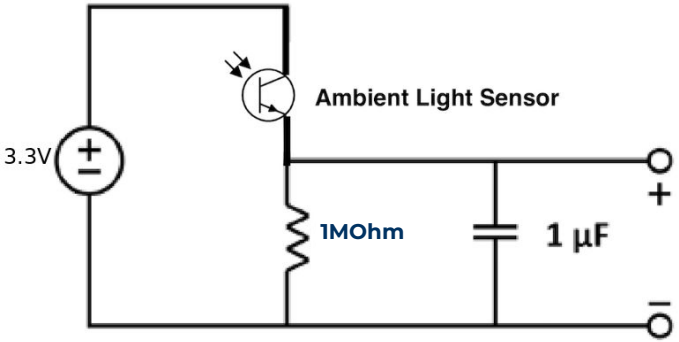
Why Imaging?

- **Module Idea:** use linear algebra techniques to capture real world images with limited sensors.
- **Today:**
  - Become familiar with our imaging setup
  - Use single-pixel scanning to capture image

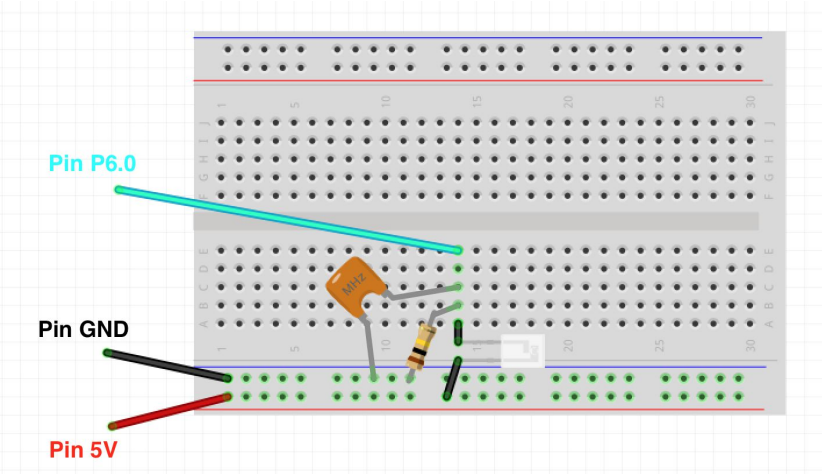
Last Week: Imaging 1

- Built our very first circuit!
  - What did this circuit do?

Circuit Diagram



Breadboard Diagram



Today's Lab: Single Pixel Scanning

- Circuit from last week measures **light** intensity
- Simulated projector illuminates image in a controlled way
- Python programming to reconstruct image

Why?

- Imaging 1:
  - Finding a link between physical quantities and voltage is powerful
  - If you can digitize it, you can do anything (IOT devices, internet, code, processing)
- Imaging 2:
  - How do we get measurements and what makes them good?
  - How do we get to an image?

Illuminating the Big Picture

- Linear dependence
  - When can you recover your image?
  - Does it matter what mask matrix you pick?
  - Does it matter how you cover the pixels?
- Invertibility
  - When can you solve $Ax = b$?
  - How does this relate to our system?
  - How does this affect the way we pick our masking matrix?

Images, Matrices, Vectors

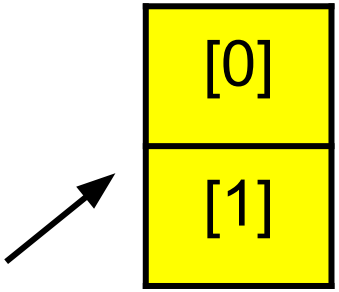
- What are the unknowns in our system?
  - The Image !
- We can do a lot of interesting processing on vectors, but we need to convert the image into one first
  - How can we do this?

Images, Matrices, Vectors

[0]	[1]
[2]	[3]
[4]	[5]

Images, Matrices, Vectors

[0]	[1]
[2]	[3]
[4]	[5]



Images, Matrices, Vectors

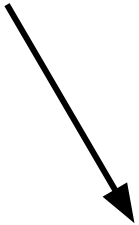
[0]	[1]
[2]	[3]
[4]	[5]

→

[0]
[1]
[2]
[3]

Images, Matrices, Vectors

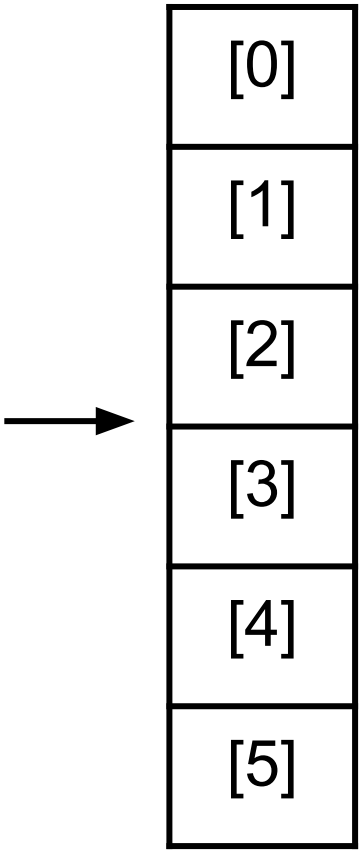
[0]	[1]
[2]	[3]
[4]	[5]



[0]
[1]
[2]
[3]
[4]
[5]

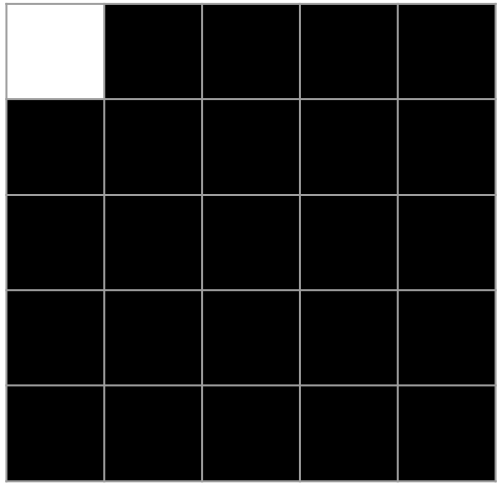
Images, Matrices, Vectors

[0]	[1]
[2]	[3]
[4]	[5]



Pixel-by-Pixel Scan of an Image

Pixel-by-Pixel Scan of an Image

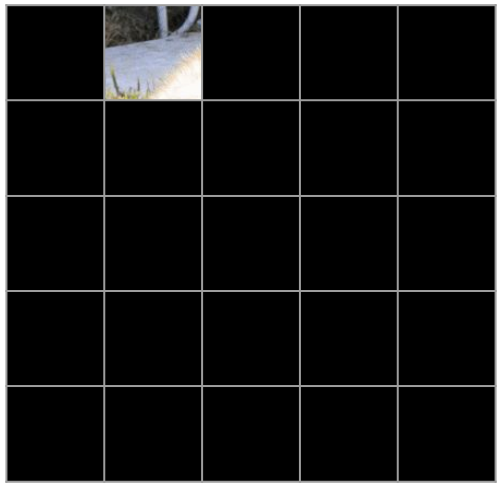


Pixel-by-Pixel Scan of an Image

Masked image

Image

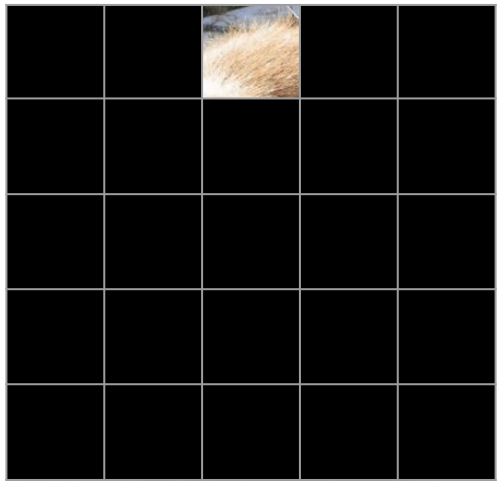
Pixel-by-Pixel Scan of an Image



Masked image

Image

Pixel-by-Pixel Scan of an Image



Masked image

Image

Poll Time!

What would you expect the dimensions of a vector representing a 2x3 image to be?

A. 2x3
B. 3x2
C. 6x1
D. 5x1

To read all the pixels of a 4x4 image, how many pixel-by-pixel scans do we need to do?

A. 4
B. 8
C. 16
D. 32

Poll Time!

What would you expect the dimensions of a vector representing a 2x3 image to be?

A. 2x3
B. 3x2
C. 6x1
D. 5x1

To read all the pixels of a 4x4 image, how many pixel-by-pixel scans do we need to do?

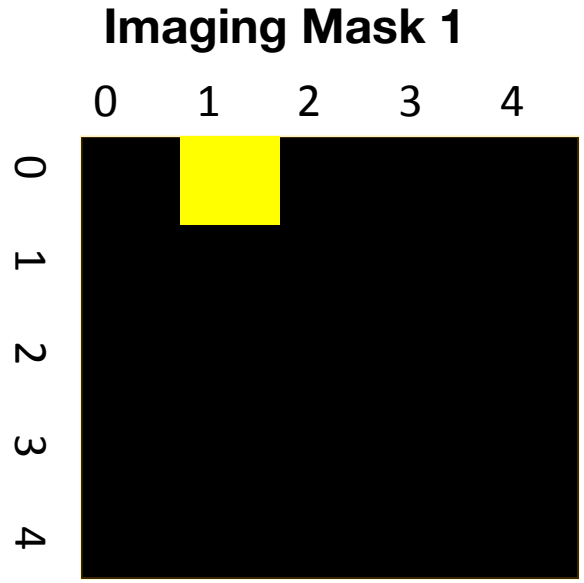
A. 4
B. 8
C. 16
D. 32

Representing our Masks in Python

mask0 =

```
np.array([ [ 1, 0, 0, 0, 0 ],
           [ 0, 0, 0, 0, 0 ],
           [ 0, 0, 0, 0, 0 ],
           [ 0, 0, 0, 0, 0 ],
           [ 0, 0, 0, 0, 0 ]]
)
```

Representing our Masks in Python



mask1 =

np.array([

[	0,	1,	0,	0,	0],
[	0,	0,	0,	0,	0],
[	0,	0,	0,	0,	0],
[	0,	0,	0,	0,	0],
[	0,	0,	0,	0,	0]]

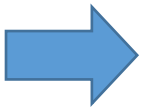
)

Turning the Masks Into Vectors

5x5 mask to 25x1 vector

mask0 =

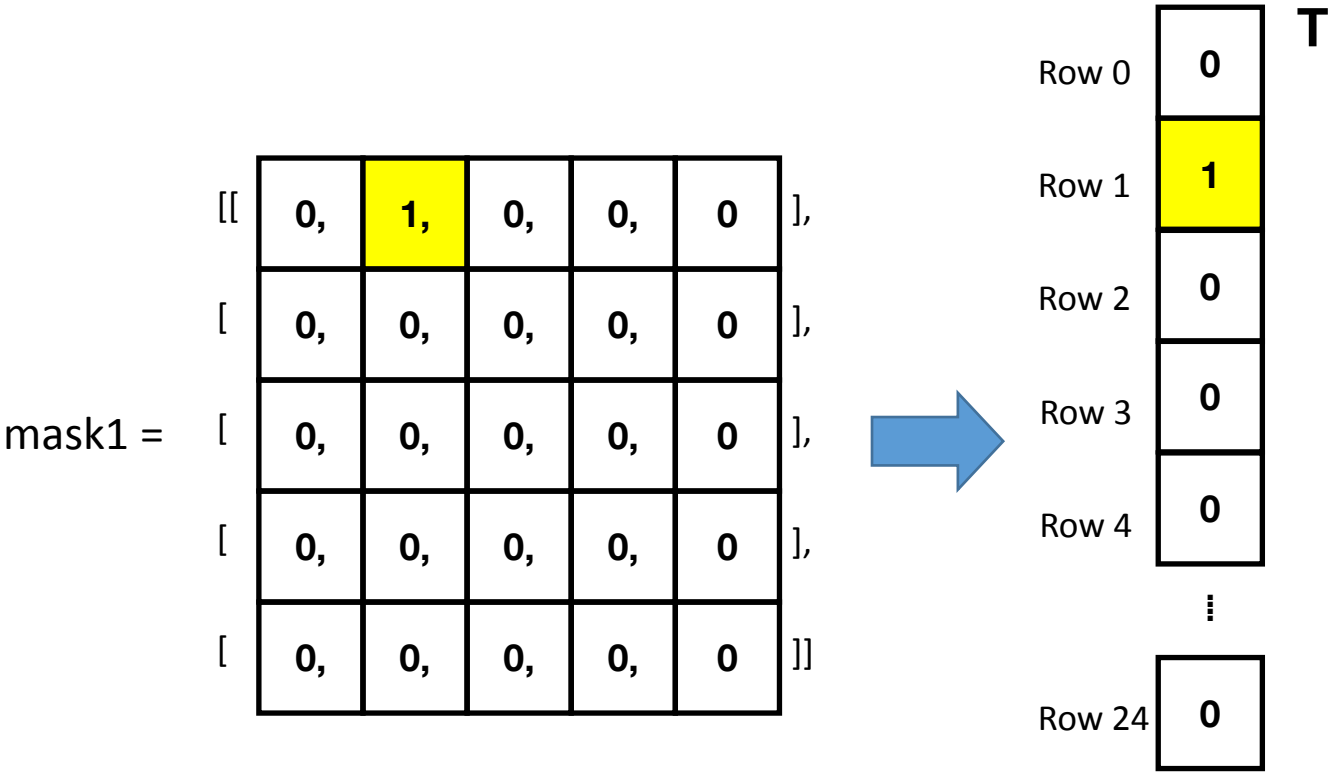
[[	1,	0,	0,	0,	0],
[	0,	0,	0,	0,	0],
[	0,	0,	0,	0,	0],
[	0,	0,	0,	0,	0],
[	0,	0,	0,	0,	0]]



T

Row 0	1
Row 1	0
Row 2	0
Row 3	0
Row 4	0
⋮	
Row 24	0

Turning the Masks Into Vectors



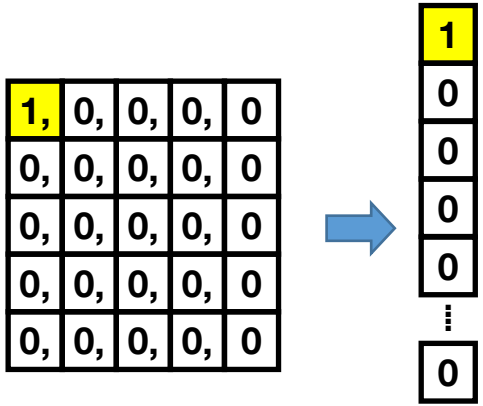
Generating the Masking Matrix from the Masks

mask0

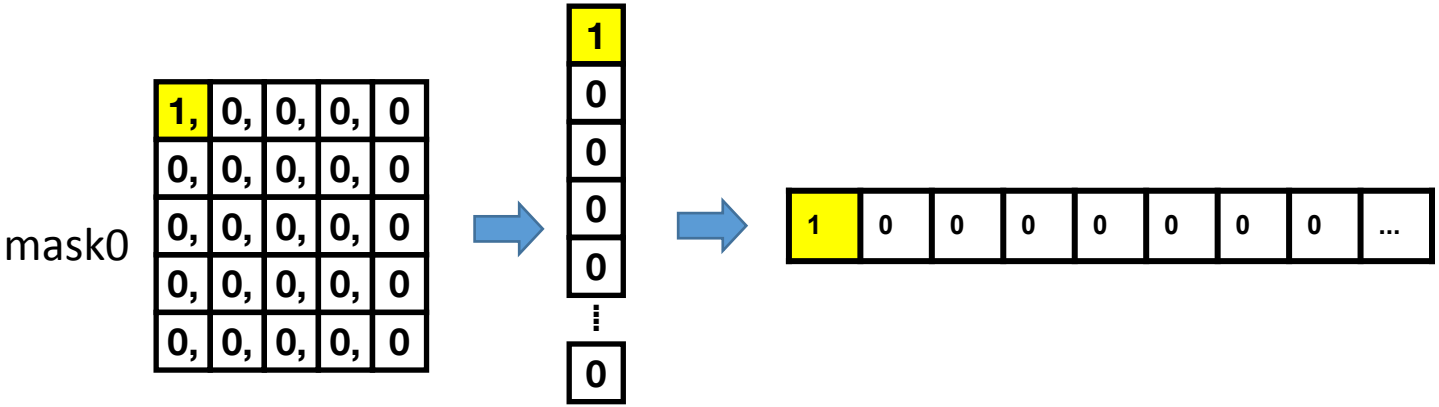
1,	0,	0,	0,	0
0,	0,	0,	0,	0
0,	0,	0,	0,	0
0,	0,	0,	0,	0
0,	0,	0,	0,	0

Generating the Masking Matrix from the Masks

mask0



Generating the Masking Matrix from the Masks



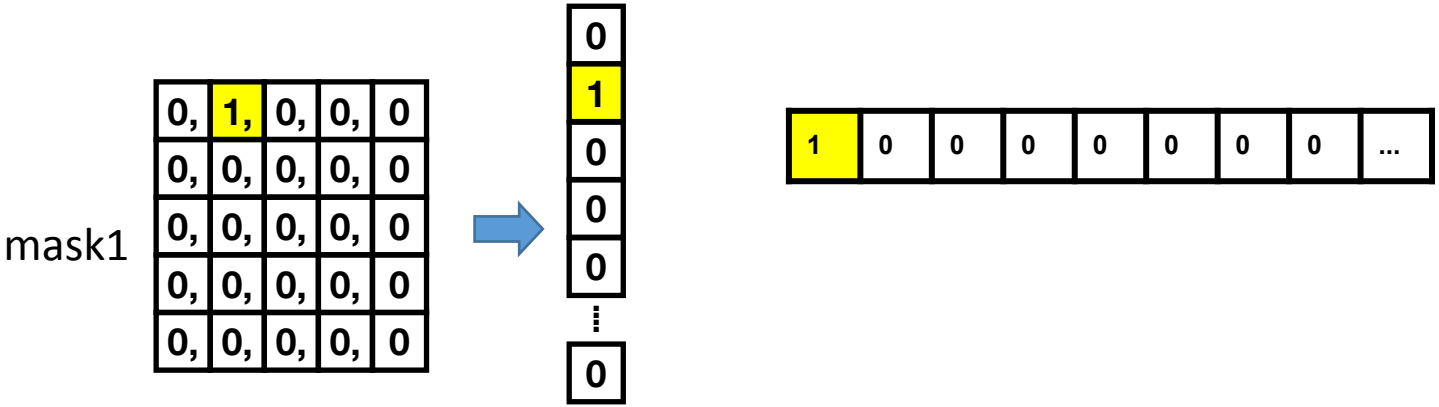
Generating the Masking Matrix from the Masks

mask1

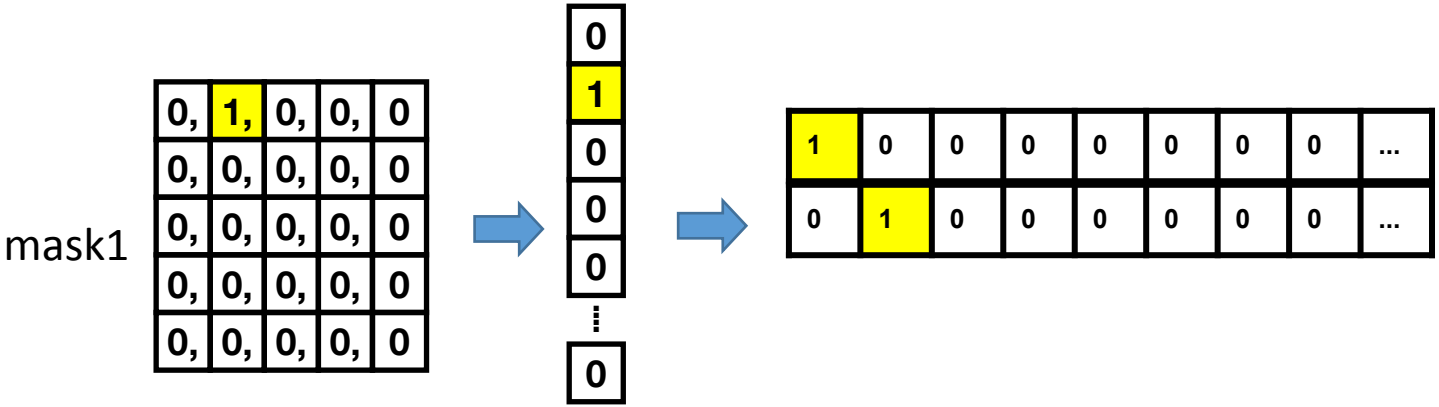
0,	1,	0,	0,	0
0,	0,	0,	0,	0
0,	0,	0,	0,	0
0,	0,	0,	0,	0
0,	0,	0,	0,	0

1	0	0	0	0	0	0	0	...

Generating the Masking Matrix from the Masks



Generating the Masking Matrix from the Masks



Generating the Masking Matrix from the Masks

1	0	0	0	0	0	0	0	...
0	1	0	0	0	0	0	0	...
0	0	1	0	0	0	0	0	...

Generating the Masking Matrix from the Masks

1	0	0	0	0	0	0	0	...
0	1	0	0	0	0	0	0	...
0	0	1	0	0	0	0	0	...
0	0	0	1	0	0	0	0	...

Generating the Masking Matrix from the Masks

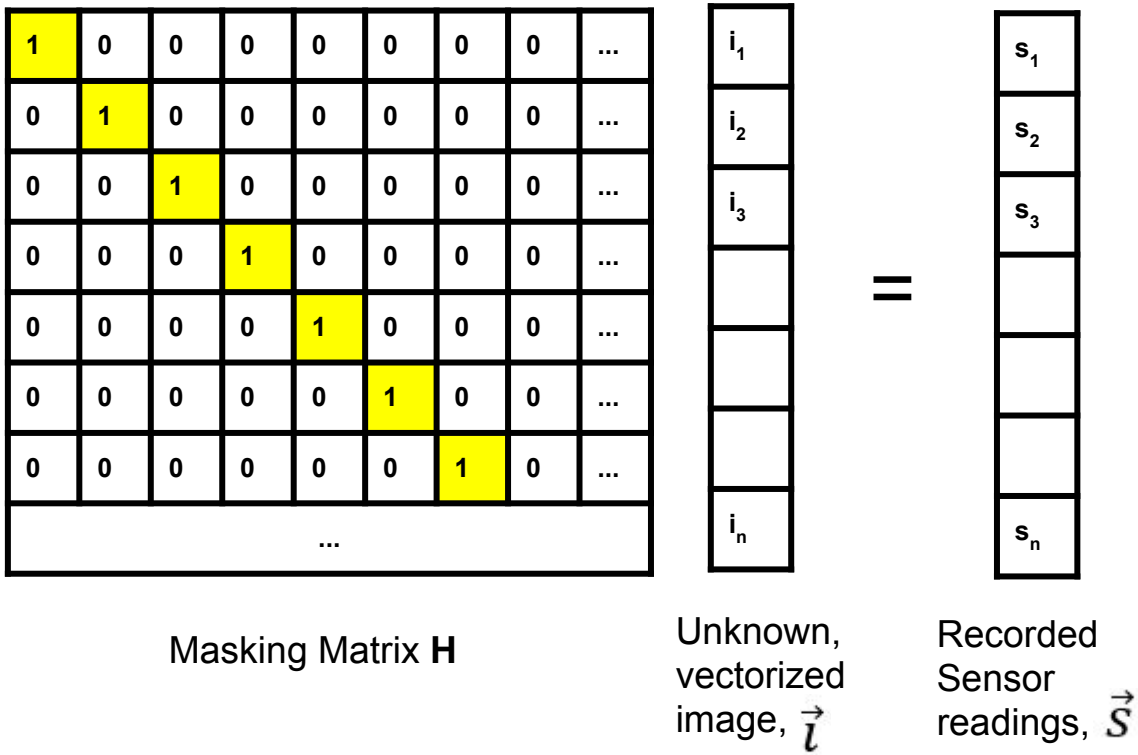
↓ each column represents a pixel

$H =$

1	0	0	0	0	0	0	0	...
0	1	0	0	0	0	0	0	...
0	0	1	0	0	0	0	0	...
0	0	0	1	0	0	0	0	...
0	0	0	0	1	0	0	0	...
0	0	0	0	0	1	0	0	...
0	0	0	0	0	0	1	0	...
...								

← each row is a mask

Measuring a Pixel is Matrix-Vector Multiplication



Masking Matrix **H**

Unknown, vectorized image, $\vec{i}$

Recorded Sensor readings, $\vec{s}$

Measuring a Pixel is Matrix-Vector Multiplication

$$\vec{s} = H\vec{\imath}$$

- We know H and we have the sensor readings, how do we get the image?
- How do we solve this?
- When can we solve this?
  - Conditions on H

Poll Time!

$$\vec{s} = H\vec{i}$$

Select all of the following that must be true for the image vector i to be recoverable from the sensor vector s.

1. H must be invertible
2. H must have linearly independent rows
3. H must be a square matrix
4. H must be the identity matrix

Poll Time!

Select all of the following that must be true for the image vector i to be recoverable from the sensor vector s.

1. H must be invertible
2. H must have linearly independent rows
3. H must be a square matrix
4. H must be the identity matrix

Poll Time!

Select all of the following that describe the relationship between H (the masking matrix), s (the sensor vector), and i (the image vector)?

1. $Hs = i$
2. $Hi = s$
3. $H^{-1}i = s$
4. $H^{-1}s = i$
5. $i * s = H$

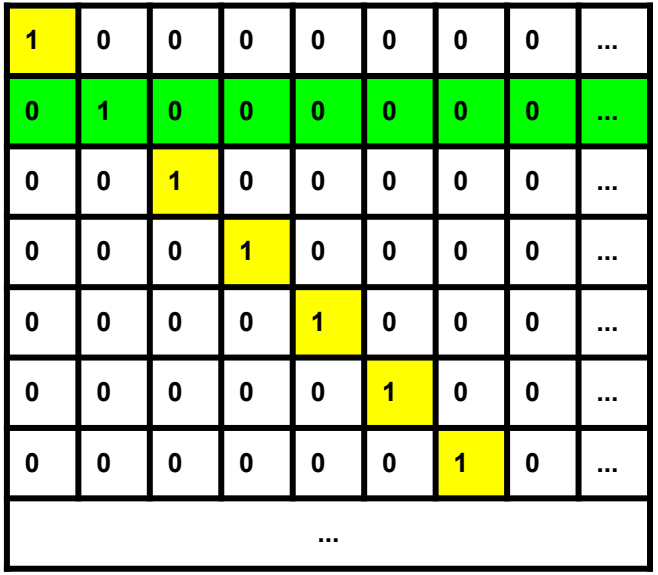
Poll Time!

Select all of the following that describe the relationship between H (the masking matrix), s (the sensor vector), and i (the image vector)?

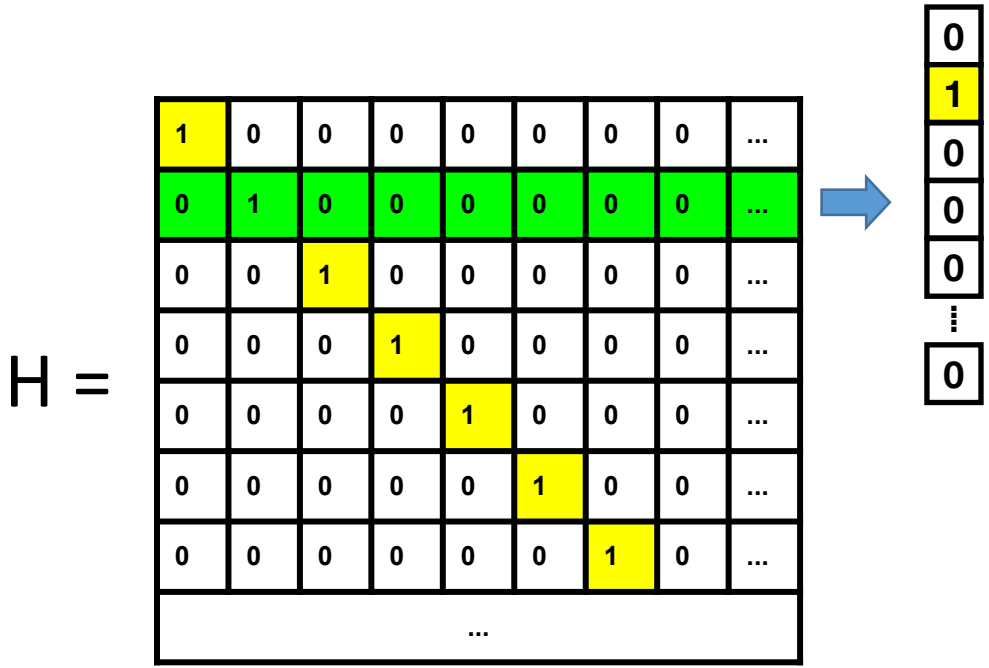
1. $Hs = i$
2. $Hi = s$
3. $H^{-1}i = s$
4. $H^{-1}s = i$
5. $i * s = H$

How Scanning Works: iPython

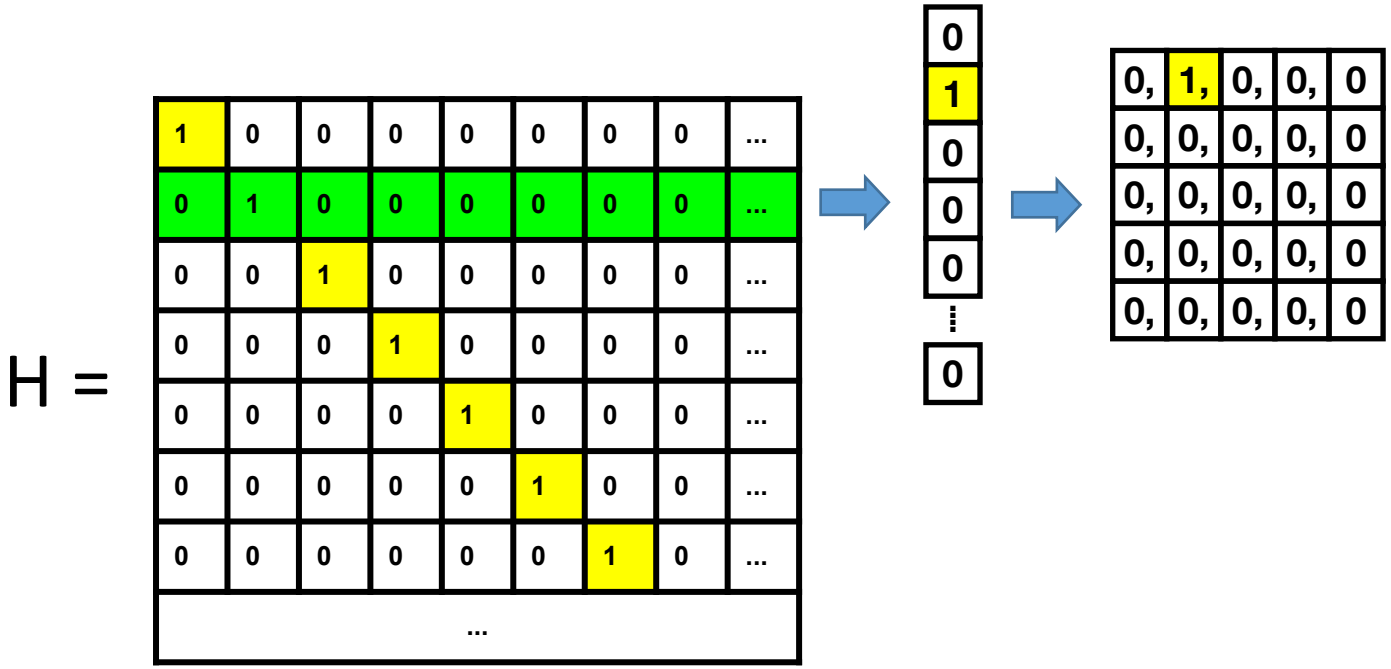
$H =$



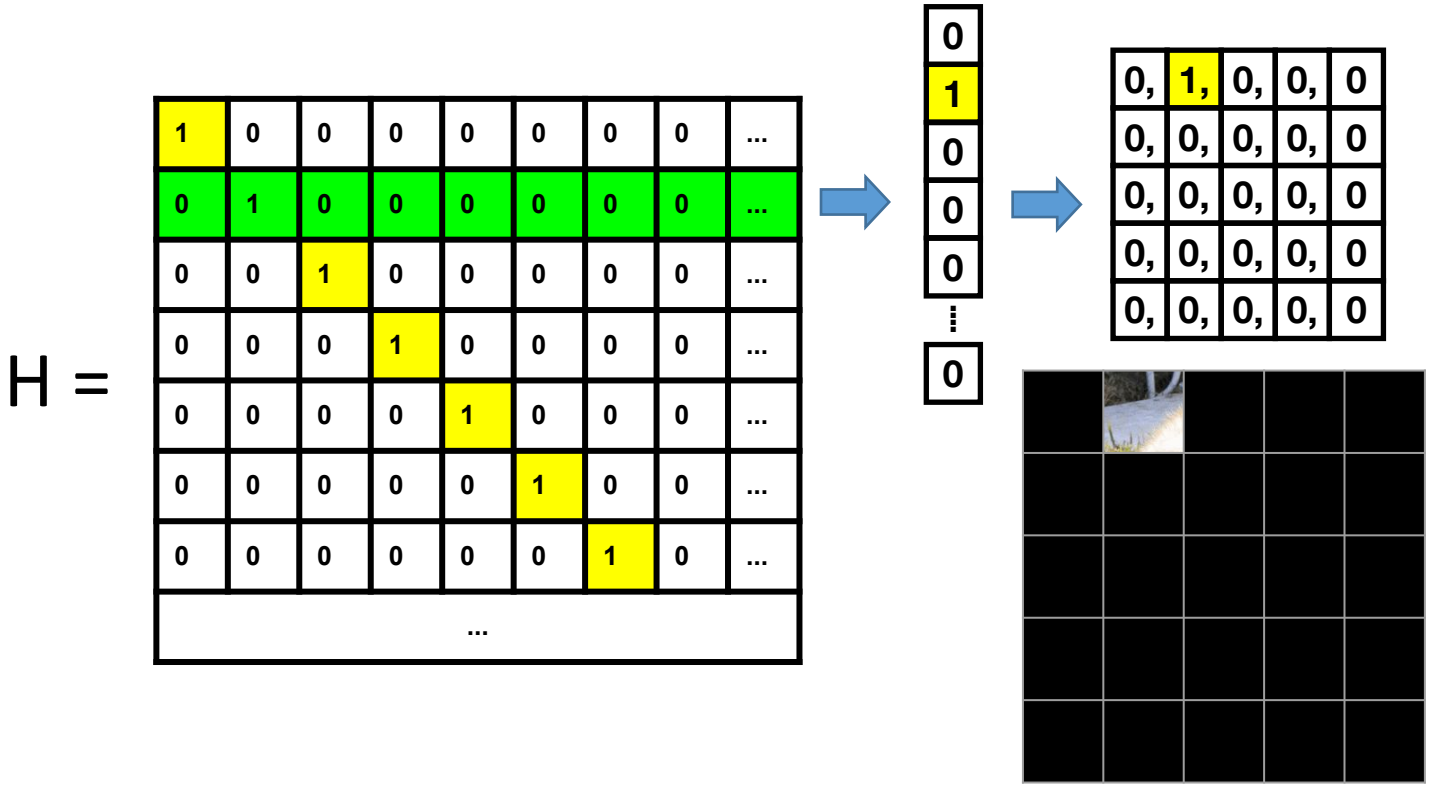
How Scanning Works: iPython



How Scanning Works: iPython



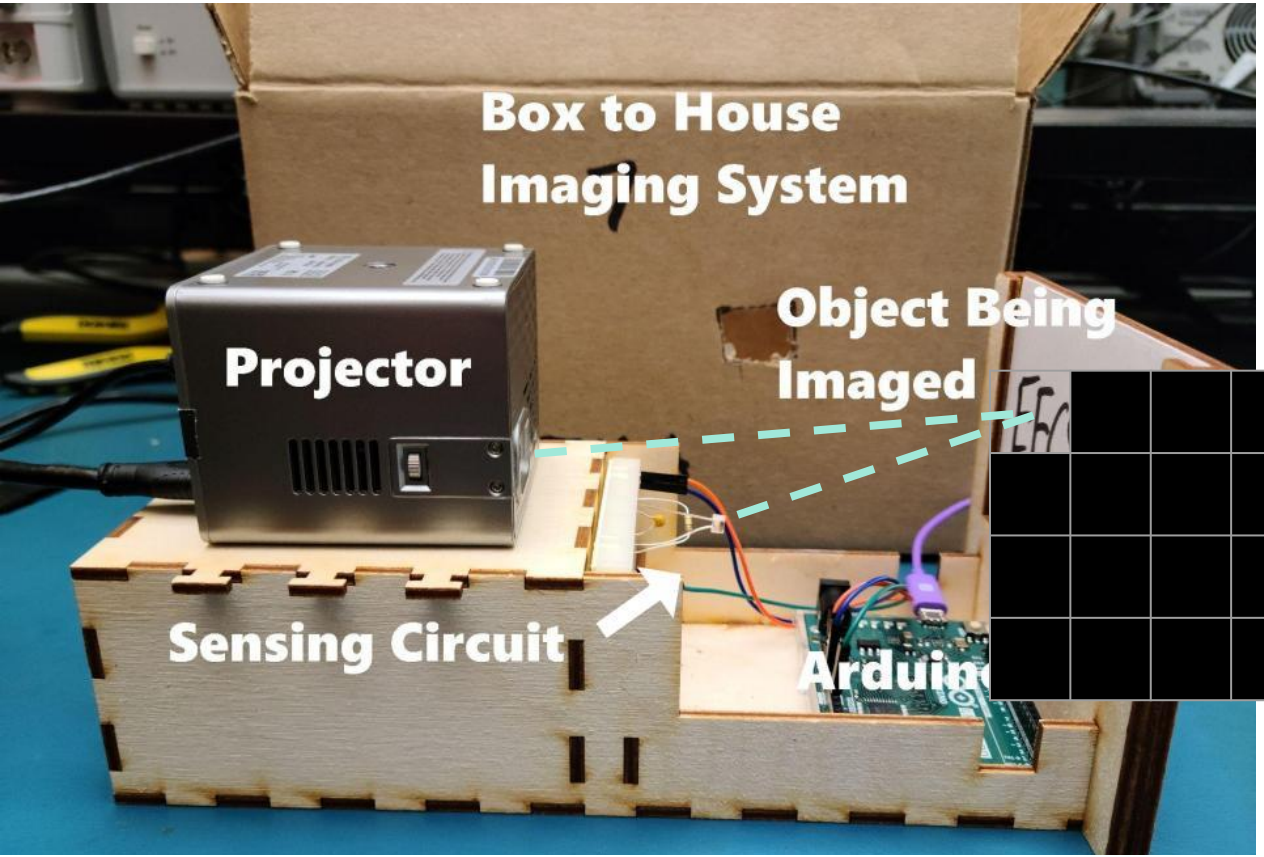
How Scanning Works in Python



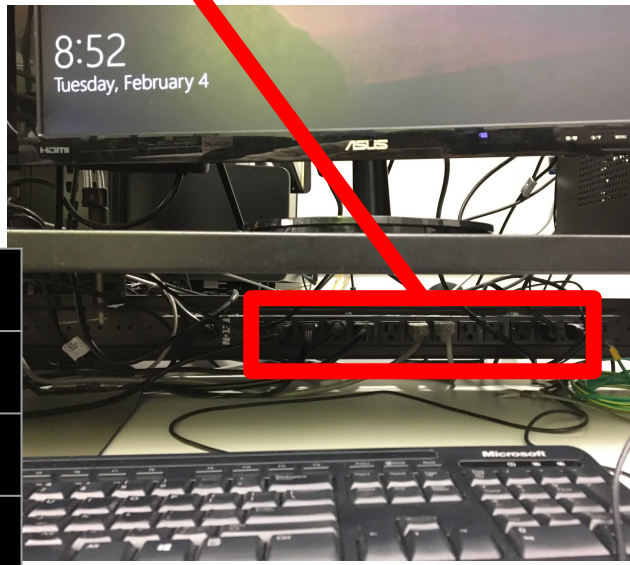
What Makes a Mask Good?

- Linearly independent columns → Invertible
  - Can’t get a solution without this
  - There is a unique solution
- What would be a bad mask?
- Food for thought: Are all invertible matrices equally as good?
  - Find out in Imaging 3 next week

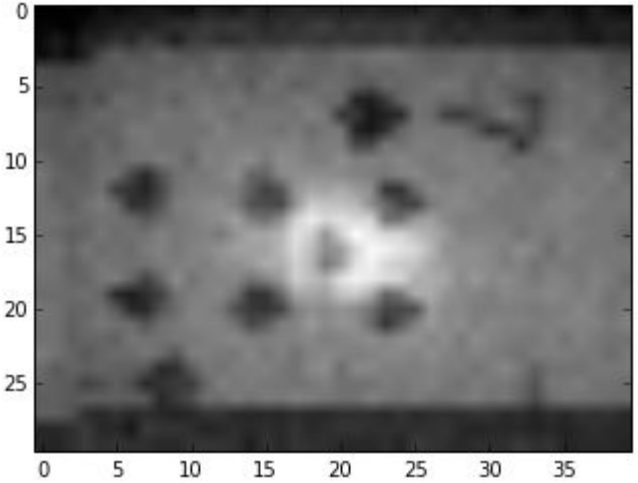
Setup



Power strip to power your projector



Sample Images



Setup

1. Draw a “simple” image
2. Use Python to project masks onto it in a dark environment
3. Measure ambient light sensor reading to get s
4. Multiply by H inverse to find i (= $H^{-1}s$)

Color Imaging!

- The masks we have been using so far have been black and white (1s and 0s). Thus, B&W images
- What if we use color masks instead?
- Make use of RGB (red green blue) channels and reconstruct three different scans
- Same system as before: $\vec{s} = H\vec{\imath}$
- Only difference is one “system” for each color
- With a bit of math/signal processing, we can get color images!

Tips for a Good Image

- READ CLOSELY. There are many small directions that help you get a good setup
- Focus projector using dial on the side
- Close the box firmly & scan under dark conditions
- Make sure the cables are plugged in, and do not disturb them during the scanning process

Debugging

1. Make sure wires/resistors/light sensor are not loose
2. Light sensor orientation: short leg goes into +ve
3. Check COM Port
4. Reupload code to Arduino after making any change in circuit
5. Check Baud Rate in Serial Monitor (115200)
6. Projector might randomly restart in the middle of the lab. Make sure brightness 0 contrast 100.
7. If you see a very bright corner in the scan, move the light sensor away from the projector