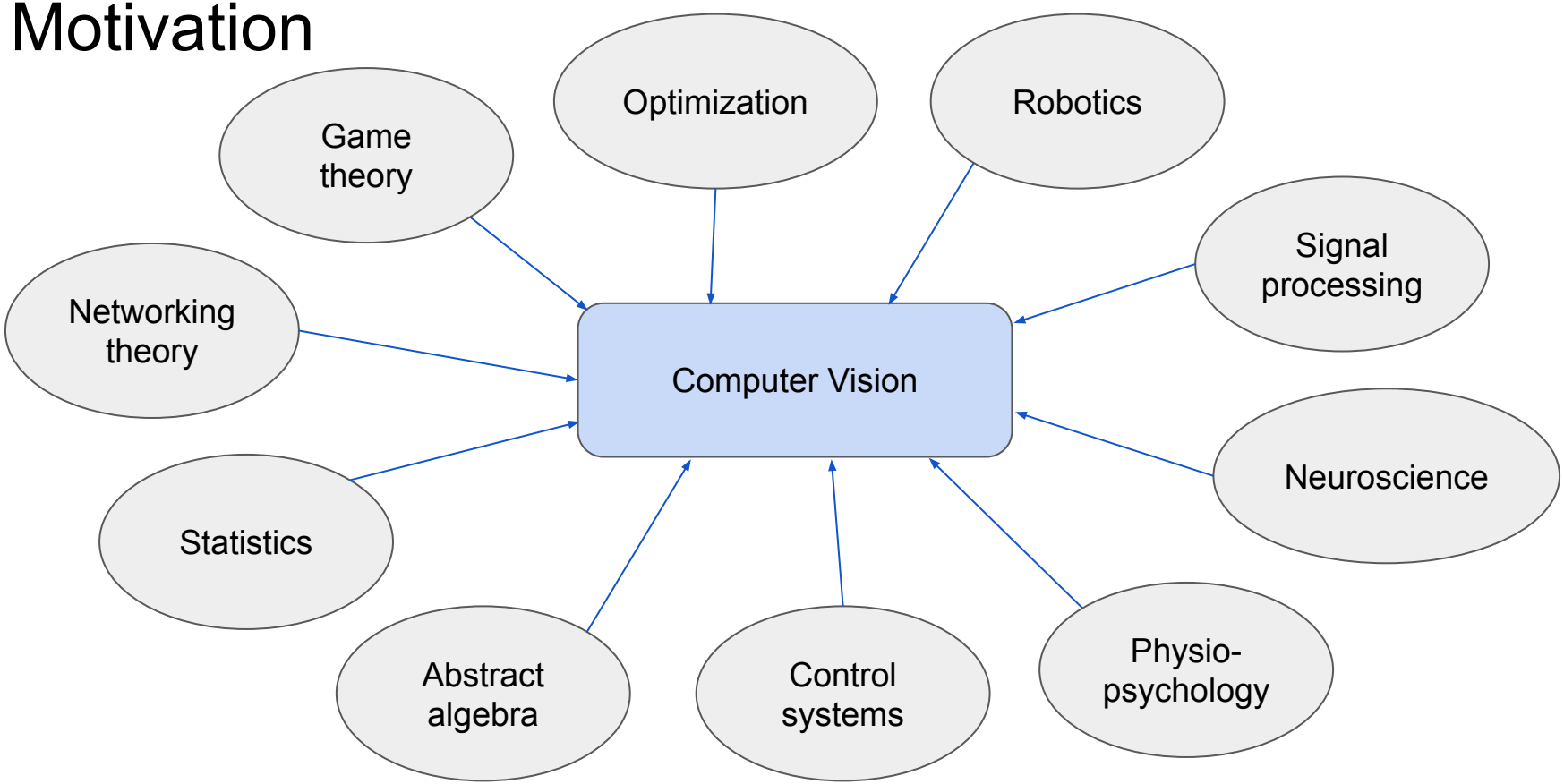
Foundations of Computer Vision

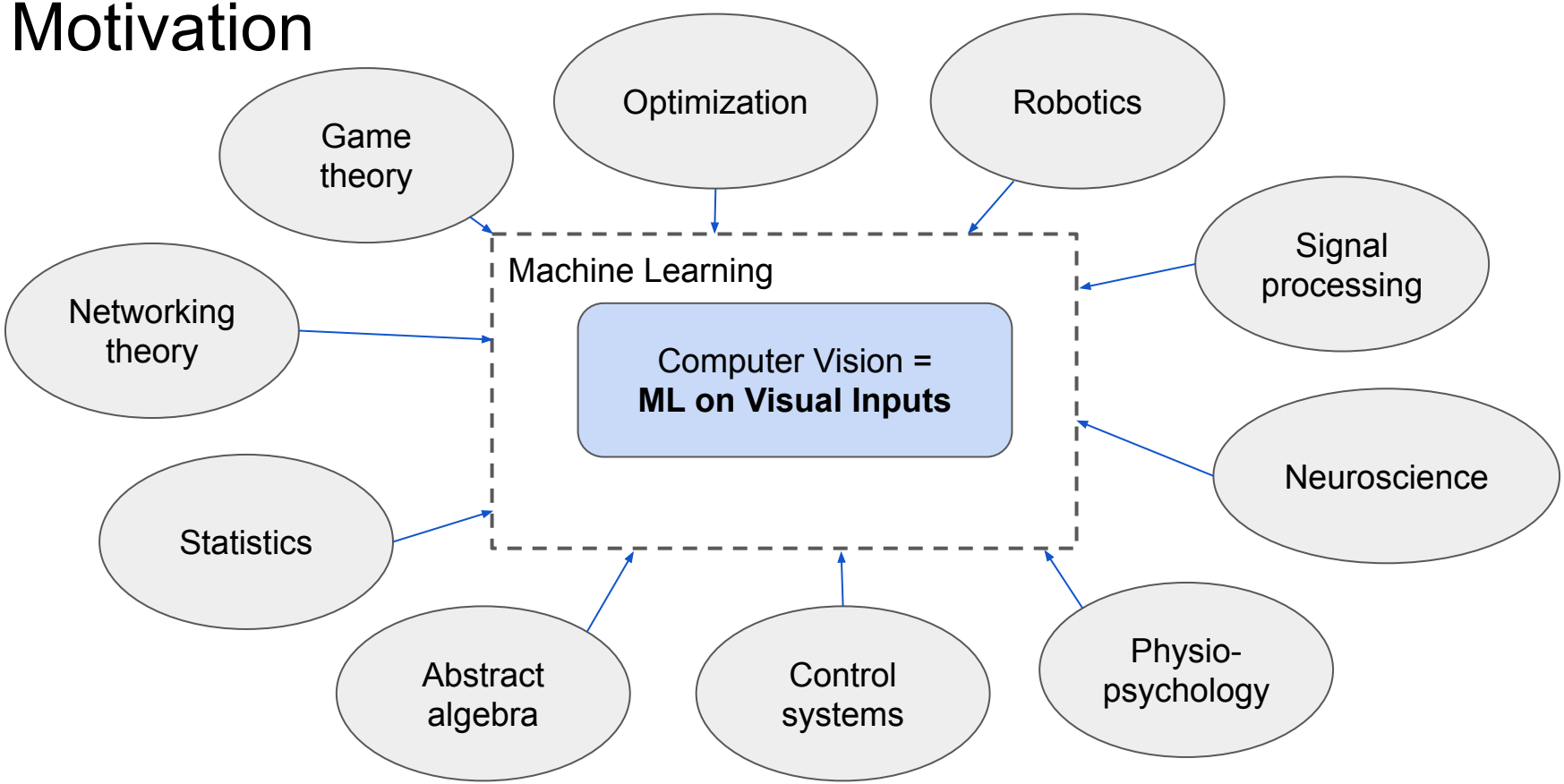
Lecture 1

Parker Koch

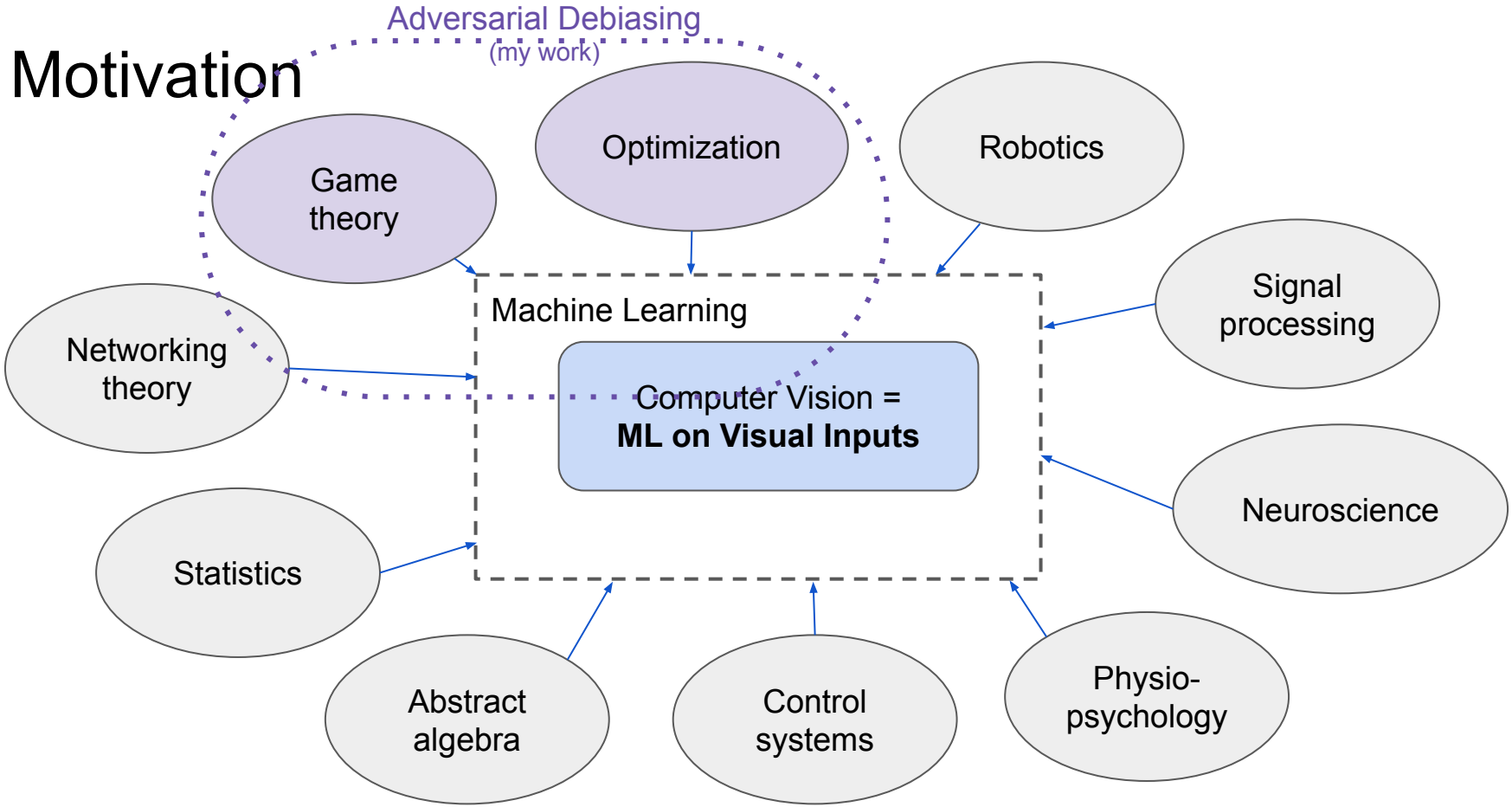
Motivation


Game theory
Optimization
Robotics
Networking theory
Signal processing
Computer Vision
Statistics
Neuroscience
Abstract algebra
Control systems
Physio-psychology

Motivation


Game theory
Optimization
Robotics
Networking theory
Machine Learning
Computer Vision = **ML on Visual Inputs**
Signal processing
Statistics
Neuroscience
Abstract algebra
Control systems
Physio-psychology

Motivation


Adversarial Debiasing
(my work)
Game theory
Optimization
Robotics
Networking theory
Machine Learning
Computer Vision =
ML on Visual Inputs
Signal processing
Neuroscience
Statistics
Abstract algebra
Control systems
Physio-psychology

Learning Objectives

By the end of this lecture, you should:

- Know how color is represented digitally
- Understand the structure of digital images and videos
- Begin thinking about images as points or functions
- Recognize the differences between kinds of image transforms
- Understand bilinear interpolation for image transformations

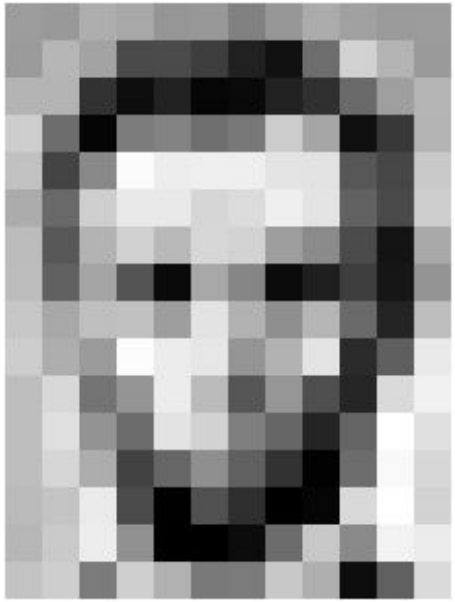
Images, Pixels, and Color

What is this an image of?

Hint: Higher numbers = whiter shades of gray

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Images, Pixels, and Color



Abraham Lincoln!

← Same information →

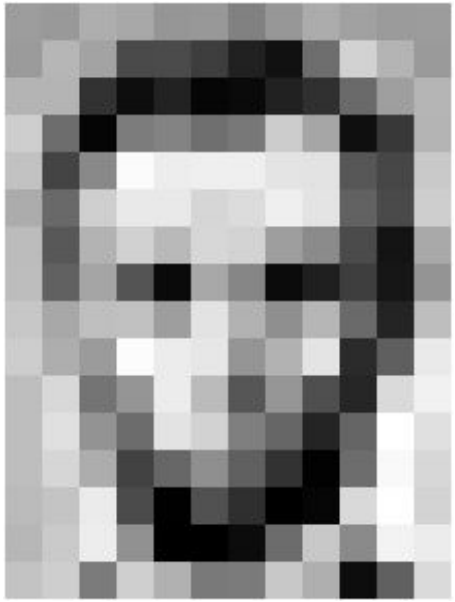
← Easy Hard →

Why?

“Semantic gap”

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Images, Pixels, and Color



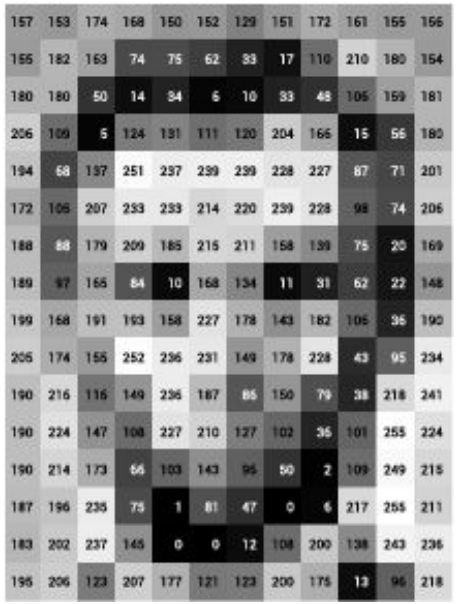
Our eyes and brains evolved to identify ← this efficiently.

Computers did not.

Vision tasks are hard because we are trying to mimic psychological tasks.
Always remember that "computer vision" is inherently about <u>human</u> vision.

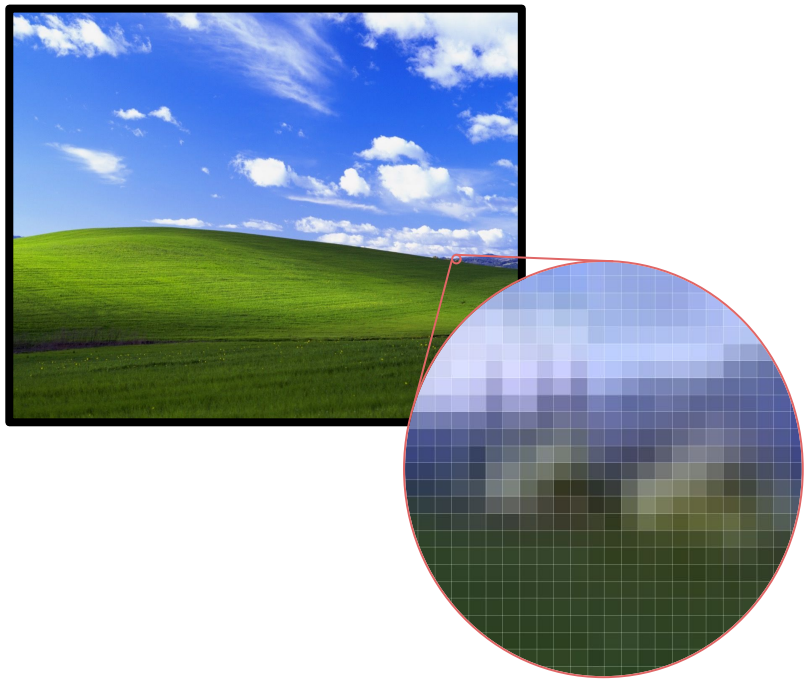
157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
196	206	123	207	177	121	123	200	175	13	96	218

Images, Pixels, and Color



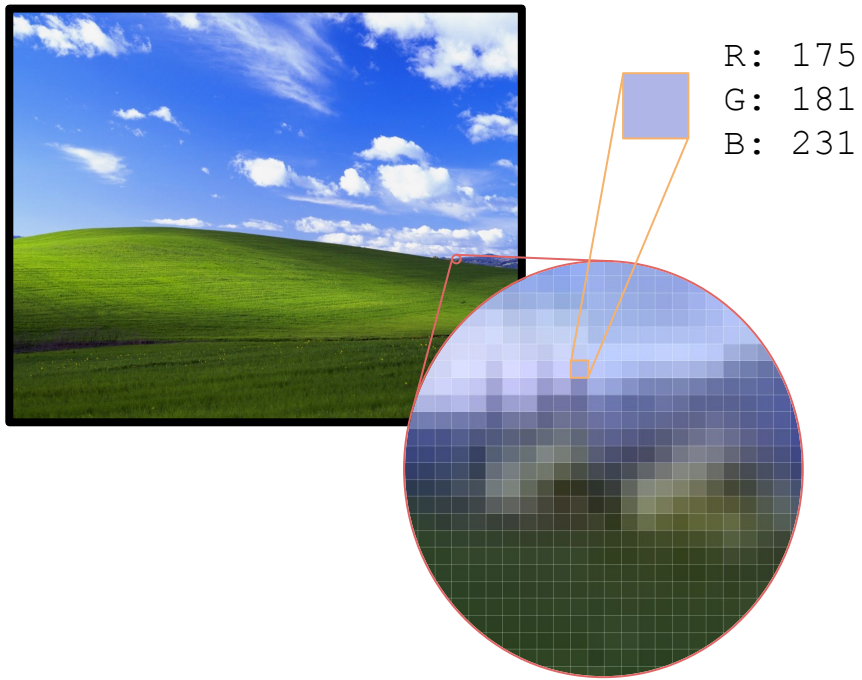
- **Digital image:** 2D array of pixels
- **Pixel:** “Picture element,” holds an atomic unit of visual data
    - Grayscale: 1D value, represents luminosity at a point

Images, Pixels, and Color

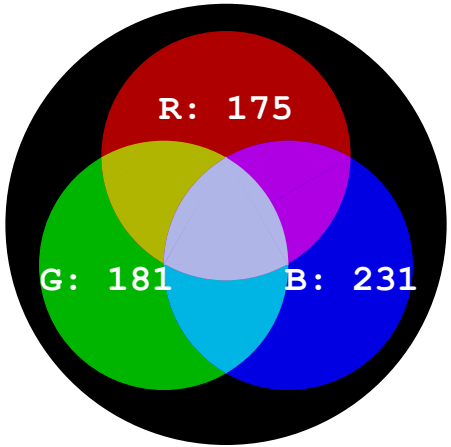


- **Digital image:** 2D array of pixels
- **Pixel:** “Picture element,” holds an atomic unit of visual data
  - Grayscale: 1D value, represents luminosity at a point
  - Color: 3D value, most commonly RGB

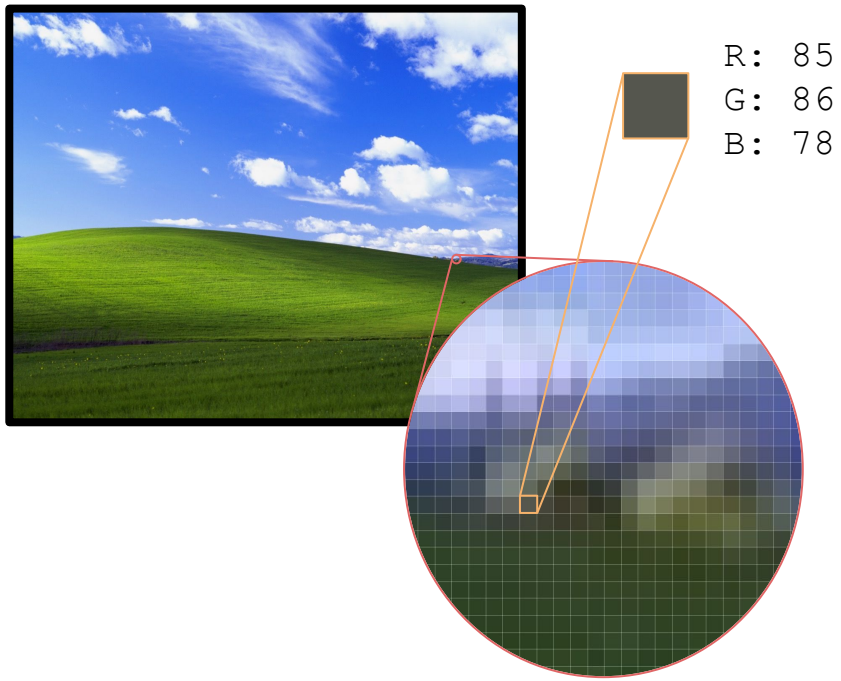
Images, Pixels, and Color



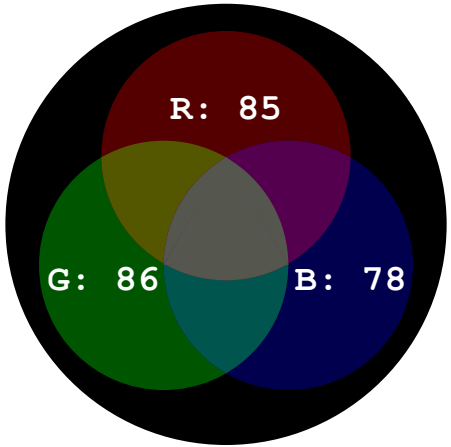
- **Digital image:** 2D array of pixels
- **Pixel:** “Picture element,” holds an atomic unit of visual data
  - Grayscale: 1D value
  - Color: 3D value



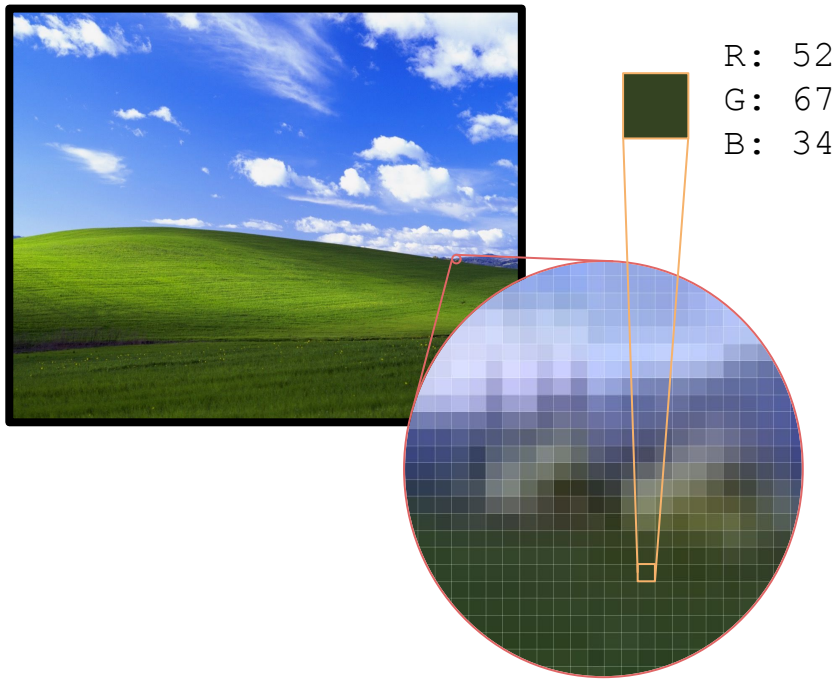
Images, Pixels, and Color



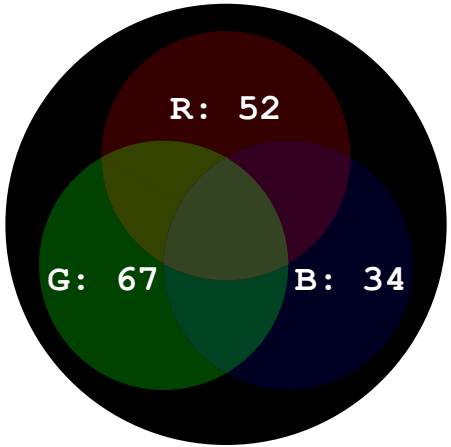
- **Digital image:** 2D array of pixels
- **Pixel:** “Picture element,” holds an atomic unit of visual data
  - Grayscale: 1D value
  - Color: 3D value



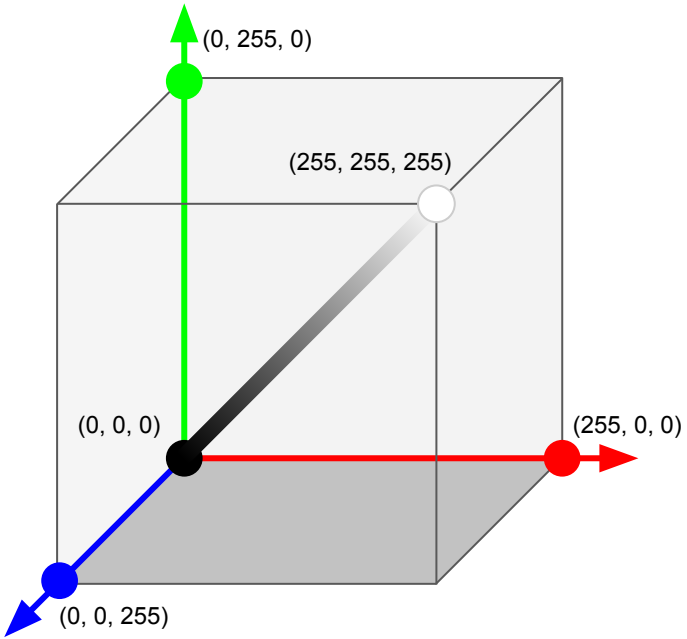
Images, Pixels, and Color



- **Digital image:** 2D array of pixels
- **Pixel:** “Picture element,” holds an atomic unit of visual data
  - Grayscale: 1D value
  - Color: 3D value

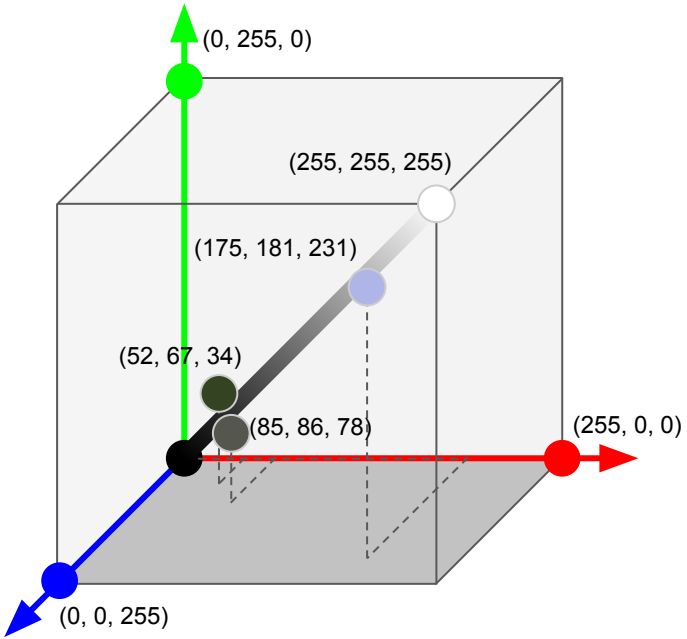


Images, Pixels, and Color



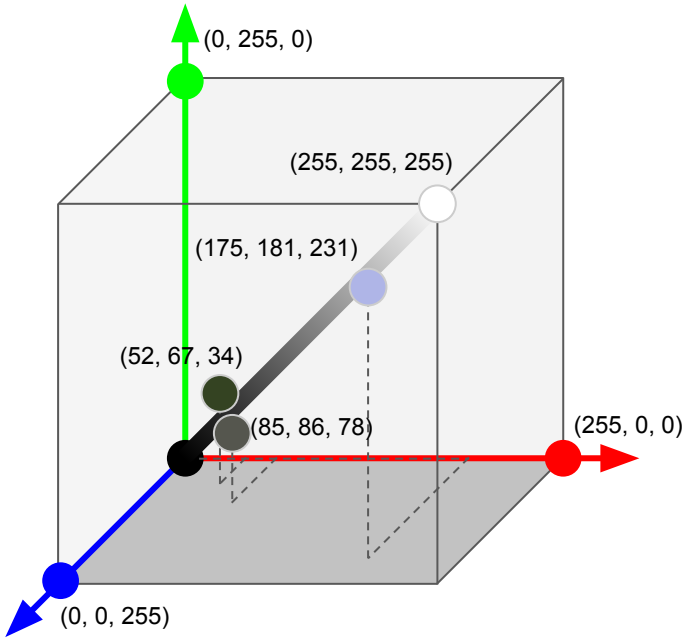
- RGB “color space”: Bounded 3D vector space
- Black = (0, 0, 0); White = (255, 255, 255)
- Gray tones = equal amounts of R, G, B

Images, Pixels, and Color



- RGB “color space”: Bounded 3D vector space
- Black = (0, 0, 0); White = (255, 255, 255)
- Gray tones = equal amounts of R, G, B

Images, Pixels, and Color



- RGB “color space”: Bounded 3D vector space
- Black = (0, 0, 0); White = (255, 255, 255)
- Gray tones = equal amounts of R, G, B

- Other color spaces exist!
- HSV: hue, saturation, value
- Lab: lightness, a/b
- CMYK: cyan, magenta, yellow, black

Images, Pixels, and Color

- Channel: A slice of an image, taking only the values from one dimension of the color space
- RGB images are three-channel; grayscale are single-channel

R

Images, Pixels, and Color

- Channel: A slice of an image, taking only the values from one dimension of the color space
- RGB images are three-channel; grayscale are single-channel

What's happening here?

R

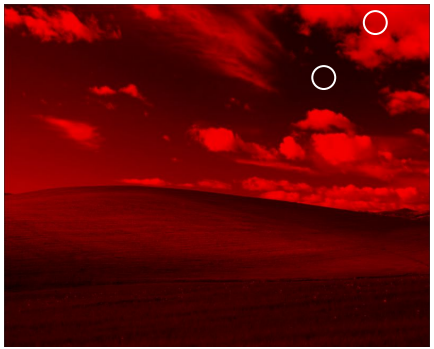
G

B

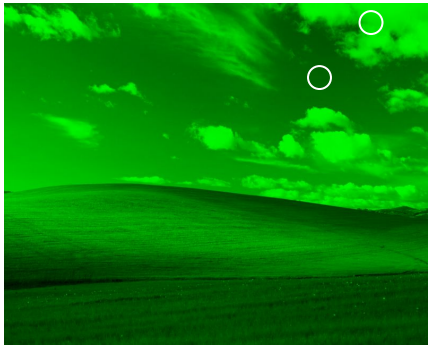
Images, Pixels, and Color

- Channel: A slice of an image, taking only the values from one dimension of the color space
- RGB images are three-channel; grayscale are single-channel

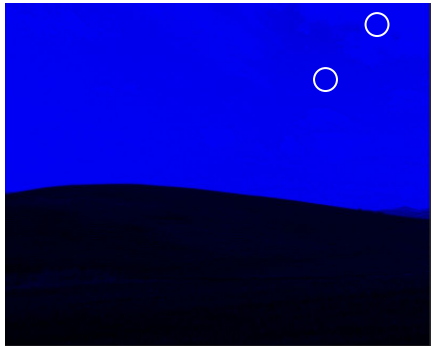
What's happening here?



R



G

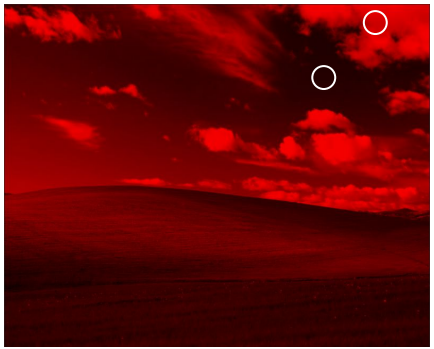


B

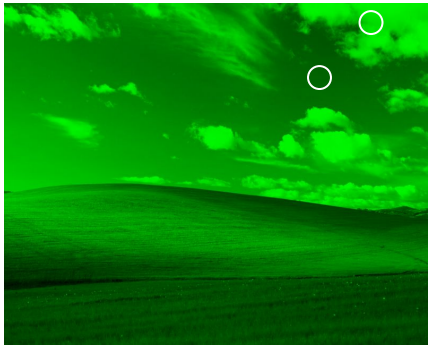
Images, Pixels, and Color

- Channel: A slice of an image, taking only the values from one dimension of the color space
- RGB images are three-channel; grayscale are single-channel

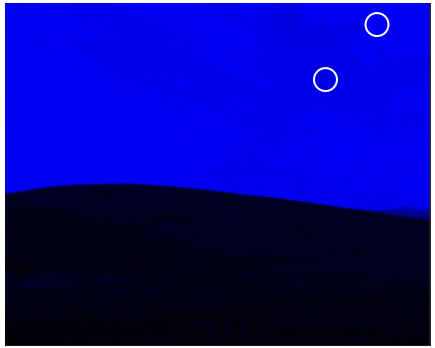
	Sky	Cloud	
R:	34	205	
G:	95	221	
B:	235	244	*Nearly equal!*



R



G



B

Video

- Video: Sequence of images (frames); 3D array of *voxels*
- Voxel: “Video pixel”
- More on video later in the course!

Learning Objectives

By the end of this lecture, you should:

- ~~Know how color is represented digitally~~
- ~~Understand the structure of digital images and videos~~
- Begin thinking about images as points or functions
- Recognize the differences between kinds of image transforms
- Understand bilinear interpolation for image transformations

Image Representations and Transforms

- How would you take the “average” of these images?

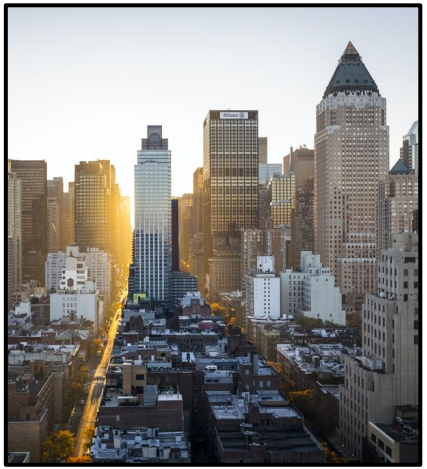
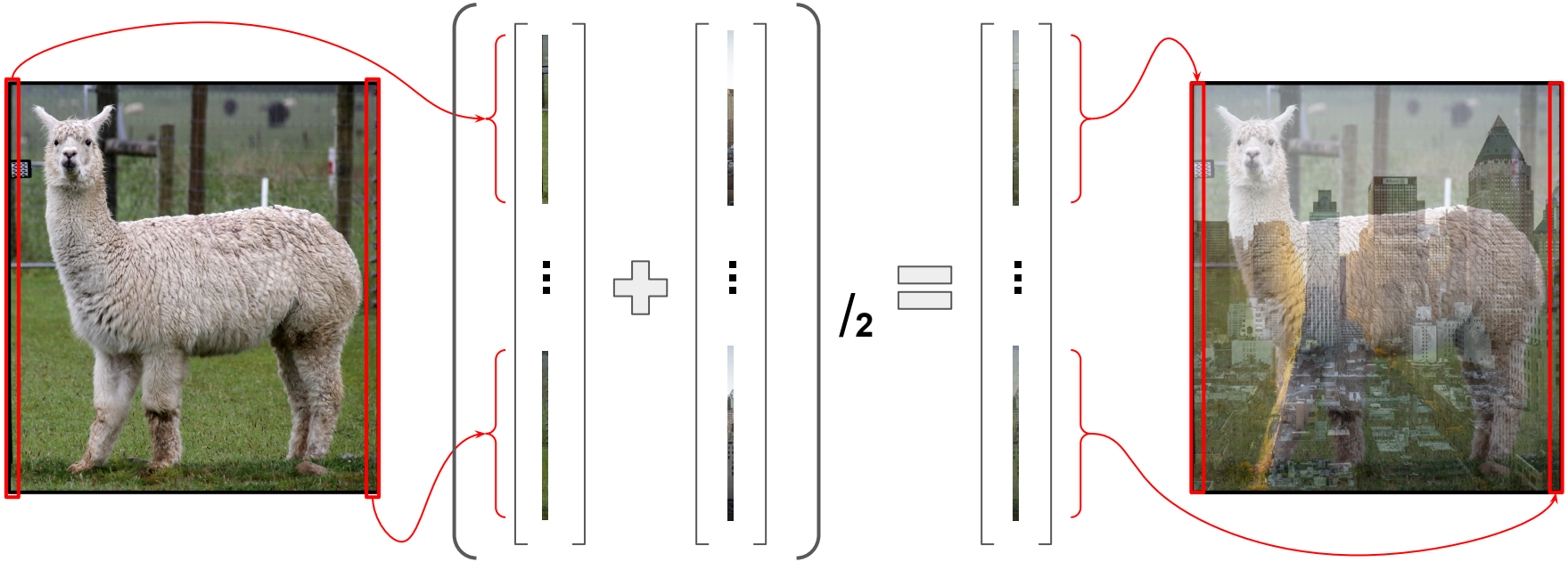


Image Representations and Transforms

- How would you take the “average” of these images?

Image Representations and Transforms

- How would you take the “average” of these ~~images~~ **vectors**?



“Vectorizing” or “flattening”

Image Space

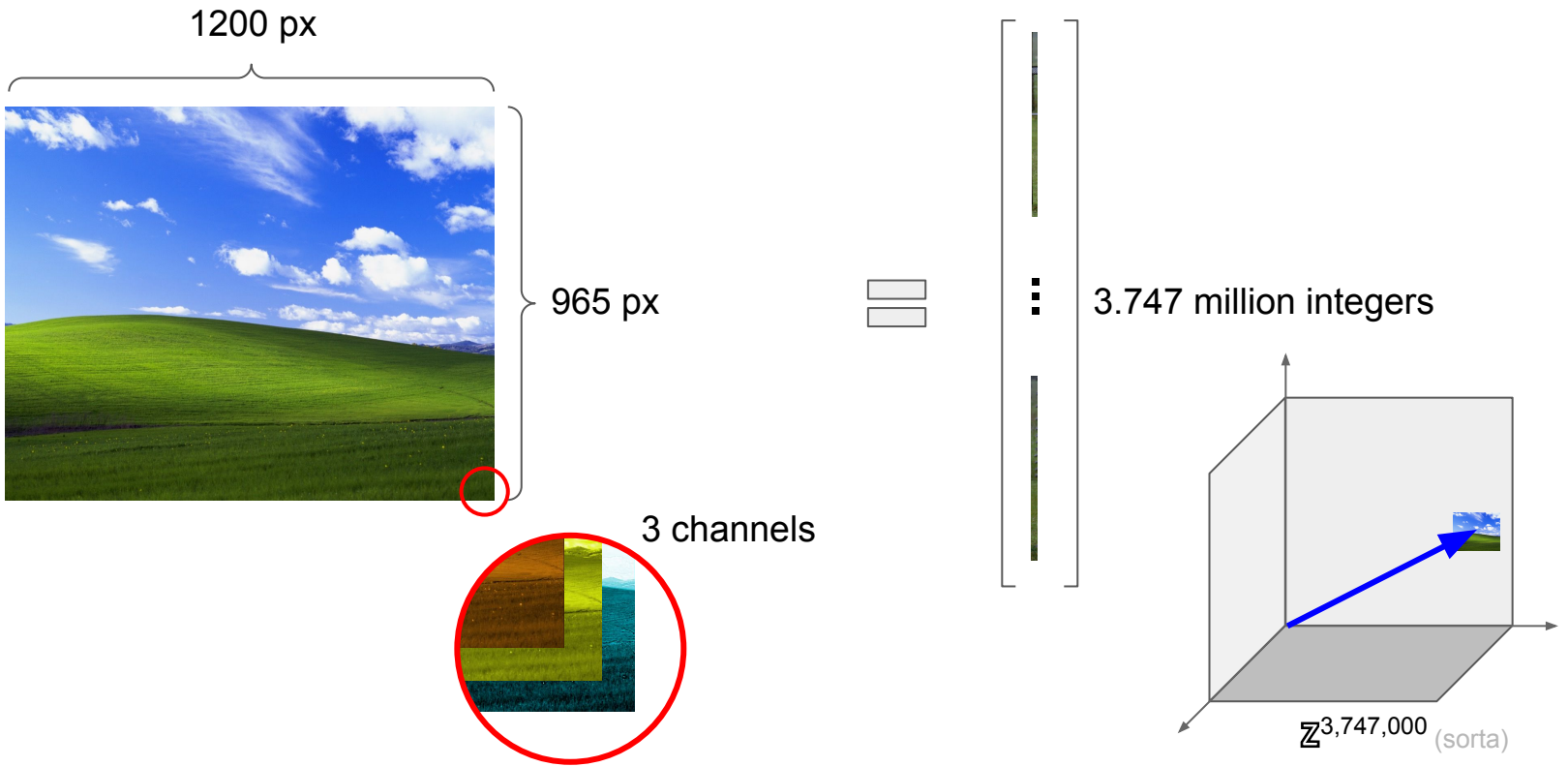

1200 px
965 px
3 channels
3.747 million integers
$\mathbb{Z}^{3,747,000}$ (sorta)

Image Space

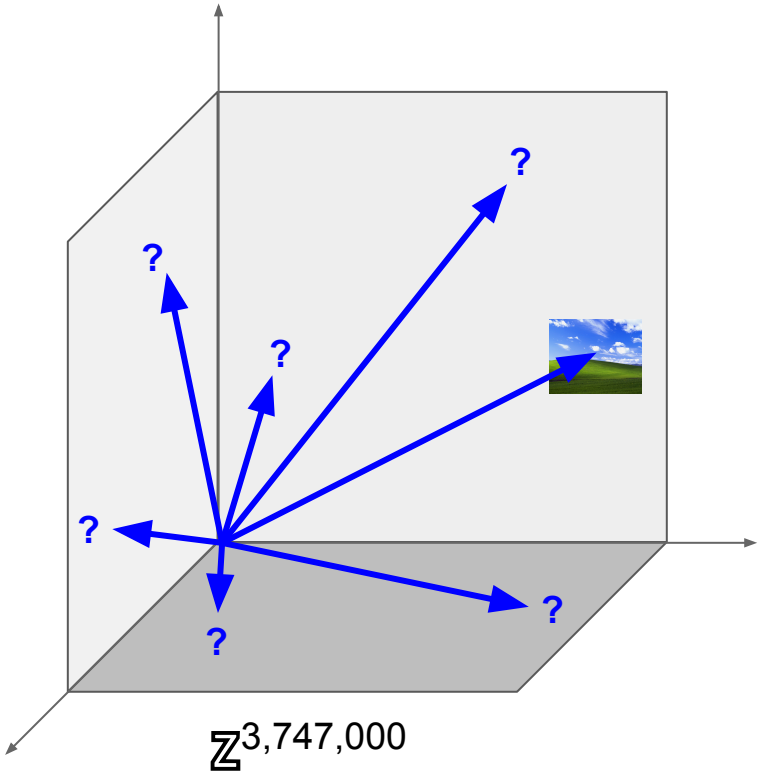


Image Space

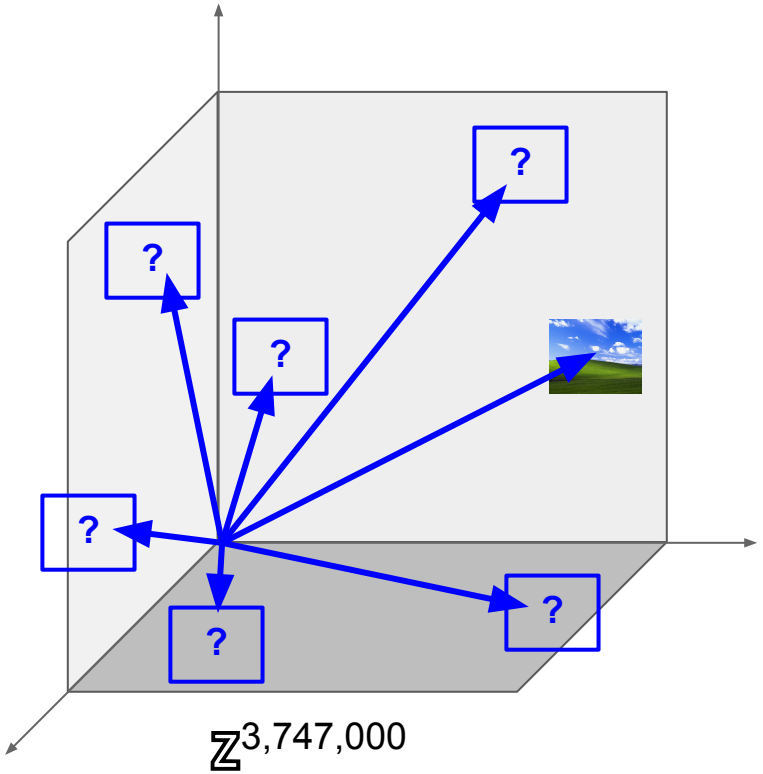
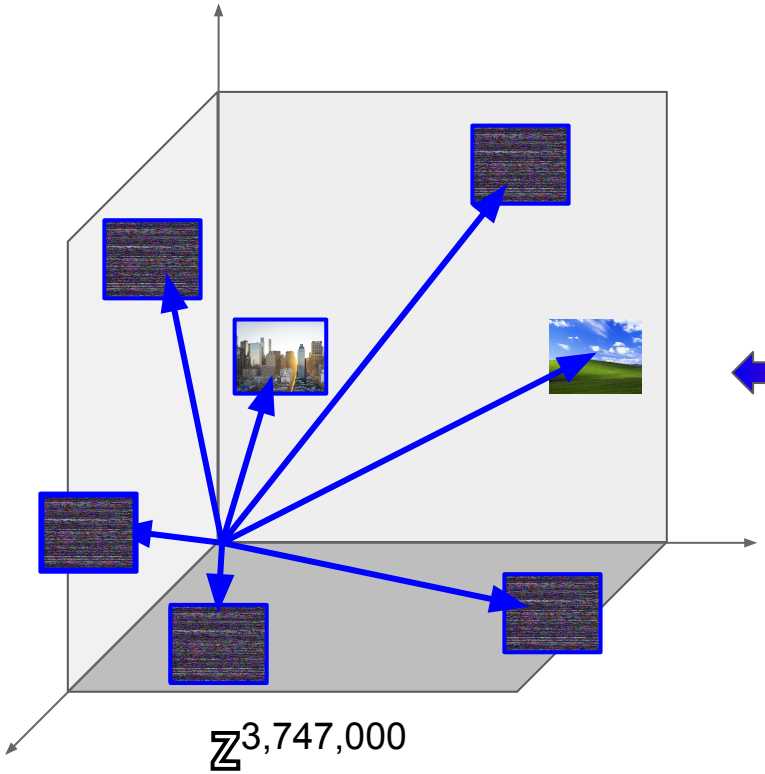


Image Space

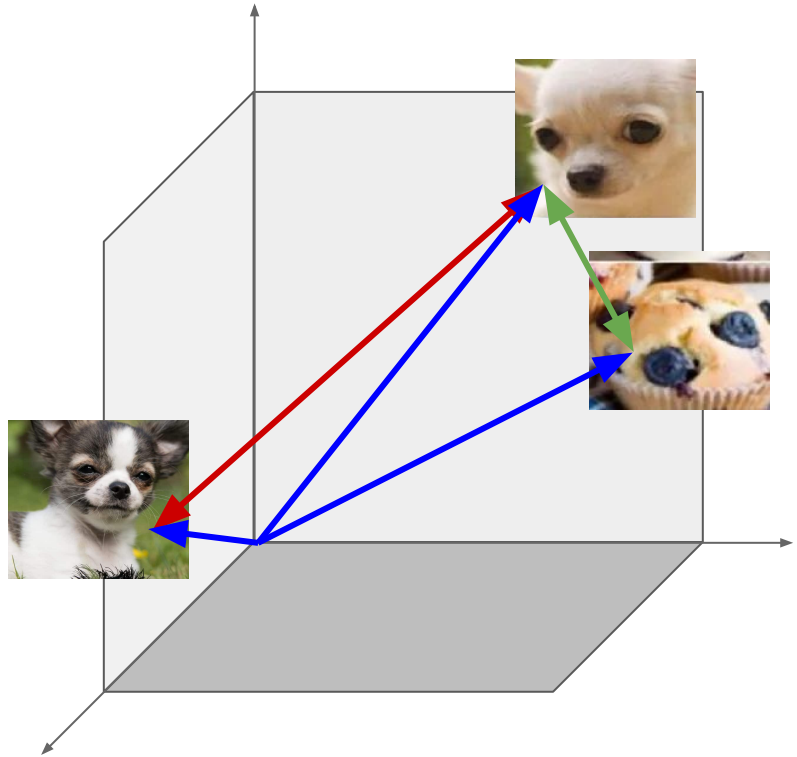


$\mathbb{Z}^{3,747,000}$

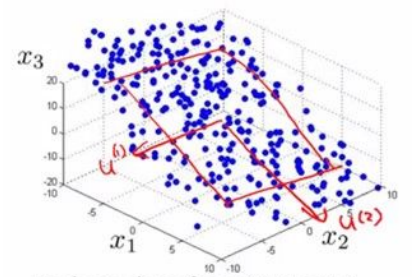
$\in \mathbb{Z}^{19,899,855}$

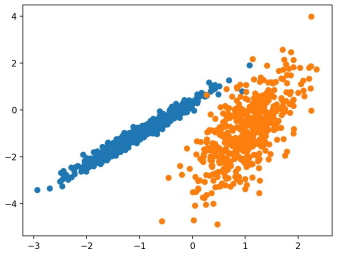
Image Vector Operations

- Distance metric
  - Pixelwise
  - Non-semantic
- PCA
- Clustering



Later in the course:



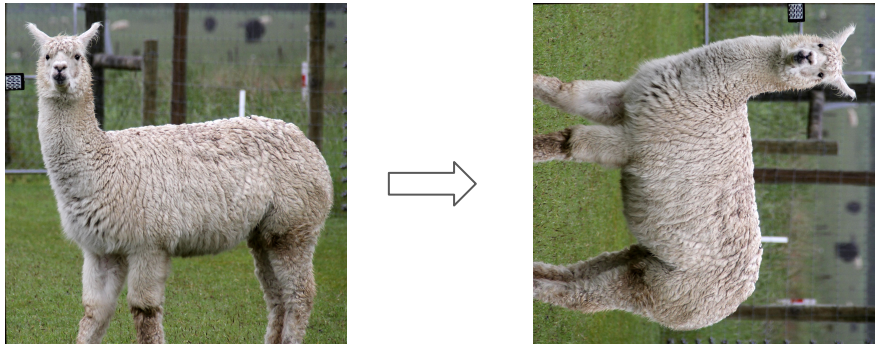


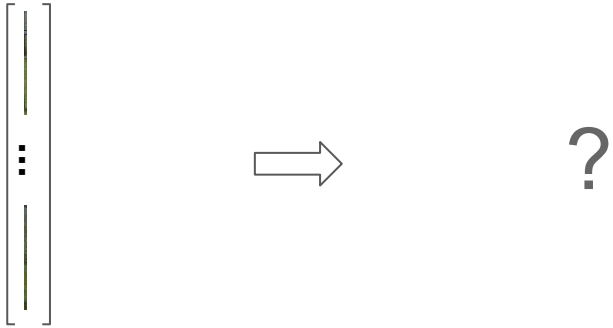
Images as Functions

Q: How do we rotate an image using its vector form?

A: Not easily.

We need to think of images as *functions* for this.





Images as Functions

Function: a mapping from an input space to an output space

Images map *pixel locations* to *pixel values* (colors)

$I(0,0) =$ = $[35, 33, 20]$

$I(4,3) =$ = $[45, 48, 31]$

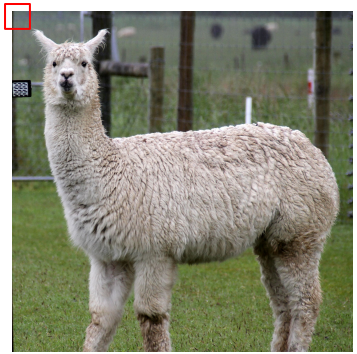
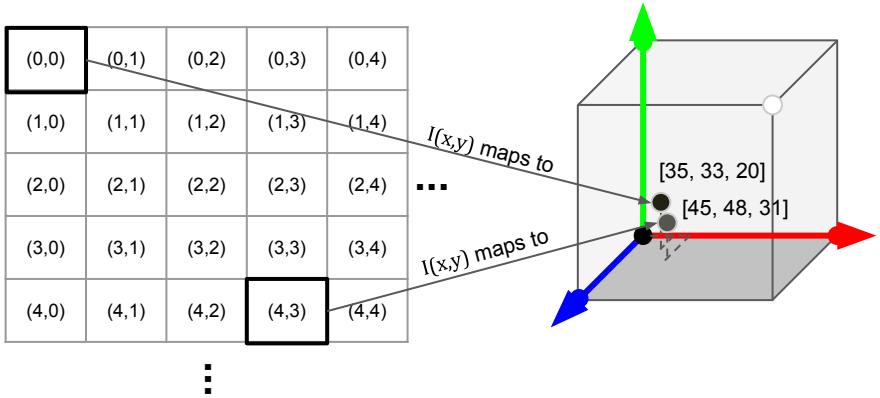


Image $I(x,y)$

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

Input space: $\mathbb{Z}^2$

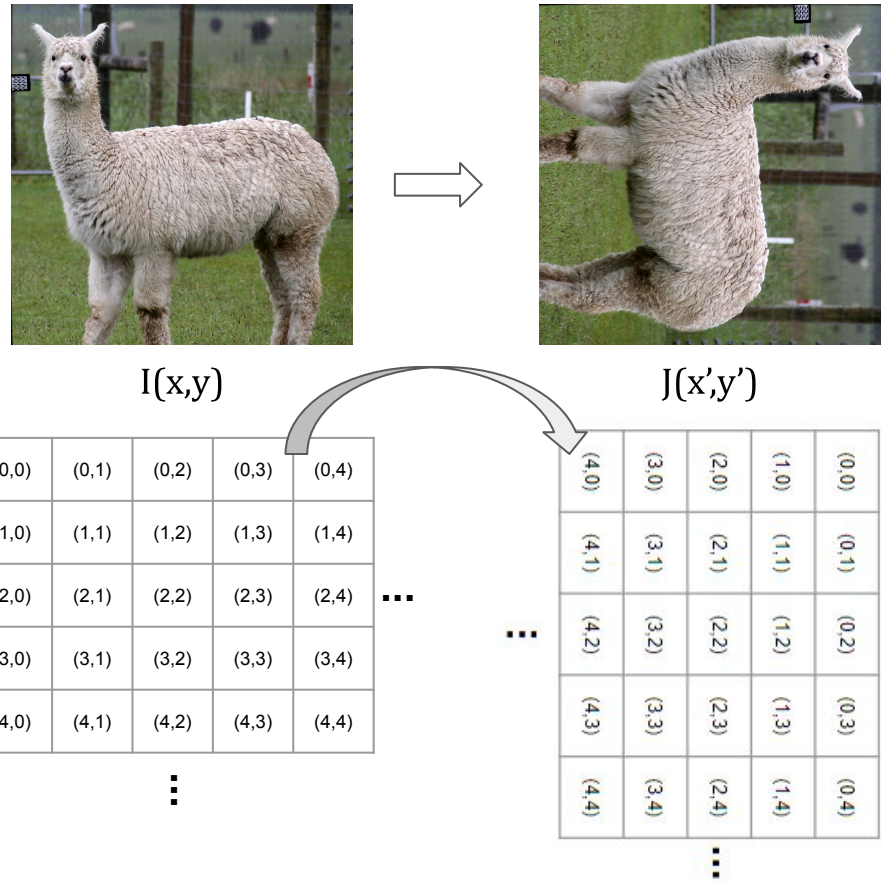
Output space: RGB



Images as Functions

Q: How do we rotate an image using its **function** form?

A: Rotate the pixel **locations**!



Domain transform: An operation on the input space of an image

- Rotation
- Translation
- Scaling (up/downsampling)
- Skew/perspective shift

Other Image Transforms

Range transforms: Operations on the pixel values, but not the locations

- Negative coloring
- Recoloring
- Brightness/contrast adjustment

Window transforms: Operations on a neighborhood of pixel values, but not locations

- Blurring
- Edge detection/highlighting

...and many that don't fit any of these categories!





Learning Objectives

By the end of this lecture, you should:

- ~~Know how color is represented digitally~~
- ~~Understand the structure of digital images and videos~~
- ~~Begin thinking about images as points or functions~~
- ~~Recognize the differences between kinds of image transforms~~
- Understand bilinear interpolation for image transformations

Interpolation (Warmup)

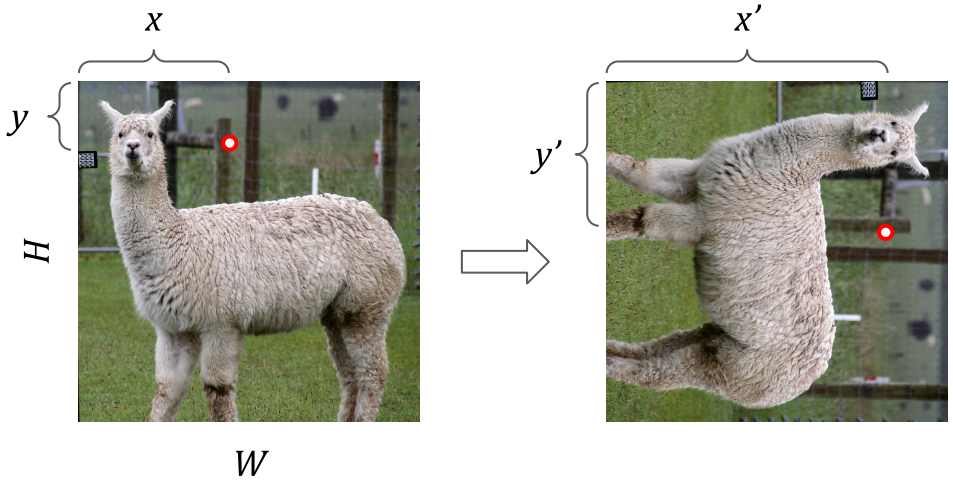
Find $J(x', y')$, the result of rotating $I(x, y)$ by 90 degrees clockwise.

Approach: Express (x, y) in terms of (x', y')

$$\begin{cases} x = & y' \\ y = & H - x' \end{cases}$$

Build $J(x', y')$ by looking up the corresponding pixels in $I(x, y)$:

$$J(x', y') = I(y', H - x')$$



General approach: Find inverse transformation, sample pixels from original image

$$J\begin{pmatrix} x' \\ y' \end{pmatrix} = I\left(T^{-1}\begin{pmatrix} x' \\ y' \end{pmatrix}\right)$$

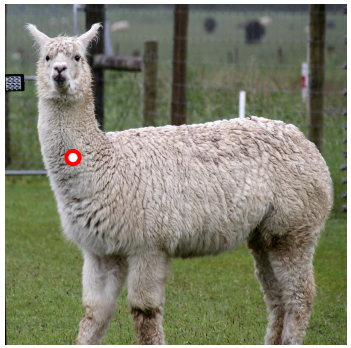
Interpolation

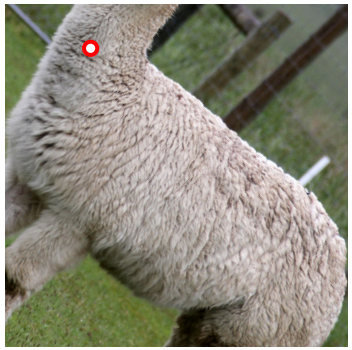
What if I asked you to rotate this image 45 degrees instead of 90?

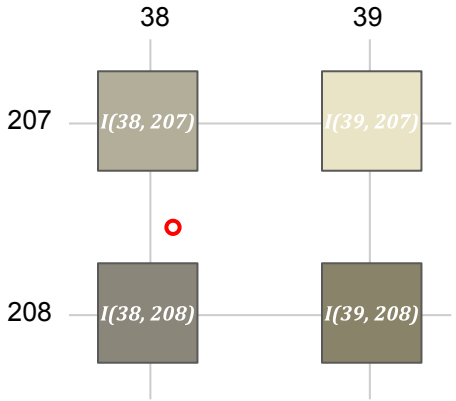
What if (x,y) aren't valid, integer pixel locations?

$$J(43, 17) = I(38.19, 207.54)$$

Approach: **Interpolate** between surrounding pixel values based on how close it is to each

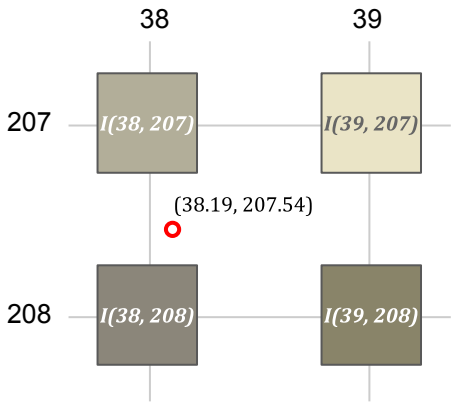




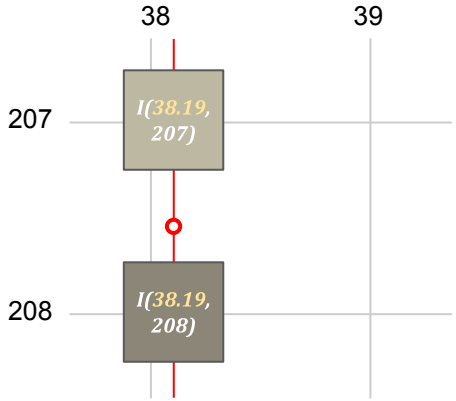


Interpolation

Step 1: Locate point to interpolate



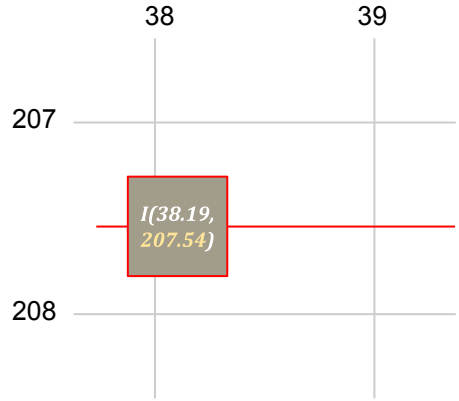
Step 2: Interpolate along one axis



$I(38.19, 207) = 0.81 * I(38, 207) + 0.19 * I(39, 207)$

$I(38.19, 208) = 0.81 * I(38, 208) + 0.19 * I(39, 208)$

Step 3: Interpolate again



$I(38.19, 207.54) = 0.46 * I(38.19, 207) + 0.54 * I(38.19, 208)$

Learning Objectives

By the end of this lecture, you should:

- ~~Know how color is represented digitally~~
- ~~Understand the structure of digital images and videos~~
- ~~Begin thinking about images as points or functions~~
- ~~Recognize the differences between kinds of image transforms~~
- ~~Understand bilinear interpolation for image transformations~~