Introduction

Motivation and Image Processing

CSC420
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/420
Slide credit: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

Toronto Computational Imaging Group

Instructors

David Lindell

Shayan
Shekarforoush

Yun-Chun Chen

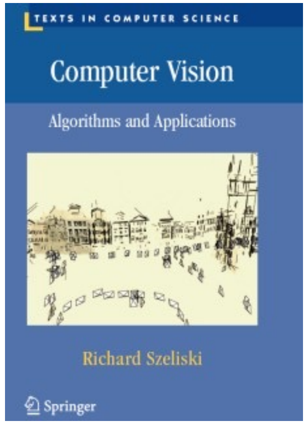
Anagh Malik

Course Email: csc420-2022-09@cs.toronto.edu

Course Info

- Class time: Mondays 2-4 pm GB 248
- Tutorials: Wednesdays 3-4 pm GB 119
- Class Website: https://www.cs.toronto.edu/~lindell/teaching/420/
- Quercus: https://q.utoronto.ca/
- Course material (lecture notes, reading material, assignments, announcements, etc.) will be posted on Quercus
- Forum: Ed Discussion (link on Quercus)
- Your grade will not depend on your participation on discussions. It's just a good way for asking questions, discussing with your instructor, TAs and your peers

Textbook: We won't directly follow any book, but extra reading in this textbook will be useful:



Rick Szeliski

Computer Vision: Algorithms and Applications

available free online: http://szeliski.org/Book/

Links to other material (papers, code, etc.) will be posted on the class webpage

Course Prerequisites

- Data structures
- Linear Algebra
- Vector calculus
- Without this you'll need some serious catching up to do!

Knowing some basics in these is a plus:

- Python
- Machine Learning
- Neural Networks
- (Solving assignments sooner rather than later)

Grading

- Assignment 1: 12%
- Assignment 2: 20%
- Assignment 3: 16%
- Assignment 4: 16%
- Ethics Module: 1% (2 surveys, 0.5 each)
- Final Exam: 34%

- Assignments: They will consist of problem sets and programming problems with the goal of deepening your understanding of the material covered in class.

Assignments

- Download from Files section on Quercus, Submitted via MarkUs
- Assignments: They will consist of problem sets and programming problems with the goal of deepening your understanding of the material covered in class.
    - Code in python
    - Please comment your code!
- Assignment 1 is out now, due Jan 27 at 11:59 PM

Assignments

Deadline

- The solutions to the assignments / project should be submitted by 11:59 pm on the date they are due.

Lateness

- Each student will be given a total of 3 free late days.
- This means that you can hand in three of the assignments one day late, or one assignment three days late.
- After you have used the 3 day budget, late assignments will not be accepted.

All info on the course website

Schedule and Syllabus

Week	Date	Description	Material	Readings	Event	Deadline
Week 1	Mon 9/1	**Lecture 1: Introduction & Linear filters**	[slides]	Szeliski 3.2 (optional) Brain mechanisms of early vision (optional) Early vision	**Assignment 1 out on Quercus**	
	Wed 11/1	**Tutorial 1**	[code]			
	Thu 12/1	**TA Office Hours**				
Week 2	Mon 16/1	**Lecture 2: Edges**	[slides]	Szeliski 4.2 (optional) Fourier Transform (optional) Computer color is broken		
	Wed 18/1	**Tutorial 2**	[code]			
	Thu 19/1	**TA Office Hours**				
Week 3	Mon 23/1	**Lecture 3: Image pyramids**	[slides]	Szeliski 3.5 (optional) Pyramid methods		
	Wed 25/1	**Tutorial 3**	[code]			
	Thu 26/1	**TA Office Hours**				
	Fri 27/1					Assignment 1 due at 11:59pm

Let’s begin!

Introduction to Intro to Image Understanding

- What is Computer Vision?
- Why study Computer Vision?
- Which cool applications can we do with it? Is vision a hard problem?

What is Computer Vision?

What is computer vision?

- A field trying to develop automatic algorithms that can “see”

What is computer vision?

- What does it mean to see?

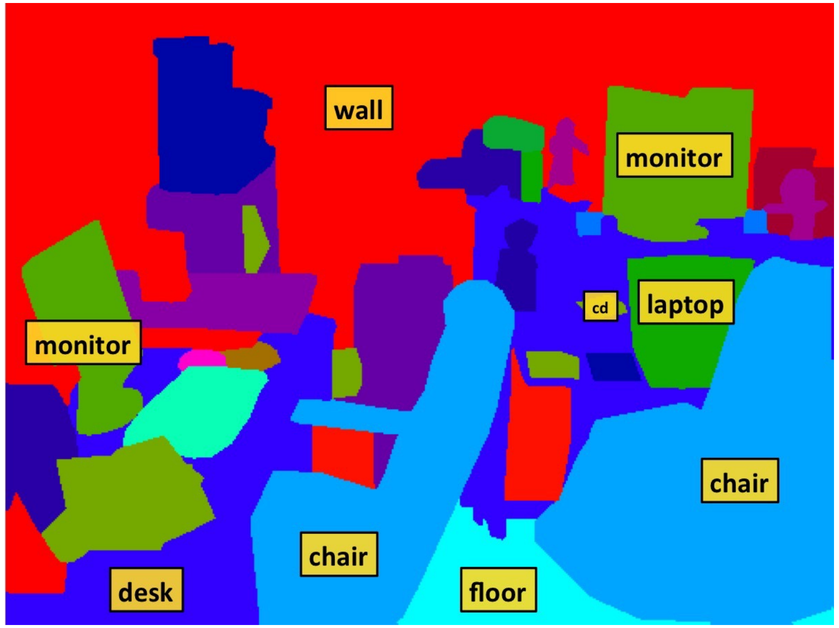
example scene

[adapted from A. Torralba]

What is computer vision?

- What does it mean to see?
  - To know what is where by looking – Marr, 1982
  - Understand where things are in the world

example scene

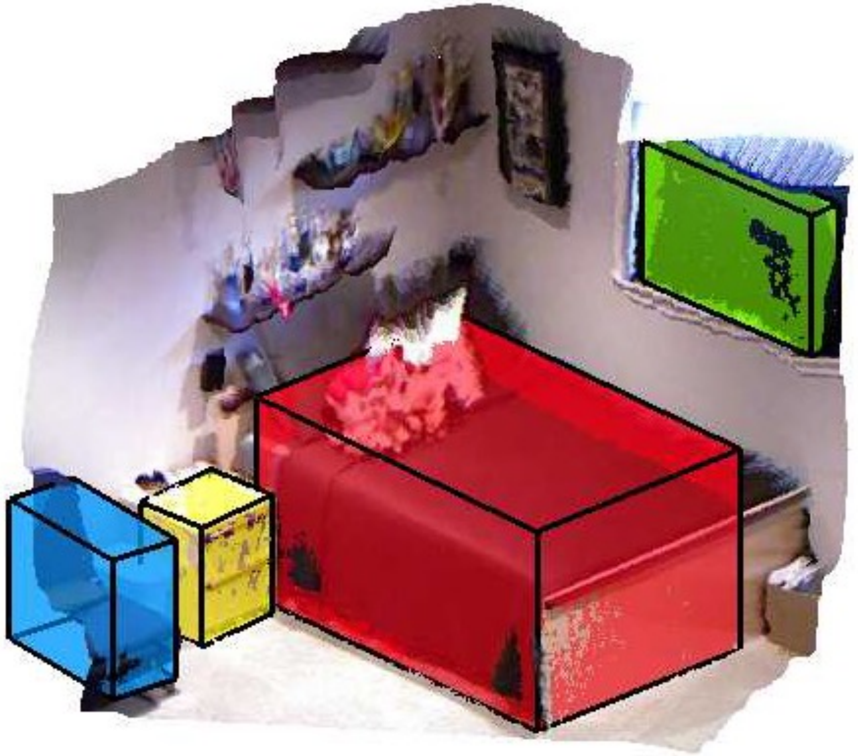


segmentation

[adapted from A. Torralba]

What is computer vision?

- What does it mean to see?
  - To know what is where by looking – Marr, 1982
  - Understand where things are in the world
  - Understand 3D structure



[adapted from A. Torralba]

What is computer vision?

- What does it mean to see?
  - To know what is where by looking – Marr, 1982
  - Understand where things are in the world
  - Understand 3D structure
  - Understand physical properties

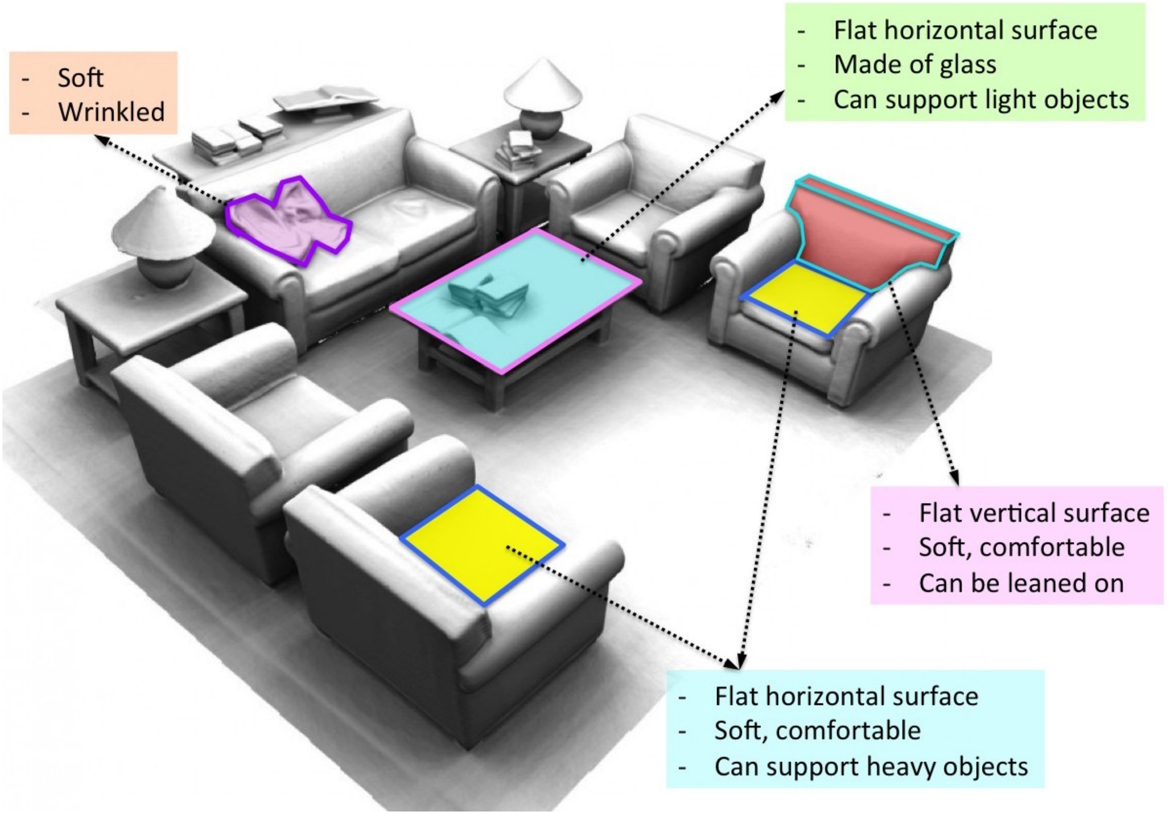


Image: Vladlen Koltun

What is computer vision?

- What does it mean to see?
  - To know what is where by looking – Marr, 1982
  - Understand where things are in the world
  - Understand 3D structure
  - Understand physical properties
  - Understand what actions are taking place

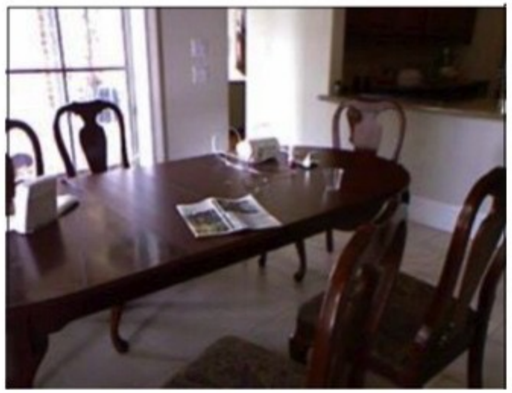
Image: www.cobblehillpuzzles.com

What is computer vision?

- Full understanding of an image?

What is computer vision?

- Full understanding of an image?
  - Can answer any question about it



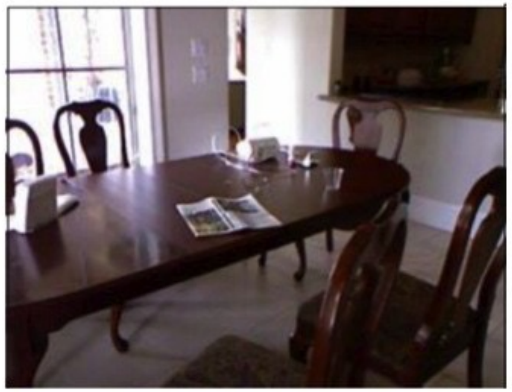
Q: What is behind the table?
A: window

Q: What is in front of the toilet?
A: door

Q: What is on the counter in the corner? A: microwave

What is computer vision?

- Full understanding of an image?
    - Can answer any question about it



Q: What is behind the table?
A: window

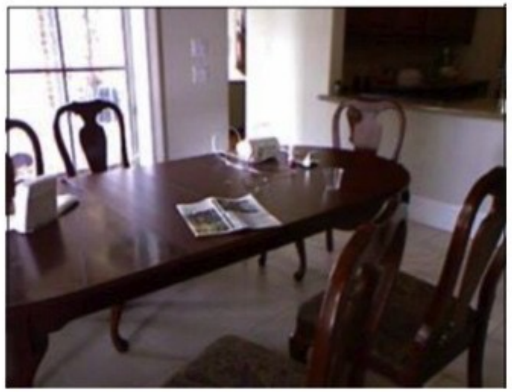
Q: What is in front of the toilet?
A: door

Q: What is on the counter in the corner? A: microwave

Q: What is the shape of the green chair? A: horse shaped

What is computer vision?

- Full understanding of an image?
    - Can answer any question about it



Q: What is behind the table?
A: window

Q: What is in front of the toilet?
A: door

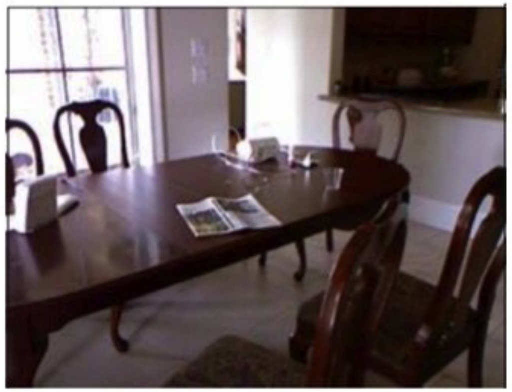
Q: What is on the counter in the corner? A: microwave

Q: What is the shape of the green chair? A: horse shaped

Q: Where is the oven?
A: on the right side of the fridge

What is computer vision?

- Full understanding of an image?
  - Can answer any question about it



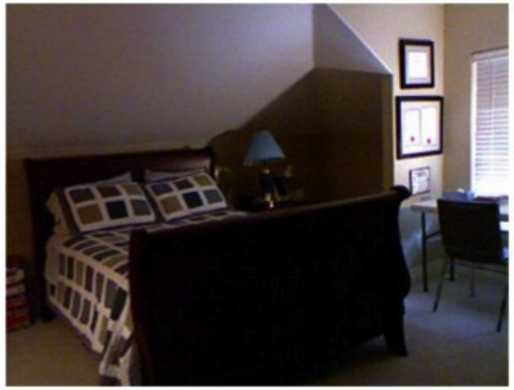
Q: What is behind the table?
A: window

Q: What is in front of the toilet?
A: door

Q: What is on the counter in the corner? A: microwave

Q: What is the shape of the green chair? A: horse shaped

Q: Where is the oven?
A: on the right side of the fridge



Q: What is the largest object?
A: bed

What is computer vision?

- Full understanding of an image?
  - Can answer any question about it

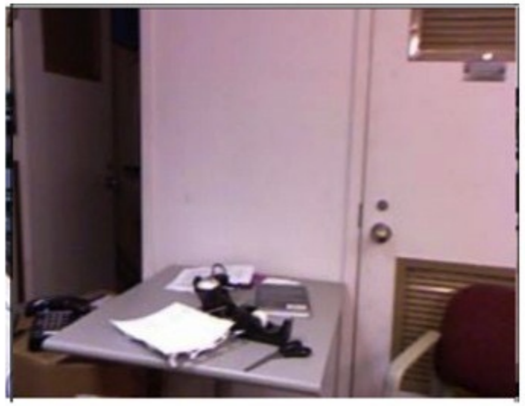
Q: Which object is red?
A: toaster

What is computer vision?

- Full understanding of an image?
  - Can answer any question about it

Q: Which object is red?
A: toaster

Q: How many drawers are there?
A: 6



Q: How many doors are open
A: 1

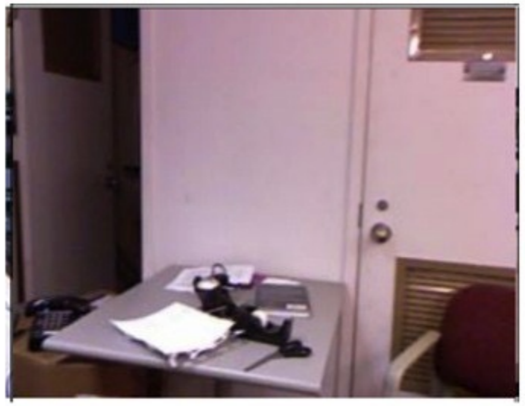
Q: How many lights are on?
A: 6

What is computer vision?

- Full understanding of an image?
  - Can answer any question about it

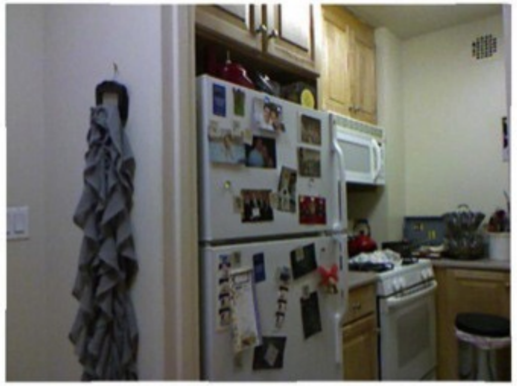
Q: Which object is red?
A: toaster

Q: How many drawers are there?
A: 6



Q: How many doors are open
A: 1

Q: How many lights are on?
A: 6



Q: Can you make pizza in this room?
A: yes

Q: Where can you sit?
A: chairs, table, floor

Why study computer vision?

Why study Computer Vision?

Because you want your robot to fold your laundry

Why study Computer Vision?

And drive you to work

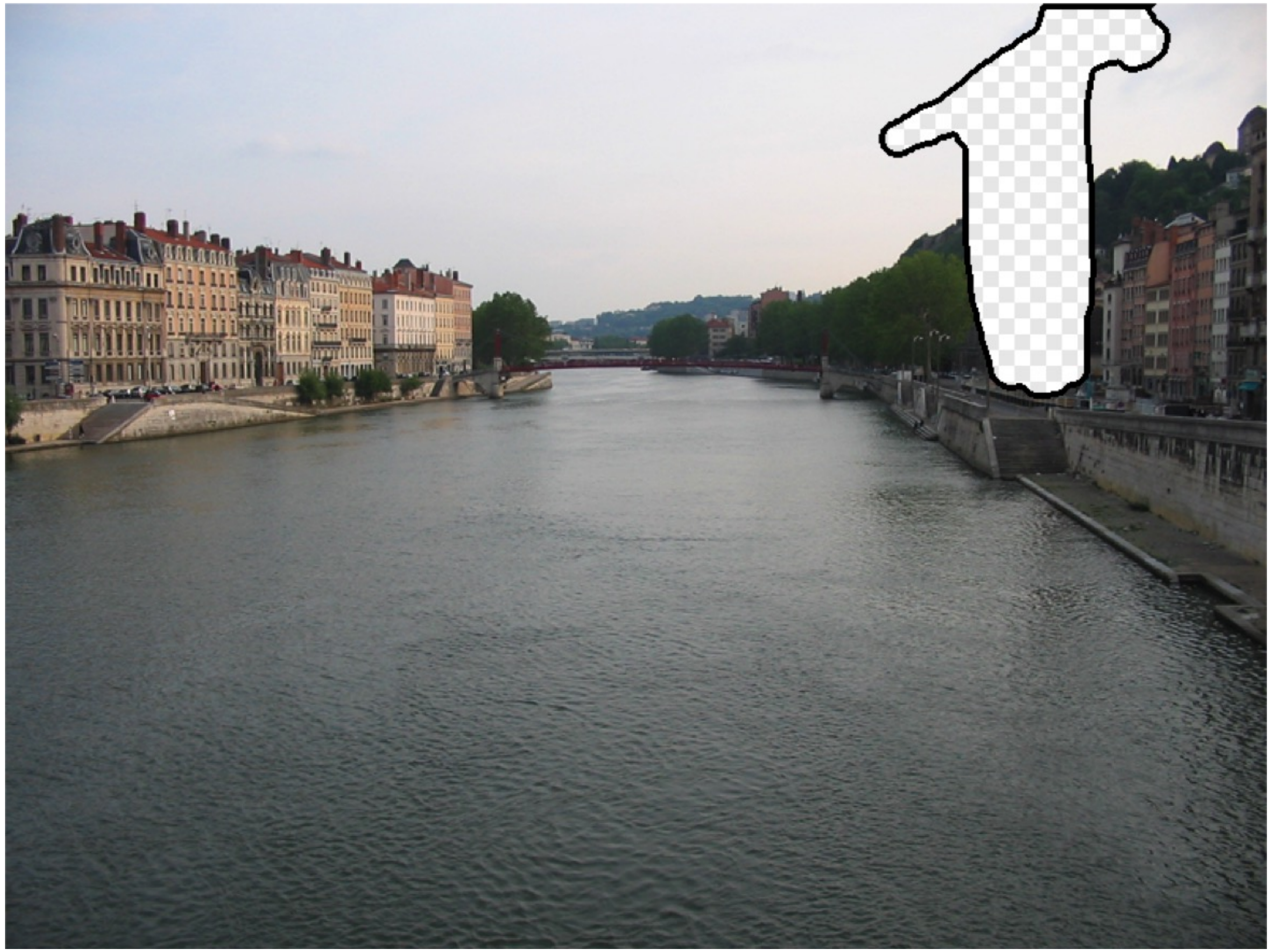
Why study Computer Vision?

Allows you to manipulate images

Scene Completion using Millions of Photographs, Hays & Efros, SIGGRAPH 2007

Why study Computer Vision?

Allows you to manipulate images



Scene Completion using Millions of Photographs, Hays & Efros, SIGGRAPH 2007

Why study Computer Vision?

Allows you to manipulate images

Scene Completion using Millions of Photographs, Hays & Efros, SIGGRAPH 2007

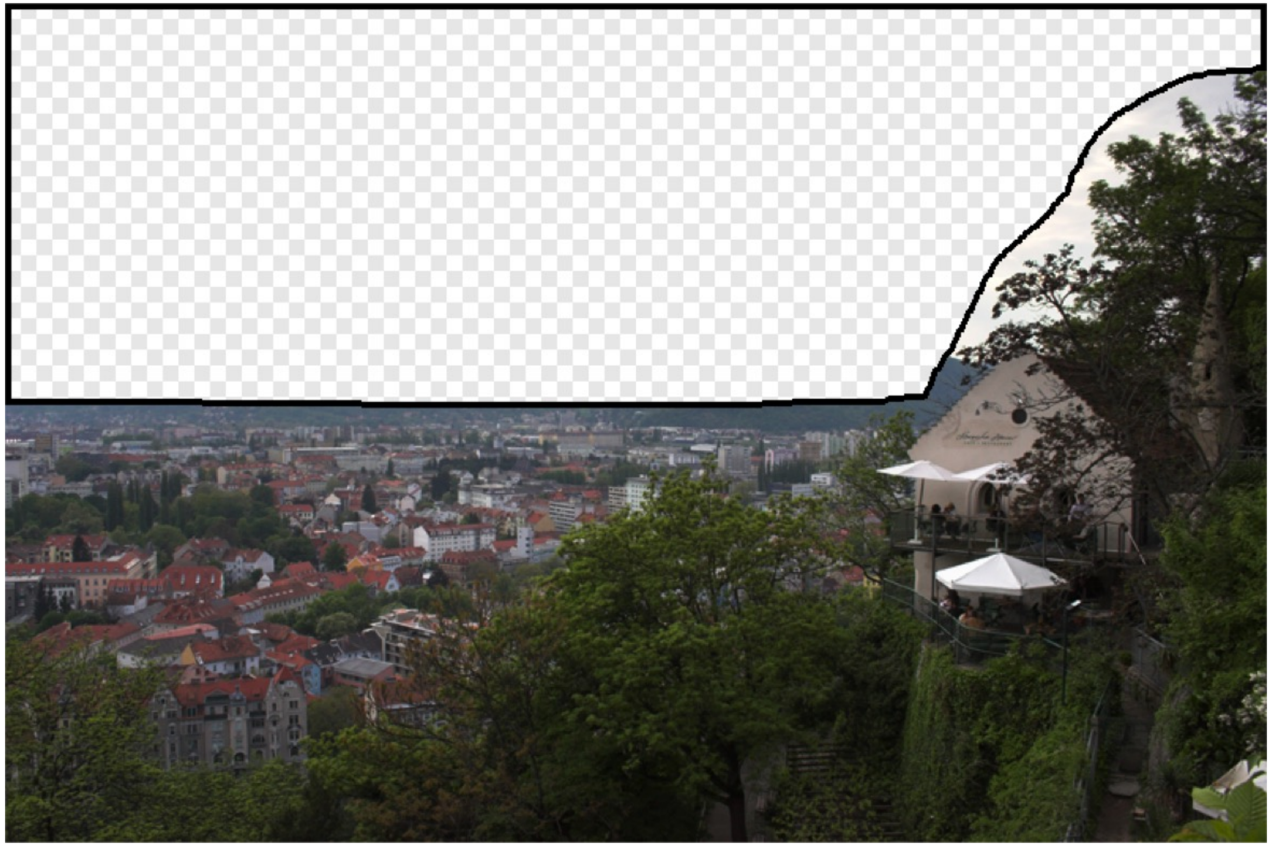
Why study Computer Vision?

Allows you to manipulate images

Scene Completion using Millions of Photographs, Hays & Efros, SIGGRAPH 2007

Why study Computer Vision?

Allows you to manipulate images



Scene Completion using Millions of Photographs, Hays & Efros, SIGGRAPH 2007

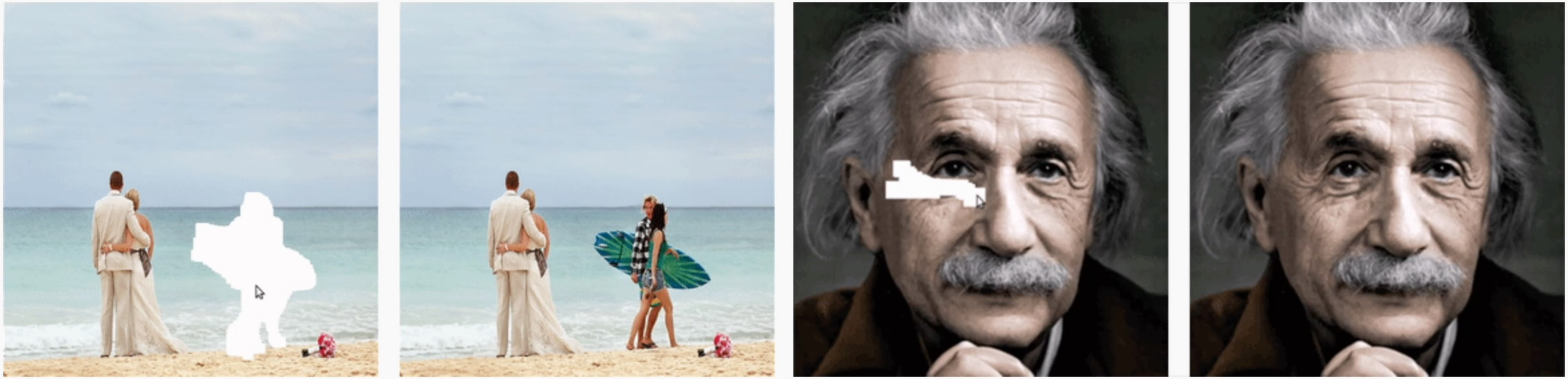
Why study Computer Vision?

Allows you to manipulate images

Scene Completion using Millions of Photographs, Hays & Efros, SIGGRAPH 2007

Why study Computer Vision?

Allows you to manipulate images



Online demo (NVIDIA inpainting demo)

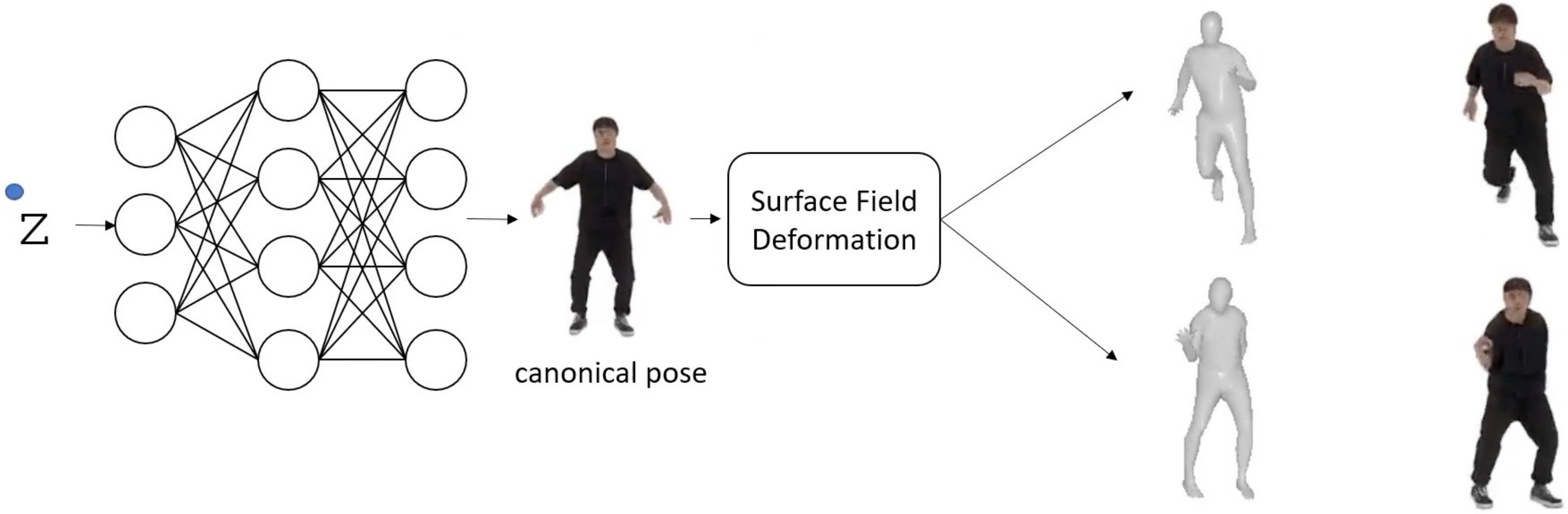
Why study Computer Vision?

Change style of images...

[Gatys, Ecker, Bethge. A Neural Algorithm of Artistic Style. Arxiv'15.]

Why study Computer Vision?

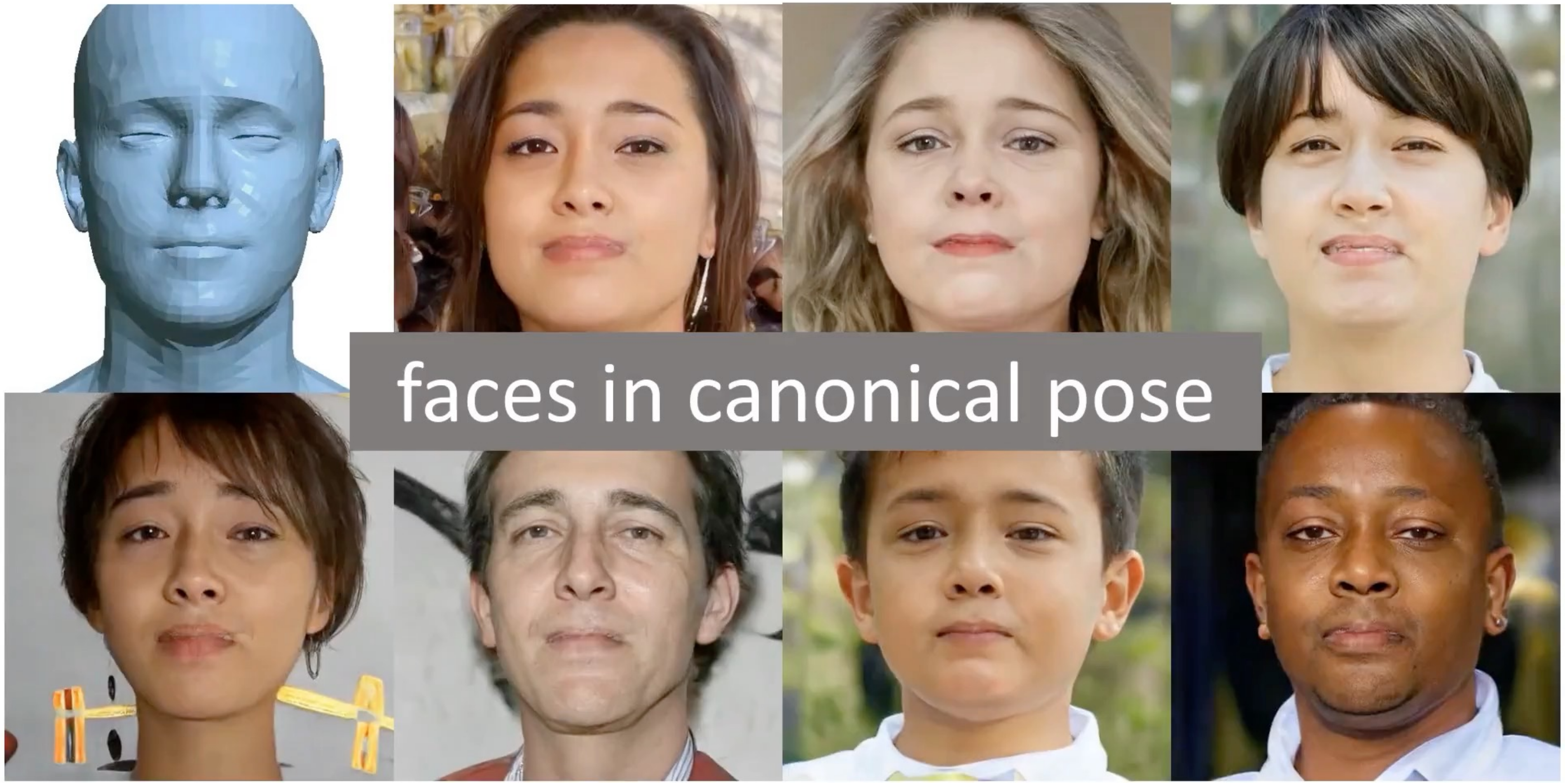
Synthesize and animate digital humans



[Bergman et al. '22]

Why study Computer Vision?

Synthesize and animate digital humans



[Bergman et al. '22]

Why study Computer Vision?

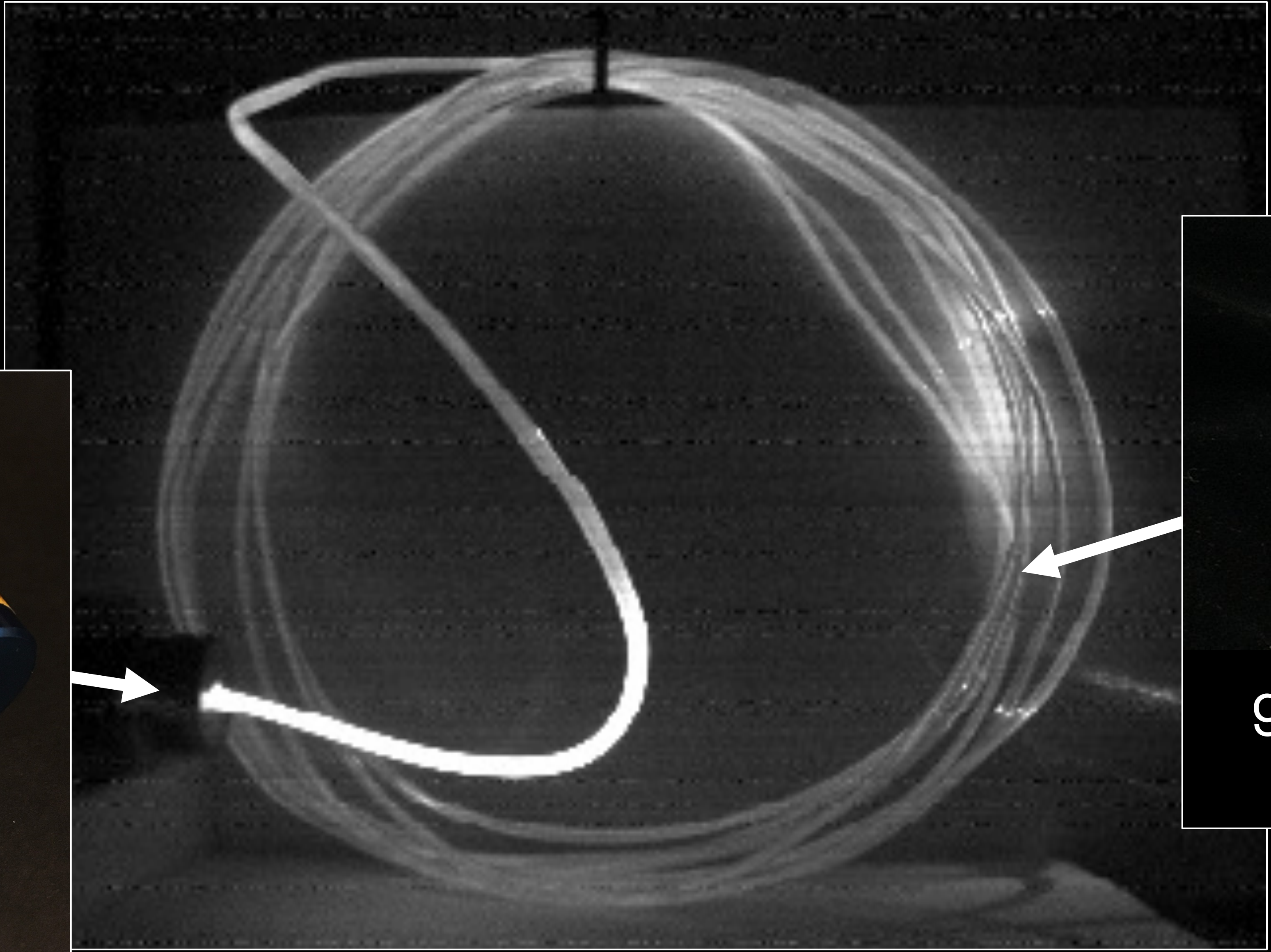
See "invisible" changes in a scene...

[Wu et al. SIGGRAPH '12]

Why study Computer Vision?

See "invisible" changes in a scene...

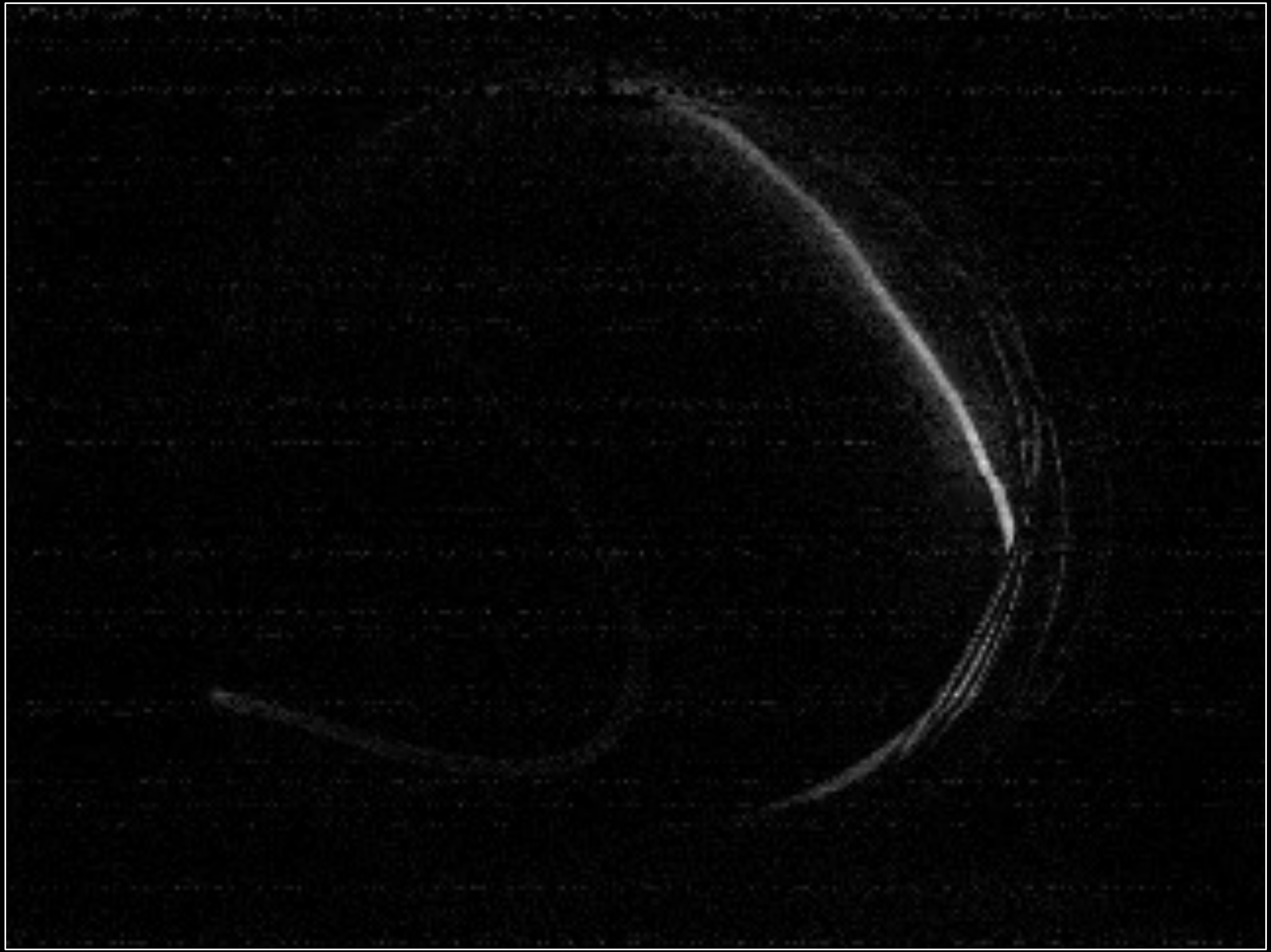
[Wu et al. SIGGRAPH '12]



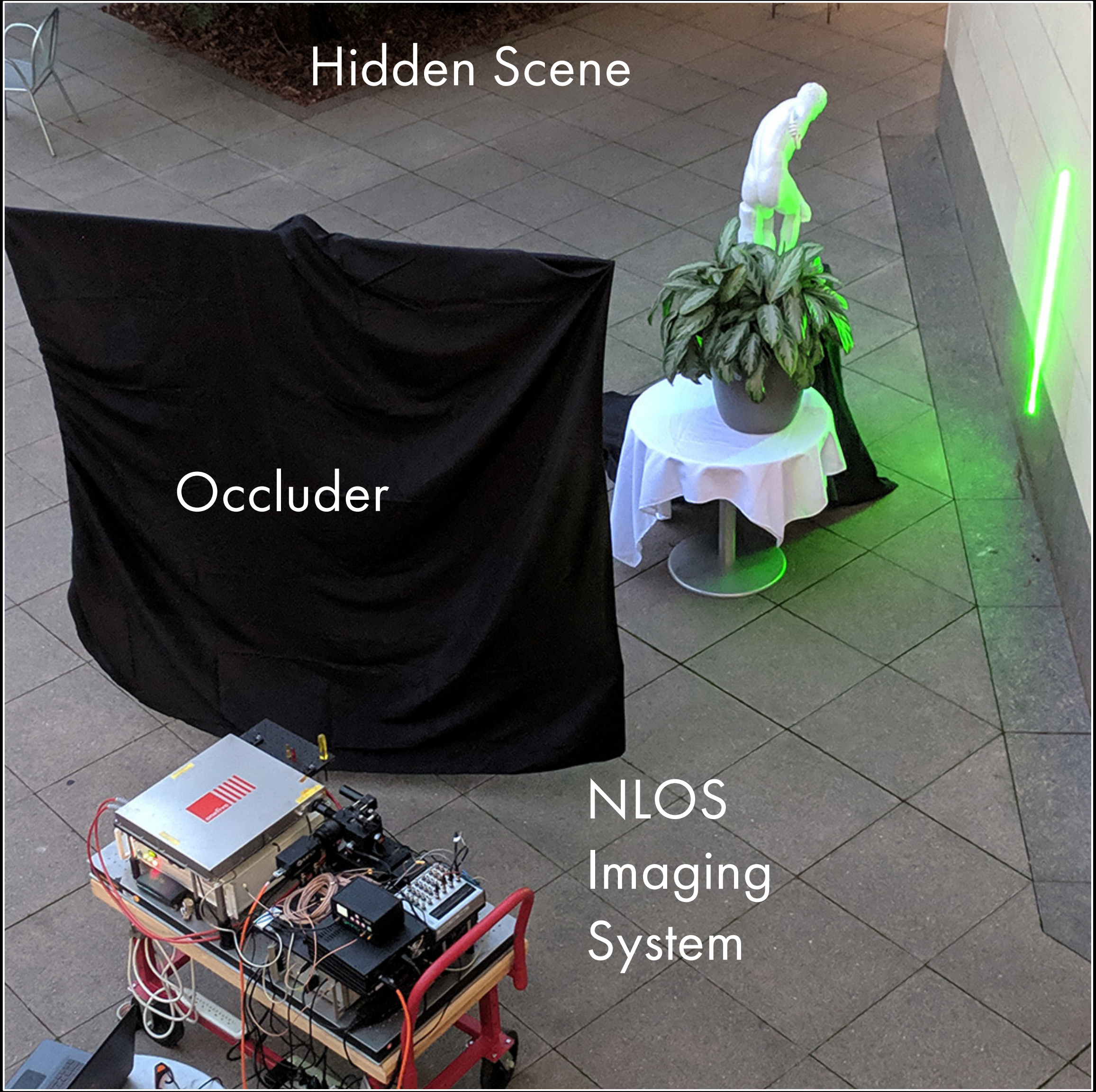
picosecond laser

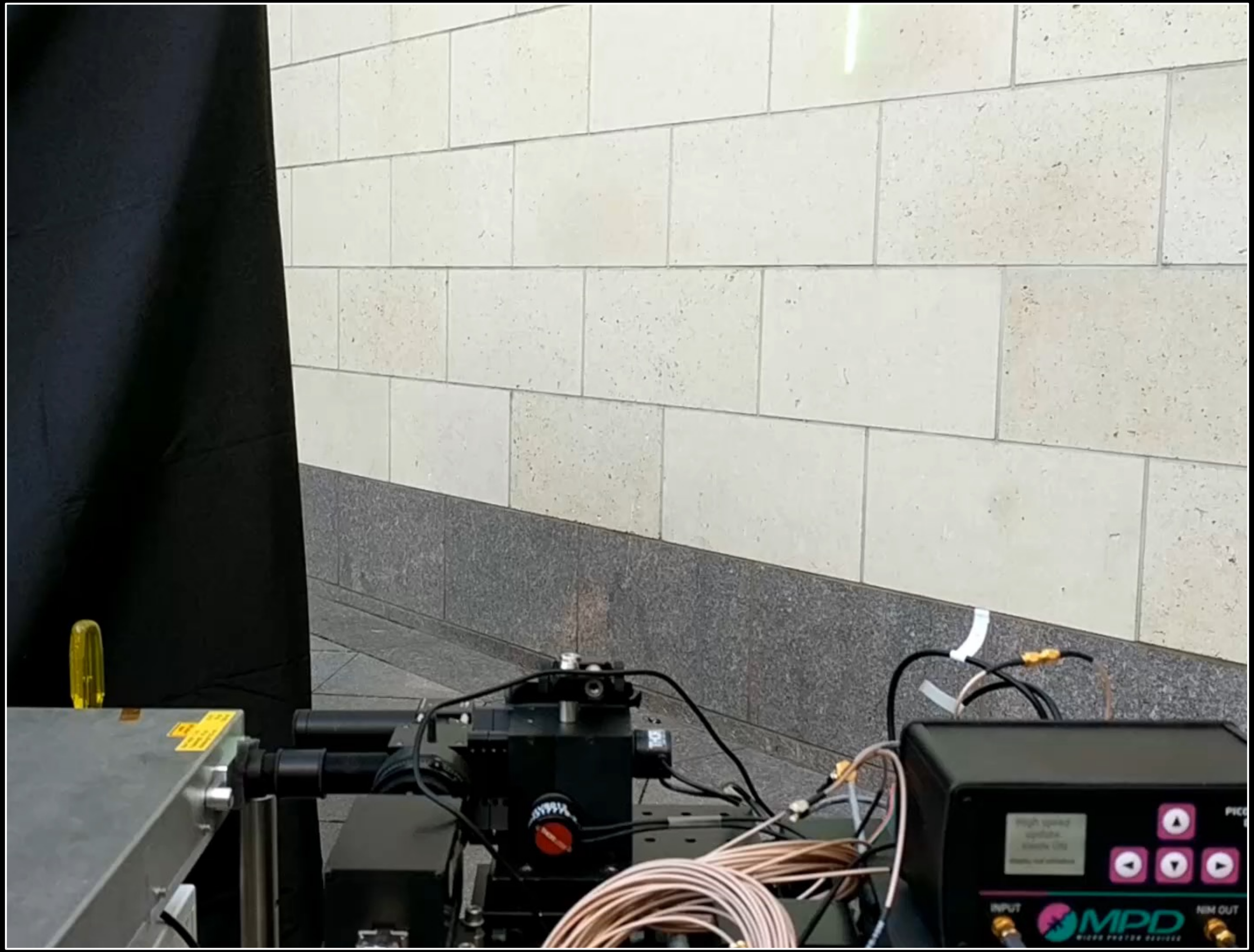
glowing fiber optic cable

regular image

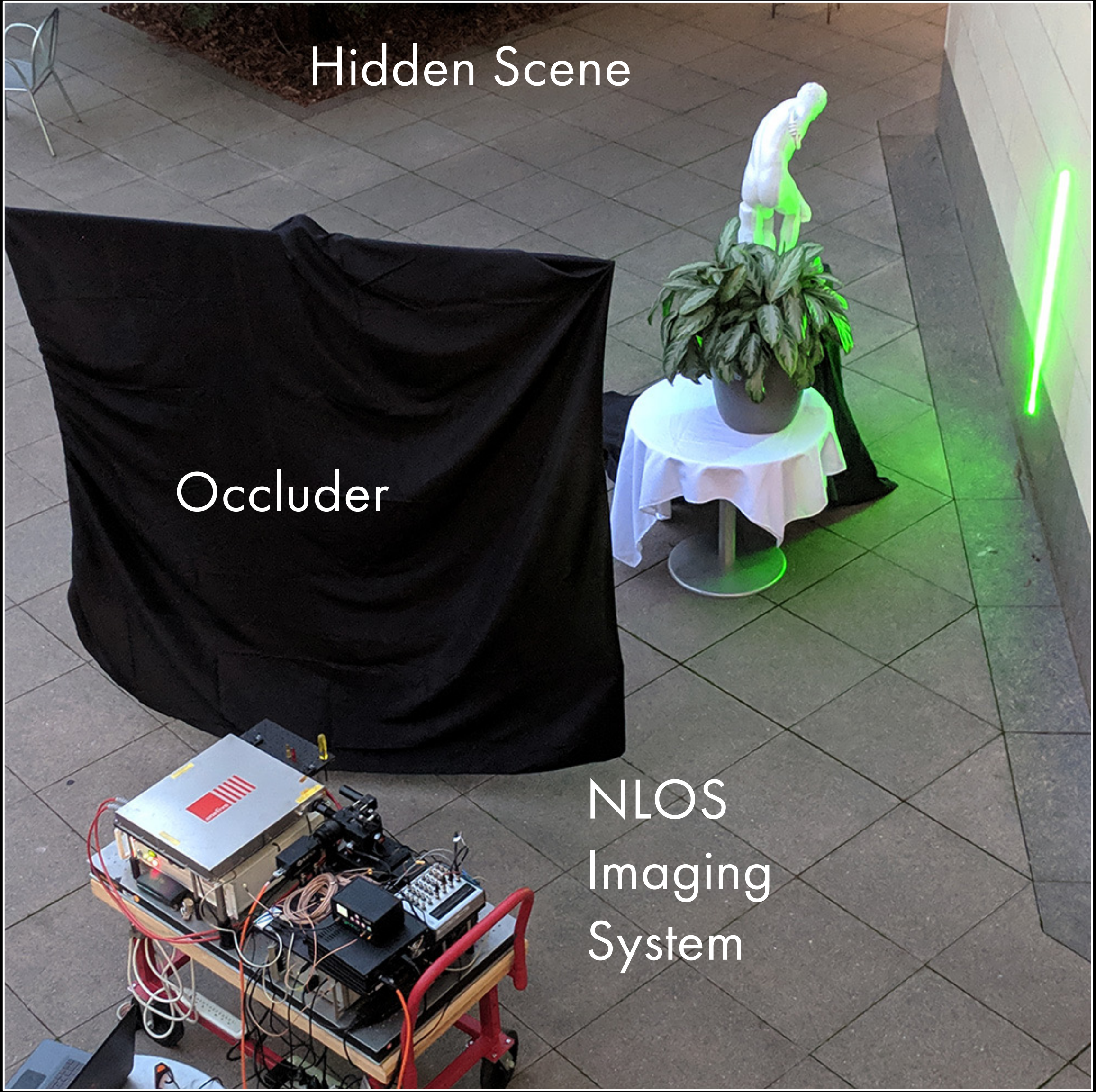


transient image

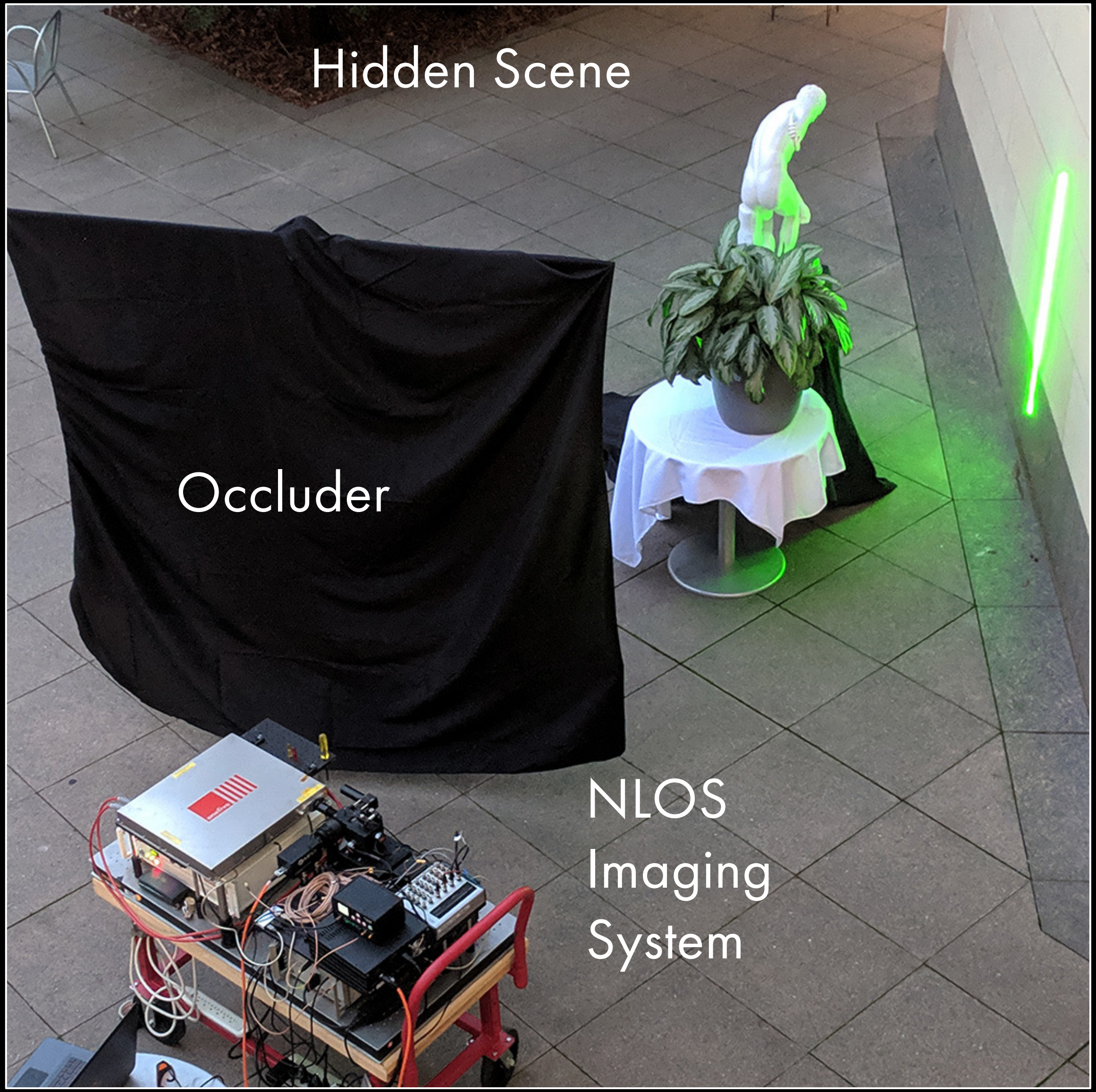




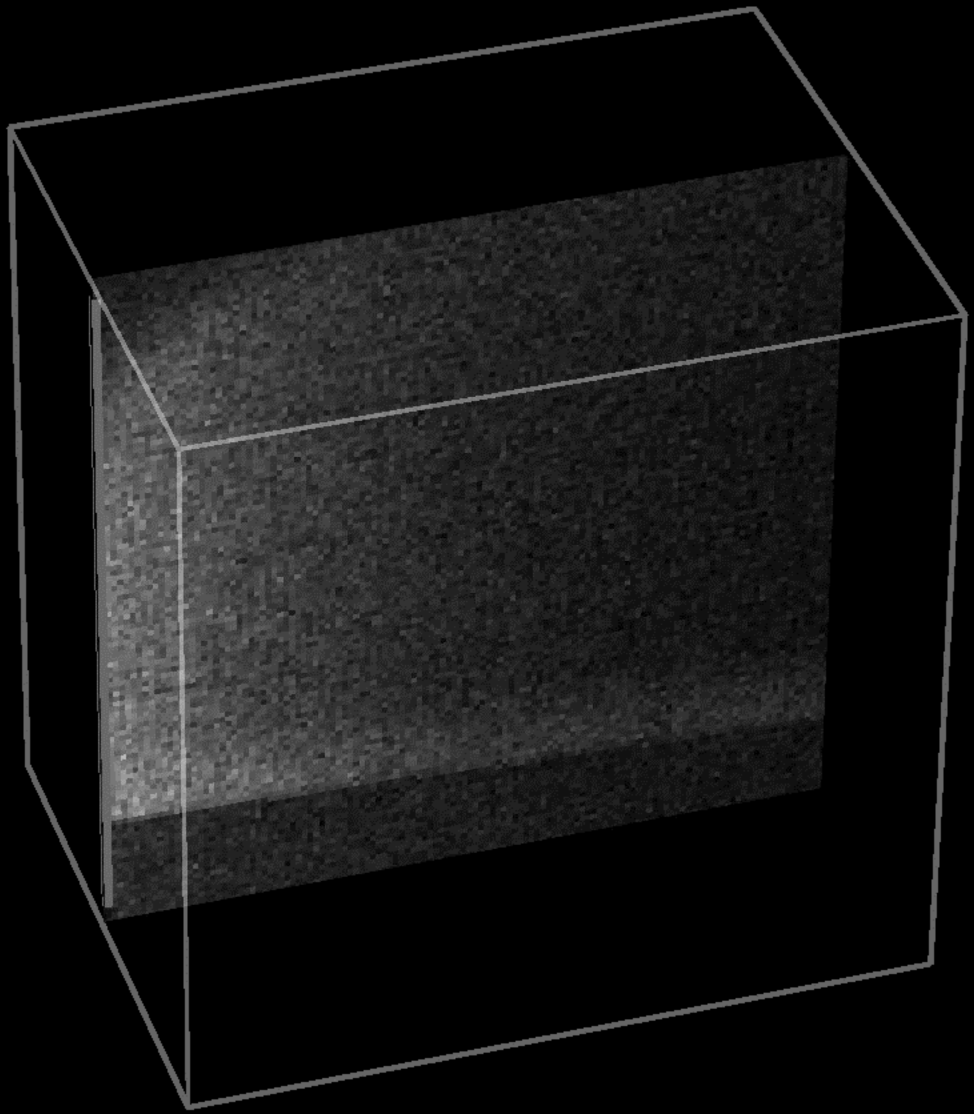
[Lindell et al. SIGGRAPH '19]



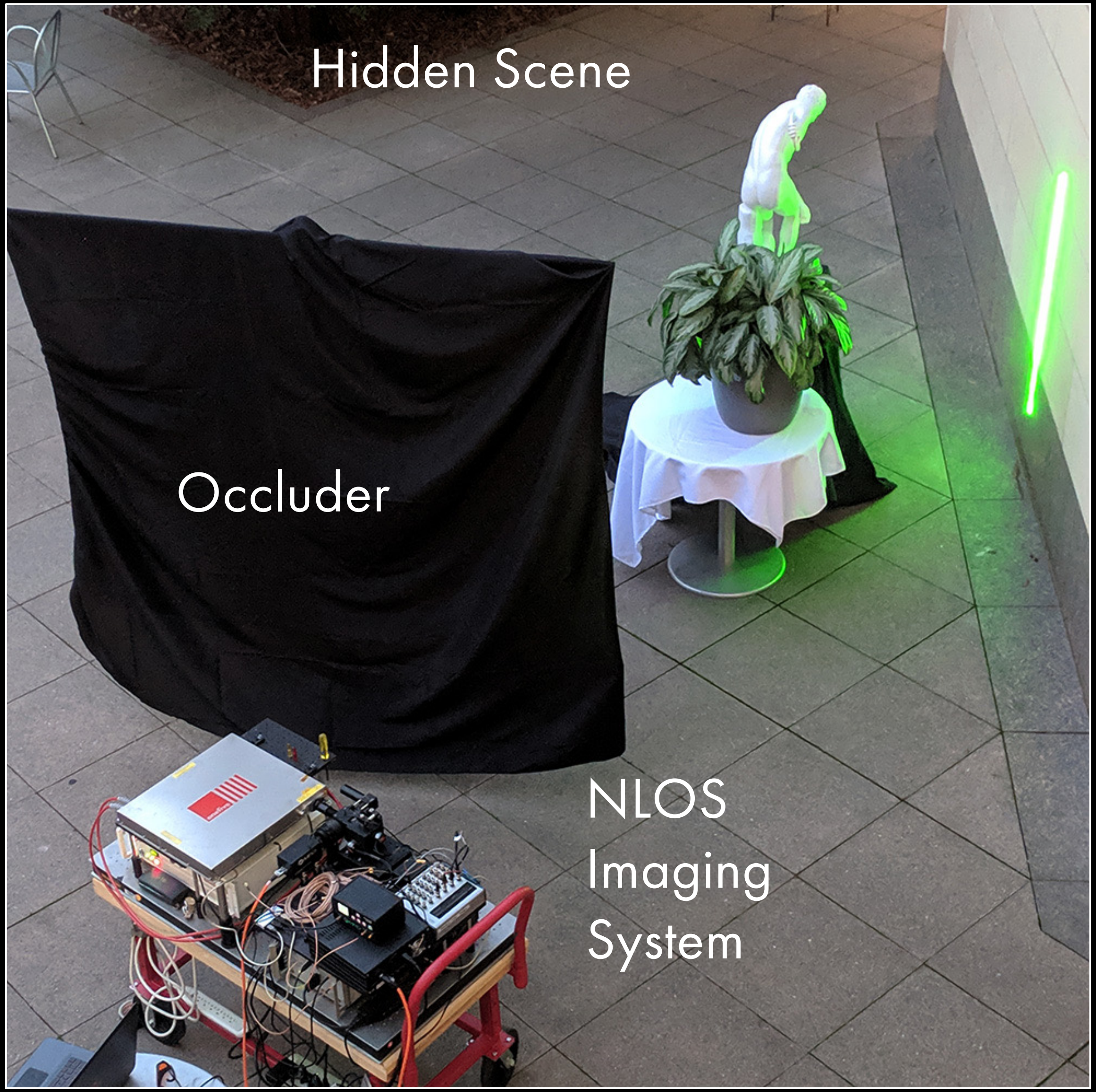
[Lindell et al. SIGGRAPH '19]



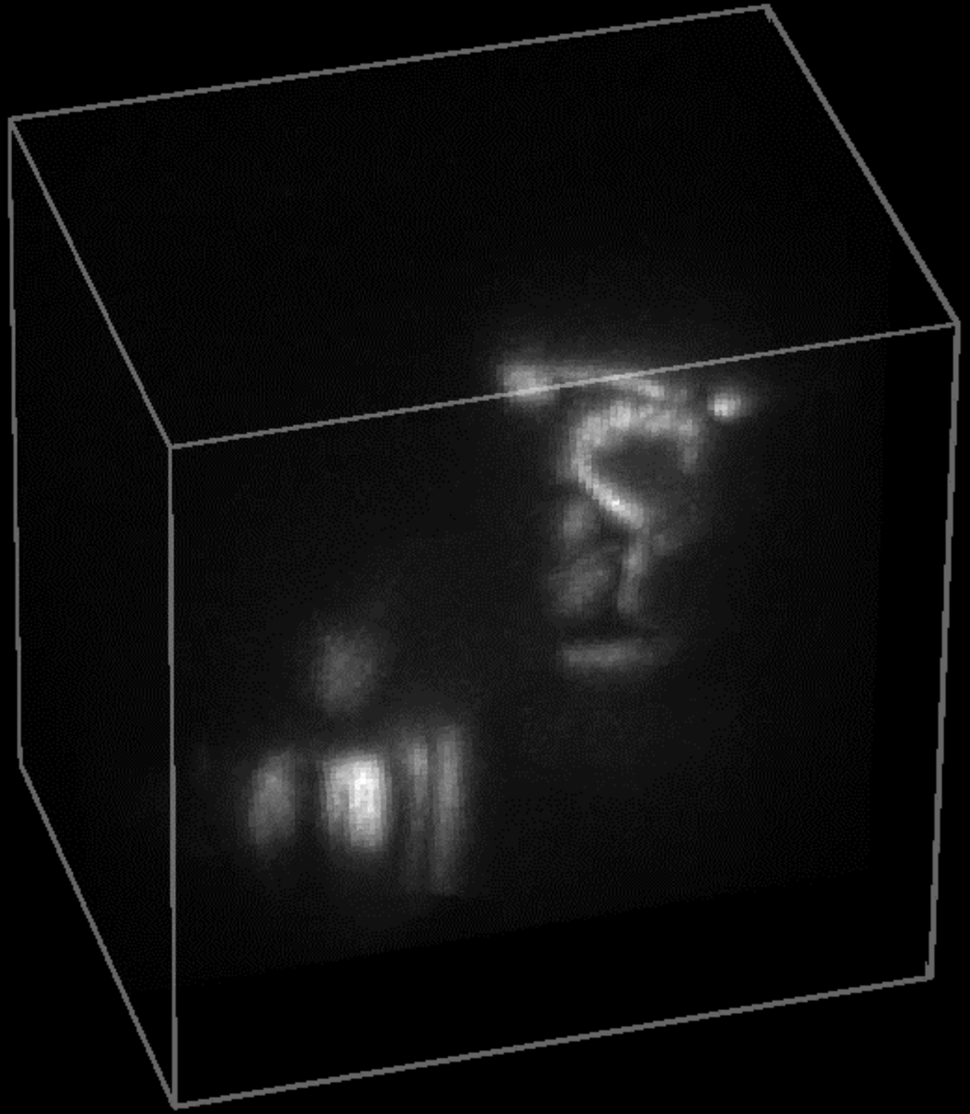
[Lindell et al. SIGGRAPH '19]



Time-resolved Measurements



[Lindell et al. SIGGRAPH '19]



3D Reconstruction

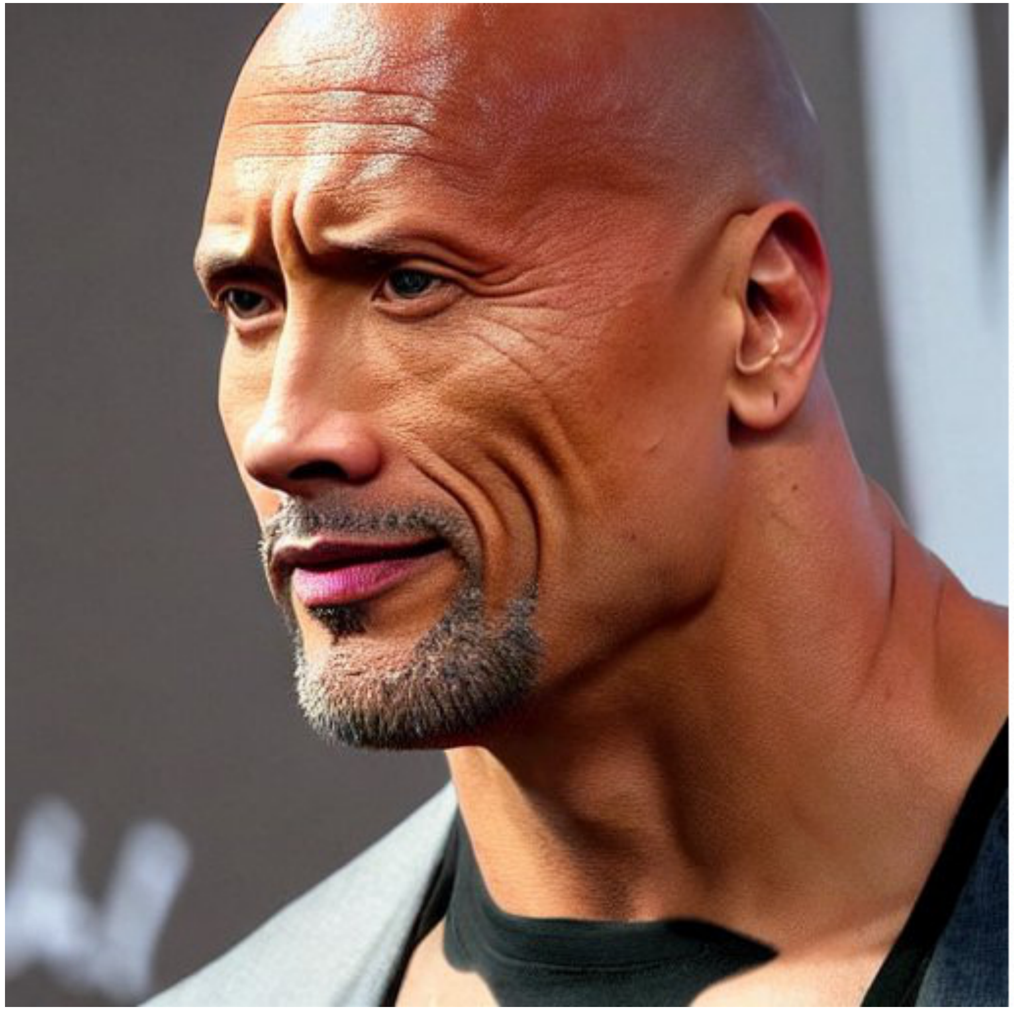
Why study Computer Vision?

Automatically caption images...

[Source: L. Zitnick, NIPS'14 Workshop on Learning Semantics]

Why study Computer Vision?

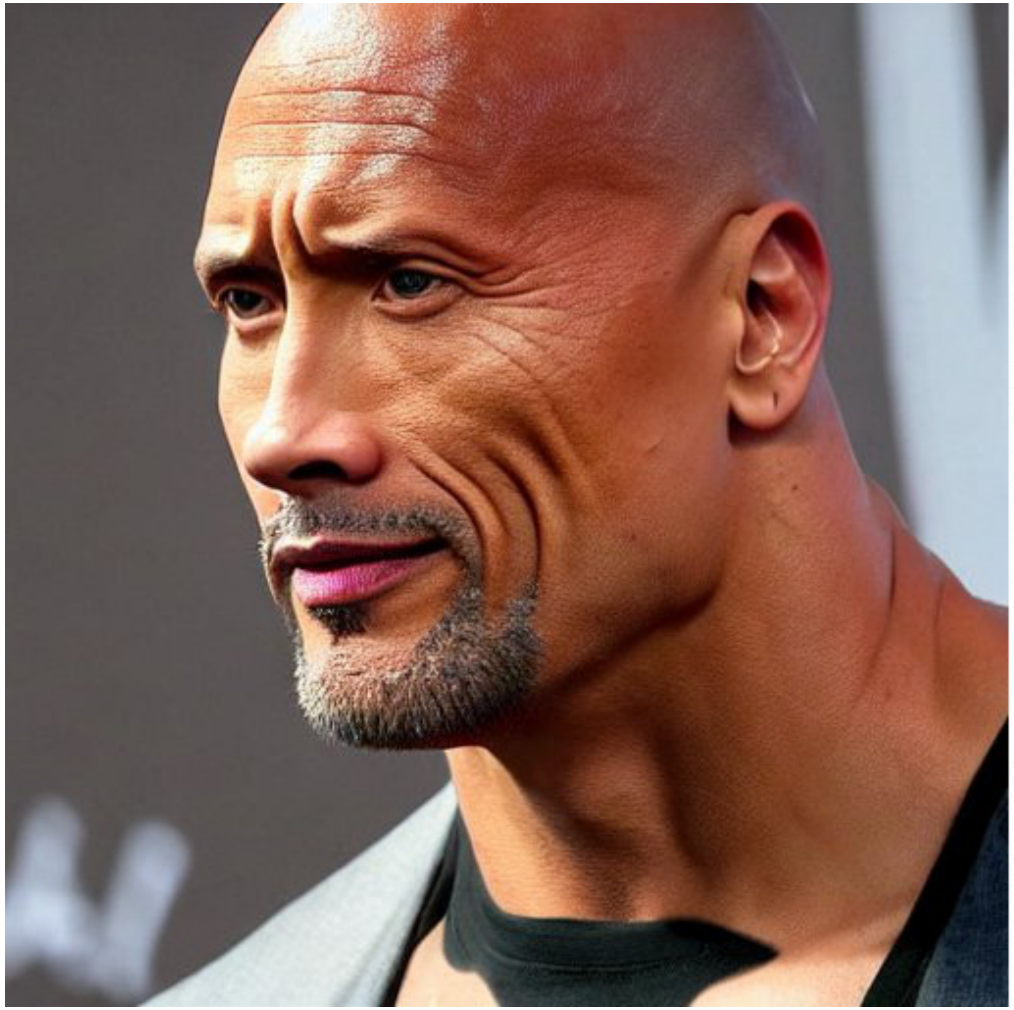
Generate an image from a caption (stable diffusion)



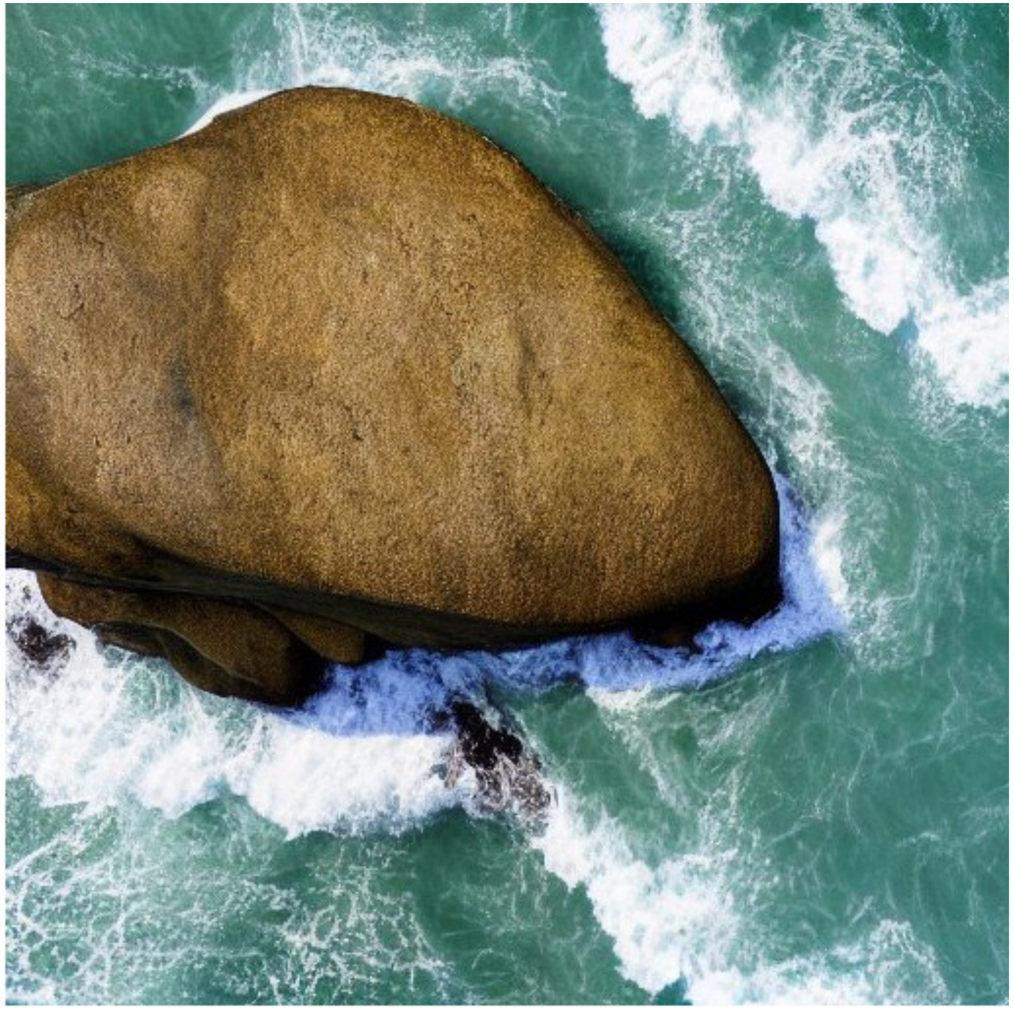
"Dwayne Johnson side view"

Why study Computer Vision?

Generate an image from a caption (stable diffusion)



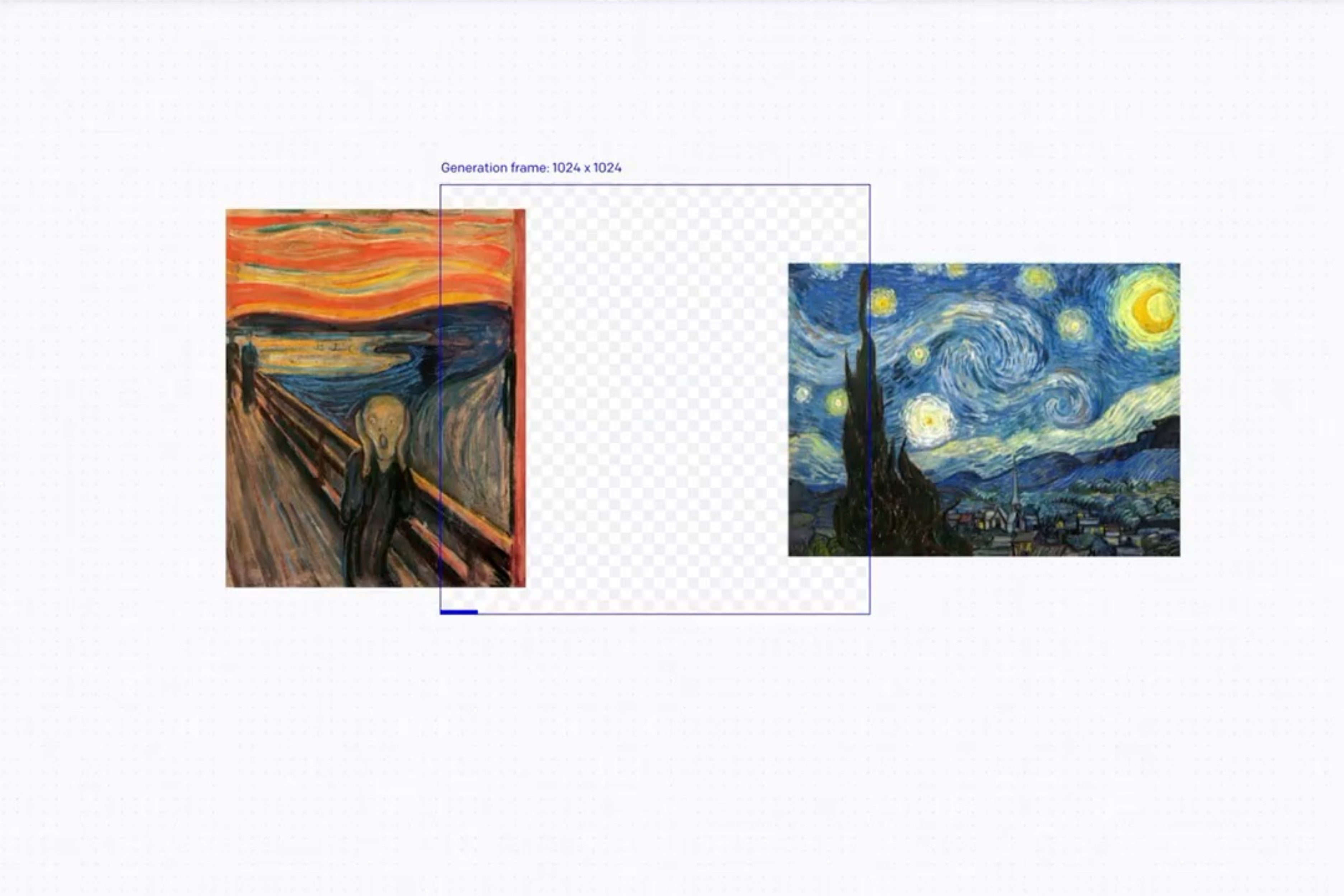
"Dwayne Johnson side view"



"Dwayne Johnson top view"

Why study Computer Vision?

Inpainting art...



Why study Computer Vision?

Movie-like image forensics

[Nayar and Nishino, Eyes for Relighting]

[Slide: N. Snavely]

Why study Computer Vision?

Movie-like image forensics

[Nayar and Nishino, Eyes for Relighting]

[Slide: N. Snavely]

Why study Computer Vision?

Movie-like image forensics

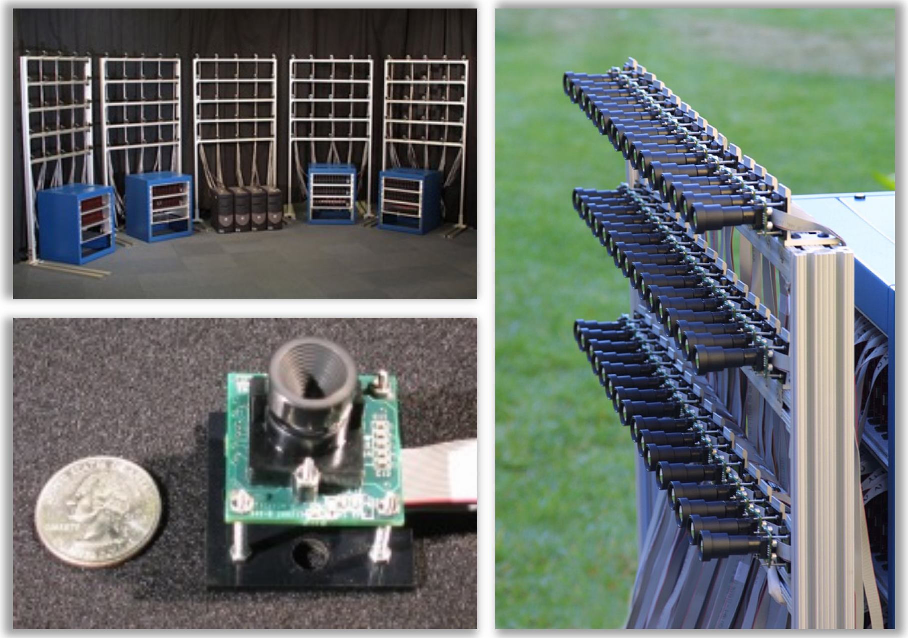
[Nayar and Nishino, Eyes for Relighting]

[Slide: N. Snavely]

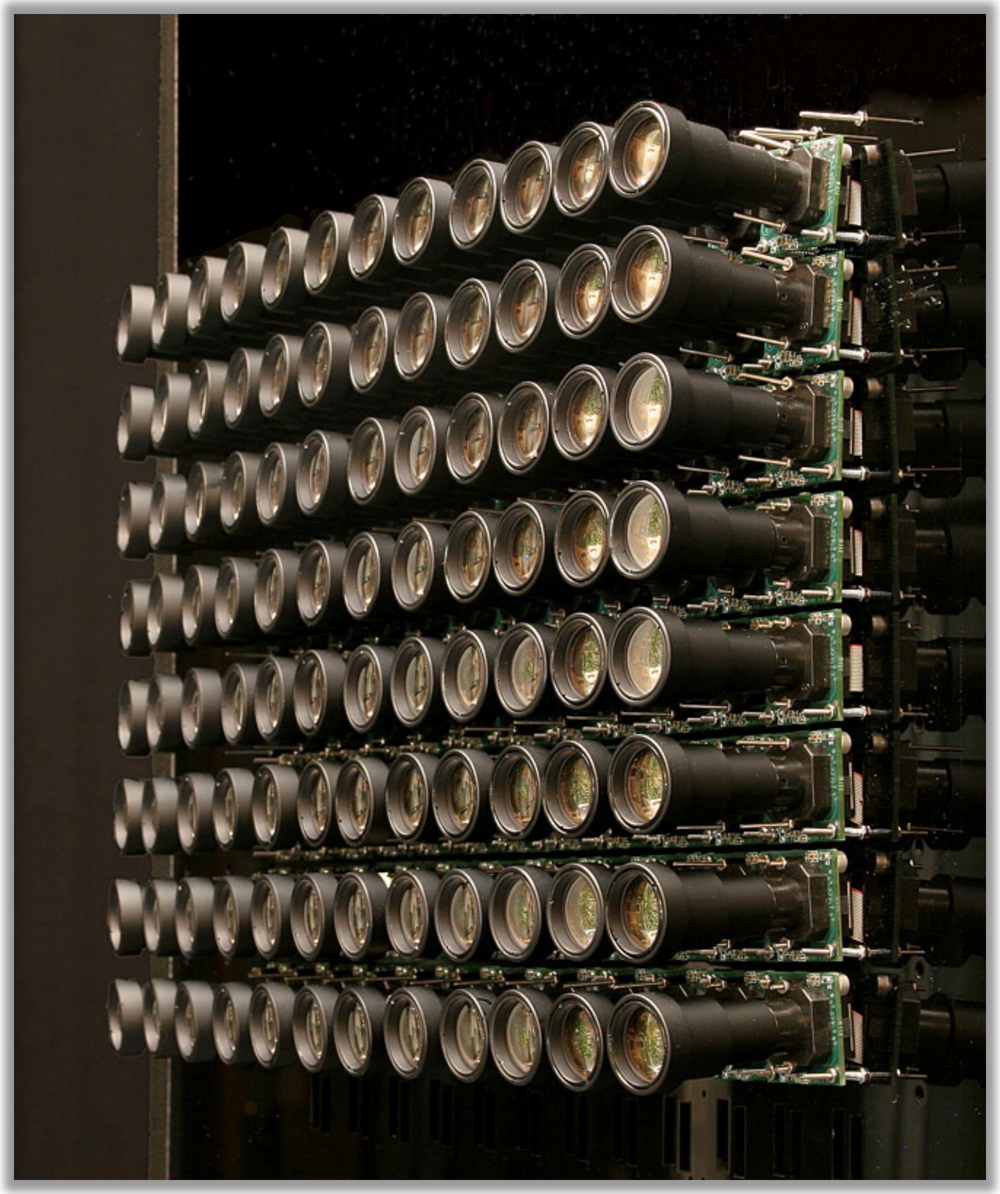
Why study Computer Vision?

Capture light fields

- Stanford Multi-Camera Array



125 cameras using custom hardware
[Wilburn et al. 2002, Wilburn et al. 2005]



How it all began

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence Group July 7, 1966

Vision Memo. No. 100.

THE SUMMER VISION PROJECT

Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

[Slide: A. Torralba]

Popular benchmarks:

The KITTI Vision Benchmark Suite
A project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

IMAGENET

COCO
Common Objects in Context

http://en.wikipedia.org/wiki/List_of_datasets_for_machine_learning_research

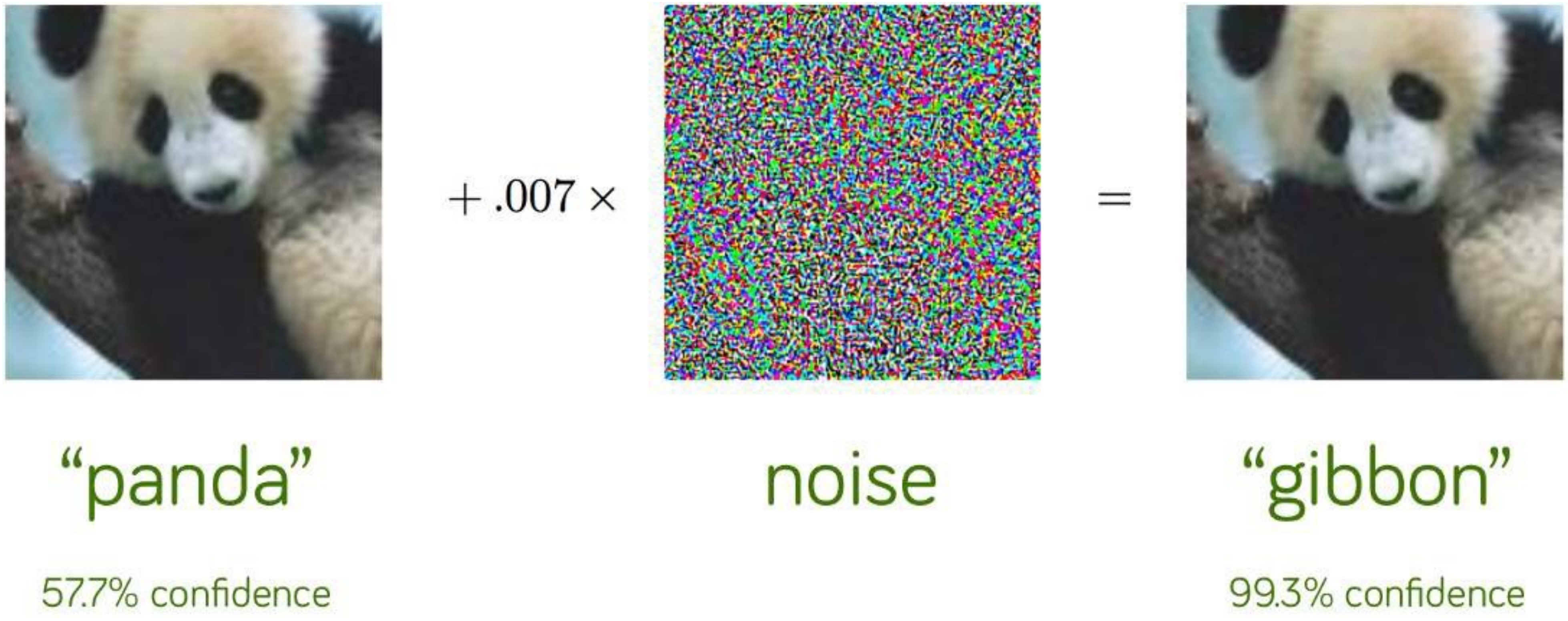
Car

	Method	Setting	Code	Moderate	Easy	Hard	Runtime	Environment	Compare
1	DenseBox2			89.32 %	93.94 %	79.81 %	5 s	GPU @ 2.5 Ghz (C/C++)	
2	DJML			88.79 %	91.31 %	77.73 %	x s	GPU @ 1.5 Ghz (Matlab + C/C++)	
3	3DOP			88.64 %	93.04 %	79.10 %	3s	GPU @ 2.5 Ghz (Matlab + C/C++)	

X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler and R. Urtasun: 3D Object Proposals for Accurate Object Class Detection. NIPS 2015.

	mean	aero plane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	dining table	dog	horse	motor bike	person	potted plant	sheep	sofa	train	tv/ monitor	submission date
Fast R-CNN + YOLO [?]	70.8	82.7	77.7	74.3	59.1	47.1	78.0	73.1	89.2	49.6	74.3	55.9	87.4	79.8	82.2	75.3	43.1	71.4	67.8	81.9	65.6	05-Jun-2015
Fast R-CNN VGG16 extra data [?]	68.8	82.0	77.8	71.6	55.3	42.4	77.3	71.7	89.3	44.5	72.1	53.7	87.7	80.0	82.5	72.7	36.6	68.7	65.4	81.1	62.7	18-Apr-2015
segDeepM [?]	67.2	82.3	75.2	67.1	50.7	49.8	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	41.5	71.9	62.2	73.2	64.6	29-Jan-2015
BabyLearning [?]	63.8	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	12-Nov-2014

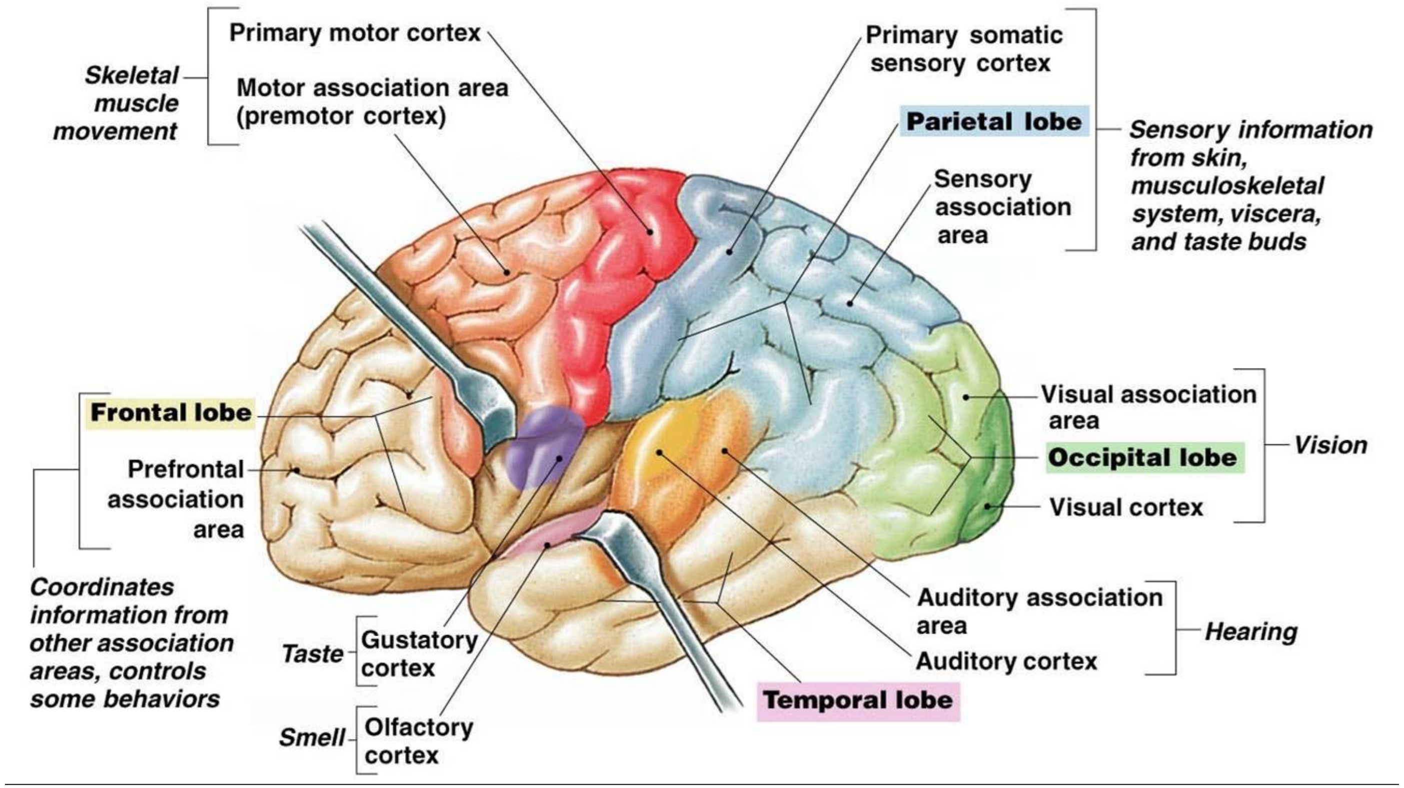
- Algorithms work pretty well
- Still some embarrassing mistakes...
- The general vision problem is not yet solved

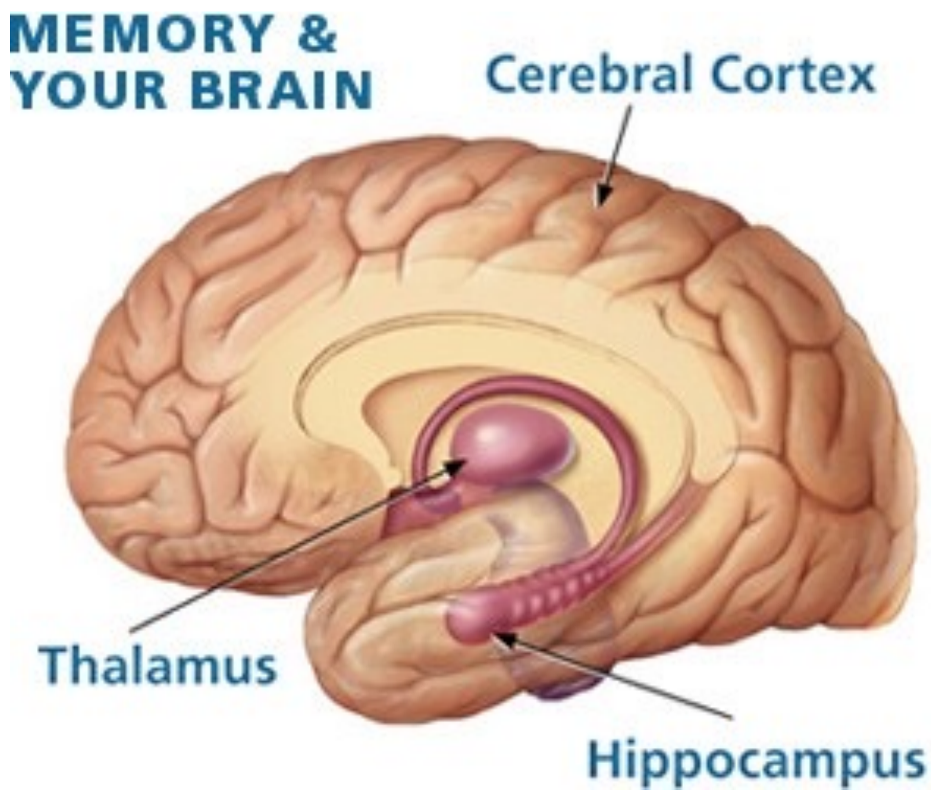


Why is vision hard?

Why is vision hard?

Half of the cerebral cortex in primates is devoted to processing visual information. This is a lot. Means that vision has to be pretty hard!





Why is vision hard?

Visual information is complicated and nuanced...

These are all dogs!

[Slide: R. Urtasun]

Why is vision hard?

Image: Karen Zack

Why is vision hard?

Image: Karen Zack

Why is vision hard?

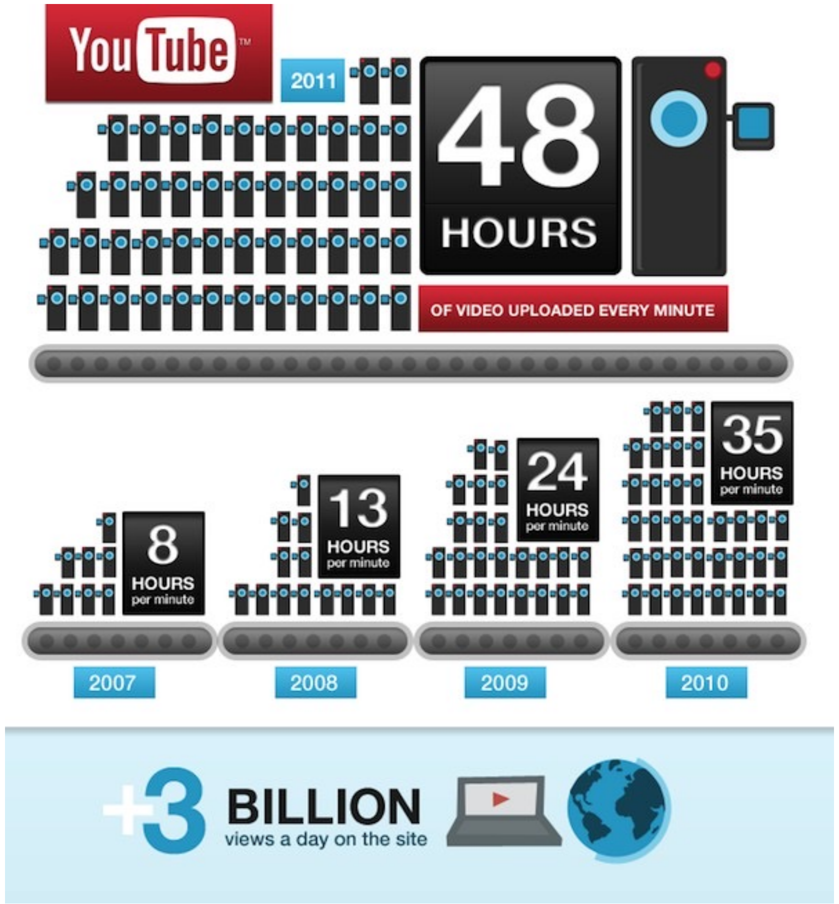
Biederman, 1987

[Slide: R. Urtasun]

Why is vision hard?

Lots of data to process:

- Thousands to millions of pixels in an image
- 400 hours of video added to YouTube per minute (2022)
- Every day, people watch one billion hours of video on YouTube (2022)
- Much more considering all other platforms



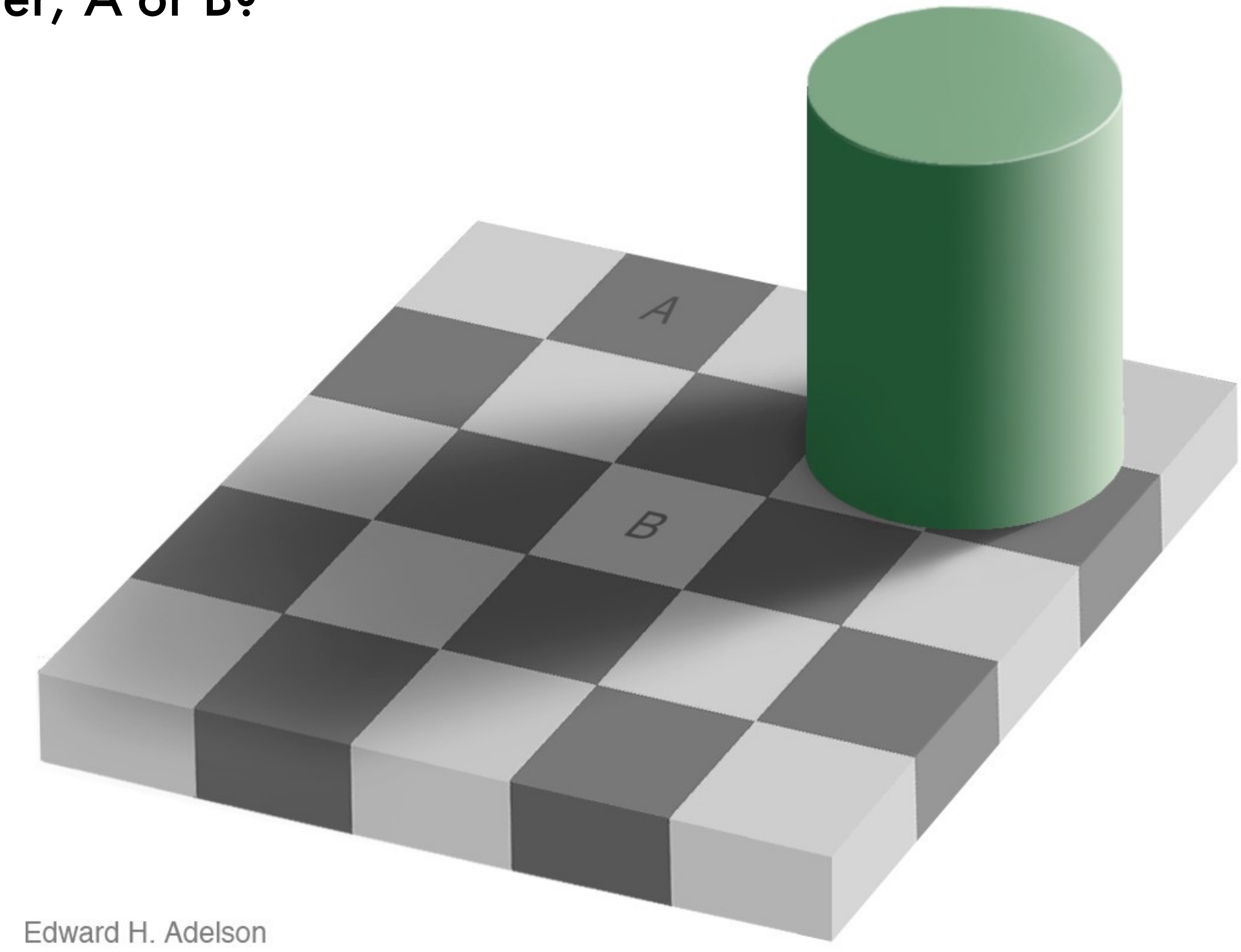
Human vision seems to work quite well.

How well does it really work?

Let’s play some games!

How good are humans?

Which square is lighter, A or B?

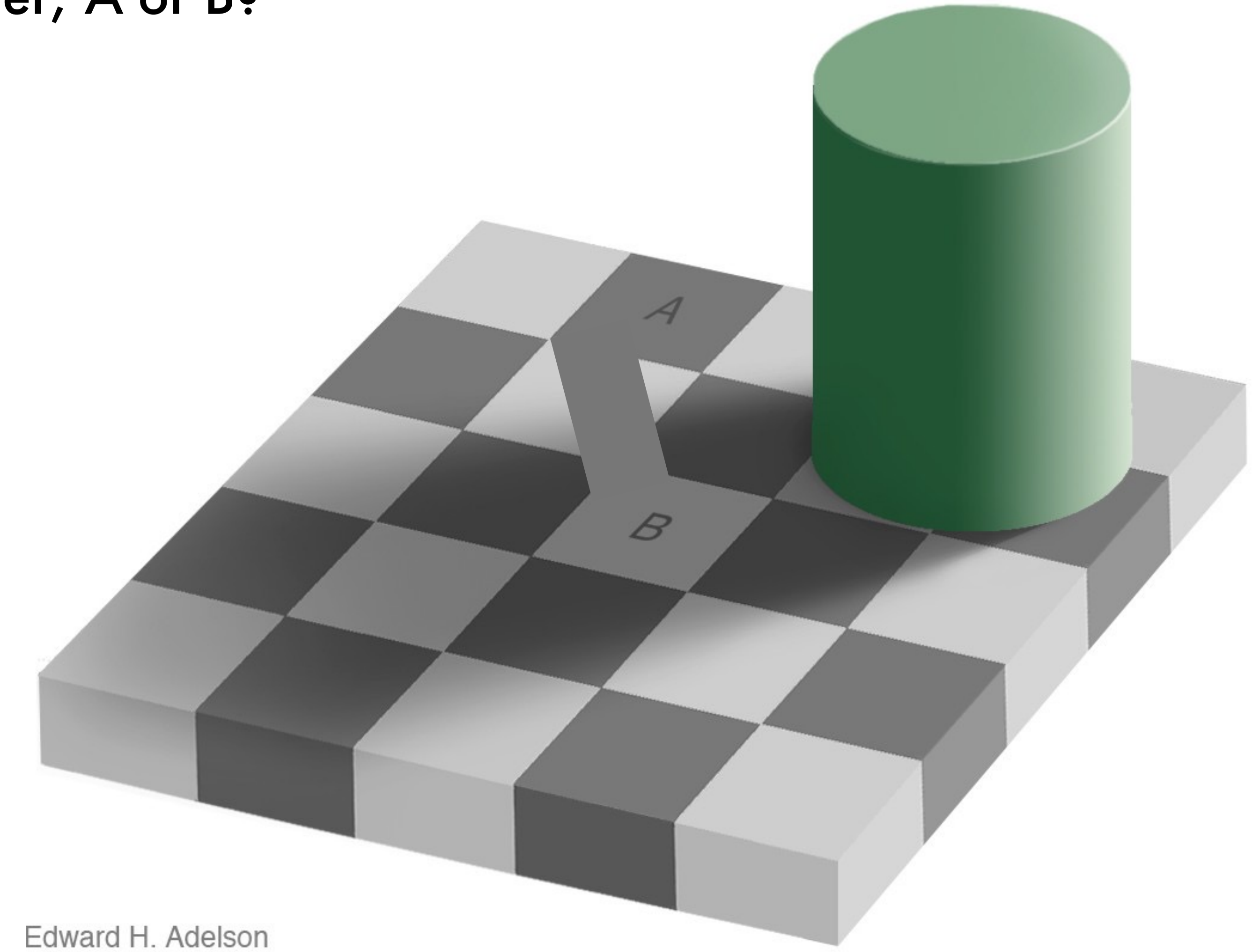


[Slide: A. Torralba]

How good are humans?

Which square is lighter, A or B?

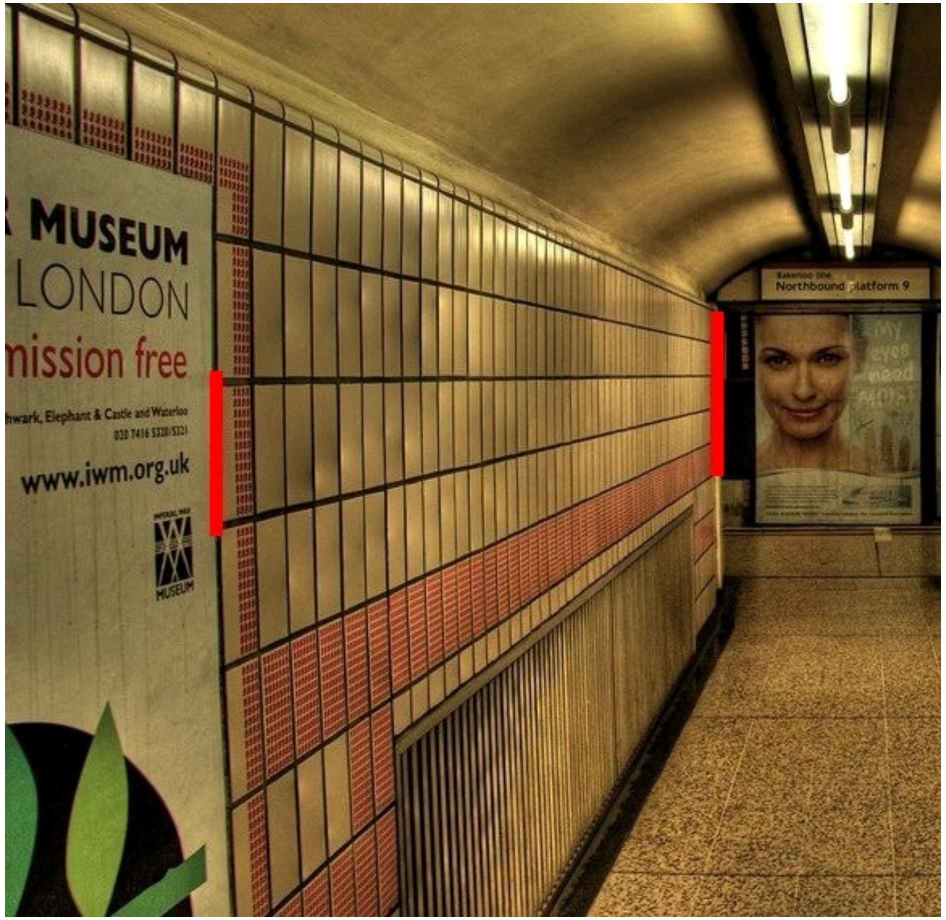
They are the same...



[Slide: A. Torralba]

How good are humans?

Which red line is longer?



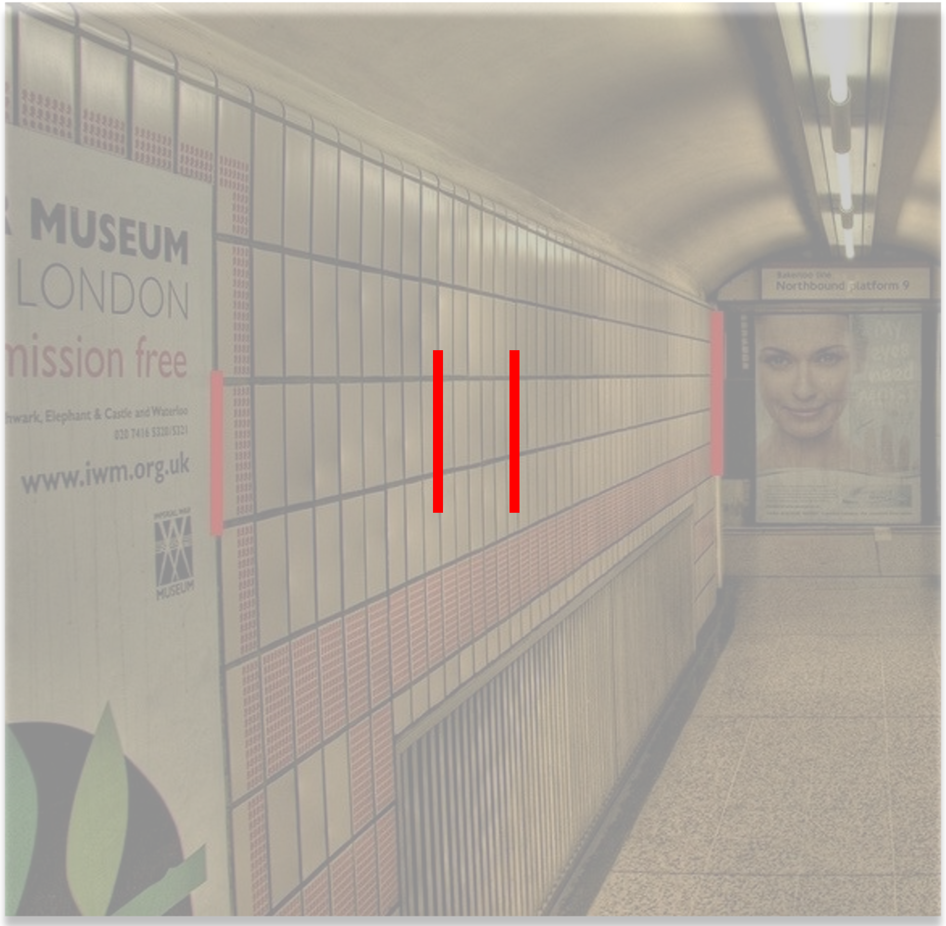
[Walt Anthony 2006]

[Slide: A. Torralba]

How good are humans?

Which red line is longer?

They are the same...

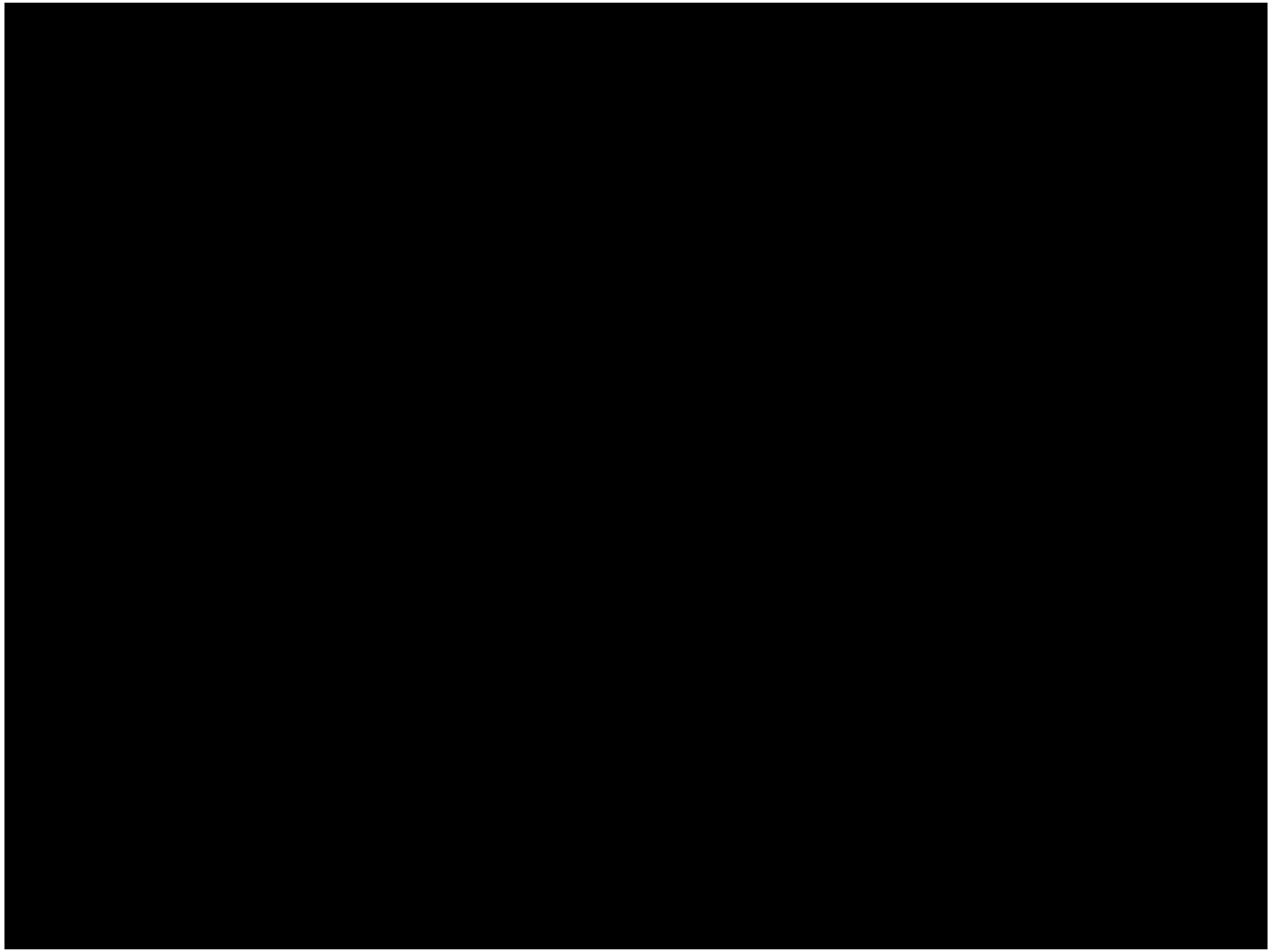


[Walt Anthony 2006]

[Slide: A. Torralba]

How good are humans?

- Count the number of times the white team pass the ball
- Concentrate, it's difficult!



[Chabris & Simons]

How good are humans?

Can you describe what this is?

[Torralba et al.]

How good are humans?

Can you describe what this is?

[Torralba et al.]

Humans can tell a lot from a little information...
we have prior knowledge that can (usually) fill in the right information

What do I need to become a good Computer Vision researcher?

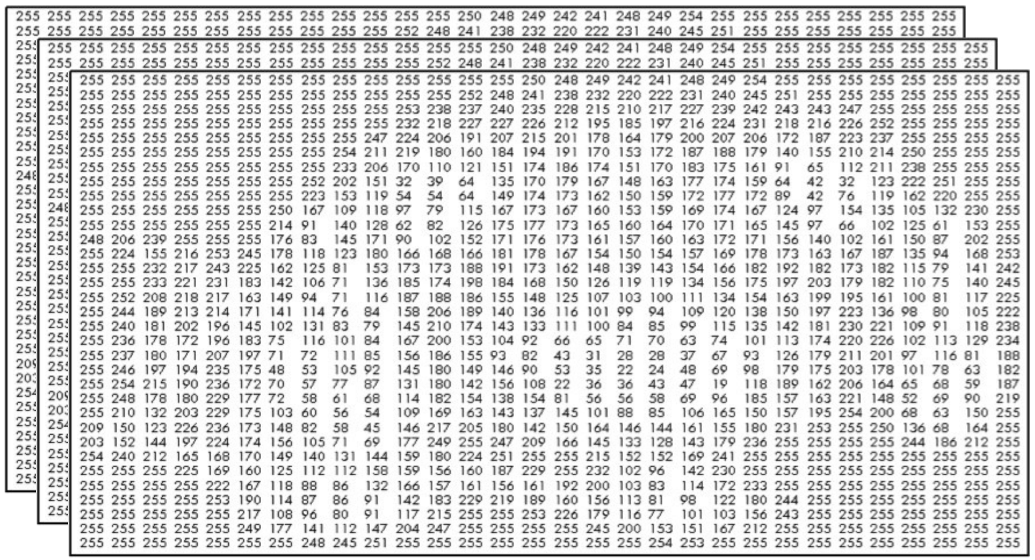
- Some math knowledge
- Good programming skills
- Imagination
- Even better intuition
- Lots of persistence
- Some luck always helps

Images

Digital Image

An image is a matrix with (typically) integer values

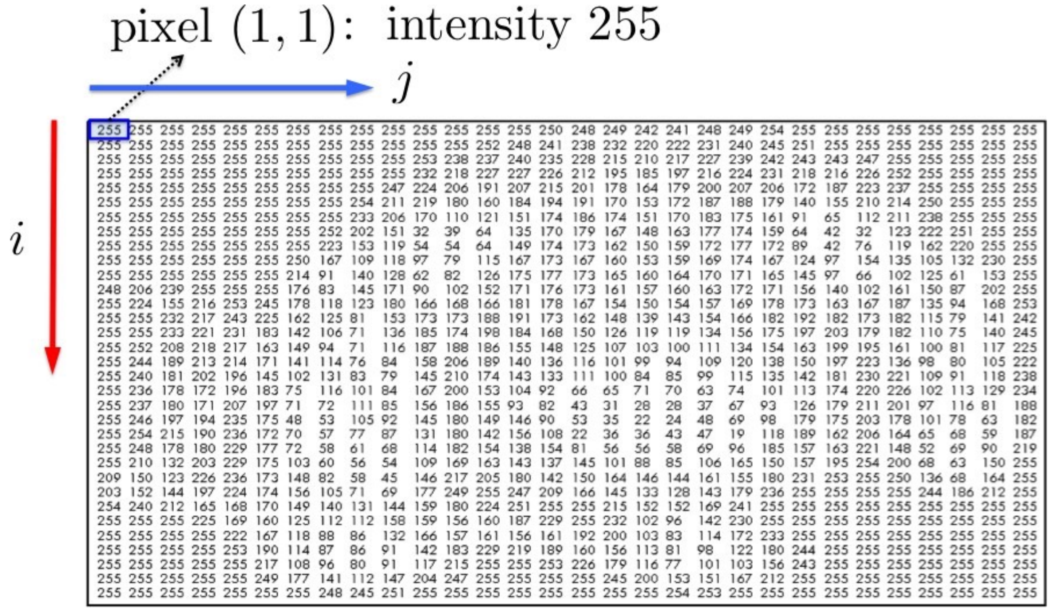
- We will typically denote the image as I
- Pixel values in the image are given by $I(i, j)$, the intensity value at each pixel
- For a grayscale image we have $I \in \mathbb{R}^{m \times n}$, color is $I \in \mathbb{R}^{m \times n \times 3}$



Digital Image

An image is a matrix with (typically) integer values

- We will typically denote the image as I
- Pixel values in the image are given by $I(i,j)$, the intensity value at each pixel
- For a grayscale image we have $I \in \mathbb{R}^{m \times n}$, color is $I \in \mathbb{R}^{m \times n \times 3}$



Digital Image

An image is a matrix with (typically) integer values

- We will typically denote the image as I
- Pixel values in the image are given by $I(i, j)$, the intensity value at each pixel
- For a grayscale image we have $I \in \mathbb{R}^{m \times n}$, color is $I \in \mathbb{R}^{m \times n \times 3}$

255	255	255	255	255	255	255	255	255	255	255	255	255	255	250	248	249	242	241	248	249	254	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	252	248	241	238	232	220	222	231	240	245	251	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	253	238	237	240	235	228	215	210	217	227	239	242	243	243	247	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	232	218	227	227	226	212	195	185	197	216	224	231	218	216	226	252	255	255	255	255
255	255	255	255	255	255	255	255	255	247	224	206	191	207	215	201	178	164	179	200	207	206	172	187	223	237	255	255	255	255
255	255	255	255	255	255	255	255	254	211	219	180	160	184	194	191	170	153	172	187	188	179	140	155	210	214	250	255	255	255
255	255	255	255	255	255	255	255	233	206	170	110	121	151	174	186	174	151	170	183	175	161	91	65	112	211	238	255	255	255
255	255	255	255	255	255	255	252	202	151	32	39	64	135	170	179	167	148	163	177	174	159	64	42	32	123	222	251	255	255
255	255	255	255	255	255	255	223	153	119	54	54	64	149	174	173	162	150	159	172	177	172	89	42	76	119	162	220	255	255
255	255	255	255	255	255	250	167	109	118	97	79	115	167	173	167	160	153	159	169	174	167	124	97	154	135	105	132	230	255
255	255	255	255	255	255	214	91	140	128	62	82	126	175	177	173	165	160	164	170	171	165	145	97	66	102	125	61	153	255
248	206	239	255	255	255	176	83	145	171	90	102	152	171	176	173	161	157	160	163	172	171	156	140	102	161	150	87	202	255
255	224	155	216	253	245	178	118	123	180	166	168	166	181	178	167	154	150	154	157	169	178	173	163	167	187	135	94	168	253
255	255	232	217	243	225	162	125	81	153	173	173	188	191	173	162	148	139	143	154	166	182	192	182	173	182	115	79	141	242
255	255	233	221	231	183	142	106	71	136	185	174	198	184	168	150	126	119	119	134	156	175	197	203	179	182	110	75	140	245
255	252	208	218	217	163	149	94	71	116	187	188	186	155	148	125	107	103	100	111	134	154	163	199	195	161	100	81	117	225
255	244	189	213	214	171	141	114	76	84	158	206	189	140	136	116	101	99	94	109	120	138	150	197	223	136	98	80	105	222
255	240	181	202	196	145	102	131	83	79	145	210	174	143	133	111	100	84	85	99	115	135	142	181	230	221	109	91	118	238
255	236	178	172	196	183	75	116	101	84	167	200	153	104	92	66	65	71	70	63	74	101	113	174	220	226	102	113	129	234
255	237	180	171	207	197	71	72	111	85	156	186	155	93	82	43	31	28	28	37	67	93	126	179	211	201	97	116	81	188
255	246	197	194	235	175	48	53	105	92	145	180	149	146	90	53	35	22	24	48	69	98	179	175	203	178	101	78	63	182
255	254	215	190	236	172	70	57	77	87	131	180	142	156	108	22	36	36	43	47	19	118	189	162	206	164	65	68	59	187
255	248	178	180	229	177	72	58	61	68	114	182	154	138	154	81	56	56	58	69	96	185	157	163	221	148	52	69	90	219
255	210	132	203	229	175	103	60	56	54	109	169	163	143	137	145	101	88	85	106	165	150	157	195	254	200	68	63	150	255
209	150	123	226	236	173	148	82	58	45	146	217	205	180	142	150	164	146	144	161	155	180	231	253	255	250	136	68	164	255
203	152	144	197	224	174	156	105	71	69	177	249	255	247	209	166	145	133	128	143	179	236	255	255	255	255	244	186	212	255
254	240	212	165	168	170	149	140	131	144	159	180	224	251	255	255	215	152	152	169	241	255	255	255	255	255	255	255	255	255
255	255	255	225	169	160	125	112	112	158	159	156	160	187	229	255	232	102	96	142	230	255	255	255	255	255	255	255	255	255
255	255	255	255	222	167	118	88	86	132	166	157	161	156	161	192	200	103	83	114	172	233	255	255	255	255	255	255	255	255
255	255	255	255	253	190	114	87	86	91	142	183	229	219	189	160	156	113	81	98	122	180	244	255	255	255	255	255	255	255
255	255	255	255	255	217	108	96	80	91	117	215	255	255	253	226	179	116	77	101	103	156	243	255	255	255	255	255	255	255
255	255	255	255	255	249	177	141	112	147	204	247	255	255	255	255	245	200	153	151	167	212	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	248	245	251	255	255	255	255	255	255	255	255	254	253	255	255	255	255	255	255	255	255	255	255

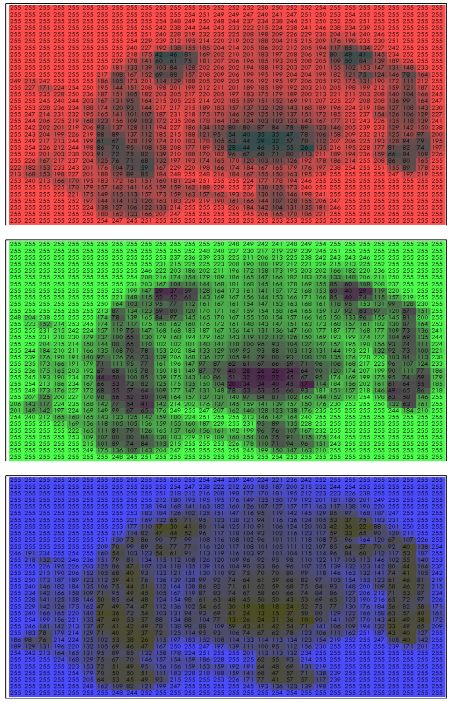
Digital Image

An image is a matrix with (typically) integer values

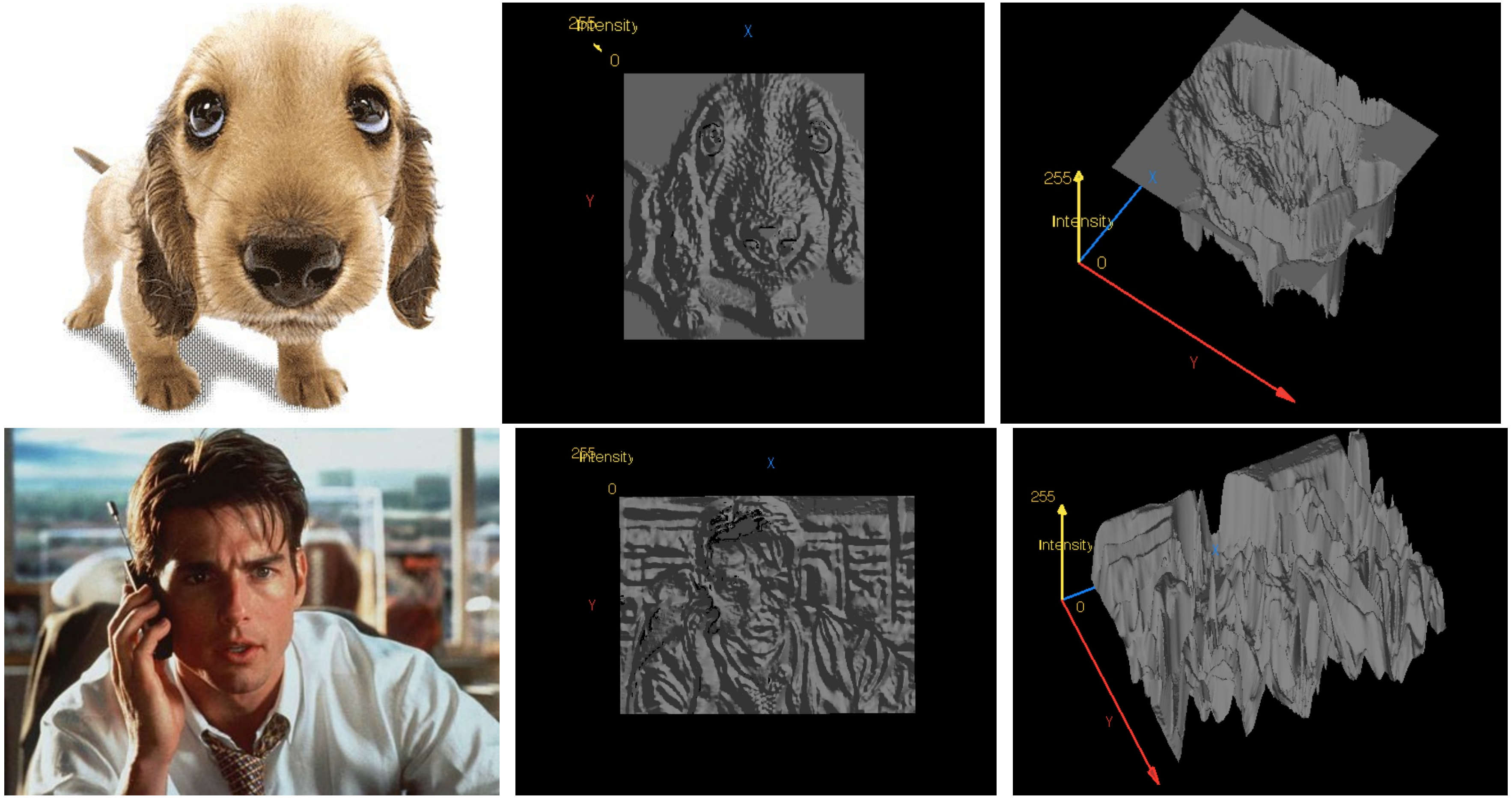
- We will typically denote the image as I

- Pixel values in the image are given by $I(i, j)$, the intensity value at each pixel

- For a grayscale image we have $I \in \mathbb{R}^{m \times n}$, color is $I \in \mathbb{R}^{m \times n \times 3}$



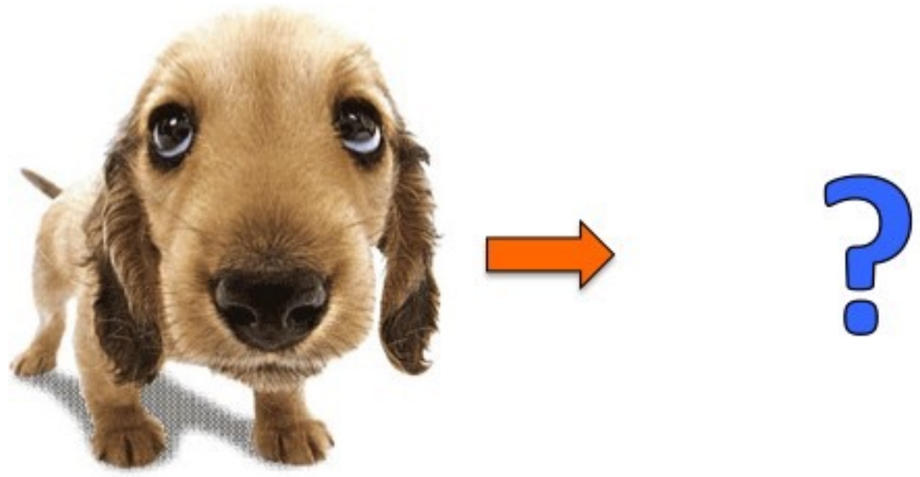
Digital Image



- We can think of a (grayscale) image as a function $f : \mathbb{R}^2 \mapsto \mathbb{R}$ giving the intensity at position (i, j)
- Intensity 0 is black and 255 is white

Digital Image

As with any function, we can apply operators to an image, e.g.:



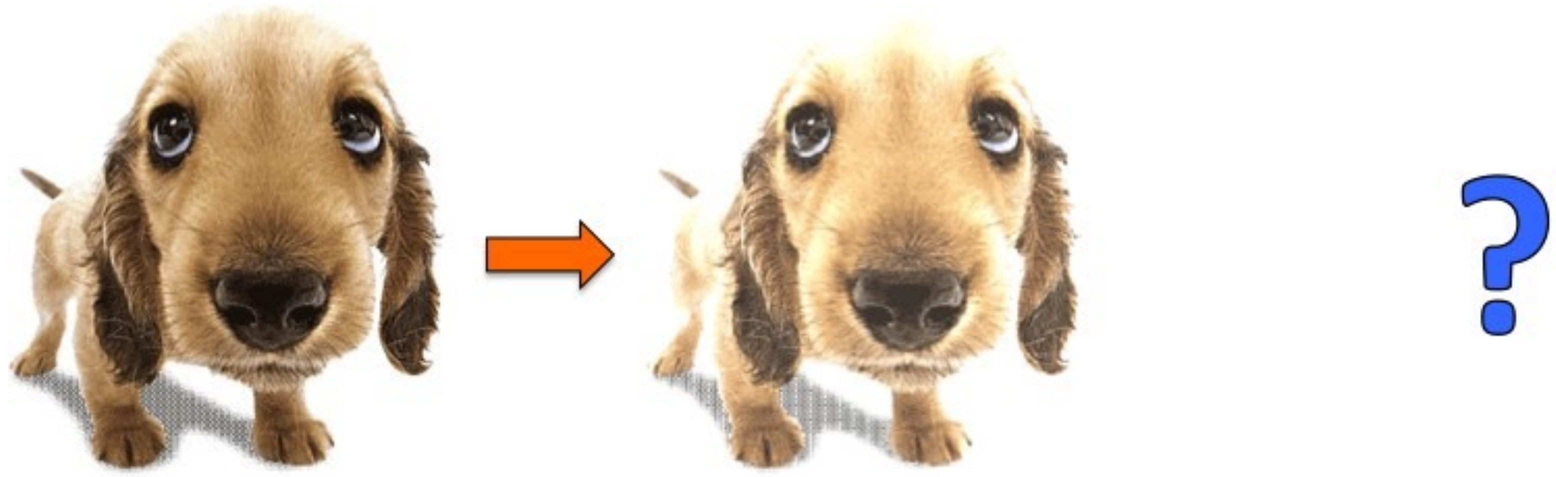
$I(i,j)$ $\qquad\qquad$ $J(i,j) = I(i,j) + 50$

We'll talk about special kinds of operators, correlation and convolution (linear filtering)

[Slide: N. Snavely]

Digital Image

As with any function, we can apply operators to an image, e.g.:



$I(i, j)$ $\quad$ $J(i, j) = I(i, j) + 50$ $\quad$ $J(i, j) = I(i, -j)$

We'll talk about special kinds of operators, correlation and convolution (linear filtering)

[Slide: N. Snavely]

Digital Image

As with any function, we can apply operators to an image, e.g.:

$I(i, j)$ $\quad$ $J(i, j) = I(i, j) + 50$ $\quad$ $J(i, j) = I(i, -j)$

We'll talk about special kinds of operators, correlation and convolution (linear filtering)

[Slide: N. Snavely]

Linear Filters

Reading: Szeliski book, Chapter 3.2

Motivation: Finding Waldo

How can we find Waldo?

[Source: R. Urtasun]

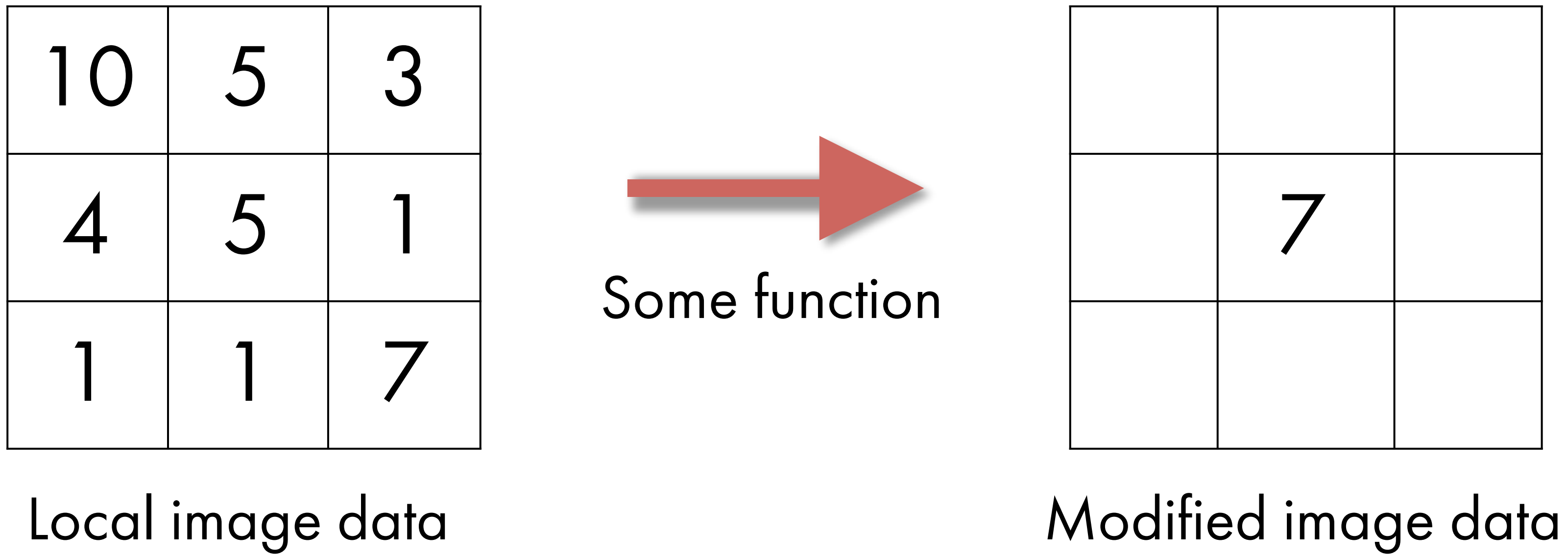
Motivation: Finding Waldo

Slide and compare! In formal language: filtering

Image Filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

- In other words, filtering



[Source: L. Zhang]

Applications of Filtering

- Enhance an image, e.g., denoise.
- Detect patterns, e.g., template matching.
- Extract information, e.g., texture, edges.

Applications of Filtering

- Enhance an image, e.g., denoise.
- Detect patterns, e.g., template matching.
- Extract information, e.g., texture, edges.

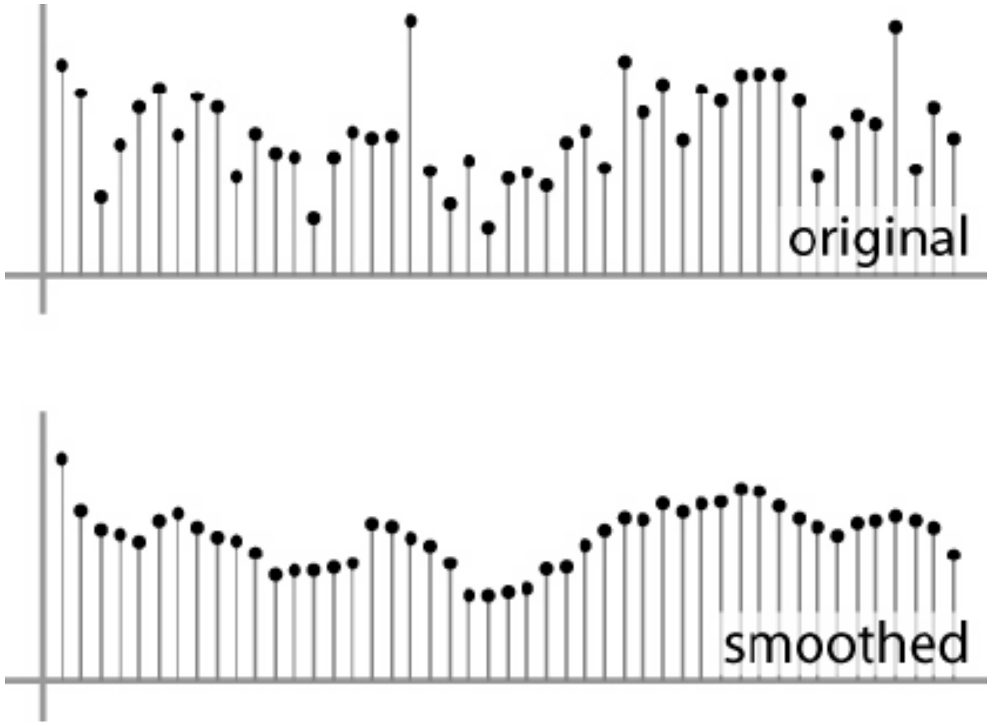
Noise reduction

Given a camera and a still scene, how can you reduce noise?

[Source: S. Seitz]

Noise reduction

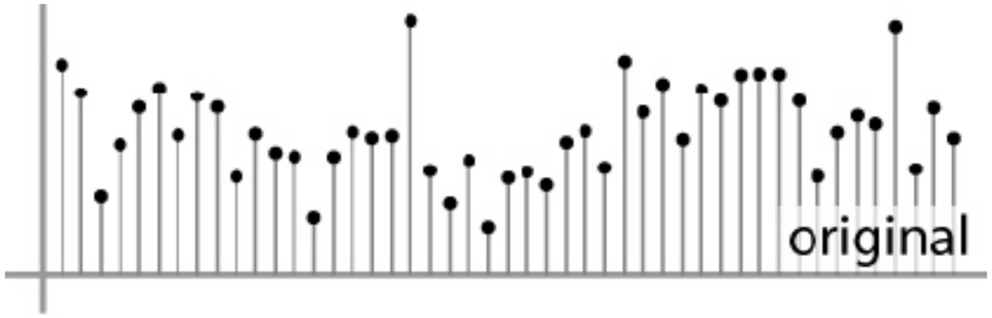
- Simplest thing: replace each pixel by the average of its neighbors.

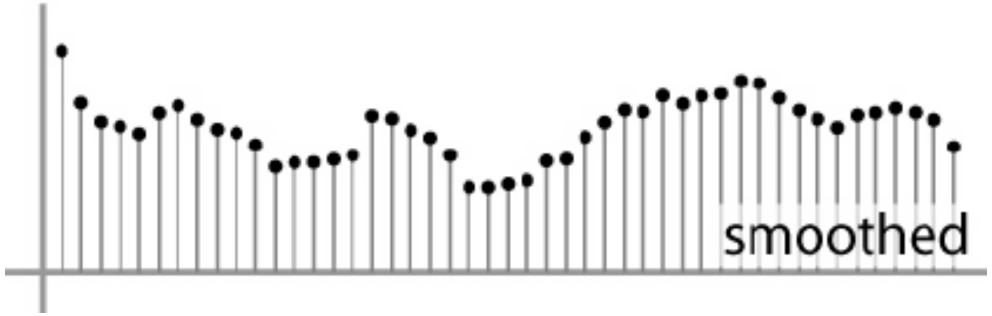


[Source: S. Marschner]

Noise reduction

- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.

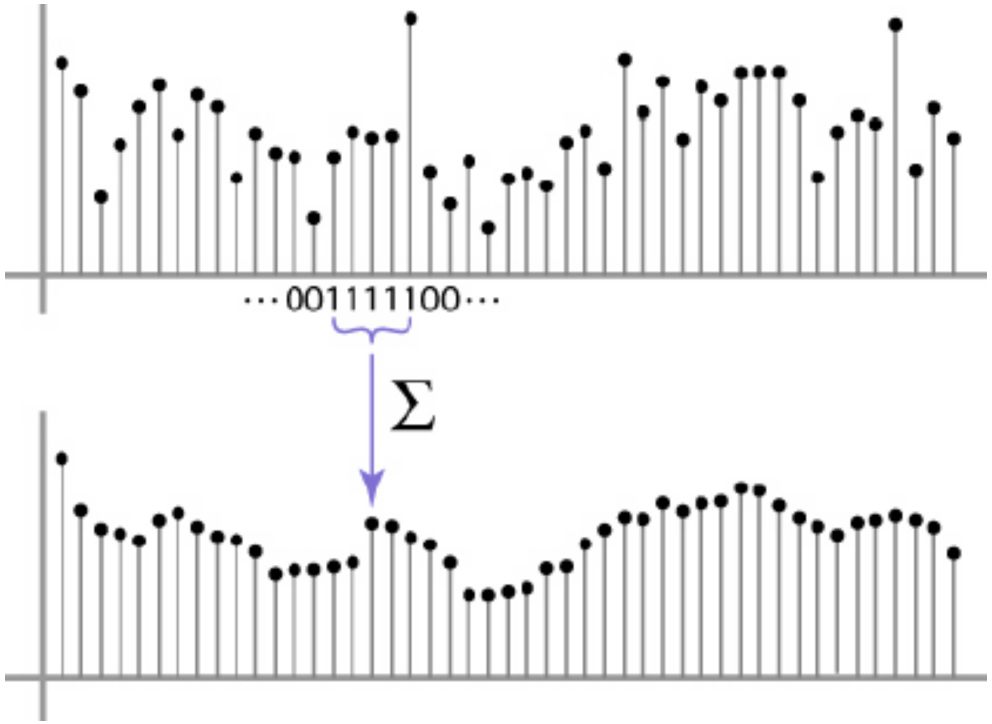




[Source: S. Marschner]

Noise reduction

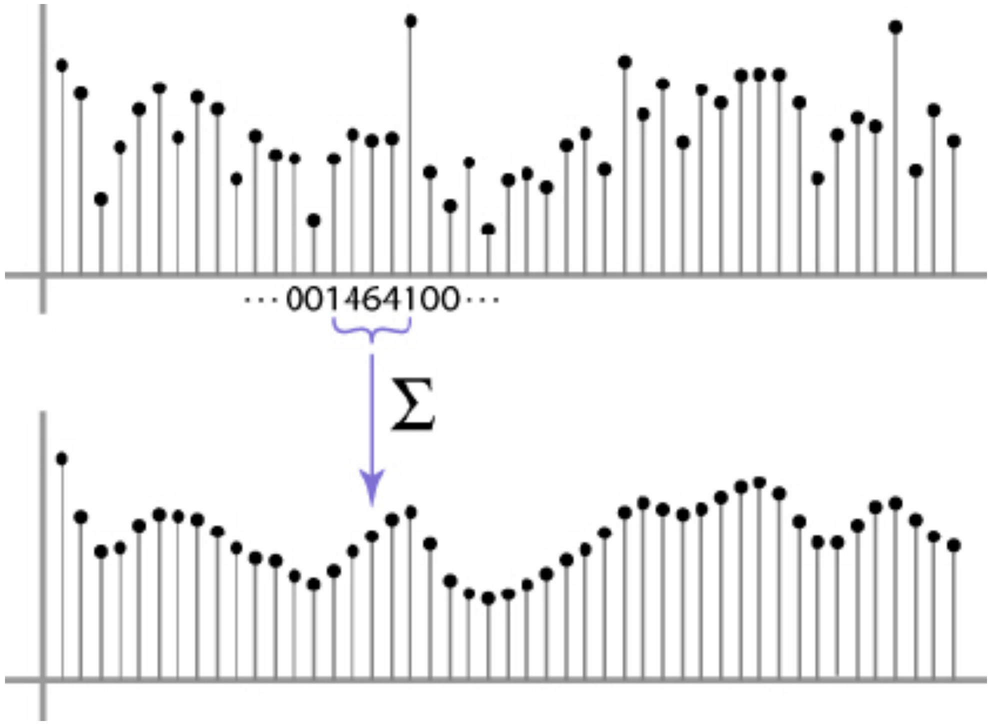
- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Moving average in 1D: [1, 1, 1, 1, 1]/5



[Source: S. Marschner]

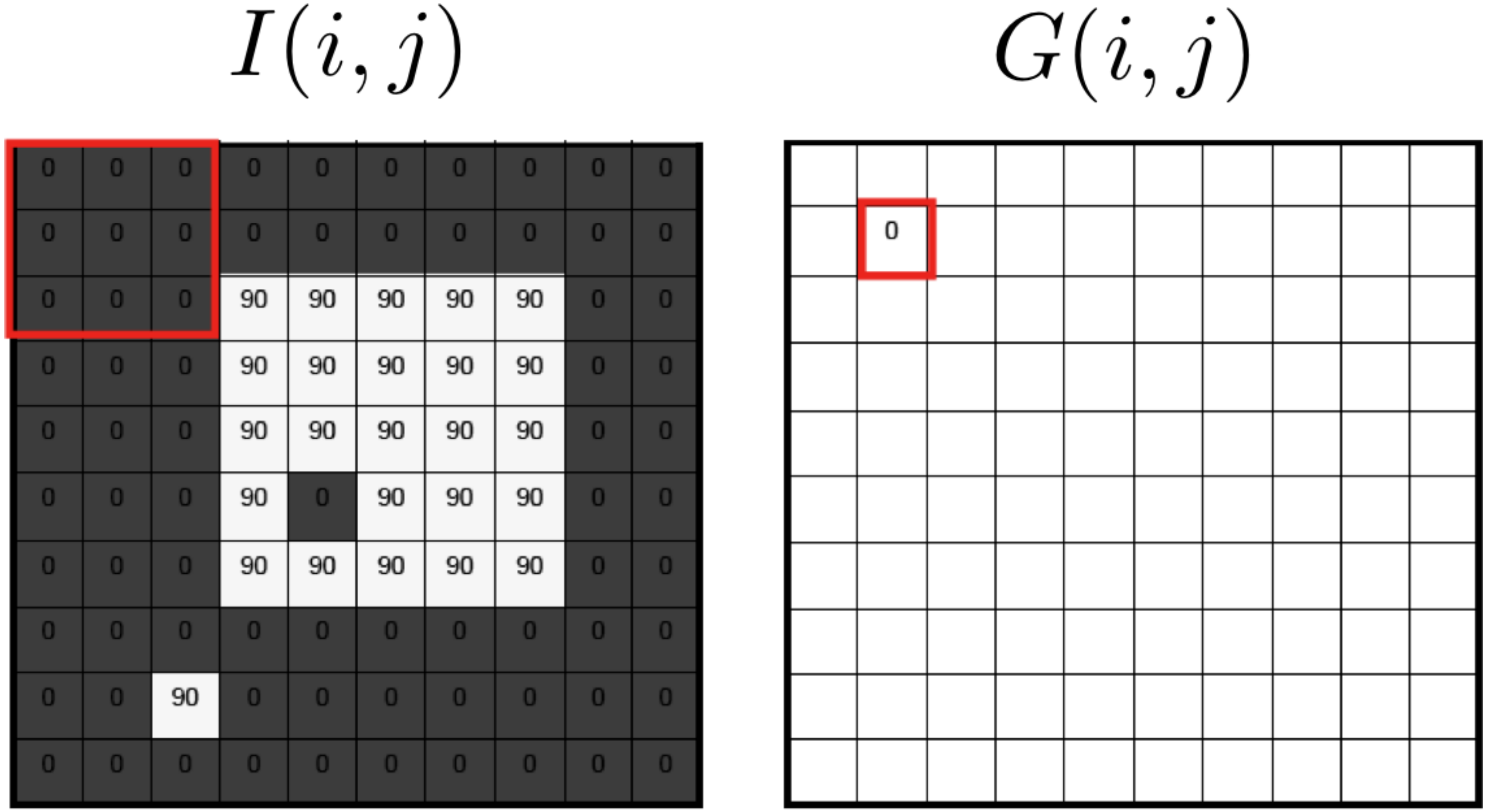
Noise reduction

- Simplest thing: replace each pixel by the average of its neighbors.
- This assumes that neighboring pixels are similar, and the noise to be independent from pixel to pixel.
- Non-uniform weights [1, 4, 6, 4, 1] / 16



[Source: S. Marschner]

Moving average in 2D



[Source: S. Seitz]

Moving average in 2D

$$I(i,j)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G(i,j)$$

	0	10							

[Source: S. Seitz]

Moving average in 2D

$I(i, j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i, j)$

	0	10	20						

[Source: S. Seitz]

Moving average in 2D

$I(i, j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i, j)$

	0	10	20	30					

[Source: S. Seitz]

Moving average in 2D

$I(i,j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i,j)$

	0	10	20	30	30				

[Source: S. Seitz]

Moving average in 2D

$I(i, j)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G(i, j)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

[Source: S. Seitz]

Linear Filtering: Correlation

Involves weighted combinations of pixels in small neighborhoods (avg. filter):

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

Linear Filtering: Correlation

Involves weighted combinations of pixels in small neighborhoods (avg. filter):

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

Linear Filtering: Correlation

Involves weighted combinations of pixels in small neighborhoods (avg. filter):

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

The output pixels value is determined as a weighted sum of input pixel values

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

The entries of the **weight kernel** or **mask** are often called the **filter coefficients**

Linear Filtering: Correlation

Involves weighted combinations of pixels in small neighborhoods (avg. filter):

$$G(i,j) = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)$$

The output pixels value is determined as a weighted sum of input pixel values

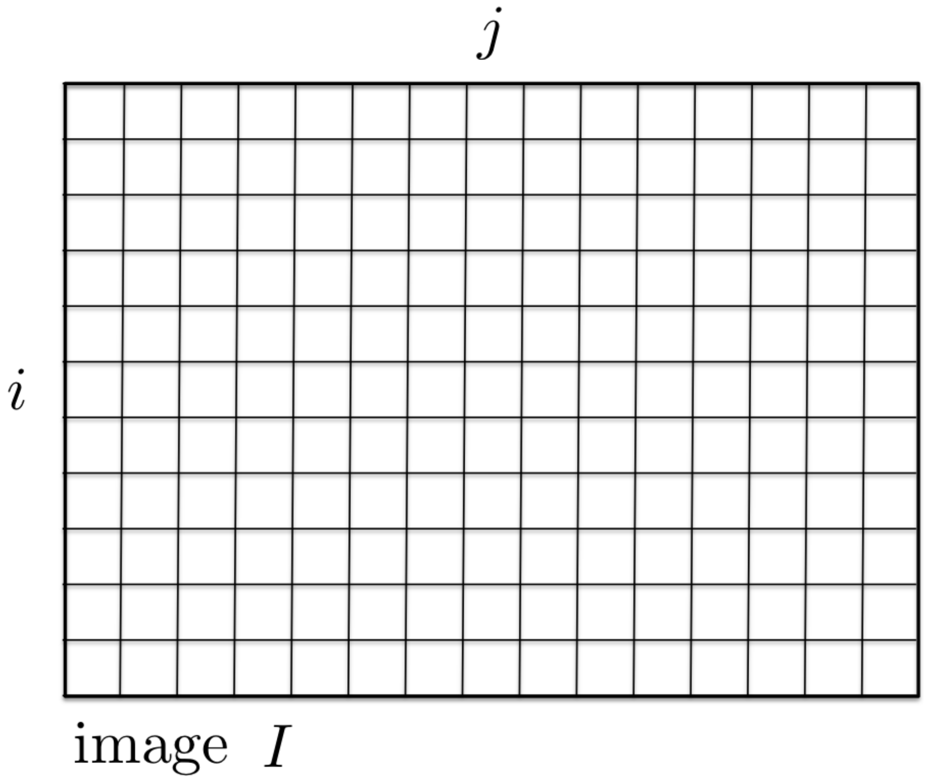
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

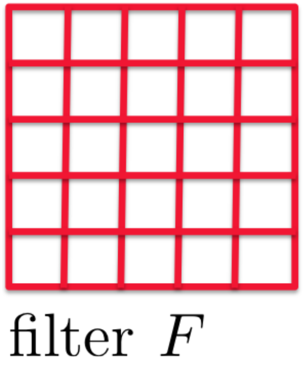
The entries of the **weight kernel** or **mask** are often called the **filter coefficients**

This operator is called the **correlation operator**

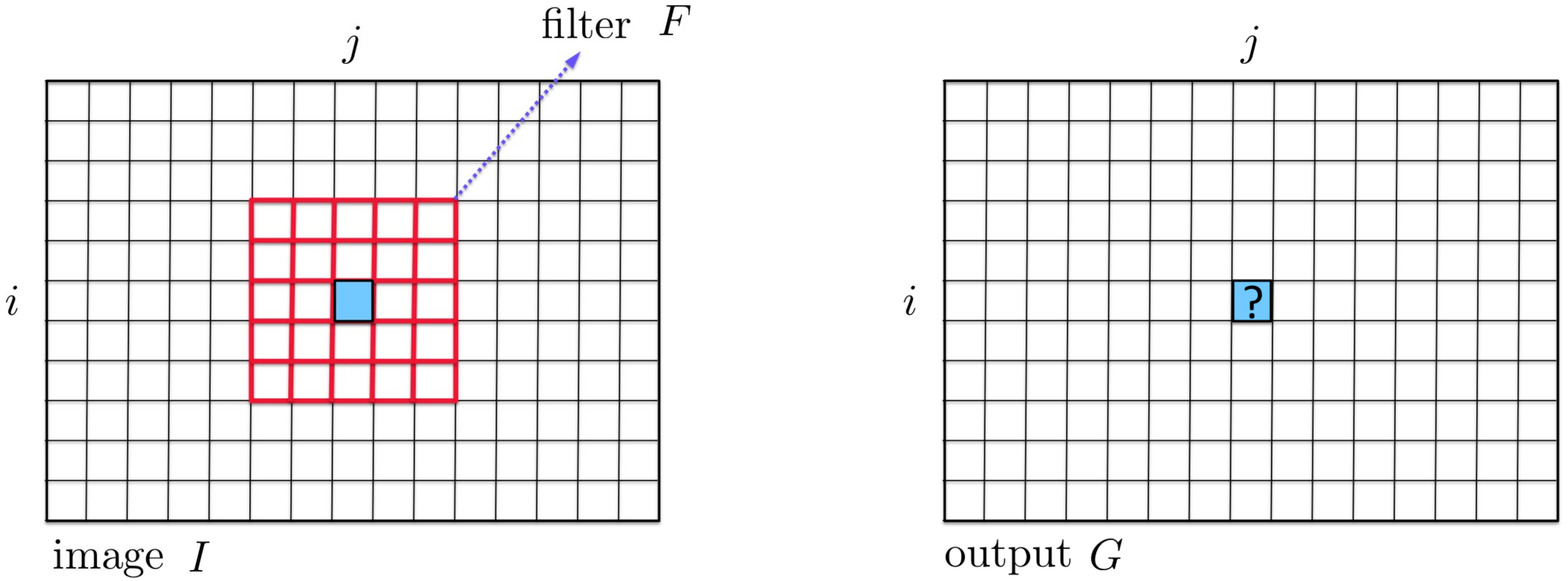
$$G = F \bigotimes I$$

Linear Filtering: Correlation

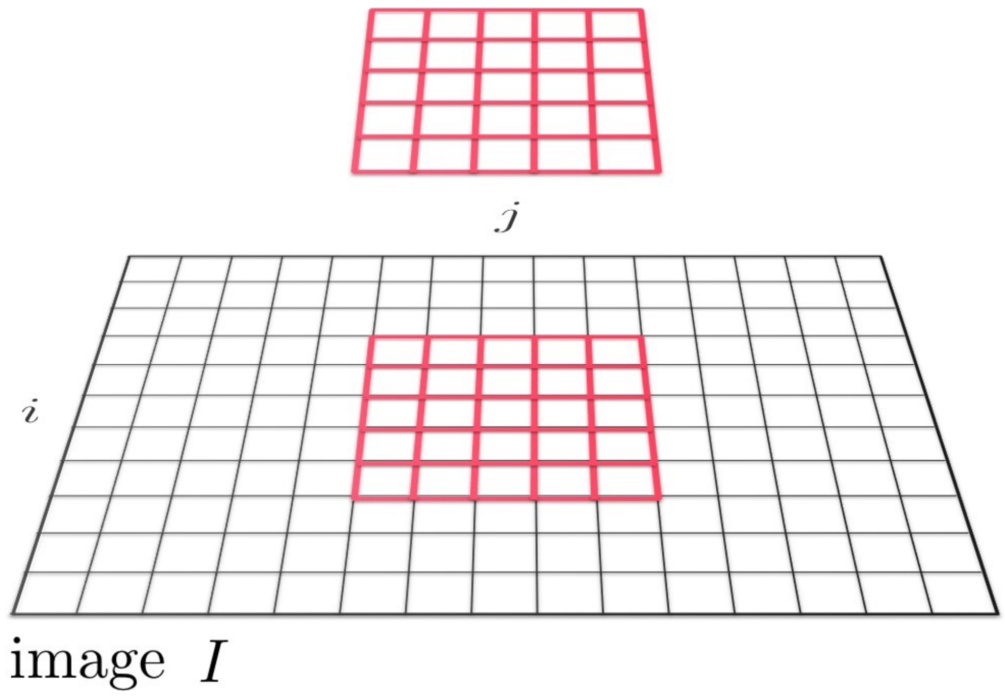


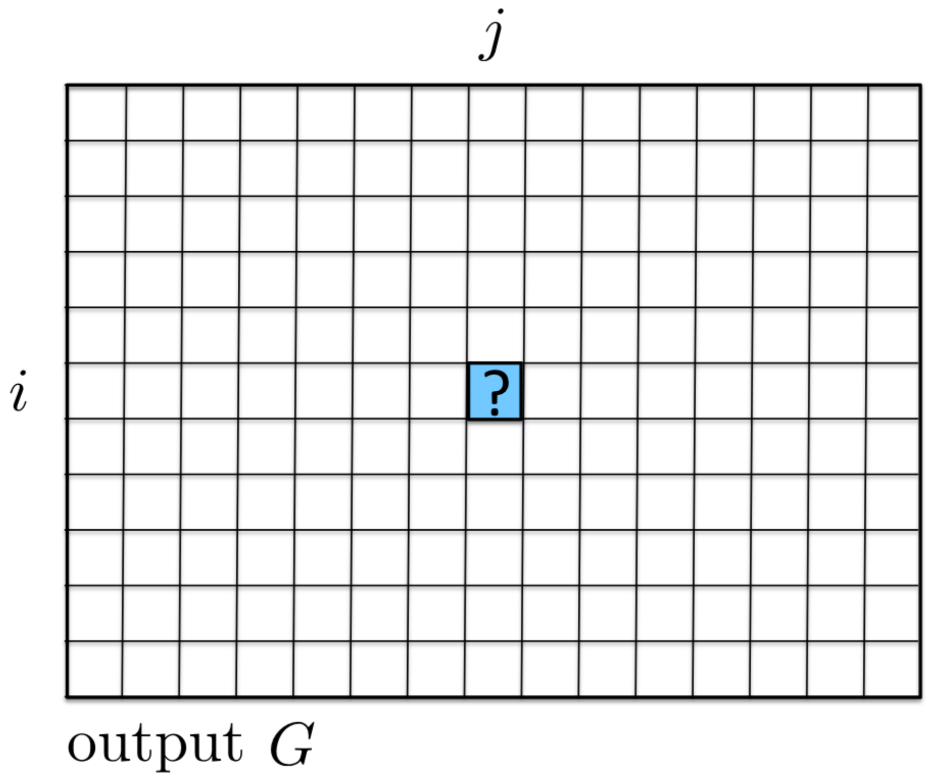


Linear Filtering: Correlation

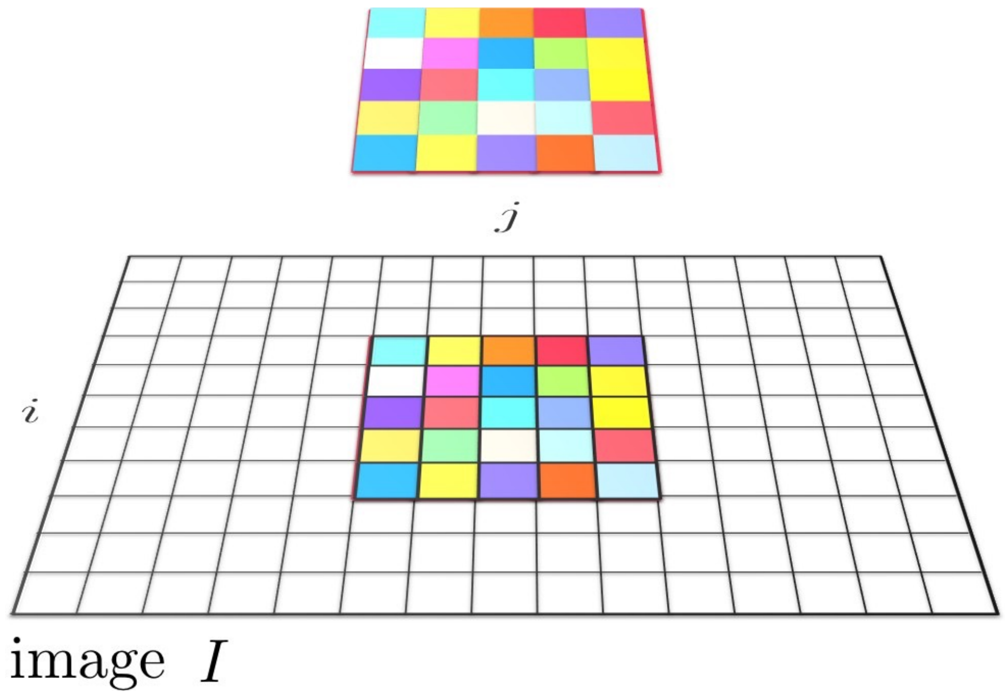


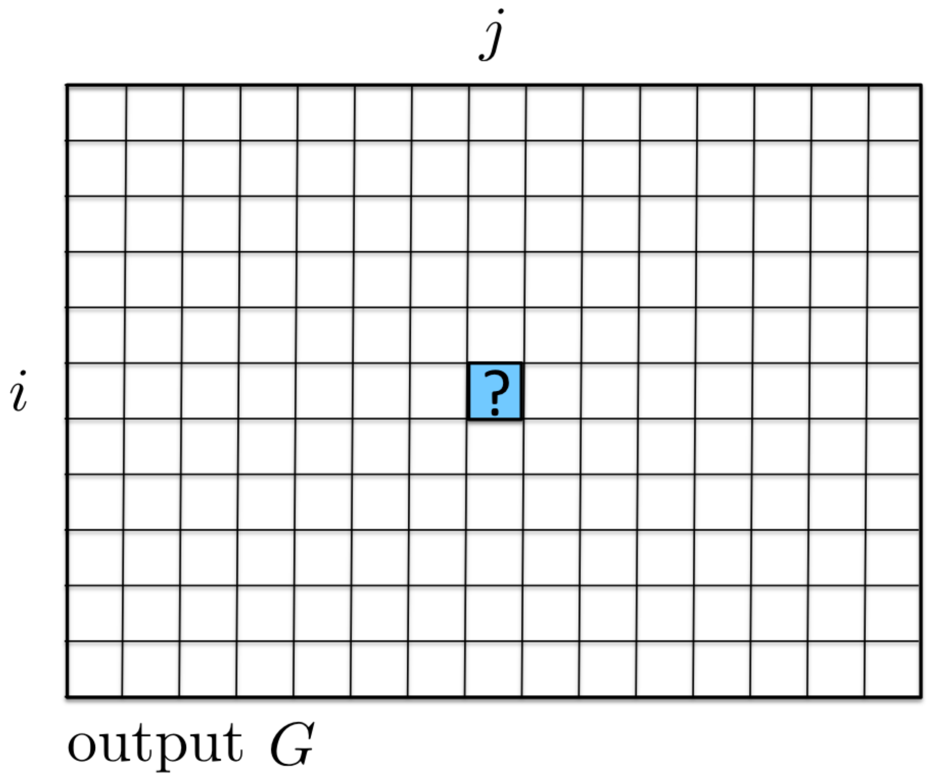
Linear Filtering: Correlation



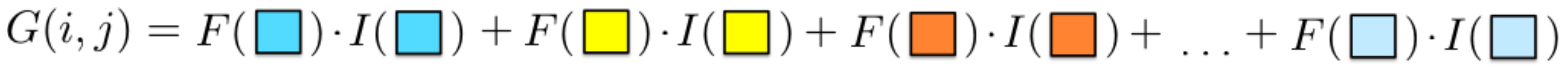


Linear Filtering: Correlation

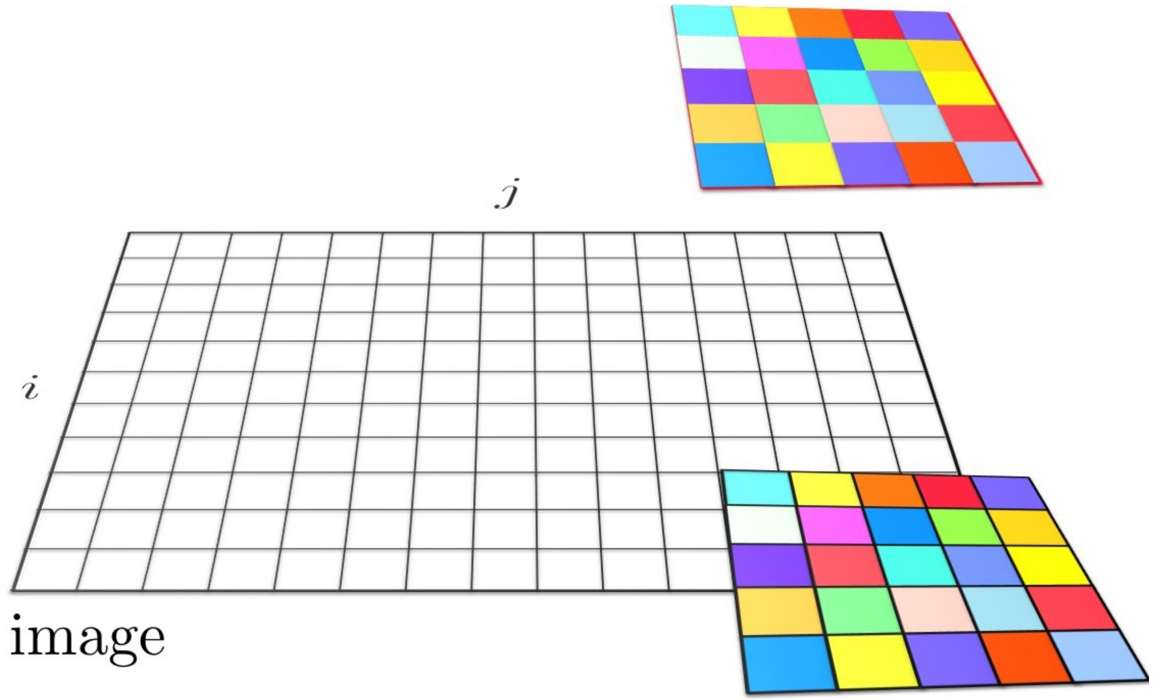


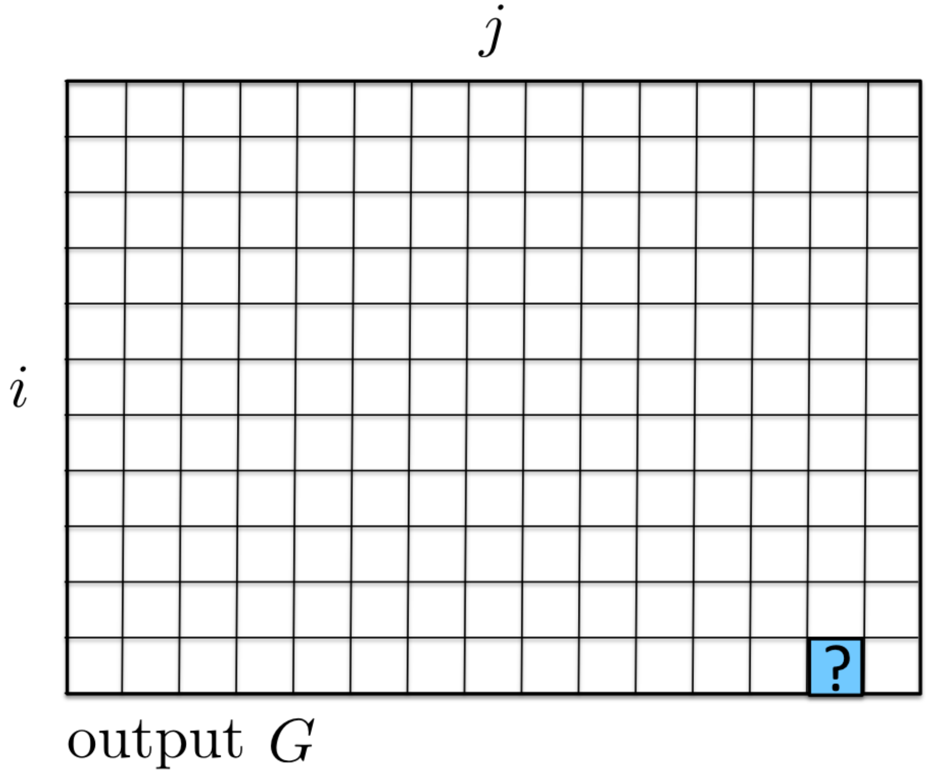


$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



Linear Filtering: Correlation





What happens at the borders?

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

$$G(i,j) = F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + F(\square) \cdot I(\square) + \ldots + F(\square) \cdot I(\square)$$

Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
  - depends on how you implement it

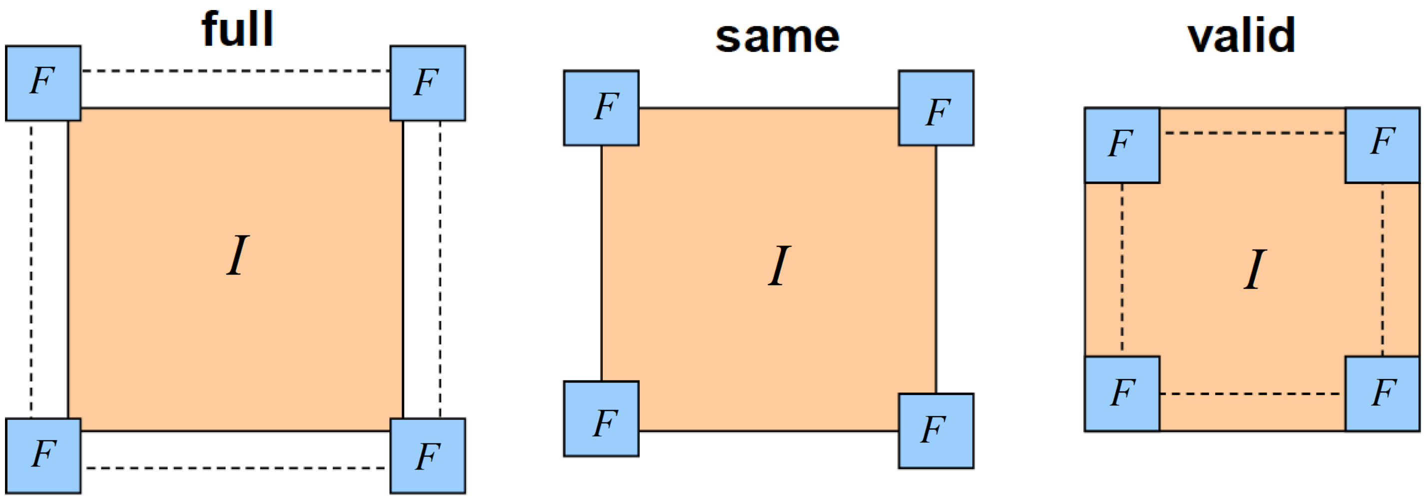
- Scipy: scipy.signal.convolve2d
  - mode = 'full' output size is bigger than the image
  - mode = 'same': output size is same as I
  - mode = 'valid': output size is smaller than the image

[Source: S. Lazebnik]

Boundary Effects

- What happens at the border of the image? What's the size of the output matrix?
  - depends on how you implement it

- Scipy: scipy.signal.convolve2d
  - mode = 'full' output size is bigger than the image
  - mode = 'same': output size is same as I
  - mode = 'valid': output size is smaller than the image



[Source: S. Lazebnik]

Correlation Example

What's the result?

Original

0	0	0
0	1	0
0	0	0

?

Correlation Example

What's the result?

Original

0	0	0
0	1	0
0	0	0

Filtered
(no change)

[Source: D. Lowe]

Correlation Example

What's the result?

Original

0	0	0
0	0	1
0	0	0

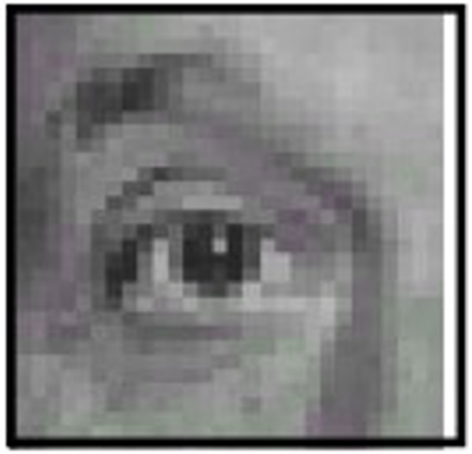
?

[Source: D. Lowe]

Correlation Example

What’s the result?

0	0	0
0	0	1
0	0	0



[Source: D. Lowe]

Correlation Example

What's the result?

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

[Source: D. Lowe]

Correlation Example

What's the result?

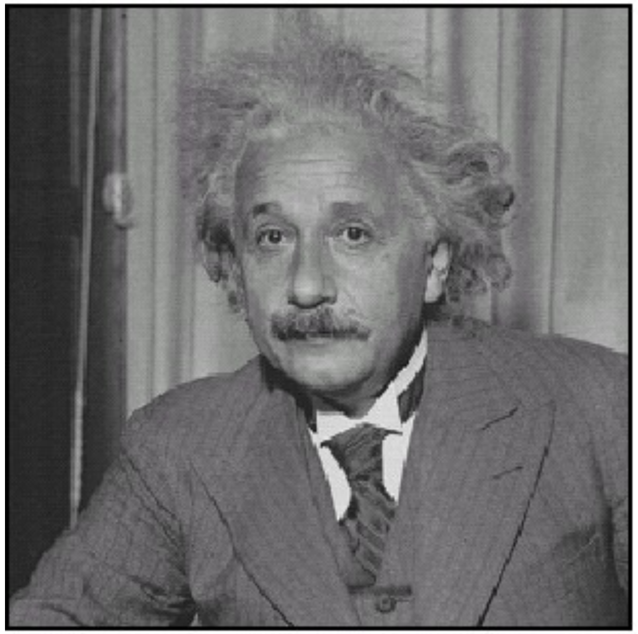
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

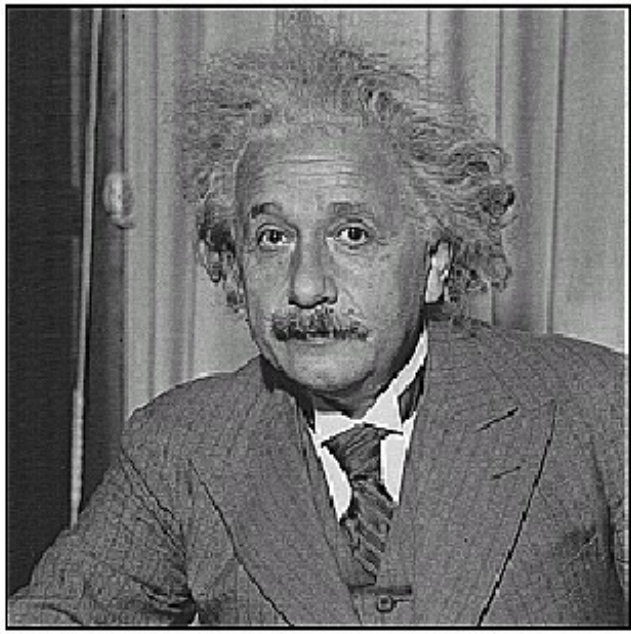
Sharpening Filter

[Source: D. Lowe]

Sharpening



before



after

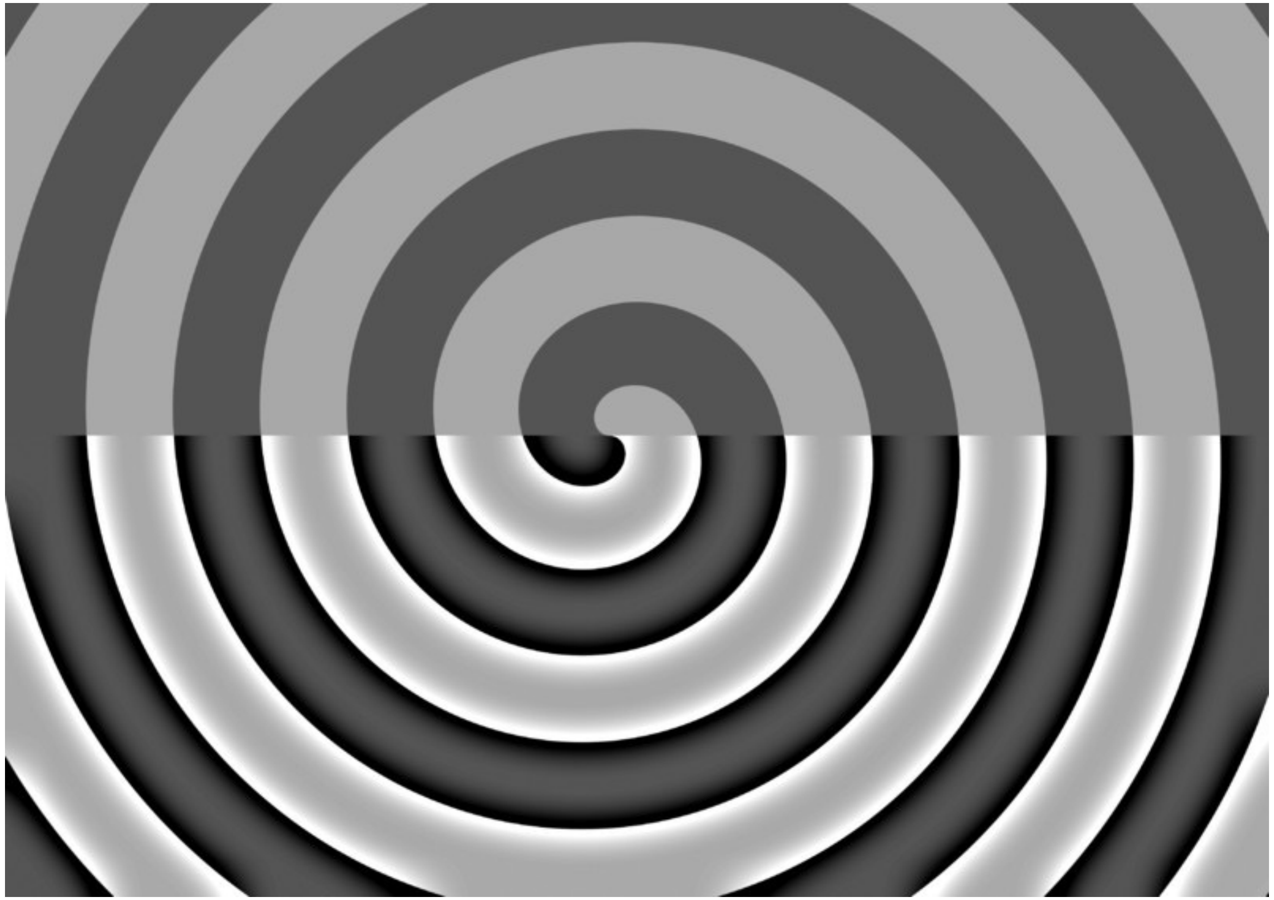
This is a prelude to edge detection (next time)!

[Source: D. Lowe]

Sharpening

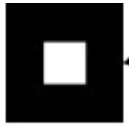
unfiltered

filtered



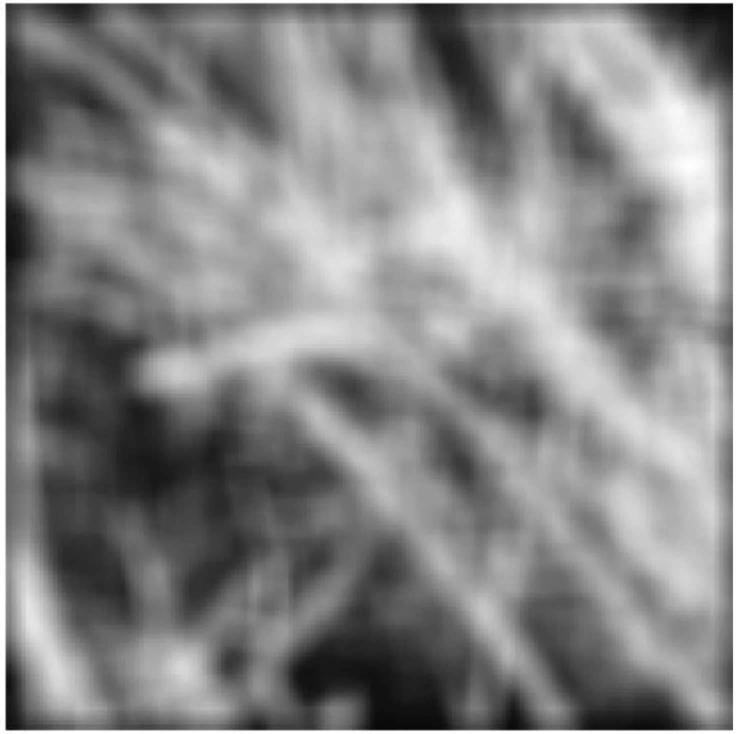
[Source: N. Snavely]

Smoothing by averaging



depicts box filter:
white = high value, black = low value

original



filtered

What if the filter size was 5 x 5 instead of 3 x 3?

[Source: K. Grauman]

Gaussian filter

What if we want nearest neighboring pixels to have the most influence on the output?

Removes high-frequency components from the image (low-pass filter).

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

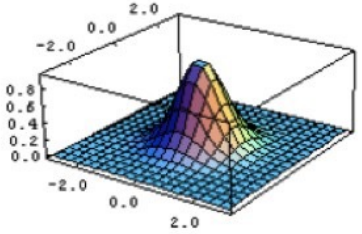
$$I(i,j)$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$F(i,j)$$

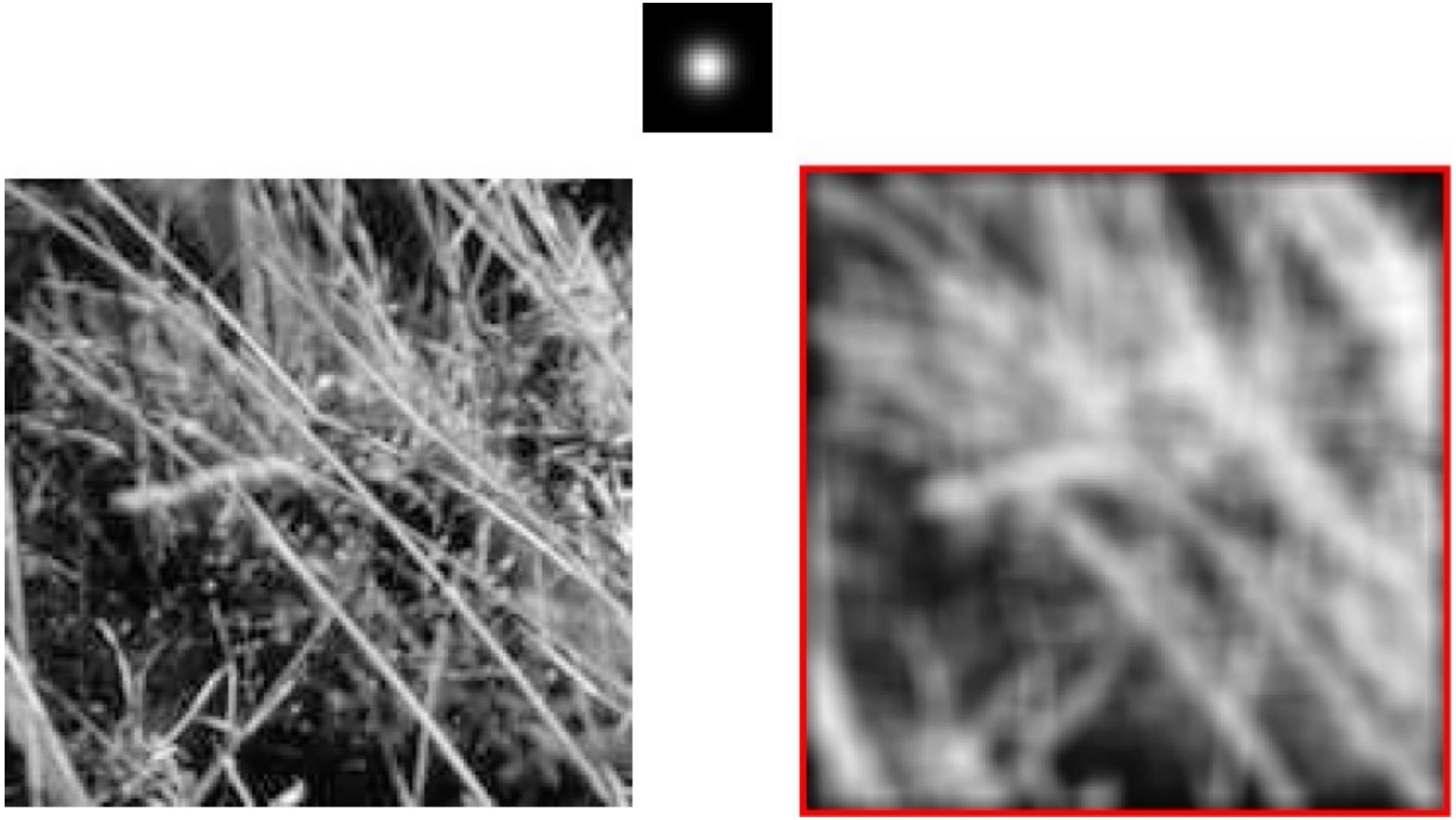
This kernel is an approximation of a 2d Gaussian function:

$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



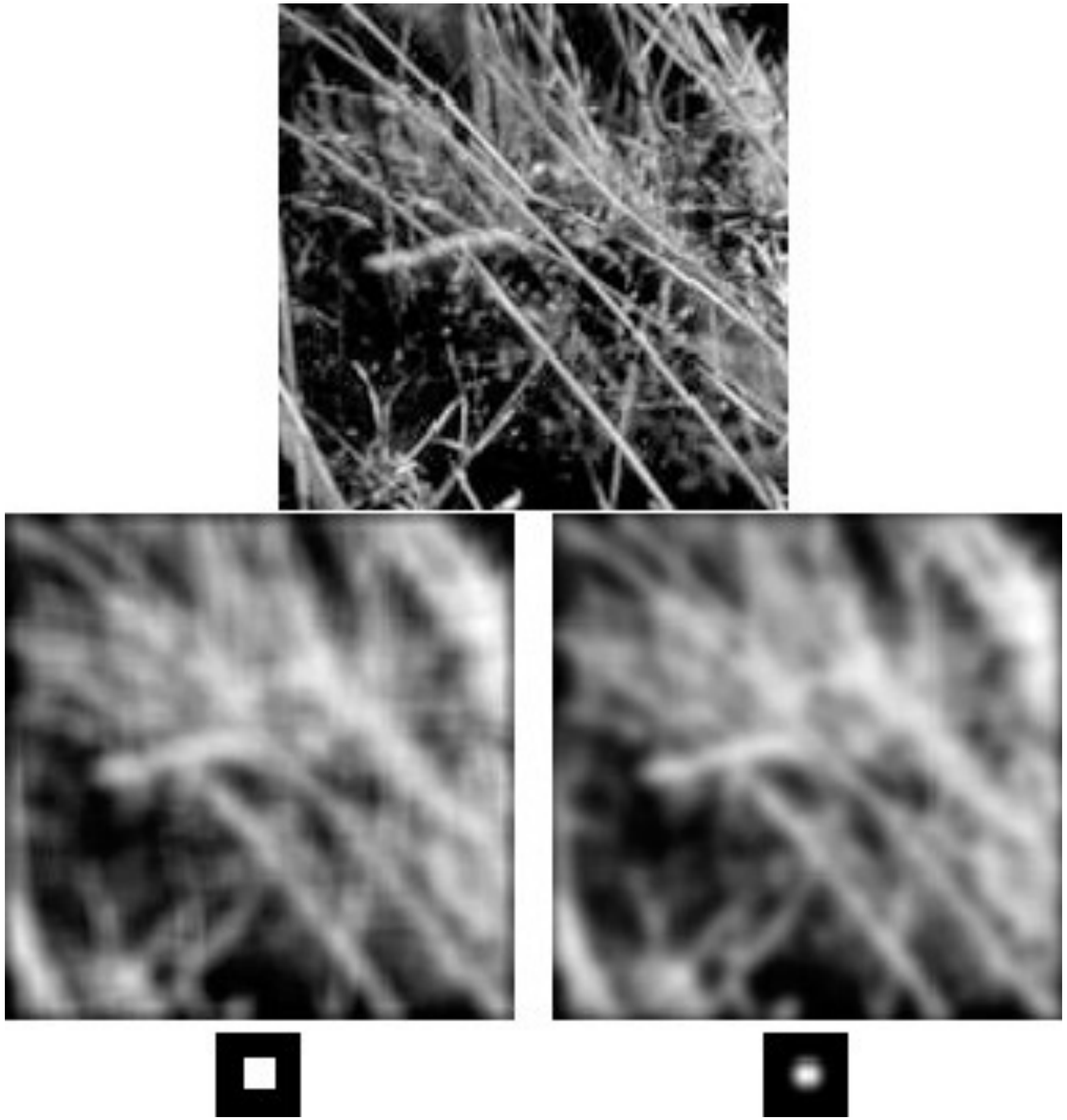
[Source: S. Seitz]

Gaussian filter



[Source: K. Grauman]

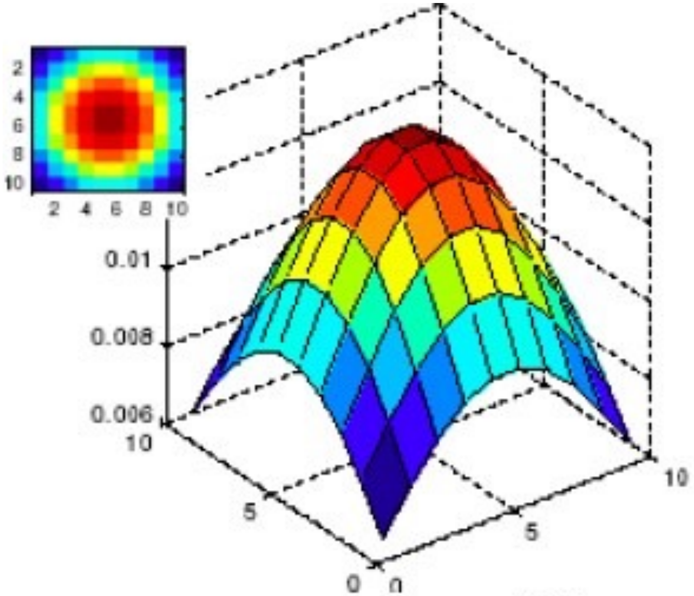
Mean vs. Gaussian filter



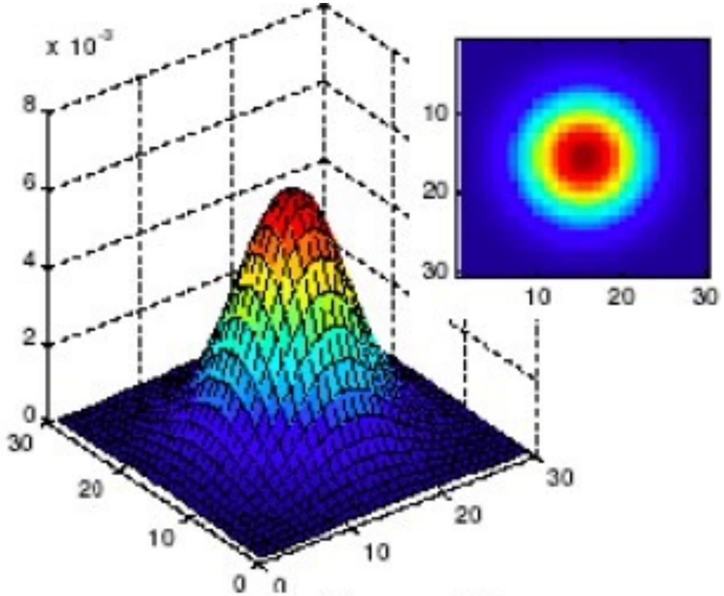
[Source: K. Grauman]

Gaussian filter parameters

Size of filter or mask: Gaussian function has infinite support, but discrete filters use finite kernels.



σ = 5 with 10 x 10 kernel

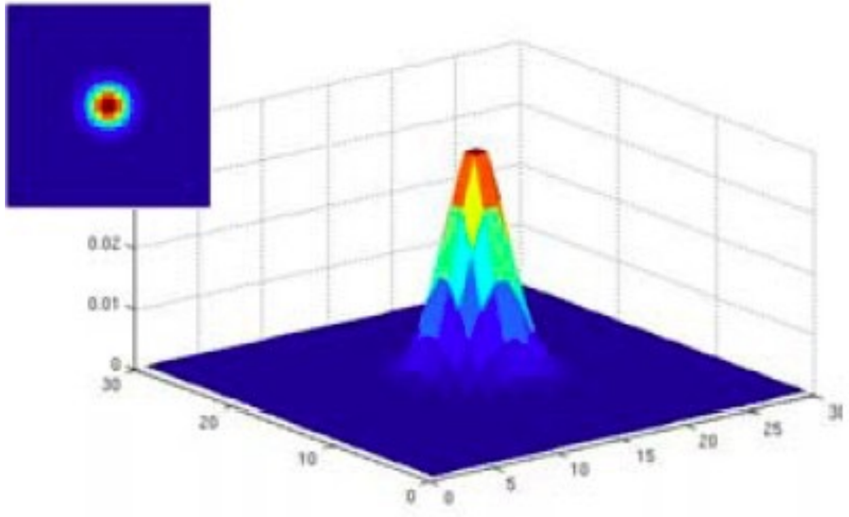


σ = 5 with 30 x 30 kernel

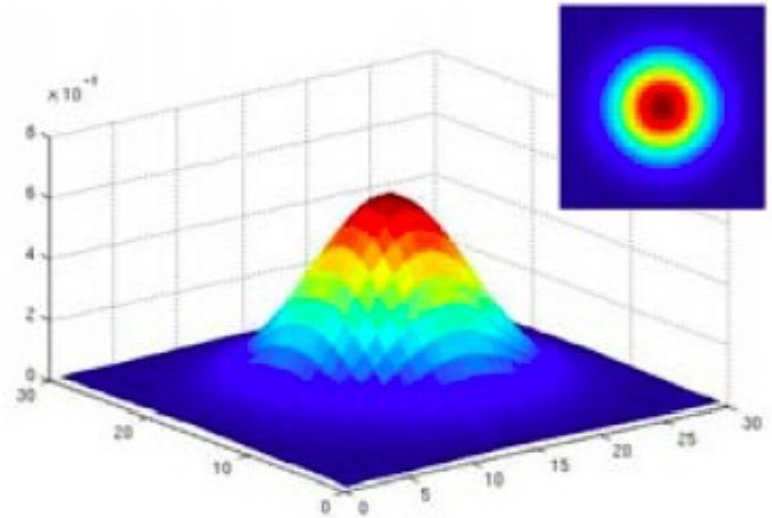
[Source: K. Grauman]

Gaussian filter parameters

Variance of the Gaussian: determines extent of smoothing.



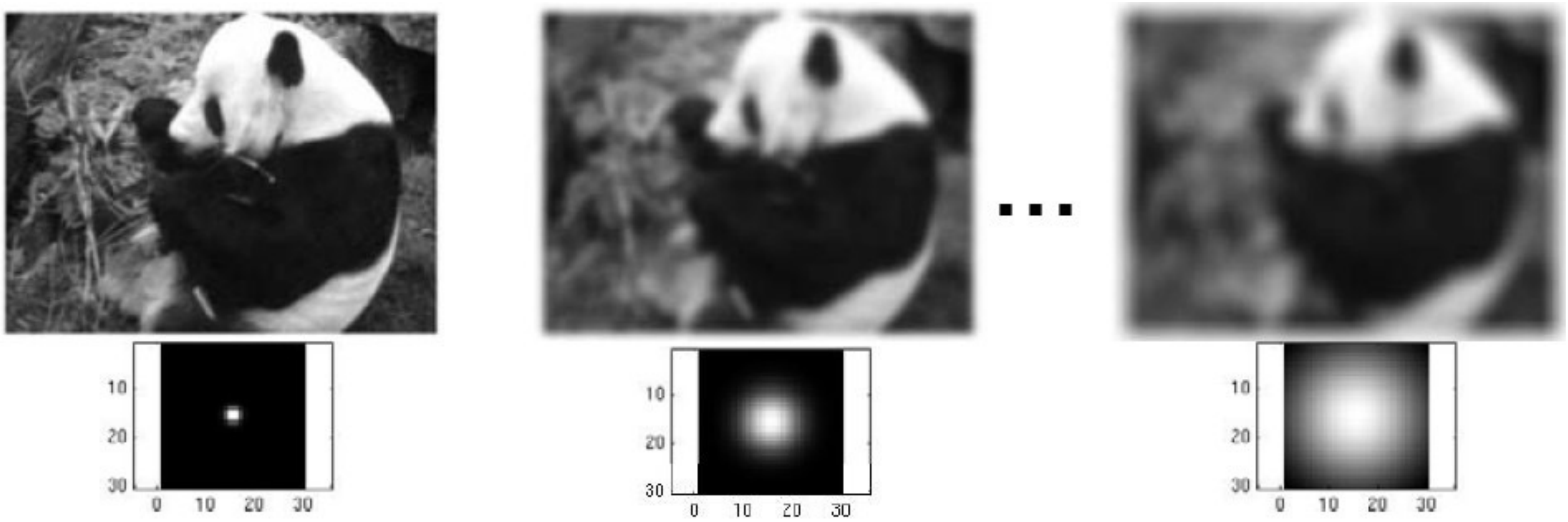
σ = 2 with
30 x 30
kernel



σ = 5 with
30 x 30
kernel

[Source: K. Grauman]

Gaussian filter parameters



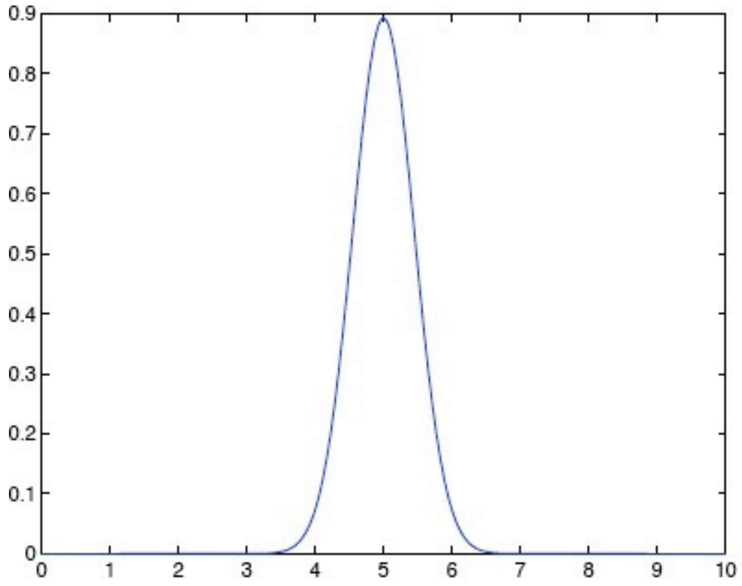

```
for sigma=1:3:10
   h = fspecial('gaussian‘, fsize, sigma);
   out = imfilter(im, h);
   imshow(out);
   pause;
end
```

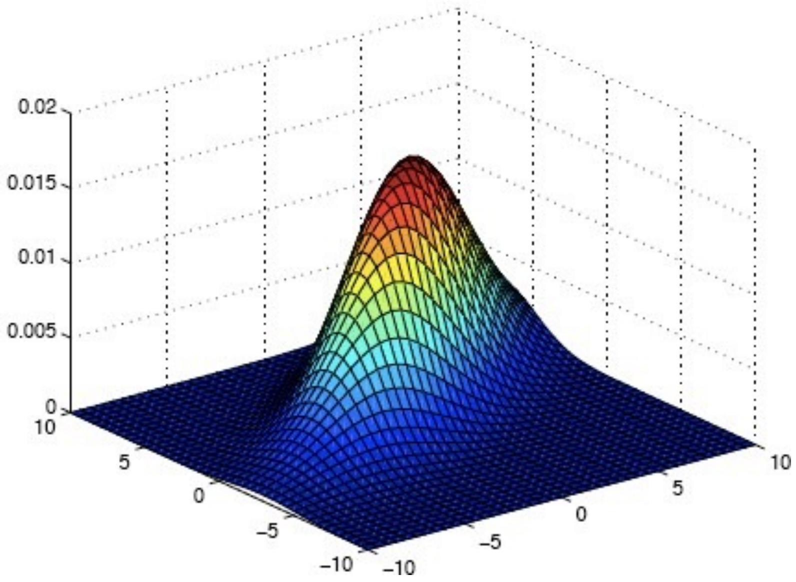
[Source: K. Grauman]

Is this the most general Gaussian?

No, the most general form is anisotropic (i.e., not symmetric) $x \in \Re^d$

$$\mathcal{N}(\mathbf{x};\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$





But the simplified version is typically used for filtering.

Properties of smoothing kernels

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.

Properties of smoothing kernels

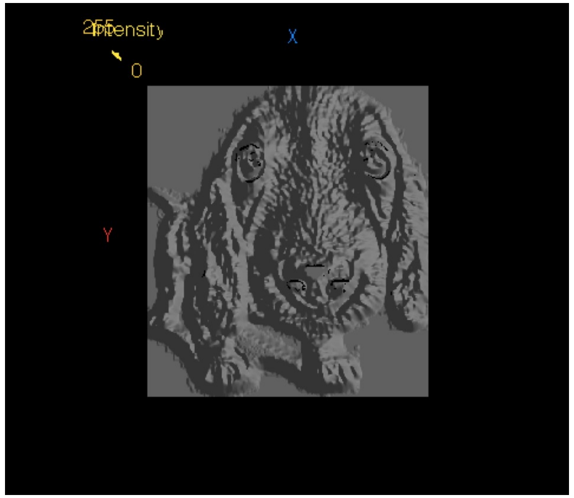
- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.

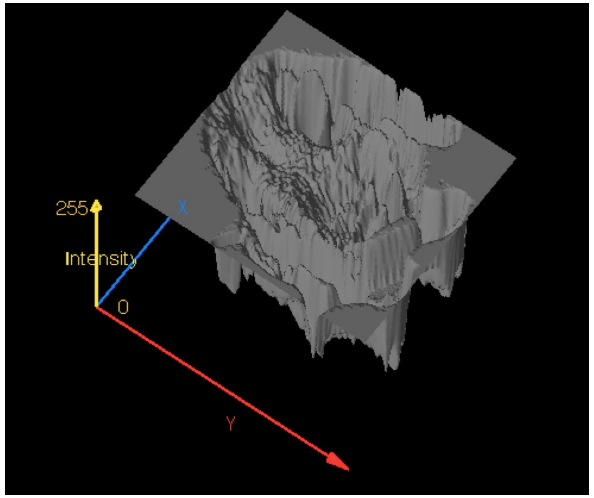
Properties of smoothing kernels

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.
- What is frequency in this context?

Properties of smoothing kernels

- All values are positive.
- They all sum to 1 to prevent re-scaling of the image.
- Remove high-frequency components; low-pass filter.
- What is frequency in this context?
- Edges!





Finding Waldo

How can we use what we just learned to find Waldo?

Finding Waldo

Correlation?

Filter F

Interlude: Correlation in Matrix form

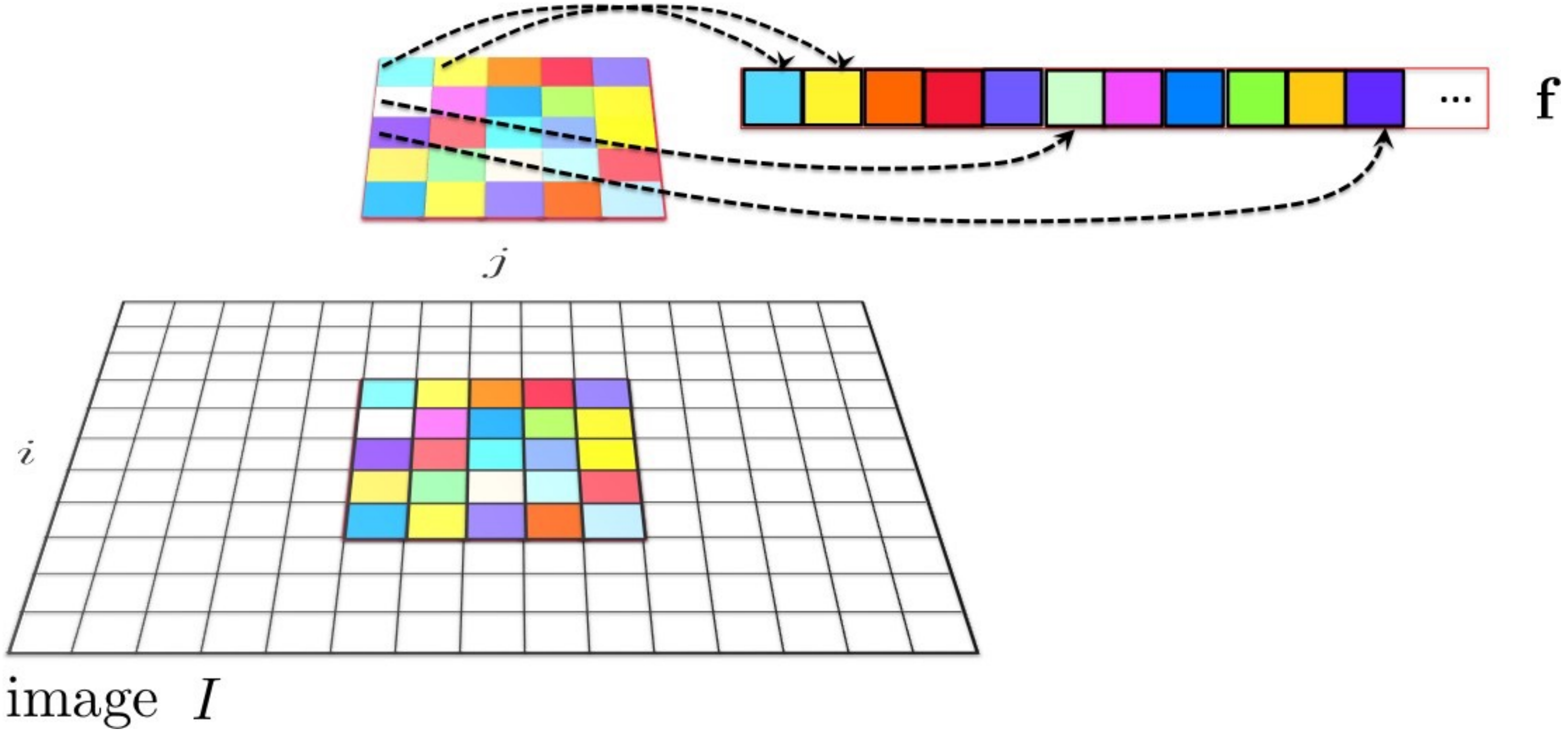
Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

Can we write that in a more compact form (with vectors)?

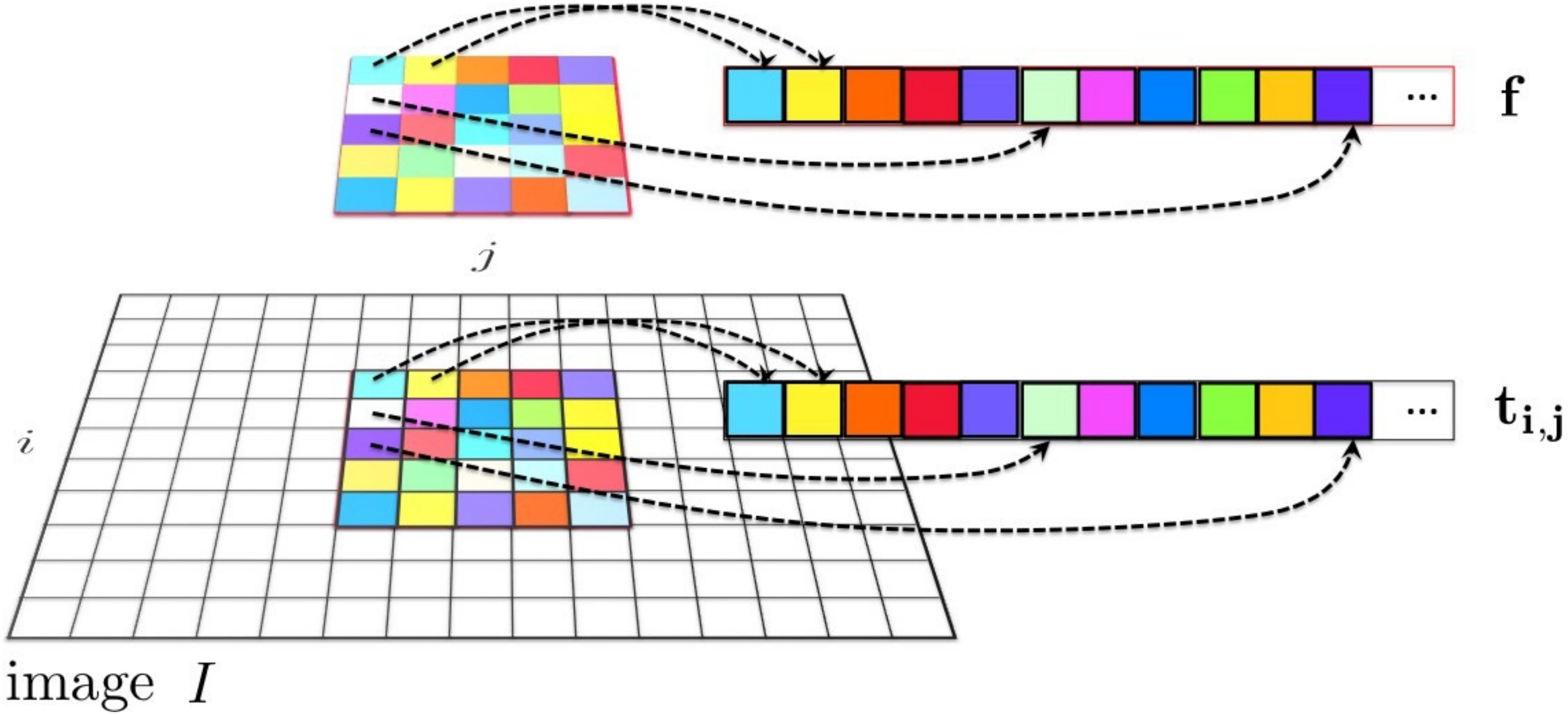
Interlude: Correlation in Matrix form

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



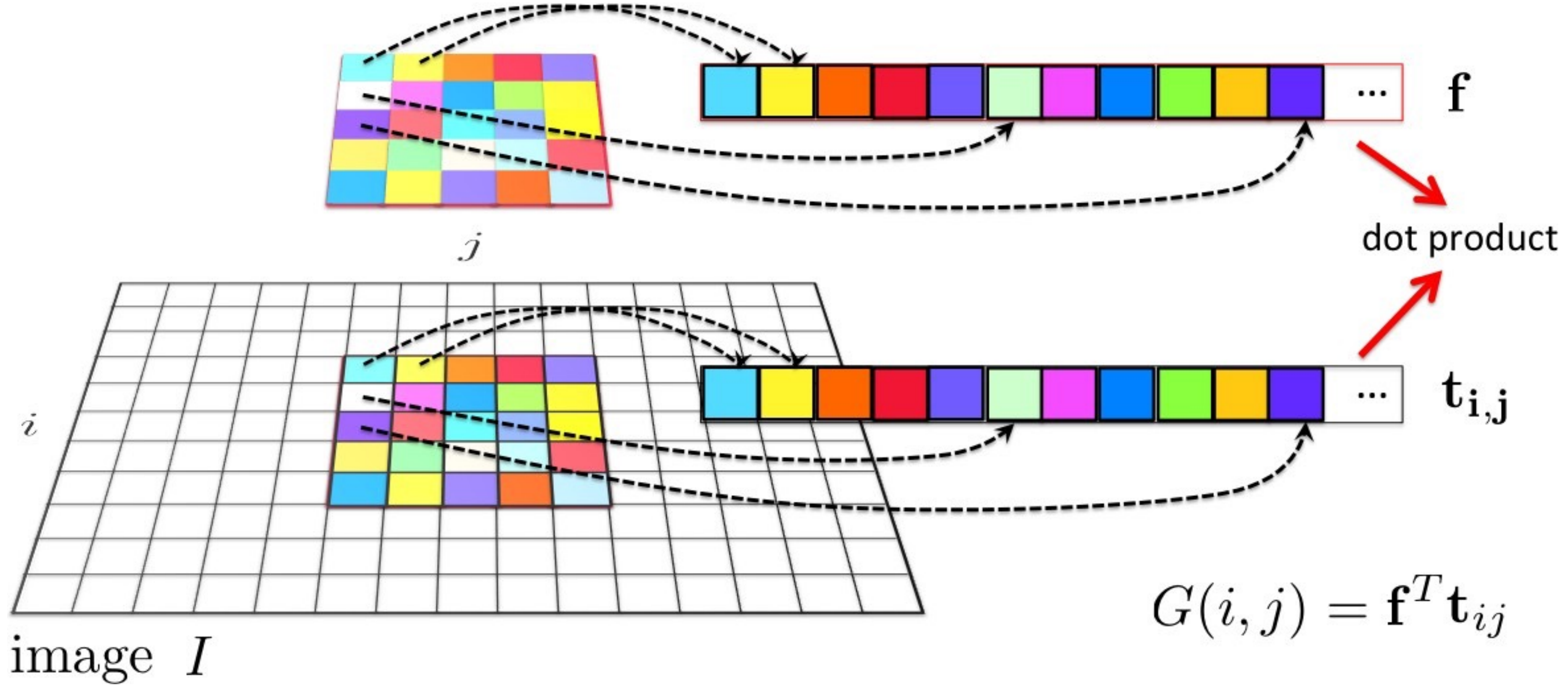
Interlude: Correlation in Matrix form

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



Interlude: Correlation in Matrix form

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$



$$G(i,j) = \mathbf{f}^T \mathbf{t}_{ij}$$

Interlude: Correlation in Matrix form

Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

Can we write that in a more compact form (with vectors)?

Interlude: Correlation in Matrix form

Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

Can we write that in a more compact form (with vectors)?

Define $\mathbf{f} = F(:), \quad T_{ij} = I(i-k:i+k, j-k:j+k), \quad \mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f} \cdot \mathbf{t}_{ij}$$

Where $\cdot$ is a dot product

Interlude: Correlation in Matrix form

Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

Can we write that in a more compact form (with vectors)?

Define $\mathbf{f} = F(:), \quad T_{ij} = I(i-k:i+k, j-k:j+k), \quad \mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f} \cdot \mathbf{t}_{ij}$$

Where $\cdot$ is a dot product

Can we write full correlation G = F $\otimes$ I in matrix form?

Interlude: Correlation in Matrix form

Remember correlation:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i+u, j+v)$$

Can we write that in a more compact form (with vectors)?

Define $\mathbf{f} = F(:), \quad T_{ij} = I(i-k : i+k, j-k : j+k), \quad \mathbf{t}_{ij} = T_{ij}(:)$

$$G(i,j) = \mathbf{f} \cdot \mathbf{t}_{ij}$$

Finding Waldo: How could we ensure to get the best “score” (e.g. 1) for an image crop that looks exactly like our filter?

Interlude: Correlation in Matrix form

Finding Waldo: How could we ensure to get the best "score" (e.g. 1) for an image crop that looks exactly like our filter?

Normalized cross-correlation:

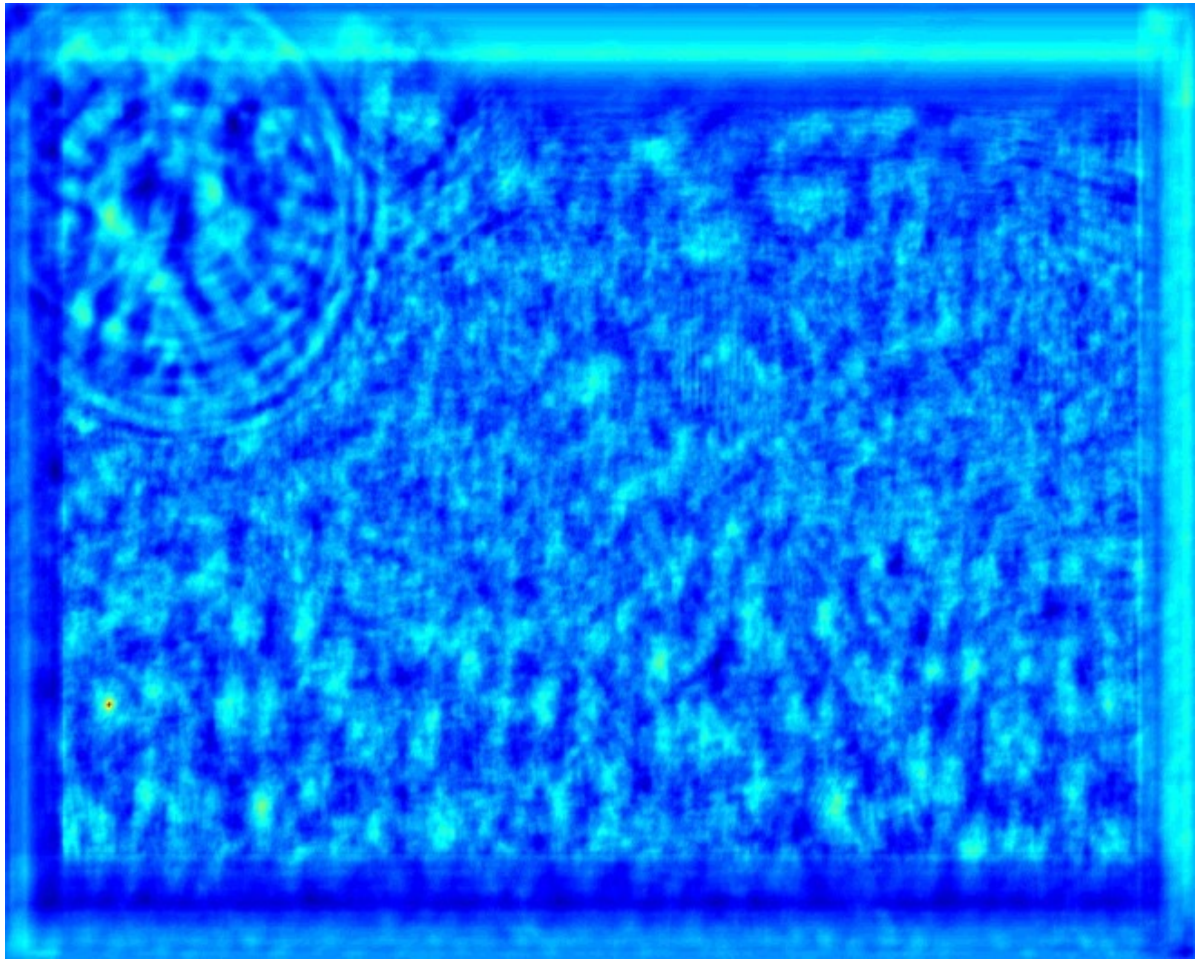
$$G(i, j) = \frac{\mathbf{f}^T \mathbf{t}_{ij}}{||\mathbf{f}|| \, ||\mathbf{t}_{ij}||}$$

Back to Finding Waldo

Image

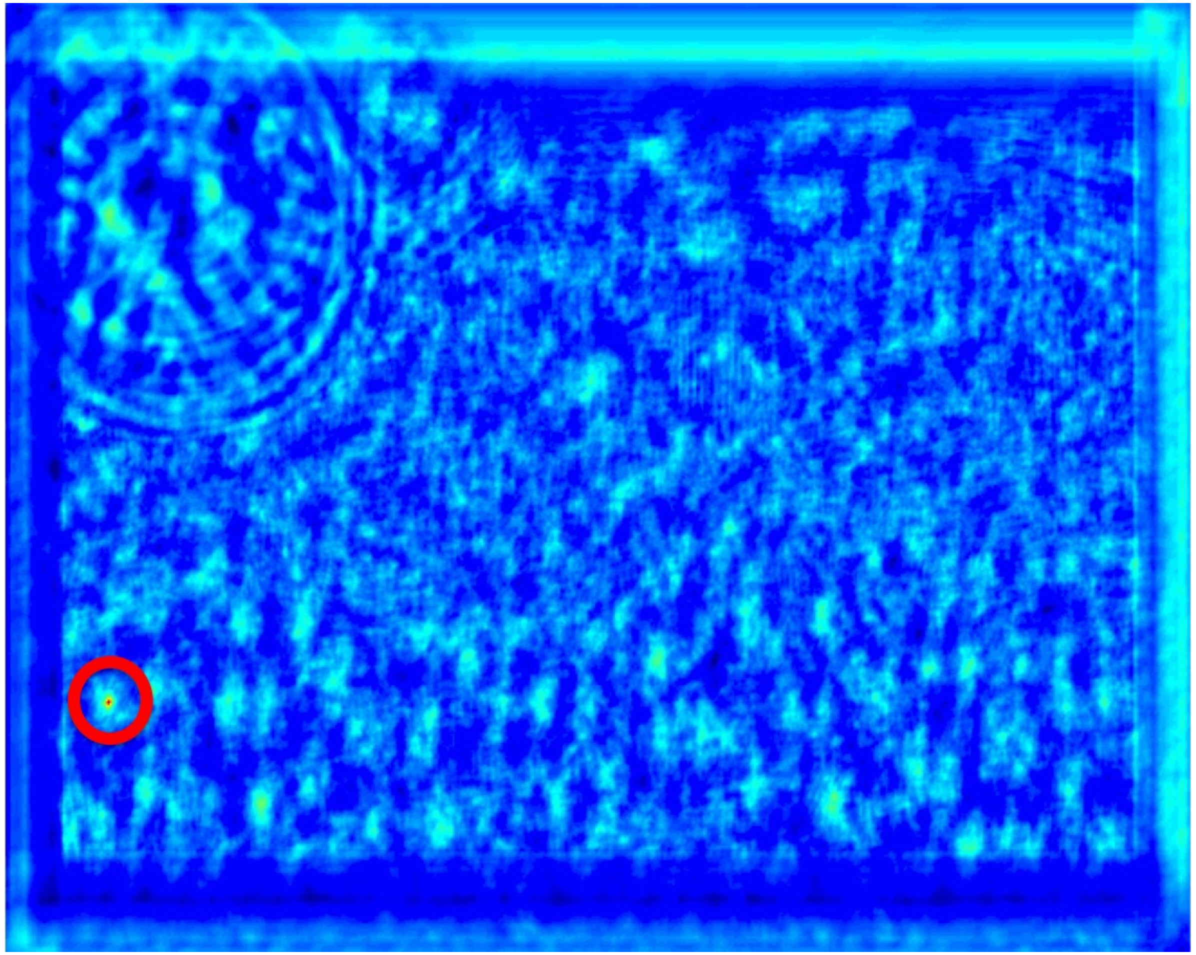
Filter

Back to Finding Waldo



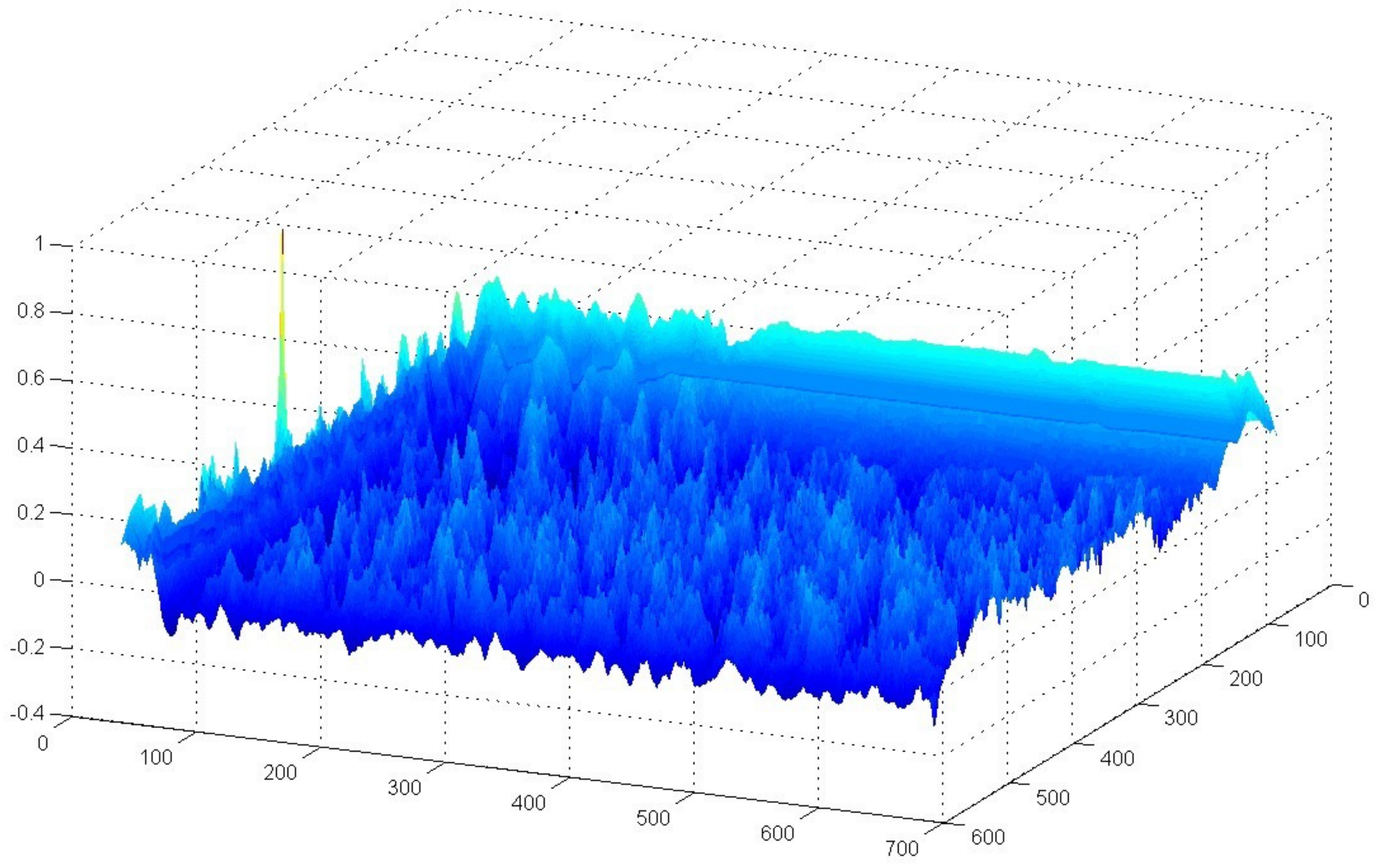
Result of normalized cross-correlation

Back to Finding Waldo



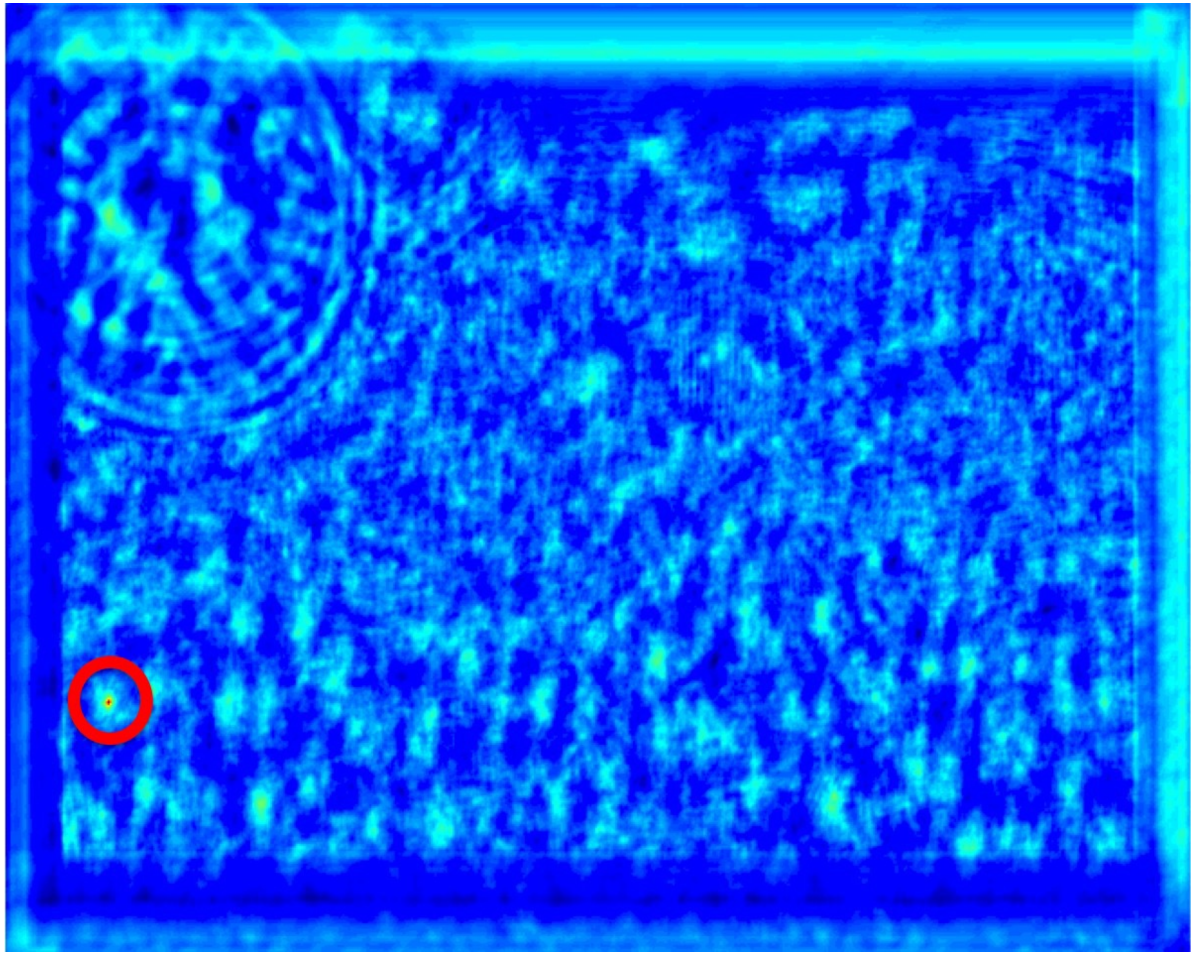
Result of normalized cross-correlation

Back to Finding Waldo



Find the highest peak

Back to Finding Waldo



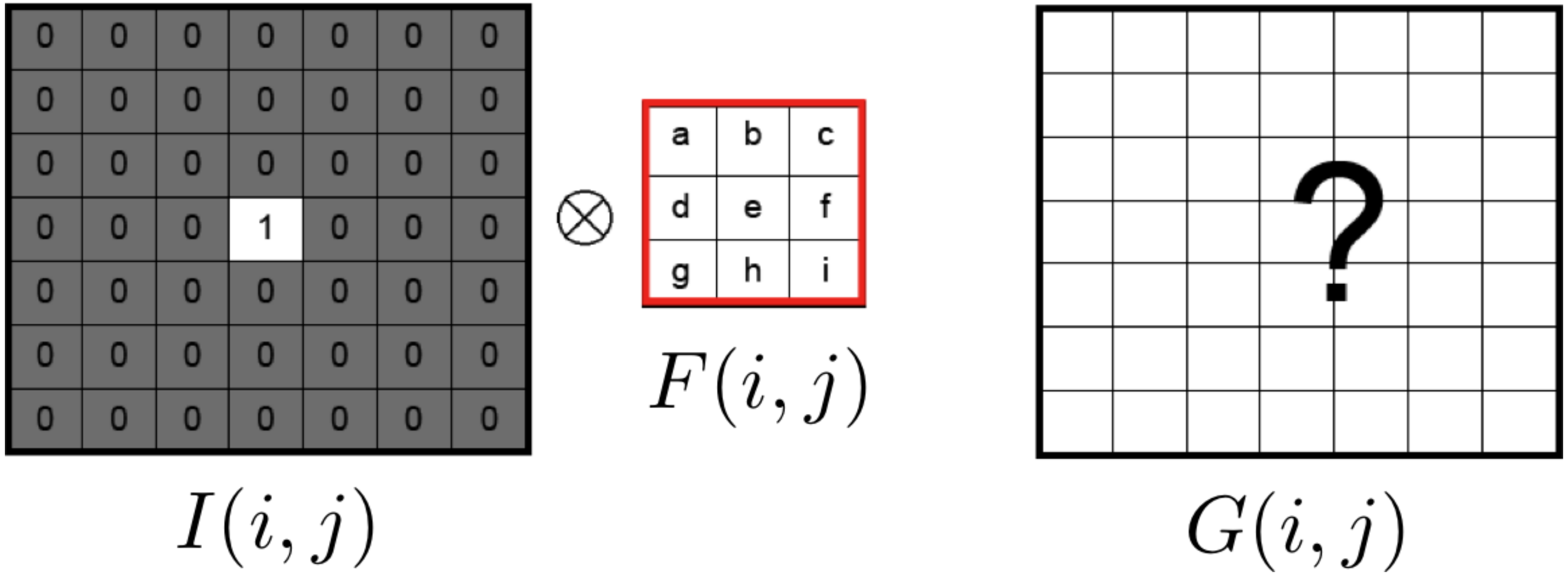
Find the highest peak

Back to Finding Waldo

With a bounding box (rectangle the size of the template) at the point...

Correlation example

What is the result of filtering the impulse signal (image) I with an arbitrary filter F?



[Source: K. Grauman]

Convolution

Convolution operator

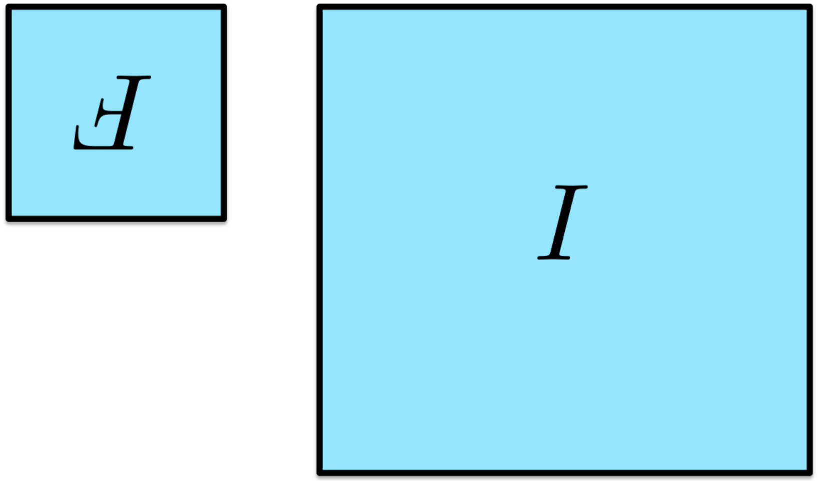
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i-u, j-v)$$

Convolution

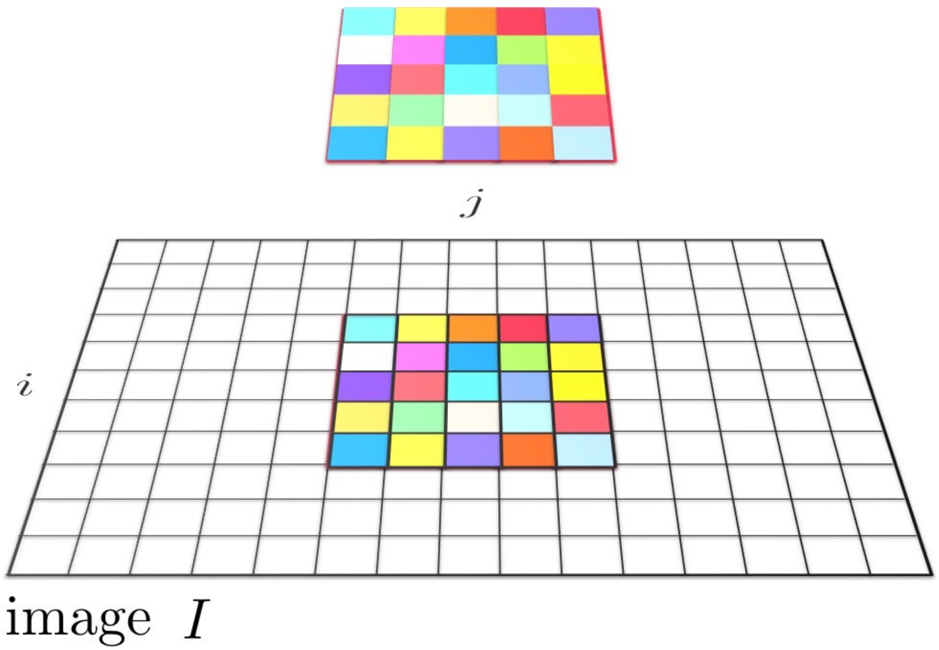
Convolution operator

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} F(u,v) \cdot I(i-u, j-v)$$

Equivalent to flipping the filter in both dimensions (bottom to top, right to left) and apply correlation.

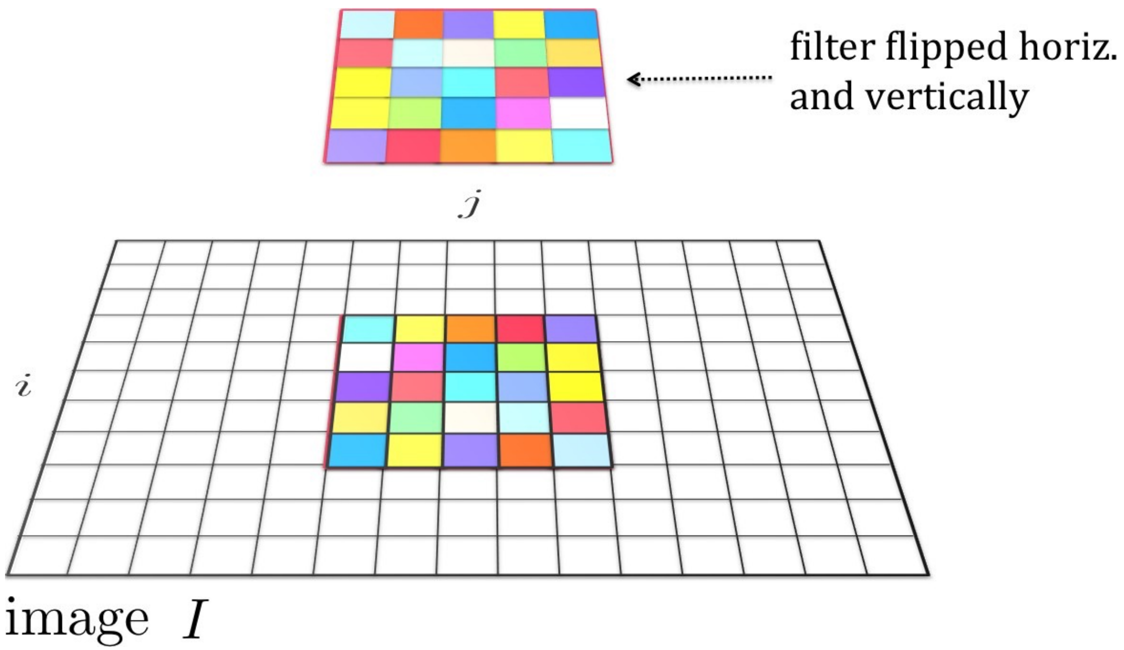


Correlation vs Convolution



Correlation

=



Convolution

Correlation vs Convolution

For a Gaussian or box filter, how will the outputs $F * I$ and $F \otimes I$ differ?

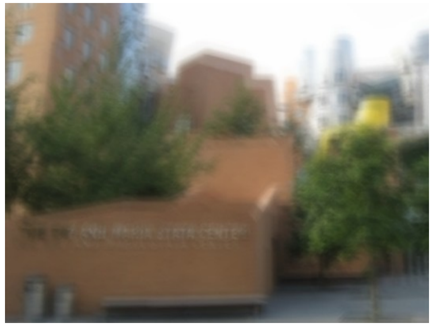
Correlation vs Convolution

For a Gaussian or box filter, how will the outputs $F * I$ and $F \otimes I$ differ?

How will the outputs differ for: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$

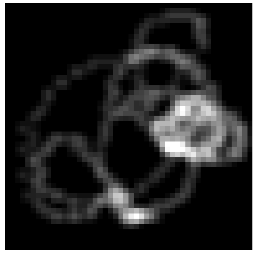
"Optical" Convolution

Camera Shake



=

*



[Fergus et al. ,SIGGRAPH 2006]

Blur in out-of-focus regions of an image

Bokeh: http://lullaby.homepage.dk/diy-camera/bokeh.html

[Source: N. Snavely]

Properties of Convolution

Commutative: $f * g = g * f$

Associative: $f * (g * h) = (f * g) * h$

Distributive: $f * (g + h) = f * g + f * h$

Assoc. with scalar multiplier: $\lambda \cdot (f * g) = (\lambda \cdot f) * g$

Properties of Convolution

Commutative: $f * g = g * f$

Associative: $f * (g * h) = (f * g) * h$

Distributive: $f * (g + h) = f * g + f * h$

Assoc. with scalar multiplier: $\lambda \cdot (f * g) = (\lambda \cdot f) * g$

The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

Properties of Convolution

Commutative: $f * g = g * f$

Associative: $f * (g * h) = (f * g) * h$

Distributive: $f * (g + h) = f * g + f * h$

Assoc. with scalar multiplier: $\lambda \cdot (f * g) = (\lambda \cdot f) * g$

The Fourier transform of two convolved images is the product of their individual Fourier transforms:

$$\mathcal{F}(f * g) = \mathcal{F}(f) \cdot \mathcal{F}(g)$$

Why is this good news?

- Hint: Think of complexity of convolution and Fourier Transform
- What if we wanted to undo the result of convolution?

Separable Filters: Speed-up Trick

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter

[Source: R. Urtasun]

Separable Filters: Speed-up Trick

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter
- Can we do faster?

[Source: R. Urtasun]

Separable Filters: Speed-up Trick

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter
- Can we do faster?
- In many cases **(not all!),** this operation can be sped up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only 2K operations**

[Source: R. Urtasun]

Separable Filters: Speed-up Trick

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter
- Can we do faster?
- In many cases **(not all!)**, this operation can be sped up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only 2K operations**
- If this is possible, then the convolutional filter is called **separable**

[Source: R. Urtasun]

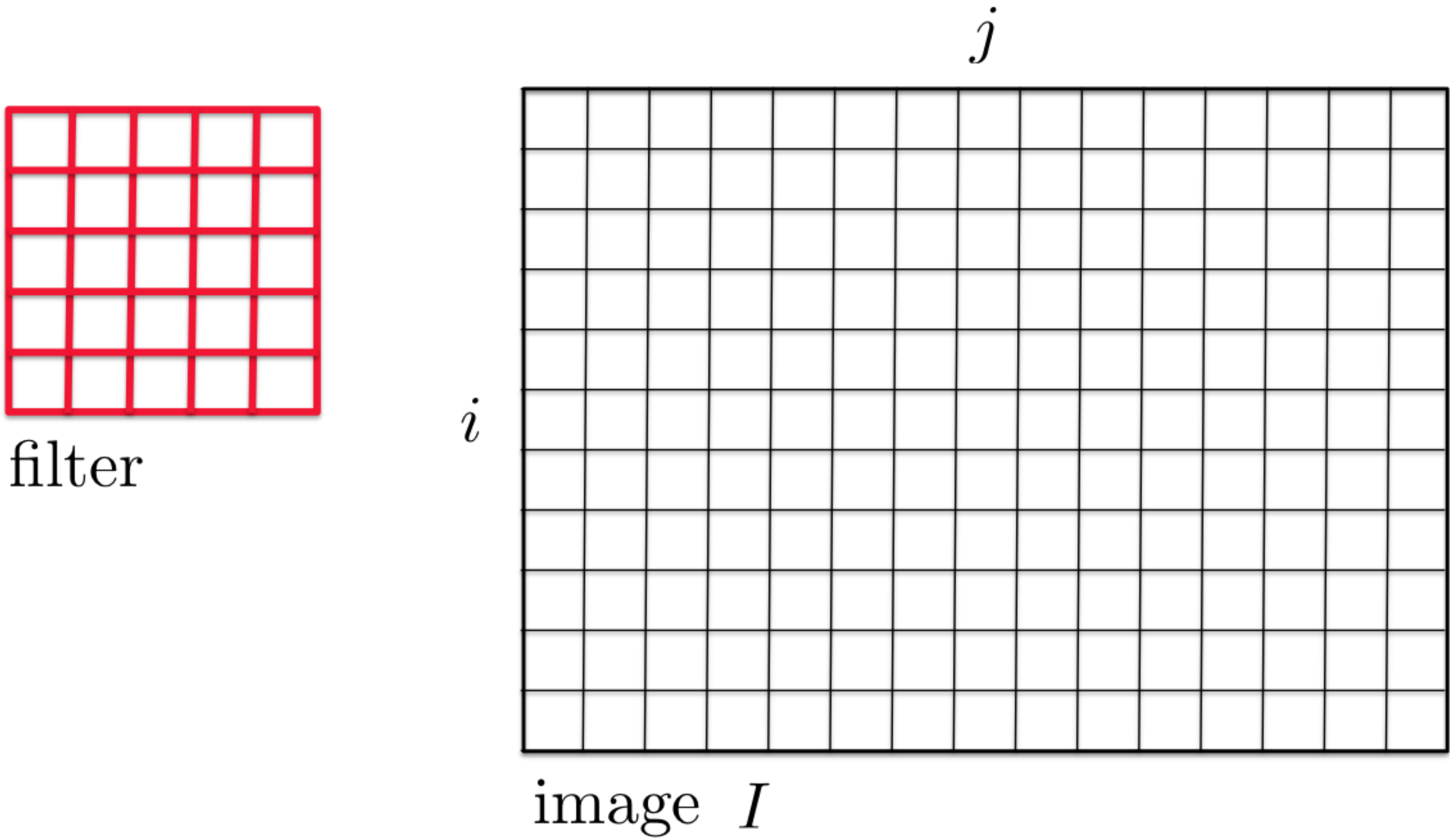
Separable Filters: Speed-up Trick

- The process of performing a convolution requires K^2 operations per pixel, where K is the size (width or height) of the convolution filter
- Can we do faster?
- In many cases **(not all!)**, this operation can be sped up by first performing a 1D horizontal convolution followed by a 1D vertical convolution, **requiring only 2K operations**
- If this is possible, then the convolutional filter is called **separable**
- And it is the outer product of two filters:

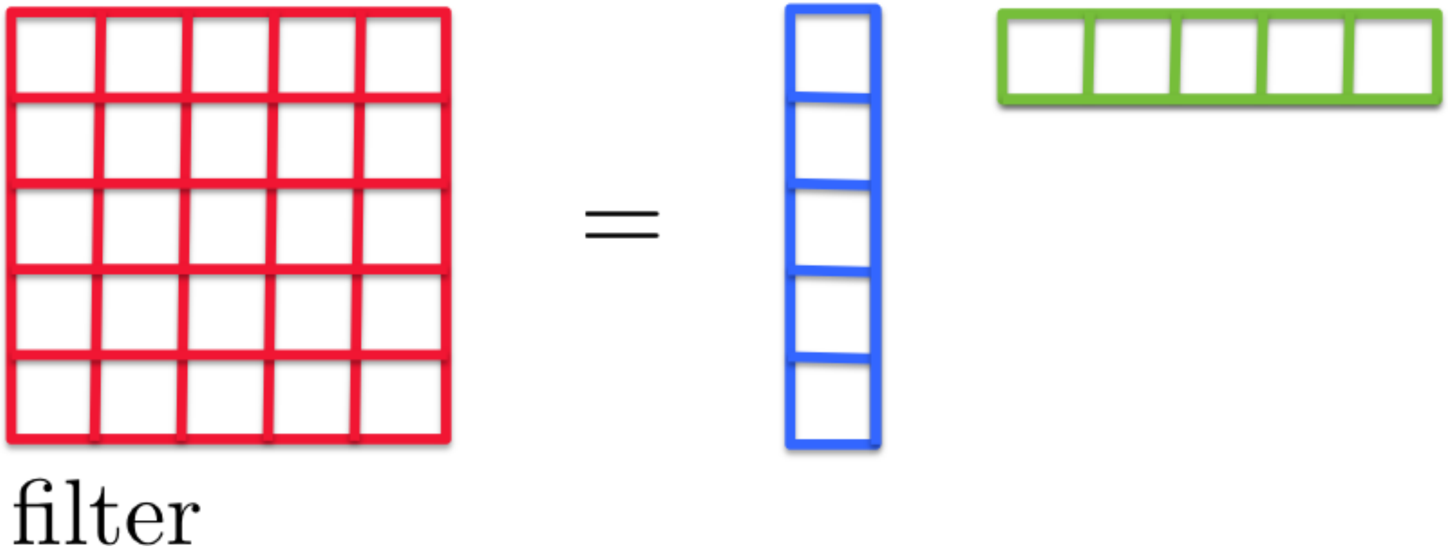
$$\mathbf{F} = \mathbf{v}\mathbf{h}^T$$

[Source: R. Urtasun]

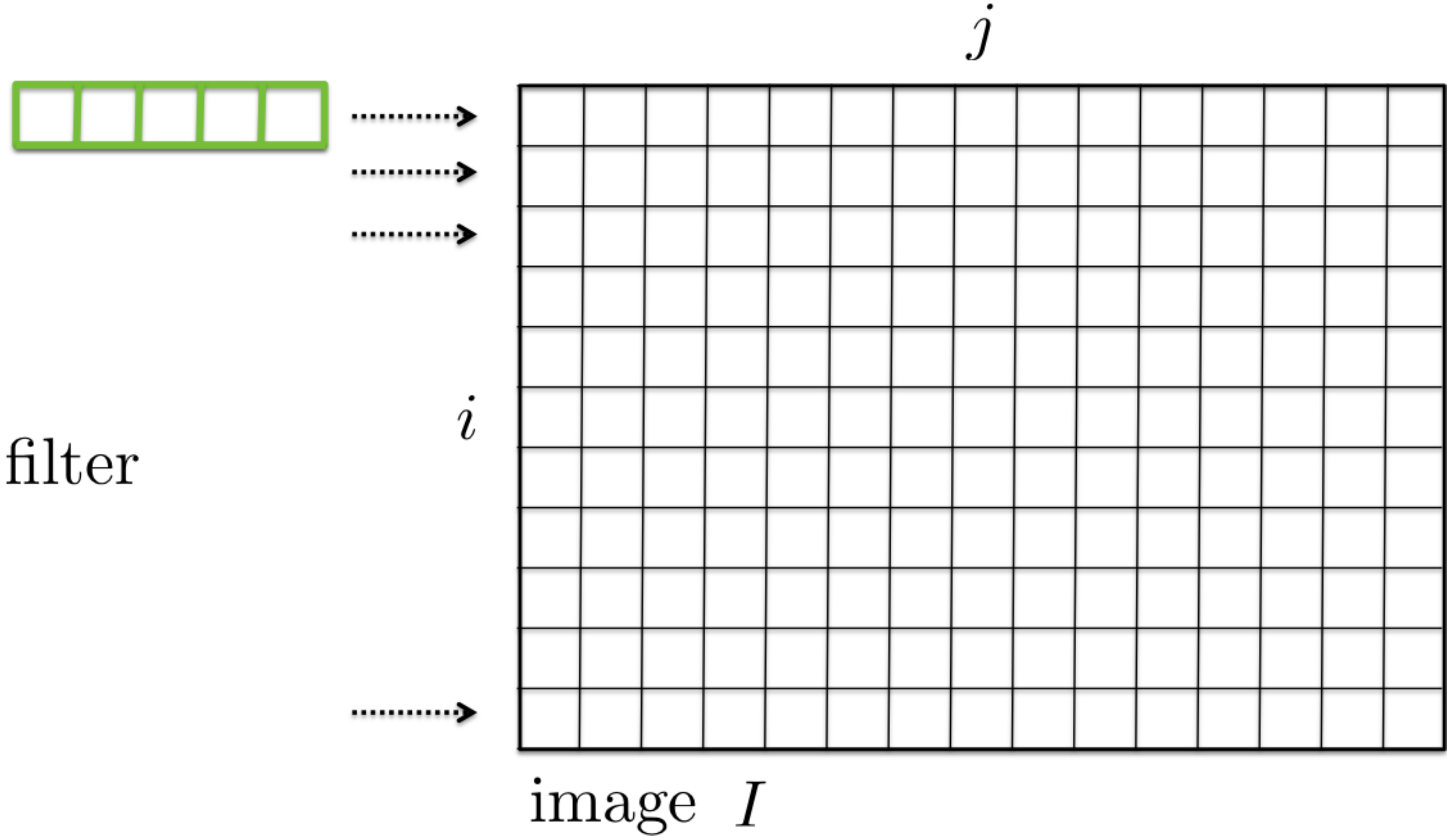
How it works



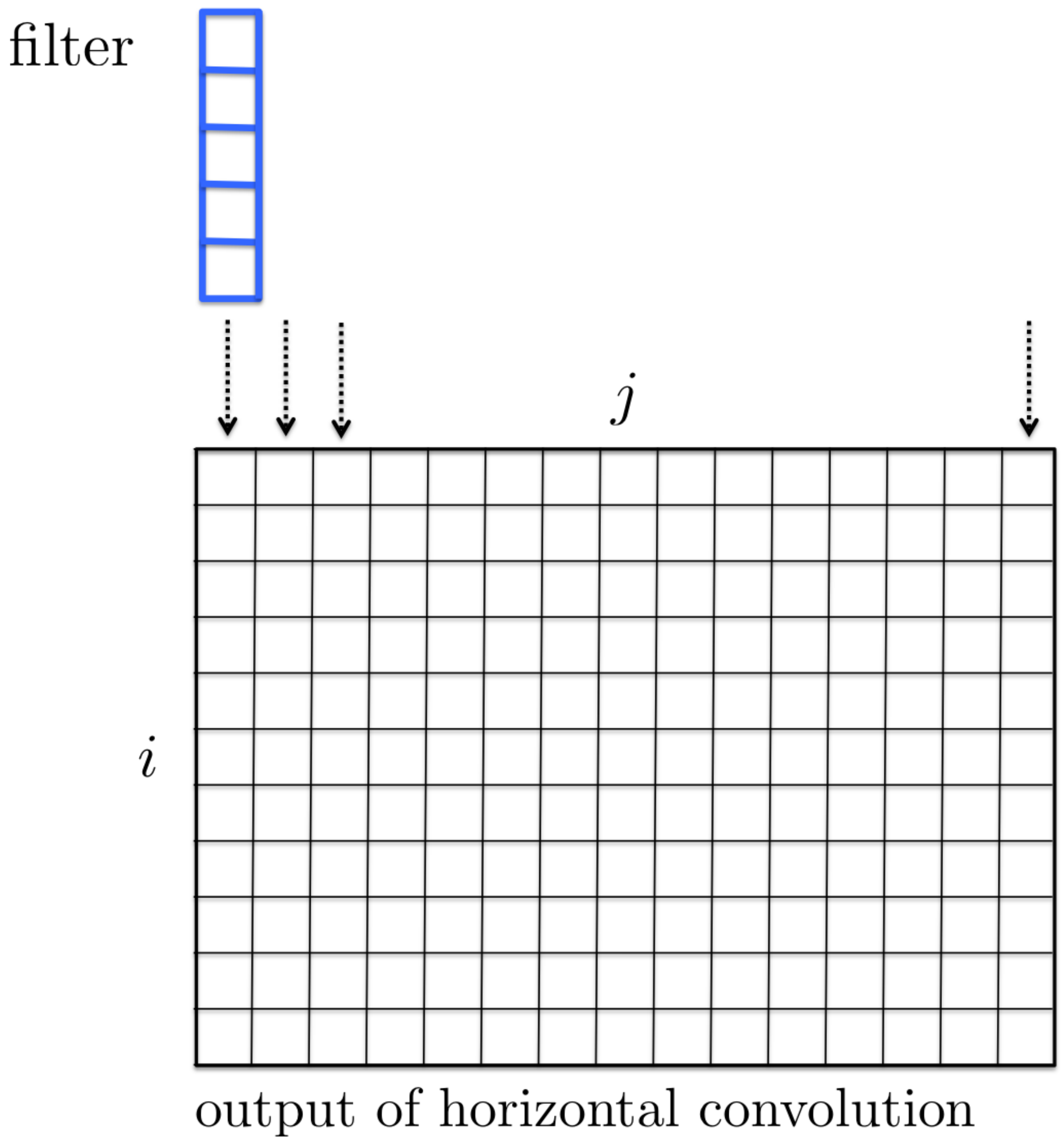
How it works



How it works



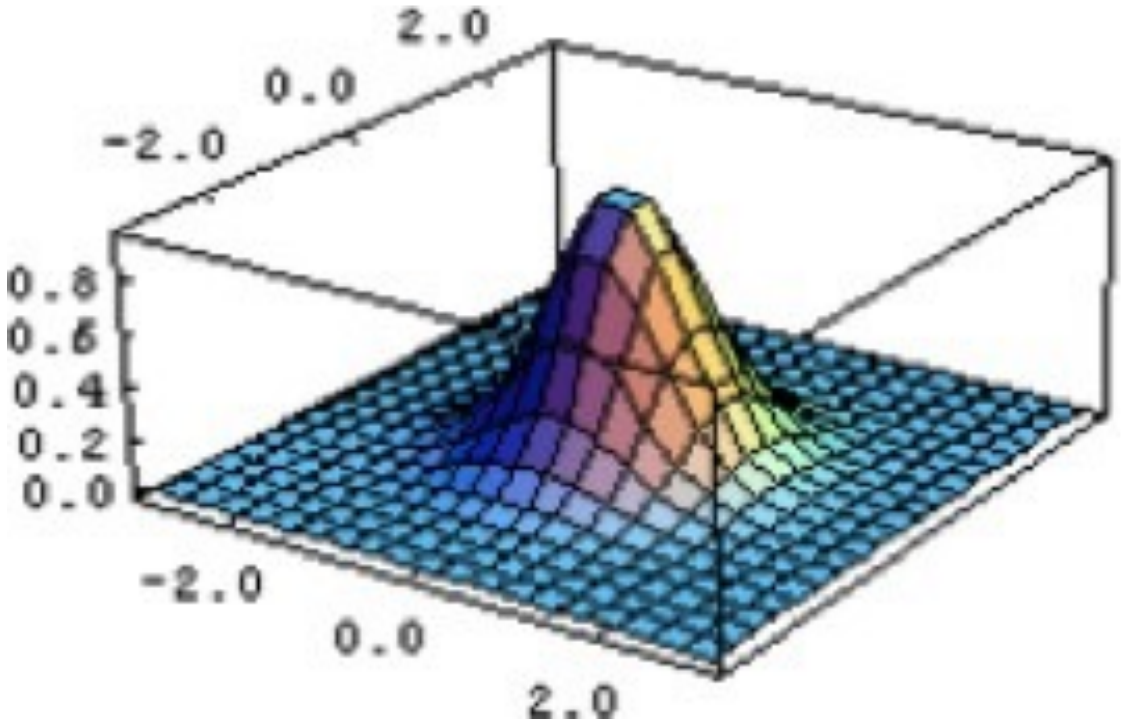
How it works



How it works

One famous separable filter we already know:

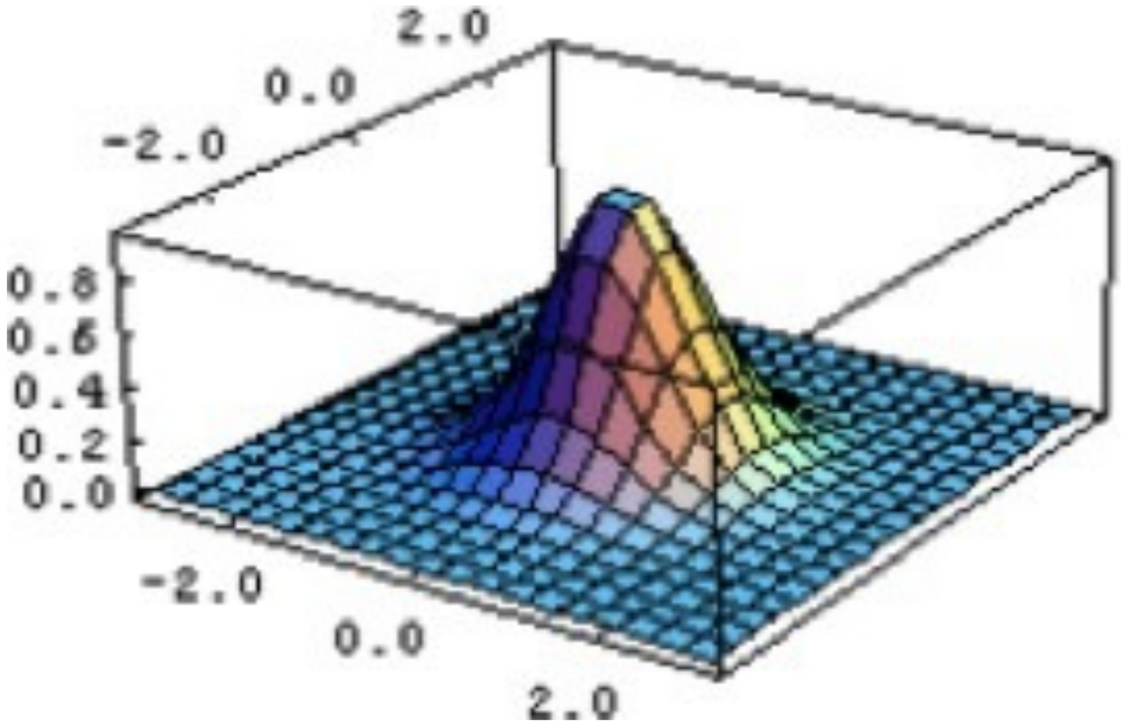
Gaussian: $f(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{\sigma^2}\right)$



How it works

One famous separable filter we already know:

Gaussian: $$f(x,y) = \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{\sigma^2}}\right) \cdot \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{\sigma^2}}\right)$$



How it works

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

[Source: R. Urtasun]

How it works

Is this separable? If yes, what's the separable version?

$$\frac{1}{K^2}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline 1 & 1 & \cdots & 1 \\ \hline \vdots & \vdots & 1 & \vdots \\ \hline 1 & 1 & \cdots & 1 \\ \hline \end{array} \qquad \frac{1}{K}\begin{array}{|c|c|c|c|}\hline 1 & 1 & \cdots & 1 \\ \hline \end{array}$$

What does this filter do?

[Source: R. Urtasun]

How it works

Is this separable? If yes, what's the separable version?

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

[Source: R. Urtasun]

How it works

Is this separable? If yes, what's the separable version?

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \qquad \frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

What does this filter do?

[Source: R. Urtasun]

How it works

Is this separable? If yes, what's the separable version?

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$$

[Source: R. Urtasun]

How it works

Is this separable? If yes, what's the separable version?

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

What does this filter do?

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing.

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing
- Look at the singular value decomposition (SVD), and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \mathrm{diag}(\sigma_i)$

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

- Inspection... this is what we were doing
- Look at the singular value decomposition (SVD), and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \mathrm{diag}(\sigma_i)$

- Python: `np.linalg.svd`

[Source: R. Urtasun]

How can we tell if a given filter F is indeed separable?

- Inspection… this is what we were doing
- Look at the singular value decomposition (SVD), and if only one singular value is non-zero, then it is separable

$$F = \mathbf{U}\Sigma\mathbf{V}^T = \sum_i \sigma_i u_i v_i^T$$

with $\Sigma = \mathrm{diag}(\sigma_i)$

- Python: `np.linalg.svd`
- $\sqrt{\sigma_1}\mathbf{u}_1$ and $\sqrt{\sigma_1}\mathbf{u}_1$ are the vertical and horizontal filters

[Source: R. Urtasun]

Summary – Stuff You Should Know

- **Correlation:** Slide a filter across image and compare (via dot product)
- **Convolution:** Flip the filter to the right and down and do correlation
- **Smooth** image with a Gaussian kernel: bigger σ means more blurring
- **Some** filters (like Gaussian) are separable: you can filter faster. First apply 1D convolution to each row, followed by another 1D conv. to each column

OpenCV:

- `Filter2D` (or `sepFilter2D`): can do both correlation and convolution
- `GaussianBlur`: create a Gaussian kernel
- `medianBlur, medianBlur, bilateralFilter`

Edges

- What does blurring take away?

[Source: S. Lazebnik]

Next time:

Edge Detection