Image Filtering

Original

Gradient Magnitude

Gaussian Blur

Median

Adaptive Thresholding

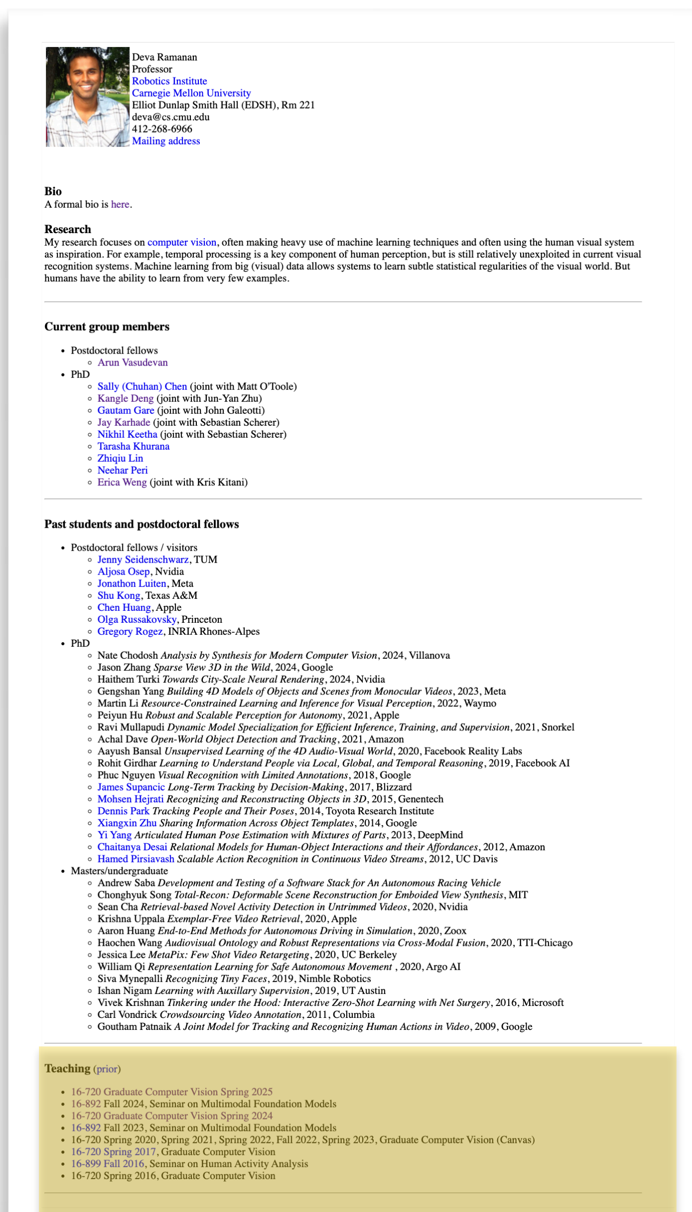
Bilateral

Logistics

Recitation today at 5pm NSH 1305, to go over HW0 and submission protocol

No CANVAS for course

Course website



Deva Ramanan
Professor
Robotics Institute
Carnegie Mellon University
Elliot Dunlap Smith Hall (EDSH), Rm 221
deva@cs.cmu.edu
412-268-6966
Mailing address

Bio
A formal bio is here.

Research
My research focuses on computer vision, often making heavy use of machine learning techniques and often using the human visual system as inspiration. For example, temporal processing is a key component of human perception, but is still relatively unexploited in current visual recognition systems. Machine learning from big (visual) data allows systems to learn subtle statistical regularities of the visual world. But humans have the ability to learn from very few examples.

Current group members

- Postdoctoral fellows
  - Arun Vasudevan
- PhD
  - Sally (Chuhan) Chen (joint with Matt O'Toole)
  - Kangle Deng (joint with Jun-Yan Zhu)
  - Gautam Gare (joint with John Galeotti)
  - Jay Karhade (joint with Sebastian Scherer)
  - Nikhil Keetha (joint with Sebastian Scherer)
  - Tarasha Khurana
  - Zhiqiu Lin
  - Neehar Peri
  - Erica Weng (joint with Kris Kitani)

Past students and postdoctoral fellows

- Postdoctoral fellows / visitors
  - Jenny Seidenschwarz, TUM
  - Aljosa Osep, Nvidia
  - Jonathon Luiten, Meta
  - Shu Kong, Texas A&M
  - Chen Huang, Apple
  - Olga Russakovsky, Princeton
  - Gregory Rogez, INRIA Rhones-Alpes
- PhD
  - Nate Chodosh *Analysis by Synthesis for Modern Computer Vision*, 2024, Villanova
  - Jason Zhang *Sparse View 3D in the Wild*, 2024, Google
  - Haithem Turki *Towards City-Scale Neural Rendering*, 2024, Nvidia
  - Gengshan Yang *Building 4D Models of Objects and Scenes from Monocular Videos*, 2023, Meta
  - Martin Li *Resource-Constrained Learning and Inference for Visual Perception*, 2022, Waymo
  - Peiyun Hu *Robust and Scalable Perception for Autonomy*, 2021, Apple
  - Ravi Mullapudi *Dynamic Model Specialization for Efficient Inference, Training, and Supervision*, 2021, Snorkel
  - Achal Dave *Open-World Object Detection and Tracking*, 2021, Amazon
  - Aayush Bansal *Unsupervised Learning of the 4D Audio-Visual World*, 2020, Facebook Reality Labs
  - Rohit Girdhar *Learning to Understand People via Local, Global, and Temporal Reasoning*, 2019, Facebook AI
  - Phuc Nguyen *Visual Recognition with Limited Annotations*, 2018, Google
  - James Supancic *Long-Term Tracking by Decision-Making*, 2017, Blizzard
  - Mohsen Hejrati *Recognizing and Reconstructing Objects in 3D*, 2015, Genentech
  - Dennis Park *Tracking People and Their Poses*, 2014, Toyota Research Institute
  - Xiangxin Zhu *Sharing Information Across Object Templates*, 2014, Google
  - Yi Yang *Articulated Human Pose Estimation with Mixtures of Parts*, 2013, DeepMind
  - Chaitanya Desai *Relational Models for Human-Object Interactions and their Affordances*, 2012, Amazon
  - Hamed Pirsiavash *Scalable Action Recognition in Continuous Video Streams*, 2012, UC Davis
- Masters/undergraduate
  - Andrew Saba *Development and Testing of a Software Stack for An Autonomous Racing Vehicle*
  - Chonghyuk Song *Total-Recon: Deformable Scene Reconstruction for Emboided View Synthesis*, MIT
  - Sean Cha *Retrieval-based Novel Activity Detection in Untrimmed Videos*, 2020, Nvidia
  - Krishna Uppala *Exemplar-Free Video Retrieval*, 2020, Apple
  - Aaron Huang *End-to-End Methods for Autonomous Driving in Simulation*, 2020, Zoox
  - Haochen Wang *Audiovisual Ontology and Robust Representations via Cross-Modal Fusion*, 2020, TTI-Chicago
  - Jessica Lee *MetaPix: Few Shot Video Retargeting*, 2020, UC Berkeley
  - William Qi *Representation Learning for Safe Autonomous Movement* , 2020, Argo AI
  - Siva Mynepalli *Recognizing Tiny Faces*, 2019, Nimble Robotics
  - Ishan Nigam *Learning with Auxillary Supervision*, 2019, UT Austin
  - Vivek Krishnan *Tinkering under the Hood: Interactive Zero-Shot Learning with Net Surgery*, 2016, Microsoft
  - Carl Vondrick *Crowdsourcing Video Annotation*, 2011, Columbia
  - Goutham Patnaik *A Joint Model for Tracking and Recognizing Human Actions in Video*, 2009, Google

Teaching (prior)

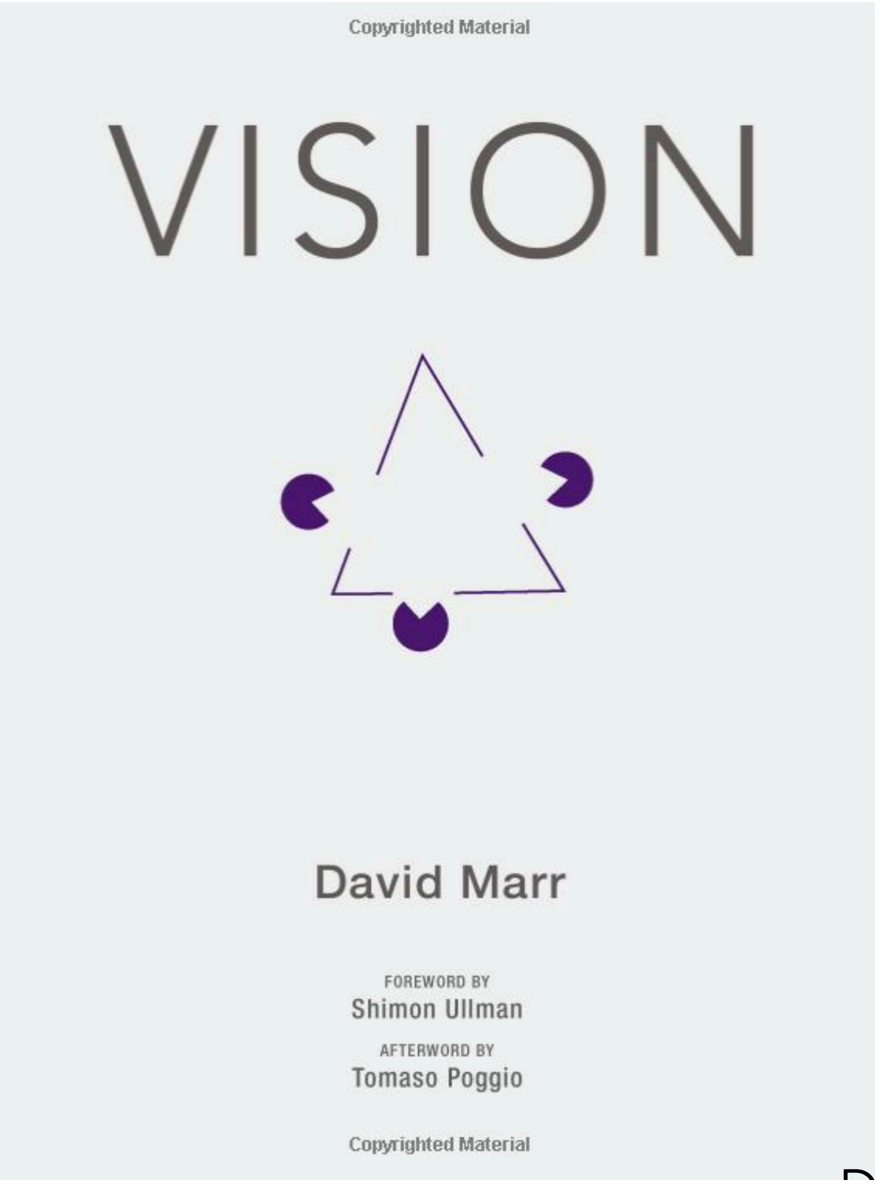
- 16-720 Graduate Computer Vision Spring 2025
- 16-892 Fall 2024, Seminar on Multimodal Foundation Models
- 16-720 Graduate Computer Vision Spring 2024
- 16-892 Fall 2023, Seminar on Multimodal Foundation Models
- 16-720 Spring 2020, Spring 2021, Spring 2022, Fall 2022, Spring 2023, Graduate Computer Vision (Canvas)
- 16-720 Spring 2017, Graduate Computer Vision
- 16-899 Fall 2016, Seminar on Human Activity Analysis
- 16-720 Spring 2016, Graduate Computer Vision

Roadmap

- **Convolution**
    - Linear Shift Invariance (LSI)
    - Convolution Properties (commutative, associative, distributive)
    - Normalized Cross-Correlation
- Edges
    - Canny edges (hysterisis)
    - derivative-of-gaussians (DoG)
    - laplacian-of-Gaussians (LoG)
- Efficiency
    - pyramids
    - linear approximiations (SVD, separability)
    - steerability

Computational perspective

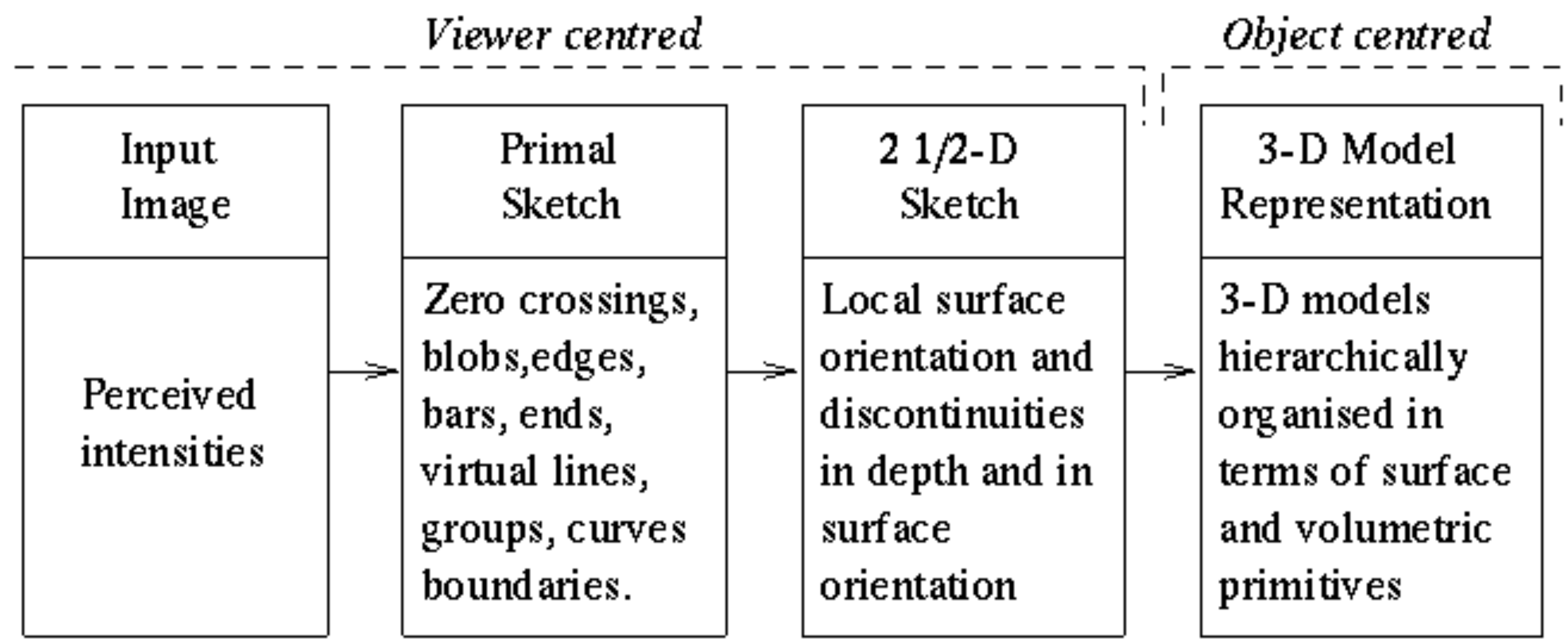
Credited with early computational approach for vision

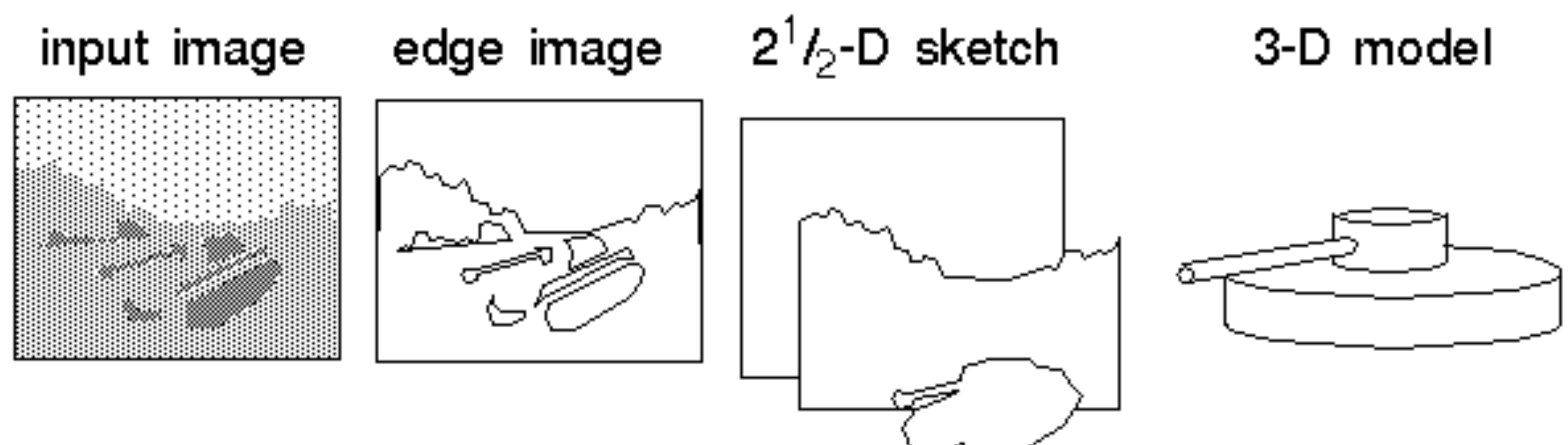


David Marr, 1982

cf, Marr Prize; Best Paper Award at International Conference on Computer Vision (ICCV)

David Marr





Low-level Mid-level High-level

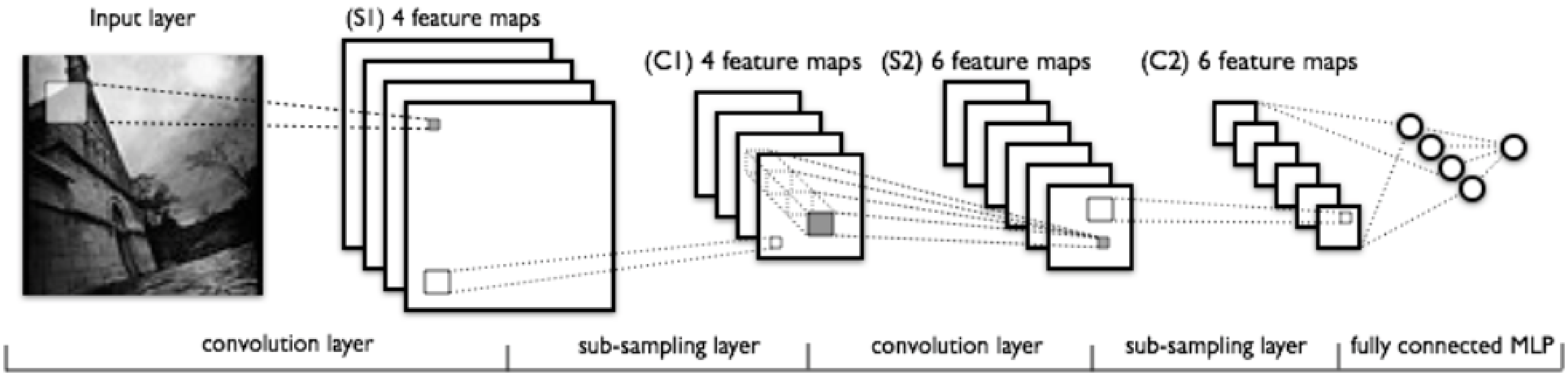
Loose structure of course

Date	Topic	References	Misc
1/16	Introduction		[First class]
1/18	Filters		Special Recitation on HW0; NSH 3305 at 5pm
1/23	Edges		HW0 - Python Intro (not graded)
1/25	Texture		
1/30	Linear Algebra Review		
2/1	SVD		HW1 - HOG Image Classification
2/6	Frequency		
2/8	Warping		
2/13	Correspondence		
2/15	Descriptors		HW2 - LK Tracking
2/20	Image Formation		
2/22	Cameras		
2/27	Motion		
2/29	Flow		HW3 - Homographies
3/5	NO CLASS [Spring Break]		
3/7	NO CLASS [Spring Break]		
3/12	Two-view		
3/14	Stereo		
3/19	SFM		
3/21	Recognition		HW4 - Reconstruction
3/26	Classifiers	Guest Lecture	
3/28	MLPs	Guest Lecture	
4/2	Training Recipes		
4/4	CNNs		
4/9	Interpreting Networks		HW5 - Neural Networks
4/11	NO CLASS [Spring Carnival]		
4/16	Radiometry		
4/18	Color		
4/23	Recent Trends		
4/25	Catchup		HW6 - Photometric Stereo

Image Processin
Geometry
Recognition
Radiometry

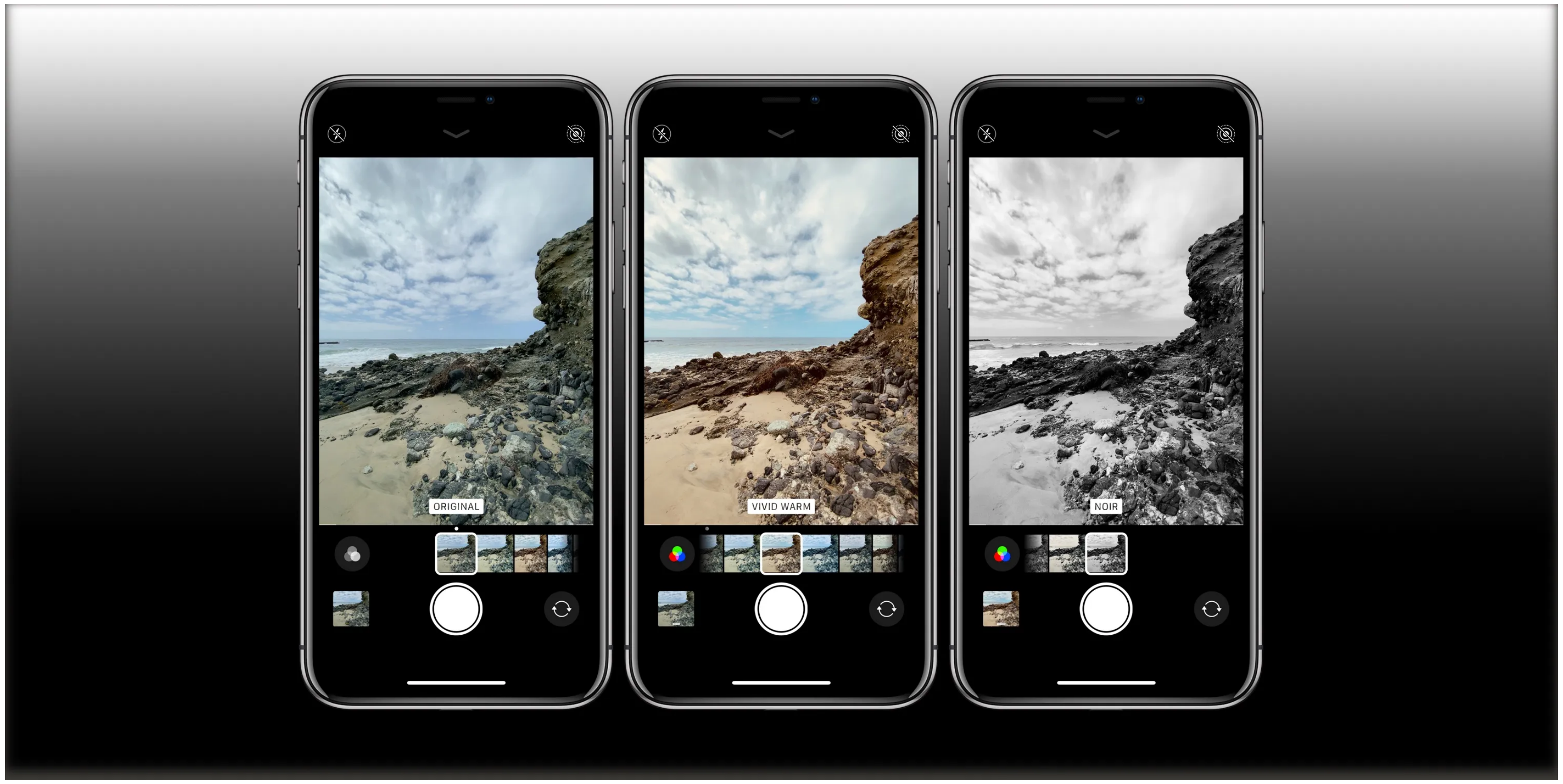
Where we are headed: filter banks

Convolutional Neural Nets (CNNs) Lecun et al 98



Consider family of low-level image processing operations

Photoshop / Instragram filters: blur, sharpen, colorize, etc….



Are certain combinations redundant? Is there a mathematical way to characterize them?

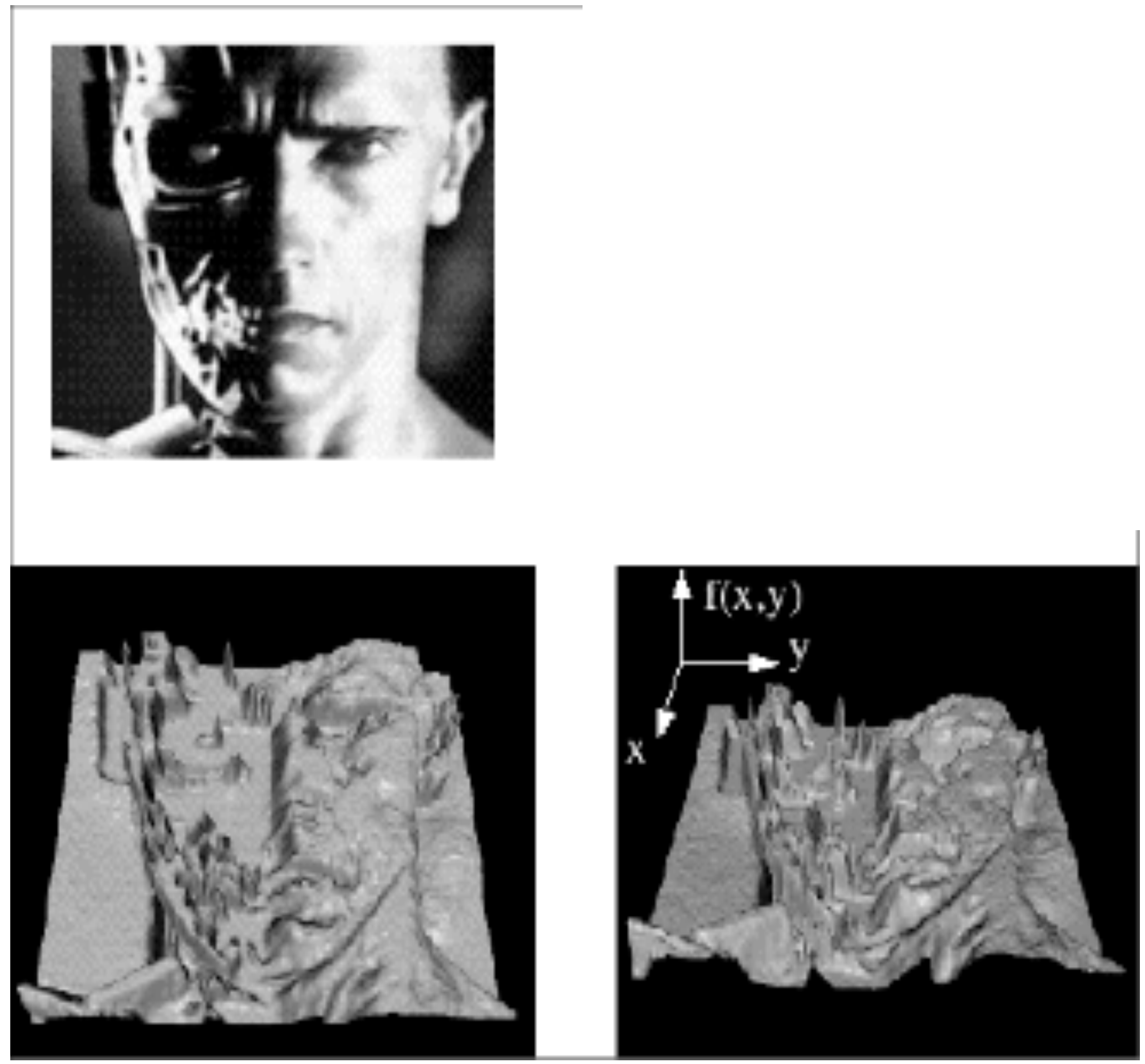
Recall: representing a digital image

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Question: how does one represent a color image? Typically values are unsigned integers from 0 to 255.
Lots of newbies trip up here in strongly typed languages (C,C++)

[Python notebook demo]

Images as height fields



Let's think of image as zero-padded functions

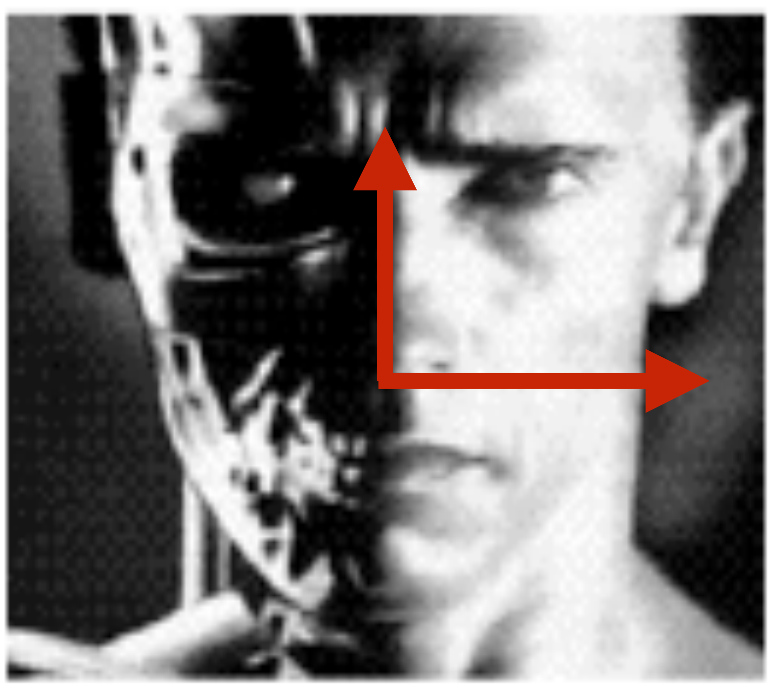
F[i,j]

Mathematically, indices i and j can vary from -inf to inf, but F is 0 outside image width+height

Aside: notation will be annoying and inconsistent across references - e.g., why not F[x,y], F(x,y),…
I'll be using my own (hopefully) self-consistent notation for my slides

Python image indexing

	0	1	2					
0	62	79	23	119	120	105	4	0
1	10	10	9	62	12	78	34	0
2	10	58	197	46	46	0	0	48
	176	135	5	188	191	68	0	49
	2	1	1	29	26	37	0	77
	0	89	144	147	187	102	62	208
	255	252	0	166	123	62	0	31
	166	63	127	17	1	0	99	30



Mathematically, more convenient to define (0,0) at center, x-axis pointing to the right, y-axis pointing up

Another point of confusion; does 'i' in im[i,j] refer to row or column? what about im[x,y]?

Unfortunately, papers, textbooks, and even python functions are inconsistent :(

Why this annoying laziness?

1) To some degree, "i" vs "x" are dummy variables. Can even write code with im[y,x] notation and it will work
2) My approach; I usually look at equations with my "math hat" convert to pseudocode with my "implementation hat"

Characterizing image transformations

62	79	23	119
10	10	9	62
10	58	197	46
176	135	5	188
2	1	1	29
0	89	144	147
255	252	0	166
166	63	127	17

$$F[i,j] \longrightarrow \boxed{T} \longrightarrow G[i,j]$$

62	79	23	119
10	10	9	62
10	58	197	46
176	135	5	188
2	1	1	29
0	89	144	147
255	252	0	166
166	63	127	17

5	4	2	3	7	4	6	5	3	6

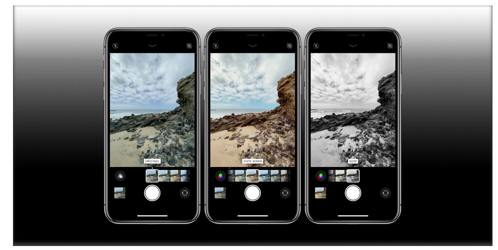
$$F[i] \longrightarrow \boxed{T} \longrightarrow G[i]$$

5	4	2	3	7	4	6	5	3	6

$$G = T(F)$$

$$G[:] = T(F[:])$$

How do we characterize image processing operations ?



Properties of "nice" functional transformations

Scaling

$$T(\alpha F) = \alpha T(F)$$

Additivity

$$T(F_1 + F_2) = T(F_1) + T(F_2)$$

Direct consequence: Linearity

$$T(\alpha_1 F_1 + \alpha_2 F_2) = \alpha_1 T(F_1) + \alpha_2 T(F_2)$$

Shift Invariance

$G[i - \alpha] = T(F[i - \alpha])$

clunky notation; operator applies to entire signal (image), not just a single element (pixel).

we'll formalize more precisely in next few slides (with *convolution*)

What would these look like for 2D images? $G[i - \alpha, j - \beta] = T(F[i - \alpha, j - \beta])$

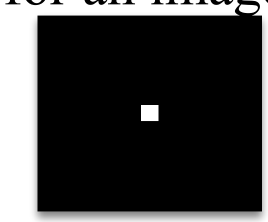
Impulse signals

[also called delta function]

$$\delta[i] = \begin{cases} 1 & i = 0 \\ 0 & \text{otherwise} \end{cases}$$

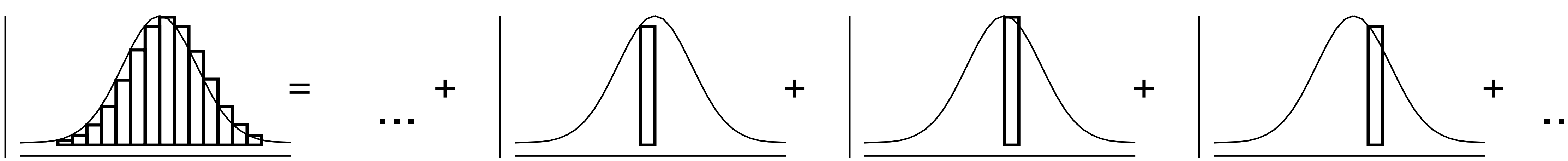
What does this look like for an image?

$$\delta[i,j] = \begin{cases} 1 & i,j = 0 \\ 0 & \text{otherwise} \end{cases}$$



(assuming [0,0] is at image center, which is *not* the default for numpy]

Any function can be written as linear combination of shifted and scaled impulse reponses



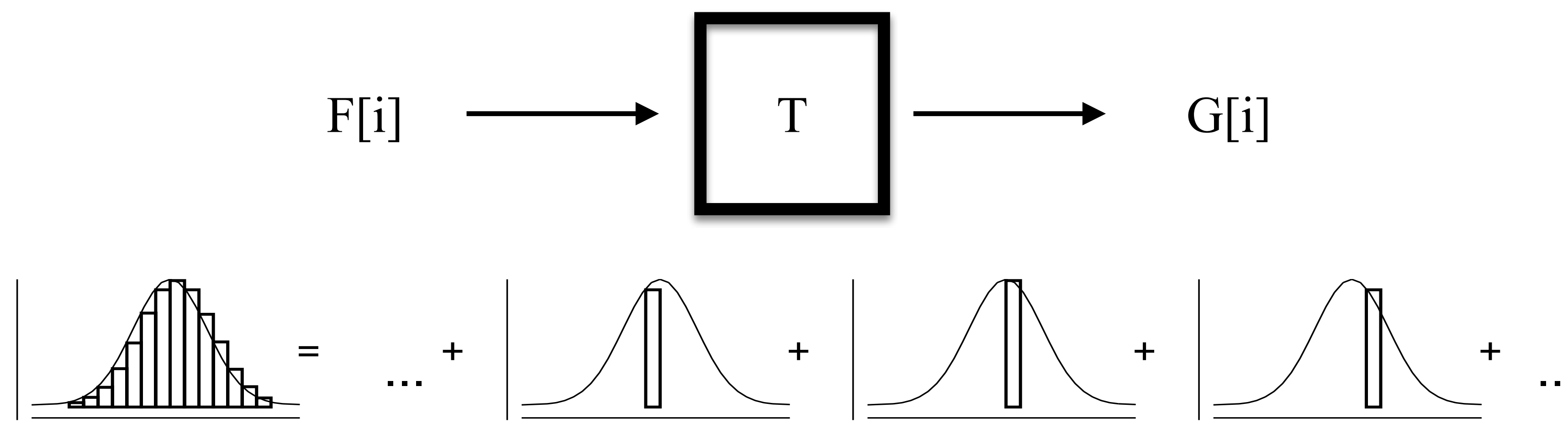
F[i] = ?

$$F[i] = F[0]\delta[i] + F[1]\delta[i-1] + \ldots$$

$$F[i] = \sum_u F[u]\delta[i-u]$$

Impulse response

Remarkable property: *any* shift-invariant linear operator is completely specified by its response to an impulse signal



$$F[i] = \sum_u F[u]\delta[i-u]$$

$$T(F[i]) = \sum_u F[u]T(\delta[i-u])$$

impulse response, filter, kernel

$$G[i] = \sum_u F[u]H[i-u] \quad \text{where} \quad H[i] = T(\delta[i]), G[i] = T(F[i])$$

$$G = F * H$$

Definition: any shift-invariant linear operator can be written as a *convolution,* written as “*”, of input F with (impulse reponse, filter, kernal) H… where **H is *literally* the response of the operator T to an impluse input**

Example

$$G[i] = F[i] * H[i] = \sum_{u} F[u]H[i-u]$$

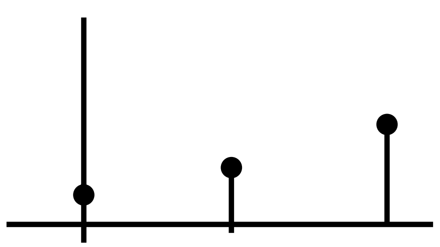
H

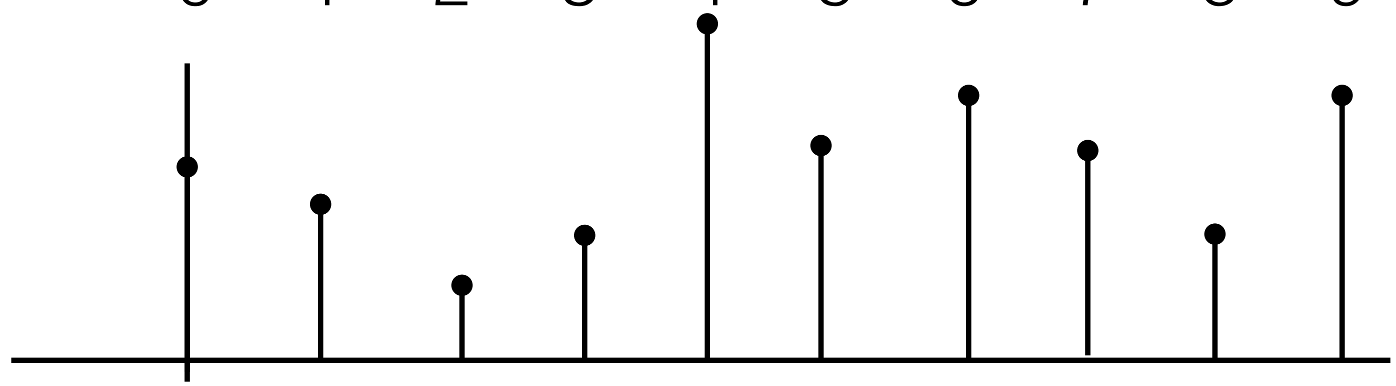
1	2	3
0	1	2

*

F

5	4	2	3	7	4	6	5	3	6
0	1	2	3	4	5	6	7	8	9





G[0] = ?
G[1] = ?

Example

$$G[i] = F[i] * H[i] = \sum_u F[u]H[i-u]$$

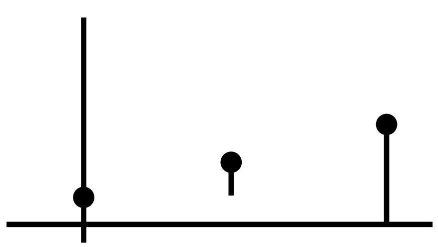
H

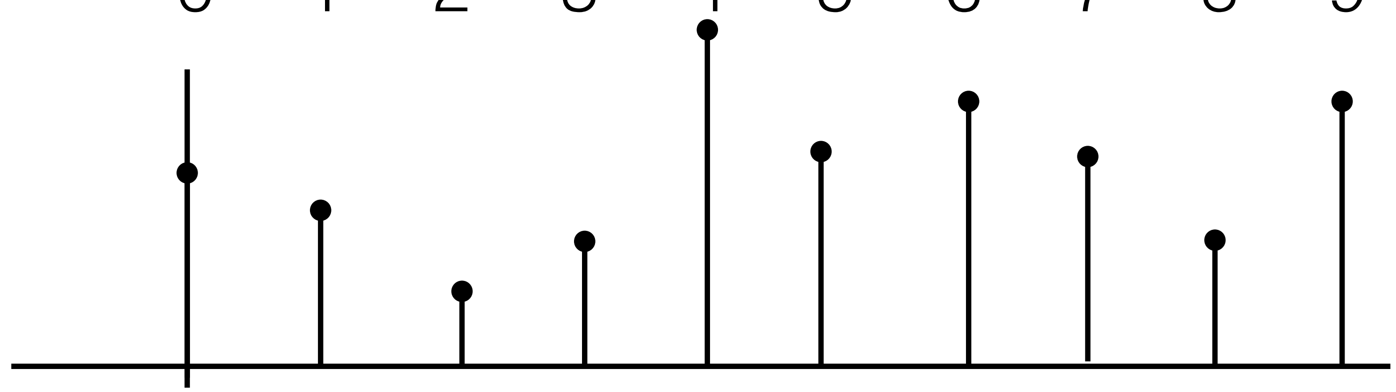
1	2	3
0	1	2

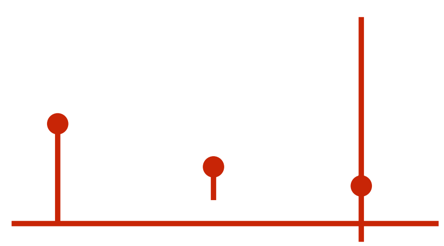
*

F

5	4	2	3	7	4	6	5	3	6
0	1	2	3	4	5	6	7	8	9







G[0] = 5x1 = 5

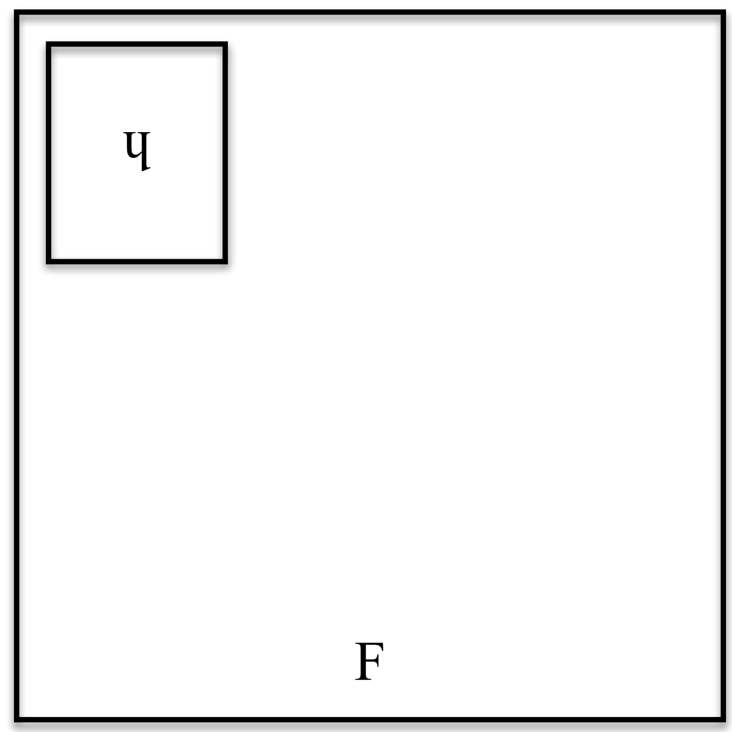
G[1] = 5x2+ 4x1 = 14

G[2] = 5x3 + 4x2 + 2x1 = 25

…

But note that np.conv(F,H) be implemented faster with parallel computations (rather than sequential, as shown above)

Preview of 2D



Properties of convolution

$$F * H = H * F$$ Commutative

$$(F * H) * G = F * (H * G)$$ Associative

$$(F * G) + (H * G) = (F + H) * G$$ Distributive

Implies that we can efficiently implement complex operations

Powerful way to think about any image transformation that satisfies additivity, scaling, and shift-invariance

Proof: commutativity

$$H * F =$$

$$=$$

Conceptually wacky: allows us to interchange the filter and image

Proof: commutativity

$$H * F = \sum_u H[u]F[i-u] = \sum_{u'} H[i-u']F[u'] \quad \text{where} \quad u' = i - u$$

$$= \sum_u F[u]H[i-u] = F * H$$

Conceptually wacky: allows us to interchange the filter and image

Size

Given F of length N and H of length M, what's size of $G = F * H$?

(e.g., try $N = 5$ and $M = 3$ with your hands :))

Size

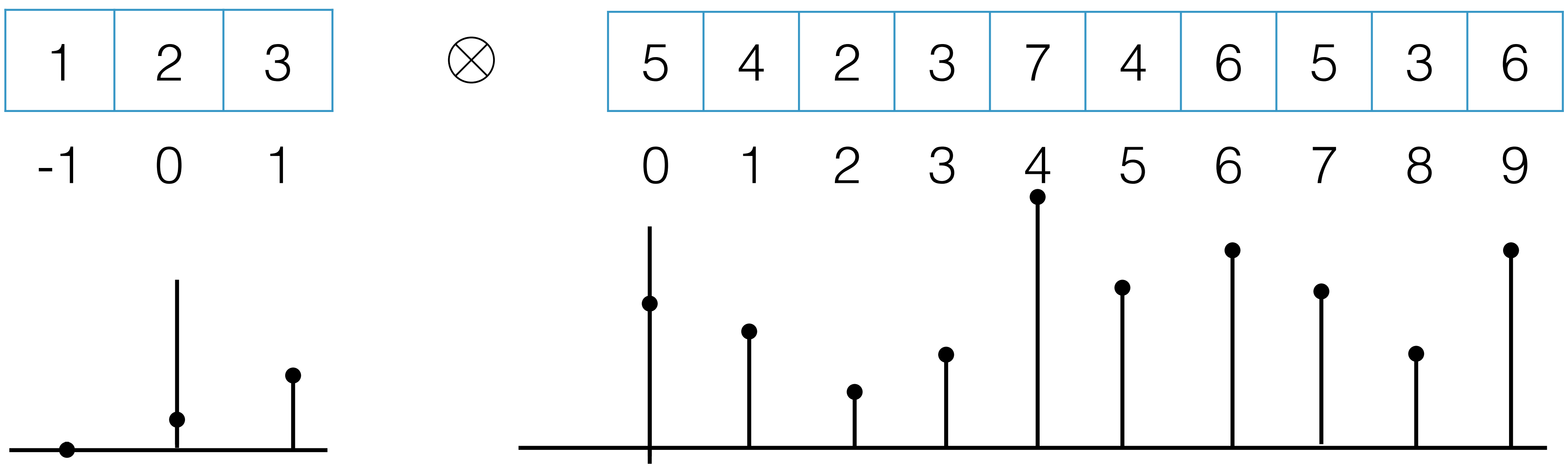
Given F of length N and H of length M, what’s size of G = F * H?

(e.g., try N = 5 and M = 3 with your hands :))

`>>>np.convolve(F,H,’full’)`	N+M-1
`>>>np.convolve(F,H,’valid’)`	N-M+1
`>>>np.convolve(F,H,’same’)`	N

A simpler approach

$$F \otimes H$$



Scan original F instead of flipped version.
What's the math (turns out to be convenient to assume filter is centered at 0)?

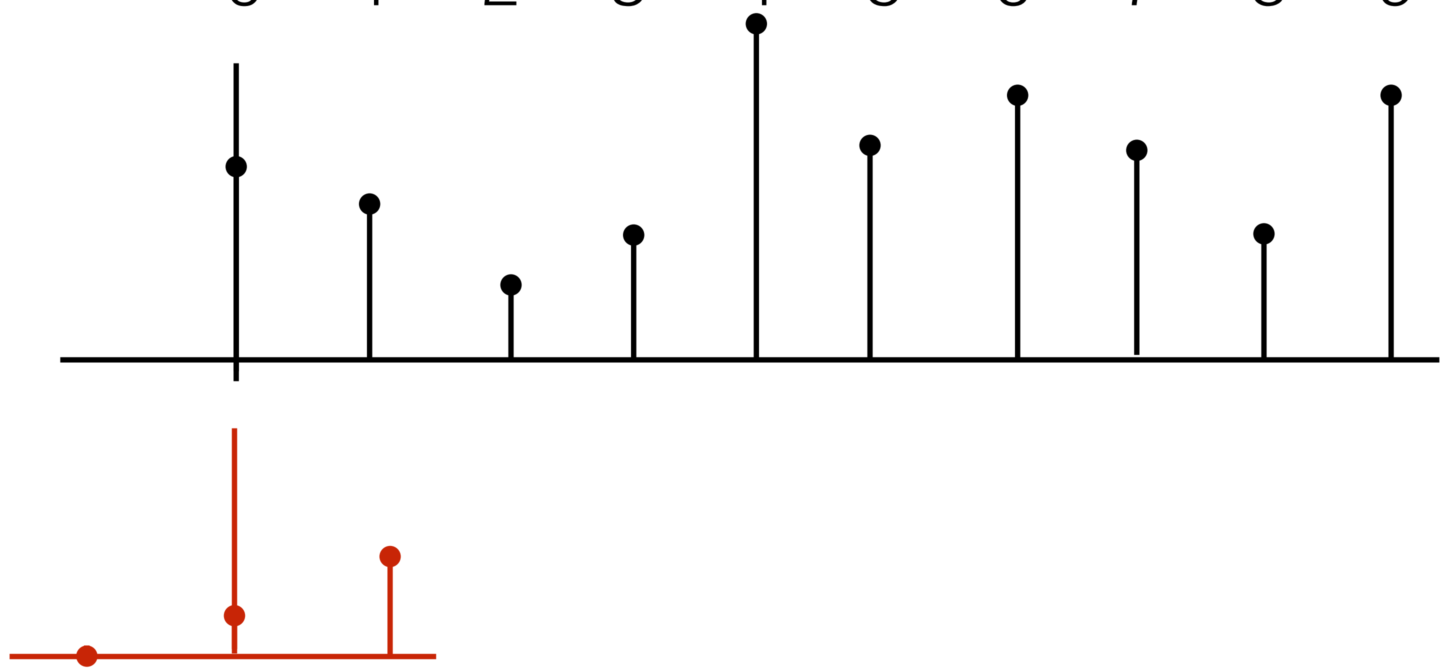
A simpler approach

$$F[i] \otimes H[i] = \sum_{u=-k}^{u=k} H[u] F[i+u]$$

1	2	3
-1	0	1

$\otimes$

5	4	2	3	7	4	6	5	3	6
0	1	2	3	4	5	6	7	8	9



Scan original F instead of flipped version.
What's the math (turns out to be convenient to assume filter is centered at 0)?

Properties

Associativity, Commutative properties do *not* hold

… but correlation is easier to think about

Convolution vs correlation (1-d)

$$G[i] = F[i] * H[i] = \sum_u F[u]H[i-u]$$ (convolution)

$$= H[i] * F[i] = \sum_u H[u]F[i-u]$$ (commutative property)

$$G[i] = F[i] \otimes H[i] = \sum_u H[u]F[i+u]$$ (cross-correlation)

$$= F[i] * H[-i]$$ (exercise for reader!)

Quiz for understanding: is cross-correlation a linear shift-invariant operation? yes

If so, what is it's associated impulse response? H[-i]

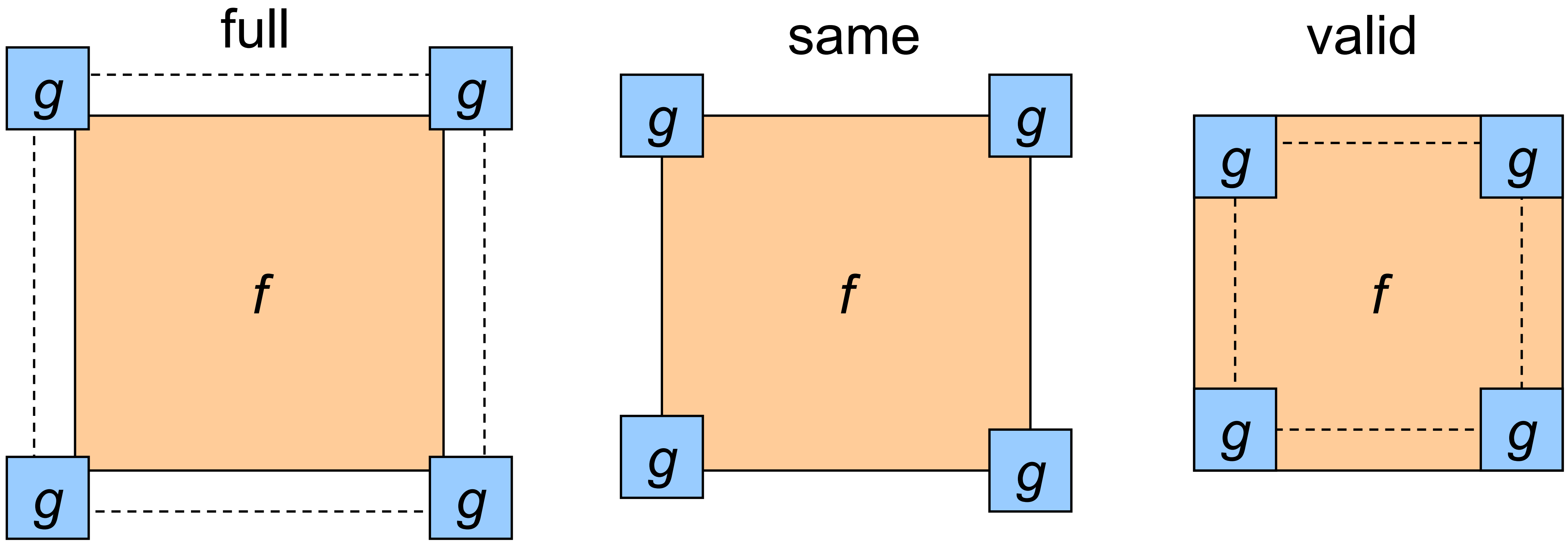
2D correlation

$$G[i,j] = F \otimes H = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

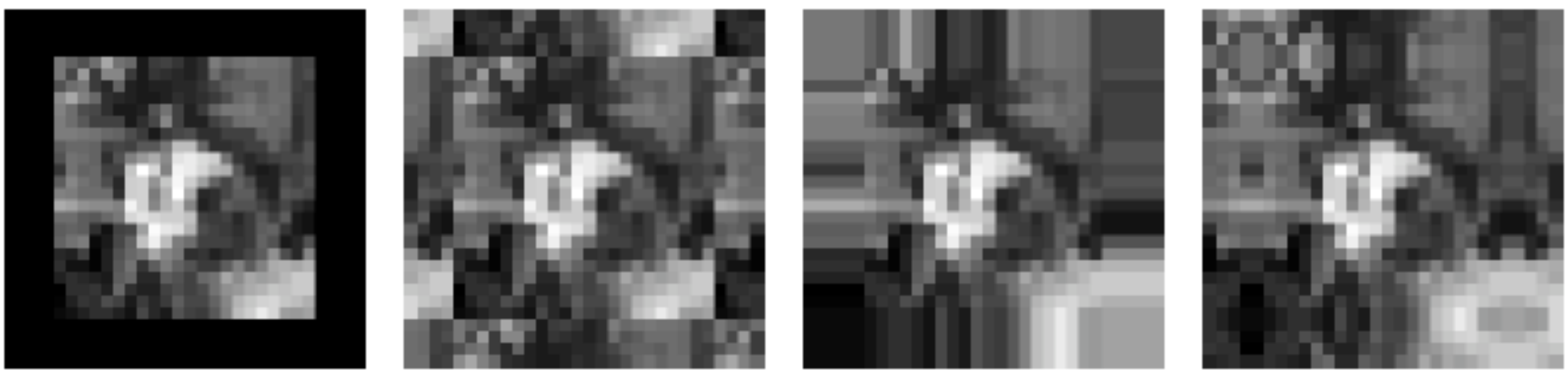
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

1	2	1
2	4	2
1	2	1

Border effects



Border padding



zero wrap clamp mirror

Examples of correlation

Original

1	2	1
2	4	2
1	2	1

/16

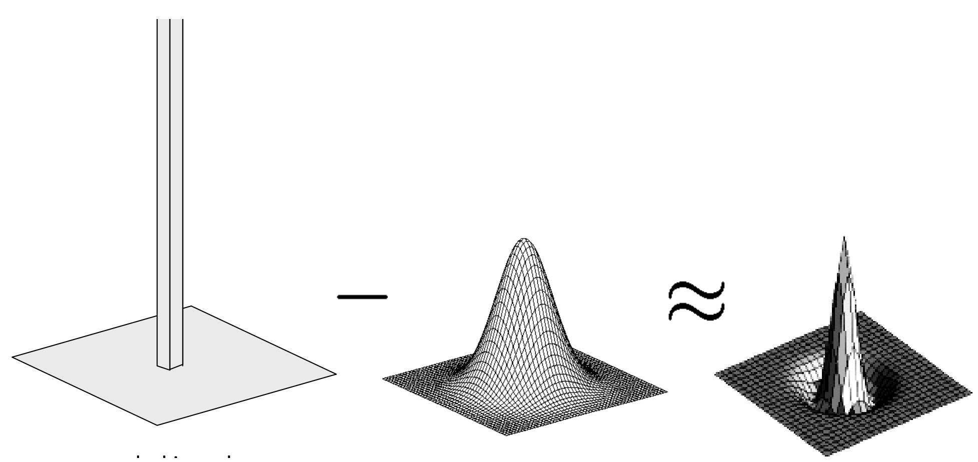
What would this look like for convolution?

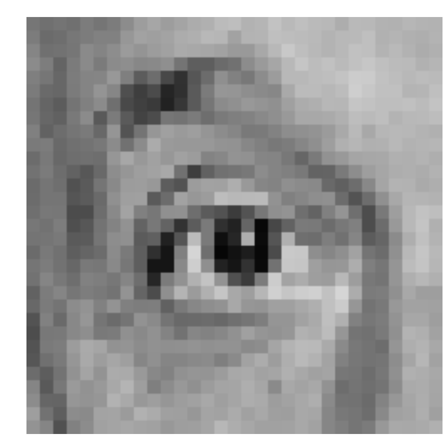
Let's say I wanted to accentuate the image "detail", or the difference between the original and blurred version. Can we do this with a single filter?

Examples of correlation

Original

$$\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} /16 \right) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$





Technically, in order to apply the distributive property, we should work with convolution filters rather than correlation filters

Unsharp filter (really should be *sharp* filter)

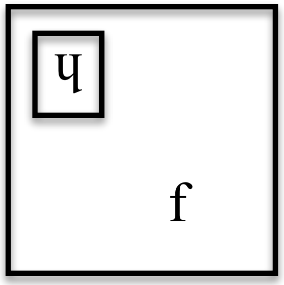
Lecture 1 Recap

Any linear shift-invariant (LSI) image operation must be representable as a convolution of an image F[i,j] with an *impulse response/ filter* H[i,j]

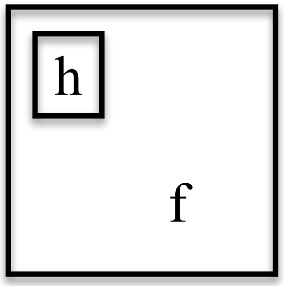
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

1	2	1
2	4	2
1	2	1

Convolution: $$G[i,j] = F * H = \sum_{u,v} F[u,v]H[i-u, i-v]$$



(Cross)-Correlation: $$G[i,j] = F \otimes H = \sum_u \sum_v H[u,v]F[i+u, j+v]$$



Convolution from-scratch

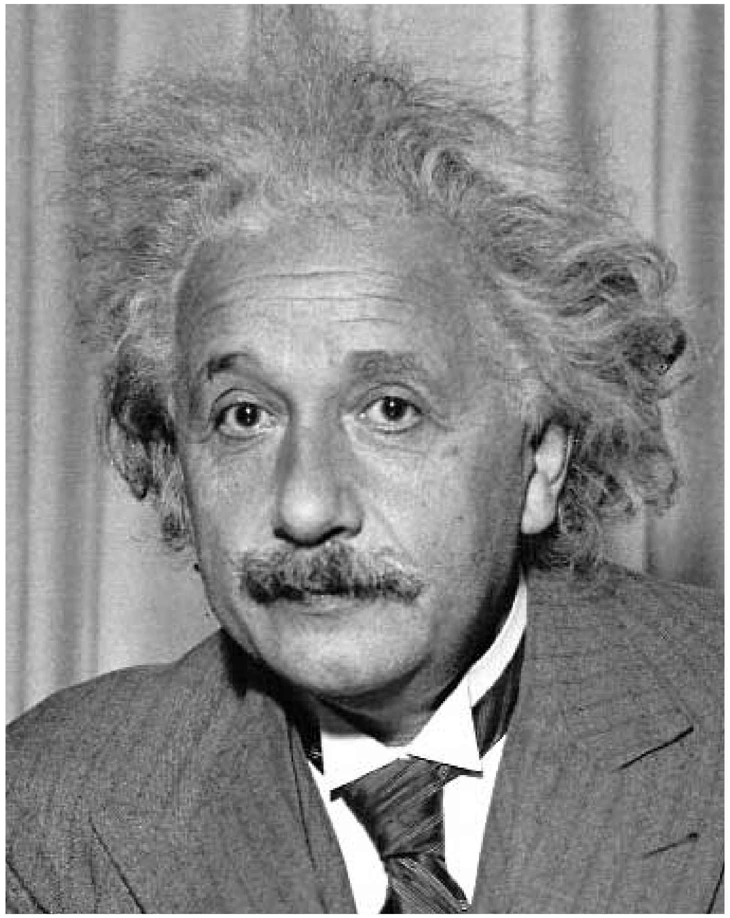
```
def convolve2d(F, H):
    # Perform 2D convolution with 'valid'-size output
    # using vector dot products
    ny, nx = H.shape
    my, mx = H.shape

    # Define output
    oy  = ny - my + 1
    ox  = nx - mx + 1
    res = np.zeros((oy,ox))

    #Flip filter left-right and up-down
    H = H[::-1,::-1]

    #Turn filter into a vector
    H = H.flatten()

    for i in range(oy):
      for j in range(ox):
        res[i,j] = np.dot(H,F[i:i+my, j:j+mx].flatten())
    return res
```

[see demo_convolution.pynb]

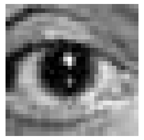
Crucial intuition for convolution (and cross-correlation):
res[i,j] is the *dot-product* between two *vectors*; the flattened filter H and the flattened local neighborhood F[i,j]

Template matching with filters

F[i,j]



H[i,j]



Can we use filters to build detectors?

Attempt 0: correlate with eye patch

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

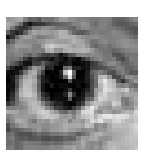
We see large responses on the shirt but not the eye. Why? Can we fix it?

Think of H[:] and F_{ij}[:] as vectors (e.g., 10x10 filter => vector of 100). G[i,j] is then the *dot product* between these 2 vectors

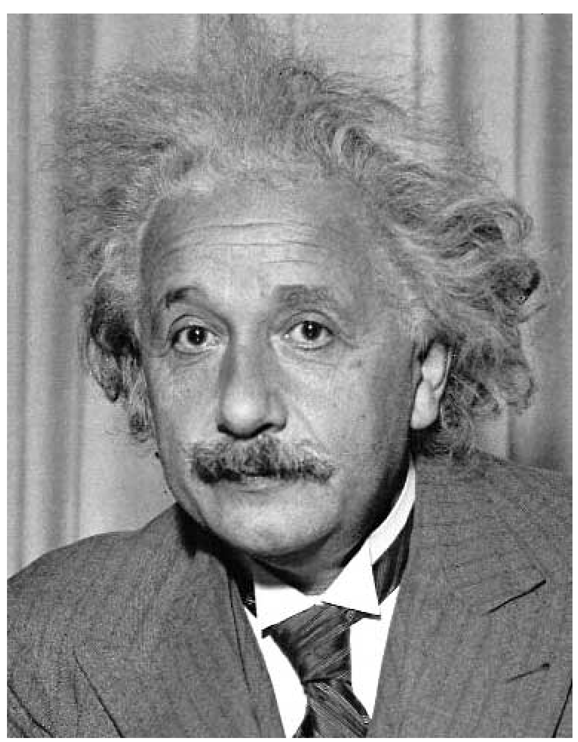
$$H = H_c + \mathbf{1}\mu_H$$

$$[H_c + \mathbf{1}\mu_H]^T F_{ij} = H_c^T F_{ij} + \mu_H \mathbf{1}^T F_{ij}$$

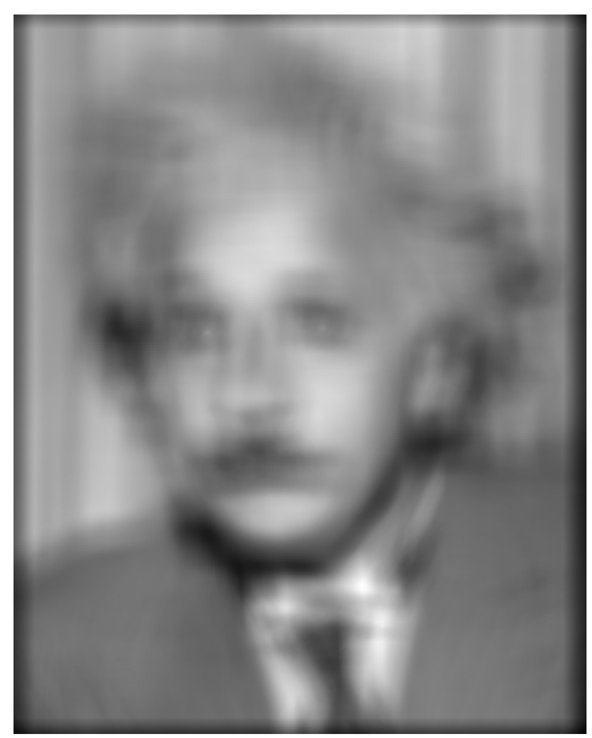
$$= H_c^T F_{ij} + \text{constant} \cdot \mu_{F_{ij}}$$



H[i,j]



F[i,j]



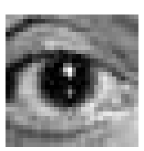
G[i,j]

Implies G[i,j] will
be large whereever F[i,j] is large

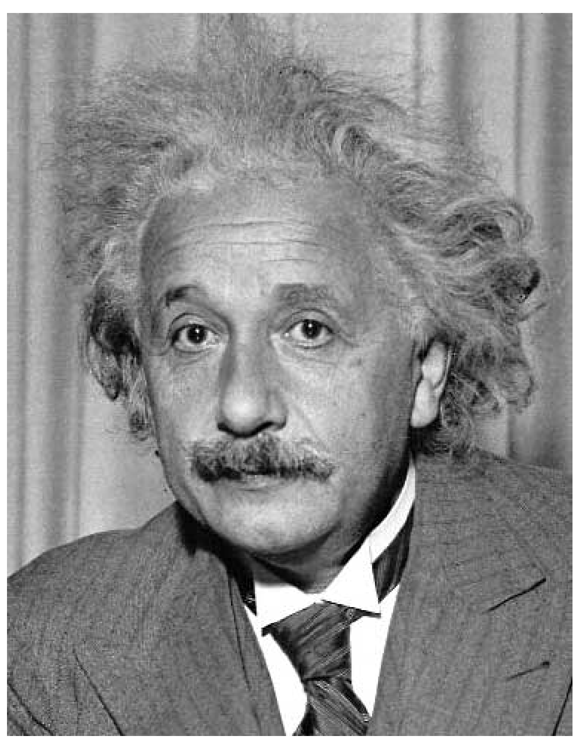
Attempt 1: correlate with zero-mean eye patch

(where $H = H_c$)

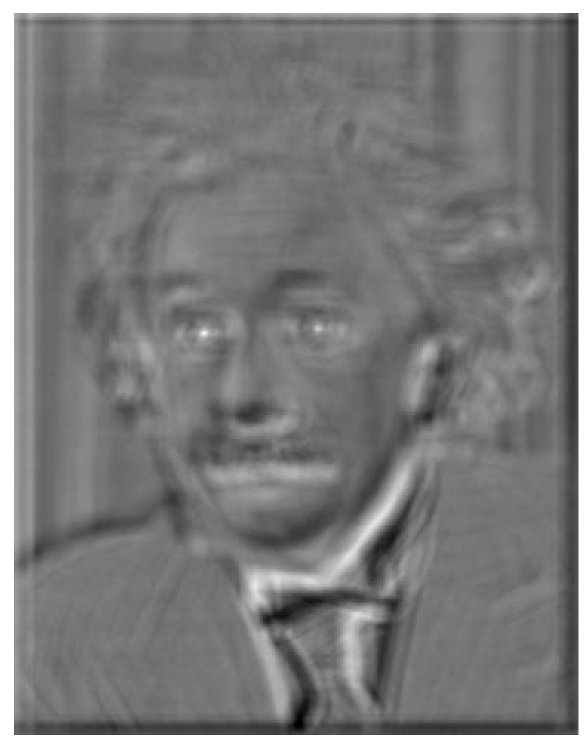
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$



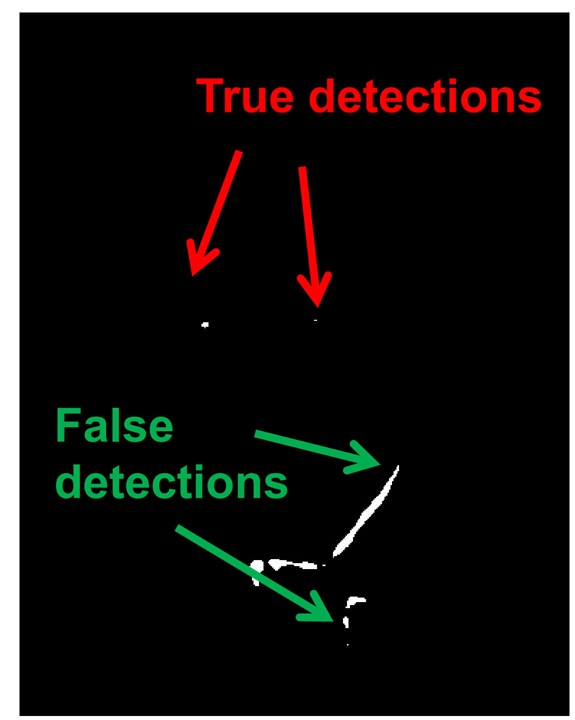
H[i,j]



F[i,j]



G[i,j]



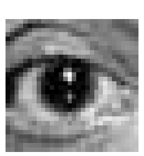
G[i,j] > thresh

Attempt 1: correlate with eye patch

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

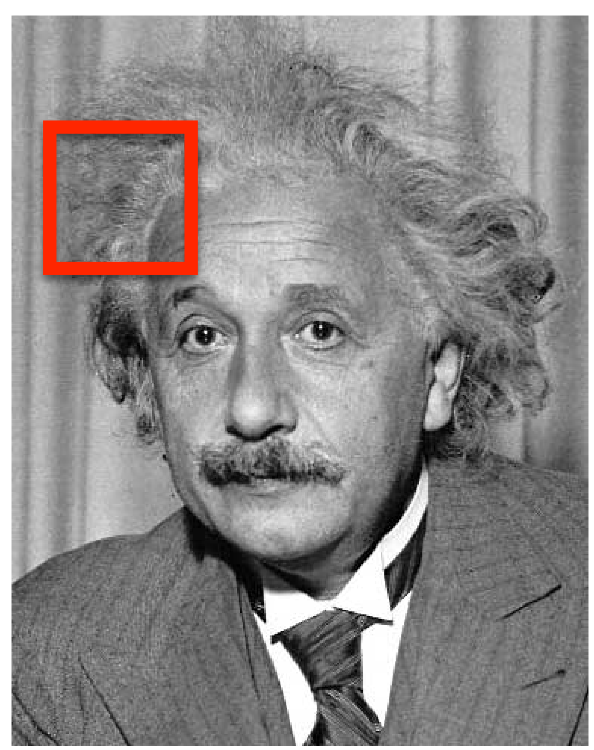
$$= H^T F_{ij} = ||H|| \, ||F_{ij}|| \cos\theta, \quad H, F_{ij} \in R^{(2K+1)^2}$$

Useful to think about correlation and convolution as inner products of vectors H[:] and F_{ij}[:]

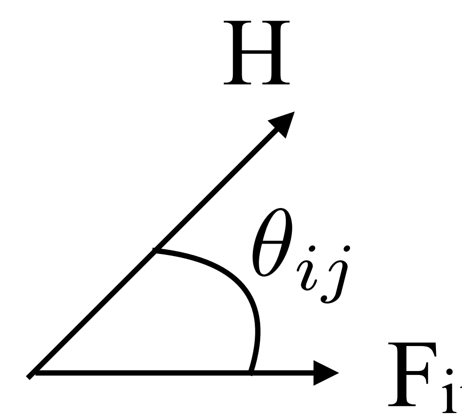


H

F_{ij}

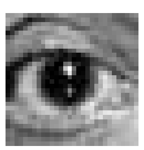


Input

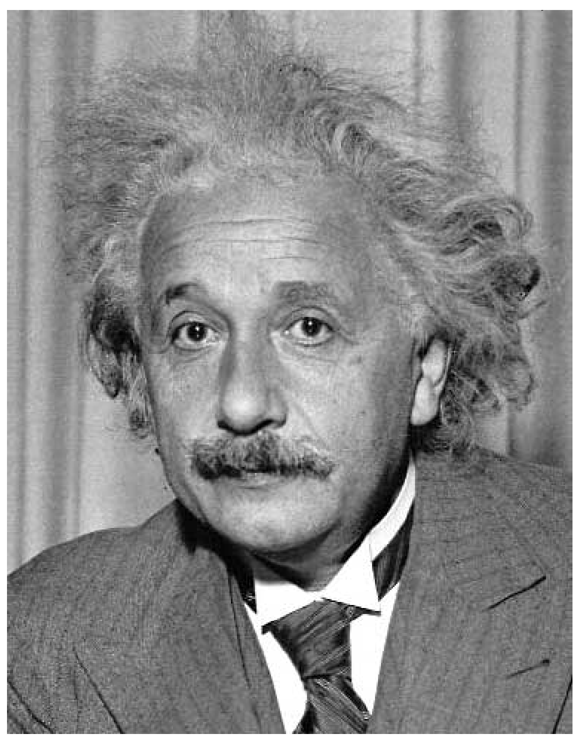


Attempt 1: correlate with eye patch (revisited)

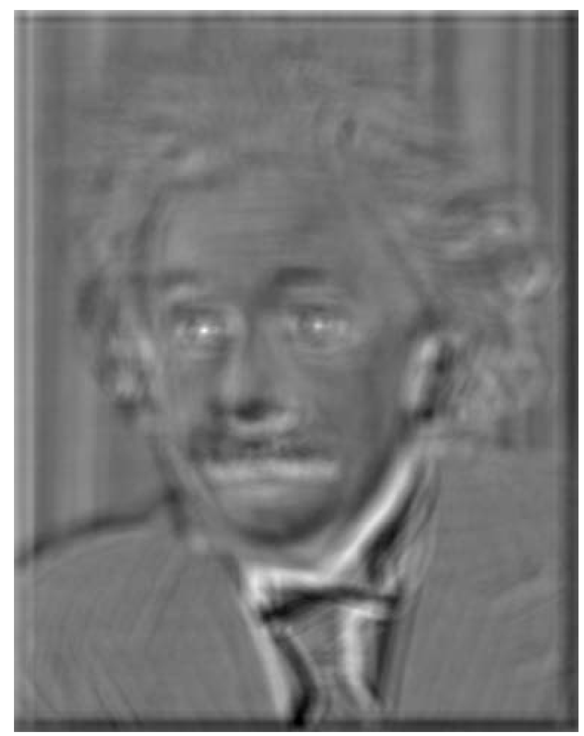
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$



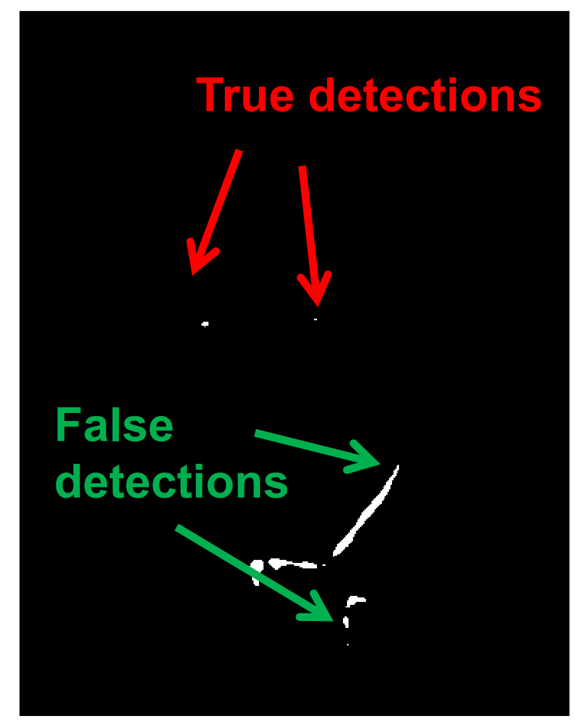
H[i,j]



F[i,j]



G[i,j]

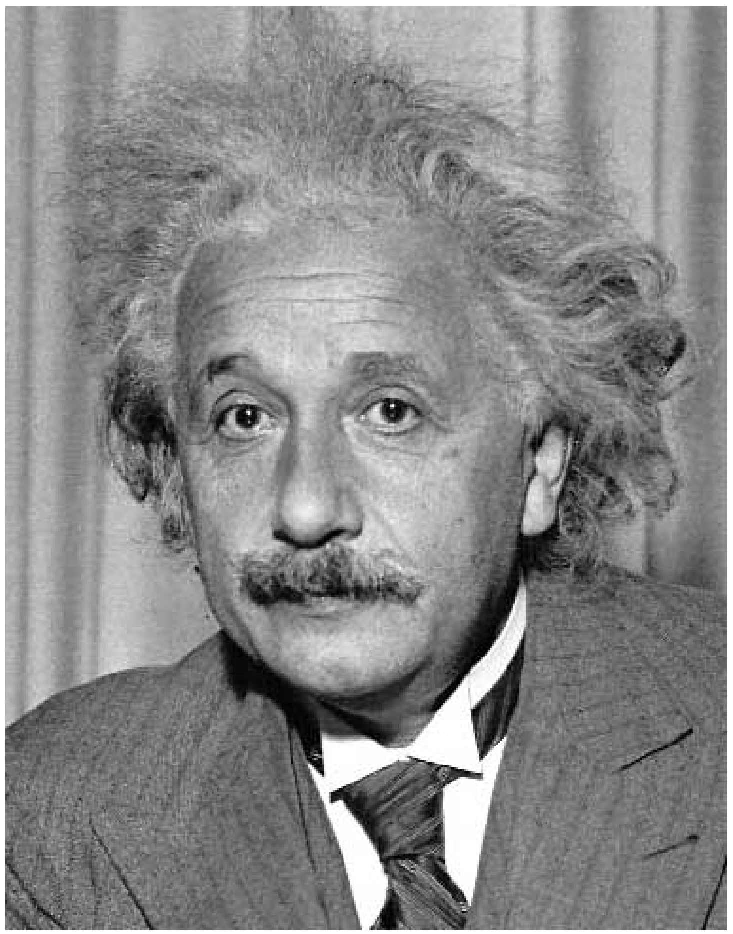


G[i,j] > thresh

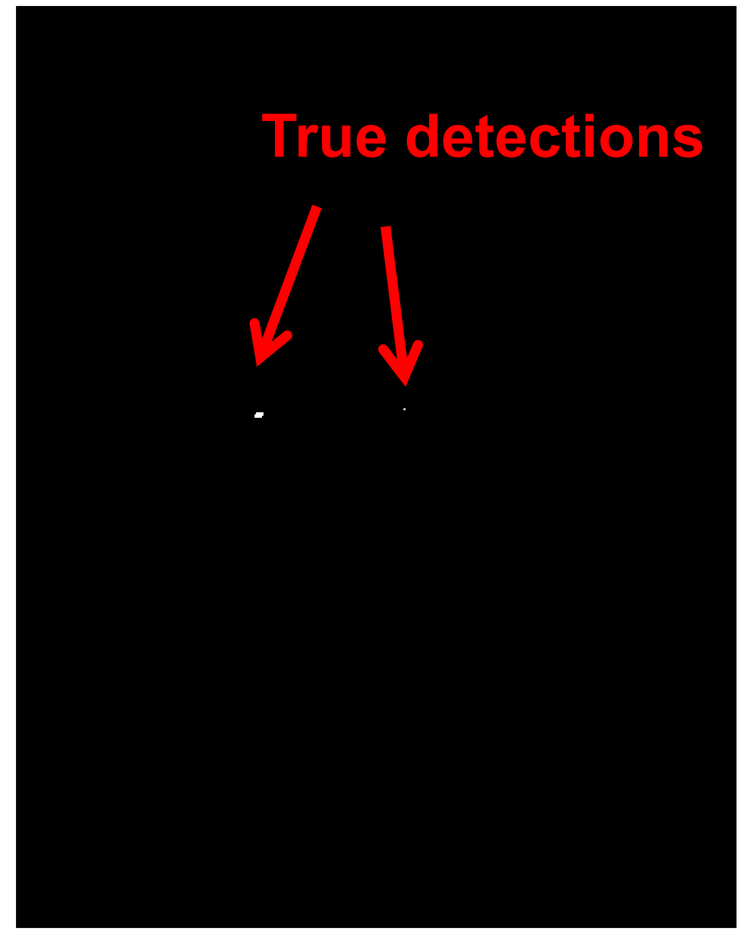
Attempt 2: sum-of-squared-difference (SSD)

$$SSD[i,j] = ||H - F_{ij}||^2$$
$$= (H - F_{ij})^T (H - F_{ij})$$

Can this be implemented with filtering?



-SSD(patch,image)



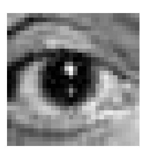
Thresholded image

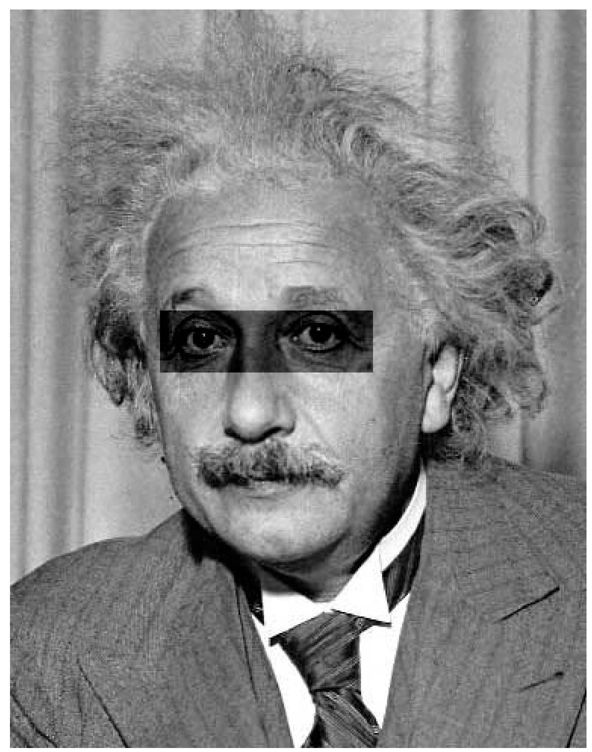
$$SSD[i,j] = H^T H - 2H^T F_{ij} + F_{ij}^T F_{ij}$$

First term: compute "L2 norm" of filter, offline
Third term: convolve image-elementwise-multipled-by-itself with a filter of all '1's.
Second term: correlation of image F with filter H

What will SSD find here?

-SSD(patch,image)





(where eyes have been darkened by .5 scale factor)

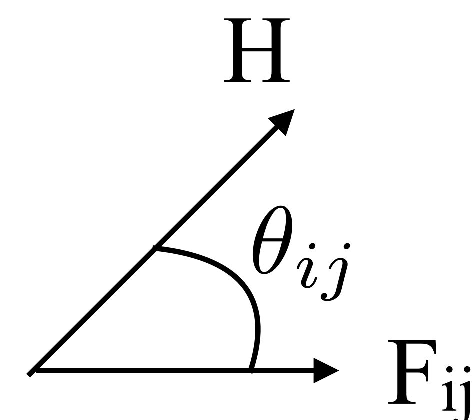
SSD will fire on shirt

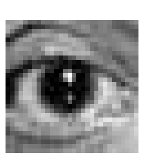
Normalized cross correlation (NCC)

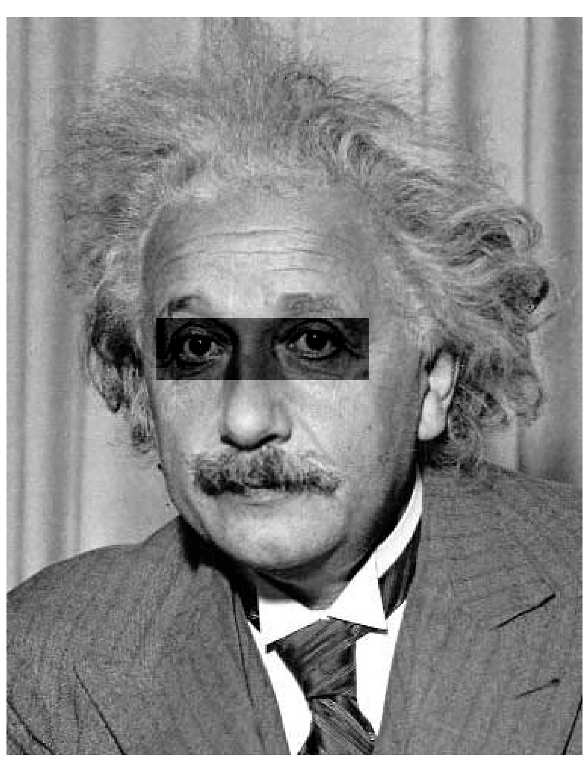
$$NCC[i,j] = \frac{H^T F_{ij}}{||H||\,||F_{ij}||}$$

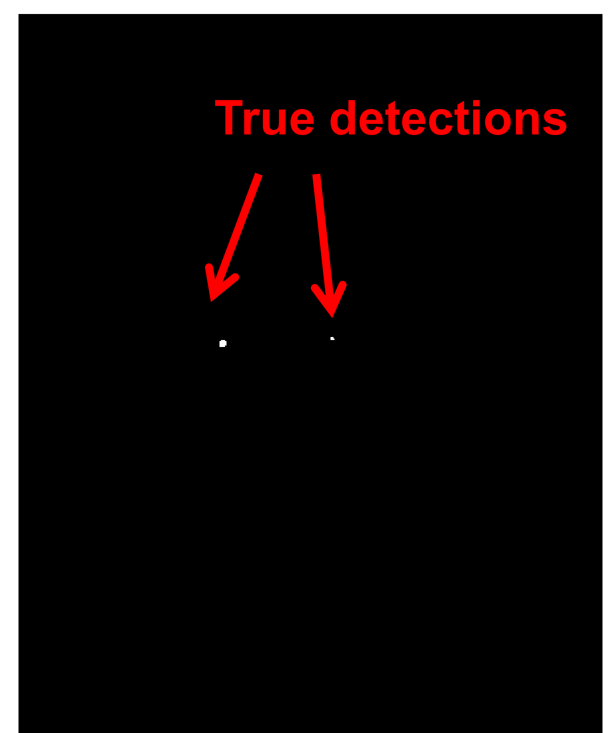
$$= \frac{H^T F_{ij}}{\sqrt{H^T H}\sqrt{F_{ij}^T F_{ij}}}$$

$$= \cos\theta$$





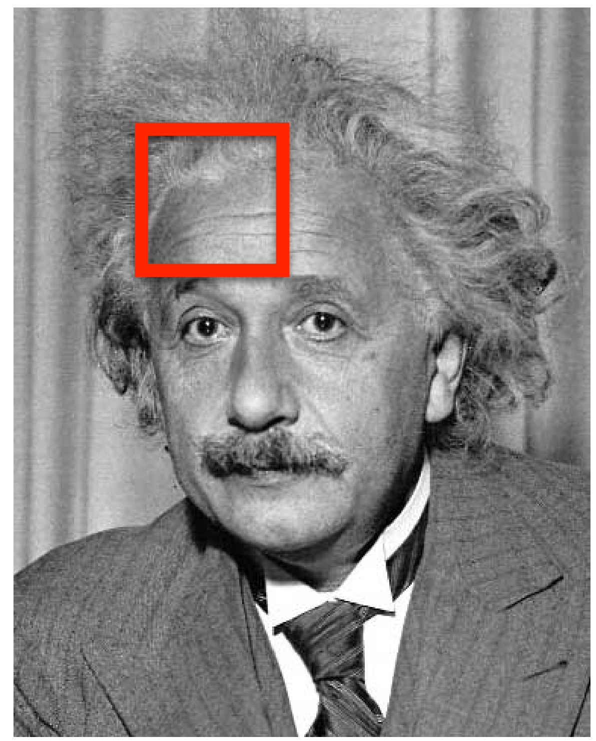


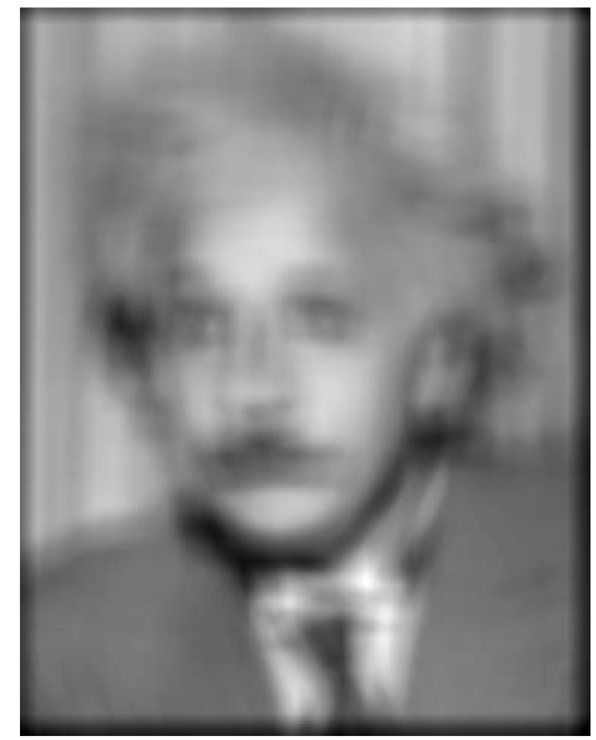


SSD vs CC vs NCC

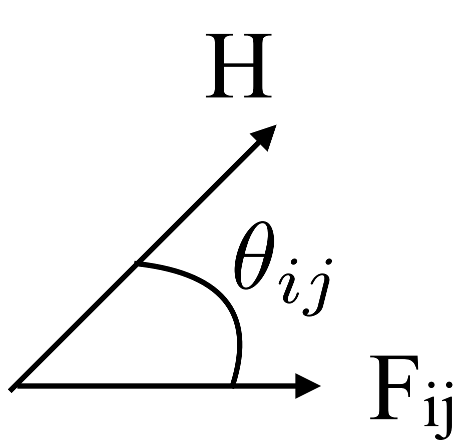
H

F_{ij}





Treat patches (H, F_{ij}) as vectors in R^N



$$SSD_{ij} = ||H - F_{ij}||^2$$

$$CC_{ij} = H^T F_{ij}$$

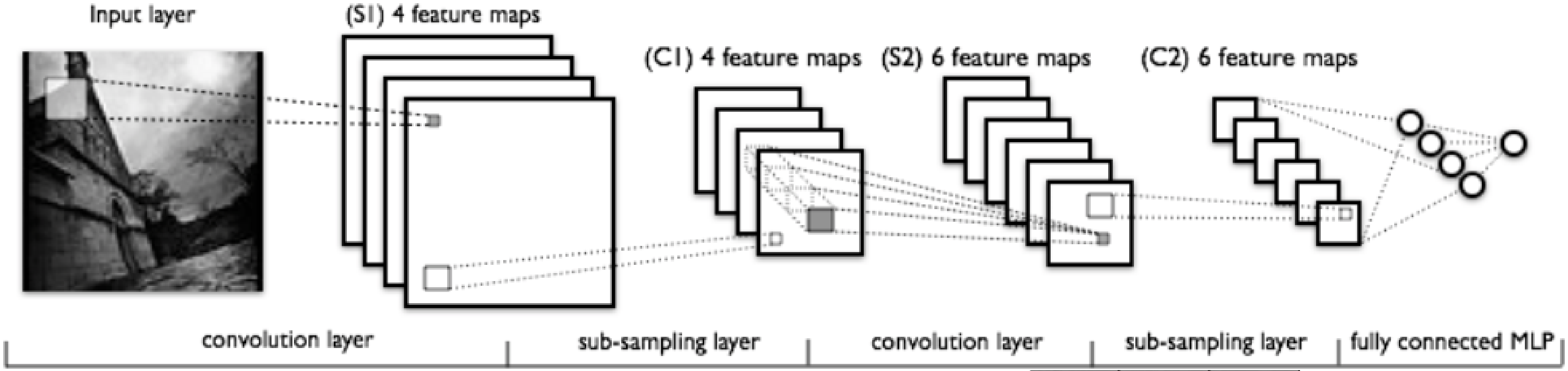
$$NCC_{ij} = \cos(\theta_{ij})$$

[see demo_template_match.pynb]

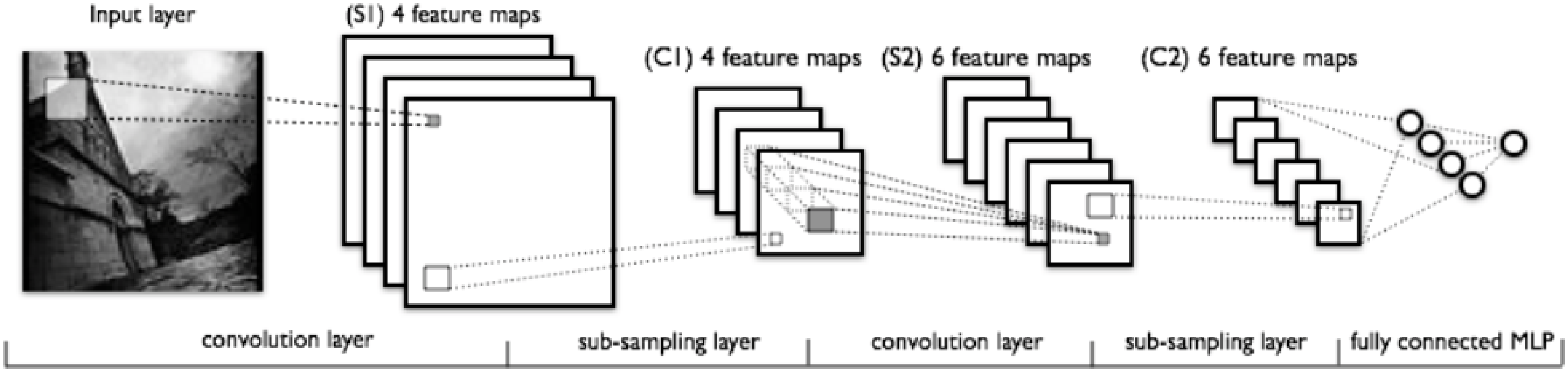
Modern filter banks

Convolutional Neural Nets (CNNs) Lecun et al 98

Learn filters from training data to look for low, mid, and high-level features



Convolutional neural nets

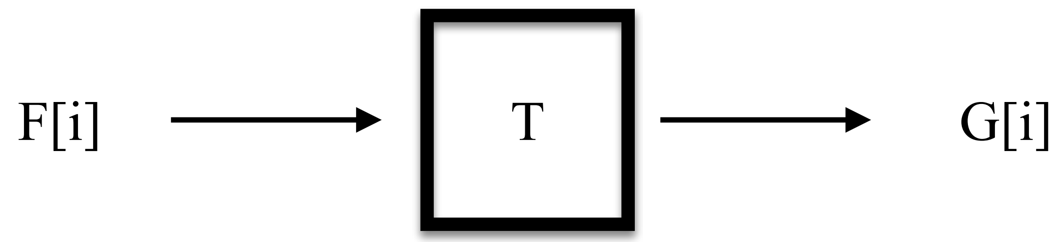


$$G = \max(0, F)$$

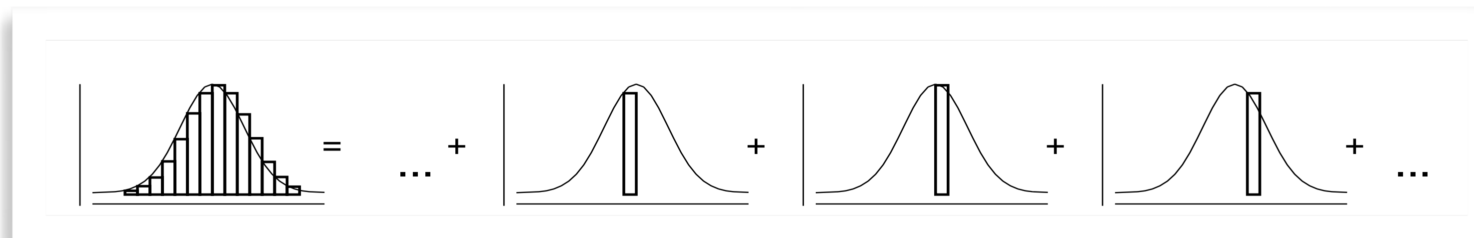
Elementwise rectification or "RELU" - rectified linear unit

Is this operation linear shift-invariant?

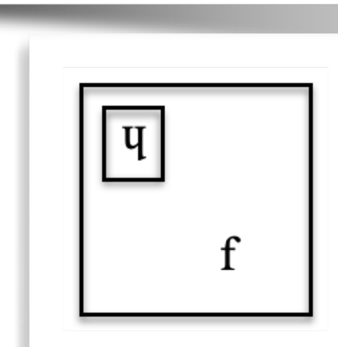
Outline

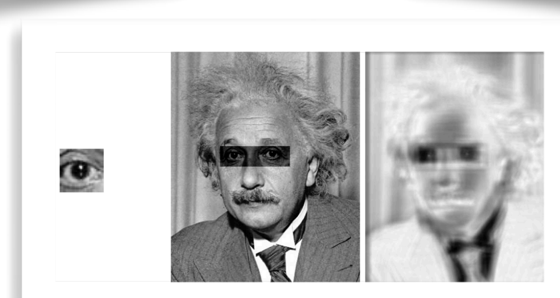


- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalized Cross-Correlation (NCC)



$$G[i] = \sum_{u} F[u]H[i-u]$$





Outline

- Convolution
  - Linear Shift Invariance (LSI)
  - Convolution Properties (commutative, associative, distributive)
  - Normalized Cross-Correlation
- **Edges**
  - Canny edges (hysterisis)
  - derivative-of-gaussians (DoG)
  - laplacian-of-Gaussians (LoG)
- Efficiency
  - pyramids
  - linear approximiations (SVD, separability)
  - steerability