

Computer Vision

CS/ECE 8690

Filiz Bunyak Ersoy

Dept of EECS

Course TA: Imad Toubal (CS PhD): itdfh@mail.missouri.edu

CS/ECE 8690 Computer Vision

Instructor:

- Dr. Filiz Bunyak Ersoy bunyak@missouri.edu
- Office: Naka Hall (Engineering Building West) #219
- Lab: Naka Hall #32



TA:

- Imad Toubal (CS PhD): itoubal@mail.missouri.edu



Meeting Time & Location: Tu & Th 11:00AM-12:15PM, Naka 353

CS/ECE 8690 Computer Vision

Course webpage: [Check CANVAS we will post](#)

- Announcements,
- Homework assignments (test data, sample programs etc.),
- Lecture notes,
- Additional reference materials (i.e. programming references)

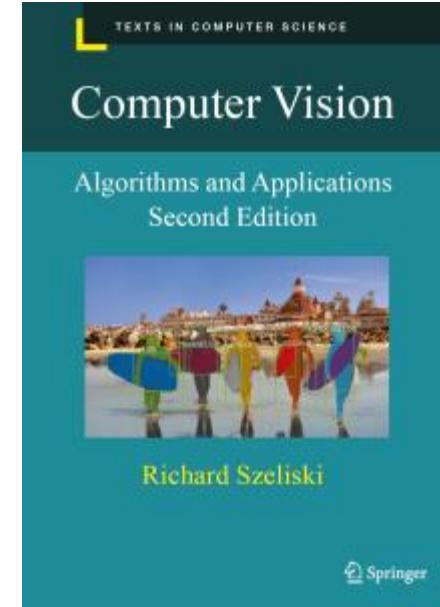
Office Hours:

- **Filiz Bunyak:** Tu & Th 12:30PM-1:30PM, Naka 219
- **Imad Toubal:** TBA
- **Slack channel –** Later this week you will receive an [invite from Imad Toubal](#)

The screenshot shows the course page for "2023SP-CMP_SC-8690-01". The left sidebar contains links for Home, Announcements, Modules, Syllabus, Assignments, Calendar, Grades, People, Support & Policies, Lockdown Browser, Library Resources, MU Connect, Search, Zoom, Discussions, Pages, Rubrics, Files, Quizzes, Collaborations, Outcomes, and Settings. The main content area displays course information: "CS/ECE 8690 Computer Vision", a brief description, textbook information ("Computer Vision: Algorithms and Applications, by Richard Szeliski, Springer, 2nd Edition New York, 2022"), and an electronic draft link. A call-to-action button "Click on Modules to begin" is present. On the right, there are sections for "Faculty Information" (Filiz Bunyak Ersoy, Department: Electrical Engineering and Computer Science, Phone: 573-882 6483, Email: bunyak@missouri.edu, Office Hours: Tu,Th: 12:30PM-1:30PM (Naka 219)) and "Teaching Assistant" (Imad Toubal, Email: itoubal@mail.missouri.edu, Office Hours: TBA). The top right corner includes course status (Unpublish, Published), import options, and course notifications.

Textbook & References

Textbook: Algorithms and Applications, by Richard Szeliski; Springer, 2nd Edition New York, 2022; Electronic draft: <http://szeliski.org/Book>



Other References:

- 1) Computer Vision: A Modern Approach, by David Forsyth and Jean Ponce, ISBN: 0-130-85198-1, Prentice Hall, 2002 or 2nd Edition 2011.
- 2) Papers from CVIU, CVPR, ICCV, ECCV, and various others will be handed out over the semester.
- 3) <https://www.imageprocessingplace.com/>
- 4) OpenCV tutorials https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- 5) Matlab tutorials <https://matlabacademy.mathworks.com/#signal-processing>
- 6) Scikit-image image processing in python <https://scikit-image.org/>

Course Topics

1. Introduction to computer vision
2. Camera, image formation, light and color
3. Image processing review
4. Feature detection and matching
5. Deep learning for computer vision
6. Multi-view image analysis
7. Image segmentation
8. Object recognition
9. Video processing and motion analysis

Organization & Grading

Homework assignments

1. 5 to 7 assignments
2. Programming Python, C++, Matlab
3. Testing using the given data sets
4. Presentation of the results + discussion of the results + theoretical questions

Grading (Subject to Change)

Homeworks (5-to-7)	45%
Quizzes	25%
Final project	25%
Attendance & class participation	5%



Organization & Grading

Homework assignments

1. 5 to 7 assignments
2. Programming Python, C++, Matlab
3. Testing using the given data sets
4. Discussion of the results + theoretical questions

Final Project

1. Individual/group project, plan for about several weeks of effort
2. Software implementation (Python, C++, Matlab)
3. In class presentation at the end of semester
4. IEEE Conference style report
5. Graduate students: Research paper review (IEEE or ACM): ICIP, CVPR, ICCV, ISBI, TIP, PAMI, CSVT, etc.

- Students will have the option to choose their own topic
- The instructor will provide a list of potential project topics
- Students who want to choose their own topic have to obtain the instructor's approval for the project .

Grading (Subject to Change)

Homeworks (5-to-7)	45%
Quizzes	25%
Final project	25%
Attendance & class participation	5%



Collaboration policy

- All homeworks, quizzes, projects should be done individually, unless otherwise stated.
- You can talk each other, to me, to your TA, get advice, ask questions on Slack — but writing and coding should be done individually, and never shared (except when specified in group projects)
- Code and report should not be fully or partially copied from the internet.
- Instructor's approval should be obtained before using any outside code or writing.
- Any code used or adapted from an outside source should clearly cite the source both in the report and in the code.



Late Policy

- Assignments will be submitted to Canvas.
- The submission deadline will be 23:59 on the due date.
- Late submissions will be accepted up to 7 days late, but there will be 10% penalty for each late day.
- You will also have a total of 3 free late days for the semester that will not be penalized.



Changes to the Course

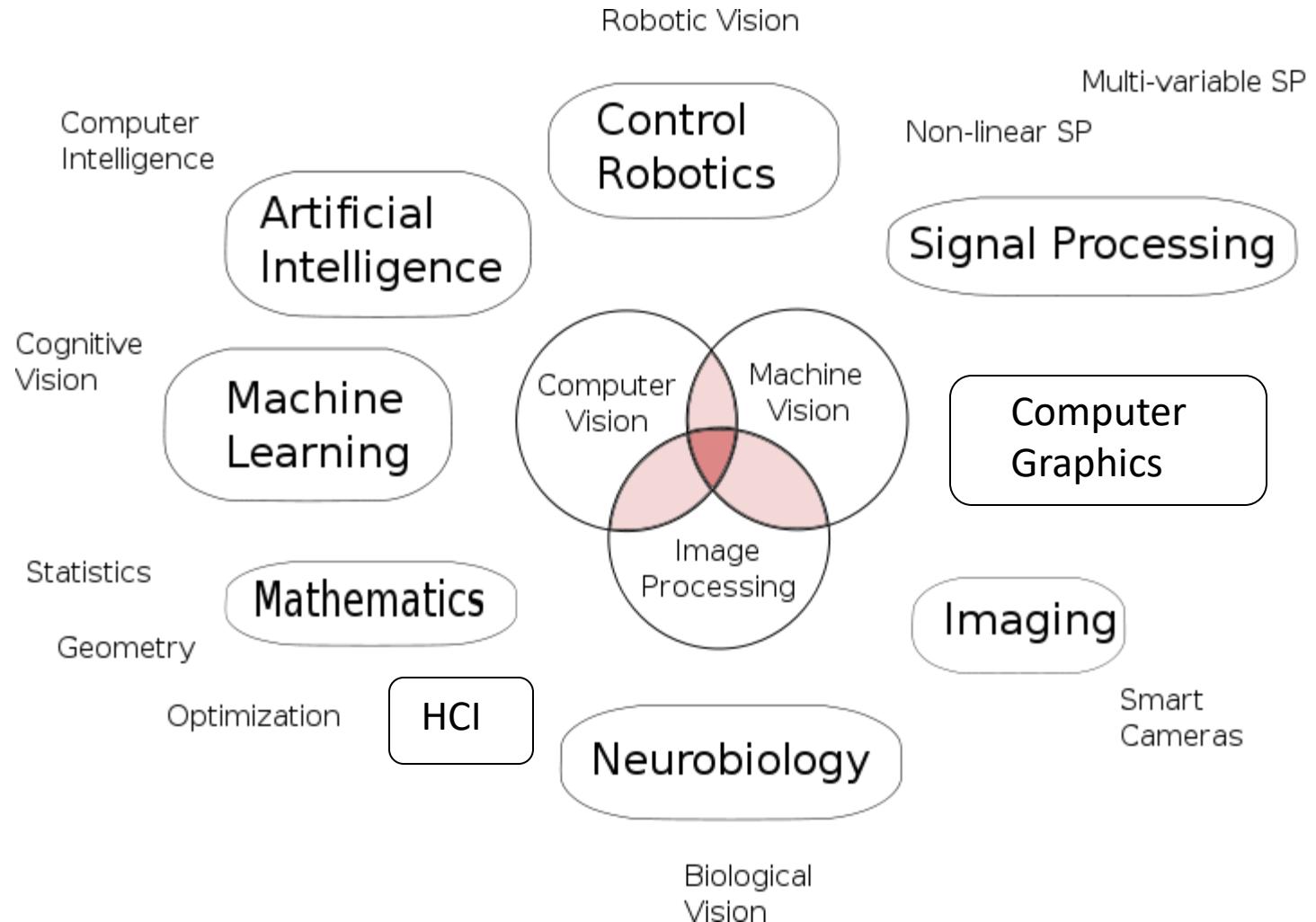
- The instructor reserves the right to make modifications or updates to the syllabus, content and grading scheme during the semester based on student feedback. Students will be informed of changes to the syllabus which will be reflected on the course CANVAS.

Introduction to Computer Vision

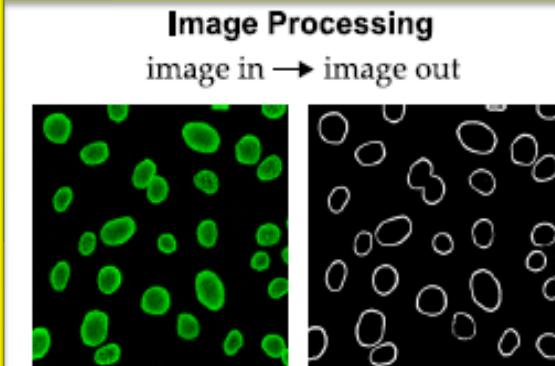
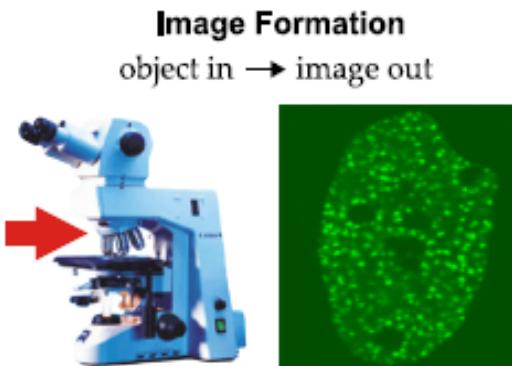
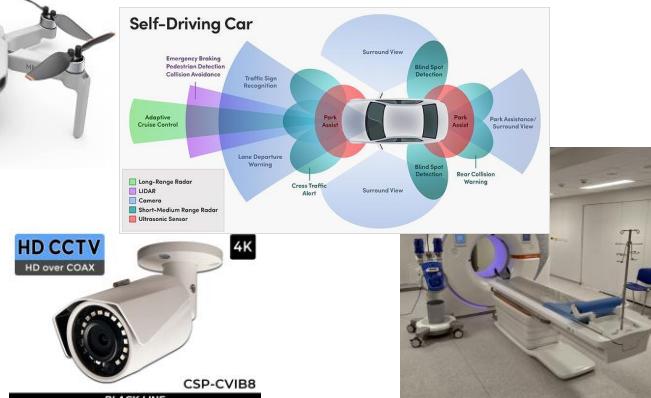
Computer Vision
CS/ECE 8690

Instructor: Filiz Bunyak Ersoy bunyak@missouri.edu
TA: Imad Toubal (CS PhD): itdfh@mail.missouri.edu

Vision is multidisciplinary



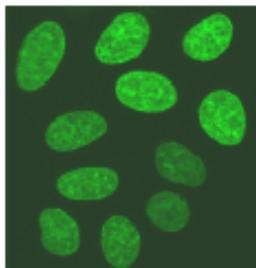
From wiki



Series of operations on image that alter its form or its value.
Result: again an image.

Act of measuring meaningful object features in an image.
Measurement : qualitative/quantitative subjective /objective

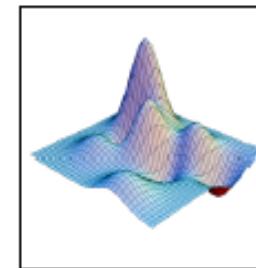
Image Analysis
image in → features out



Obj	Area	Perim
1	324.2	98.5
2	406.7	140.3
3	487.1	159.2
4	226.3	67.8
5	531.8	187.6
6	649.5	203.1
7	582.6	196.4
8	498.0	162.9
9	543.2	195.1

Computer Graphics
numbers in → image out

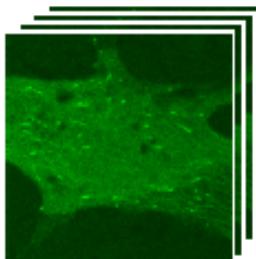
X	Y	I
-3.54	-2.32	0.50
-2.78	-1.90	0.12
-1.15	0.42	3.09
0.45	1.65	5.89
1.83	2.18	7.72
2.98	3.33	2.07
4.21	3.96	-4.58
5.62	4.54	-11.45
7.16	5.02	-3.63



Produce image from given primitives,
(in some sense inverse of image analysis)

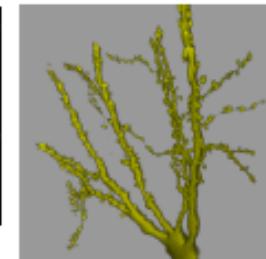
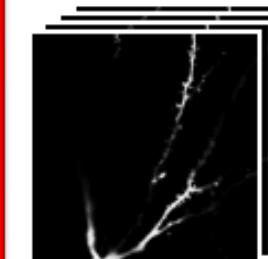
Produce high-level interpretation of what is contained in image.
Measurement + reasoning + making inferences.

Computer Vision
image in → interpretation out



The series shows microtubule growth in a live neuron. The average speed of the distal ends is comparable in the cell body, dendrites, axons, and growth cones.

Visualization
image in → representation out



High-dimension image data into a more **primitive representation** to facilitate exploring the data.

Relationships Between Computer Vision & Related Fields

- Human Visual System, Human Perception
- Optics/Physics of Light
- Architecture, Hardware, Cameras, Sensors
- Embedded processing, Parallel computing
- Signal Processing, Coding, Compression
- Video Processing
- Computer Vision
- AI, Machine Learning, Pattern Recognition
- Computer Graphics
- Multimedia Systems, CBIR
- Applied Statistics
- Numerical Methods, Optimization, Mathematical Models

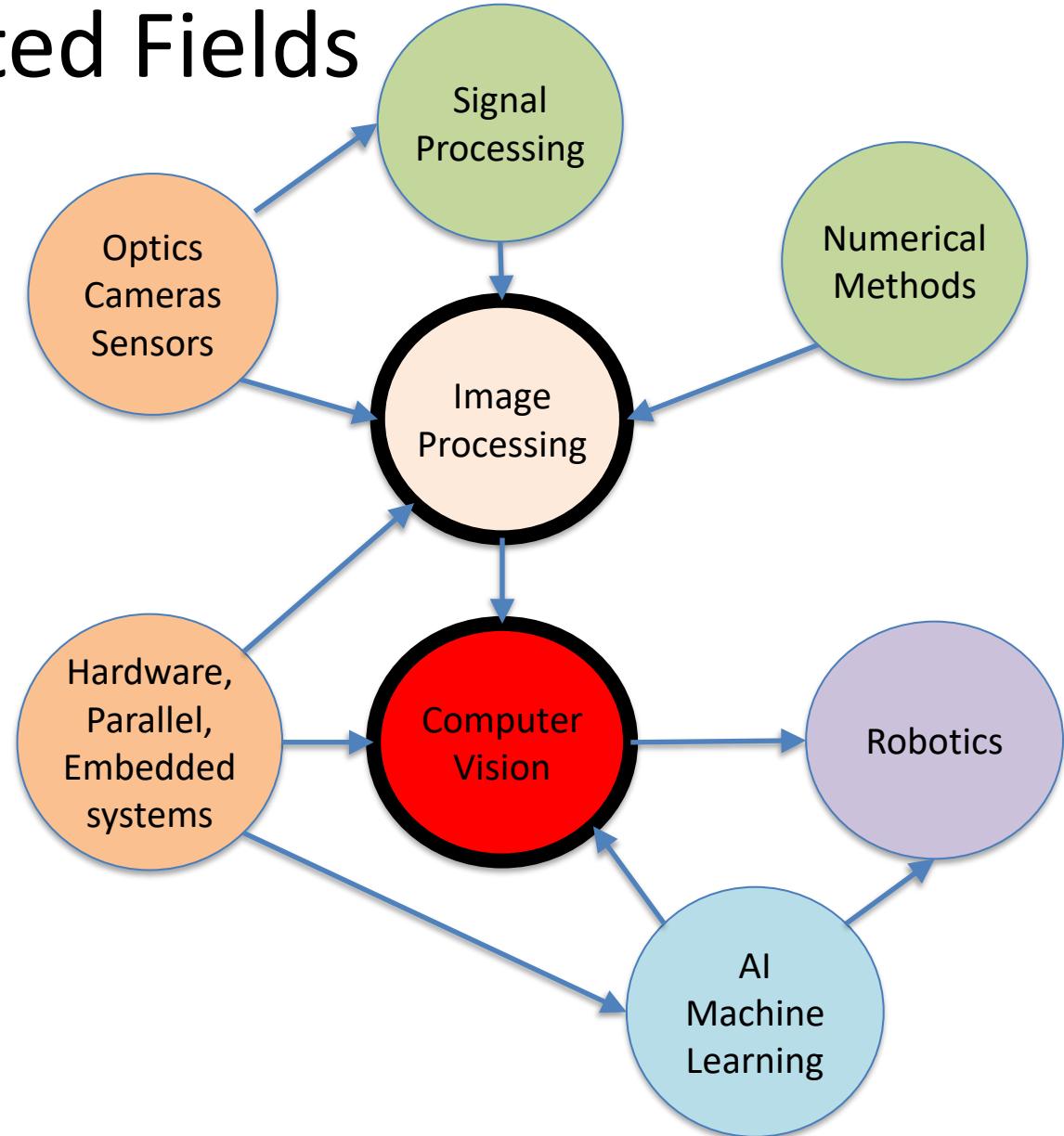


Image Processing System

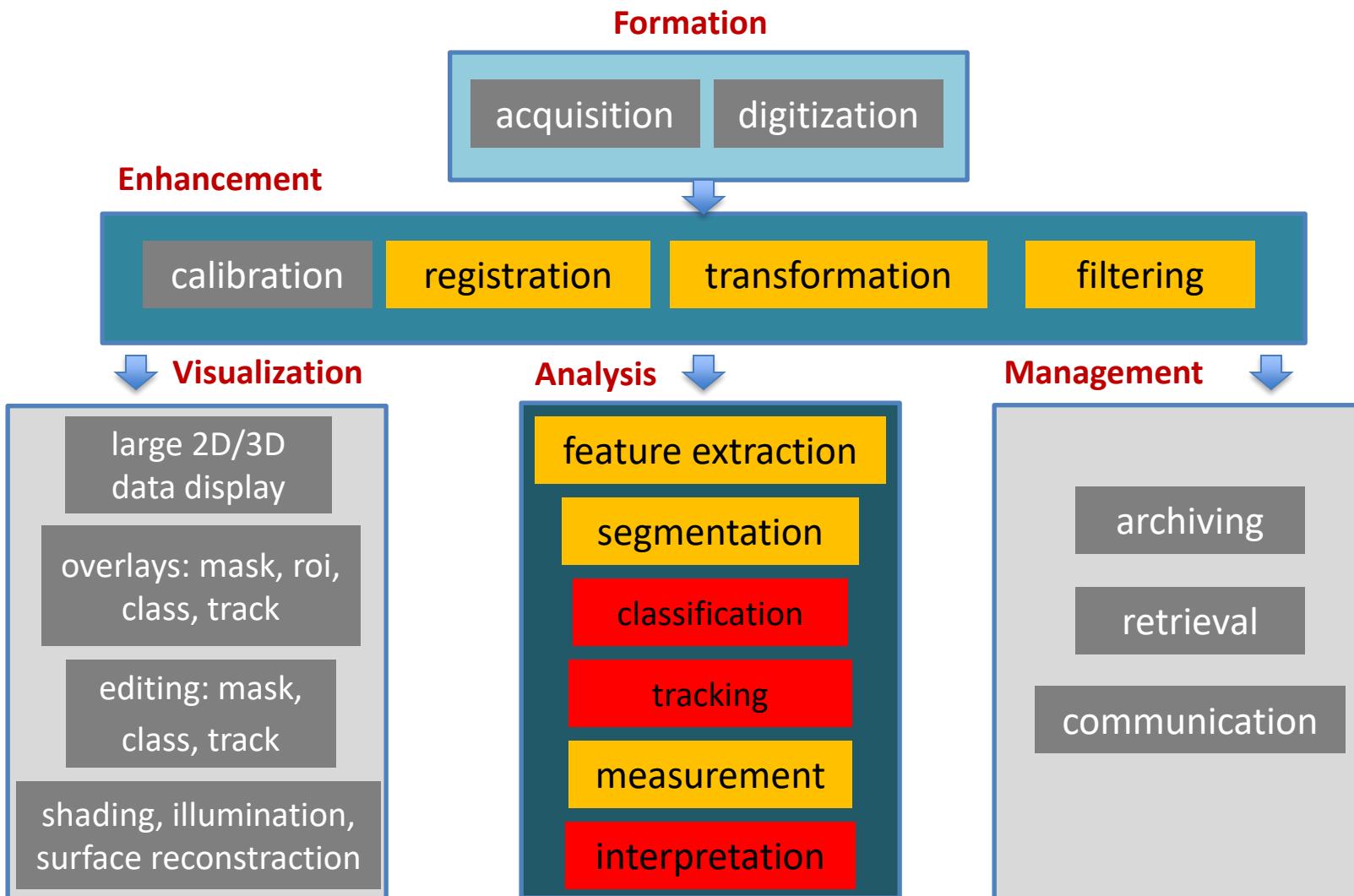
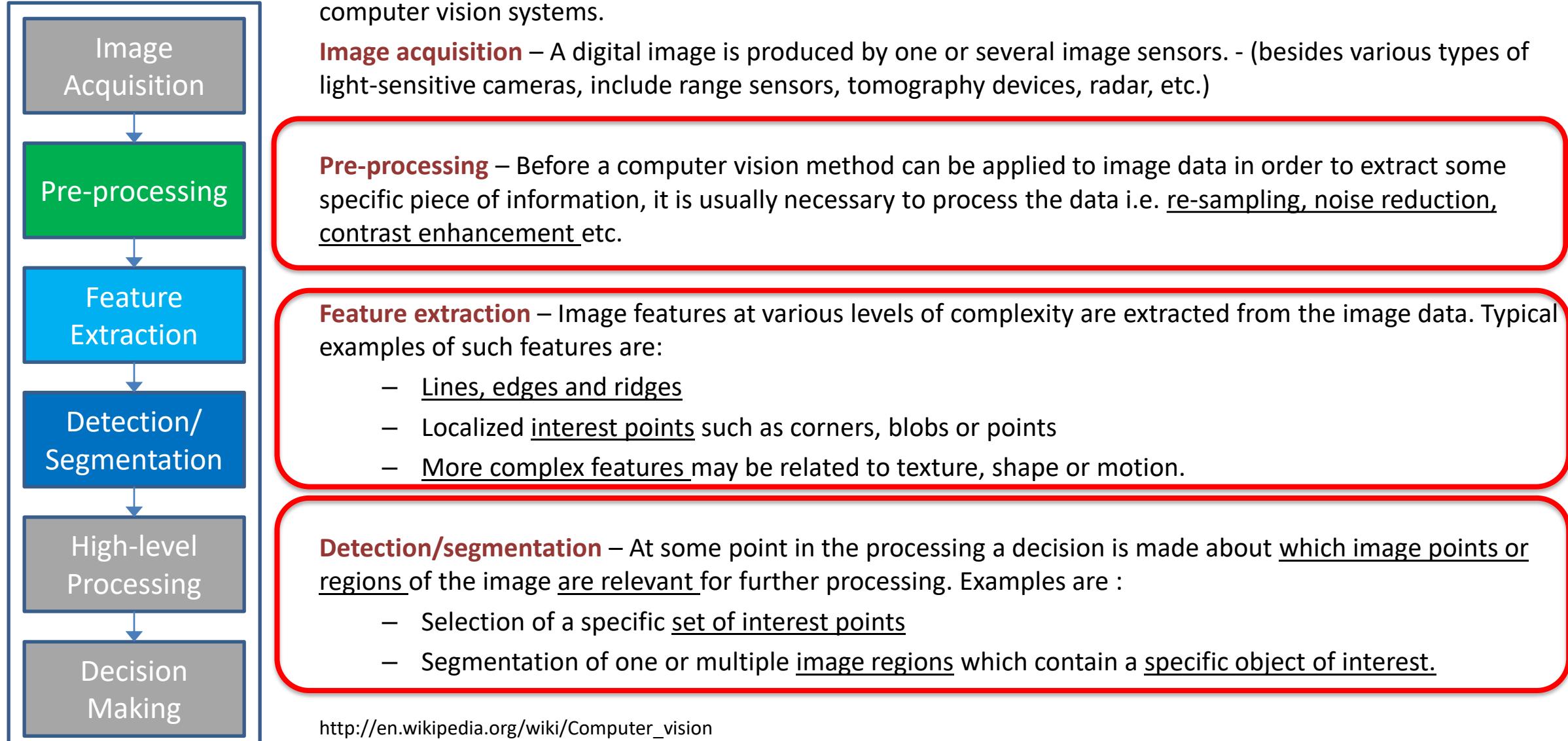


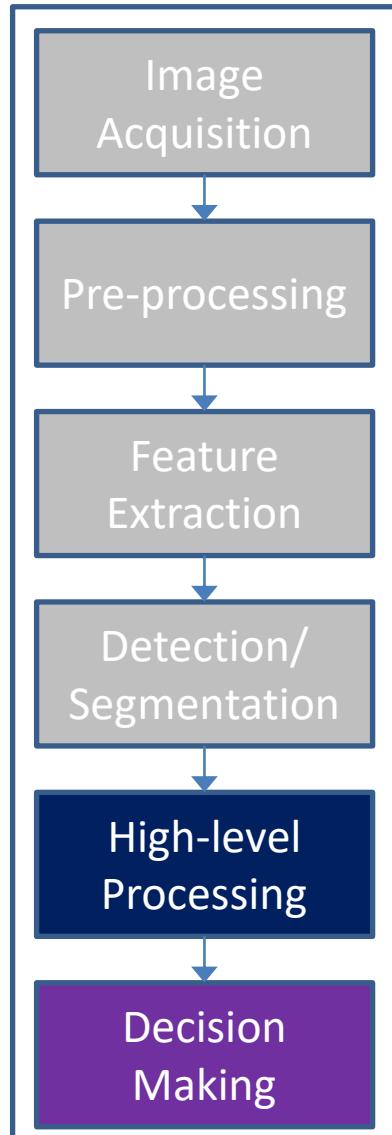
Figure adapted from: Biomedical Image Processing by Thomas M. Deserno

Typical Tasks in Computer Vision Systems

Actual tasks are highly application dependent, but there are typical functions which are found in many computer vision systems.



Typical Tasks in Computer Vision Systems



Actual tasks are highly application dependent, but there are typical functions which are found in many computer vision systems.

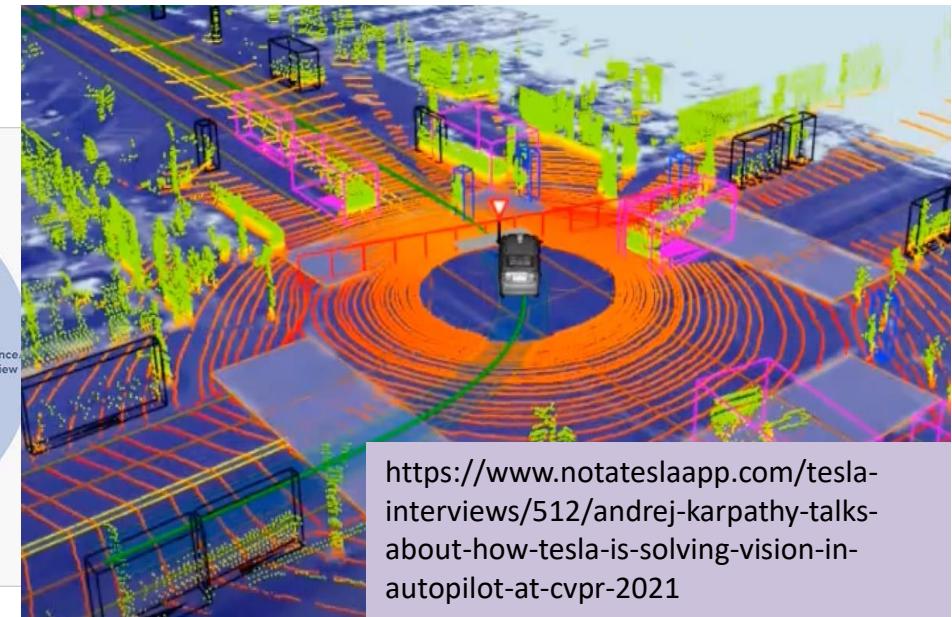
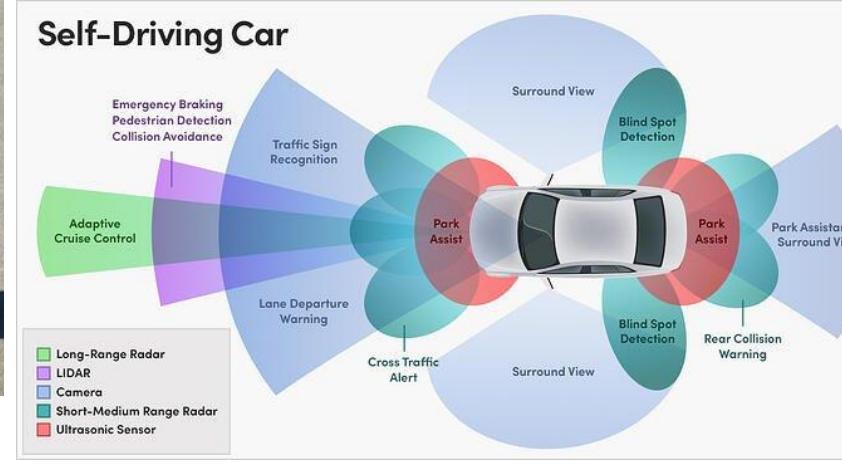
- **High-level processing :** At this step the input is typically a small set of data, for example a set of points or an image region which is assumed to contain a specific object. The remaining processing deals with, for example:
 - Verification that the data satisfy model-based and application specific assumptions.
 - Estimation of application specific parameters, such as object pose or object size.
 - Image recognition – classifying a detected object into different categories.
 - Image registration – comparing and combining two different views of the same object.
- **Decision making:** Making the final decision required for the application, for example:
 - Pass/fail on automatic inspection applications
 - Match/no-match in recognition applications
 - Etc.....

MOTIVATION & SOME APPLICATIONS!

Self-Driving Cars

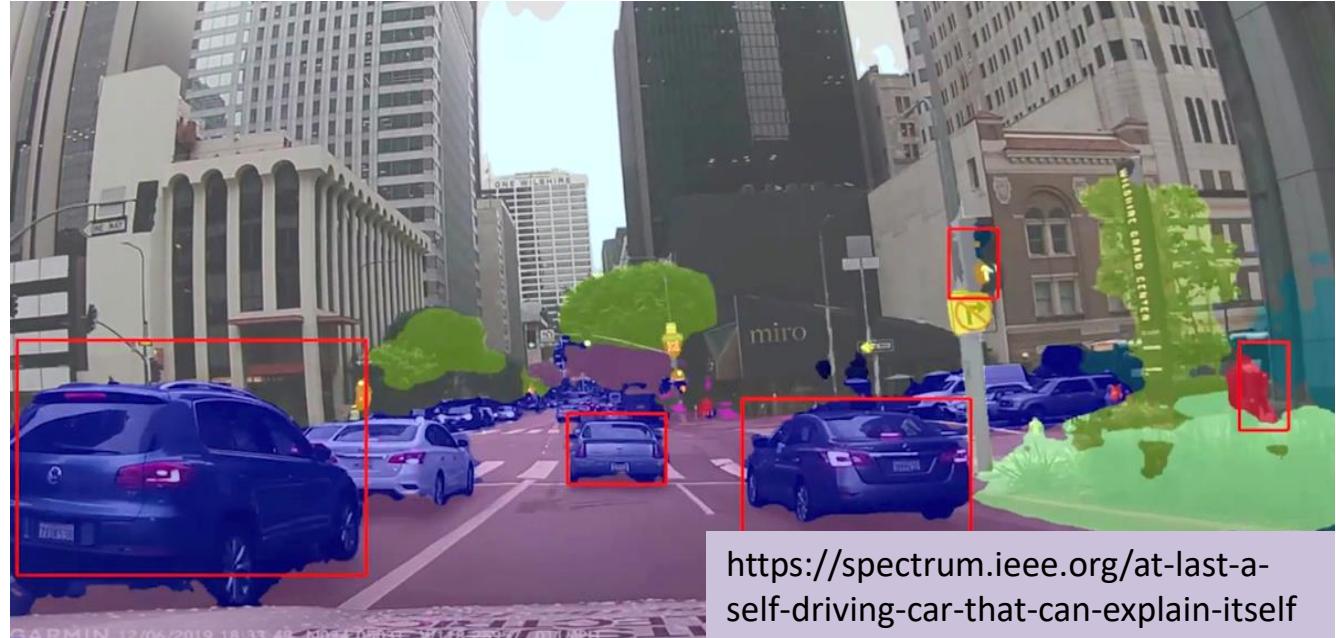


self-driving cars, (Montemerlo,
Becker et al.2008) © 2008 Wiley
(Szeliski Computer Vision Book)



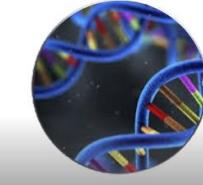
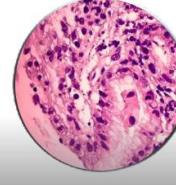
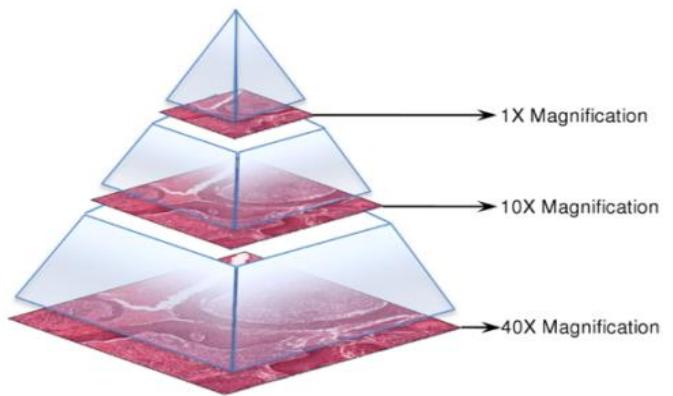
<https://www.notateslaapp.com/tesla-interviews/512/andrej-karpathy-talks-about-how-tesla-is-solving-vision-in-autopilot-at-cvpr-2021>

<https://www.nvidia.com/en-us/research/computer-vision/>

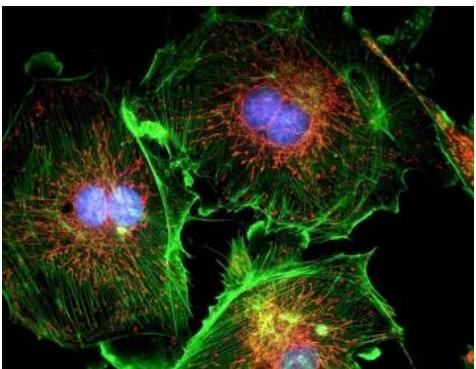


<https://spectrum.ieee.org/at-last-a-self-driving-car-that-can-explain-itself>

BioMedical Image Analytics



A screenshot of the NVIDIA CLARA software interface. The top navigation bar includes "NVIDIA CLARA", "WELCOME", "DATA", "MODELS", "OPERATORS", "PIPELINES", "JOBS", and "DOCS". Below the navigation bar, there is a search bar with the placeholder "Search pipelines...". Under the "PIPELINES" tab, four pipeline cards are listed: "Brain Tumor Segmentation" (status: 0 IN QUEUE), "De Novo Sequence Assembly" (status: 1 IN QUEUE), "Endoscopy Object Detection" (status: 0 IN QUEUE), and "Multi Organ Segmentation" (status: 3 IN QUEUE). The bottom of the window has a green bar with the text "NVIDIA EGX".



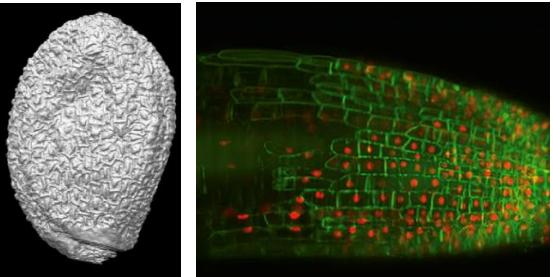
<https://developer.nvidia.com/blog/deploying-healthcare-ai-workflows-with-clara-deploy-app-framework-updated/>

Imaging is now used as a tool for discovery throughout basic life sciences, and biomedical & clinical research.

Plant Sciences & Precision Agriculture



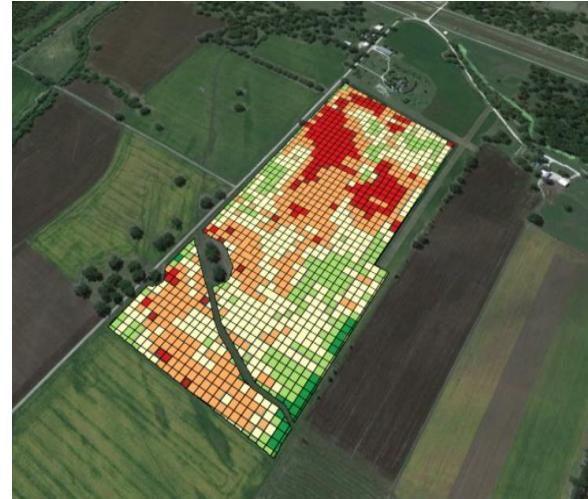
<http://blog.machinefinder.com/24533/>john-deere-agriculture-data-webinar-recap



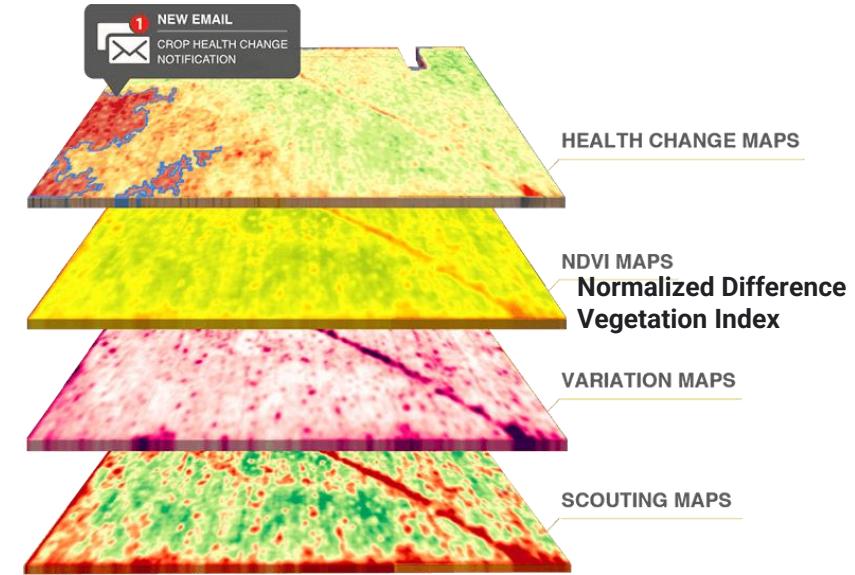
Micro-tomography
image of seed
Dr. Mendoza

Fluorescent microscopy
Cells @ root tip

[http://www.embl.de/training/
events/2015/PLA15-01/](http://www.embl.de/training/events/2015/PLA15-01/)

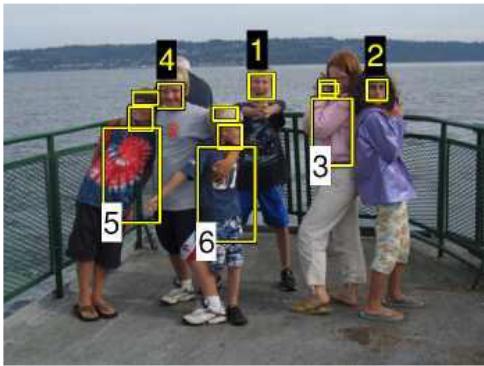


[http://dronelife.com/wpcontent/
uploads/2014/12/Agribotix-drone-created-
fertilizer-prescription-map.png](http://dronelife.com/wpcontent/uploads/2014/12/Agribotix-drone-created-fertilizer-prescription-map.png)



<https://www.farmersedge.ca/satellite-imagery/>

Surveillance, Security, Traffic...



<https://www.nytimes.com/2020/01/19/style/ring-video-doorbell-home-security.html>

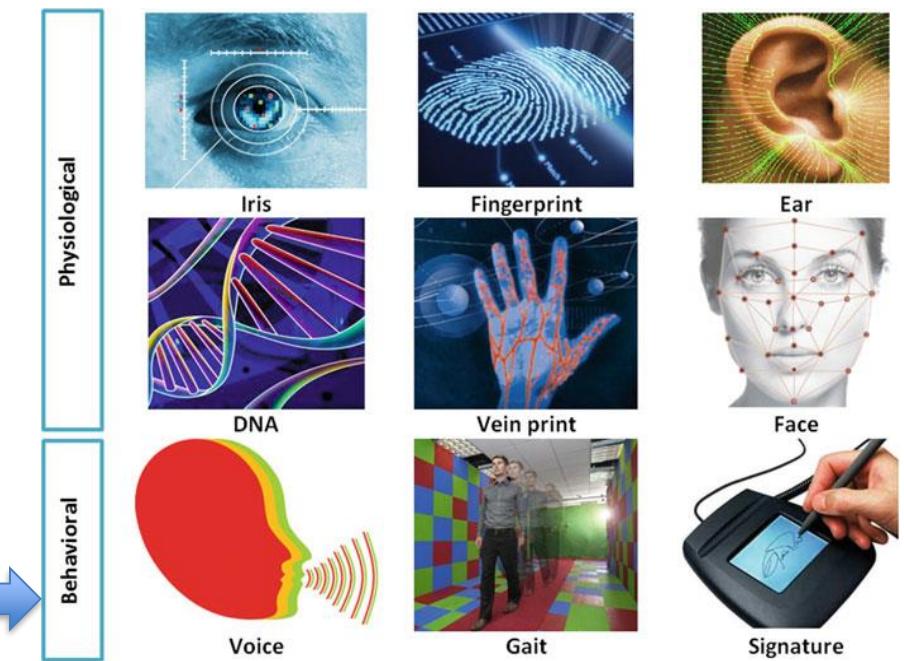
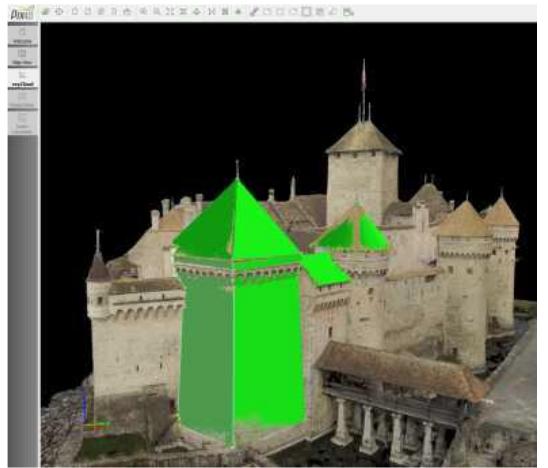


<https://spacenet.ai/>



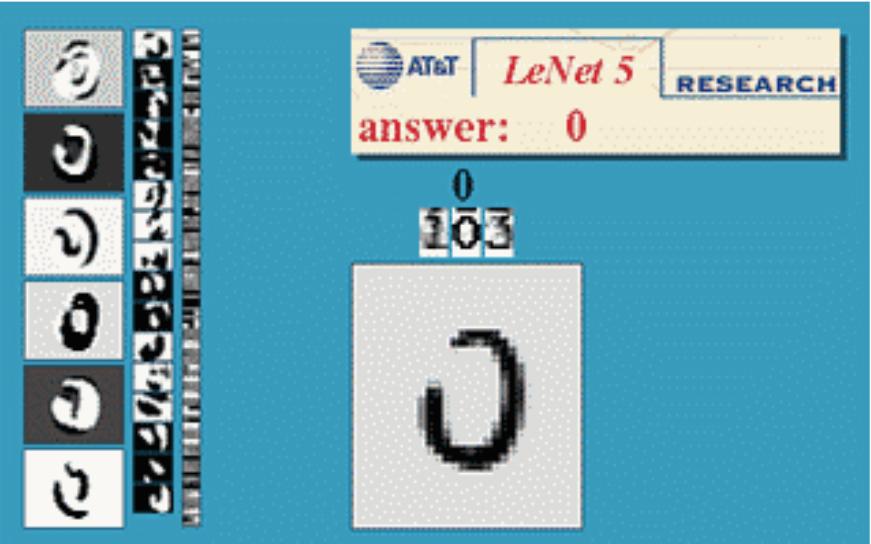
Bouchrika, Imed. "A survey of using biometrics for smart visual surveillance: Gait recognition." In *Surveillance in Action*, pp. 3-23. Springer, Cham, 2018

drone-based photogrammetry,
<https://www.pix4d.com/blog/mapping-chillon-castle-with-drone>.
(Szeliski, Computer Vision Book)



Some industrial applications of computer vision

(Szeliski, R., 2021. *Computer vision: algorithms and applications*. Chapter 1)



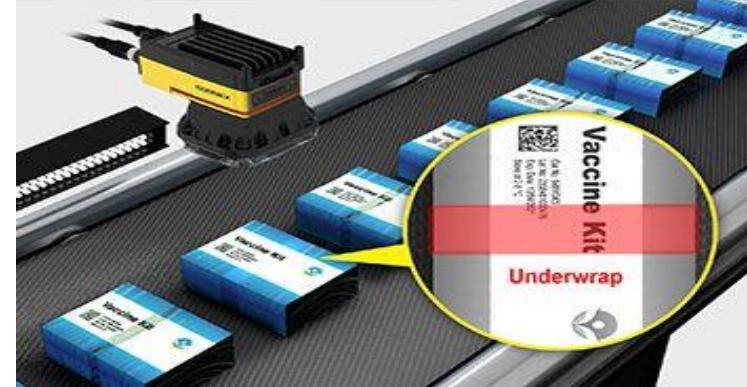
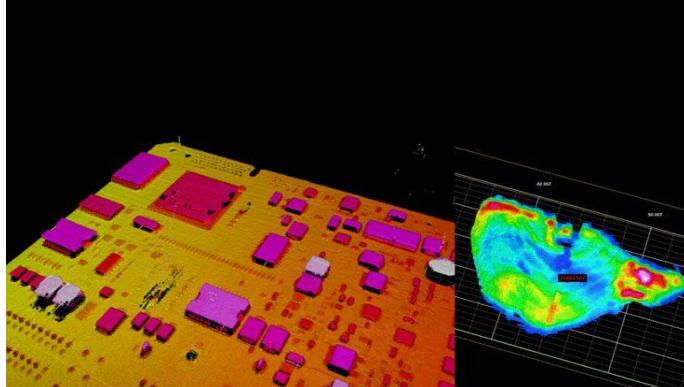
optical character recognition (OCR)
<http://yann.lecun.com/exdb/lenet>



mechanical inspection,
<http://www.cognitens.com>



warehouse picking, <https://covariant.ai>



Some consumer applications of computer vision

Szeliski, R., 2021. *Computer vision: algorithms and applications*. Chapter 1



image stitching: merging different views
(Szeliski and Shum 1997) © 1997 ACM



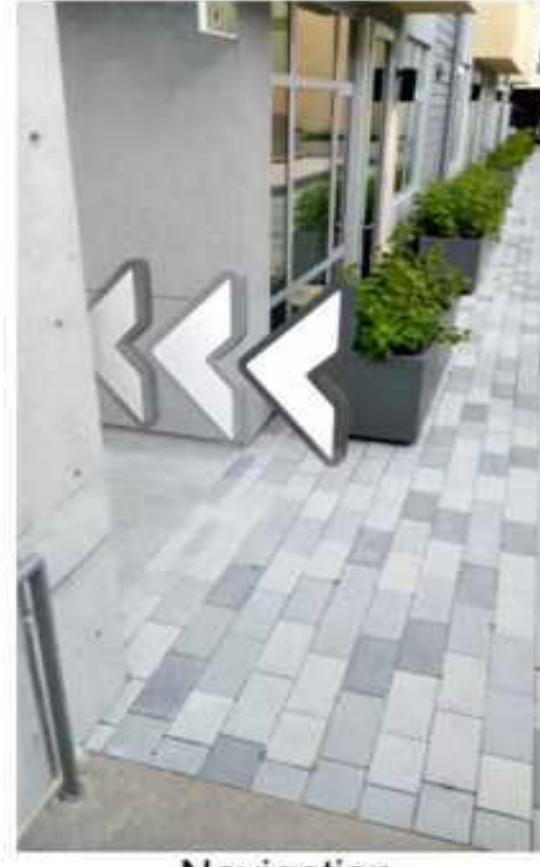
exposure bracketing:
Merging different
exposures



morphing: blending between two
photographs (Gomes, Darsa et
al. 1999) © 1999 Morgan Kaufmann

Some consumer applications of computer vision

Szeliski, R., 2021. *Computer vision: algorithms and applications*. Chapter 1



Examples of computer vision algorithms from the 1990s

Szeliski, R., 2021. *Computer vision: algorithms and applications*. Chapter 1



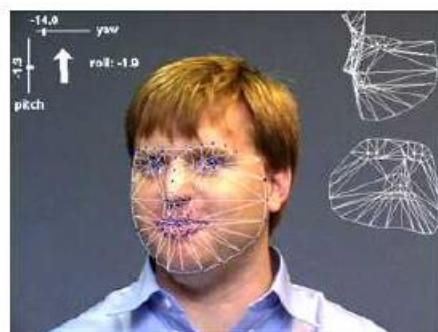
(a)



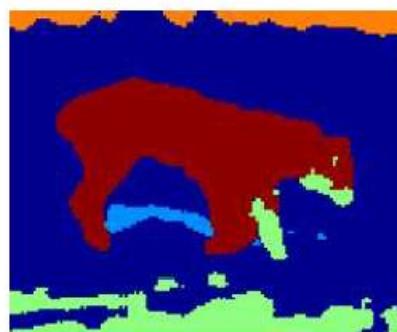
(b)



(c)



(d)



(e)



(f)

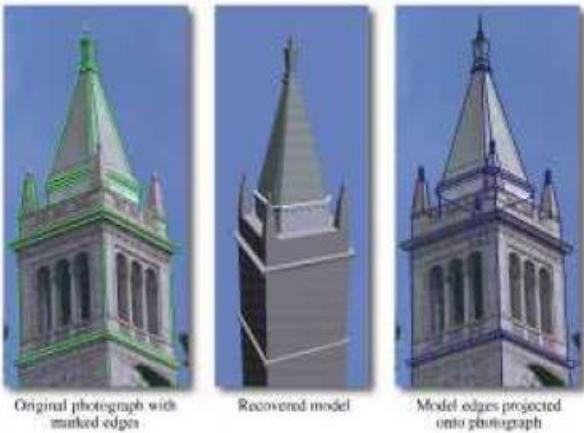
- (a) Factorizationbased structure from motion (Tomasi and Kanade 1992) © 1992 Springer,
- (b) dense stereo matching (Boykov, Veksler, and Zabih 2001),
- (c) multi-view reconstruction (Seitz and Dyer 1999) © 1999 Springer,
- (d) face tracking (Matthews, Xiao, and Baker 2007),
- (e) image segmentation (Belongie, Fowlkes et al. 2002) © 2002 Springer,
- (f) face recognition (Turk and Pentland 1991).

Examples of computer vision algorithms from the 2000s

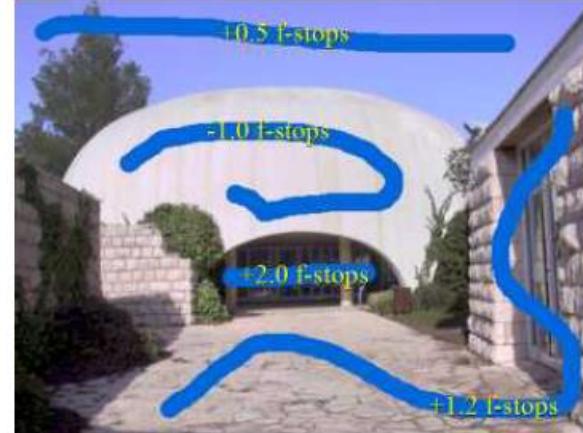
Szeliski, R., 2021. *Computer vision: algorithms and applications*. Chapter 1



(a)



(b)



(c)



(d)



(e)

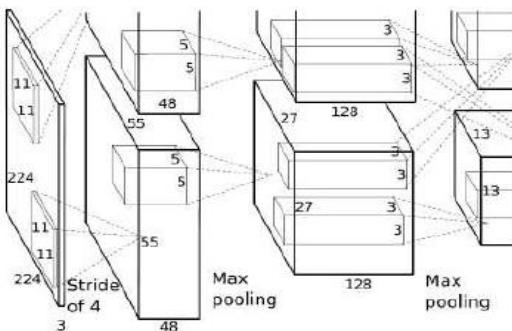


(f)

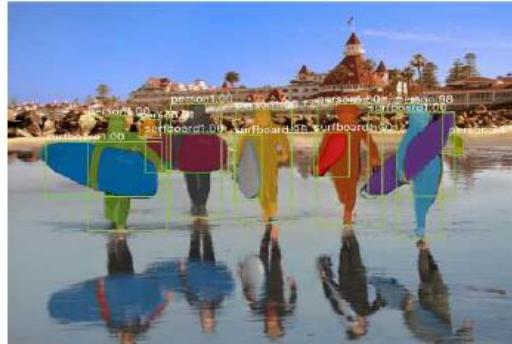
- (a) image-based rendering (Gortler, Grzeszczuk et al. 1996),
- (b) image-based modeling (Debevec, Taylor, and Malik 1996) © 1996 ACM,
- (c) interactive tone mapping (Lischinski, Farbman et al. 2006)
- (d) texture synthesis (Efros and Freeman 2001),
- (e) feature-based recognition (Fergus, Perona, and Zisserman 2007),
- (f) region-based recognition (Mori, Ren et al. 2004) © 2004 IEEE.

Examples of computer vision algorithms from the 2010s

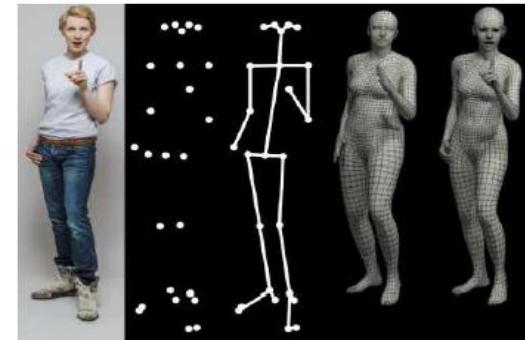
Szeliski, R., 2021. *Computer vision: algorithms and applications*. Chapter 1



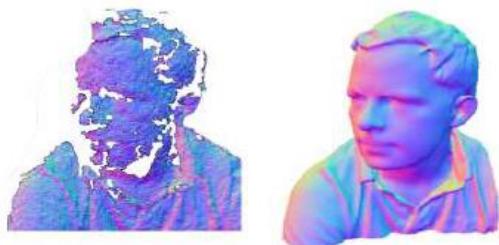
(a)



(b)



(c)



(d)



(e)



(f)

- (a) the SuperVisiondeep neural network © Krizhevsky, Sutskever, and Hinton (2012);
- (b) object instance segmentation(He, Gkioxari et al. 2017) © 2017 IEEE;
- (c) whole body, expression, and gesture fitting from a single image (Pavlakos, Choutas et al. 2019) © 2019 IEEE;
- (d) fusing multiple color depth images using the KinectFusion real-time system (Newcombe, Izadi et al. 2011) © 2011 IEEE;
- (e) smartphone augmented reality with real-time depth occlusion effects (Valentin, Kowdle et al. 2018) © 2018 ACM;
- (f) 3D map computed in real-time on a fully autonomous Skydio R1 drone (Cross 2019).

Widely Available Imaging Image Processing & Computer Vision

- Smartphone cameras
- Surveillance cameras
- Social networks
- Biometrics
- Robotics
- 3D Vision
- Autonomous systems: UAVs, UGVs, UWVs
- Pattern recognition
- Video conferencing, Image compression, restoration
- Virtual reality (VR), Augmented reality (AR), Mixed reality (MR)
- Media: Entertainment, Film, News
- Document processing, OCR, handwriting
- Manufacturing
- Geospatial imagery, Remote sensing, Agriculture, Mining
- Space imaging
- Defense imaging
- Healthcare, Biomedical

5 Biggest Computer Vision Trends in 2022

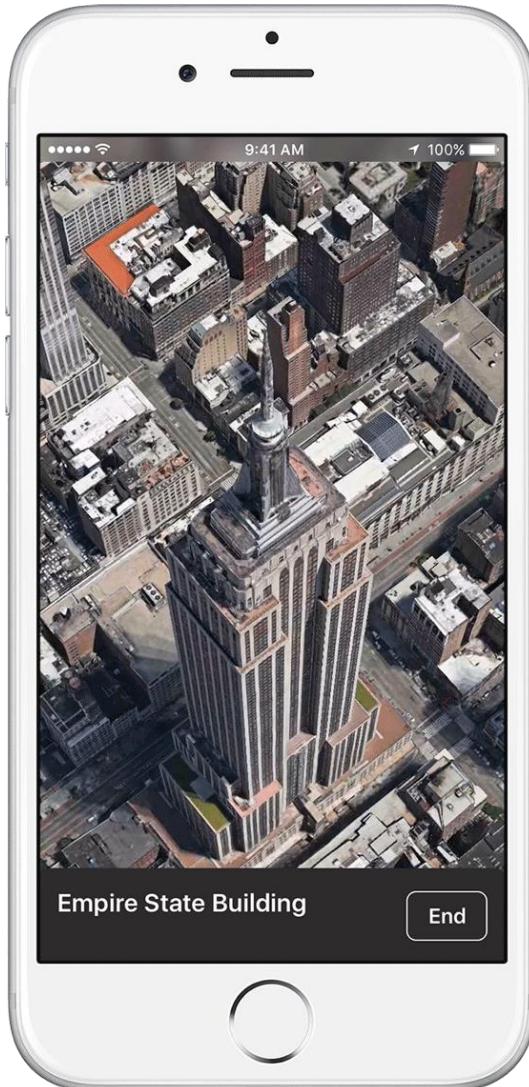
Forbes Mar 2022

(<https://www.forbes.com/sites/bernardmarr/2022/03/04/the-5-biggest-computer-vision-trends-in-2022/?sh=5abcca9f19b3>)

1. Data-centric computer vision
2. Computer vision in health and safety
3. Computer vision in retail
4. Computer vision in connected and autonomous cars
5. Computer vision at the edge (as close as possible to the data source)

SOME COMPUTER VISION EXAMPLES FROM DR. Szeliski (COMPUTER VISION BOOK)

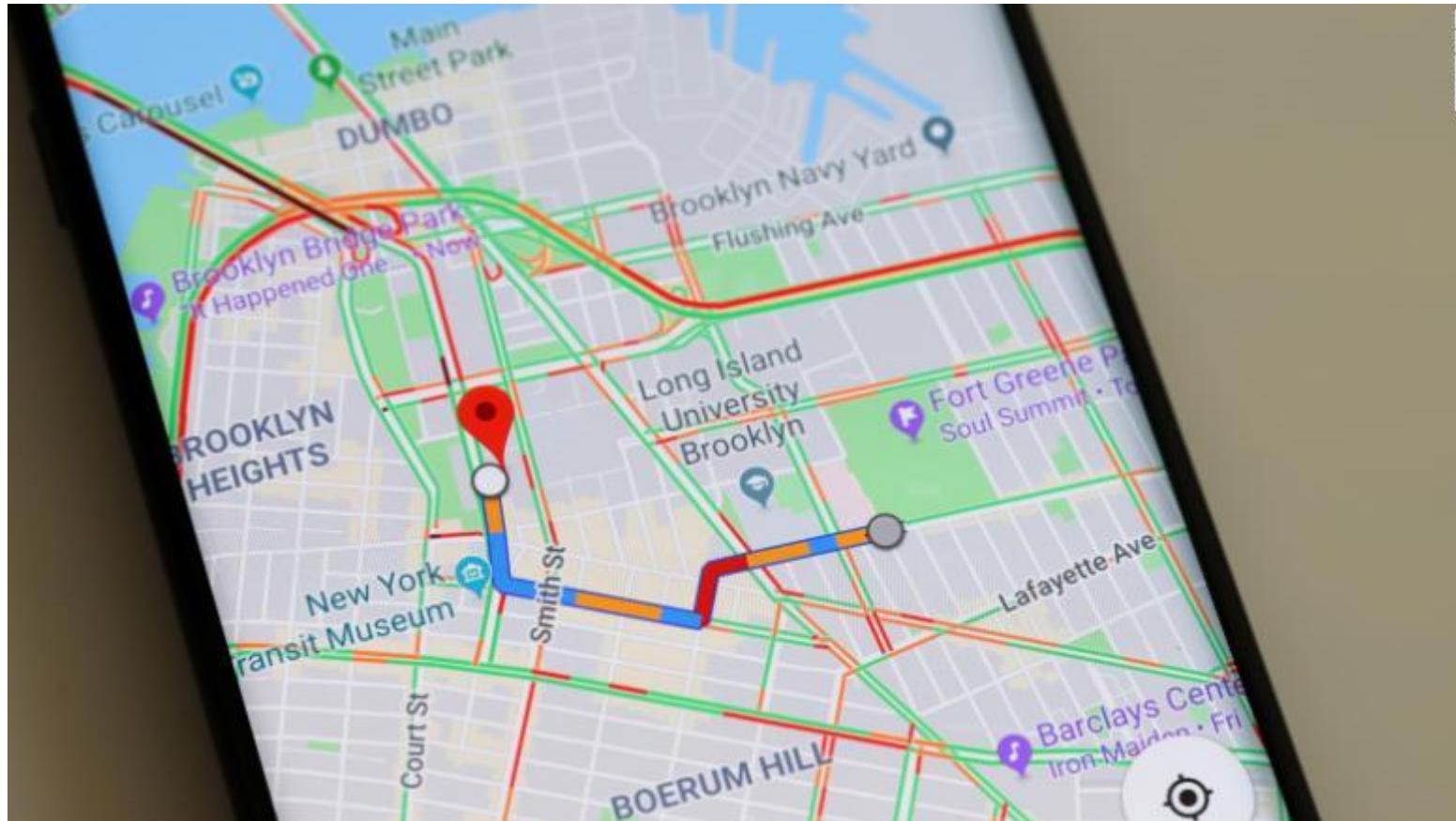
3D Maps



Apple Maps

Slide credit:2022 Richard Szeliski, The University of Washington

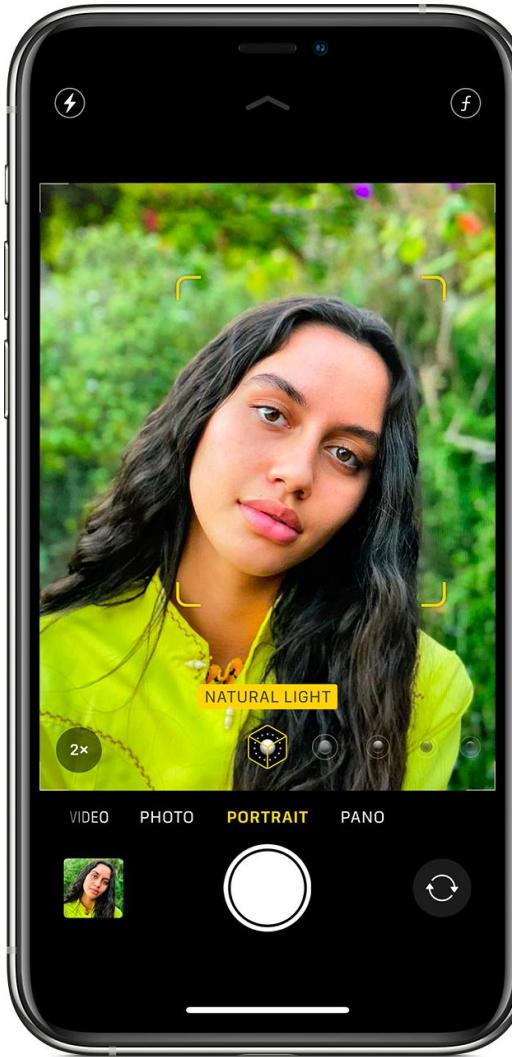
2D Maps



Google Maps

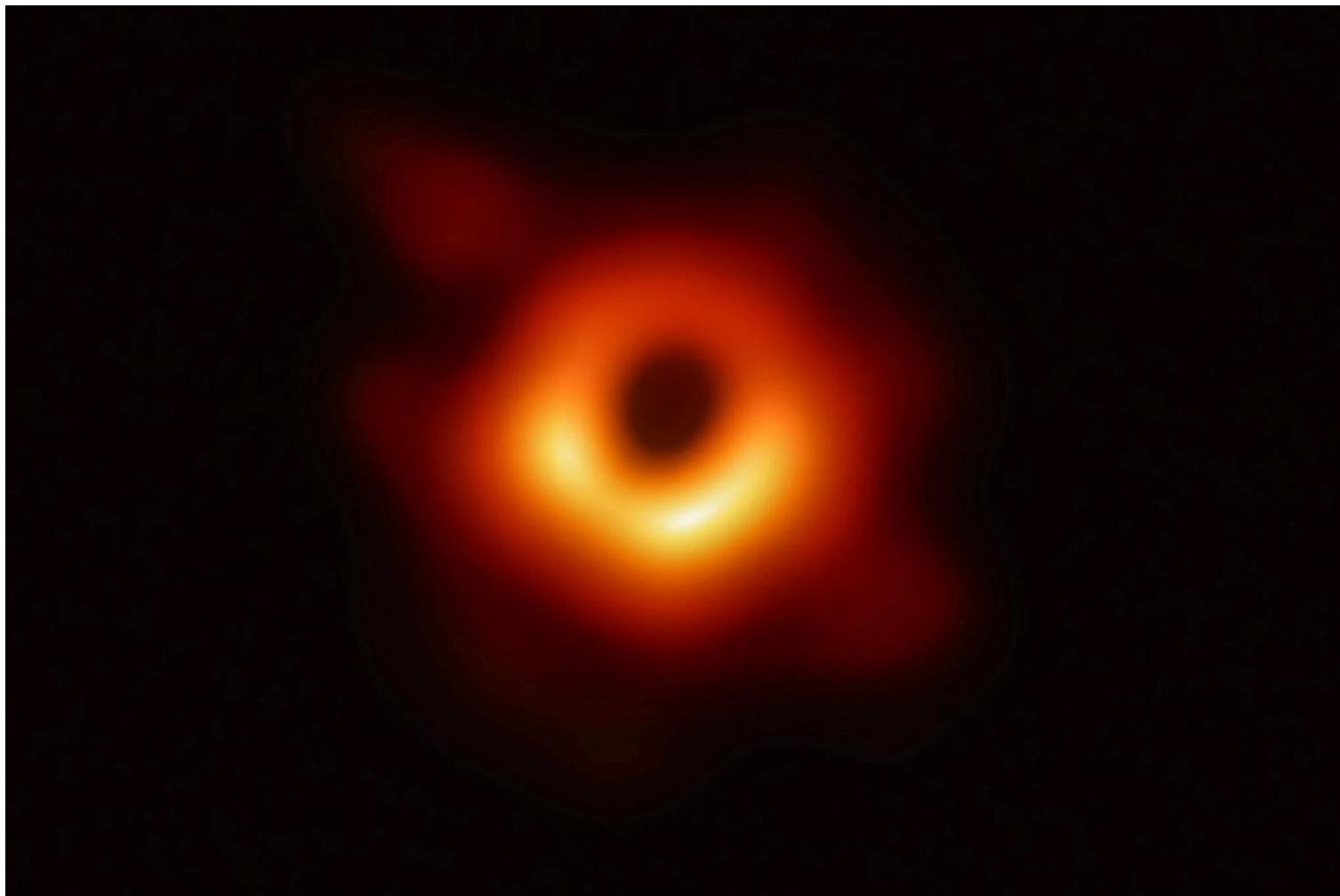
Slide credit: 2022 Richard Szeliski, The University of Washington

Computational photography



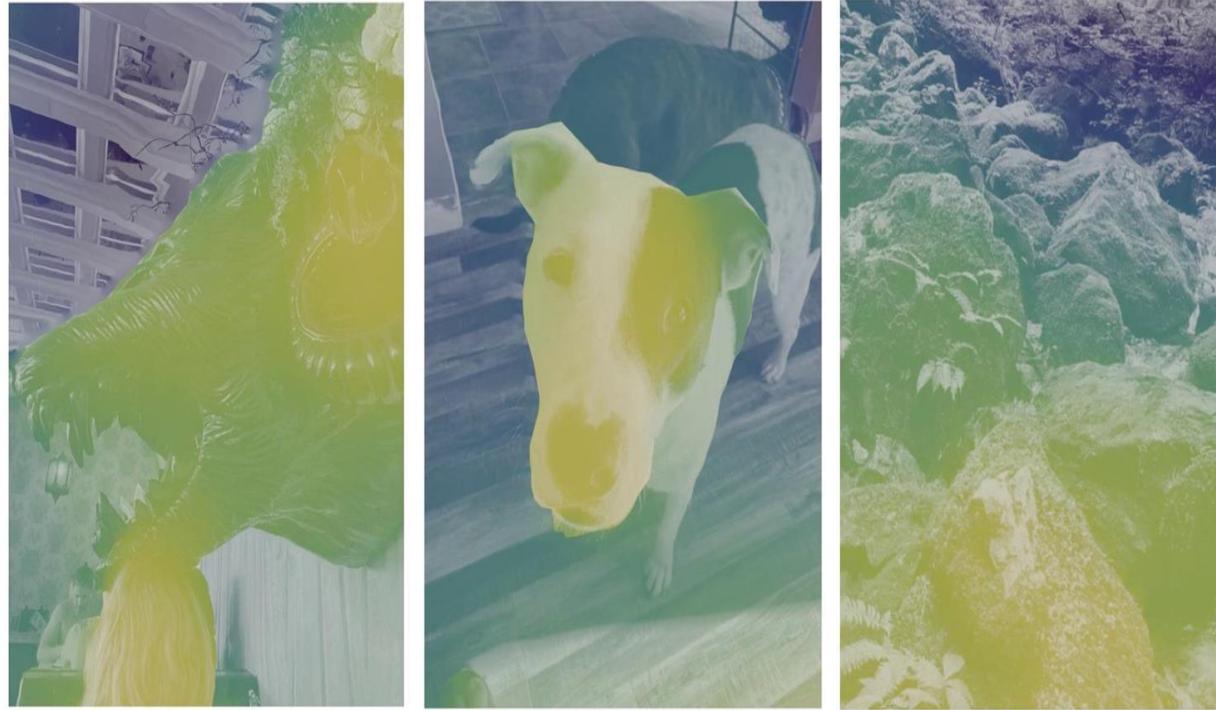
Portrait mode
simulating wider aperture

Even wider aperture...



[How scientists captured the first image of a black hole, 2019](#)

3D photos



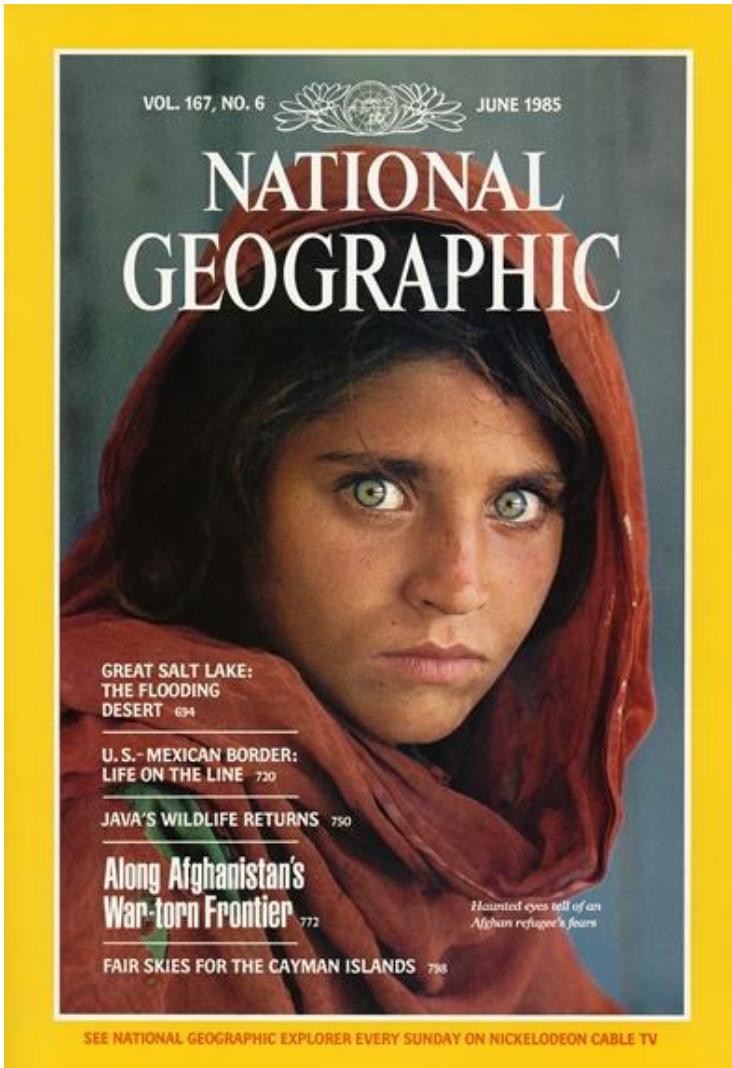
3D Photos on Facebook

Estimate depth from photo to create animation

<https://ai.facebook.com/blog/-powered-by-ai-turning-any-2d-photo-into-3d-using-convolutional-neural-nets/>

Slide credit: 2022 Richard Szeliski, The University of Washington

Face recognition



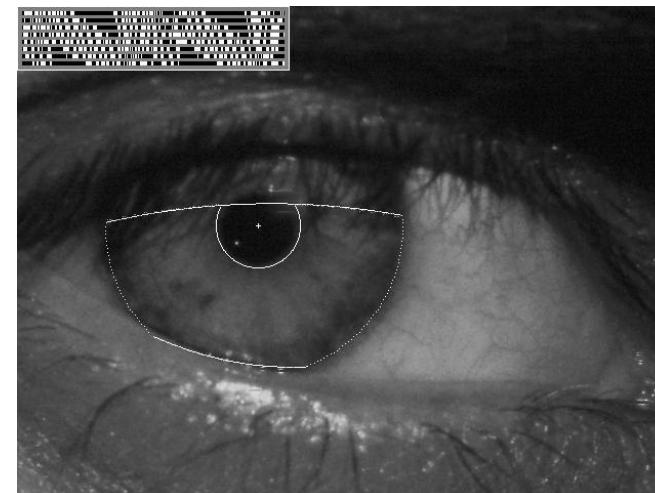
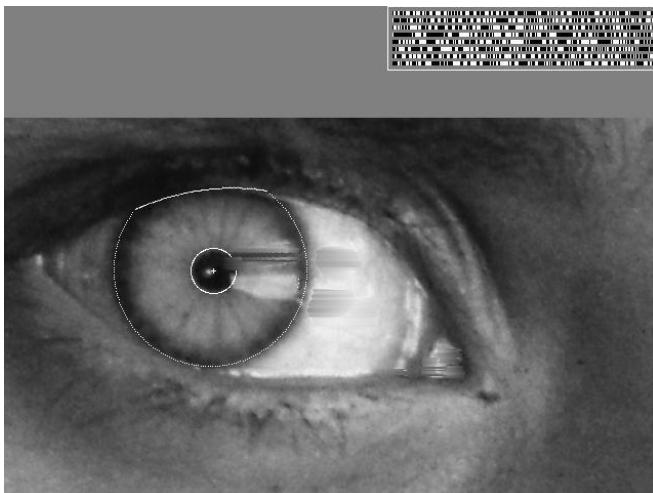
Who is she?

Slide credit:2022 Richard Szeliski, The University of Washington

Vision-based biometrics



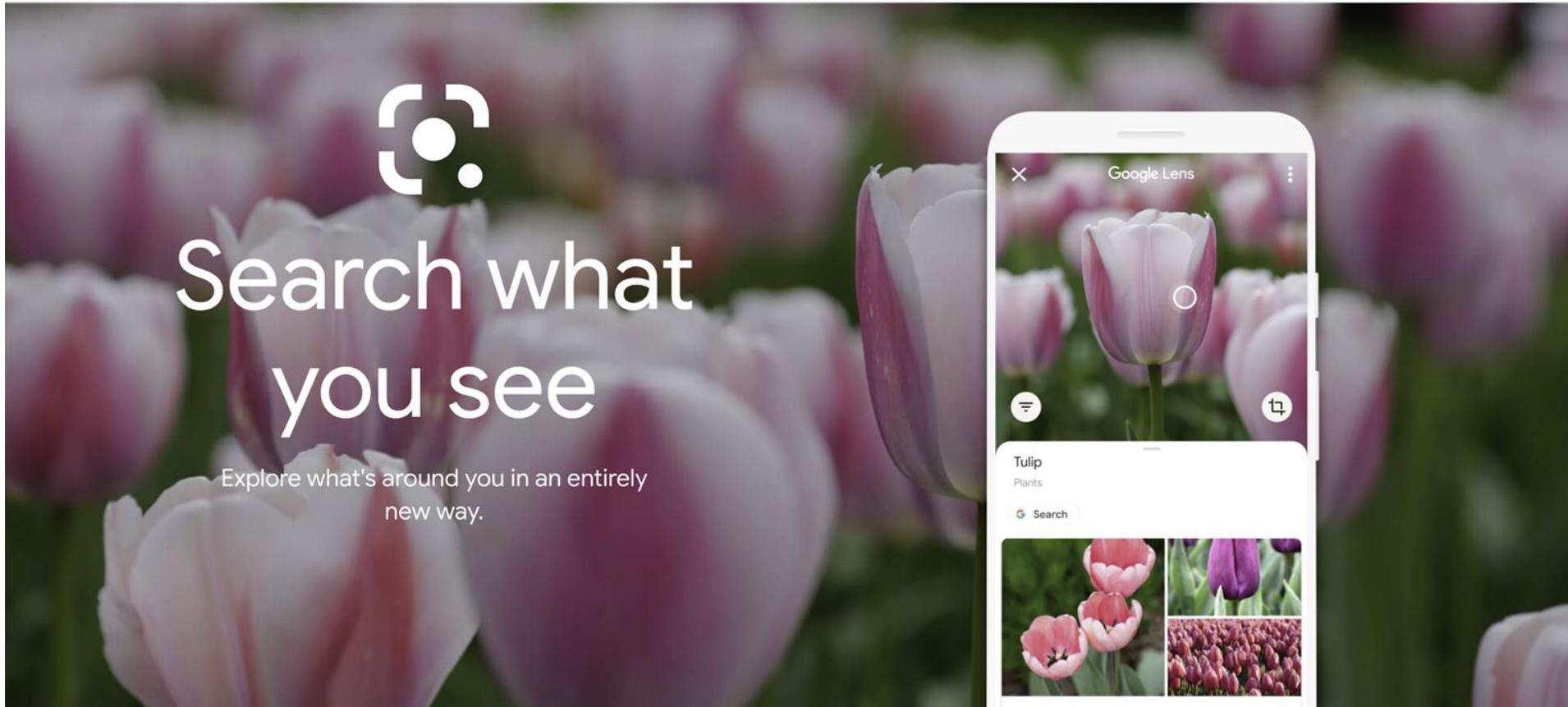
“How the Afghan Girl was Identified by Her Iris Patterns” Read the [story](#)



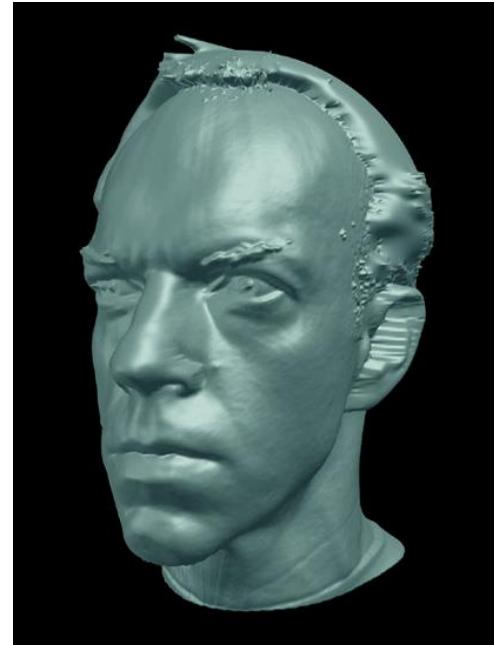
Object recognition

Google Lens

Download



Special effects: shape capture



The Matrix movies, ESC Entertainment, XYZRGB, NRC

Sports



Sportvision first down line
Nice [explanation](#) on www.howstuffworks.com

Games



Microsoft's XBox Kinect

Virtual Reality



Oculus Quest, Beat Saber

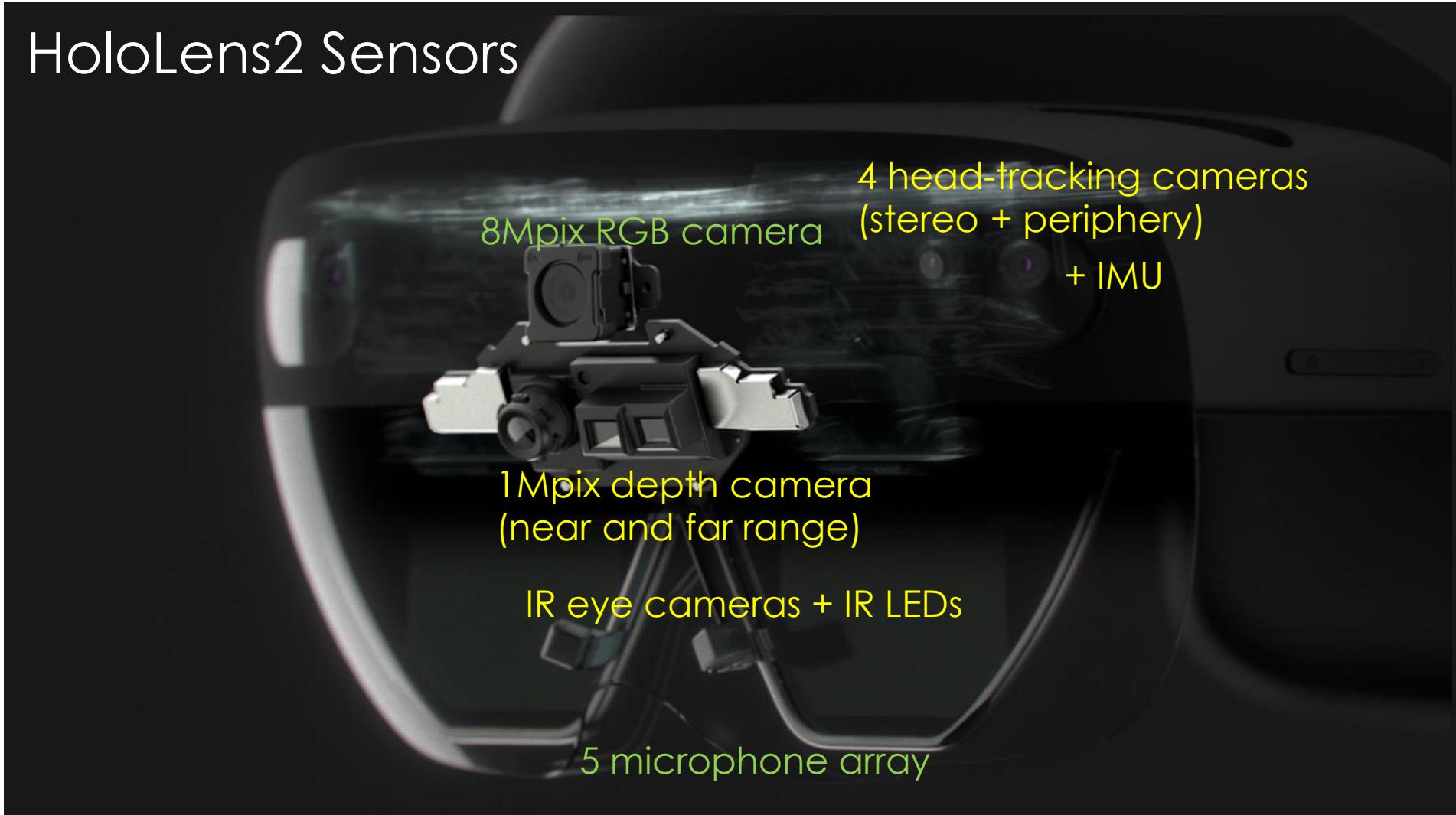
Slide credit:2022 Richard Szeliski, The University of Washington

Augmented Reality

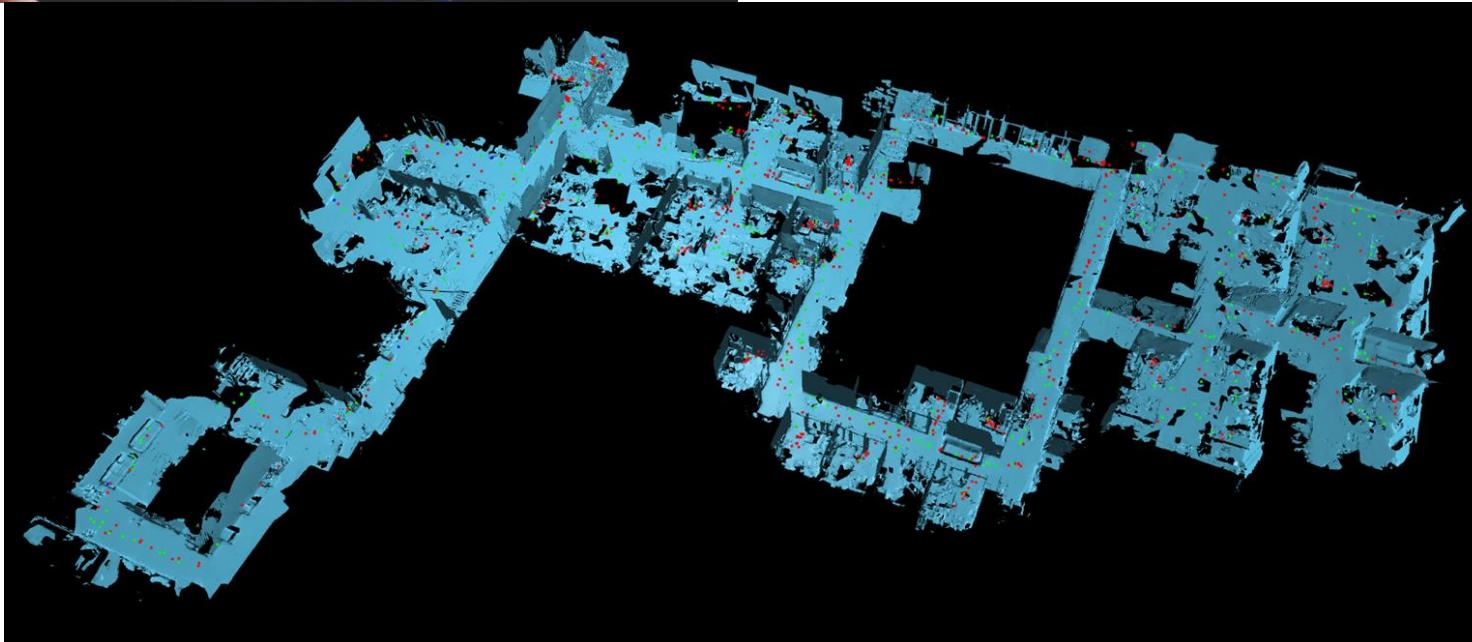
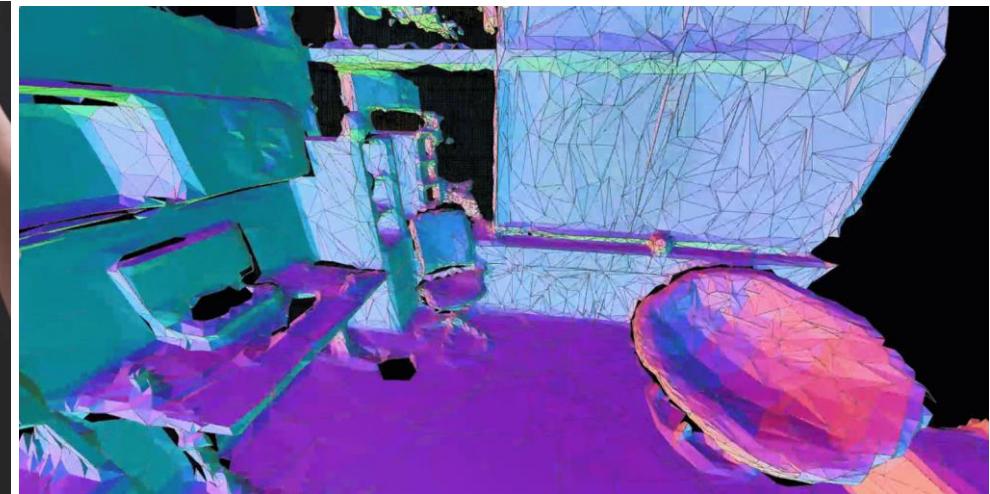


Microsoft Hololens 2

HoloLens2 Sensors



Augmented Reality



Slide credit:2022 Richard Szeliski, The University of Washington

Phone-based AR



<https://www.youtube.com/watch?v=0Pi-jzy6ESE>

Slide credit: 2022 Richard Szeliski, The University of Washington

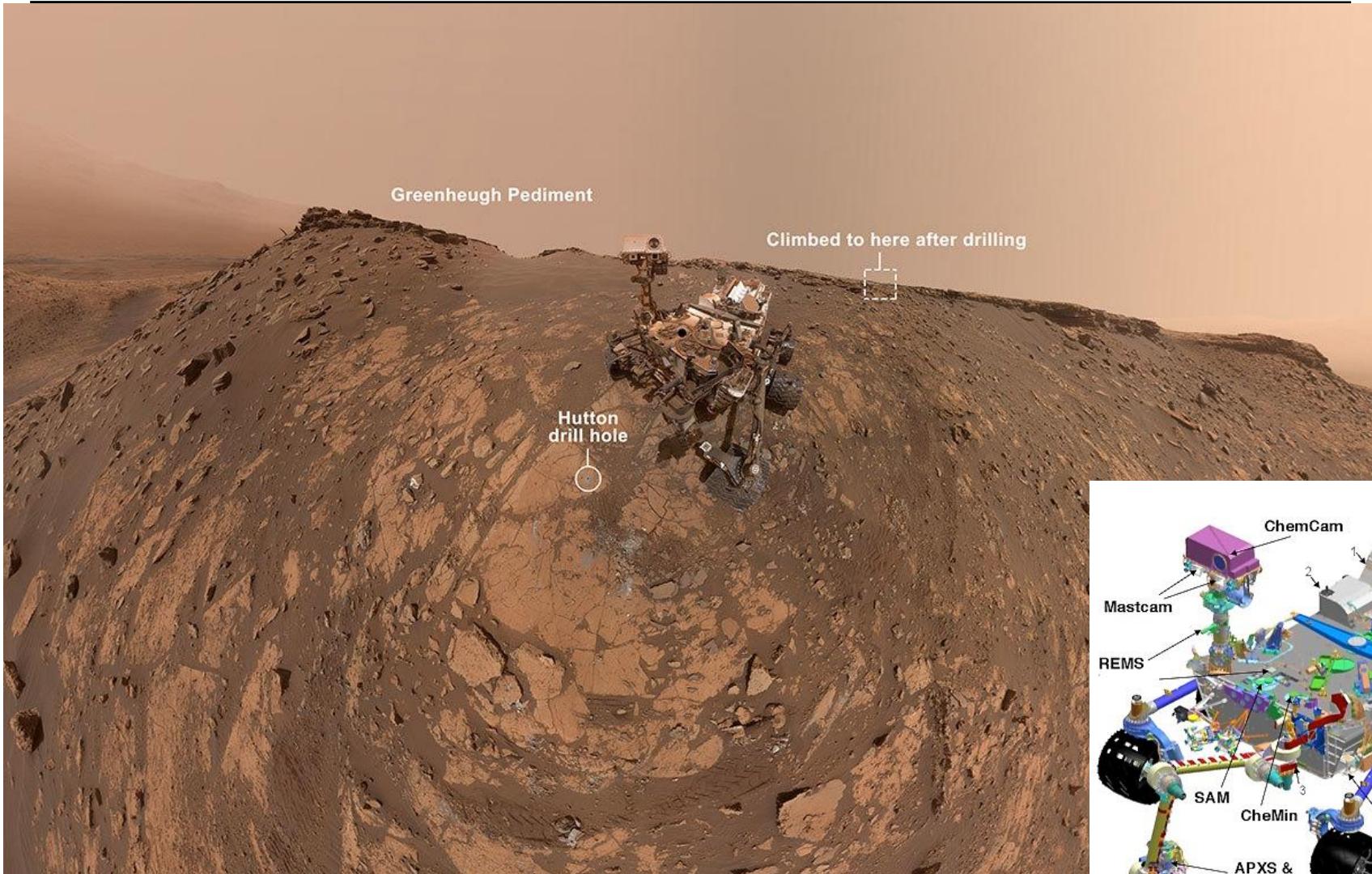
Body Tracking



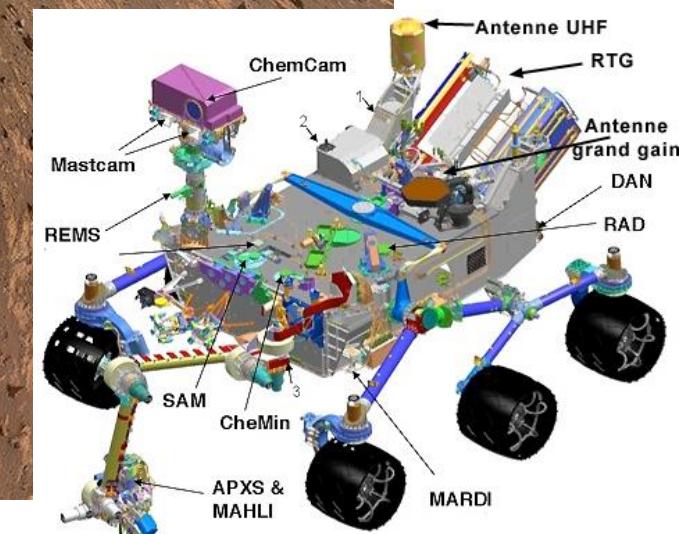
Magic Eraser – Background Remover (Google Pixel)



Robotics



NASA's Mars Curiosity Rover (self portrait)
[https://en.wikipedia.org/wiki/Curiosity_\(rover\)](https://en.wikipedia.org/wiki/Curiosity_(rover))



Smart cars

Slide content courtesy of Amnon Shashua

The screenshot shows the Mobileye website homepage. At the top, there are two tabs: "manufacturer products" (highlighted in blue) and "consumer products". Below the tabs, the slogan "Our Vision. Your Safety." is displayed above an overhead view of a silver car. Three cameras are illustrated: a "rear looking camera" at the back, a "forward looking camera" at the front, and a "side looking camera" on the left side. In the bottom section, there are three main features: "EyeQ Vision on a Chip" (with an image of a chip), "Vision Applications" (with an image of a pedestrian crossing a street), and "AWS Advance Warning System" (with an image of a dashboard display). To the right, there are two columns: "News" (listing articles like "Mobileye Advanced Technologies Power Volvo Cars World First Collision Warning With Auto Brake System" and "Volvo: New Collision Warning with Auto Brake Helps Prevent Rear-end") and "Events" (listing events like "Mobileye at Equip Auto, Paris, France" and "Mobileye at SEMA, Las Vegas, NV").

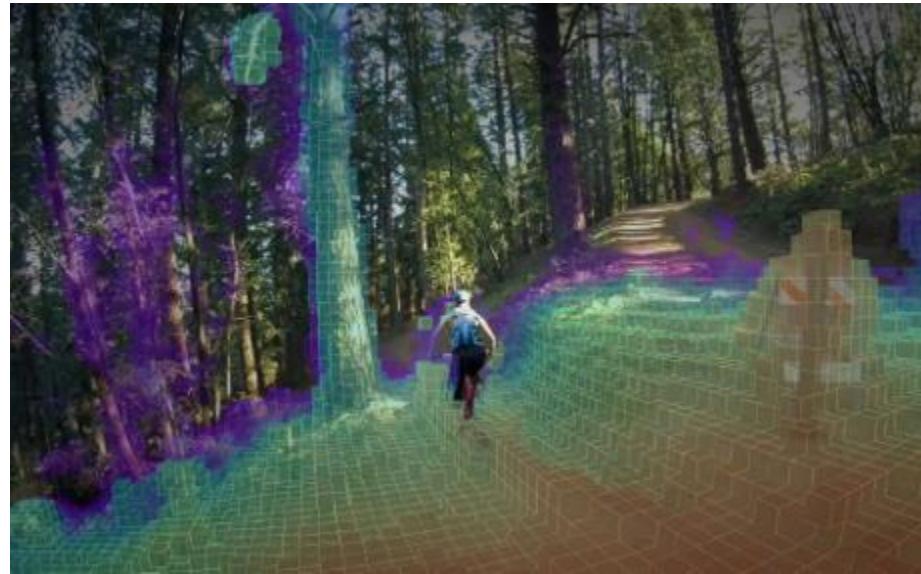
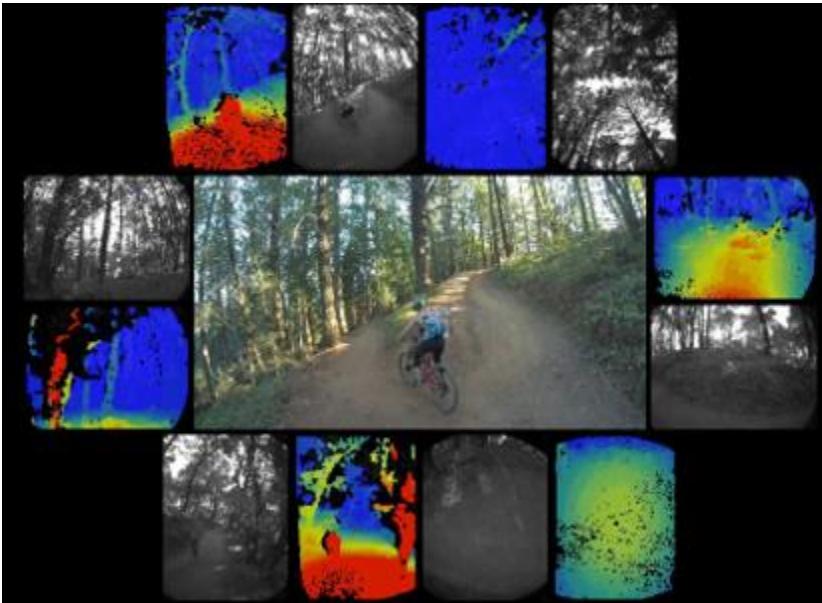
Mobileye

- Vision systems currently in high-end BMW, GM, Volvo models

Self-driving cars

<https://waymo.com/tech/>

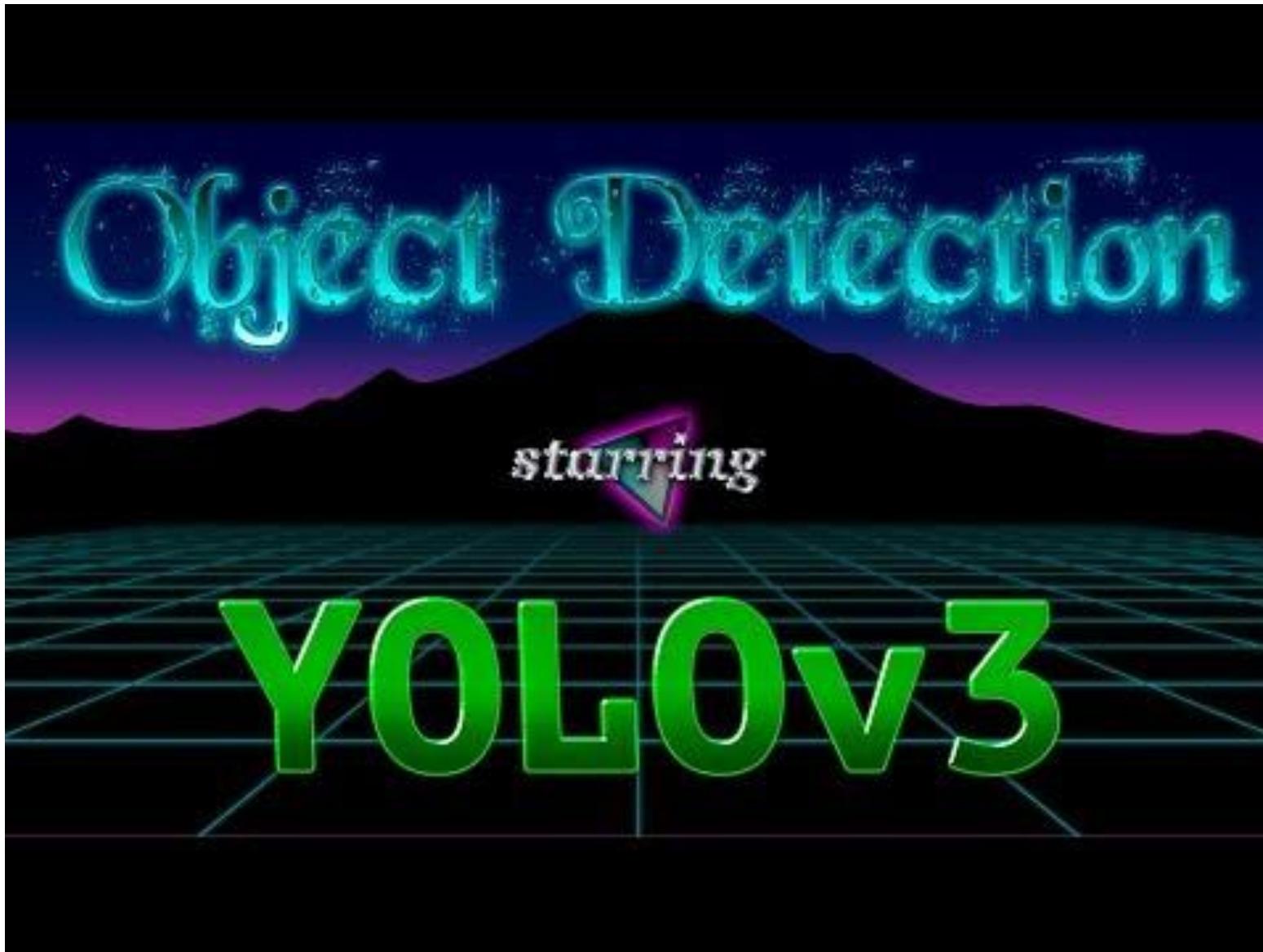
Drones



<https://www.skydio.com/>

SAMPLE COMPUTER VISION TASKS INVOLVED IN THESE APPLICATIONS

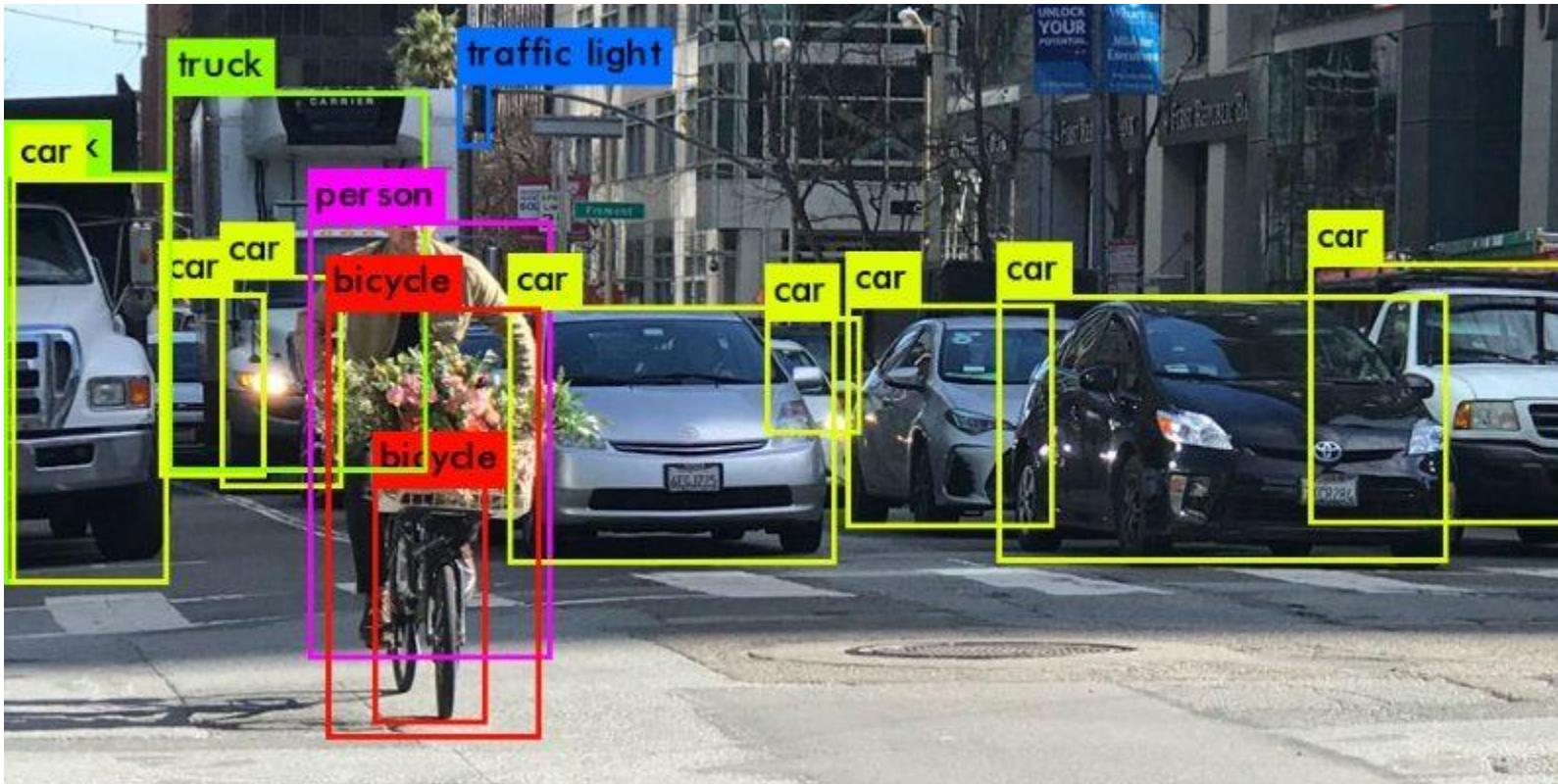
Research: Yolo



Research: StyleGan



Object Detection



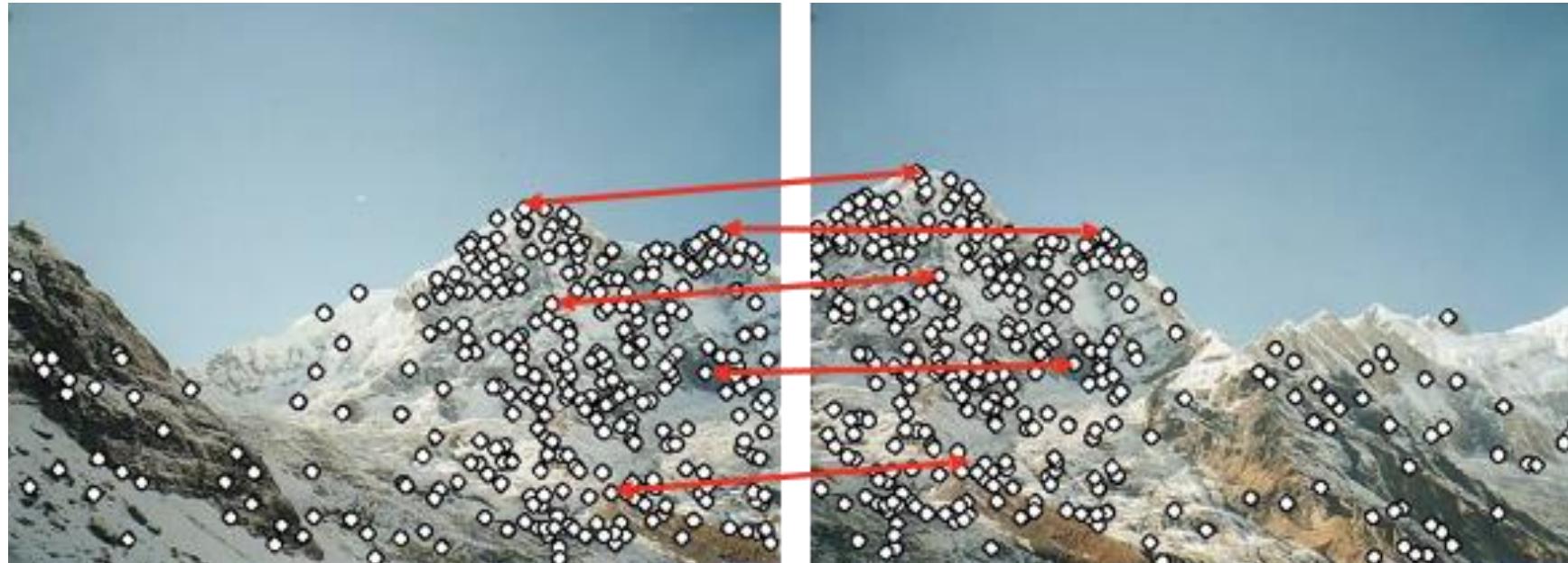
<https://heartbeat.fritz.ai/introduction-to-basic-object-detection-algorithms-b77295a95a63>

Segmentation

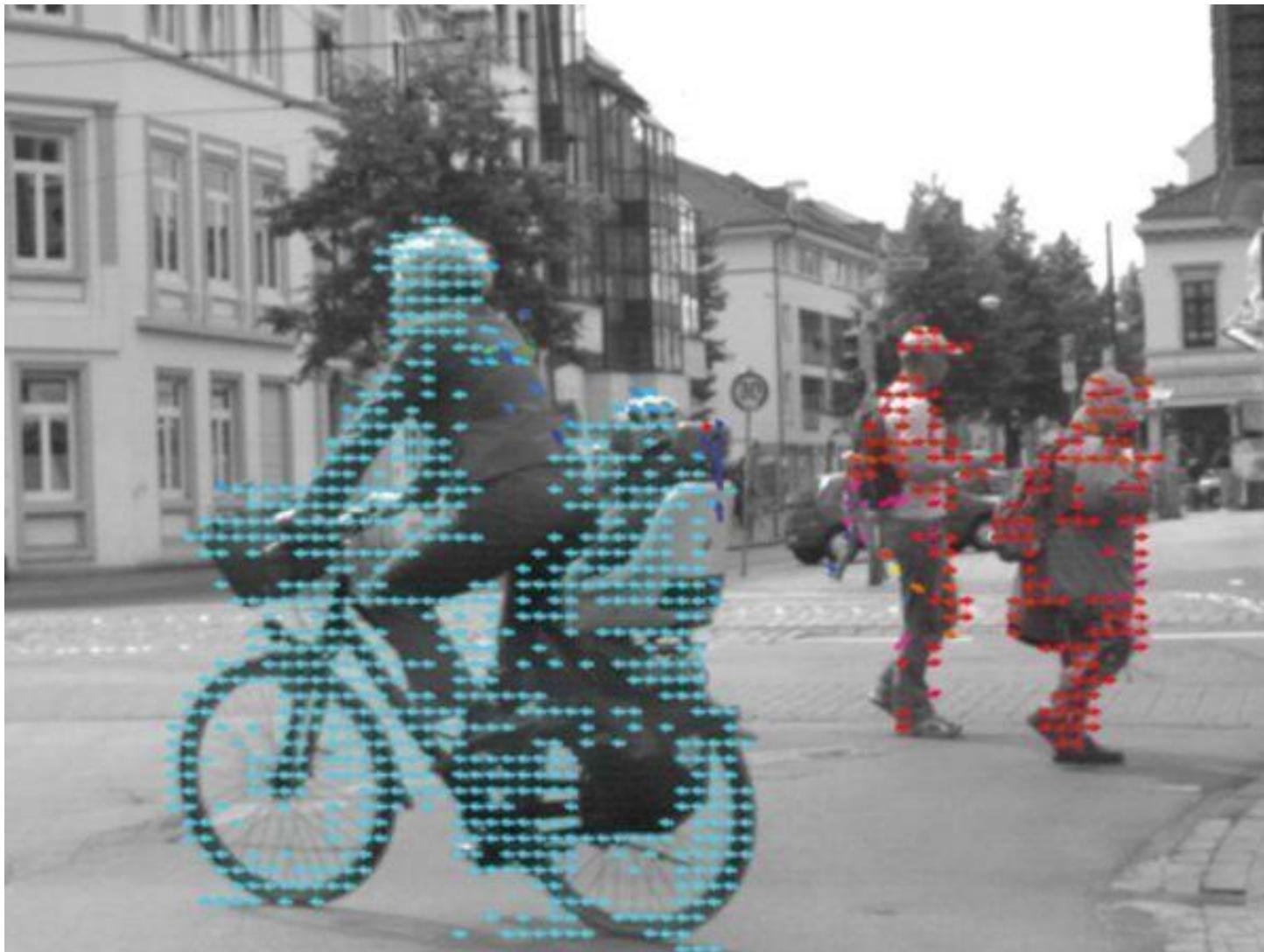


<https://gts.ai/how-do-we-solve-the-challenges-faced-due-to-semantic-segmentation/>

Features



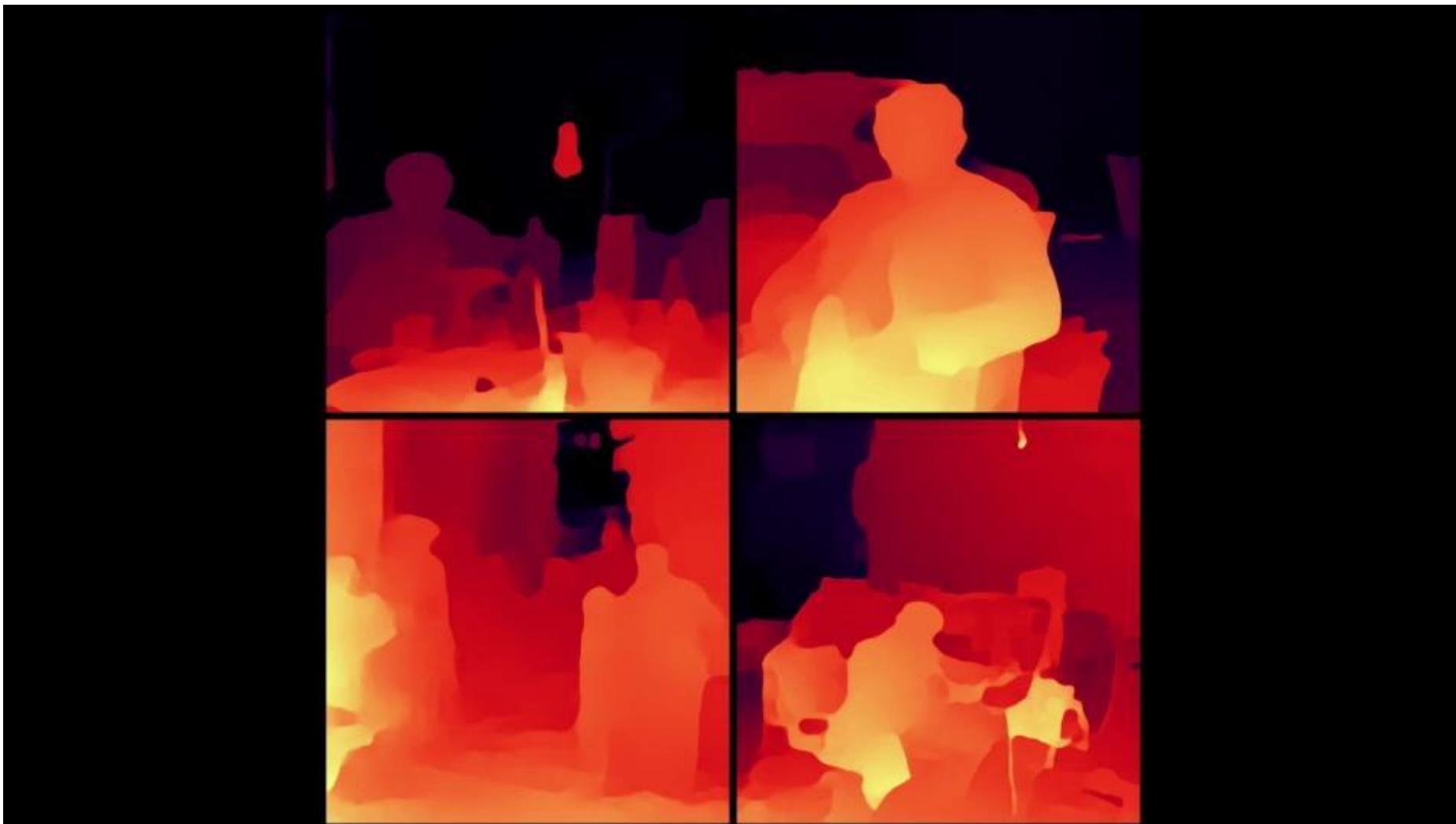
Optical Flow



<https://www.commonlounge.com/discussion/1c2eaa85265f47a3a0a8ff1ac5fbce51>

Slide credit: 2022 Richard Szeliski, The University of Washington

Stereo and Depth



3D Mapping (SLAM and SfM)



Top Trends in Computer Vision

Insights from CVPR 2022

October 2022



Introduction

Computer vision remains a dynamic and evolving field. Technological advances introduce new opportunities and efficiencies, and they are met with challenges in the form of new theoretical and societal considerations.

From privacy and algorithmic fairness to the feasibility of wide-scale adoption, we have entered one of the most exciting eras in computer vision.

Computer Vision Market *In Focus*

\$12.14 billion USD

Market size in 2022

7.0%

Compound Annual Growth Rate (CAGR)

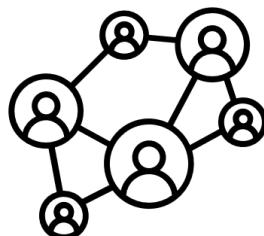
\$20.88 billion USD

Market size by 2030

Source: [Grand View Research](#)

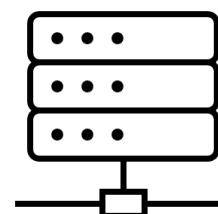
Environmental Factors Shaping Computer Vision

Increasing Industry Demand



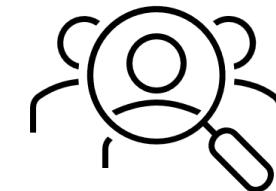
Industries ranging from finance and healthcare to retail and security and beyond are exploring how computer vision supports their emerging needs. Such emphasis means research continues to focus on ways to access and manipulate data in strategic, efficient, and highly accurate new ways.

Data Accessibility



The quality and integrity of data remains pivotal to results. Computer vision researchers are exploring how to achieve highly accurate results with smaller data sets, as well as with new techniques. In addition, more emphasis has been placed on opportunities with synthetic data to expand the use cases, availability, and address security issues around data sets.

Data Privacy and Bias



As computer vision techniques progress, how the data is used becomes a chief consideration. Advanced algorithms create unparalleled results, but personal security, bias, and societal factors come into play. Continued work will focus on the ethics surrounding these achievements.

Top Trends in Computer Vision

Based on these developments, today's computer vision landscape continues to evolve.

What follows is a summary of key observations, developments, and considerations for the year ahead, informed by insights from CVPR 2022.



Autonomous vehicles are gaining traction.

"There was more push towards autonomous driving with solid applications. What will happen is that even if you don't have a 100% autonomous car, the process of trying to get there will add so many new features."

– Rama Chellappa, John Hopkins Univ., CVPR 2022 General Chair

53.6%

The compound annual growth rate (CAGR) at which the global autonomous vehicles market is expected to expand from 2022 to 2030, according to [Grand View Research](#).



The lines
between
computer vision
and computer
graphics
continue to
blur.

*"Half the papers in computer vision
look like computer graphics. Instead
of collecting data you can now
simulate and that is very powerful."*

– Rama Chellappa, John Hopkins
Univ., CVPR 2022 General Chair



50+ NeRF Papers were presented at CVPR 2022.



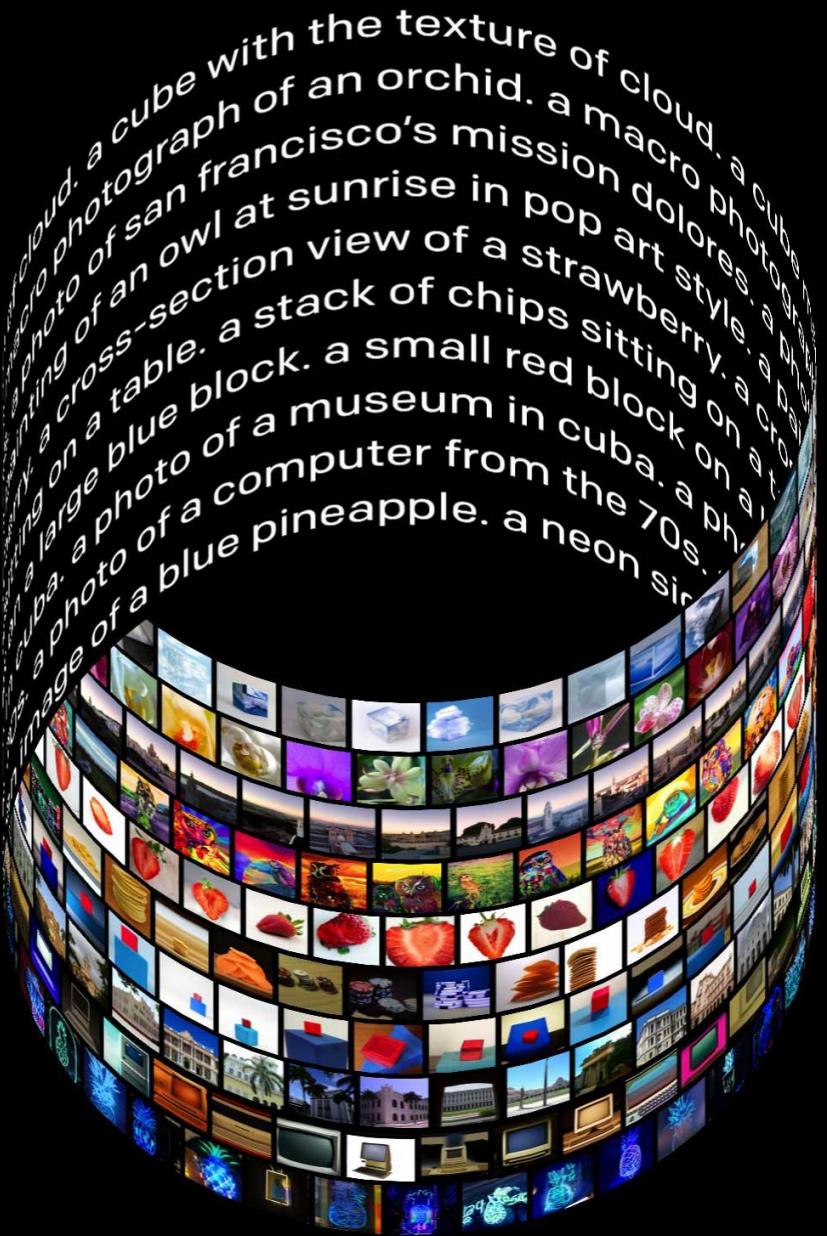
With the intersection of computer graphics and computer vision, NeRF research has risen in prominence.

"NeRF research is a hot focus right now. It continues to generate jaw-dropping images and is a beautiful blend of computer graphics and computer vision. Computer vision scientists think of cameras as scientific measuring devices that can do more than capture visually pleasing 2D images. These algorithms are a continuation of that. The cameras will be designed to get better computational photography, unifying computer graphics, computational photography, and computer vision."

-- Kristin Dana, Rutgers University, CVPR 2022 Program Chair



Source: [NeRF in the Dark, CVPR 2022](#)



Source: <https://openai.com/blog/dall-e/>

Content generation has become a burgeoning area of development.

"Another trend is content generation: DALL-E can now generate images out of open AI. It makes some computational sense that we should be able to do it. When we think and have a text description, our brains generate an image even though we haven't seen it, like when we read a book and generate an image in our heads. The algorithms are capturing that ability, and it's remarkable. But with these content generation algorithms comes the potential for bias, and we have our work ahead of us in considering how they can and should be used."

-- Kristin Dana, Rutgers University, CVPR 2022 Program Chair

There's a reemergence of focus on classic computer vision.

"The community is at a unique junction where while some papers focus on core technical research combining classical and modern deep networks, others focus on classical problems and innovative solutions."

-- Richa Singh, IIT Jodhpur, CVPR 2022
Program Chair





There's a general movement toward synthetic data.

"There's a tendency to move from real data to synthetic data if it is working, if it is effective. Cameras can only capture what has happened; whereas synthesis can imagine and produce whatever you wish. So, there is more variety in the synthetic data. And the privacy concerns are less."

– Rama Chellappa, John Hopkins Univ.,
CVPR 2022 General Chair

Facial recognition
research continues to
be the largest focus of
biometric work.

"The Computer Vision, Pattern Recognition, and Machine Learning community at large is focusing on developing ingenious algorithms not only for difficult scenarios, unconstrained environments, but also being trustworthy and dependable."

-- Richa Singh, IIT Jodhpur, CVPR
2022 Program Chair





What the Future Holds

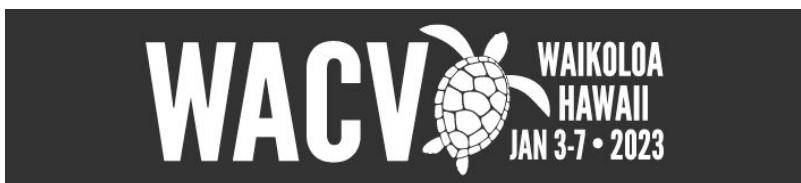
These trends summarize only a fraction of all that is occurring in computer vision today, and as baseball legend Yogi Berra famously said, “It's tough to make predictions, especially about the future.”

There's so much more to come as relates to computer vision, and with such a rapidly evolving space, only the next round of research will tell us what's next.

Stay connected with the Computer Society to see how the field continues to change and grow.

Stay abreast of the latest computer vision trends:

- IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2023
 - <https://wacv2023.thecvf.com/home>
- IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR) 2023
 - <https://cvpr2023.thecvf.com/>



Computer Vision CS/ECE 8690: Introduction to Image Processing Image Formation and Color

Slide References:

- Szeliski, “Computer Vision” Chapter 2
- Gonzalez & Woods, “Digital Image Processing” Chapter 2
 -

Instructor: Filiz Bunyak Ersoy bunyak@missouri.edu

TA: Imad Toubal (CS PhD): itdfh@mail.missouri.edu

Some Computer Vision Resources

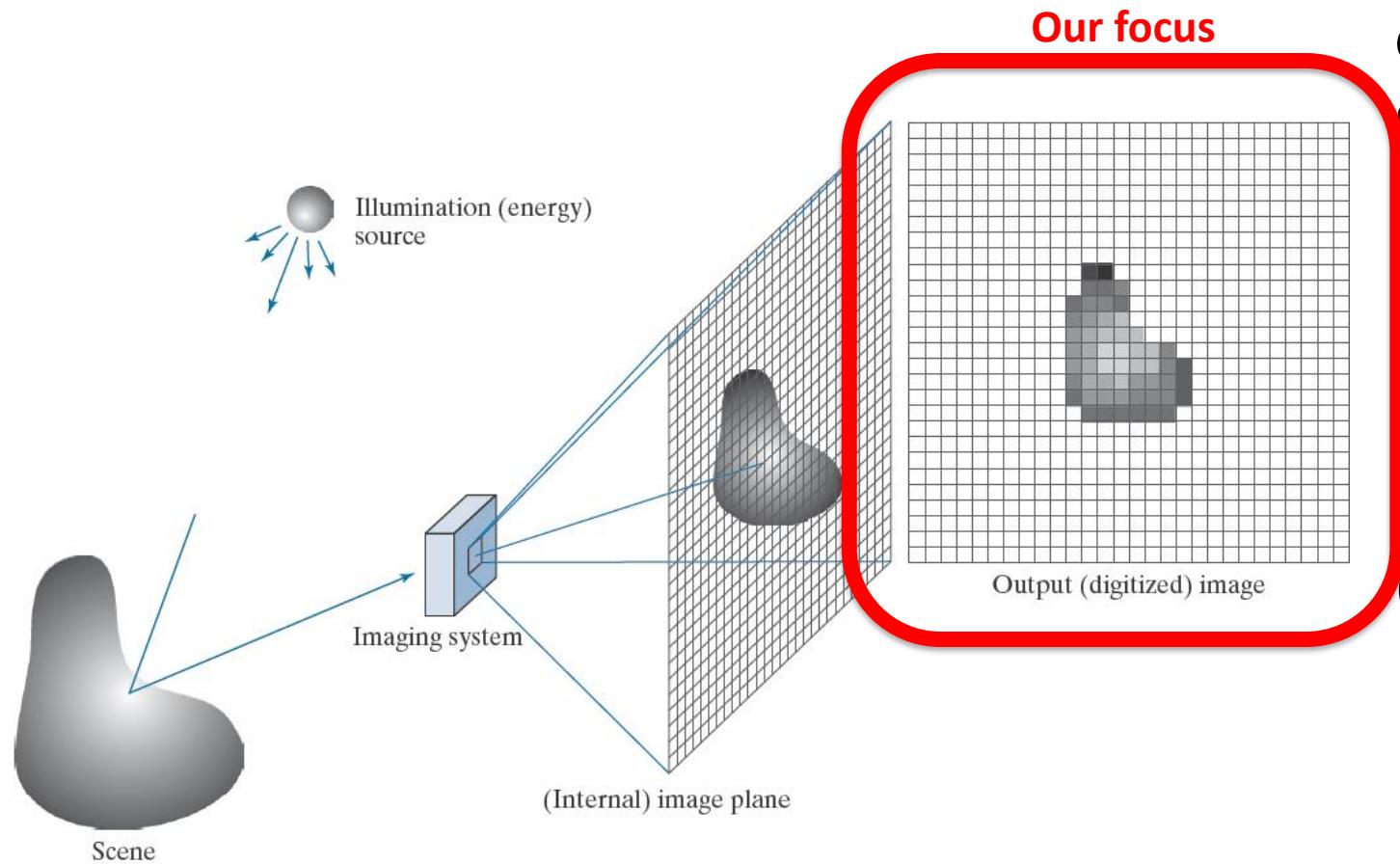
- Richard Szeliski's <https://szeliski.org/Book/>
- <https://www.imageprocessingplace.com/>
- Awesome-computer-vision (A curated list of awesome computer vision resources by Prof. Jia-Bin Huang, University of Maryland)
<https://github.com/jbhuang0604/awesome-computer-vision>
 - Books, Courses, Papers, Software, Datasets, Tutorials, Talks, Blogs, etc.
- Top computer vision conferences and papers:
 - [CVPR](#): IEEE Conference on Computer Vision and Pattern Recognition
 - [ICCV](#): International Conference on Computer Vision
 - [ECCV](#): European Conference on Computer Vision
 - [NIPS](#): Neural Information Processing Systems

Coding Resources

Coding: Python, numpy, scipy, Pytorch

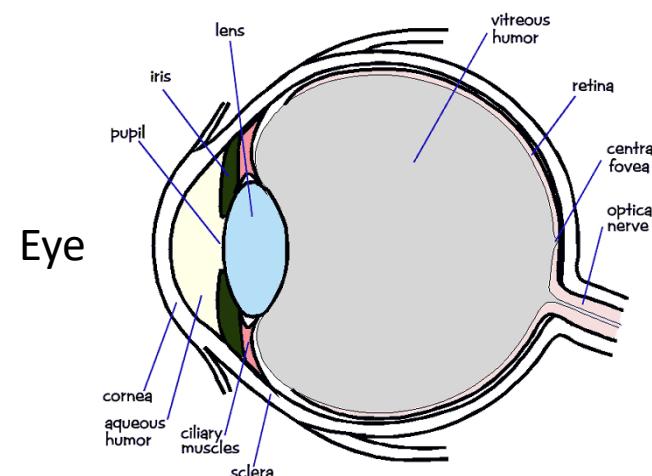
- Programming with Python:
 - <https://swcarpentry.github.io/python-novice-inflammation/setup.html>
 - [Analyzing Patient Data – Programming with Python \(swcarpentry.github.io\)](#)
 - <https://www.w3schools.com/python/>
- Python tutorial: <https://docs.python.org/3/tutorial/>
- Python Programming And Numerical Methods: A Guide For Engineers And Scientists: <https://pythonnumericalmethods.berkeley.edu/notebooks/Index.html>
- NumPy quickstart: <https://numpy.org/doc/stable/user/quickstart.html>
- <https://scikit-image.org/>
- Tutorial will be announced next week

Image Acquisition

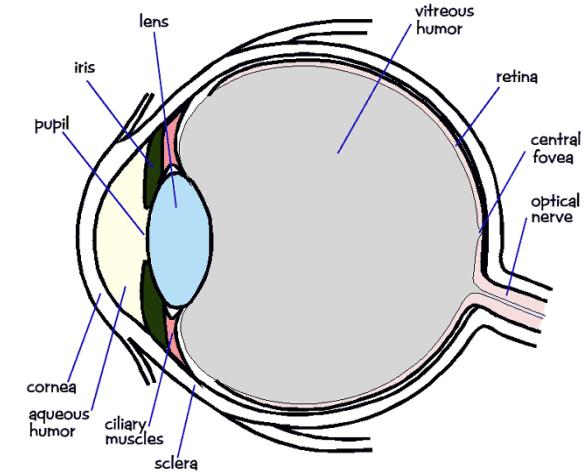
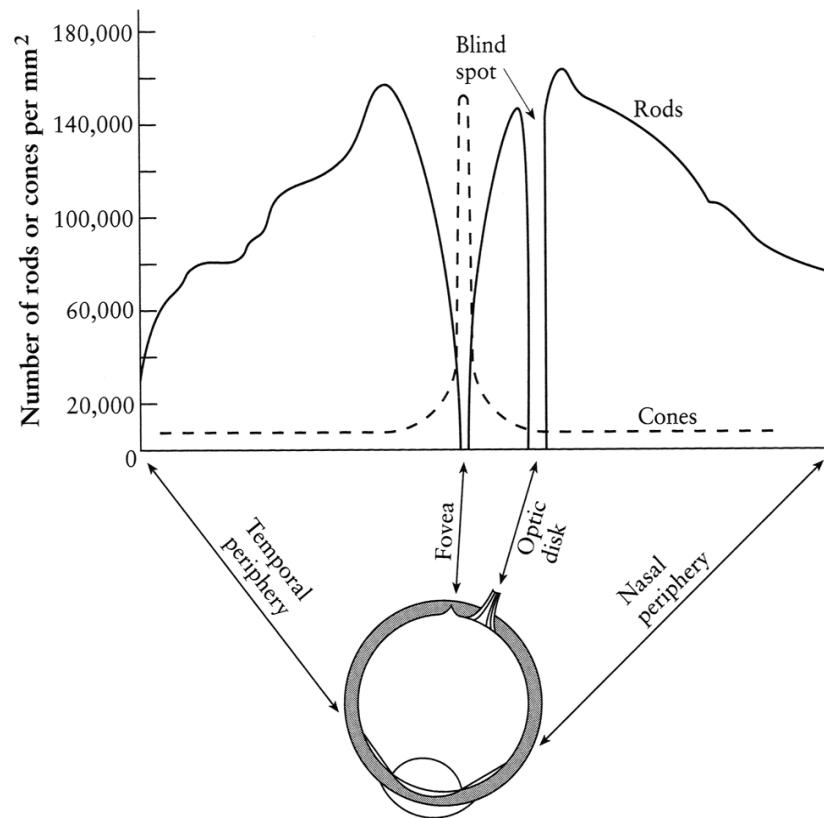


Gonzalez& Woods Figure 2.15 An example of digital image acquisition.

- Illumination (energy) source.
- A scene.
- Imaging system.
- Projection of the scene onto the image plane.
- Digitized image.



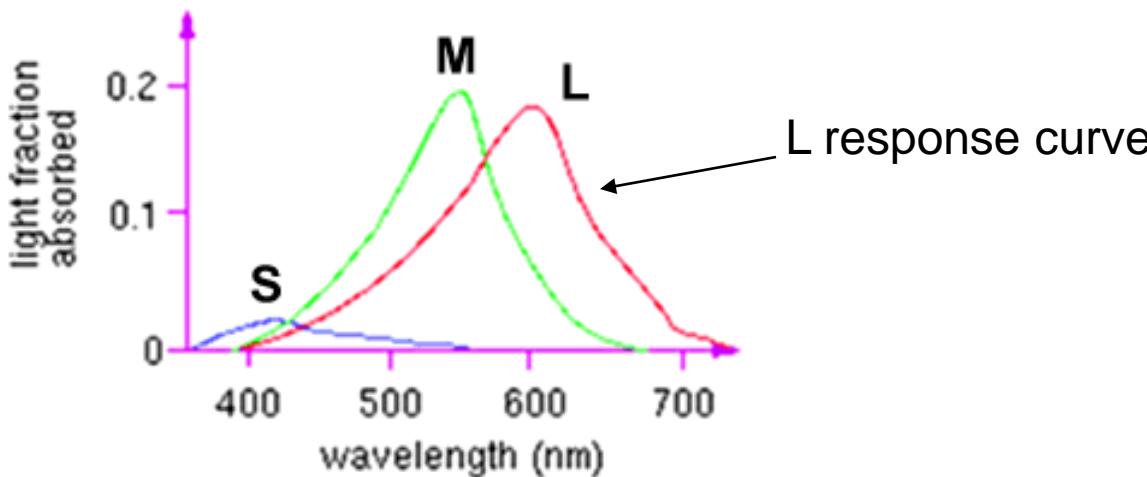
Perception: Density of rods and cones



Rods and cones are non-uniformly distributed on the retina

- **Rods** responsible for intensity,
- **Cones** responsible for color
- **Fovea** - Small region (1 or 2°) at the center of the visual field containing the highest density of **cones** (and no rods).
- Less visual acuity (low vision) in the periphery—many rods wired to the same neuron

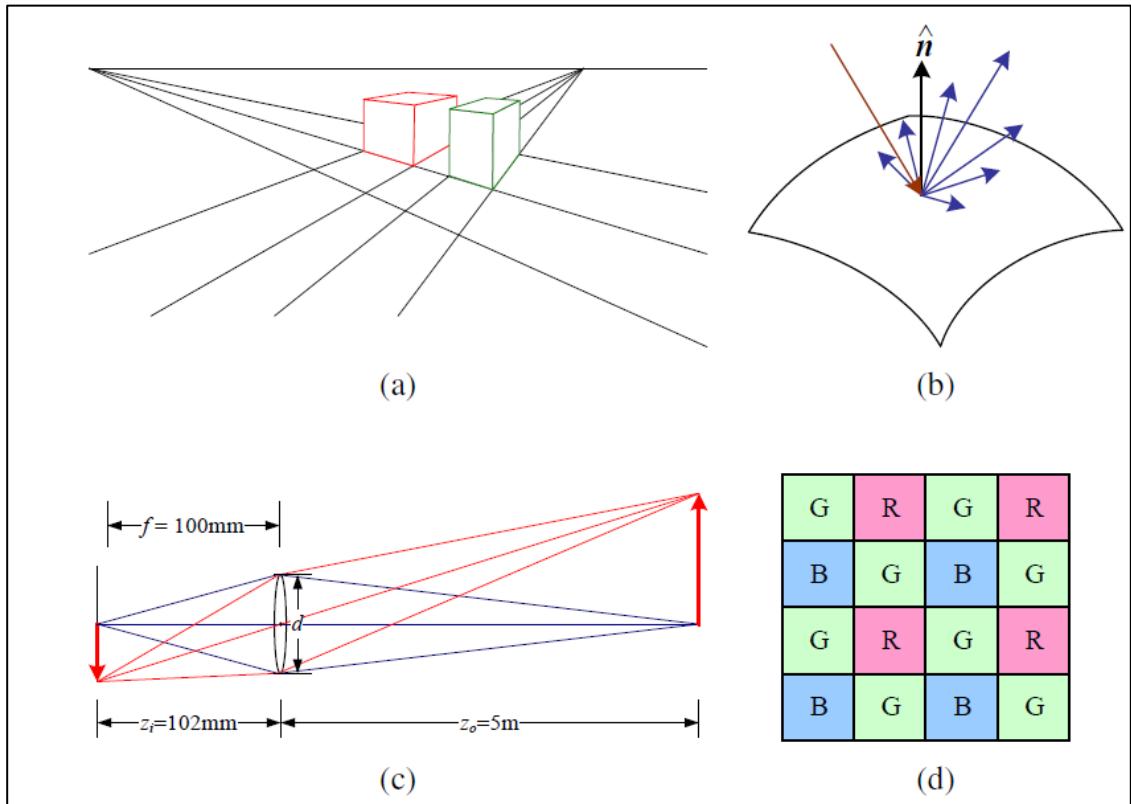
Color perception



Three types of cones

- Each is sensitive in a different region of the spectrum
 - but regions overlap
 - **Short (S)** corresponds to blue
 - **Medium (M)** corresponds to green
 - **Long (L)** corresponds to red
- Different sensitivities: we are more sensitive to green than red
 - varies from person to person (and with age)
- Colorblindness—deficiency in at least one type of cone

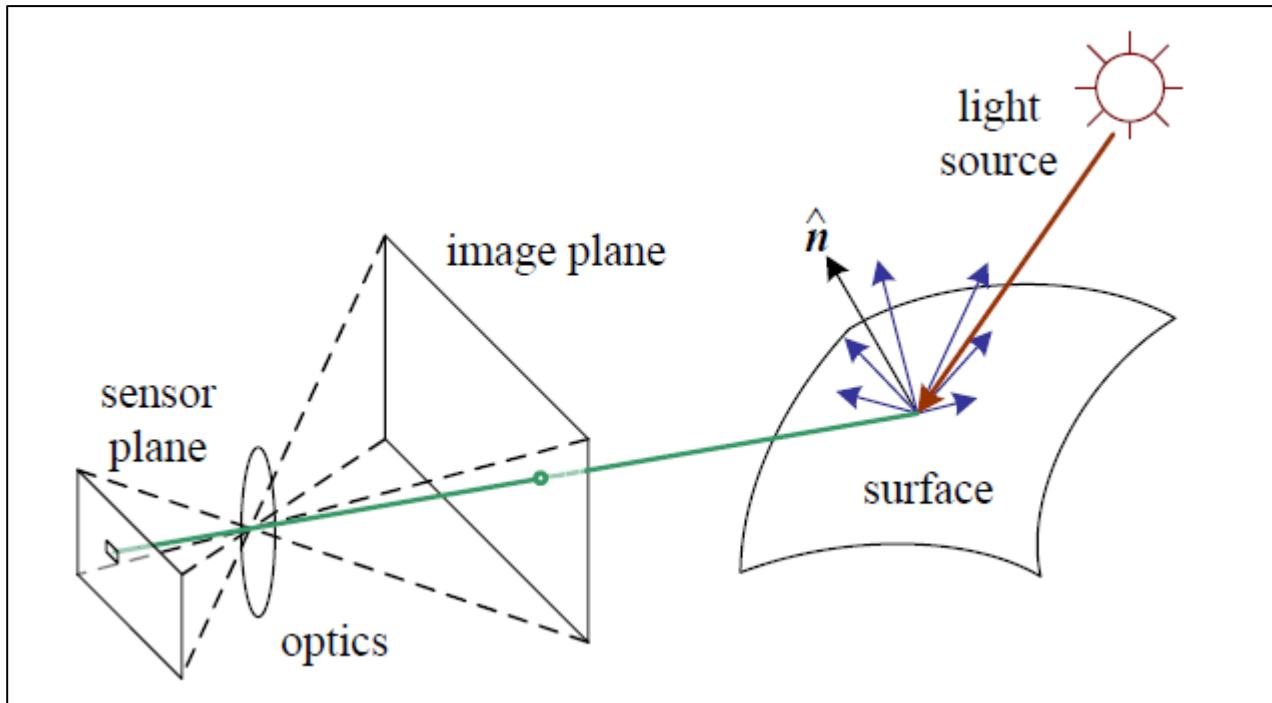
Image Formation



A few components of the image formation process:

- (a) perspective projection;
- (b) light scattering when hitting a surface;
- (c) lens optics;
- (d) Bayer color filter array.

Photometric Image Formation



A simplified model of photometric image formation.

Light is emitted by one or more light sources and is then reflected from an object's surface.

A portion of this light is directed towards the camera.

This simplified model ignores multiple reflections, which often occur in real-world scenes.

Photometric image formation

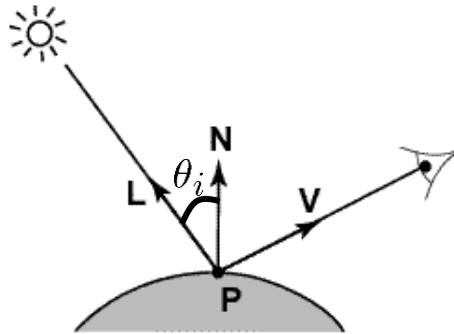
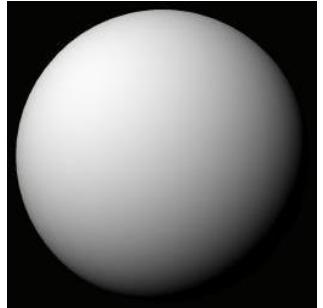
- when we discuss shape from shading

Lens optics + perspective geometry

- When we discuss multi-view geometry (stereo, 3D reconstruction etc.)

Shape from Shading

Application: Composite Image Detection



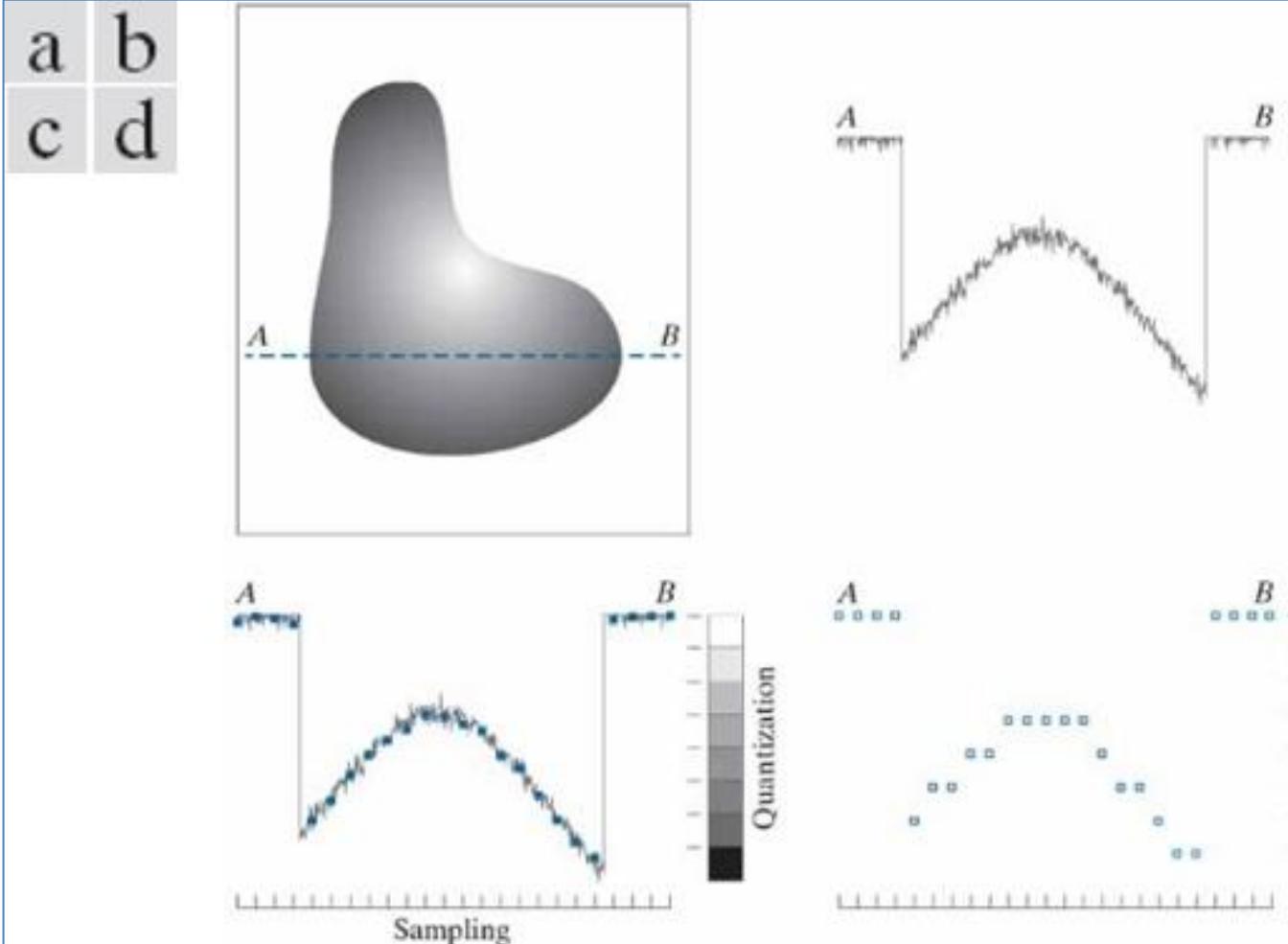
Fake photo



Real photo

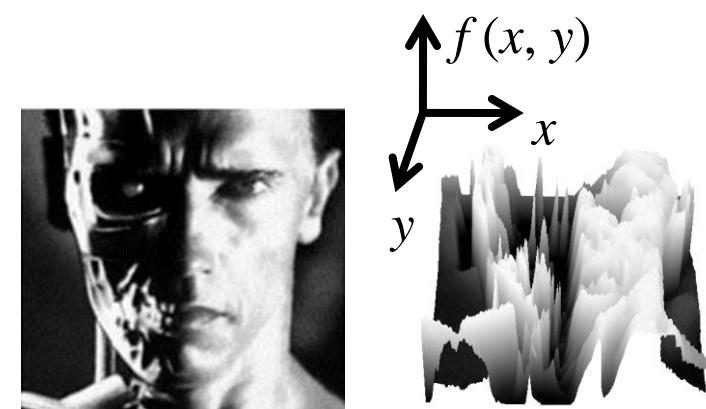


Sampling and Quantization



Gonzalez & Woods Figure 2.16

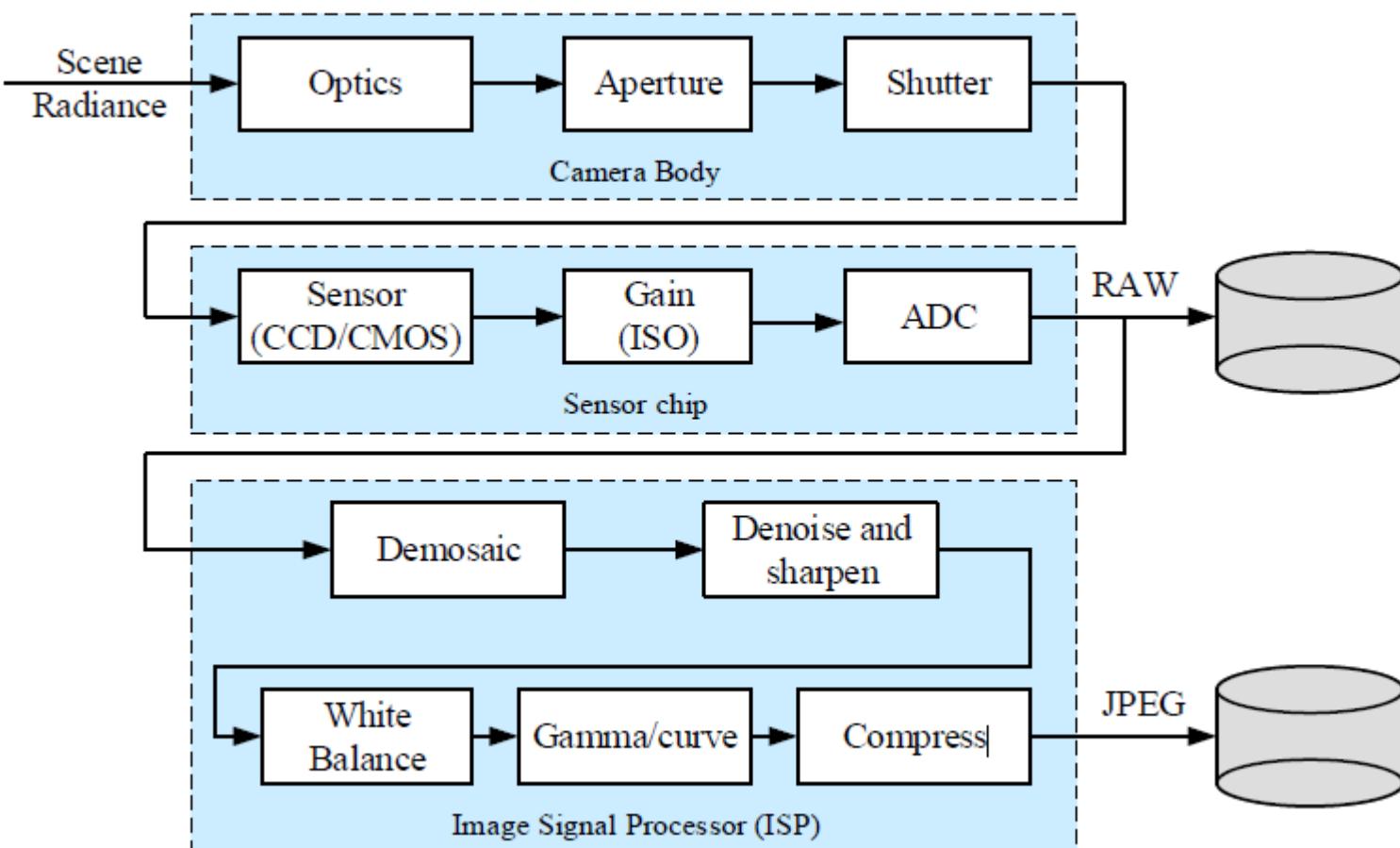
- (a) Continuous image.
- (b) A scan line showing intensity variations along line **AB** in the continuous image.
- (c) Sampling and quantization.
- (d) Digital scan line. (The black border in (a) is included for clarity. It is not part of the image).



A **digital** image is a discrete (**sampled, quantized**) version of this function

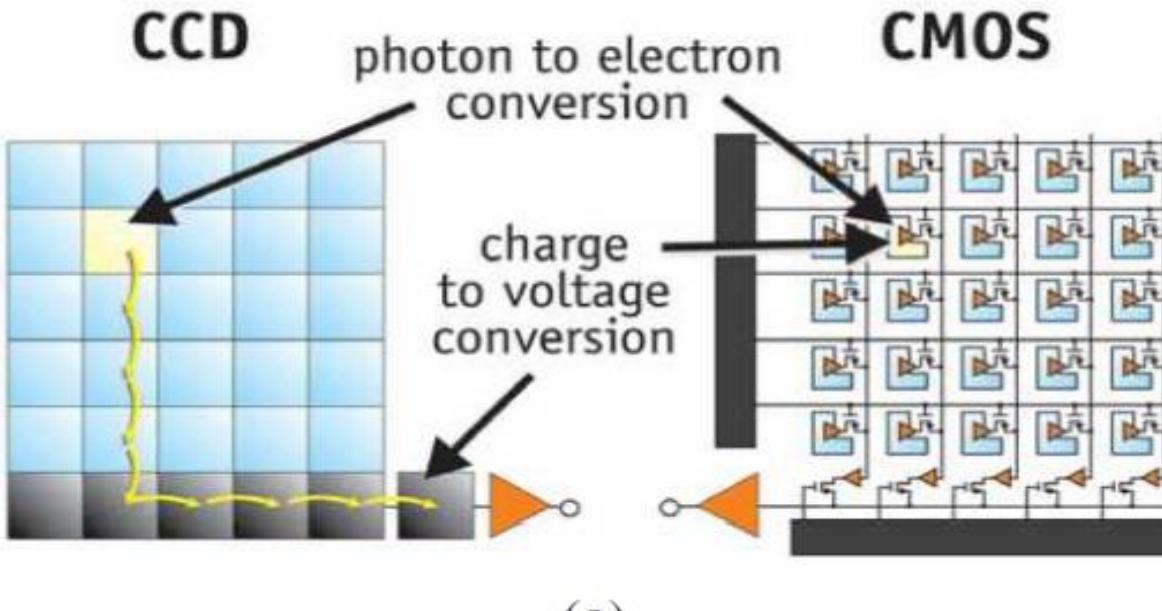
Digital Camera

Light falling on an imaging sensor is usually picked up by an active sensing area, integrated for the duration of the exposure, and passed to a set of sense amplifiers.



The main factors affecting the performance of a digital image sensor:

1. shutter speed,
2. sampling pitch,
3. fill factor,
4. chip size,
5. analog gain,
6. sensor noise,
7. resolution (and quality) of the analog-to-digital converter.



Digital imaging sensors:

- (a) CCDs move photogenerated charge from pixel to pixel and convert it to voltage at the output node;
- (b) CMOS imagers convert charge to voltage inside each pixel ([Litwiller 2005](#)) © 2005 Photonics Spectra;

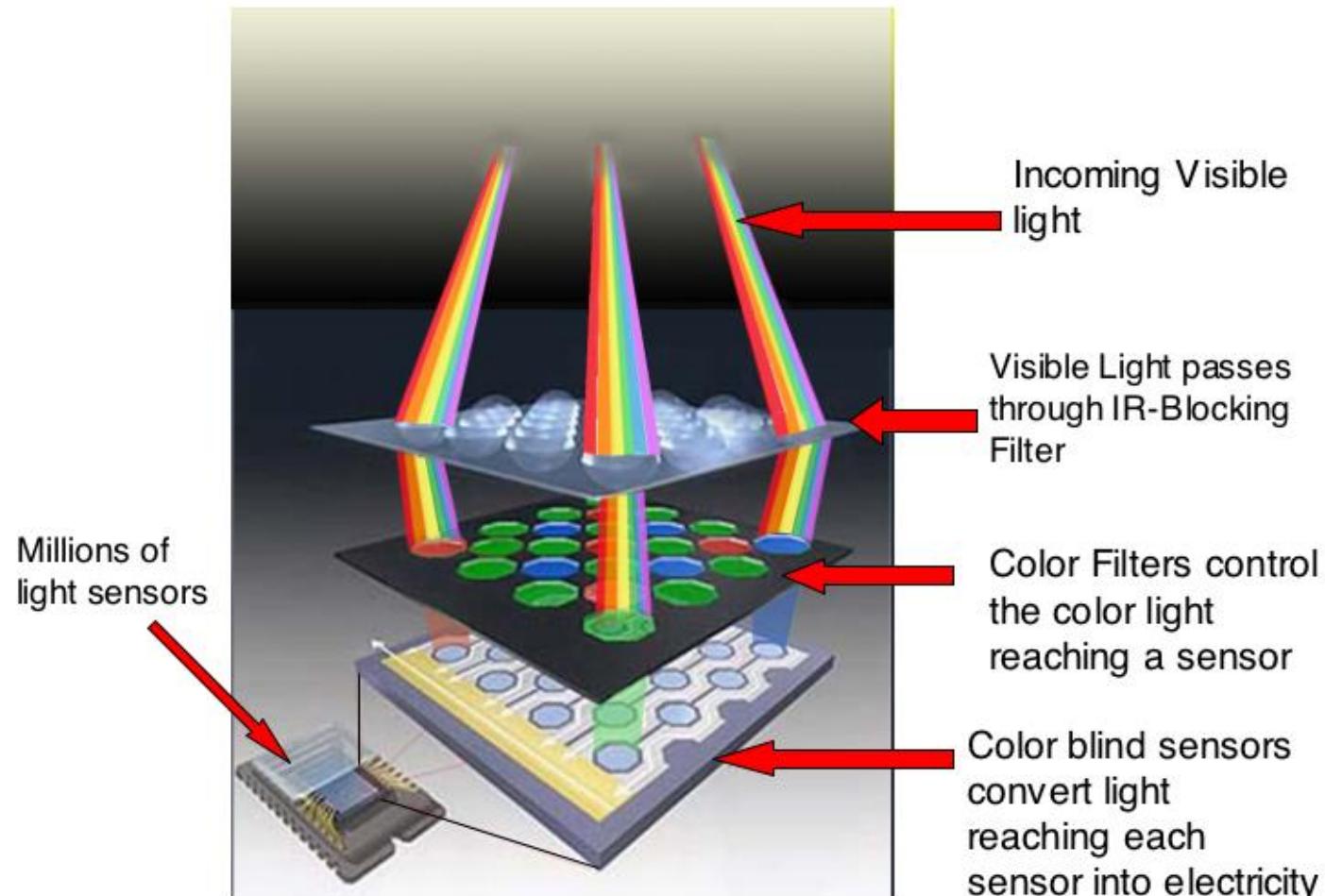
The two main kinds of sensor used in digital still and video cameras today are:

- charge-coupled device (CCD) and
- complementary metal oxide on silicon (CMOS).

Note: Traditionally, CCD sensors outperformed CMOS in quality-sensitive applications, such as digital SLRs, while CMOS was better for low-power applications, but today CMOS is used in most digital cameras.

From light to pixels

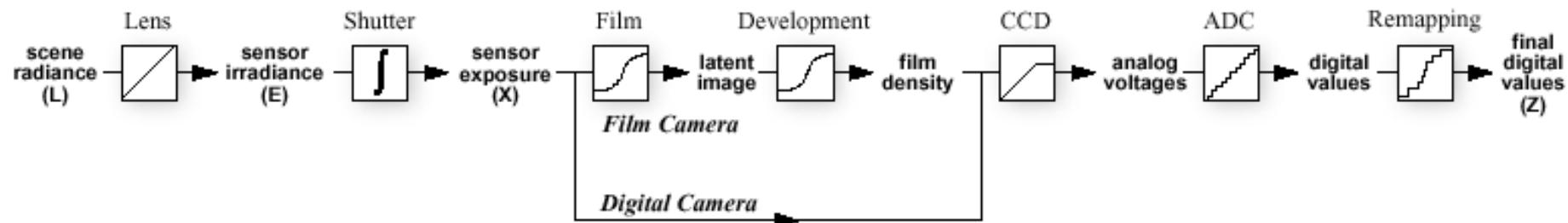
RGB Inside the Camera



Camera response function

Now how about the mapping f from radiance to pixels?

- It's also complex, but better understood
- This mapping f known as the film or camera response function



How can we recover radiance values given pixel values?

Why should we care?

- Useful if we want to estimate material properties
- Shape from shading requires radiance
- Enables creating high dynamic range images

What does the response function depend on?

$$f(\text{shutter speed}, \text{aperture}, \text{film stock}, \text{digitizer}, \dots)$$

Factors Affecting Digital Image Sensor

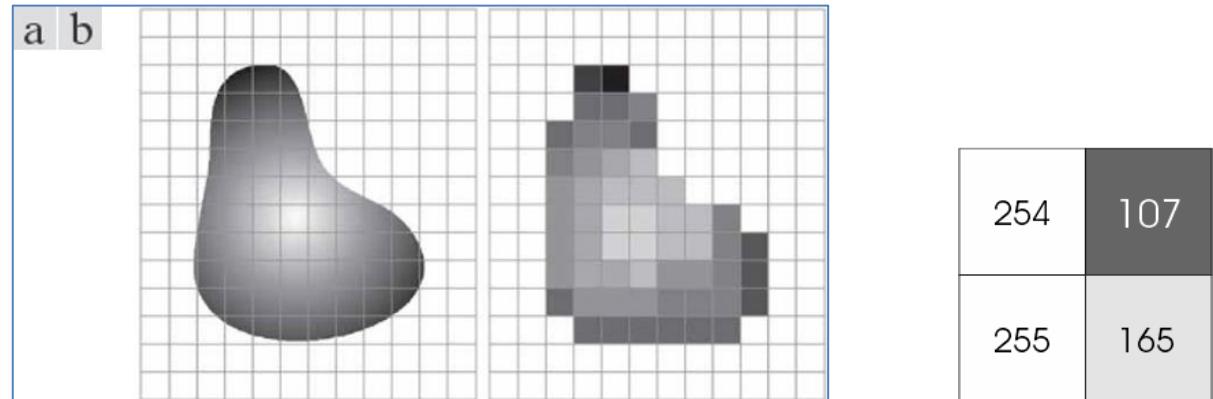
Main factors affecting the performance of a digital image sensor:

1. **Shutter speed (exposure time):** directly controls the amount of light reaching the sensor and determines under- or over-exposure. For dynamic scenes, determines the amount of motion blur in the resulting picture.
2. **Sampling pitch:** physical spacing between adjacent sensor cells on the imaging chip. A sensor with a smaller sampling pitch has a higher sampling density and hence provides a higher resolution (in terms of pixels) for a given active chip area. However, a smaller pitch also means that each sensor has a smaller area and cannot accumulate as many photons; this makes it not as light sensitive and more prone to noise.
3. **Fill factor:** the active sensing area size as a fraction of the theoretically available sensing area. Higher fill factors are usually preferable, as they result in more light capture and less aliasing .
4. **Chip size:** larger chip size is preferable, since each sensor cell can be more photo-sensitive. However, larger chips are more expensive to produce, not only because fewer chips can be packed into each wafer, but also because the probability of a chip defect goes up exponentially with the chip area.
5. **Analog gain:** Before analog-to-digital conversion, the sensed signal is usually boosted by a sense amplifier. In theory, a higher gain allows the camera to perform better under low light conditions (less motion blur due to long exposure times when the aperture is already maxed out). In practice, however, higher ISO (sensor sensitivity) settings usually amplify the sensor noise.
6. **Sensor noise:** Throughout the whole sensing process, noise is added from various sources. The final amount of noise present in a sampled image depends on all of these quantities, as well as the incoming light (controlled by the scene radiance and aperture), the exposure time, and the sensor gain. Also, for low light conditions where the noise is due to low photon counts.

What is an image?

- An image is an array, or a matrix, of square **pixels** (picture elements) arranged in columns and rows.
 - In a (8-bit) greyscale image each picture element has an assigned intensity that ranges from 0 to 255. A
th
 - An 8-bit greyscale image has 256 greyscales.
 - A “true colour” image has 24 bit colour depth = $8 \times 8 \times 8$ bits
 - = $256 \times 256 \times 256$ colours
 - = **~16 million colours.**

Matrix of pixel values



Access the pixels using their coordinates i.e. $I(3,2)$

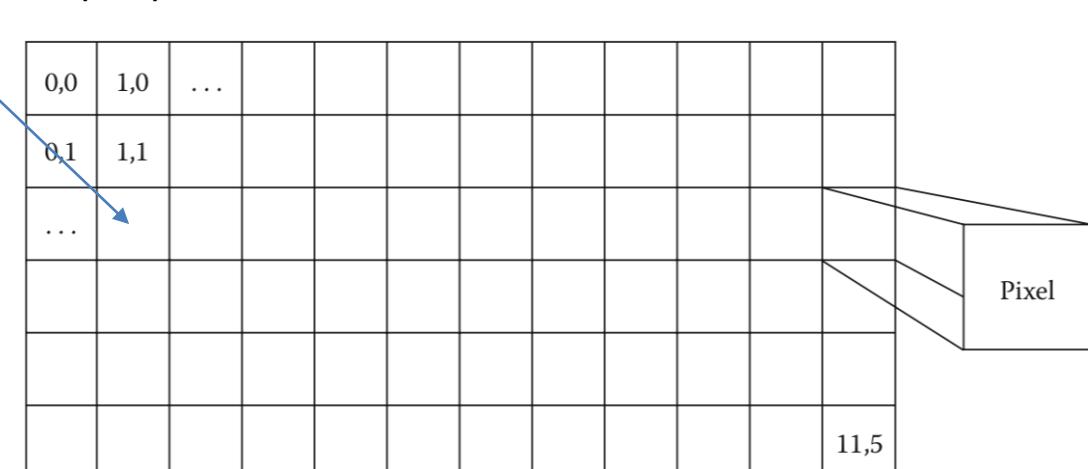
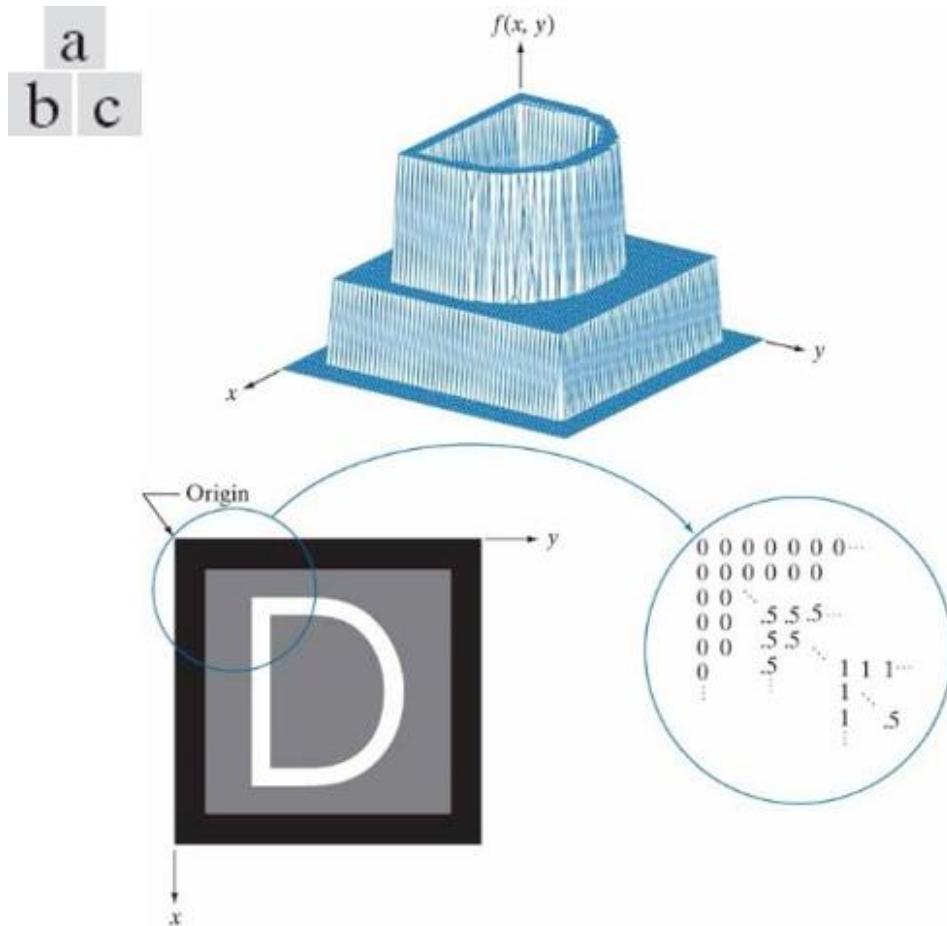


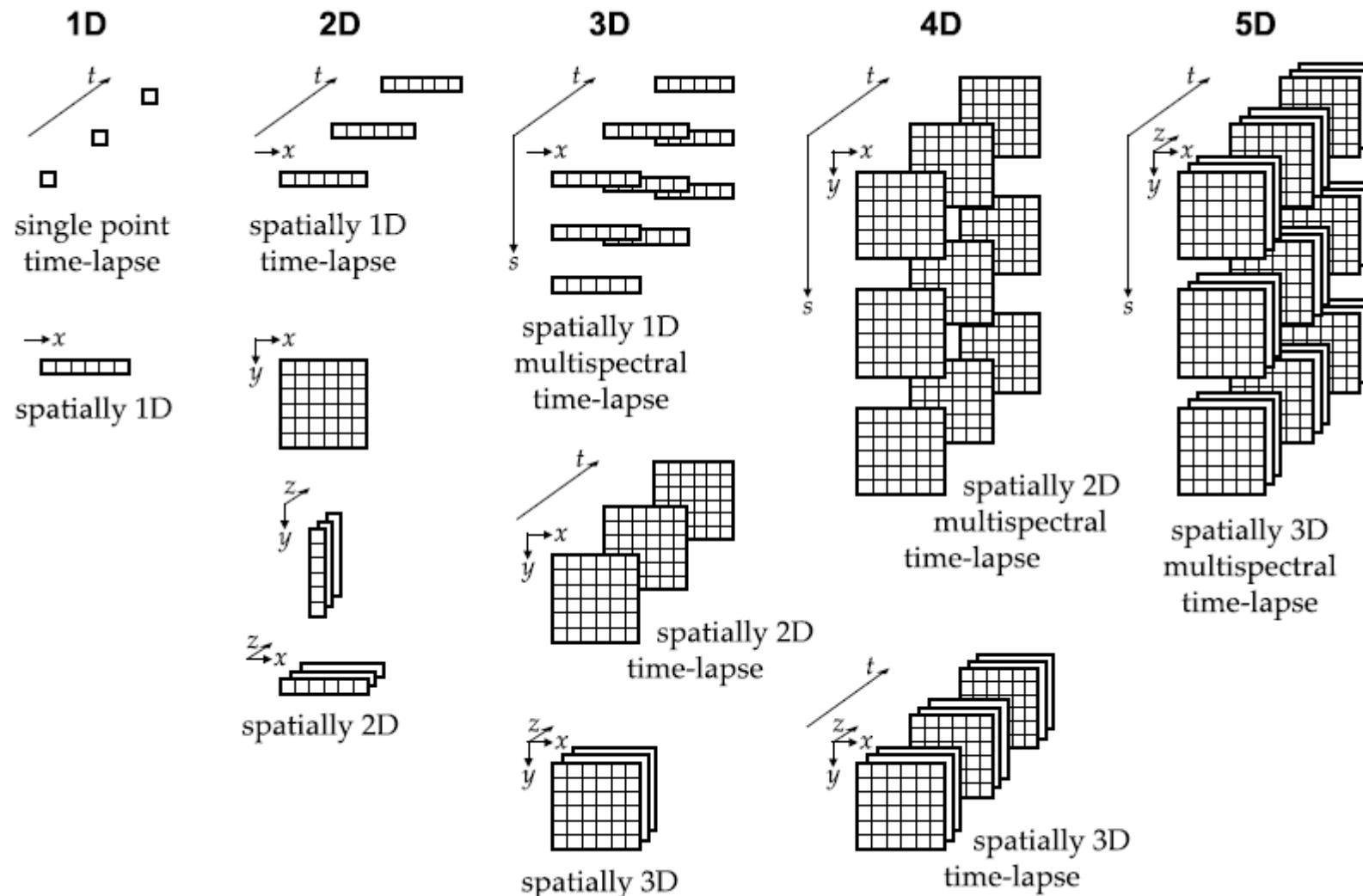
Figure 1.1 An example image of width 12 pixels and height 6 pixels.

What is an Image?



- (a) Image plotted as a surface.
- (b) Image displayed as a visual intensity array.
- (c) Image shown as a 2-D numerical array. (The numbers 0, .5, and 1 represent black, gray, and white, respectively.)

Terminology: Image as Matrix



- **Biological Image Analysis Primer** Erik Meijering and Gert van Cappellen
- E. Meijering, G. Cappellen. "Quantitative biological image analysis." *Imaging cellular and molecular biological functions*. Springer, 2007. 45-70.

Reading Assignment:

Reading assignment:

- CANVAS\Review-Tutorials\review_of_matrices_and_vectors.ppt

Spatial Resolution

a
b
c
d



Effects of reducing spatial resolution.
The images shown are at:

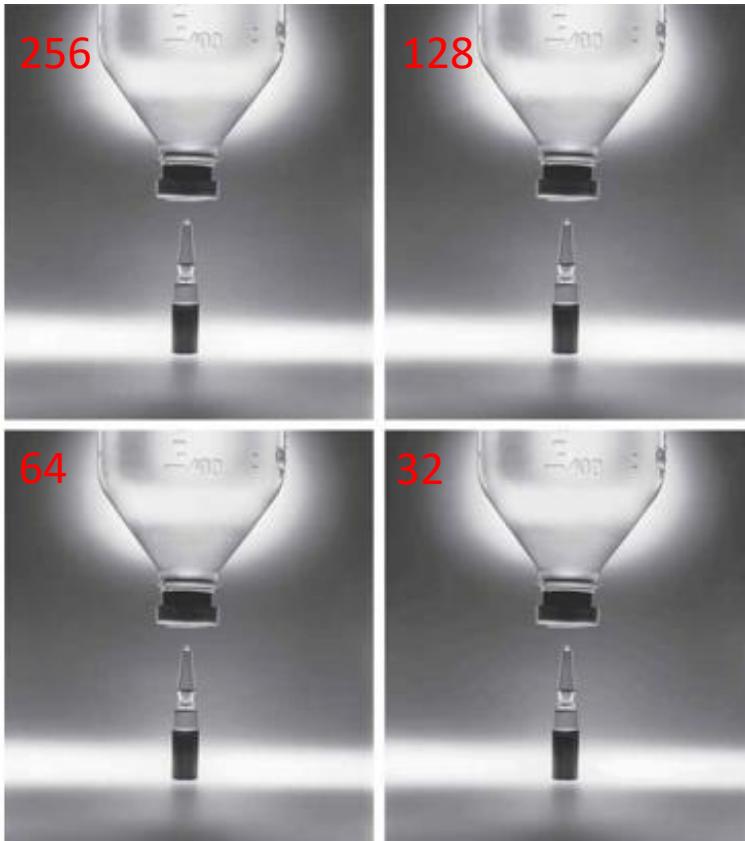
- (a) 930 dpi,
- (b) 300 dpi,
- (c) 150 dpi, and
- (d) 72 dpi.

Dots per inch (DPI, or dpi^[1]) is a measure of spatial [printing](#), [video](#) or [image scanner](#) dot density, in particular the number of individual dots that can be placed in a line within the span of 1 [inch](#) (2.54 cm).

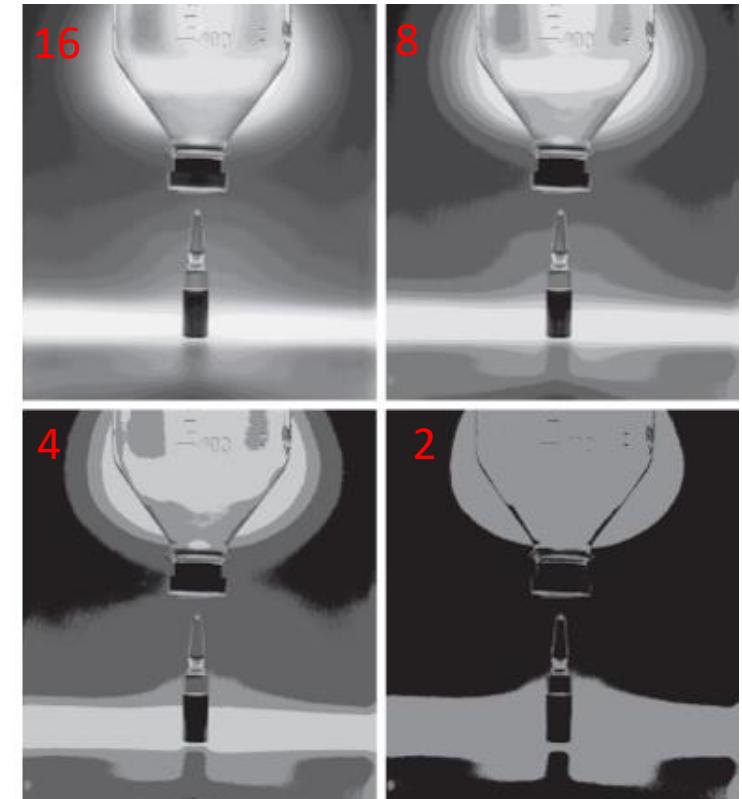
Number of Intensity Levels

(a) 2022×1800 ,

a
b
c
d



e
f
g
h



Intensity levels, while keeping the image size constant. (Original image courtesy of the National Cancer Institute.)

Content Complexity



a b c

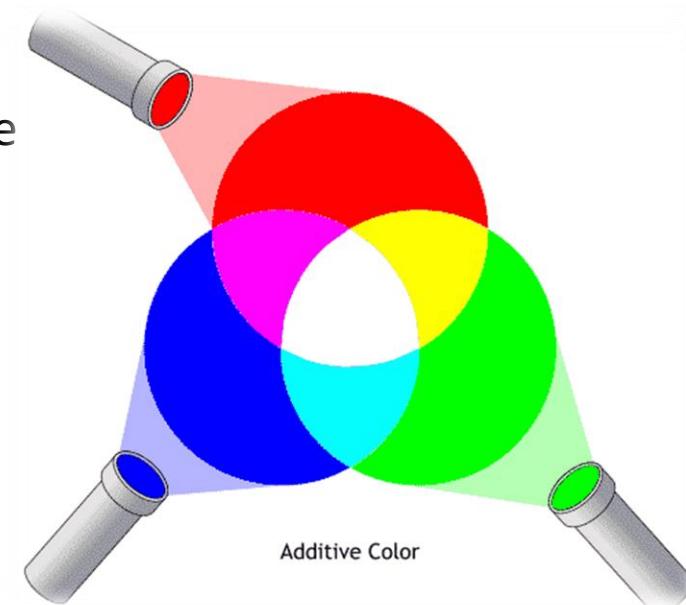
- (a) Image with a low level of detail.
- (b) Image with a medium level of detail.
- (c) Image with a relatively large amount of detail.
(Image (b) courtesy of the Massachusetts Institute of Technology.)

Representing Color

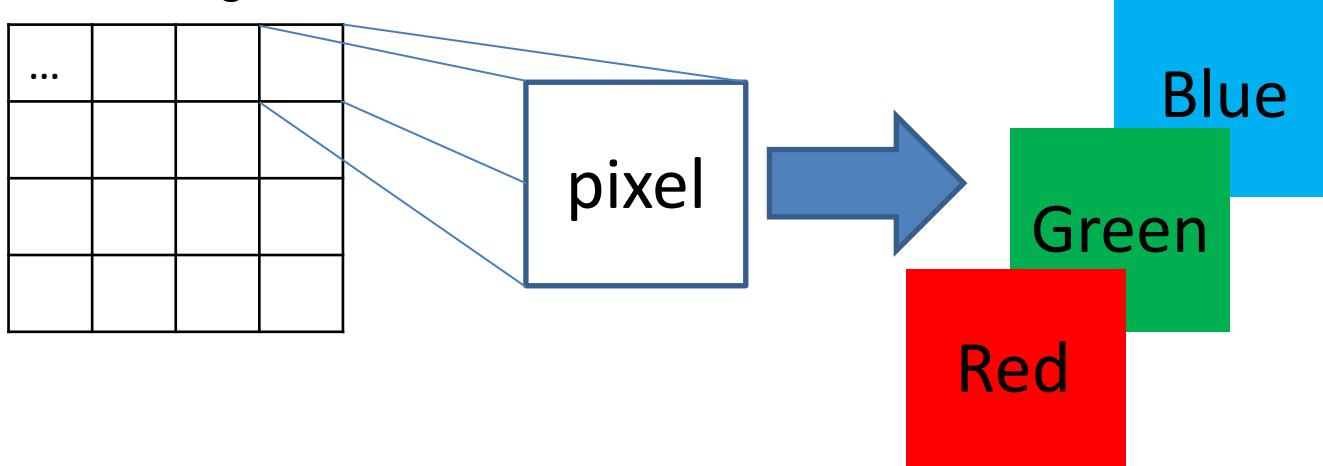
- Pixels represent a single point of intensity level or color, assigned to a coordinate

Representing color: At each pixel in RGB color space, the combination of

- Red channel (R),
- Green channel (G), and
- Blue channel (B) produces the final color in the image.

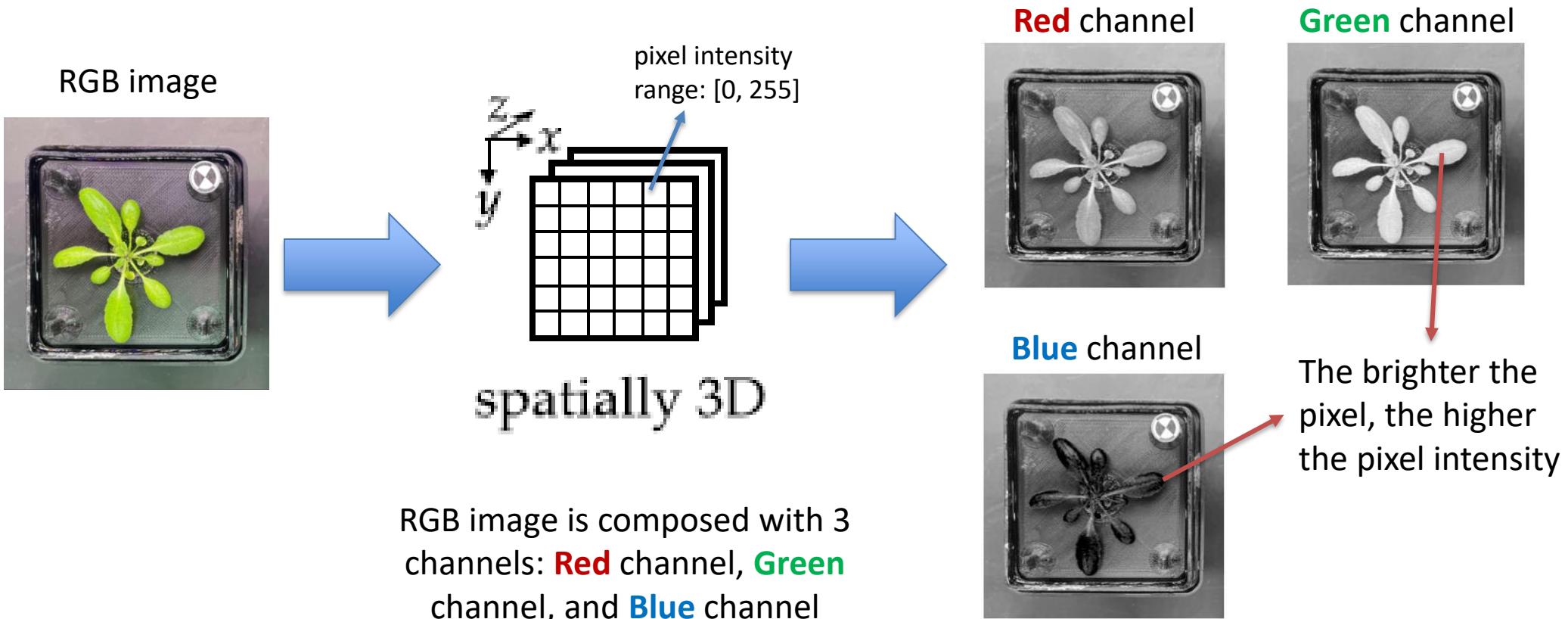


An image



Red	Green	Blue	Color
• 255 • 255	• 255 • 255	• 255 • 0	• White • Yellow

What do the red, green, and blue channels look like in a real image?

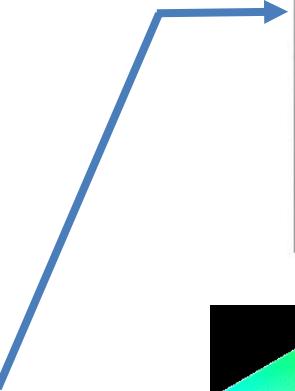
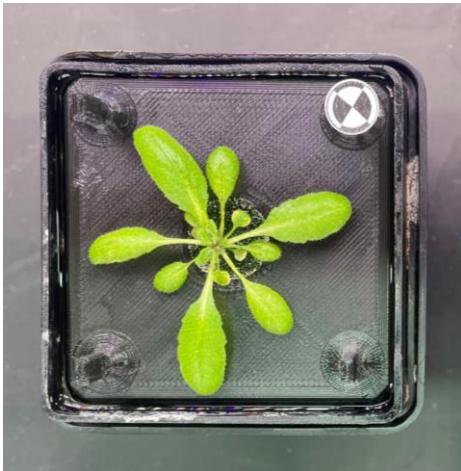


Color Space: RGB

RGB (3 channels)

- The RGB color space splits each pixel into three colors, representing the three primary color components, **red**, **green**, and **blue**.

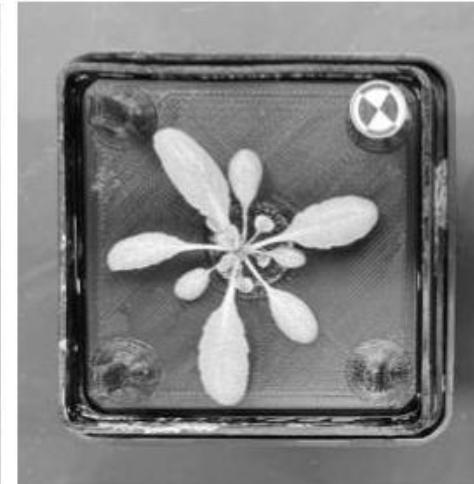
RGB image



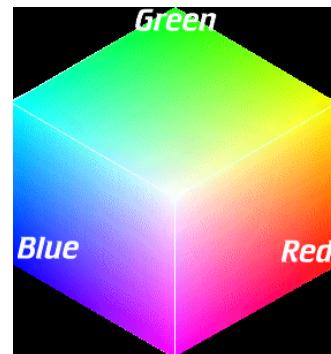
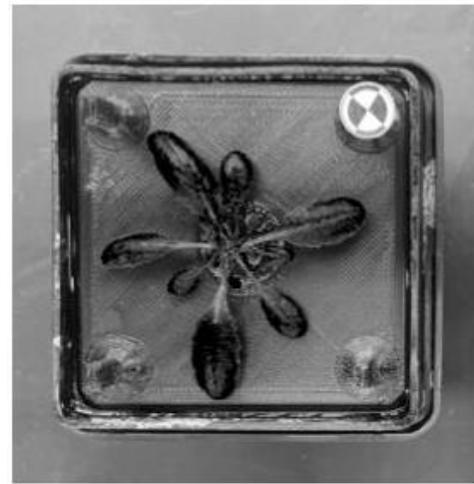
Red channel



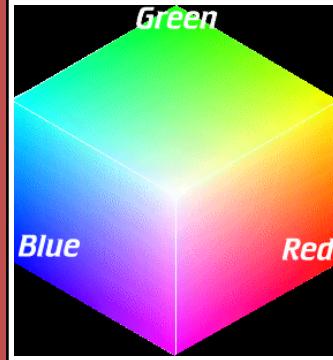
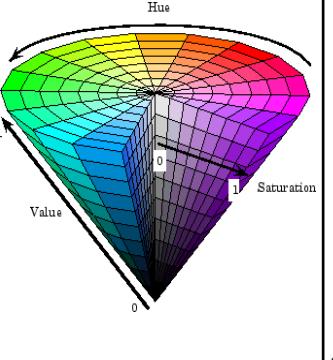
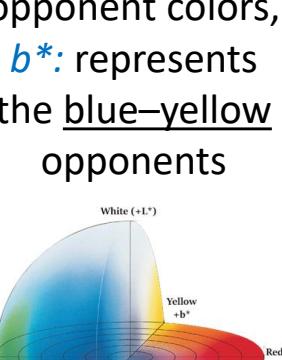
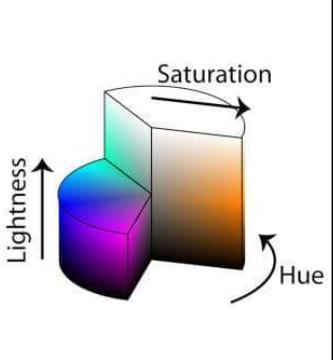
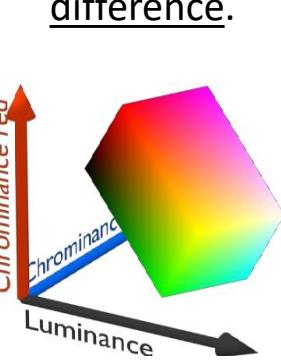
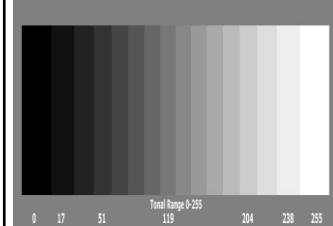
Green channel



Blue channel



Color space

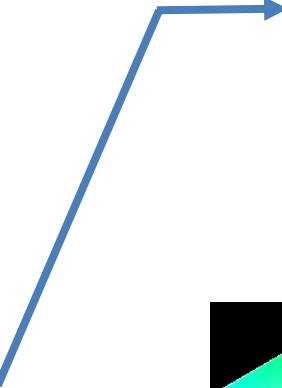
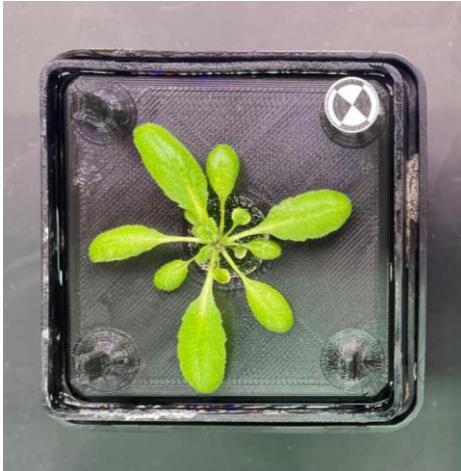
Color space:	RGB	HSV	L*a*b*	HSL	YCbCr	Grayscale
Number of channels:	3	3	3	3	3	1
Description of each channel:	R: red, G: green, B: blue.	H: hue, S: saturation, V: value	L^* : perceptual lightness, a^* : relative to the green-red opponent colors, b^* : represents the blue-yellow opponents	H: hue, S: saturation, L: lightness	Y: luminance Cb: blue difference, Cr: red difference.	only intensity information
						

Color spaces RGB

RGB (3 channels)

- The RGB color space splits each pixel into three colors, representing the three primary color components, **red**, **green**, and **blue**.

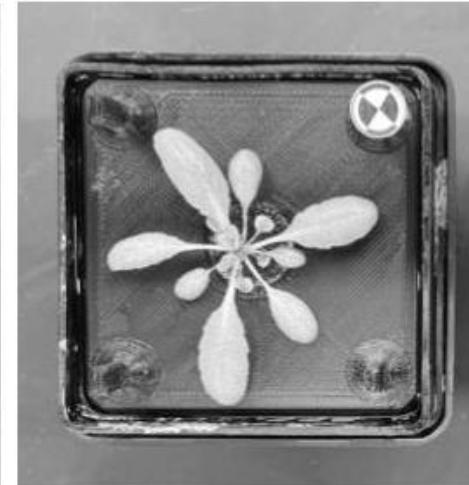
RGB image



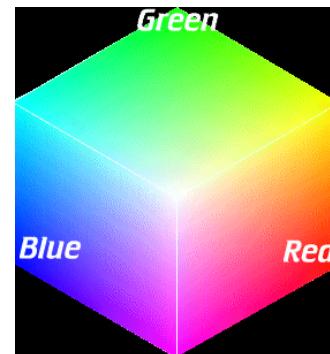
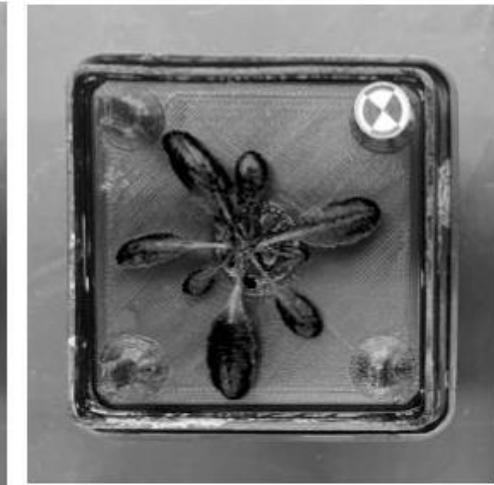
Red channel



Green channel



Blue channel

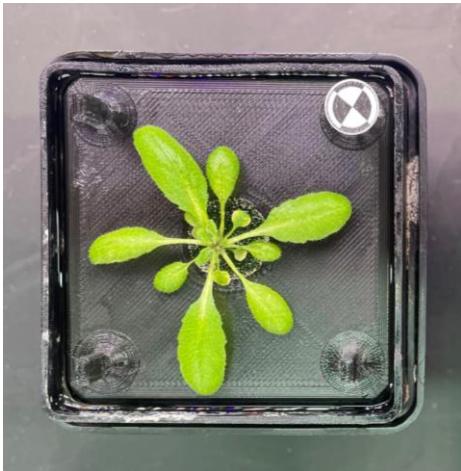


Color spaces HSV

HSV (3 channels)

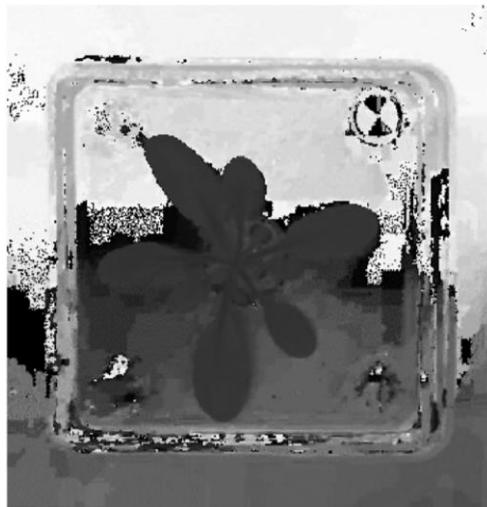
- The HSV color space represents the separate hue, saturation, and value components of a pixel.

RGB image



convert
to HSV

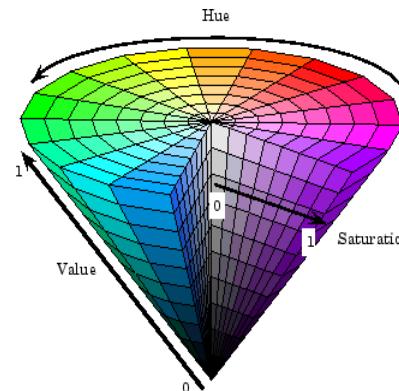
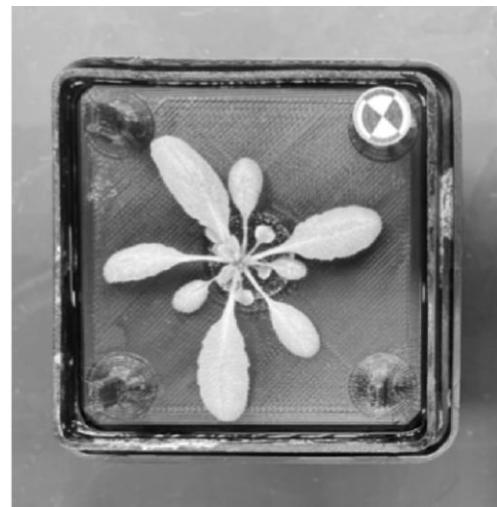
Hue channel



Saturation channel



Value channel

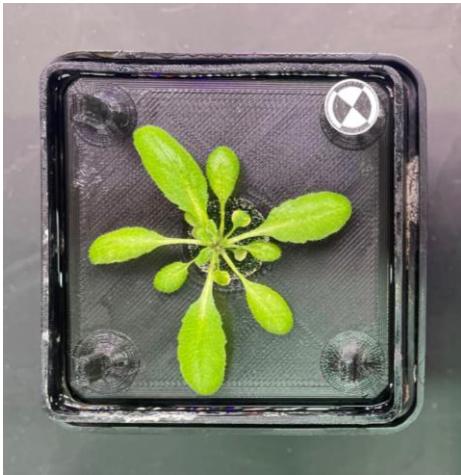


Color spaces L*a*b*

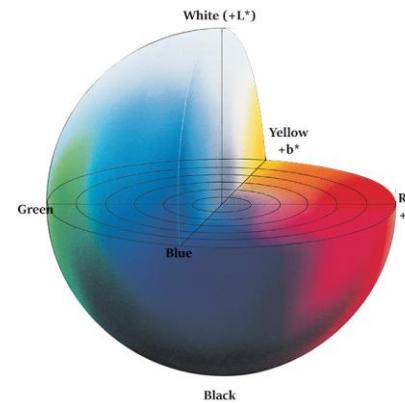
L*a*b*
(3 channels)

- L*a*b* expresses color as three values: L^* for perceptual lightness, a^* is relative to the green-red opponent colors, b^* represents the blue-yellow opponents

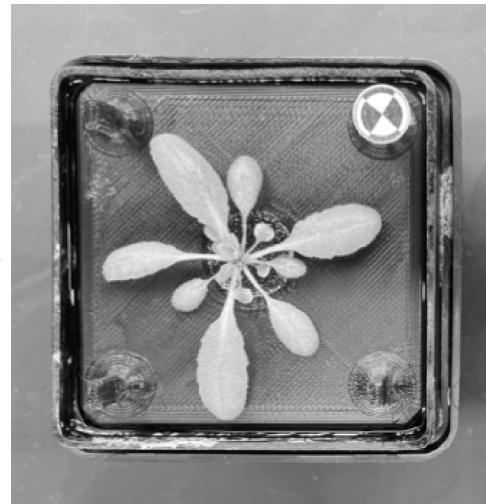
RGB image



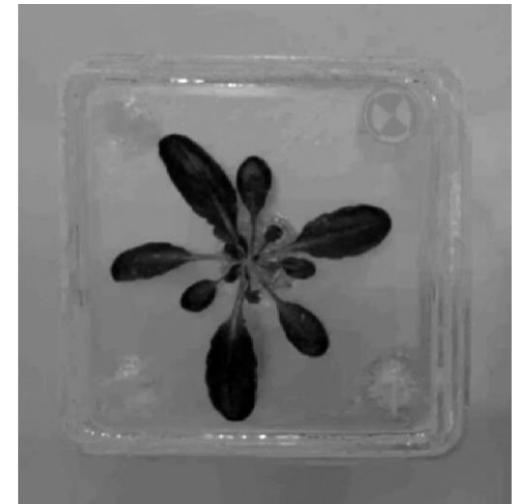
convert to
L*a*b*



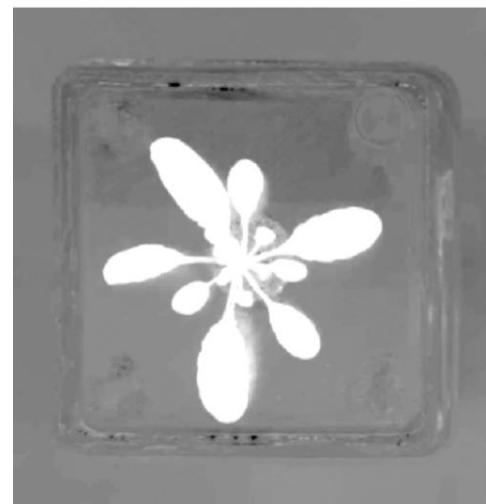
L* channel



a* channel



b* channel



Color Space Transformations

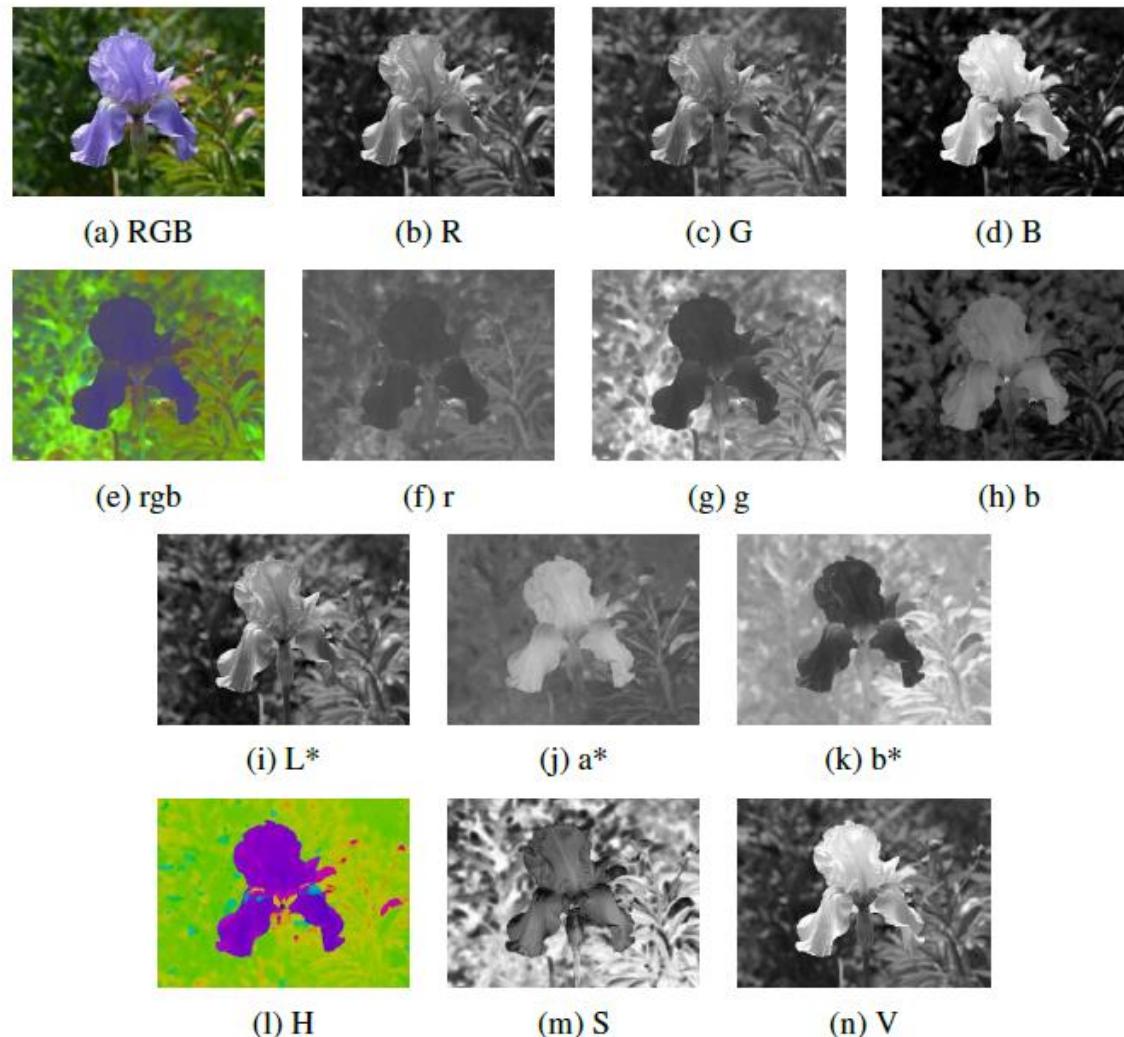


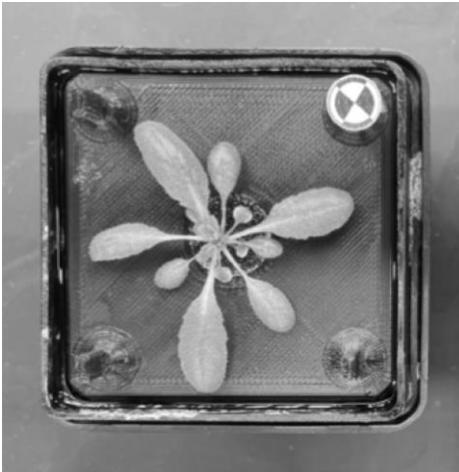
Figure 2.33 Color space transformations: (a–d) RGB; (e–h) rgb . (i–k) $L^*a^*b^*$; (l–n) HSV.
Note that the rgb , $L^*a^*b^*$, and HSV values are all re-scaled to fit the dynamic range of the printed page.

Color spaces Grayscale

Grayscale
(1 channel)

- The pixel in grayscale image carries only intensity information

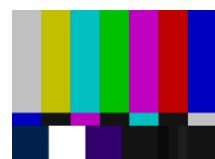
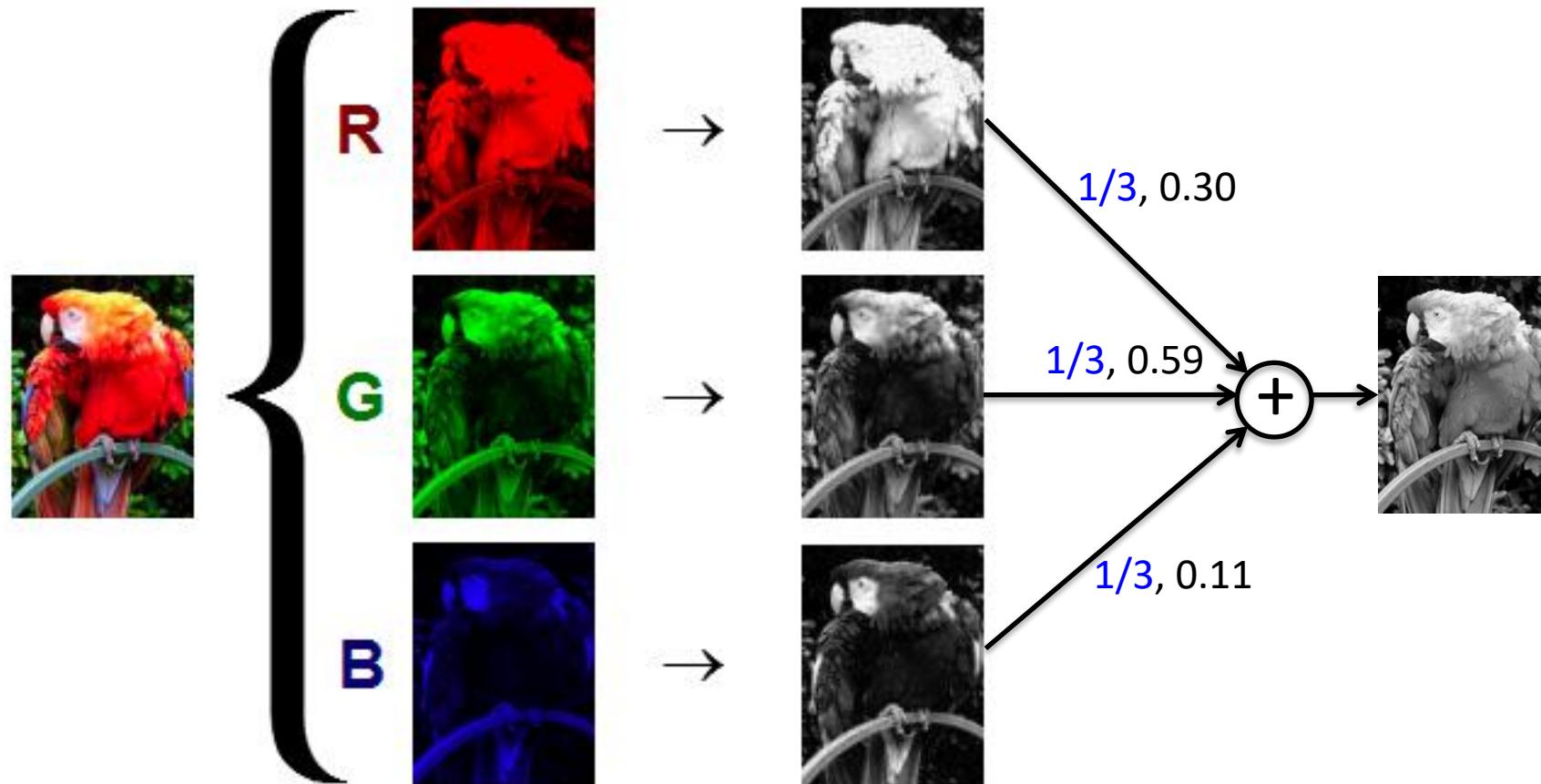
Grayscale image



For example, from RGB to grayscale, for each pixel:

$$\frac{R + G + B}{3}$$

RGB to Grayscale (rgbtogray.c)



Other RGB weights to convert to grayscale (lightness/ intensity/ luma):

$$Y = 0.3 R + 0.59 G + 0.11 B \quad (\text{ITU Rec. 601 NTSC})$$

$$L = 0.2126 R + 0.7152 G + 0.0722 B \quad (\text{ITU Rec. 709 HDTV})$$

Other Resources

- ICCV 2019 Tutorial on Understanding Color and the In-Camera Image Processing Pipeline for Computer Vision
- https://www.eecs.yorku.ca/~mbrown/ICCV2019_Brown.html

Point Operations (Intensity Transformations)

CS/ECE 8690 Computer Vision

Instructor: Filiz Bunyak Ersoy bunyak@missouri.edu

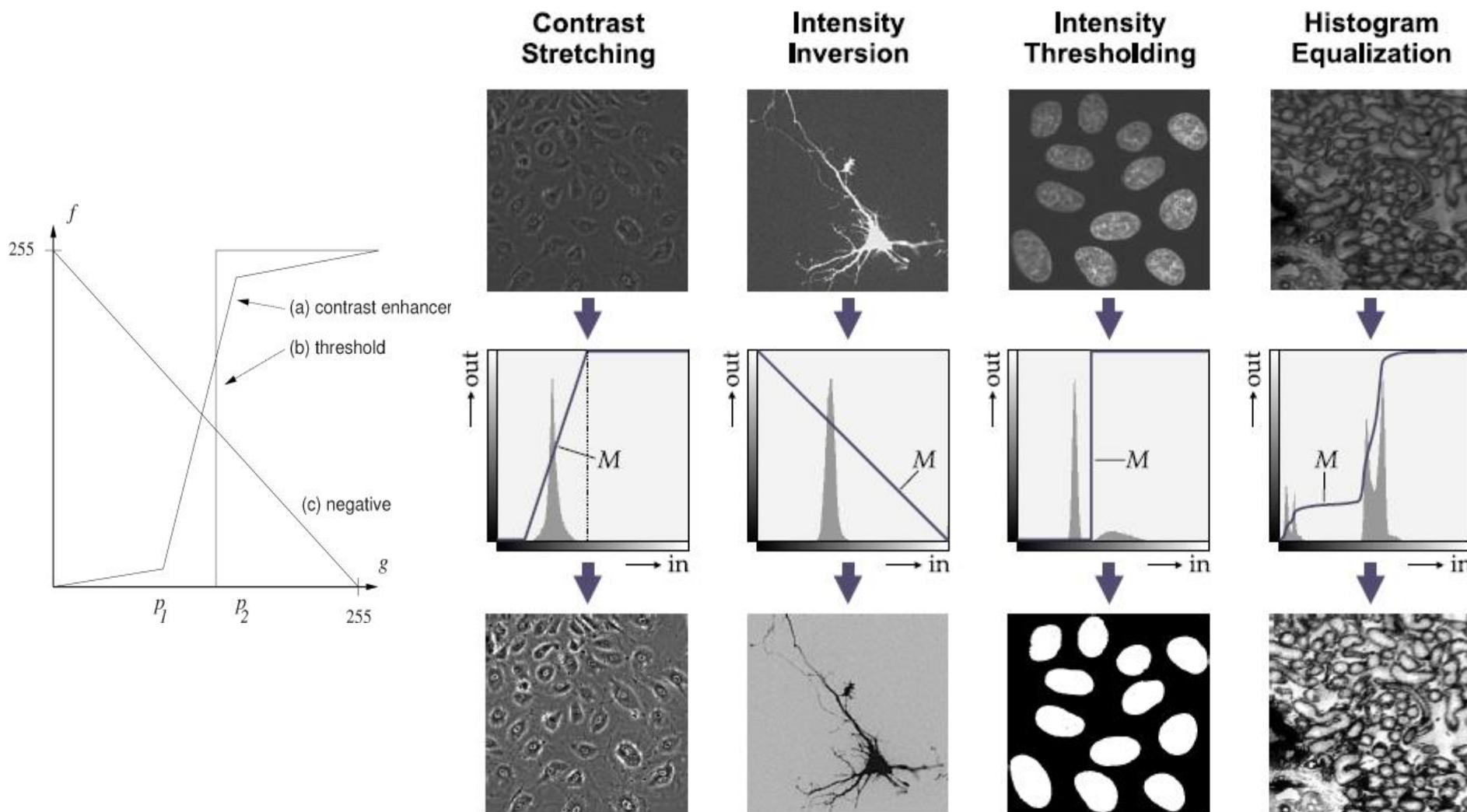
TA: Imad Toubal (CS PhD): itdfh@mail.missouri.edu

Image Processing

Three classes of operations that are most commonly used:

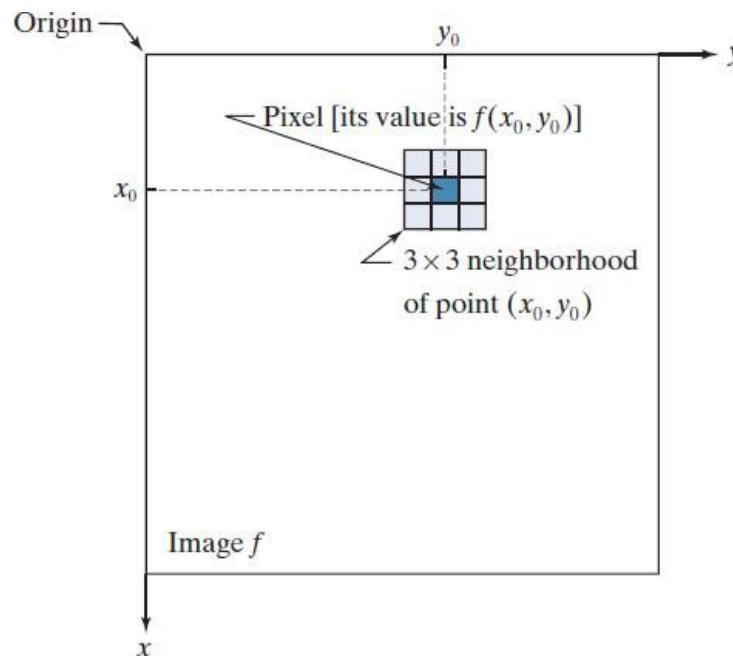
1. Intensity transformation (Point operators/processes),
2. Local image filtering,
 1. Linear Filtering
 2. Non-Linear Filtering
 3. Morphological Filtering
3. Geometrical transformation

Point Operators (Intensity Transformation)



Neighborhood of a Pixel

A 3×3 neighbourhood about a point (x_0, y_0) in an image. The neighborhood is moved from pixel to pixel in the image to generate an output image. Recall from Chapter 2 that the value of a pixel at location (x_0, y_0) is $f(x_0, y_0)$, the value of the image at that location.



Point operators

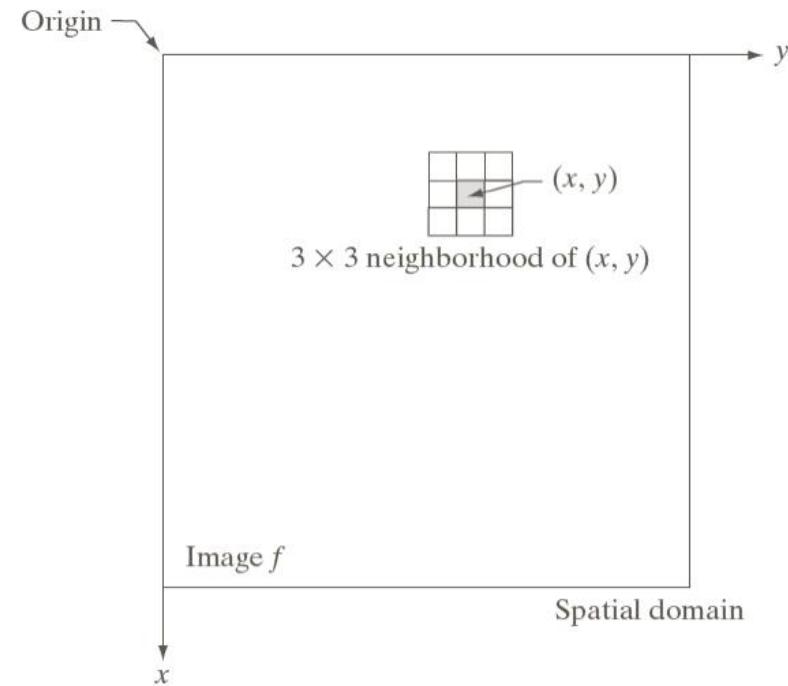
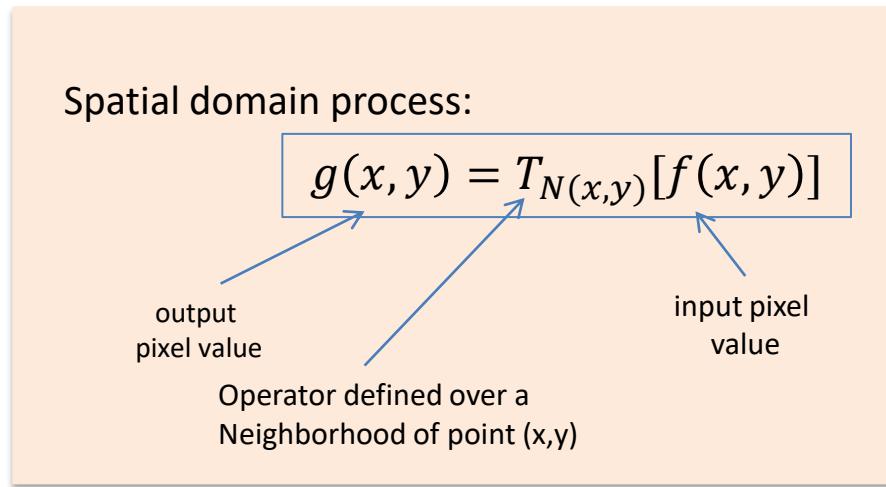
Point Operators:

Each **output pixel's value** depends on:

- Corresponding **input pixel value**
- Some **globally collected information or parameters**

Examples of such operators include

- *brightness and contrast adjustments*
- color correction and transformations



For point operators' neighborhood:

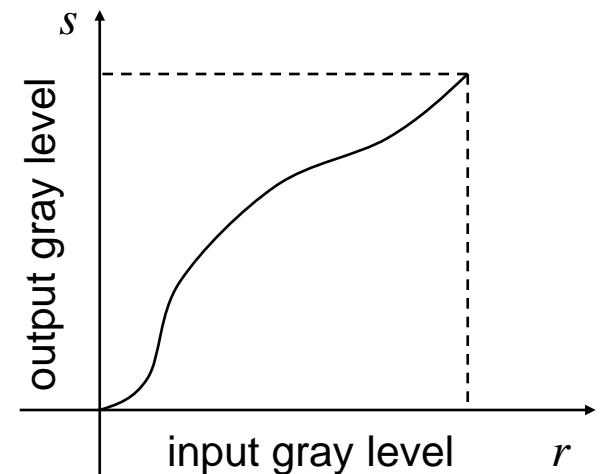
$$N(x, y) = (x, y)$$

More on Point Operations: Intensity Transformation

- Map a given gray or color level r to a new level s

$$f(x, y) \rightarrow g(x, y)$$
$$x = 1, \dots, M, \quad y = 1, \dots, N$$
$$s = T(r)$$
$$r, s = 0, \dots, 255$$

- Memory-less
 - output at (x, y) only depend on the input intensity at the same point
 - Pixels of the same intensity gets the same transformation
- Does not bring in new information,
- May cause loss of information **HOW???**
- But can improve visual appearance or make features easier to detect



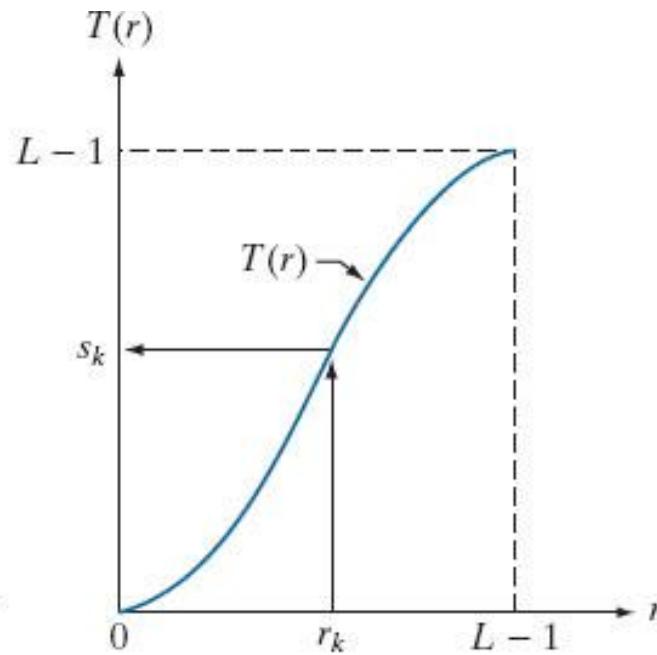
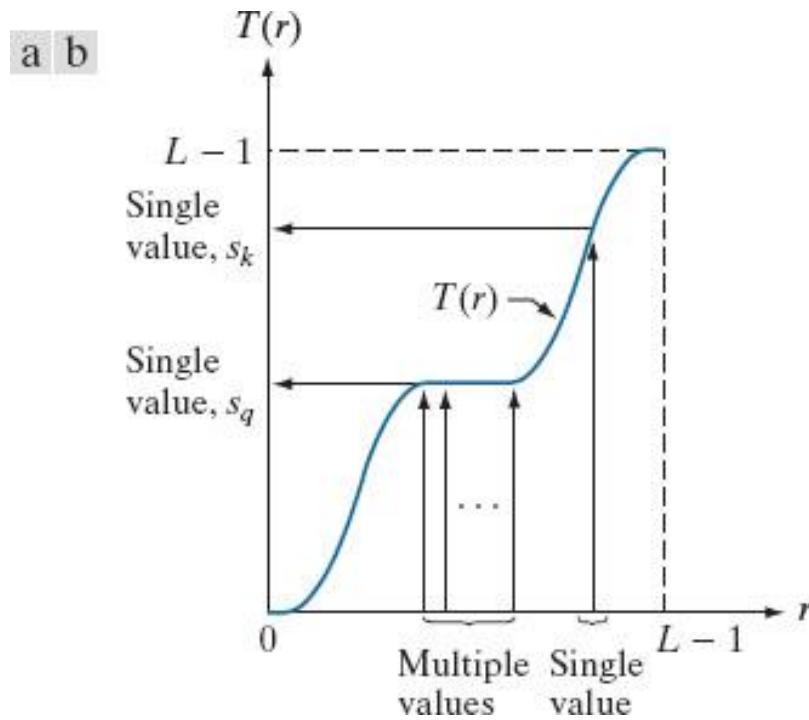
Typically, $T(\cdot)$ is monotonically increasing (but not always)

WHY??

Intensity Mapping

- (a) Monotonic increasing function, showing how multiple values can map to a single value.
- (b) Strictly monotonic increasing function. This is a one-to-one mapping, both ways.

In mathematics, a monotonic function is a function between ordered sets that preserves or reverses the given order.



A function is strictly monotonically increasing if it is always increasing on its domain and its graph is never horizontal; that is, the derivative of the function is strictly greater than zero on its domain.

Point operators: Some Pixel Transforms

Some common operators:

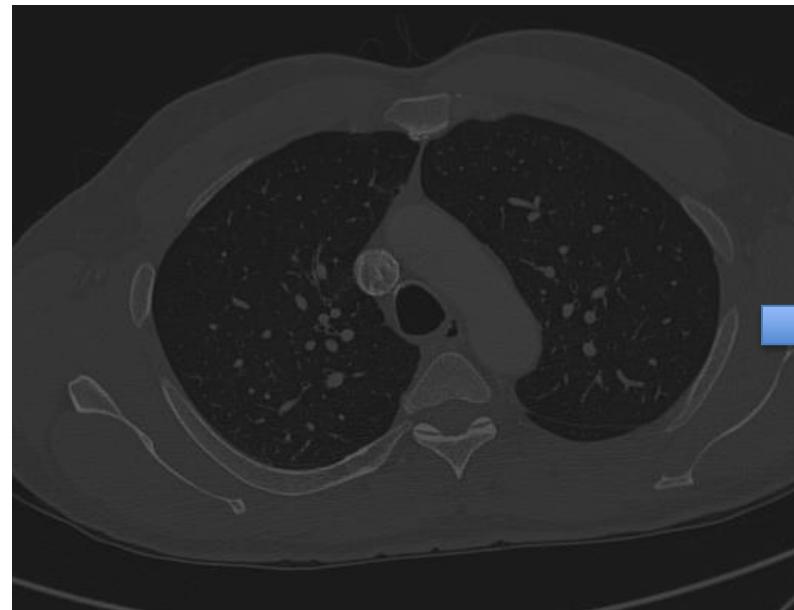
Multiplication and addition with a constant:

$$g(x) = af(x) + b$$

a: gain ($a > 0$)

b: bias

(controls contrast and brightness)



Spatially varying gain and bias:

$$g(x) = a(x)f(x) + b(x)$$

Power-law (Gamma) Transformation:

$$g(x) = c[f(x)]^\gamma$$

Linear Blend:

$$g(x) = (1 - \alpha) f_0(x) + \alpha f_1(x)$$

(cross dissolve between two images or
morphing)

Point operators: Case Study Contrast Enhancement with Constant Gain & Bias

Constant Gain & Bias:

Multiplication & addition with a constant $g(x) = af(x) + b$

Goal: Enhance contrast.

Lets try **a=3.2** and **b= -83.2**

Lowest intensity value in the image : 26

Highest intensity value in the image : 105

Range before transform: $(105-26)=79$

Gain:

$$26 \rightarrow 3.2 \cdot 26 = 83.2$$

$$105 \rightarrow 3.2 \cdot 105 = 336$$

$$\text{Range After: } 336 - 83.2 = \underline{\underline{252.8}}$$

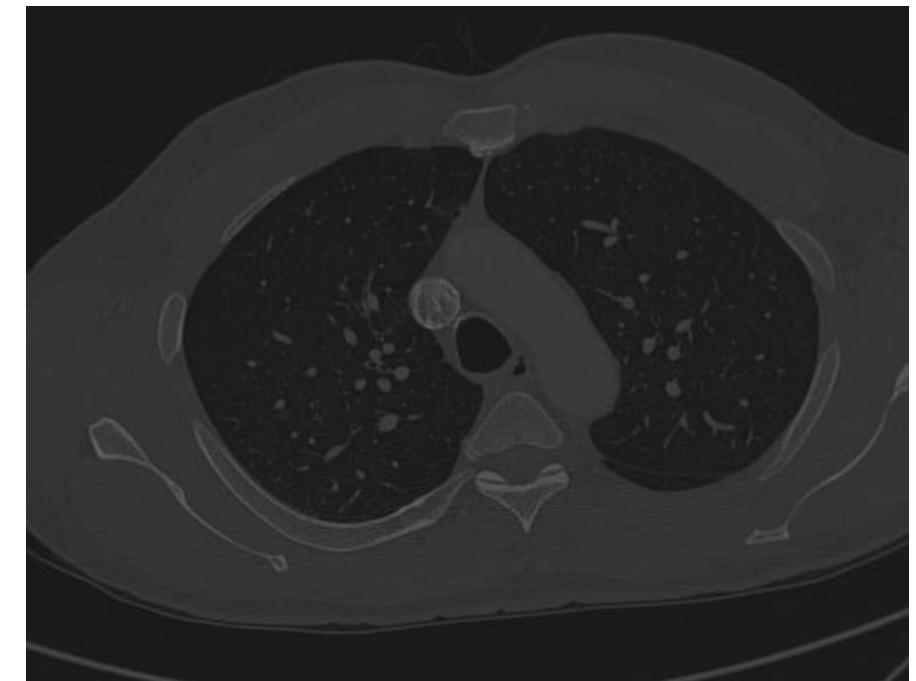
is this enough?

Gain+Bias:

$$26 \rightarrow 3.2 \cdot 26 - 83.2 = 0$$

$$105 \rightarrow 3.2 \cdot 105 - 83.2 = 252.8$$

$$\text{Range After: } 255 - 0 = \underline{\underline{252.8}}$$

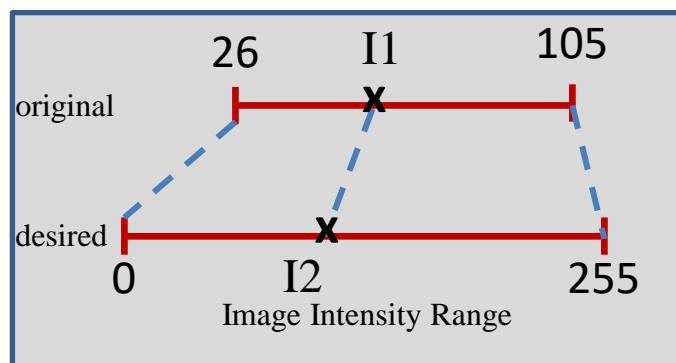


Point operators: Case Study Contrast Enhancement with Constant Gain & Bias

Constant Gain & Bias: Multiplication and addition
with a constant

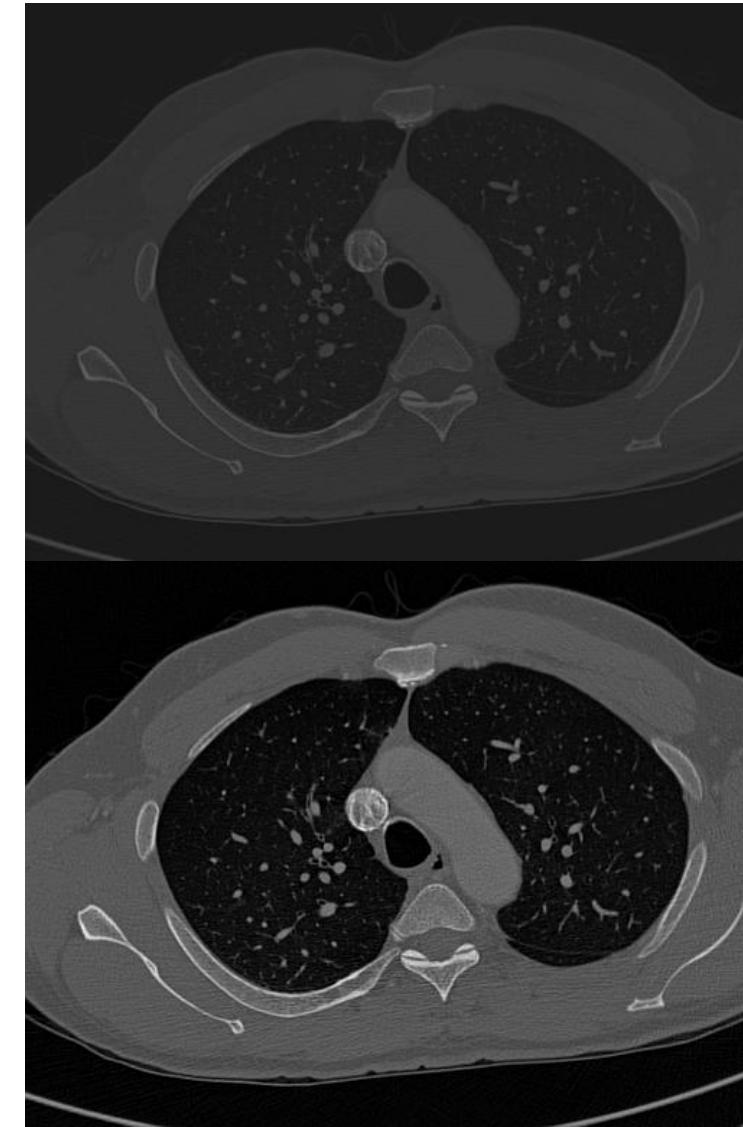
$$g(x) = af(x) + b$$

Original Range: [26-105]
Desired Range: [0-255]



$$\frac{I2 - 0}{255 - 0} = \frac{I1 - 26}{105 - 26}$$

$$I2 = \left(\frac{255}{105 - 26} \right) I1 + \left(\frac{-26 * 255}{105 - 26} \right)$$



Point operators 1: Pixel Transforms

Contrast Enhancement with Constant Gain & Bias:

$$g(x) = af(x) + b$$

a: gain ($a>0$) & b: bias

Let's try **a=1.7** and **b= -125.8**

Min(I) : 74

Max(I) : 224

Let's try to expand the range of values

Range before transform: $(224-74)=150$

$$74 \rightarrow 1.7 \cdot 74 = 125.8$$

$$224 \rightarrow 1.7 \cdot 224 = 380.8$$

Range after gain: $380.8 - 125.8 = 255$

$$74 \rightarrow 1.7 \cdot 74 = 125.8 - 125.8 = 0$$

$$224 \rightarrow 1.7 \cdot 224 = 380.8 - 125.8 = 255$$

After gain+bias: $255 - 0 = 255$

Matlab Program:

```
I1 = imread('pout.tif');  
a=1.7;  
b=-125.8;  
I2=a*I1+b; % use matrix operations  
subplot(1,2,1); imshow(I1);  
subplot(1,2,2); imshow(I2);  
imwrite(I2,'pout2.tif');
```



Original Range: [74-224]
Expected Range: [0-255]

Output Range: [0-129]
What went wrong????

Point operators 1: Pixel Transforms

What happened???

Expected

Original range 75-224, $a=1.7$

$$74 \rightarrow 1.7 \cdot 74 = 125.8$$

$$224 \rightarrow 1.7 \cdot 224 = 380.8$$

What happened???

Obtained

Original range 75-224, $a=1.7$

$$74 \rightarrow 1.7 \cdot 74 = 126$$

$$224 \rightarrow 1.7 \cdot 224 = 255$$

Because when read, $I1$ was `uint8` !!!

- Be careful about variable types.
Even while using Matlab !
- Always verify your output,
easiest test: check your intensity range



$a=1.7$



Even worse when $a=2.5$

Point operators 1: Pixel Transforms

Multiplication and addition with a constant:

$$g(x) = af(x) + b$$

a: gain ($a > 0$) & b: bias

Motivation: Enhance contrast



Original Range: [74-224]
Expected Range: [0-255]



Output Range: [0-129]



Output Range: [0-255]
What is wrong???? Finally

Matlab Program:

```
I1 = imread('pout.tif');
I1=double(I1);
a=1.7;
b=-125.8;
I2=a*I1+b;
subplot(1,2,1); imshow(uint8(I1));
subplot(1,2,2); imshow(uint8(I2));
imwrite(uint8(I2),'pout2.tif');
```

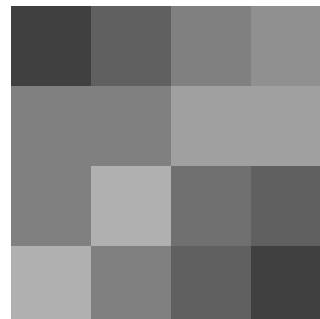
Intensity Histogram

- Intensity histogram represents the **count/frequency** of various **intensity levels** in the image.
- For each intensity level, count the number of pixels having that level
- Nearby levels can be grouped to form **bins**.

Example:

Image size: 4x4

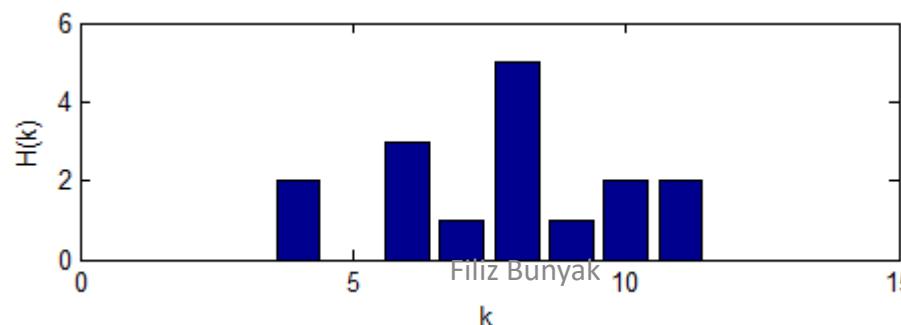
Intensity levels: 4bits/pixel



4	6	8	9
8	8	10	10
8	11	7	6
11	8	6	4

What type of information
is lost????

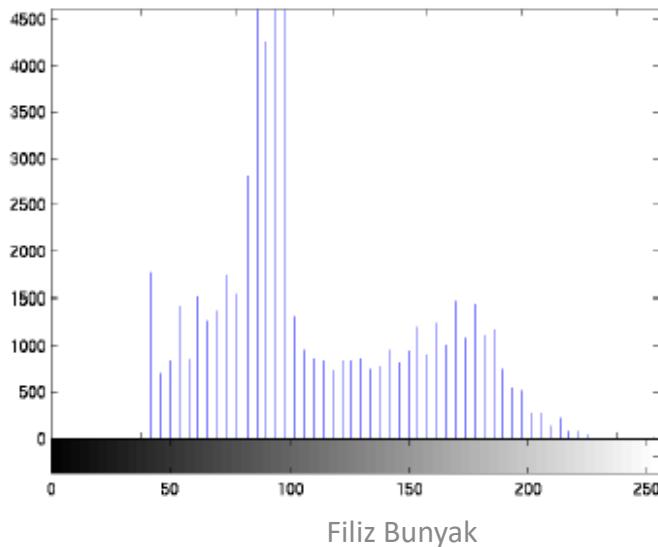
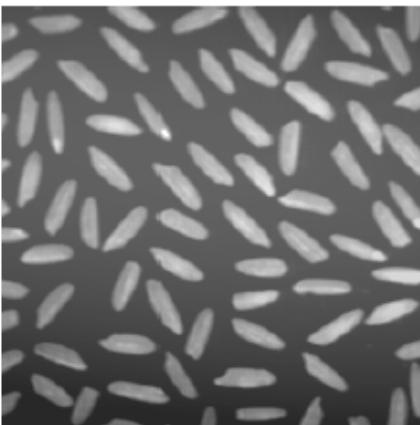
$$H(k) = \begin{matrix} 0 & 0 & 0 & 0 & 2 & 0 & 3 & 1 & 5 & 1 & 2 & 2 & 2 & 0 & 0 & 0 & 0 \end{matrix} \quad \text{sum}(H) == 16$$
$$k = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix}$$



Intensity Histogram

- Intensity histogram represents the count/frequency of various intensity levels in the image.
- For each intensity level, count the number of pixels having that level
- Nearby levels can be grouped to form bins.

```
I = imread('rice.tif');  
imshow(I)  
figure, imhist(I,64)
```



Matlab functions:

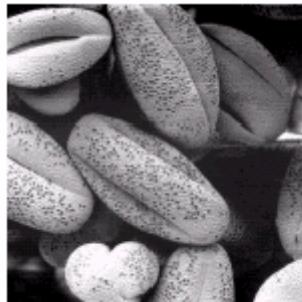
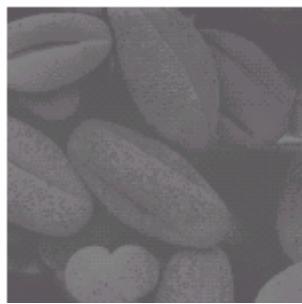
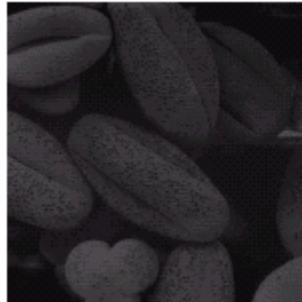
>> imhist -> specifically for image
>> hist -> for 1D arrays

How to convert 2D/3D image (matrix) to 1D array
> A=I(:)

Contrast, Brightness and Histogram

How do you think the histograms for these images look like?

- Where do they concentrate???
- How is the range???
- Balanced? Unbalanced???

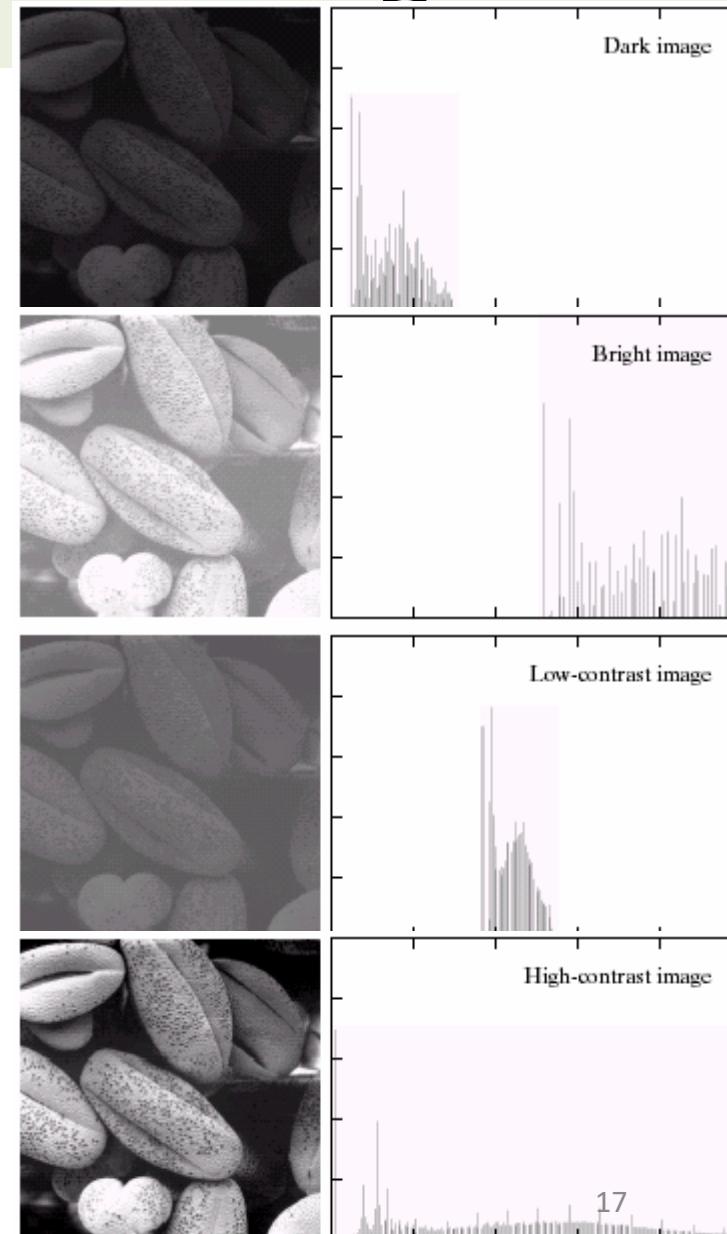


Contrast, Brightness and Histogram

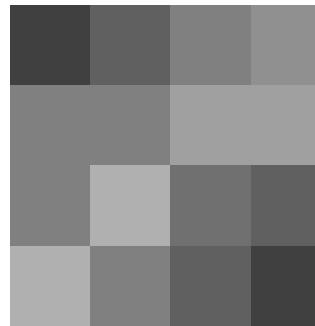
“unbalanced” histograms do not fully utilize the dynamic range

- **Low contrast image:** narrow luminance range
- **Under-exposed image:** concentrating on the dark side
- **Over-exposed image:** concentrating on the bright side

“balanced” histogram gives more pleasant look and reveals rich details



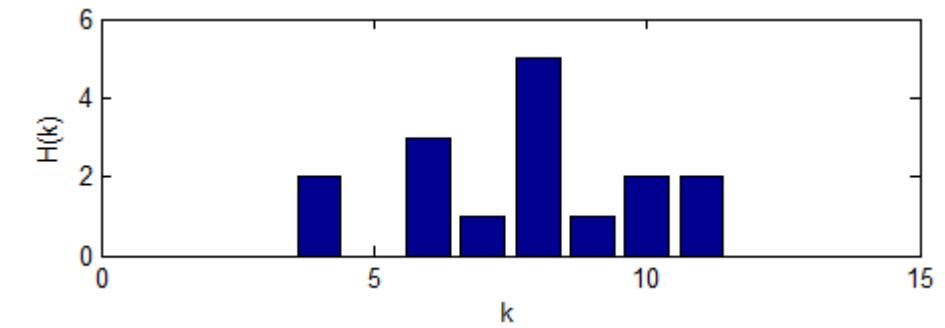
Intensity Transformations Example: Full-Scale Contrast Stretch



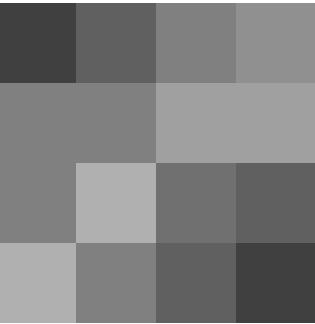
4	6	8	9
8	8	10	10
8	11	7	6
11	8	6	4

$$H(k) = \begin{matrix} 0 & 0 & 0 & 0 & 2 & 0 & 3 & 1 & 5 & 1 & 2 & 2 & 0 & 0 & 0 & 0 \\ k = 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix}$$

$r_{\min}=4$, $r_{\max}=11$,
#bits $B=4$ desired range $[0, 2^B-1]=[0,15]$



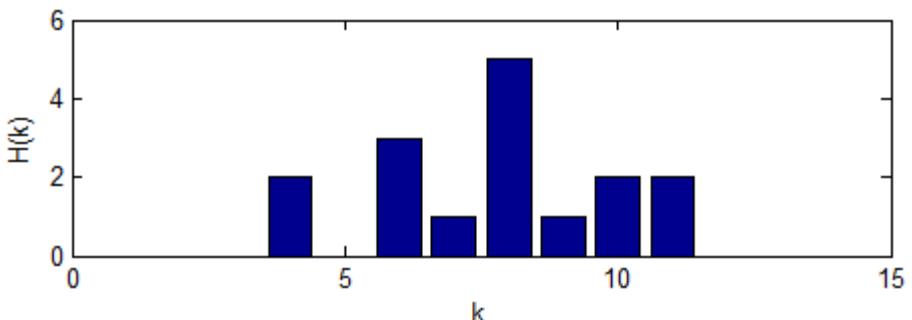
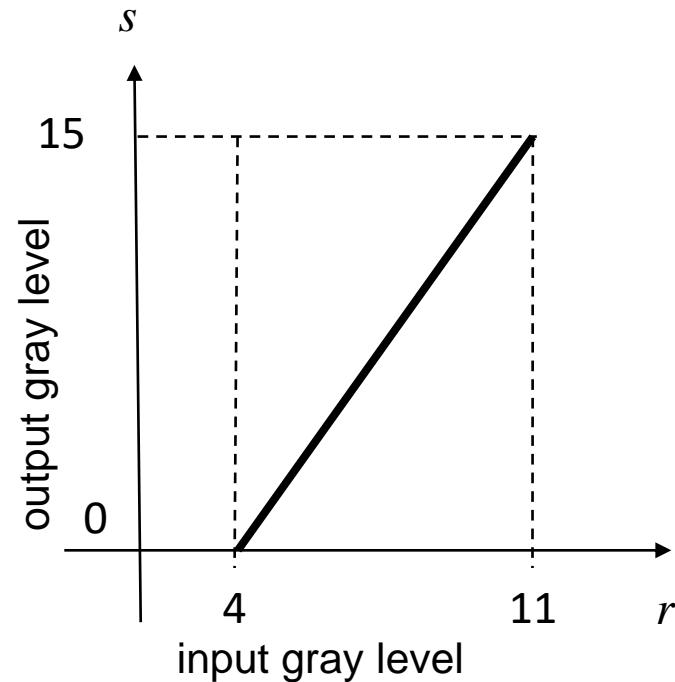
Intensity Transformations Example: Full-Scale Linear Contrast Stretch



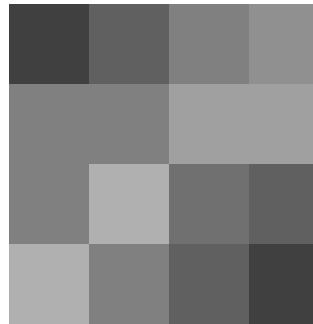
4	6	8	9
8	8	10	10
8	11	7	6
11	8	6	4

$$\begin{aligned} H(k) = & \quad 0 \ 0 \ 0 \ 0 \ 2 \ 0 \ 3 \ 1 \ 5 \ 1 \ 2 \ 2 \ 0 \ 0 \ 0 \ 0 \\ k = & \quad 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \end{aligned}$$

$r_{\min}=4$, $r_{\max}=11$,
#bits $B=4$ desired range $[0, 2^B-1]=[0,15]$



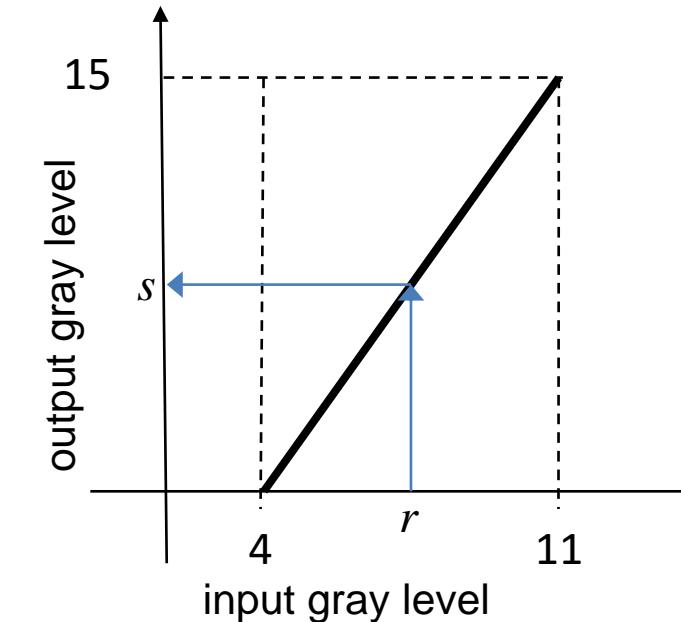
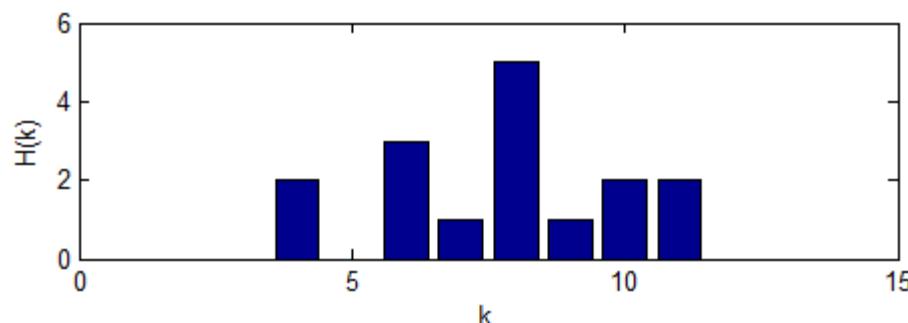
Intensity Transformations Example: Full-Scale Contrast Stretch



4	6	8	9
8	8	10	10
8	11	7	6
11	8	6	4

$$\begin{aligned} H(k) = & \quad 0 \ 0 \ 0 \ 0 \ 2 \ 0 \ 3 \ 1 \ 5 \ 1 \ 2 \ 2 \ 0 \ 0 \ 0 \ 0 \\ k = & \quad 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \end{aligned}$$

$r_{min}=4$, $r_{max}=11$,
#bits $B=4$ desired range $[0, 2^B-1]=[0,15]$

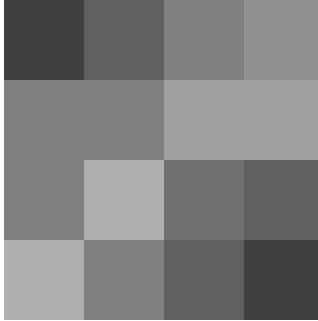


$$s = \text{round}\left((2^B - 1) \frac{r - r_{min}}{r_{max} - r_{min}}\right)$$

$$= \text{round}\left(15 \frac{r - 4}{11 - 4}\right)$$

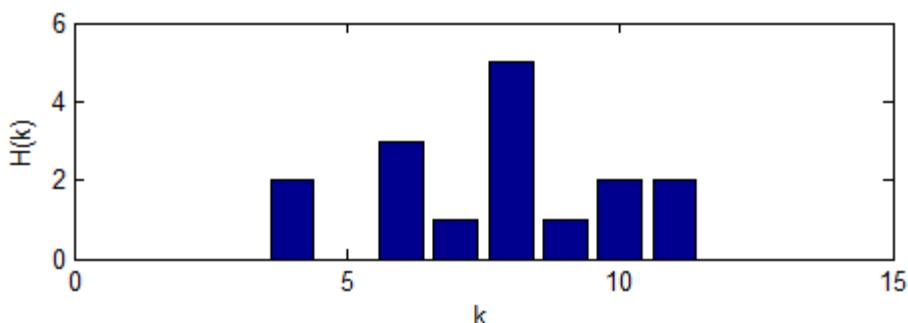
$$s = T(r) = \text{round}\left(\frac{15}{7}r - \frac{60}{7}\right)$$

Intensity Transformations Example: Full-Scale Contrast Stretch



4	6	8	9
8	8	10	10
8	11	7	6
11	8	6	4

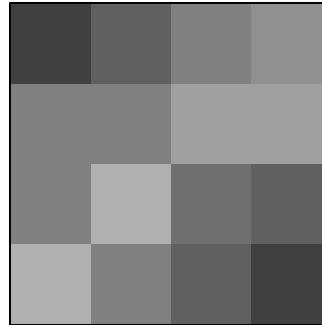
$$H(k) = \begin{matrix} 0 & 0 & 0 & 0 & 2 & 0 & 3 & 1 & 5 & 1 & 2 & 2 & 0 & 0 & 0 & 0 \\ k = \begin{matrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \end{matrix} \end{matrix}$$



$r \rightarrow s$

$$\begin{aligned} 4 &\rightarrow \text{round} \left(\frac{15}{7} 4 - \frac{60}{7} \right) = 0 \\ 6 &\rightarrow \text{round} \left(\frac{15}{7} 6 - \frac{60}{7} \right) = 4 \\ 7 &\rightarrow \text{round} \left(\frac{15}{7} 7 - \frac{60}{7} \right) = 6 \\ 8 &\rightarrow \text{round} \left(\frac{15}{7} 8 - \frac{60}{7} \right) = 9 \\ 9 &\rightarrow \text{round} \left(\frac{15}{7} 9 - \frac{60}{7} \right) = 11 \\ 10 &\rightarrow \text{round} \left(\frac{15}{7} 10 - \frac{60}{7} \right) = 13 \\ 11 &\rightarrow \text{round} \left(\frac{15}{7} 11 - \frac{60}{7} \right) = 15 \end{aligned}$$

Intensity Transformations Example: Full-Scale Contrast Stretch

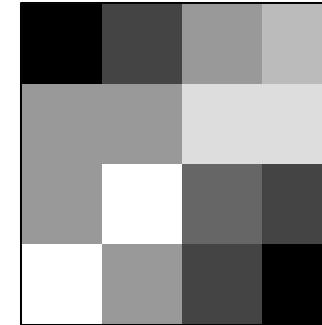


4	6	8	9
8	8	10	10
8	11	7	6
11	8	6	4

$$\begin{aligned} H(k) = & \quad 0 \ 0 \ 0 \ 0 \ 2 \ 0 \ 3 \ 1 \ 5 \ 1 \ 2 \ 2 \ 0 \ 0 \ 0 \ 0 \\ k = & \quad 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \end{aligned}$$

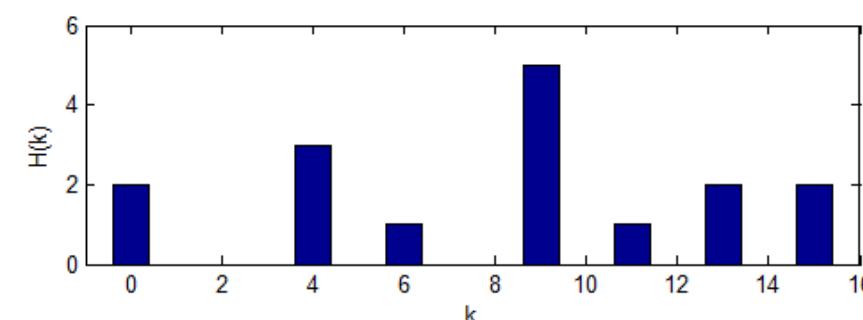
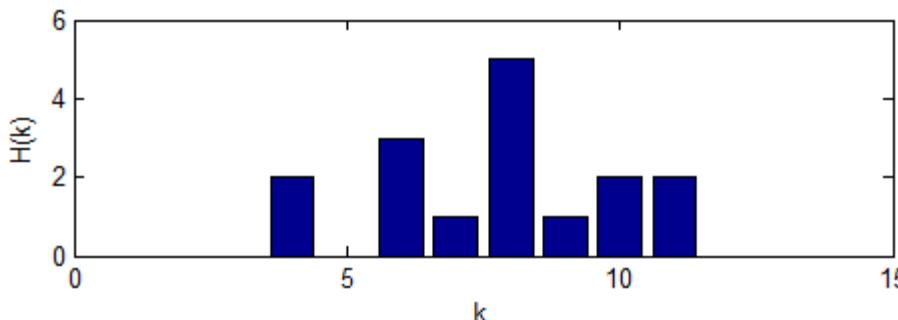
$r \rightarrow s$

 $4 \rightarrow 0$
 $6 \rightarrow 4$
 $7 \rightarrow 6$
 $8 \rightarrow 9$
 $9 \rightarrow 11$
 $10 \rightarrow 13$
 $11 \rightarrow 15$



0	4	9	11
9	9	13	13
9	15	6	4
15	9	4	0

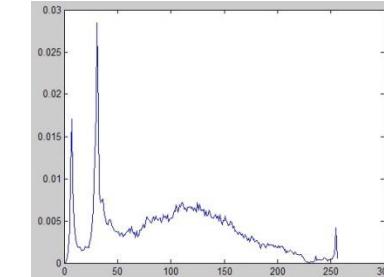
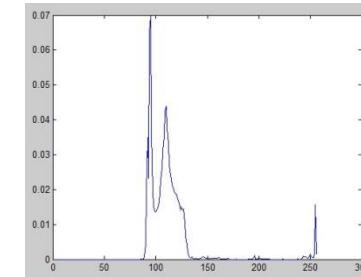
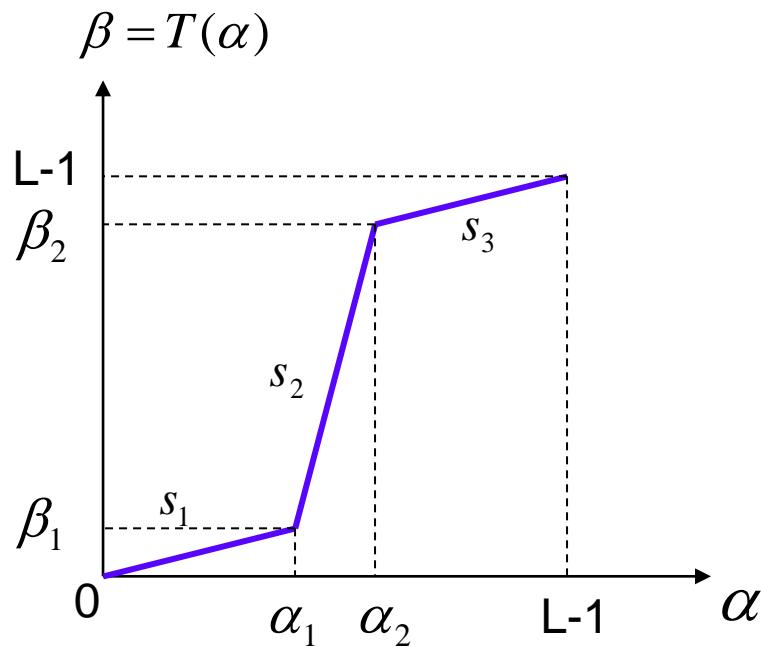
$$\begin{aligned} H(k) = & \quad 2 \ 0 \ 0 \ 0 \ 3 \ 0 \ 1 \ 0 \ 0 \ 5 \ 0 \ 1 \ 0 \ 2 \ 0 \ 2 \\ k = & \quad 0 \ 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12 \ 13 \ 14 \ 15 \end{aligned}$$



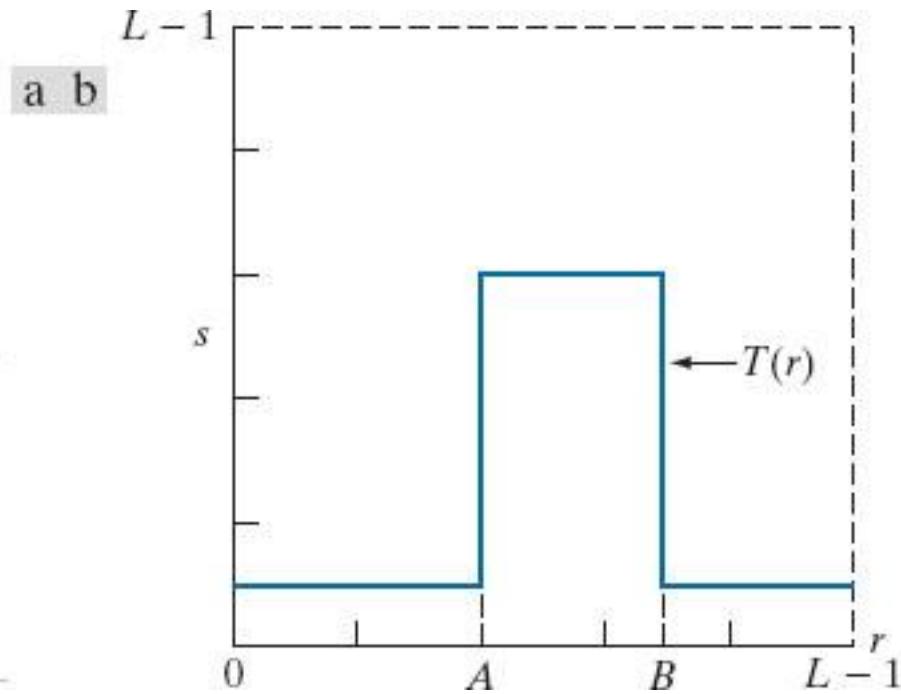
Contrast Stretching: Does not always need to be full scale

Stretch the over-concentrated gray-levels

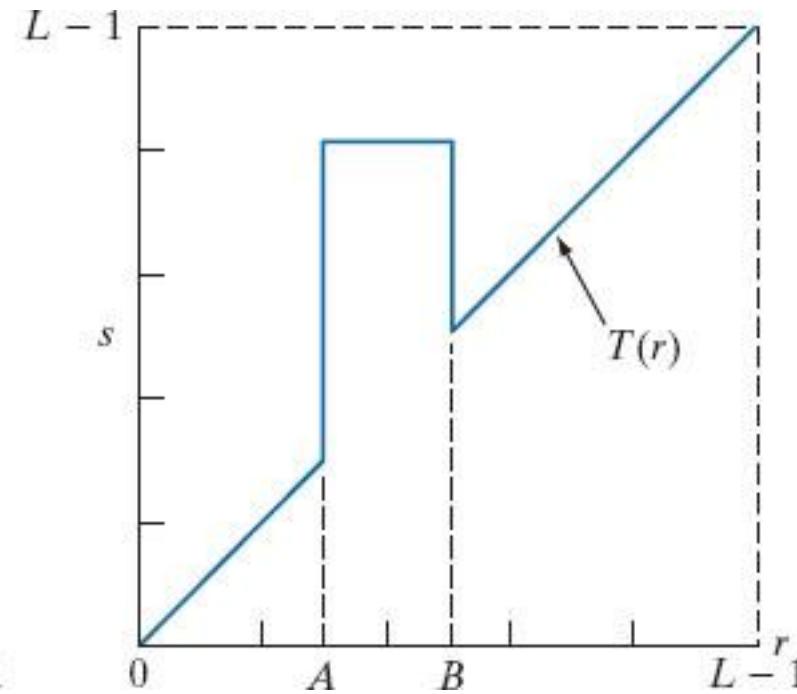
Piece-wise linear function, where the slope in the stretching region is greater than 1.



Intensity Transformations



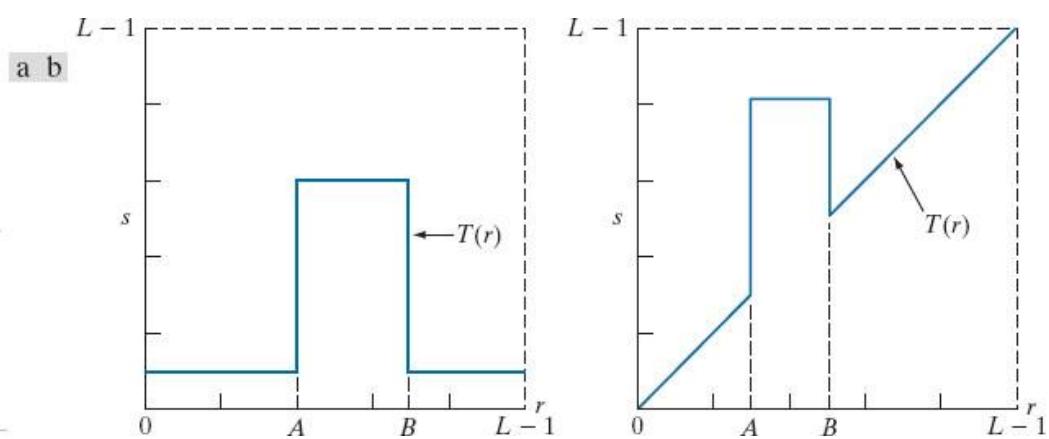
(a) This transformation function highlights range $[A,B]$ and reduces all other intensities to a lower level.



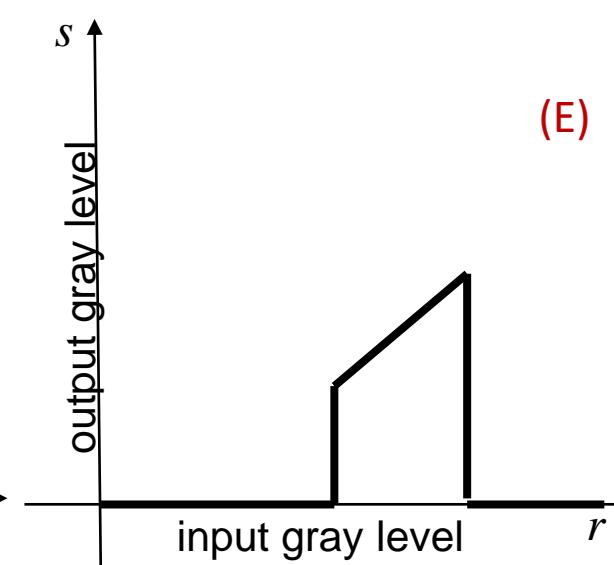
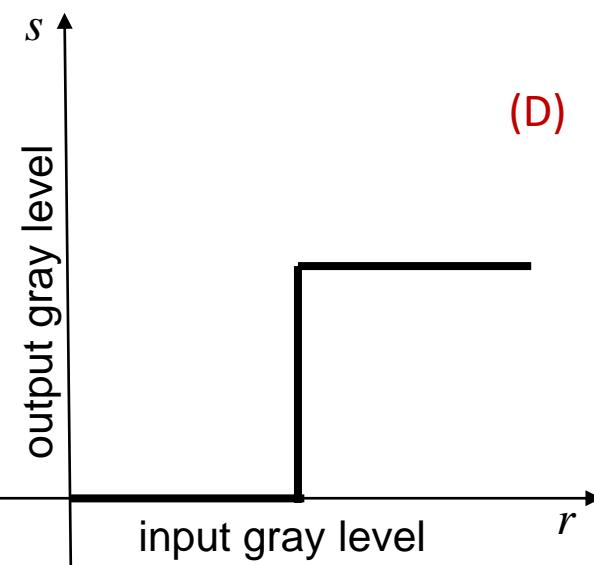
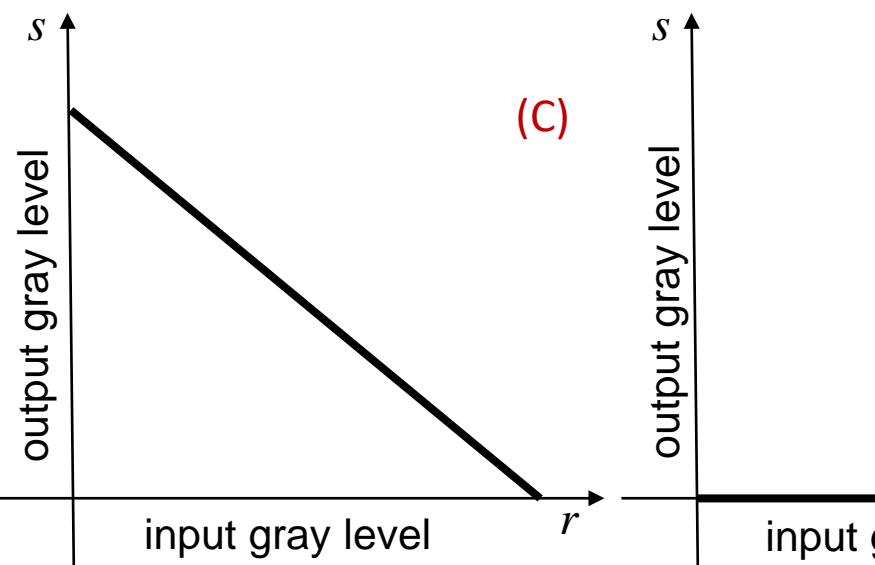
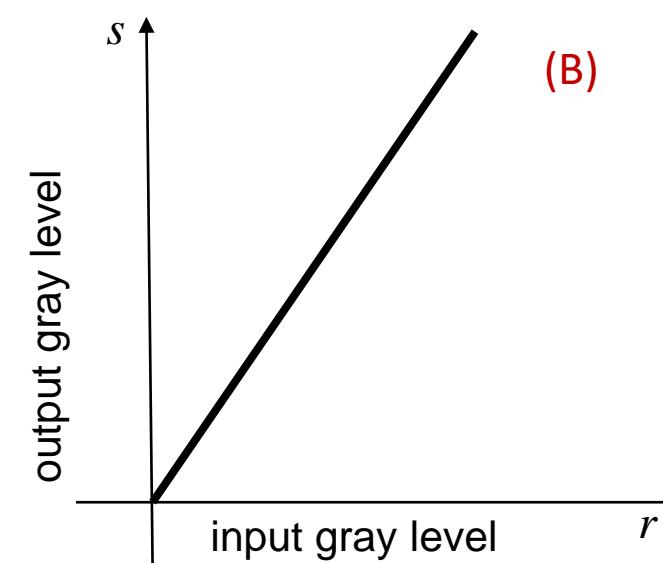
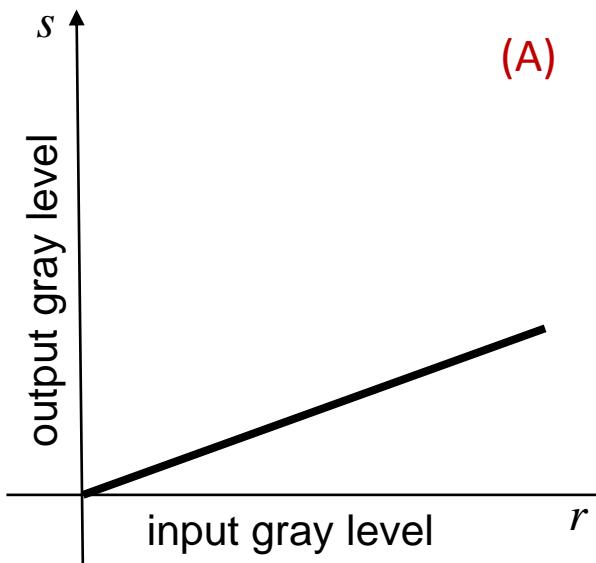
(b) This function highlights range $[A,B]$ and leaves other intensities unchanged.

Slicing Transformation

- (a) Aortic angiogram.
- (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale.
- (c) Result of using the transformation in Fig. 3.11(b), with the selected range set near black, so that the grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)

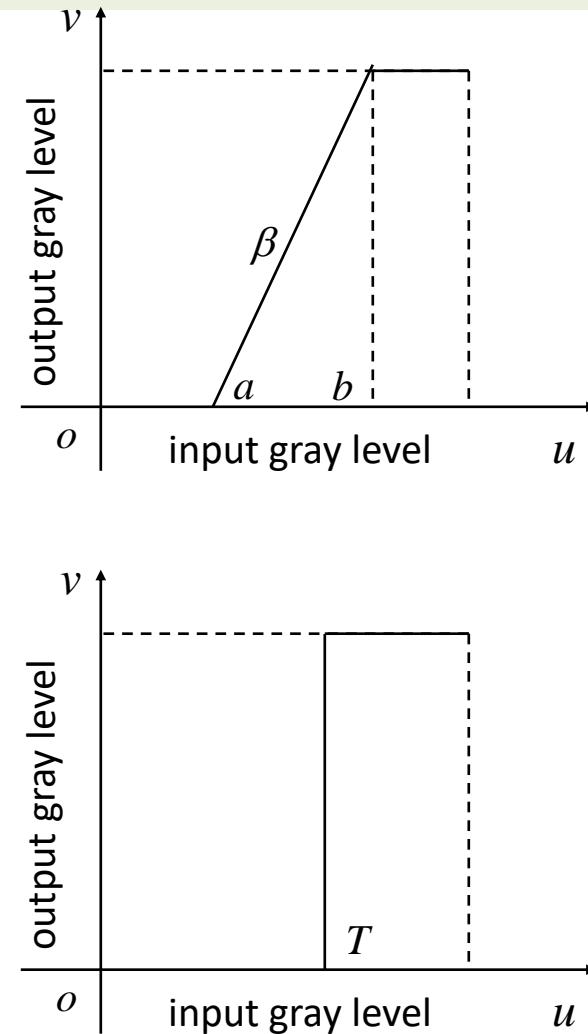


What do these transforms do?

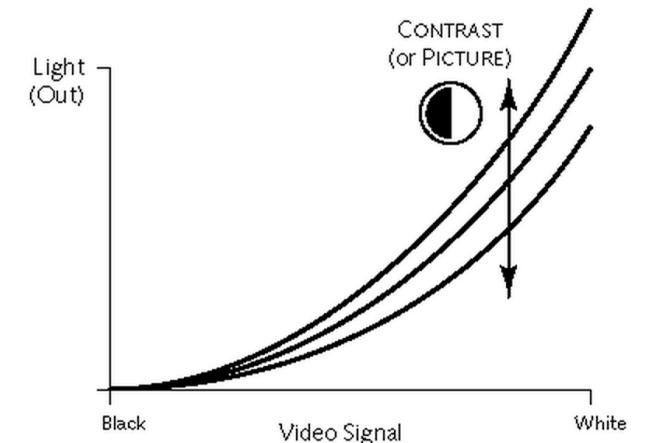
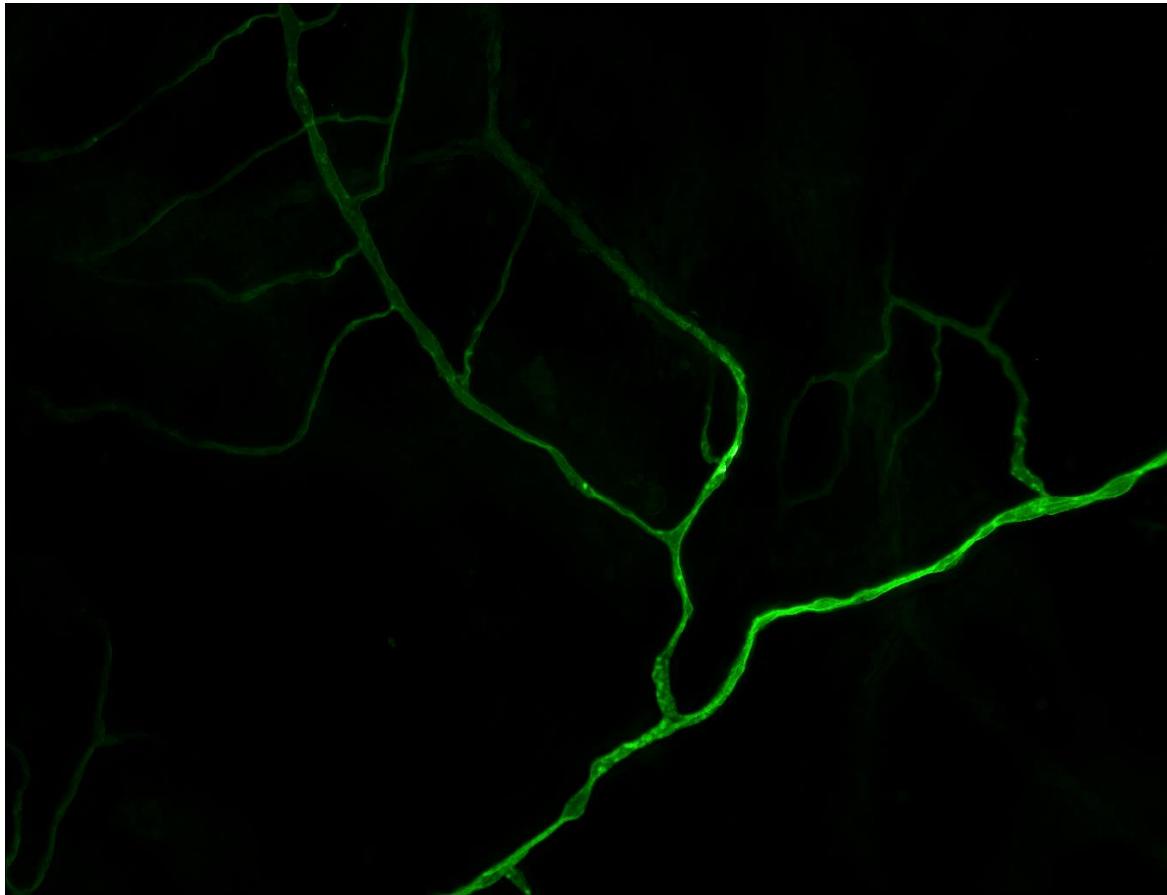


Clipping & Thresholding

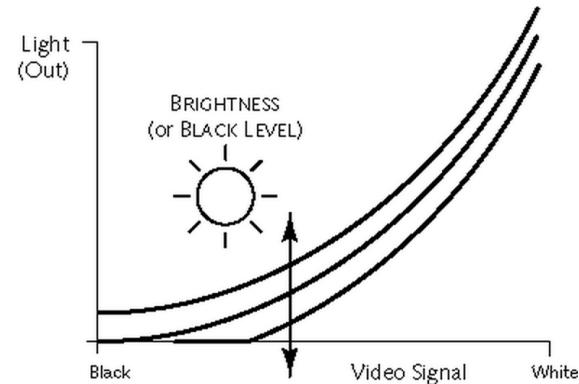
- Clipping
 - Special case of contrast stretching with $\alpha = \gamma = 0$
 - Useful for noise reduction when interested signal mostly lie in range $[a,b]$
- Thresholding
 - Special case of clipping with $a = b = T$
 - Useful for binarization of scanned binary images
 - documents, signatures, fingerprints



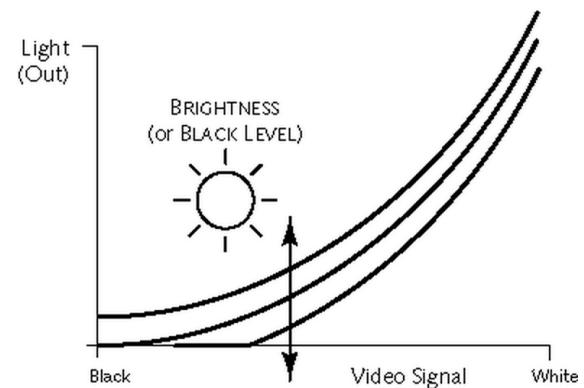
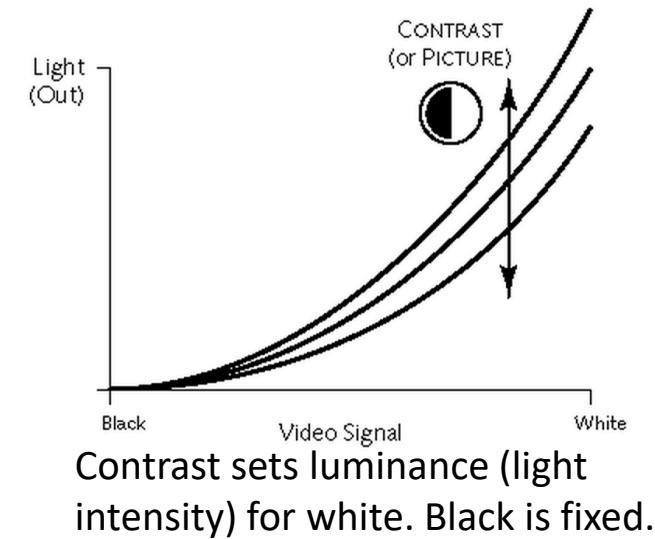
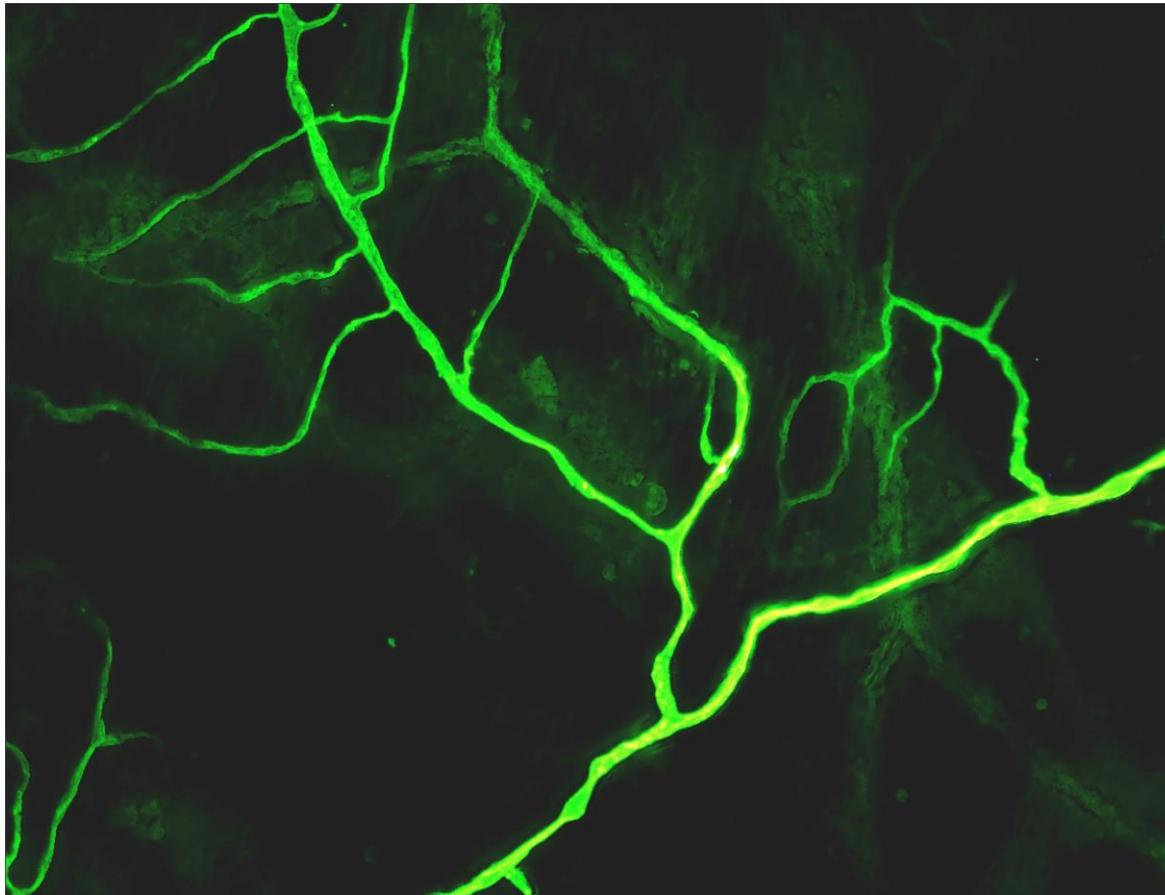
Brightness, Contrast, Sharpen



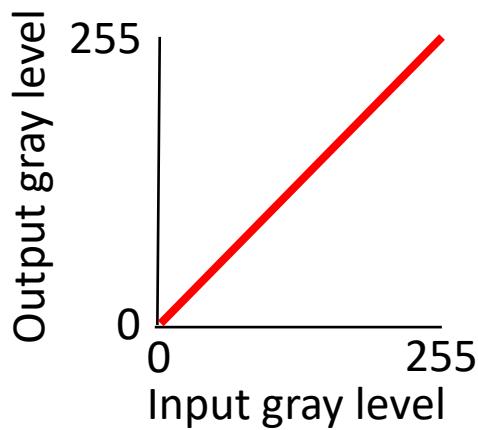
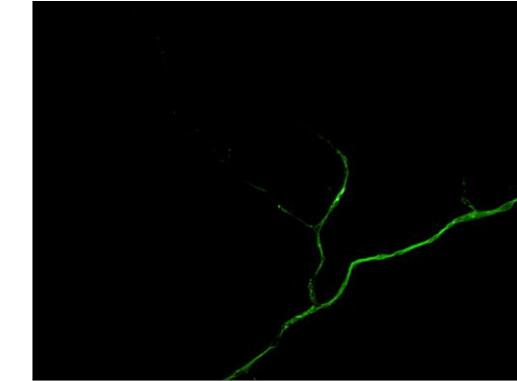
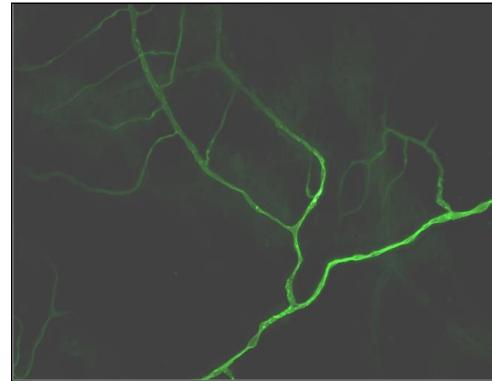
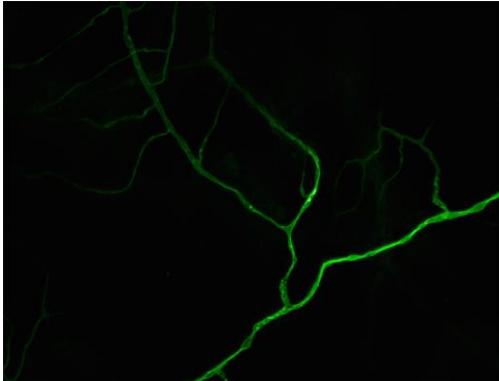
Contrast sets luminance (light intensity) for white. Black is fixed.



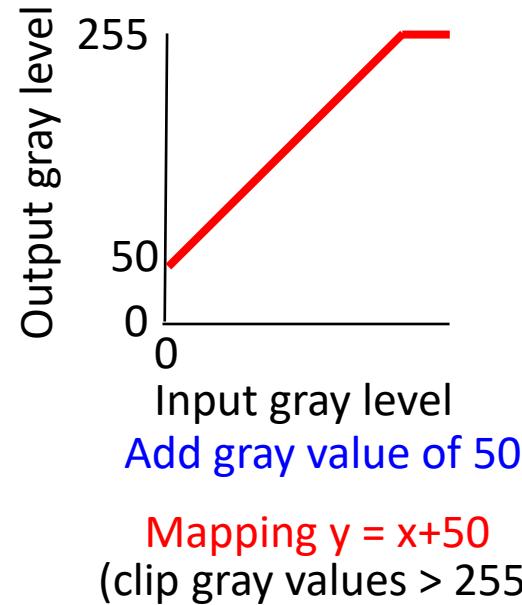
Brightness, Contrast, Sharpen



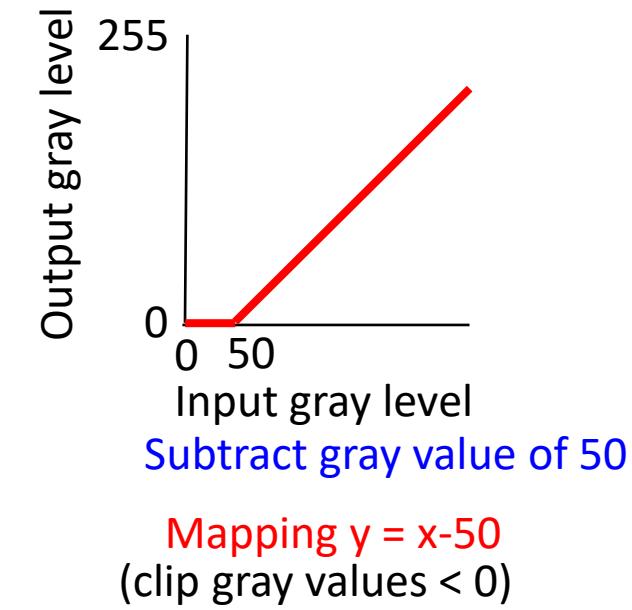
Brightness ONLY – Change the Offset – Add or Subtract Gray value (Gray level)



Mapping $y = x$

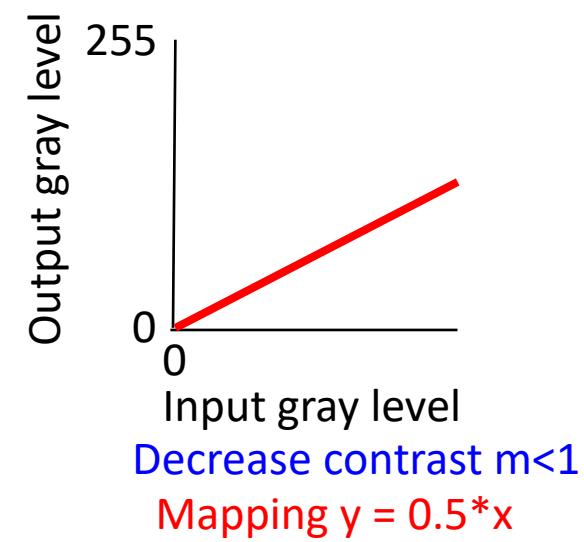
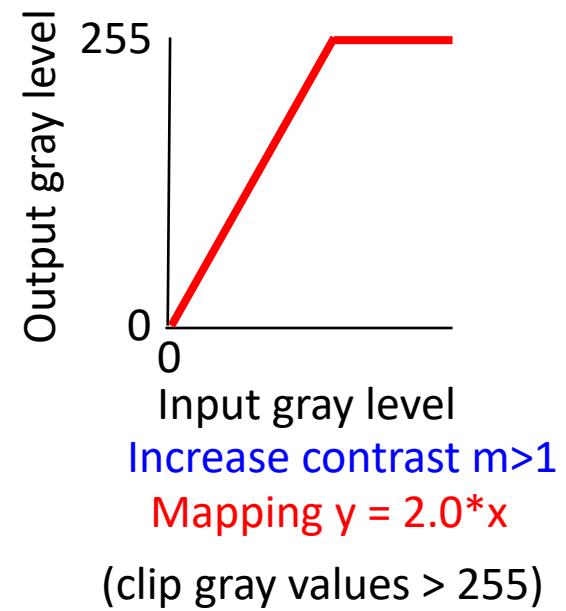
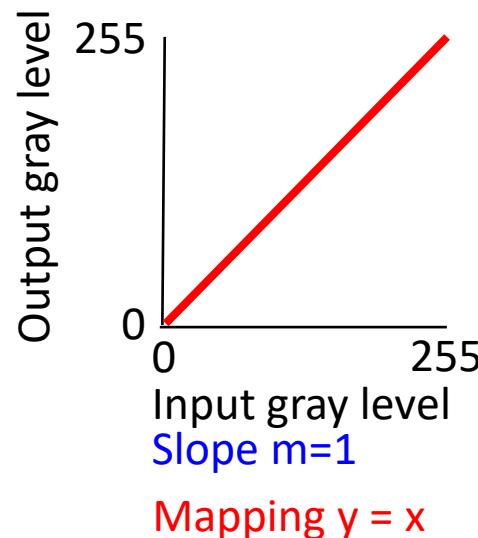
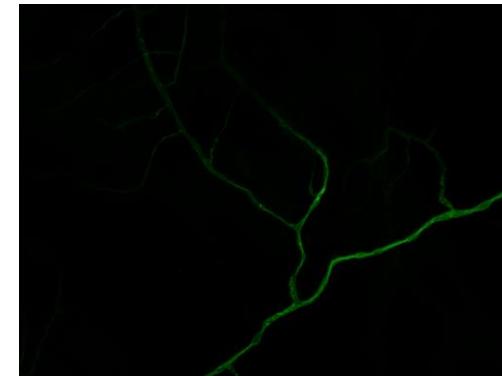
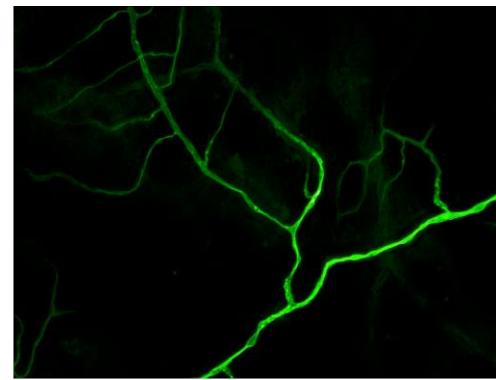
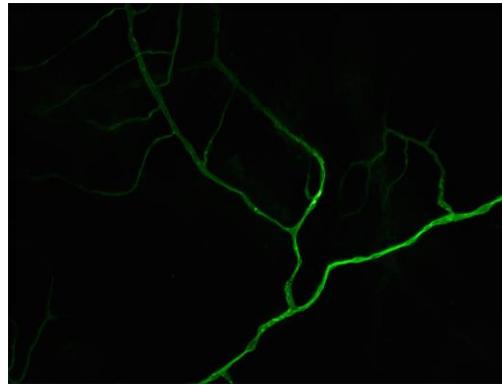


Mapping $y = x+50$
(clip gray values > 255)

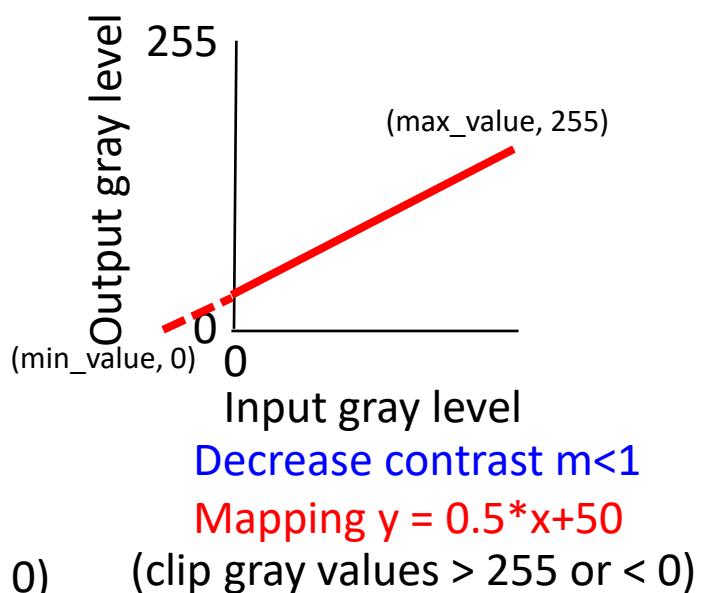
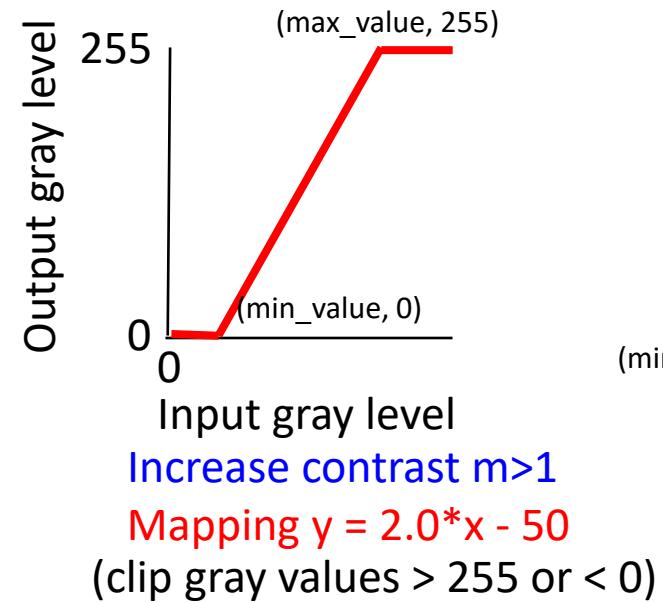
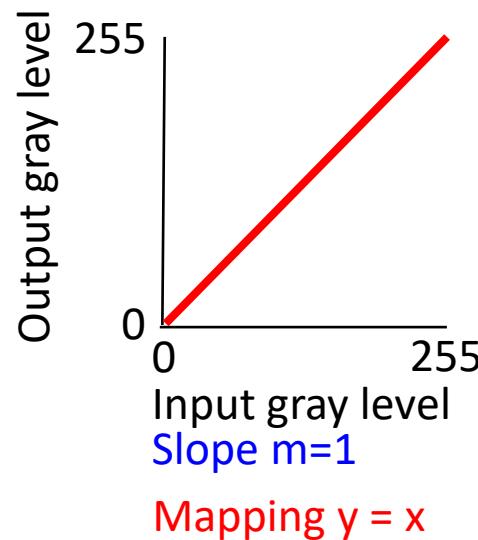


Mapping $y = x-50$
(clip gray values < 0)

Contrast ONLY – Change the Slope of the IO Mapping



Brightness & Contrast – Change the Offset and Slope of the IO Mapping



Python for Image Processing and Vision

Imad Toubal, Dr. Filiz Bunyak
CS/ECE 8690 Computer
Vision



Outline

1. Why Python?
2. Installation (Anaconda)
3. Packages (Numpy, OpenCV, Matplotlib, ...etc)

Why Python?

- Easy to learn and use
- Large community
- Easy-to-use packages
- I can help you better!



Installation

The preferred method is to install Anaconda, which is a Python distribution that enables you to set up different disposable environments for different projects. To install, go to <https://www.anaconda.com/distribution/>

The screenshot shows the Anaconda website homepage. At the top, there is a navigation bar with links: Products (underlined), Pricing, Solutions, Resources, Partners, Blog, Company, and a Contact Sales button. Below the navigation bar, a green banner states "Individual Edition is now ANACONDA DISTRIBUTION". The main heading "ANACONDA DISTRIBUTION" is displayed in large green letters. Below it, the text "The world's most popular open-source Python distribution platform" is written. To the right, a callout box titled "Anaconda Distribution" features a "Download" button with an Apple icon, indicating availability for macOS. Below the download button, it says "For MacOS" and "Python 3.9 • 64-Bit Graphical Installer • 688 MB".

Setting up an environment

To create a new environment, open a terminal and type:

```
$ conda create -n myenv python=3.7
```

You'll then need to activate the environment:

```
$ conda activate myenv
```

Installing Packages

We can install packages using:

```
$ pip install numpy scipy matplotlib scikit-image opencv-python
```

or:

```
$ conda install numpy scipy matplotlib scikit-image opencv
```

Integrated development environment

Options include:

- Jupyter Notebooks
- Visual Studio Code
- PyCharm
- Spyder
- ...etc.

Test your installation

```
# Importing packages
import numpy as np
import matplotlib.pyplot as plt
import cv2

# Reading the image using OpenCV and
# converting it to grayscale using numpy
img = cv2.imread('lena.png')
img = np.mean(img, axis=2)

# Displaying the image using matplotlib
plt.imshow(img, cmap='gray')
plt.show()
```





Arrays in Numpy

```
>>> x = np.array([[1, 2, 3],  
...                 [4, 5, 6]])  
>>> y = np.ones((2, 3)) # 2x3 matrix of ones  
>>> y  
array([[1., 1., 1.],  
       [1., 1., 1.]])  
>>> w = np.random.rand(2, 3) # 2x3 matrix of random numbers  
>>> w  
array([[0.123, 0.456, 0.789],  
       [0.987, 0.654, 0.321]])  
>>> v = np.zeros(5) # 1D array of zeros  
>>> v  
array([0., 0., 0., 0., 0.])
```

Indexing

```
>>> x = np.array([[1, 2, 3],  
...                 [4, 5, 6],  
...                 [7, 8, 9]])  
>>> x[0] # First row  
array([1, 2, 3])  
>>> x[0, 1] # First row, second column  
2  
>>> x[:, 0] # First column, all rows  
array([1, 4, 7])  
>>> x[1:3, 1:3] # Second row and column, all rows and columns  
array([[5, 6],  
       [8, 9]])
```

Basic operations

```
>>> x + z # Element-wise addition  
array([[2., 3., 4.],  
       [5., 6., 7.]])  
>>> x * z # Element-wise multiplication  
array([[1., 2., 3.],  
       [4., 5., 6.]])  
>>> x @ z # Matrix multiplication  
array([[14, 32],  
       [32, 77]])
```

The `@` operator is used for matrix multiplication. The `np.dot` function can also be used.

Basic types

```
>>> x = np.array([1, 2, 3])
>>> x.dtype # Data type
dtype('int64')
>>> x = np.array([1, 2, 3], dtype=np.float32)
>>> x.dtype
dtype('float32')
>>> y = x.astype(np.uint8) # Convert to unsigned 8-bit integer
>>> y.dtype
dtype('uint8')
```

`int64` is the default integer type in Numpy. `float32` is the default floating point type. You can use `astype` to convert between types.

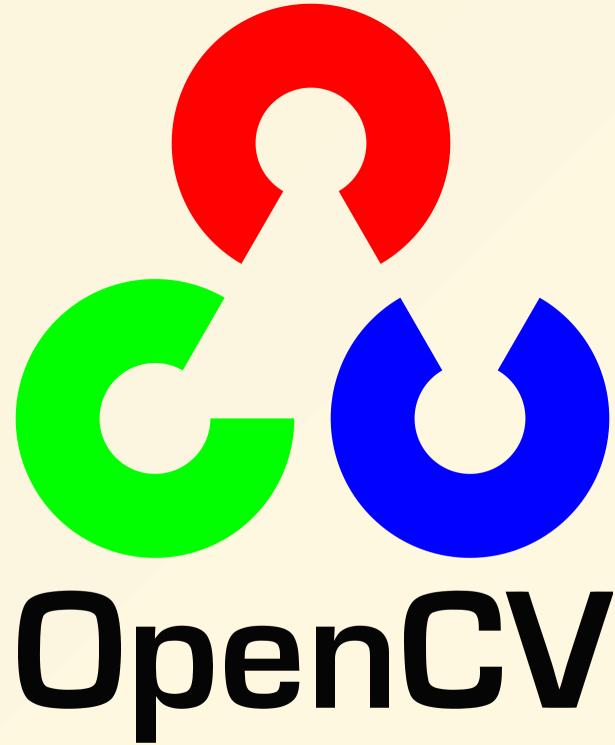
Basic functions

```
>>> x = np.array([[1, 2, 3],  
...                 [4, 5, 6],  
...                 [7, 8, 9]])  
>>> x.sum() # Sum of all elements  
45  
>>> x.sum(axis=0) # Sum of columns  
array([12, 15, 18])  
>>> x.sum(axis=1) # Sum of rows  
array([ 6, 15, 24])  
>>> x.mean() # Mean of all elements  
5.0
```

`mean` and `sum` work the same way (they take the same arguments).

Linear algebra

```
>>> x = np.array([[1, 2, 3],  
...                 [4, 5, 6],  
...                 [7, 8, 9]])  
>>> np.linalg.inv(x) # Inverse of x  
array([[-0.333,  0.667, -0.333],  
       [ 0.667, -1.333,  0.667],  
       [-0.333,  0.667, -0.333]])  
>>> np.linalg.det(x) # Determinant of x  
6.661338147750939e-16  
>>> np.linalg.eig(x) # Eigenvalues and eigenvectors of x  
(array([ 1.611e+01, -1.116e+00, -1.303e-15]),  
 array([[ -0.231, -0.785, -0.577],  
        [ -0.524, -0.343,  0.577],  
        [ -0.818,  0.517,  0.577]]))
```



Reading and writing images

```
>>> import cv2
>>> # read image
>>> img = cv2.imread('lena.png')
>>> img
array([[[  0,  64, 128],
       [  0,  64, 128],
       ...,
       [  0,  64, 128],
       [  0,  64, 128]]], dtype=uint8)
>>> img.shape
(512, 512, 3)
>>> # write image
>>> cv2.imwrite('lena_copy.png', img)
```



Color

```
>>> import cv2
>>> import numpy as np
>>> img = cv2.imread('lena.png')
>>> # Convert to grayscale
>>> img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
>>> # Save the grayscale image
>>> cv2.imwrite('lena-gray.png', img)
```

Note that OpenCV uses BGR (Blue, Green Red) color space by default. Many other libraries use RGB (Red, Green, Blue).



Resizing

```
>>> import cv2  
>>> img = cv2.imread('lena.png')  
>>> # Resize to 256x256  
>>> img = cv2.resize(img, (256, 256))
```

or using `rescale`:

```
>>> img = cv2.rescale(img, 0.5)
```

`rescale` takes the ratio of the new size to the old size as an argument.

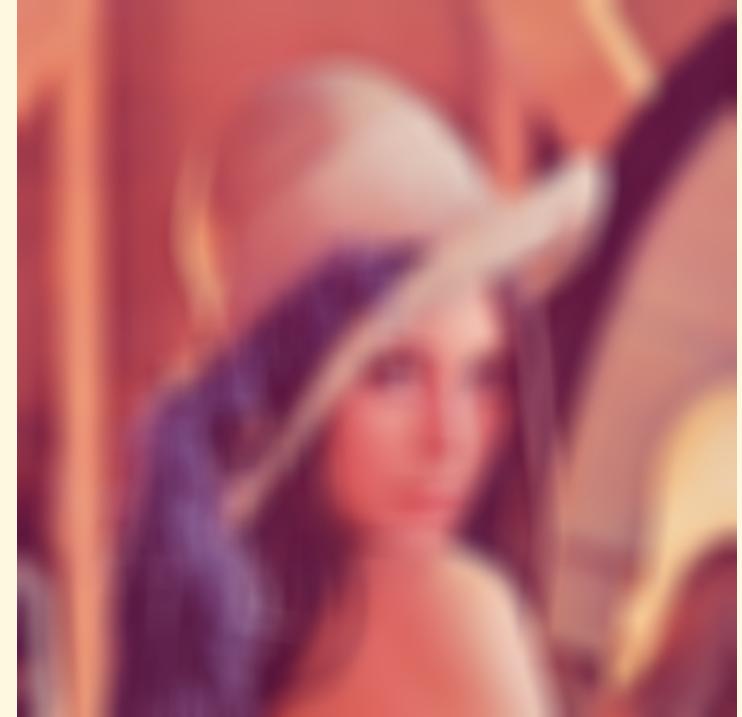


Filters

```
>>> import cv2  
>>> img = cv2.imread('lenna.png')  
>>> # Blur  
>>> img = cv2.blur(img, (25, 25))  
>>> cv2.imwrite('lenna-blur.png', img)
```

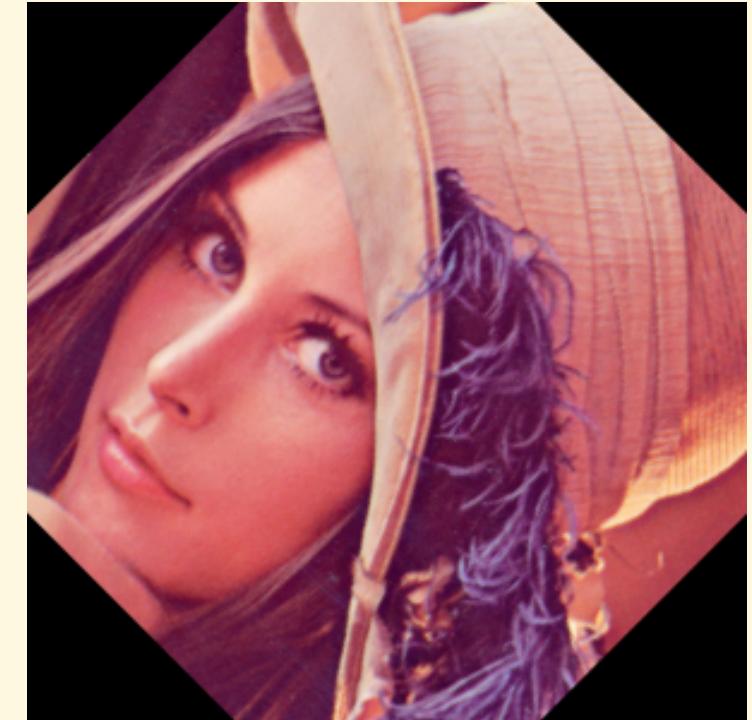
You can also apply arbitrary filters using
filter2D:

```
>>> kernel = np.ones((5, 5), np.float32) / 25  
>>> img = cv2.filter2D(img, -1, kernel)
```



Advanced

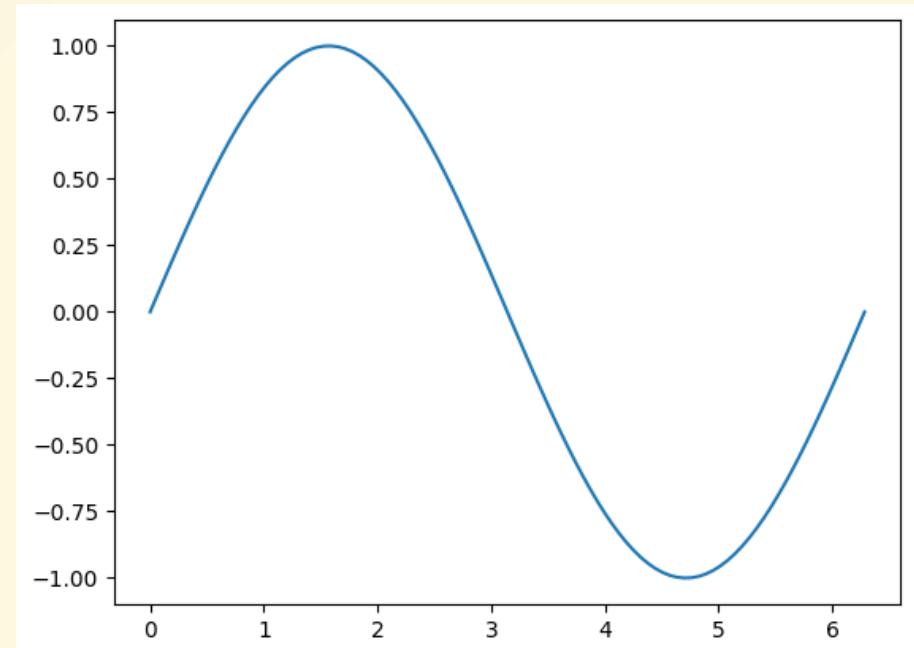
```
>>> import cv2
>>> img = cv2.imread('lenna.png')
>>> # Crop
>>> img = img[100:400, 100:400]
>>> # Rotate
>>> M = cv2.getRotationMatrix2D((150, 150), 45, 1)
>>> img = cv2.warpAffine(img, M, (300, 300))
>>> # Flip
>>> img = cv2.flip(img, 1)
>>> # Save
>>> cv2.imwrite('lenna-advanced.png', img)
```



matplotlib

Plotting

```
>>> import matplotlib.pyplot as plt  
>>> import numpy as np  
>>> # Creating the data  
>>> x = np.linspace(0, 2 * np.pi, 100)  
>>> y = np.sin(x)  
>>> # Plotting  
>>> plt.plot(x, y)  
>>> plt.show()
```

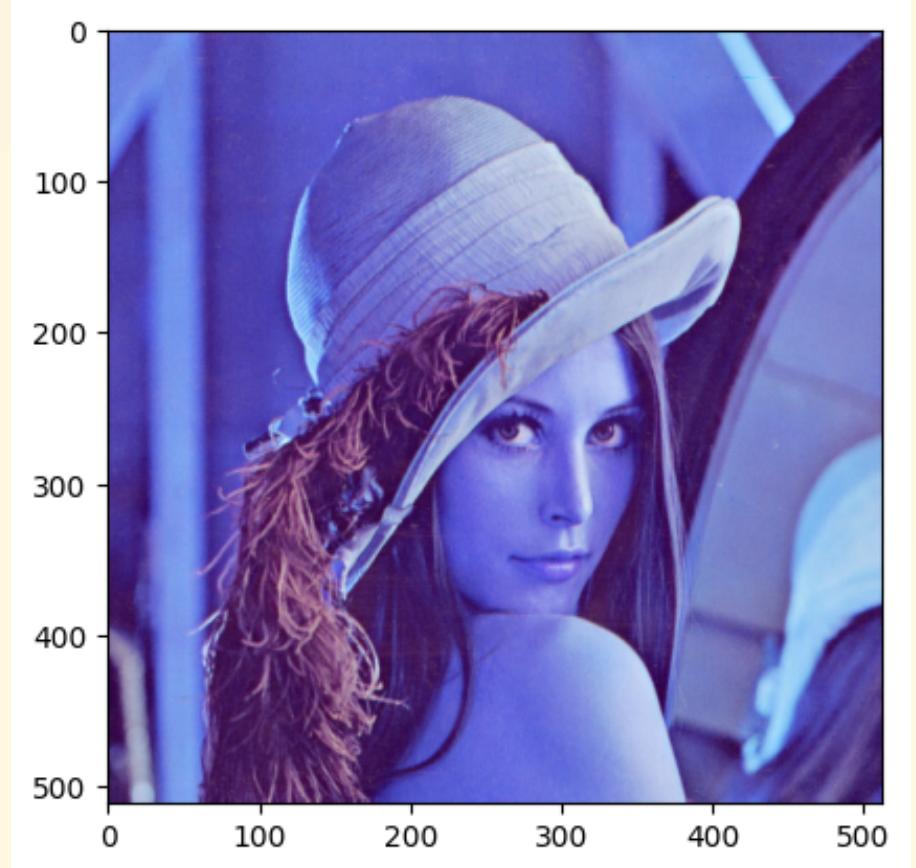


If you're using Jupyter, you can use
`%matplotlib inline` to display the
graph in the notebook.

The `imshow` function

```
>>> import matplotlib.pyplot as plt  
>>> import cv2  
>>> img = cv2.imread('lenna.png')  
>>> plt.imshow(img)
```

Why does lenna look blue?

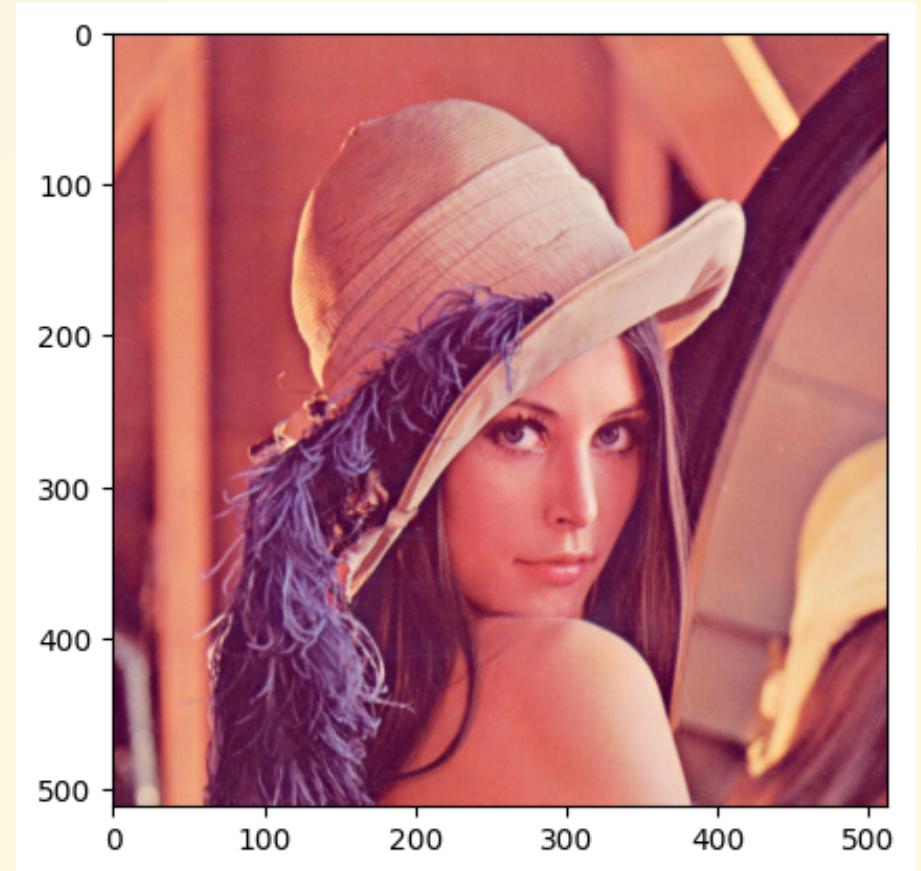


The `imshow` function

```
>>> import matplotlib.pyplot as plt  
>>> import cv2  
>>> img = cv2.imread('lenna.png')  
>>> plt.imshow(img)
```

Remember that OpenCV uses BGR by default. We can fix this by converting the image to RGB:

```
>>> img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  
>>> plt.imshow(img)
```



References

- [NumPy](#)
- [OpenCV](#)
- [Matplotlib](#)
- [jupyter](#)

Office hours

- TuThu 10:00-11:00 email: itdfh@umsystem.edu

Thanks! Questions?

Local Image Filtering

CS/ECE 8690 Computer Vision

Instructor: Filiz Bunyak Ersoy bunyak@missouri.edu
TA: Imad Toubal (CS PhD): itdfh@mail.missouri.edu

Image Processing

Three classes of operations that are most commonly used:

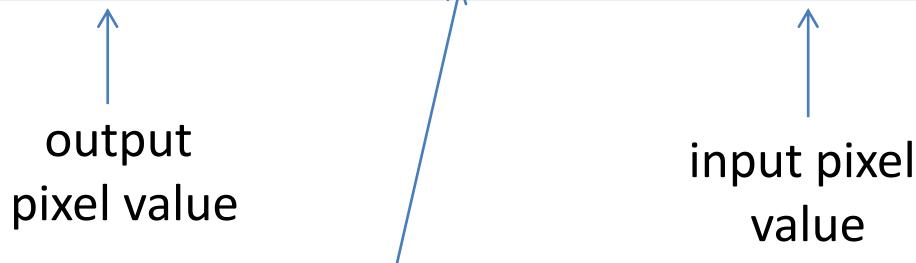
1. Intensity transformation (Point operators/processes),
2. Local image filtering,
 1. Linear Filtering
 2. Non-Linear Filtering
 3. Morphological Filtering
3. Geometrical transformation

Neighborhood Operations

Spatial Image Filtering

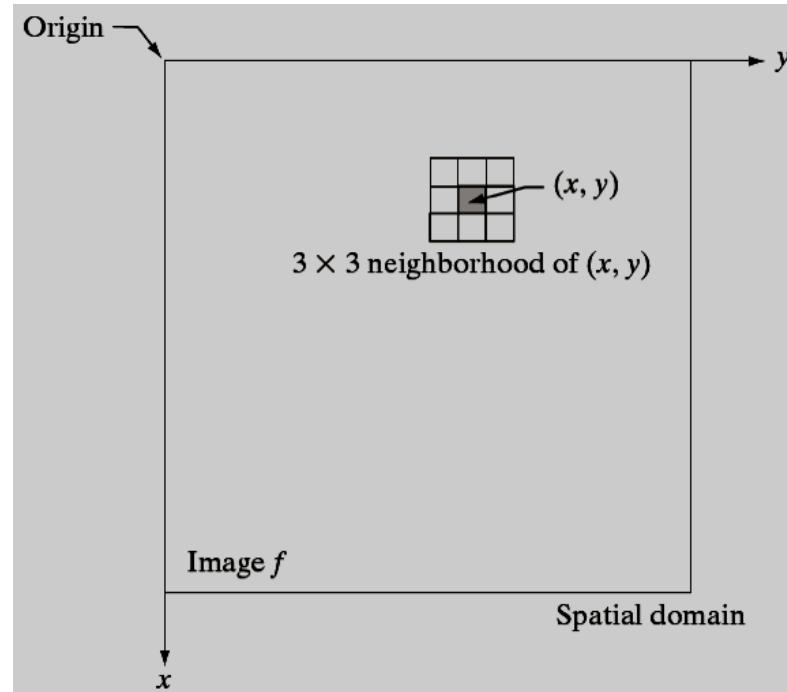
Spatial domain process:

$$g(x, y) = T_{N(x,y)}[f(x, y)]$$



For point operators neighborhood:

$$N(x, y) = (x, y)$$



Filtering Basics

Spatial domain process:

$$g(x, y) = T_{N(x,y)}[f(x, y)]$$

output pixel value Operator defined over a Neighborhood of point (x,y) input pixel value

Why filtering is important?

Filtering is needed to

- Enhance images (smoothing, noise removal etc.)
- Extract information (edges, texture etc.)
- Detect patterns

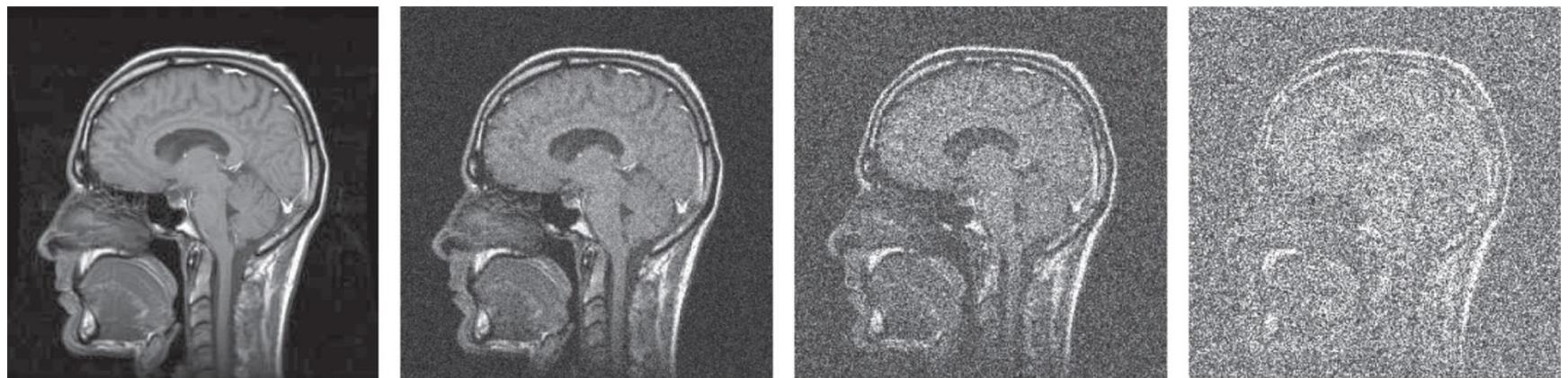
Filtering types:

- Spatial domain
- Frequency domain

Filtering types:

- Linear
- Non-linear

Noise



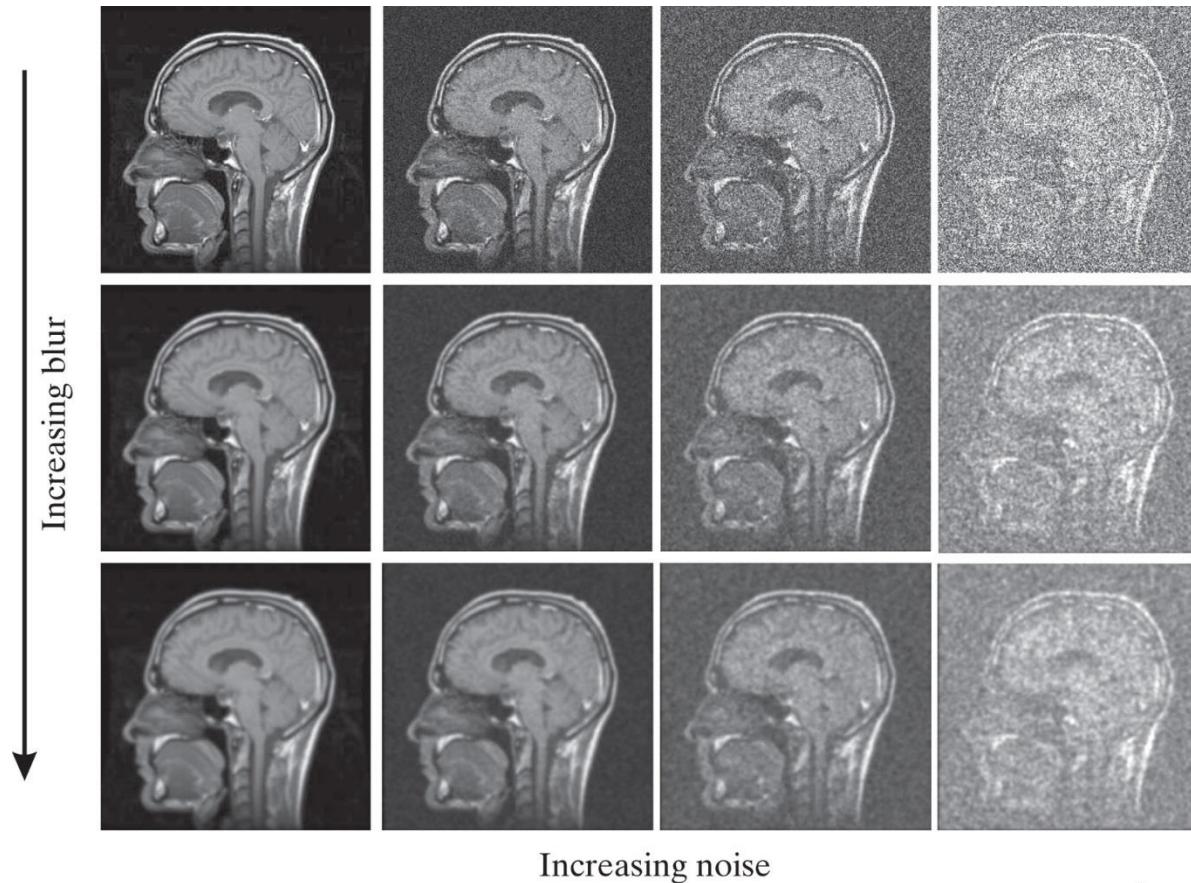
Increasing Noise



Copyright ©2015 Pearson Education, All Rights Reserved

Signal-to-Noise Ratio (SNR)

The output of a medical imaging system g is a random variable that consists of two components f (deterministic signal) and g (random noise).



Common types of noise

- **Salt and pepper noise:** random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution



Original



Salt and pepper noise



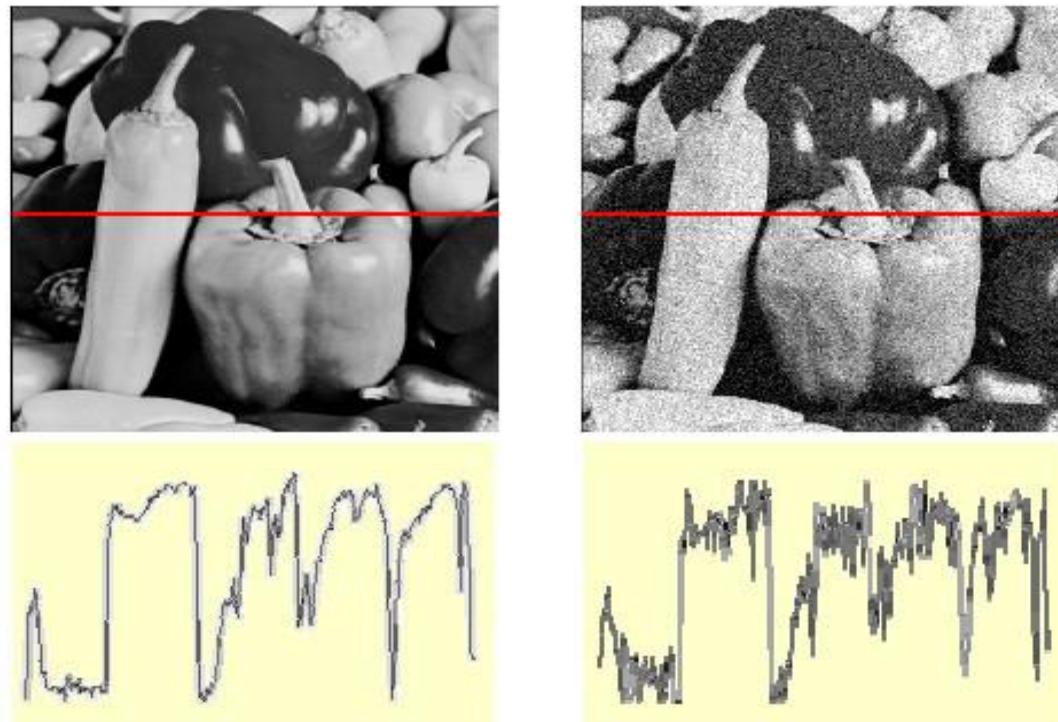
Impulse noise



Gaussian noise

Gaussian noise

Image



What is impact
of the sigma?

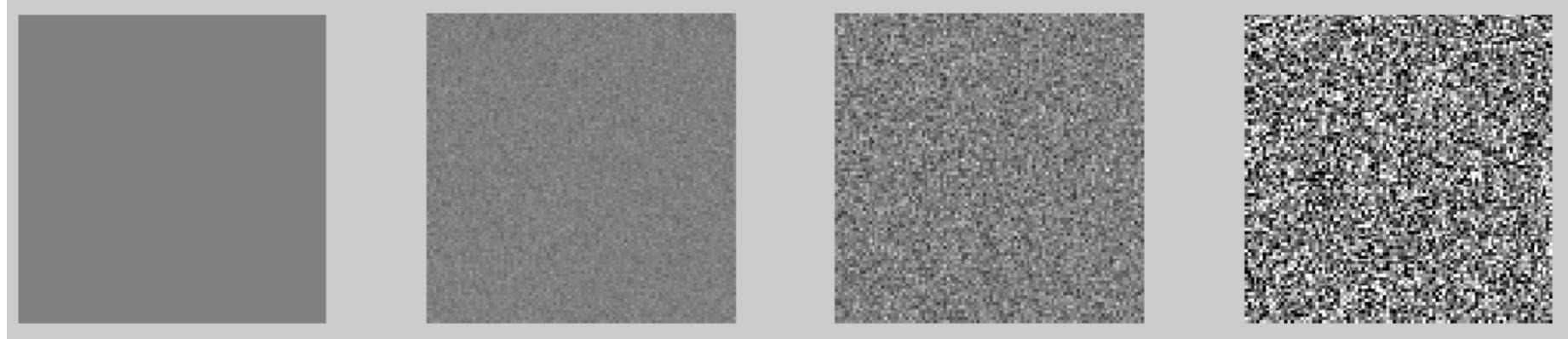
$$f(x, y) = \overbrace{\hat{f}(x, y)}^{\text{Ideal Image}} + \overbrace{\eta(x, y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
 $\eta(x, y) \sim \mathcal{N}(\mu, \sigma)$

```
>> noise = randn(size(im)).*sigma;  
>> output = im + noise;
```

The values that the noise can take on are Gaussian-distributed. Principal sources of Gaussian noise arise during acquisition e.g. [sensor noise](#) caused by poor illumination and/or high temperature, and/or transmission e.g. [electronic circuit noise](#).

Gaussian Noise

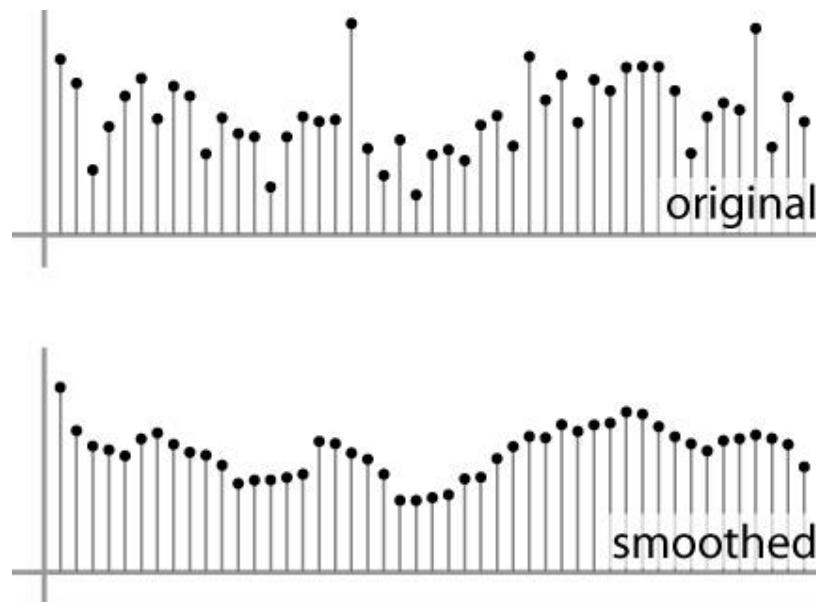


```
I=0.5*ones(100,100);  
I2=imnoise(I,'gaussian',0,0.001);  
I3=imnoise(I,'gaussian',0,0.01);  
I4=imnoise(I,'gaussian',0,0.1);
```

How can we “smooth” away noise in an image?

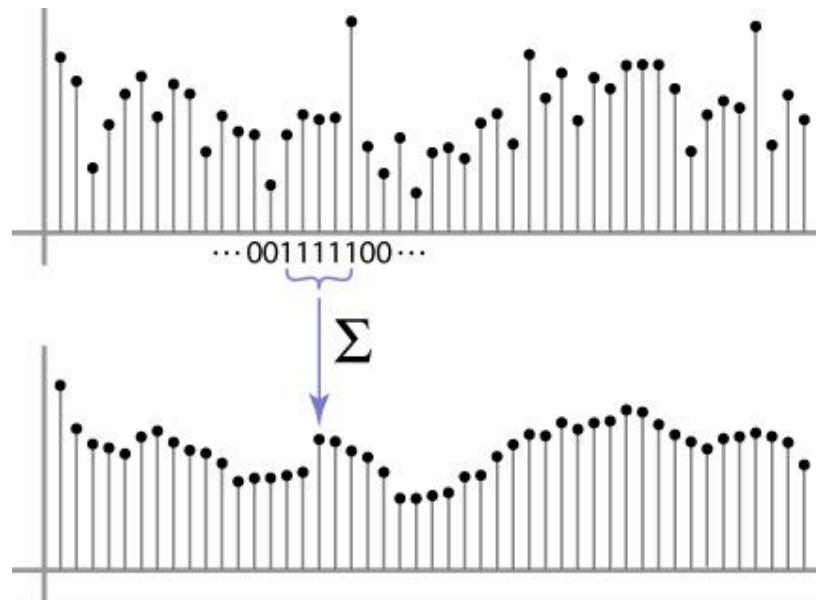
First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:



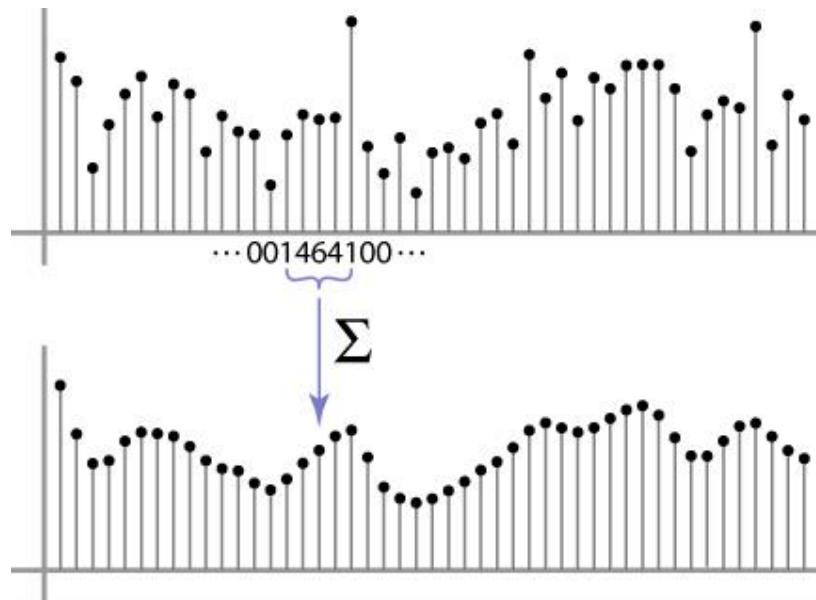
Weighted Moving Average

- Can add weights to our moving average
- *Weights* [1, 1, 1, 1, 1] / 5



Weighted Moving Average

- Non-uniform weights $[1, 4, 6, 4, 1] / 16$



Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

			0							

Moving Average In 2D

$$F[x, y]$$

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

$$G[x, y]$$

0	10									

Moving Average In 2D

$$F[x, y]$$

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

$$G[x, y]$$

			0	10	20					

Moving Average In 2D

$F[x, y]$

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

$G[x, y]$

Moving Average In 2D

$$F[x, y]$$

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

$$G[x, y]$$

Moving Average In 2D

$$F[x, y]$$

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

$$G[x, y]$$

	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		

Cross-correlation filtering

Let's write this down as an equation. Assume the averaging window is $(2k+1) \times (2k+1)$:

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

We can generalize this idea by allowing different weights for different neighboring pixels:

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

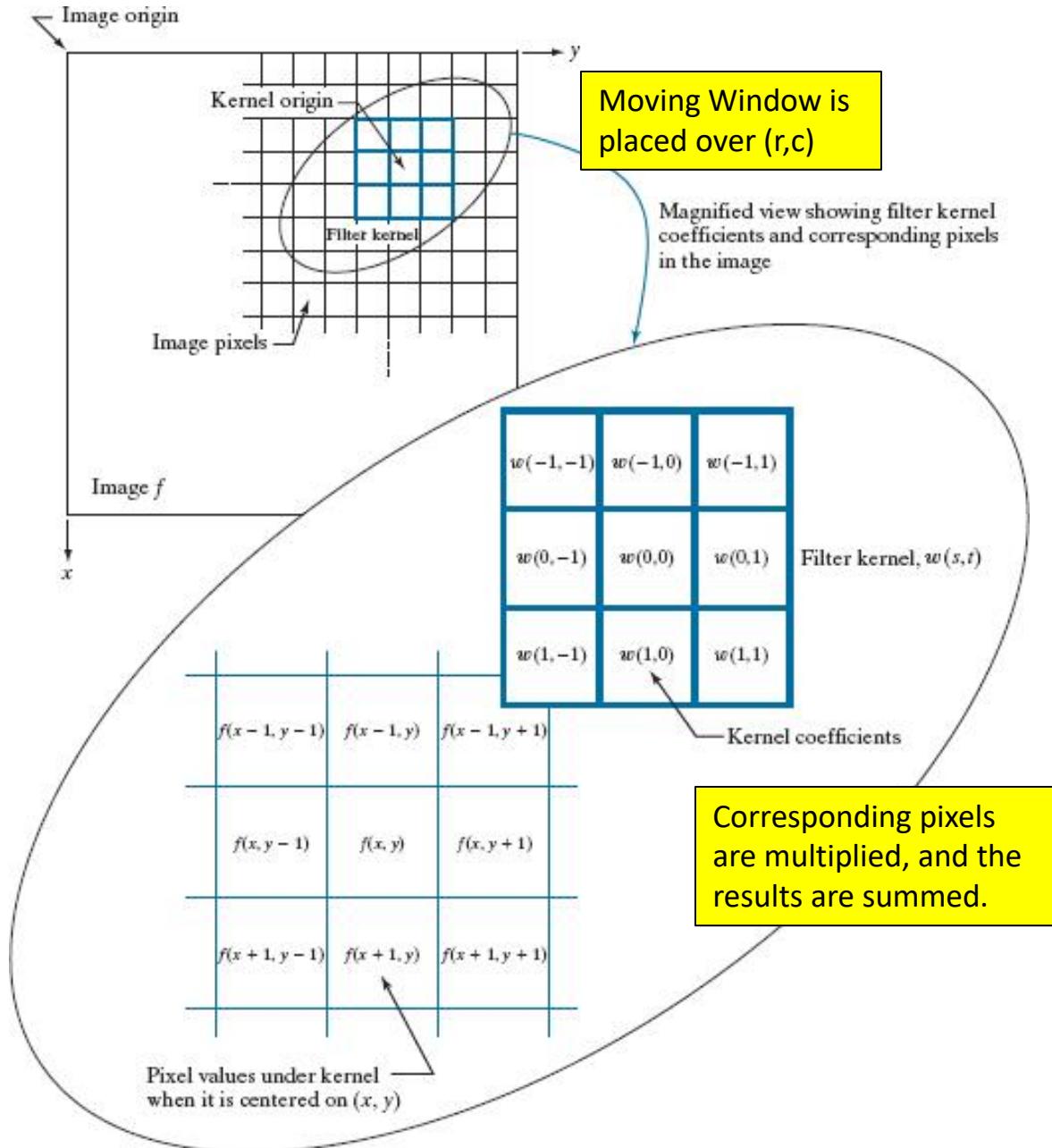
This is called cross-correlation operation and written:

$$G = H \otimes F$$

H is called the “filter”, ‘kernel’ or “mask”

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

Figure 3.34. Mechanics of linear spatial filtering. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.



Spatial Correlation

The correlation of a filter $w(x, y)$ of size $m \times n$
with an image $f(x, y)$, denoted as $w(x, y) \star f(x, y)$

$$w(x, y) \star f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$

Spatial Convolution

The convolution of a filter $w(x, y)$ of size $m \times n$ with an image $f(x, y)$, denoted as $w(x, y) \star f(x, y)$

$$w(x, y) \star f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x-s, y-t)$$

Comparison: Convolution vs. Correlation

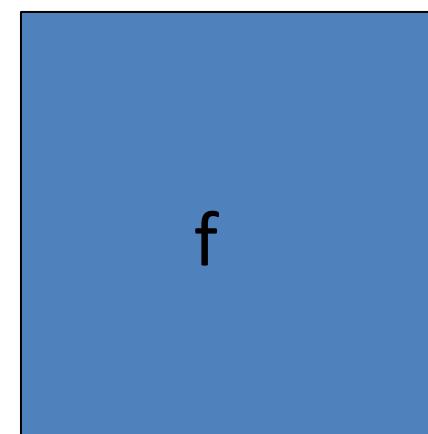
- Correlation $g=f \otimes h$ $g(i,j) = \sum_{k,l} f(i+k, j+l) \cdot h(k, l)$
- Convolution $g=f^*h$ $g(i,j) = \sum_{k,l} f(i-k, j-l) \cdot h(k, l)$

Note (Matlab)

- **imfilter:** by default uses correlation to filter images. (you can specify filtering by convolution)
- **conv2:** uses convolution

Convolution:

Flip the filter in both dimensions
(bottom to top, right to left)
Then apply cross-correlation

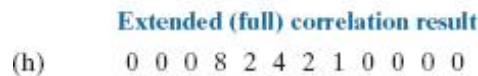
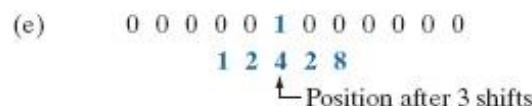
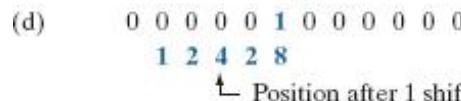
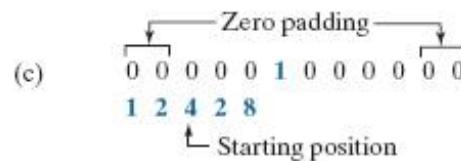
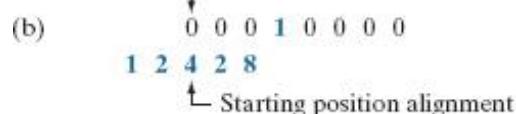
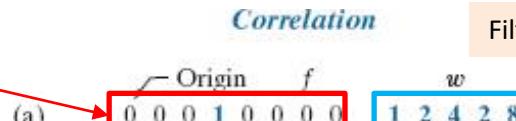


Gonzalez & Woods Figure 3.35

$$w(x, y) \quad f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x+s, y+t)$$

$$w(x, y) \quad f(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x-s, y-t)$$

Discrete unit impulse function



Filter/Kernel

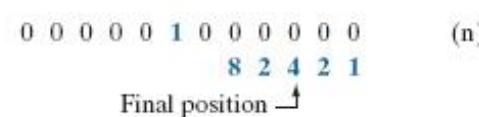
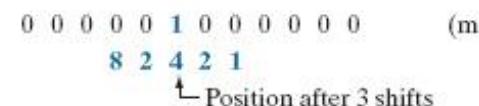
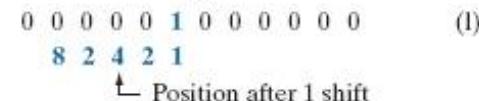
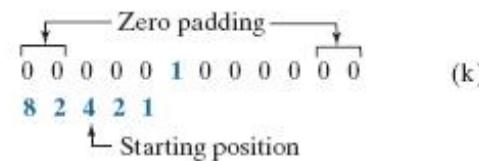
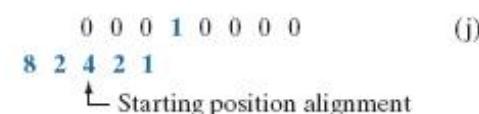
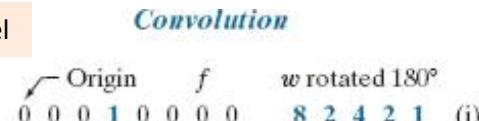


Figure 3.35.
 Illustration of 1-D
 correlation and
 convolution of a
 kernel, w , with a
 function f consisting
 of a discrete unit
 impulse.

Comparison: Convolution vs. Correlation

- Correlation $f \otimes h$ $g(i, j) = \sum_{k,l} f(i + k, j + l) \cdot h(k, l)$
- Convolution $f * h$ $g(i, j) = \sum_{k,l} f(i - k, j - l) \cdot h(k, l)$

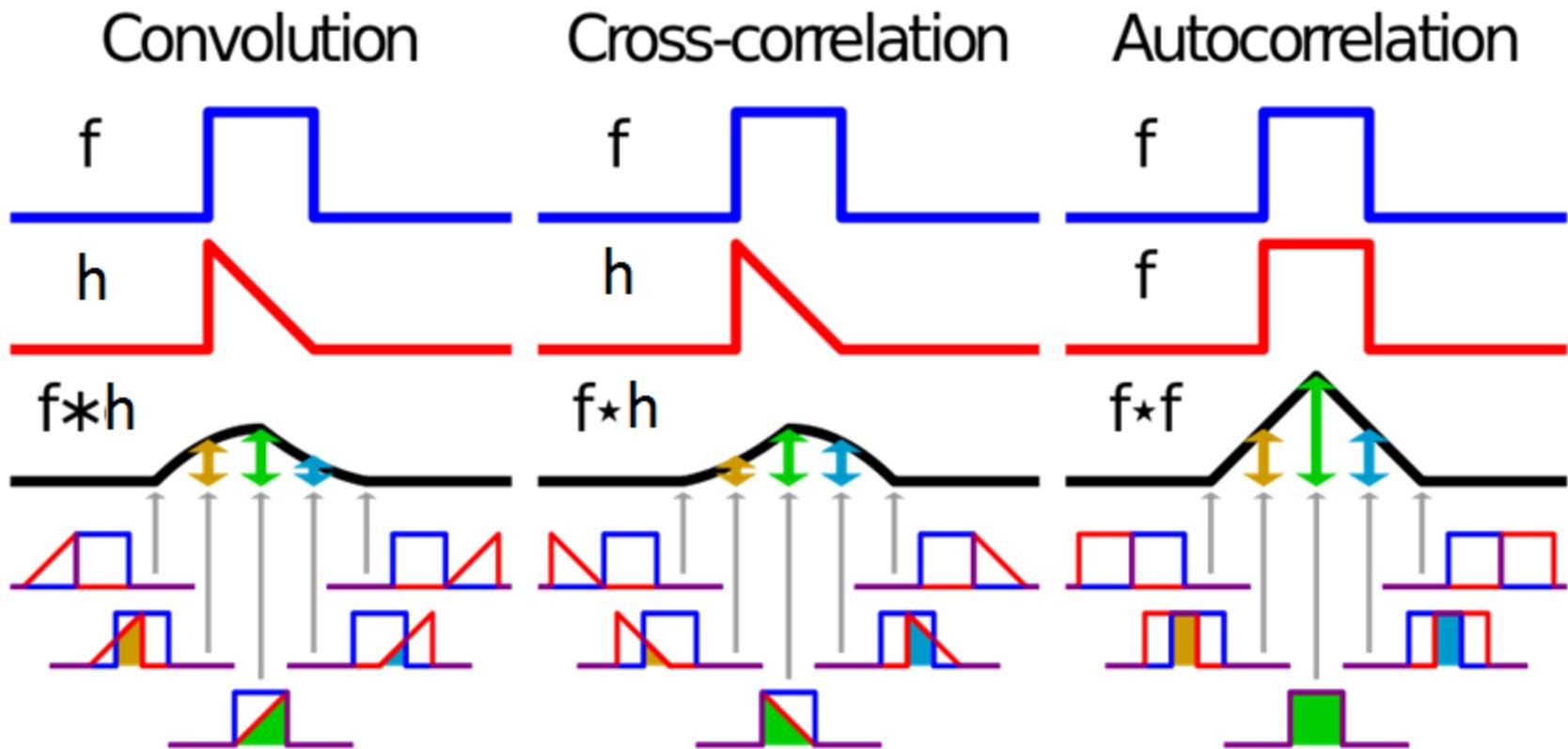
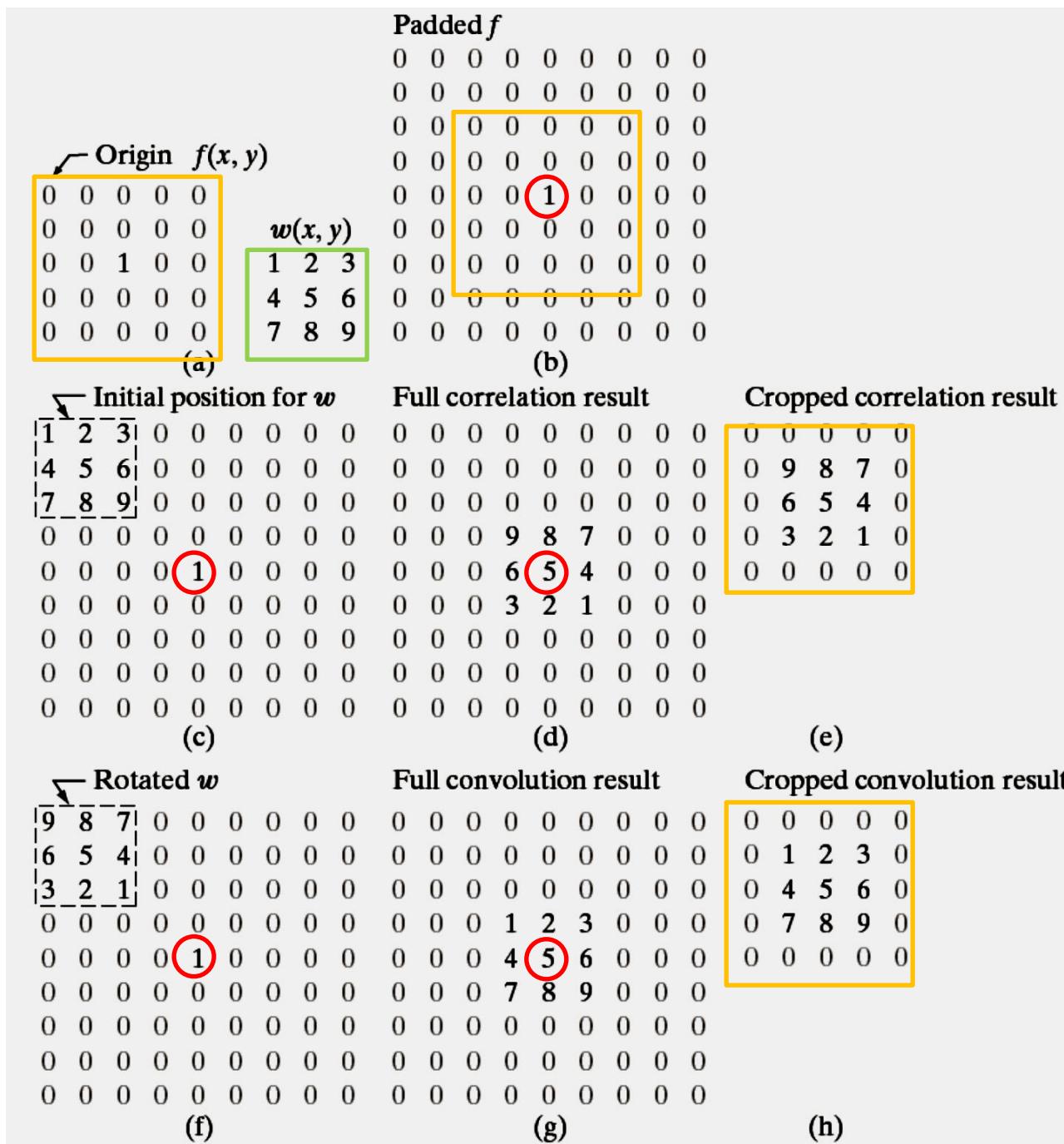


FIGURE 3.30
Correlation
(middle row) and convolution (last
row) of a 2-D filter with a 2-D
discrete, unit
impulse. The 0s
are shown in gray
to simplify visual
analysis.

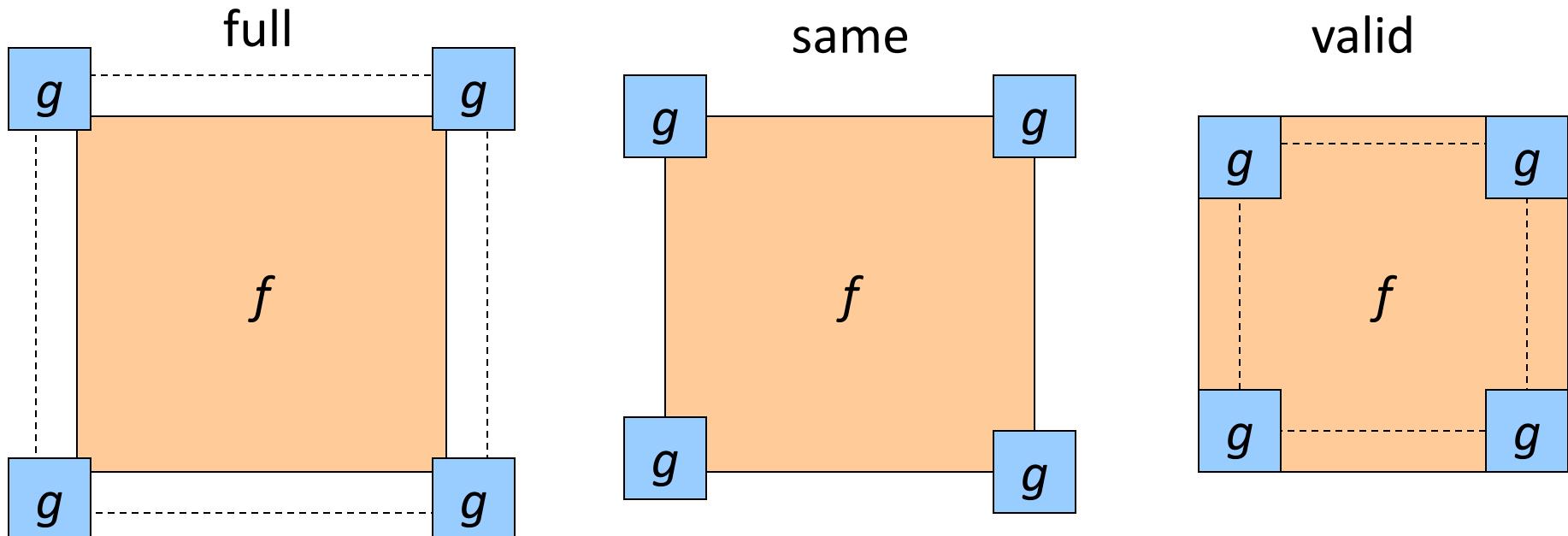
Correlation

Convolution



Boundary issues

- What is the size of the output?
- MATLAB: output size / “shape” options
 - *shape* = ‘full’: output size is sum of sizes of f and g
 - *shape* = ‘same’: output size is same as f
 - *shape* = ‘valid’: output size is difference of sizes of f and g



Smoothing Spatial Filters

- Smoothing filters are used for blurring and for noise reduction
- Blurring is used in removal of small details and bridging of small gaps in lines or curves
- Smoothing spatial filters include linear filters and nonlinear filters.

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \quad \frac{1}{4.8976} \times \begin{array}{|c|c|c|} \hline 0.3679 & 0.6065 & 0.3679 \\ \hline 0.6065 & 1.0000 & 0.6065 \\ \hline 0.3679 & 0.6065 & 0.3679 \\ \hline \end{array}$$

Examples of smoothing kernels: (a) is a box kernel; (b) is a Gaussian kernel.

Averaging filter

- What values belong in the kernel H for the moving average example?

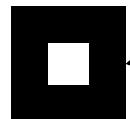
$$\begin{array}{c}
 F[x, y] \\
 \otimes \\
 \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 \end{array}
 \end{array}
 \quad
 \begin{array}{c}
 H[u, v] \\
 \otimes \\
 \begin{array}{|c|c|c|} \hline
 1 & 1 & 1 \\ \hline
 1 & 1 & 1 \\ \hline
 1 & 1 & 1 \\ \hline
 \end{array}
 \end{array}$$

$\frac{1}{9}$

“box filter”

$$G = H \otimes F$$

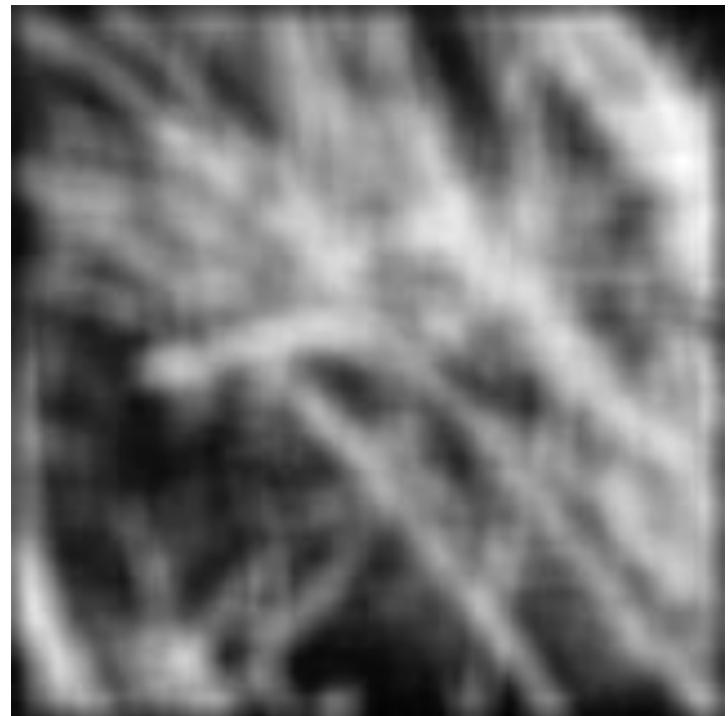
Smoothing by averaging



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



original



filtered

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

Source: S. Seitz

Smoothing by averaging – Effect of Filter Size

Square averaging filter of size $m=3,5,9,15,35$

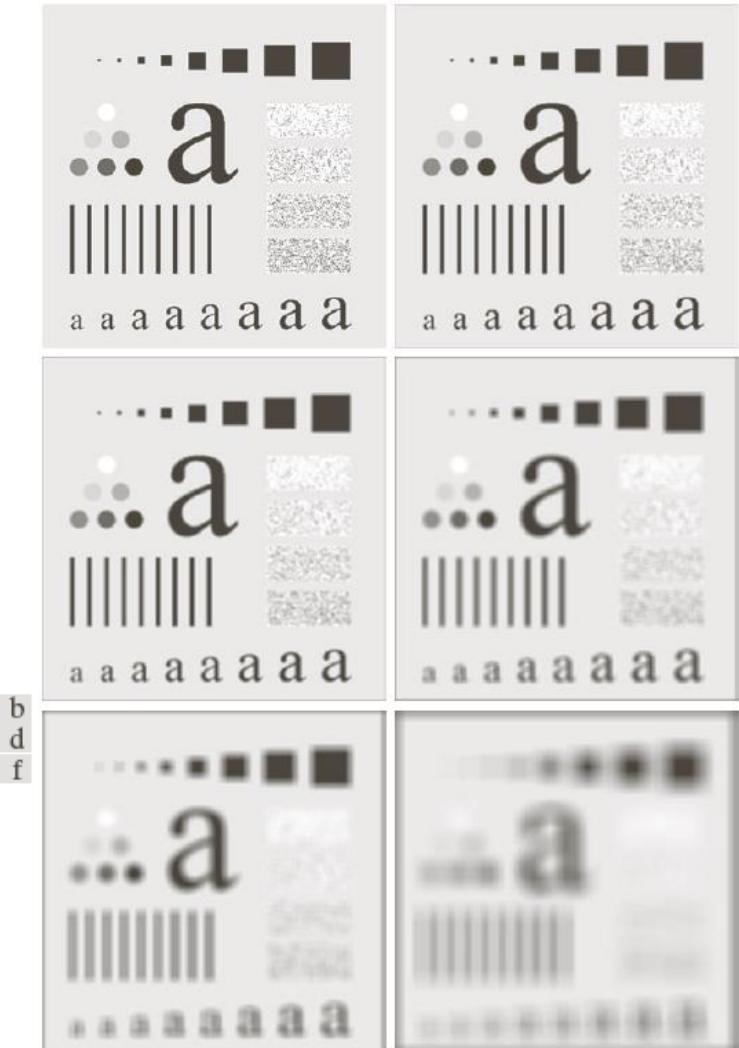


FIGURE 3.33 (a) Original image, of size 500×500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35 , respectively. The black squares at the top are of sizes $3, 5, 9, 15, 25, 35, 45$, and 55 pixels; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50×120 pixels.

Gaussian filter

Gaussian kernel gives less weight to pixels further from the center of the window.

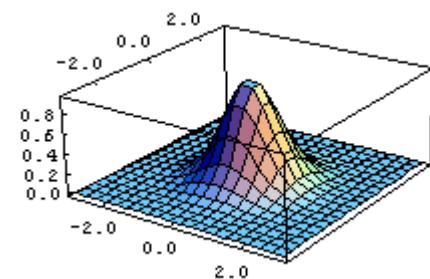
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$F[x, y]$$

$$\frac{1}{16} \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix}$$
$$H[u, v]$$

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

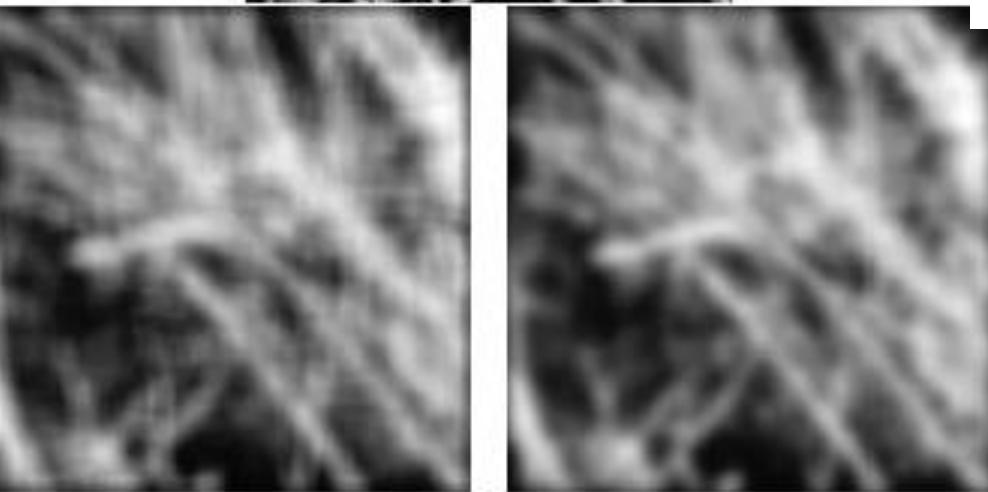
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



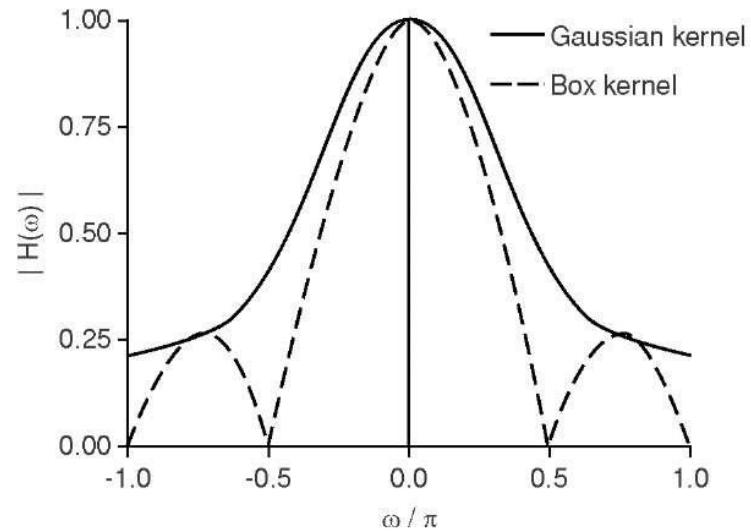
Removes high-frequency components from the image (“low-pass filter”).

Source: S. Seitz

Mean vs. Gaussian Filtering



S. Seitz

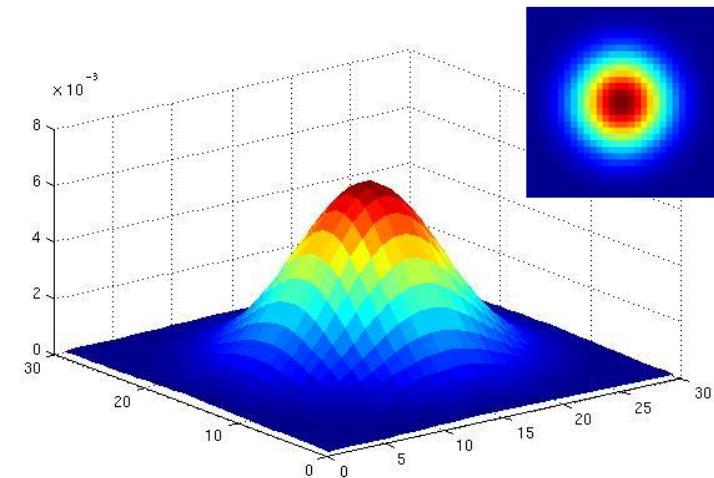
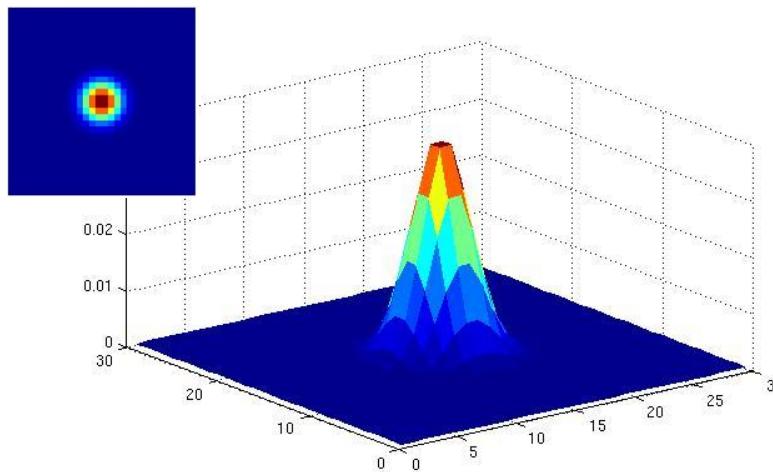


Frequency responses of the Gaussian and the box smoothing kernel. While the box kernel shows a steeper drop-off near low spatial frequencies ($m = 0$), it completely suppresses certain frequencies and allows higher frequencies to pass again. The Gaussian kernel has less smoothing action, but it does not show undesirable behavior in the stopband.

M.H. Haidekker

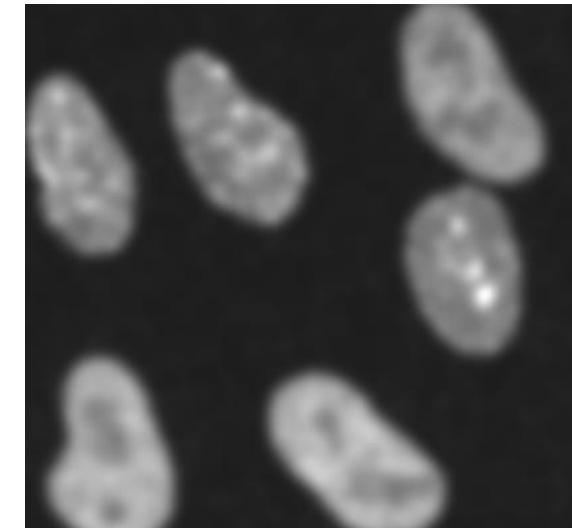
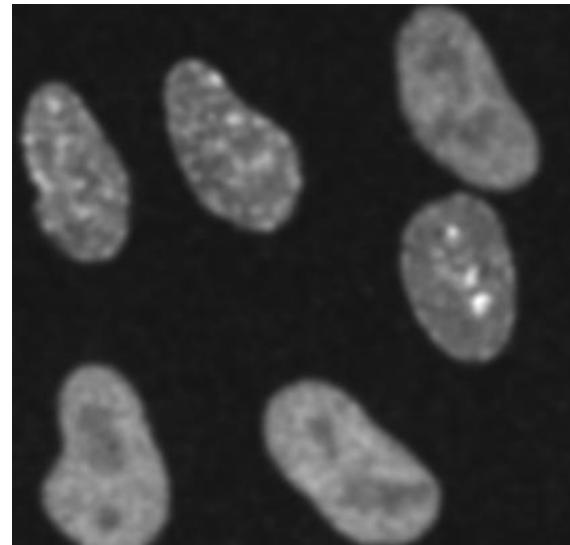
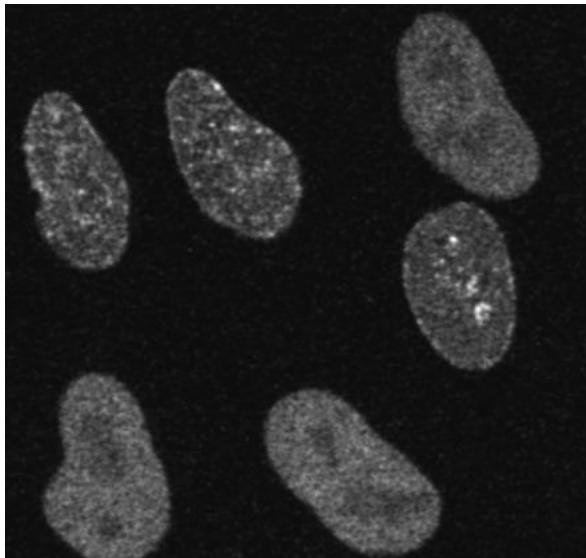
Gaussian filters

- What parameters matter here?
- **Variance of Gaussian:** determines extent of smoothing



Smoothing (Lowpass Filtering) with a Gaussian Filter

Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



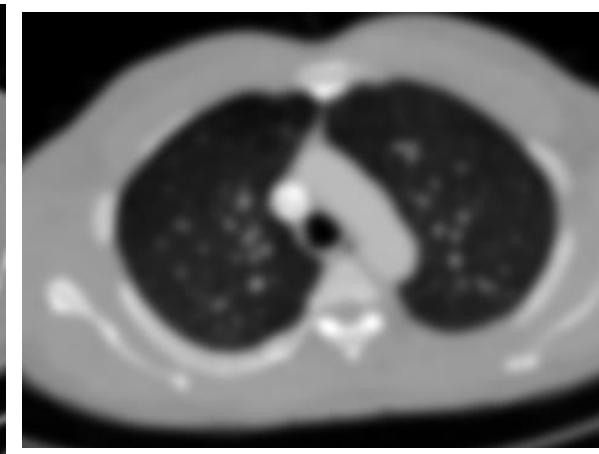
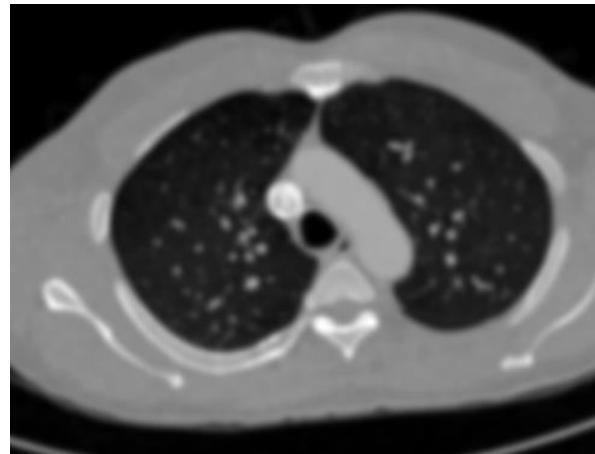
□

■

■

Smoothing (Lowpass Filtering) with a Gaussian Filter

Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.



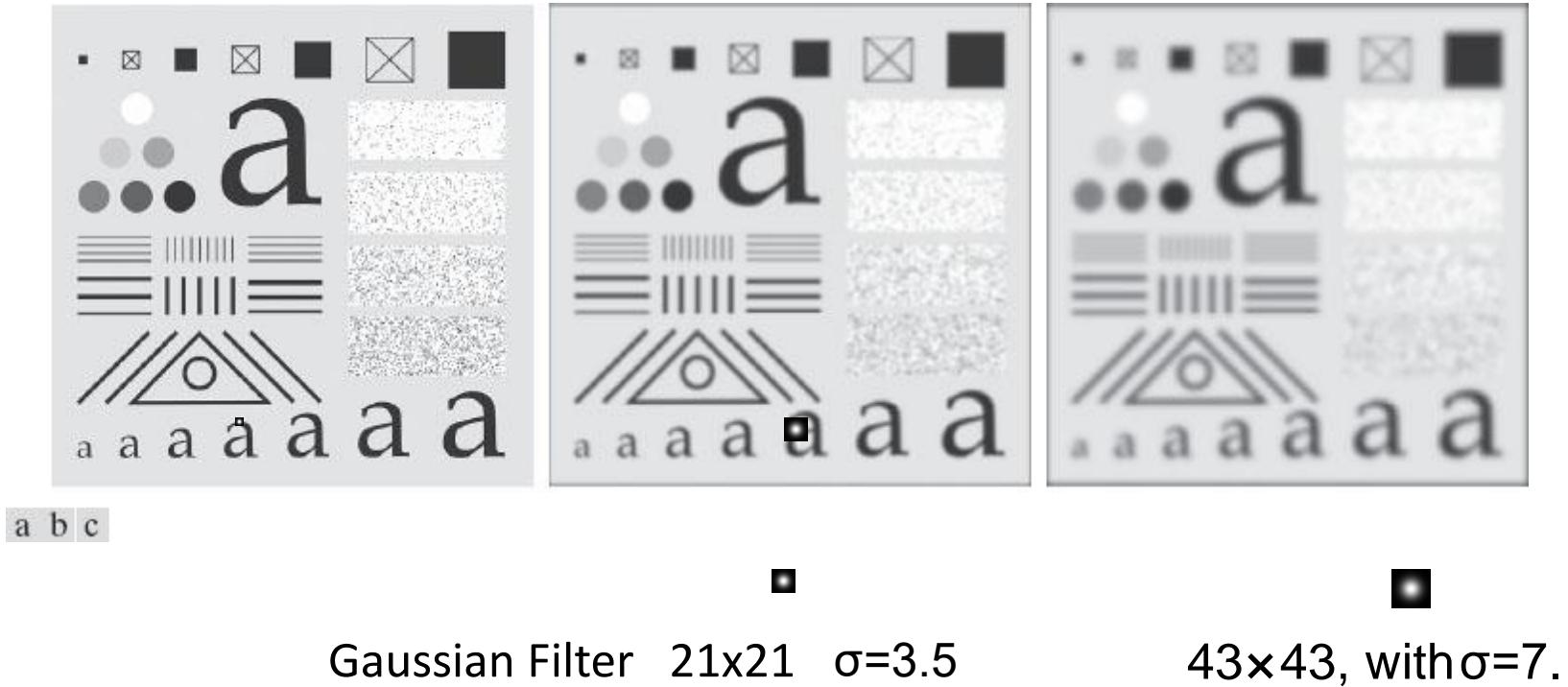
•

•

•

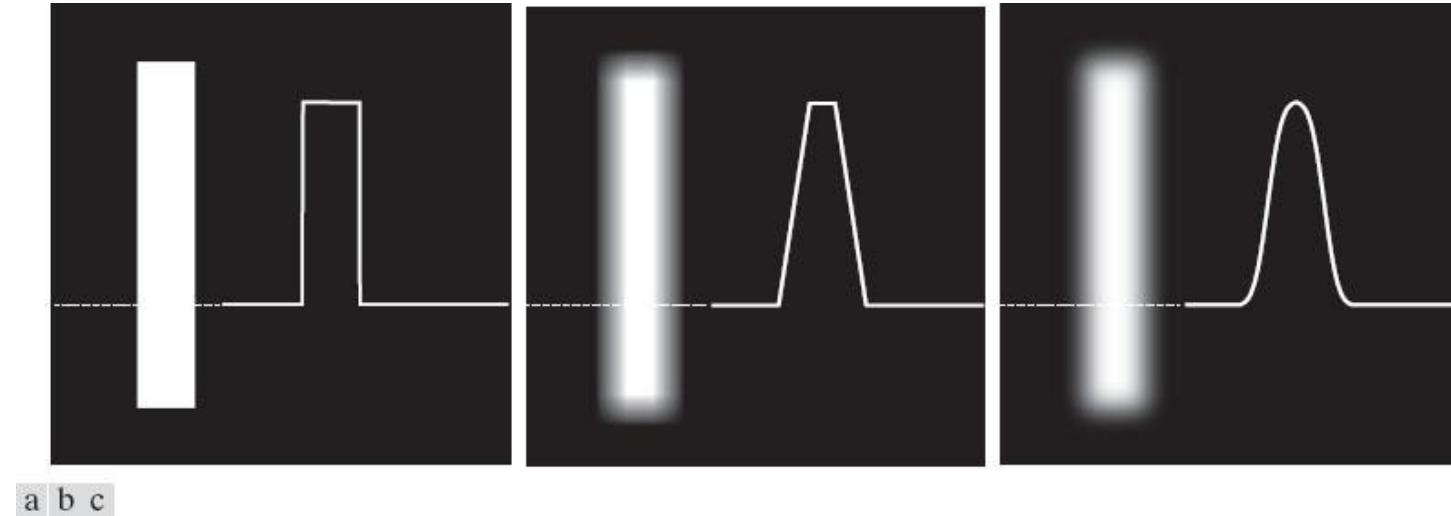
Smoothing (Lowpass Filtering) with a Gaussian Filter

Gonzalez & Woods Figure 3.42



Smoothing (Lowpass Filtering) with a Box Filter and Gaussian Filter

Gonzalez & Woods Figure 3.44



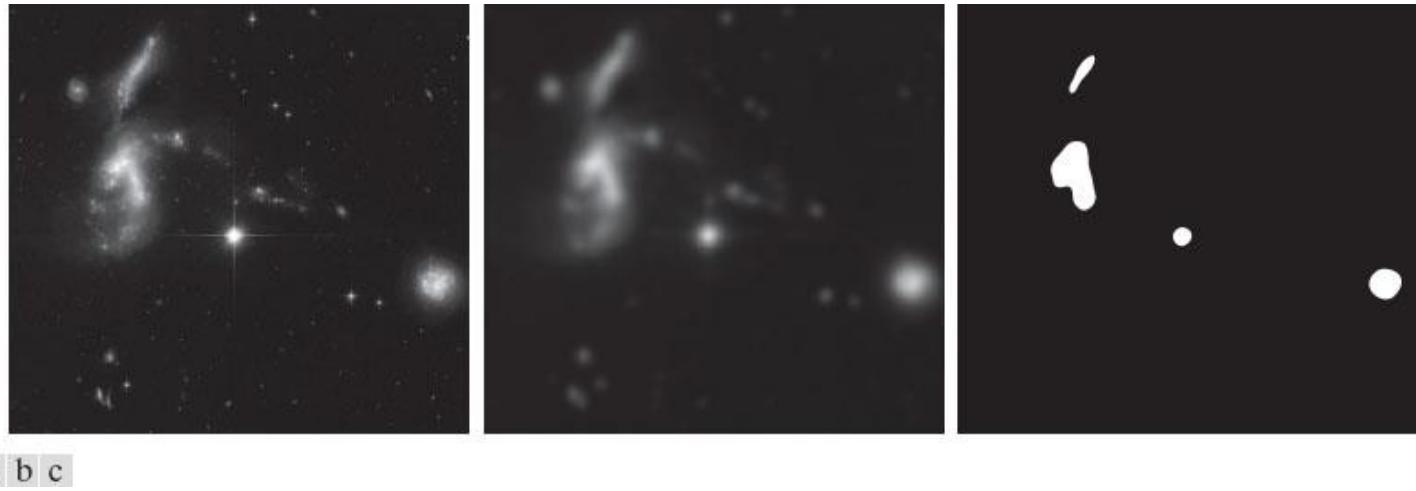
Using Lowpass Filtering and Thresholding for Region Extraction

Gonzalez & Woods Figure 3.47

Hubble Telescope image of the Hickson Compact Group.

- (a) 2566x2758 image.
- (b) Result of lowpass filtering with a Gaussian kernel.
- (c) Result of thresholding the filtered image (intensities were scaled to the range [0, 1]).

The Hickson Compact Group contains dwarf galaxies that have come together, setting off thousands of new star clusters. (Original image courtesy of NASA.)



Shading Correction using Lowpass Filtering

Gonzalez & Woods Figure 3.48

- (a) Image shaded by a shading pattern oriented in the -45 degree direction.
- (b) Estimate of the shading patterns obtained using lowpass filtering.
- (c) Result of dividing (a) by (b). (See Section 9.8 for a morphological approach to shading correction).



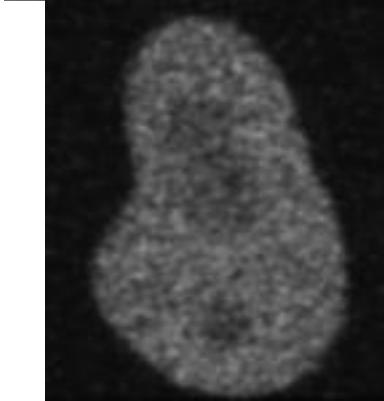
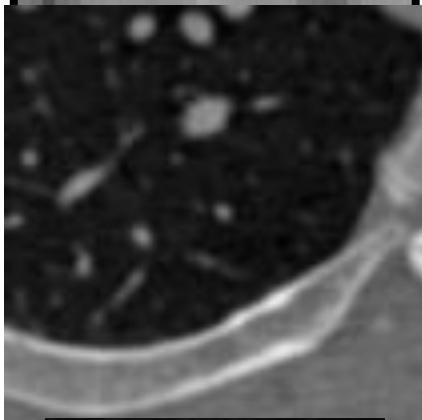
a b c

2048x2048 checkerboard image, 128x128 squares

(b) Result of lowpass filtering the image with a 512x512 Gaussian kernel

Practice with linear filters

Original



Blur (with a box filter)

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



$(1/81) * \text{ones}(9,9)$

Practice with linear filters



$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

-

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

?

Original

Practice with linear filters



Original

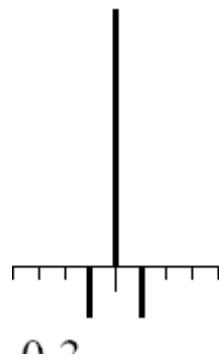
0	0	0
0	2	0
0	0	0

-

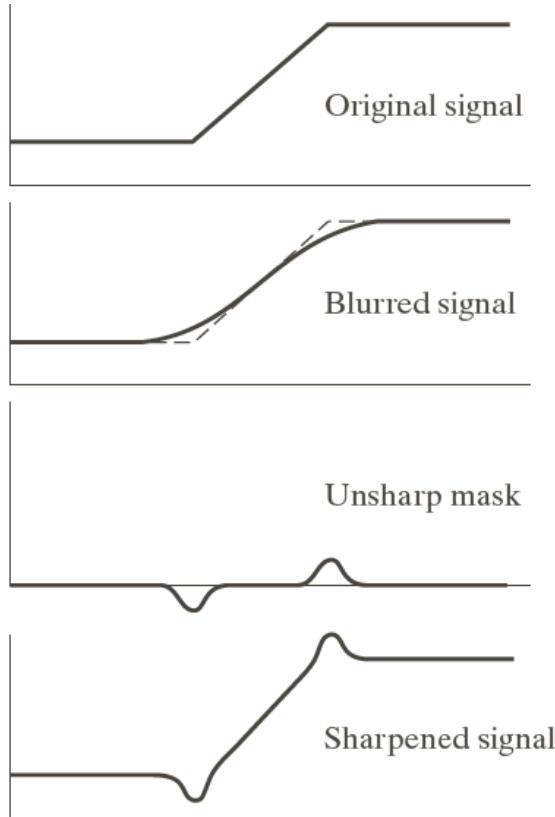
$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1



Sharpening filter:
accentuates differences with
local average

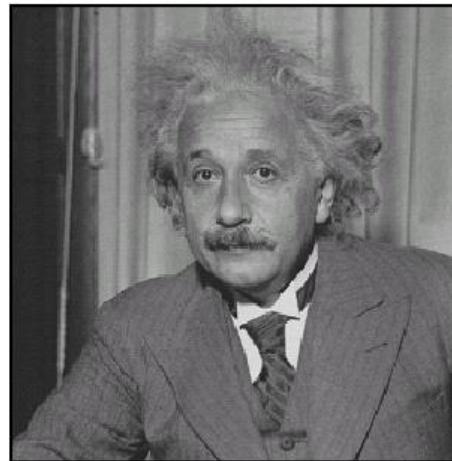


Sharpening

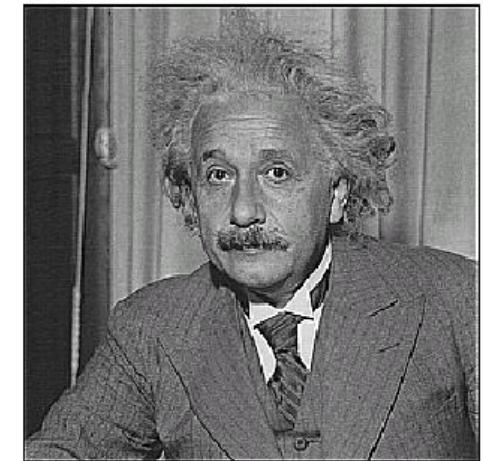


a
b
c
d

FIGURE 3.39 1-D illustration of the mechanics of unsharp masking.
(a) Original signal.
(b) Blurred signal with original shown dashed for reference.
(c) Unsharp mask.
(d) Sharpened signal, obtained by adding (c) to (a).



before



after

Unsharp Masking and Highboost Filtering

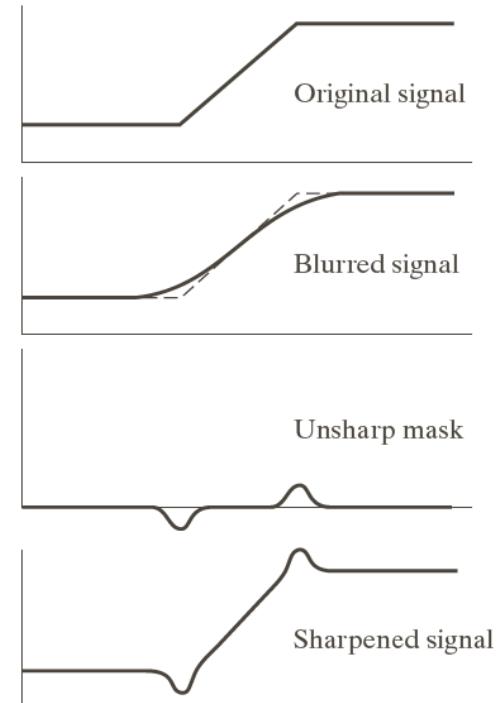
- Unsharp masking

Sharpen images consists of subtracting an unsharp (smoothed) version of an image from the original image

e.g., printing and publishing industry

- Steps

1. Blur the original image
2. Subtract the blurred image from the original
3. Add the mask to the original



Unsharp Masking and Highboost Filtering

Let $\bar{f}(x, y)$ denote the blurred image, unsharp masking is

$$g_{mask}(x, y) = f(x, y) - \bar{f}(x, y)$$

Then add a weighted portion of the mask back to the original

$$g(x, y) = f(x, y) + k * g_{mask}(x, y) \quad k \geq 0$$

when $k > 1$, the process is referred to as highboost filtering.

Unsharp Masking and Highboost Filtering: Example



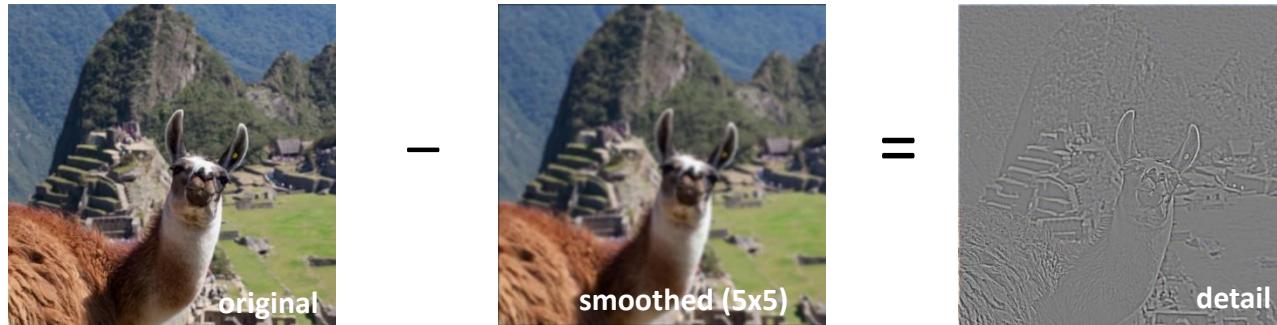
a
b
c
d
e

FIGURE 3.40

- (a) Original image.
- (b) Result of blurring with a Gaussian filter.
- (c) Unsharp mask.
- (d) Result of using unsharp masking.
- (e) Result of using highboost filtering.

Sharpening revisited

- What does blurring take away?



(This “detail extraction” operation is also called a ***high-pass filter***)

Let's add it back:

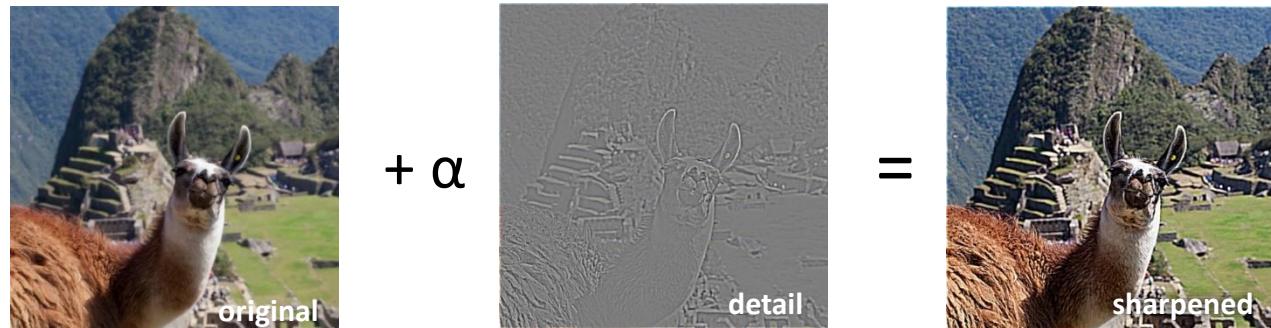
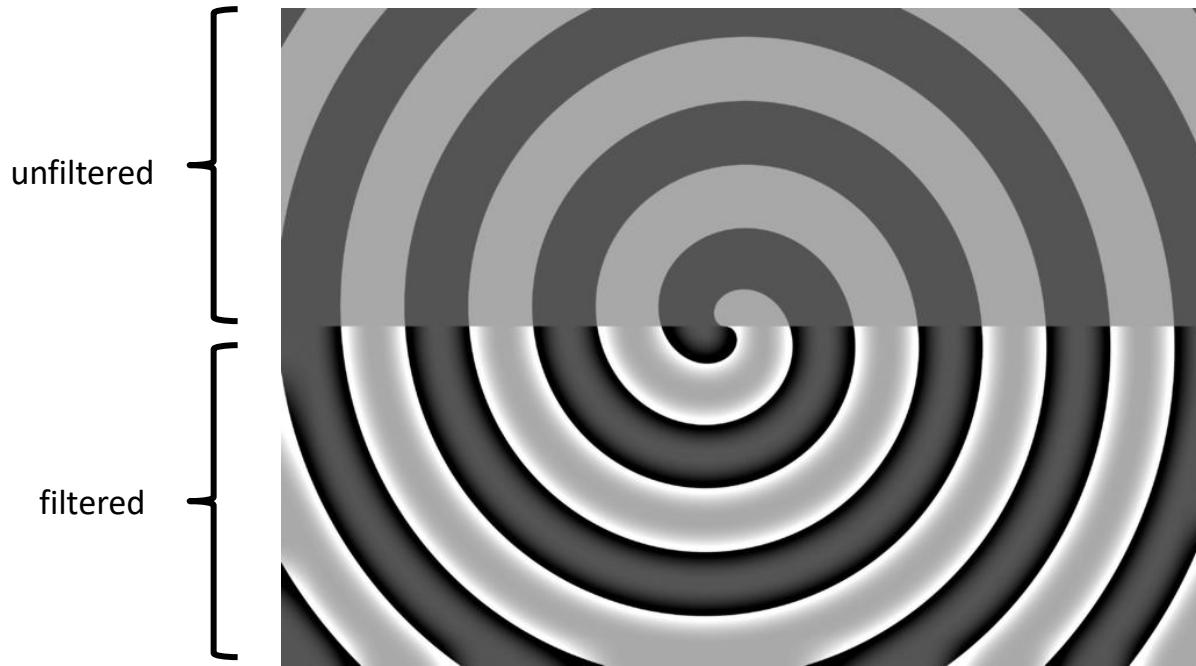


Photo credit: <https://www.flickr.com/photos/geezaweezer/16089096376/>

Sharpen filter



Note how in sharpened region of the image (also known as “unsharp mask filtered” image), the edges are transformed so that the image gets darker on one side and brighter on the other – in this way, the edge is emphasized. The sharpened image “anticipates” the edges.

Basic gradient filters

Horizontal Gradient

0	0	0
-1	0	1
0	0	0

or

-1	0	1
----	---	---

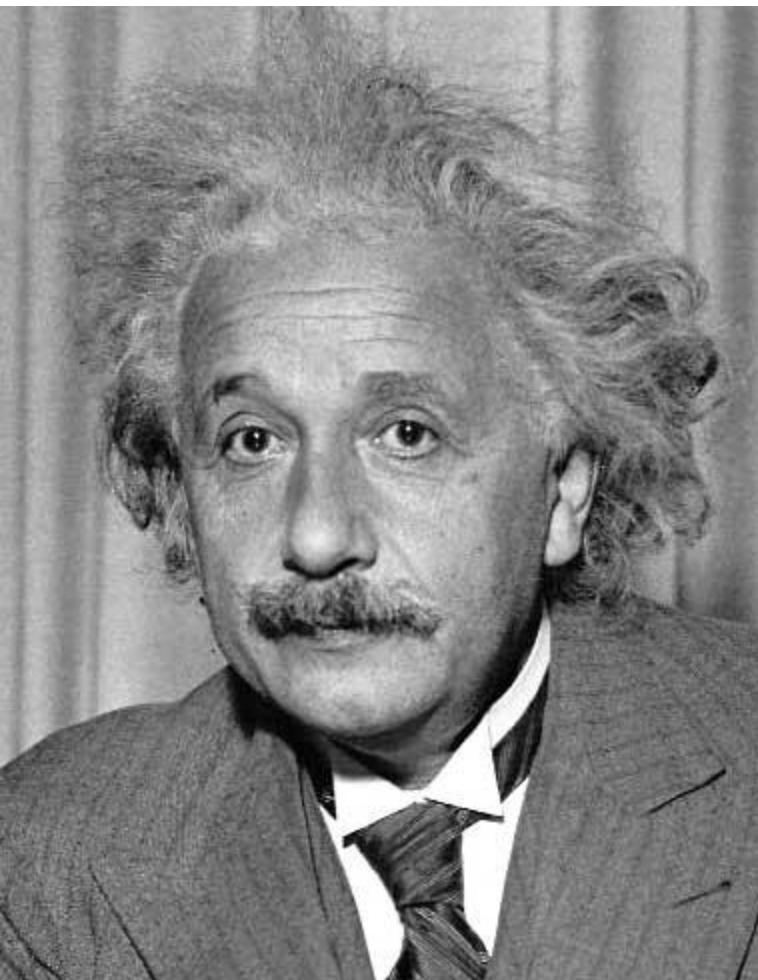
Vertical Gradient

0	1	0
0	0	0
0	-1	0

or

-1
0
1

Other filters



1	0	-1
2	0	-2
1	0	-1

Sobel



Vertical Edge
(absolute value)

More on filters

- Cross-correlation/convolution is useful for, e.g.,
 - Blurring
 - Sharpening
 - Edge Detection
 - Interpolation
- Convolution has a number of nice properties
 - Commutative, associative
 - Convolution corresponds to product in the Fourier domain
- More sophisticated filtering techniques can often yield superior results for these and other tasks:
 - Polynomial (e.g., bicubic) filters
 - Steerable filters
 - Median filters
 - Bilateral Filters
 - ...

Convolution Basic Properties

- Commutativity: $f * g = g * f$
- Associativity: $f * (g * h) = (f * g) * h$
- Distributivity: $f * (g + h) = f * g + f * h$
- Linearity: $(af + bg) * h = af * h + bg * h$
- Differentiation (one variable) $\frac{d}{dx}(f * g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$
- Differentiation (more general) $\frac{\partial}{\partial x_i}(f * g) = \frac{\partial f}{\partial x_i} * g = f * \frac{\partial g}{\partial x_i}$
- Note:
- *Linear function satisfies properties of*
 - *Superposition (additivity)*
 - *Homogeneity of degree 1 (scalar multiplication)*

Table 3.5

Some fundamental properties of convolution and correlation. A dash means that the property does not hold.

Property	Convolution	Correlation
Commutative	$f \star g = g \star f$	—
Associative	$f \star (g \star h) = (f \star g) \star h$	—
Distributive	$f \star (g + h) = (f \star g) + (f \star h)$	$f \star (g + h) = (f \star g) + (f \star h)$

Separable Filters

- Convolution is associative: $f * (g * h) = (f * g) * h$
- In some cases, filter is separable and we can apply the filter in two steps
- Then instead of one 2D convolution, we can apply two 1D convolutions.

$$f * k = f * (v * h) = (f * v) * h$$

2D convolution

1D convolution 1D convolution

WHY? What is the advantage?

Separable Filters

- Image size: NxN
- Filter size: KxK
- 2D Filtering cost: N²xK²

$$f * k = f * (v * h) = (f * v) * h$$

2D convolution 1D convolution 1D convolution

Some Separable Linear Filters

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

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

$$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

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

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

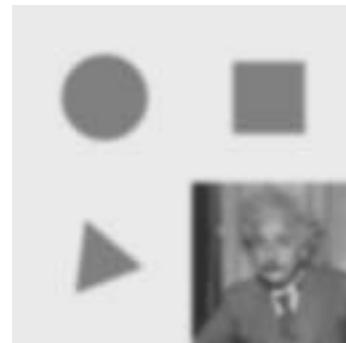
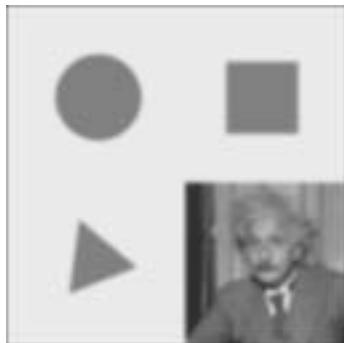
$$\frac{1}{K} \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}$$

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

$$\frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

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

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



a) box, K=5

b) bilinear

c) Gaussian

d) Sobel

e) corner

CONVOLUTION OPERATION & DEEP LEARNING

AI vs. ML vs. DL

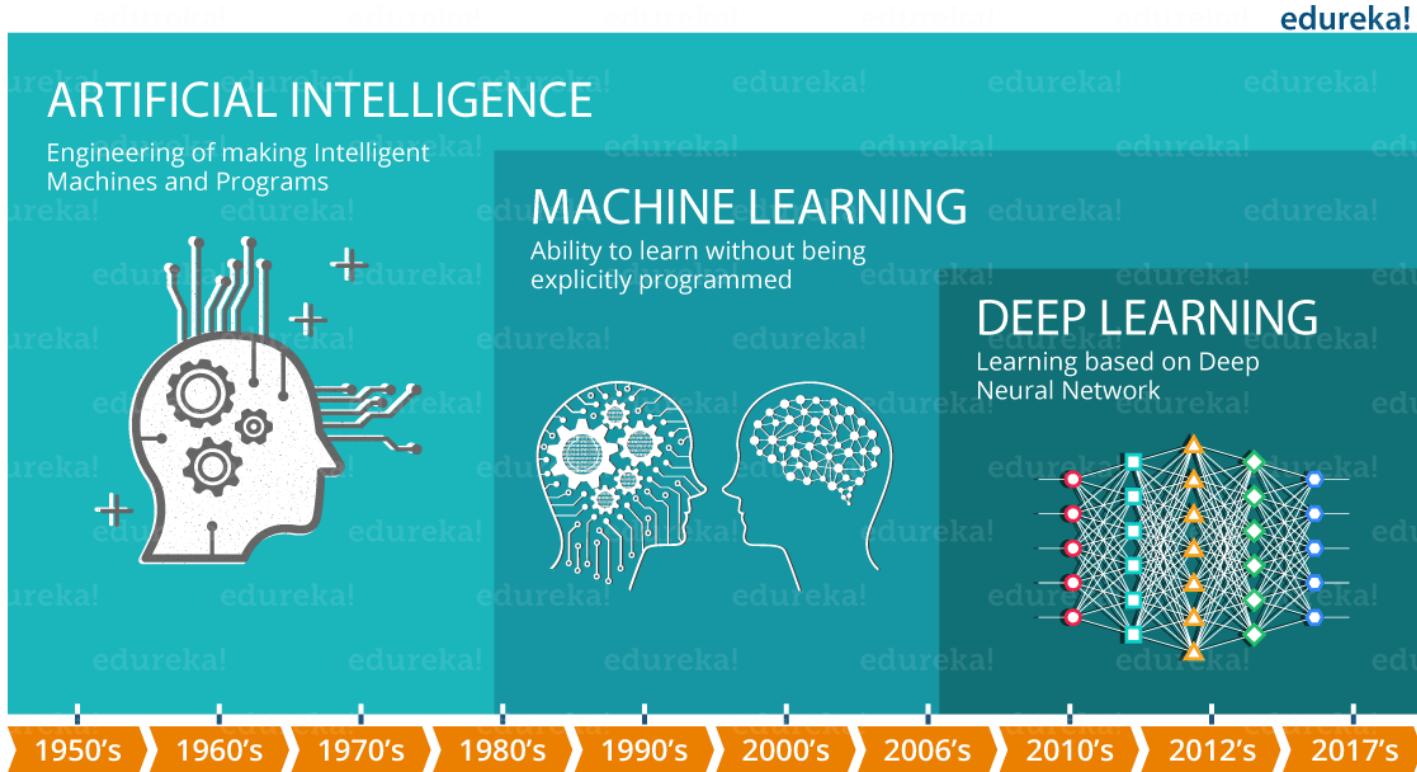


Figure from: <https://www.edureka.co/blog/what-is-deep-learning>

Artificial Intelligence: making a computer think intelligently, in the similar manner the intelligent humans think (John McCarthy: Computer scientist known as the father of AI).

Machine learning: ability to learn without being explicitly programmed (Arthur Samuel 1959).

Deep learning: learning based on artificial neural networks.

Brief Introduction to Convolutional Neural Networks (CNN)

CNN consists of three main types of layers:

1-Convolutional layers: core building block of a CNN.

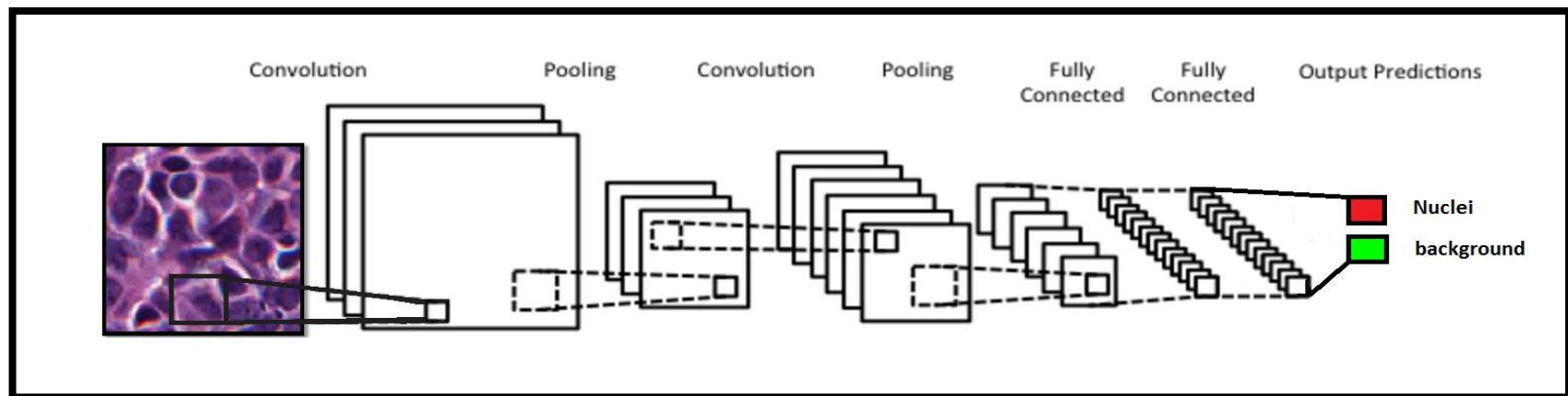
- a set of learnable filters (or kernels).
- each filter is convolved across the width and height of the input volume

2-Pooling layers:

- progressively reduce the spatial size of the representation to reduce the amount of parameters

3-Fully connected layer:

- Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks.



CNN Applications

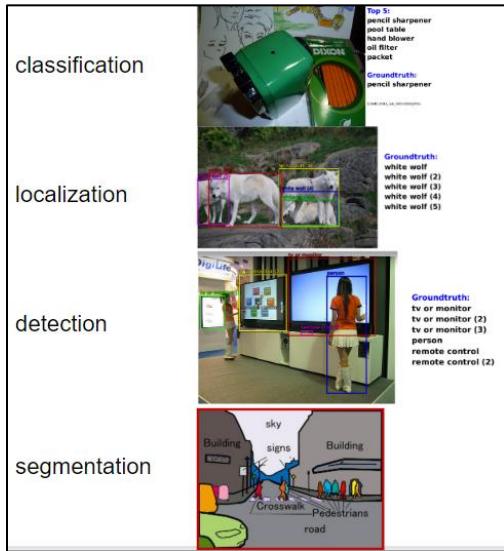


Figure: Pierre Sermanet, Google Research

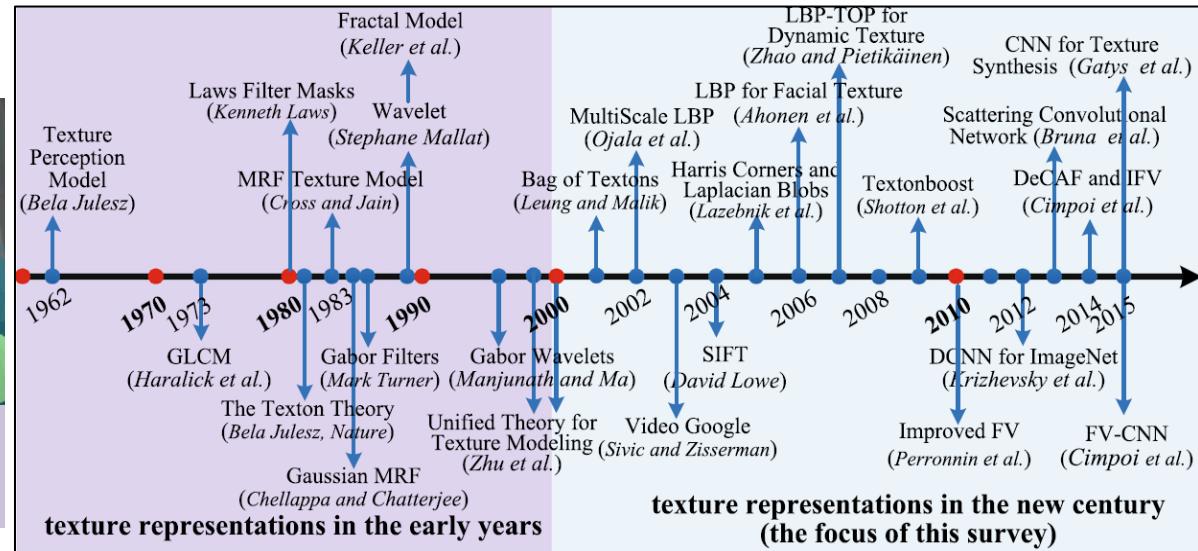
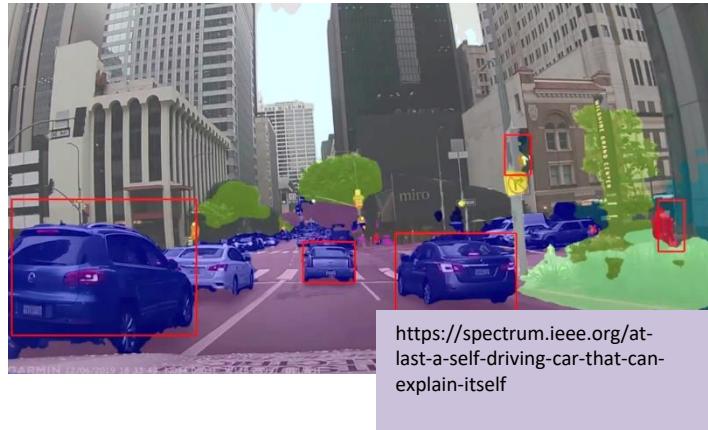
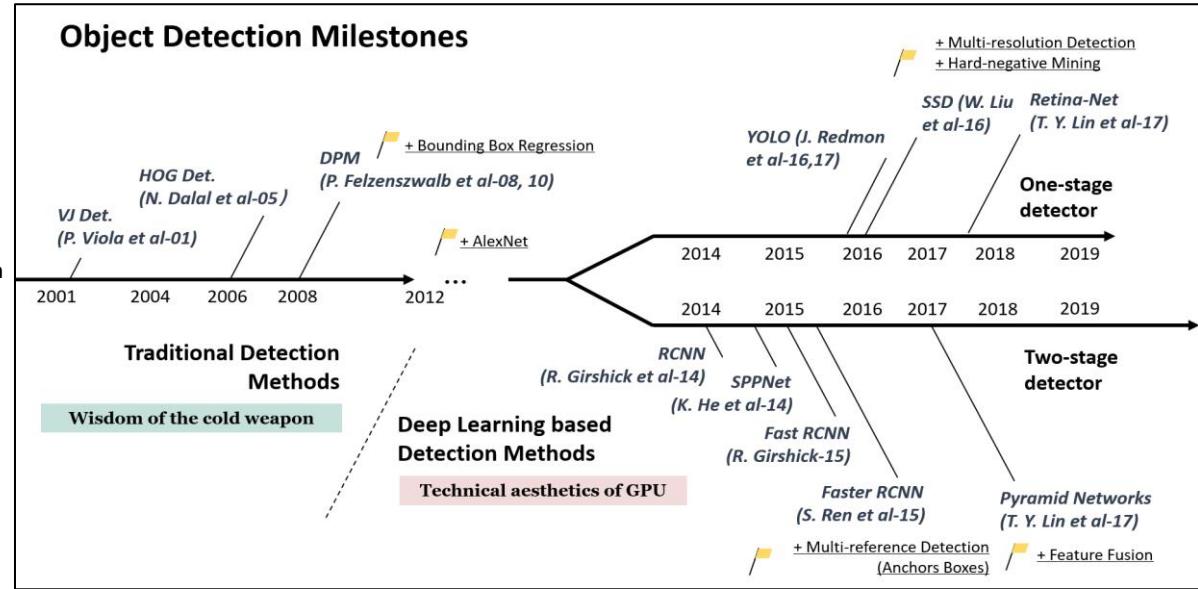


Figure 13.42

Numerical example illustrating the various functions of a CNN, including recognition of an input image. A sigmoid activation function was used throughout.

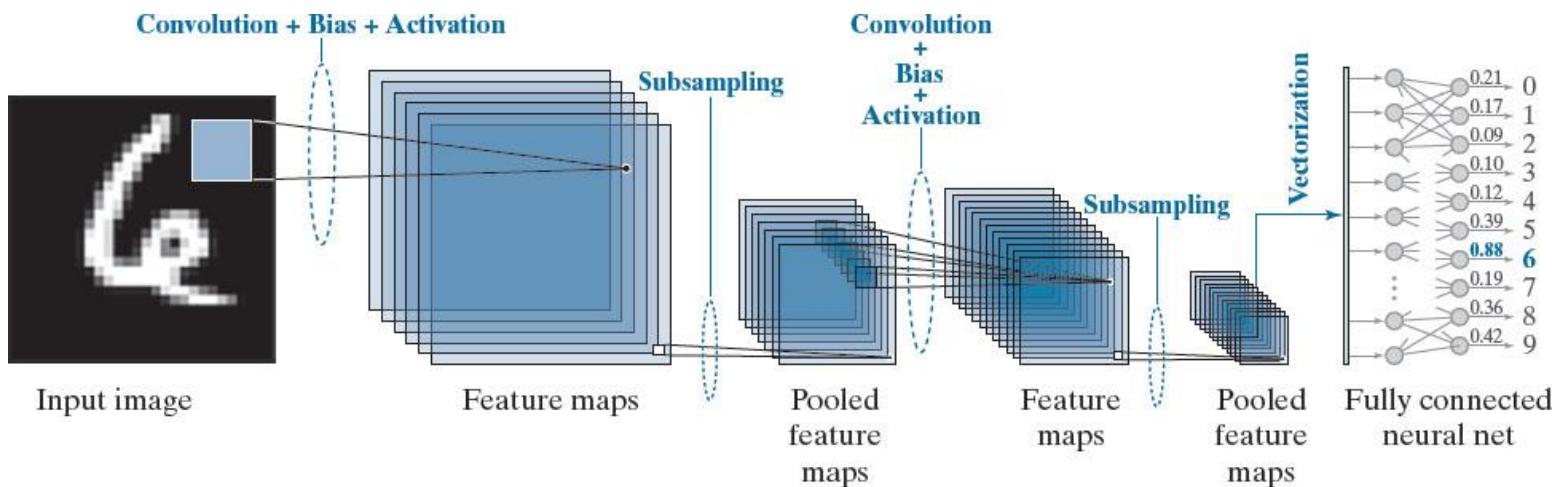


Figure 13.43

Top: The weights (shown as images of size 5x5) corresponding to the six feature maps in the first layer of the CNN in Fig. 13.42.

Bottom: The weights corresponding to the twelve feature maps in the second layer.



Received April 29, 2022, accepted May 10, 2022, date of publication May 17, 2022, date of current version May 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3175864

Hybrid Skip: A Biologically Inspired Skip Connection for the UNet Architecture

**NIKOLAOS ZIOULIS^{ID1,2}, GEORGIOS ALBANIS¹, PETROS DRAKOULIS^{ID1},
FEDERICO ALVAREZ^{ID2}, (Member, IEEE), DIMITRIOS ZARPALAS^{ID1},
AND PETROS DARAS^{ID1}, (Senior Member, IEEE)**

¹Visual Computing Laboratory, Centre for Research and Technology Hellas, Information Technologies Institute, 57001 Thessaloniki, Greece

²Signals, Systems and Radiocommunications Department, Universidad Politécnica de Madrid, 28040 Madrid, Spain

Corresponding author: Nikolaos Zioulis (nzioulis@gmail.com)

This work was supported by the European Commission H2020 Funded Project ATLANTIS (GA 951900).

ABSTRACT In this work we introduce a biologically inspired long-range skip connection for the UNet architecture that relies on the perceptual illusion of hybrid images, being images that simultaneously encode two images. The fusion of early encoder features with deeper decoder ones allows UNet models to produce finer-grained dense predictions. While proven in segmentation tasks, the network's benefits are down-weighted for dense regression tasks as these long-range skip connections additionally result in texture transfer artifacts. Specifically for depth estimation, this hurts smoothness and introduces false positive edges which are detrimental to the task due to the depth maps' piece-wise smooth nature. The proposed HybridSkip connections show improved performance in balancing the trade-off between edge preservation, and the minimization of texture transfer artifacts that hurt smoothness. This is achieved by the proper and balanced exchange of information that HybridSkip connections offer between the high and low frequency, encoder and decoder features, respectively. The code and models will be made available in the project page.

INDEX TERMS Computer vision, dense depth estimation, UNet, skip connections, scale-space, hybrid images, spherical vision, monocular inference.

I. INTRODUCTION

Skip connections, specifically, the bypassing of convolutional layer blocks within a convolutional neural network (CNN) architecture, are a core building block of modern data-driven models [59]. Residual blocks [15], [16] use short-range skip connections with identity mappings and residual functions to improve information propagation in both forward and backward passes. They are the basic building block of ResNets, one of the most popular and better performing CNN backends.

At the same time, UNet [47] is another autoencoder CNN architecture that relies on long-range skip connections, forwarding early encoder features to their corresponding resolution features on the decoder's side. Different from residual skip connections, UNet concatenates the encoder and decoder features, allowing the network to implicitly learn their fusion through the decoder's convolutional layers. However, it is a challenging problem as there exists a semantic gap between the encoder features and the corresponding decoder ones,

The associate editor coordinating the review of this manuscript and approving it for publication was Tai-Hoon Kim.

which stems from the higher level concepts and semantic information that is progressively encoded into CNNs. Despite this challenge, UNet remains a dominant architecture, especially for semantic segmentation, surpassing fully convolution networks (FCN) [32], mainly because it offers higher boundary preservation performance.

Consequently, various works have focused on overcoming this encoder-decoder semantic gap in UNet's long-range skip connections. Straightforward approaches add learnable operations to lessen the gap with MultiResUNet [21] relying on residual blocks. More involved approaches utilize gating-based spatial attention [38] to attend to the encoder features in a localized manner, or semantic embedding branches and global convolutions [63]. The search for an appropriate encoder-decoder skip connection led to the use of neural architecture search (NAS) [56] to identify the squeeze-and-excite operation [18] as the more prominent candidate mapping function.

Notably, all these works have applied their proposed skip connection in semantic segmentation, the downstream task that UNet was initially applied at. Yet recently, UNet-like architectures are increasingly being used in depth estimation

as well [6], [12]–[14], [33], [44], [65], as the long-range skip connections offer higher boundary preservation performance. However, the latter is a dense regression task, in contrast to the former, which is a dense classification task. For depth estimation, the model’s parameters encode a continuous function approximation, whereas for semantic segmentation the model focuses on learning a high-dimensional decision surface. The core difference lies in the nature of depth maps, which are piecewise smooth functions [20], meaning that compared to semantic segmentation, the smoothness property needs to also hold for the predicted output, whereas for segmentation, the preservation of the boundary is the only secondary trait of importance. Consequently, for a regression task like depth estimation, skip connections usually result in texture transfer artifacts which hurt smoothness, and introduce false positive boundaries.

In this work, we design a biologically-inspired skip connection based on the way humans process visual input [5], [29], specifically the decomposition into different spatial frequencies that happens early on in the visual pathway. Higher spatial frequencies become imperceptible with farther viewpoints, with the reverse holding for closer viewpoints. The human visual system assimilates higher spatial frequencies into lower ones as viewing distance increases, a mechanism that has been exploited by prior work to generate illusions [39]. We exploit this mechanism as well, to facilitate the exchange of information between the encoder and decoder features, taking into account their higher and lower frequency nature respectively resulting from the autoencoder’s inductive bias.

More specifically, we contribute the following:

- We design a lightweight and plug-n-play hybrid feature skip connection for the UNet architecture. It performs a blending-based information exchange between the higher and lower level feature maps partaking in a long-range skip connection, prior to their fusion.
- We experimentally demonstrate the efficacy of various skip connections in a dense regression task, while taking into account their performance differentials on secondary traits as well; namely boundary preservation and smoothness.
- We demonstrate that our proposed skip connection strikes a better balance at boosting direct depth, boundary and smoothness performance, compared to other state-of-the-art skip connections.

II. RELATED WORK

The UNet CNN architecture [47] was initially introduced for semantic segmentation and was the first architecture to include long-range skip connections, forwarding information from the encoder to the decoder via feature fusion. The skip connection improves detail preservation by propagating the early encoder features near the prediction features, boosting semantic segmentation accuracy by allowing for thinner structure segmentation, rendering UNet the standard architecture for this task. More information about the

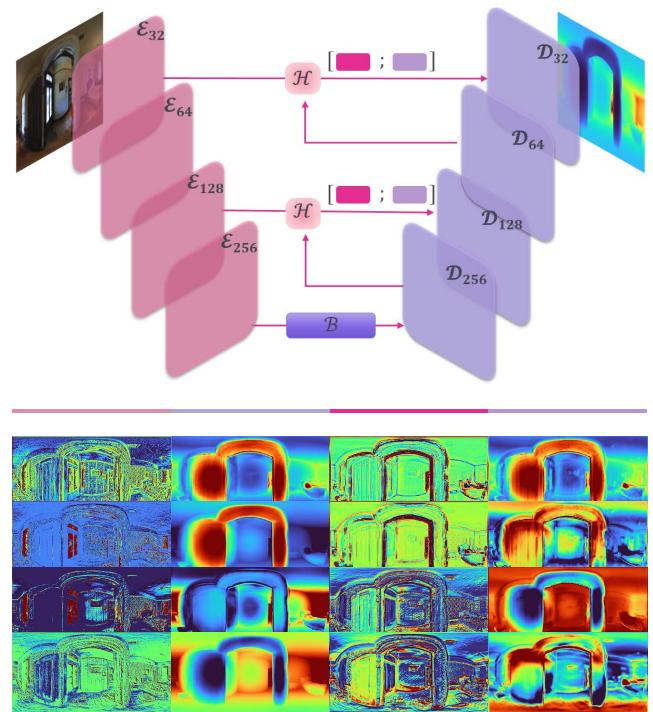


FIGURE 1. Long-range skip connections are instrumental to the popular UNet architecture but are also challenged by the semantic gap between the encoder \mathcal{E} and decoder \mathcal{D} features. While they allow UNets to capture high resolution details, this is not always beneficial to dense regression tasks that need to overcome texture transfer and also preserve smoothness. We introduce a biologically inspired skip connection that balances the effect of the high frequency encoder features and the dominant structural information carried by the decoder ones. From left to right, each bottom rows visualizes encoder and decoder features maps before and after the hybrid skip connection from a trained dense depth regression model.

UNet architecture and the importance of the skip connection can be found in various surveys about U-shaped network architectures [31], [42].

Due to its efficacy, it has received a lot of attention and multiple variants have surfaced, with some notable examples being UNet++ [66], U²Net [43], UNet 3+ [19], VNet [36], YNet [34], WNet [57] and nnUNet [23]. Further, it has been gaining traction for tasks other than semantic segmentation such as image reconstruction, with examples being inpainting [30], view-synthesis [2], [46] and relighting [62], as well as depth estimation [12]–[14], [65]. Adding to the latter, in the recent Mobile AI 2021 Challenge [22] on single image depth estimation, 7 out of 10 submissions used UNet architectures. Also, the detail preserving nature of early encoder features allowed its application as the discriminator architecture in high quality synthesis tasks [49].

Nonetheless, its strength also presents as one of its main weaknesses. While skip connections propagate details near the prediction layers, facilitating more detailed dense signal reconstructions, they are not necessarily optimal in their pure identity mapping form. The reason for this is that the raw fusion of early encoder and late decoder information is hindered by their semantic gap. CNNs typically extract high spatial-frequency details (e.g. edges, texture, lines) in the

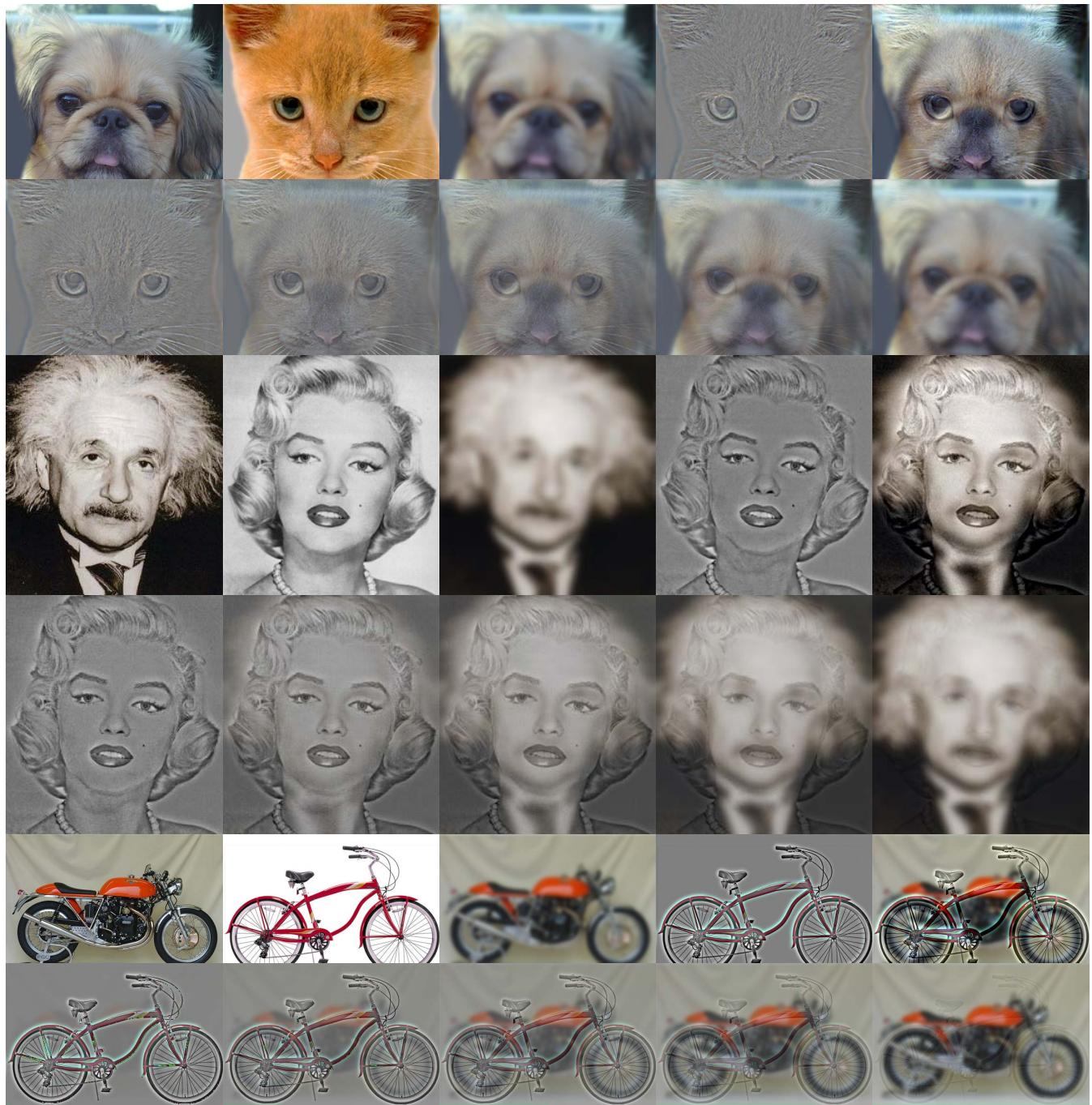


FIGURE 2. The hybrid images [39] human vision based illusion that encode a dual image. From left to right, for the top row of each example pair: i) first and ii) second image, iii) low pass filtered first image, iv) high pass filtered second image, and v) the hybrid image which changes with viewing distance (from second to first, by zooming in and out the document respectively). The bottom row of each example pair shows the blending of the low and high pass images using an interpolated blending factor from 0.1 to 0.9 that mathematically simulates the physical viewing distance change. At the bottom of each image the green bar indicates the interpolation factor value.

early stages, while at the deeper layers the network produces category-specific features representations [3], [15].

Among the techniques designed to address this semantic gap, ExFuse [63] used a complex skip connection, replacing the identity mapping with a cascade comprising a semantic embedding branch and a global convolution module. Results in both an FCN and a UNet demonstrated its

efficacy in improving semantic segmentation performance. Approaching the same problem from another perspective, Attention UNet [38] introduced a novel attention gate as the skip connection. Each skip connection softly attends to the incoming encoder features using a gating signal. Initially, additive attention between the projections of the gating signal and encoder features is used to generate an attention grid

after aggregating and projecting the result. This is then resampled and used to reduce or preserve the importance of the encoder features in a localised manner. Results in medical segmentation showcased an improvement over vanilla UNet. The concept was similarly applied to the UNet++ architecture, resulting in Attention UNet++ [28], which adapts the gating signal to the nesting levels and shorter skip connections.

More recently, in MultiResUNet [21] the identity mapping skip connection was replaced by a series of residual blocks that aim at alleviating the semantic gap between the encoder and decoder features. Taking into account that earlier encoder features suffer from a bigger semantic gap, more blocks were used in the earlier features than the ones closer to the bottleneck. Apart from an improvement in dense and boundary segmentation, the residual skip connections also exhibited robustness to noise. In a similar fashion, MAPUNet [58], inspired by UNet++ [66], and UNet 3+ [19], exploited multi-scale feature fusion and supervision for monocular depth estimation. Moreover, a UNet++ variant with residual blocks and dense gated convolution based attention [60] was used for monocular depth estimation using sparse depth measurements [64]. Finally, NasUNet [56] employed neural architecture search to look for an efficient and effective UNet architecture, a finding shared by [48] as well. Their search resulted in identifying the Squeeze-and-Excite operation [18] as the most dominant replacement for the standard (identity) skip connection. Apart from the identity mapping, the search performed included traditional and dilated [7] convolutions, as well as separable depthwise convolutions [50]. Evaluation in different medical segmentation datasets showed performance increases at a fraction of the parameters and reduced memory cost.

Evidently, all aforementioned works focused on segmentation tasks, while all works using UNet's skip connections in reconstruction or regression tasks rely on the vanilla UNet. Dense regression tasks impose more stringent requirements compared to segmentation tasks, as the predicted signals need to exhibit richer properties. A notable example are depth images that need to preserve edges and their magnitude, while also varying smoothly in areas where no significant discontinuities manifest [20]. Compared to previous works, we focus on UNet networks used for regression and holistically assess the efficacy of these advanced skip connections [21], [38], [56], [63] to improve performance and preserve properties like boundaries and smoothness. Further, we propose a biological vision inspired skip connection based on scale space theory, that better preserves the output signal's secondary properties simultaneously.

III. APPROACH

Our work focuses solely on the long-range skip connections found in the UNet architecture, and specifically the fusion of features coming from different depths of the model. Encoder features are learned earlier (shallower) and on higher

resolutions, while decoder features are learned later (deeper) and on lower resolutions than the correspondingly encoder ones that they will be fused with. Our inspiration stems from the Hybrid Images [39]. We briefly introduce them in Section III-A, following with our proposed Hybrid Skip connection in Section III-B.

A. HYBRID IMAGES

Hybrid images \mathcal{H} are dual images that jointly encode two different images, \mathcal{A} and \mathcal{B} , but only one is largely perceived. Their interpretation changes with viewing distance, creating a smooth optical illusion which has been used to study patients [27], face identification [35], create two-layer QR codes [61], or even used for recreational art. They are generated by the blending of two different spatial resolution images:

$$\mathcal{H} = 0.5 \mathbf{f}_l(\mathcal{A}) + 0.5 \mathbf{f}_h(\mathcal{B}), \quad (1)$$

where \mathbf{f}_l and \mathbf{f}_h are a low-pass and high-pass filter respectively. Essentially, image \mathcal{A} is highly blurred, making it visible from farther distances, while image \mathcal{B} is composed by edges, which are only visible from close up. Figure 2 shows the resulting illusion and intermediate representations.

B. HYBRID SKIP CONNECTION

The UNet architecture's success relies on the long-range skip connection [42], [59] that fuses early encoder features $\mathcal{E} \in \mathbb{R}^{F \times H \times W}$ with late decoder features $\mathcal{D} \in \mathbb{R}^{F \times H \times W}$. The typical UNet fusion scheme is a learnable fusion using a convolutional layer receiving as input the concatenation of the encoder and decoder features:

$$\mathcal{F} = H_i([s(\mathcal{E}); \mathcal{D}]), \quad (2)$$

where $H(\cdot)$ denotes the convolution function of the i th layer, and without loss of generality $s(\cdot)$ denotes the encoder features' skip function, which for the typical UNet is the identity mapping. It is this multi-scale propagation of earlier encoder features to the late decoder layers that allows UNet architectures to capture finer details. Yet, there are challenges associated with this fusion scheme, namely the semantic gap between \mathcal{E} and \mathcal{D} as well as the different spatial frequencies of these two feature maps.

Earlier CNN blocks capture lower level features like lines and edges, while later CNN blocks capture higher level features and concepts, a fact that constitutes their – straightforward – fusion an inefficient approach. Further, earlier encoder features are captured in higher resolutions and contain higher spatial frequencies, while later decoder features contain lower spatial frequencies and are typically upsampled at the skip connection fusion step. Bilinear interpolation of a lower resolution feature map results in low spatial frequencies [9].

Hybrid images, as represented by Eq. (1), blend together two images of different spatial frequencies, toggling the perception of one or the other via how the human visual system's perception changes with viewing distance. The latter

mechanism can be generalized to alpha blending:

$$\mathcal{H}_a(\mathcal{A}, \mathcal{B}) = \alpha \mathbf{f}_a(\mathcal{A}) + (1 - \alpha) \mathbf{f}_b(\mathcal{B}), \quad (3)$$

where \mathbf{f}_a and \mathbf{f}_b are two filters converting \mathcal{A} and \mathcal{B} into different frequency images. The blending coefficient α controls the viewing distance, and therefore, converts the dual image to a distinctly perceived representation. Figure 2 shows the transition from one image to the other as α is interpolated in $[0.1, 0.9]$.

Considering the skip connection fusing the semantically and spectrally different feature maps \mathcal{E} and \mathcal{D} , we rely on the following hybrid feature functions:

$$\mathcal{H}_\delta^d(\mathcal{E}, \mathcal{D}) = \delta \mathcal{D} + (1 - \delta) \mathbf{f}_l(\mathcal{E}) \quad (4)$$

$$\mathcal{H}_\epsilon^e(\mathcal{E}, \mathcal{D}) = \epsilon \mathcal{E} + (1 - \epsilon) \mathbf{f}_h(\mathcal{D}), \quad (5)$$

where $\delta, \epsilon \in \mathbb{R}^F$ are two alpha blending vectors. These are combined to form the hybrid skip connection's fusion function:

$$\mathcal{F}_{hybrid} = H_i([\mathcal{H}_\epsilon^e(\mathcal{E}, \mathcal{D}); \mathcal{H}_\delta^d(\mathcal{E}, \mathcal{D})]). \quad (6)$$

Compared to most other non-identity skip connections [21], [38], [56], [63], the hybrid skip connection presented in Eq. (4), (5) and (6) facilitates a bidirectional information exchange between the encoder \mathcal{E} and decoder \mathcal{D} features, whereas the aforementioned skip connections only focus on bridging the semantic gap between \mathcal{E} and \mathcal{D} by increasing the semantic information carried by the encoder features \mathcal{E} .

Analyzing HybridSkip: There are multiple ways that \mathcal{F}_{hybrid} can be analyzed. From an attention perspective it can be considered as a mix of heterogeneous feature boosting [53] using a soft attention [24] on the respective features. The decoder features attend to the encoder ones, and vice versa, boosting specific features depending on the blending factors. While traditional channel attention simply scale entire feature maps (e.g. the squeeze-and-excite skip connection in [56]) and grid based attention only focuses on spatial feature selection (i.e. [38]), our hybrid approach is distinctly different, albeit it combines these two concepts. The channel attended encoder(decoder) features boosting the respective channel attended decoder(encoder) are directly related to the spatial information as already learned by the features.

From a spectral processing point of view, it can be considered as a selective alignment or focusing of the spatial frequencies of the blended feature maps. Considering that the early encoder features \mathcal{E} contain higher frequencies than the upsampled late decoder features \mathcal{D} , the second term in Eq. (4) and (5) is essentially a band-pass filtered feature map as low/high frequency inputs are passed through a high/low frequency filter. Therefore, both terms blend inputs from a frequency spectrum lying in the middle of the two opposite end, spatial frequency wise, original feature maps.

Considering the semantic gap, it is apparent that the hybrid skip connection closes the gap in a symmetric fashion by using both inputs to derive the features to be fused. In contrast to most approaches, it does not seek to close the gap by

\epsilon decreases, the structural edges derived from the decoder features become more dominant in the fused encoder features, accentuating these edges compared to those encoded in \mathcal{E} . Similarly, as δ decreases, the smoothed detailed edges encoded in the encoder features progressively add texture to the decoder features. With appropriate blending factors, both directions tend to reduce texture transfer and preserve the edges that matter, leading to a balancing effect between the smoothness and boundary preservation properties of the resulting fused feature maps, and eventually the predicted signal. Notably, the process is distinct for each feature map, meaning that with δ and ϵ being learnable parameters of the model, it encodes a dual representation of these features and learns which one is more appropriate during training.

IV. RESULTS

Experimental Setup: For our analysis we use a dense regression task, namely depth estimation, which requires the balancing of both boundary preservation and smoothness of the predicted depth maps, apart from its direct depth estimation performance. To fully exploit rich depth maps that include both smooth regions, as well as lots of foreground to background depth discontinuities, we use an omnidirectional image benchmark [1]. It includes spherical panoramas that capture entire indoor scenes, containing a lot of flat surfaces (ceiling, floors, tables, etc.), as well as a plurality of foreground objects given their omnidirectional field of view, resulting in a rich piece-wise smooth depth map. Similar to most works on spherical depth estimation [6], [10], [12], [41], [67], we evaluate depths up to 10m and use standard metrics for depth estimation, as well as boundary preservation [17], [26] and surface orientation [55].

Implementation Details: Our implementation is based on *moai* [37] which uses PyTorch 1.8 [40], PyTorch Lightning 1.0.7 [11] and Kornia 0.4.1 [45]. For all experiments we use the same UNet architecture and supervision scheme used in Pano3D [1], fixing the learning rate (0.0002), optimizer (default parameterized Adam [25]), batch size (4) and random number generator seed. Thus, only the skip connection varies from experiment to experiment. We use the Pano3D low resolution (512×256) Matterport3D (M3D) train and test splits for all experiments and apply no data augmentation, training for 60 epochs. For the low pass and high pass filters \mathbf{f}_l and \mathbf{f}_h , we use a discrete isotropic Gaussian and a discrete isotropic Laplacian filter respectively.

A. ANALYSING THE HYBRID SKIP CONNECTION

In this section we seek to understand the proper design of the hybrid skip connection. Our analysis focuses on one hand on the kernel size K of the low and high pass filters \mathbf{f}_l and \mathbf{f}_h respectively, and on the other hand on the choice of the encoder and decoder blending factors ϵ and δ respectively. Regarding the latter, one approach would be to use con-

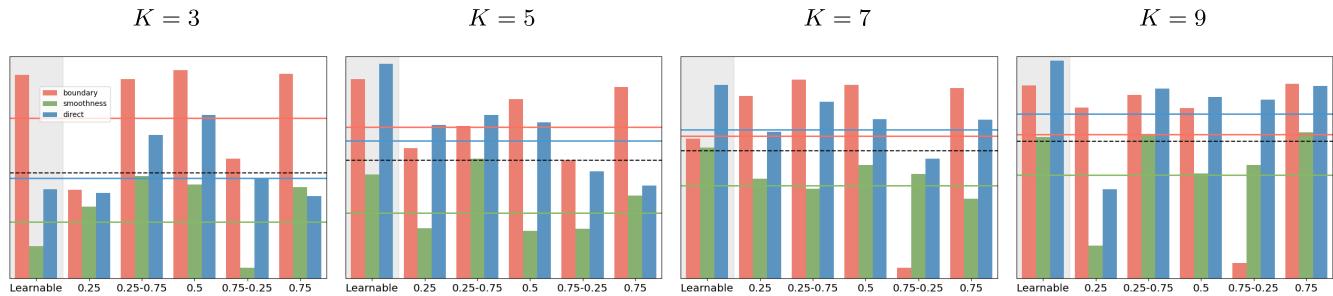


FIGURE 3. Direct depth (i_d - pale blue), boundary (i_b - tomato) and smoothness (i_s - pale green) performance indicators across different kernel sizes and blending factors. The indicator colored horizontal lines denote the average across all blending factors for each kernel size group K , while the black dashed line indicates the average of all three performance indicators. Each bar plots uses a distinct scale which has been normalized to lie in the same range for clarity. Evidently, with increasing kernel sizes we observe increased average performance, especially for the direct depth and smoothness indicators, while the boundary indicators only slightly benefit from lower kernel sizes. Nonetheless, when considering all indicators jointly, increasing kernel sizes achieve a balanced and gradual performance increase.

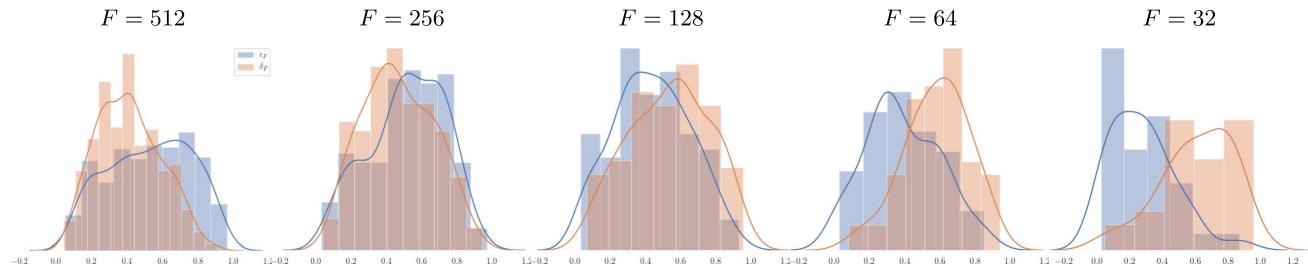


FIGURE 4. The learned encoder ϵ_F and decoder δ_F blending factors of the $K = 9$ model across the 5 hybrid skip connections of features F . From left to right the model's skip connection transition from the bottleneck to the output layers. It is observed that as we progress from the bottleneck ($F = 512$) towards the prediction layer ($F = 32$) a blending factor switch manifests across the HybridSkip connections used in each scale. The HybridSkip connections closer to the bottleneck focus on the structure given by the encoder (identity and low-pass) features, which nonetheless are closer to the bottleneck, while the skip connections closer to the output focus on the decoder features and their high-pass information.

stant blending factors, explicitly controlling the information exchange between the two feature maps \mathcal{E} and \mathcal{D} .

This way, the encoder and decoder blending factors would be $\epsilon = \mathbf{1} * \hat{\epsilon}$ and $\delta = \mathbf{1} * \hat{\delta}$, with $\mathbf{1}$ denoting a vector of ones with length F corresponding to the feature maps of each skip connection. Another approach would be to consider the blending factors as parameters of the model, and jointly optimize them with the convolutional UNet parameters. This would allow the model to adapt the blending factors to each separate feature instead. In this case, the blending factors are given by $\epsilon = \sigma(\hat{\epsilon})$ and $\delta = \sigma(\hat{\delta})$, with the hat symbols denoting the model's parameters, and σ being the sigmoid function constraining the blending factors to lie in the $[0, 1]$ range. When using learnable blending factors, the parameters $\hat{\epsilon}$ and $\hat{\delta}$ are initialized using a zero mean and unit variance normal distribution $\mathcal{N}(0, 1)$.

To perform an aggregated analysis among many metrics of different performance traits, namely direct depth, boundary preservation and smoothness, we use the following indicators derived from the metrics used in [1], which aggregate accuracy and error metrics:

$$\begin{aligned} i_d &= ((1.0 - \delta_{1.25}) \times RMSE)^{-1} \\ i_b &= ((1.0 - F_1^{1.0} + F_1^{0.25} + F_1^{0.5}/3) \times dbe^{\text{acc}})^{-1} \\ i_s &= ((1.0 - \alpha_{11.25} + \alpha_{22.50} + \alpha_{30.0}/3) \times RMSE^o)^{-1}, \end{aligned}$$

where F_1^t are the F_1 scores of the precision and recall boundary metrics at each threshold level t . The bar plots in Figure 3 present the results across different kernel sizes

and blending factors. For the former we experiment with $K = \{3, 5, 7, 9\}$ and for the latter, apart from the learnable blending factors, we also use the following explicit blendings $\{0.25, (0.25, 0.75), 0.5, (0.75, 0.25), 0.75\}$, with the tuples referring to (ϵ, δ) combinations. Two trends are observed, first, that an increasing kernel size provides consistent performance gains, and second, that the learnable blending factors are also a consistently good performer across different kernel sizes. Consequently, we use the $K = 9$ kernel size with learnable blending factors as our baseline hybrid skip connection UNet model.

From an interpretation perspective, analysing the learnable blending factors offers an insight on how the hybrid skip connections behave. We illustrate the distribution of the blending factor coefficients of the $K = 9$ model across its 5 skip connections in Figure 4. We observe an interesting and reasonable trend where the deeper layers focus on the structure offered by the incoming encoder features and their low-pass outputs (the encoder features in this case are not early encoder features), while as we progress towards the layers closer to the output, the blending factors indicate that the focus shifts on the predicted signal and its dominant edges, suppressing encoder features resulting into texture transfer.

B. COMPARISON WITH OTHER SKIP CONNECTIONS

We additionally compare the performance of the proposed hybrid skip connection to other approaches used for long range skip connections. More specifically, we present

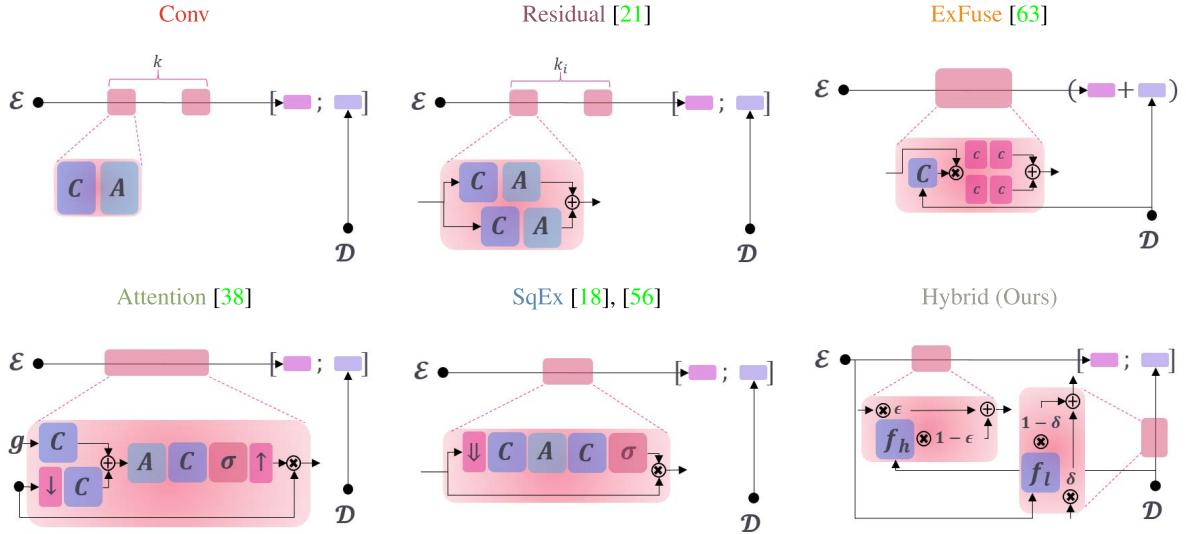


FIGURE 5. The architectures of the different skip connections used in the experiments, namely the straightforward convolution layer stack (**Conv**), the residual unit stack (**Residual**), the grid attention skip connection (**Attention**), the Squeeze-and-Excite (**SqEx**) and **ExFuse** skip connections, as well as our proposed **Hybrid** one. The operations included are convolution (C), activation (A), downsampling (\downarrow), upsampling (\uparrow), sigmoid (σ), separable convolutions (c), global average pooling (\Downarrow), elementwise tensor addition (\oplus) and multiplication (\otimes), concatenation ($:$), as well as low (f_l) and high pass (f_h) filtering. E and D denote the input encoder and decoder features of each skip connection, while g is the gate input used in [38].

TABLE 1. Direct depth metrics performance across all compared skip connections. Best three performers are denoted with bold faced **light green** (1st), **light blue** (2nd) and **light purple** (3rd) following the respective ranking order.

UNet Model	Direct Depth									
	Error ↓				Accuracy ↑					
	RMSE	RMSLE	AbsRel	SqRel	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$	$\delta_{1.05}$	$\delta_{1.1}$	
Vanilla [47]	0.4055	0.1158	0.1083	0.0649	89.43%	97.34%	99.09%	36.67%	62.12%	
Conv	0.3974	0.0670	0.1095	0.0663	89.43%	97.46%	99.09%	38.53%	61.79%	
Attention [38]	0.3974	0.0664	0.1074	0.0636	89.67%	97.61%	99.19%	35.90%	61.69%	
SqEx [18], [56]	0.3993	0.0672	0.1097	0.0672	89.57%	97.51%	99.09%	36.11%	61.65%	
ExFuse [63]	0.3913	0.0865	0.1043	0.0688	90.42%	97.60%	99.07%	40.34%	64.77%	
Residual [21]	0.3965	0.1068	0.1093	0.0679	89.61%	97.46%	99.13%	37.58%	62.22%	
Hybrid (Ours)	0.3937	0.0639	0.1010	0.0596	90.76%	97.72%	99.17%	38.75%	64.41%	

TABLE 2. Extra parameters, depth boundary and smoothness preservation metrics. Same colorization scheme as Table 1.

UNet Model	Depth Discontinuity					Depth Smoothness				Model Performance	
	Error ↓		Accuracy ↑			Accuracy ↑		Error ↓		Parameters ↓	
	dbe^{acc}	dbe^{comp}	$F_1^{0.25}$	$F_1^{0.5}$	F_1^1	$\alpha_{11.25^\circ}$	$\alpha_{22.5^\circ}$	α_{30°	$RMSE^\circ$		
Vanilla [47]	1.279	4.110	48.89%	42.14%	32.33%	63.02%	77.94%	83.12%	15.95	27.69M	
Conv	1.226	4.101	51.39%	45.92%	37.58%	62.88%	78.10%	83.35%	15.83	+6M (21.66%)	
Attention [38]	1.321	3.891	51.34%	45.66%	38.05%	62.60%	77.61%	82.93%	15.99	+2M (8.17%)	
SqEx [18], [56]	1.344	3.931	48.83%	41.12%	32.42%	66.22%	79.79%	84.54%	14.76	+88K (0.31%)	
ExFuse [63]	1.528	4.865	51.60%	46.20%	37.20%	63.86%	78.76%	83.84%	15.50	+18M (64.99%)	
Residual [21]	1.865	4.372	53.59%	48.28%	41.22%	62.89%	78.09%	83.38%	15.71	+4M (14.44%)	
Hybrid (Ours)	1.312	3.733	49.41%	42.94%	34.42%	64.24%	78.82%	83.86%	15.36	+1K (0.01%)	

results for a straightforward convolutional (Conv) skip connection stacking $k = 3 \times 3$ convolution layers, and the stacked residual unit skip connection [21] (Residual), where k_i units are stacked, with $i \in \{1, \dots, 5\}$ indicating the i th encoder-decoder layers. Apart from the stacked approaches, we also compare against the attention UNet [38] skip connection (Attention), and the NAS identified [56] Squeeze-and-Excite [18] (SqEx) skip connection. Finally, we adapt the ExFuse [63] skip connection for the UNet architecture, using the decoder features as the high-level feature map fed into the semantic embedding branch, and following it up with a

9×9 global convolution. Notably, compared to all other skip connections concatenating the encoder and decoder feature maps, ExFuse performs a residual skip connection by adding them. Illustrations describing each skip connection used in our experiments can be found in Figure 5.

Tables 1 and 2 present the performance of each skip connection on the M3D test set for the direct depth estimation metrics, as well as the boundary and smoothness preservation ones respectively. Evidently, the hybrid skip connection (learnable blending factors, $K = 9$) outperforms the other skip connections approaches for dense regression in two

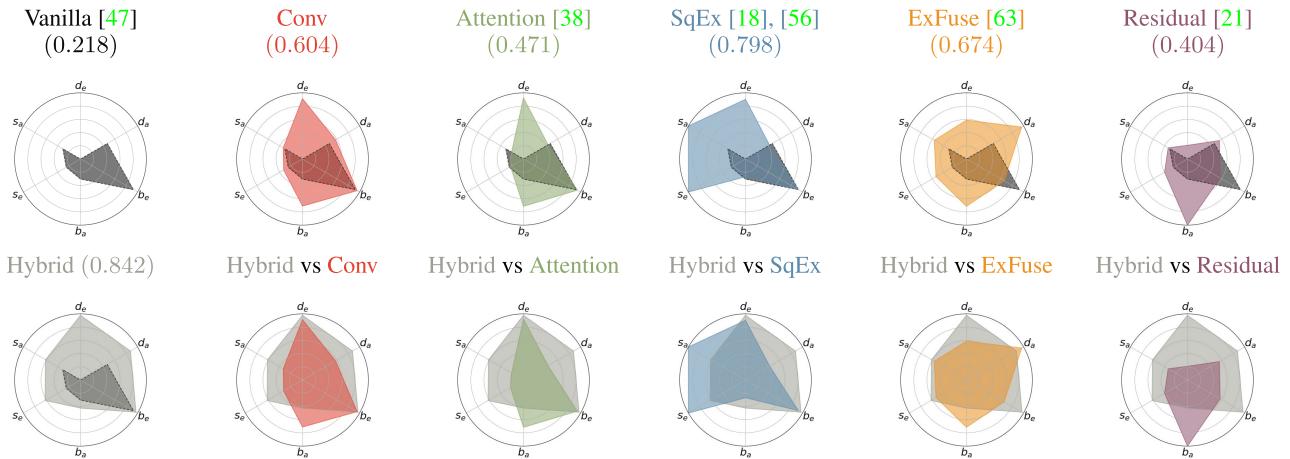


FIGURE 6. Normalized performance indicators for direct depth (d), boundary (b) and smoothness (s) preservation accuracy (a) and error (e) metrics. The numbers inside the parenthesis indicate the area covered by each different approach, with the largest area of the hybrid skip connection indicating its more balanced performance across all traits.

TABLE 3. Direct depth metrics performance metrics for the HybridSkip connection ablation experiments. Same colorization scheme as Table 1. Since \mathcal{F}_{blend} is a (learnable) blending of the encoder and decoder features, with no spatial filters applied, we duplicate the row and adjust the colorized ranking only with respect to the two different kernel sizes.

Model	Kernel	Direct Depth									
		RMSE	RMSLE	Error ↓	AbsRel	SqRel	Accuracy ↑	$\delta_{1.25}$	$\delta_{1.25^2}$	$\delta_{1.25^3}$	$\delta_{1.05}$
\mathcal{F}_{hybrid}	$K = 9$	0.3937	0.0639	0.1010	0.0596	90.76%	97.72%	99.17%	38.75%	64.41%	
		0.3921	0.0718	0.1095	0.0658	89.24%	97.50%	99.14%	38.47%	62.58%	
		0.4105	0.0678	0.1090	0.0660	89.36%	97.43%	99.07%	34.77%	61.38%	
\mathcal{F}_{blend}	N / A	0.4006	0.0670	0.1095	0.0649	89.35%	97.54%	99.12%	35.78%	61.09%	
		0.4006	0.0670	0.1095	0.0649	89.35%	97.54%	99.12%	35.78%	61.09%	
		0.3912	0.0646	0.1039	0.0611	90.40%	97.69%	99.16%	36.46%	62.66%	
\mathcal{F}_{hybrid}	$K = 7$	0.4017	0.0684	0.1106	0.0698	88.86%	97.22%	98.96%	39.57%	62.76%	
		0.4002	0.1200	0.1106	0.0681	89.06%	97.36%	99.05%	36.49%	61.43%	

TABLE 4. Depth boundary and smoothness preservation metrics for the HybridSkip connection ablation experiments. Same colorization and arrangement scheme as Table 1.

Model	Kernel	Depth Discontinuity						Depth Smoothness			Error ↓ $RMSE^o$
		Error ↓ dbe^{acc}	Error ↓ dbe^{comp}	$F_0^{0.25}$	Accuracy ↑ $F_0^{0.5}$	F_1^1	Accuracy ↑ $\alpha_{11.25^o}$	Accuracy ↑ $\alpha_{22.5^o}$	Accuracy ↑ α_{30^o}		
\mathcal{F}_{hybrid}	$K = 9$	1.312	3.733	49.41%	42.94%	34.42%	64.24%	78.82%	83.86%	15.36	
		1.360	3.960	50.26%	43.60%	35.62%	63.76%	78.43%	83.52%	15.65	
		1.371	3.833	47.86%	41.07%	30.56%	63.05%	77.90%	83.04%	15.93	
\mathcal{F}_{blend}	N / A	1.308	4.098	50.74%	44.66%	36.25%	63.09%	78.04%	83.23%	15.90	
		1.308	4.098	50.74%	44.66%	36.25%	63.09%	78.04%	83.23%	15.90	
		1.661	4.472	51.10%	44.14%	35.95%	63.97%	78.72%	83.79%	15.45	
\mathcal{F}_{hybrid}	$K = 7$	1.377	3.786	51.36%	44.54%	35.97%	62.99%	77.80%	82.97%	16.04	
		1.266	3.941	52.14%	45.94%	37.78%	61.99%	77.42%	82.72%	16.33	

aspects. First, it offers the largest gain in terms of improving direct depth estimation performance. Second, it additionally offers the more balanced performance increase compared to a vanilla UNet [47] across the secondary – competing – performance traits. Finally, it does so at a reduced extra parameter cost (last column in Table 2). While SqEx (Residual) offers an important performance boost for preserving the depth map’s smoothness (boundaries), it does so at the expense of preserving boundaries (smoothness). The (second) better balanced approach is that of ExFuse which manages to offer reasonable performance gains across all performance traits. These comparisons can be more easily discerned in Figure 6 that illustrates radar plots across different

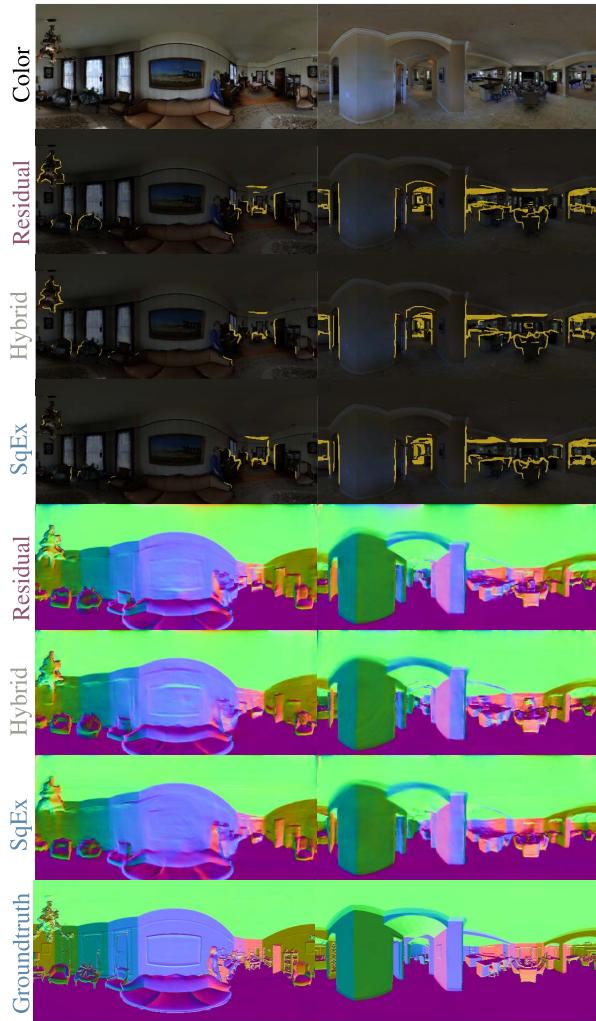
normalized accuracy (a) and error (e) indicators for all performance axes:

$$\begin{aligned} d_a &= (0.2 \times (\delta_{1.05} + \delta_{1.1} + \delta_{1.25} + \delta_{1.25^2} + \delta_{1.25^3}))^{-1} \\ b_a &= (F_1^{1.0} + F_1^{0.5} + F_1^{0.25})/3)^{-1}, s_a = (\alpha_{11.25^o} + \alpha_{22.5^o} + \alpha_{30^o})/3)^{-1} \\ d_e &= (RMSE \times RMSLE)^{-1}, s_e = (RMSE^o)^{-1} \\ b_e &= (dbe^{acc} \times dbe^{comp})^{-1}. \end{aligned}$$

Figure 7 presents qualitative results of our $K = 9$ hybrid skip model, compared to the SqEx and Residual models. The latter are the better performing models in terms of surface and boundary preservation respectively, but clearly showcase the difficulty in achieving a balance between these two traits,

TABLE 5. Direct depth metrics performance across different architectures. Same colorization scheme as Table 1.

Model	Direct Depth								
	RMSE	RMSLE	AbsRel	SqRel	$\delta_{1.25}$	$\delta_{1.25^2}$	Accuracy ↑		
Vanilla UNet [47]	0.4055	0.1158	0.1083	0.0649	89.43%	97.34%	99.09%	36.67%	62.12%
BiFuse [54]	0.4243	0.0668	0.1142	0.1427	90.68%	97.22%	98.71%	41.18%	65.36%
HoHoNet [51]	0.3718	0.0603	0.0998	0.0871	92.14%	97.80%	99.08%	42.65%	69.72%
UNet++ [66]	0.4544	0.0736	0.1236	0.1274	87.81%	96.65%	98.49%	37.18%	60.94%
SqEx UNet++ [18], [66]	0.4507	0.0716	0.1231	0.1526	88.76%	96.80%	98.57%	37.20%	62.87%
HybridSkip UNet (Ours)	0.3937	0.0639	0.1010	0.0596	90.76%	97.72%	99.17%	38.75%	64.41%

**FIGURE 7.** Qualitative results on the M3D test set samples. From top to bottom: i) input color image, ii-iv) predicted depth boundaries for the Residual, Hybrid and SqEx models, v-vii) predicted normal maps for the same models, and viii) the groundtruth normal maps.

as their improved performance on one, translates to a reduced performance on the other. In contrast, the hybrid skip model strikes a better balance in preserving both traits.

C. HYBRID SKIP ABLATION

We additionally perform an ablation study of the three functional components that jointly formulate the HybridSkip connection. First, we examine a scenario where only the learnable blending of the encoder and decoder features is

introduced, denoted as $\mathcal{F}_{blend} = H_i([\delta \mathcal{D} + (1-\delta)\mathcal{E}; \epsilon \mathcal{E} + (1-\epsilon)\mathcal{D}])$. Then we also conduct two experiments where only a single filter is applied either only at the encoder features (low pass) or the decoder ones (high pass) respectively denoted as $\mathcal{F}_{low} = H_i([\mathbf{f}_l(\mathcal{E}); \mathcal{D}])$ and $\mathcal{F}_{high} = H_i([\mathcal{E}; \mathbf{f}_h(\mathcal{D})])$. The results for the two larger kernels (*i.e.* $K = 7, K = 9$) are presented in Tables 3 and 4, where the former includes the metrics related to direct depth estimation performance and the latter includes the metrics related to the secondary traits, depth smoothness and boundary preservation.

While each functional component in isolation may improve performance along a single axis or metric, it is evident that their combination leads to the most balanced performance boost. Interestingly, we observe that the preservation of structural edges is easy to achieve, but at the expense of smoothness or direct performance, something that is better mitigated when all components co-exist as a HybridSkip connection. However, the discrepancy with respect to boundary preservation between $K = 7$ and $K = 9$, with the smaller kernel showing improved accuracy, indicates the selection of the kernel parameters should be tuned on a per-dataset basis.

D. OTHER ARCHITECTURES

Finally, we examine the behavior of other UNet architectures, and established models for 360° depth estimation with respect to their preservation of additional estimated signal traits. Specifically, for the former, we use UNet++ [66] and a SqEx UNet++, which is a UNet++ extended with squeeze-and-excite [18] skip connections, which were found to be the most balanced alternative in the skip comparison experiments in Section IV-B. For the latter, we employ the state-of-the-art BiFuse [54] and HoHoNet [51] models. All experiments are done using the same training scheme and our rich supervision, as presented in the previous experiments, essentially only switching the architecture for each different experiment, even for the BiFuse and HoHoNet models, for a fairer comparison.

Tables 5 and 6 present the direct and secondary metrics respectively, including the baseline UNet and our proposed vanilla UNet variant with HybridSkip connections. While HoHoNet, a model specialized for the 360° domain, produces high quality depth estimation, followed by our model, its behaviour with respect to preserving discontinuity and smoothness is largely reduced, showcasing worse performance even compared to the vanilla UNet. On the other hand, BiFuse largely favours smoothness instead of boundary preservation, whereas both UNet++ variants naturally

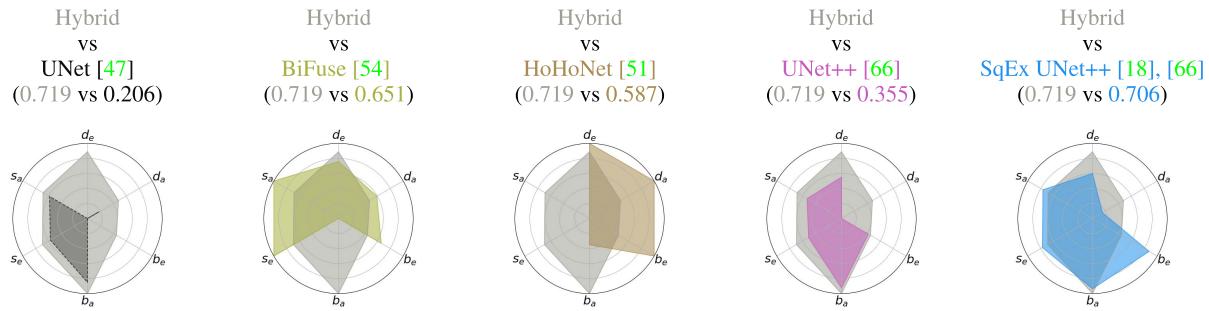


FIGURE 8. Same scheme as Figure 6, with larger numbers inside the parenthesis indicating the area covered by each different approach, with larger areas indicating more balanced performance across all traits.

TABLE 6. Number of parameters, depth boundary and smoothness preservation metrics for different architectures. Same colorization scheme as Table 1.

Model	Depth Discontinuity					Depth Smoothness			
	Error ↓ d_{be}^{acc}	Error ↓ d_{be}^{comp}	Accuracy ↑ $F_1^{0.25}$	Accuracy ↑ $F_1^{0.5}$	Accuracy ↑ F_1^1	Accuracy ↑ $\alpha_{11.25^\circ}$	Accuracy ↑ $\alpha_{22.5^\circ}$	Accuracy ↑ α_{30°	Error ↓ $RMSE^\circ$
Vanilla UNet [47]	1.279	4.110	48.89%	42.14%	32.33%	63.02%	77.94%	83.12%	15.95
BiFuse [54]	1.321	3.580	41.42%	33.83%	28.70%	69.73%	80.98%	84.99%	13.89
HoHoNet [51]	1.109	4.019	45.10%	36.33%	30.50%	55.63%	72.86%	79.10%	18.75
UNet++ [66]	1.235	3.986	48.18%	42.61%	34.27%	62.98%	77.20%	82.37%	16.27
SqEx UNet++ [18], [66]	1.144	3.982	48.00%	42.49%	34.82%	66.14%	79.15%	83.85%	14.98
HybridSkip UNet (Ours)	1.312	3.733	49.41%	42.94%	34.42%	64.24%	78.82%	83.86%	15.36

show improved boundary preservation. As also seen in the experiments comparison different skip connections, the SqEx UNet++ balances the two secondary traits better, offering good results for smoothness as well, compared to the pure UNet++ architecture, overcoming the deficits of skip connections. Nonetheless, its direct depth estimation performance is still at similar levels to UNet++, and inferior to the better performing 360° depth estimation models. Overall, our hybrid skip connection vanilla UNet model, offers the more balanced performance across direct and secondary trait metrics, as illustrated in Figure 8, with only the SqEx UNet++ model coming close in terms of balanced performance.

V. CONCLUSION

In this work we have designed a hybrid skip connection for the UNet architecture which relies on long range skip connections fusing features with a large semantic and spectral gap. The simultaneous blending and spatial nature of the hybrid skip connection allows for a balanced performance boost across all performance traits for depth estimation, a dense regression task, with minimal parameter overhead. These results indicate that it may be worth exploring the hybrid image concept in the various existing UNet modifications [19], [36], [43], [52], [66], or even CNN architectures without long range skip connections, with a recent report [4] providing interesting evidence about their interplay with CNNs. Potential explorations may include short range skip connections (e.g. residual units), or integration within basic CNN buildings blocks (e.g. squeeze-and-excite operations or Octave Convolutions [8]). One limitation is the design of the filters themselves, which are currently performed on the spatial domain and whose parameters remain fixed during training. While larger kernel sizes may provide a more balanced performance improvement as illustrated in

Figure 3, each output signal's distribution may be more tuned to specific kernel parameters (e.g. $K = 7$ showing better boundary preservation in Table 4). Spectral or learnable filtering may allow models to better adapt to the task and data at hand. Further, experimenting with adaptive blending will open up dynamic dual feature representations instead of the fixed blending factors that statically choose representations at the end of the model's training. It also remains to be seen if these balanced skip connections also boost performance in other downstream tasks like segmentation.

REFERENCES

- [1] G. Albanis, N. Zioulis, P. Drakoulis, V. Gkitsas, V. Sterzentsenko, F. Alvarez, D. Zarpalas, and P. Daras, "Pano3D: A holistic benchmark and a solid baseline for 360 ° depth estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 3727–3737.
- [2] B. Attal, S. Ling, A. Gokaslan, C. Richardt, and J. Tompkin, "MatryOD-Shka: Real-time 6DoF video view synthesis using multi-sphere images," in *Computer Vision*, A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 441–459.
- [3] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
- [4] A. Borji, "CNNs and transformers perceive hybrid images similar to humans," 2022, *arXiv:2203.11678*.
- [5] T. F. Brady and A. Oliva, "Spatial frequency integration during active perception: Perceptual hysteresis when an object recedes," *Frontiers Psychol.*, vol. 3, p. 462, Oct. 2012.
- [6] H.-X. Chen, K. Li, Z. Fu, M. Liu, Z. Chen, and Y. Guo, "Distortion-aware monocular depth estimation for omnidirectional images," *IEEE Signal Process. Lett.*, vol. 28, pp. 334–338, 2021.
- [7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.
- [8] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, Y. Shuicheng, and J. Feng, "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3435–3444.

- [9] B. Cheng, R. Xiao, J. Wang, T. Huang, and L. Zhang, "High frequency residual learning for multi-scale image classification," in *Proc. 30th Brit. Mach. Vis. Conf. (BMVC)*, 2020, pp. 1–14.
- [10] M. Eder, T. Price, T. Vu, A. Bapati, and J.-M. Frahm, "Mapped convolutions," 2019, *arXiv:1906.11096*.
- [11] (2019). Falcon, WA, USA. *Pytorch Lightning*. [Online]. Available: <https://github.com/PyTorchLightning/pytorch-lightning>
- [12] B. Y. Feng, W. Yao, Z. Liu, and A. Varshney, "Deep depth estimation on 360° images with a double quaternion loss," in *Proc. Int. Conf. 3D Vis. (3DV)*, Nov. 2020, pp. 524–533.
- [13] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3828–3838.
- [14] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon, "3D packing for self-supervised monocular depth estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2485–2494.
- [15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
- [16] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.
- [17] J. Hu, M. Ozay, Y. Zhang, and T. Okatanı, "Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 1043–1051.
- [18] J. Hu, L. Shen, and G. Sun, "Squeeze- and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
- [19] H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y.-W. Chen, and J. Wu, "UNet 3+: A full-scale connected UNet for medical image segmentation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 1055–1059.
- [20] J. Huang, A. B. Lee, and D. Mumford, "Statistics of range images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2000, pp. 324–331.
- [21] N. Ibtehaz and M. S. Rahman, "MultiResUNet: Rethinking the U-Net architecture for multimodal biomedical image segmentation," *Neural Netw.*, vol. 121, pp. 74–87, Jan. 2020.
- [22] A. Ignatov et al., "Fast and accurate single-image depth estimation on mobile devices, mobile AI 2021 challenge: Report," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 2545–2557.
- [23] F. Isensee, J. Petersen, A. Klein, D. Zimmerer, F. Paul Jaeger, S. Kohl, J. Wasserthal, G. Köhler, T. Norajitra, J. S. Wirkert, and H. K. Maier-Hein, "NNU-Net: Self-adapting framework for u-net-based medical image segmentation," 2018, *arXiv:1809.10486*.
- [24] S. Jetley, N. A. Lord, N. Lee, and P. H. Torr, "Learn to pay attention," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–14.
- [25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [26] T. Koch, L. Liebel, F. Fraundorfer, and M. Korner, "Evaluation of CNN-based single-image depth estimation methods," in *Proc. Eur. Conf. Comput. Vis. (ECCV) Workshops*, Sep. 2018, pp. 1–17.
- [27] V. Laprévote, A. Oliva, C. Delerue, P. Thomas, and M. Boucart, "Patients with schizophrenia are biased toward low spatial frequency to decode facial expression at a glance," *Neuropsychologia*, vol. 48, no. 14, pp. 4164–4168, Dec. 2010.
- [28] C. Li, Y. Tan, W. Chen, X. Luo, Y. Gao, X. Jia, and Z. Wang, "Attention UNet++: A nested attention-aware U-Net for liver CT image segmentation," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 345–349.
- [29] T. Lindeberg, *Scale-Space Theory in Computer Vision*, vol. 256. Cham, Switzerland: Springer, 2013.
- [30] G. Liu, A. F. Reda, J. K. Shih, T.-C. Wang, A. Tao, and A. Catanzaro, "Image inpainting for irregular holes using partial convolutions," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 1–16.
- [31] L. Liu, J. Cheng, Q. Quan, F.-X. Wu, Y.-P. Wang, and J. Wang, "A survey on U-shaped networks in medical image segmentations," *Neurocomputing*, vol. 409, pp. 244–258, Oct. 2020.
- [32] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.
- [33] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 4040–4048.
- [34] S. Mehta, E. Mercan, J. Bartlett, D. Weaver, G. J. Elmore, and A. Shapiro, "Y-Net: Joint segmentation and classification for diagnosis of breast biopsy images," in *Medical Image Computing and Computer Assisted Intervention*, A. F. Frangi, J. A. Schnabel, C. Davatzikos, C. Alberola-López, and G. Fichtinger, Eds. Cham, Switzerland: Springer, 2018, pp. 893–901.
- [35] S. Miillet, R. Caldara, and P. G. Schyns, "Local jekyll and global Hyde: The dual identity of face identification," *Psychol. Sci.*, vol. 22, no. 12, pp. 1518–1526, Dec. 2011.
- [36] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proc. 4th Int. Conf. 3D Vis. (3DV)*, Oct. 2016, pp. 565–571.
- [37] (2021). Moai: Accelerating Modern Data-Driven Workflows. [Online]. Available: <https://github.com/ai-in-motion/moai>
- [38] O. Oktay, J. Schlemper, L. Le Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention U-Net: Learning where to look for the pancreas," 2018, *arXiv:1804.03999*.
- [39] A. Oliva, A. Torralba, and P. G. Schyns, "Hybrid images," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 527–532, 2006.
- [40] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 8026–8037.
- [41] G. Pintore, M. Agus, E. Almansa, J. Schneider, and E. Gobbetti, "SliceNet: Deep dense depth estimation from a single indoor panorama using a slice-based representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11536–11545.
- [42] N. S. Punn and S. Agarwal, "Modality specific U-Net variants for biomedical image segmentation: A survey," 2021, *arXiv:2107.04537*.
- [43] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane, and M. Jagersand, "U2-Net: Going deeper with nested U-structure for salient object detection," *Pattern Recognit.*, vol. 106, Oct. 2020, Art. no. 107404.
- [44] M. Ramamonijsa, M. Firman, J. Watson, V. Lepetit, and D. Turmukhambetov, "Single image depth prediction with wavelet decomposition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 11089–11098.
- [45] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski, "Kornia: An open source differentiable computer vision library for PyTorch," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 3674–3683.
- [46] G. Riegler and V. Koltun, "Free view synthesis," in *Computer Vision*, A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm, Eds. Cham, Switzerland: Springer, 2020, pp. 623–640.
- [47] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput.-Assist. Intervent.* Springer, 2015, pp. 234–241.
- [48] A. G. Roy, N. Navab, and C. Wachinger, "Recalibrating fully convolutional networks with spatial and channel 'squeeze and excitation' blocks," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 540–549, Aug. 2018.
- [49] E. Schonfeld, B. Schiele, and A. Khoreva, "A U-Net based discriminator for generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8207–8216.
- [50] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," 2014, *arXiv:1403.1687*.
- [51] C. Sun, M. Sun, and H.-T. Chen, "HoHoNet: 360 indoor holistic understanding with latent horizontal features," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 2573–2582.
- [52] J. M. J. Valanarasu, V. A. Sindagi, I. Hacihaliloglu, and V. M. Patel, "KiUNet: Overcomplete convolutional architectures for biomedical image and volumetric segmentation," 2020, *arXiv:2010.01663*.
- [53] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang, "Residual attention network for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3156–3164.
- [54] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, "BiFuse: Monocular 360 depth estimation via bi-projection fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 462–471.

- [55] R. Wang, D. Geraghty, K. Matzen, R. Szeliski, and J.-M. Frahm, "VPLNet: Deep single view normal estimation with vanishing points and lines," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 689–698.
- [56] Y. Weng, T. Zhou, Y. Li, and X. Qiu, "NAS-UNet: Neural architecture search for medical image segmentation," *IEEE Access*, vol. 7, pp. 44247–44257, 2019.
- [57] X. Xia and B. Kulic, "W-Net: A deep model for fully unsupervised image segmentation," 2017, *arXiv:1711.08506*.
- [58] Y. Yang, Y. Wang, C. Zhu, M. Zhu, H. Sun, and T. Yan, "Mixed-scale UNet based on dense atrous pyramid for monocular depth estimation," *IEEE Access*, vol. 9, pp. 114070–114084, 2021.
- [59] J. C. Ye and W. K. Sung, "Understanding geometry of encoder-decoder CNNs," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7064–7073.
- [60] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. Huang, "Free-form image inpainting with gated convolution," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4471–4480.
- [61] T. Yuan, Y. Wang, K. Xu, R. R. Martin, and S. Hu, "Two-layer QR codes," *IEEE Trans. Image Process.*, vol. 28, no. 9, pp. 4413–4428, Sep. 2019.
- [62] X. Zhang, S. Fanello, Y.-T. Tsai, T. Sun, T. Xue, R. Pandey, S. Orts-Escalano, P. Davidson, C. Rhemann, P. Debevec, J. T. Barron, R. Ramamoorthi, and W. T. Freeman, "Neural light transport for relighting and view synthesis," *ACM Trans. Graph.*, vol. 40, no. 1, pp. 1–17, Feb. 2021.
- [63] Z. Zhang, X. Zhang, C. Peng, X. Xue, and J. Sun, "Exfuse: Enhancing feature fusion for semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 269–284.
- [64] T. Zhao, S. Pan, W. Gao, C. Sheng, Y. Sun, and J. Wei, "Attention UNet++ for lightweight depth estimation from sparse depth samples and a single RGB image," *Vis. Comput.*, vol. 38, no. 5, pp. 1–12, May 2022.
- [65] L. Zhou and M. Kaess, "Windowed bundle adjustment framework for unsupervised learning of monocular depth estimation with U-Net extension and clip loss," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 3283–3290, Apr. 2020.
- [66] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, D. Stoyanov, Z. Taylor, G. Carneiro, T. Syeda-Mahmood, A. Martel, L. Maier-Hein, J. Manuel R.S. Tavares, A. Bradley, J. P. Papa, V. Belagiannis, J. C. Nascimento, Z. Lu, S. Conjeti, M. Moradi, H. Greenspan, and A. Madabhushi, Eds. Cham, Switzerland: Springer, 2018, pp. 3–11. International Publishing.
- [67] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras, "Omnidepth: Dense depth estimation for indoors spherical panoramas," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 448–465.



NIKOLAOS ZIOULIS received the degree from the Aristotle University of Thessaloniki, in 2012. He has been an Electrical and Computer Engineer working with the Information Technologies Institute (ITI) of the Centre for Research and Technology Hellas (CERTH), since October 2013. His research interests include the intersection of computer vision and graphics technologies, and more specifically, in volumetric 3D capturing and rendering, 3D scene understanding, and tele-immersive applications.



GEORGIOS ALBANIS received the degree from the Electrical and Computer Engineering Department, Aristotle University of Thessaloniki, in 2015. His master thesis was about designing and developing a smart metering board, providing accurate electrical measurements and communicating them to a web-server. He worked with Noriker Power Ltd., U.K., as a Project Engineer with broad duties encompassing engineering design, controls, and monitoring; site commissioning; and software development. Particularly, he was heavily involved in

software development, including back-end systems both on real-time controllers and cloud-based databases, and on user interfaces (across several devices). He has been an Electrical and Computer Engineer working with the Information Technologies Institute (ITI) of the Centre for Research and Technology Hellas (CERTH), since March 2018. After joining CERTH, he has been involved in four European research and development projects. His current research interests include 3D/4D computer vision, deep learning, and multimodal mixed reality (augmented and virtual reality).



PETROS DRAKOULIS received the B.Sc. degree in information technology from Alexander TEI and the M.Sc. degree (Hons.) in digital media and computational intelligence from the Aristotle University of Thessaloniki. In 2018, he joined the Visual Computing Laboratory, ITI-CERTH, where he works as a Research Assistant and a Software Developer. His main research interests include software engineering, visual computing, machine learning, and graphics.



FEDERICO ALVAREZ (Member, IEEE) received the degree (Hons.) in telecom engineering and the Ph.D. degree (*cum laude*) from the Universidad Politécnica de Madrid (UPM), in 2003 and 2009, respectively. He is currently an Associate Professor and the Head of the Visual Telecommunications Applications Research Group (GATV), UPM. He is the author and coauthor of over 80 papers and several books, book chapters, and holds patents in the field of ICT networks and audiovisual technologies. He had participated in national and international standardization forums (DVB and CENELEC TC206). He is a member of the program committee of several scientific conferences.



DIMITRIOS ZARPALAS has joined the Information Technologies Institute, in 2007, and is currently working as a Postdoctoral Research Associate. His current research interests include real time tele-immersion applications (3D reconstruction of moving humans and their compression); 3D computer vision; 3D medical image processing; shape analysis of anatomical structures; and 3D object recognition, motion capturing, and evaluation, while in the past has also worked in indexing, search, and retrieval and classification of 3D objects and 3D model watermarking. His involvement with those research areas has led to the coauthoring of three book chapter, one article in refereed journals, and 42 papers in international conferences. He has been involved in more than ten research projects funded by EC and a Greek Secretariat of research and technology. He is a member of the Technical Chamber of Greece.



PETROS DARAS (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the Aristotle University of Thessaloniki, Greece, in 1999, 2002, and 2005, respectively. He is currently a Research Director and the Chair of the Visual Computing Laboratory coordinating the research effort of more than 80 scientists and engineers. His main research interests include visual content processing, multimedia indexing, and machine learning. His involvement with those research areas has led to the coauthoring of more than 300 articles in refereed journals and international conferences. He has been involved in more than 50 projects, funded by the EC and the Greek Ministry of Research and Technology.



Hybrid images

Aude Oliva*
MIT-BCS

Antonio Torralba†
MIT-CSAIL

Philippe. G. Schyns‡
University of Glasgow



Figure 1: A hybrid image is a picture that combines the low-spatial frequencies of one picture with the high spatial frequencies of another picture producing an image with an interpretation that changes with viewing distance. In this figure, the people may appear sad, up close, but step back a few meters and look at the expressions again.

Abstract

We present *hybrid images*, a technique that produces static images with two interpretations, which change as a function of viewing distance. Hybrid images are based on the multiscale processing of images by the human visual system and are motivated by masking studies in visual perception. These images can be used to create compelling displays in which the image appears to change as the viewing distance changes. We show that by taking into account perceptual grouping mechanisms it is possible to build compelling hybrid images with stable percepts at each distance. We show examples in which hybrid images are used to create textures that become visible only when seen up-close, to generate facial expressions whose interpretation changes with viewing distance, and to visualize changes over time within a single picture.

Keywords: Hybrid images, human perception, scale space

1 Introduction

Here we exploit the multiscale perceptual mechanisms of human vision to create visual illusions (*hybrid images*) where two different interpretations of a picture can be perceived by changing the viewing distance or the presentation time. We use and extend the method originally proposed by Schyns and Oliva [1994; 1997; 1999]. Fig. 1 shows an example of a hybrid image assembled from two images

in which the faces displayed different emotions. High spatial frequencies correspond to faces with "sad" expressions. Low spatial frequencies correspond to the same faces with "happy" and "surprise" emotions (i.e., the emotions are, from left to right: happy, surprise, happy and happy). To switch from one interpretation to the other one can step away a few meters from the picture.

Artists have effectively employed low spatial frequency manipulation to elicit a percept that changes when relying on peripheral vision (e.g., [Livingstone 2000; Dali 1996]). Inspired by this work, Setlur and Gooch [2004] propose a technique that creates facial images with conflicting emotional states at different spatial frequencies. The images produce subtle expression variations with gaze changes. In this paper, we demonstrate the effectiveness of *hybrid images* in creating images with two very different possible interpretations.

Hybrid images are generated by superimposing two images at two different spatial scales: the low-spatial scale is obtained by filtering one image with a low-pass filter; the high spatial scale is obtained by filtering a second image with a high-pass filter. The final image is composed by adding these two filtered images. Note that *hybrid images* are a different technique than *picture mosaics* [Silvers 1997]. *Picture mosaics* have two interpretations: a local one (which is given by the content of each of the pictures that compose the mosaic) and a global one (which is best seen at some predefined distance). Hybrid images, however, contain two coherent global image interpretations, one of which is of the low spatial frequencies, the other of high spatial frequencies.

We illustrate this technique with several proof-of-concept examples. We show how this technique can be applied to create face pictures that change expression with viewing distance, to display two configurations of a scene in a single picture, and to present textures that disappear when viewed at a distance.

2 The design of hybrid images

A hybrid image (H) is obtained by combining two images (I_1 and I_2), one filtered with a low-pass filter (G_1) and the second one fil-

Copyright © 2006 by the Association for Computing Machinery, Inc.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail permissions@acm.org.

© 2006 ACM 0730-0301/06/0700-0527 \$5.00

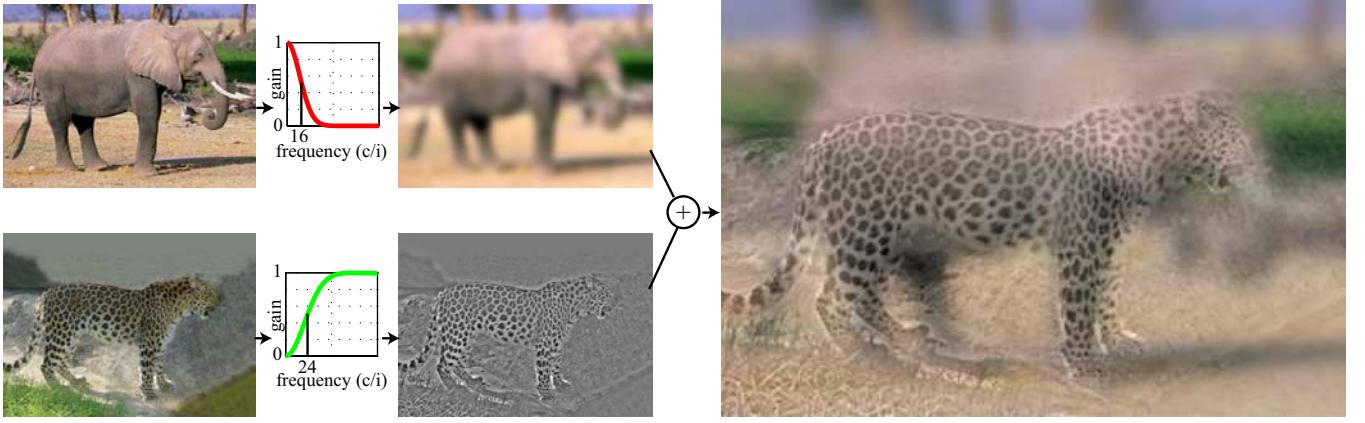


Figure 2: hybrid images are generated by superimposing two images at two different spatial scales: the low-spatial scale is obtained by filtering one image with a low-pass filter, and the high spatial scale is obtained by filtering a second image with a high-pass filter. The final hybrid image is composed by adding these two filtered images.

tered with a high-pass filter ($1 - G_2$): $H = I_1 \cdot G_1 + I_2 \cdot (1 - G_2)$, the operations are defined in the Fourier domain. Hybrid images are defined by two parameters: the frequency cut of the low resolution image (the one to be seen at a far distance), and the frequency cut of the high resolution image (the one to be seen up close). An additional parameter can be added by introducing a different gain for each frequency channel. For the hybrids shown in this paper we have set the gain to 1 for both spatial channels. We use gaussian filters (G_1 and G_2) for the low-pass and the high-pass filters. We define the cut-off frequency of each filter as the frequency for with the amplitude gain of the filter is 1/2.

Figure 2 illustrates the process used to create one hybrid image. The distance at which each component of a hybrid image is best seen and the distance at which the hybrid percept alternates can be fully determined as a function of the image size and the cut-off frequencies of the filters (expressed in cycles/image¹). When viewing the images in this paper, switch between interpretations by stepping a few meters away from the picture. Note that the larger you display the images, the farther you will have to go in order to see the alternative image interpretation.

2.1 The perception of hybrid images

In the following section we describe the motivation behind hybrid images, as they relate to studies in human perception. We will provide the framework for understanding the mechanisms involved in perception of double image percepts.

Visual psychophysics research has shown that human observers are able to comprehend the meaning of a novel image within a short glance (100 msec [Potter 1975]). This phenomenal performance of rapid image understanding can be experienced while watching fast scene edits in an action movie or in a music video. Research in human perception has suggested that image understanding efficiency is based on a multi-scale, global to local analysis of the visual input [Burt and Adelson 1983; Majaj et al. 2002]: an initial

analysis of the global structure and the spatial relationships between components guides the analysis of local details [Schyns and Oliva 1994; Watt 1987]. The global precedence hypothesis of image analysis (“seeing the forest before the trees”, [Navon 1977]) implies a coarse-to-fine frequency analysis of an image, where the low spatial frequency components, which are contrasted and carried by the fast magnocellular pathway, dominate early visual processing [Hughes et al. 1996; Lindeberg 1993; Parker et al. 1992; Schyns and Oliva 1994; Sugase et al. 1999].

Using hybrid stimuli, Schyns and Oliva [1994] tested the role that spatial frequency bands play for the interpretation of natural images. When the task required identifying a scene image quickly, human observers interpreted the low spatial frequency band (at a frequency cutoff of 8 cycles/image) before the high spatial frequency band (from 24 cycles/image): when showed hybrid images for 30 ms only, observers identified the low spatial scale (e.g., they would answer “cheetah” when presented with the image from Fig. 3) whereas for 150 ms duration, they identified the high spatial scale first (e.g., tiger in Fig. 3). Interestingly, participants were unaware that the visual stimuli had two interpretations. Additional experiments suggested that the spatial frequency band preferentially selected for interpreting an image depends on the task the viewer must solve. Using hybrid faces similar to the one in Fig. 5.b, Schyns and Oliva [1999] showed that when participants were asked to determine the emotion of an hybrid face image displayed for only 50 ms (happy, angry or neutral), they selected the low spatial frequency face (angry in Fig. 5.b), but when they had to determine the gender of the same image, they used the low spatial frequency components of the hybrid as often as the high. Again, participants did not report noticing presence of two emotions or two genders in these images. These results demonstrated that the selection of frequency bands for fast image recognition is a flexible mechanism: The image analysis might still unfold according to a low to high spatial scale processing, but human observers are able to quickly select the frequency band, low or high, that conveyed the most information to solve a given task and interpret the image. Importantly, when selecting a spatial frequency, observers were not conscious of the information in the other spatial scale.

¹We use the units *cycle/image* for spatial frequencies as they are independent of the image resolution. The output of a gaussian filter with a cutoff frequency of 16 *cycles/image* will be the same independently of the resolution of the original image. The units *cycle/degree of visual angle* are used to describe the resolution observed when the image has a fixed size and is seen from a fixed distance.

In the study of human perception, hybrid images allow characterizing the role of different frequency channels for image recognition, and evaluate the time course of spatial frequency processing. Hybrid images provide a new paradigm in which images interpretation can be modulated by playing with viewing distance or presenta-



Figure 3: Perceptual grouping between edges and blobs. The three images are perceived as a tiger when seen up-close and as a cheetah from far away. The differences among the three images is the degrees of alignment between the edges and blobs. Image a) contains two images superimposed without alignment. In image b), the eyes are aligned. And in image c), the head pose and the locations of eyes and mouth are aligned. Under proper alignment, the residual frequency band does not manage to build a percept. When seen up-close, it is difficult to see the cheetah's face, which is perfectly masked by the tiger's face. From far away, the tiger's edges are assimilated to the cheetah's face.



Figure 4: Color at high spatial frequencies is used to enhance the bicycle up-close. From a distance, one sees a motorcycle. The shape of the motorcycle is interpreted as shadows up-close.

tion time. For a given distance of viewing, or a given temporal frequency a particular band of spatial frequency dominates visual processing. Visual analysis of the hybrid image still unfolds from global to local perception, but within the selected frequency band, for a given viewing distance, the observer will perceive the global structure of the hybrid first (that the image in Fig. 3 represents a head), and take an additional hundred milliseconds to organize the local information into a coherent percept (organization of blobs if the image is viewed at a far distance, or organization of edges for close viewing).

2.2 Rules of perceptual grouping and hybrid images

In theory, one can combine any two images to create a hybrid picture. In practice, aesthetically pleasing hybrid images require following some rules that we describe in this section. In successful

Hybrid images, when one percept dominates, consciously switching to the alternative interpretation becomes almost impossible. Only when the viewing distance changes can we switch to the alternative interpretation. In a hybrid image it is important that the alternative image is perceived as noise (lacking internal organization) or that it blends with the dominant subband.

Rules of perceptual grouping modulate the effectiveness of hybrid images. Low spatial frequencies (blobs) lack a precise definition of object shapes and region boundaries, which require the visual system to group the blobs together to form a meaningful interpretation of the coarse scale. When observers are presented with ambiguous forms they interpret the elements in the simplest way. Observers prefer an arrangement having fewer rather than more elements, having a symmetrical rather than an asymmetrical composition and generally respecting other Gestalt rules of perception.

Symmetry and repetitiveness of a pattern in the low spatial frequencies are bad: they form a strong percept that it is difficult to eliminate perceptually. If the image in the high spatial frequencies lacks the same strong grouping cues, the image interpretation corresponding to the low spatial frequencies will always be available, even when viewing from a short distance. By introducing accidental alignments it is possible to reduce the influence of one spatial channel over the other. For instance, in Fig. 2 the top of the elephant (low spatial frequencies) is aligned with the horizon line (both low and high spatial frequencies). Therefore, when seeing the image up close, the top edge of the elephant can be explained by some of the fine edges. This reduces the saliency of the elephant. Fig. 3 shows several examples of hybrid images with different degrees of agreement between the low and high spatial frequencies.

Color provides a very strong grouping cue that can be used to create more compelling illusions. For instance, in Fig. 4 color is used only in the high spatial frequencies to enhance the bicycle and to reinforce the interpretation of the motorcycle as shadows when the image is viewed up close.

The importance of correctly choosing the cut-off frequencies for the filters is illustrated in Fig. 5. In Fig. 5.a, both filters have a strong overlap, and consequently, there is not a clean transition between the two faces. For the hybrid image on Fig. 5.b, the two filters have little overlap. The result is a cleaner image that produces an

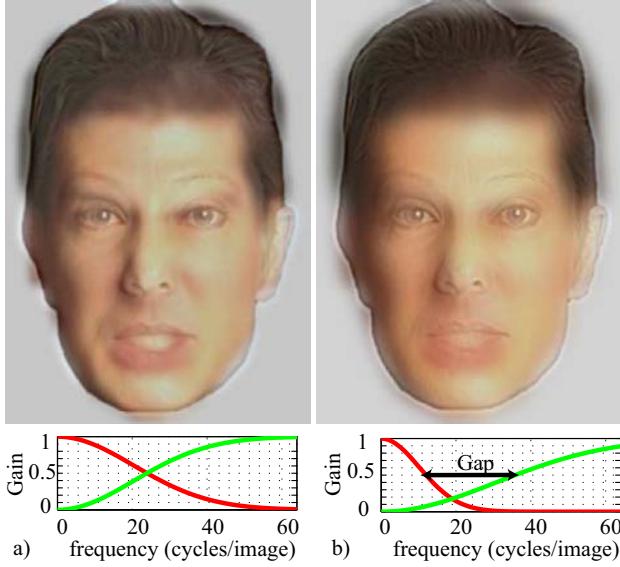


Figure 5: An angry man or a thoughtful woman? Both hybrid images are produced by combining the faces of an angry man (low spatial frequencies) and a stern woman (high spatial frequencies). You can switch the percept by watching the picture from a few meters. a) Bad hybrid image. The image looks ambiguous from up close due to the filter overlap. b) Good Hybrid image.

unambiguous interpretation (it looks like a woman from up close and as a man from far away). This is especially important when the images are not perfectly aligned.

One interesting observation is that when the images are properly constructed, the observer seems to perceive the masked image a noise. Hybrid images break one important statistical property of real-world natural images (Fig. 6), i.e., the correlations between outputs of pass-band filters at consecutive spatial scales. Fig. 6.a shows the cross-correlation matrix obtained between the different levels of a Laplacian pyramid for a natural image. The edges found at one scale are correlated with the edges found in the scales below and above. The same thing is obtained when two images are superimposed (additive transparency). In this case there is not a simple filter to separate both images (and the percept of the two images is mixed independently of the distance at which we observe the image). Fig. 6.c shows the correlation matrix obtained when an image is blurred (with a cutoff frequency of $16c/i$) and then corrupted with additive white noise. The correlation matrix reveals which scales are dominated by the noise, as they do not have the cross-scale correlations we'd expect from a natural image. In the case of a hybrid image, the correlation matrix (Fig. 6.d) reveals the existence of two groups.

Fig. 7 shows the output of a Laplacian pyramid applied to the hybrid image from Fig. 5.b. Low frequency channels and high frequency channels see different images. Note that each subband is also a hybrid image itself. If you move away from the page, you will see that, one by one, the subbands take the identity of the low-scales. At reading distance, the four images on the top row are interpreted as an angry man; the bottom, a stern woman. As you step back from the images, you will see that the angry man's face begins to appear in more subbands. The finer the scale of each subband, the farther you have to go in order to see the switch of images.

In summary, two primary mechanisms can be exploited to create compelling hybrid images. The first is maximizing the correlation

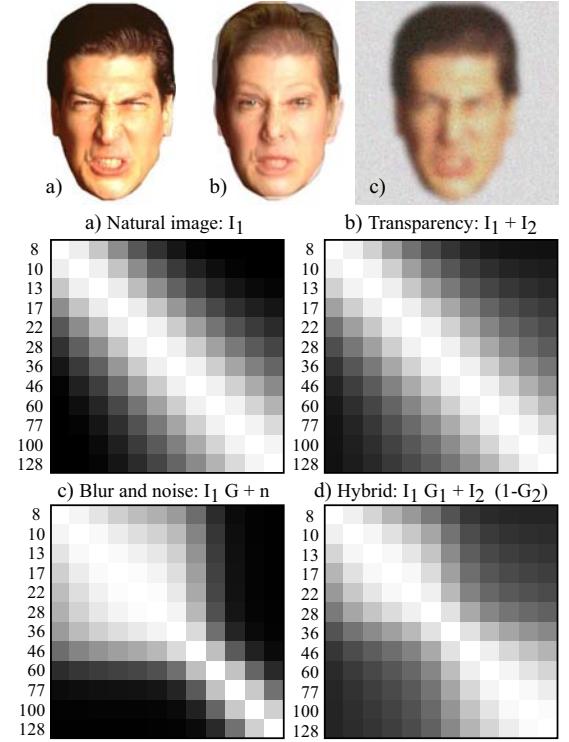


Figure 6: Correlations across levels of a Laplacian pyramid for images following several manipulations. a) Natural image, b) two images added, c) blurry image with additive white noise, and d) hybrid image ($f_1 = 16 \text{ cycles}/\text{image}$, $f_2 = 48 \text{ cycles}/\text{image}$).

between edges in the two scales so that they blend. The second resides in the fact that the remaining edges that do not correlate with other edges across scales can be perceived as noise. This is the case in Fig. 5.b, for which there is a very compelling blending of edges across scales, but, when viewing the image up close, there seems to be some low-spatial frequency noise.

2.3 Capacity of scale space

Up to now, hybrid images have been obtained by mixing two images, but could it be possible to combine more than two images and still have a coherent percept that transitions as we change viewing distance? In a study about text masking, Majaj et al. [2002] created a stimulus superimposing 4 letters, each containing energy at different spatial scales. As the observer moves away from the stimuli, they report the image switching from one letter to another. The results are interesting, but the lack of good grouping cues between the multiple scales creates an image that looks distorted. Also, multiple letters are visible at any given time. Superposition of multiple images remains an open issue.

3 Applications

In this section we discuss some applications (see video complementing the paper for additional examples).

Private font: We can use the hybrid images to display text that is not visible for people standing at some distance from the

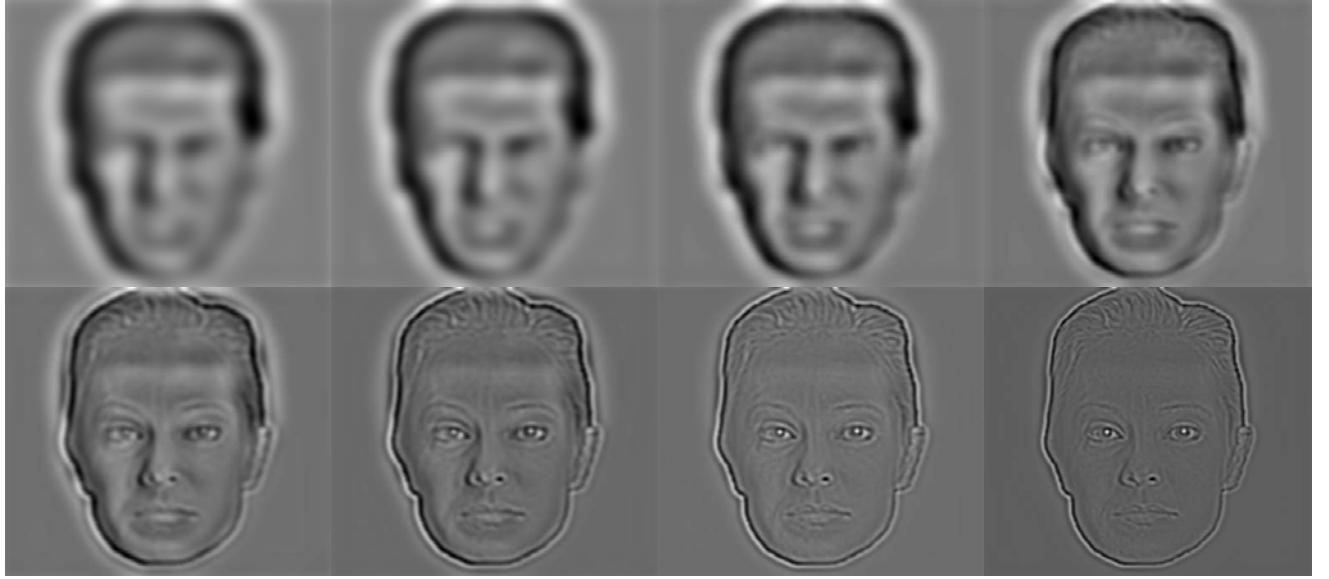


Figure 7: Output of a Laplacian pyramid revealing the components of the hybrid image of Fig. 5.b.

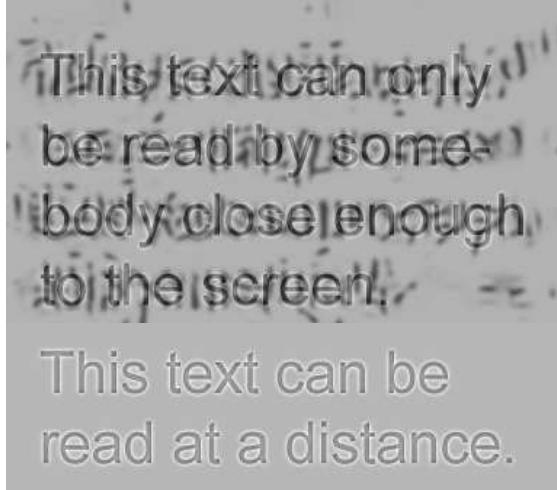


Figure 8: The hybrid font becomes invisible at few meters. The bottom text remains easy to read at relatively long distances.

screen. Commercial products for user privacy generally rely on head mounted displays or on polarized screens for which visibility decreases with viewing angle. Hybrid fonts comprises two components: the high spatial frequencies (which will contain the text) and the low spatial frequencies (which will contain the masking image).

For the high pass filter we use a gaussian filter with a width (σ) adjusted so that $\sigma < n_p$, where n_p is the thickness of a letter's stroke measured in pixels. The low-frequency channel (masking signal) contains a text-like texture [Portilla and Simoncelli 2000]. Solomon and Pelli [1994] have shown that letters are more effectively masked by a noise in the frequency band of 3 cycles per letter. Therefore we adjust the cut-off frequency of the low-pass filter to be $3 * n$ with n being the number of letters in a text line. The goal is to reduce the interference of the noise with the text when we viewing up close, while having an effective masking noise when looking from further away. In the example shown in Fig. 8 the text is only

readable from a distance below one meter. From a distance of about two meters, the text is unreadable. Masking of the low spatial frequencies is very important in producing this effect (Fig. 8). The text in the bottom has only been high-pass filtered, and there is no masking at low spatial frequencies, therefore it remains easy to read at relatively long distances.

Hybrid textures: We can create textures that disappear with viewing distance. An example of this idea is shown in Fig. 9. This figure shows an example of a woman's face that turns into a cat when looking close. Note that this effect can not be obtained by superimposing the woman's face and the cat's face using transparency. Using transparency (additive superposition) creates a face that will not change with distance.

Changing faces: Hybrid images are especially powerful to create images of faces that change expressions, identity, or pose as we vary the viewing distance. Fig. 1 shows a compelling example of changes of facial expression. The edges at multiple scales blend producing images that look natural at all distances. In the case of face images, correct alignment between facial features is important in order to create pictures that seem unaltered. In case of misalignment, the best is to apply a distortion (affine warping) to the face that will be in the low spatial frequencies.

Time changes: Fig. 9 shows an example of using an hybrid image to show two states of a house by combining two picture taken at two different instants.

4 Conclusion

We have described the technique, hybrid images, which permits creating images with two interpretations that change as a function of viewing distance. Despite the simplicity of the technique, the images produce very compelling surprise effects on naive observers. They also provide an interesting new visualization tool to morph two complementary images into one. Creating compelling hybrid images is an open and challenging problem, as it relies on perceptual grouping mechanisms that interact across different spatial scales.



Figure 9: right) Cat woman: the texture corresponding to the cat's face disappears when the image is viewed from a few meters. Left) The house under construction. When you view the image at a short distance, the house is seen under construction, but if you step away from the picture you will see its final state.

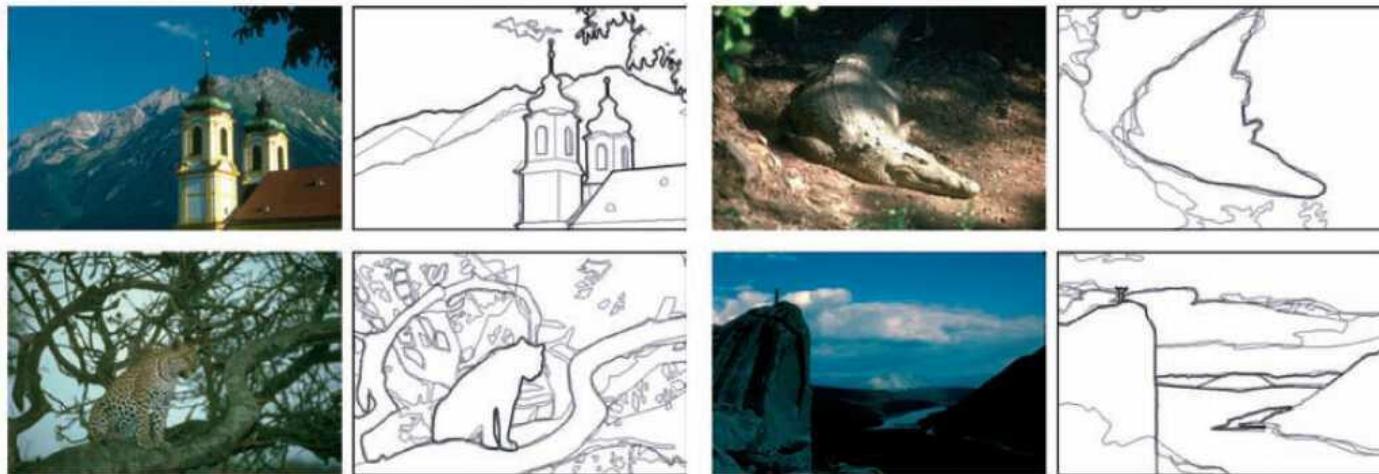
References

- BURT, D. C., AND ADELSON, E. H. 1983. The laplacian pyramid as a compact image code. *IEEE Transaction on Communications* 31, 532–540.
- DALI, S. 1996. *The Salvador Dali Museum Collection*. Bulfinch Press.
- HUGHES, H. C., NOZAWA, G., AND KITTERLE, F. 1996. Global precedence, spatial frequency channels, and the statistics of natural images. *Journal of Cognitive Neuroscience* 8, 197–230.
- LINDEBERG, T. 1993. Detecting salient blob-like images structures and their spatial scales with a scale-space primal sketch: a method for focus of attention. *Int. J. Comp. Vis* 11, 283–318.
- LIVINGSTONE, M. S. 2000. Is it warm? is it real? or just low spatial frequency? *Science* 290, 5495, 1299.
- MAJAJ, N., PELLI, D., KURSHAN, P., AND PALOMARES, M. 2002. The role of spatial frequency channels in letter identification. *Vision Research* 42, 1165–1184.
- NAVON, D. 1977. Forest before trees: the precedence of global features in visual perception. *Cognitive psychology* 9, 353–383.
- OLIVA, A., AND SCHYNS, P. 1997. Coarse blobs or fine edges? evidence that information diagnosticity changes the perception of complex visual stimuli. *Cognitive psychology* 34, 1, 72–107.
- PARKER, D., LISHMAN, J., AND HUGHES, J. 1992. Temporal integration of spatially filtered visual images. *Perception* 21, 147–160.
- PARKER, D., LISHMAN, J., AND HUGHES, J. 1996. Role of coarse and fine information in face and object processing. *J. Exp. Psychol. Hum. Percept. Perform.* 22, 1448–1466.
- PORRILLA, J., AND SIMONCELLI, E. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comp. Vis* 40, 49–71.
- POTTER, M. 1975. Meaning in visual scenes. *Science* 187, 965–966.
- SCHYNS, P., AND OLIVA, A. 1994. From blobs to boundary edges: Evidence for time- and spatial-scale-dependent scene recognition. *Psychological Science* 5, 195–200.
- SCHYNS, P., AND OLIVA, A. 1999. Dr. angry and mr. smile: when categorization flexibly modifies the perception of faces in rapid visual presentations. *Cognition* 69, 243–265.
- SETLUR, V., AND GOOCH, B. 2004. Is that a smile?: gaze dependent facial expressions. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering*, ACM Press, New York, NY, USA, 79–151.
- SILVERS, R. 1997. *Photomosaics*. Henry Holt and Company, Inc.
- SOLOMON, J., AND PELLI, A. 1994. The visual filter mediating letter identification recognition. *Nature* 369, 395–397.
- SUGASE, Y., YAMANE, S., UENO, S., AND KAWANO, K. 1999. Global and fine information coded by single neurons in the temporal visual cortex. *Nature* 400, 869–873.
- WATT, R. 1987. Scanning from coarse to fine spatial scales in the human visual system after onset of a stimulus. *J. Opt.Soc.Am: A*, 4, 2006–2021.

Feature Extraction – Edges

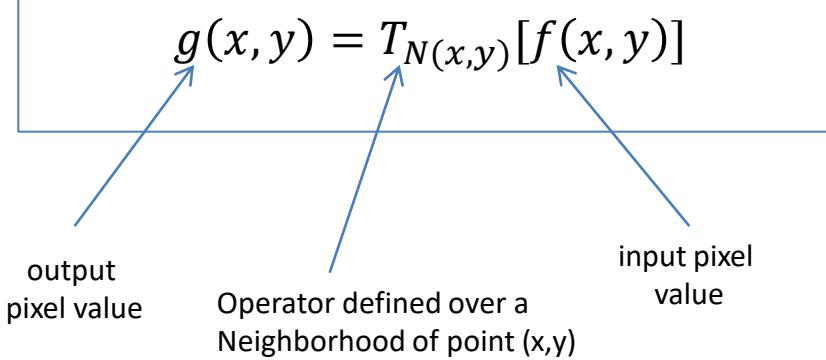
EDGES AND CONTOURS

SZELISKI BOOK CHAPTER 7.2



Filtering Basics

Spatial domain process:



Why filtering is important?

Filtering is needed to

- Enhance images (smoothing, noise removal etc.)
- Extract information (edges, texture etc.)
- Detect patterns

Filtering types:

- Spatial domain
- Frequency domain

Filtering types:

- Linear
- Non-linear

Gonzalez & Woods Table 10.1

First four central digital derivatives (finite differences) for samples taken uniformly, $\Delta x = 1$ units apart.

	$f(x+2)$	$f(x+1)$	$f(x)$	$f(x-1)$	$f(x-2)$
$2f'(x)$		1	0	-1	
$f''(x)$		1	-2	1	
$2f'''(x)$	1	-2	0	2	-1
$f''''(x)$	1	-4	6	-4	1

Gonzalez & Woods Figure 10.2

- (a) Image.
 - (b) Horizontal intensity profile that includes the isolated point indicated by the arrow.
 - (c) Subsampled profile; the dashes were added for clarity. The numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10-4) for the first derivative and Eq. (10-7) for the second.

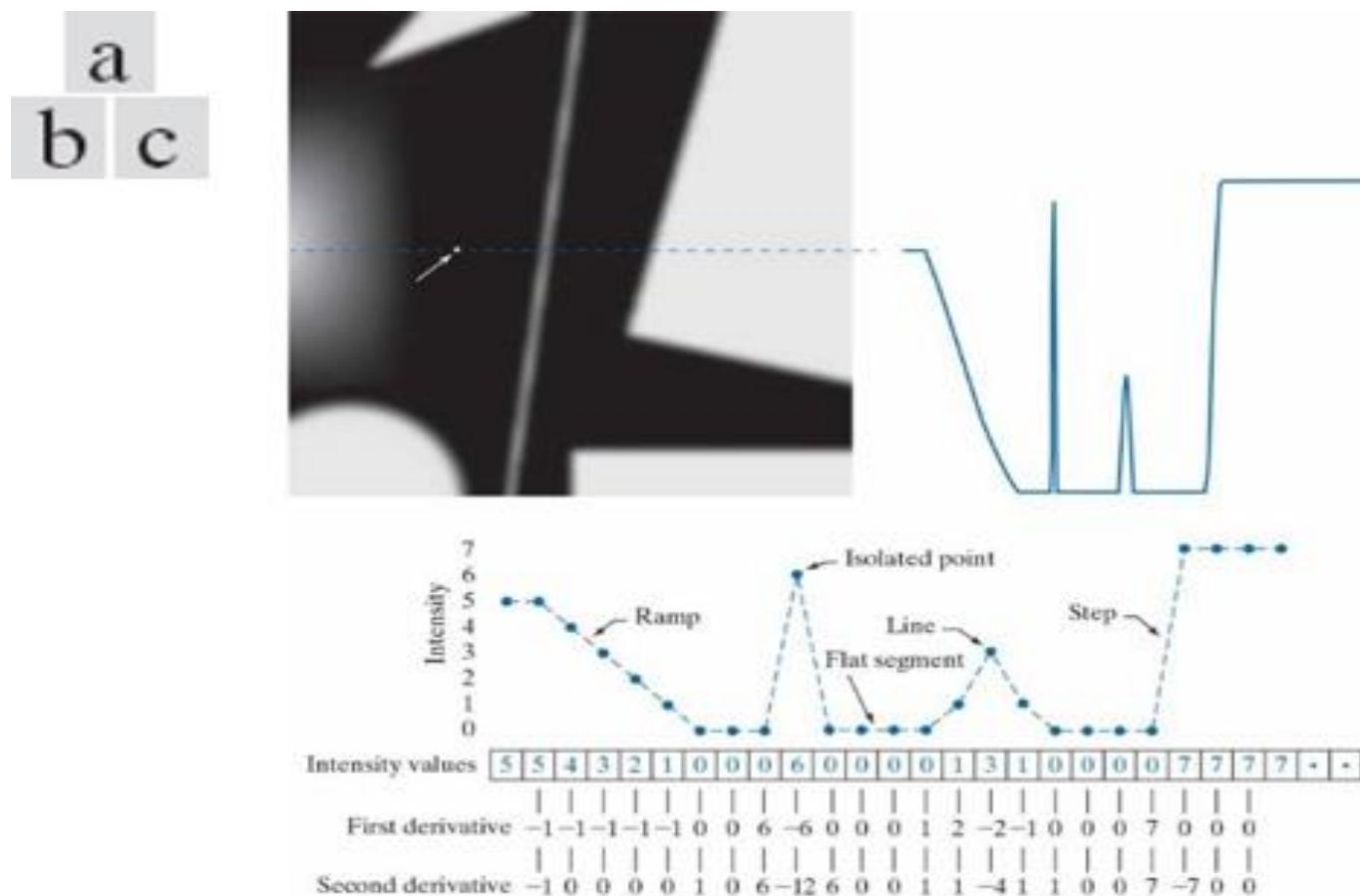
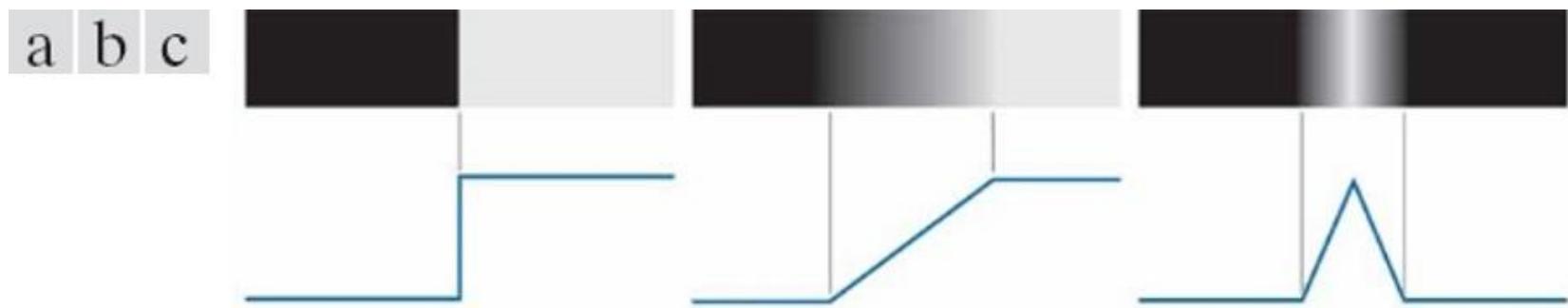


Figure 10.8

From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.



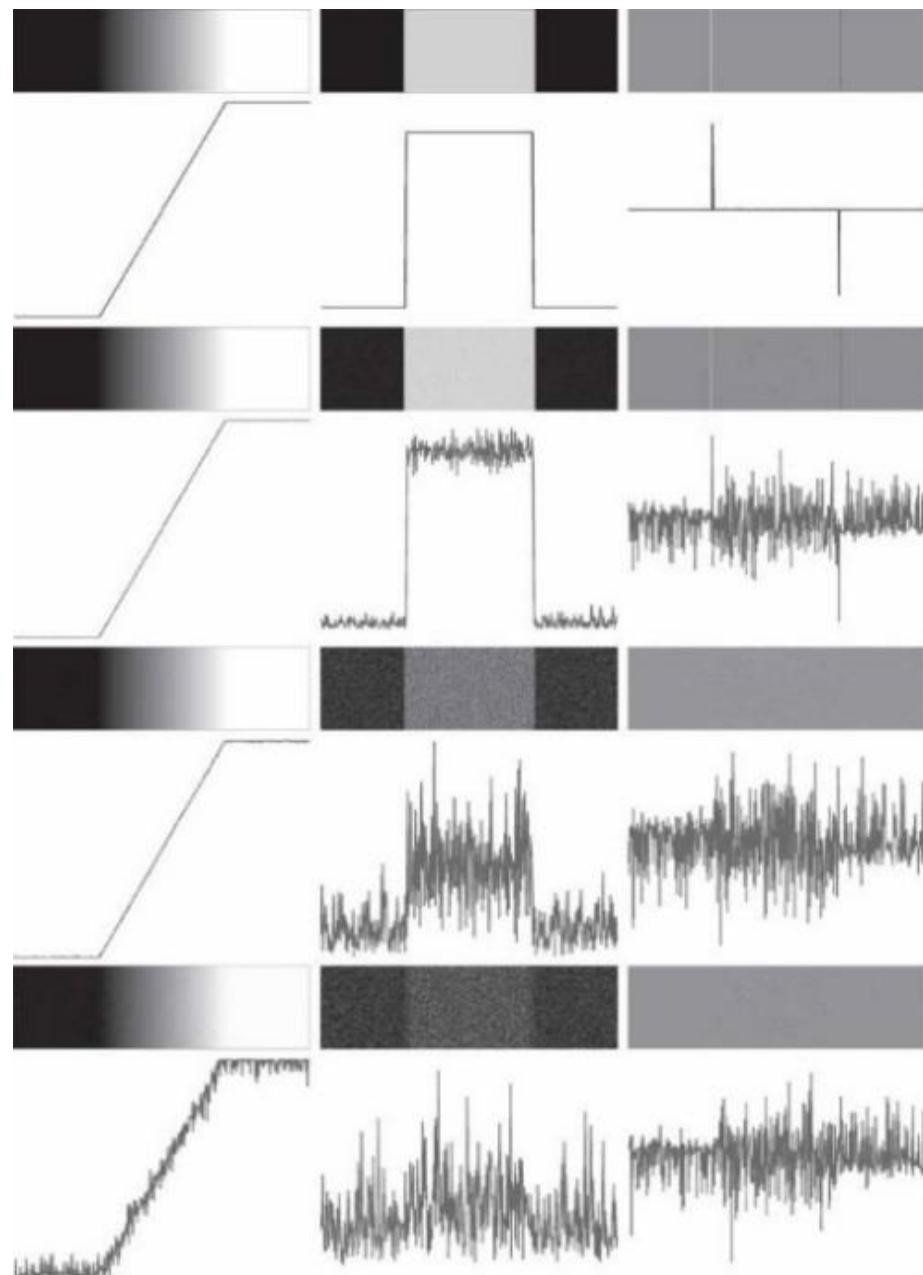
Gonzalez & Woods

Figure 10.11

First column: 8-bit images with values in the range [0,255], and intensity profiles of a ramp edge corrupted by Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively.

Second column: First-derivative images and intensity profiles.

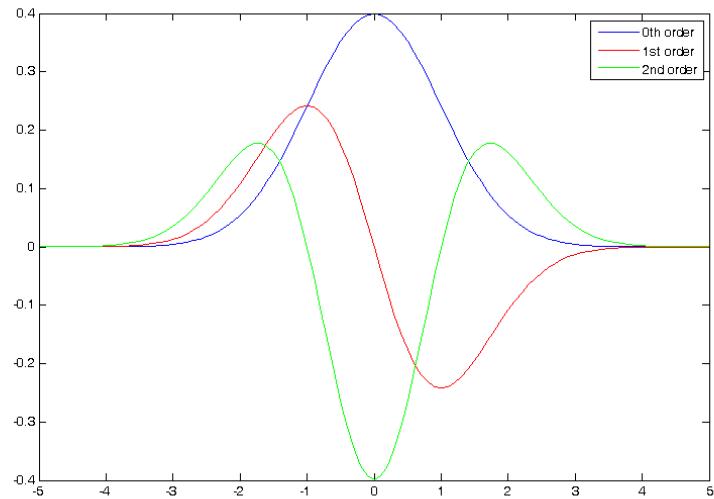
Third column: Second-derivative images and intensity profiles.



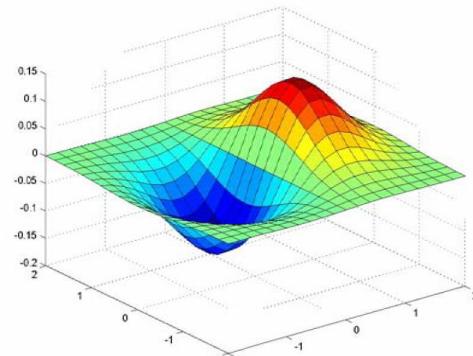
The 1D Gaussian and its derivatives

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

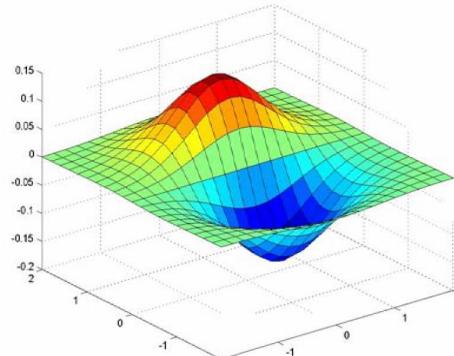
$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma} \left(\frac{x}{\sigma} \right) G_\sigma(x)$$



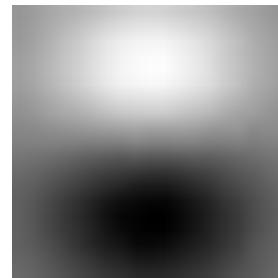
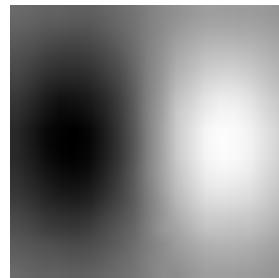
Derivative of Gaussian filter



x-direction



y-direction



The Sobel operator

- Common approximation of derivative of Gaussian

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

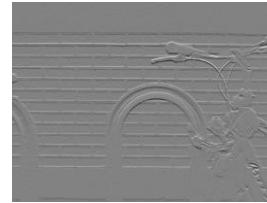
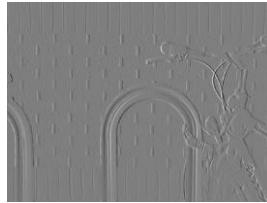
s_x

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

s_y

- The standard definition of the Sobel operator omits the $1/8$ term
 - doesn't make a difference for edge detection
 - the $1/8$ term **is** needed to get the right gradient magnitude

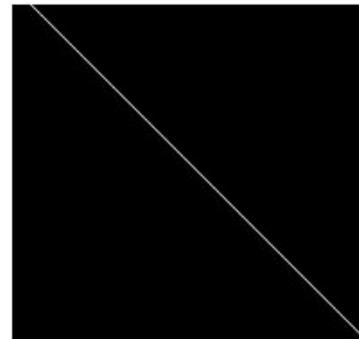
Sobel operator: example



Source: Wikipedia



Image with Edge



Edge Location

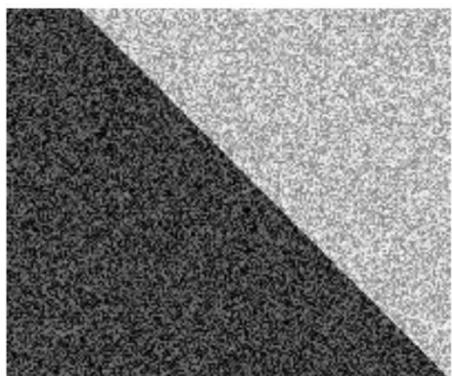
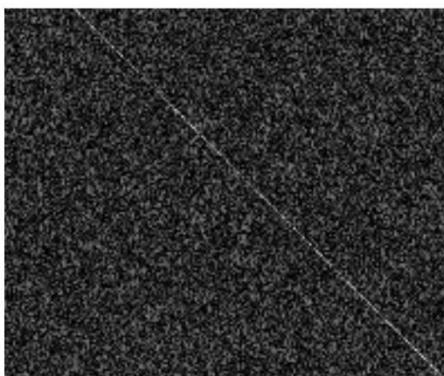
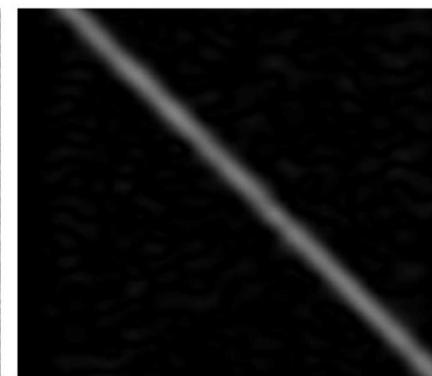


Image + Noise

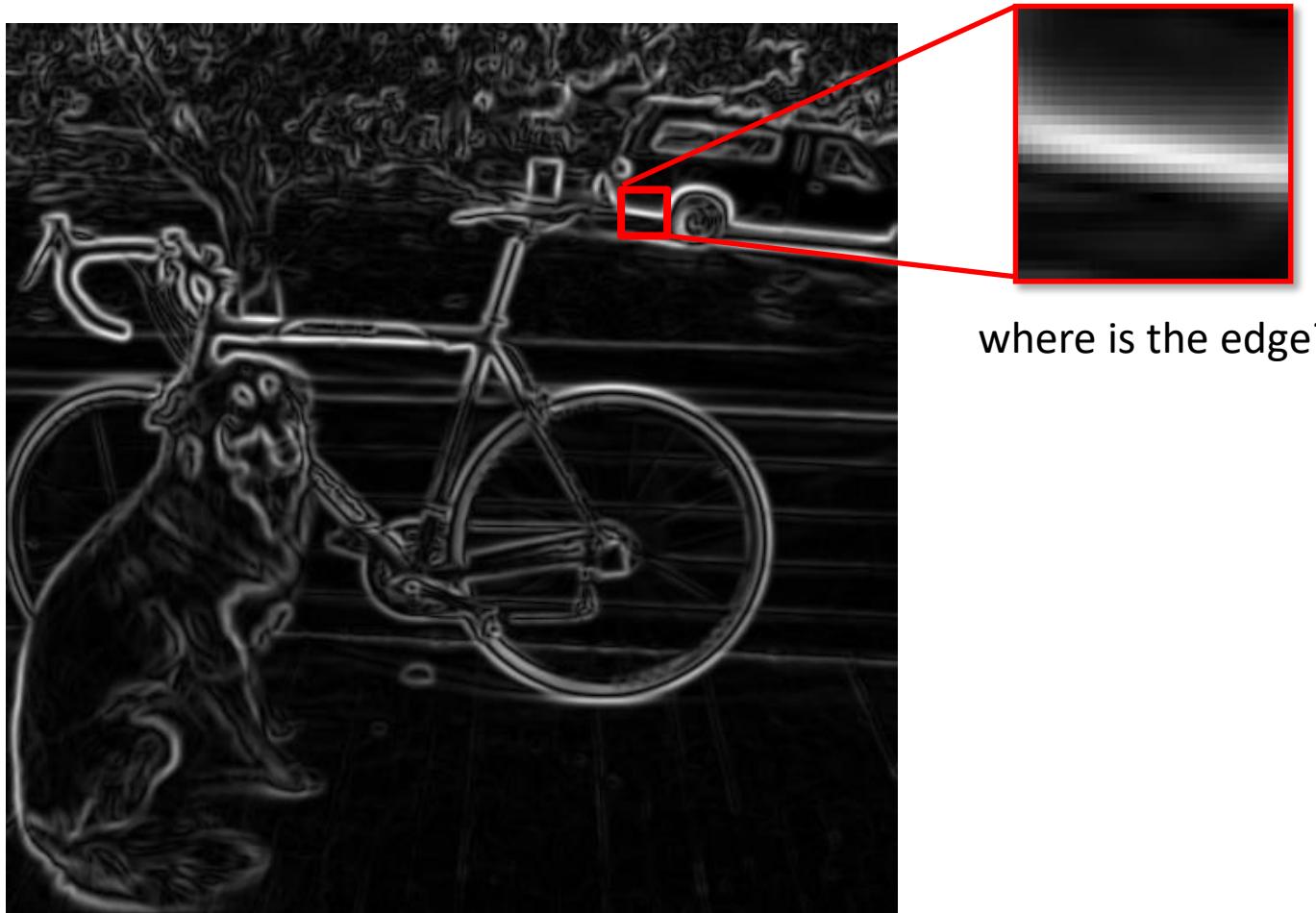


Derivatives detect
edge *and* noise



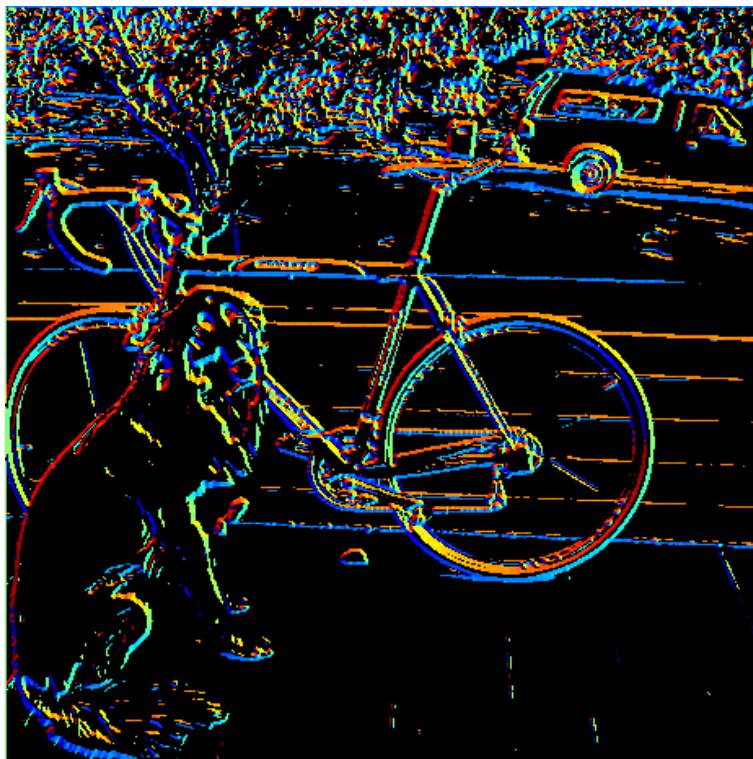
Smoothed derivative removes
noise, but blurs edge

Finding Edges --- Beyond Gradient Computation



Get Orientation at Each Pixel

- Get orientation (below, threshold at minimum gradient magnitude)



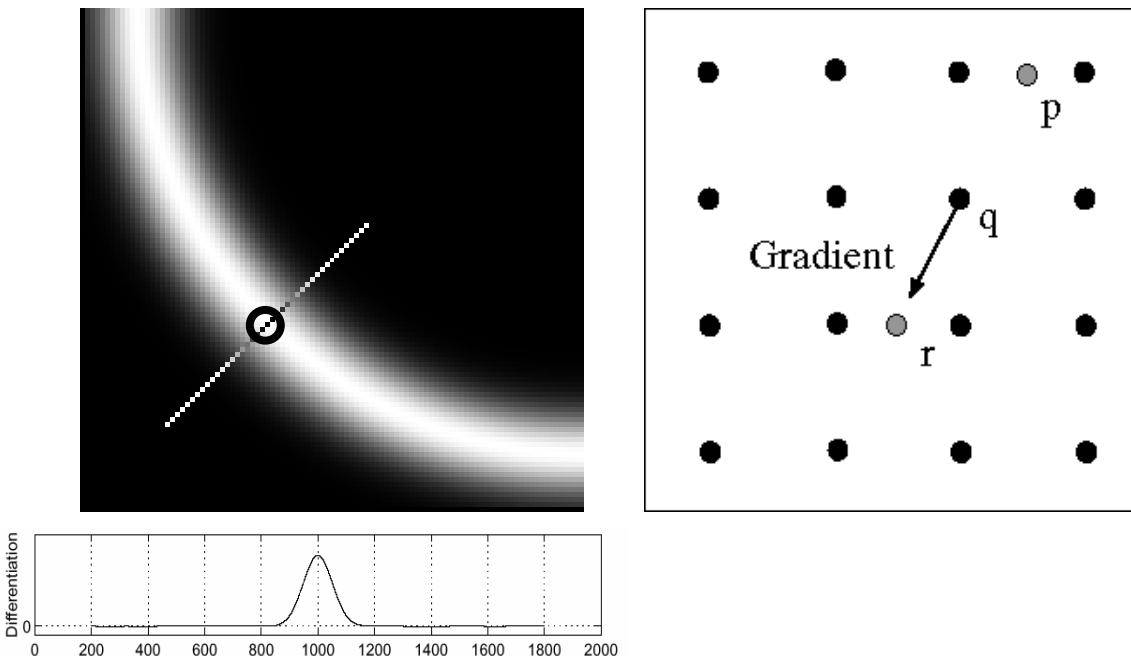
$\theta = \text{atan2}(gy, gx)$

360

Gradient orientation angle

0

Non-maximum suppression



- Check if pixel is local maximum along gradient direction
 - requires *interpolating* pixels p and r

Before Non-max Suppression



After Non-max Suppression



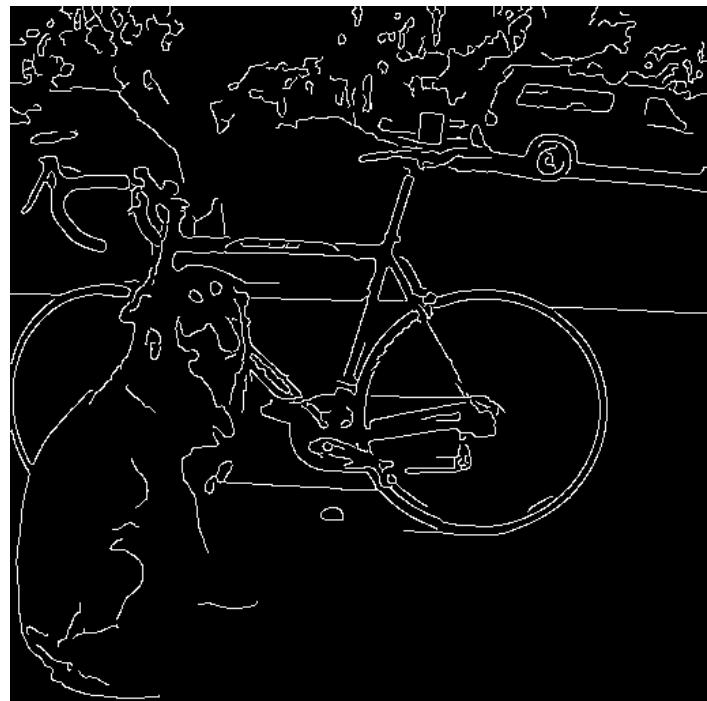
Thresholding edges

- Still some noise
- Only want strong edges
- 2 thresholds, 3 cases
 - $R > T$: strong edge
 - $R < T$ but $R > t$: weak edge
 - $R < t$: no edge
- Why two thresholds?



Connecting edges

- Strong edges are edges!
- Weak edges are edges iff they connect to strong
- Look in some neighborhood (usually 8 closest)



Canny edge detector



MATLAB: `edge(image, 'canny')`

OpenCV: `cv.Canny()`

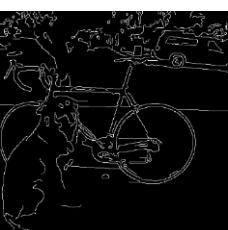


1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding (hysteresis):
 - Define two thresholds: low and high
 - Use the high threshold to start edge curves and the low threshold to continue them



Source: D. Lowe, L. Fei-Fei, J. Redmon

Canny edge detector

- Our first computer vision pipeline!
- Still a widely used edge detector in computer vision

J. Canny, [*A Computational Approach To Edge Detection*](#), IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

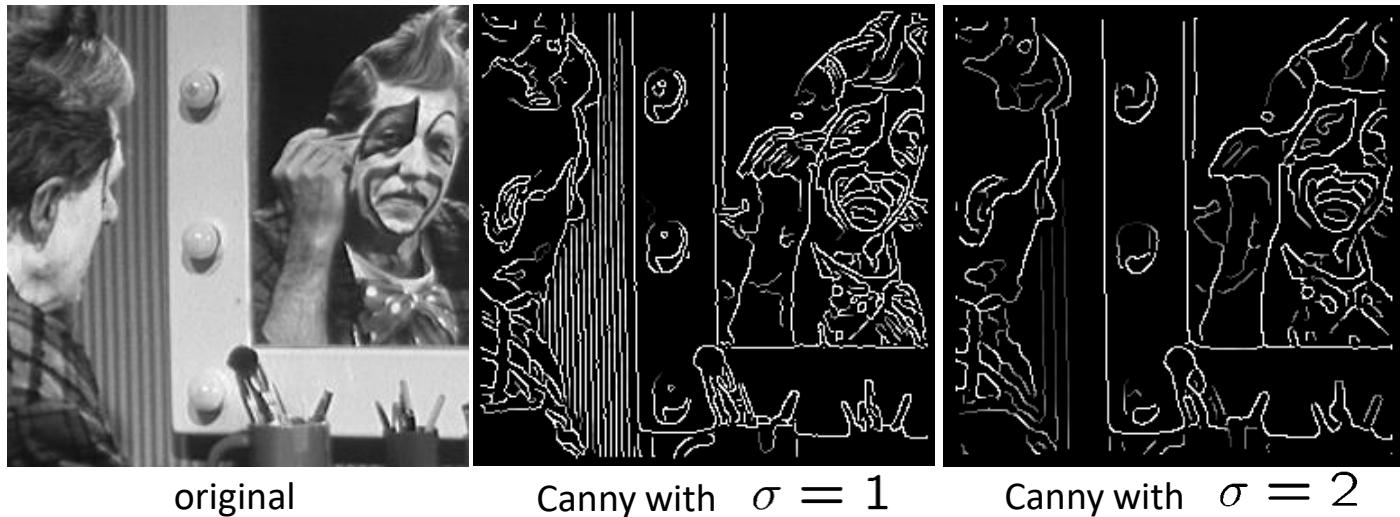
- Depends on several parameters:

high threshold

low threshold

σ : width of the Gaussian blur

Canny edge detector

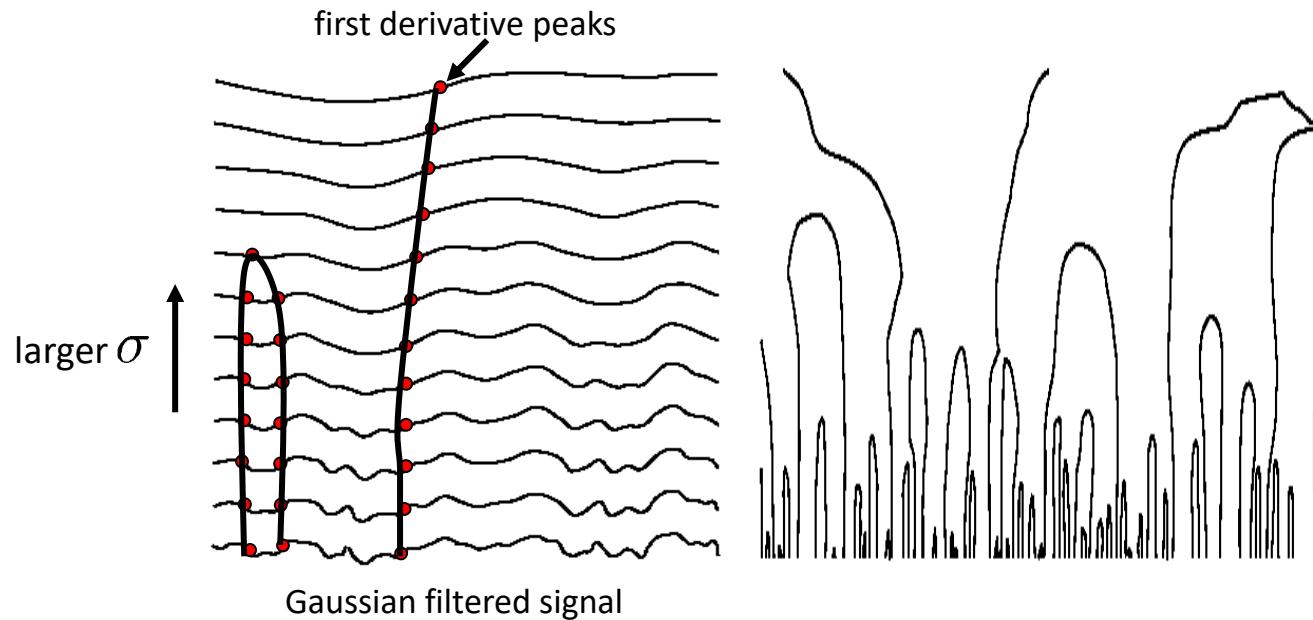


- The choice of σ depends on desired behavior
 - large σ detects “large-scale” edges
 - small σ detects fine edges

Source: S. Seitz

Scale space

[Witkin 83]



- Properties of scale space (w/ Gaussian smoothing)
 - edge position may shift with increasing scale (σ)
 - two edges may merge with increasing scale
 - an edge may **not** split into two with increasing scale

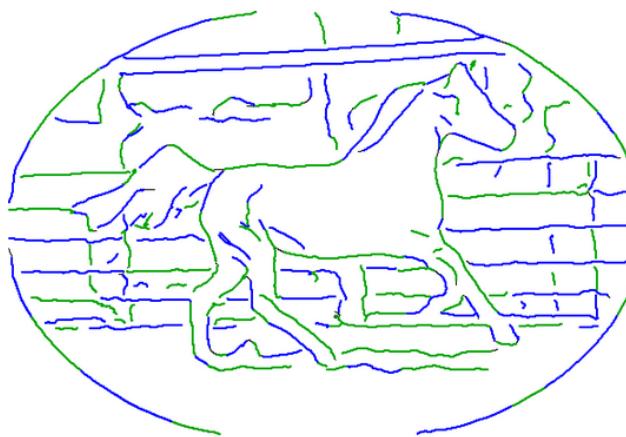
Feature Extraction – Blobs

Feature Extraction

Regions/Segmentation

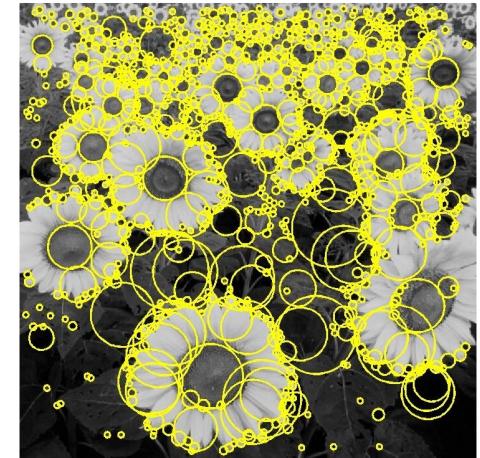


Contours/Line segments

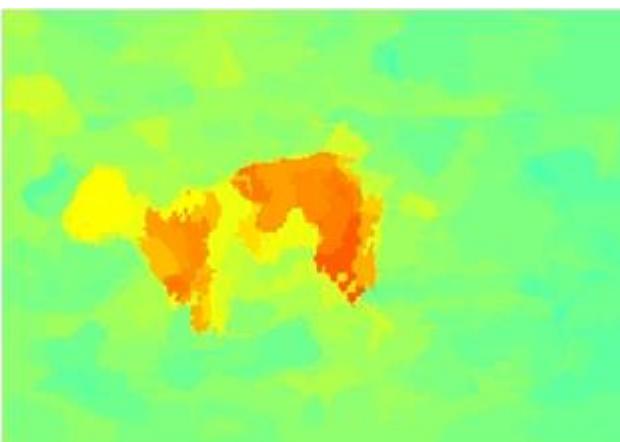


Interest Points

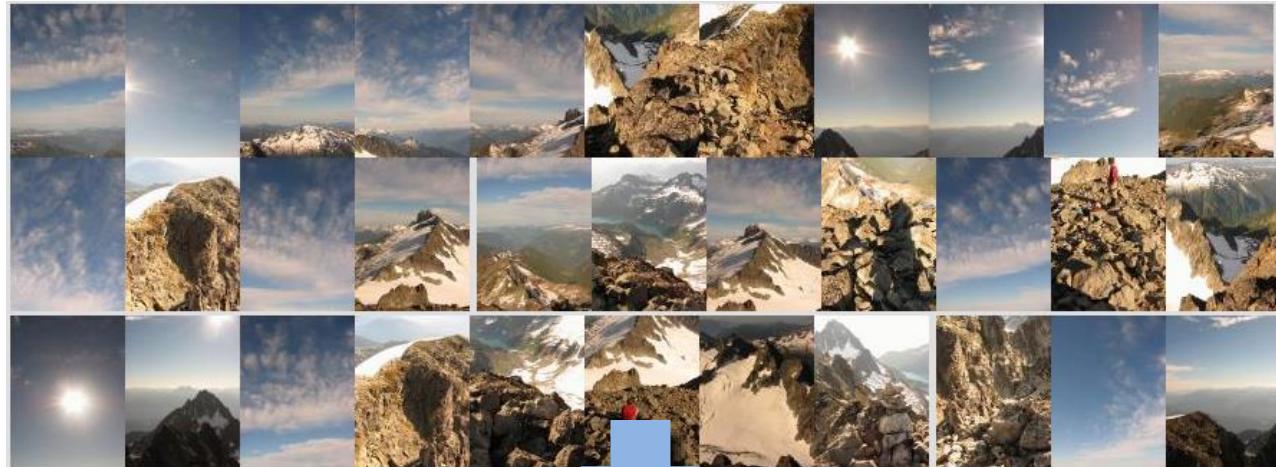
Blobs



Corners



Motivation: Automatic panoramas



Credit: Matt Brown

Motivation: Automatic panoramas



GigaPan:

<http://gigapan.com/>

Also see Google Zoom Views:

<https://www.google.com/culturalinstitute/beta/project/gigapixels>

Image matching



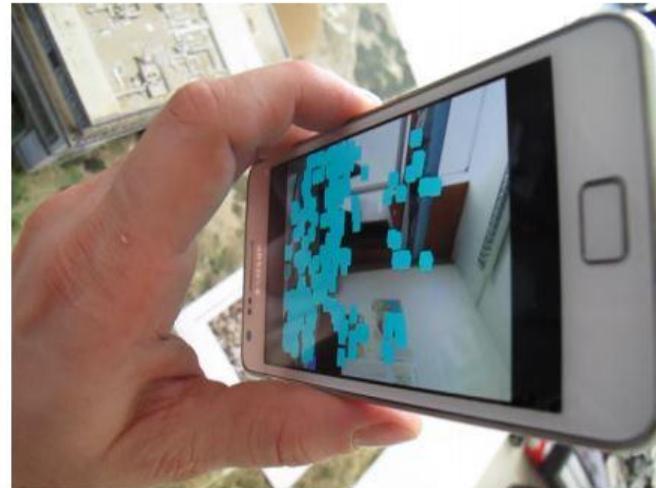
by [Diva Sian](#)



by [swashford](#)

Application: Visual SLAM

- (aka Simultaneous Localization and Mapping)



Example: structure from motion



More motivation...

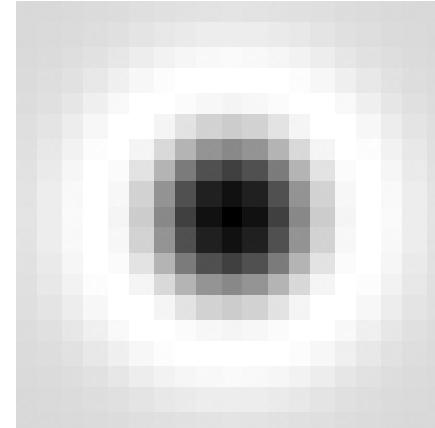
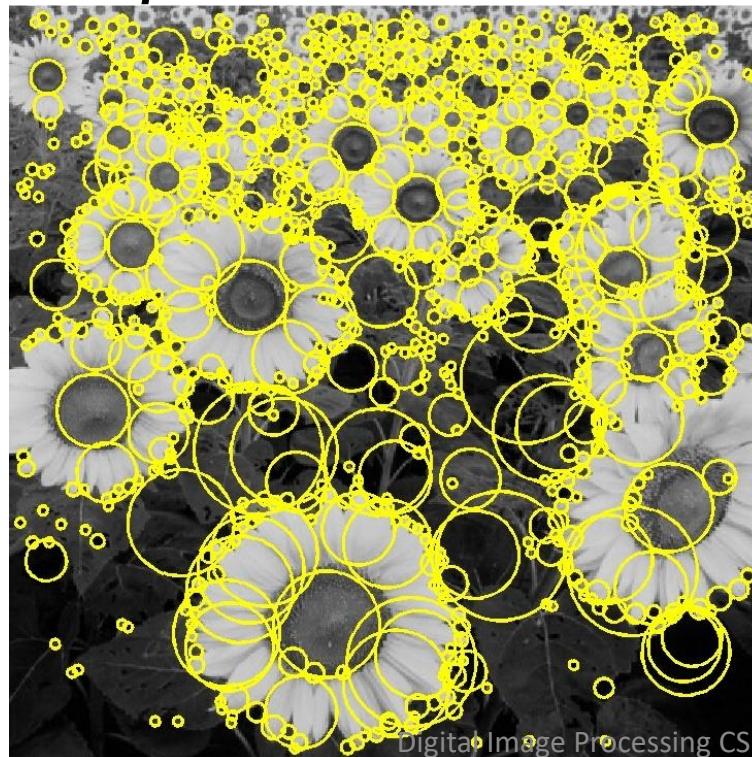
Feature points are used for:

- Image alignment (e.g., mosaics)
- 3D reconstruction
- Motion tracking (e.g. for AR)
- Object recognition
- Image retrieval
- Robot/car navigation
- ... other



Blob detection: Basic idea

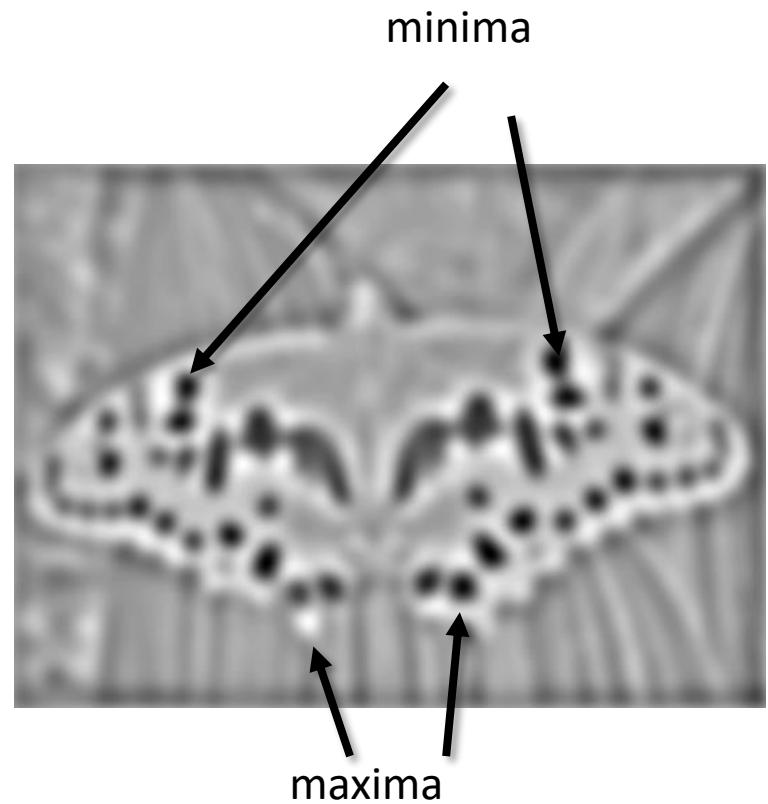
- To detect blobs, convolve the image with a “blob filter” at **multiple scales** and look for **extrema of filter response** in the resulting *scale space*



Blob detection: Basic idea



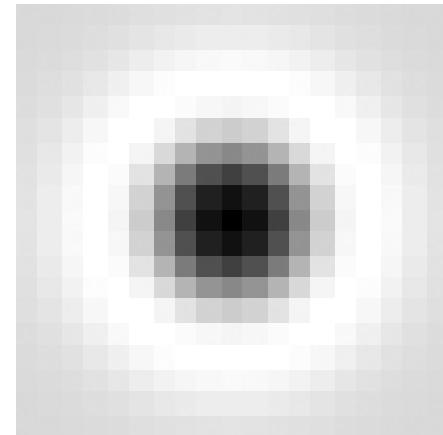
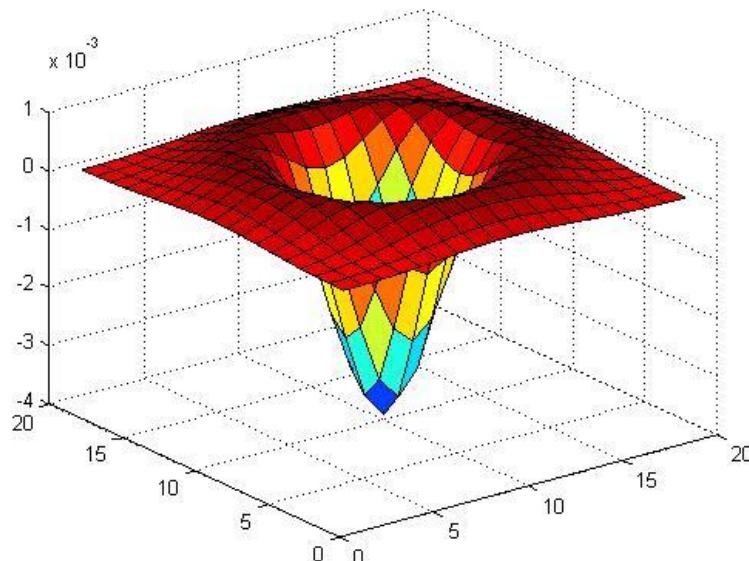
$$* \quad \bullet =$$



- Find maxima *and minima* of blob filter response in space *and scale*

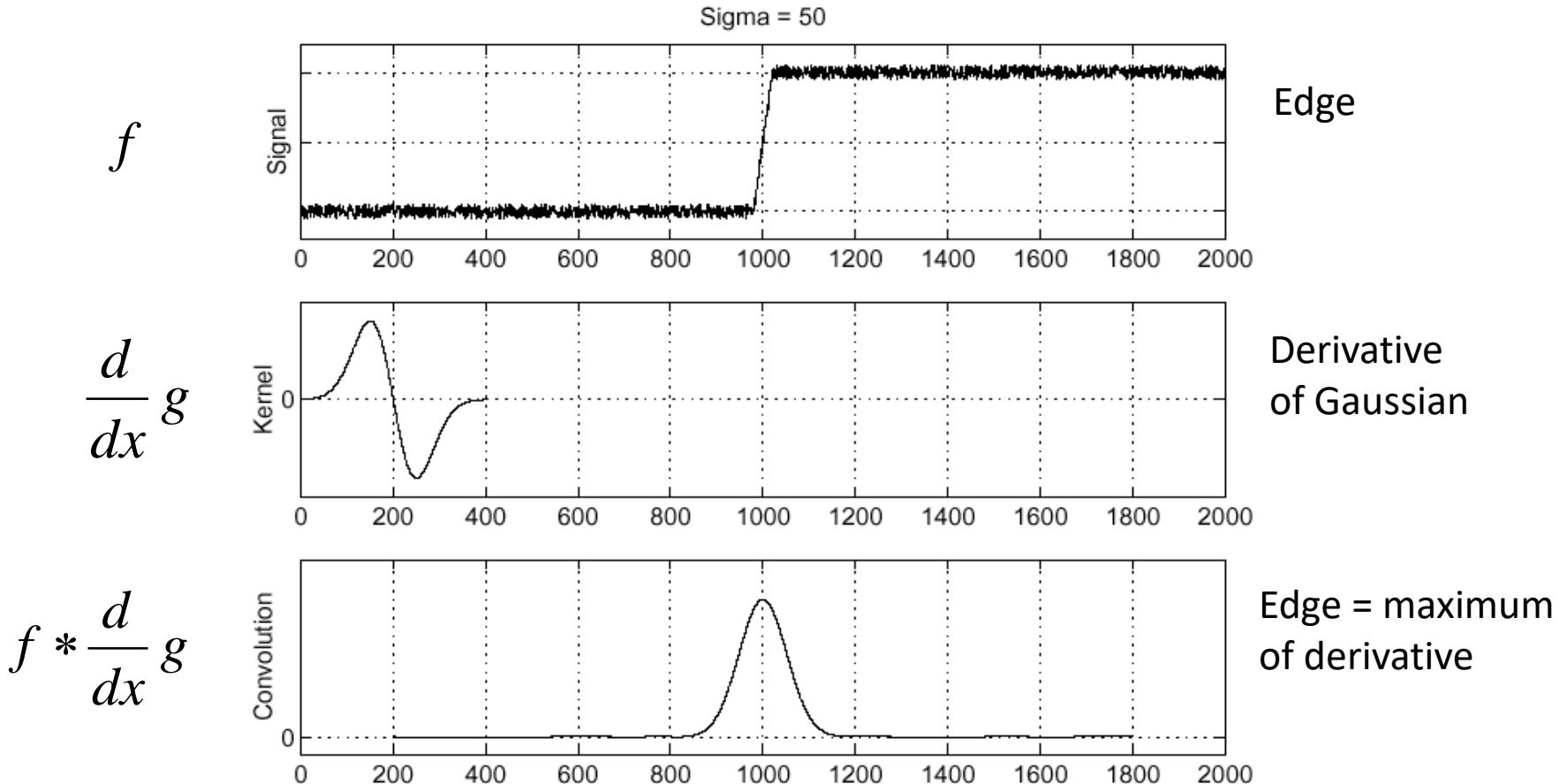
Blob filter

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

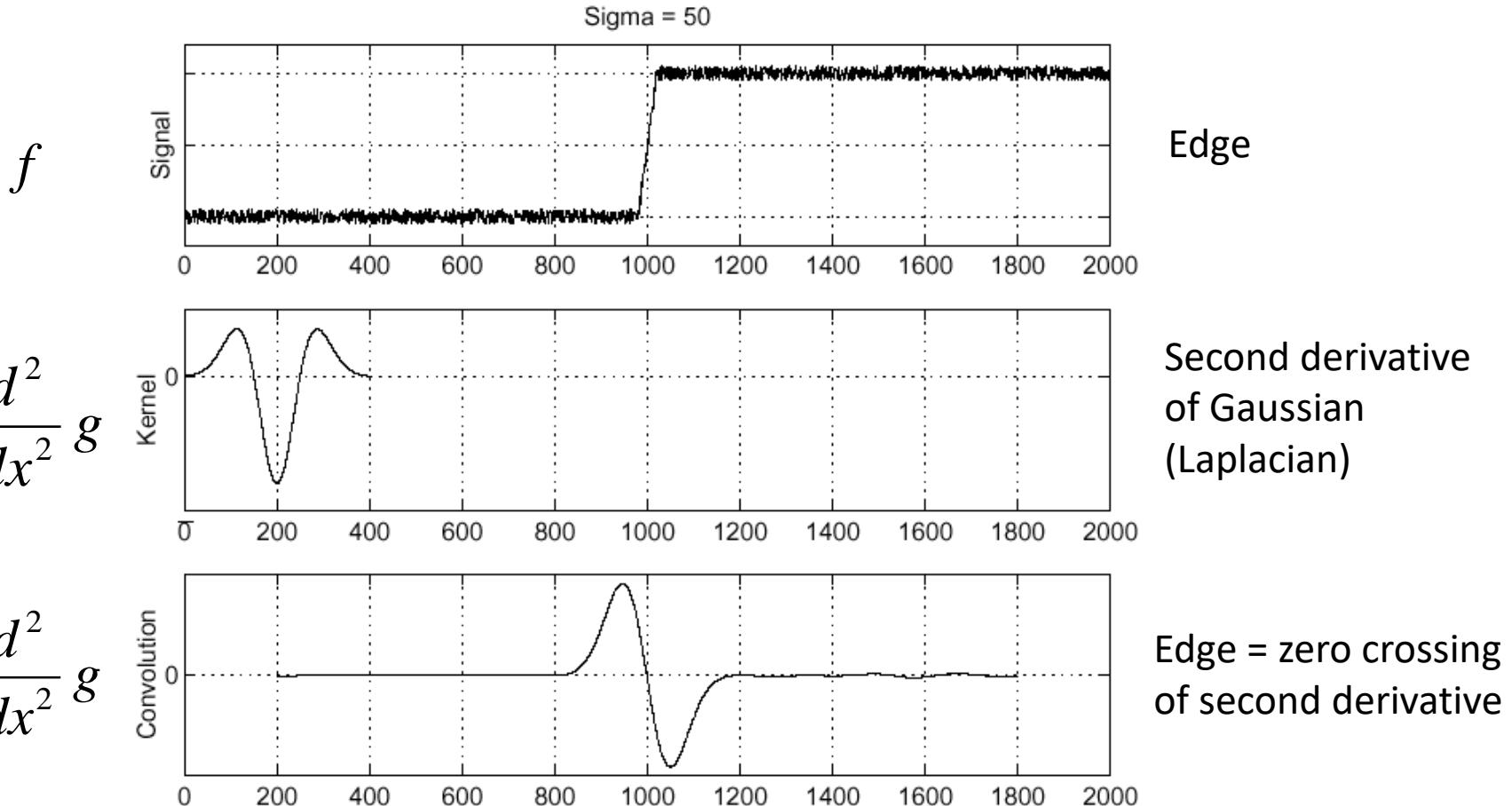


$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

Recall: Edge detection

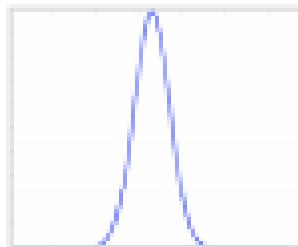


Edge detection, Take 2

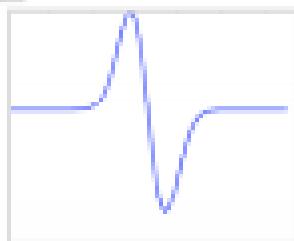


1D Gaussian and Derivatives

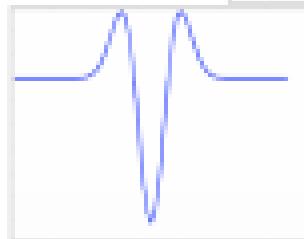
$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



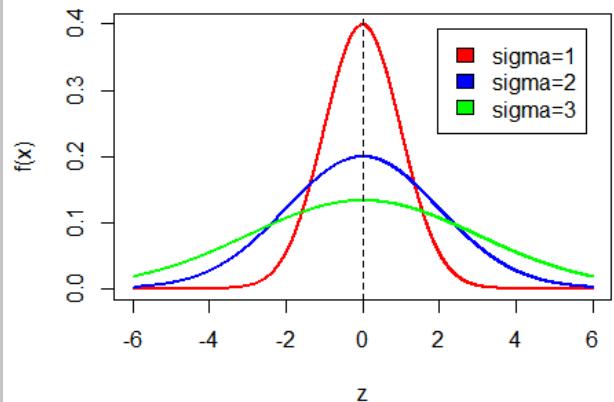
$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$



$$g''(x) = \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right) e^{-\frac{x^2}{2\sigma^2}}$$

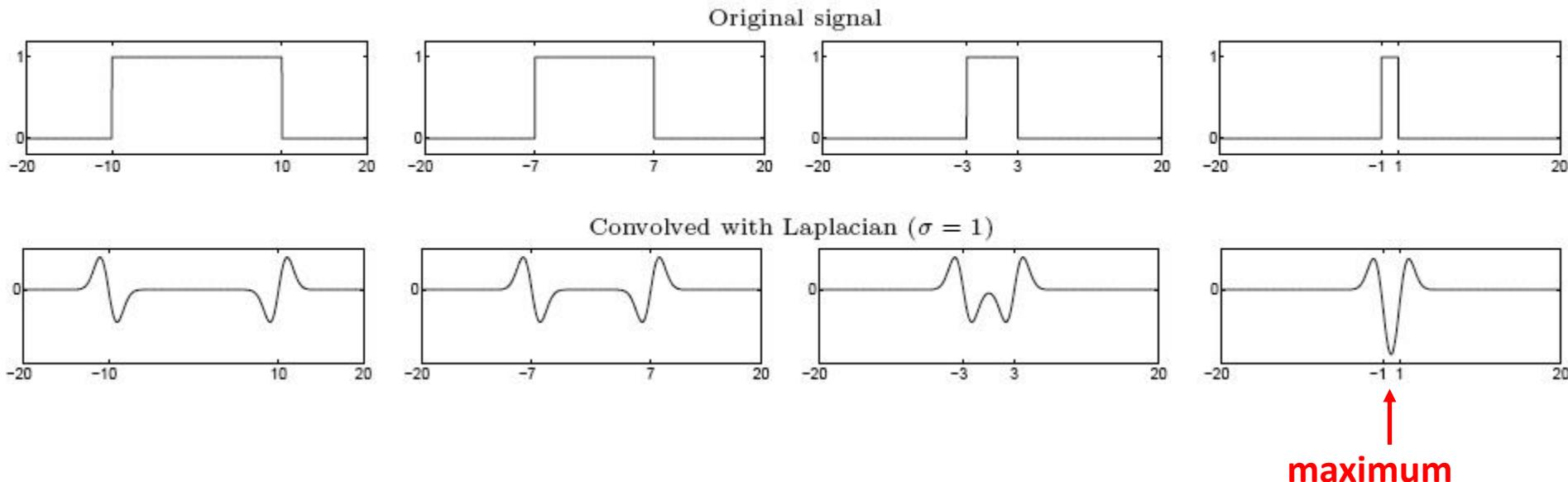
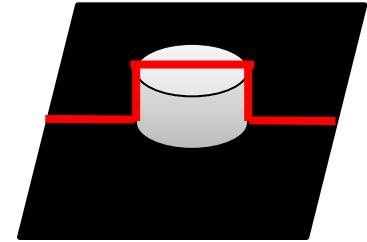


Normal density function with different variance

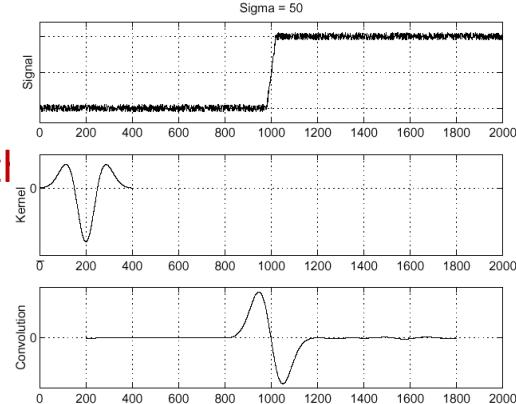


From edges to blobs

- Edge = ripple
- Blob = superposition of two ripples

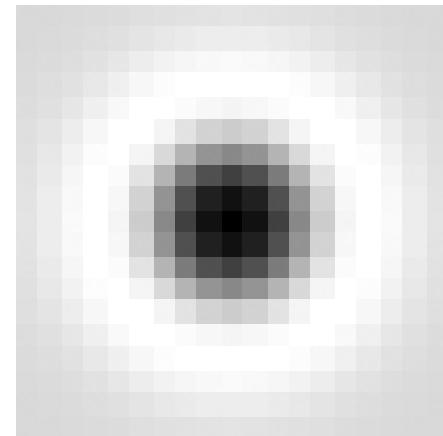
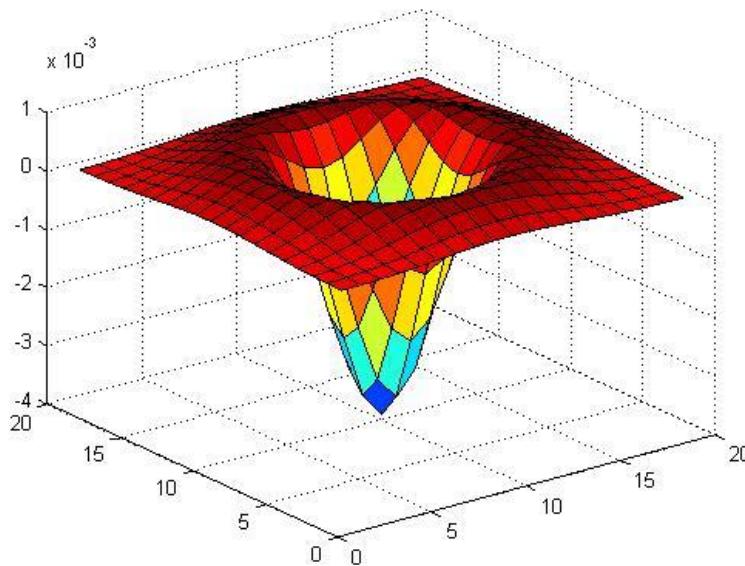


Spatial selection: the magnitude of the Laplacian response will achieve a maximum at the center of the blob, provided the scale of the Laplacian is “matched” to the scale of the edge.



Blob detection in 2D

- Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D

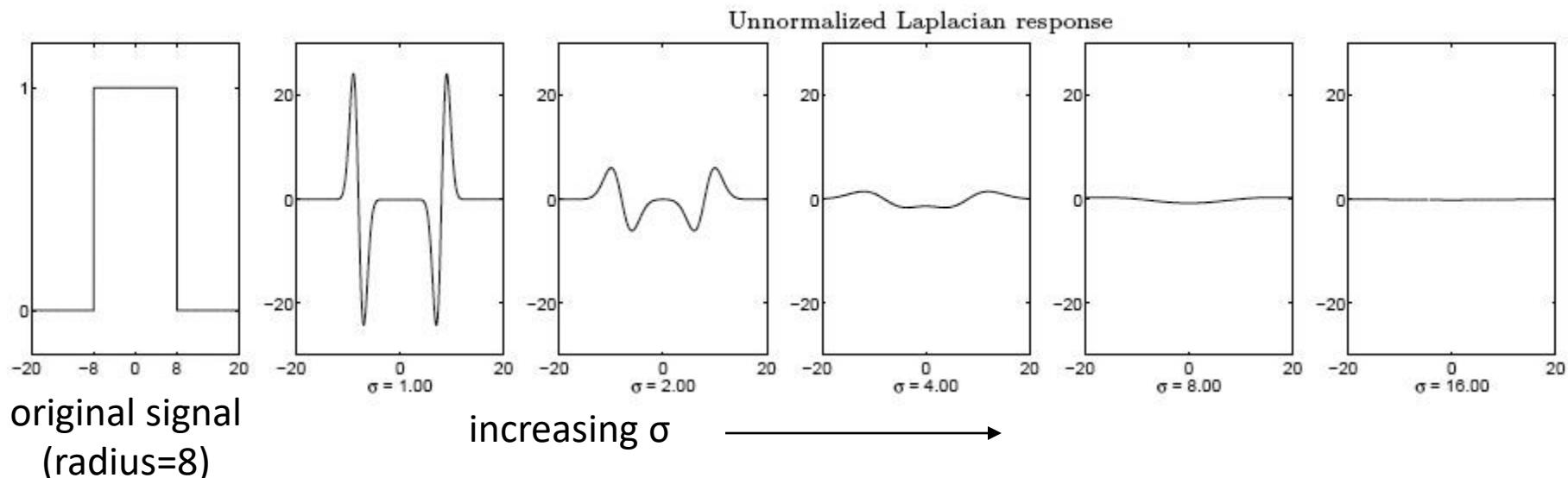


Scale-normalized:

$$\nabla_{\text{norm}}^2 g = \sigma^2 \left(\frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \right)$$

Scale selection

- We want to find the **characteristic scale** of the **blob** by convolving it with **Laplacians** at several **scales** and looking for the **maximum response**
- However, Laplacian response decays as scale increases:

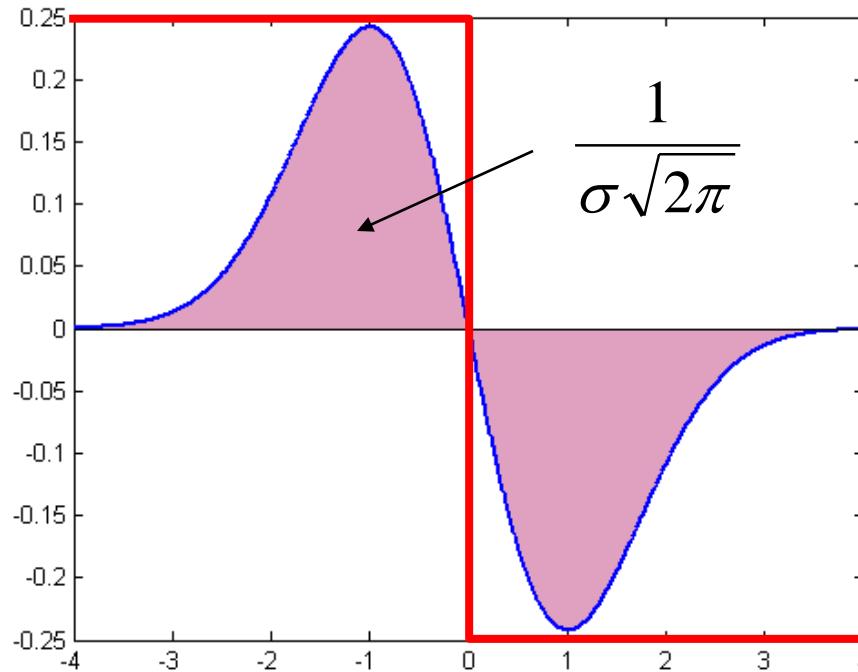


Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases
- To keep response the same (scale-invariant), must multiply Gaussian derivative by σ
- Laplacian is the second Gaussian derivative, so it must be multiplied by σ^2

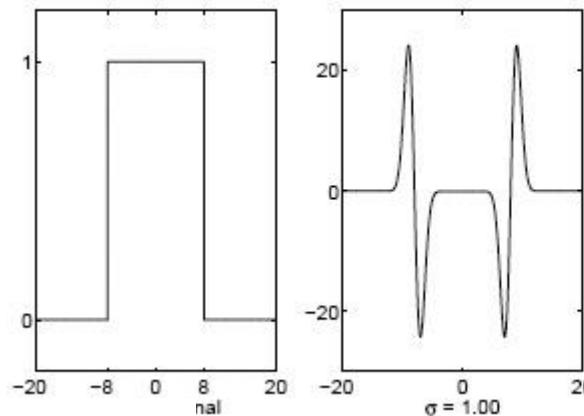
Scale normalization

- The response of a derivative of Gaussian filter to a perfect step edge decreases as σ increases

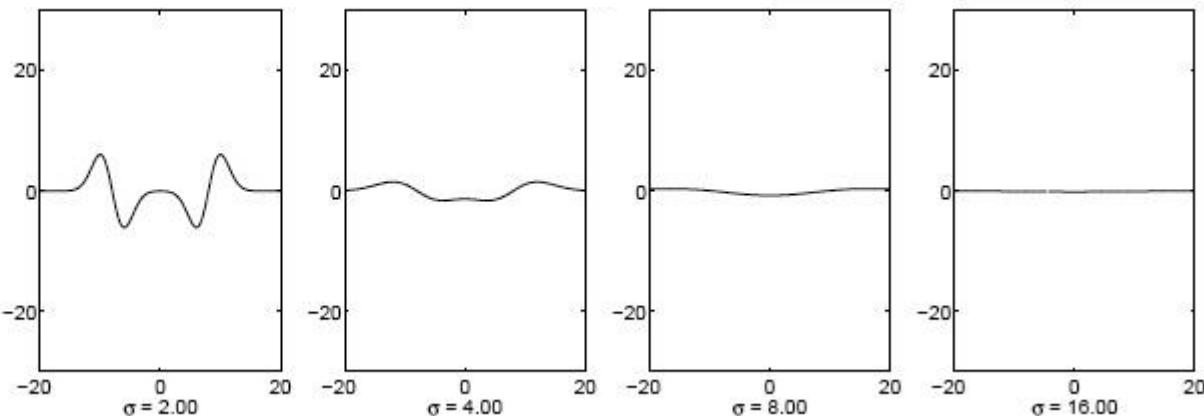


Effect of scale normalization

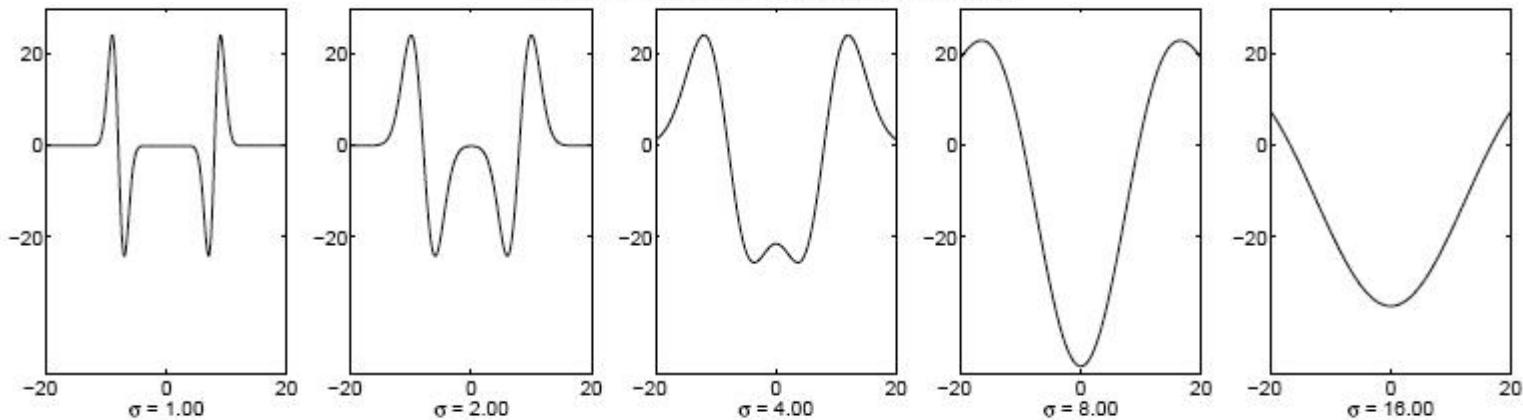
Original signal



Unnormalized Laplacian response

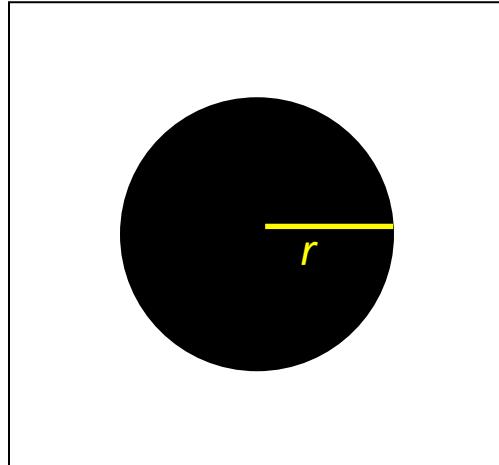


Scale-normalized Laplacian response

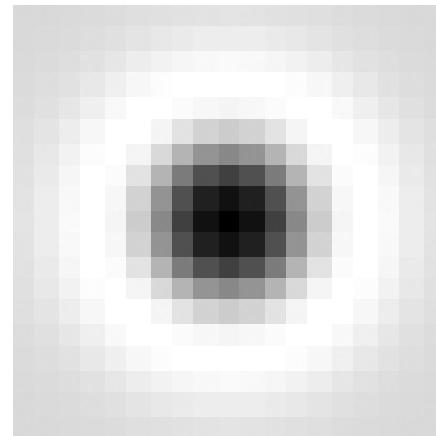


Scale selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r ?



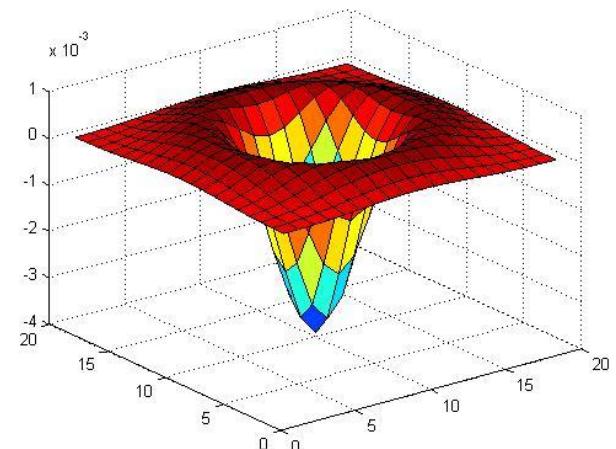
image



Digital Image Processing CS 4650/7650
ECE 4655/7655

Laplacian

Slide Credit: Dr. Svetlana Lazebnik



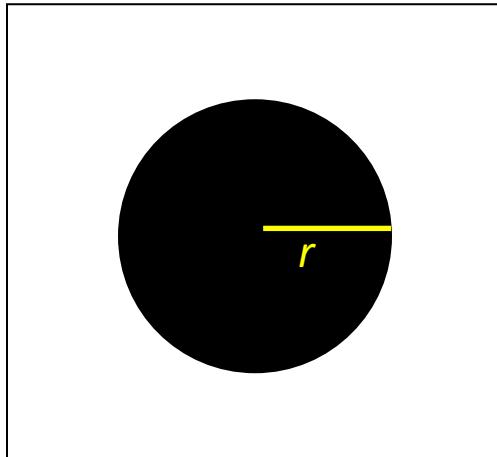
Scale selection

- At what scale does the Laplacian achieve a maximum response to a binary circle of radius r ?
- To get maximum response, the zeros of the Laplacian have to be aligned with the circle
- The Laplacian is given by (up to scale):

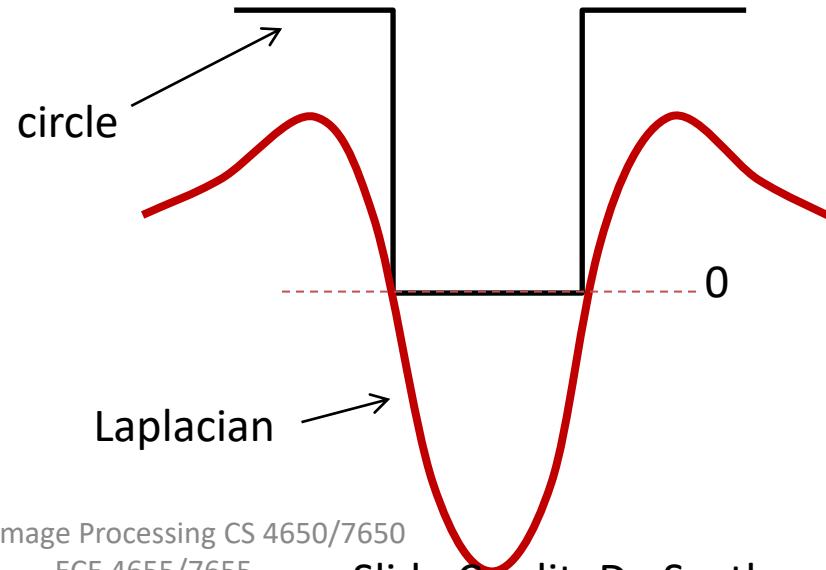
$$(x^2 + y^2 - 2\sigma^2) e^{-(x^2+y^2)/2\sigma^2}$$

- Therefore, the maximum response occurs at

$$\sigma = r / \sqrt{2}.$$

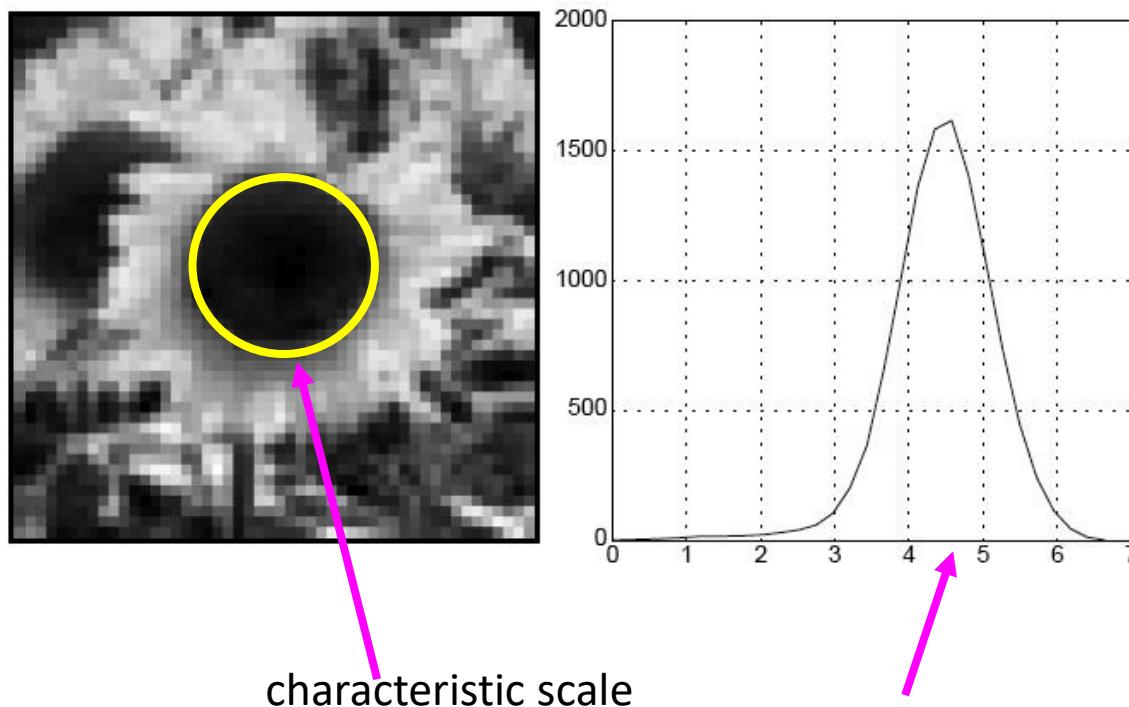


image



Characteristic scale

- We define the **characteristic scale of a blob** as the **scale that produces peak of Laplacian response** in the blob center



T. Lindeberg (1998). ["Feature detection with automatic scale selection."](#)
International Journal of Computer Vision **30** (2); pp 77--116.

Digital Image Processing CS 4650/7650

ECE 4655/7655

Slide Credit: Dr. Svetlana Lazebnik

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales

Scale-space blob detector: Example



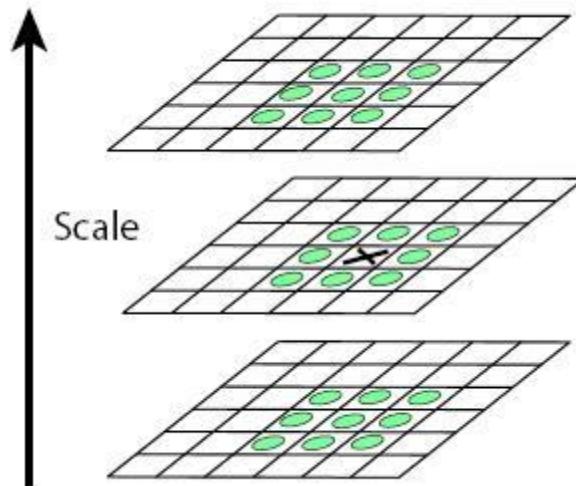
Scale-space blob detector: Example



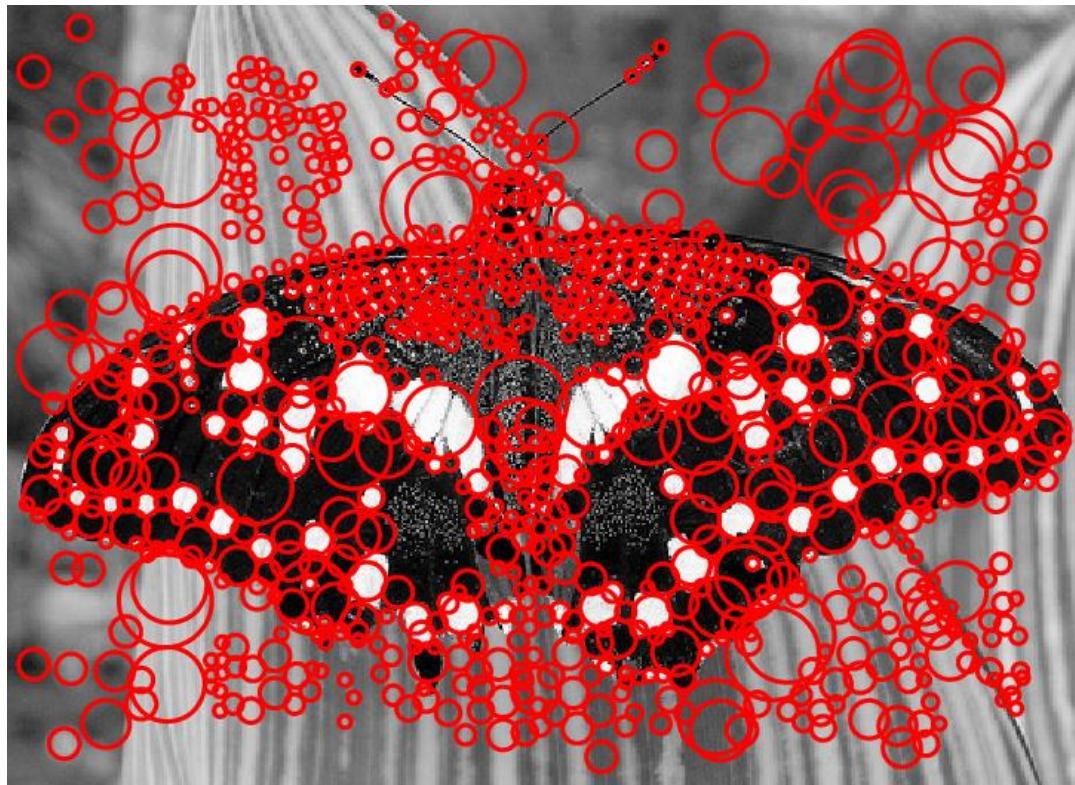
$\sigma = 11.9912$

Scale-space blob detector

1. Convolve image with scale-normalized Laplacian at several scales
2. Find maxima of squared Laplacian response in scale-space



Scale-space blob detector: Example



Efficient implementation

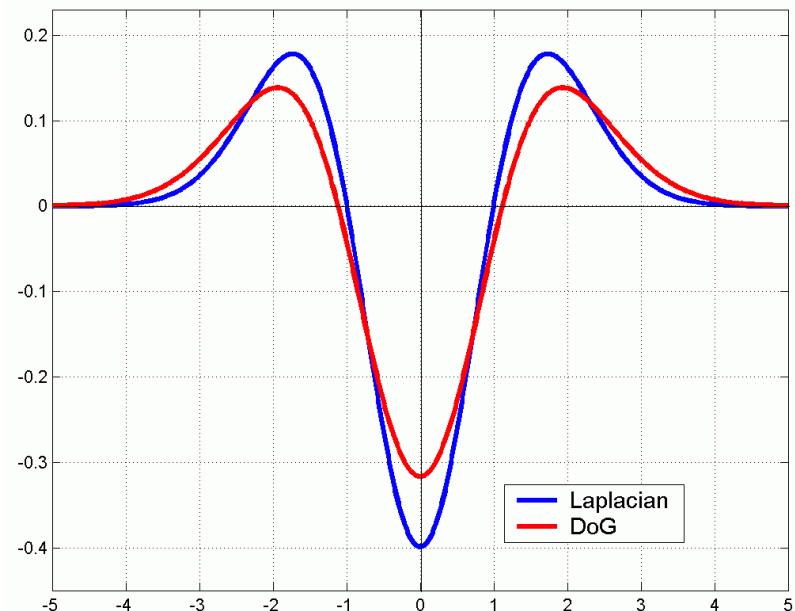
- Approximating the Laplacian with a difference of Gaussians:

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

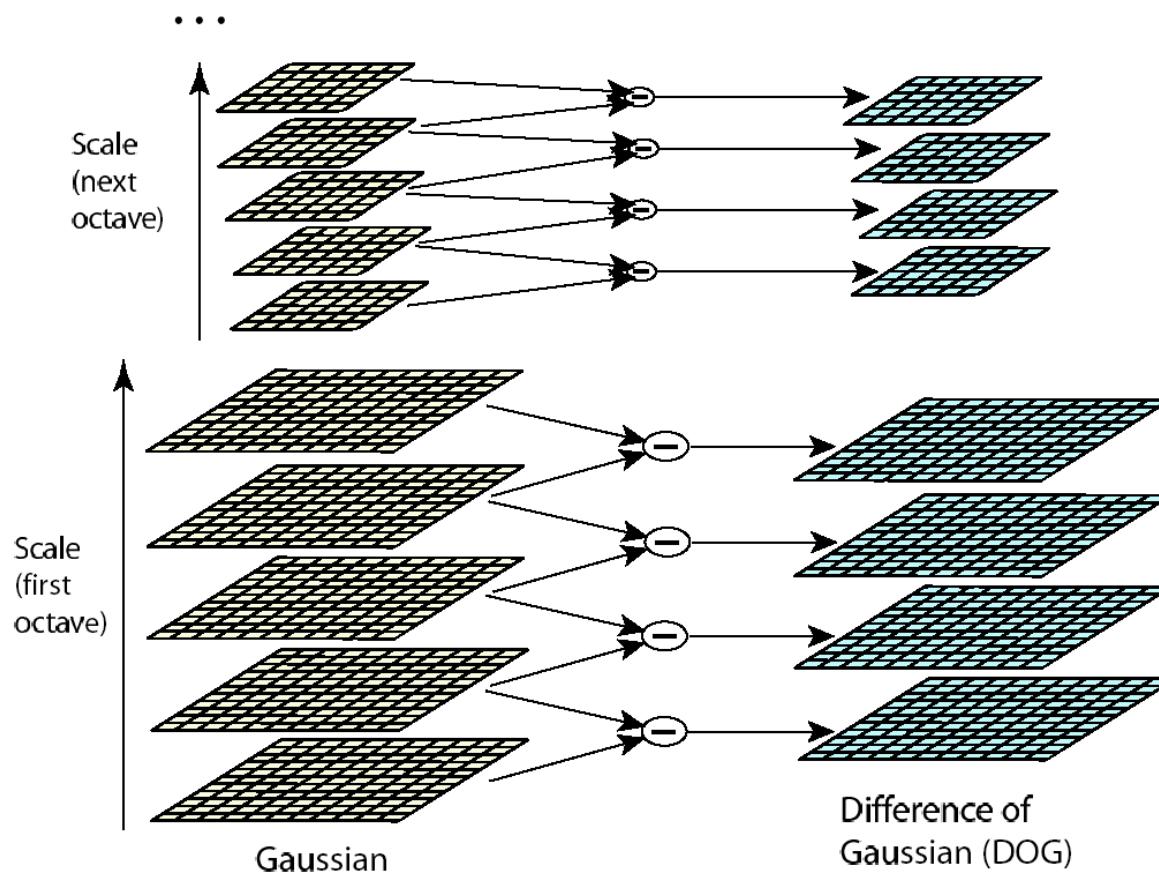
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

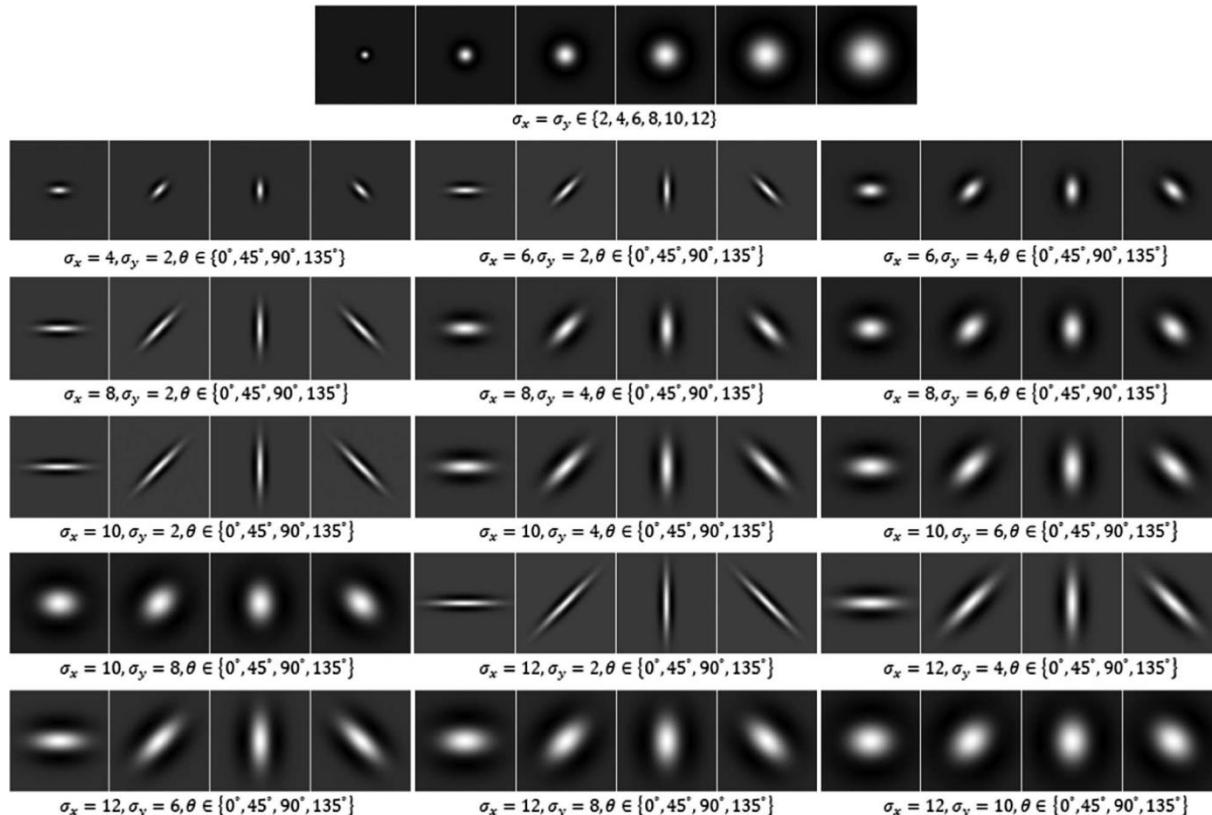


Efficient implementation

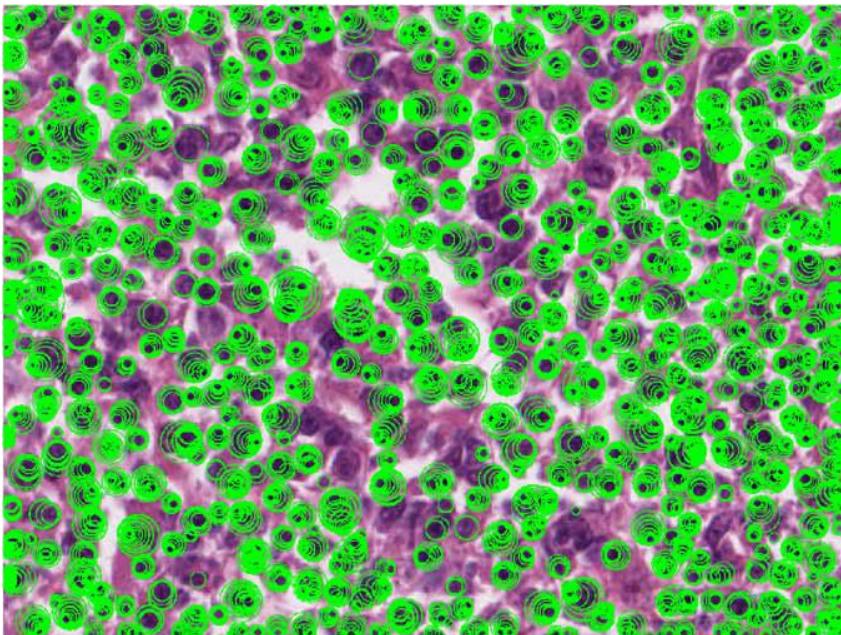


David G. Lowe. "[Distinctive image features from scale-invariant keypoints.](#)" IJCV 60 (2), pp. 91-110, 2004.

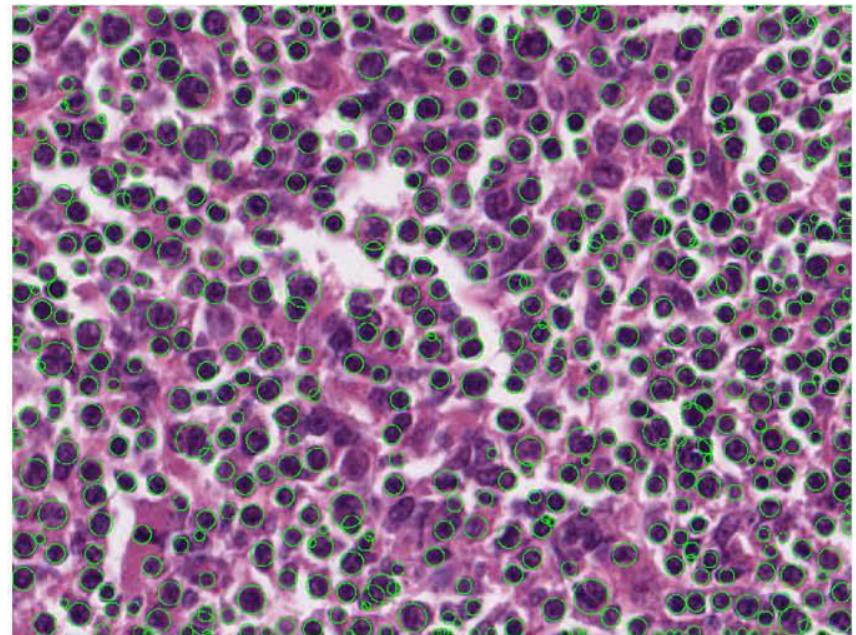
Generalized Laplacian of Gaussian filter



Kong, Hui, Hatice Cinar Akakin, and Sanjay E. Sarma. "A generalized Laplacian of Gaussian filter for blob detection and its applications." *IEEE transactions on cybernetics* 43.6 (2013): 1719-1733.



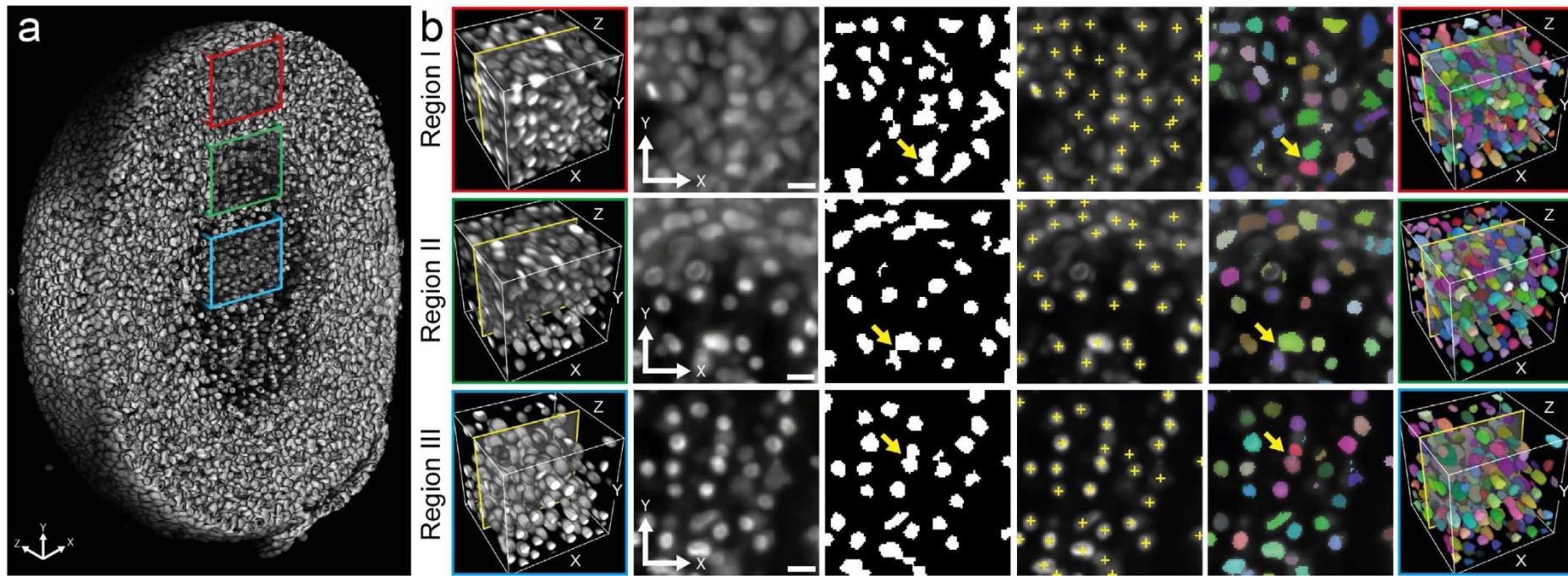
(a)



(b)

(a) and (b) Detected blobs by circular LoG blob detector before and after pruning, respectively.

Kong, Hui, Hatice Cinar Akakin, and Sanjay E. Sarma. "A generalized Laplacian of Gaussian filter for blob detection and its applications." *IEEE transactions on cybernetics* 43.6 (2013): 1719-1733.



Multiscale image analysis reveals structural heterogeneity of the cell microenvironment in homotypic spheroids

Alexander Schmitz

, Sabine C. Fischer

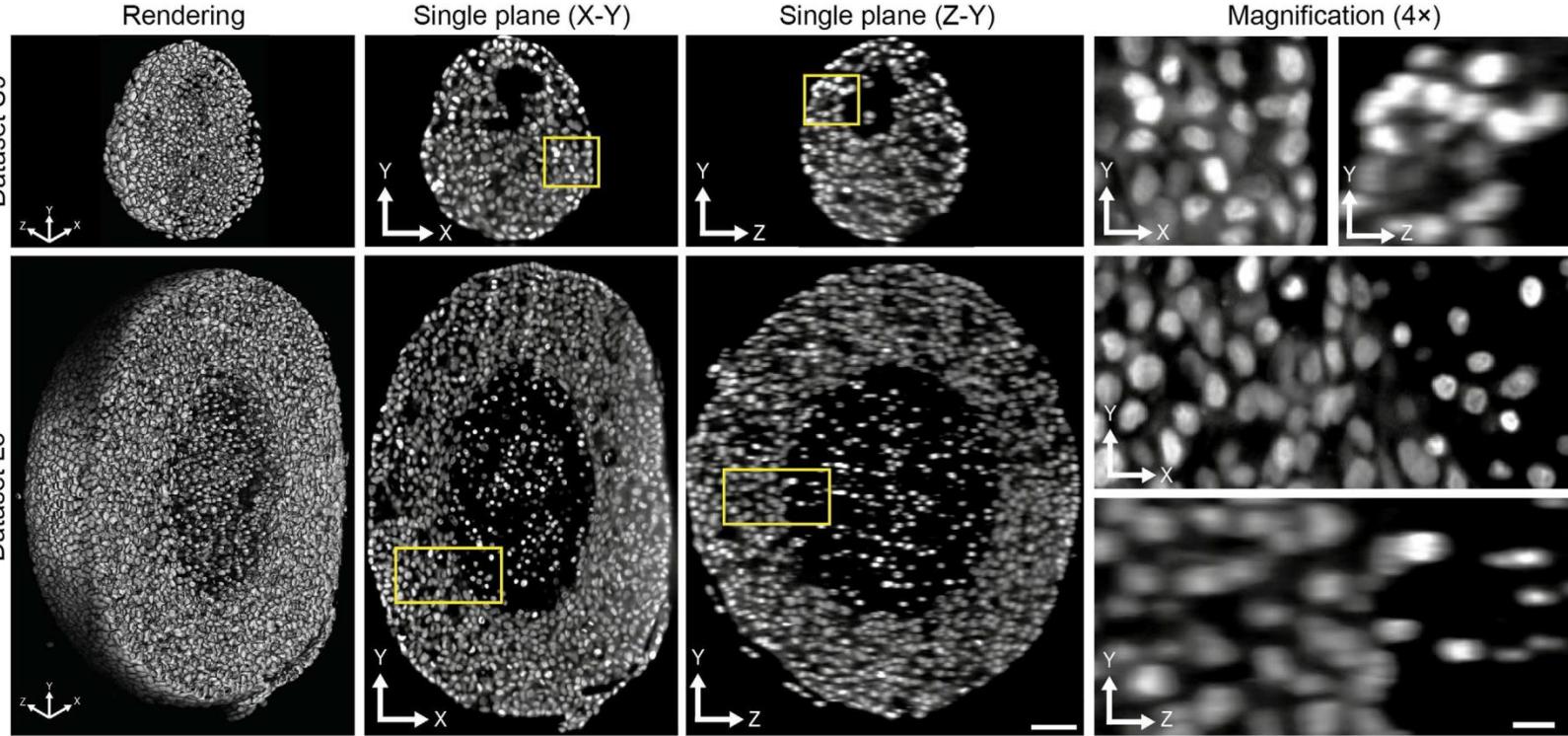
, Christian Mattheyer

, Francesco Pampaloni

& Ernst H. K. Stelzer

Scientific Reports 7, Article number: 43693

Dataset S9



HW 2 Test Images



HW 2 Test Images



HW 2 Test Images



Feature Detection and Matching

Szeliski Computer Vision Book
2nd Edition Chapter 7.1 available online at
<https://szeliski.org/Book/>

Gonzalez & Woods Digital Image Processing Book
Chapter 12.6-12.7

Some Relevant References

- David Lowe's (2004) paper, which describes the development and refinement of his Scale Invariant Feature Transform (SIFT).
- Survey and evaluation papers covering both feature detection: (Schmid, Mohr, and Bauckhage 2000; Mikolajczyk, Tuytelaars, Schmid et al. 2005; Tuytelaars and Mikolajczyk 2007)
- Feature descriptors (Mikolajczyk and Schmid 2005). Shi and Tomasi (1994) and Triggs(2004) also provide nice reviews of feature detection techniques.

Recent Surveys:

- Ma, Jiayi, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. "Image matching from handcrafted to deep features: A survey." International Journal of Computer Vision (2021)
- Jing, Junfeng, Tian Gao, Weichuan Zhang, Yongsheng Gao, and Changming Sun. "Image feature information extraction for interest point detection: A comprehensive review." IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

Features Detection and Matching

Feature detection and matching are an essential component of many computer vision applications. Sample Application:

- Align the two images so that they can be seamlessly stitched into a composite mosaic.
- Establish a dense set of correspondences so that a 3D model can be constructed, or an in-between view can be generated.
- Point features can be used to find a sparse set of corresponding locations in different images :
 - Motion tracking,
 - Object recognition,
 - Indexing and database retrieval,
 - Robot navigation... other



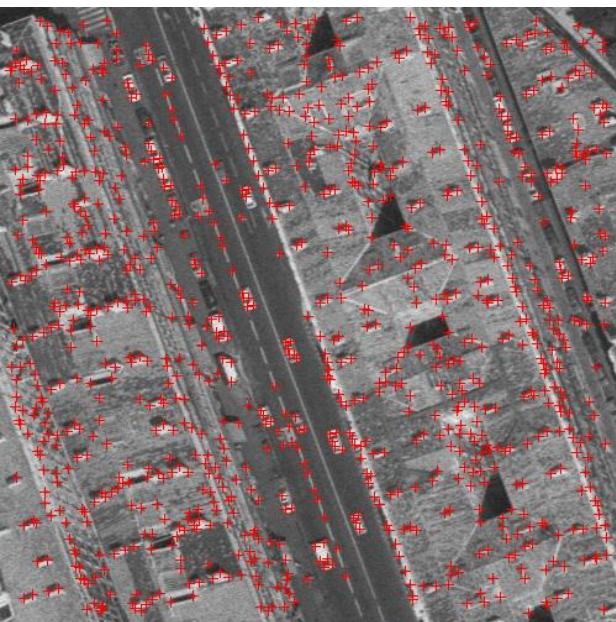
Type of Features

- Keypoint features (or interest points, corners): specific locations in the images, such as mountain peaks, building corners....
 - Described by the appearance of patches of pixels surrounding the point location (Section 7.1).
- Edges, contours:, e.g., the profile of mountains against the sky, (Section 7.2).
 - Can be matched based on their orientation and local appearance (edge profiles)
 - Edges can be grouped into curves or straight line segments (they can also be used to find vanishing points -> camera parameters)
- Regions

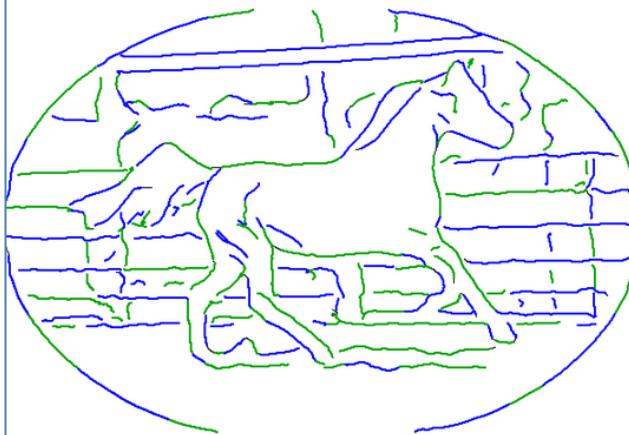


Local features

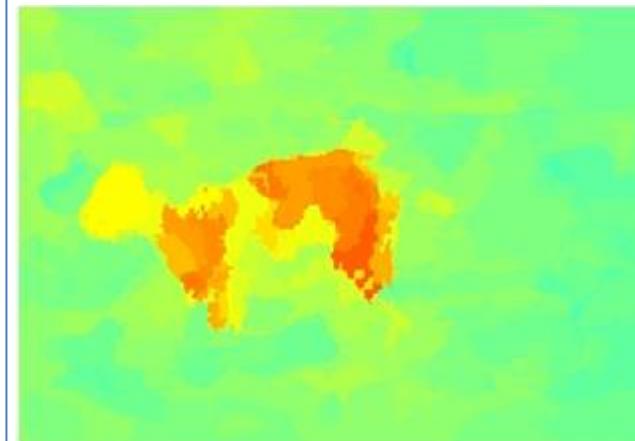
Interest Points



Contours/Line segments



Regions/Segmentation



Slide adapted
from by Cordelia Schmid

Properties of Local Features

Locality

- features are local, so robust to occlusion and clutter

Distinctiveness

- can differentiate a large database of objects

Quantity

- hundreds or thousands in a single image

Efficiency

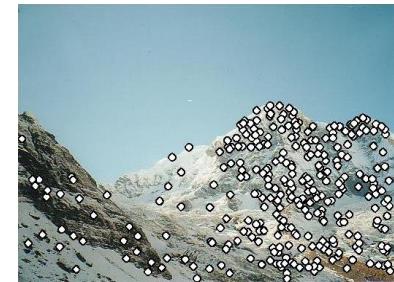
- real-time performance achievable

Generality

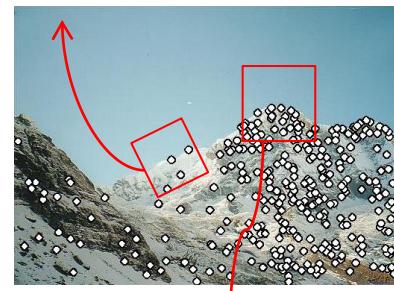
- exploit different types of features in different situations

Local Features: Detection and Matching Pipeline

1. **Feature detection (extraction)** : each image is searched for locations that are likely to match well in other images.
2. **Feature description** : each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors.
3. **Feature matching**: efficiently searches for likely matching candidates in other images.
4. **Feature tracking** : is an alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.



$$\mathbf{x}_1 = [x_1^{(1)}, \dots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \dots, x_d^{(2)}]$$

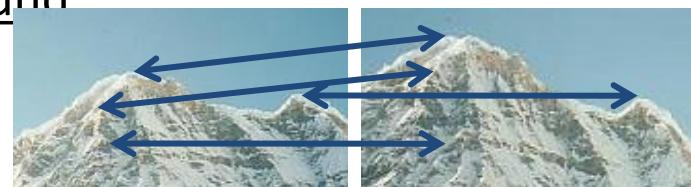
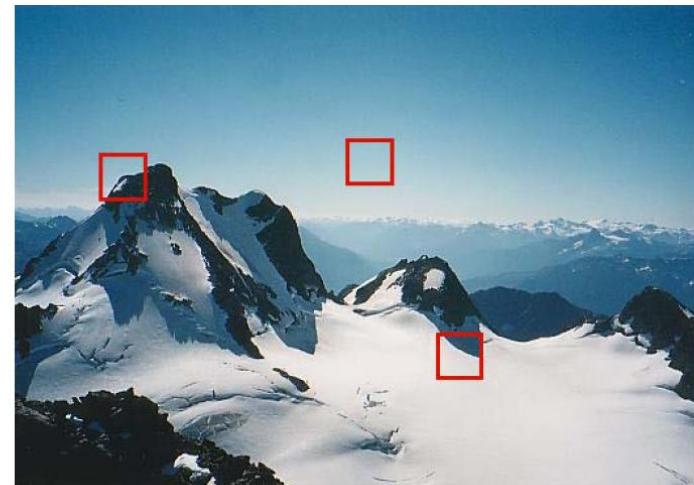
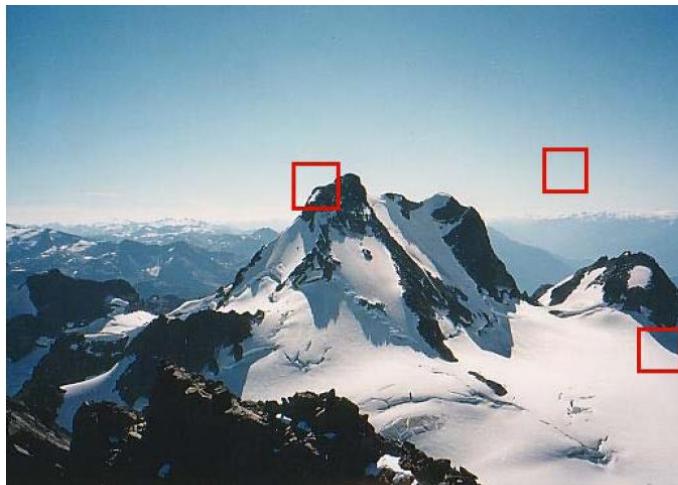


Figure credit: Kristen Grauman

Step 1: Feature Detection

Feature detection (extraction) : each image is searched for locations that are likely to match well in other images.

- How can we find image locations where we can reliably find correspondences with other images, i.e., what are good features to track (Shi and Tomasi 1994; Triggs 2004)?



Feature Detectors (Sec 7.1.1)

Observations:

- Textureless patches: Nearly impossible to localize.
- Patches with large contrast changes (gradients): Easier to localize (straight line segments at a single orientation suffer from the aperture problem it is only possible to align the patches along the direction normal to the edge direction) .
- Patches with gradients in at least two (significantly) different orientations: Easiest to localize.

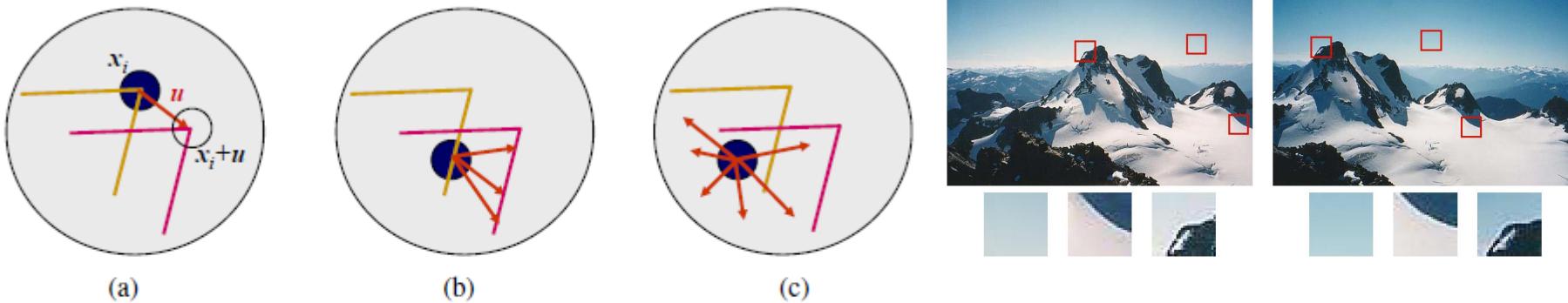
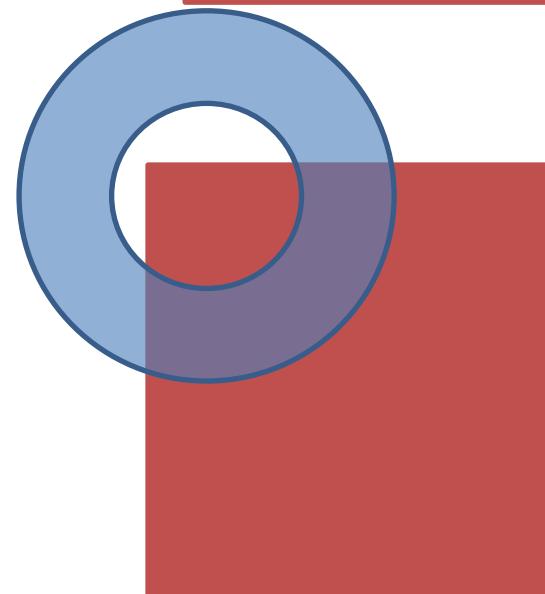
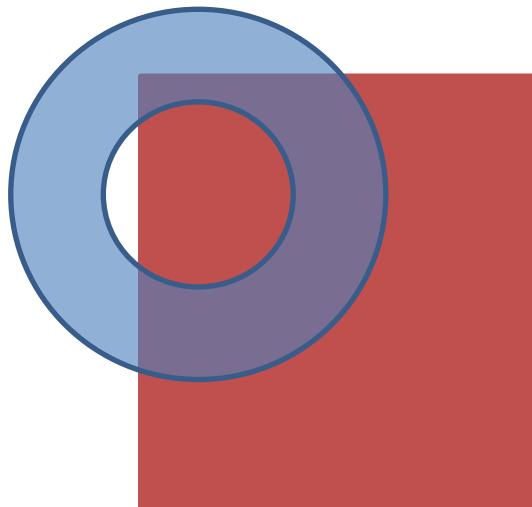
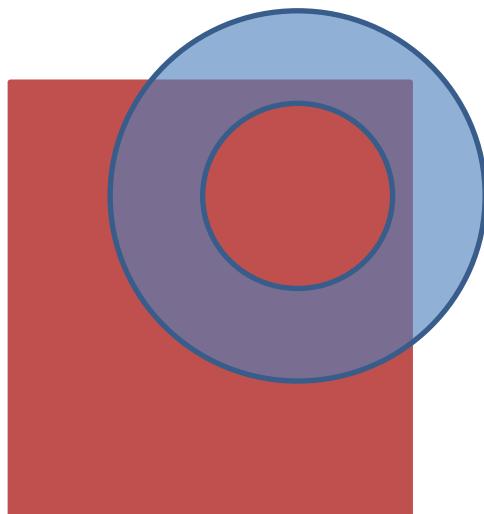
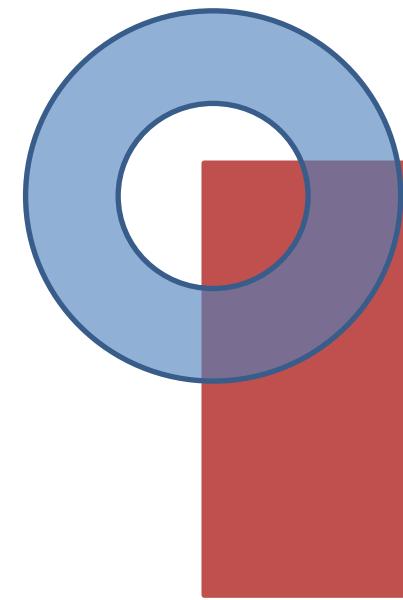
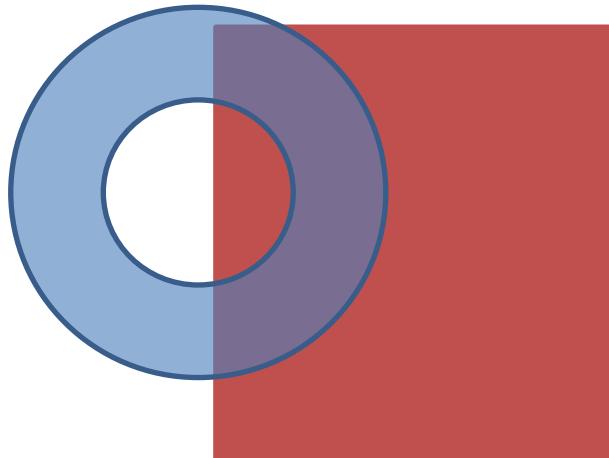
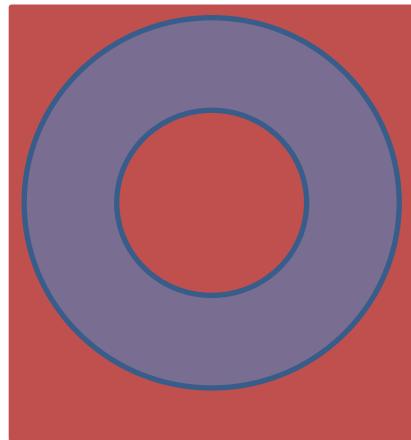


Figure 4.4 Aperture problems for different image patches: (a) stable (“corner-like”) flow; (b) classic aperture problem (barber-pole illusion); (c) textureless region. The two images I_0 (yellow) and I_1 (red) are overlaid. The red vector u indicates the displacement between the patch centers and the $w(x_i)$ weighting function (patch window) is shown as a dark circle.

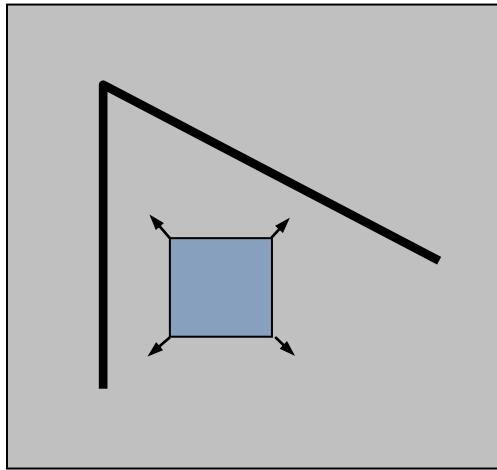
Aperture Problem



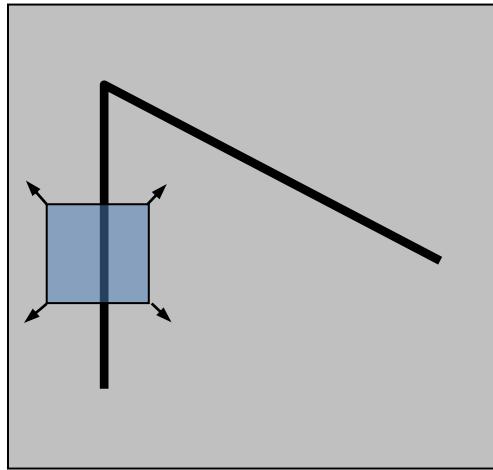
Feature Detection: Auto-Correlation Function

Local measure of feature uniqueness

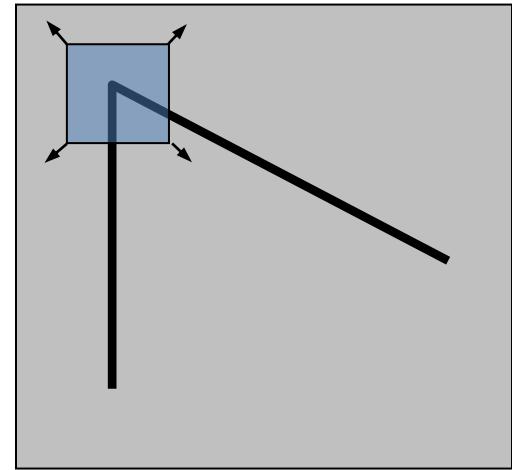
- How does the window change when you shift it?
- Shifting the window in *any direction* causes a *big change*



“flat” region:
no change in all
directions



“edge”:
no change along
the edge direction

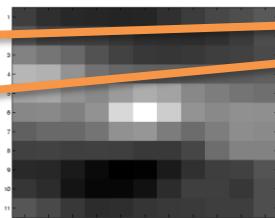


“corner”:
significant change
in all directions

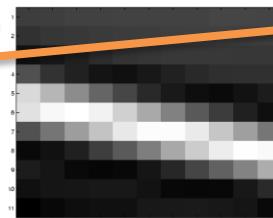
Feature Detection: Auto-Correlation Function



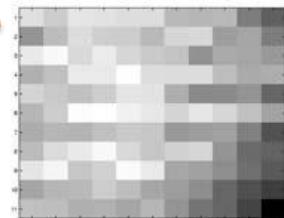
Good unique minimum



1D aperture problem



No good peak



Feature Detection: Auto-Correlation Function

$$A = \begin{bmatrix} G(\sigma) \otimes I_x^2 & G(\sigma) \otimes I_x I_y \\ G(\sigma) \otimes I_x I_y & G(\sigma) \otimes I_y^2 \end{bmatrix}$$

A is referred to variously as

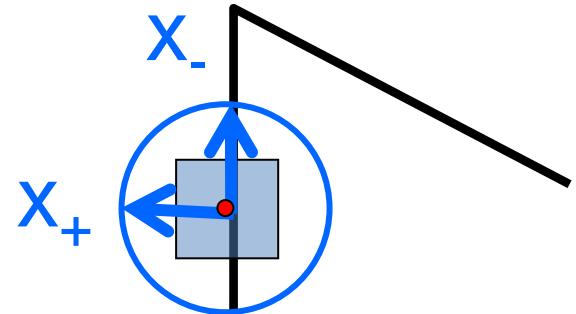
- ***structure tensor***,
- ***autocorrelation matrix*** or
- ***second moment matrix***.

It captures the intensity structure of the local neighborhood, and its eigenvalues provide a rotationally invariant description of the neighborhood.

Feature Detection: Mathematics

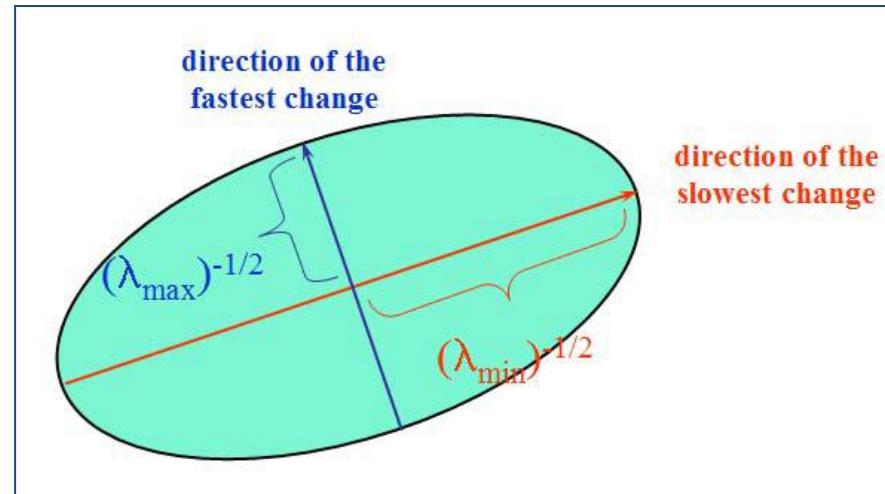
This can be rewritten:

$$E(u, v) = \sum_{(x,y) \in W} [u \ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_A \begin{bmatrix} u \\ v \end{bmatrix}$$



Eigenvalues and eigenvectors of A

- Define shifts with the smallest and largest change (E value)
- x_+ = direction of largest increase in E.
- λ_+ = amount of increase in direction x_+
- x_- = direction of smallest increase in E.
- λ_- = amount of increase in direction x_-



Quick eigenvalue/eigenvector review

The **eigenvectors** of a matrix \mathbf{A} are the vectors \mathbf{x} that satisfy:

$$A\mathbf{x} = \lambda\mathbf{x}$$

The scalar λ is the **eigenvalue** corresponding to \mathbf{x}

The eigenvalues are found by solving:

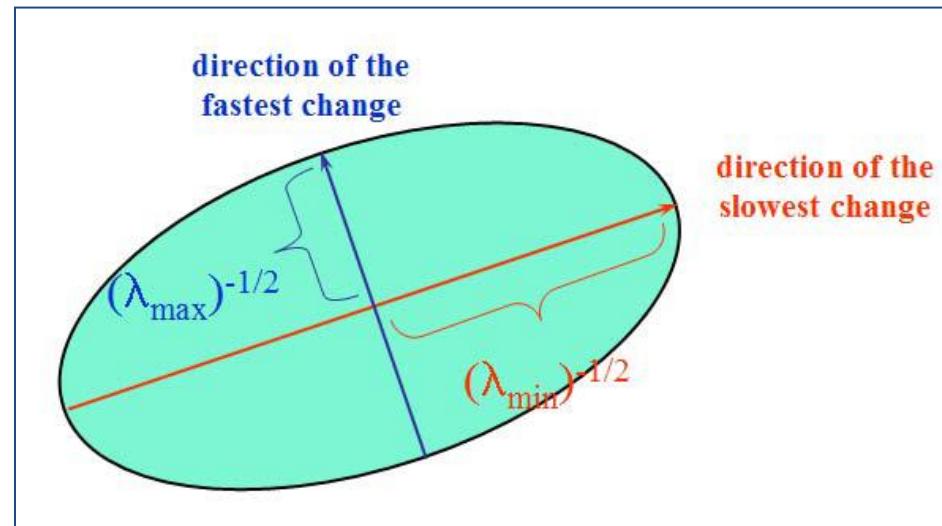
$$\det(A - \lambda I) = 0 \quad \det \begin{bmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{bmatrix} = 0$$

The solution:

$$\lambda \pm = \frac{1}{2} \left[(a_{11} + a_{22}) \pm \sqrt{4a_{12}a_{21} + (a_{11}-a_{22})^2} \right]$$

Once you know λ , you find \mathbf{x} by solving

$$\begin{bmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 0$$



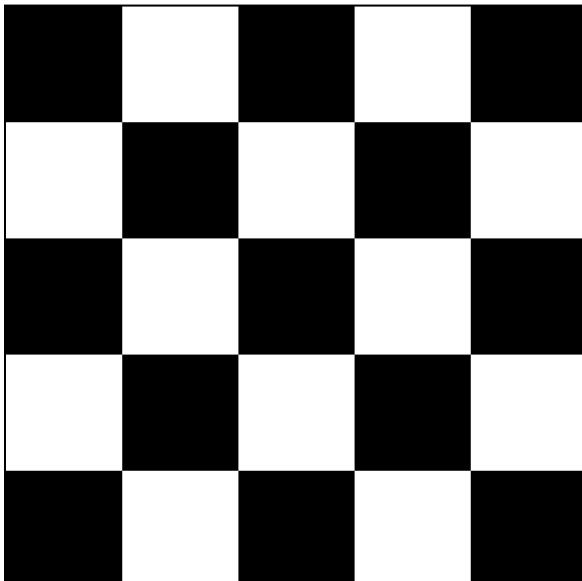
Feature Detection: Mathematics

How are λ_+ , x_+ , λ_- , and x_- relevant for feature detection?

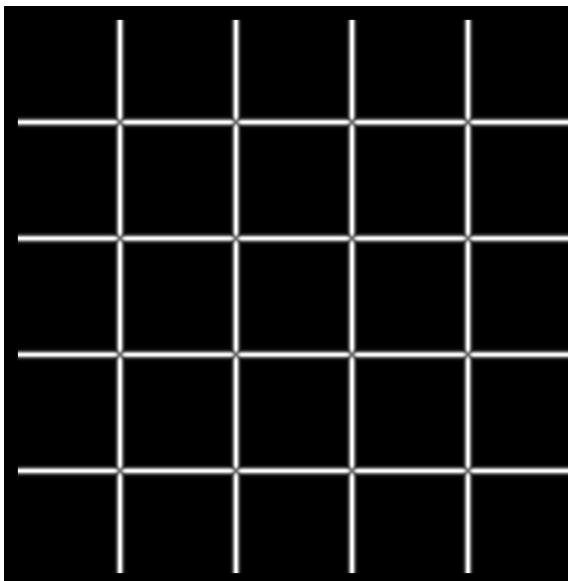
- What's our feature scoring function?

Want $E(u,v)$ to be *large* for small shifts in *all* directions

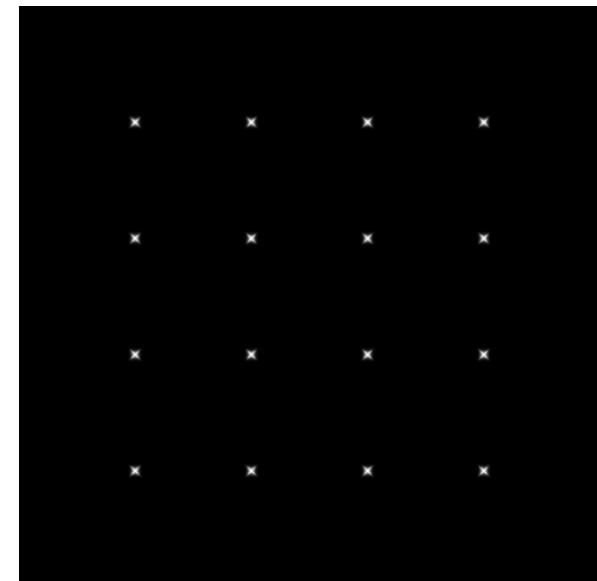
- the *minimum* of $E(u,v)$ should be large, over all unit vectors $[u \ v]$
- this minimum is given by the smaller eigenvalue (λ_-) of H



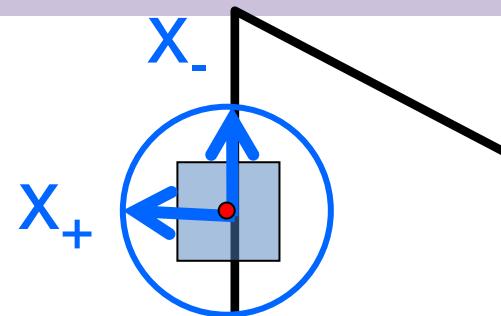
I



λ_+



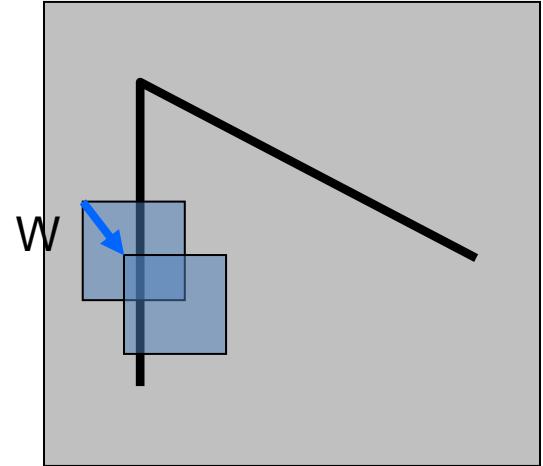
λ_-



Derivation: Auto-Correlation Function

Consider shifting the window W by (u,v)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD “error” of $E(u,v)$:



$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

Derivation: Feature Detection: Mathematics

Small Motion Assumption

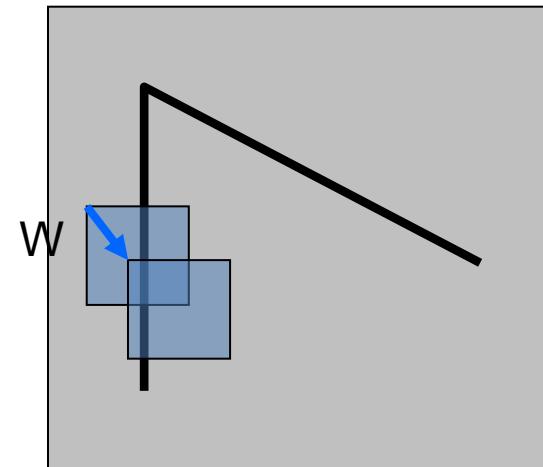
$$E(u, v) = \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2$$

From Taylor
series expansion:

$$I(x + u, y + v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$



Derivation: Feature Detection: Mathematics

$$\begin{aligned} E(u, v) &= \sum_{(x,y) \in W} [I(x + u, y + v) - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix} - I(x, y)]^2 \\ &\approx \sum_{(x,y) \in W} [[I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}]^2 \end{aligned}$$

$$E(u, v) = \sum_{(x,y) \in W} [u \ v] \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Auto-correlation matrix
(structure tensor, second moment matrix)
Slide by Xuejin Chen

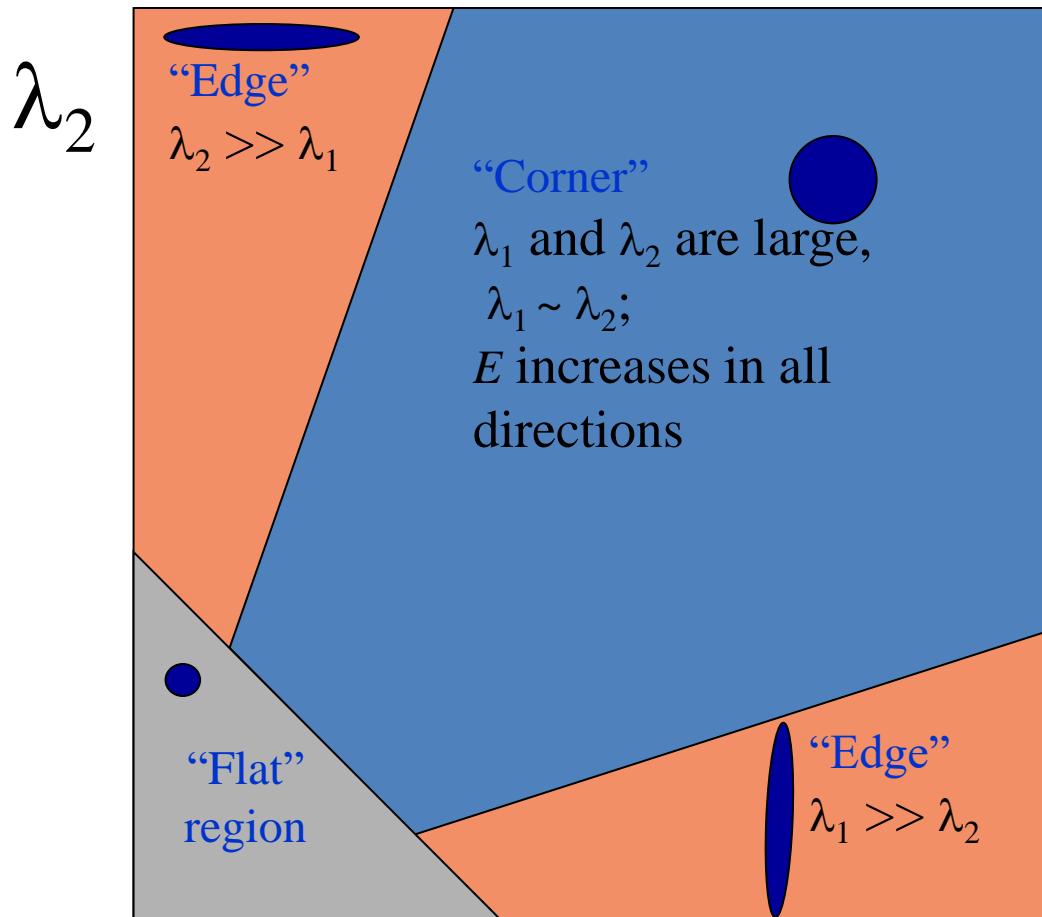
Feature Detection: Mathematics

Classification of image points using eigenvalues of A

Eigenvalues and eigenvectors of A

- Define shifts with the smallest and largest change (E value)
- \mathbf{x}_+ = direction of largest increase in E.
- λ_+ = amount of increase in direction \mathbf{x}_+
- \mathbf{x}_- = direction of smallest increase in E.
- λ_- = amount of increase in direction \mathbf{x}_-

λ_1 and λ_2 are small;
E is almost constant
in all directions



Some Feature Detectors

Shi and Tomasi 1994

- Find points with large response ($\lambda_>$ threshold)
- Choose those points where $\lambda_>$ is a local maximum as features

Harris & Stephens 1988

$$\begin{aligned} R &= \det(\mathbf{A}) - \alpha (\text{trace}(\mathbf{A}))^2 \\ &= \lambda_0 \lambda_1 - \alpha (\lambda_0 + \lambda_1)^2 \end{aligned}$$

Harmonic Mean (Brown, Szeliski, Winder 2005)

$$f = \frac{\det(A)}{\text{trace}(A)} = \frac{\lambda_0 \lambda_1}{\lambda_0 + \lambda_1}$$

Outline of Feature Point Detection

$$A = \begin{bmatrix} G(\sigma) \otimes I_x^2 & G(\sigma) \otimes I_x I_y \\ G(\sigma) \otimes I_x I_y & G(\sigma) \otimes I_y^2 \end{bmatrix}$$

1. Compute **gradient images** I_x and I_y by convolving the original image with derivatives of Gaussians
2. Compute the three images corresponding to the **products of these gradients** ($I_x I_x$, $I_x I_y$, $I_y I_y$).
3. Smooth $I_x I_x$, $I_x I_y$, $I_y I_y$ using a weighting matrix (i.e. convolving each of these images with a larger Gaussian). This **reduces noise and improves stability and reliability** of the detector.
4. Compute a **scalar interest measure** using one of the formulas discussed above.
5. Find **local maxima above a certain threshold** and report them as **detected feature point locations**.

Case Study: Harris Detector

$$R = \det(A) - \alpha (\text{trace}(A))^2$$

$$\det(A) = \lambda_+ \lambda_-$$

$$\text{trace}(A) = \lambda_+ + \lambda_-$$

α – empirical constant, $\alpha = 0.04-0.06$)

- R depends only on eigenvalues of A
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region

λ_2

“Edge”

$$R < 0$$

“Corner”

$$R > 0$$

“Flat”

$$|R| \text{ small}$$

“Edge”

$$R < 0$$

λ_1

- The trace is the sum of the diagonals, i.e., $\text{trace}(A) = a_{11} + a_{22}$
- **Very similar to λ_- but less expensive (no square root)**

Slide adapted
from Xuejin Chen

Harris Detector: Workflow

The Algorithm:

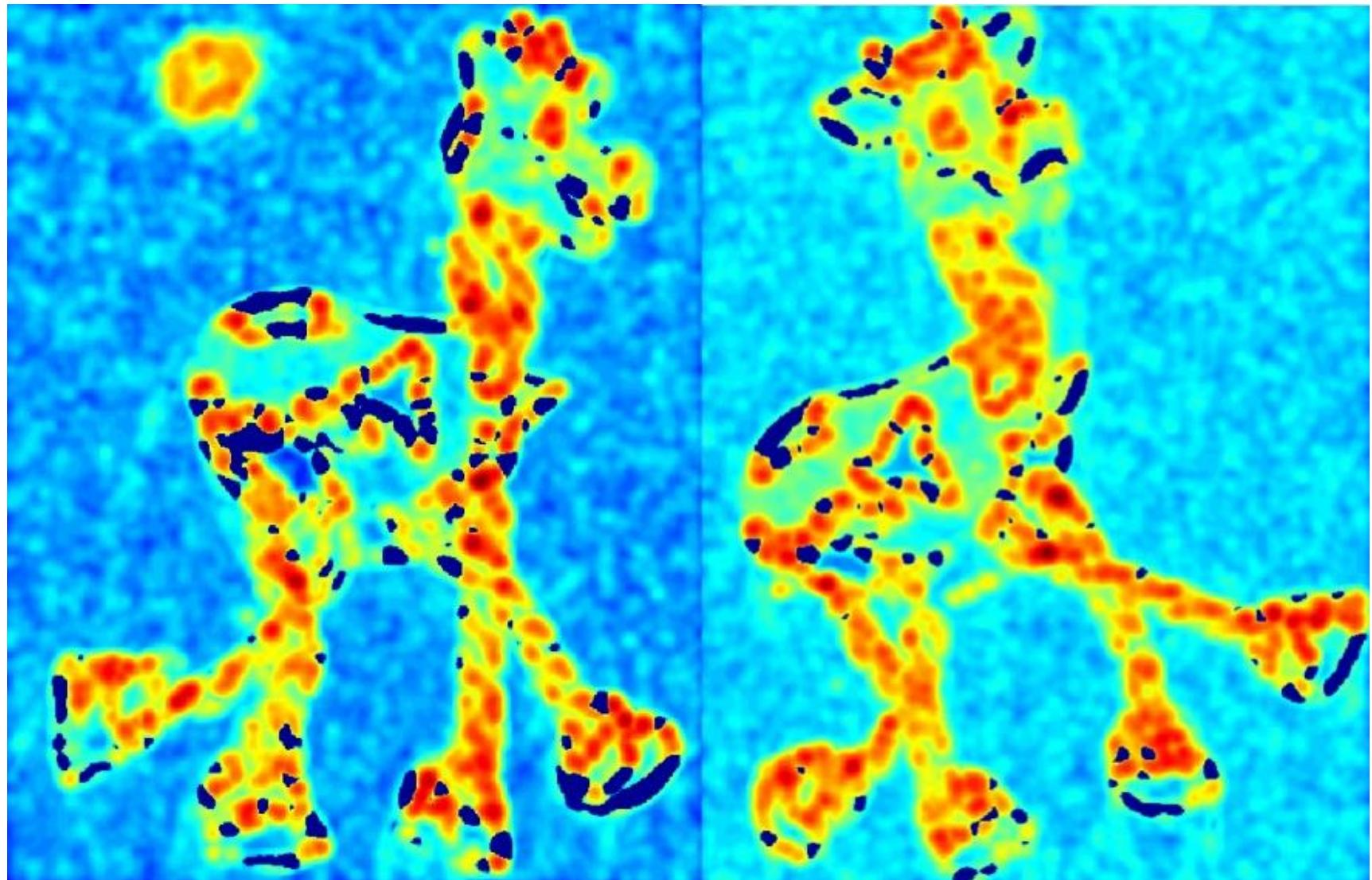
1. Find points with large corner response function R ($R >$ threshold)
2. Take the points of local maxima of R



Slide by Steve Seitz, Rick Szeliski, Jiun-Hung Chen

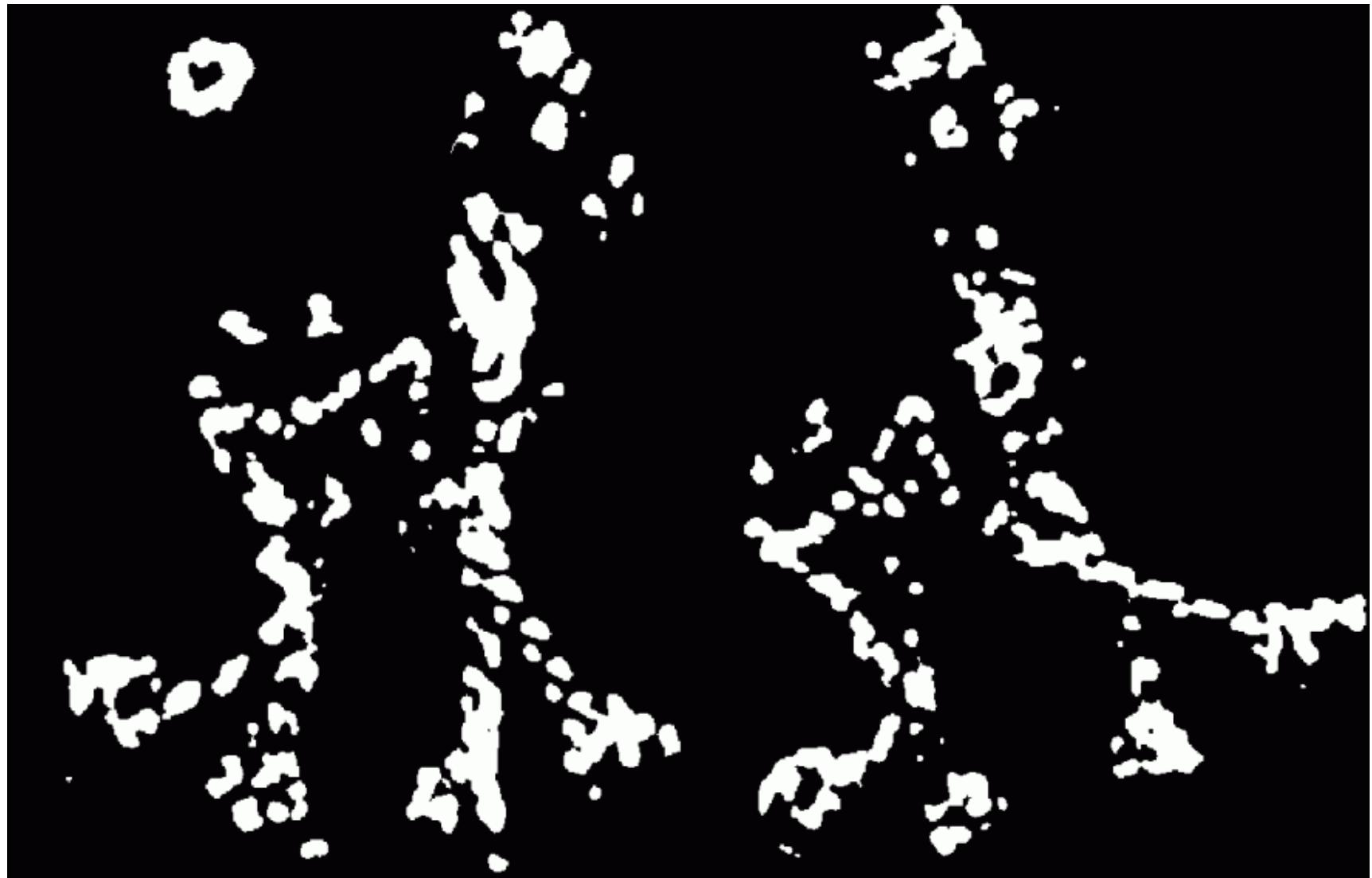
Harris Detector: Workflow

Compute corner response R



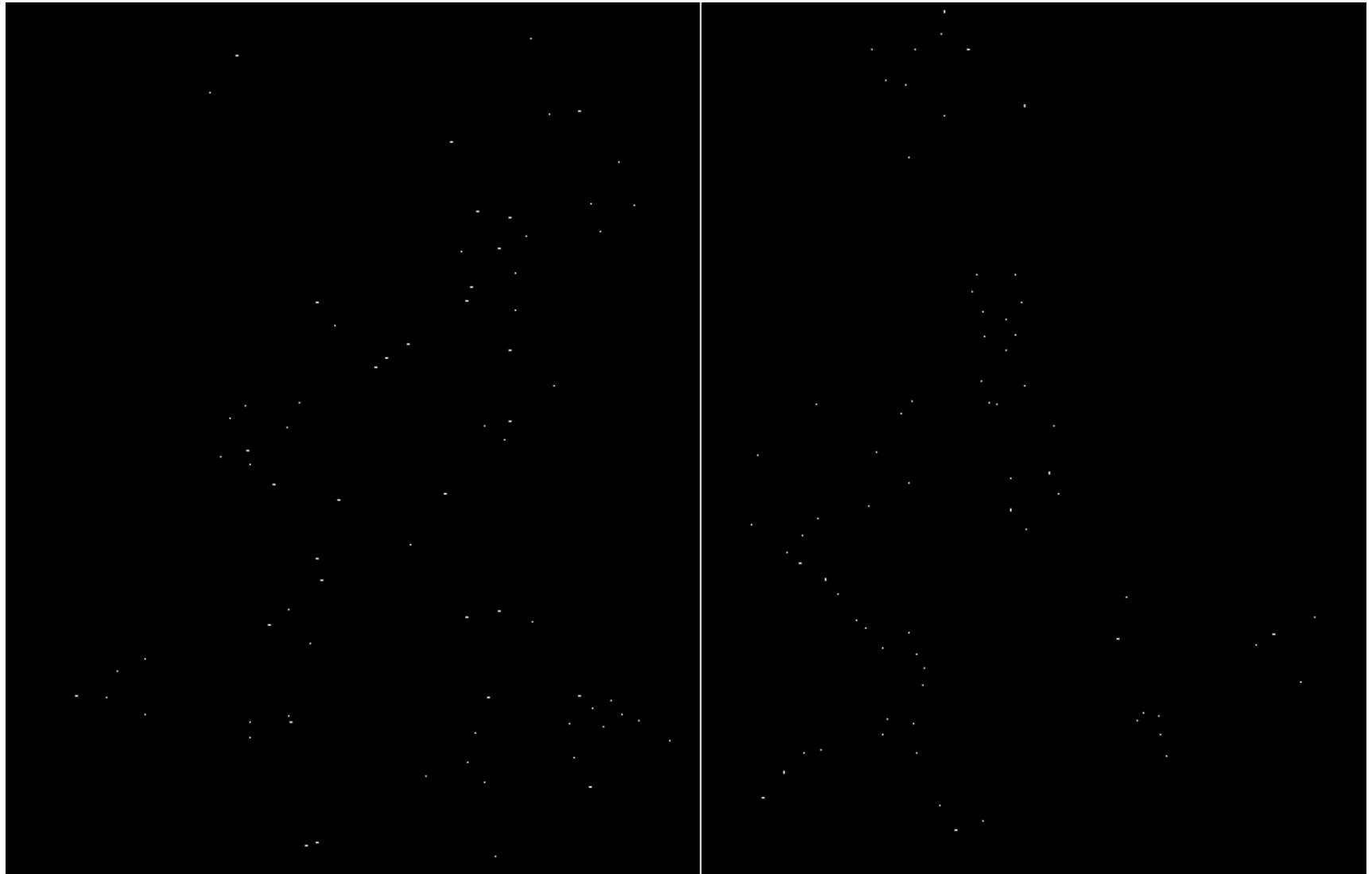
Harris Detector: Workflow

Find points with large corner response: $R > \text{threshold}$



Harris Detector: Workflow

Take only the points of local maxima of R



Harris Detector: Workflow



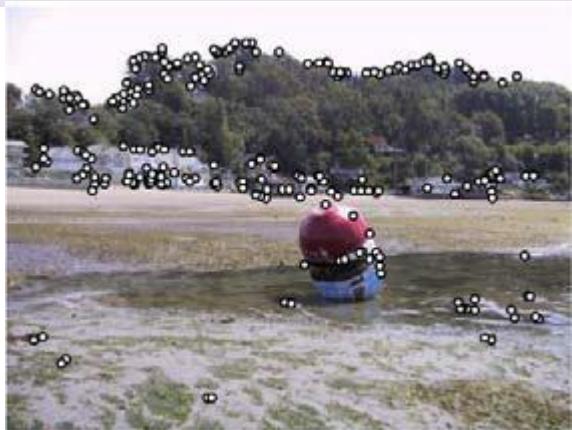
Slide by Steve Seitz, Rick Szeliski, Jiun-Hung Chen

Outline of Feature Detection

$$A = \begin{bmatrix} G(\sigma) \otimes I_x^2 & G(\sigma) \otimes I_x I_y \\ G(\sigma) \otimes I_x I_y & G(\sigma) \otimes I_y^2 \end{bmatrix}$$

1. Compute **gradient images** I_x and I_y by convolving the original image with derivatives of Gaussians
2. Compute the three images corresponding to the **products of these gradients** ($I_x I_x$, $I_x I_y$, $I_y I_y$).
3. Smooth the outer product from (2) using a weighting matrix (i.e. convolving each of these images with a larger Gaussian). This **reduces noise and improves stability and reliability** of the detector.
4. Compute a **scalar interest measure** using one of the formulas discussed above.
5. **Find local maxima above a certain threshold** and report them as **detected feature point locations**.

Adaptive Non-Maximal Suppression (ANMS)



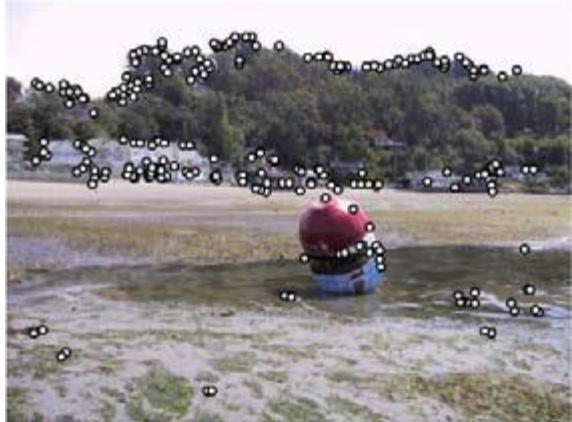
(a) Strongest 250



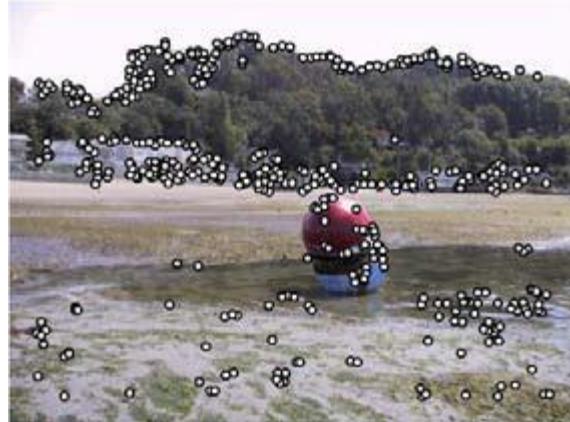
(b) Strongest 500

- Most detectors:
 - Simply look for **local maxima in the interest function**,
 - this can lead to **uneven distribution of feature points** across the image e.g. points will be denser in regions of high contrast.
- To mitigate the problem Brown, Szeliski, Winder 2005 proposed
 - Detect features that are both **local maxima and**
 - whose **response value is significantly (10%) greater than of all its neighbors within radius r**.

Adaptive Non-Maximal Suppression (ANMS)



(a) Strongest 250



(b) Strongest 500



(c) ANMS 250, r = 24



(d) ANMS 500, r = 16

Keypoint Detection and Matching Pipeline



1. **Feature detection (extraction)** : each image is searched for locations that are likely to match well in other images.
2. **Feature description** : each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors.
3. **Feature matching**: efficiently searches for likely matching candidates in other images.
4. **Feature tracking** : is an alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.

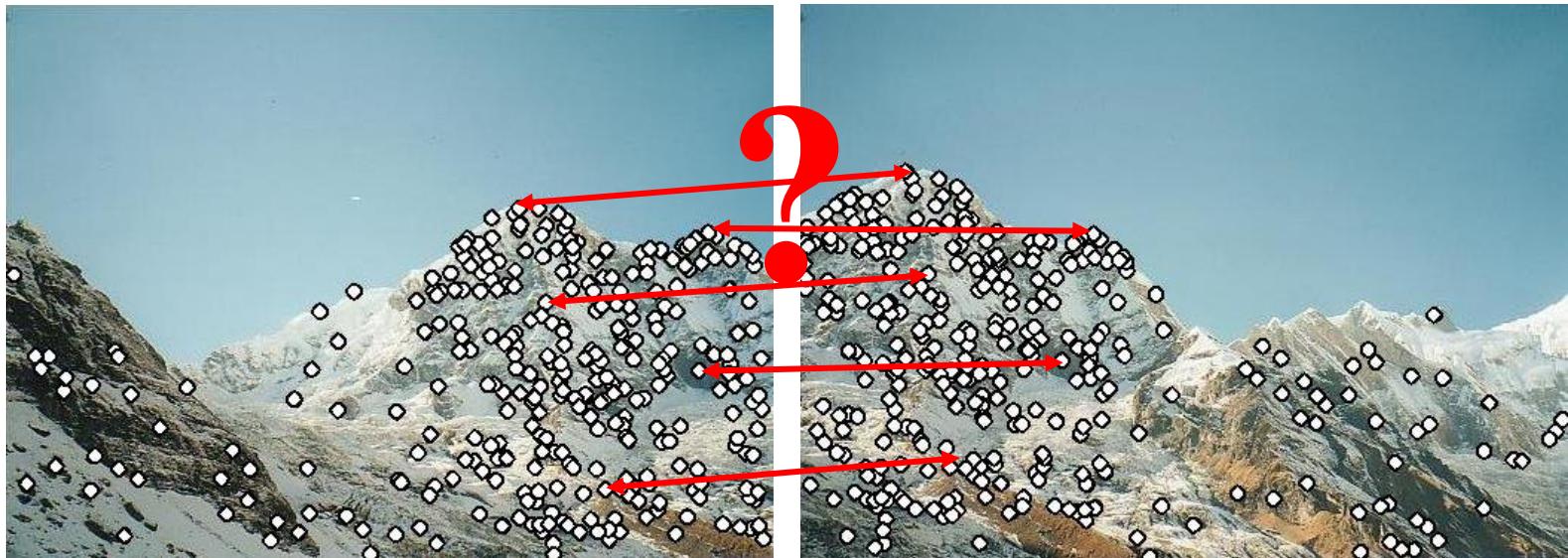
Relevant references:

- David Lowe's (2004) paper, which describes the development and refinement of his Scale Invariant Feature Transform (SIFT).
- Survey and evaluation papers covering both feature detection: (Schmid, Mohr, and Bauckhage 2000; Mikolajczyk, Tuytelaars, Schmid et al. 2005; Tuytelaars and Mikolajczyk 2007)
- Feature descriptors (Mikolajczyk and Schmid 2005). Shi and Tomasi (1994) and Triggs(2004) also provide nice reviews of feature detection techniques.

Step 2: Feature descriptors

We know how to detect good points

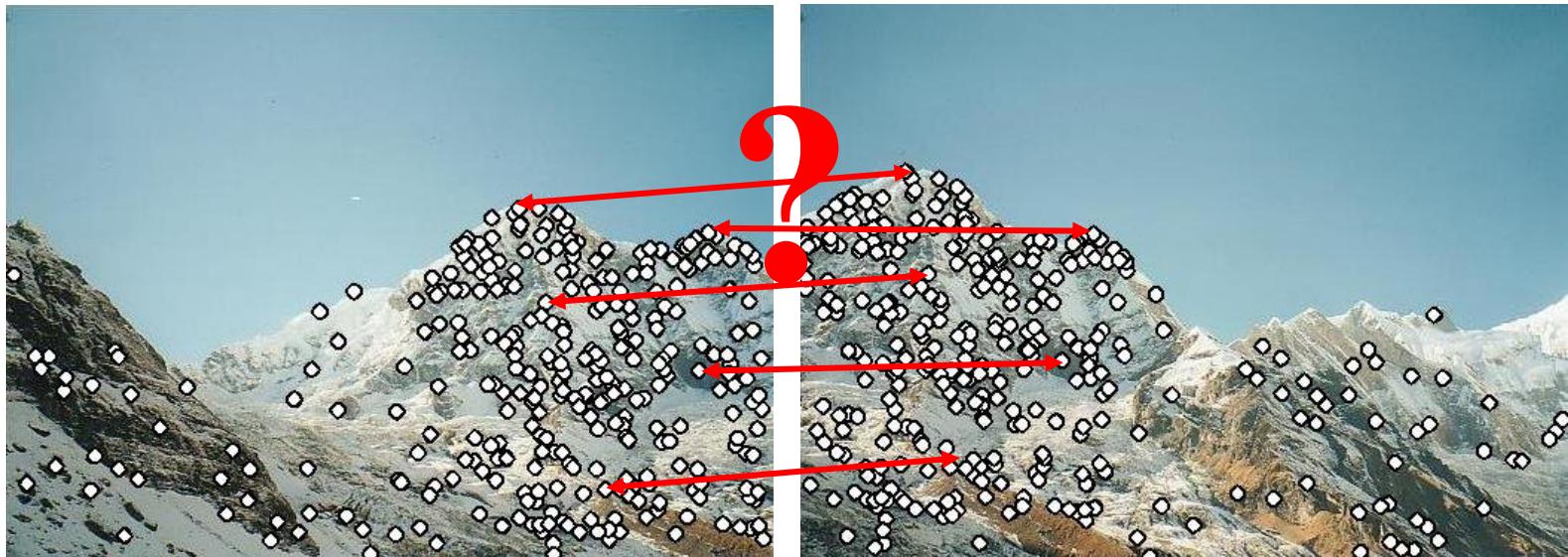
Next question: **How to match them?**



Feature descriptors

We know how to detect good points

Next question: **How to match them?**



Lots of possibilities (this is a popular research area)

- Simple option: match square windows around the point
- State of the art approach: SIFT
 - David Lowe, UBC <http://www.cs.ubc.ca/~lowe/keypoints/>

Measuring Repeatability

Large number of feature detectors,
which ones to use???

- Schmid, Mohr, and Bauckhage 2000 proposed repeatability.
- Repeatability of feature detector: Frequency with which keypoints detected in an image are found within ϵ pixels of corresponding location in a transformed image.
- They apply
 - Rotation
 - Scale
 - Illumination
 - Viewpoint changes
- + Noise

Difficulties

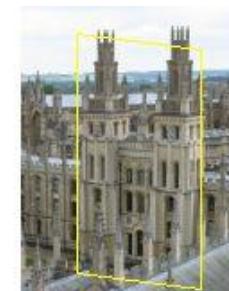
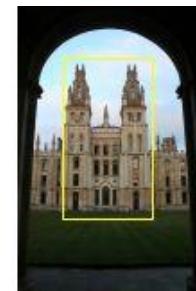
Finding the object despite possibly large changes in

- scale
- viewpoint,
- lighting and
- partial occlusion

→ requires Invariance



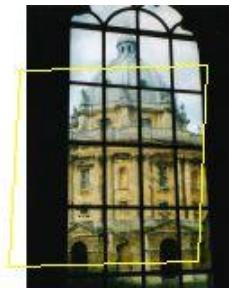
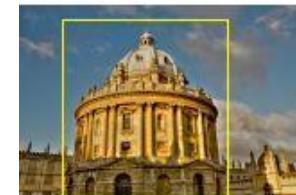
Scale



Viewpoint



Lighting



Occlusion

Good features should have the following properties [Tuytelaars-FGV 2008]:

Repeatability: Given two images of the same object or scene, taken under different viewing conditions, a high percentage of the features detected on the scene part visible in both images should be found in both images.

Distinctiveness/informativeness: The intensity patterns underlying the detected features should show a lot of variation, such that features can be distinguished and matched.

Locality: The features should be local, so as to reduce the probability of occlusion and to allow simple model approximations of the geometric and photometric deformations between two images taken under different viewing conditions (e.g., based on a local planarity assumption).

Quantity: The number of detected features should be sufficiently large, such that a reasonable number of features are detected even on small objects. However, the optimal number of features depends on the application. Ideally, the number of detected features should be adaptable over a large range by a simple and intuitive threshold. The density of features should reflect the information content of the image to provide a compact image representation.

Accuracy: The detected features should be accurately localized, both in image location, as with respect to scale and possibly shape.

Efficiency: Preferably, the detection of features in a new image should allow for time-critical applications.

Good features should have the following properties [Tuytelaars-FGV 2008]:

Repeatability (arguably the most important property of all), can be achieved in two different ways:

- **Invariance:** When large deformations are to be expected, the preferred approach is to model these mathematically if possible, and then develop methods for feature detection that are unaffected by these mathematical transformations.
- **Robustness:** In case of relatively small deformations, it often suffices to make feature detection methods less sensitive to such deformations, i.e., the accuracy of the detection may decrease, but not drastically so. Typical deformations that are tackled using robustness are image noise, discretization effects, compression artifacts, blur, etc. Also geometric and photometric deviations from the mathematical model used to obtain invariance are often overcome by including more robustness.

Invariance

Suppose we are comparing two images I_1 and I_2

- I_2 may be a transformed version of I_1
- What kinds of transformations are we likely to encounter in practice?

We'd like to find the same features regardless of the transformation

- This is called transformational ***invariance***
- Most feature methods are designed to be invariant to
 - **Translation,**
 - **2D rotation,**
 - **Scale**
- They can usually also handle
 - Limited **3D rotations** (SIFT works up to about 60 degrees)
 - Limited **affine transformations** (some are fully affine invariant)
 - Limited **illumination/contrast changes**

How to achieve invariance

Need both of the following:

1. Make sure your **detector** is invariant

- Harris is invariant to translation and rotation
- Scale is trickier
 - common approach is to detect features at many scales using a Gaussian pyramid (e.g., MOPS)
 - More sophisticated methods find “the best scale” to represent each feature (e.g., SIFT)

2. Design an **invariant feature descriptor**

- A descriptor captures the information in a region around the detected feature point
- The simplest descriptor: a square window of pixels
 - What's this invariant to?
- Let's look at some better approaches...

Scale Invariance

1. Extract features at variety of scales
2. Perform same operation at multiple resolutions in a pyramid
3. Match features at the same level

Suitable when images do not go through large scale changes! (aerial images, panorama with fixed focal length...)

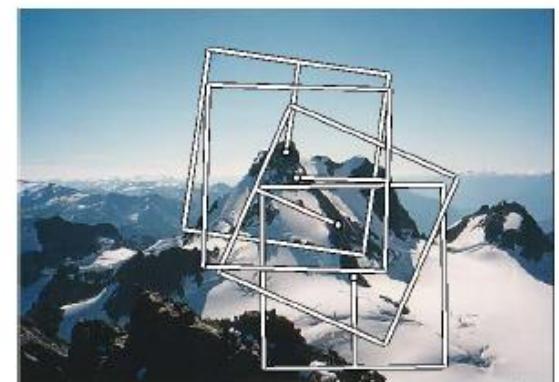
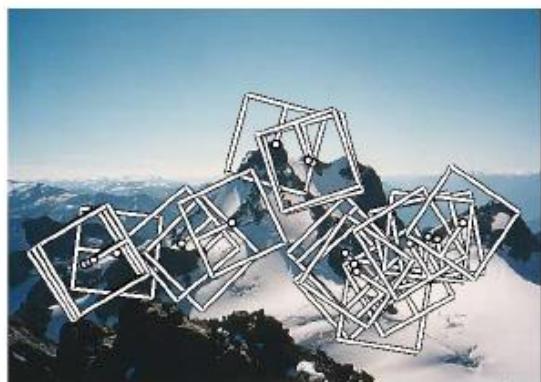
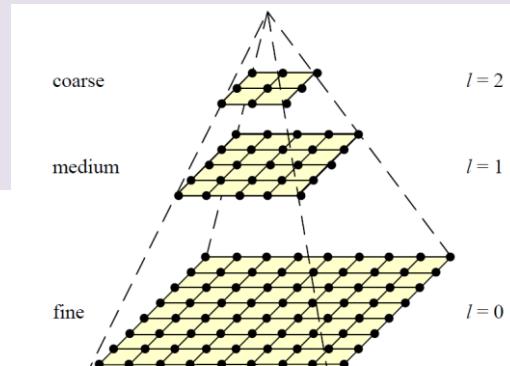


Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Scale Invariance

Problem: for most object recognition scale of object in the image is unknown

Solution: Instead of extracting features at many different scales then matching all of them.

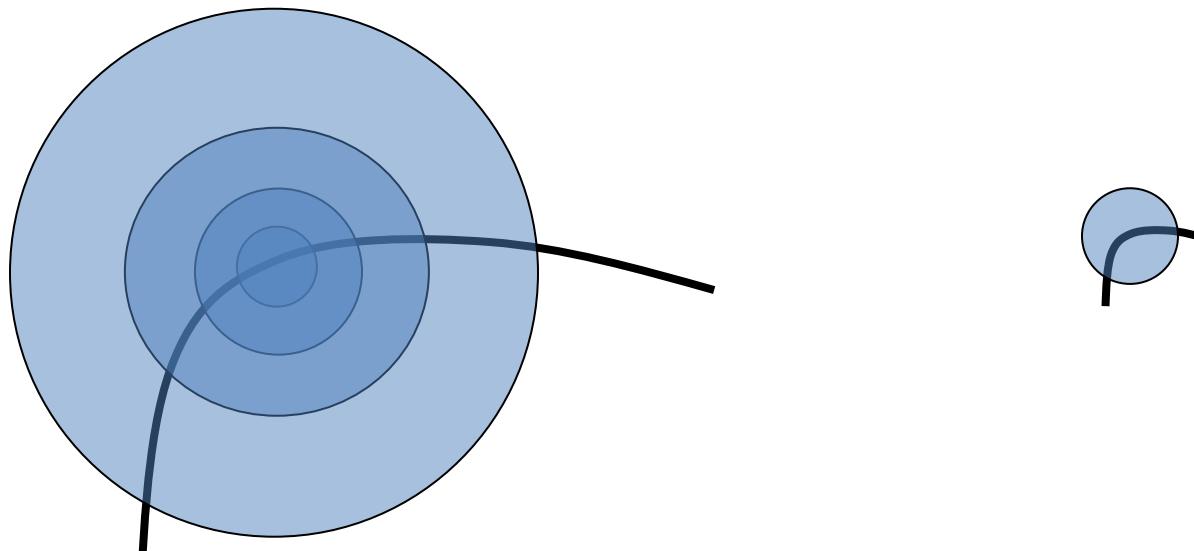
More efficient: Extract features that are stable in both location and scale (Lowe 2004, Mikolajczyk&Smith2004)

- Earlier on scale selection Lindeberg 1993&1998:
proposed – extrema in LoG (Laplacian of Gaussian)
- Lowe: DoG (difference of Gaussians)-look for 3D
(space+scale) maxima

More details on Multi-resolution representation See Szeliski Book Section 3.5.3.

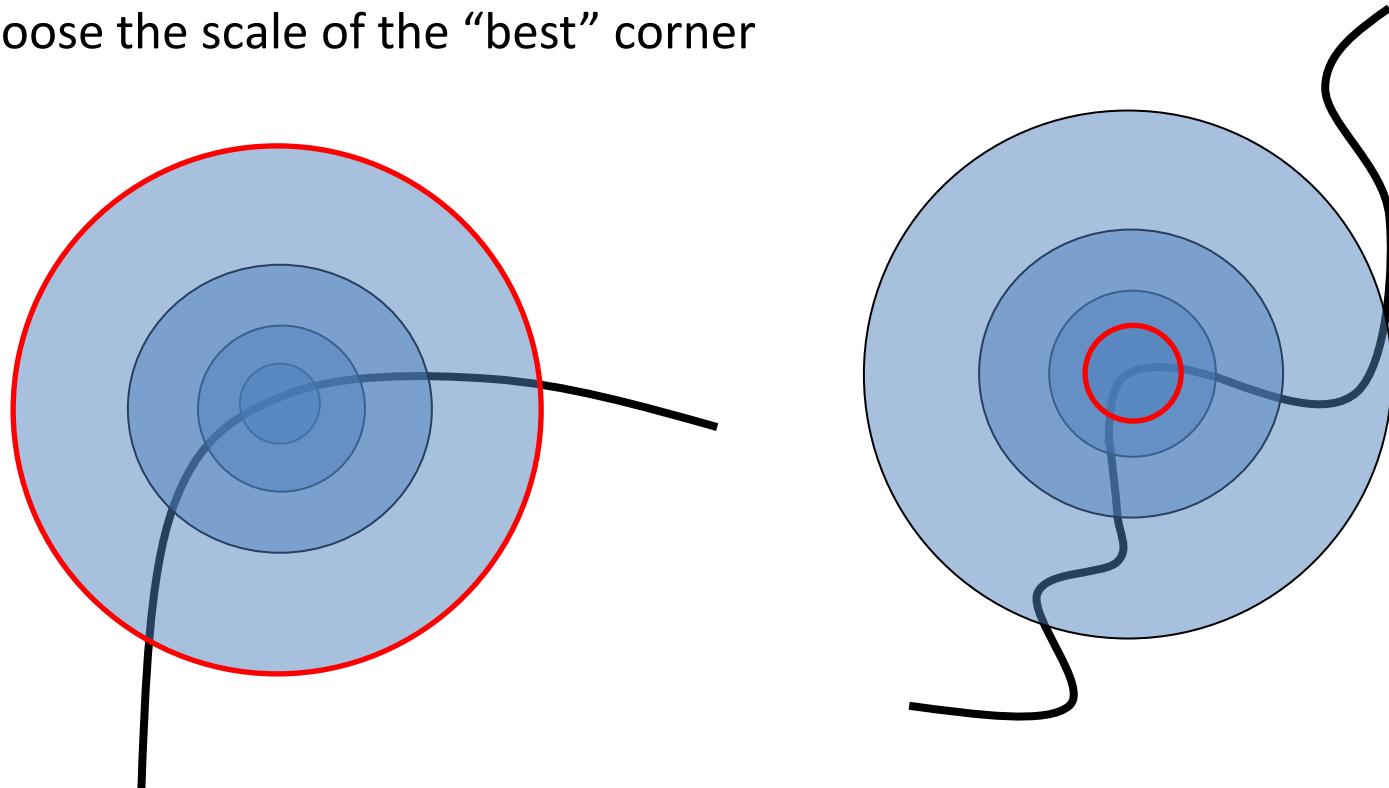
Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



Scale Invariant Detection

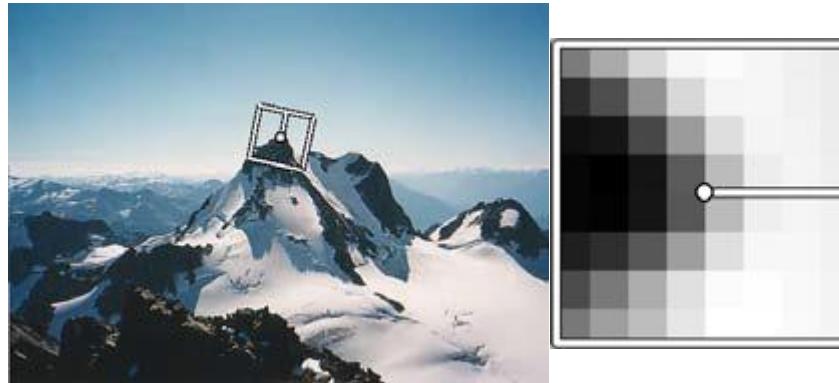
- The problem: how do we choose corresponding circles *independently* in each image?
- Choose the scale of the “best” corner



Rotational Invariance and Orientation Estimation

Most matching and object recognition algorithm need to deal with image rotation.

1. **Estimate dominant orientation** at each detected keypoint.
2. **Scaled and Oriented patch** around the detected point
(Feature Description)



Orientation Estimation

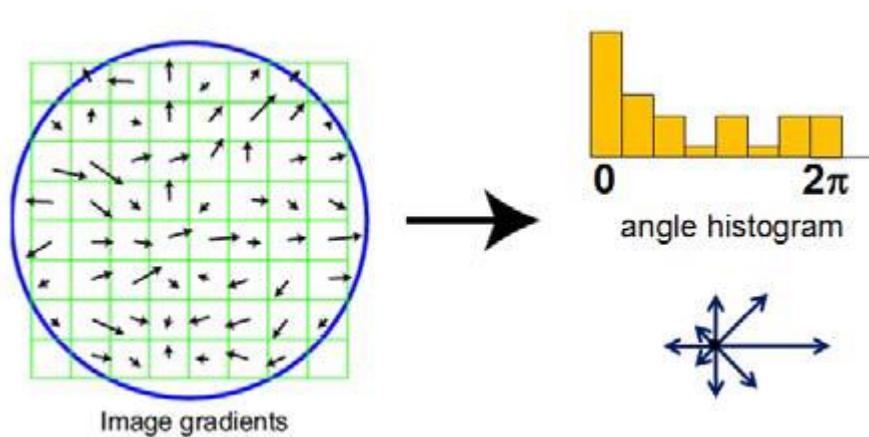
Simplest orientation estimate:

- **Average gradient** within a region around the keypoint

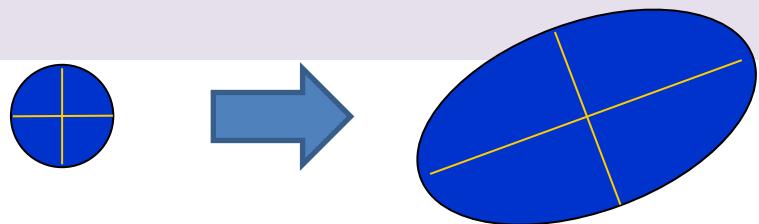
Problem: averaged signed gradients can be small and unreliable

More reliable:

- **Histogram of orientations** computed around a keypoint
- Lowe 2004: 36bin histogram of edge orientations weighted by
 - Gradient magnitude
 - Distance to the center
- **To determine orientation:** Find all peaks within 80% of global maxima. Three bin parabolic fit



Affine Invariance



1. Fit an ellipse to auto-correlation matrix or Hessian matrix using eigenvalue analysis
2. Use principal axis and ratio of this fit as affine coordinate frame

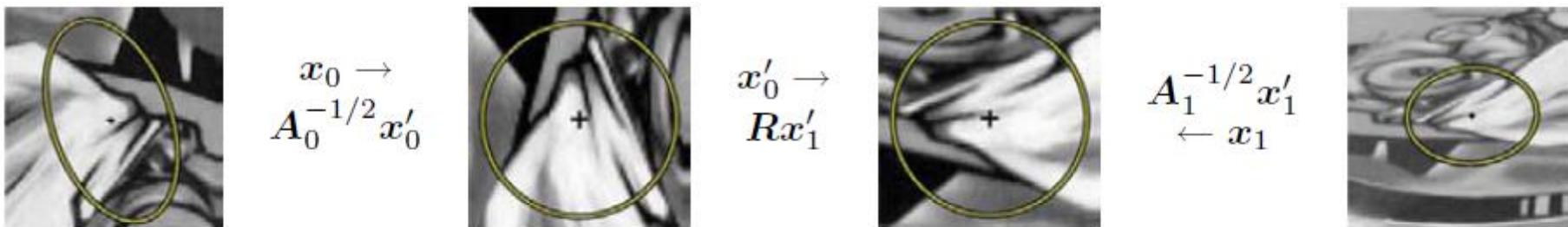
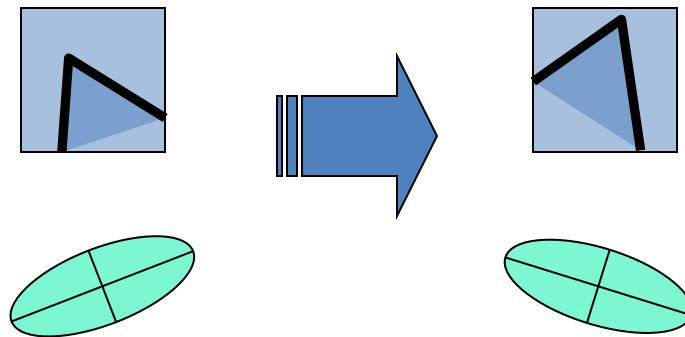


Figure 4.14 Affine normalization using the second moment matrices, as described by Mikolajczyk, Tuytelaars, Schmid *et al.* (2005) © 2005 Springer. After image coordinates are transformed using the matrices $A_0^{-1/2}$ and $A_1^{-1/2}$, they are related by a pure rotation R , which can be estimated using a dominant orientation technique.

Case Study: Harris Detector Some Properties

- Rotation invariance



Ellipse rotates but its shape (i.e. eigenvalues)
remains the same

Corner response R is invariant to image rotation

Case Study: Harris Detector Some Properties

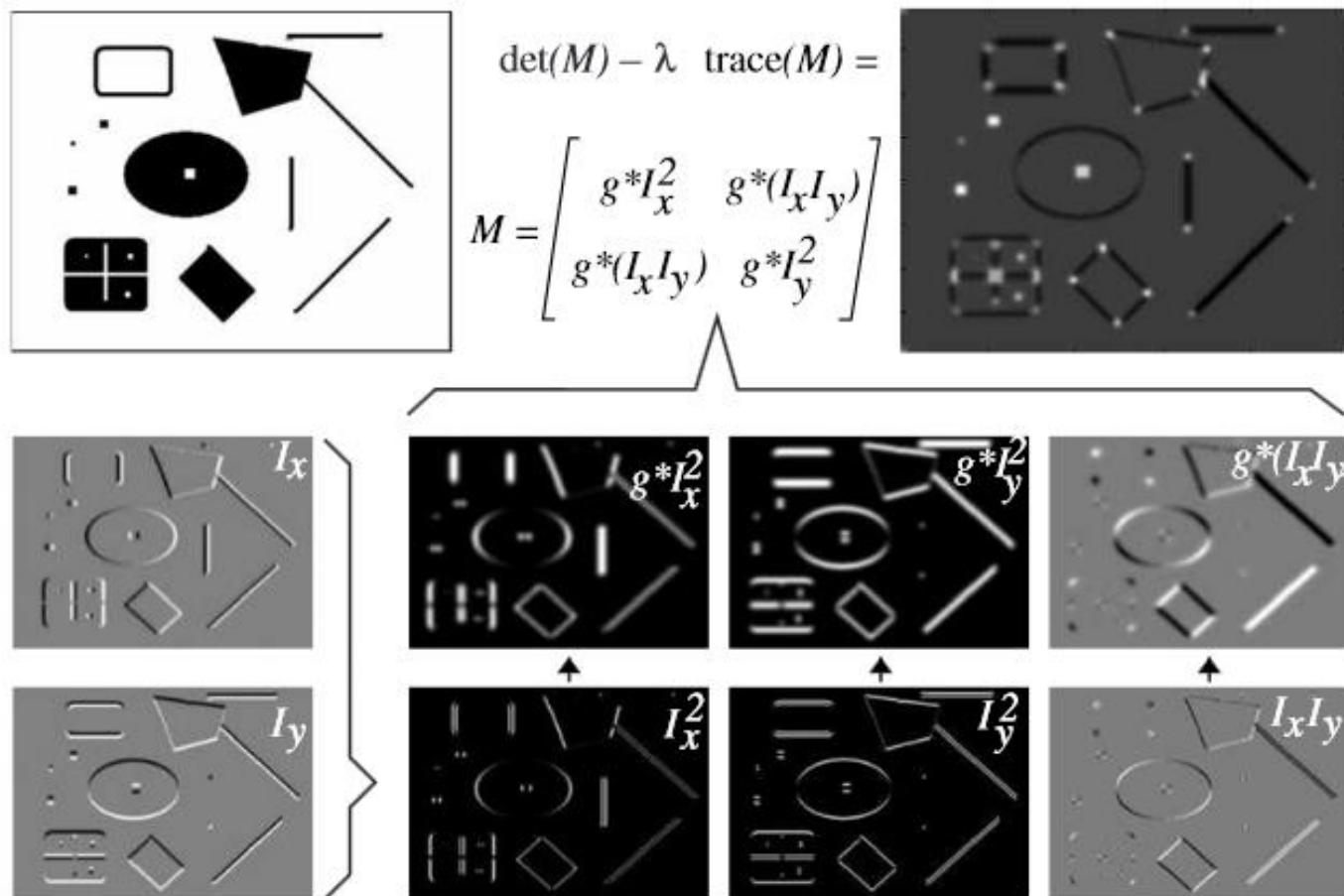
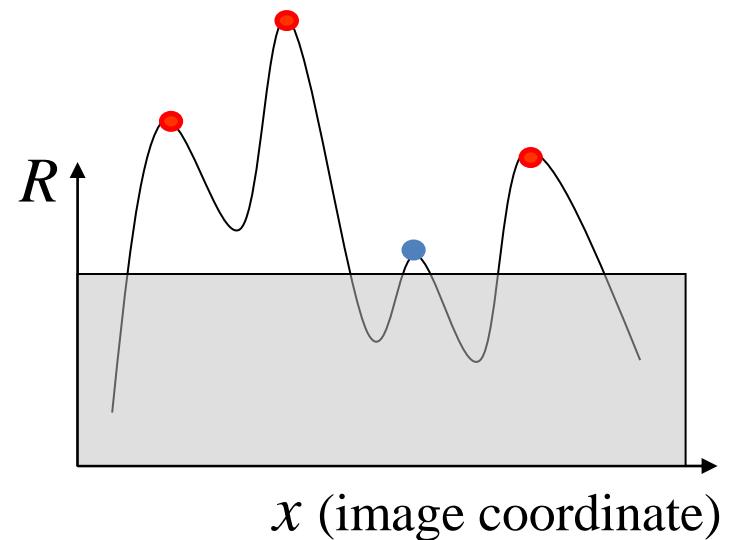
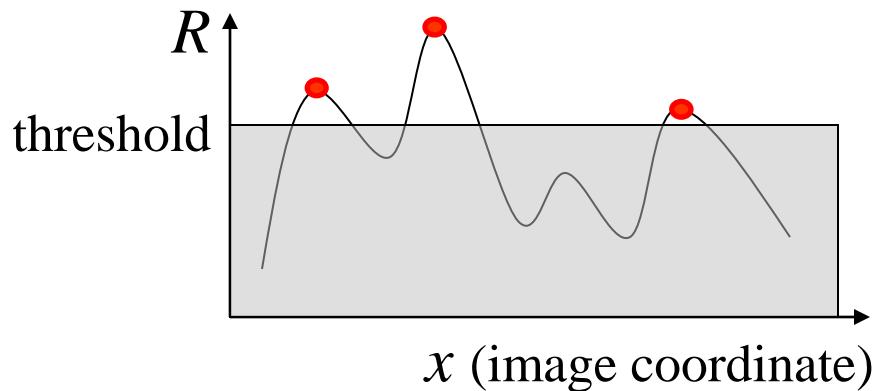


Fig. 3.1 Illustration of the components of the second moment matrix and Harris cornerness measure.

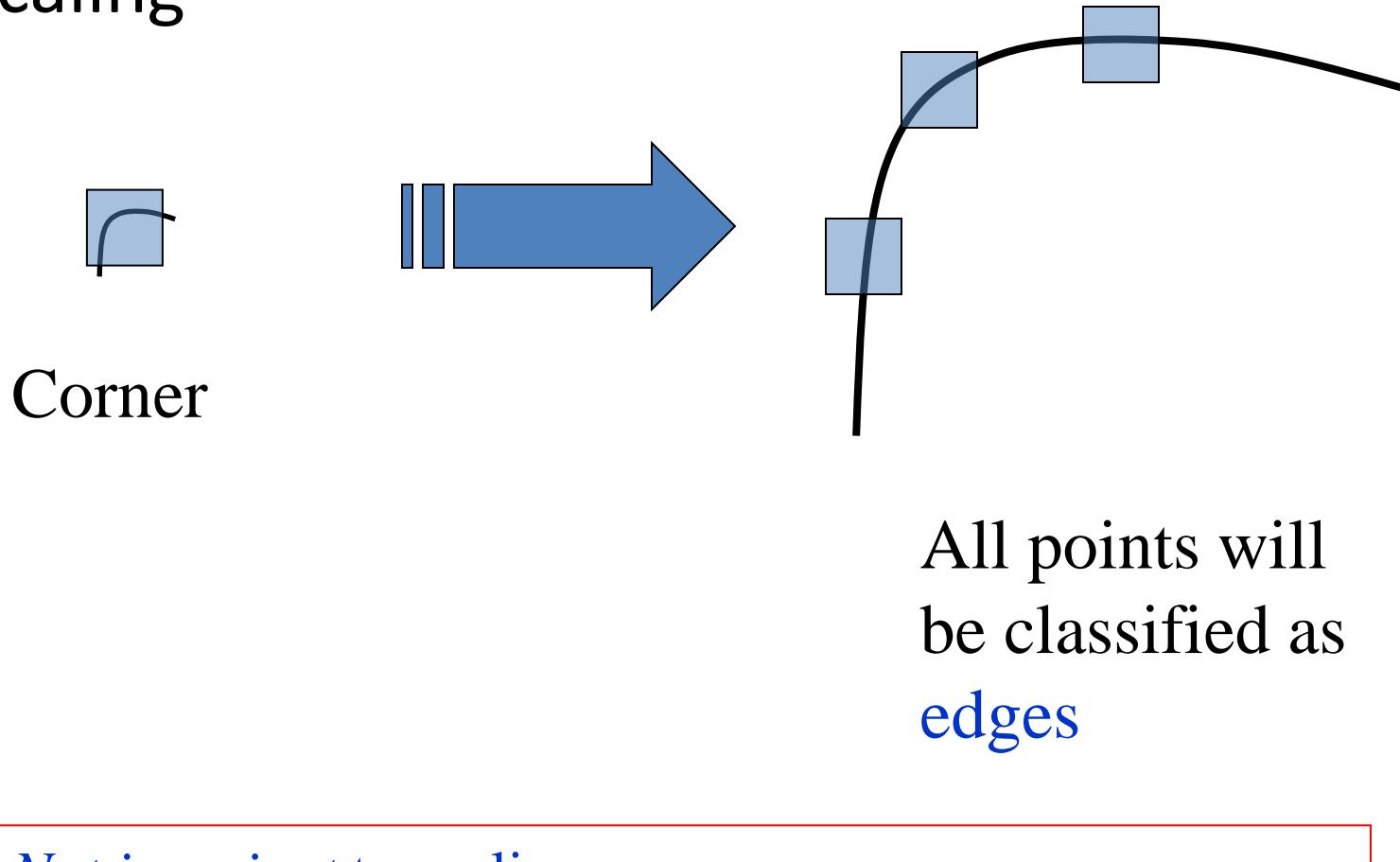
Case Study: Harris Detector Some Properties

- **Partial** invariance to *affine intensity* change
 - ✓ Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$
 - ✓ Intensity scale: $I \rightarrow a I$

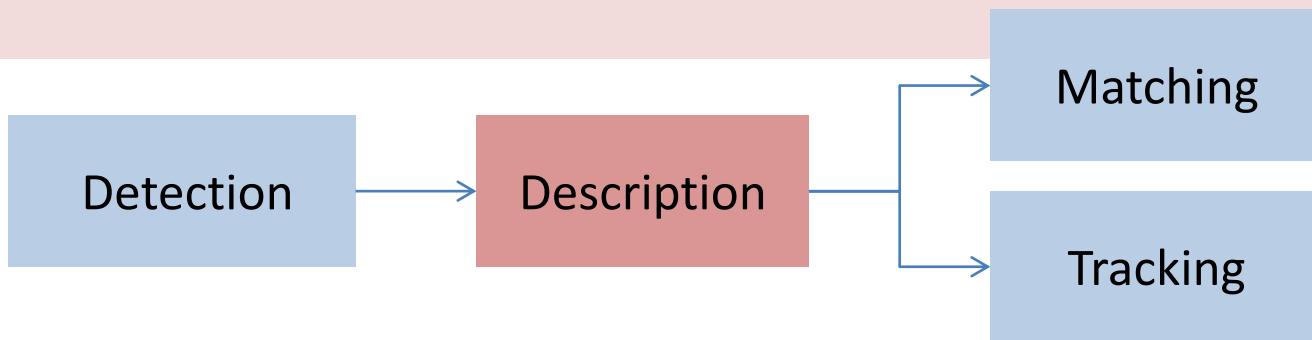


Case Study: Harris Detector Some Properties

- Scaling



Keypoint Detection and Matching Pipeline



1. **Feature detection (extraction)** : each image is searched for locations that are likely to match well in other images.
2. **Feature description** : each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors.
3. **Feature matching**: efficiently searches for likely matching candidates in other images.
4. **Feature tracking** : is an alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.

Relevant references:

- David Lowe's (2004) paper, which describes the development and refinement of his Scale Invariant Feature Transform (SIFT).
- Survey and evaluation papers covering both feature detection: (Schmid, Mohr, and Bauckhage 2000; Mikolajczyk, Tuytelaars, Schmid et al. 2005; Tuytelaars and Mikolajczyk 2007)
- Feature descriptors (Mikolajczyk and Schmid 2005). Shi and Tomasi (1994) and Triggs(2004) also provide nice reviews of feature detection techniques.

Multiscale Oriented PatcheS descriptor

Brown, Szeliski, Winder CVPR 2005

- Interest points
 - Multi-scale Harris corners
 - Orientation from blurred gradient
 - Geometrically invariant to rotation
- Descriptor vector
 - Bias/gain normalized sampling of local patch (8x8)
 - Photometrically invariant to affine changes in intensity
- [Brown, Szeliski, Winder, CVPR'2005]

Rotation invariance for feature descriptors

Find dominant orientation of the image patch

- This is given by \mathbf{x}_+ , the eigenvector of \mathbf{H} corresponding to λ_+
 - λ_+ is the *larger* eigenvalue
- Rotate the patch according to this angle

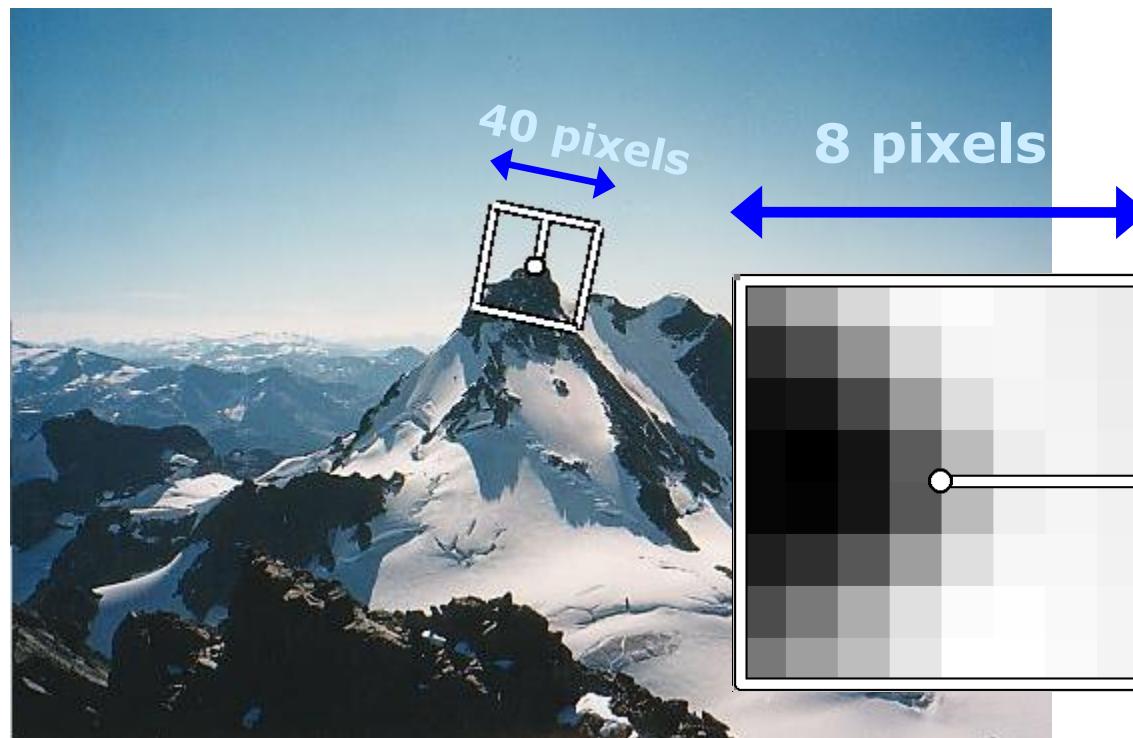


Figure by Matthew Brown

Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature

- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window : $I' = (I - \mu)/\sigma$



Adapted from slide by Matthew Brown

Detections at multiple scales

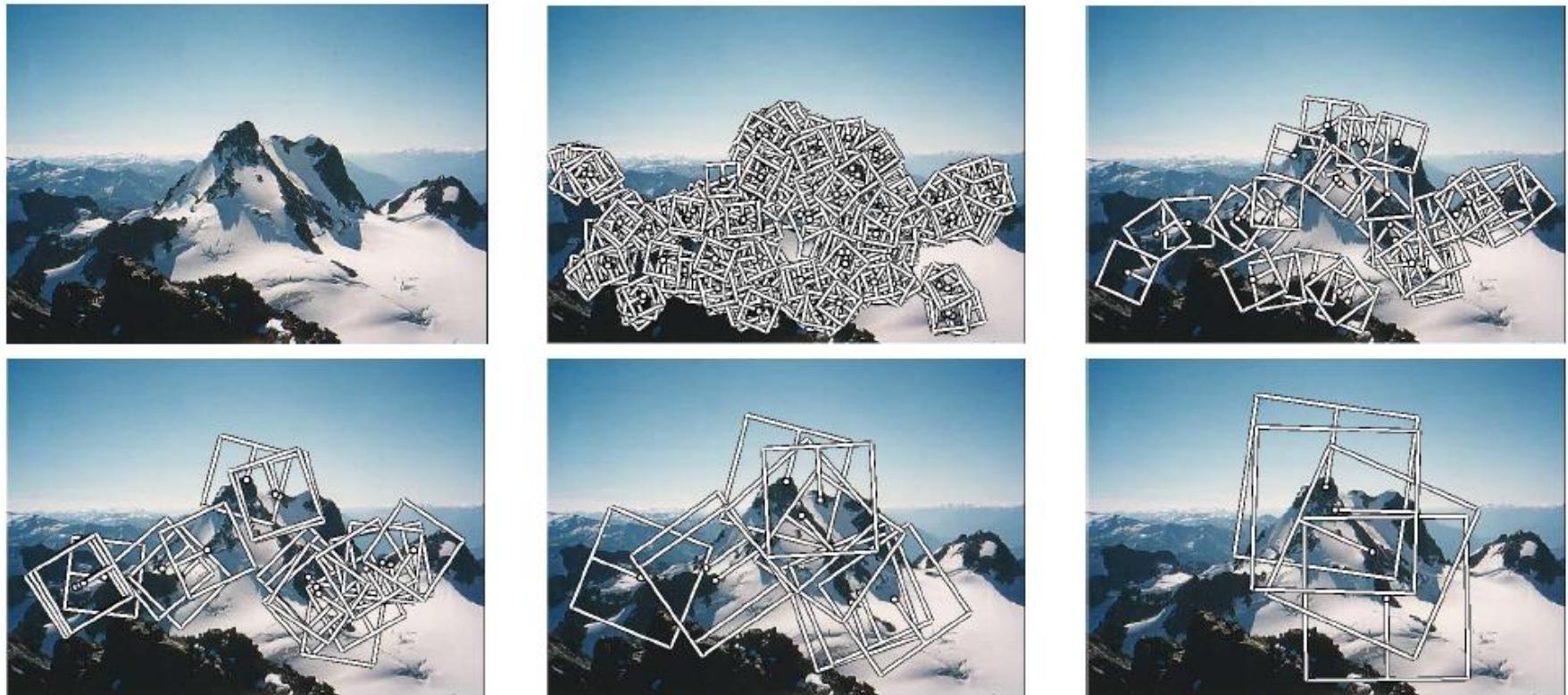


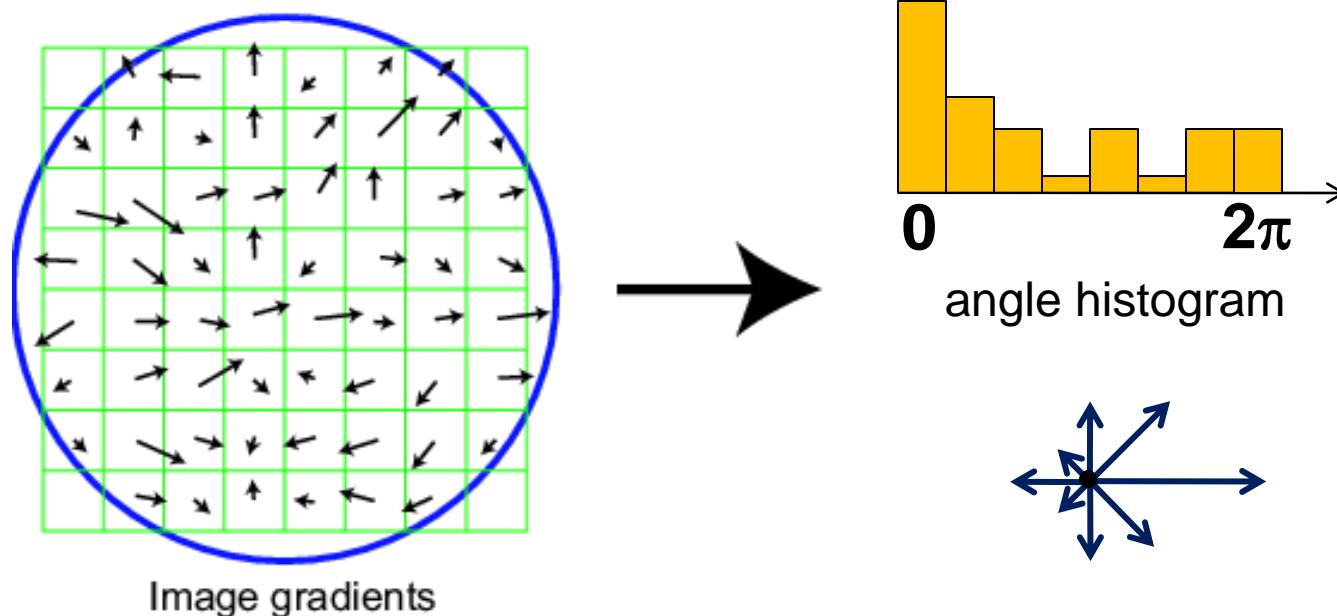
Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matier images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.

Scale Invariant Feature Transform

David Lowe ICCV 1999 (7K citations)

Basic idea:

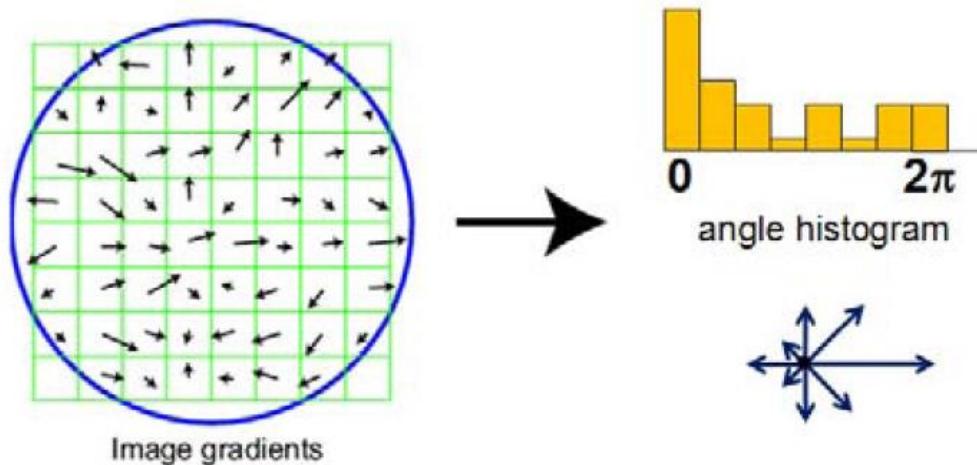
- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations



Adapted from slide by David Lowe

Rotation Invariance and orientation estimation

- Rotationally invariant descriptor but they have poor discriminability
- Dominant orientation
 - Average gradient within a region around the keypoint (larger window than detection window to make it more reliable)
 - Orientation histogram

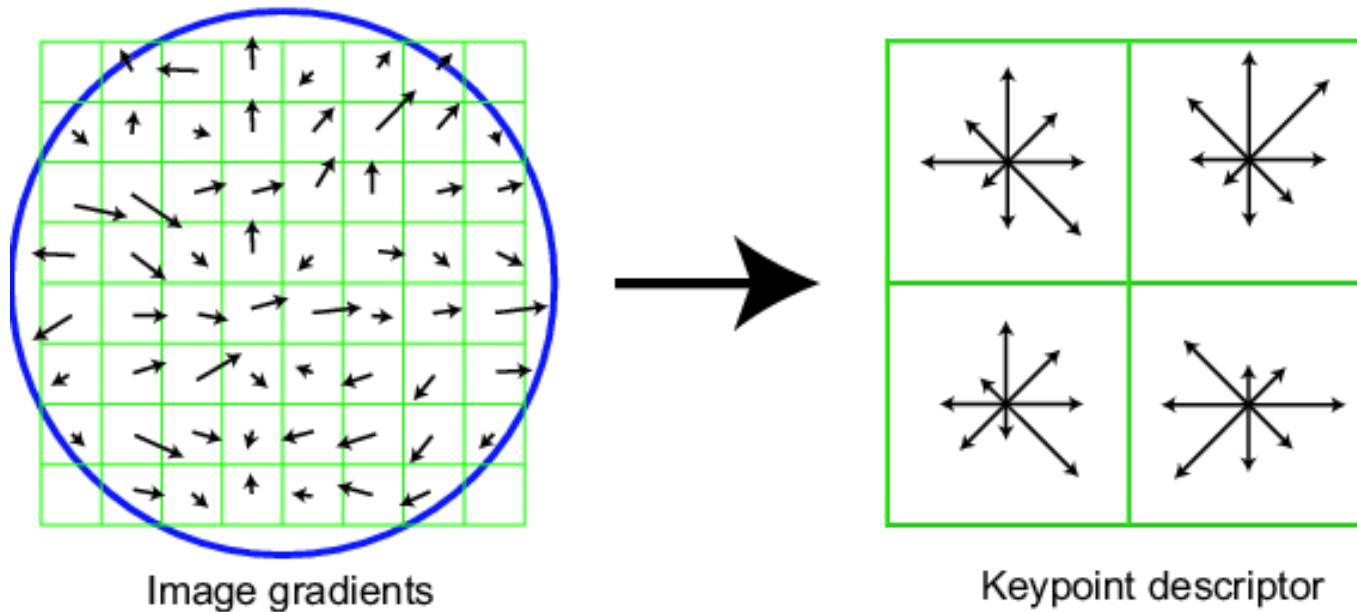


SIFT descriptor

(Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision* 60, no. 2 (2004): 91-110.) **68,406 citations**

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor



Adapted from slide by David Lowe

Properties of SIFT

Extraordinarily robust matching technique

- Can handle changes in viewpoint
 - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
 - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
 - https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html
 - <https://www.mathworks.com/help/vision/ref/detectsiftfeatures.html>



Descriptors – GLOH

C. S. Krystian Mikolajczyk *TPAMI* 2005

- Similar to SIFT: Divide the feature into log-polar bins instead of dividing the feature into square.
 - 17 log-polar location bins
 - 16 orientation bins
 - We get $17 \times 16 = 272$ dimensions.
- Apply PCA analysis, keep 128 components

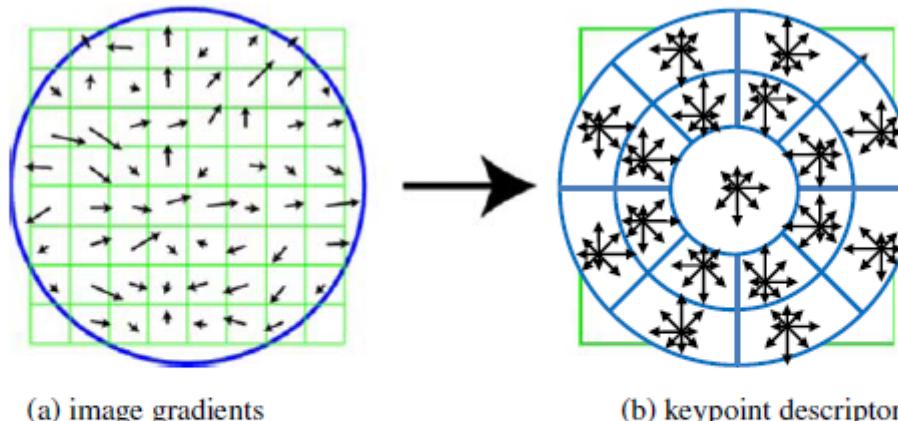


Figure 4.19 The gradient location-orientation histogram (GLOH) descriptor uses log-polar bins instead of square bins to compute orientation histograms (Mikolajczyk and Schmid 2005).



Keypoint Detection and Matching Pipeline



1. **Feature detection (extraction)** : each image is searched for locations that are likely to match well in other images.
2. **Feature description** : each region around detected keypoint locations is converted into a more compact and stable (invariant) descriptor that can be matched against other descriptors.
3. **Feature matching**: efficiently searches for likely matching candidates in other images.
4. **Feature tracking** : is an alternative to the third stage that only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.

Relevant references:

- David Lowe's (2004) paper, which describes the development and refinement of his Scale Invariant Feature Transform (SIFT).
- Survey and evaluation papers covering both feature detection: (Schmid, Mohr, and Bauckhage 2000; Mikolajczyk, Tuytelaars, Schmid et al. 2005; Tuytelaars and Mikolajczyk 2007)
- Feature descriptors (Mikolajczyk and Schmid 2005). Shi and Tomasi (1994) and Triggs(2004) also provide nice reviews of feature detection techniques.

Feature matching

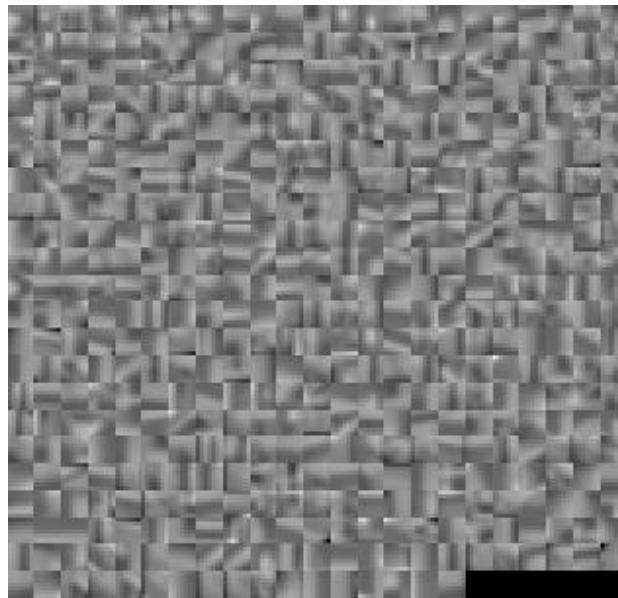
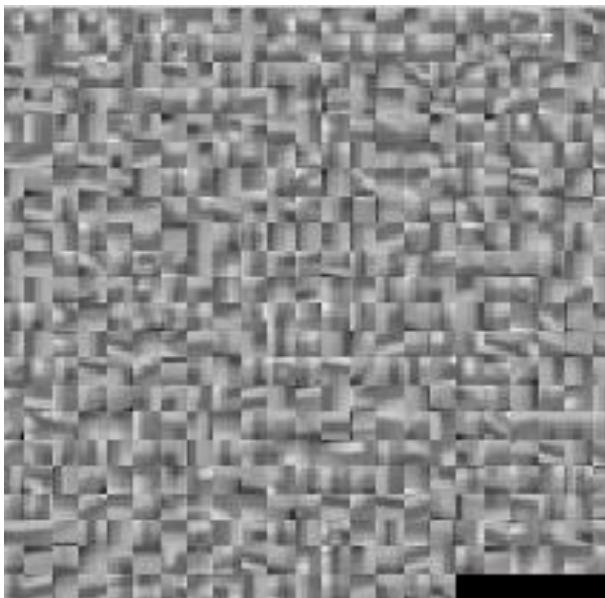
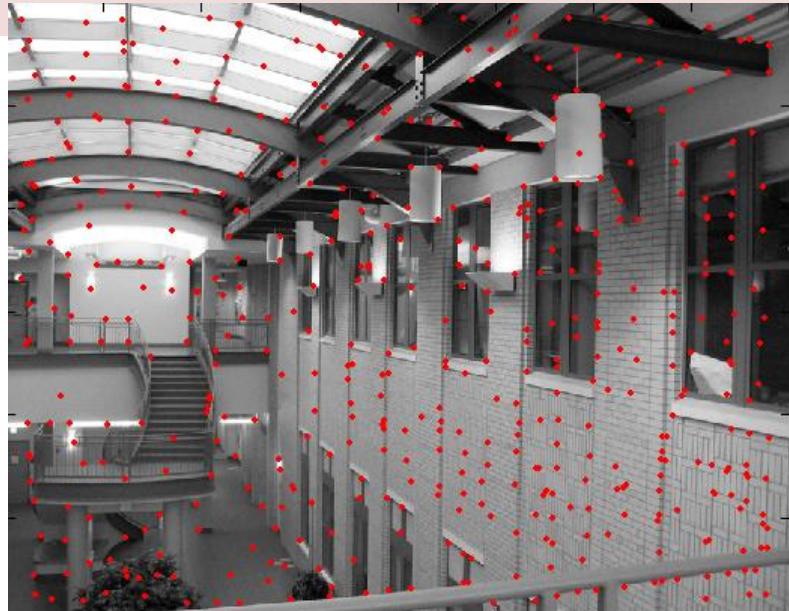
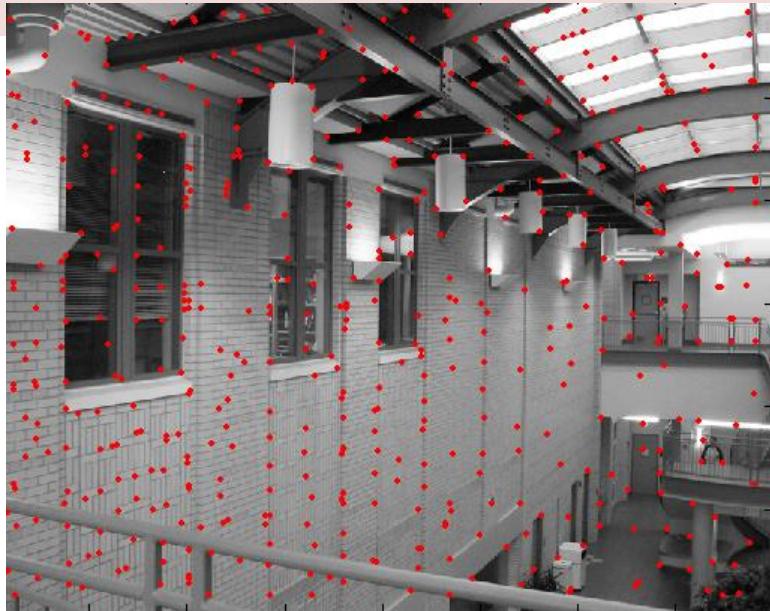
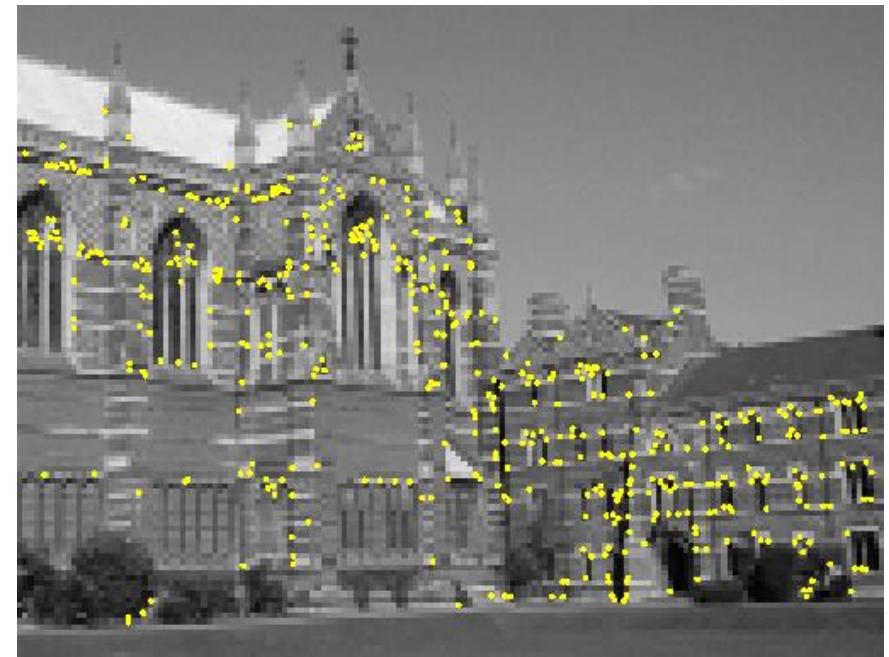
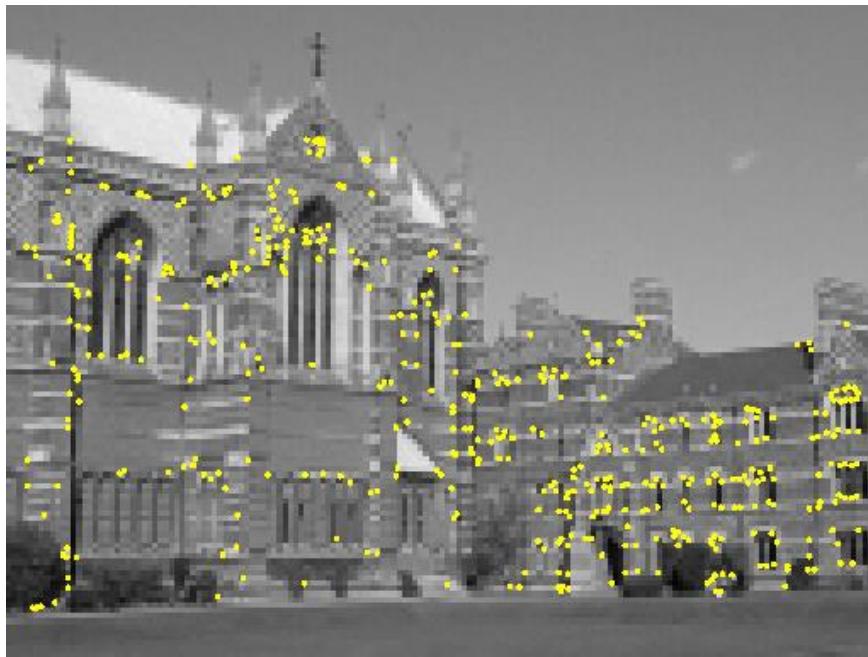


Illustration – Matching



Interest points extracted with Harris detector (~ 500 points)

Illustration - Matching

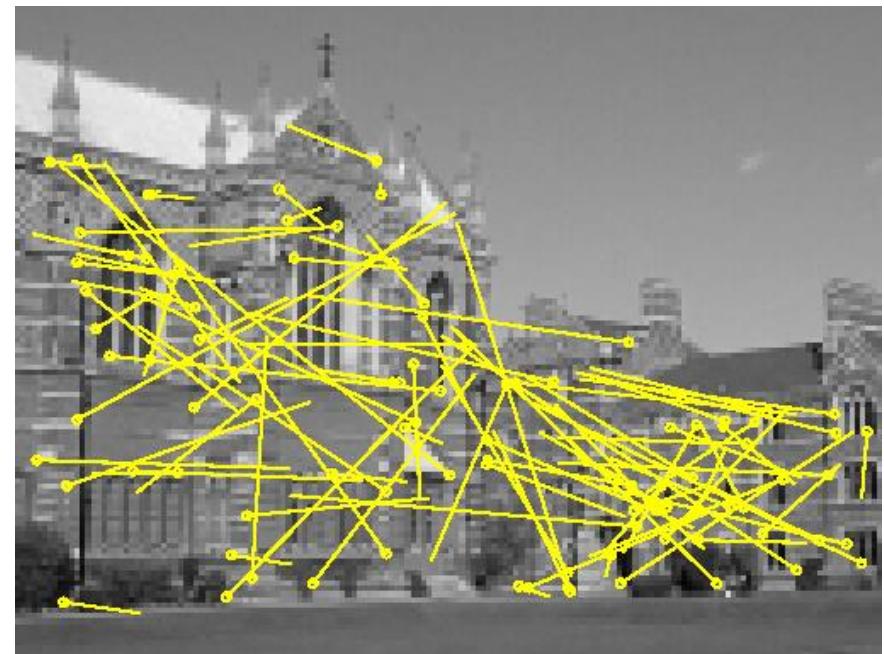


Interest points matched based on cross-correlation (188 pairs)

Illustration - Matching



99 inliers



89 outliers

Feature Matching

First Step: Extract features and their descriptors from two or more images

Second step: Establish some preliminary feature matches between these images.

Divide this problem into two separate components:

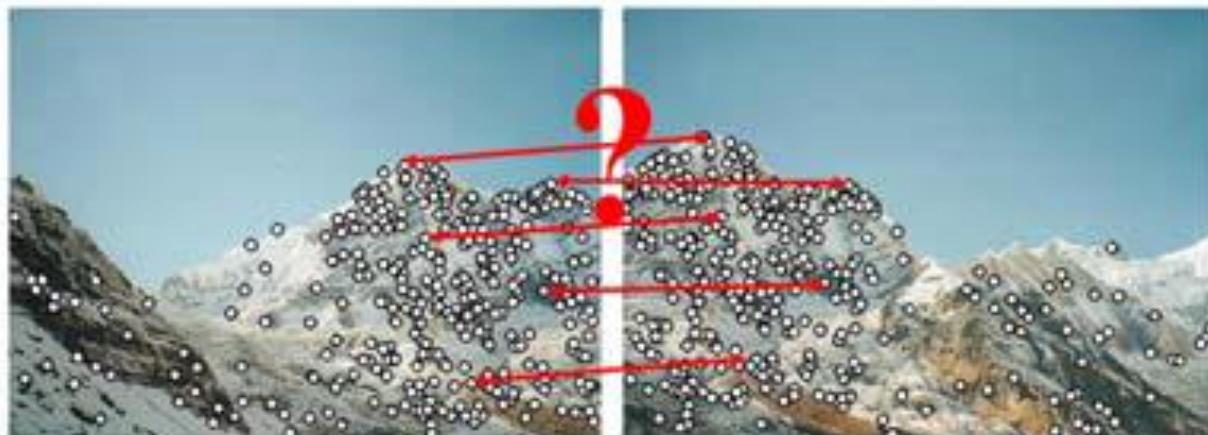
- 1. Matching strategy:** which correspondences are passed on to the next stage for further processing.
- 2. Efficient data structures and algorithms:** to perform this matching as quickly as possible.

Matching Strategy and Error Rates

Determining which feature matches are reasonable to process further depends on the context:

Case 1: Image stitching or tracking objects in a video

- Two images that **overlap to a fair amount**
- **Most features** in one image are **likely to match** the other image,
- **Some may not match** because they are **occluded** or their **appearance has changed** too much.



Matching Strategy and Error Rates

Case 2: Recognize how many known objects appear in a cluttered

- Most of the features may not match.
- Large number of potentially matching objects must be searched
 - requires more efficient strategies

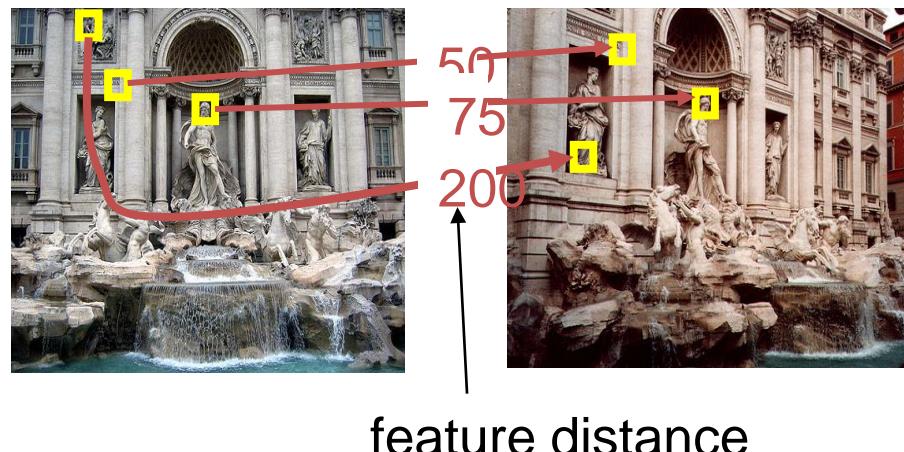


Feature matching

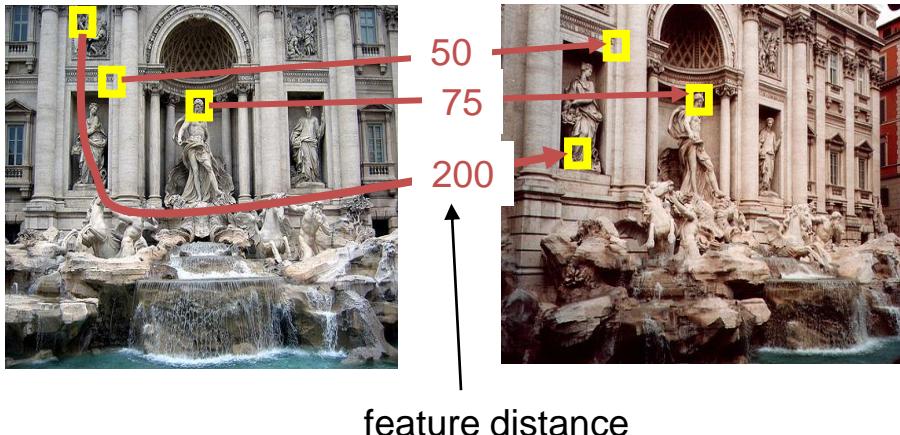
Given a feature in I_1 , how to find the best match in I_2 ?

1. Define distance function that compares two descriptors
2. Compute distances between detections in

Assumption: Feature descriptors have been designed so that Euclidean(vector magnitude) distances in feature space can be directly used for ranking potential matches.



Match Measures for matching Descriptors / Patches



$$D = \sum_{p_1, p_2} (I_1(p_1) - I_2(p_2))^2$$

$$D = \sum_{p_1, p_2} |I_1(p_1) - I_2(p_2)|$$

$$D = 1 - \sum_{p_1, p_2} \frac{(I_1(p_1) - \bar{I}_1)(I_2(p_2) - \bar{I}_2)}{\sigma_{Ip1}\sigma_{Ip2}}$$

L2 Norm

- Sensitive to outliers
- Sensitive to brightness and contrast variations

L1 Norm

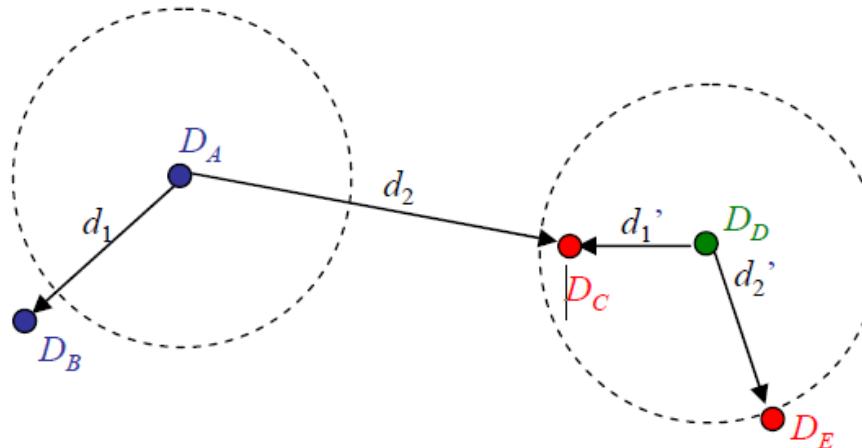
- Less sensitive to outliers
- Sensitive to brightness and contrast variations

Inverse Normalized Correlation

- Sensitive to outliers
- Invariant to local offset and scale changes in intensity/color

Matching Strategy

Sample Comparison



Fixed distance threshold (dashed circles):

- D_A fails to match D_B
- D_D incorrectly matches D_C and D_E .

Nearest neighbor :

- D_A correctly matches D_B
- D_D incorrectly matches D_C

Nearest neighbor distance ratio (NNDR) matching:

- Small NNDR d_1/d_2 correctly matches D_A with D_B ,
- Large NNDR d_01/d_02 correctly rejects matches for D_D .

Some Matching Strategies

1. Distance thresholding
2. Nearest match
3. Nearest neighbor distance ratio

Matching Strategy 1: Distance Thresholding

1. Set a threshold (maximum distance)
 2. Return all matches from other images within this threshold.
- **Threshold too high** results -> too many **false positives**, i.e., incorrect matches
 - **Threshold too low** results -> too many **false negatives**, i.e., too many correct matches being missed.

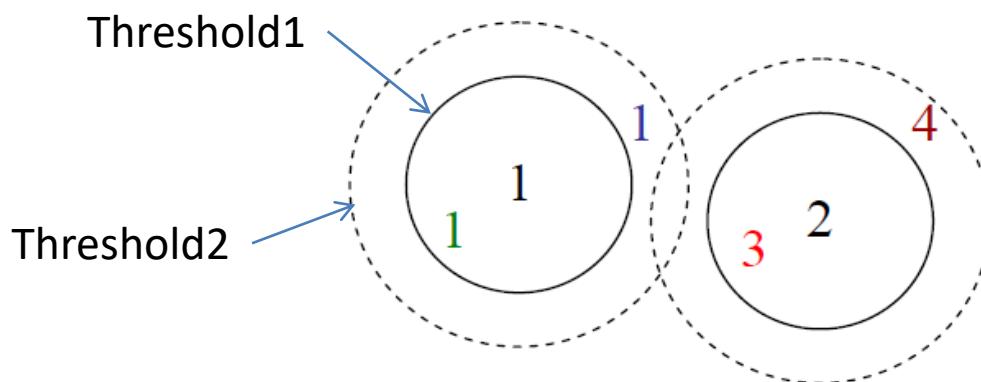
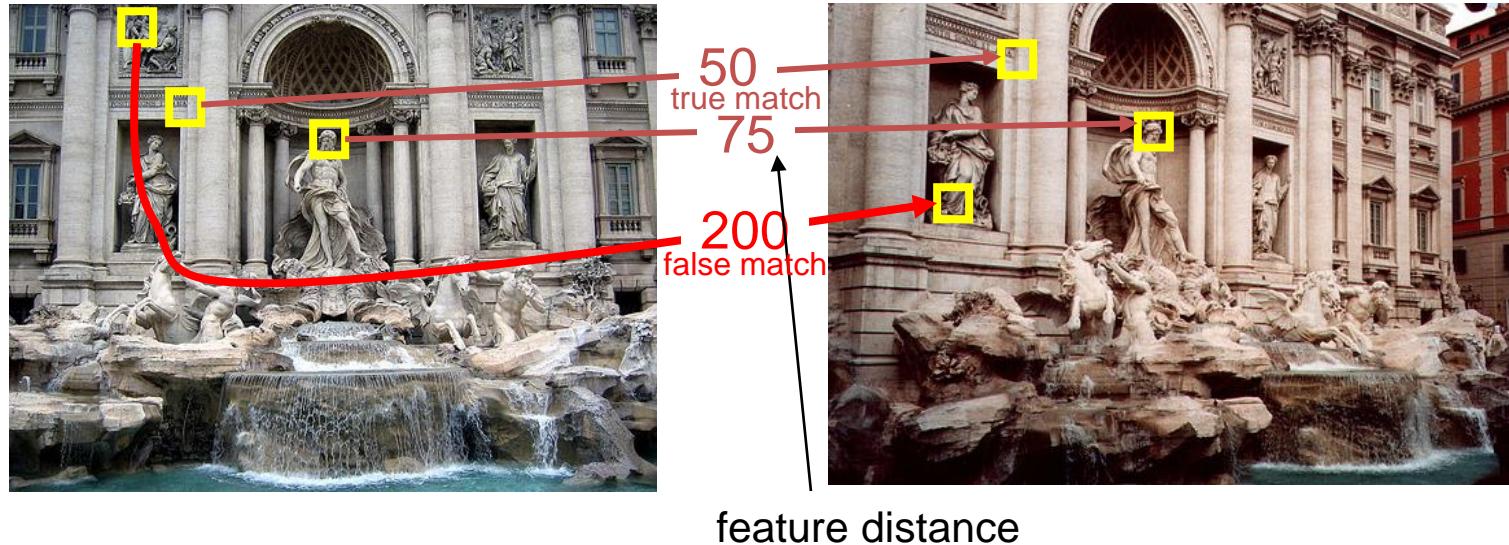


Figure 4.22 False positives and negatives: The black digits 1 and 2 are features being matched against a database of features in other images. At the current threshold setting (the solid circles), the green 1 is a *true positive* (good match), the blue 1 is a *false negative* (failure to match), and the red 3 is a *false positive* (incorrect match). If we set the threshold higher (the dashed circles), the blue 1 becomes a true positive but the brown 4 becomes an additional false positive.

Evaluating the Results: True/false positives



The distance threshold affects performance

True positives = # of detected matches that are correct

- Suppose we want to maximize these—how to choose threshold?

False positives = # of detected matches that are incorrect

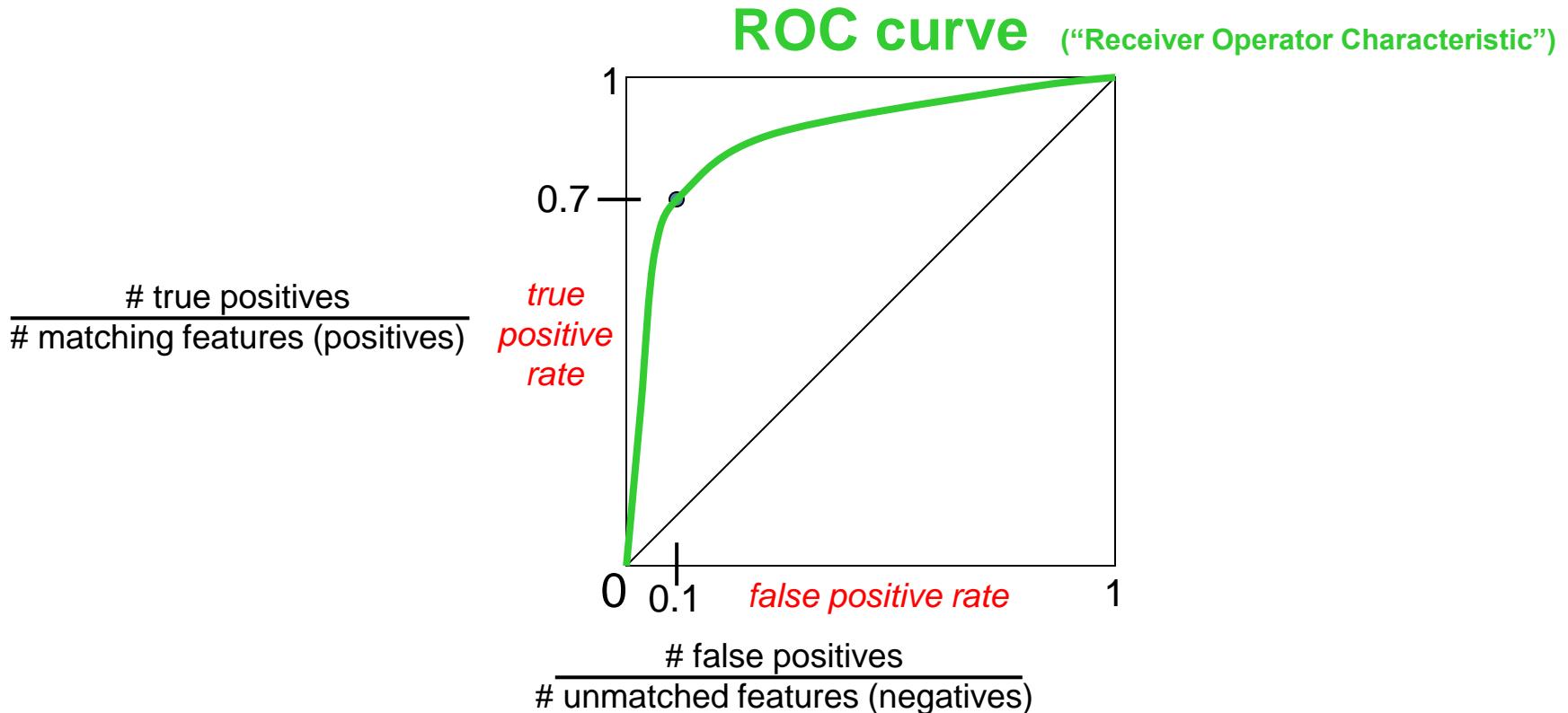
- Suppose we want to minimize these—how to choose threshold?

Evaluating the Results

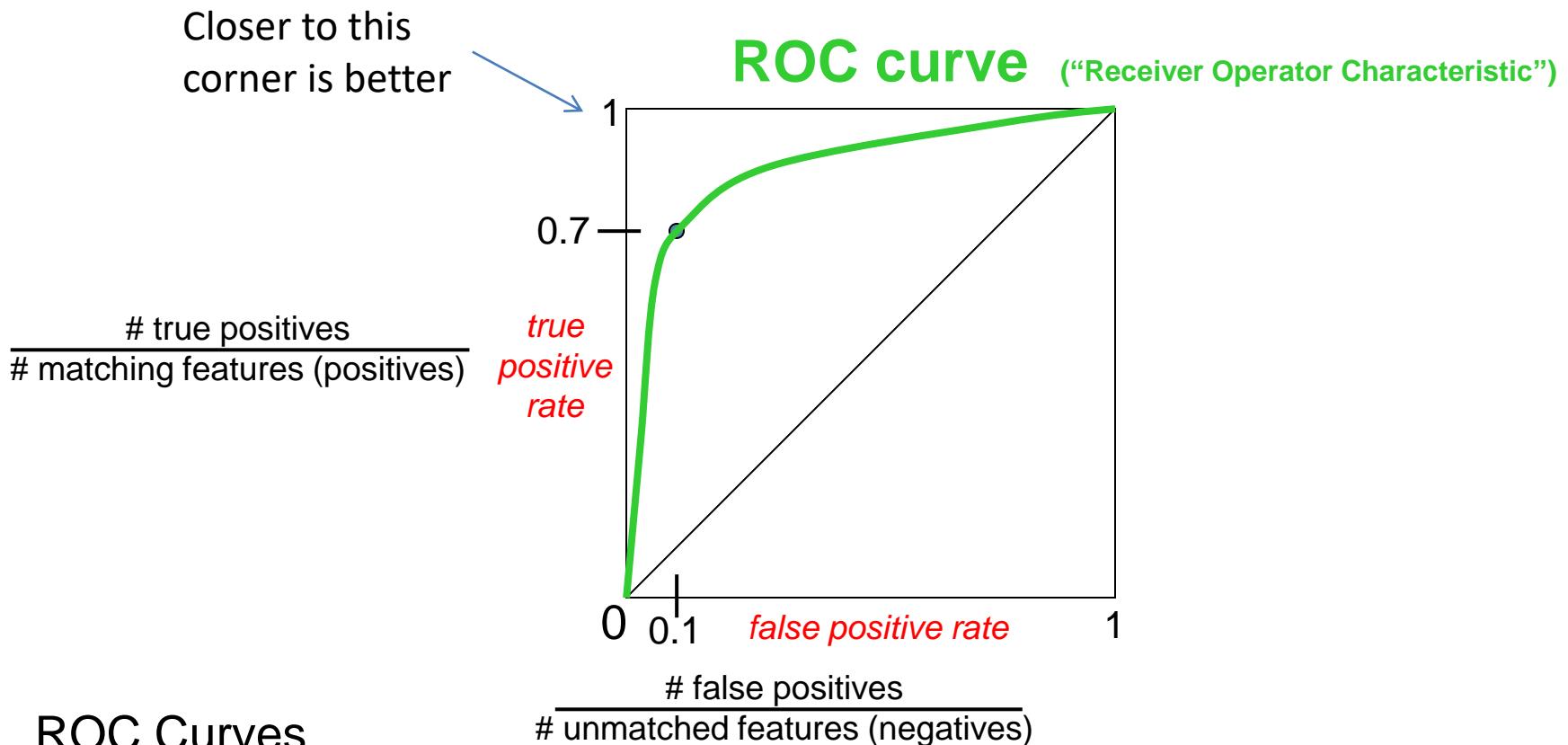
- **TP:** **true positives**, number of correct matches;
 - **FN:** **false negatives**, matches that were not correctly detected;
 - **FP:** **false positives**, proposed matches that are incorrect;
 - **TN:** **true negatives**, non-matches that were correctly rejected.
-
- **True positive rate :** $TPR = TP / (TP + FN) = TP / P$
 - **False positive rate:** $FPR = FP / (FP + TN) = FP / N$
 - **Positive predictive value:** $PPV = TP / (TP + FP) = TP / P'$
 - **Accuracy:** $ACC = (TP + TN) / (P + N)$

Evaluating the results

How can we measure the performance of a feature matcher?



Evaluating the results



- Generated by counting # current/incorrect matches, for different thresholds
- Want to maximize area under the curve (AUC)
- Useful for comparing different feature matching methods
- For more info: http://en.wikipedia.org/wiki/Receiver_operating_characteristic

Matching Strategy 2: Nearest Neighbor Match

The problem with using a fixed threshold :

Useful range of thresholds

- Difficult to set
- Can vary a lot as we move to different parts of the feature space
(Lowe 2004; Mikolajczyk and Schmid 2005).

Better strategy: Simply match the nearest neighbor in feature space.

Problem: Some features may have no matches e.g.,

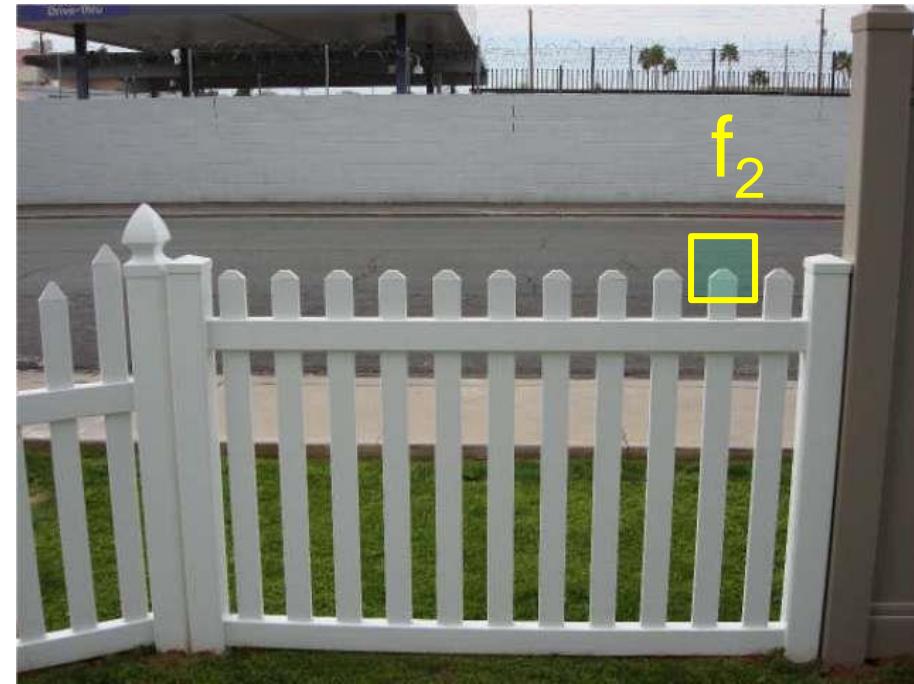
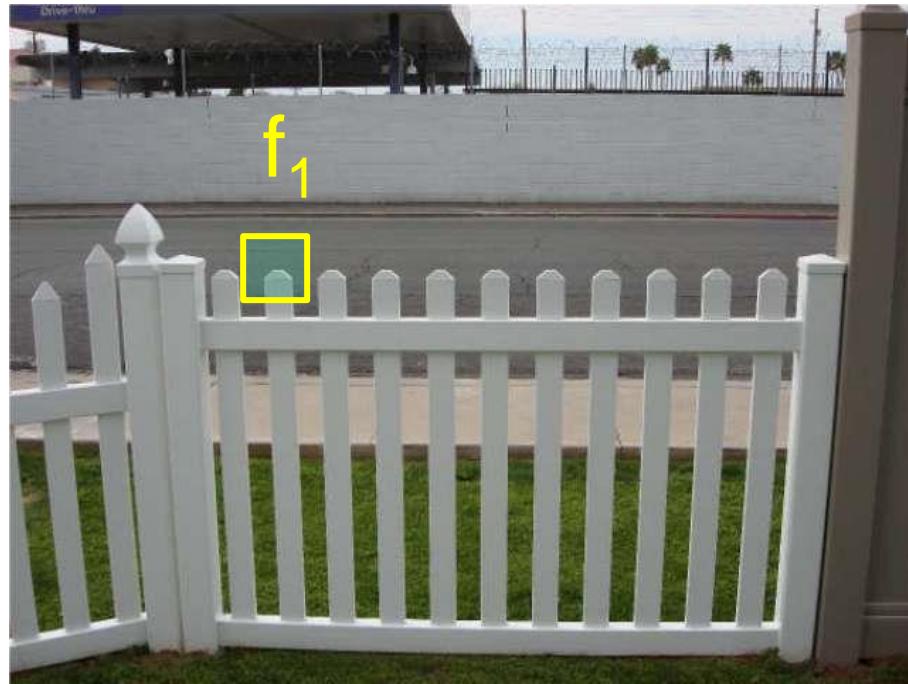
- part of background clutter in object recognition
- occluded in the other image

Solution: Threshold is still used to reduce the number of false positives.

Matching Strategy 3: Nearest Neighbor Distance Ratio

How to define the difference between two features f_1 , f_2 ?

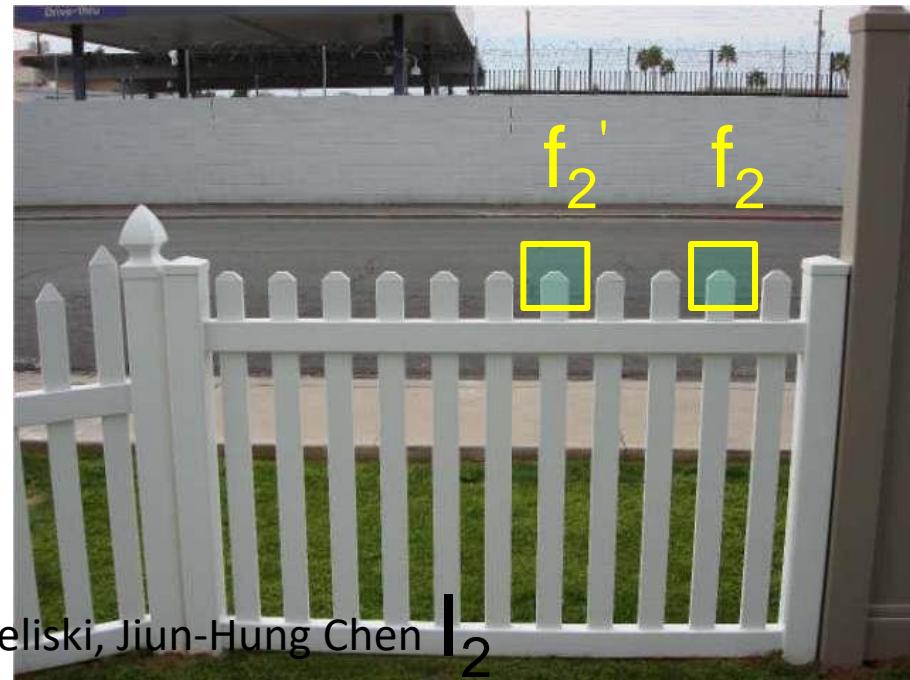
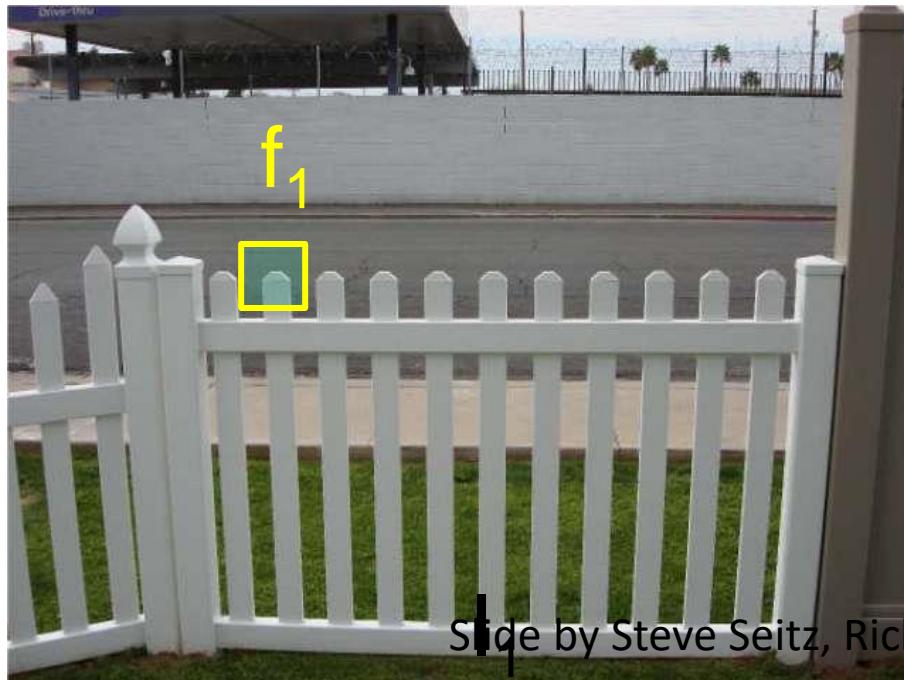
- Simple approach is $\text{SSD}(f_1, f_2)$
 - sum of square differences between entries of the two descriptors
 - can give good scores to very ambiguous (bad) matches



Matching Strategy 3: Nearest Neighbor Distance Ratio

How to define the difference between two features f_1 , f_2 ?

- Better approach: ratio distance = $\text{SSD}(f_1, f_2) / \text{SSD}(f_1, f_2')$
 - f_2 is best SSD match to f_1 in I_2
 - f_2' is 2nd best SSD match to f_1 in I_2
 - gives small values for ambiguous matches



More on feature detection/description

Publications

Region detectors

- *Harris-Affine & Hessian Affine*: [K. Mikolajczyk](#) and [C. Schmid](#), Scale and Affine invariant interest point detectors. In IJCV 1(60):63-86, 2004. [PDF](#)
- *MSER*: [J. Matas](#), [O. Chum](#), [M. Urban](#), and [T. Pajdla](#), Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. [PDF](#)
- *IBR & EBR*: [T. Tuytelaars](#) and [L. Van Gool](#), Matching widely separated views based on affine invariant regions. In IJCV 1 (59):61-85, 2004. [PDF](#)
- *Salient regions*: [T. Kadir](#), [A. Zisserman](#), and [M. Brady](#), An affine invariant salient region detector. In ECCV p. 404-416, 2004. [PDF](#)

Region descriptors

- *SIFT*: [D. Lowe](#), Distinctive image features from scale invariant keypoints. In IJCV 2(60):91-110, 2004. [PDF](#)

Performance evaluation

- [K. Mikolajczyk](#), [T. Tuytelaars](#), [C. Schmid](#), [A. Zisserman](#), [J. Matas](#), [F. Schaffalitzky](#), [T. Kadir](#) and [L. Van Gool](#), A comparison of affine region detectors. Technical Report, accepted to IJCV. [PDF](#)
- [K. Mikolajczyk](#), [C. Schmid](#), A performance evaluation of local descriptors. Technical Report, accepted to PAMI. [PDF](#)

Affine Covariant Features

K. Mikolajczyk maintains a web page for affine covariant features:

<http://www.robots.ox.ac.uk/~vgg/research/affine/>

Region Detectors:

- *Harris-Affine & Hessian Affine:* [K. Mikolajczyk](#) and [C. Schmid](#), Scale and Affine invariant interest point detectors. In IJC V 60(1):63-86, 2004. [PDF](#)
- *MSER:* [J. Matas](#), [O. Chum](#), [M. Urban](#), and [T. Pajdla](#), Robust wide baseline stereo from maximally stable extremal regions. In BMVC p. 384-393, 2002. [PDF](#)
- *IBR & EBR:* [T.Tuytelaars](#) and [L. Van Gool](#), Matching widely separated views based on affine invariant regions . In IJCV 59(1):61-85, 2004. [PDF](#)
- *Salient regions:* [T. Kadir](#), [A. Zisserman](#), and [M. Brady](#), An affine invariant salient region detector. In ECCV p. 404-416, 2004. [PDF](#)
- *All Detectors - Survey:* [T. Tuytelaars](#) and [K. Mikolajczyk](#), Local Invariant Feature Detectors - Survey. In CVG, 3(1):1-110, 2008. [PDF](#)

Region Descriptors

- *SIFT:* [D. Lowe](#), Distinctive image features from scale invariant keypoints. In IJCV 60(2):91-110, 2004. [PDF](#)

Performance Evaluation

- [K. Mikolajczyk](#), [T. Tuytelaars](#), [C. Schmid](#), [A. Zisserman](#), [J. Matas](#), [F. Schaffalitzky](#), [T. Kadir](#) and [L. Van Gool](#), A comparison of affine region detectors. In IJCV 65(1/2):43-72, 2005. [PDF](#)
- [K. Mikolajczyk](#), [C. Schmid](#), A performance evaluation of local descriptors. In PAMI 27(10):1615-1630 . [PDF](#)

The site also contains link for

- Software and
- Test data

RECENT WORKS

Image Matching: Local Features & Beyond



Links

- [2020 workshop livestream](#)
- [2021 workshop livestream](#)
- [2019-2021 Image Matching Benchmark](#) (used in previous editions of the challenge)
- [2020 Image Matching Challenge](#)
- [2021 Image Matching Challenge](#)
- [2022 Image Matching Challenge](#)

• Stay tuned for news about the 2023 Image Matching Challenge!

Contact

Please reach us with any questions at image-matching@googlegroups.com.

Image Matching: Local Features & Beyond

CVPR 2023 Workshop

How do Hand-Crafted Features Compare with Learned Features?

Comparative Evaluation of Hand-Crafted and Learned Local Features

Johannes L. Schönberger¹ Hans Hardmeier¹ Torsten Sattler¹ Marc Pollefeys^{1,2}

¹ Department of Computer Science, ETH Zürich ² Microsoft Corp.

{jsch, harhans, sattlert, pomarc}@inf.ethz.ch

CVPR 2017

- “Hand-crafted features still perform on par or better than recent learned features for image-based reconstruction.
- The current generation of learned descriptors shows a high variance across different datasets and applications.
- The next generation of learned descriptors needs more training data.”

How do Hand-Crafted Features Compare with Learned Features?

Image Matching across Wide Baselines: From Paper to Practice

Yuhe Jin¹ Dmytro Mishkin² Anastasia Mishchuk³
Jiří Matas² Pascal Fua³ Kwang Moo Yi¹ Eduard Trulls⁴

¹University of Victoria

²Czech Technical University in Prague

³École Polytechnique Fédérale de Lausanne

⁴Google Research

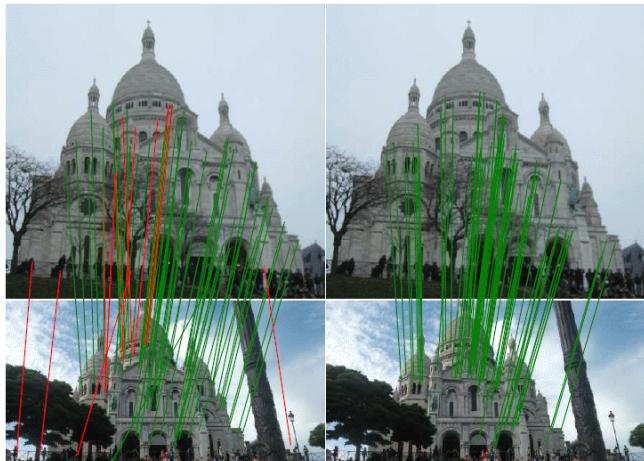


Figure 1. Every paper claims to outperform the state of the art. Is this possible, or an artifact of insufficient validation? On the left, we show stereo matches obtained with **D2-Net** (2019) [33], a state-of-the-art local feature, using OpenCV RANSAC with its default settings. On the right, we show **SIFT** (1999) [48] with a carefully tuned MAGSAC [29] – notice how the latter performs much better. We fill this gap with a new, modular benchmark for sparse image matching, with dozens of built-in methods.

Contributions

- Dataset with 30k images with depth maps and ground truth poses
- A modular pipeline incorporating dozens of methods for feature extraction and matching, and pose estimation
- Two downstream tasks – stereo and multi-view reconstruction – evaluated with downstream and intermediate metrics
- A thorough study of dozens of methods and techniques, hand-crafted and learned, and their combination, along with a procedure for hyper-parameter selection

Viewpoint and Illumination Variations Dataset

HPatches: A benchmark and evaluation of handcrafted and learned local descriptors

Vassileios Balntas*
Imperial College London
v.balntas@imperial.ac.uk

Karel Lenc*
University of Oxford
karel@robots.ox.ac.uk

Andrea Vedaldi
University of Oxford
vedaldi@robots.ox.ac.uk

Krystian Mikolajczyk
Imperial College London
k.mikolajczyk@imperial.ac.uk

<https://github.com/hpatches/hpatches-dataset>



Figure 1. Examples of image sequences; note the diversity of scenes and nuisance factors, including viewpoint, illumination, focus, reflections and other changes.

- Reproducible, patch-based: Descriptor evaluation should be done on patches to eliminate the detector related factors.
- Diverse: Representative of many different scenes and image capturing conditions.
- Real: Real data more challenging than a synthesized one due to nuisance factors that cannot be modelled in image transformations.
- Large: For accurate and stable evaluation; to provide substantial training sets for learning based descriptors.
- Multitask: Use cases, from matching image pairs to image retrieval.

A Contemporary Example of Learned Features

SuperPoint: Self-Supervised Interest Point Detection and Description

CVPR 2018

- Self-supervised framework for training interest point detectors and descriptors
- Fully-convolutional model operates on full-sized images and jointly computes pixel-level interest point locations and associated descriptors in one forward pass.

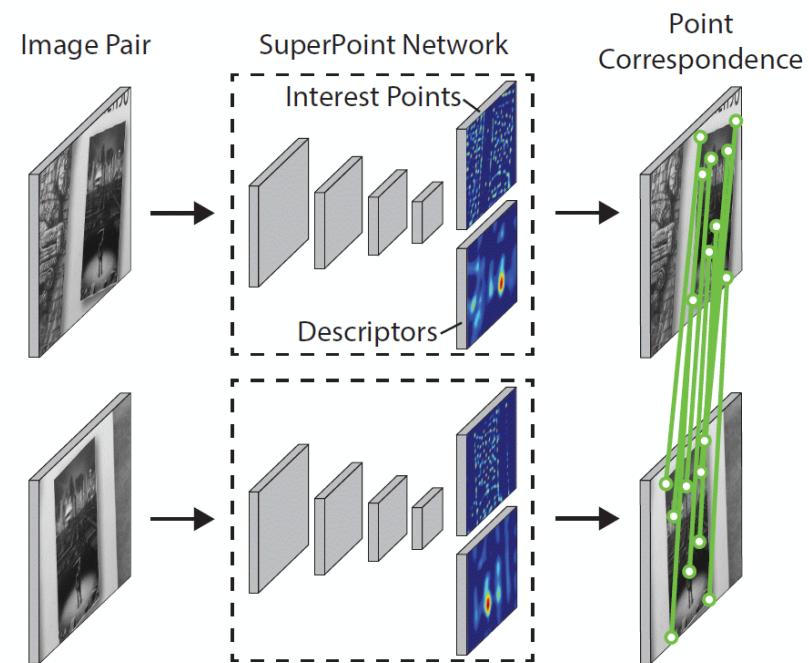


Figure 1. **SuperPoint for Geometric Correspondences.** We present a fully-convolutional neural network that computes SIFT-like 2D interest point locations and descriptors in a single forward pass and runs at 70 FPS on 480×640 images with a Titan X GPU.

Self-Supervised Training

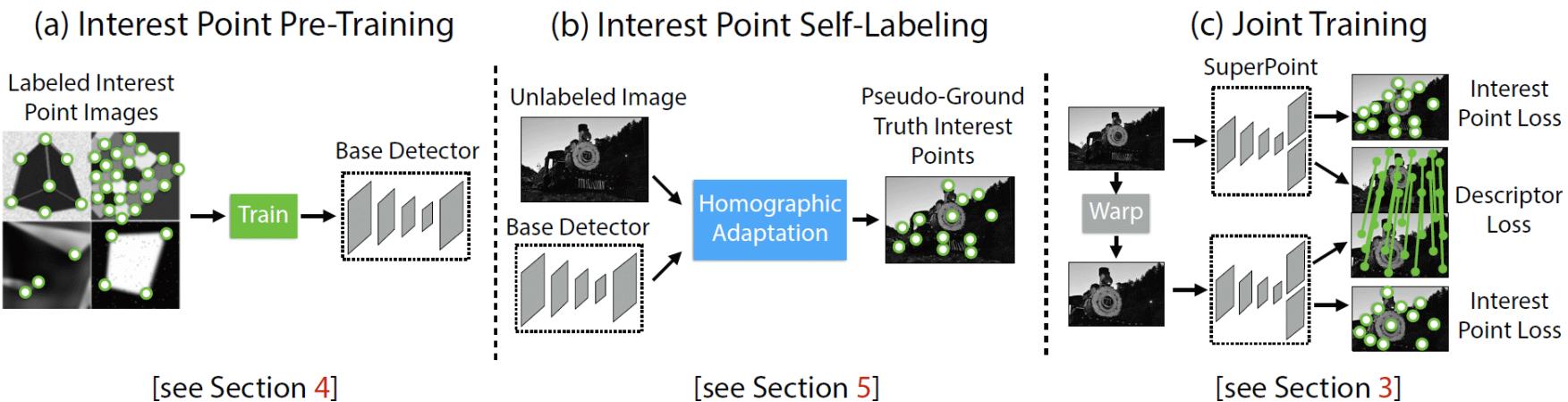


Figure 2. **Self-Supervised Training Overview.** In our self-supervised approach, we (a) pre-train an initial interest point detector on synthetic data and (b) apply a novel Homographic Adaptation procedure to automatically label images from a target, unlabeled domain. The generated labels are used to (c) train a fully-convolutional network that jointly extracts interest points and descriptors from an image.

Superpoint Architecture

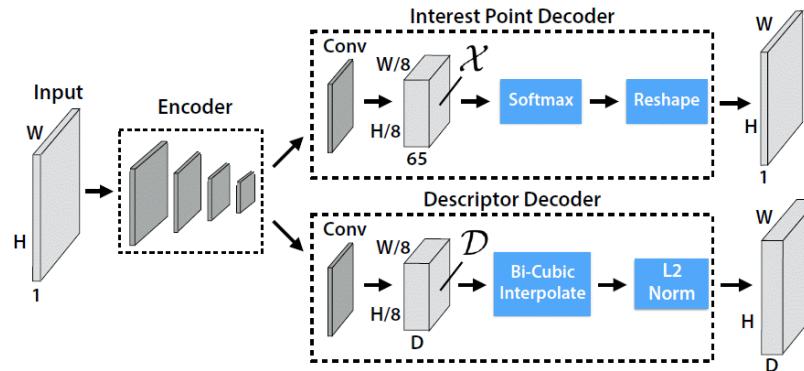


Figure 3. **SuperPoint Decoders.** Both decoders operate on a shared and spatially reduced representation of the input. To keep the model fast and easy to train, both decoders use non-learned upsampling to bring the representation back to $\mathbb{R}^{H \times W}$.

- The interest point detector head computes $H_c \times W_c \times 65$ and outputs a tensor sized $H \times W$.
- The 65 channels correspond to local, non-overlapping 8×8 grid regions of pixels plus an extra “no interest point” dustbin.
- After a channel-wise softmax, the dustbin dimension is removed and $H_c \times W_c \times 64 \rightarrow H \times W$ reshape is performed.
- The descriptor head computes $H_c \times W_c \times D$ and outputs a tensor sized $H \times W \times D$.
- To output a dense map of L2-normalized fixed length descriptors, first output a semi-dense grid of descriptors (e.g., one every 8 pixels).
- The decoder then performs bicubic interpolation of the descriptor and then L2-normalizes to unit length.

Ma, Jiayi, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. "Image matching from handcrafted to deep features: A survey." *International Journal of Computer Vision* (2021)

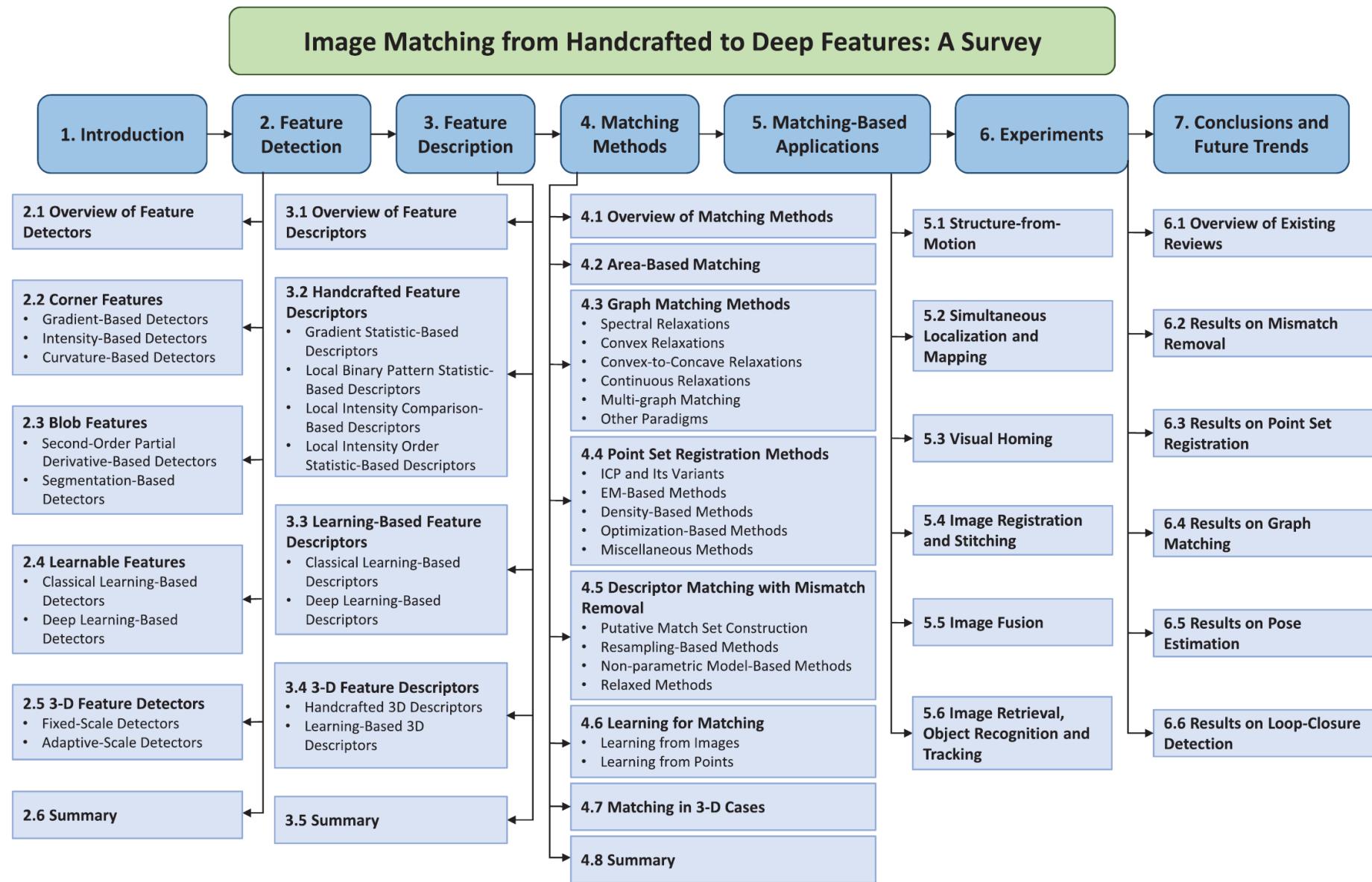


Image Feature Information Extraction for Interest

Point Detection: A Comprehensive Review

Junfeng Jing, Tian Gao, Weichuan Zhang, Member, IEEE, Yongsheng Gao, Senior Member, IEEE,
Changming Sun

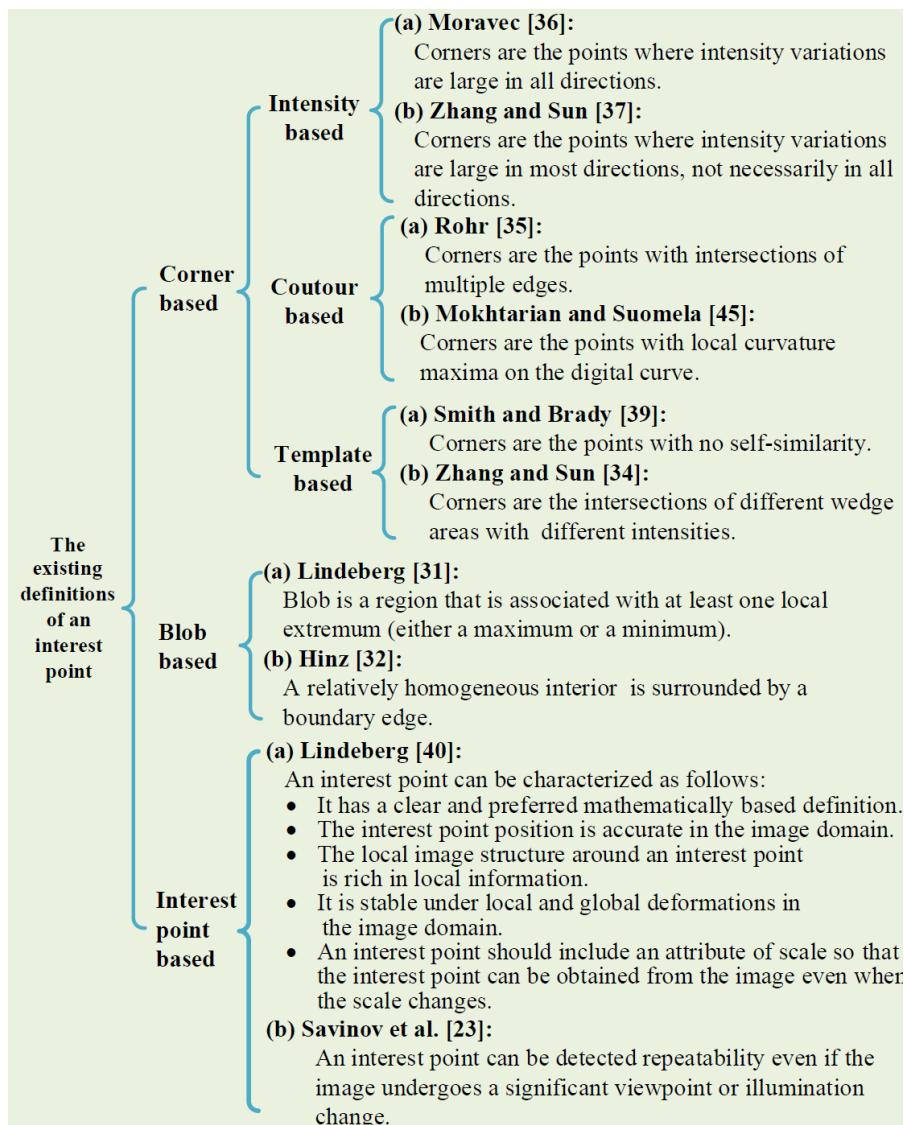


Image Feature Information Extraction for Interest

Point Detection: A Comprehensive Review

Junfeng Jing, Tian Gao, Weichuan Zhang, Member, IEEE, Yongsheng Gao, Senior Member, IEEE,
Changming Sun

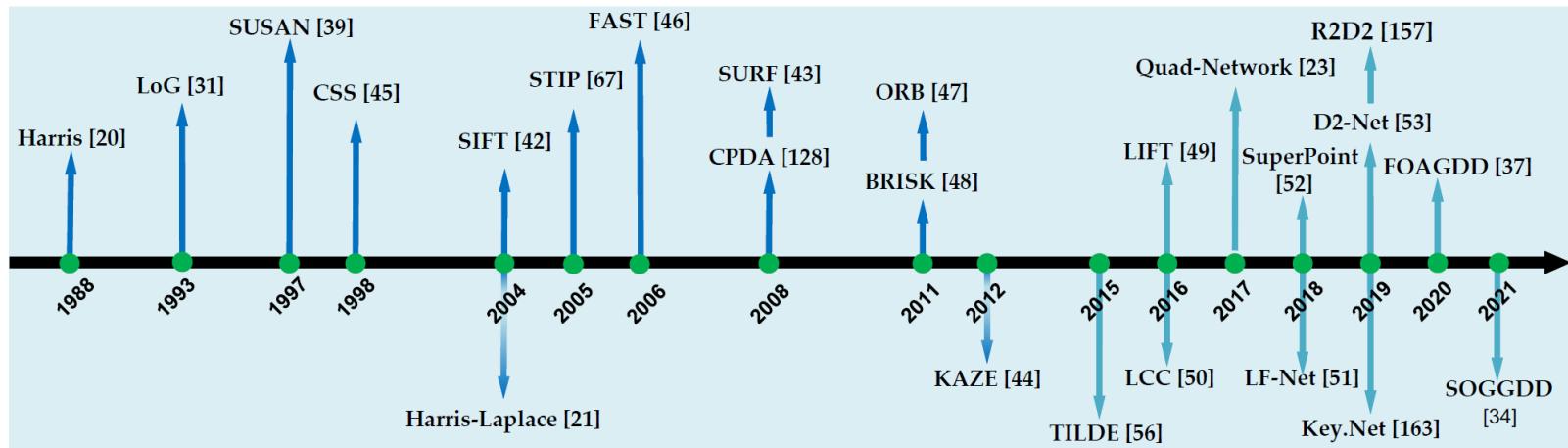


Fig. 4 Representative interest point detection methods. Starting from the Harris method [20], first-order intensity variation based interest point detection methods gained remarkable popularity. In 1993, Lindeberg [31] presented a Laplacian of Gaussian method for detecting blobs from images in a multi-scale space. Along this approach, second-order intensity variation based blob detection methods (e.g., Harris-Laplace [21], SIFT [42], SURF [43], and KAZE [44]) were proposed for achieving blob detection. In 1997, Smith and Brady [39] proposed the SUSAN method which marks the beginning of self-dissimilarity based interest point detection. In 1998, Mokhtarian and Suomela [45] proposed the CSS method which represents the edge contour based interest point detection methods and it received considerable popularity. In 2006, Rosten et al. [46] proposed the FAST detector which is one of the first machine learning methods for interest point detection. Subsequently, the ORB [47] and BRISK [48] algorithms are optimized for the FAST detector [46]. In 2016, Yi et al. [49] proposed the LIFT detector which is one of the first deep learning methods for interest point detection. And then, LCC [50], Quad-Network [23], LF-Net [51], SuperPoint [52], and D2-Net [53] were proposed by Lenc and Vedaldi, Savinov et al., Ono et al., Zhang and Rusinkiewicz, and Dusmanu et al. respectively further promoting the development of deep learning based interest point detection. In 2020 and 2021, FOAGDD [37] and SOGGDD [34] were proposed by Zhang and Sun which proved for the first time that first-order and second-order intensity variation information along multiple directions contributes significantly in improving the performance of interest point detection.

Image Feature Information Extraction for Interest

Point Detection: A Comprehensive Review

Junfeng Jing, Tian Gao, Weichuan Zhang, Member, IEEE, Yongsheng Gao, Senior Member, IEEE,
Changming Sun

TABLE 1 Popular datasets for performance evaluation on interest point detection.

Dataset Name	#Images	#Categories	Content	Synthetic or Real	Highlights
VGG [174]	48	8	Scenes	Real	Scenes with viewpoint and scale changes.
Rosten [46]	37	3	Objects	Synthetic and real	Objects with viewpoint or scale changes.
Strecha [175]	19	2	Scenes	Real	Scenes with viewpoint and scale changes.
ImageNet [164]	14,197,122	21,841	Objects	Real	More than 20,000 typical categories and each contains hundreds of images.
DTU [176]	135,660	60	Objects	Synthetic	Scenes with light, scale, and viewpoint changes.
EF [177]	38	5	Scenes	Real	Scenes with drastic light changes.
Ade-RMF [178]	76	38	Scenes	Synthetic and real	Scenes with viewpoint or scale changes.
Rome [179]	95,961	4	Scenes	Real	Scenes with rotation, scale, light, viewpoint, and photometric changes.
Symbench [180]	92	46	Scenes	Real	Scenes with light changes.
Heinly [181]	65	7	Scenes	Real	Scenes with rotation, light, scale, and image quality changes.
Webcam [56]	120	6	Scenes	Real	Scenes with light, viewpoint, and photometric changes.
Viewpoints [182]	30	5	Scenes	Real	Scenes with viewpoint and rotation changes.
ETH [183]	16,323	9	Scenes	Real	Scenes with illumination, rotation, scale, and viewpoint changes.
HPatches [184]	1,856	116	Scenes	Real	There are 116 scenes with viewpoint and photometric changes.
InLoc [185]	356	7	Objects	Real	Objects with large viewpoint and illumination changes.
Aachen Day-Night [186]	4,328	98	Scenes	Real	Scenes with dramatic season, weather, and day-night changes.
Image Matching [187]	35,279	48	Scenes	Real	Scenes with large light, scale, viewpoint, and photometric changes.

Image Feature Information Extraction for Interest

Point Detection: A Comprehensive Review

Junfeng Jing, Tian Gao, Weichuan Zhang, Member, IEEE, Yongsheng Gao, Senior Member, IEEE,
Changming Sun

- The experimental results for the fifteen state-of-the-art methods on the three evaluation criteria in [17], [53], [183] are illustrated in Fig. 6, Fig. 7, and Table 2. From the three evaluation criteria in [17], [53], [183], it can be observed that the FOAGDD [37] and SOGGDD [34] methods achieve the first and second overall detection performances. The main reason is that compared with other algorithms, these two methods [34], [37] use multi-directional filtering instead of filtering along the horizontal and vertical directions

[34] W. Zhang, C. Sun, Corner detection using second-order generalized Gaussian directional derivative representations, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (4) (2021) 1213–1224.

[37] W. Zhang, C. Sun, Corner detection using multi-directional structure tensor with multiple scales, *International Journal of Computer Vision* 128 (2) (2020) 438–459.

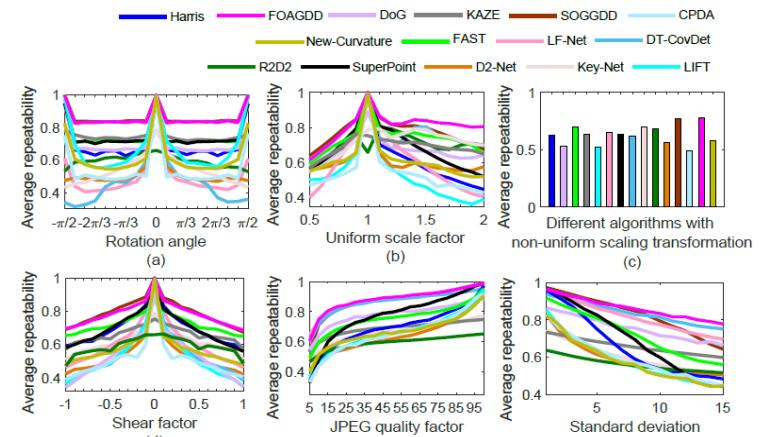


Fig. 6 Average repeatabilities of the fifteen methods under image rotation, uniform scaling, non-uniform scaling, shear transformation, lossy JPEG compression, and noise degradations.

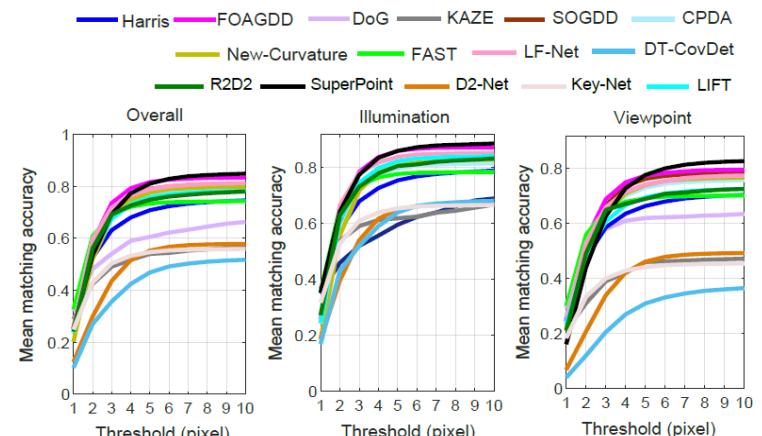


Fig. 7 Mean matching accuracy results of the fifteen detection methods with Hard-Net++ [197].

Efficient Feature Matching

- Exhaustive search
 - for each feature in one image, look at *all* the other features in the other image(s)
- Hashing
 - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
- Nearest neighbor techniques
 - kd -trees and their variants

Efficient Matching

Decided on a matching strategy, still need to search efficiently for potential candidates.

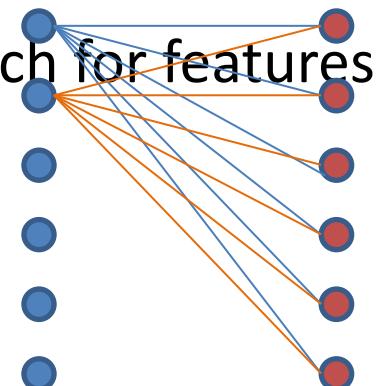
Simplest Solution: Exhaustive Search

- Compare all features against all other features in each pair of potentially matching images.

Problem: Quadratic in the number of extracted features → impractical for most applications.

Better approach: indexing structure to rapidly search for features near a given feature.

- multi-dimensional search tree or
- a hash table etc.



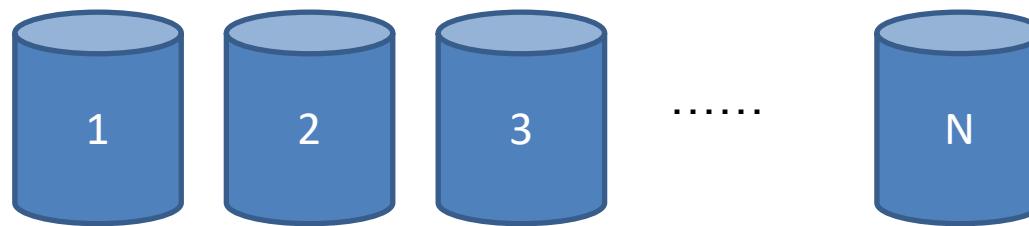
Indexing Structures

Indexing structures can be built

- Independently for each image: useful when we want to only consider certain potential matches, e.g., searching for a particular object
- Globally for all the images in a given database: potentially faster, since it removes the need to iterate over each image. For extremely large databases (millions of images or more)
- More efficient structures (e.g. vocabulary trees (Nister and Stewenius 2006)) based on ideas from document retrieval.

Efficient Matching

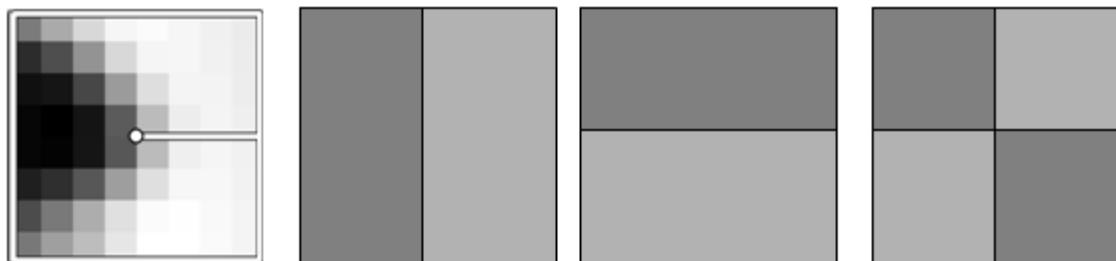
- Multi-dimensional hashing
 - Map descriptor into buckets
 - When matching, new feature is hashed into a bucket
 - Search for nearby buckets to return potential candidates



Haar Wavelets

MOPS, Brown, Szeliski, and Winder (2005)

- 3D index by performing sum over the quadrants



- Normalize the 3 values by their expected standard deviations
- $V_n \rightarrow$ two nearest bins (of 10) \rightarrow index $2^3 = 8$ bins
- Query: primary 3D bin, k nearest neighbors for further process

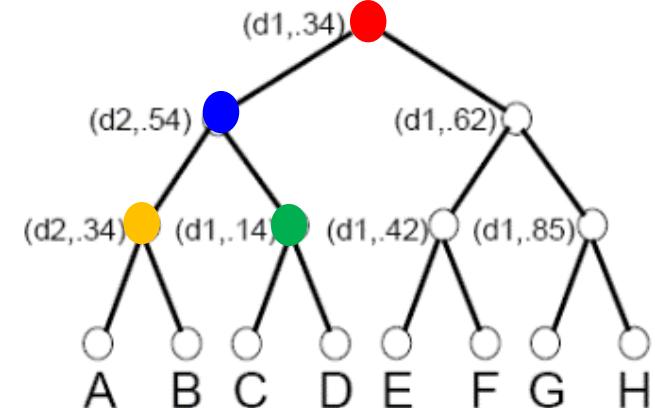
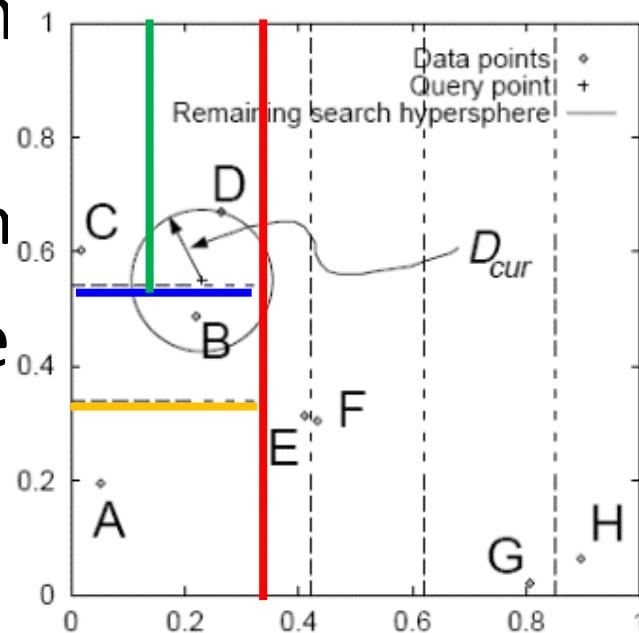
More Structures

Page 232, 233

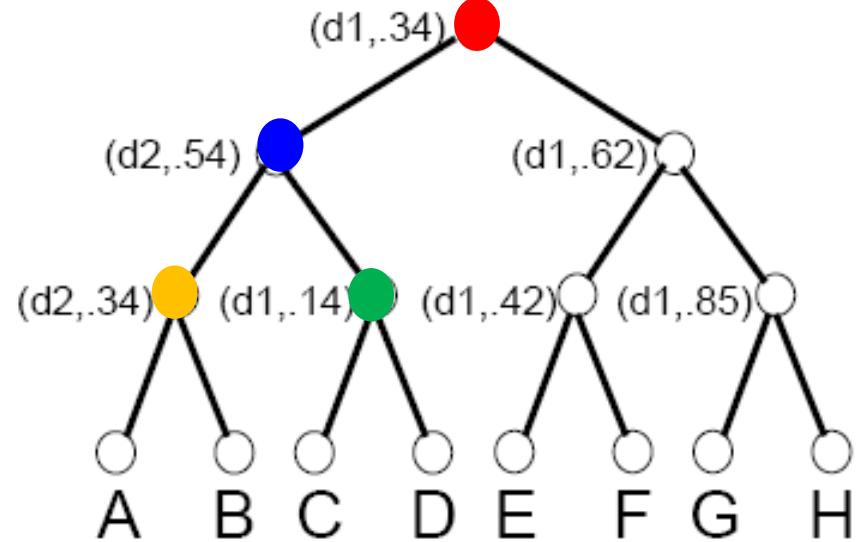
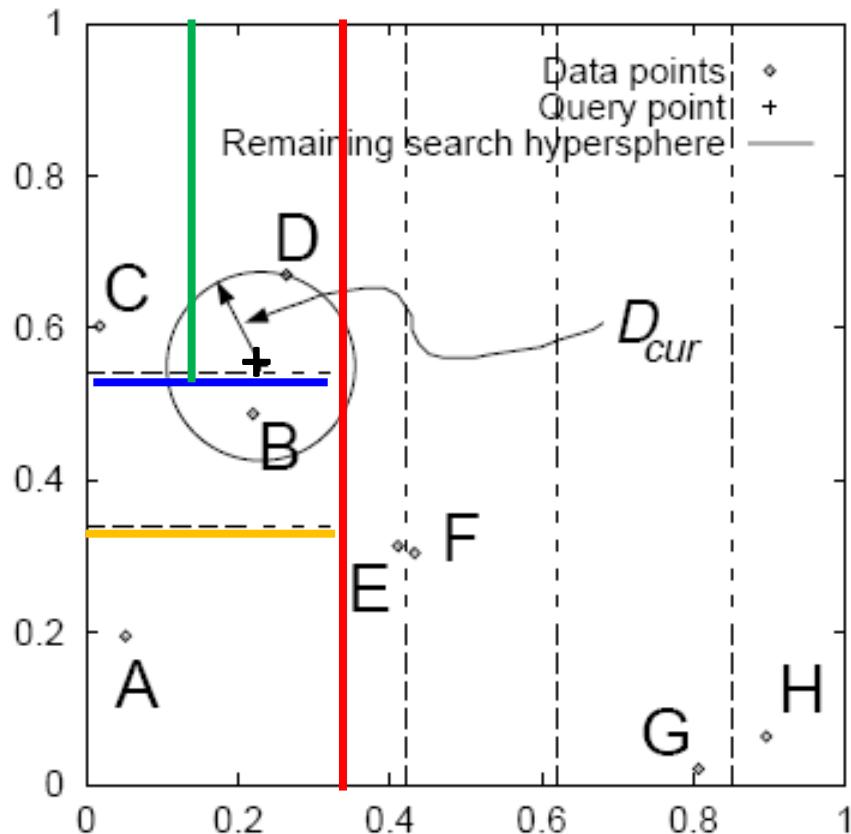
- Locality Sensitive Hashing
 - More widely applicable
 - Use unions of independently computed hashing functions to index the features
- Parameter-Sensitive Hashing
 - More sensitive to the distribution of points in parameter space
- High-D vectors to binary codes
 - Compared using Hamming distance
 - Accommodate arbitrary kernel functions

KD Tree

- Multi-dimensional search tree
- Recursively divide multi-D feature space along alternating axis-aligned hyperplanes
- Choose the axis to split that maximizes information gain
- Depth as search number
- Query: nearest neighbor



KD Tree



Query Time:

1. Locate the query point in its appropriate bin (+->D)
2. Search nearby leaves in the tree (C,B,...) until it can guarantee that nearest neighbor has been found

Additional Data Structure

Page 234

- Slicing
 - use a series of 1D binary searches on point list sorted along different dimensions to efficiently cull down a list of candidate points that lie within a hypercube of the query point
- Reweight the matches at different levels of the indexing tree
 - Less sensitive to discretization errors in tree construction
- Metric tree
 - A small number of prototypes at each level in a hierarchy
 - Visual words → classical information retrieval, fast

Indexing Structure

- Survey and comparison on indexing structures
Muja and Lowe (2009)
- Kd tree works best
- Rapid computation of image feature correspondences remains a challenging open research problem

Illustration - Matching



Interest points matched based on cross-correlation (188 pairs)

Feature match verification and densification

Once we have some hypothetical matches, Verify Inliers and outliers

- **Geometry alignment**
 - Example: if we expect the whole image to be translated or rotated in the matching view
-> we can fit a global geometric transform and keep only those feature matches that are sufficiently close to this estimated transformation.
- **RANSAC, Random sampling** : The process of selecting a small set of seed matches and then verifying a larger set is often called random sampling or RANSAC.

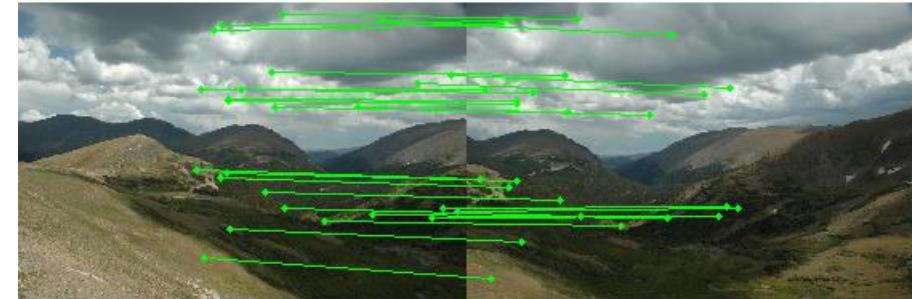
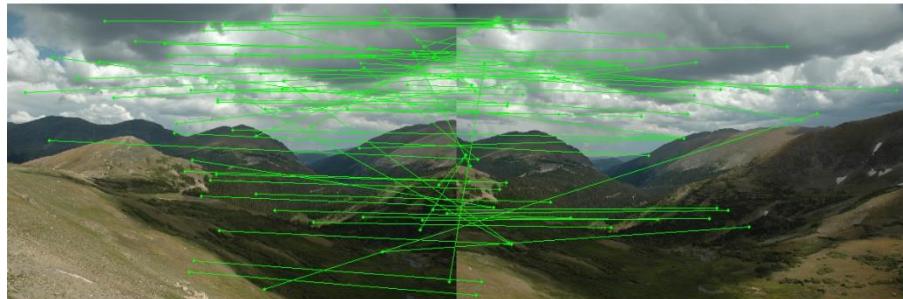
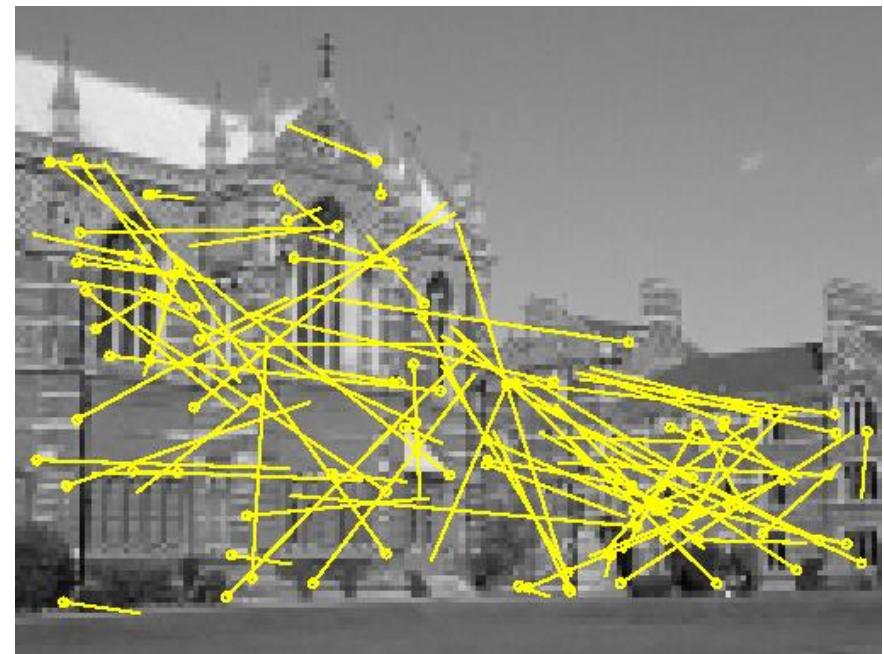


Figure from Xuejin Chen

Illustration - Matching



99 inliers



89 outliers

Feature Tracking

Alternative to Feature Matching

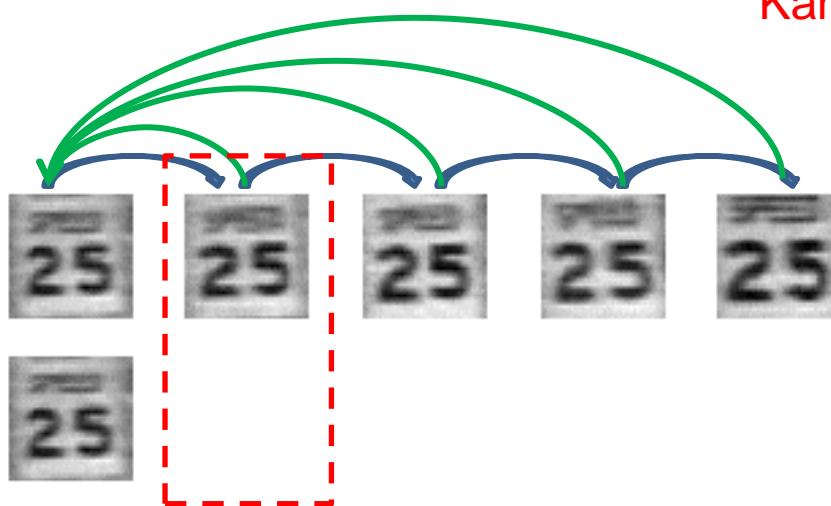
- only searches a small neighborhood around each detected feature and is therefore more suitable for video processing.

Issues

- Long image sequence: large appearance change
 - Whether to continue matching against the originally detected patch
 - Fail when original patch appearance changes
 - or
 - Re-sample each subsequent frame at the matching location
 - Risk: features drift from its original location

Feature Tracking

- Affine motion model (Shi and Tomasi 1994)
 - Compare patches in neighboring frames using a translational model
 - Use the location to initialize an affine registration between the patch in the current frame and the base frame



Kanade–Lucas–Tomasi (KLT) tracker.

Detect new features in regions where the tracking has fail

- The area around the predicted feature location is searched with an incremental registration algorithm

A lot of Expansions

Page 236

- Tracking combined with structure from motion
- Tie together the corners
- Tracking in video with large number of moving objects or points
- Special purpose recognizer: Learning algorithms
 - Train classifiers on sample patches and their affine deformation, fast and reliable, fast motion is supported
- Survey ... (Yilmaz, Javed, and Shah 2006)

Image Alignment and Stitching-1: Geometric Transformations

Gonzalez & Woods Chapter 2

Szeliski Computer Vision Book Chapter 3.6

Image Alignment

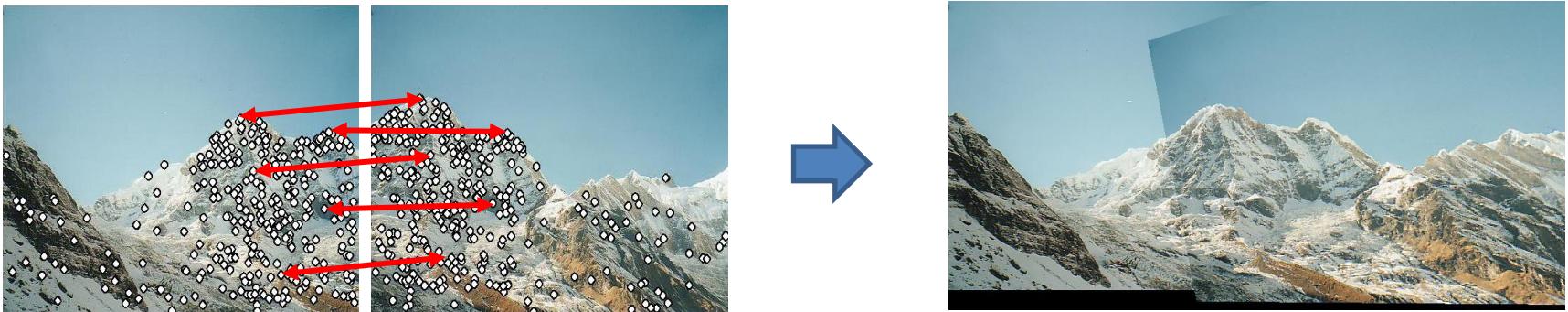
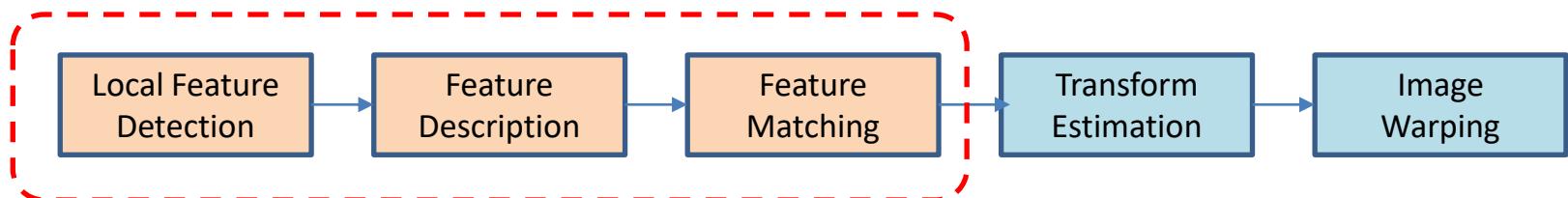


Image Processing

1. Point operators/point processes

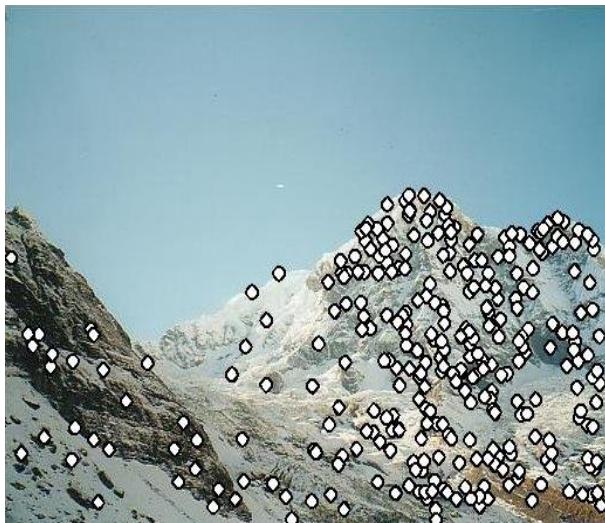
1. Contrast Enhancement
2. Histogram

2. Neighborhood (area-based) operators

1. Linear Filtering – Smoothing, edge detection, blob detection...
2. Non-Linear Filtering – Median filter etc.....
3. Morphological Filtering –Gonzalez&Woods Chapter 9

3. Geometric transformations : rotation, shears, perspective deformations

What is the geometric relationship
between these two images?

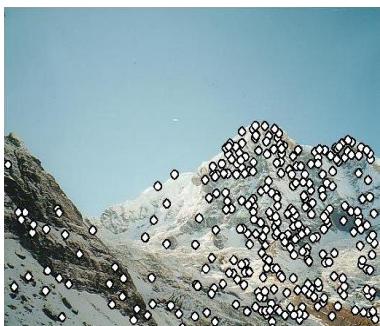


Geometric Transformations

(Gonzalez & Woods Sec 2.5)

- Geometric transformations are used to modify the spatial arrangement of pixels in an image.
- These transformations are called *rubber-sheet transformations* because they may be viewed as analogous to “printing” an image on a rubber sheet, then stretching or shrinking the sheet according to a predefined set of rules.
- Geometric transformations of digital images consist of two basic operations:
 1. Spatial transformation of coordinates.
 2. Intensity interpolation that assigns intensity values to the spatially transformed pixels.

**What is the geometric relationship between these two images?
We need to figure out how to compute it using feature matches.**



Basic Set of Geometric Transformations

(Szeliski Sec 3.6 & 2.1.2)

- Point Processes, Linear & NonLinear Filtering:
transform the range of the image
$$g(x)=h(f(x))$$
- Geometric Transformations: transforms the domain.
$$g(x)=f(h(x))$$

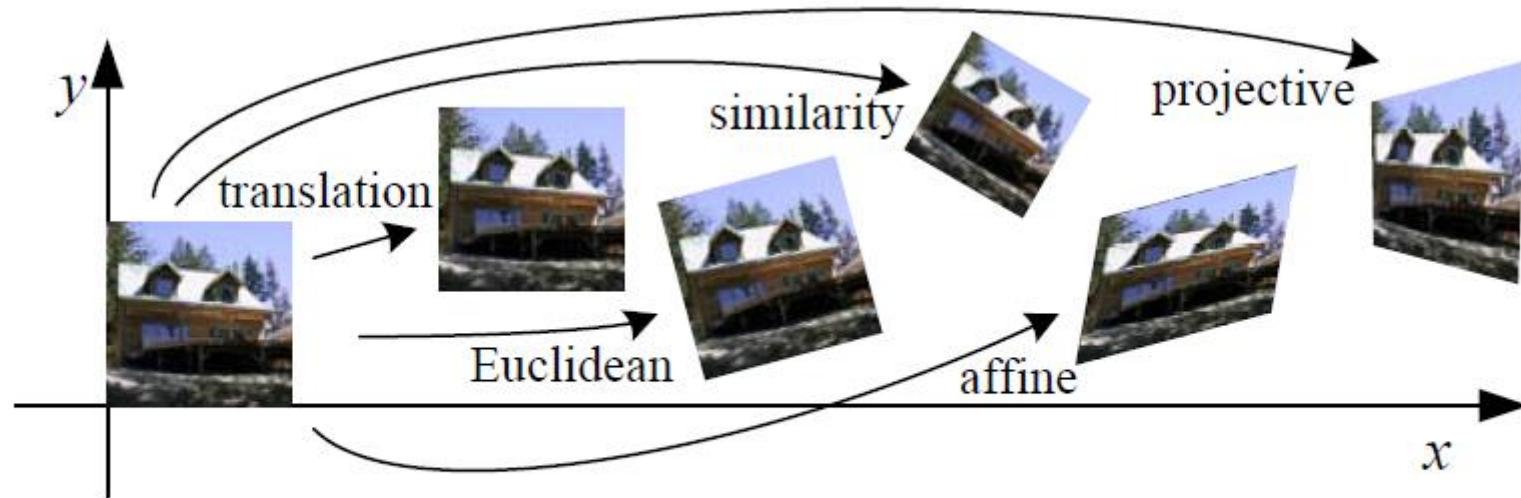
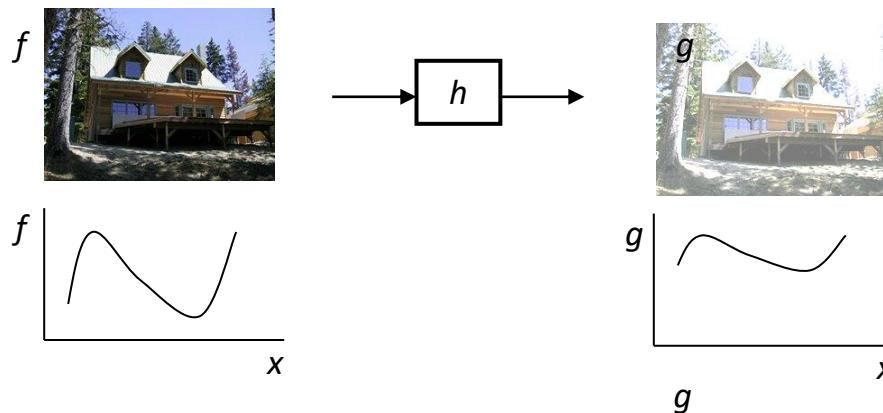


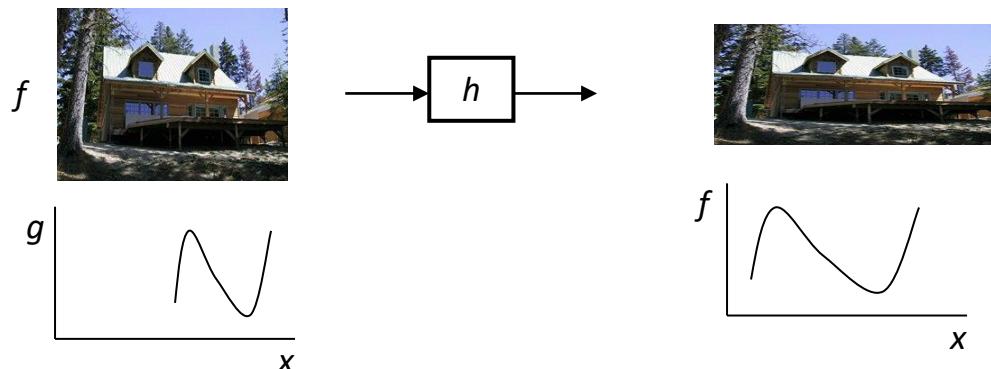
Figure 3.45 Basic set of 2D geometric image transformations.

Image Warping

- image filtering: change *range* of image $g(x) = h(f(x))$



- image warping: change *domain* of image $g(x) = f(h(x))$



Parametric (global) warping

- Examples of parametric warps:



translation

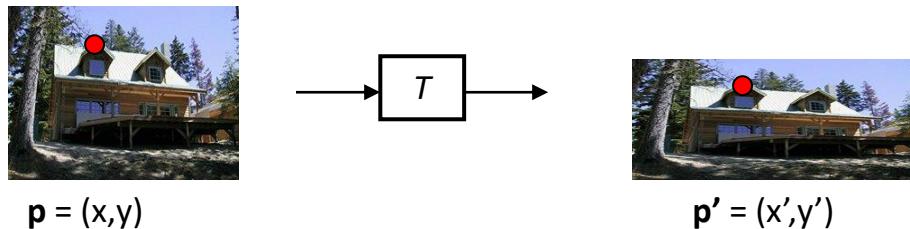


rotation



aspect

Parametric (global) warping



- Transformation T is a coordinate-changing machine:
$$p' = T(p)$$
- What does it mean that T is global?
 - Is the same for any point p
 - can be described by just a few numbers (parameters)
- Let's consider *linear* xforms (can be represented by a 2×2 matrix):

$$p' = Tp \quad \boxed{\begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}}$$

Common linear transformations

- Uniform scaling by s :



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

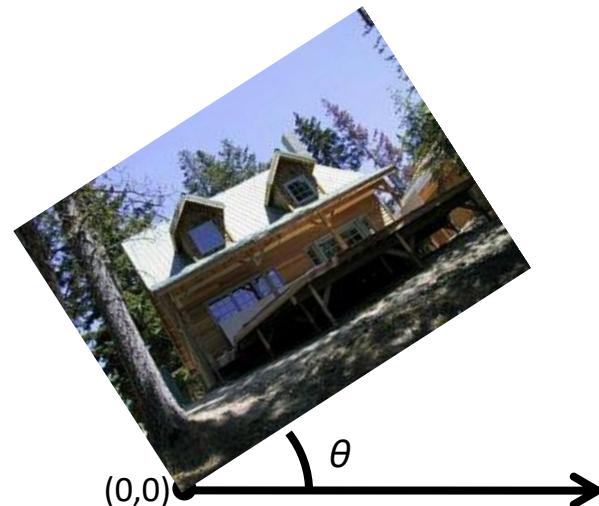
$$\mathbf{S} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

What is the inverse?

Common linear transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Rotation by angle θ (about the origin)



$$\mathbf{R} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

What is the inverse?
For rotations:
 $\mathbf{R}^{-1} = \mathbf{R}^T$

2x2 Matrices

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

- What types of transformations can be represented with a 2x2 matrix?

2D mirror across Y axis?

$$x' = -x$$

$$y' = y$$

2D mirror across line $y = x$?

$$x' = y$$

$$y' = x$$

2x2 Matrices

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

- What types of transformations can be represented with a 2x2 matrix?

2D mirror across Y axis?

$$\begin{array}{lcl} x' & = & -x \\ y' & = & y \end{array} \quad \mathbf{T} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

2D mirror across line $y = x$?

$$\begin{array}{lcl} x' & = & y \\ y' & = & x \end{array} \quad \mathbf{T} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

2x2 Matrices

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \mathbf{T} \begin{bmatrix} x \\ y \end{bmatrix}$$

- What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$x' = x + t_x$$

$$y' = y + t_y$$

All 2D Linear Transformations

- Linear transformations are combinations of ...

- Scale,
 - Rotation,
 - Shear, and
 - Mirror

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Properties of linear transformations:

- Origin maps to origin
 - Lines map to lines
 - Parallel lines remain parallel
 - Ratios are preserved
 - Closed under composition

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} i & j \\ k & l \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D Translation?

$$\begin{aligned}x' &= x + t_x \\y' &= y + t_y\end{aligned}\quad \text{NO!}$$

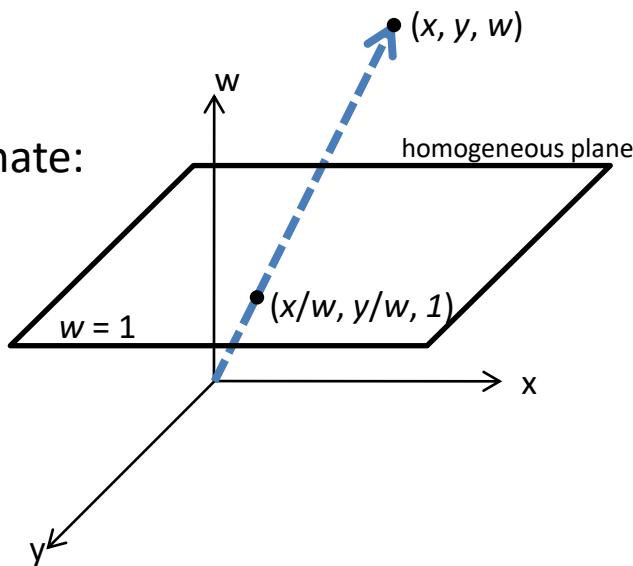
Translation is not a linear operation on 2D coordinates

Homogeneous coordinates

Trick: add one more coordinate:

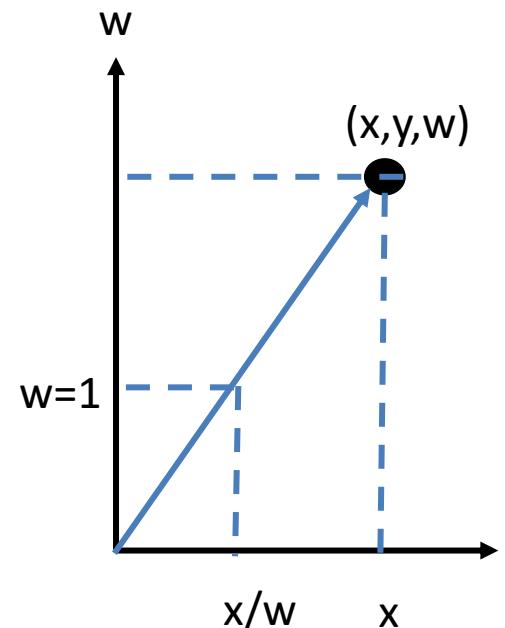
$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image
coordinates



Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$



Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths	

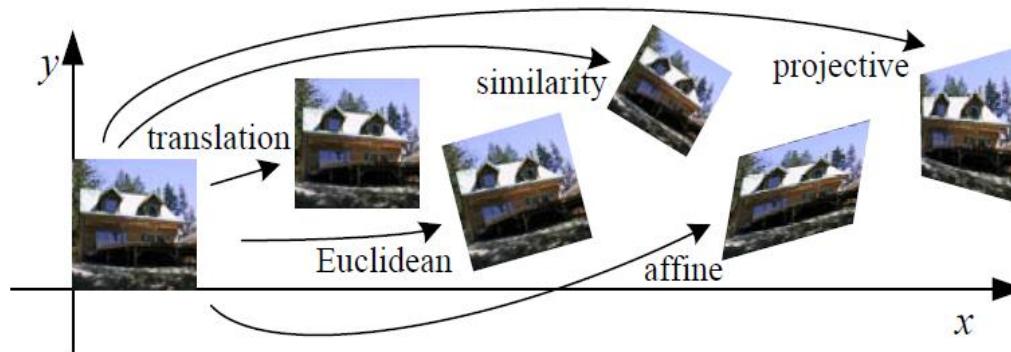


Figure 3.45 Basic set of 2D geometric image transformations.

Translation

$$x' = x + t$$

$$x' = [\mathbf{I} \ \mathbf{t}]x$$

Homogeneous coordinates:
a point becomes a column
vector whose third component is 1

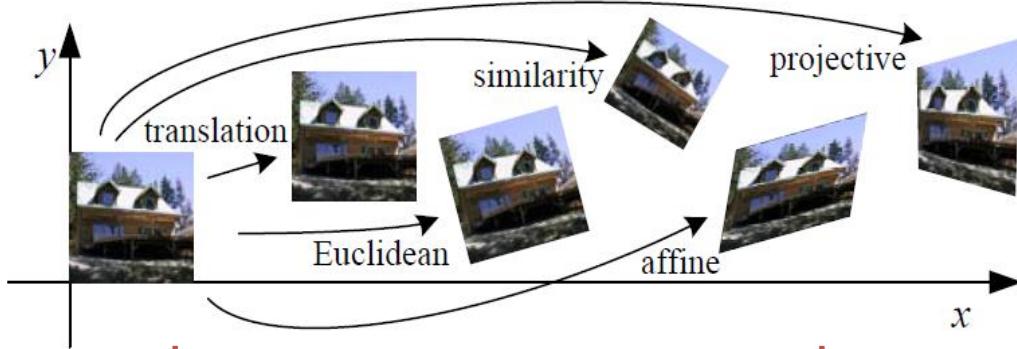
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotation + Translation
(2D Rigid body motion
2D Euclidean Transformation
- Preserves Euclidean distances)

$$x' = [\mathbf{R} \ \mathbf{t}]x$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Transformation	Matrix	# DoF	Preserves	Icon
similarity	$\left[\begin{array}{c c} sR & t \end{array} \right]_{2 \times 3}$	4	angles	
affine	$\left[\begin{array}{c} A \end{array} \right]_{2 \times 3}$	6	parallelism	
projective	$\left[\begin{array}{c} \tilde{H} \end{array} \right]_{3 \times 3}$	8	straight lines	



Scaled Rotation + Translation
(similarity transform
 $a^2+b^2=1$ no longer required)

Affine
(A is arbitrary)

Projective (Perspective Transform/Homography)
(H arbitrary)

$$x' = [sR \ t]x$$

$$x' = [A]x$$

$$x' = [H]x$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

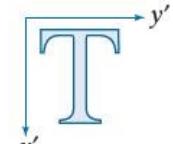
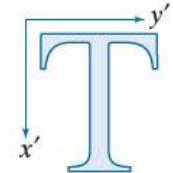
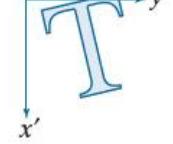
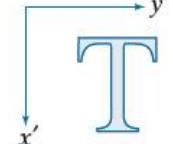
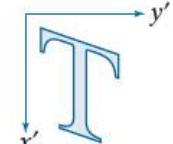
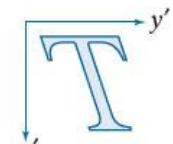
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine Transformations

$$x' = Ax$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Affine transformations are combinations of ...
 - Linear transformations, and
 - Translations
- Properties of affine transformations:
 - Origin does not necessarily map to origin
 - Lines map to lines
 - Parallel lines remain parallel
 - Ratios are preserved
 - Closed under composition

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to -1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x \cos \theta - y \sin \theta$ $y' = x \sin \theta + y \cos \theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

Is this an affine transformation?



Slide credit: Noah Snavely

Projective Transformations/ Perspective Transform /Homography

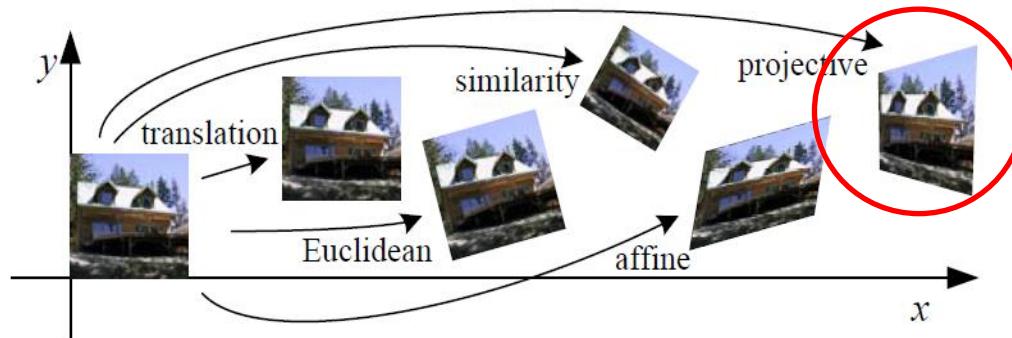


Figure 3.45 Basic set of 2D geometric image transformations.

Projective (Perspective
Transform/Homography)
(H arbitrary)

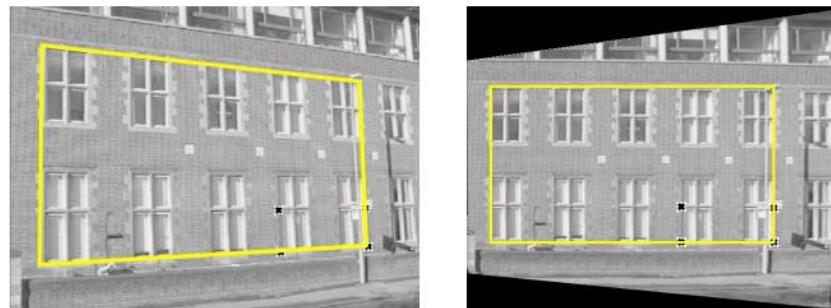
$$x' = [H]x$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homographies

- Homographies ...
 - Affine transformations, and
 - Projective warps
- Properties of projective transformations:
 - Origin does not necessarily map to origin
 - Lines map to lines
 - Parallel lines do not necessarily remain parallel
 - Ratios are not preserved
 - Closed under composition

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$



from Hartley & Zisserman

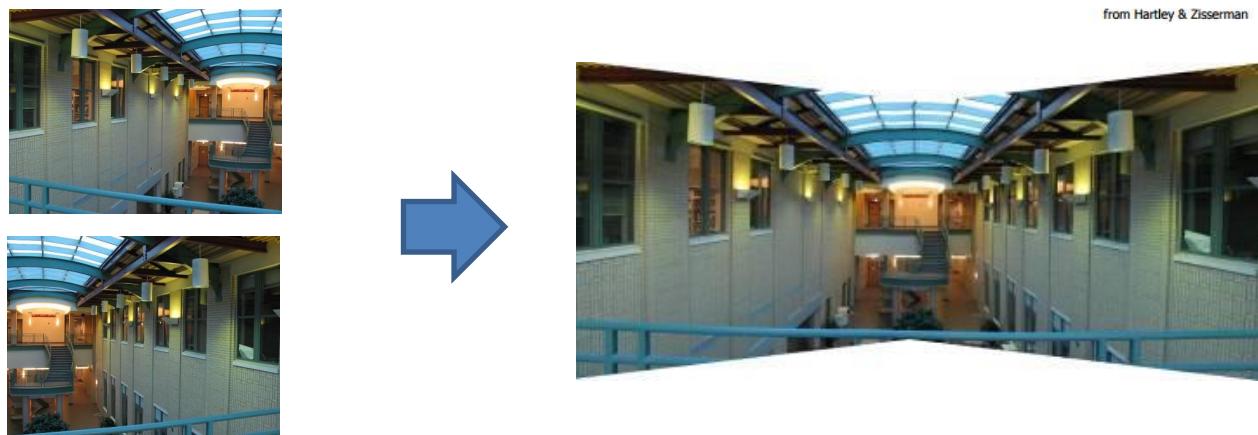


Image credit: Hartley & Zisserman (top) Noah Snavely (bottom)

Homographies

Planar homography relates the transformation between two planes (up to a scale factor):

a planar surface viewed by two camera positions (images taken from [3](#) and [2](#))

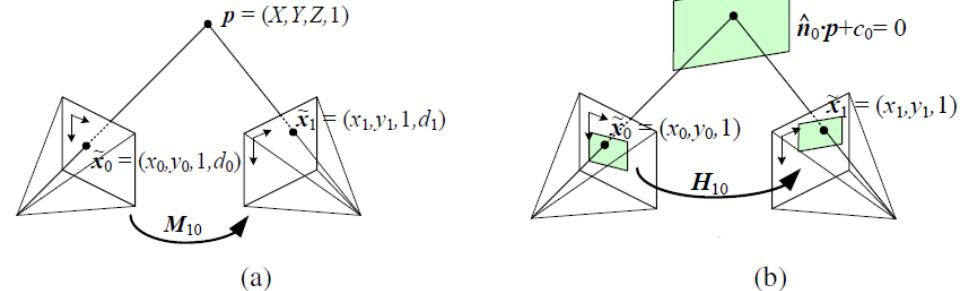
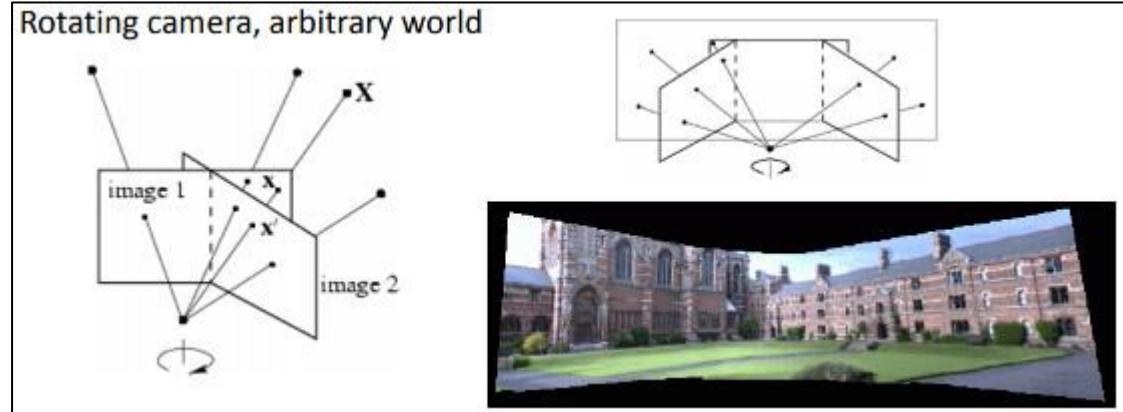


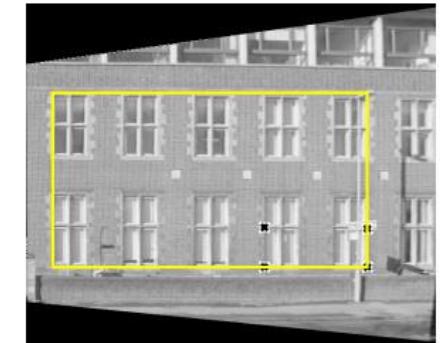
Figure 2.12 A point is projected into two images: (a) relationship between the 3D point coordinate $(X, Y, Z, 1)$ and the 2D projected point $(x, y, 1, d)$; (b) planar homography induced by points all lying on a common plane $\hat{n}_0 \cdot p + c_0 = 0$.

a rotating camera around its axis of projection, equivalent to consider that the points are on a plane at infinity (image taken from [2](#))



1. https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html
2. 2D projective transformations (homographies), Christiano Gava, Gabriele Bleser
3. Computer Vision: Algorithms and Applications, Richard Szeliski

Homography Transform



from Hartley & Zisserman



https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html

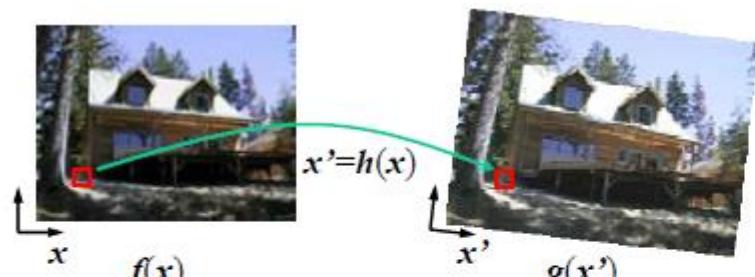
Warping

```
procedure forwardWarp( $f, h, \text{out } g$ ):
```

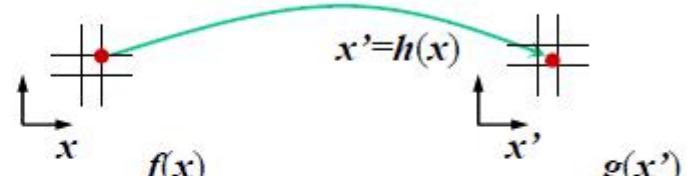
For every pixel x in $f(x)$

1. Compute the destination location $x' = h(x)$.
2. Copy the pixel $f(x)$ to $g(x')$.

Algorithm 3.1 Forward warping algorithm for transforming an image $f(x)$ into an image $g(x')$ through the parametric transform $x' = h(x)$.



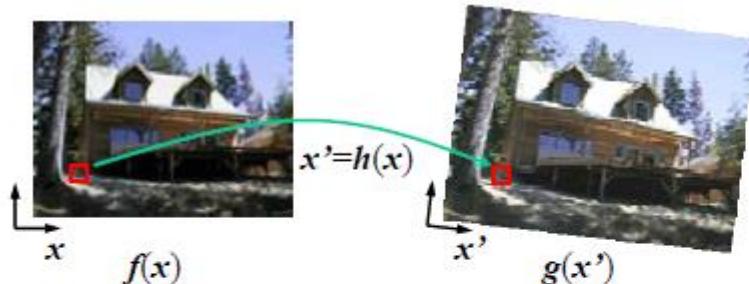
(a)



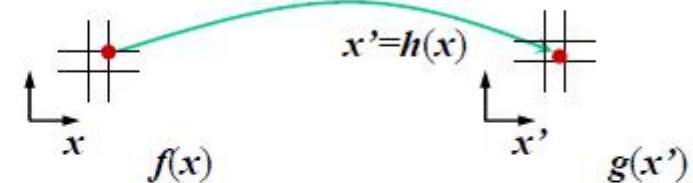
(b)

Figure 3.46 Forward warping algorithm: (a) a pixel $f(x)$ is copied to its corresponding location $x' = h(x)$ in image $g(x')$; (b) detail of the source and destination pixel locations.

Forward Warping - Problems



(a)



(b)

Figure 3.46 Forward warping algorithm: (a) a pixel $f(x)$ is copied to its corresponding location $x' = h(x)$ in image $g(x')$; (b) detail of the source and destination pixel locations.

Problems:

1. What if x' has a non integer value?

- Solution: Round x' to the nearest coordinate, copy pixel there -> severe aliasing
- Solution: Distribute the value among 4 nearest neighbors (weighted bilinear) -> moderate aliasing but blur (loss of high resolution details).

2. Cracks & Holes: Especially when magnifying

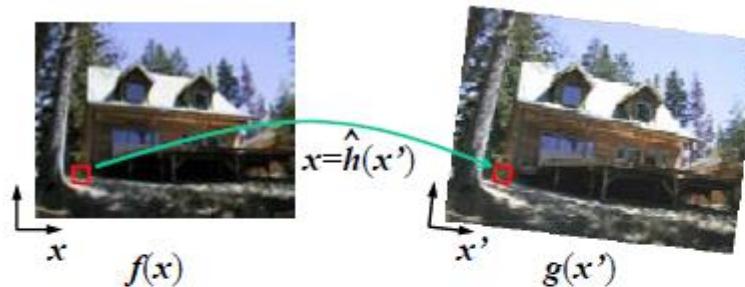
Backward Warping

```
procedure inverseWarp( $f, h, \text{out } g$ ):
```

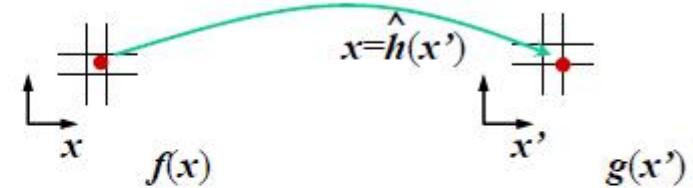
For every pixel x' in $g(x')$

1. Compute the source location $x = \hat{h}(x')$
2. Resample $f(x)$ at location x and copy to $g(x')$

Algorithm 3.2 Inverse warping algorithm for creating an image $g(x')$ from an image $f(x)$ using the parametric transform $x' = h(x)$.



(a)



(b)

Figure 3.47 Inverse warping algorithm: (a) a pixel $g(x')$ is sampled from its corresponding location $x = \hat{h}(x')$ in image $f(x)$; (b) detail of the source and destination pixel locations.

Backward Warping

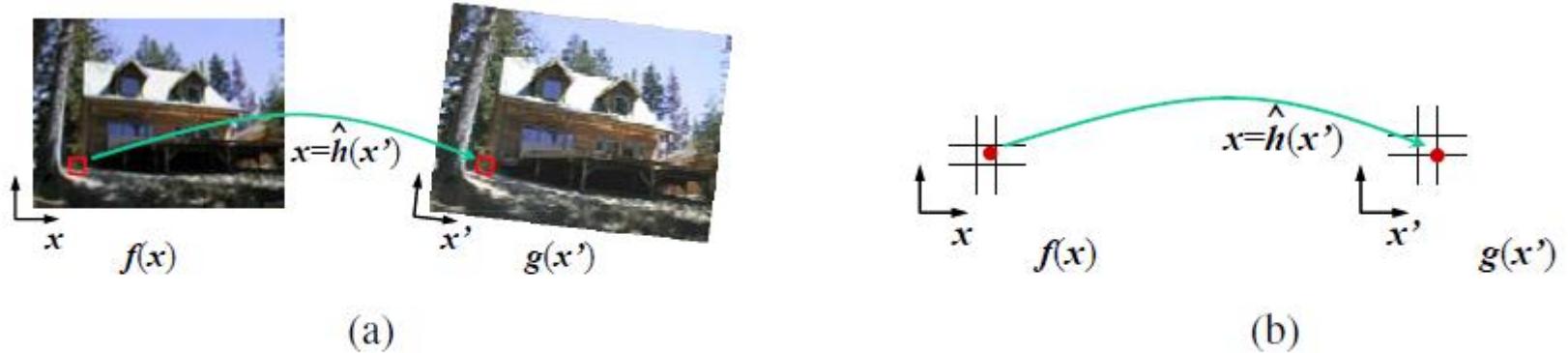
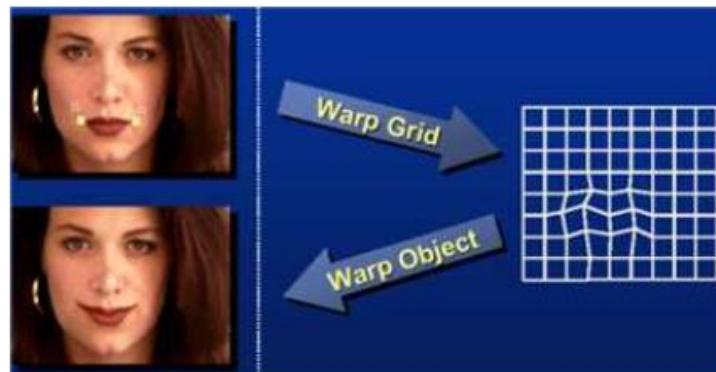


Figure 3.47 Inverse warping algorithm: (a) a pixel $g(x')$ is sampled from its corresponding location $x = \hat{h}(x')$ in image $f(x)$; (b) detail of the source and destination pixel locations.

- No holes or cracks!
- Resampling an image at non-integer locations
 - >well studied problem (general image interpolation)
- Warping: reformulated as resampling a source image!

Beyond Global Warping



(a)



(b)



(c)



(d)

Figure 3.49 *Image warping alternatives* (Gomes, Darsa et al. 1999) © 1999 Morgan Kaufmann: (a) sparse control points → deformation grid; (b) denser set of control point correspondences; (c) oriented line correspondences; (d) uniform quadrilateral grid.

Beyond Global Warping

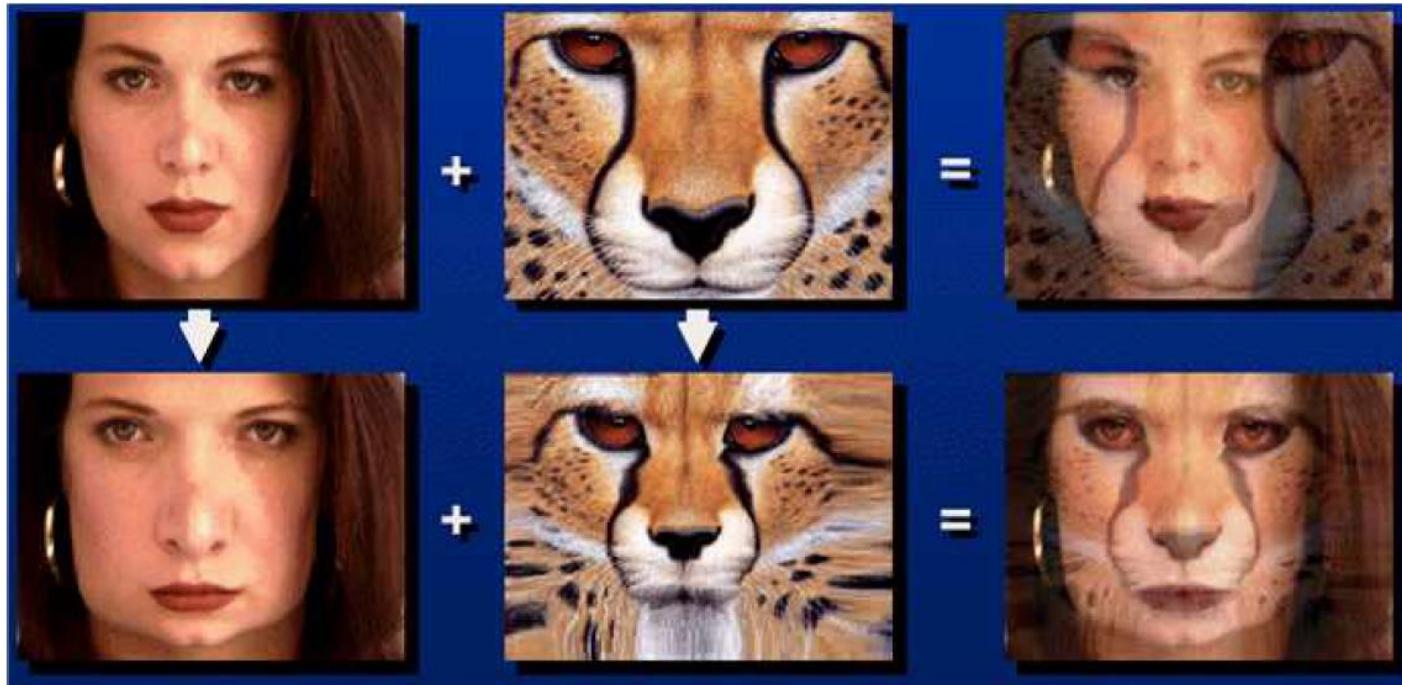
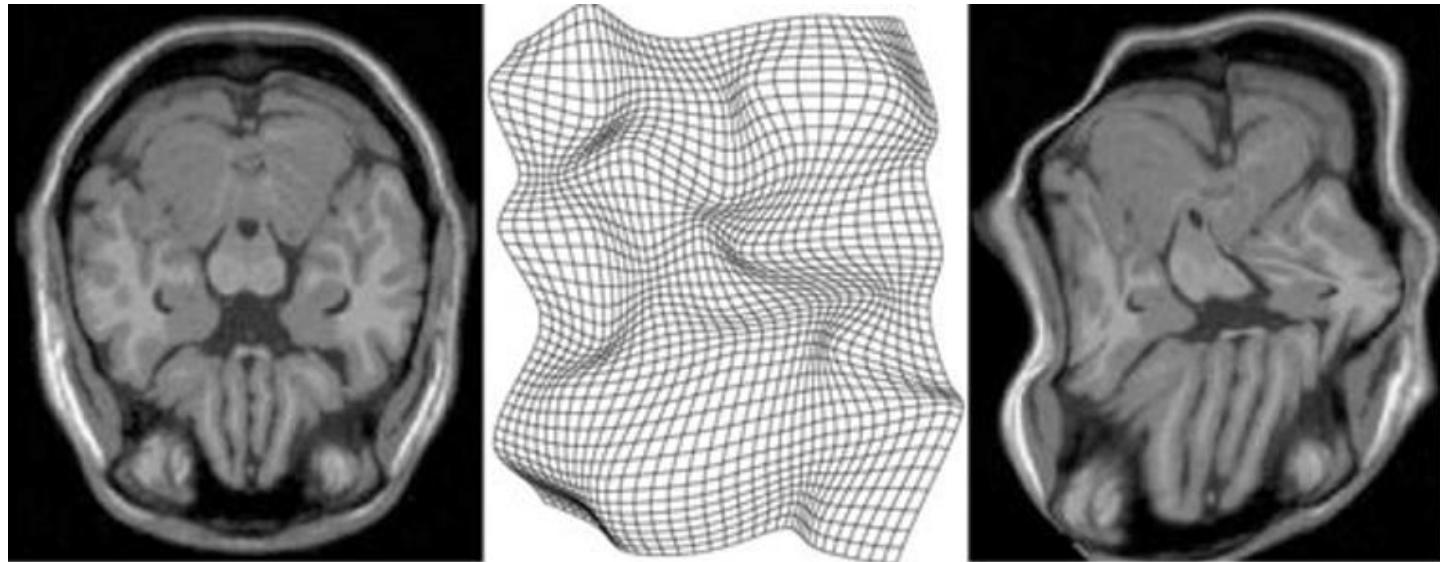


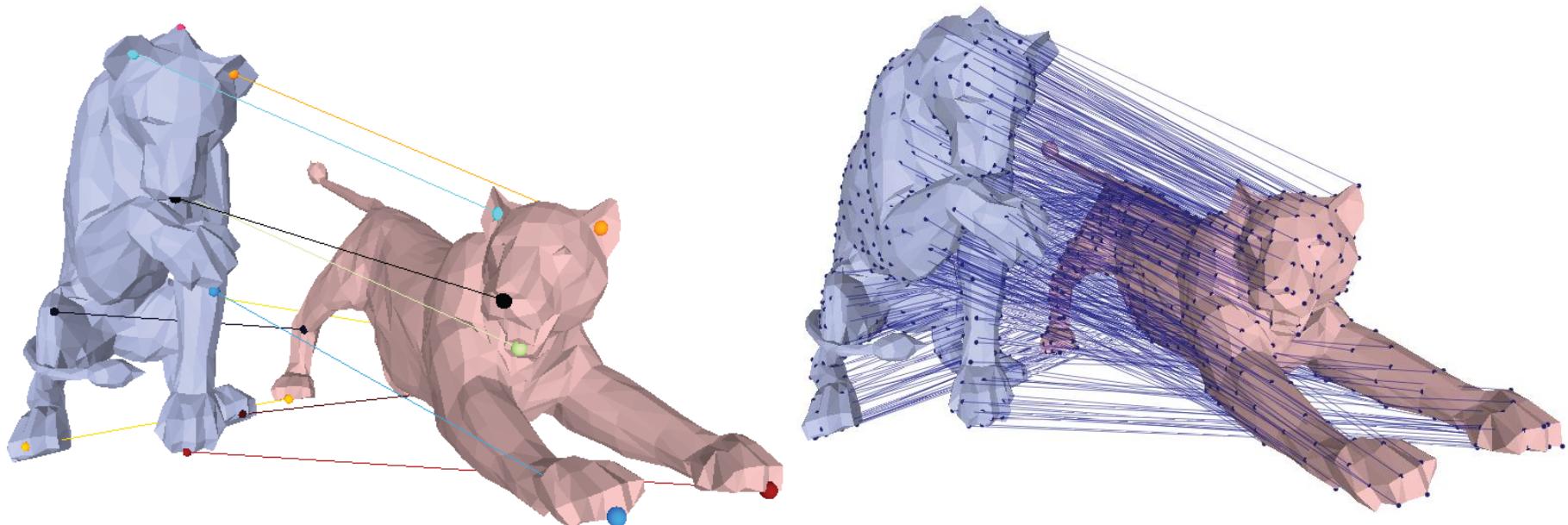
Figure 3.51 *Image morphing (Gomes, Darsa et al. 1999) © 1999 Morgan Kaufmann. Top row: if the two images are just blended, visible ghosting results. Bottom row: both images are first warped to the same intermediate location (e.g., halfway towards the other image) and the resulting warped images are then blended resulting in a seamless morph.*

Non-rigid transformation example showing image, warp field, and warped image



Durech, Eduard F. "Deep Convolutional Neural Network for Non-rigid Image Registration." *arXiv preprint arXiv:2104.12034* (2021).

A dense matching result between two surfaces undergoing
a large non-rigid deformation



Zeng, Yun, Chaohui Wang, Yang Wang, Xianfeng Gu, Dimitris Samaras, and Nikos Paragios. "Dense non-rigid surface registration using high-order graph matching." In *2010 IEEE computer society conference on computer vision and pattern recognition*, pp. 382-389. IEEE, 2010.

Blending

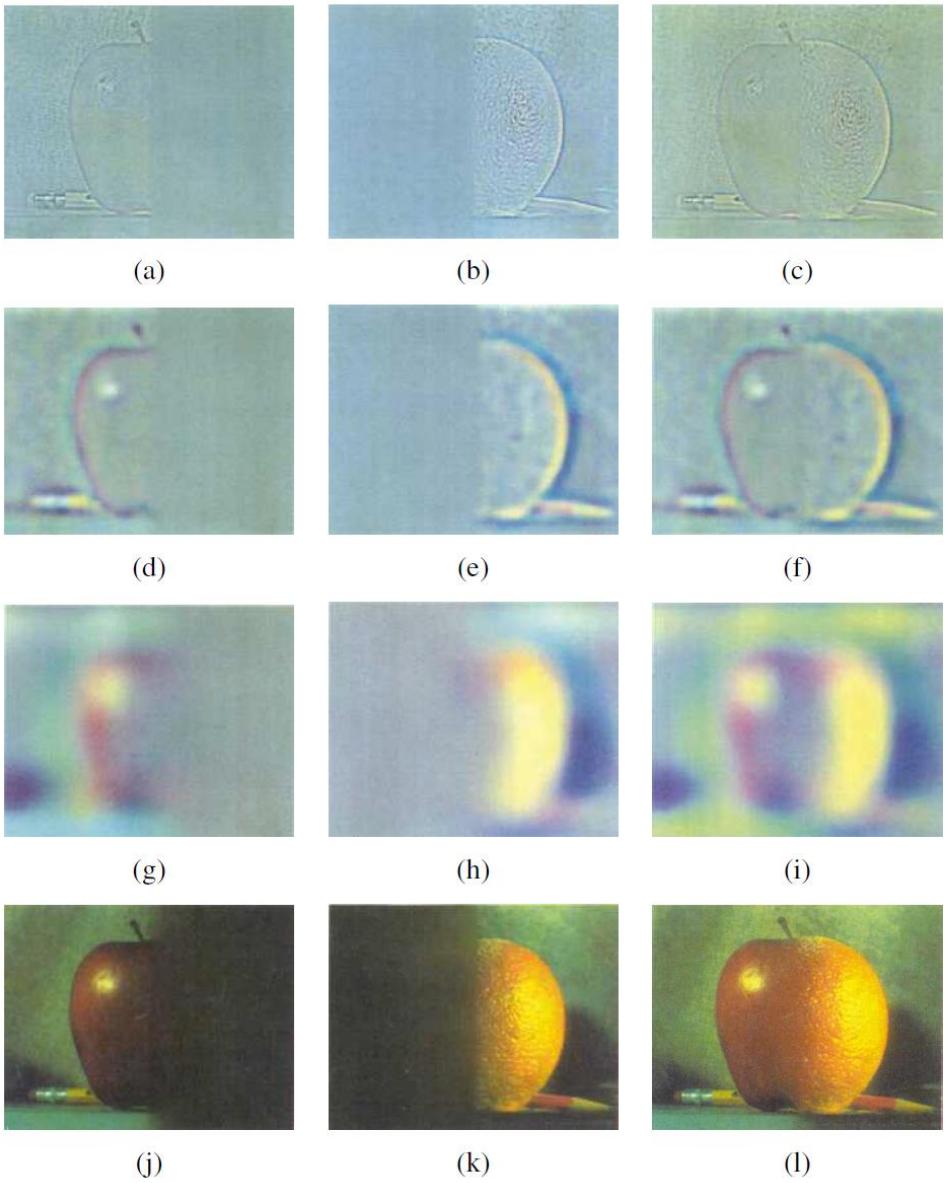
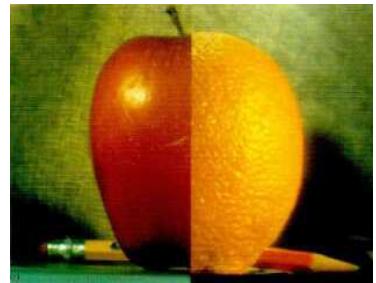
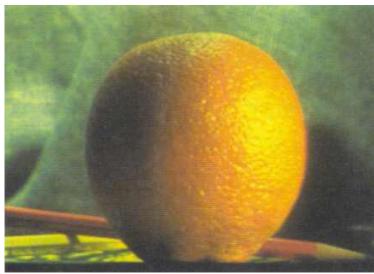
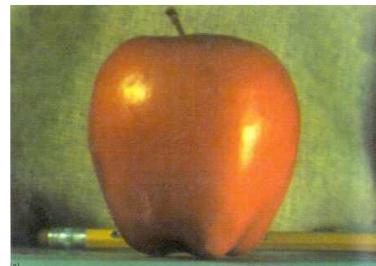


Figure 3.41 Laplacian pyramid blending (Burt and Adelson 1983b) © 1983 ACM: (a) original image of apple, (b) original image of orange, (c) regular splice, (d) pyramid blend.

Figure 3.42 Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM. The first three rows show the high, medium, and low-frequency parts of the Laplacian pyramid (taken from levels 0, 2, and 4). The left and middle columns show the original apple and orange images weighted by the smooth interpolation functions, while the right column shows the averaged contributions.

Next

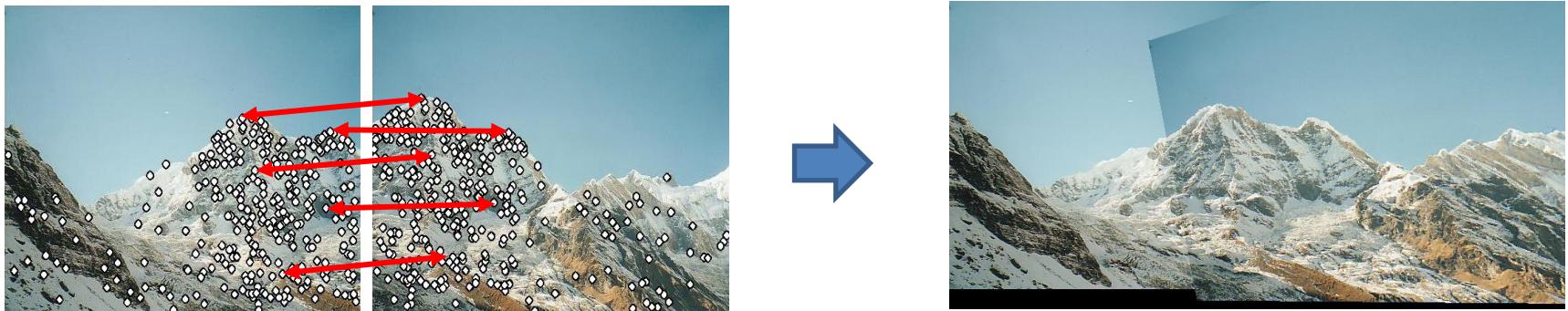
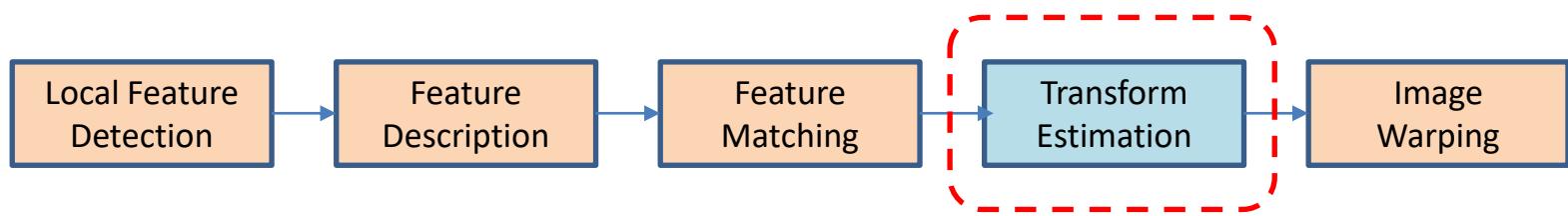


Image Feature Information Extraction for Interest Point Detection: A Comprehensive Review

Junfeng Jing, Tian Gao, Weichuan Zhang, *Member, IEEE*, Yongsheng Gao, *Senior Member, IEEE*, Changming Sun

Abstract—Interest point detection is one of the most fundamental and critical problems in computer vision and image processing. In this paper, we carry out a comprehensive review on image feature information (IFI) extraction techniques for interest point detection. To systematically introduce how the existing interest point detection methods extract IFI from an input image, we propose a taxonomy of the IFI extraction techniques for interest point detection. According to this taxonomy, we discuss different types of IFI extraction techniques for interest point detection. Furthermore, we identify the main unresolved issues related to the existing IFI extraction techniques for interest point detection and any interest point detection methods that have not been discussed before. The existing popular datasets and evaluation standards are provided and the performances for fifteen state-of-the-art approaches are evaluated and discussed. Moreover, future research directions on IFI extraction techniques for interest point detection are elaborated.

Index Terms—Interest point detection, image feature information extraction, taxonomy, performance evaluation, development trend on image feature information extraction.

1 INTRODUCTION

INTEREST points have been found very important in human perception of shapes [1] and have been one of the key image features which are widely used as critical cues for various image processing and image understanding tasks such as camera calibration [2], image registration [3], object recognition [4], 3D reconstruction [5], human-computer interaction [6], autonomous driving [7], face recognition [8], traffic analysis [9], multi-object detection and tracking [10], image-guided robotic-assisted surgery [11], surveillance [12], and augmented reality [13].

Although there exist many interest point detection surveys that are on the problems of specific interest point detections, such as blob detection [14], local invariant feature detection [15], interest point detection for visual tracking [16], edge contour based corner detection [17], interest point detection for image matching [18], 3D keypoint detection [19], and local feature detection for 3D reconstruction [5], our investigation indicates that the existing surveys on interest point detection do not accurately describe the characteristics, advantages, and disadvantages of the interest point detection algorithms (e.g., the Harris method [20] and the Harris-Laplace

method [21]). Our research indicates that the key issue for image interest point detection is how to directly or indirectly extract and analyze image feature information (IFI) such as first-order, second-order, or high-order intensity variation information, image frequency variation information, image local patterns (e.g., edges or contours), or high level feature representations obtained from machine learning based network architectures. The quality of the IFI extracted from an image directly affects the performance of interest point detection (e.g., the repeatability of interest point detection under image affine transformations). Furthermore, there has been active developments on image interest point detection in the past decade. However, there lacks a comprehensive survey on IFI extraction techniques for interest point detection. In this paper, we intend to fill this gap, and a taxonomy of the IFI extraction techniques for interest point detection is proposed. According to this classification method, we discuss different types of IFI extraction techniques in depth, and put forward our own views on the different IFI extraction techniques in the literature.

The number of papers on interest point detection is overwhelming. Because of this, compiling any exhaustive review of the approaches is out of the scope of a paper with a reasonable length. As a result, we have limited our focus on 2D image interest point detection methods that were published in top quality journals or conferences and are highly cited. It is worth to note that the existing blob detection techniques include two categories [22]: interest point detection [23] and interest region detection [24], [25]. Considering that our theme is on 2D point like features, this paper only discusses the point feature detection category on blob detection techniques, and it does not discuss the region based blob detection techniques.

The highlights of this article comprise five aspects. First, a taxonomy of the existing IFI extraction techniques

- J. Jing is with College of Electrical and Information, Xi'an Polytechnic University, Xi'an, 710048, China.
E-mail: jingjunfeng0718@sina.com;
- T. Gao is with School of Computer Science and Engineering Nanjing University of Science and Technology, Nanjing, 210094, China.
E-Mail: gaotian970228@163.com
- W. Zhang is with the Institute for Integrated and Intelligent Systems, Griffith University, QLD, Australia, and also with CSIRO Data61, PO Box 76, Epping, NSW 1710, Australia.
E-mail: zwc2003@163.com
- Y. Gao is with the Institute for Integrated and Intelligent Systems, Griffith University, QLD, Australia.
E-mail: yongsheng.gao@griffith.edu.au
- C. Sun is with CSIRO Data61, PO Box 76, Epping, NSW 1710, Australia.
E-mail: changming.sun@csiro.au

for interest point detection is presented. Second, different types of IFI extraction techniques for image interest point detection are analyzed and summarized. Meanwhile, the properties of the existing IFI extraction techniques for interest point detection are analyzed in detail. Third, comprehensive performance evaluations are used for performance testing for fifteen state-of-the-art algorithms. Fourth, the results of the experiments are systematically analyzed and summarized. Fifth, the development trend on IFI extraction for interest point detection is presented.

The remainder of the article is arranged as follows. In Section 2, related background and main challenges on interest point detection are summarized. Section 3 presents a taxonomy of the existing IFI extraction techniques for image interest point detection. In terms of this taxonomy, we discuss different types of IFI extraction techniques in depth and put forward our own views on different IFI extraction techniques in the literature. In Section 4, the existing popular datasets and evaluation standards are summarized and the performances for the fifteen state-of-the-art approaches are discussed. In Section 5, the development trend on IFI extraction for interest point detection is elaborated. Finally, conclusions are presented in Section 6.

2 GENERIC INTEREST POINT DETECTION

In the following, we firstly provide a brief background on the definition of an interest point. Then the main challenges on interest point detection are presented.

2.1 Background

It is indicated in [26] that interest points include not only junctions, corners, and blobs, but also locations with significant texture variations. In fact, the terms ‘keypoint’ [27], ‘dominant point’ [28], ‘junction’ [29], ‘critical point’ [30], ‘corner’ [20], ‘blob’ [31], [32], and ‘saliency feature’ [33] are special subclasses of ‘interest point’. Up to now, scholars defined point features from the perspective of corners [34], [35], blobs [31], and interest points [21], and these definitions are shown in Fig. 1. Furthermore, the existing classifications of corners fall into three categories: intensity-based [36], [37], contour-based [1], [38], and template-based [34], [39]. As illustrated in Fig. 1, the existing definitions of interest points include fuzzy concepts and precise mathematical definitions. There are contradictions between some definitions. For example, Moravec [36] defined that corners are the points where the intensity variations are large in all orientations. However, Zhang and Sun [37] defined that corners are the points where the intensity variations are large in most orientations, not necessarily in all orientations. Following these different definitions, different types of interest point detection methods extract IFI based on what they require from images and determine the interest points according to the extracted IFI.

Meanwhile, our investigation indicates that Lindeberg’s definition [40] on interest points is relatively comprehensive and has the following properties: (1) It has a clear and preferred mathematically based definition. (2) The interest point position is accurate in the image domain. (3) The

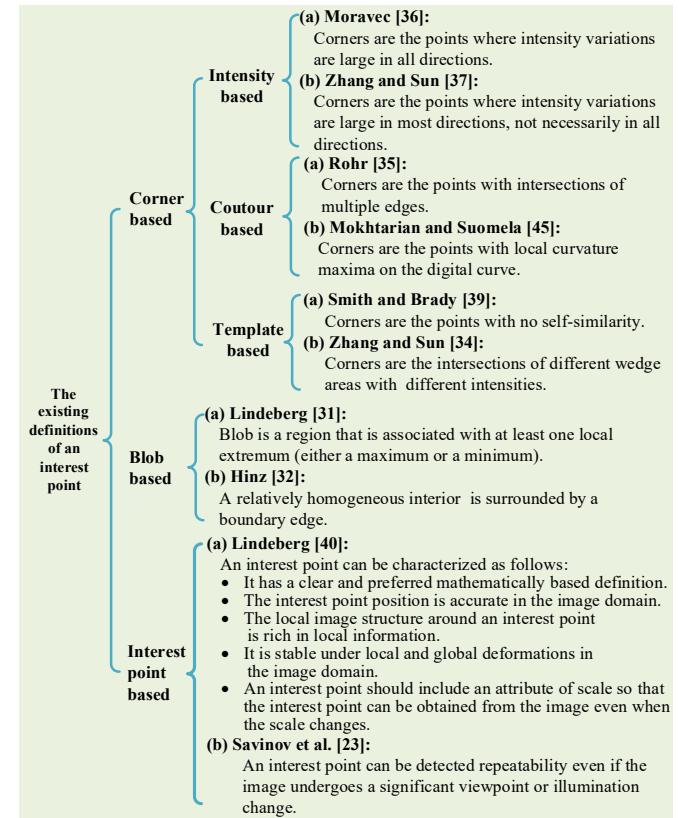


Fig. 1 Definitions of interest points.

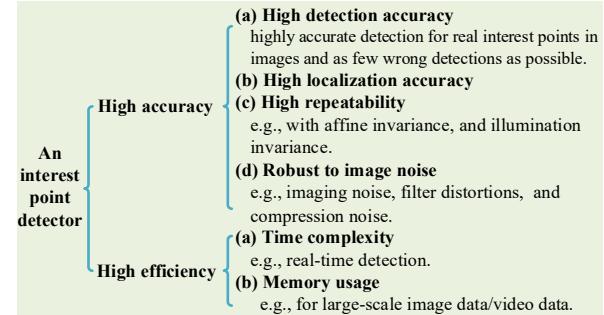


Fig. 2 Main challenges for interest point detection.

local image structure around an interest point is rich with local information. (4) It is stable under local and global deformations in the image domain. (5) An interest point should include an attribute of scale so that the interest point can be obtained from the image even when the scale changes.

2.2 Main challenges

The ultimate goal of image interest point detection is to develop a general method that achieves both high accuracy/quality and high efficiency as shown in Fig. 2.

On the one hand, high accuracy/quality interest point detectors should have the ability to accurately localize and detect interest points from images. It is well known that the reason why image interest point detection has been a research hot topic is that interest point detection has a wide range of applications in the field of image processing and computer vision. Most interest point based computer

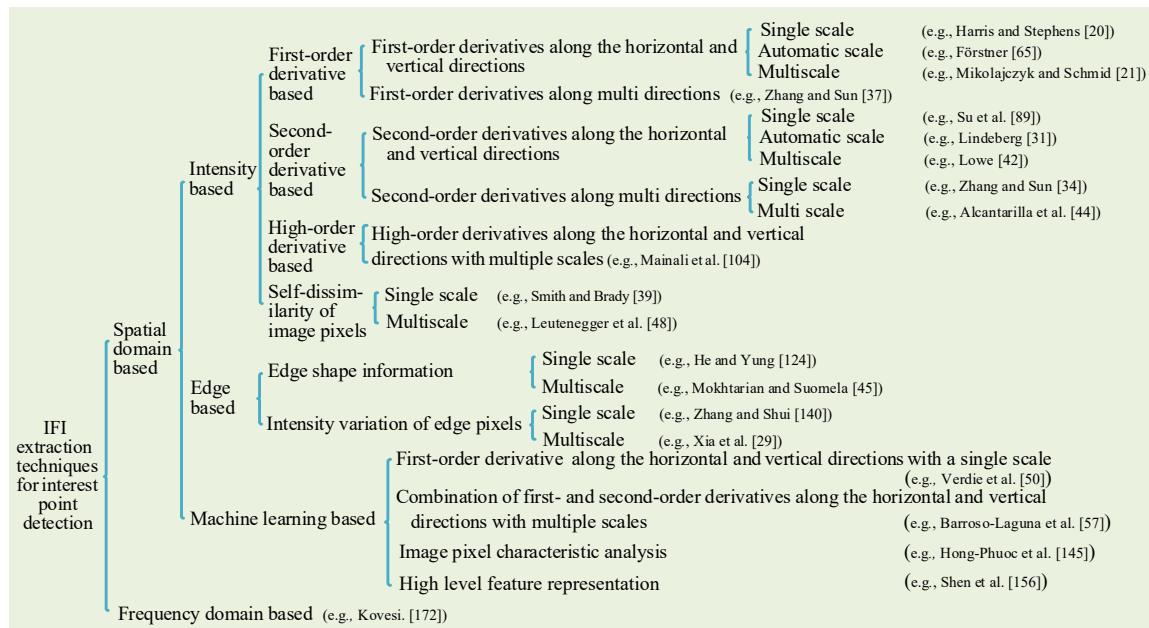


Fig. 3 Classification of the IFI extraction techniques in the existing interest point detection methods.

vision tasks rely on interest point detectors for finding local feature or interest point correspondences between different images for object tracking [4] and 3D reconstruction [5]. For this purpose, high accuracy/quality interest point detectors should have high repeatability under different imaging conditions such as lighting directions, illuminations, camera poses, resolutions, blurs, and weather conditions. Other challenges for interest point detection come from digital artifacts, noise, and filtering distortion. On the other hand, high accuracy/quality interest point detectors should have the ability to detect interest points in real-time with acceptable memory and storage demands. The reason is that real-time processing is an indispensable requirement in many interest point based computer vision tasks such as autonomous driving [7] and image-guided robotic-assisted surgery [11], [41].

Our investigation also indicates that more accurate extraction of IFI makes interest point detection algorithms more likely to achieve better accuracy/quality, but it increases the complexity of the algorithms. This is a dilemma that still cannot be reconciled. This issue will be discussed in detail later in the paper.

3 IMAGE FEATURE INFORMATION EXTRACTION TECHNIQUES FOR INTEREST POINT DETECTION

In this section, we first introduce a taxonomy of the existing IFI extraction techniques for image interest point detection. According to this classification method, we discuss different types of IFI extraction techniques for interest point detection in depth. The representative interest point detection methods are shown in Fig. 4.

3.1 A taxonomy on existing IFI extraction techniques for interest point detection

The existing IFI extraction techniques for interest point extraction can be classified into two main categories:

the spatial domain and the frequency domain based IFI extractions as illustrated in Fig. 3. In addition, the spatial domain based IFI extraction techniques fall into intensity based, edge based, and machine learning based IFI extractions. Intensity based IFI extraction techniques extract image intensity variation information directly from images. Edge based IFI extraction techniques firstly detect edges from an input image by an edge detection algorithm (e.g., the Canny algorithm [54] and the line segment detector [55]), and then the shape or local structure information on edge pixels is obtained. The existing machine learning based IFI extraction techniques extract the first- or second-order derivatives information from images [56], [57], or they learn high-level image abstractions from a designed network architecture. Frequency domain based IFI extraction techniques firstly transform the image from spatial domain to frequency domain, and then local structure information in the image is extracted in frequency domain.

There is a considerable degree of correlation between the aforementioned IFI extraction techniques given in Fig. 3 and the definitions of interest points shown in Fig. 1. In the following, we describe in detail how different IFI extraction techniques detect corners, blobs, and interest points from images according to the classification of the IFI extraction techniques.

3.2 Intensity based IFI extraction techniques for interest point detection

The existing intensity based IFI extraction techniques can be divided into first-order derivative based, second-order derivative based, high-order derivative based, and self-dissimilarity on image pixels based approaches.

3.2.1 First-order intensity variation based interest point detection

The existing first-order derivative based IFI extraction techniques for interest point detection include: first-order

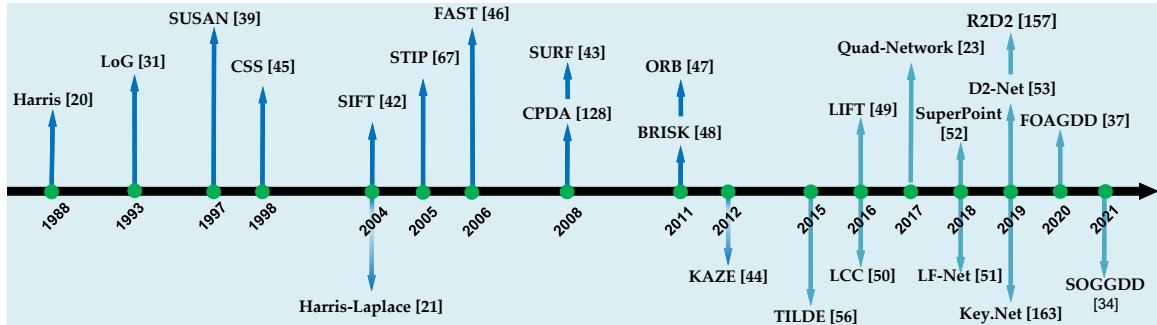


Fig. 4 Representative interest point detection methods. Starting from the Harris method [20], first-order intensity variation based interest point detection methods gained remarkable popularity. In 1993, Lindeberg [31] presented a Laplacian of Gaussian method for detecting blobs from images in a multi-scale space. Along this approach, second-order intensity variation based blob detection methods (e.g., Harris-Laplace [21], SIFT [42], SURF [43], and KAZE [44]) were proposed for achieving blob detection. In 1997, Smith and Brady [39] proposed the SUSAN method which marks the beginning of self-dissimilarity based interest point detection. In 1998, Mokhtarian and Suomela [45] proposed the CSS method which represents the edge contour based interest point detection methods and it received considerable popularity. In 2006, Rosten et al. [46] proposed the FAST detector which is one of the first machine learning methods for interest point detection. Subsequently, the ORB [47] and BRISK [48] algorithms are optimized for the FAST detector [46]. In 2016, Yi et al. [49] proposed the LIFT detector which is one of the first deep learning methods for interest point detection. And then, LCC [50], Quad-Network [23], LF-Net [51], SuperPoint [52], and D2-Net [53] were proposed by Lenc and Vedaldi, Savinov et al., Ono et al., Zhang and Rusinkiewicz, and Dusmanu et al. respectively further promoting the development of deep learning based interest point detection. In 2020 and 2021, FOAGDD [37] and SOGGDD [34] were proposed by Zhang and Sun which proved for the first time that first-order and second-order intensity variation information along multiple directions contributes significantly in improving the performance of interest point detection.

horizontal and vertical directional derivatives based, multi-direction first-order directional derivatives based, gradient amplitude information based, and gradient direction information based methods.

Following Moravec's definition [36] that corners are the points with large variation in intensity in each direction, Harris and Stephens [20] constructed a 2×2 structure tensor for finding corners which have large variations on image intensities in the horizontal and vertical orientations. In [20], the input image $I(x, y)$ was smoothed by an isotropic Gaussian filter $g_\sigma(x, y)$ with a scale factor σ ($\sigma > 0$)

$$h(x, y) = I(x, y) \otimes g_\sigma(x, y), \quad (1)$$

where (x, y) represents a pixel coordinate in the image. And then the first-order horizontal and vertical (orthogonal) directional derivatives ($h_x(x, y)$ and $h_y(x, y)$) of $h(x, y)$ are derived to construct a 2×2 structure tensor S

$$S = \begin{bmatrix} h_x^2(x, y) & h_x(x, y)h_y(x, y) \\ h_x(x, y)h_y(x, y) & h_y^2(x, y) \end{bmatrix}. \quad (2)$$

In [20], the attribute of each image pixel (x, y) is determined by the eigenvalues λ_1 and λ_2 ($\lambda_1 < \lambda_2$) of structure tensor S . (1) If both λ_1 and λ_2 are close to zero, the pixel belongs to a flat region. (2) If λ_1 (or λ_2) is close to zero and λ_2 (or λ_1) is a large positive value, the pixel is an edge point. (3) If both λ_1 and λ_2 are large positive values, the pixel is a corner. To reduce the complexity of the algorithm, the determinant of the structure tensor $\text{Det}(S)$ and the trace of the structure tensor $\text{Trace}(S)$ are applied to construct the Harris corner measure as follows

$$H_a = \text{Det}(S) - k \times \text{Trace}^2(S), \quad (3)$$

where k is a constant (e.g., $k=0.04$). From Equation (2), it can be derived that if both the first-order horizontal and vertical directional derivatives ($h_x(x, y)$ and $h_y(x, y)$) are large enough, the Harris method has the ability to detect corners very well.

Following the Harris method, many different filters (e.g., Gaussian filter [58], anisotropic nonlinear diffusion [59], and Box filter [60]) with a single scale were used to extract image intensity variation information along a pair of orthogonal directions for detecting corners [58]–[65]. Shi and Tomasi [58] indicated that it is better to utilize the small eigenvalue of structure tensor S as the corner measure for detecting corners from an input image. For avoiding the influence of Gaussian image smoothing on the location of a feature, Brox et al. [59] replaced the isotropic Gaussian filter in the Harris method with an anisotropic nonlinear diffusion for detecting interest points. Loog and Lauze [64] combined the corner measure of the Harris method and visual salience theory [66] for detecting interest points from an input image. Mainali et al. [60] replaced the Gaussian filter in the Harris method with a Box filter to decrease the algorithm complexity of the Harris method.

Meanwhile, many algorithms extended the framework of the Harris algorithm from a single scale to an adaptive scale or multi-scales for improving the accuracy performance of corner detection. Laptev and Lindeberg [67] extended the Harris method in the spatial domain into the space-temporal domain, and the first-order orthogonal directional derivatives in the spatial domain and derivative in the temporal domain are used to construct a space-temporal second-moment matrix for detecting interest points in the scale space in video. Mikolajczyk and Schmid [21] extended the single-scale Harris detector to a

corner detector with adaptive scale selection. Furthermore, they proposed a scale invariant Harris-Laplace interest point detection method by combining the Harris corner detection method [20] with the normalized Laplace operator [68]. Gao et al. [69] argued that it is appropriate to detect corners by using filters that are consistent with the human visual system. Then the log-Gabor wavelets with multiple scales were employed to extract multi-scale first-order orthogonal directional derivatives from an input image and construct multi-scale 2×2 structure tensors for detecting corners. In view of the shearlets for having the ability to efficiently obtain the anisotropic intensity variation information in images, Duval-Poo et al. [70] applied the multi-scale shearlet filter to obtain IFI along a pair of orthogonal directions for detecting corners.

In addition, first-order orthogonal directional derivatives based IFI extraction techniques are often extended for detecting interest points in color or other type of images. Weijer et al. [71] extended the photometric invariance theory [72] and presented a novel class of derivatives which is denoted as photometric quasi-invariants for improving the robustness of the obtained feature information. Then a corner detection method was proposed for color images in the framework of the Harris method. Ruzon and Tomasi [73] utilized the earth mover's distance (EMD) technique [74] to identify the discrepancies of the color distributions for improving the accuracy of image corner detection. Weijer et al. [75] presented a salient point detector based on the distinctiveness of the local color information. The first-order orthogonal directional derivatives of the color image are used to measure the color distinctiveness [14] of a pixel. If the distinctiveness of the color derivatives of the pixel in a given neighborhood is the local maximum and larger than a given threshold, the pixel is marked as a salient point. Kenney et al. [76] combined the Harris method and an optical flow estimation technique for detecting corners in multi-channel images. In order to improve the robustness of corner detection in different imaging conditions, Stöttinger et al. [77] extended the Harris method, and the multi-scale image intensity variation information along a pair of orthogonal directions is used for selecting light invariant corners in an arbitrary color space.

In [37], Zhang and Sun proved that first-order horizontal and vertical directional derivatives based interest point detection methods cannot accurately detect interest points. Furthermore, they indicated that the first-order anisotropic Gaussian directional derivatives (FOAGDDs) of a step edge has only one local maximum and one local minimum. While the FOAGDDs of different types of corners (e.g., L-type, Y- or T-type corner, X-type corner, and star-type corner) have different numbers of local maxima and local minima. Finally, the multi-scale FOAGDDs of the image are used to construct a multi-directional structure tensor at multiple scales for detecting corners. Wang et al. [78] followed the FOAGDD method [37] for corner detection. In their method, an input image is smoothed by the multi-directional shearlet filters with multiple scales, and a multi-directional structure tensor at multiple scales are constructed for detecting corners.

It is worth to note that gradient amplitude or gradient direction information can also be used to detect interest

points in high local symmetry/mirror regions in images, because the center of a circular blob feature has a high degree of local mirror symmetry. Reisfeld et al. [79] applied the concept of symmetry to compute a symmetry map from an input image. For each pixel, the symmetry strength is obtained by looking at the magnitude and the direction of the derivatives of neighboring pixels. Pixels with high symmetry are marked as interest points. Heidemann [80] extended the method in [79] and interest points are detected in color images. Loy and Zelinsky [81] proposed a fast interest point detection algorithm and used the information on the gradient direction distribution of pixels to detect interest points in locally radial symmetric regions in the image. Maver [82] exploited the concept of self-similarity to detect interest points associated with symmetrical regions in images in scale space.

3.2.2 Second-order derivative based interest point detection

The existing second-order derivative based IFI extraction techniques for interest point detection include: second-order horizontal and vertical directional derivatives based and multi-direction second-order directional derivatives based methods.

In [83], corresponding to Equation (1), the second-order horizontal, vertical, and cross derivatives ($\hbar_{xx}(x, y)$, $\hbar_{yy}(x, y)$, and $\hbar_{xy}(x, y)$) of $\hbar(x, y)$ are derived to construct an Hessian matrix

$$\mathbf{H} = \begin{bmatrix} \hbar_{xx} & \hbar_{xy} \\ \hbar_{xy} & \hbar_{yy} \end{bmatrix}. \quad (4)$$

Interest points are defined as the local extremum (either a maximum or a minimum) of the determinant of the Hessian

$$C = |\mathbf{H}(\hbar)| = \hbar_{xx}\hbar_{yy} - (\hbar_{xy})^2. \quad (5)$$

Following the Beaudet method [83], second-order derivative based IFI extraction techniques were widely used for interest point detection.

Lindeberg [31] defined a blob as a region associated with at least one local extremum (either a maximum or a minimum) and then a multi-scale Laplacian of Gaussian (LoG) filter is used to extract the second-order horizontal and vertical directional derivatives from an input image for detecting blobs. In [68], Lindeberg utilized the normalized Laplace operator with adaptive scales for detecting corners. Abdeljaoued and Ebrahimi [84] followed the scale-space theory [85] to detect interest points as

$$K_{\text{norm}}(x, y, \sigma) = \sigma^2(\hbar_{xx}\hbar_y^2 + \hbar_{yy}\hbar_x^2 - 2\hbar_{xy}\hbar_x\hbar_y), \quad (6)$$

which aims to detect interest points with different scales in scale space by the normalized operator. In order to reduce computational complexity on obtaining LoG [31] and efficiently construct a scale space pyramid, Lowe [42] proposed the scale invariant feature transform (SIFT) method which approximated the normalized LoG filter by a difference of Gaussian (DoG) filter

$$D(x, y, \sigma) = (g(x, y, q\sigma) - g(x, y, \sigma)) \otimes I(x, y), \quad (7)$$

where q is a constant. The candidate interest points are detected by seeking for local maxima in a DoG pyramid.

Meanwhile, the eigenvalues (λ_{\max} , λ_{\min} , with $\lambda_{\max} > \lambda_{\min}$) of the Hessian matrix (the second-order derivative extracted by DoG) are applied to suppress edge responses

$$\begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix},$$

where D_{xx} and D_{yy} denote the second-order orthogonal directional derivatives of $D(x, y)$, and D_{xy} denotes the second-order cross derivative of $D(x, y)$. For each candidate interest point, if the ratio of the two eigenvalues (λ_{\max} and λ_{\min}) is larger than ς (e.g., $\varsigma=6$), i.e.,

$$\frac{\lambda_{\max}}{\lambda_{\min}} > \varsigma, \quad (8)$$

the candidate interest point may be marked as an edge point.

Following the LoG [31] and DoG [42] methods, different types of filters (e.g., piece-wise triangle filters [86] or Box filters [87], [88]) were used to obtain the second-order derivatives along a pair of orthogonal directions with multiple scales or adaptive scales from an input image for detecting interest points. Bay et al. [43] aimed to reduce the computational complexity in image convolution of the DoG method [42]. Box filters [87] and integral images are utilized to obtain a similar second-order horizontal and vertical directional derivatives for detecting interest points from images in scale space. To improve the accuracy and repeatability of interest points detection, Marimon et al. [86] applied piece-wise triangle filters to approximate the second-order Gaussian derivatives along a pair of orthogonal directions with multiple scales for searching extrema in the scale space. Su et al. [89] applied the second-order Gaussian horizontal and vertical directional derivatives to construct an Hessian matrix for detecting junctions in blood vessel images.

Alcantarilla et al. [44] indicated that a Gaussian filter with the same degree of smoothing on details and noise in images reduces the localization accuracy of detected interest points, and they proposed a KAZE interest point detector. Additive operator splitting schemes [90] are used to approximate the Perona and Malik diffusion equation [91] and the local maxima of the Hessian in a nonlinear scale space are used for detecting interest points. Later, an accelerated KAZE method [92] was presented to reduce the computational complexity of the KAZE method [44]. In [93], Mainali et al. proved that the cosine modulated Gaussian filter has a balanced scale-space atom with minimal spread in both scale and space. Then the designed filter is employed to obtain the second-order horizontal and vertical directional derivatives from an input image for detecting interest points with an adaptive scale in the framework of the LoG method [31]. Furthermore, edges are eliminated using the condition given in (8) from detected interest points. To improve the robustness of interest point detection in abrupt variations of images caused by illumination changes or geometric transformations, Miao and Jiang [94] utilized the weighted rank order filter to replace the LoG filter to detect interest points in the framework of the LoG method. In [27], Salti et al. demonstrated that the multi-scale wave equation [95] can be effectively used to detect keypoints with significant symmetries at different scales.

Then the multi-scale wave equation [95] was utilized to obtain the second-order derivatives from an input image for finding local extrema points as keypoints in scale space. Li et al. [96] utilized the geometric algebra theory [97] to detect interest points in multispectral images in the framework of the DoG method. In order to improve the accuracy of interest point detection in low contrast images, Miao et al. [98] utilized the zero-norm LoG filter for detecting interest points from images. Because shearlet filters have good IFI extraction ability and noise robustness, Duval-Poo et al. [99] employed the multi-scale shearlet filters to detect blobs in the framework of the DoG method.

Alternately, multi-directional second-order derivatives with multiple scales were applied to detect interest points from an input image. Kong et al. [100] applied multi-directional generalized (isotropic and anisotropic) LoG filters to smooth the input image in scale space, and then the amplitude responses in different directions are multiplied by an empirical coefficient. Then the sum of the amplitude responses in different directions are used to detect blobs in scale space. Li and Shui [101] followed the Kong et al. [100] method for detecting blobs from an input image, and edge suppression using the condition in (8) is applied for edge points removal from detected blobs. Wang et al. [102] presented a normalized method for the LoG filters [31] and then the multi-directional normalized filters with multiple scales are used to obtain the multi-directional second-order derivatives from an input image. Then the smallest directional derivative in multiple directional derivatives is used to detect blobs in scale space. Ghahremani et al. [103] introduced in detail how the DoG method [42] should set multi-scale parameters. Then a novel discrete multi-scale pyramid is designed using the proposed scale-space blurring ratio and smoothness of the pyramid for improving the accuracy of interest point detection. Zhang and Sun [34] applied the second-order generalized Gaussian directional derivative (SOGGDD) filters to derive the SOGGDD representations of a step edge, L-, Y-, X-, and star-type corners. The SOGGDD representations of a step edge and different types of corners indicate that the SOGGDD of a step edge is zero in each direction and the SOGGDD filters along a pair of orthogonal directions cannot obtain enough IFI to accurately describe the second-order directional derivative at corners. Take an L-type corner and a step edge as examples as illustrated in Fig. 5, it can be seen that the second-order isotropic Gaussian directional derivative (SOIGDD) of the corner is very large in many directions and the SOIGDD of the step edge is zero in all orientations. Meanwhile, it can be seen from Fig. 5 that the SOIGDDs of the corner along the orthogonal directions are zero. It means that the ratio of the two eigenvalues of the 2×2 Hessian matrix cannot be used to properly describe the attribute of an image pixel. It is indispensable to extract the second-order derivative with multiple directions for detecting corners.

3.2.3 High-order derivative based interest point detection

Mainali et al. [104] proved that the 10th-order multi-scale Gaussian derivative filter reaches the jointly optimal Heisenberg's uncertainty principle [105] of its impulse response in scale and space domains simultaneously. Then,

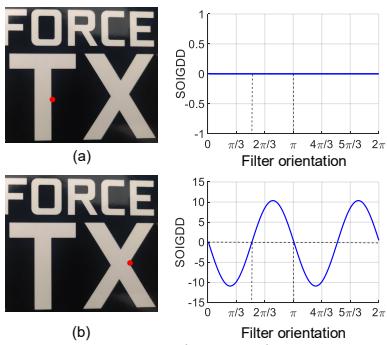


Fig. 5 A point on a step edge and an L-type corner (the red dots) are shown in (a)-(b) in the first column. Their corresponding second-order isotropic Gaussian directional derivatives (SOIGDDs) are shown in the second column.

the 10th-order horizontal and vertical Gaussian directional derivatives are obtained from an input image for detecting interest points. Furthermore, a 2×2 matrix based on the 10th-order derivatives is constructed which is similar to condition (8) for eliminating edge pixels from the detected interest points. In order to improve the accuracy of interest point detection when the input image is blurred, Saad and Hirakawa [106] extended the SIFT [42] method in which Gaussian filters with multiple scales are used to obtain the 4th-order Gaussian horizontal and vertical directional derivatives from an input image for detecting interest points in scale space.

3.2.4 Self-dissimilarity of image pixels based interest point detection

It is well-known that the apex of a wedge is not self-similar. The apex of the wedge corresponds to a corner point. Given the size of a circular mask, the similarity ratio information between the gray values at the pixels within the circular mask and the gray value at the center pixel of the template can be used to measure the self-similarity of the center pixel.

Smith and Brady [39] used the similarity ratio information for detecting corners from an input image where a corner is defined as a point with smallest univalue segment assimilating nucleus (SUSAN). Trajkovic and Hedley [107] extended the SUSAN method, and the corner measure of an image pixel is defined as minimum intensity variations over all possible orientations. If the corner measure is higher than a given threshold, the image pixel is marked as a corner. Self-dissimilarity of image pixels can also be measured by local entropy. Cazorla et al. [108] applied the SUSAN method to detect candidate corners, and then a probabilistic edge model and a log-likelihood test are combined for determining junction points. Following the SUSAN method [39], Rosten et al. [46], [109] presented the features from the accelerated segment test (FAST) algorithm. The pixel value differences between a target point and a circular neighborhood template are utilized for detecting corners from an input image. Bennett and Lasenby [110] proposed a method for detecting corners on a chess-board. A ring of 16 pixels around each pixel in a chess-board image are sampled at a constant radius with equal angular spacing. The gray values of these 16 pixels are analyzed to determine whether the center pixel belongs

to an edge, a corner, or a flat region. Bok et al. [111] utilized the special characteristics of corners on a chess-board (e.g., four black-white gray value changes on a circular boundary around a corner) to automatically localize corners, and gradient slopes are used to discard false corners.

It was indicated in [48] that the FAST method does not account for scale variations, and a scale space modification of the FAST method is employed for detecting corners. Rublee et al. [47] applied the multi-scale pyramidal representation [42] into the FAST method and corners are detected in scale space. Furthermore, the Harris method is employed for edge rejection. Buoncompagni et al. [112] presented a keypoint detection algorithm based on the FAST method and the BRIEF descriptor [113]. If candidate points detected by the FAST method have high repeatability in target detection and image matching, these candidates are marked as keypoints; otherwise, they are deleted.

3.3 Edge contour based IFI extraction techniques for interest point detection

The existing edge contour based IFI extraction techniques fall into two categories: edge shape based and intensity variation on edge pixels based techniques.

3.3.1 Edge shape based interest point detection

In general, the existing edge shape-based interest point detection methods can be divided into two categories [114]: direct curvature estimation methods and indirect curvature estimation methods.

Direct curvature estimation techniques detect corners by estimating the cosine of angle, local curvature, or tangential deflection at each point on edge contours. In the early developed algorithms, angular change [115]–[117] and isolated discontinuity points in the mean slope [118] were used as measures for detecting corners. Rattarangsi and Chin [119] analyzed the curvature characteristics of the L-, END-, and STAIR-models in the Gaussian scale space and developed a multiple-scale corner detection method. Mokhtarian and Suomela [45] presented a corner detection method with multi-scale based on the curvature scale-space (CSS) representations [120], [121]. For a given parametric vector equation $\Gamma(l) = \{u(l), v(l)\}$ of a planar curve, the Gaussian filter with multiple scales $g(l, \sigma)$ are employed to smooth the planar curve. Then the multi-scale curvature $\kappa(l, \sigma)$ on the smoothed curve is

$$\kappa(l, \sigma) = \frac{\dot{U}(l, \sigma)\ddot{V}(l, \sigma) - \ddot{U}(l, \sigma)\dot{V}(l, \sigma)}{\left(\dot{U}^2(l, \sigma) + \dot{V}^2(l, \sigma)\right)^{3/2}}, \quad (9)$$

with $\dot{U}(l, \sigma) = u(l) \otimes \dot{g}(l, \sigma)$, $\dot{V}(l, \sigma) = v(l) \otimes \dot{g}(l, \sigma)$, $\ddot{U}(l, \sigma) = u(l) \otimes \ddot{g}(l, \sigma)$, and $\ddot{V}(l, \sigma) = v(l) \otimes \ddot{g}(l, \sigma)$. \otimes refers to convolution, σ is the scale factor, and $\dot{g}(l, \sigma)$ and $\ddot{g}(l, \sigma)$ are the first- and second-order derivatives of $g(l, \sigma)$.

Inspired by the method in [119], Zhong and Liao [122] presented a theoretical analysis on the direct curvature scale space (DCSS) for detecting corners on edge contours. In order to improve the peak value of a corner response while suppressing noise through Gaussian smoothing, Zhang et al. [123] extended the CSS method [45] with the application of multi-scale curvature product of pixels

for detecting corners. In order to eliminate false corners in the corner set, He and Yung [124] extended the CSS method [45] by applying an adaptive local threshold based on the mean curvature within a region of support. Pham et al. [125] proposed an algorithm to detect corners by finding the points that are intersections of multiple edges. Zhang et al. [126] replaced the multi-scale Gaussian filtering [119] by Laplacian of Gaussian to analyze the behaviors of curvatures of the L-, END-, and STAIR-models in a multi-scale space for detecting corners. Zhang et al. [38] identified that the existing curvature calculation [45] is not robust to local contour shape change and noise in the discrete domain and has poor accuracy for corner detection when corners are closely located. And then discrete curvature representations of the L-, END-, and STAIR-models are derived and investigated for detecting corners. For improving the computation efficiency, Wang et al. [127] applied a difference operation to approximate the intensity derivative for calculating the contour curvature and defined the local maximum point as a corner point.

Meanwhile, indirect curvature estimation techniques have also been widely used for detecting corners on edge contours. Zhu and Chirlian [30] indicated that existing discrete curvature calculation methods have quantization errors. Therefore, shape representation (e.g., for a T-type corner), shape characteristics analysis, and shape recognition are used for detecting corners on edge contours. In order to enhance the robustness for corner detection with image affine transformations, Awrangjeb and Lu [128] applied the chord-to-point accumulation (CPDA) technique [129] for calculating the discrete curvature and detecting corners on edge contours. Zhang et al. [130] presented an edge curve based corner detector where the derivative information between the position of a point on the edge curve and the position of adjacent points are used to construct a gradient correlation matrix (GCM) for detecting corners. Teng et al. [131] extended the CPDA method [128] and the ends of a given chord and the midpoint on the curve segment between the ends of the chord can be used to form a triangle. The ratio of the chord length to the sum of the other two arm lengths of the triangle is computed for detecting corners on edge contours. Mustafa et al. [132] applied the multi-scale bilateral filters to segment an input image and then corners are detected at the intersection of the borders of three or more regions.

3.3.2 Intensity variation of edge pixels based interest point detection

Interest point detection algorithms based on intensity variations on edge pixels determine interest points by analyzing the intensity variation on edge pixels. Kitchen and Rosenfeld [133] used the first- and second-order intensity derivative of an edge pixel (x, y) as a corner measure $\mathfrak{J}(x, y)$

$$\begin{aligned} \mathfrak{J}(x, y) = & I_{xx}(x, y)I_y^2(x, y) + I_{yy}(x, y)I_x^2(x, y) \\ & - 2I_{xy}(x, y)I_x(x, y)I_y(x, y), \\ \mathfrak{S}(x, y) = & \frac{\mathfrak{J}(x, y)}{I_x^2(x, y)I_y^2(x, y)}. \end{aligned} \quad (10)$$

In order to localize junctions and specify their corresponding orientations, Elias and Laganière [134] used the consistency of the gradient directions on edge pixels for detecting corners. Azzopardi and Petkove [135] utilized the weighted geometric mean of the blurred and shifted responses of the selected Gabor filters for detecting keypoints on contours. To improve the affine robustness of the algorithm, Shui and Zhang [136] employed the anisotropic Gaussian directional derivative (ANDD) filters [137] to derive the ANDD representations of the L-, Y-, X-, and star-type corners. Then the differences of the geometric shapes of the directional derivatives between edges and corners are used for detecting corners on edge contours. To improve detection accuracy, Xia et al. [29] applied a contrario method [138] based on the statistical model of gradient information in the neighborhood of feature points to replace the method about constructing a linear scale space to obtain the scale information of junctions. To identify the discrepancies between corners and edges on edge contours, Zhang et al. [139] used the magnitude responses of the imaginary parts of the Gabor directional derivative filters to detect corners. In order to improve the noise robustness of the algorithm, Zhang and Shui [140] applied the distribution differences of gradient directions at edge pixels and corners for detecting corners on edge contours. In order to obtain stronger affine robustness and estimate the geometric structure of a junction, Xue et al. [141] extended the ACJ method [29] by applying junction branch length estimation and constructing an anisotropic nonlinear scale space.

3.4 Machine learning based IFI extraction techniques for interest point detection

The existing machine learning based IFI extraction techniques for interest point detection include: first-order derivative based, first- and second-order derivative combination based, image pixel characteristics analysis based, and high level feature representation based methods.

Rosten et al. [46], [109] followed the SUSAN method [39] and proposed the features from the accelerated segment test (FAST) algorithm based on the comparison of pixel values between a target point and a circular neighborhood template. Meanwhile, the decision tree technique [142] was applied for speeding up corner detection. Lepetit and Fua [143] extended the FAST method where the LoG filters [31] were used to detect candidate interest points in scale space, and then the randomized trees technique [142] was used to select interest points from the candidates. Trujillo and Olague [144] applied the random combination of forty-six basic operators (e.g., first- and second-order horizontal and vertical directional derivatives, autocorrelation matrix, and curvature) in the framework of the genetic programming technique [33] for training an interest point detector. The detector was required to have a high detection repeatability on interest points under image affine transformations. Verdie et al. [56] utilized the first-order horizontal and vertical directional derivatives of images under drastic imaging condition changes of weather and lighting for training a piece-wise linear regressor and performing interest point detection. Barroso-Laguna

et al. [57] followed the ideas of the Harris [20] and DoG methods [42]. The first- and second-order horizontal and vertical directional derivatives are used to train a detector according to the criterion that the detected interest points under image affine transformations should have a high detection repeatability. Hong-Phuoc et al. [145] applied a sparse coding technique to analyze the complexity of interest points on images, and an image pixel where the information complexity reaches a certain level is determined as an interest point.

Although it is indicated in [146] that one cannot accurately determine how a black box operation extracts the required IFI, machine learning techniques are believed to have good capability to learn high-level image abstractions, and these techniques have also been applied for interest point detection and have demonstrated appealing successes on interest point detection and interest points based computer vision tasks (e.g., interest point based image matching). In the following, we will illustrate the high-level semantic abstraction information based interest point detection methods in terms of supervised [49], [147]–[152] and unsupervised [23], [50]–[53], [153]–[158] based approaches.

The goal of supervised learning is to utilize input-label pairs to learn a function that predicts a label (or a distribution over labels) given the input [159]. Yi et al. [49] proposed a learned invariant feature transform (LIFT) method by training a convolutional neural network on image patches corresponding to the same feature but viewed under different ambient conditions. Zhang et al. [148] extended the Lenc and Vedaldi method [50] by using features detected by the TILDE method [56] as a guidance. Huo et al. [149] combined the score map extracted by a handcrafted detector (such as DoG) with the feature selection technique proposed in [148] to improve the performance of a covariant local feature detector. Noh et al. [150] proposed a detector and descriptor framework based on an attention network to detect keypoints and to describe them simultaneously. The framework first obtains the saliency response map of the image through the saliency convolution layer trained by feature matching based image retrieval. Benbihi et al. [152] applied a simple method in which the gradient responses of deep learning features are used for detecting interest points. Xu et al. [160] applied dilated convolution to obtain a feature response map with multi-resolution for improving the scale invariant property of the detection method and solve the problem of detecting occluded interest points by adding tags of occluded interest points in the training set.

On the contrary, unsupervised learning methods do not require such labels or other targets [159]. To improve the robustness of interest point detection with image affine transformations, Lenc and Vedaldi [50] proposed a covariant constraint to transform interest point detection to a regression processing, which applied a trained local transformation predictor for replacing a learned score map to indicate the probability of local features. The Quad-Networks [23] method is the first unsupervised interest point detection method, not relying on a handcrafted feature detector for the training set generation. The training process is based on the rank robustness of the corresponding

point sets under image affine transformations. A point with local extrema response score is defined as an interest point. Zhang et al. [153] extended the method in [23] for detecting interest points in texture images. Ono et al. [51] trained a neural network using the high repeatability of corresponding point pairs to obtain multi-scale feature score maps, and interest points are determined as the points with local maximum scores. Detone et al. [52] introduced a self-supervised learning detector for detecting features from images. Bhowmik et al. [154] extended the SuperPoint method [52], and a reinforcement learning technique [161] is applied to a matching task based on local features (detected by SuperPoint [52]) for detecting interest points from an input image. Sarlin et al. [155] applied the MobileNet [162] and the decoder part of the SuperPoint [52] method for obtaining a final score map from an input image. The points with local extreme score values are determined as interest points. Shen et al. [156] proposed an end-to-end trainable matching network based on a receptive field, and sparse correspondences between images are computed for presenting a new interest point detector. Barroso-Laguna et al. [163] presented the Key.Net method which combined hand-crafted and learned convolutional neural network filters in a shallow multi-scale architecture for extracting interest points from input images with high repeatability under image affine transformations. Dusmanu et al. [53] proposed a framework to accomplish feature detection and description by looking for the local maxima in the response score map of the fourth convolutional layer of VGG-16 pre-trained on ImageNet [164]. To improve the repeatability and reliable performance for image matching, Revaud et al. [157] proposed a repeatable and reliable detector and descriptor (R2D2) network which is trained by maximizing the cosine similarity between corresponding image patches for detecting interest points. Luo et al. [165] applied the light-weight L2-Net [166] to replace the VGG backbone [167] used in D2-Net [53] and deformable convolutional networks [168] to obtain the score map of interest points. Meanwhile, the non-maximum suppression technique was used to obtain the final feature point sets. Yan et al. [169] proposed an equivalent probability formula of the five properties (sparsity, repeatability, invariability, discriminability, and information complexity) about an interest point and its corresponding neighborhood region to build a model loss function. Meanwhile, in order to fit the five non-differentiable properties mentioned earlier, the expectation maximization and efficient approximation techniques were applied to replace the commonly used gradient descent based model optimization methods. Finally, the combination among the five non-differentiable properties and the SuperPoint [52] model were used to obtain an interest point detector and a descriptor. Wang et al. [170] utilized the attention mechanism of vision transformer [171] with multi scale for extracting feature information and detecting keypoints in the detection mechanism of D2-Net [53].

3.5 Frequency domain based IFI extraction techniques for interest point detection

Frequency domain based interest point detection methods extract frequency information for detecting interest points.

TABLE 1 Popular datasets for performance evaluation on interest point detection.

Dataset Name	#Images	#Categories	Content	Synthetic or Real	Highlights
VGG [174]	48	8	Scenes	Real	Scenes with viewpoint and scale changes.
Rosten [46]	37	3	Objects	Synthetic and real	Objects with viewpoint or scale changes.
Strecha [175]	19	2	Scenes	Real	Scenes with viewpoint and scale changes.
ImageNet [164]	14,197,122	21,841	Objects	Real	More than 20,000 typical categories and each contains hundreds of images.
DTU [176]	135,660	60	Objects	Synthetic	Scenes with light, scale, and viewpoint changes.
EF [177]	38	5	Scenes	Real	Scenes with drastic light changes.
Ade-RMF [178]	76	38	Scenes	Synthetic and real	Scenes with viewpoint or scale changes.
Rome [179]	95,961	4	Scenes	Real	Scenes with rotation, scale, light, viewpoint, and photometric changes.
Symbench [180]	92	46	Scenes	Real	Scenes with light changes.
Heinly [181]	65	7	Scenes	Real	Scenes with rotation, light, scale, and image quality changes.
Webcam [56]	120	6	Scenes	Real	Scenes with light, viewpoint, and photometric changes.
Viewpoints [182]	30	5	Scenes	Real	Scenes with viewpoint and rotation changes.
ETH [183]	16,323	9	Scenes	Real	Scenes with illumination, rotation, scale, and viewpoint changes.
HPatches [184]	1,856	116	Scenes	Real	There are 116 scenes with viewpoint and photometric changes.
InLoc [185]	356	7	Objects	Real	Objects with large viewpoint and illumination changes.
Aachen Day-Night [186]	4,328	98	Scenes	Real	Scenes with dramatic season, weather, and day-night changes.
Image Matching [187]	35,279	48	Scenes	Real	Scenes with large light, scale, viewpoint, and photometric changes.

Kovesi [172] applied the phase congruency information to construct a second-order moment matrix in the frequency domain for detecting corners. Chen et al. [173] proposed a block-wise scale-space representation method to reduce the computation complexity of the LoG [31] and DoG [42] methods. An input image is decomposed into blocks, and then the frequency domain convolution and parallel implementation are used for detecting interest points in each block of the images.

4 DATASETS AND PERFORMANCE EVALUATION

In this section, the popular datasets for interest point detection are firstly presented. Secondly, the existing performance evaluation criteria for interest point detection are introduced. Thirdly, the detection performances for the most representative methods are evaluated, discussed, and summarized.

4.1 Image datasets

Datasets have played a very important role for interest point detection. Datasets not only are a means of evaluating the performances of interest point detection algorithms, but also help advance the field of interest point detection greatly. The attributes of popular datasets for performance evaluation on interest point detection are summarized in Table 1.

Currently, images (e.g., the Block image and the Lab image) with ground truths [136] have been widely used for evaluating the detection accuracy and the localization error of a given interest point detector. Meanwhile, image datasets (e.g., the VGG dataset [174]), which include transformed conditions with zoom, rotation, image blur, viewpoint change, light change, and JPEG compression, have been utilized to provide data support for evaluating an interest point detector as to whether it can accurately detect corresponding interest points in image pairs under image affine transformations. However, the data volume of the original VGG image set [174] is too small, which limits the fairness and objectivity on evaluation results. To solve this problem, many datasets (e.g., Strecha [175], ImageNet [164], DTU [176], EF [177], Ade-RMF [178], Symbench [180], Heinly [181], Webcam [56], and Viewpoints [182]) that focus on seasonal changes, light changes, automatic exposure, and image quality changes have been utilized to expand the dataset library for performance evaluation on interest point detection under different image transformations. Furthermore, other image datasets (e.g., Rome [179], HPatches [184], ETH [183], and InLoc [185]) have been employed for evaluating the performance of image matching or 3D reconstruction using detected interest points.

4.2 Evaluation criteria

The existing evaluation criteria on interest point detection [5], [14], [16], [17], [46], [107], [132], [174], [176], [177], [188]–[193] can be divided into four groups: detection capability and localization accuracy metrics for interest point detectors on test images with ground truths [188], [189], repeatability metrics for interest point detectors under image affine transformations [14], [17], [46], [174], [176], [177], [190], performance evaluation on interest point detectors for visual applications [5], [16], [107], [158], and execution time and memory usage [17], [43], [56], [60], [93], [132].

The detection capability and localization accuracy metrics are often used for performance evaluation on corner detection methods [37], [140]. The advantage of this evaluation criterion is that it allows researchers to intuitively determine whether real corners are detected and what the accuracy on corner localization is. However, it is impossible for users to mark reference interest points one by one in thousands of test images [128]. Considering that interest point detectors need to have strong repeatability and real time performances for certain vision tasks (e.g., simultaneous localization and mapping), they should have high probabilities to find the local correspondences between different images within a specified time. Therefore, most performance evaluation criteria on interest point detectors are designed to evaluate the repeatability [14], [17], [46], [174], [176], [177], [190] and execution time [17], [43], [56], [60], [93], [132]. Another way for evaluating the performances of interest point detection algorithms is to assess the performances of the interest point detectors for computer vision tasks such as target tracking [16], [107], image matching [53], [158], [194], [195], or 3D reconstruction [183]. It is worth to note that in addition to the performances of interest point detection algorithms, the performances of these vision tasks are also significantly affected by the descriptors.

There exist a popular repeatability criteria [17], [128], a popular image matching criteria [53], and a popular 3D reconstruction criteria [183] for evaluating the performances of interest point detection algorithms. In the following, the three evaluation criteria [53], [128], [183] are briefly introduced.

4.2.1 Repeatability under image affine transformation

In [128], the average repeatability ξ_{avg} and localization error ϑ_{error} explicitly measure the performance of a detector by comparing the detected interest points from the original image and the detected interest points from an affine transformed image. The definition of the average repeatability is

$$\xi_{\text{avg}} = \frac{P_{Nm}}{2} \left(\frac{1}{P_{No}} + \frac{1}{P_{Nt}} \right), \quad (11)$$

where P_{No} is the number of detected interest points from the original image, P_{Nt} is the number of detected interest points from the transformed image, and P_{Nm} is the number of matched interest points between the original and the transformed images. For an interest point $\mu=(x_o, y_o)$ detected from the original image, its corresponding location

in the transformed image is point $\nu=(x_t, y_t)$. If one interest point is detected from the transformed image and its location is near point $\nu=(x_t, y_t)$ (e.g., within 4 pixels), it means that a pair of matched interest points is obtained.

The localization error is defined as the average distance of all matched interest point pairs between the original and the transformed images. Let $\{(\hat{x}_p, \hat{y}_p), (x_p, y_p): p = 1, 2, \dots, P_{Nm}\}$ be the matched interest point pairs. Then the localization error ϑ_{error} is

$$\vartheta_{\text{error}} = \sqrt{\frac{1}{P_{Nm}} \sum_{p=1}^{P_{Nm}} ((\hat{x}_p - x_p)^2 + (\hat{y}_p - y_p)^2)}. \quad (12)$$

4.2.2 Mean matching accuracy

In [53], for image I_1 and image I_2 with their corresponding homography matrix being \widehat{M} , the sets of extracted interest points from the two images by an interest point detection method are defined as ϱ and ζ respectively, and the sets of descriptors corresponding to the detected interest points ϱ and ζ are defined as γ_1 and γ_2 respectively. When a descriptor in γ_1 corresponding to an interest point in ϱ and a descriptor in γ_2 corresponding to an interest point in ζ are the nearest neighbors to each other, the two interest points are marked as a pair of possible matching points. The number of all possible matching pairs $\{\varrho_i, \zeta_i\}$ ($i = 1, 2, \dots, N_{\text{possible}}$) of the detected interest points between the two images are marked as N_{possible} . For a pair of possible matching points $\{\varrho_i, \zeta_i\}$, their corresponding coordinates are marked as $(x_{\varrho_i}, y_{\varrho_i}, 1)$ and $(u_{\zeta_i}, v_{\zeta_i}, 1)$ respectively. Then the distance error (Ψ_{error}) based on the homography matrix \widehat{M} is calculated as

$$\Psi_{\text{error}} = \| (x_{\varrho_i}, y_{\varrho_i}, 1)^\top - \widehat{M}(u_{\zeta_i}, v_{\zeta_i}, 1)^\top \|_2. \quad (13)$$

If Ψ_{error} is less than a given pixel error threshold ($P_{th}, P_{th} \in \{1, 2, \dots, 10\}$), the possible matching pair is marked as a real matching pair. The number of real matching pairs is defined as N_{match} . Then the mean matching accuracy Λ between the two images is defined as

$$\Lambda = \frac{N_{\text{match}}}{N_{\text{possible}}}. \quad (14)$$

4.2.3 3D reconstruction

According to the protocol developed in [183], after matching two sets of different local image features with a given descriptor, structure-from-motion (SfM) is used for obtaining the camera pose and initial sparse reconstruction, and then multi-view stereo (MVS) is utilized on the output of SfM for obtaining a dense reconstruction of a given scene. The SfM and MVS analyses are made via COLMAP [183].

The 3D reconstruction evaluation criteria [183] contains two parts: (1) Image matching metrics; (2) 3D reconstruction metrics. Four different metrics (i.e., putative match ratio, precision, matching score, and recall) are employed by Heinly et al. [181] for evaluating the accuracy of image matching results. Ten metrics (i.e., number of registered images, reconstructed sparse points, number of observations, mean track length, mean reprojection error, number of inlier pairs, number of inlier matches, reconstructed dense points, mean pose error, and mean

dense error) are utilized for evaluating the completeness and accuracy of 3D reconstruction results.

4.3 Experimental Results

Comprehensive performance evaluation of fifteen state-of-the-art methods (Harris [20], FAST [46], DoG [42], KAZE [44], FOAGDD [37], SOGGDD [34], CPDA [128], New-Curvature [38], LIFT [49], D2-Net [53], LF-Net [51], R2D2 [157], SuperPoint [52], KeyNet [163], and DT-CovDet [148]) are reported in this section. Thirty base images [196] which contain different types of scenes without ground truths are utilized to measure the average repeatabilities of the fifteen methods under different image affine transformations, JPEG image compressions, and image noise degradations. Furthermore, the HPatches dataset [184] (image sets not patch sets) with the HardNet++ descriptor [197] is applied for image matching evaluation. Furthermore, execution times are also investigated for the fifteen methods. The original codes for eleven of these methods in [34], [37], [38], [49], [51]–[53], [128], [148], [157], [163] and the descriptor algorithm (i.e., HardNet++ [197]) are from the authors. The codes for the DoG method [42] is from [198]. The code for the Harris method [20] is from [199]. The codes for the FAST [109] and KAZE [44] methods are from MATLAB.

4.3.1 Average repeatabilities under image affine transformation

In this experiment, thirty images [196] which contain different types of scenes without ground truths are utilized to measure the average repeatabilities for the fifteen methods. With affine transformation, JPEG compression, and noise degradation of each original image, a total of 6,510 transformed test images are obtained as follows:

- Rotation: An input image is rotated with an interval of $\pi/18$ within the range of $[-\pi/2, \pi/2]$, not including 0.
- Uniform scaling: The scale factors $s_x = s_y$ are in $[0.5, 2]$ with an interval of 0.1, not including 1.
- Non-uniform scaling: The scale s_x is in $[0.7, 1.5]$ and s_y is in $[0.5, 1.8]$ with an interval of 0.1, not including $s_x = s_y$.
- Shear transformations: The shear factor f is in $[-1, 1]$ with an interval of 0.1, not including 0, with the following equation

$$\begin{bmatrix} m' \\ n' \end{bmatrix} = \begin{bmatrix} 1 & f \\ 0 & 1 \end{bmatrix} \begin{bmatrix} m \\ n \end{bmatrix}.$$

- Lossy JPEG compression: The compression factor is in $[5, 100]$ with an interval of 5.
- Gaussian noise: Zero-mean white Gaussian noise is added to the input image at 15 variances in $[1, 15]$ with an interval of 1.

With this criteria, the performances of the fifteen methods with their default tuneable parameters are evaluated. In this experiment, if an interest point is detected in a transformed image, and it is near the ground truth location (say within 4 pixels), then a repeated interest point is detected. The average repeatabilities with different rotation, uniform scaling, non-uniform scaling, shear transformation, lossy JPEG compression, and noise

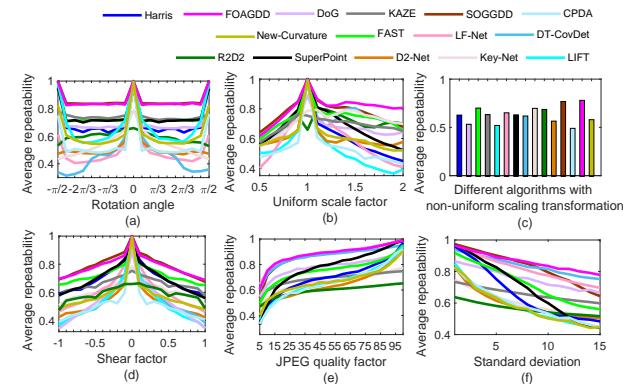


Fig. 6 Average repeatabilities of the fifteen methods under image rotation, uniform scaling, non-uniform scaling, shear transformation, lossy JPEG compression, and noise degradations.

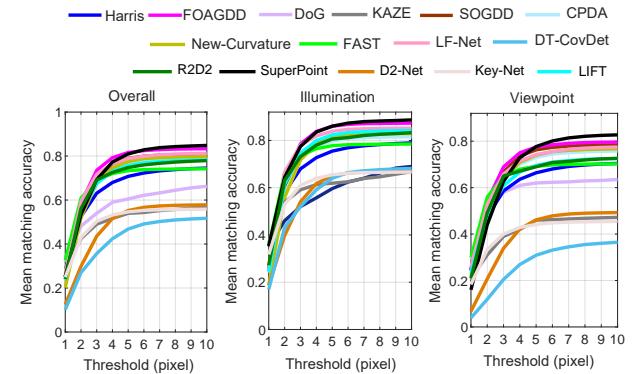


Fig. 7 Mean matching accuracy results of the fifteen detection methods with Hard-Net++ [197].

degradation of the fifteen methods are shown in Fig. 6. It can be found that the FOAGDD method [37] and the SOGGDD method [34] achieved the best and the second best average detection performance in this evaluation criteria respectively.

4.3.2 Mean matching accuracy on the HPatches dataset

In this experiment, 108 out of 116 sequences from the HPatches dataset [184] are selected and 8 sequences with an image resolution beyond 1200×1600 pixels are discarded as some methods are not able to handle higher resolutions [53]. The fifteen methods with their default tuneable parameters and the HardNet++ descriptor [197] are evaluated. The mean matching accuracy results for the overall performance, on illumination and viewpoint changes with thresholds from 1 to 10 pixels averaged over all image pairs are illustrated in Fig. 7. From Fig. 7, it is observed that the FOAGDD method [37] and the SuperPoint method [52] achieved the first and second matching score performances respectively under this evaluation criteria.

4.3.3 3D reconstruction on the ETH dataset

In this experiment, the ETH dataset [183], which contains 9 scenes, is utilized for investigating how an interest point detector with a given descriptor performs in terms of building a 3D model from a set of available 2D images. The detected interest points by the six methods (i.e., DoG [42],

LIFT [49], SuperPoint [52], D2-Net [53], FOAGDD [37], and SOGGDD [34]) are obtained first, and then descriptors are obtained by using HardNet++ [197]. Following [183], [200], no nearest neighbour ratio test is employed to better expose the matching performance of the descriptor. Meanwhile, the same metrics and protocols [53], [183] are used for analysing the 3D models. For the SfM models, the number of registered images (#Reg. Images), the number of reconstructed sparse points (#Sparse Points), the mean track lengths of the 3D points (Track Length), and the mean reprojection error (Reproj. Error) are obtained. For the MVS models, the number of reconstructed dense points (#Dense Points) are obtained. The results for the 3D reconstruction of the six methods are summarized in Table 2. It can be observed from Table 2 that the FOAGDD and SOGDD methods achieve the best and the second best overall performances respectively.

TABLE 2 Comparison results for SfM (\dagger denotes that the result is provided by [200]).

Dataset	Method	#Reg. Images	#Sparse Points	#Dense Points	Track Length	Reproj. Error (pixels)
Fountain 11 images	DoG \dagger	11	16K	303K	4.91	0.47
	LIFT	11	11K	303K	4.57	0.92
	SuperPoint	11	15K	305K	4.58	0.89
	D2-Net	11	21K	301K	4.12	1.49
	FOAGDD	11	32K	306K	5.95	0.71
	SOGGDD	11	28K	305K	4.97	0.83
Herzjesu 8 images	DoG \dagger	8	8K	239K	4.30	0.50
	LIFT	8	9K	240K	4.03	0.81
	SuperPoint	8	5.6K	242K	4.41	0.85
	D2-Net	8	15K	224K	3.16	1.45
	FOAGDD	8	17K	242K	4.46	0.78
	SOGGDD	8	16K	240K	4.21	0.82
South Building 128 images	DoG \dagger	128	159K	2.12M	5.18	0.62
	LIFT	128	127K	2.11M	5.06	0.92
	SuperPoint	128	127K	2.13M	7.17	0.89
	D2-Net	128	183K	2.11M	3.55	1.47
	FOAGDD	128	217K	2.18M	8.65	0.81
	SOGGDD	128	211K	2.13M	7.49	0.84
Madrid Metro– polis– 1344 images	DoG \dagger	793	306K	1.23M	3.84	0.93
	LIFT	732	276K	1.13M	4.55	1.28
	SuperPoint	715	276K	1.15M	4.61	1.24
	D2-Net	795	291K	0.98M	5.19	1.41
	FOAGDD	805	315K	1.18M	5.36	1.16
	SOGGDD	802	314K	1.15M	5.48	1.22
Gendar– men– markt 1463 images	DoG \dagger	1018	827K	2.06M	2.56	1.09
	LIFT	1009	802K	2.03M	2.22	1.32
	SuperPoint	1129	816K	2.09M	3.22	1.29
	D2-Net	1227	811K	2.52M	4.18	1.48
	FOAGDD	1229	828K	3.11M	5.19	1.13
	SOGGDD	1186	820K	3.03M	5.42	1.21

4.4 Execution time and memory usage

Three test images ('Vegetable' [184], 'Graffiti' [174], and 'Bark' [174]) are used in this experiment. The size of these three images are 1200×1600 , 800×640 , and 765×512 pixels respectively. Each interest point detector has been performed in MATLAB (R2018b) using an Intel Core i7-5930K CPU with 128 GB of memory. The GPU model is NVIDIA GeForce GTX TITAN X with 12 GB memory. The threshold for each detector is tuned to detect about 1,500 interest points on each test image. For each test image, an interest point detection method was run 150 times and the mean execution time and memory usage were calculated.

The comparisons on execution time and memory usage are shown in Table 3. It can be seen that the FAST method [109] has the shortest execution time and the New-Curvature method [38] has the least memory usage in this evaluation criteria.

TABLE 3 Comparisons on execution time and memory usage (image size is in pixels. Execution time and memory usage are in second and MB respectively).

Detector	House		Book		Groceries	
	Time (s)	Memory (MB)	Time (s)	Memory (MB)	Time (s)	Memory (MB)
Harris	0.892	121.1	0.172	33.7	0.136	28.4
FOAGDD	43.212	412.8	43.415	339.8	41.816	316.5
SOGGDD	166.201	111.6	29.360	44.6	22.031	34.7
DoG	0.515	199.5	0.133	37.5	0.130	28.2
KAZE	0.750	729.5	0.251	192.6	0.182	148.5
New-Curvature	74.465	33.5	6.971	14.3	2.607	8.3
CPDA	7.342	361.8	5.702	317.6	11.902	325.6
FAST	0.044	39.4	0.023	36.8	0.023	33.2
D2-Net	1.082	1534.4	0.309	1472.8	0.238	1467.5
LF-Net	0.538	1543.6	0.183	1485.5	0.154	1453.4
LIFT	24.017	616.3	7.043	277.4	5.520	247.2
DT-CovDet	0.441	1811.4	0.276	1787.2	0.221	1778.2
Key.Net	4.150	2390.7	1.441	2317.2	1.193	2299.5
SuperPoint	3.445	1948.4	1.145	679.1	0.971	567.6
R2D2	1.029	1440.7	0.762	1432.0	0.594	1431.4

4.5 Summary of experimental results

In this subsection, the experimental results of the fifteen state-of-the-art methods are summarized and analyzed as follows.

The experimental results for the fifteen state-of-the-art methods on the three evaluation criteria in [17], [53], [183] are illustrated in Fig. 6, Fig. 7, and Table 2. From the three evaluation criteria in [17], [53], [183], it can be observed that the FOAGDD [37] and SOGDD [34] methods achieve the first and second overall detection performances. The main reason is that compared with other algorithms, these two methods [34], [37] use multi-directional filtering instead of filtering along the horizontal and vertical directions. The use of multi-directional filtering can extract more accurate IFI for interest point detection. Meanwhile, it can also be found that the detection performances of the other intensity based methods (Harris [20], Harris-Laplace [21], DoG [42], and KAZE [44]) are relatively poor. The reason is that the authors of the above four methods have not considered how to accurately extract IFI. The first- or second-order derivative extraction along the orthogonal directions cannot accurately describe the IFI of an interest point [34], [37]. Take a blob type interest point model as an example as illustrated in Fig. 8(a), its corresponding SOIGDD is shown in the second column of Fig. 8. It can be observed that the horizontal and vertical SOIGDDs do not contain enough IFI for the blob type interest point. After the blob interest point undergoes an affine image transformation as illustrated in Fig. 8(b), its corresponding SOIGDD is illustrated in the second column of Fig. 8. It can also be found that the horizontal and vertical SOIGDDs shown in Fig. 8 do not contain enough IFI for the blob type interest point. Furthermore, there is a large difference on the SOIGDDs between the original blob and the blob undergoing an affine image transformation. Then the blob interest point may not be properly detected under such or similar affine image transformations.

Two other performance measures for interest point detection are execution time and memory usage. As illustrated in Table 3, the FAST algorithm [109] has the shortest execution time among the fifteen methods. It can also be observed that the LF-Net, DT-CovDet, Harris, and DoG methods [20], [42], [43], [51], [148] are excellent with performance indicators on execution time. It is worth to note that the SOGGDD, LIFT, and FOAGDD methods [29], [34], [37], [49], [136] cannot meet the needs of real-time applications. They can be implemented using GPU or FPGA to improve the speed performance. Meanwhile, the Harris method [20] has the least memory usage among the fifteen methods as illustrated in Table 3. The memory usages for the intensity based methods and edge based methods do not have significant differences. Furthermore, it can be found that the memory usages for machine learning based methods (LIFT [49], D2-Net [53], DT-CovDet [148], LF-Net [51], R2D2 [157], SuperPoint [52], and Key.Net [163]) are much higher than other algorithms.

5 SOME FUTURE RESEARCH DIRECTIONS FOR INTEREST POINT DETECTION

Although considerable progresses have been made in theory and performance, image interest point detection is still an open problem and faces a range of challenges.

- Multi-directional IFI extraction with multiple scales: We show that most multi-scale IFI extraction techniques focus on the multi-scale IFI extraction along the two orthogonal directions. As shown in Fig. 8, the existing multi-scale IFI extraction approaches along the orthogonal directions cannot properly extract IFI. It is necessary to extract IFI from an input image along multi-directions with multiple scales. Up to now, no one has proposed a theoretical framework about how to accurately extract multi-directional IFI with multiple scales for better detecting interest points.

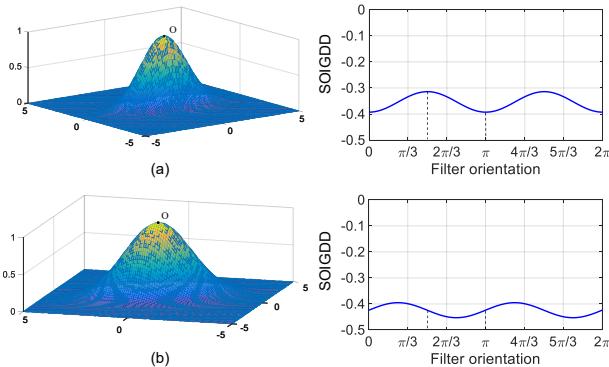


Fig. 8 Examples of SOIGDD changes with image affine image transformation. (a) a blob-type interest point, (b) the blob-type interest point undergoing an affine image transformation.

- IFI extraction for adjacent interest points detection: The existing IFI extraction approaches for interest point detection (especially for intensity variation based IFI extraction) have not considered how to accurately extract the IFI for adjacent interest points. Take a local region of a Chinese chessboard image as an example, the interest

point detection results from the FOAGDD method [37] for this region are illustrated in Fig. 9 (b). It can be found that the adjacent interest points cannot be accurately detected by the FOAGDD method. The reason is that there are mutual influences for the IFIs between the adjacent interest points. The mutual influences of the IFIs lead to the existing interest point detection methods not accurately detecting the adjacent interest points, or only detecting one of the interest points, or the localizations of the detected interest points being seriously offset as illustrated in Fig. 9 (b).

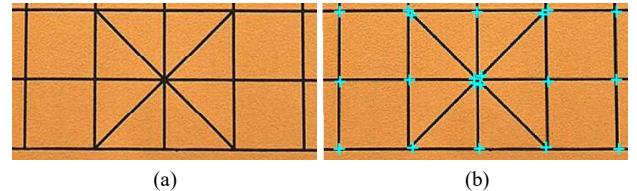


Fig. 9 Examples of the adjacent interest points detection results by an interest point detector. (a) a local Chinese chessboard image, (b) the adjacent interest points detection results from the FOAGDD method.

- IFI extraction for interest point detection under different imaging conditions: The existing interest point detection methods have not fully considered how to detect interest points from the same scene under different imaging conditions. Take a building scene as an example, the interest point detection results from the FOAGDD detector [37] under different illuminations are illustrated in Fig. 10. It can be found that some obvious interest points in the window area (marked by 'O') cannot be detected under different illuminations as illustrated in Fig. 10(a) and (b). The reason is that the magnitude information of the IFI for an interest point at the same position will change with the change of imaging conditions.



Fig. 10 Examples of the interest point detection results by an interest point detector under different light conditions.

Furthermore, our research also indicates that different imaging conditions will cause the first-order derivatives at edge points to be larger than that at interest points in many directions. Take the test image 'House' under different imaging condition as an example, an interest point and an edge point at the same location in Fig. 11(a) and (d) are marked by 'O' and '□' respectively. It can be seen from Fig. 11(b) and (c) that the corresponding FOAGDD of the interest point is larger than the FOAGDD of the edge point in most directions. Then the FOAGDD method [37] can easily detect the interest point in the scene in Fig. 11(a). If

the scene undergoes a different imaging condition as shown in Fig. 11(d), it can be seen from Fig. 11(e) and (f) that the FOAGDD for the interest point is less than the FOAGDD for the edge point in most directions. Then the interest point may not be detected with such or similar imaging conditions by the FOAGDD detector [37]. Therefore, in order to better detect interest points from an input image, in addition to the amplitude information of the intensity variations of the image pixels, it is necessary to combine other techniques (e.g., image illumination robust strategy [63] and the difference between the waveforms of the intensity variations at the interest points and the step edges [37]) to extract and analyze the IFI for image pixels.

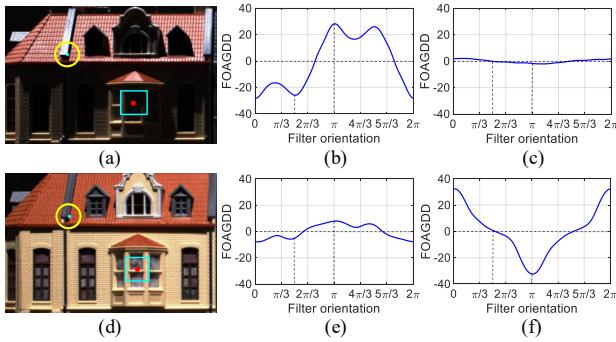


Fig. 11 Examples of the FOAGDDs at an interest point and an edge point at the same locations under different imaging conditions. (a) and (d) are the images with different lighting conditions, (b) and (e) are the corresponding FOAGDD responses with multi-directions for an interest point (marked by '○'), and (c) and (f) are the corresponding FOAGDD responses with multi-directions for an edge point (marked by '□').

- IFI extraction for interest point detection under image affine transformations: The existing interest point detection methods have not considered how to detect interest points from an input image under affine transformations. Take the test image 'v_bird' as an example as shown in Fig. 12(a), an interest point and an edge point at the same location in Fig. 12(a) and (d) are marked by '○' and '□' respectively. As illustrated in Fig. 12(b) and (c), the corresponding FOAGDD for the interest point is larger than the FOAGDD at the edge point in most directions. Then the interest point can be easily detected by the FOAGDD method [37] from the original image. If the original image undergoes an affine transformation as illustrated in Fig. 12(d), the corresponding FOAGDD for the interest point is less than the FOAGDD for the edge point in many directions as illustrated in Fig. 12(e) and (f). Then the interest point may not be detected with such or similar image affine transformations. Within the scope of our investigations, no one has proposed how to accurately extract IFI for accurately detecting interest points under the conditions of image affine transformations.

- Deep learning techniques for interest point detection: In recent years, deep learning techniques have shown tremendous improvements for many computer vision tasks such as image segmentation and image classification. From the experimental results on the three performance evaluation criteria [128], [183], [184], it can be found that the existing deep learning based interest point detectors

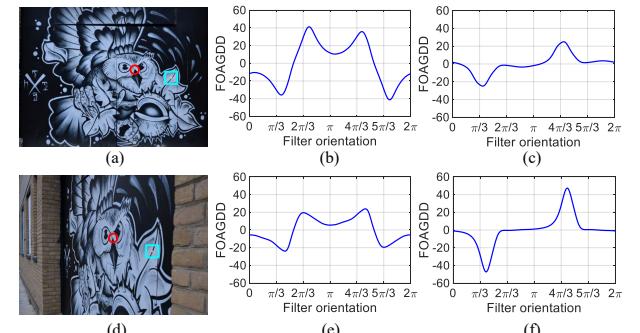


Fig. 12 Examples of the FOAGDDs at an interest point and an edge point at the same location under image affine transformations. (a) and (d) are the images under different affine transformations, (b) and (e) are the corresponding FOAGDD responses with multi-directions for an interest point (marked by '○'), and (c) and (f) are the corresponding FOAGDD responses with multi-directions for an edge point (marked by '□').

(e.g., LIFT [49], D2-Net [53], LF-Net [51], DT-CovDet [148], R2D2 [157], SuperPoint [52], and Key.Net [163]) have no obvious advantages on detection, image matching, and 3D reconstruction performances over the intensity based methods. One of the possible reasons is that the bias in the understanding of properties of interest points by the current deep learning based detectors. By further exploring and discovering the characteristics of interest points, the potential capacity of deep learning approaches for accurate interest point detection may be improved in the future.

- The evaluation and application of interest point detection in different computer vision tasks: It is worth to note that interest point detectors are rarely used alone, but a step in a complete pipeline, which introduces additional requirements for the detected interest points. Take image matching as an example, it is difficult to evaluate an interest point detector without a given descriptor. Although an interest point detector is extremely repeatable with image affine transformations, the corresponding points extracted in local regions that are not discriminative enough for the descriptor to differentiate will significantly reduce the accuracy of image matching. Another typical example is image matching for image transformation estimation. In order to better estimate a homography, the detected points and their correspondences are preferably evenly distributed on different images. However, if a detector has good repeatability in image affine transformations and localization, while its corresponding detected points only spread in a very small region of an image, the transformation estimation results will be most probably not accurate [158]. Tan et al. [201] investigated feature matching in stereo images encouraging uniform point distribution. Therefore, a promising research direction is to fairly evaluate the performance of interest point detectors without relying on specific applications and customize modern image interest point detection techniques to meet different requirements of actual vision tasks such as structure-from-motion, object tracking, and image-guided robotic-assisted surgery.

6 CONCLUSION

As a contemporary interest point detection survey, this paper presented a taxonomy of image feature information extraction techniques for interest point detection, discussed advantages and disadvantages of existing interest point detection methods, introduced existing popular datasets and evaluation criteria, and analyzed the performances for fifteen most representative interest point detection methods in terms of average recall rates under different camera positions and lighting changes, average repeatabilities under different image affine transformations, JPEG compressions, and noise degradations, and mean matching accuracy on the HPatches dataset with the HardNet++ descriptor. Furthermore, the development trend for image feature information extraction for interest point detection is elaborated. We hope that this survey will not only enable researchers to better understand image interest point detection but also inspire future research activities.

ACKNOWLEDGMENTS

This work was supported in part by the Innovation Capability Support Program of Shaanxi (No. 2021TD-29), in part by the Youth Innovation Team of Shaanxi Universities, and in part by the National Natural Science Foundation of China (No. 62176204).

REFERENCE

- [1] F. Attneave, Some informational aspects of visual perception, *Psychological Review* 61 (3) (1954) 183.
- [2] Z. Zhang, Flexible camera calibration by viewing a plane from unknown orientations, in: Proceedings of the IEEE Conference on International Conference on Computer Vision, 1999, pp. 666–673.
- [3] J. Ma, J. Jiang, H. Zhou, J. Zhao, X. Guo, Guided locality preserving feature matching for remote sensing image registration, *IEEE Transactions on Geoscience and Remote Sensing* 56 (8) (2018) 4435–4447.
- [4] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, M. J. Milford, Visual place recognition: A survey, *IEEE Transactions on Robotics* 32 (1) (2015) 1–19.
- [5] B. Fan, Q. Kong, X. Wang, Z. Wang, S. Xiang, C. Pan, P. Fua, A performance evaluation of local features for image based 3D reconstruction, *IEEE Transactions on Image Processing* 28 (10) (2019) 4774–4789.
- [6] J. R. Tang, H. Cheng, Y. Zhao, H. L. Guo, Structured dynamic time warping for continuous hand trajectory gesture recognition, *Pattern Recognition* 80 (2018) 21–31.
- [7] H. Huang, H. Zhou, H. Qin, M. Sheng, Underwater vehicle visual servo and target grasp control, in: Proceedings of the IEEE Conference on International Conference on Robotics and Biomimetics, 2016, pp. 1619–1624.
- [8] L. He, H. Li, Q. Zhang, Z. Sun, Dynamic feature learning for partial face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7054–7063.
- [9] S. R. E. Datondji, Y. Dupuis, P. Subirats, P. Vasseur, A survey of vision-based traffic monitoring of road intersections, *IEEE Transactions on Intelligent Transportation Systems* 17 (10) (2016) 2681–2698.
- [10] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, D. Tran, Detect-and-track: Efficient pose estimation in videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 350–359.
- [11] D. J. Mirota, H. Wang, R. H. Taylor, M. Ishii, G. L. Gallia, G. D. Hager, A system for video based navigation for endoscopic endonasal skull base surgery, *IEEE Transactions on Medical Imaging* 31 (4) (2011) 963–976.
- [12] X. Wang, Intelligent multi-camera video surveillance: A review, *Pattern Recognition Letters* 34 (1) (2013) 3–19.
- [13] D. Amin, S. Govilkar, Comparative study of augmented reality SDKs, *International Journal on Computational Science and Applications* 5 (1) (2015) 11–26.
- [14] C. Schmid, R. Mohr, C. Bauckhage, Evaluation of interest point detectors, *International Journal of Computer Vision* 37 (2) (2000) 151–172.
- [15] T. Tuytelaars, K. Mikolajczyk, Local invariant feature detectors: A survey, *Foundations and Trends in Computer Graphics and Vision* 3 (3) (2008) 177–280.
- [16] S. Gauglitz, T. Höllerer, M. Turk, Evaluation of interest point detectors and feature descriptors for visual tracking, *International Journal of Computer Vision* 94 (3) (2011) 335.
- [17] M. Awrangjeb, G. Lu, C. S. Fraser, Performance comparisons of contour-based corner detectors, *IEEE Transactions on Image Processing* 21 (9) (2012) 4167–4179.
- [18] O. Miksik, K. Mikolajczyk, Evaluation of local detectors and descriptors for fast feature matching, in: Proceedings of the International Conference on Pattern Recognition, 2012, pp. 2681–2684.
- [19] F. Tombari, S. Salti, L. Di Stefano, Performance evaluation of 3D keypoint detectors, *International Journal of Computer Vision* 102 (1–3) (2013) 198–220.
- [20] C. Harris, M. Stephens, A combined corner and edge detector, in: Alvey Vision Conference, 1988, pp. 23.1–23.6.
- [21] K. Mikolajczyk, C. Schmid, Scale and affine invariant interest point detectors, *International Journal of Computer Vision* 60 (1) (2004) 63–86.
- [22] Y. Li, S. Wang, T. Qi, X. Ding, A survey of recent advances in visual feature detection, *Neurocomputing* 149 (2015) 736–751.
- [23] N. Savinov, A. Seki, L. Ladicky, T. Sattler, M. Pollefeys, Quadr-networks: Unsupervised learning to rank for interest point detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1822–1830.
- [24] J. Matas, O. Chum, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, *Image and Vision Computing* 22 (10) (2004) 761–767.
- [25] C. Varytimidis, K. Rapantzikos, Y. Avrithis, S. Kollias, α -shapes for local feature detection, *Pattern Recognition* 50 (2016) 56–73.
- [26] K. Mikolajczyk, C. Schmid, Indexing based on scale invariant interest points, in: Proceedings of the IEEE International Conference on Computer Vision, 2002, pp. 525–531.
- [27] S. Salti, A. Lanza, L. Di Stefano, Keypoints from symmetries by wave propagation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2898–2905.
- [28] H. Nasser, P. Ngo, I. Debled-Rennesson, Dominant point detection based on discrete curve structure and applications, *Journal of Computer and System Sciences* 95 (2018) 177–192.
- [29] G. S. Xia, J. Delon, Y. Gousseau, Accurate junction detection and characterization in natural images, *International Journal of Computer Vision* 106 (1) (2014) 31–56.
- [30] P. Zhu, P. M. Chirlian, On critical point detection of digital shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17 (8) (1995) 737–748.
- [31] T. Lindeberg, Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention, *International Journal of Computer Vision* 11 (3) (1993) 283–318.
- [32] S. Hinz, Fast and subpixel precise blob detection and attribution, in: Proceedings of the IEEE International Conference on Image Processing, Vol. 3, 2005, pp. 453–457.
- [33] W. Banzhaf, P. Nordin, R. E. Keller, F. D. Francone, *Genetic Programming*, Springer, 1998.
- [34] W. Zhang, C. Sun, Corner detection using second-order generalized Gaussian directional derivative representations, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (4) (2021) 1213–1224.
- [35] K. Rohr, Recognizing corners by fitting parametric models, *International Journal of Computer Vision* 9 (3) (1992) 213–230.
- [36] H. Moravec, Visual mapping by a robot rover, in: Proceedings of the International Joint Conference on Artificial Intelligence, Vol. 1, 1979, pp. 598–600.
- [37] W. Zhang, C. Sun, Corner detection using multi-directional structure tensor with multiple scales, *International Journal of Computer Vision* 128 (2) (2020) 438–459.

- [38] W. Zhang, C. Sun, T. Breckon, N. Alshammary, Discrete curvature representations for noise robust image corner detection, *IEEE Transactions on Image Processing* 28 (9) (2019) 4444–4459.
- [39] S. M. Smith, J. M. Brady, SUSAN: A new approach to low level image processing, *International Journal of Computer Vision* 23 (1) (1997) 45–78.
- [40] T. Lindeberg, Image matching using generalized scale-space interest points, *Journal of Mathematical Imaging and Vision* 52 (1) (2015) 3–36.
- [41] A. Shademan, R. S. Decker, J. Opfermann, S. Leonard, P. C. Kim, A. Krieger, Plenoptic cameras in surgical robotics: Calibration, registration, and evaluation, in: *Proceedings of the IEEE International Conference on Robotics and Automation*, 2016, pp. 708–714.
- [42] D. G. Lowe, Distinctive image features from scale-invariant keypoints, *International Journal of Computer Vision* 60 (2) (2004) 91–110.
- [43] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-up robust features (SURF), *Computer Vision and Image Understanding* 110 (3) (2008) 346–359.
- [44] P. F. Alcantarilla, A. Bartoli, A. J. Davison, KAZE features, in: *Proceedings of the European Conference on Computer Vision*, 2012, pp. 214–227.
- [45] F. Mokhtarian, R. Suomela, Robust image corner detection through curvature scale space, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (12) (1998) 1376–1381.
- [46] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, in: *Proceedings of the European Conference on Computer Vision*, 2006, pp. 430–443.
- [47] E. Rublee, V. Rabaud, K. Konolige, G. R. Bradski, ORB: An efficient alternative to SIFT or SURF, in: *Proceedings of the IEEE Conference on International Conference on Computer Vision*, 2011, pp. 2564–2571.
- [48] S. Leutenegger, M. Chli, R. Siegwart, BRISK: Binary robust invariant scalable keypoints, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2011, pp. 2548–2555.
- [49] K. M. Yi, E. Trulls, V. Lepetit, P. Fua, LIFT: Learned invariant feature transform, in: *Proceedings of the European Conference on Computer Vision*, 2016, pp. 467–483.
- [50] K. Lenc, A. Vedaldi, Learning covariant feature detectors, in: *Proceedings of the European Conference on Computer Vision*, 2016, pp. 100–117.
- [51] Y. Ono, E. Trulls, P. Fua, K. M. Yi, LF-Net: Learning local features from images, in: *Proceedings of the Advances in Neural Information Processing Systems*, 2018, pp. 6234–6244.
- [52] D. DeTone, T. Malisiewicz, A. Rabinovich, SuperPoint: Self-supervised interest point detection and description, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 224–236.
- [53] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, T. Sattler, D2-Net: A trainable CNN for joint description and detection of local features, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8092–8101.
- [54] J. Canny, A computational approach to edge detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (6) (1986) 679–698.
- [55] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, G. Randall, LSD: A fast line segment detector with a false detection control, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (4) (2008) 722–732.
- [56] Y. Verdie, K. M. Yi, P. Fua, V. Lepetit, TILDE: A temporally invariant learned detector, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5279–5288.
- [57] A. Barroso-Laguna, E. Riba, D. Ponsa, K. Mikolajczyk, Key.Net: Keypoint Detection by Handcrafted and Learned CNN Filters, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1–9.
- [58] J. Shi, C. Tomasi, Good features to track, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994, pp. 593–600.
- [59] T. Brox, J. Weickert, B. Burgeth, P. Mrázek, Nonlinear structure tensors, *Image and Vision Computing* 24 (1) (2006) 41–55.
- [60] P. Mainali, Q. Yang, G. Lafruit, L. Van Gool, R. Lauwereins, Robust low complexity corner detector, *IEEE Transactions on Circuits and Systems for Video Technology* 21 (4) (2011) 435–445.
- [61] S. Ando, Image field categorization and edge/corner detection from gradient covariance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (2) (2000) 179–190.
- [62] Y. Andreopoulos, I. Patras, Incremental refinement of image salient-point detection, *IEEE Transactions on Image Processing* 17 (9) (2008) 1685–1699.
- [63] M. Gevrekci, B. K. Gunturk, Illumination robust interest point detection, *Computer Vision and Image Understanding* 113 (4) (2009) 565–571.
- [64] M. Loog, F. Lauze, The improbability of Harris interest points, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (6) (2010) 1141–1147.
- [65] W. Förstner, A framework for low level feature extraction, in: *Proceedings of the European Conference on Computer Vision*, 1994, pp. 383–394.
- [66] D. Gao, V. Mahadevan, N. Vasconcelos, On the plausibility of the discriminant center-surround hypothesis for visual saliency, *Journal of Vision* 8 (7) (2008) 13–13.
- [67] I. Laptev, T. Lindeberg, On space-time interest points, *International Journal of Computer Vision* 64 (2-3) (2005) 107–123.
- [68] T. Lindeberg, Feature detection with automatic scale selection, *International Journal of Computer Vision* 30 (2) (1998) 79–116.
- [69] X. Gao, F. Sattar, R. Venkateswarlu, Multiscale corner detection of gray level images based on LoG-Gabor wavelet transform, *IEEE Transactions on Circuits and Systems for Video Technology* 17 (7) (2007) 868–875.
- [70] M. A. Duval-Poo, F. Odene, E. De Vito, Edges and corners with shearlets, *IEEE Transactions on Image Processing* 24 (11) (2015) 3768–3780.
- [71] J. Van de Weijer, T. Gevers, J.-M. Geusebroek, Edge and corner detection by photometric quasi-invariants, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27 (4) (2005) 625–630.
- [72] J.-M. Geusebroek, R. Van den Boomgaard, A. W. M. Smeulders, H. Geerts, Color invariance, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (12) (2001) 1338–1350.
- [73] M. A. Ruzon, C. Tomasi, Edge, junction, and corner detection using color distributions, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (11) (2001) 1281–1295.
- [74] Y. Rubner, C. Tomasi, L. J. Guibas, A metric for distributions with applications to image databases, in: *Proceeding of the IEEE International Conference on Computer Vision*, 1998, pp. 59–66.
- [75] J. Van de Weijer, T. Gevers, A. D. Bagdanov, Boosting color saliency in image feature detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (1) (2005) 150–156.
- [76] C. S. Kenney, M. Zuliani, B. Manjunath, An axiomatic approach to corner detection, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 1, 2005, pp. 191–197.
- [77] J. Stöttinger, A. Hanbury, N. Sebe, T. Gevers, Sparse color interest points for image retrieval and object categorization, *IEEE Transactions on Image Processing* 21 (5) (2012) 2681–2692.
- [78] M. Wang, W. Zhang, C. Sun, A. Sowmya, Corner detection based on shearlet transform and multi-directional structure tensor, *Pattern Recognition* 103 (2020) 107299.
- [79] D. Reisfeld, H. Wolfson, Y. Yeshurun, Context-free attentional operators: The generalized symmetry transform, *International Journal of Computer Vision* 14 (2) (1995) 119–130.
- [80] G. Heidemann, Focus-of-attention from local color symmetries, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26 (7) (2004) 817–830.
- [81] G. Loy, A. Zelinsky, Fast radial symmetry for detecting points of interest, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (8) (2003) 959–973.
- [82] J. Maver, Self-similarity and points of interest, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (7) (2010) 1211–1226.
- [83] P. Beaudet, Rotational invariant image operators, in: *Proceedings of the International Conference of Pattern Recognition*, 1978, pp. 579–583.
- [84] Y. Abdeljaoued, T. Ebrahimi, Feature point extraction using scale-space representation, in: *Proceedings of the IEEE International Conference on Image Processing*, Vol. 5, 2004, pp. 3053–3056.
- [85] T. Lindeberg, *Scale-Space Theory in Computer Vision*, Kluwer Academic Publishers, 1994.

- [86] D. Marimon, A. Bonnin, T. Adamek, R. Gimeno, DARTs: Efficient scale-space extraction of DAISY keypoints, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 2416–2423.
- [87] M. McDonnell, Box-filtering techniques, *Computer Graphics and Image Processing* 17 (1) (1981) 65–70.
- [88] T. Gao, J. Jing, C. Liu, W. Zhang, Y. Gao, C. Sun, Fast corner detection using approximate form of second-order Gaussian directional derivative, *IEEE Access* 8 (2020) 194092–194104.
- [89] R. Su, C. Sun, T. D. Pham, Junction detection for linear structures based on Hessian, correlation and shape information, *Pattern Recognition* 45 (10) (2012) 3695–3706.
- [90] J. Weickert, B. T. H. Romeny, M. A. Viergever, Efficient and reliable schemes for nonlinear diffusion filtering, *IEEE Transactions on Image Processing* 7 (3) (1998) 398–410.
- [91] P. Perona, J. Malik, Scale-space and edge detection using anisotropic diffusion, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12 (7) (1990) 629–639.
- [92] P. F. Alcantarilla, A. Bartoli, A. J. Davison, Fast explicit diffusion for accelerated features in nonlinear scale spaces, in: Proceedings of the British Machine Vision Conference, 2013, pp. 1281–1298.
- [93] P. Mainali, G. Lafruit, Q. Yang, B. Geelen, L. Van Gool, R. Lauwereins, SIFER: scale-invariant feature detector with error resilience, *International Journal of Computer Vision* 104 (2) (2013) 172–197.
- [94] Z. Miao, X. Jiang, Interest point detection using rank order LoG filter, *Pattern Recognition* 46 (11) (2013) 2890–2901.
- [95] F. Mainardi, The fundamental solutions for the fractional diffusion-wave equation, *Applied Mathematics Letters* 9 (6) (1996) 23–28.
- [96] Y. Li, W. Liu, X. Li, Q. Huang, X. Li, GA-SIFT: A new scale invariant feature transform for multispectral image using geometric algebra, *Information Sciences* 281 (2014) 559–572.
- [97] D. Hestenes, G. Sobczyk, Clifford algebra to geometric calculus: A unified language for mathematics and physics, Springer Science & Business Media, 2012.
- [98] Z. Miao, X. Jiang, K.-H. Yap, Contrast invariant interest point detection by zero-norm LoG filter, *IEEE Transactions on Image Processing* 25 (1) (2015) 331–342.
- [99] M. A. Duval-Poo, N. Noceti, F. Odone, E. De Vito, Scale invariant and noise robust interest points with shearlets, *IEEE Transactions on Image Processing* 26 (6) (2017) 2853–2867.
- [100] H. Kong, H. C. Akakin, S. E. Sarma, A generalized Laplacian of Gaussian filter for blob detection and its applications, *IEEE Transactions on Cybernetics* 43 (6) (2013) 1719–1733.
- [101] O. Li, P.-I. Shui, Subpixel blob localization and shape estimation by gradient search in parameter space of anisotropic Gaussian kernels, *Signal Processing* 171 (2020) 107495.
- [102] G. Wang, C. Lopez-Molina, B. De Baets, Automated blob detection using iterative Laplacian of Gaussian filtering and unilateral second-order Gaussian kernels, *Digital Signal Processing* 96 (2020) 102592.
- [103] M. Ghahremani, Y. Liu, B. Tiddeman, FFD: Fast feature detector, *IEEE Transactions on Image Processing* 30 (2020) 1153–1168.
- [104] P. Mainali, G. Lafruit, K. Tack, L. Van Gool, R. Lauwereins, Derivative-based scale invariant image feature detector with error resilience, *IEEE Transactions on Image Processing* 23 (5) (2014) 2380–2391.
- [105] W. Heisenberg, Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik, in: Original Scientific Papers Wissenschaftliche Originalarbeiten, Springer, 1985, pp. 478–504.
- [106] E. Saad, K. Hirakawa, Defocus blur-invariant scale-space feature extractions, *IEEE Transactions on Image Processing* 25 (7) (2016) 3141–3156.
- [107] M. Trajkovic, M. Hedley, Fast corner detection, *Image and Vision Computing* 16 (2) (1998) 75–87.
- [108] M. Cazorla, F. Escolano, D. Gallardo, R. Rizo, Junction detection and grouping with probabilistic edge models and Bayesian A*, *Pattern Recognition* 35 (9) (2002) 1869–1881.
- [109] E. Rosten, R. Porter, T. Drummond, Faster and better: A machine learning approach to corner detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32 (1) (2010) 105–119.
- [110] S. Bennett, J. Lasenby, ChESS-Quick and robust detection of chess-board features, *Computer Vision and Image Understanding* 118 (2014) 197–210.
- [111] Y. Bok, H. Ha, I. S. Kweon, Automated checkerboard detection and indexing using circular boundaries, *Pattern Recognition Letters* 71 (2016) 66–72.
- [112] S. Buoncompagni, D. Maio, D. Maltoni, S. Papi, Saliency-based keypoint selection for fast object detection and matching, *Pattern Recognition Letters* 62 (2015) 32–40.
- [113] M. Calonder, V. Lepetit, C. Strecha, P. Fua, BRIEF: Binary robust independent elementary features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 778–792.
- [114] M. Awrangjeb, G. Lu, C. S. Fraser, Performance comparisons of contour-based corner detectors, *IEEE Transactions on Image Processing* 21 (9) (2012) 4167–79.
- [115] A. Rosenfeld, E. Johnston, Angle detection on digital curves, *IEEE Transactions on Computers* C-22 (9) (1973) 875–878.
- [116] A. Rosenfeld, J. S. Weszka, An improved method of angle detection on digital curves, *IEEE Transactions on Computers* 24 (9) (1975) 940–941.
- [117] C.-H. Teh, R. T. Chin, On the detection of dominant points on digital curves, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (8) (1989) 859–872.
- [118] H. Freeman, L. S. Davis, A corner-finding algorithm for chain-coded curves, *IEEE Transactions on Computers* 26 (3) (1977) 297–303.
- [119] A. Rattarangsi, R. T. Chin, Scale-based detection of corners of planar curves, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1 (1990) 923–930.
- [120] F. Mokhtarian, A. Mackworth, Scale-based description and recognition of planar curves and two-dimensional shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (1) (1986) 34–43.
- [121] F. Mokhtarian, A. Mackworth, A theory of multiscale, curvature-based shape representation for planar curves, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14 (8) (1992) 789–805.
- [122] B. Zhong, W. Liao, Direct curvature scale space: Theory and corner detection, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (3) (2007) 508–512.
- [123] X. Zhang, M. Lei, D. Yang, Y. Wang, L. Ma, Multi-scale curvature product for robust image corner detection in curvature scale space, *Pattern Recognition Letters* 28 (5) (2007) 545–554.
- [124] X. He, N. H. C. Yung, Corner detector based on global and local curvature properties, *Optical Engineering* 47 (5) (2008) 057008–1–057008–12.
- [125] T.-A. Pham, M. Delalandre, S. Barrat, J.-Y. Ramel, Accurate junction detection and characterization in line-drawing images, *Pattern Recognition* 47 (1) (2014) 282–295.
- [126] X. Zhang, Y. Qu, D. Yang, H. Wang, J. Kymer, Laplacian scale-space behavior of planar curve corners, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37 (11) (2015) 2207–2217.
- [127] Z. Wang, D. Chen, J. Gong, C. Wang, Fast high-precision ellipse detection method, *Pattern Recognition* 111 (2021) 107741.
- [128] M. Awrangjeb, G. Lu, Robust image corner detection based on the chord-to-point distance accumulation technique, *IEEE Transactions on Multimedia* 10 (6) (2008) 1059–1072.
- [129] J. H. Han, T. Poston, Chord-to-point distance accumulation and planar curvature: a new approach to discrete curvature, *Pattern Recognition Letters* 22 (10) (2001) 1133–1144.
- [130] X. Zhang, H. Wang, A. W. Smith, X. Ling, B. C. Lovell, D. Yang, Corner detection based on gradient correlation matrices of planar curves, *Pattern Recognition* 43 (4) (2010) 1207–1223.
- [131] S. W. Teng, R. M. N. Sadat, G. Lu, Effective and efficient contour-based corner detectors, *Pattern Recognition* 48 (7) (2015) 2185–2197.
- [132] A. Mustafa, H. Kim, A. Hilton, MSFD: Multi-scale segmentation-based feature detection for wide-baseline scene reconstruction, *IEEE Transactions on Image Processing* 28 (3) (2018) 1118–1132.
- [133] L. Kitchen, A. Rosenfeld, Gray-level corner detection, *Pattern Recognition Letters* 1 (2) (1982) 95–102.
- [134] R. Elias, R. Laganière, JUDOCA: Junction detection operator based on circumferential anchors, *IEEE Transactions on Image Processing* 21 (4) (2011) 2109–2118.
- [135] G. Azzopardi, N. Petkov, Trainable COSFIRE filters for keypoint detection and pattern recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35 (2) (2012) 490–503.

- [136] P.-L. Shui, W.-C. Zhang, Corner detection and classification using anisotropic directional derivative representations, *IEEE Transactions on Image Processing* 22 (8) (2013) 3204–3218.
- [137] P. Shui, W. Zhang, Noise-robust edge detector combining isotropic and anisotropic Gaussian kernels, *Pattern Recognition* 45 (2012) 806–820.
- [138] A. Desolneux, L. Moisan, J.-M. Morel, Meaningful alignments, *International Journal of Computer Vision* 40 (1) (2000) 7–23.
- [139] W.-C. Zhang, F.-P. Wang, L. Zhu, Z.-F. Zhou, Corner detection using Gabor filters, *IET Image Processing* 8 (11) (2014) 639–646.
- [140] W.-C. Zhang, P.-L. Shui, Contour-based corner detection via angle difference of principal directions of anisotropic Gaussian directional derivatives, *Pattern Recognition* 48 (9) (2015) 2785–2797.
- [141] N. Xue, G.-S. Xia, X. Bai, L. Zhang, W. Shen, Anisotropic scale junction detection and matching for indoor images, *IEEE Transactions on Image Processing* 27 (1) (2017) 78–91.
- [142] J. R. Quinlan, Induction of decision trees, *Machine Learning* 1 (1) (1986) 81–106.
- [143] V. Lepetit, P. Fua, Keypoint recognition using randomized trees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28 (9) (2006) 1465–1479.
- [144] L. Trujillo, G. Olague, Automated design of image operators that detect interest points, *Evolutionary Computation* 16 (4) (2008) 483–507.
- [145] T. Hong-Phuoc, Y. He, L. Guan, SCK: A sparse coding based key-point detector, in: *Proceedings of the IEEE International Conference on Image Processing*, 2018, pp. 3768–3772.
- [146] R. C. Fong, A. Vedaldi, Interpretable explanations of black boxes by meaningful perturbation, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3429–3437.
- [147] A. Richardson, E. Olson, Learning convolutional filters for interest point detection, in: *Proceedings of the IEEE International Conference on Robotics and Automation*, 2013, pp. 631–637.
- [148] X. Zhang, F. X. Yu, S. Karaman, S.-F. Chang, Learning discriminative and transformation covariant local feature detectors, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6818–6826.
- [149] Z. Huo, Y. Zhang, H. Liu, J. Wang, X. Liu, J. Zhang, Improved covariant local feature detector, *Pattern Recognition Letters* 135 (2020) 1–7.
- [150] H. Noh, A. Araujo, J. Sim, T. Weyand, B. Han, Large-scale image retrieval with attentive deep local features, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3456–3465.
- [151] F. P. Di, M. C. Dal, K. Tieu, S. Mattoccia, KCNN: Extremely-efficient hardware keypoint detection with a compact convolutional neural network, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 682–690.
- [152] A. Benbihi, M. Geist, C. Pradalier, ELF: Embedded localisation of features in pre-trained CNN, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 7940–7949.
- [153] L. Zhang, S. Rusinkiewicz, Learning to detect features in texture images, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6325–6333.
- [154] A. Bhowmik, S. Gumhold, C. Rother, E. Brachmann, Reinforced feature points: Optimizing feature detection and description for a high-level task, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4948–4957.
- [155] P.-E. Sarlin, C. Cadena, R. Siegwart, M. Dymczyk, From coarse to fine: Robust hierarchical localization at large scale, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12716–12725.
- [156] X. Shen, C. Wang, X. Li, Z. Yu, J. Li, C. Wen, M. Cheng, Z. He, RF-Net: An end-to-end image matching network based on receptive field, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8132–8140.
- [157] J. Revaud, P. Weinzaepfel, C. De Souza, N. Pion, G. Csurka, Y. Cabon, M. Humenberger, R2D2: Repeatable and reliable detector and descriptor, in: *Proceedings of the IEEE Conference on Neural Information Processing Systems*, 2019, pp. 1–12.
- [158] P. Truong, S. Apostolopoulos, A. Mosinska, S. Stucky, C. Ciller, S. D. Zanet, GLAMpoints: Greedily learned accurate match points, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 10732–10741.
- [159] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, MIT Press, 2016.
- [160] J. Xu, B. Song, X. Yang, X. Nan, An improved deep keypoint detection network for space targets pose estimation, *Remote Sensing* 12 (23) (2020) 3857.
- [161] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Machine Learning* 8 (3–4) (1992) 229–256.
- [162] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, MobileNetV2: Inverted residuals and linear bottlenecks, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.
- [163] A. Barroso-Laguna, E. Riba, D. Ponsa, K. Mikolajczyk, KeyNet: Keypoint detection by handcrafted and learned CNN filters, in: *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 5836–5844.
- [164] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A large-scale hierarchical image database, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
- [165] Z. Luo, L. Zhou, X. Bai, H. Chen, J. Zhang, Y. Yao, S. Li, T. Fang, L. Quan, ASLFeat: Learning local features of accurate shape and localization, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6589–6598.
- [166] Y. Tian, B. Fan, F. Wu, L2-Net: Deep learning of discriminative patch descriptor in Euclidean space, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 661–669.
- [167] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: *Proceedings of the International Conference on Learning Representations*, 2014.
- [168] X. Zhu, H. Hu, S. Lin, J. Dai, Deformable ConvNets V2: More deformable, better results, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9308–9316.
- [169] P. Yan, Y. Tan, Y. Tai, D. Wu, H. Luo, X. Hao, Unsupervised learning framework for interest point detection and description via properties optimization, *Pattern Recognition* 112 (2021) 107808.
- [170] Z. Wang, X. Li, Z. Li, Local representation is not enough: Soft point-wise transformer for descriptor and detector of local features, in: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, 2021, pp. 1150–1156.
- [171] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.
- [172] P. Kovesi, Phase congruency detects corners and edges, in: *Proceedings of the Australian Pattern Recognition Society Conference*, 2003, pp. 309–318.
- [173] J. Chen, L.-Y. Duan, F. Gao, J. Cai, A. C. Kot, T. Huang, A low complexity interest point detector, *IEEE Signal Processing Letters* 22 (2) (2014) 172–176.
- [174] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Van Gool, A comparison of affine region detectors, *International Journal of Computer Vision* 65 (1–2) (2005) 43–72.
- [175] C. Strecha, W. Von Hansen, L. Van Gool, P. Fua, U. Thoennessen, On benchmarking camera calibration and multi-view stereo for high resolution imagery, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
- [176] H. Aanæs, A. L. Dahl, K. S. Pedersen, Interesting interest points, *International Journal of Computer Vision* 97 (1) (2012) 18–35.
- [177] C. L. Zitnick, K. Ramnath, Edge foci interest points, in: *Proceedings of the IEEE Conference on International Conference on Computer Vision*, 2011, pp. 359–366.
- [178] H. S. Wong, T.-J. Chin, J. Yu, D. Suter, Dynamic and hierarchical multi-structure geometric model fitting, in: *Proceedings of the IEEE Conference on International Conference on Computer Vision*, 2011, pp. 1044–1051.
- [179] D. Crandall, A. Owens, N. Snavely, D. Huttenlocher, Discrete-continuous optimization for large-scale structure from motion, in: *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 3001–3008.

- [180] D. C. Hauagge, N. Snavely, Image matching using local symmetry features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 206–213.
- [181] J. Heinly, E. Dunn, J.-M. Frahm, Comparative evaluation of binary features, in: Proceedings of the Conference on European Conference on Computer Vision, 2012, pp. 759–773.
- [182] K. Moo Yi, Y. Verdie, P. Fua, V. Lepetit, Learning to assign orientations to feature points, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 107–116.
- [183] J. L. Schonberger, H. Hardmeier, T. Sattler, M. Pollefeys, Comparative evaluation of hand-crafted and learned local features, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1482–1491.
- [184] V. Balntas, K. Lenc, A. Vedaldi, K. Mikolajczyk, HPatches: A benchmark and evaluation of handcrafted and learned local descriptors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5173–5182.
- [185] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, A. Torii, InLoc: Indoor visual localization with dense matching and view synthesis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7199–7209.
- [186] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, et al., Benchmarking 6dof outdoor visual localization in changing conditions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 8601–8610.
- [187] G. Research, Image matching benchmark datasets, <https://www.cs.ubc.ca/research/image-matching-challenge/current/data/> (2021).
- [188] F. Mohanna, F. Mokhtarian, Performance evaluation of corner detection algorithms under similarity and affine transforms., in: Proceedings of the British Machine Vision Conference, 2001, pp. 1–10.
- [189] T. Dickscheid, F. Schindler, W. Förstner, Coding images with local features, International Journal of Computer Vision 94 (2) (2011) 154–174.
- [190] F. Fraundorfer, H. Bischof, A novel performance evaluation method of local detectors on non-planar scenes, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2005, pp. 33–33.
- [191] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.
- [192] Y. Jiang, Y. Xu, Y. Liu, Performance evaluation of feature detection and matching in stereo visual odometry, Neurocomputing 120 (2013) 380–390.
- [193] H. Aanæs, R. R. Jensen, G. Vogiatzis, E. Tola, A. B. Dahl, Large-scale data for multiple-view stereopsis, International Journal of Computer Vision 120 (2) (2016) 153–168.
- [194] M. Agrawal, K. Konolige, M. R. Blas, CenSurE: Center surround extrema for realtime feature detection and matching, in: European Conference on Computer Vision, 2008, pp. 102–115.
- [195] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, E. Trulls, Image matching across wide baselines: From paper to practice, International Journal of Computer Vision 129 (2) (2021) 517–547.
- [196] K. Bowyer, C. Kranenburg, S. Dougherty, Edge detector evaluation using empirical ROC curves, in: IEEE Conference on Computer Vision and Pattern Recognition, Vol. 1, 1999, pp. 354–359.
- [197] M. Anastasiya, M. Dmytro, R. Filip, M. Jiri, Working hard to know your neighbor's margins: Local descriptor learning loss, in: Proceedings of the Neural Information Processing Systems, 2017.
- [198] A. Vedaldi, B. Fulkerson, VLFeat: An open and portable library of computer vision algorithms, <http://www.vlfeat.org/> (2008).
- [199] P. Kovesi, Code of Harris corner detector, <https://peterkovesi.com/matlabfn/> (2005).
- [200] Y. Tian, V. Balntas, T. Ng, A. Barroso-Laguna, Y. Demiris, K. Mikolajczyk, D2D: Keypoint extraction with describe to detect approach, in: Proceedings of the Asian Conference on Computer Vision, 2020, pp. 9308–9326.
- [201] X. Tan, C. Sun, X. Sirault, R. Furbank, T. D. Pham, Feature matching in stereo images encouraging uniform spatial distribution, Pattern Recognition 48 (8) (2015) 2530–2542.



Junfeng Jing received the MS degree in electrical engineering from Xi'an Polytechnic University in China and the PhD degree in mechatronic engineering from Xidian University in China. He is a professor at Xi'an Polytechnic University, China. His research interests include artificial intelligence, machine vision, image processing, pattern recognition, and industrial robot control.



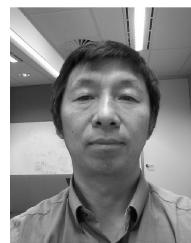
Tian Gao received the MS degree in the school of electronic and information, Xi'an Polytechnic University, China. He is currently pursuing the PhD degree with School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include computer vision, image processing, deep learning, and SLAM.



Weichuan Zhang received the MS degree in signal and information processing from the Southwest Jiaotong University in China and the PhD degree in signal and information processing in National Lab of Radar Signal Processing, Xidian University, China. He is a research fellow at Griffith University, QLD, Australia. His research interests include computer vision, image analysis, and pattern recognition. He is a member of the IEEE.



Yongsheng Gao received the B.Sc. and M.Sc. degrees in electronic engineering from Zhejiang University, Hangzhou, China, in 1985 and 1988, respectively, and the Ph.D. degree in computer engineering from Nanyang Technological University, Singapore. He is currently a Professor with the School of Engineering and Built Environment, Griffith University, and the Director of ARC Research Hub for Driving Farming Productivity and Disease Prevention, Australia. He had been the Leader of Biosecurities Group, Queensland Research Laboratory, National ICT Australia (ARC Centre of Excellence), a consultant of Panasonic Singapore Laboratories, and an Assistant Professor in the School of Computer Engineering, Nanyang Technological University, Singapore. His research interests include smart farming, machine vision for agriculture, biosecurity, face recognition, biometrics, image retrieval, computer vision, pattern recognition, environmental informatics, and medical imaging.

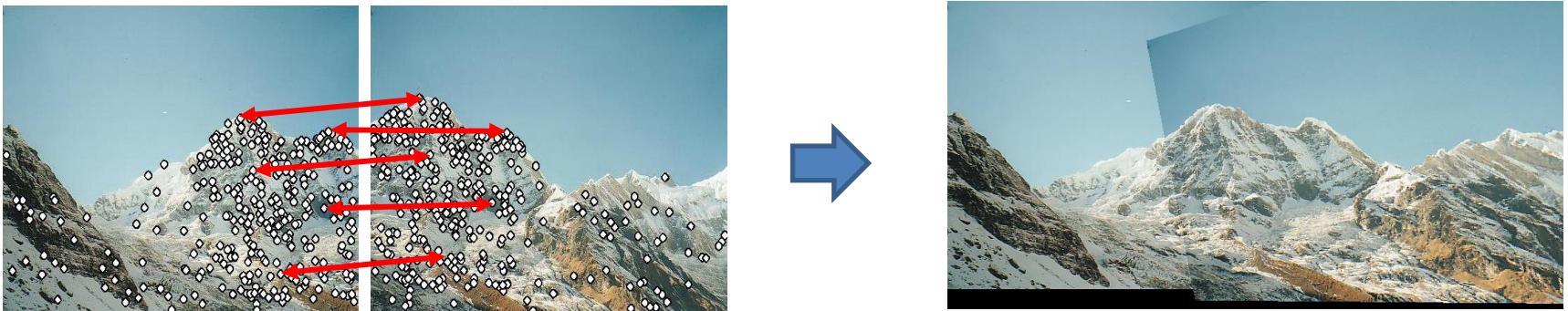
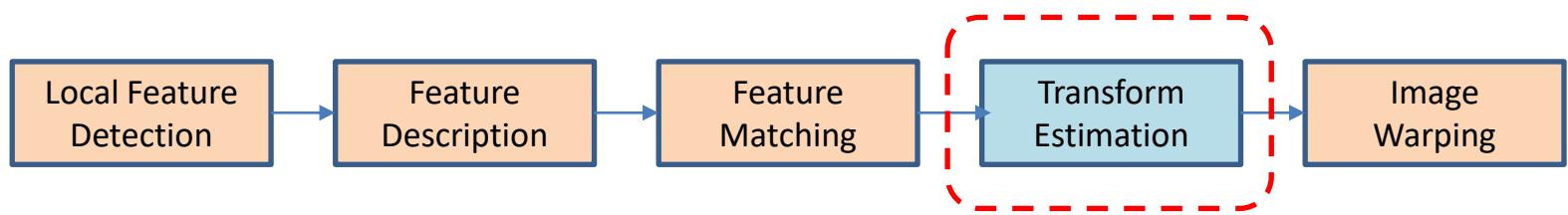


Changming Sun received his PhD degree in computer vision from Imperial College London, London, UK in 1992. He then joined CSIRO, Sydney, Australia, where he is currently a Principal Research Scientist carrying out research and working on applied projects. He is also a Conjoint Professor at the School of Computer Science and Engineering of the University of New South Wales. He has served on the program/organizing committees of various international conferences. He is an Associate Editor of the EURASIP Journal on Image and Video Processing. His current research interests include computer vision, image analysis, and pattern recognition.

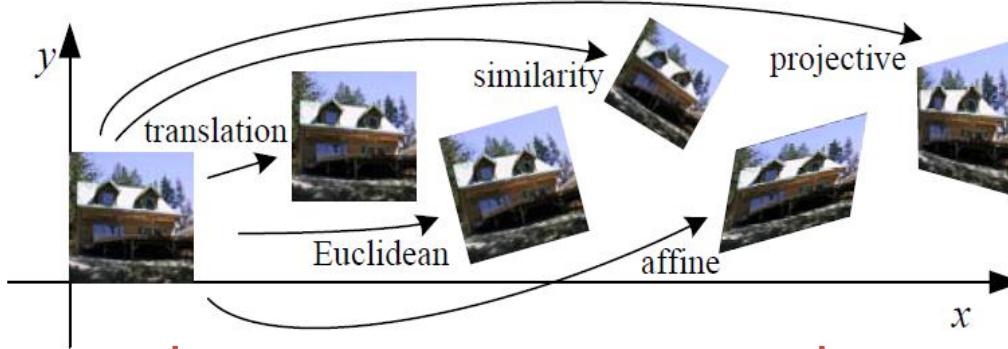
Image Alignment and Stitching-2: Transformation Estimation

Szeliski Computer Vision Book
Chapter 3.6 – Geometric transformations
Chapter 8.1 – 2D alignment using least
squares

Image Alignment



Transformation	Matrix	# DoF	Preserves	Icon
similarity	$\left[\begin{array}{c c} sR & t \end{array} \right]_{2 \times 3}$	4	angles	
affine	$\left[\begin{array}{c} A \end{array} \right]_{2 \times 3}$	6	parallelism	
projective	$\left[\begin{array}{c} \tilde{H} \end{array} \right]_{3 \times 3}$	8	straight lines	



Scaled Rotation + Translation
(similarity transform
 $a^2+b^2=1$ no longer required)

Affine
(A is arbitrary)

Projective (Perspective Transform/Homography)
(H arbitrary)

$$x' = [sR \ t]x$$

$$x' = [A]x$$

$$x' = [H]x$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

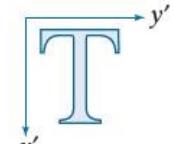
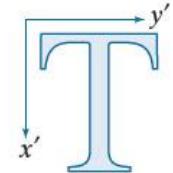
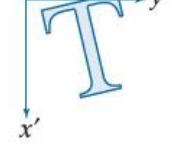
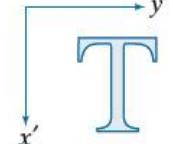
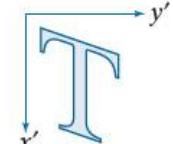
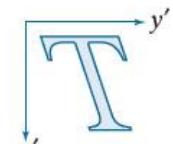
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine Transformations

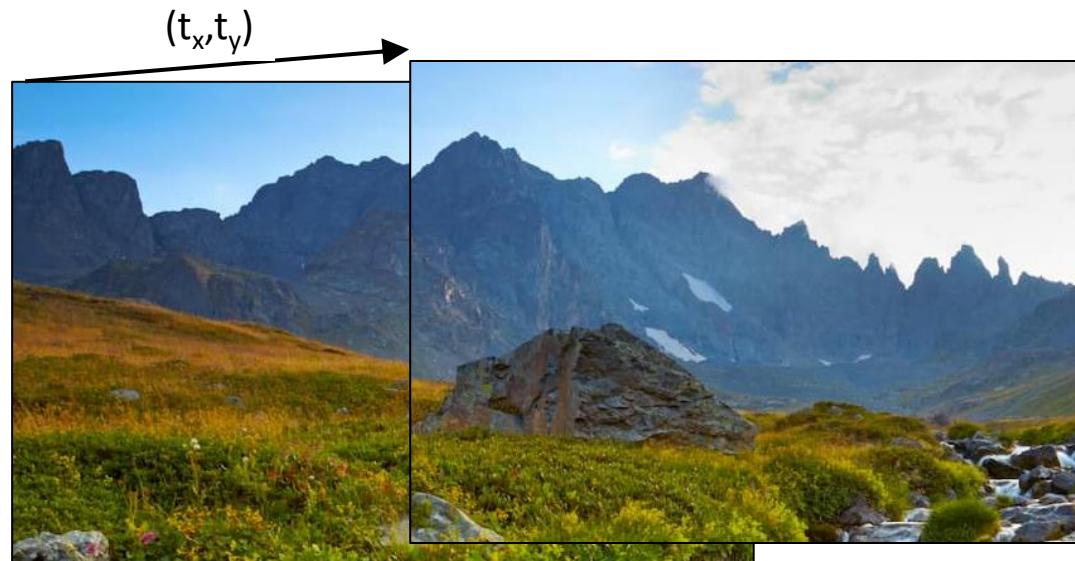
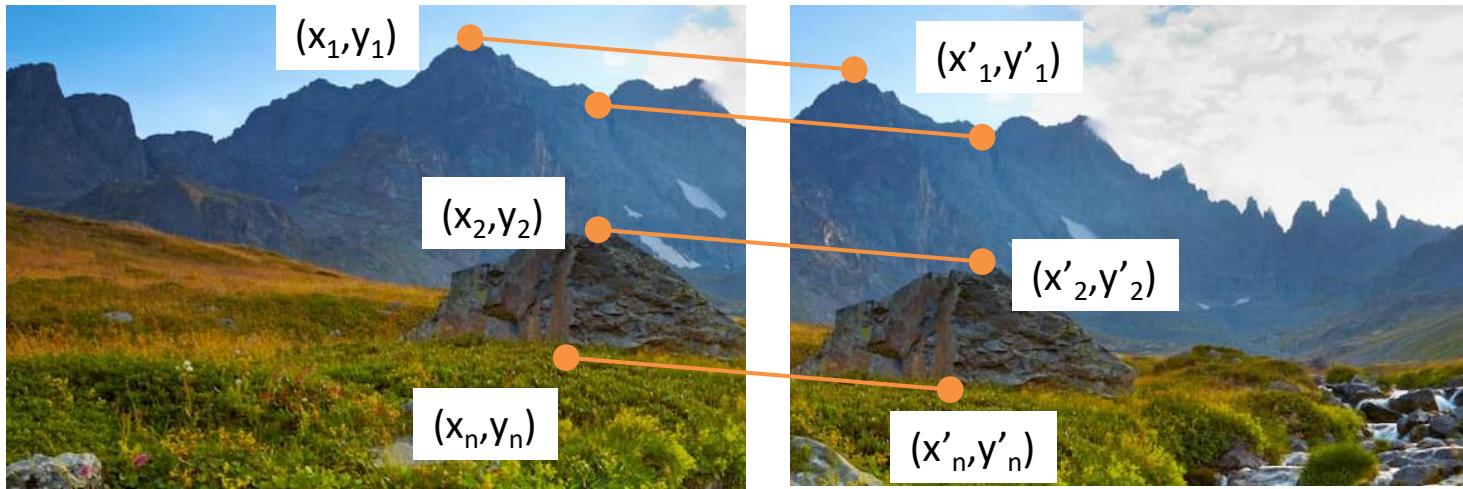
$$x' = Ax$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

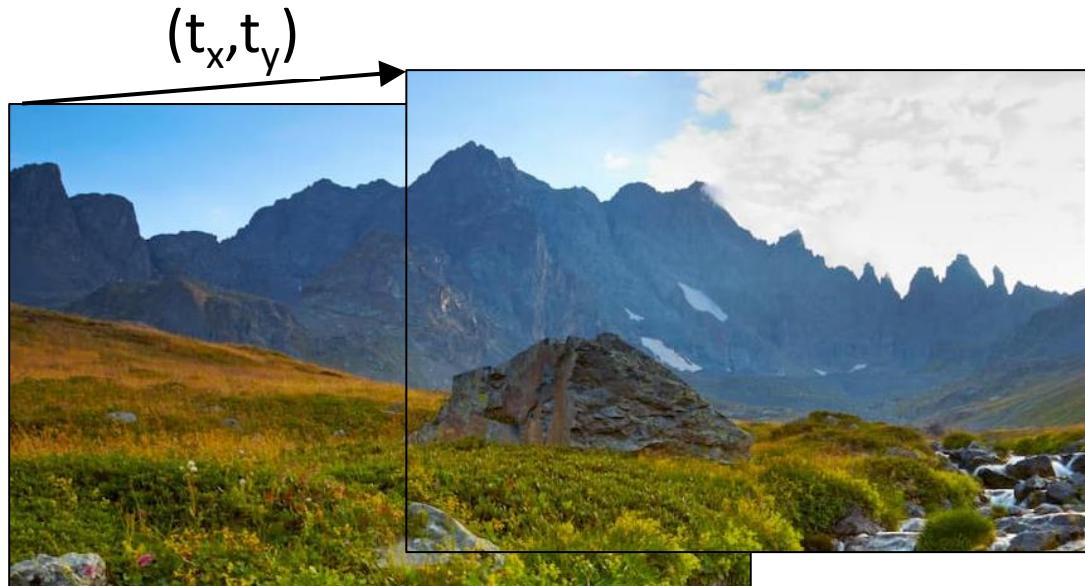
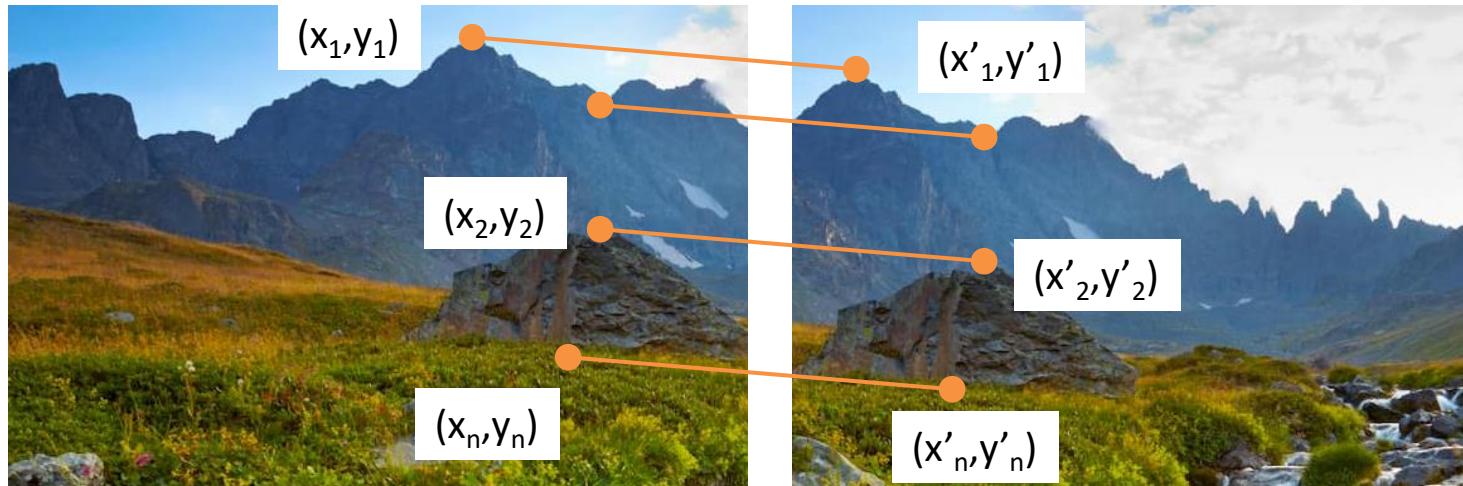
- Affine transformations are combinations of ...
 - Linear transformations, and
 - Translations
- Properties of affine transformations:
 - Origin does not necessarily map to origin
 - Lines map to lines
 - Parallel lines remain parallel
 - Ratios are preserved
 - Closed under composition

Transformation Name	Affine Matrix, A	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = y$	
Scaling/Reflection (For reflection, set one scaling factor to -1 and the other to 0)	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = c_x x$ $y' = c_y y$	
Rotation (about the origin)	$\begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x \cos \theta - y \sin \theta$ $y' = x \sin \theta + y \cos \theta$	
Translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + t_x$ $y' = y + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & s_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x + s_v y$ $y' = y$	
Shear (horizontal)	$\begin{bmatrix} 1 & 0 & 0 \\ s_h & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x' = x$ $y' = s_h x + y$	

Transformation=Translation



Transformation=Translation



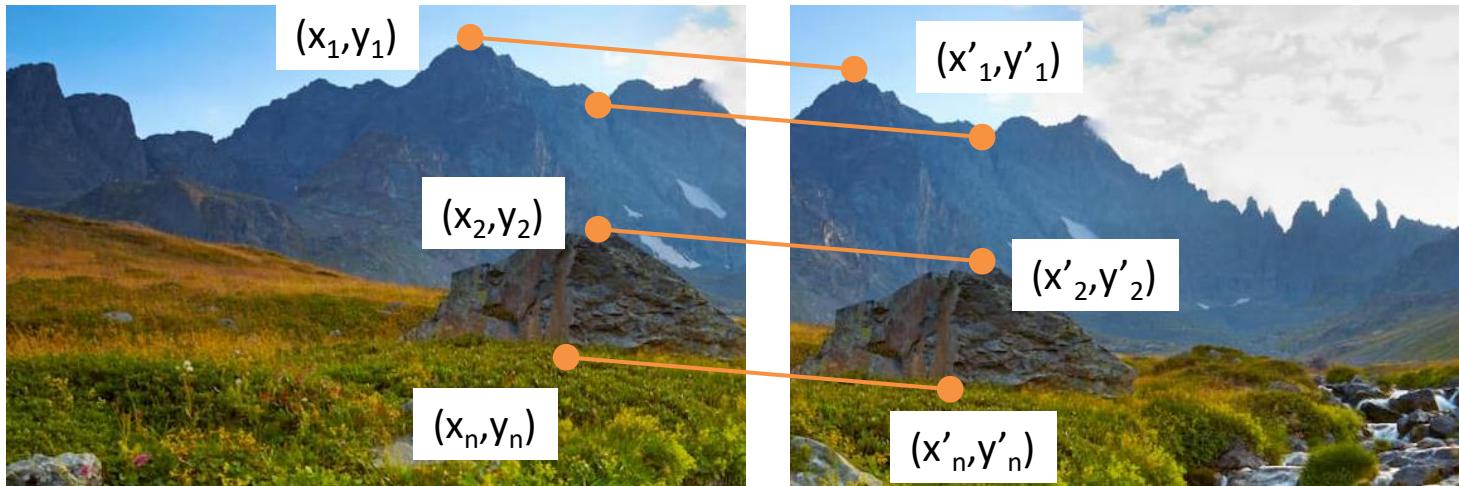
$$t_{xi} = x'_i - x_i$$
$$t_{yi} = y'_i - y_i$$

Mean translation

$$t_x = \frac{1}{n} \sum_{i=1}^n (x'_i - x_i)$$

$$t_y = \frac{1}{n} \sum_{i=1}^n (y'_i - y_i)$$

Transformation=Translation



Another Formulation

$$x'_i = x_i + t_{xi}$$

$$y'_i = y_i + t_{yi}$$

- System of linear equations
 - What are the knowns? Unknowns?
 - How many unknowns?
 - How many equations (per match)?

- Problem: more equations than unknowns
 - “Overdetermined” system of equations
 - *Least squares* solution

Least squares formulation

For each point $(\mathbf{x}_i, \mathbf{y}_i)$

$$x'_i = x_i + t_{xi}$$

$$y'_i = y_i + t_{yi}$$



Residuals:

$$r_{xi}(t_x) = (x_i + t_x) - x'_i$$

$$r_{yi}(t_y) = (y_i + t_y) - y'_i$$

Minimize:

$$Cost(t_x, t_y) = \sum_{i=1}^n (r_{xi}(t_x)^2 + r_{yi}(t_y)^2)$$

Least squares formulation

- Goal: minimize sum of squared residuals

$$C(\mathbf{x}_t, \mathbf{y}_t) = \sum_{i=1}^n (r_{\mathbf{x}_i}(\mathbf{x}_t)^2 + r_{\mathbf{y}_i}(\mathbf{y}_t)^2)$$

- “Least squares” solution
- For translations, is equal to mean (average) displacement

Least squares formulation

$$t_x = x'_1 - x_1$$

$$t_y = y'_1 - y_1$$

$$t_x = x'_2 - x_2$$

$$t_y = y'_2 - y_2$$

$$t_x = x'_n - x_n$$

$$t_y = y'_n - y_n$$



$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

 $=$

$$\begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\mathbf{A}_{2n \times 2} \mathbf{t}_{2 \times 1} = \mathbf{b}_{2n \times 1}$$

Least squares formulation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\mathbf{A}_{2n \times 2} \mathbf{t}_{2 \times 1} = \mathbf{b}_{2n \times 1} \quad \longrightarrow$$

Solve for $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$

$$\begin{aligned} \mathbf{A} \mathbf{t} &= \mathbf{b} \\ \mathbf{A}^{-1} \mathbf{A} \mathbf{t} &= \mathbf{A}^{-1} \mathbf{b} \\ \mathbf{t} &= \mathbf{A}^{-1} \mathbf{b} \end{aligned}$$

Least squares formulation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$

$$\mathbf{A}_{2n \times 2} \mathbf{t}_{2 \times 1} = \mathbf{b}_{2n \times 1}$$



Solve for $\begin{bmatrix} t_x \\ t_y \end{bmatrix}$

A is not a square matrix Inverse(A) does not exist!

$\mathbf{A} \mathbf{t} = \mathbf{b}$
 $\mathbf{A}^{-1} \mathbf{A} \mathbf{t} = \mathbf{A}^{-1} \mathbf{b}$
 $\mathbf{t} = \mathbf{A}^{-1} \mathbf{b}$

Least squares formulation

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x'_1 - x_1 \\ y'_1 - y_1 \\ x'_2 - x_2 \\ y'_2 - y_2 \\ \vdots \\ x'_n - x_n \\ y'_n - y_n \end{bmatrix}$$
$$\| At - b \|^2$$

A $\begin{bmatrix} t \\ \end{bmatrix}$ = **b** 

$$\begin{aligned} At &= b \\ A^T A t &= A^T b \\ (A^T A)^{-1} A^T A t &= (A^T A)^{-1} A^T b \\ t &= (A^T A)^{-1} A^T b \end{aligned}$$

Last slide covered on 2/21/2023

Affine transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

Residuals :

$$r_{x_i}(a, b, c, d, e, f) = (ax_i + by_i + c) - x'_i$$

$$r_{y_i}(a, b, c, d, e, f) = (dx_i + ey_i + f) - y'_i$$

Cost :

$$C(a, b, c, d, e, f) =$$

$$\sum_{i=1}^n (r_{x_i}(a, b, c, d, e, f)^2 + r_{y_i}(a, b, c, d, e, f)^2)$$

Affine transformations --- Matrix Form

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} ax + by + c \\ dx + ey + f \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ \vdots & & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

A
 $2n \times 6$

t
 6×1

b
 $2n \times 1$

Affine transformations --- Matrix Form

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & 1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & 1 \\ \vdots & & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} = \begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ \vdots \\ x'_n \\ y'_n \end{bmatrix}$$

A
 $2n \times 6$ **t**
 6×1 = **b**
 $2n \times 1$

$$\begin{aligned} A t &= b \\ A^T A t &= A^T b \\ (A^T A)^{-1} A^T A t &= (A^T A)^{-1} A^T b \\ t &= (A^T A)^{-1} A^T b \end{aligned}$$

$$\| At - b \|^2$$

Matlab: `t = A \ b;`

Python:
`t = numpy.linalg.lstsq(A, b)`

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

Not linear!

$$x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \quad \xrightarrow{\hspace{10em}} \quad \begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & & \vdots & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

A
 $2n \times 9$

h
 9
0
 $2n$

Defines a least squares problem: minimize $\|Ah - 0\|^2$

Problem statement

minimize $\mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x}$ s.t. $\mathbf{x}^T \mathbf{x} = 1$
(non-trivial lsq solution to $\mathbf{A} \mathbf{x} = 0$)

Solution

$[\mathbf{v}, \lambda] = \text{eig}(\mathbf{A}^T \mathbf{A})$
 $\lambda_1 < \lambda_{2..n}: \mathbf{x} = \mathbf{v}_1$

Recap: Two Common Optimization Problems

Problem statement

$$\text{minimize } \|\mathbf{Ax} - \mathbf{b}\|^2$$

(least squares solution to $\mathbf{Ax} = \mathbf{b}$)

Solution

$$\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

$\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$ (matlab)

Problem statement

$$\text{minimize } \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} \text{ s.t. } \mathbf{x}^T \mathbf{x} = 1$$

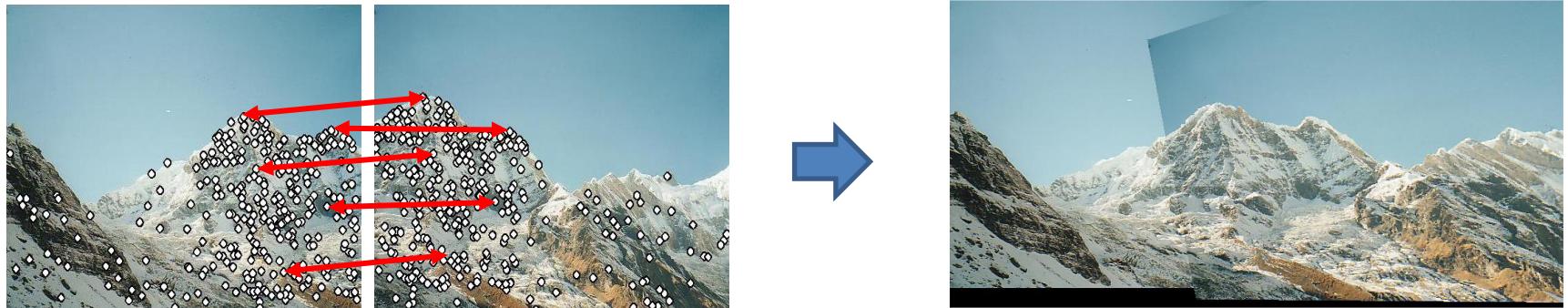
(non-trivial lsq solution to $\mathbf{Ax} = 0$)

Solution

$$[\mathbf{v}, \lambda] = \text{eig}(\mathbf{A}^T \mathbf{A})$$

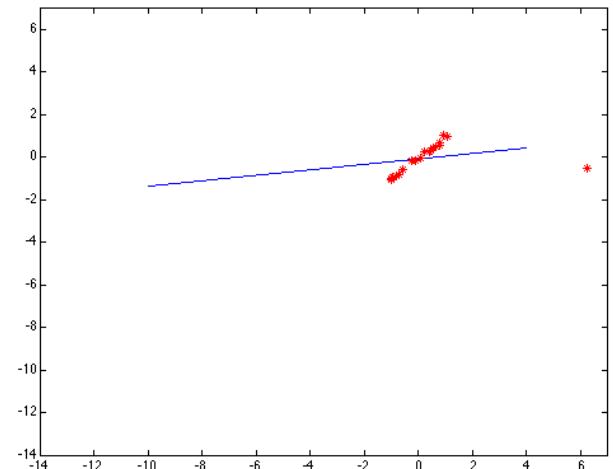
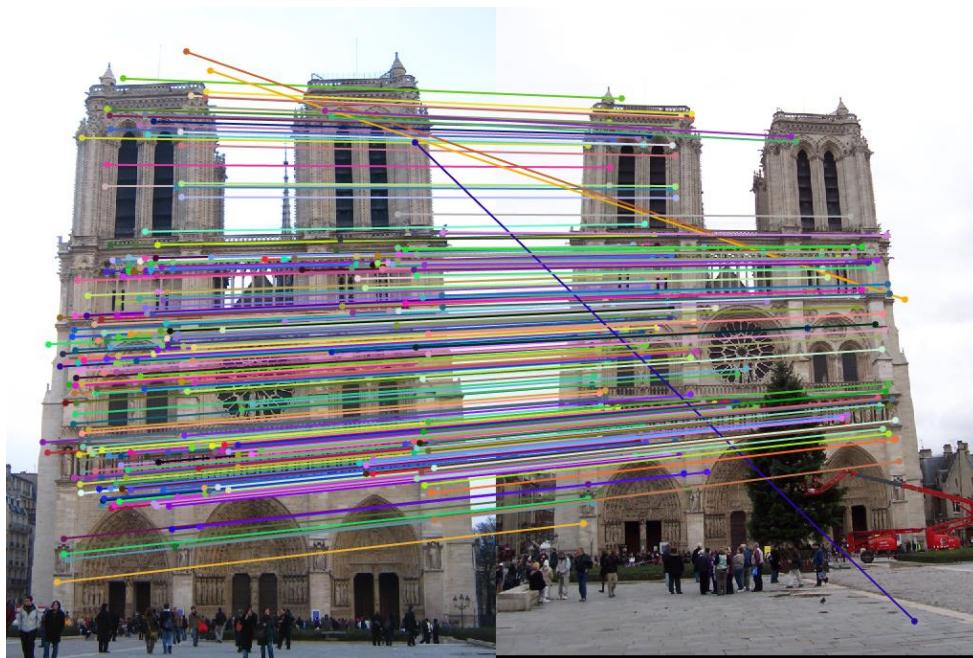
$$\lambda_1 < \lambda_{2..n}: \mathbf{x} = \mathbf{v}_1$$

Image Alignment



ANY PROBLEM ????

Image Alignment

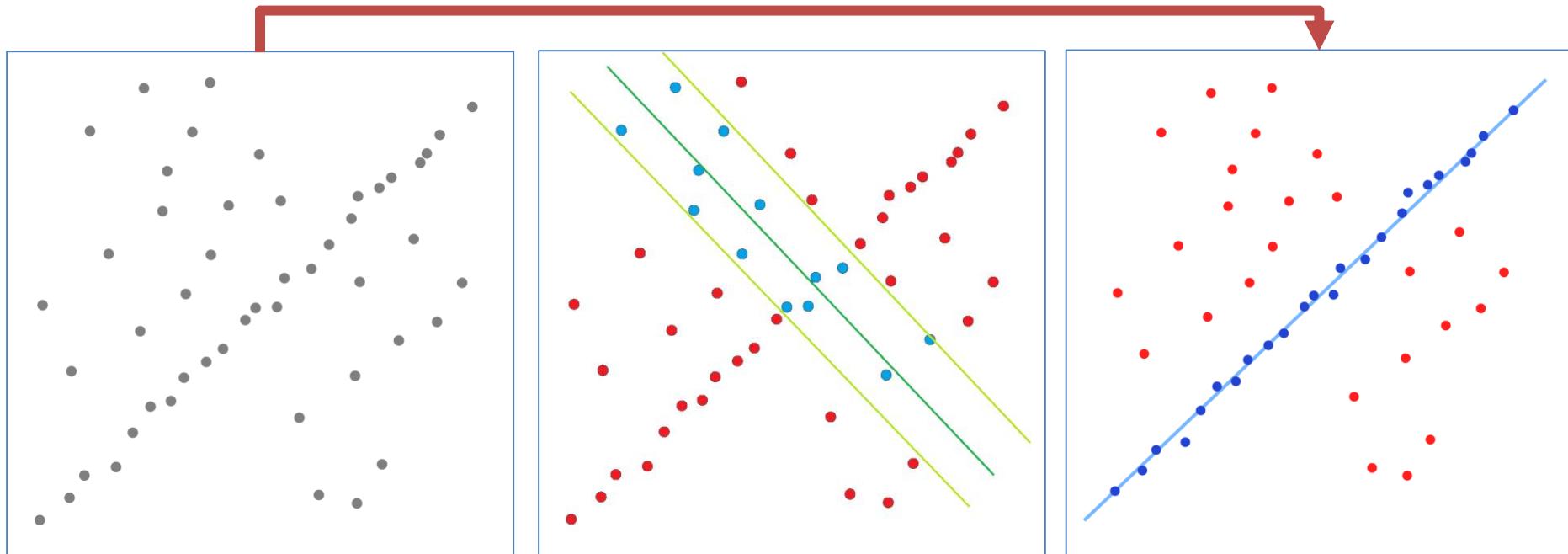


OUTLIERS will adversely affect parameter estimation!

RANDom SAmple Consensus : RANSAC

Using RANSAC for estimating geometric transforms in computer vision

- Random sample consensus, or RANSAC, is an iterative method for estimating a mathematical model from a data set that contains outliers.
- The RANSAC algorithm works by identifying the outliers in a data set and estimating the desired model using data that does not contain outliers.



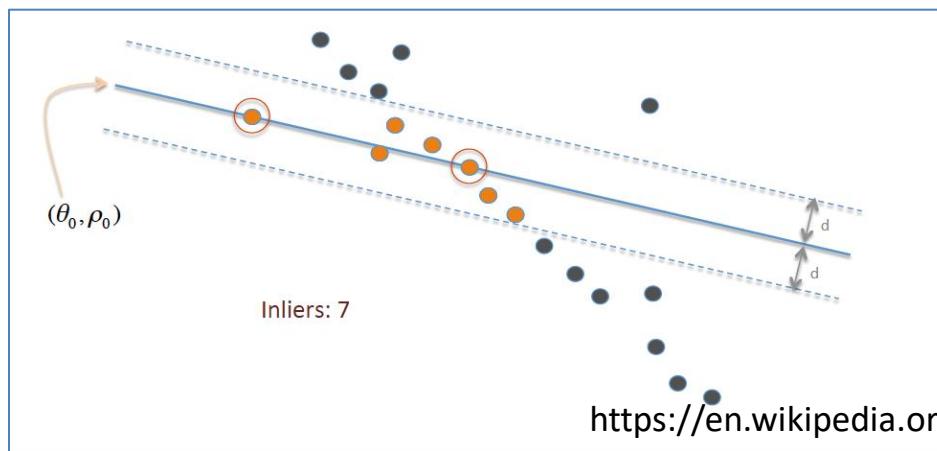
RANDOM SAMPLE CONSENSUS : RANSAC

RANSAC is accomplished with the following steps:

1. **Sample:** randomly select a subset of the data set (hypothetical inliers)
2. **Solve:** Fitting a model to the selected subset (hypothetical inliers)
3. **Score:** Test data against the fitted model.
 - All the data points (of the original data) that fit the estimated model well, according to some **model-specific loss function**, are called the consensus set (i.e., the set of inliers for the model).
 - The estimated model is reasonably good if sufficiently many data points have been classified as a part of the consensus set.

Repeat: steps 1-3 until best model is found with high confidence.

- (To converge to a sufficiently good model parameter set, this procedure is repeated a fixed number of times, each time producing
 - either the rejection of a model because too few points are a part of the consensus set,
 - or a refined model with a consensus set size larger than the previous consensus set.)



RANSAC Pseudocode

Given:

```
data - A set of observations.  
model - A model to explain the observed data points.  
n - The minimum number of data points required to estimate the model parameters.  
k - The maximum number of iterations allowed in the algorithm.  
t - A threshold value to determine data points that are fit well by the model (inlier).  
d - The number of close data points (inliers) required to assert that the model fits well to the data.
```

Return:

```
bestFit - The model parameters which may best fit the data (or null if no good model is found).
```

```
iterations = 0  
bestFit = null  
bestErr = something really large // This parameter is used to sharpen the model parameters to the best data fitting as  
iterations goes on.
```

```
while iterations < k do  
    maybeInliers := n randomly selected values from data  
    maybeModel := model parameters fitted to maybeInliers  
    confirmedInliers := empty set  
    for every point in data do  
        if point fits maybeModel with an error smaller than t then  
            add point to confirmedInliers  
        end if  
    end for  
    if the number of elements in confirmedInliers is > d then  
        // This implies that we may have found a good model.  
        // Now test how good it is.  
        betterModel := model parameters fitted to all the points in confirmedInliers  
        thisErr := a measure of how well betterModel fits these points  
        if thisErr < bestErr then  
            bestFit := betterModel  
            bestErr := thisErr  
        end if  
    end if  
    increment iterations  
end while  
  
return bestFit
```

RANSAC

Good

- Robust to outliers
- Applicable for larger number of objective function parameters than Hough transform
- Optimization parameters are easier to choose than Hough transform

Bad

- Computational time grows quickly with fraction of outliers and number of parameters
- Not as good for getting multiple fits (though one solution is to remove inliers after each fit and repeat)

Common applications

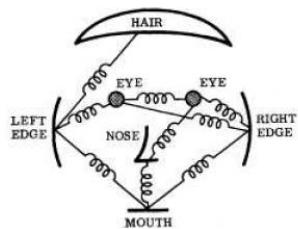
- Computing a homography (e.g., image stitching)
- Estimating fundamental matrix (relating two views)

Visual Object Recognition

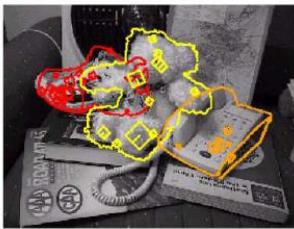
Filiz Bunyak

Dept. of Electrical Engineering & Computer Science
University of Missouri---Columbia
bunyak@missouri.edu, Naka 219

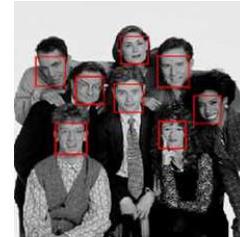
Various Kinds of Recognition



(a)



(b)



(c)



(d)



(e)



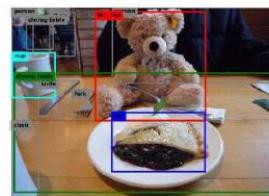
(f)



(g)



(h)



(i)

Figure 6.1 Various kinds of recognition: (a) face recognition with pictorial structures (Fischler and Elschlager 1973) © 1973 IEEE; (b) instance (known object) recognition (Lowe 1999) © 1999 IEEE; (c) real-time face detection (Viola and Jones 2004) © 2004 Springer; (d) feature-based recognition (Fergus, Perona, and Zisserman 2007) © 2007 Springer; (e) instance segmentation using Mask R-CNN (He, Gkioxari et al. 2017) © 2017 IEEE; (f) pose estimation (Güler, Neverova, and Kokkinos 2018) © 2018 IEEE; (g) panoptic segmentation (Kirillov, He et al. 2019) © 2019 IEEE; (h) video action recognition (Feichtenhofer, Fan et al. 2019); (i) image captioning (Lu, Yang et al. 2018) © 2018 IEEE.

What is “Recognition”?

- Verification: is that a lamp?



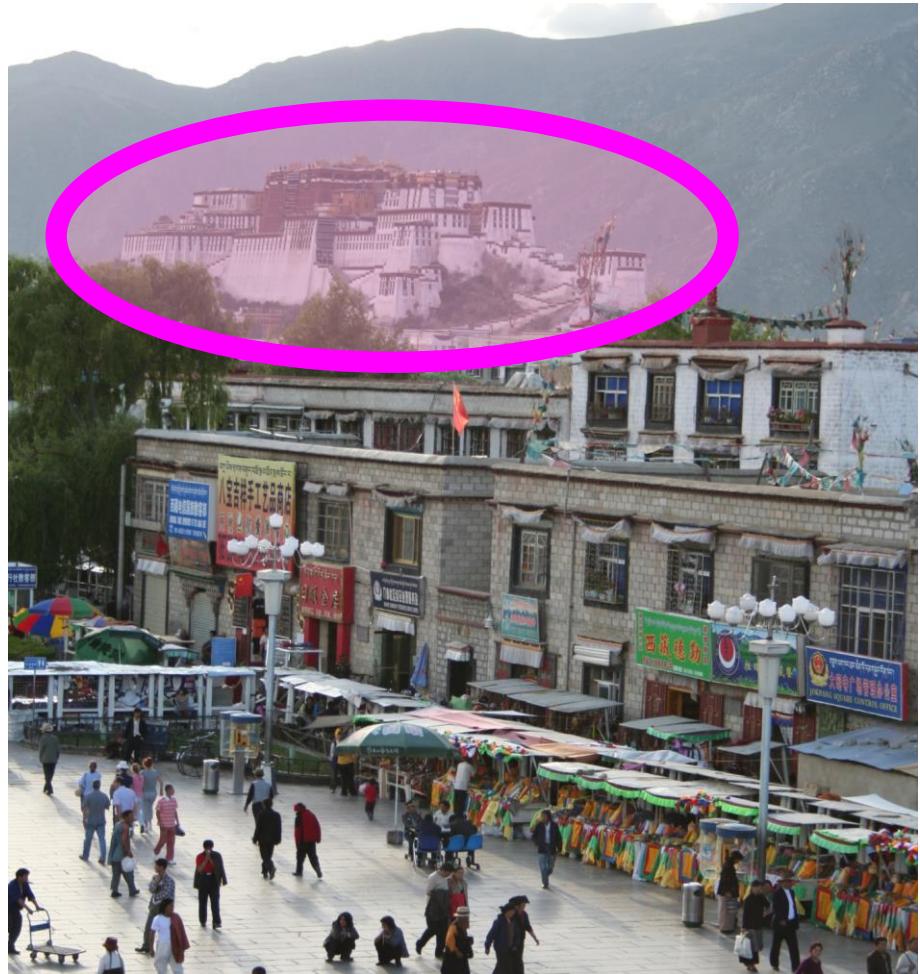
What is “Recognition”?

- Verification: is that a lamp?
- Detection: where are the people?



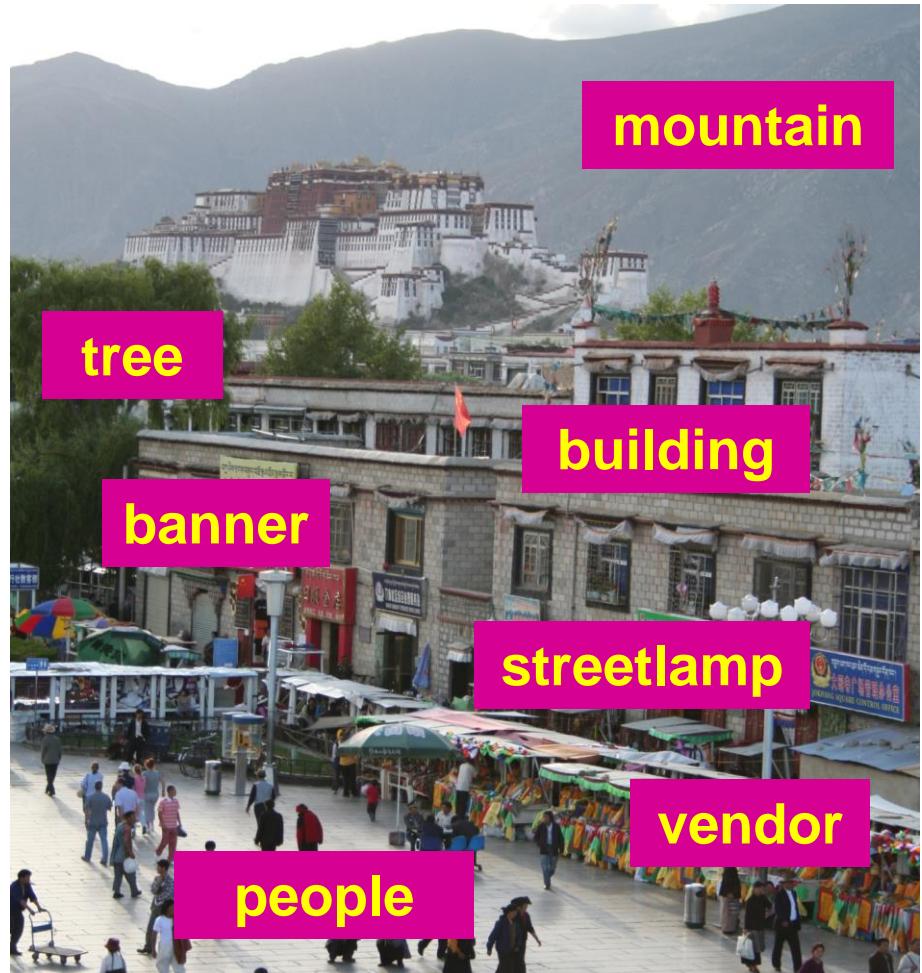
What is “Recognition”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?



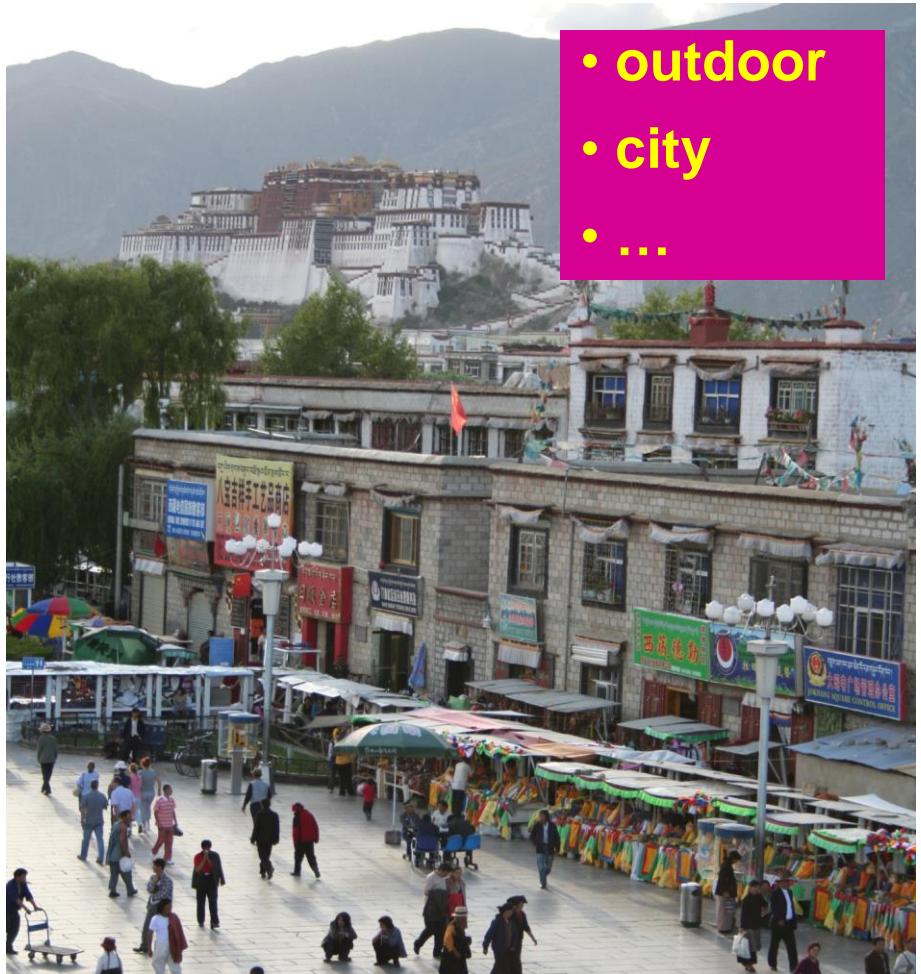
What is “Recognition”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization



What is “Recognition”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
- Scene and context categorization



What is “Recognition”?

- Verification: is that a lamp?
- Detection: where are the people?
- Identification: is that Potala Palace?
- Object categorization
- Scene and context categorization
- Activity / Event Recognition

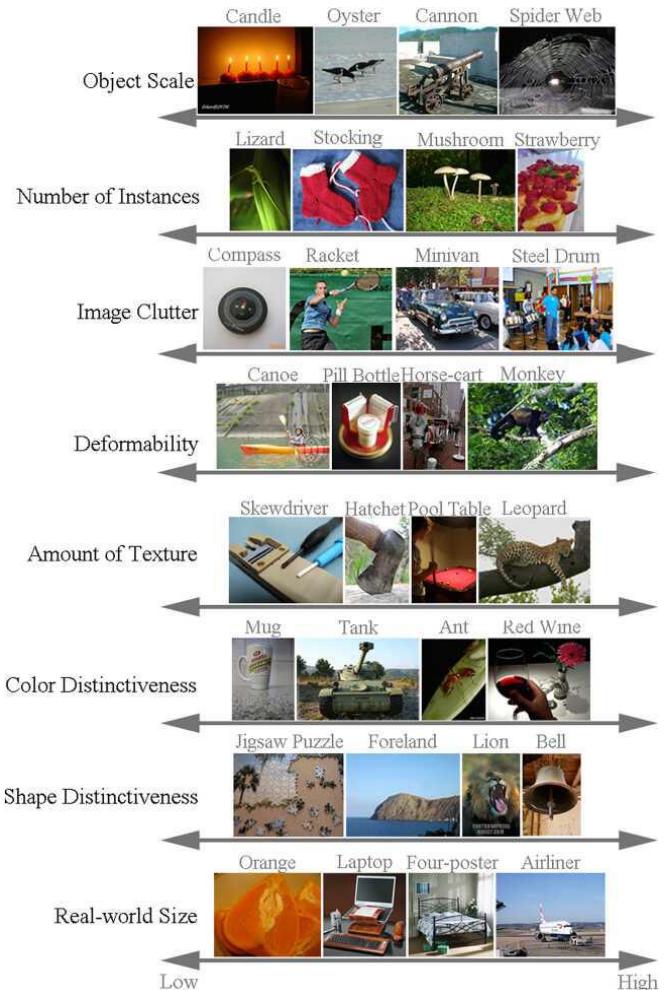
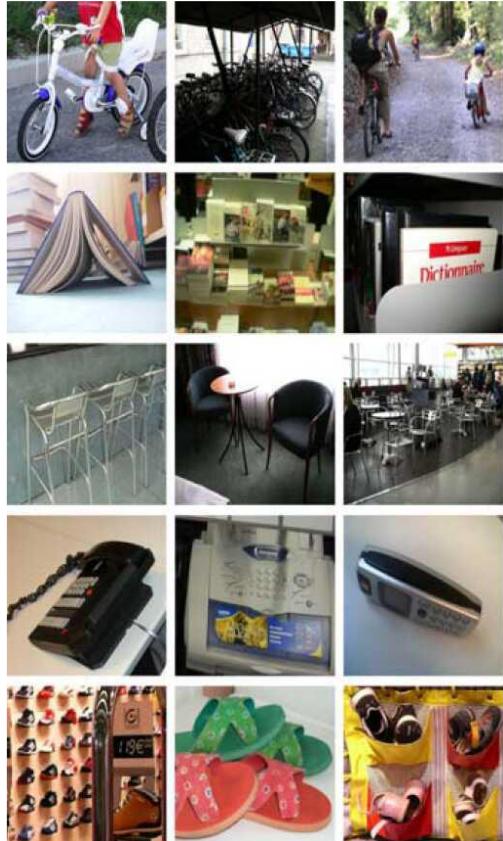


Instance Recognition



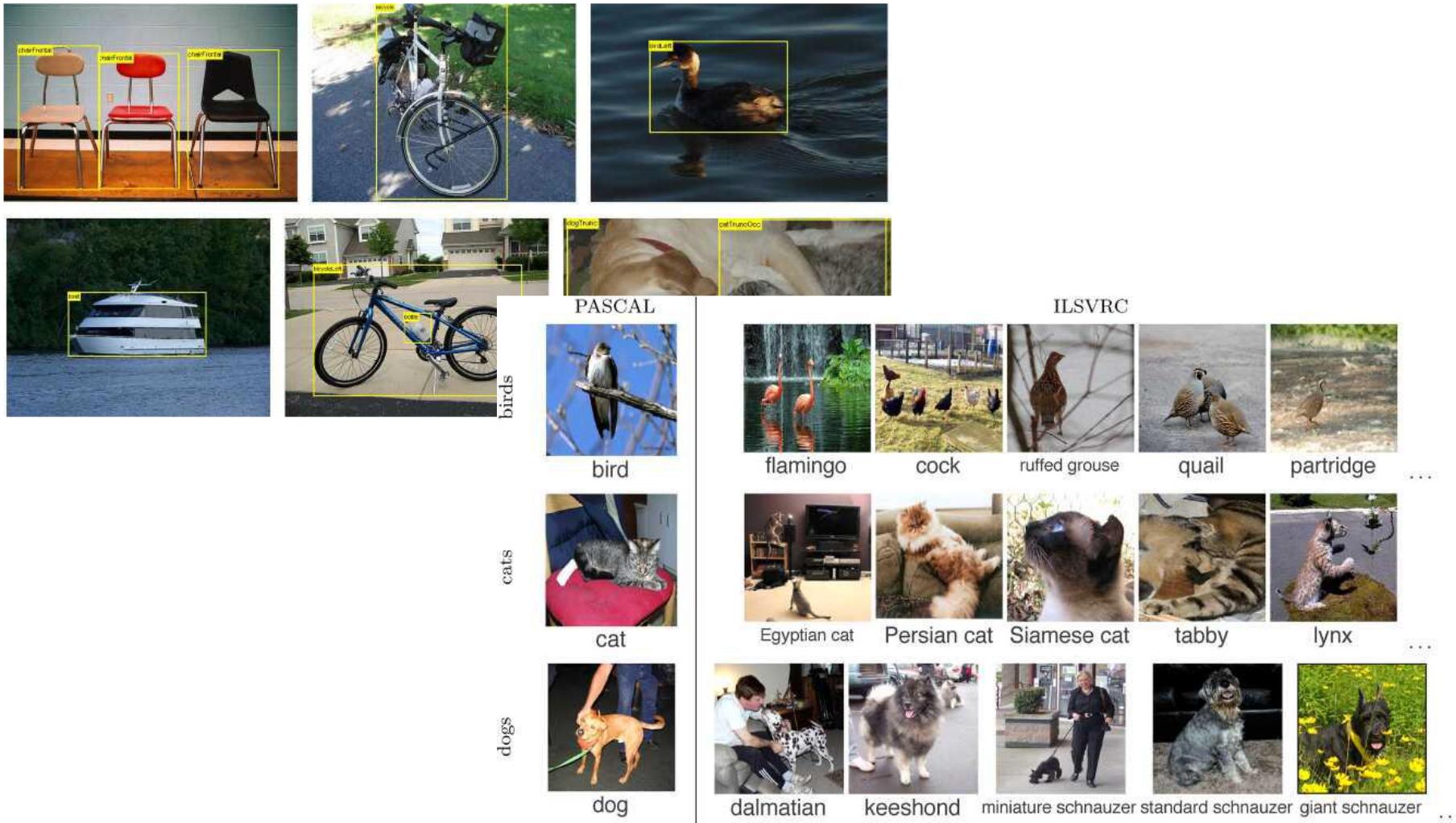
- Figure 6.2 Recognizing objects in a cluttered scene ([Lowe 2004](#)) © 2004 Springer. Two of the training images in the database are shown on the left. They are matched to the cluttered scene in the middle using SIFT features, shown as small squares in the right image.
- The affine warp of each recognized database image onto the scene is shown as a larger parallelogram in the right image.

Image Classification



- Figure 6.4 Challenges in image recognition: (a) sample images from the Xerox 10 class dataset ([Csurka, Dance et al. 2006](#)) © 2007 Springer; (b) axes of difficulty and variation from the ImageNet dataset ([Russakovsky, Deng et al. 2015](#)) © 2015 Springer

Image Classification Datasets



- Figure 6.5 Sample images from two widely used image classification datasets: (a) Pascal Visual Object Categories (VOC) ([Everingham, Eslami et al. 2015](#)) © 2015 Springer; (b) ImageNet ([Russakovsky, Deng et al. 2015](#)) © 2015 Springer.

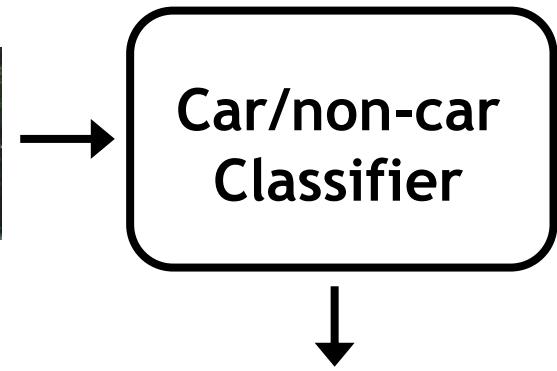
Part-1: Before Deep Learning Global Appearance + Sliding Window

References

- **AAAI'08 Tutorial on Visual Recognition** by [Kristen Grauman](#) (*University of Texas at Austin*) and [Bastian Leibe](#) (*ETH Zurich*)
<http://www.vision.ee.ethz.ch/~bleibe/teaching/tutorial-aaai08/>
 - Computer Vision CSE 576 by Steve Seitz, Rick Szeliski, Jiun-Hung Chen
<http://www.cs.washington.edu/education/courses/cse576/08sp/>
 - Image Understanding by Xuejin Chen, <http://staff.ustc.edu.cn/~xjchen99/teaching/teaching.html>
 - www.ens-lyon.fr/LIP/Arenaire/ERVision/features1_lyon.ppt
 - by Cordelia Schmid
 - **Multi-Image Matching using Multi-Scale Oriented Patches** by Matthew Brown, Richard Szeliski, and Simon Winder CVPR 2005
 - **Related Book:** K. Grauman and B. Leibe. *Visual Object Recognition. Synthesis Lectures on Artificial Intelligence and Machine Learning.* Morgan and Claypool Publishers, April 2011, Vol. 5, No. 2, Pages 1-181.(doi:10.2200/S00332ED1V01Y201103AIM011)
 - **Comp776: Computer Vision** by [Svetlana Lazebnik](#) (*University of North Carolina – Chapel Hill*)
 - Lectures:
 - Bag of features
 - Bag of features , classifiers
- **ICCV 2009 Short Course on Object Recognition** -- by Fei-Fei Li, Rob Fergus, and Antonio Torralba
<http://people.csail.mit.edu/torralba/shortCourseRLOC/>

Detection via classification: Main idea

Basic component: a binary classifier

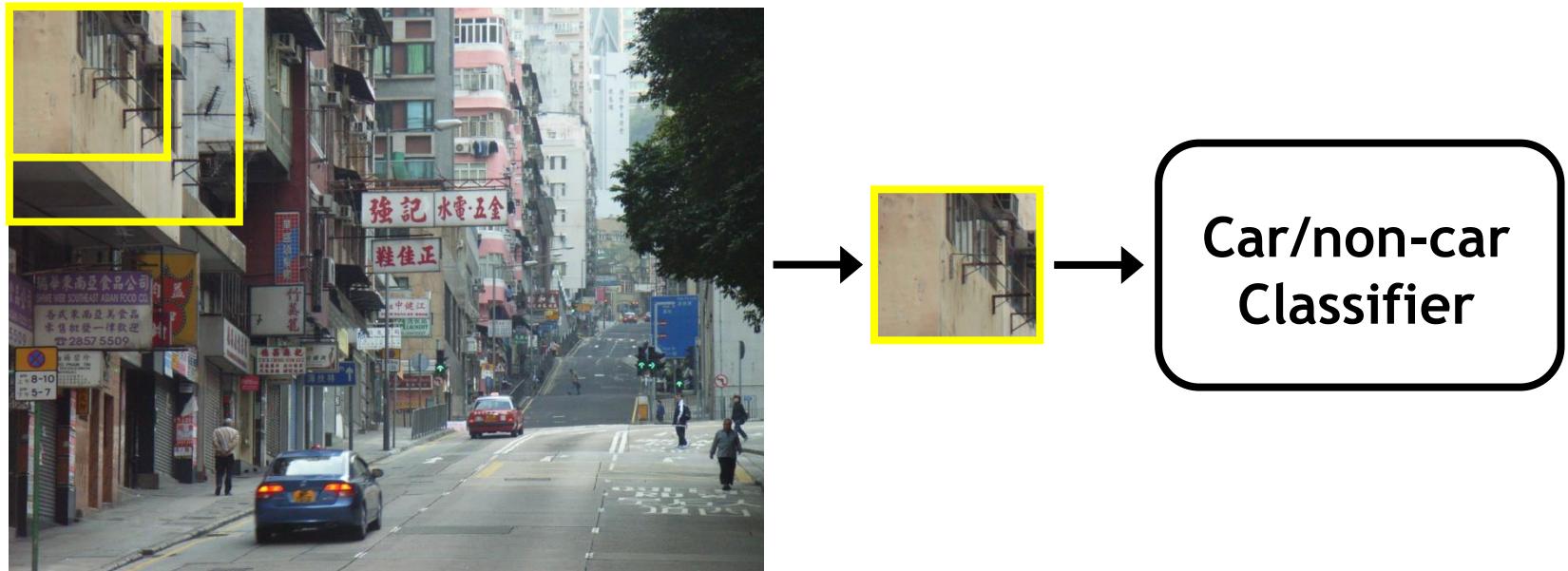


Yes, car.

No, not a car.

Detection via classification: Main idea

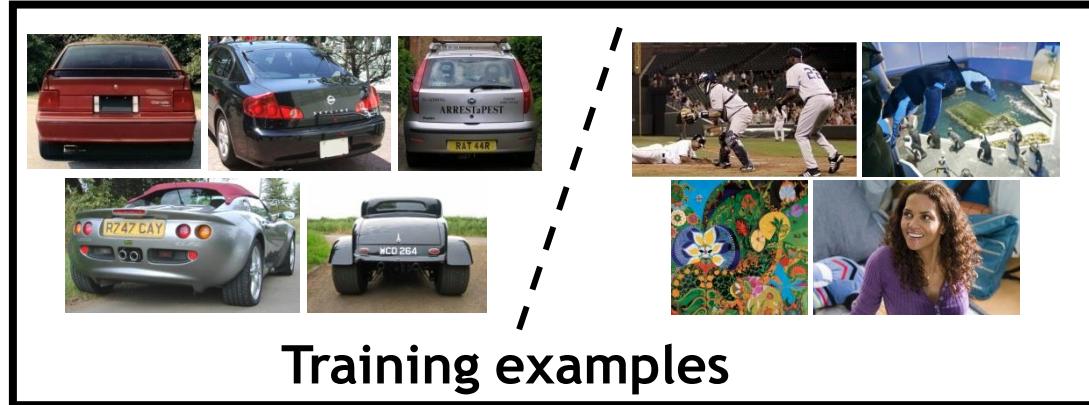
If object may be in a cluttered scene, slide a window around looking for it.



Detection via classification: Main idea

Fleshing out this pipeline a bit more, we need to:

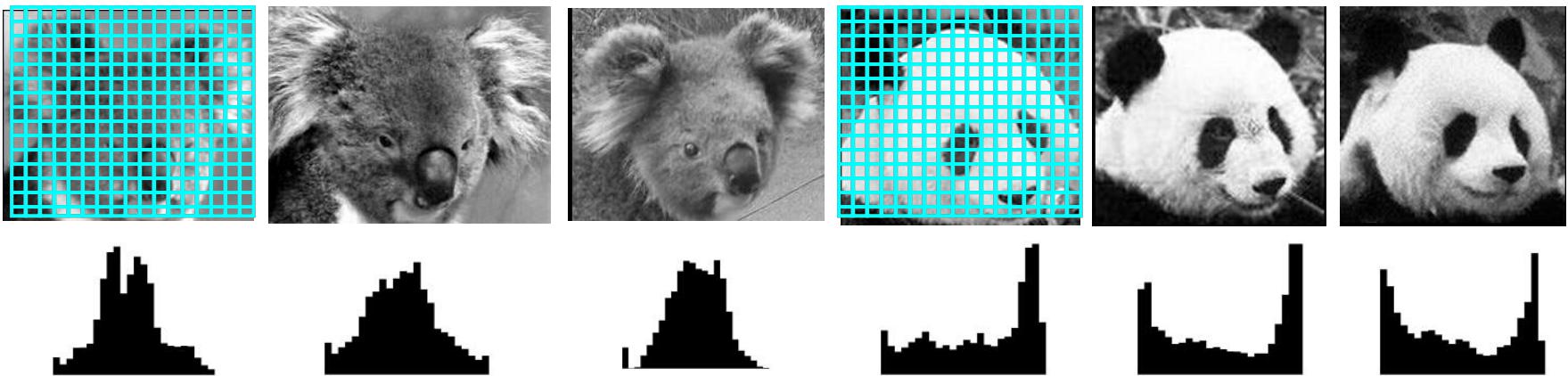
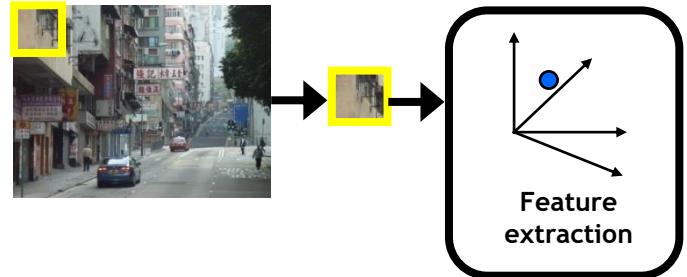
1. Obtain training data
2. Define features
3. Define classifier



Detection via classification: Main idea

- Consider all subwindows in an image
 - Sample at multiple scales and positions
- Make a decision per window:
 - “Does this contain object category X or not?”
- This section focus specifically on methods using a global representation (i.e., not part-based, not local features).

Feature extraction: global appearance

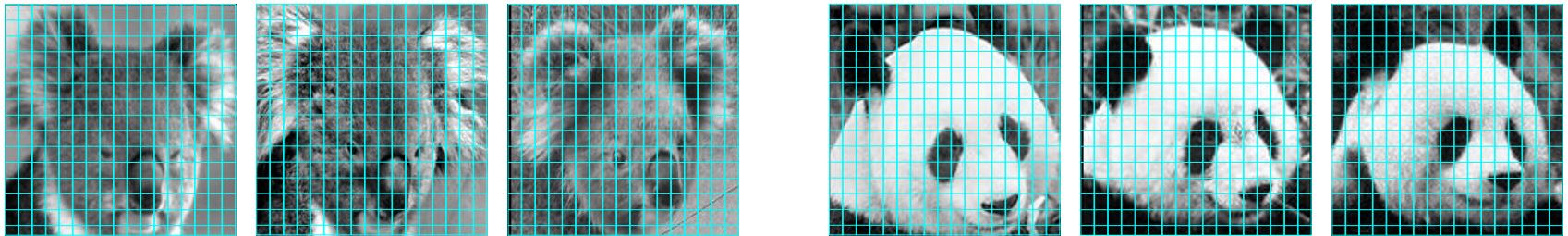


Simple holistic descriptions of image content

- grayscale / color histogram
- vector of pixel intensities

Feature extraction: global appearance

- Pixel-based representations sensitive to small shifts



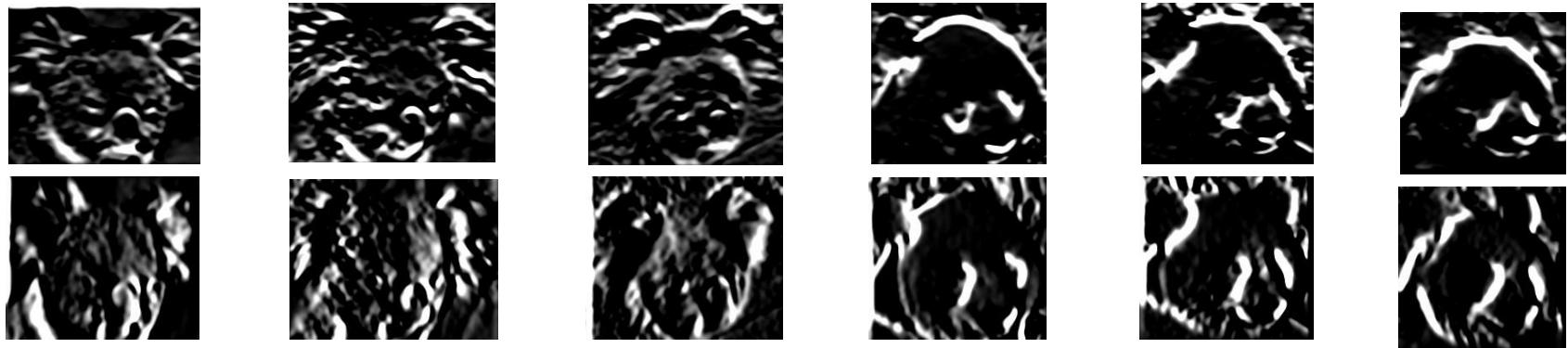
- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation



Cartoon example:
an albino koala

Gradient-based representations

- Consider edges, contours, and (oriented) intensity gradients



Gradient-based representations: Matching edge templates

- Example: Chamfer matching



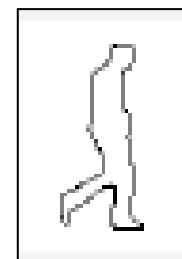
Input
image



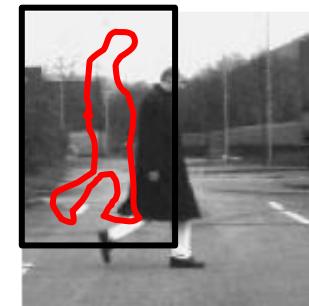
Edges
detected



Distance
transform



Template
shape



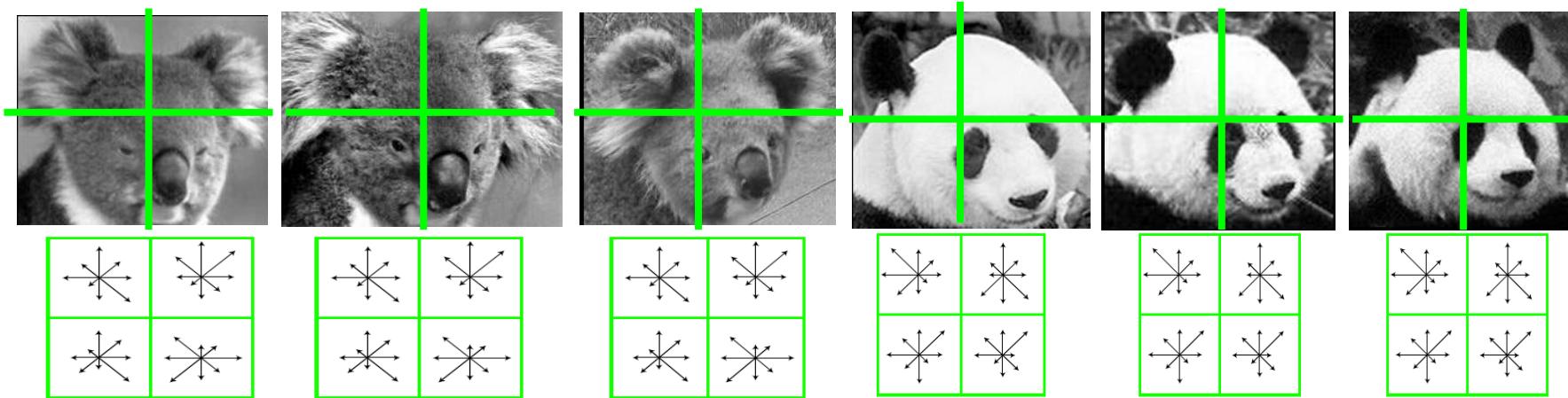
Best
match

At each window position,
compute average min
distance between points on
template (T) and input (I).

$$D_{chamfer}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

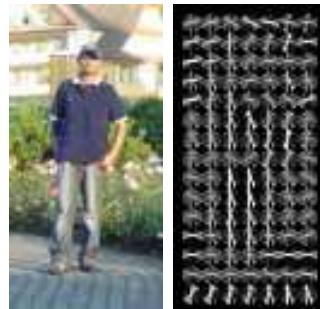
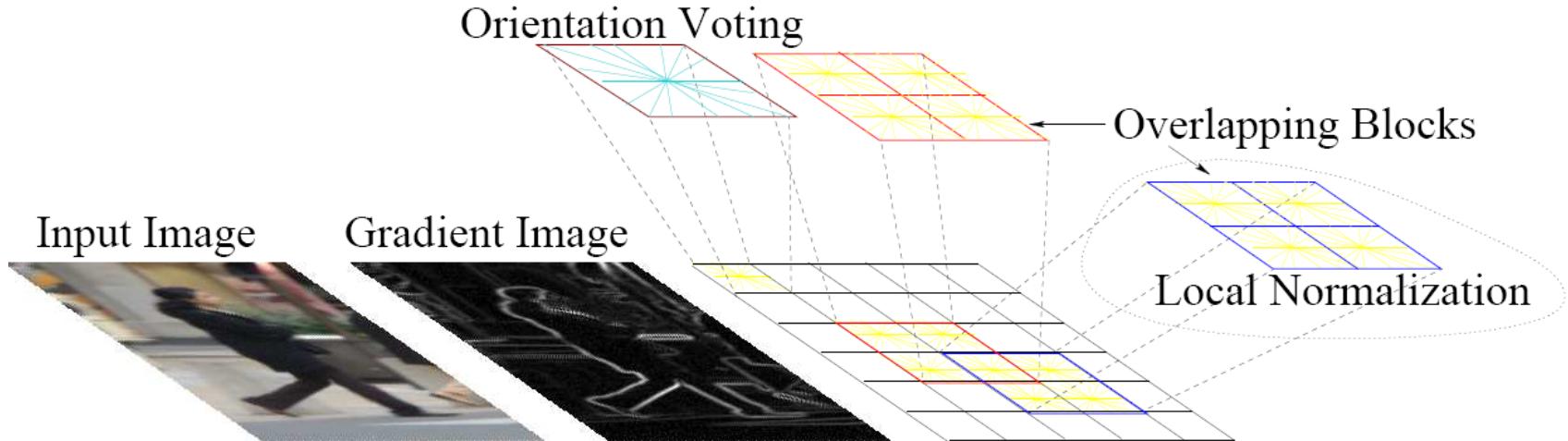
Gradient-based representations

- Consider edges, contours, and (oriented) intensity gradients



- Summarize local distribution of gradients with histogram
 - Locally orderless: offers invariance to small shifts and rotations
 - Contrast-normalization: try to correct for variable illumination

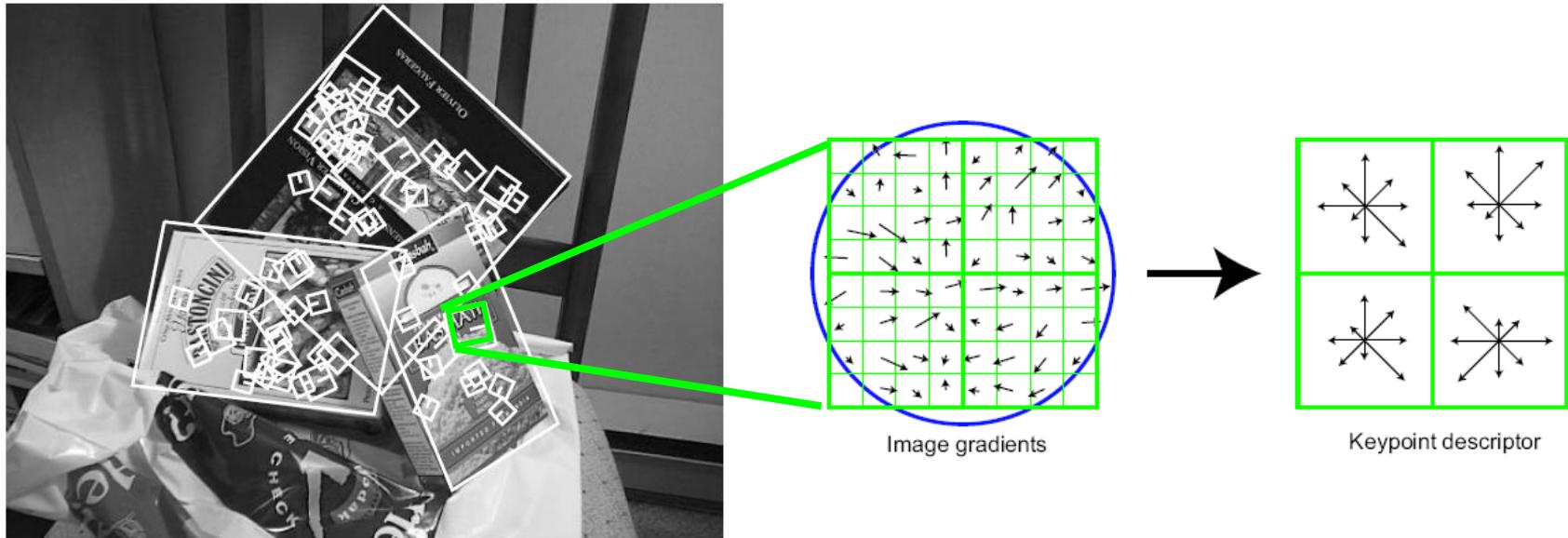
Gradient-based representations: Histograms of oriented gradients (HoG)



Map each grid cell in the input window to a histogram counting the gradients per orientation.

Code available:
<http://pascal.inrialpes.fr/soft/olt/>

Gradient-based representations: Scale Invariant Feature Transform (SIFT)



**Local patch descriptor
(more on this later)**

Code: <http://vision.ucla.edu/~vedaldi/code/sift/sift.html>

Binary: <http://www.cs.ubc.ca/~lowe/keypoints/>

Lowe, ICCV 1999

Classifier construction

- How to compute a decision for each subwindow?



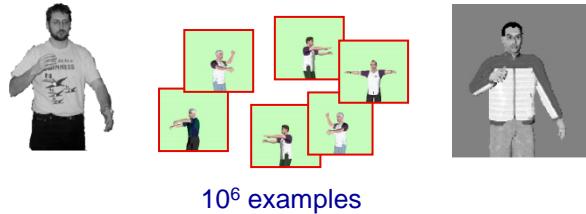
Image feature

Discriminative vs. generative models

- **Generative:**
 - + possibly interpretable
 - + can draw samples
 - - models variability unimportant to classification task
 - - often hard to build good model with few parameters
- **Discriminative:**
 - + appealing when infeasible to model data itself
 - + excel in practice
 - - often can't provide uncertainty in predictions
 - - non-interpretable

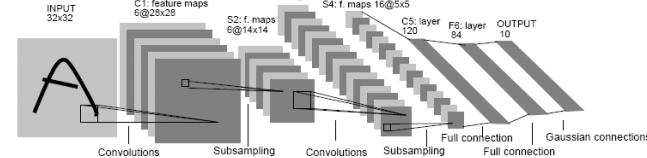
Discriminative methods

Nearest neighbor



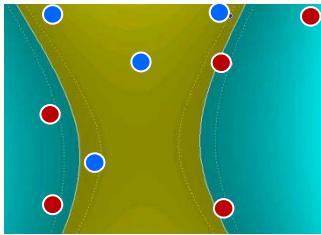
Shakhnarovich, Viola, Darrell 2003
Berg, Berg, Malik 2005...

Neural networks



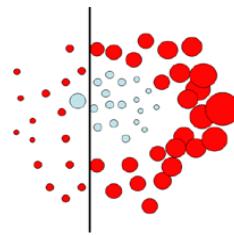
LeCun, Bottou, Bengio, Haffner 1998
Rowley, Baluja, Kanade 1998
...

Support Vector Machines



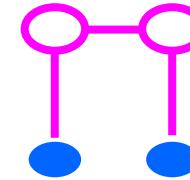
Guyon, Vapnik
Heisele, Serre, Poggio,
2001,...

Boosting



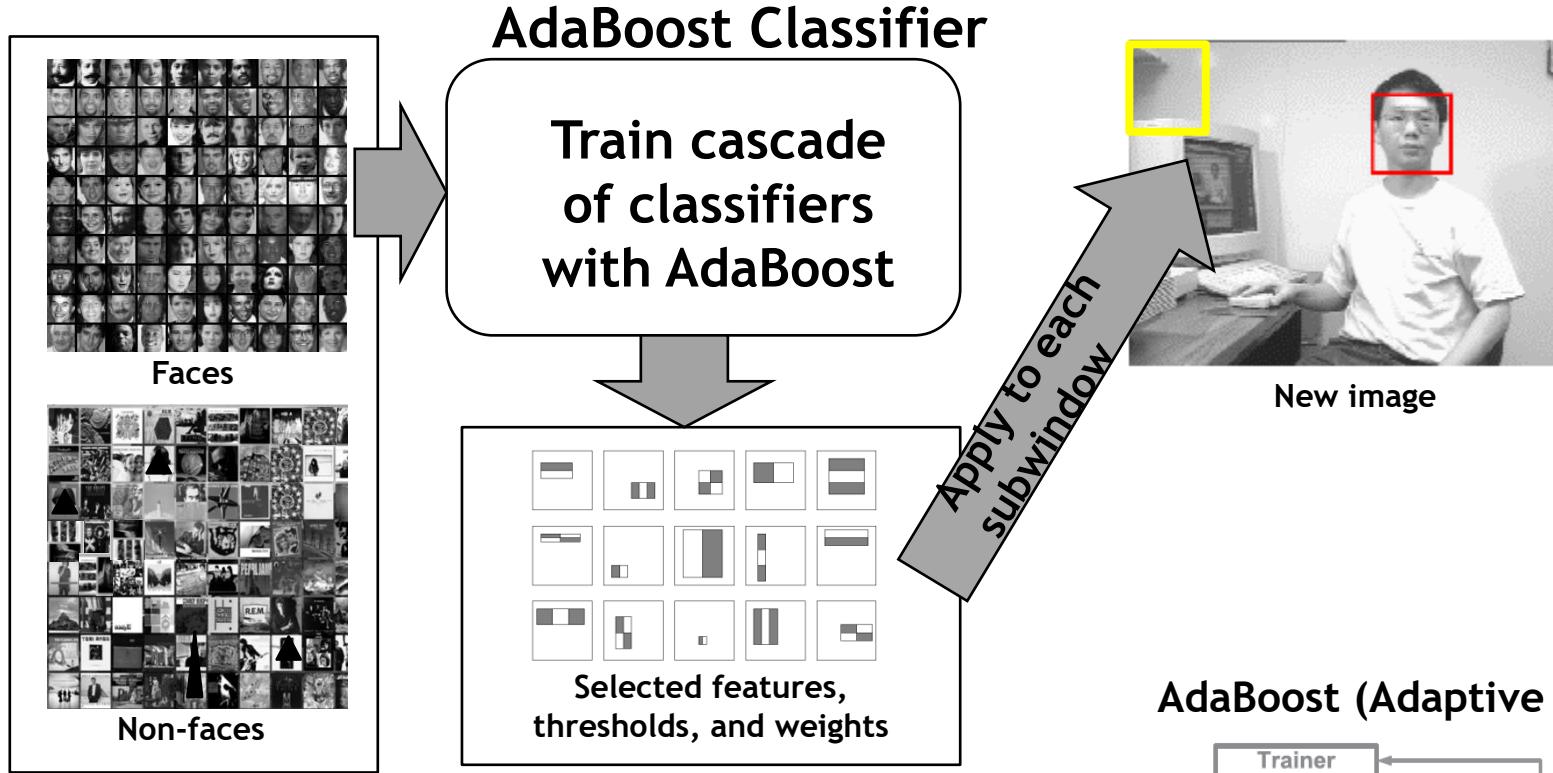
Viola, Jones 2001,
Torralba et al. 2004,
Opelt et al. 2006,...

Conditional Random Fields



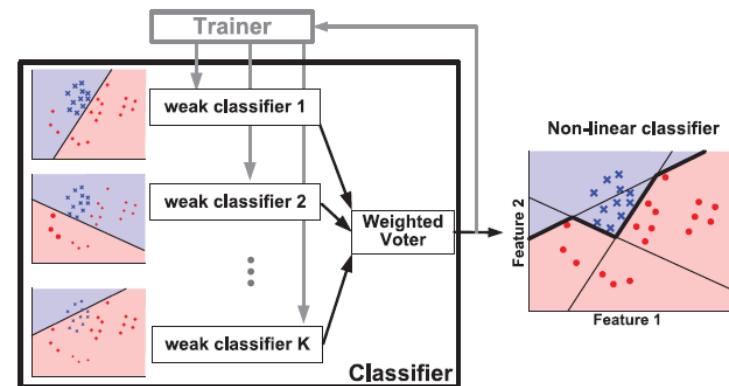
McCallum, Freitag, Pereira
2000; Kumar, Hebert 2003
...

Viola-Jones Face Detector: Summary

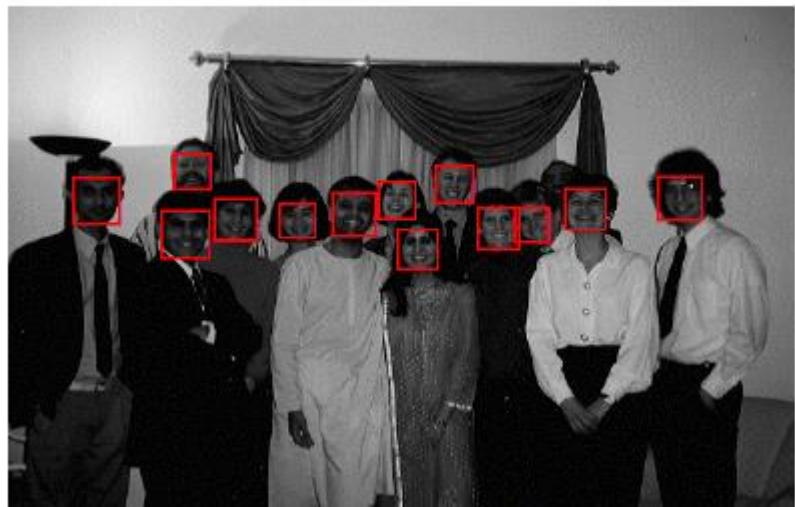
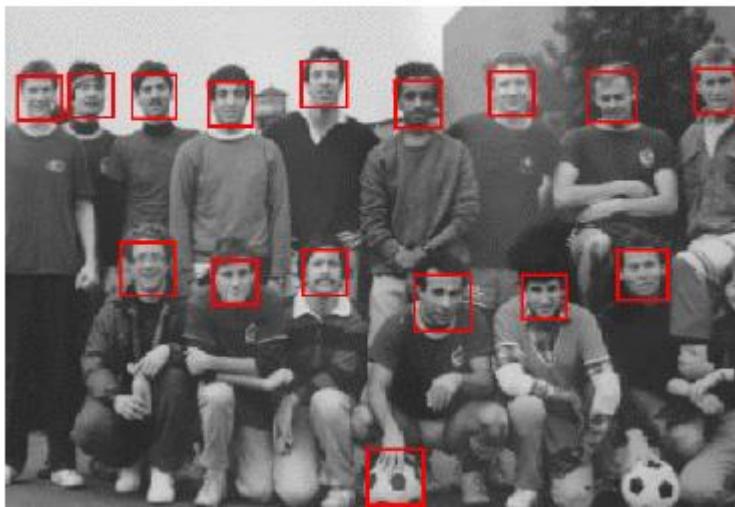
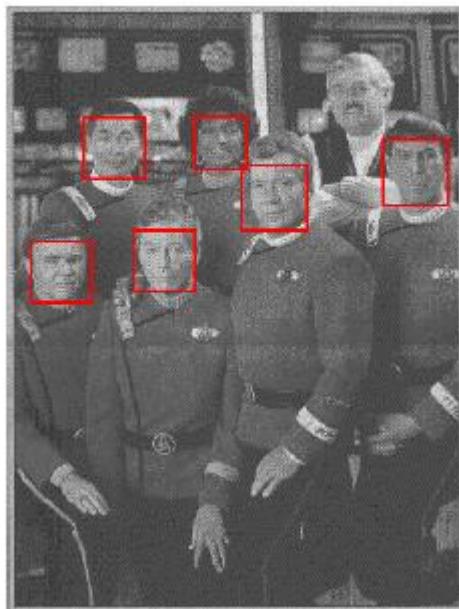
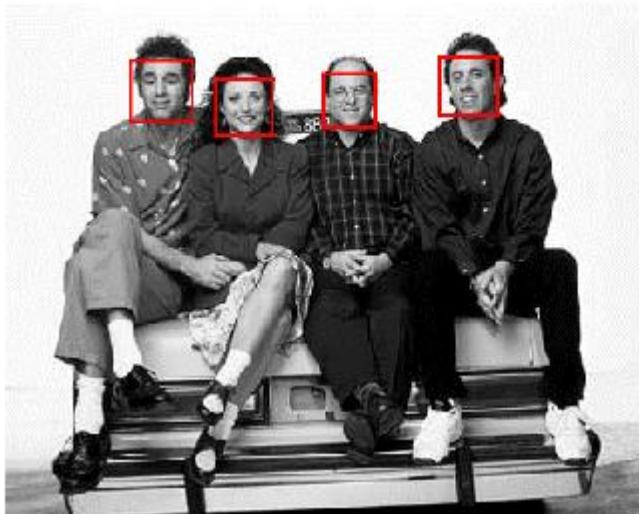


- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- Implementation available in OpenCV

AdaBoost (Adaptive Boosting)

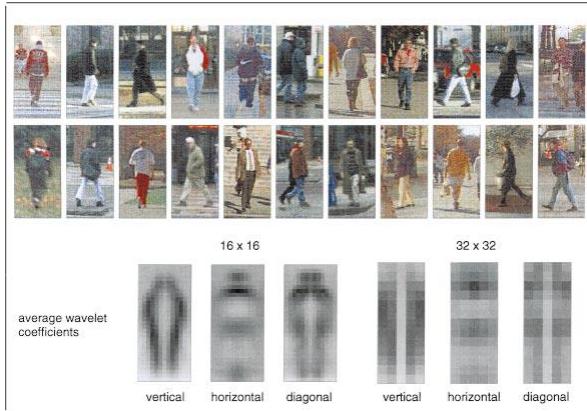


Viola-Jones Face Detector: Results



Pedestrian detection

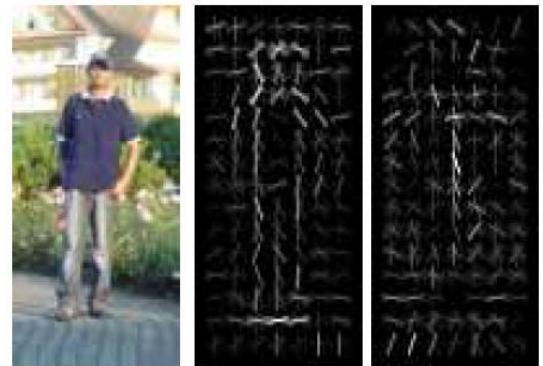
- Detecting upright, walking humans also possible using sliding window's appearance/texture; e.g.,



SVM with Haar wavelets
[Papageorgiou & Poggio, IJCV
2000]



Space-time rectangle
features [Viola, Jones &
Snow, ICCV 2003]



SVM with HoGs [Dalal &
Triggs, CVPR 2005]

Highlights

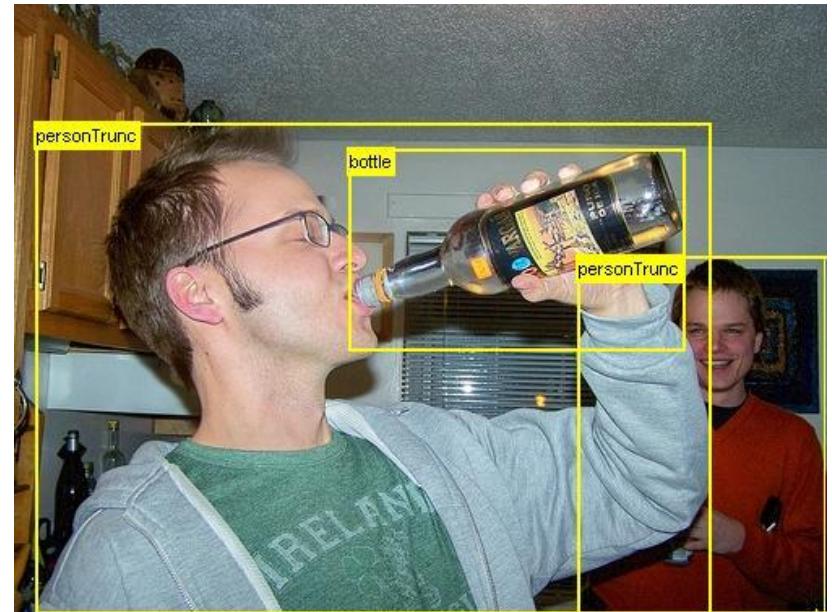
- **Sliding window detection and global appearance descriptors:**
 - Simple detection protocol to implement
 - Good feature choices critical
 - Past successes for certain classes

Limitations

- **High computational complexity**
 - For example: 250,000 locations x 30 orientations x 4 scales = 30,000,000 evaluations!
 - If training binary detectors independently, means cost increases linearly with number of classes
- **With so many windows, false positive rate better be low**

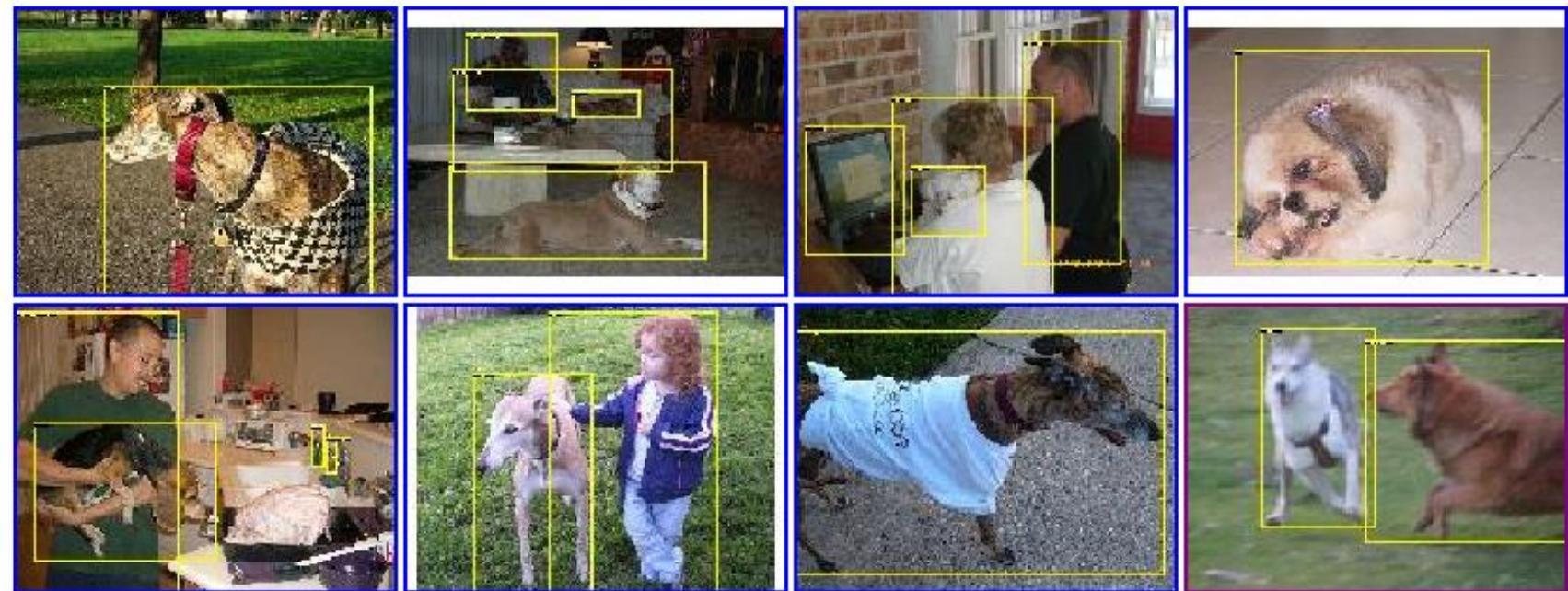
Limitations (continued)

- Not all objects are “box” shaped



Limitations (continued)

- Non-rigid, deformable objects not captured well with representations assuming a fixed 2d structure; or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions



Limitations (continued)

- If considering windows in isolation, context is lost



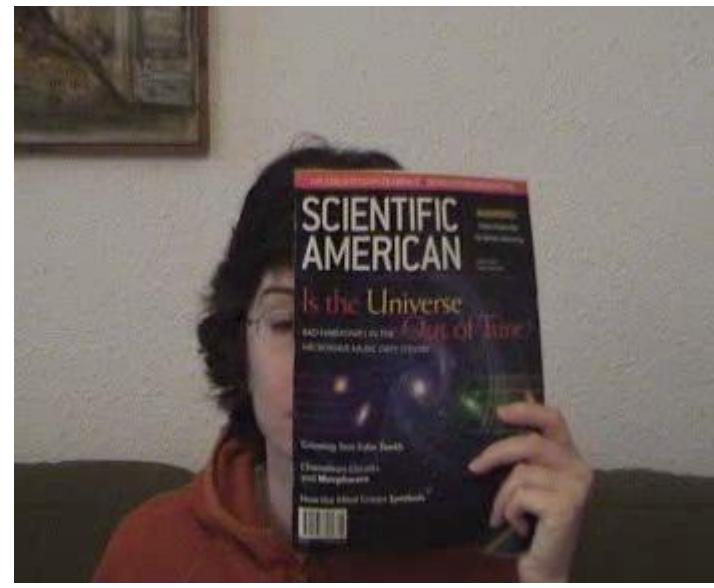
Sliding window



Detector's view

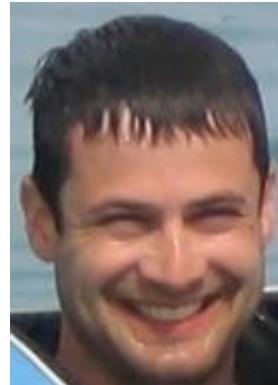
Limitations (continued)

- In practice, often entails large, cropped training set (expensive)
- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions



Global representations: limitations

- Success may rely on alignment
-> sensitive to viewpoint
- All parts of the image or window impact the description -> sensitive to occlusion, clutter

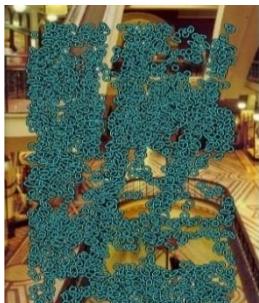


Local representations

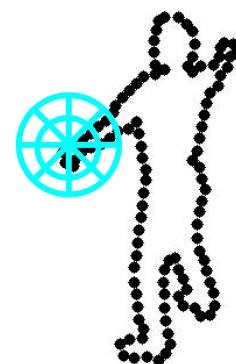
- Describe component regions or patches separately.
- Many options for detection & description...



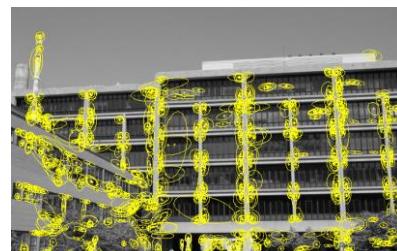
SIFT [Lowe 99]



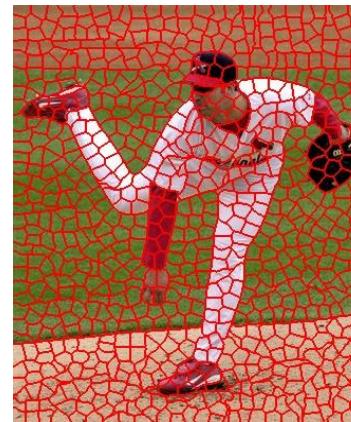
Salient regions
[Kadir 01]



Shape context
[Belongie 02]



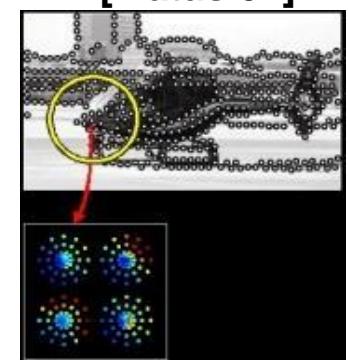
Harris-Affine
[Mikolajczyk 04]



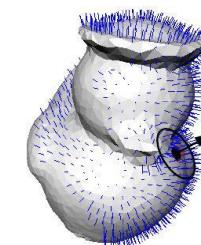
Superpixels
[Ren et al.]



Maximally Stable
Extremal Regions
[Matas 02]



Geometric Blur
[Berg 05]



Spin images
[Johnson 99]

Invariant local features

Subset of local feature types designed to be invariant to

- Scale
- Translation
- Rotation
- Affine transformations
- Illumination

- 1) Detect interest points
- 2) Extract descriptors

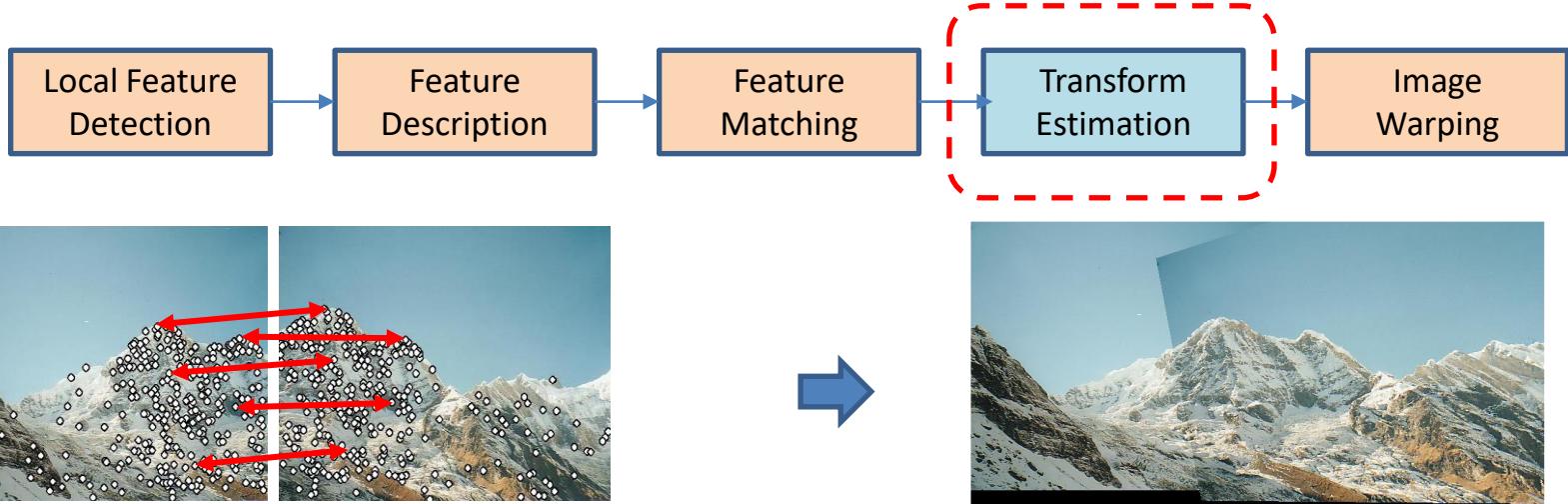


y_1
 y_2
...
 y_d

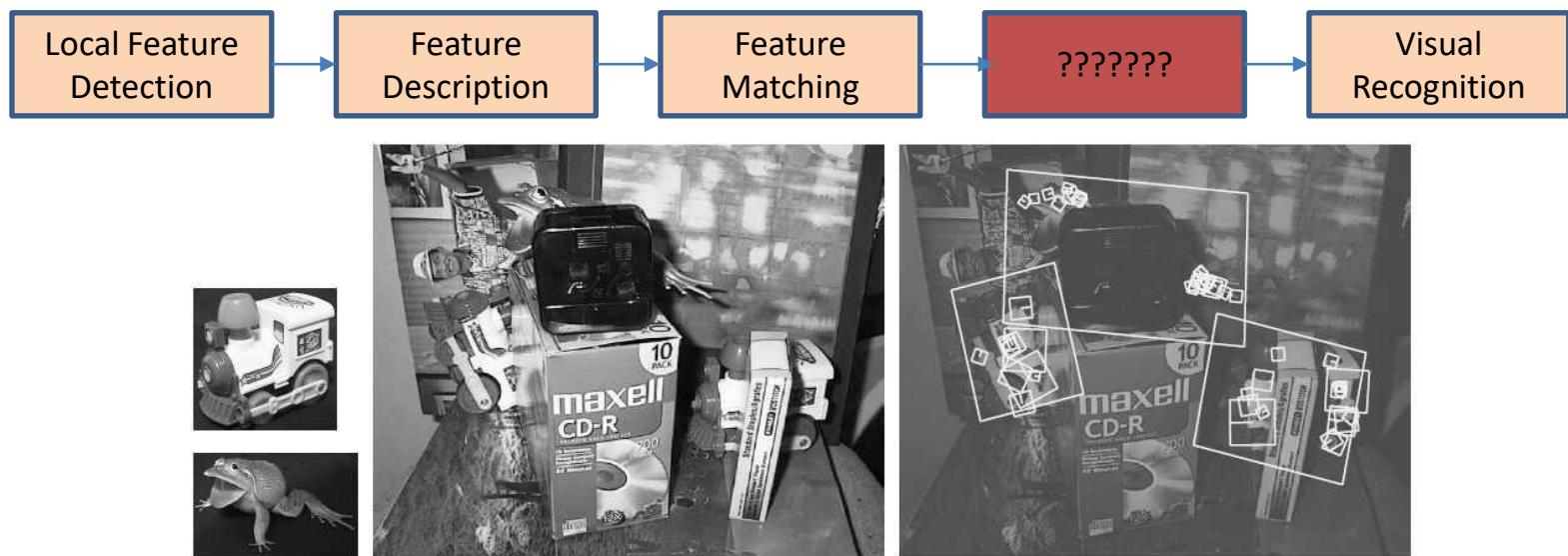
x_1
 x_2
...
 x_d

[Mikolajczyk01, Matas02, Tuytelaars04, Lowe99, Kadir01, ...]

Image Alignment



Visual Recognition



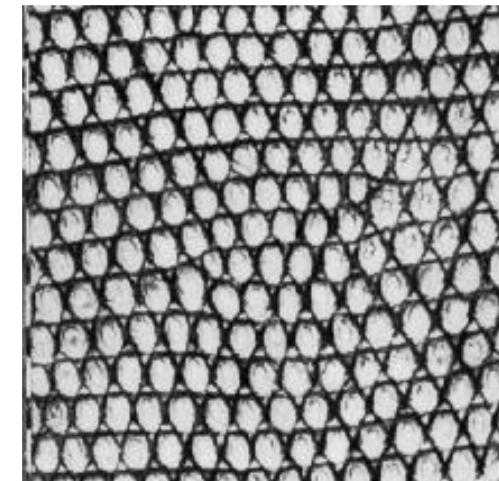
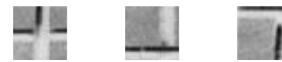
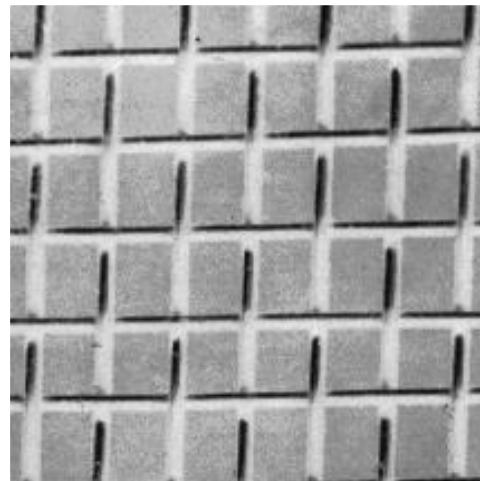
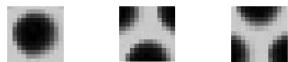
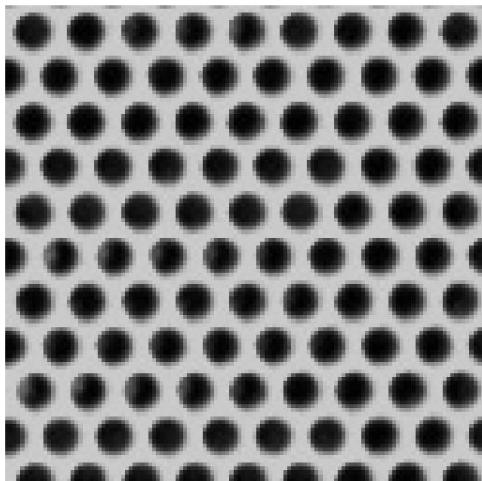
Bag-of-features models



Slide adapted from
Svetlana Lazebnik

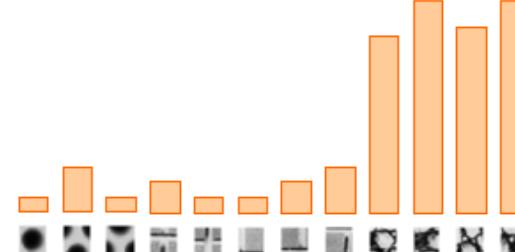
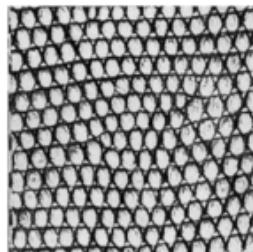
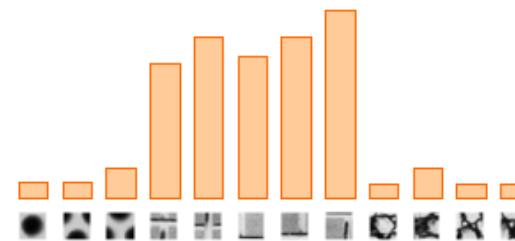
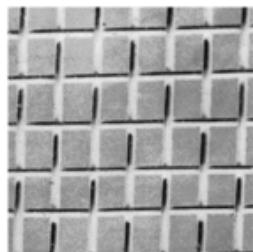
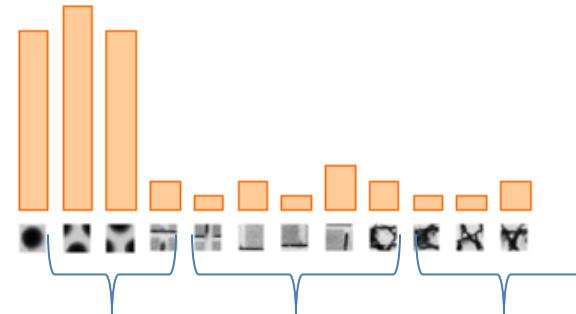
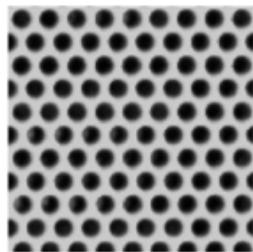
Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Origin 1: Texture recognition

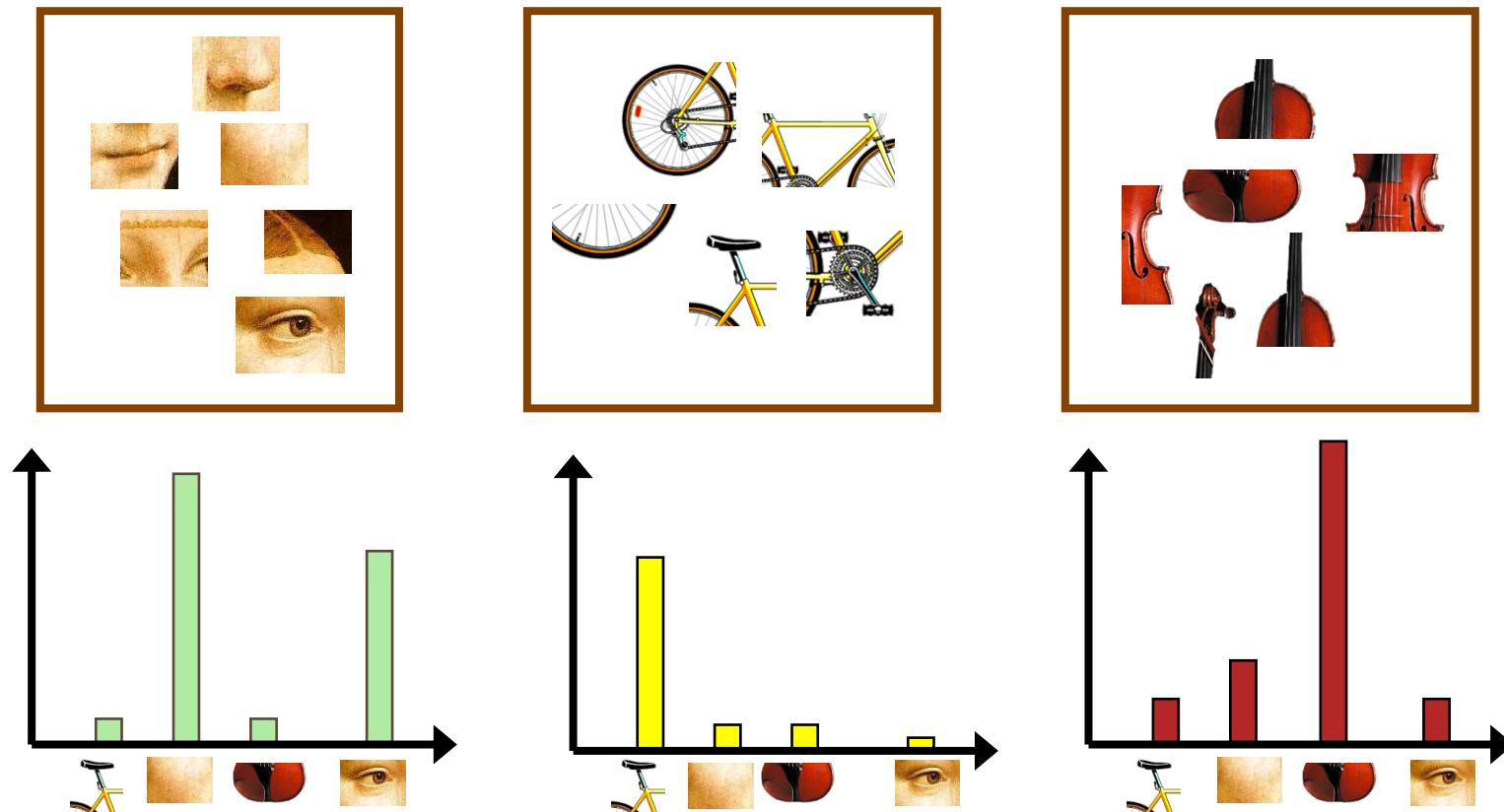


Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Slide adapted from Svetlana Lazebnik

Bag-of-features steps

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”



Bag of Features

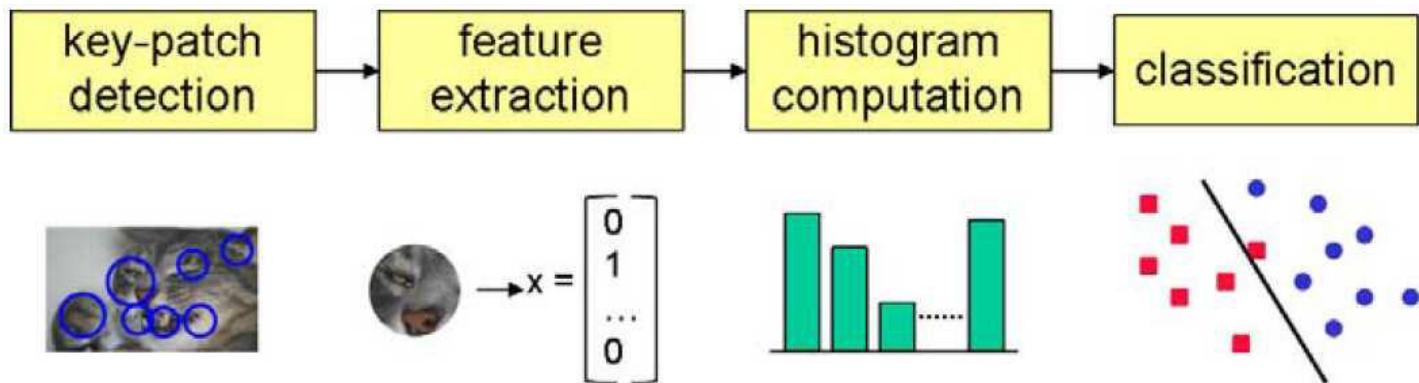
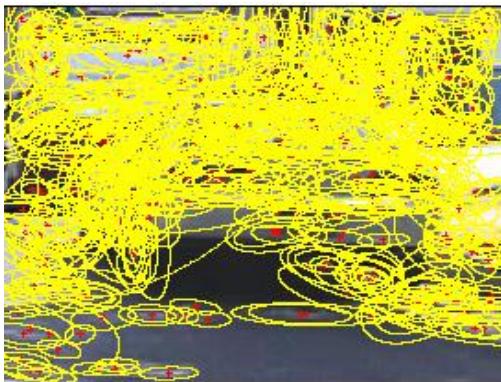
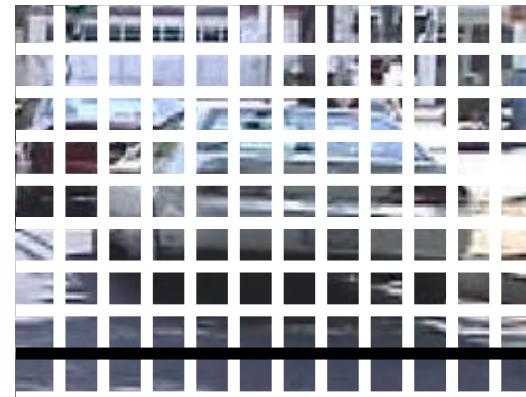


Figure 6.6 A typical processing pipeline for a bag-of-words category recognition system (Csurka, Dance et al. 2006) © 2007 Springer. Features are first extracted at keypoints and then quantized to get a distribution (histogram) over the learned visual words (feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm, such as a support vector machine.

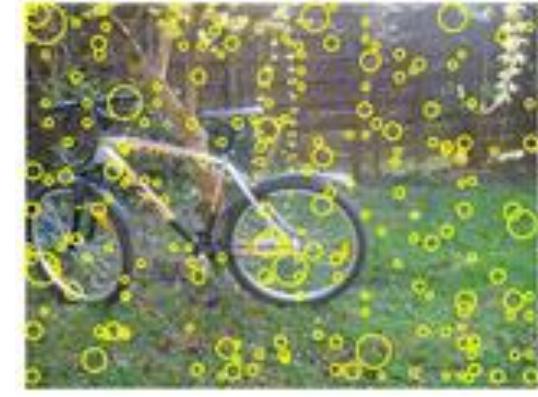
1. Feature extraction: Sampling strategies



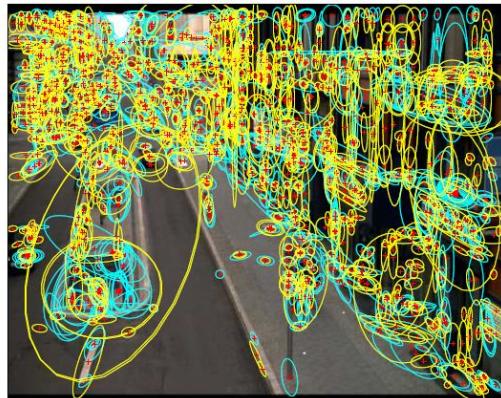
Sparse, at
interest points



Dense, uniformly



Randomly



Multiple interest
operators

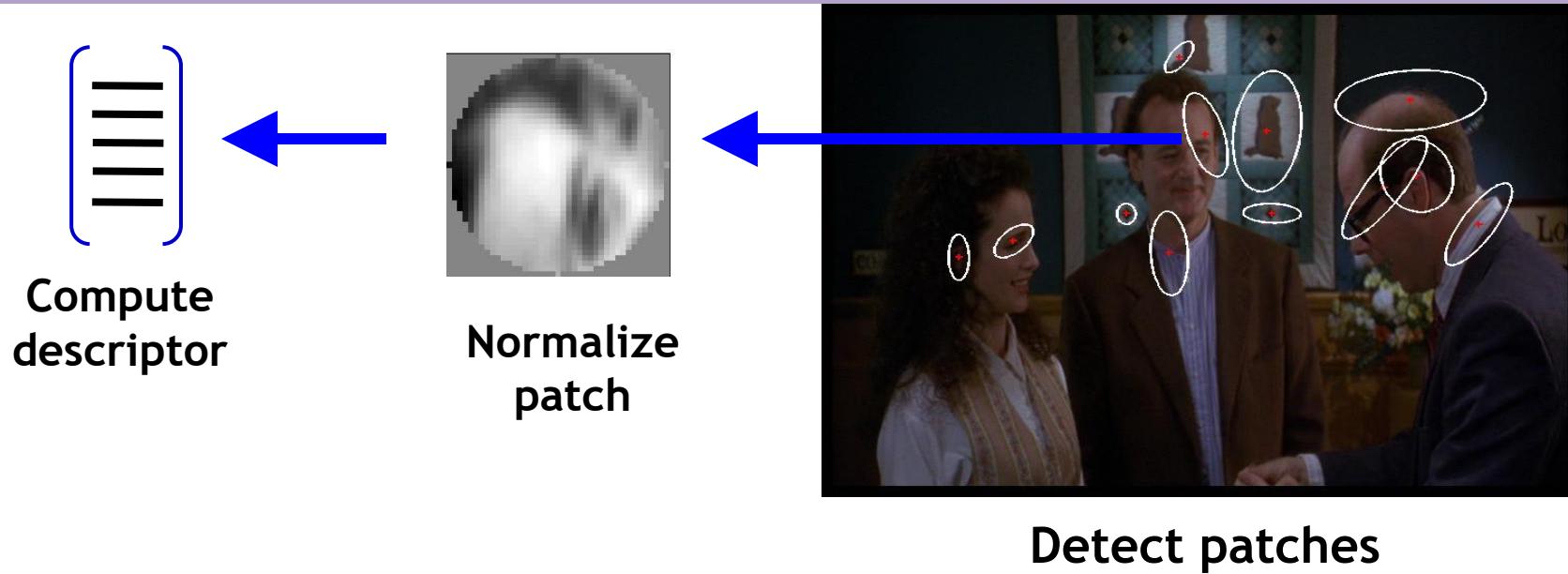
Image credits: F-F. Li, E. Nowak, J. Sivic

- To find specific, textured objects, sparse sampling from interest points often more reliable.
- Multiple complementary interest operators offer more image coverage.
- For object categorization, dense sampling offers better coverage.

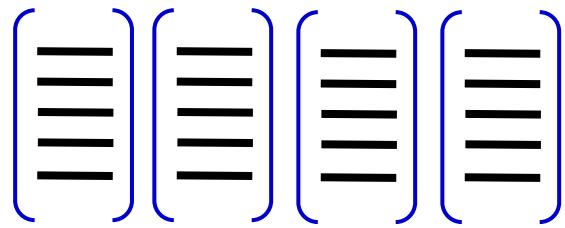
[See Nowak, Jurie & Triggs, ECCV 2006]

K. Grauman, B. Leibe

1. Feature extraction



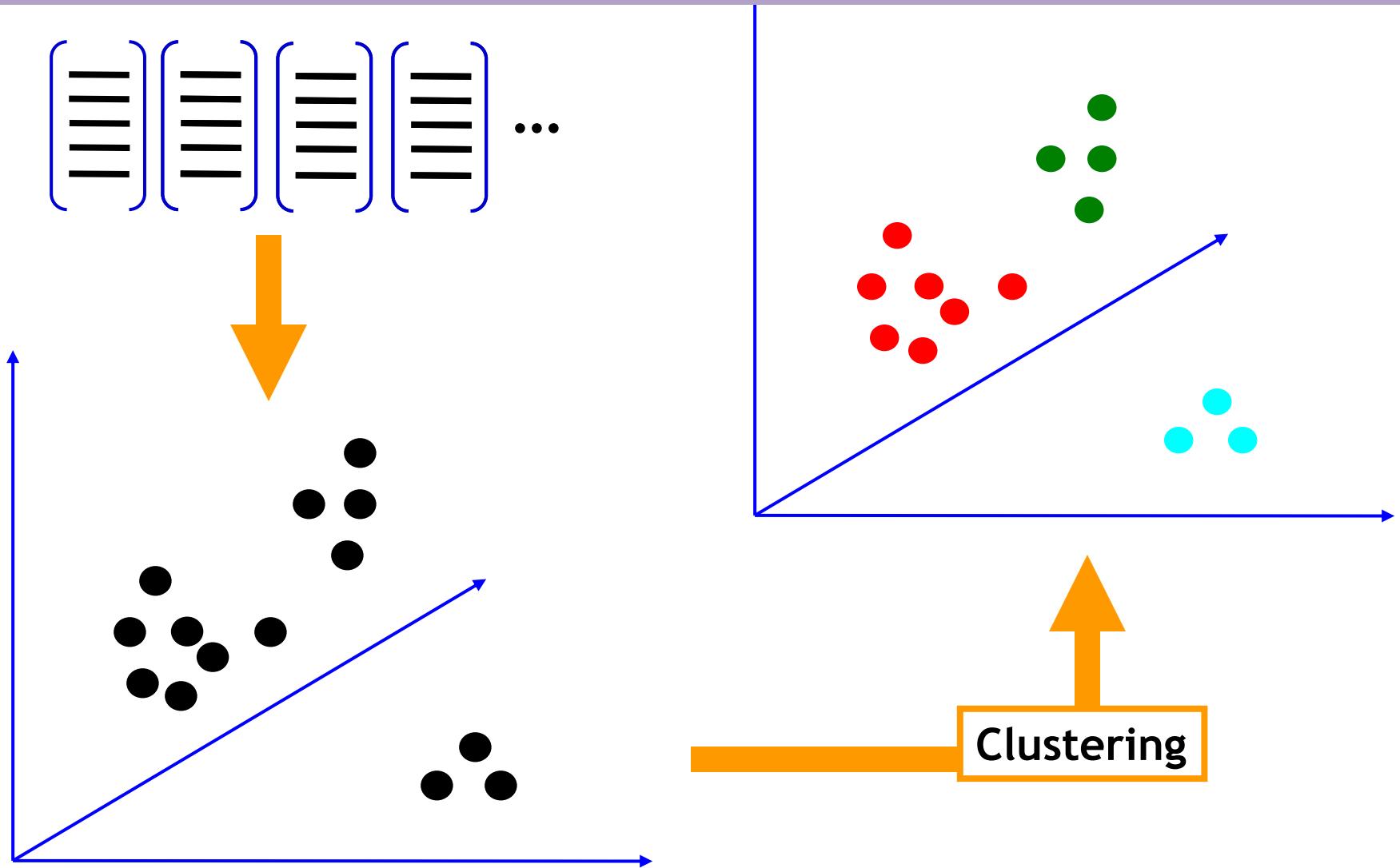
1. Feature extraction



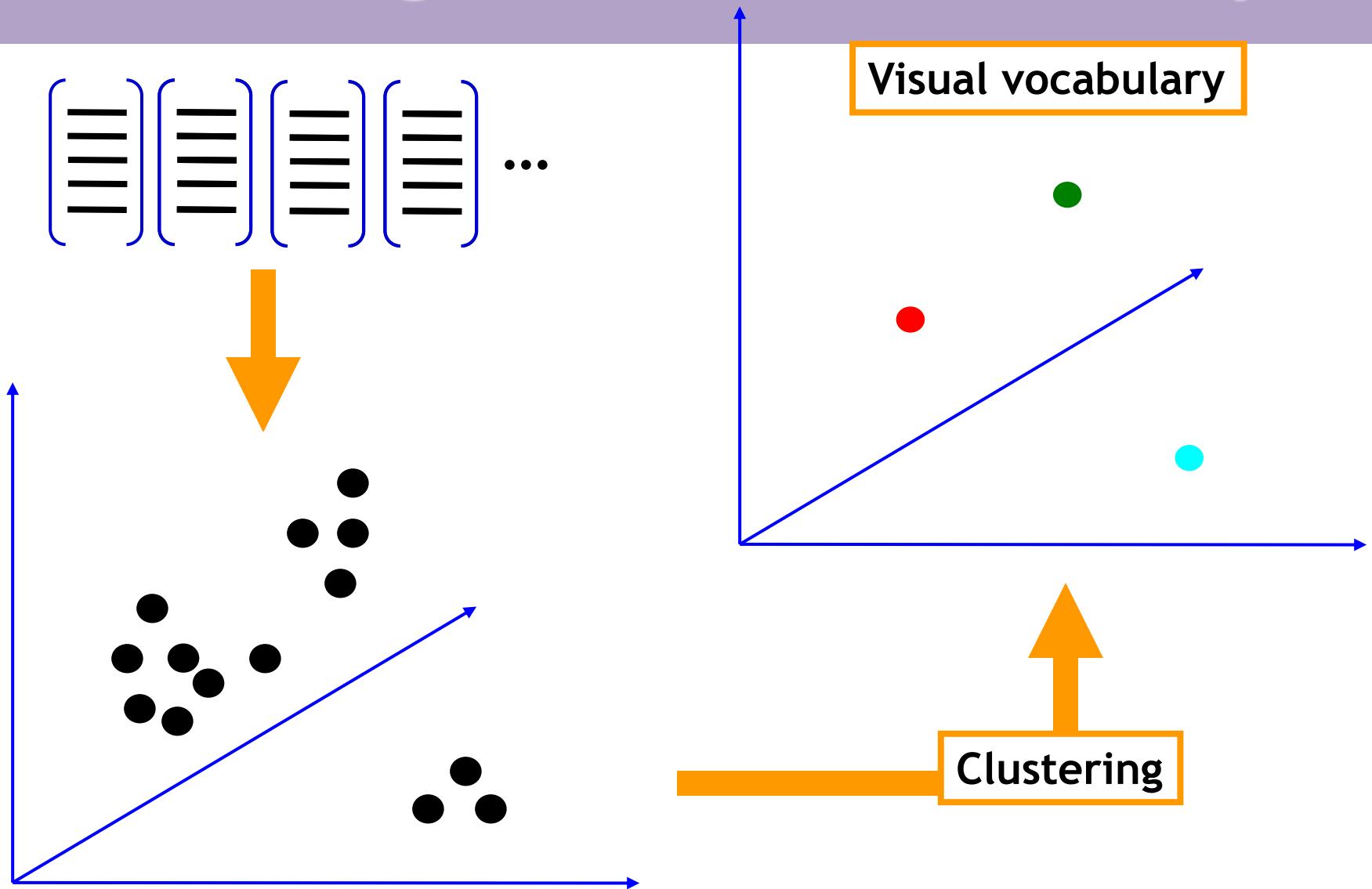
Set of descriptors



2. Learning the visual vocabulary

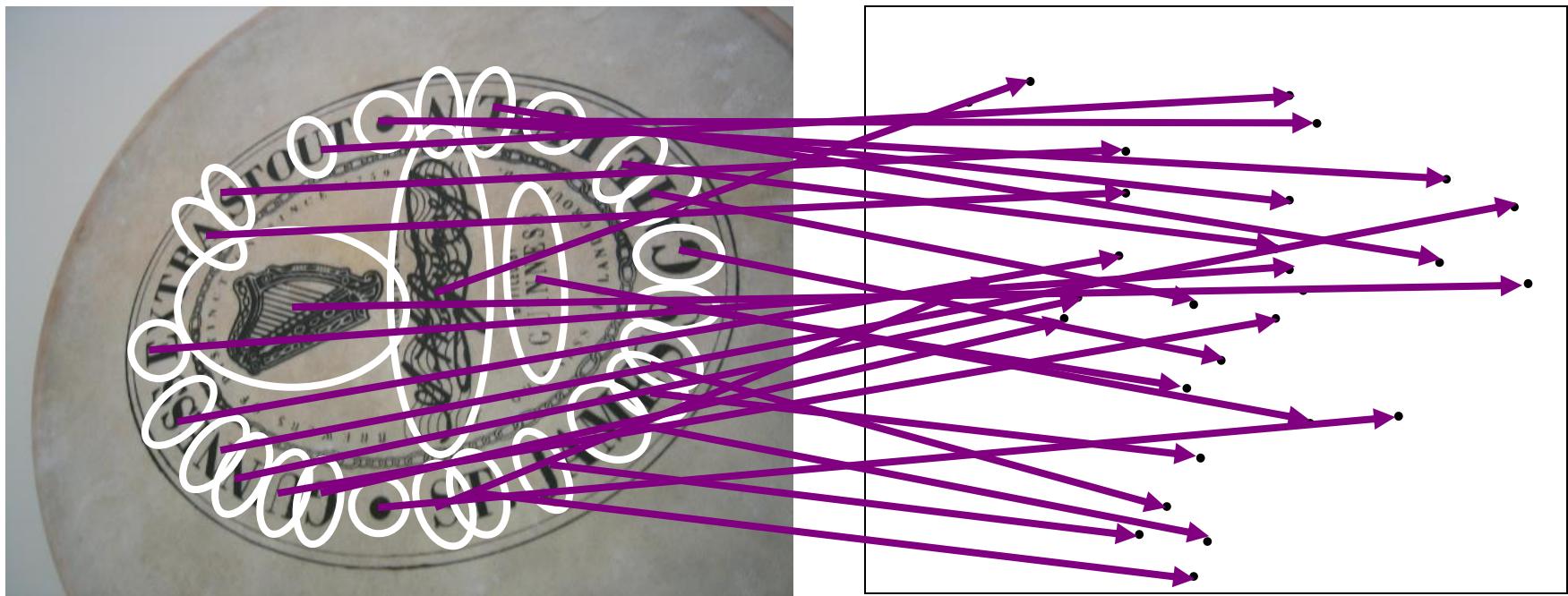


2. Learning the visual vocabulary



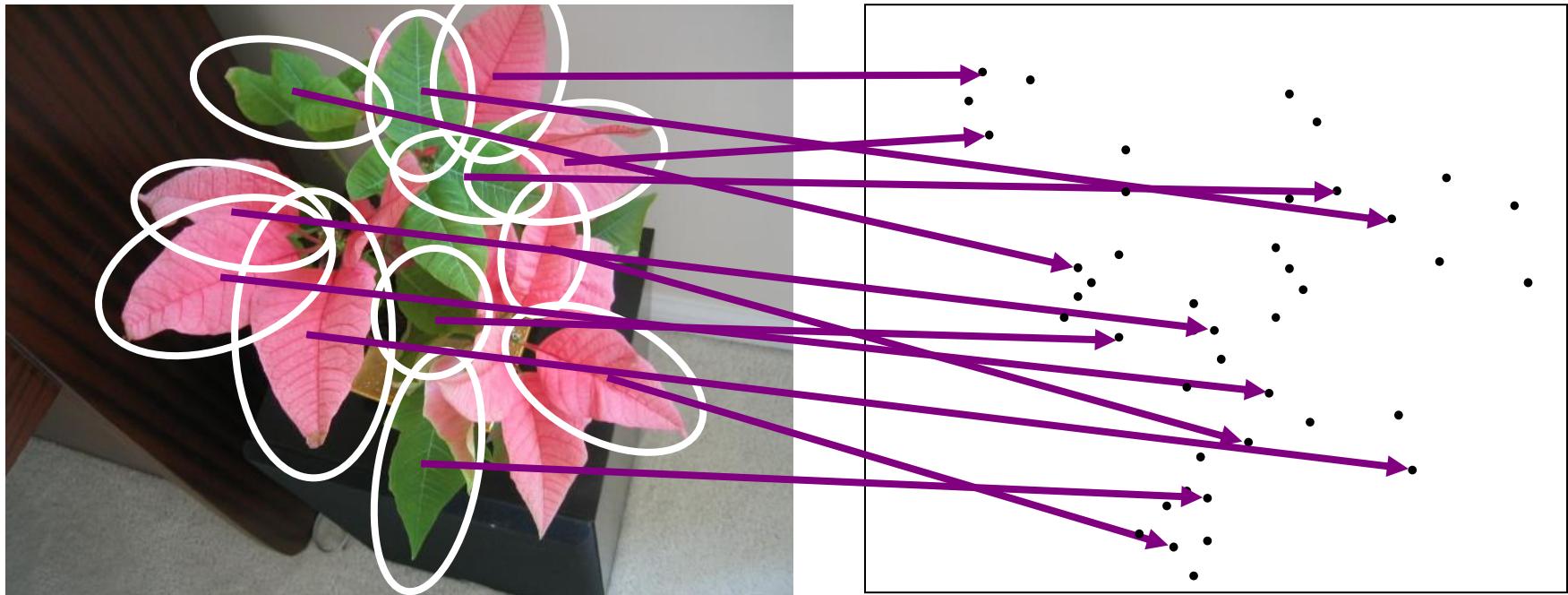
Visual words: main idea

- Extract some local features from several images ...

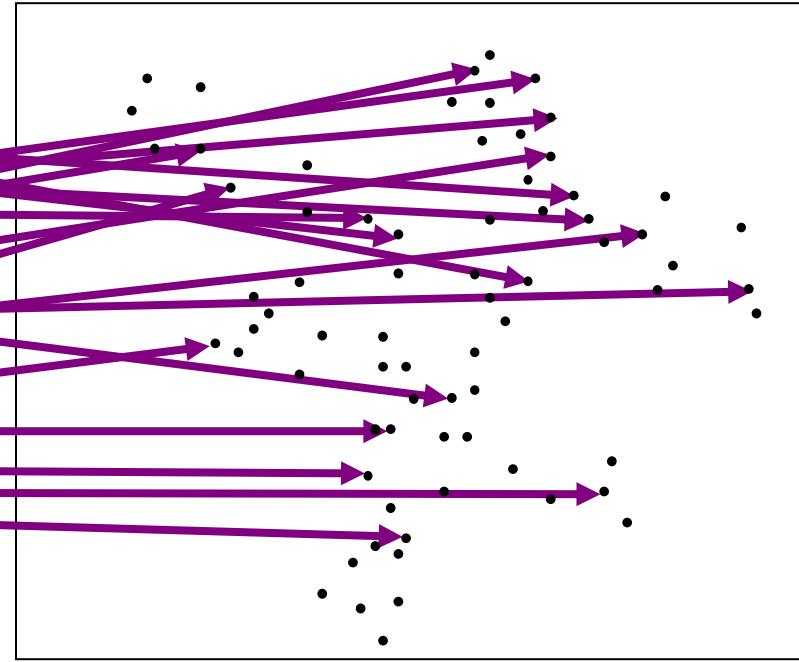
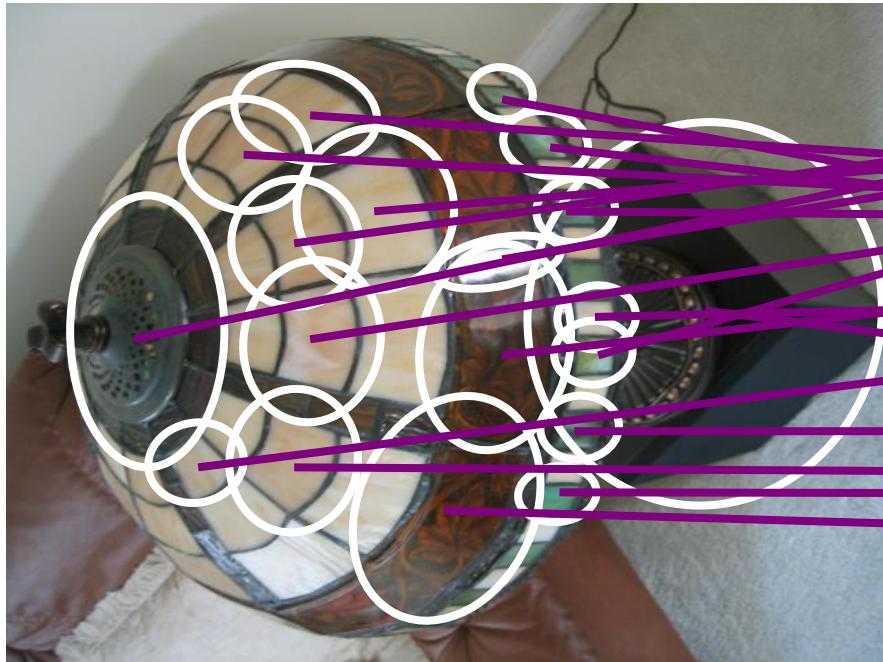


e.g., SIFT descriptor space: each point is 128-dimensional

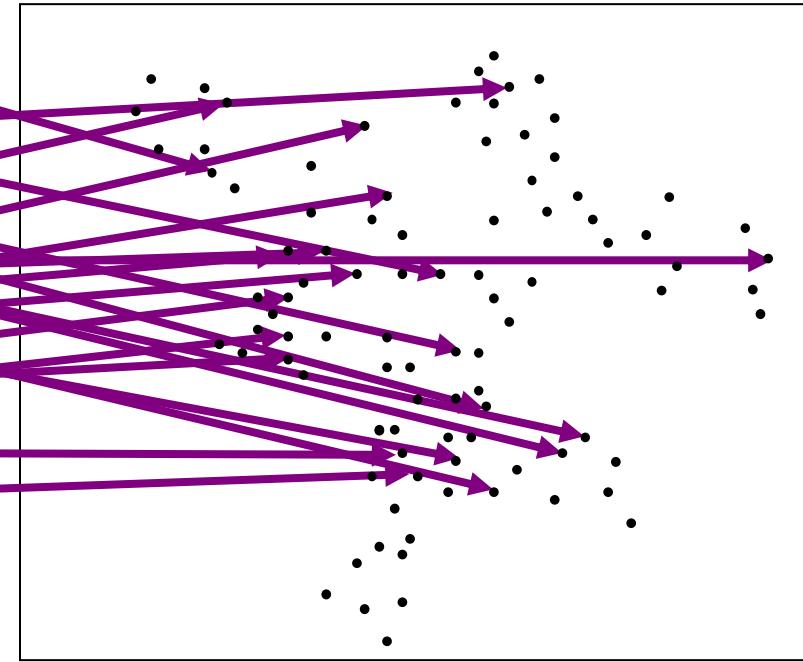
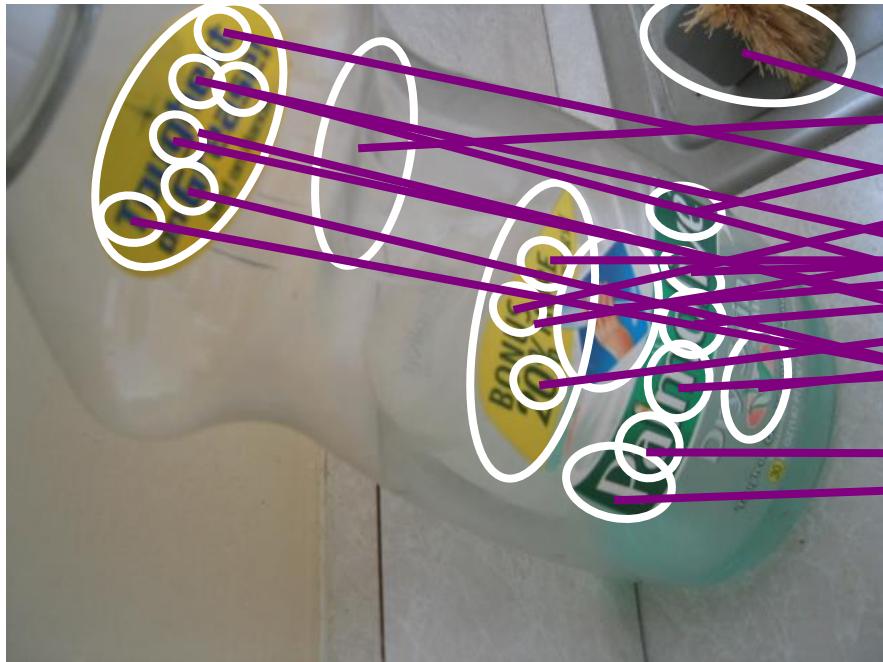
Visual words: main idea



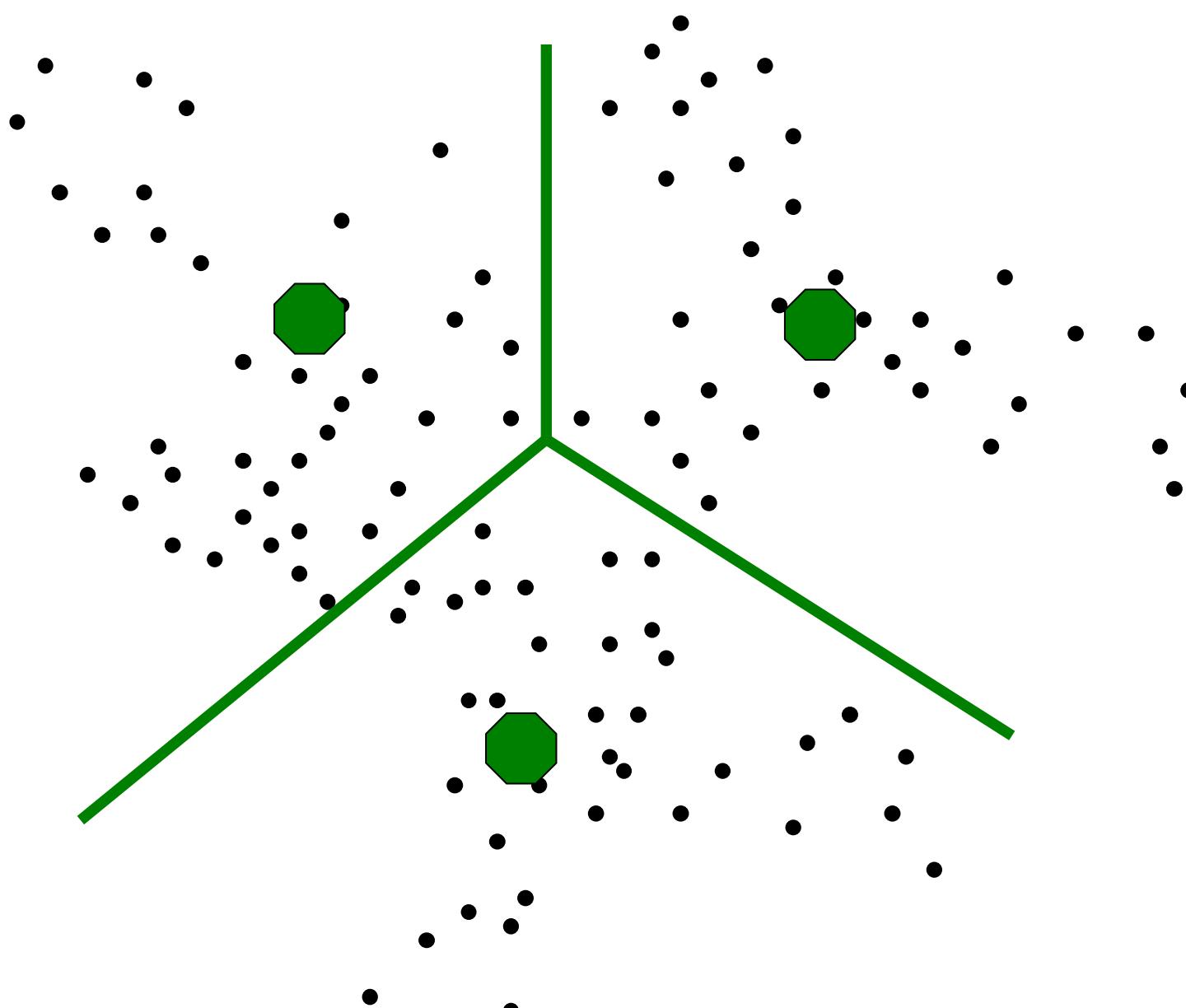
Visual words: main idea



Visual words: main idea







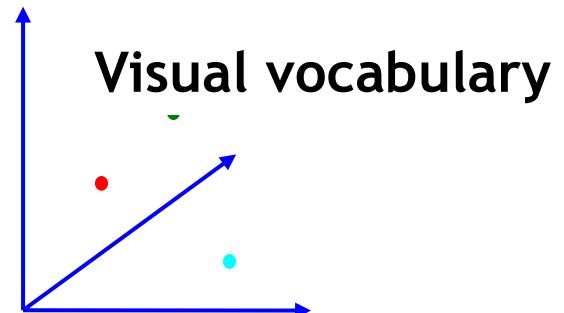
Clustering and vector quantization

Visual Codebook construction

- Clustering is a common method for learning a **visual vocabulary** or **codebook**
 - Unsupervised learning process
 - Each **cluster center** produced by k-means becomes a **codevector**
 - **Codebook** can be learned on separate **training set**
 - Provided the training set is sufficiently representative, the codebook will be “universal”

Vector Quantization

- The codebook is used for quantizing features
 - A *vector quantizer* takes a feature vector and maps it to the index of the nearest codevector in a codebook
 - Codebook = visual vocabulary
 - Codevector = visual word



Clustering / quantization methods

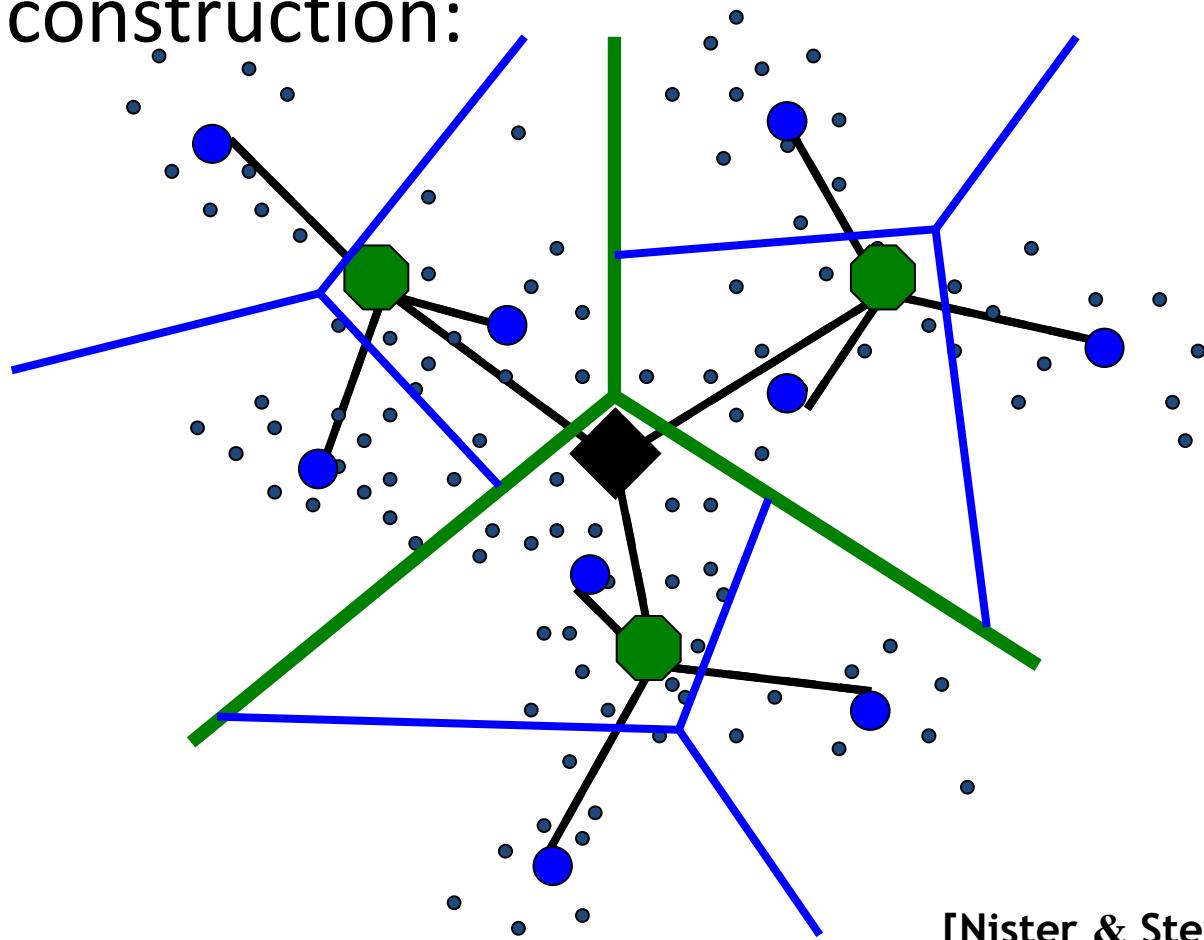
- k-means (typical choice), agglomerative clustering, mean-shift,...
- Hierarchical clustering: allows faster insertion / word assignment while still allowing large vocabularies
 - Vocabulary tree [Nister & Stewenius, CVPR 2006]

Visual vocabularies: Issues

- How to choose vocabulary size?
 - Too small: visual words not representative of all patches
 - Too large: quantization artifacts, overfitting
- Computational efficiency

Example: Recognition with Vocabulary Tree

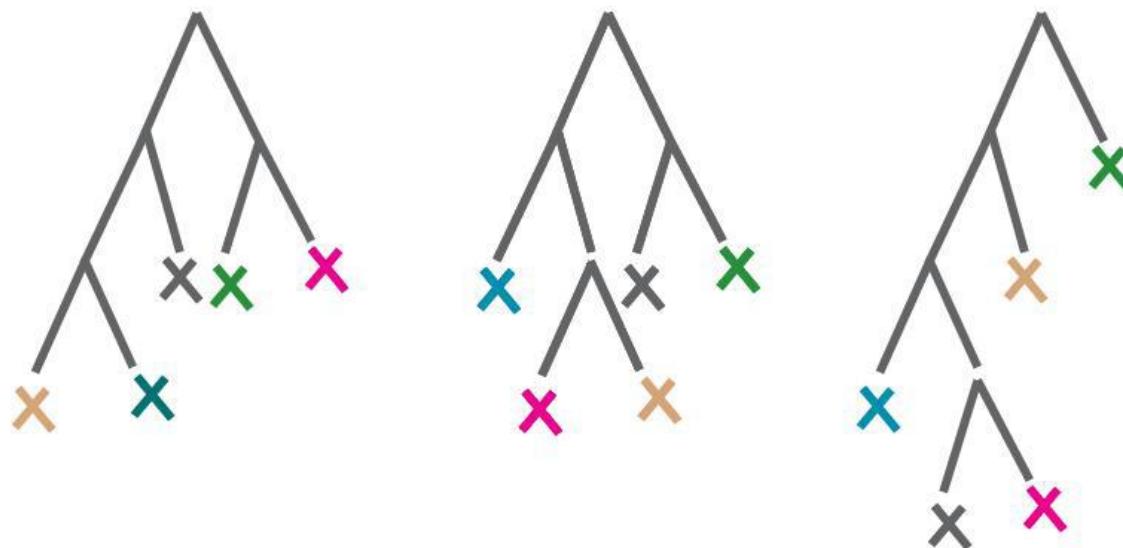
- Tree construction:



[Nister & Stewenius, CVPR'06]

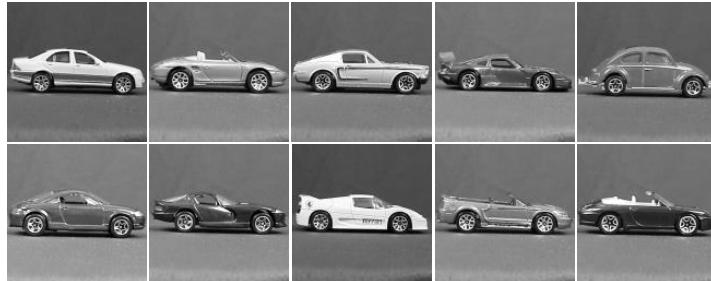
Vocabulary formation

- Ensembles of trees provide additional robustness



Moosmann, Jurie, & Triggs 2006; Yeh, Lee, & Darrell 2007; Bosch, Zisserman, & Munoz 2007; ...

Example codebook



Appearance codebook



Codevector

Clustering and vector quantization

Visual Codebook construction

- Clustering is a common method for learning a visual vocabulary or codebook
 - Unsupervised learning process
 - Each cluster center produced by k-means becomes a codevector
 - Codebook can be learned on separate training set
 - Provided the training set is sufficiently representative, the codebook will be “universal”

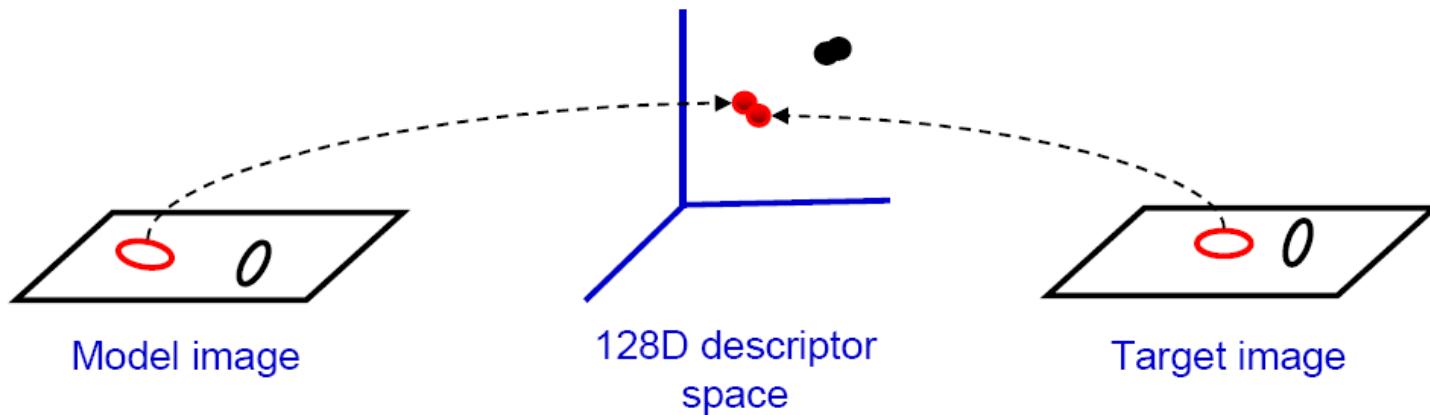
Vector Quantization

- The codebook is used for quantizing features
 - A *vector quantizer* takes a **feature vector** and maps it to the **index of the nearest codevector** in a codebook
 - Codebook = visual vocabulary
 - Codevector = visual word



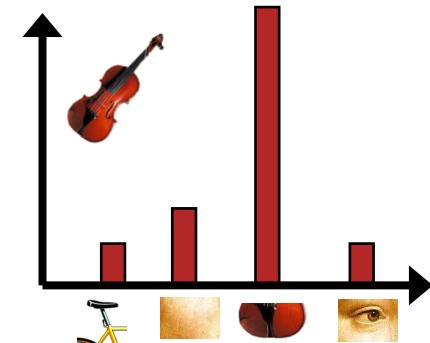
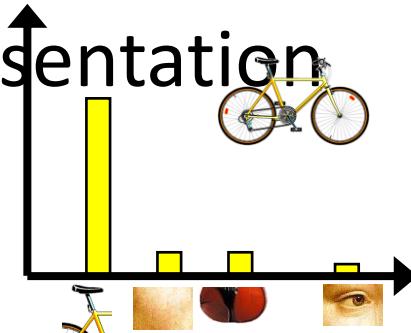
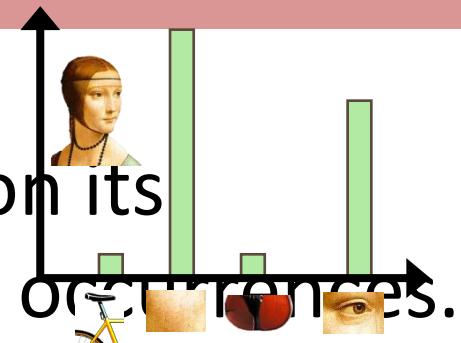
Indexing local features

- When we see close points in feature space, we have similar descriptors, which indicates similar local content.



Bags of visual words

- Summarize entire image based on its distribution (histogram) of word occurrences.
- Analogous to bag of words representation commonly used for documents.

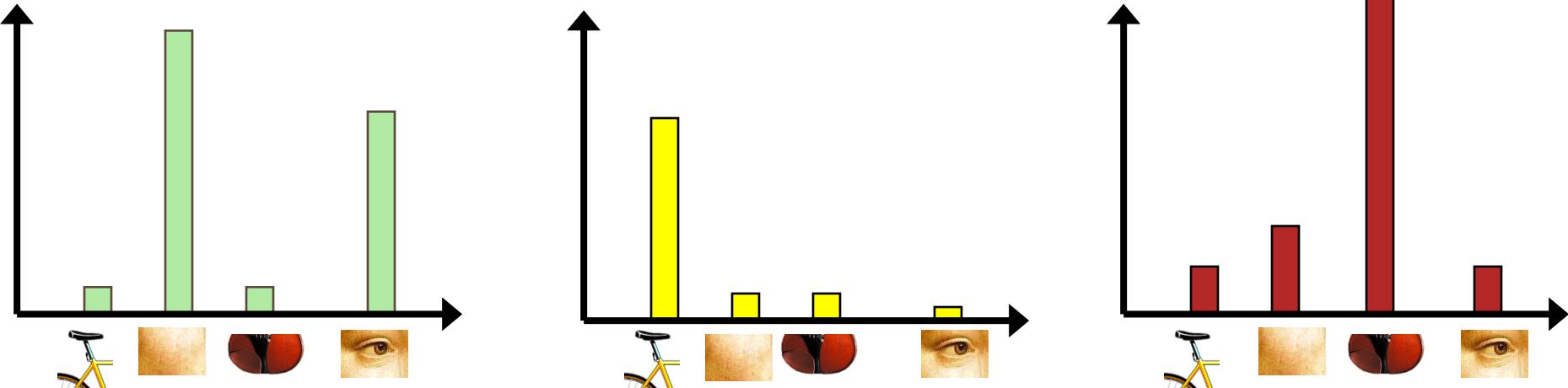


K. Grauman, B. Leibe

Image credit: Fei-Fei Li

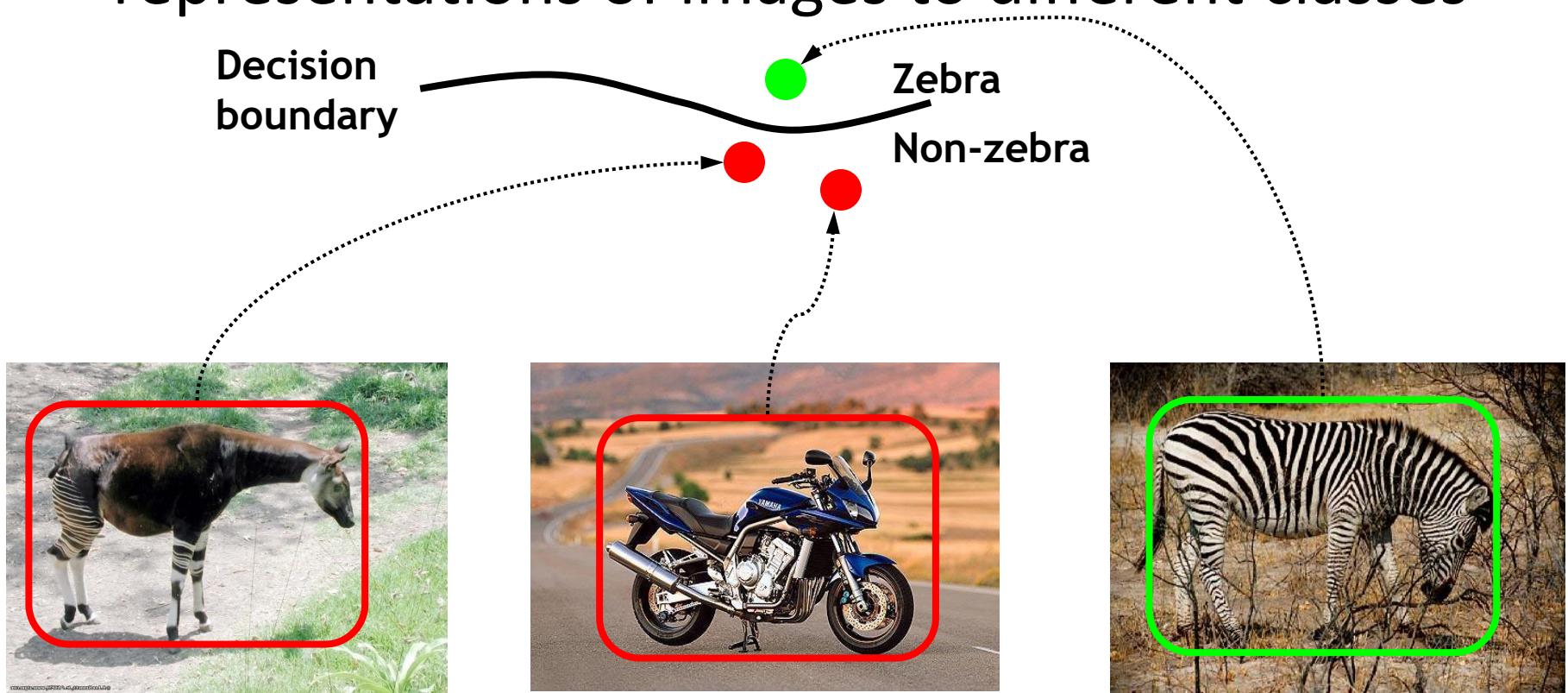
Image classification

- Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?



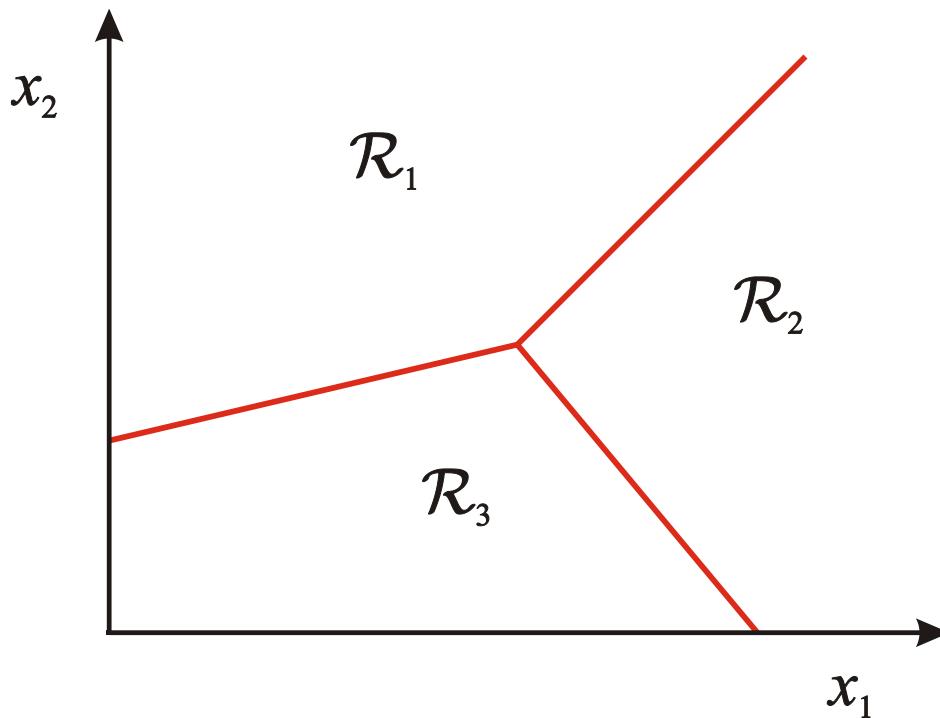
Classifiers

- Learn a decision rule assigning bag-of-features representations of images to different classes



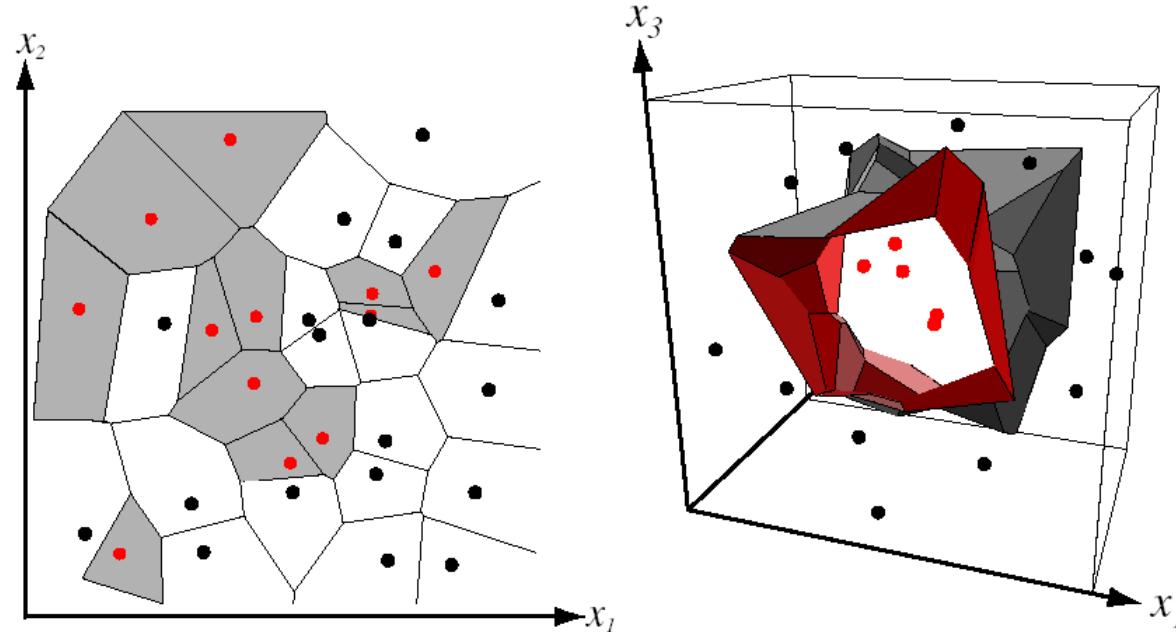
Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



Nearest Neighbor Classifier

- Assign label of nearest training data point to each test data point



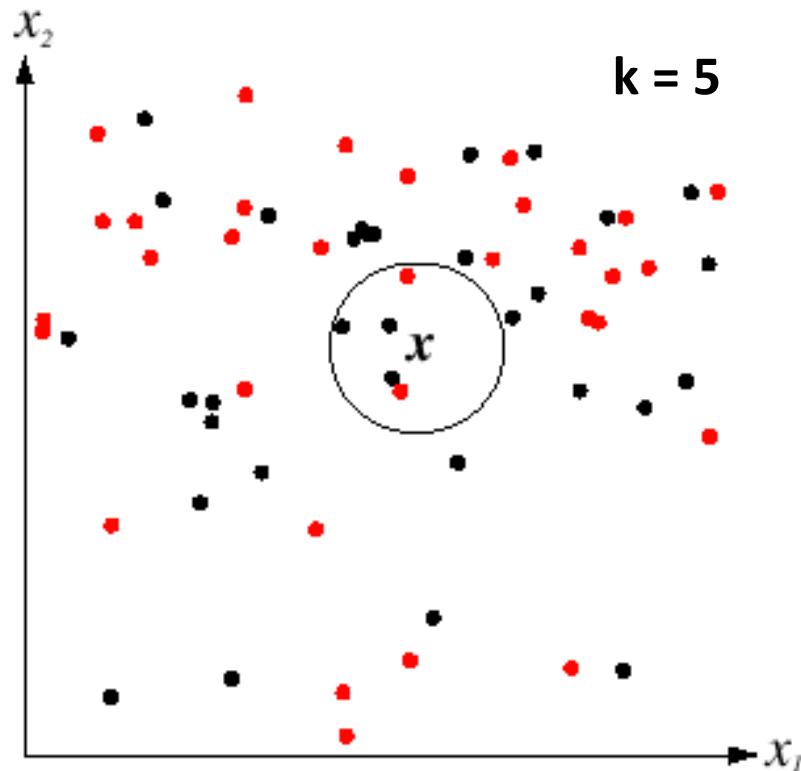
from Duda *et al.*

**Voronoi partitioning of feature space
for two-category 2D and 3D data**

Source: D. Lowe

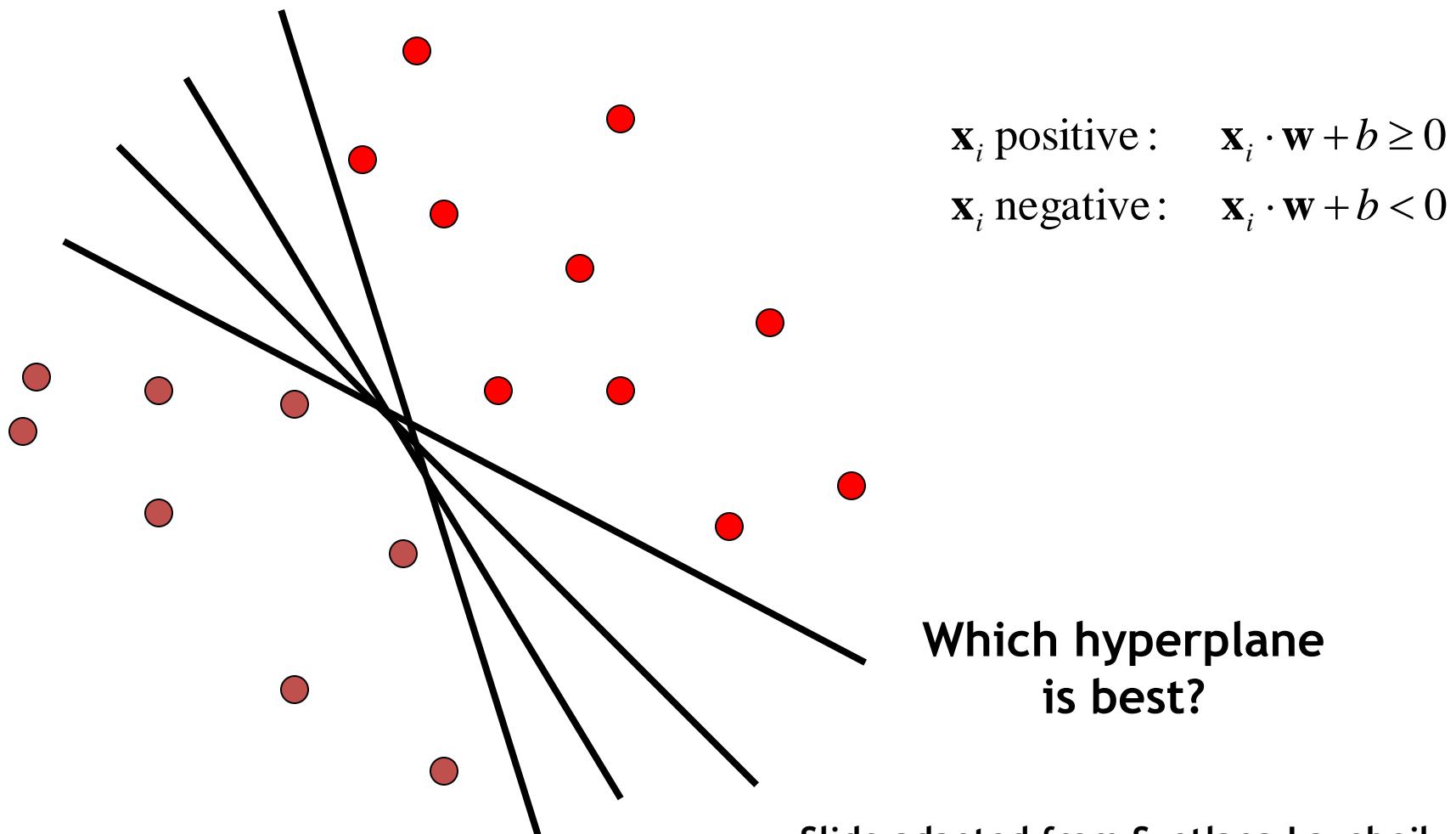
K-Nearest Neighbors

- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify
- Works well provided there is lots of data and the distance function is good



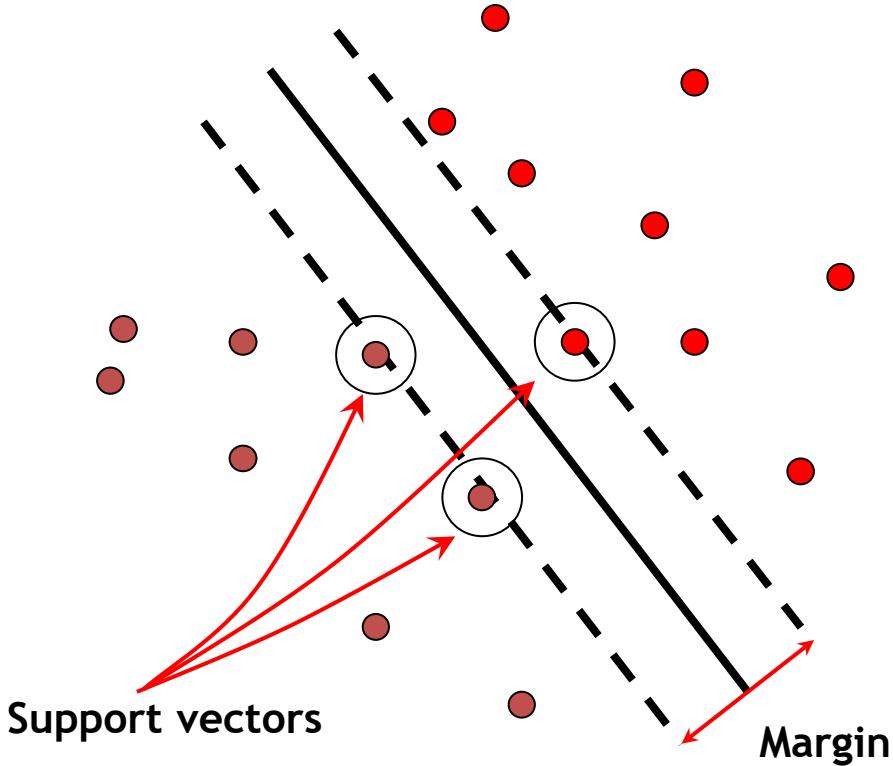
Linear classifiers

- Find linear function (*hyperplane*) to separate positive and negative examples



Support vector machines

- Find hyperplane that maximizes the *margin* between the positive and negative examples



$$\mathbf{x}_i \text{ positive } (y_i = 1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1): \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

For support vectors,

$$\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$$

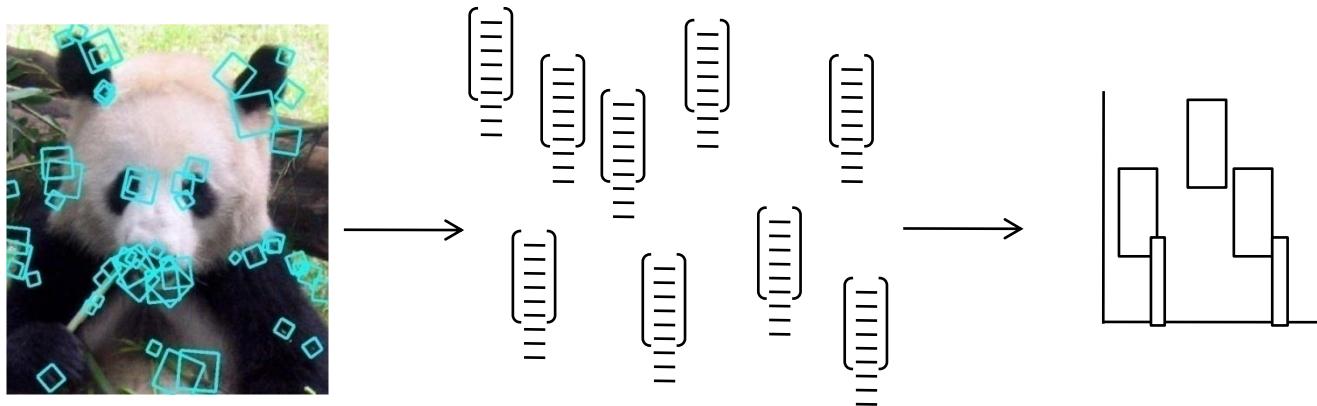
Distance between point and
hyperplane:

$$\frac{|\mathbf{x}_i \cdot \mathbf{w} + b|}{\|\mathbf{w}\|}$$

Therefore, the margin is $2 / \|\mathbf{w}\|$

Learning and recognition with bag of words histograms

- Bag of words representation makes it possible to describe the unordered point set with a single vector (of fixed dimension across image examples)



- Provides easy way to use distribution of feature types with various learning algorithms requiring vector input.

Bags of words: pros and cons

- + flexible to geometry / deformations / viewpoint
- + compact summary of image content
- + provides vector representation for sets
- + has yielded good recognition results in practice

- basic model ignores geometry – must verify afterwards, or encode via features
- background and foreground mixed when bag covers whole image
- interest points or sampling: no guarantee to capture object-level parts
- optimal vocabulary formation remains unclear

Image Alignment

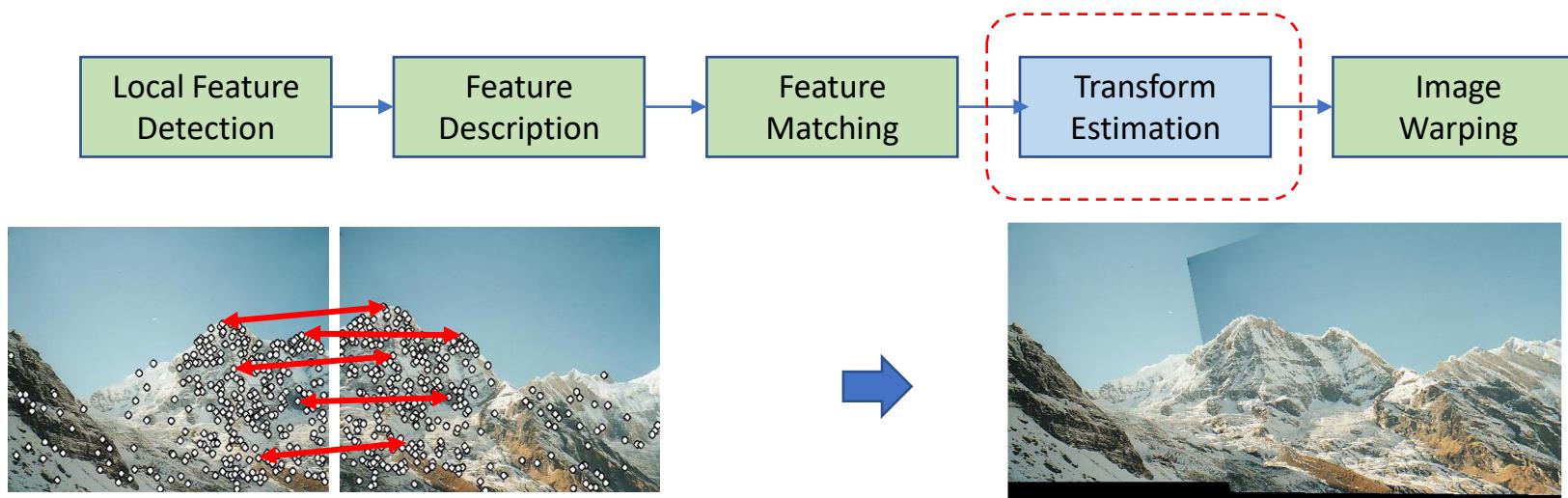
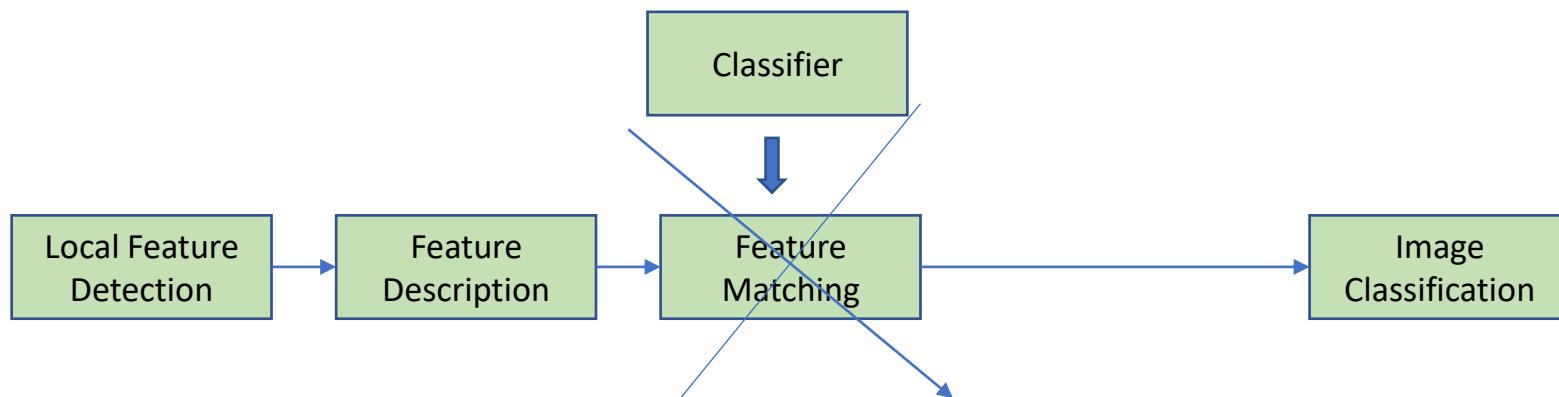


Image Classification



Traditional Machine Learning

I'll be borrowing slides from



UNIVERSITY OF
MICHIGAN

EECS 498-007 / 598-005
Deep Learning for Computer Vision
Fall 2019

Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification and object detection. Recent developments in neural network approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of neural-network based deep learning methods for computer vision. During this course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. We will cover learning algorithms, neural network architectures, and practical engineering tricks for training and fine-tuning networks for visual recognition tasks.

Instructor



Justin Johnson

Graduate Student Instructors



Yunseok Jang



Kibok Lee



Luowei Zhou



Lecture 2: Image Classification

Image Classification

Input: image



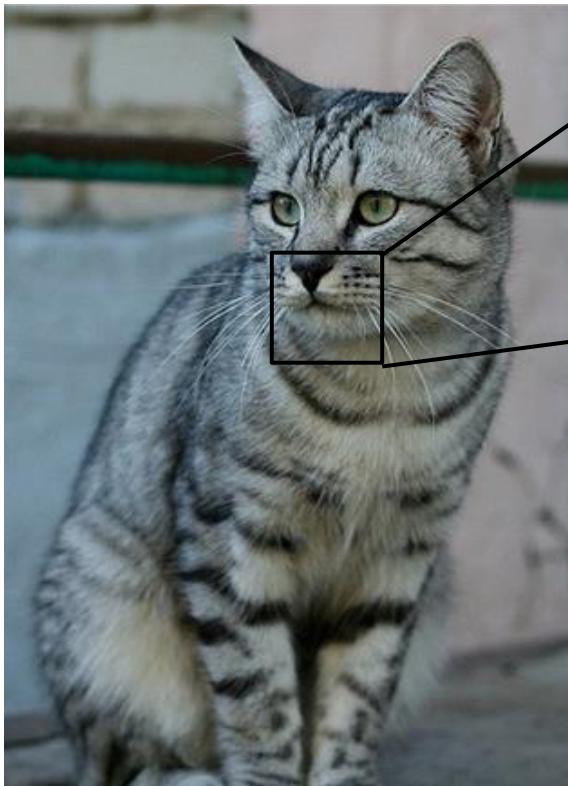
This image by Nikita is
licensed under CC-BY 2.0

Output: Assign image to one
of a fixed set of categories



cat
bird
deer
dog
truck

Problem: Semantic Gap



This image by [Nikita](#) is
licensed under [CC-BY 2.0](#)

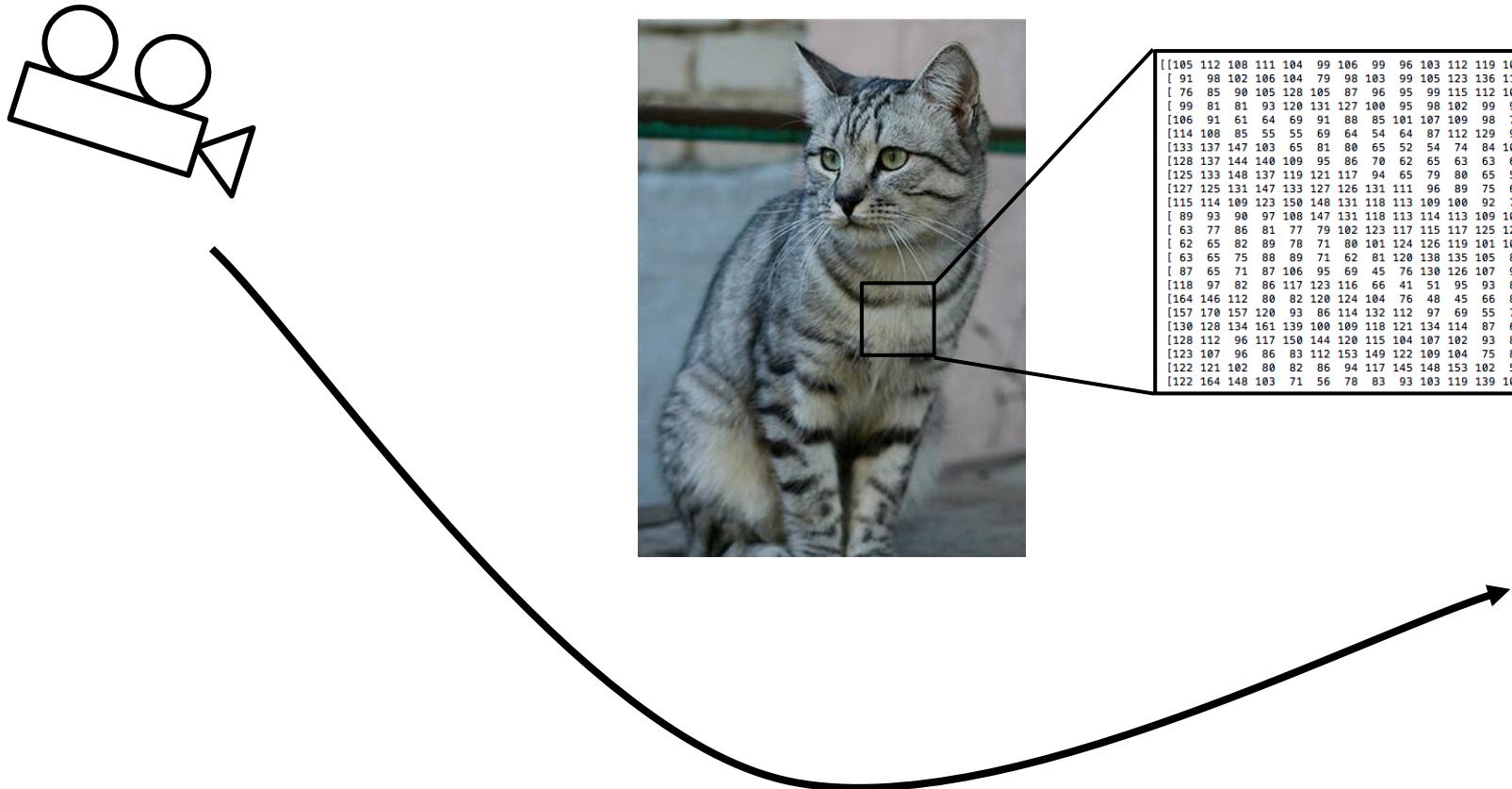
[105 112 108 111 104 99 106 99 96 103 112 119 104 97 93 87]
[91 98 102 106 104 79 98 103 99 105 123 136 110 105 94 85]
[76 85 90 105 128 105 87 96 95 99 115 112 106 103 99 85]
[99 81 81 93 120 131 127 100 95 98 102 99 96 93 101 94]
[106 91 61 64 69 91 88 85 101 107 109 98 75 84 96 95]
[114 108 85 55 55 69 64 54 64 87 112 129 98 74 84 91]
[133 137 147 103 65 81 80 65 52 54 74 84 102 93 85 82]
[128 137 144 140 109 95 86 70 62 65 63 63 60 73 86 101]
[125 133 148 137 119 121 117 94 65 79 80 65 54 64 72 98]
[127 125 131 147 133 127 126 131 111 96 89 75 61 64 72 84]
[115 114 109 123 150 148 131 118 113 109 100 92 74 65 72 78]
[89 93 90 97 108 147 131 118 113 114 113 109 106 95 77 80]
[63 77 86 81 77 79 102 123 117 115 117 125 125 130 115 87]
[62 65 82 89 78 71 80 101 124 126 119 101 107 114 131 119]
[63 65 75 88 89 71 62 81 120 138 135 105 81 98 110 118]
[87 65 71 87 106 95 69 45 76 130 126 107 92 94 105 112]
[118 97 82 86 117 123 116 66 41 51 95 93 89 95 102 107]
[164 146 112 80 82 120 124 104 76 48 45 66 88 101 102 109]
[157 170 157 120 93 86 114 132 112 97 69 55 70 82 99 94]
[130 128 134 161 139 100 109 118 121 134 114 87 65 53 69 86]
[128 112 96 117 150 144 120 115 104 107 102 93 87 81 72 79]
[123 107 96 86 83 112 153 149 122 109 104 75 80 107 112 99]
[122 121 102 80 82 86 94 117 145 148 153 102 58 78 92 107]
[122 164 148 103 71 56 78 83 93 103 119 139 102 61 69 84]

What the computer sees

An image is just a big grid of numbers between [0, 255]:

e.g. 800 x 600 x 3
(3 channels RGB)

Challenges: Viewpoint Variation



All pixels change when
the camera moves!

Challenges: Intraclass Variation



This image is CC0 1.0 public domain

Challenges: Fine-Grained Categories

Maine Coon



[This image is free for use under the Pixabay License](#)

Ragdoll



[This image is CCO public domain](#)

American Shorthair



[This image is CCO public domain](#)

Challenges: Background Clutter



[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

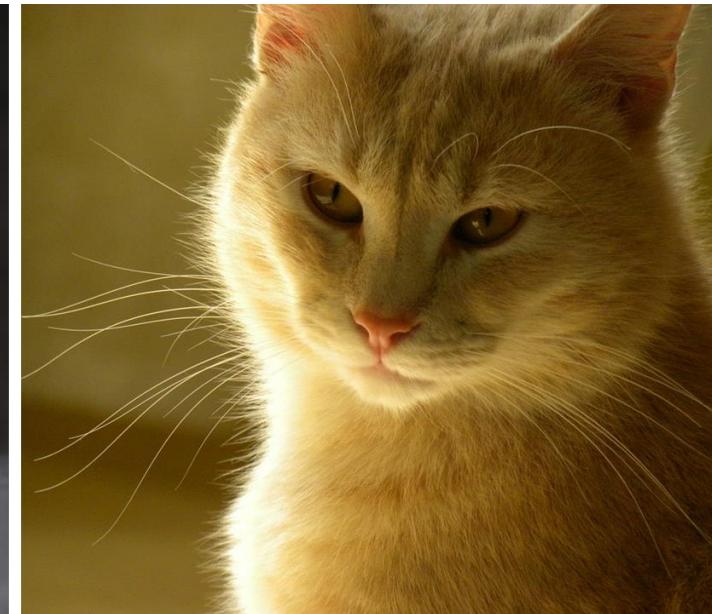
Challenges: Illumination Changes



[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

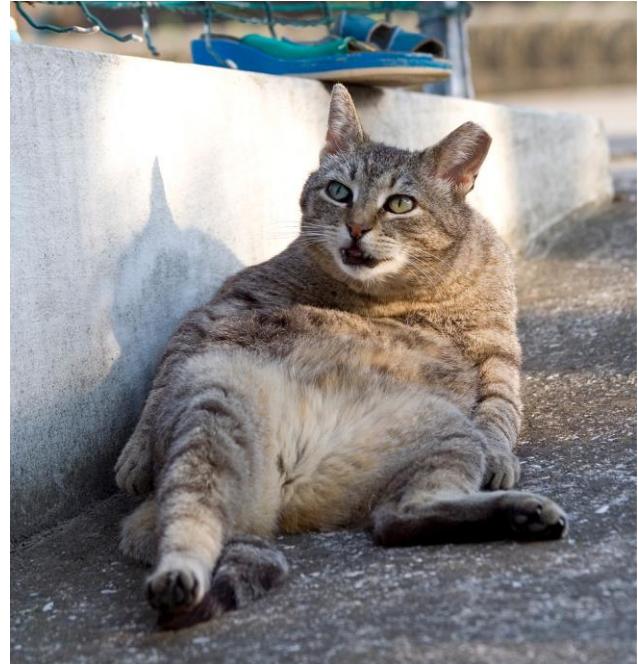


[This image is CC0 1.0 public domain](#)



[This image is CC0 1.0 public domain](#)

Challenges: Deformation



[This image by Umberto Salvagnin](#) is
licensed under CC-BY 2.0



[This image by Umberto Salvagnin](#) is
licensed under CC-BY 2.0



[This image by sare bear](#) is licensed
under CC-BY 2.0



[This image by Tom Thai](#) is licensed
under CC-BY 2.0

Challenges: Occlusion



[This image is CC0 1.0 public domain](#)



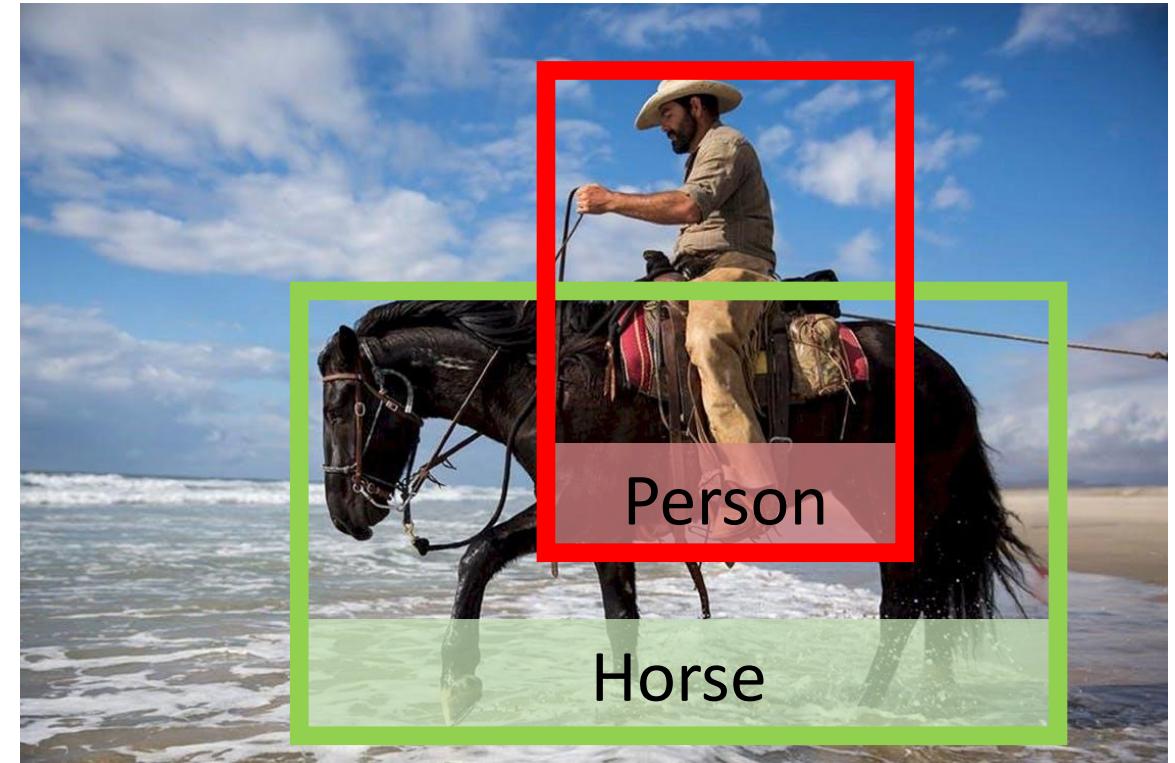
[This image is CC0 1.0 public domain](#)



[This image by jonsson is licensed under CC-BY 2.0](#)

Image Classification: Building Block for other tasks!

Example: Object Detection



[This image](#) is free to use under the [Pexels license](#)

Image Classification: Building Block for other tasks!

Example: Object Detection



Background
Horse
Person
Car
Truck

[This image](#) is free to use under the [Pexels license](#)

Image Classification: Building Block for other tasks!

Example: Object Detection



Background
Horse
Person
Car
Truck

[This image](#) is free to use under the [Pexels license](#)

Image Classification: Building Block for other tasks!

Example: Image Captioning



riding
cat
horse
man
when
...
<STOP>

What word
to say next?

Caption:
Man

[This image](#) is free to use under the [Pexels license](#)

Image Classification: Building Block for other tasks!

Example: Image Captioning



riding

cat

horse

man

when

...

<STOP>

What word
to say next?

Caption:
Man riding

[This image](#) is free to use under the [Pexels license](#)

Image Classification: Building Block for other tasks!

Example: Image Captioning



riding

cat

horse

man

when

...

<STOP>

What word
to say next?

Caption:
Man riding horse

[This image](#) is free to use under the [Pexels license](#)

Image Classification: Building Block for other tasks!

Example: Image Captioning



riding
cat
horse
man
when
...
<STOP>

What word
to say next?

Caption:
Man riding horse

[This image](#) is free to use under the [Pexels license](#)

An Image Classifier

```
def classify_image(image):  
    # Some magic here?  
    return class_label
```

Unlike e.g. sorting a list of numbers,

no obvious way to hard-code the algorithm
for recognizing a cat, or other classes.

You could try ...



Find edges



Find corners



?

Machine Learning: Data-Driven Approach

1. Collect a dataset of images and labels
2. Use Machine Learning to train a classifier
3. Evaluate the classifier on new images

Example training set

```
def train(images, labels):  
    # Machine learning!  
    return model
```

```
def predict(model, test_images):  
    # Use model to predict labels  
    return test_labels
```

airplane



automobile



bird



cat

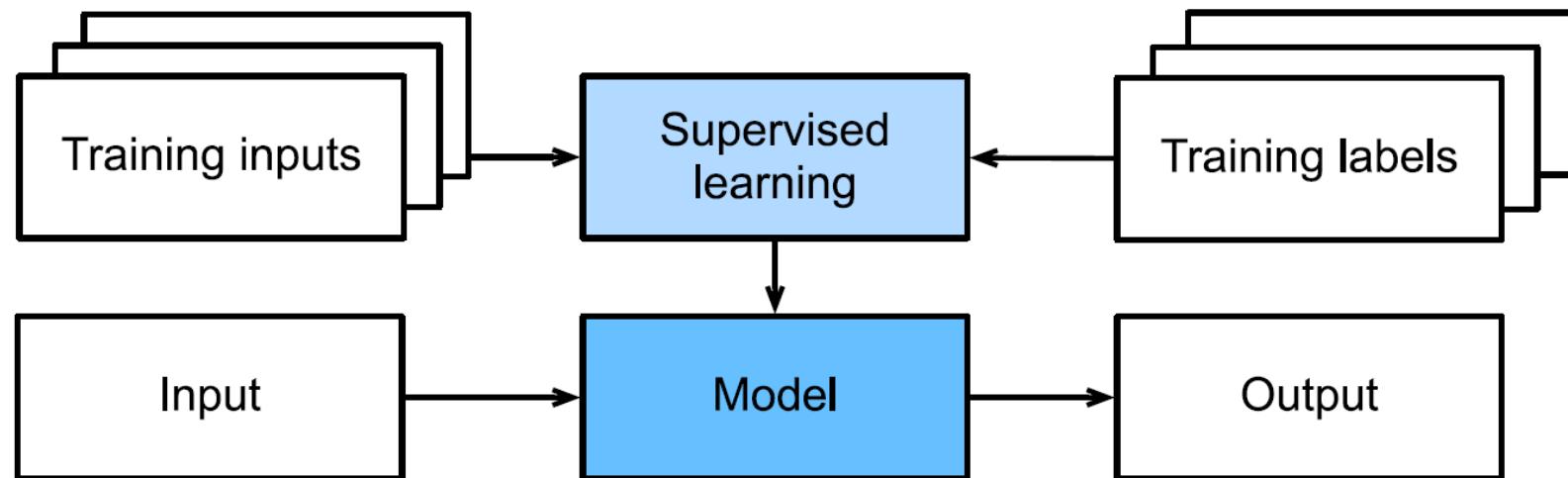


deer

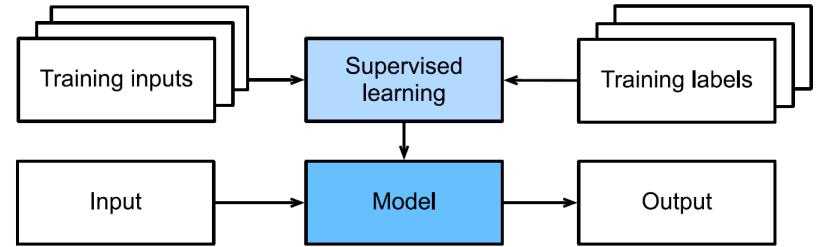


Supervised learning

- Goal: provide best output predictions for novel inputs



Supervised learning



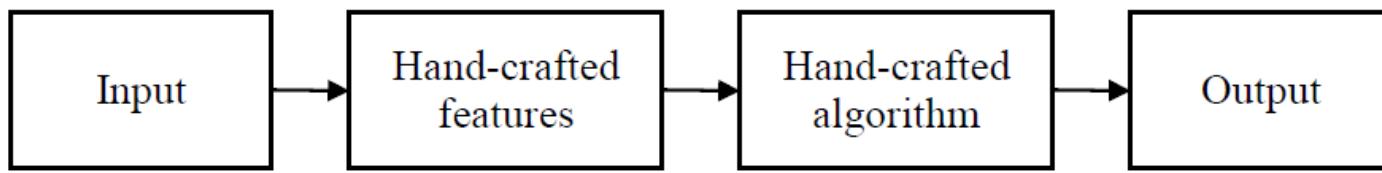
- Goal: provide best output predictions for novel inputs
- How can we predict future performance?
- Placeholder answer: minimize *empirical risk*

$$E_{\text{Risk}}(\mathbf{w}) = \frac{1}{N} \sum L(\mathbf{y}_i, \mathbf{f}(\mathbf{x}_i; \mathbf{w})). \quad (5.1)$$

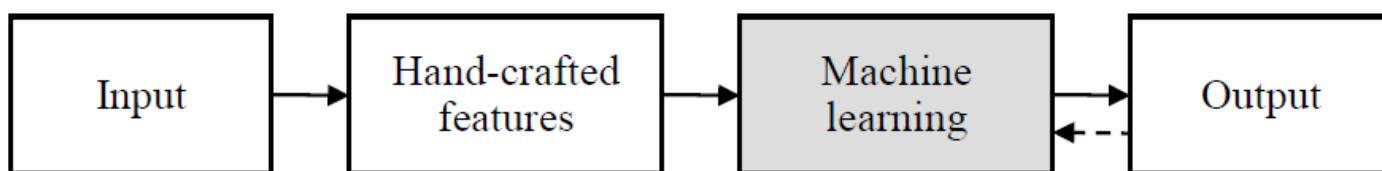
The loss function L measures the “cost” of predicting an output $\mathbf{f}(\mathbf{x}_i; \mathbf{w})$ for input \mathbf{x}_i and model parameters \mathbf{w} when the corresponding target is \mathbf{y}_i .

- Where have we seen this before?
- What are potential *models* and *loss functions*?

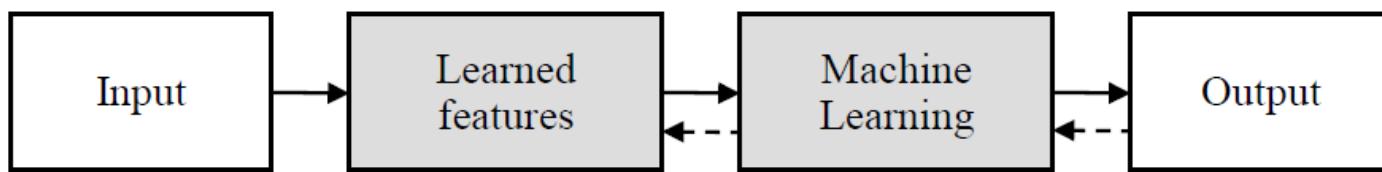
Traditional and deep learning



(a) Traditional vision pipeline

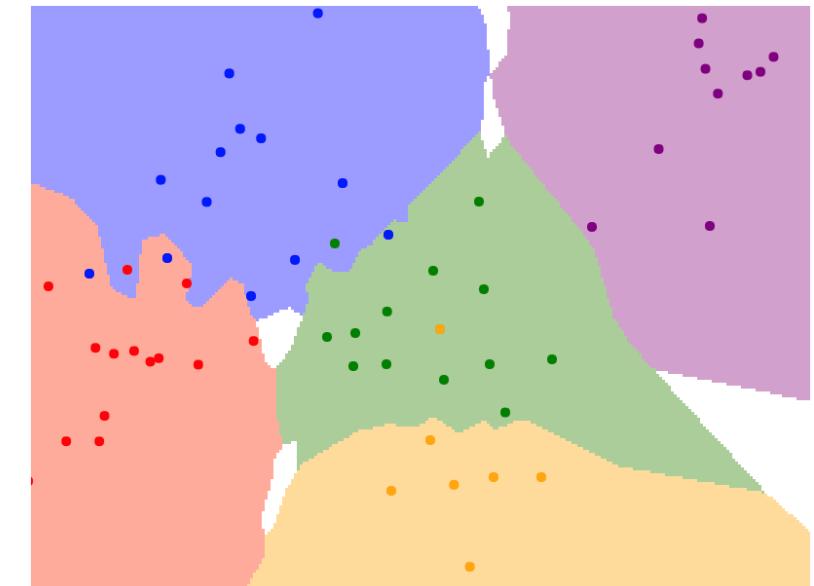


(b) Classic machine learning pipeline



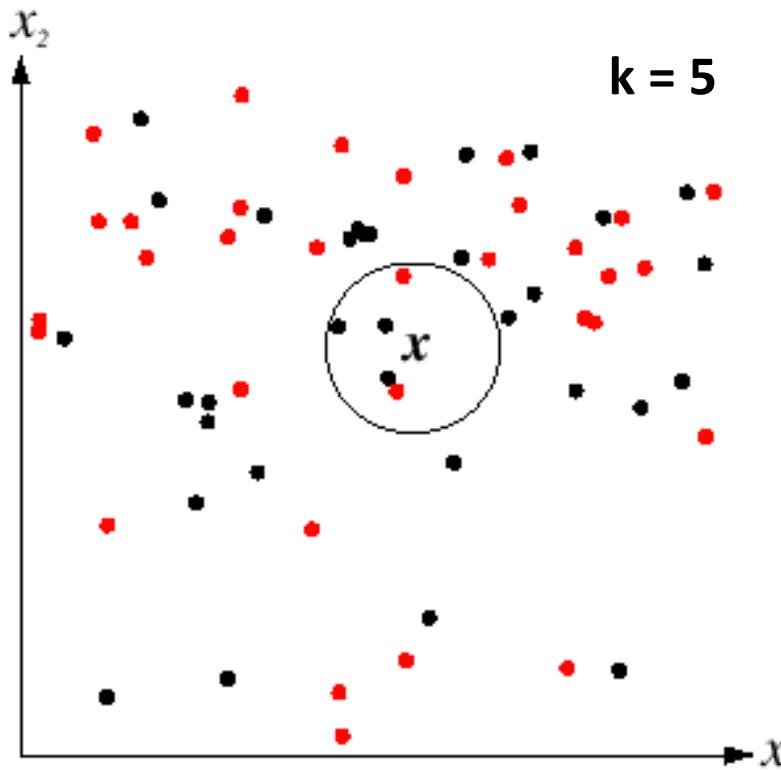
(c) Deep learning pipeline

Nearest neighbors



K-Nearest Neighbors

- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify
- Works well provided there is lots of data and the distance function is good



Source: D. Lowe

First classifier: Nearest Neighbor

```
def train(images, labels):  
    # Machine learning!  
    return model
```

```
def predict(model, test_images):  
    # Use model to predict labels  
    return test_labels
```

→ Memorize all data and labels

→ Predict the label of the most similar training image

Distance Metric to compare images

L1 distance:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

test image

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

training image

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

-

pixel-wise absolute value differences

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

add
→ 456

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

Nearest Neighbor Classifier

```

import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

```

Nearest Neighbor Classifier

Memorize training data

```

import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

```

Nearest Neighbor Classifier

For each test image:
 Find nearest training image
 Return label of nearest image

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

Nearest Neighbor Classifier

Q: With N examples,
how fast is training?

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

Nearest Neighbor Classifier

Q: With N examples,
how fast is training?

A: $O(1)$

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

Nearest Neighbor Classifier

Q: With N examples,
how fast is training?

A: O(1)

Q: With N examples,
how fast is testing?

```
import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred
```

Nearest Neighbor Classifier

Q: With N examples,
how fast is training?

A: O(1)

Q: With N examples,
how fast is testing?

A: O(N)

```

import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

```

Nearest Neighbor Classifier

Q: With N examples,
how fast is training?

A: O(1)

Q: With N examples,
how fast is testing?

A: O(N)

This is **bad**: We can afford slow training, but we need fast testing!

```

import numpy as np

class NearestNeighbor:
    def __init__(self):
        pass

    def train(self, X, y):
        """ X is N x D where each row is an example. Y is 1-dimension of size N """
        # the nearest neighbor classifier simply remembers all the training data
        self.Xtr = X
        self.ytr = y

    def predict(self, X):
        """ X is N x D where each row is an example we wish to predict label for """
        num_test = X.shape[0]
        # lets make sure that the output type matches the input type
        Ypred = np.zeros(num_test, dtype = self.ytr.dtype)

        # loop over all test rows
        for i in xrange(num_test):
            # find the nearest training image to the i'th test image
            # using the L1 distance (sum of absolute value differences)
            distances = np.sum(np.abs(self.Xtr - X[i,:]), axis = 1)
            min_index = np.argmin(distances) # get the index with smallest distance
            Ypred[i] = self.ytr[min_index] # predict the label of the nearest example

        return Ypred

```

Nearest Neighbor Classifier

There are many methods for fast / approximate nearest neighbors; e.g. see

<https://github.com/facebookresearch/faiss>

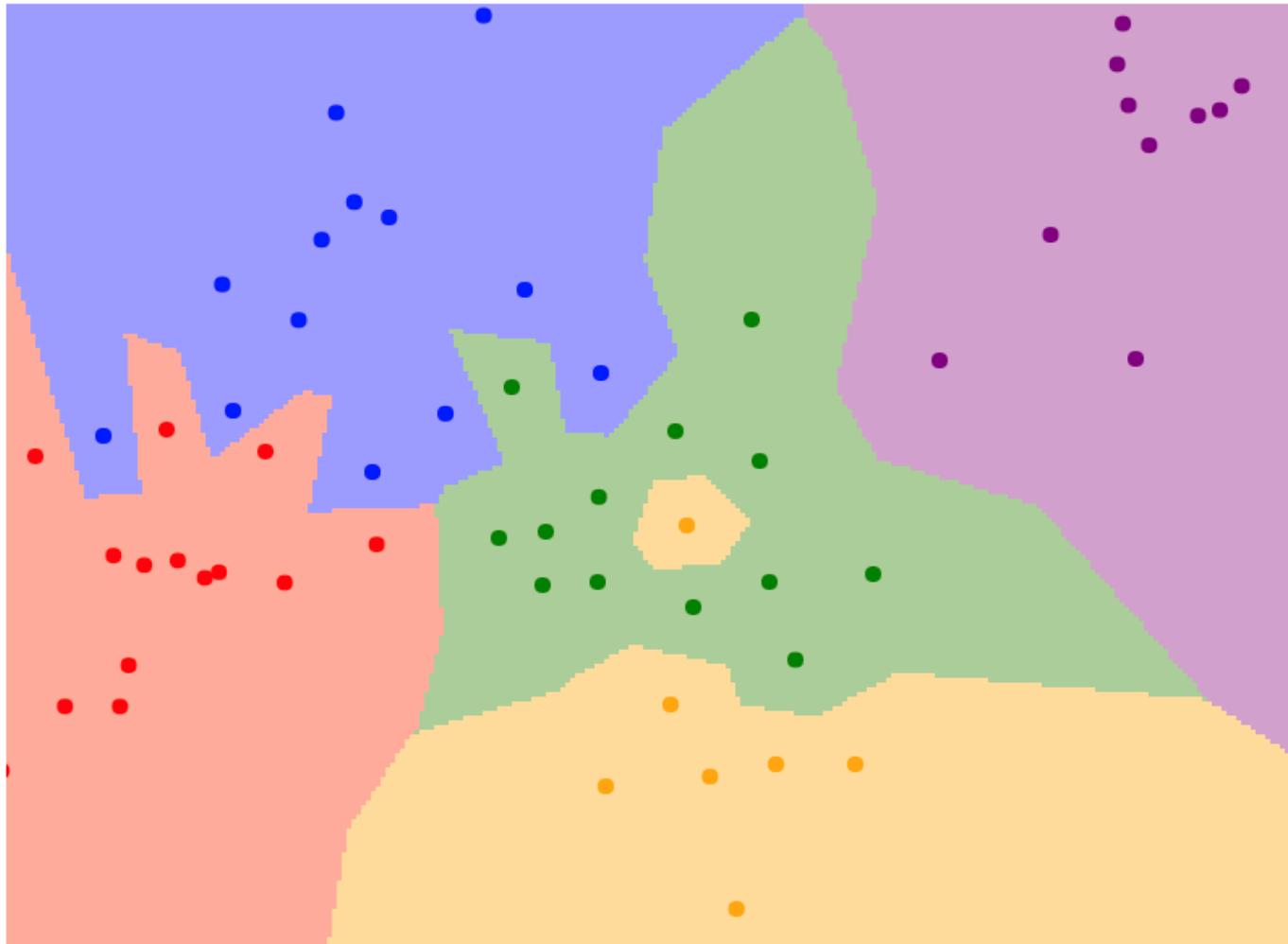
What does this look like?



What does this look like?

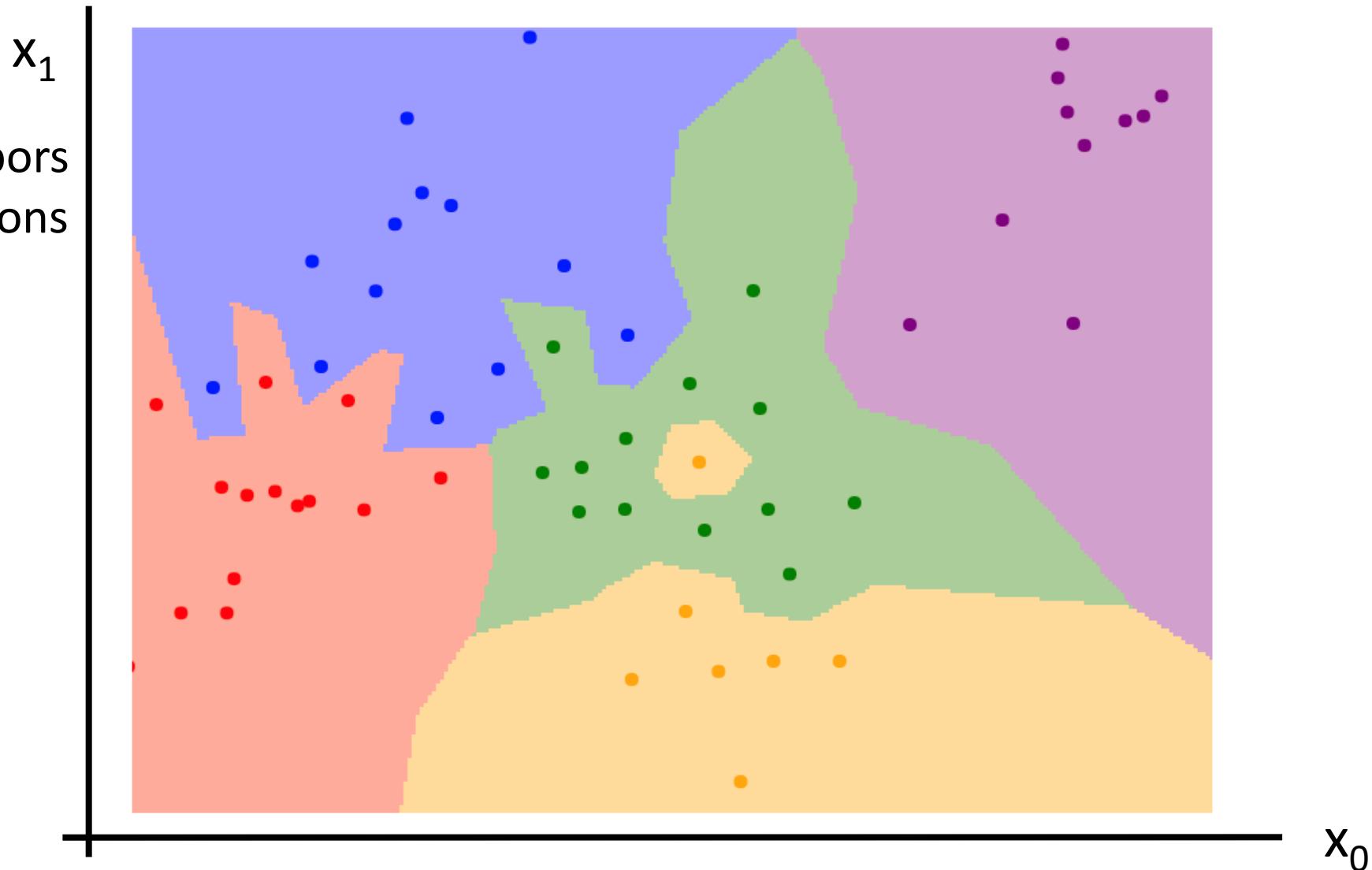


Nearest Neighbor Decision Boundaries



Nearest Neighbor Decision Boundaries

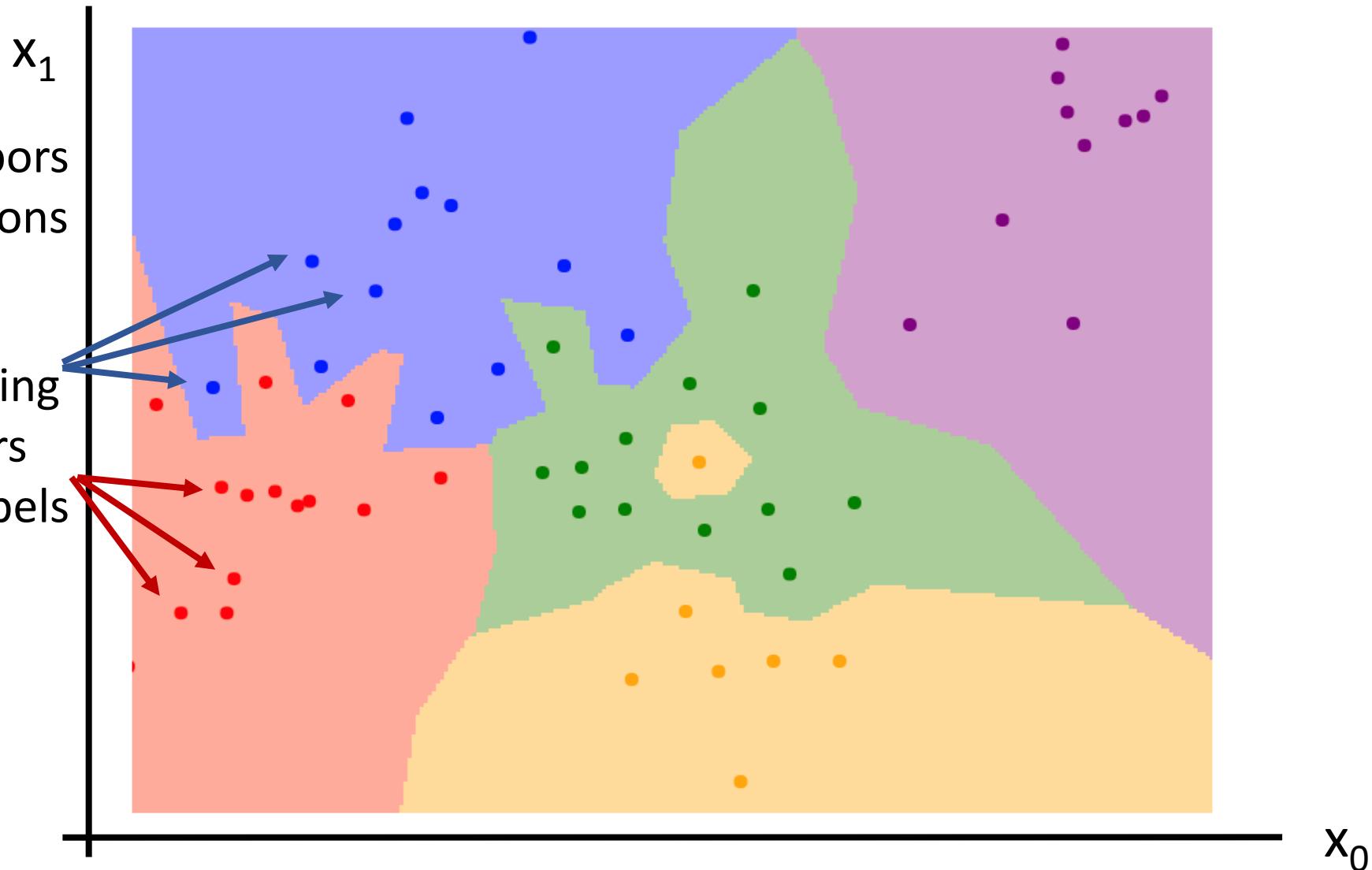
Nearest neighbors
in two dimensions



Nearest Neighbor Decision Boundaries

Nearest neighbors
in two dimensions

Points are training
examples; colors
give training labels

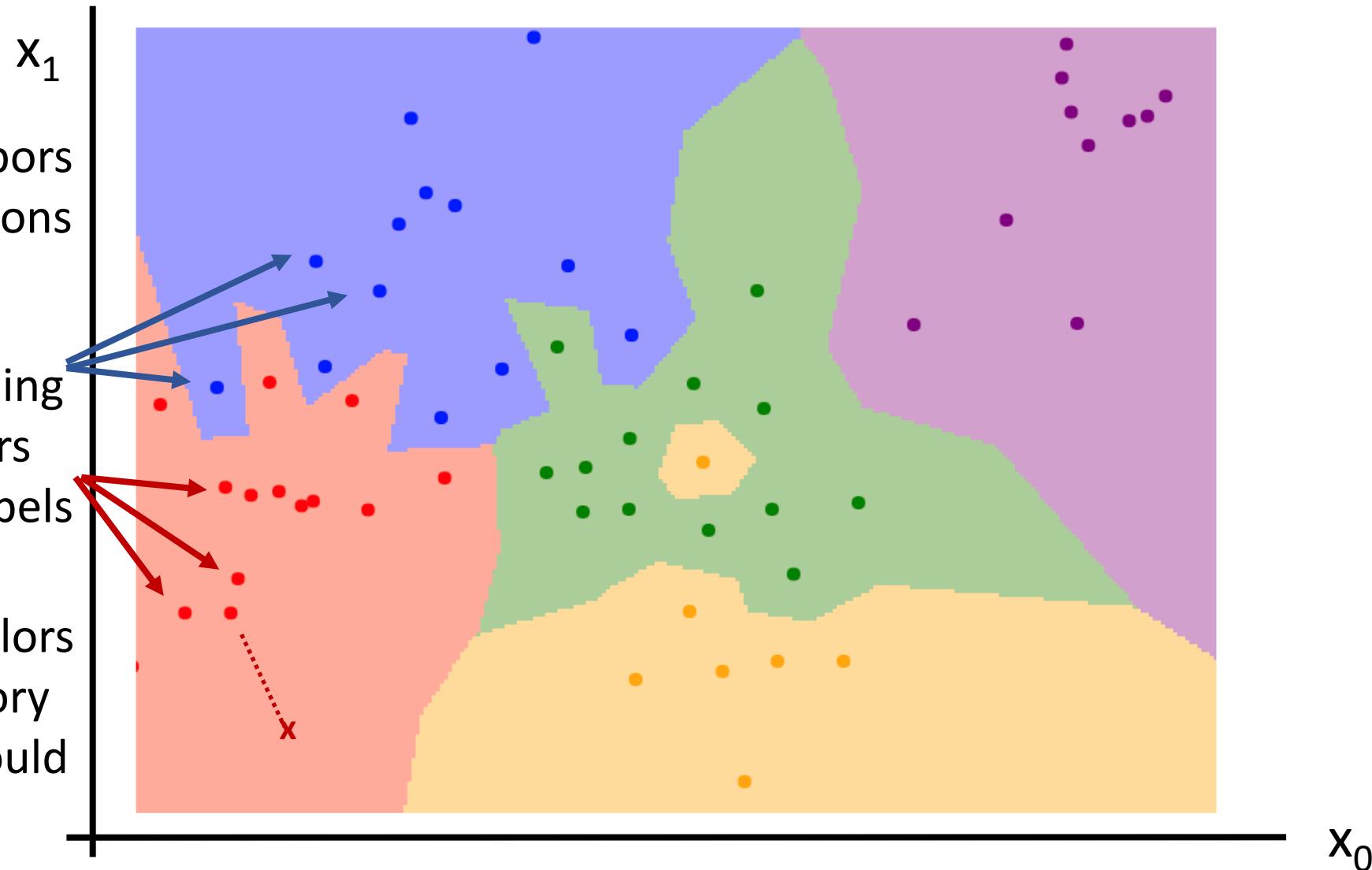


Nearest Neighbor Decision Boundaries

Nearest neighbors
in two dimensions

Points are training
examples; colors
give training labels

Background colors
give the category
a test point would
be assigned



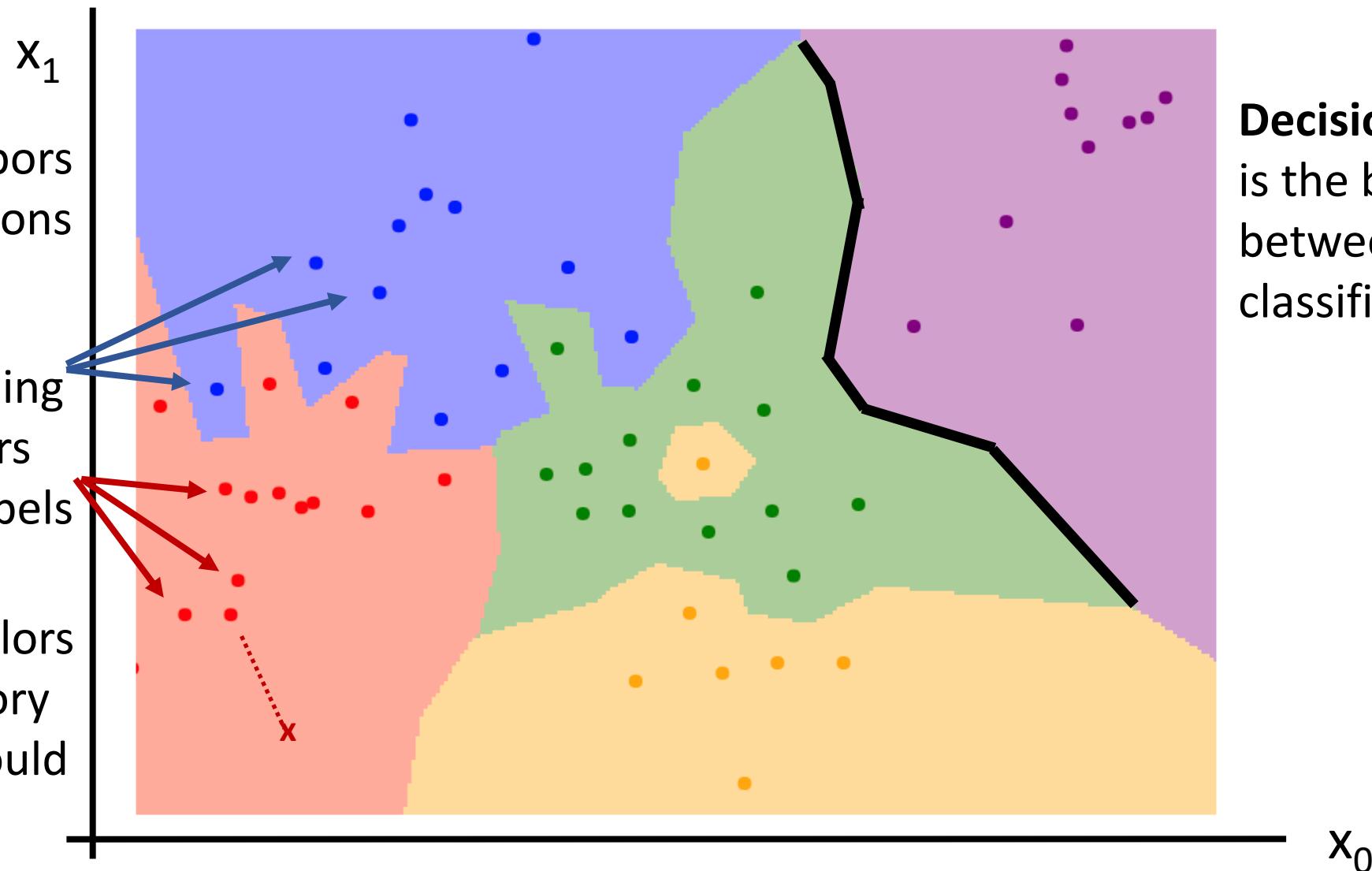
Nearest Neighbor Decision Boundaries

Nearest neighbors
in two dimensions

Points are training
examples; colors
give training labels

Background colors
give the category
a test point would
be assigned

Decision boundary
is the boundary
between two
classification regions

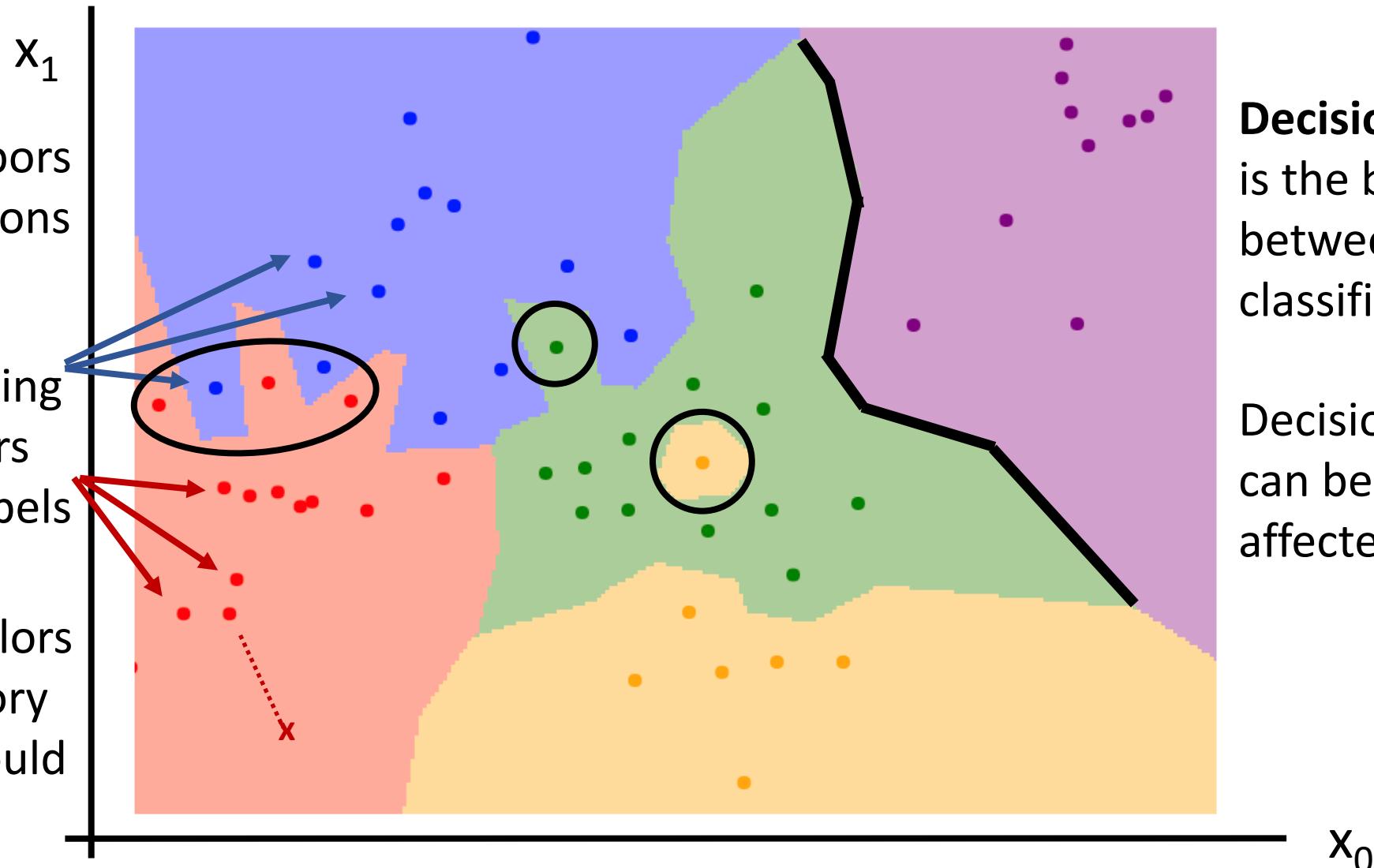


Nearest Neighbor Decision Boundaries

Nearest neighbors
in two dimensions

Points are training
examples; colors
give training labels

Background colors
give the category
a test point would
be assigned



Decision boundary
is the boundary
between two
classification regions

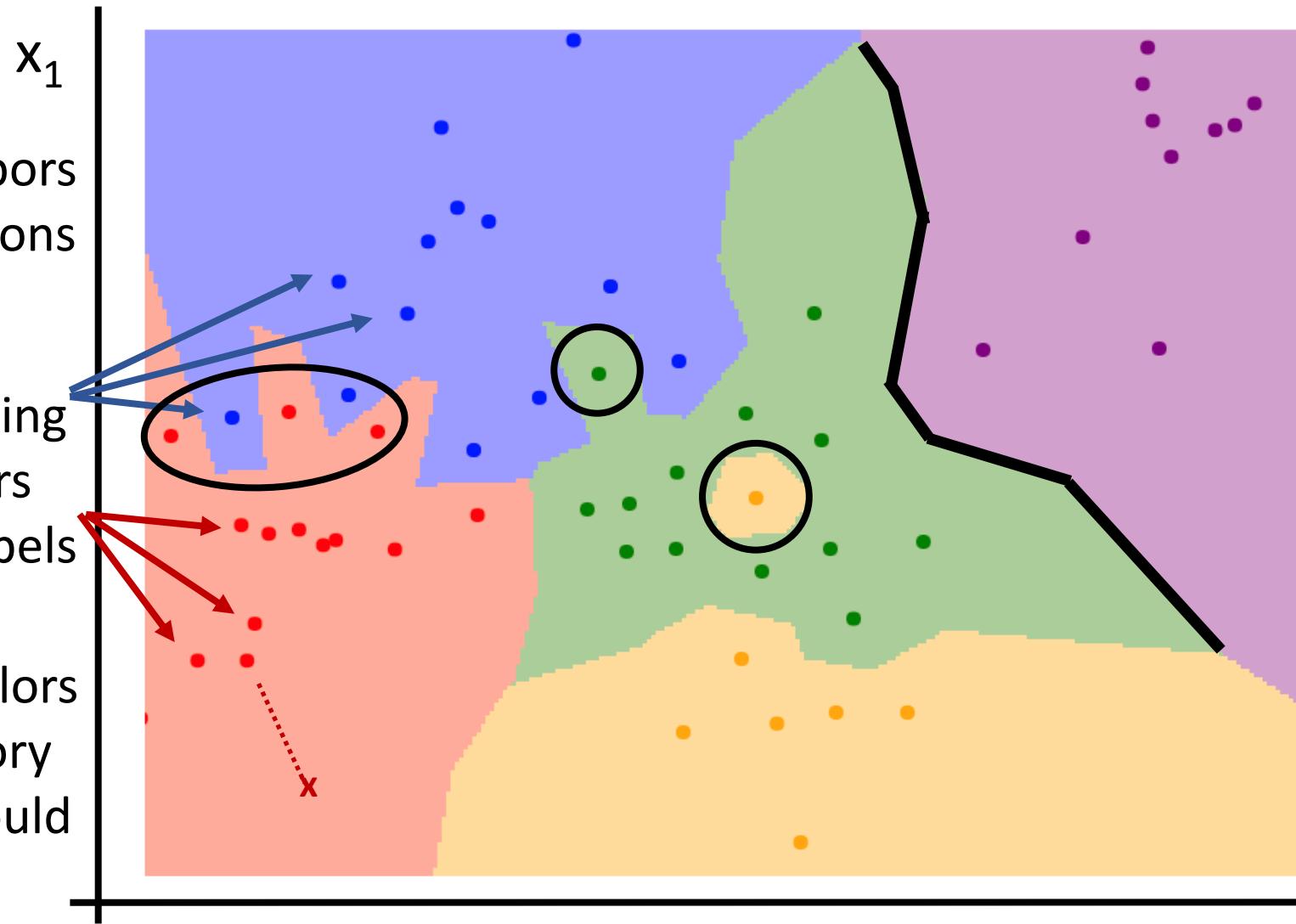
Decision boundaries
can be noisy;
affected by outliers

Nearest Neighbor Decision Boundaries

Nearest neighbors
in two dimensions

Points are training
examples; colors
give training labels

Background colors
give the category
a test point would
be assigned



Decision boundary
is the boundary
between two
classification regions

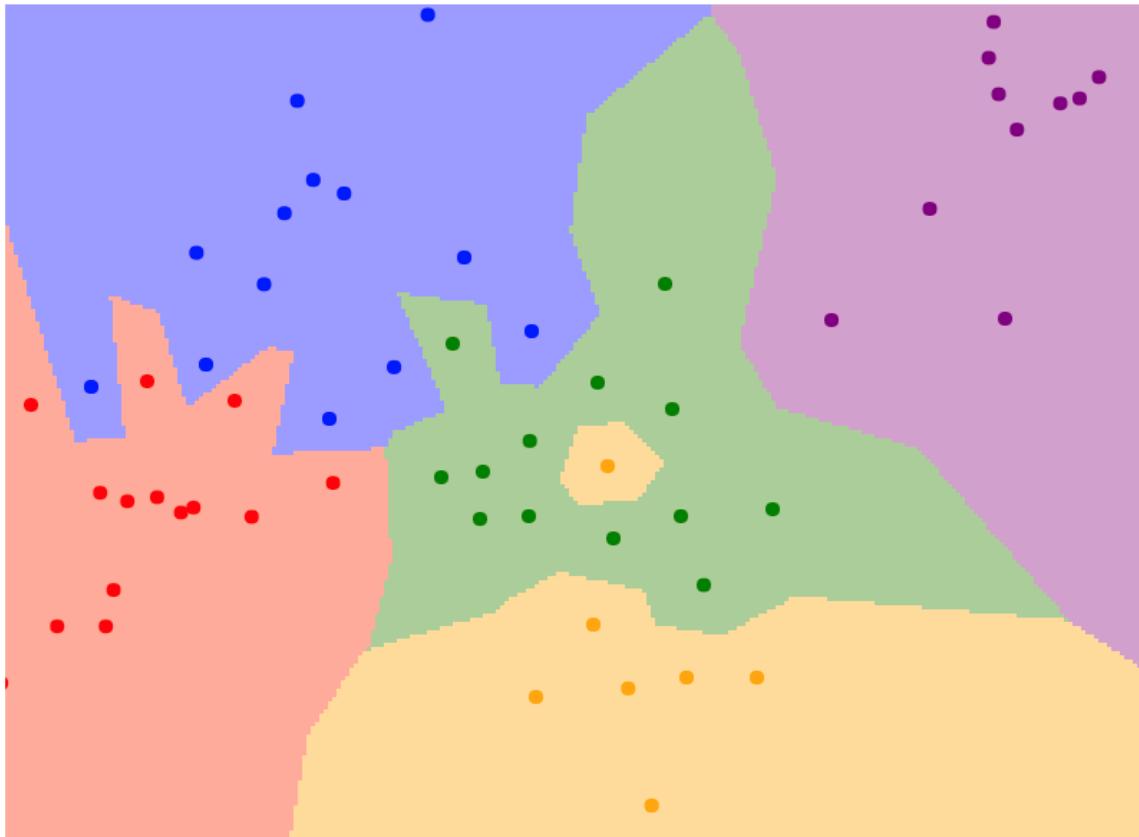
Decision boundaries
can be noisy;
affected by outliers

How to smooth out
decision boundaries?
Use more neighbors!

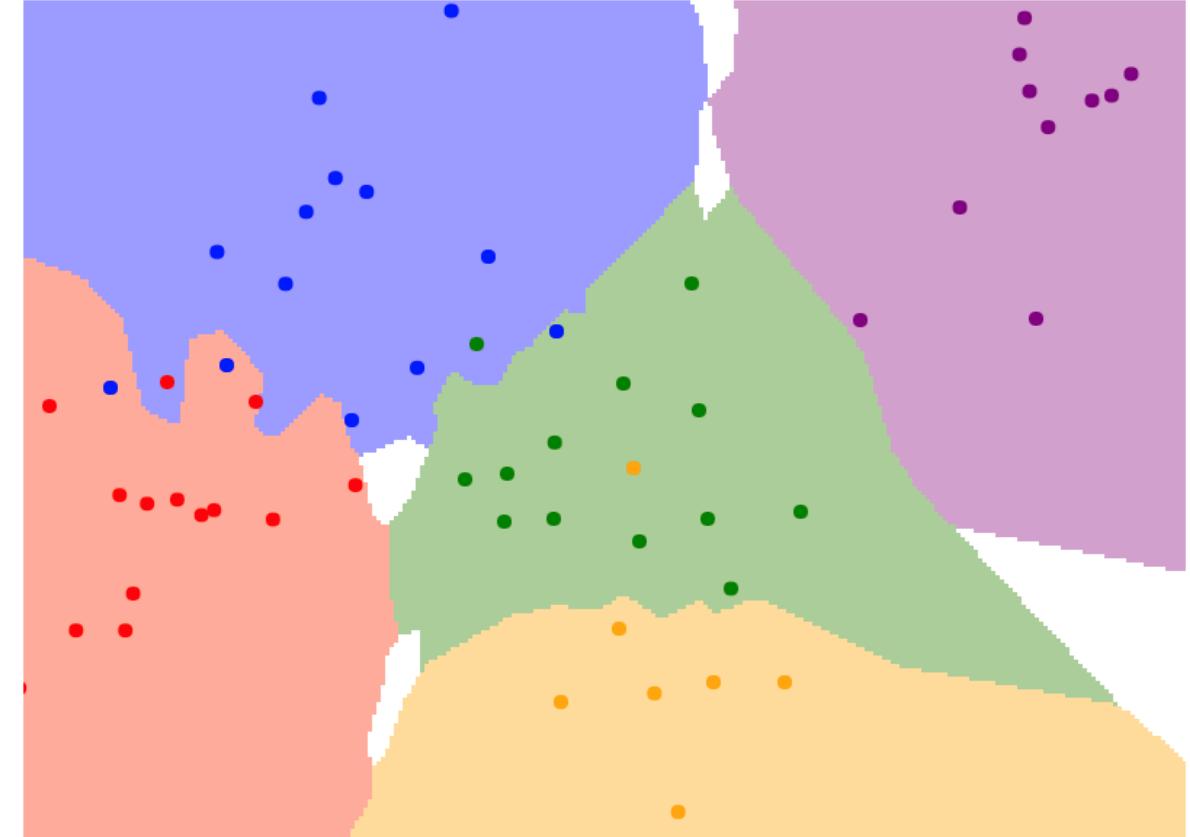
K-Nearest Neighbors

Instead of copying label from nearest neighbor,
take **majority vote** from K closest points

$K = 1$



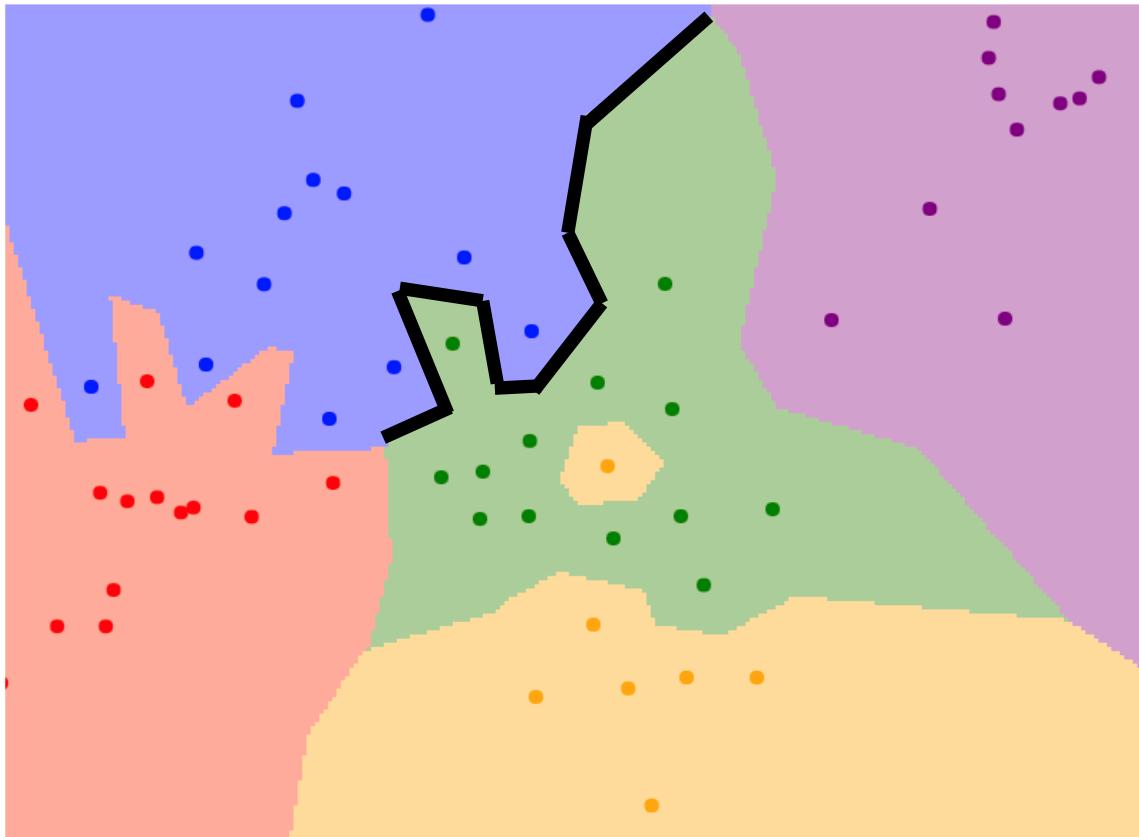
$K = 3$



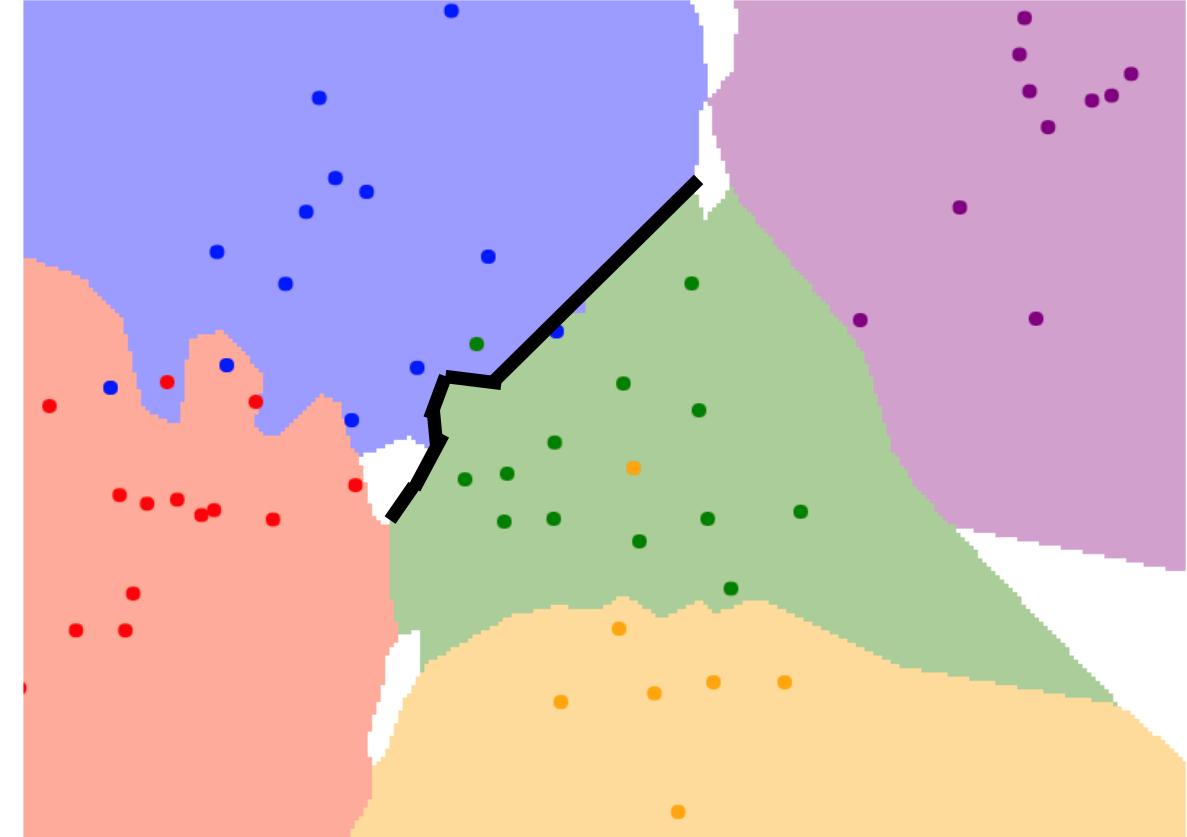
K-Nearest Neighbors

Using more neighbors helps smooth out rough decision boundaries

$K = 1$



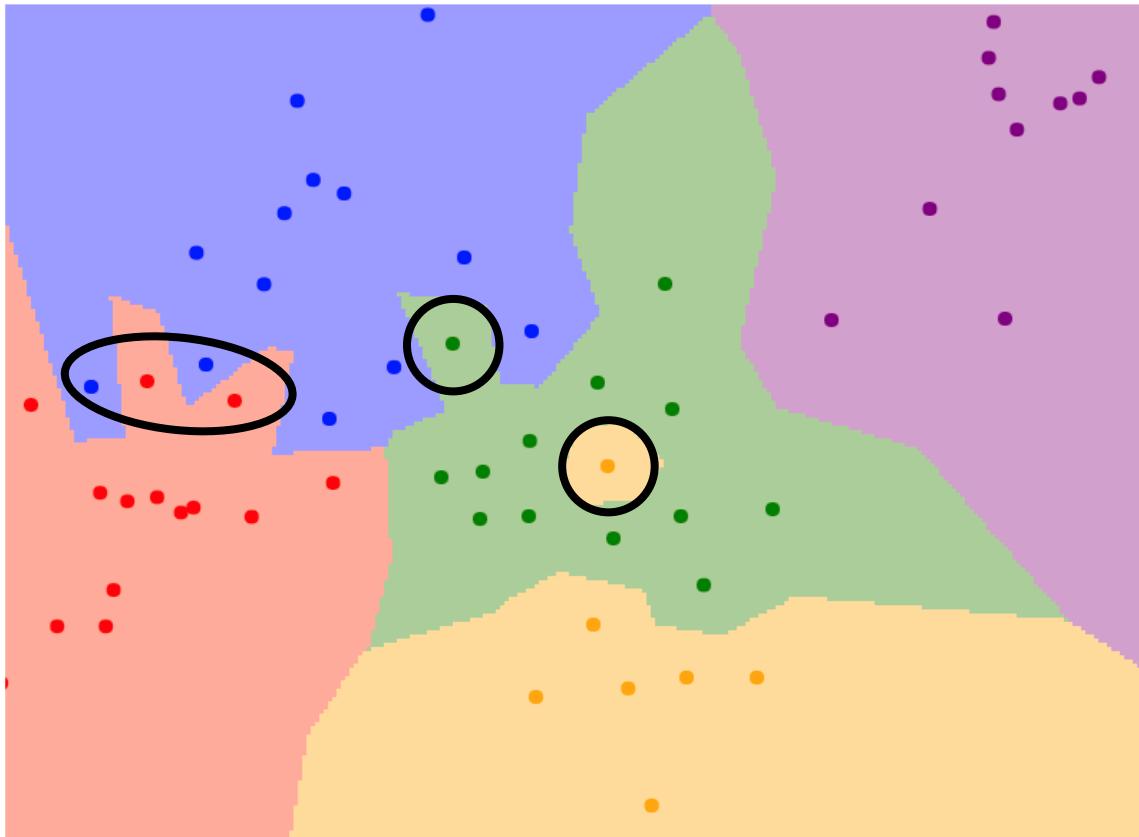
$K = 3$



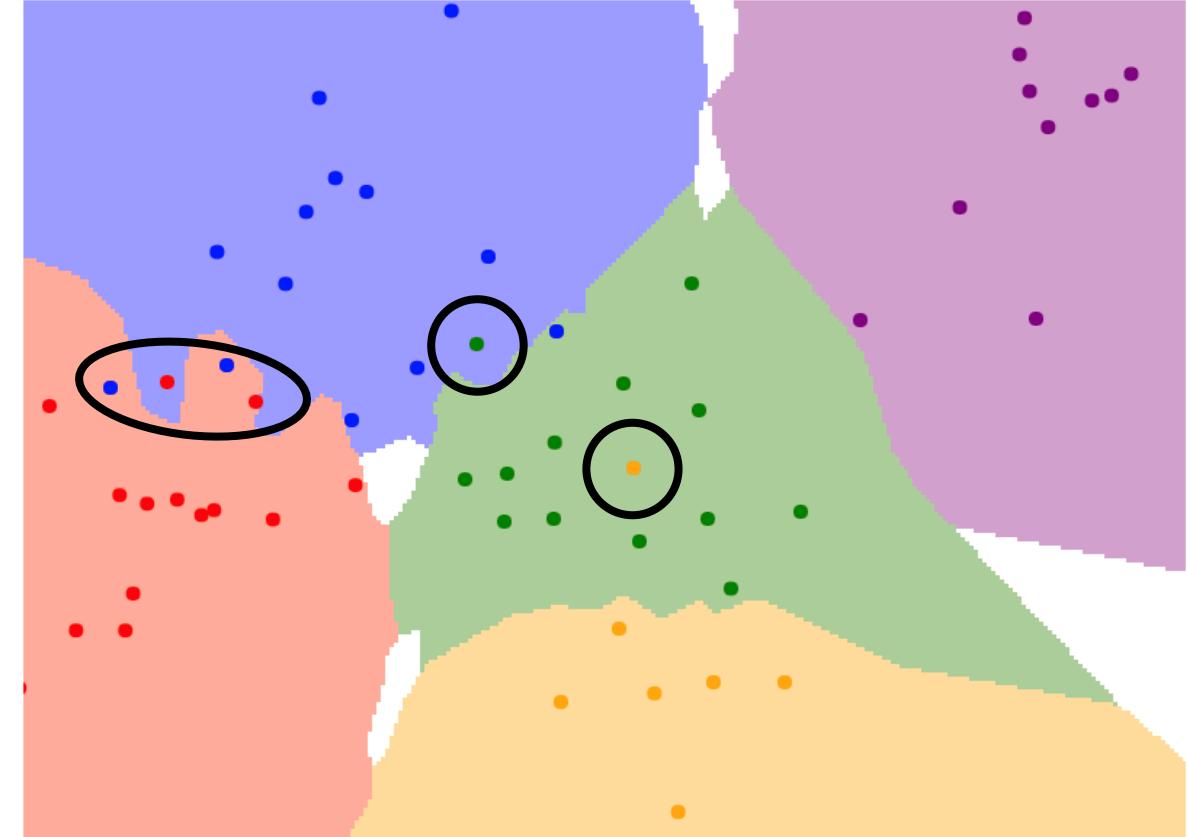
K-Nearest Neighbors

Using more neighbors helps reduce the effect of outliers

$K = 1$



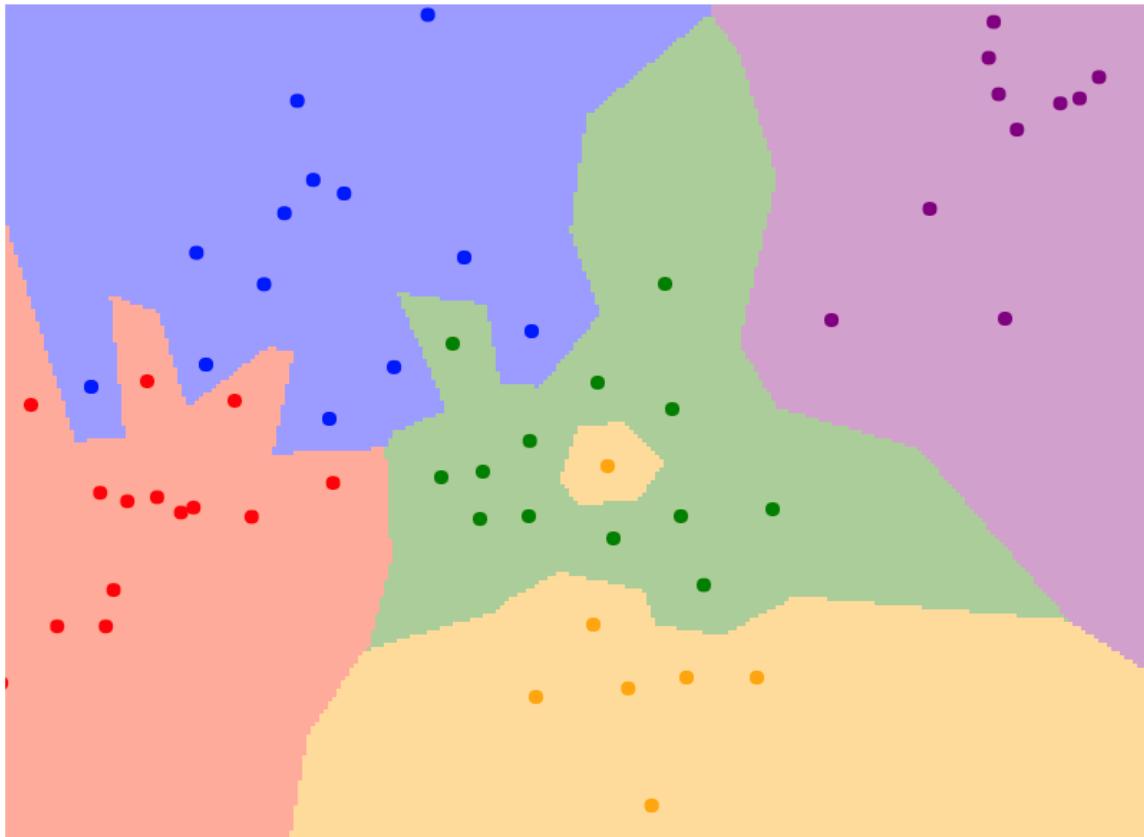
$K = 3$



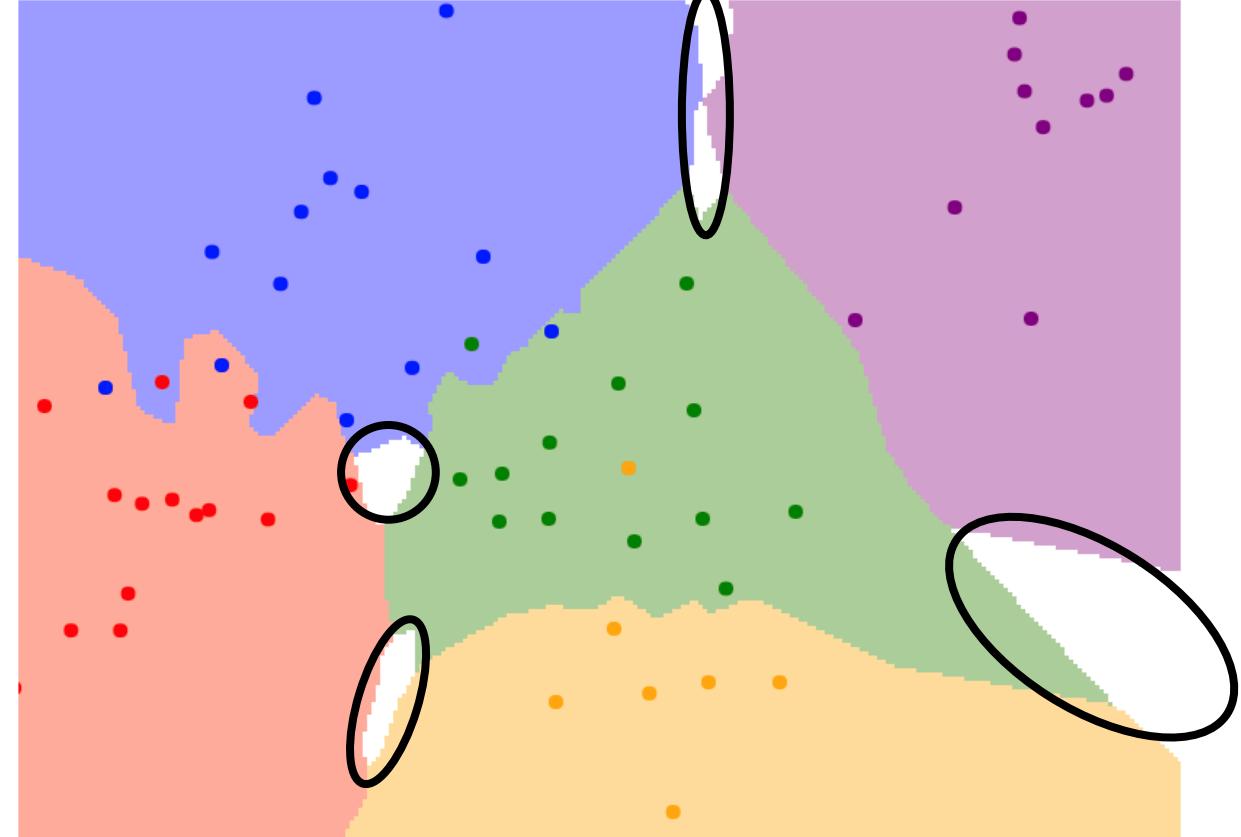
K-Nearest Neighbors

When $K > 1$ there can be ties between classes.
Need to break somehow!

$K = 1$



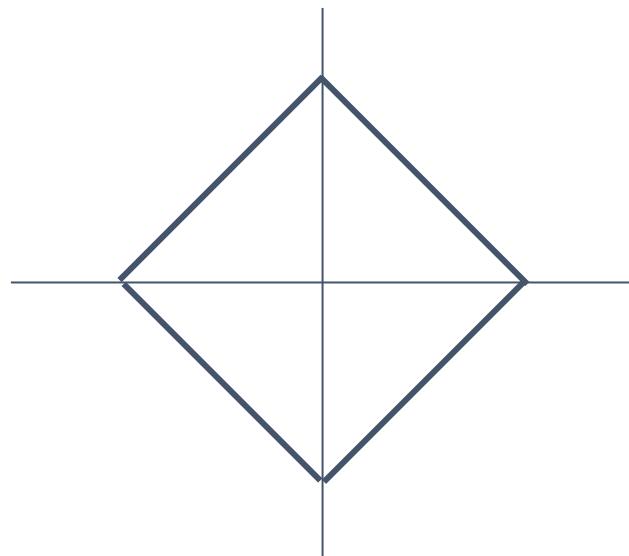
$K = 3$



K-Nearest Neighbors: Distance Metric

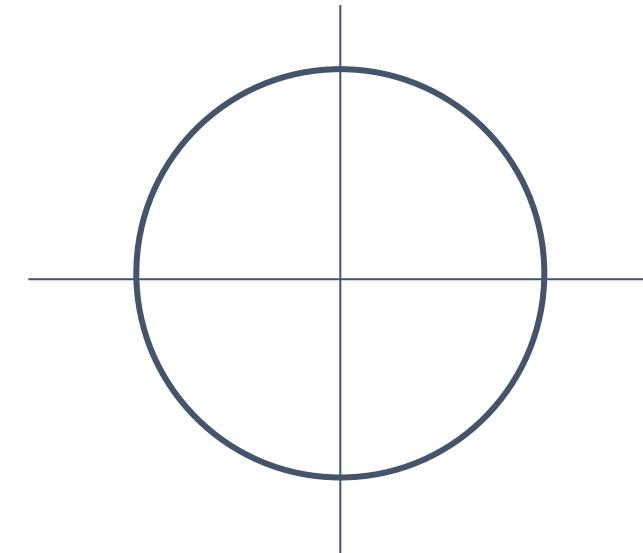
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

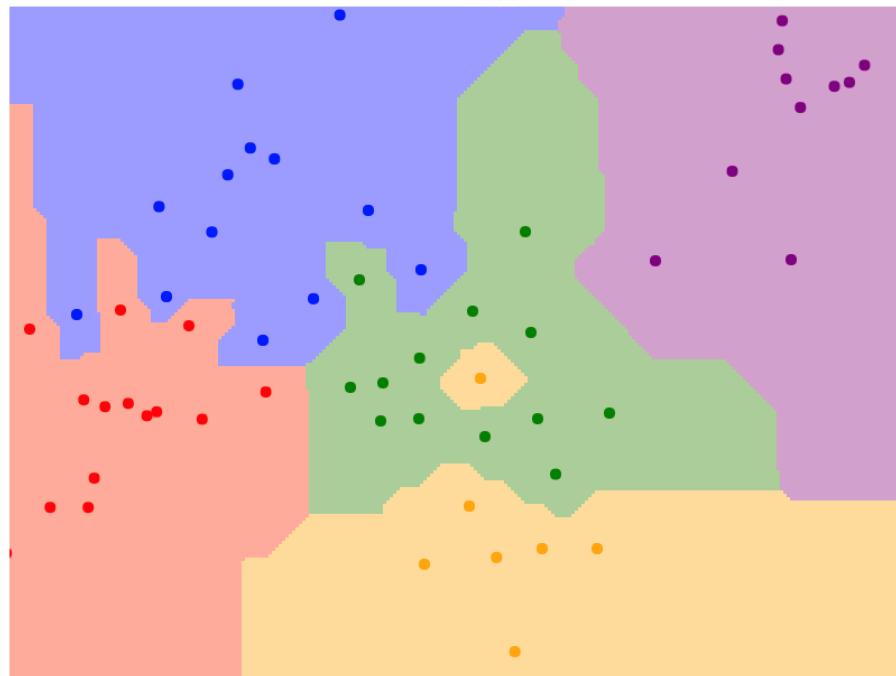
$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K-Nearest Neighbors: Distance Metric

L1 (Manhattan) distance

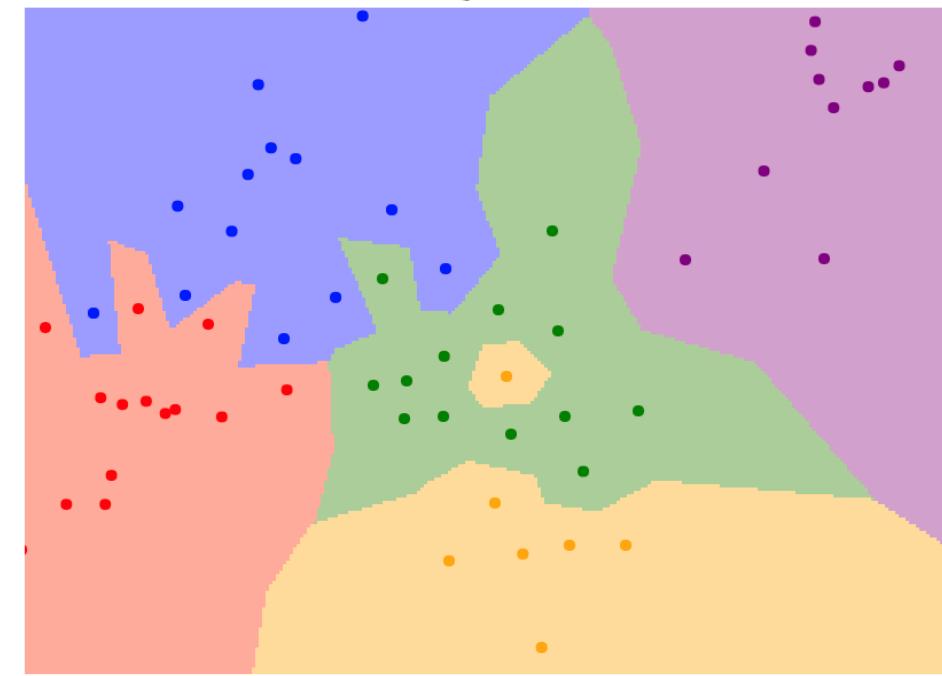
$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



$K = 1$

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



$K = 1$

K-Nearest Neighbors: Distance Metric

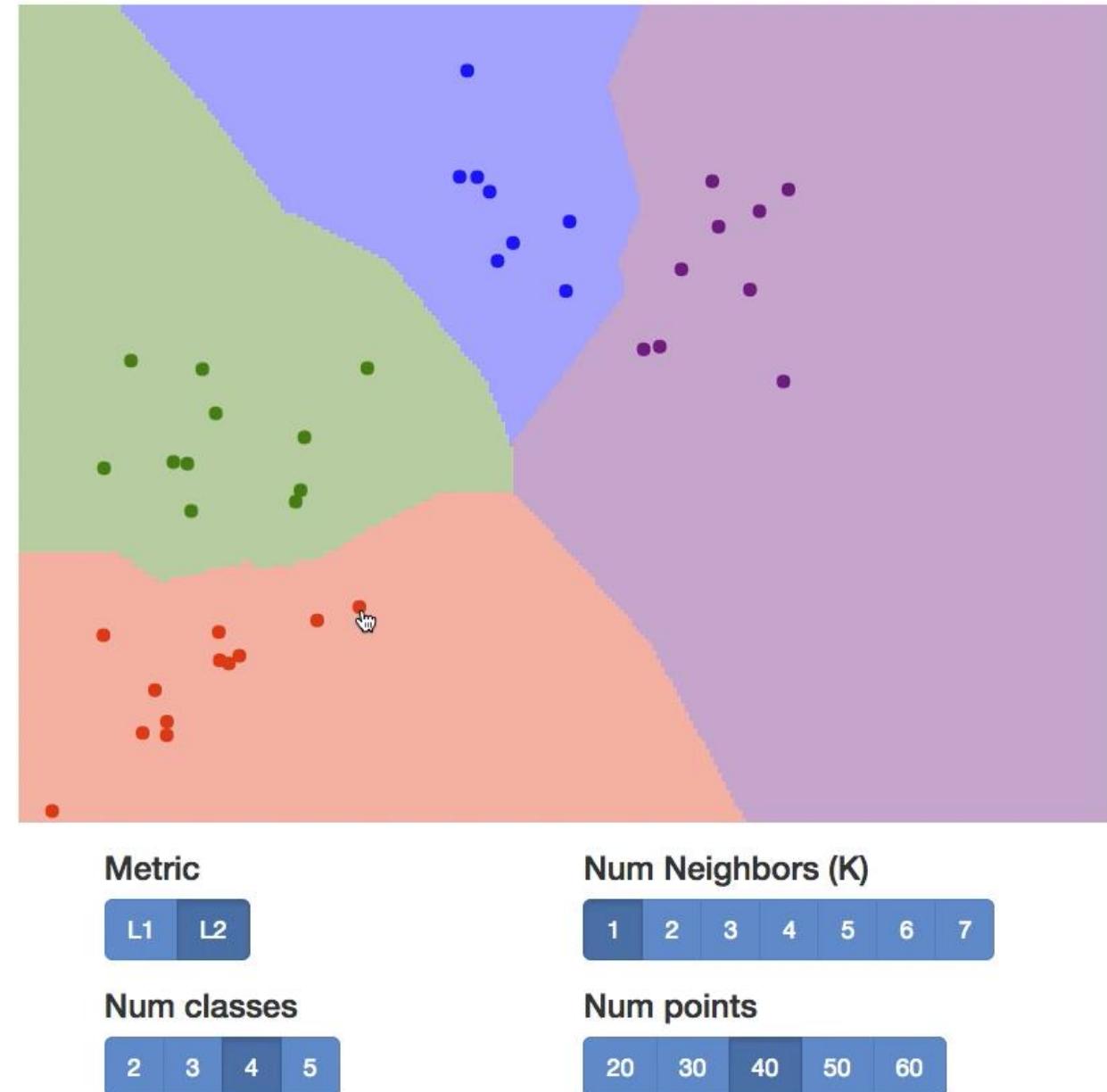
With the right choice of distance metric, we can apply K-Nearest Neighbor to any type of data!

K-Nearest Neighbors: Web Demo

Interactively move points around
and see decision boundaries change

Play with L1 vs L2 metrics

Play with changing number of
training points, value of K



<http://vision.stanford.edu/teaching/cs231n-demos/knn/>

Hyperparameters

What is the best value of **K** to use?

What is the best **distance metric** to use?

These are examples of **hyperparameters**: choices about our learning algorithm that we don't learn from the training data; instead we set them at the start of the learning process

Hyperparameters

What is the best value of **K** to use?

What is the best **distance metric** to use?

These are examples of **hyperparameters**: choices about our learning algorithm that we don't learn from the training data; instead we set them at the start of the learning process

Very problem-dependent. In general need to try them all and see what works best for our data / task.

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

Your Dataset

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: $K = 1$ always works perfectly on training data

Your Dataset

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

train

test

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train

test

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train

test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train

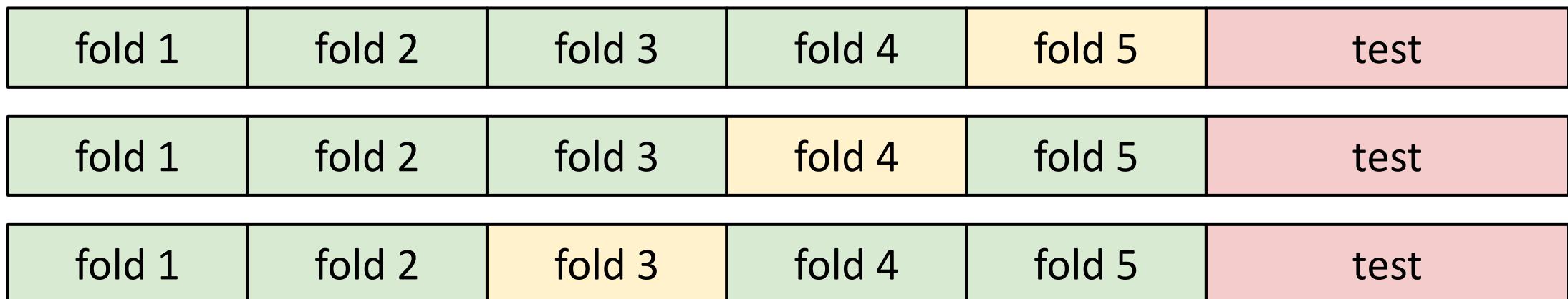
validation

test

Setting Hyperparameters

Your Dataset

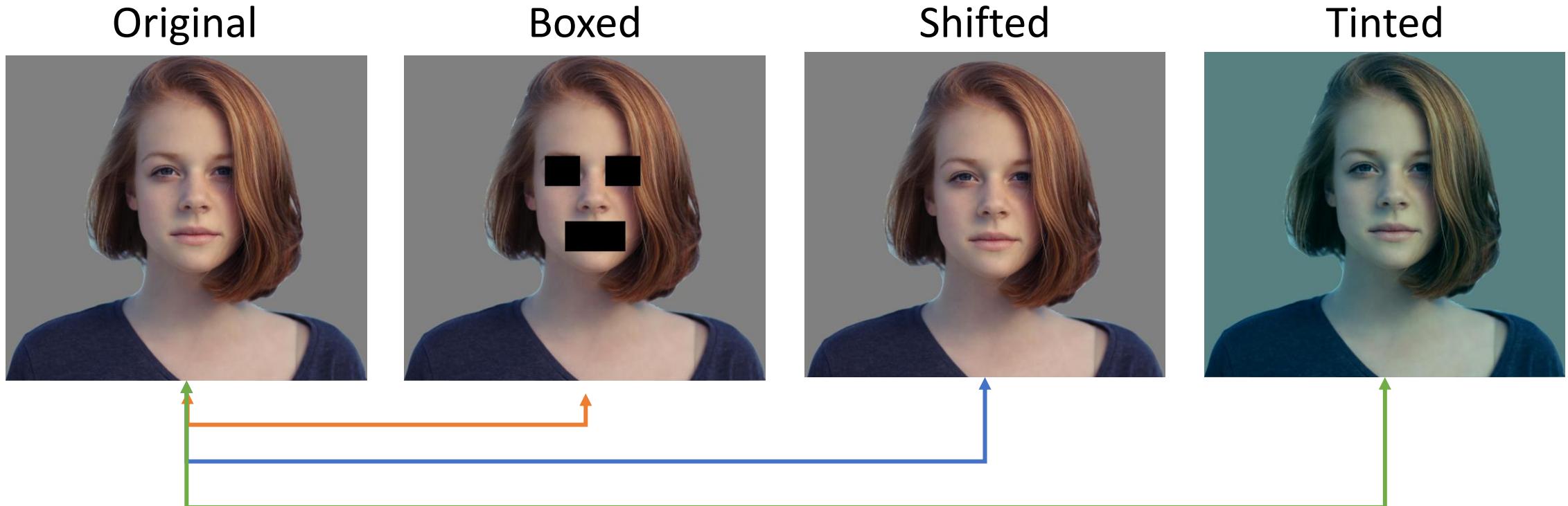
Idea #4: Cross-Validation: Split data into **folds**, try each fold as validation and average the results



Useful for small datasets, but (unfortunately) not used too frequently in deep learning

K-Nearest Neighbor on raw pixels is seldom used

- Very slow at test time
- Distance metrics on pixels are not informative



(all 3 images have same L2 distance to the one on the left)

Original image is
CC0 public domain

Nearest Neighbor with Convolutional Neural Network (ConvNet or CNN) features works well!



Devlin et al, "Exploring Nearest Neighbor Approaches for Image Captioning", 2015

Nearest Neighbor with ConvNet features works well!

Example: Image Captioning with Nearest Neighbor



A bedroom with a bed and a couch.



A cat sitting in a bathroom sink.



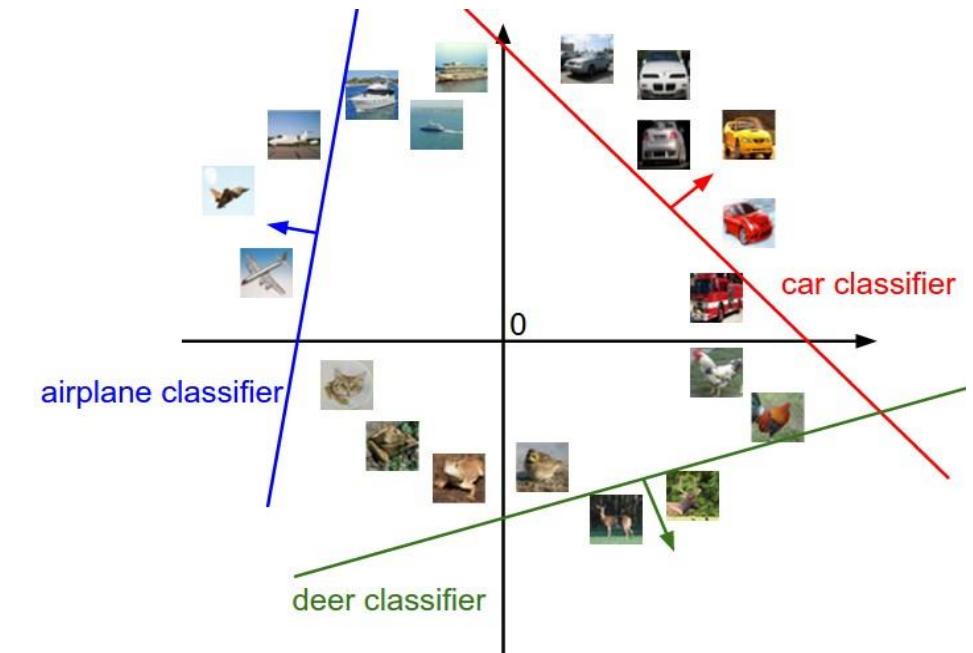
A train is stopped at a train station.



A wooden bench in front of a building.

Devlin et al, "Exploring Nearest Neighbor Approaches for Image Captioning", 2015

Lecture 3: Linear Classifiers



Neural Network

Linear
classifiers



[This image](#) is [CC0 1.0](#) public domain

Recall CIFAR10

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck

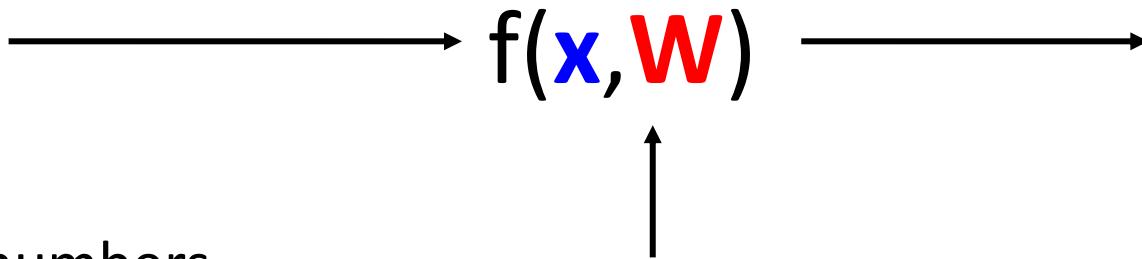


50,000 training images
each image is **32x32x3**

10,000 test images.

Parametric Approach

Image



10 numbers giving
class scores

Array of **32x32x3** numbers
(3072 numbers total)

W
parameters
or weights

Parametric Approach: Linear Classifier

Image



$$f(x, W) = Wx$$

Array of **32x32x3** numbers
(3072 numbers total)

$$f(\mathbf{x}, \mathbf{W})$$

10 numbers giving
class scores

W

parameters
or weights

Parametric Approach: Linear Classifier $(3072,)$

Image



Array of $32 \times 32 \times 3$ numbers
(3072 numbers total)

$$f(x, W) = Wx$$

(10,) (10, 3072)

W
parameters
or weights

10 numbers giving
class scores

Parametric Approach: Linear Classifier

(3072,)

Image



$$f(x, W) = Wx + b \quad (10,)$$

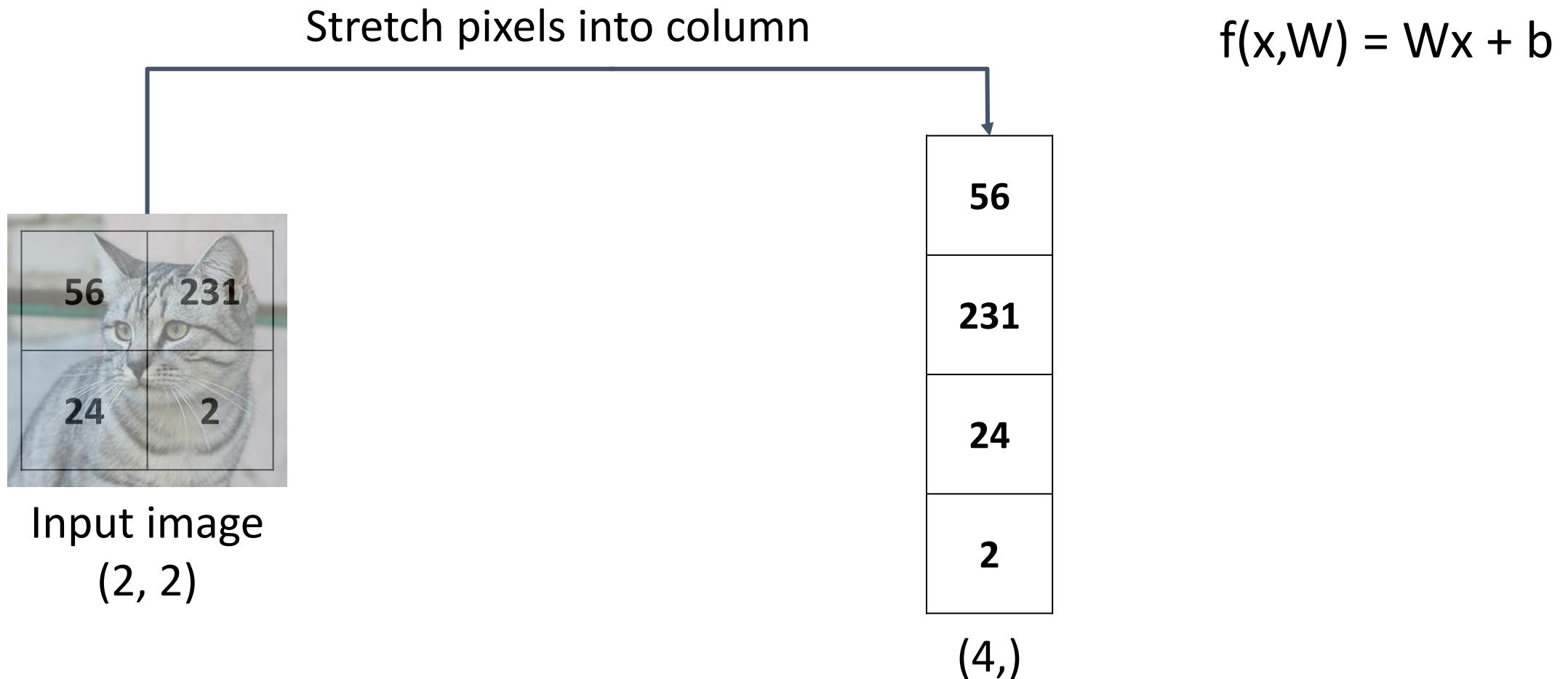
(10,) (10, 3072)

10 numbers giving
class scores

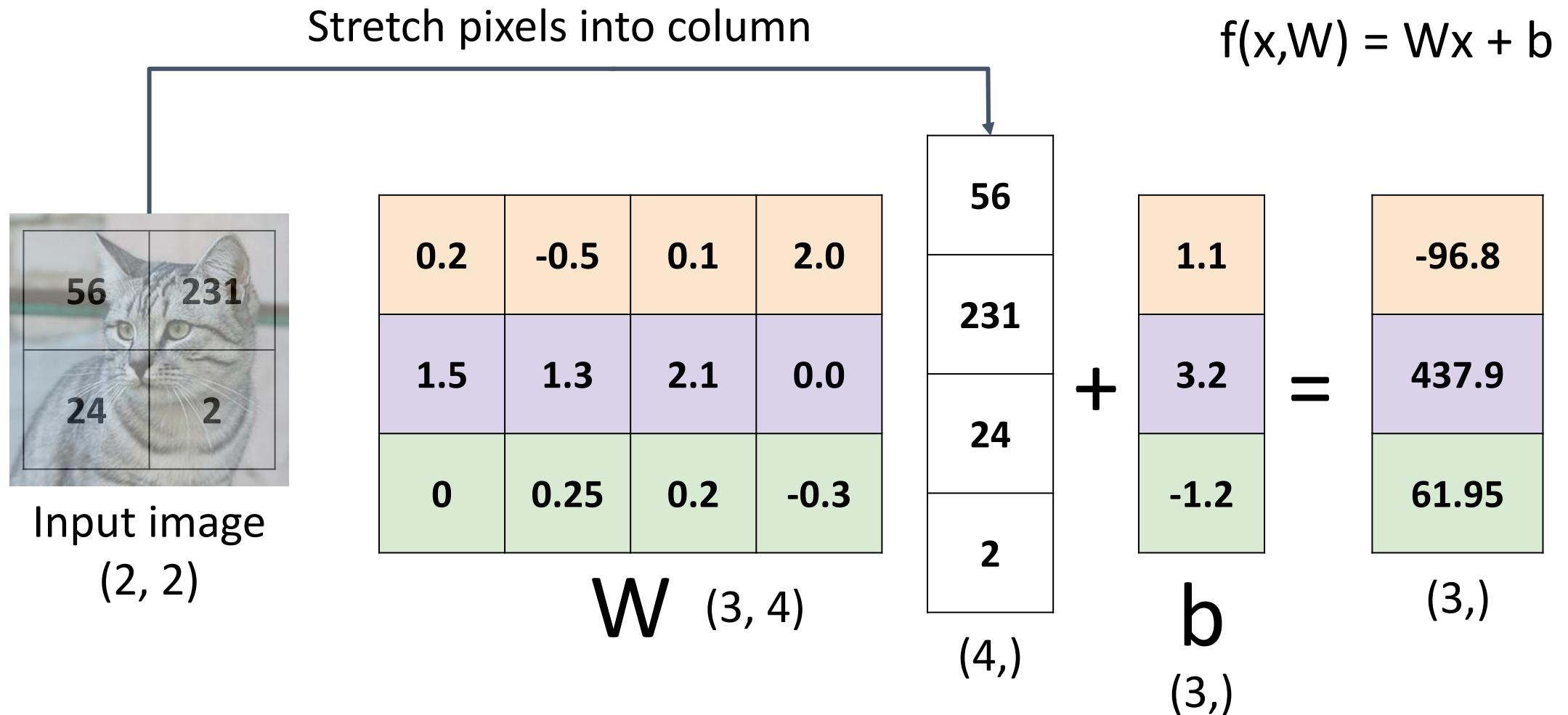
Array of **32x32x3** numbers
(3072 numbers total)

W
parameters
or weights

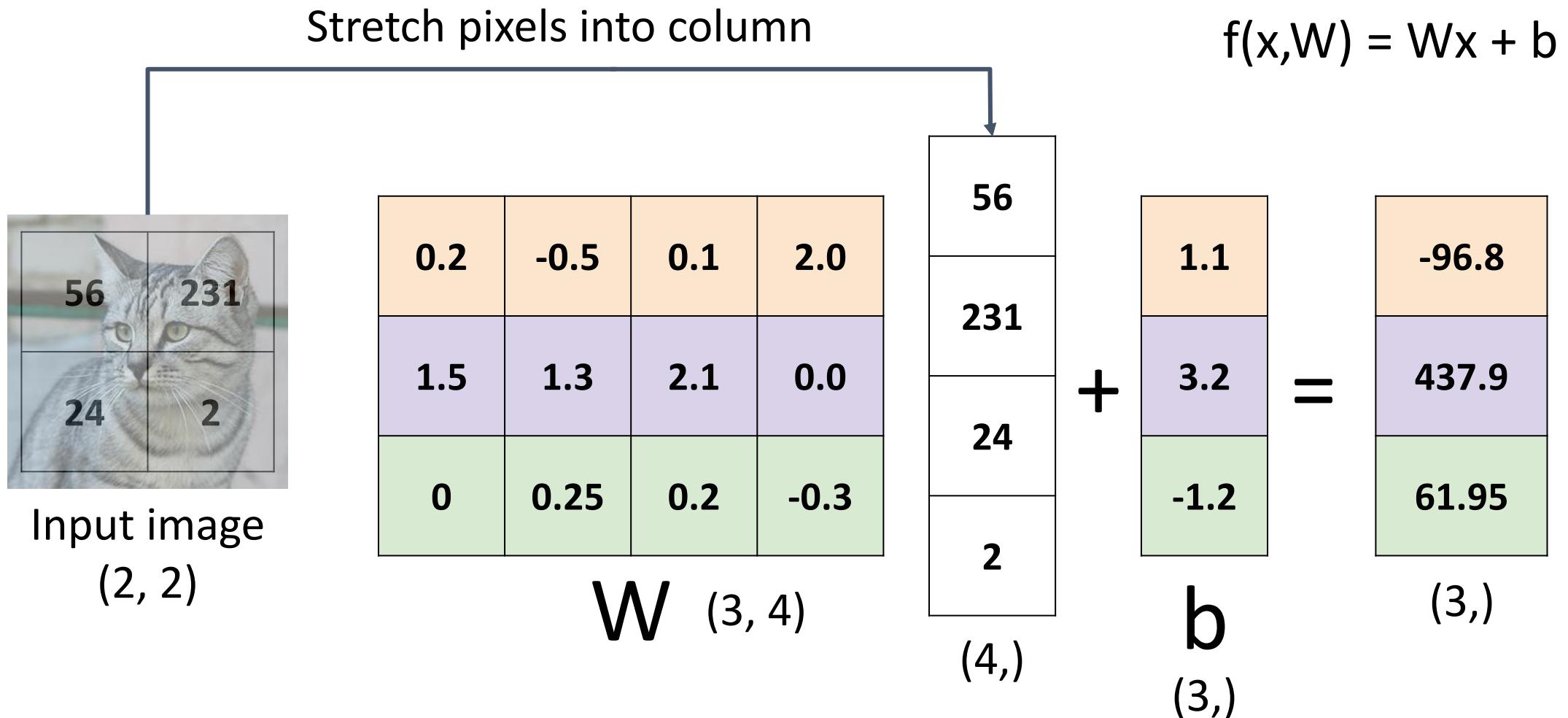
Example for 2x2 image, 3 classes (cat/dog/ship)



Example for 2x2 image, 3 classes (cat/dog/ship)



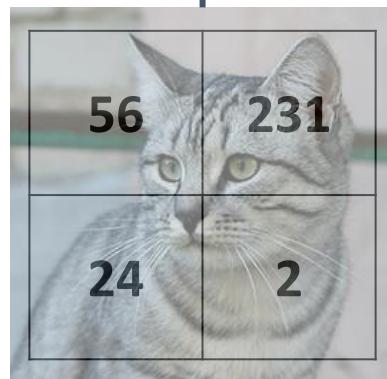
Linear Classifier: Algebraic Viewpoint



Linear Classifier: Bias Trick

Add extra one to data vector;
bias is absorbed into last
column of weight matrix

Stretch pixels into column



Input image
(2, 2)

0.2	-0.5	0.1	2.0	1.1
1.5	1.3	2.1	0.0	3.2
0	0.25	0.2	-0.3	-1.2

W (3, 5)



$$\begin{matrix} 56 \\ 231 \\ 24 \\ 2 \\ 1 \end{matrix} = \begin{matrix} -96.8 \\ 437.9 \\ 61.95 \end{matrix} \quad (3,) \quad (5,)$$

Linear Classifier: Predictions are Linear!

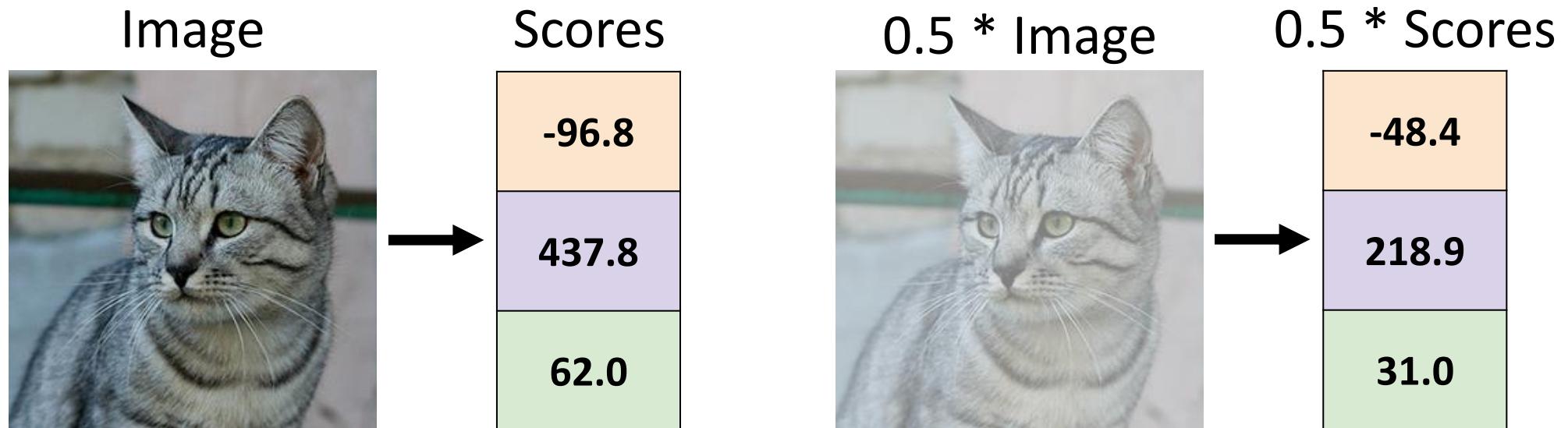
$$f(x, W) = Wx \quad (\text{ignore bias})$$

$$f(cx, W) = W(cx) = c * f(x, W)$$

Linear Classifier: Predictions are Linear!

$$f(x, W) = Wx \quad (\text{ignore bias})$$

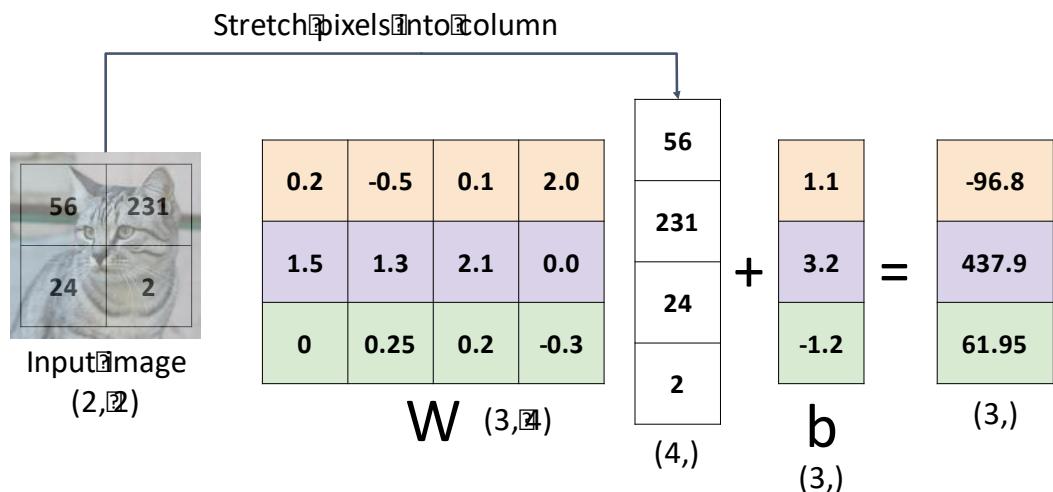
$$f(cx, W) = W(cx) = c * f(x, W)$$



Interpreting a Linear Classifier

Algebraic Viewpoint

$$f(x, W) = Wx + b$$

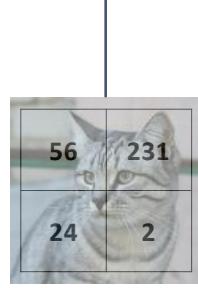


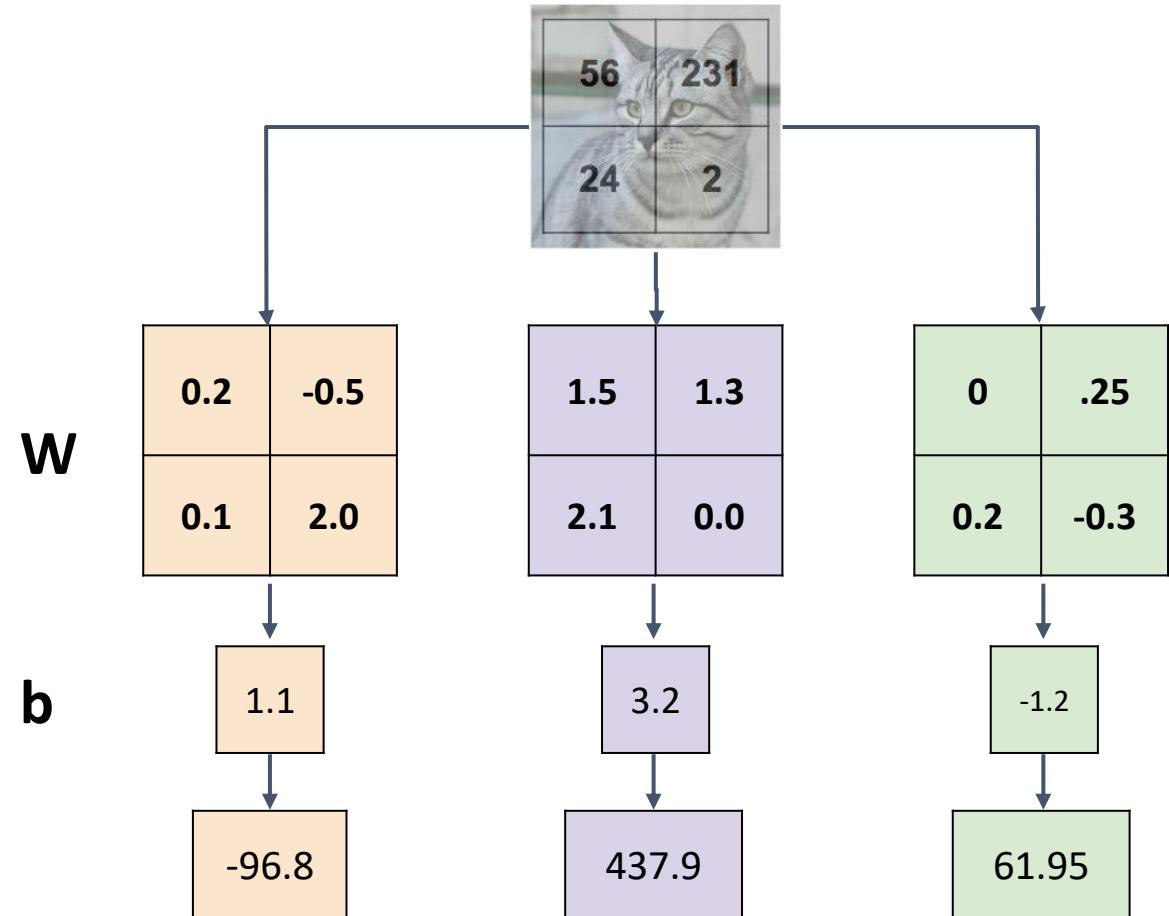
Interpreting a Linear Classifier

Algebraic Viewpoint

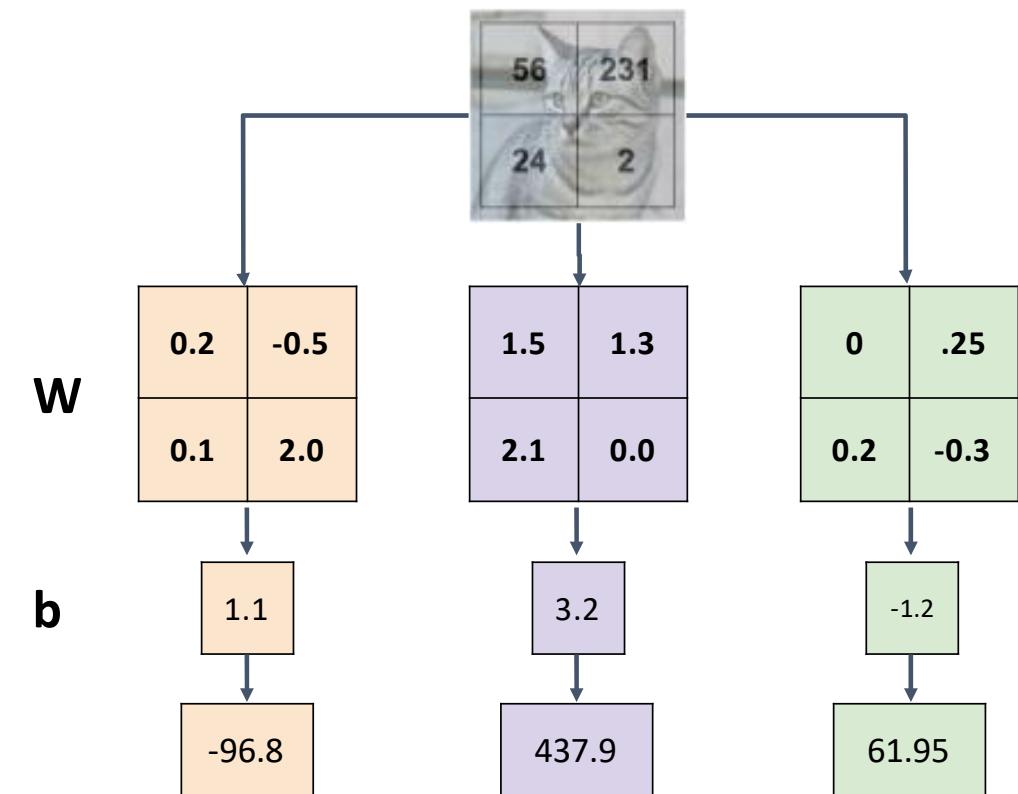
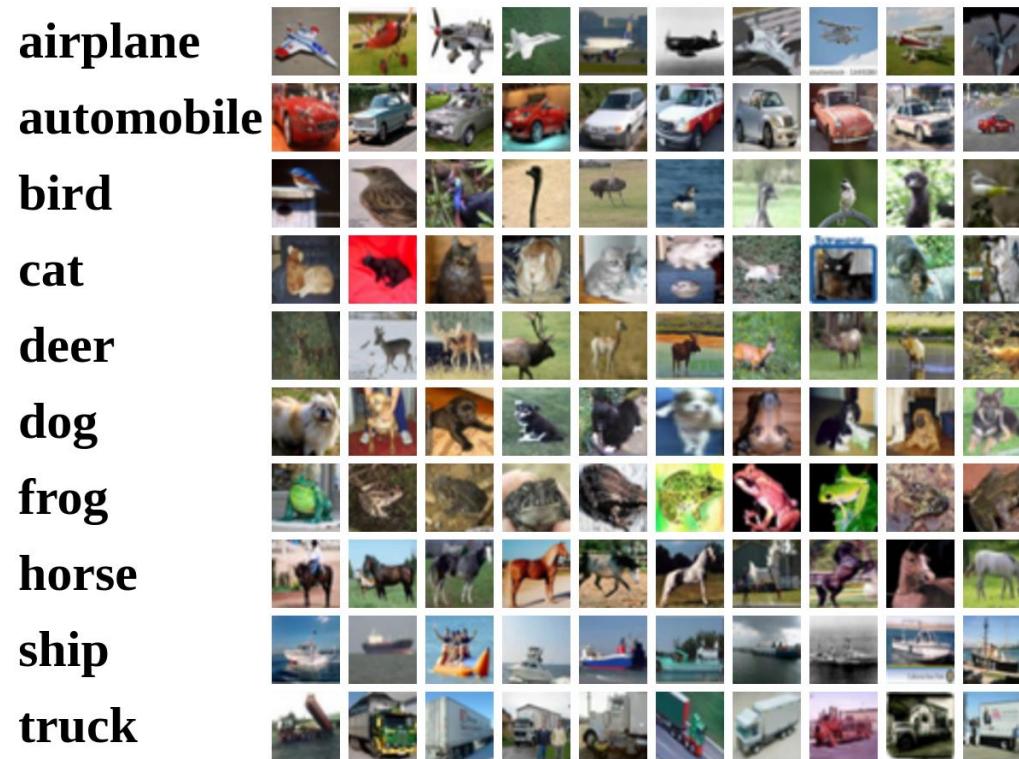
$$f(x, W) = Wx + b$$

Stretch pixels into column

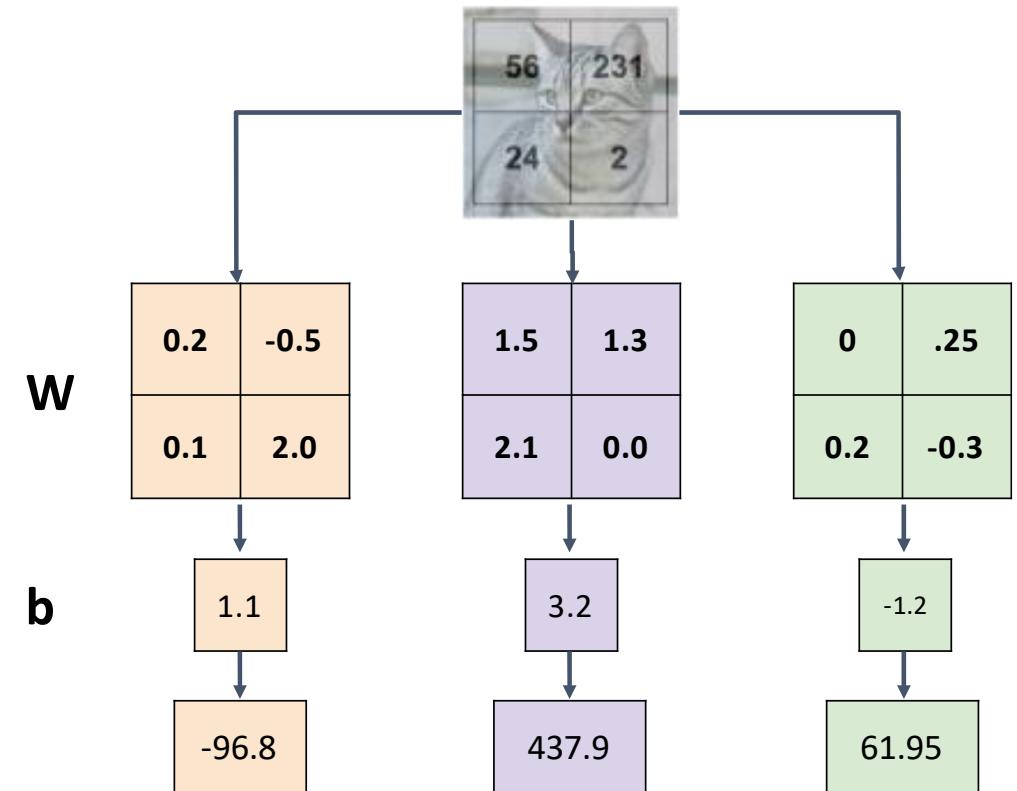
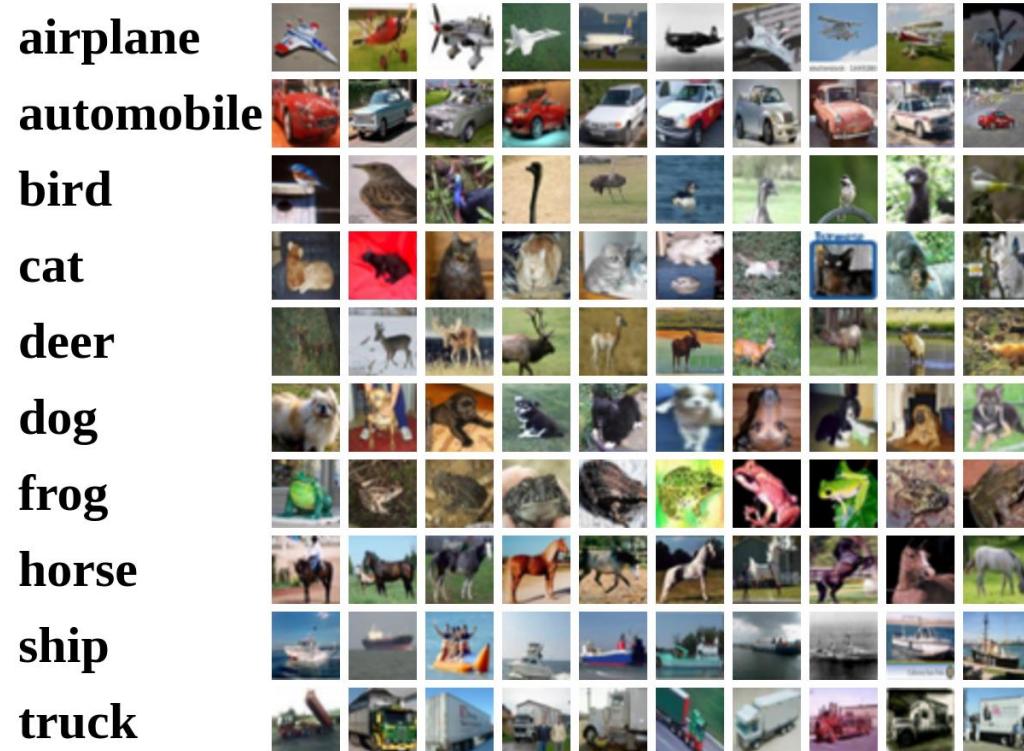

$$\text{Input Image } (2, 2) \rightarrow \begin{pmatrix} 56 & 231 \\ 24 & 2 \end{pmatrix}$$
$$W \quad (3, 4)$$
$$\begin{pmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{pmatrix}$$
$$b \quad (3,)$$
$$\begin{pmatrix} 56 \\ 231 \\ 24 \\ 2 \end{pmatrix}$$
$$+ \begin{pmatrix} 1.1 \\ 3.2 \\ -1.2 \end{pmatrix} = \begin{pmatrix} -96.8 \\ 437.9 \\ 61.95 \end{pmatrix} \quad (3,)$$



Interpreting a Linear Classifier

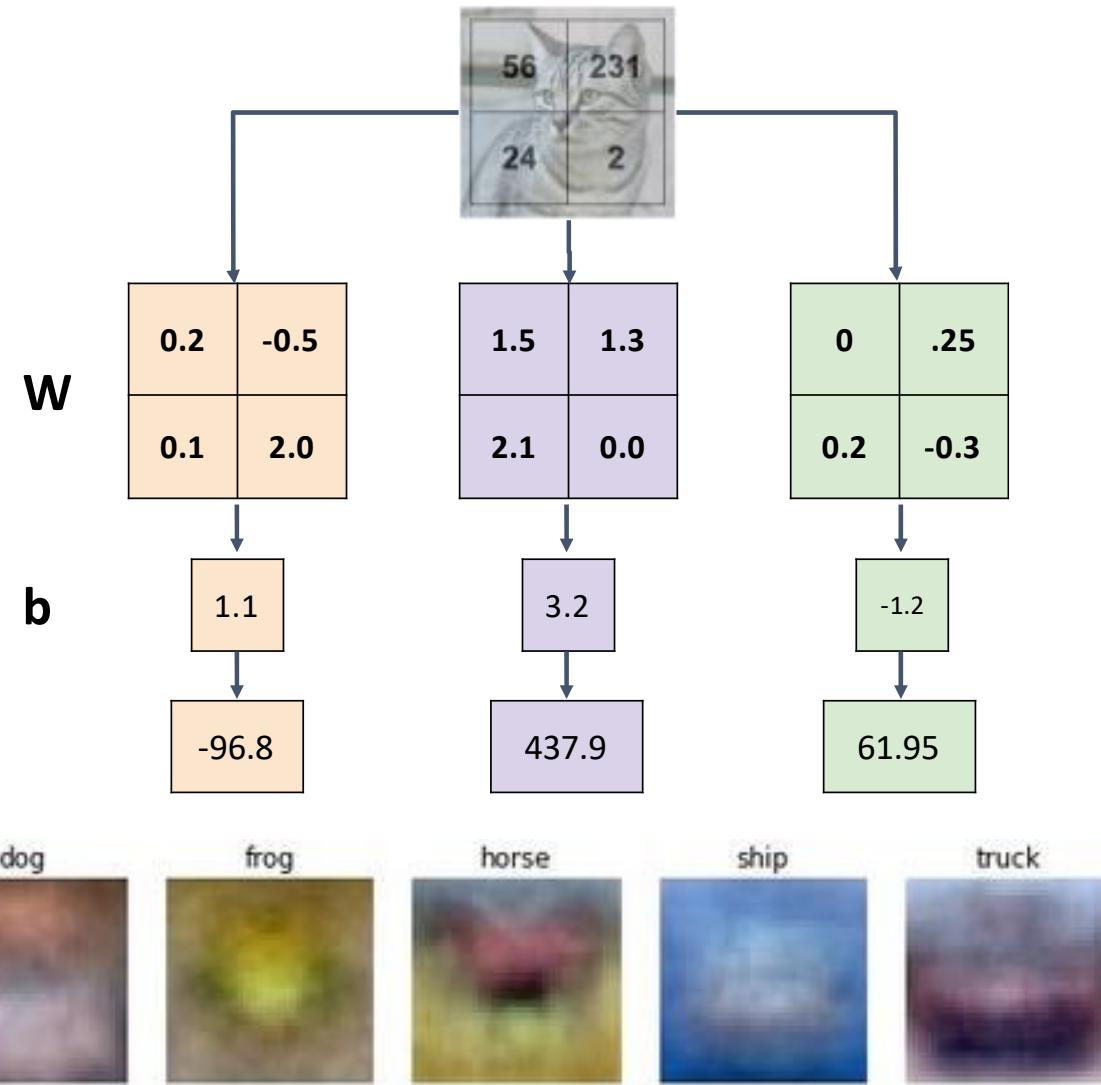


Interpreting a Linear Classifier: Visual Viewpoint



Interpreting a Linear Classifier: Visual Viewpoint

Linear classifier has one
“template” per category

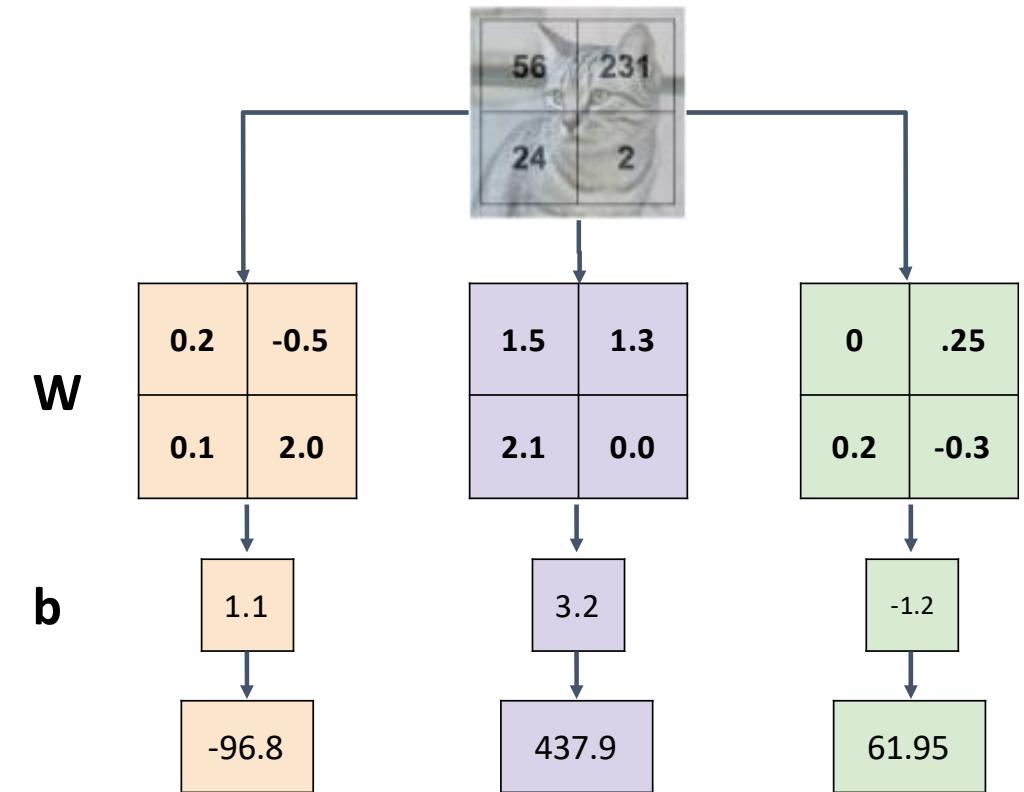


Interpreting a Linear Classifier: Visual Viewpoint

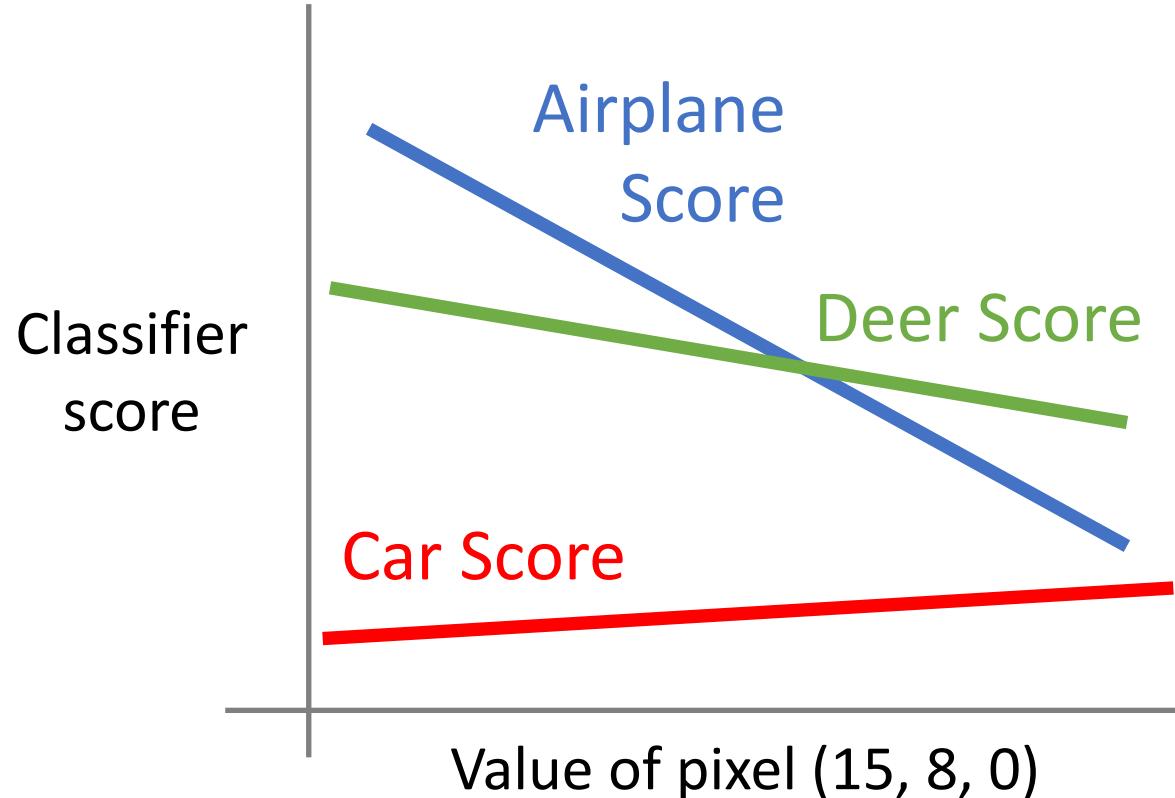
Linear classifier has one
“template” per category

A single template cannot capture
multiple modes of the data

e.g. horse template has 2 heads!



Interpreting a Linear Classifier: Geometric Viewpoint

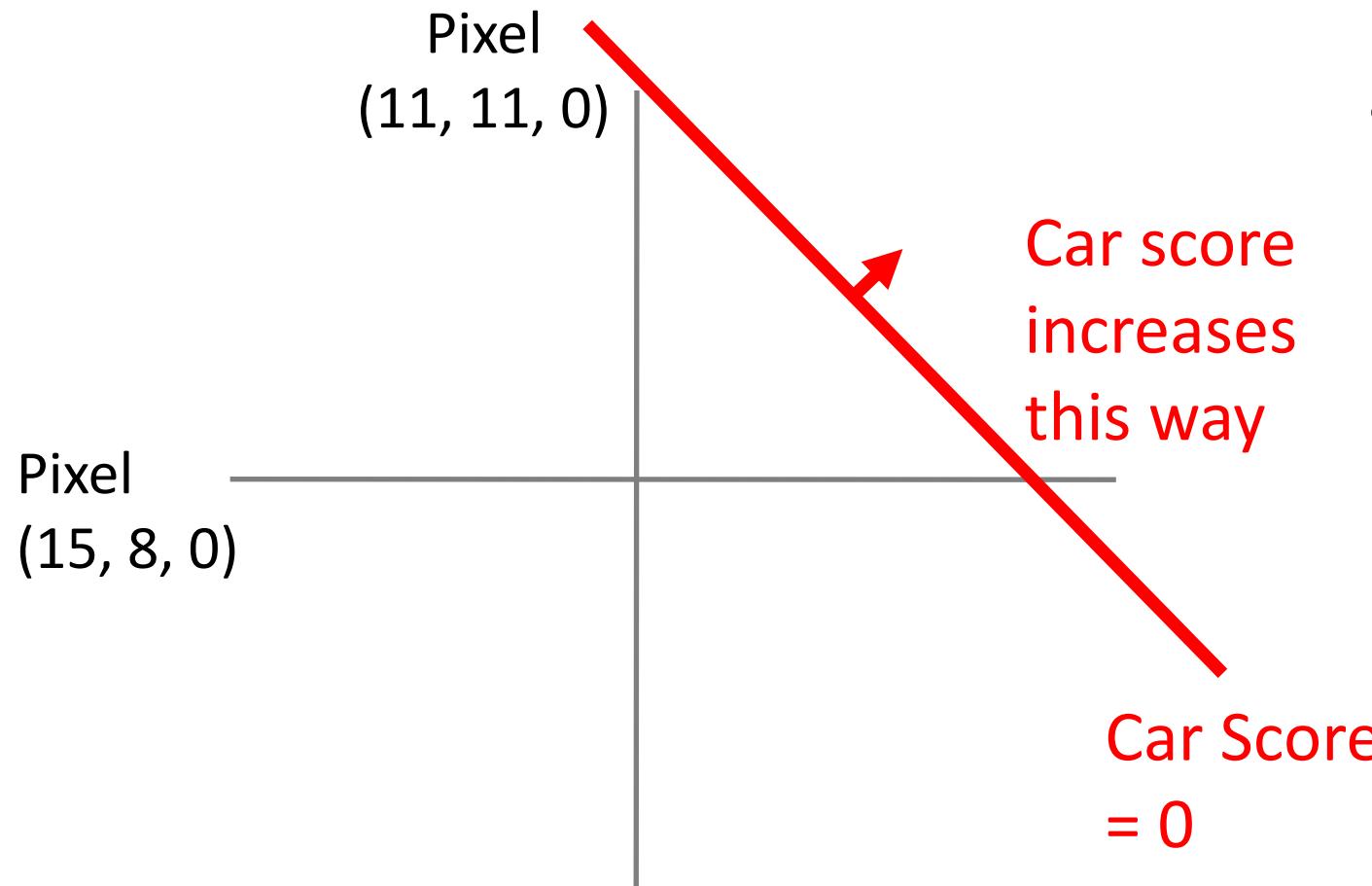


$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

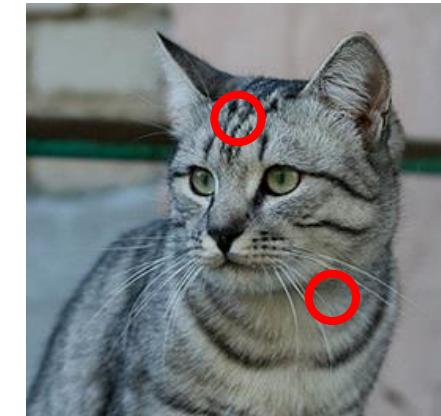


Array of **32x32x3** numbers
(3072 numbers total)

Interpreting a Linear Classifier: Geometric Viewpoint

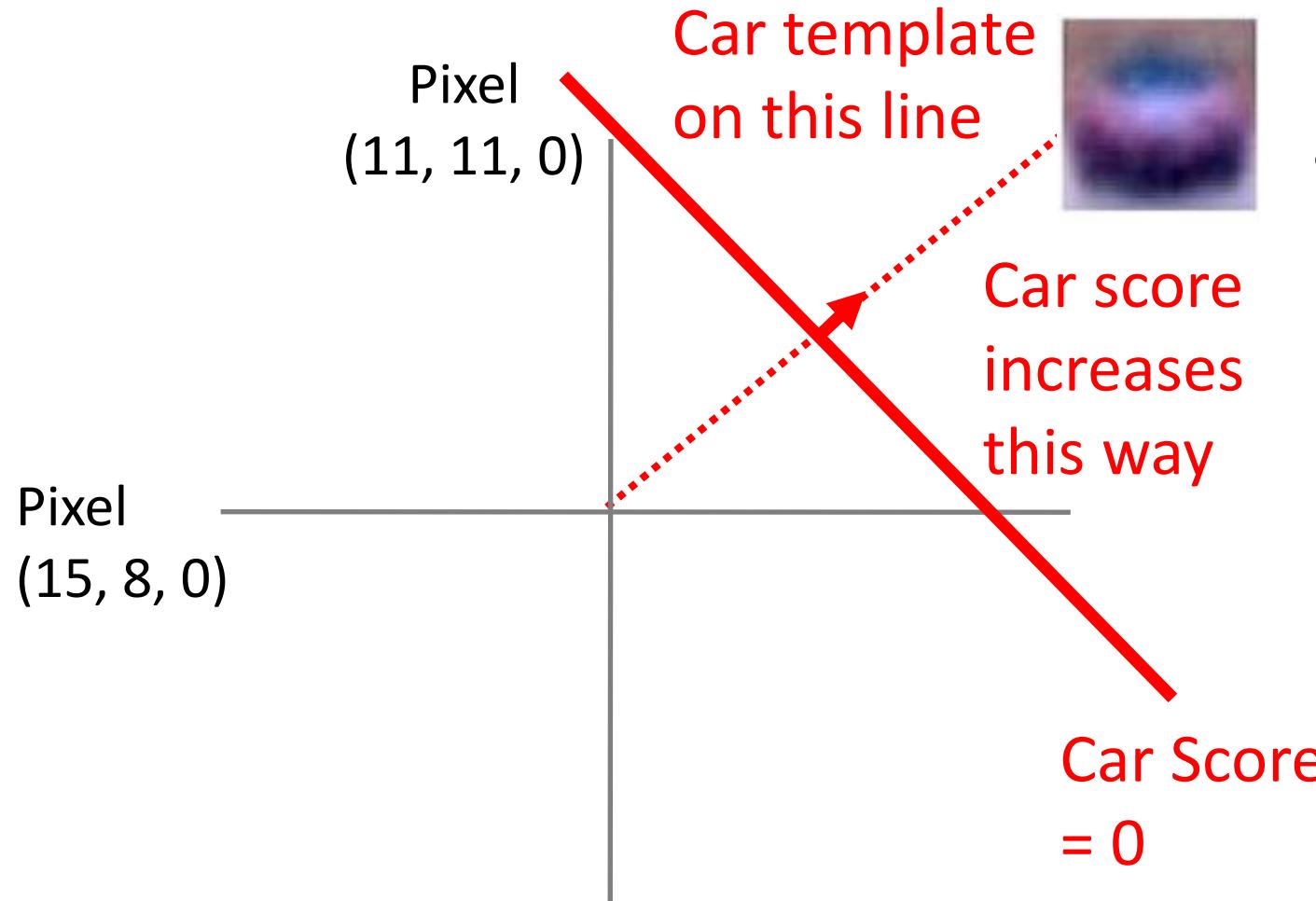


$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



Array of **32x32x3** numbers
(3072 numbers total)

Interpreting a Linear Classifier: Geometric Viewpoint

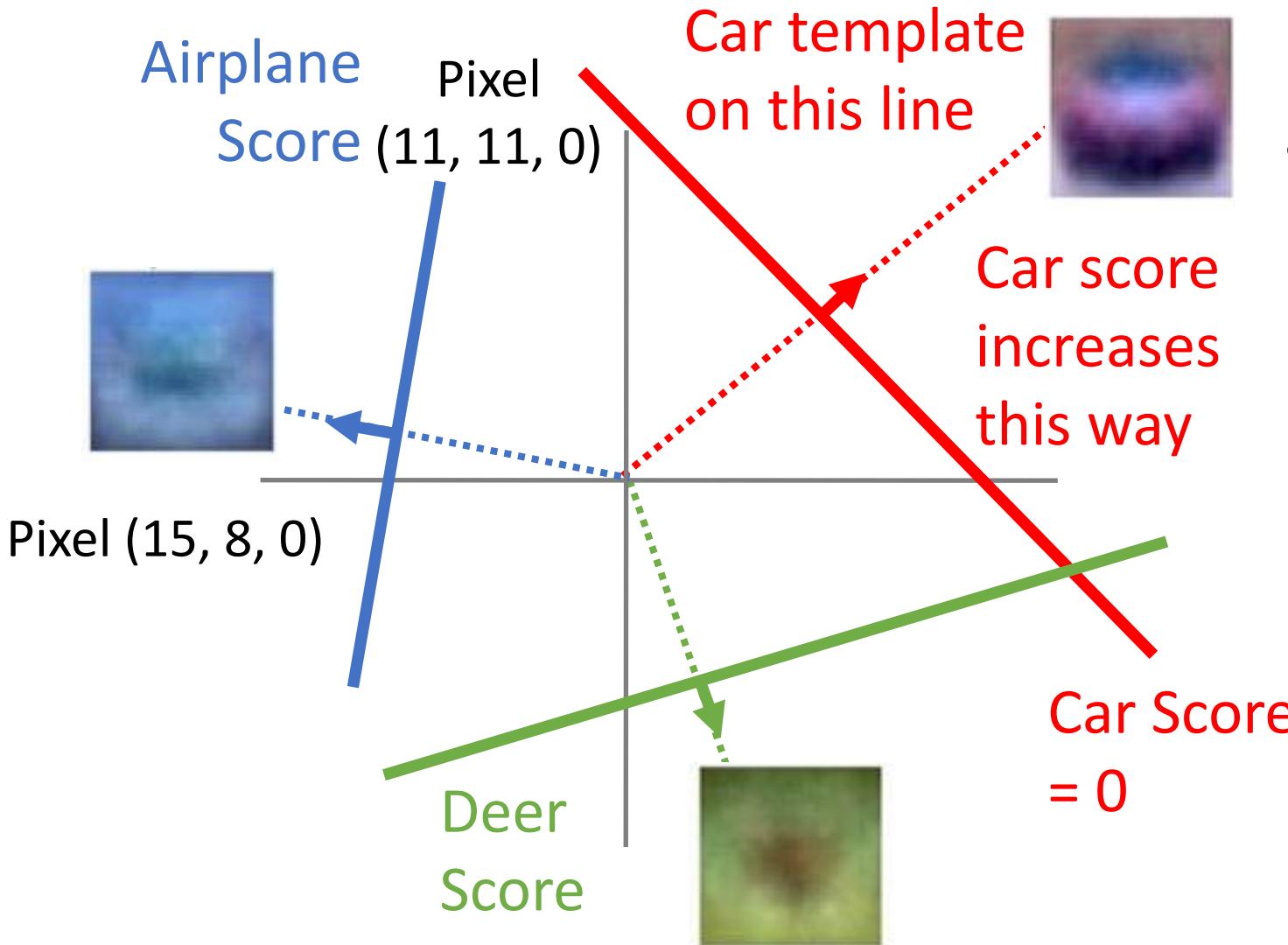


$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$



Array of **32x32x3** numbers
(3072 numbers total)

Interpreting a Linear Classifier: Geometric Viewpoint

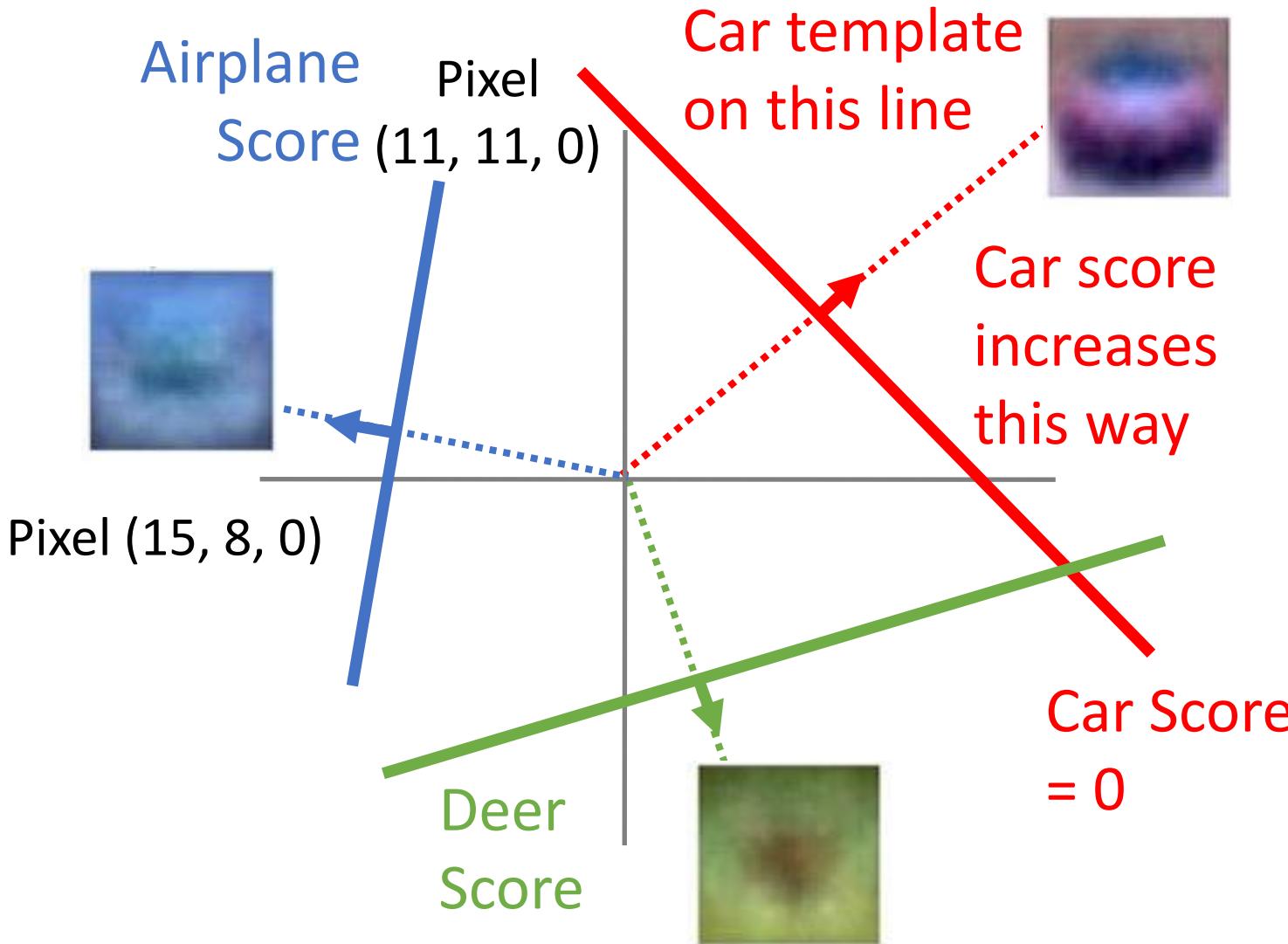


$$f(\mathbf{x}, \mathbf{W}) = \mathbf{W}\mathbf{x} + \mathbf{b}$$

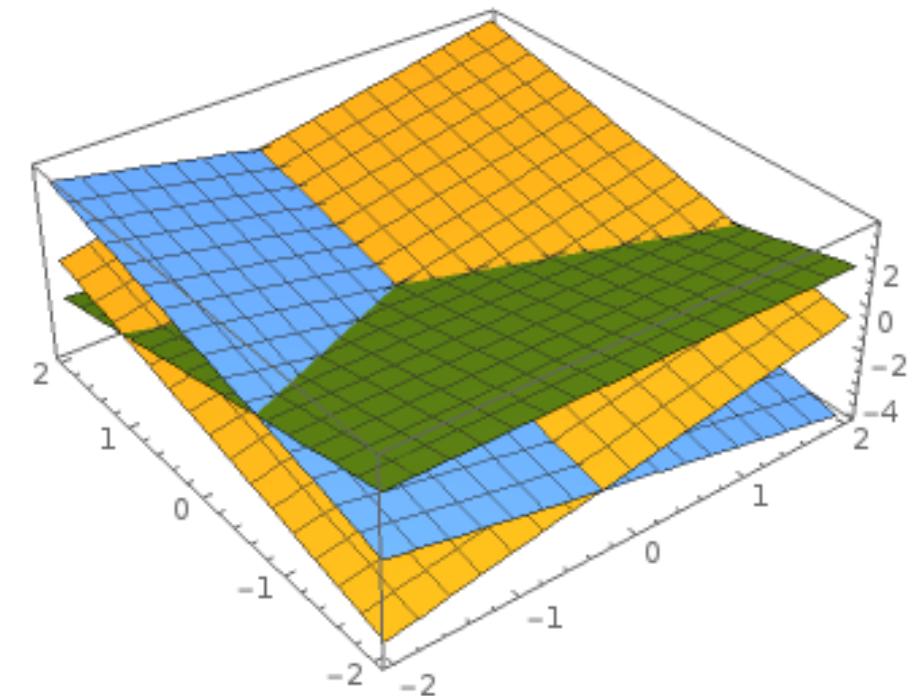


Array of **32x32x3** numbers
(3072 numbers total)

Interpreting a Linear Classifier: Geometric Viewpoint



Hyperplanes carving up a high-dimensional space



Plot created using [Wolfram Cloud](#)

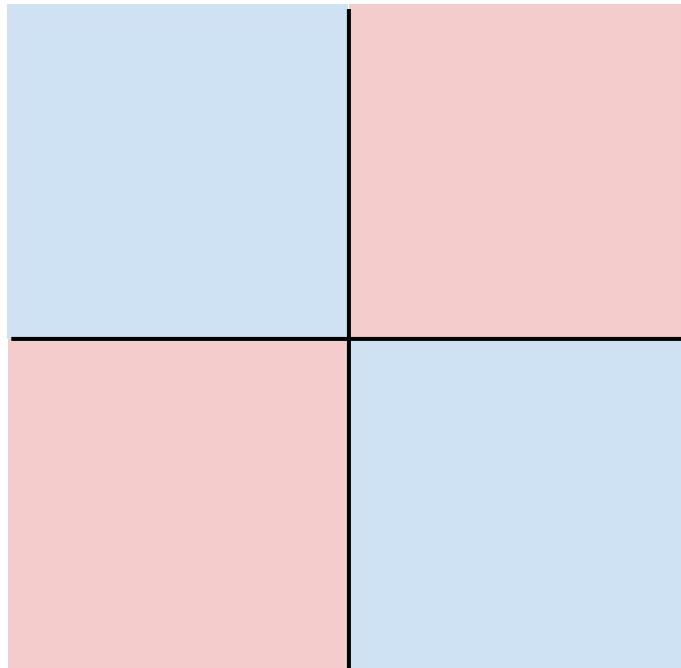
Hard Cases for a Linear Classifier

Class 1:

First and third quadrants

Class 2:

Second and fourth quadrants

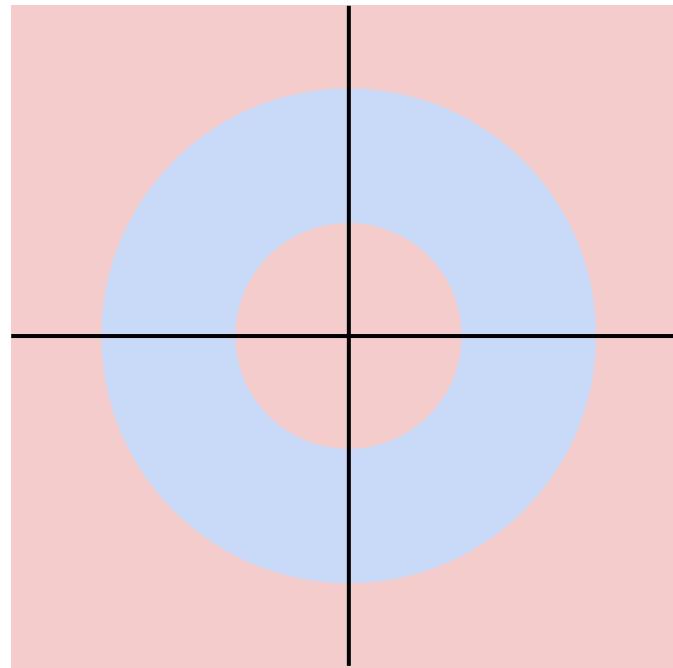


Class 1:

$1 \leq \text{L2 norm} \leq 2$

Class 2:

Everything else

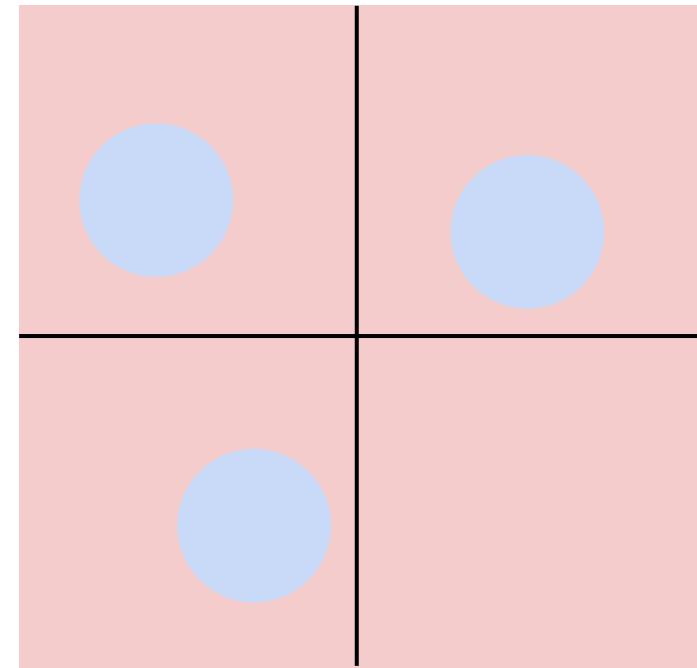


Class 1:

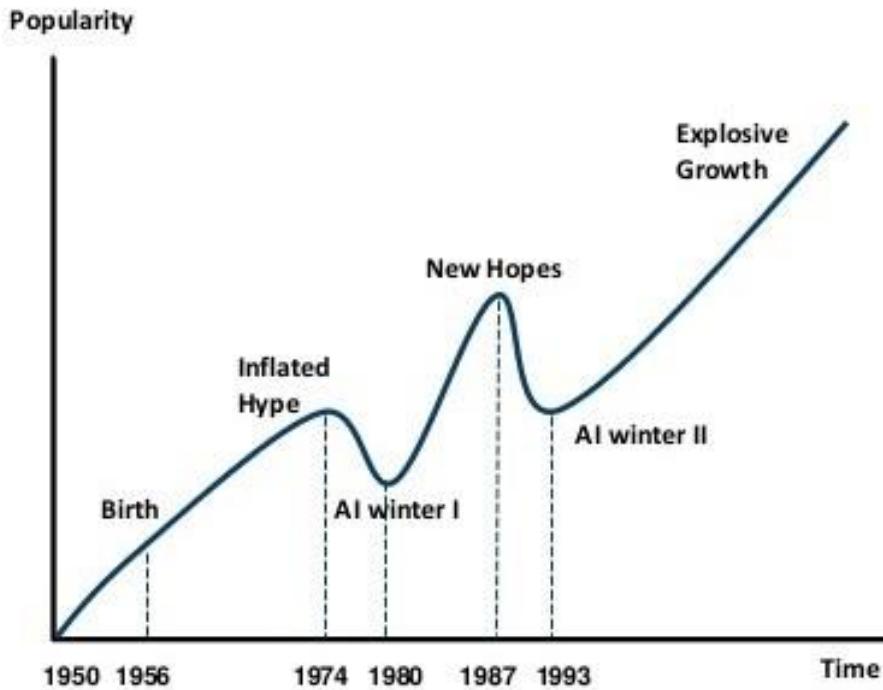
Three modes

Class 2:

Everything else

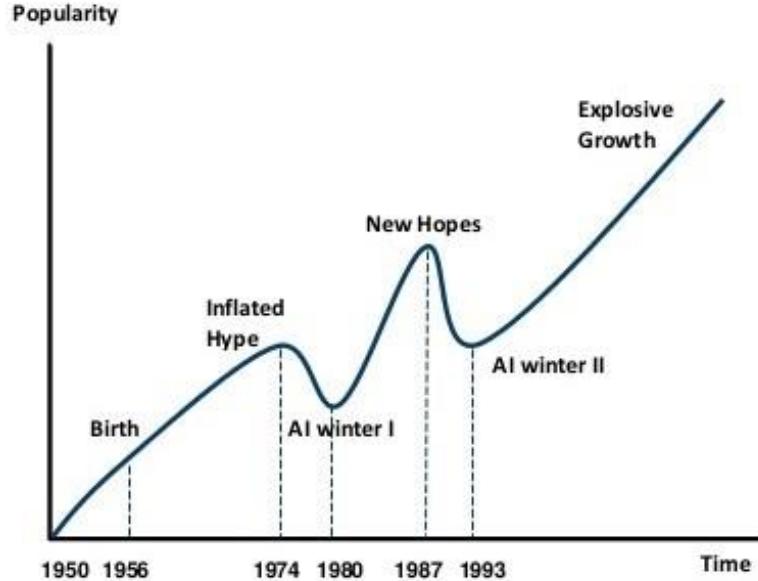


AI HAS A LONG HISTORY OF BEING “THE NEXT BIG THING”...



Timeline of AI Development	
▪ 1950s-1960s:	First AI boom - the age of reasoning, prototype AI developed
▪ 1970s:	AI winter I
▪ 1980s-1990s:	Second AI boom: the age of Knowledge representation (appearance of expert systems capable of reproducing human decision-making)
▪ 1990s:	AI winter II
▪ 1997:	Deep Blue beats Gary Kasparov
▪ 2006:	University of Toronto develops Deep Learning
▪ 2011:	IBM's Watson won Jeopardy
▪ 2016:	Go software based on Deep Learning beats world's champions

- In 1958, Frank Rosenblatt created the perceptron learning algorithm, the simplest type of neural network with only one layer of neurons connecting inputs to outputs.
- The New York Times sensationalized reported the perceptron to be *“the embryo of an electronic computer that the Navy expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”*
- However, it was proven that the single layer perceptron could only recognise linearly separable patterns, but not more complex types such as the XOR (exclusive OR) function.^[i]
- The field of neural networks stagnated for a decade, until it was realised that multi-layer perceptrons, with sufficient computing power and data, were a very effective way of modelling more complex non-linear functions.
- This is the principle behind current research in deep learning with many complex layers of neurons, such as those used by Google DeepMind's AlphaGo to defeat Lee Sedol at the game of Go



Timeline of AI Development	
▪ 1950s-1960s:	First AI boom - the age of reasoning, prototype AI developed
▪ 1970s:	AI winter I
▪ 1980s-1990s:	Second AI boom: the age of Knowledge representation (appearance of expert systems capable of reproducing human decision-making)
▪ 1990s:	AI winter II
▪ 1997:	Deep Blue beats Gary Kasparov
▪ 2006:	University of Toronto develops Deep Learning
▪ 2011:	IBM's Watson won Jeopardy
▪ 2016:	Go software based on Deep Learning beats world's champions

The hard problems are easy, and the easy problems are hard
Moravec’s Paradox

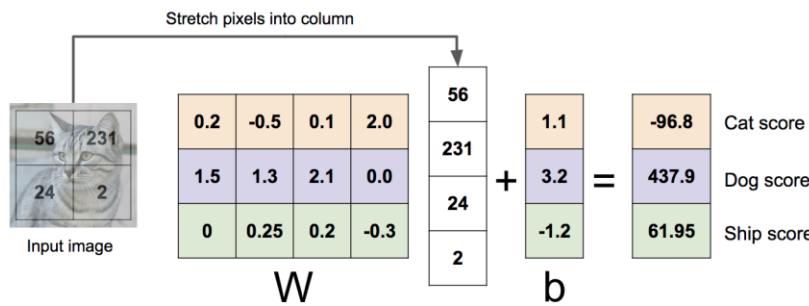
Another fundamental limitation, known as Moravec’s Paradox, states that

- *“It is comparatively easy to make computers exhibit adult level performance on intelligence tests, playing checkers or calculating pi to a billion digits,*
- *but difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility... The mental abilities of a child that we take for granted –*
 - *recognizing a face,*
 - *lifting a pencil, or*
 - *walking across a room –*
- *in fact solve some of the hardest engineering problems ever conceived... Encoded in the large, highly evolved sensory and motor portions of the human brain is a billion years of experience about the nature of the world and how to survive in it.”* [v]
- In short, the hard problems are easy, and the easy problems are hard. This explains why research into computer vision and robotics had made so little progress by the 1970s.

Linear Classifier: Three Viewpoints

Algebraic Viewpoint

$$f(x, W) = Wx$$



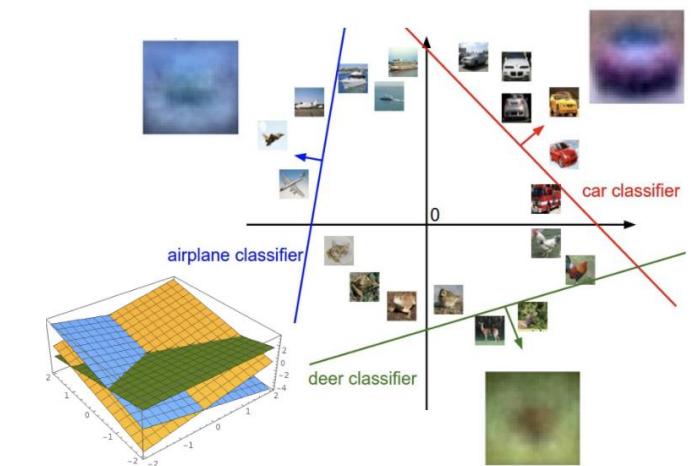
Visual Viewpoint

One template per class



Geometric Viewpoint

Hyperplanes cutting up space



So Far: Defined a linear score function

$$f(x, W) = Wx + b$$



airplane	-3.45	-0.51	3.42
automobile	-8.87	6.04	4.64
bird	0.09	5.31	2.65
cat	2.9	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	-4.34
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

Given a W , we can compute class scores for an image x .

But how can we actually choose a good W ?

Cat image by Nikita is licensed under CC-BY 2.0; Car image is CC0 1.0 public domain; Frog image is in the public domain

Choosing a good W

$$f(x,W) = Wx + b$$



airplane	-3.45	-0.51	3.42
automobile	-8.87	6.04	4.64
bird	0.09	5.31	2.65
cat	2.9	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	-4.34
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

TODO:

1. Use a **loss function** to quantify how good a value of W is
2. Find a W that minimizes the loss function (**optimization**)

Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**;
cost function)

Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**;
cost function)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**;
cost function)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^N$$

Where x_i is image and
 y_i is (integer) label

Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier
High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^N$$

Where x_i is image and
 y_i is (integer) label

Loss for a single example is

$$L_i(f(x_i, W), y_i)$$

Loss Function

A **loss function** tells how good our current classifier is

Low loss = good classifier

High loss = bad classifier

(Also called: **objective function**; **cost function**)

Negative loss function sometimes called **reward function**, **profit function**, **utility function**, **fitness function**, etc

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^N$$

Where x_i is image and
 y_i is (integer) label

Loss for a single example is

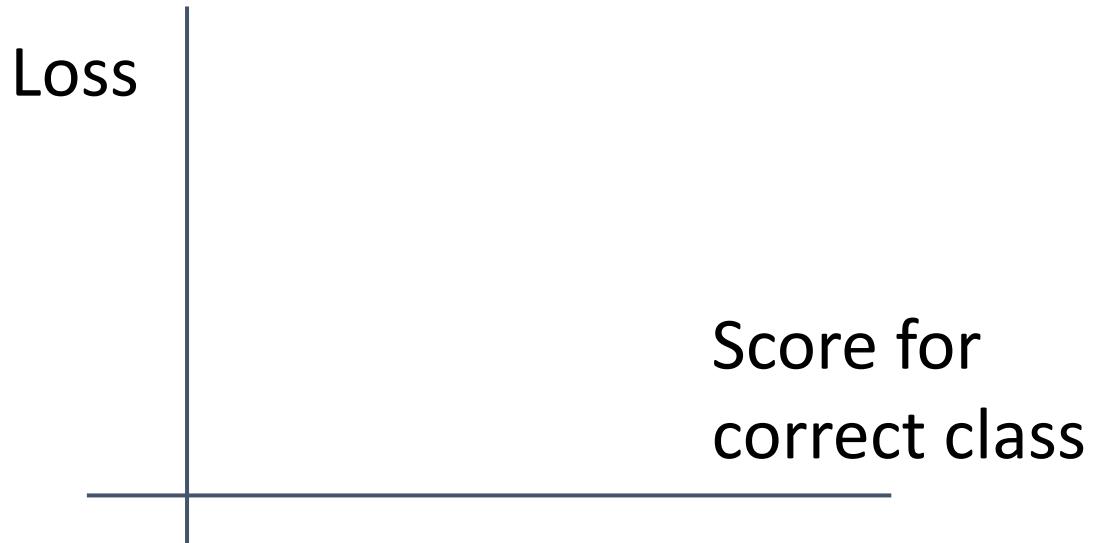
$$L_i(f(x_i, W), y_i)$$

Loss for the dataset is average of per-example losses:

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

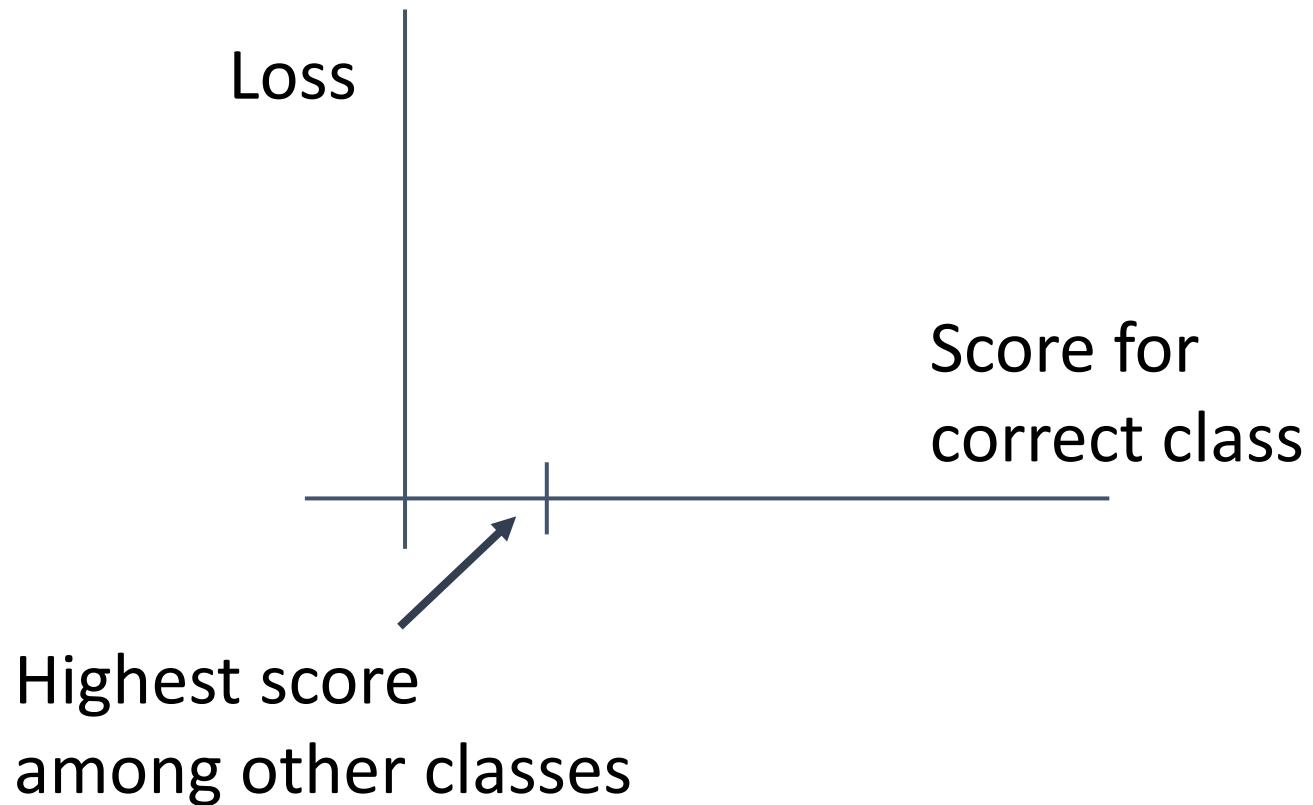
Multiclass SVM Loss

“The score of the correct class should be higher than all the other scores”



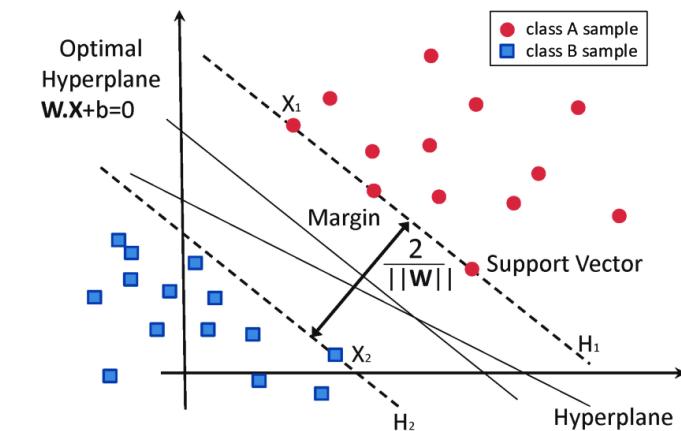
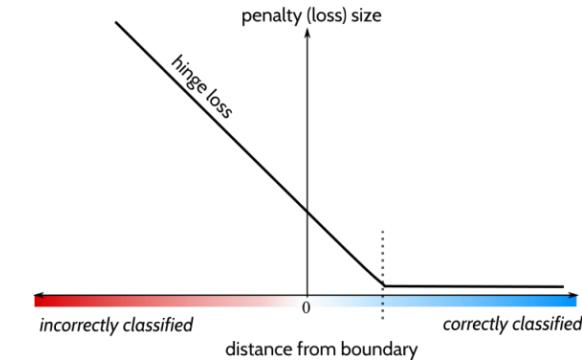
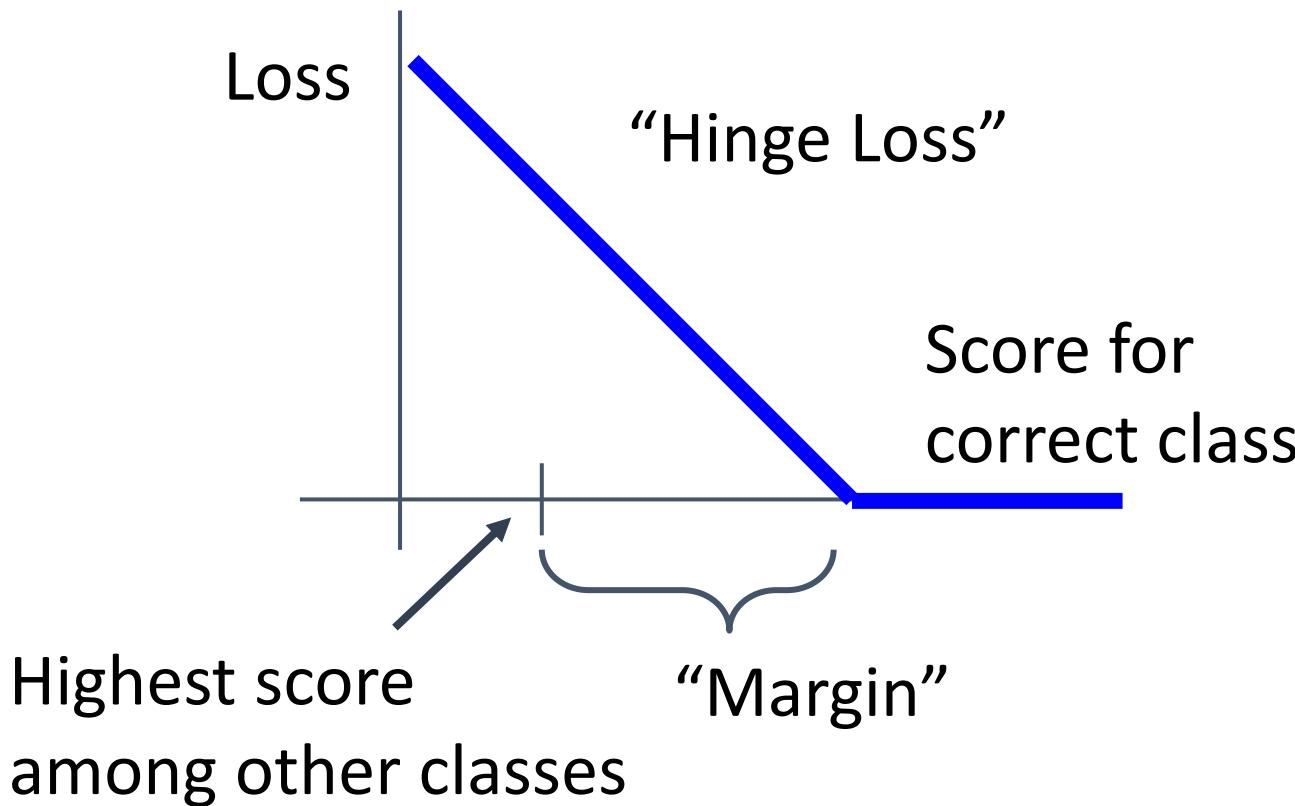
Multiclass SVM Loss

"The score of the correct class should be higher than all the other scores"



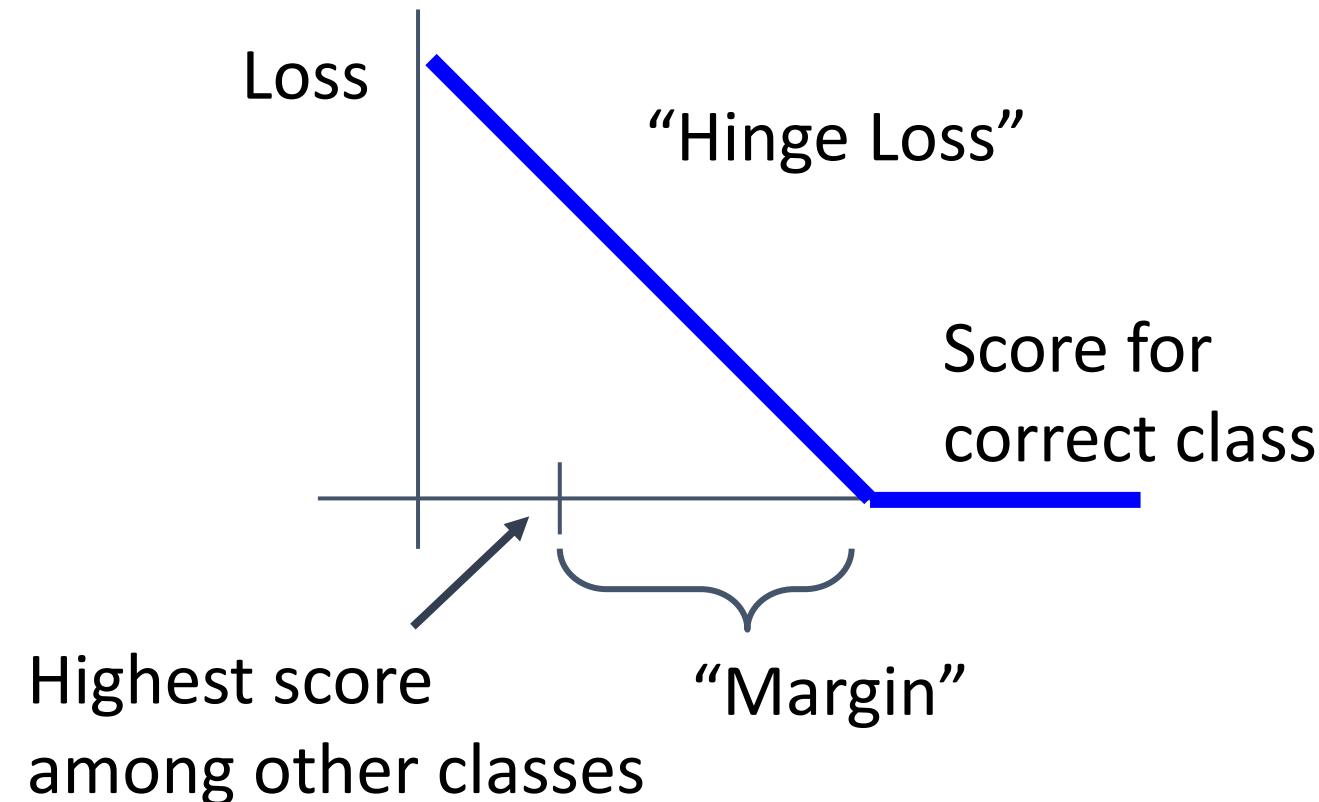
Multiclass SVM Loss

"The score of the correct class should be higher than all the other scores"



Multiclass SVM Loss

"The score of the correct class should be higher than all the other scores"

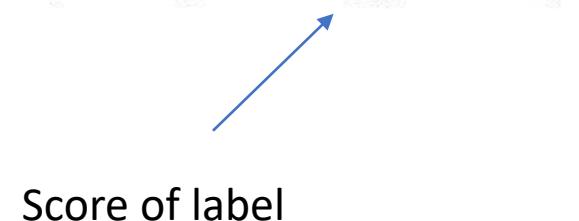


Given an example (x_i, y_i)
(x_i is image, y_i is label)

Let $s = f(x_i, W)$ be scores

Then the SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



Regularization: Beyond Training Error

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)$$

Data loss: Model predictions
should match training data

Regularization: Beyond Training Error

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \lambda R(W)$$

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too well* on training data

Regularization: Beyond Training Error

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i) + \lambda R(W)$$

λ = regularization strength
(hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too well* on training data

Regularization: Beyond Training Error

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i) + \lambda R(W)$$

λ = regularization strength
(hyperparameter)

Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too well* on training data

Simple examples

L2 regularization: $R(W) = \sum_k \sum_l W_{k,l}^2$

L1 regularization: $R(W) = \sum_k \sum_l |W_{k,l}|$

Elastic net (L1 + L2): $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

More complex:

Dropout

Batch normalization

Cutout, Mixup, Stochastic depth, etc...

Regularization: Beyond Training Error

$$L(W) = \frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i) + \lambda R(W)$$

λ = regularization strength
(hyperparameter)

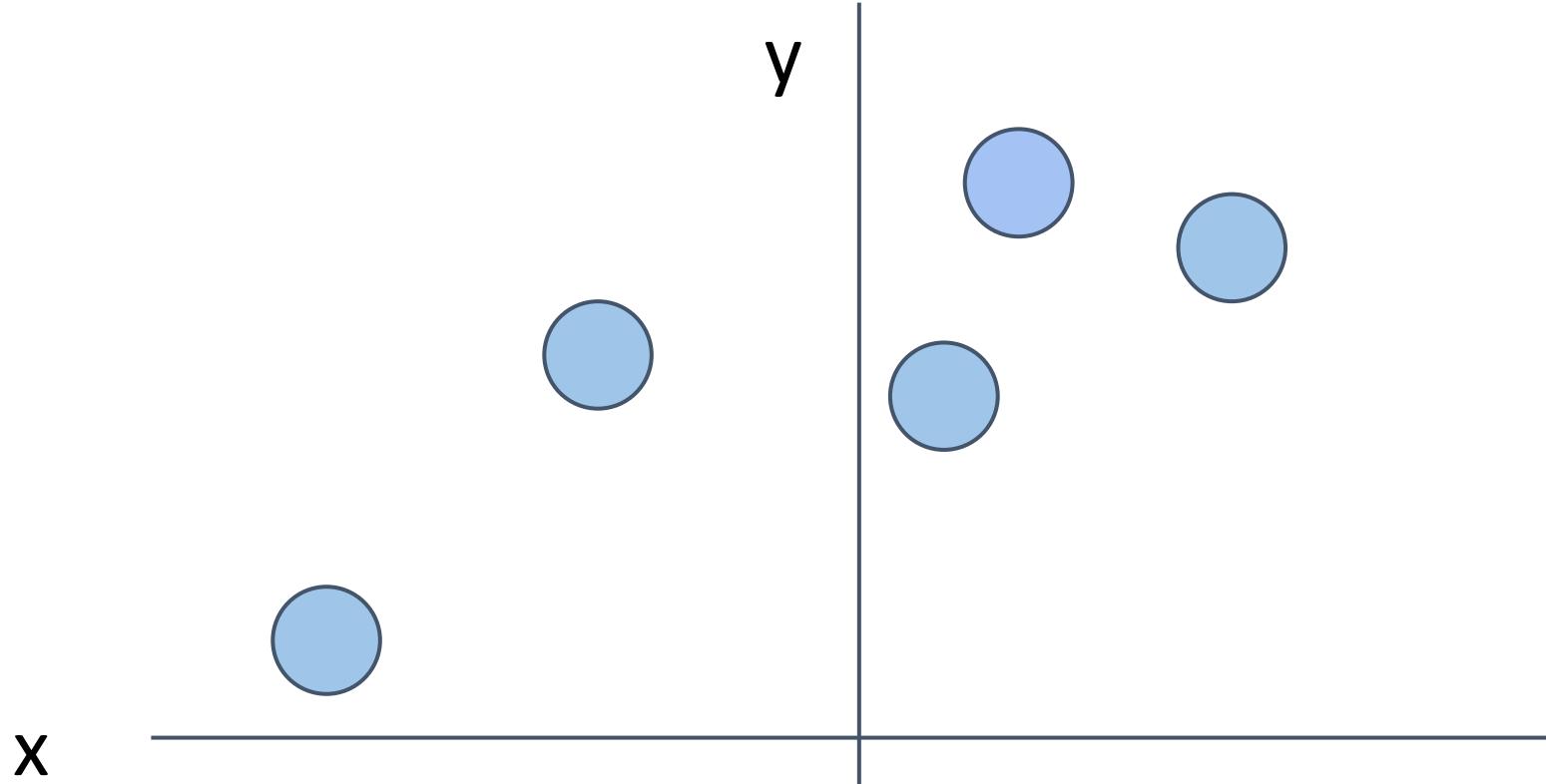
Data loss: Model predictions should match training data

Regularization: Prevent the model from doing *too well* on training data

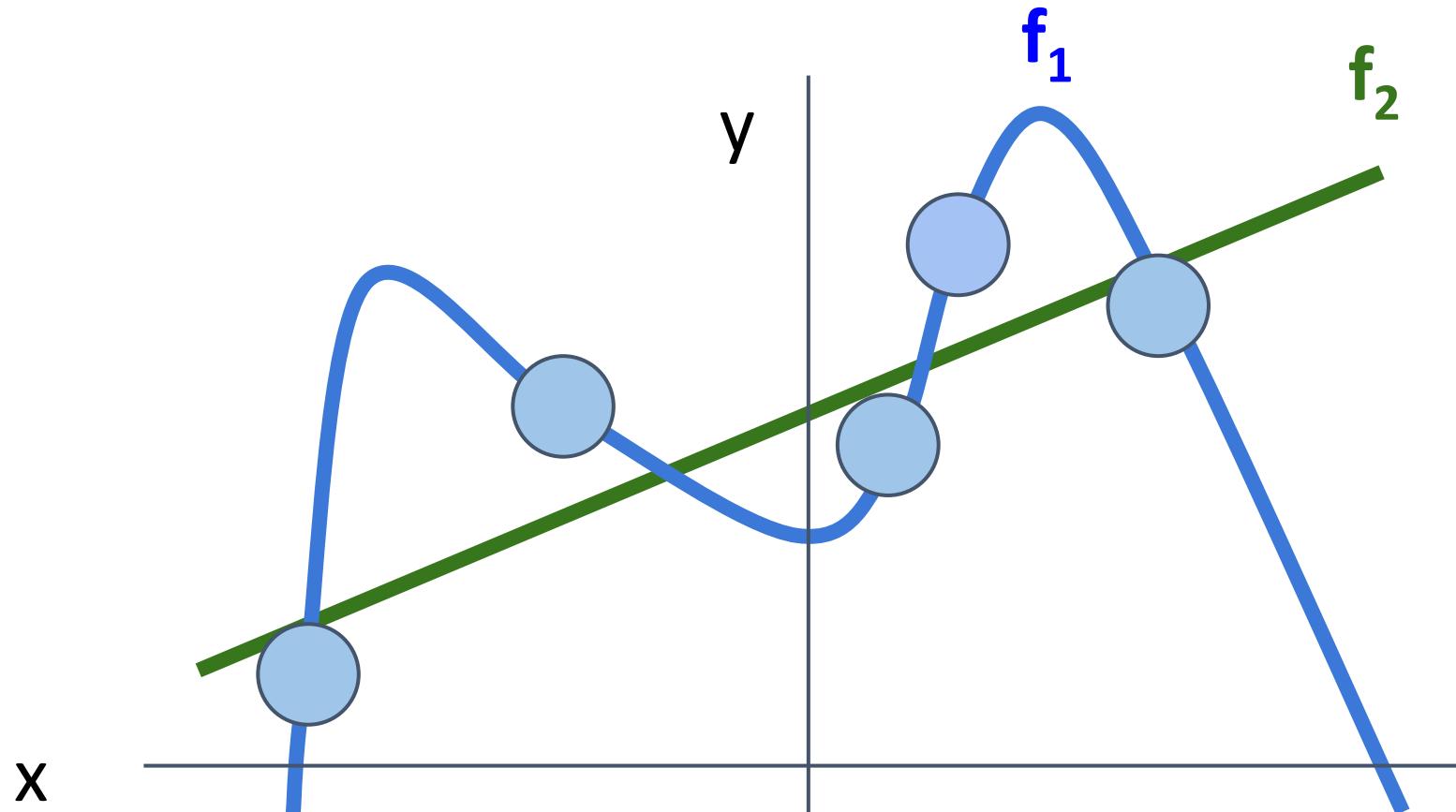
Purpose of Regularization:

- Express preferences in among models beyond "minimize training error"
- Avoid **overfitting**: Prefer simple models that generalize better
- Improve optimization by adding curvature

Regularization: Prefer Simpler Models

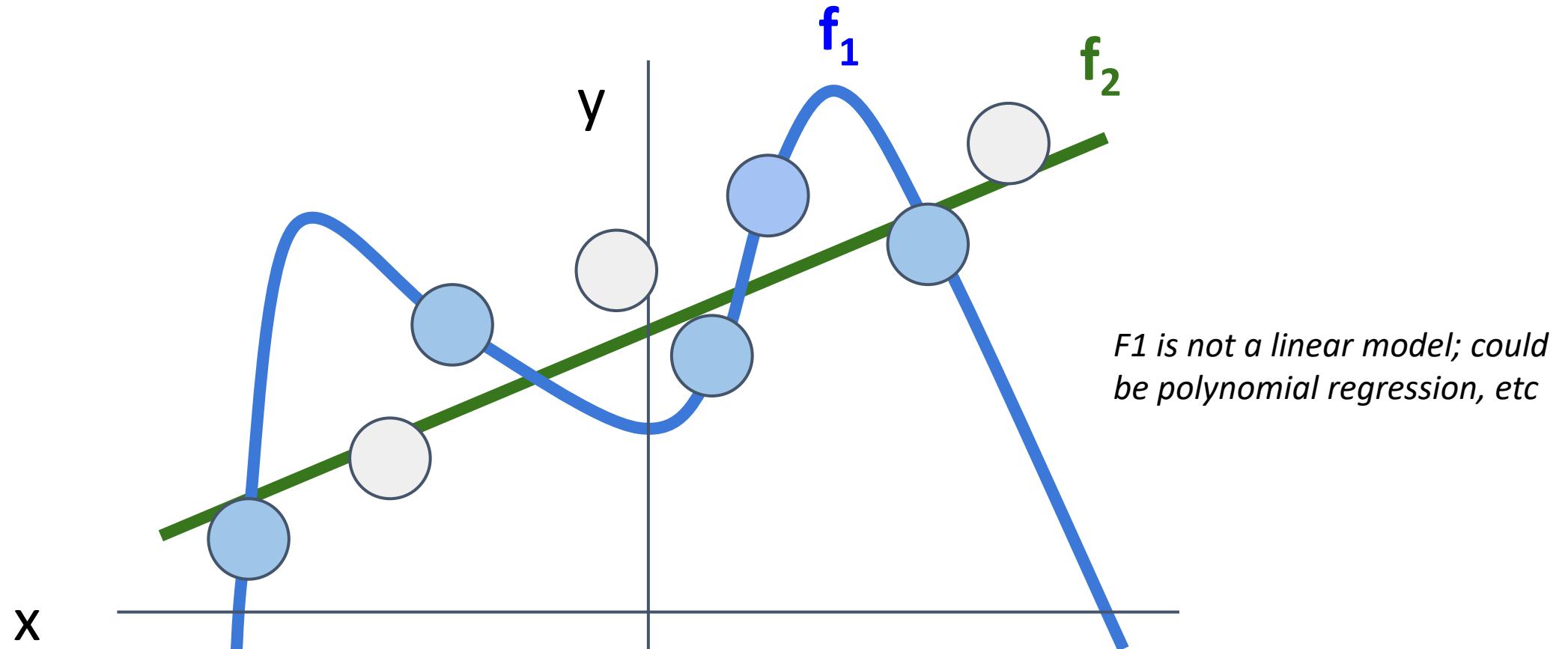


Regularization: Prefer Simpler Models



The model f_1 fits the training data perfectly
The model f_2 has training error, but is simpler

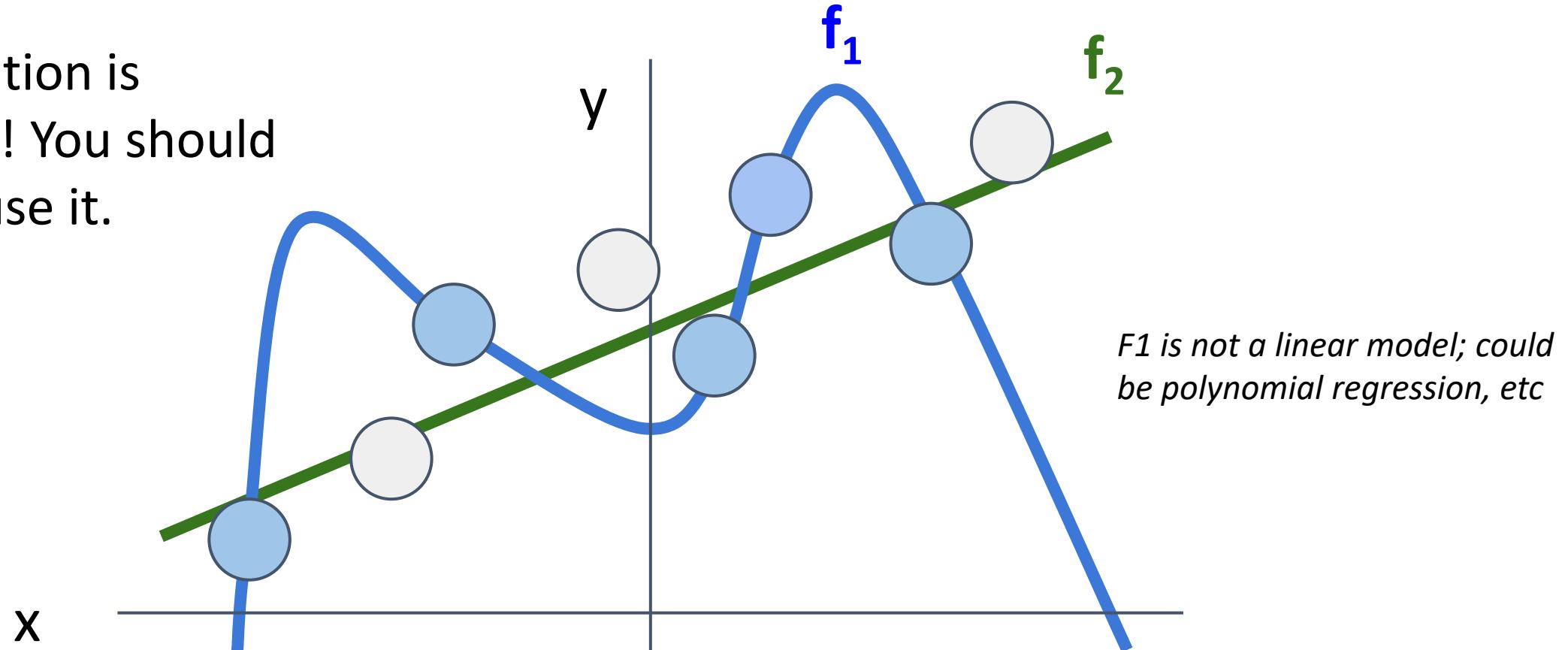
Regularization: Prefer Simpler Models



Regularization pushes against fitting the data
too well so we don't fit noise in the data

Regularization: Prefer Simpler Models

Regularization is important! You should (usually) use it.



Regularization pushes against fitting the data *too well* so we don't fit noise in the data

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



cat **3.2**

car 5.1

frog -1.7

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

cat **3.2**

car 5.1

frog -1.7

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



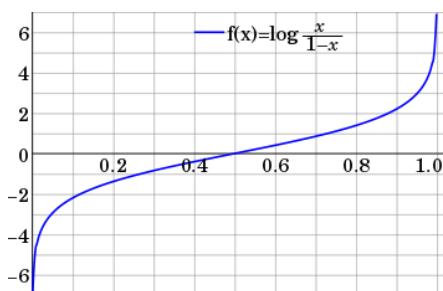
cat	3.2
car	5.1
frog	-1.7

Unnormalized log-probabilities / logits

$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax function



Logit Functions and Machine Learning

The Logit function is used similarly to the sigmoid function in [neural networks](#). The sigmoid, or [activation, function](#) produces a probability, whereas the Logit function takes a probability and produces a real number between negative and positive infinity. Like the sigmoid function, Logit functions are often placed as the last layer in a neural network as can simplify the data. For example, a Logit function is often used in the final layer of a neural network used in classification tasks. As the network determines probabilities for classification, the Logit function can transform those probabilities to real numbers.

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**

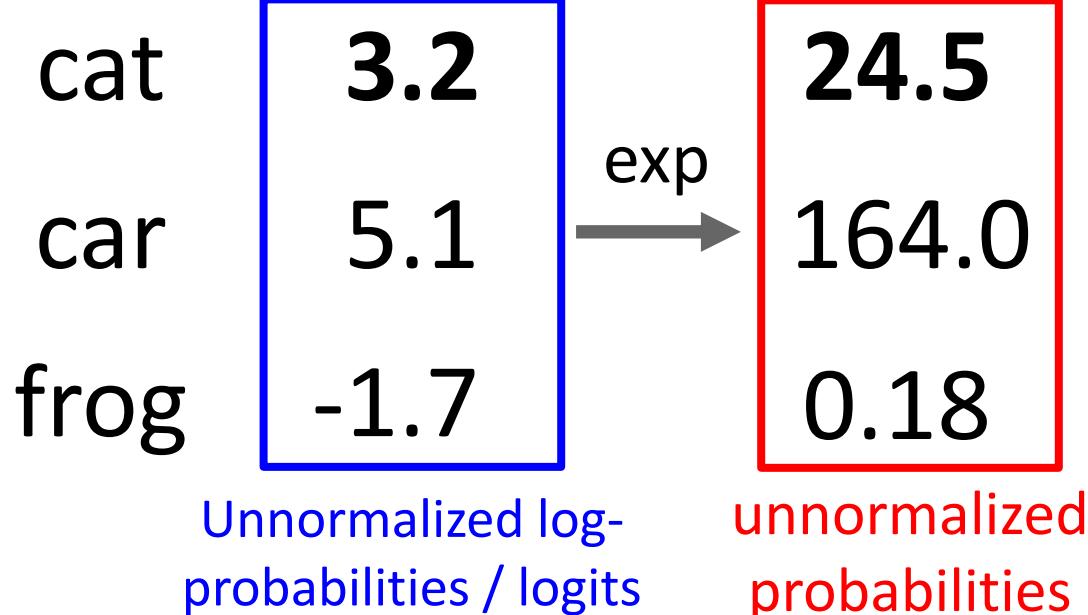


$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

Probabilities
must be ≥ 0



Cross-Entropy Loss (Multinomial Logistic Regression)

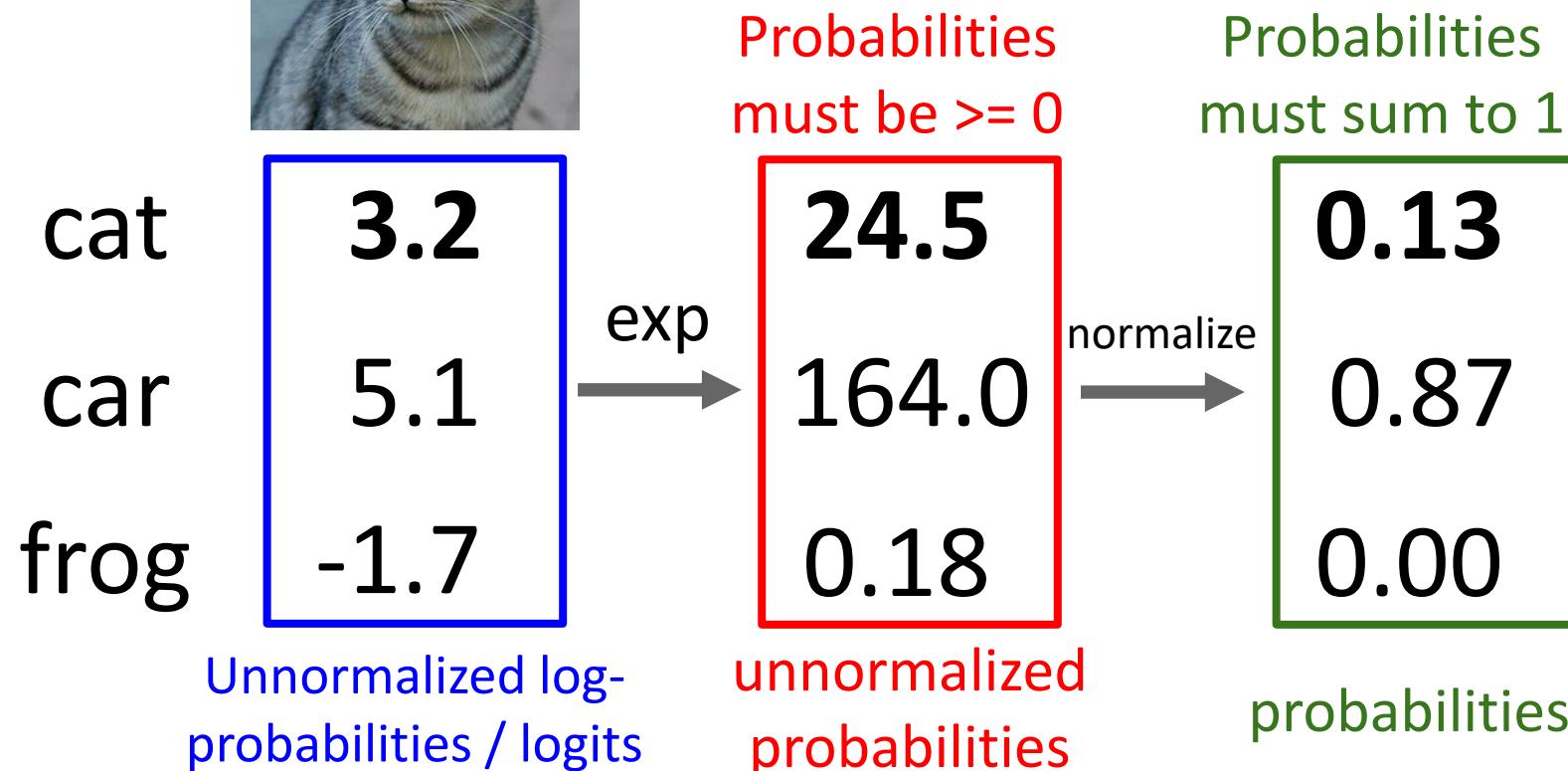
Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function



Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

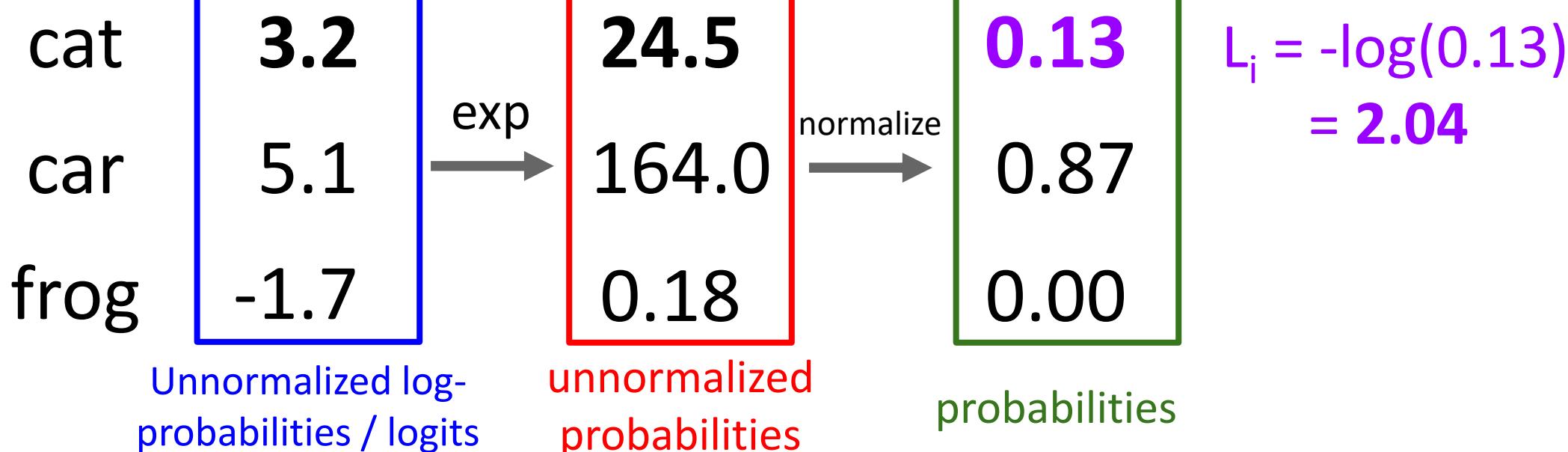
$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

Probabilities
must be ≥ 0

Probabilities
must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$



Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

Probabilities
must be ≥ 0

Probabilities
must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

cat
car
frog

3.2
5.1
-1.7

Unnormalized log-
probabilities / logits

\exp

24.5
164.0
0.18

unnormalized
probabilities

normalize

0.13
0.87
0.00

probabilities

$$L_i = -\log(0.13) \\ = 2.04$$

Maximum Likelihood Estimation
Choose weights to maximize the
likelihood of the observed data
(next lecture)

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

Probabilities
must be ≥ 0

Probabilities
must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

cat
car
frog

3.2
5.1
-1.7

Unnormalized log-
probabilities / logits

\exp

24.5
164.0
0.18

unnormalized
probabilities

normalize

0.13
0.87
0.00

probabilities

Compare \leftarrow

1.00
0.00
0.00

Correct
probs

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

Probabilities
must be ≥ 0

Probabilities
must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

cat
car
frog

3.2
5.1
-1.7

Unnormalized log-
probabilities / logits

\exp

24.5
164.0
0.18

unnormalized
probabilities

normalize

0.13
0.87
0.00

probabilities

Compare \leftarrow

*Kullback–Leibler
divergence*

$$D_{KL}(P \| Q) = \sum_y P(y) \log \frac{P(y)}{Q(y)}$$

1.00
0.00
0.00

Correct
probs

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

Probabilities
must be ≥ 0

Probabilities
must sum to 1

$$L_i = -\log P(Y = y_i | X = x_i)$$

cat

3.2

\rightarrow
exp

24.5

normalize
 \rightarrow

164.0

0.18

car

5.1

-1.7

Unnormalized log-
probabilities / logits

0.13

0.87

0.00

probabilities

Compare \leftarrow

1.00

Cross Entropy

$$H(P, Q) =$$

$$H(p) + D_{KL}(P \| Q)$$

0.00

0.00

Correct
probs

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

cat

3.2

Maximize probability of correct class

$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

car

5.1

frog -1.7

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

cat

3.2

Maximize probability of correct class

$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

car

5.1

Q: What is the min /
max possible loss L_i ?

frog

-1.7

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

cat

3.2

Maximize probability of correct class

$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

car

5.1

Q: What is the min /
max possible loss L_i ?

A: Min 0, max +infinity

frog

-1.7

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

cat

3.2

Maximize probability of correct class

$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

car

5.1

frog

-1.7

Q: If all scores are
small random values,
what is the loss?

Cross-Entropy Loss (Multinomial Logistic Regression)

Want to interpret raw classifier scores as **probabilities**



$$s = f(x_i; W)$$

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$$

Softmax
function

cat

3.2

Maximize probability of correct class

$$L_i = -\log P(Y = y_i | X = x_i)$$

Putting it all together:

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

car

5.1

Q: If all scores are small random values, what is the loss?

A: $-\log(1/C)$
 $\log(10) \approx 2.3$

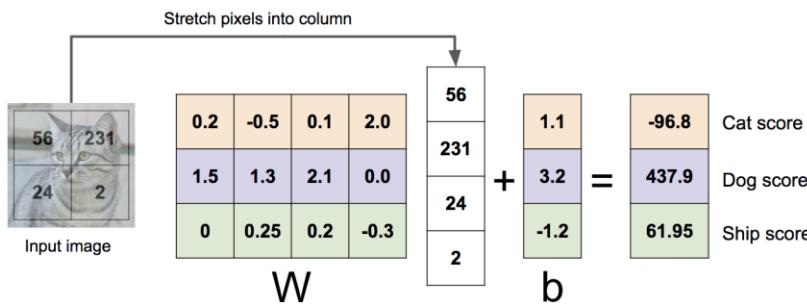
frog

-1.7

Recap: Three ways to think about linear classifiers

Algebraic Viewpoint

$$f(x, W) = Wx$$



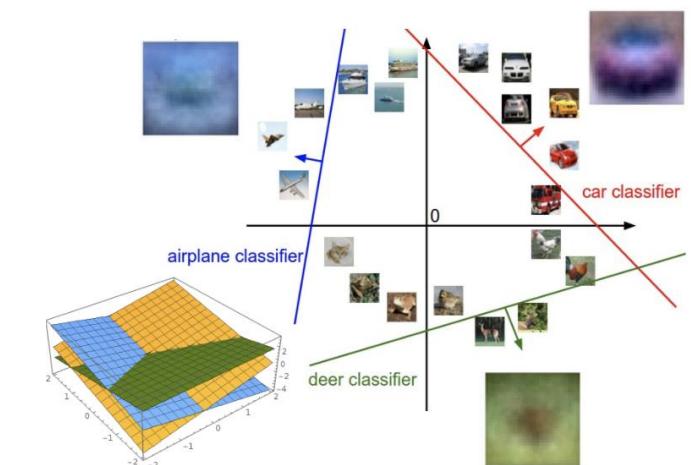
Visual Viewpoint

One template per class



Geometric Viewpoint

Hyperplanes cutting up space



Recap: Loss Functions quantify preferences

- We have some dataset of (x, y)
- We have a **score function**:
- We have a **loss function**:

$$s = f(x; W) = Wx$$

Linear classifier

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right)$$

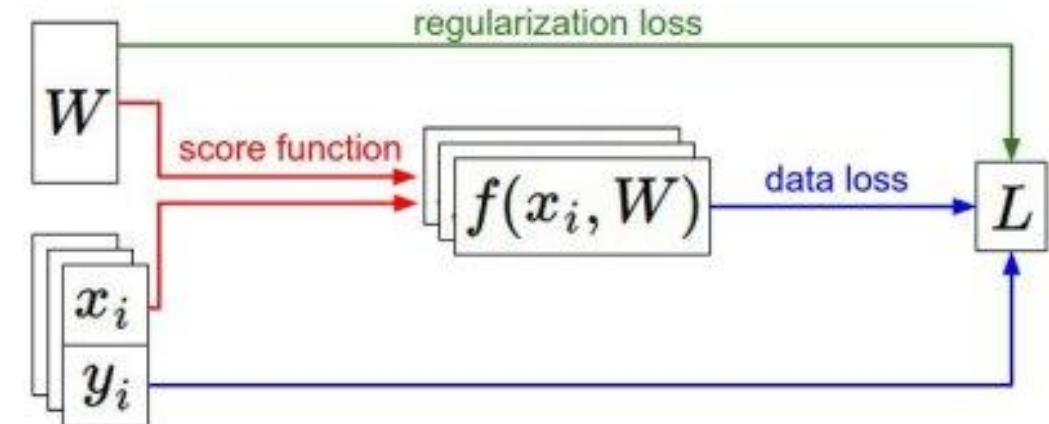
Softmax

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

SVM

$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W)$$

Full loss



Recap: Loss Functions quantify preferences

- We have some dataset of (x, y)
- We have a **score function**:
- We have a **loss function**:

$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{sj}}\right) \text{ Softmax}$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \text{ SVM}$$

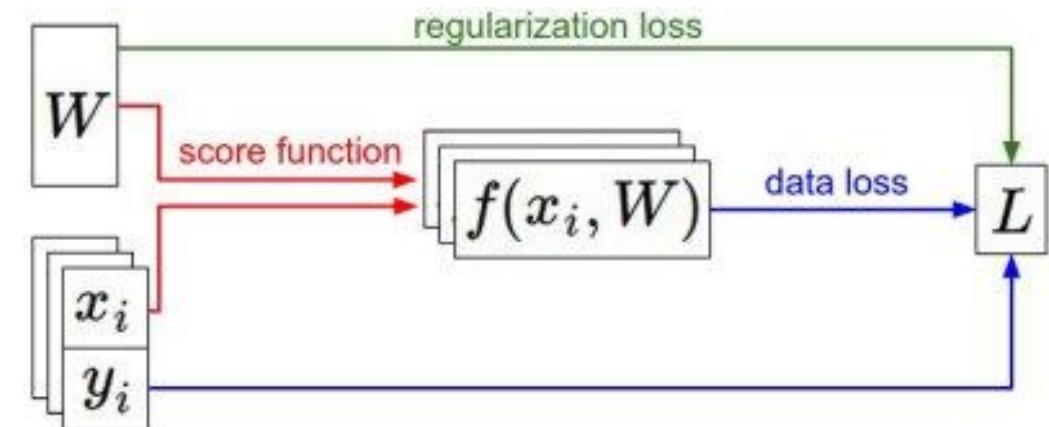
$$L = \frac{1}{N} \sum_{i=1}^N L_i + R(W) \text{ Full loss}$$

Q: How do we find the best W ?

$$s = f(x; W) = Wx$$

Linear classifier

A: Next lecture

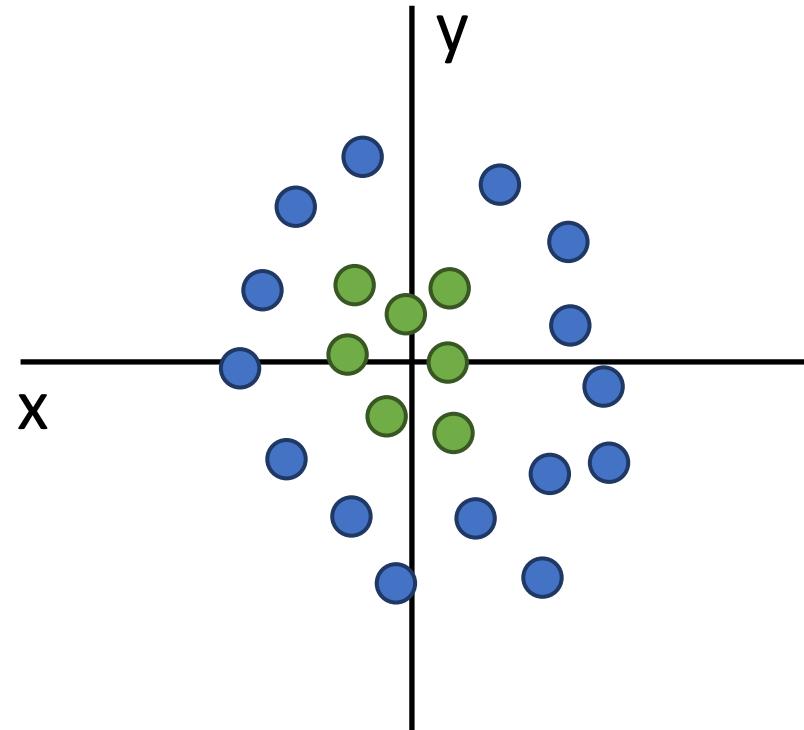




Lecture 5: Neural Networks

Problem: Linear Classifiers aren't that powerful

Geometric Viewpoint

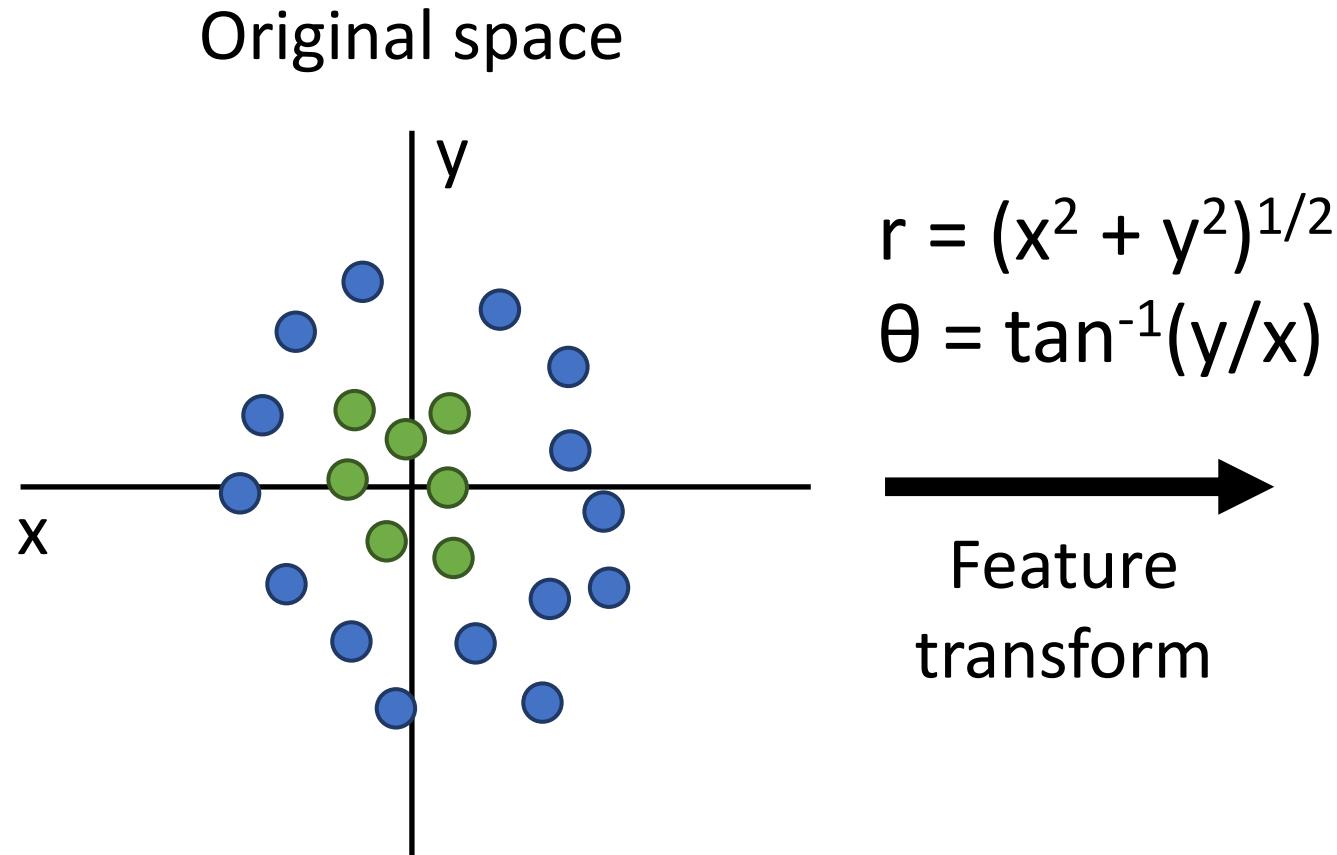


Visual Viewpoint

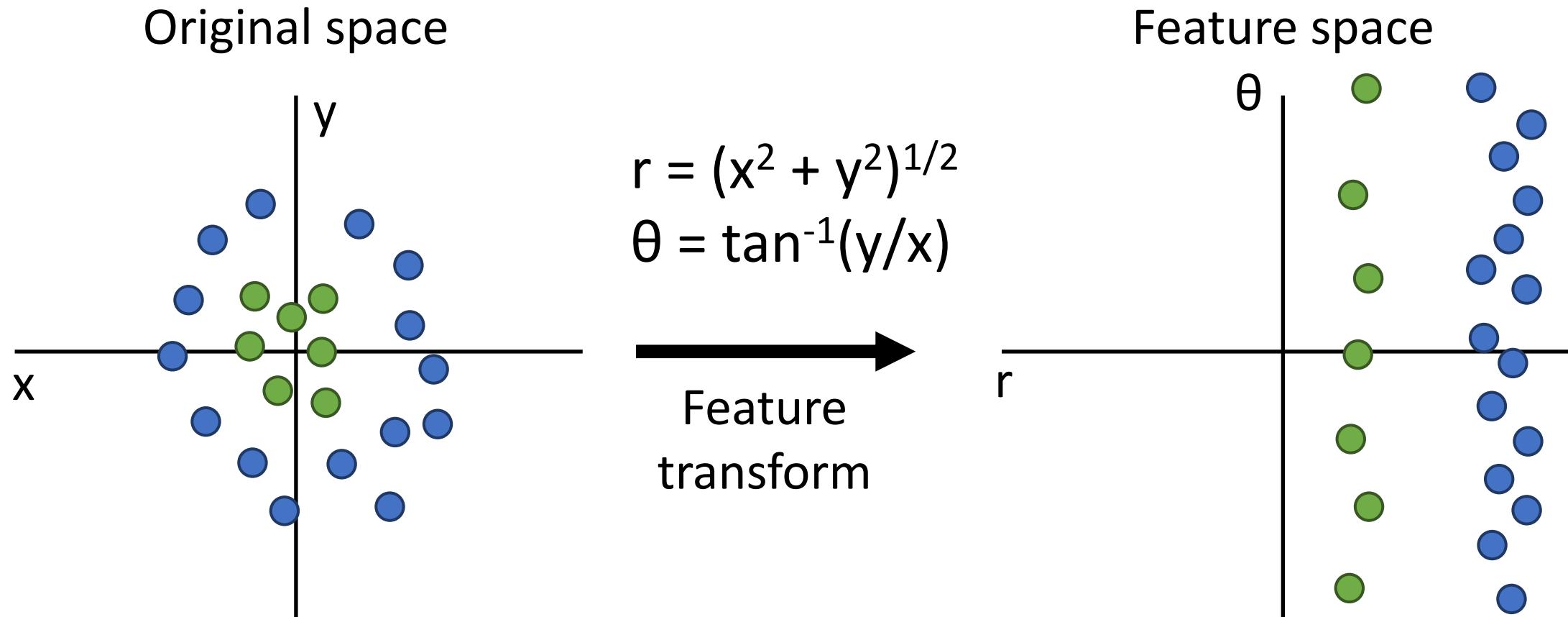
One template per class:
Can't recognize different
modes of a class



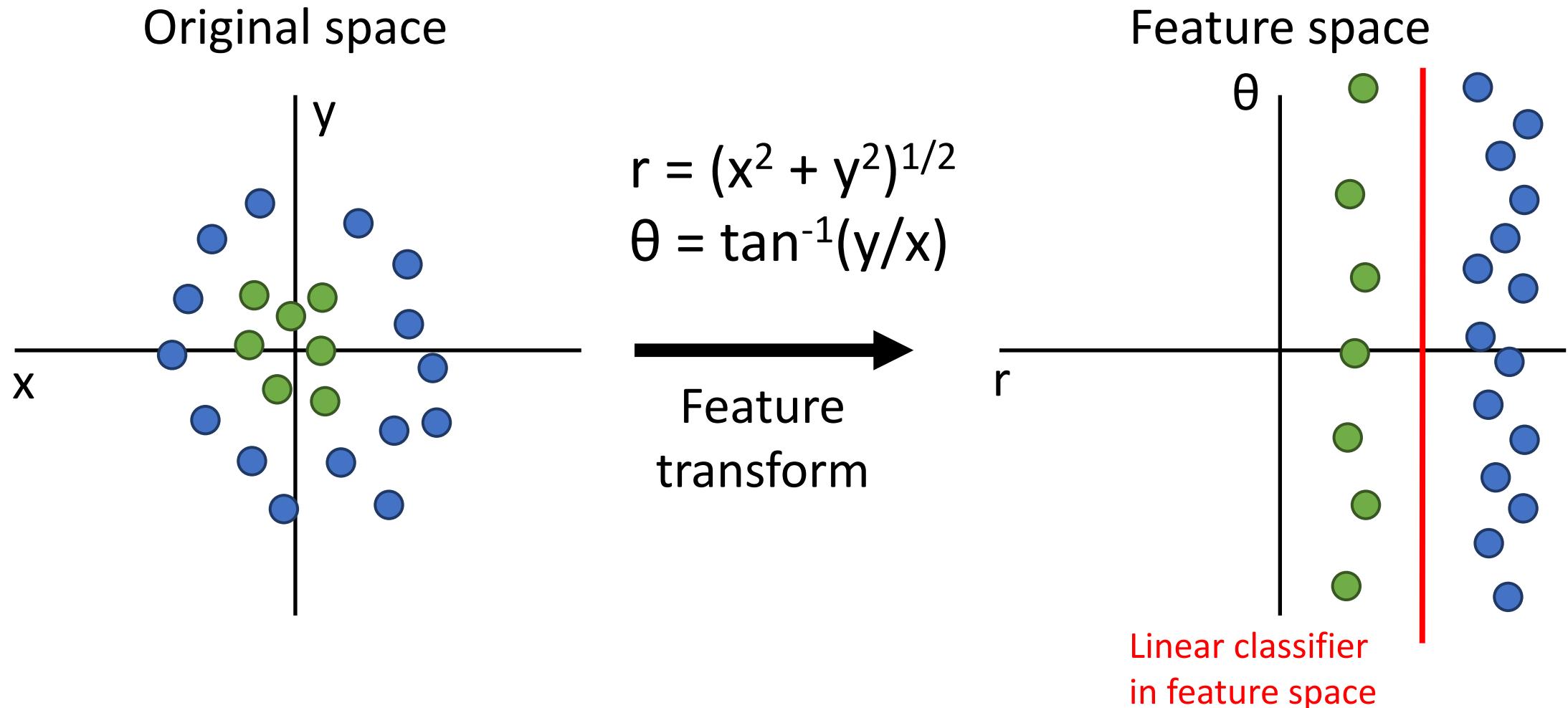
One solution: Feature Transforms



One solution: Feature Transforms



One solution: Feature Transforms



One solution: Feature Transforms

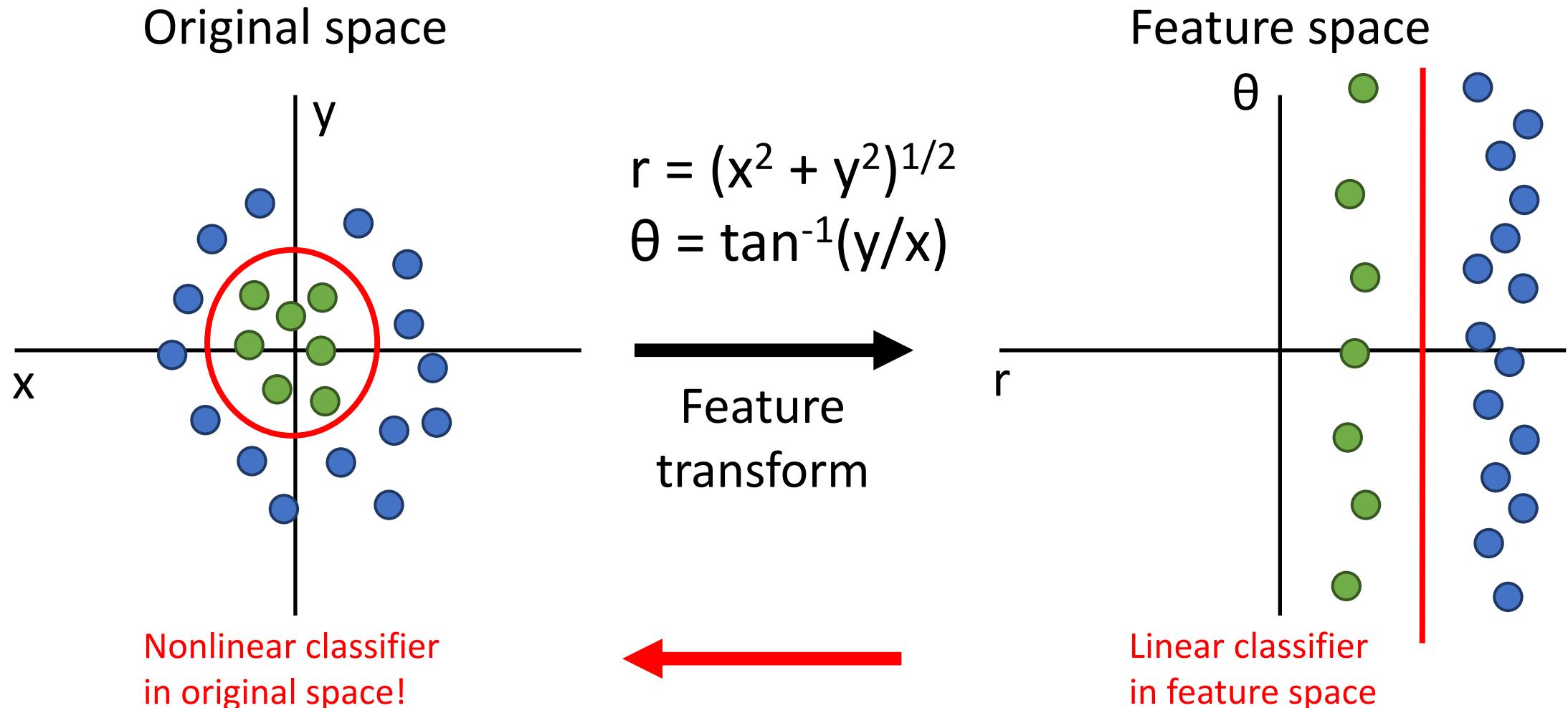
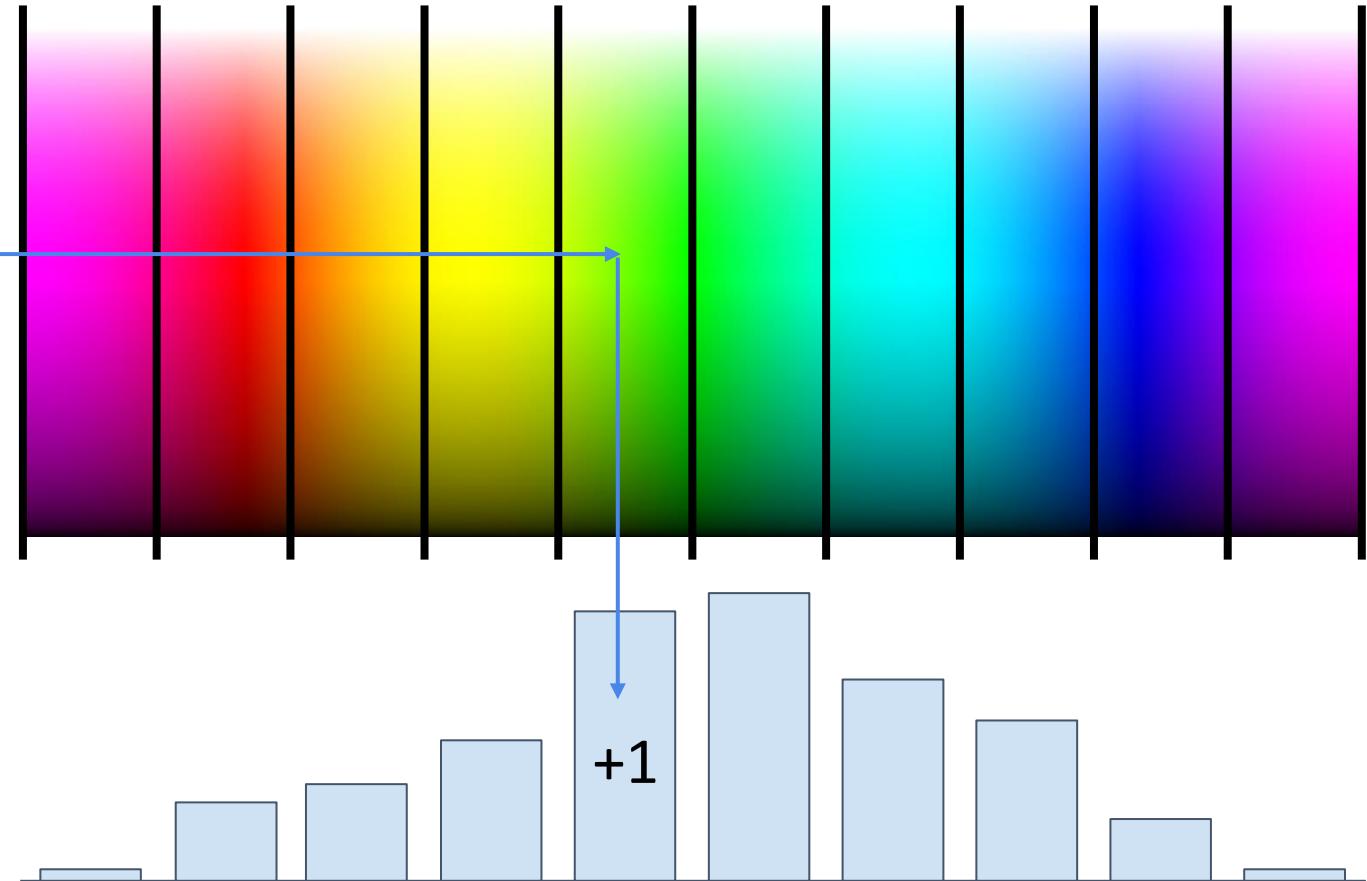


Image Features: Color Histogram



Ignores texture,
spatial positions

[Frog image](#) is in the public domain

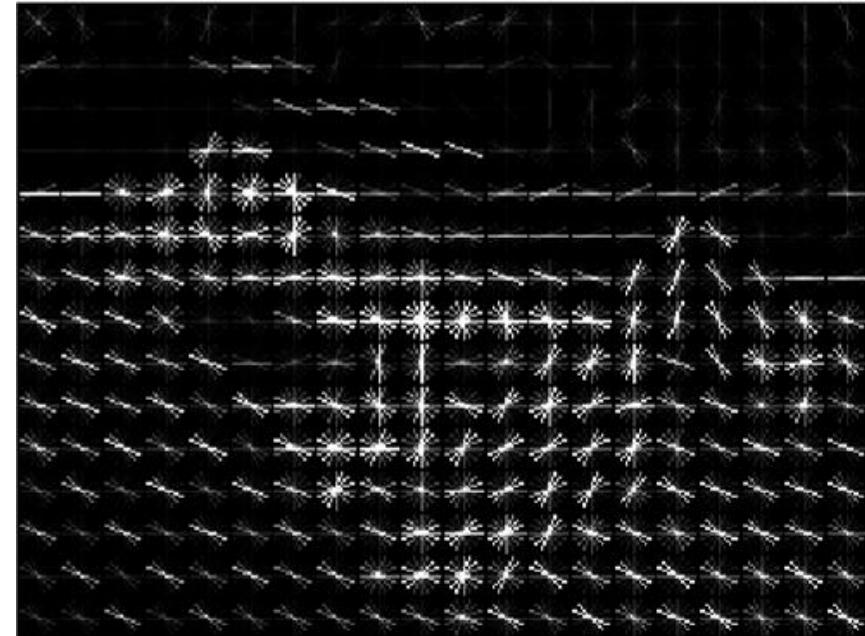
Image Features: Histogram of Oriented Gradients (HoG)



1. Compute edge direction / strength at each pixel
2. Divide image into 8x8 regions
3. Within each region compute a histogram of edge directions weighted by edge strength

Lowe, "Object recognition from local scale-invariant features", ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

Image Features: Histogram of Oriented Gradients (HoG)

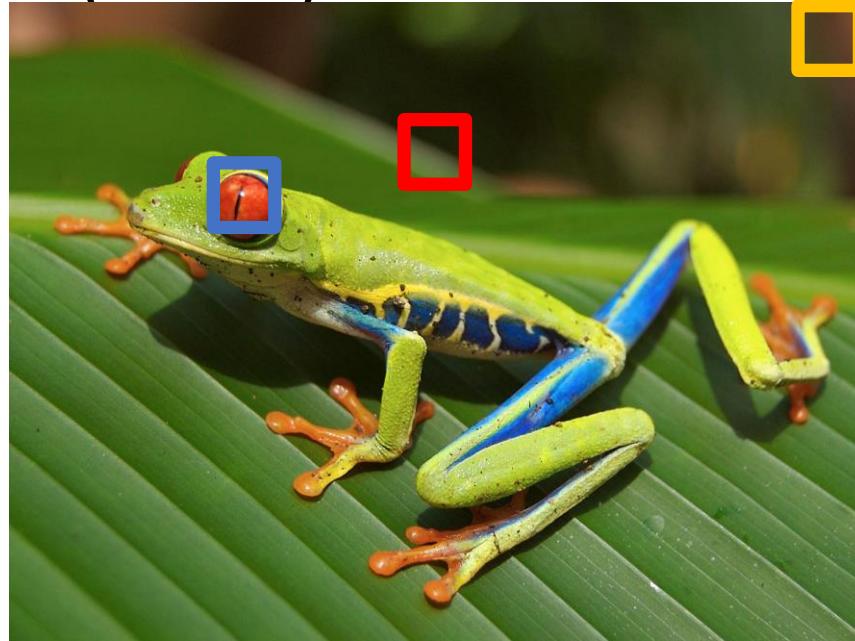


1. Compute edge direction / strength at each pixel
2. Divide image into 8×8 regions
3. Within each region compute a histogram of edge directions weighted by edge strength

Example: 320×240 image gets divided into 40×30 bins; 8 directions per bin; feature vector has $30 * 40 * 9 = 10,800$ numbers

Lowe, "Object recognition from local scale-invariant features", ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

Image Features: Histogram of Oriented Gradients (HoG)



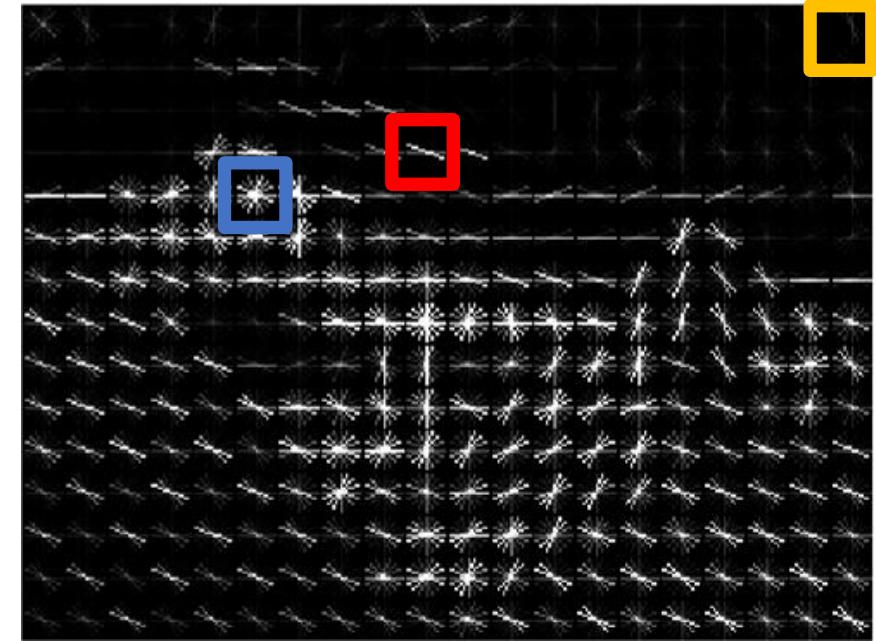
Weak edges

Strong diagonal edges



Edges in all directions

Captures
texture and
position,
robust to
small image
changes



1. Compute edge direction / strength at each pixel
2. Divide image into 8x8 regions
3. Within each region compute a histogram of edge directions weighted by edge strength

Example: 320x240 image gets divided into 40x30 bins; 8 directions per bin; feature vector has $30 \times 40 \times 9 = 10,800$ numbers

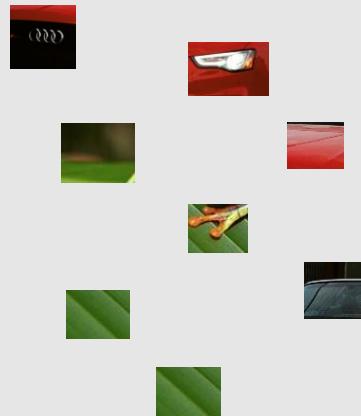
Lowe, "Object recognition from local scale-invariant features", ICCV 1999
Dalal and Triggs, "Histograms of oriented gradients for human detection," CVPR 2005

Image Features: Bag of Words (Data-Driven!)

Step 1: Build codebook



Extract random patches



Cluster patches to form “codebook” of “visual words”

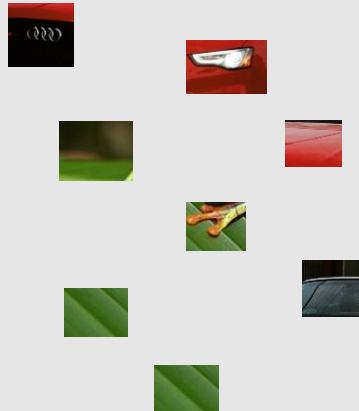


Image Features: Bag of Words (Data-Driven!)

Step 1: Build codebook



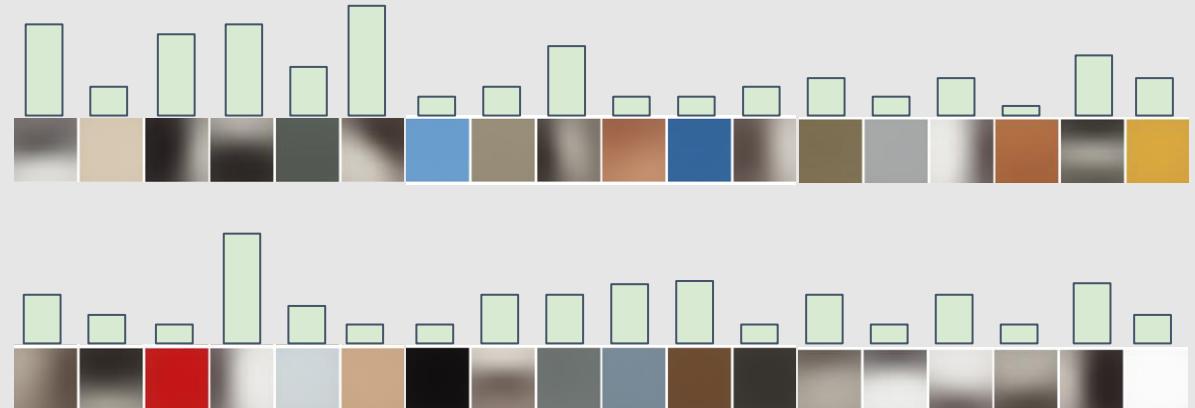
Extract random patches



Cluster patches to form “codebook” of “visual words”

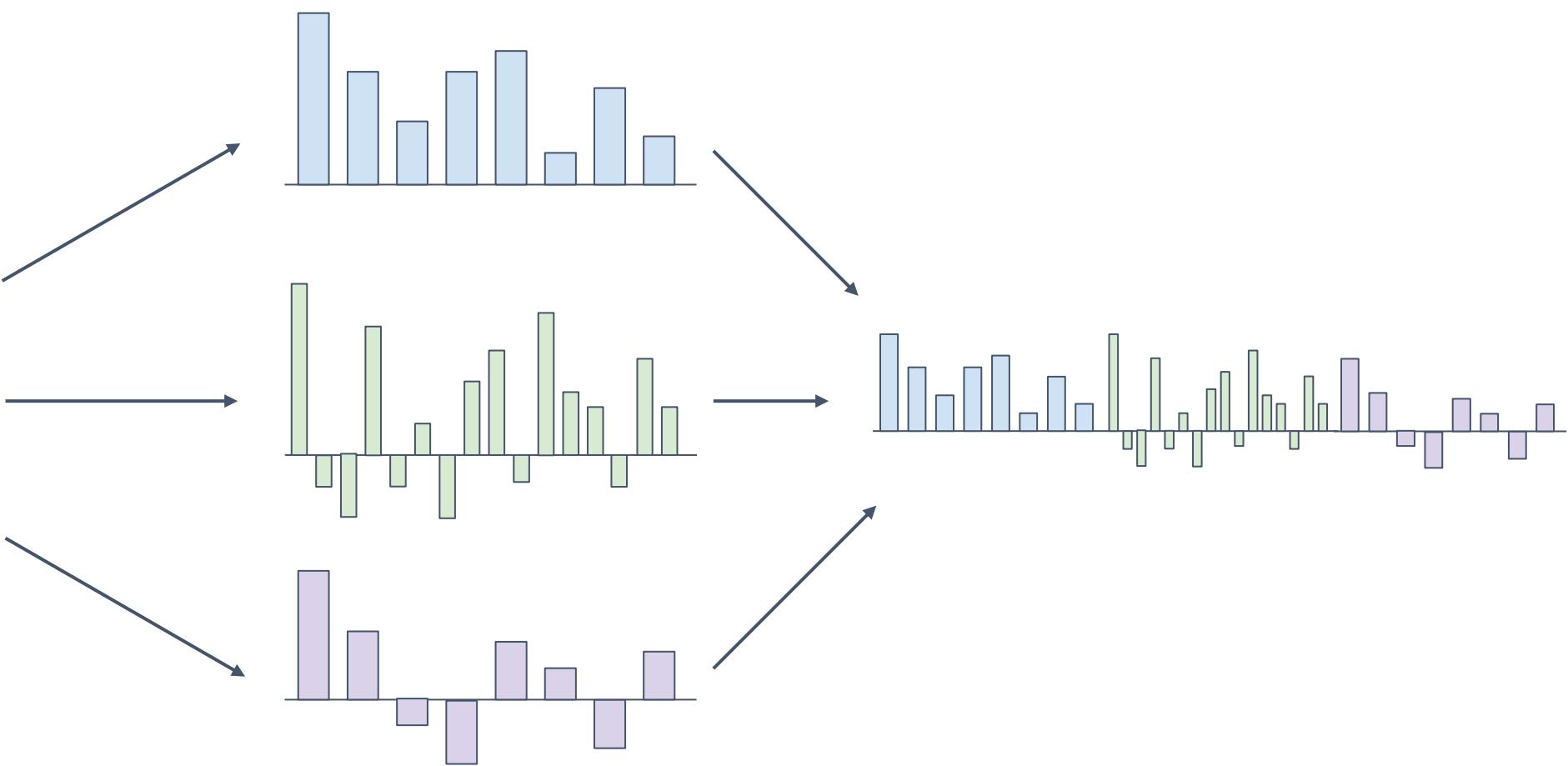


Step 2: Encode images



Fei-Fei and Perona, “A bayesian hierarchical model for learning natural scene categories”, CVPR 2005

Image Features



Example: Winner of 2011 ImageNet challenge

Low-level feature extraction \approx 10k patches per image

- SIFT: 128-dim
 - color: 96-dim
- }
- reduced to 64-dim with PCA

FV extraction and compression:

- $N=1,024$ Gaussians, $R=4$ regions \Rightarrow 520K dim x 2
- compression: $G=8$, $b=1$ bit per dimension

One-vs-all SVM learning with SGD

Late fusion of SIFT and color systems

Image Features

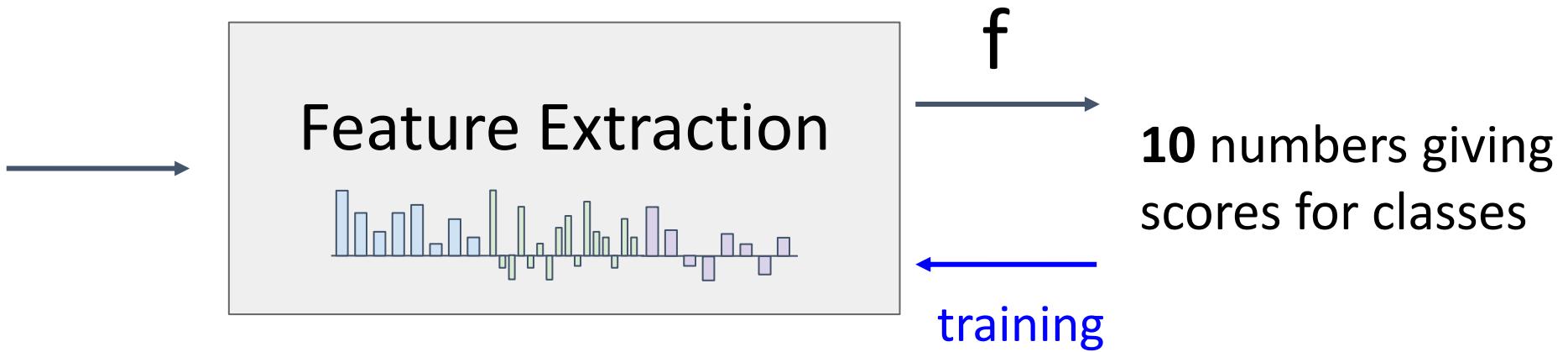
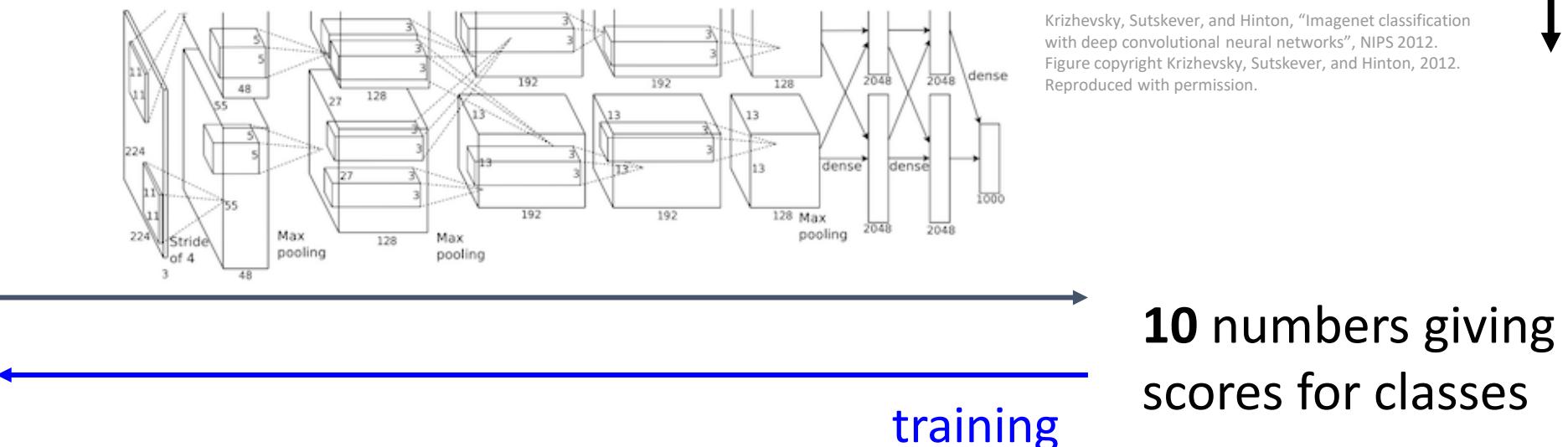
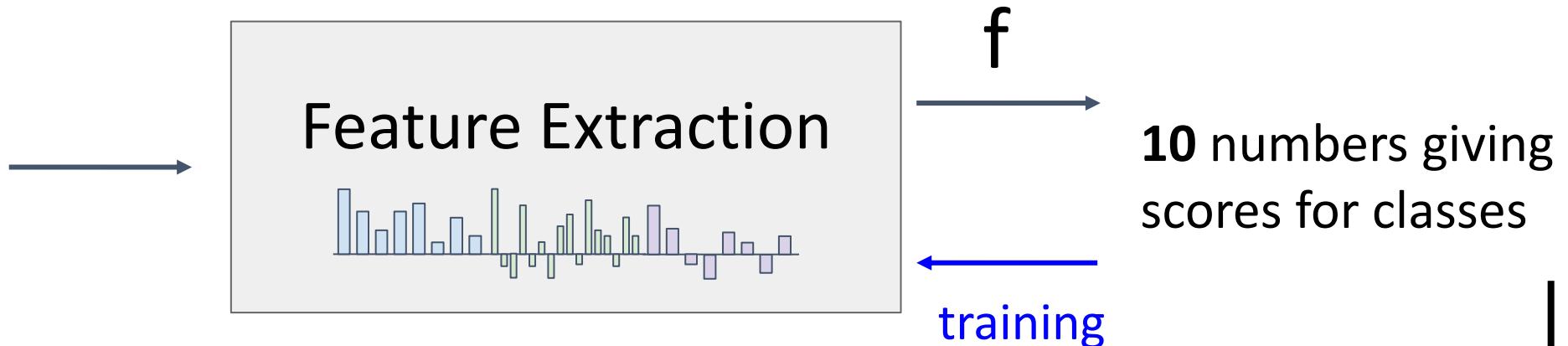


Image Features vs Neural Networks



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012.
Figure copyright Krizhevsky, Sutskever, and Hinton, 2012.
Reproduced with permission.

Neural Networks

(Before) Linear score function:

$$f = Wx$$

$$x \in \mathbb{R}^D, W \in \mathbb{R}^{C \times D}$$

Neural Networks

(Before) Linear score function:

$$f = W\mathbf{x}$$

(Now) 2-layer Neural Network

$$f = W_2 \max(0, W_1 \mathbf{x})$$

$$W_2 \in \mathbb{R}^{C \times H} \quad W_1 \in \mathbb{R}^{H \times D} \quad \mathbf{x} \in \mathbb{R}^D$$

(In practice we will usually add a learnable bias at each layer as well)

Neural Networks

(Before) Linear score function:

$$f = Wx$$

(Now) 2-layer Neural Network

$$f = W_2 \max(0, W_1 x)$$

or 3-layer Neural Network

$$f = W_3 \max(0, W_2 \max(0, W_1 x))$$

$$W_3 \in \mathbb{R}^{C \times H_2} \quad W_2 \in \mathbb{R}^{H_2 \times H_1} \quad W_1 \in \mathbb{R}^{H_1 \times D} \quad x \in \mathbb{R}^D$$

(In practice we will usually add a learnable bias at each layer as well)

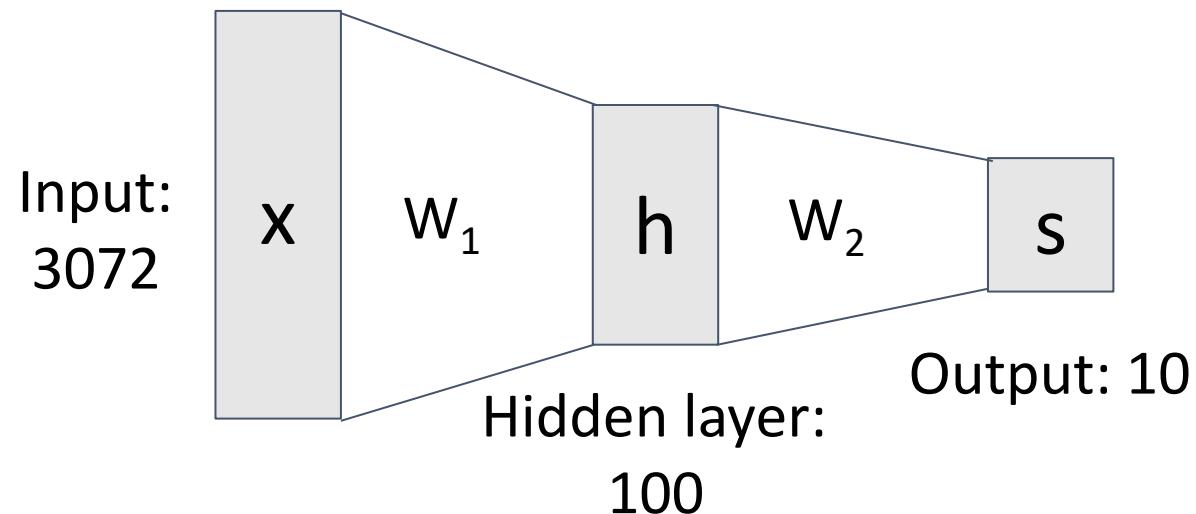
Neural Networks

(Before) Linear score function:

$$f = Wx$$

(Now) 2-layer Neural Network

$$f = W_2 \max(0, W_1 x)$$



$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

Neural Networks

(Before) Linear score function:

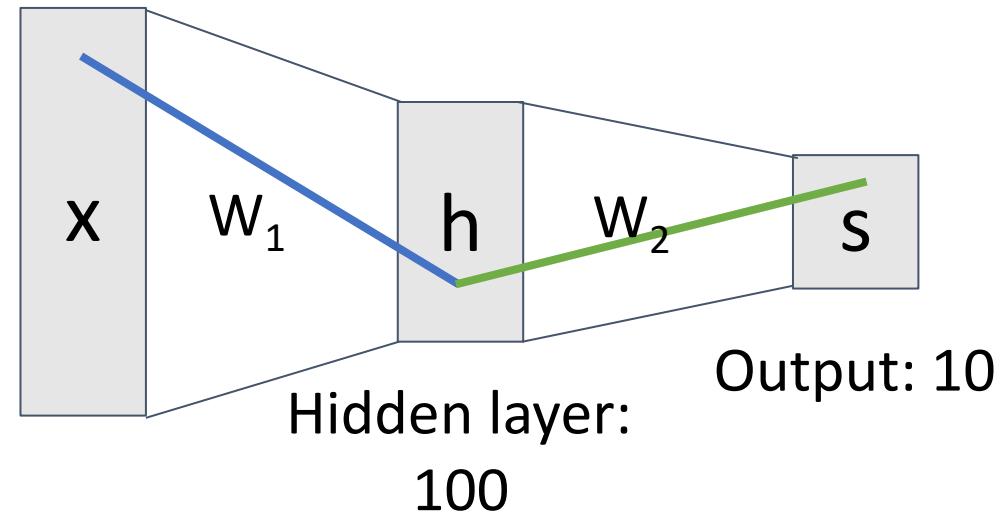
$$f = \mathbf{W}\mathbf{x}$$

(Now) 2-layer Neural Network

$$f = W_2 \max(0, W_1 \mathbf{x})$$

Element (i, j)
of W_1 gives
the effect on
 h_i from x_j

Input:
3072



Element (i, j)
of W_2 gives
the effect on
 s_i from h_j

$$\mathbf{x} \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

Neural Networks

(Before) Linear score function:

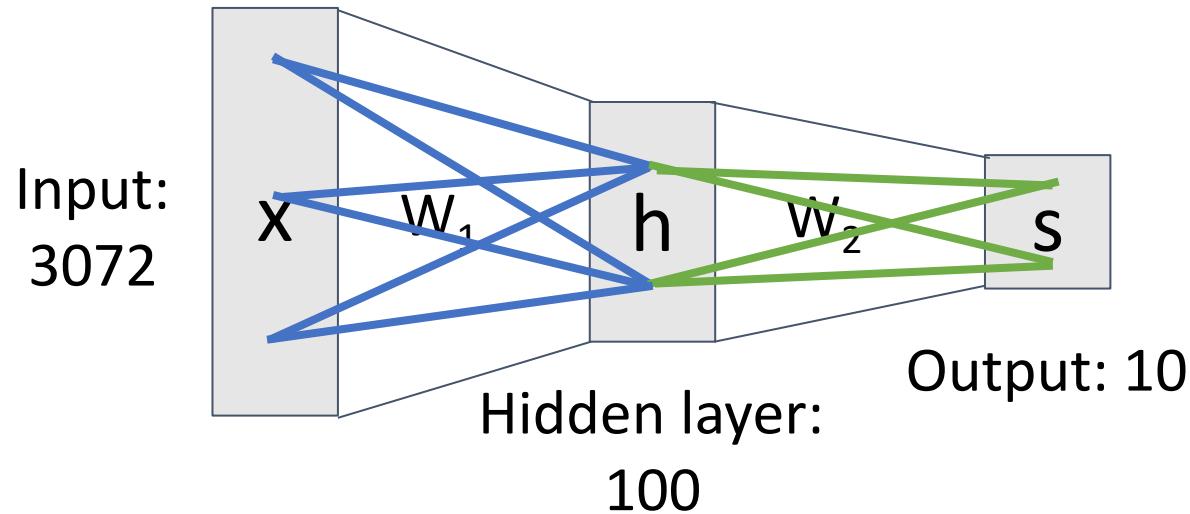
$$f = \mathbf{W}\mathbf{x}$$

(Now) 2-layer Neural Network

$$f = W_2 \max(0, W_1 \mathbf{x})$$

Element (i, j) of W_1
gives the effect on
 h_i from x_j

All elements
of \mathbf{x} affect all
elements of \mathbf{h}



Element (i, j) of W_2
gives the effect on
 s_i from h_j

All elements
of \mathbf{h} affect all
elements of \mathbf{s}

Fully-connected neural network
Also “Multi-Layer Perceptron” (MLP)

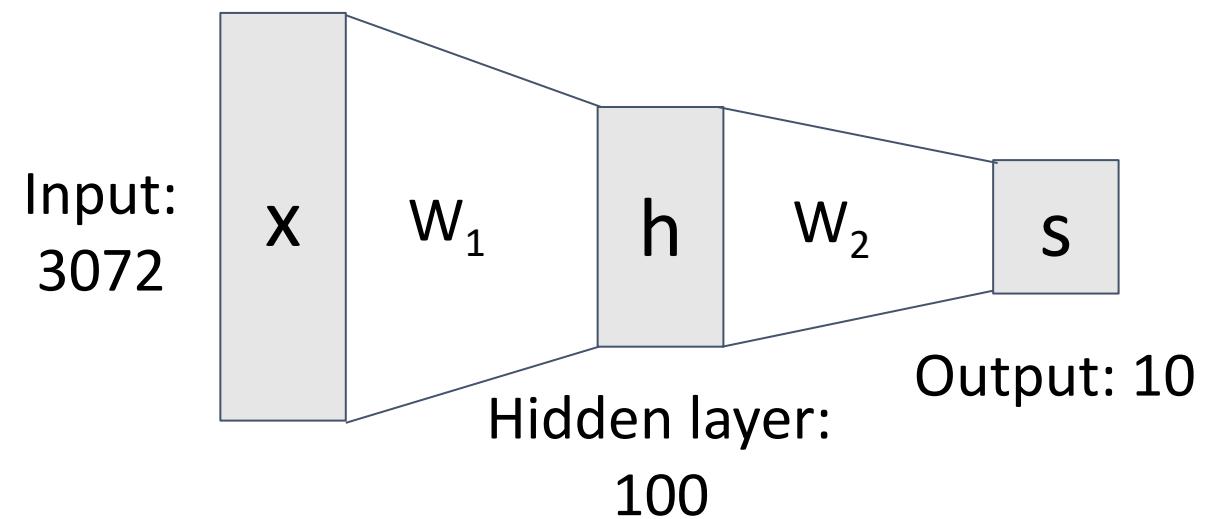
Neural Networks

Linear classifier: One template per class



(Before) Linear score function:

(Now) 2-layer Neural Network



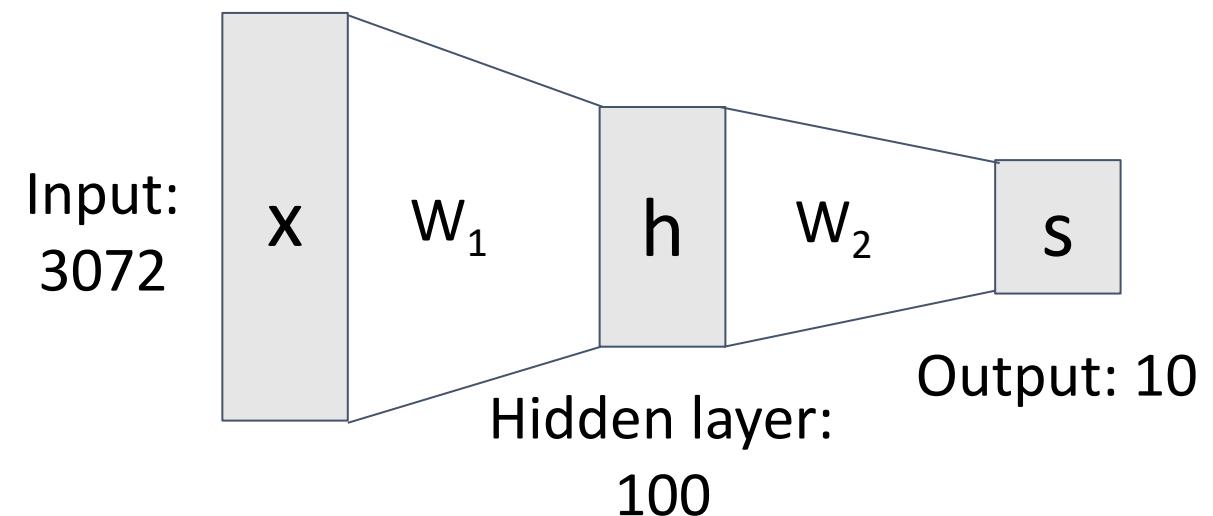
Neural Networks

Neural net: first layer is bank of templates;
Second layer recombines templates



(Before) Linear score function:

(Now) 2-layer Neural Network



$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

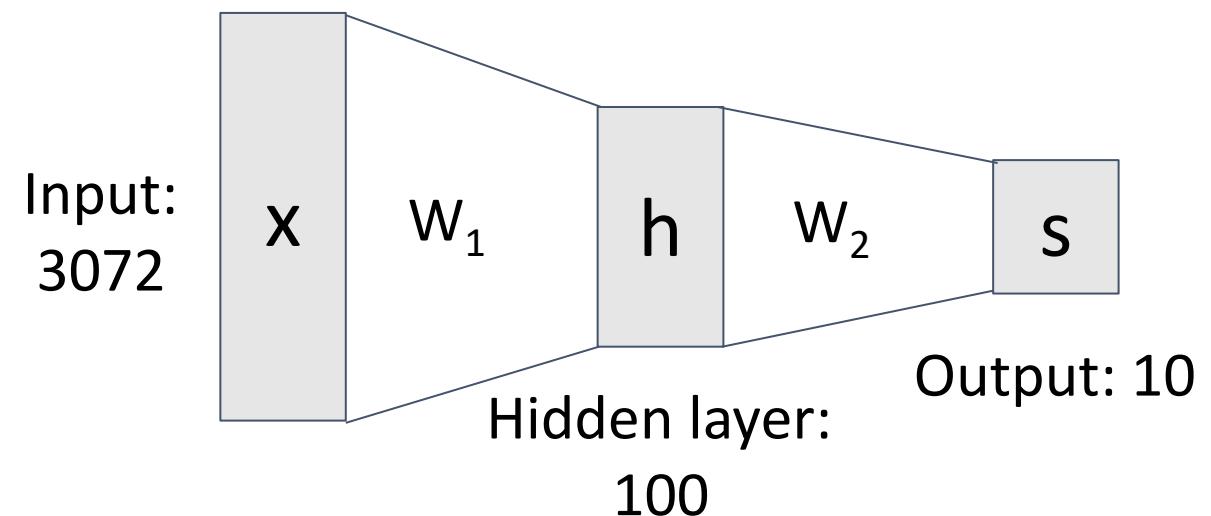
Neural Networks

Can use different templates to cover multiple modes of a class!



(Before) Linear score function:

(Now) 2-layer Neural Network



$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

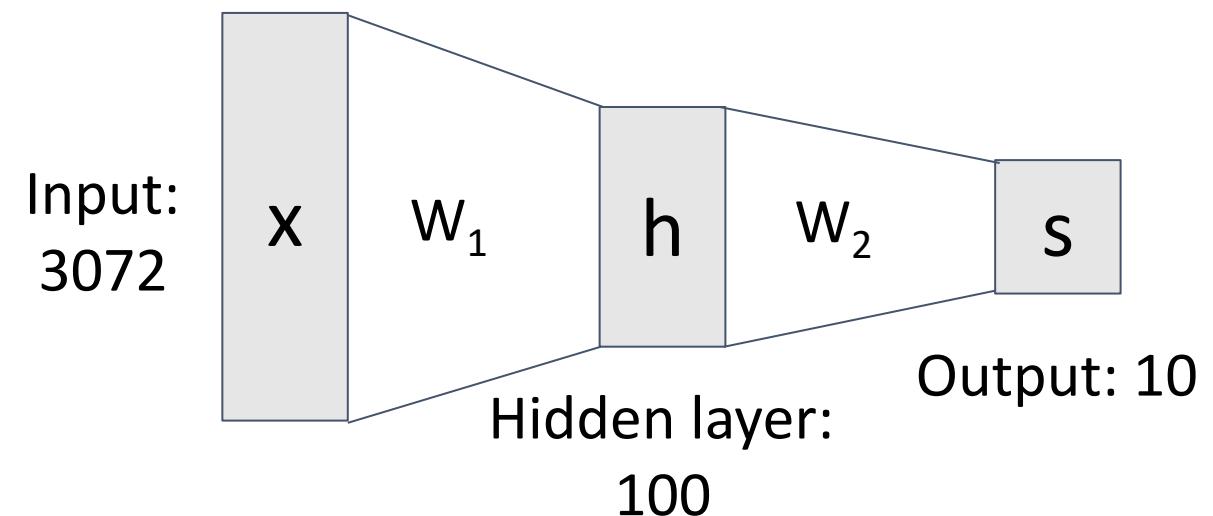
Neural Networks

“Distributed representation”:
Most templates not interpretable!



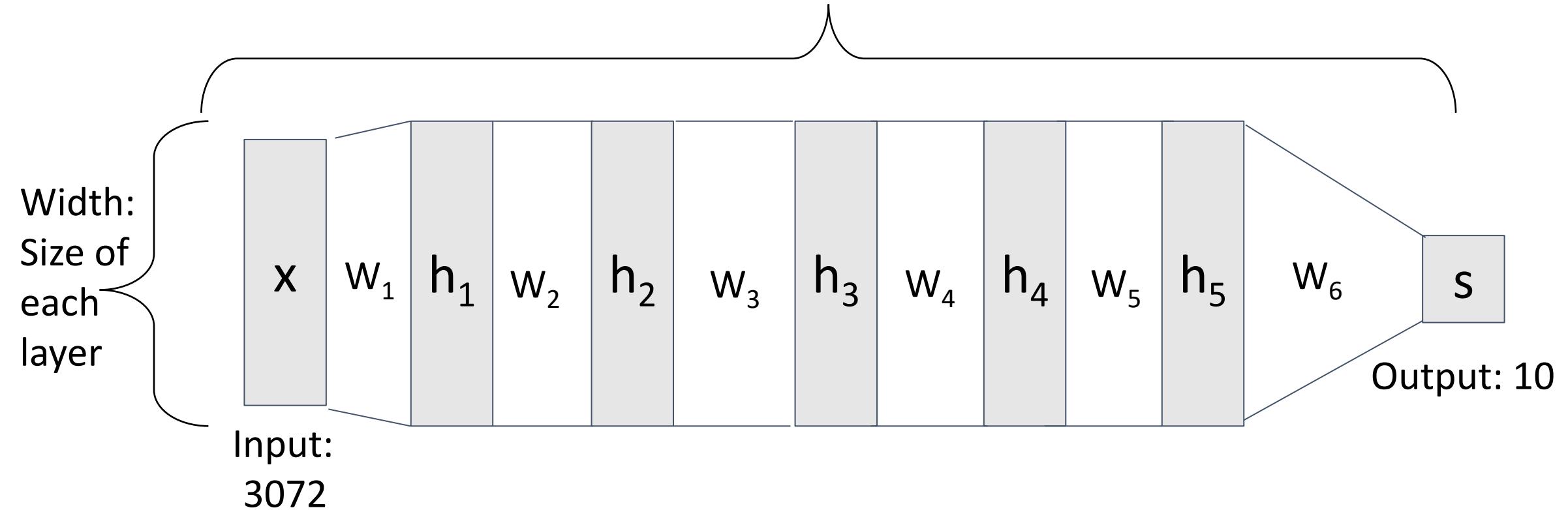
(Before) Linear score function:

(Now) 2-layer Neural Network



Deep Neural Networks

Depth = number of layers



$$s = W_6 \max(0, W_6 \max(0, W_5 \max(0, W_4 \max(0, W_3 \max(0, W_2 \max(0, W_1 x))))))$$

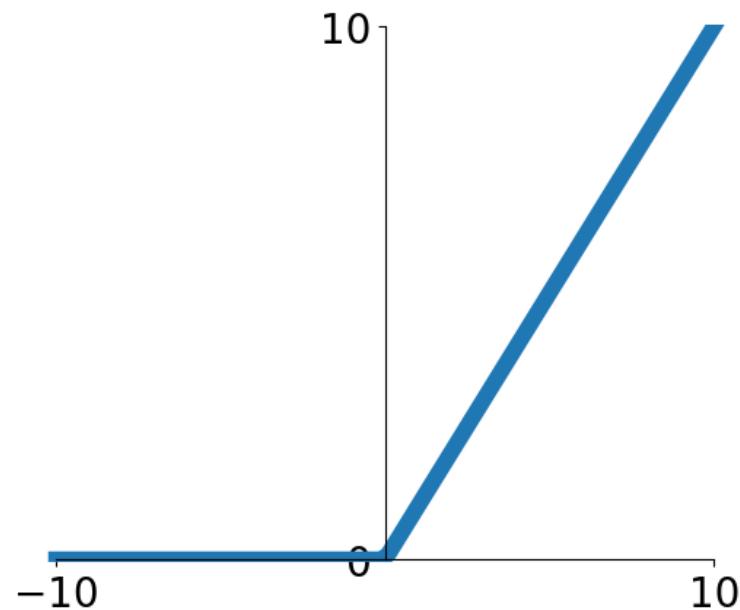
Activation Functions

2-layer Neural Network

The function $ReLU(z) = \max(0, z)$
is called “Rectified Linear Unit”

$$f = W_2 \boxed{\max(0, W_1 x)}$$

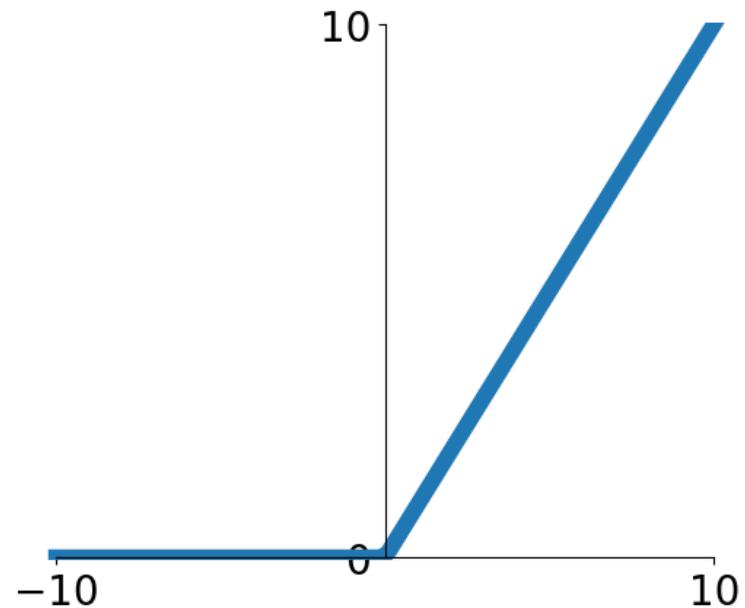
This is called the **activation function** of
the neural network



Activation Functions

2-layer Neural Network

The function $ReLU(z) = \max(0, z)$ is called “Rectified Linear Unit”



$$f = W_2 \boxed{\max(0, W_1 x)}$$

This is called the **activation function** of the neural network

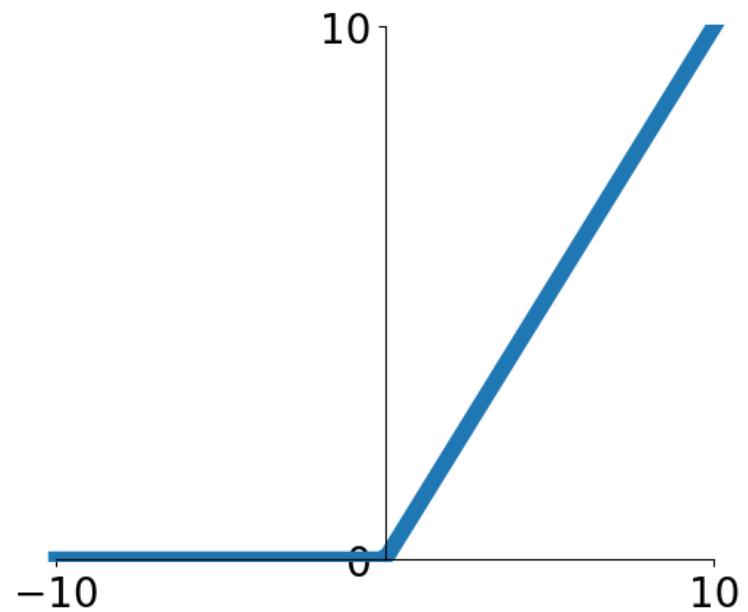
Q: What happens if we build a neural network with no activation function?

$$s = W_2 W_1 x$$

Activation Functions

2-layer Neural Network

The function $ReLU(z) = \max(0, z)$ is called “Rectified Linear Unit”



$$f = W_2 \boxed{\max(0, W_1 x)}$$

This is called the **activation function** of the neural network

Q: What happens if we build a neural network with no activation function?

$$s = W_2 W_1 x$$

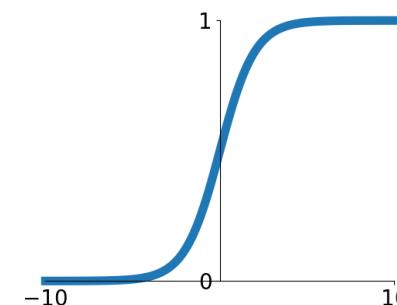
$$W_3 = W_2 W_1 \in \mathbb{R}^{C \times H} \quad s = W_3 x$$

A: We end up with a linear classifier!

Activation Functions

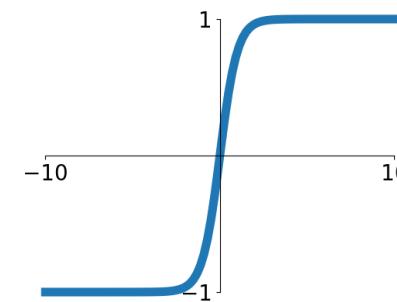
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



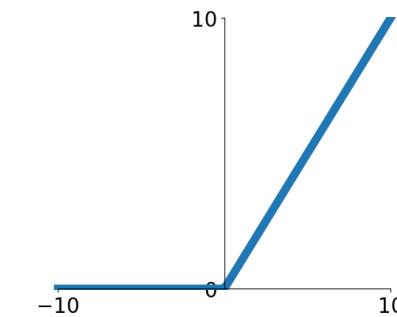
tanh

$$\tanh(x)$$



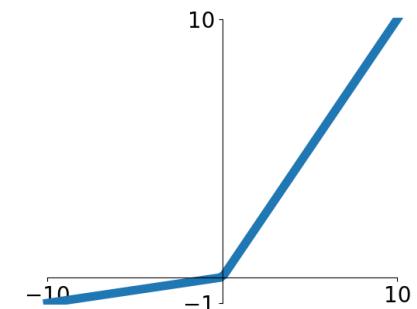
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

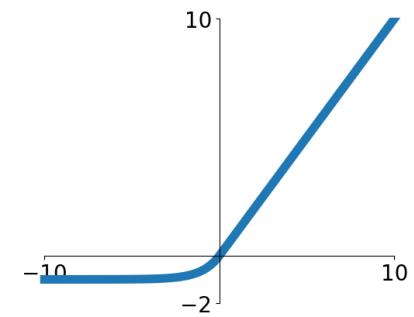


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

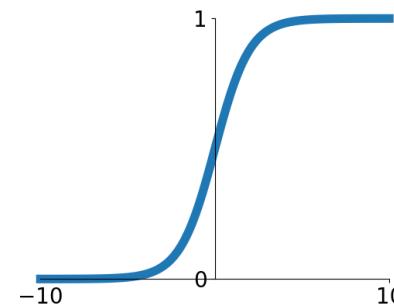
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation Functions

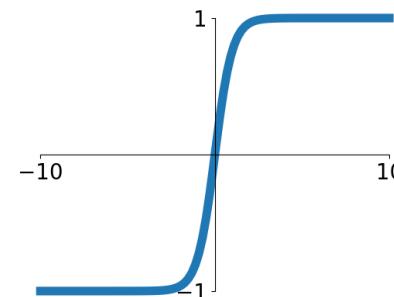
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



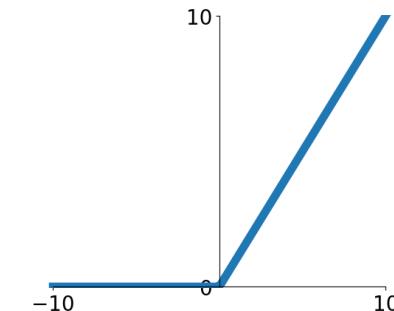
tanh

$$\tanh(x)$$



ReLU

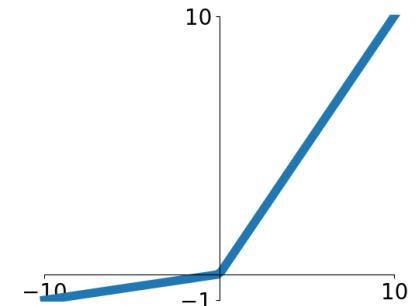
$$\max(0, x)$$



ReLU is a good default choice
for most problems

Leaky ReLU

$$\max(0.1x, x)$$

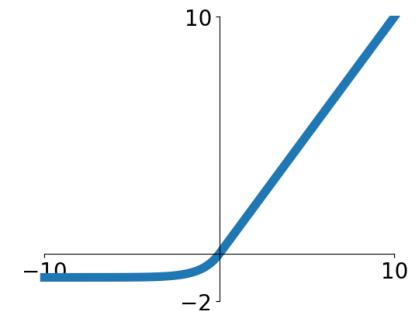


Maxout

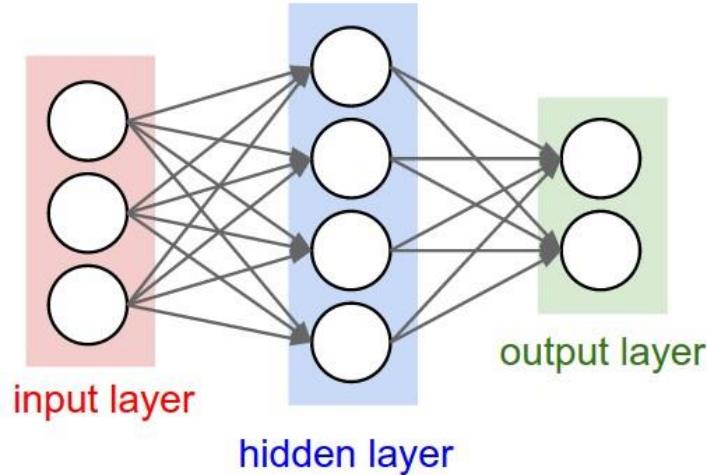
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Neural Net in <20 lines!



Initialize weights
and data

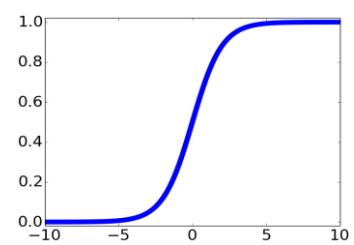
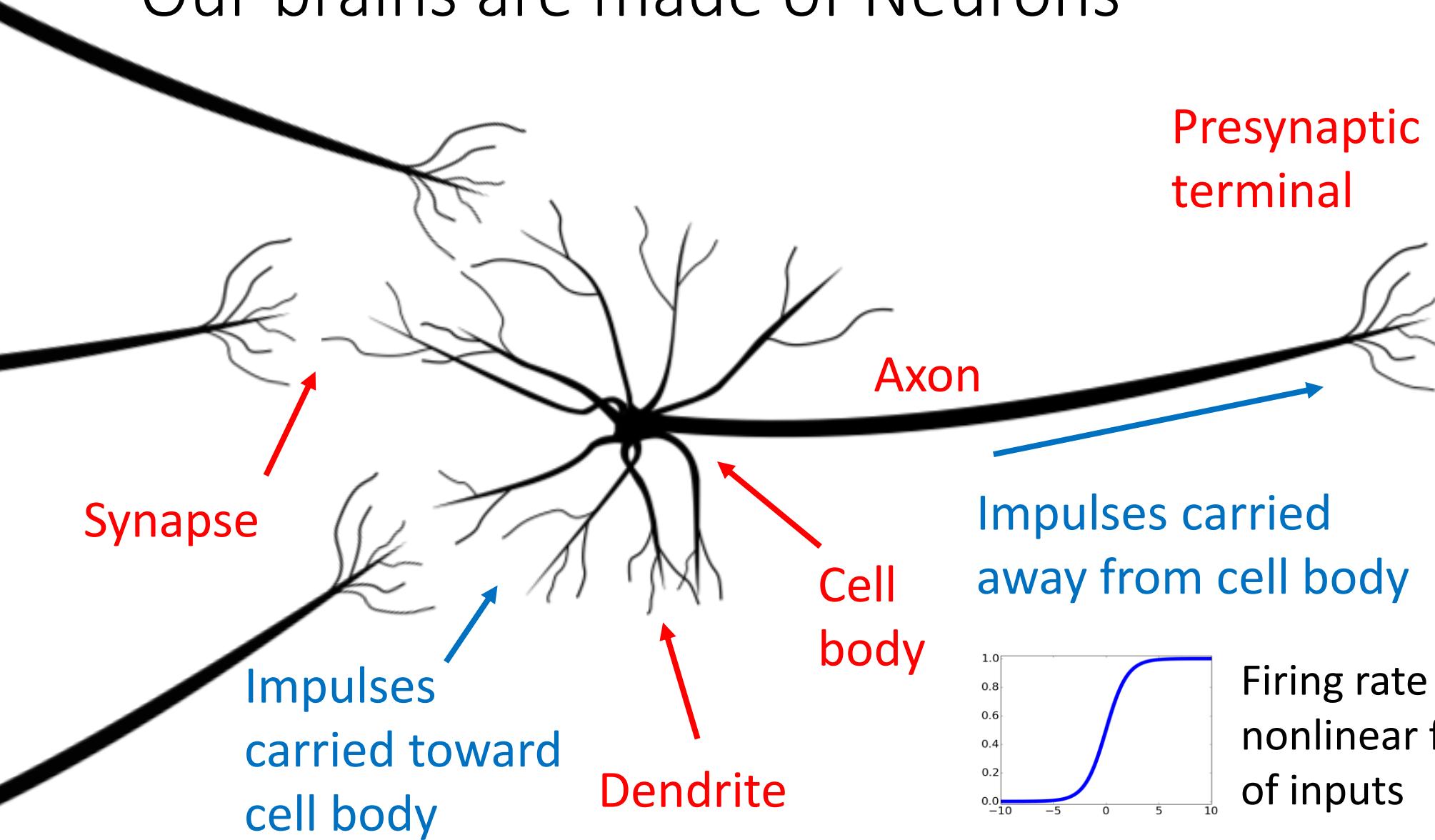
Compute loss
(sigmoid activation,
L2 loss)

Compute
gradients

Stochastic Gradient
Descent (SGD) step

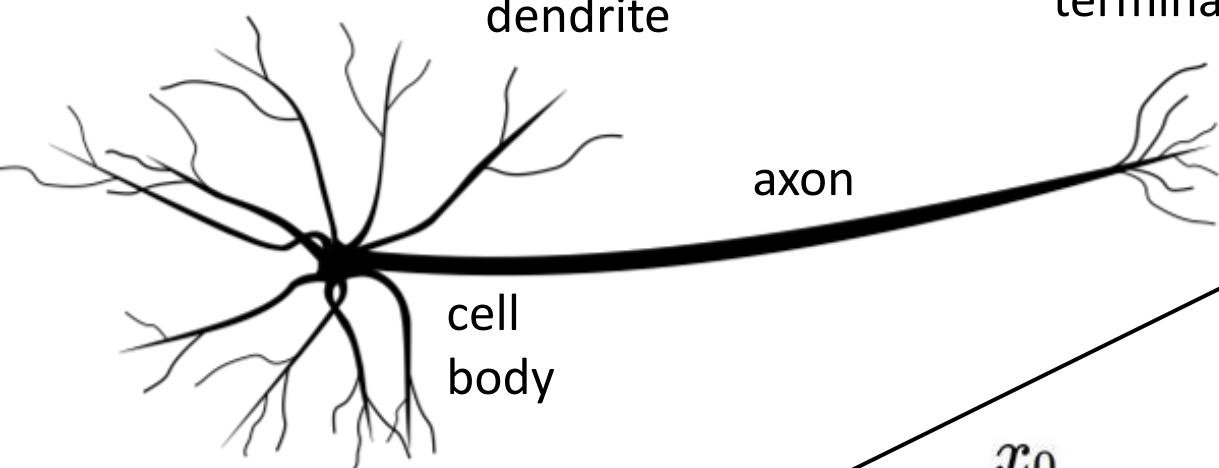
```
1 import numpy as np
2 from numpy.random import randn
3
4 N, Din, H, Dout = 64, 1000, 100, 10
5 x, y = randn(N, Din), randn(N, Dout)
6 w1, w2 = randn(Din, H), randn(H, Dout)
7 for t in range(10000):
8     h = 1.0 / (1.0 + np.exp(-x.dot(w1)))
9     y_pred = h.dot(w2)
10    loss = np.square(y_pred - y).sum()
11    dy_pred = 2.0 * (y_pred - y)
12    dw2 = h.T.dot(dy_pred)
13    dh = dy_pred.dot(w2.T)
14    dw1 = x.T.dot(dh * h * (1 - h))
15    w1 -= 1e-4 * dw1
16    w2 -= 1e-4 * dw2
```

Our brains are made of Neurons



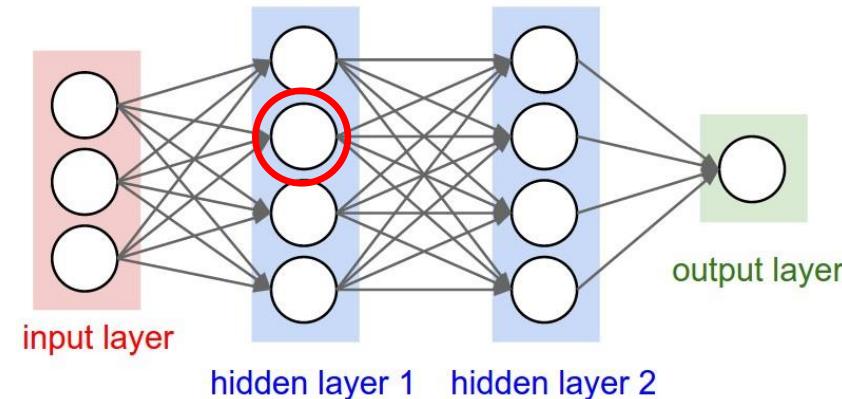
Firing rate is a
nonlinear function
of inputs

Biological Neuron



presynaptic
terminal

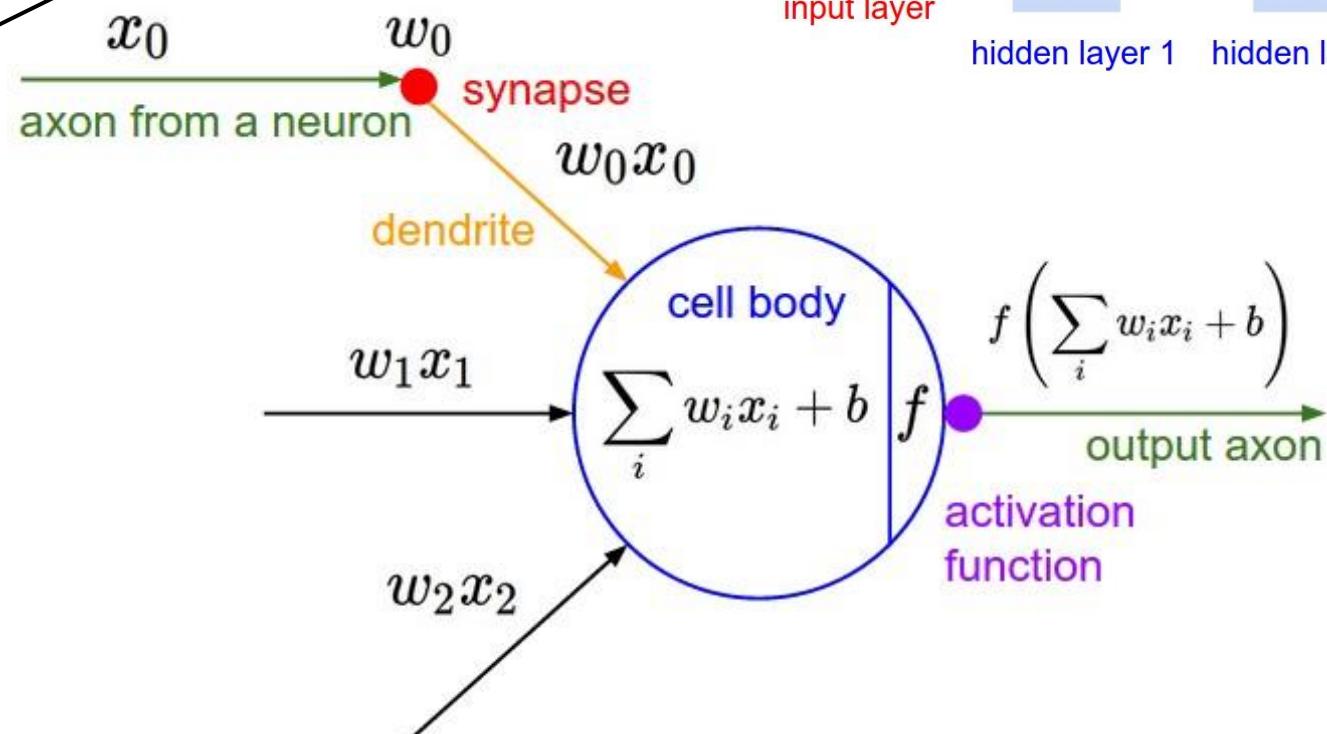
Artificial Neuron



input layer

hidden layer 1 hidden layer 2

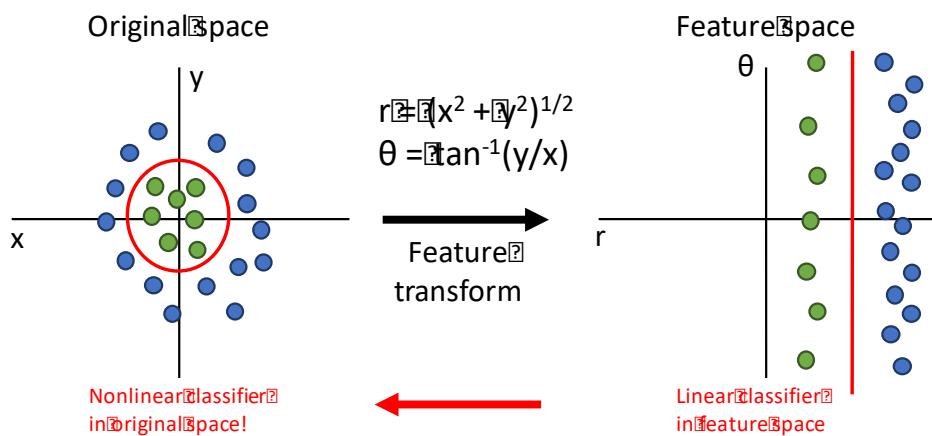
output layer



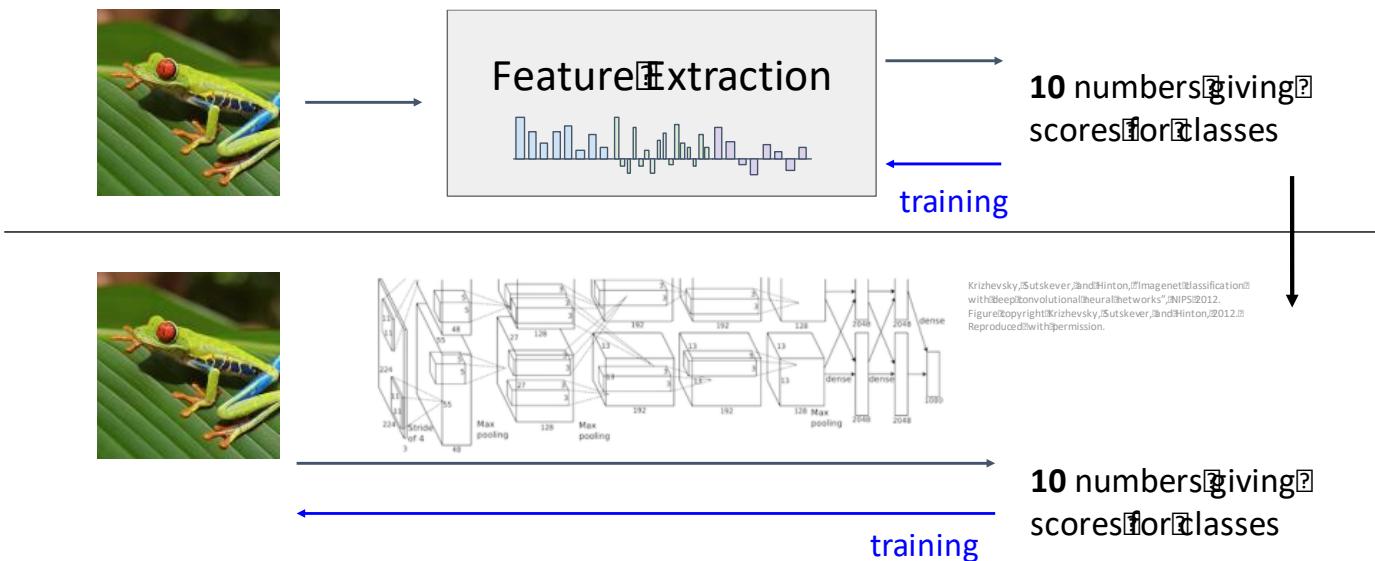
[Neuron image](#) by Felipe Perucho
is licensed under [CC-BY 3.0](#)

Summary

Feature transform + Linear classifier
allows nonlinear decision boundaries



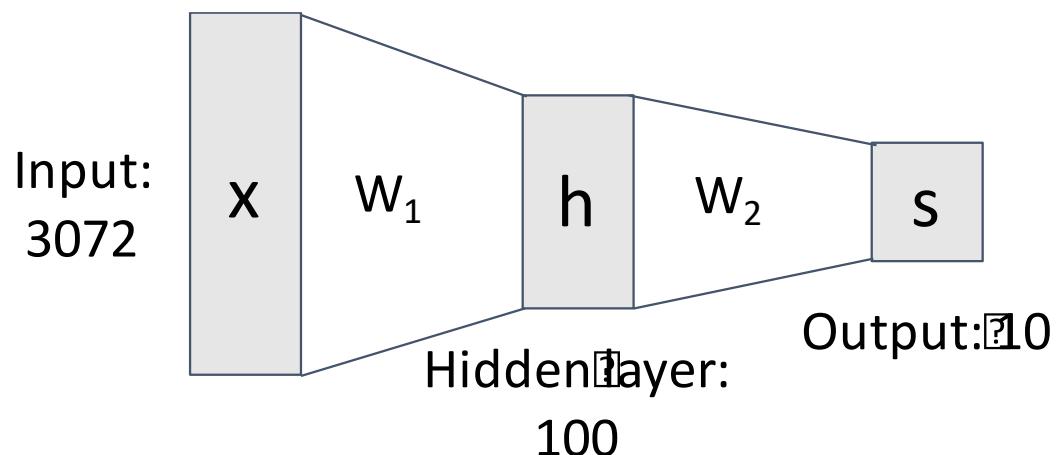
Neural Networks as learnable feature transforms



Summary

From linear classifiers to
fully-connected networks

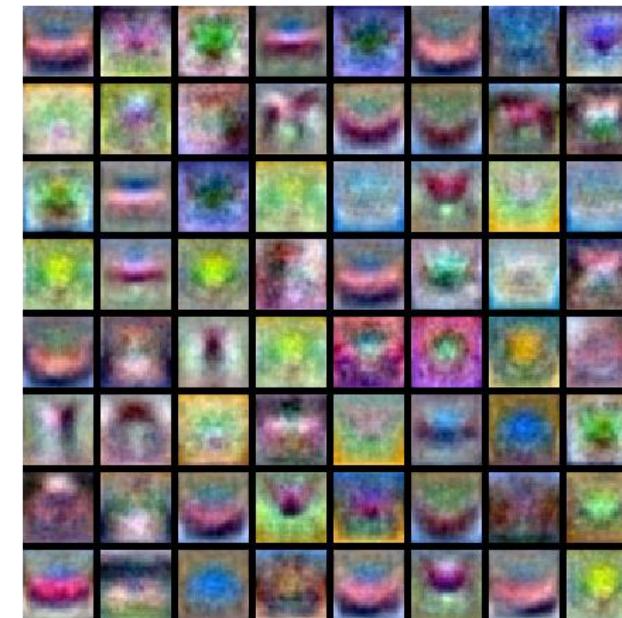
$$f = W_2 \max(0, W_1 x)$$



Linear classifier: One template per class



Neural networks: Many reusable templates



Final Project Guidelines

3/9/2023

Final Project+ Presentation

GOAL: The goal of the project is to give you practice at doing research and development in computer vision

Literature + Implementation + Presentation + Report

PROJECT CAN BE:

- Application of existing algorithms to a new problem;
- A variation on an existing algorithm;
- Design, implementation, and testing of a new algorithm.

RESOURCES

Data:

- Your MS/PhD research data
- Benchmark data from various sources
 - General: <https://paperswithcode.com/datasets?mod=images&page=1>
 - Medical: http://grand-challenge.org/All_Challenges/

Literature: Check recent papers in (see following slides)

Final Project+ Presentation (Literature)

Literature: Check **recent** papers in <https://ieeexplore.ieee.org/Xplore/home.jsp>

Conferences:

- General Image Processing, Computer Vision, Pattern Recognition:
 - <https://openaccess.thecvf.com/CVPR2022> ---- Start from Here
 - International Conference on Computer Vision (ICCV),
 - IEEE CVPR, IEEE ICPR, IEEE ICIP
- IEEE ISBI: International Symposium on BIOMEDICAL IMAGING: From Nano to Macro
- MICCAI: Medical Image Computing and Computer Assisted Interventions
- IEEE EMBC: Annual Conference of IEEE Engineering in Medicine and Biology Society
-

Journals:

General Image Processing, Computer Vision, Pattern Recognition:

- IEEE Trans. Pattern Analysis and Machine Intelligence
- IEEE Trans. Image Processing
- International Journal of Computer Vision

Final Project & Presentation



Steps:

1. Search potential project topics
2. Submit 1 page **Project Proposal** of what you plan to do (**Deadline Nov 1st**)
 - You can talk to me before that
 - Try to skim some of the relevant references before you write up this proposal .
4. Do literature search on related previous work,
5. **Progress update Nov 17**, 3-4 slides
 - Problem being solved,
 - Previous work you found,
 - Algorithm you're using,
 - What's novel,
 - How you plan to test your method (inputs & outputs).
6. Implement and test,
7. Give a **presentation** in class (**Dec 6-9**)
8. Report due **Dec 13**: write up a description of your work, 8-12 pages in length. The paper should be in the form of a conference paper. Be sure to compare your results to previous work and include references to at least three related papers.

Final Project Proposal (March 21) – 1 or 2 pages

1. Team members

- Individual
- Group of two students.

2. Title of project

3. Scope of your project

1. Problem being solved
2. Previous work you found
3. Algorithm/method you will be using
4. Data (inputs & outputs)
5. How you plan to test your method

- Submit to Canvas
- If you are in a team of two, each person should upload the project proposal idea individually for grading.

Final Project Progress Update (April 13)

3-4 slides updating progress since project proposal

1. Problem being solved
2. Previous work you found
3. Algorithm/method you will be using
4. Data (inputs & outputs)
5. How you plan to test your method

Final Project Update: General Comments

- Start with MVA (minimum viable product)
 1. Simplify your problem: list your assumptions
 2. Start coding your pipeline
 3. Show preliminary results
 4. Explain limitations
 5. Extend your approach to overcome these limitations

Final Project Report (Due May 9)

4-8 pages double-column IEEE style conference paper. **Highly Recommend use of Latex/Overleaf**
<https://www.overleaf.com/project>

Should have the following sections:

1. **Introduction & overview** : introduce the problem/application, review the relevant materials, give an overview of your system.
2. **Method**: describe your method. Flow charts and figures help a lot.
3. **Experimental results**: tables, graphs, images
4. **Discussion**: interpret and elaborate on your result, discuss how to improve, discuss advantages disadvantages of your approach. Interpretation should use computer vision terminology and should be very clear. (i.e image is distorted is NOT a clear statement!)
5. **Conclusion**
6. **References**: Cite any relevant material you used in the paper, paper, web sites.

IMPORTANT: References for the final report should be more detailed than a regular paper. Any material text/images etc. you have used in the report, any piece of code that comes from a library (opencv), somebody else, sample code from a paper, download from internet etc..... need to be clearly specified.

Where to Start

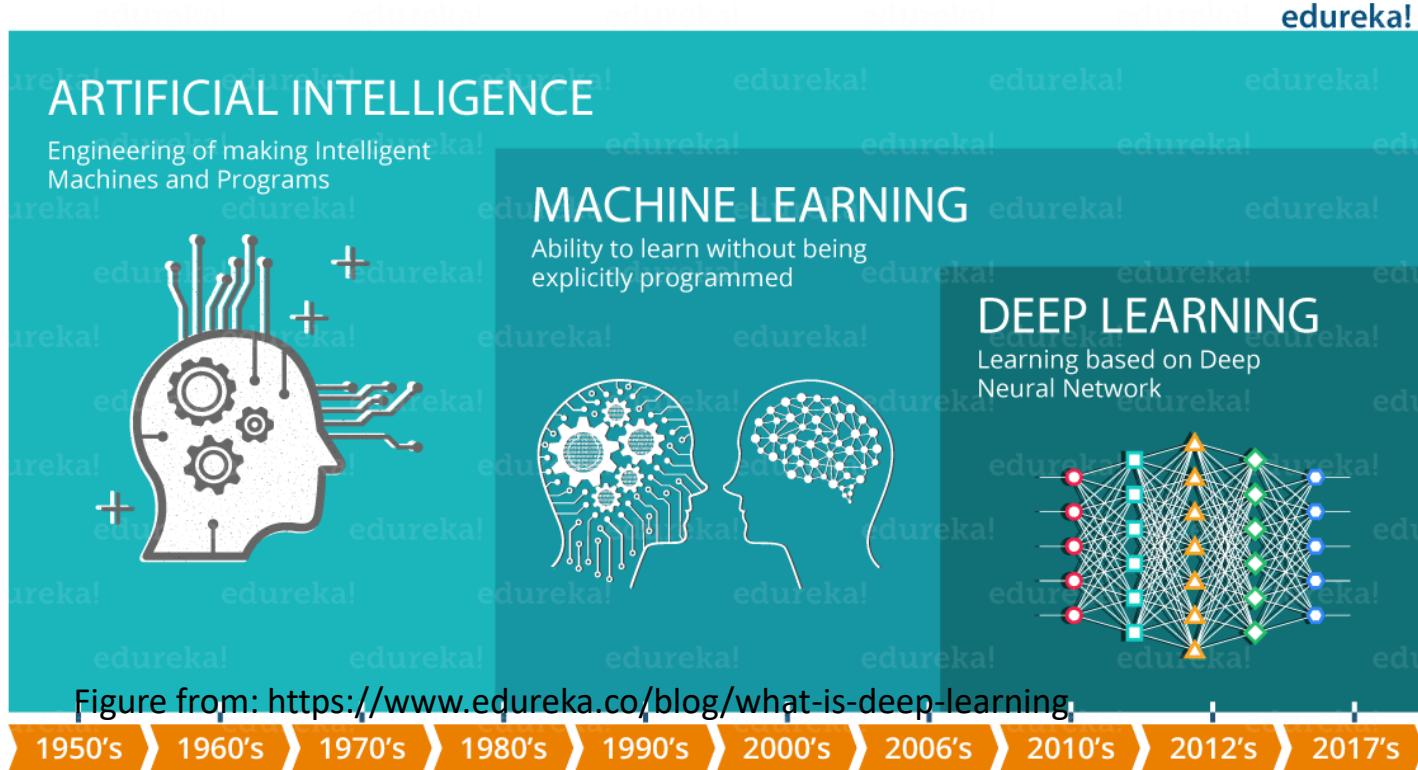
- <https://paperswithcode.com/area/computer-vision>
- 3591 benchmarks • 1146 tasks • 2446 datasets • 31575 papers with code

Computer Vision with Deep Learning

References

1. Szeliski, Computer Vision, Chapter 5.3 and 5.4
2. Liu, Li, Jie Chen, Paul Fieguth, Guoying Zhao, Rama Chellappa, and Matti Pietikäinen. "From BoW to CNN: Two decades of texture representation for texture classification." *International Journal of Computer Vision* 127, no. 1 (2019): 74-109.
3. Mostly complete chart of neural networks <https://www.asimovinstitute.org/neural-network-zoo/>
4. Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).
5. Jiao, Licheng, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. "A Survey of Deep Learning-Based Object Detection." *IEEE Access* 7 (2019): 128837-128868.
6. 2014 - CVPR Tutorial on Deep Learning for Vision - Object Detection, **Pierre Sermanet, Google Research**
7. Slides by Míriam Bellver, Computer Vision Reading Group, UPC, 28th October, 2016

AI vs. ML vs. DL



Artificial Intelligence: making a computer think intelligently, in the similar manner the intelligent humans think (John McCarthy: Computer scientist known as the father of AI).

Machine learning: ability to learn without being explicitly programmed (Arthur Samuel 1959).

Deep learning: learning based on artificial neural networks.

Neural Networks

(Before) Linear score function:

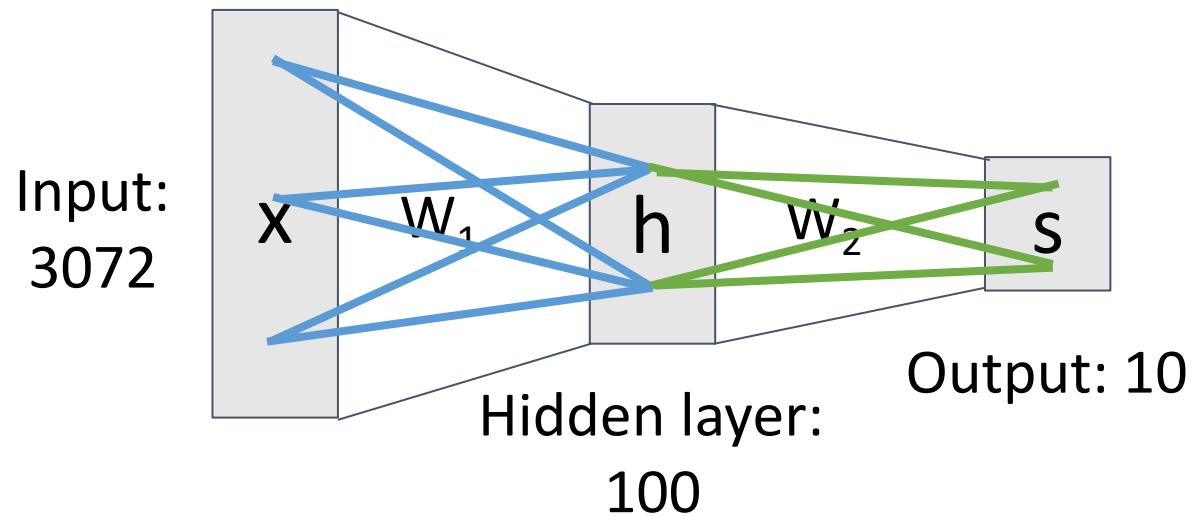
$$f = Wx$$

(Now) 2-layer Neural Network

$$f = W_2 \max(0, W_1 x)$$

Element (i, j) of W_1
gives the effect on
 h_i from x_j

All elements
of x affect all
elements of h



Element (i, j) of W_2
gives the effect on
 s_i from h_j

All elements
of h affect all
elements of s

Fully-connected neural network
Also “Multi-Layer Perceptron” (MLP)

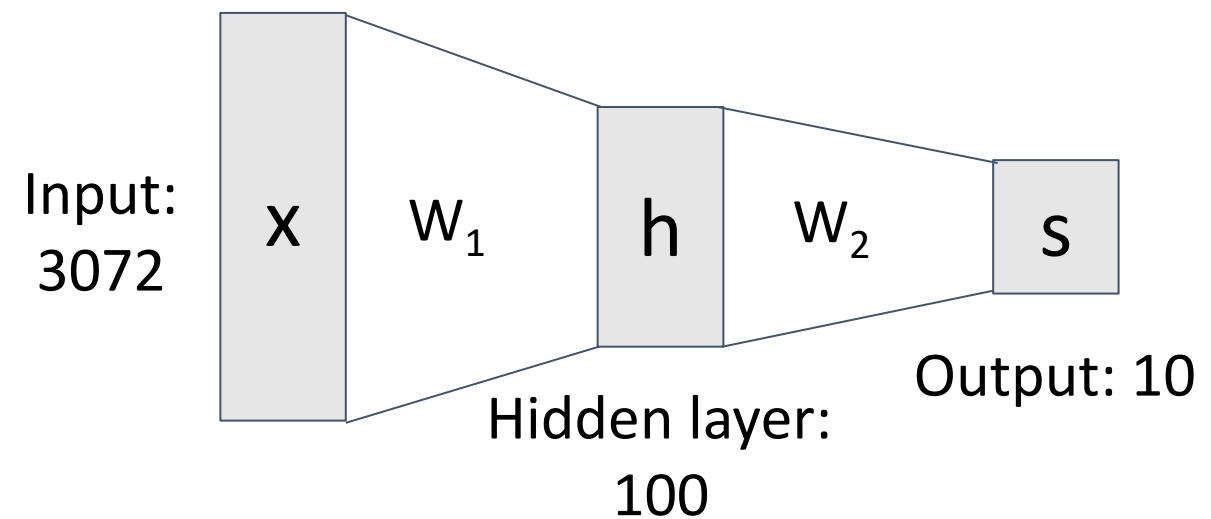
Neural Networks

Linear classifier: One template per class



(Before) Linear score function:

(Now) 2-layer Neural Network



$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

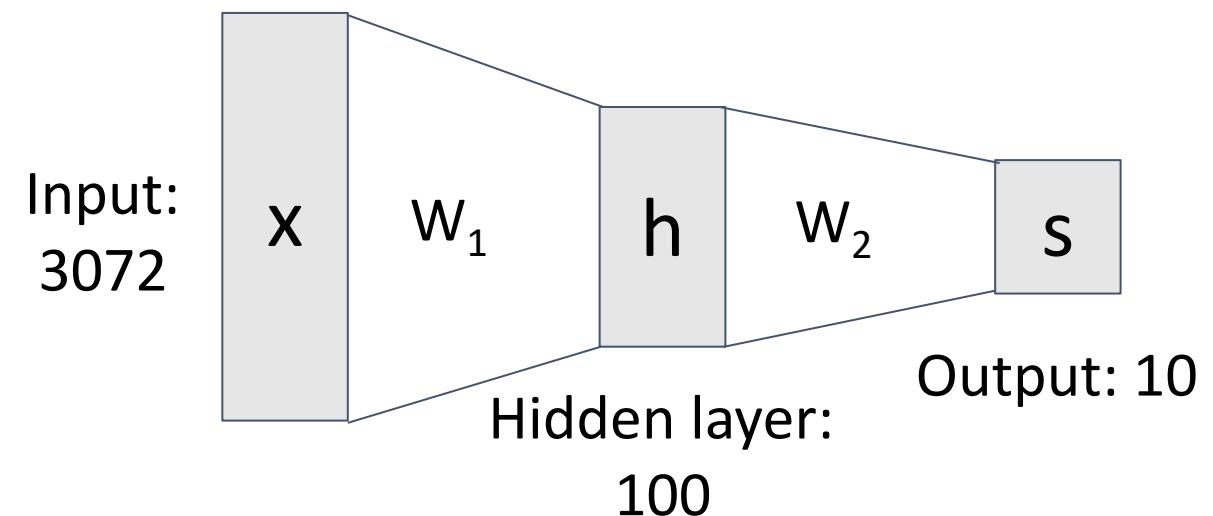
Neural Networks

Neural net: first layer is bank of templates;
Second layer recombines templates



(Before) Linear score function:

(Now) 2-layer Neural Network



$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

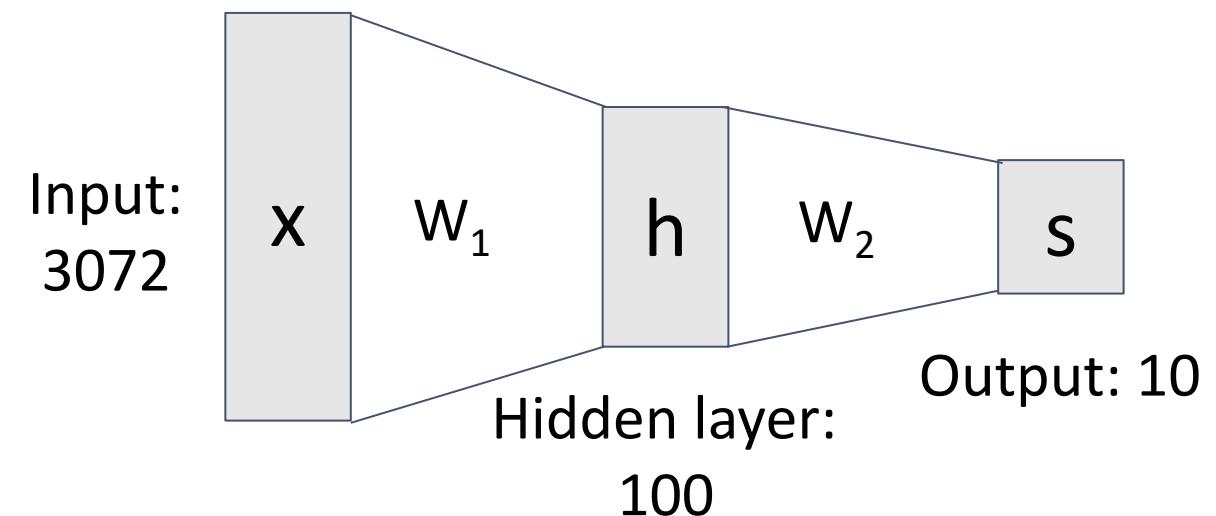
Neural Networks

Can use different templates to cover multiple modes of a class!



(Before) Linear score function:

(Now) 2-layer Neural Network



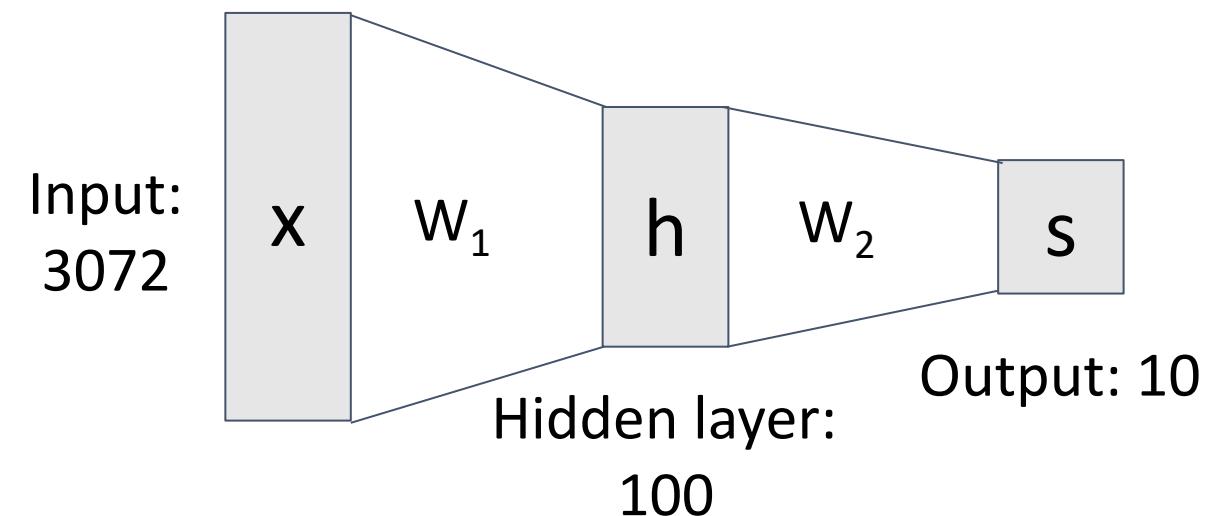
$$x \in \mathbb{R}^D, W_1 \in \mathbb{R}^{H \times D}, W_2 \in \mathbb{R}^{C \times H}$$

Neural Networks

“Distributed representation”:
Most templates not interpretable!

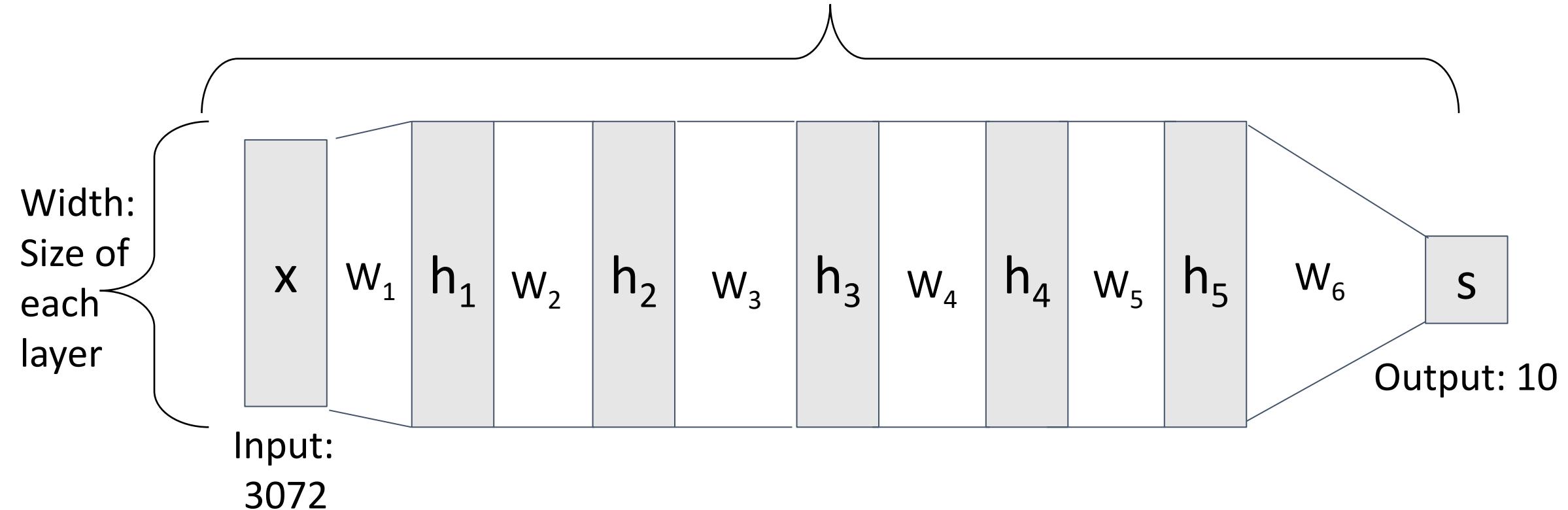


(Before) Linear score function:
(Now) 2-layer Neural Network



Deep Neural Networks

Depth = number of layers



$$s = W_6 \max(0, W_6 \max(0, W_5 \max(0, W_4 \max(0, W_3 \max(0, W_2 \max(0, W_1 x))))))$$

Neural Networks

- The area of Neural Networks has originally been primarily inspired by the goal of modeling biological neural systems.
- But has since diverged and become a matter of engineering and achieving good results in Machine Learning tasks.
- Neural Networks are modeled as collections of neurons that are connected in an acyclic graph. In other words, the outputs of some neurons can become inputs to other neurons.

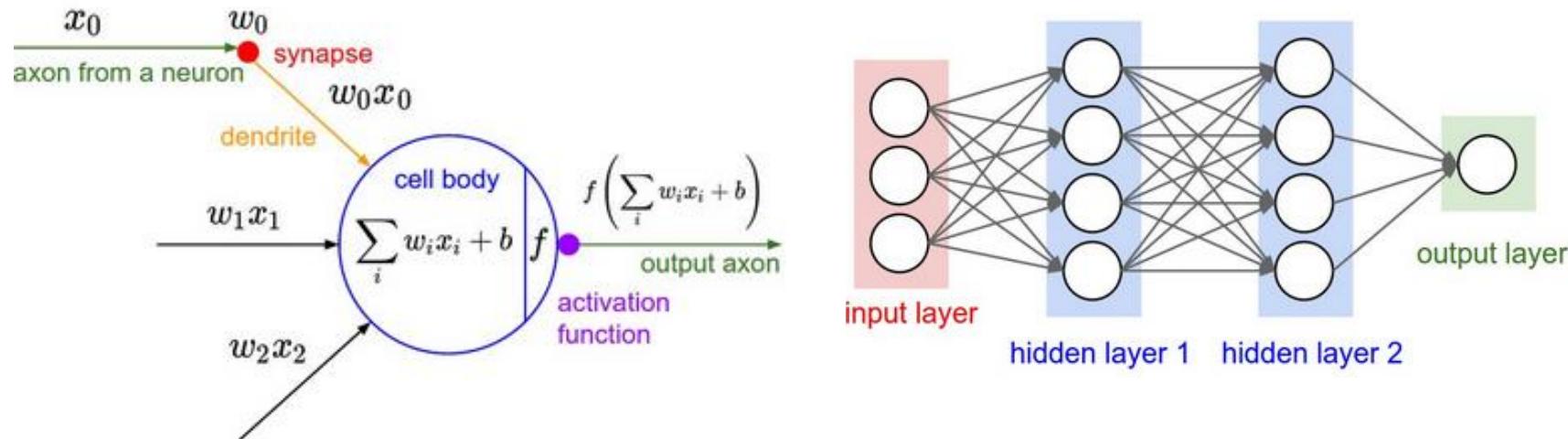
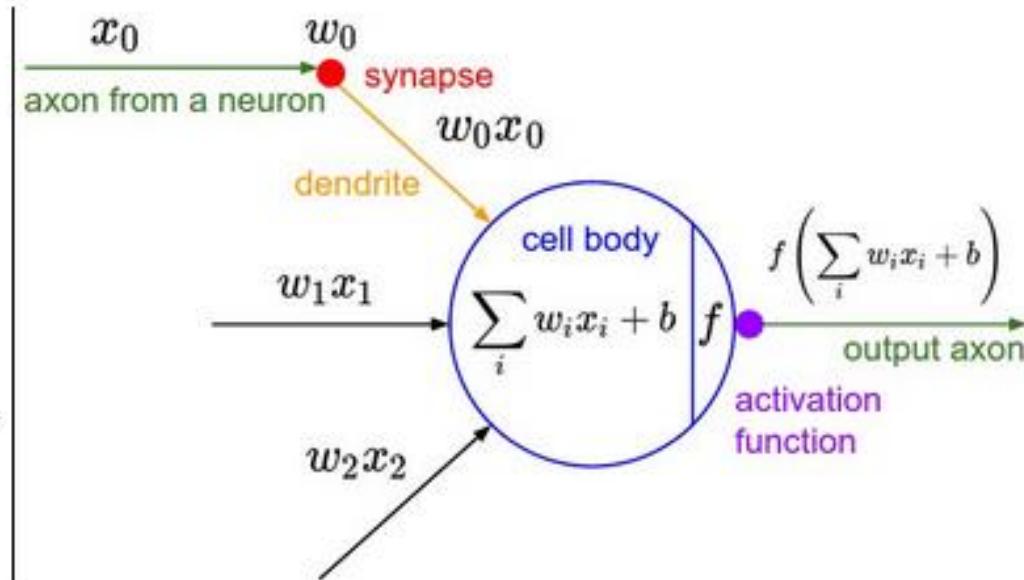
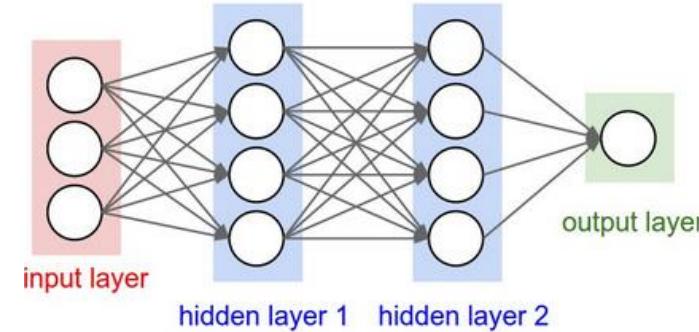
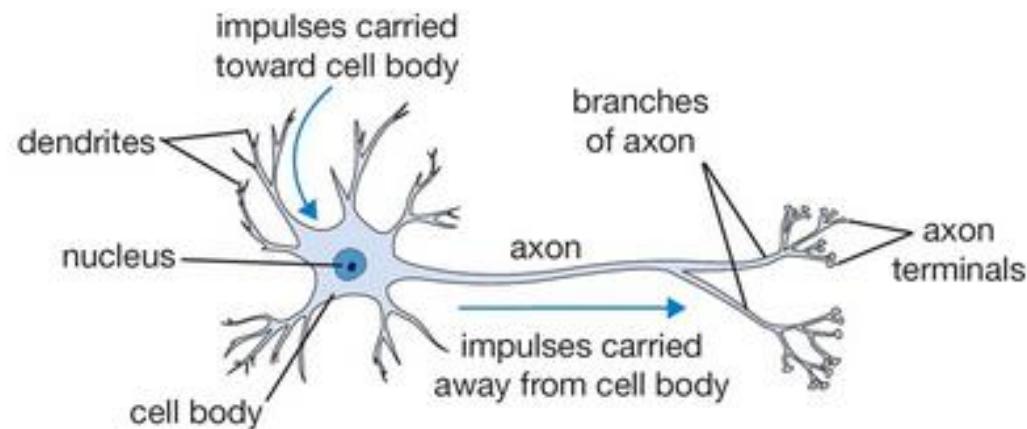


Figure from: <http://cs231n.github.io/neural-networks-1/>

Neural Networks



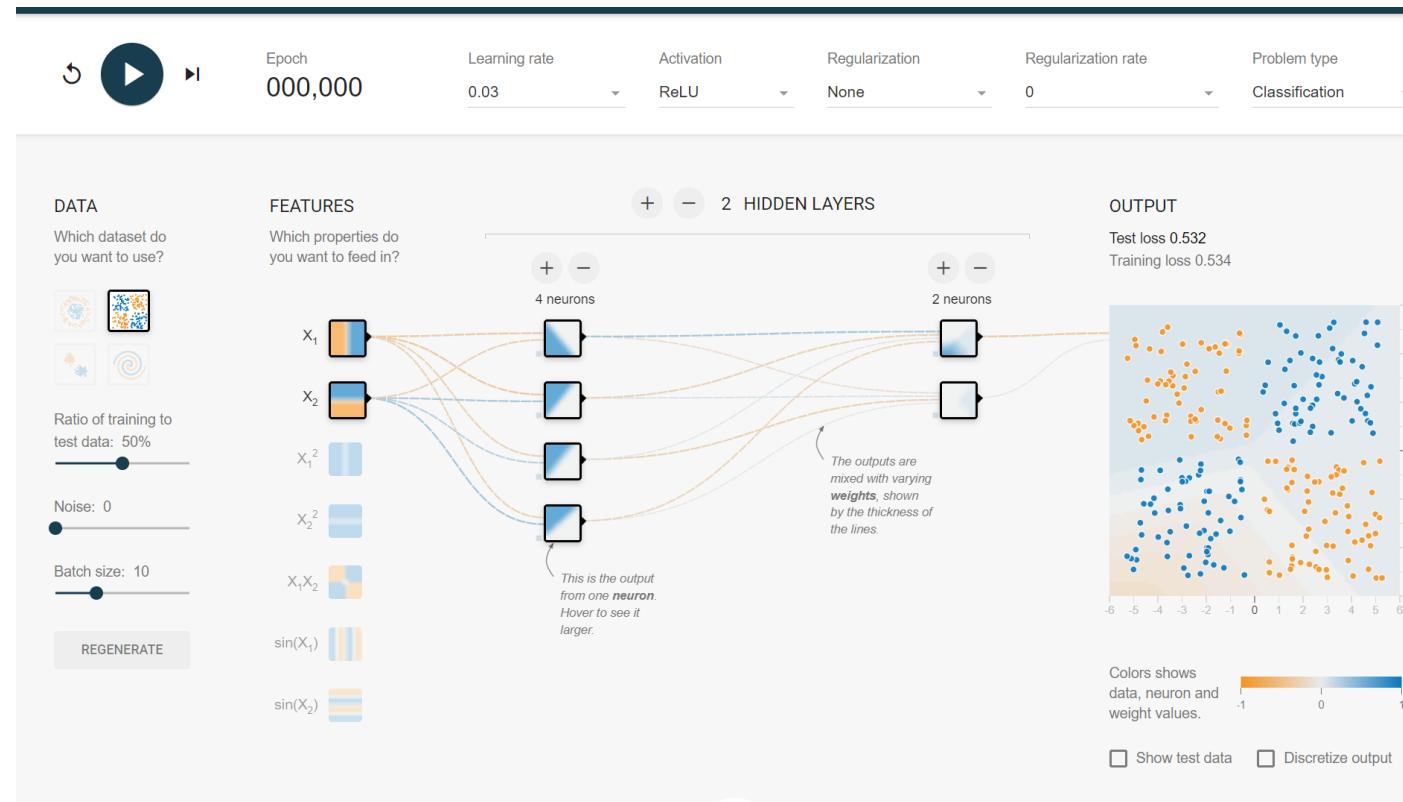
A cartoon drawing of a biological neuron (left) and its mathematical model (right).

Figure from: <http://cs231n.github.io/neural-networks-1/>

Tinker With a Neural Network in Your Browser

<http://playground.tensorflow.org/>

- Trying different architectures in a systematic way for the four given input patterns.
- Related work: Ablation study for deep learning where you systematically remove parts of the input to see which parts of the input are relevant to the networks output.
- Meyers, Richard, Melanie Lu, Constantin Waubert de Puiseau, and Tobias Meisen. "Ablation Studies in Artificial Neural Networks." *arXiv preprint arXiv:1901.08644* (2019).



<https://www.asimovinstitute.org/neural-network-zoo/>

- Input Cell
- Backfed Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Gated Memory Cell
- Kernel
- Convolution or Pool

A mostly complete chart of
Neural Networks

©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org

Perceptron (P)



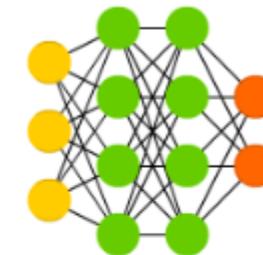
Feed Forward (FF)



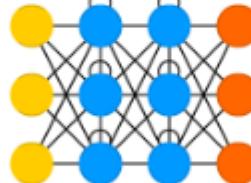
Radial Basis Network (RBF)



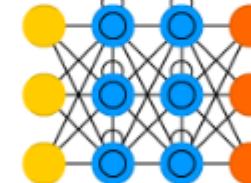
Deep Feed Forward (DFF)



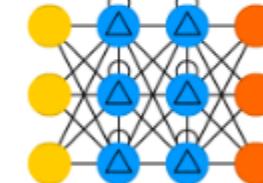
Recurrent Neural Network (RNN)



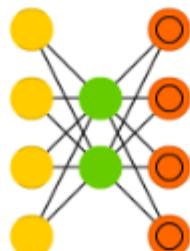
Long / Short Term Memory (LSTM)



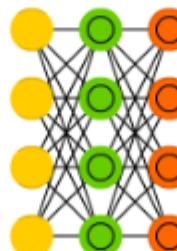
Gated Recurrent Unit (GRU)



Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)

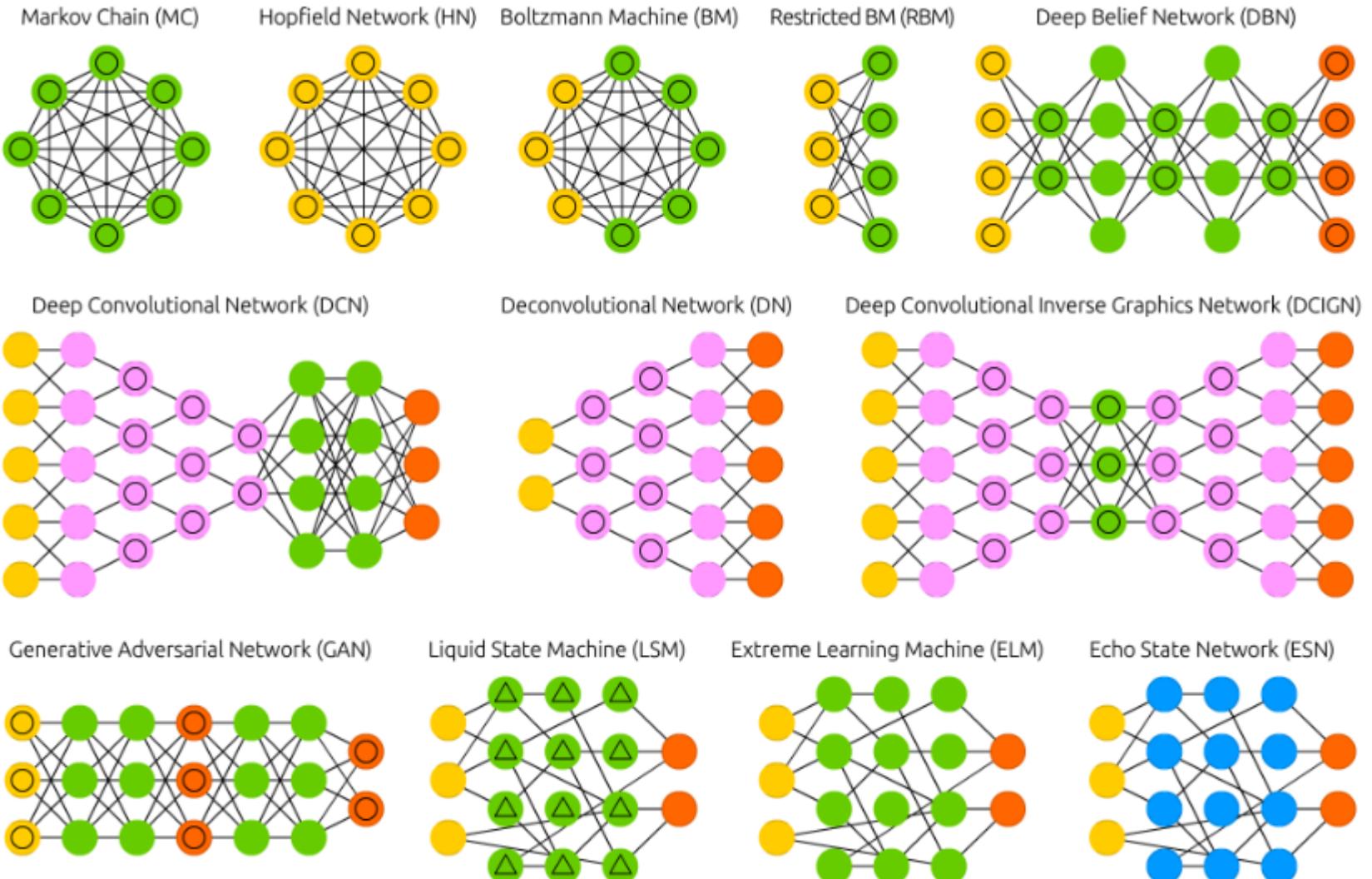


Sparse AE (SAE)



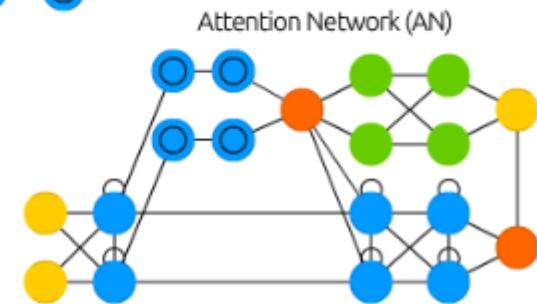
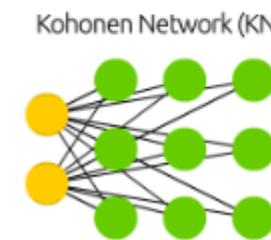
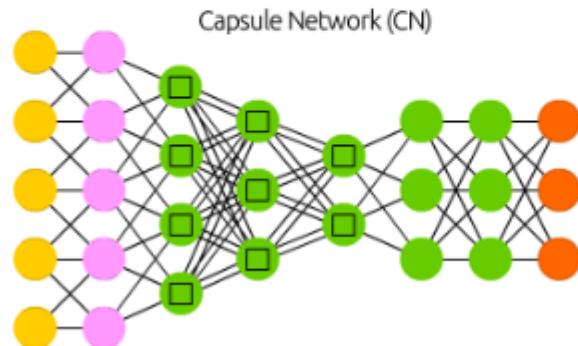
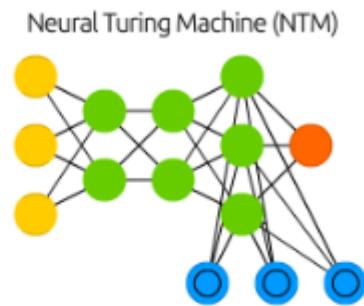
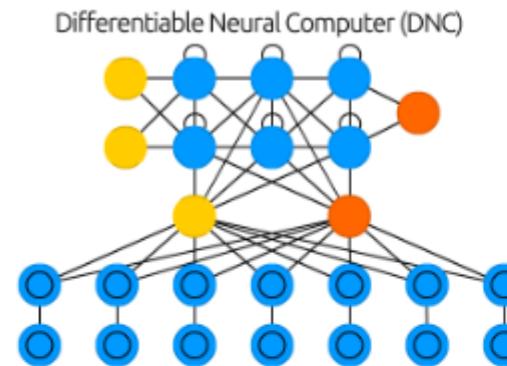
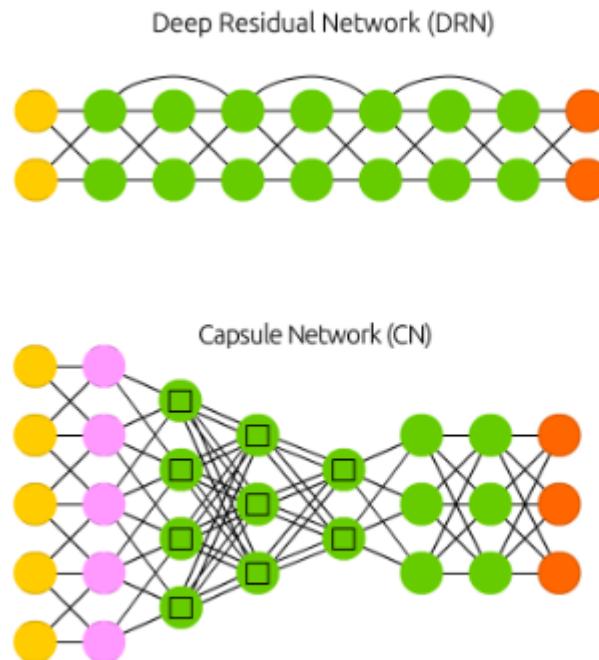
<https://www.asimovinstitute.org/neural-network-zoo/>

- Input Cell
- Backfed Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Capsule Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Gated Memory Cell
- Kernel
- Convolution or Pool



<https://www.asimovinstitute.org/neural-network-zoo/>

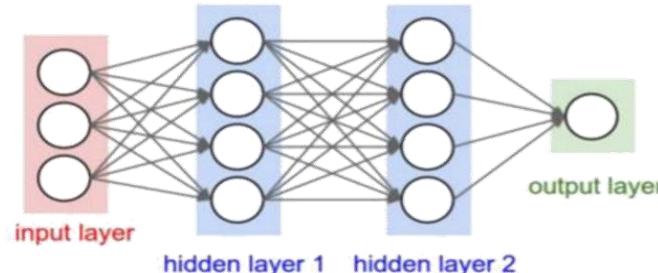
- (●) Input Cell
- (○) Backfed Input Cell
- (△) Noisy Input Cell
- (●) Hidden Cell
- (○) Probabilistic Hidden Cell
- (△) Spiking Hidden Cell
- (□) Capsule Cell
- (●) Output Cell
- (○) Match Input Output Cell
- (●) Recurrent Cell
- (○) Memory Cell
- (△) Gated Memory Cell
- (●) Kernel
- (○) Convolution or Pool



Neural Networks vs. Convolutional Neural Networks (CNN)

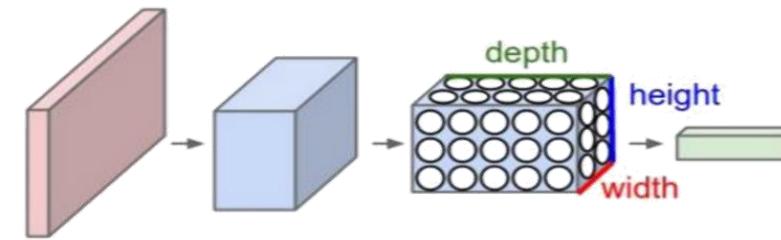
Regular Neural Nets

- Input: 1D vector
- Neurons fully connected
- Layers: input, output, one or two hidden layers



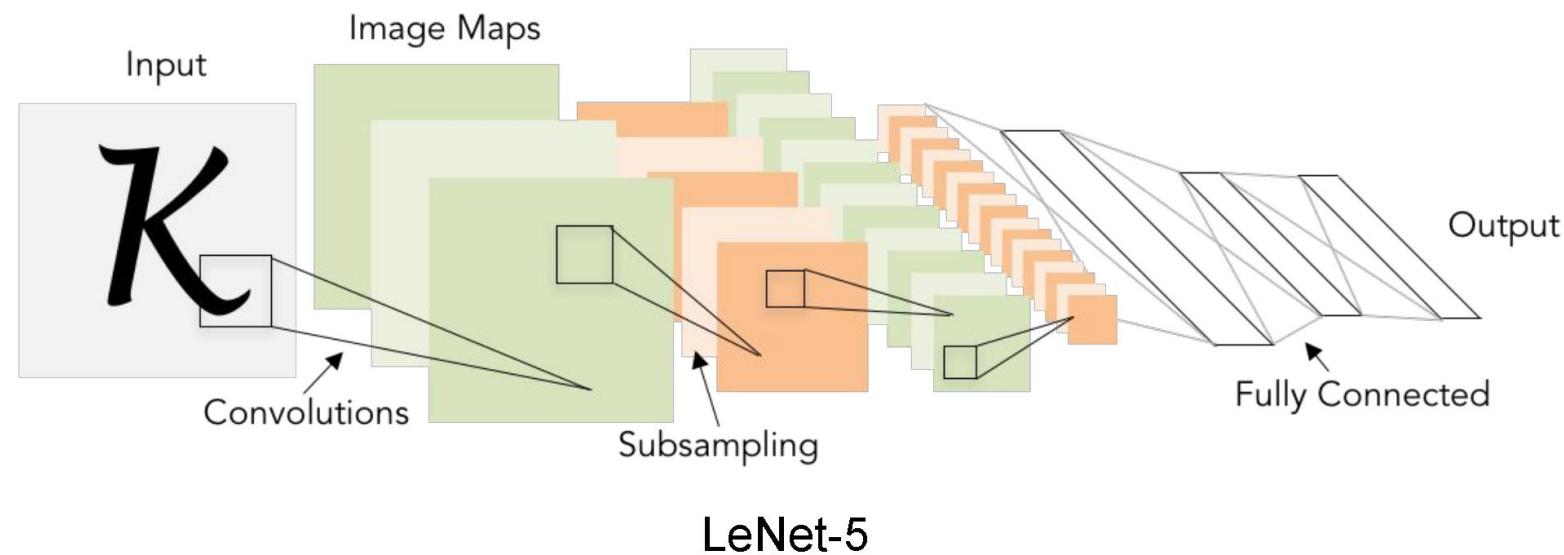
Convolutional Neural Nets

- Input: 3D volume
- Neurons are locally connected
- Higher number of hidden layers



A bit of history: Gradient-based learning applied to document recognition *[LeCun, Bottou, Bengio, Haffner 1998]*

Proceedings of the IEEE, 53K citations



Slide Credit: N. Snavely

A bit of history: ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]

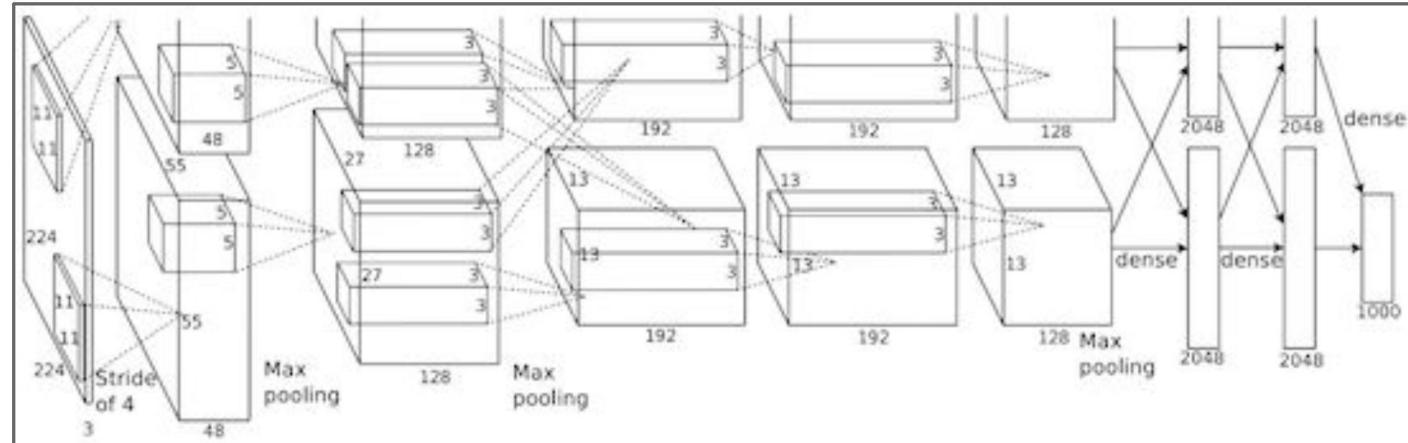
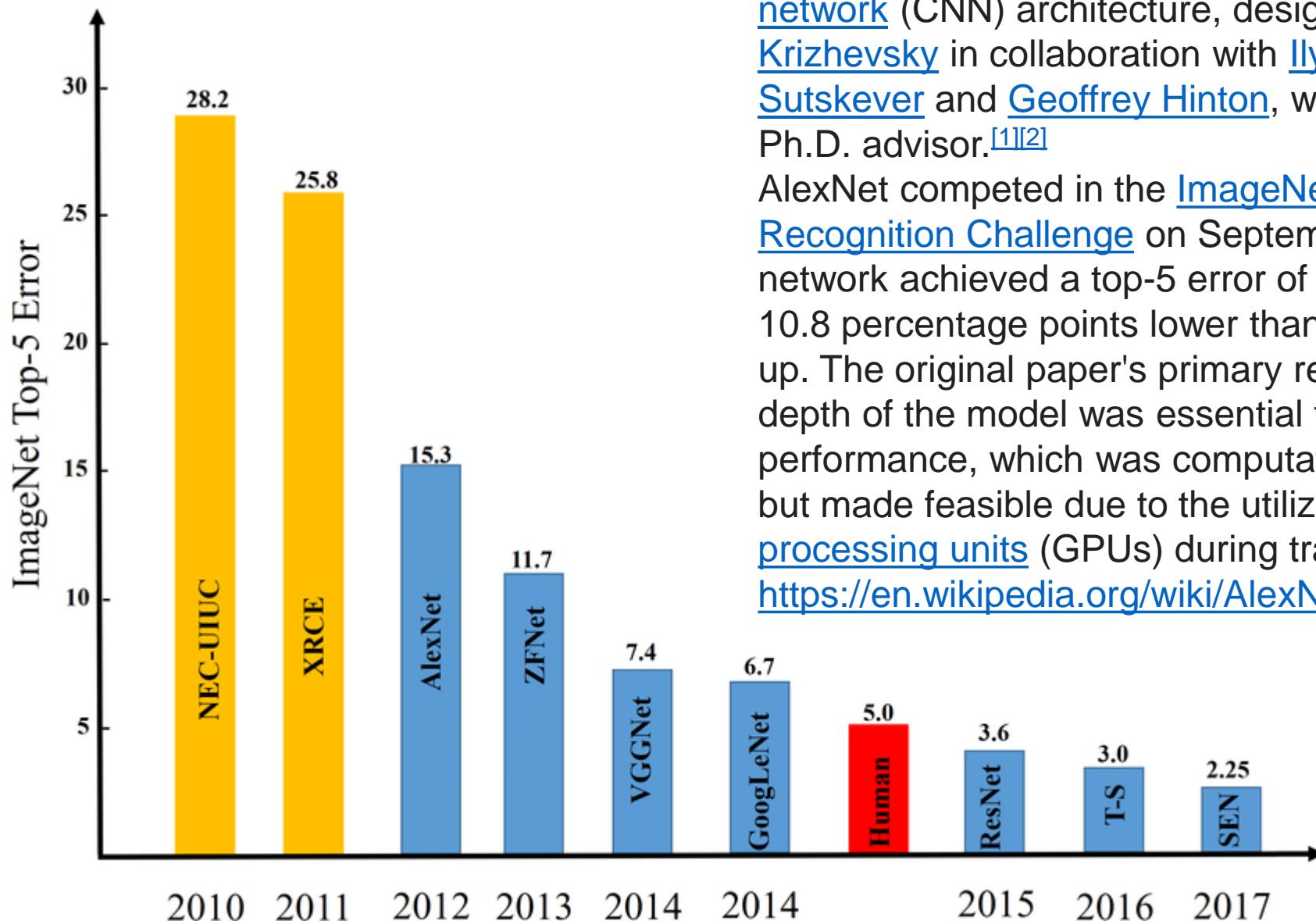


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

“AlexNet”

Slide Credit: N. Snavely



AlexNet is the name of a [convolutional neural network](#) (CNN) architecture, designed by [Alex Krizhevsky](#) in collaboration with [Ilya Sutskever](#) and [Geoffrey Hinton](#), who was Krizhevsky's Ph.D. advisor.^{[1][2]}

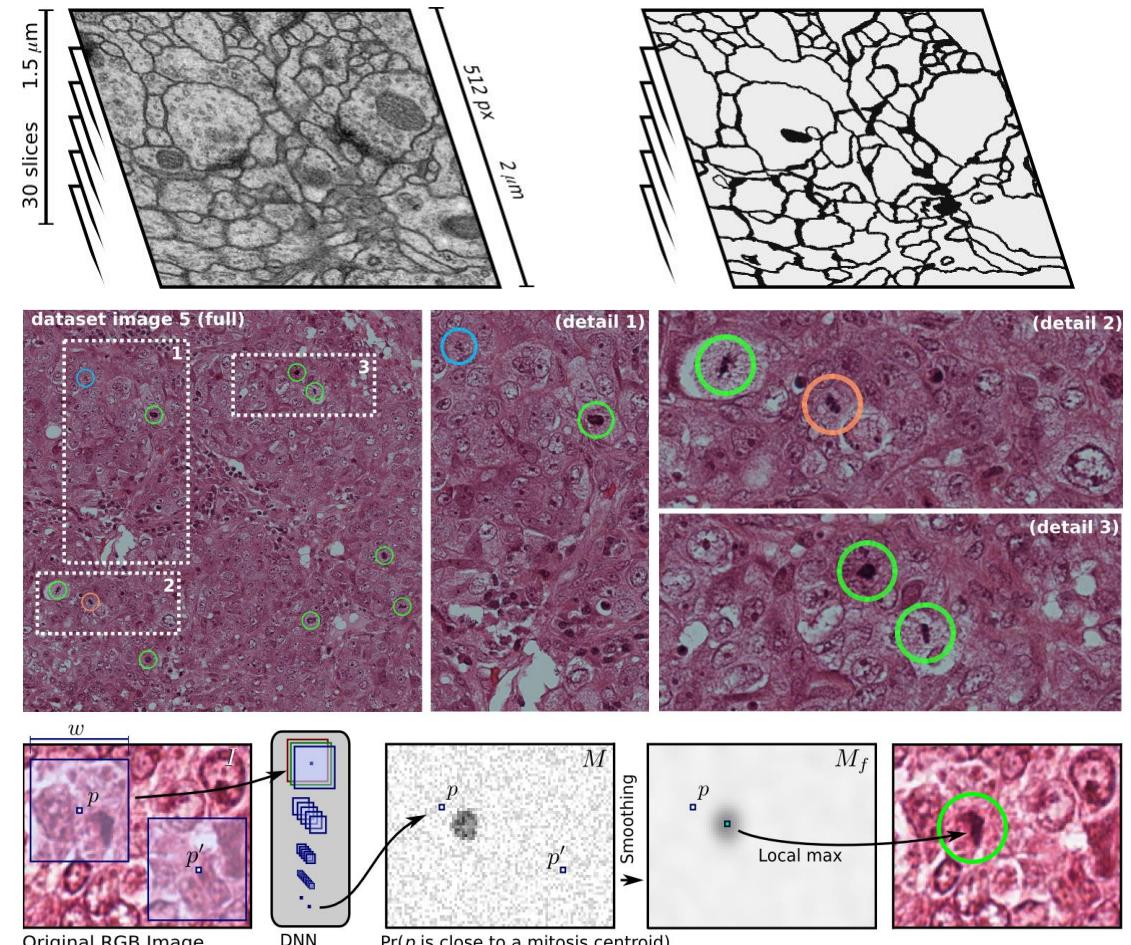
AlexNet competed in the [ImageNet Large Scale Visual Recognition Challenge](#) on September 30, 2012.^[3] The network achieved a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up. The original paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of [graphics processing units](#) (GPUs) during training.^[2]

<https://en.wikipedia.org/wiki/AlexNet>

Deep Learning in Biomedical Image Analysis

Deep Learning approaches have recently been successfully used in biomedical image analysis tasks:

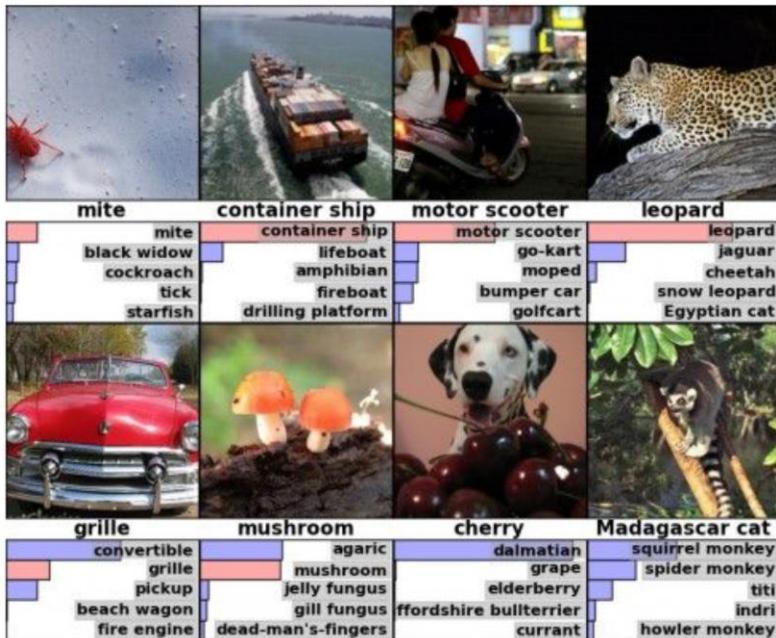
- Cell classification challenge [1]
- Mitosis detection challenge [2]
- Neuronal membrane segmentation challenge [3]



1. Zhimin Gao, Jianjia Zhang, Luping Zhou, and Lei Wang. HEp-2 cell image classification with convolutional neural networks. *IEEE Workshop Pattern Recognition Techniques for Indirect Immunofluorescence Images* 2014
2. C Ciresan, Alessandro Giusti, Luca M Gambardella, and Jurgen Schmidhuber. Mitosis detection in breast cancer histology images with deep neural networks. *Medical Image Computing & Computer-Assisted Intervention*, 2013.
3. Dan Ciresan, Alessandro Giusti, Luca M Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, 2012.

Fast-forward to today: ConvNets are everywhere

Classification

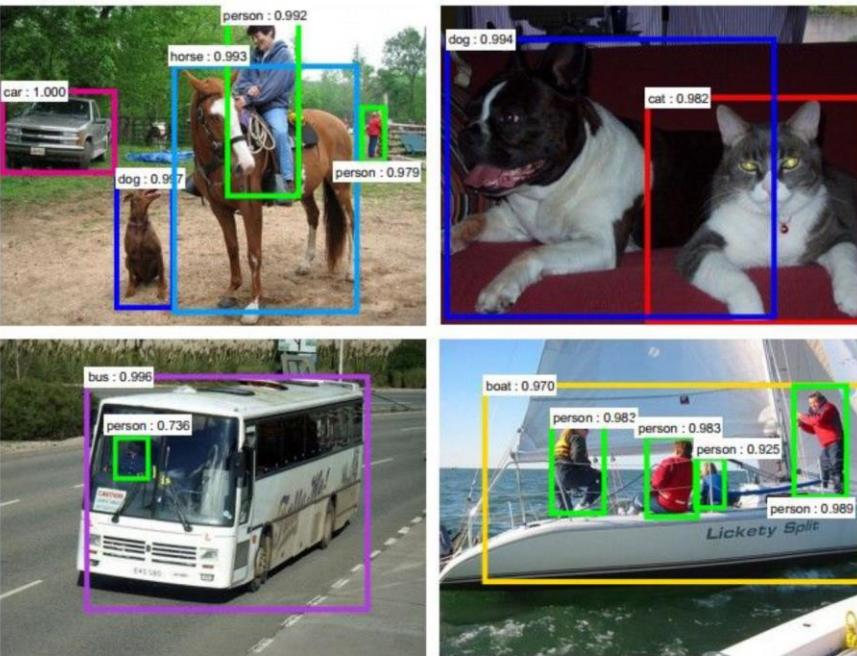


Retrieval



Fast-forward to today: ConvNets are everywhere

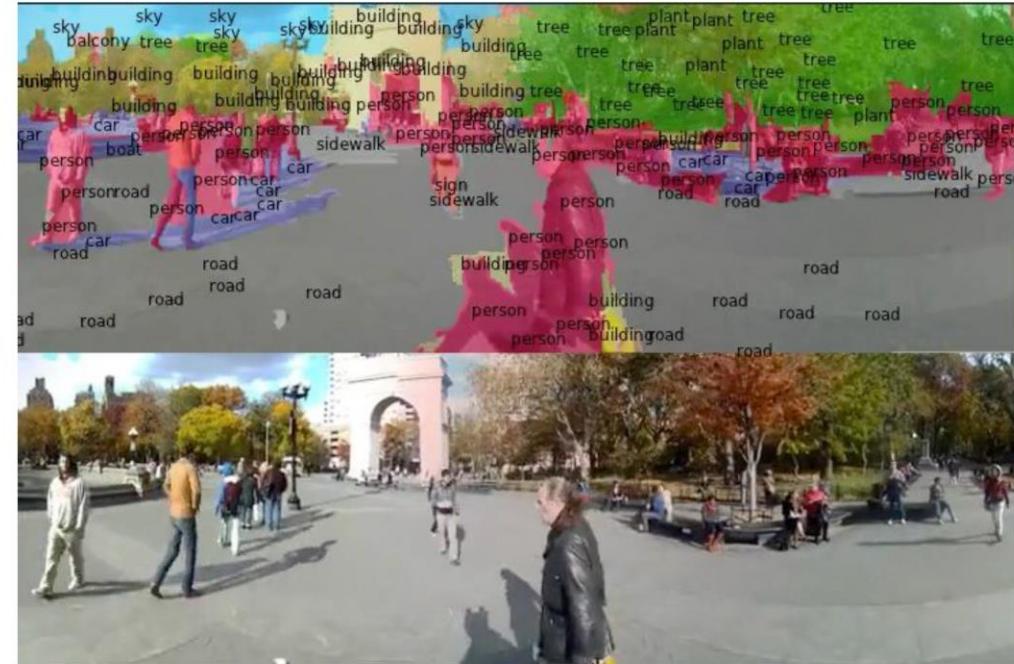
Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

Segmentation



Figures copyright Clement Farabet, 2012.
Reproduced with permission.

[Farabet et al., 2012]

Fast-forward to today: ConvNets are everywhere

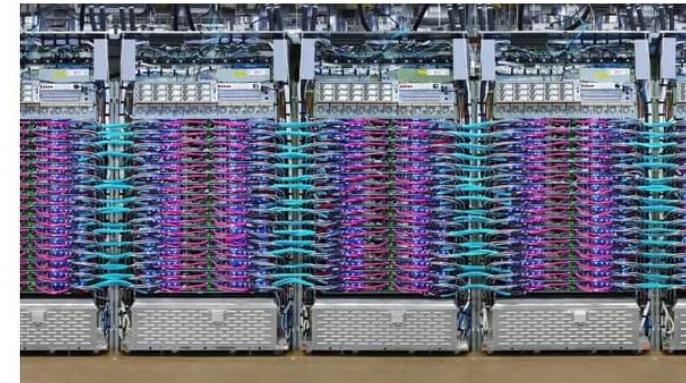


Self-driving cars (video courtesy Tesla)

<https://www.tesla.com/AI>



NVIDIA Tesla V100 GPU



Cloud TPU v3 Pod

100+ petaflops

<https://cloud.google.com/tpu/>

Slide Credit: N. Snavely

Caption-to-image

An astronaut Teddy bears A
bowl of soup

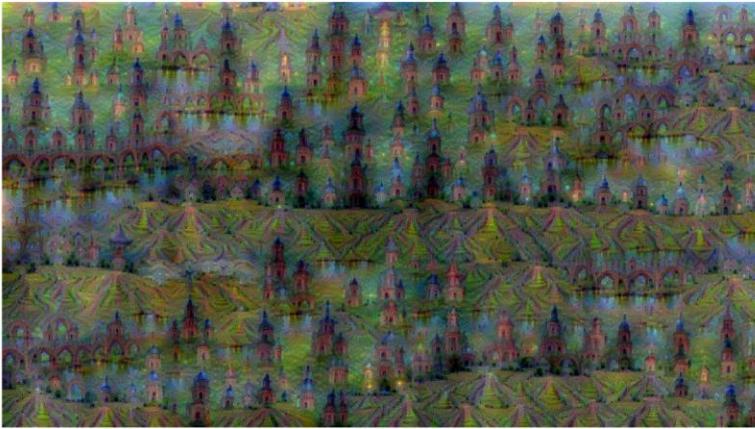
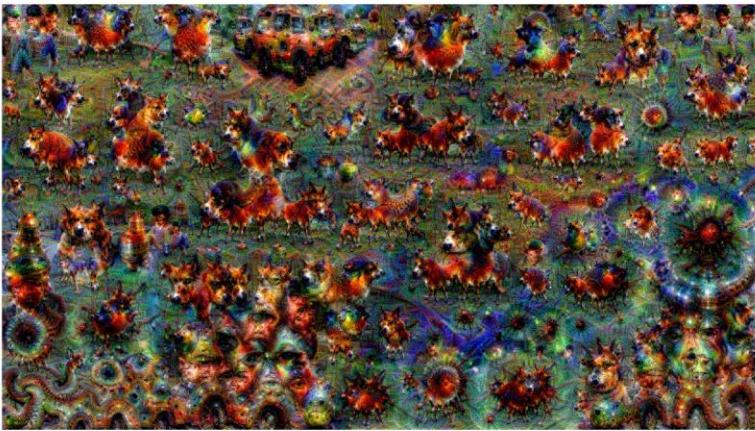
riding a horse lounging in a
tropical resort in space playing
basketball with cats in space

in a photorealistic style in the
style of Andy Warhol **as a pencil
drawing**

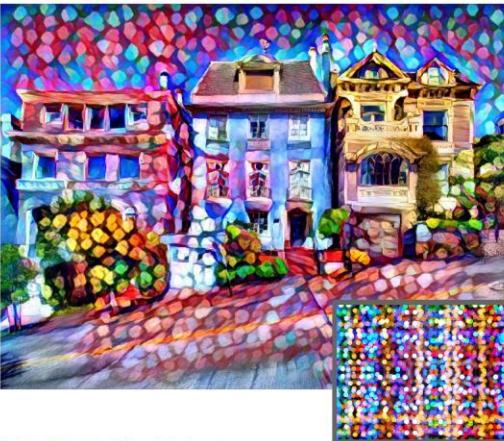


<https://openai.com/dall-e-2/>

Slide Credit: N. Snavely

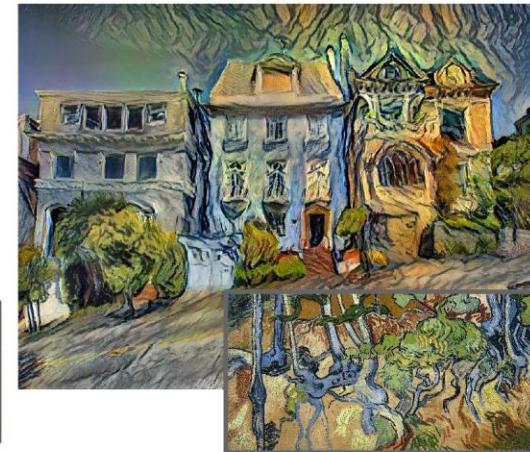
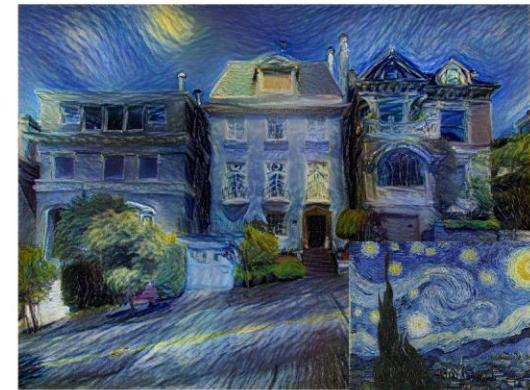


Figures copyright Justin Johnson, 2015. Reproduced with permission. Generated using the Inceptionism approach from a [blog post](#) by Google Research.



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain

Bokeh image is in the public domain
Stylized Images copyright Justin Johnson, 2017;
reproduced with permission



Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

Life Before Deep Learning

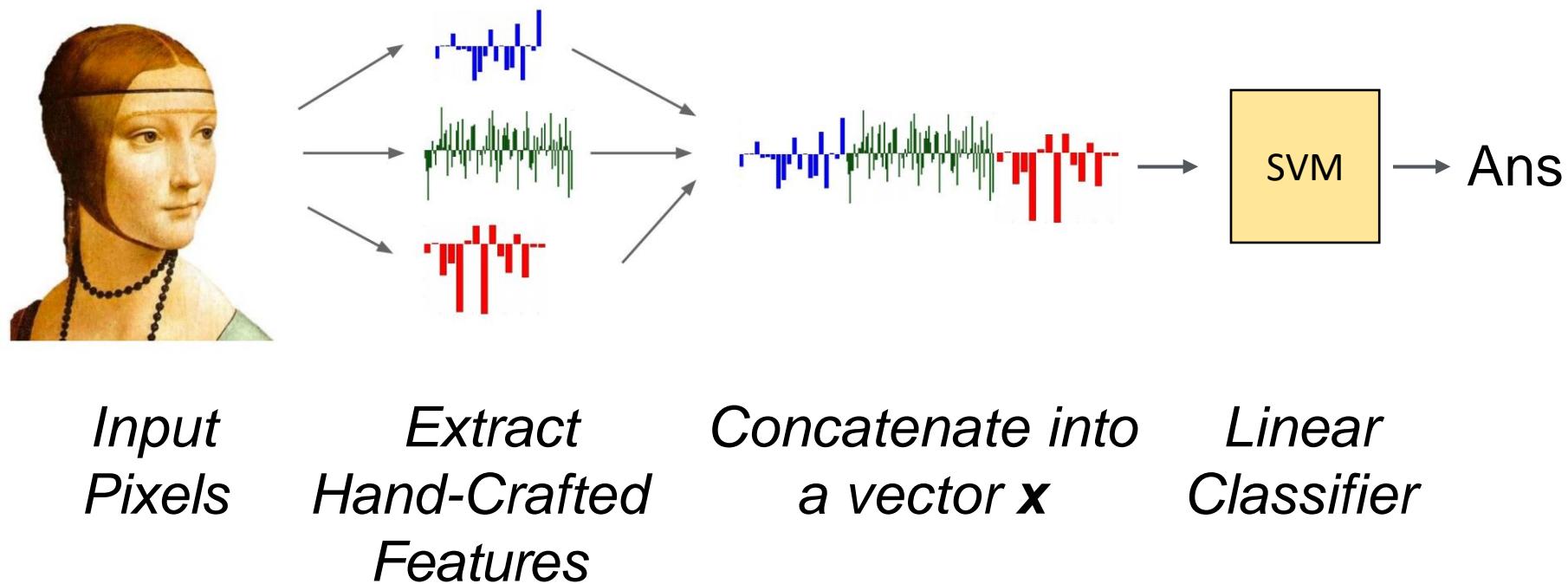
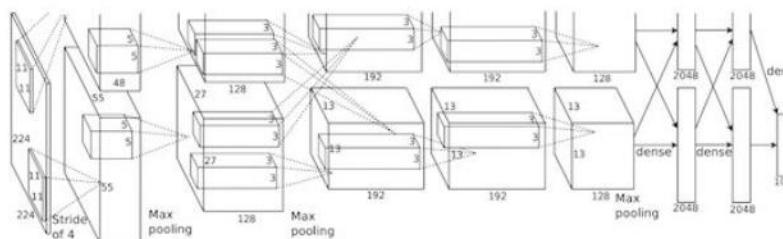
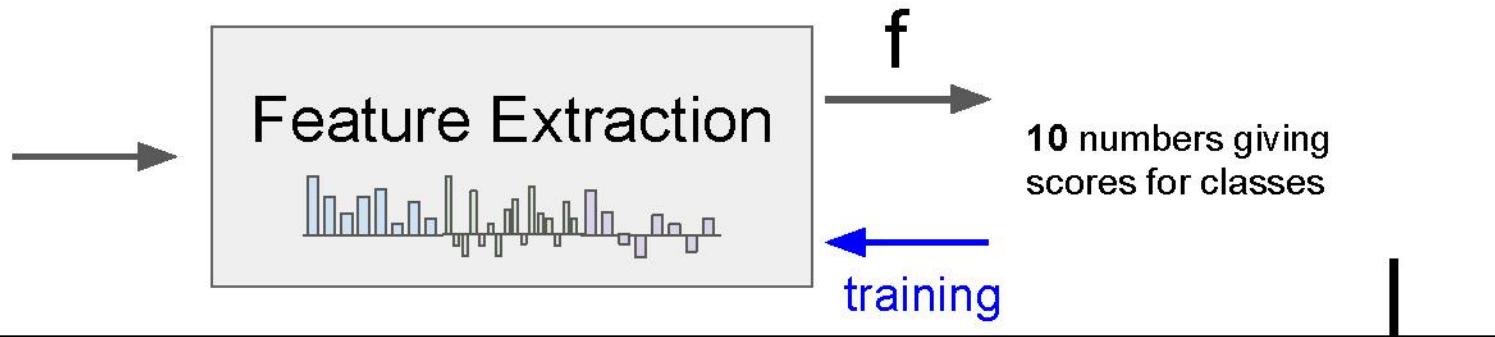


Figure: Karpathy 2016

Image features vs ConvNets



Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012.
Figure copyright Krizhevsky, Sutskever, and Hinton, 2012.
Reproduced with permission.



Brief Introduction to Convolutional Neural Networks (CNN)

CNN consists of three main types of layers:

1-Convolutional layers: core building block of a CNN.

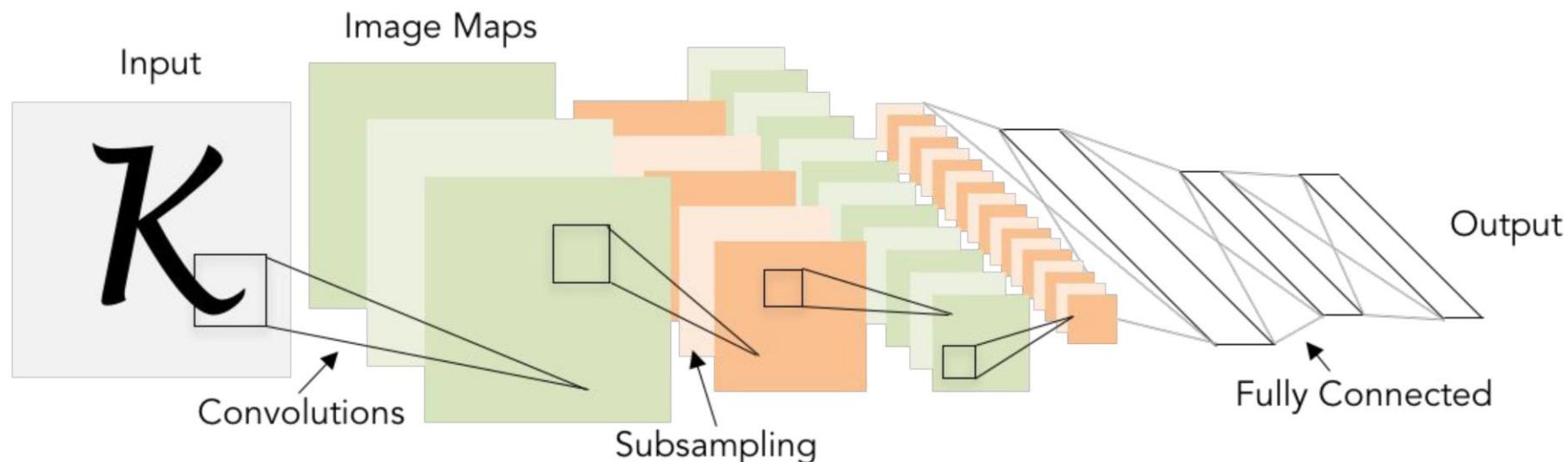
- a set of learnable filters (or kernels).
- each filter is convolved across the width and height of the input volume

2-Pooling layers:

- progressively reduce the spatial size of the representation to reduce the amount of parameters

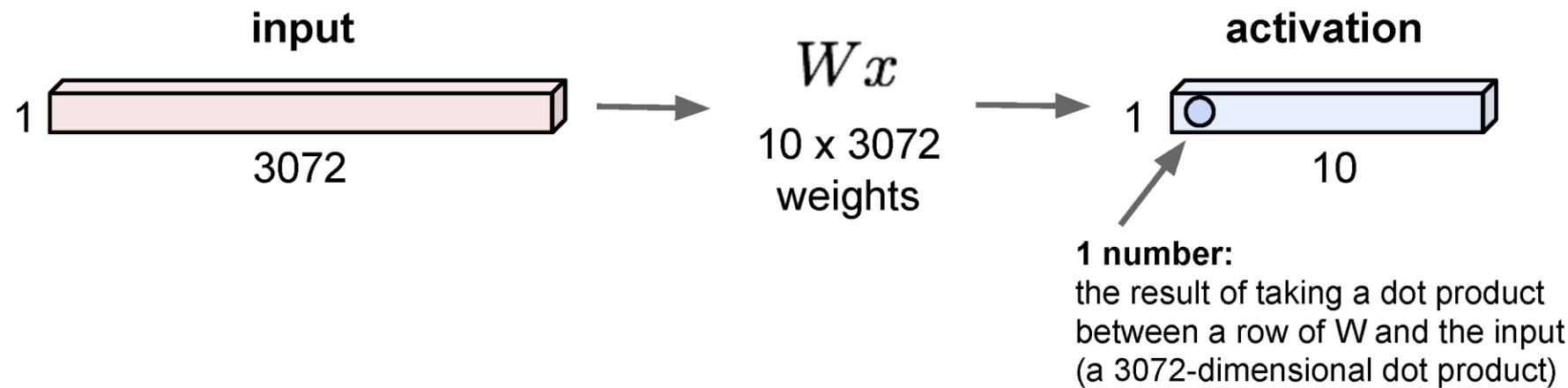
3-Fully connected layer:

- Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks.



Fully Connected Layer

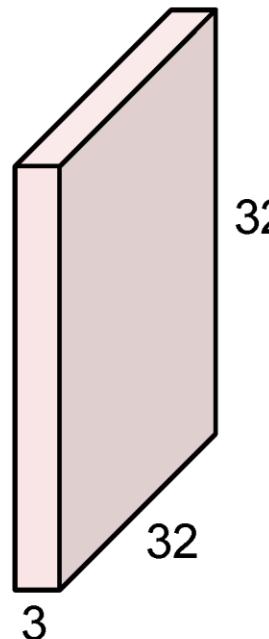
32x32x3 image -> stretch to 3072 x 1



Same as a linear classifier!

Convolution Layer

32x32x3 image



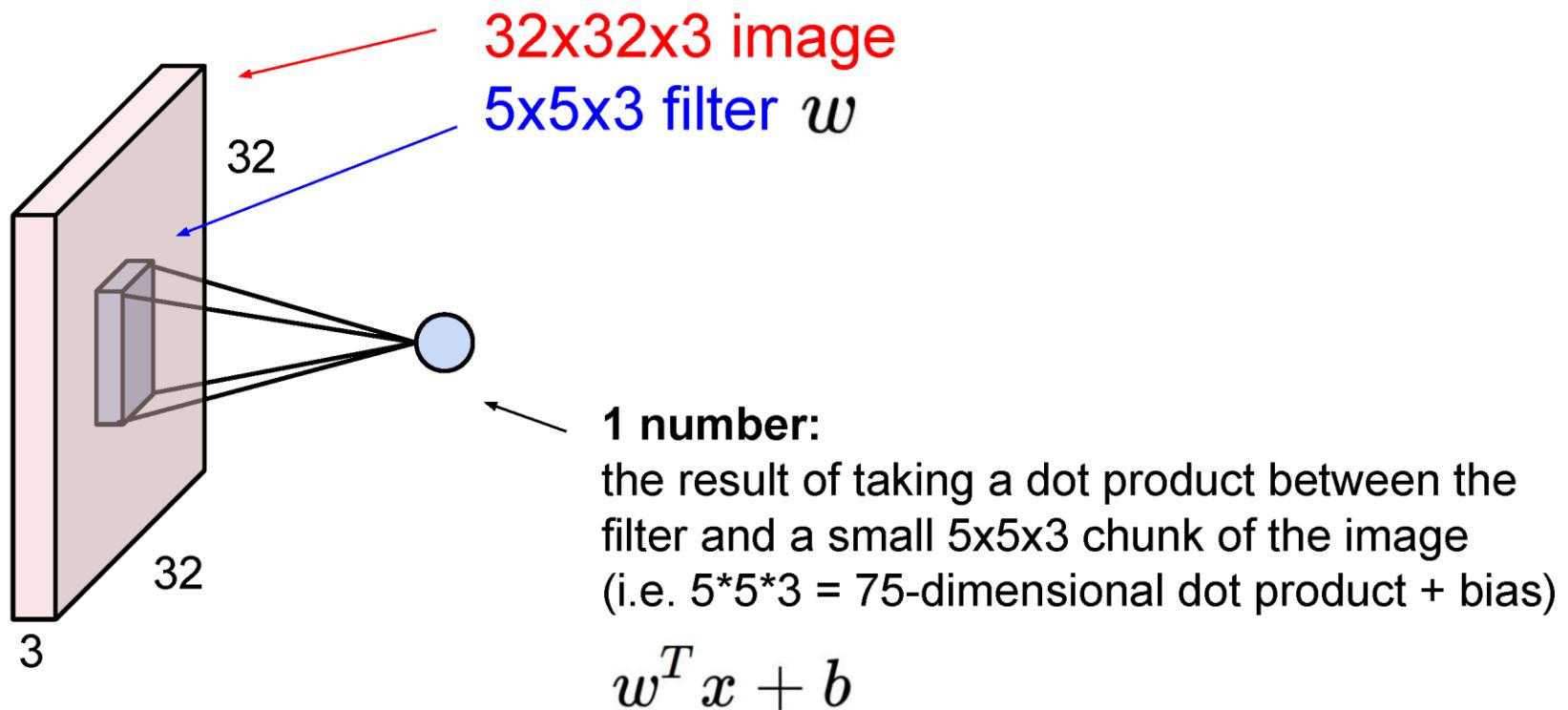
5x5x3 filter



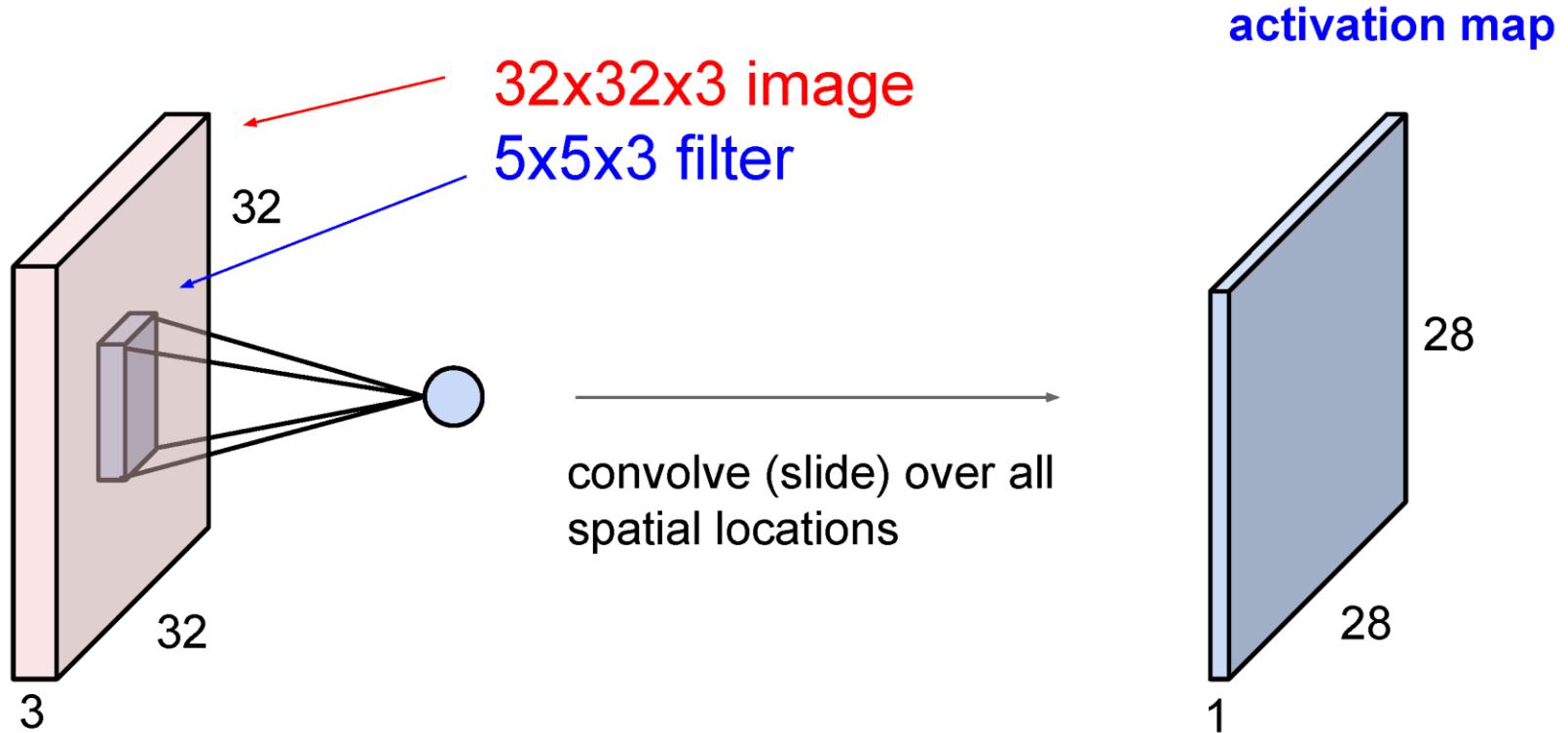
Filters always extend the full depth of the input volume

Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution Layer

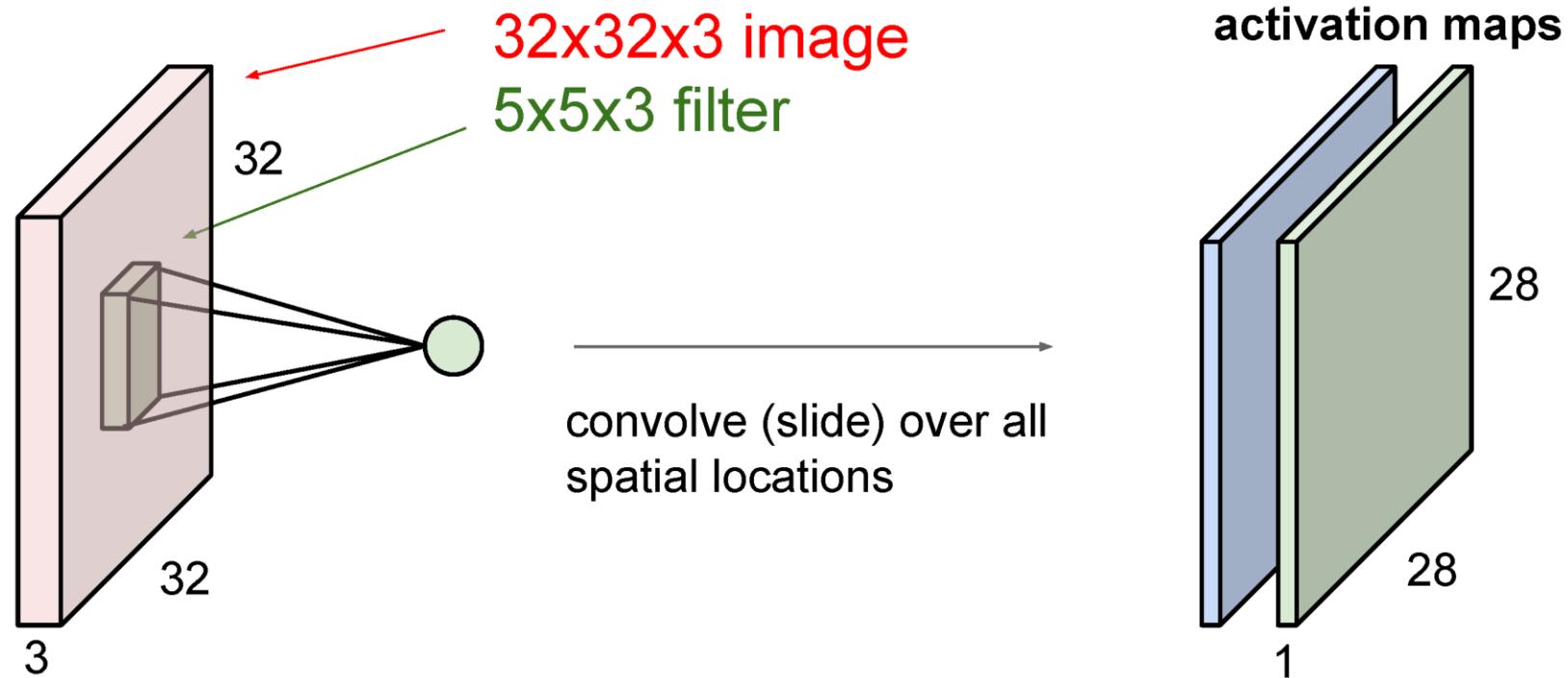


Convolution Layer

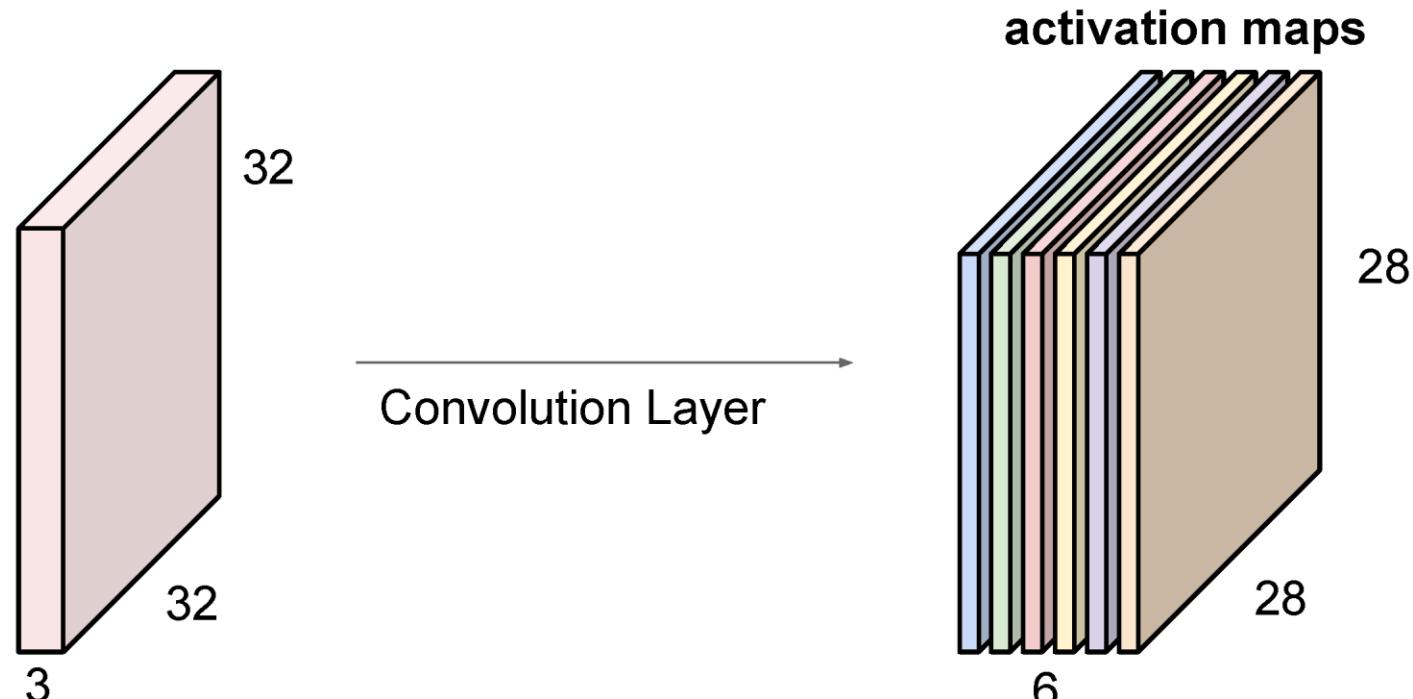


Convolution Layer

consider a second, **green** filter



For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

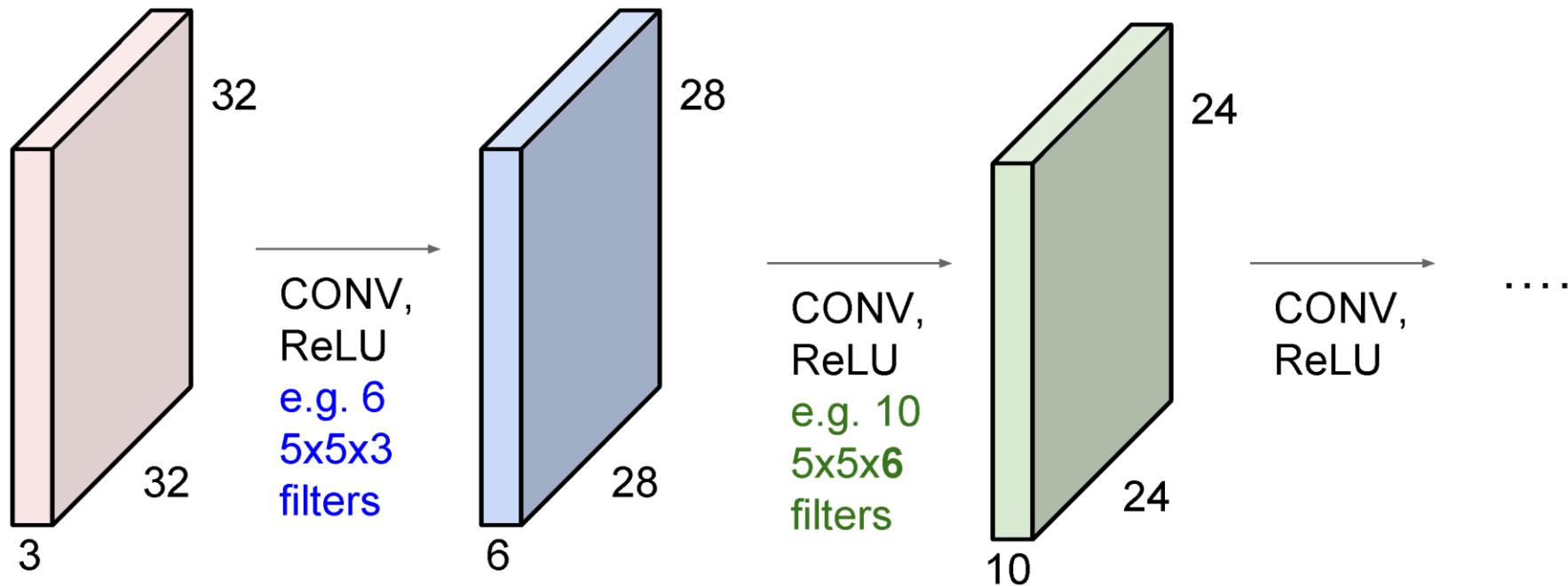


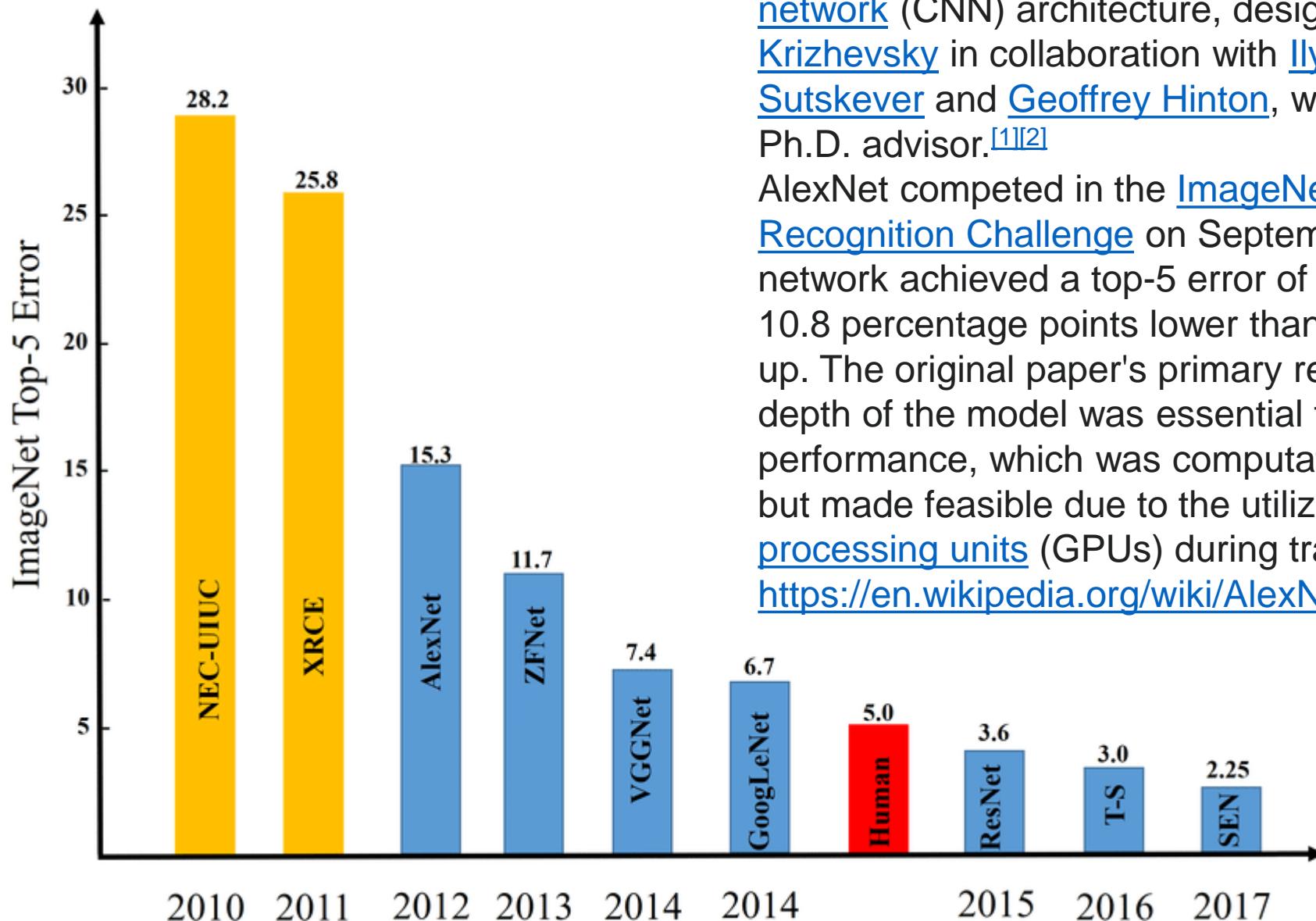
We stack these up to get a “new image” of size 28x28x6!

(total number of parameters: $6 \times (5 \times 5 \times 3 + 1) = 456$)

Slide Credit: N. Snavely

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



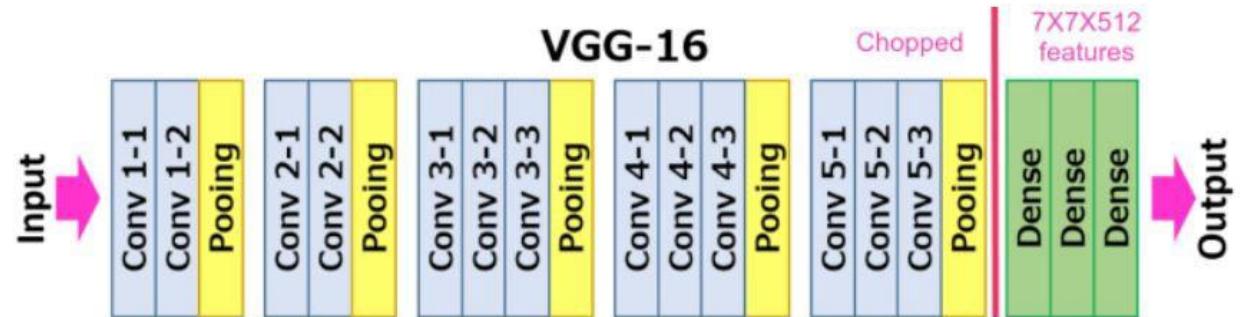


AlexNet is the name of a [convolutional neural network](#) (CNN) architecture, designed by [Alex Krizhevsky](#) in collaboration with [Ilya Sutskever](#) and [Geoffrey Hinton](#), who was Krizhevsky's Ph.D. advisor.^{[1][2]}

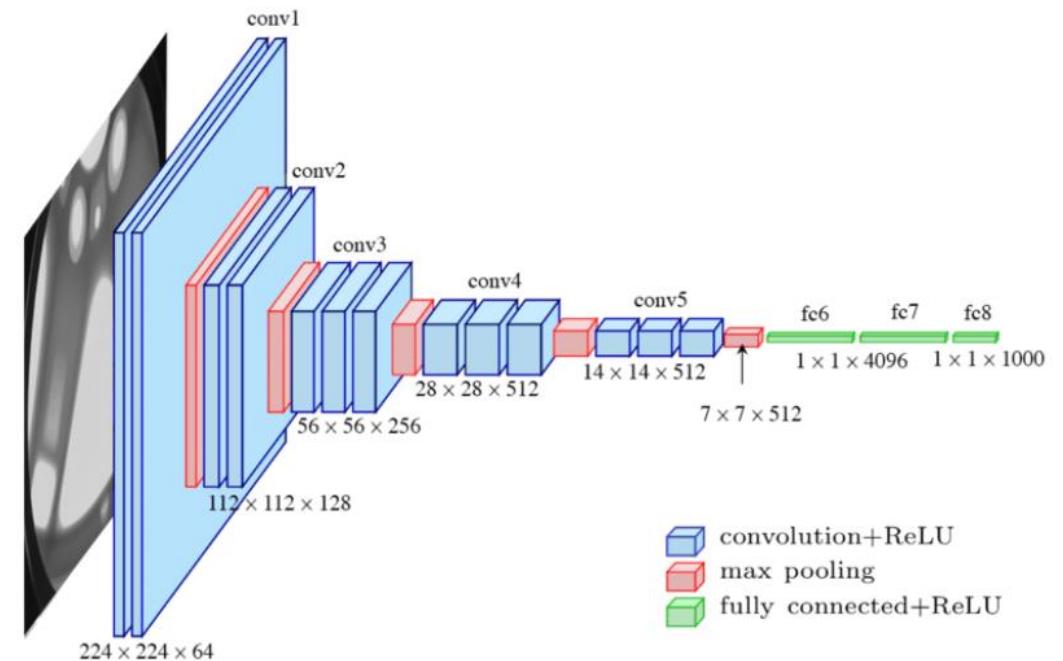
AlexNet competed in the [ImageNet Large Scale Visual Recognition Challenge](#) on September 30, 2012.^[3] The network achieved a top-5 error of 15.3%, more than 10.8 percentage points lower than that of the runner up. The original paper's primary result was that the depth of the model was essential for its high performance, which was computationally expensive, but made feasible due to the utilization of [graphics processing units](#) (GPUs) during training.^[2]

<https://en.wikipedia.org/wiki/AlexNet>

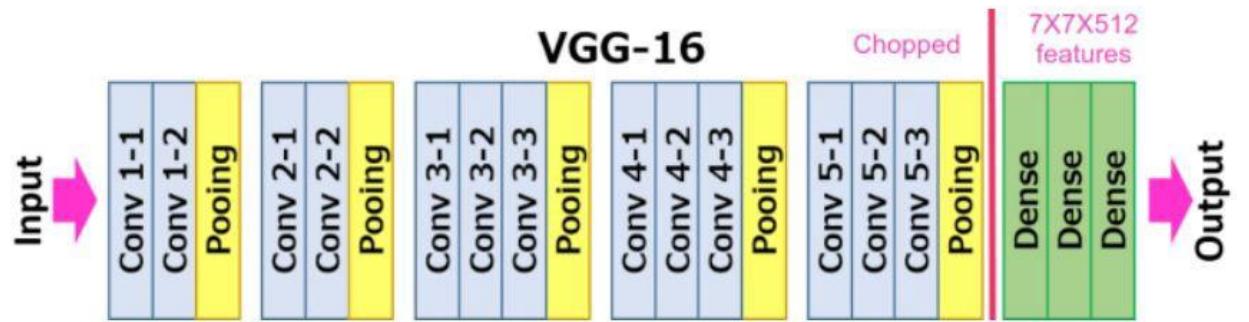
VGG 16



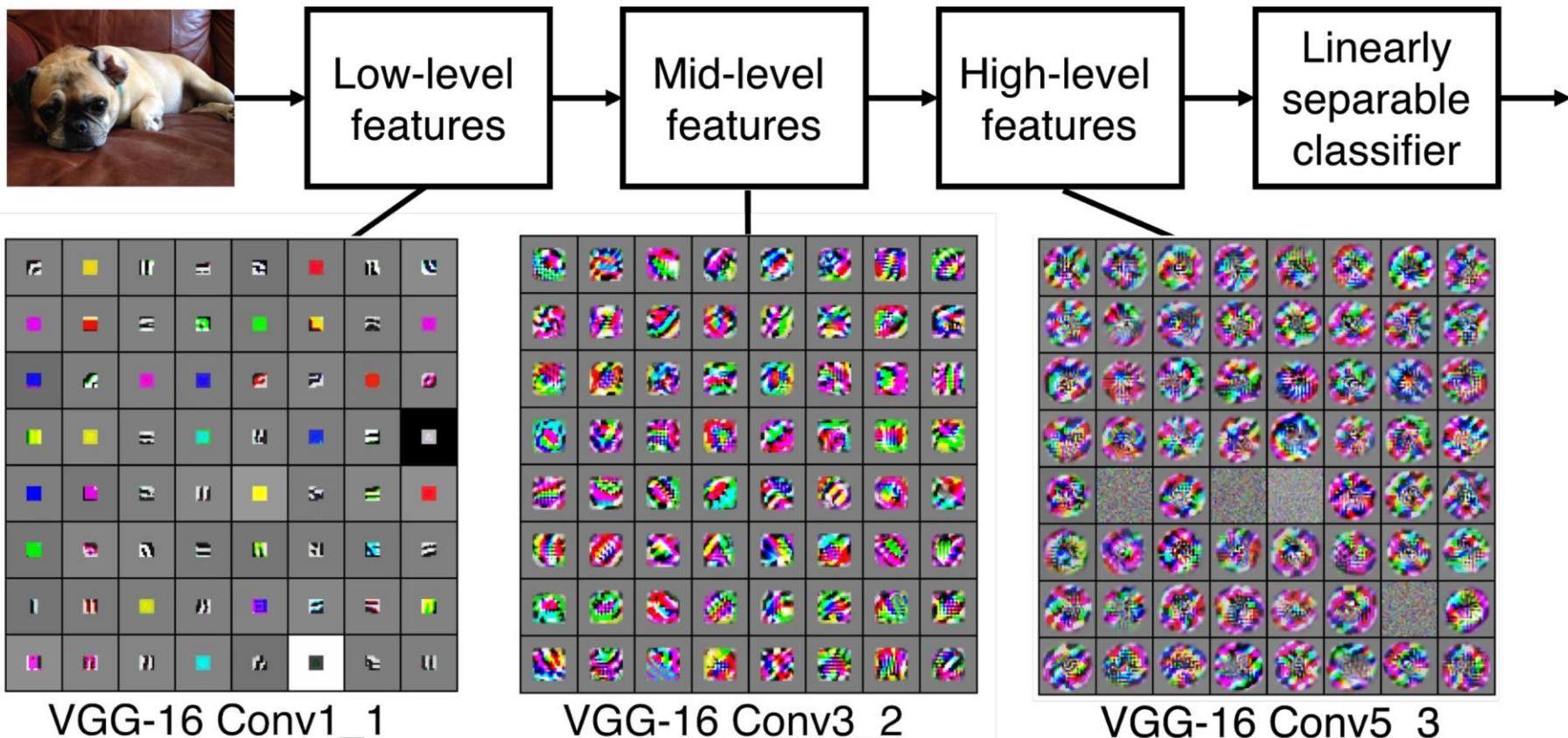
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014
- Karen Simonyan & Andrew Zisserman from Visual Geometry Group, Department of Engineering Science, University of Oxford
- VGG 16 won the 1st and 2nd place in object detection and classification.



Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*. – 97K citations

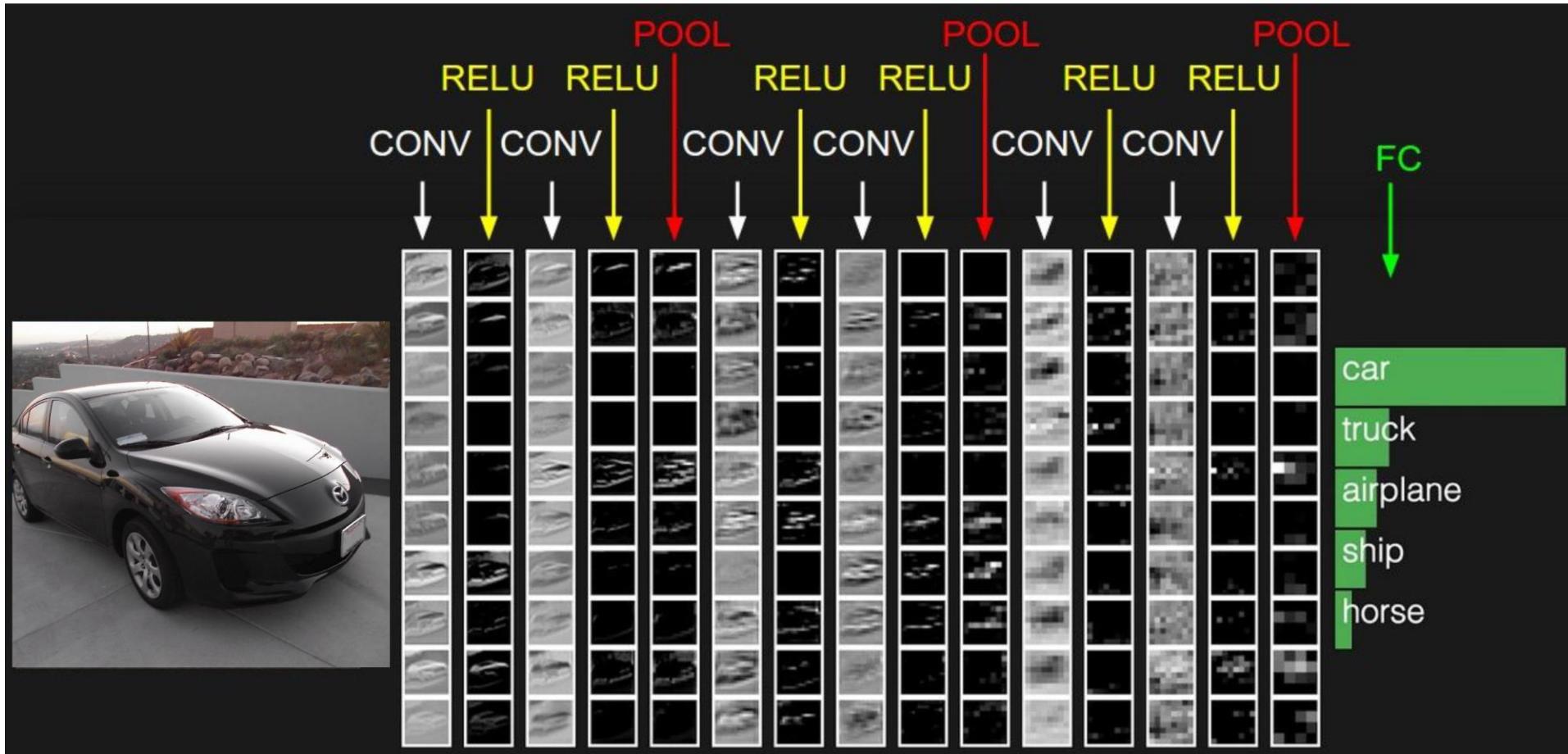


Preview



Slide Credit: N. Snavely

preview:



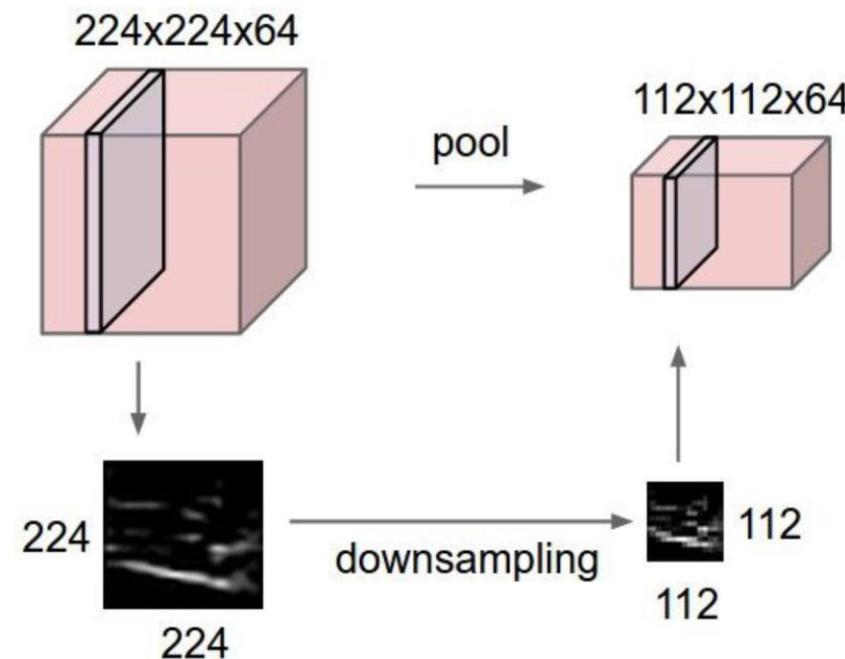
Slide Credit: N. Snavely

Convolutional layer—properties

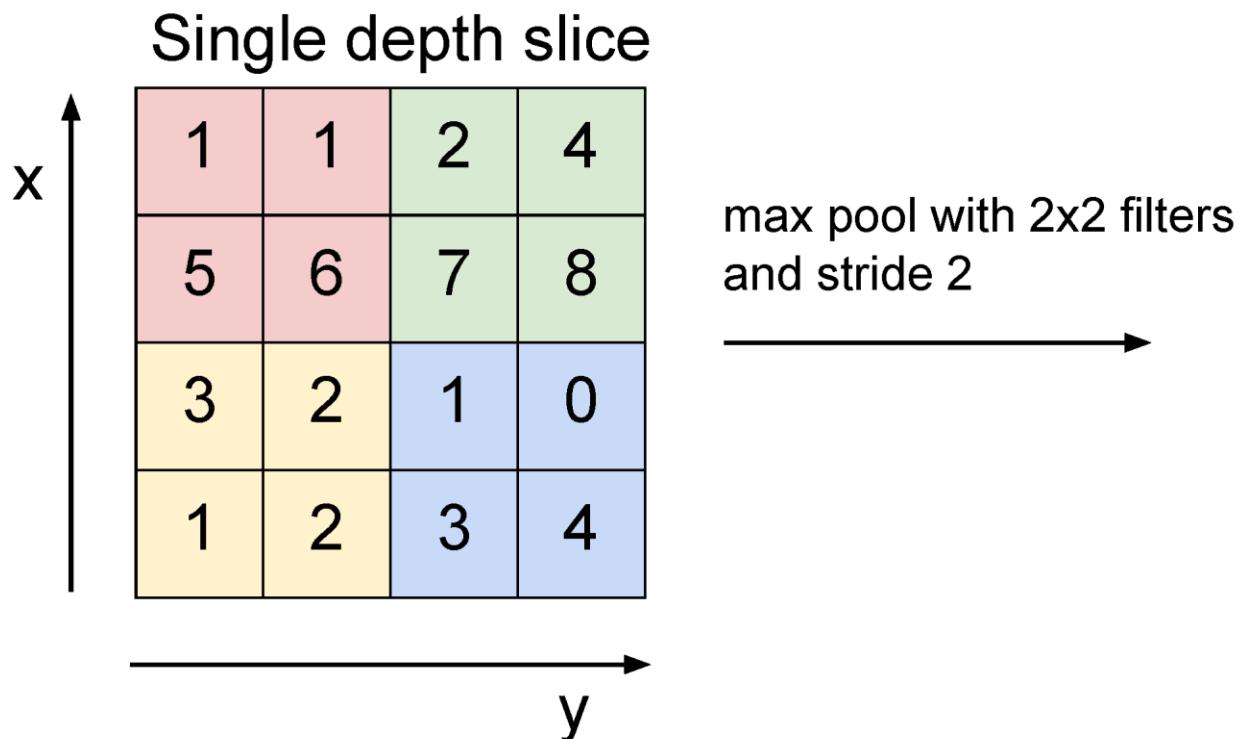
- Small number of parameters to learn compared to a fully connected layer
- Preserves spatial structure—output of a convolutional layer is shaped like an image
- **Translation equivariant:** passing a translated image through a convolutional layer is (almost) equivalent to translating the convolution output (but be careful of image boundaries)

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:

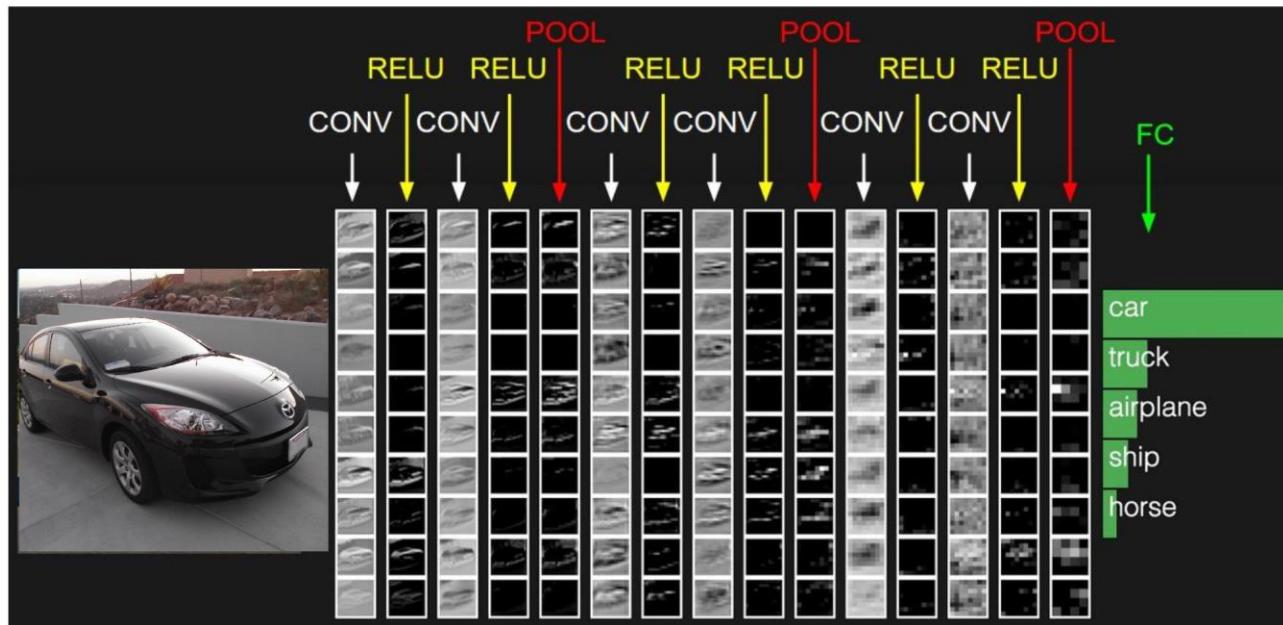


MAX POOLING



Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



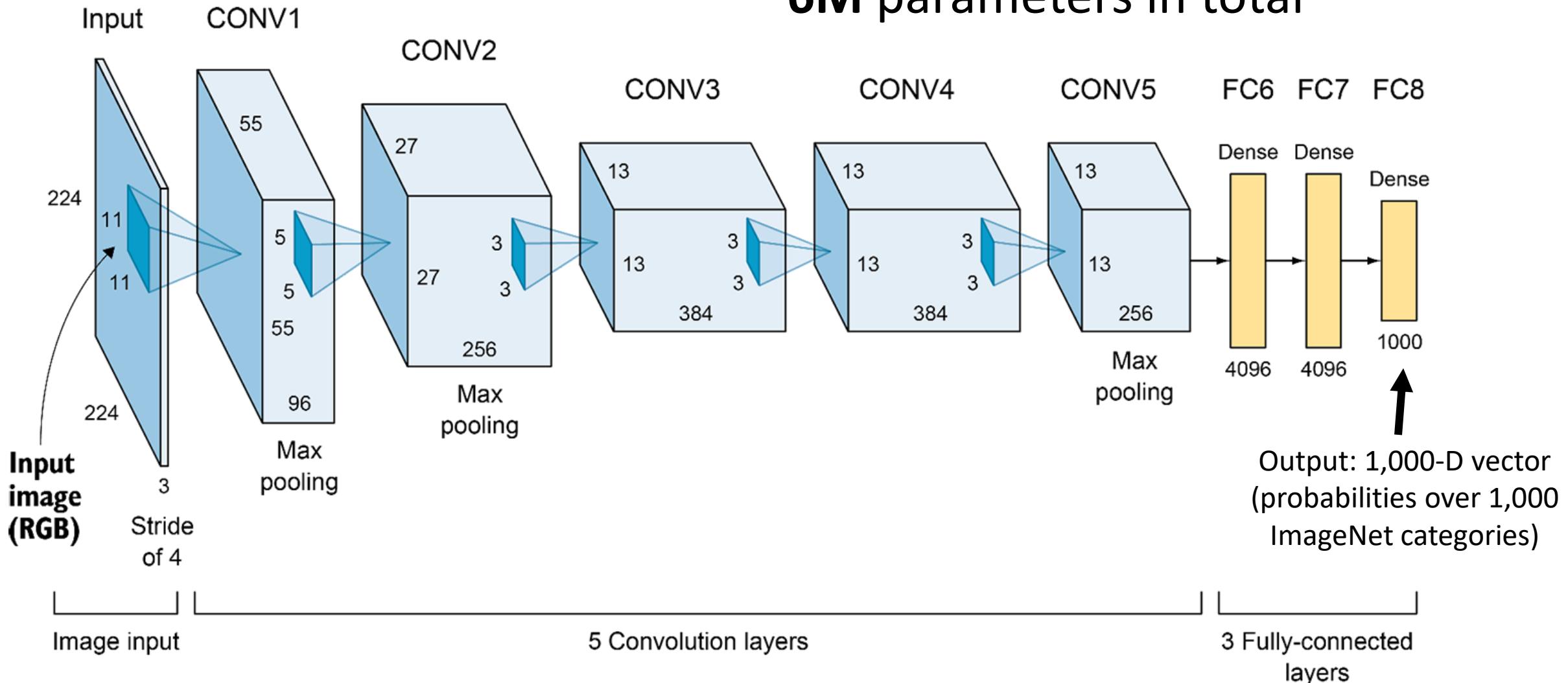
Slide Credit: N. Snavely

CNN training on CIFAR 10

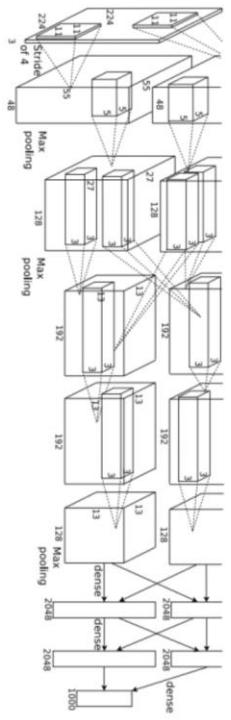
- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

AlexNet (2012)

6M parameters in total

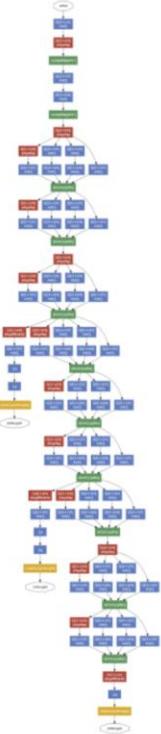


“AlexNet”



[Krizhevsky et al. NIPS 2012]

“GoogLeNet”



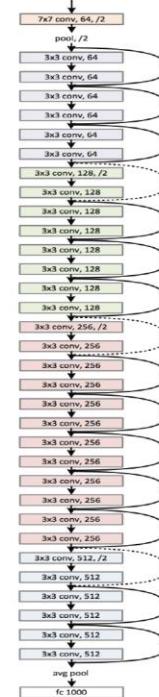
[Szegedy et al. CVPR 2015]

“VGG Net”



[Simonyan & Zisserman,
ICLR 2015]

“ResNet”



[He et al. CVPR 2016]

Table 1. Top 5 accuracy, top 1 accuracy, and the number of parameters of AlexNet, VGG, Inception, and ResNet in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) challenge.

Architecture	Number of Parameters	Top 5 Accuracy	Top 1 Accuracy
AlexNet	62,378,344	84.60%	63.30%
VGG16	138,357,544	91.90%	74.40%
GoogLeNet	23,000,000	92.2%	74.80%
ResNet-152	25,000,000	94.29%	78.57%
DenseNet	8,062,504	93.34%	76.39%
Xception	22,910,480	94.50%	79.00%

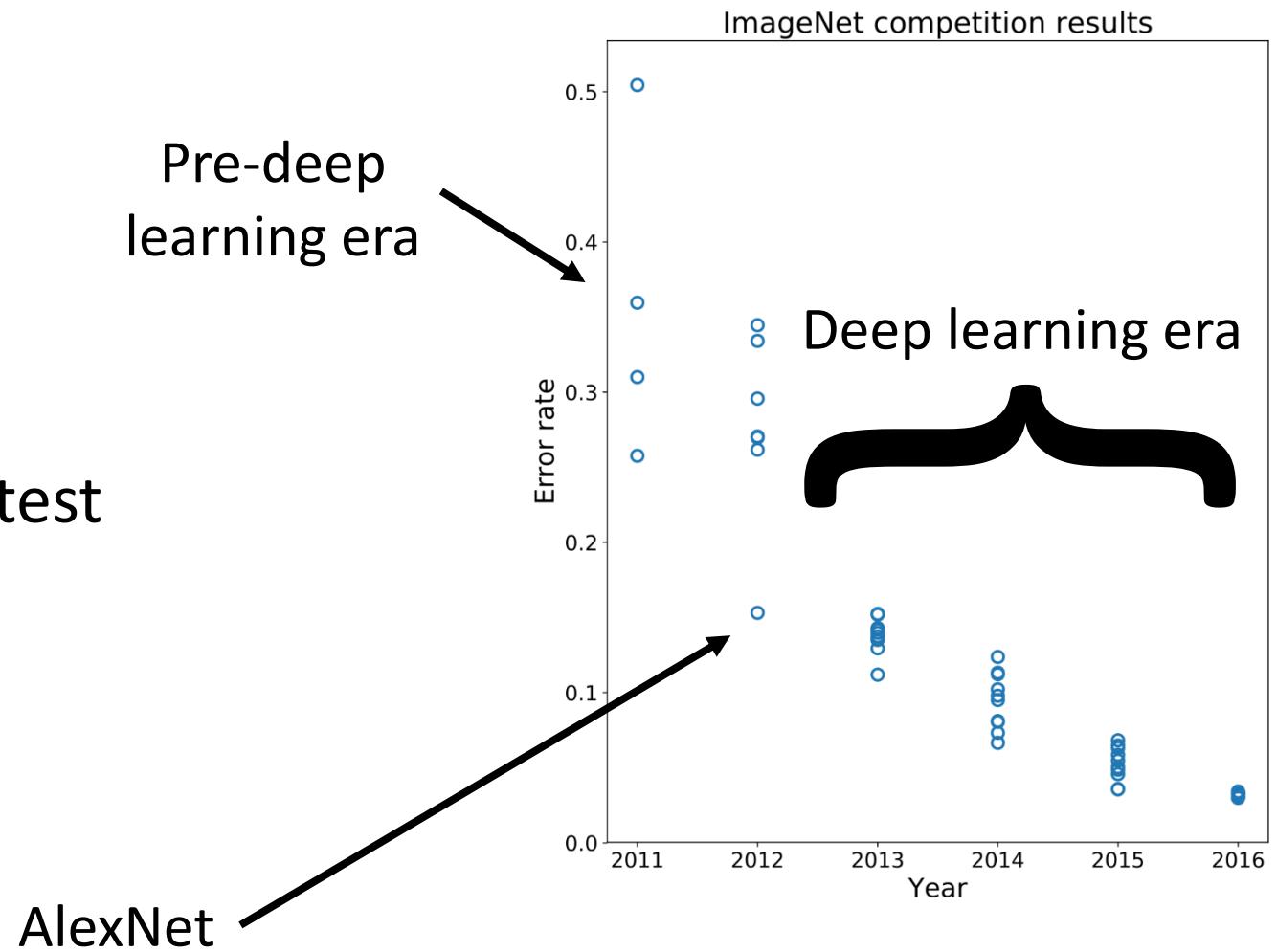
Slide modified from:
N. Snavely

Big picture

- A convolutional neural network can be thought of as a function from images to class scores
 - With millions of adjustable weights...
 - ... leading to a very non-linear mapping from images to features / class scores.
 - We will set these weights based on classification accuracy on training data...
 - ... and hopefully our network will generalize to new images at test time

Performance improvements on ILSVRC

- ImageNet Large-Scale Visual Recognition Challenge
- Held from 2011-2017
- 1000 categories, 1000 training images per category
- Test performance on held-out test set of images



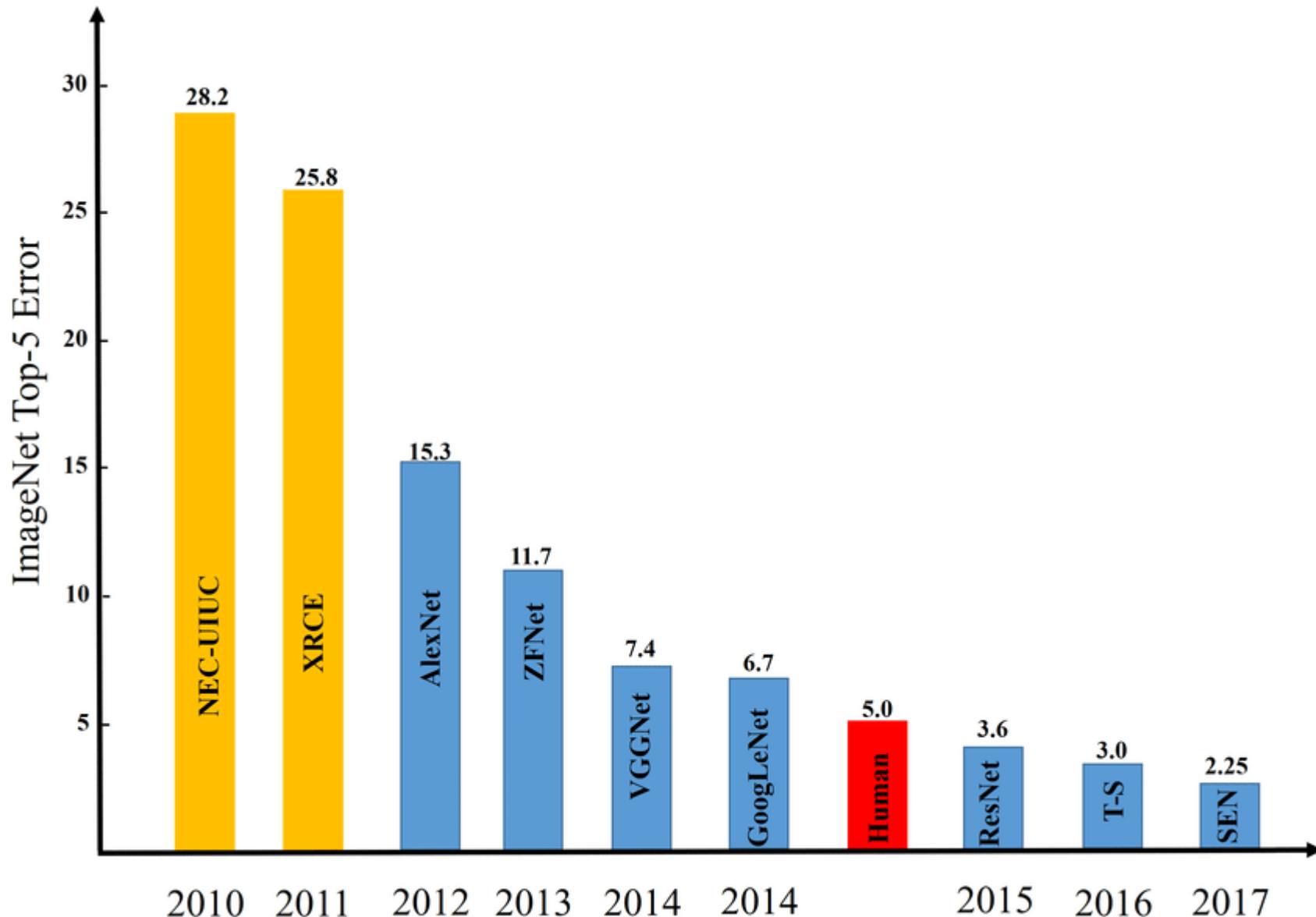


Image credit: Zaid Alyafeai, Lahouari Ghouti

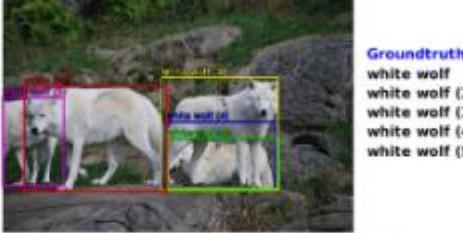
Deep Learning Beyond Image Classification

Computer Vision Tasks & Deep Learning

classification



localization



detection



segmentation



Difficulty

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

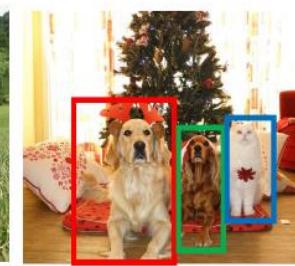
Classification + Localization



CAT

Single Object

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

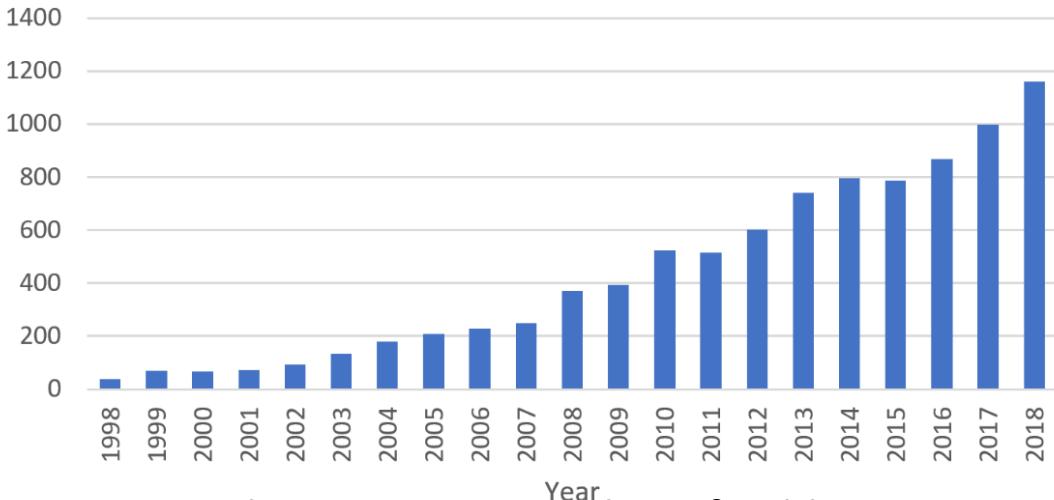
This image is CC0 public domain

Figure: Fei-Fei Li & Justin Johnson & Serena Yeung, Stanford

Object Detection using Deep Learning

Object Detection

- **PASCAL** pascallin.ecs.soton.ac.uk/challenges/VOC
- **ImageNet** www.image-net.org/challenges/LSVRC/2014
- **Sun** sundatabase.mit.edu
- **Microsoft COCO** mscoco.org



The increasing number of publications in object detection from 1998 to 2018. (Data from Google scholar advanced search: all in title: “object detection” AND “detecting objects”). Zou Arxiv 2019

	# classes	average # categories per image	average # instances per image	average object scale	average resolution	# images				# objects			
						total	train	val	test	total	train	val	test
PASCAL	20	1.521	2.711	0.207	469x387	22k	6k	6k	10k	42k?	14k	14k	-
ImageNet13	200	1.534	2.758	0.170	482x415	516k	456k	20k	40k	648k?	480k	56k	-
Sun	4919	9.8	16.9	0.1040	732x547	16873	-			285k	-		
COCO	91	3.5	7.6	0.117	578x483	328k	164k	82k	82k	2500k	~1250k	~625k	~625k

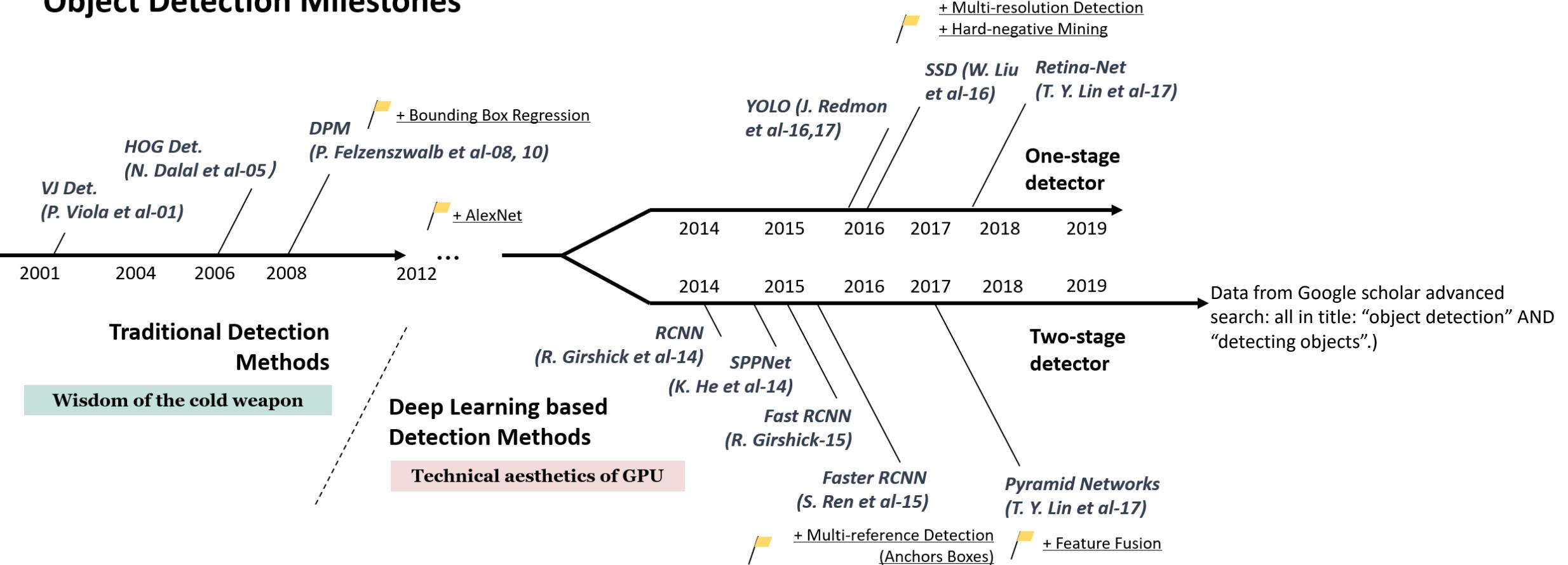
The pascal visual object classes (voc) challenge. Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. *International journal of computer vision* 88, no. 2 (2010): 303-338.

Imagenet: A large-scale hierarchical image database. Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248-255. IEEE, 2009.

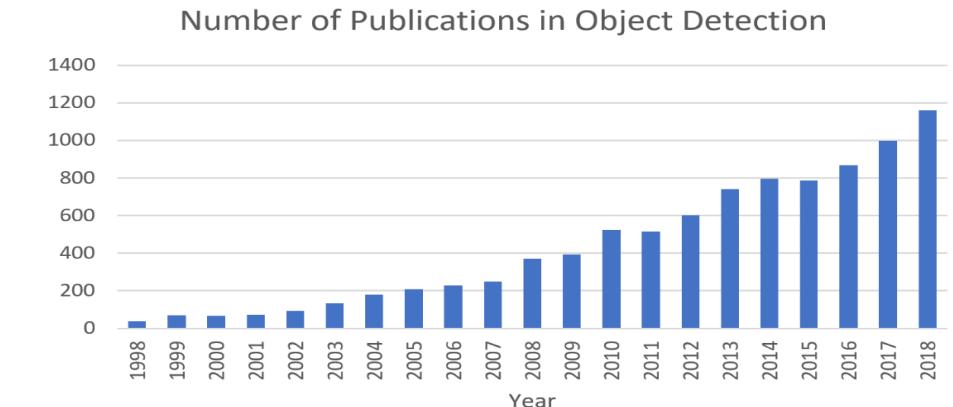
SUN Database: Large-scale Scene Recognition from Abbey to Zoo, Jianxiong Xiao, James Hays, Krista Ehinger, Aude Oliva, and Antonio Torralba. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2010*.

Microsoft COCO: Common Objects in Context, Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick, <http://arxiv.org/abs/1405.0312>, May 2014

Object Detection Milestones



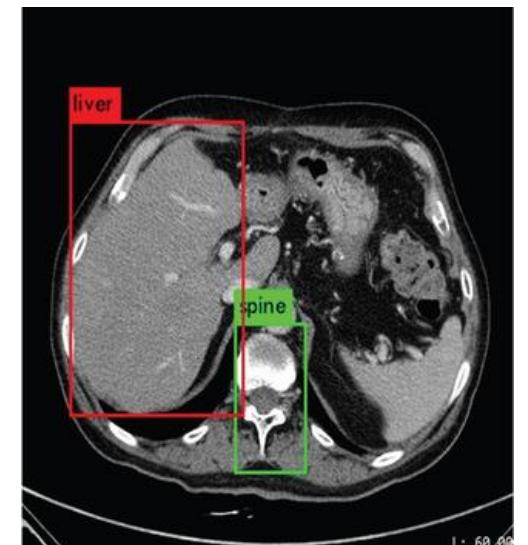
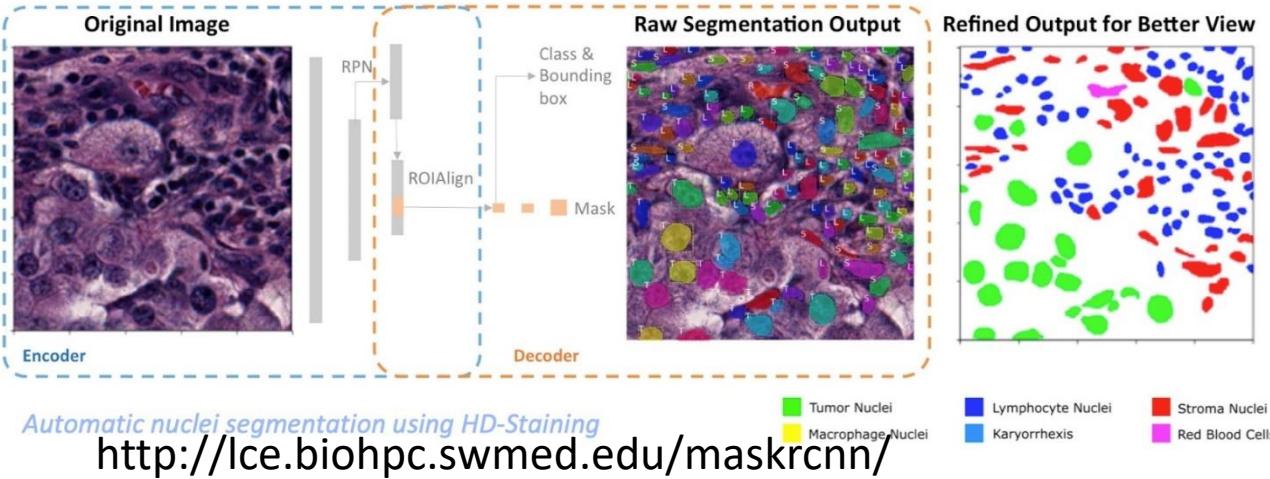
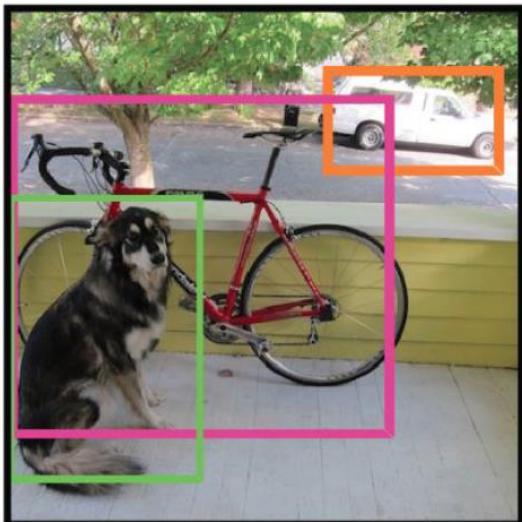
Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).



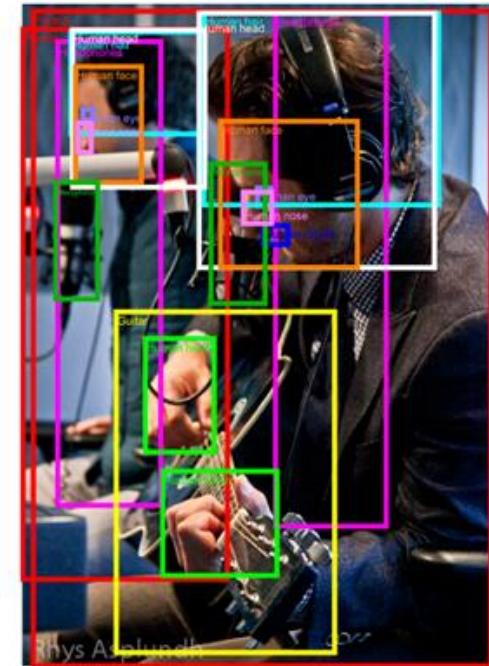
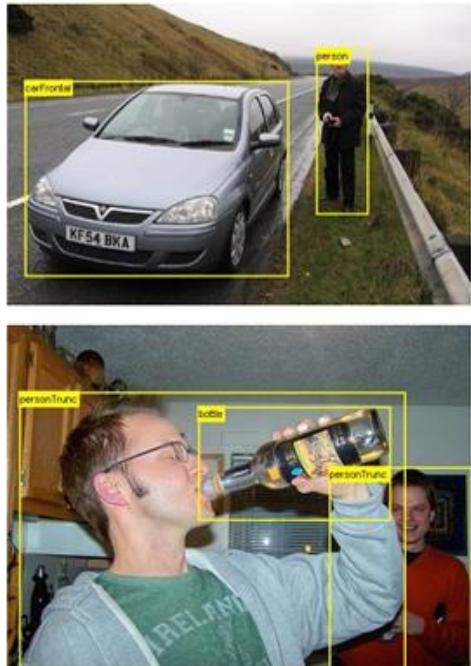
Deep Object Detection

1. Two-stage detection (RCNN and its variants): coarse-to-fine process
2. One-stage detection (SSD, YOLO, RetinaNet etc.): complete in one step

Pang, Shanchen, Tong Ding, Sibo Qiao, Fan Meng, Shuo Wang, Pibao Li, and Xun Wang. "A novel YOLOv3-arch model for identifying cholelithiasis and classifying gallstones on CT images." *PLoS one* 14, no. 6 (2019): e0217647.



Object Detection Benchmark Datasets



Some example images and annotations in (a) PASCAL-VOC07, (b) ILSVRC, (c) MS-COCO, and (d) Open Images.

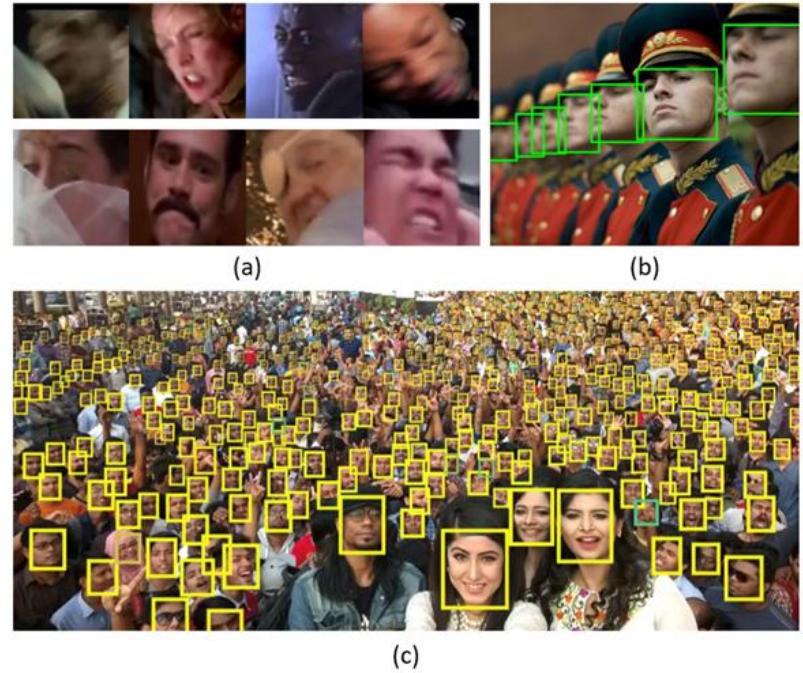
Applications & Challenges



- (a) small pedestrians,
- (b) hard negatives,
- (c) Dense and occluded pedestrians.



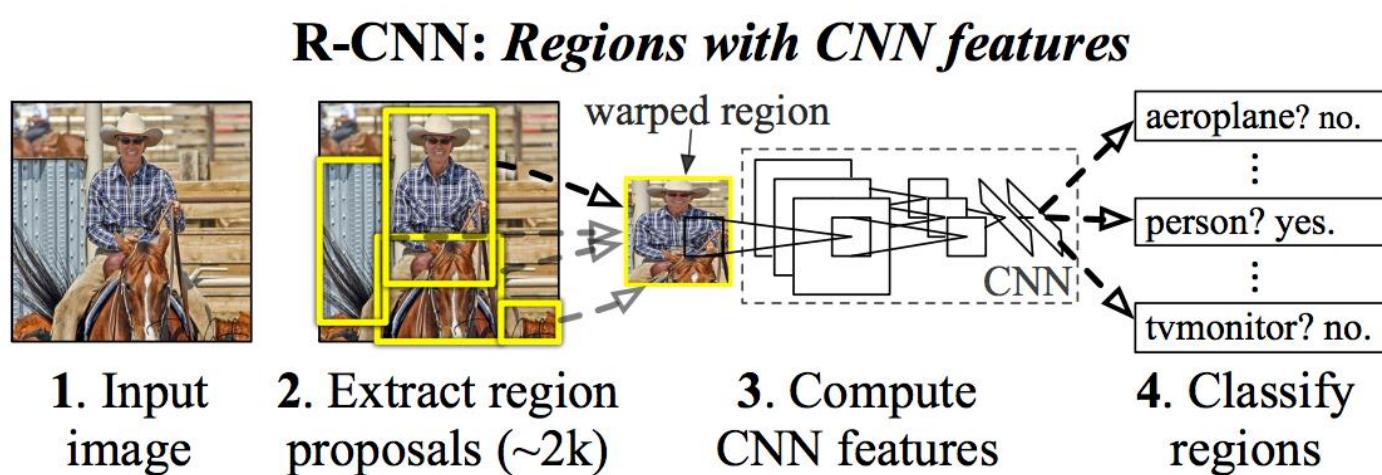
- (a) Illumination changes. Image from pxhere (free of copyrights).
- (b) Motion blur. Image from GTSRB Dataset [81].
- (c) Detection under bad weather. Image from Flickr and Max Pixel (free of copyrights).



- (a) Intra-class variation, image from WildestFaces Dataset [70].
- (b) Face occlusion, image from UFDD Dataset [69].
- (c) Multi-scale face detection. Image from P. Hu et al. CVPR2017 [322].

Two-Stage Detection

- Stage-1: sparse set of candidates
- Stage-2: classify each candidate into one of FG/BG classes
- Examples: R-CNN, Fast R-CNN, Faster R-CNN



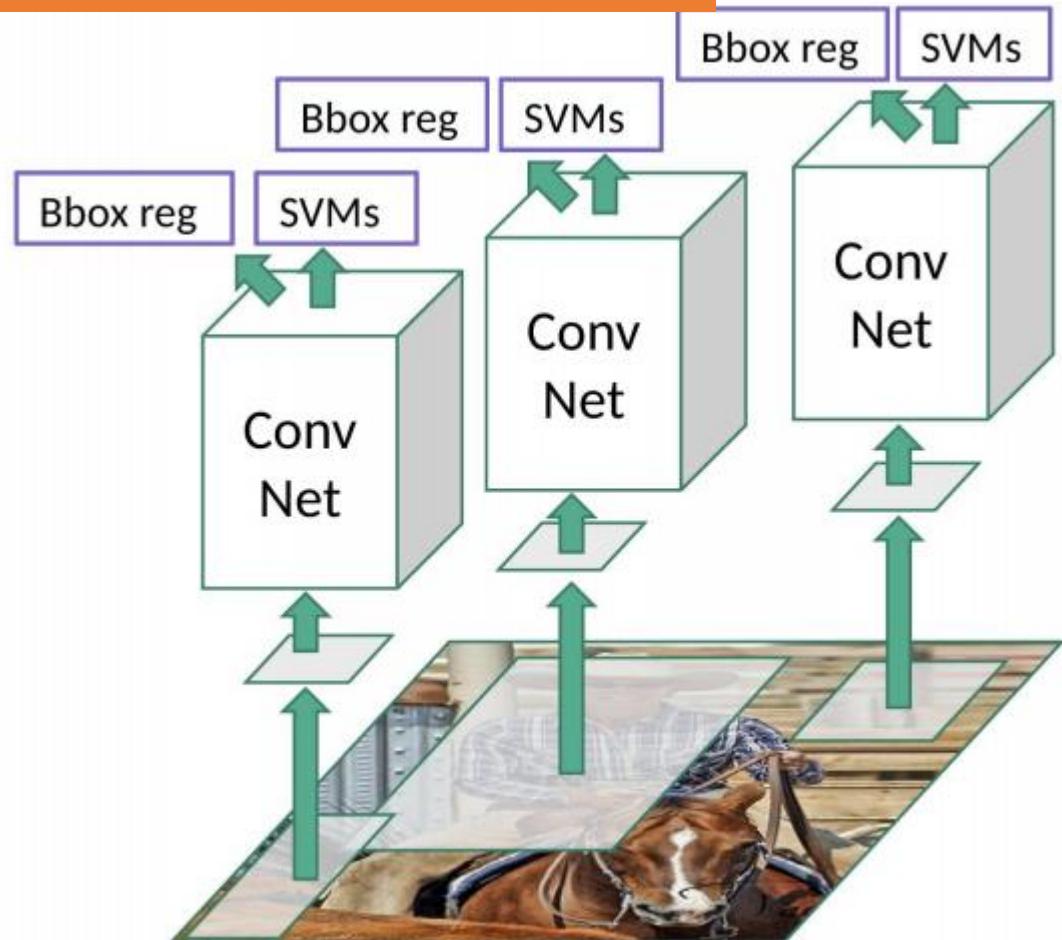
R-CNN

1. Extraction of a set of object proposals (object candidate boxes) by **selective search**.
2. Each proposal is
 1. rescaled to a fixed size image and
 2. fed into a CNN model trained on ImageNet (say, AlexNet [40]) to extract features.
3. Finally, linear SVM classifiers are used to predict the presence of an object within each region and to recognize object categories.

Drawbacks

- redundant feature computations on a large number of overlapped proposals (over 2000 boxes from one image) leads to an extremely slow detection speed.

Mean Average Precision on VOC07
33.7% to 58.5%



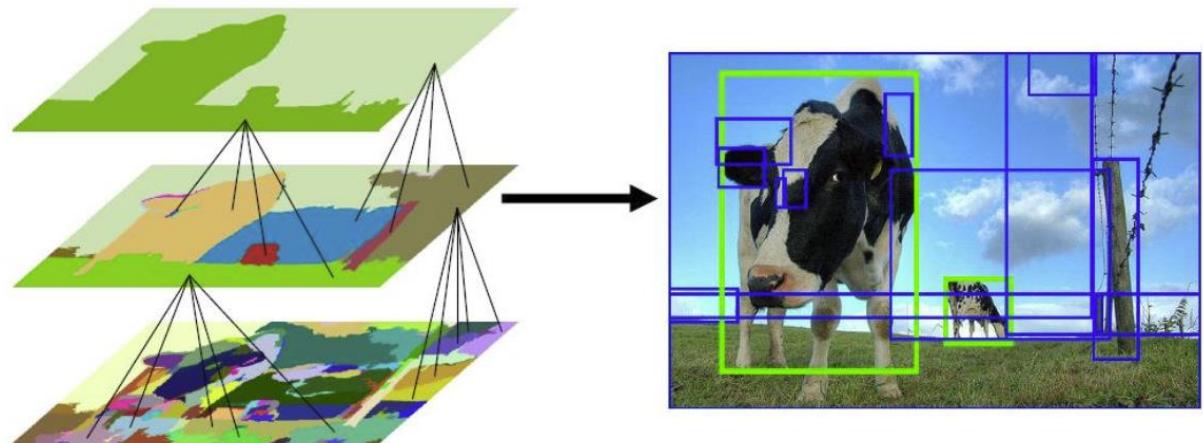
Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587. 2014. 29K citations.

- Last slide covered on March 14

Selective Search

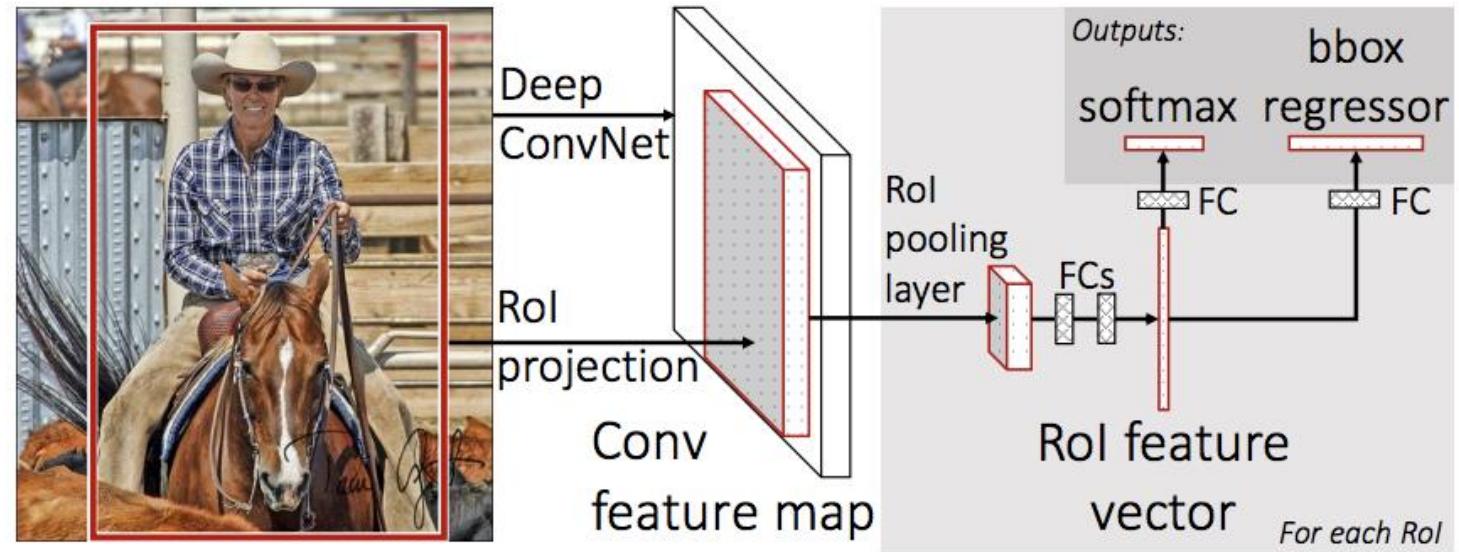
1. First, color similarities, texture similarities, region size, and region filling are used as **non-object-based segmentation**. Therefore we obtain **many small segmented areas** as shown at the bottom left of the image above.
2. Then, bottom-up approach is used that **small segmented areas** are **merged together to form larger segmented areas**.
3. Thus, **about 2K region proposals (bounding box candidates)** are **generated** as shown in the image.

Uijlings, Jasper RR, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. "Selective search for object recognition." *International journal of computer vision* 104, no. 2 (2013): 154-171. 3550 citations.



Fast R-CNN

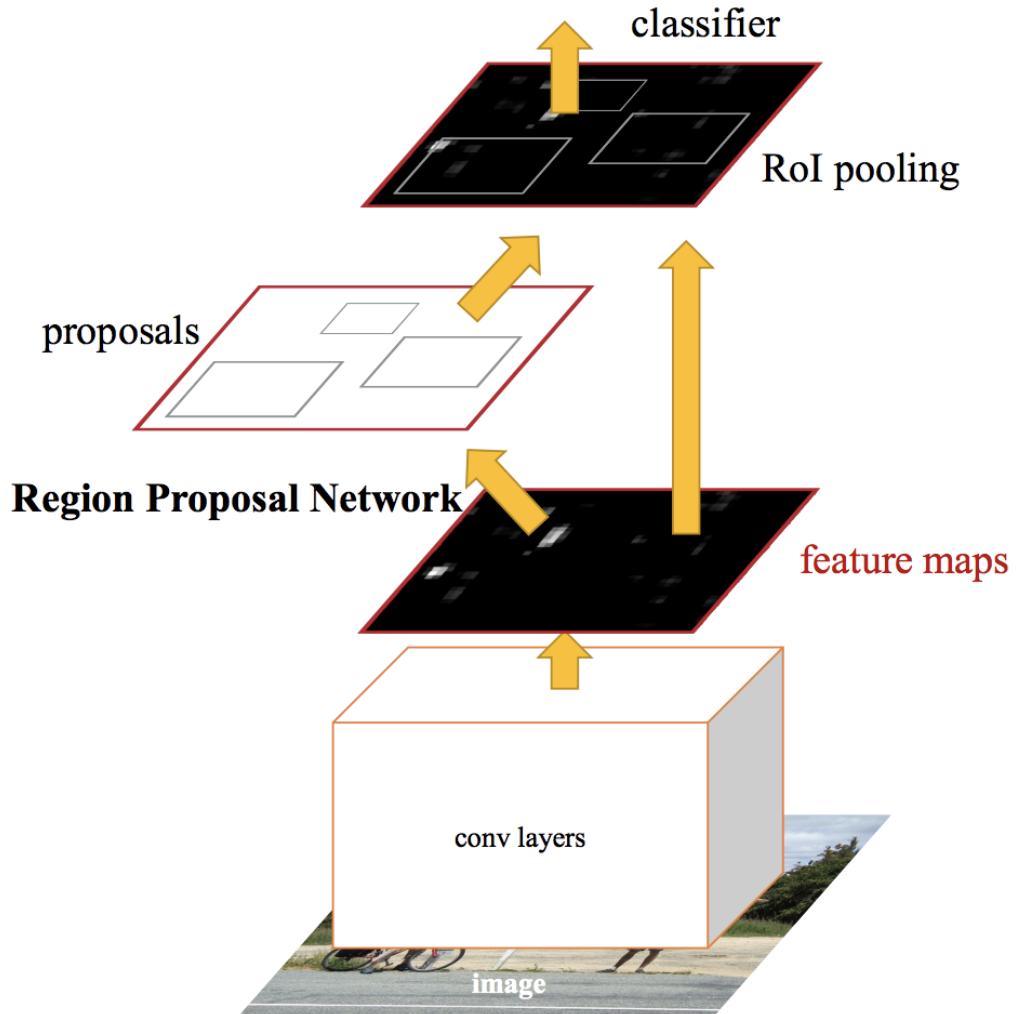
Mean Average Precision on VOC07
33.7% to 58.5% to 70.0%
Speed x200 compared to RCNN



- Instead of feeding the region proposals to the CNN, input image is fed to the CNN to generate a convolutional feature map.
- From the convolutional feature map, region of proposals are identified and warped them into squares and by using a ROI pooling layer.

Faster R-CNN

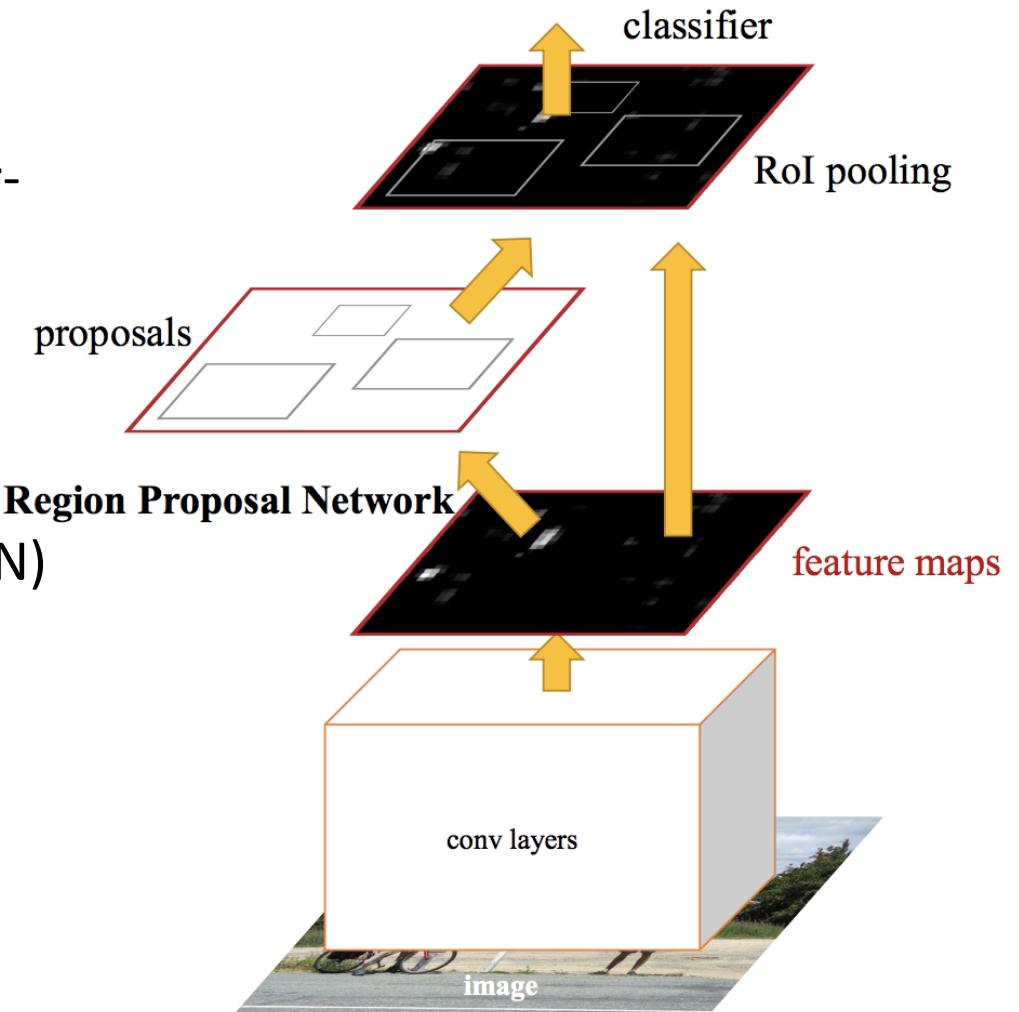
- Eliminates the selective search algorithm and lets the network learn the region proposals.
- Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map.
- Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals.
- The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.



Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In *Advances in neural information processing systems*, pp. 91-99. 2015. 55K citations.

Faster R-CNN

- Faster RCNN is the first end-to-end, and the first near-realtime deep learning detector
- (COCO mAP@.5=42.7%, COCO [mAP@\[.5,.95\]=21.9%](#), VOC07 mAP=73.2%, VOC12 mAP=70.4%, 17fps with ZFNet [45]).
- The main contribution: Region Proposal Network (RPN) that enables nearly cost-free region proposals.

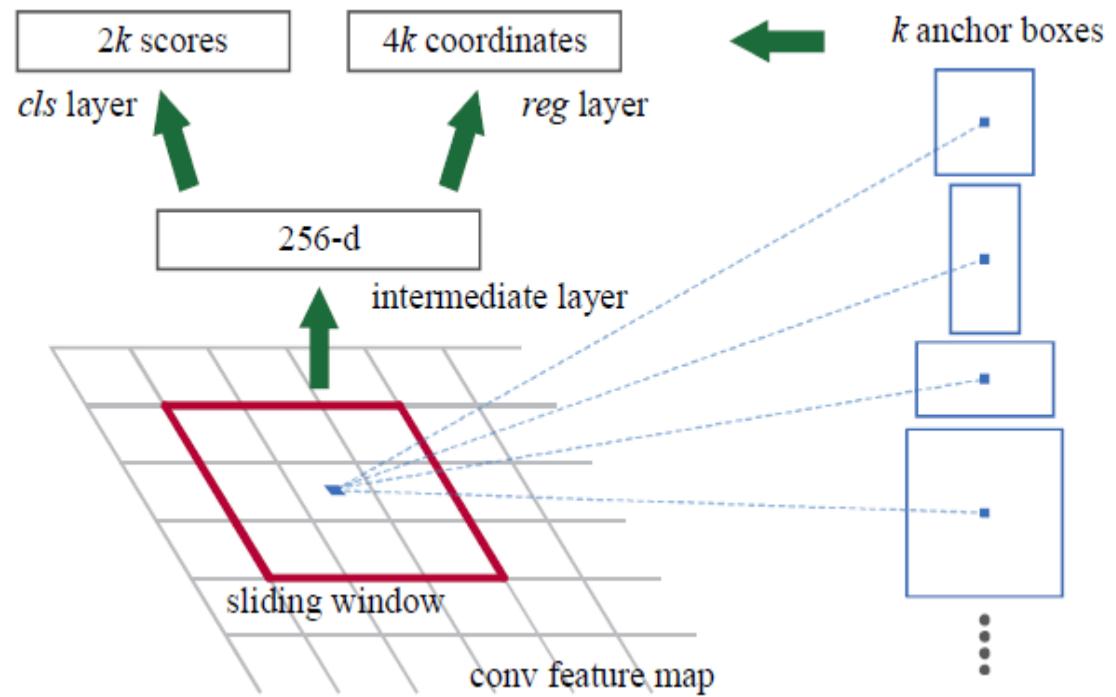


Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In *Advances in neural information processing systems*, pp. 91-99. 2015. 55K citations.

Region Proposal Network

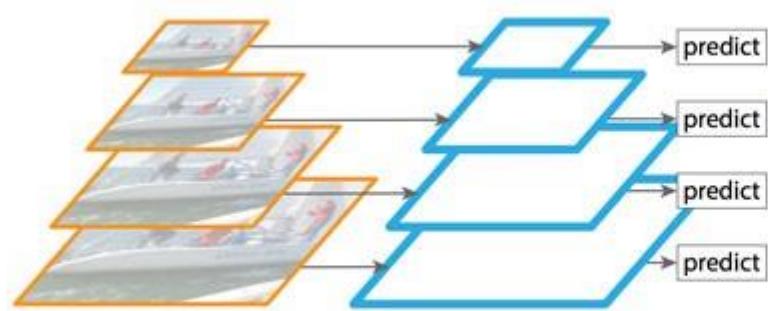
1. First, the picture goes through conv layers and feature maps are extracted
2. Then a **sliding window** is used in RPN for each location over the feature map.
3. For each location, k ($k=9$) anchor boxes are used (**3 scales of 128, 256 and 512, and 3 aspect ratios of 1:1, 1:2, 2:1**) for generating region proposals.
4. A **cls** layer outputs $2k$ scores **whether there is object or not** for k boxes.
5. A **reg** layer outputs $4k$ for the **coordinates** (box center coordinates, width and height) of k boxes.
6. With a size of $W \times H$ feature map, there are WHk anchors in total.

Thus, **RPN network is to pre-check which location contains object**. And the corresponding locations and **bounding boxes will pass to detection network** for detecting the object class and returning the bounding box of that object.

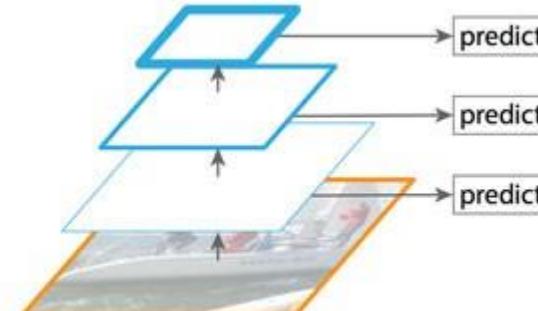


Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In *Advances in neural information processing systems*, pp. 91-99. 2015. 13444 citations.

Feature Pyramid Network (FPN)



Pyramid of images



Pyramid of feature maps

- Replaces the feature extractor of detectors like Faster R-CNN
- Generates multiple feature map layers (**multi-scale feature maps**) with better quality information than the regular feature pyramid for object detection.

- Lin, Tsung-Yi, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. "Feature Pyramid Networks for Object Detection." In *CVPR*, vol. 1, no. 2, p. 3. 2017.
- https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

Feature Pyramid Network (FPN)

- Before FPN, most of the deep learning-based detectors run detection only on a network's top layer.
- Although the features in deeper layers of a CNN are beneficial for category recognition, it is not conducive to localizing objects.
- To this end, a topdown architecture with lateral connections is developed in FPN for building high-level semantics at all scales.
- Using FPN in a basic Faster R-CNN system, it achieves state-of-the-art single model detection results on the MSCOCO dataset
- FPN has now become a basic building block of many latest detectors.

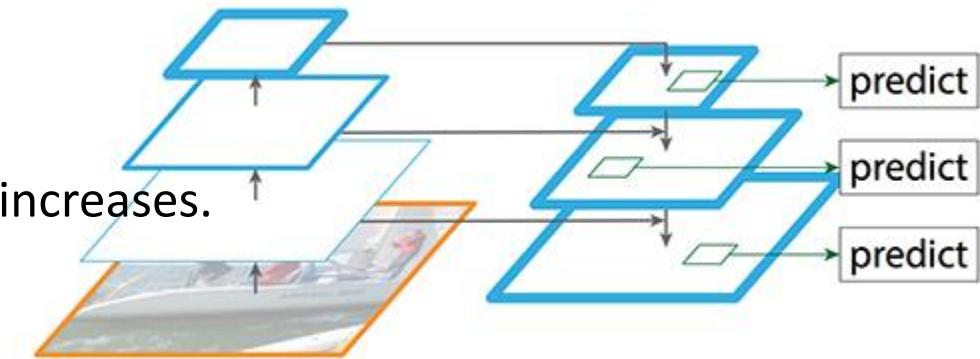
FPN DataFlow

Bottom-up pathway:

- usual convolutional network for feature extraction.
- as we go up, the spatial resolution decreases, the **semantic value** increases.

Top-down pathway:

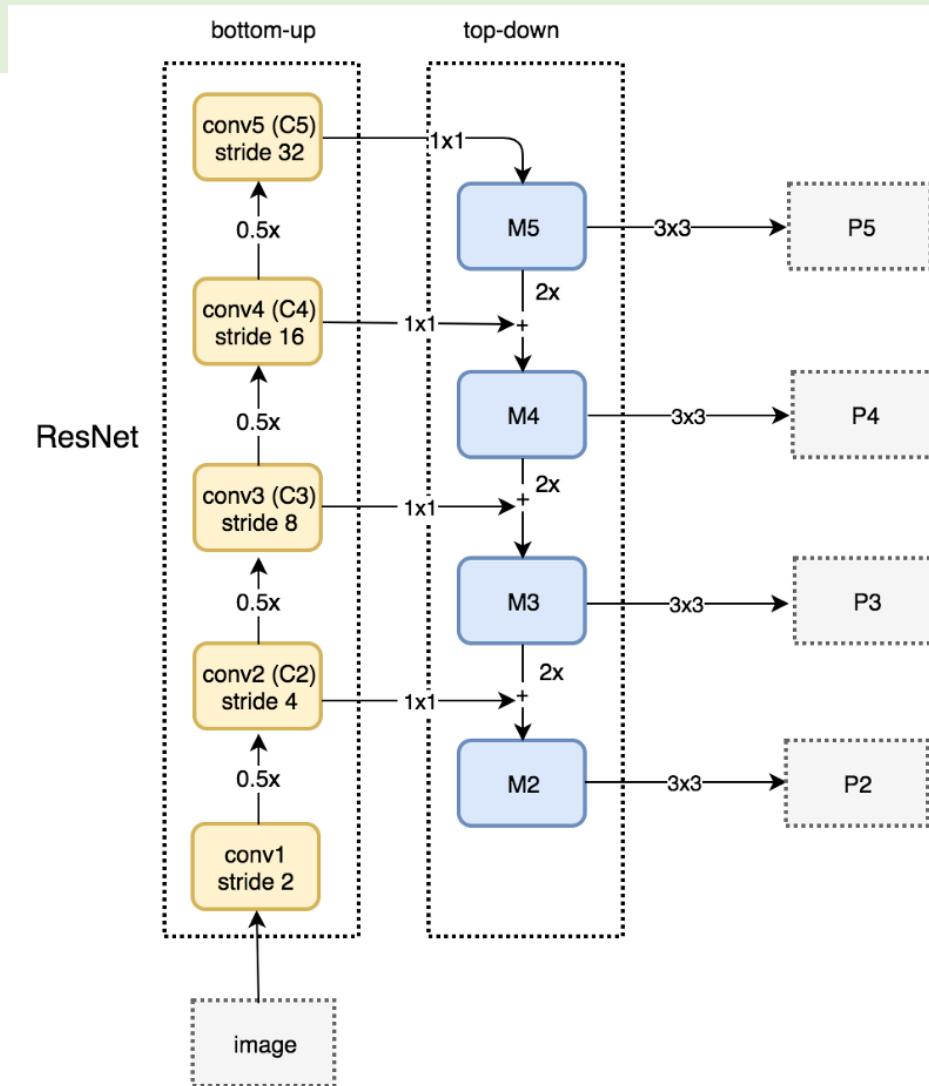
- construct higher resolution layers from a semantic rich layer.
- reconstructed layers are semantic strong but the locations of objects are not precise (after downsampling and upsampling).
- lateral connections between reconstructed layers and the corresponding feature maps
 - help the detector to predict the location better.
 - acts as skip connections to make training easier (similar to ResNet)



Top-down pathway

- **C5->M5:** 1×1 convolution filter to reduce C5 channel depth to 256-d
- **M5->P5:** 3×3 convolution, the first feature map layer used for object prediction.
- Top-down path:
 - upsample the previous layer by 2 using nearest neighbors upsampling.
 - apply a 1×1 convolution to the corresponding feature maps
 - add them element-wise.
 - apply a 3×3 convolution again to output the next feature map layers for object detection.

This filter reduces the aliasing effect of upsampling.

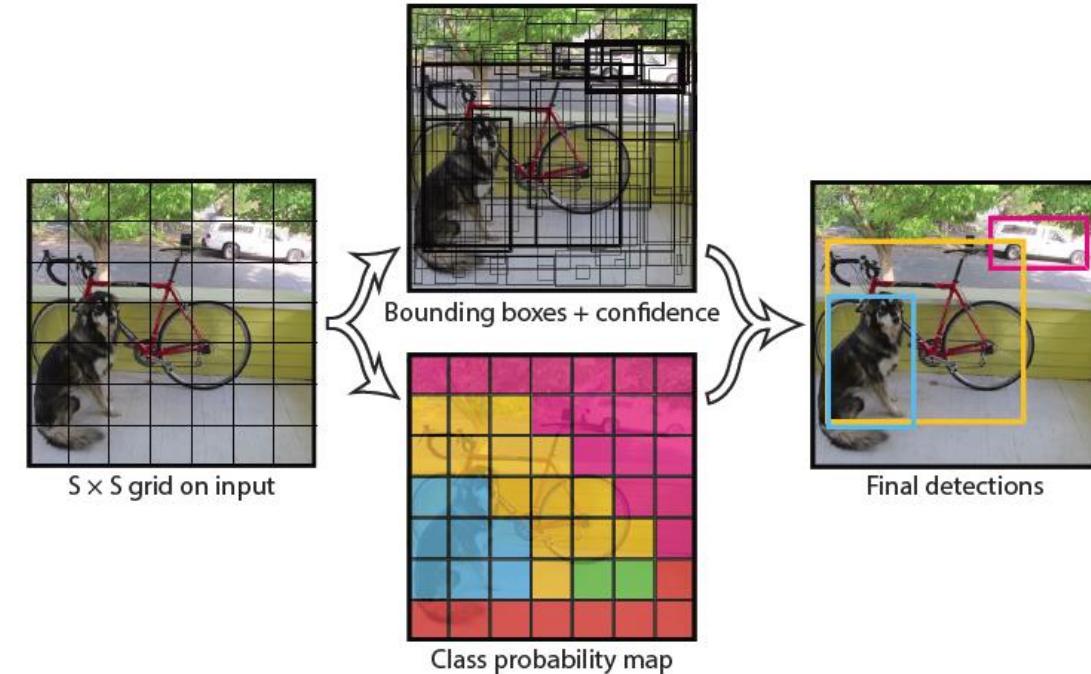


Single-Stage Detectors

- Regular dense sampling of objects/scales/aspect ratios
- Examples: YOLO, SSD, RetinaNet

CNN based One-stage Detectors: You Only Look Once (YOLO)

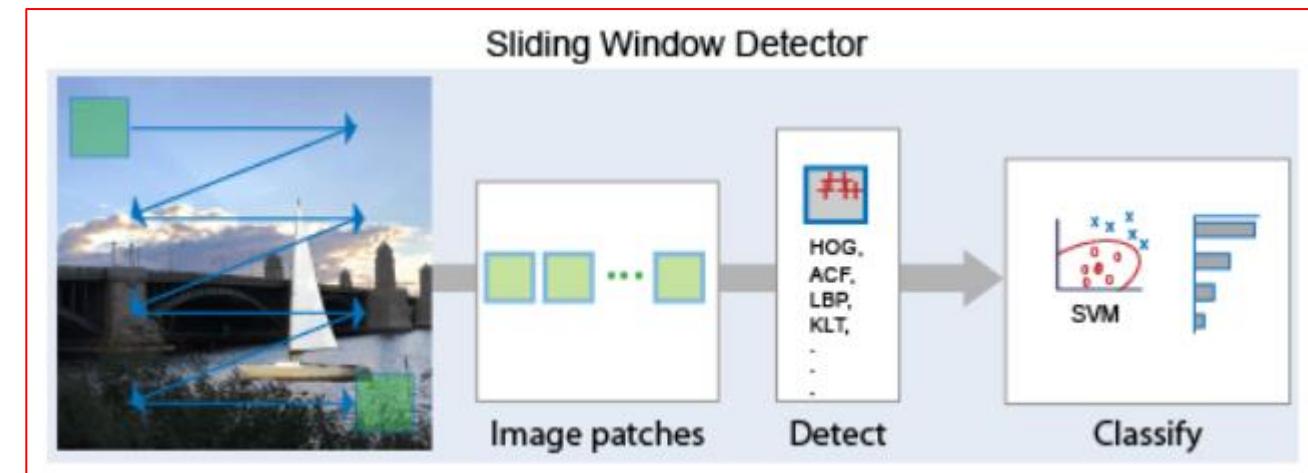
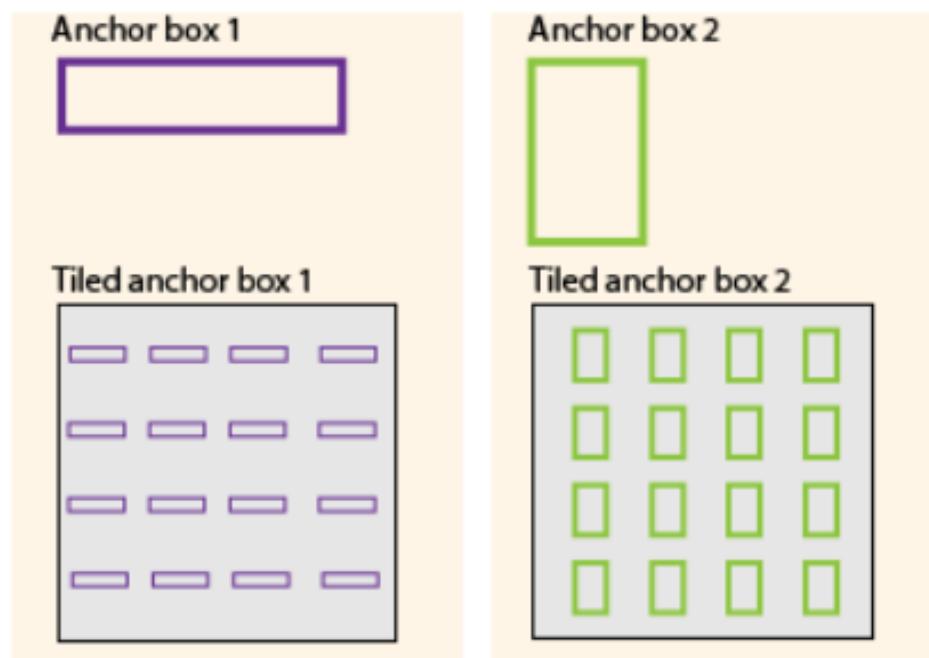
- Instead of having two networks: Region Proposals Network + Classifier Network
- **In Single-shot architectures, bounding boxes and confidences for multiple categories are predicted directly with a single network**
- e.g. : Overfeat, YOLO



What Is an Anchor Box?

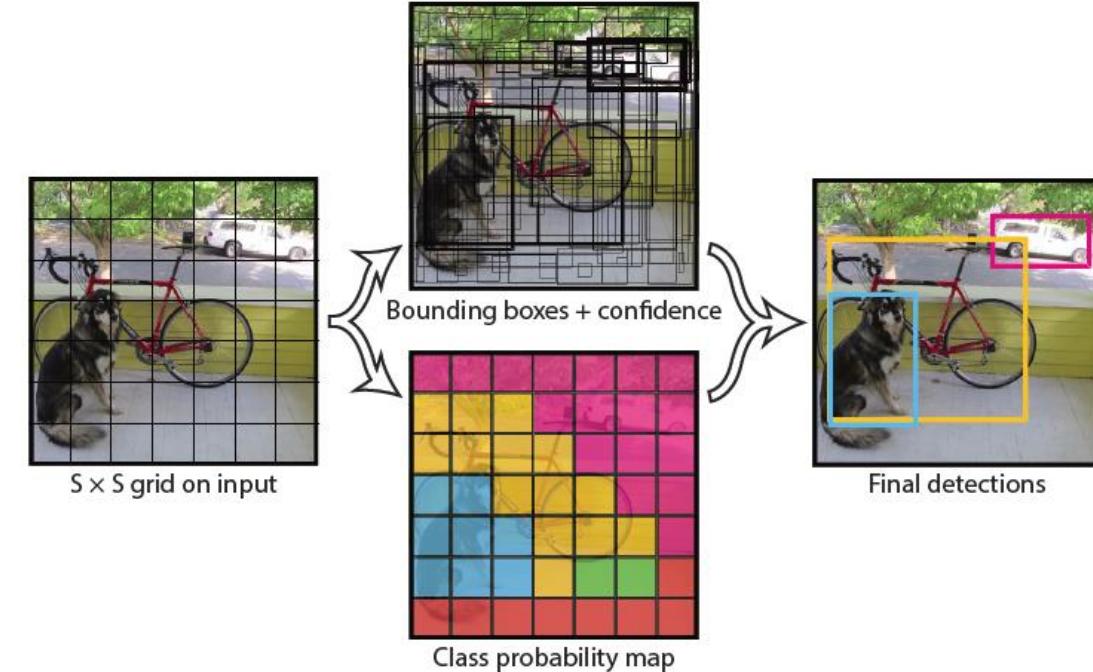
Anchor boxes are a set of predefined bounding boxes of a certain height and width. These boxes are defined to capture the scale and aspect ratio of specific object classes you want to detect and are typically chosen based on object sizes in your training datasets. During detection, the predefined anchor boxes are tiled across the image. The network predicts the probability and other attributes, such as background, intersection over union (IoU) and offsets for every tiled anchor box. The predictions are used to refine each individual anchor box. You can define several anchor boxes, each for a different object size. Anchor boxes are fixed initial boundary box guesses.

The network does not directly predict bounding boxes, but rather predicts the probabilities and refinements that correspond to the tiled anchor boxes. The network returns a unique set of predictions for every anchor box defined. The final feature map represents object detections for each class. The use of anchor boxes enables a network to detect multiple objects, objects of different scales, and overlapping objects.



CNN based One-stage Detectors: You Only Look Once (YOLO)

- First one-stage detector in deep learning era.
- Apply a single neural network to the full image.
- Divides the image into regions and predicts bounding boxes and probabilities for each region simultaneously.
- YOLO is extremely fast: a fast version of YOLO runs at 155fps with VOC07 mAP=52.7%, while its enhanced version runs at 45fps with VOC07 mAP=63.4% and VOC12 mAP=57.9%.
- In spite of its great improvement of detection speed, YOLO suffers from a drop of the localization accuracy compared with two-stage detectors, especially for some small objects. YOLO's subsequent versions [48, 49] and the latter proposed SSD [21] has paid more attention to this problem.



YOLOv3 (You Only Look Once)

Single shot detector, and one of the faster object detector

Uses darknet-53 deep architecture (fully convolutional layers)

makes detections at three different scales

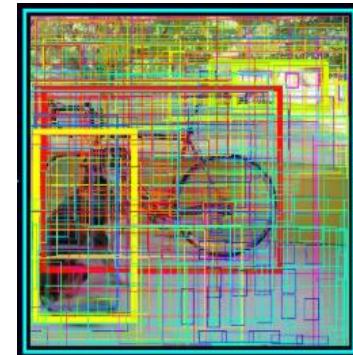
Uses 9 anchor boxes. Three for each scale

The up sampled layers concatenate with the previous layers, help preserve the fine grained features

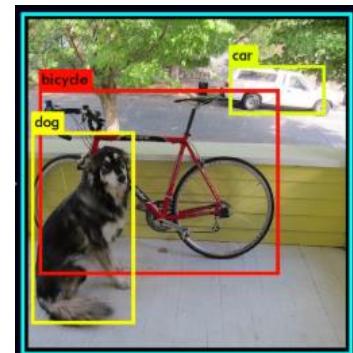
Helps address the issue of detecting small objects.



Generate grid on input



Compute Bboxes & confidence

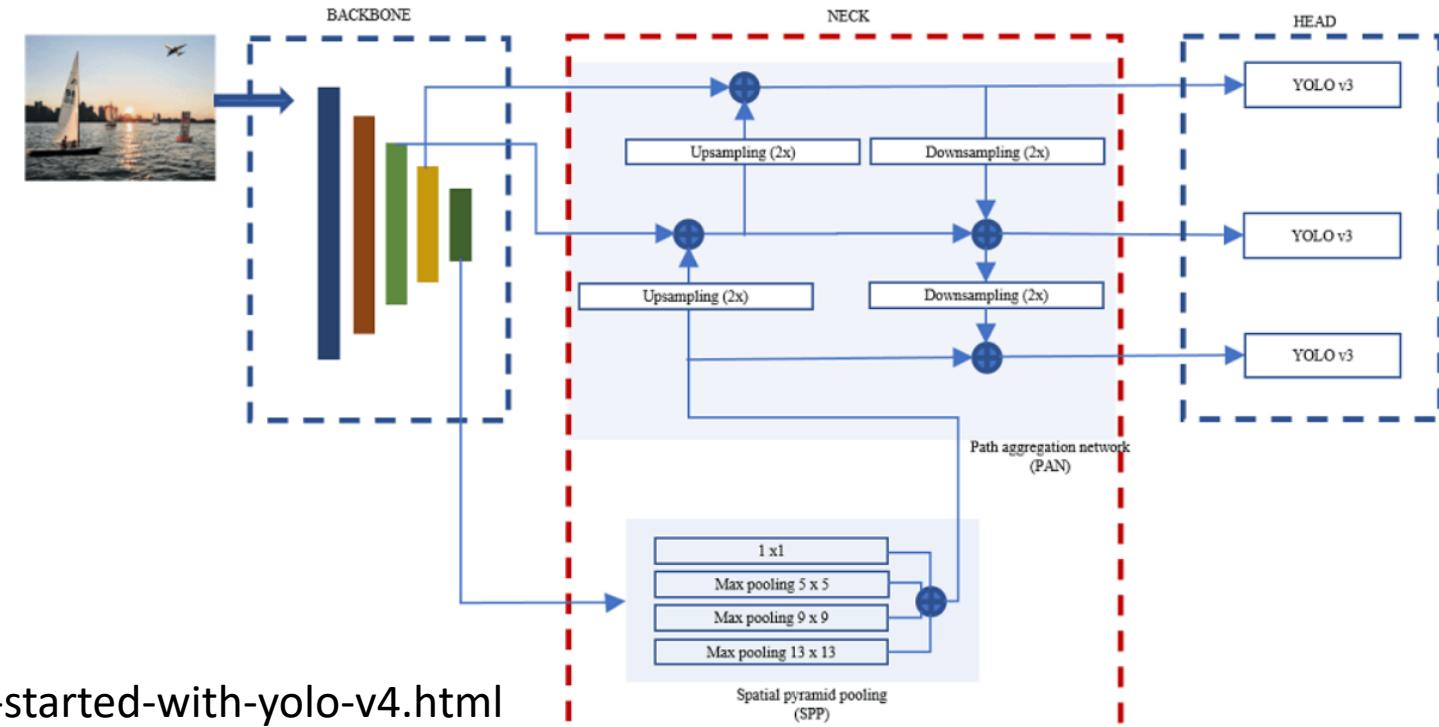


Final detection

YOLOv3	YOLO
YOLOv3 (Better, not Faster, Stronger) Since YOLOv3 has more layers	
Darknet-53 network architecture	Darknet-19 network architecture
Solve the issue of small object, with three scales, and concatenate the up sample with previous layer	Struggled with small object detections, because of the loss of fine-grained features as the layers down sampled the input
Detection is done by applying 1×1 detection kernels on feature maps of three different sizes at three different places in the network	Fully convolutional network and its eventual output is generated by applying a 1×1 kernel on a feature map
YOLOv3 predicts bounding boxes at 3 different scales	predicts 10x less the number of bounding boxes predicted by YOLOv3

YOLO v4

- The you only look once version 4 (YOLO v4) object detection network is a one-stage object detection network and is composed of three parts: backbone, neck, and head.
- The **backbone** can be a pretrained convolutional neural network such as VGG16 or CSPDarkNet53 trained on COCO or ImageNet data sets. The backbone of the YOLO v4 network acts as the feature extraction network that computes feature maps from the input images.
- The **neck** connects the backbone and the head. It is composed of a spatial pyramid pooling (SPP) module and a path aggregation network (PAN). The neck concatenates the feature maps from different layers of the backbone network and sends them as inputs to the head.
- The **head** processes the aggregated features and predicts the bounding boxes, objectness scores, and classification scores. The YOLO v4 network uses one-stage object detectors, such as YOLO v3, as detection heads.



Single Shot MultiBox Detector (SSD)

Contributions:

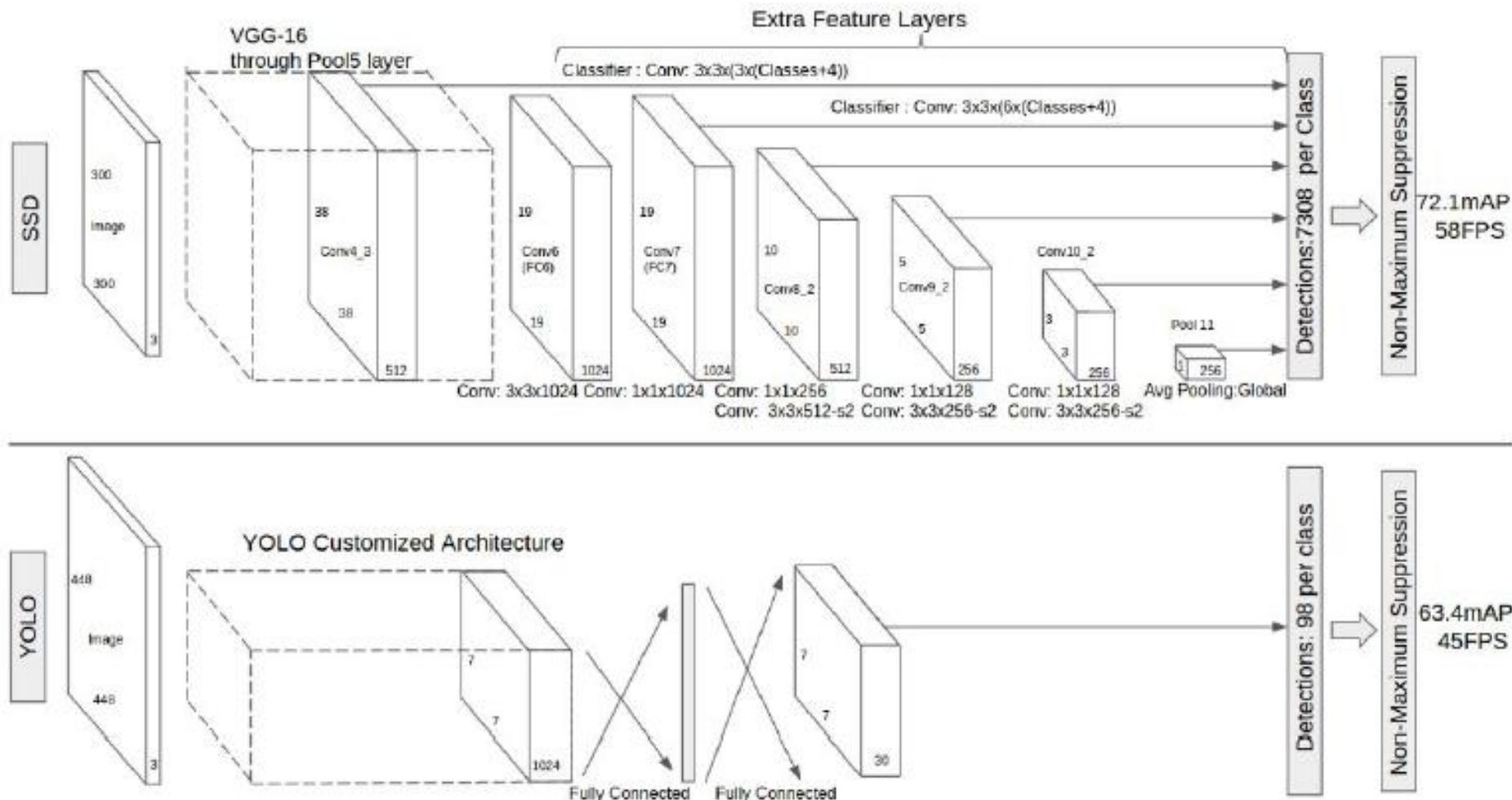
- ▷ A single-shot detector for multiple categories that is faster than state of the art single shot detectors (YOLO) and as accurate as Faster R-CNN
- ▷ Predicts category scores and boxes offset for a fixed set of default BBs **using small convolutional filters applied to feature maps**
- ▷ Predictions of different scales from feature maps of different scales, and separate predictions by aspect ratio
- ▷ End-to-end training and high accuracy, **improving speed vs accuracy trade-off**

- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37
- Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

Single Shot MultiBox Detector (SSD)

- Second one-stage detector in deep learning era.
- **SSD**: First deep network based object detector that does **not resample pixels or features** for bounding box hypotheses and is **as accurate as approaches that do**.
- The main contribution: introduction of the multi-reference and multi-resolution detection techniques
- Significantly improves the detection accuracy of a one-stage detector, especially for some small objects.
- SSD has advantages in terms of both detection speed and accuracy (VOC07 mAP=76.8%, VOC12 mAP=74.9%, COCO mAP@.5=46.5%, mAP@[.5,.95]=26.8%, a fast version runs at 59fps).
- The main difference between SSD and any previous detectors:
 - SSD detects objects of different scales on different layers of the network,
 - Previous detectors only run detection on their top layers.
- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37

The Single Shot Detector (SSD)



- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37

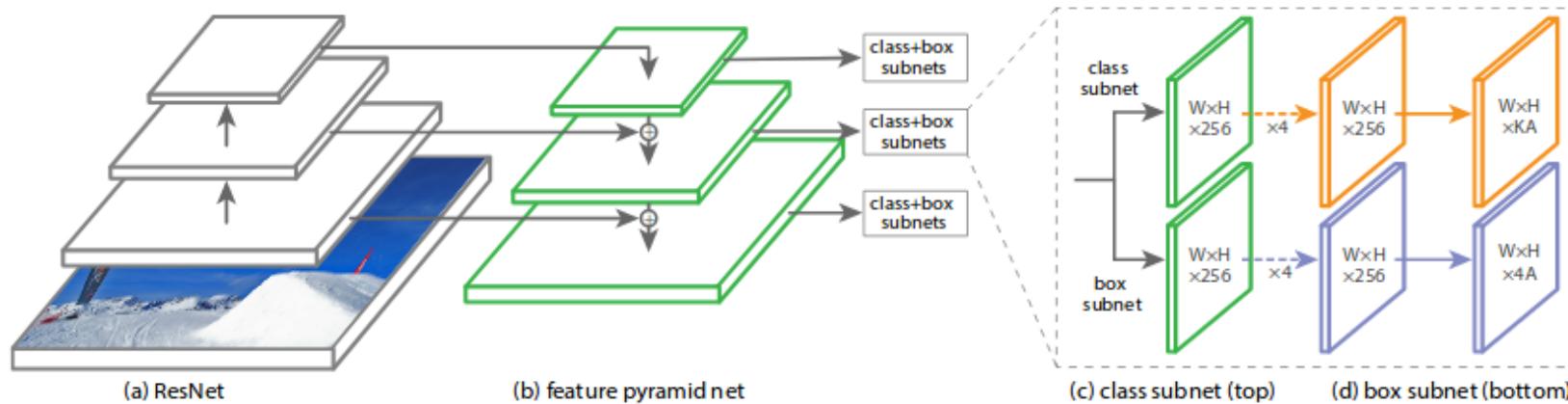
RetinaNet

- Despite of its high speed and simplicity, the one-stage detectors have trailed the accuracy of two-stage detectors for years.
- T.-Y. Lin et al. have discovered the reasons behind and proposed RetinaNet in 2017.
- They claimed that the extreme foreground-background class imbalance encountered during training of dense detectors is the central cause.
- To this end, a new loss function named “focal loss” has been introduced in RetinaNet by reshaping the standard cross entropy loss so that detector will put more focus on hard, misclassified examples during training.
- Focal Loss enables the one-stage detectors to achieve comparable accuracy of two-stage detectors while maintaining very high detection speed. (COCO mAP@.5=59.1%, mAP@[.5, .95]=39.1%).

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” IEEE PAMI, 2018.

Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

RetinaNet



- **Backbone:** Feature Pyramid Network (FPN) on top of a feedforward ResNet architecture [16] to generate a rich, multi-scale convolutional feature pyramid.
- **Two subnetworks:**
 - Subnet-1: for classifying anchor boxes
 - Subnet-2: for regressing from anchor boxes to ground-truth object boxes

RetinaNet

Schultheiss, Manuel, Sebastian A. Schober, Marie Lodde, Jannis Bodden, Juliane Aichele, Christina Müller-Leisse, Bernhard Renger, Franz Pfeiffer, and Daniela Pfeiffer. "A robust convolutional neural network for lung nodule detection in the presence of foreign bodies." *Scientific Reports* 10, no. 1 (2020): 1-9.

Chest radiographies with nodules detected by RetinaNet.

Green rectangles: ground-truth is marked

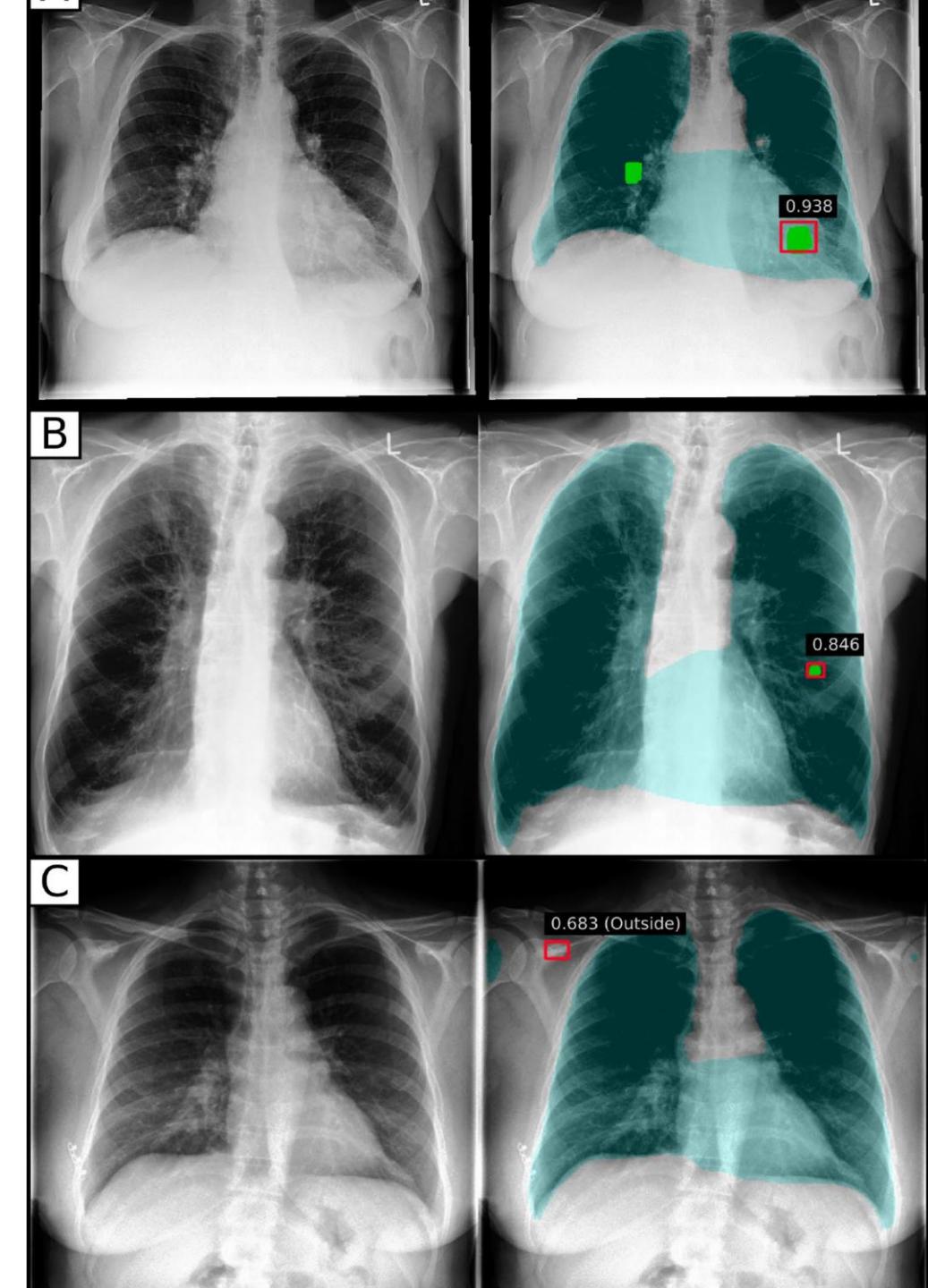
Red: predictions

Cyan: Predicted lung segmentation masks

(A) True-positive prediction (0.938) marked with a red rectangle by the CNN and an undetected, false-negative nodule in the left lung lobe.

(B) True-positive prediction within the right lung lobe.

(C) False-positive prediction outside of the chest.



Key Ideas behind RetinaNet

Two-stage detectors:

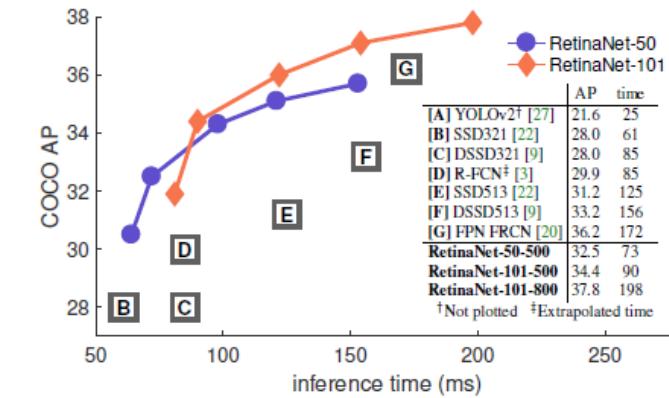
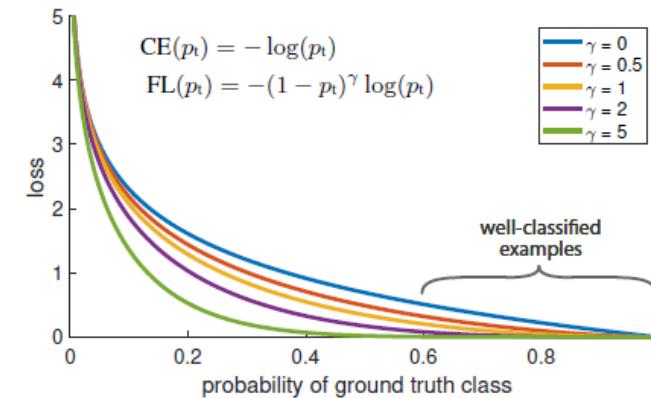
1. Set of sparse candidates are determined,
2. A classifier is applied on these candidate locations

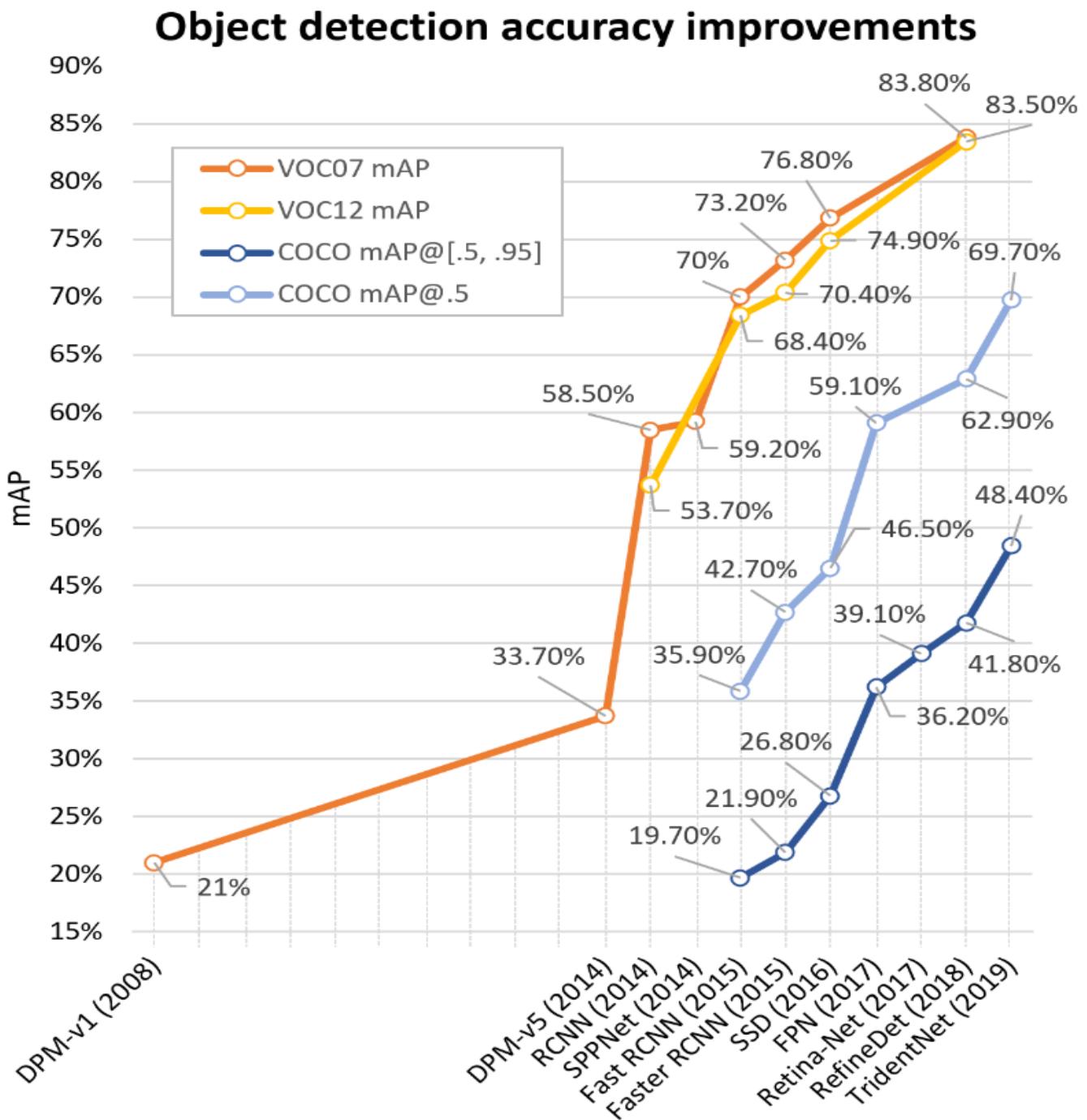
One-stage detectors:

- Applied over a regular, dense sampling of possible object locations
- Faster and simpler
- But lower accuracy compared to two-stage detectors.

RetinaNet:

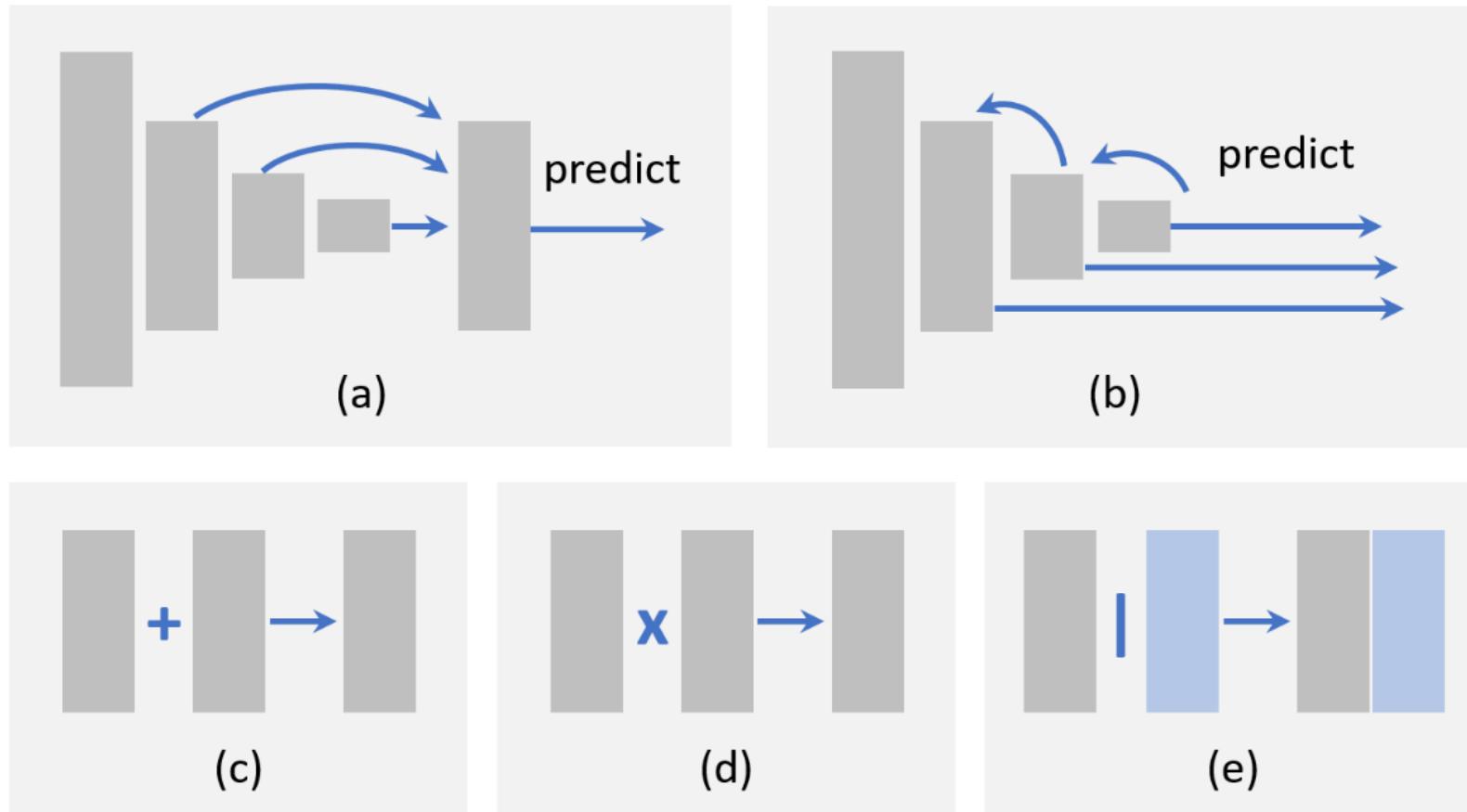
- Extreme foreground-background class imbalance encountered during training of dense detectors is the central cause for low performance.
- **Proposed solution:** reshape the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples.
- **Focal Loss :** focuses training on a sparse set of hard examples and prevents the vast number of easy negatives from overwhelming the detector during training.





Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

Feature Fusion

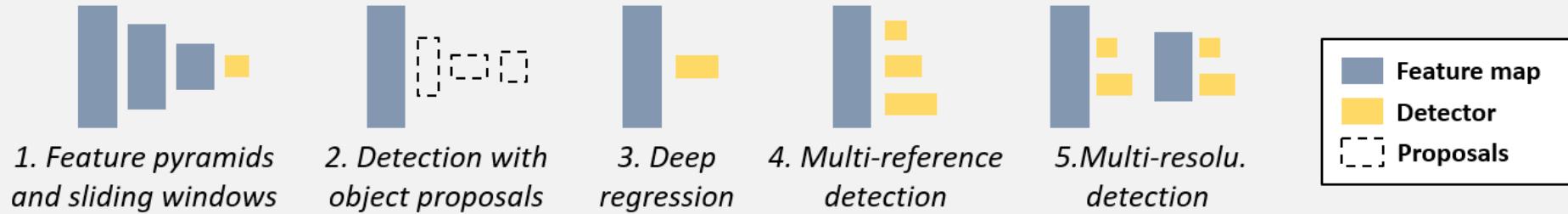


feature fusion methods:

- (a) Bottom-up fusion,
- (b) top-down fusion,
- (c) element-wise sum,
- (d) element-wise product, and
- (e) concatenation.

Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

Evolution of Multi-scale Detection



Year: 2001 2006 2008 2013 2014 2015 2016 2017 2018 2019

Feature Pyramids and Sliding Windows

@VJ Det. (P. Viola et al-CVPR2001), @HOG Det. (N. Dalal et al-CVPR2005), @DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010), @ Exemplar SVM (T. Malisiewicz et al-ICCV2011), @ Overfeat (P. Sermanet et al-ICLR2014) ...

Detection with Object Proposals

Deep Regression

@DNN Det. (C. Szegedy et al-NIPS2013), @YOLO (J. Redmon et al-CVPR2016) ...

@RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014) @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015) ...

Deep Regression

Multi-reference Detection

Faster-RCNN (S. Ren et al-NIPS2015), @SSD (W. Liu et al-ECCV2016), @YOLOv2 (J. Redmon et al-CVPR2017), @TridentNet (Y. Li et al-arXiv19) ...

Multi-resolution Detection

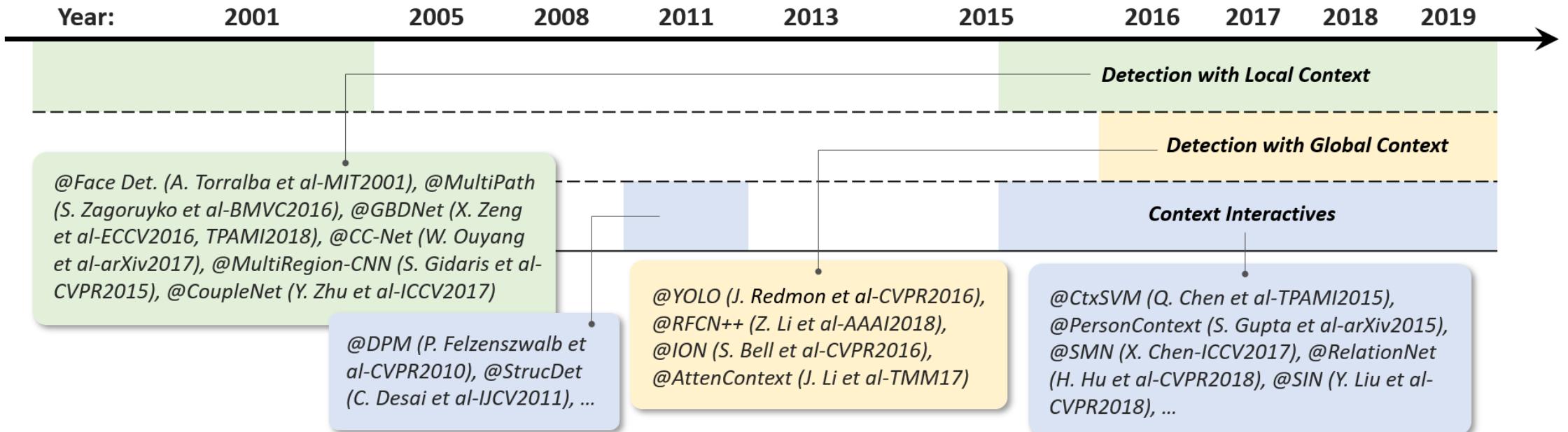
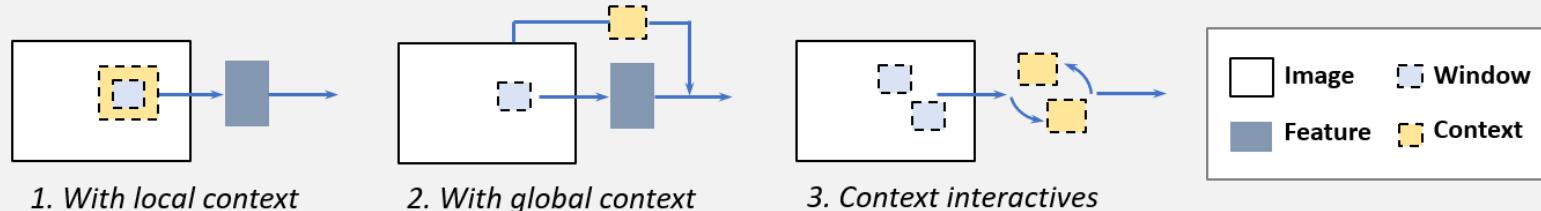
@SSD (W. Liu et al-ECCV2016), @Unified Det. (Z. Cai et al-ECCV2016) @FPN (T. Y. Lin et al-CVPR2017), @RetinaNet(T. Y. Lin et al-ICCV2017), @RefineDet (Zhang et al-CVPR18) ...

Evolution of Bounding Box Regression

weak ← *Invariance of translation and scale* → strong

Year:	2001	2006	2008	2013	2014	2015	2016	2017	2018	2019
Method	<i>Without Bounding Box Regression</i>	<i>From Bounding Box to Bounding Box</i>				<i>From Feature to Bounding Box</i>				
Remarks			Icing on the cake, optional				Essential, integrated with the model			
	@VJ Det. (P. Viola et al-CVPR2001), @HOG Det. (N. Dalal et al-CVPR2005), @ Exemplar SVM (T. Malisiewicz et al-ICCV2011) ...	@DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010)		@Overfeat (P. Sermanet et al-ICLR2014), @RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014) @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015), @YOLO (J. Redmon et al-CVPR2016), @SSD (W. Liu et al-ECCV2016), @YOLOv2 (J. Redmon et al-CVPR2017), @Unified Det. (Z. Cai et al-ECCV2016) @FPN (T. Y. Lin et al-CVPR2017), @RetinaNet(T. Y. Lin et al-ICCV2017), @RefineDet (Zhang et al-CVPR18), @TridentNet (Y. Li et al-arXiv19) ...						

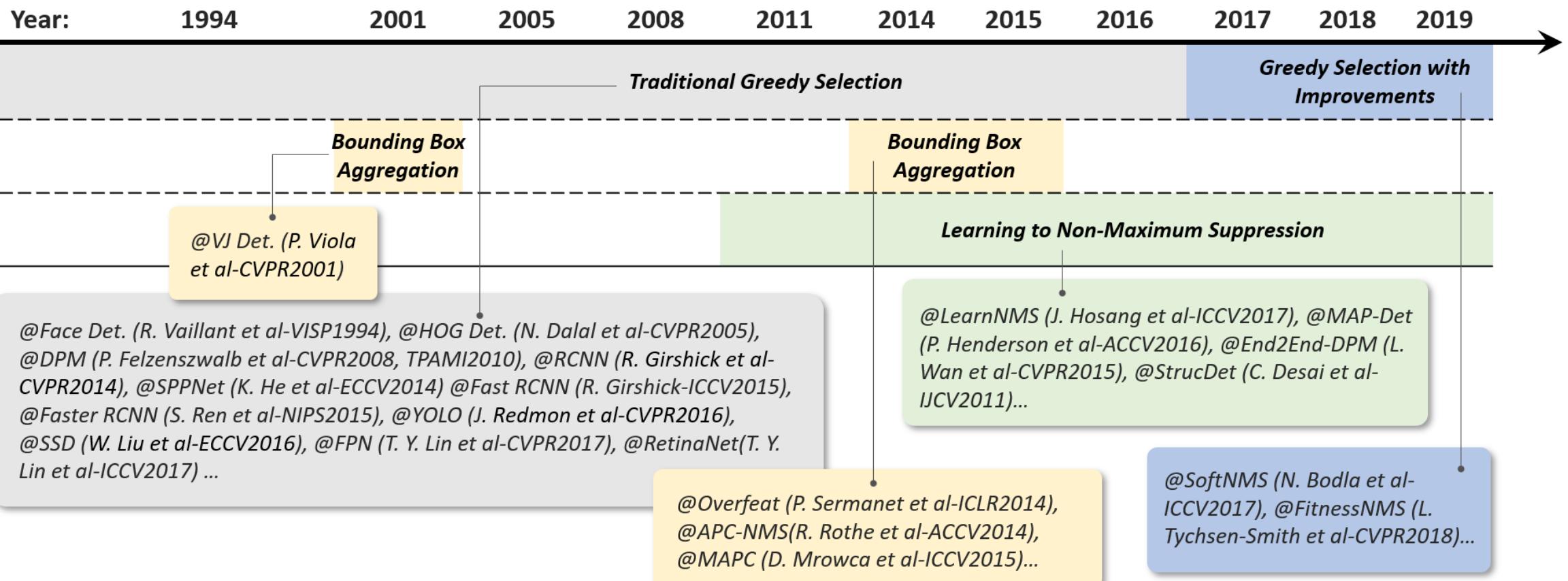
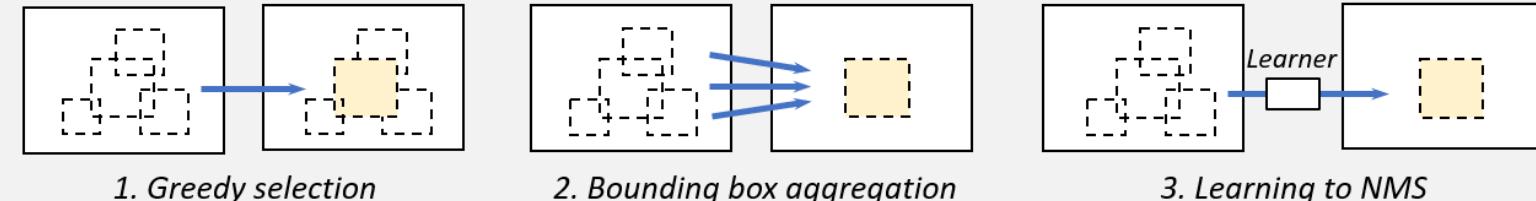
Evolution of Context Priming in Object Detection



Visual objects are usually embedded in a typical context with the **surrounding environments**. Our brain takes advantage of the associations among objects and environments to facilitate visual perception and cognition [160]. Context priming has long been used to improve detection. There are three common approaches in its evolutionary history:

1. detection with local context,
2. detection with global context,
3. context interactives.

Evolution of Non-Max Suppression

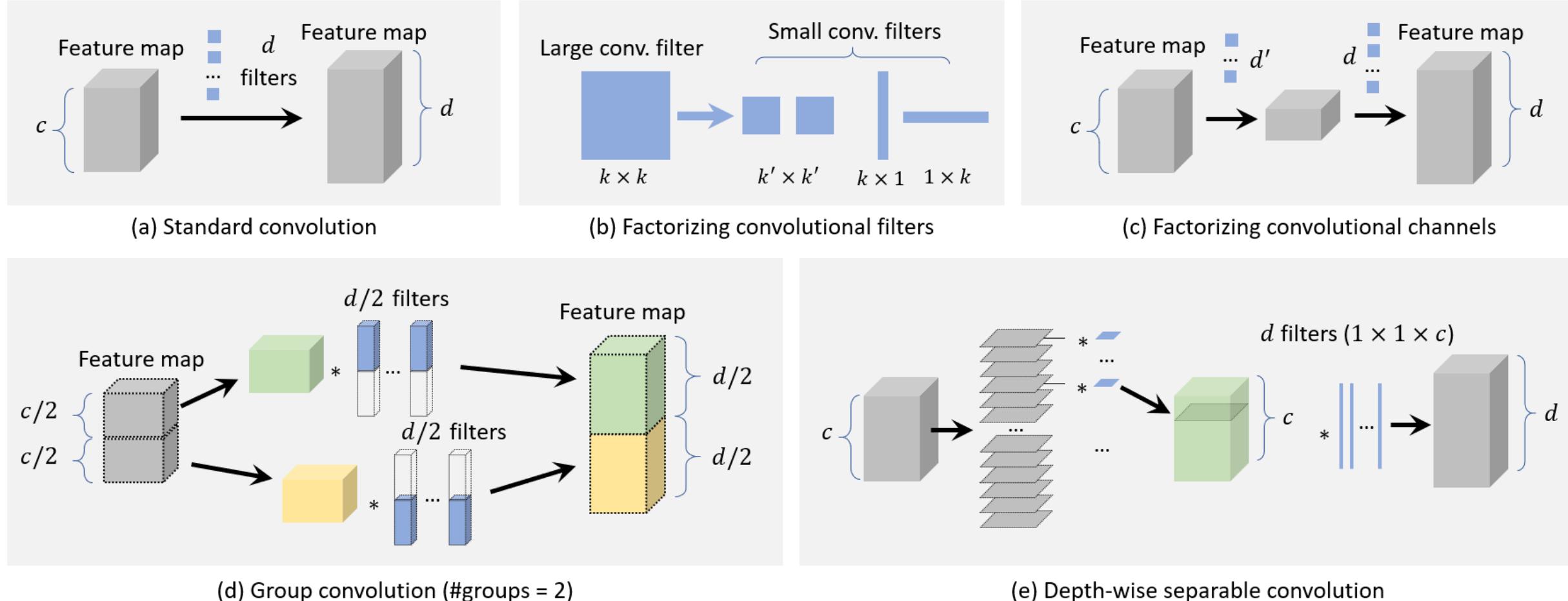


Evolution of Hard Negative Mining

Year:	1994	2001	2005	2008	2014	2015	2016	2017	2018	2019	
Method	Bootstrap				Without Hard Negative Mining		Bootstrap + New Loss Functions				
Remarks	Bootstrap was widely used to deal with the insufficient computing resources of early time				By simply balancing the weights between object and background classes		Focusing on hard examples. Computing power is no longer a problem.				
<p>@Face Det. (H. A. Rowley et al-CMUTechRep1995), @Haar Det. (C. P. Papageorgiou et al-ICCV1998), @VJ Det. (P. Viola et al-CVPR2001), @HOG Det. (N. Dalal et al-CVPR2005), @DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010)...</p>				<p>@RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014) @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015), @YOLO (J. Redmon et al-CVPR2016)...</p>				<p>@SSD (W. Liu et al-ECCV2016), @FasterPed (L. Zhang et al-ECCV2016), @OHEM (A. Shrivastava et al-CVPR2016), @RetinaNet (T. Y. Lin et al-ICCV2017), @RefineDet (Zhang et al-CVPR18)...</p>			

A hard negative is when you take that falsely detected patch, and explicitly create a negative example out of that patch, and add that negative to your training set. When you retrain your classifier, it should perform better with this extra knowledge, and not make as many false positives.

An overview of speed up methods of a CNN's convolutional layer and the comparison of their computational complexity



Object Detection with Deep Learning

CS/ECE 8690

Filiz Bunyak Ersoy

References

1. Szeliski, Computer Vision, Chapter 5.3 and 5.4
2. Liu, Li, Jie Chen, Paul Fieguth, Guoying Zhao, Rama Chellappa, and Matti Pietikäinen. "From BoW to CNN: Two decades of texture representation for texture classification." *International Journal of Computer Vision* 127, no. 1 (2019): 74-109.
3. Mostly complete chart of neural networks <https://www.asimovinstitute.org/neural-network-zoo/>
4. Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).
5. Jiao, Licheng, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. "A Survey of Deep Learning-Based Object Detection." *IEEE Access* 7 (2019): 128837-128868.
6. 2014 - CVPR Tutorial on Deep Learning for Vision - Object Detection, **Pierre Sermanet, Google Research**
7. Slides by Míriam Bellver, Computer Vision Reading Group, UPC, 28th October, 2016

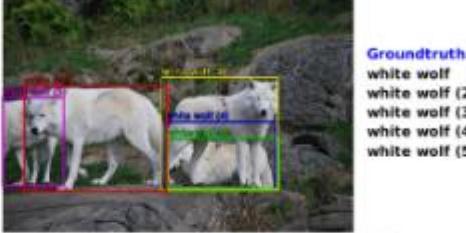
Deep Learning Beyond Image Classification

Computer Vision Tasks & Deep Learning

classification



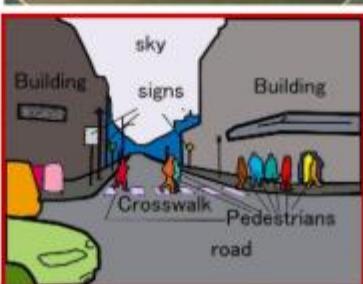
localization



detection



segmentation



Difficulty

Semantic Segmentation



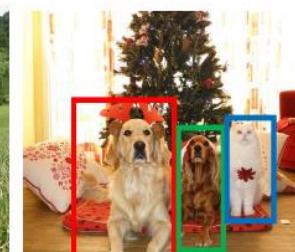
GRASS, CAT,
TREE, SKY

Classification
+ Localization



CAT

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

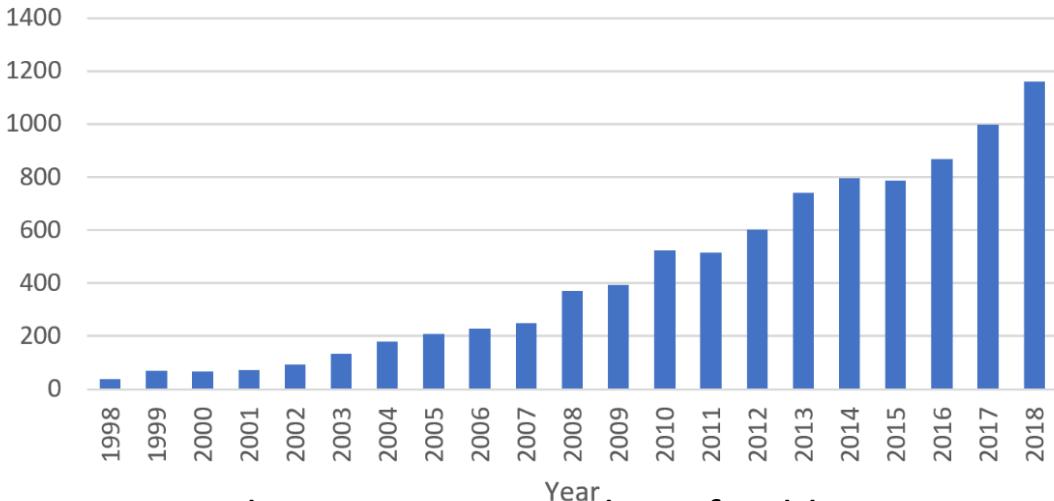
This image is CC0 public domain

Figure: Fei-Fei Li & Justin Johnson & Serena Yeung, Stanford

Object Detection using Deep Learning

Object Detection

- **PASCAL** pascallin.ecs.soton.ac.uk/challenges/VOC
- **ImageNet** www.image-net.org/challenges/LSVRC/2014
- **Sun** sundatabase.mit.edu
- **Microsoft COCO** mscoco.org



The increasing number of publications in object detection from 1998 to 2018. (Data from Google scholar advanced search: all in title: “object detection” AND “detecting objects”). Zou Arxiv 2019

	# classes	average # categories per image	average # instances per image	average object scale	average resolution	# images				# objects			
						total	train	val	test	total	train	val	test
PASCAL	20	1.521	2.711	0.207	469x387	22k	6k	6k	10k	42k?	14k	14k	-
ImageNet13	200	1.534	2.758	0.170	482x415	516k	456k	20k	40k	648k?	480k	56k	-
Sun	4919	9.8	16.9	0.1040	732x547	16873	-			285k	-		
COCO	91	3.5	7.6	0.117	578x483	328k	164k	82k	82k	2500k	~1250k	~625k	~625k

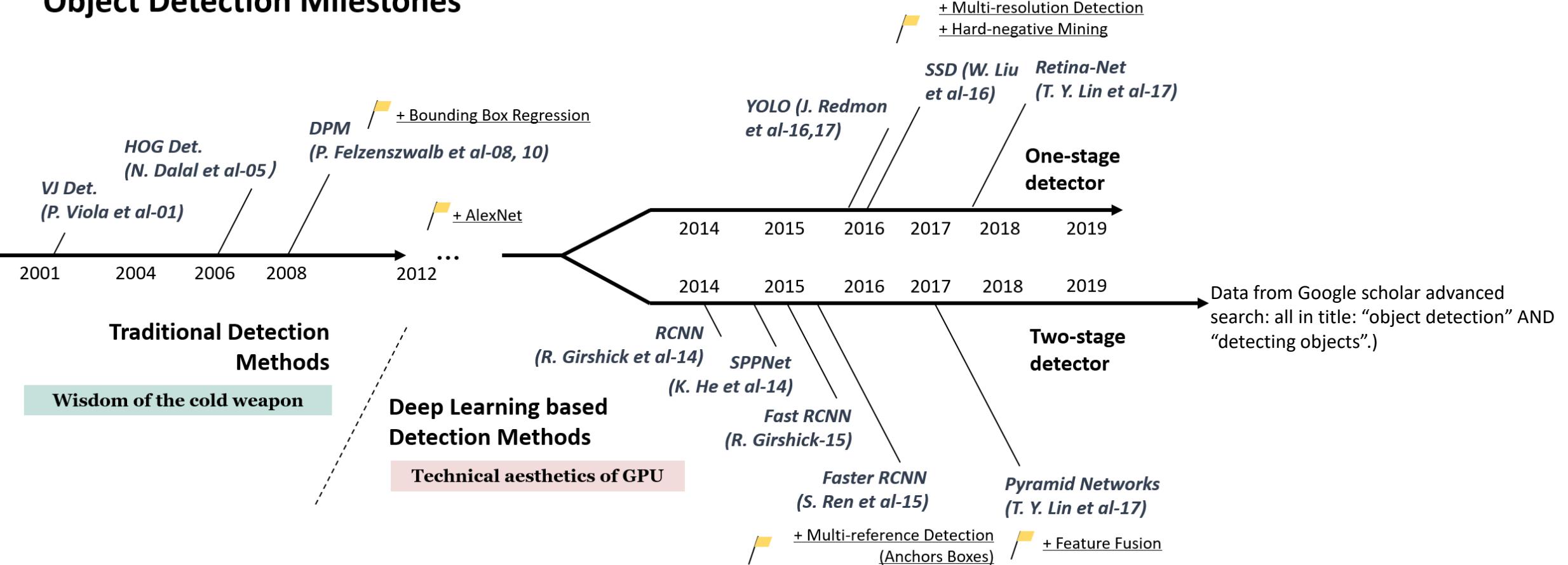
The pascal visual object classes (voc) challenge. Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. *International journal of computer vision* 88, no. 2 (2010): 303-338.

Imagenet: A large-scale hierarchical image database. Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248-255. IEEE, 2009.

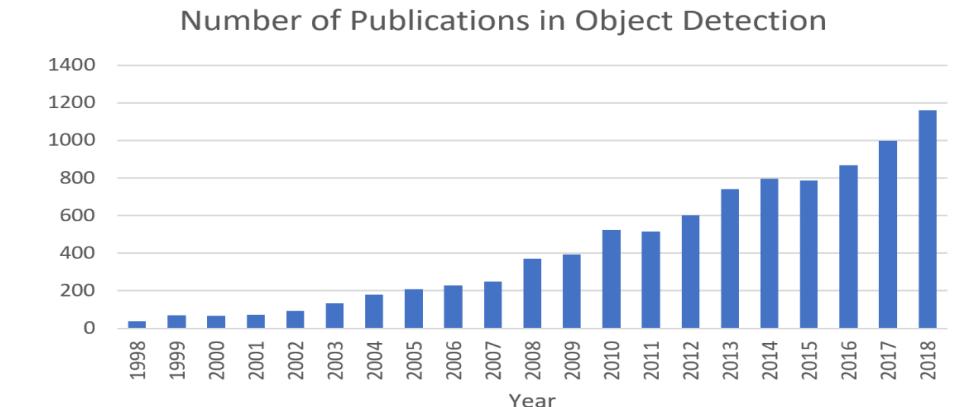
SUN Database: Large-scale Scene Recognition from Abbey to Zoo, Jianxiong Xiao, James Hays, Krista Ehinger, Aude Oliva, and Antonio Torralba. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2010*.

Microsoft COCO: Common Objects in Context, Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C. Lawrence Zitnick, <http://arxiv.org/abs/1405.0312>, May 2014

Object Detection Milestones



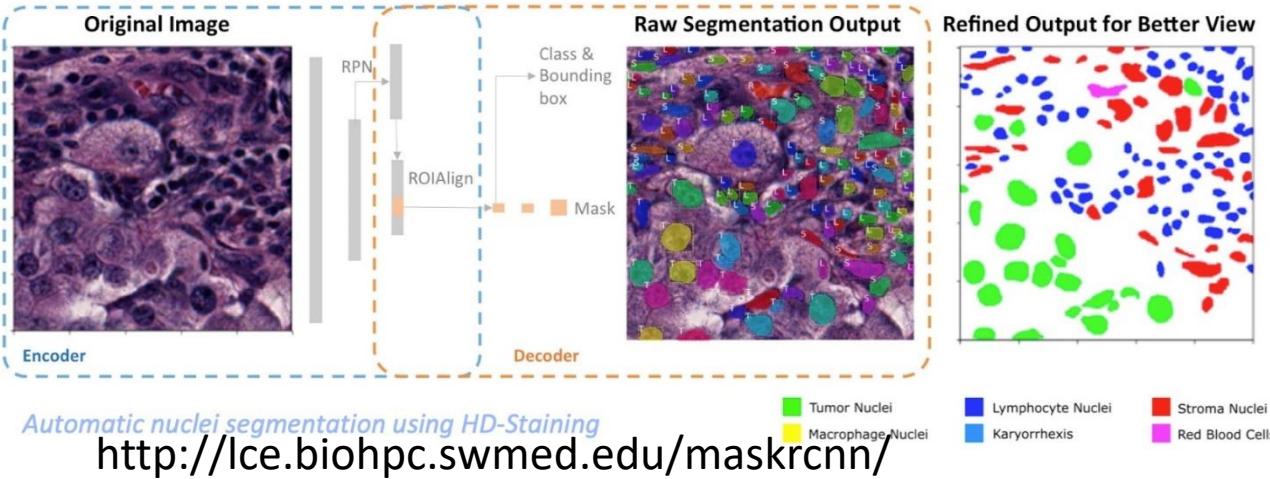
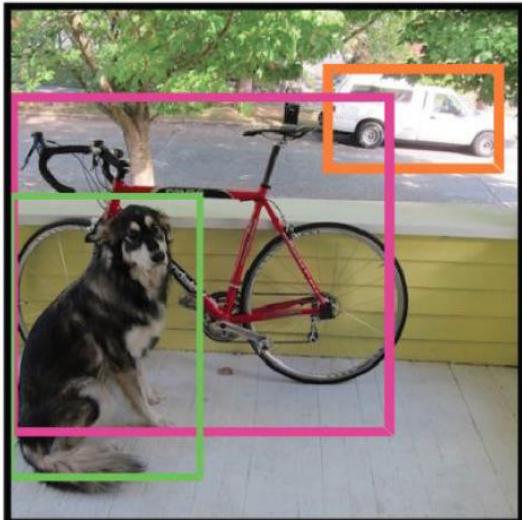
Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).



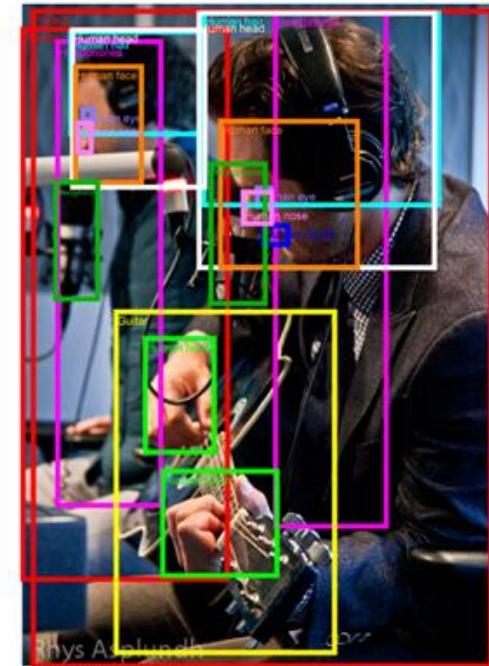
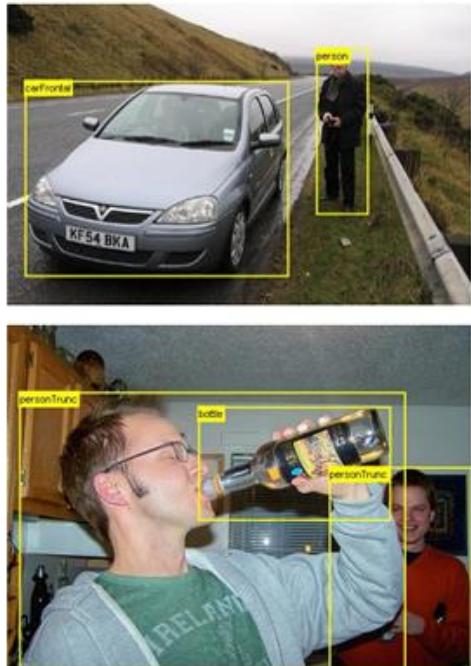
Deep Object Detection

1. Two-stage detection (RCNN and its variants): coarse-to-fine process
2. One-stage detection (SSD, YOLO, RetinaNet etc.): complete in one step

Pang, Shanchen, Tong Ding, Sibo Qiao, Fan Meng, Shuo Wang, Pibao Li, and Xun Wang. "A novel YOLOv3-arch model for identifying cholelithiasis and classifying gallstones on CT images." *PLoS one* 14, no. 6 (2019): e0217647.



Object Detection Benchmark Datasets



Some example images and annotations in (a) PASCAL-VOC07, (b) ILSVRC, (c) MS-COCO, and (d) Open Images.

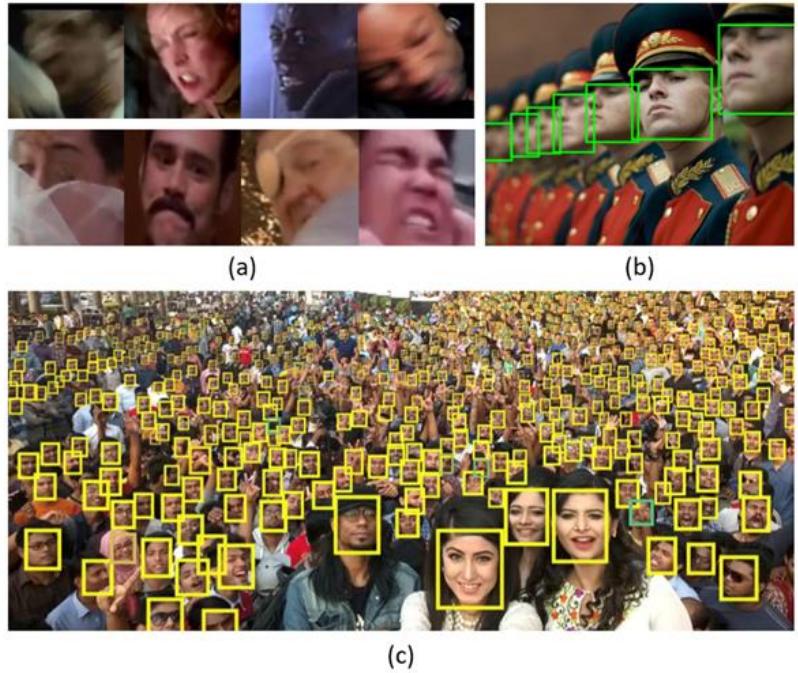
Applications & Challenges



- (a) small pedestrians,
- (b) hard negatives,
- (c) Dense and occluded pedestrians.



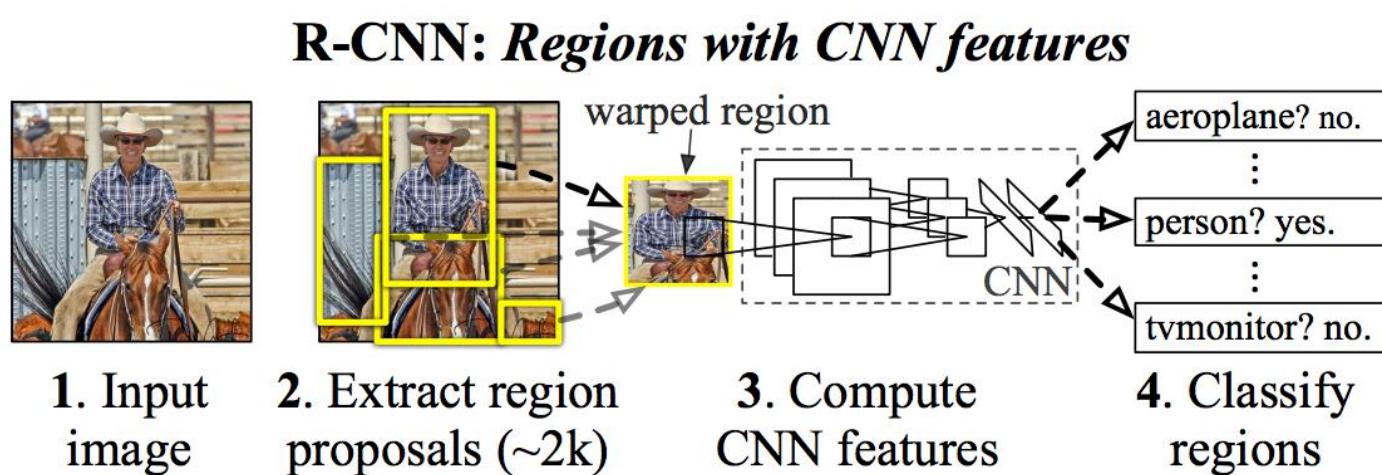
- (a) Illumination changes. Image from pxhere (free of copyrights).
- (b) Motion blur. Image from GTSRB Dataset [81].
- (c) Detection under bad weather. Image from Flickr and Max Pixel (free of copyrights).



- (a) Intra-class variation, image from WildestFaces Dataset [70].
- (b) Face occlusion, image from UFDD Dataset [69].
- (c) Multi-scale face detection. Image from P. Hu et al. CVPR2017 [322].

Two-Stage Detection

- Stage-1: sparse set of candidates
- Stage-2: classify each candidate into one of FG/BG classes
- Examples: R-CNN, Fast R-CNN, Faster R-CNN



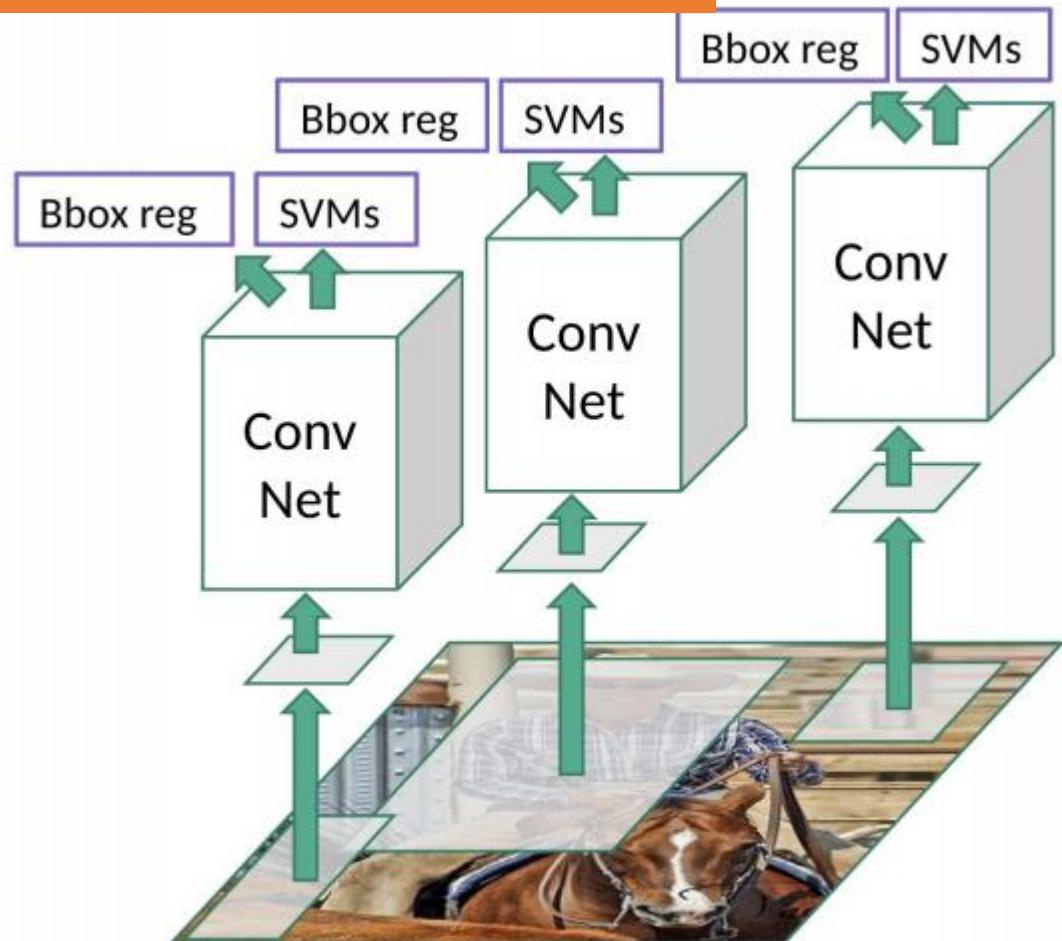
R-CNN

1. Extraction of a set of object proposals (object candidate boxes) by **selective search**.
2. Each proposal is
 1. rescaled to a fixed size image and
 2. fed into a CNN model trained on ImageNet (say, AlexNet [40]) to extract features.
3. Finally, linear SVM classifiers are used to predict the presence of an object within each region and to recognize object categories.

Drawbacks

- redundant feature computations on a large number of overlapped proposals (over 2000 boxes from one image) leads to an extremely slow detection speed.

Mean Average Precision on VOC07
33.7% to 58.5%



Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587. 2014. 29K citations.

Training

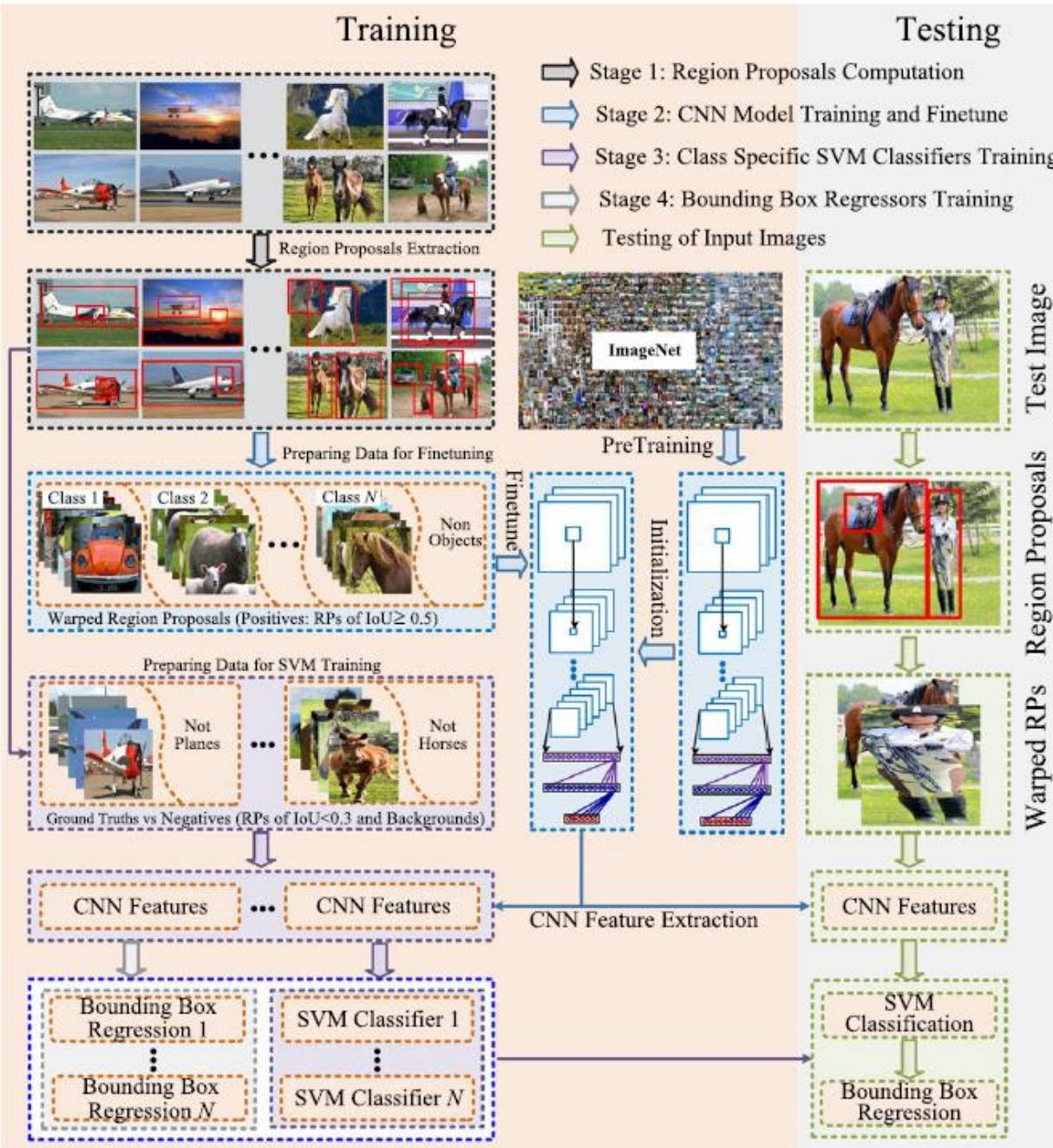


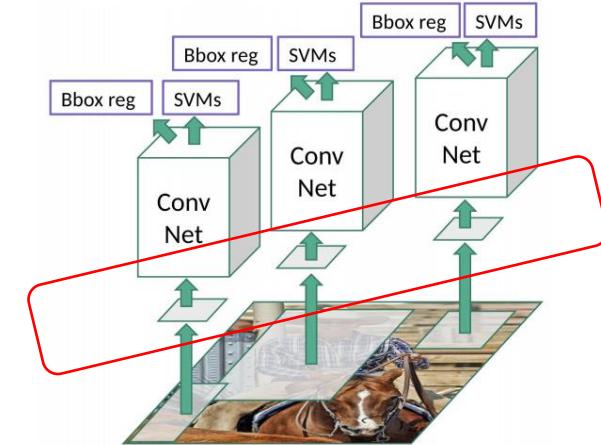
Illustration of the RCNN detection framework (Girshick et al. 2014, 2016)

Figure credit: Liu, Li, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. "Deep learning for generic object detection: A survey." *International journal of computer vision* 128 (2020): 261-318.

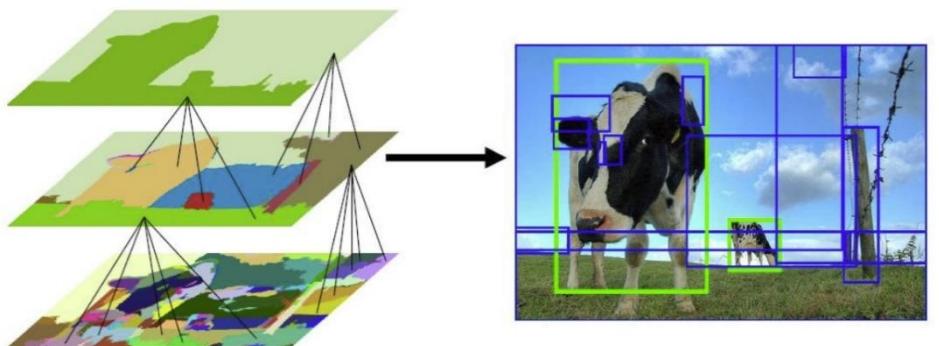
- Last slide covered on March 14

Selective Search

1. First, color similarities, texture similarities, region size, and region filling are used as **non-object-based segmentation**. Therefore we obtain **many small segmented areas** as shown at the bottom left of the image above.
2. Then, bottom-up approach is used that **small segmented areas are merged together to form larger segmented areas**.
3. Thus, **about 2K region proposals (bounding box candidates)** are generated as shown in the image.

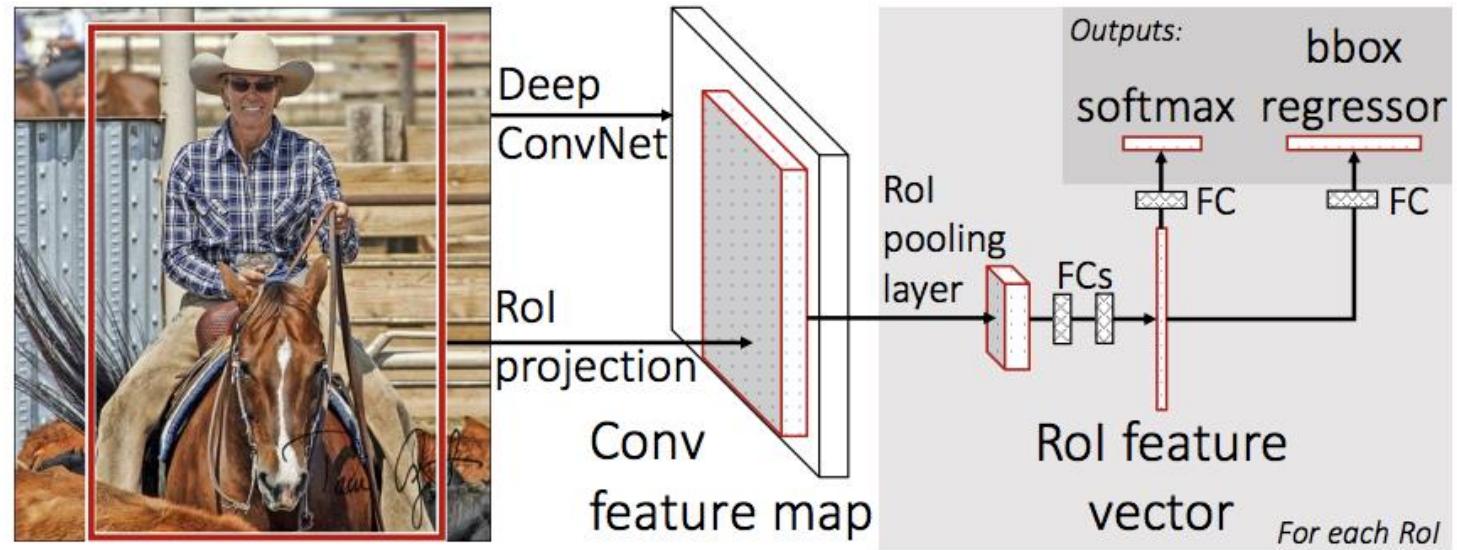


Uijlings, Jasper RR, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. "Selective search for object recognition." *International journal of computer vision* 104, no. 2 (2013): 154-171. 3550 citations.



Fast R-CNN

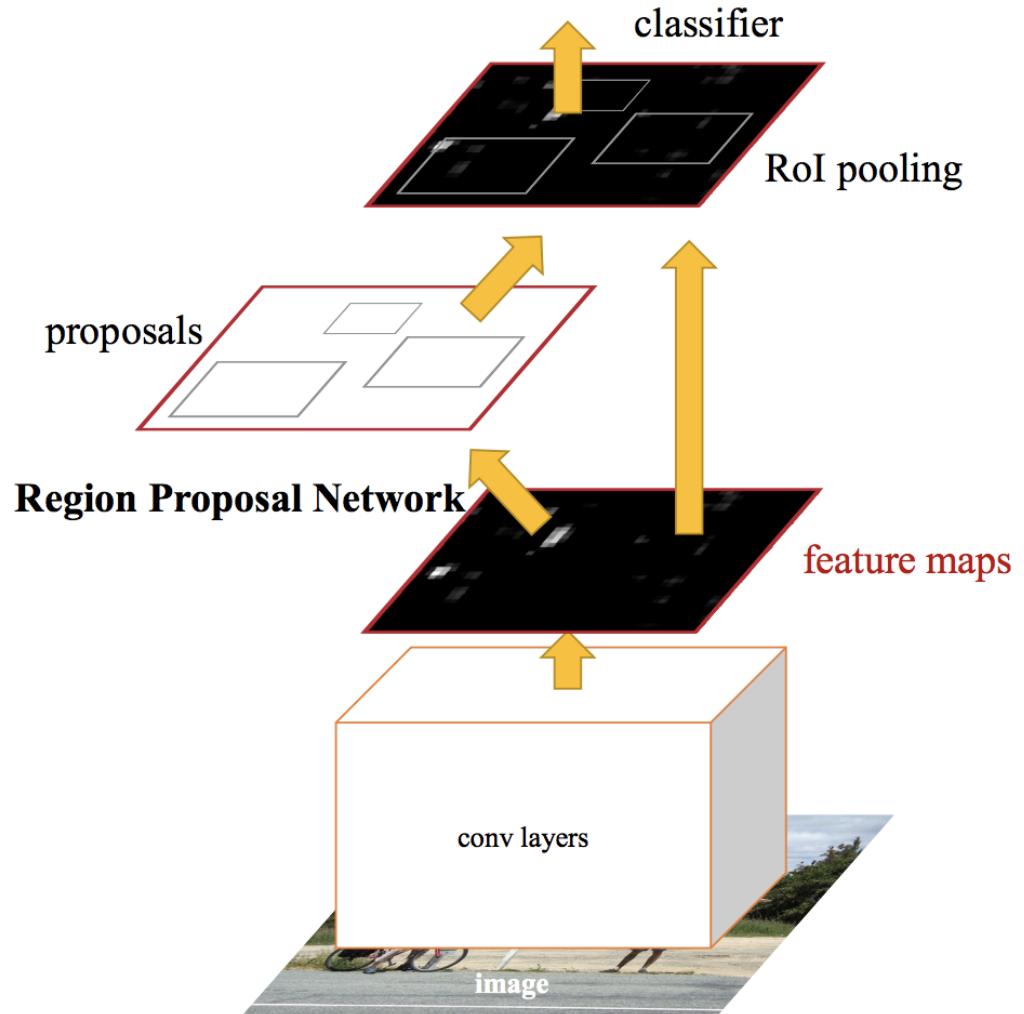
Mean Average Precision on VOC07
33.7% to 58.5% to 70.0%
Speed x200 compared to RCNN



- Instead of feeding the region proposals to the CNN, input image is fed to the CNN to generate a convolutional feature map.
- From the convolutional feature map, region of proposals are identified and warped them into squares and by using a ROI pooling layer.

Faster R-CNN

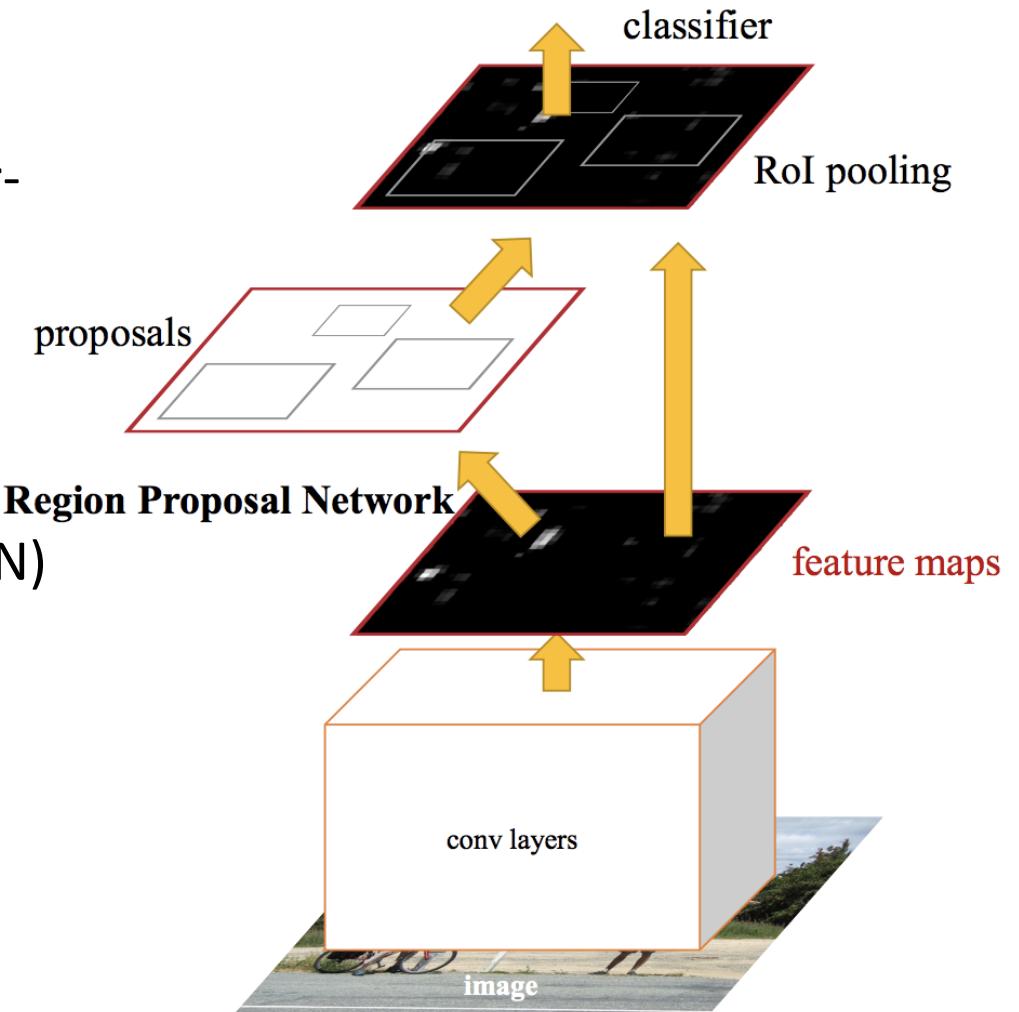
- Eliminates the selective search algorithm and lets the network learn the region proposals.
- Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map.
- Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals.
- The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.



Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In *Advances in neural information processing systems*, pp. 91-99. 2015. 55K citations.

Faster R-CNN

- Faster RCNN is the first end-to-end, and the first near-realtime deep learning detector
- (COCO mAP@.5=42.7%, COCO [mAP@\[.5,.95\]=21.9%](#), VOC07 mAP=73.2%, VOC12 mAP=70.4%, 17fps with ZFNet [45]).
- The main contribution: Region Proposal Network (RPN) that enables nearly cost-free region proposals.

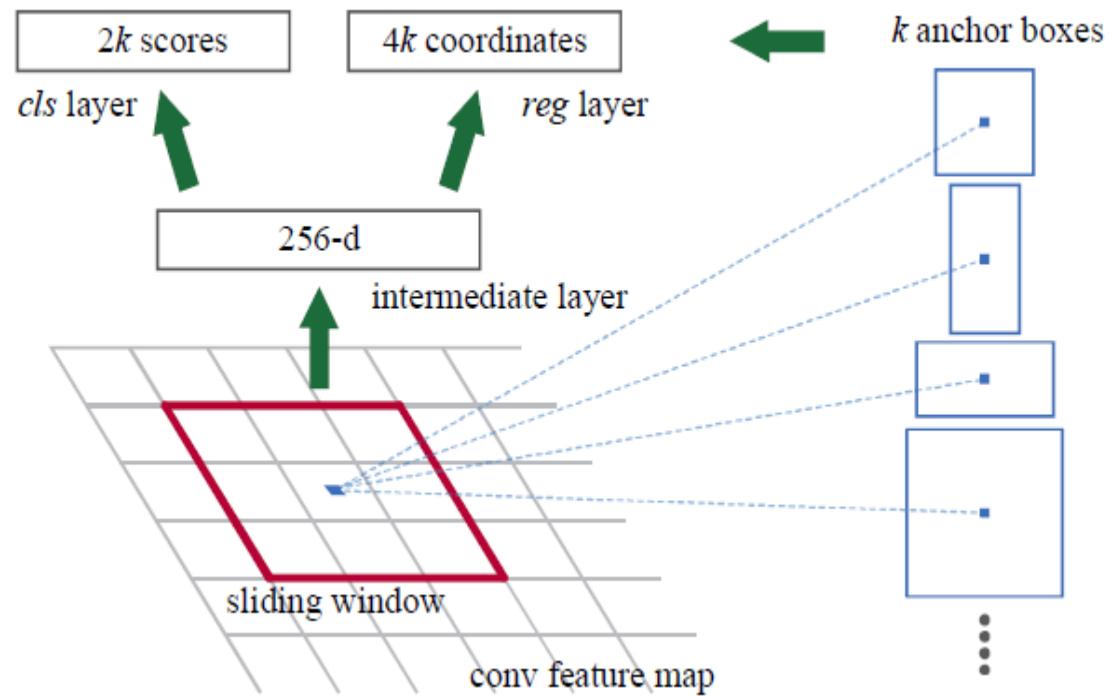


Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In *Advances in neural information processing systems*, pp. 91-99. 2015. 55K citations.

Region Proposal Network

1. First, the picture goes through conv layers and feature maps are extracted
2. Then a **sliding window** is used in RPN for each location over the feature map.
3. For each location, k ($k=9$) anchor boxes are used (**3 scales of 128, 256 and 512, and 3 aspect ratios of 1:1, 1:2, 2:1**) for generating region proposals.
4. A **cls** layer outputs $2k$ scores **whether there is object or not** for k boxes.
5. A **reg** layer outputs $4k$ for the **coordinates** (box center coordinates, width and height) of k boxes.
6. With a size of $W \times H$ feature map, there are WHk anchors in total.

Thus, **RPN network is to pre-check which location contains object**. And the corresponding locations and **bounding boxes will pass to detection network** for detecting the object class and returning the bounding box of that object.



Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In *Advances in neural information processing systems*, pp. 91-99. 2015. 13444 citations.

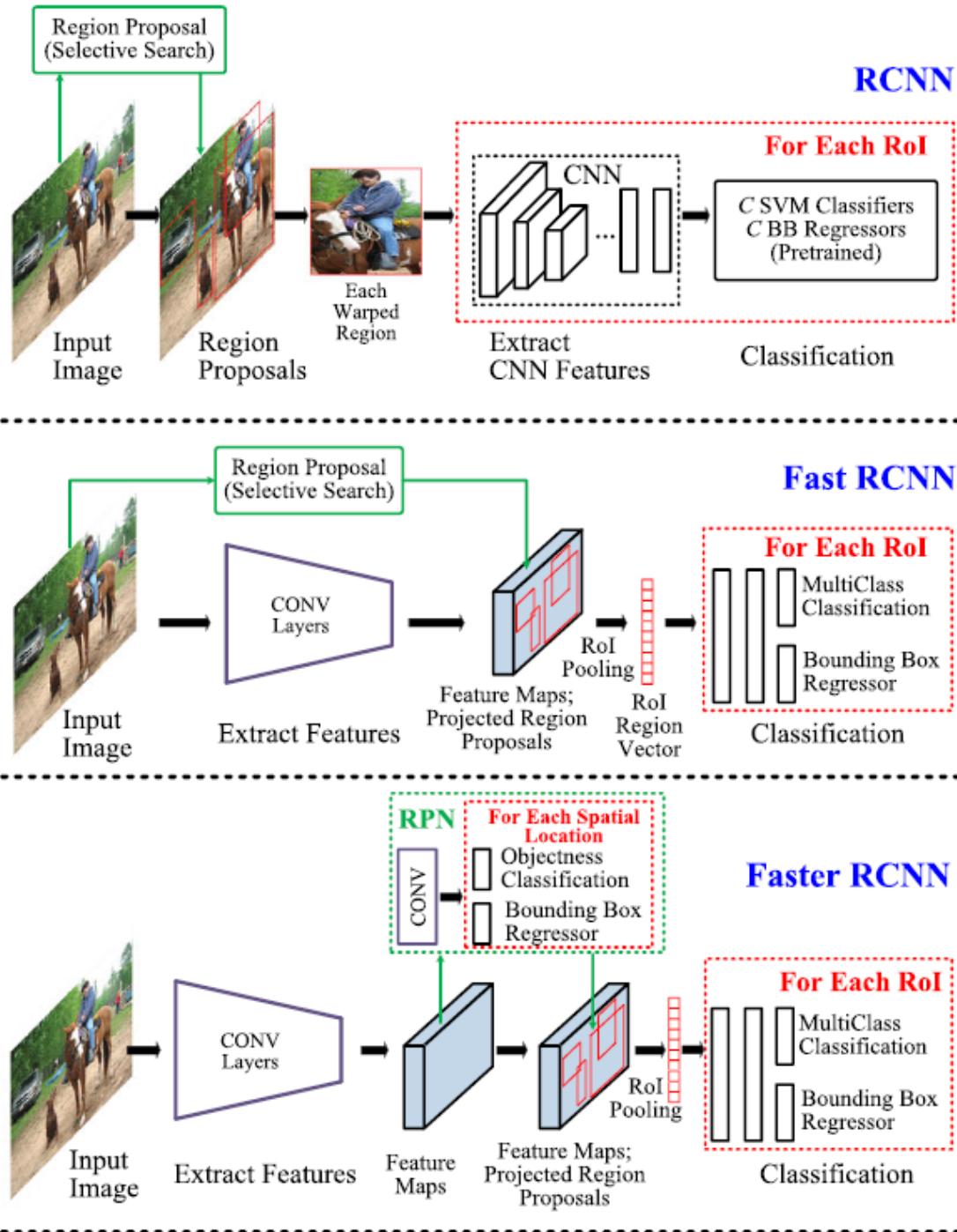
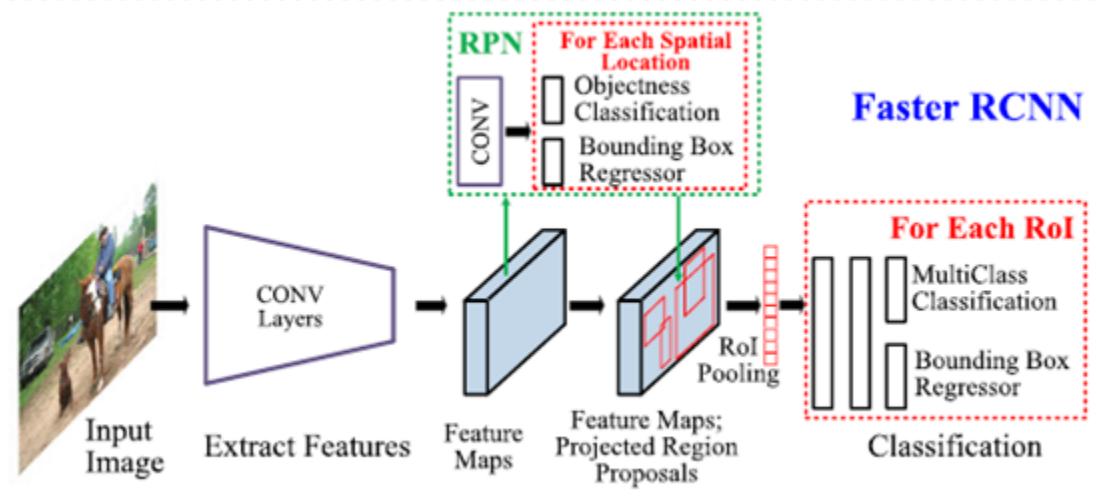


Figure credit: Liu, Li, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. "Deep learning for generic object detection: A survey." *International journal of computer vision* 128 (2020): 261-318.



RFCN (Region based Fully Convolutional Network)

While Faster RCNN is an order of magnitude faster than Fast RCNN, the fact that the region-wise sub-network still needs to be applied per ROI (several hundred ROIs per image) led Dai et al. (2016c) to propose the RFCN detector which is *fully convolutional* (no hidden FC layers) with almost all computations shared over the entire image.

RFN differs from Faster RCNN only in the ROI sub-network. In Faster RCNN, the computation after the ROI pooling layer cannot be shared, so Dai et al. (2016c) proposed using all CONV layers to construct a shared ROI sub-network, and ROI crops are taken from the last layer of CONV features

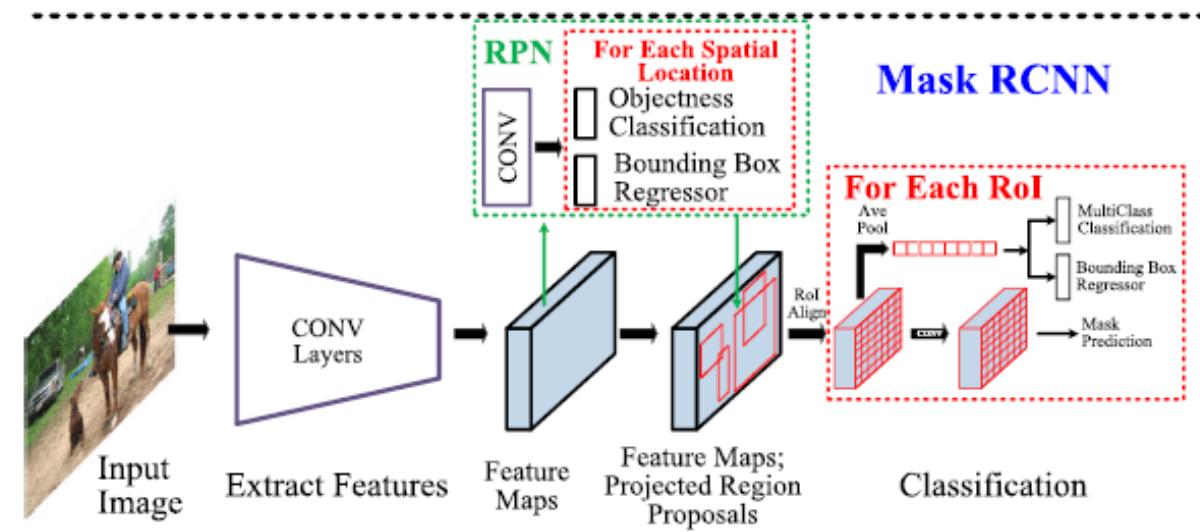
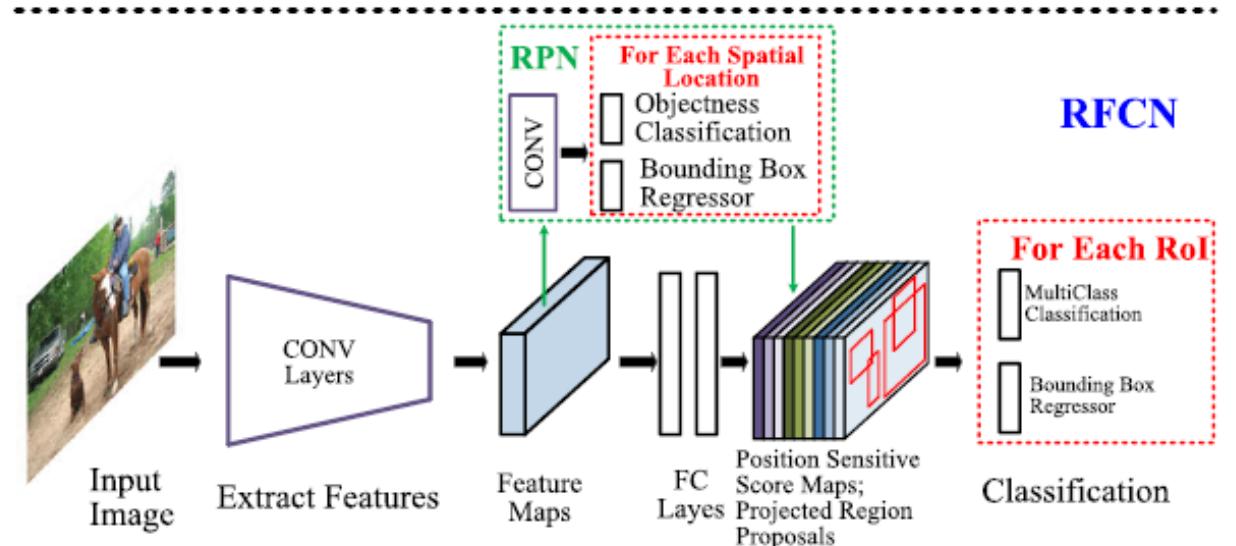
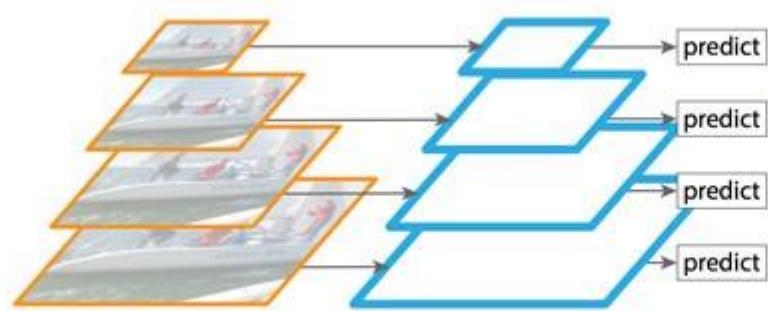
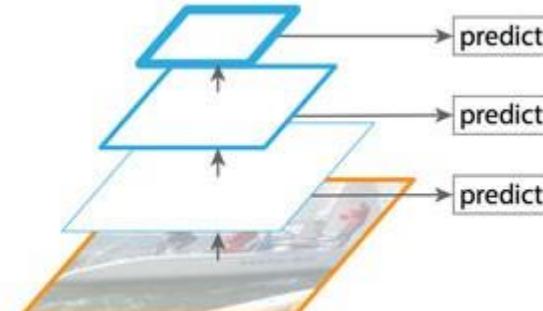


Figure credit: Liu, Li, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. "Deep learning for generic object detection: A survey." *International journal of computer vision* 128 (2020): 261-318.

Feature Pyramid Network (FPN)



Pyramid of images



Pyramid of feature maps

- Replaces the feature extractor of detectors like Faster R-CNN
- Generates multiple feature map layers (**multi-scale feature maps**) with better quality information than the regular feature pyramid for object detection.

- Lin, Tsung-Yi, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. "Feature Pyramid Networks for Object Detection." In *CVPR*, vol. 1, no. 2, p. 3. 2017.
- https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c

Feature Pyramid Network (FPN)

- Before FPN, most of the deep learning-based detectors run detection only on a network's top layer.
- Although the features in deeper layers of a CNN are beneficial for category recognition, it is not conducive to localizing objects.
- To this end, a topdown architecture with lateral connections is developed in FPN for building high-level semantics at all scales.
- Using FPN in a basic Faster R-CNN system, it achieves state-of-the-art single model detection results on the MSCOCO dataset
- FPN has now become a basic building block of many latest detectors.

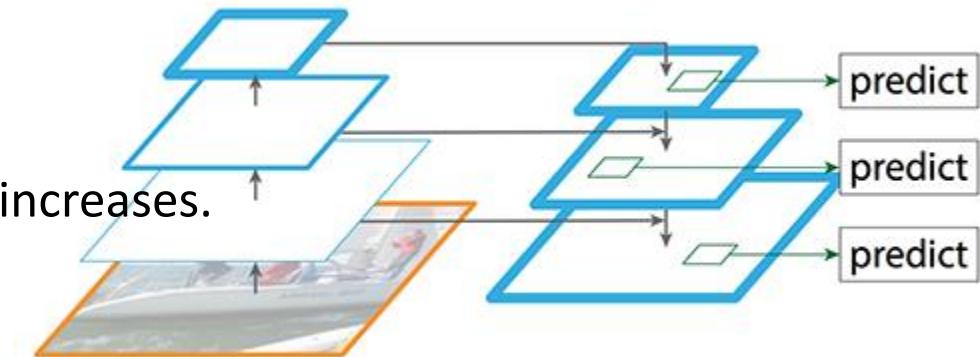
FPN DataFlow

Bottom-up pathway:

- usual convolutional network for feature extraction.
- as we go up, the spatial resolution decreases, the **semantic value** increases.

Top-down pathway:

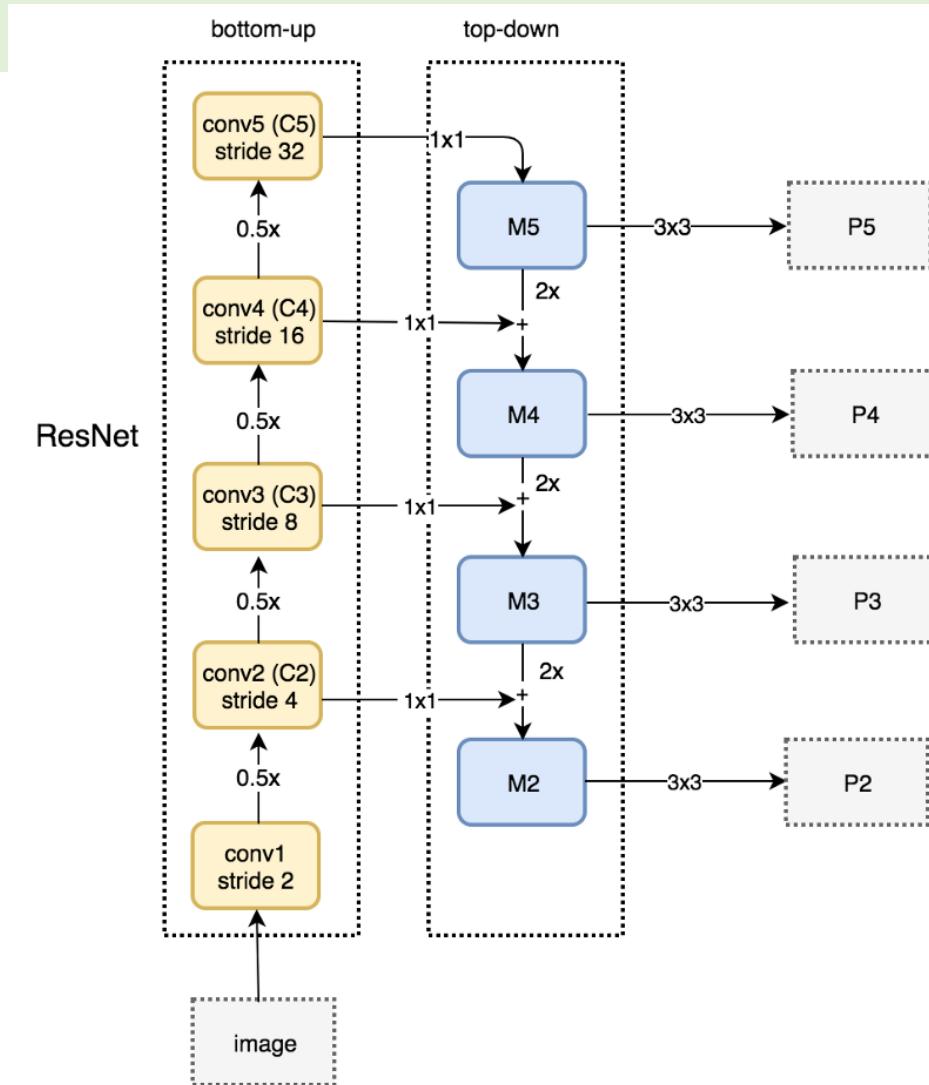
- construct higher resolution layers from a semantic rich layer.
- reconstructed layers are semantic strong but the locations of objects are not precise (after downsampling and upsampling).
- lateral connections between reconstructed layers and the corresponding feature maps
 - help the detector to predict the location better.
 - acts as skip connections to make training easier (similar to ResNet)



Top-down pathway

- **C5->M5:** 1×1 convolution filter to reduce C5 channel depth to 256-d
- **M5->P5:** 3×3 convolution, the first feature map layer used for object prediction.
- Top-down path:
 - upsample the previous layer by 2 using nearest neighbors upsampling.
 - apply a 1×1 convolution to the corresponding feature maps
 - add them element-wise.
 - apply a 3×3 convolution again to output the next feature map layers for object detection.

This filter reduces the aliasing effect of upsampling.

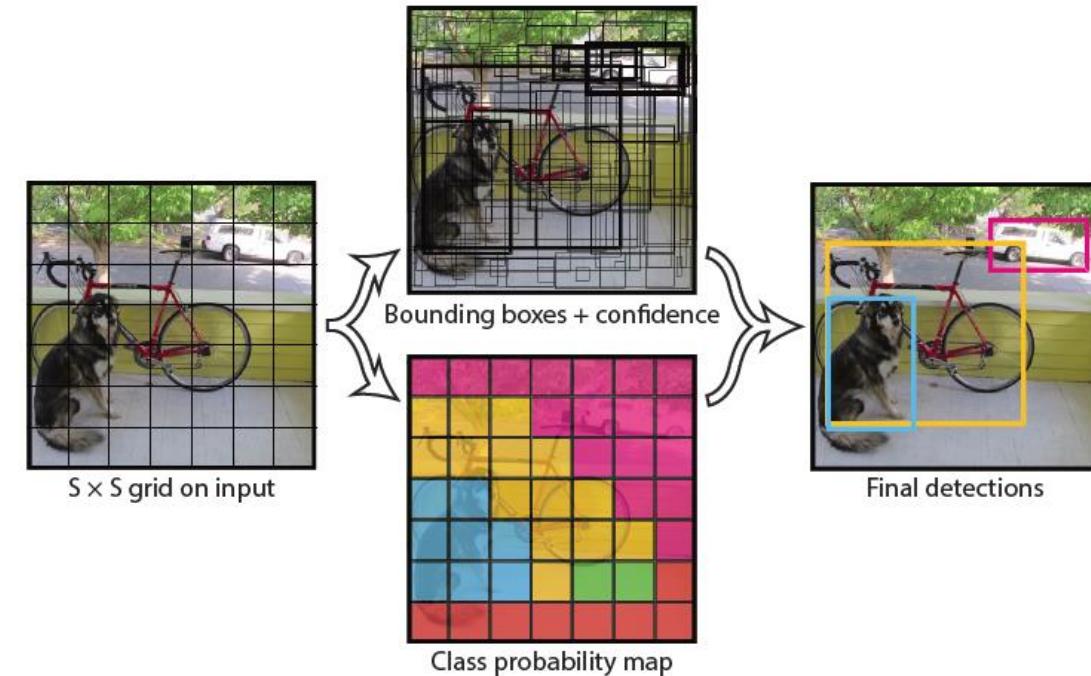


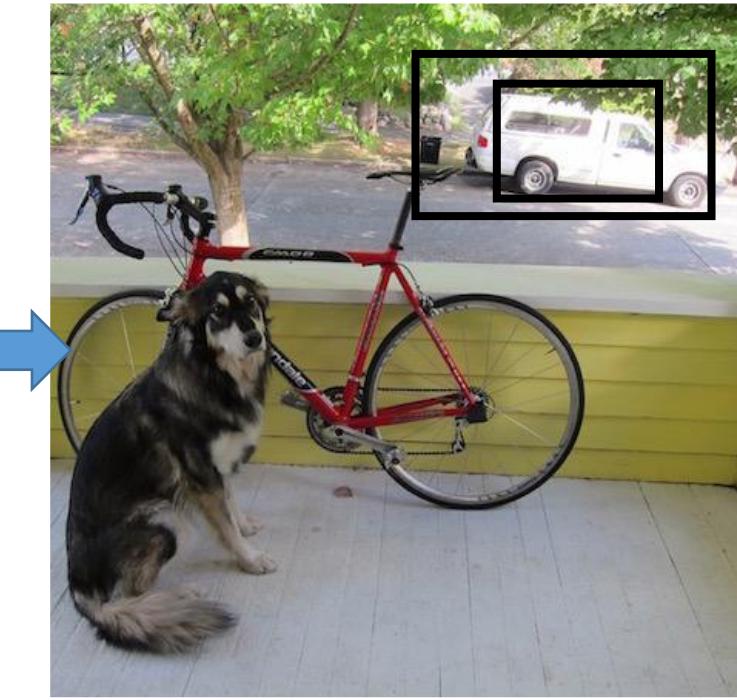
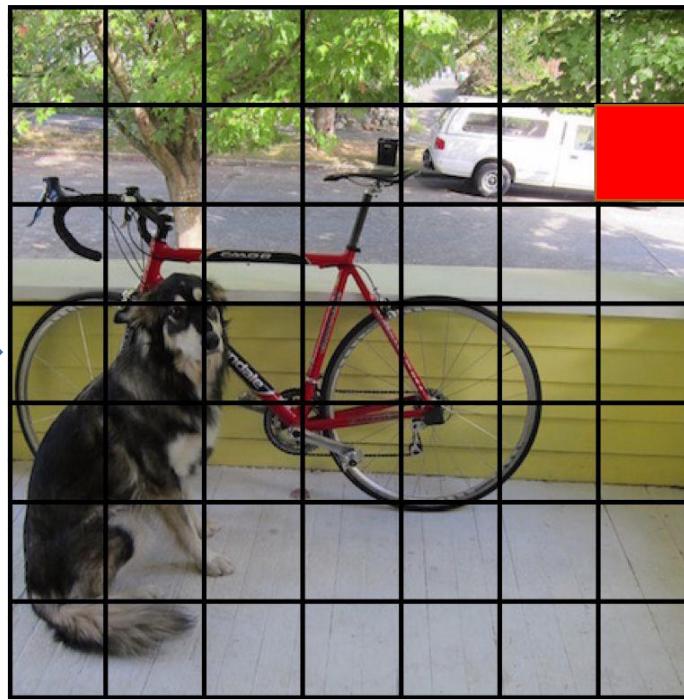
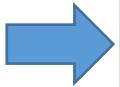
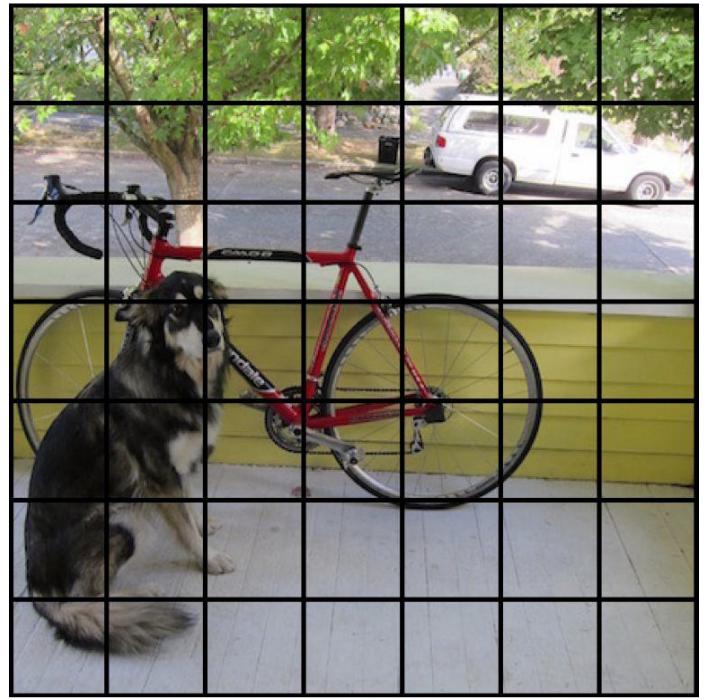
Single-Stage Detectors

- Regular dense sampling of objects/scales/aspect ratios
- Examples: YOLO, SSD, RetinaNet

CNN based One-stage Detectors: You Only Look Once (YOLO)

- Instead of having two networks: Region Proposals Network + Classifier Network
- **In Single-shot architectures, bounding boxes and confidences for multiple categories are predicted directly with a single network**
- e.g. : Overfeat, YOLO



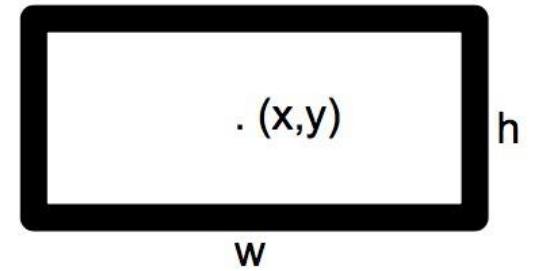


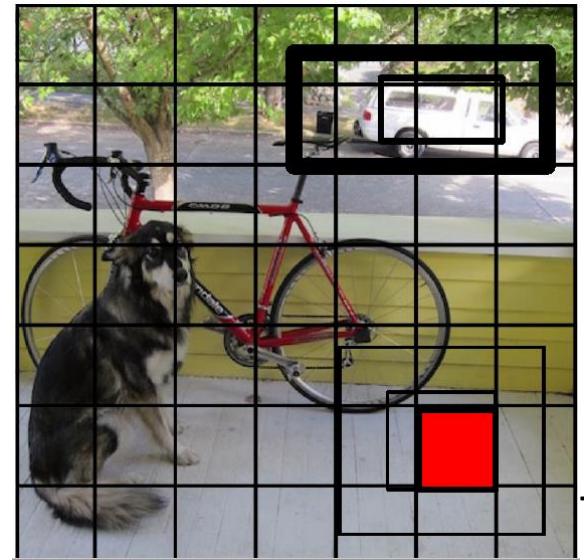
We split the image into an $S \times S$ grid

Each cell predicts
B boxes(x, y, w, h) and
confidences of each box: $P(\text{Object})$

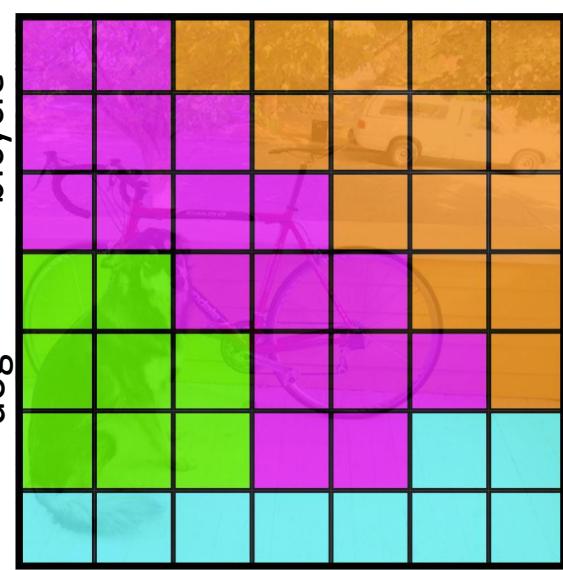


Each box predicts
(x, y, h, w) +
 $P(\text{Object})$

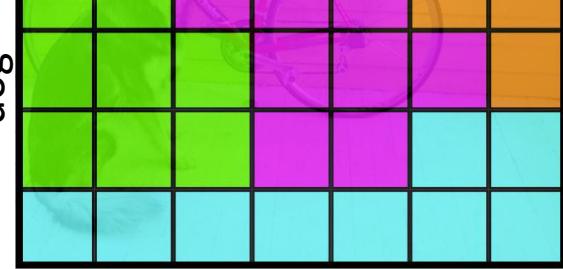




bicycle

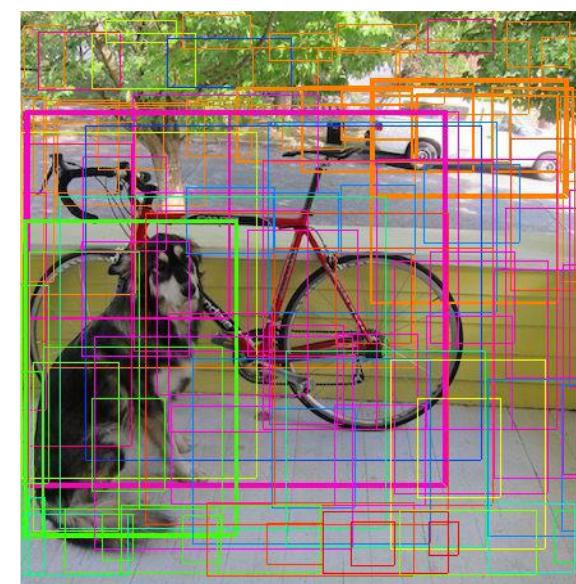
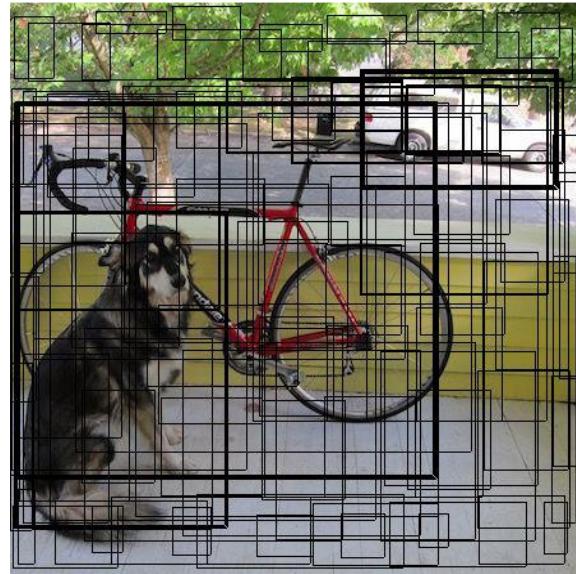


dog

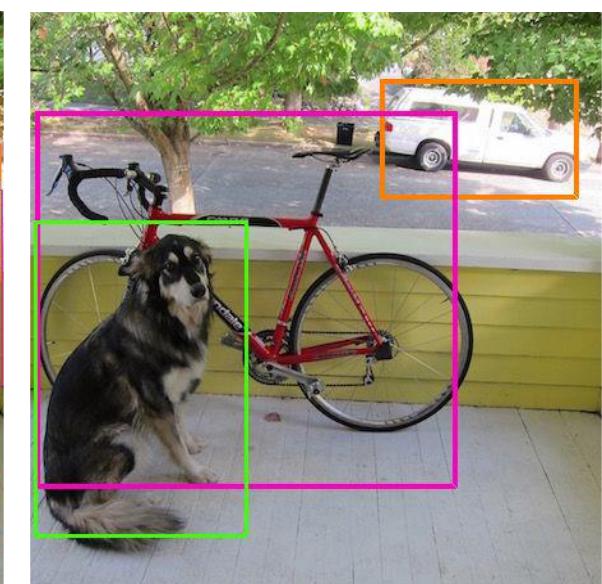


car

Dining table



Combine box and
class predictions
 $P(\text{class}|\text{Object}) * P(\text{Object})$
 $= P(\text{class})$



Thresholding and
Non Maximum Suppression

CNN based One-stage Detectors: You Only Look Once (YOLO)

- First one-stage detector in deep learning era.
- Apply a single neural network to the full image.
- Divides the image into regions and predicts bounding boxes and probabilities for each region simultaneously.
- YOLO is extremely fast: a fast version of YOLO runs at 155fps with VOC07 mAP=52.7%, while its enhanced version runs at 45fps with VOC07 mAP=63.4% and VOC12 mAP=57.9%.
- In spite of its great improvement of detection speed, YOLO suffers from a drop of the localization accuracy compared with two-stage detectors, especially for some small objects. YOLO's subsequent versions [48, 49] and the latter proposed SSD [21] has paid more attention to this problem.

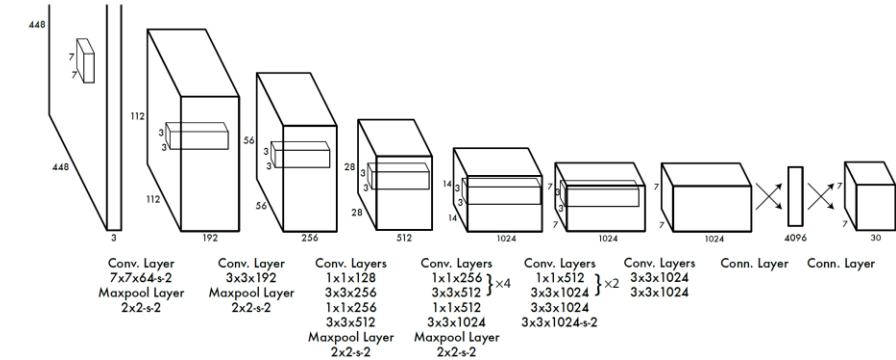


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

YOLOv3 (You Only Look Once)

Single shot detector, and one of the faster object detector

Uses darknet-53 deep architecture (fully convolutional layers)

makes detections at three different scales

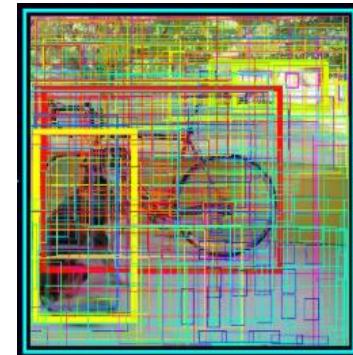
Uses 9 anchor boxes. Three for each scale

The up sampled layers concatenate with the previous layers, help preserve the fine grained features

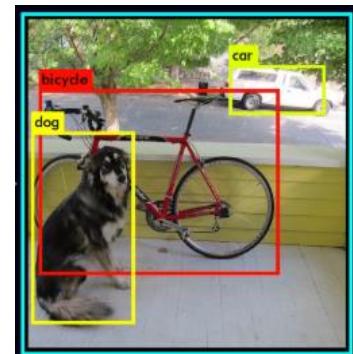
Helps address the issue of detecting small objects.



Generate grid on input



Compute Bboxes & confidence



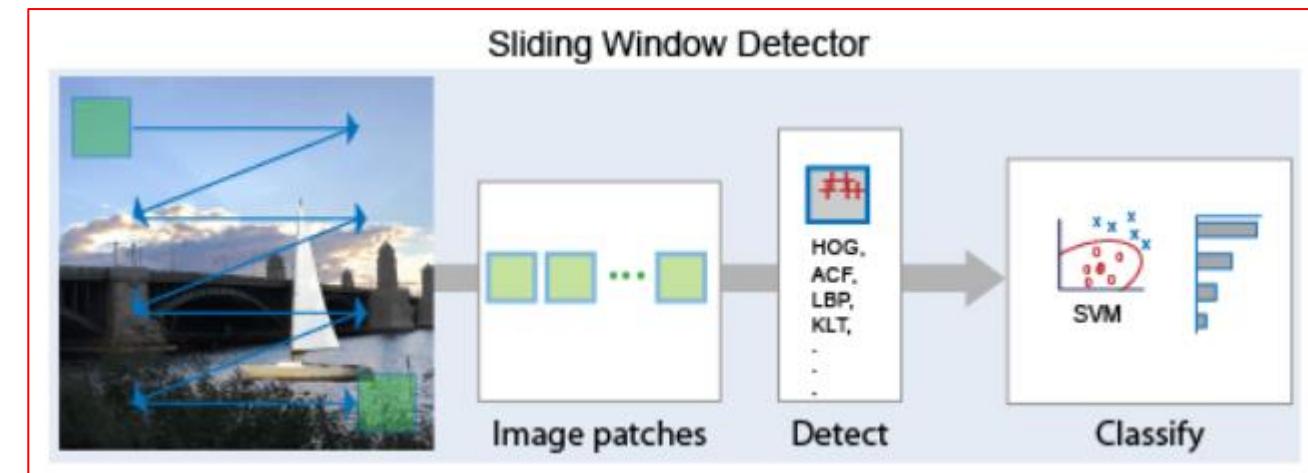
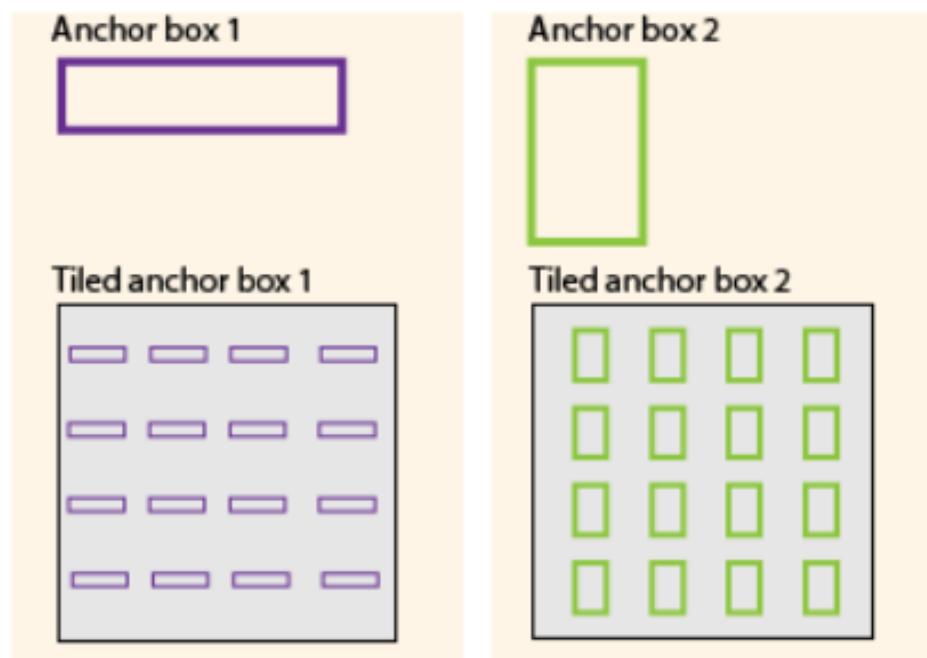
Final detection

YOLOv3	YOLO
YOLOv3 (Better, not Faster, Stronger) Since YOLOv3 has more layers	
Darknet-53 network architecture	Darknet-19 network architecture
Solve the issue of small object, with three scales, and concatenate the up sample with previous layer	Struggled with small object detections, because of the loss of fine-grained features as the layers down sampled the input
Detection is done by applying 1×1 detection kernels on feature maps of three different sizes at three different places in the network	Fully convolutional network and its eventual output is generated by applying a 1×1 kernel on a feature map
YOLOv3 predicts bounding boxes at 3 different scales	predicts 10x less the number of bounding boxes predicted by YOLOv3

What Is an Anchor Box?

Anchor boxes are a set of predefined bounding boxes of a certain height and width. These boxes are defined to capture the scale and aspect ratio of specific object classes you want to detect and are typically chosen based on object sizes in your training datasets. During detection, the predefined anchor boxes are tiled across the image. The network predicts the probability and other attributes, such as background, intersection over union (IoU) and offsets for every tiled anchor box. The predictions are used to refine each individual anchor box. You can define several anchor boxes, each for a different object size. Anchor boxes are fixed initial boundary box guesses.

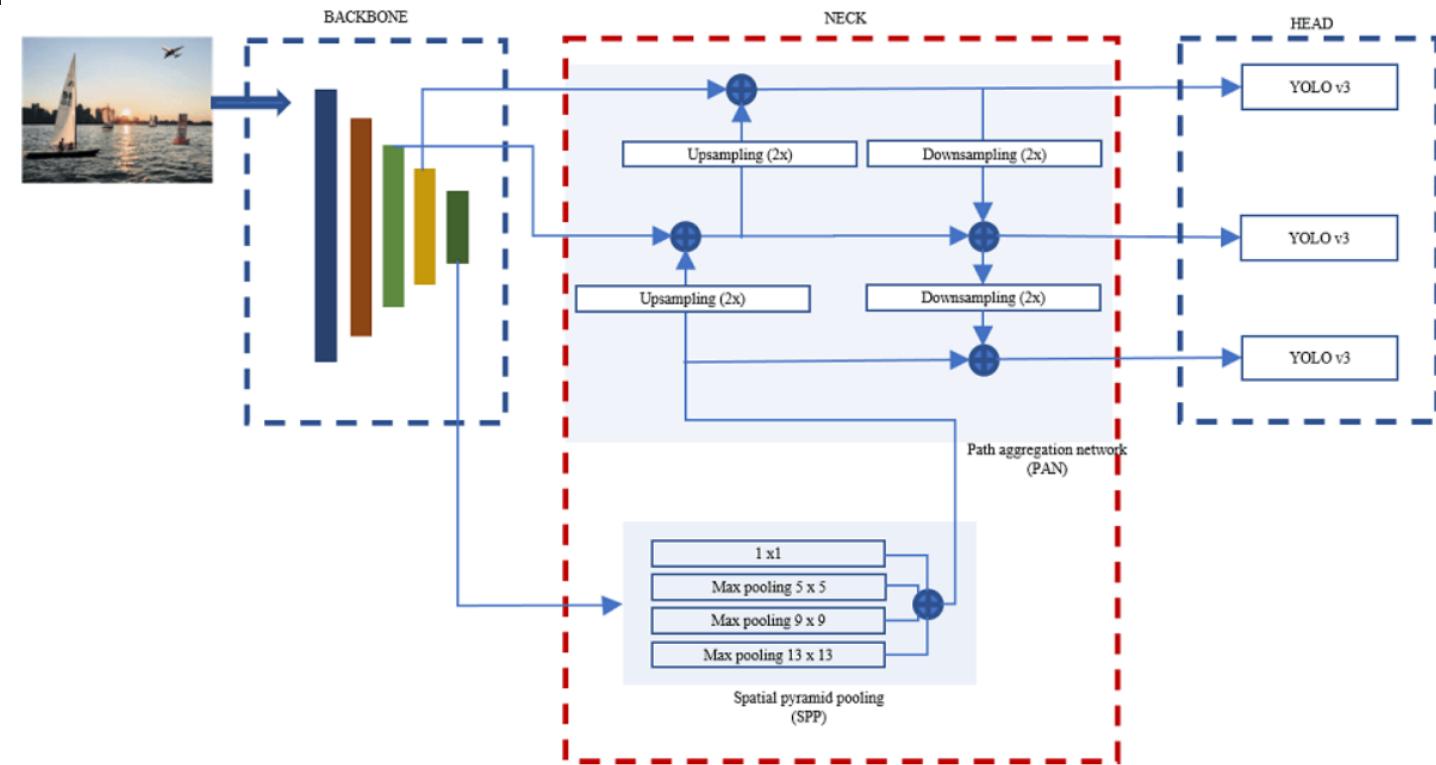
The network does not directly predict bounding boxes, but rather predicts the probabilities and refinements that correspond to the tiled anchor boxes. The network returns a unique set of predictions for every anchor box defined. The final feature map represents object detections for each class. The use of anchor boxes enables a network to detect multiple objects, objects of different scales, and overlapping objects.



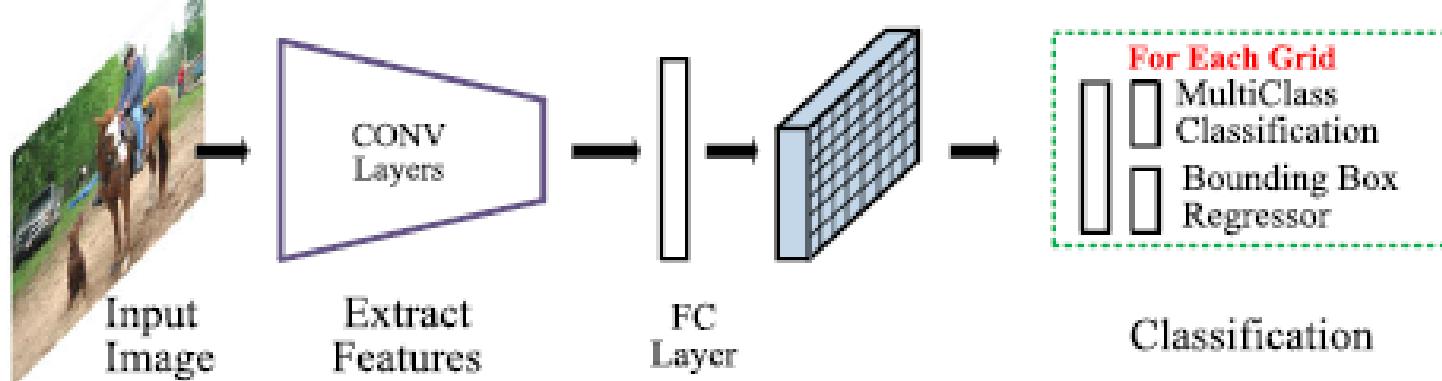
YOLO v4

Composed of three parts

- The **backbone** can be a pretrained convolutional neural network such as VGG16 or CSPDarkNet53 trained on COCO or ImageNet data sets. The backbone of the YOLO v4 network acts as the **feature extraction network** that computes feature maps from the input images.
- The **neck** connects the backbone and the head. It is composed of a spatial pyramid pooling (SPP) module and a path aggregation network (PAN). The neck **concatenates the feature maps from different layers** of the backbone network and sends them as inputs to the head.
- The **head** processes the aggregated features and **predicts the bounding boxes, objectness scores, and classification scores**. The YOLO v4 network uses one-stage object detectors, such as YOLO v3, as detection heads.



YOLO



SSD

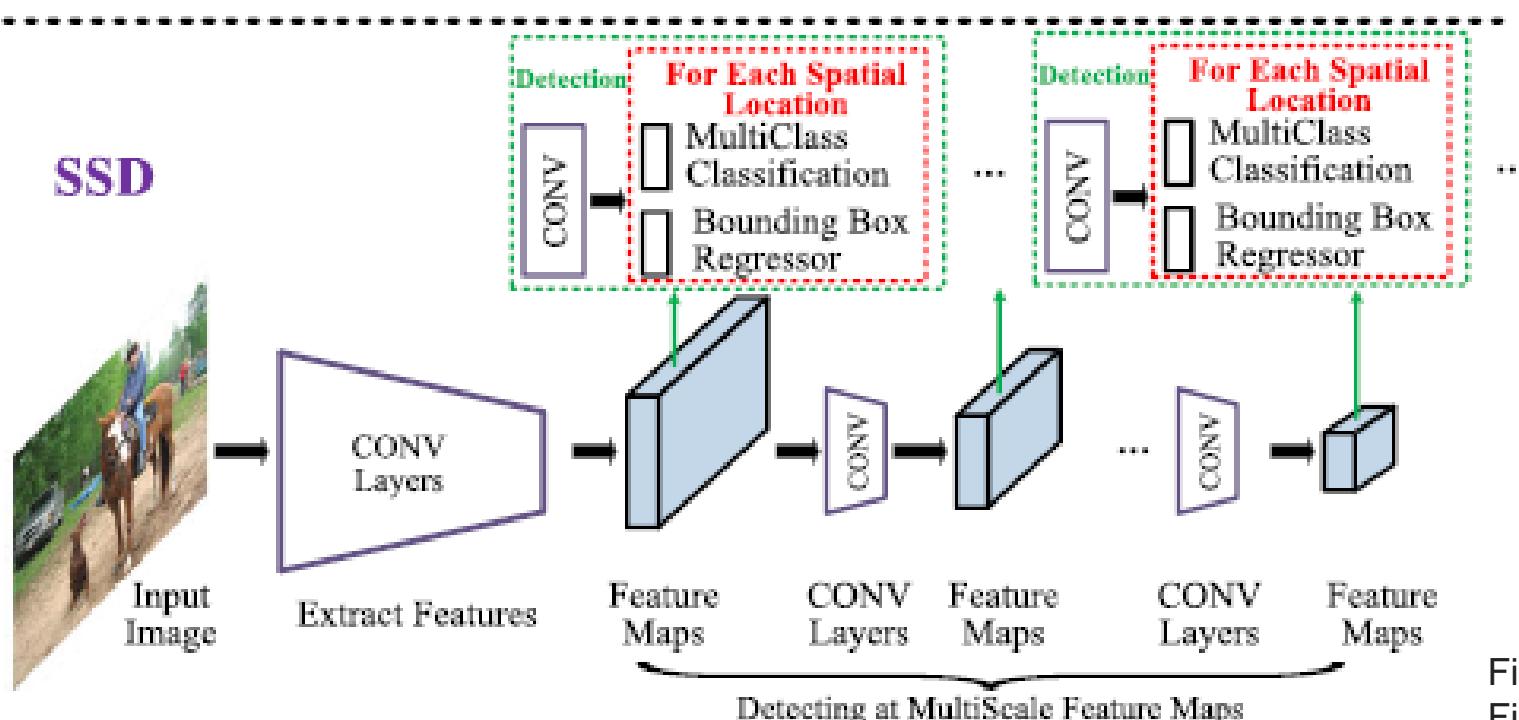
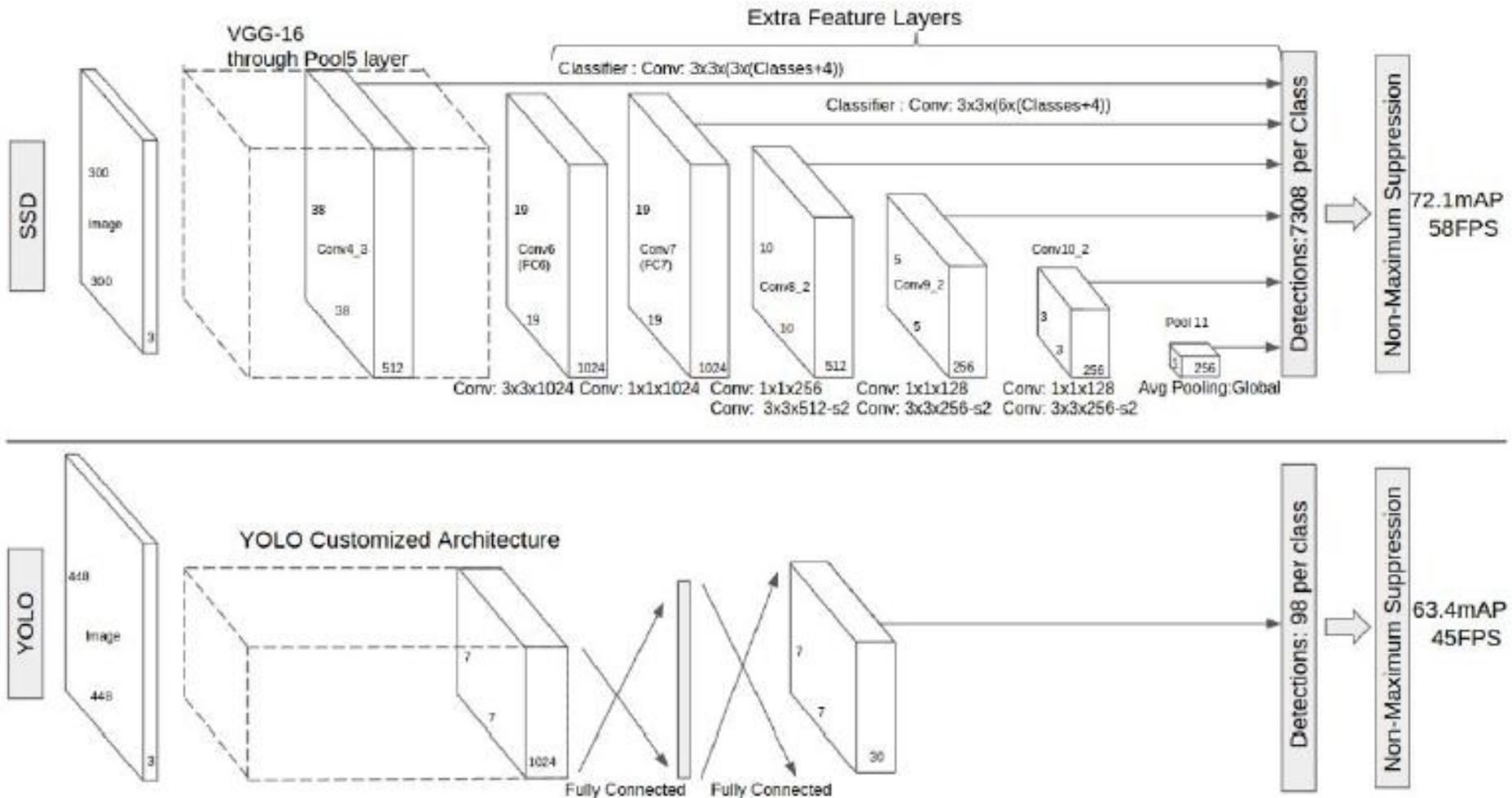


Figure credit: Liu, Li, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. "Deep learning for generic object detection: A survey." *International journal of computer vision* 128 (2020): 261-318.

The Single Shot Detector (SSD)



- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37

The Single Shot Detector (SSD)

- SSD In order to preserve real-time speed without sacrificing too much detection accuracy, Liu et al. (2016) proposed SSD (Single Shot Detector),
 - faster than YOLO (Redmon et al. 2016) and
 - with an accuracy competitive with region based detectors such as Faster RCNN (Ren et al. 2015).
- To achieve fast detection speed, while still retaining high detection quality, SSD effectively combines ideas from
 - RPN in Faster RCNN (Ren et al. 2015),
 - YOLO (Redmon et al. 2016) and
 - multiscale CONV features (Hariharan et al. 2016) Like YOLO, SSD predicts a fixed number of bounding boxes and scores, followed by an NMS step to produce the final detection. The CNN network in SSD is fully convolutional, whose early layers are based on a standard architecture, such as VGG (Simonyan and Zisserman 2015), followed by several auxiliary CONV layers, progressively decreasing in size.
- The information in the last layer may be too coarse spatially to allow precise localization, so SSD performs detection over multiple scales by operating on multiple CONV feature maps, each of which predicts category scores and box offsets for bounding boxes of appropriate sizes.

Single Shot MultiBox Detector (SSD)

Contributions:

- ▷ A single-shot detector for multiple categories that is faster than state of the art single shot detectors (YOLO) and as accurate as Faster R-CNN
- ▷ Predicts category scores and boxes offset for a fixed set of default BBs **using small convolutional filters applied to feature maps**
- ▷ Predictions of different scales from feature maps of different scales, and separate predictions by aspect ratio
- ▷ End-to-end training and high accuracy, **improving speed vs accuracy trade-off**

- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37
- Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

Single Shot MultiBox Detector (SSD)

- Second one-stage detector in deep learning era.
- **SSD**: First deep network based object detector that does **not resample pixels or features** for bounding box hypotheses and is **as accurate as approaches that do**.
- The main contribution: introduction of the multi-reference and multi-resolution detection techniques
- Significantly improves the detection accuracy of a one-stage detector, especially for some small objects.
- SSD has advantages in terms of both detection speed and accuracy (VOC07 mAP=76.8%, VOC12 mAP=74.9%, COCO mAP@.5=46.5%, mAP@[.5,.95]=26.8%, a fast version runs at 59fps).
- The main difference between SSD and any previous detectors:
 - SSD detects objects of different scales on different layers of the network,
 - Previous detectors only run detection on their top layers.
- W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37

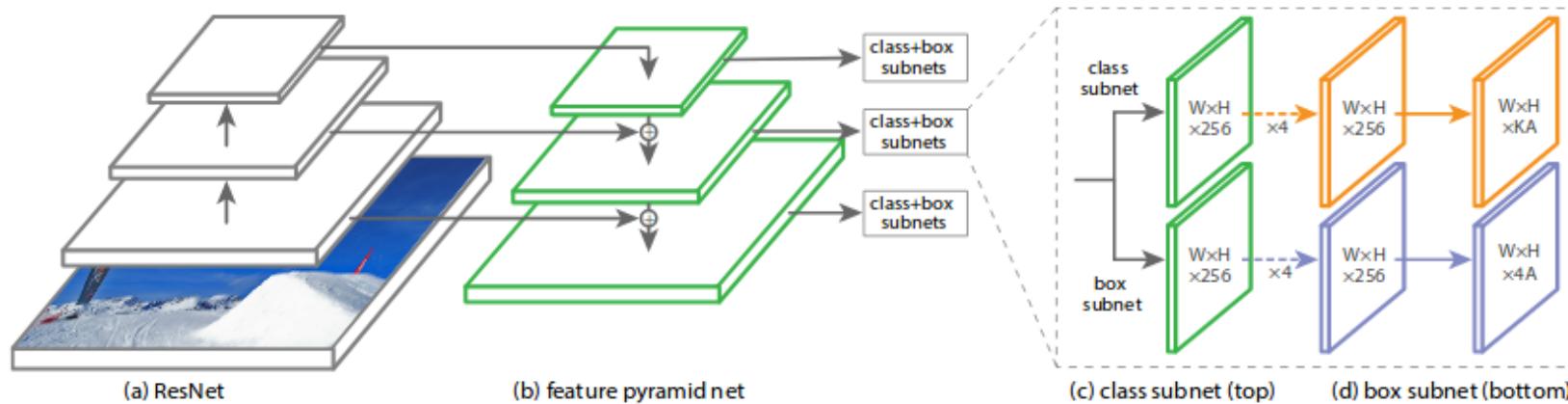
RetinaNet

- Despite of its high speed and simplicity, the one-stage detectors have trailed the accuracy of two-stage detectors for years.
- T.-Y. Lin et al. have discovered the reasons behind and proposed RetinaNet in 2017.
- They claimed that the extreme foreground-background class imbalance encountered during training of dense detectors is the central cause.
- To this end, a new loss function named “focal loss” has been introduced in RetinaNet by reshaping the standard cross entropy loss so that detector will put more focus on hard, misclassified examples during training.
- Focal Loss enables the one-stage detectors to achieve comparable accuracy of two-stage detectors while maintaining very high detection speed. (COCO mAP@.5=59.1%, mAP@[.5, .95]=39.1%).

T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” IEEE PAMI, 2018.

Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

RetinaNet



- **Backbone:** Feature Pyramid Network (FPN) on top of a feedforward ResNet architecture [16] to generate a rich, multi-scale convolutional feature pyramid.
- **Two subnetworks:**
 - Subnet-1: for classifying anchor boxes
 - Subnet-2: for regressing from anchor boxes to ground-truth object boxes

RetinaNet

Schultheiss, Manuel, Sebastian A. Schober, Marie Lodde, Jannis Bodden, Juliane Aichele, Christina Müller-Leisse, Bernhard Renger, Franz Pfeiffer, and Daniela Pfeiffer. "A robust convolutional neural network for lung nodule detection in the presence of foreign bodies." *Scientific Reports* 10, no. 1 (2020): 1-9.

Chest radiographies with nodules detected by RetinaNet.

Green rectangles: ground-truth is marked

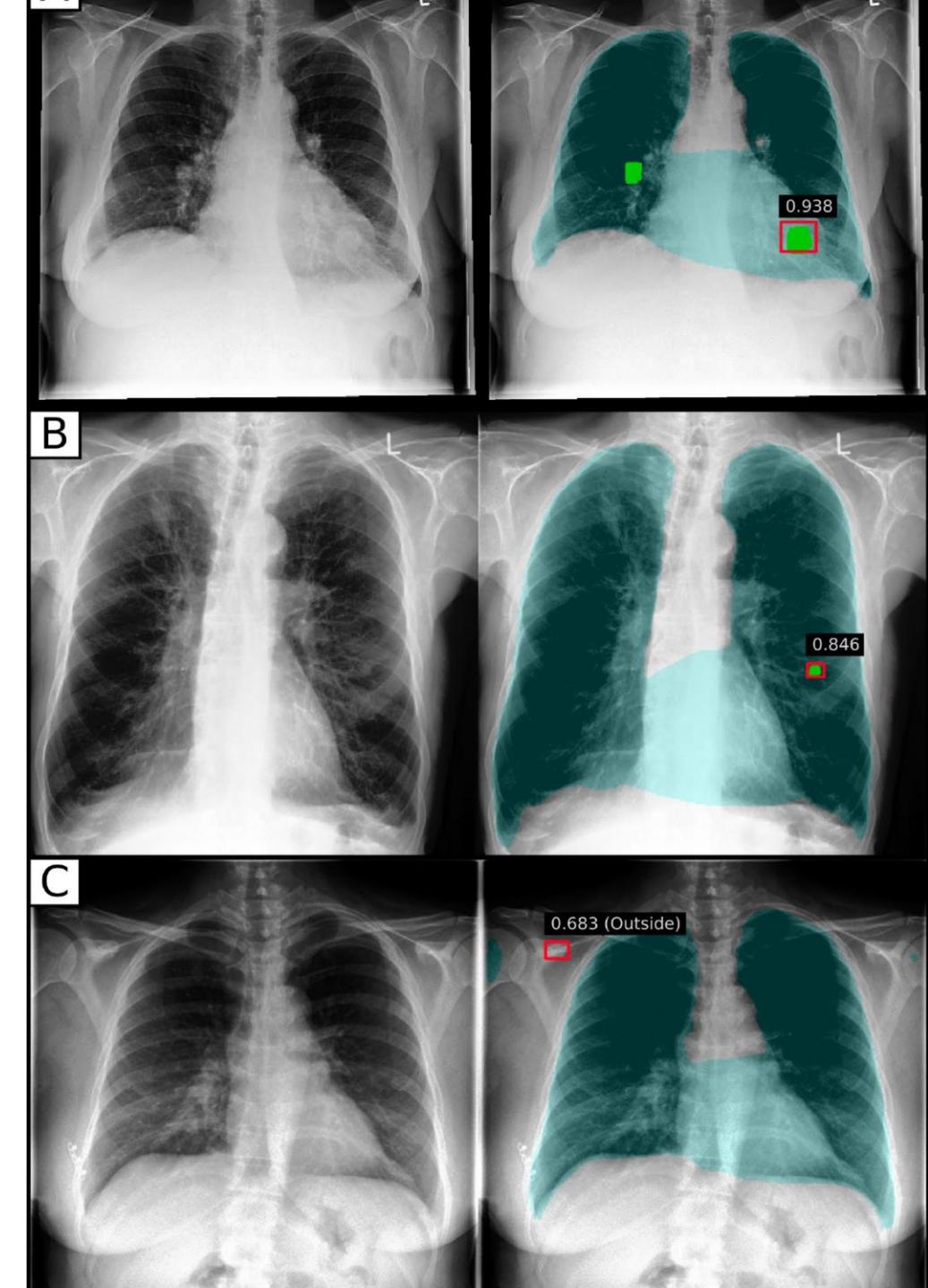
Red: predictions

Cyan: Predicted lung segmentation masks

(A) True-positive prediction (0.938) marked with a red rectangle by the CNN and an undetected, false-negative nodule in the left lung lobe.

(B) True-positive prediction within the right lung lobe.

(C) False-positive prediction outside of the chest.



Key Ideas behind RetinaNet

Two-stage detectors:

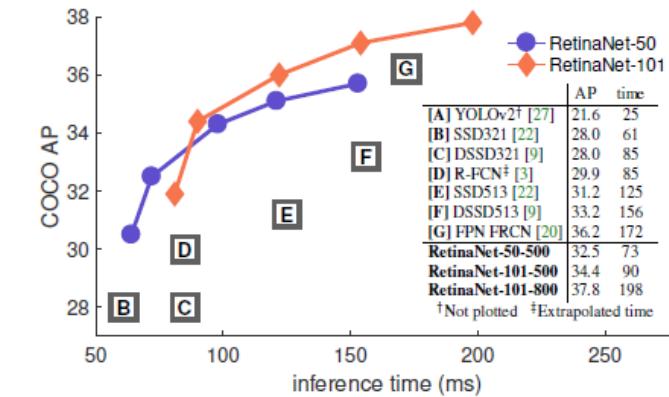
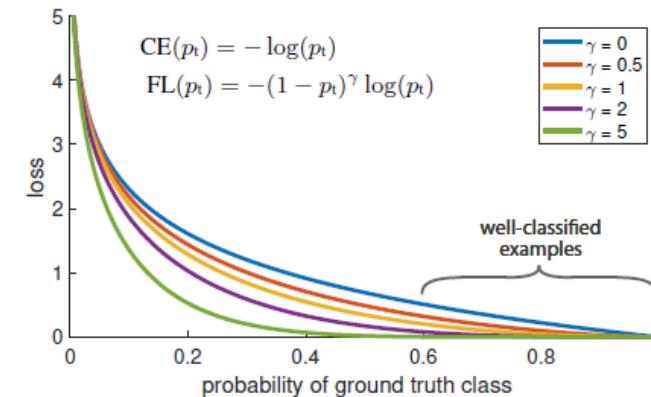
1. Set of sparse candidates are determined,
2. A classifier is applied on these candidate locations

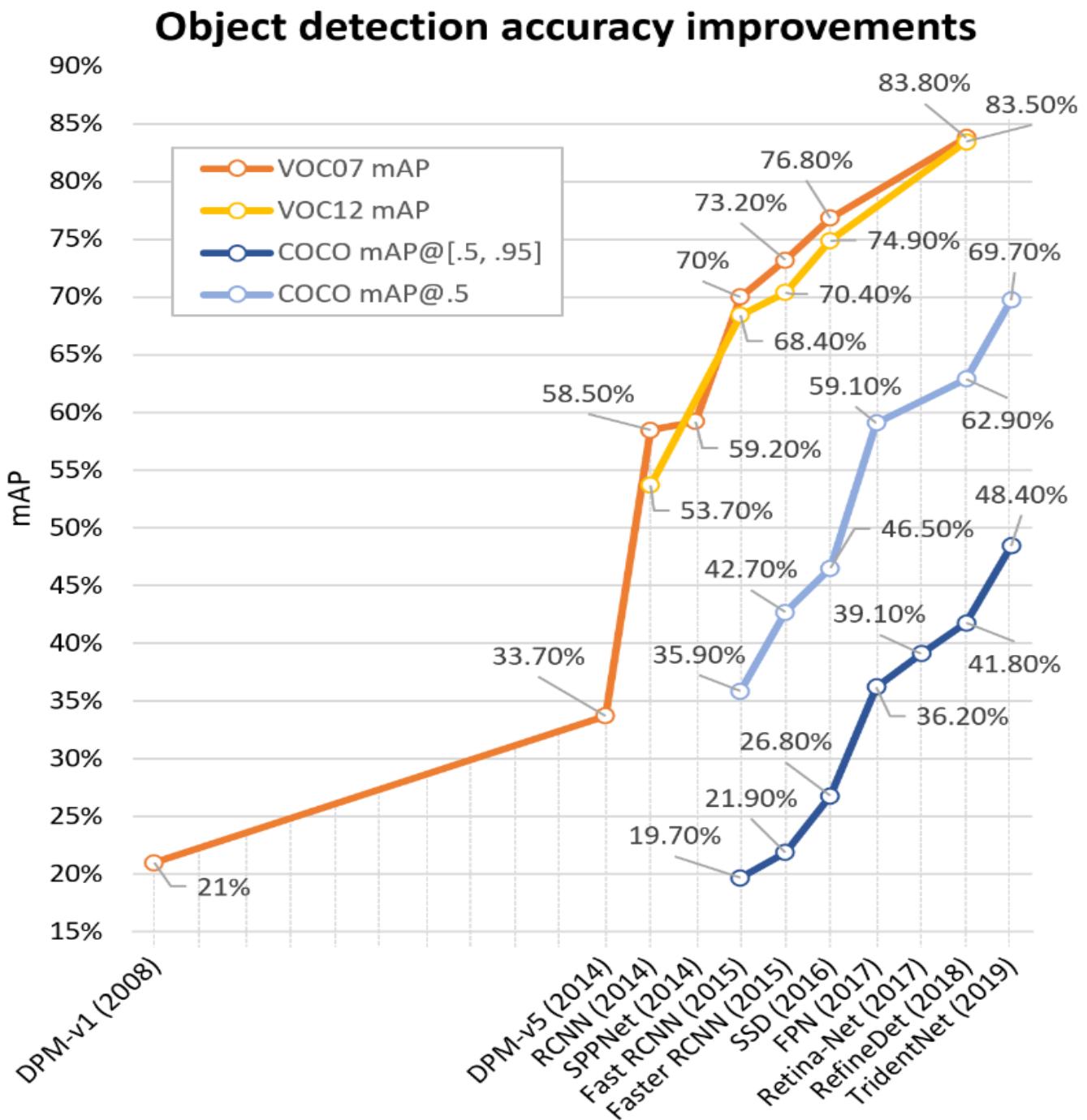
One-stage detectors:

- Applied over a regular, dense sampling of possible object locations
- Faster and simpler
- But lower accuracy compared to two-stage detectors.

RetinaNet:

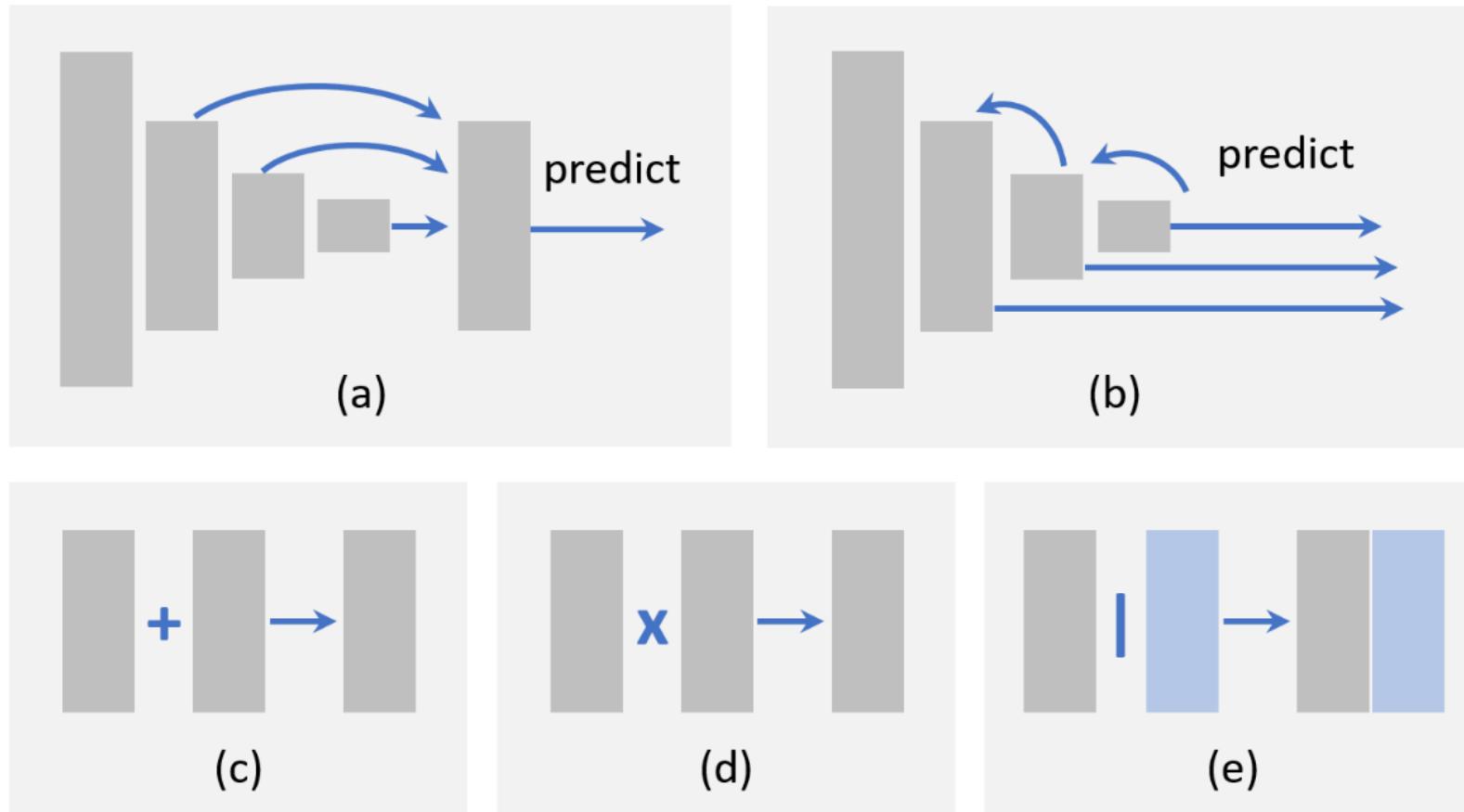
- Extreme foreground-background class imbalance encountered during training of dense detectors is the central cause for low performance.
- Proposed solution: reshape the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples.
- Focal Loss : focuses training on a sparse set of hard examples and prevents the vast number of easy negatives from overwhelming the detector during training.





Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

Feature Fusion

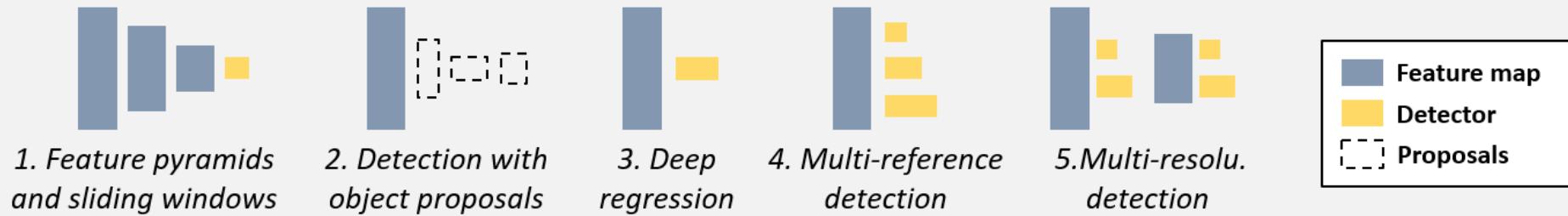


feature fusion methods:

- (a) Bottom-up fusion,
- (b) top-down fusion,
- (c) element-wise sum,
- (d) element-wise product, and
- (e) concatenation.

Zou, Zhengxia, Zhenwei Shi, Yuhong Guo, and Jieping Ye. "Object Detection in 20 Years: A Survey." *arXiv preprint arXiv:1905.05055* (2019).

Evolution of Multi-scale Detection



Year: 2001 2006 2008 2013 2014 2015 2016 2017 2018 2019

Feature Pyramids and Sliding Windows

@VJ Det. (P. Viola et al-CVPR2001), @HOG Det. (N. Dalal et al-CVPR2005), @DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010), @ Exemplar SVM (T. Malisiewicz et al-ICCV2011), @ Overfeat (P. Sermanet et al-ICLR2014) ...

Detection with Object Proposals

Deep Regression

@DNN Det. (C. Szegedy et al-NIPS2013), @YOLO (J. Redmon et al-CVPR2016) ...

@RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014) @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015) ...

Deep Regression

Multi-reference Detection

Faster-RCNN (S. Ren et al-NIPS2015), @SSD (W. Liu et al-ECCV2016), @YOLOv2 (J. Redmon et al-CVPR2017), @TridentNet (Y. Li et al-arXiv19) ...

Multi-resolution Detection

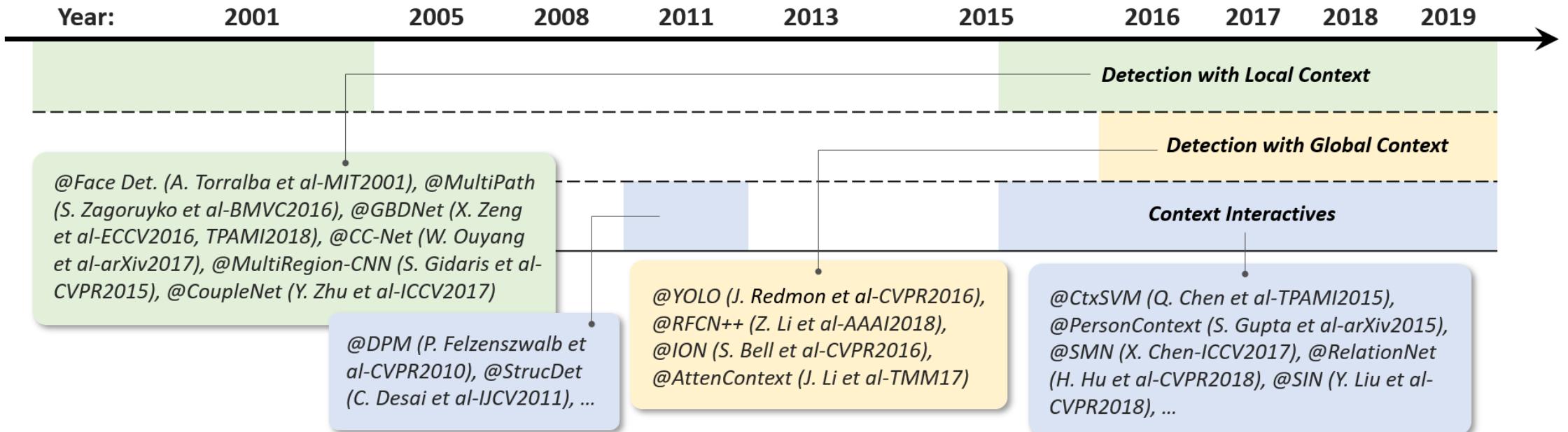
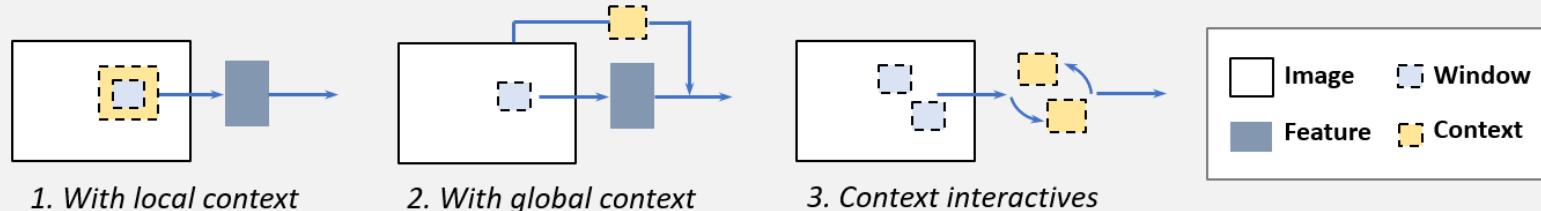
@SSD (W. Liu et al-ECCV2016), @Unified Det. (Z. Cai et al-ECCV2016) @FPN (T. Y. Lin et al-CVPR2017), @RetinaNet(T. Y. Lin et al-ICCV2017), @RefineDet (Zhang et al-CVPR18) ...

Evolution of Bounding Box Regression

weak ← *Invariance of translation and scale* → strong

Year:	2001	2006	2008	2013	2014	2015	2016	2017	2018	2019
Method	<i>Without Bounding Box Regression</i>	<i>From Bounding Box to Bounding Box</i>				<i>From Feature to Bounding Box</i>				
Remarks			Icing on the cake, optional				Essential, integrated with the model			
	@VJ Det. (P. Viola et al-CVPR2001), @HOG Det. (N. Dalal et al-CVPR2005), @ Exemplar SVM (T. Malisiewicz et al-ICCV2011) ...	@DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010)		@Overfeat (P. Sermanet et al-ICLR2014), @RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014) @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015), @YOLO (J. Redmon et al-CVPR2016), @SSD (W. Liu et al-ECCV2016), @YOLOv2 (J. Redmon et al-CVPR2017), @Unified Det. (Z. Cai et al-ECCV2016) @FPN (T. Y. Lin et al-CVPR2017), @RetinaNet(T. Y. Lin et al-ICCV2017), @RefineDet (Zhang et al-CVPR18), @TridentNet (Y. Li et al-arXiv19) ...						

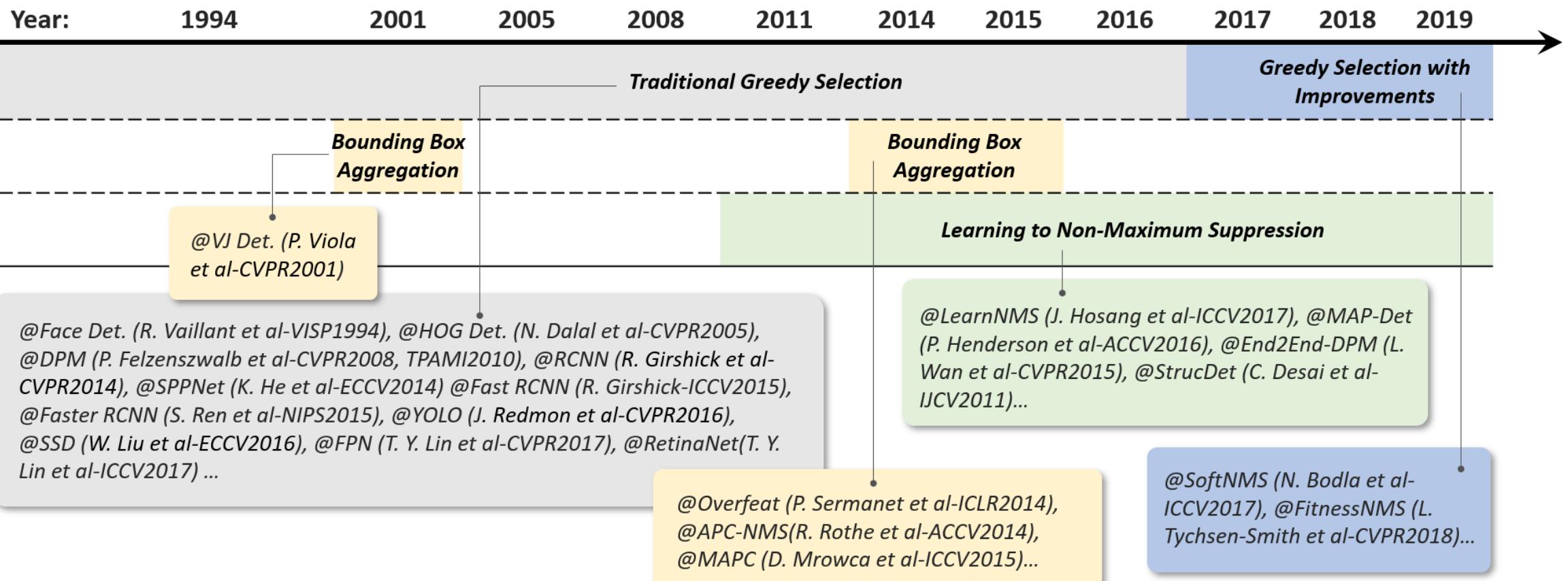
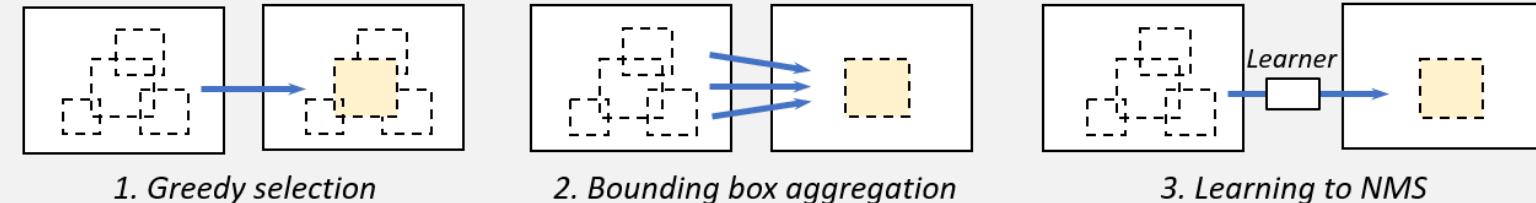
Evolution of Context Priming in Object Detection



Visual objects are usually embedded in a typical context with the **surrounding environments**. Our brain takes advantage of the associations among objects and environments to facilitate visual perception and cognition [160]. Context priming has long been used to improve detection. There are three common approaches in its evolutionary history:

1. detection with local context,
2. detection with global context,
3. context interactives.

Evolution of Non-Max Suppression

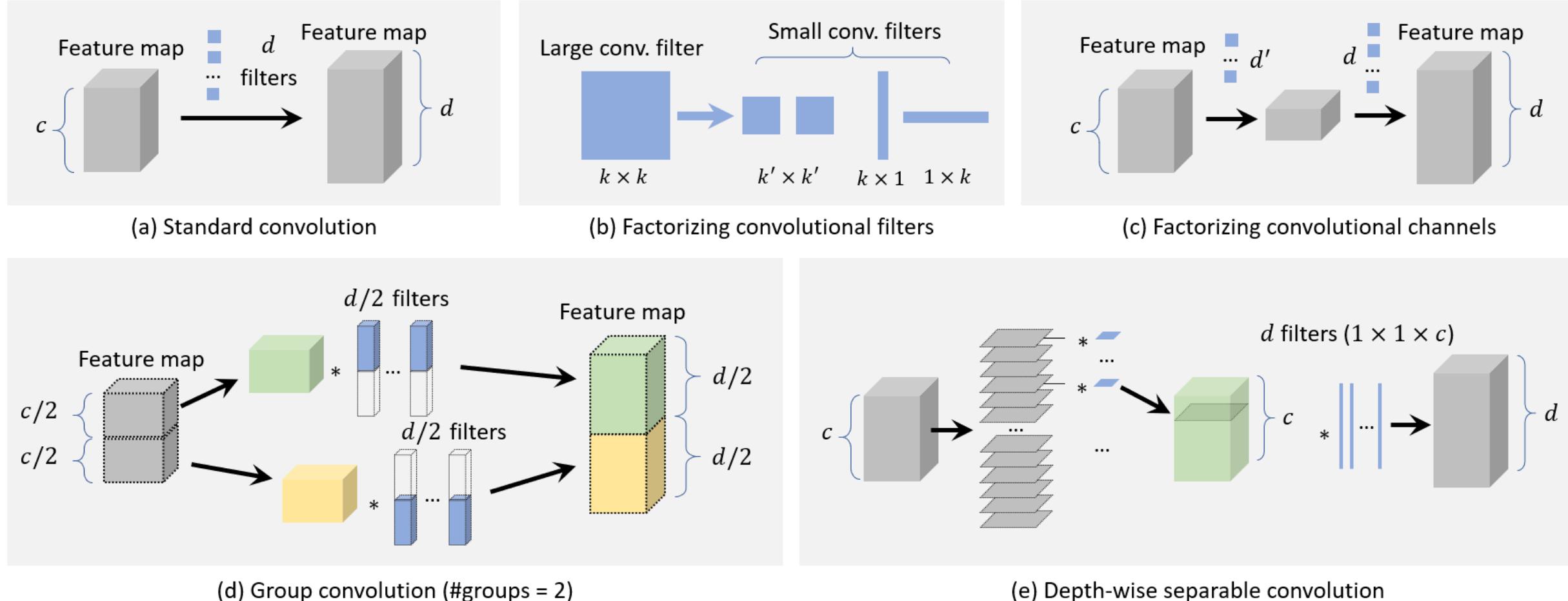


Evolution of Hard Negative Mining

Year:	1994	2001	2005	2008	2014	2015	2016	2017	2018	2019	
Method	Bootstrap				Without Hard Negative Mining		Bootstrap + New Loss Functions				
Remarks	Bootstrap was widely used to deal with the insufficient computing resources of early time				By simply balancing the weights between object and background classes		Focusing on hard examples. Computing power is no longer a problem.				
<p>@Face Det. (H. A. Rowley et al-CMUTechRep1995), @Haar Det. (C. P. Papageorgiou et al-ICCV1998), @VJ Det. (P. Viola et al-CVPR2001), @HOG Det. (N. Dalal et al-CVPR2005), @DPM (P. Felzenszwalb et al-CVPR2008, TPAMI2010)...</p>				<p>@RCNN (R. Girshick et al-CVPR2014), @SPPNet (K. He et al-ECCV2014) @Fast RCNN (R. Girshick-ICCV2015), @Faster RCNN (S. Ren et al-NIPS2015), @YOLO (J. Redmon et al-CVPR2016)...</p>				<p>@SSD (W. Liu et al-ECCV2016), @FasterPed (L. Zhang et al-ECCV2016), @OHEM (A. Shrivastava et al-CVPR2016), @RetinaNet (T. Y. Lin et al-ICCV2017), @RefineDet (Zhang et al-CVPR18)...</p>			

A hard negative is when you take that falsely detected patch, and explicitly create a negative example out of that patch, and add that negative to your training set. When you retrain your classifier, it should perform better with this extra knowledge, and not make as many false positives.

An overview of speed up methods of a CNN's convolutional layer and the comparison of their computational complexity



Semantic Segmentation using Deep Learning

CS/ECE 8690 Computer Vision

Filiz Bunyak Ersoy

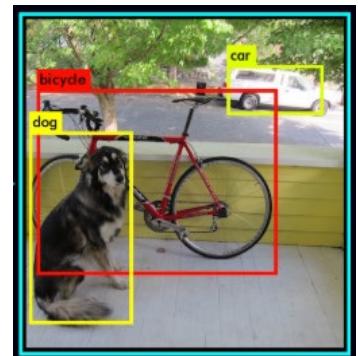
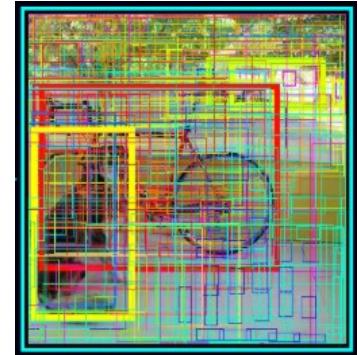
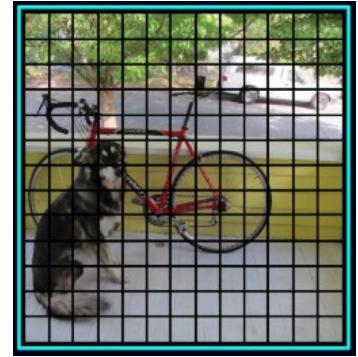
References

Slides generated using following references

1. Garcia-Garcia, Alberto, Sergio Orts-Escalano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. "A survey on deep learning techniques for image and video semantic segmentation." *Applied Soft Computing* 70 (2018): 41-65.
2. Minaee, Shervin, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. "Image segmentation using deep learning: A survey." *IEEE transactions on pattern analysis and machine intelligence* 44, no. 7 (2021): 3523-3542.
3. Fei-Fei Li & Justin Johnson & Serena Yeung,
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf
4. http://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf
5. Justin Johnson --- Deep Learning for Computer Vision
https://web.eecs.umich.edu/~justincj/slides/eecs498/WI2022/598_WI2022_lecture15.pdf

So Far

Single-stage Deep Detectors



Two-stage Deep Detectors

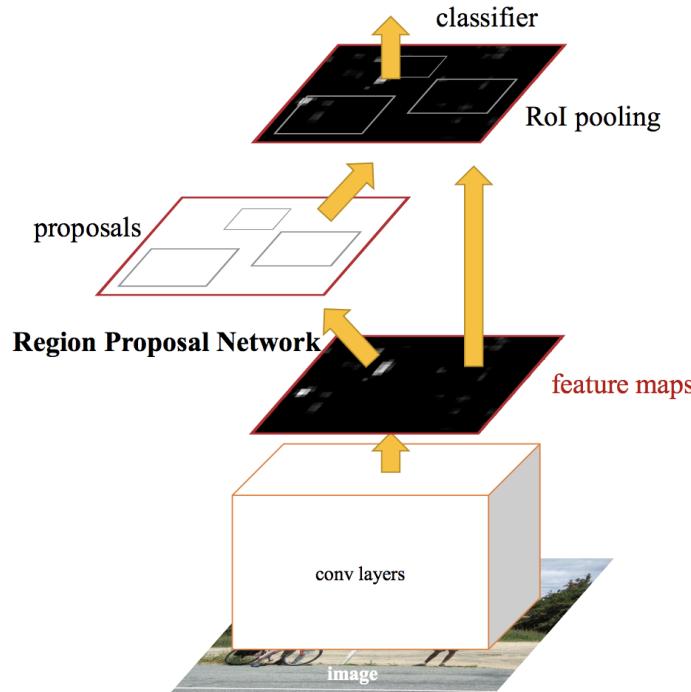
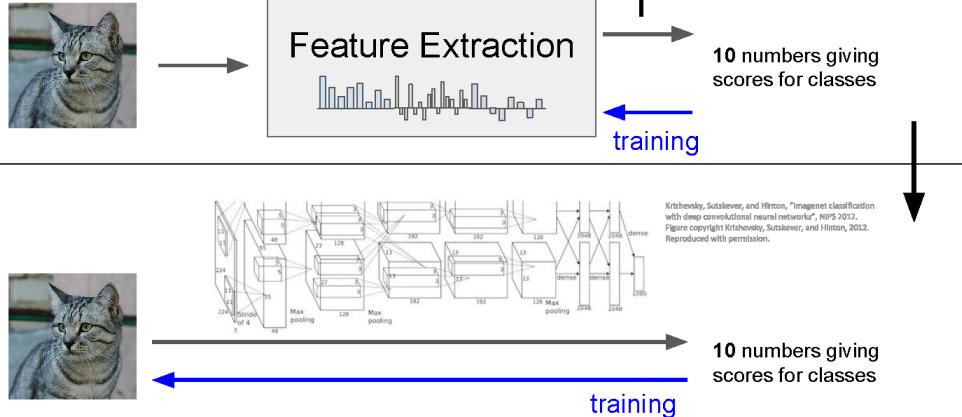


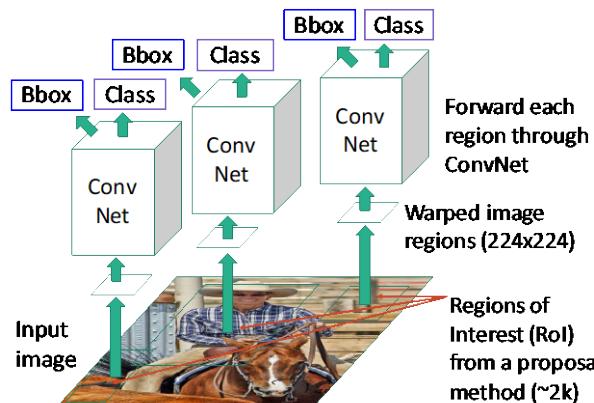
Image Classification

Image features vs ConvNets

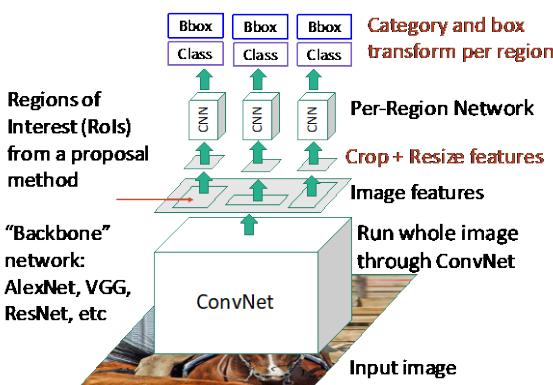


Summary

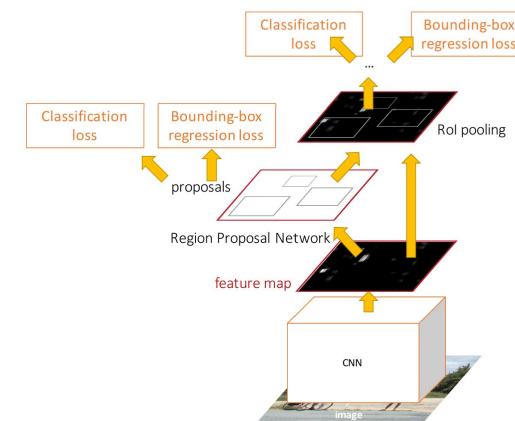
“Slow” R-CNN: Run CNN independently for each region



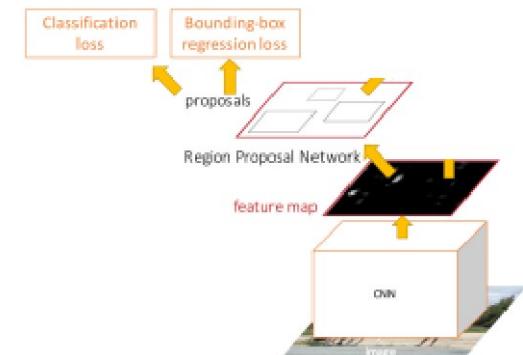
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN

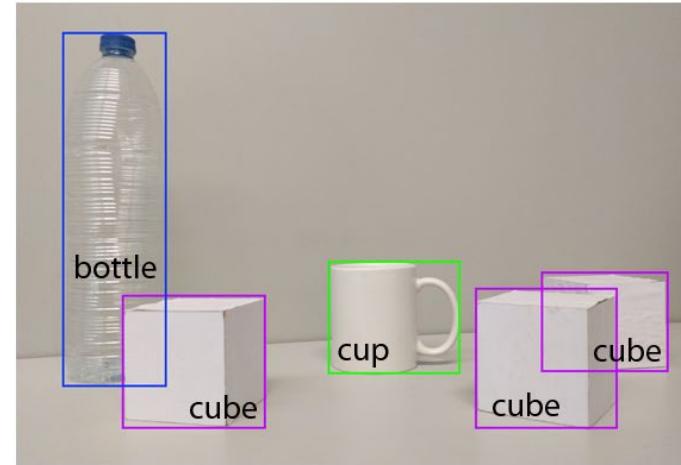


Single-Stage: Fully convolutional detector

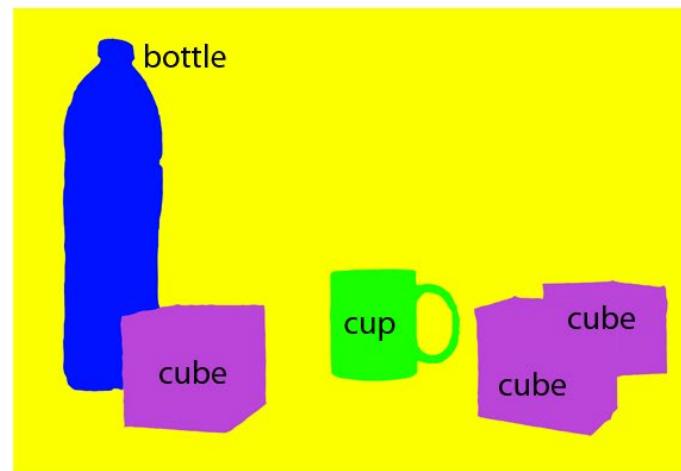




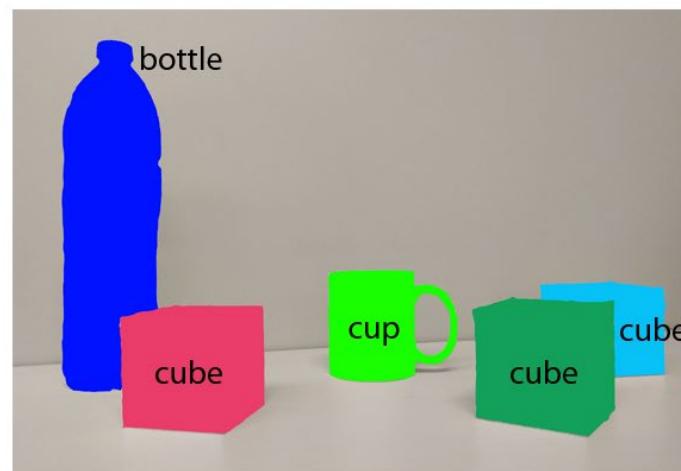
(a) Image classification



(b) Object localization



(c) Semantic segmentation



(d) Instance segmentation

A. Garcia-Garcia, S. Orts-Escalano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. "A review on deep learning techniques applied to semantic segmentation." *arXiv preprint arXiv:1704.06857* (2017).

Semantic Segmentation Datasets

- **Visual Object Classes Challenge 2012**
<http://host.robots.ox.ac.uk/pascal/VOC/voc2012/>
- COCO: Common Objects in Context
- COCO is a large-scale object detection, segmentation, and captioning dataset. <http://cocodataset.org/>

COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints



Popular Datasets

Name and reference	Purpose	Year	Classes	Data	Resolution	Sequence	Synthetic/ real	Samples (training)	Samples (validation)	Samples (test)
PASCAL VOC 2012 Segmentation [30]	Generic	2012	21	2D	Variable	×	R	1464	1449	Private
PASCAL-Context [31]	Generic	2014	540 (59)	2D	Variable	×	R	10,103	N/A	9637
PASCAL-Part [32]	Generic-Part	2014	20	2D	Variable	×	R	10,103	N/A	9637
SBD [33]	Generic	2011	21	2D	Variable	×	R	8498	2857	N/A
Microsoft COCO [34]	Generic	2014	+80	2D	Variable	×	R	82,783	40,504	81,434
SYNTHIA [35]	Urban (Driving)	2016	11	2D	960 × 720	×	S	13,407	N/A	N/A
Cityscapes (fine) [36]	Urban	2015	30 (8)	2D	2048 × 1024	✓	R	2975	500	1525
Cityscapes (coarse) [36]	Urban	2015	30 (8)	2D	2048 × 1024	✓	R	22,973	500	N/A
CamVid [37]	Urban (Driving)	2009	32	2D	960 × 720	✓	R	701	N/A	N/A
CamVid-Sturgess [38]	Urban (Driving)	2009	11	2D	960 × 720	✓	R	367	100	233
KITTI-Layout [39,40]	Urban/Driving	2012	3	2D	Variable	×	R	323	N/A	N/A
KITTI-Ros [41]	Urban/Driving	2015	11	2D	Variable	×	R	170	N/A	46
KITTI-Zhang [42]	Urban/Driving	2015	10	2D/3D	1226 × 370	×	R	140	N/A	112
Stanford background [43]	Outdoor	2009	8	2D	320 × 240	×	R	725	N/A	N/A
SiftFlow [44]	Outdoor	2011	33	2D	256 × 256	×	R	2688	N/A	N/A
Youtube-Objects-Jain [45]	Objects	2014	10	2D	480 × 360	✓	R	10,167	N/A	N/A
Adobe's Portrait Segmentation [29]	Portrait	2016	2	2D	600 × 800	×	R	1500	300	N/A
MINC [46]	Materials	2015	23	2D	Variable	×	R	7061	2500	5000
DAVIS [47,48]	Generic	2016	4	2D	480p	✓	R	4219	2023	2180
NYUDv2 [49]	Indoor	2012	40	2.5D	480 × 640	×	R	795	654	N/A
SUN3D [50]	Indoor	2013	–	2.5D	640 × 480	✓	R	19,640	N/A	N/A
SUNRGBD [51]	Indoor	2015	37	2.5D	Variable	×	R	2666	2619	5050
RGB-D Object Dataset [52]	Household objects	2011	51	2.5D	640 × 480	✓	R	207,920	N/A	N/A
ShapeNet Part [53]	Object/Part	2016	16/50	3D	N/A	×	S	31,963	N/A	N/A
Stanford 2D-3D-S [54]	Indoor	2017	13	2D/2.5D/3D	1080 × 1080	✓	R	70,469	N/A	N/A
3D Mesh [55]	Object/Part	2009	19	3D	N/A	×	S	380	N/A	N/A
Sydney Urban Objects Dataset [56]	Urban (Objects)	2013	26	3D	N/A	×	R	41	N/A	N/A
Large-Scale Point Cloud Classification Benchmark [57]	Urban/Nature	2016	8	3D	N/A	×	R	15	N/A	15

A. Garcia-Garcia, S. Orts-Escalano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez. "A review on deep learning techniques applied to semantic segmentation." *arXiv preprint arXiv:1704.06857* (2017).

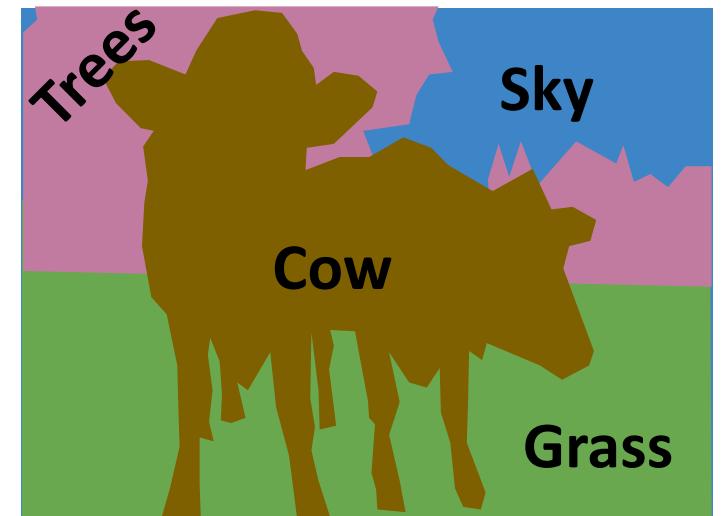
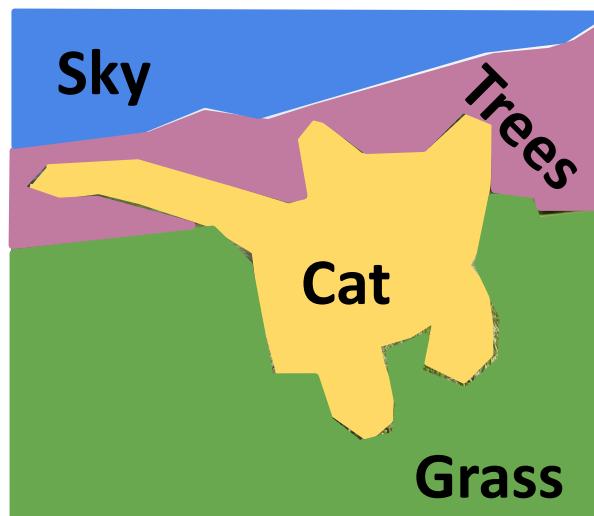
Semantic Segmentation

Label each pixel in the image with a category label

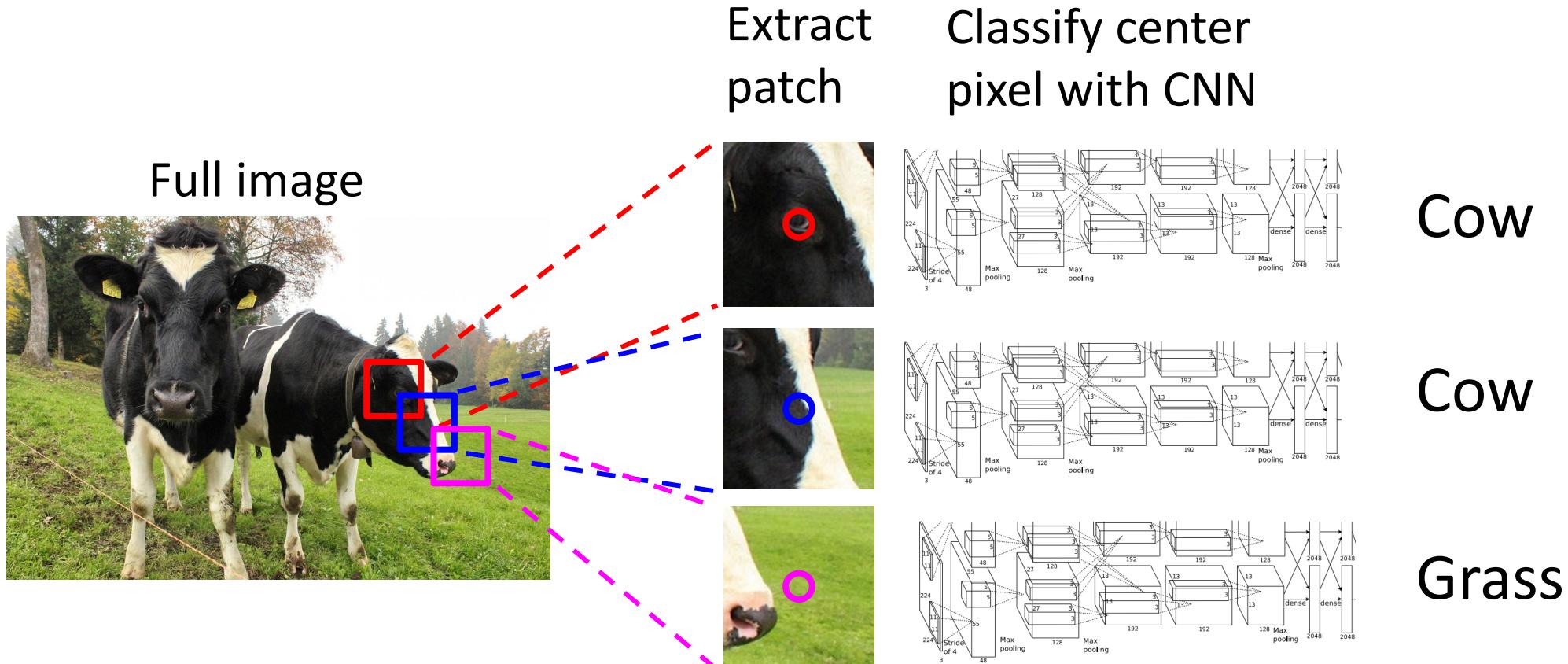
Don't differentiate instances, only care about pixels



This image is CCO public domain

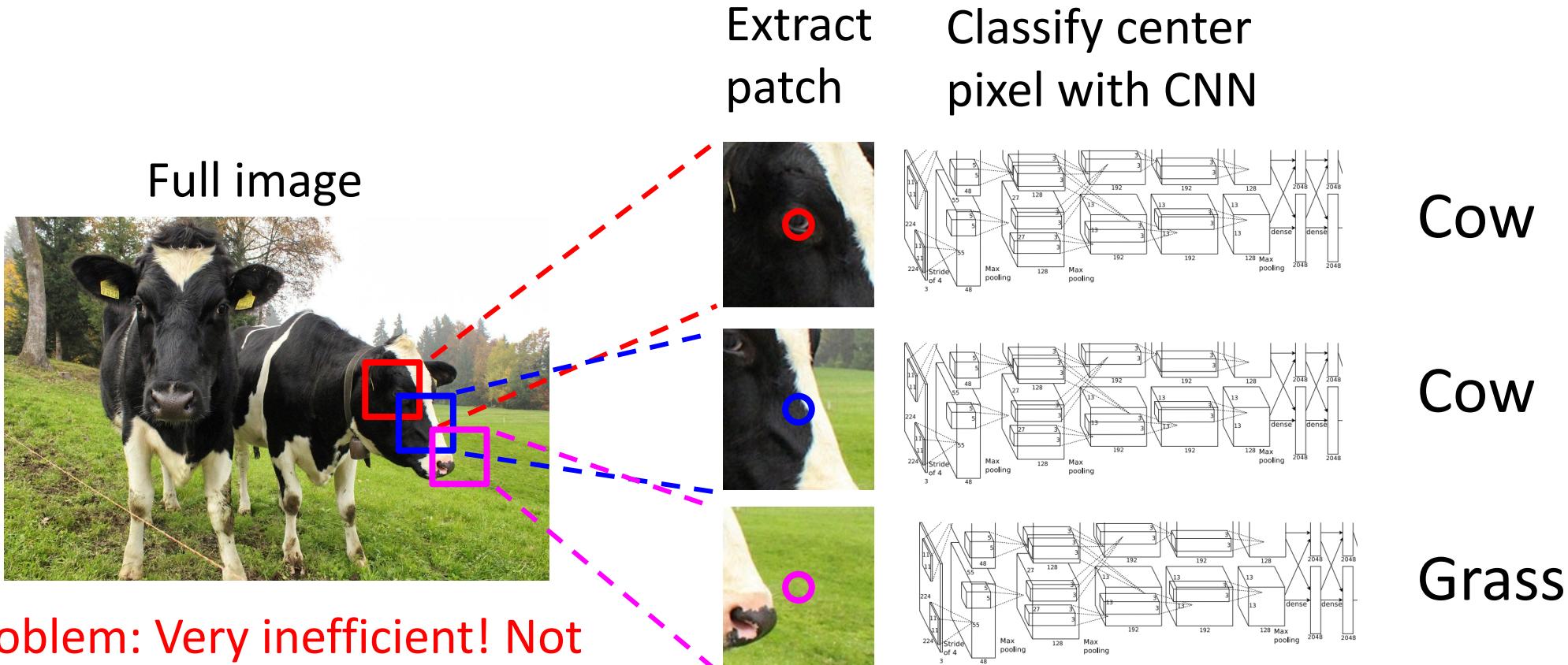


Semantic Segmentation Idea-1: Sliding Window



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

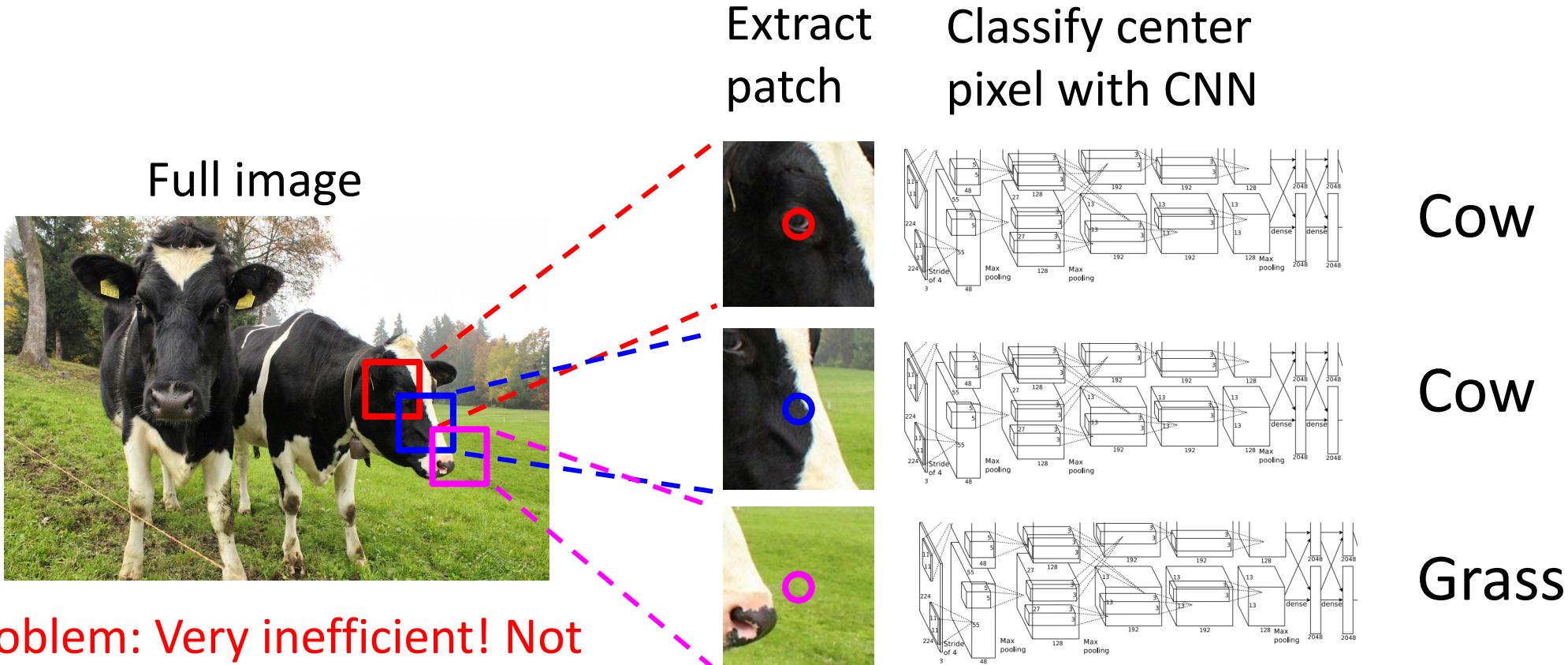
Semantic Segmentation Idea-1: Sliding Window



Problem: Very inefficient! Not reusing shared features between overlapping patches

Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation Idea-1: Sliding Window

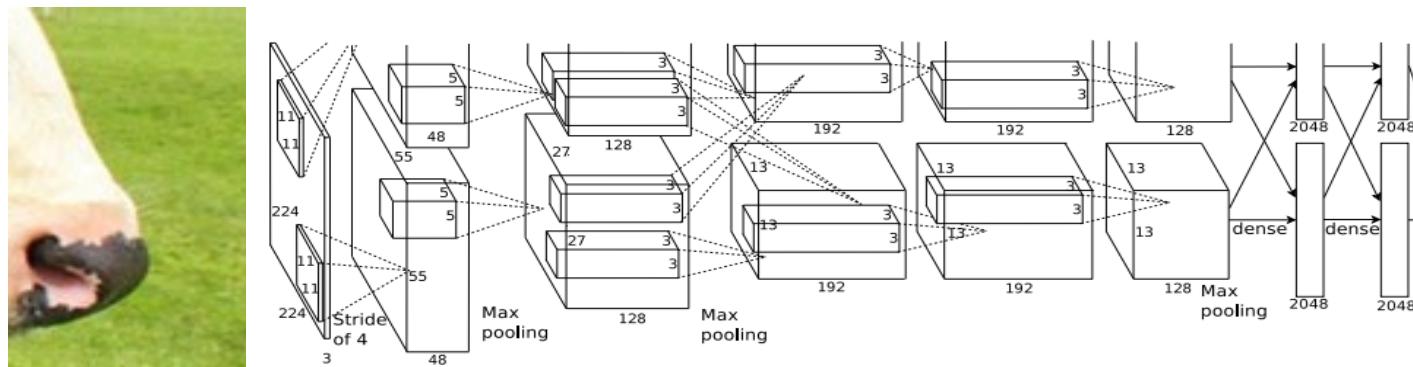


Problem: Very inefficient! Not reusing shared features between overlapping patches

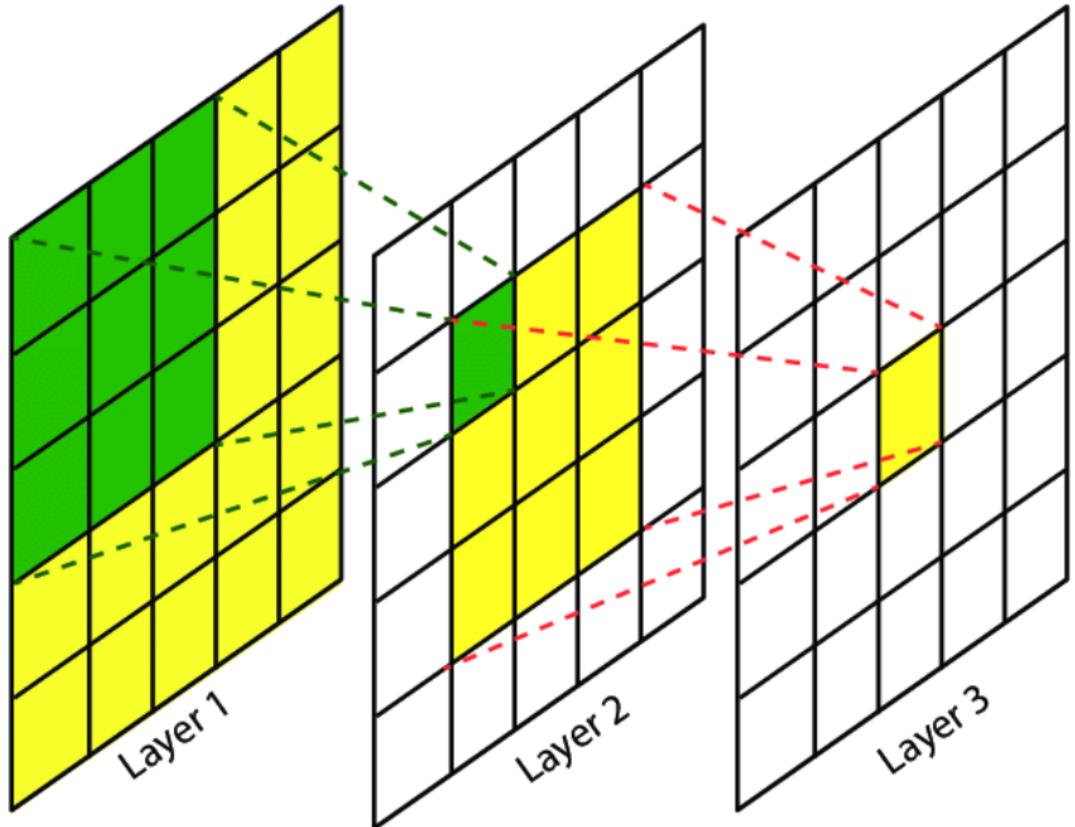
Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Problems of classical CNNs in Segmentation

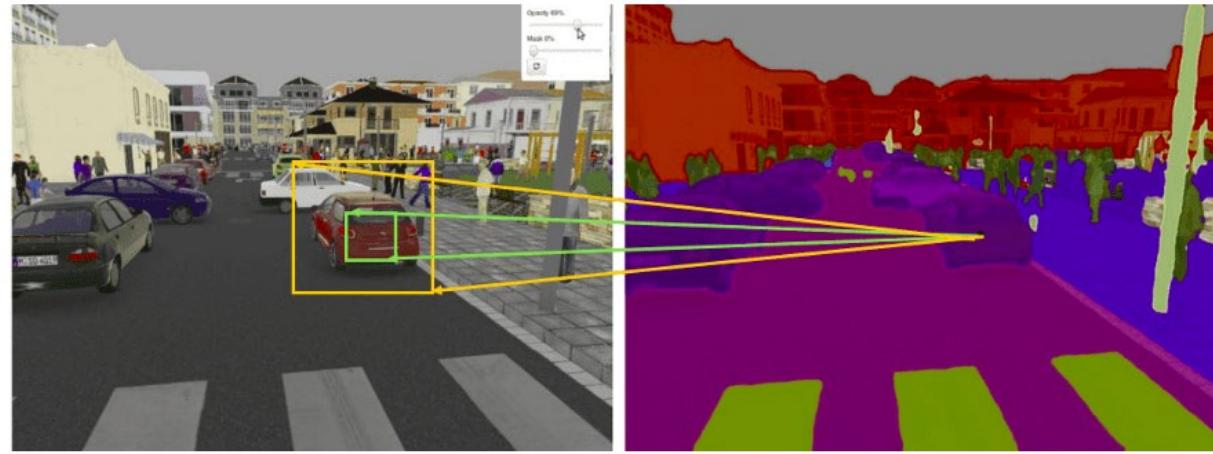
1. Fully connected layers: requires fixed sized patch
2. Pooling layers :
 - increase the field of view and are able to aggregate the context while discarding the ‘where’ information.
 - However, semantic segmentation requires the exact alignment of class maps and thus, needs the ‘where’ information to be preserved.



Receptive Field



The receptive field of each convolution layer with a 3×3 kernel. The green area marks the receptive field of one pixel in Layer 2, and the yellow area marks the receptive field of one pixel in Layer 3.



The green and orange rectangles are two different receptive fields. Which one would you prefer? The green and orange rectangles are two different receptive fields. Which one would you prefer?

Source: [Nvidia's blog](#)

Semantic Segmentation Idea-2: Fully Convolutional Networks

Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." In *Proceedings of the IEEE CVPR*. 2015. 38K citations.

CNN architectures for dense predictions without any fully connected layers.

- Segmentation maps to be generated for image of any size and was also much faster compared to the patch classification approach.
- Almost all the subsequent state of the art approaches on semantic segmentation adopted this paradigm.

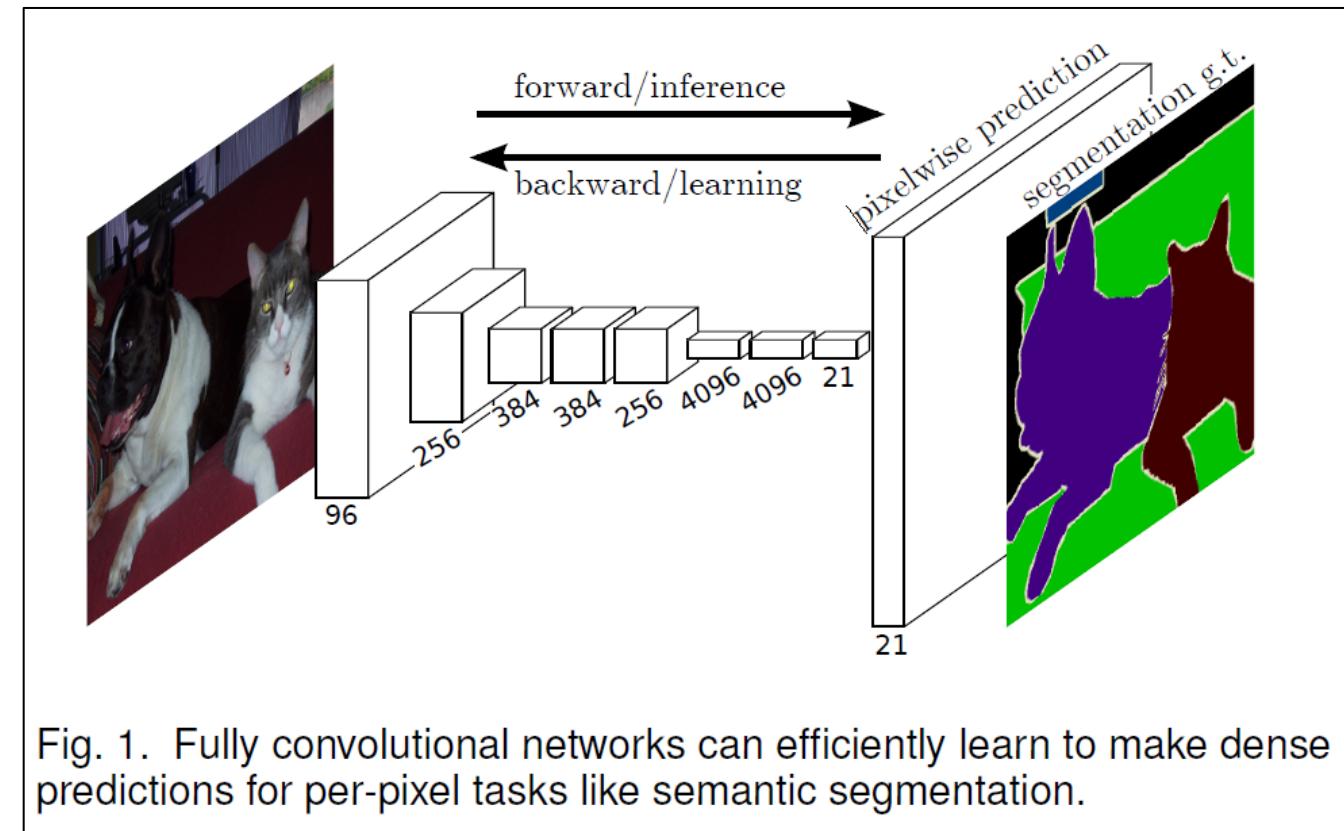
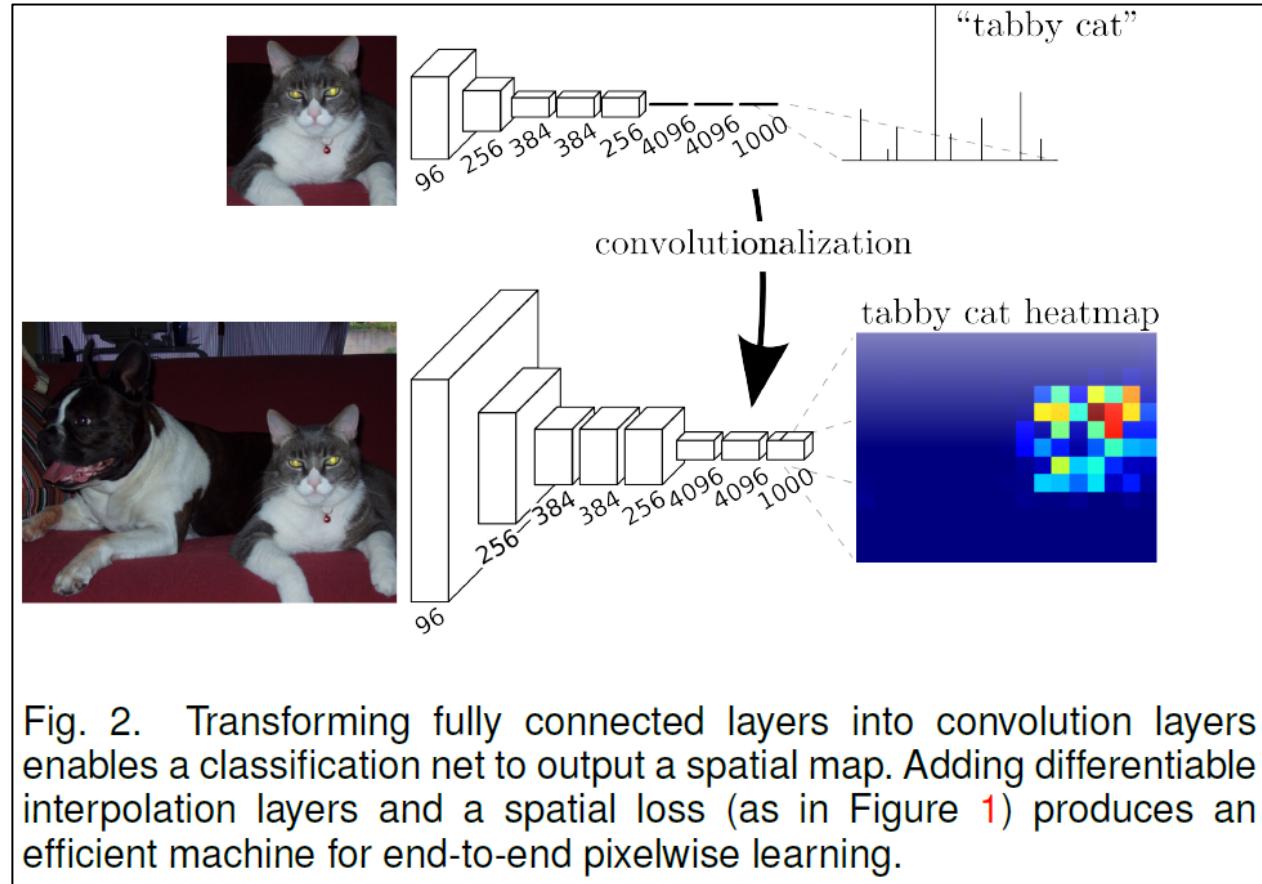


Fig. 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

Semantic Segmentation Idea-2: Fully Convolutional Networks

- Includes only convolutional layers, which enables it to output a segmentation map whose size is the same as that of the input image.
- To handle arbitrarily-sized images, the authors modified existing CNN architectures, such as VGG16 and GoogLeNet, by removing all fully-connected layers such that the model outputs a spatial segmentation map instead of classification scores.



Semantic Segmentation Idea-2: Fully Convolutional Networks

Through the use of skip connections in which feature maps from the final layers of the model are up-sampled and fused with feature maps of earlier layers, the model combines semantic information (from deep, coarse layers) and appearance information (from shallow, fine layers) in order to produce accurate and detailed segmentations

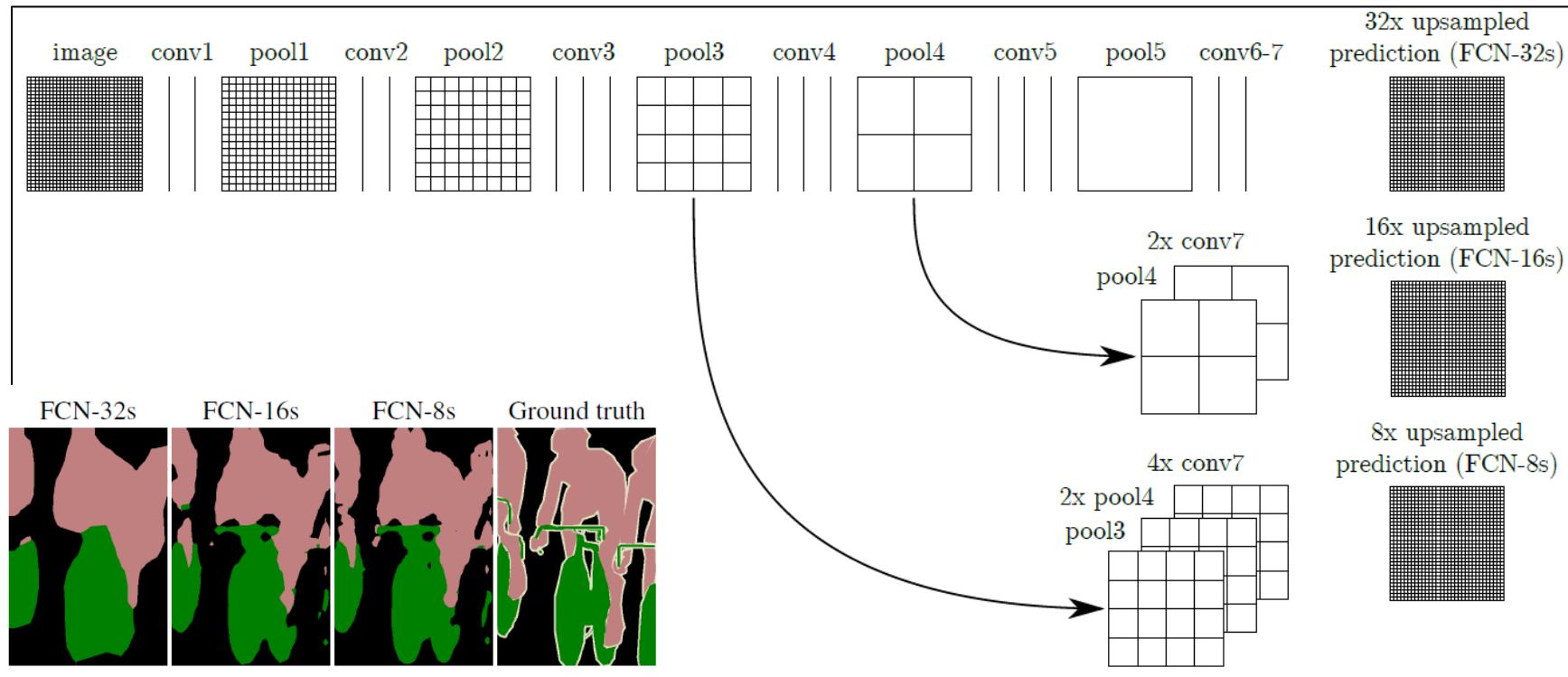


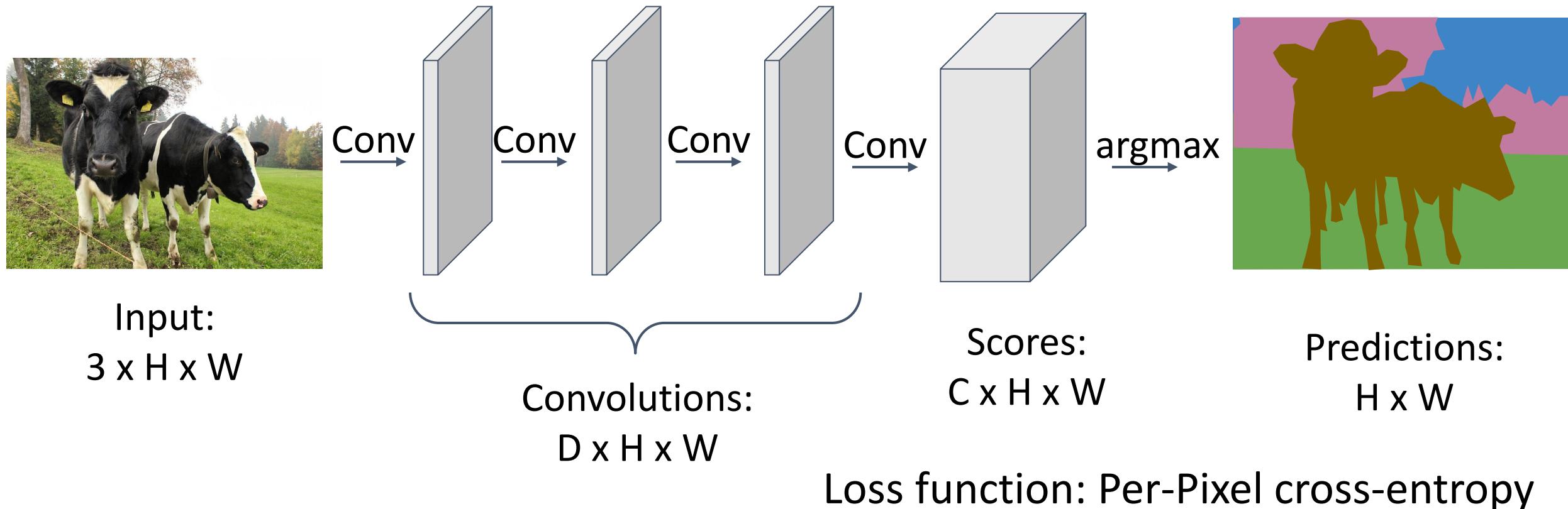
Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

Learn to combine coarse, high layer information with fine, low layer information. Pooling and prediction layers are shown as grids that reveal relative spatial coarseness, while intermediate layers are shown as vertical lines.

- First row (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step.
- Second row (FCN-16s): Combining predictions from both the final layer and the pool4 layer, at stride 16, lets the net predict finer details, while retaining high-level semantic information.
- Third row (FCN-8s): Additional predictions from pool3, at stride 8, provide further precision.

Semantic Segmentation-2: Fully Convolutional Network

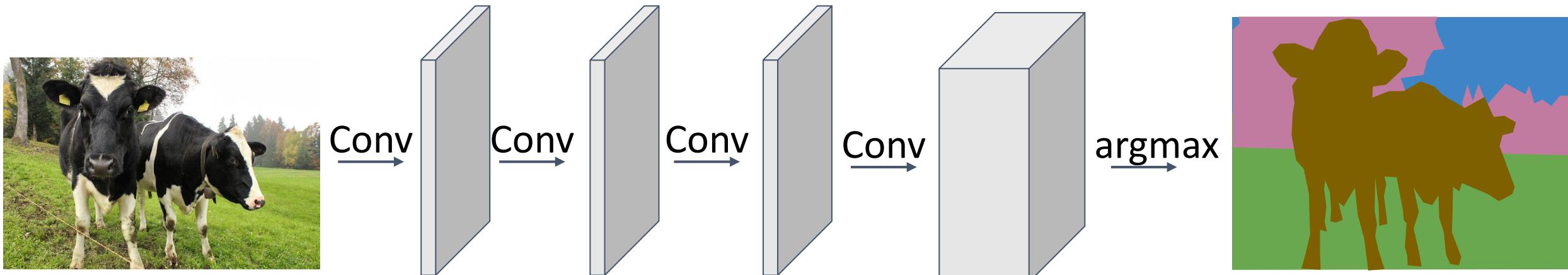
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!

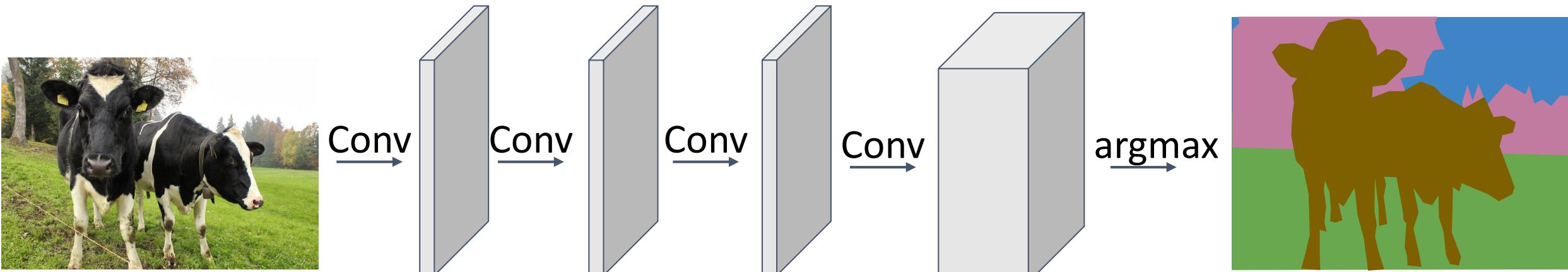


Input: **Problem #1:** Effective receptive
 $3 \times H \times W$ field size is linear in number of
conv layers: With L 3×3 conv
layers, receptive field is $1+2L$

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

Semantic Segmentation: Fully Convolutional Network

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Input: $3 \times H \times W$ **Problem #1:** Effective receptive field size is linear in number of conv layers: With L 3×3 conv layers, receptive field is $1+2L$

Problem #2: Convolution on high resolution images is expensive! Recall ResNet stem aggressively downsamples

Long et al, "Fully convolutional networks for semantic segmentation", CVPR 2015

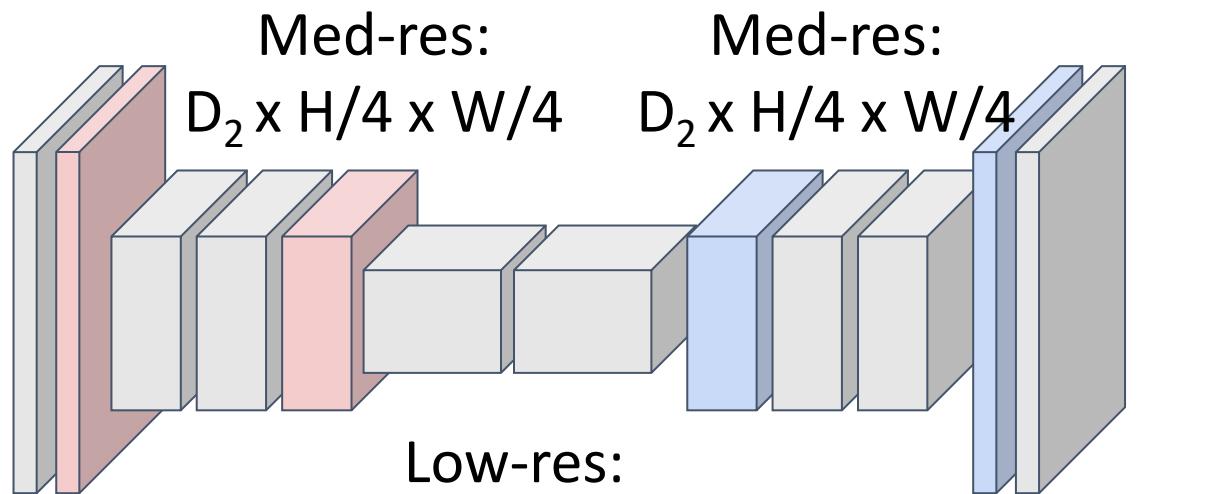
Semantic Segmentation Idea-3: Fully Convolutional Network

Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



Input:
 $3 \times H \times W$

High-res:
 $D_1 \times H/2 \times W/2$



Low-res:

$D_3 \times H/4 \times W/4$

High-res:
 $D_1 \times H/2 \times W/2$



Predictions:
 $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

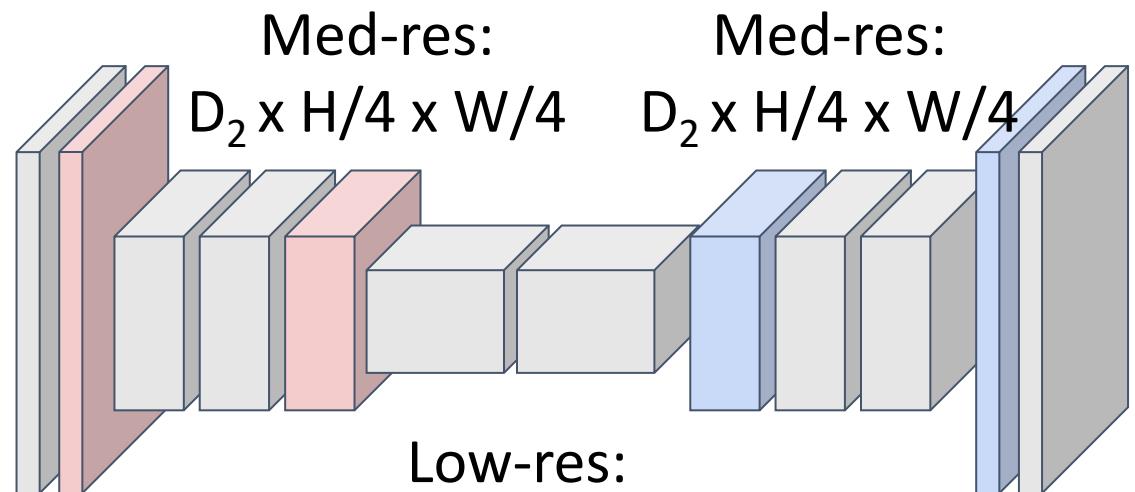
Semantic Segmentation Idea-3: Fully Convolutional Network

Downsampling:
Pooling, strided
convolution



Input:
 $3 \times H \times W$

High-res:
 $D_1 \times H/2 \times W/2$



Design network as a bunch of convolutional layers, with
downsampling and **upsampling** inside the network!



Predictions:
 $H \times W$

SegNet

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." *IEEE transactions on pattern analysis and machine intelligence* 39, no. 12 (2017): 2481-2495. 14K citations

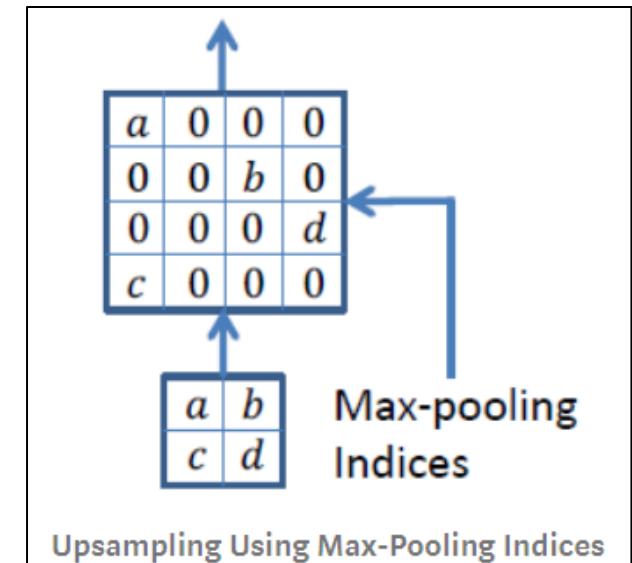
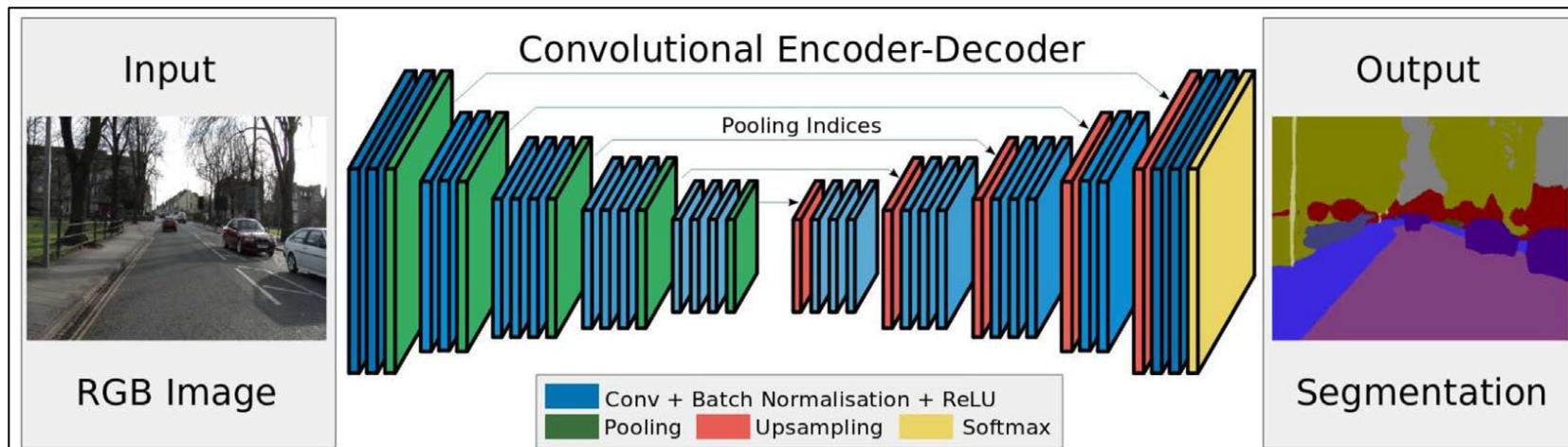
SegNet has an **encoder** network and a corresponding **decoder** network, followed by a final pixelwise classification layer.

Encoder

- At the encoder, convolutions and max pooling are performed.
- There are 13 convolutional layers from VGG-16. (The original fully connected layers are discarded.)
- While doing 2×2 max pooling, the corresponding max pooling indices (locations) are stored.

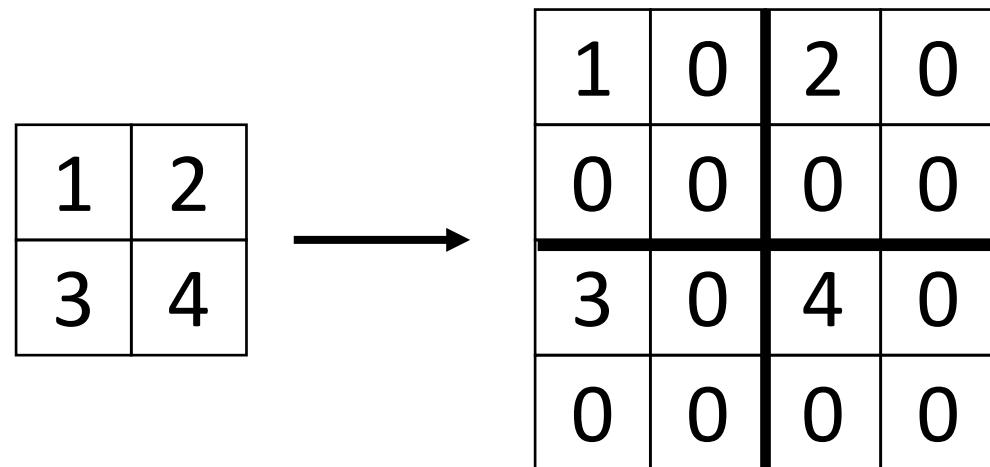
Decoder:

- At the decoder, upsampling and convolutions are performed. At the end, there is softmax classifier for each pixel.
- During upsampling, the max pooling indices at the corresponding encoder layer are recalled to upsample as shown above.
- Finally, a K-class softmax classifier is used to predict the class for each pixel.



In-Network Upsampling: “Unpooling”

Bed of Nails



Input
 $C \times 2 \times 2$

Output
 $C \times 4 \times 4$

In-Network Upsampling: “Unpooling”

Bed of Nails

1	2
3	4



1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Nearest Neighbor

1	2
3	4



1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Input
 $C \times 2 \times 2$

Output
 $C \times 4 \times 4$

Input
 $C \times 2 \times 2$

Output
 $C \times 4 \times 4$

In-Network Upsampling: Bilinear Interpolation

1		2	
3		4	



1.00	1.25	1.75	2.00
1.50	1.75	2.25	2.50
2.50	2.75	3.25	3.50
3.00	3.25	3.75	4.00

Input: $C \times 2 \times 2$

Output: $C \times 4 \times 4$

$$f_{x,y} = \sum_{i,j} f_{i,j} \max(0, 1 - |x - i|) \max(0, 1 - |y - j|) \quad i \in \{\lfloor x \rfloor - 1, \dots, \lceil x \rceil + 1\}$$

Use two closest neighbors in x and y
to construct linear approximations

$$j \in \{\lfloor y \rfloor - 1, \dots, \lceil y \rceil + 1\}$$

In-Network Upsampling: Bicubic Interpolation

1		2	
3		4	



0.68	1.02	1.56	1.89
1.35	1.68	2.23	2.56
2.44	2.77	3.32	3.65
3.11	3.44	3.98	4.32

Input: $C \times 2 \times 2$

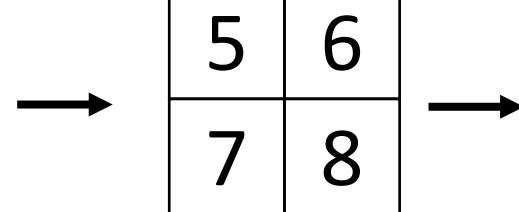
Output: $C \times 4 \times 4$

Use **three** closest neighbors in x and y to
construct **cubic** approximations
(This is how we normally resize images!)

In-Network Upsampling: “Max Unpooling”

Max Pooling: Remember which position had the max

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



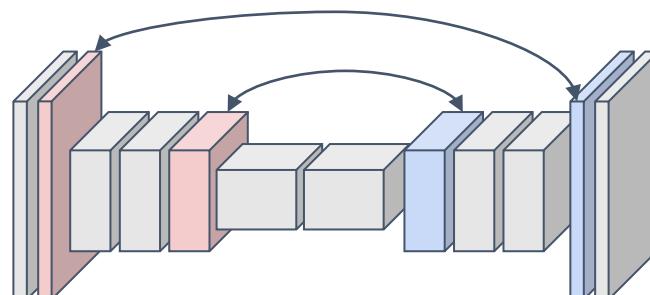
5	6
7	8

Rest
of
net

1	2
3	4

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Max Unpooling: Place into remembered positions



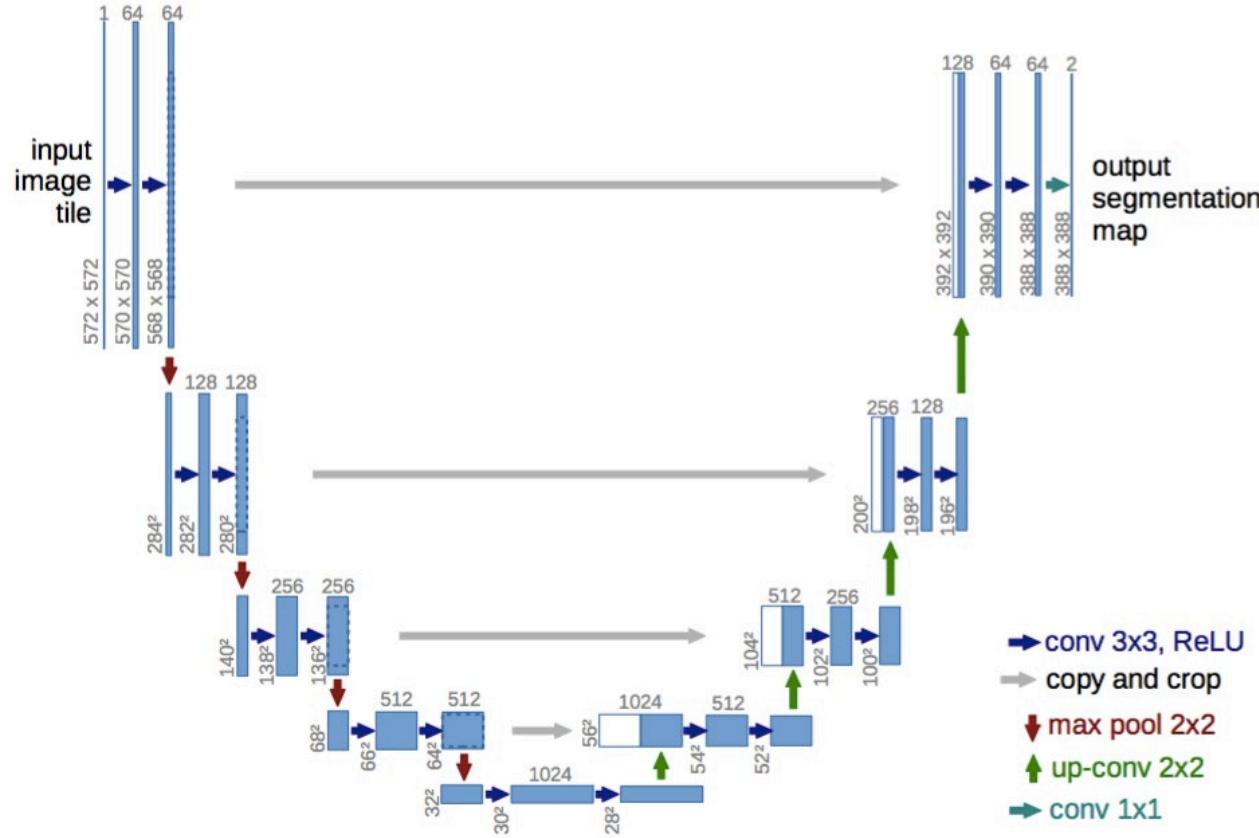
Pair each downsampling layer with an upsampling layer

Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

U-Net

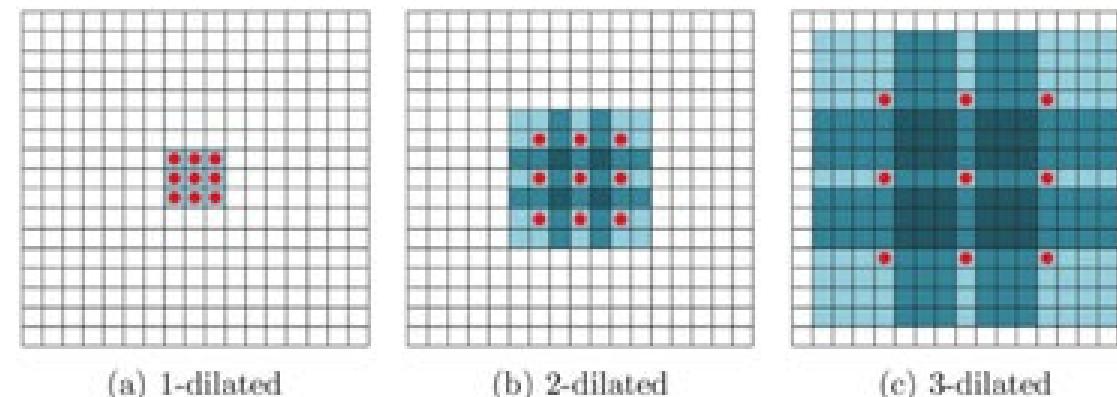
Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234-241. Springer, Cham, 2015. 58K citations

- Encoder gradually reduces the spatial dimension with pooling layers and decoder gradually recovers the object details and spatial dimension.
 - Shortcut connections from encoder to decoder help decoder recover the object details better
 - **Difference from SegNet:** Instead of using pooling indices, the entire feature maps are transferred from encoder to decoder, then with concatenation to perform convolution.



Semantic Segmentation Idea-4: Dilated/Atrous Convolutions

- Pooling helps in classification networks because receptive field increases. But this is not the best thing to do for segmentation because pooling decreases the resolution.
- Dilated (a.k.a. “atrous”) convolution introduces to convolutional layers another parameter, the dilation rate.
- For example a 3×3 kernel with a dilation rate of 2 will have the same size receptive field as a 5×5 kernel while using only 9 parameters, thus enlarging the receptive field with no increase in computational cost.



Semantic Segmentation Idea-3: Dilated/Atrous Convolutions

- Dilated convolutions have been popular in the field of real-time segmentation, and many recent publications report the use of this technique. Some of the most important include
 - DeepLab family [76],
 - Multiscale context aggregation[77],
 - Dense Upsampling Convolution and Hybrid Dilated Convolution (DUC-HDC) [78],
 - densely connected Atrous Spatial Pyramid Pooling (DenseASPP) [79],
 - Efficient Network (ENet) [80].

[76] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

[77] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” 2015, arXiv:1511.07122.

[78] P. Wang et al., “Understanding convolution for semantic segmentation,” in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2018, pp. 1451–1460.

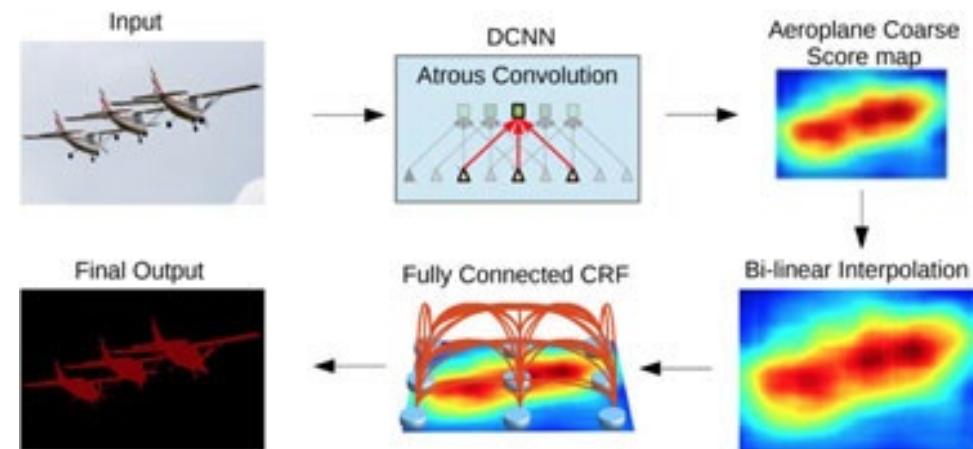
[79] M. Yang, K. Yu, C. Zhang, Z. Li, and K. Yang, “DenseASPP for semantic segmentation in street scenes,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 3684–3692.

[80] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, “ENet: A deep neural network architecture for real-time semantic segmentation,” 2016, arXiv:1606.02147.

DeepLab

DeepLabv1 [36] and DeepLabv2 [76], developed by Chen et al., are among the most popular image segmentation models. The latter has three key features (Fig. 23).

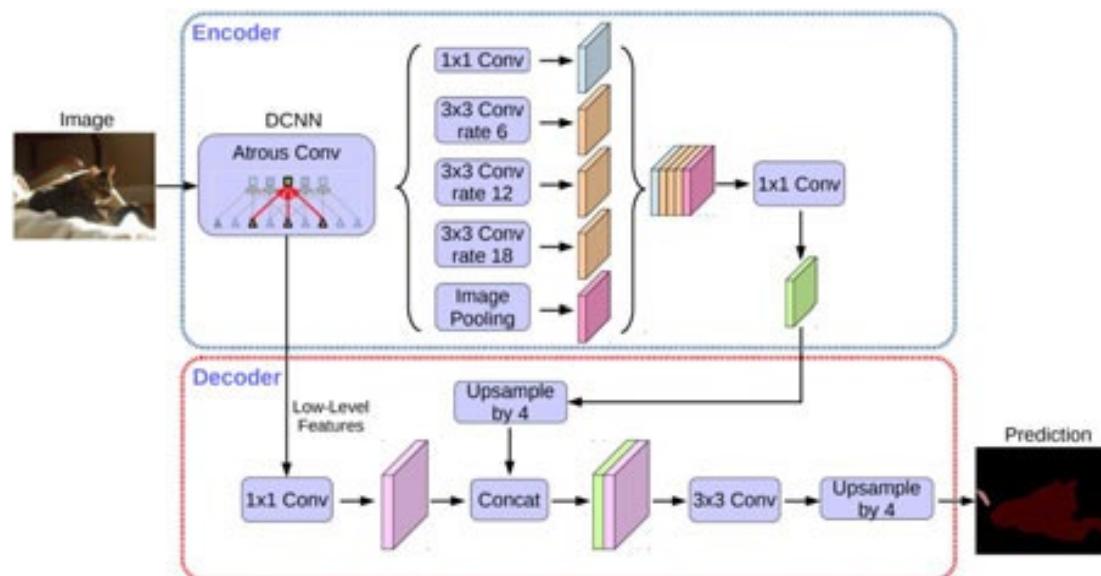
1. Use of dilated convolution to address the decreasing resolution in the network caused by max-pooling and striding.
2. Atrous Spatial Pyramid Pooling (ASPP), which probes an incoming convolutional feature layer with filters at multiple sampling rates, thus capturing objects as well as multiscale image context to robustly segment objects at multiple scales.
3. Improved localization of object boundaries by combining methods from deep CNNs, such as fully convolutional VGG-16 or ResNet 101, and probabilistic graphical models, specifically fully-connected CRFs.



[76] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2018.

DeepLab

- Subsequently, Chen et al. [13] proposed DeepLabv3, which combines cascaded and parallel modules of dilated convolutions.
- The parallel convolution modules are grouped in the ASPP. A 1 1 convolution and batch normalization are added in the ASPP.
- All the outputs are concatenated and processed by another 1 1 convolution to create the final output with logits for each pixel.



MULTI-SCALE CONTEXT AGGREGATION BY DILATED CONVOLUTIONS

Fisher Yu (Princeton University), Vladlen Koltun (Intel Labs), ICLR 2016, 8K citations.

- Dilated convolutional layer (also called as atrous convolution in [DeepLab](#)) allows for exponential increase in field of view without decrease of spatial dimensions.
- Last two pooling layers from pretrained classification network (here, VGG) are removed and subsequent convolutional layers are replaced with dilated convolutions.
- In particular, convolutions between the pool-3 and pool-4 have dilation 2 and convolutions after pool-4 have dilation 4. With this module (called *frontend module* in the paper), dense predictions are obtained without any increase in number of parameters.
- A module (called *context module* in the paper) is trained separately with the outputs of frontend module as inputs. This module is a cascade of dilated convolutions of different dilations so that multi scale context is aggregated and predictions from frontend are improved.

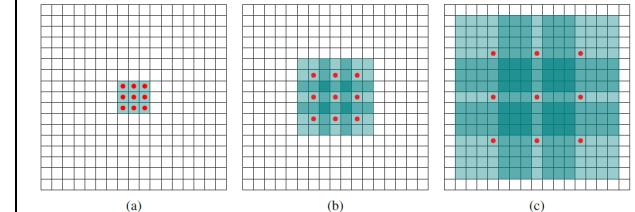


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3 . (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7 . (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15 . The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

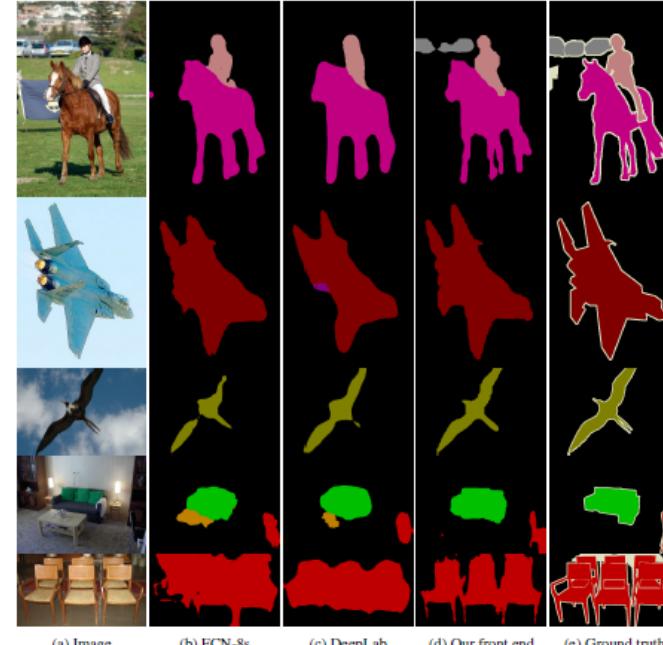
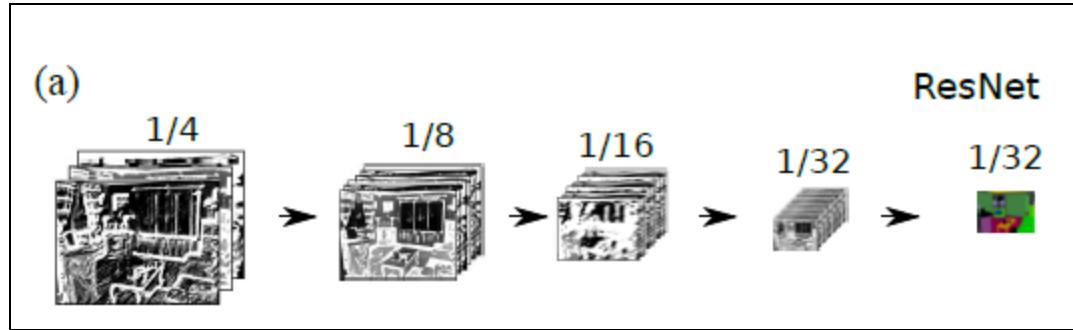


Figure 2: Semantic segmentations produced by different adaptations of the VGG-16 classification network. From left to right: (a) input image, (b) prediction by FCN-8s ([Long et al., 2015](#)), (c) prediction by DeepLab ([Chen et al., 2015a](#)), (d) prediction by our simplified front-end module, (e) ground truth.

RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation

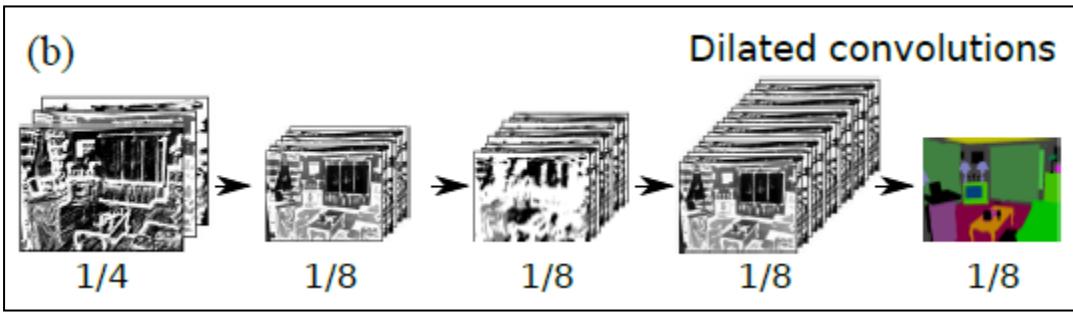
Guosheng Lin, Anton Milan, Chunhua Shen, Ian Reid, CVPR 2017, (2K citations)



Standard multi-layer CNNs, such as

ResNet:

- downscale the feature maps,
- lose fine structures along the way.

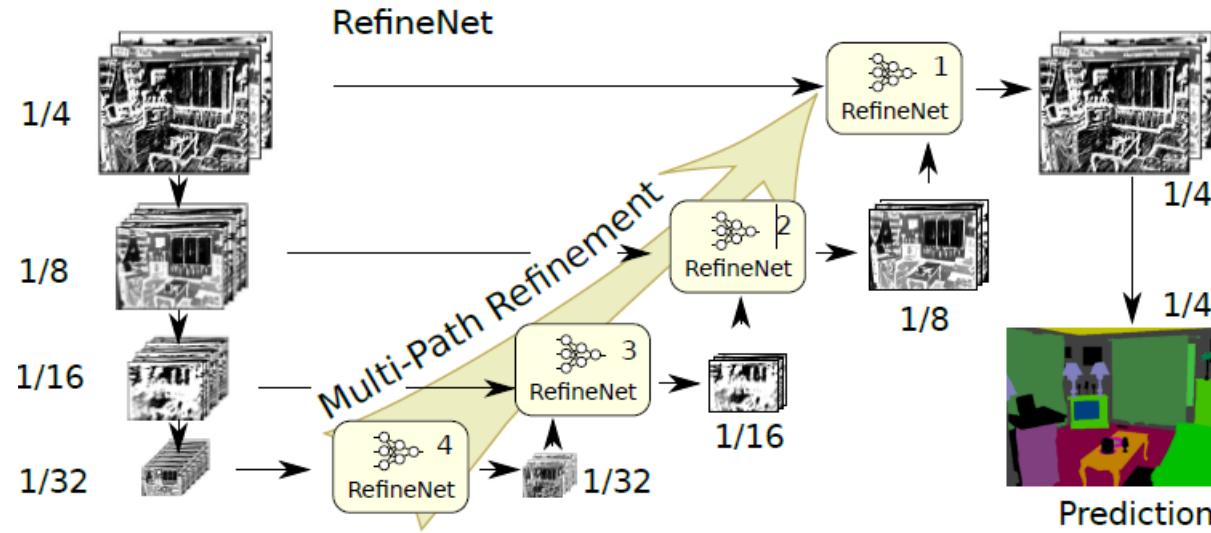


Dilated convolutions (b):

- remedy this shortcoming by introducing atrous filters,
- but are computationally expensive to train
- quickly reach memory limits even on modern GPUs.

RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation

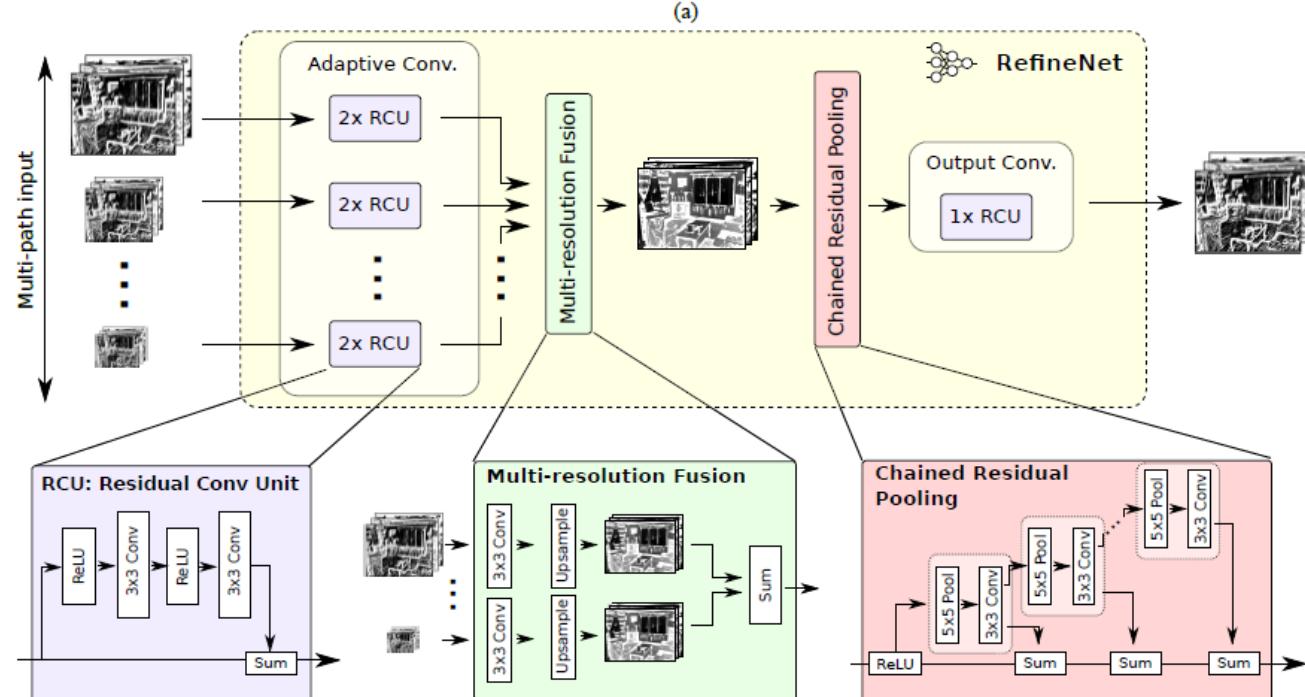
Guosheng Lin, Anton Milan, Chunhua Shen, Ian Reid, CVPR 2017, (2K citations)



RefineNet : exploits various levels of detail at different stages of convolutions and fuses them to obtain a high-resolution prediction without the need to maintain large intermediate feature maps.

RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation

Guosheng Lin, Anton Milan, Chunhua Shen, Ian Reid, CVPR 2017, (2K citations)



Residual convolution unit.

- Adaptive convolution set that mainly fine-tunes the pretrained ResNet weights for our task.
- Each input path is passed sequentially through two residual convolution units (RCU), which is a simplified version of the convolution unit in the original ResNet,
- Batch-normalization layers are removed
- The filter number for each input path is set to 512 for RefineNet-4 and 256

CS/ECE 8690

Image Segmentation Unsupervised

Gonzalez & Woods – Chapter 10

Szeliski – Chapter 5.2

Forsyth & Ponce – Chapter 14

Instructor: Filiz Bunyak Ersoy bunyak@missouri.edu

Recap

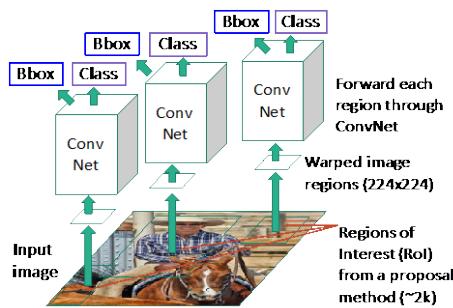
	Week	Lecture Topics	HW due
1	1-Jan 17-19	Syllabus, introduction to CV, image formation	
2	2-Jan24-26	Intensity transformations, filtering, python tutorial	
3	3-Jan 31-Feb 2	Edge detection, blob detection	Hw1a (Feb 2)
4	4-Feb 7-9	Feature matching	Hw1b (Feb 9)
5	5-Feb 14-16	Geometric transformations, PyTorch training	
6	6-Feb 21-23	Image Alignment	Hw2 (Feb 23)
7	7-Feb 28-March 2	Intro to Visual Recognition	
8	8-Mar 7-9	Image Classification	Hw3a (Mar 7)
9	9-Mar 14-16	Deep Detection	
10	10-Mar 21-23	Segmentation	Project Proposal Due, Hw3b (Mar 21)
	11-Mar 28-30	Spring Break	
11	12-Apr 4-6	Motion Analysis & Video	Hw4a (Apr 4)
12	13-Apr 11-13	Motion Analysis & Video	Project Progress, Hw4b (Apr 13)
13	14- Apr 18-20	Multi-view Image Analysis	
14	14-Apr 25-27	Multi-view Image Analysis	
15	15-May 2-4	Final Project Presentations	Final Project Due
16	16-May 9-11	Final Exam Week / Final Project Report (Dec 12)	

HWs

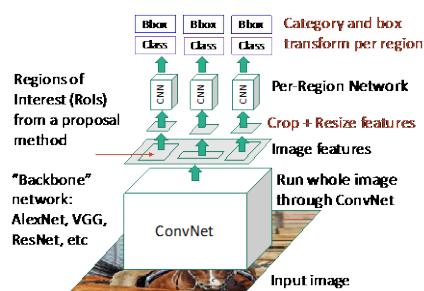
1. Homework 1A: Point Processes in C/C++ [40 pts]
2. Homework 1B: Experiments with Color Edge Detection [60 pts]
3. Homework 2: Feature Detection, Description, and Matching [100 pts]
4. Homework 3A: Object Detection using Pre-trained Deep Learning Networks [40 pts]
5. Homework 3B: Image Classification [60 pts]
6. Homework 4A: Semantic Segmentation using Pre-trained Deep Learning Networks [40 pts]
7. Homework 4B: Moving Object Detection using Simple Background Subtraction [60 pts]

Summary

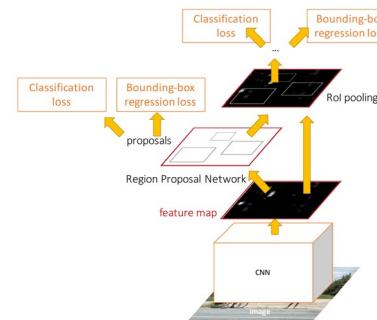
“Slow” R-CNN: Run CNN independently for each region



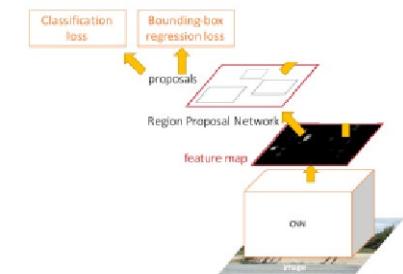
Fast R-CNN: Apply differentiable cropping to shared image features



Faster R-CNN: Compute proposals with CNN

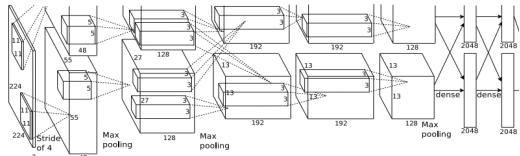
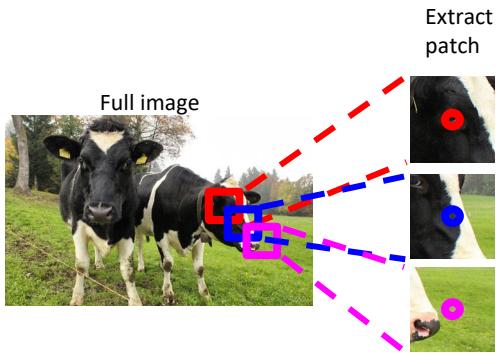


Single-Stage: Fully convolutional detector



Deep Segmentation - Summary

1) Sliding window image patch classification



2) Fully Convolutional Networks

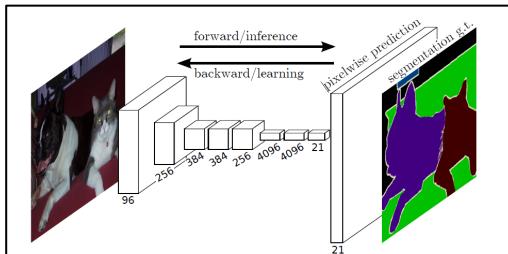
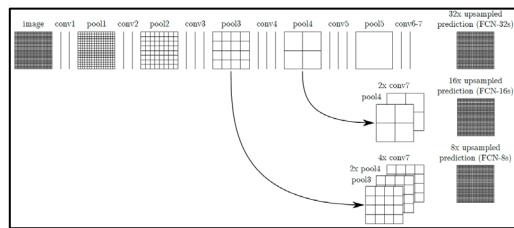
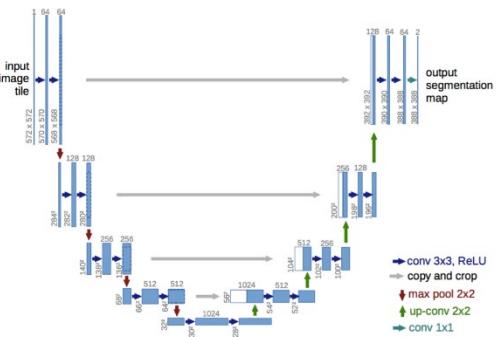
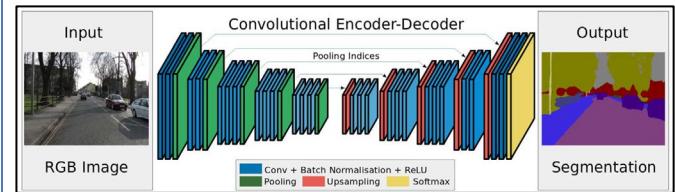


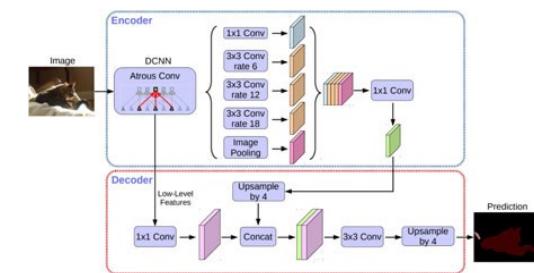
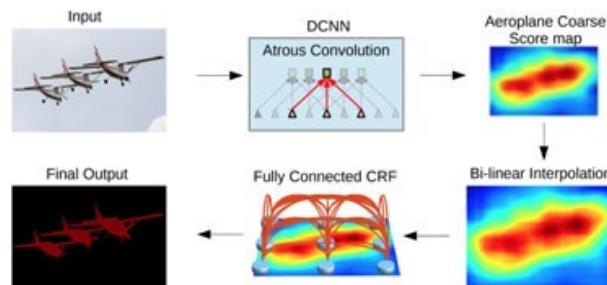
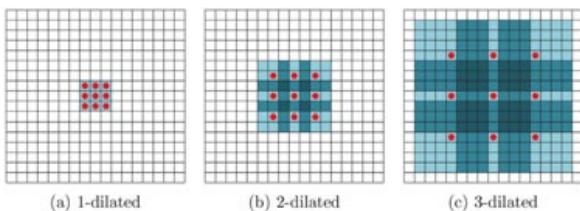
Fig. 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.



3) Encoder-decoder Networks

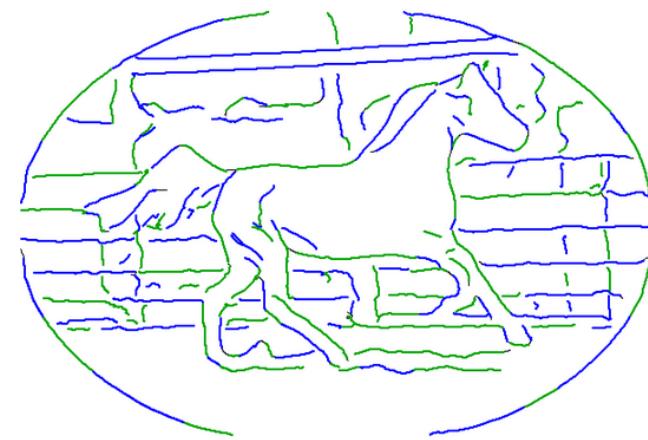


4) Dilated/Atrous Convolutions --- DeepLab variants



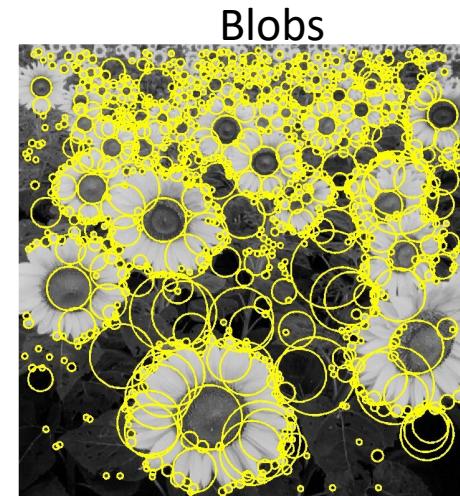
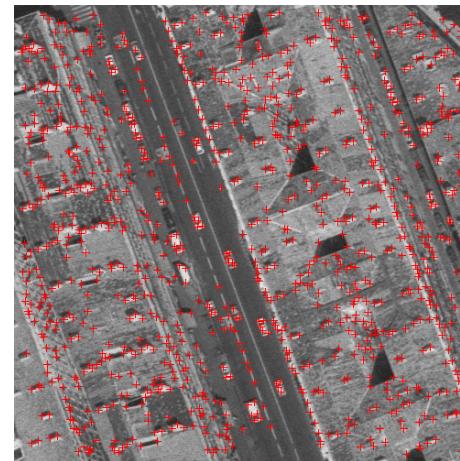
Feature Analysis

Contours/Line segments

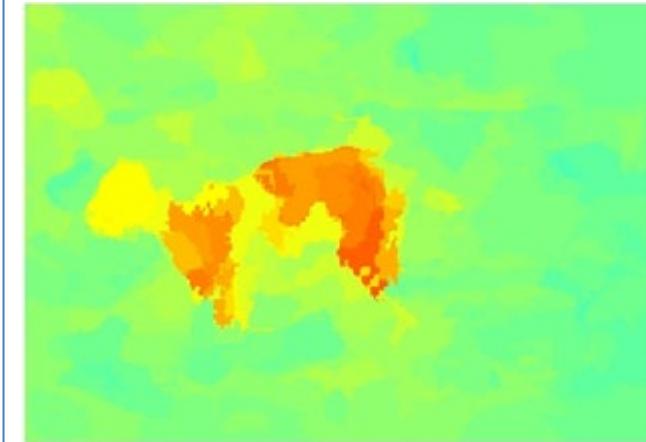


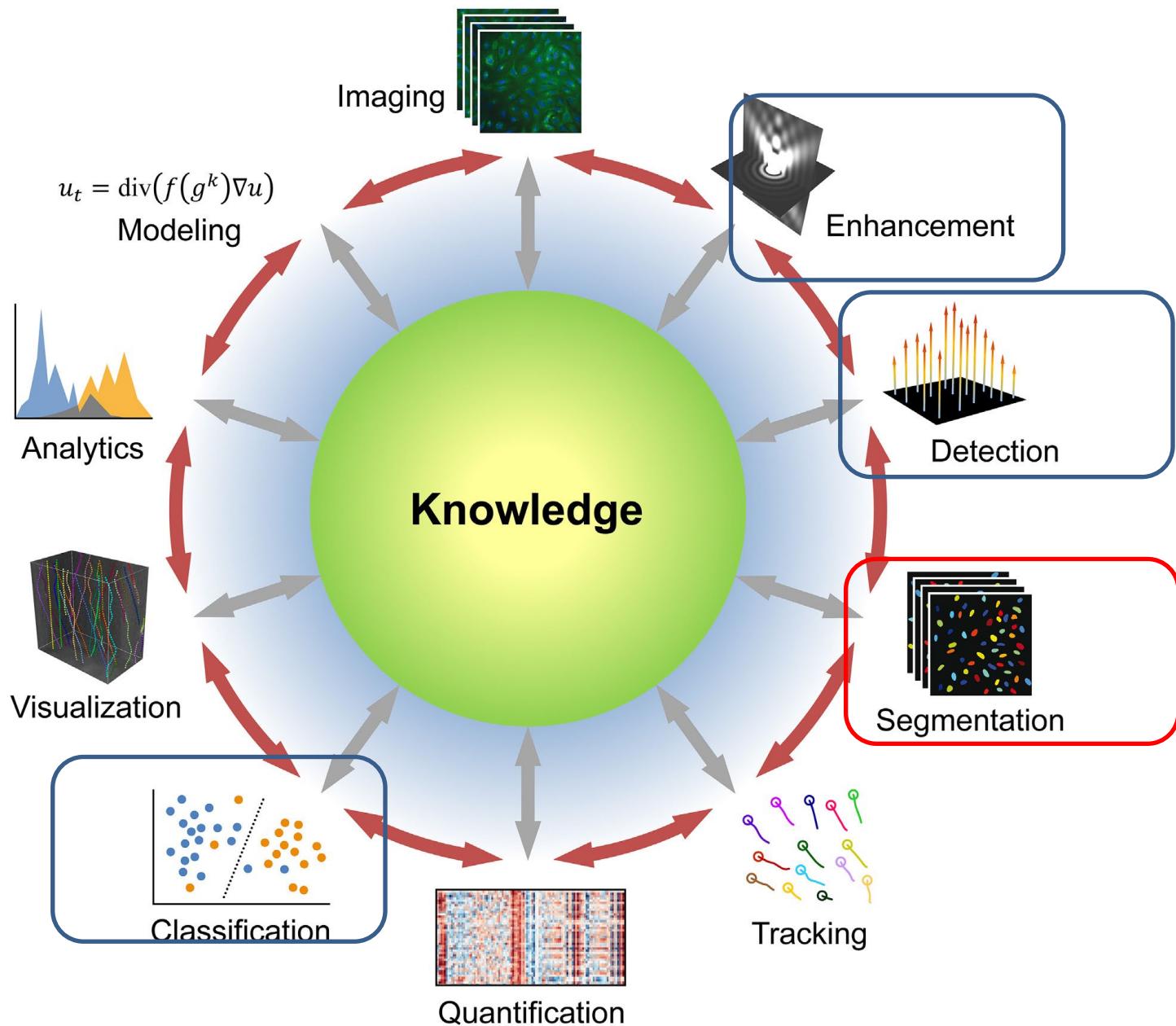
Interest Points

Corners



Regions/Segmentation





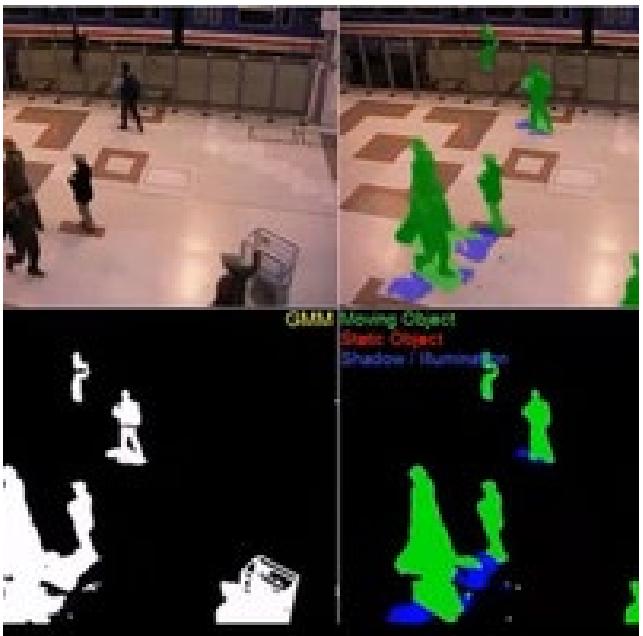
Common tasks in image analysis.

Application: intelligence on the road

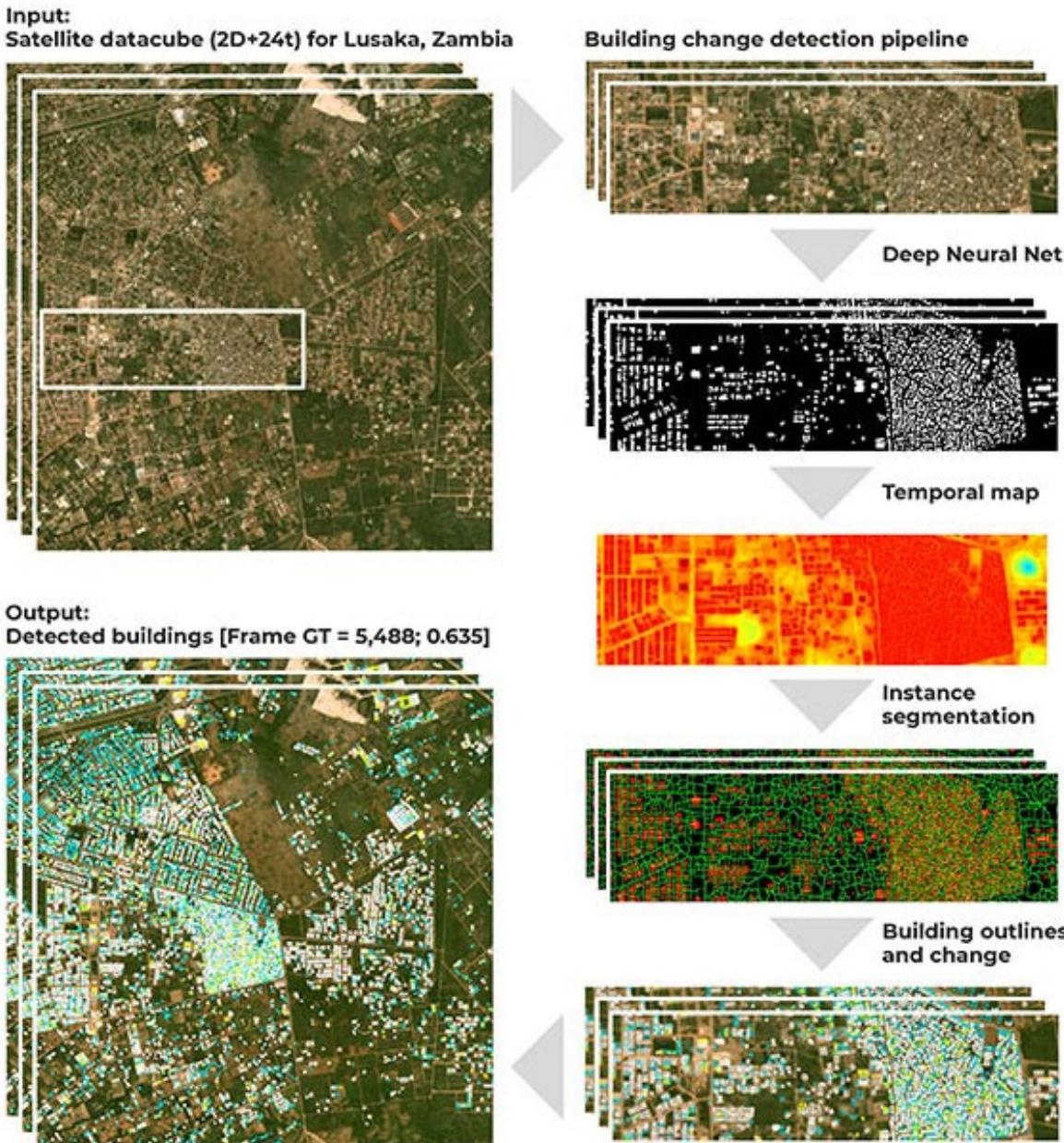


Slide credit: Ashok Veeraraghavan

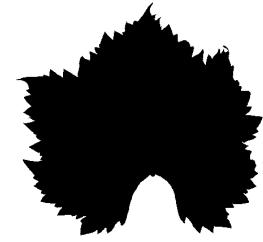
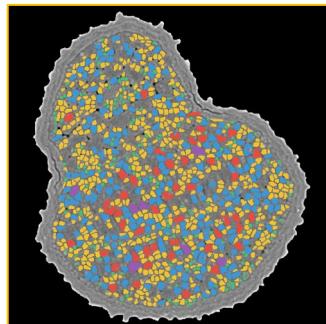
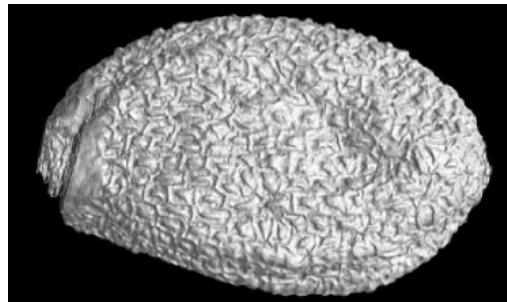
Application: Surveillance & Geospatial



CVPR 2014 Change Detection
Challenge Winner

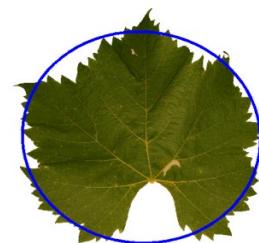


Plant Sciences & Agriculture



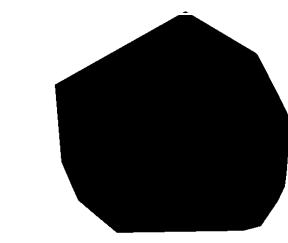
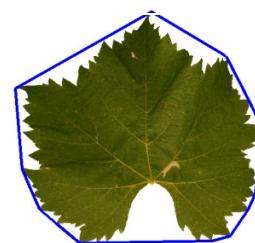
Original Image

Binary Mask



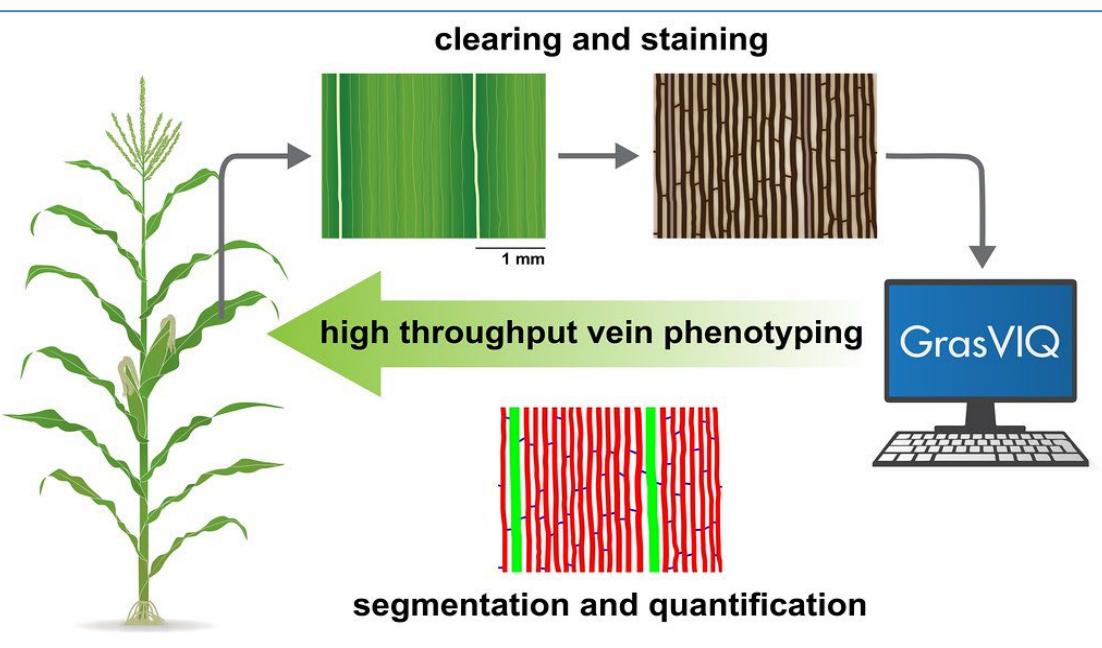
Contour

Fit Ellipse

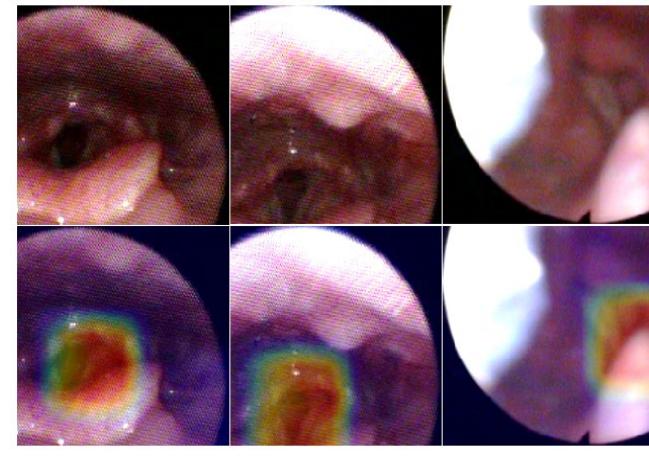
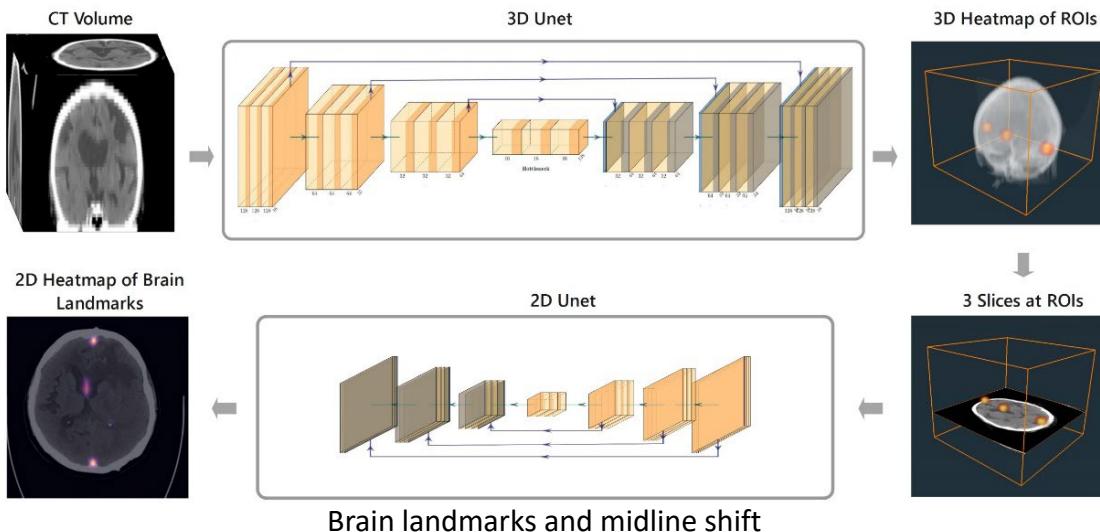
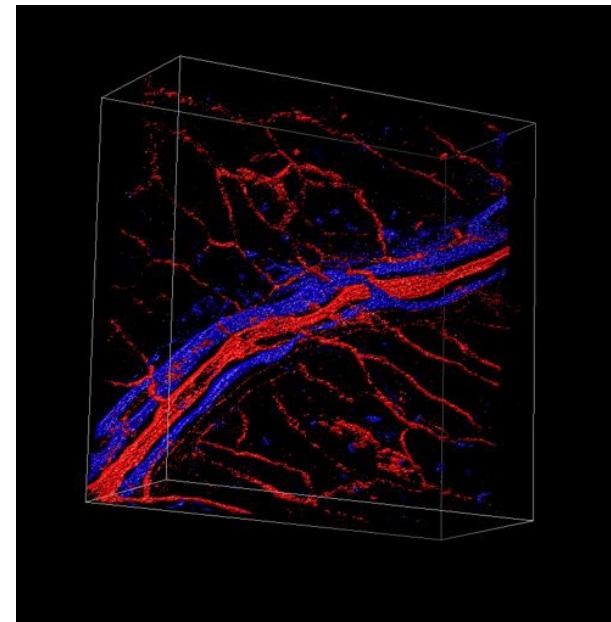
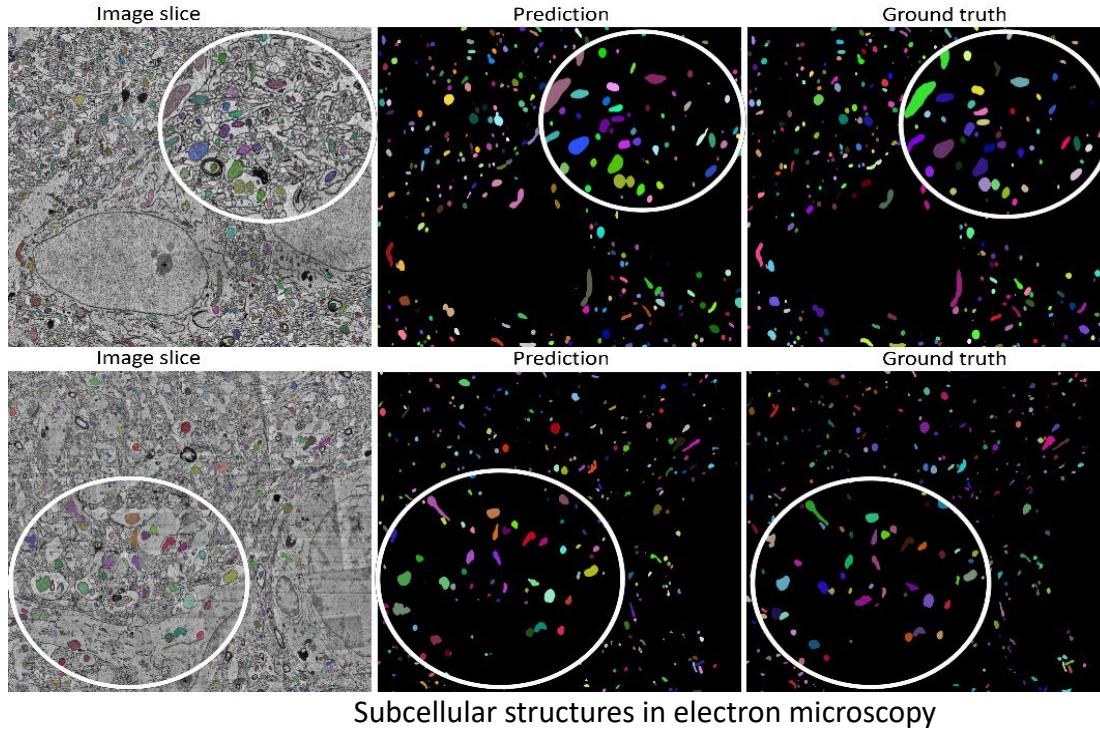


Convex Hull

Convex Hull Binary



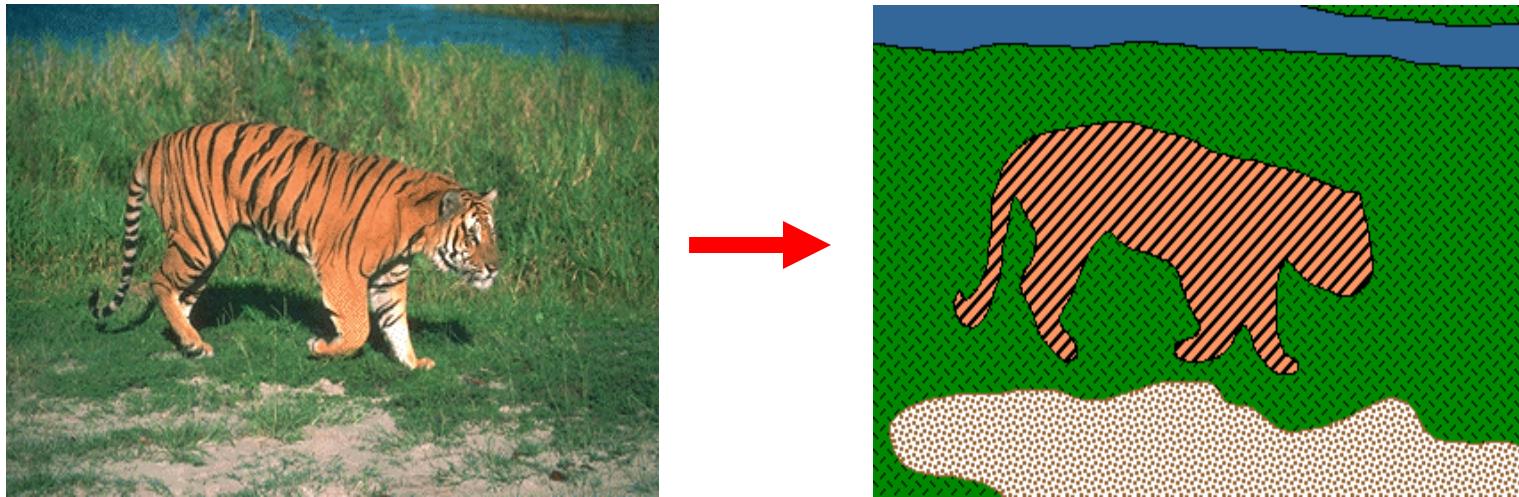
Application: Biomedical



Segmentation

- The purpose of image segmentation is to partition an image into ***meaningful regions*** with respect to a ***particular application***
- The segmentation is based on **measurements** taken from the image and might be
 - *greylevel,*
 - *color,*
 - *texture,*
 - *Depth,*
 - *motion.....*

From images to objects



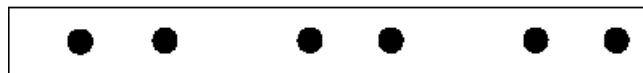
- What Defines an Object?
 - Subjective problem, but has been well-studied
 - Gestalt Laws seek to formalize this
 - proximity, similarity, continuation, closure, common fate

Segmentation cues

- What Defines an Object? (what are properties of pixels that belong together?)
 - Subjective problem, but has been well-studied
 - Gestalt Laws seek to formalize this: proximity, similarity, continuation, closure, common fate...



Not grouped



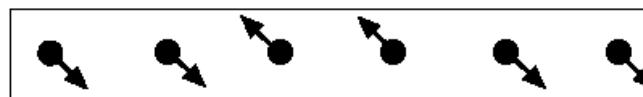
Proximity



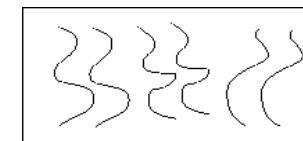
Similarity



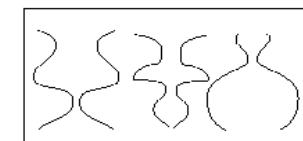
Similarity



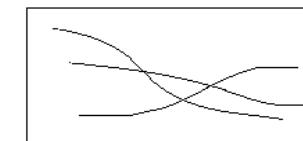
Common Fate



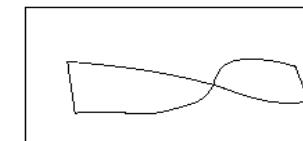
Parallelism



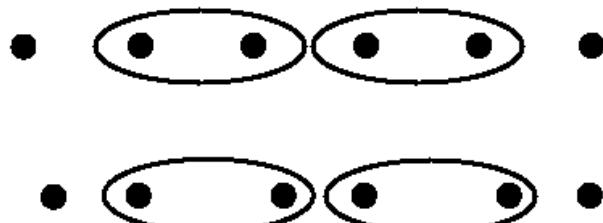
Symmetry



Continuity



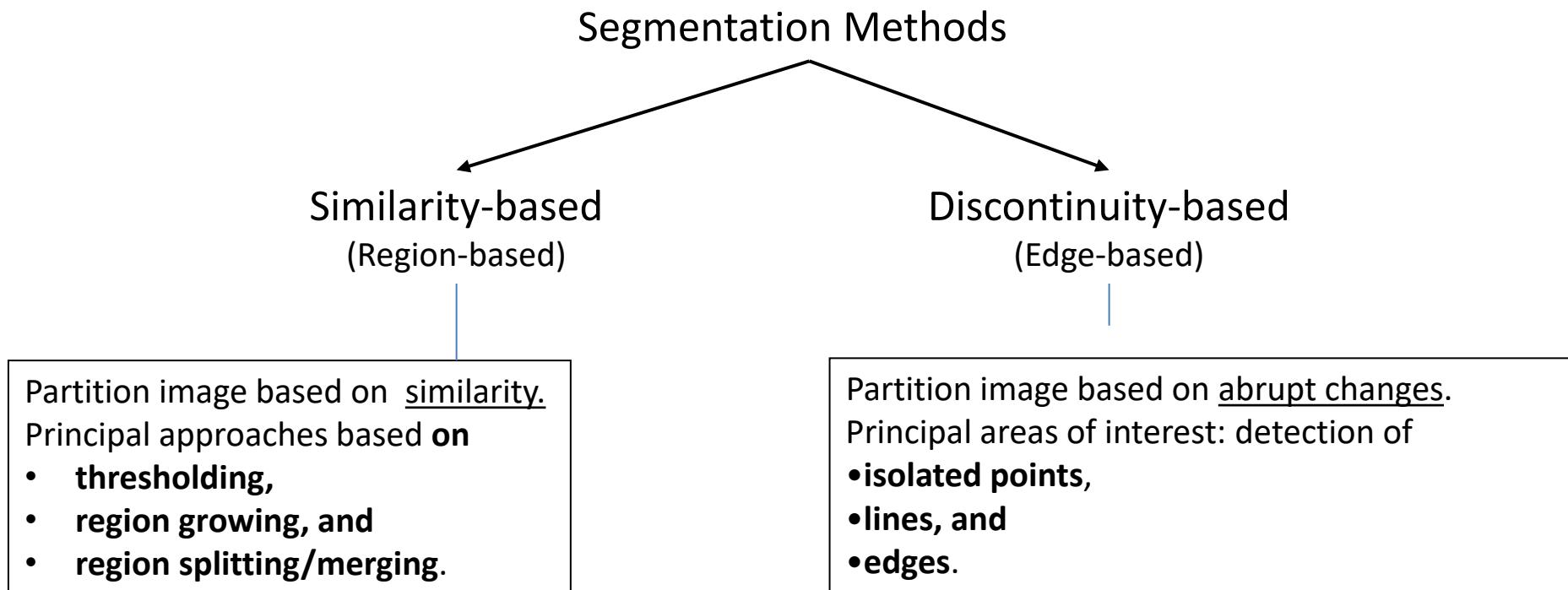
Closure



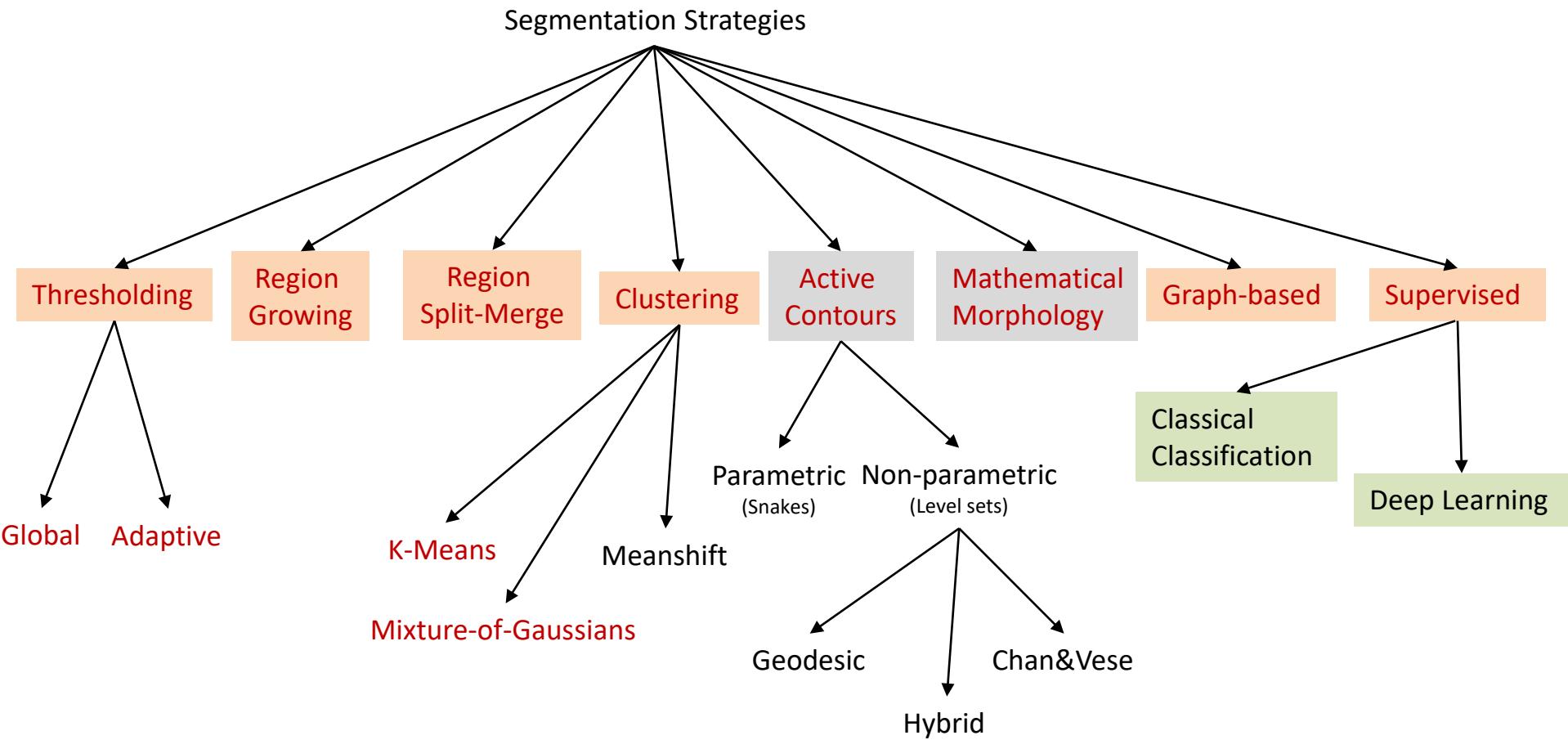
Common Region

Classification of Segmentation Methods

Segmentation algorithms generally are based on one of two basic properties **similarity** and **discontinuity**:

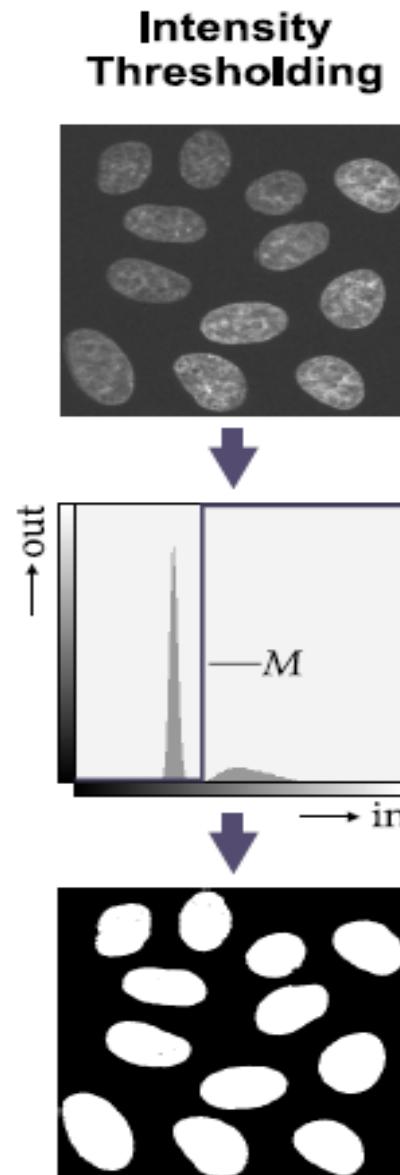


Some Image Segmentation Strategies



1. Global Thresholding

- Binary images can be obtained from gray-scale images by thresholding operations.
- A thresholding operation chooses some of the pixels
 - as the foreground pixels that make up the objects of interest and
 - the rest as background pixels.



2. Adaptive/Local Thresholding Based on Local Image Properties

- Compute a threshold at every point (x,y) based on one or more specified properties in a neighborhood
- Thresholding based on local mean and variance

Local standard deviation for neighborhood centered at (x,y)

Local mean for neighborhood centered at (x,y)

$$T_{xy} = a\sigma_{xy} + bm_{xy}$$

$$T_{xy} = a\sigma_{xy} + bm_G$$

global mean

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T_{xy} \\ 0 & \text{if } f(x,y) \leq T_{xy} \end{cases}$$

Significant power (with a modest increase in computation) can be added to variable thresholding by using predicates (Q) based on the parameters computed in the neighborhood of a point (x, y) :

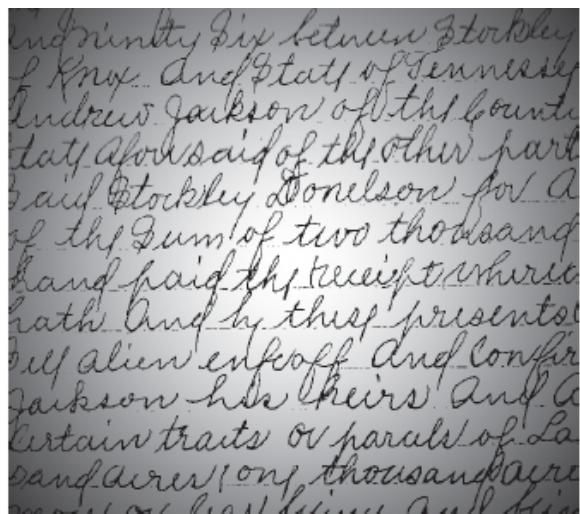
$$g(x,y) = \begin{cases} 1 & \text{if } Q(\text{local parameters}) \text{ is TRUE} \\ 0 & \text{if } Q(\text{local parameters}) \text{ is FALSE} \end{cases}$$

Example predicate

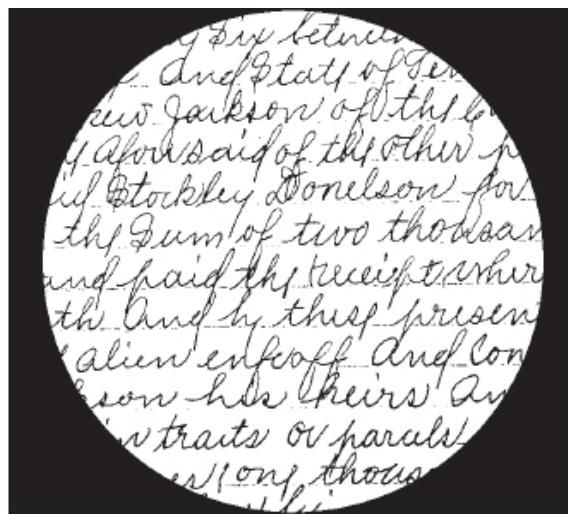
$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} \text{TRUE} & \text{if } f(x,y) > a\sigma_{xy} \text{ AND } f(x,y) > bm_{xy} \\ \text{FALSE} & \text{otherwise} \end{cases}$$

Gonzalez & Woods Figure 10.44

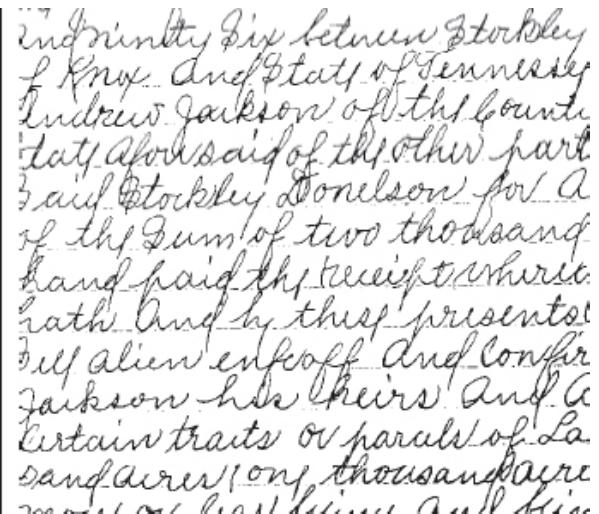
(a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.



a b c



Global thresholding



Local thresholding

3. Segmentation by Region Growing

Start from selected seed points in the image and to iteratively add connected points to form labeled regions. Assumes (and suffers from) a similar image model as in the case of intensity thresholding.

Ordinary region growing:

- The most straightforward implementation: ordinary region growing,

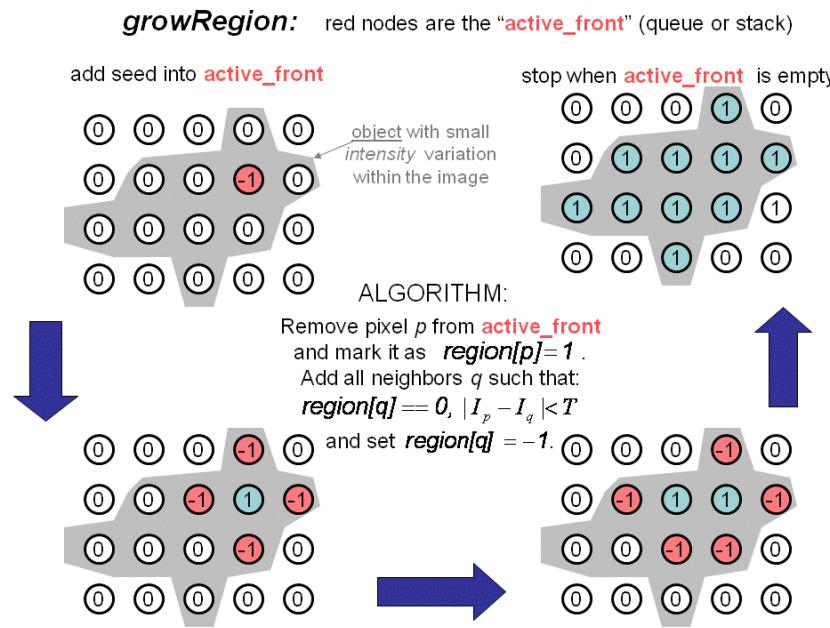


Figure: Ulas Bagci, UCF

3. Segmentation by Region Growing

Region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria for growth.

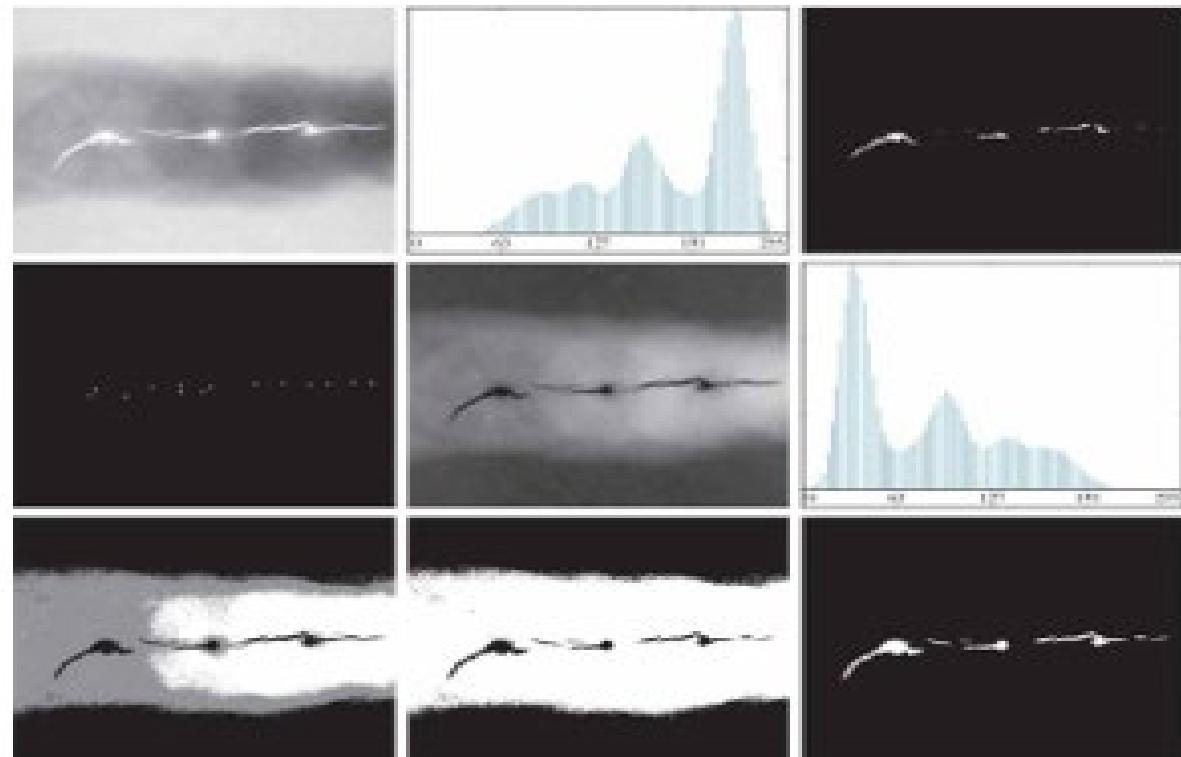
The basic approach:

- Start with a set of “seed” points,
- From the seeds grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as ranges of intensity or color).

1. Find all connected components in $S(x, y)$ and reduce each connected component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.
2. Form an image f_Q such that, at each point (x, y) , $f_Q(x, y) = 1$ if the input image satisfies a given predicate, Q , at those coordinates, and $f_Q(x, y) = 0$ otherwise.
3. Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.
4. Label each connected component in g with a different region label (e.g., integers or letters). This is the segmented image obtained by region growing.

3. Segmentation by Region Growing

- (a) X-ray image of a defective weld.
- (b) Histogram.
- (c) Initial seed image.
- (d) Final seed image (the points were enlarged for clarity).
- (e) Absolute value of the difference between the seed value (255) and (a).
- (f) Histogram of (e).
- (g) Difference image thresholded using dual thresholds.
- (h) Difference image thresholded with the smallest of the dual thresholds.
- (i) Segmentation result obtained by region growing.
(Original image courtesy of X-TEK Systems, Ltd.)



a	b	c
d	e	f
g	h	i

4. Segmentation by Region Splitting and Merging

- Subdivide an image initially into a set of disjoint regions and then merge and/or split the regions in an attempt to satisfy the conditions of segmentation
- Hierarchical split-and-merge schemes: operating per resolution layer and using some uniformity predicate.

The preceding discussion can be summarized by the following procedure in which, at any step, we

1. Split into four disjoint quadrants any region R_i for which $Q(R_i) = \text{FALSE}$.
2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_k) = \text{TRUE}$.
3. Stop when no further merging is possible.

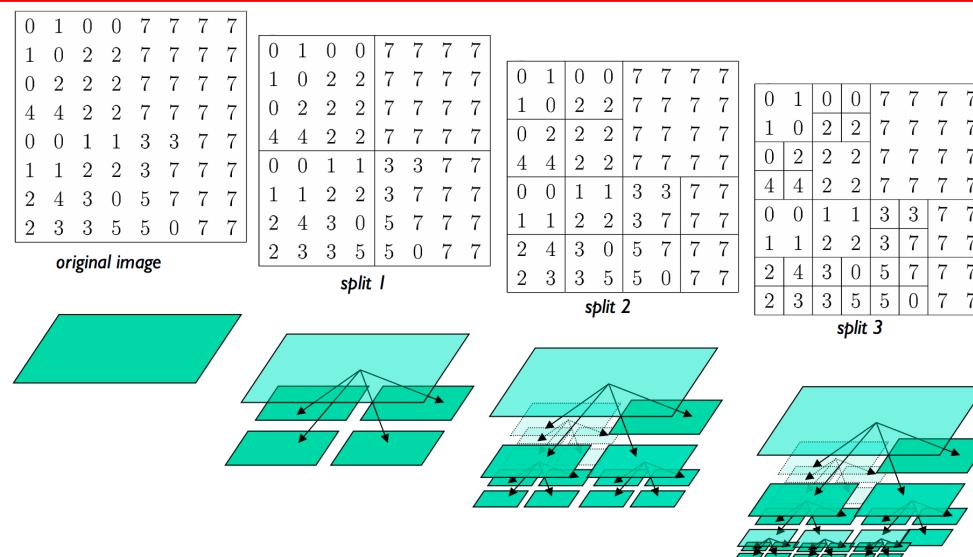


Figure: Ulas Bagci, UCF

Quadtree

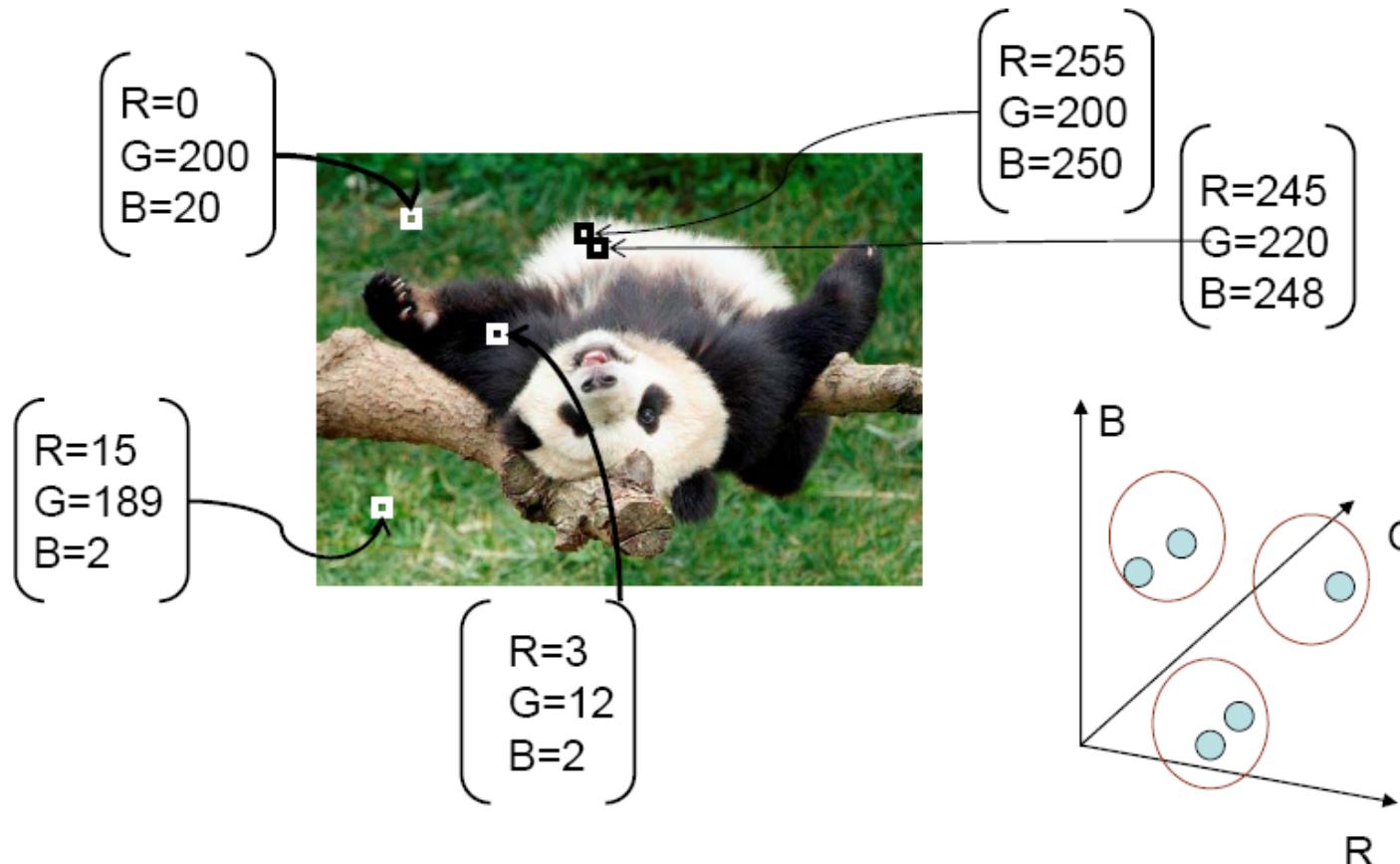


<http://devmag.org.za/2011/02/23/quadtrees-implementation/>

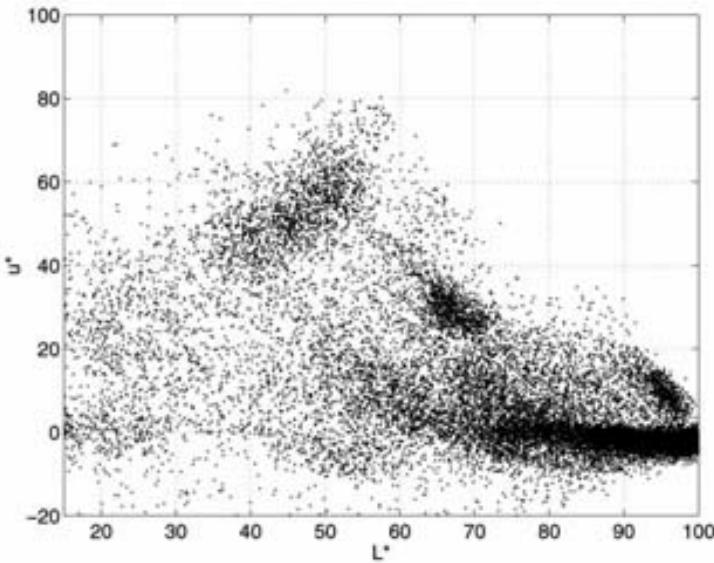
Geometry in Action: <https://www.ics.uci.edu/~eppstein/gina/quadtree.html>

5. Segmentation as clustering

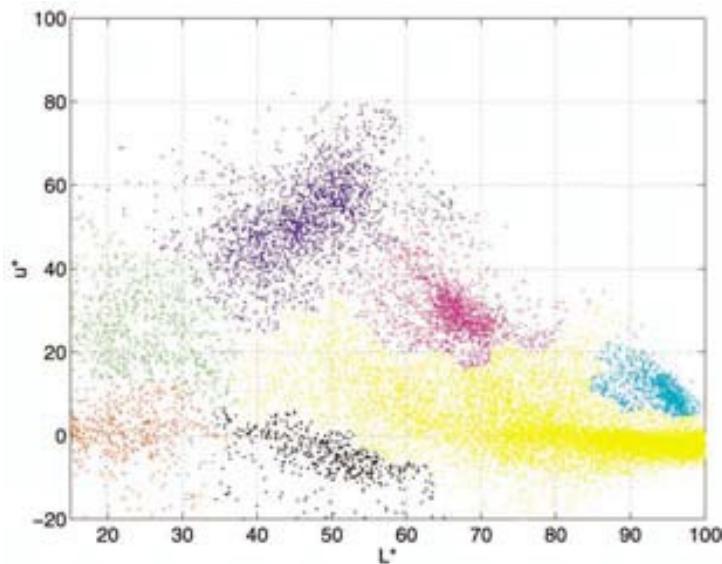
- Cluster similar pixels (features) together



Segmentation as clustering

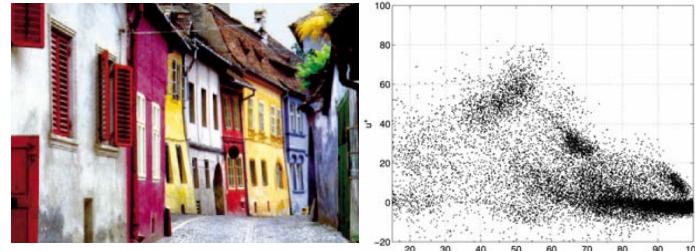


How to choose the representative colors?



Mode Finding (K-means, Mixture of Gaussians) + Meanshift

1. Pixel → color, position, texture.... → Feature vector
 - Feature vector : Samples from unknown probability density function
2. Try to find clusters (modes) in this distribution



K-Means & MoG

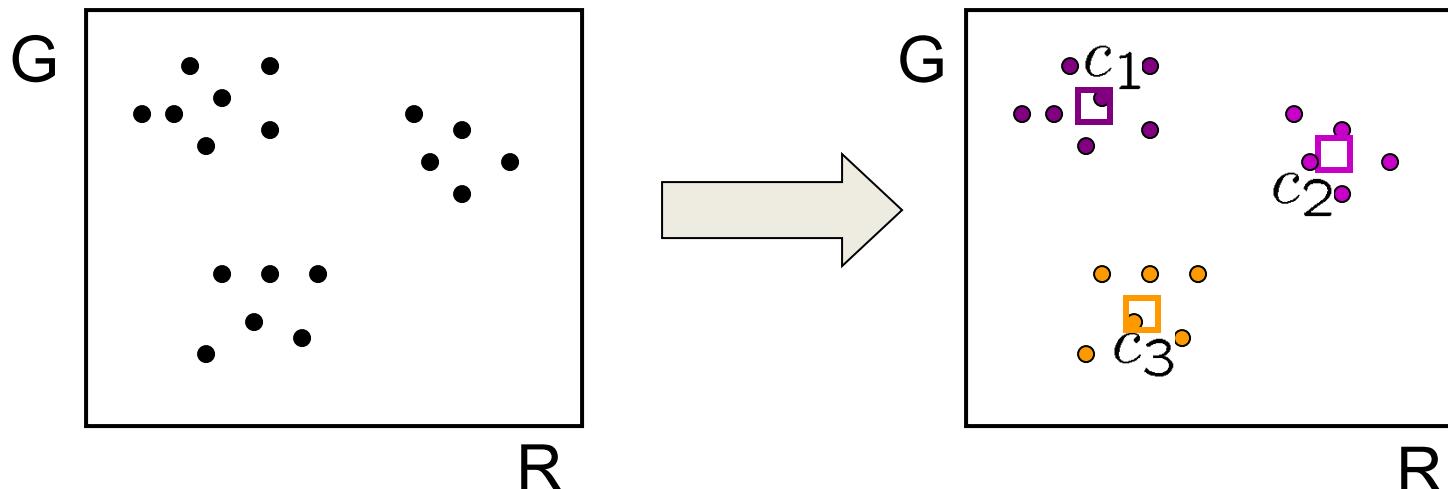
- Parametric model
- Superposition of simpler distributions (Gaussians)
- Center & Shape (covariance) can be estimated

Meanshift

- Non-parametric
- Smooth the distribution
- Find peaks & regions corresponding to peaks

Clustering

- How to choose the representative colors?
 - This is a clustering problem!



Objective

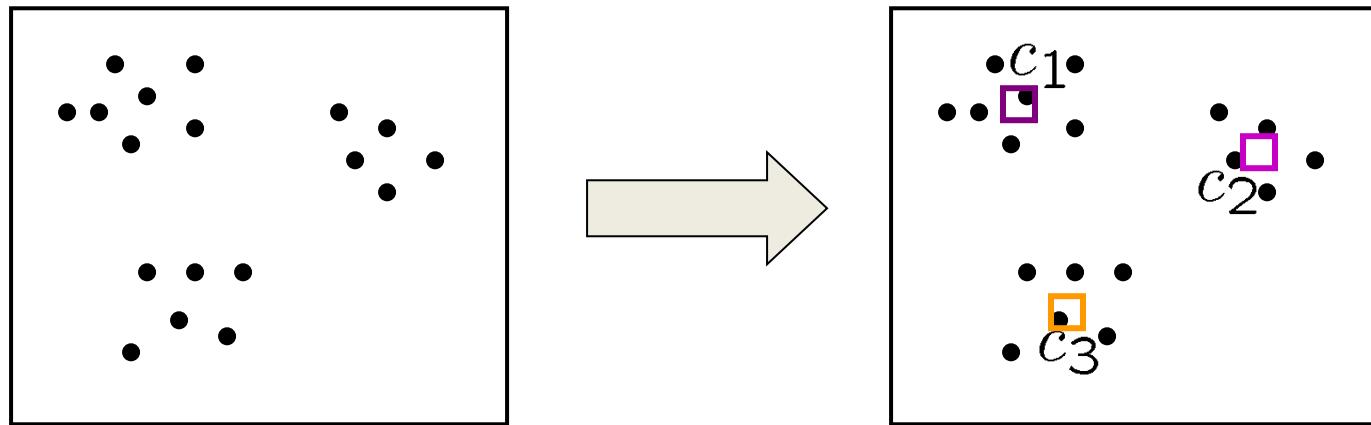
- Each point should be as close as possible to a cluster center
 - Minimize sum squared distance of each point to closest center

$$\sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2$$

Break it down into subproblems

1) Suppose I tell you the cluster centers c_i

- Q: how to determine which points to associate with each c_i ?
- A: for each point p , choose closest c_i



2) Suppose I tell you the points in each cluster

- Q: how to determine the cluster centers?
- A: choose c_i to be the mean of all points in the cluster

K-means clustering

K-means clustering algorithm

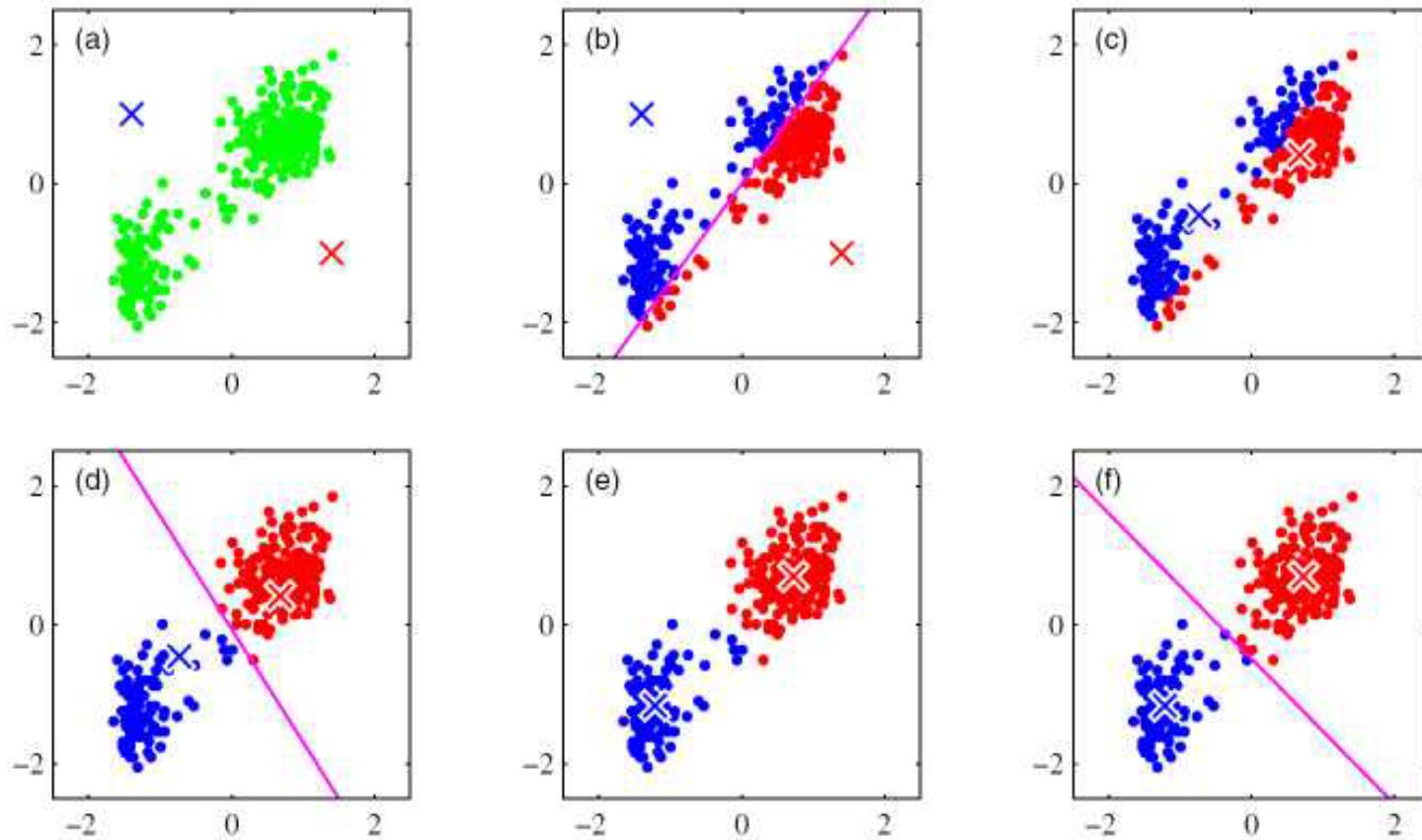
1. Randomly initialize the cluster centers, c_1, \dots, c_K
2. Given cluster centers, determine points in each cluster
 - For each point p , find the closest c_i . Put p into cluster i
3. Given points in each cluster, solve for c_i
 - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2

Properties

- Probability density : superposition of spherically symmetric distributions
- Will always converge to *some* solution
- Can be a “local minimum”
 - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2$$

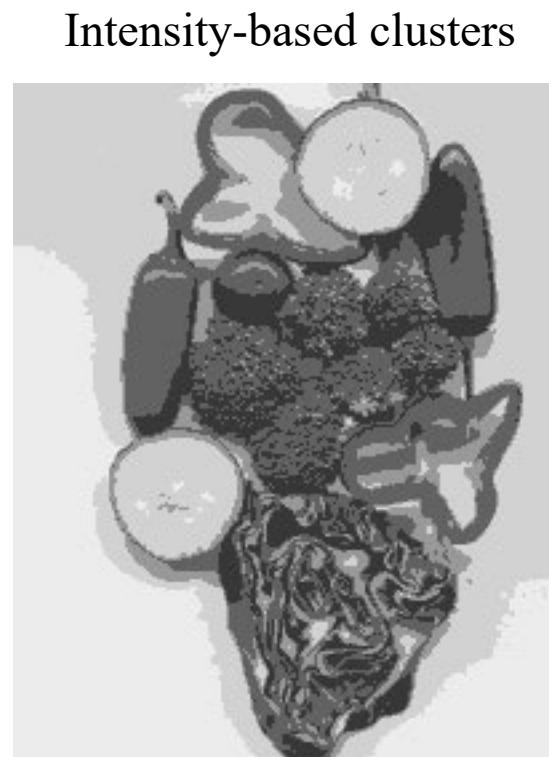
K-means Clustering



Szeliski Figure 5.16 The k-means algorithm starts with a set of samples and the number of desired clusters (in this case, $k = 2$) (Bishop 2006) © 2006 Springer. It iteratively assigns samples to the nearest mean, and then re-computes the mean center until convergence.

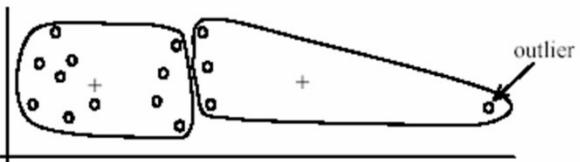
K-Means clustering

- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
 - Clusters don't have to be spatially coherent



K-Means pros and cons

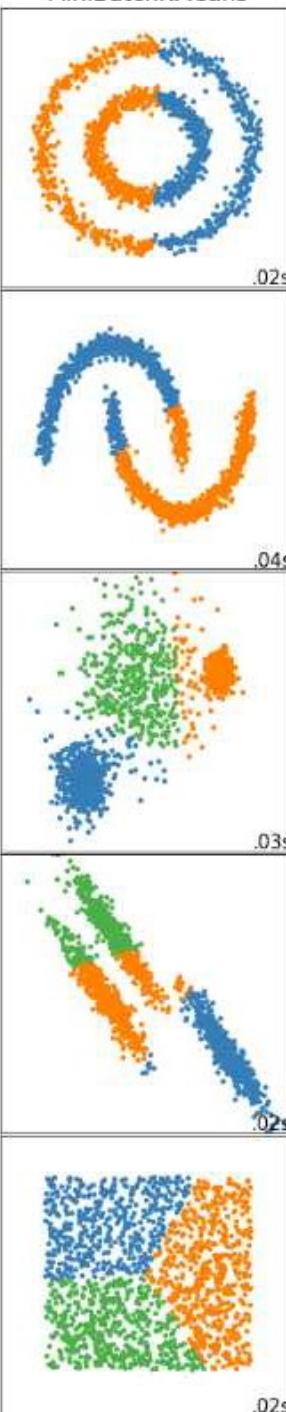
- Pros
 - Simple and fast
 - Converges to a local minimum of the error function
- Cons
 - Need to pick K
 - Sensitive to initialization
 - Sensitive to outliers
 - Only finds “spherical” clusters



(A): Undesirable clusters



(B): Ideal clusters



Some Improvements

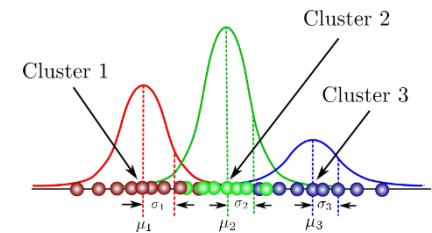
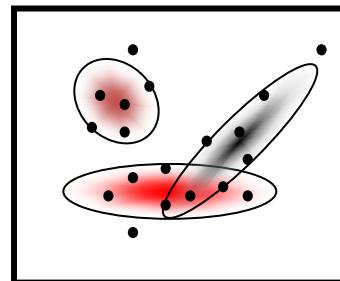
- Split / Merge cluster centers based on their statistics
- Better center initialization

K-Means++

- Can we prevent arbitrarily bad local minima?
- 1. Randomly choose first center.
- 2. Pick new center with prob. proportional to: $\|p - c_i\|^2$
(contribution of p to total error)
- 3. Repeat until k centers.
- expected error = $O(\log k) * \text{optimal}$
- [Arthur & Vassilvitskii 2007](#)

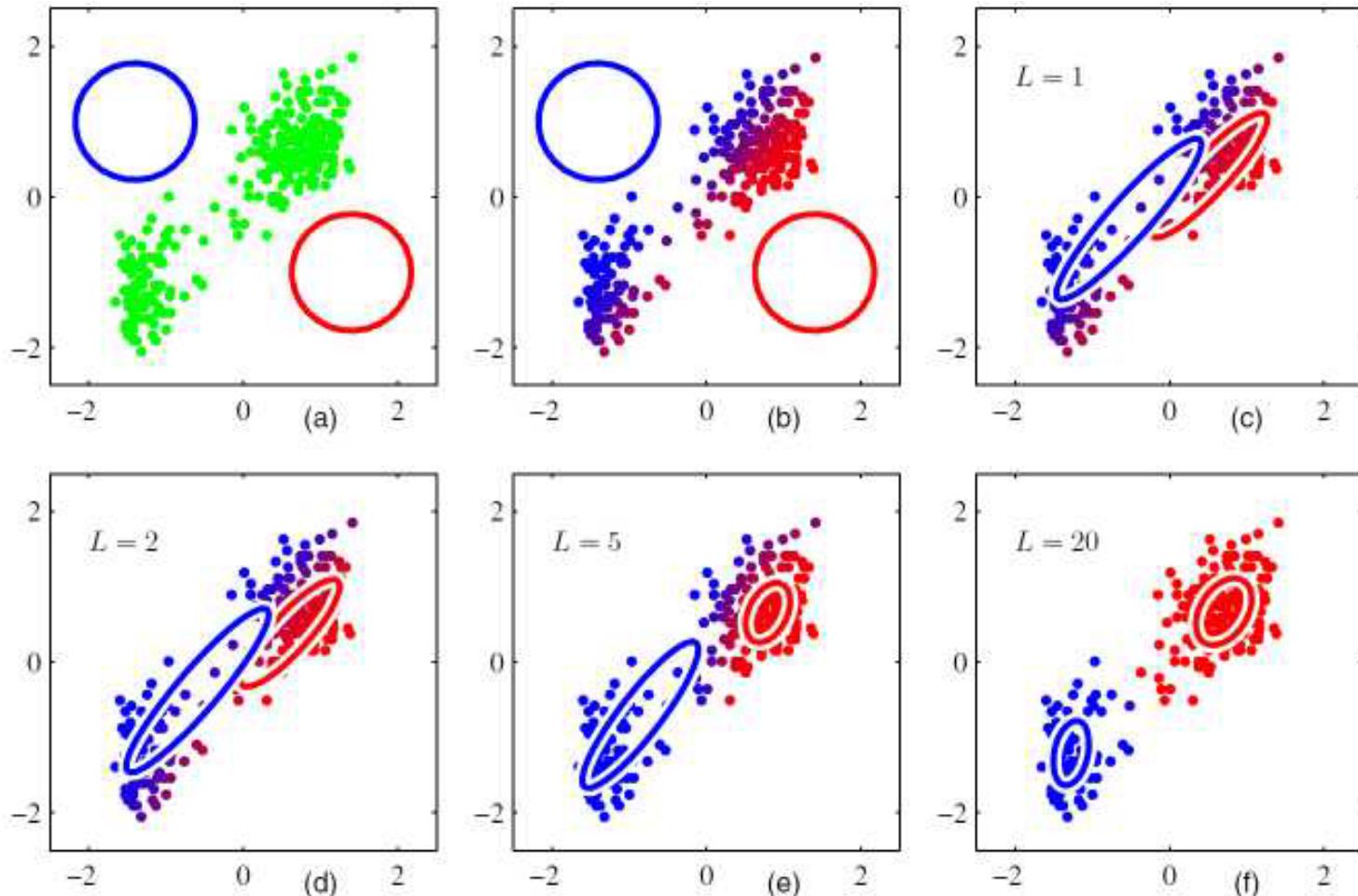
Mixture of Gaussians

K-Means	Mixture of Gaussians
Samples from Probability Density Function	Samples from Probability Density Function
Parametric model	Parametric model
Spherically symmetric distributions	Center + shape (covariance matrix)
Point association: Nearest-neighbor	Point association: Mahalanobis distance (center + shape (covariance))



Mahalanobis distance: $d(x_i, \mu_k; \Sigma_k) = (x_i - \mu_k)^T \Sigma_k^{-1} (x_i - \mu_k)$

Gaussian Mixture Modeling



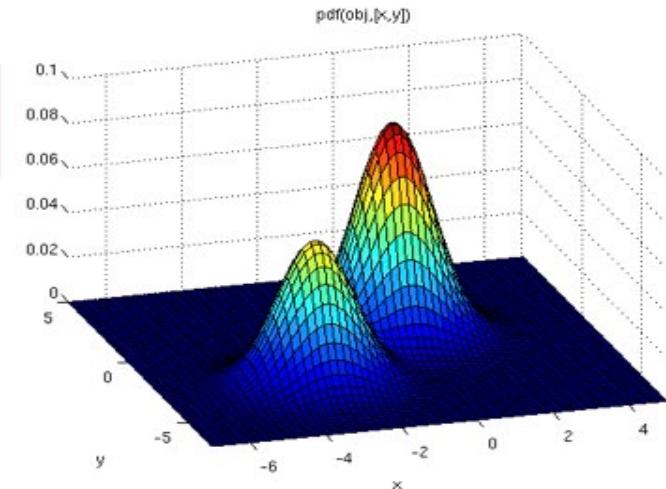
Szeliski Figure 5.17 Gaussian mixture modeling (GMM) using expectation maximization (EM) (Bishop 2006) © 2006 Springer. Samples are softly assigned to cluster centers based on their Mahalanobis distance (inverse covariance weighted distance), and the new means and covariances are recomputed based on these weighted assignments.

Mixture of Gaussians --- GMM or MoG

Represent data distribution as “mixture” of multivariate Gaussians:

$$p(\mathbf{x}|\{\pi_k, \mu_k, \Sigma_k\}) = \sum_k \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)$$

K components
Mixing coefficients Gaussian means and covariances



Normal (Gaussian) distribution:

$$\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) = \frac{1}{|\Sigma_k|} e^{-d(\mathbf{x}, \mu_k; \Sigma_k)}$$

How do we actually fit this distribution?

Expectation Maximization (EM) algorithm

Goal: Iteratively compute (a local) maximum likely estimate for the unknown mixture parameters $\{\pi_k, \mu_k, \Sigma_k\}$

1. **Expectation Stage (E step):** Estimate “ownership” probability of each point belonging to each component (how likely a sample x_i was generated from the k^{th} Gaussian cluster)

$$z_{ik} = \frac{1}{Z_i} \pi_k \mathcal{N}(x | \mu_k, \Sigma_k) \quad \text{with} \quad \sum_k z_{ik} = 1,$$

2. **Maximization stage (M step):** given ownership probabilities update parameter values

$$\begin{aligned}\mu_k &= \frac{1}{N_k} \sum_i z_{ik} x_i, \\ \Sigma_k &= \frac{1}{N_k} \sum_i z_{ik} (x_i - \mu_k)(x_i - \mu_k)^T, \\ \pi_k &= \frac{N_k}{N},\end{aligned}$$

$$N_k = \sum_i z_{ik}.$$

Estimate of number of sample points assigned to each cluster.

N: number of sample points

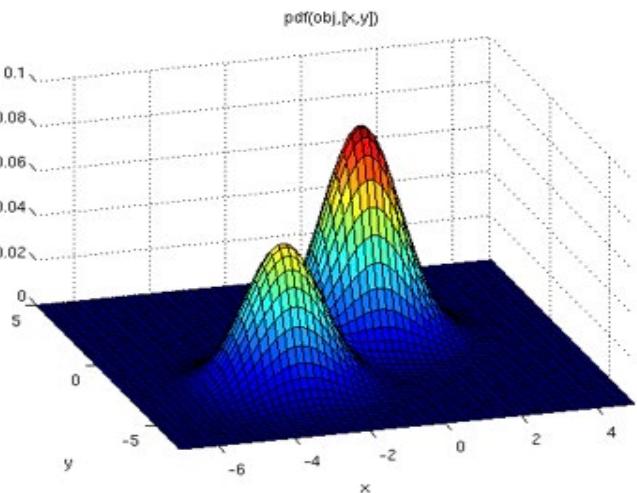
EM: “soft” version of k -means

Given k :

1. Select initial centroids (c_1, \dots, c_k) at random.

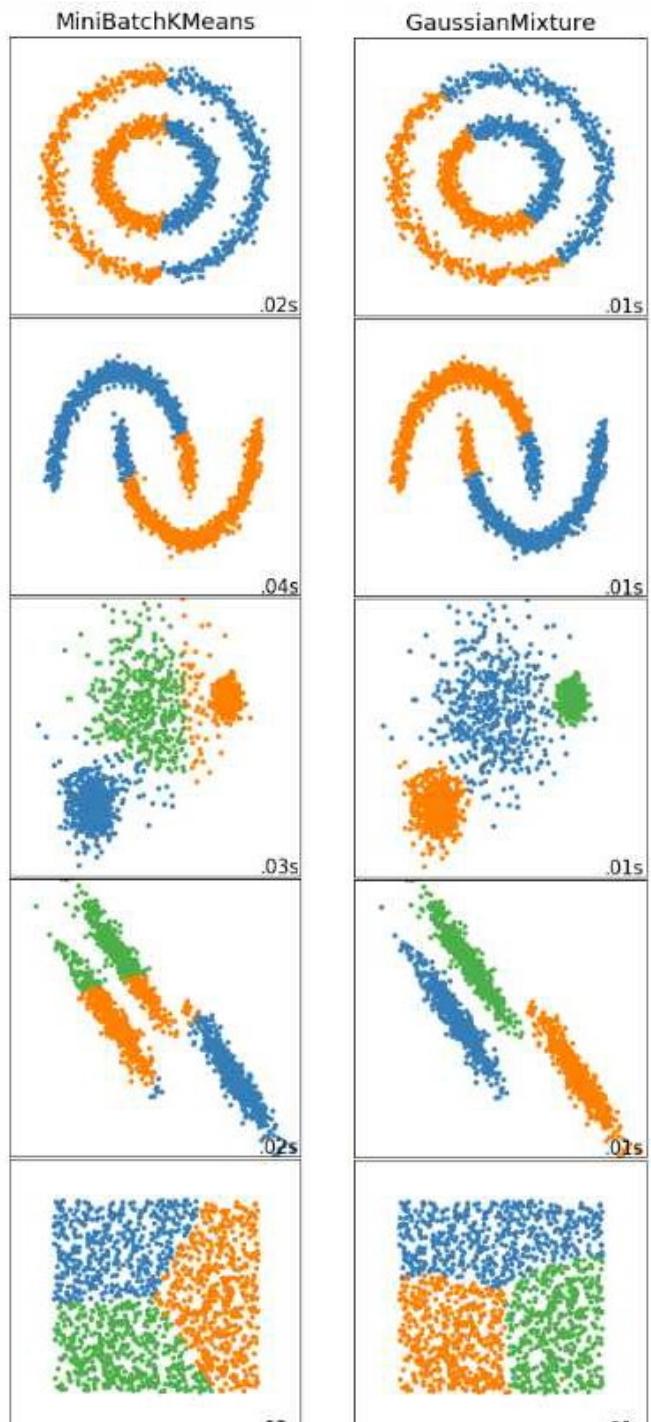
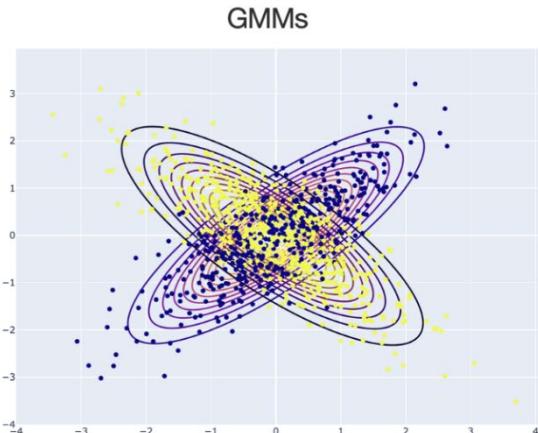
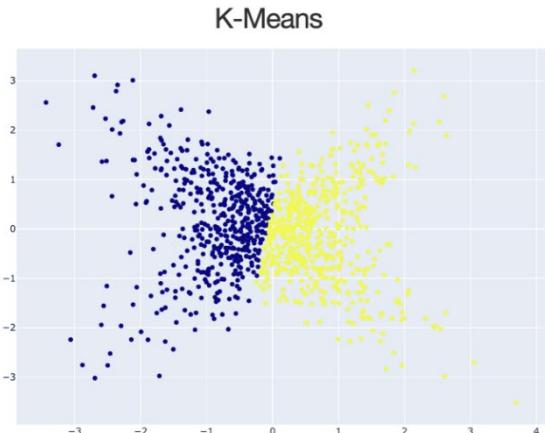
E-step ~~2. Assign each point to nearest centroid.~~ compute the probability of each object being in cluster
and covariance

M-step ~~3. Compute each centroid's new location as the mean of the locations of points assigned to it.~~ weighed by the probability of being in that cluster.
4. Repeat from 2. until no change.



k -means vs GMMs

k -means is a special case of GMMs.



<https://towardsdatascience.com/expectation-maximization-for-gmm-explained-5636161577ca>

Applications of EM

- Turns out this is useful for all sorts of problems
 - any clustering problem
 - any model estimation problem
 - missing data problems
 - finding outliers
 - segmentation problems
 - segmentation based on color
 - segmentation based on motion
 - foreground/background separation
 - ...

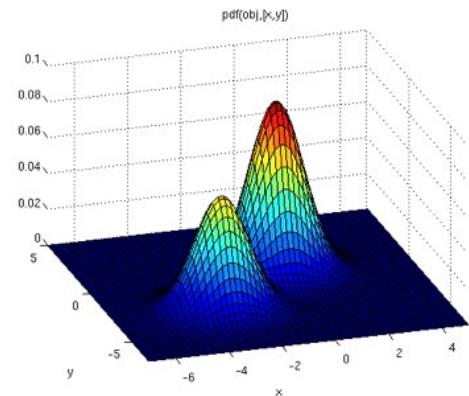
GMMs

Pros

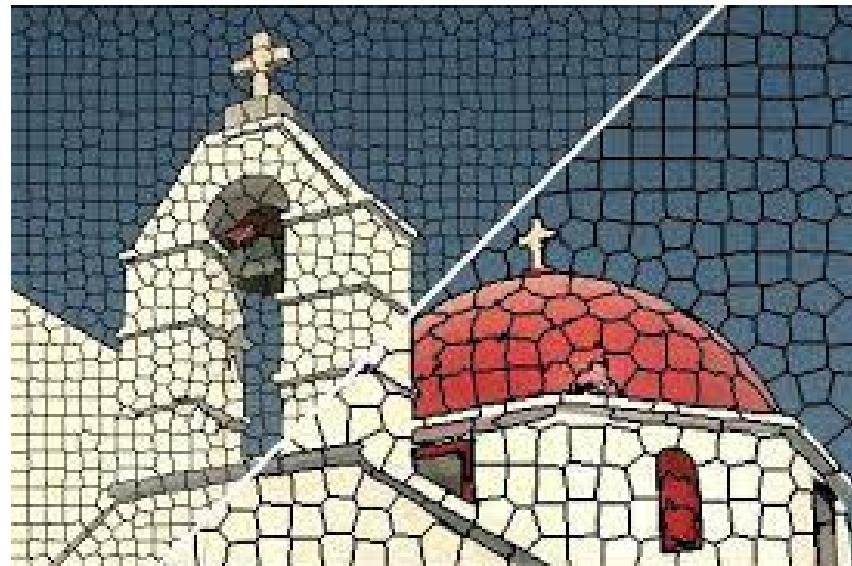
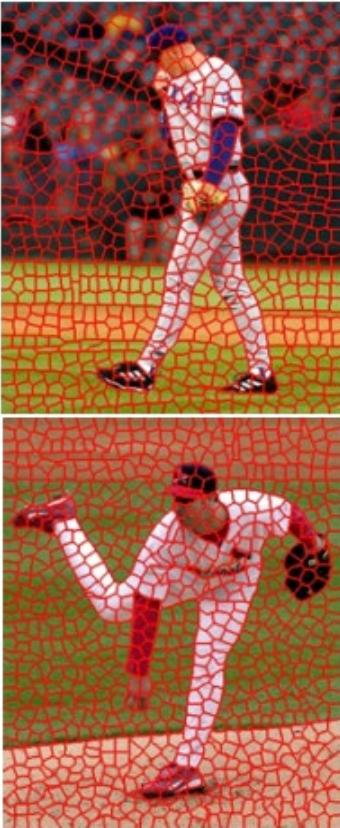
- Still fairly simple and efficient
- Model more complex distributions

Cons

- Need to set K



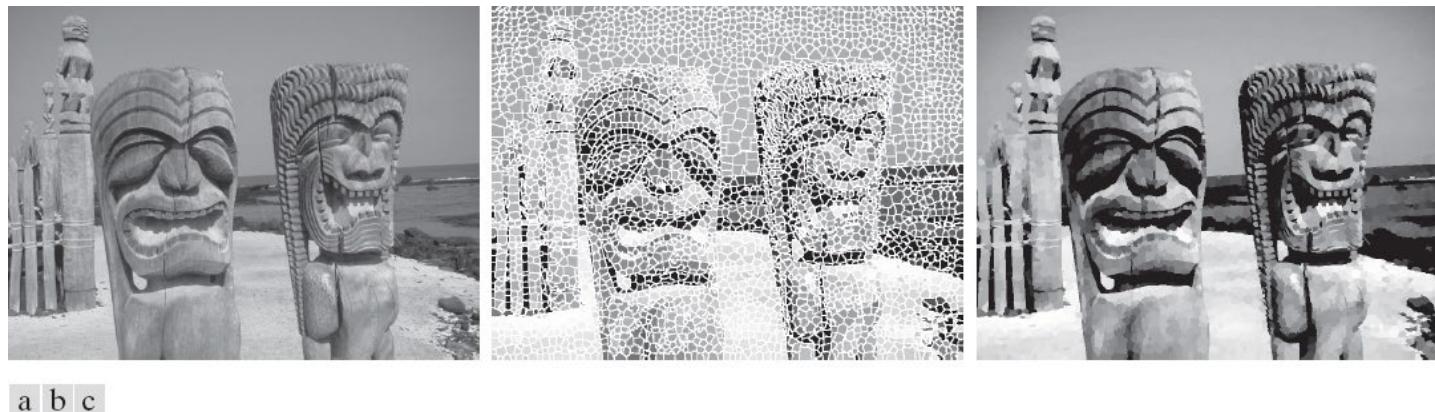
As k increases → “superpixels”



Why may this representation be useful?

Region Segmentation using Superpixels

- The idea behind *superpixels* is to replace the standard pixel grid by grouping pixels into primitive regions that are more perceptually meaningful than individual pixels.
- The objectives are to
 - lessen computational load,
 - improve the performance of segmentation algorithms by reducing irrelevant detail.
- One important requirement of any superpixel representation is adherence to boundaries. This means that boundaries between regions of interest must be preserved in a superpixel image.



Gonzalez & Woods Figure 10.50

- (a) Image of size 600x480 (**480,000**) pixels.
- (b) Image composed of **4,000** superpixels (the boundaries between superpixels (in white) are superimposed on the superpixel image for reference—the boundaries are not part of the data).
- (c) Superpixel image.

Gonzalez & Woods Figure 10.51

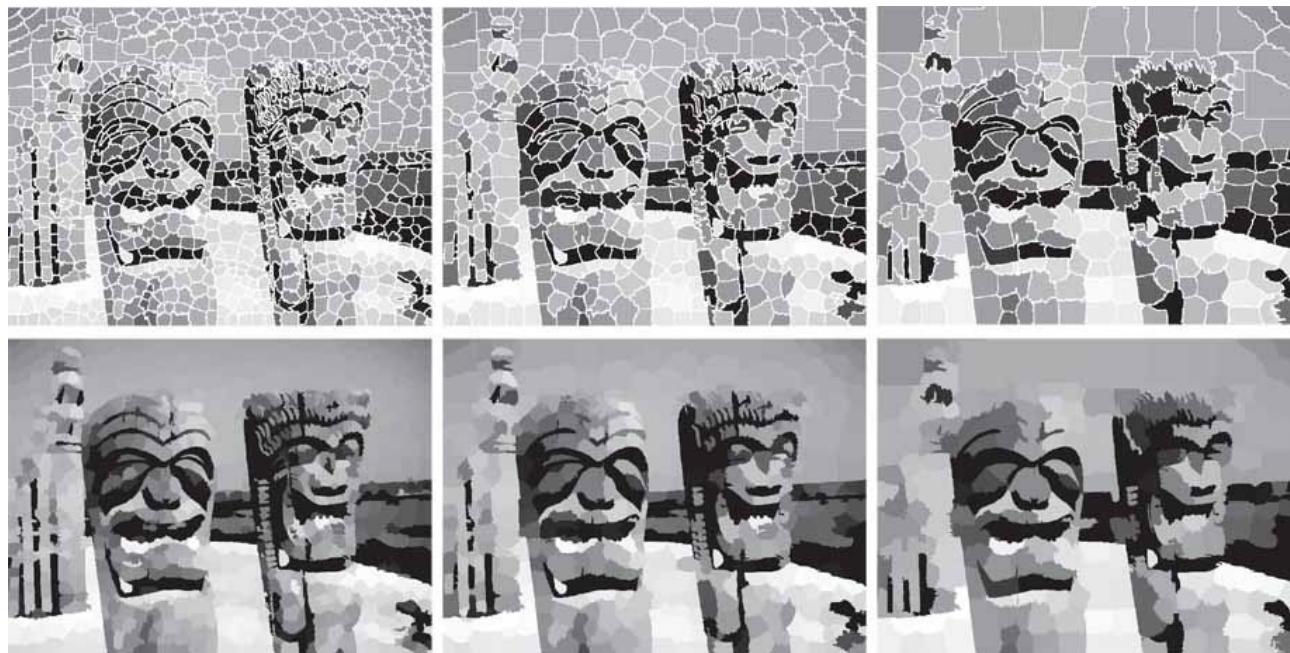
(a) Original image. (b) Image composed of 40,000 superpixels. (c) Difference between (a) and (b).



a | b | c

Gonzalez & Woods Figure 10.52

Top row: Results of using 1,000, 500, and 250 superpixels in the representation of Fig. 10.50(a). As before, the boundaries between superpixels are superimposed on the images for reference. Bottom row: Superpixel images.



SLIC (Simple Linear Iterative Clustering) Superpixel Algorithm

- Developed by **Achanta et al. [2012]**
- SLIC is a modification of the k -means algorithm.
- SLIC observations typically use (but are not limited to) 5-dimensional vectors containing three color components and two spatial coordinates i.e. $\mathbf{z} = [r, g, b, x, y]$

$$\mathbf{z} = \begin{bmatrix} r \\ g \\ b \\ x \\ y \end{bmatrix}$$

n_{sp} denote the desired number of superpixels
 n_{tp} denote the total number of pixels in the image.
initial superpixel centers, $\mathbf{m}_i = [r_i \ g_i \ b_i \ x_i \ y_i]^T$,
 $i = 1, 2, \dots, n_{sp}$, are obtained by sampling the image on a regular grid space
 $s = [n_{tp}/n_{sp}]^{1/2}$

1. **Initialize the algorithm:** Compute the initial superpixel cluster centers,

$$\mathbf{m}_i = [r_i \ g_i \ b_i \ x_i \ y_i]^T, i = 1, 2, \dots, n_{sp}$$

by sampling the image at regular grid steps, s . Move the cluster centers to the lowest gradient position in a 3×3 neighborhood.

For each pixel location, p , in the image, set a label $L(p) = -1$ and a distance $d(p) = \infty$.

2. **Assign samples to cluster centers:** For each cluster center \mathbf{m}_i , $i = 1, 2, \dots, n_{sp}$, compute the distance, $D_i(p)$ between \mathbf{m}_i and each pixel p in a $2s \times 2s$ neighborhood about \mathbf{m}_i . Then, for each p and $i = 1, 2, \dots, n_{sp}$, if $D_i < d(p)$, let $d(p) = D_i$ and $L(p) = i$.

3. **Update the cluster centers:** Let C_i denote the set of pixels in the image with label $L(p) = i$. Update \mathbf{m}_i :

$$\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \quad i = 1, 2, \dots, n_{sp}$$

where $|C_i|$ is the number of pixels in set C_i , and the \mathbf{z} 's are given by [Eq. \(10-86\)](#).

4. **Test for convergence:** Compute the Euclidean norms of the differences between the mean vectors in the current and previous steps. Compute the residual error, E , as the sum of the n_{sp} norms. If $E < T$, where T a specified nonnegative threshold, go to Step 5. Else, go back to Step 2.

5. **Post-process the superpixel regions:** Replace all the superpixels in each region, C_i , by their average value, \mathbf{m}_i .

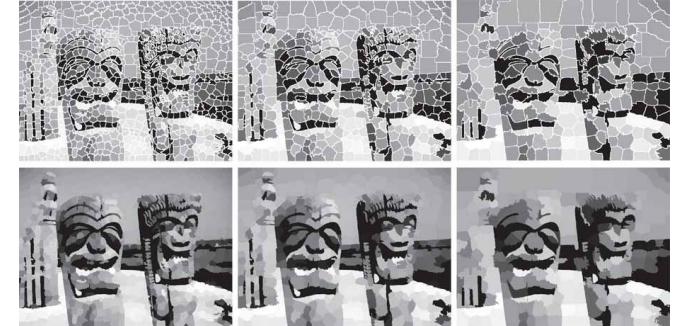
SLIC Distance measure

- Euclidean distances in CIELAB color space for more perceptually meaningful distances

$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

$$D_s = d_{lab} + \frac{m}{C} d_{xy} ,$$

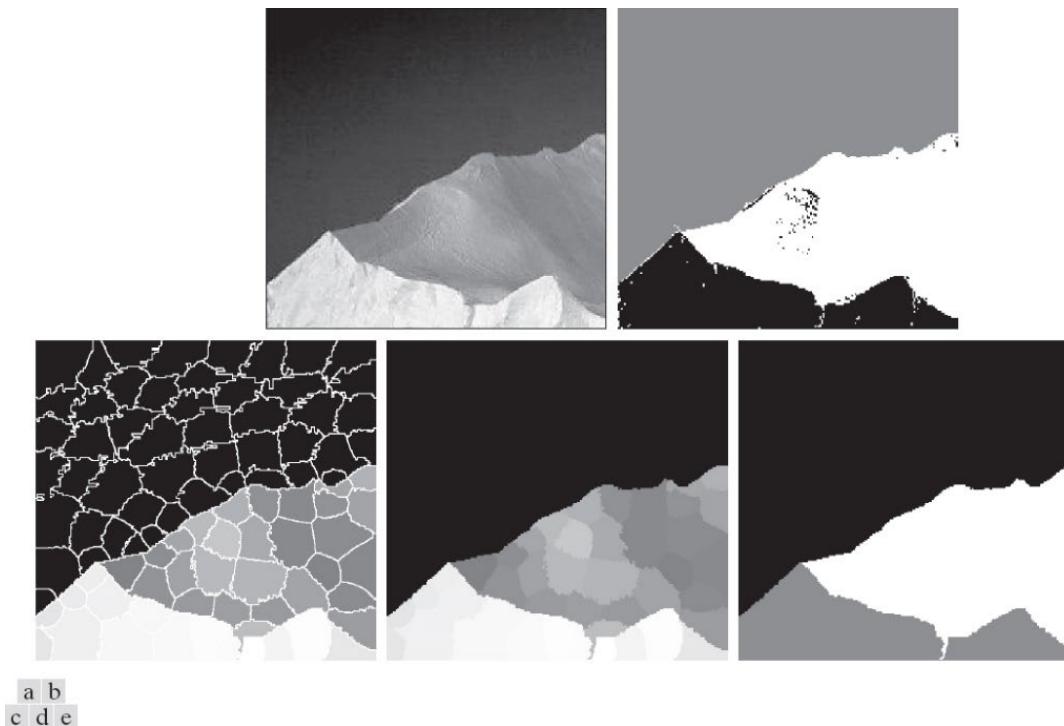


Algorithm 1 Efficient superpixel segmentation

- 1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S .
 - 2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
 - 3: **repeat**
 - 4: **for** each cluster center C_k **do**
 - 5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (Eq. 1).
 - 6: **end for**
 - 7: Compute new cluster centers and residual error E { $L1$ distance between previous centers and recomputed centers}
 - 8: **until** $E \leq \text{threshold}$
 - 9: Enforce connectivity.
-

Gonzalez & Woods Figure 10.53

- (a) Image of size 533x566 (301,678) pixels.
- (b) Image segmented using the *k*-means algorithm.
- (c) 100-element superpixel image showing boundaries for reference.
- (d) Same image without boundaries.
- (e) Superpixel image (d) segmented using the *k*-means algorithm. (Original image courtesy of NOAA.)



SLIC Examples

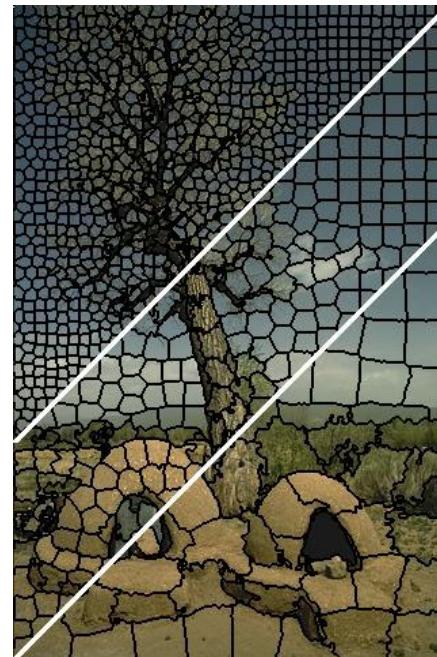
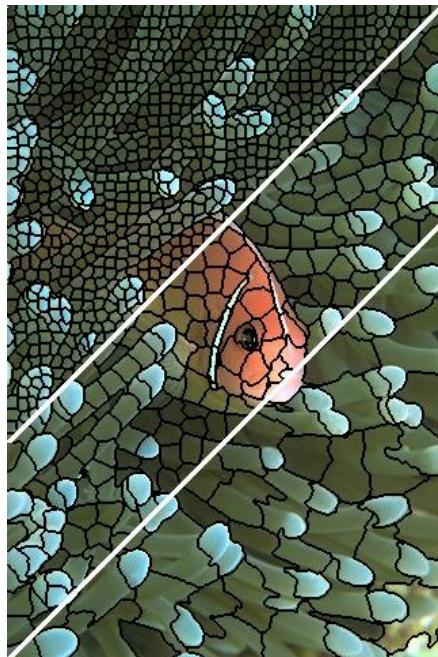
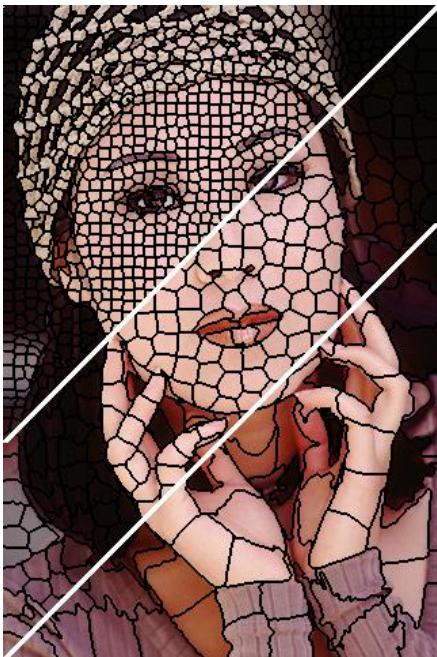
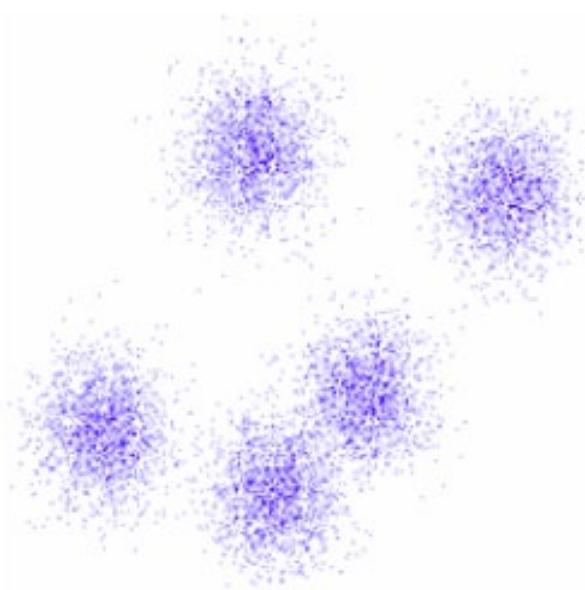


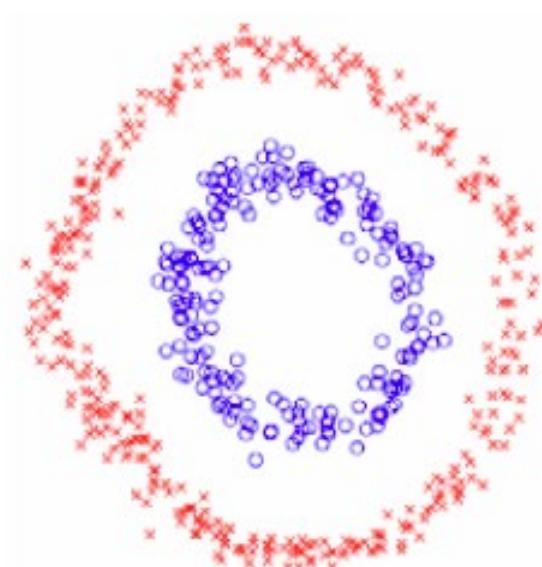
Image segmented using our algorithm into superpixels of (approximate) size 64, 256, and 1024 pixels. The superpixels are compact, uniform in size, and adhere well to region boundaries.

Clustering

- Two different criteria
 - Compactness, e.g., k-means, mixture models
 - Connectivity, e.g., spectral clustering



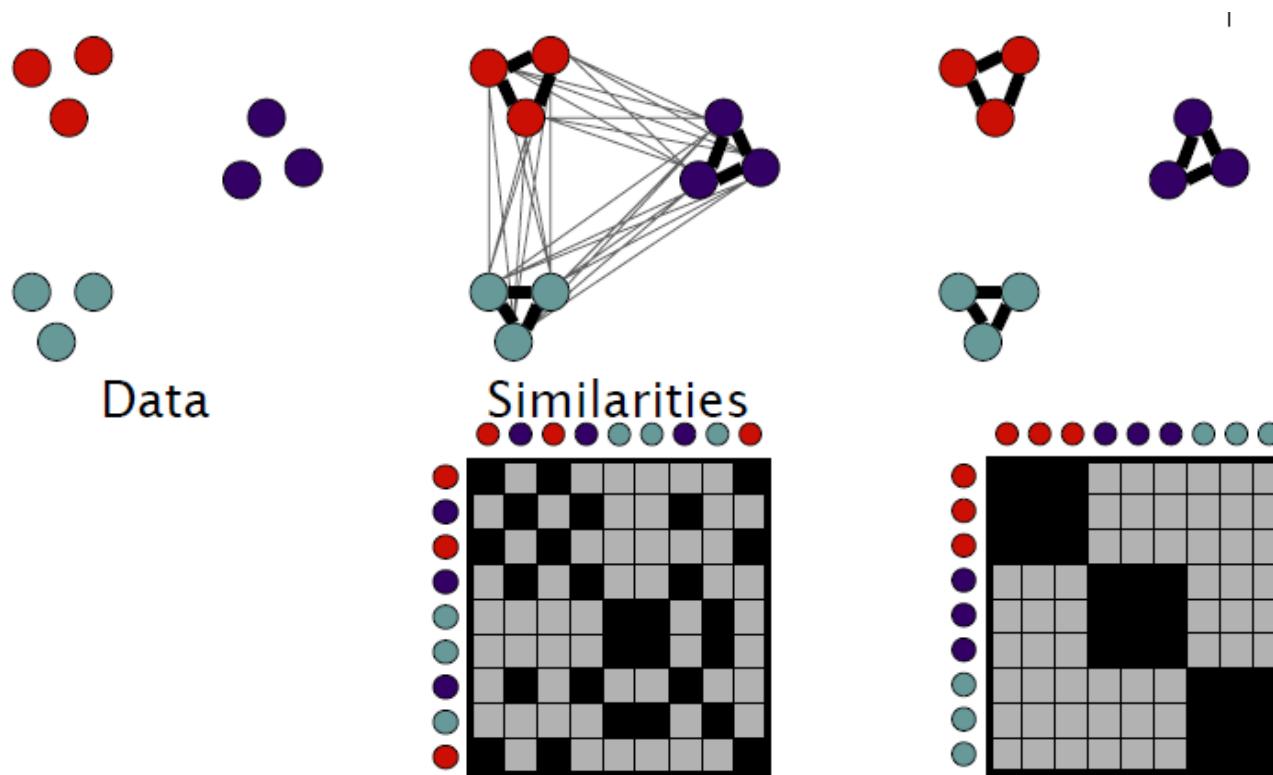
Compactness



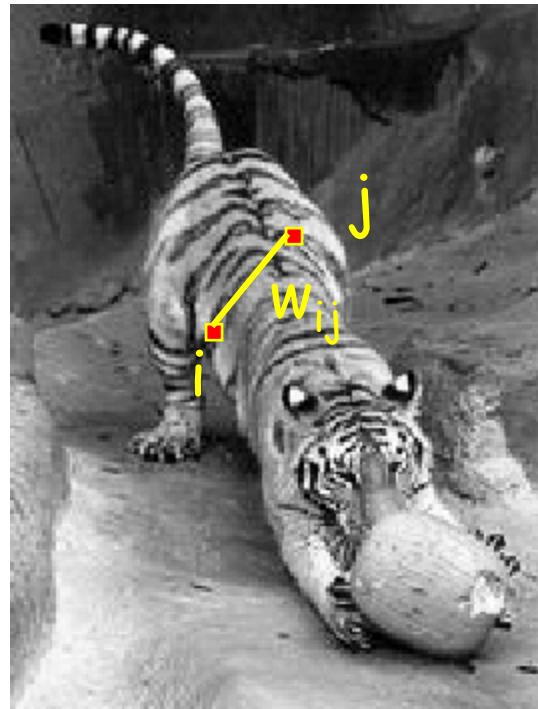
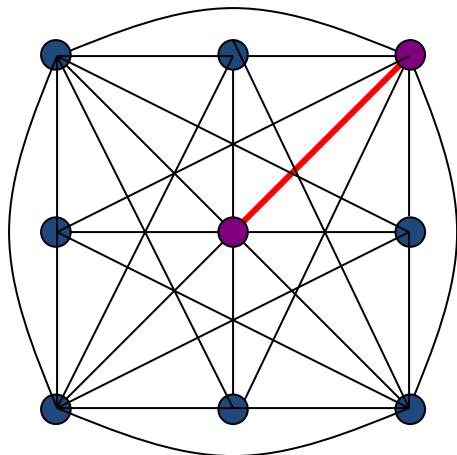
Connectivity



Graph Clustering



Images as graphs



1. Node for every pixel
2. Edge between every pair of pixels (p,q)
3. Each edge is weighted by the *affinity* or similarity of the two nodes (w_{pq})

Which edges to include?

Fully connected

- Captures all pairwise similarity
- Infeasible for most images

Neighboring pixels

- Very fast to compute
- Only captures very local interactions

Local neighborhood

- Reasonably fast, graph still very sparse
- Good tradeoff



Measuring affinity

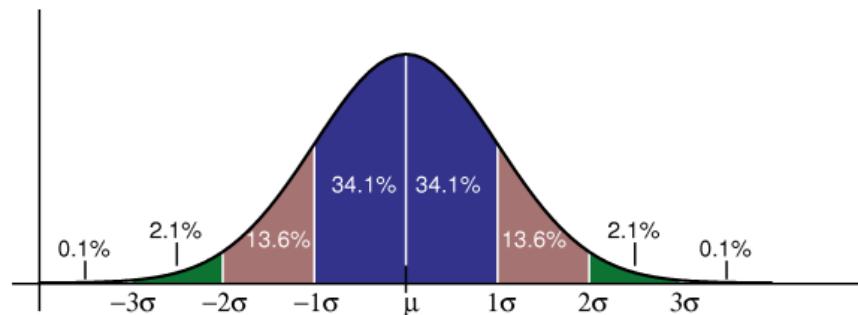
Suppose

- we represent each pixel by a feature vector \mathbf{x} ,
- we define a distance function appropriate for this feature representation

Then

- we can convert the distance between two feature vectors into an affinity (similarity) with the help of a generalized Gaussian kernel:

$$\exp\left(-\frac{1}{2\sigma^2} \text{dist}(\mathbf{x}_i, \mathbf{x}_j)^2\right)$$



Measuring Affinity

In general: $aff(x, y) = \exp\left(-\frac{1}{2\sigma_d^2}\|f(x) - f(y)\|^2\right)$

Examples:

- **Distance:** $f(x) = location(x)$
- **Intensity:** $f(x) = intensity(x)$
- **Color:** $f(x) = color(x)$
- **Texture:** $f(x) = filterbank(x)$

- Note: Can also modify distance metric

slide credit: Forsyth & Ponce

Measuring Affinity

Distance

$$aff(x, y) = \exp\left\{-\left(\frac{1}{2\sigma_d^2}\right)\left(\|x - y\|^2\right)\right\}$$

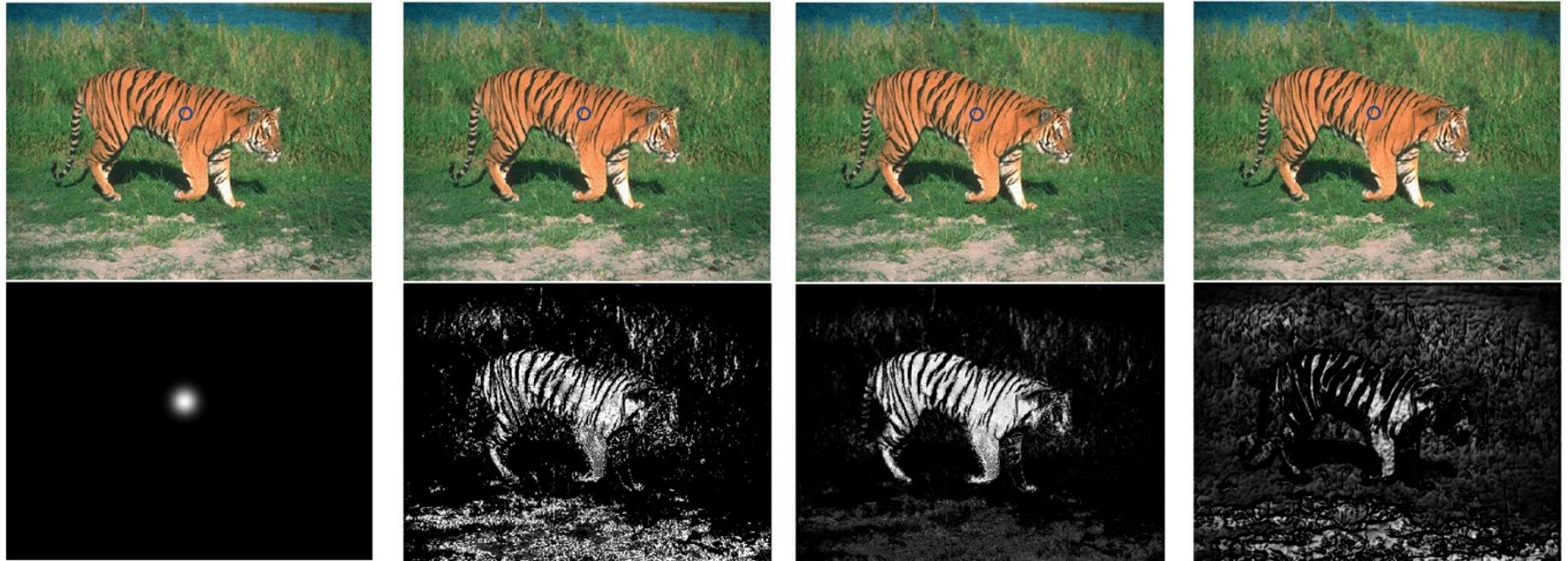
Intensity

$$aff(x, y) = \exp\left\{-\left(\frac{1}{2\sigma_i^2}\right)\left(\|I(x) - I(y)\|^2\right)\right\}$$

Color

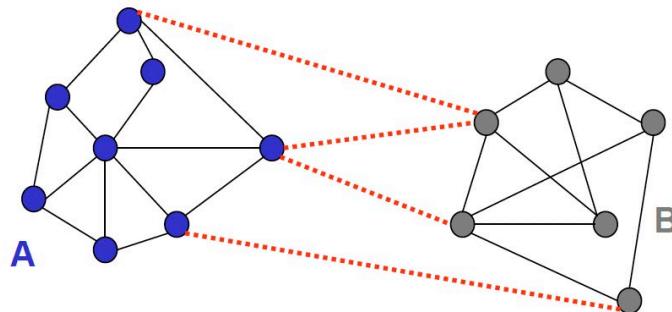
$$aff(x, y) = \exp\left\{-\left(\frac{1}{2\sigma_t^2}\right)\left(\|c(x) - c(y)\|^2\right)\right\}$$

Measuring Affinity



$f(x) = \text{location}(x)$ $f(x) = \text{intensity}(x)$ $f(x) = \text{color}(x)$ $f(x) = \text{filterbank}(x)$
(Texture)

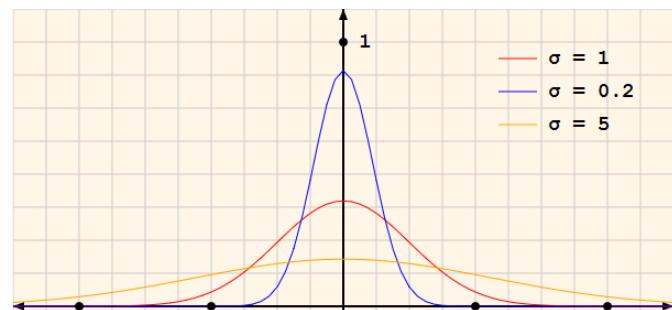
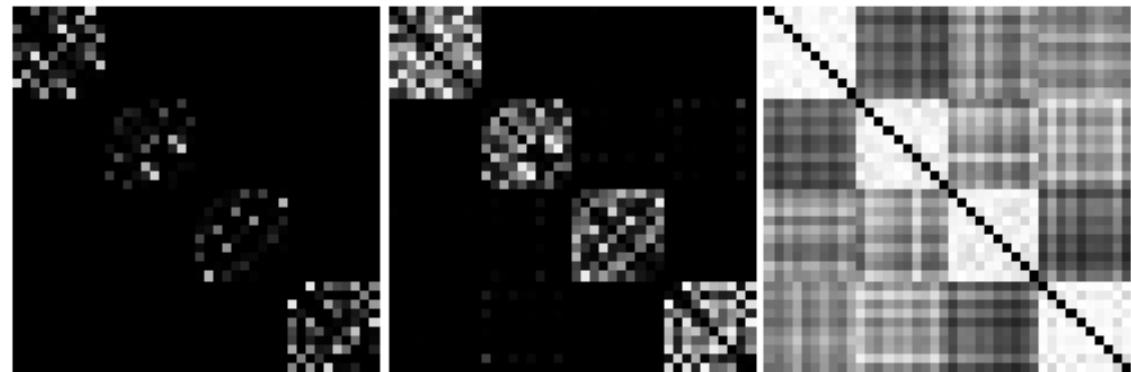
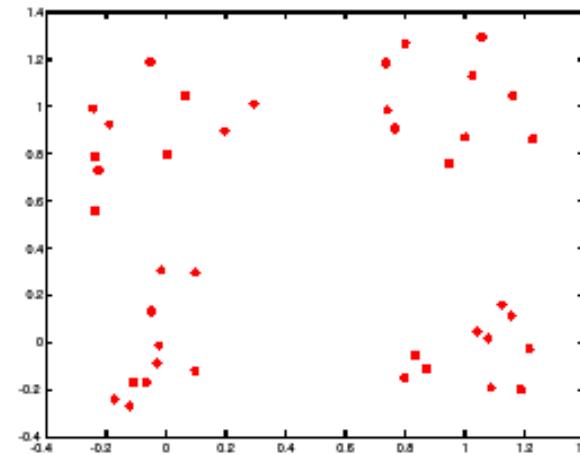
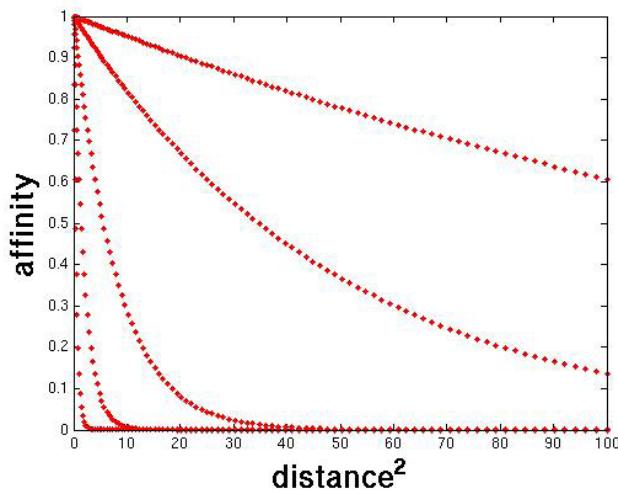
Segmentation by graph partitioning



- Break graph into segments
- Delete links that cross between segments
 - Similar pixels should be in the same segments
 - Dissimilar pixels should be in different segments
 - Easiest to break links that have low similarity (low weight)

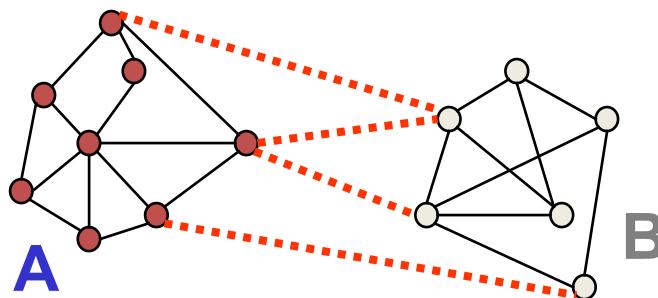
Scale affects affinity

- Small σ : group only nearby points



Slide credit: S. Lazebnik

Graph cut

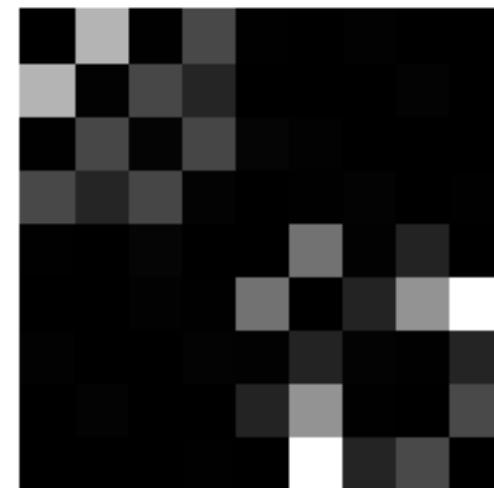
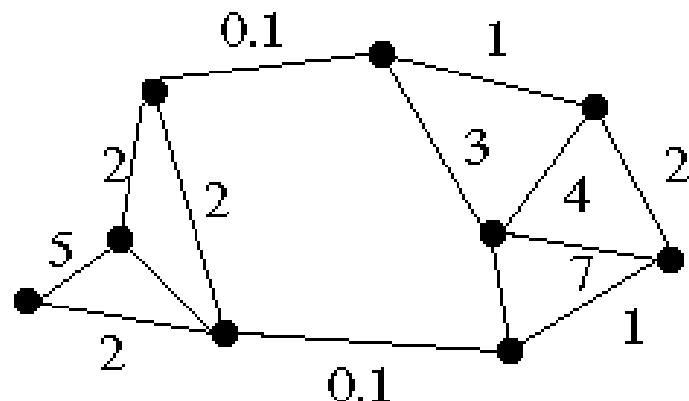


- Set of edges whose removal makes a graph disconnected
- **Cost of a cut:** sum of weights of cut edges
- A graph cut gives us a segmentation
 - What is a “good” graph cut and how do we find one?

Minimum cut (Min Cut)

- We can do segmentation by finding the *minimum cut* in a graph
 - Efficient algorithms exist for doing this

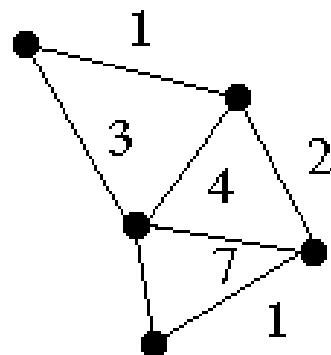
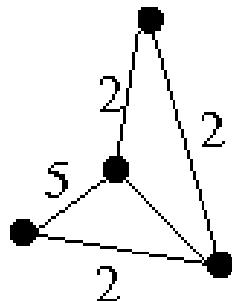
Minimum cut example



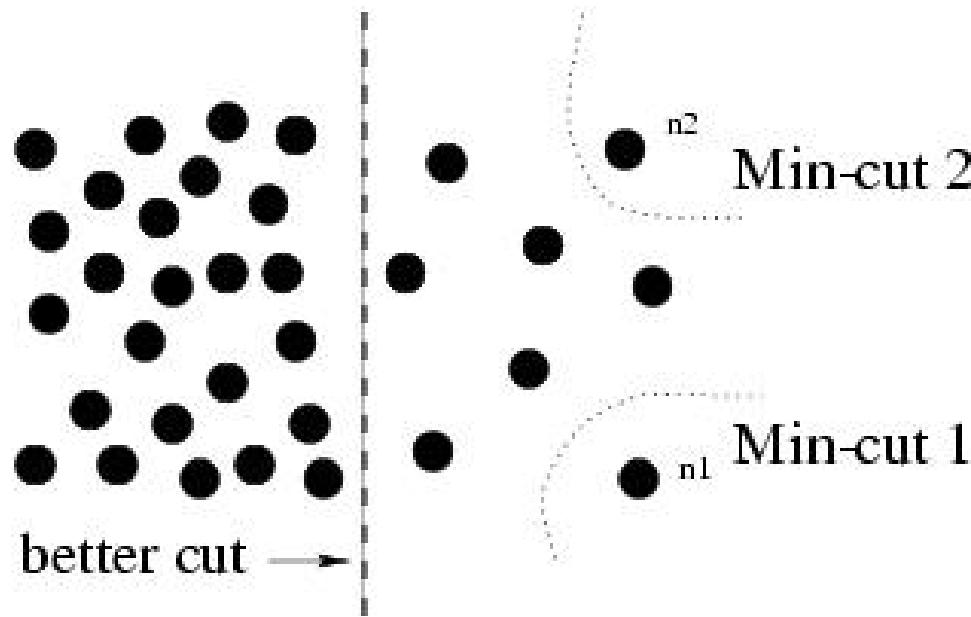
Minimum cut

- We can do segmentation by finding the *minimum cut* in a graph
 - Efficient algorithms exist for doing this

Minimum cut example



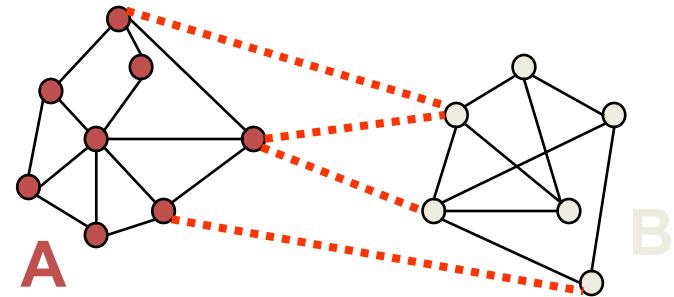
But min cut is not always the best cut...



Drawback:

- Weight of cut proportional to number of edges
- Biased towards cutting small, isolated components

Normalized Cut

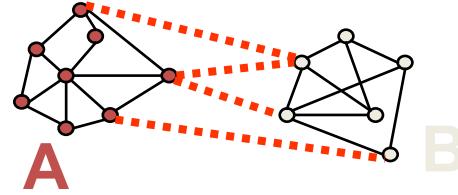


Min-Cut Drawback: tends to cut off very small, isolated components

Normalized Cut

- penalizes large segments
 - fix by normalizing for size of segments
-
- $\text{cut}(A, B) = \text{sum of weights of all edges between } A \text{ and } B$
 - $\text{volume}(A) = \text{sum of costs of all edges that touch } A$

$$Ncut(A, B) = \frac{\text{cut}(A, B)}{\text{volume}(A)} + \frac{\text{cut}(A, B)}{\text{volume}(B)}$$



Normalized cut

- Computing the optimal normalized cut is NP-complete.
- Shi & Malik suggest a real valued assignment of nodes to groups.
- Let X be an indicator vector

$$x_i = \begin{cases} +1 & \text{iff } i \in A \\ -1 & \text{iff } i \in B \end{cases}$$

- W : adjacency matrix of the graph
- D : diagonal matrix with diagonal entries $D(i, i) = \sum_j W(i, j)$
- Normalized cut cost can be written as

$$\frac{y^T (D - W) y}{y^T D y}$$

- y : indicator vector
 - value should be 1 in the i th position if the i th feature point belongs to A
 - a negative constant otherwise

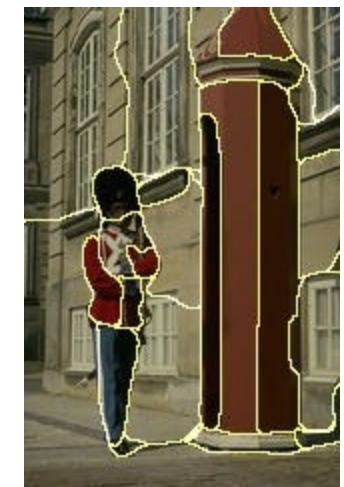
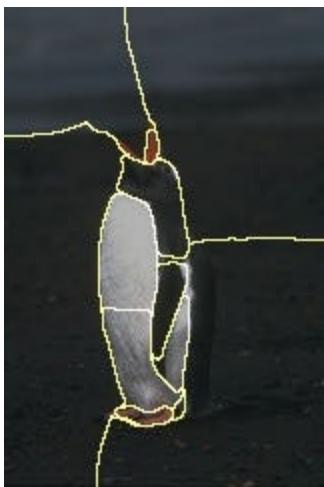
Recursive normalized cuts

1. Given an image or image sequence, set up a weighted graph: $G=(V, E)$
 - Vertex for each pixel
 - Edge weight for nearby pairs of pixels
2. Solve for eigenvectors with the smallest eigenvalues:
$$(D - W)y = \lambda Dy$$
 - Use the eigenvector with the second smallest eigenvalue to bipartition the graph
 - Note: this is an approximation
4. Recursively repartition the segmented parts if necessary

Example results



Results: Berkeley Segmentation Engine



<http://www.cs.berkeley.edu/~fowlkes/BSE/>

Normalized cuts: Pro and con

- Pros
 - Generic framework, can be used with many different features and affinity formulations
- Cons
 - High storage requirement and time complexity
 - Bias towards partitioning into equal segments

Spectral clustering - main algorithms

In [multivariate statistics](#), **spectral clustering** techniques make use of the [spectrum \(eigenvalues\)](#) of the [similarity matrix](#) of the data to perform [dimensionality reduction](#) before clustering in fewer dimensions.

The general approach: to spectral clustering is to use a standard clustering method (such as k-means) on relevant eigenvectors of a Laplacian matrix of A.

A: Affinity matrix

- Build the matrix $V \in \mathbb{R}^{n \times k}$ with the eigenvectors as columns
- Interpret the rows of V as new data points $Z_i \in \mathbb{R}^k$

	v_1	v_2	v_3
Z_1	v_{11}	v_{12}	v_{13}
\vdots	\vdots	\vdots	\vdots
Z_n	v_{n1}	v_{n2}	v_{n3}

- Cluster the points Z_i with the k -means algorithm in \mathbb{R}^k .

Spectral Clustering

Laplacian matrix [edit]

Given a simple graph G with n vertices v_1, \dots, v_n , its Laplacian matrix $L_{n \times n}$ is defined element-wise as^[1]

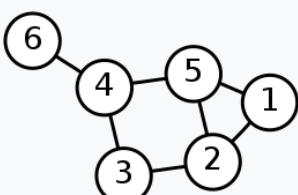
$$L_{i,j} := \begin{cases} \deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise,} \end{cases}$$

or equivalently by the matrix

$$L = D - A,$$

where D is the degree matrix and A is the adjacency matrix of the graph. Since G is a simple graph, A only contains 1s or 0s and its diagonal elements are all 0s.

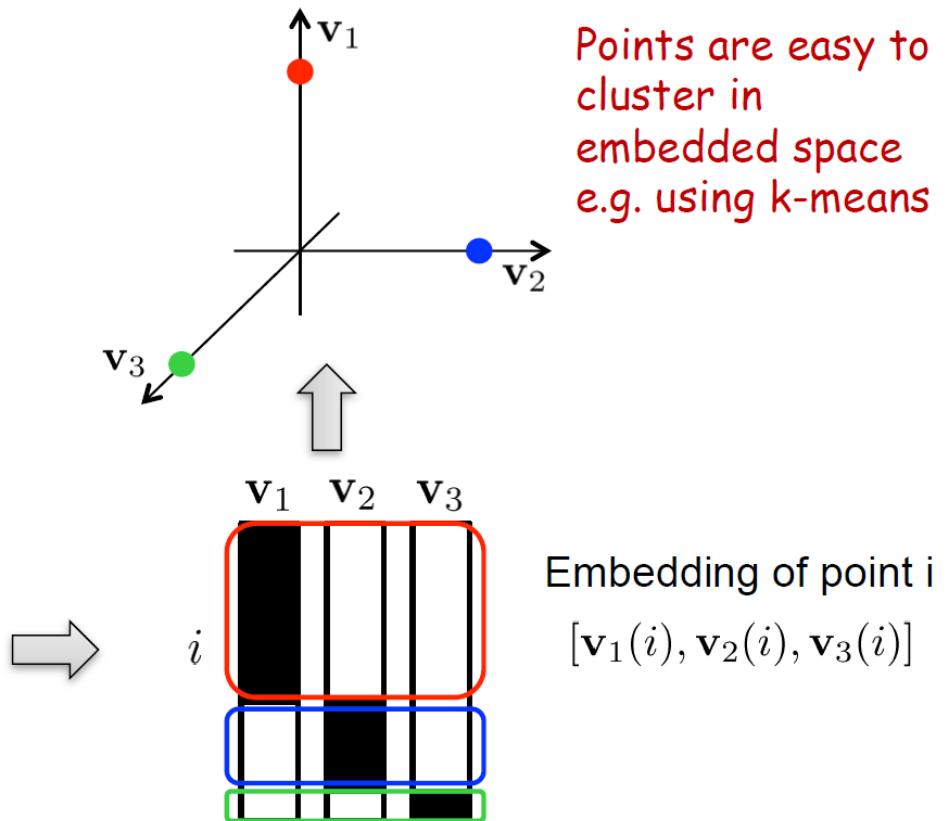
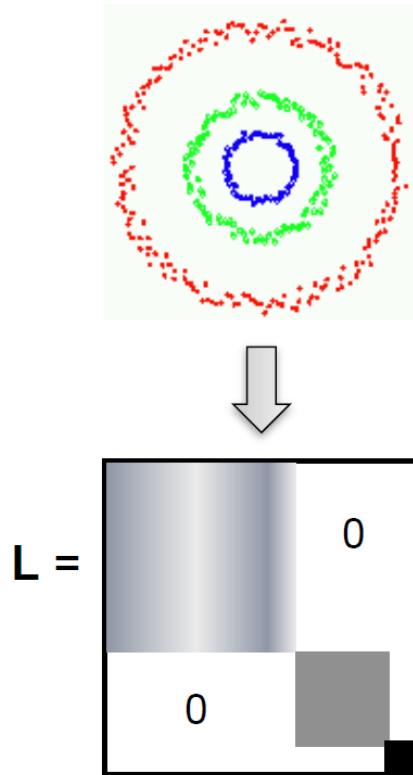
Here is a simple example of a labelled, undirected graph and its Laplacian matrix.

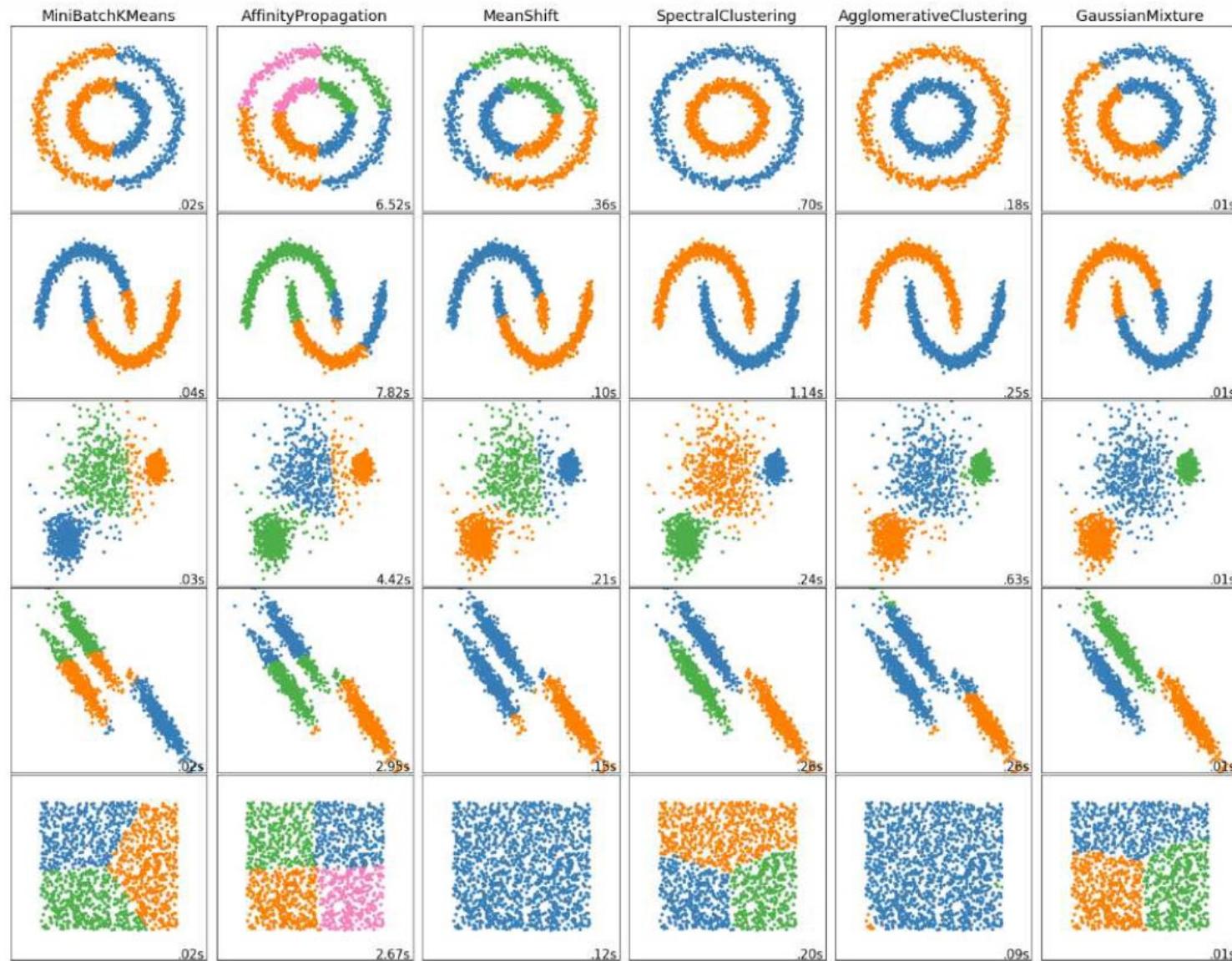
Labelled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Spectral Clustering - Intuition

Eigenvectors of the Laplacian matrix provide an embedding of the data based on similarity.

Disconnected subgraphs





Szeliski Book Figure 5.15 Comparison of different clustering algorithms on some toy datasets, generated using a simplified version of https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html#sphx-glr-auto-examples-cluster-plot-cluster-comparison-py.

Active Contours

While lines, vanishing points, and rectangles are commonplace in the man-made world, curves corresponding to object boundaries are even more common, especially in the natural environment.

Three related approaches to locating such boundary curves in images:

1. **Snakes** inventors (Kass, Witkin, and Terzopoulos 1988): is an energy-minimizing, two-dimensional spline curve that evolves (moves) towards image features such as strong edges.
2. **Level set techniques**: evolve the curve as the zero-set of a characteristic function, which allows them to easily change topology and incorporate region-based statistics.
3. **Intelligent scissors** (Mortensen and Barrett 1995) (Section 5.1.3), allow the user to sketch in real time a curve that clings to object boundaries.

All three of these are examples iteratively move towards their final solution under the combination of image and optional user-guidance forces.

Segmentation with Active Contours

- Active contours became a favorite approach in recent years particularly within the biomedical image processing community.
- Advantages:
 - Express and incorporate physical constraints to segmentation process
 - PDE-based representation and rich numerical solutions
 - Direct extension to higher dimensions

Active Contours/Deformable Contours Snakes and Level sets

Given **initial contour/curve** \mathcal{C} near desired object

Active contours evolve \mathcal{C} , subject to constraints from a given image to fit exact object boundary.

Parametric active contours (aka **snakes**): represented explicitly as parameterized curves.

Geometric active contours: implicitly represented by **level sets**.

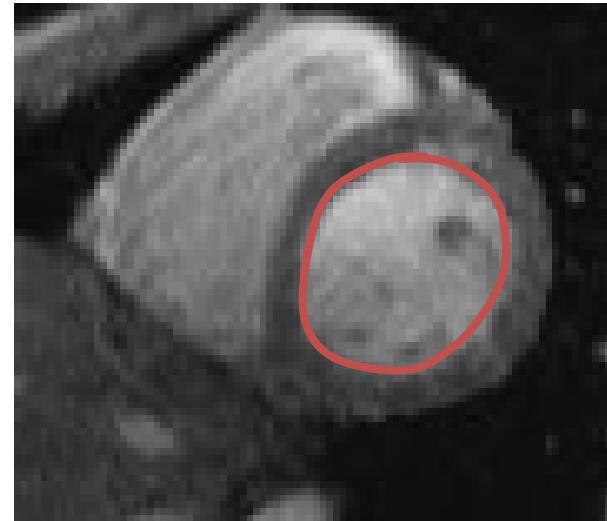
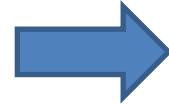
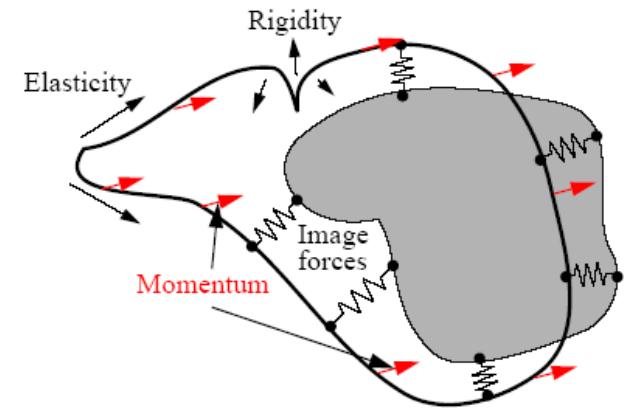
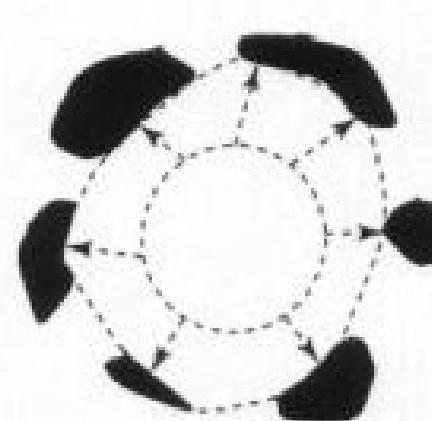
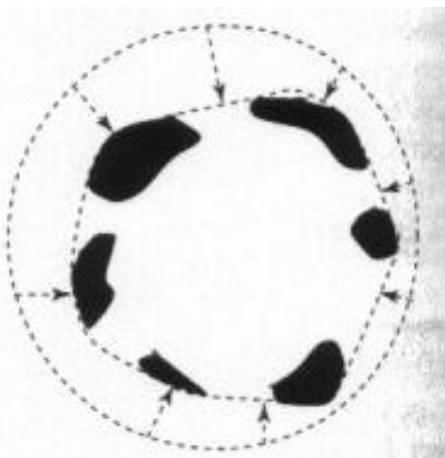
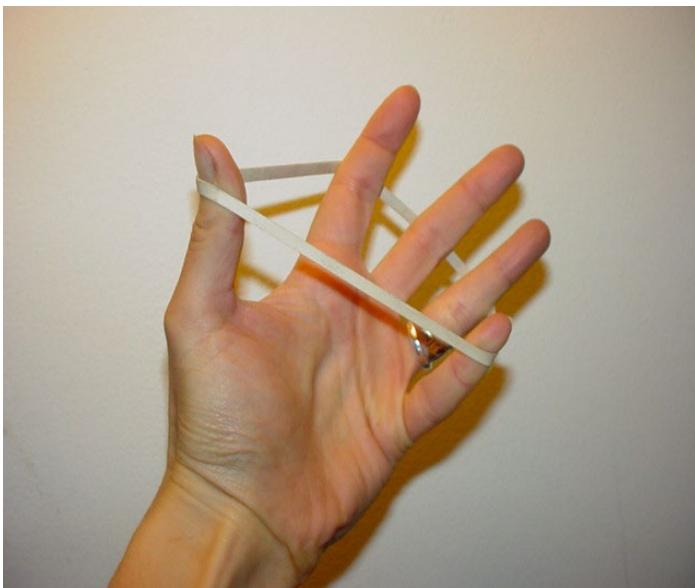
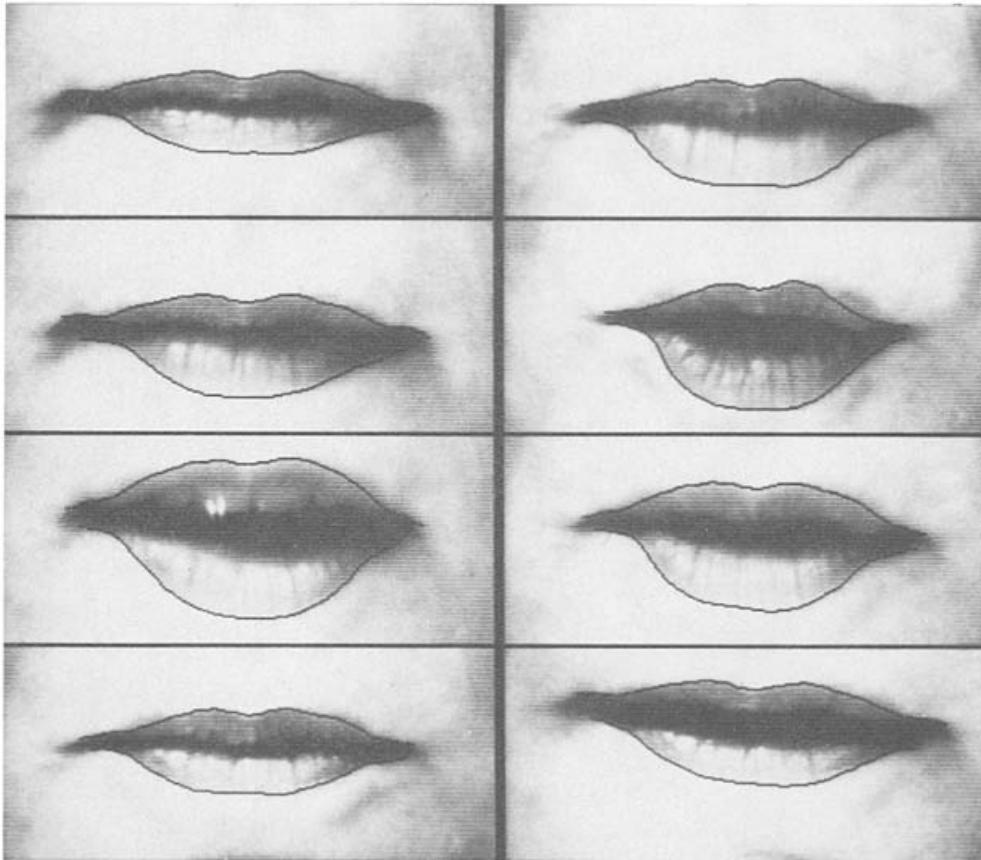


Figure credit: Yuri Boykov

Deformable contours: intuition

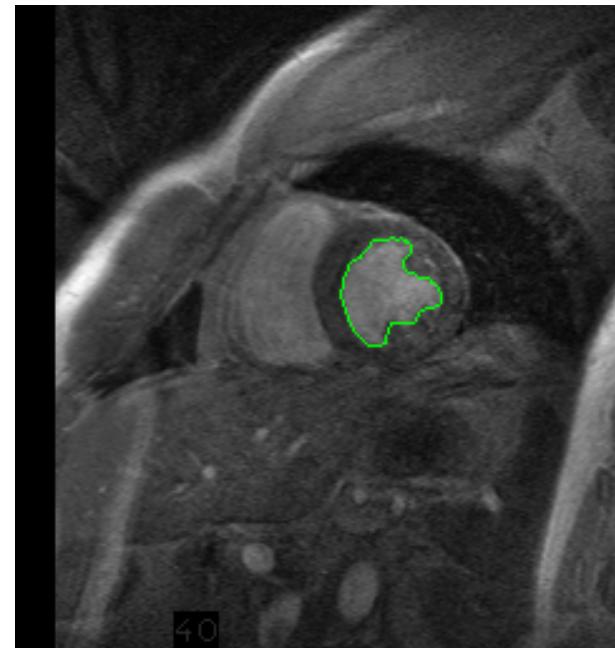
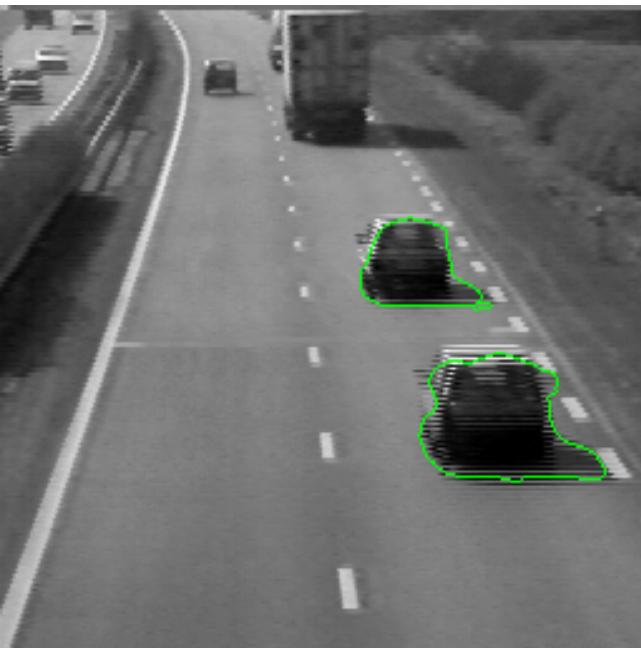


Why do we want to fit deformable shapes?



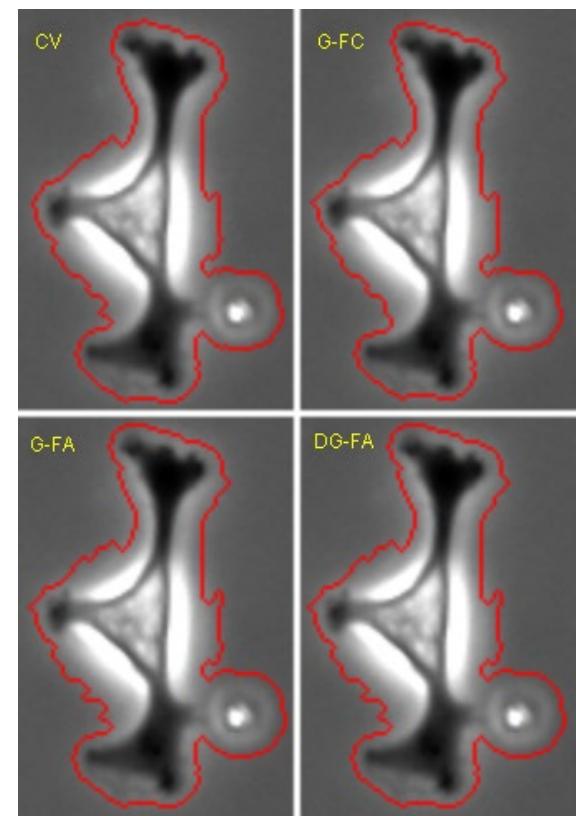
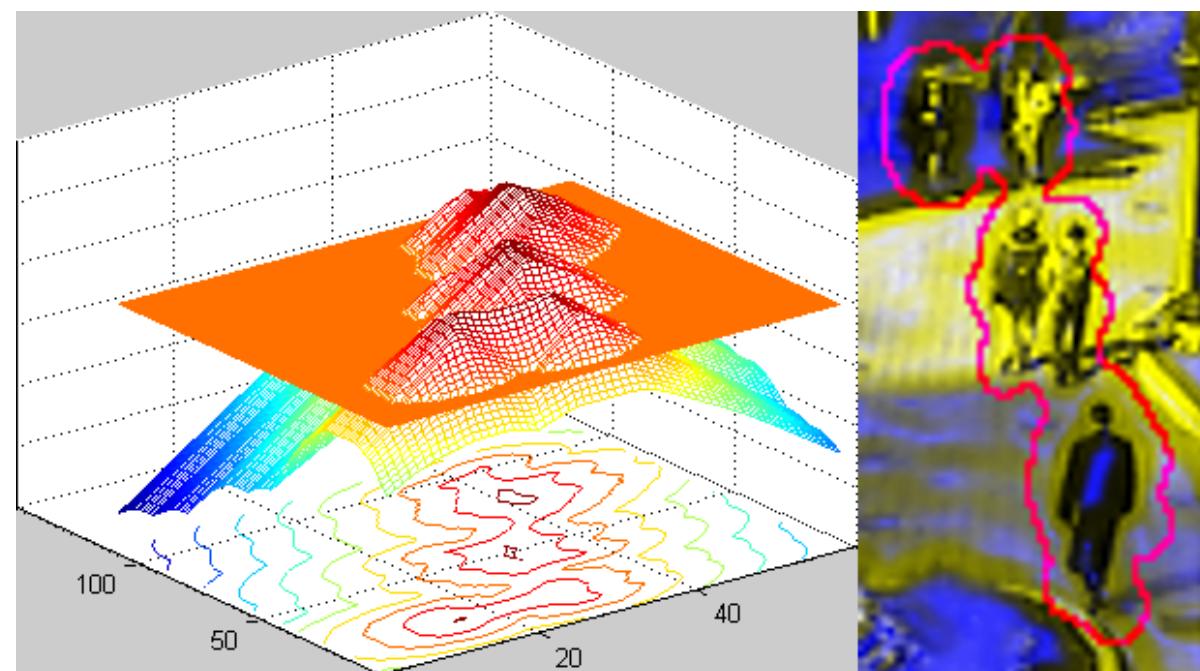
- Non-rigid, deformable objects can change their shape over time, e.g. lips, hands...

Why do we want to fit deformable shapes?



- Non-rigid, deformable objects can change their shape over time.

Active Contours

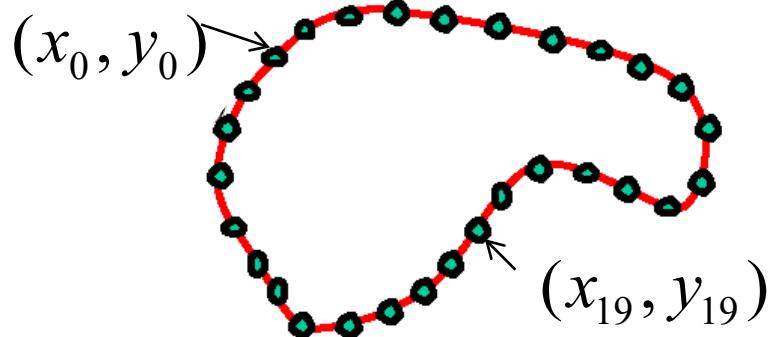


Design Issues

1. Representation of the contours: Parametric (snakes) vs. Non-Paramteric (level sets)
2. Defining the energy functions
 1. External
 2. Internal
3. Minimizing the energy function

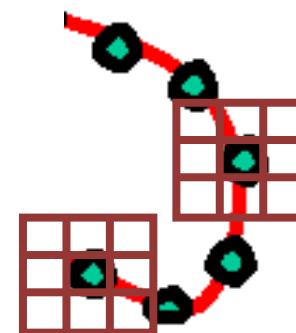
Parametric Representation (Snakes)

- Discrete representation of the contour: list of 2d point positions (“vertices”).



$$v_i = (x_i, y_i), \quad \text{for } i = 0, 1, \dots, n - 1$$

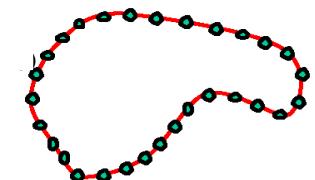
- At each iteration, we'll have the option to move each vertex to another nearby location (“state”).



Energy function

The total energy (cost) of the current snake is defined as:

$$E_{total} = E_{internal} + E_{external}$$



Internal energy: encourage *prior* shape preferences: e.g., smoothness, elasticity, particular known shape.

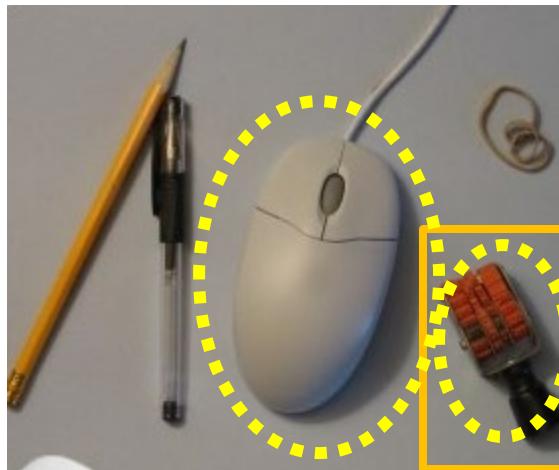
External energy (“image” energy): encourage contour to fit on places where **image structures** exist, e.g., edges.

A **good fit** between the current deformable contour and the target shape in the image will yield a **low** value for this **cost function**.

External energy: intuition

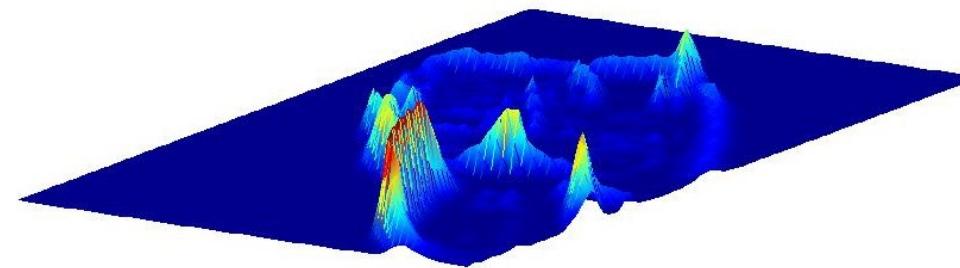
- Measure how well the curve matches the image data
- “Attract” the curve toward different image features
 - Edges, lines, texture gradient, etc.

External image energy



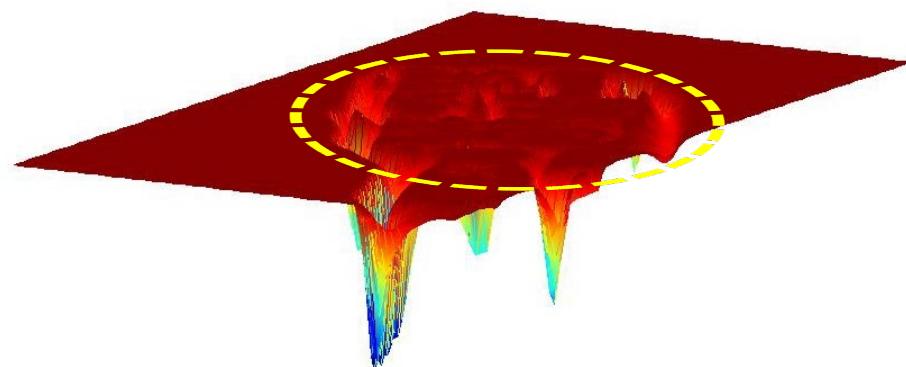
How do edges affect “snap” of rubber band?

Think of external energy from image as gravitational pull towards areas of high contrast



Magnitude of gradient

$$G_x(I)^2 + G_y(I)^2$$



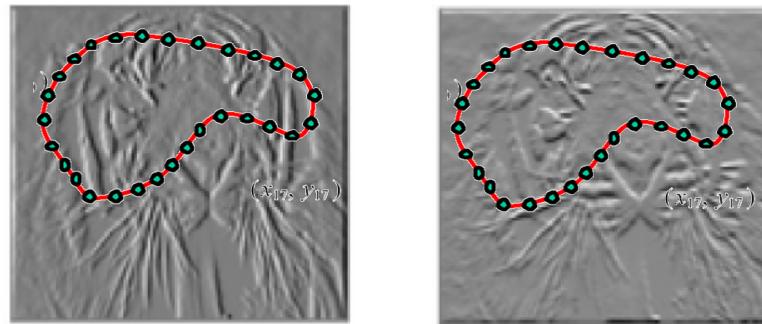
- (Magnitude of gradient)

$$-(G_x(I)^2 + G_y(I)^2)$$

Kristen Grauman

External image energy

- Gradient images $G_x(x, y)$ and $G_y(x, y)$



- External energy at a point on the curve is:

$$E_{external}(\nu) = -(|G_x(\nu)|^2 + |G_y(\nu)|^2)$$

- External energy for the whole curve:

$$E_{external} = - \sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

Internal energy

For a *continuous* curve, a common internal energy term is the “bending energy”.

At some point $v(s)$ on the curve, this is:

$$E_{internal}(v(s)) = \alpha \left| \frac{dv}{ds} \right|^2 + \beta \left| \frac{d^2v}{d^2s} \right|^2$$

Tension,
Elasticity Stiffness,
Curvature



Total energy: function of the weights

$$E_{total} = E_{internal} + \gamma E_{external}$$

$$E_{external} = - \sum_{i=0}^{n-1} |G_x(x_i, y_i)|^2 + |G_y(x_i, y_i)|^2$$

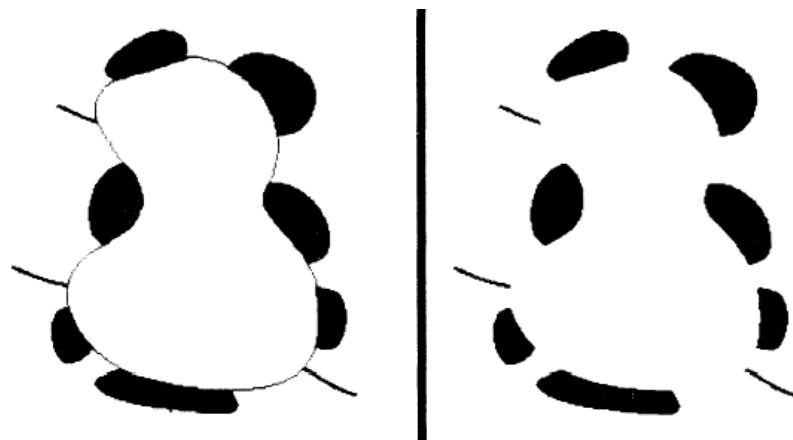
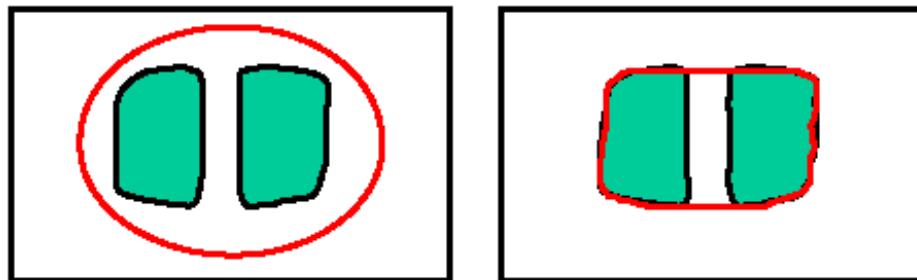
spatial image gradients

$$E_{internal} = \sum_{i=0}^{n-1} \alpha (\bar{d} - \|\nu_{i+1} - \nu_i\|)^2 + \beta \|\nu_{i+1} - 2\nu_i + \nu_{i-1}\|^2$$

*derivatives relative
to contour positions*

Dealing with missing data

- The preferences for low-curvature, smoothness help deal with missing data:

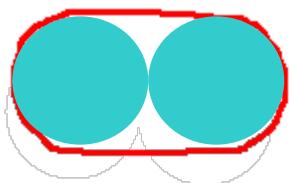


Illusory contours found!

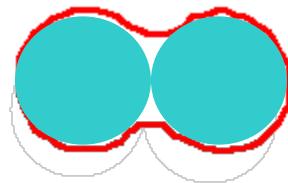
[Figure from Kass et al. 1987]

Total energy: function of the weights

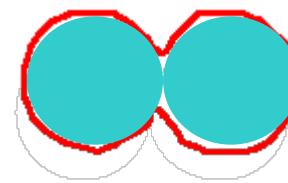
- e.g., α weight controls the penalty for internal elasticity



large α



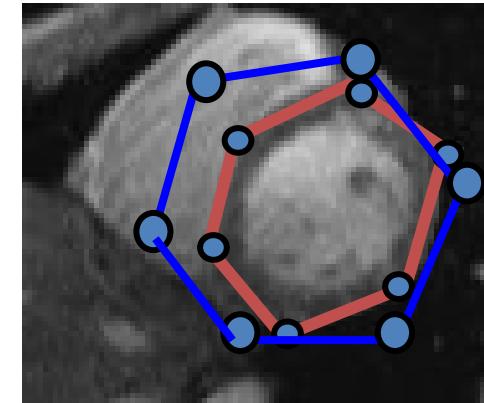
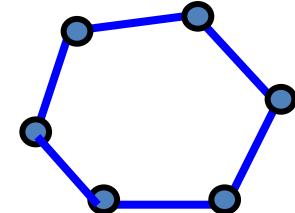
medium α



small α

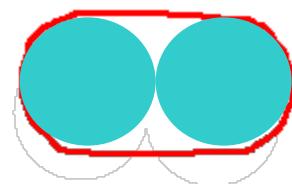
Recap: deformable contour

- A simple elastic snake is defined by:
 - A set of n points,
 - An internal energy term (tension, bending, plus optional shape prior)
 - An external energy term (gradient-based)
- To use to segment an object:
 - Initialize in the vicinity of the object
 - Modify the points to minimize the total energy

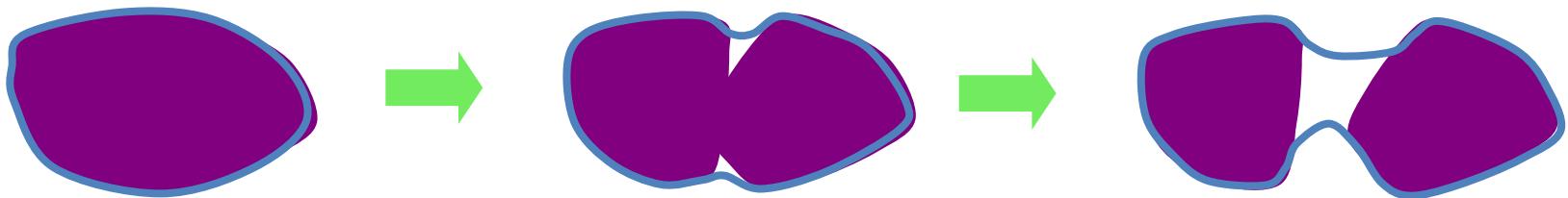


Limitations

- May over-smooth the boundary



- Cannot follow topological changes of objects



Limitations

- External energy: snake does not really “see” object boundaries in the image unless it gets very close to it.

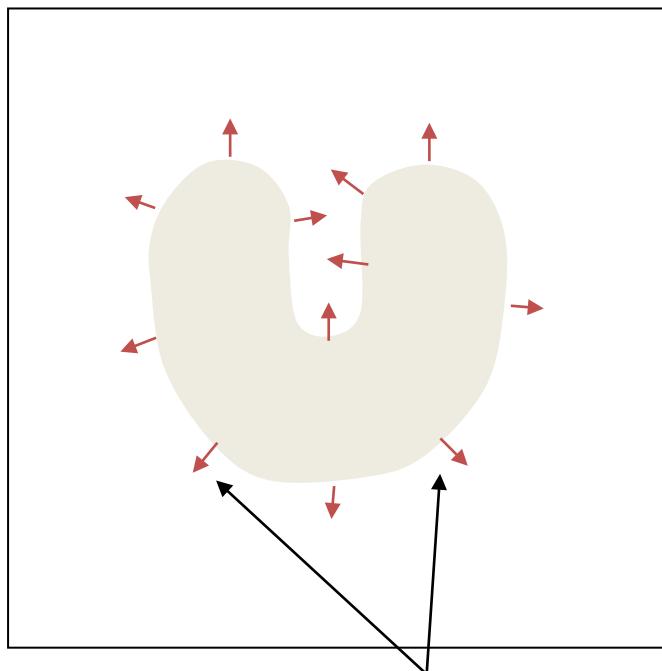
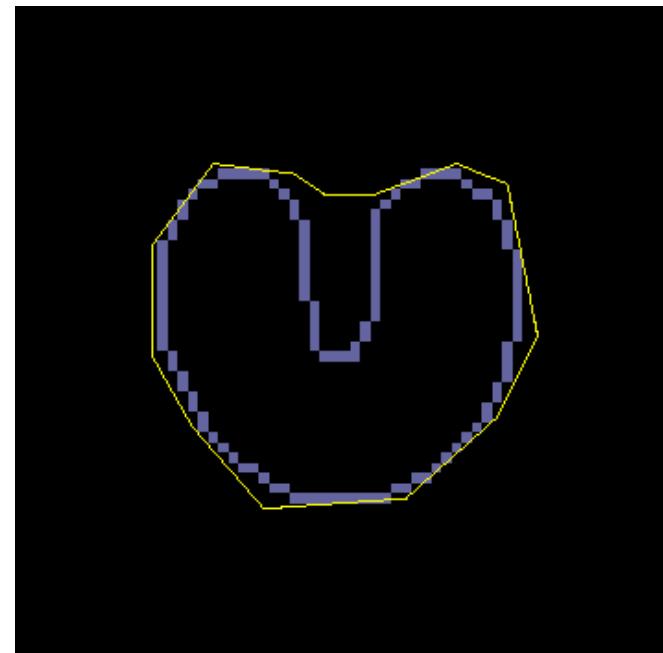


image gradients ∇I
are large only directly on the boundary



Distance transform

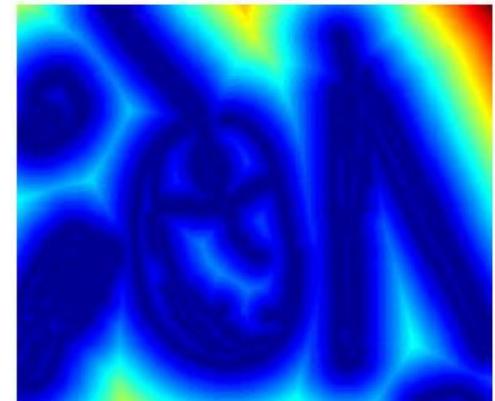
- External image can instead be taken from the **distance transform** of the edge image.



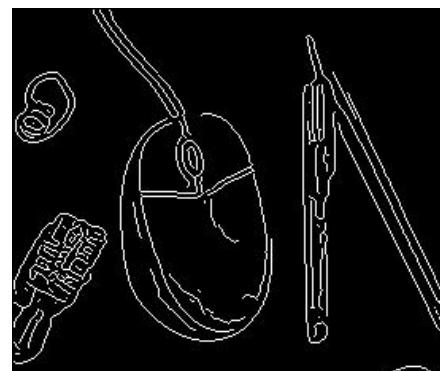
original



-gradient



distance transform



edges

Value at (x,y) tells how far that position is from the nearest edge point (or other binary mage structure)

>> **help bwdist**

Kristen Grauman

Deformable contours: pros and cons

Pros:

- Useful to track and fit non-rigid shapes
- Contour remains connected
- Possible to fill in “subjective” contours
- Flexibility in how energy function is defined, weighted.

Cons:

- Must have decent initialization near true boundary, may get stuck in local minimum
- Parameters of energy function must be set well based on prior information

Level set-based Active Contours

Active contours evolve a curve \mathcal{C} , subject to constraints from a given image.

In level set based active contour methods the curve \mathcal{C} is represented implicitly via a function ϕ by $\mathcal{C} = \{(x,y) | \phi(x,y)=0\}$

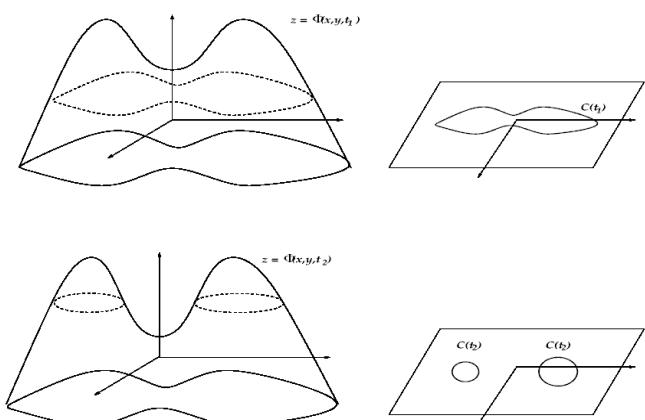
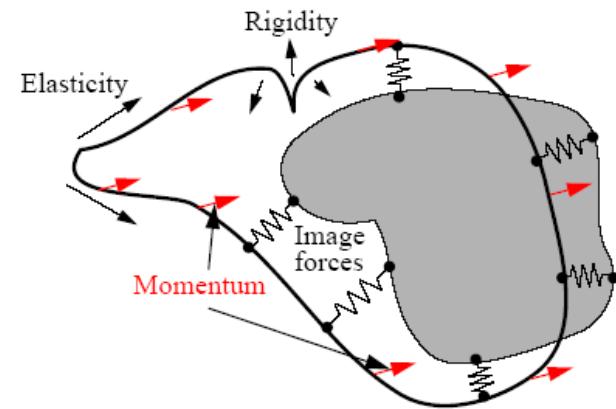
The evolution of the curve is given by the zero-level curve of the function $\phi(t,x,y)$.

Evolving \mathcal{C} in a normal direction with speed F amounts to solving the differential equation

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| F; \quad \phi(0, x, y) = \phi_0(x, y)$$

Normal
direction

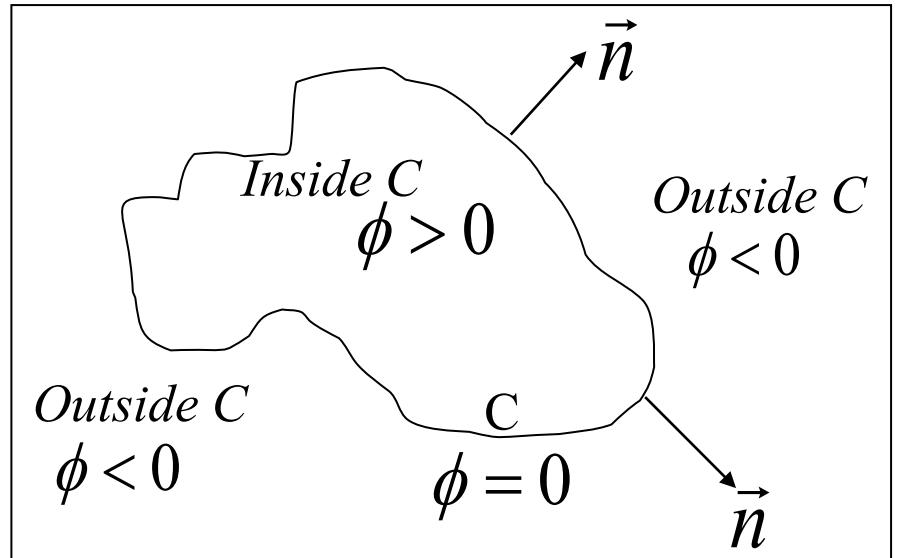
speed



Levelset Representation

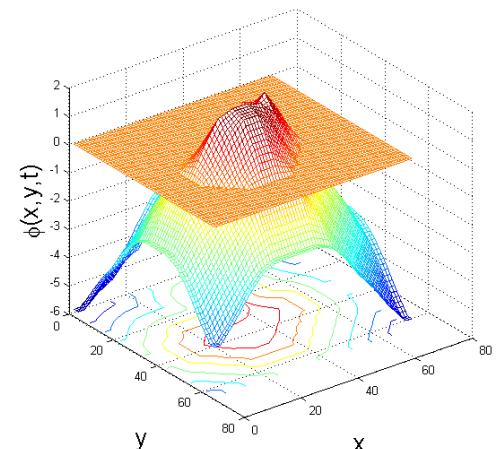
(S. Osher, J. Sethian 87)

$$C = \{(x, y) \mid \phi(x, y) = 0\}$$



$$\text{Normal } \vec{n} = \frac{\nabla \phi}{|\nabla \phi|}, \quad \text{Curvature } K = \operatorname{div} \left(\frac{\nabla \phi}{|\nabla \phi|} \right)$$

- * Allows automatic topology changes, cusps, merging and breaking.
- Originally developed for tracking fluid interfaces.



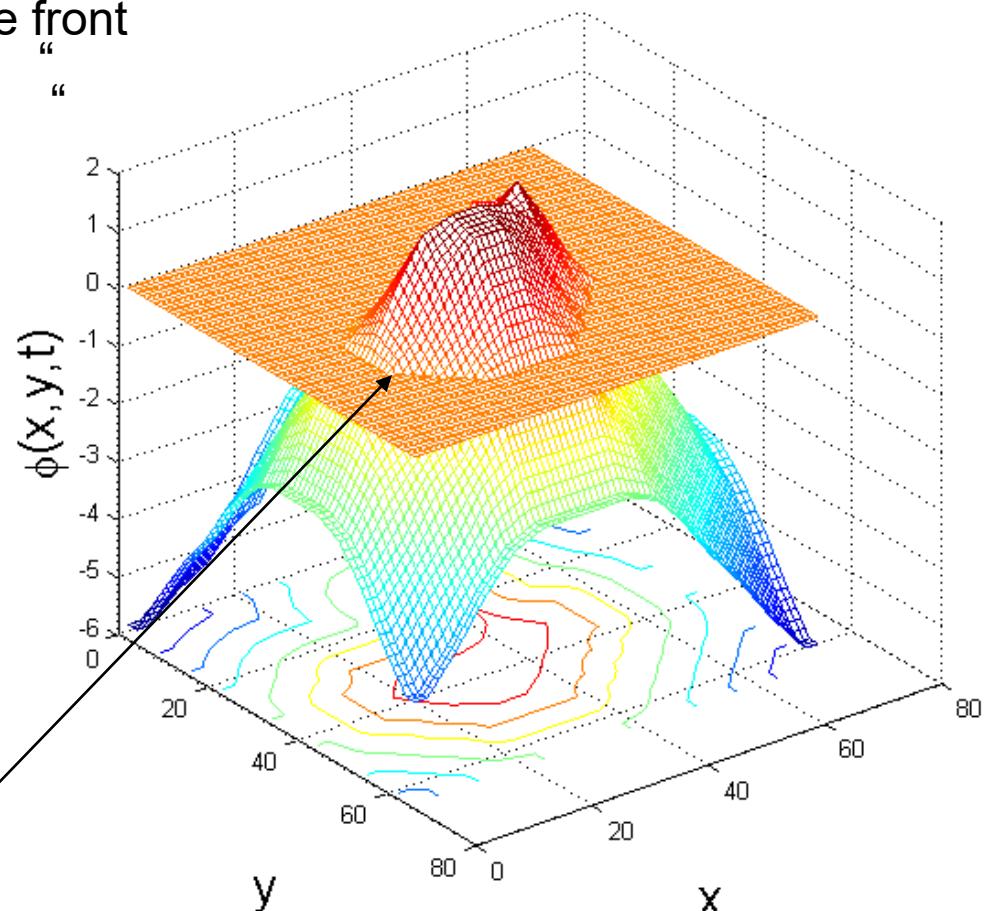
Levelset Representation

Usual choice for ϕ : signed distance to the front $\phi(0)$

$$\phi(x,y,0) = \begin{cases} -d(x,y, \phi) & \text{if } (x,y) \text{ inside the front} \\ 0 & \text{on " " } \\ d(x,y, \phi) & \text{" outside " } \end{cases}$$

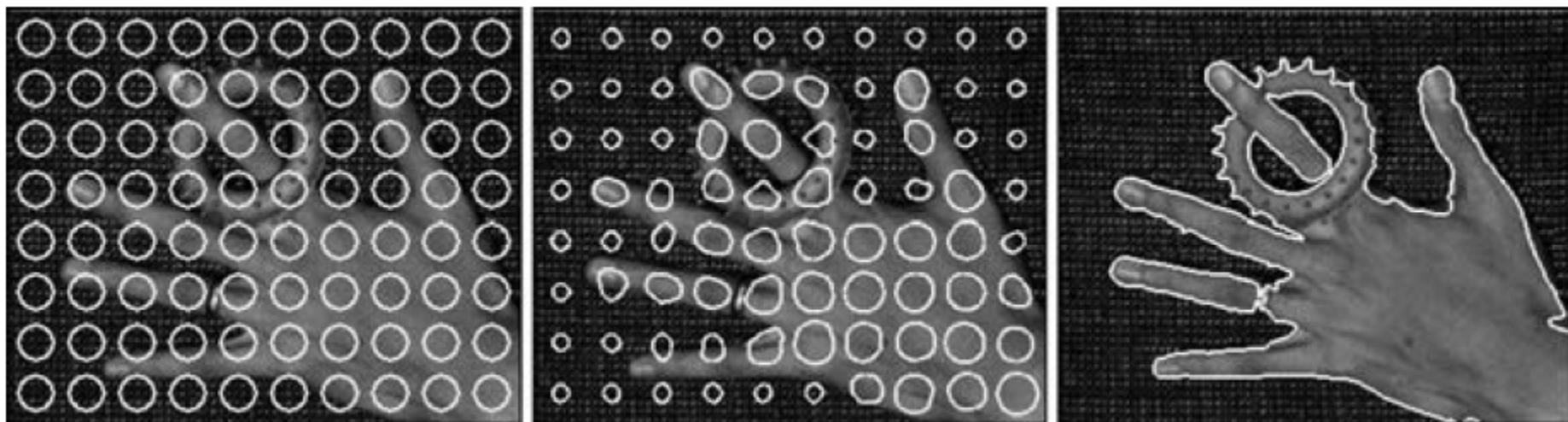
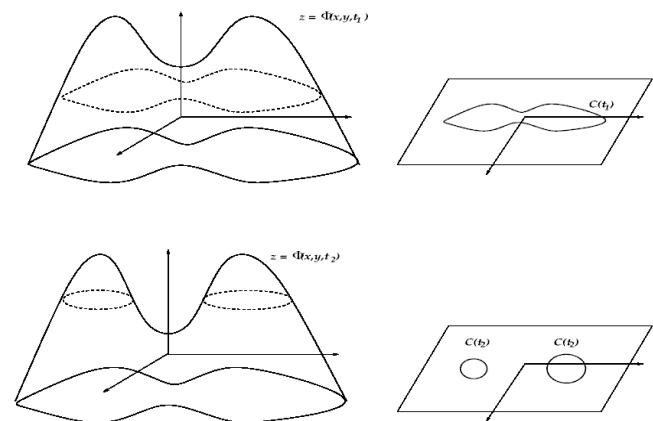
-8	-7	-6	-5	-5	-4	-3	-2	-3	-3	-4	-5	-6	-7
-7	-6	-5	-4	-4	-3	-2	-1	-2	-2	-3	-4	-5	-6
-6	-5	-4	-3	-3	-2	-1	0	-1	-1	-2	-3	-4	-5
-5	-4	-3	-2	-2	-2	-1	0	0	0	-1	-2	-3	-4
-4	-3	-2	-1	-1	-1	-1	0	1	0	-1	-2	-3	-4
-3	-2	-1	0	0	0	0	1	2	1	0	-1	-2	-3
-2	-1	0	1	1	1	1	2	2	1	0	-1	-2	-3
-2	-1	0	1	2	2	2	3	2	1	0	-1	-2	-3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

Contour



Advantages of Level set-based Active Contours

- **Topological flexibility:** Unlike parametric approaches such as classical snake, level set based approaches ensure topological flexibility (allows merge & splits).



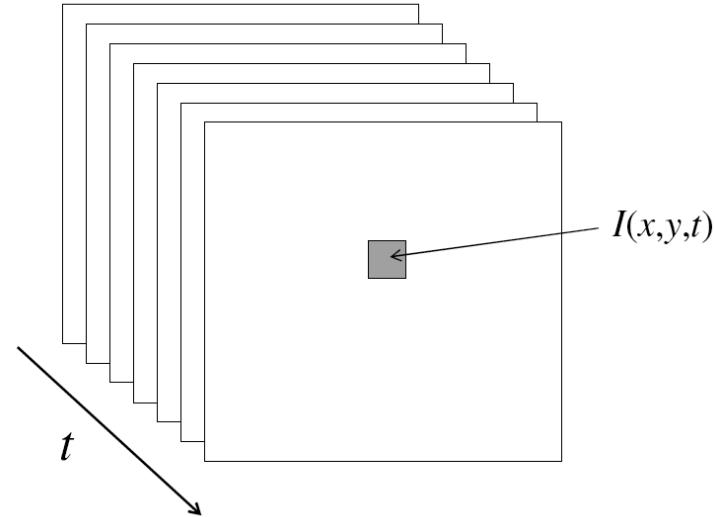
Motion Analysis and Video Processing

CS/ECE 8690 Computer Vision

Spring 2023

Motion Analysis & Video Processing

- Video analysis
- Low level motion
- Detection of moving objects
- Tracking objects



- Video: a sequence of frames captured over time
- Now image data is a function of space (x,y) and time (t)

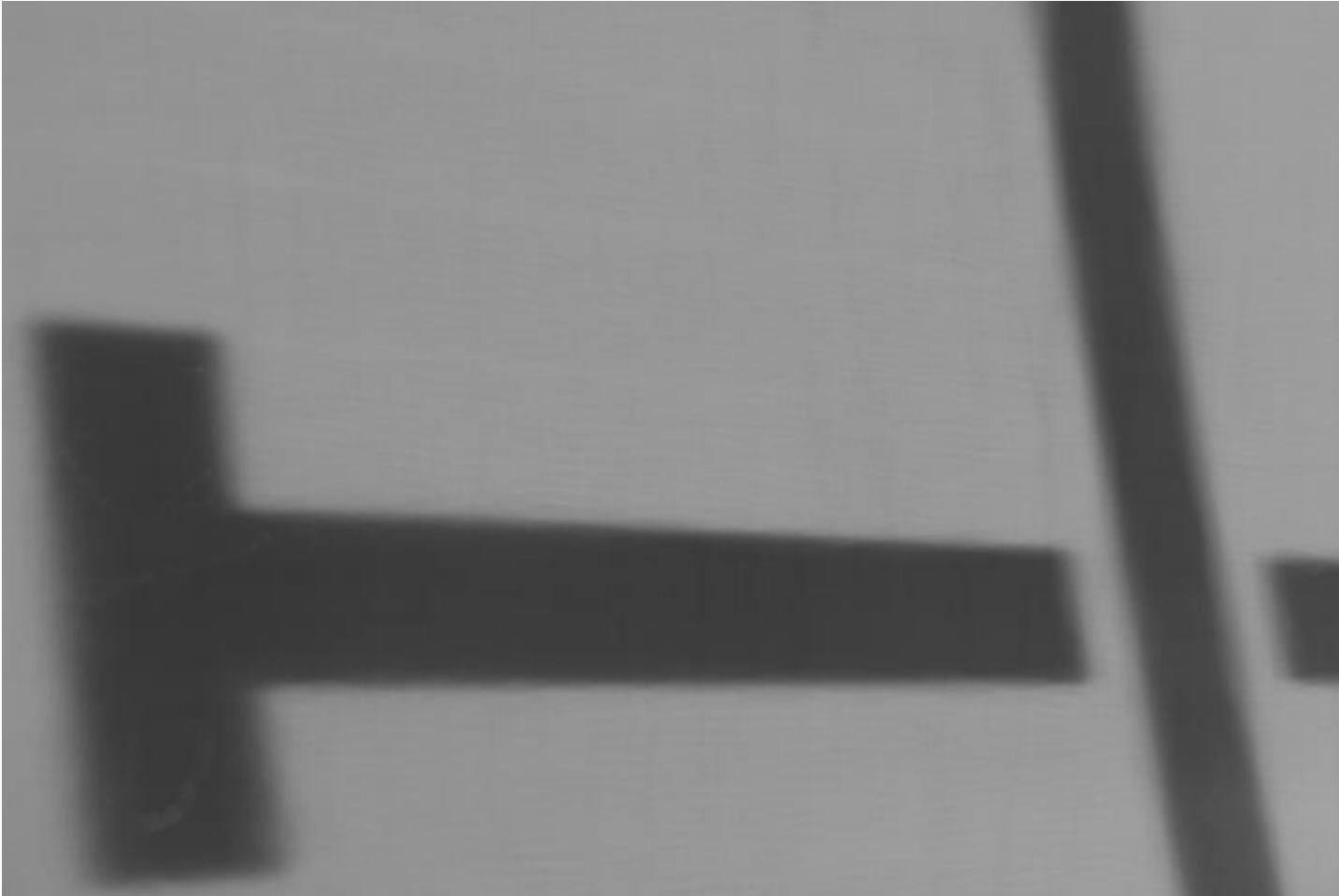
Why estimate visual motion?

1. **Dynamics:** Visual Motion indicates dynamics in the scene
 - Moving objects, behavior
 - Track objects and analyze trajectories
2. **Quality:** Visual Motion can be annoying -- improve video quality
 - Camera instabilities, jitter
 - Measure it; remove it (stabilize)
3. **Spatial Structure:** Visual Motion reveals spatial layout and 3D structure

Input Video



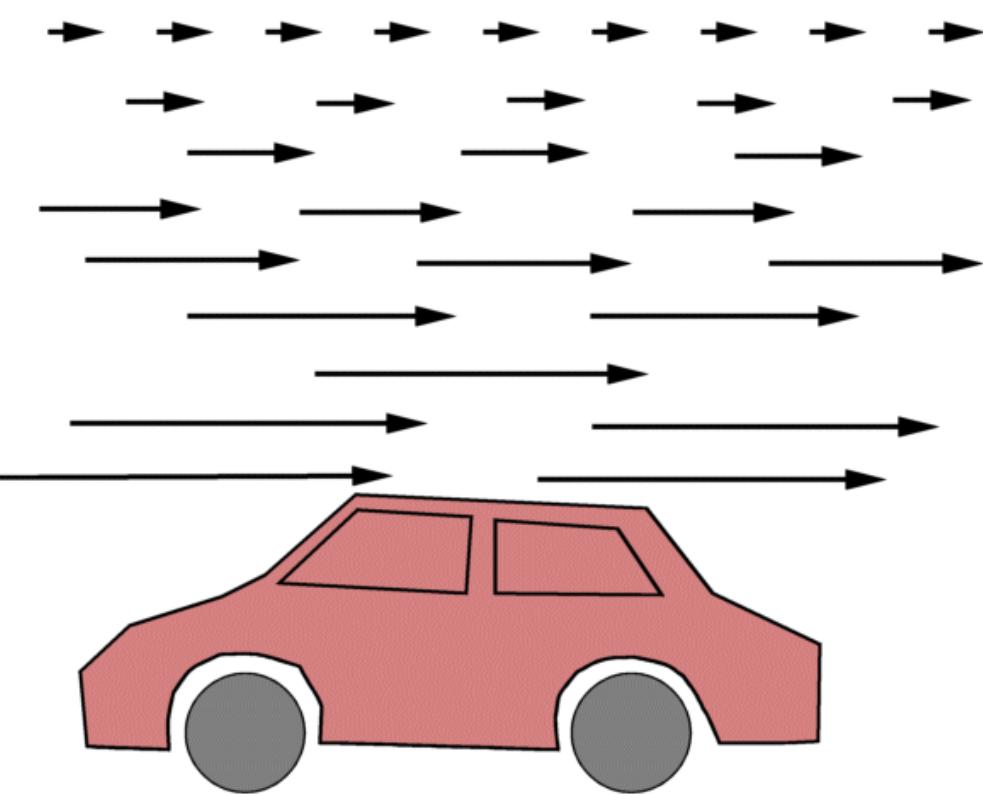
Average Image



Camera Jitter



Motion Parallax



- Motion parallax is a monocular depth cue arising from the relative velocities of objects moving across the retinae of a moving person.
- The term parallax refers to a change in position. Thus, motion parallax is a change in position caused by the movement of the viewer.
- Motion parallax arises from the motion of the observer in the environment.
- You can observe this phenomenon in the video below. The closer the cloud to the plane, the faster it appears to move.

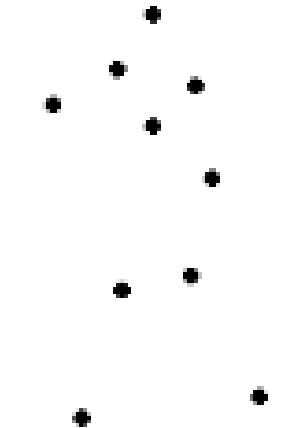
http://www.infovis.net/imagenes/T1_N144_A6_DifMotion.gif

- <http://psychlab1.hanover.edu/Classes/Sensation/MotionParallax.html>
- <https://isle.hanover.edu/Ch07DepthSize/Ch07MotionParallaxExpl.html>

Object versus Motion

Siskind 1997

- Many **actions** can be described entirely in terms of the **relative motions without segmenting or identifying objects.**
- To a large extent, the human conceptualization of the world treats events orthogonally from the objects that fill the roles of those events.



Point light stimuli showing different actions
(Demo by R. Blake)

Bobick 1996

- **Actions** can be recognized from the **motion** itself as opposed to first constructing a **3D model** of a person, and then recognizing the action of the model.
- Motion perception studies in psychology (Johansson's Moving Light Displays).



In spite of strong blurring the action in this video can be recognized (Demo: L. Davis & A. Bobick)

Why estimate visual motion?

1. **Dynamics:** Visual Motion indicates dynamics in the scene
 - Moving objects, behavior
 - Track objects and analyze trajectories
2. **Quality:** Visual Motion can be annoying -- improve video quality
 - Camera instabilities, jitter
 - Measure it; remove it (stabilize)
3. **Spatial Structure:** Visual Motion reveals spatial layout and 3D structure

Moving Object Detection

Goal: Segment moving regions from the rest of the image (background).

Rationale: Provide focus of attention for later processes such as

- Tracking,
- Classification,
- Activity/action/behavior/event analysis

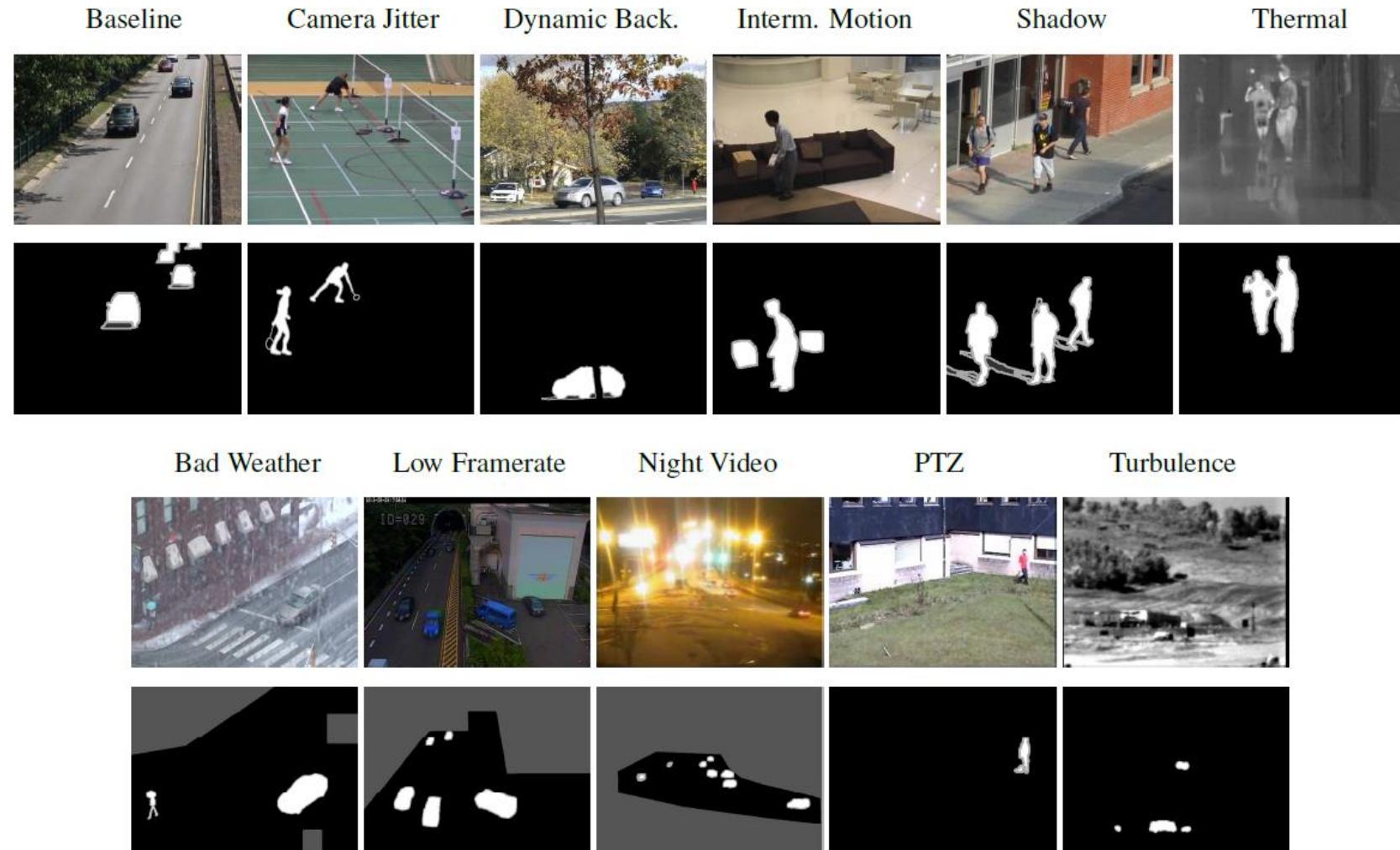


Figure 1. Sample video frames from each of the 11 categories in the new dataset available at [www.changedetection.net](http://changedetection.net). The categories in the first row exist in both 2012 and 2014 dataset while the ones in the second row only exist in 2014 dataset.

Approaches

Change Detection

1. **Optical Flow Analysis:** Characteristics of flow (velocity) vectors of moving objects over time are used to detect changed regions.
 - Advantage: can be used in the presence of camera motion.
 - Disadvantage: computationally expensive.
2. **Temporal differencing:** Compute change between consecutive frames:
$$| \text{frame}_i - \text{frame}_{i-1} | > \text{Th}$$
 - Advantage: very adaptive to dynamic environments.
 - Disadvantage: has problems in extraction of all relevant feature pixels (aperture problem).
3. **Background subtraction:** Moving regions are detected through difference between the current frame and a reference background image.
$$| \text{frame}_i - \text{Background}_i | > \text{Th}$$
 - Advantage: provides the most complete feature data.
 - Disadvantage: sensitive to dynamic scene changes due to lighting and extraneous events.

Temporal Differencing

- Rely on pixelwise differences between consecutive frames.
- They do not maintain a background model,

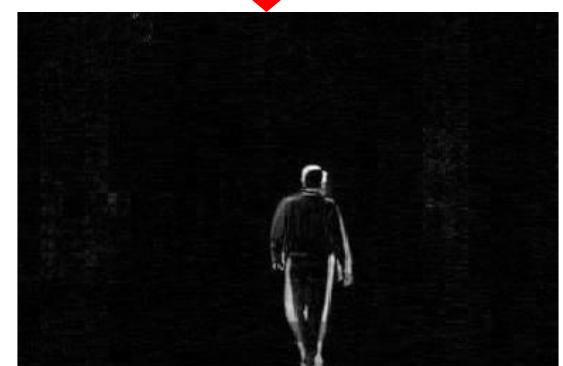
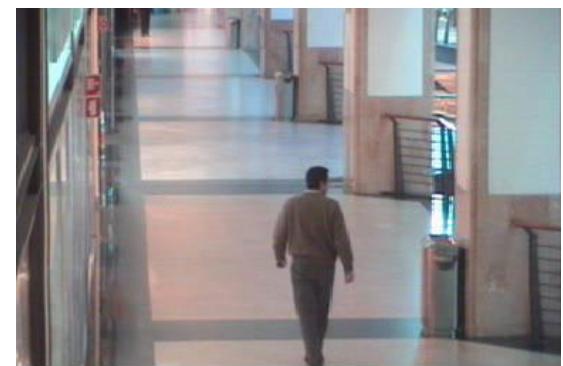
$$| \text{frame}_i - \text{frame}_{i-1} | > \text{Th}$$

Advantages:

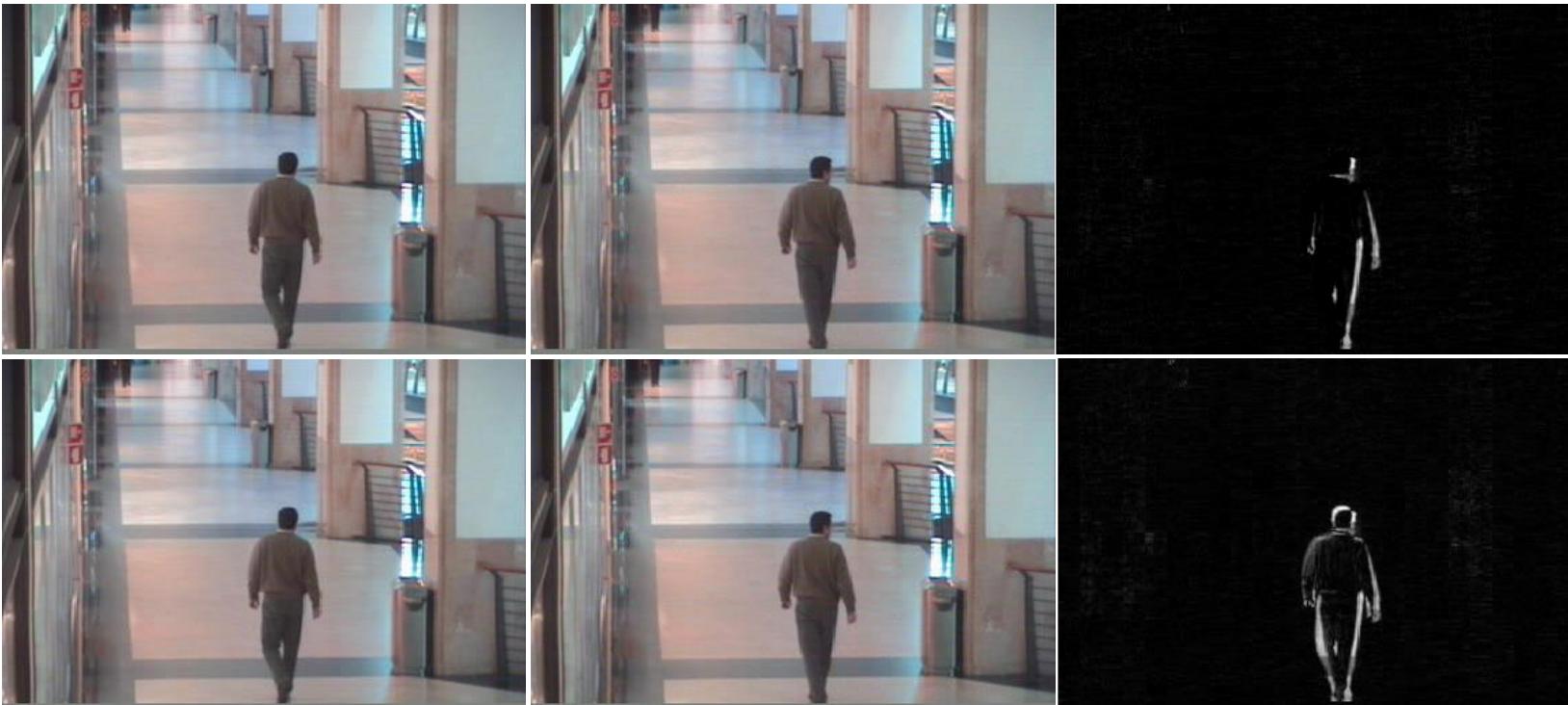
- They are simple and fast
- They can quickly adapt to different changes
 - thus are suitable for dynamic backgrounds, illumination changes, uncovered background by removed objects etc.

Disadvantage:

- Without an explicit background model,
cannot detect slow **moving or stopped objects**
- **Foreground aperture problem**
fail to detect parts of objects (particularly large
objects with homogeneous interiors result in holes).



Attention to Variable Types!



Approaches

Change Detection

1. **Optical Flow Analysis:** Characteristics of flow (velocity) vectors of moving objects over time are used to detect changed regions.
 - Advantage: can be used in the presence of camera motion.
 - Disadvantage: computationally expensive.
2. **Temporal differencing:** Compute change between consecutive frames:
$$| \text{frame}_i - \text{frame}_{i-1} | > \text{Th}$$
 - Advantage: very adaptive to dynamic environments.
 - Disadvantage: has problems in extraction of all relevant feature pixels (aperture problem).
3. **Background subtraction:** Moving regions are detected through difference between the current frame and a reference background image.
$$| \text{frame}_i - \text{Background}_i | > \text{Th}$$
 - Advantage: provides the most complete feature data.
 - Disadvantage: sensitive to dynamic scene changes due to lighting and extraneous events.

Background Subtraction

- Background subtraction based methods maintain and use an explicit background model
- Moving regions are detected through difference between the current frame and a reference background image.

$$| \text{frame}_i - \text{Background}_i | > \text{Th}$$

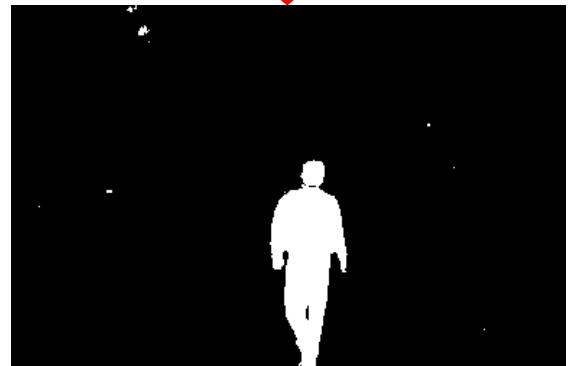
Advantage: Provides the most complete feature data.

- Do not suffer from foreground aperture problem
- Can detect slow moving or stopped objects

Disadvantage: sensitive to dynamic scene changes due to lighting and extraneous events i.e. uncovered background by removed objects etc.

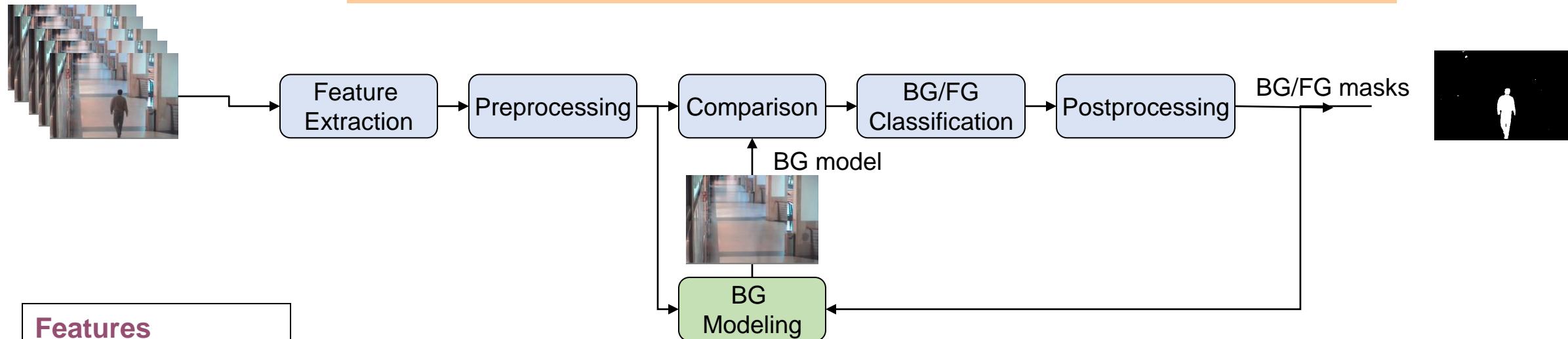
Widely used:

- Traffic monitoring (counting vehicles, detecting & tracking vehicles, pedestrians),
- Human action recognition (run, walk, jump, squat),
- Human-computer interaction
- Object tracking



Background Subtraction

1. Compare incoming frames to a reference image (background model),
2. Locate regions in the incoming frame that have significantly changed.



Features

- Luminance
- Color
- Edge maps
- Albedo (reflectance) image
- Intrinsic images
- Region statistics

Preprocessing

- Spatial smoothing
- Temporal smoothing
- Color space conversions

Comparison

- Differentiation
- Likelihood ratioing

Classification

- Thresholding
- Clustering

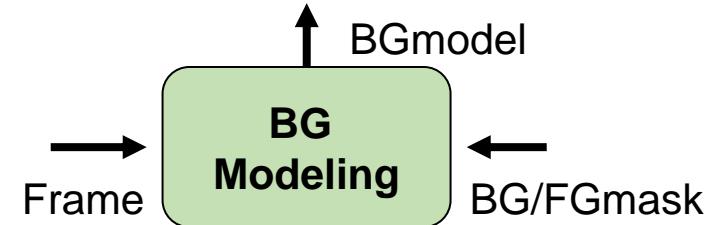
Postprocessing

- morphological filtering
- connectivity analysis
- color analysis
- edge analysis
- shadow elimination

Background Modeling

Background modeling involves:

1. Bootstrapping
2. Representation
3. Maintenance (update)



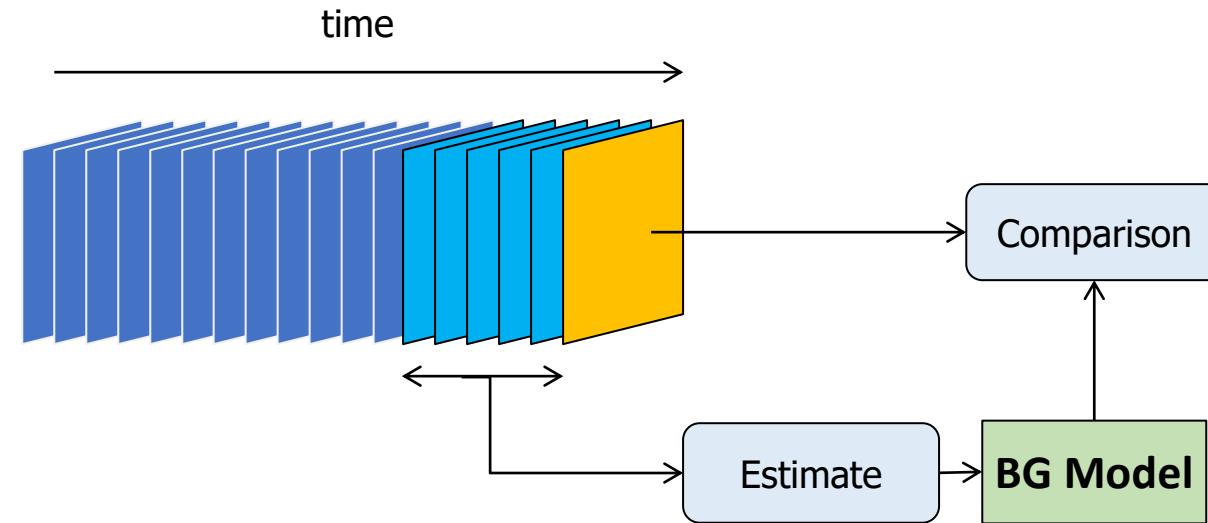
Bootstrapping:

- A clean background, free of moving objects, is not always available.
- Bootstrapping takes a video sequence in which moving objects are present, and outputs a background model describing static parts of the scene.

Maintenance:

- Backgrounds change over time;
- non-adaptive methods are only suitable for controlled environments and short-term tracking without significant changes in the scene.

Classes of Background Subtraction Techniques-1



Non-recursive techniques

- Use a sliding window approach
- Store a buffer of previous L frames
- Estimate the background model based on temporal variation of each pixel within the buffer.

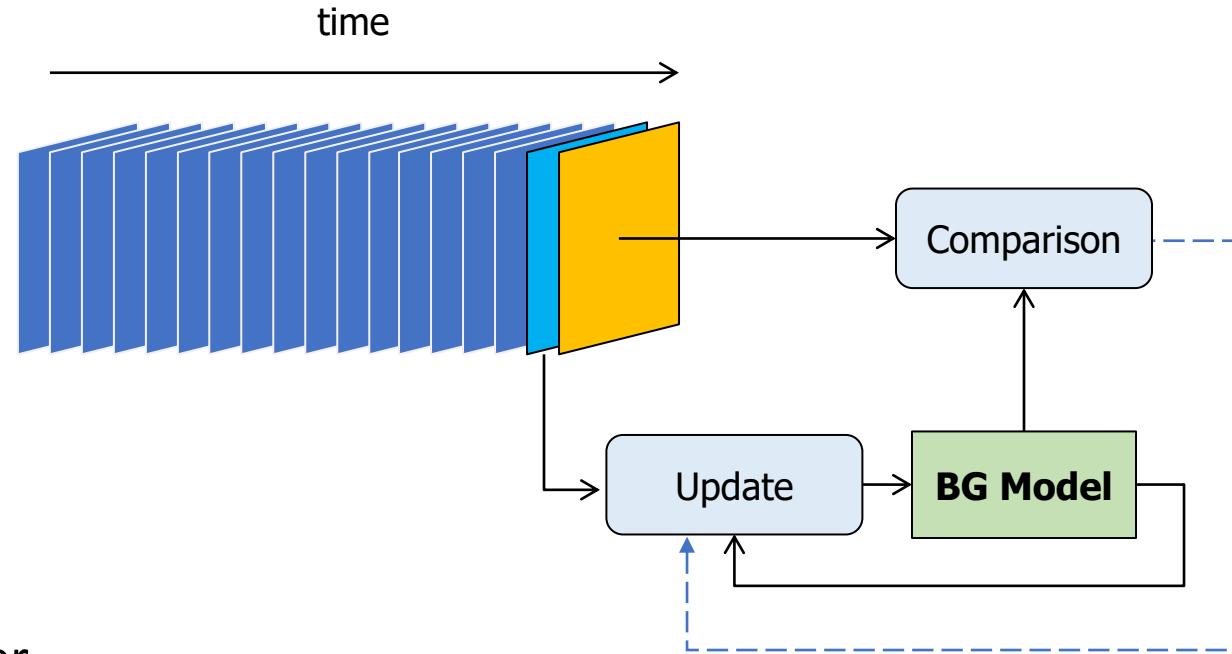
Advantage:

- Highly adaptive since they don't depend on the history beyond the frames stored in the buffer.

Disadvantage:

- Storage requirements are significant, especially in case of objects moving slowly.

Classes of Background Subtraction Techniques-2



Recursive techniques

- Do not maintain a buffer.
- Recursively update a single background model based on each input frame.

Advantage:

- Require less storage.

Disadvantage:

- Any error in the background affect the model for a longer period of time (blending).

Traditional Methods-1

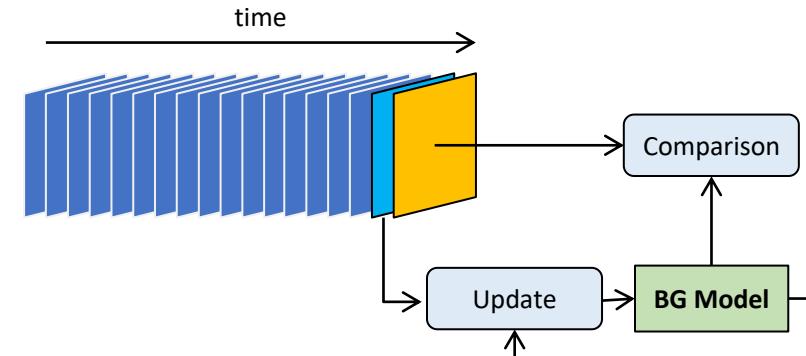
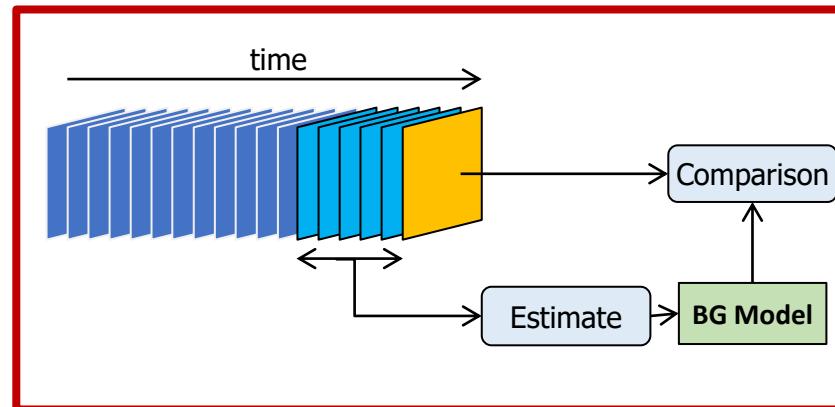
Median filter: The background model at each pixel is modeled as the median of the pixel's recent history.

- non-recursive and uni-modal
- fast but very memory consuming
- recursive version: Approximated median filter

Mean & threshold: The background model at each pixel is modeled as the average of the pixel's recent history.

- non-recursive and uni-modal
- result is too sensitive to the threshold value
- it is difficult to choose an appropriate threshold
- recursive alternative: running average

$$BG(t+1) = \alpha \text{frame}(t) + (1 - \alpha) BG(t)$$



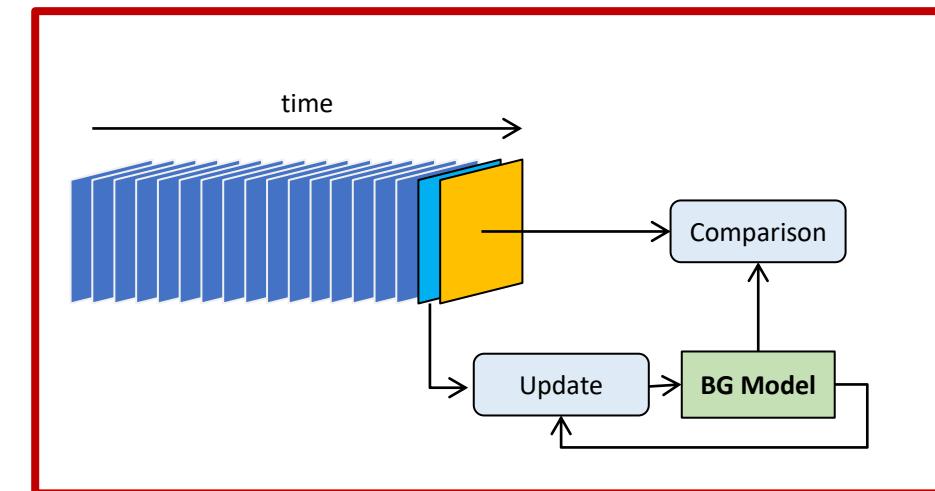
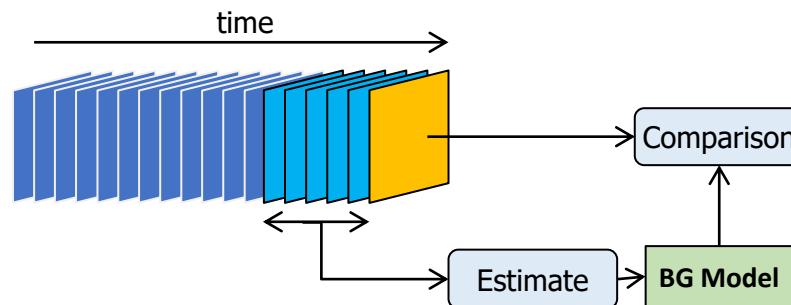
Traditional Methods-2

Temporal Derivative:

- Minimum, maximum and maximum temporal derivative for each pixel are saved.
- Pixels that deviates from the minimum or the maximum for more than the maximum temporal derivative are considered as part of the foreground.

Mean and Covariance (Single Gaussian): The histogram of the previous values for each pixel is modeled with a Gaussian distribution (μ, σ) .

- uni-modal and recursive techniques where
- Update equations:
- $\mu(t+1) = \alpha \text{frame}(t) + (1 - \alpha)\mu(t)$
- $\sigma^2(t+1) = \alpha(\text{frame}(t) - \mu(t))^2 + (1 - \alpha)\sigma^2$



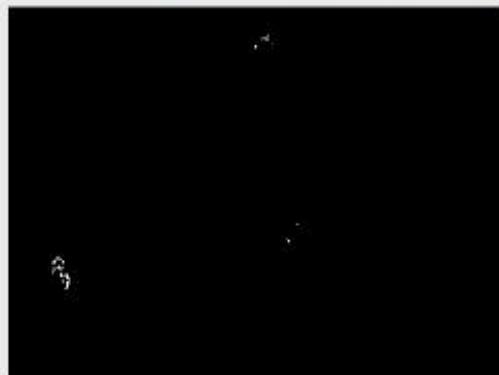
Traditional Approaches for Background Modeling-Frame Differencing

- Frame differencing

- The estimated background is just the previous frame.
- Works for certain object speeds and frame rates.
- Very sensitive to threshold.



high threshold



absolute difference



low threshold

Traditional Approaches for Background Modeling-Median

(median)



Traditional Approaches for Background Modeling-Averaging

- A standard method of adaptive background modeling is based on averaging the images over time
 - The background must be visible most of the time.
 - Objects must move continuously.
 - Not robust when the scene contains multiple, slowly moving objects.

Problems with Basic Methods

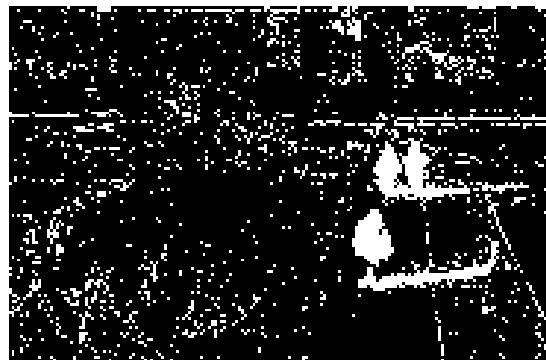
BGmodel



frame #216



BGsubtraction Low Threshold



BGsubtraction High Threshold



Challenging Situations in Moving Object Detection-1

1. Moved objects: A background object that moved should not be considered part of the foreground forever after.



2. Gradual illumination changes alter the appearance of the background (time of day).



3. Sudden illumination changes alter the appearance of the background (light switch).



4. Periodic movement of the background: Background may fluctuate, requiring models which can represent disjoint sets of pixel values (waving trees).

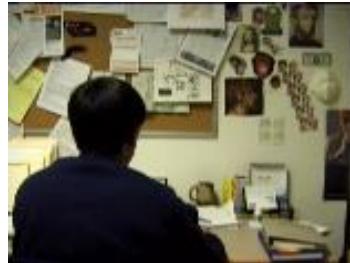


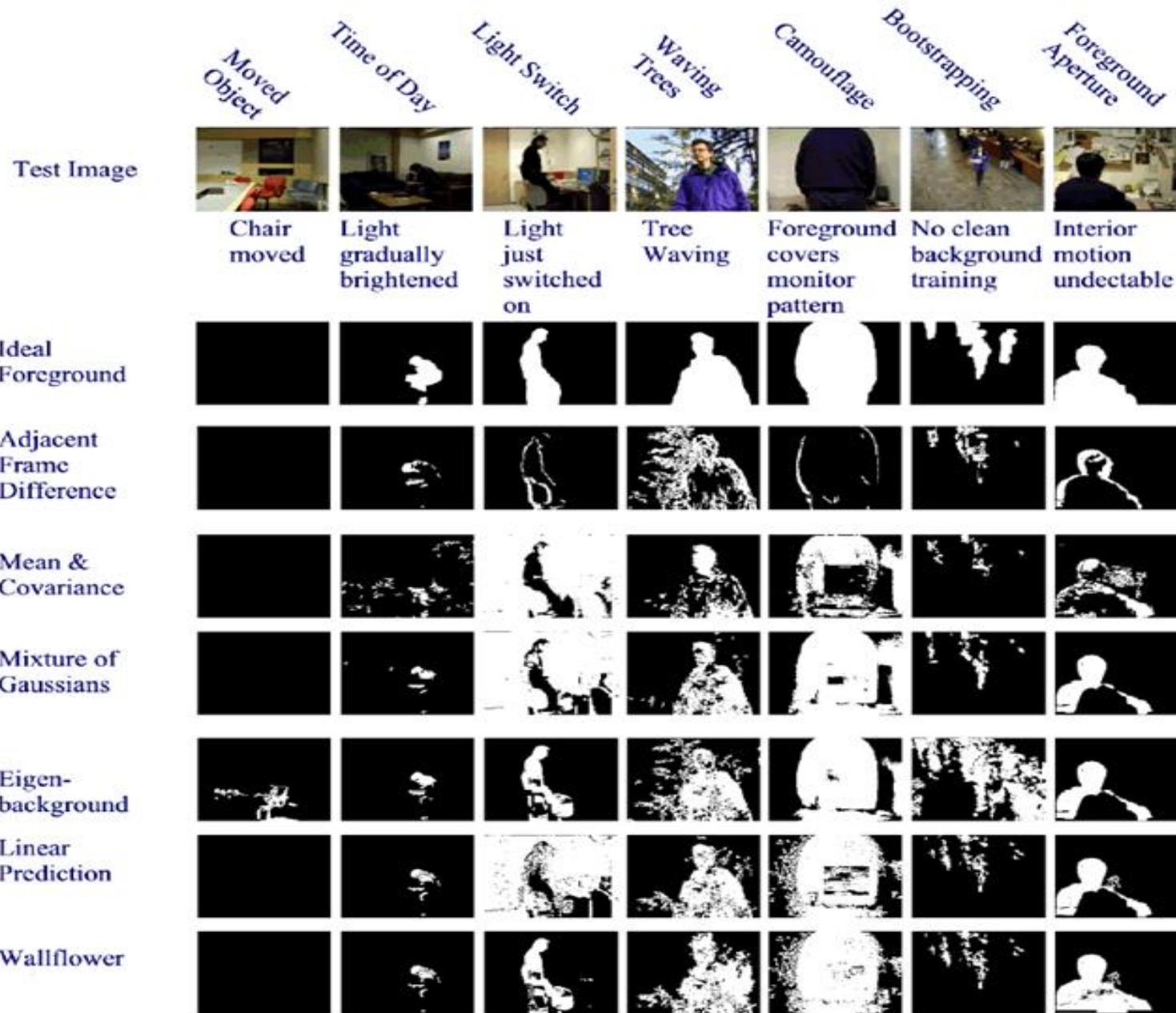
5. Camouflage: A foreground objects' pixel characteristics similar to modeled background.

Picture from: Toyama, K., Krumm, J., Brumitt, B., Meyers, B., 1999. Wallflower: Principles and practice of background maintenance. In: Proceedings of IEEE International Conference on Computer Vision, pp. 255-261.

Challenging Situations in Moving Object Detection-2

6. **Bootstrapping:** A training period absent of foreground objects is not always available.
7. **Foreground aperture:** When an homogeneously colored object moves, change in interior pixels can not be detected.
8. **Sleeping person:** When a foreground object becomes motionless it cannot be distinguished from a background.
9. **Waking person:** When an object initially in the background moves, both the object and the background appear to change.
10. **Shadows:** Foreground objects' cast shadows appear different than modeled background.





Picture from: Toyama, K., Krumm, J., Brumitt, B., Meyers, B., 1999. Wallflower: Principles and practice of background maintenance. In: Proceedings of IEEE International Conference on Computer Vision, pp. 255-261.

Test images: <http://research.microsoft.com/en-us/um/people/jckrumm/wallflower/testimages.htm>

#citations 10813

Case Study :

- "Adaptive background mixture models for real-time tracking",
- C. Stauffer and E. Grimson,
- *IEEE Computer Vision and Pattern Recognition Conference*, Vol.2, pp. 246-252, 1998

Requirements

A robust moving object detection system should be able to handle:

1. Variations in lighting (i.e., gradual and sudden)
2. Multiple moving objects
3. Moving scene clutter (i.e., tree branches, sea waves, etc.)
4. Other arbitrary changes in the scene (i.e., parked cars, camera oscillations, etc.)

Stauffer & Grimson's Moving Object Detection using Mixture of Gaussians (MoG) Method

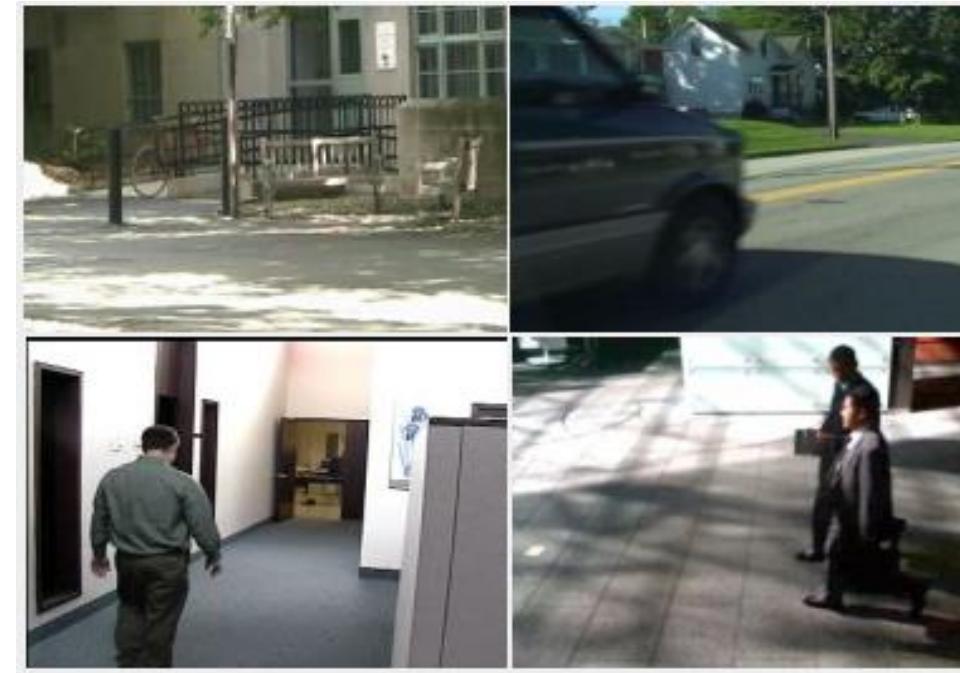
Rationale

If each pixel resulted from a single particular surface

- If lighting is constant: a single Gaussian is adequate.
- If lighting changes over time: a single, adaptive Gaussian is adequate.

In practice,

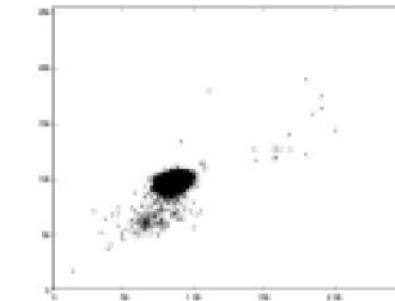
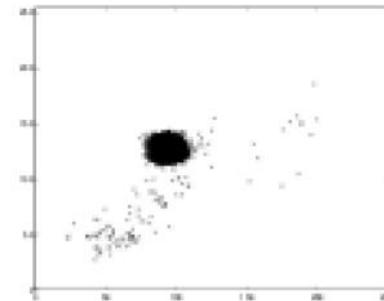
- multiple surfaces often appear in the field of view of a particular pixel (repetitive motion i.e tree branches).
- the lighting conditions change.
- scene changes



Solution: Mixture of Gaussians (Multiple, Adaptive Gaussians)

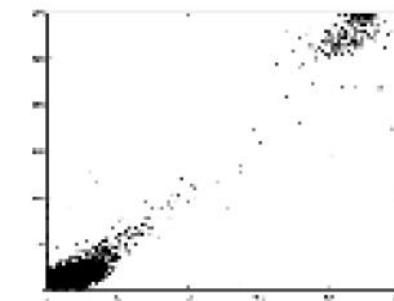
Background mixture models

Change in time



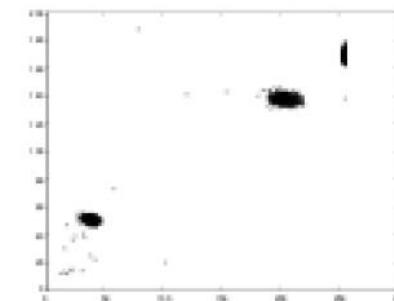
(a)

Specularities on
water surface



(b)

Flicker of monitor

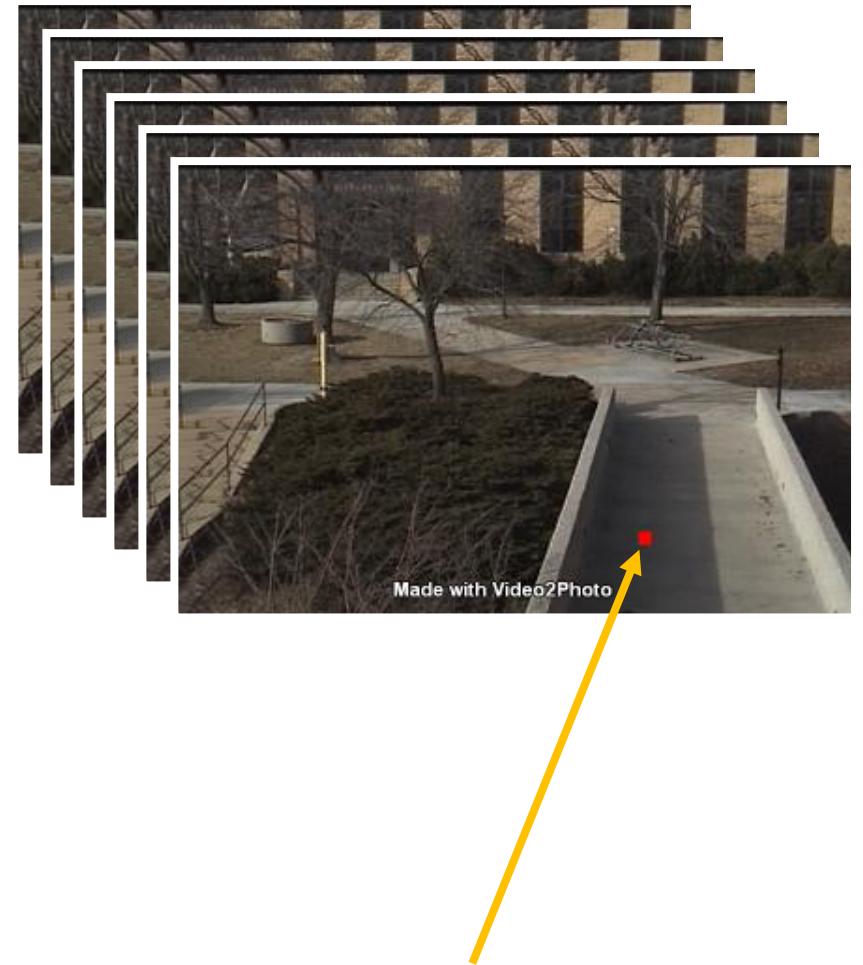


(c)

Fig. Adaptive Background Mixture Models for Real-Time Tracking, Chris Stauer & W.E.L. Grimson

Approach

1. **Model** the values of each pixel as a mixture of Gaussians.
2. **Classify** each pixel as background or foreground.
3. **Group** foreground pixels using connected components and **track** from frame to frame using a multiple hypothesis tracker.
4. **Adapt** the model parameters over time to deal with:
 - Lighting changes
 - Repetitive motions of scene elements (e.g., swaying trees)
 - Slow-moving objects
 - Introducing or removing objects from the scene (i.e., parked cars)

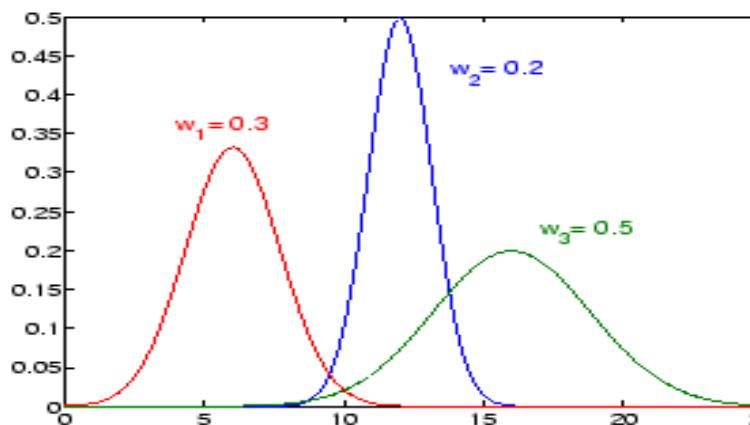


Modeling Pixel Values using Mixtures of Gaussians

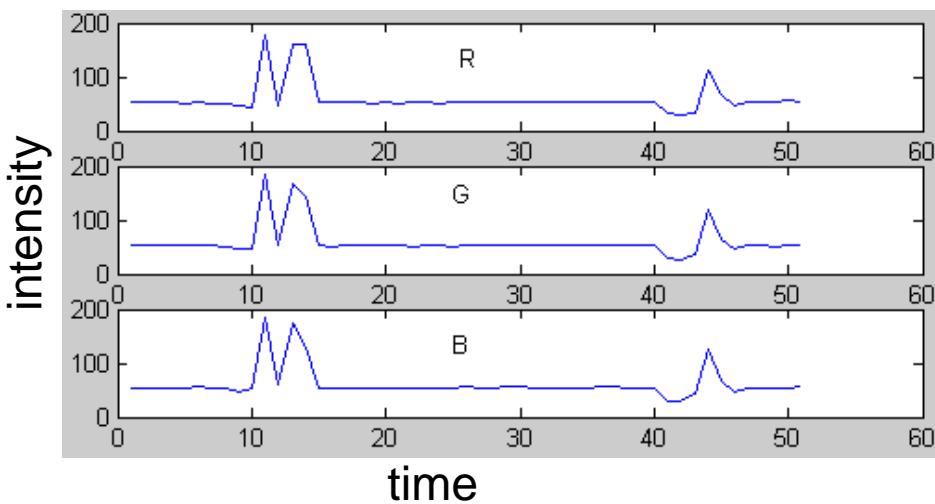
- Consider the values of a particular pixel over time as a "pixel process" (i.e., time series).
- At any time t , we know the history or each pixel at location (x, y) :

$$\{X_1, X_2, \dots, X_t\} = \{I(x, y, i), i = 1, 2, \dots, t\}$$

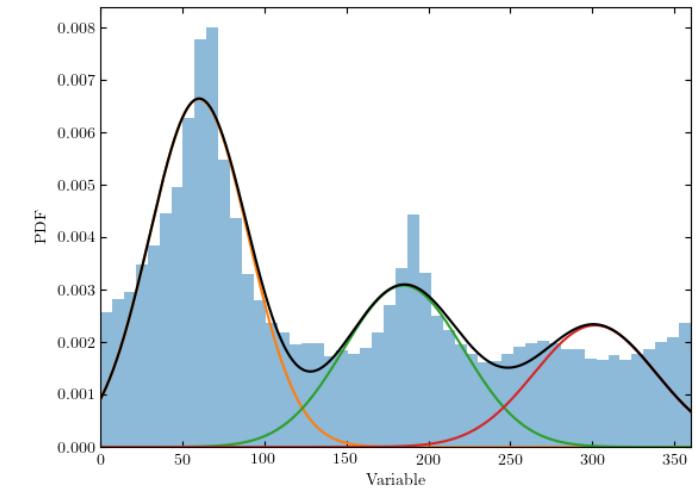
- The recent history of each pixel, $\{X_1, X_2, \dots, X_t\}$, is modeled by a mixture of K Gaussians.



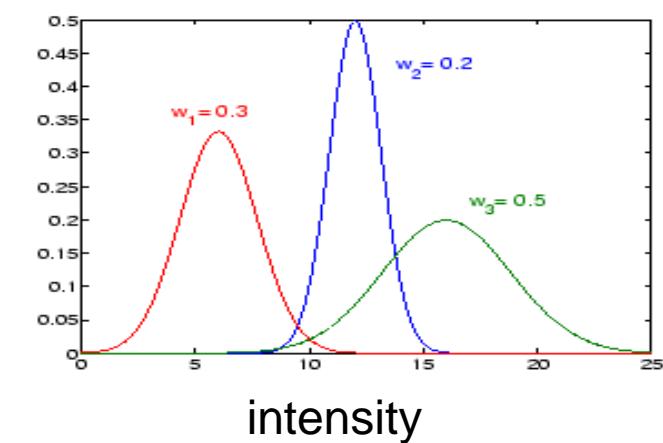
Background Model



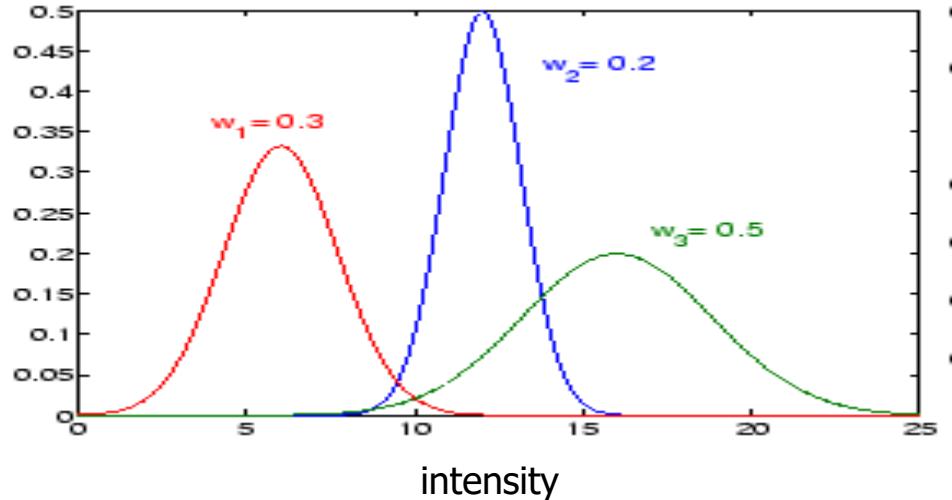
Color history of the specified pixel



Color distribution of the specified pixel



Pixel Model



The probability of observing the current pixel value

$$P(X_t) = \sum_{i=1}^K \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu)^T \Sigma^{-1} (X_t - \mu)}$$

Xt: current pixel value

μ : mean

Σ : standard deviation

n: number of channels/dimensions

K: number of Gaussians

w: weight

Means of the Gaussians

a) Image from a sequence in which a person is walking across the scene.



(a)

Colors that are observed less frequently.



(c)

c) The means of the Gaussian with the second-highest weight.

b) The mean of the highest-weighted Gaussians at each pixels position.



(b)

Most temporally persistent per-pixel color.
Represent the stationary background



(d)

d) Background subtraction result.

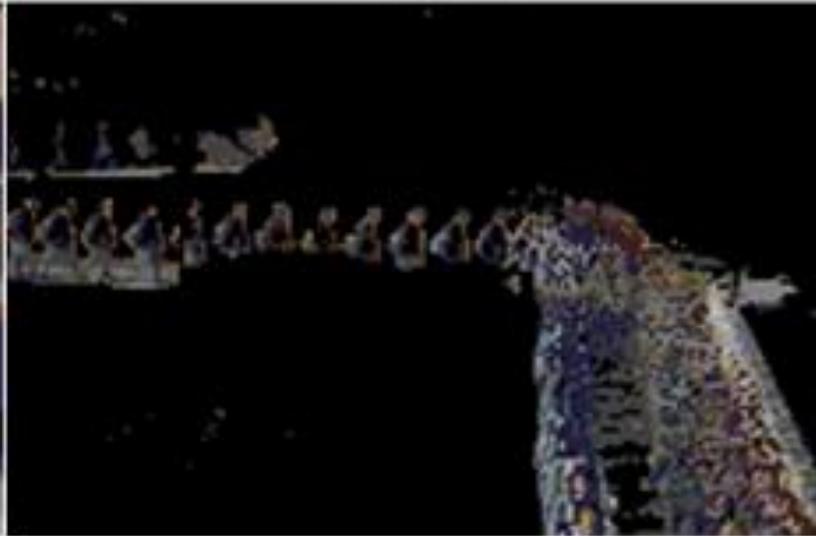
FG=pixels in the current frame that matched a low-weighted Gaussian.

Means of the Gaussians

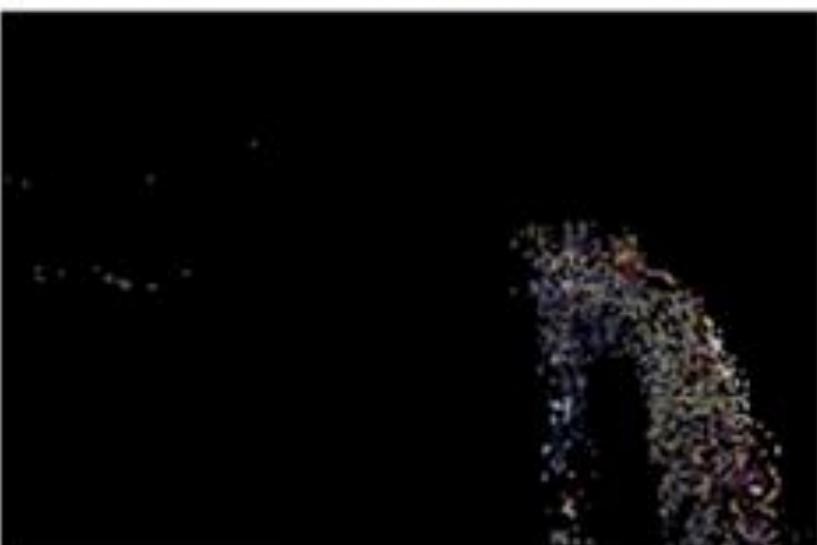
model-1



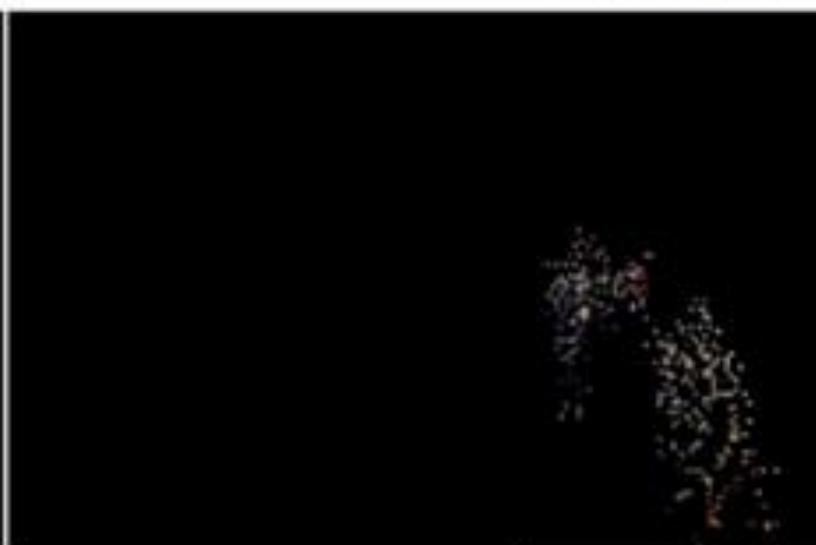
model-2



model-3



model-4



Steps of Mixture of Gaussians Method

1-New Pixel Classification:

Using a K-means clustering, color vector of each new pixel is assigned to a Gaussian distribution.

2-Model Update:

Parameters of the distributions are updated.

3-FG/BG Labeling:

Distributions are labeled as foreground or background based on the ratio w/σ .

This process results in a background model $BG+FG$ and a binary foreground mask FG_{mask} identifying the moving regions/objects.

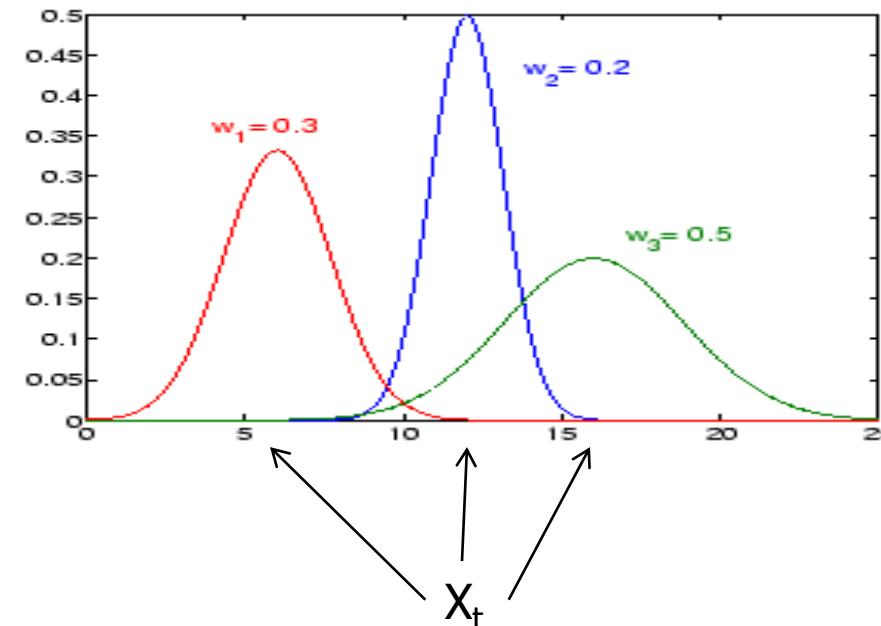
Step 1: New Pixel Classification

Compare the new pixel value X_t against the existing K Gaussians

- X_t is within T_m standard deviations of a distribution → Declare a match
- None of the K distributions match the current pixel value → No match

No Match:

- Replace the least probable distribution.
- Parameters of the new distribution
 - μ = new pixel value
 - σ^2 = initial high variance
 - w = initial low prior weight.



Step 2: Model Update

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$

$M_{k,t}$: match mask

1 for the matched distribution

0 for all the other distributions

The μ and σ parameters for unmatched distributions remain the same.

$$\mu_t = (1 - \rho)\mu_{t-1} + \rho X_t$$

$$\sigma_t^2 = (1 - \rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t)$$

where

$$\rho = \alpha\eta(X_t | \mu_k, \sigma_k)$$

Step 3: Background/Foreground Classification

Goal: Determine which Gaussians are most likely produced by background processes.

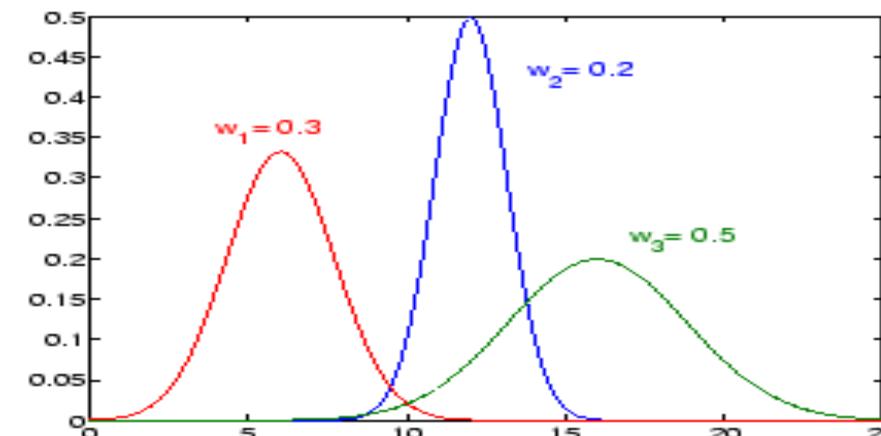
Heuristic: Gaussian distributions with the most supporting evidence (high w and low σ) are considered as produced by background processes the remaining distributions are considered as produced by foreground processes.

Process:

1. Order the Gaussians by the value of w/σ ,
2. Choose the highest B distributions as the background model,

$$B = \operatorname{argmin}_b \left(\sum_{k=1}^b \omega_k > T \right)$$

T is a measure of the minimum portion of the data that should belong to the background.



Background Foreground Labeling

Why w/σ ?

Observations:

- Moving objects are expected to produce more variance than a “static” (background) object - **VARIANCE**
- There should be more data supporting the background distributions because they are repeated, whereas pixel values from different objects are often not the same color - **PERSISTANCE**

Determining the background Gaussians (cont'd)

- To implement this idea, the Gaussians are ordered by the value of w/σ (i.e., w is weight/prior probability).
- The first B distributions are chosen as the background model, where

where

$$B = \operatorname{argmin}_b \left(\sum_{k=1}^b \omega_k > T \right)$$

$$T = (\#background_pixels) / (\#total\ pixels)$$

Datasets

- ChangeDetection.net (For more information: <http://www.changedetection.net/>)
- Background Models Challenge (For more information: <http://bmc.iut-auvergne.com/>)
- Stuttgart Artificial Background Subtraction Dataset (For more information: <http://www.vis.uni-stuttgart.de/index.php?id=sabs>)
- SBMI dataset (For more information: <http://sbmi2015.na.icar.cnr.it/>)
- SBMnet dataset (For more information: <http://pione.dinf.usherbrooke.ca/dataset/>)
- Kalsotra, Rudrika, and Sakshi Arora. "A Comprehensive Survey of Video Datasets for Background Subtraction." *IEEE Access* 7 (2019): 59143-59171.

Some References

- Wang, Yi, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, Yannick Benezeth, and Prakash Ishwar. "CDnet 2014: an expanded change detection benchmark dataset." In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 387-394. 2014.
- Wang, Rui, Filiz Bunyak, Guna Seetharaman, and Kannappan Palaniappan. "Static and moving object detection using flux tensor with split gaussian models." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 414-418. 2014.
- FTSG demo: https://www.youtube.com/watch?v=QRhlBo_YJdE

Dense Motion Estimation

Optical Flow Estimation and Analysis
Szeliski Book Chapter 9

Why estimate visual motion?

1. **Dynamics:** Visual Motion indicates dynamics in the scene
 - Moving objects, behavior
 - Track objects and analyze trajectories
2. **Quality:** Visual Motion can be annoying -- improve video quality
 - Camera instabilities, jitter
 - Measure it; remove it (stabilize)
3. **Spatial Structure:** Visual Motion reveals spatial layout and 3D structure

Approaches

1. **Optical Flow Analysis:** Characteristics of flow (velocity) vectors of moving objects over time are used to detect changed regions.
 - **Advantage:** can be used in the presence of camera motion.
 - **Disadvantage:** computationally expensive.
2. **Temporal differencing:** Similar to background subtraction but the estimated background is the previous frame.
$$| \text{frame}_i - \text{frame}_{i-1} | > \text{Th}$$
 - **Advantage:** very adaptive to dynamic environments.
 - **Disadvantage:** has problems in extraction of all relevant feature pixels (aperture problem).
3. **Background subtraction:** Moving regions are detected through difference between the current frame and a reference background image.
$$| \text{frame}_i - \text{Background}_i | > \text{Th}$$
 - **Advantage:** provides the most complete feature data.
 - **Disadvantage:** sensitive to dynamic scene changes due to lighting and extraneous events.

Optical Flow based Moving Object Detection

Step 1: Compute optical flow vectors then

Step 2: Group the flow vectors that belong to the same motion to identify moving objects.

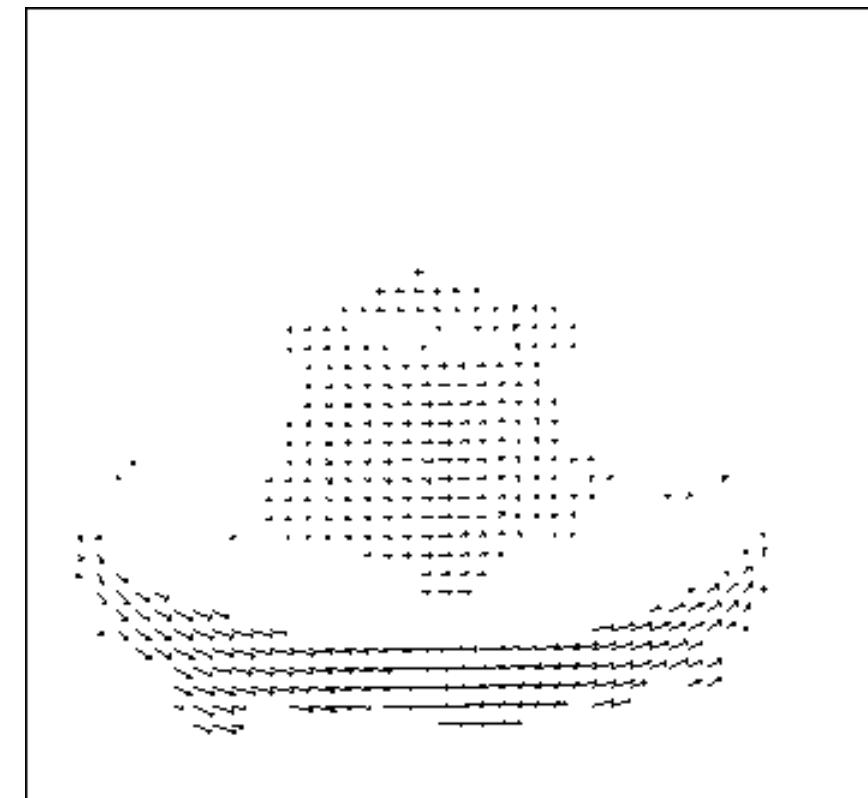
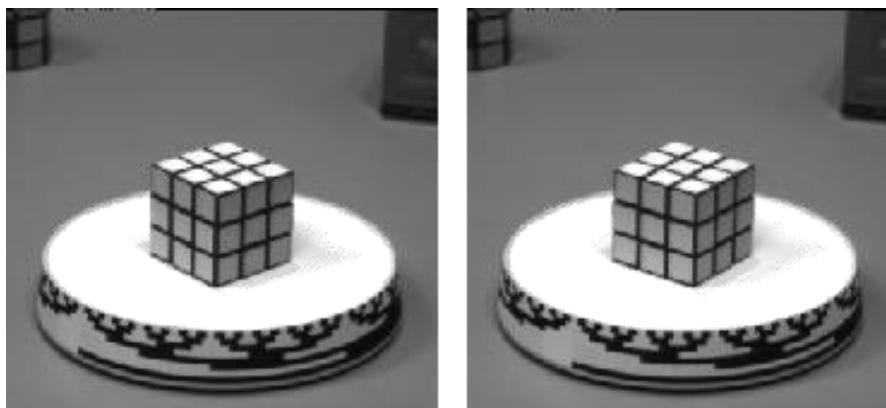
Advantage: can be used even with non-stationary cameras.

Disadvantage:

- Reliable motion field computation under real-world conditions is challenging and computationally expensive
- Can not deal with stopped objects

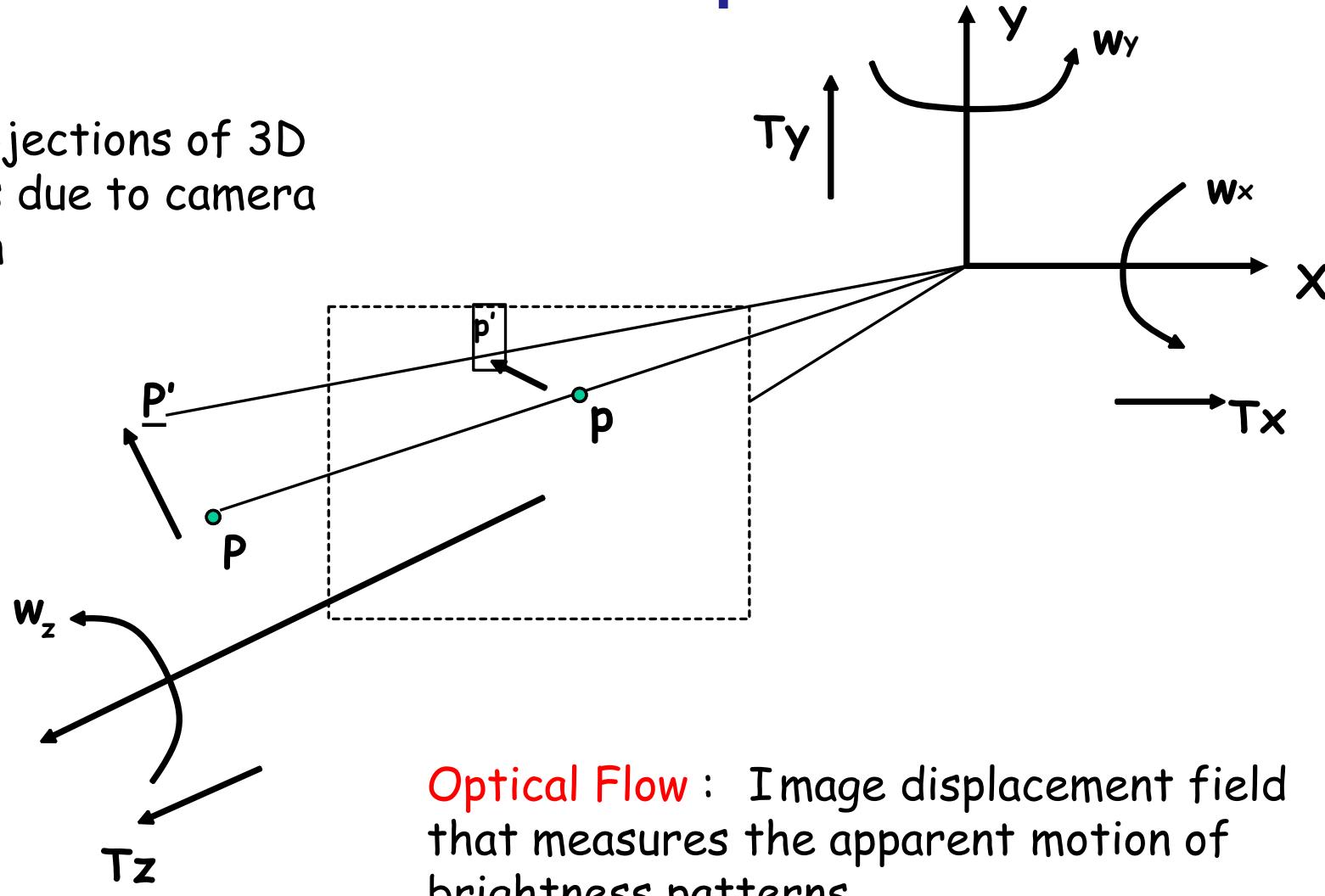
Motion field

- The motion field is the projection of the 3D scene motion into the image



Motion Field & Optical Flow

Motion Field : 2D projections of 3D displacement vectors due to camera and/or object motion



Optical Flow : Image displacement field that measures the apparent motion of brightness patterns

Optical flow

Optical flow is the *apparent motion* of brightness patterns in the image

Ideally: optical flow = motion field



Hamburg taxi sequence

Regularization-based optical flow
(Nagel and Enkelmann 1986)

However: apparent motion can be caused by lighting changes without any actual motion

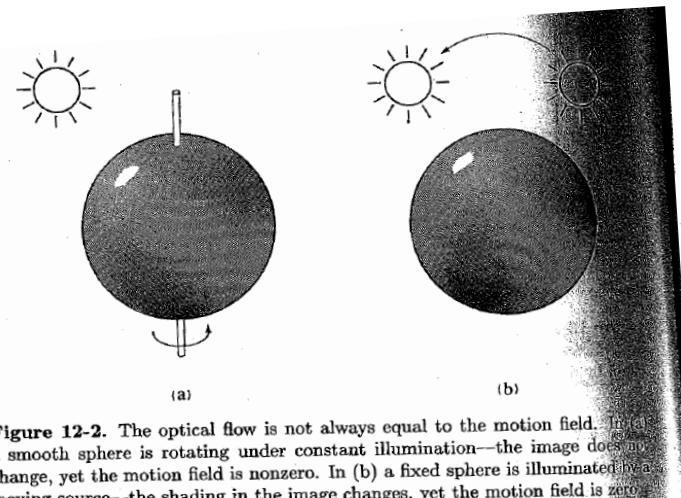
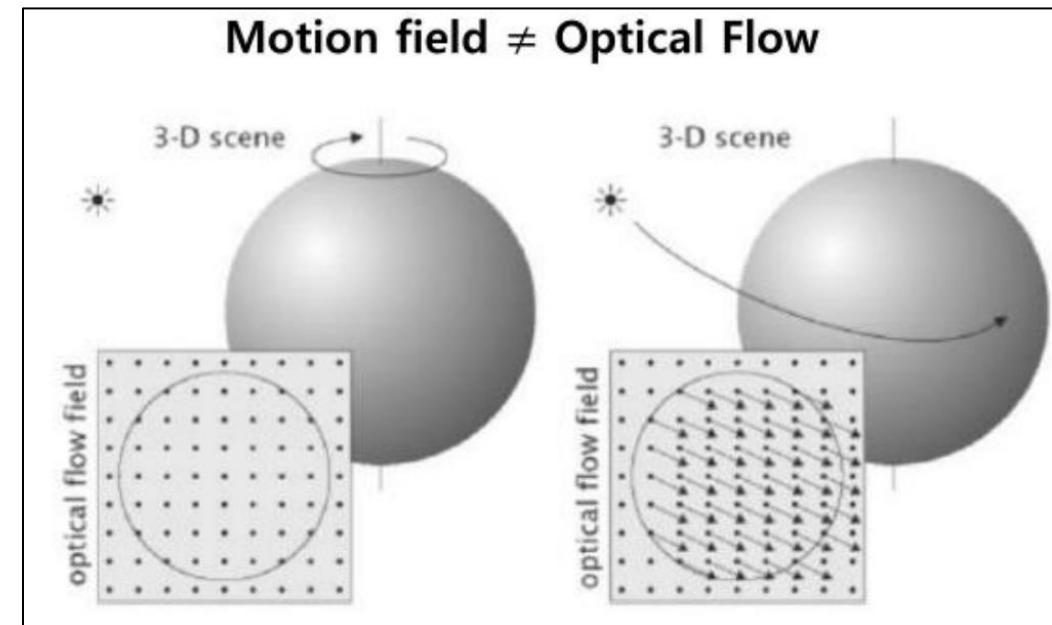
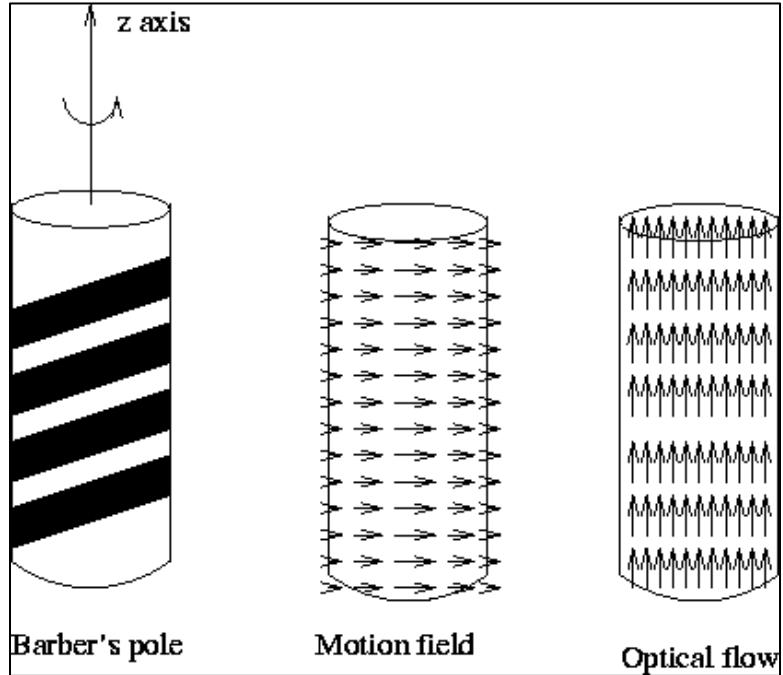


Figure 12-2. The optical flow is not always equal to the motion field. In (a), a smooth sphere is rotating under constant illumination—the image does not change, yet the motion field is nonzero. In (b) a fixed sphere is illuminated by a moving source—the shading in the image changes, yet the motion field is zero.

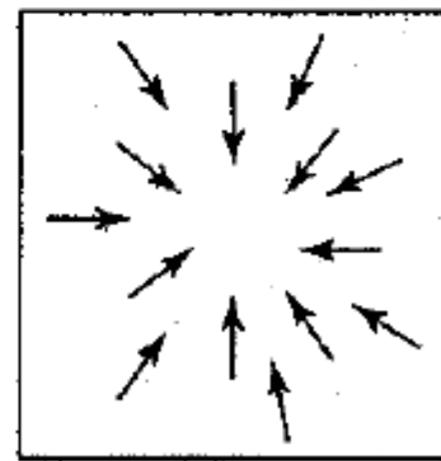
Figure from Horn book

Motion Field vs. Optical Flow

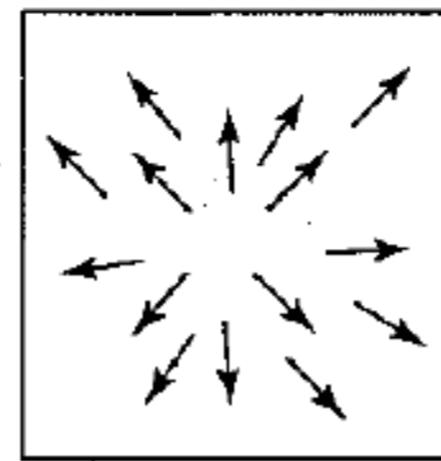


https://en.wikipedia.org/wiki/Barberpole_illusion

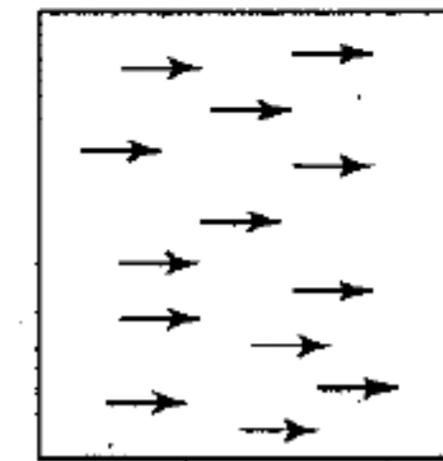
Typical Motion Fields



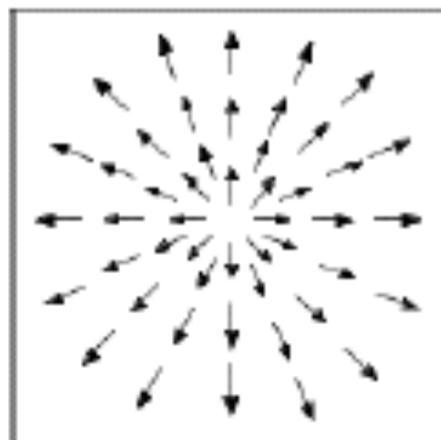
Zoom out



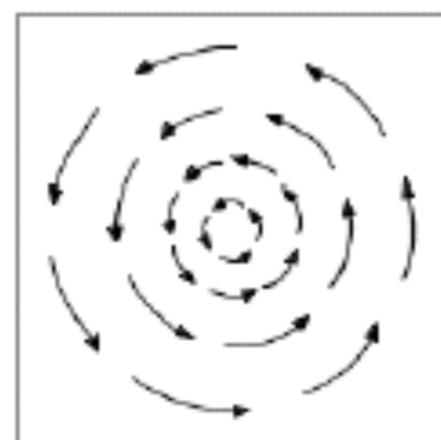
Zoom in



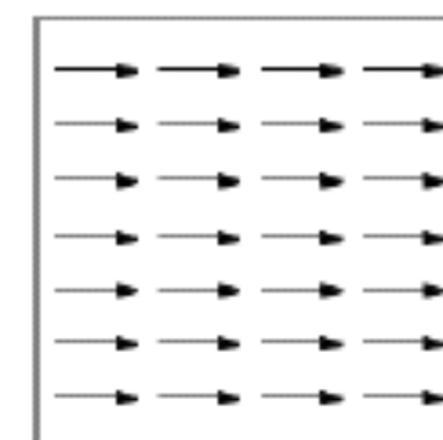
Pan right to left



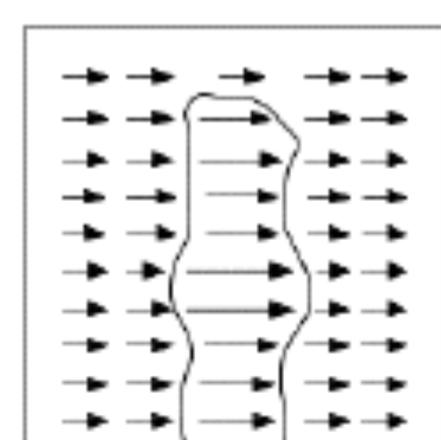
Forward motion



Rotation



Horizontal translation

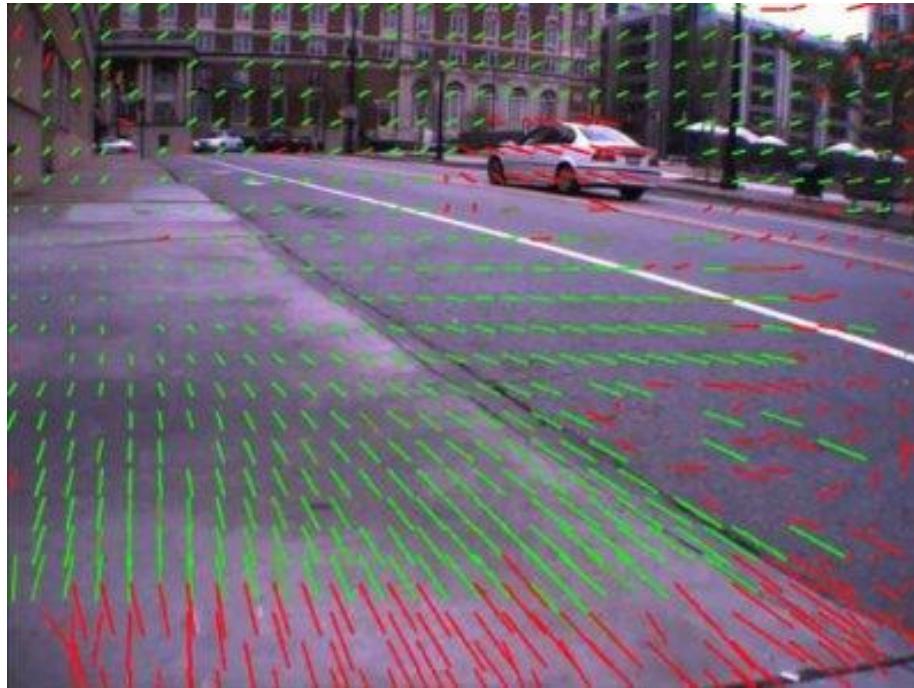


Closer objects appear to move faster!!

Motion estimation techniques

- Feature-based methods
 - 1-Extract visual features (corners, textured areas) and
 - 2-Track them over multiple frames
 - Sparse motion fields, but more robust tracking
 - Suitable when image motion is large (10s of pixels)
- Direct methods
 - 1-Directly recover image motion at each pixel from spatio-temporal image brightness variations
 - Dense motion fields, but sensitive to appearance variations
 - Suitable for video and when image motion is small

Sparse Correspondences vs Dense Optical Flow



2D Flow Vectors



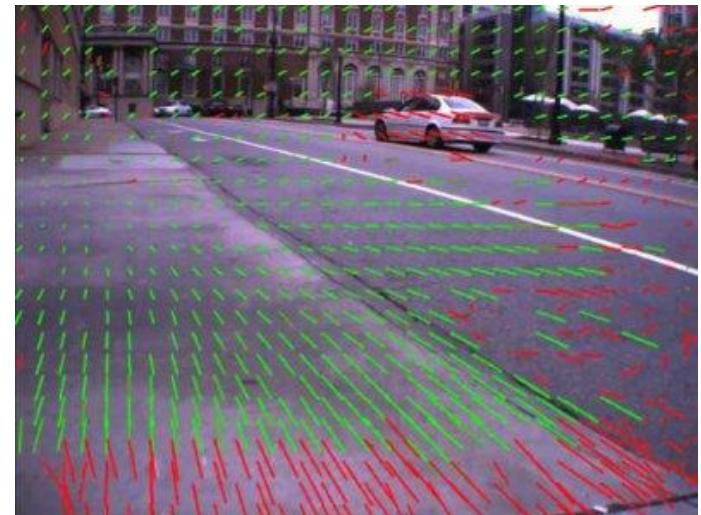
Hue (Angle) – Saturation (Length) Visualization

Slide credit: Harpreet S Sawhney

Motion estimation techniques

- **Feature-based methods**

- 1-Extract visual features (corners, textured areas) and
- 2-Track them over multiple frames
 - Sparse motion fields, but more robust tracking
 - Suitable when image motion is large (10s of pixels)



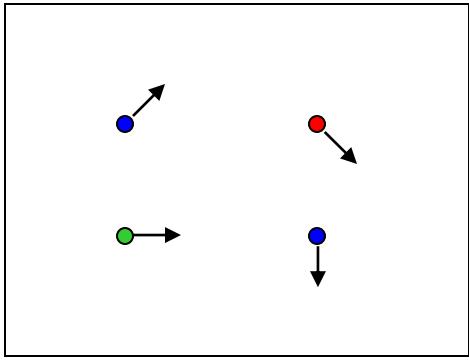
What are good features to track?

- Recall the Harris corner detector
- Can measure quality of features from just a single image
- Automatically select candidate “templates”

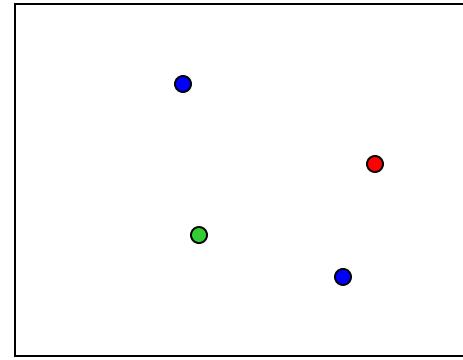
Motion estimation techniques

- Feature-based methods
 - 1-Extract visual features (corners, textured areas) and
 - 2-Track them over multiple frames
 - Sparse motion fields, but more robust tracking
 - Suitable when image motion is large (10s of pixels)
- Direct methods
 - 1-Directly recover image motion at each pixel from spatio-temporal image brightness variations
 - Dense motion fields, but sensitive to appearance variations
 - Suitable for video and when image motion is small

Problem definition: optical flow



$H(x, y)$



$I(x, y)$

How to estimate pixel motion from image H to image I ?

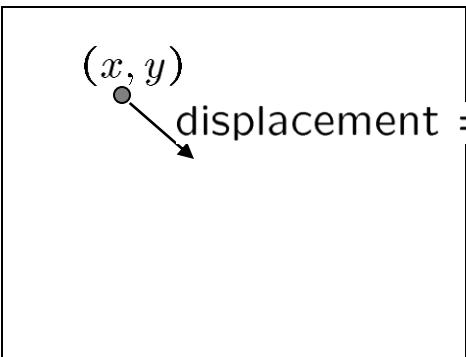
- Solve pixel correspondence problem
 - given a pixel in H , look for **nearby** pixels of the **same color** in I

Key assumptions

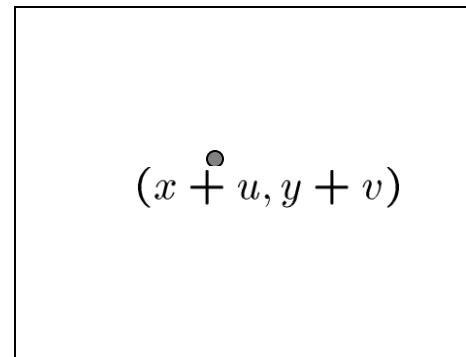
- **Color/Brightness constancy**: a point in H looks the same in I
- **Small motion**: points do not move very far

This is called the **optical flow** problem

Optical flow constraints (grayscale images)



$$H(x, y)$$



$$I(x, y)$$

Let's look at these constraints more closely

- brightness constancy: Q: what's the equation?

- small motion: (u and v are less than 1 pixel)

- suppose we take the Taylor series expansion of I :

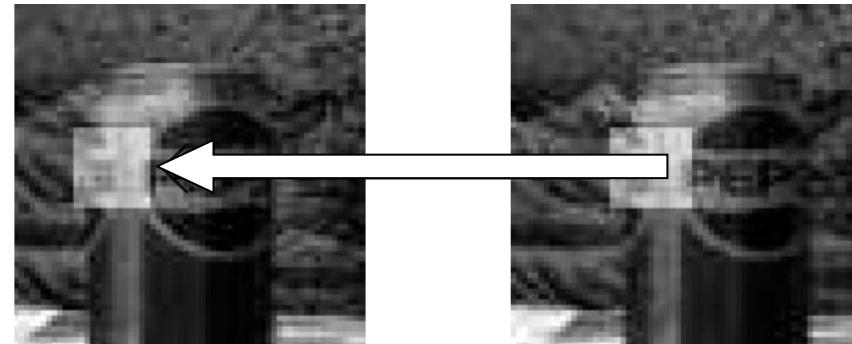
$$\begin{aligned} I(x+u, y+v) &= I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms} \\ &\approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \end{aligned}$$

Differential Techniques

Goal:

Compute the 2-D motion field
from spatiotemporal patterns of
image intensities.

Assumption: Intensity is conserved.



Brightness Constraint

$$I(x, y, t - 1) = I(x + u(x, y), y + v(x, y), t)$$

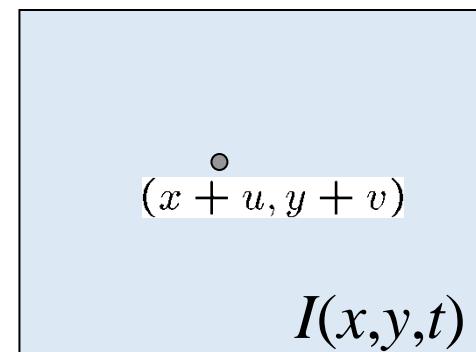
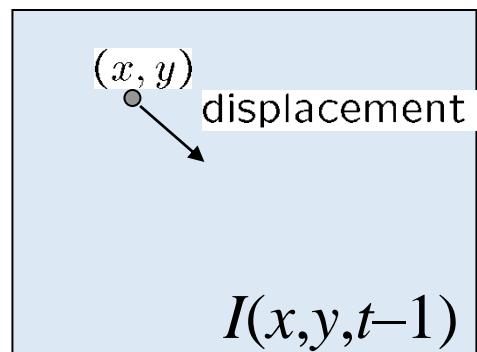
Taylor series expansion of I:

$$I(x + u, y + v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

Brightness Constraint

$$\frac{dI(x,y,t)}{dt} = 0$$

$$\frac{dI(x,y,t)}{dt} = \frac{\partial I}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial I}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t}$$

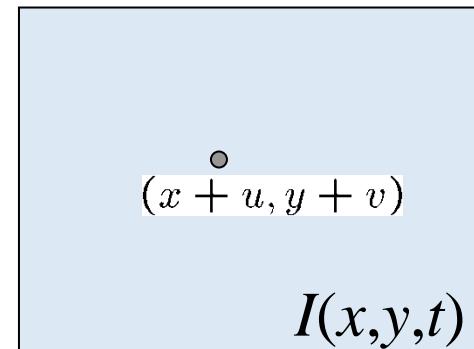
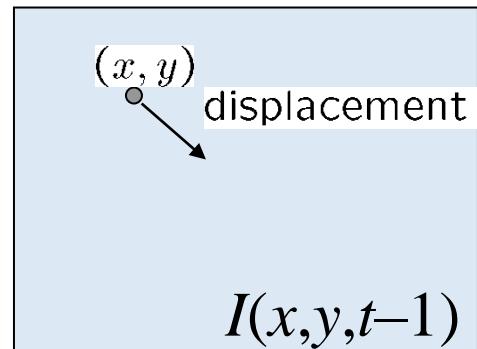


Brightness Constraint

$$\frac{dI(x,y,t)}{dt} = 0$$

$$\frac{dI(x,y,t)}{dt} = \frac{\partial I}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial I}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$$

u *v*

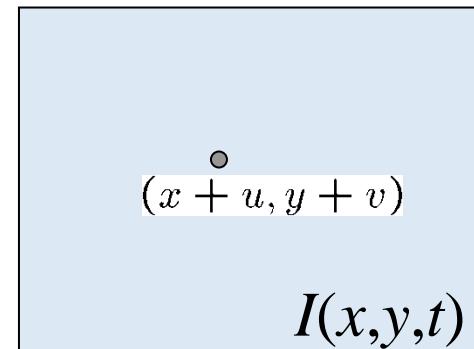
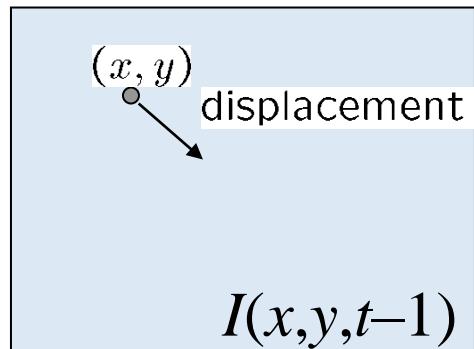


Brightness Constraint

$$\frac{dI(x,y,t)}{dt} = 0$$

$$\frac{dI(x,y,t)}{dt} = \frac{\partial I}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial I}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$$

$$I_x \ u \quad I_y \ v \quad I_t$$



Brightness Constraint

$$\frac{dI(x,y,t)}{dt} = 0$$

$$\frac{dI(x,y,t)}{dt} = \frac{\partial I}{\partial x} \frac{\partial x}{\partial t} + \frac{\partial I}{\partial y} \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$$

$$I_x \ u \quad I_y \ v \quad I_t$$

$$I_x u + I_y v + I_t = 0$$

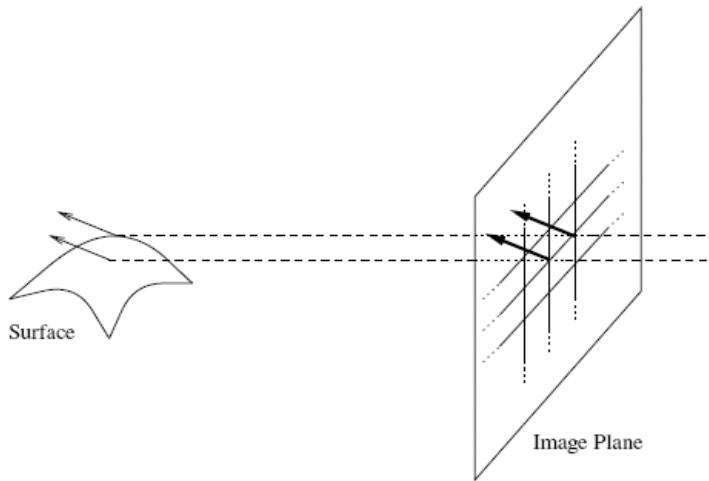
What are knowns ?
What are unknowns?
How many equations?

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -[I_t]$$

Additional Constraints

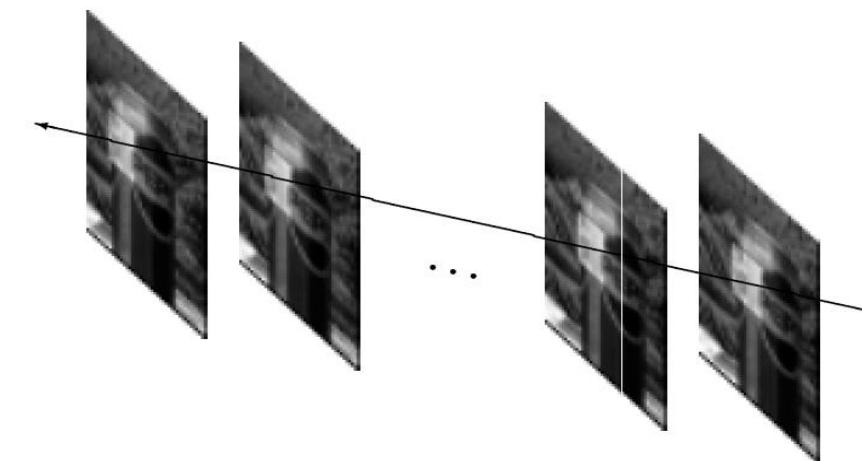
Spatial Coherence

Assumption: Neighboring points in the scene belong to the same surface and hence typically have similar motions.



Temporal Persistence

Assumption: Image motion of a surface patch changes gradually over time.



Picture from: M. Black. *Robust incremental optical flow*.
PhD thesis, Yale University, 1992.

Solving the aperture problem (grayscale image)

$$\begin{bmatrix} I_x(p_i) & I_y(p_i) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -[I_t(p_i)]$$

How to get more equations for a pixel?

Spatial coherence constraint: pretend the pixel's neighbors have the same (u,v)

- If we use a 5x5 window, that gives us 25 equations per pixel

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$
$$A \quad d = b$$
$$25 \times 2 \quad 2 \times 1 \quad 25 \times 1$$

Lucas & Kanade (Least Squares)

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

$$\begin{array}{ccc} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{array}$$

$$A^{-1}A d = A^{-1}\mathbf{b}$$

$$d = A^{-1}\mathbf{b}$$

Lucas & Kanade (Least Squares)

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

~~$A^{-1}A d = A^{-1}b$~~
 ~~$d = A^{-1}b$~~

Pseudo Inverse →

$$A^T A d = A^T b$$

25x2 2x1 25x1
2x25 25x2 2x1 2x25 25x1
2x2 2x1 2x1

$$A^T A d = A^T b$$
$$d = (A^T A)^{-1} A^T b$$

Lucas & Kanade (Least Squares)

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix}$$

$$\mathbf{A}^T \mathbf{A} \mathbf{d} = \mathbf{A}^T \mathbf{b}$$

Pseudo Inverse \rightarrow $\mathbf{d} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$A^T A \qquad \qquad \qquad A^T b$$

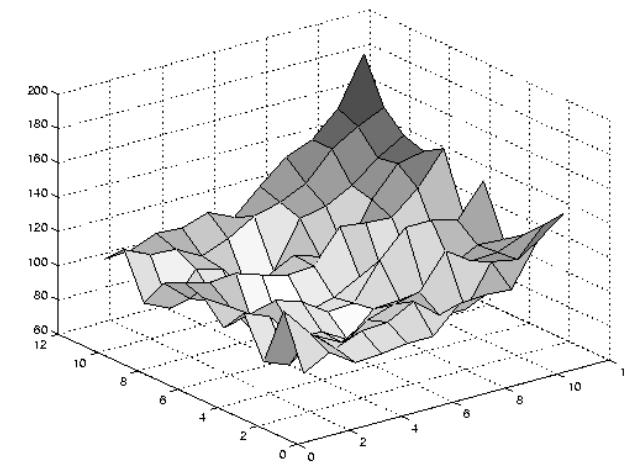
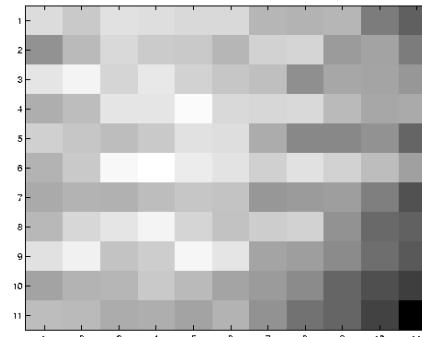
Conditions for solvability

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$A^T A \qquad \qquad \qquad A^T b$$

When is this solvable?

- $A^T A$ should be invertible
- $A^T A$ should not be too small
 - eigenvalues λ_1 and λ_2 of $A^T A$ should not be too small
- $A^T A$ should be well-conditioned
 - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

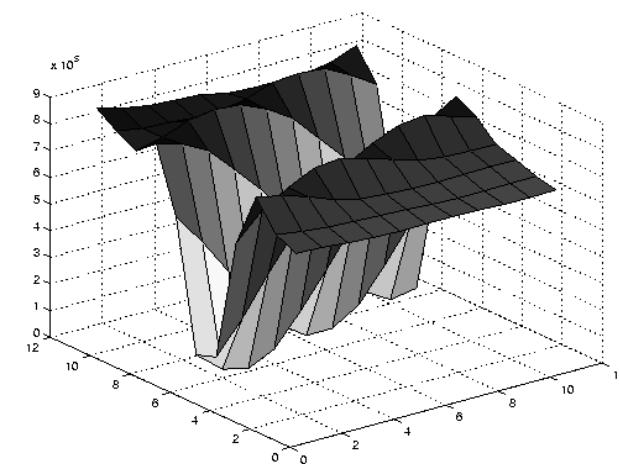
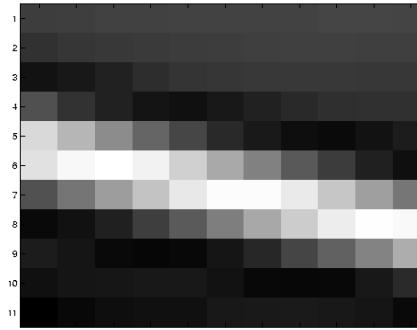
Low-texture region



When is this solvable?

- $\mathbf{A}^T \mathbf{A}$ should not be too small
 - eigenvalues λ_1 and λ_2 of $\mathbf{A}^T \mathbf{A}$ should not be too small
 - gradients have small magnitude
 - small λ_1 , small λ_2

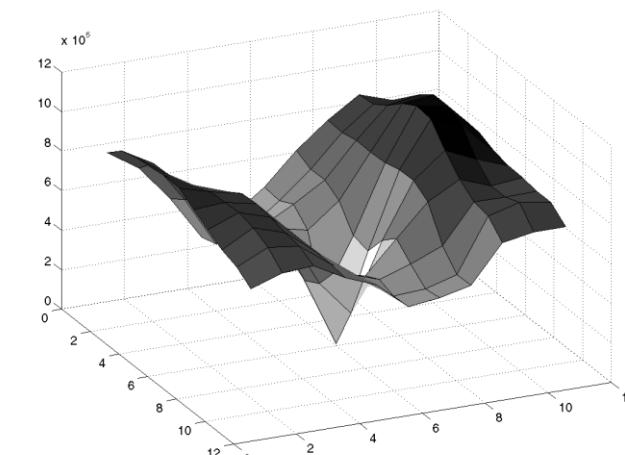
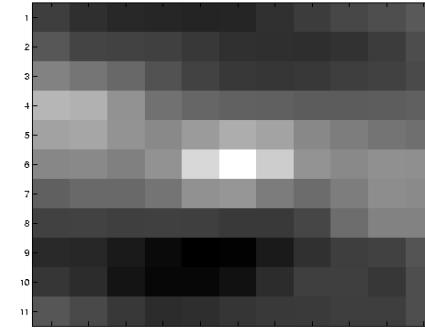
Edge



When is this solvable?

- $\mathbf{A}^T \mathbf{A}$ should be well-conditioned
 - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)
 - gradients very large or very small
 - large λ_1 , small λ_2

High-texture region

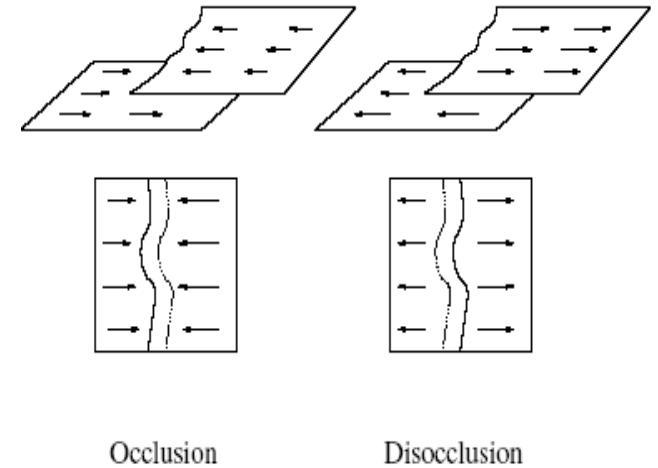


- gradients are different,
- large λ_1 , large λ_2

Problems in Optical Flow Estimation

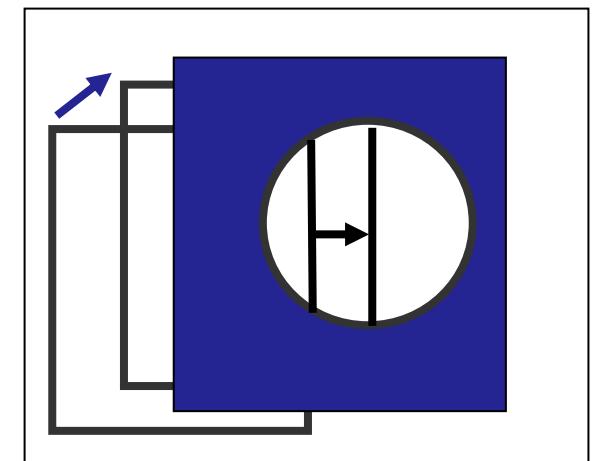
Optical flow estimation is an ill-posed problem.

1. Existence of a solution (occlusion problem): No correspondence can be established for covered/uncovered background points.
2. Uniqueness of the solution (aperture problem): If the components of the velocity (or displacement) is treated as independent variables, the number of unknowns is twice the number of observations (elements of the frame difference).
3. Continuity of the solution: Motion estimation is highly sensitive to the presence of observation noise in video images.



Occlusion

Disocclusion



Lucas & Kanade (Least Squares)

$$\mathbf{A}^T \mathbf{A} \mathbf{d} = \mathbf{A}^T \mathbf{b}$$

Pseudo Inverse \rightarrow $\mathbf{d} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$
$$A^T A \qquad \qquad \qquad A^T b$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\sum I_x^2 \sum I_y^2 - (\sum I_x I_y)^2} \begin{bmatrix} \sum I_y^2 & -\sum I_x I_y \\ -\sum I_x I_y & \sum I_x^2 \end{bmatrix} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

Lucas & Kanade (Least Squares)

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{\Sigma I_x^2 \Sigma I_y^2 - (\Sigma I_x I_y)^2} \begin{bmatrix} \Sigma I_y^2 & -\Sigma I_x I_y \\ -\Sigma I_x I_y & \Sigma I_x^2 \end{bmatrix} \begin{bmatrix} -\Sigma I_x I_t \\ -\Sigma I_y I_t \end{bmatrix}$$

$$u = \frac{-\Sigma I_y^2 \Sigma I_x I_t + \Sigma I_x I_y \Sigma I_y I_t}{\Sigma I_x^2 \Sigma I_y^2 - (\Sigma I_x I_y)^2}$$

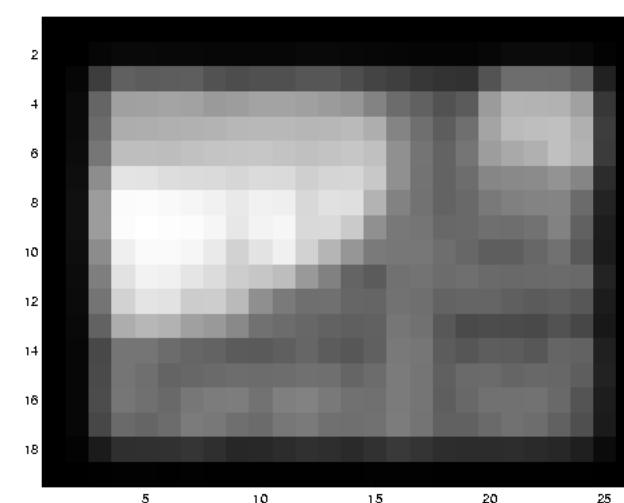
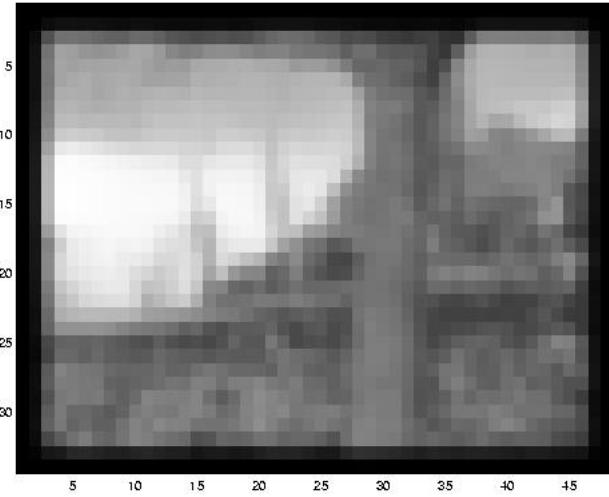
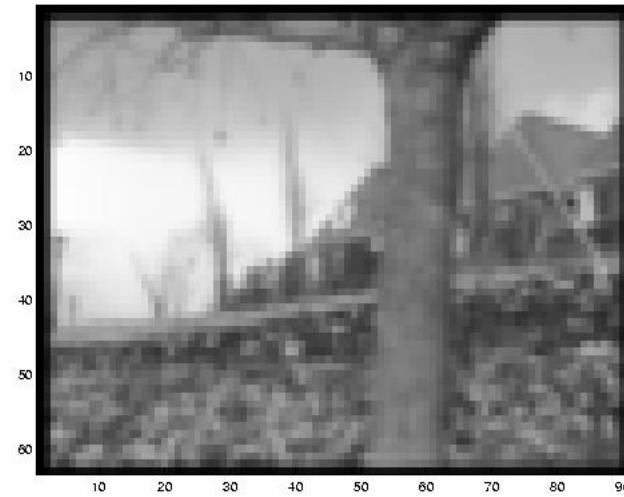
$$v = \frac{\Sigma I_x I_t \Sigma I_x I_y - \Sigma I_x I_y^2 \Sigma I_y I_t}{\Sigma I_x^2 \Sigma I_y^2 - (\Sigma I_x I_y)^2}$$

Revisiting the small motion assumption

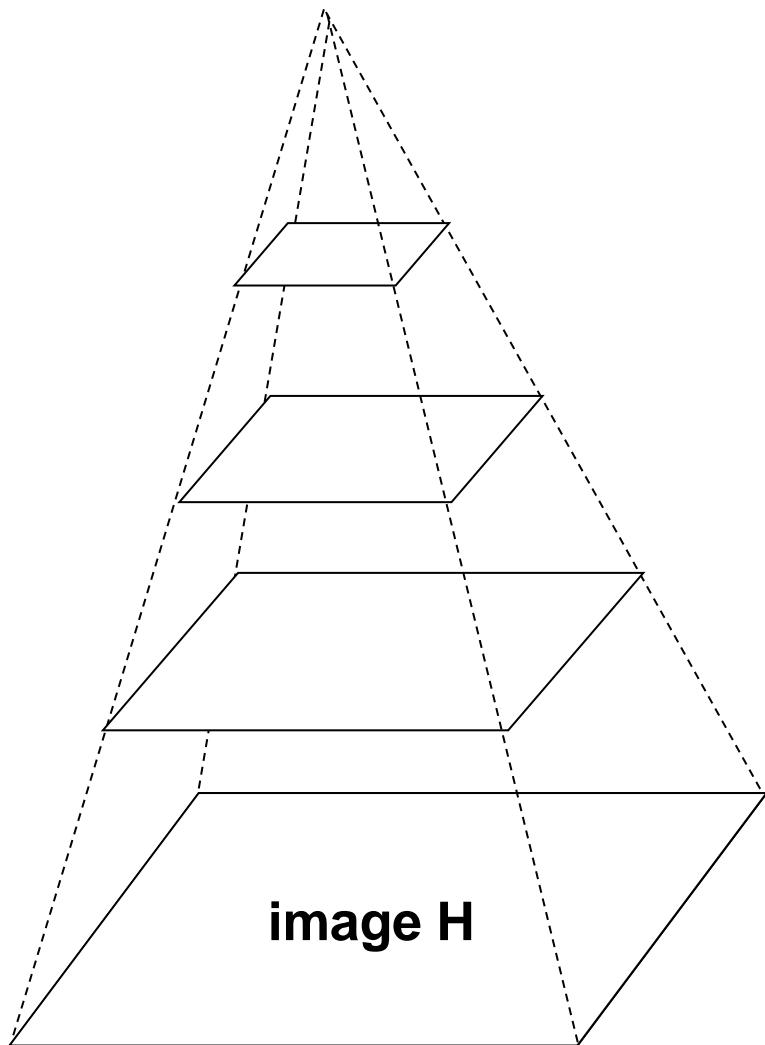


- Is this motion small enough?
 - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)
 - How might we solve this problem?

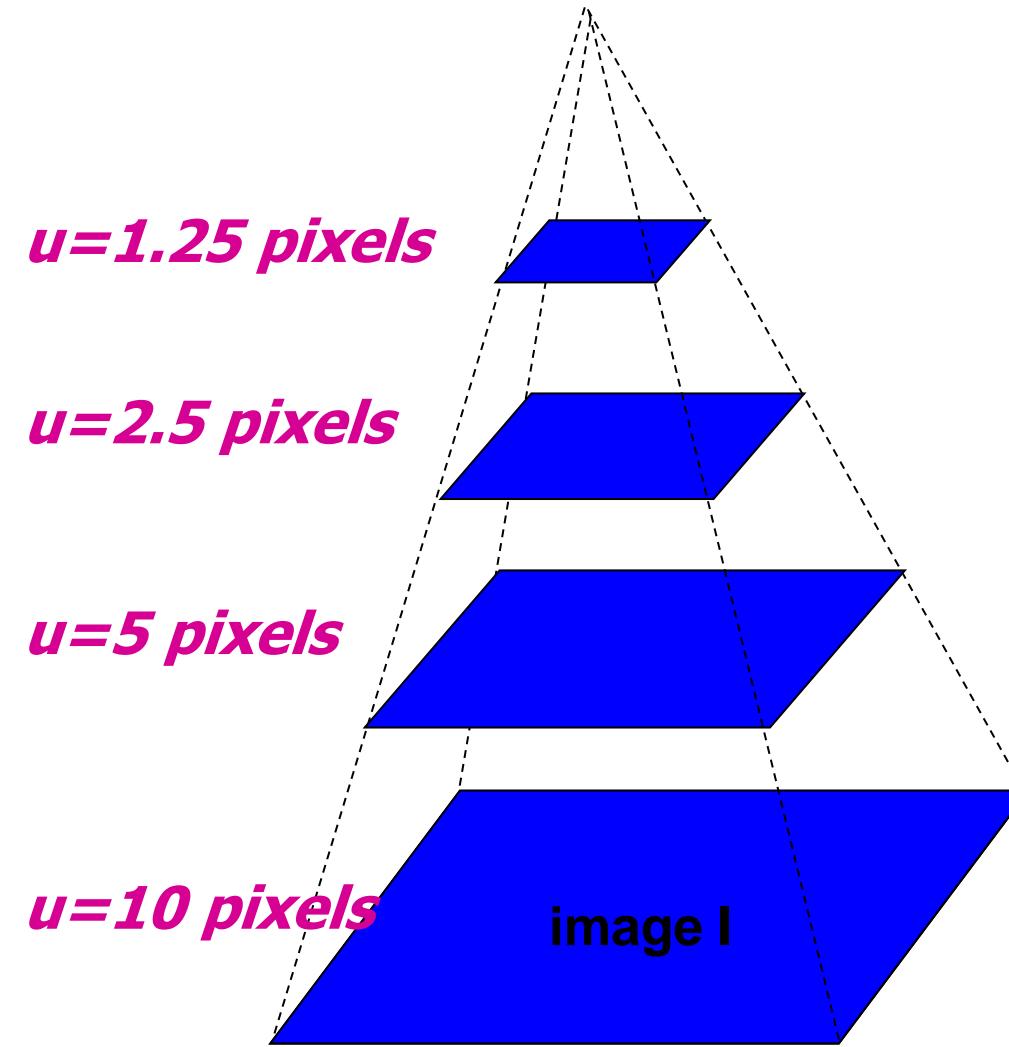
Reduce the resolution!



Coarse-to-fine optical flow estimation

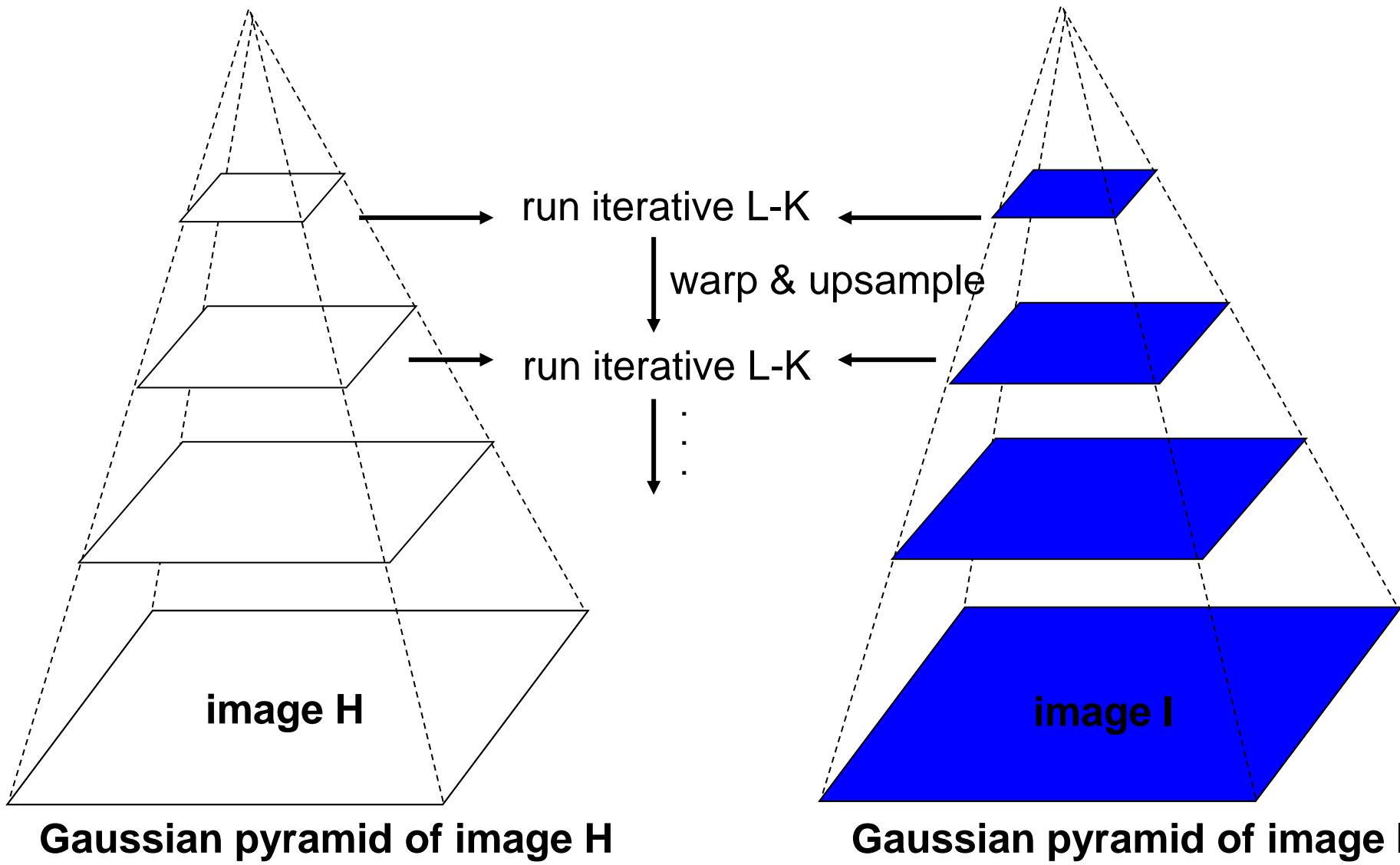


Gaussian pyramid of image H



Gaussian pyramid of image I

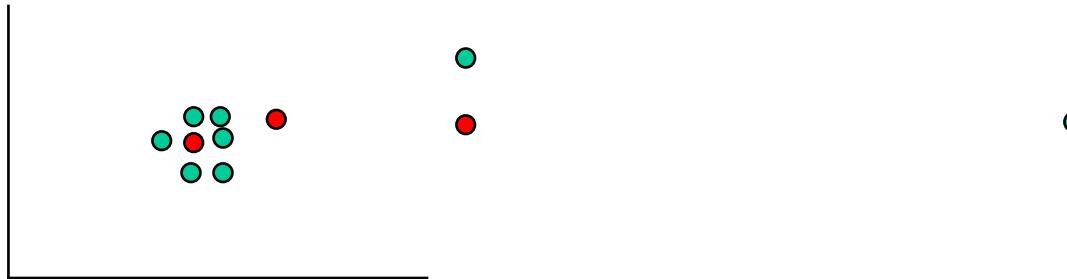
Coarse-to-fine optical flow estimation



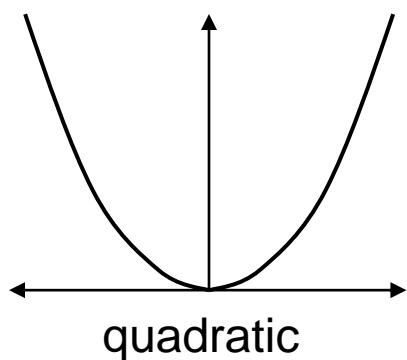
Robust methods

L-K minimizes a sum-of-squares error metric

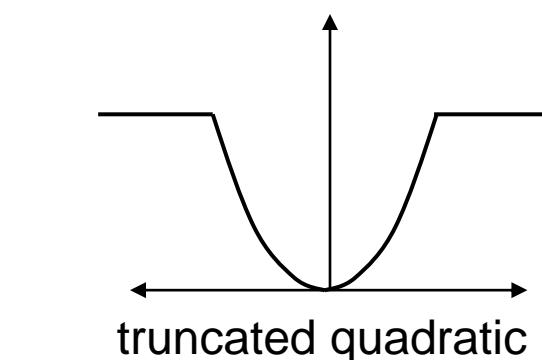
- least squares techniques overly sensitive to outliers



Error metrics

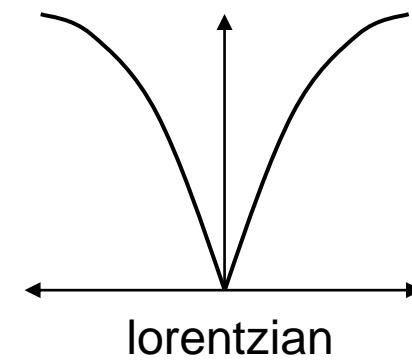


quadratic



truncated quadratic

$$\rho_{\alpha,\lambda}(x) = \begin{cases} \lambda x^2 & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}} \\ \alpha & \text{otherwise} \end{cases}$$



lorentzian

$$\rho_\sigma(x) = \log \left(1 + \frac{1}{2} \left(\frac{x}{\sigma} \right)^2 \right)$$

Robust optical flow

Robust Horn & Schunk

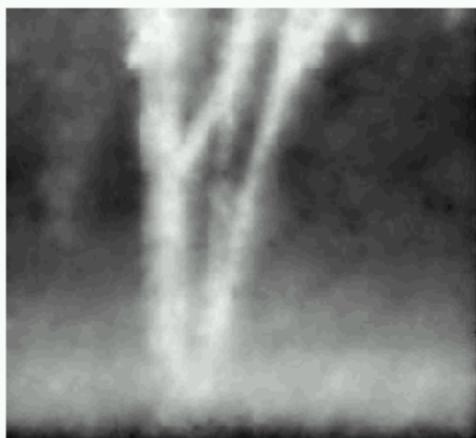
$$\int \int \rho(I_t + \nabla I \cdot [u \ v]) + \lambda^2 \rho(\|\nabla u\|^2 + \|\nabla v\|^2) \ dx \ dy$$

Robust Lucas-Kanade

$$\sum_{(x,y) \in W} \rho(I_t + \nabla I \cdot [u \ v])$$



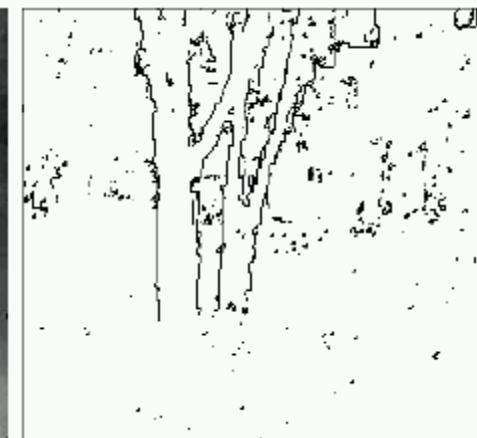
first image



quadratic flow



lorentzian flow



detected outliers

Reference

- Black, M. J. and Anandan, P., A framework for the robust estimation of optical flow, *Fourth International Conf. on Computer Vision (ICCV)*, 1993, pp. 231-236
<http://www.cs.washington.edu/education/courses/576/03sp/readings/black93.pdf>

Optical Flow VFX:

PAINTING THE AFTERLIFE IN **WHAT DREAMS MAY COME**



The final shot was enabled with extensive development of tracking techniques, optical flow and a specialized particles tool to produce the painterly effects.

Slide Credit: Szeliski

Automatic Extraction of 2D Layers



Input Sequence

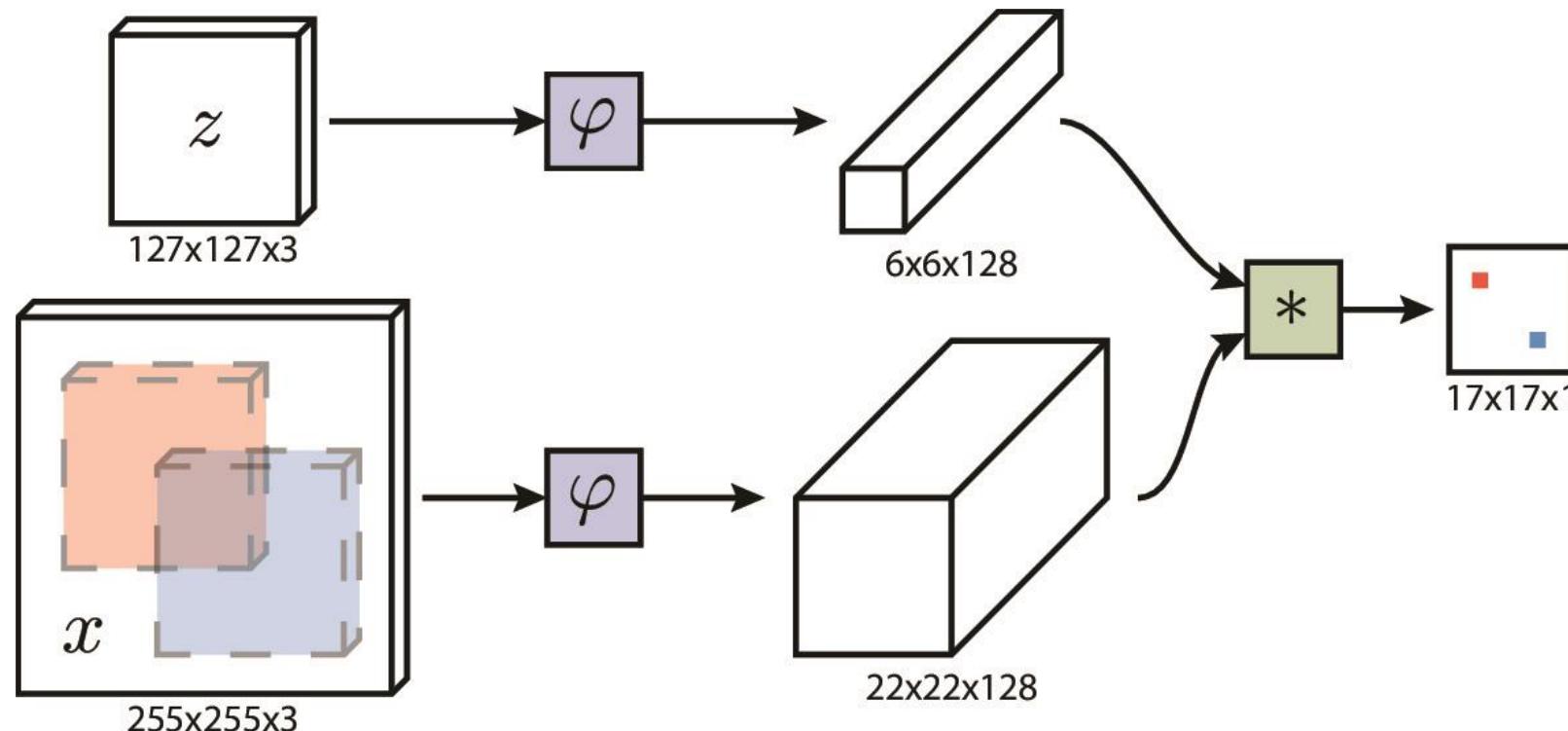


Layers

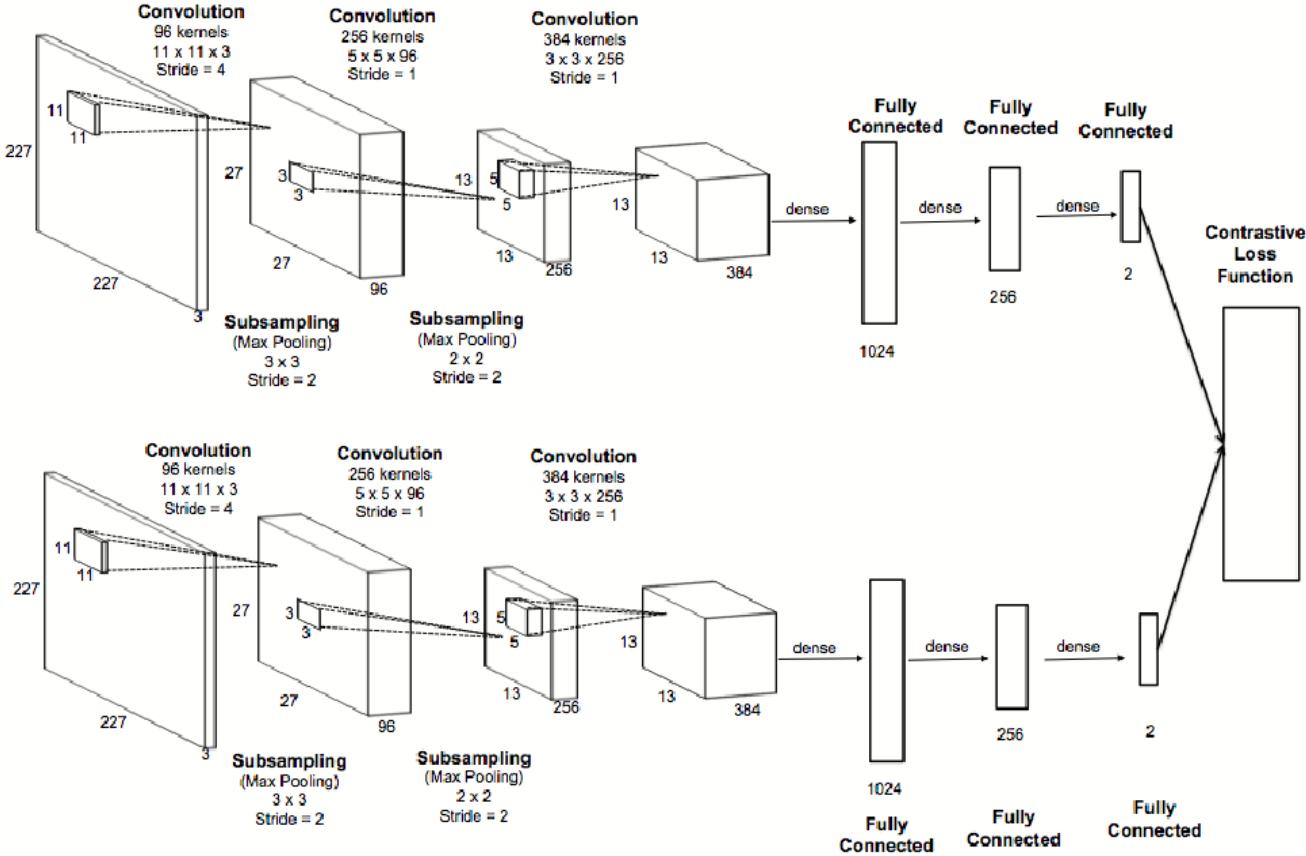
DEEP LEARNING APPROACHES

Deep Learning-based Optical Flow Estimation

- While optical flow estimation needs precise per-pixel localization, it also requires finding correspondences between two input images.
- This involves not only learning image feature representations, but also learning to match them at different locations in the two images.
- In this respect, optical flow estimation fundamentally differs from previous applications of CNNs.



Siamese Network

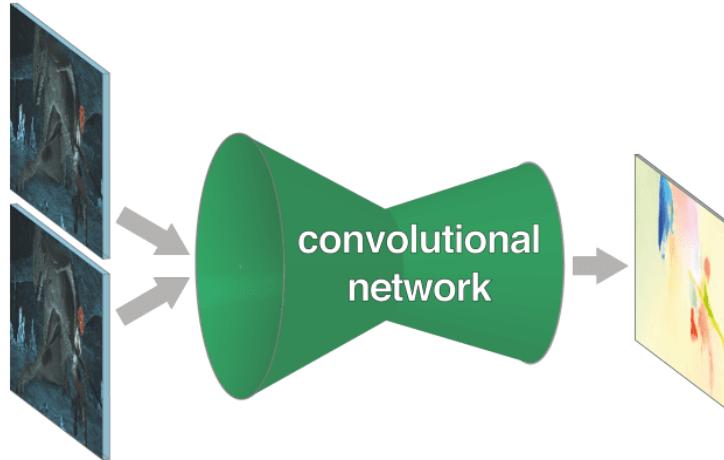


$$Y * D^2 + (1 - Y) * \max(\text{margin} - D, 0)^2$$

Contrastive loss: how good a job the siamese network is distinguishing between the image pairs.
 Y: label.

- 1: image pairs are same class,
 - 0: image pairs are of a different class.
- The $D_{\{w\}}$: Euclidean distance between the outputs of the sister network embeddings.

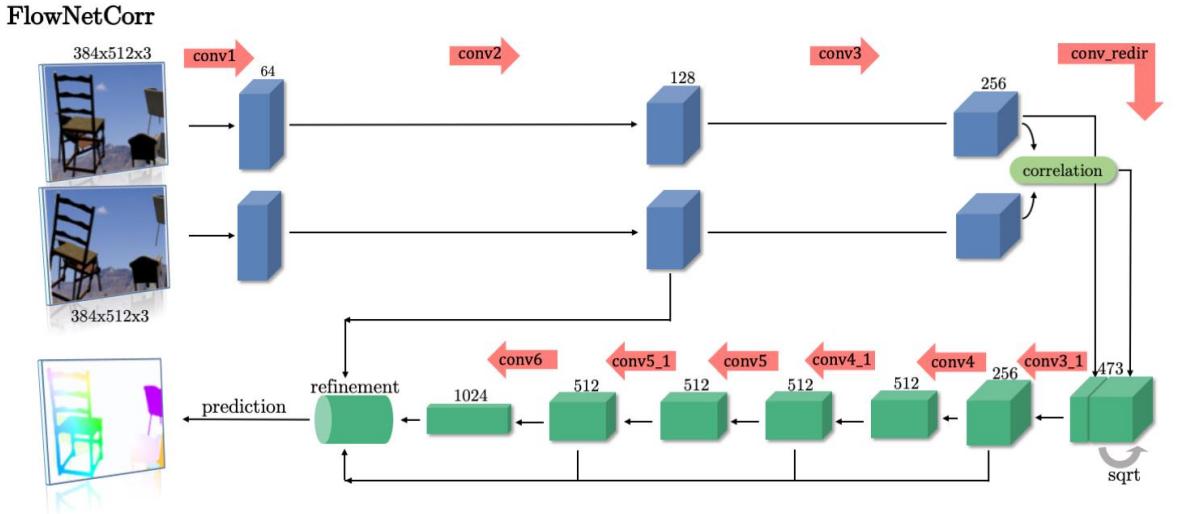
[FlowNet](#): Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning optical flow with convolutional networks." *IEEE ICCV 2015* --- Citations: 3465 + 632



- The first deep learning optical flow architecture has been introduced by FlowNet
- FlowNet directly estimates the optical flow using a generic CNN U-Net architecture.
- Due to the lack of data with optical flow groundtruth the authors generated a new dataset: “Flying Chairs”, Meyer *et. al.* [57], which is a synthetic dataset with optical flow ground truth.
- It consists of more than 20K image pairs and corresponding flow fields of 3D chair models moving with just affine motion in front of random backgrounds.
- This dataset is necessary for network convergence, since CNN typically has a very large number of trainable weights (tens of Millions) requiring a considerable number of input data to avoid overfitting.

FlowNet: Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning optical flow with convolutional networks." *IEEE ICCV 2015* --- Citations: 3465 + 632

- **FlownetS (Flownet Simple)**
 - consists of a CNN receiving two stacked input RGB frames
 - supervisely trained on the optical flow groundtruth.
- **FlowNetC**
 - supervisely trained on the groundtruth,
 - instead of working with a stacked input the two frames are fed to two identical branches which are merged on a later stage by a correlation layer,
- **FlowNetC correlation layer:**
 - perform cross-correlation (cost volume) between the feature maps of the two input,
 - enabling the network to compare each patch with no additional learnt parameters (at the correlation layer).
- Both networks: upsample feature maps by upconvolutions at the output side of the network to increase the resolution
- Skip connections are used to connect layers on the contractive part to the expanding (refinement) part providing additional information of flow level features at the upsampling stage.



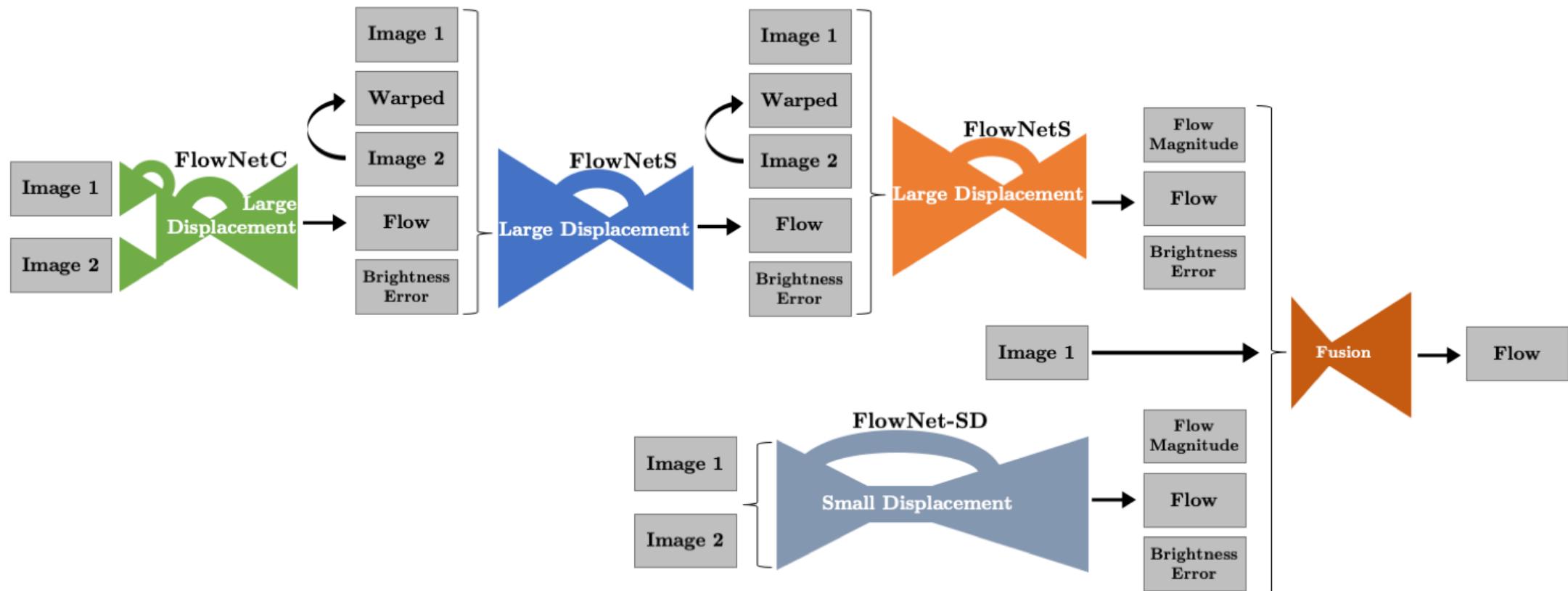
FlowNetC Details

- Correlation layer is used to perform multiplicative patch comparisons between two feature maps.
- More specifically, given two multi-channel feature maps f_1, f_2 , with w, h , and c being their width, height and number of channels. The ‘correlation’ of two patches centered at x_1 in the first map and x_2 in the second map is then defined as:

$$c(x_1, x_2) = \sum_{o \in [-k, k] \times [-k, k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle$$

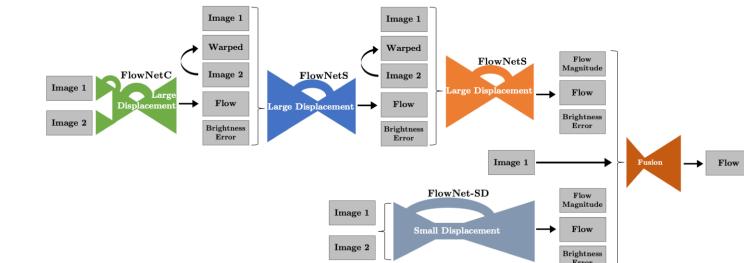
- where x_1 and x_2 : the center of the first map and the second map respectively, and the square space patch of size $K = 2k+1$.
- For computation reasons, the maximum displacement is limited.
- To be specific, for each location x_1 ,
- the range of x_2 by computing correlations in a neighborhood of size $D = 2d+1$,
- d is a given maximum displacement.
- The size of an output is $(w \cdot h \cdot D^2)$. Afterwards, the feature map is concatenated, which is extracted from f_1 using convolution layer, with the output.

FlowNet2.0: Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.



FlowNet2.0: Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

- Stacks FlownetC and FlownetS and newdesigned FlowNet-SD (Small Displacement)
- Flownet-SD: has larger input feature maps and is trained on a dataset with small displacement, ChairsSDHom.
- After each subnetwork the flow is warped and compared with the second image and the error is fed to a fusion network that takes as inputs
 - the estimated flows,
 - the flows magnitudes and
 - the brightness error after warping.
- The fusion network contracts and expands to full resolution producing the final flow fields.
- Due to the large size of FlowNet 2.0, i.e., around 38 Millions parameters, its subnetworks have been trained sequentially by training one subnetwork while freezing the weights of the others.
- Moreover, the authors have generated a new more realistic dataset, Things3D [37] to be robust to untextured regions and to produce flow magnitude histograms close to those of the UCF101 [83] dataset, which is composed of real sequences.



PWC-Net : Inspired by Pyramid Processing for Flow Estimation

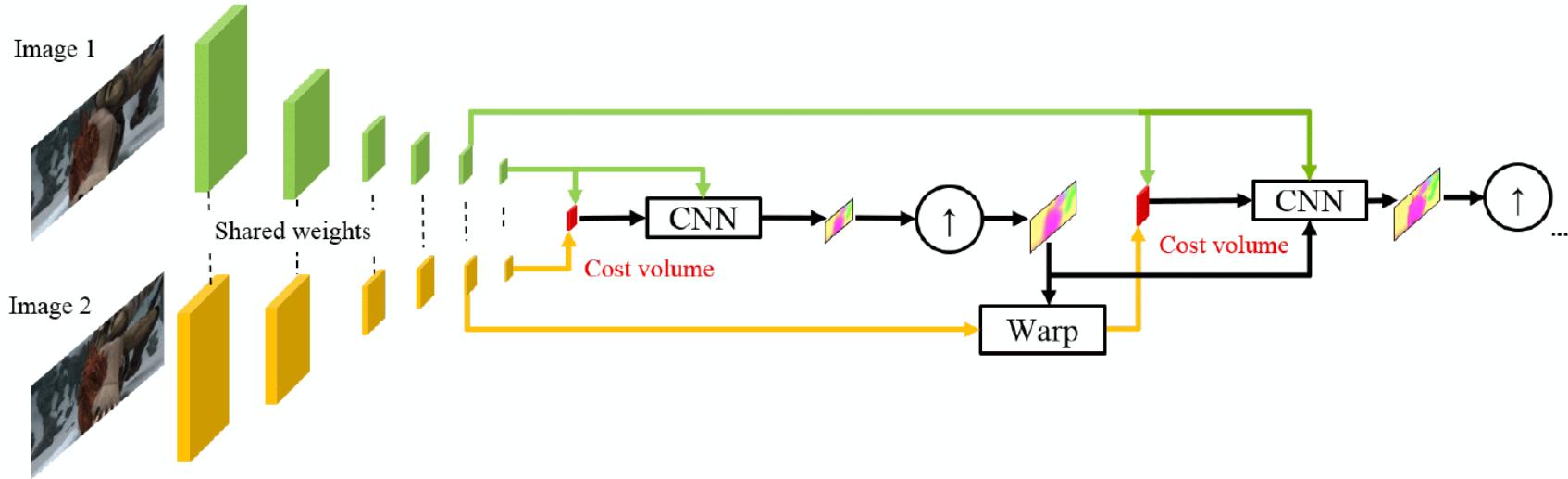
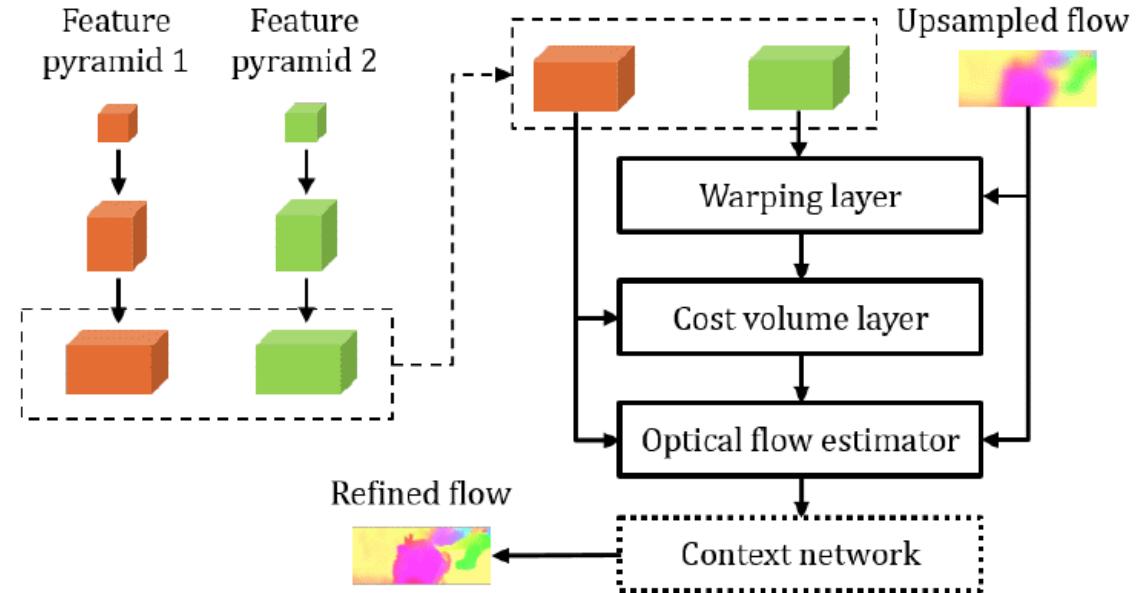
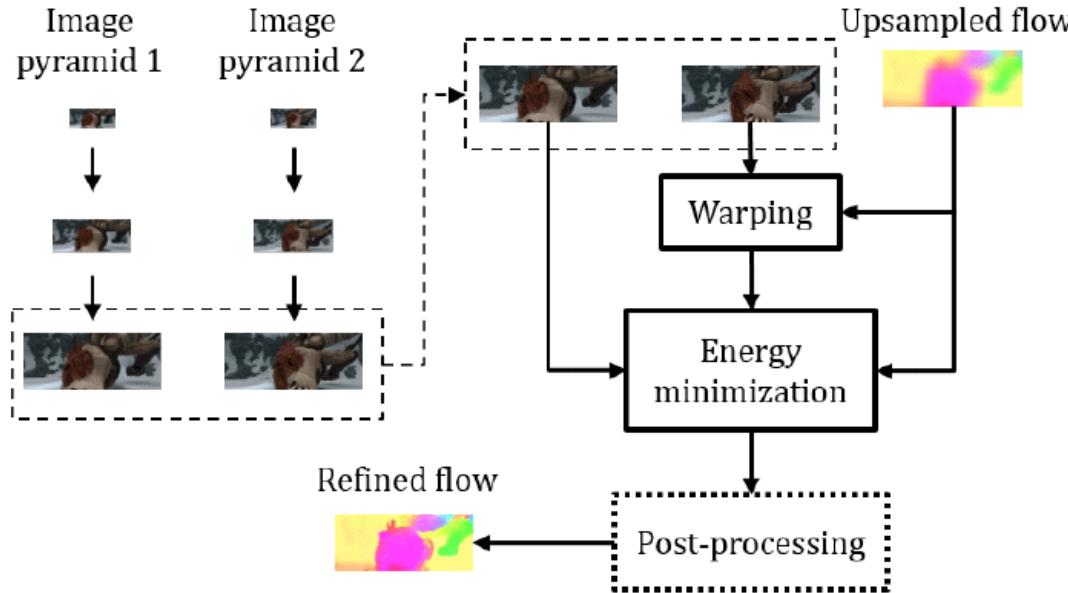


Fig. 2. Network architecture of PWC-Net. We only show the flow estimation modules at the top two levels. For the rest of the pyramidal levels, the flow estimation modules have the same structure as the second to top level.

- Replace the fixed image pyramid with learnable feature pyramids
- Warping, as in traditional estimation, is a layer to estimate large motion
- Cost volume is computed using features of the first image and the warped features of the second image
- The cost volume, features of the first image, and the upsampled flow are fed to a CNN to estimate flow at the current level, which is then upsampled to the next (third) level.
- The process repeats until the desired level

Traditional Coarse-to-Fine vs. PWC-Net

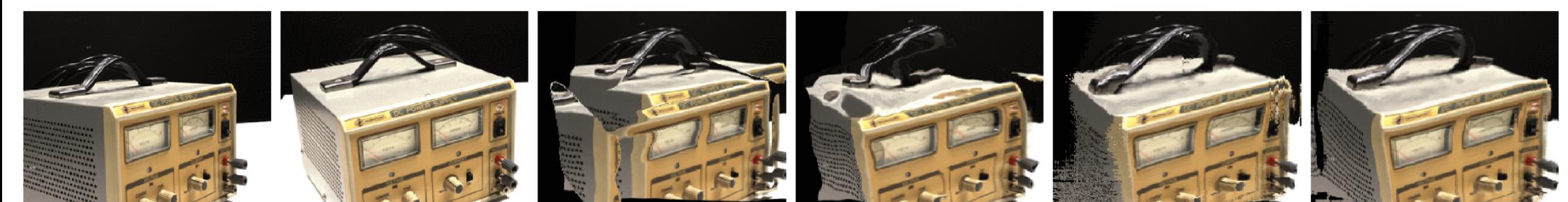


- Feature Pyramid Extractor: L layers of Conv filters with 16, 32, 64, 96, 128 and 192 feature channels
- Warping Layer: Upsample to the next finest level and warp with rescaled flow:
- Cost Volume Layer: Correlation with motion range of d pixels → $c_w^l(x) = c_2^l(x + 2 \times \text{up}_2(w^{l+1})(x))$
- Optical Flow Estimator: Multi-layer CNN with Cost Volume, Image 1 Features and Upsampled flow as inputs.

$$cv^l(x_1, x_2) = \frac{1}{N} \left(c_1^l(x_1) \right)^T c_w^l(x_2)$$



DGC-Net: Dense Geometric Correspondence Network



Reference

Target

Flow based warping

DGC-Net based warping

- Closely related to optical flow estimation where ConvNets (CNNs)
- Optical flow methods do not deal well with the strong geometric transformations
- Coarse-to-fine CNN-based framework leverages the advantages of optical flow approaches and extends them to the case of large transformations providing dense and subpixel accurate estimates.
- Trained on synthetic transformations and demonstrates very good performance to unseen, realistic, data.
- Apply to the problem of relative camera pose estimation: Outperforms existing dense approaches.

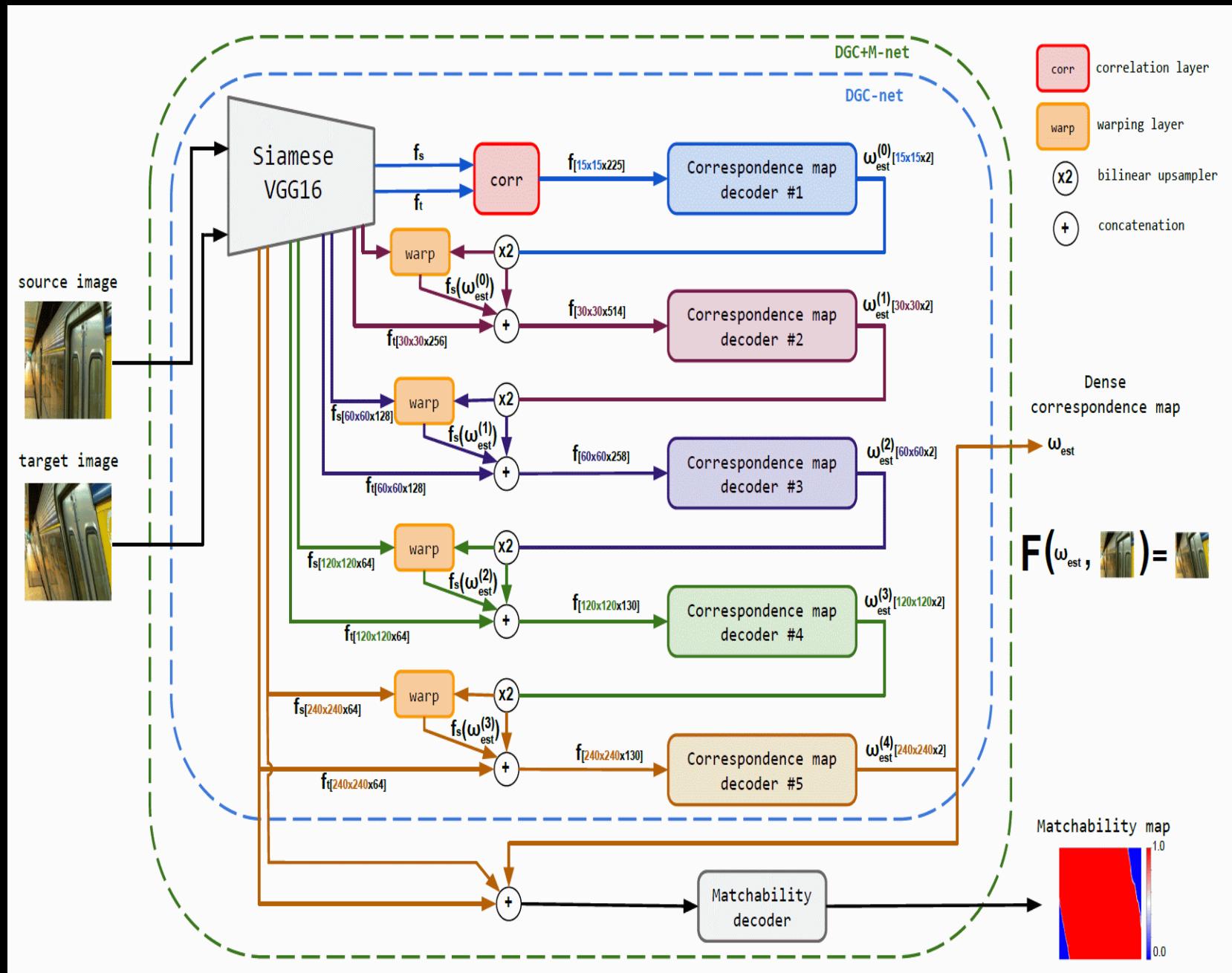
1. Feature pyramid creator.

2. Correlation layer estimates the pairwise similarity score of the source and target feature descriptors.

3. Fully convolutional correspondence map decoders predict the dense correspondence map between input image pair at each level of the feature pyramid.

4. Warping layer warps features of the source image using the upsampled transforming grid from a correspondence map decoder.

5. The matchability decoder is a tiny CNN that predicts a confidence map with higher scores for those pixels in the source image that have correspondences in the target.



References for Optical Flow

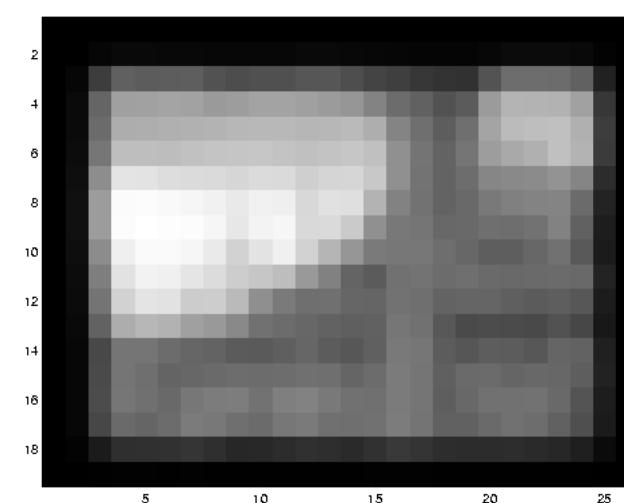
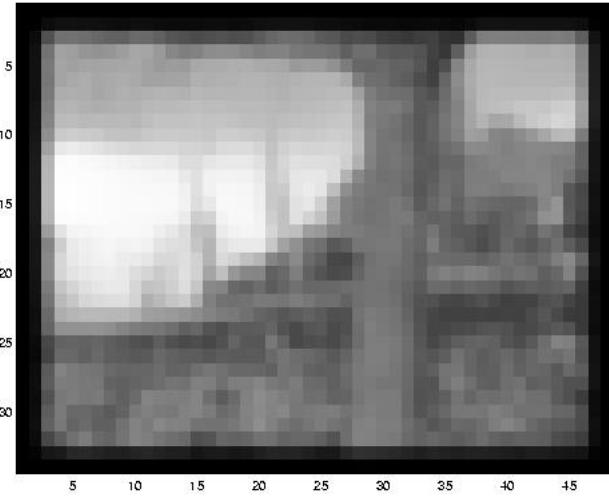
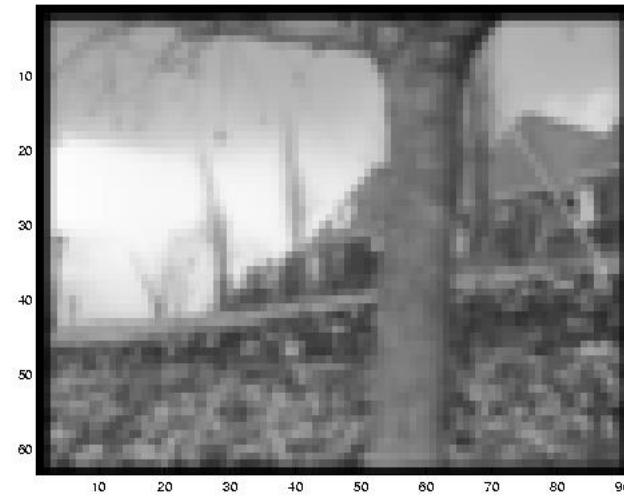
- Baker S, Scharstein D, Lewis JP, Roth S, Black MJ, Szeliski R. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*. 2011 Mar 1;92(1):1-31. – 1285 citations
- Sun, Deqing, Stefan Roth, and Michael J. Black. "Secrets of optical flow estimation and their principles." *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010.—685 citations
- Fleet, David, and Yair Weiss. "Optical flow estimation." *Handbook of mathematical models in computer vision*. Springer US, 2006. 237-257.-257 citations
- Savian, Stefano, Mehdi Elahi, and Tammam Tillo. "Optical flow estimation with deep learning, a survey on recent advances." *Deep biometrics* (2020): 257-287.
- <https://paperswithcode.com/task/optical-flow-estimation>

Dense Motion Estimation

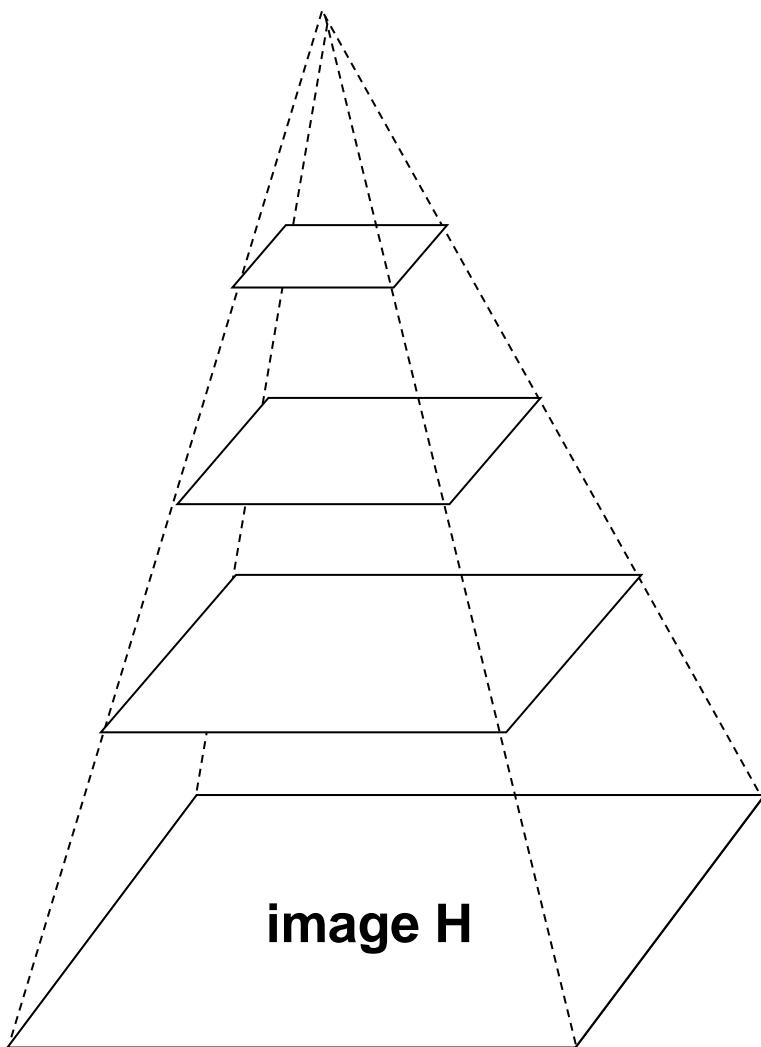
Part II

Optical Flow Estimation and Analysis
Szeliski Book Chapter 9

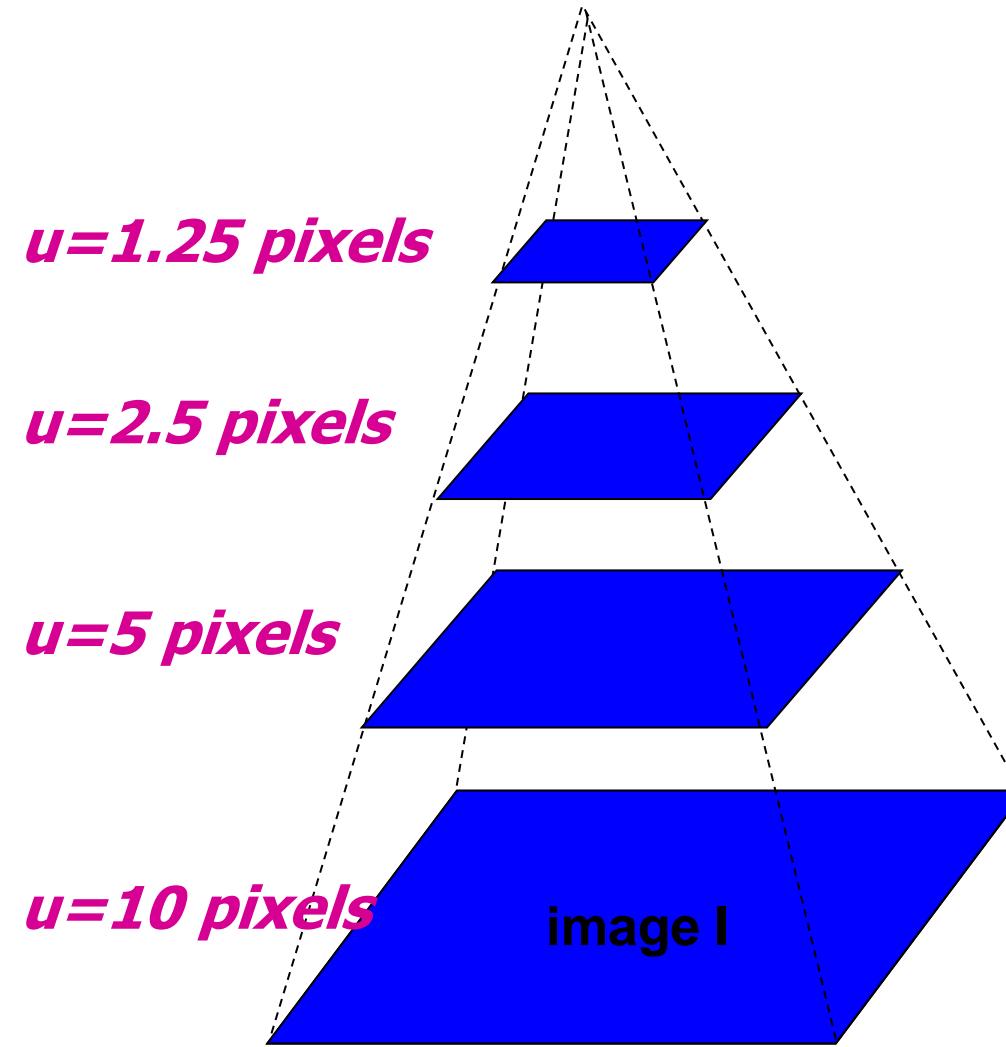
Reduce the resolution!



Coarse-to-fine optical flow estimation

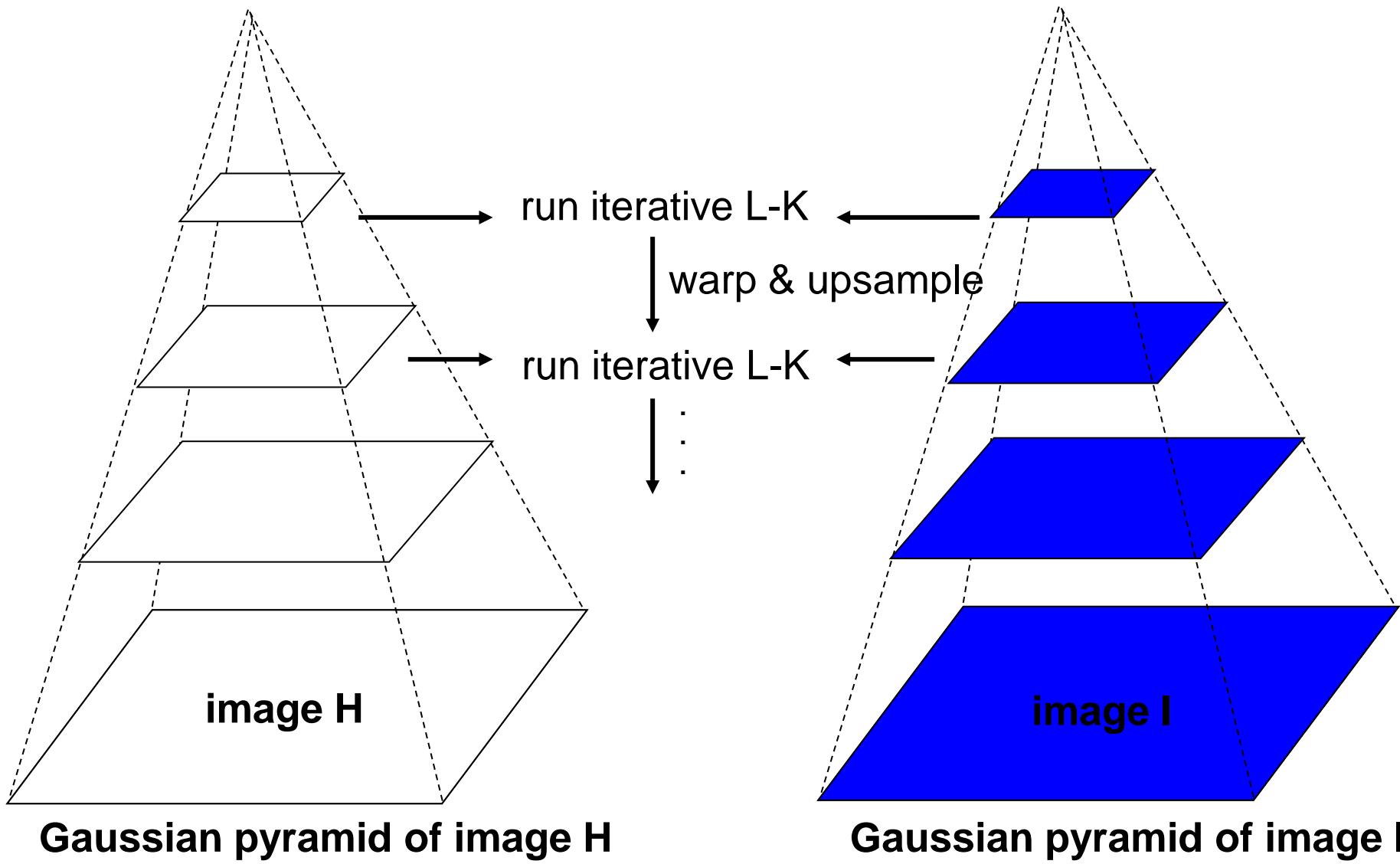


Gaussian pyramid of image H



Gaussian pyramid of image I

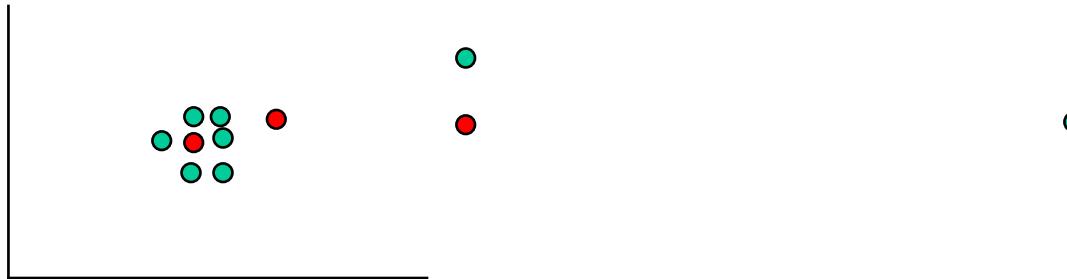
Coarse-to-fine optical flow estimation



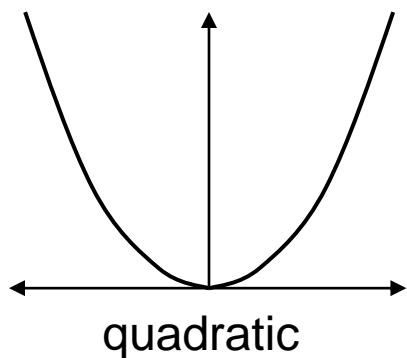
Robust methods

L-K minimizes a sum-of-squares error metric

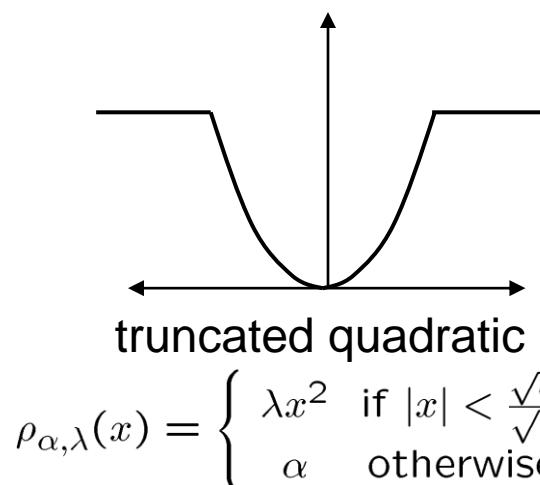
- least squares techniques overly sensitive to outliers



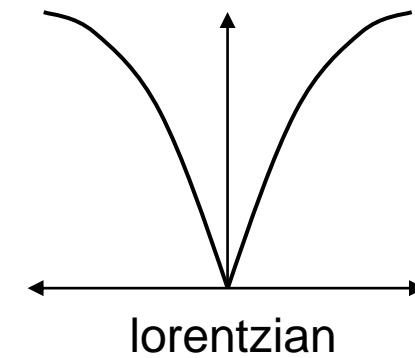
Error metrics



$$\rho(x) = x^2$$



$$\rho_{\alpha,\lambda}(x) = \begin{cases} \lambda x^2 & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}} \\ \alpha & \text{otherwise} \end{cases}$$



$$\rho_\sigma(x) = \log \left(1 + \frac{1}{2} \left(\frac{x}{\sigma} \right)^2 \right)$$

Robust optical flow

Robust Horn & Schunk

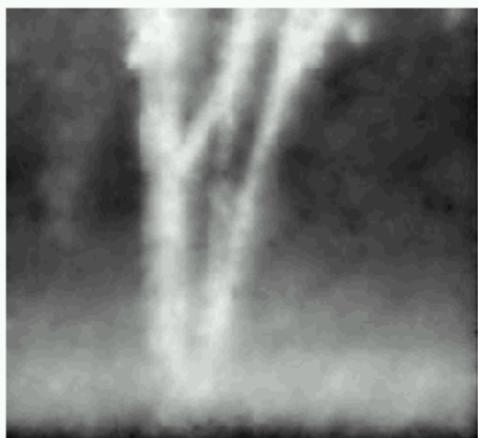
$$\int \int \rho(I_t + \nabla I \cdot [u \ v]) + \lambda^2 \rho(\|\nabla u\|^2 + \|\nabla v\|^2) \ dx \ dy$$

Robust Lucas-Kanade

$$\sum_{(x,y) \in W} \rho(I_t + \nabla I \cdot [u \ v])$$



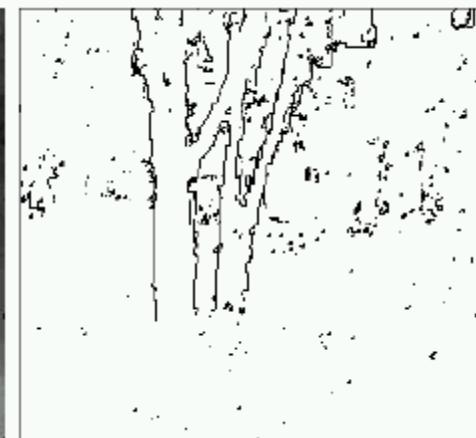
first image



quadratic flow



lorentzian flow



detected outliers

Reference

- Black, M. J. and Anandan, P., A framework for the robust estimation of optical flow, *Fourth International Conf. on Computer Vision (ICCV)*, 1993, pp. 231-236
<http://www.cs.washington.edu/education/courses/576/03sp/readings/black93.pdf>

Dense Flow Visualization



Flow vectors are color coded based on their magnitudes

Dense optical flow algorithm output can be encoded as the HSV color scheme. Using the `cv2.cartToPolar` function, we can convert the displacement coordinates (dx , dy) into polar coordinates as magnitude and angle for every pixel. Here we can encode angle and magnitude as Hue and Value respectively, while Saturation remains constant.
<https://learnopencv.com/optical-flow-in-opencv/>

SAMPLE APPLICATIONS OF OPTICAL FLOW

Automatic Extraction of 2D Layers



Input Sequence



Layers

Sun, Deqing, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "Models matter, so does training: An empirical study of cnns for optical flow estimation." *IEEE transactions on pattern analysis and machine intelligence* 42, no. 6 (2019)

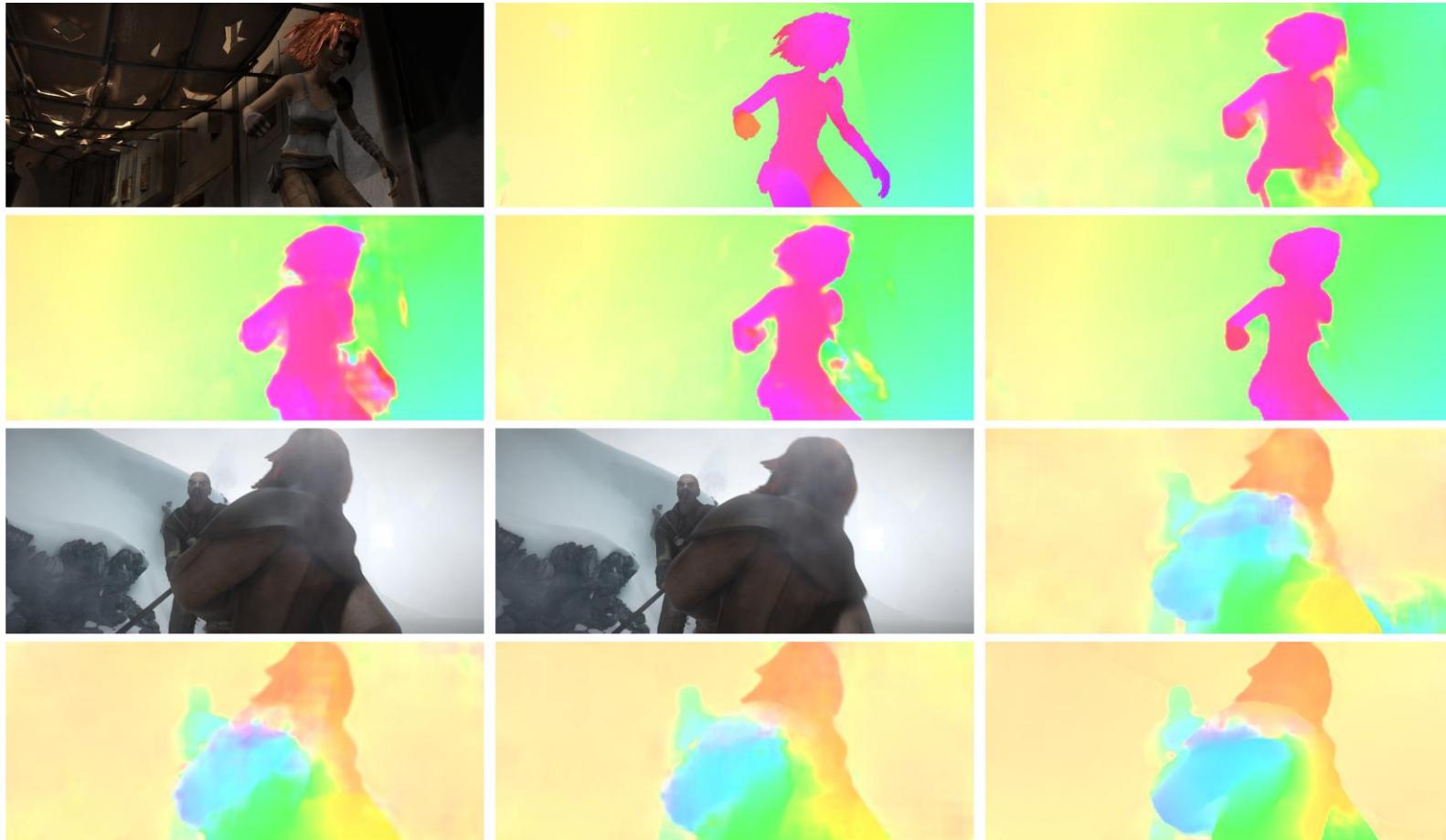


Fig. 5. Results on Sintel training and test sets. Context network, DenseNet connections, and fine-tuning all improve the results. Small and rapidly moving objects, e.g., the left arm in “Market_5”, are still challenging to the pyramid-based PWC-Net.

Sun, Deqing, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "Models matter, so does training: An empirical study of cnns for optical flow estimation." *IEEE transactions on pattern analysis and machine intelligence* 42, no. 6 (2019)

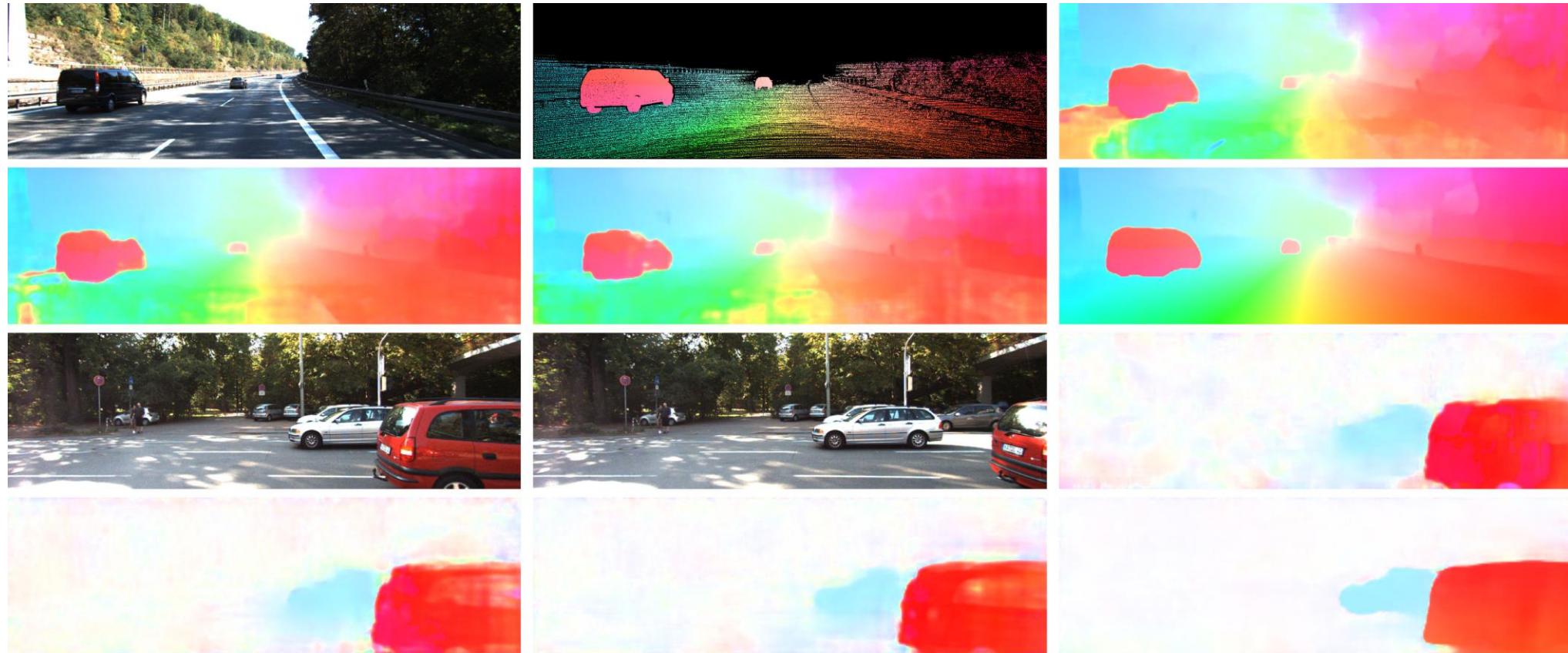


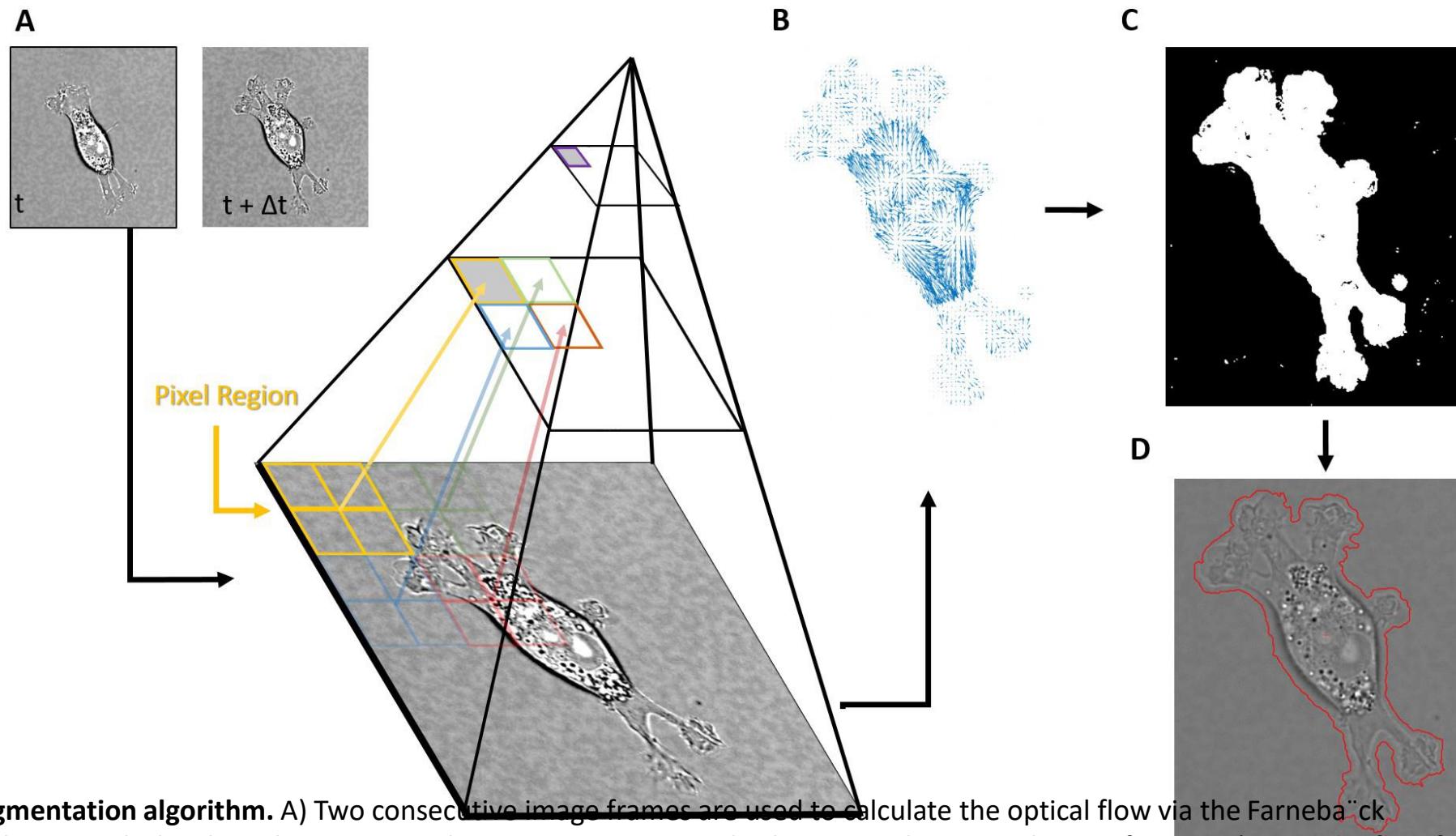
Fig. 7. Results on KITTI 2015 training and test sets. Fine-tuning fixes large regions of errors and recovers sharp motion boundaries.

Sun, Deqing, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "Models matter, so does training: An empirical study of cnns for optical flow estimation." *IEEE transactions on pattern analysis and machine intelligence* 42, no. 6 (2019)



Fig. 9. Results on Middlebury and HD1K test sets. PWC-Net_ROB has not been trained using the training data of Middlebury but performs reasonably well on the test set. It cannot recover the fine motion details of the twigs in Grove though. PWC-Net_ROB has reasonable results in the regions occluded by the windshield wipers in sequence 02 of the HD1K test set.

Robitaille, Michael C., Jeff M. Byers, Joseph A. Christodoulides, and Marc P. Raphael. "Robust optical flow algorithm for general single cell segmentation." *Plos one* 17, no. 1 (2022): e0261763



Overview of the segmentation algorithm. A) Two consecutive image frames are used to calculate the optical flow via the Farneback algorithm, which utilizes a multi-level resolution pyramid to estimate intensity displacements between the two frames. B) Once the flow field is calculated, a threshold Th is applied to the magnitude of the individual flow vectors, leaving only flow vectors with a magnitude above Th . C) The pixels associated with the remaining flow field form a binary mask to estimate where the cell is in the image. D) The mask is closed and filled to create the segmented perimeter of the cell.

Optical Flow VFX: PAINTING THE AFTERLIFE IN **WHAT DREAMS MAY COME**

https://www.fxguide.com/fxfeatured/art_of_optical_flow/



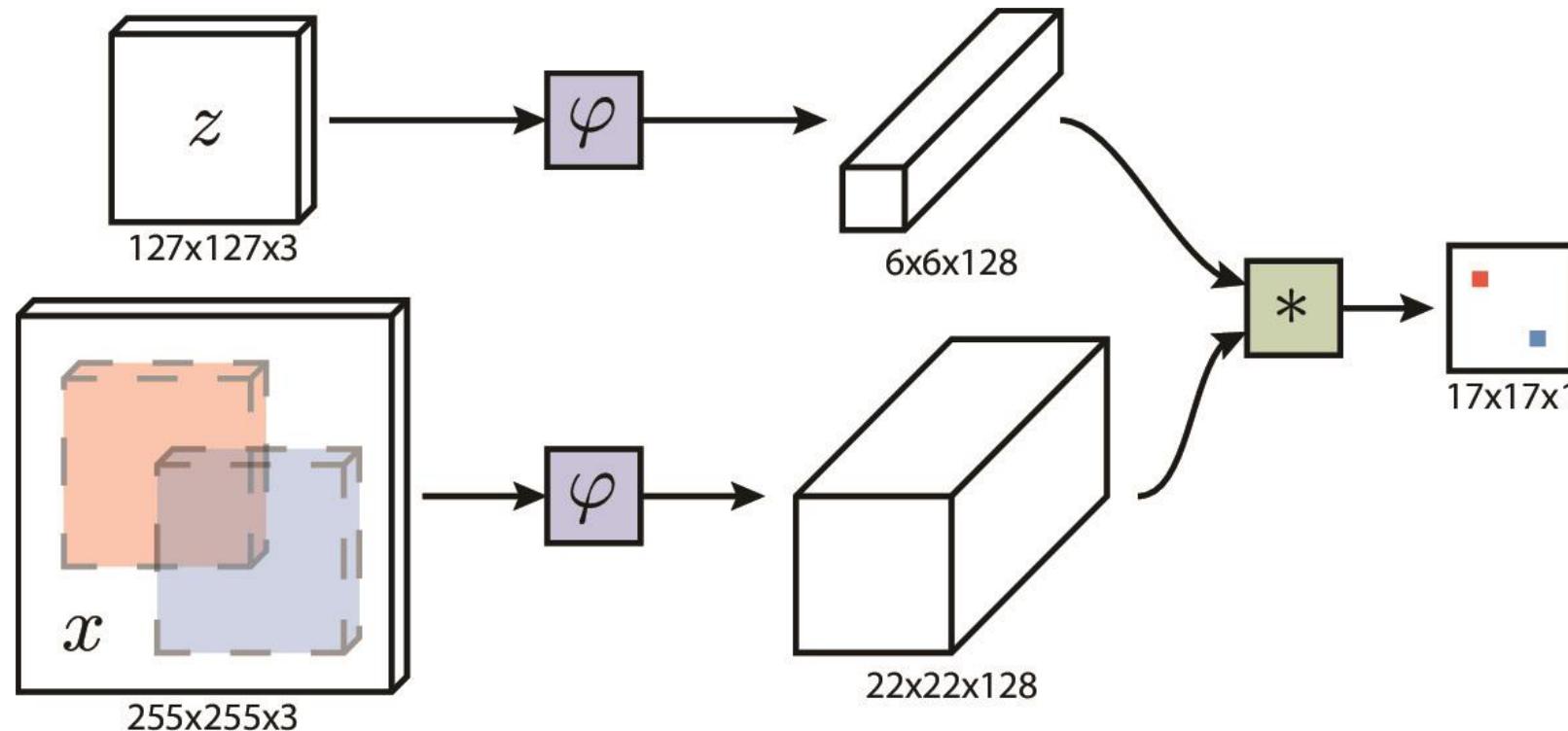
The final shot was enabled with extensive development of tracking techniques, optical flow and a specialized particles tool to produce the painterly effects.

Slide Credit: Szeliski

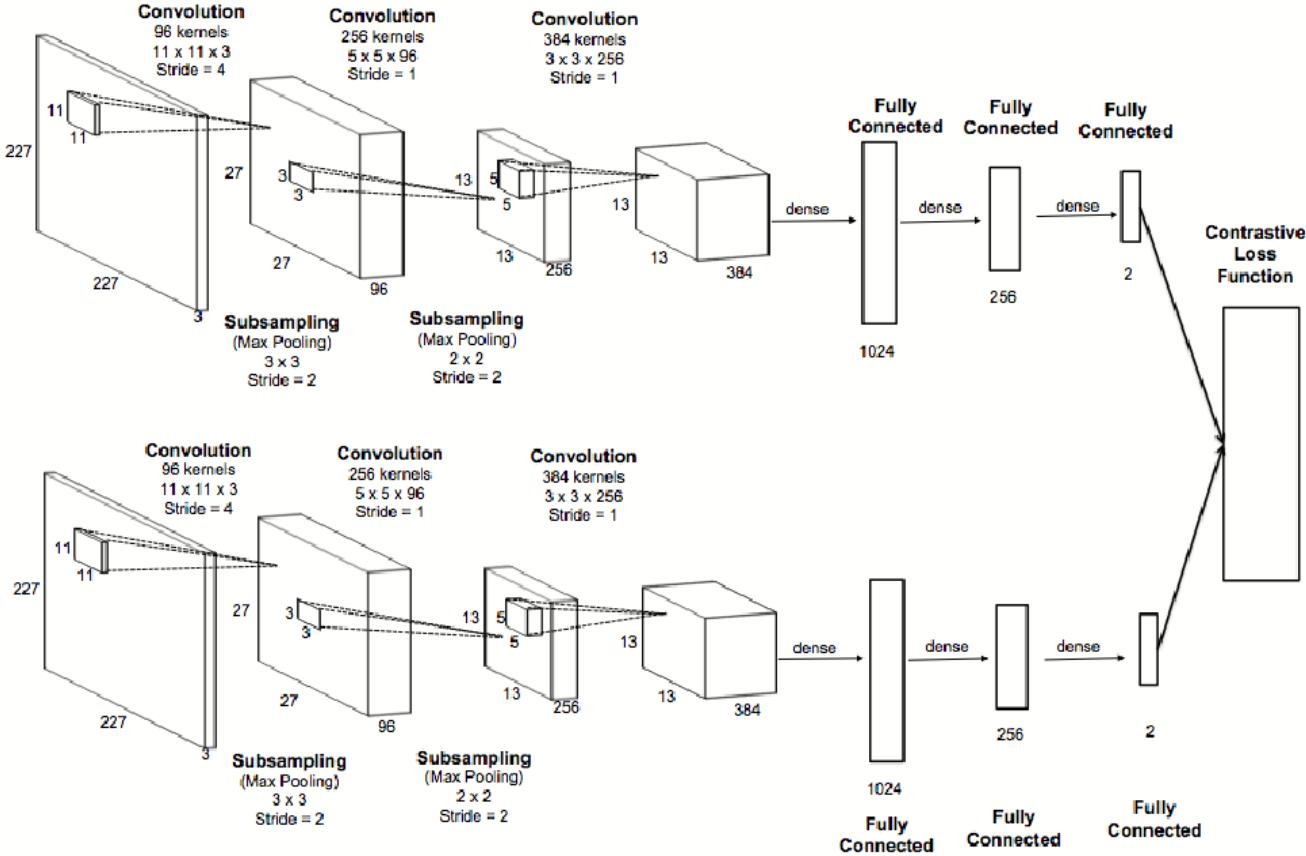
DEEP LEARNING APPROACHES

Deep Learning-based Optical Flow Estimation

- While optical flow estimation needs **precise per-pixel localization**, it also requires finding correspondences **between two input images**.
- This involves not only learning image feature representations, but also learning to match them at different locations in the two images.
- In this respect, optical flow estimation fundamentally differs from previous applications of CNNs.



Siamese Network

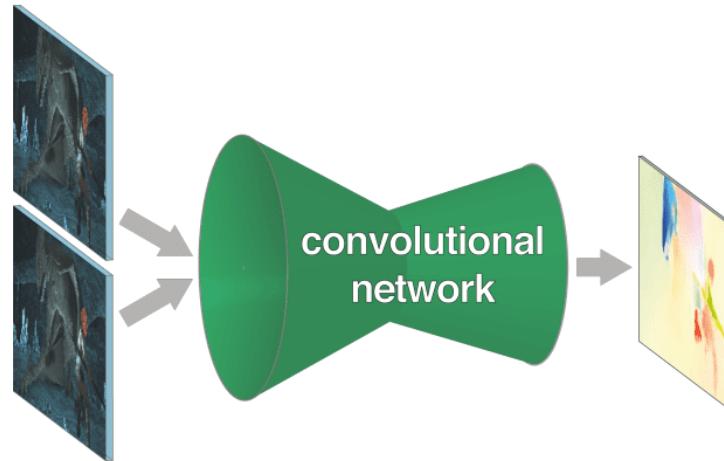


$$Y * D^2 + (1 - Y) * \max(\text{margin} - D, 0)^2$$

Contrastive loss: how good a job the siamese network is distinguishing between the image pairs.
 Y: label.

- 1: image pairs are same class,
 - 0: image pairs are of a different class.
- The $D_{\{w\}}$: Euclidean distance between the outputs of the sister network embeddings.

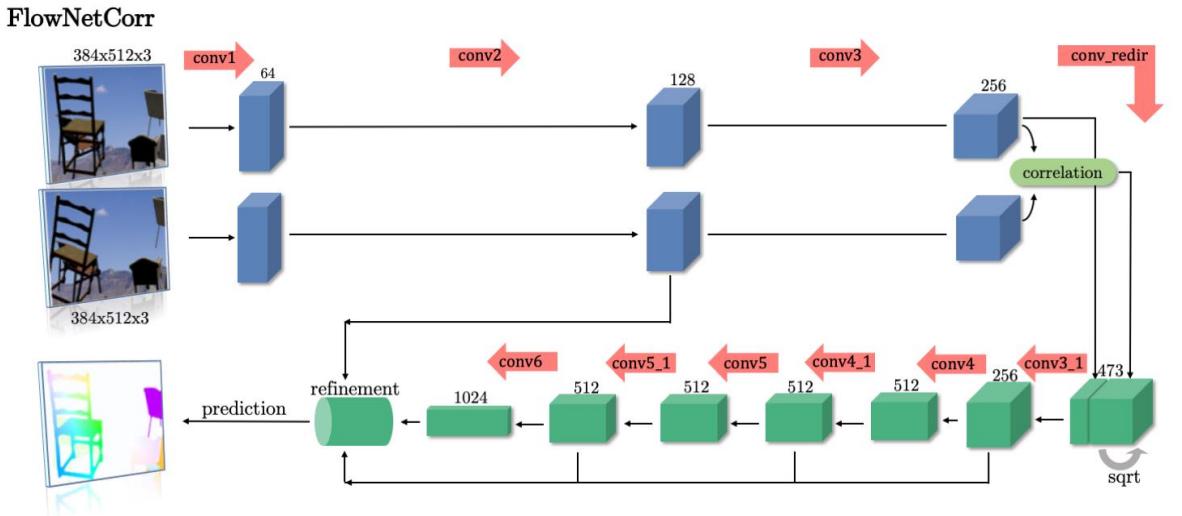
[FlowNet](#): Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning optical flow with convolutional networks." *IEEE ICCV 2015* --- Citations: 3465 + 632



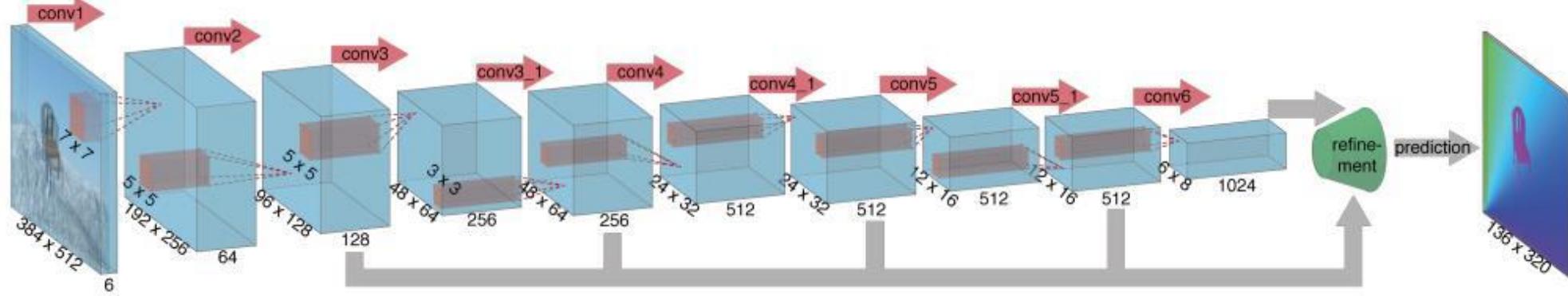
- The first deep learning optical flow architecture has been introduced by FlowNet
- FlowNet directly estimates the optical flow using a generic CNN U-Net architecture.
- Due to the lack of data with optical flow groundtruth the authors generated a new dataset: “Flying Chairs”, Meyer *et. al.* [57], which is a synthetic dataset with optical flow ground truth.
- It consists of more than 20K image pairs and corresponding flow fields of 3D chair models moving with just affine motion in front of random backgrounds.
- This dataset is necessary for network convergence, since CNN typically has a very large number of trainable weights (tens of Millions) requiring a considerable number of input data to avoid overfitting.

FlowNet: Dosovitskiy, Alexey, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning optical flow with convolutional networks." *IEEE ICCV 2015* --- Citations: 3465 + 632

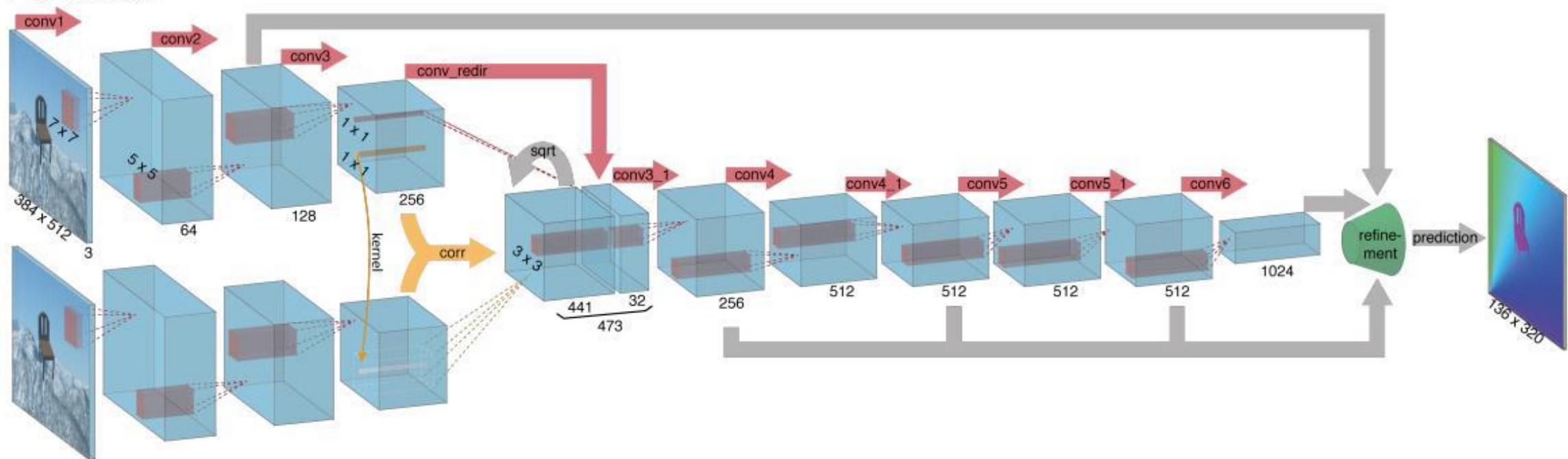
- **FlownetS (Flownet Simple)**
 - consists of a CNN receiving two stacked input RGB frames
 - supervisely trained on the optical flow groundtruth.
- **FlowNetC**
 - supervisely trained on the groundtruth,
 - instead of working with a stacked input the two frames are fed to two identical branches which are merged on a later stage by a correlation layer,
- **FlowNetC correlation layer:**
 - perform cross-correlation (cost volume) between the feature maps of the two input,
 - enabling the network to compare each patch with no additional learnt parameters (at the correlation layer).
- Both networks: upsample feature maps by upconvolutions at the output side of the network to increase the resolution
- Skip connections are used to connect layers on the contractive part to the expanding (refinement) part providing additional information of flow level features at the upsampling stage.



FlowNetSimple



FlowNetCorr



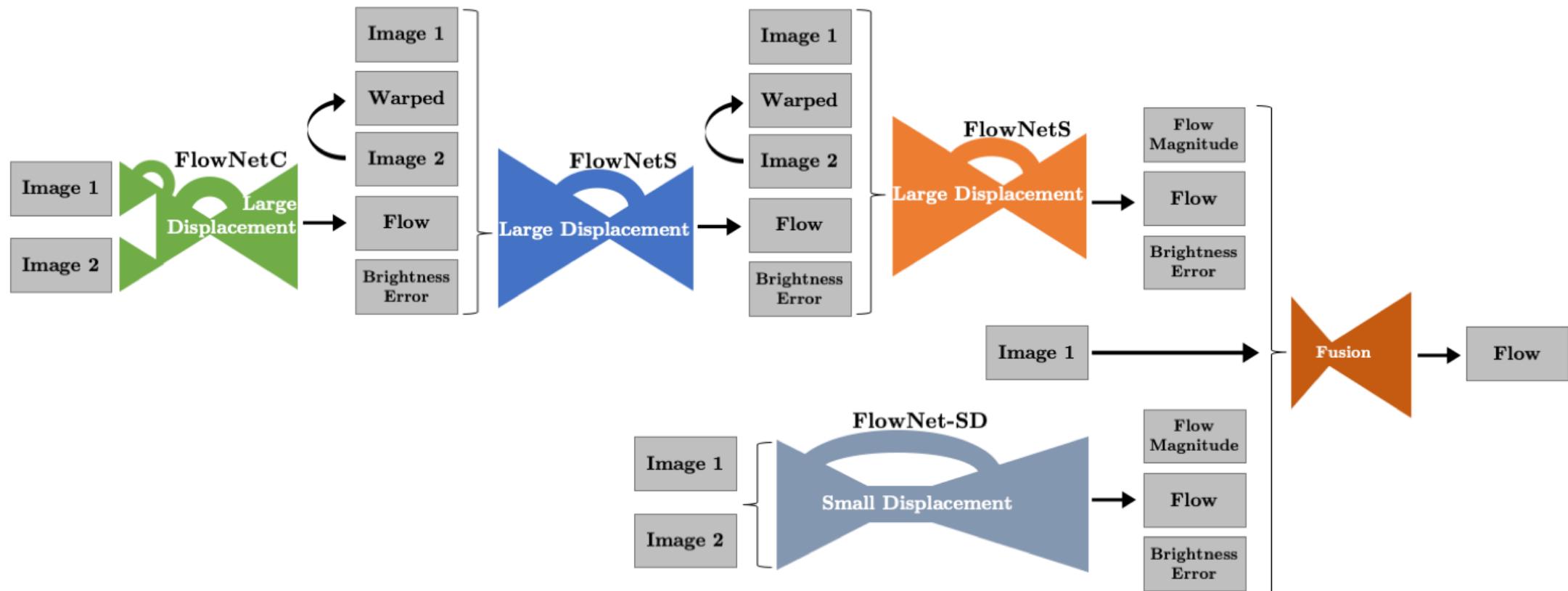
FlowNetC Details

- Correlation layer is used to perform multiplicative patch comparisons between two feature maps.
- More specifically, given two multi-channel feature maps f_1, f_2 , with w, h , and c being their width, height and number of channels. The ‘correlation’ of two patches centered at x_1 in the first map and x_2 in the second map is then defined as:

$$c(x_1, x_2) = \sum_{o \in [-k, k] \times [-k, k]} \langle f_1(x_1 + o), f_2(x_2 + o) \rangle$$

- where x_1 and x_2 : the center of the first map and the second map respectively, and the square space patch of size $K = 2k+1$.
- For computation reasons, the maximum displacement is limited.
- To be specific, for each location x_1 ,
- the range of x_2 by computing correlations in a neighborhood of size $D = 2d+1$,
- d is a given maximum displacement.
- The size of an output is $(w \cdot h \cdot D^2)$. Afterwards, the feature map is concatenated, which is extracted from f_1 using convolution layer, with the output.

FlowNet2.0: Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

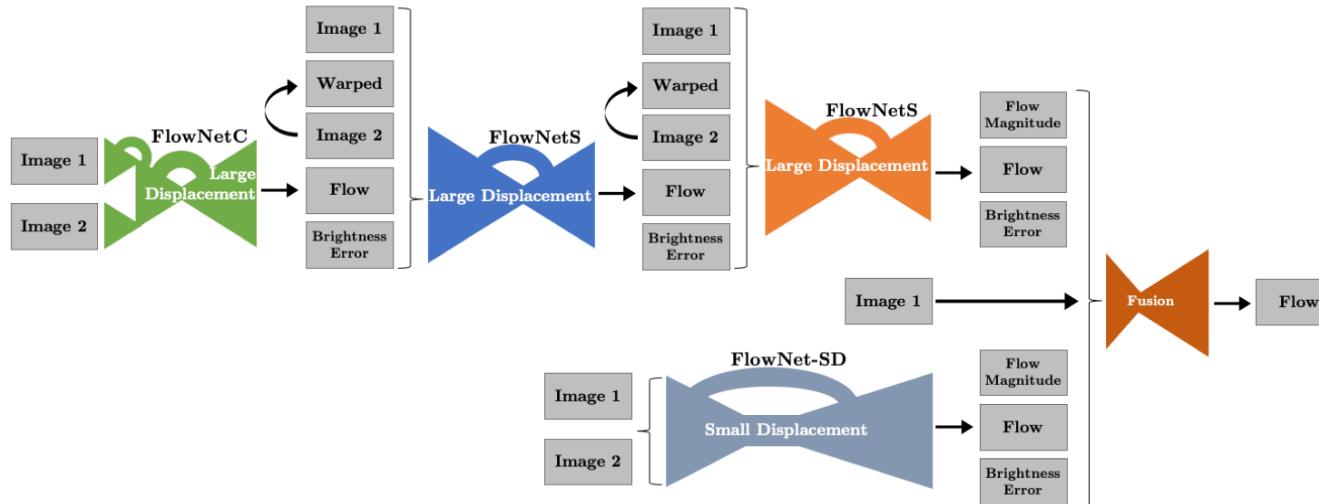


FlowNet2.0: Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2017.

- Stacks FlownetC and FlownetS and newdesigned FlowNet-SD (Small Displacement)

- Flownet-SD: has larger input with small displacement, Ch

- After each subnetwork the first second image and the error inputs
 - the estimated flows,
 - the flows magnitudes and
 - the brightness error after warping



- The fusion network contracts the final flow fields.

- Due to the large size of FlowNet 2.0, i.e., around 38 Millions parameters, its subnetworks have been trained sequentially by training one subnetwork while freezing the weights of the others.

- Moreover, the authors have generated a new more realistic dataset, Things3D [37] to be robust to untextured regions and to produce flow magnitude histograms close to those of the UCF101 [83] dataset, which is composed of real sequences.

PWC-Net : Inspired by Pyramid Processing for Flow Estimation

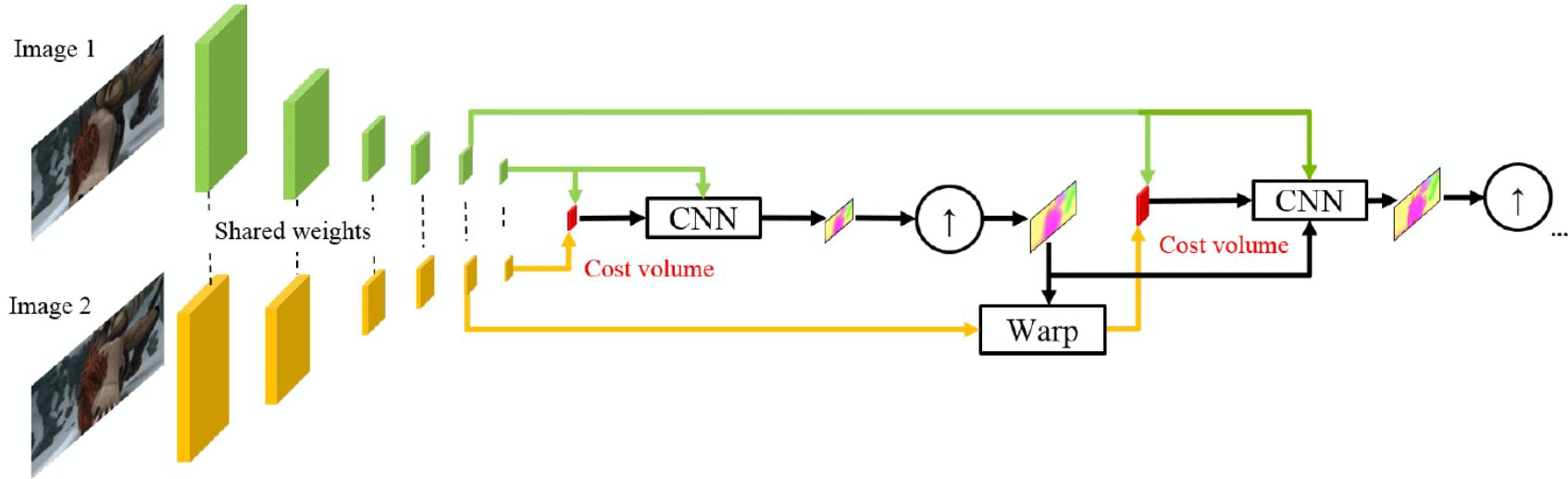
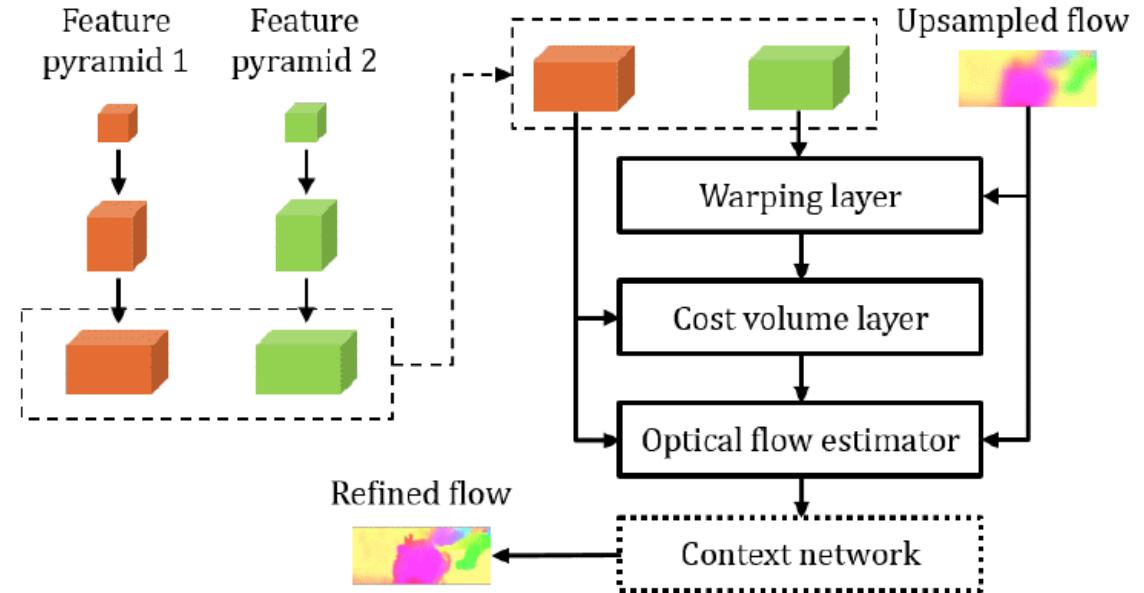
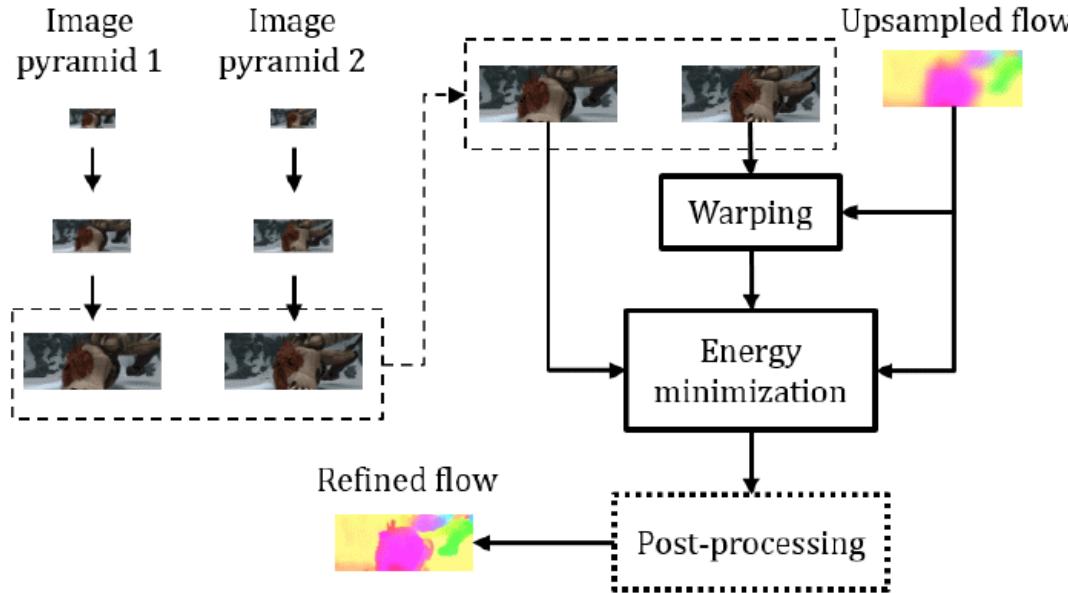


Fig. 2. Network architecture of PWC-Net. We only show the flow estimation modules at the top two levels. For the rest of the pyramidal levels, the flow estimation modules have the same structure as the second to top level.

- Replace the fixed image pyramid with learnable feature pyramids
- Warping, as in traditional estimation, is a layer to estimate large motion
- Cost volume is computed using features of the first image and the warped features of the second image
- The cost volume, features of the first image, and the upsampled flow are fed to a CNN to estimate flow at the current level, which is then upsampled to the next (third) level.
- The process repeats until the desired level

Traditional Coarse-to-Fine vs. PWC-Net



- Feature Pyramid Extractor: L layers of Conv filters with 16, 32, 64, 96, 128 and 192 feature channels
- Warping Layer: Upsample to the next finest level and warp with rescaled flow:
- Cost Volume Layer: Correlation with motion range of d pixels → $c_w^l(x) = c_2^l(x + 2 \times \text{up}_2(w^{l+1})(x))$
- Optical Flow Estimator: Multi-layer CNN with Cost Volume, Image 1 Features and Upsampled flow as inputs.

$$cv^l(x_1, x_2) = \frac{1}{N} \left(c_1^l(x_1) \right)^T c_w^l(x_2),$$

DGC-Net: Dense Geometric Correspondence Network



- Closely related to optical flow estimation where ConvNets (CNNs)
- Optical flow methods do not deal well with the strong geometric transformations
- Coarse-to-fine CNN-based framework leverages the advantages of optical flow approaches and extends them to the case of large transformations providing dense and subpixel accurate estimates.
- Trained on synthetic transformations and demonstrates very good performance to unseen, realistic, data.
- Apply to the problem of relative camera pose estimation: Outperforms existing dense approaches.

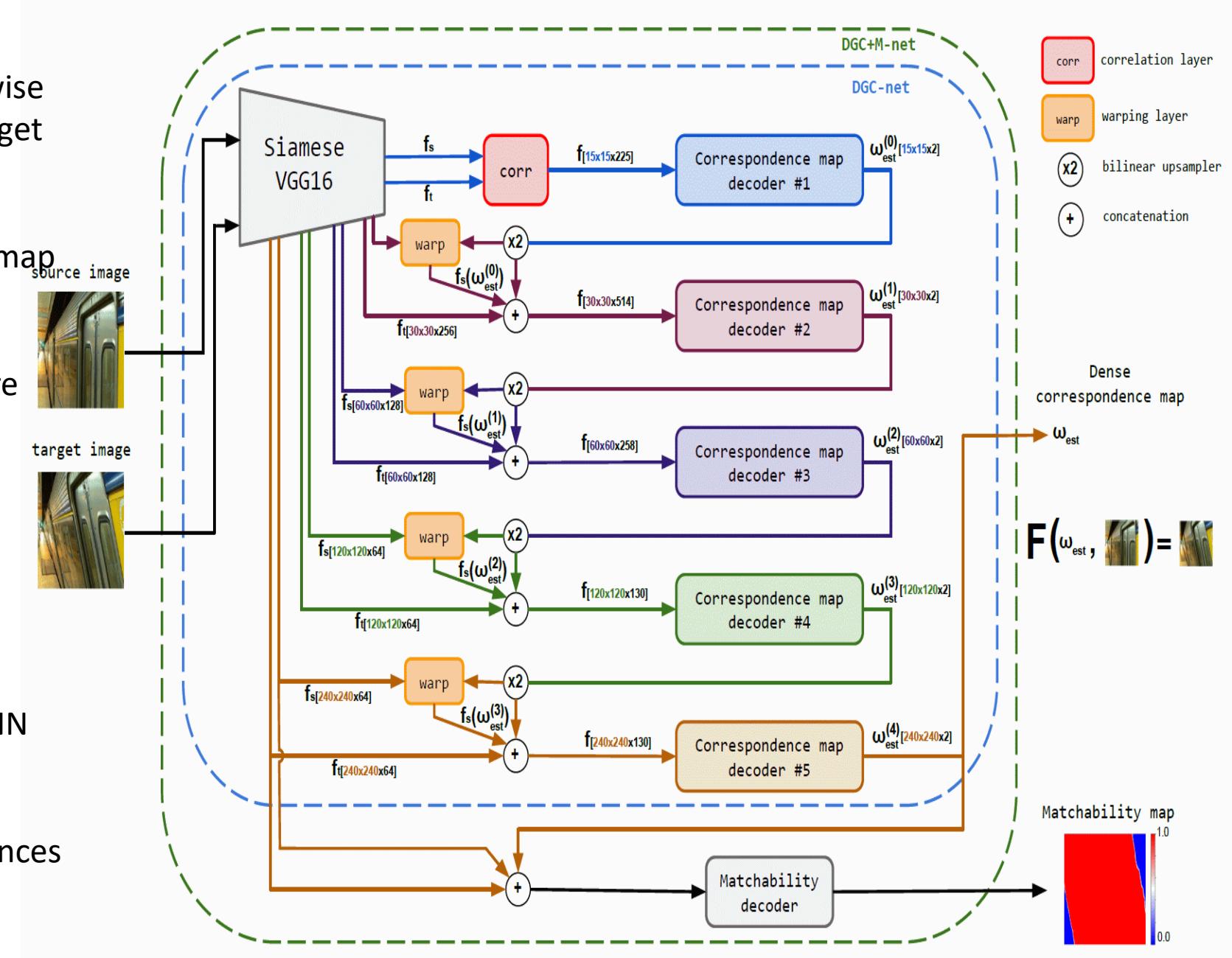
1. Feature pyramid creator.

1. Correlation layer estimates the pairwise similarity score of the source and target feature descriptors.

2. Fully convolutional correspondence map decoders predict the dense correspondence map between input image pair at each level of the feature pyramid.

3. Warping layer warps features of the source image using the upsampled transforming grid from a correspondence map decoder.

4. The matchability decoder is a tiny CNN that predicts a confidence map with higher scores for those pixels in the source image that have correspondences in the target.



References for Optical Flow

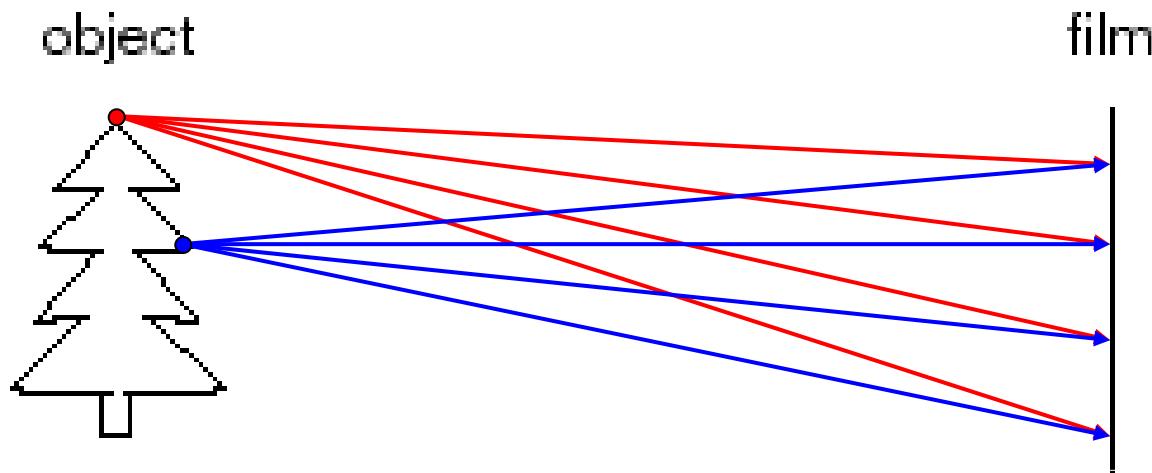
- Baker S, Scharstein D, Lewis JP, Roth S, Black MJ, Szeliski R. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*. 2011 Mar 1;92(1):1-31. – 1285 citations
- Sun, Deqing, Stefan Roth, and Michael J. Black. "Secrets of optical flow estimation and their principles." *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010.—685 citations
- Fleet, David, and Yair Weiss. "Optical flow estimation." *Handbook of mathematical models in computer vision*. Springer US, 2006. 237-257.-257 citations
- Savian, Stefano, Mehdi Elahi, and Tammam Tillo. "Optical flow estimation with deep learning, a survey on recent advances." *Deep biometrics* (2020): 257-287.
- <https://paperswithcode.com/task/optical-flow-estimation>

Image Formation and Projection

Szeliski 2nd Edition

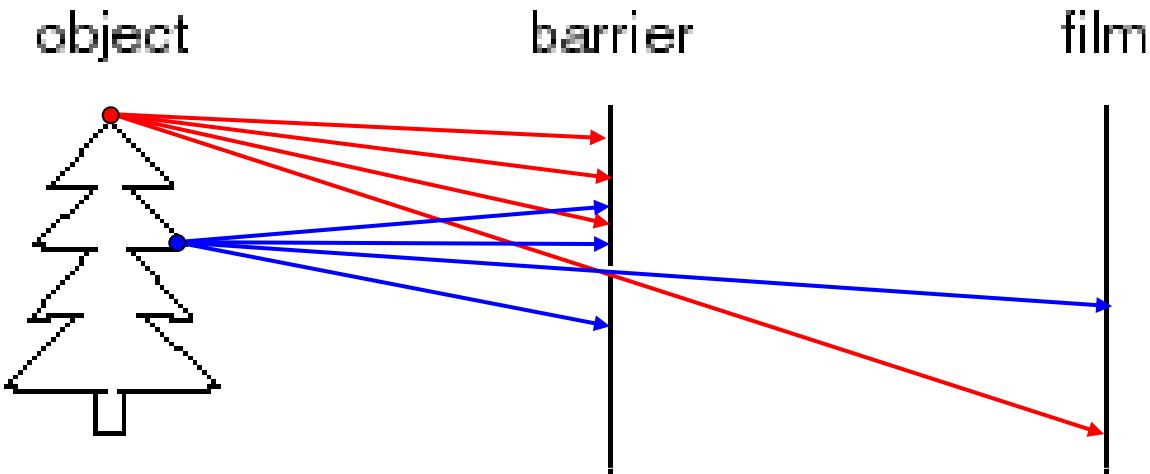
Chapter 2.1

Image formation



- Let's design a camera
 - Idea 1: put a piece of film in front of an object
 - Do we get a reasonable image?

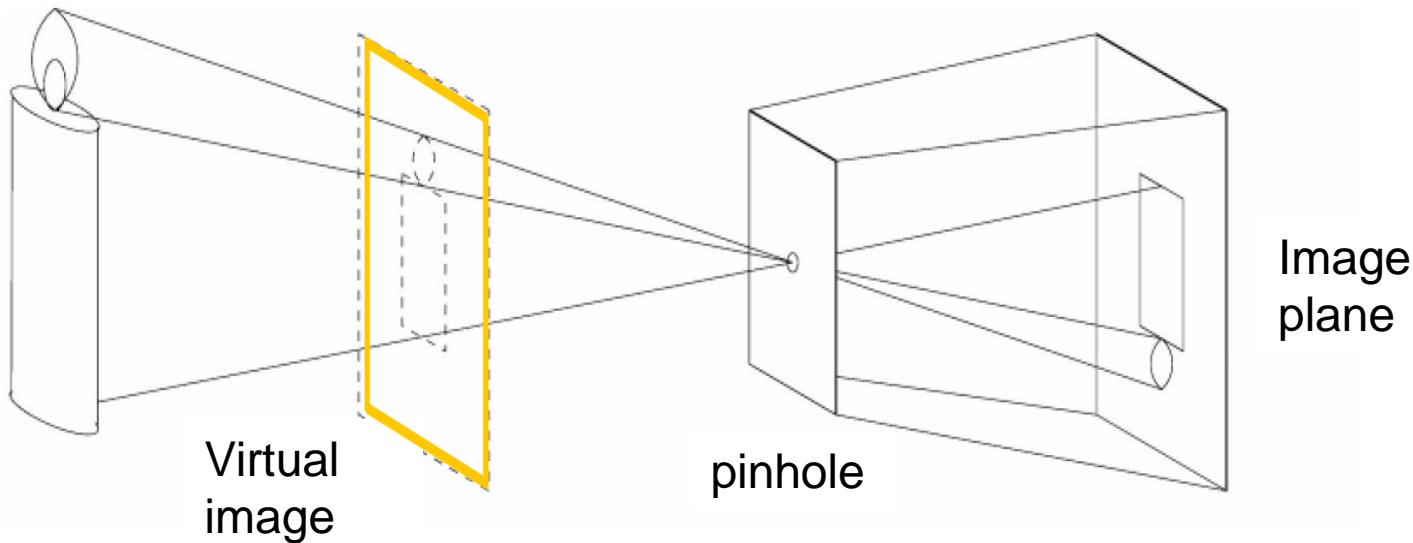
Pinhole camera



- Add a barrier to block off most of the rays
 - This reduces blurring
 - The opening is known as the **aperture**
 - How does this transform the image?

Pinhole camera

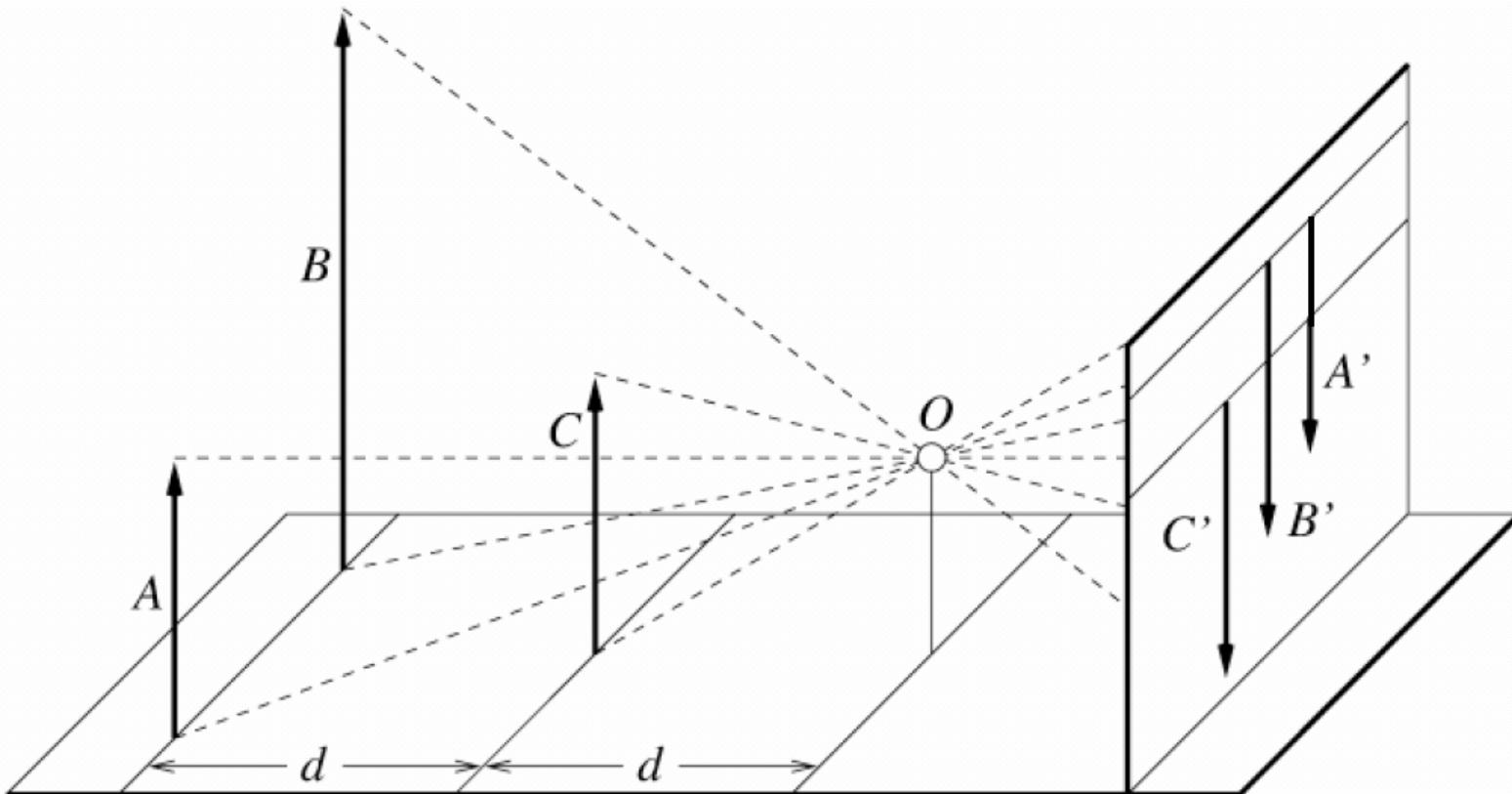
- Pinhole camera is a simple model to approximate imaging process, perspective **projection**.



If we treat pinhole as a point, only one ray from any given point can enter the camera.

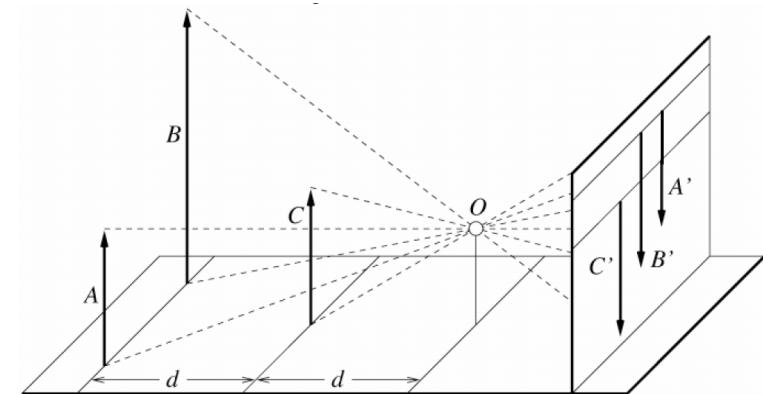
Perspective effects

- Far away objects appear smaller



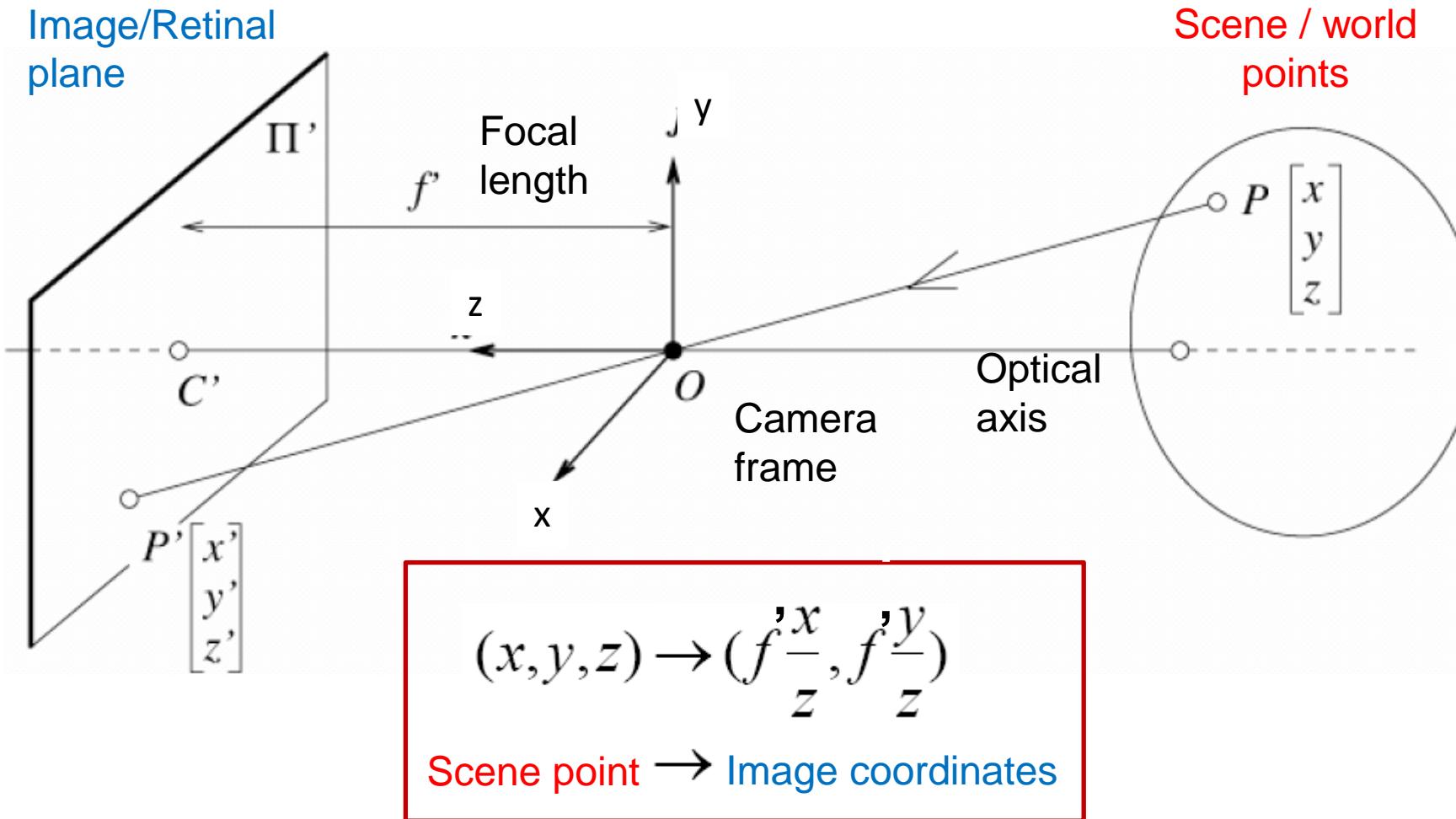
Projection properties

- Many-to-one: any points along same ray map to same point in image
- Points \rightarrow points
- Lines \rightarrow lines (collinearity preserved)
- Distances and angles are **not** preserved
- Degenerate cases:
 - Line through focal point projects to a point.
 - Plane through focal point projects to line
 - Plane perpendicular to image plane projects to part of the image.



Perspective projection equations

- 3d world mapped to 2d projection in image plane



(x', y') : image coordinates

(x, y, z) : world coordinates

$$x'/f' = x/z$$

$$y'/f' = y/z$$

$$x' = f' \cdot x/z$$

$$y' = f' \cdot y/z$$

Side Modified from
Forsyth and Ponce

Homogeneous coordinates

(x', y') : image coordinates

(x, y, z) : world coordinates

$$x'/f' = x/z$$

$$y'/f' = y/z$$

$$x' = f' \cdot x/z$$

$$y' = f' \cdot y/z$$

Is this a linear transformation?

- no—division by z is nonlinear

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image
coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous scene
coordinates

$$(x, y, z) \rightarrow (f \frac{x}{z}, f \frac{y}{z})$$

Scene point \rightarrow Image coordinates

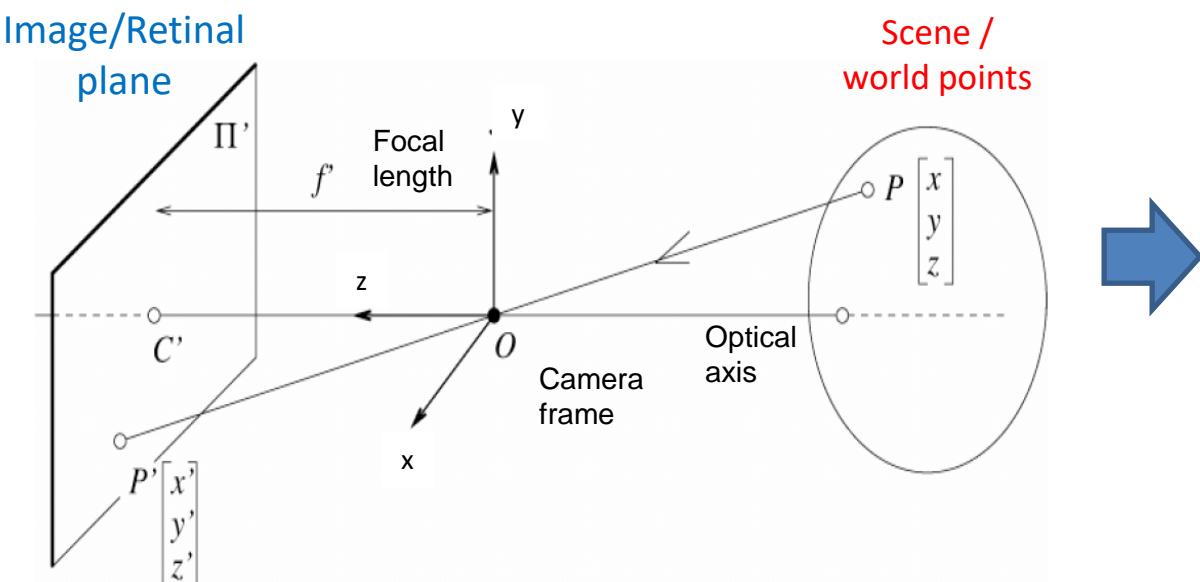
Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Perspective Projection Matrix

- Projection is a matrix multiplication using homogeneous coordinates:



$$(x, y, z) \rightarrow (f \frac{x}{z}, f \frac{y}{z})$$

Scene point → Image coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f' & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ z/f' \end{bmatrix} \Rightarrow (f \frac{x}{z}, f \frac{y}{z})$$

Scene point -> Image coordinates

divide by the third coordinate to convert back to non-homogeneous coordinates

Weak perspective

- Approximation: treat magnification as constant
- Assumes scene depth \ll average distance to camera

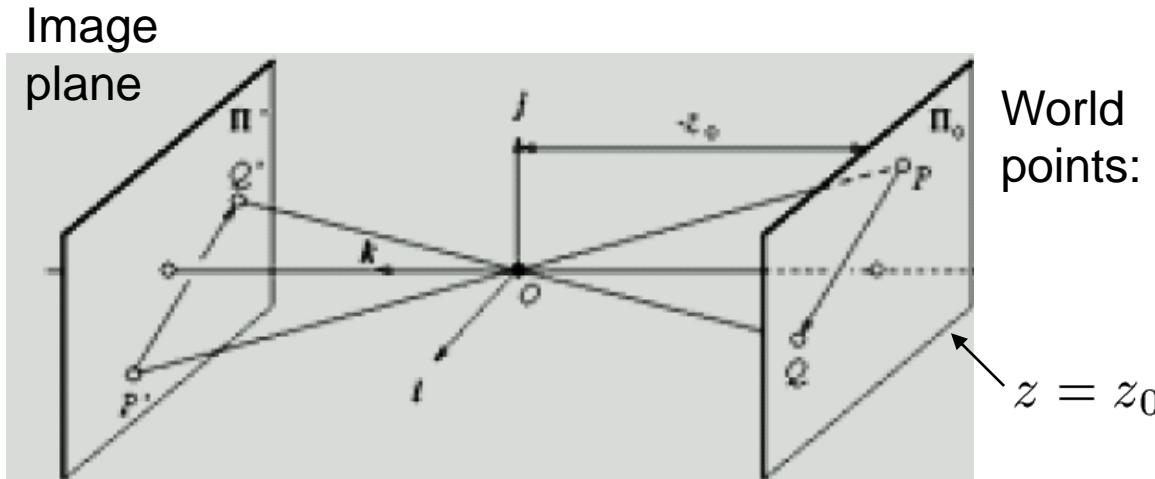
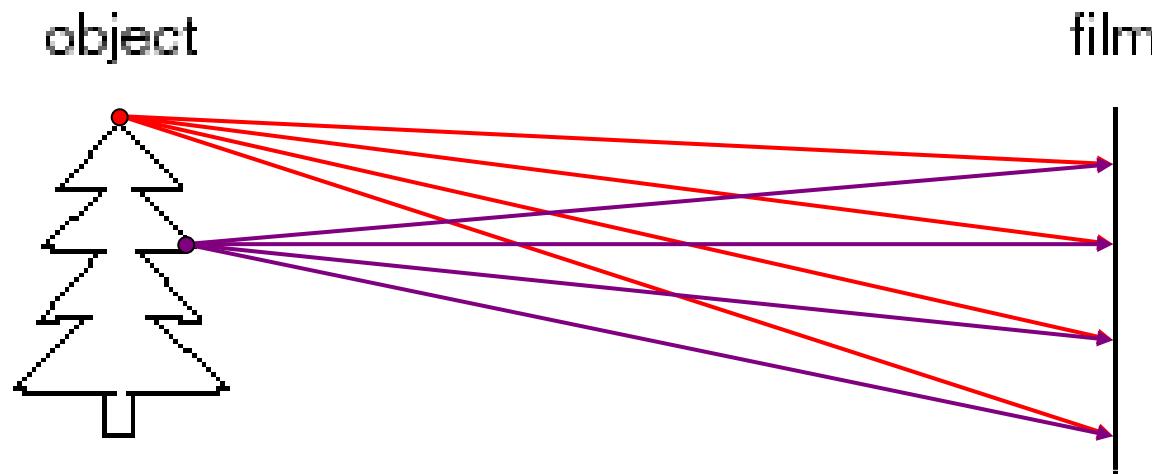
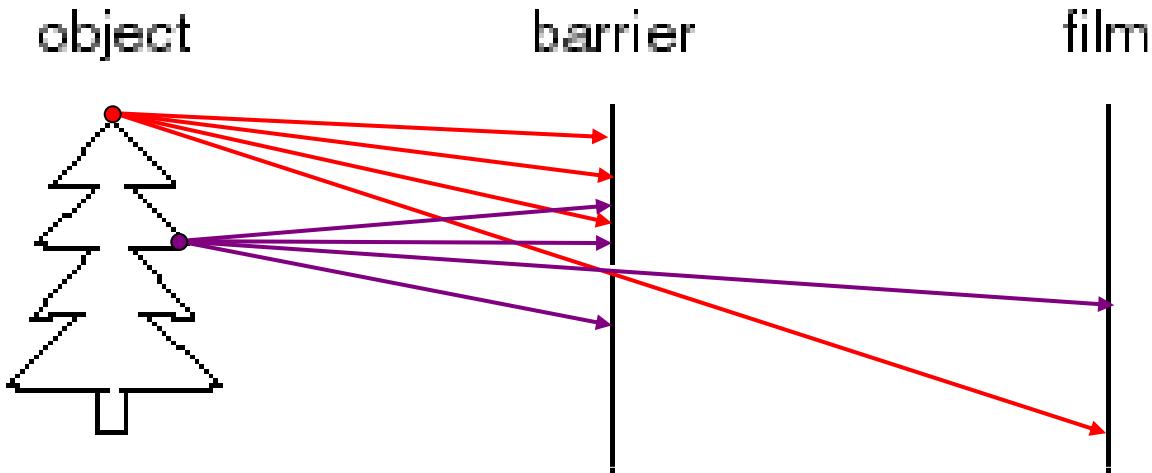


Image formation



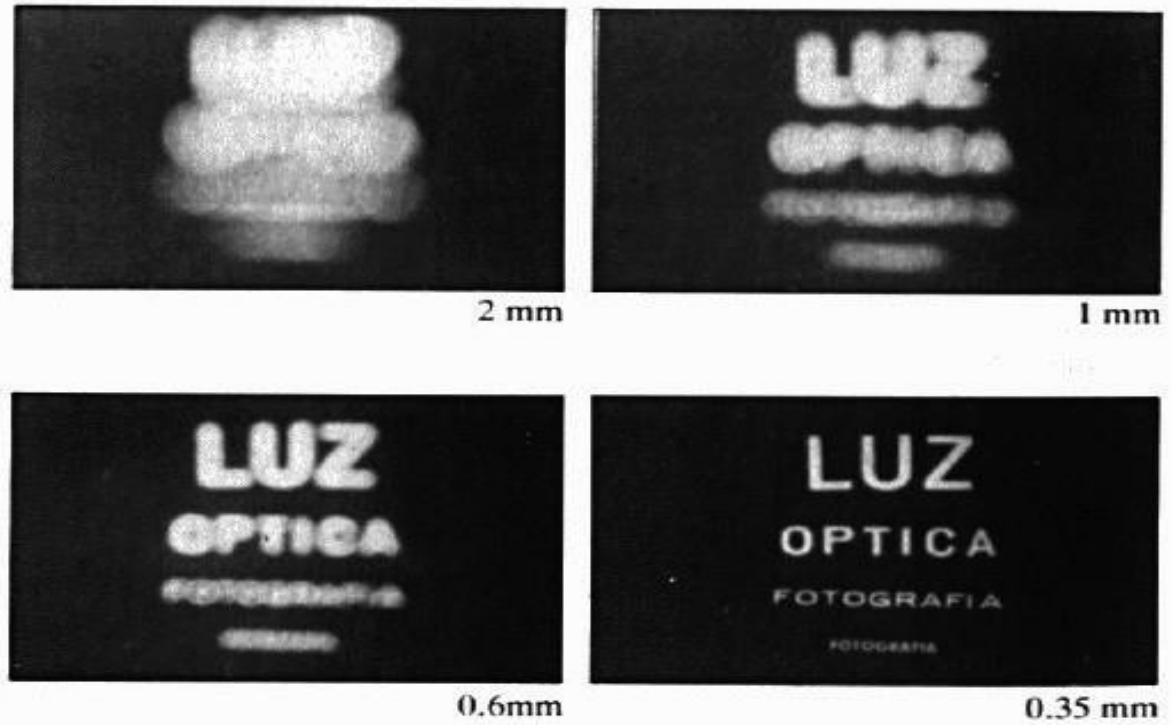
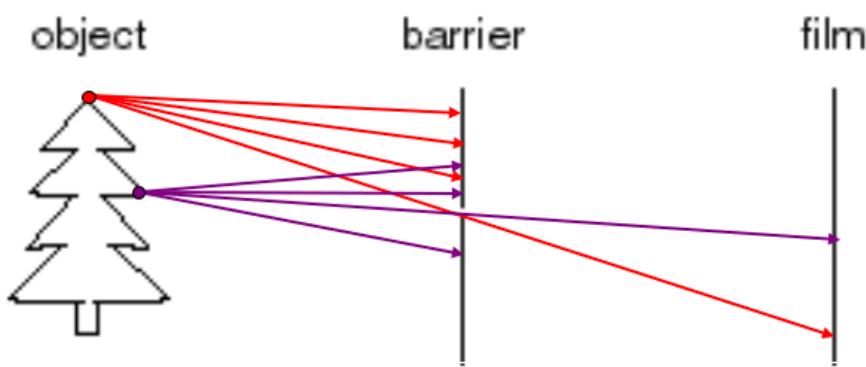
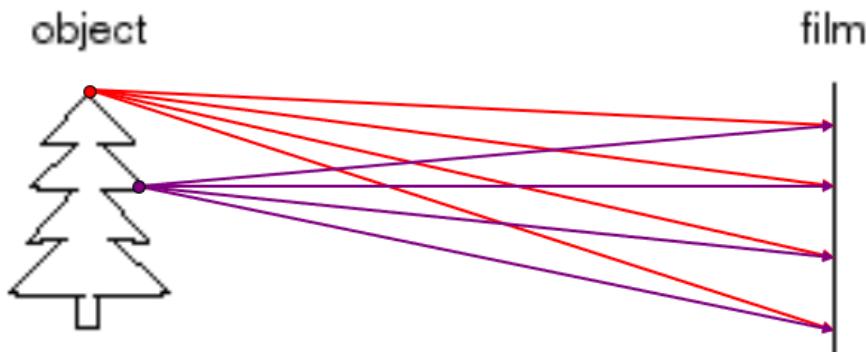
- Let's design a camera
 - Idea 1: put a piece of film in front of an object
 - Do we get a reasonable image?

Pinhole camera



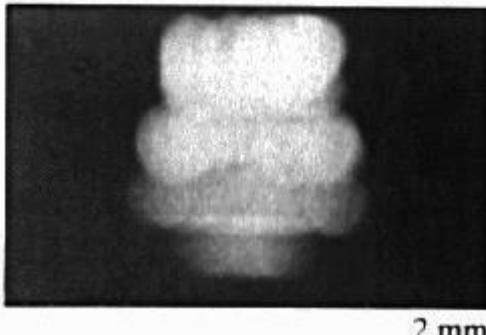
- Add a barrier to block off most of the rays
 - This reduces blurring
 - The opening known as the **aperture**
 - How does this transform the image?

Shrinking the aperture



- Why not make the aperture as small as possible?
 - Less light gets through
 - *Diffraktion* effects...

Shrinking the aperture



2 mm



1 mm



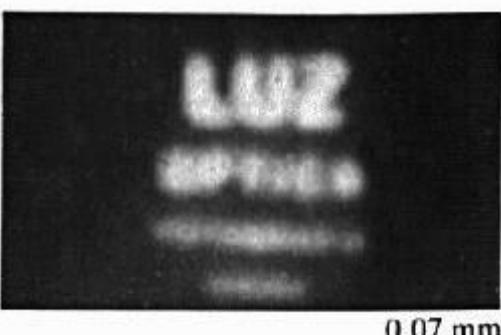
0.6mm



0.35 mm



0.15 mm



0.07 mm

Diffraction of Light Through an Aperture

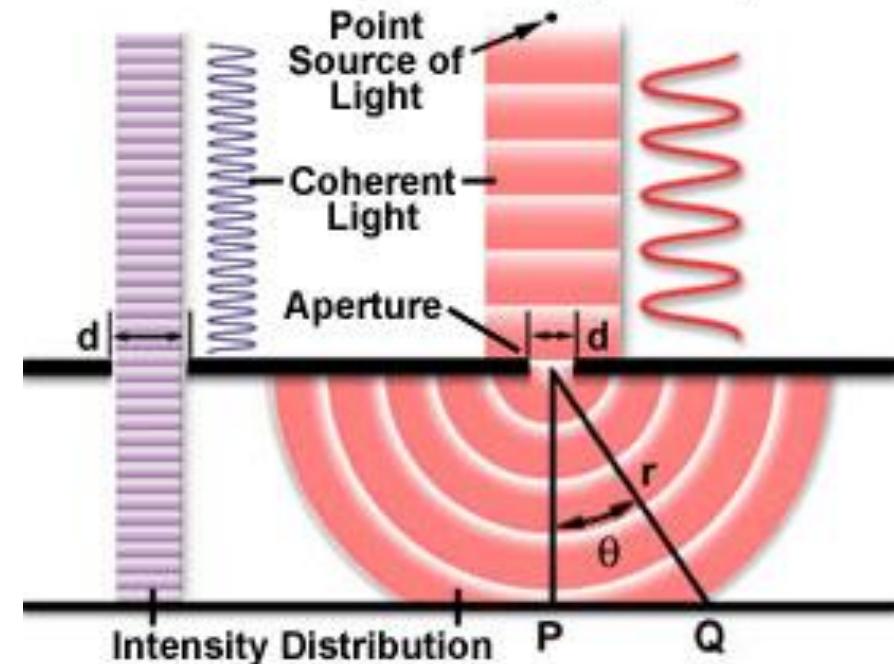
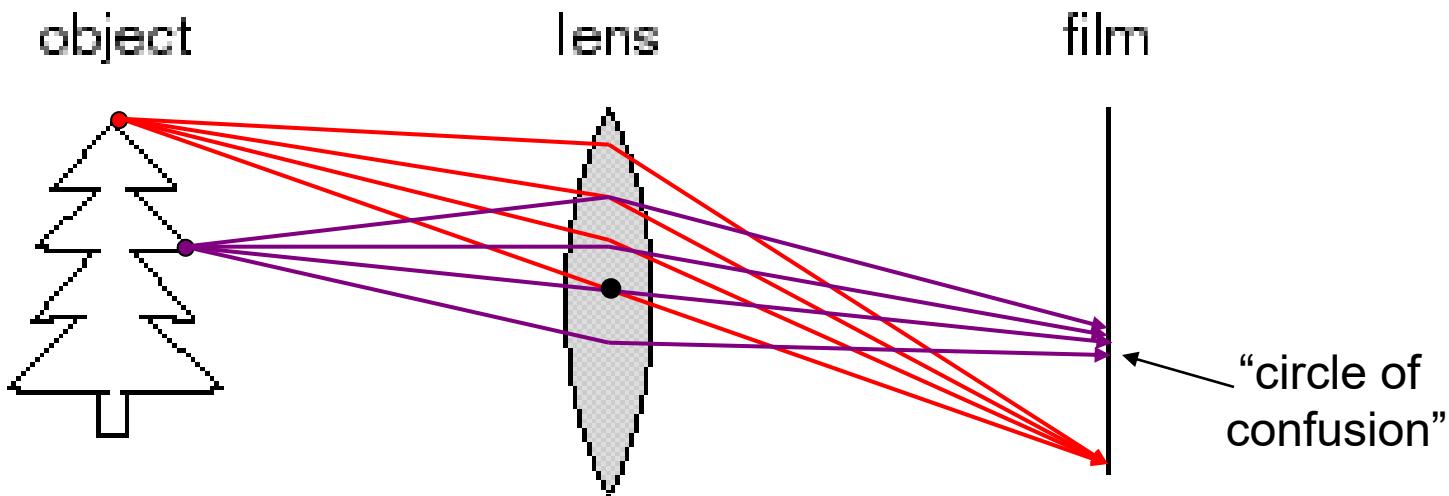


Figure 3

<https://micro.magnet.fsu.edu/primer/lightandcolor/diffractionintro.html>

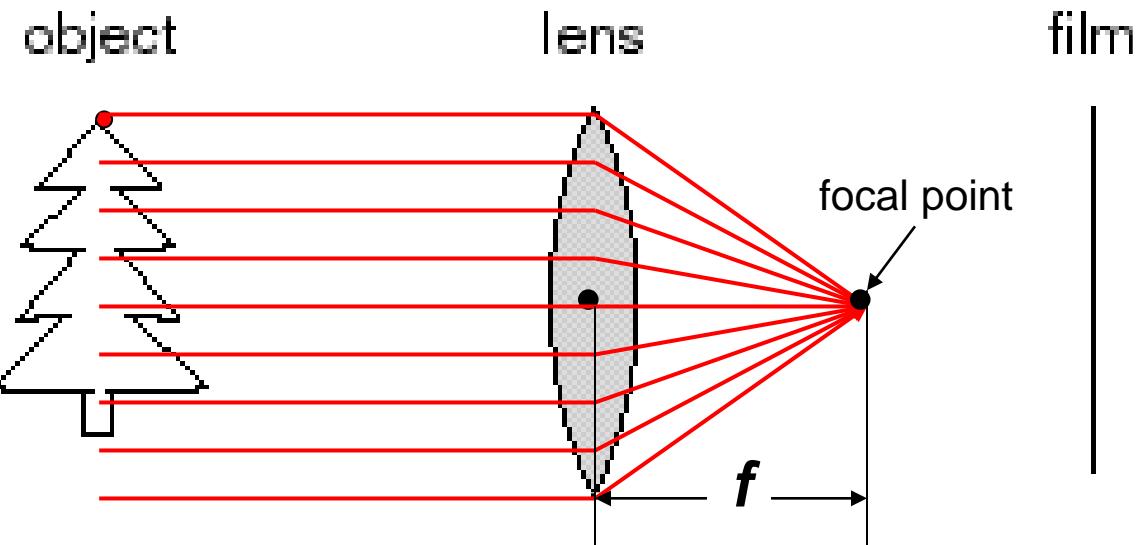
Slide modified from Seitz&Szeliski

Adding a lens



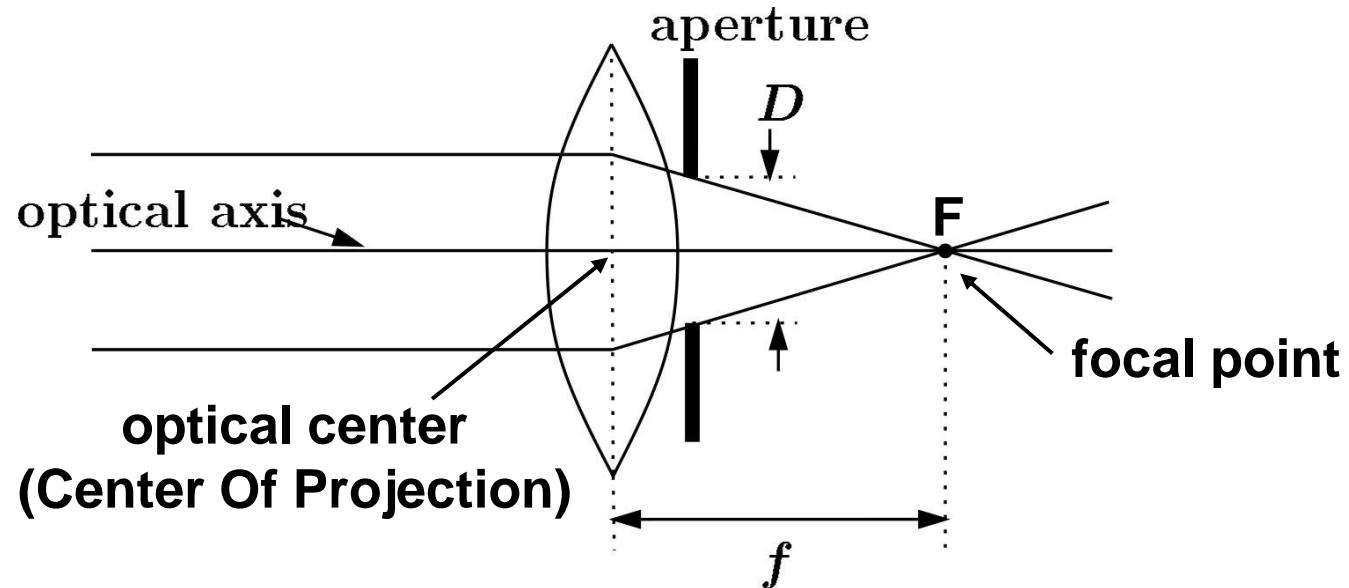
- A lens focuses light onto the film
 - There is a specific distance at which objects are “in focus”
 - other points project to a “circle of confusion” in the image
 - Changing the shape of the lens changes this distance

Adding a lens



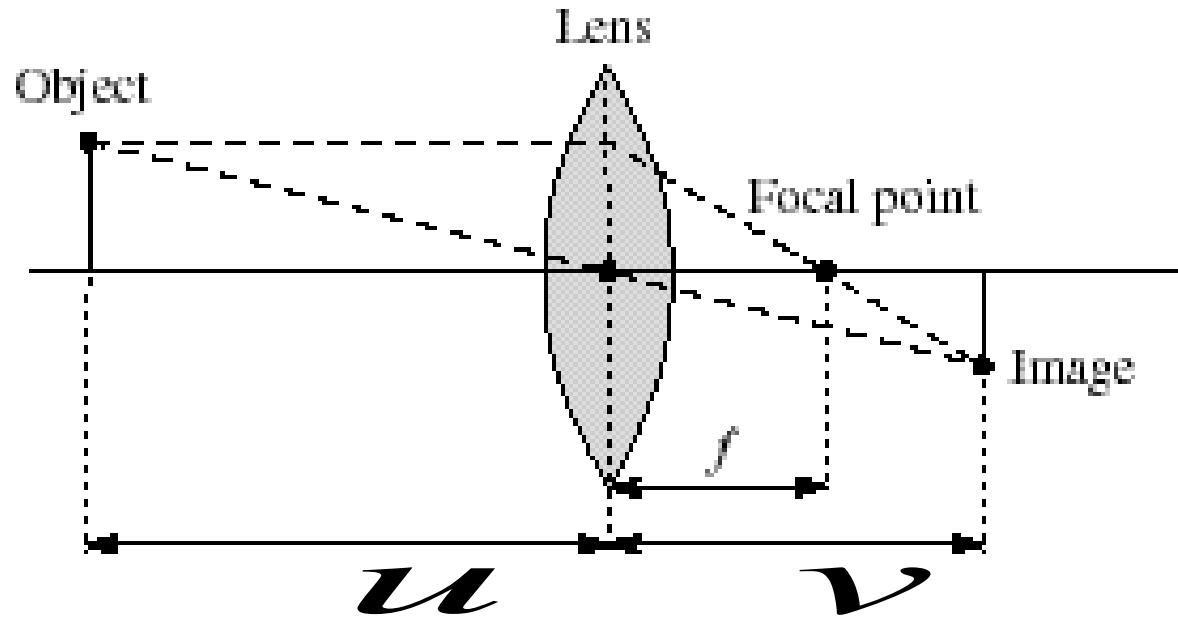
- A lens focuses light onto the film
 - Rays passing through the center are not deviated
 - All parallel rays converge to one point on a plane located at the *focal length* f

Cameras with lenses



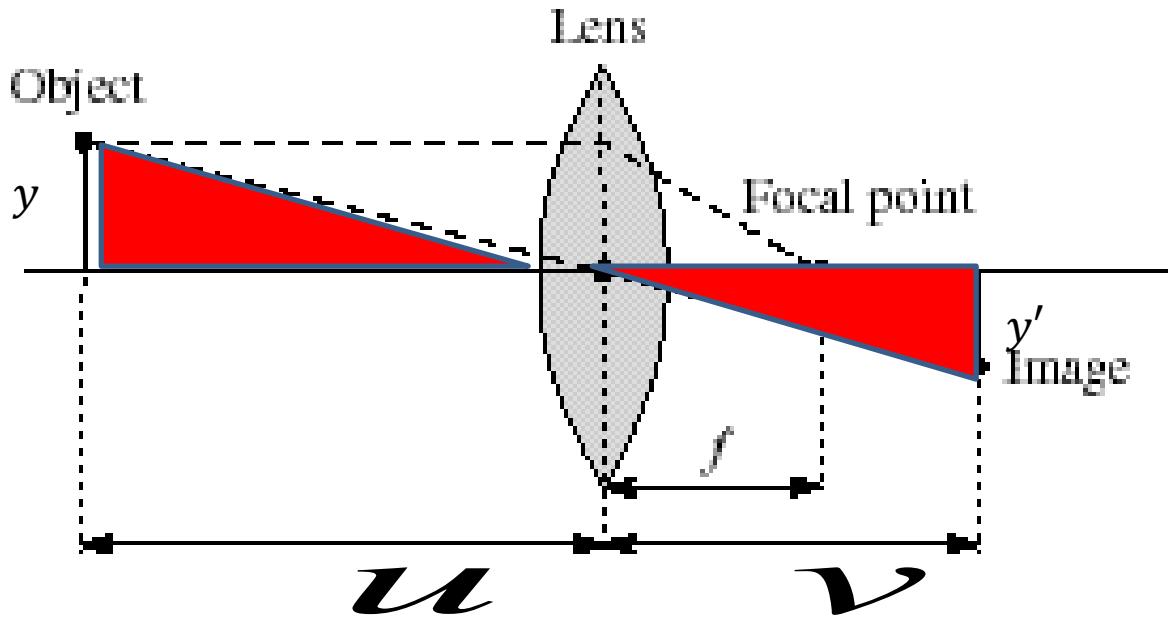
- A lens focuses parallel rays onto a single focal point
- Gather more light, while keeping focus; make pinhole perspective projection practical

Thin lens equation



- How to relate u & v given f
- u : distance of object from optical center
- v : distance at which it will be in focus
- f : focal length

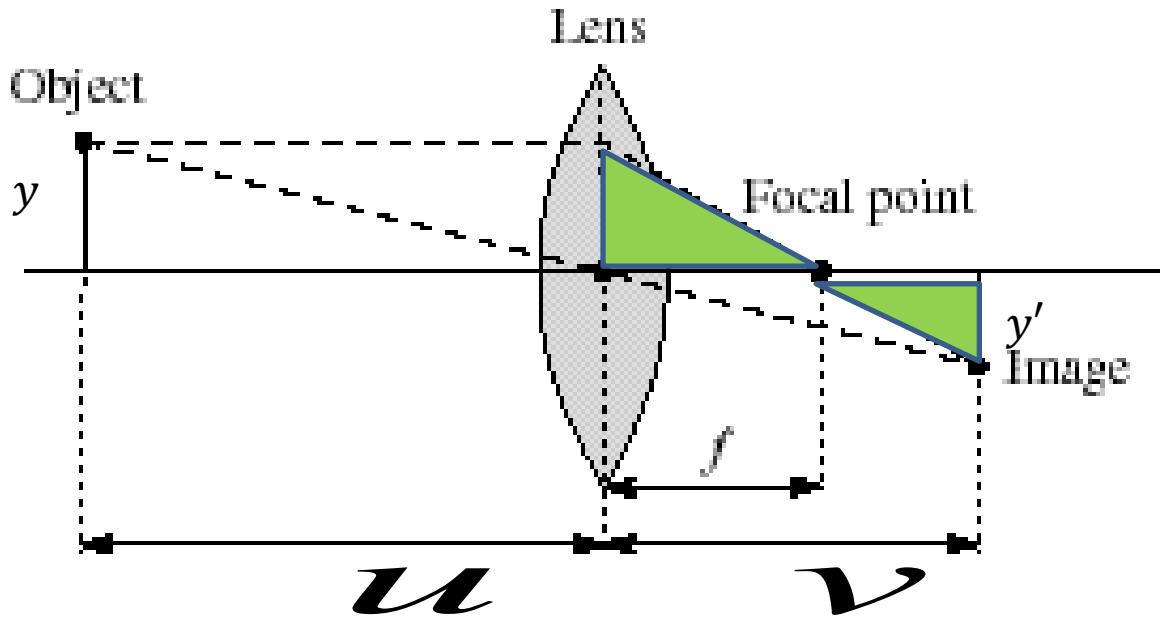
Thin lens equation



$$\frac{y'}{y} = \frac{v}{u}$$

- How to relate u & v given f
- u : distance of object from optical center
- v : distance at which it will be in focus
- f : focal length

Thin lens equation

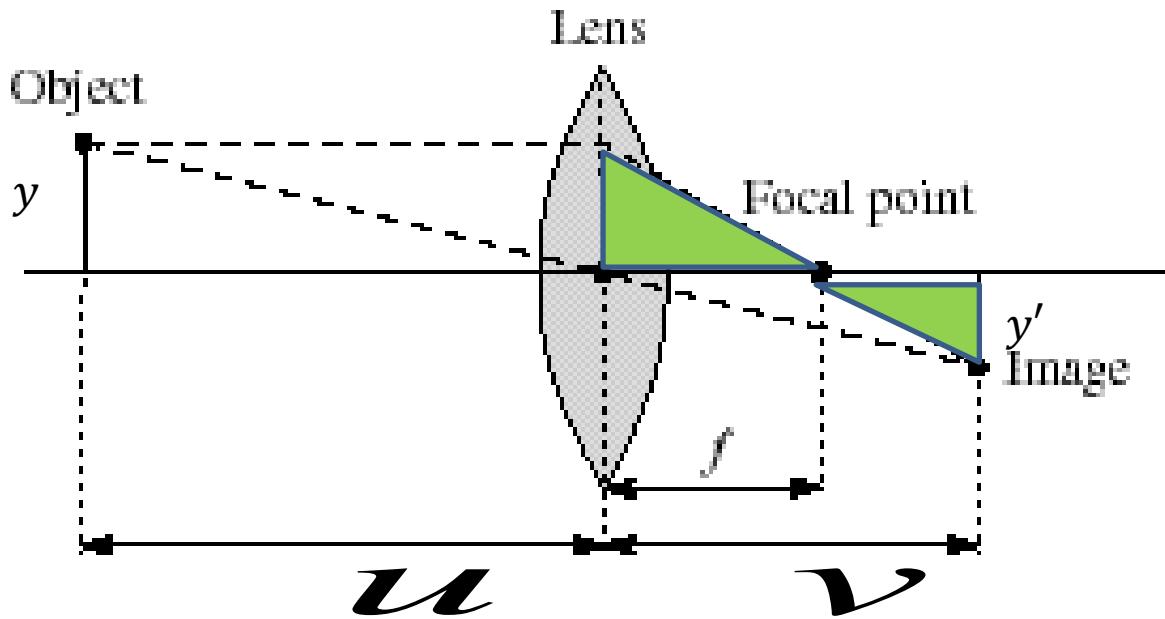


$$\frac{y'}{y} = \frac{v}{u}$$

$$\frac{y'}{y} = \frac{(v - f)}{f}$$

- How to relate u & v given f
- u : distance of object from optical center
- v : distance at which it will be in focus
- f : focal length

Thin lens equation



$$\frac{y'}{y} = \frac{v}{u}$$

$$\frac{y'}{y} = \frac{(v - f)}{f}$$

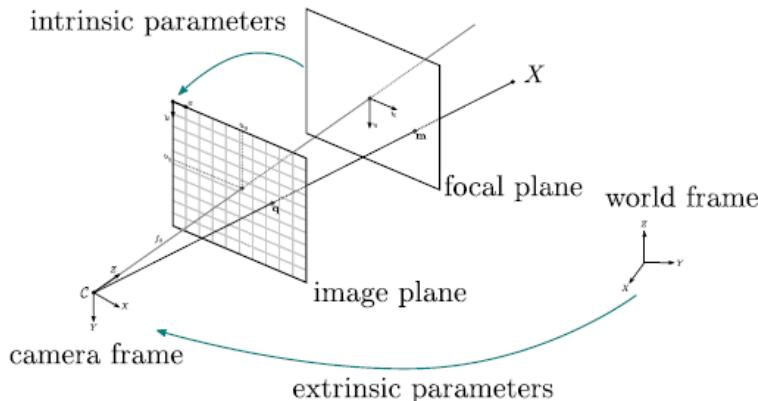
$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

- Any object point satisfying this equation is in focus

Camera parameters

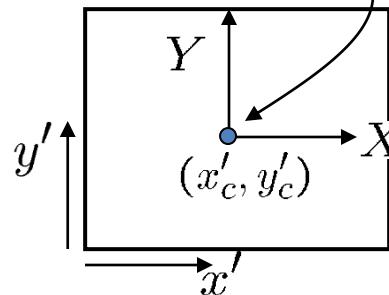
A camera is described by several parameters

- Translation \mathbf{T} of the optical center from the origin of world coords
- Rotation \mathbf{R} of the image plane
- focal length f , principle point (x'_c, y'_c) , pixel size (s_x, s_y)
- blue parameters are called “**extrinsics**,” red are “**intrinsics**”



Projection equation

$$\mathbf{X} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \boldsymbol{\Pi} \mathbf{X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$\boldsymbol{\Pi} = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$$

intrinsics projection rotation translation identity matrix

- The definitions of these parameters are **not** completely standardized
 - especially intrinsics—varies from one book to another

Pinhole Camera Model

$$sp = K[R|t]P$$

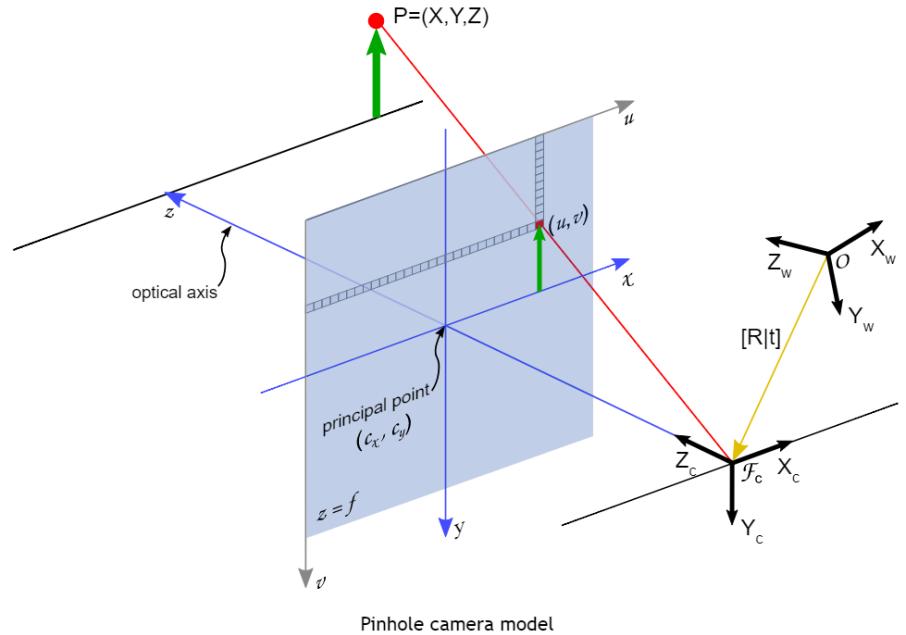
or

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$(x_d, y_d) = L(\tilde{x}, \tilde{y})$$

where:

- (X, Y, Z) are the coordinates of a 3D point in world space.
- (u, v) are the coordinates (in pixels) of the projection of (X, Y, Z) on the image plane.
- K is a 3x3 matrix of intrinsic camera parameters.
- $[R|t]$ is a 3x4 matrix of extrinsic camera parameters, mapping world space to camera space. It is composed of a 3D rotation followed by translation.
- (c_x, c_y) is the camera's principal point in pixels, where its origin is projected on the image plane. Usually is at the image center.
- f_x, f_y are the camera's horizontal and vertical focal lengths, respectively, expressed in pixel units.
- s is a scale factor.



Intrinsic and Extrinsic Camera Parameters

- Nice description and illustrations by Aqeel Anwar:

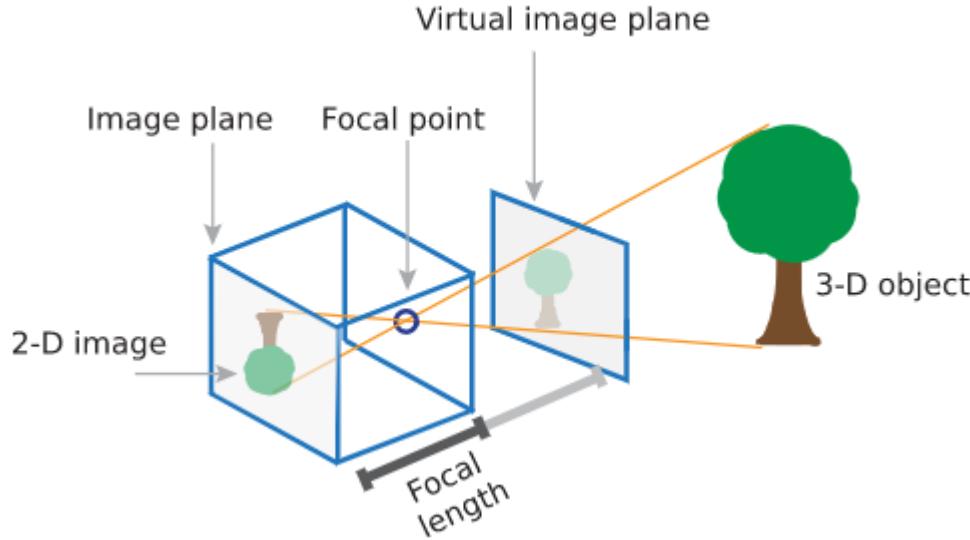
<https://towardsdatascience.com/what-are-intrinsic-and-extrinsic-camera-parameters-in-computer-vision-7071b72fb8ec>

Depth Estimation Stereo Correspondence

- Szeliski Computer Vision Book 1st Edition Chapter 11
- Szeliski Computer Vision Book 2nd Edition Chapter 12.1
 - Many slides from Noah Snavely

Review of Image Formation Camera Parameters

Szeliski Chapter 2.1.4 3D to 2D Projections --- Camera Matrix



Light rays pass through the aperture and project an inverted image on the opposite side of the camera. Think of the virtual image plane as being in front of the camera and containing the upright image of the scene.

$$\text{Scale factor } \mathbf{W} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

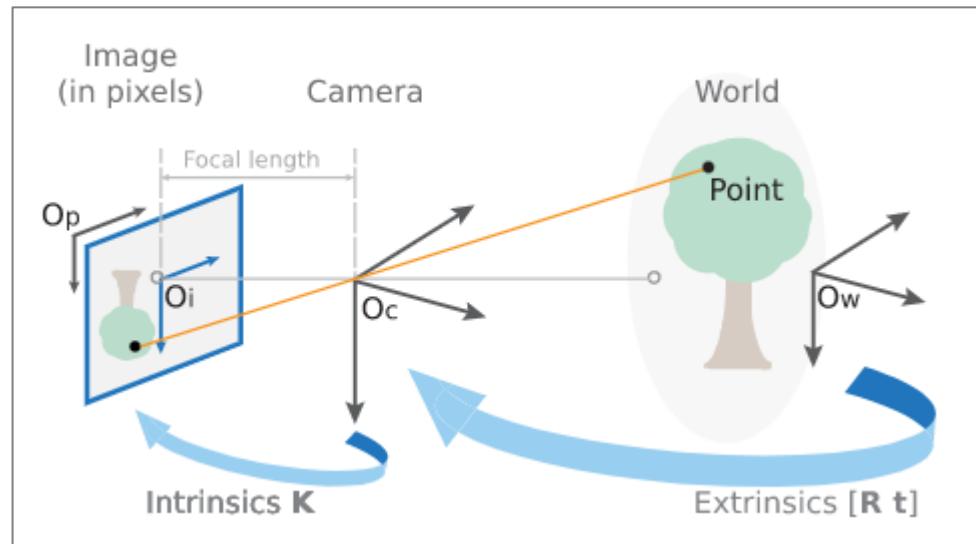
Image points World points

$\mathbf{P} = \mathbf{K}[\mathbf{R} \ \mathbf{t}]$

Camera matrix Extrinsics
Intrinsics matrix Rotation and Translation

- The pinhole camera parameters are represented in a 3-by-4 matrix called the **camera matrix**.
- **Camera matrix:** maps the 3-D world scene into the image plane.
- **The calibration algorithm** calculates the camera matrix using the extrinsic and intrinsic parameters.
- **Extrinsic parameters:** represent the location of the camera in the 3-D scene.
- **Intrinsic parameters:** represent the optical center and focal length of the camera.

Szeliski Chapter 2.1.4 3D to 2D Projections --- Camera Matrix



$$\text{Scale factor } W \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

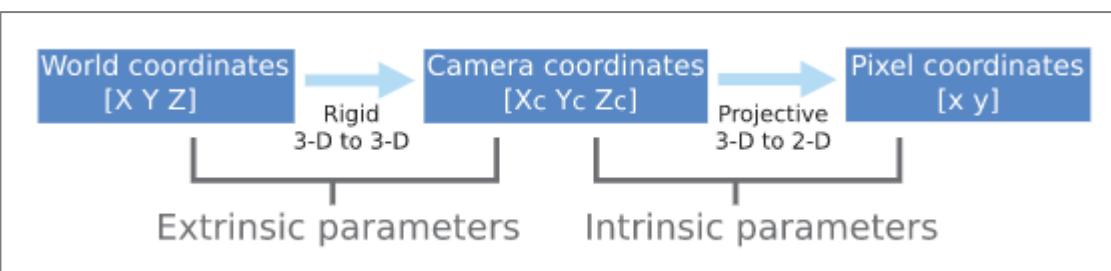
Image points World points

$P = K[R \ t]$

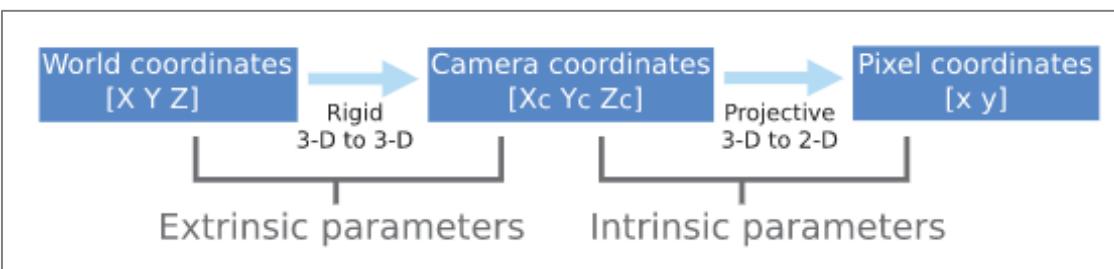
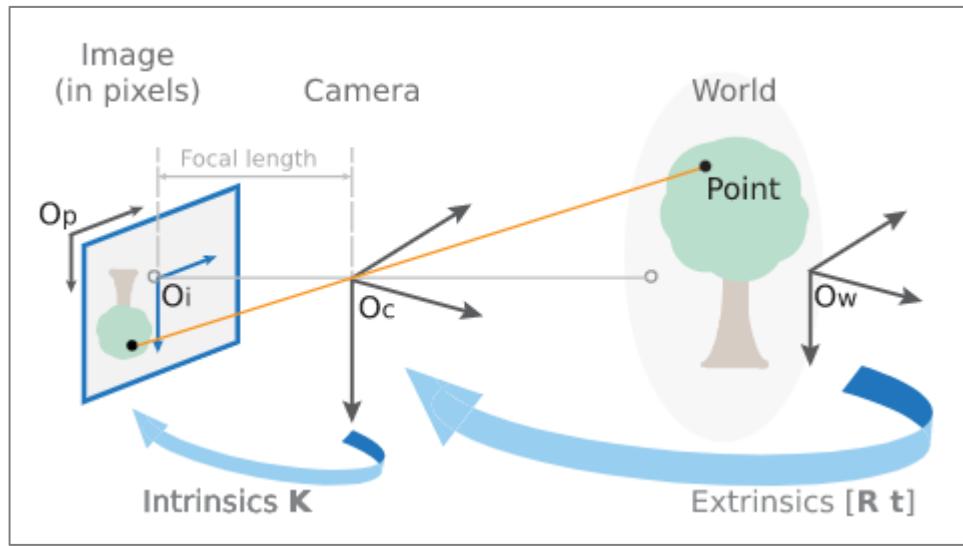
Camera matrix Intrinsics matrix

Extrinsics Rotation and Translation

- The pinhole camera parameters are represented in a **3-by-4** matrix called the **camera matrix**.
- **Camera matrix:** maps the 3-D world scene into the image plane.
- **The calibration algorithm** calculates the camera matrix using the extrinsic and intrinsic parameters.
- **Extrinsic parameters:** represent the location of the camera in the 3-D scene.
- **Intrinsic parameters:** represent the optical center and focal length of the camera.



Szeliski Chapter 2.1.4 3D to 2D Projections --- Camera Matrix



$$w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

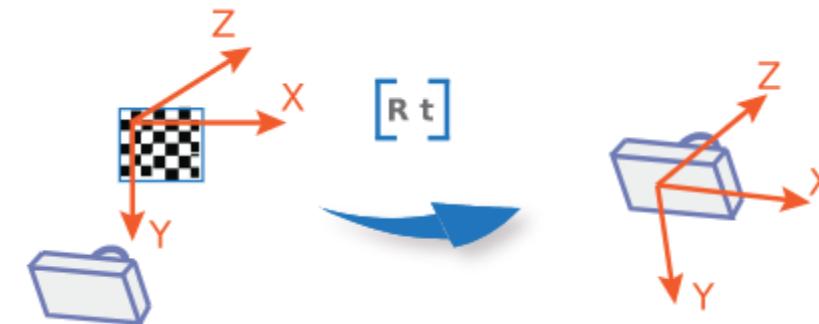
Scale factor Image points World points

$$P = K[R \ t]$$

Camera matrix Intrinsics matrix Extrinsic Rotation and Translation

Extrinsic Parameters

- The extrinsic parameters consist of a rotation, R , and a translation, t .
- The origin of the camera's coordinate system is at its optical center and its x - and y -axis define the image plane.



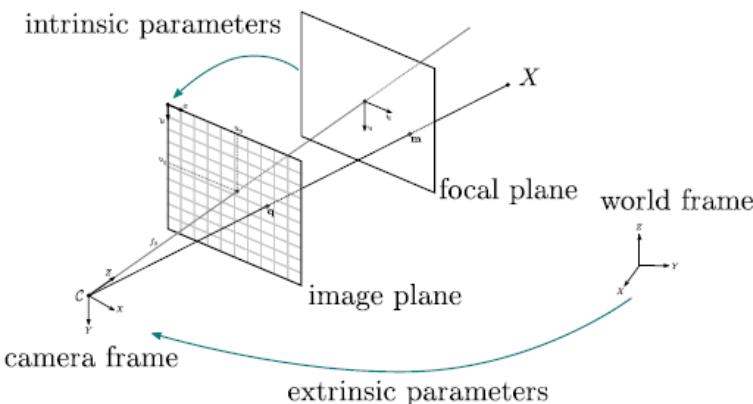
Intrinsic Parameters

- The intrinsic parameters include the focal length, the optical center, also known as the *principal point*, and the skew coefficient.

Camera parameters

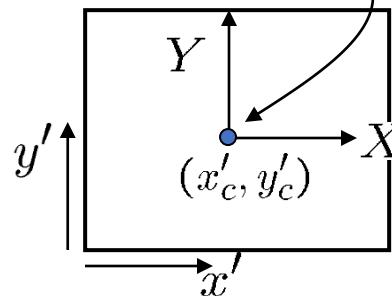
A camera is described by several parameters

- Translation \mathbf{T} of the optical center from the origin of world coords
- Rotation \mathbf{R} of the image plane
- focal length f , principle point (x'_c, y'_c) , pixel size (s_x, s_y)
- blue parameters are called “extrinsics,” red are “intrinsics”



Projection equation

$$\mathbf{x} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \boldsymbol{\Pi} \mathbf{X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

$$\boldsymbol{\Pi} = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$$

intrinsics projection rotation translation identity matrix

- The definitions of these parameters are **not** completely standardized
 - especially intrinsics—varies from one book to another

Pinhole Camera Model

$$sp = K[R|t]P$$

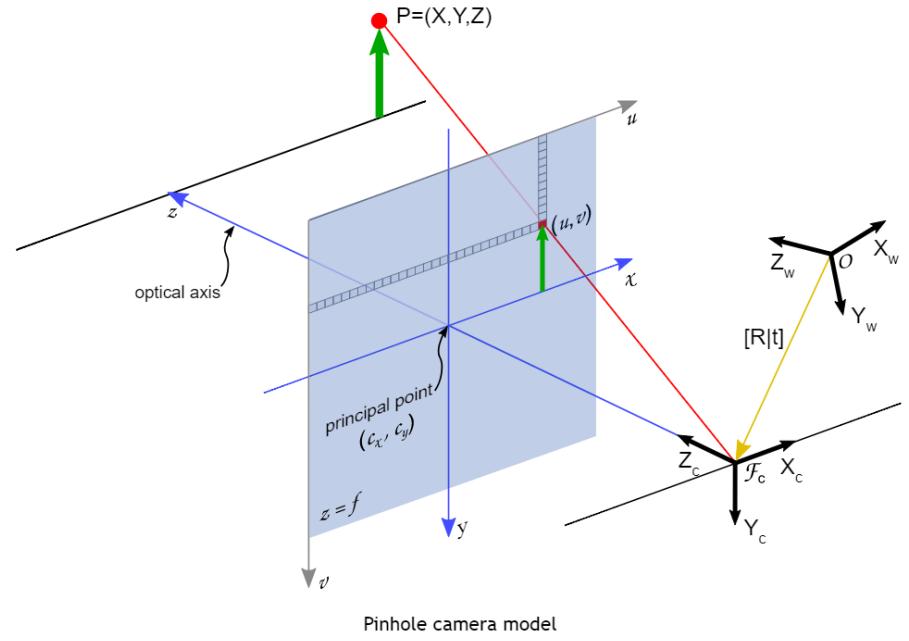
or

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$(x_d, y_d) = L(\tilde{x}, \tilde{y})$$

where:

- (X, Y, Z) are the coordinates of a 3D point in world space.
- (u, v) are the coordinates (in pixels) of the projection of (X, Y, Z) on the image plane.
- K is a 3x3 matrix of intrinsic camera parameters.
- $[R|t]$ is a 3x4 matrix of extrinsic camera parameters, mapping world space to camera space. It is composed of a 3D rotation followed by translation.
- (c_x, c_y) is the camera's principal point in pixels, where its origin is projected on the image plane. Usually is at the image center.
- f_x, f_y are the camera's horizontal and vertical focal lengths, respectively, expressed in pixel units.
- s is a scale factor.



Depth Estimation Stereo Correspondence

- Szeliski Computer Vision Book 1st Edition Chapter 11 --- Stereo Correspondence
- Szeliski Computer Vision Book 2nd Edition Chapter 12.1 --- Depth Estimation
- Many slides from Noah Snavely

Szeliski 2nd Edition Chapter 12: Depth Estimation

1. How to build a more complete 3D model, e.g., a sparse or dense depth map that assigns relative depths to pixels in the input images.
2. Multi-view stereo algorithms that produce complete 3D volumetric or surface-based object models, as well
3. Monocular depth recovery algorithms that infer plausible depths from just a single image.

Szeliski 2nd Edition Chapter 12: Depth Estimation

1. How to build a more complete 3D model, e.g., a sparse or dense depth map that assigns relative depths to pixels in the input images.
2. Multi-view stereo algorithms that produce complete 3D volumetric or surface-based object models, as well
3. Monocular depth recovery algorithms that infer plausible depths from just a single image

Szeliski Figure 12.1

Depth estimation algorithms can convert a pair of color images (a–b) into a depth map (c) (Scharstein, Hirschmüller et al. 2014) © 2014 Springer



Convert a sequence of images into a 3D model (Knapitsch, Park et al. 2017) © 2017 ACM



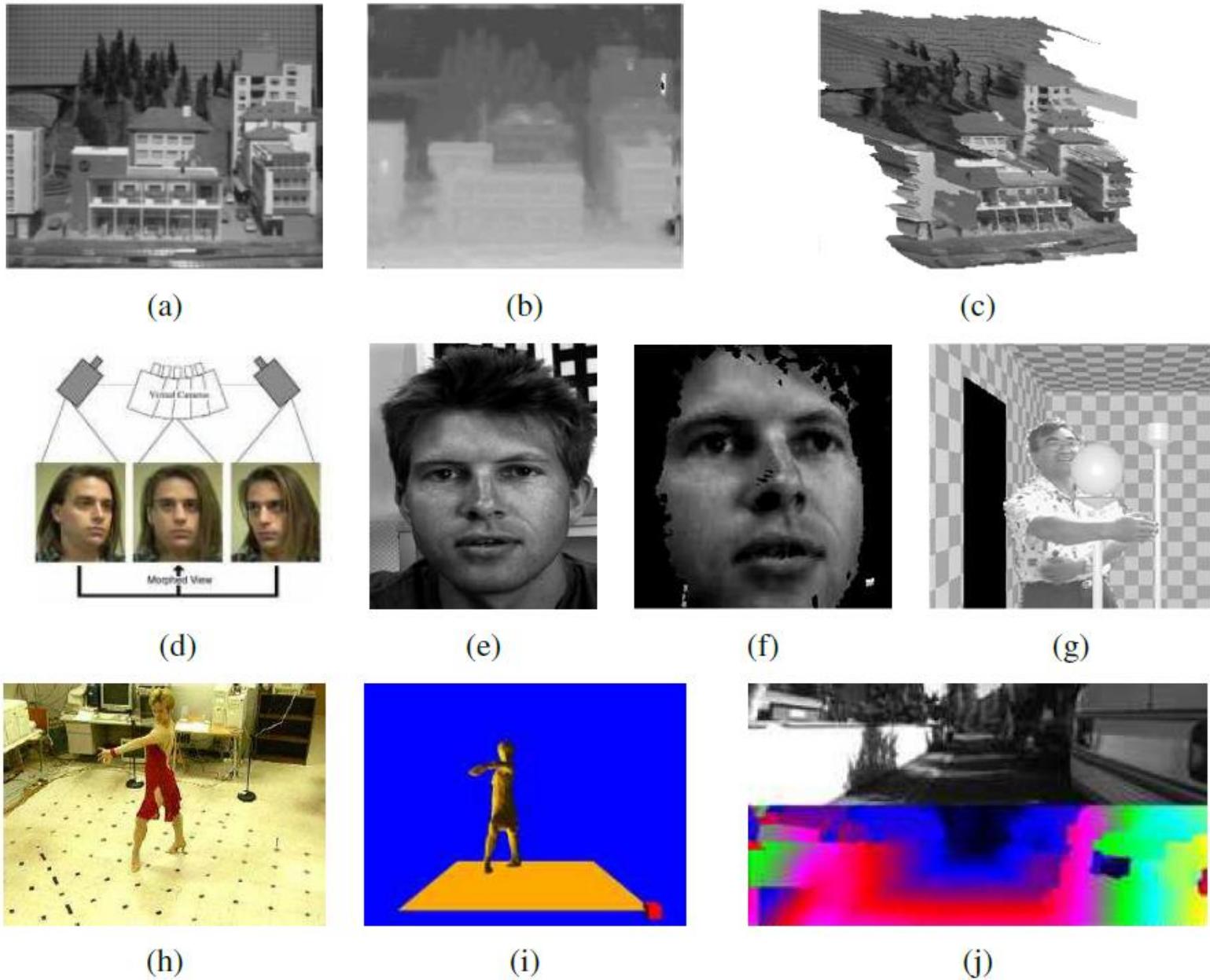
Convert a single image into a depth map (Li, Dekel et al. 2019) © 2019 IEEE.



Applications of Stereo Vision

Szeliski Figure 12.2

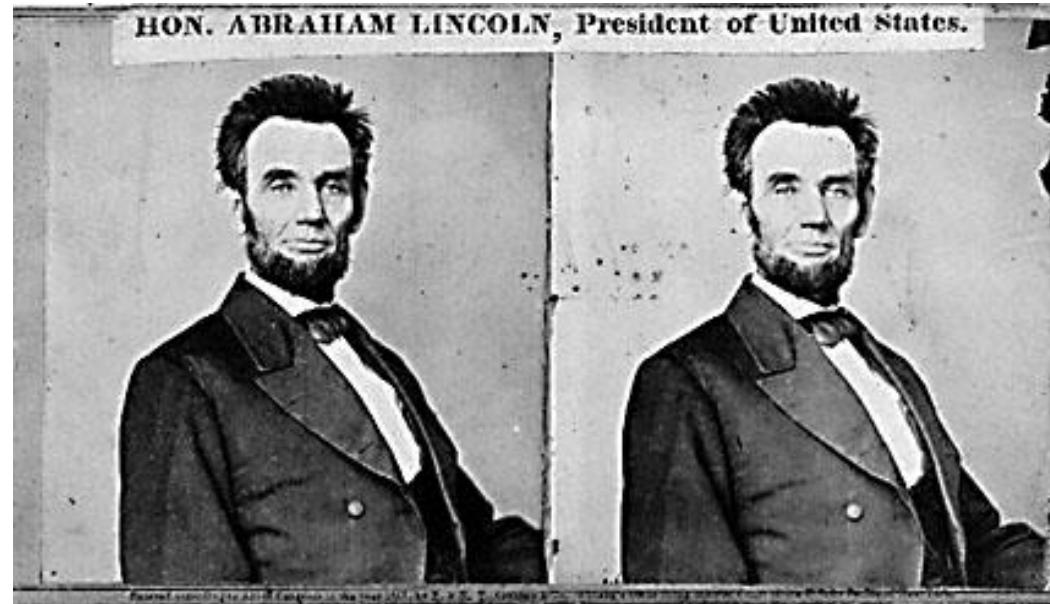
- a) input image,
- b) computed depth map, and
- c) new view generation from multi-view stereo (Matthies, Kanade, and Szeliski 1989) © 1989 Springer;
- d) view morphing between two images (Seitz and Dyer 1996) © 1996 ACM;
- e-f) 3D face modeling (images courtesy of Fr'ed'eric Devernay);
- g) z-keying live and computer generated imagery (Kanade, Yoshida et al. 1996) © 1996 IEEE;
- h-i) building 3D surface models from multiple video streams in Virtualized Reality (Kanade, Rander, and Narayanan 1997) © 1997 IEEE;
- j) computing depth maps for autonomous navigation (Geiger, Lenz, and Urtasun 2012) © 2012 IEEE.



Stereo

Stereo Matching: Process of taking two or more images and building a 3D model of the scene

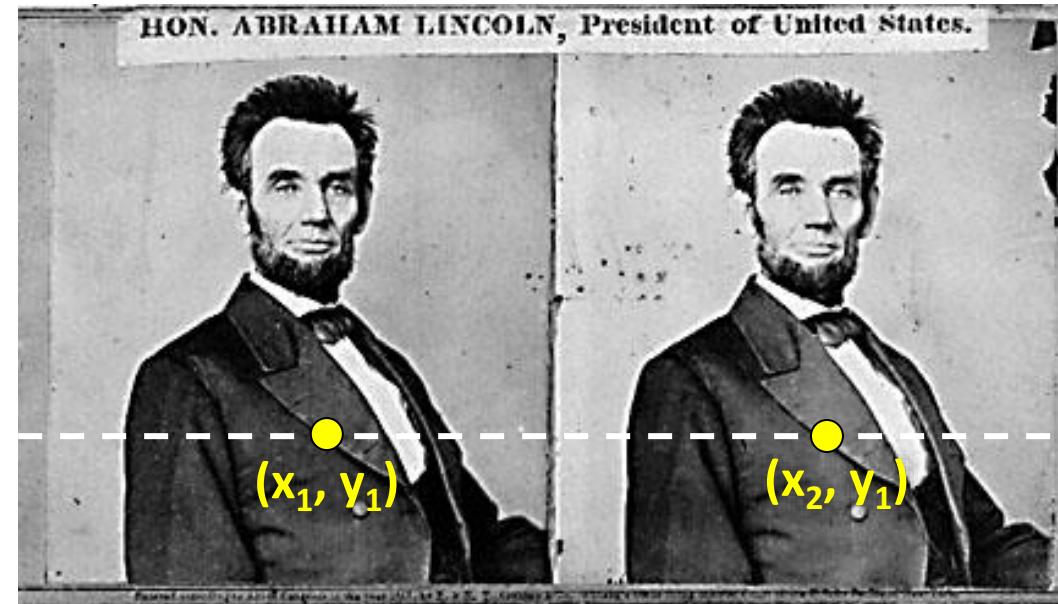
1. Finding matching pixels in the images and
2. Converting their 2D positions into 3D depths.



- Given two images from different viewpoints
 - How can we compute the depth of each point in the image?
 - Based on *how much each pixel moves* between the two images

Slide credit:
Noah Snavely

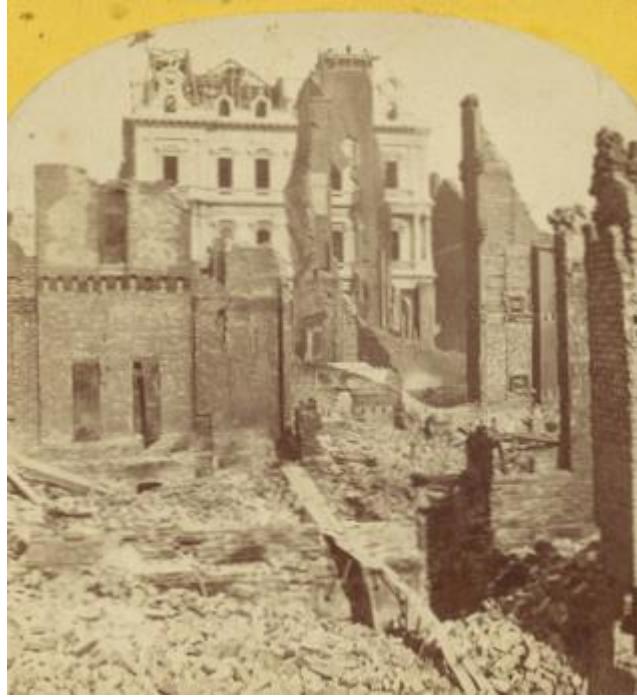
Epipolar geometry



Two images captured by a purely horizontal translating camera (*rectified* stereo pair)

$$x_2 - x_1 = \text{the } \textbf{disparity} \text{ of pixel } (x_1, y_1)$$

Disparity = inverse depth



<http://stereo.nypl.org/view/41729>

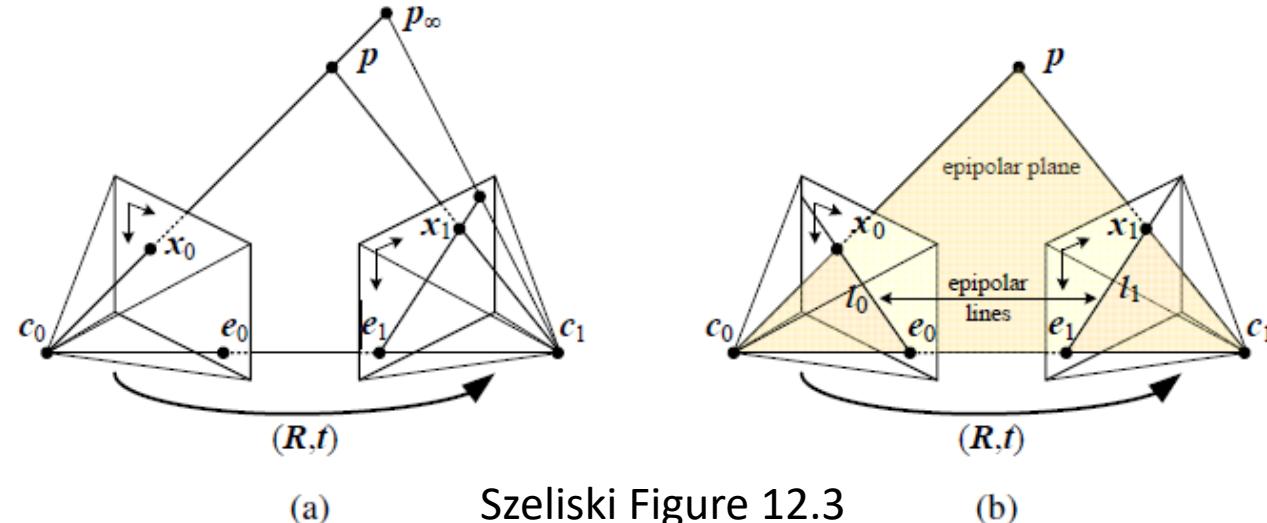
(Or, hold a finger in front of your face and wink each eye in succession.)

Slide credit:
Noah Snavely

Epipolar Geometry

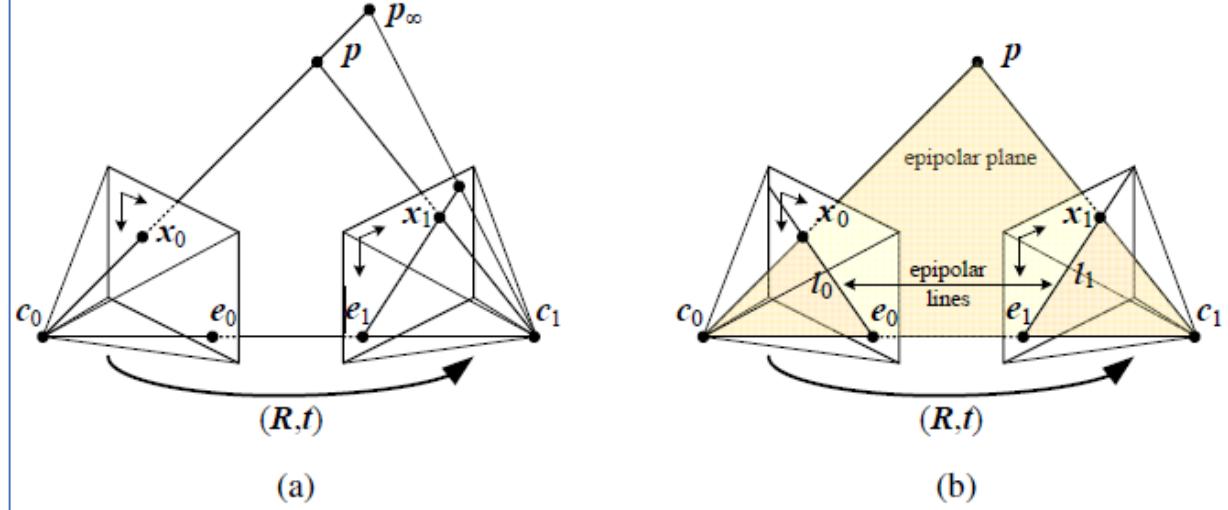
Stereo Matching: Process of taking two or more images and building a 3D model of the scene

- **Given a pixel in one image, how can we compute its correspondence in the other image?**
 - **Earlier:** we saw that a variety of search techniques can be used to match pixels based on their local appearance as well as the motions of neighboring pixels.
 - **Stereo matching:** we have some additional information for the cameras:
 - positions and
 - calibration data
 - How can we exploit this information to reduce the number of potential correspondences ?
 - and hence both speed up the matching and increase its reliability?



Epipolar Geometry

- **p**: point of interest in 3D;
- **c₀** and **c₁**: camera centers
- **x₀** and **x₁**: projection of p to image 0 and image 1
- **Epipole e₁** : projection of the original camera center c₀ into the second camera
- **Epipole e₀** : projection of the original camera center c₁ into the first camera
- -----
- **Pixel in one image x₀** : projects to an epipolar line segment in the other image.
- **Epipolar line segment** :
 - End-1: projection of the original viewing ray at infinity p_{inf}
 - End-2: projection of the original camera center c₀ into the second camera, which is known as the epipole e₁.
- **Project the epipolar line in the second image back into the first**: we get another line (segment), this time bounded by the other corresponding epipole e₀.



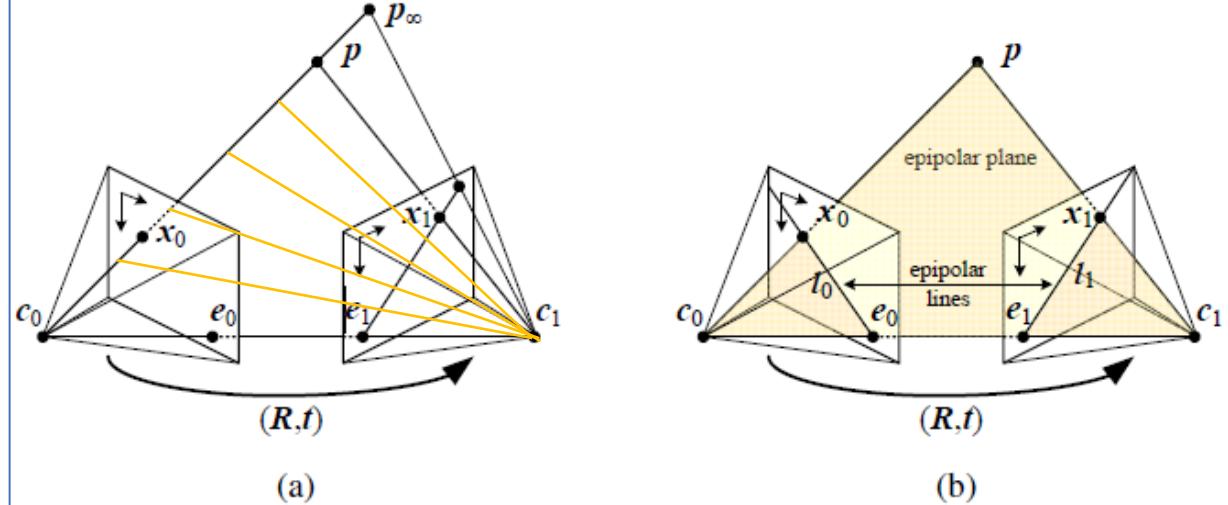
Epipolar geometry:

- (a) epipolar line segment corresponding to one ray;
- (b) corresponding set of epipolar lines and their epipolar plane.

- **Epipolar plane**: passes through both camera centers c_0 and c_1 as well as the point of interest p (Faugeras and Luong 2001; Hartley and Zisserman 2004).
- **Extending both line segments to infinity**: we get a pair of corresponding epipolar lines (Figure 12.3b),
- Epipolar lines are the intersection of the two images with epipolar plane

Epipolar Geometry

- **p**: point of interest in 3D;
- **c₀** and **c₁**: camera centers
- **x₀** and **x₁**: projection of p to image 0 and image 1
- **Epipole e₁** : projection of the original camera center c₀ into the second camera
- **Epipole e₀** : projection of the original camera center c₁ into the first camera
- -----
- **Pixel in one image x₀** : projects to an epipolar line segment in the other image.
- **Epipolar line segment** :
 - End-1: projection of the original viewing ray at infinity p_{inf}
 - End-2: projection of the original camera center c₀ into the second camera, which is known as the epipole e₁.
- **Project the epipolar line in the second image back into the first**: we get another line (segment), this time bounded by the other corresponding epipole e₀.



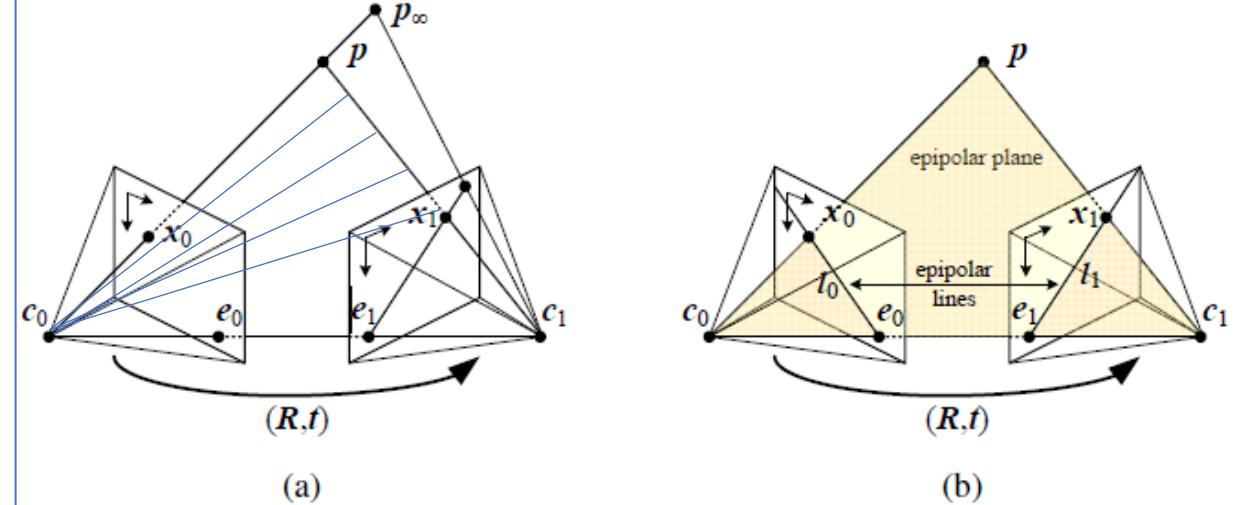
Epipolar geometry:

- (a) epipolar line segment corresponding to one ray;
- (b) corresponding set of epipolar lines and their epipolar plane.

- **Epipolar plane**: passes through both camera centers c₀ and c₁ as well as the point of interest p (Faugeras and Luong 2001; Hartley and Zisserman 2004).
- **Extending both line segments to infinity**: we get a pair of corresponding epipolar lines (Figure 12.3b),
- Epipolar lines are the intersection of the two images with epipolar plane

Epipolar Geometry

- **p**: point of interest in 3D;
- **c₀** and **c₁**: camera centers
- **x₀** and **x₁**: projection of p to image 0 and image 1
- **Epipole e₁** : projection of the original camera center c₀ into the second camera
- **Epipole e₀** : projection of the original camera center c₁ into the first camera
- -----
- **Pixel in one image x₀** : projects to an epipolar line segment in the other image.
- **Epipolar line segment** :
 - End-1: projection of the original viewing ray at infinity p_{inf}
 - End-2: projection of the original camera center c₀ into the second camera, which is known as the epipole e₁.
- **Project the epipolar line in the second image back into the first:** we get another line (segment), this time bounded by the other corresponding epipole e₀.



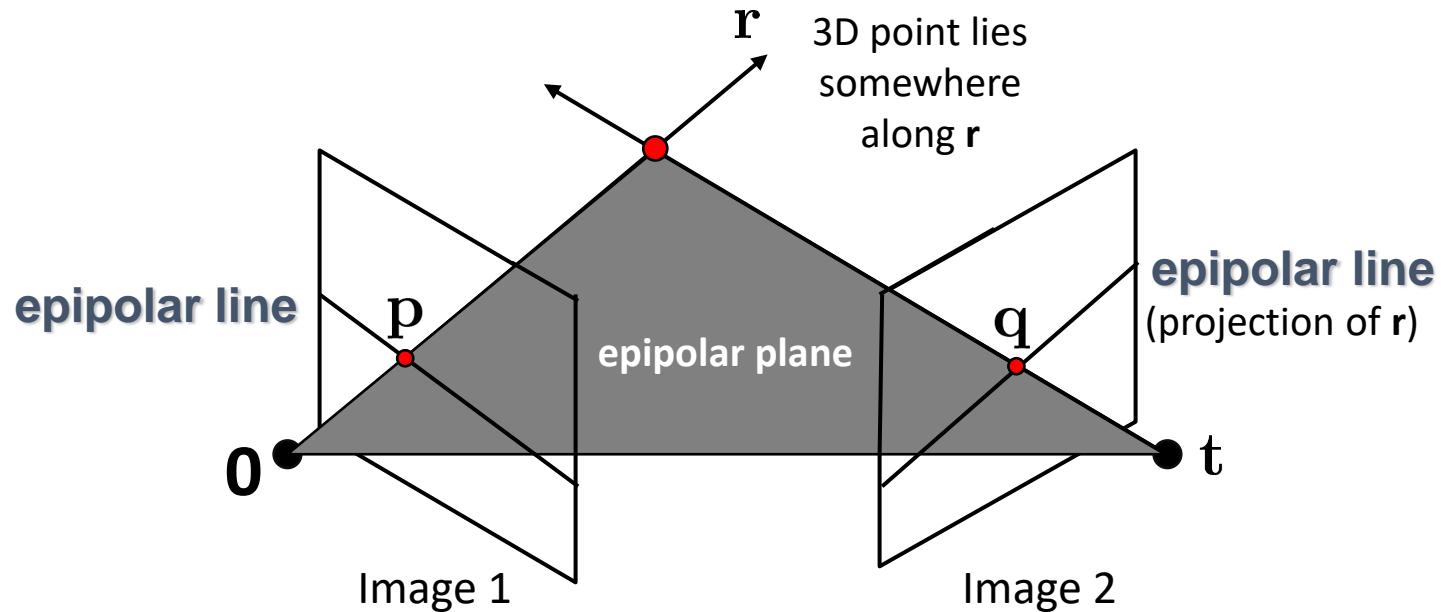
Epipolar geometry:

- (a) epipolar line segment corresponding to one ray;
- (b) corresponding set of epipolar lines and their epipolar plane.

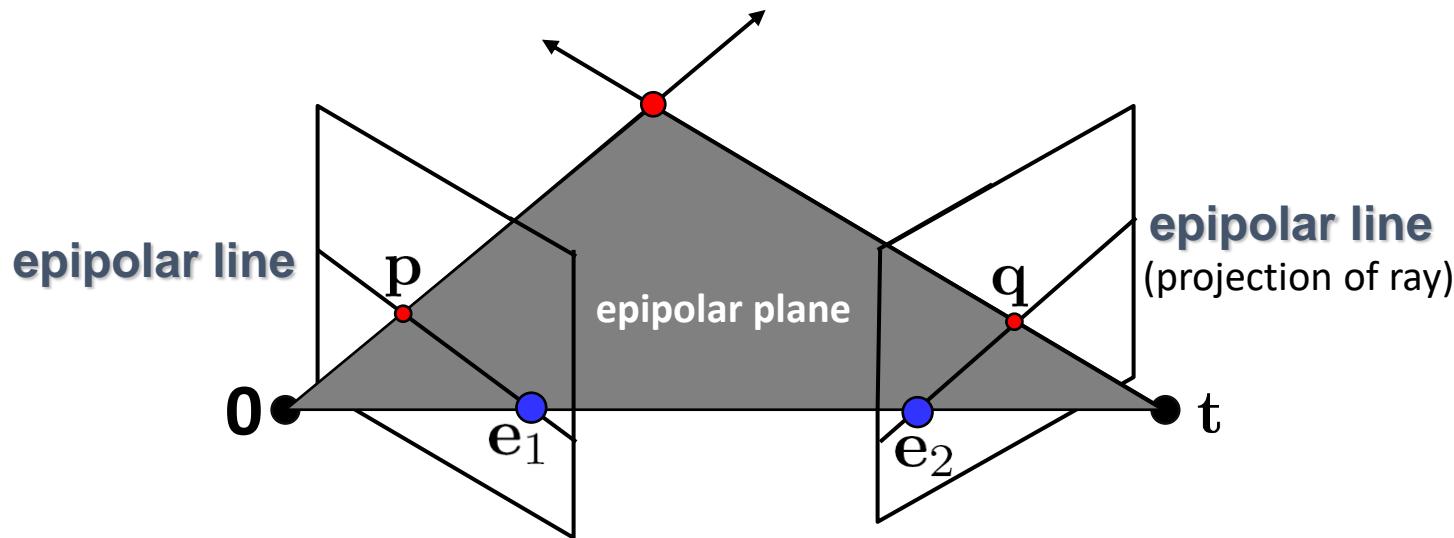
- **Epipolar plane**: passes through both camera centers c_0 and c_1 as well as the point of interest p (Faugeras and Luong 2001; Hartley and Zisserman 2004).
- **Extending both line segments to infinity**: we get a pair of corresponding epipolar lines (Figure 12.3b),
- Epipolar lines are the intersection of the two images with epipolar plane

Two-view geometry

- Where do epipolar lines come from?

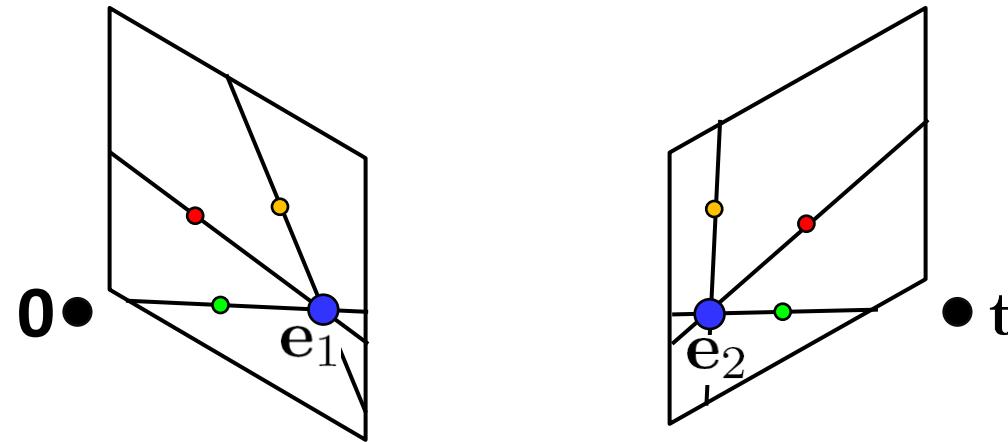


Fundamental matrix



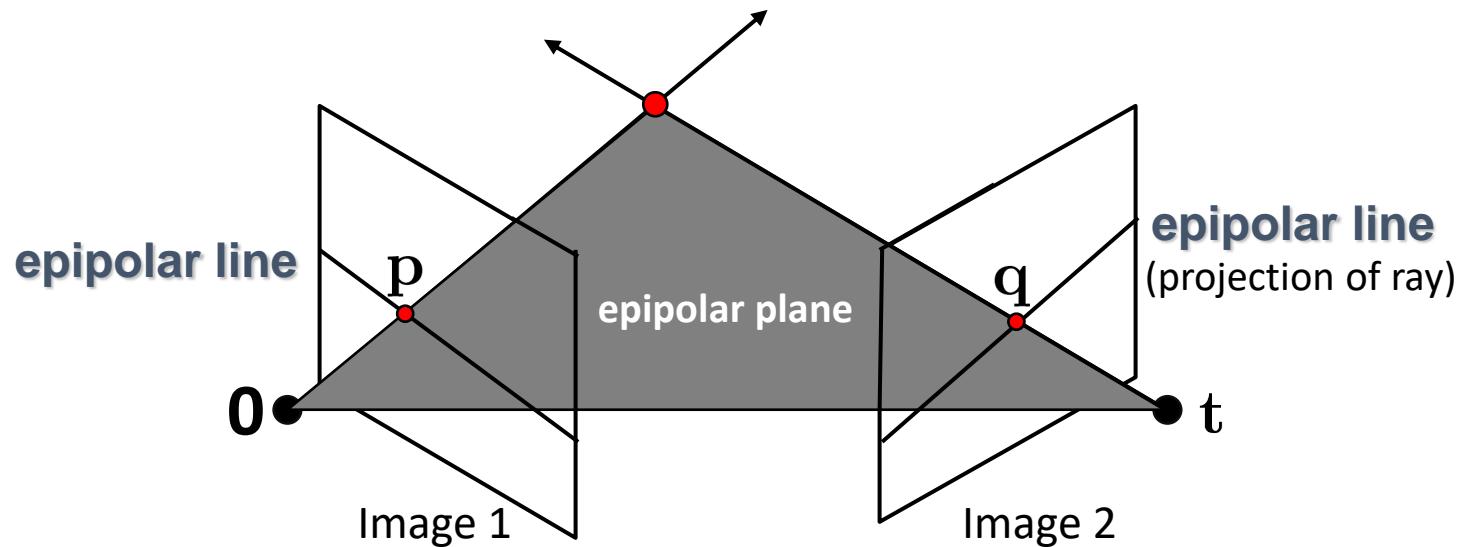
- Two Special points: e_1 and e_2 (the *epipoles*): projection of one camera into the other

Fundamental matrix



- Two Special points: e_1 and e_2 (the *epipoles*): projection of one camera into the other
- All of the epipolar lines in an image pass through the epipole
- Epipoles may or may not be inside the image

Chapter 11.3: Fundamental matrix



- This *epipolar geometry* of two views is described by a Very Special **3×3** matrix \mathbf{F} , called the *fundamental matrix*
- \mathbf{F} maps (homogeneous) *points* in image 1 to *lines* in image 2!
- The epipolar line (in image 2) of point p is: $\mathbf{F}p$
- *Epipolar constraint* on corresponding points: $\mathbf{q}^T \mathbf{F}p = 0$

Fundamental Matrix & Essential Matrix

Image A

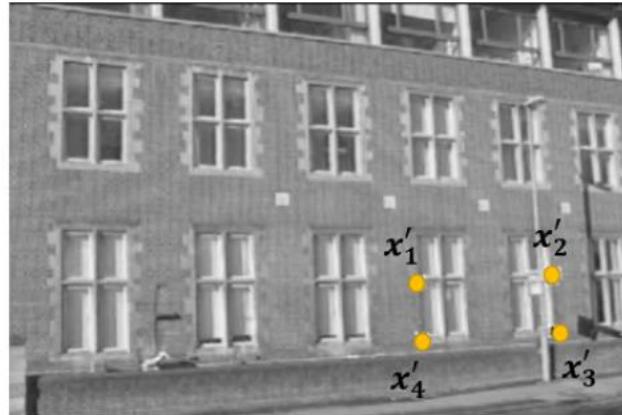
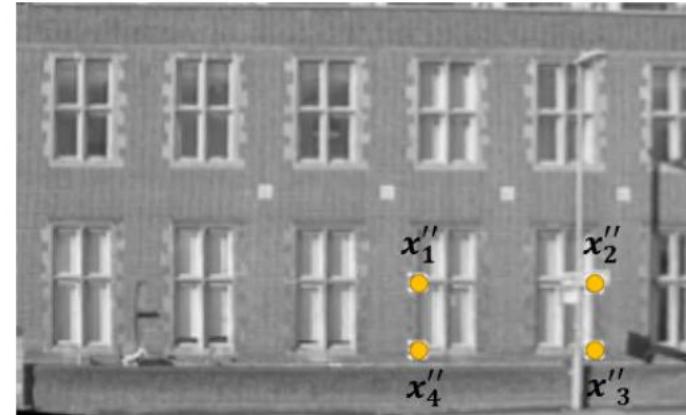


Image B



- The **fundamental matrix** and **the essential matrix** are two matrices frequently used in stereo geometry whenever one wants to describe geometric relationships between pairs of images either recorded from a stereo camera or a monocular camera moving in the environment.
- Suppose we have two images of the same scene and know a set of corresponding points within the two images. For example, we know that point x'_1 , x'_2 , x'_3 , x'_4 in image A correspond to the points x''_1 , x''_2 , x''_3 , x''_4 in the image B:

Fundamental Matrix & Essential Matrix

- The fundamental and the essential matrices are **3x3 homogeneous matrices** with **rank 2**.
- Such rank deficiency is used for formulating the so-called **coplanarity constraint** that can be expressed as:

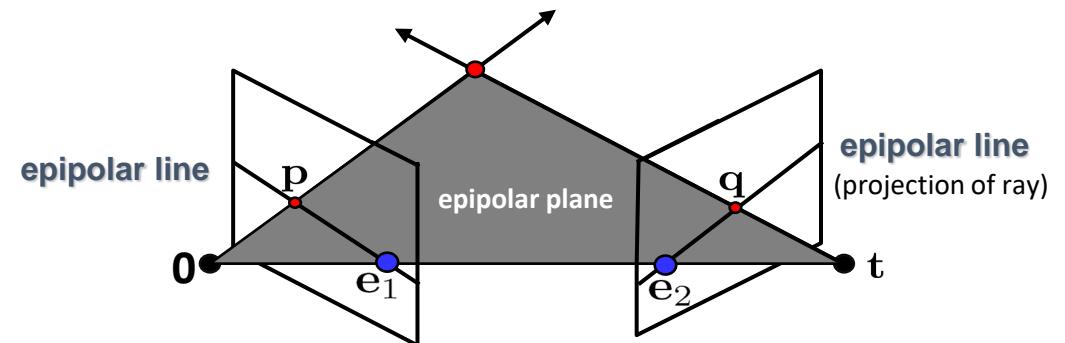
$$\mathbf{x}'^\top \mathbf{F} \mathbf{x}'' = 0$$

$$\mathbf{x}'^\top \mathbf{E} \mathbf{x}'' = 0$$

- where **F** and **E** are the fundamental and the essential matrix,
- **x'** and **x''** are the projections of the same point in the two images.
- The coplanarity constraint equations must hold for all corresponding points.
- F and E are generally unknowns, but they can be estimated given a set of corresponding points by solving a system of linear equations.
- This can be done with the so-called eight-point algorithm, which requires at least 8 corresponding points to estimate the essential or fundamental matrix straightforwardly.
- The method can work with more than 8 points, and additional points can improve the estimate's accuracy.

Properties of the Fundamental Matrix

- $\mathbf{F}\mathbf{p}$ is the epipolar line associated with \mathbf{p}
- $\mathbf{F}^T\mathbf{q}$ is the epipolar line associated with \mathbf{q}
- $\mathbf{F}\mathbf{e}_1 = \mathbf{0}$ and $\mathbf{F}^T\mathbf{e}_2 = \mathbf{0}$
- \mathbf{F} is rank 2
- How many parameters does \mathbf{F} have?



Fundamental matrix song

<http://danielwedge.com/fmatrix/>

Example

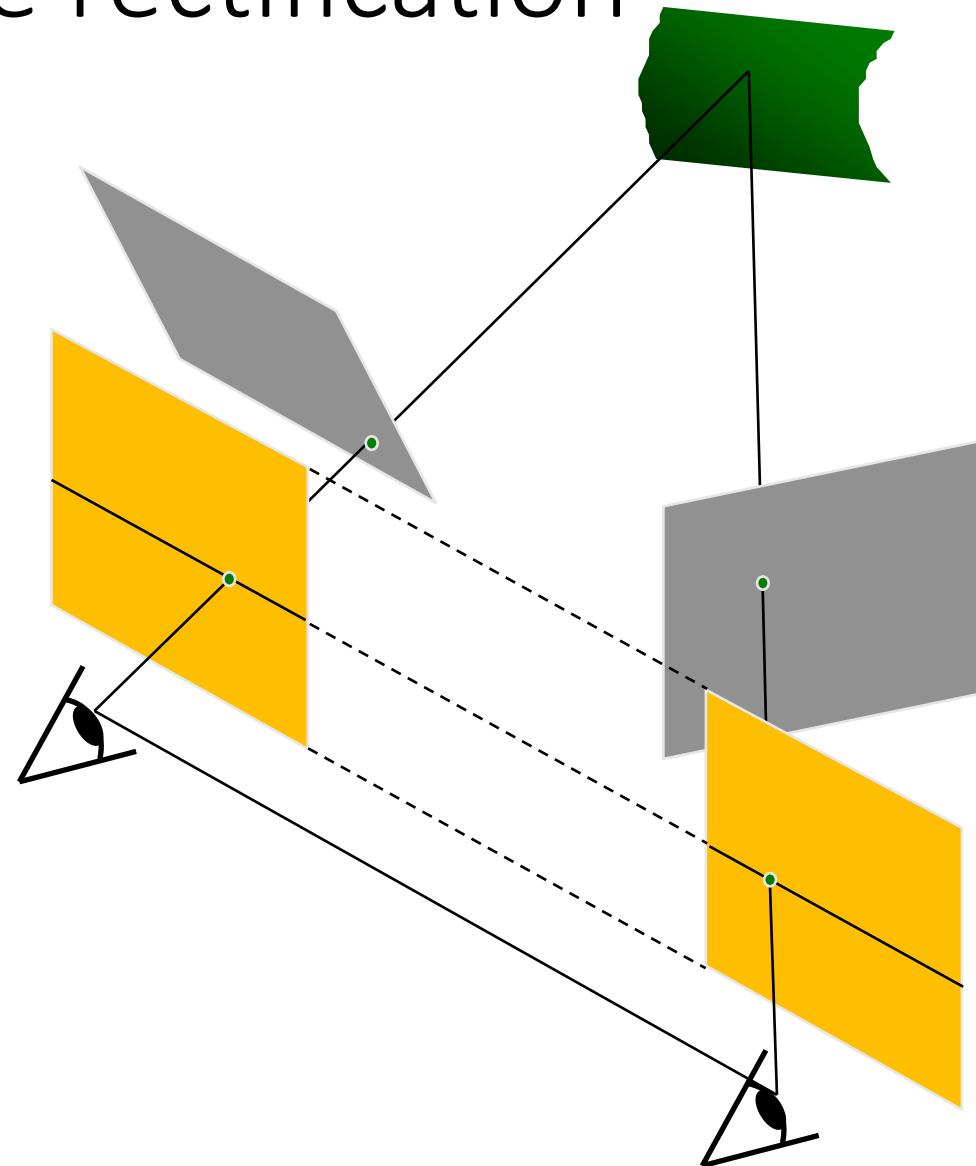


<https://www.cs.cornell.edu/courses/cs5670/2022sp/demos/FundamentalMatrix/?demo=demo1>

Slide credit: Noah Snavely

Chapter 12.1: Stereo image rectification

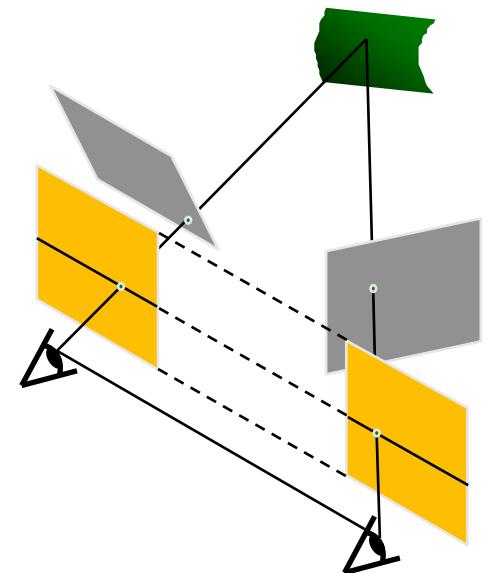
- reproject image planes onto a common plane
 - plane parallel to the line between optical centers
- pixel motion is horizontal after this transformation
- two homographies, one for each input image reprojection
 - C. Loop and Z. Zhang. [Computing Rectifying Homographies for Stereo Vision](#). CVPR 1999.



Stereo image rectification

Approach-1:

1. Section 11.3: compute **fundamental matrix** (or for the calibrated **essential matrix**) (Zhang 1998a,b; Faugeras and Luong 2001; Hartley and Zisserman 2004).
2. Once this geometry has been computed, we can use the epipolar line corresponding to a pixel in one image to constrain the search for corresponding pixels in the other image.
 - One way to do this is to use a general correspondence algorithm, such as optical flow, but to only consider locations along the epipolar.



Approach-2:

A more efficient algorithm:

- First rectifying (i.e., warping) the input images so that corresponding horizontal scanlines are epipolar lines (Loop and Zhang 1999; Faugeras and Luong 2001; Hartley and Zisserman 2004).
- Afterwards, it is possible to match horizontal scanlines independently or to shift images horizontally while computing matching scores (Figure 12.4).

A simple way to rectify the two images:

1. Rotate both cameras so that they are looking perpendicular to the line joining the camera centers c_0 and c_1 . (As there is a degree of freedom in the tilt, the smallest rotations that achieve this should be used.)
2. Determine the desired twist around the optical axes, make the up vector (the camera y-axis) perpendicular to the camera center line. This ensures that corresponding epipolar lines are horizontal and that the disparity for points at infinity is 0.
3. Re-scale the images, if necessary, to account for different focal lengths, magnifying the smaller image to avoid aliasing.

(The full details of this procedure can be found in Fusiello, Trucco, and Verri (2000) and Exercise 12.1.)

Stereo Image Rectification

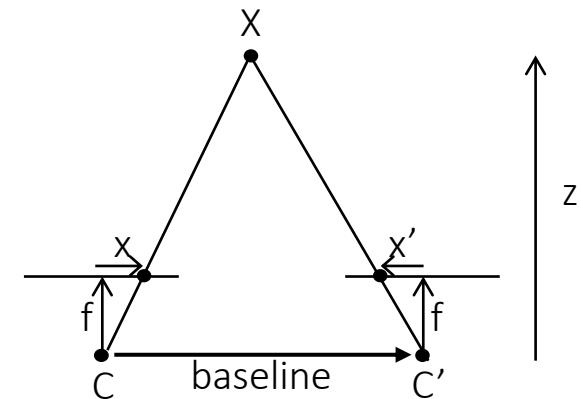
The resulting standard rectified geometry is employed in a lot of stereo camera setups and stereo algorithms, and leads to a very simple inverse relationship

- Z: 3D depths
- d: disparities
- F: the focal length (measured in pixels),
- B: baseline (distance between the cameras)

$$d = f \frac{B}{Z}$$

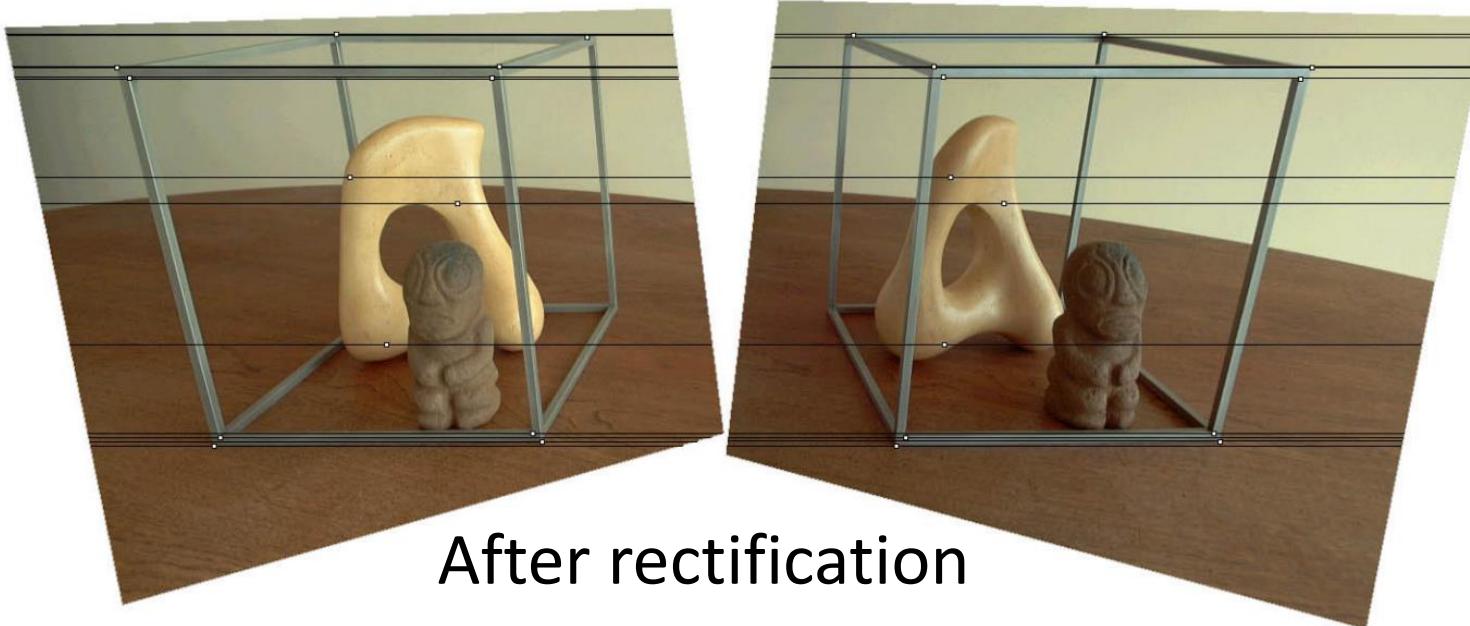
$$x' = x + d(x, y), \quad y' = y$$

- Describes the relationship between corresponding pixel coordinates in the left and right images.
- The task of extracting depth from a set of images then becomes one of estimating the disparity map $d(x; y)$.
- After rectification, we can easily compare the similarity of pixels at corresponding locations $(x; y)$ and $(x'; y') = (x + d; y)$ and store them in a disparity space image (DSI) $C(x; y; d)$ for further processing.



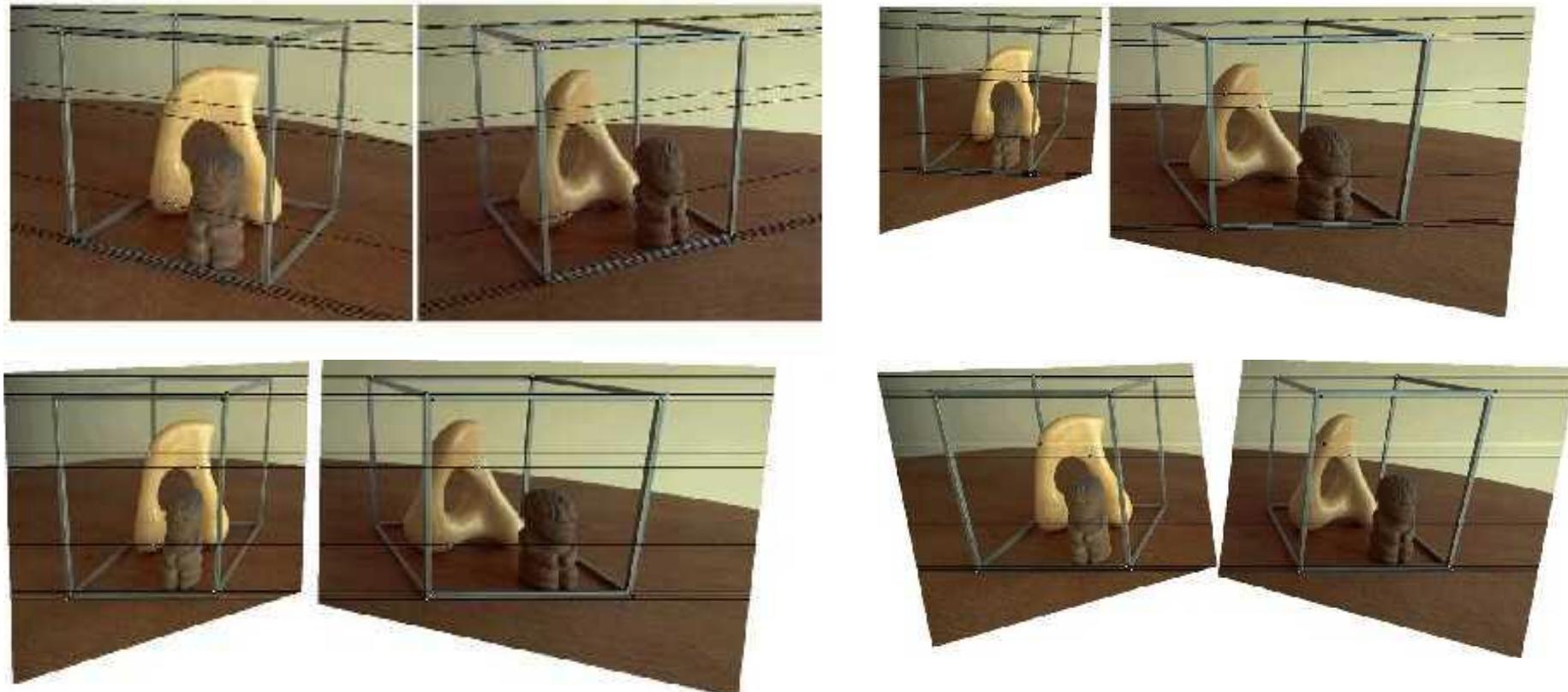


Original stereo pair



After rectification

The multi-stage stereo rectification algorithm of Loop and Zhang (1999) © 1999 IEEE.



The multi-stage stereo rectification algorithm of Loop and Zhang (1999) © 1999 IEEE.

- a) Original image pair overlaid with several epipolar lines;
- b) Images transformed so that epipolar lines are parallel;
- c) Images rectified so that epipolar lines are horizontal and in vertical correspondence;
- d) Final rectification that minimizes horizontal distortions.

Szeliski Figure 12.4

Basic stereo matching algorithm

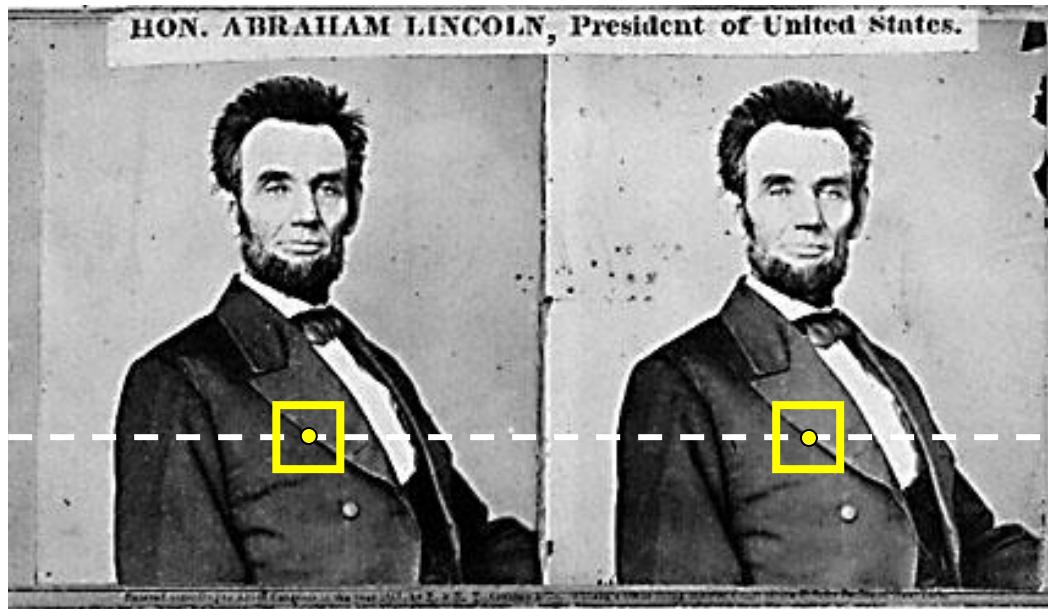
- **Match Pixels in Conjugate Epipolar Lines**
 - Assume brightness constancy
 - This is a challenging problem
 - Hundreds of approaches. A good survey and evaluation:
 - vision.middlebury.edu/stereo
 - <https://paperswithcode.com/task/stereo-matching-1>
 - <https://paperswithcode.com/task/stereo-depth-estimation>

Stereo reconstruction pipeline

Steps

1. Calibrate cameras
2. Rectify images
3. Compute disparity
4. Estimate depth

Your basic stereo matching algorithm



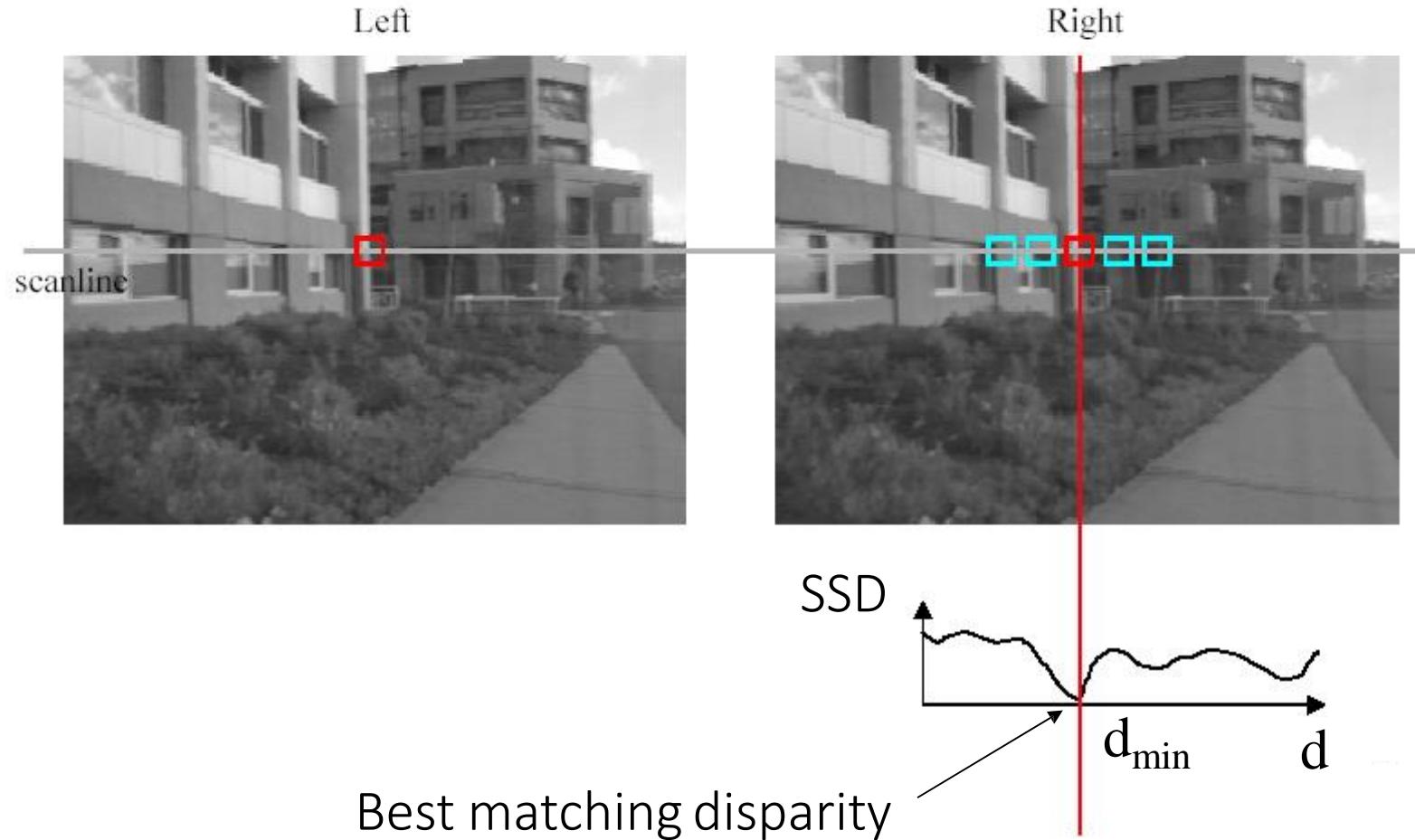
For each epipolar line

 For each pixel in the left image

- compare with every pixel on same epipolar line in right image
- pick pixel with minimum match cost

Improvement: match **windows**

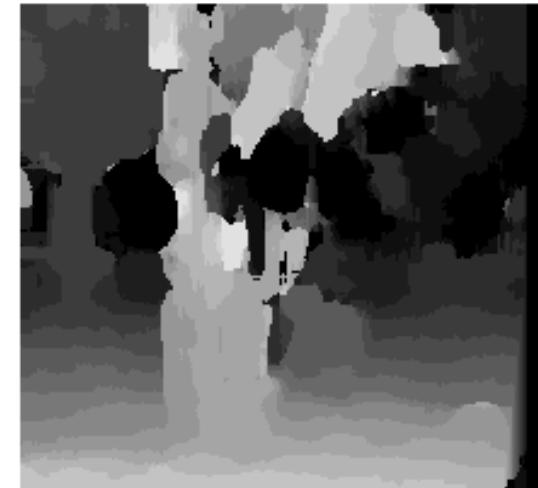
Stereo matching based on SSD



Window size



$W = 3$



$W = 20$

Effect of window size

- Smaller window
 - + more detail
 - more noise
- Larger window
 - + less noise
 - less detail

Better results with *adaptive window*

- T. Kanade and M. Okutomi, [A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment](#), ICRA 1991.
- D. Scharstein and R. Szeliski. [Stereo matching with nonlinear diffusion](#). IJCV, July 1998

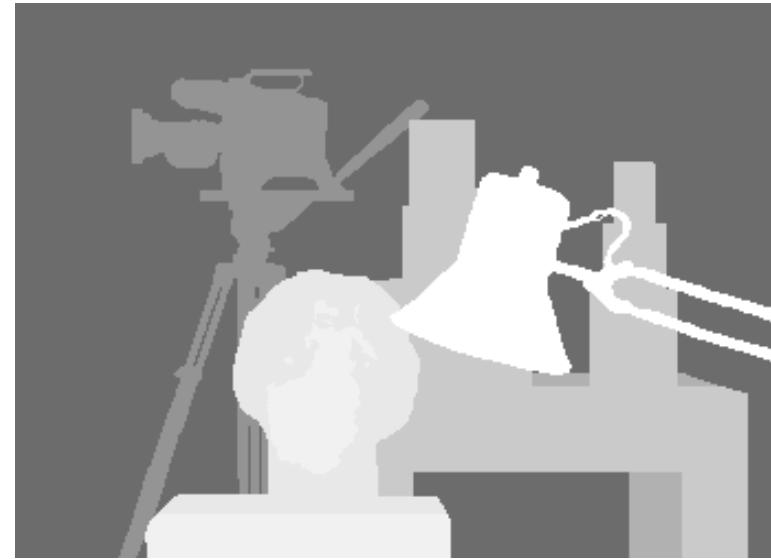
Slide credit: Noah Snavely

Stereo results

- Data from University of Tsukuba
- Similar results on other images without ground truth



Scene



Ground truth

Slide credit: Noah Snavely