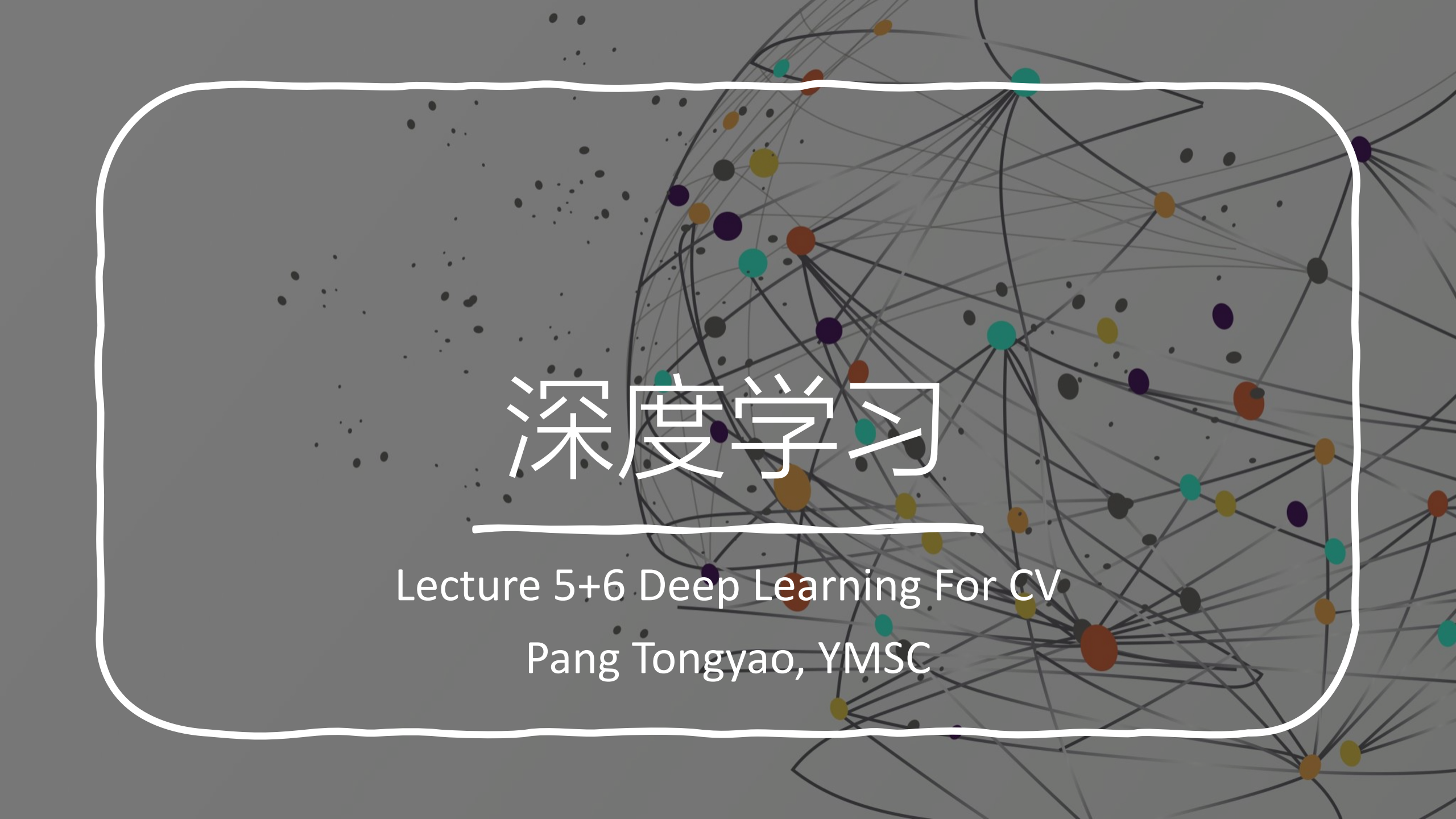


深度学习

Lecture 5+6 Deep Learning For CV

Pang Tongyao, YMSC

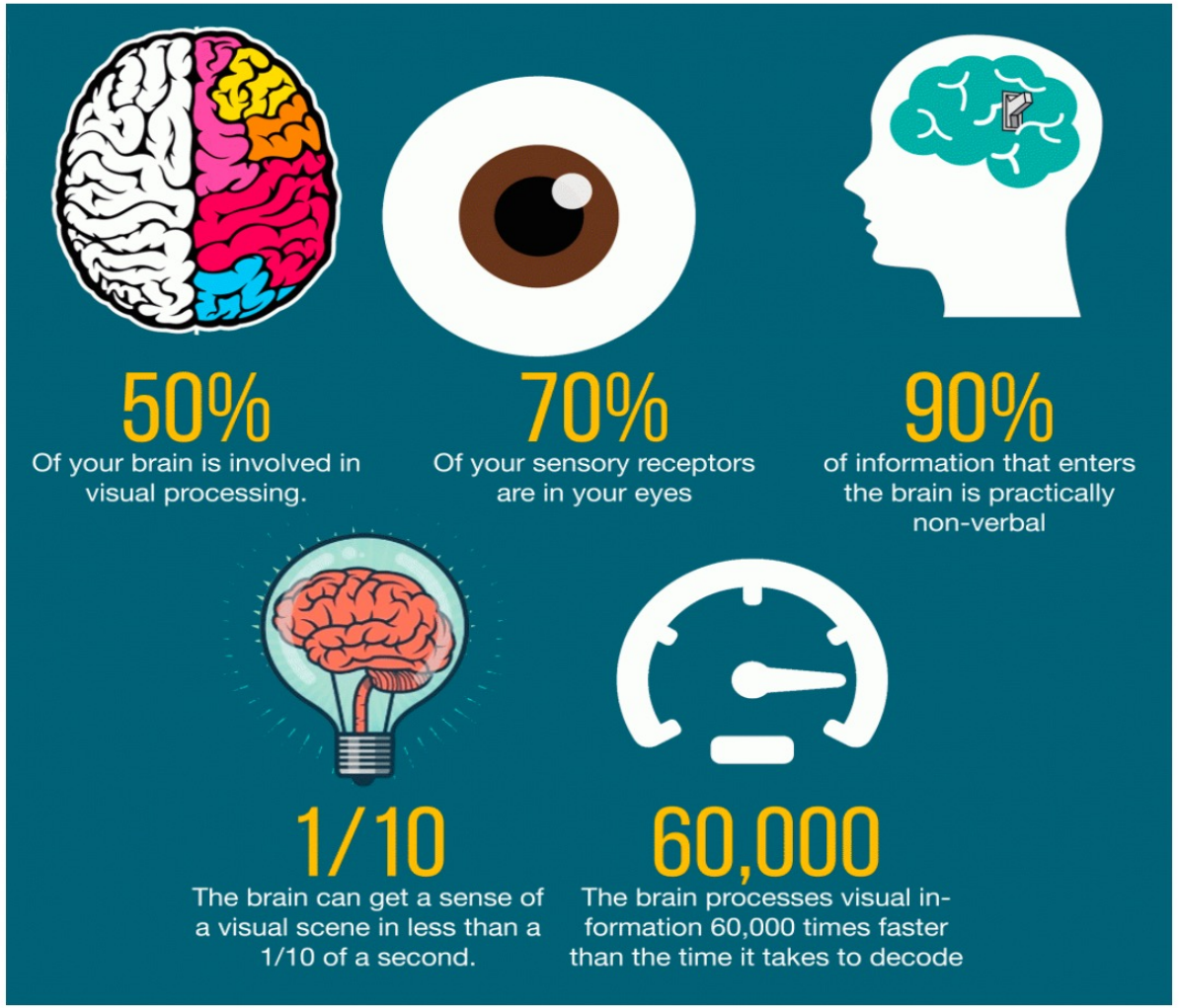
Vision is So Important

Cambrian Explosion

- There is an rapid increase in animal diversity and abundance, as manifest in the fossil record, between ~540 and 520 million years ago.
- Possible Cause: Evolution of the eye
  - Predator-prey relationships changed dramatically after eyesight evolved
  - Where animals lose vision in unlighted environments such as caves, diversity of animal forms tends to decrease.

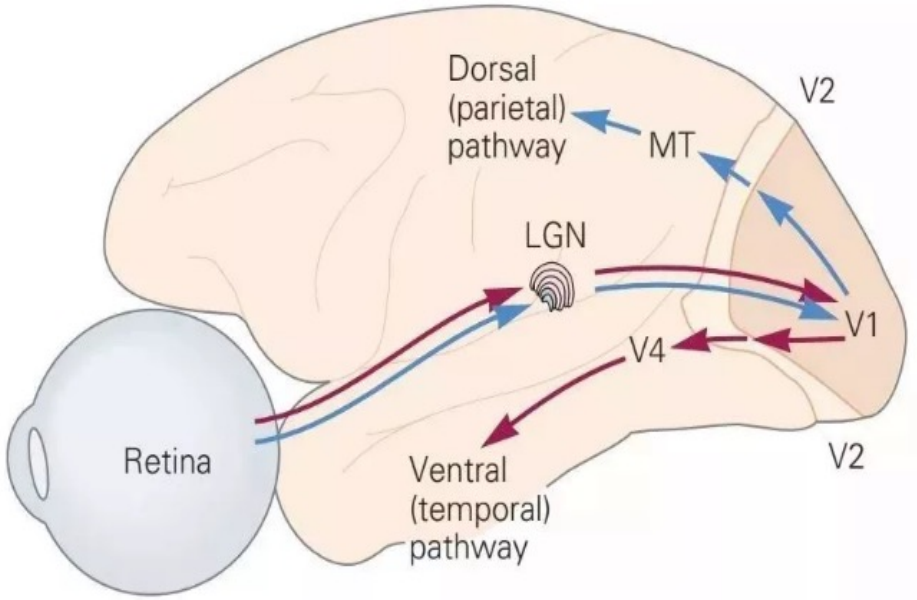
Human Vision System

- Human eye can transmit data to the brain via the optic nerve at about 10^9 bit/s.
- The visual cortex has around O(10^11) neurons.
- A 4-years-old child has seen 50 times more data than the biggest LLMs.



Human Vision System

- **LGN:** Receive information from retina and relay the visual image to V1.
- **V1 (Primary Visual Cortex):** Processes basic visual features (edges, orientation)
- **V2 (Secondary Visual Cortex):** More complex processing (shape, color, contour)
- **V4:** Color processing, shape perception, object recognition
- **V5:** Motion perception



人类视觉系统中的主要成像通路

（图片来自*Principles of Neural Science*第五版）

Human Vision System

- **Efficient coding hypothesis** (By Horace Barlow, 1961): neurons should encode information as efficiently as possible in order to maximize neural resources.
  - Retinal receptors can receive information at 10^9 bit/s
  - The optical Nerve only has a transmission capacity of 10^6 bit/s
  - The overall transmission is further reduced to 10^2 bit/s in high-level interpretation
- V1 Sparse Coding Hypothesis, V1 Saliency Hypothesis …

Human Vision System

- **V1 Saliency Hypothesis**: V1 transforms the visual inputs into a saliency map of the visual field to guide visual attention or direction of gaze.

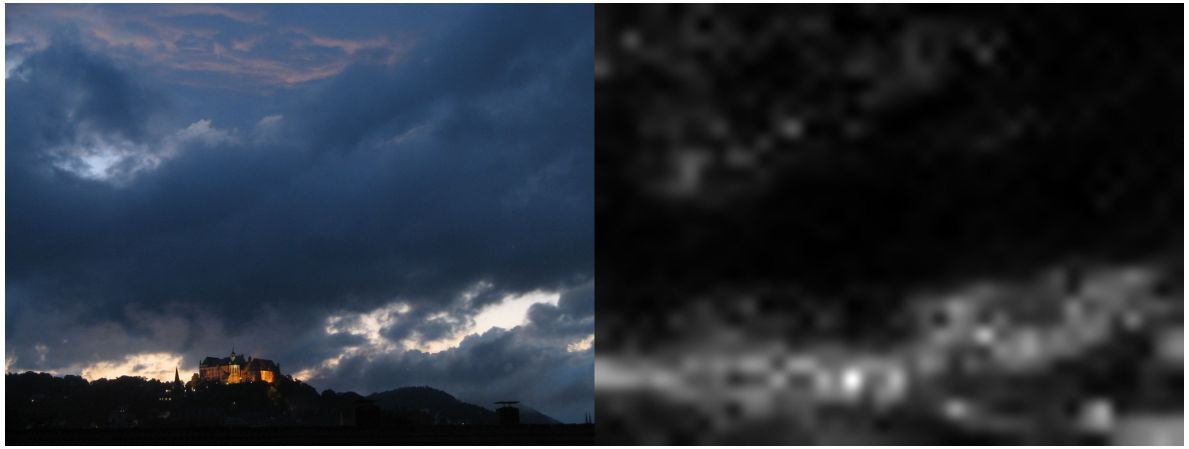
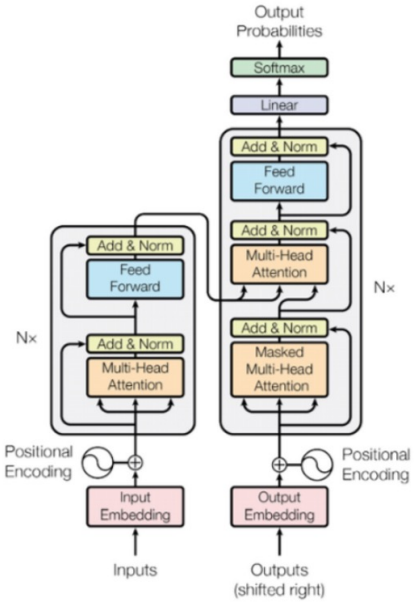


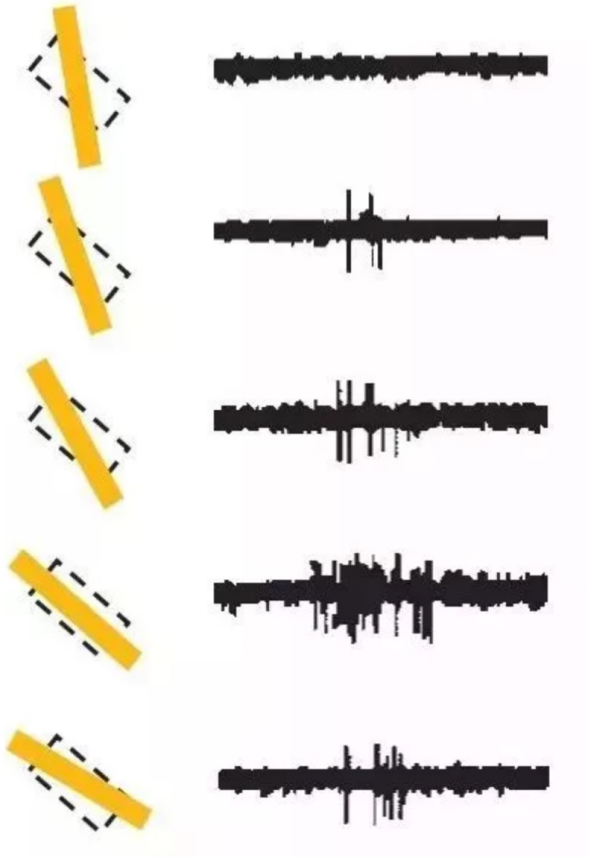
Illustration: Saliency map in biological view

Transformer

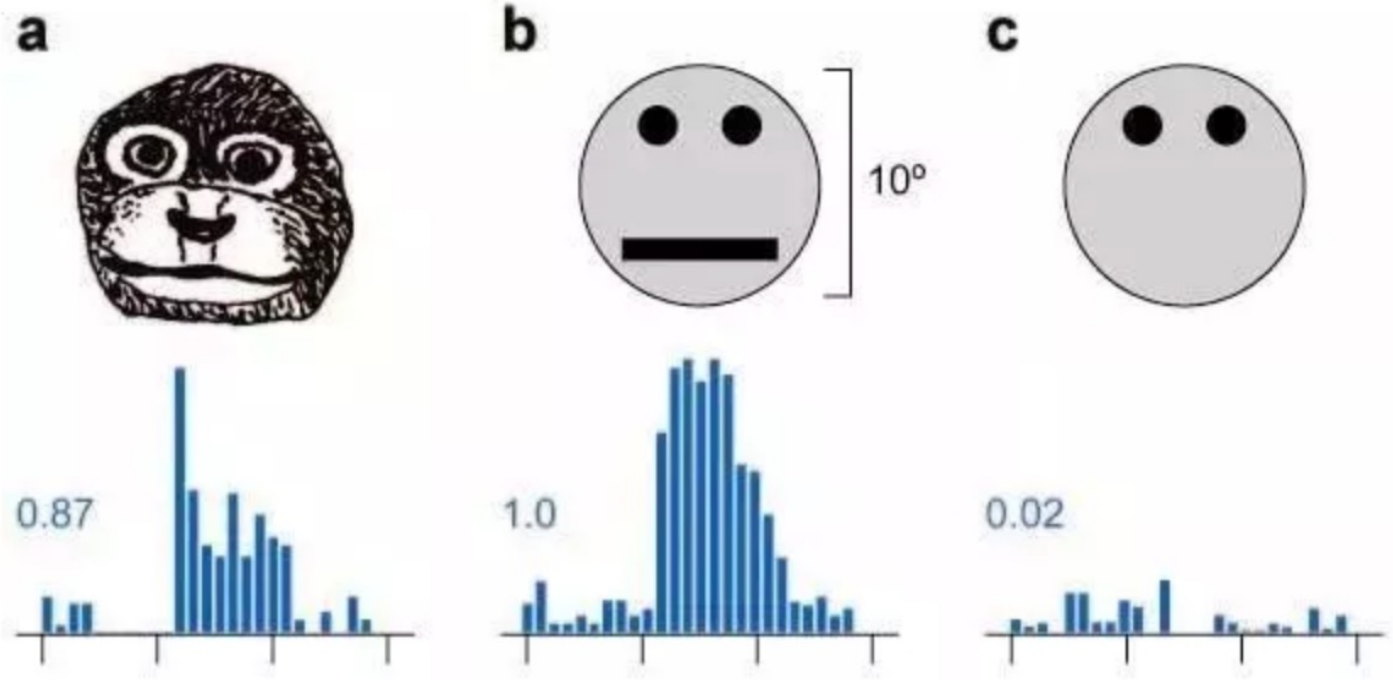
Attention Is All You Need



Animal Vision System



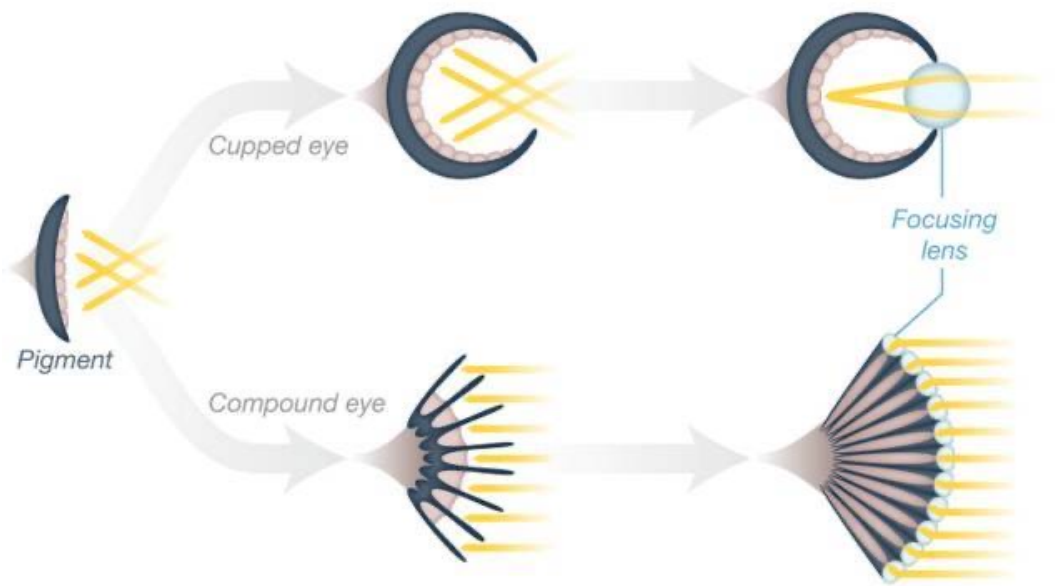
Hubel and Wiesel: Nobel Prize in Physiology or Medicine 1981
Neurons in V1 of cats respond selectively to specific orientations



Face-selective cells in brain of monkeys respond strongly to different faces and are not affected by factors such as facial orientation.

Animal Vision System

- Compound eye: lack of focusing abilities; low space resolution but high temporal resolution; rich color version.



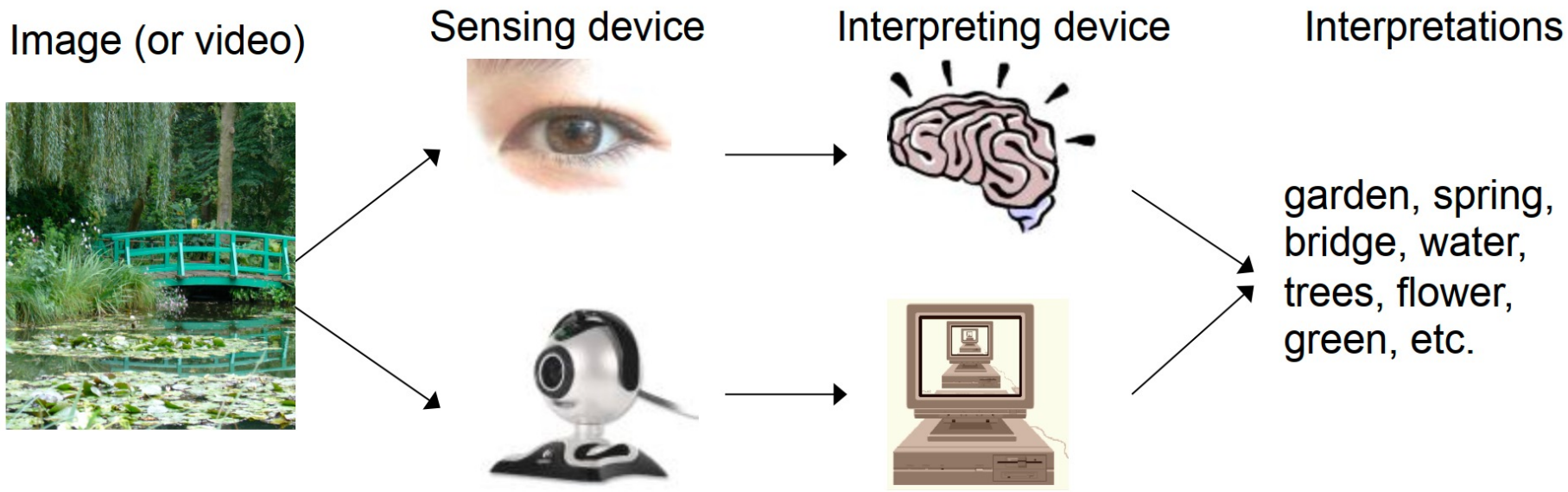
Left: human vision; Right: moth vision

螳螂虾 has 12 types of color-sensing cells, but its neural system is too simple to process them effectively.

What is Computer Vision

Computer Vision

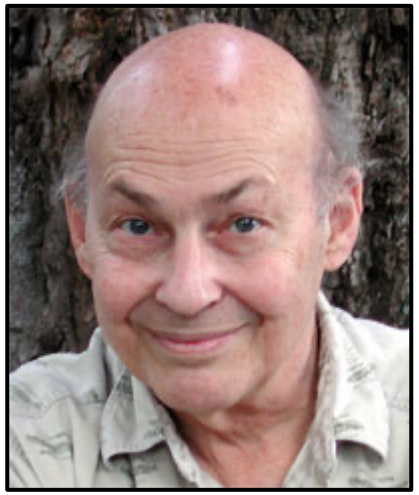
- Computer Version: having a computer interpret an image at the same level of detail and causality as a human being.



It is not easy…

- Origin of computer vision: an MIT undergraduate summer project

In 1966, Marvin Minsky at MIT asked his undergraduate student Gerald Jay Sussman to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" (Boden 2006, p. 781).

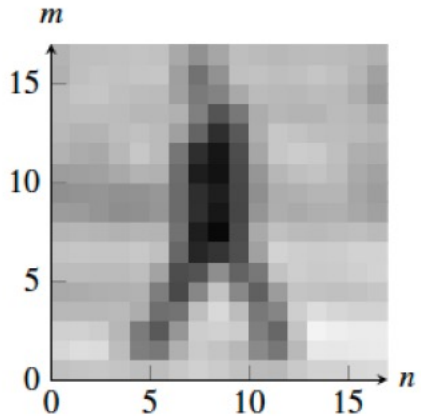


Marvin Minsky, MIT
Turing award,1969

Now, we know the problem is slightly difficult than Minsky thought.

Why it is so hard

What we see

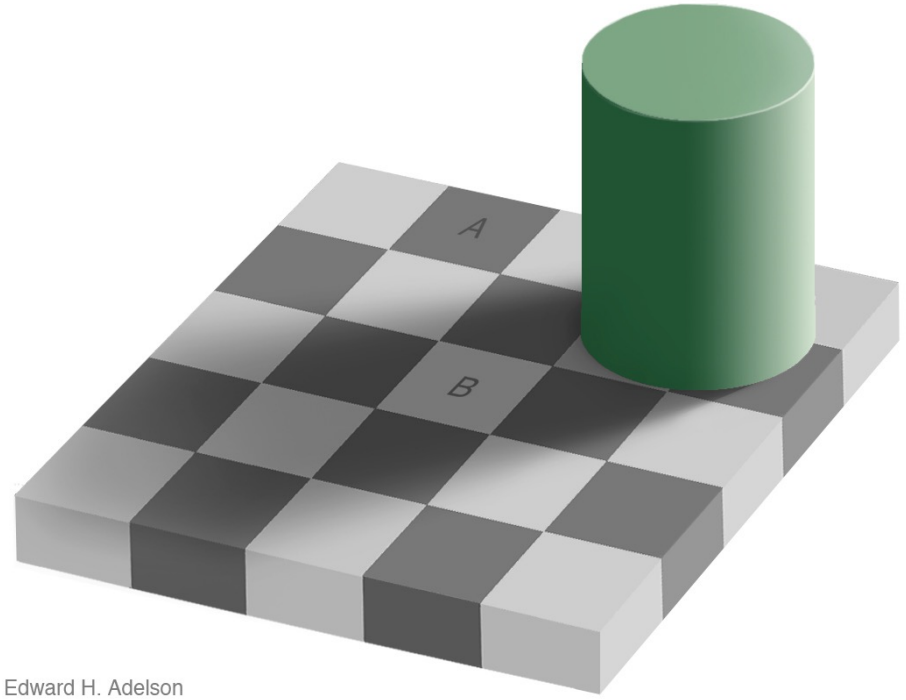


What the machine gets

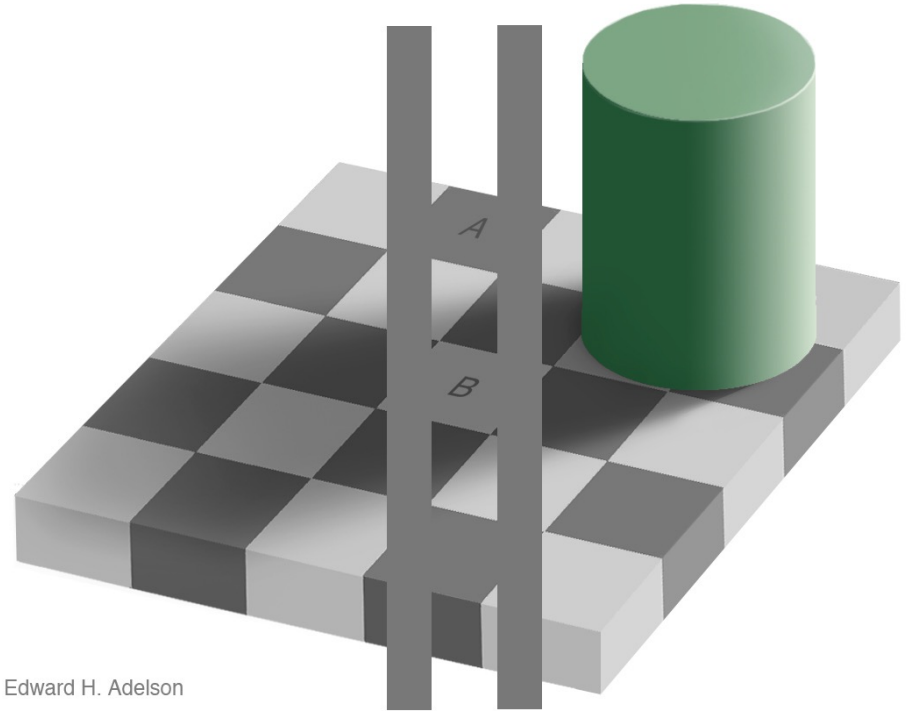
$$\mathbf{I} = \begin{bmatrix}
160 & 175 & 171 & 168 & 168 & 172 & 164 & 158 & 167 & 173 & 167 & 163 & 162 & 164 & 160 & 159 & 163 & 162 \\
149 & 164 & 172 & 175 & 178 & 179 & 176 & 118 & 97 & 168 & 175 & 171 & 169 & 175 & 176 & 177 & 165 & 152 \\
161 & 166 & 182 & 171 & 170 & 177 & 175 & 116 & 109 & 169 & 177 & 173 & 168 & 175 & 175 & 159 & 153 & 123 \\
171 & 174 & 177 & 175 & 167 & 161 & 157 & 138 & 103 & 112 & 157 & 164 & 159 & 160 & 165 & 169 & 148 & 144 \\
163 & 163 & 162 & 165 & 167 & 164 & 178 & 167 & 77 & 55 & 134 & 170 & 167 & 162 & 164 & 175 & 168 & 160 \\
173 & 164 & 158 & 165 & 180 & 180 & 150 & 89 & 61 & 34 & 137 & 186 & 186 & 182 & 175 & 165 & 160 & 164 \\
152 & 155 & 146 & 147 & 169 & 180 & 163 & 51 & 24 & 32 & 119 & 163 & 175 & 182 & 181 & 162 & 148 & 153 \\
134 & 135 & 147 & 149 & 150 & 147 & 148 & 62 & 36 & 46 & 114 & 157 & 163 & 167 & 169 & 163 & 146 & 147 \\
135 & 132 & 131 & 125 & 115 & 129 & 132 & 74 & 54 & 41 & 104 & 156 & 152 & 156 & 164 & 156 & 141 & 144 \\
151 & 155 & 151 & 145 & 144 & 149 & 143 & 71 & 31 & 29 & 129 & 164 & 157 & 155 & 159 & 158 & 156 & 148 \\
172 & 174 & 178 & 177 & 177 & 181 & 174 & 54 & 21 & 29 & 136 & 190 & 180 & 179 & 176 & 184 & 187 & 182 \\
177 & 178 & 176 & 173 & 174 & 180 & 150 & 27 & 101 & 94 & 74 & 189 & 188 & 186 & 183 & 186 & 188 & 187 \\
160 & 160 & 163 & 163 & 161 & 167 & 100 & 45 & 169 & 166 & 59 & 136 & 184 & 176 & 175 & 177 & 185 & 186 \\
147 & 150 & 153 & 155 & 160 & 155 & 56 & 111 & 182 & 180 & 104 & 84 & 168 & 172 & 171 & 164 & 168 & 167 \\
184 & 182 & 178 & 175 & 179 & 133 & 86 & 191 & 201 & 204 & 191 & 79 & 172 & 220 & 217 & 205 & 209 & 200 \\
184 & 187 & 192 & 182 & 124 & 32 & 109 & 168 & 171 & 167 & 163 & 51 & 105 & 203 & 209 & 203 & 210 & 205 \\
191 & 198 & 203 & 197 & 175 & 149 & 169 & 189 & 190 & 173 & 160 & 145 & 156 & 202 & 199 & 201 & 205 & 202 \\
153 & 149 & 153 & 155 & 173 & 182 & 179 & 177 & 182 & 177 & 182 & 185 & 179 & 177 & 167 & 176 & 182 & 180
\end{bmatrix}$$

The camera is a measurement device, not a vision system.

Perception vs. Measurement



Edward H. Adelson

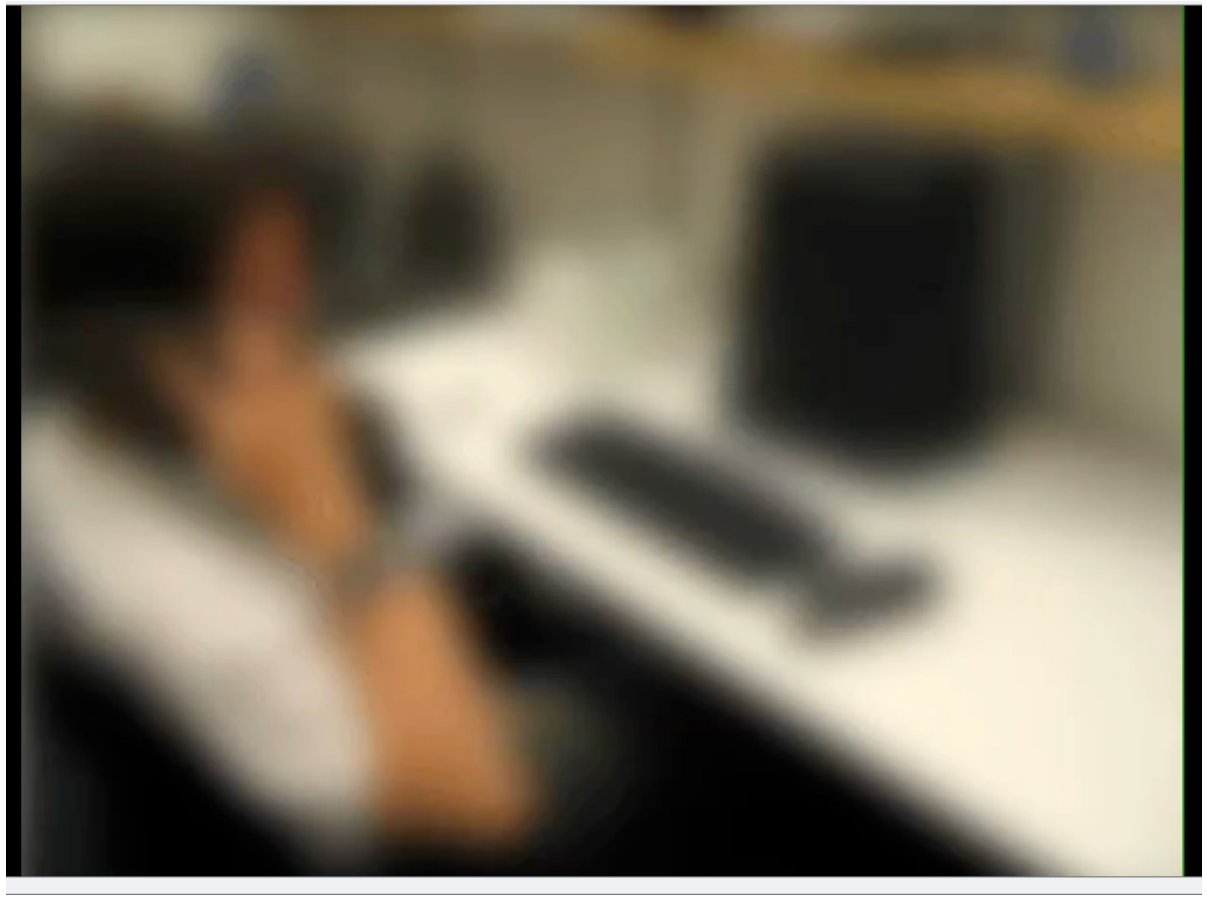


Edward H. Adelson

Are the color of A and B the same?

YES!

Measurements are inaccurate



Measurements are inaccurate

Measurements are insufficient

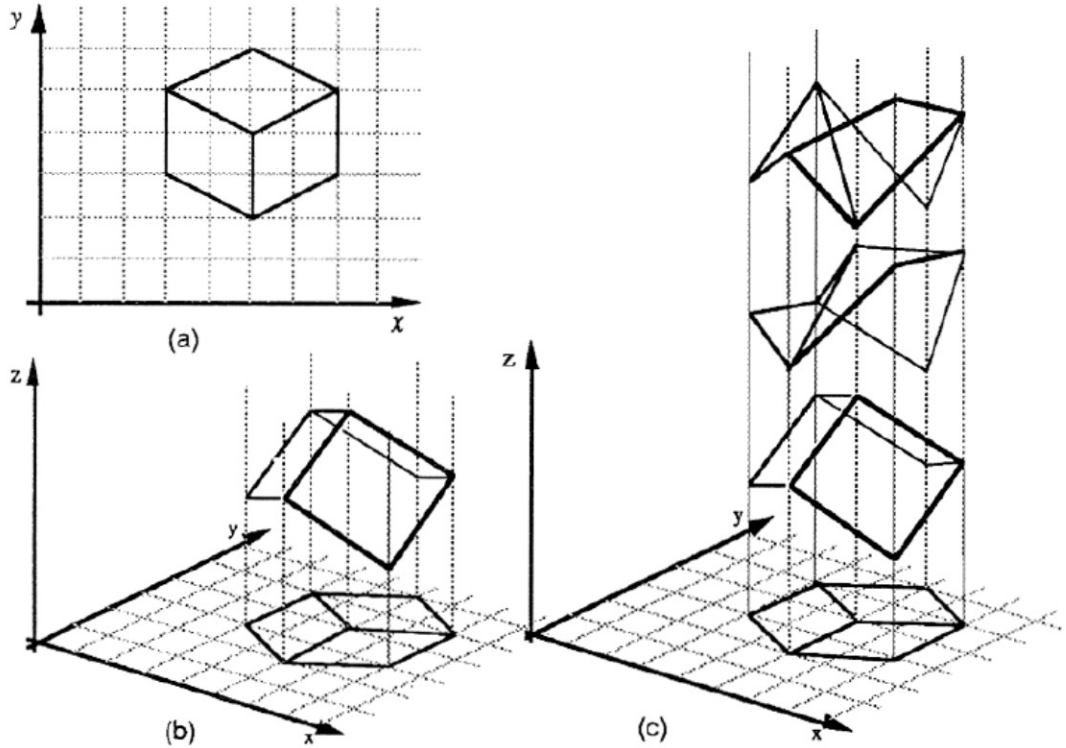
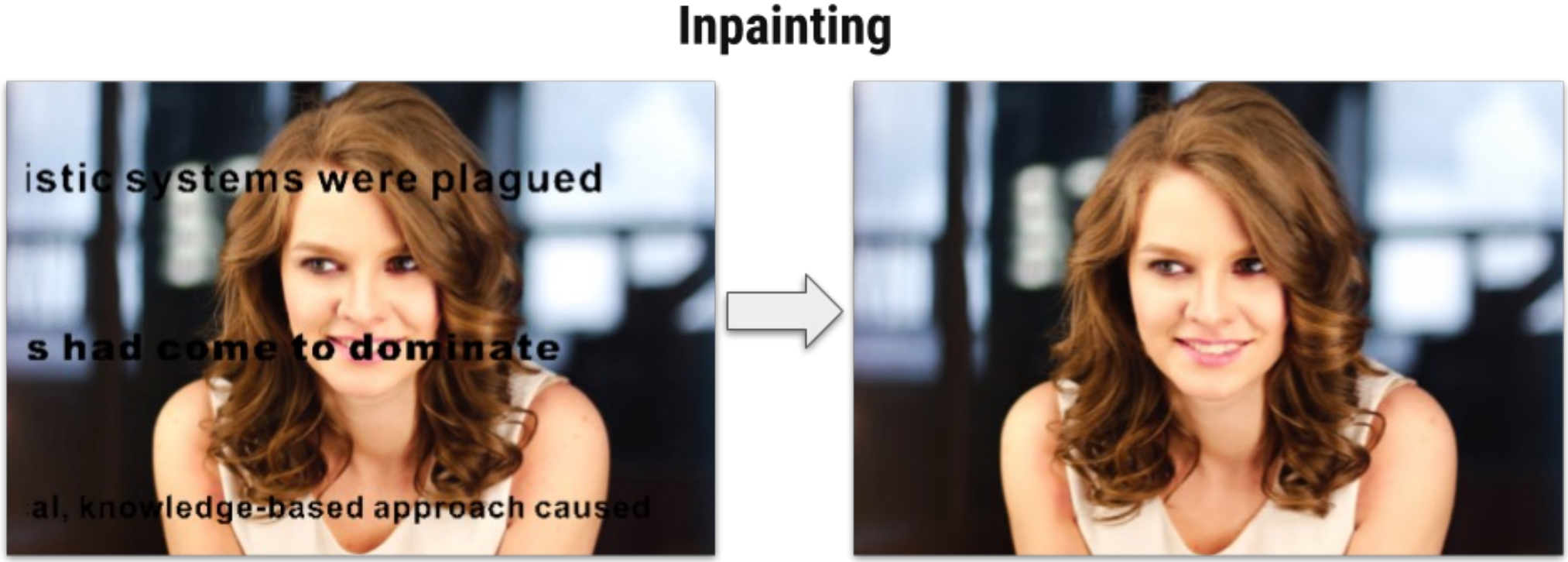


Figure 1. (a) A line drawing provides information only about the x, y coordinates of points lying along the object contours. (b) The human visual system is usually able to reconstruct an object in three dimensions given only a single 2D projection (c) Any planar line-drawing is geometrically consistent with infinitely many 3D structures.

Sinha & Adelson 93

Measurements are insufficient

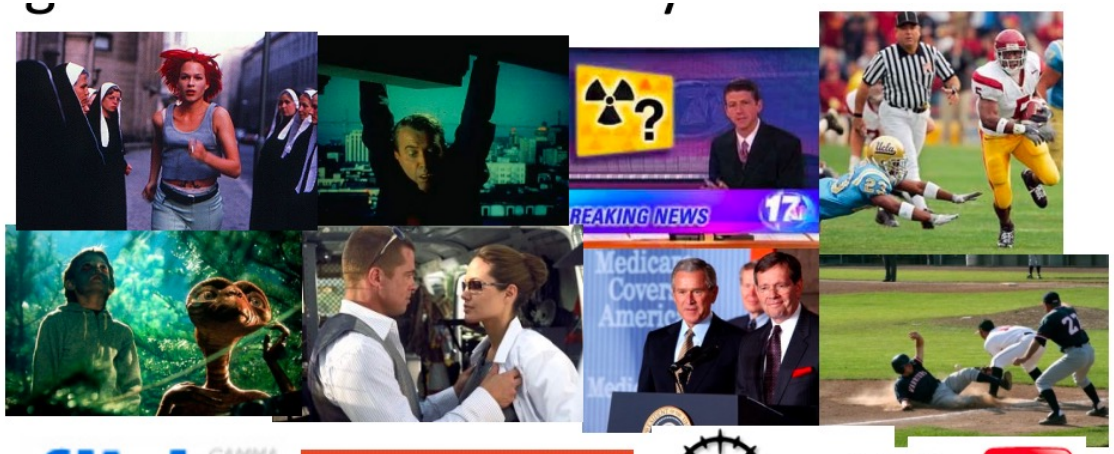


Measurements are varying

Measurements are extrinsic

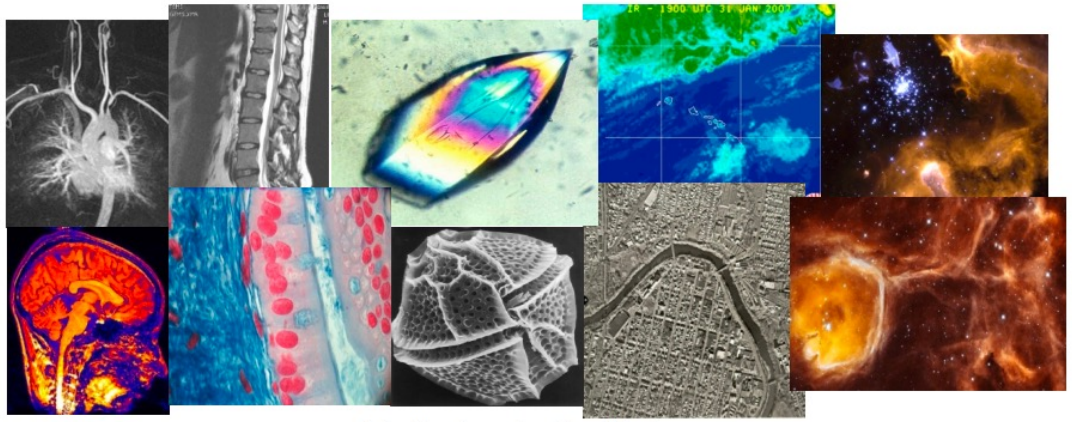
Vision is critically important in artificial intelligence.

Visual data is everywhere



Google Image Search™ Picasa™ flickr™ webshots picsearch™ YouTube Broadcast Yourself™

Surveillance and security

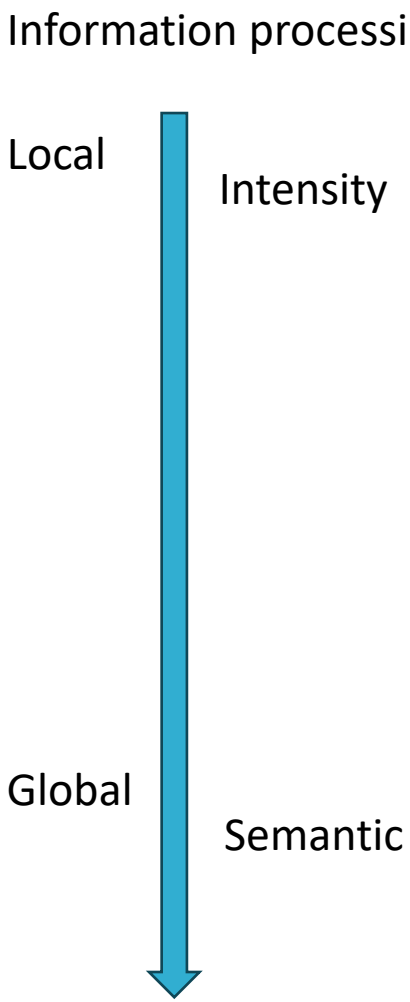


Medical and scientific images

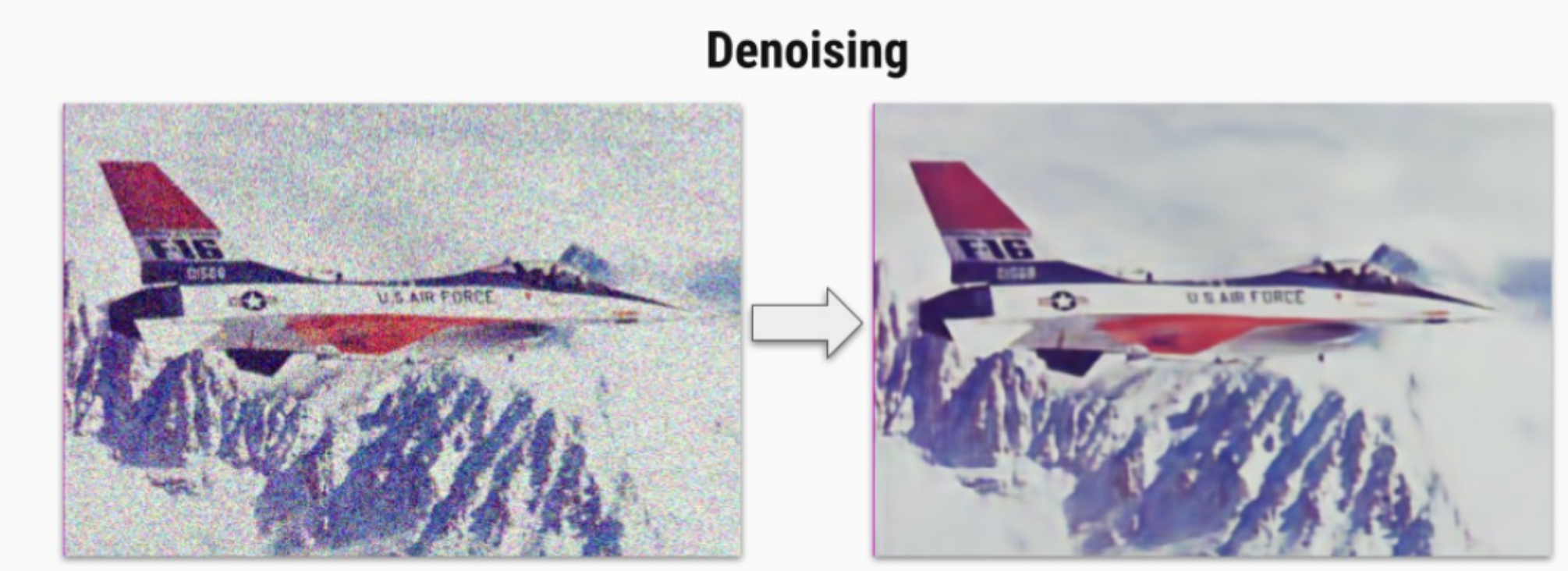
Slide from Feifei Li

Computer Vision Tasks

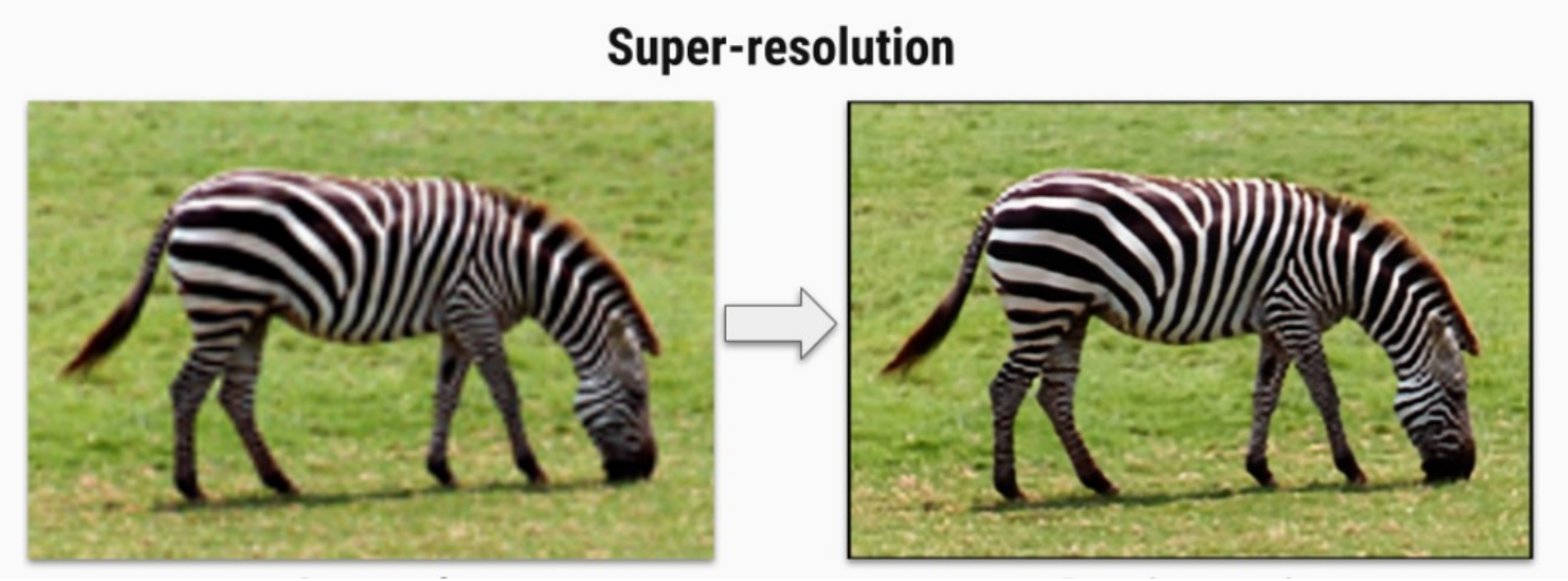
- Low Level Vision
  - Image denoising/deblurring
  - Super resolution
  - Enhancement
- Mid Level Vision
  - Reconstruction
  - Depth Estimation
  - Motion Estimation
- High Level Vision
  - Classification
  - Recognition
  - Deep understandings



Low Level Vision



Low Level Vision

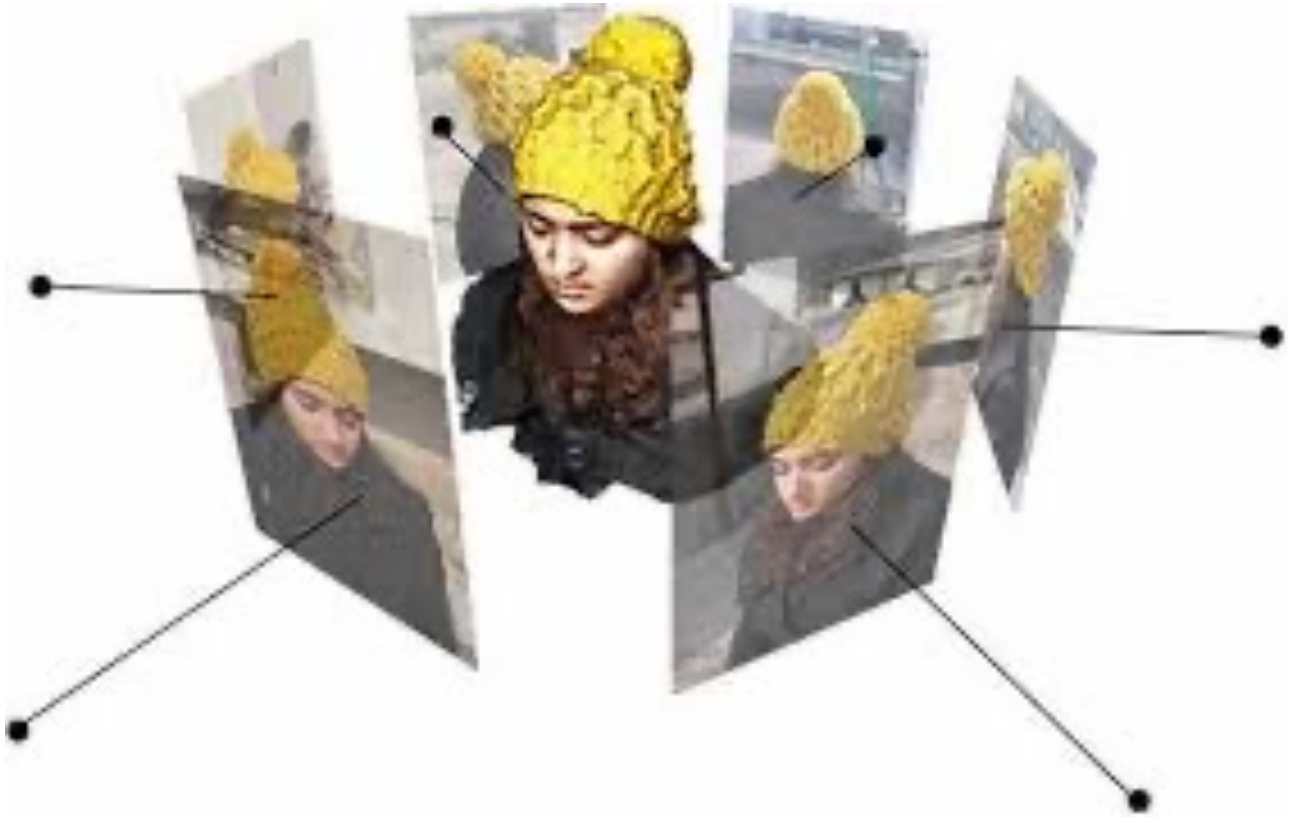


Low Level Vision

Dehazing

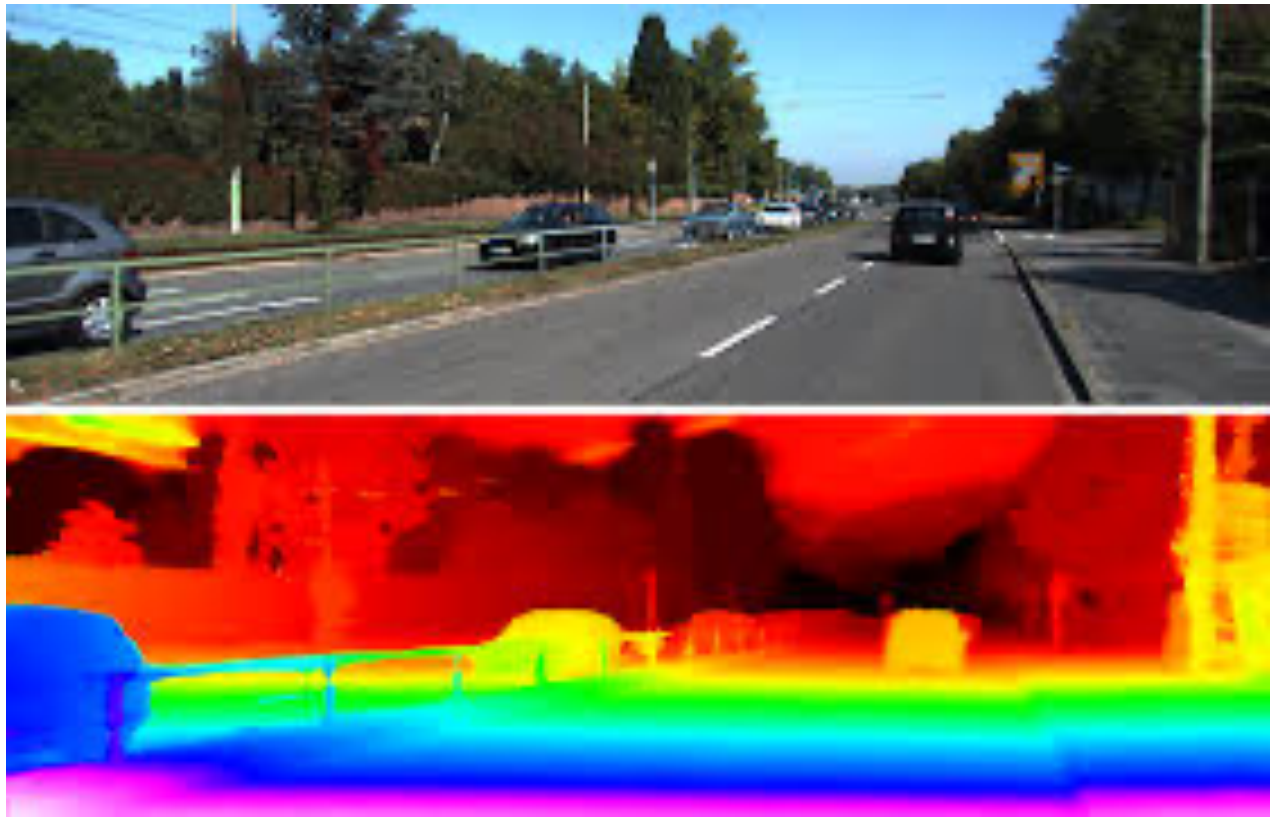
Mid Level Vision

3D Reconstruction



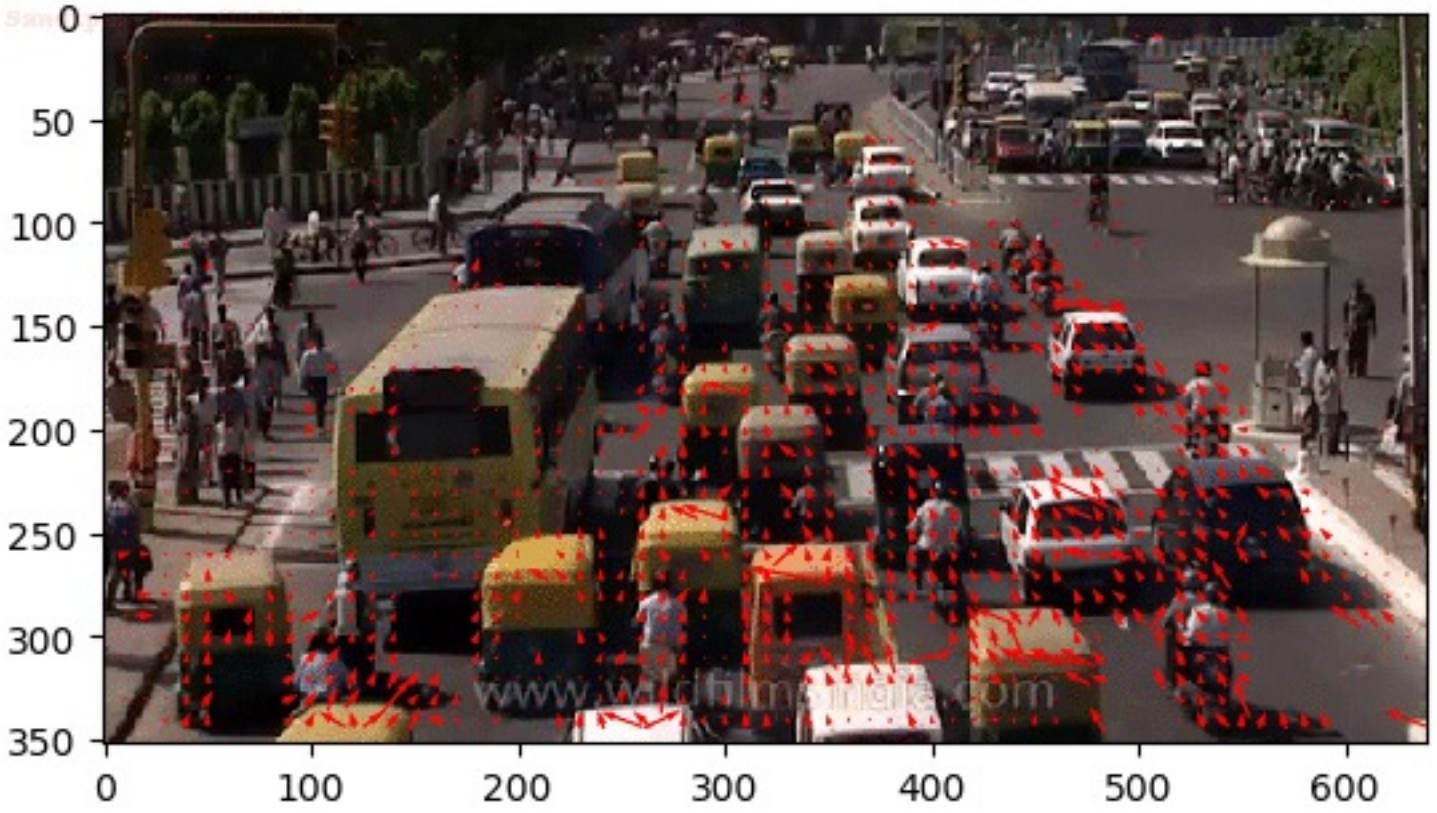
Mid Level Vision

Depth Estimation



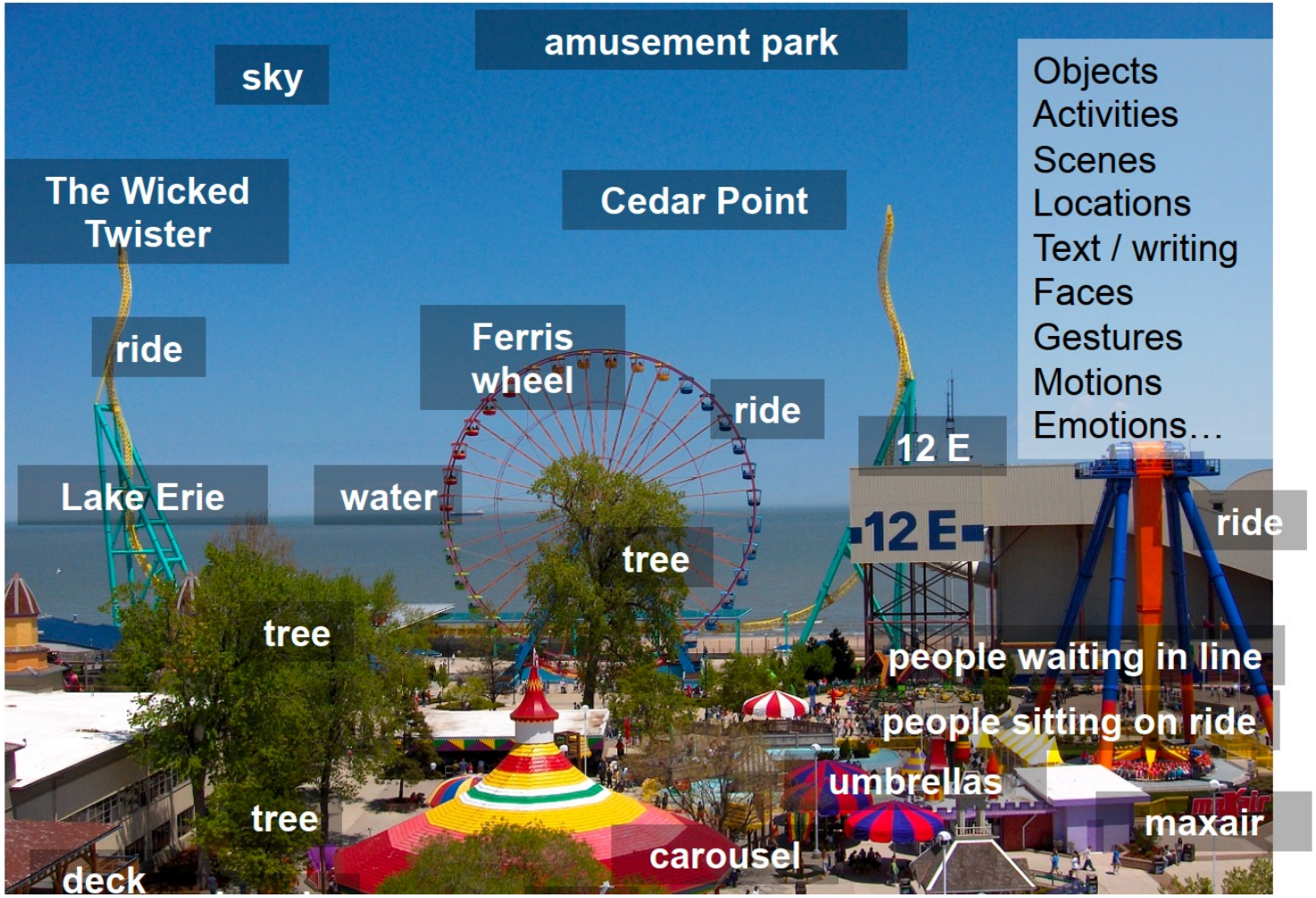
Mid Level Vision

Motion Estimation



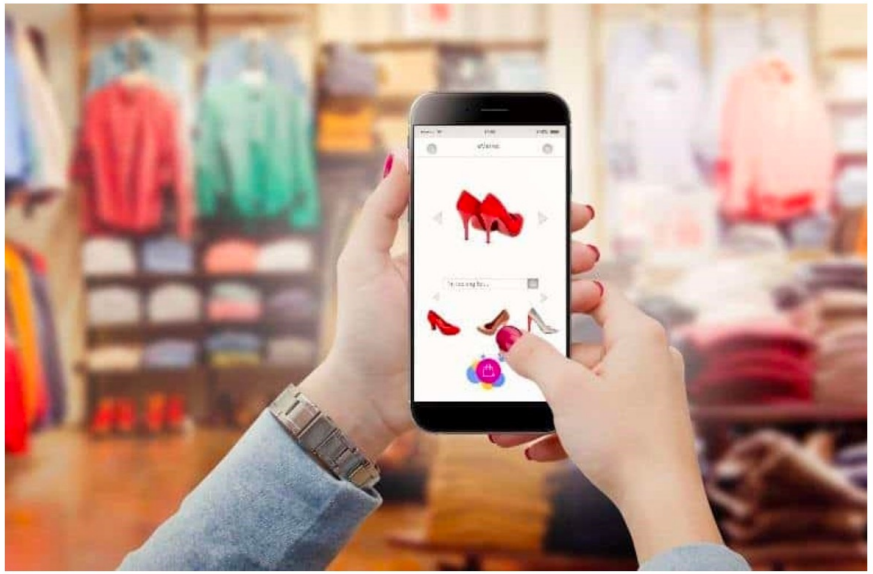
High Level Vision

Object detection



More Applications

Visual Search



Virtual Reality

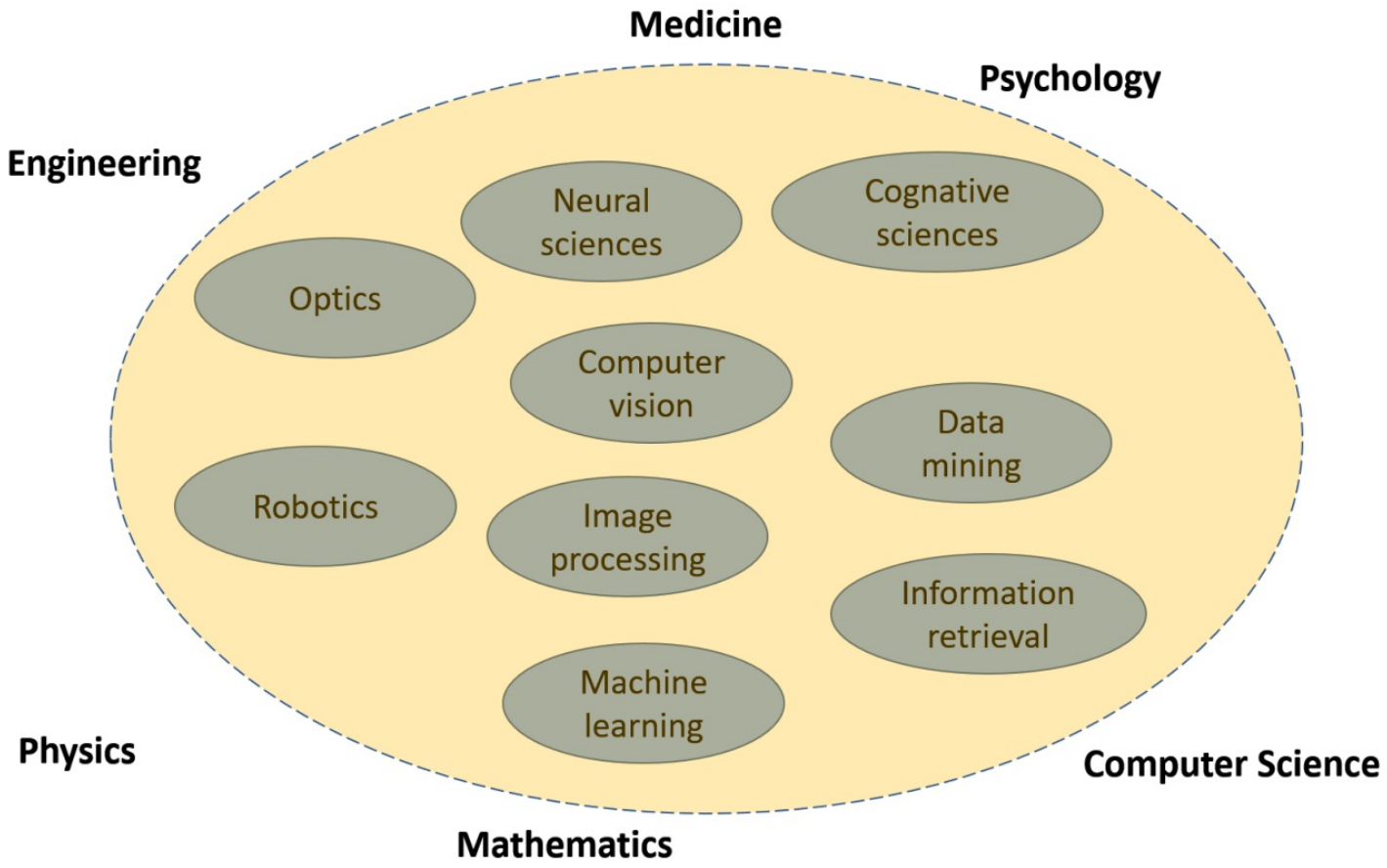
More Applications

Self-driving

Embodied Intelligence

Algorithm Developments

Related Disciplines



Filtering

Median filtering for image denoising

Filtering

Canny edge detection: high pass filter

Modeling and Optimization

- Total Variation (TV) model in low level tasks

$$\min \frac{1}{2}||Ax - y||_2^2 + \lambda TV(x)$$

- Optimization methods: ADMM, Proximal gradient,…

Modeling and Optimization

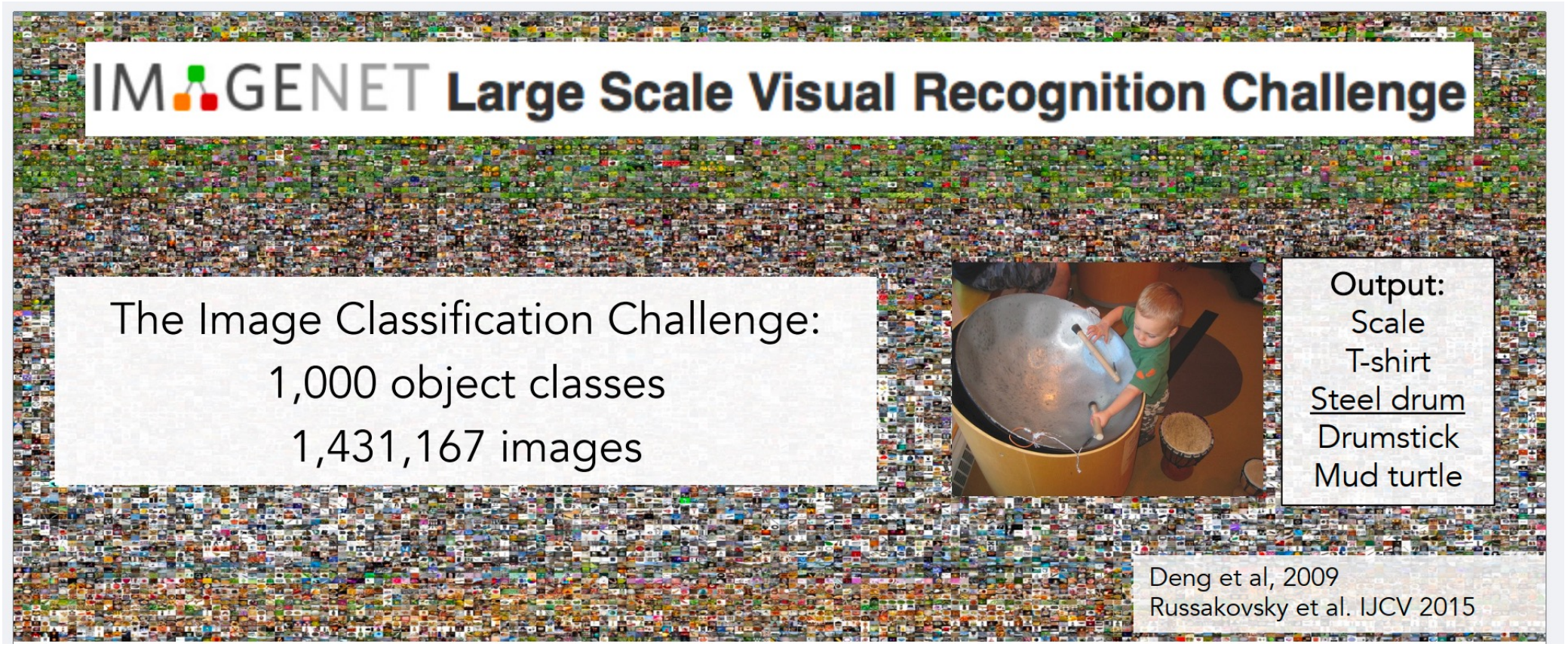
- Segmentation based on energy minimization

$$E^*_{\text{snake}} = \int_0^1 E_{\text{snake}}(\mathbf{v}(s))\,ds = \int_0^1 (E_{\text{internal}}(\mathbf{v}(s)) + E_{\text{image}}(\mathbf{v}(s)) + E_{\text{con}}(\mathbf{v}(s)))\,ds$$

snake energy model

- Optimization: gradient descent

Deep Learning



Deep Learning

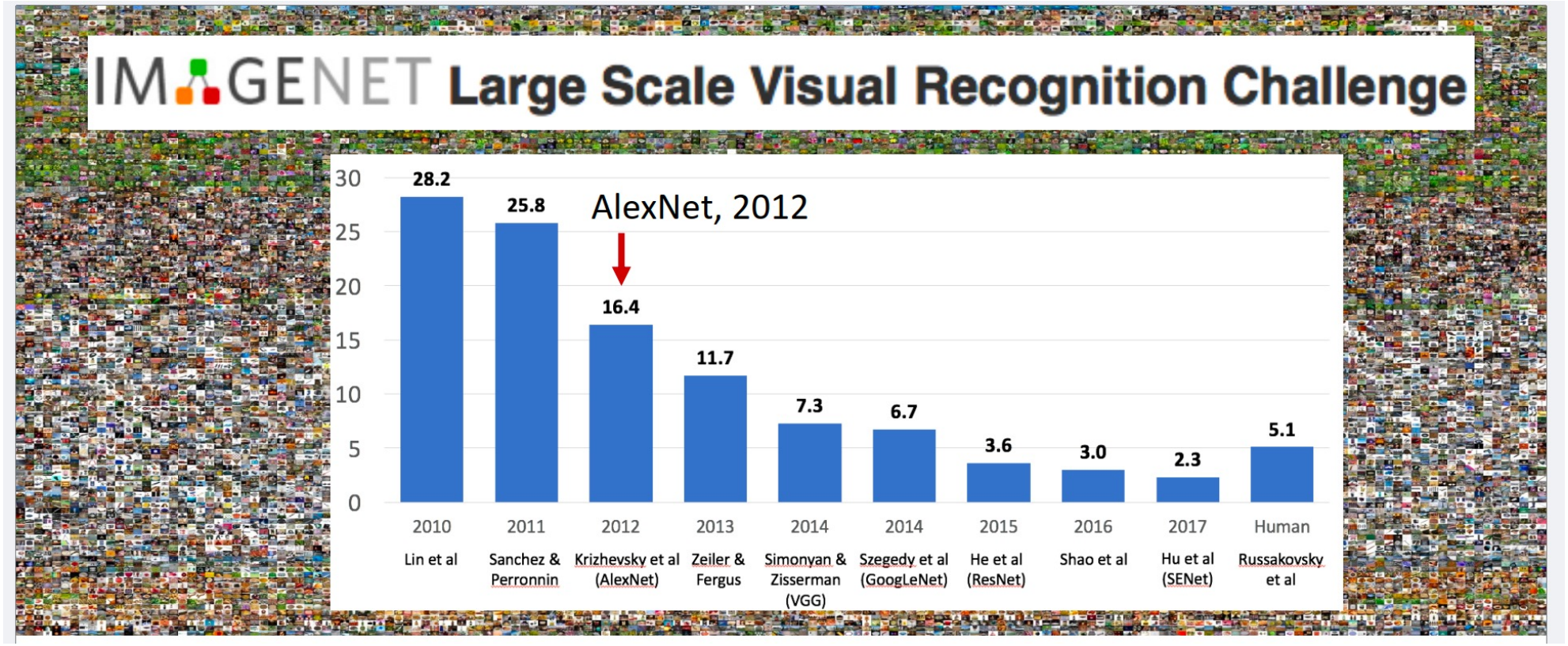


Image Formation

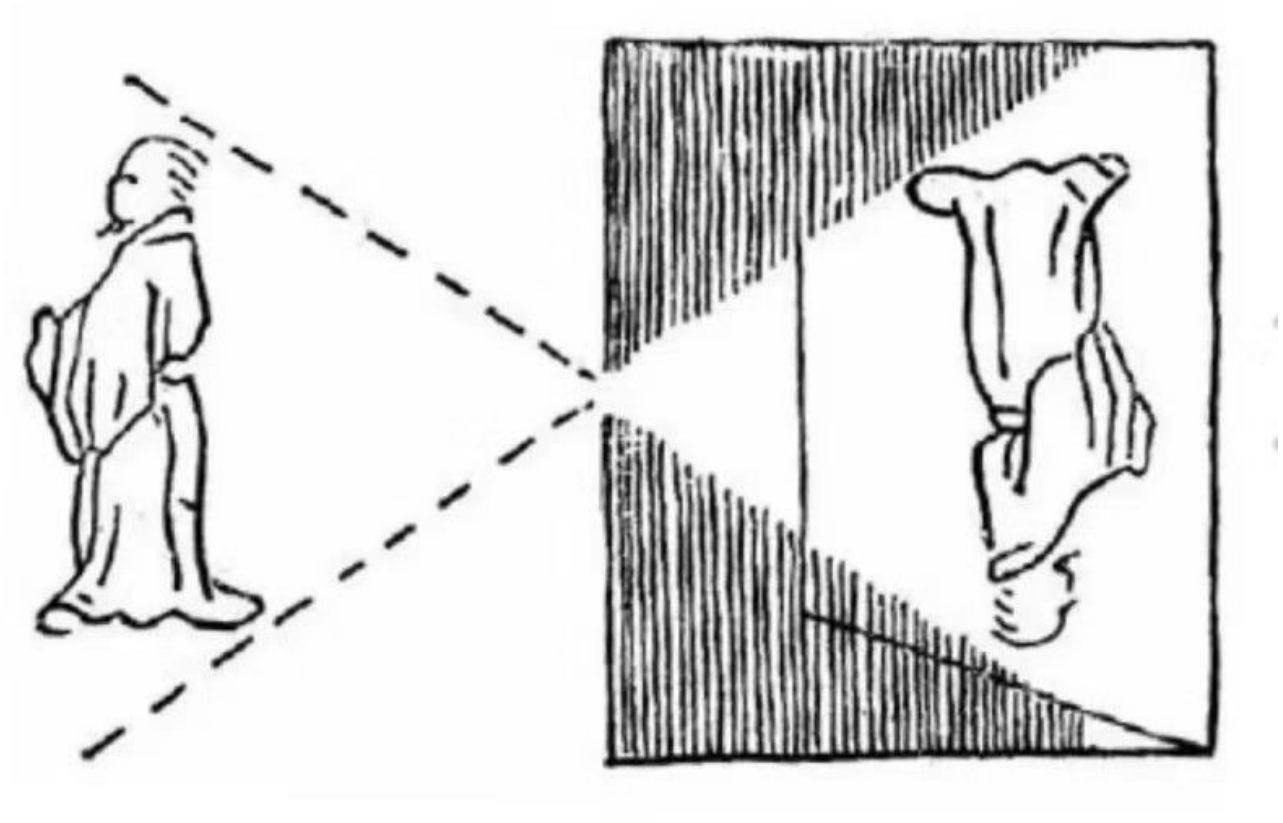
Image Formation

- Imaging: Projection of 3D scene onto 2D plane. We need to understand the geometric and photometric relation between the scene and its image.
  - Pinhole Camera
  - Image Formation using Lenses

This lecture is mainly adapted from the lecture notes by Shree K. Nayar.
https://fpcv.cs.columbia.edu

Pinhole Camera

- A very long history



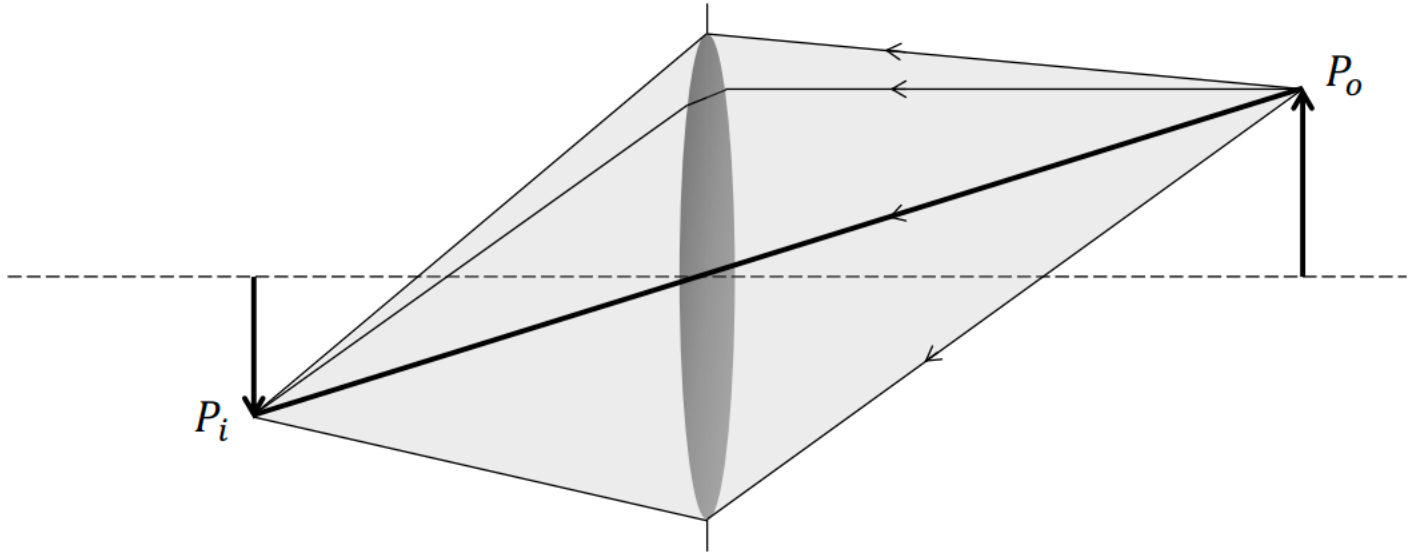
“景到，在午有端，与景长。说在端。”

—《墨经》

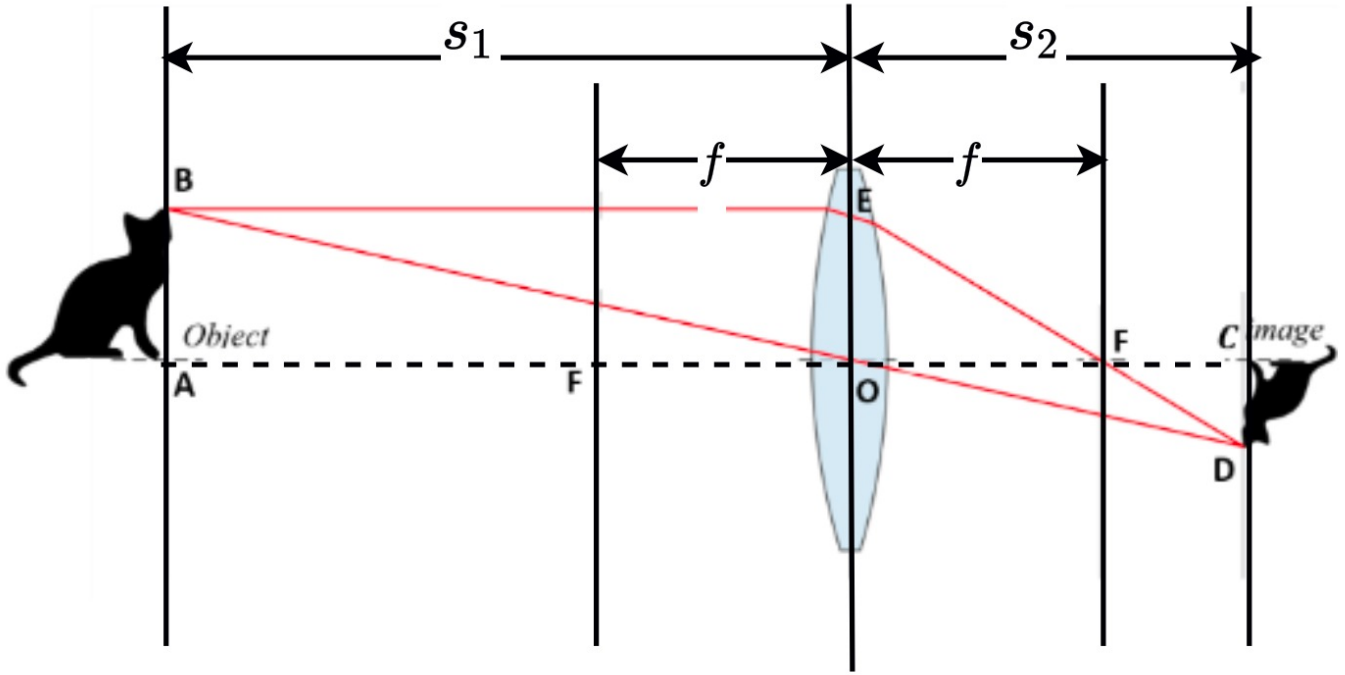
物体的投影之所以会出现倒像，是因为光线为直线传播，在针孔的地方，不同方向射来的光束互相交叉而形成倒影。

Image Formation Using Lenses

- All of the rays of light received by the lens from the point P_0 are refracted to converge at the point P_i. Gather more light!
- Focal length (f) determines the lens' bending power.



Gaussian Lens Law



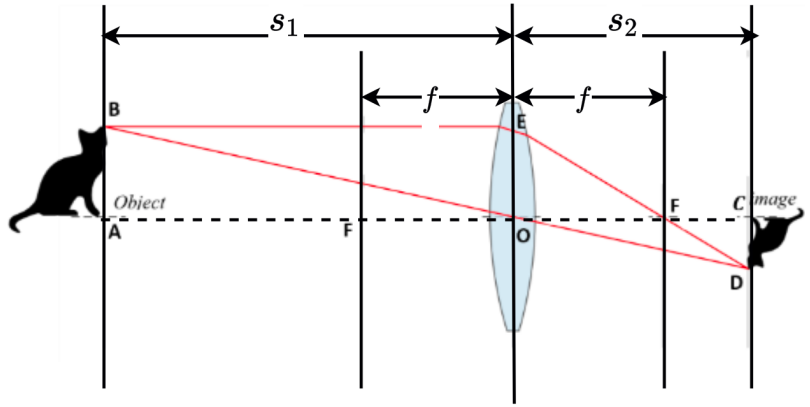
s_1: object distance

s_2: image distance

f: focal length

$$\frac{1}{s_1}+\frac{1}{s_2}=\frac{1}{f}$$

Gaussian Lens Law



$$\frac{CD}{EO} = \frac{CF}{OF} = \frac{S_2 - f}{f}$$

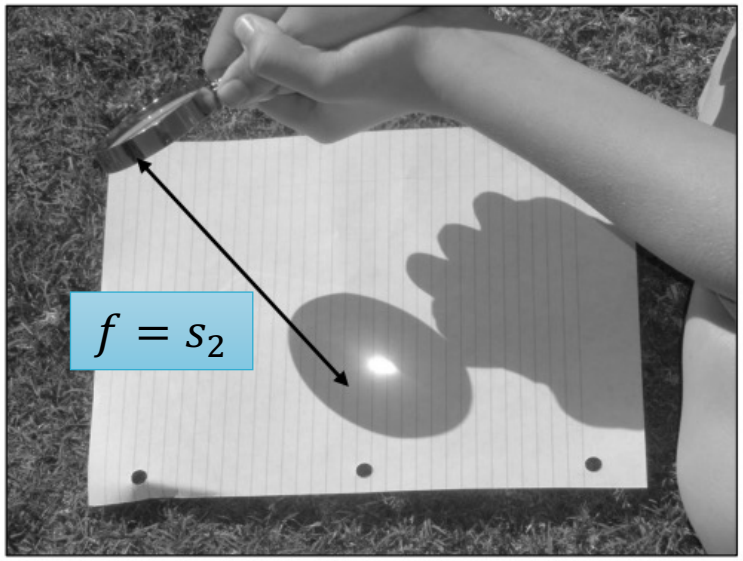
$$EO = AB \longrightarrow \frac{CD}{EO} = \frac{CD}{AB} = \frac{OC}{AO} = \frac{S_2}{S_1}$$

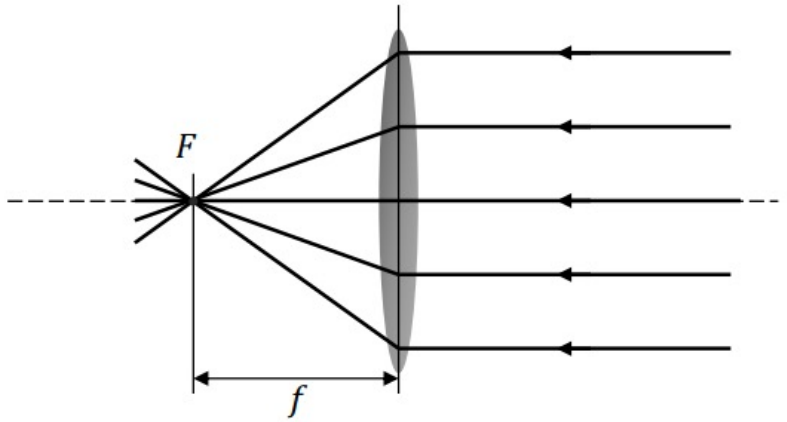
Therefore

$$\frac{S_2}{f} - 1 = \frac{S_2}{S_1} \Rightarrow \frac{1}{f} - \frac{1}{S_2} = \frac{1}{S_1} \Rightarrow \frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}.$$

Focal Length

$$\frac{1}{s_1} + \frac{1}{s_2} = \frac{1}{f} \Rightarrow If\ s_1 = \infty, then\ f = s_2$$

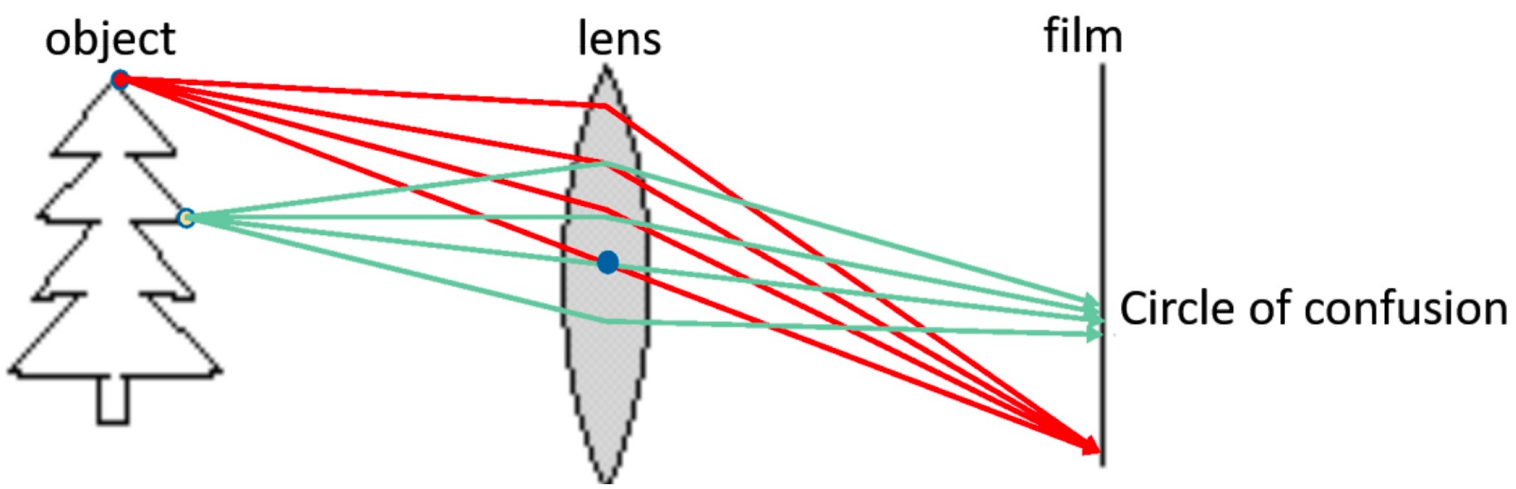


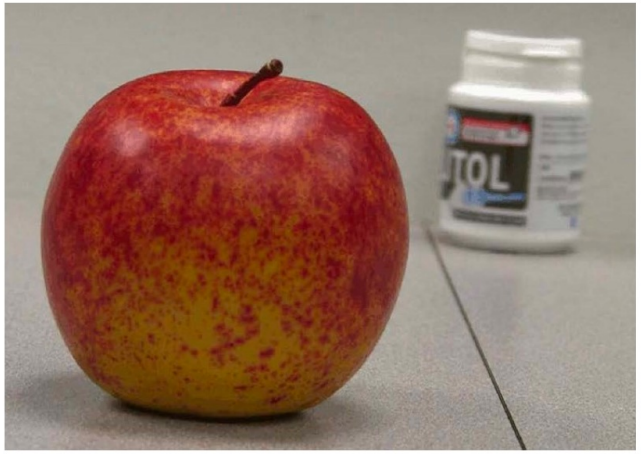


Focal length: distance at which incoming rays that are parallel to the optical axis converge.

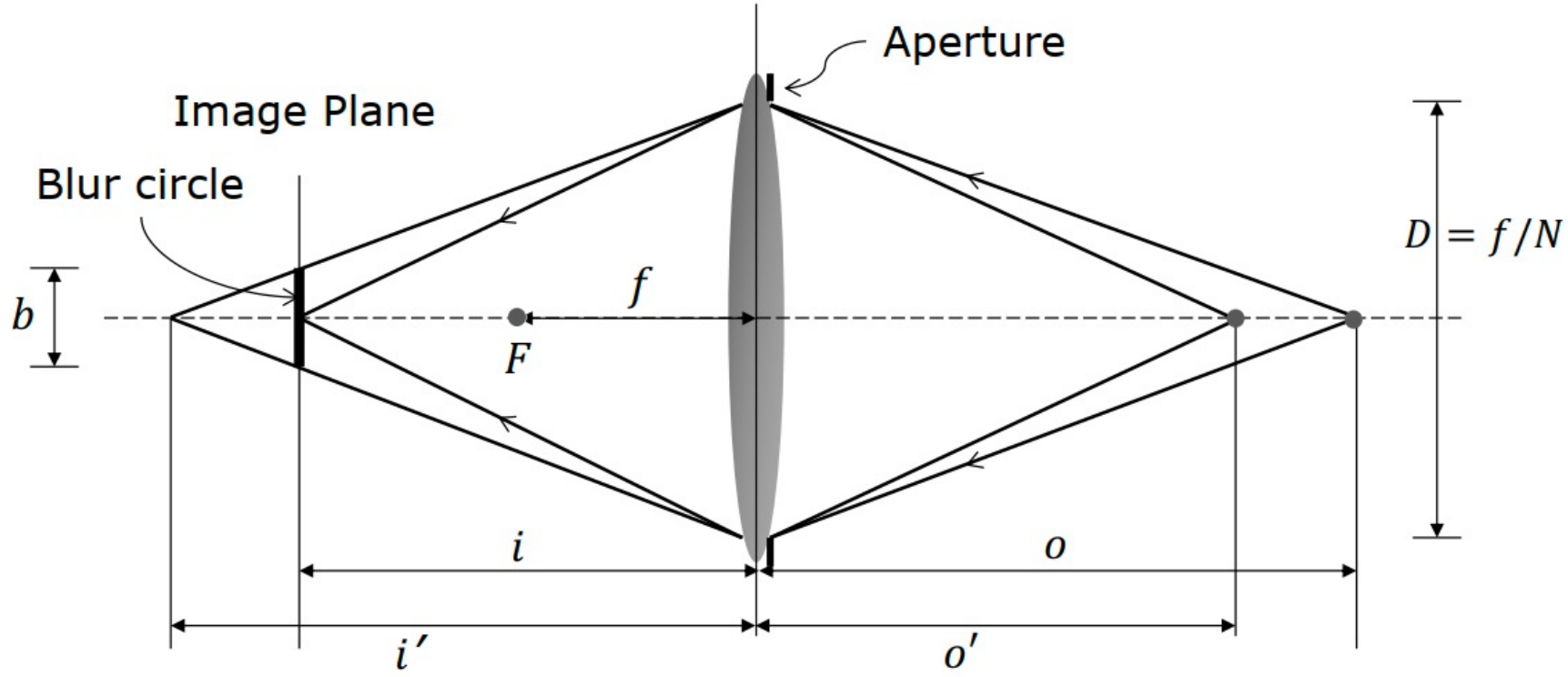
Lens Defocus

- Only one plane in the scene that is perfectly focused.
- Any object that lies outside of this plane will be out of focus in the image.





Lens Defocus



From similar triangles:

$$\frac{b}{D} = \frac{|i' - i|}{i'}$$

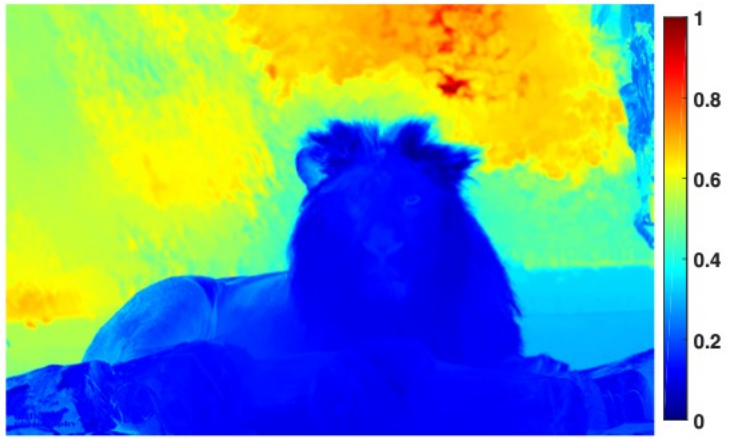
Blur circle diameter:

$$b = \frac{D}{i'}|i' - i| \qquad b \propto D \propto \frac{1}{N}$$

Out of focus

- The objects out of the focus plane are blurred.

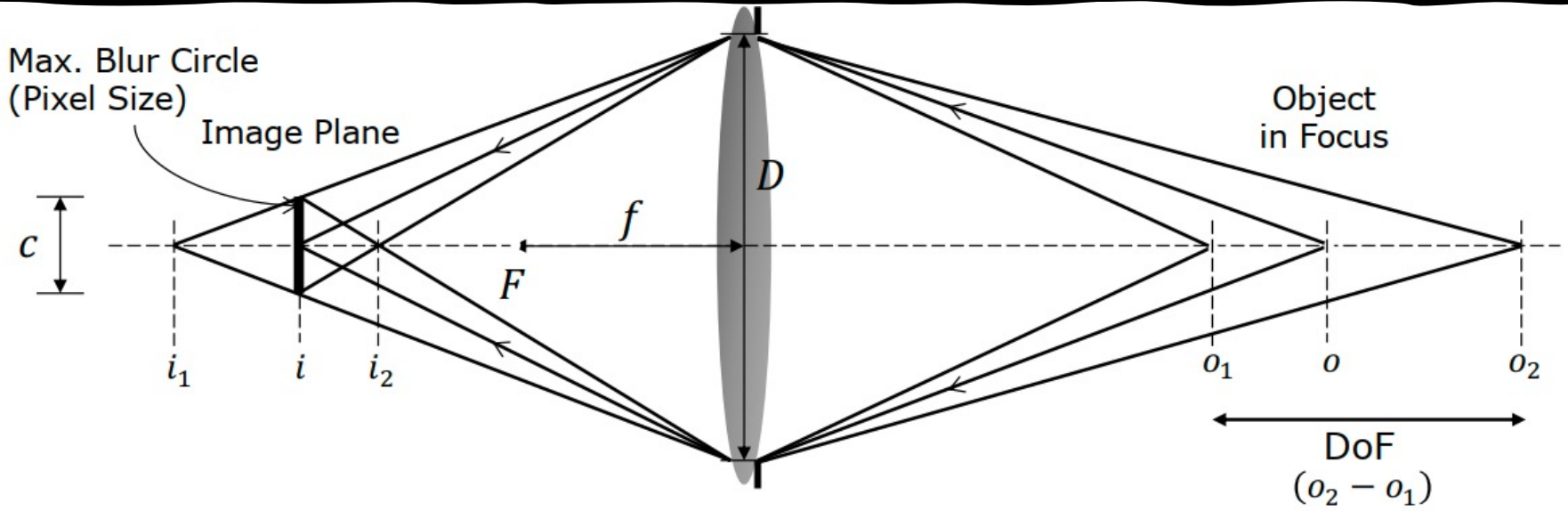
blurred image



defocus map

in-focus regions

Depth of Field (DoF)



If o_1 and o_2 are the nearest and farthest distances respectively for which blur circle is maximum c, then:

$$c = \frac{f^2(o - o_1)}{No_1(o - f)} \qquad c = \frac{f^2(o_2 - o)}{No_2(o - f)} \qquad \boxed{1}$$

Depth of Field: $\boxed{o_2 - o_1 = \frac{2of^2cN(o - f)}{f^4 - c^2N^2(o - f)^2}}$

Depth of Field (DoF)

- DoF: Range of object distances over which the image is “sufficiently well” focused, i.e., range over which blur, is less than pixel size.

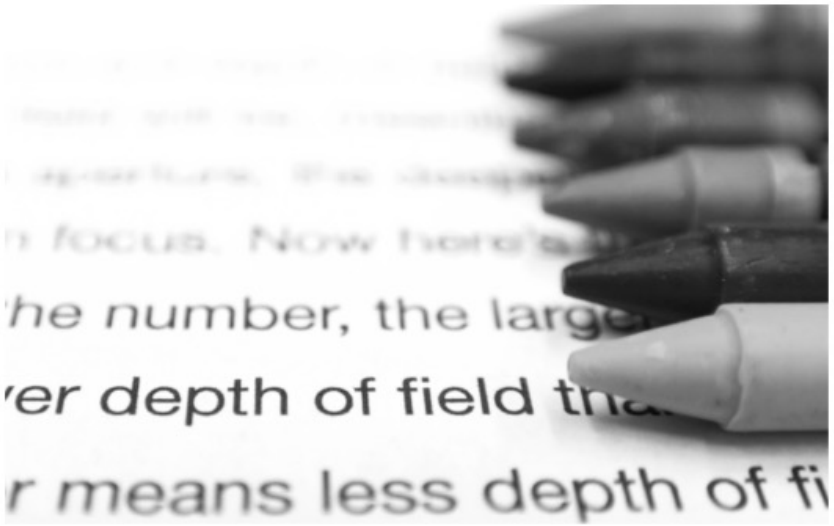
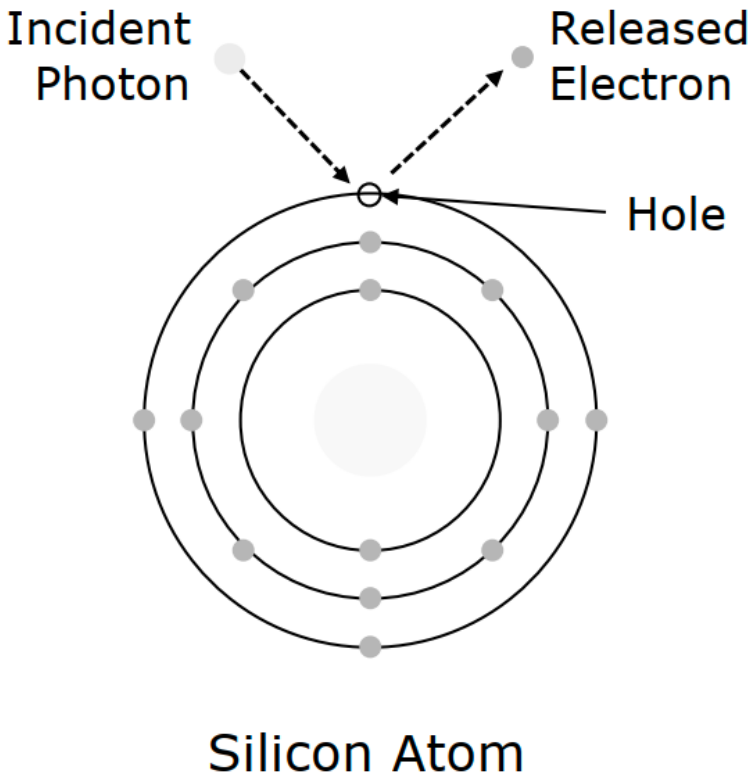


Image Sensing

- Converting light into electric charge



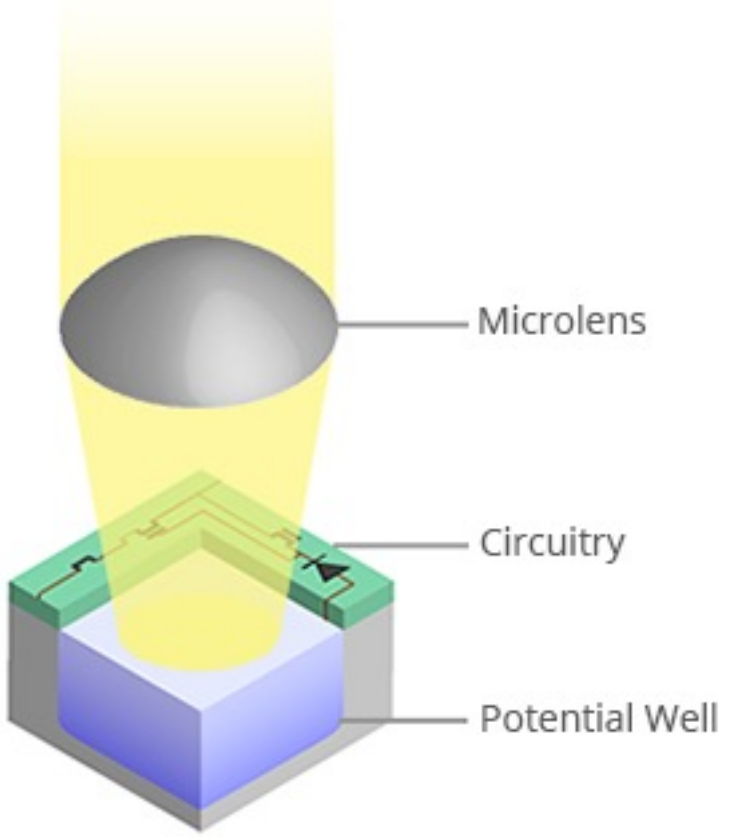
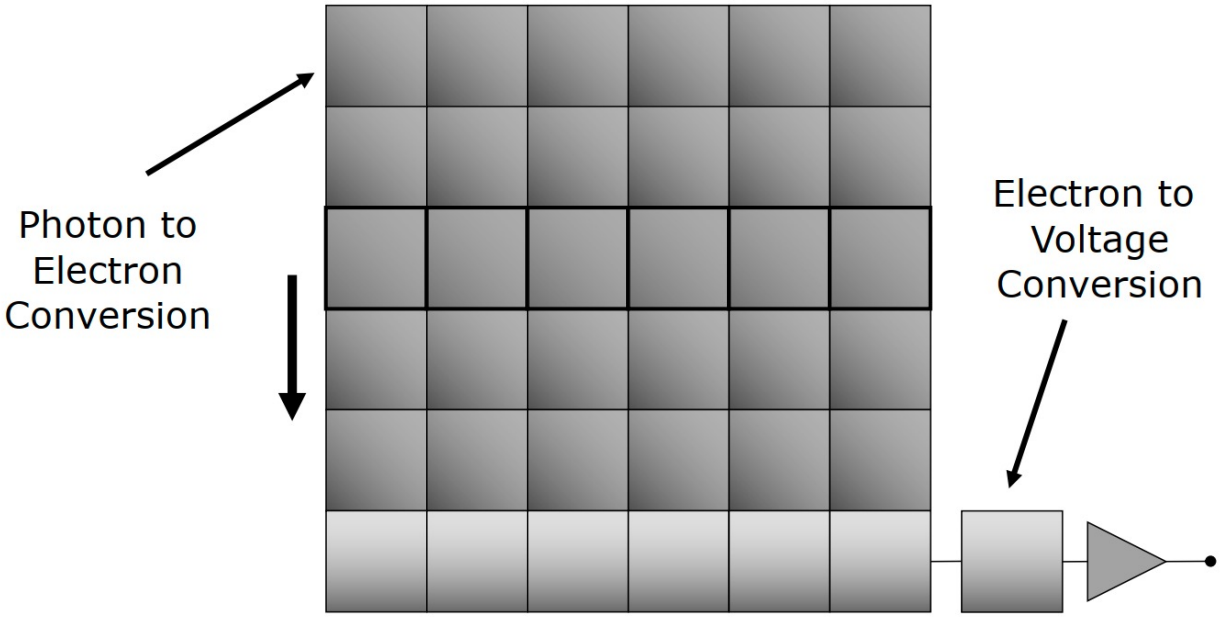


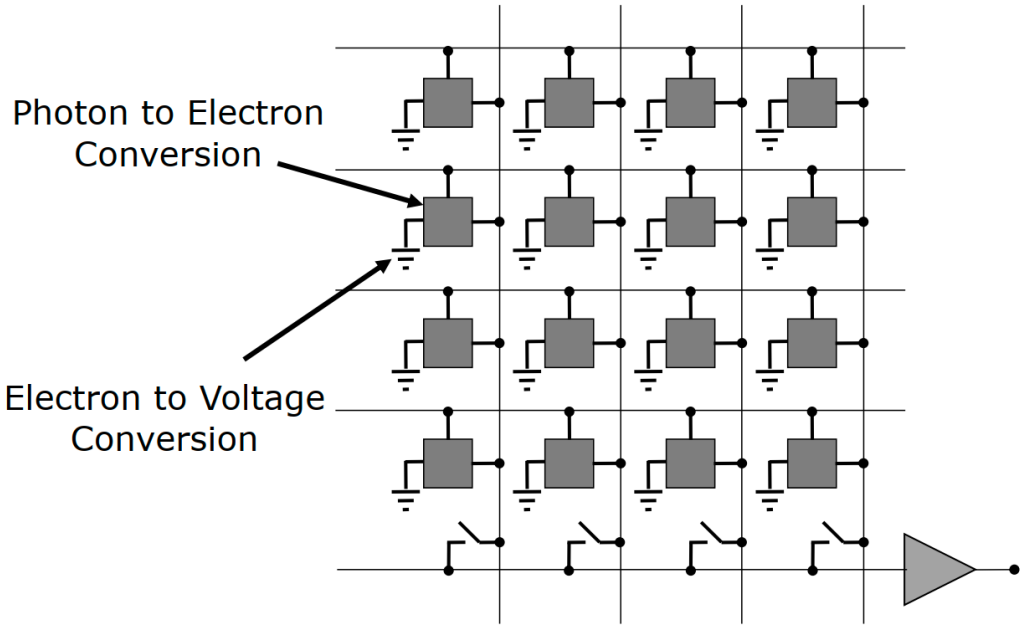
Image Sensors

CCD: Charge Coupled Device



lower speed, higher quality

CMOS: Complementary Metal-Oxide Semiconductor



higher speed, lower quality

Sensor Dynamic Range

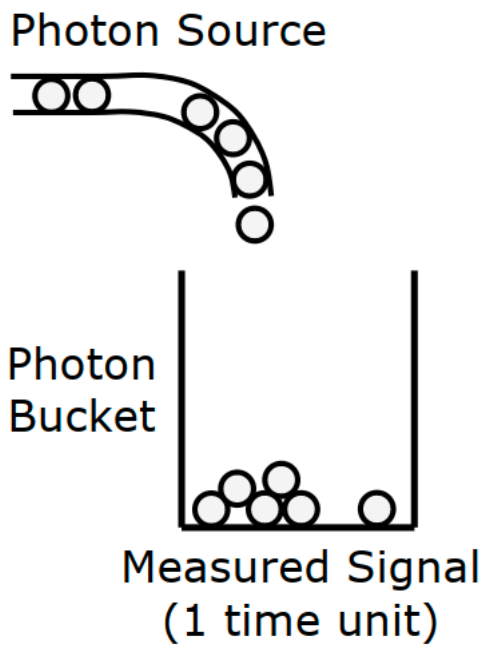
Dynamic range = $20 \log \left(\frac{B_{max}}{B_{min}} \right)$

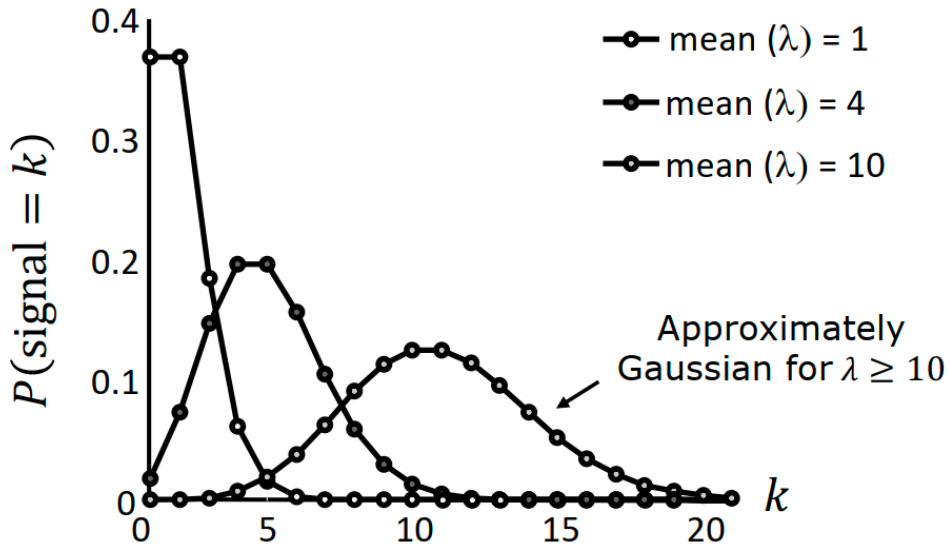
B_{max} : The maximum possible photon energy (full potential well)

B_{min} : The minimum detectable photon energy (in the presence of noise)

Sensor	$B_{max}:B_{min}$	dB
Human Eye	1,000,000:1	120
HDR Display	200,000:1	106
Digital Camera	4096:1	72.2
Film Camera	2948:1	66.2
Digital Video	45:1	33.1

Sensor Noise





$$P(\text{signal} = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

Photon Shot Noise: Poisson Distribution

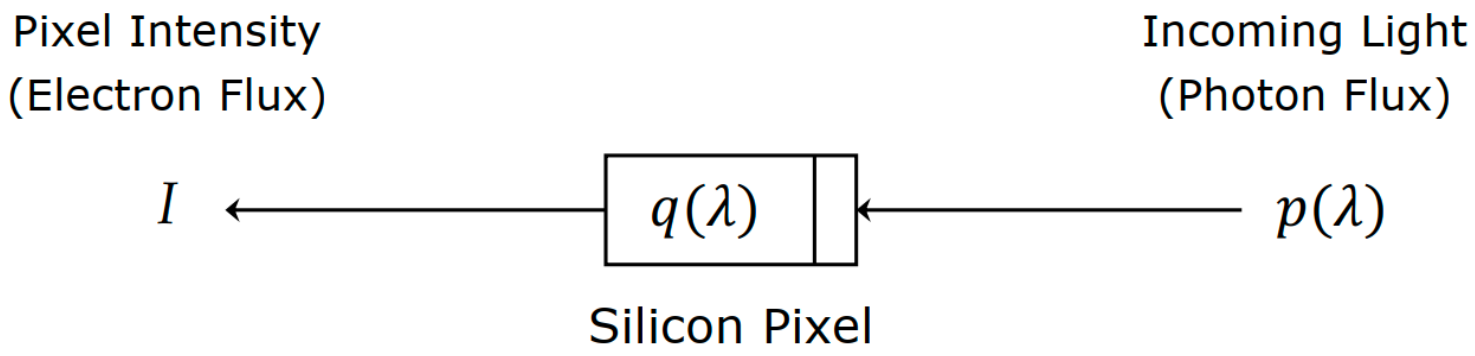
Sensor Noise

- Read out noise: scene independent Gaussian noise; introduced during the conversion of electrons to a voltage; depending on sensor quality
- Quantization noise: introduced during converting the measured analog voltage to an integer value using an analog-to-digital convertor(ADC)
- Dark current noise: electrons are generated due to its temperature, even when there is no light arriving; significant only for long exposures.
- Fixed pattern noise: due to imprecisions inherent to the manufacturing of image sensors; can be compensated for by calibrating the response.

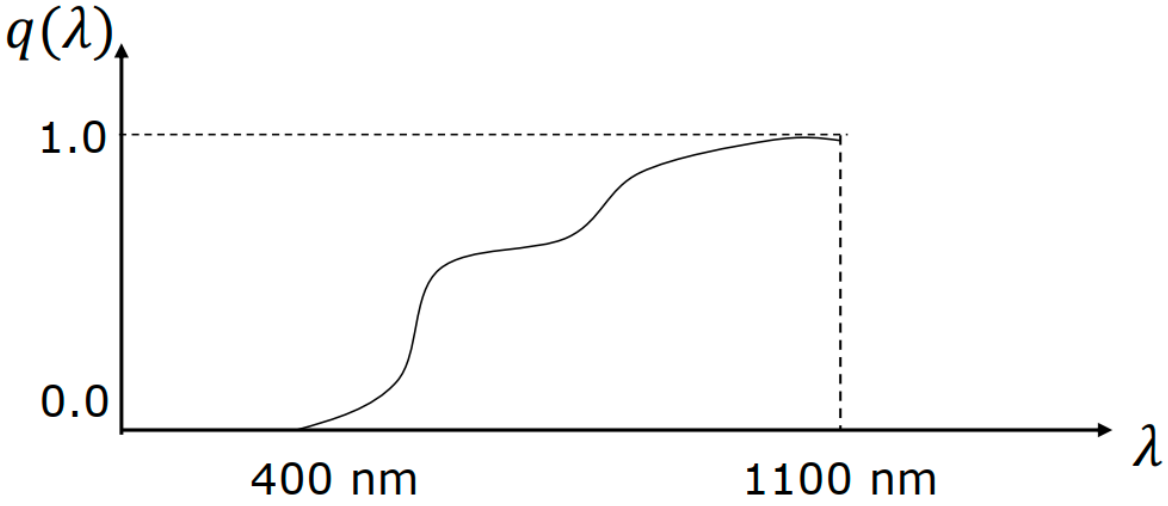
Sensing Color

- Quantum Efficiency

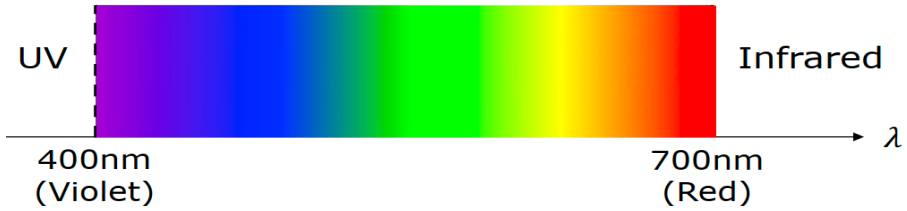
$$q(\lambda) = \frac{\text{Electron Flux Generated}}{\text{Photon Flux of wavelength } \lambda}$$

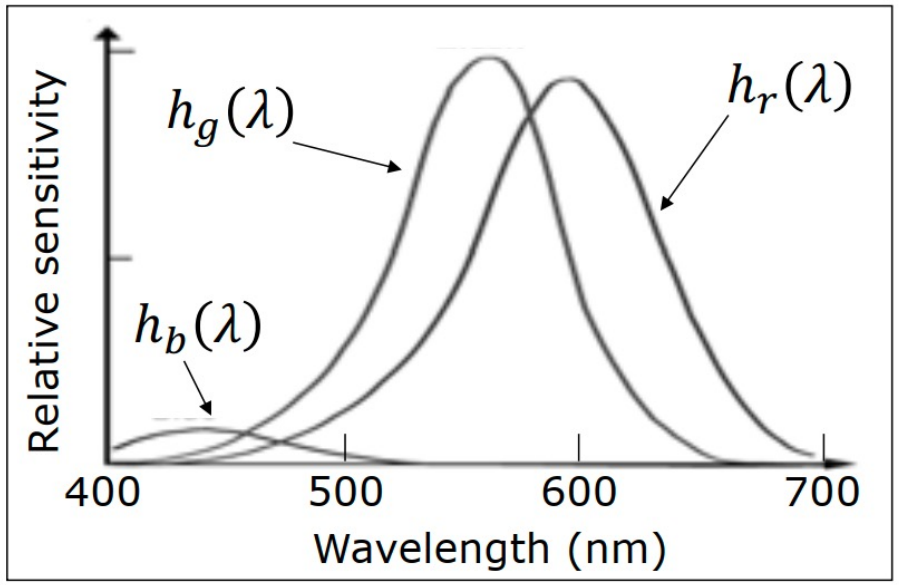


Quantum Efficiency



Quantum Efficiency of Silicon





Quantum Efficiency of three types of cones in Human eye

Tristimulus Values

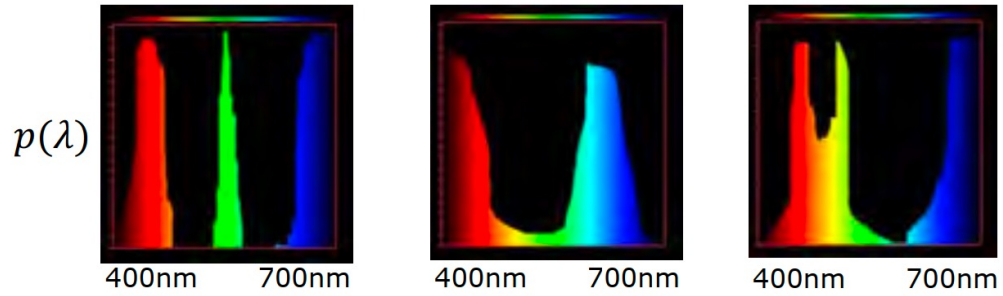
- The eye does not measure the complete spectral distribution $p(\lambda)$, but rather, only produces three intensities (R,G,B)

$$R = \int_{-\infty}^{\infty} h_r(\lambda) p(\lambda) d\lambda$$

$$G = \int_{-\infty}^{\infty} h_g(\lambda) p(\lambda) d\lambda$$

$$B = \int_{-\infty}^{\infty} h_b(\lambda) p(\lambda) d\lambda$$

There exists indistinguishable distributions that generate the same R,G,B values



Same perceived color:

(R,G,B)=(115,60,108)=

The Mixing of Colors

Young's Experiment on Color Mixture

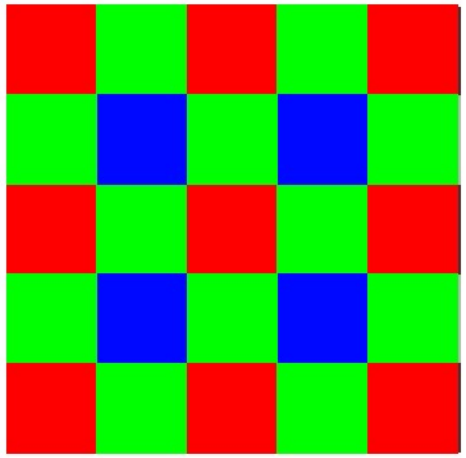
Human Sensation of nearly all colors can be produced using 3 wavelengths!

$$(\lambda_r,\ \lambda_g,\ \lambda_b) = (650,530,410)nm$$

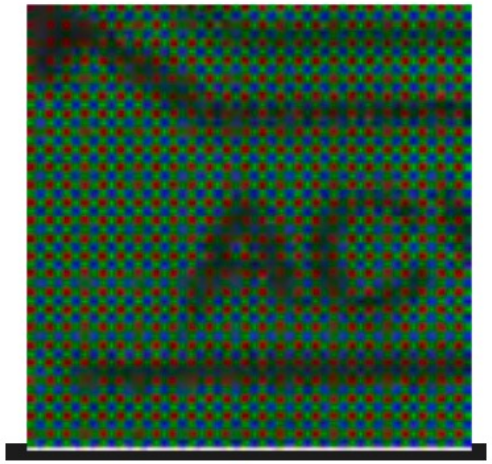
Hence, cameras and displays often use 3 filters:

(red, green, blue)

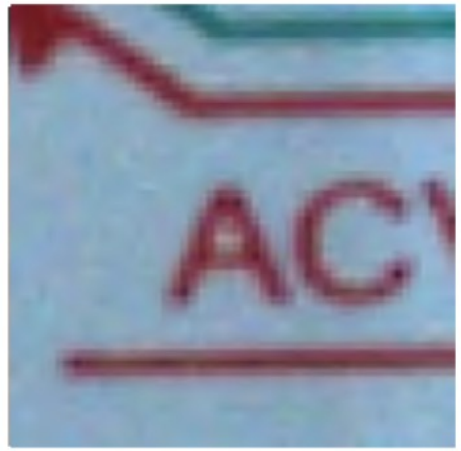
Sensing Color



Bayer Pattern
(Color Filter Mosaic)



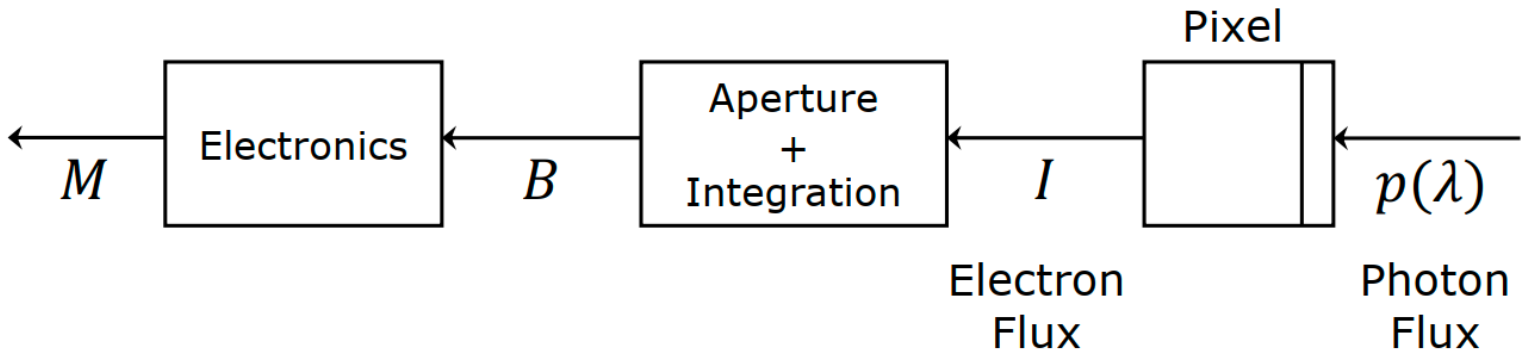
Raw Image



Interpolated Image

Color Filled in by Interpolation (Demosaicing)

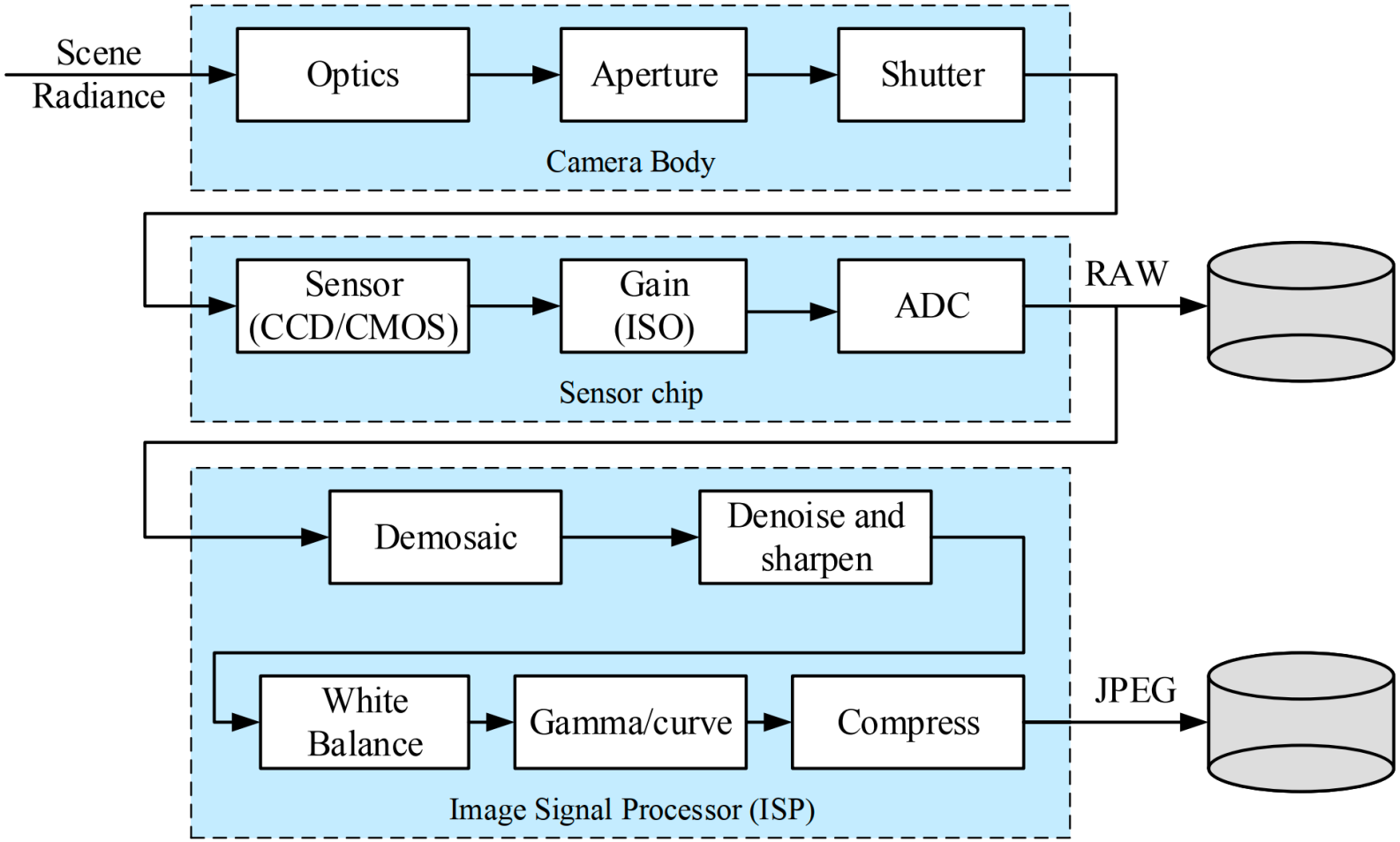
Image Sensing Pipeline



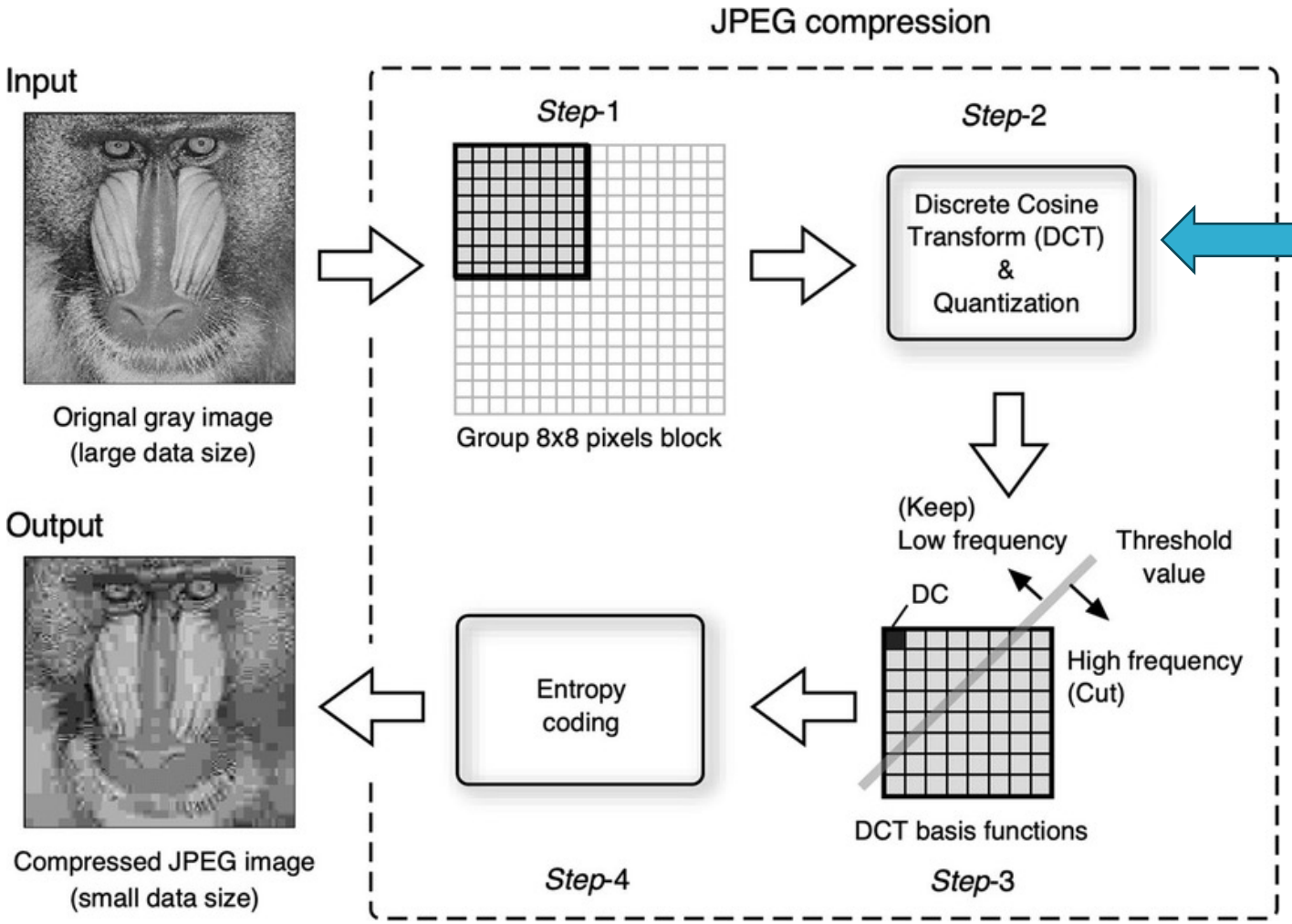
Measured Brightness: $\boxed{M = f(B) = f(I \cdot e)}$

where Exposure, $e = \left(\frac{\pi d^2}{4}\right) \cdot t$

Image Sensing Pipeline



JPEG Compression



Quantization (lossy):

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right)$$

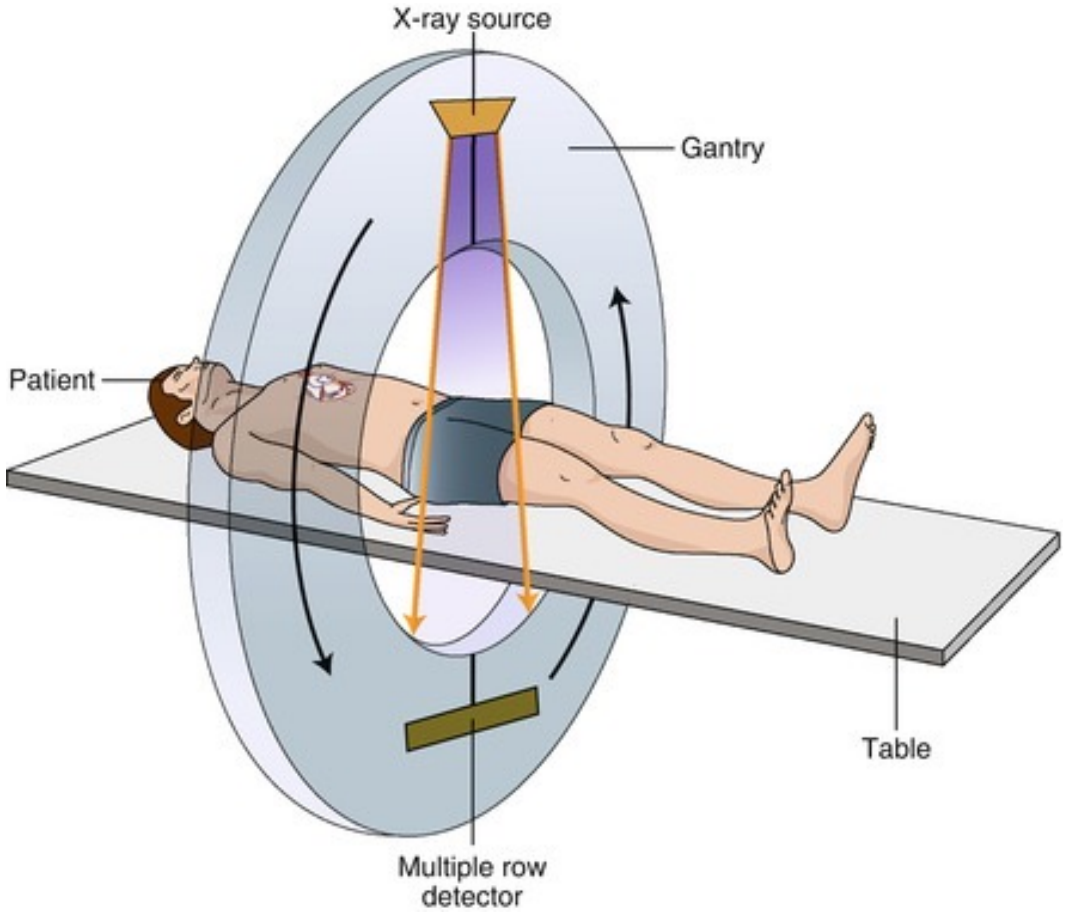
for $j, k = 0,1, \ldots, 7.$

G: the unquantized DCT coefficients
Q: the quantization matrix

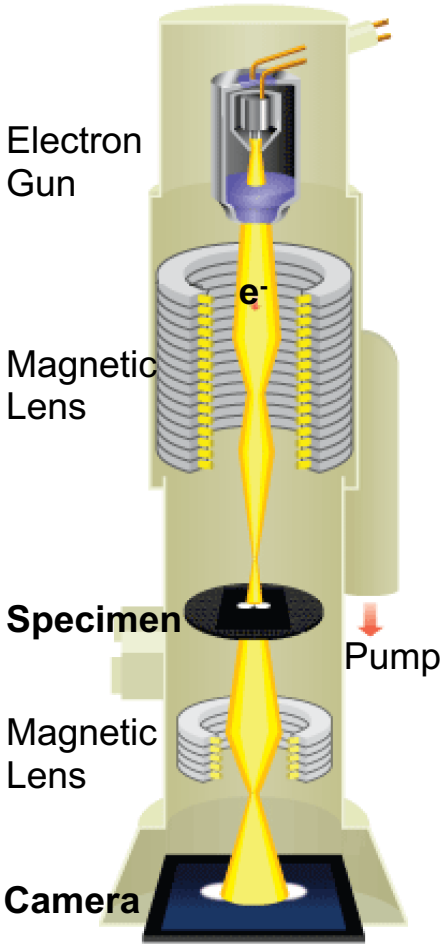
PNG Compression

- PNG is a loss-less compression format: the compressed file can reconstruct the source image exactly.
- Compression algorithm: **Deflate,** a combination of LZ77 and Huffman coding.
- JPEG vs. PNG
  - JPEG: smaller size; faster loading time
  - PNG: higher quality; supports transparency.

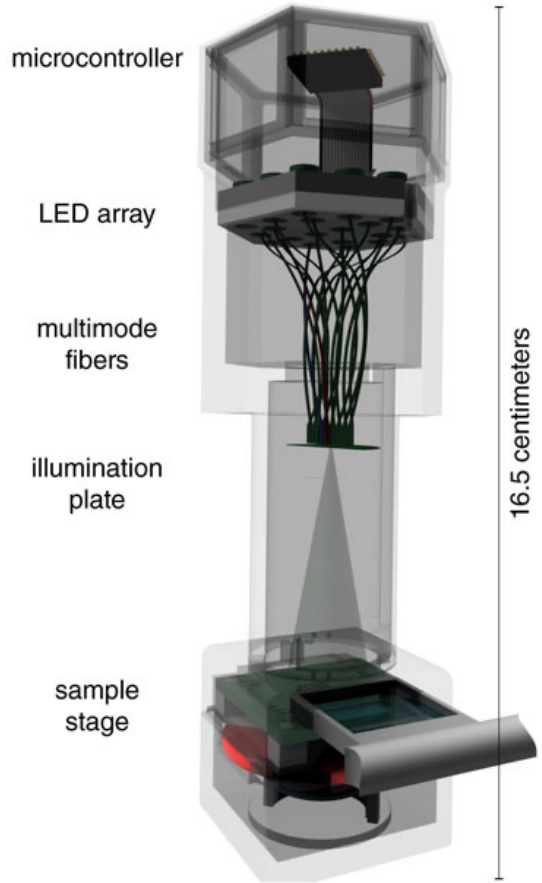
Image Devices Beyond Cameras



CT scanner

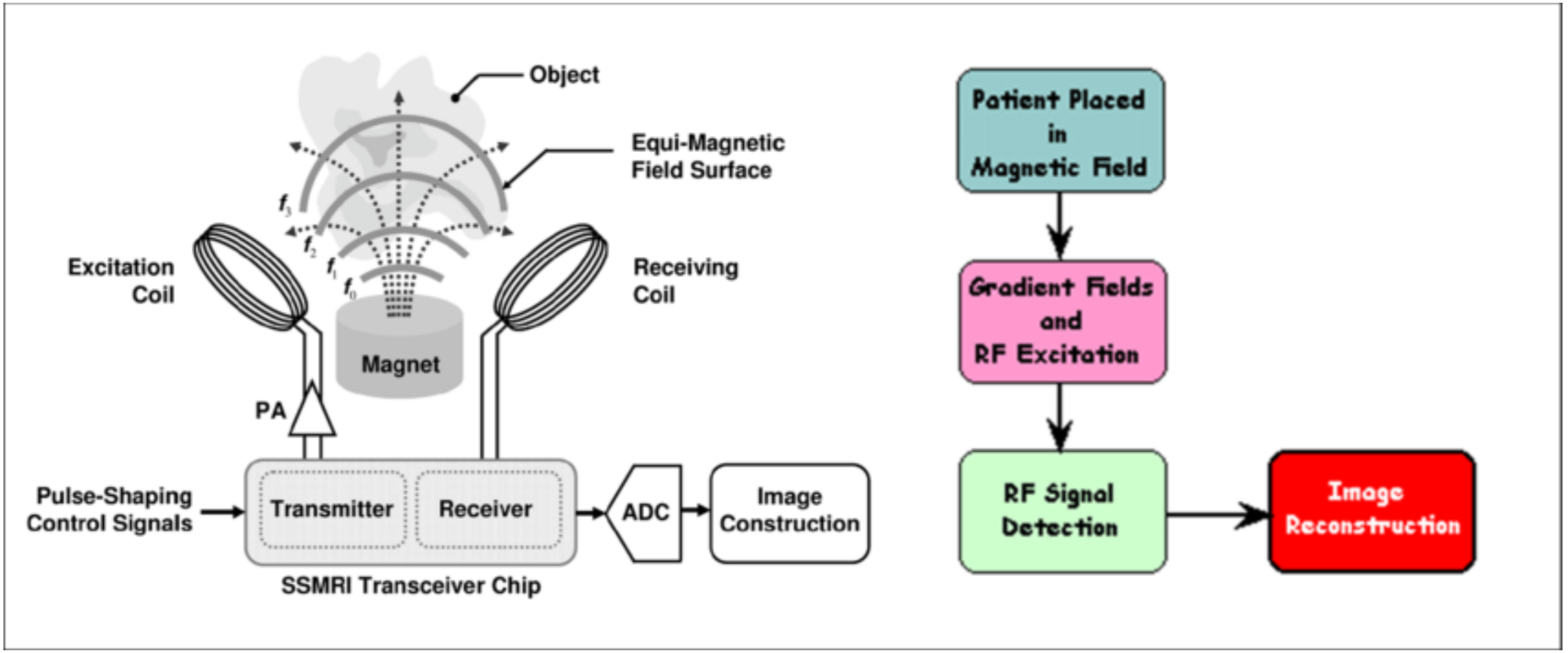


Electron Microscope



Lensless Microscope

Image Devices Beyond Cameras



Magnetic Resonance Imaging (MRI)

Credits

Slides are adapted from

- cs231n: Deep Learning for Computer Vision@Stanford
- cs231A: Computer Vision:From 3D Perception to 3D Reconstruction and Beyond@Stanford
- 6.8300/6.8301: Advances in Computer Vision@MIT
- CSCI 1430: Computer Vision@Brown

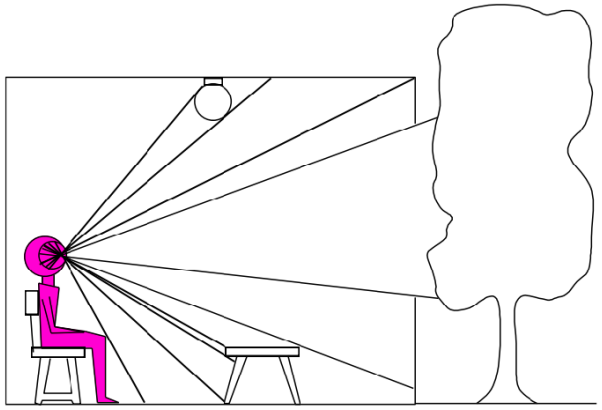
3D Vision

Cameras: 3D to 2D

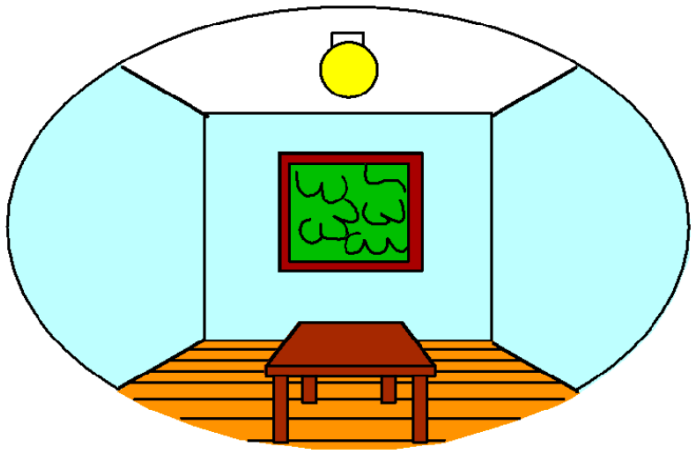
Projective Transformation

3D world

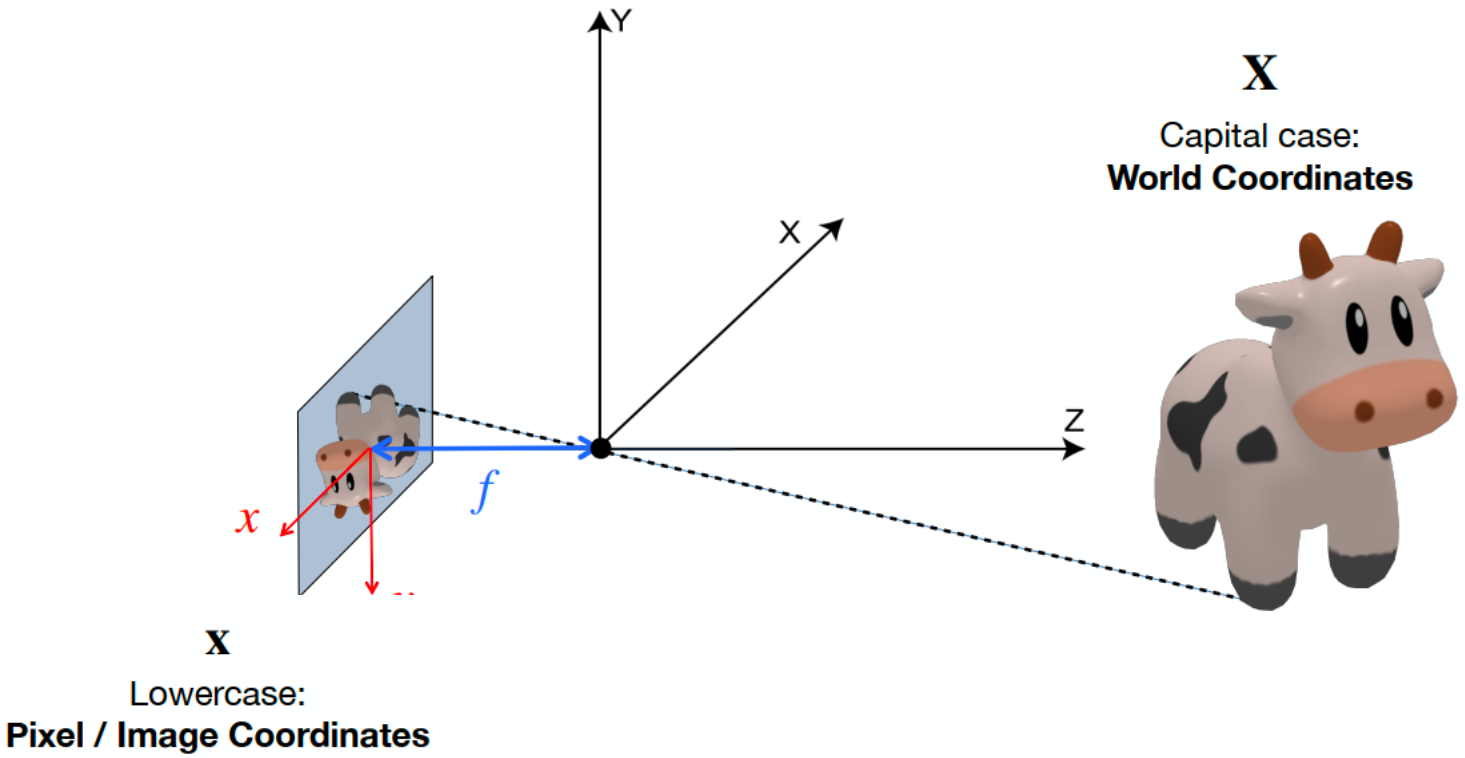
2D image



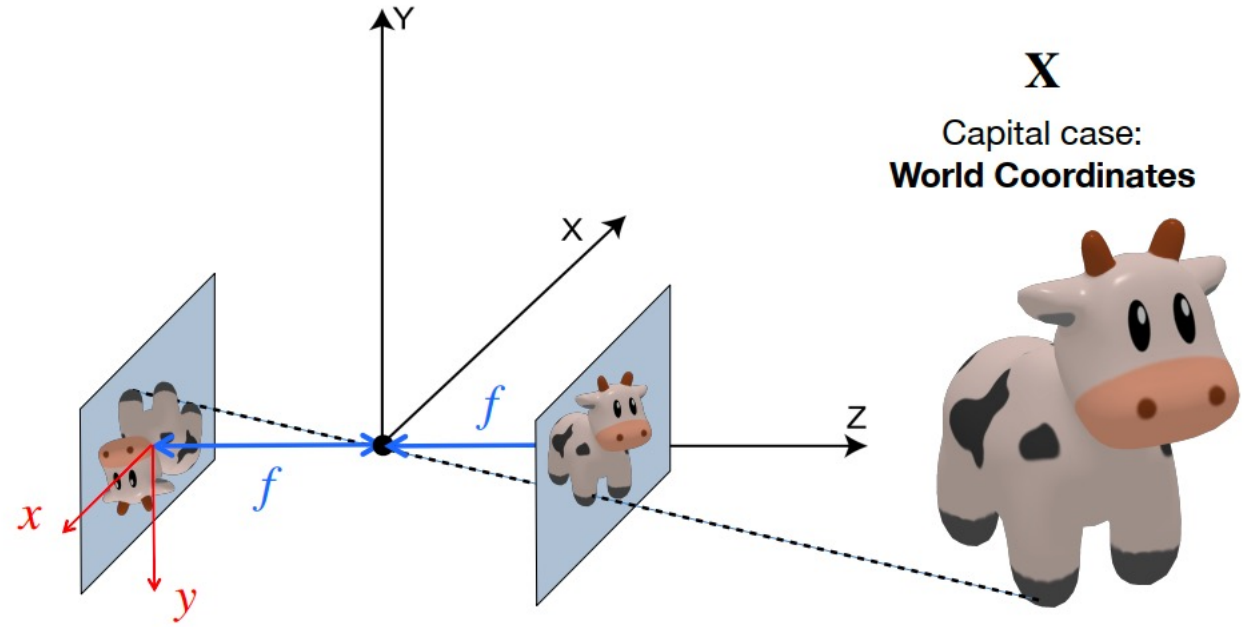
Point of observation



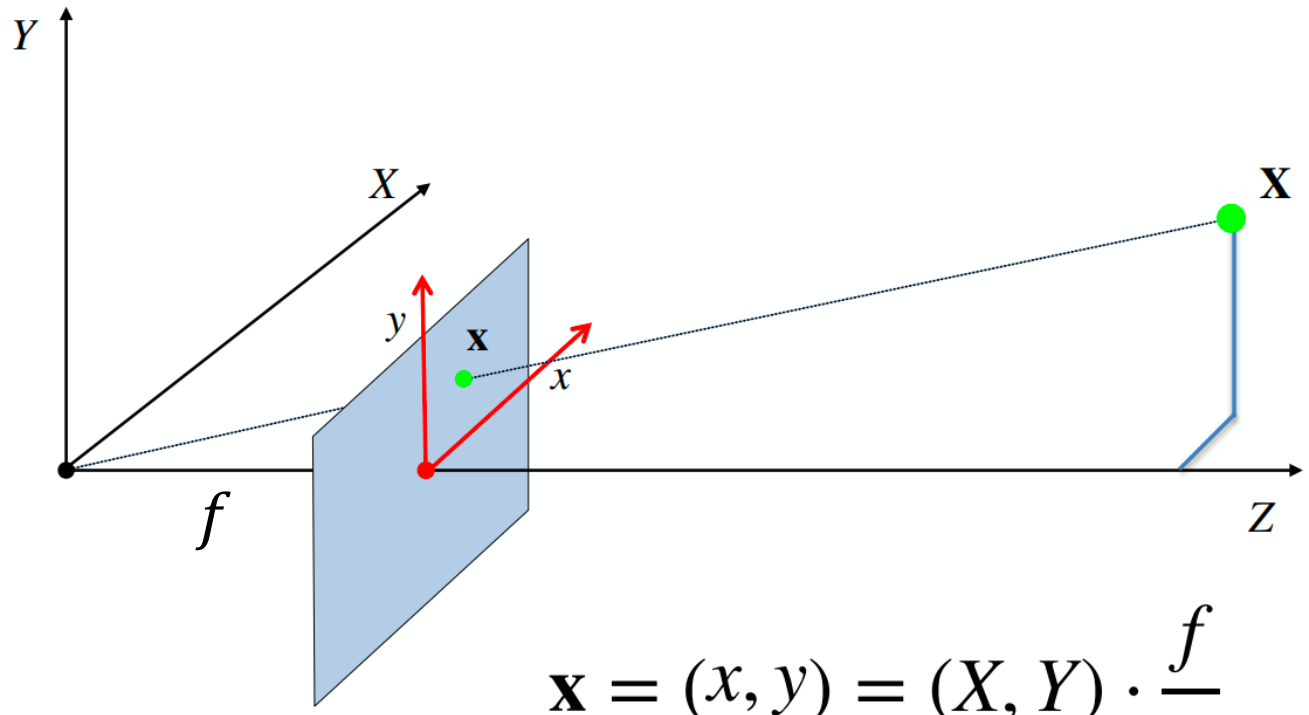
Perspective Projection



Perspective Projection



Perspective Projection



$$\mathbf{x} = (x, y) = (X, Y) \cdot \frac{f}{Z}$$

Image / Pixel Coordinates

World Coordinates

Homogeneous Coordinates

- Converting to homogeneous coordinates

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- Converting from homogeneous coordinates

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x/w \\ y/w \end{pmatrix}$$

Scale invariance in projection space

$$k\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} kx \\ ky \\ kw \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{kx}{kw} \\ \frac{ky}{kw} \end{bmatrix} = \begin{bmatrix} \frac{x}{w} \\ \frac{y}{w} \end{bmatrix}$$

Homogeneous Coordinates

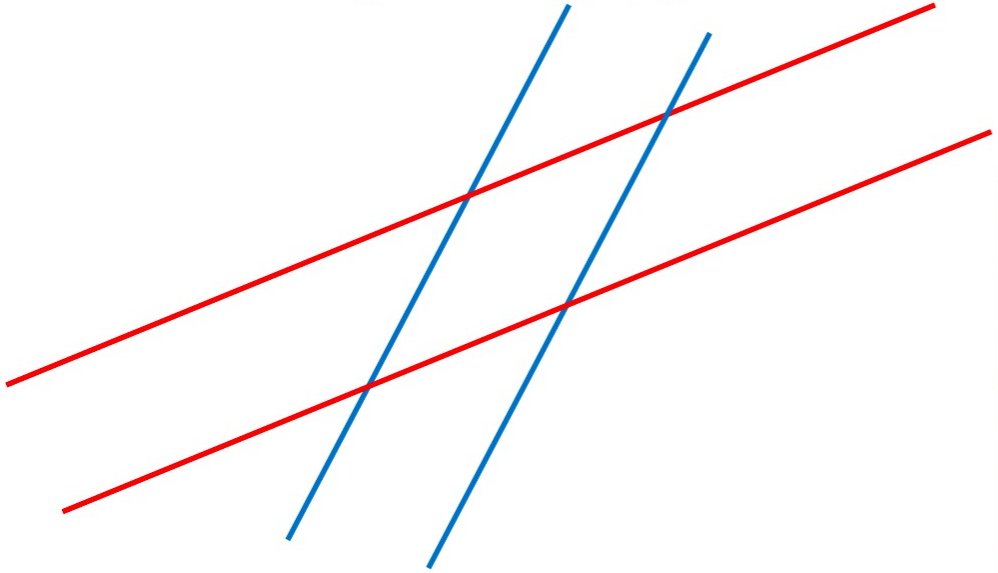
Cartesian Coordinates

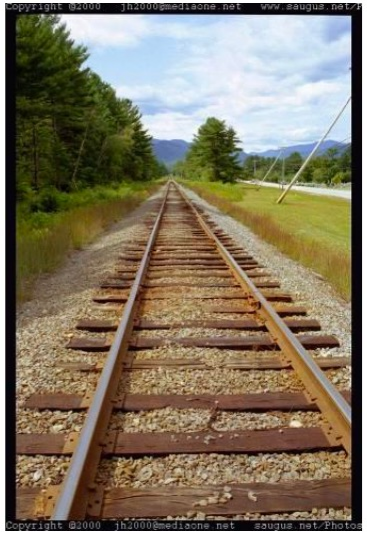
e.g., we can uniformly scale the projective space, and it will still produce the same image -> scale ambiguity

Homogeneous Coordinates

Cartesian: (Inf, Inf)
Homogeneous: (1, 1, 0)

Cartesian: (Inf, Inf)
Homogeneous: (1, 2, 0)





Basic 2D Transformations

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & \alpha_x \\ \alpha_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shear

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta \\ \sin\Theta & \cos\Theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Affine

Affine is any combination of translation, scale, rotation, and shear

Perspective Projection

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\mathbf{X} \Rightarrow w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \overbrace{\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}}^{\mathbf{K}} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Allowing origin shift (u_0, v_0) and skew (s)

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \quad \mathbf{0}]\mathbf{X} \Rightarrow w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

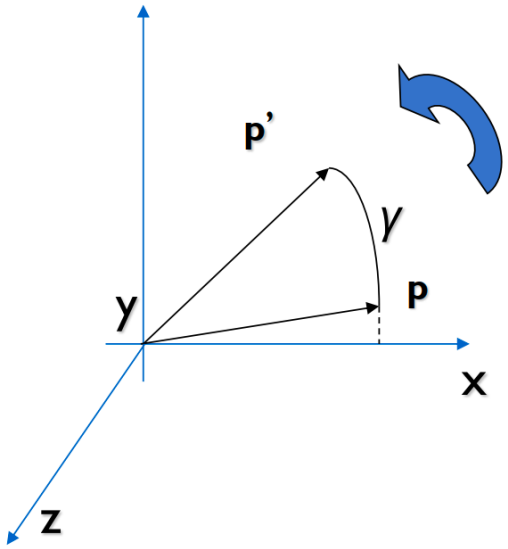
Camera Translation

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{t}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\ v\\ 1\end{bmatrix} = \begin{bmatrix}f_x & s & u_0\\ 0 & f_y & v_0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & t_x\\ 0 & 1 & 0 & t_y\\ 0 & 0 & 1 & t_z\end{bmatrix}\begin{bmatrix}x\\ y\\ z\\ 1\end{bmatrix}$$

Camera Rotation

3D Rotation

Rotation around the coordinate axes, counter-clockwise:



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Camera Matrix

$$\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{X}$$

Extrinsic Matrix

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Degree of Freedom: 5 3+3

3D Vision

- Camera Simulation

$$\underbrace{\mathbf{x}}_{\text{Unknown}} = \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}_{\text{Known}}\underbrace{\mathbf{X}}_{\text{Known}}$$

- Camera Calibration

$$\underbrace{\mathbf{x}}_{\text{Known}} = \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}_{\text{Unknown}}\underbrace{\mathbf{X}}_{\text{Known}}$$

$$\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$$

Intrinsics Calibration

Extrinsics Calibration / Pose Estimation

3D Vision

- Non-blind 3D Reconstruction

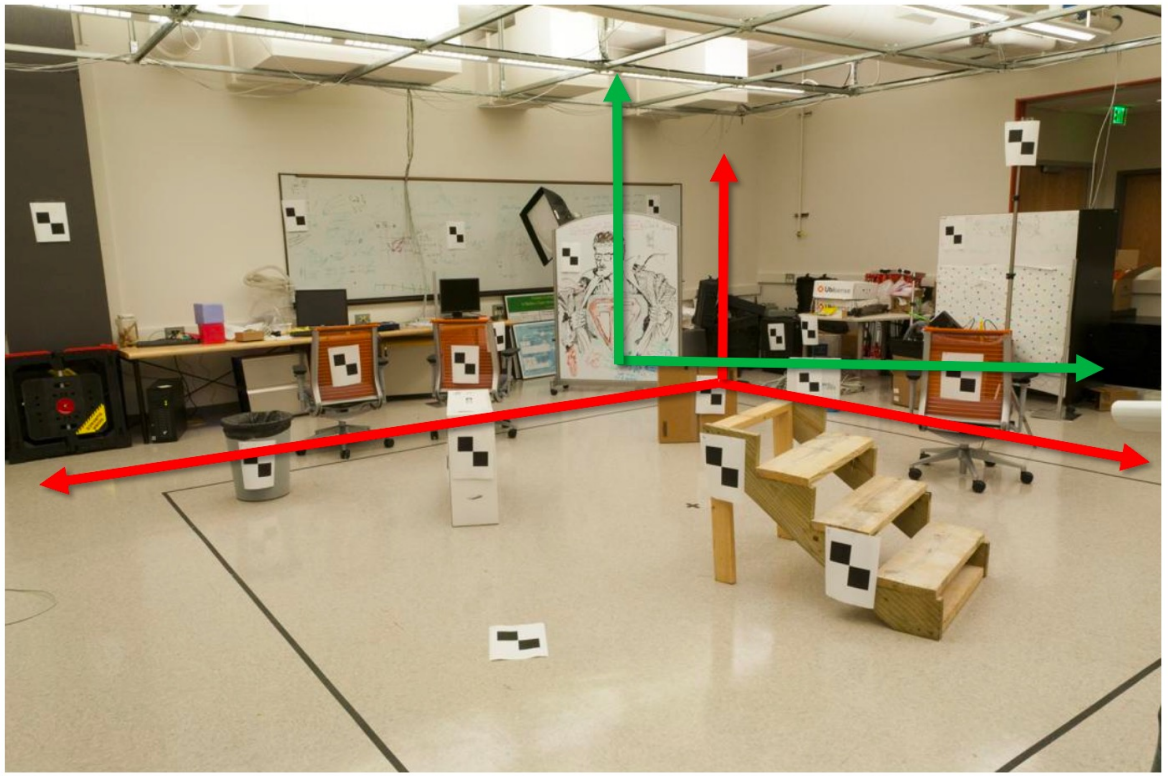
$$\underbrace{\mathbf{x}}_{\text{Known}} = \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}}_{\text{Known}\quad\text{Unknown}}$$

- Blind 3D Reconstruction

$$\underbrace{\mathbf{x}}_{\text{Known}} = \underbrace{\mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}}_{\text{Unknown}\quad\text{Unknown}}$$

Camera Calibration

World vs Camera Coordinates



Camera Calibration

Known 2D image coords

Known 3D world locations

Substitute the corresponding points into the equations, and then solve the unknown camera parameters via least squares.

Unknown Camera Parameters

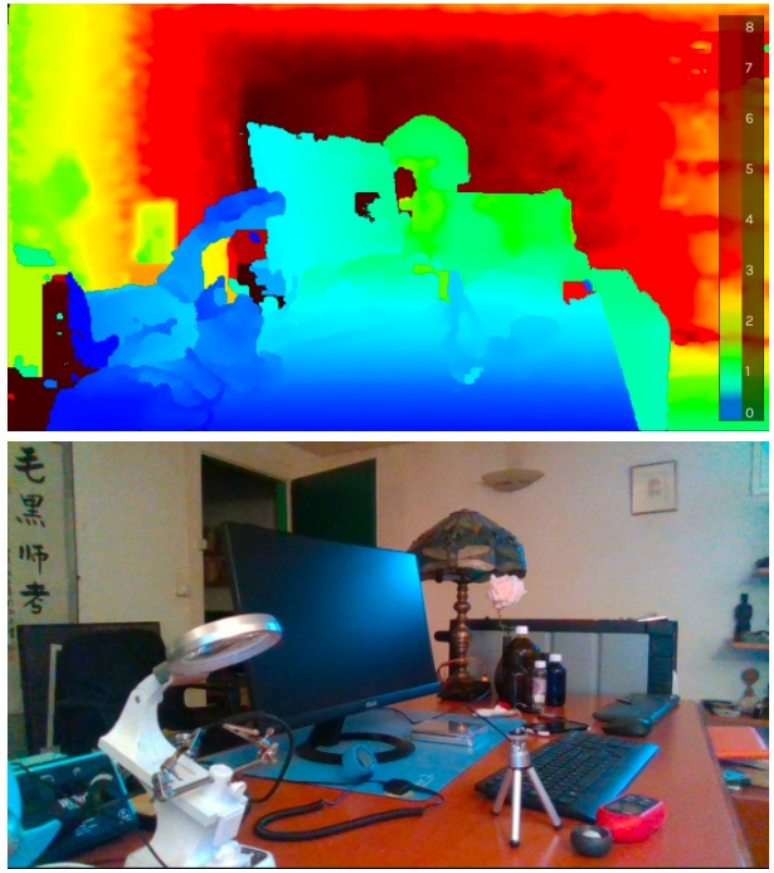
Known 2D image coords

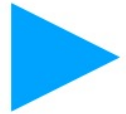
$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Known 3D locations

M is 3x4, so 12 unknowns; but scale ambiguity exists, only 11 degree of freedom.

“Up-project”: 2D to 3D





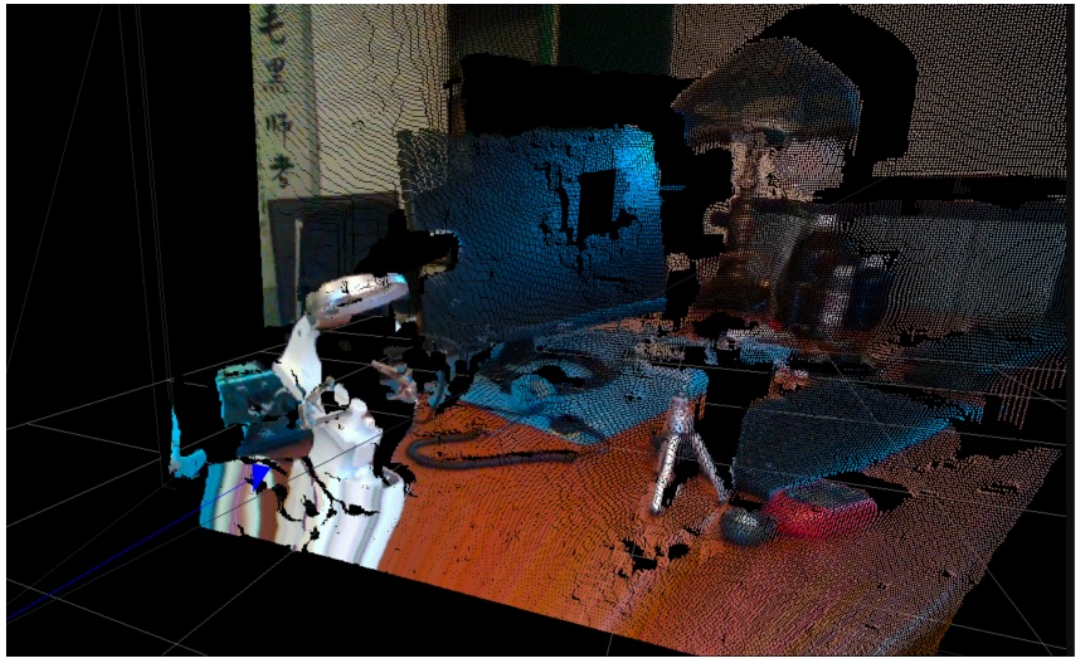


Image credits: https://www.calvert.ch/maurice/2018/06/12/intel-realsense-d435-review/

“Upproject”: 2D to 3D

- Intrinsic matrix: $\mathbf{x} = \boldsymbol{K}\boldsymbol{X}'$

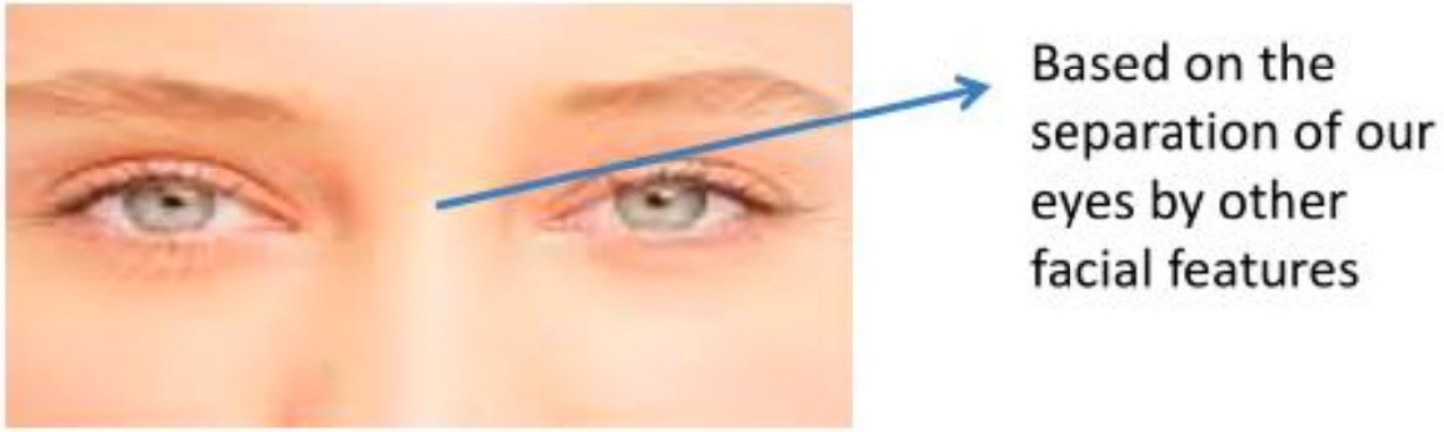
$$w\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \boldsymbol{K}\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} \Longrightarrow \begin{bmatrix} X' \\ Y' \end{bmatrix} = Z'\boldsymbol{K}^{-1}\begin{bmatrix} u \\ v \end{bmatrix}$$

- Extrisic matrix: $\boldsymbol{X}' = [\boldsymbol{R}, \boldsymbol{t}]\boldsymbol{X}$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \boldsymbol{R}^{-1}\left(\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} - \mathbf{t}\right)$$

Stereo Vision

- **Binocular disparity**, a difference in image location of an object seen by the left and right eyes.
- Stereopsis: the brain combines the two images into a single, three-dimensional perception.



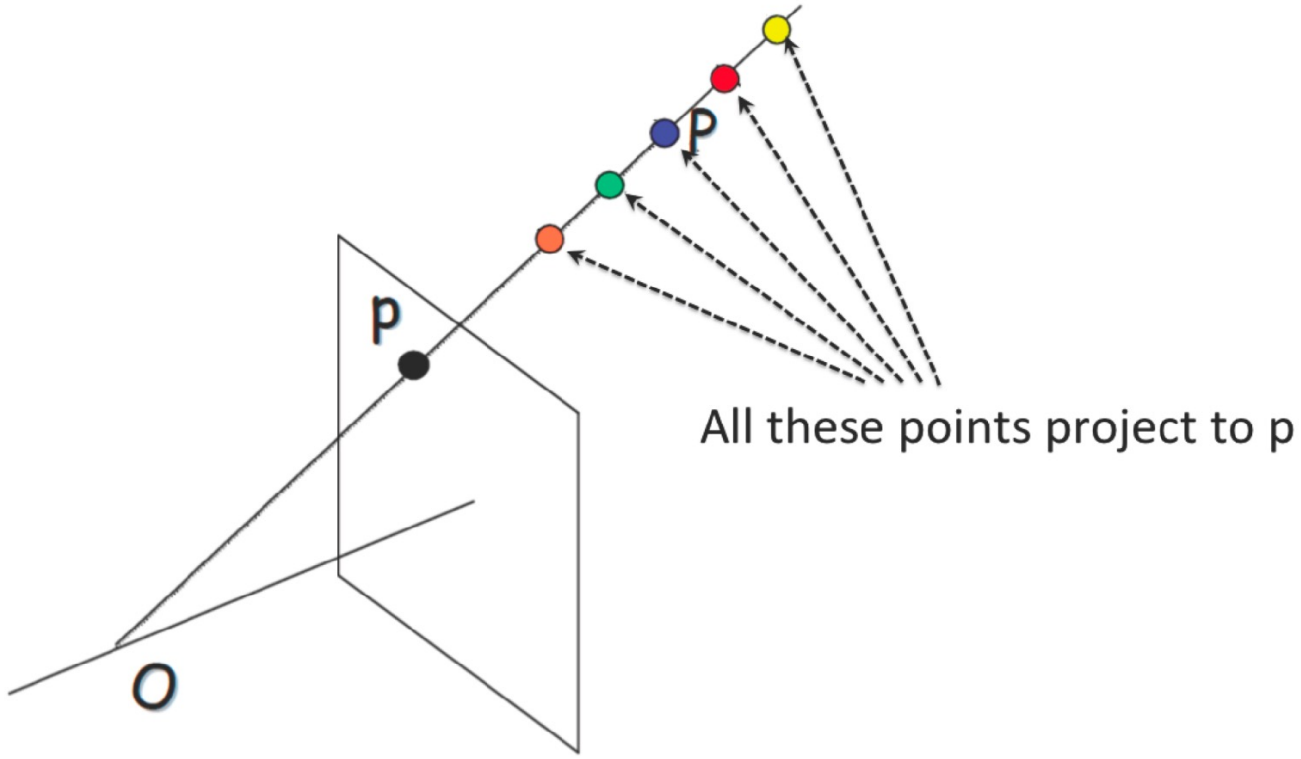
Stereo vision

Multi-camera survillance

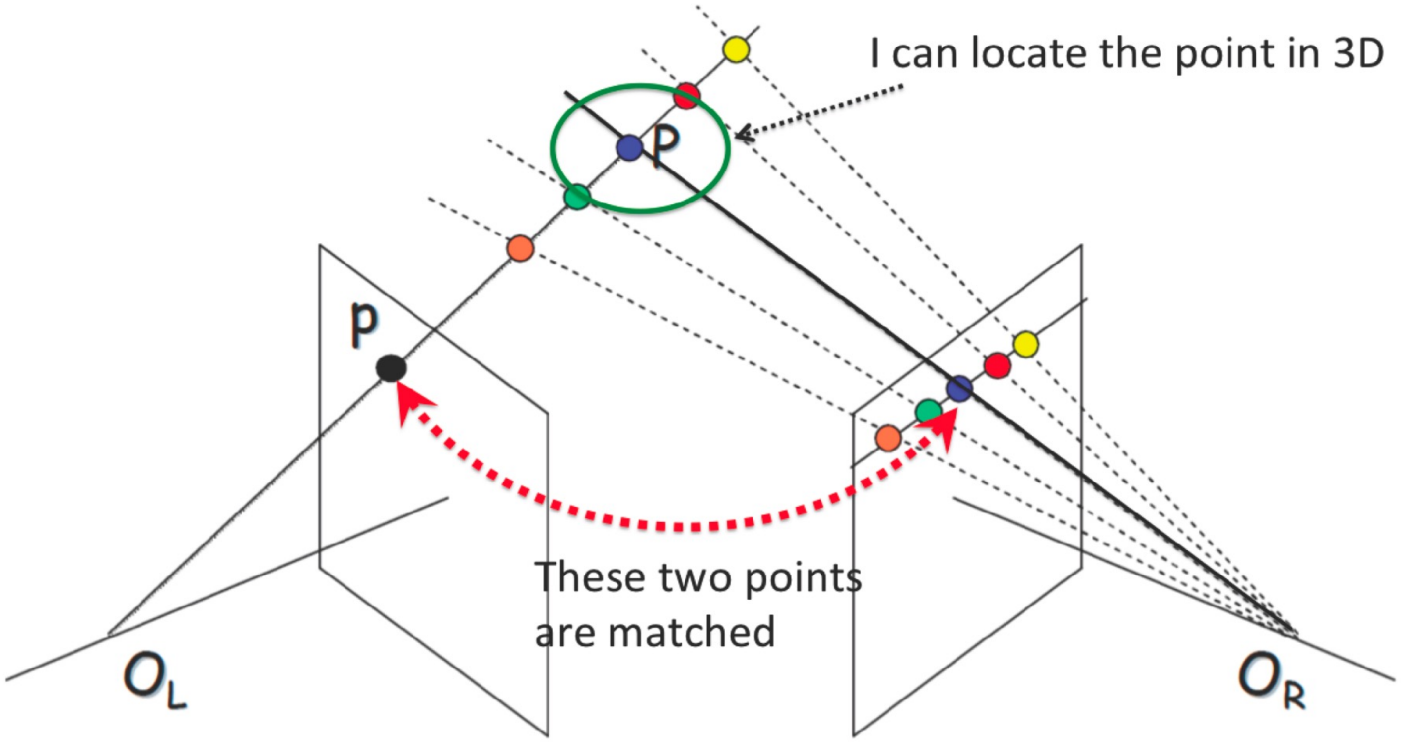
Stereo camera for driverless car

Multi-camera systems for 3D perception

Stereo Vision

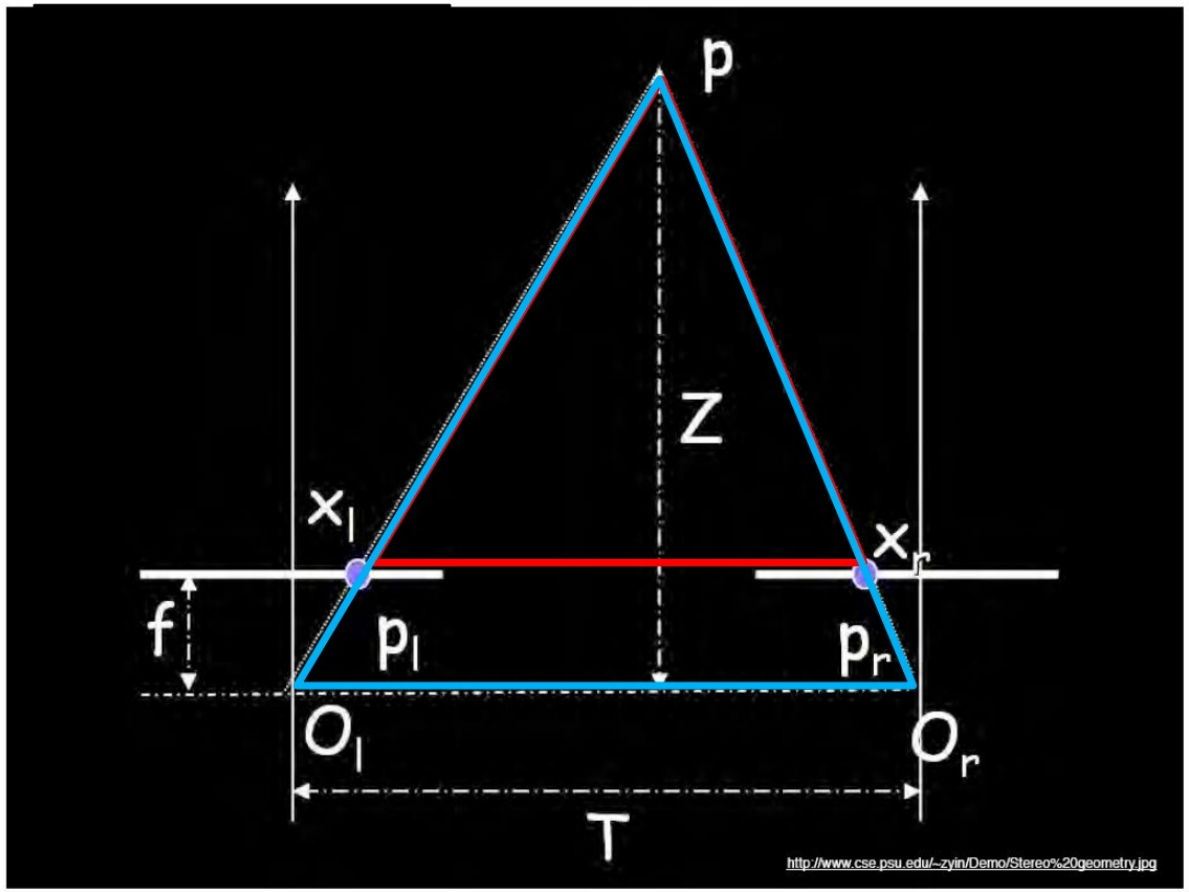


Stereo Vision



Two views provide disparity, from which depth information can be recovered.

Simple Stereo Vision



$$\frac{Z}{T} = \frac{Z - f}{T + x_r - x_l}$$

$$Z = \frac{Tf}{x_r - x_l}$$

Disparity

Stereo Vision

The larger the disparity is, the closer the object is to the camera.

Stereo Vision

Requirements:

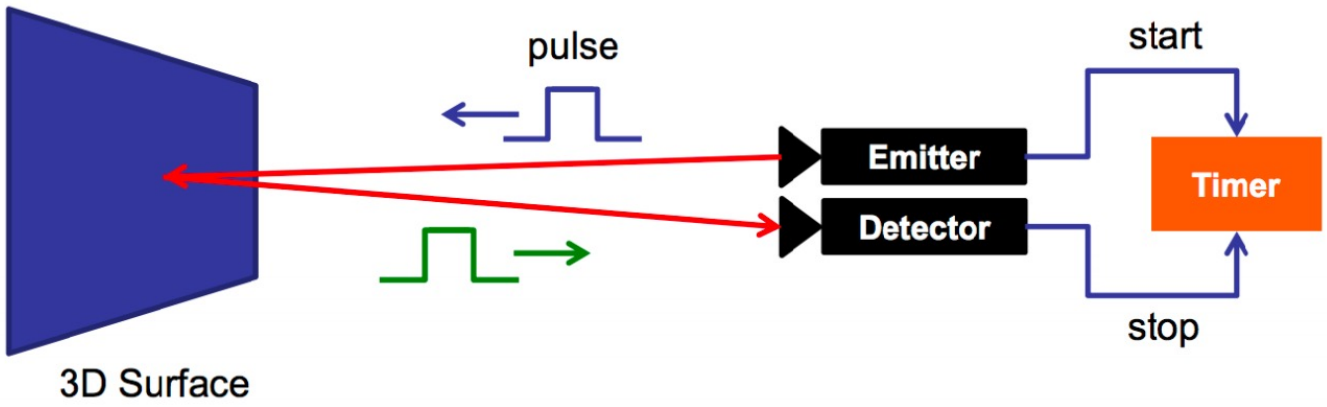
- Calibration for the two cameras.
  - Intrinsic matrices for both cameras (**K**)
  - Baseline distance T in parallel camera case
  - **R**, **t** in non-parallel case
- Corresponding pixels

$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

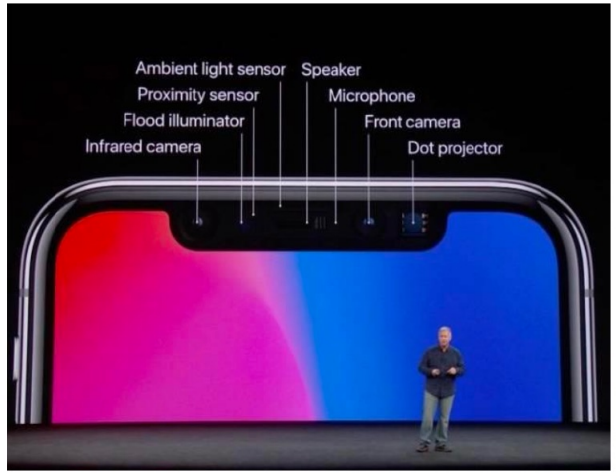
Known Known Unknown

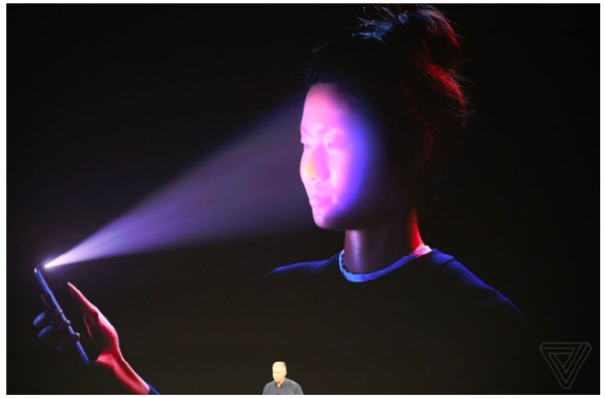
Depth Camera

- Lidar: Sonar for light



Lidar in iPhone X





Depth Camera

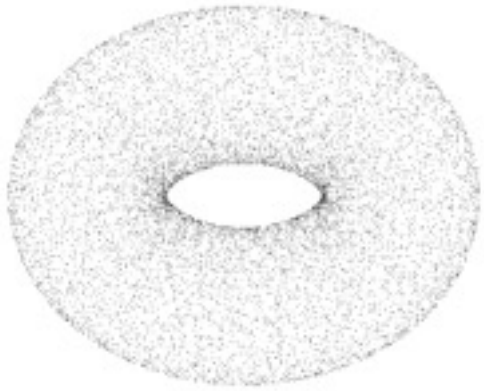
Depth cameras produce depth maps which can be processed into point clouds.

3D Representation

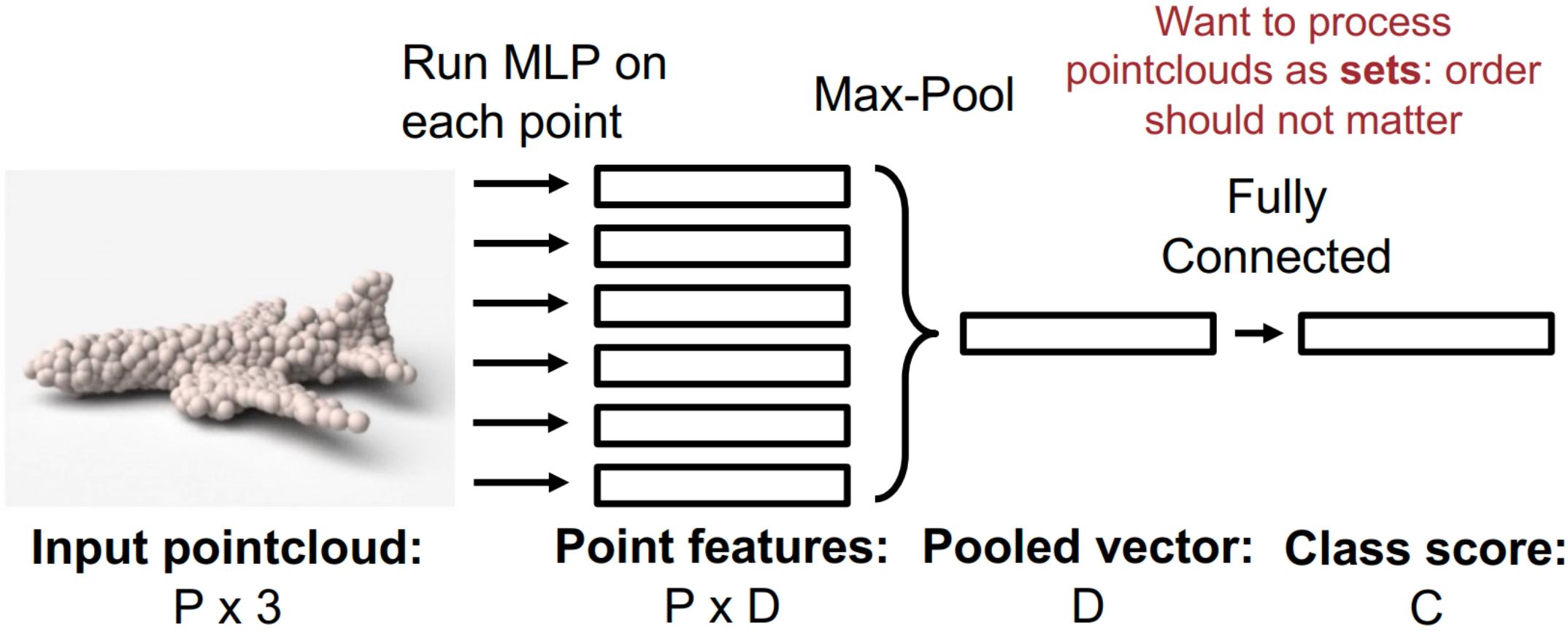
- Point Clouds: a discrete set of data points in space with their Cartesian coordinates (X, Y, Z). Data: $P \times 3$
  - Can represent fine structures without huge numbers of points
  - Doesn't explicitly represent the surface of the shape: rendering or other applications requires post-processing
  - introduce new distance: compare point clouds as sets!

$$d_{CD}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} \|x - y\|_2^2 + \sum_{y \in S_2} \min_{x \in S_1} \|x - y\|_2^2$$

the sum of L2 distance to each point's nearest neighbor in the other set

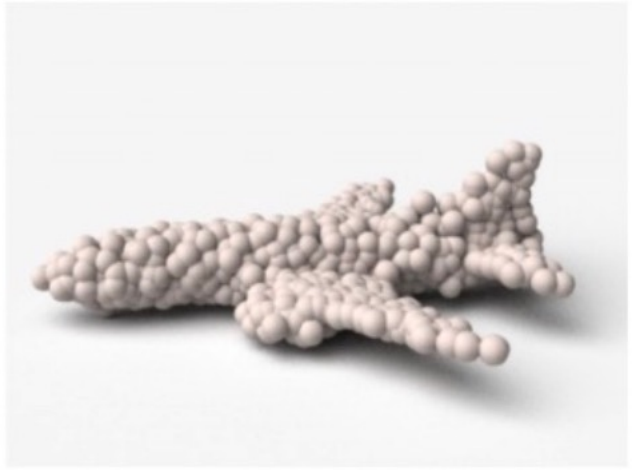


PointNet



Qi et al, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", CVPR 2017
Qi et al, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space", NeurIPS 2017

PointNet Application



PointNet

airplane?

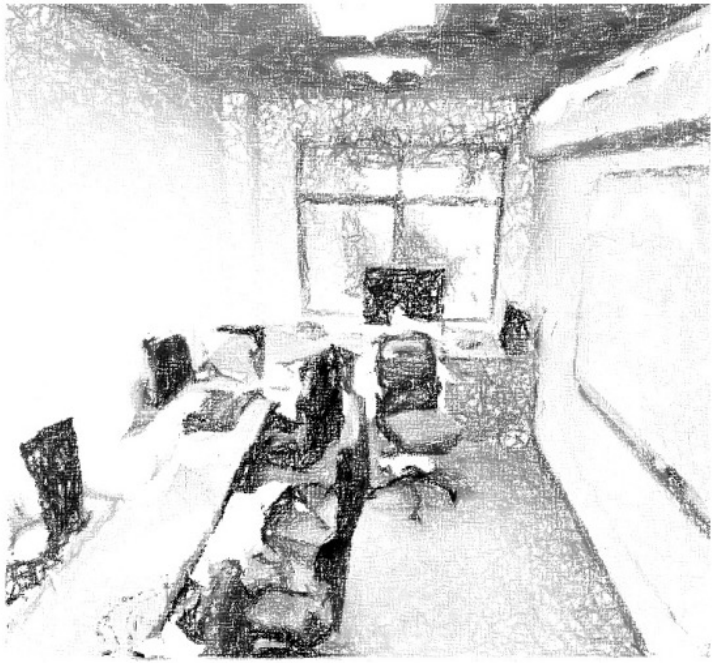
3D Object Classification

mug?

table?

car?

PointNet Application



Point cloud inputs
P x 3

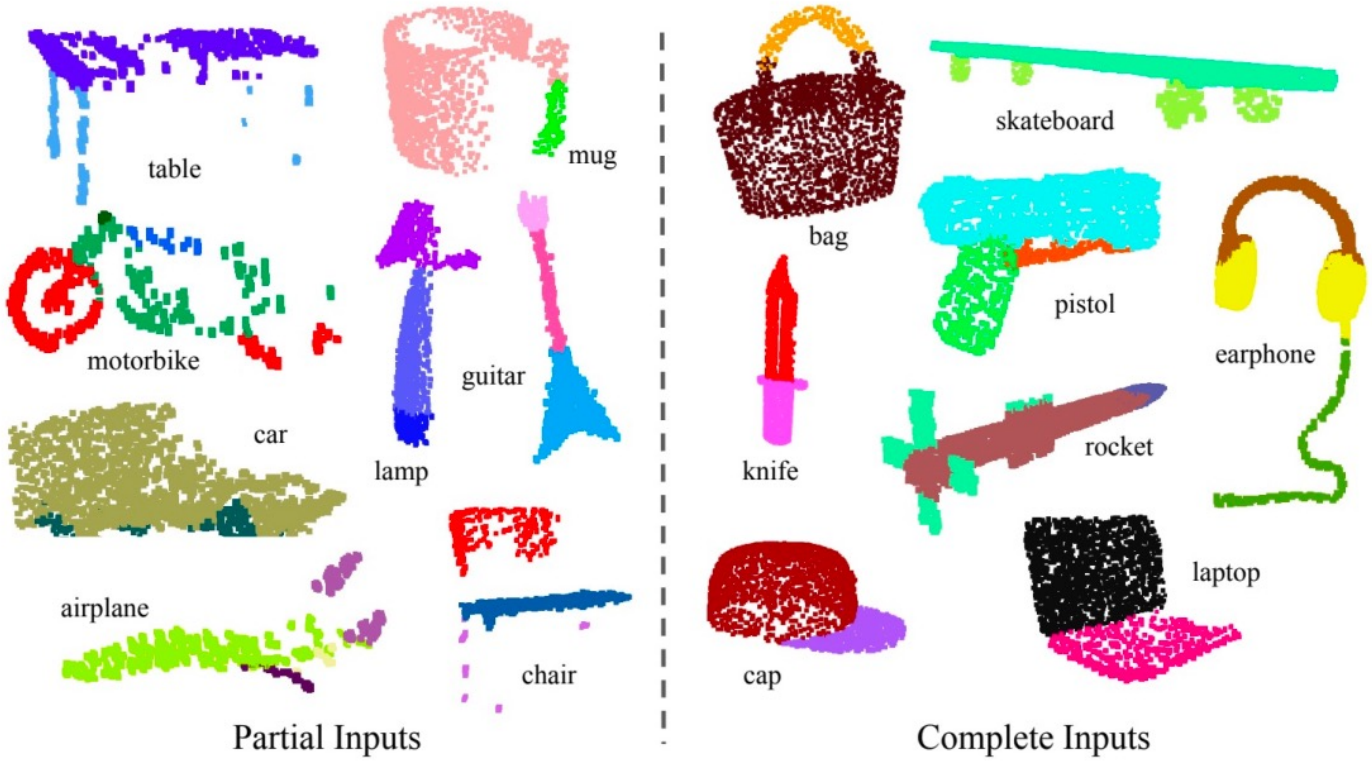
Run MLP on each point

Copy&Concat

Point features:
P x D

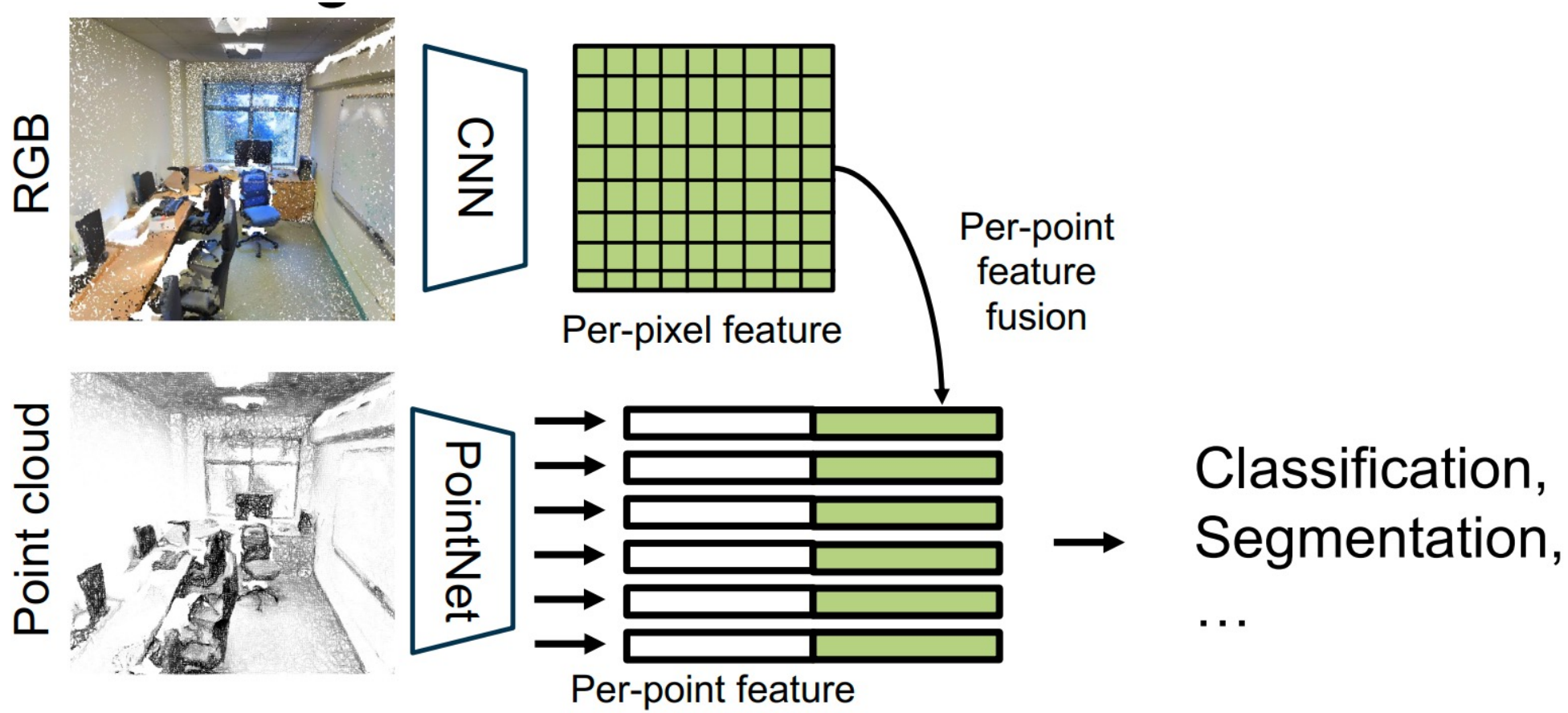
Per-point classification

PointNet Application



Object Part Segmentation

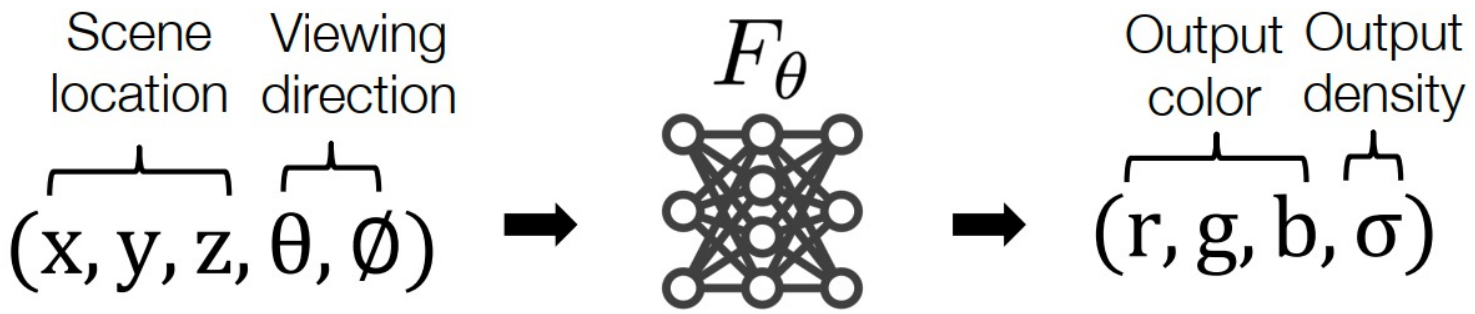
PointCould +RGB

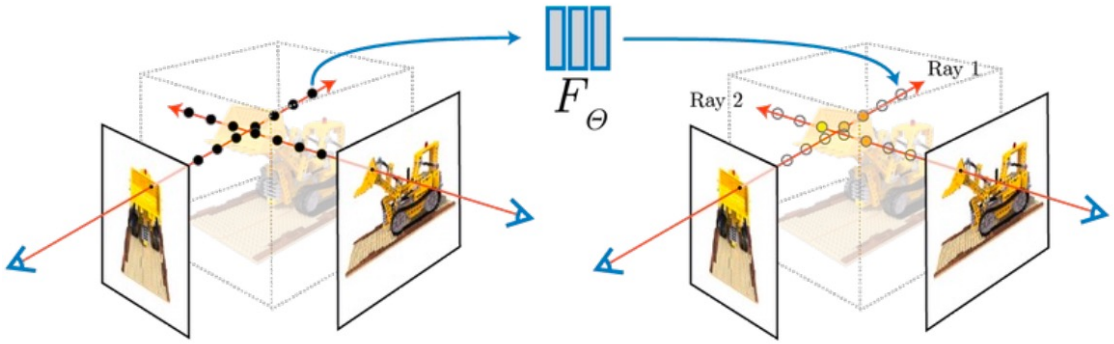


Wang et al, “DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion”, CVPR 2019

3D Representation

- Neural Radiance Fields (NeRF)





Mildenhall et al, “Representing Scenes as Neural Radiance Fields for View Synthesis”, ECCV 2020

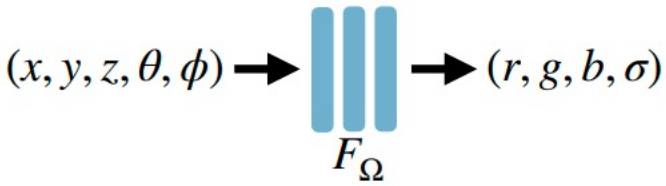
NeRF

Input Images

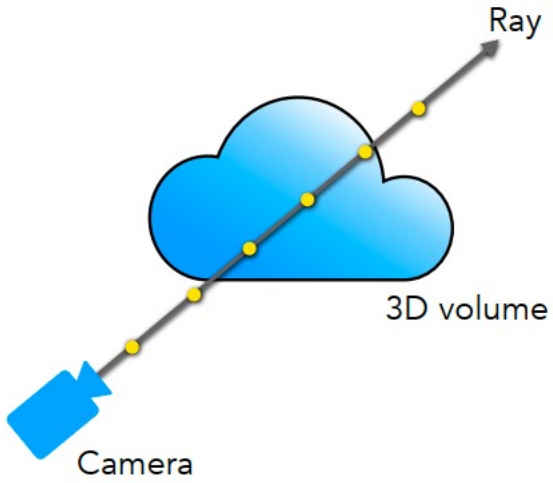
Optimize NeRF

Render new views

Three Components

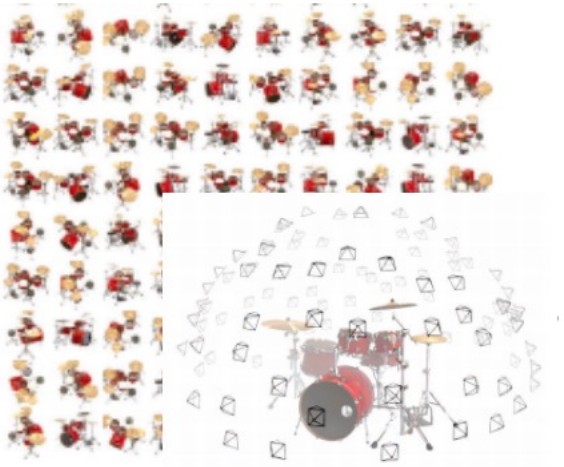


Neural Volumetric 3D Scene Representation



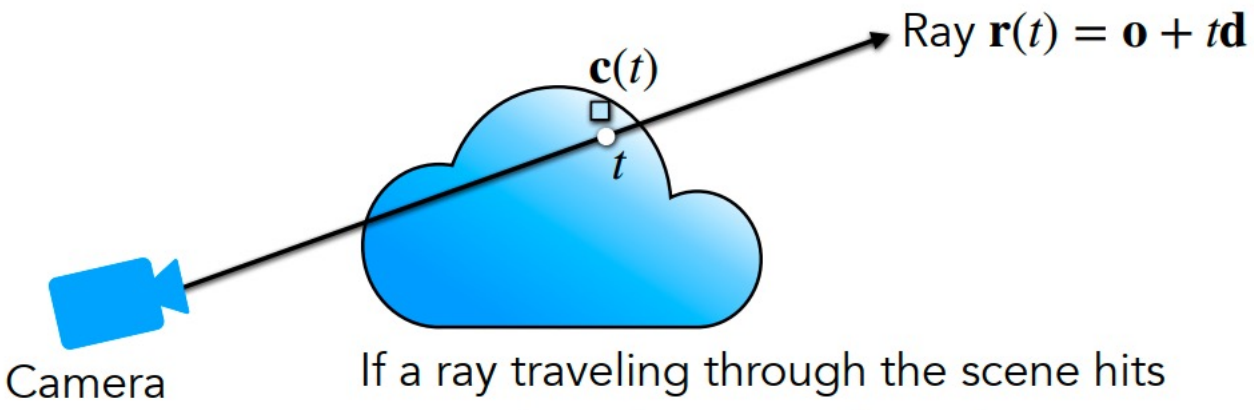
Differentiable Volumetric Rendering Function

Objective: Synthesize all training views



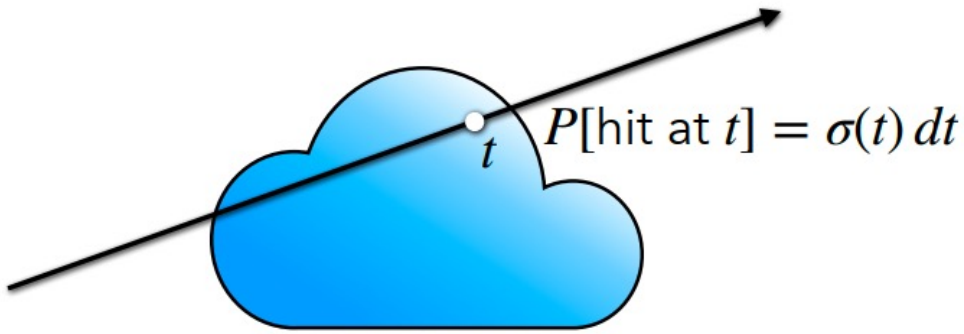
Optimization via Analysis-by-Synthesis

Rendering



If a ray traveling through the scene hits a particle at distance t along the ray, we return its color $\mathbf{c}(t)$

Rendering

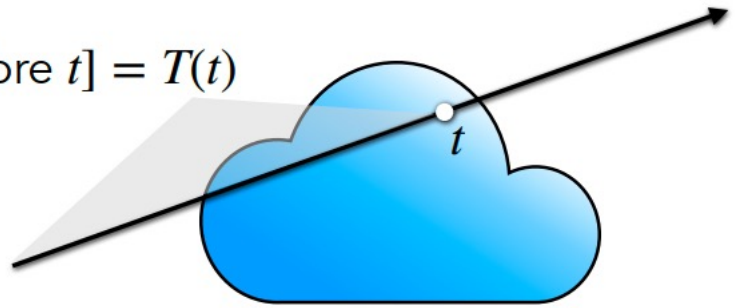


This notion is *probabilistic:* chance that ray hits a particle in a small interval around t is $\sigma(t)\,dt$.

σ is called the "volume density"

Rendering

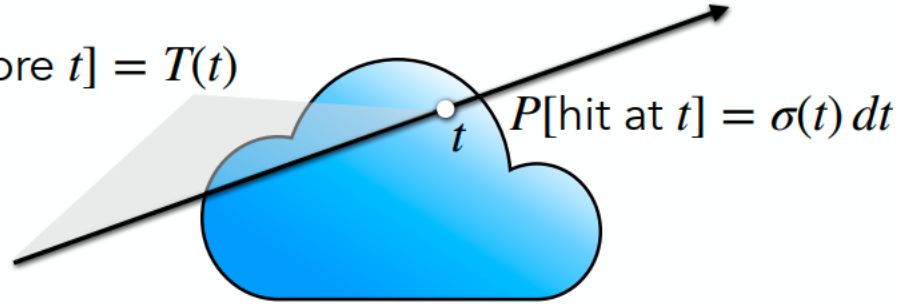
$P[\text{no hits before } t] = T(t)$



To determine if t is the *first* hit along the ray, need to know $T(t)$: the probability that the ray makes it through the volume up to t.
$T(t)$ is called "transmittance"

Rendering

$P[\text{no hits before } t] = T(t)$

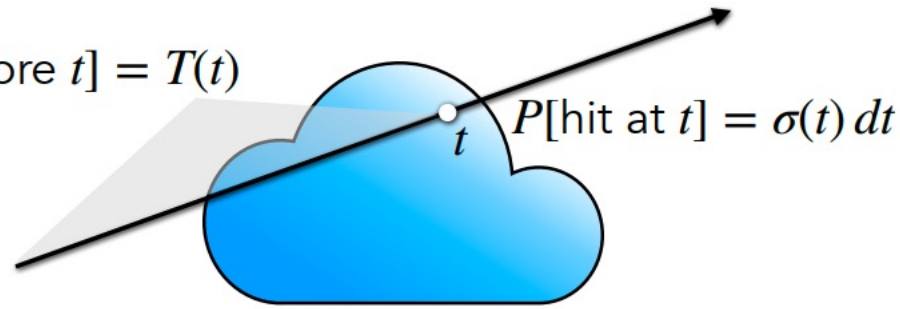


The product of these probabilities tells us how much you see the particles at t:

$P[\text{first hit at } t] = P[\text{no hit before } t] \times P[\text{hit at } t] = T(t)\sigma(t)dt$

Rendering

$P[\text{no hits before } t] = T(t)$



σ and T are related by the probabilistic fact that

$P[\text{no hit before } t + dt] = P[\text{no hit before } t] \times P[\text{no hit at } t]$

$$T(t + dt) = T(t) \times (1 - \sigma(t)dt)$$

Rendering

$$T(t+dt) = T(t)(1-\sigma(t)dt)$$

Taylor expansion for T ⇒ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange ⇒ $\frac{T'(t)}{T(t)}dt = -\sigma(t)dt$

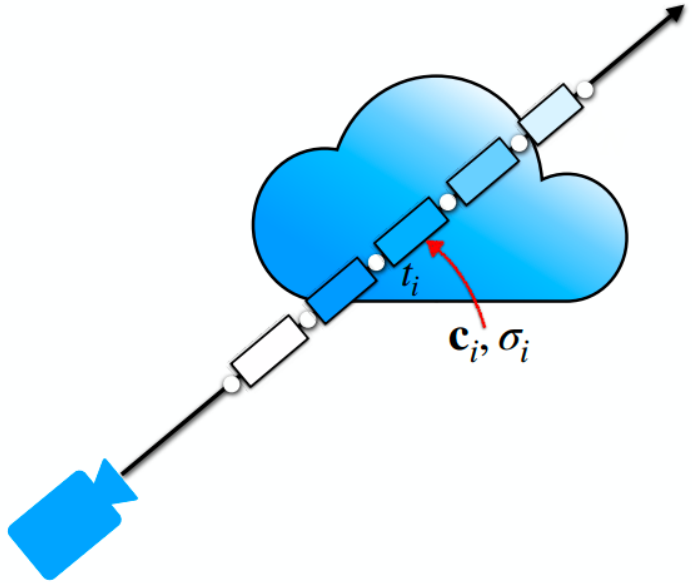
Integrate ⇒ $\log T(t) = -\int_{t_0}^{t} \sigma(s)ds$

Exponentiate ⇒ $T(t) = \exp\left(-\int_{t_0}^{t} \sigma(s)ds\right)$

The expected color returned by the ray will be

$$\int_{t_0}^{t_1} T(t)\sigma(t)\mathbf{c}(t)\,dt$$

Approximation



We assume volume density and color are roughly constant within each interval

$$\int T(t)\sigma(t)\mathbf{c}(t)\,dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i\,dt$$

$$\text{For } t \in [t_i, t_{i+1}],\ T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i\,ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i\,ds\right)$$

$$\exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right) = T_i \qquad \exp\left(-\sigma_i(t - t_i)\right)$$

$$= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp\left(-\sigma_i(t - t_i)\right)\,dt \quad = \sum_{i=1}^{n} T_i\mathbf{c}_i\left(1 - \exp(-\sigma_i\delta_i)\right)$$

Rendering

Rendering model for ray r(t) = o + td:

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

weights, colors

t = point along ray
C = Color of Pixel
c = color of point

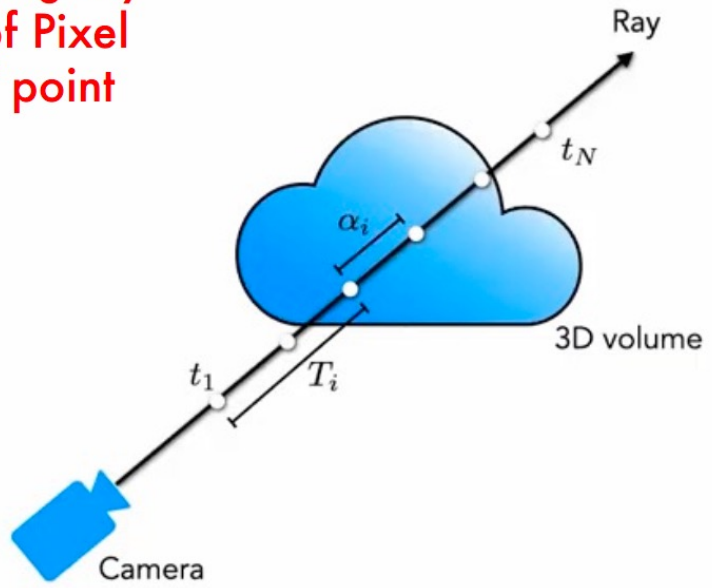
How much light is blocked earlier along ray:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$ Transparency

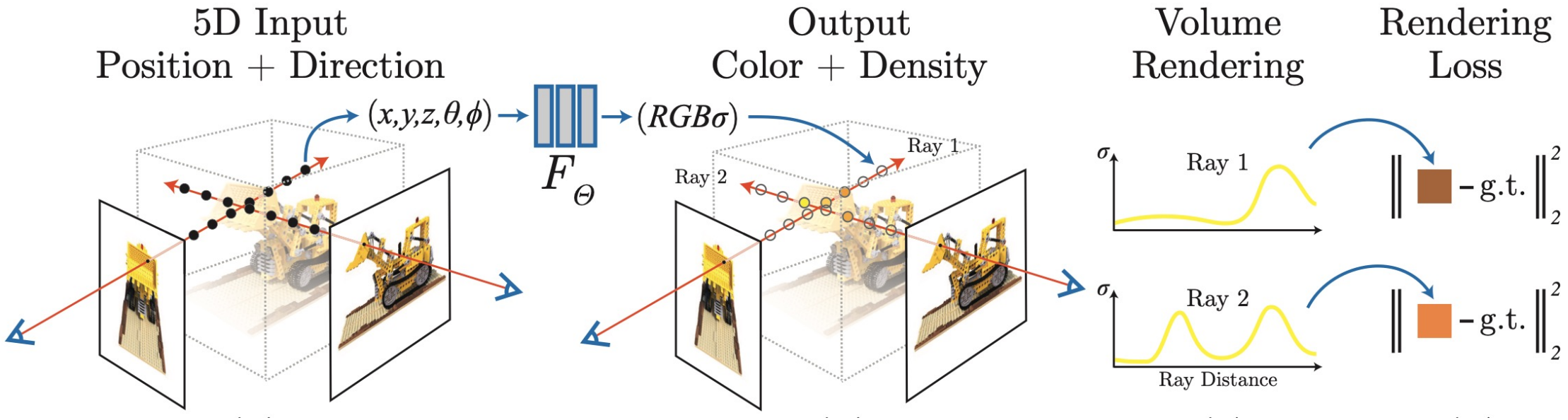
How much light is contributed by ray segment *i*:

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

Function of segment length δt_i and volume density σ

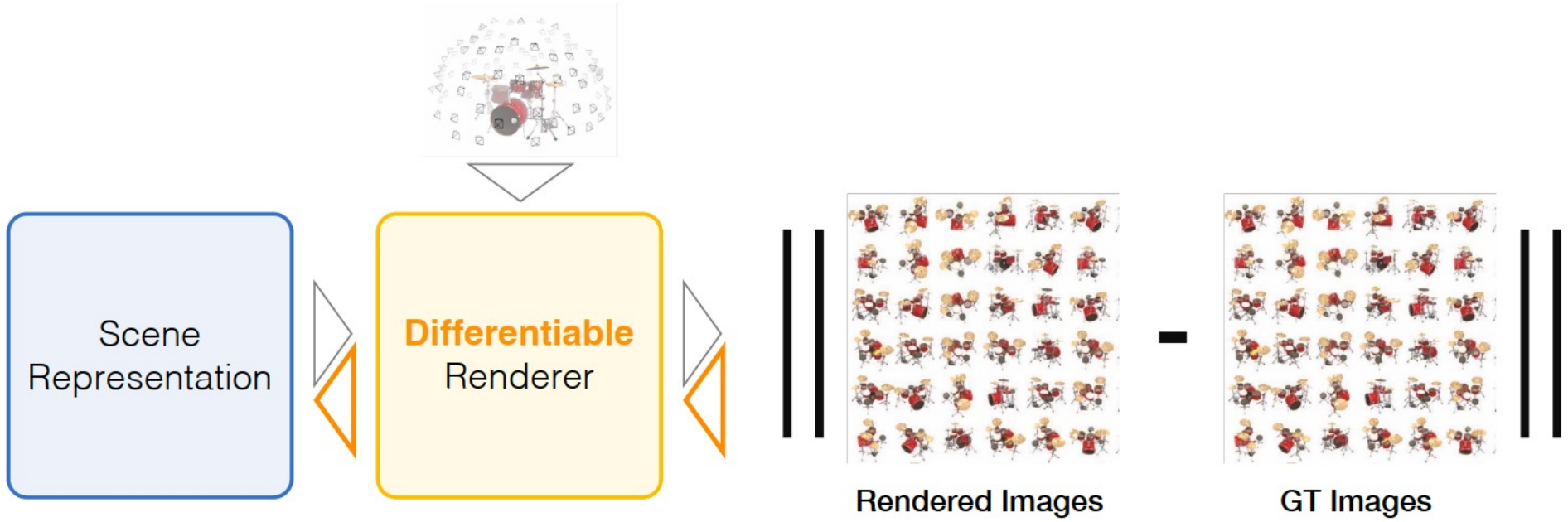


Training NeRF



- Volume rendering is naturally differentiable
- Using gradient descent to optimize F_θ

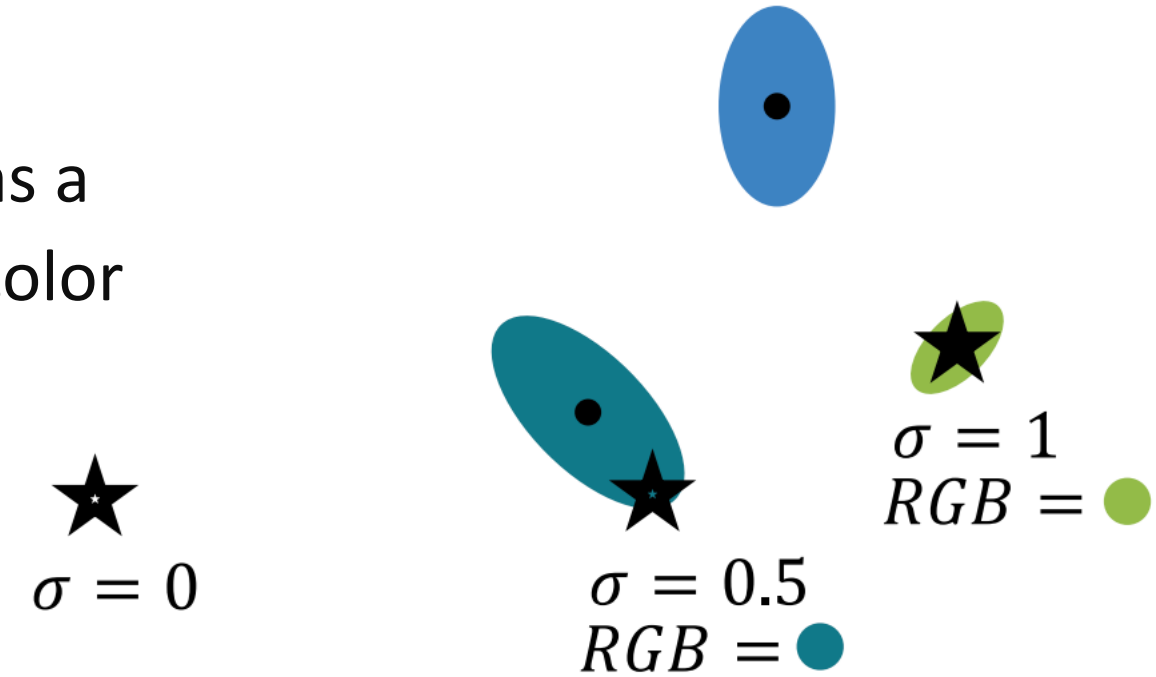
Differentiable Rendering



Similar pipeline for training Nerf and Gaussian splatting

Gaussian Splatting

- Gaussian splatting represents a 3D scene using a set of 3D Gaussians.
- Each Gaussian can be thought of as a soft “blob”, defined by its shape, color and transparency.



Kerbl et al. “3D Gaussian Splatting for Real-Time Radiance Field Rendering "

Rendering

- Rendering along a ray

$$\int_{t_0}^{t_1} T(t)\sigma(t)\mathbf{c}(t)\,dt \qquad T(t) = \exp\left(-\int_{t_0}^{t}\sigma(s)ds\right)$$

σ(t): volume density, $\boldsymbol{c}(t)$: color

- Now consider the volume density representation of sparse 3D Gaussian

$$\sigma(\boldsymbol{x},\xi) = \sum_k w_k\, r_k'(\boldsymbol{x},\xi)$$

$\boldsymbol{x} = (x_1, x_2)$ is the first two coordinates. Next, we consider rendering along the third axis.

Zwicker et al. “EWA Splatting"

Rendering

- Rendering along the third axis:

$$I(\mathbf{x}) = \int_0^L c(\mathbf{x},\xi) \sum_{k\in\mathbb{N}} w_k r'_k(\mathbf{x},\xi) e^{-\int_0^\xi \sum_{j\in\mathbb{N}} w_j r'_j(\mathbf{x},\mu)\mathrm{d}\mu} \mathrm{d}\xi$$

$$= \sum_{k\in\mathbb{N}} w_k \left(\int_0^L c(\mathbf{x},\xi)\, r'_k(\mathbf{x},\xi)\, e^{-\sum_{j\in\mathbb{N}} w_j \int_0^\xi r'_j(\mathbf{x},\mu)\mathrm{d}\mu} \mathrm{d}\xi \right)$$

$$= \sum_{k\in\mathbb{N}} w_k \left(\int_0^L c(\mathbf{x},\xi)\, r'_k(\mathbf{x},\xi) \prod_j e^{-w_j \int_0^\xi r'_j(\mathbf{x},\mu)\mathrm{d}\mu} \mathrm{d}\xi \right)$$

approximation: $e^{-x} \approx 1 - x$ and the color is constant inside each Gaussian ellipsoid

$$= \sum_{k\in\mathbb{N}} w_k c(\mathbf{x}) \left(\int_0^L r'_k(\mathbf{x},\xi) \prod_j \left(1 - w_j \int_0^\xi r'_j(\mathbf{x},\mu)\mathrm{d}\mu \right) \mathrm{d}\xi \right)$$

Rendering

- More assumptions to simplify the computation
  - The Gaussian ellipsoids are truncated with local support (one standard deviation) that does not overlap

$$\int_0^{\xi} r'_j(\mathbf{x},\mu)\mathrm{d}\mu \approx \int_0^{L} r'_j(\mathbf{x},\mu)$$

  - The Gaussian ellipsoids are ordered front to back
  - Ignore the transparency variation inside each Gaussian ellipsoid
- Thus,

$$\sum_{k\in\mathbb{N}} w_k c(\mathbf{x}) \left(\int_0^{L} r'_k(\mathbf{x},\xi) \prod_j \left(1 - w_j \int_0^{\xi} r'_j(\mathbf{x},\mu)\mathrm{d}\mu \right) \mathrm{d}\xi \right)$$

$$= \sum_{k\in\mathbb{N}} w_k c(\mathbf{x}) \left(\int_0^{L} r'_k(\mathbf{x},\xi) \prod_j \left(1 - w_j \int_0^{L} r'_j(\mathbf{x},\mu)\mathrm{d}\mu \right) \mathrm{d}\xi \right) = \sum_{k\in\mathbb{N}} w_k c(\mathbf{x}) \left(\int_0^{L} r'_k(\mathbf{x},\xi)\,\mathrm{d}\xi \right) \prod_{j<k} \left(1 - w_j \int_0^{L} r'_j(\mathbf{x},\mu)\mathrm{d}\mu \right)$$

Rendering

- Finally, we have the rendering formula for sparse Gaussian representation

$$I(\mathbf{x}) = \sum_{k \in \mathbb{N}} w_k c(\mathbf{x}) q_k(\mathbf{x}) \prod_{j=0}^{k-1} (1 - w_j q_j(\mathbf{x}))$$

$$q_k(\mathbf{x}) = \int_{\mathbb{R}} r'_k(\mathbf{x}, \xi) \, \mathrm{d}\xi$$

- Recall that $r'_k(\boldsymbol{x}, \xi)$ is the 3D Gaussian ellipsoid. $q_k(\boldsymbol{x})$ is the projection of $r'_k(\boldsymbol{x}, \xi)$ along the third axis.

Projection of 3D Gaussian

- nD Gaussian

$$\mathcal{G}^{(n)}_{\mathbf{p},\mathbf{\Sigma}}(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mathbf{p})^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{p})}$$

- Consider Invertible affine transform $\varphi(\mathbf{x}) = \mathbf{M}\mathbf{x} + \mathbf{b}$

$$\begin{aligned}
\mathcal{G}^{(n)}_{\mathbf{p},\mathbf{\Sigma}}(\mathbf{x}) &= \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\mathbf{M}^{-1}(\varphi(\mathbf{x})-\mathbf{b})-\mathbf{p}\right)^T\mathbf{\Sigma}^{-1}\left(\mathbf{M}^{-1}(\varphi(\mathbf{x})-\mathbf{b})-\mathbf{p}\right)} \\
&= \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left(\mathbf{M}^{-1}\varphi(\mathbf{x})-\mathbf{M}^{-1}\mathbf{b}-\mathbf{p}\right)^T\mathbf{\Sigma}^{-1}\left(\mathbf{M}^{-1}\varphi(\mathbf{x})-\mathbf{M}^{-1}\mathbf{b}-\mathbf{p}\right)} \\
&= \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}\left[\mathbf{M}^{-1}(\varphi(\mathbf{x})-\mathbf{b}-\mathbf{M}\mathbf{p})\right]^T\mathbf{\Sigma}^{-1}\left[\mathbf{M}^{-1}(\varphi(\mathbf{x})-\mathbf{b}-\mathbf{M}\mathbf{p})\right]} \\
&= \frac{|\mathbf{M}|}{(2\pi)^{\frac{n}{2}}|\mathbf{M}|^{\frac{1}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}|\mathbf{M}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\varphi(\mathbf{x})-\varphi(\mathbf{p}))^T\left(\mathbf{M}^{-1}\right)^T\mathbf{\Sigma}^{-1}\mathbf{M}^{-1}(\varphi(\mathbf{x})-\varphi(\mathbf{p}))} \\
&= |\mathbf{M}|\,\mathcal{G}^{(n)}_{\varphi(\mathbf{p}),\mathbf{M}\mathbf{\Sigma}\mathbf{M}^T}(\varphi(\mathbf{x}))
\end{aligned}$$

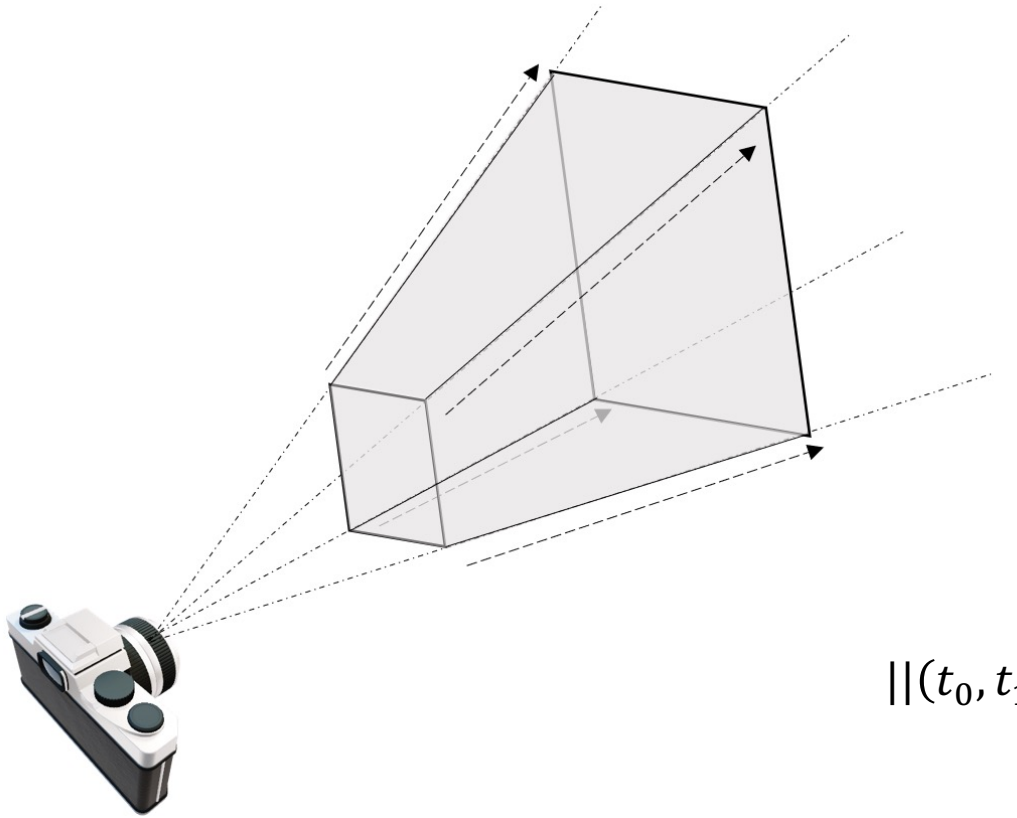
- Integrate along one axis

$$\int_{\mathbb{R}} \mathcal{G}^{(3)}_{\mathbf{p},\mathbf{\Sigma}}(\mathbf{x}, x_2)\,\mathrm{d}x_2 = \mathcal{G}^{(2)}_{\bar{\mathbf{p}},\bar{\mathbf{\Sigma}}}(\mathbf{x})$$

$$\bar{p} = p[1{:}\,2],$$
$$\bar{\Sigma} = \Sigma[1{:}\,2,1{:}\,2]$$

Frustum

- For pin-hole camera, the ray are not parallel



A non-affine projective transformation

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \phi(\mathbf{t}) = \begin{pmatrix} t_0/t_2 \\ t_1/t_2 \\ \|(t_0, t_1, t_2)^T\| \end{pmatrix}$$

$||(t_0, t_1, t_2)^T||$ can be used to order the Gaussian ellipsoids!

Projection of Gaussian Ellipsoid

- The mapping ϕ is no longer affine. The projection of 3D Gaussian ellipsoid under ϕ is no longer 2D Gaussian.
- Local affine approximation: 1st order Taylor expansion

$$\phi_k(\mathbf{t}) = \phi_k(\mathbf{t}_k) + \mathbf{J}_k(\mathbf{t} - \mathbf{t}_k)$$

$\boldsymbol{t}_k$: center of a Gaussian

$$\mathbf{J}_k = \begin{pmatrix} \frac{\partial x_0}{\partial t_0} & \frac{\partial x_0}{\partial t_1} & \frac{\partial x_0}{\partial t_2} \\ \frac{\partial x_1}{\partial t_0} & \frac{\partial x_1}{\partial t_1} & \frac{\partial x_1}{\partial t_2} \\ \frac{\partial x_2}{\partial t_0} & \frac{\partial x_2}{\partial t_1} & \frac{\partial x_2}{\partial t_2} \end{pmatrix} = \begin{pmatrix} 1/t_{k,2} & 0 & -t_{k,0}/t_{k,2}^2 \\ 0 & 1/t_{k,2} & -t_{k,1}/t_{k,2}^2 \\ \frac{t_{k,0}}{\|\mathbf{t}_k\|} & \frac{t_{k,1}}{\|\mathbf{t}_k\|} & \frac{t_{k,2}}{\|\mathbf{t}_k\|} \end{pmatrix}$$

- Together with the camera pose
- Covariance matrix

$$\begin{aligned} \mathbf{t} &= \varphi(\mathbf{u}) = \mathbf{M}\mathbf{u} + \mathbf{b} \\ \mathbf{x} &= \phi_k(\mathbf{t}) = \mathbf{x}_k + \mathbf{J}_k(\mathbf{t} - \mathbf{t}_k) \\ &= \mathbf{x}_k + \mathbf{J}_k(\mathbf{M}\mathbf{u} + \mathbf{b} - \mathbf{t}_k) \\ &= \mathbf{J}_k\mathbf{M}\mathbf{u} + \mathbf{x}_k + \mathbf{J}_k(\mathbf{b} - \mathbf{t}_k) \end{aligned}$$

Projection of Gaussian Ellipsoid

- Together with the camera extrinsic calibration

$$\begin{aligned}\mathbf{t} &= \varphi(\mathbf{u}) = \mathbf{M}\mathbf{u} + \mathbf{b} \\ \mathbf{x} &= \phi_k(\mathbf{t}) = \mathbf{x}_k + \mathbf{J}_k(\mathbf{t} - \mathbf{t}_k) \\ &= \mathbf{x}_k + \mathbf{J}_k(\mathbf{M}\mathbf{u} + \mathbf{b} - \mathbf{t}_k) \\ &= \mathbf{J}_k\mathbf{M}\mathbf{u} + \mathbf{x}_k + \mathbf{J}_k(\mathbf{b} - \mathbf{t}_k)\end{aligned}$$

- Covariance matrix after transformation

$$\mathbf{\Sigma}' = \mathbf{J}_k\mathbf{M}\mathbf{\Sigma}\mathbf{M}^T\mathbf{J}_k^T$$

- 2D Covariance matrix: ignore the third axis

Optimization of Covariance Matrix

- Covariance matrix: positive definite. Reparameterization

$$\boldsymbol{\Sigma} = \mathbf{RSS}^T\mathbf{R}^T$$

R: rotation, **S**: diagonal matrix

- Rotation matrix representation by unit quaternion

$$R(q) = 2\begin{pmatrix} \frac{1}{2} - (q_j^2 + q_k^2) & (q_i q_j - q_r q_k) & (q_i q_k + q_r q_j) \\ (q_i q_j + q_r q_k) & \frac{1}{2} - (q_i^2 + q_k^2) & (q_j q_k - q_r q_i) \\ (q_i q_k - q_r q_j) & (q_j q_k + q_r q_i) & \frac{1}{2} - (q_i^2 + q_j^2) \end{pmatrix}$$

- Gradient calculation

$$\boldsymbol{M = RS}$$

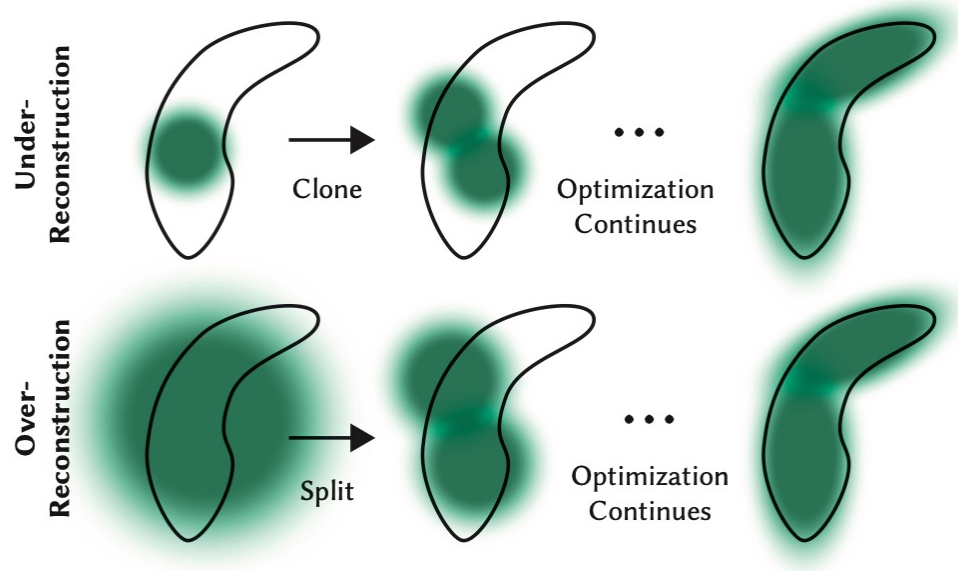
$$\frac{\partial M}{\partial q_r} = 2\begin{pmatrix} 0 & -s_y q_k & s_z q_j \\ s_x q_k & 0 & -s_z q_i \\ -s_x q_j & s_y q_i & 0 \end{pmatrix}, \quad \frac{\partial M}{\partial q_i} = 2\begin{pmatrix} 0 & s_y q_j & s_z q_k \\ s_x q_j & -2s_y q_i & -s_z q_r \\ s_x q_k & s_y q_r & -2s_z q_i \end{pmatrix}$$

$$\frac{\partial M}{\partial q_j} = 2\begin{pmatrix} -2s_x q_j & s_y q_i & s_z q_r \\ s_x q_i & 0 & s_z q_k \\ -s_x q_r & s_y q_k & -2s_z q_j \end{pmatrix}, \quad \frac{\partial M}{\partial q_k} = 2\begin{pmatrix} -2s_x q_k & -s_y q_r & s_z q_i \\ s_x q_r & -2s_y q_k & s_z q_j \\ s_x q_i & s_y q_j & 0 \end{pmatrix}$$

More about training

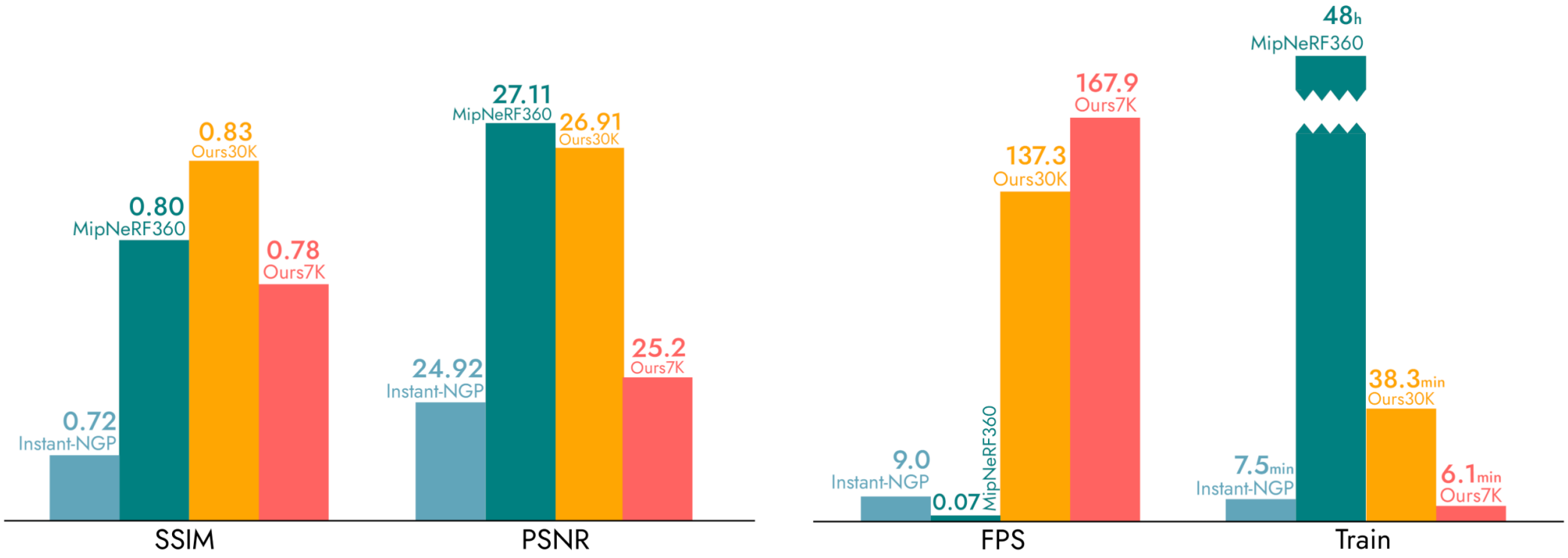
- Initialize the 3D Gaussian using point cloud data

- Adaptive density control



Novel-view Synthesis

Evaluation



But more parameters of Gaussian splatting: 350 - 700MB (3-6m of Gaussians), vs. INGP: 15 - 50MB and MipNeRF360: 8.6MB

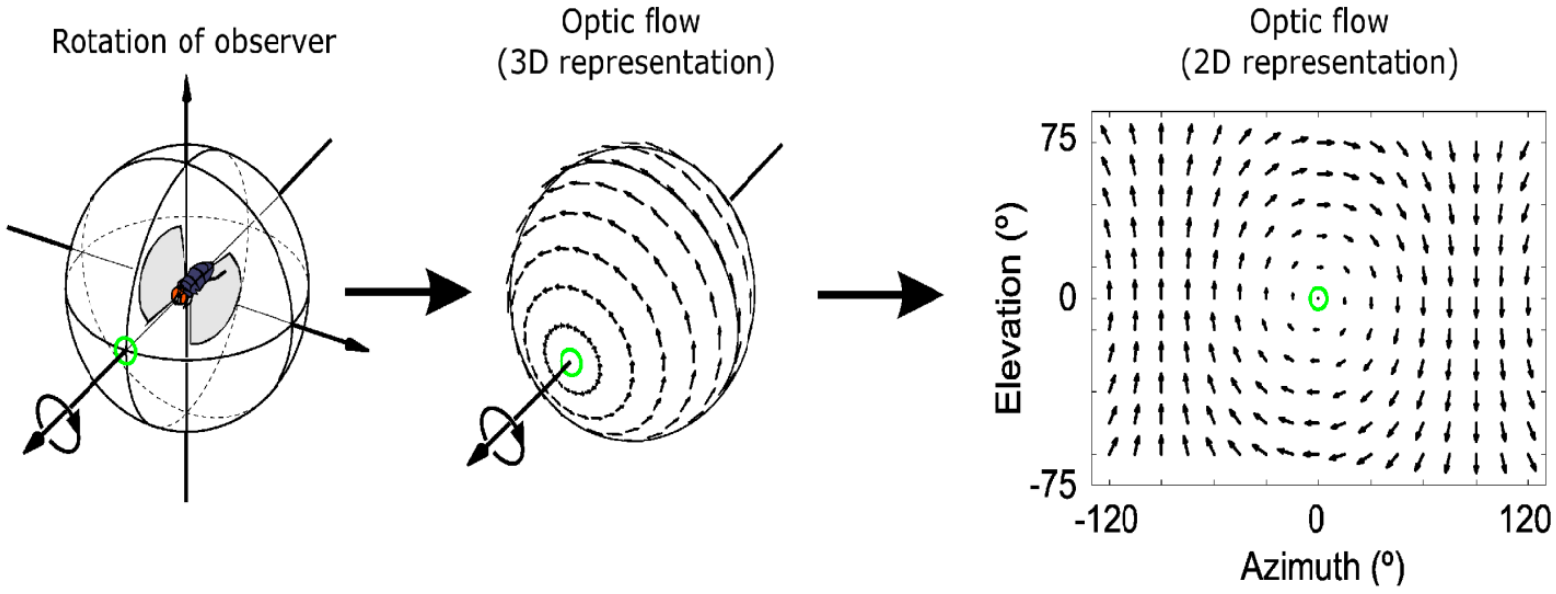
Motion Estimation

The slides are adapted from 6.8300/6.8301: Advances in Computer Vision by Charles Herrmann.

Motion is important

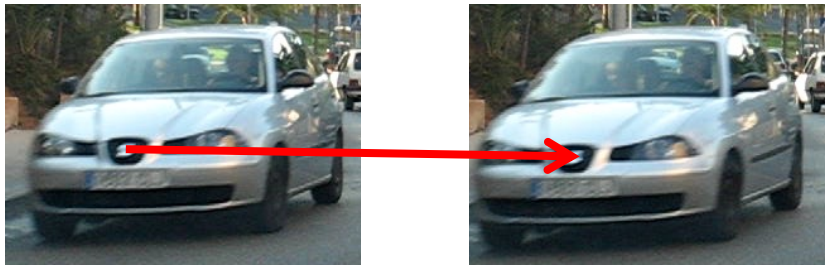
Optical Flow

- Optical flow is the vector field $\mathrm{v}(x, y, t)$ specifying the current velocity of the pixel at position (x, y) and at time stamp t.



from wiki

Fundamental assumption: Brightness constancy



$$I_t(\mathbf{p}) \approx I_{t+1}(\mathbf{p} + \mathbf{w_p})$$

Matching-based Motion Estimation

- Similarity between a pixel in image 1 with pixels in image 2

$$\min_{\mathbf{w_p}} \left(I_t(\mathbf{p}) - I_{t+1}(\mathbf{p} + \mathbf{w_p}) \right)^2$$

Image 1

Image 2

Matching-based Motion Estimation

- Similarity between a pixel in image 1 with pixels in image 2

$$\min_{\mathbf{w_p}} \left(I_t(\mathbf{p}) - I_{t+1}(\mathbf{p} + \mathbf{w_p}) \right)^2$$

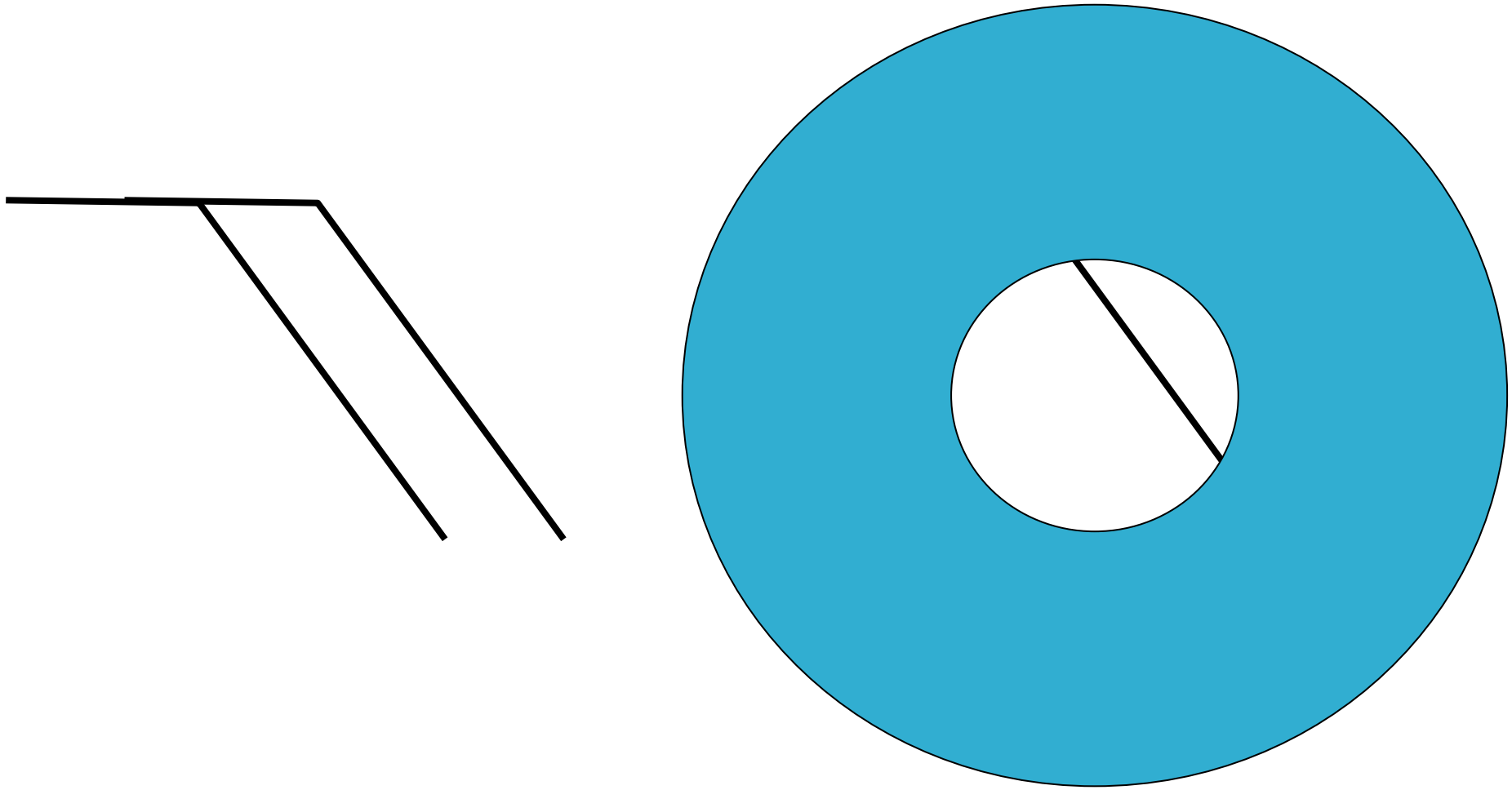
- Approximation:

$$I_{t+1}(\mathbf{p} + \mathbf{w_p}) \approx I_{t+1}(\mathbf{p}) + \nabla I_{t+1}\,(\boldsymbol{p})^T \mathbf{w_p}$$

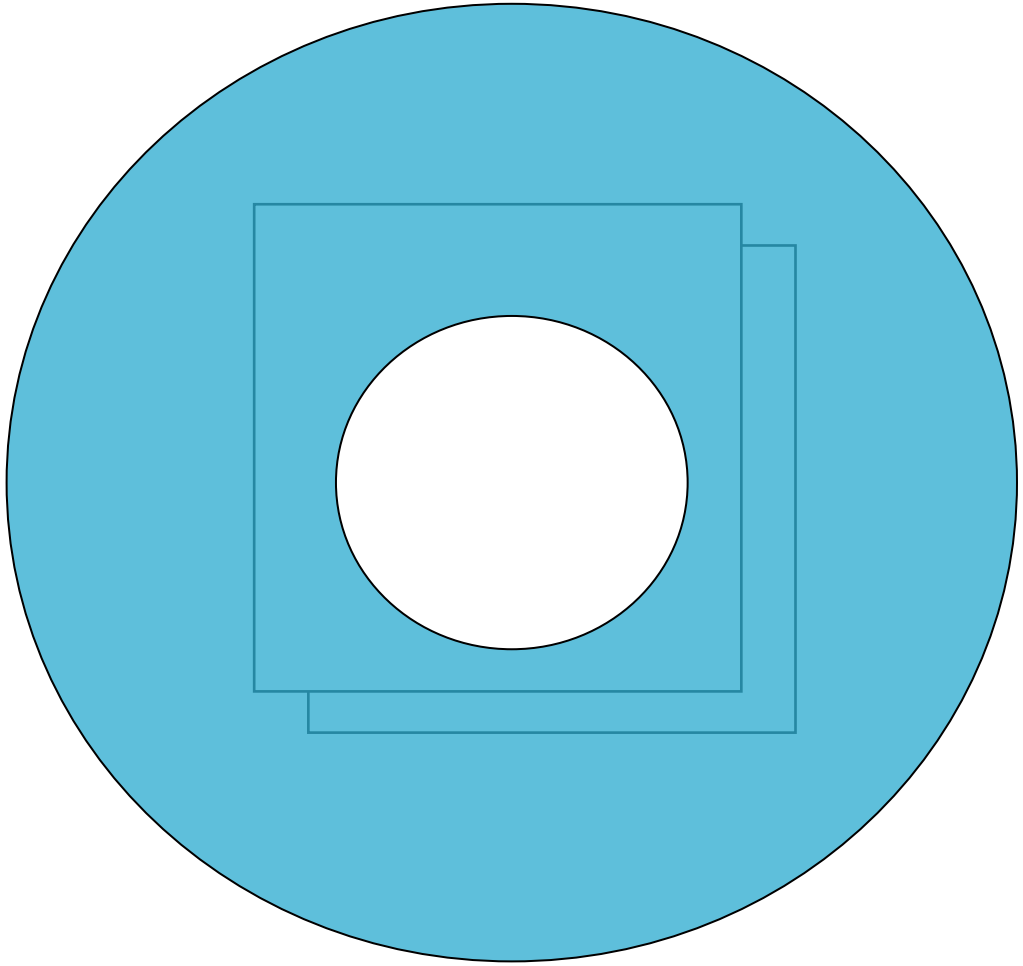
Solve

$$\min_{\mathbf{w_p}} \left(I_t(\mathbf{p}) - I_{t+1}(\mathbf{p}) - \nabla I_{t+1}\,(\boldsymbol{p})^T \mathbf{w_p} \right)^2$$

Aperture problem



More Ambiguities



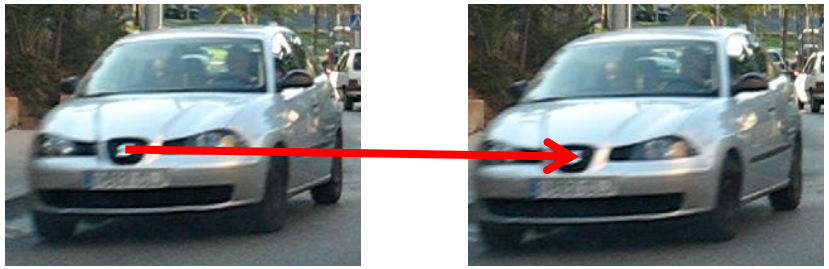
Solving Ambiguities

- For each pixel, The brightness consistency constraint only gives 1 equation, which is under-determined for estimating its optical flow.
- **Lucas-Kanade flow** with local consistency assumption : flow field is locally the same

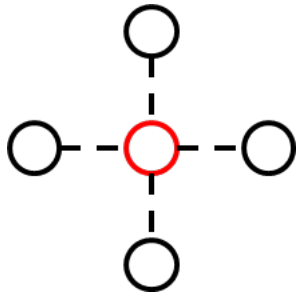
$$\min_{\mathbf{w_p}} \sum_{\mathbf{q}\in \boldsymbol{N_p}} \left(I_t(\mathbf{q}) - I_{t+1}\left(\mathbf{q} + \mathbf{w_p}\right)\right)^2$$

Energy Minimization

- Smooth prior



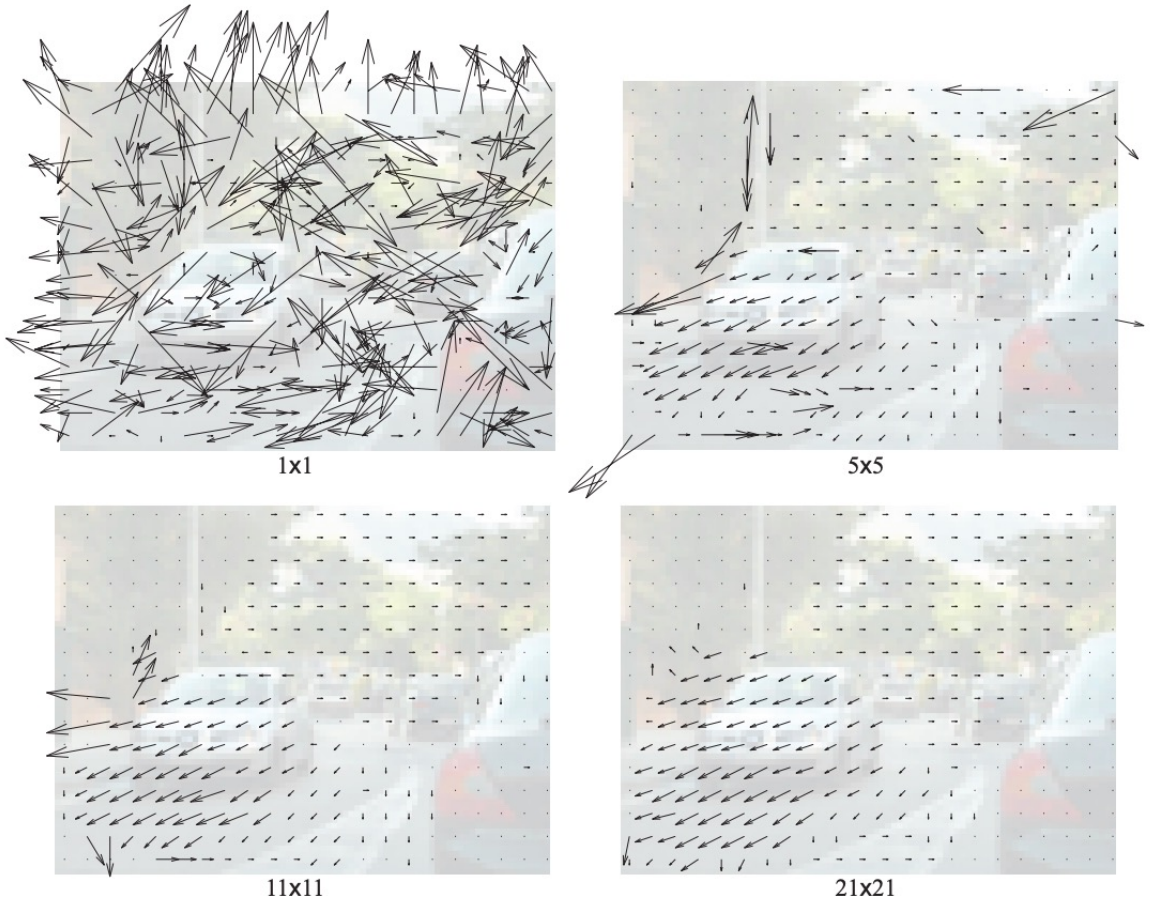
Data term (Constancy)



Prior term (Smoothness)

$$E(\mathbf{w}) = \sum_{\mathbf{p}} \left| I_t(\mathbf{p}) - I_{t+1}(\mathbf{p} + \mathbf{w_p}) \right|^2 + \lambda \sum_{\mathbf{q} \in N_\mathbf{p}} \left| \mathbf{w_p} - \mathbf{w_q} \right|^2$$

Effect of Batch size



Computational Concern

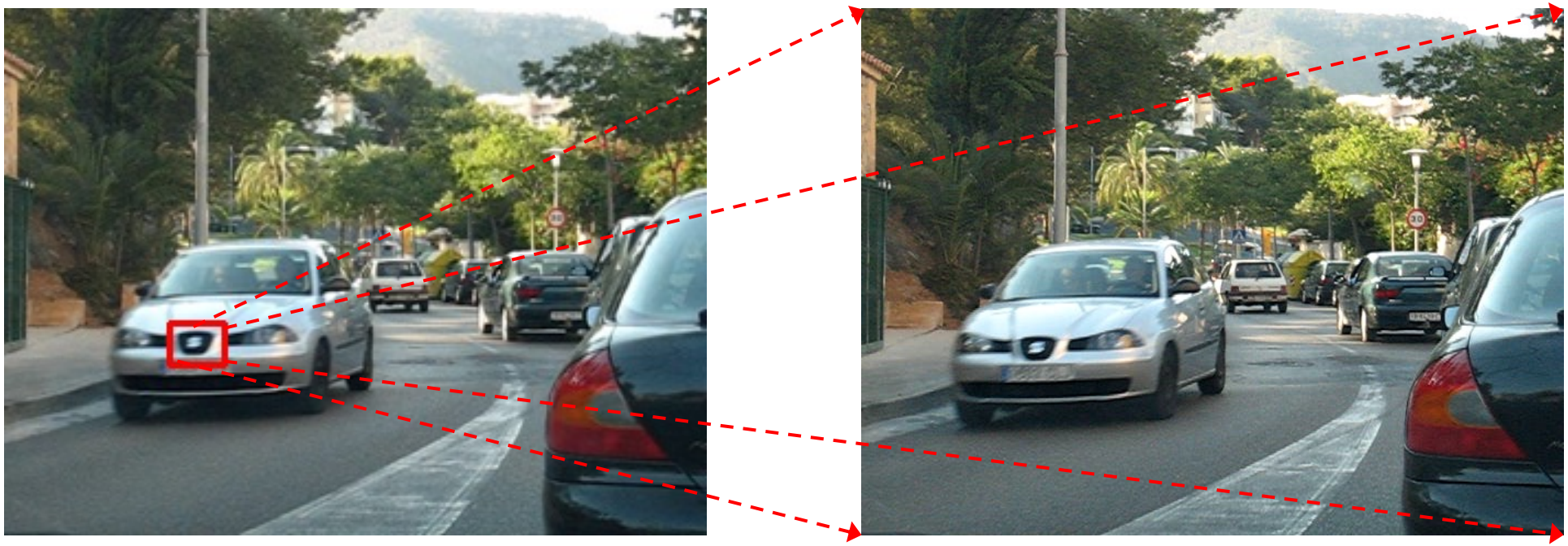


Image 1

Image 2

Searching in the whole range for patch matching is computationally expansive!

Approximation

- If the motion is small, we can use 1st order Taylor expansion for approximation, and then solve a least square problem instead:

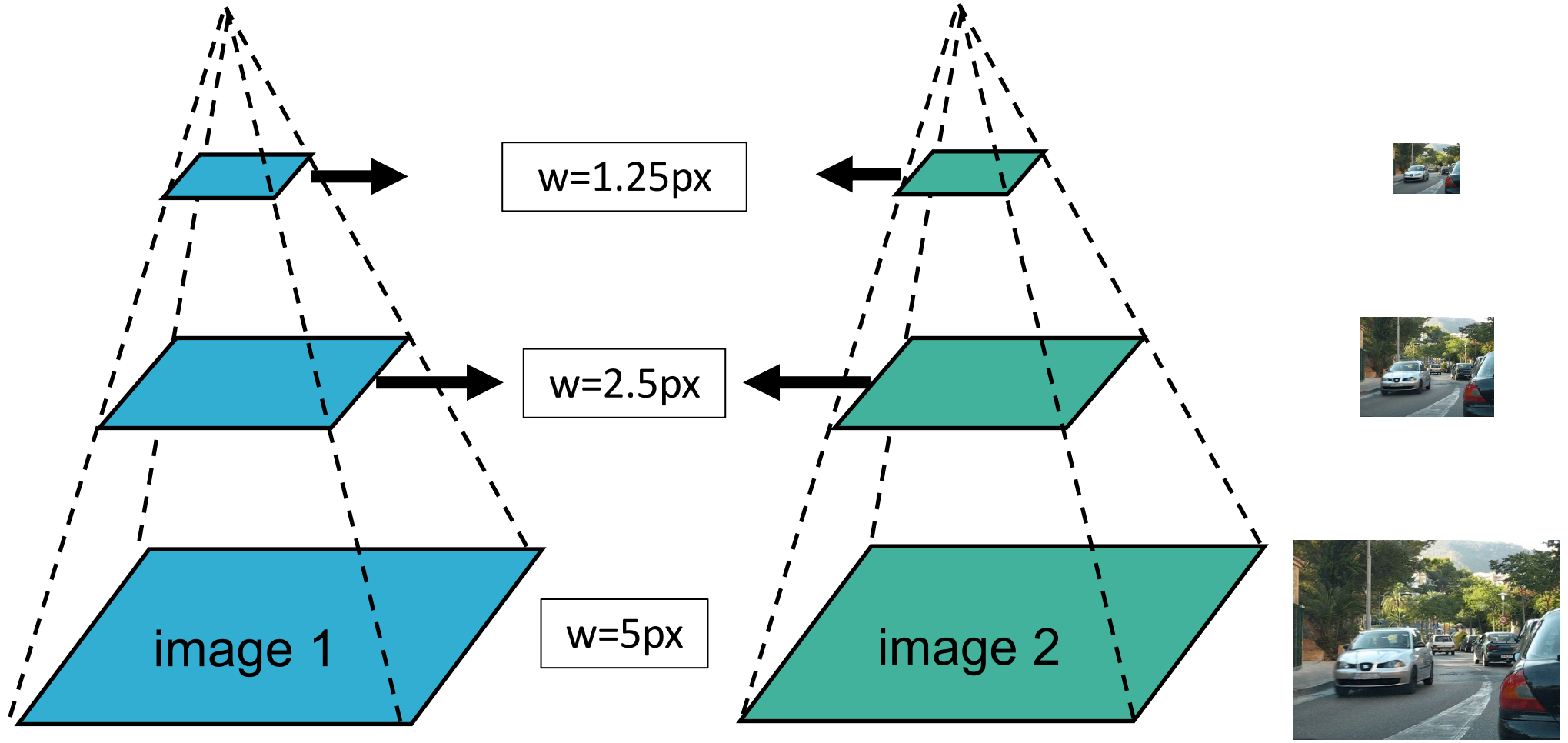
$$\min_{\mathbf{w_p}} \sum_{\mathbf{q}\in N_p} \left(I_t(\mathbf{q}) - I_{t+1}(\boldsymbol{q}) - \nabla I_{t+1}\,(\boldsymbol{q})^T \mathbf{w_p}\right)^2$$

or

$$\min_{\mathbf{w_p}} E(\mathbf{w}) = \sum_{\mathbf{p}} \left|I_t(\mathbf{p}) - I_{t+1}(\mathbf{p}) - \nabla I_{t+1}\,(\boldsymbol{p})^T \mathbf{w_p}\right|^2 + \lambda \sum_{\mathbf{q}\in N_\mathbf{p}} \left|\mathbf{w_p} - \mathbf{w_q}\right|^2$$

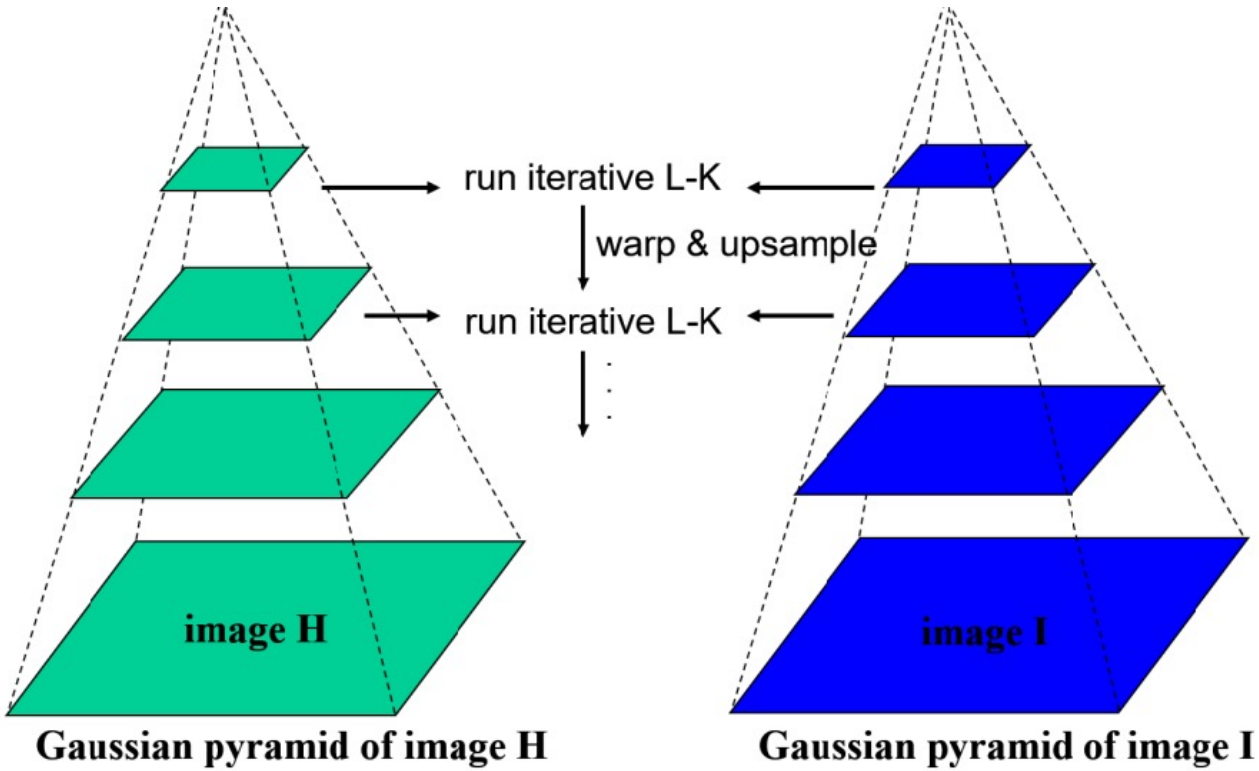
How to handle large motion?

Coarse to Fine Estimation



Smaller motion for coarser scale

Coarse to Fine Estimation



At each level of the pyramid,
1. Estimate velocity at each pixel by solving Lucas-Kanade equations
2. Warp H towards I using the estimated flow field (image warping)
3. Repeat until convergence

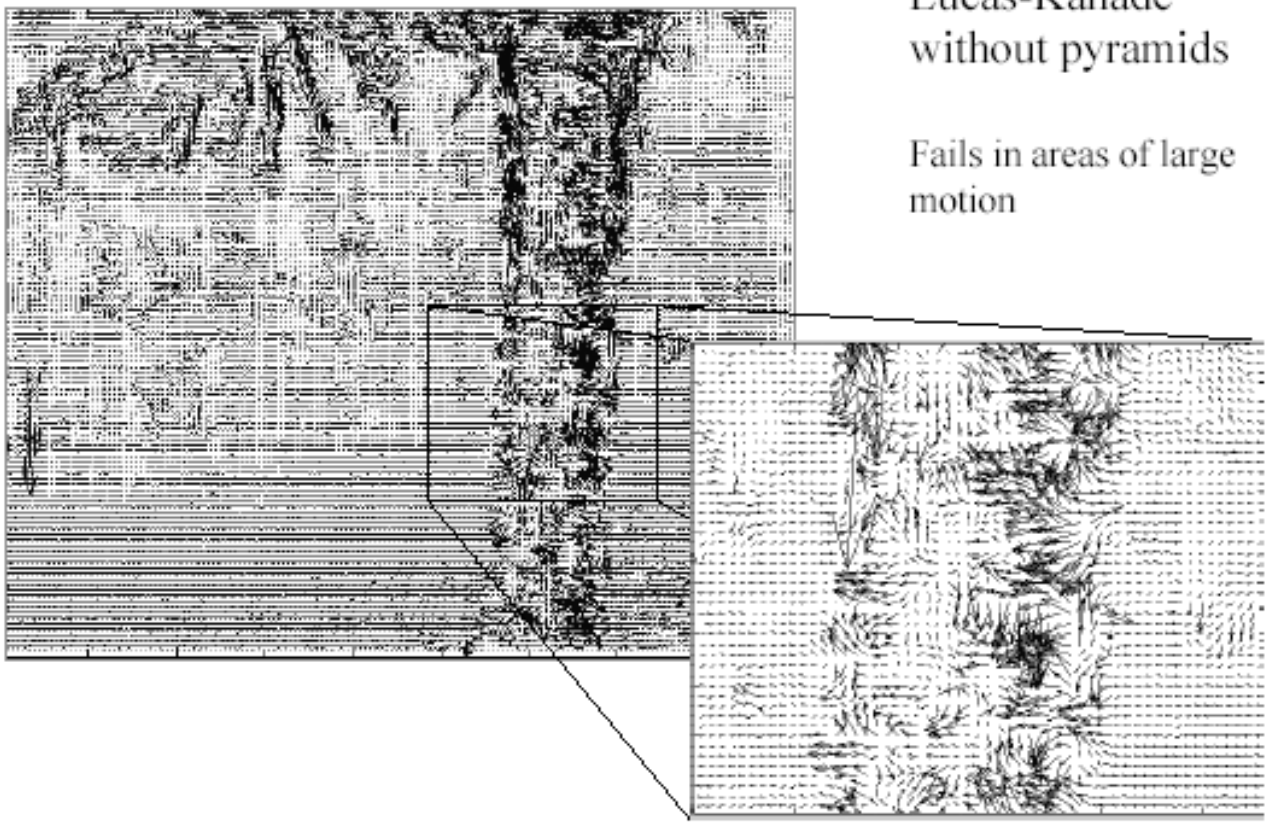
Current estimate

Small increment

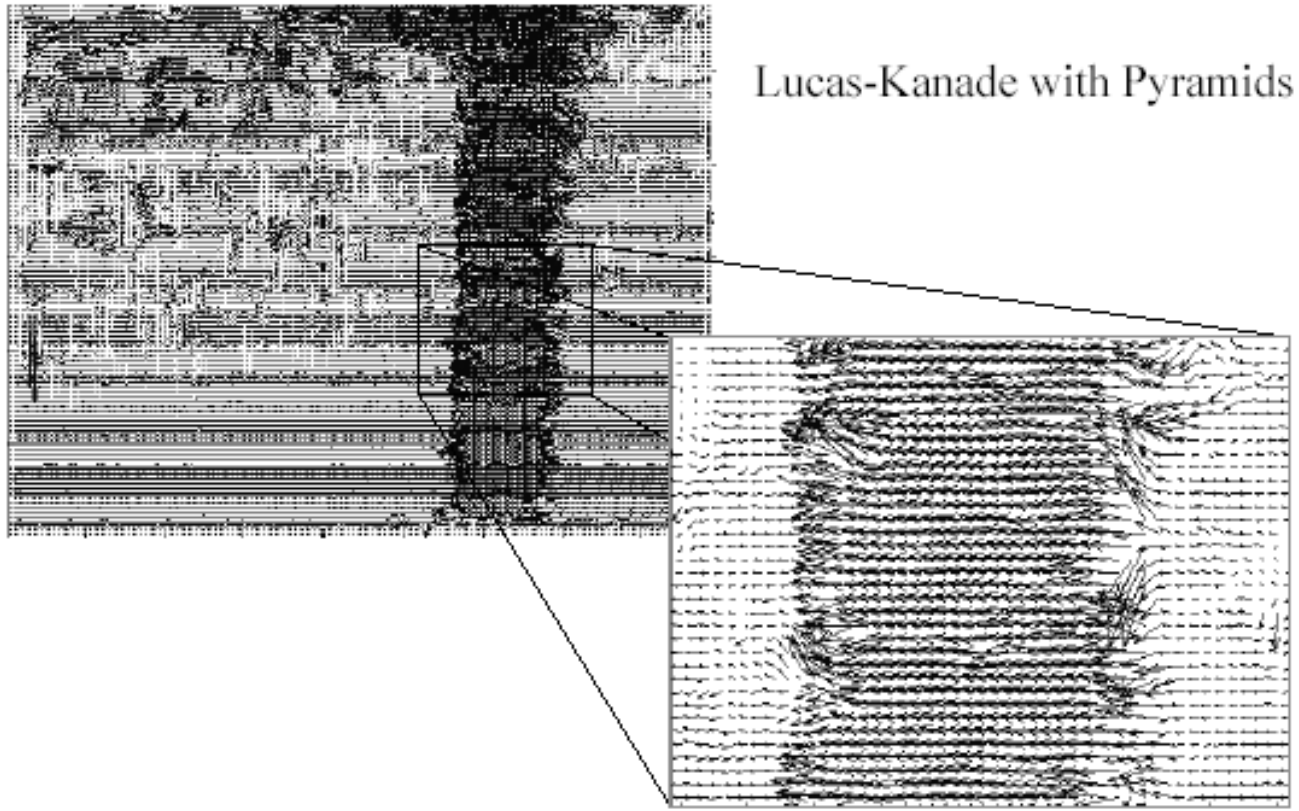
Warped image

$$I(\mathbf{q}+\mathbf{w_p}) = I(\mathbf{q}+\mathbf{w_p^k}+\boldsymbol{\delta}\mathbf{w}_p) = \mathbf{I_w}(\mathbf{q}+\boldsymbol{\delta}\mathbf{w}_p)$$

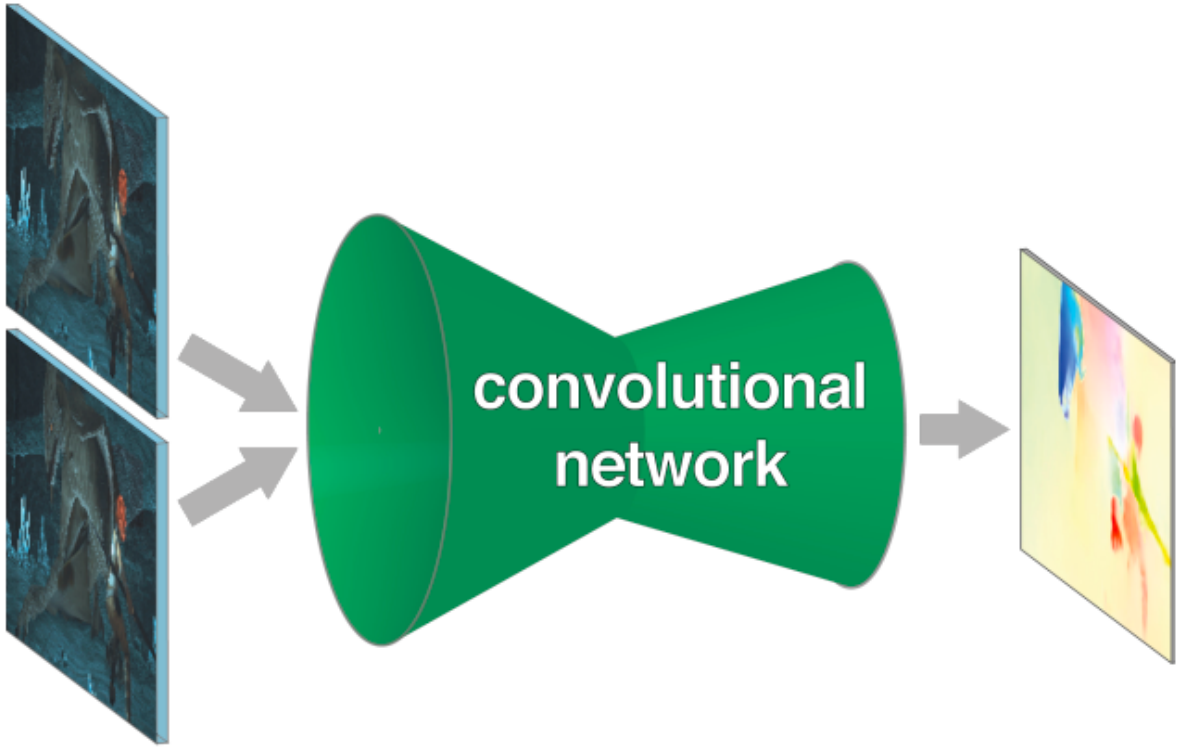
Comparison


Lucas-Kanade
without pyramids
Fails in areas of large
motion

Comparison



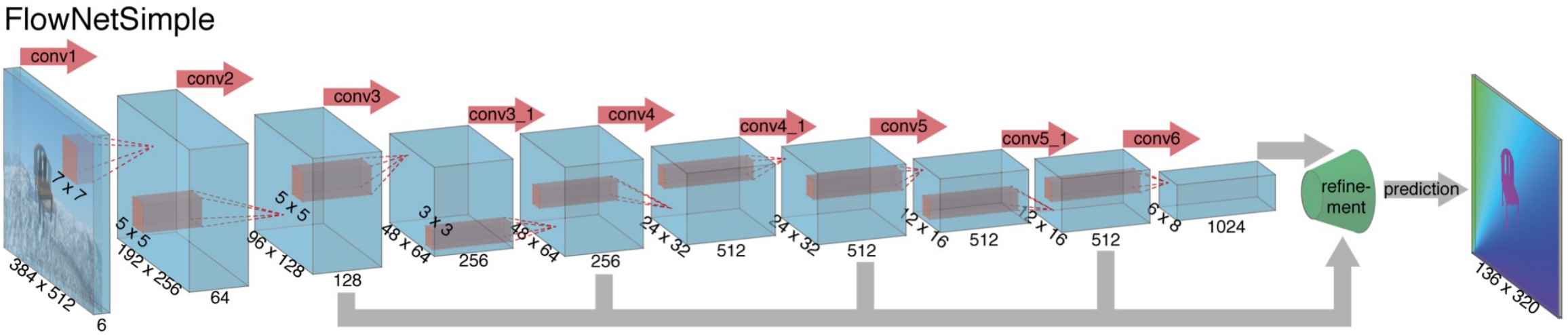
Supervised Method



$$\min \sum_{k} L(f_\theta(I_1, I_2), v)$$

Training Data Synthesis

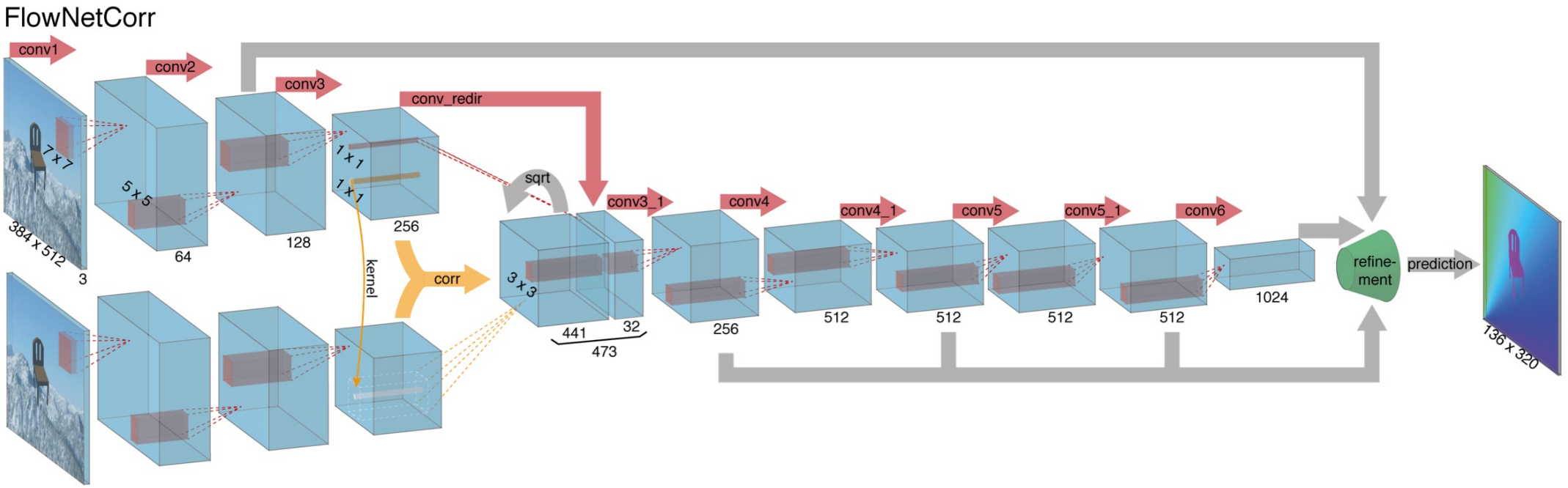
FlowNet



Stack both input images together and feed them through a rather generic network, allowing the network to decide itself how to process the image pair to extract the motion information.

Fischer et al. “FlowNet: Learning Optical Flow with Convolutional Networks”,2015

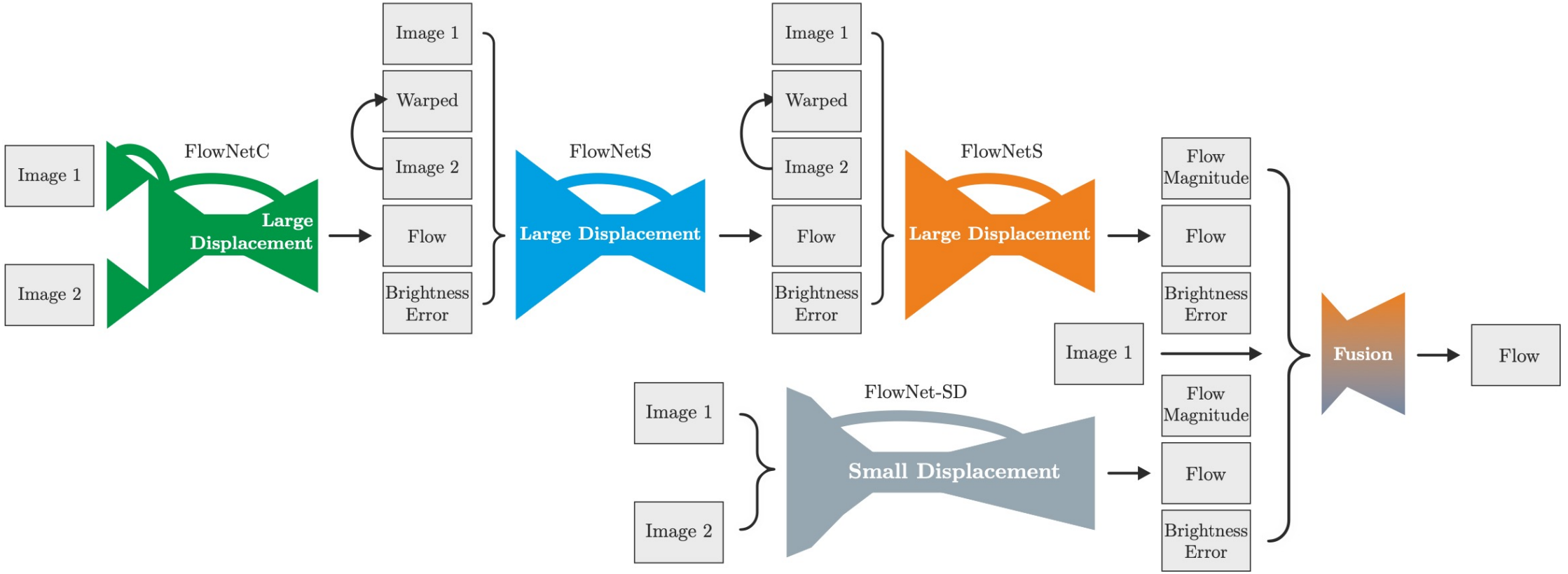
FlowNet



Resembles the standard matching approach when one first extracts features from patches of both images and then com- pares those feature vectors.

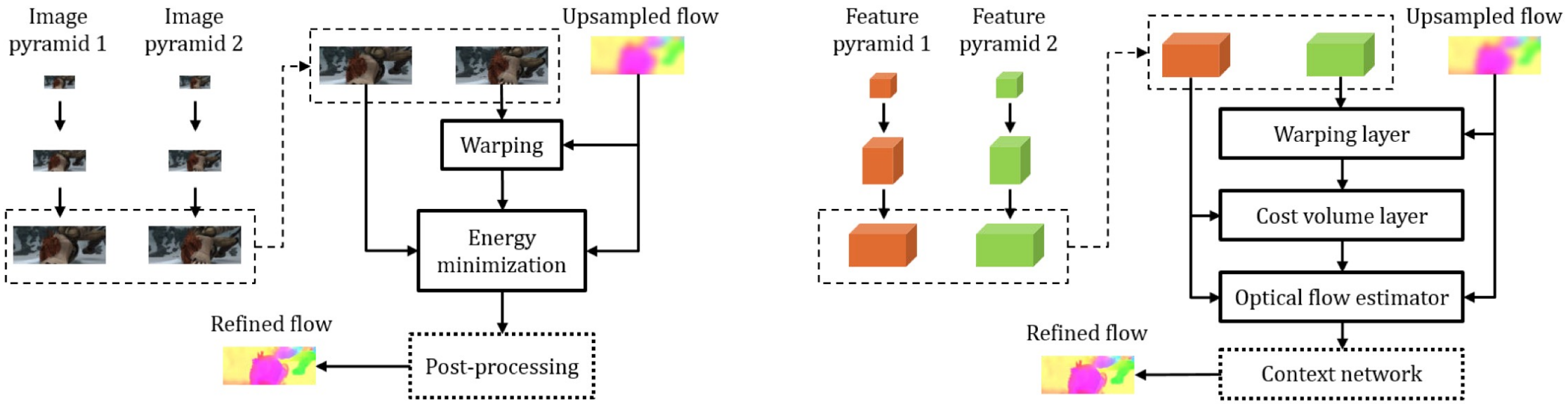
Fischer et al. “FlowNet: Learning Optical Flow with Convolutional Networks”,2015

FlowNet2



"FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks "

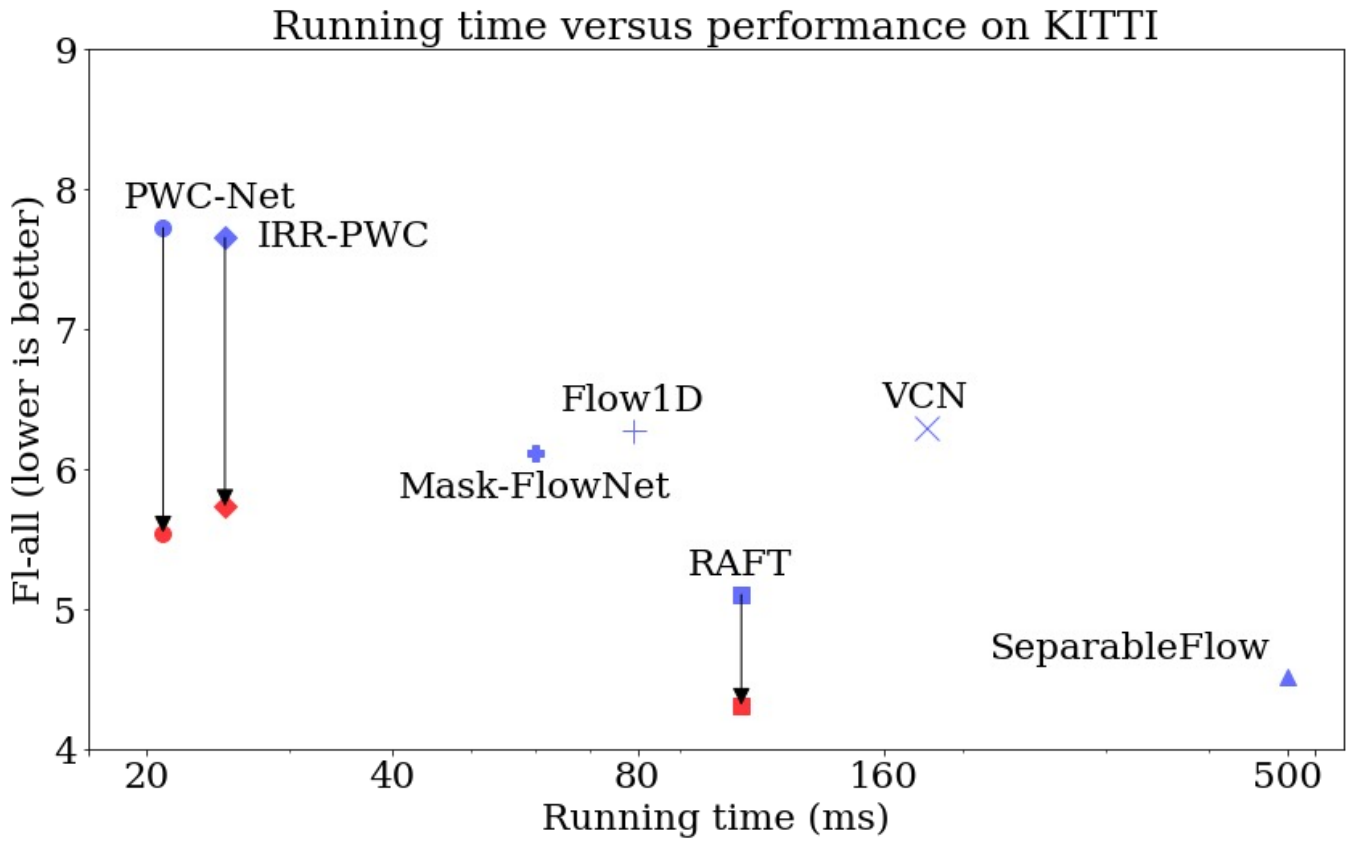
PWC-Net



Traditional coarse-to-fine approach (left) vs. PWC-Net (right)

Cost volume: $\mathbf{cv}^l(\mathbf{x}_1, \mathbf{x}_2) = \frac{1}{N}\left(\mathbf{c}_1^l(\mathbf{x}_1)\right)^\mathsf{T} \mathbf{c}_w^l(\mathbf{x}_2) \quad \mathbf{c}_w^l(\mathbf{x}) = \mathbf{c}_2^l\left(\mathbf{x} + \mathrm{up}_2(\mathbf{w}^{l+1})(\mathbf{x})\right)$

Training Matters



- *Dataset matters*
  FlyingChairs ⇨ AutoFlow
- *Gradient clipping matters*
  NoGC ⇨ YesGC
- *Training schedule matters*
  Piecewise ⇨ OneCycle
- *Training iterations matter*
  standard ⇨ 4-10 times more

Better training significantly improves performance

Sun et al. “Disentangling Architecture and Training for Optical Flow”, 2022

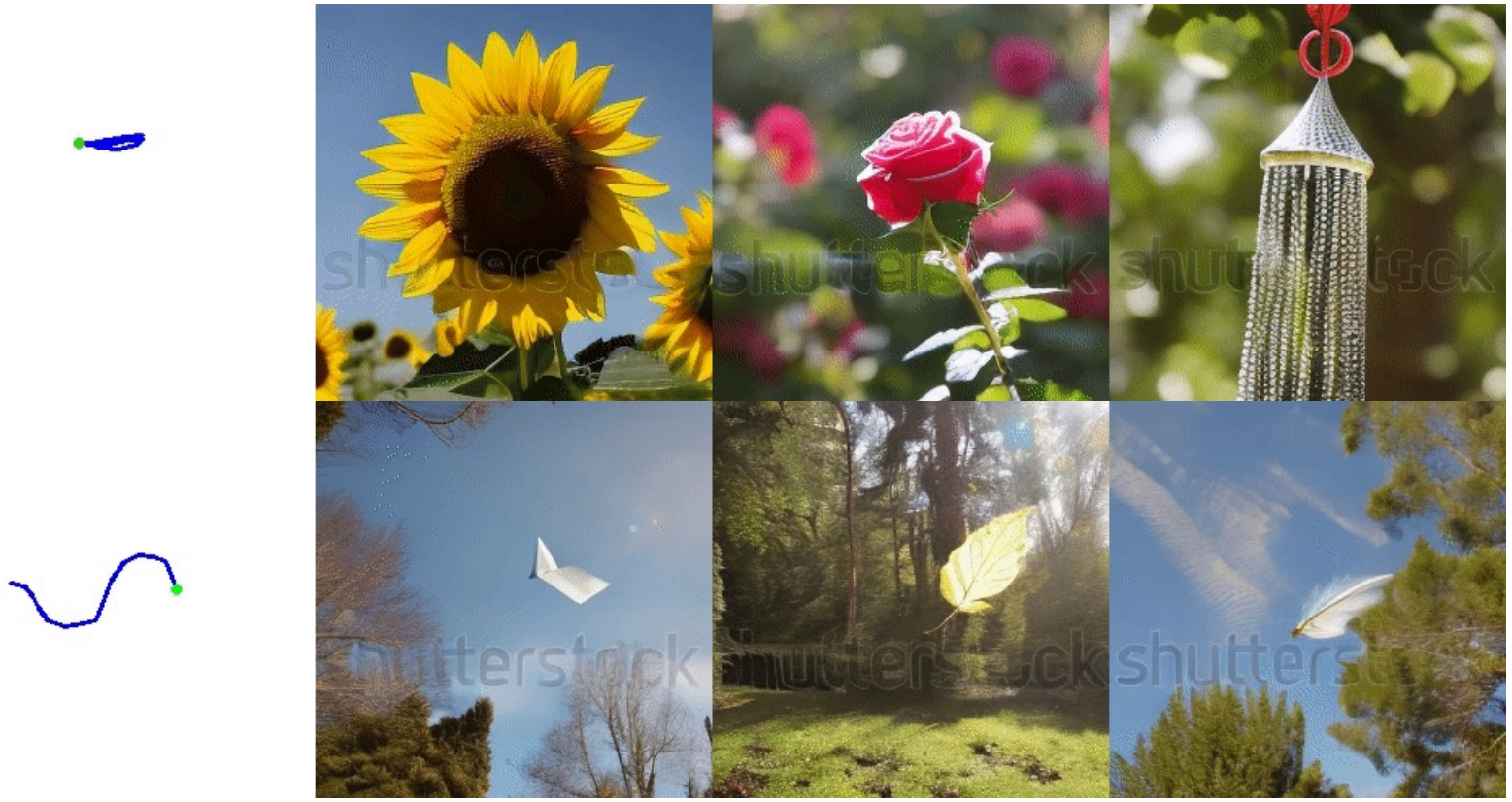
Applications

- Video frame interpolation

Jiang et al. “Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation”, 2018

Applications

- Video generation



Wang et al. “MotionCtrl: A Unified and Flexible Motion Controller for Video Generation “, 2024

Detection and Segmentation

Semantic Segmentation

- Semantic Segmentation follows three steps:

1. Classifying: Classifying a certain object in the image.
2. Localizing: Finding the object and drawing a bounding box around it.
3. Segmentation: Grouping the pixels in a localized image by creating a segmentation mask.

- It can also be thought of as the classification of images at a pixel level.

An overview of Semantic Image Segmentation

Input

Segmented

1: Person
2: Bench
3: Plant/Grass
4: Cat

One-hot Encoding

3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 1 1 3 3 3 3 3
3 3 3 3 3 3 3 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 3 3 3 3
3 3 3 3 3 3 3 1 1 1 3 3 3 3
3 3 3 3 3 3 1 1 1 1 3 3 3 3
3 3 3 3 3 3 1 1 1 1 3 3 3 3
3 3 2 3 3 3 1 1 1 1 1 2 3 3
3 3 1 1 1 1 1 1 1 1 1 2 2 2
3 3 1 1 1 1 1 1 1 1 2 2 2 2
4 4 1 1 2 2 2 2 2 2 2 2 2 2
4 4 1 1 3 2 3 3 3 3 3 2 2 3
4 1 1 1 1 2 3 3 3 3 3 2 3 3

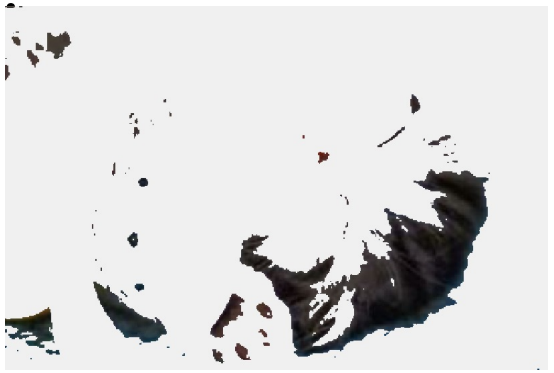
Semantic Labels

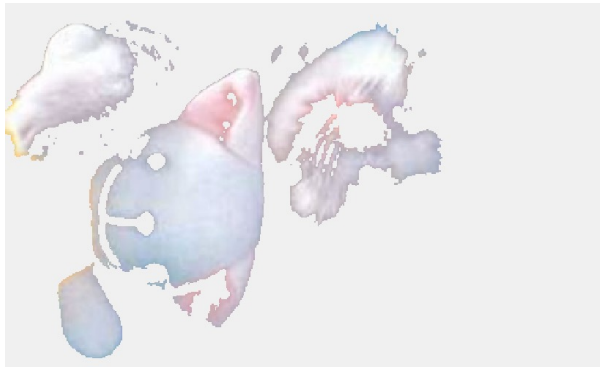
Image credit: V7 Labs

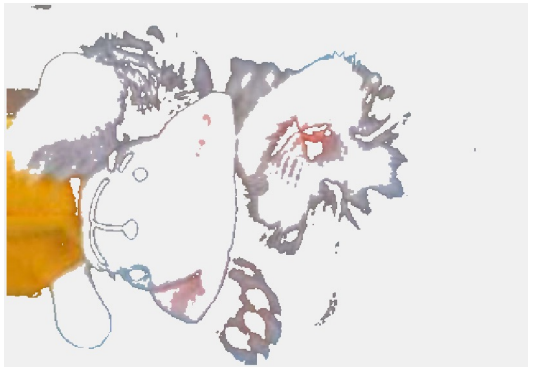
Clustering

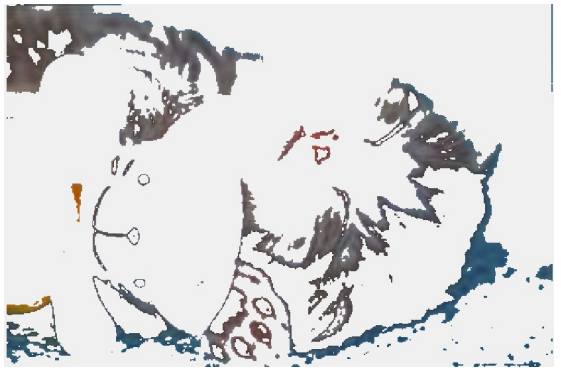
- View segmentation as clustering: pixels sharing certain features such as color, intensity, or texture are grouped together and represented as a single entity.

5 clusters



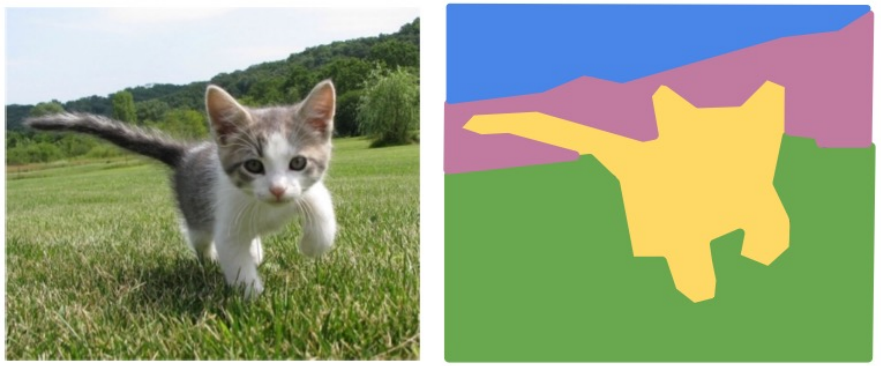






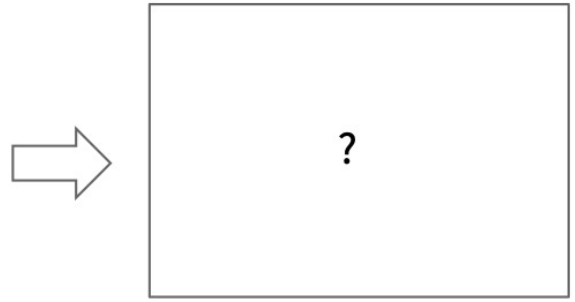
https://ai.stanford.edu/~syyeung/cvweb/tutorial3.html

Deep Learning Methods



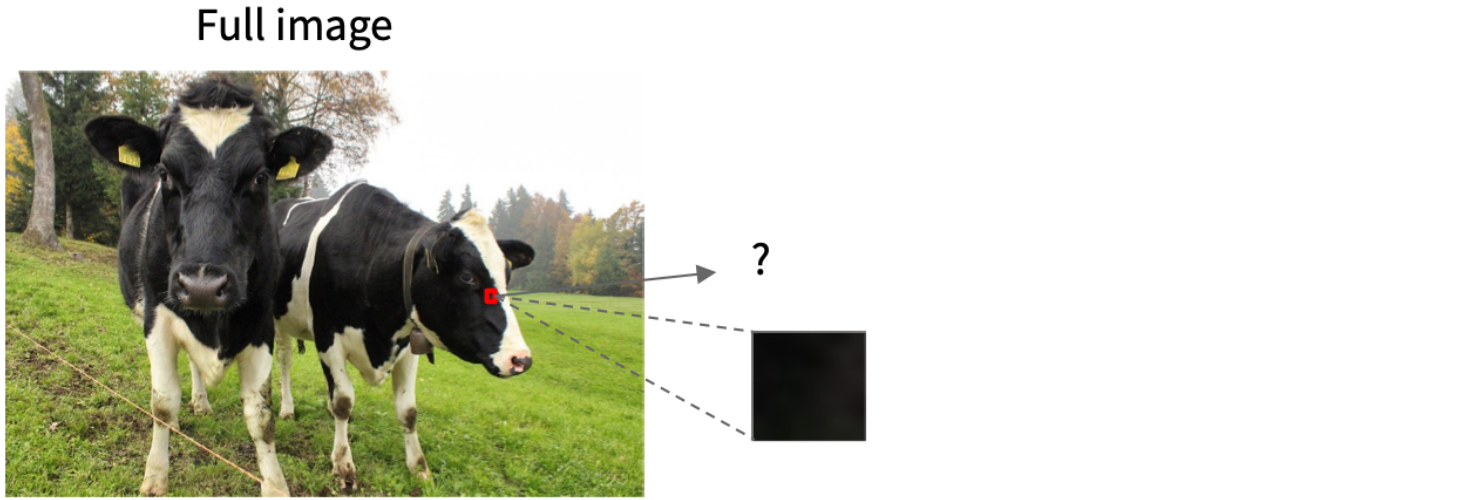
GRASS, CAT, TREE,
SKY, ...

Paired training data: for each training image, each pixel is labeled with a semantic category.



At test time, classify each pixel of a new image.

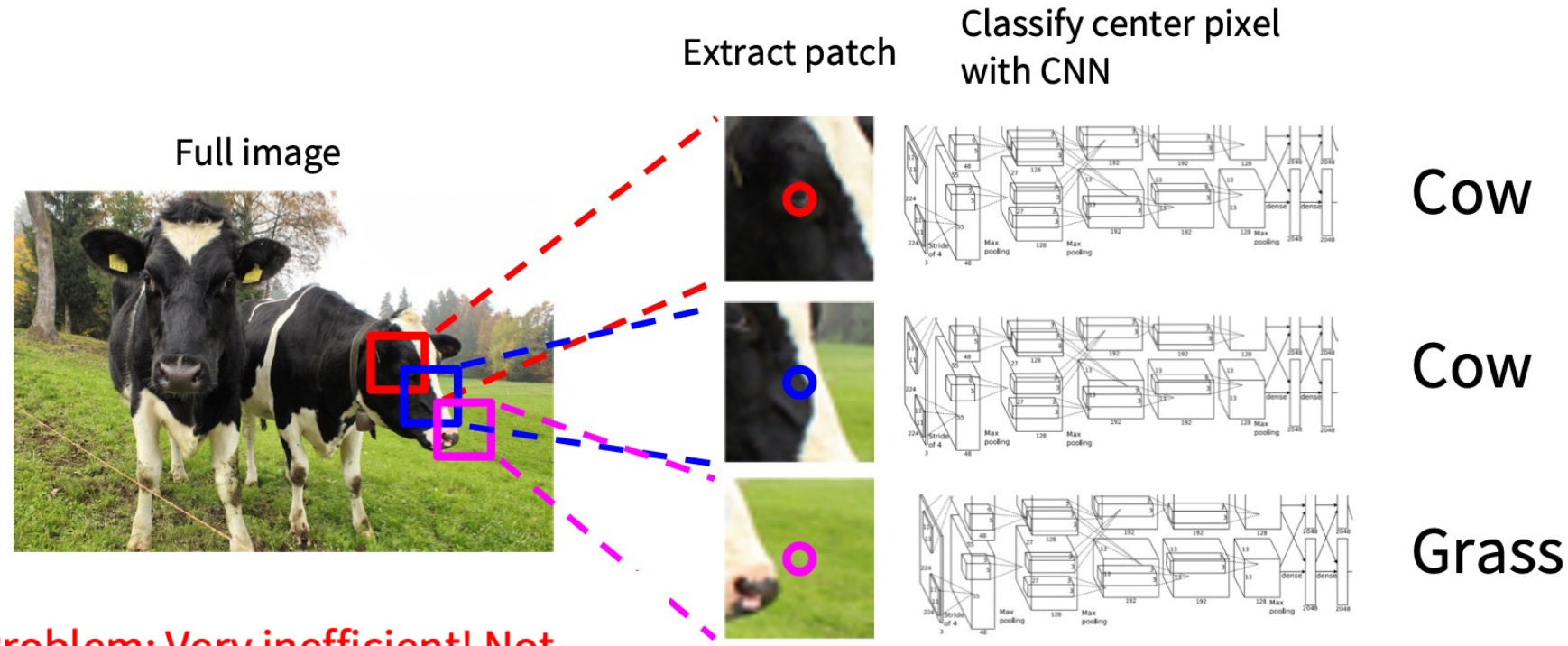
Deep Learning Methods



Impossible to classify without context

Q: how do we include context?

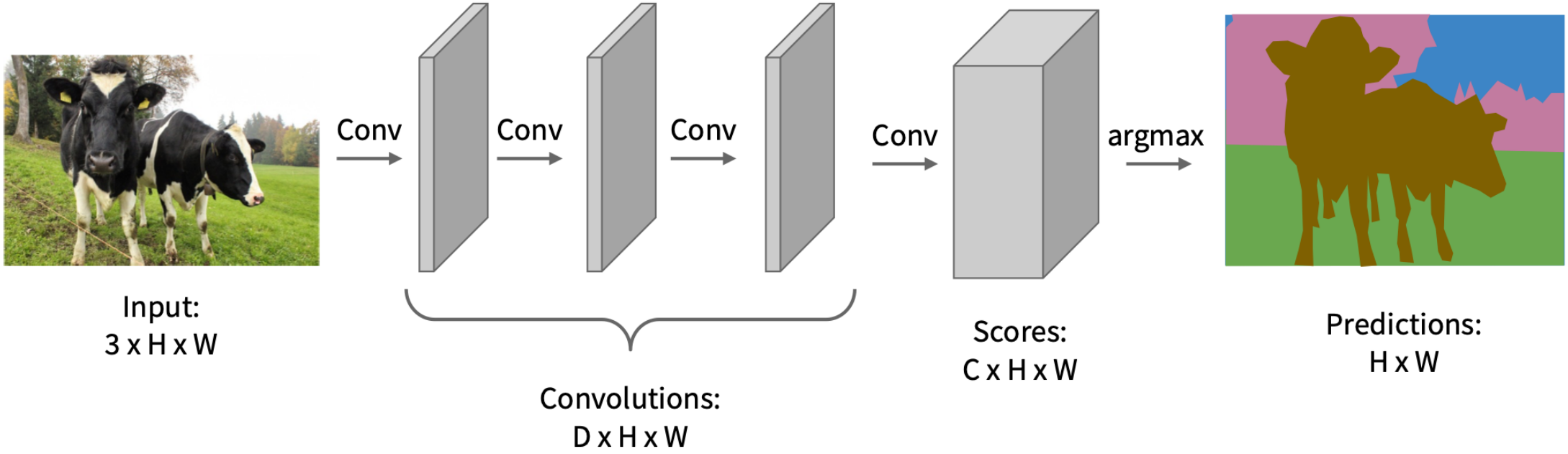
Deep Learning Methods



Problem: Very inefficient! Not reusing shared features between overlapping patches

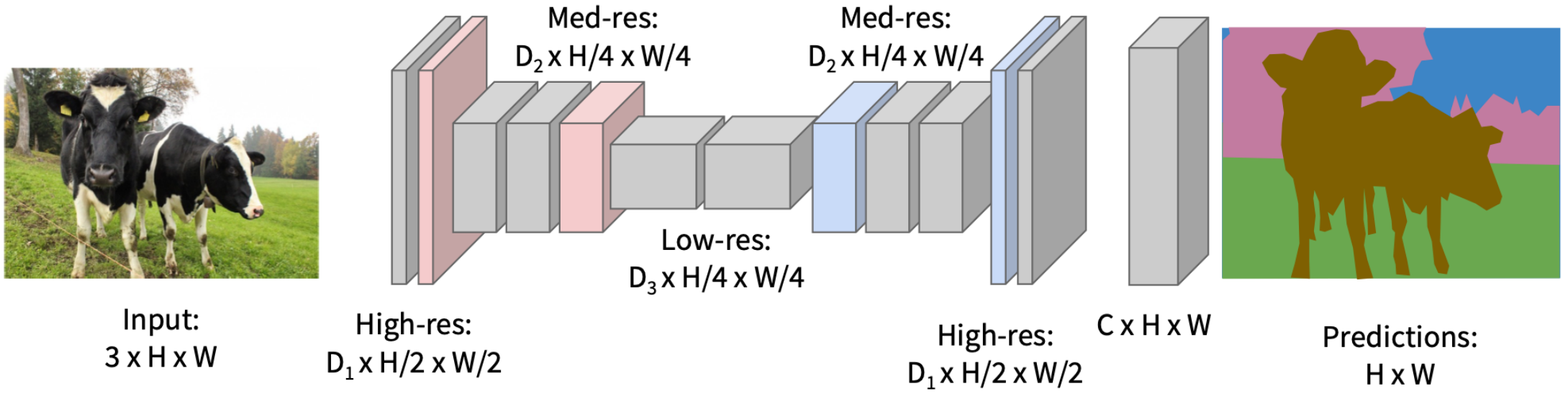
Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Deep Learning Methods



Different from classification architectures which often reduce feature spatial sizes to go deeper, semantic segmentation requires the output size to be the same as input size.

Deep Learning Methods



Design network as a bunch of convolutional layers, with downsampling and upsampling inside the network!

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015 Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Convolution as Matrix Multiplication

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Downsampling

Transposed convolution multiplies by the transpose of the same matrix:

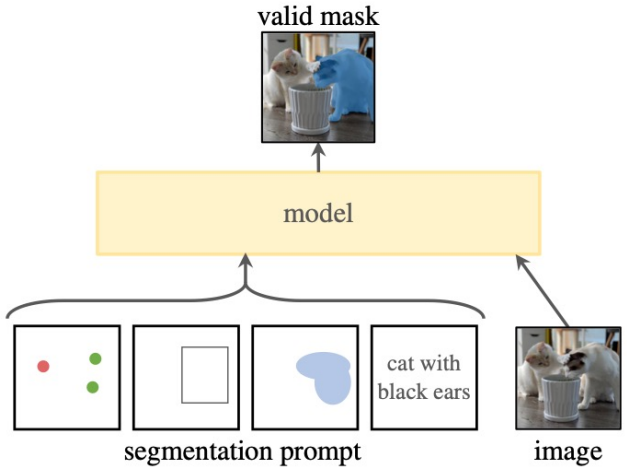
$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

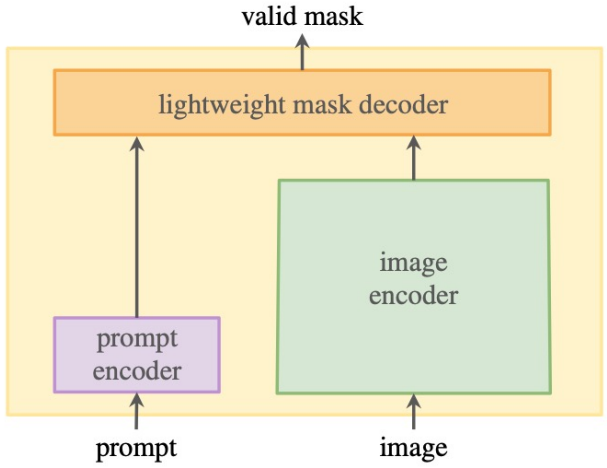
Example: 1D transposed conv, kernel size=3, stride=2, padding=0

Upsampling

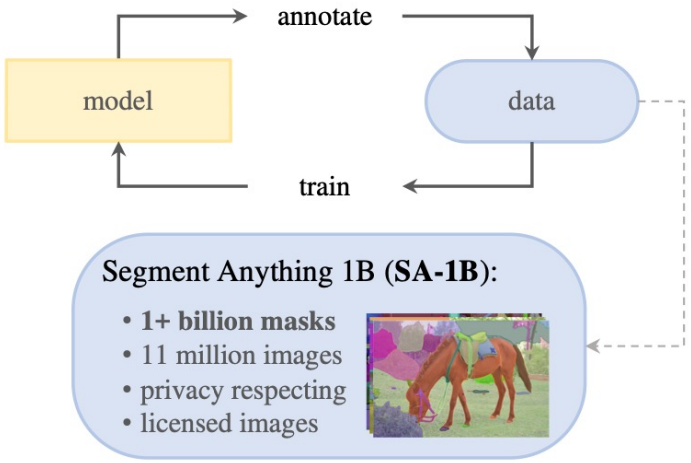
Segment Everything (SAM)



(a) **Task**: promptable segmentation



(b) **Model**: Segment Anything Model (**SAM**)



(c) **Data**: data engine (top) & dataset (bottom)

Foundation model for segmentation

Promotable Segmentation

- Pre-training: simulate a sequence of prompts (e.g., points, boxes, masks) for each training sample and compares the model's mask predictions against the ground truth.

Figure 3: Each column shows 3 valid masks generated by SAM from a single ambiguous point prompt (green circle).

Overview

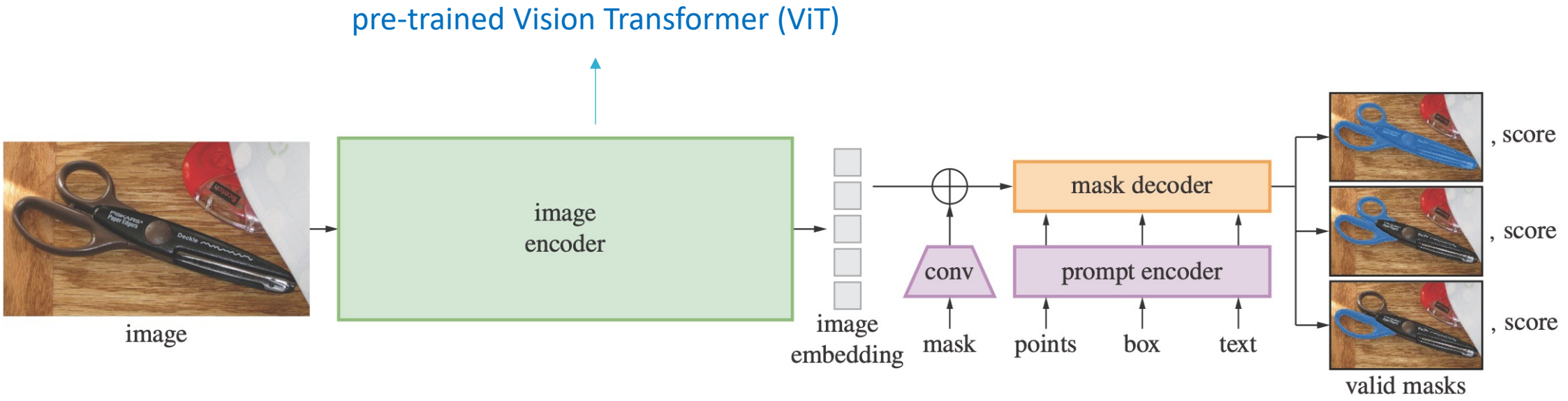
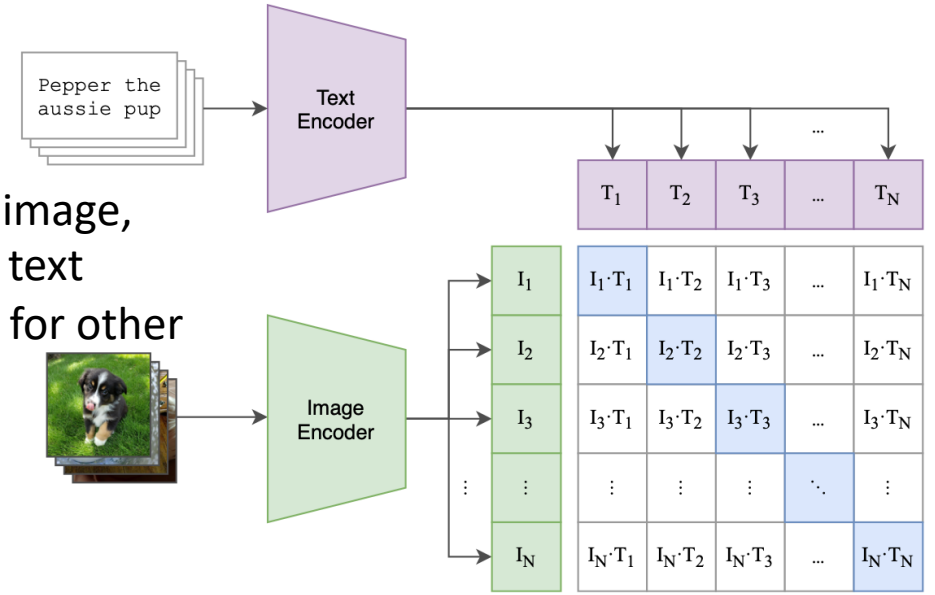


Figure 4: Segment Anything Model (SAM) overview. A heavyweight image encoder outputs an image embedding that can then be efficiently queried by a variety of input prompts to produce object masks at amortized real-time speed. For ambiguous prompts corresponding to more than one object, SAM can output multiple valid masks and associated confidence scores.

Prompt Encoder

- Two sets of prompts: sparse (points, boxes, text) and dense (masks).
  - Points and boxes: positional encodings
  - Dense prompts (i.e., masks) are embedded using convolutions
  - free-form text: text encoder from Contrastive Language- Image Pre-training (CLIP)

CLIP is trained for predict (image, text) pairings. The learned text encoder can be transfered for other downstream tasks



"Learning Transferable Visual Models From Natural Language Supervision", ICML,2021

Segment Anything Data Engine

High-quality, diverse, large amount of data is very important.

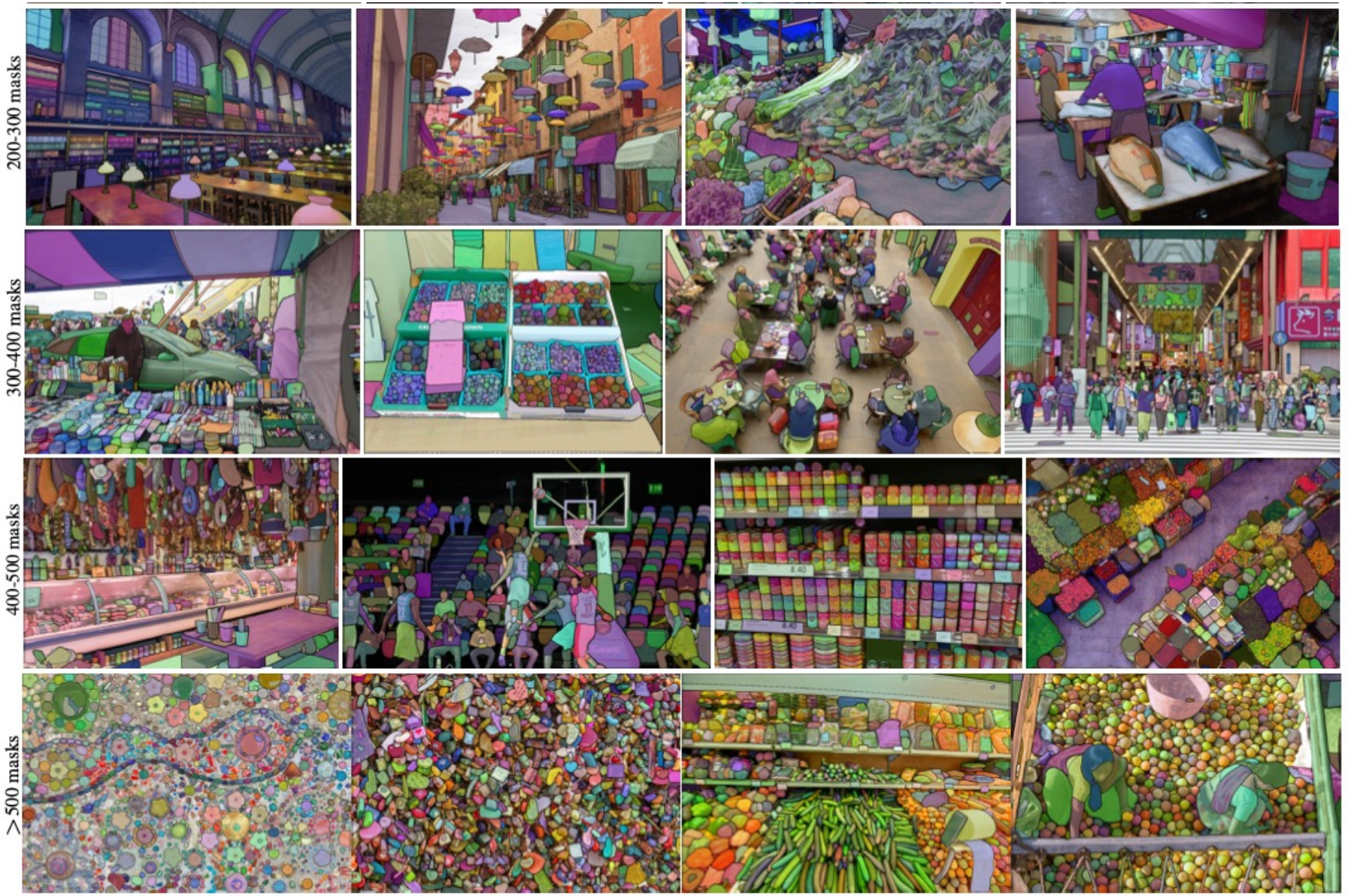
The data engine has three stages:

(1) a model-assisted manual annotation stage: interactive segmentation with a team of professional annotators

(2) a semi-automatic stage with a mix of automatically predicted masks and model-assisted annotation

(3) a fully automatic stage in which our model generates masks without annotator input.

Final dataset: 11M images, 1.1B high-quality masks



Example images with overlaid masks from the newly introduced dataset

Zero-Shot Edge Detection

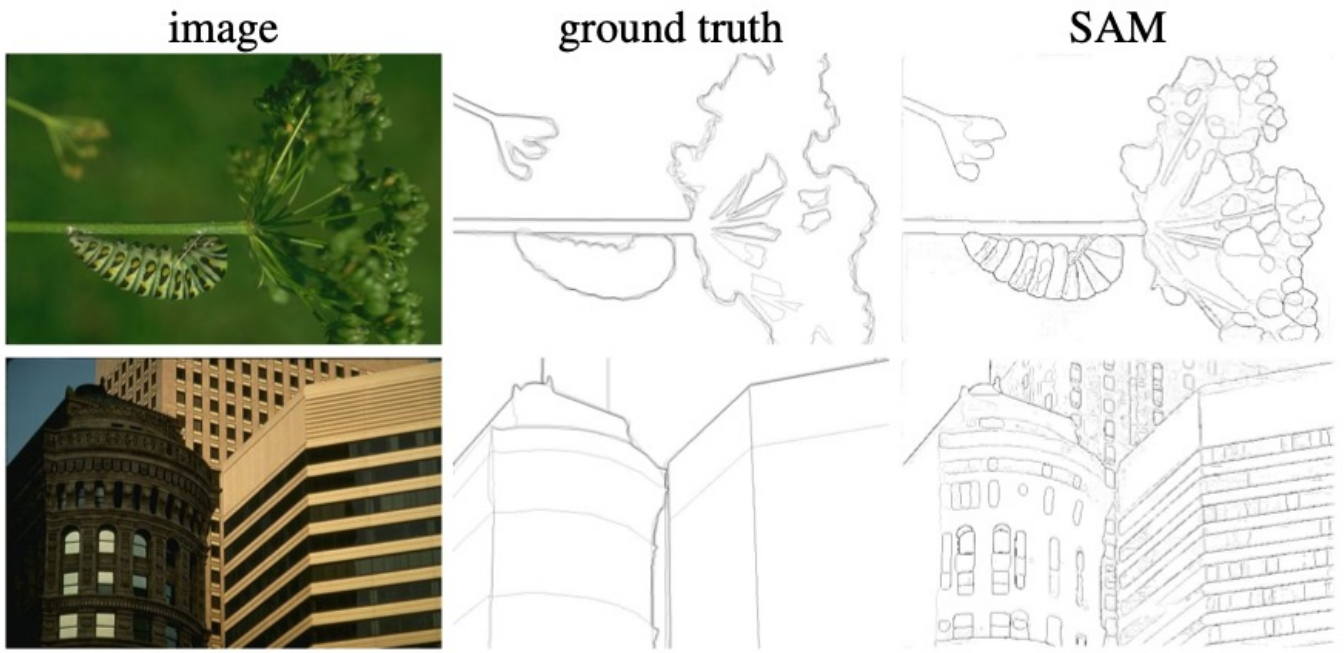
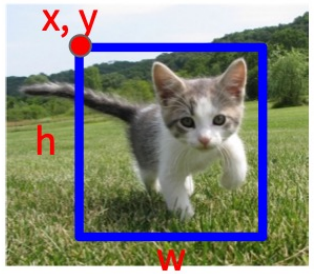


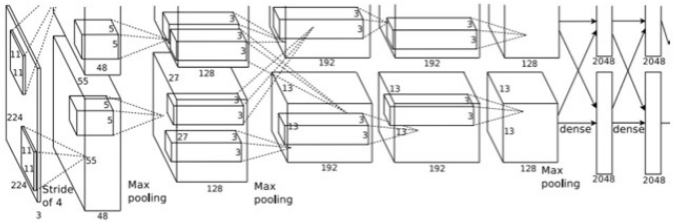
Figure 10: Zero-shot edge prediction on BSDS500. SAM was not trained to predict edge maps nor did it have access to BSDS images or annotations during training.

Specifically, they prompt SAM with a 16×16 regular grid of foreground points resulting in 768 predicted masks (3 per point). Then, edge maps are computed using Sobel filtering of unthresholded mask probability maps.

Object Detection: Single Object



This image is CC0 public domain



Vector: 4096

Fully Connected: 4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Correct label: Cat

Softmax Loss

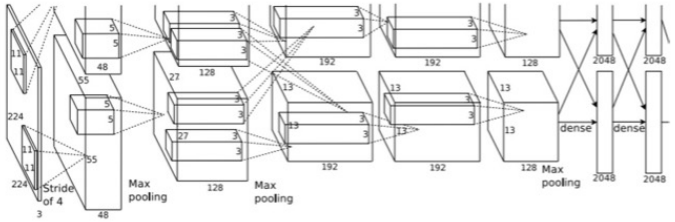
Fully Connected: 4096 to 4

Box Coordinates (x, y, w, h)

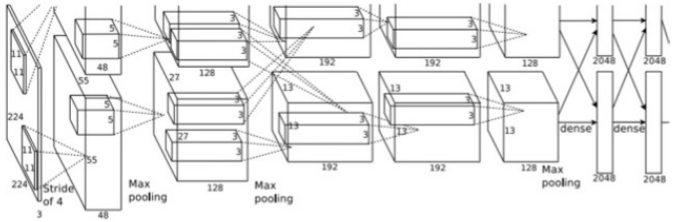
L2 Loss

Correct box: (x', y', w', h')

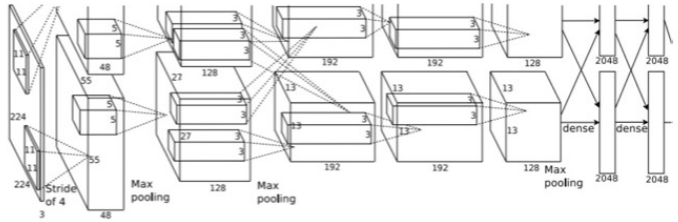
Object Detection: Multiple Objects



CAT: (x, y, w, h)



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

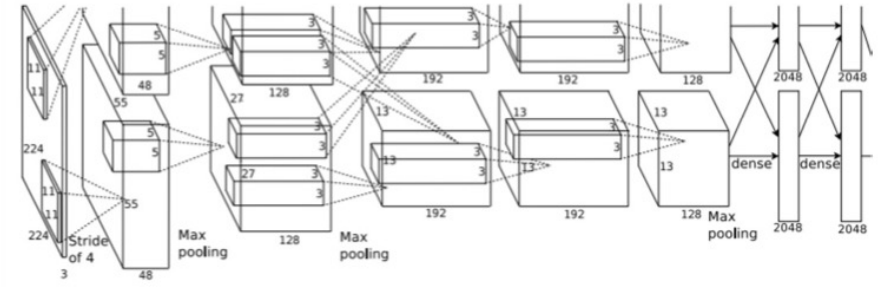


DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Each image needs a different number of outputs!

Object Detection: Multiple Objects

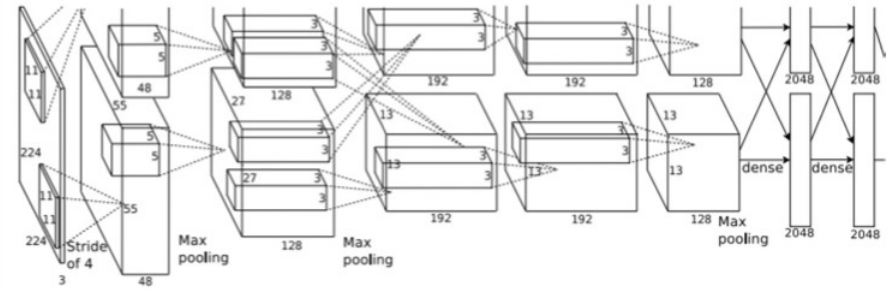
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

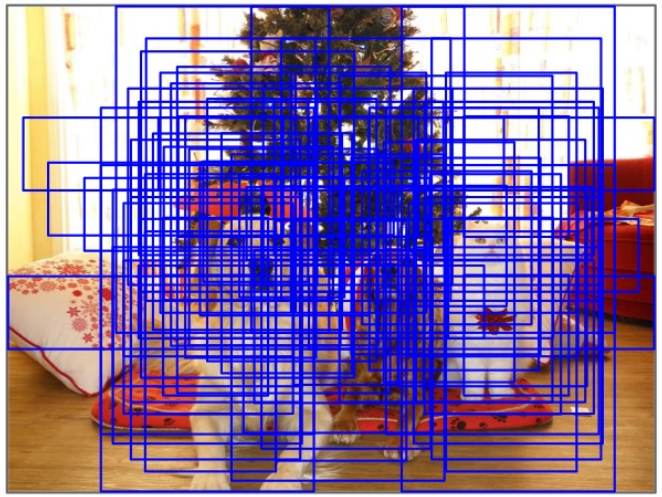
Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

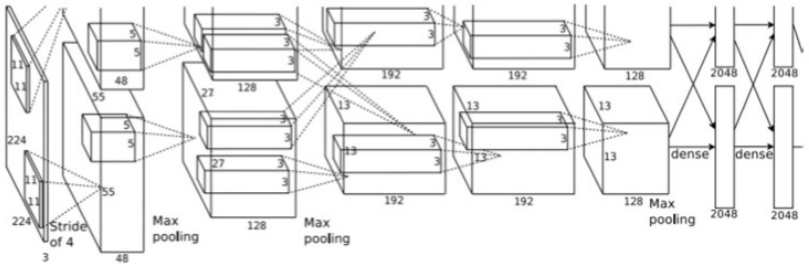


Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects



Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

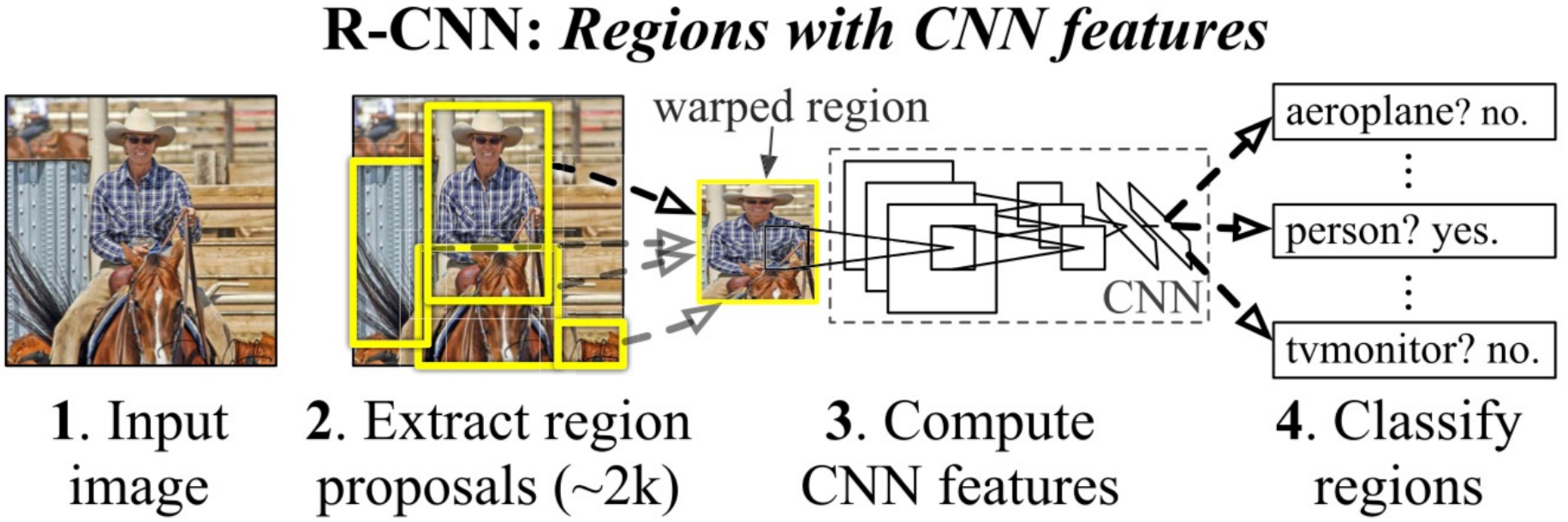


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

The Region Proposal-based Methods

- Generate region proposals at first and then classify each proposal into different object categories.



Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

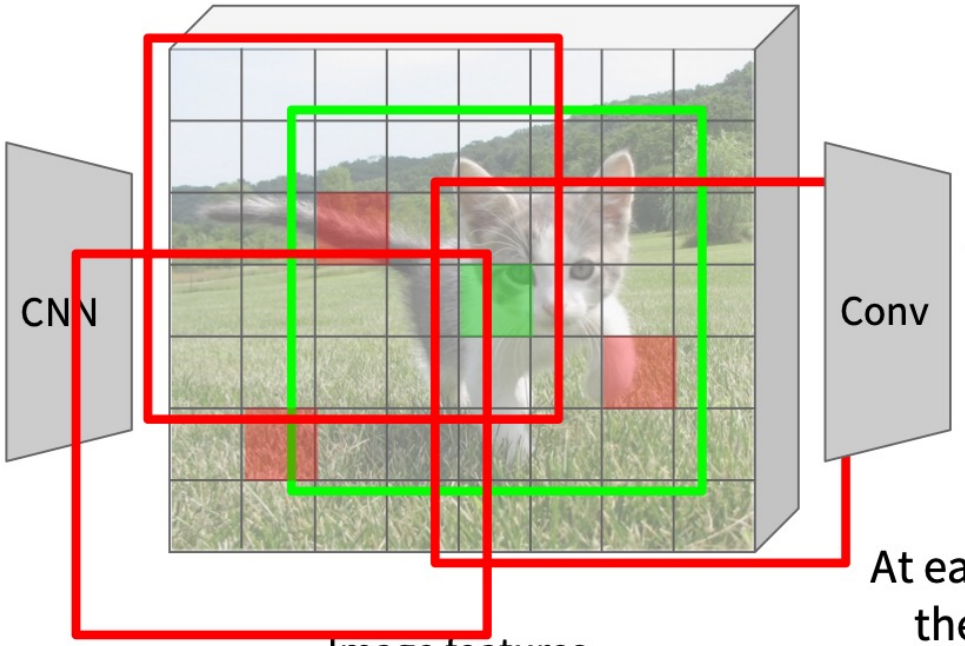


Image features
(e.g. 512 x 20 x 15)

Imagine an anchor box of fixed size at each point in the feature map

Anchor is an object?
1 x 20 x 15

At each point, predict whether the corresponding anchor contains an object (binary classification)

Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

CNN

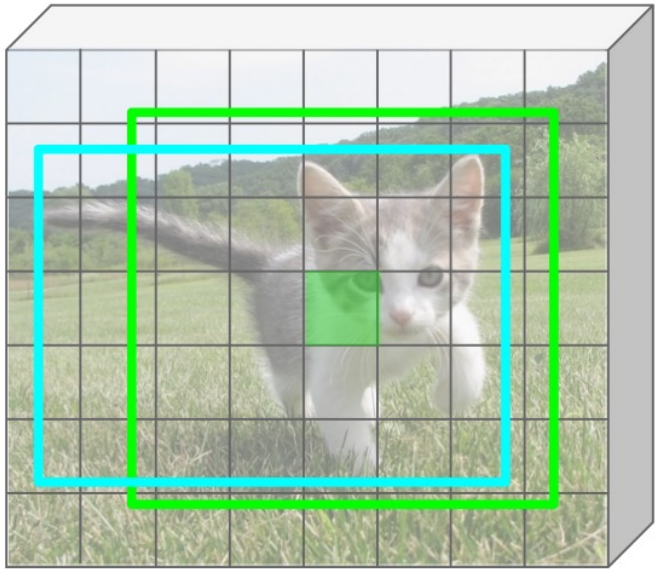


Image features
(e.g. 512 x 20 x 15)

Conv

Anchor is an object?
1 x 20 x 15

Box corrections
4 x 20 x 15

For positive boxes, also predict a corrections from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network

Input Image
(e.g. 3 x 640 x 480)

CNN

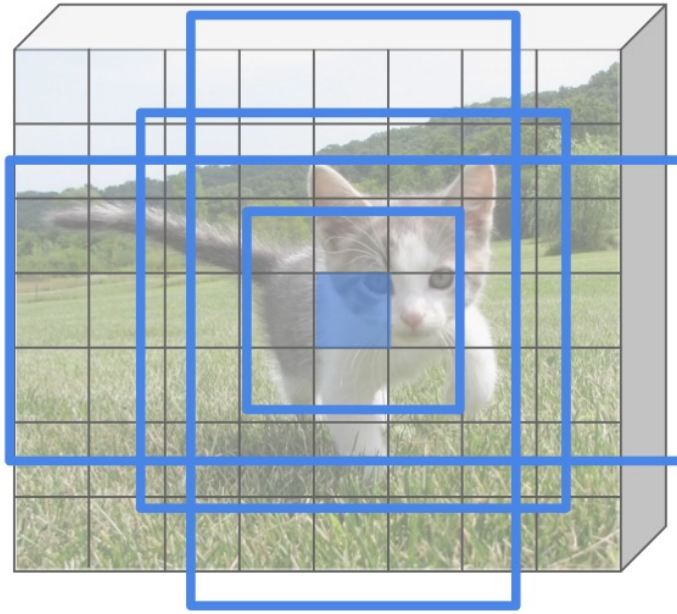


Image features
(e.g. 512 x 20 x 15)

Conv

In practice use K different anchor boxes of different size / scale at each point

Anchor is an object?
K x 20 x 15

Box transforms
4K x 20 x 15

Sort the K*20*15 boxes by their “objectness” score, take top ~300 as our proposals

Non-Maximum Suppression

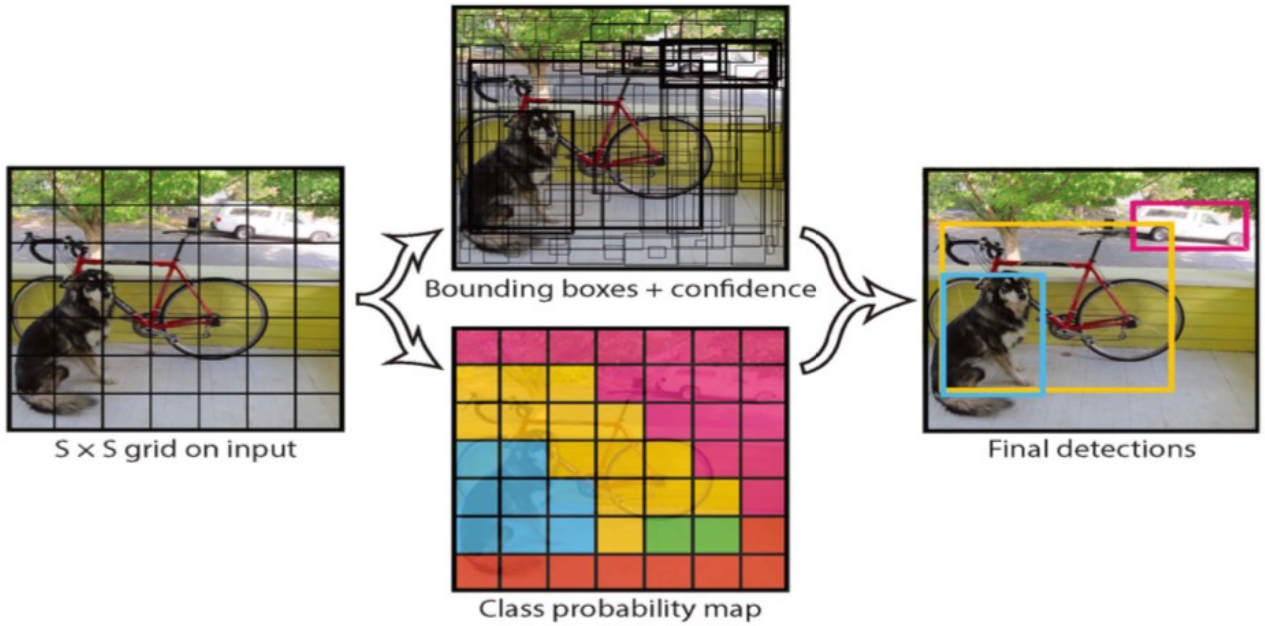
- Intersection Over Union (IoU)

$$IOU(Box1, Box2) = Intersection\_Size(Box1, Box2) / Union\_Size(Box1, Box2)$$

- Non-Maximum Suppression (NMS)
  1. Sort the bounding boxes based on their confidence scores.
  2. Select the bounding box with the highest confidence score and save it as a detection.
  3. Remove all the bounding boxes that have a significant overlap with the selected bounding box. The amount of overlap is typically determined by a predefined threshold value.
  4. Repeat steps 2 and 3 until no more bounding boxes remain.

Regression/Classification-Based Framework

- YOLO: map straightly from image pixels to bounding box coordinates and class probabilities to reduce time expense.



"You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Regression/Classification-Based Framework

- Training loss

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2. \qquad (6)
\end{aligned}
$$

- i: cell center, j: bounding box
- (x_i, y_i) center of the box
- (w_i, h_i) height and width of the box
- C_i: confidence score

YOLO11

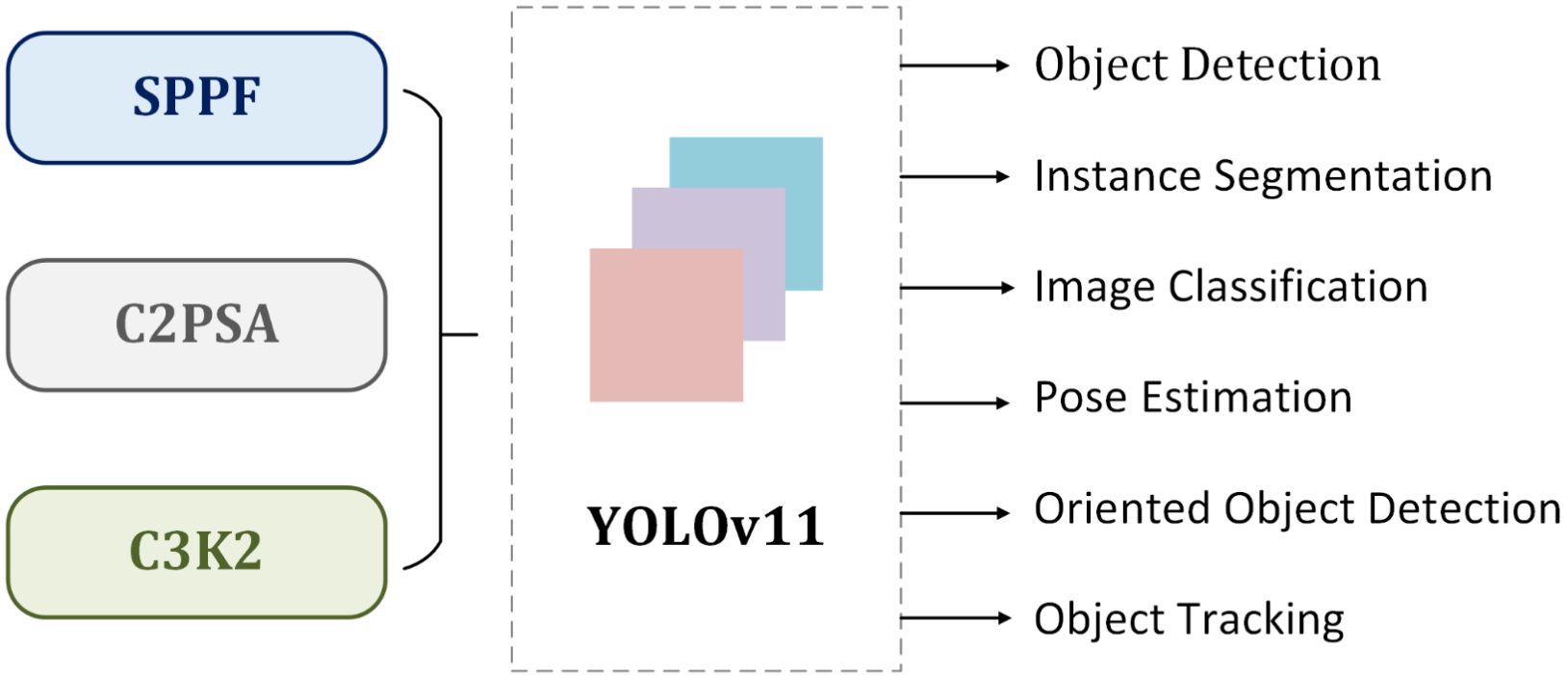
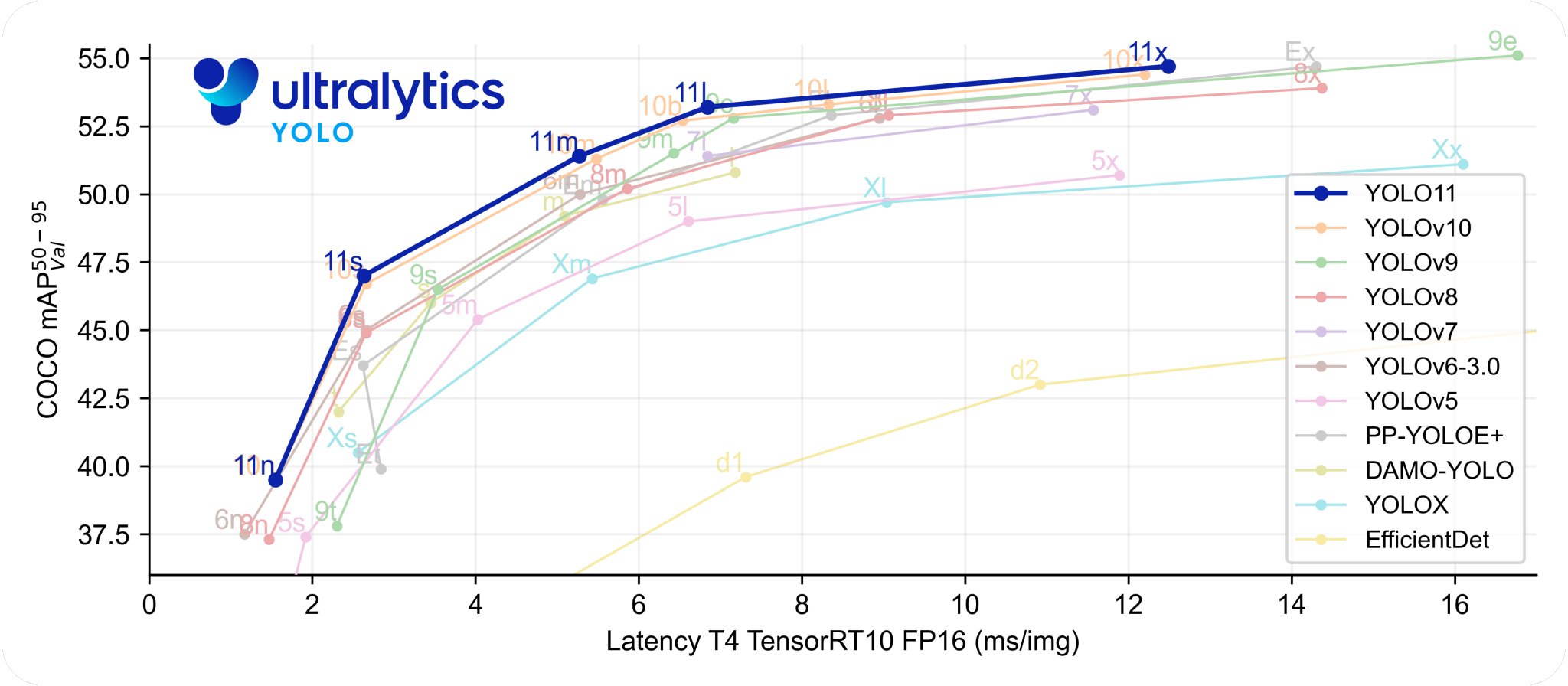


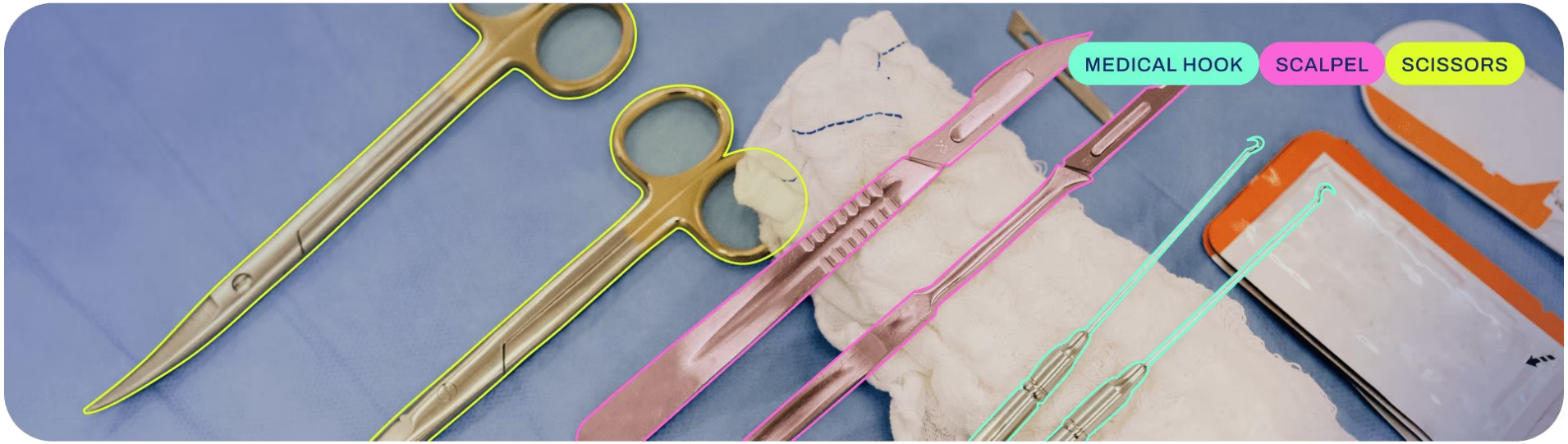
Figure 1: Key architectural modules in YOLO11

YOLO11

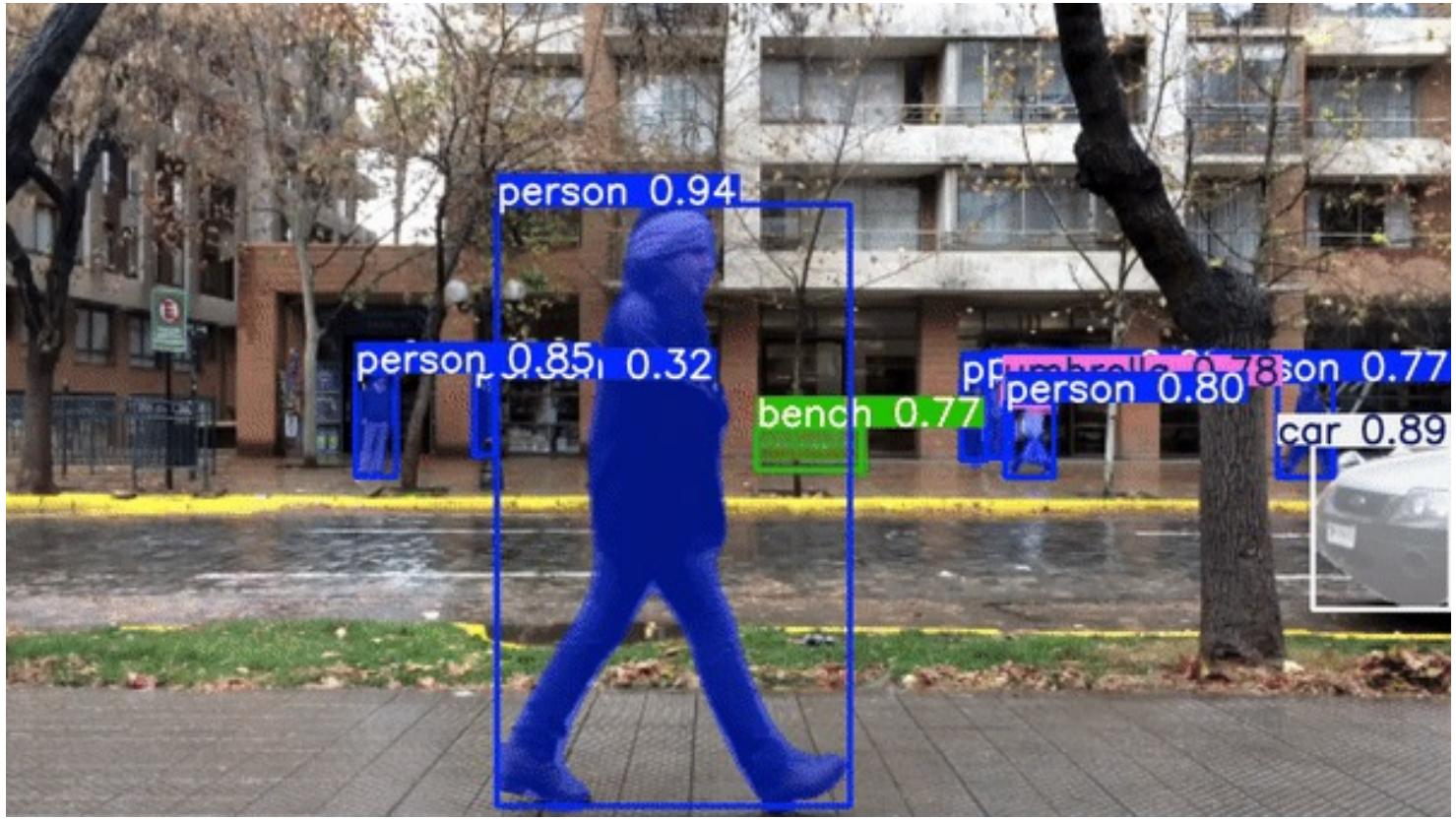

ultralytics
YOLO
COCO mAP$^{50-95}_{val}$
Latency T4 TensorRT10 FP16 (ms/img)
55.0
52.5
50.0
47.5
45.0
42.5
40.0
37.5
0
2
4
6
8
10
12
14
16
11n
11s
11m
11l
11x
YOLO11
YOLOv10
YOLOv9
YOLOv8
YOLOv7
YOLOv6-3.0
YOLOv5
PP-YOLOE+
DAMO-YOLO
YOLOX
EfficientDet

Instance Segmentation

- Instance segmentation goes a step further than object detection and involves identifying individual objects in an image and segmenting.
- The output is a set of masks or contours that outline each object in the image, along with class labels and confidence scores.



Instance Segmentation

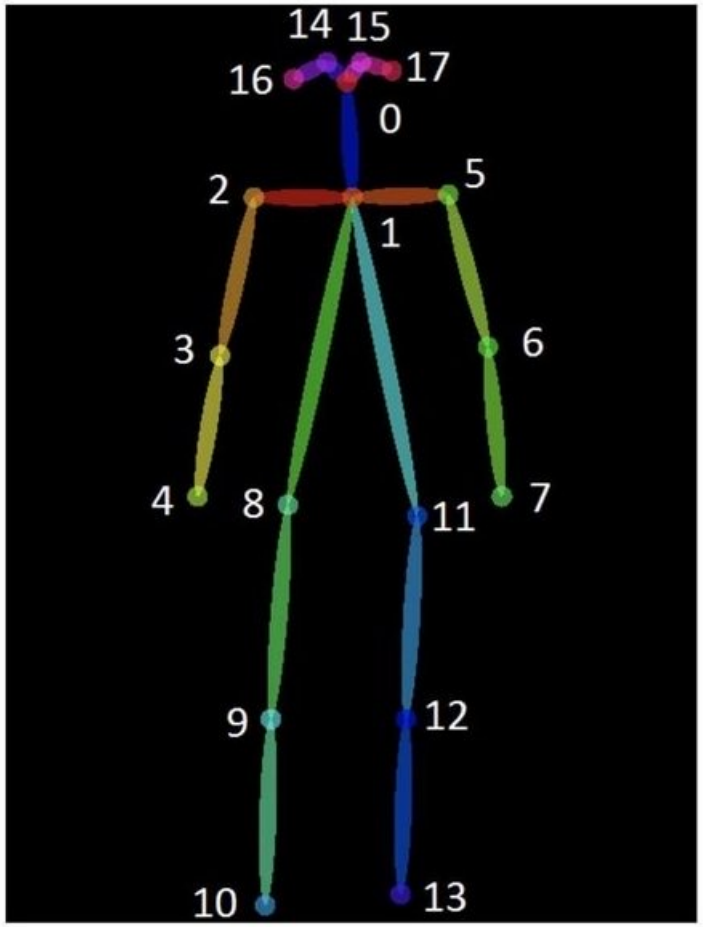


Pose Estimation

- A challenging task because the body’s appearance changes dynamically.
- YOLO11 performs pose estimation by detecting and predicting key points on the object, such as joints in a human body.
- Pose estimation layers are added in the head, and the network is trained to predict the coordinates of key points.
- A post-processing step connects the points to form the skeleton structure, enabling real-time pose recognition.

Pose Estimation

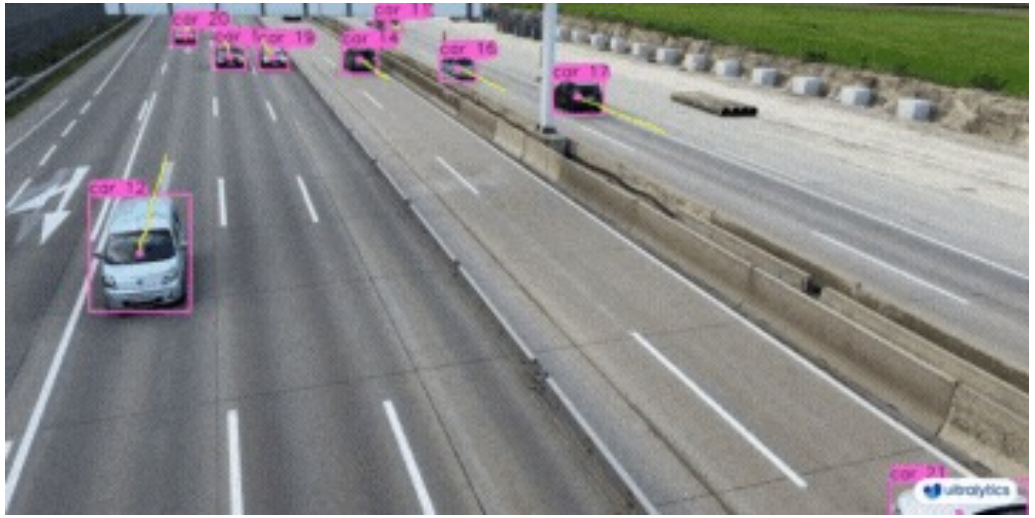
- Key points: "nose", "left_eye", "right_eye", "left_ear", "right_ear", "left_shoulder", "right_shoulder", "left_elbow", "right_elbow", "left_wrist", "right_wrist", "left_hip", "right_hip", "left_knee", "right_knee", "left_ankle", "right_ankle”
- Each keypoint is annotated with three numbers (x,y,v), where x and y mark the coordinates, and v indicates if the keypoint is visible.

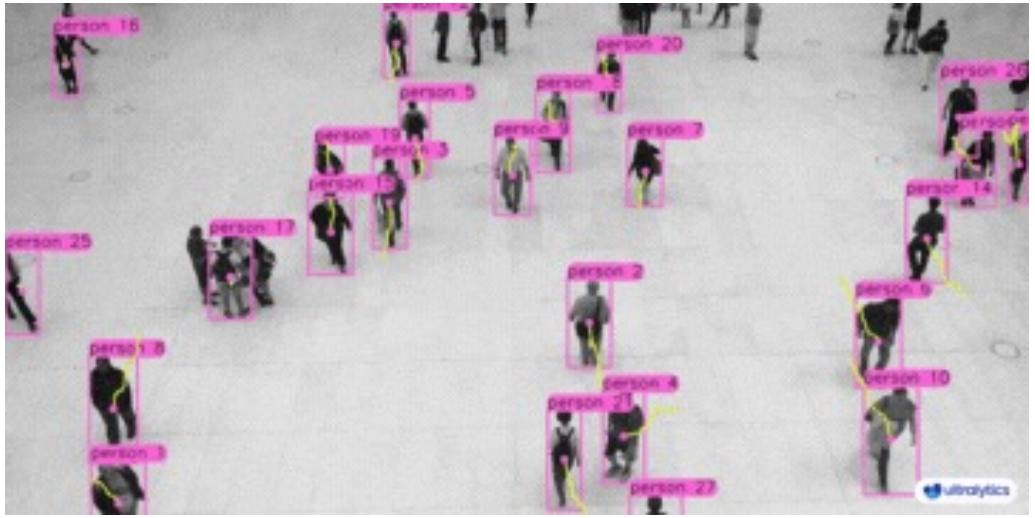


Pose Estimation

Object Tracking

Object Tracking: creating a unique ID for each of the initial detections, and then tracking each of the objects as they move around frames in a video





YOLO11: Real-Time Tracking&Multiple Tracker Support