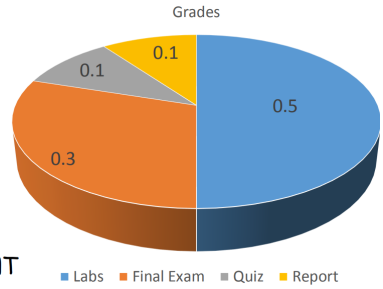
Lecture Note

- Middle-Term Report 10%
- Final Exam 30%
- Labs 50%
- Quiz: 10%



- LABS are VERY IMPORTANT

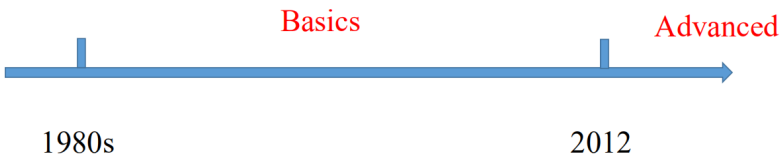
Exams test you on

- General knowledge of visual technology
- Ability to model simple tasks
- Understand the general flow of the visual system (implement)

1. Introduction

- The first **digital** image was produced in 1920
- Charge-coupled device(CCD)
- Larry Roberts at MIT(1960): extracting 3D geometrical information from 2D perspective views of blocks

 Father of computer vision



Applications	
Image classification	Building a model (function) from hypothesis space □ Image to label □ Matrix to number
Object detection	Input: image Output: locations of objects
Image segmentation	Input: image Output: regions, structures
Pose estimation	Input: image Output: configuration
Image captioning	generating textual description of an image
Visual Question Answering (VQA)	• Given an image and a question (text) about the image □ Aim to provide an accurate natural language answ
Image generation	
Object tracking	Input: video □ Output: trajectory
Object re-identification	Input: images (multi-camera) Output: associations

- Image classification
- Object detection
- Image segmentation
- Pose estimation
- Visual language
  - Image captioning
  - VQA…
- Object tracking
- Object identification
- View synthesis (3D)

column vectors $\boldsymbol{x} = [x\ y]^T$. Some commonly used matrices are $\boldsymbol{R}$ for rotations, $\boldsymbol{K}$ for calibration matrices, and $\boldsymbol{I}$ for the identity matrix. Homogeneous coordinates (Section 2.1) are denoted with a tilde over the vector, e.g., $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x, y, 1) = \tilde{w}\bar{\boldsymbol{x}}$ in $\mathcal{P}^2$. The cross product operator in matrix form is denoted by $[\,]_\times$.

标准化

2. Image formation

formation process: 3D (real-world) to 2D (matrix)

(a) Perspective projection

(b) Light scattering when hitting a surface

(c) Lens optics

(d) Bayer color filter array

2.1 Geometric primitives and transformations (几何图元和变换)

(Vanishing points and vanishing lines)

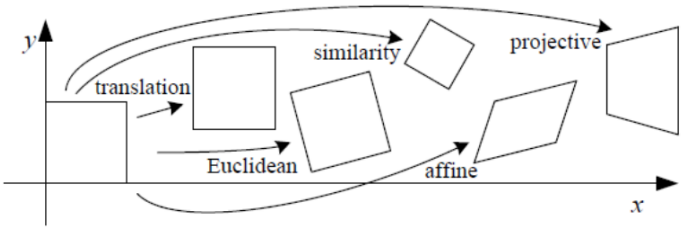
2D points, lines

3D points, lines, plane

- 2D transformations
  - Translation $x' = x + t = \begin{bmatrix} I & t \end{bmatrix} \bar{x}$ $\quad \bar{x}' = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \bar{x}$
  - Rotation + translation $x' = Rx + t = \begin{bmatrix} R & t \end{bmatrix} \bar{x}$

旋转 θ °

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



- Hierarchy of 2D coordinate transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$	8	straight lines	

- Hierarchy of 3D coordinate transformations

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{3\times4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{3\times4}$	6	lengths	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{3\times4}$	7	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{3\times4}$	12	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{4\times4}$	15	straight lines	

2.2 Projections

- Use a linear 3D to 2D projection matrix

- Orthography
  - Orthographic projection

$$x = [I_{2\times2}|0]\, p$$

$$\tilde{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tilde{p}$$

(2D on the left, 3D on the right)

  - Scaled orthography
    - ✓ First project the world points onto a local fronto-parallel image plane
    - ✓ Then scale this image using regular perspective projection

$$x = [sI_{2\times2}|0]\, p$$

- Perspective
  - The most commonly used projection
  - Points projected onto the image plane by dividing them by their z component

$$\text{inhomogeneous} \quad \bar{x} = \mathcal{P}_z(p) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \quad \text{homogeneous} \quad \tilde{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{p}$$

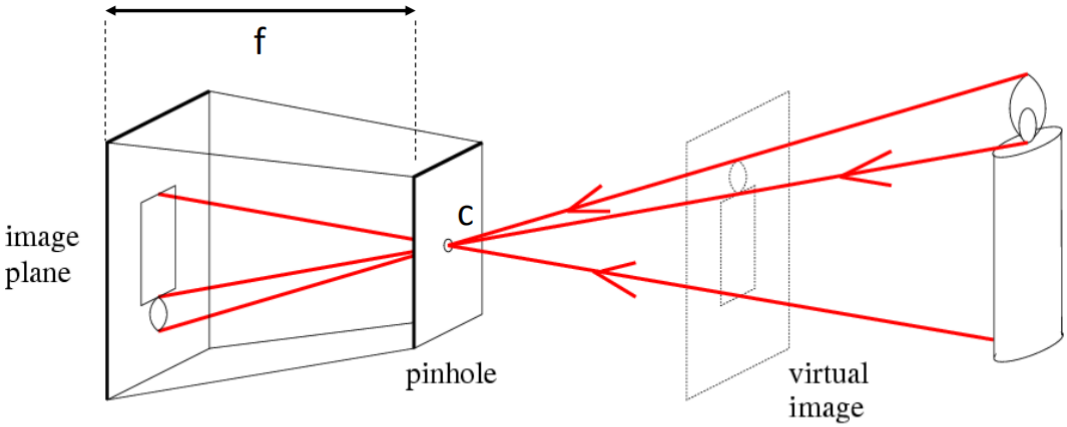
  - A two-step projection
    - ✓ First project 3D points into normalized device coordinates in the range
    - ✓ Then rescale these coordinates to integer pixel coordinates

the near and far z *clipping planes*

$$z_{\text{range}} = z_{\text{far}}^{1} - z_{\text{near}}^{0}$$

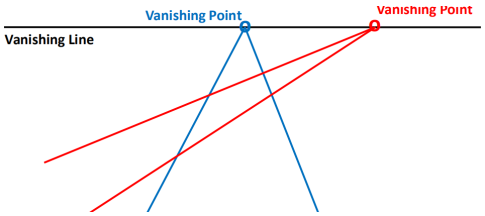
$$\tilde{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{\text{far}}/z_{\text{range}} & z_{\text{near}} z_{\text{far}}/z_{\text{range}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{p}$$

Pinhole camera model

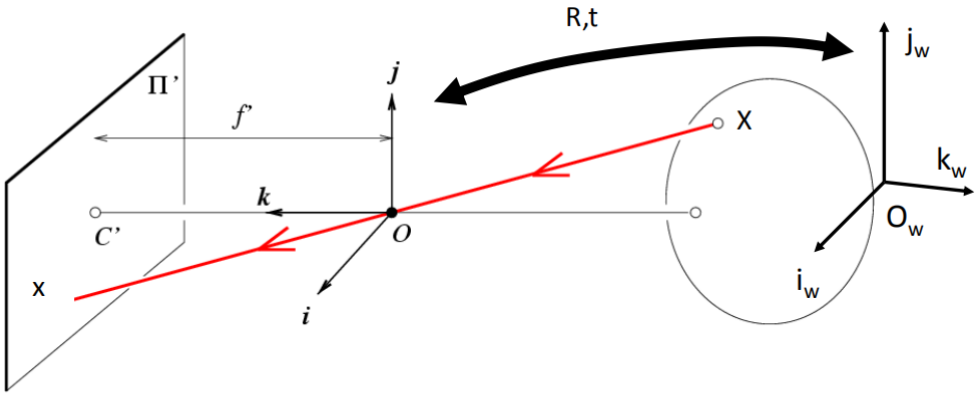


f = focal length
c = center of the camera

- What is lost?
  - Length
  - Angles
- What is preserved?
  - Straight lines are still straight



Camera projection matrix



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

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

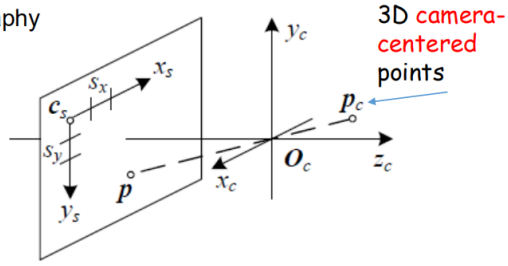
- Pixel values indexed by integer pixel coordinates
- Starting at the upper-left corner of the image
  - ✓ First scale the pixel values by the pixel spacing
  - ✓ Then describe the orientation of the sensor array relative to the camera projection center

the sensor planes at location

a sensor homography

$$p = \begin{bmatrix} R_s \mid c_s \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = M_s \bar{x}_s$$

3D rotation | origin | scale | integer pixel coordinates



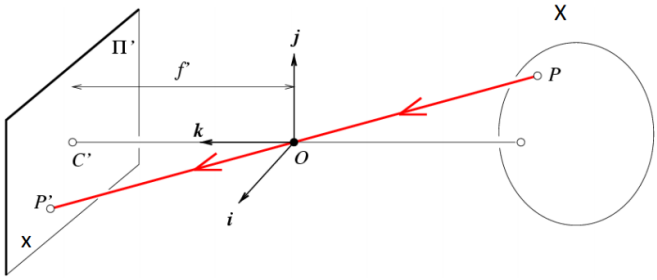
- The relationship between the 3D pixel center and the 3D camera-centered point is given by an unknown scaling s
  - ➢ The calibration matrix describes the camera intrinsics

$$p = sp_c \qquad \tilde{x}_s = sM_s^{-1}p_c = Kp_c$$

the sensor planes at location | 3D camera-centered points | pixel address | **calibration matrix**

Projection (Camera) matrix

- Intrinsic Assumptions
  - ➢ Unit aspect ratio
  - ➢ Optical center at (0,0)
  - ➢ No skew
- Extrinsic Assumptions
  - ➢ No rotation
  - ➢ Camera at (0,0,0)

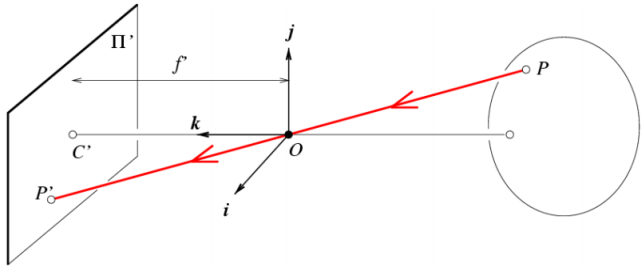


K

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

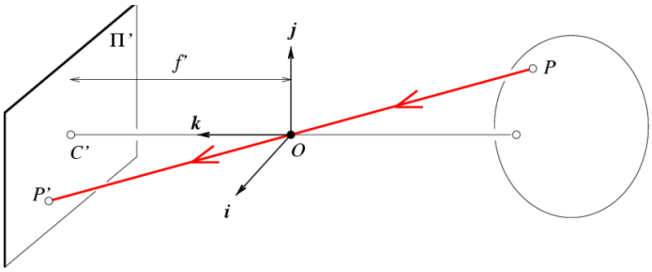
Perspective

- Intrinsic Assumptions
  - Unit aspect ratio
  -
  - No skew
- Extrinsic Assumptions
  - No rotation
  - Camera at (0,0,0)



$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f & 0 & u_0 & 0\\0 & f & v_0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

- Intrinsic Assumptions
  -
  -
  - No skew
- Extrinsic Assumptions
  - No rotation
  - Camera at (0,0,0)



$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}\mathbf{X} \Rightarrow w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}\alpha & 0 & u_0 & 0\\0 & \beta & v_0 & 0\\0 & 0 & 1 & 0\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix}$$

.....

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

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

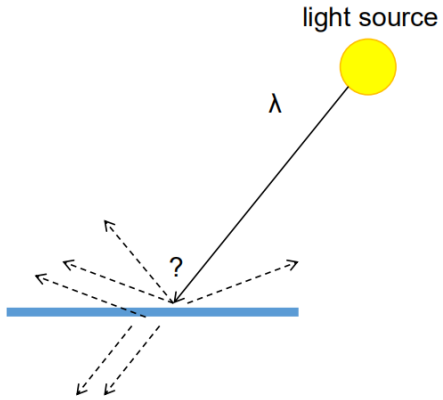
- Vanishing point = Projection from infinity

$$\mathbf{p} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\begin{bmatrix}x\\y\\z\\0\end{bmatrix} \Rightarrow \mathbf{p} = \mathbf{KR}\begin{bmatrix}x\\y\\z\end{bmatrix} \Rightarrow \mathbf{p} = \mathbf{K}\begin{bmatrix}x_R\\y_R\\z_R\end{bmatrix}$$

$$w\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f & 0 & u_0\\0 & f & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}x_R\\y_R\\z_R\end{bmatrix} \Rightarrow \quad u = \frac{fx_R}{z_R} + u_0 \qquad v = \frac{fy_R}{z_R} + v_0$$

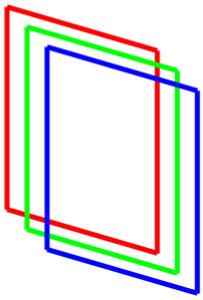
2.3 Photometric image formation

- Absorption 吸收
- Diffusion 漫射
- Reflection反射
- Transparency 透射
- Refraction 折射
- Fluorescence 荧光反应
- Subsurface scattering 次表面散射
- Phosphorescence 磷光
- Interreflection 相互反射



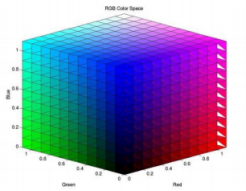
2.4 The digital camera

Image: three matrices



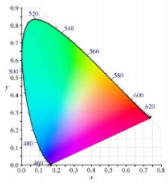
Color Spaces

- RGB



$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697} \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

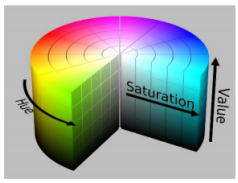
- CIE XYZ



Luminance

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}$$

- HSV
  - Hue
  - Saturation
  - Value



$$C = V \times S_{HSV}$$
$$H' = \frac{H}{60^\circ}$$
$$X = C \times (1 - |H' \mod 2 - 1|)$$
$$m = V - C$$

$$(R_1, G_1, B_1) = \begin{cases} (0,0,0) & \text{if } H \text{ is undefined} \\ (C,X,0) & \text{if } 0 \le H' \le 1 \\ (X,C,0) & \text{if } 1 < H' \le 2 \\ (0,C,X) & \text{if } 2 < H' \le 3 \\ (0,X,C) & \text{if } 3 < H' \le 4 \\ (X,0,C) & \text{if } 4 < H' \le 5 \\ (C,0,X) & \text{if } 5 < H' \le 6 \end{cases}$$

$$(R, G, B) = (R_1 + m, G_1 + m, B_1 + m)$$

- Color filter array layout
- Interpolated pixel values
  - The luminance signal is mostly determined by green values
  - The visual system is much more sensitive to high frequency detail in luminance than in chrominance

G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb

3. Image processing

- (a) original image
- (b) increased contrast
- (c) change in hue
- (d) quantized colors

- (e) blurred
- (f) rotated

3.1 Point Operation

- A general image processing operator
  - Continuous domain $g(\boldsymbol{x}) = h(f(\boldsymbol{x}))$ or $g(\boldsymbol{x}) = h(f_0(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$
  - Discrete images $g(\overset{\text{pixel}}{i,j}) = h(f(i,j))$ gain controls brightness
  - Multiplication and addition $g(\boldsymbol{x}) = af(\boldsymbol{x}) + b$ spatially varying
  - Dyadic (two-input) operator $g(\boldsymbol{x}) = (1-\alpha)f_0(\boldsymbol{x}) + \alpha f_1(\boldsymbol{x})$ linear blend operator
  - Gamma correction
    - ✓ Remove the non-linear mapping between input radiance and quantized pixel values

$$g(\boldsymbol{x}) = [f(\boldsymbol{x})]^{1/\gamma} \qquad \gamma \approx 2.2$$

pixel locations

• Brightening a picture by adding a constant value to all three channels==>increases the apparent intensity of each pixel, affect the pixel's hue and saturation.

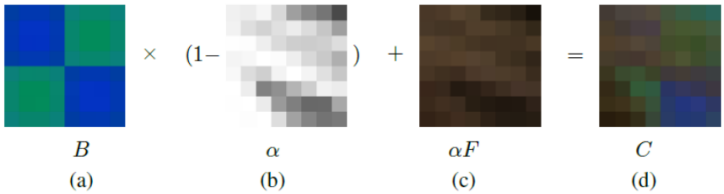
Some color ratio images multiplied by the middle gray value for better visualization

Compositing and Matting

- Matting: extracting the object from the original image
- Compositing: inserting one image into another image

$$C = (1-\alpha)B + \alpha F$$

(a) source image
(b) extracted foreground object
(c) alpha matte shown in grayscale
(d) new composite



Histogram equalization

• A technique for adjusting image **intensities** to enhance **contrast**

 find an intensity mapping function *f(I)* such that the resulting histogram is flat.

 An image would have a linearized cumulative distribution function (CDF)

$$c(I) = \frac{1}{N}\sum_{i=0}^{I} h(i) = c(I-1) + \frac{1}{N}h(I)$$

normalization

$$x^* = \frac{x - x_{min}}{x_{max} - x_{min}}$$

standardization

$$x^* = = \frac{x - \mu}{\sigma}$$

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} P_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j \qquad k = 1, 2, 3 \ldots L-1$$

s_k：目标像素值
r_k：原始像素值
L：灰度级（8位256）
$P_r(r_j)$：r_j在原始图中的概率
MN：图像的像素总数cols×rows
n_j：原始图像中，像素值为 j 的个数

Histogram Equalization

• Motivation

The motivation for this transformation comes from thinking of the intensities of f and g as continuous random variables X, Y on $[0, L-1]$ with Y defined by

$$Y = T(X) = (L-1)\int_0^X p_X(x)dx, \tag{2}$$

概率密度函数

分布函数

where p_X is the probability density function of f. T is the cumulative distributive function of X multiplied by $(L-1)$. Assume for simplicity that T is differentiable and invertible. It can then be shown that Y defined by $T(X)$ is uniformly distributed on $[0, L-1]$, namely that $p_Y(y) = \frac{1}{L-1}$.

$$\begin{aligned}\int_0^y p_Y(z)dz &= \text{probability that } 0 \le Y \le y \\ &= \text{probability that } 0 \le X \le T^{-1}(y) \\ &= \int_0^{T^{-1}(y)} p_X(w)dw\end{aligned}$$

$$\frac{d}{dy}\left(\int_0^y p_Y(z)dz\right) = p_Y(y) = p_X(T^{-1}(y))\frac{d}{dy}(T^{-1}(y)).$$

Note that $\frac{d}{dy}T(T^{-1}(y)) = \frac{d}{dy}y = 1$, so

$$\frac{dT}{dx}|_{x=T^{-1}(y)}\frac{d}{dy}(T^{-1}(y)) = (L-1)p_X(T^{-1}(y))\frac{d}{dy}(T^{-1}(y)) = 1,$$

which means $p_Y(y) = \frac{1}{L-1}$.

变上、下限积分求导公式

$$\frac{d}{dx}\int_a^{\varphi(x)} f(t)dt = f(\varphi(x))\cdot\varphi'(x)$$

$$\frac{d}{dx}\int_{\phi(x)}^{b} f(t)dt = -f(\phi(x))\cdot\phi'(x)$$

$$\frac{d}{dx}\int_{\phi(x)}^{\varphi(x)} f(t)dt = f(\varphi(x))\cdot\varphi'(x) - f(\phi(x))\cdot\phi'(x)$$

$$[f^{-1}(x)]' = \frac{1}{f'[f^{-1}(x)]}$$

e.g. （假定图像的灰度级范围是 [0, 9]）：

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

图 5 原始图像

第一步，计算原始图像的灰度直方图 n_k。

$n(0)=3$，即原始图像中灰度级为0的像素的个数是3，直接从图5中可以数出来。按这样的方式分别得出 $n(1), n(2), ..., n(9)$。这里我们可以得到 $n_k=[3,2,4,4,1,1,4,1,2,3]$。

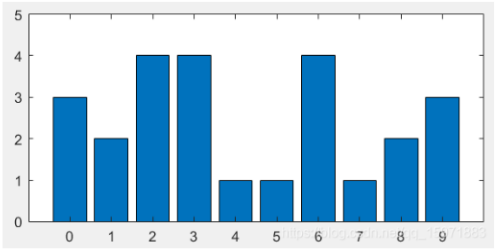


图 6 原始图像的灰度直方图

像素总个数：25

第三步，计算原始图像的灰度分布频率。

$$p_r(k)=n_k/N=[3/25,2/25,4/25,4/25,1/25,1/25,4/25,1/25,2/25,3/25],\ k=0,1,2,...,9$$

第四步，计算原始图像的灰度累积分布频率。

$$s_k=\sum_{i=0}^{k}\frac{n_i}{N}=[3/25,5/25,9/25,13/25,14/25,15/25,19/25,20/25,22/25,25/25],$$
$$k=0,1,2,...,9$$

第五步，将归一化的 s_k 乘以 $L-1$再四舍五入，以使得均衡化后图像的灰度级与归一化前的原始图像一致。

$s_0=\frac{3}{25}*(L-1)=\frac{3}{25}*9=1.08$，四舍五入之后其值为1，也就是说原始图像中灰度级0对应均衡化后的灰度级1，即0→1。

同理，$s_1=1.8$，四舍五入之后为2，即1→2；$s_2=3.24$，四舍五入之后为3，即2→3；$s_3=4.68$，四舍五入之后为5，即3→5；$s_4=5.04$，四舍五入之后为5，即4→5；$s_5=5.4$，四舍五入之后为5，即5→5；$s_6=6.84$，四舍五入之后为7，即6→7；$s_7=7.2$，四舍五入之后为7，即7→7；$s_8=7.92$，四舍五入之后为8，即8→8；$s_9=9$，四舍五入之后为9，即9→9。

以上的映射关系，就是变换函数 $T(r)$ 的作用。

e.g. （假定图像的灰度级范围是 [0, 9]）：

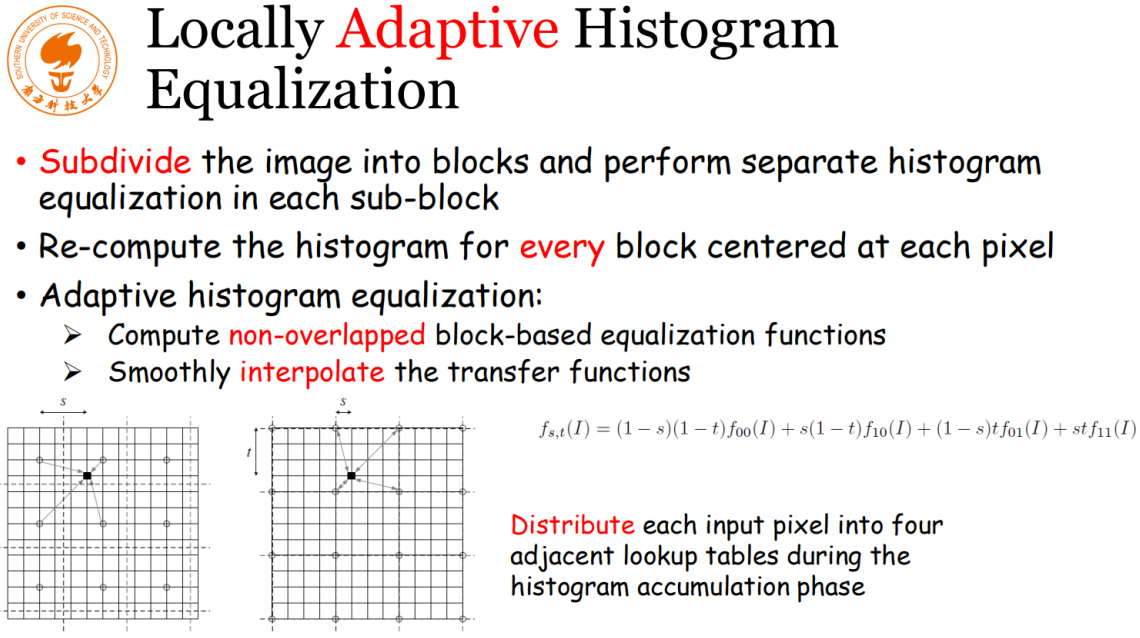
1	3	9	9	8
2	1	3	7	3
3	6	0	6	4
6	8	2	0	5
2	9	2	6	0

图 5 原始图像

第六步，根据以上映射关系，参照原始图像中的像素，可以写出直方图均衡化之后的图像，如图7所示。

2	5	9	9	8
3	2	5	7	5
5	7	1	7	5
7	8	3	1	5
3	9	3	7	1

图 7 直方图均衡化之后的图像



3.2 Linear filtering

The most commonly used type of neighborhood operator is a *linear filter*

Linear shift-invariant (LSI) operators

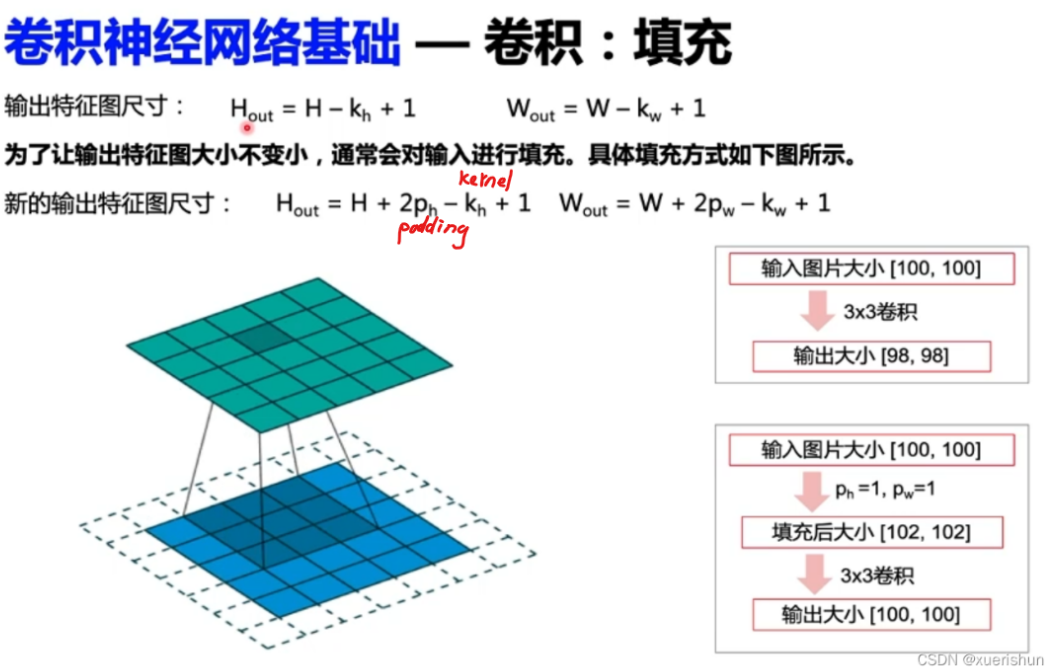
correlation operator: $g = f \otimes h$.

convolution operator: $g = f * h$ (中心对称)

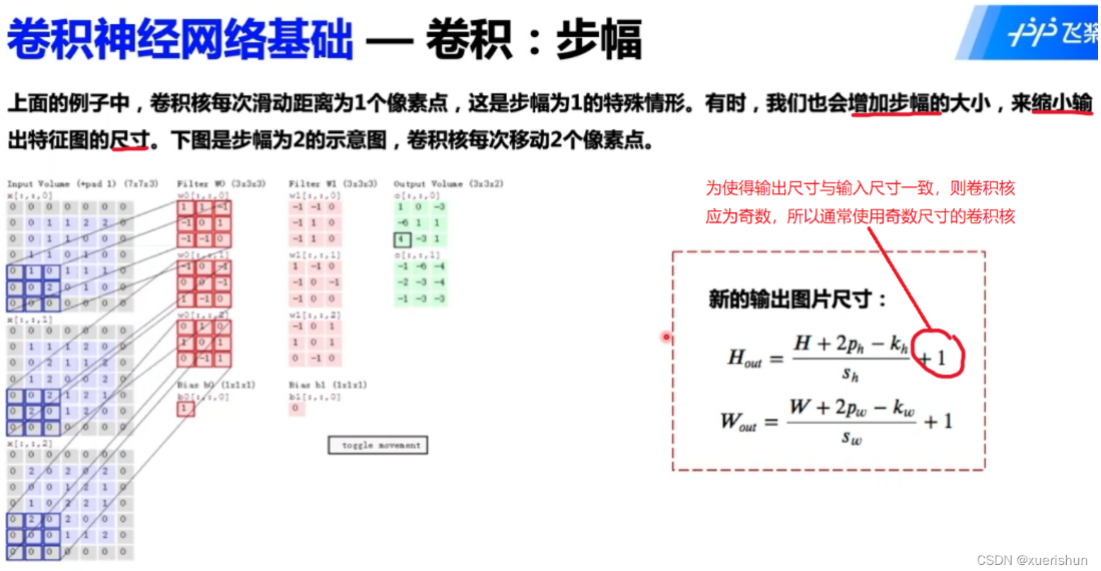
[h: kernel]

Padding

填充可改变卷积处理后图像尺寸的大小，当2P等于K-1时保持图像尺寸不变，**为抵消2P**则K应为奇数，所以通常**卷积核的尺寸为奇数**



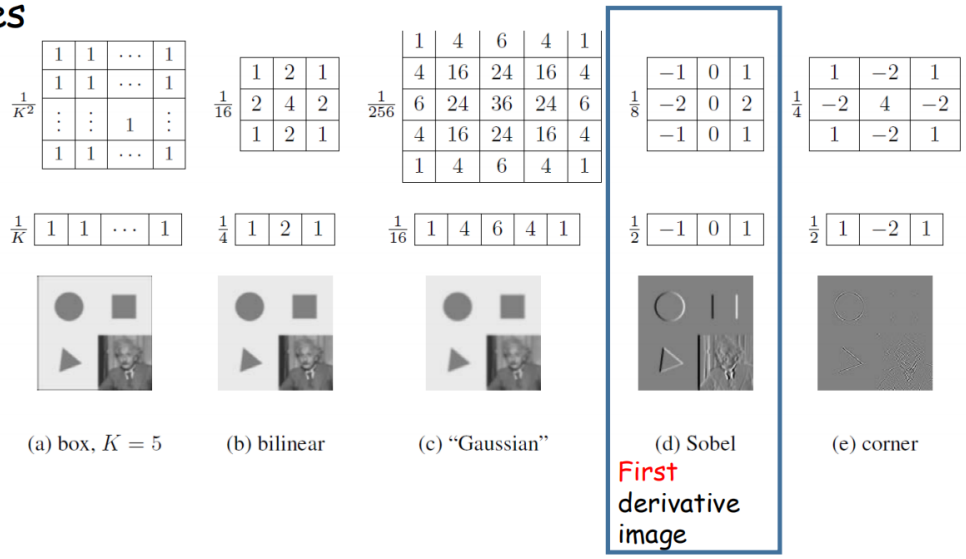
padding=kernel//2



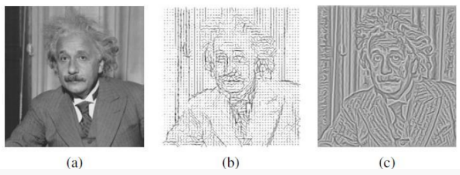
Separable Filtering

A convolution kernel whose operation can be significantly sped up by first performing a one-dimensional horizontal convolution followed by a one-dimensional vertical convolution(which requires a total of 2K operations per pixel)

- Examples

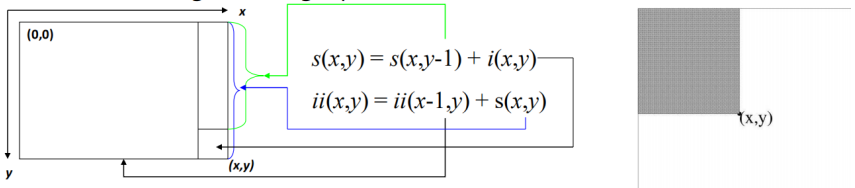


Band-pass and Steerable Filtering

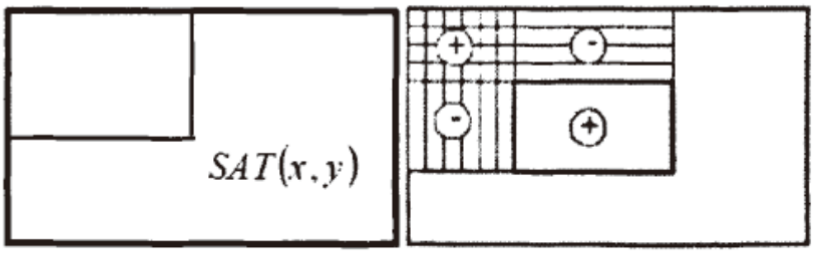
Band-Pass 带通滤波器	Steerable Filtering
Sobel and corner operators are simple examples of band-pass and oriented fifilters	Sobel operator is a simple approximation to a *directional* or *oriented* filter
Gaussian Smoothing +***Laplacian*** **operator**:[(undirected) second derivative of a two-dimensional image] = *Laplacian of Gaussian* (**LoG**) filter $\nabla^2 G(x,y;\sigma) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) G(x,y;\sigma),$ (a) original image of Einstein; (b) orientation map computed from the second-order oriented energy; (c) Original image with oriented structures enhanced.  (a) (b) (c)	***Gaussian*** Smoothing + ***directional derivative***
nice *scale-space properties*	construct both feature descriptors (Section 4.1.3) and edge detectors

Summed area table (integral image)

- Def: The *integral image* $ii(x,y)$ at location (x,y), is the sum of the pixel values above and to the left of (x,y), inclusive
  - Using the following two recurrences, where $i(x,y)$ is the pixel value of original image at the given location and $s(x,y)$ is the cumulative column sum, we can calculate the integral image representation of the image in a single pass



构造积分图像Summed Area Table

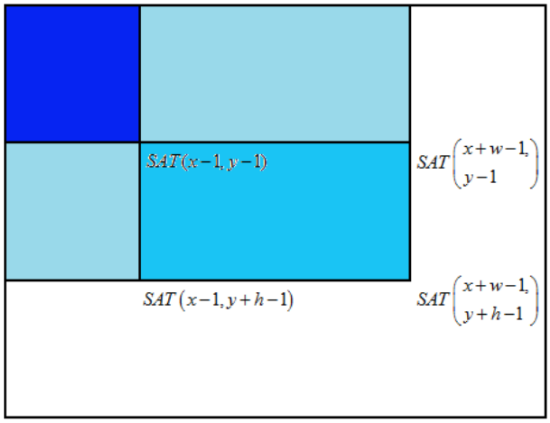


积分图像中，每个点存储是其左上方所有像素之和：

$$SAT(x, y) = \sum_{x' \le x, y' \le y} I(x', y')$$

其中I(x,y)表示图像(x,y)位置的像素值。
积分图像可以采用增量的方式计算：

$$SAT(x, y) = SAT(x, y-1) + SAT(x-1, y) + I(x, y) - SAT(x-1, y-1)$$



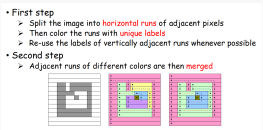
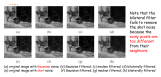
利用积分图计算可计算矩形区域内像素和：

$$\begin{aligned} RecSum(r) &= SAT(x-1, y-1) + SAT(x+w-1, y+h-1) \\ &\quad - SAT(x-1, y+h-1) - SAT(x+w-1, y-1) \end{aligned}$$

$SAT(x-1, y+h-1)$

所以，无论矩形r的尺寸大小，只需查找积分图像4次就可以求得任意矩形内像素值的和。

3.3 More neighborhood operators

Median Filtering	Bilateral Filtering	Morphology	Distance transforms
Select the median from neighborhood（Filter away shot noise, not as efficient at averaging away regular Gaussian noise)	It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels(can be based on a Gaussian distribution).	• Binary images ➢ First convolve the binary image with a binary structuring element ✓ It can be any shape, from a simple 3 × 3 box filter : $c = f \otimes s$ ➢ Then select a binary output value depending on the thresholded result of the convolution $\theta(f,t) = \begin{cases} 1 & \text{if } f \geq t, \\ 0 & \text{else,} \end{cases}$ • **dilation**: $\text{dilate}(f,s) = \theta(c,1)$; • **erosion**: $\text{erode}(f,s) = \theta(c,S)$; S number of pixels • **majority**: $\text{maj}(f,s) = \theta(c,S/2)$; • **opening**: $\text{open}(f,s) = \text{dilate}(\text{erode}(f,s),s)$; • **closing**: $\text{close}(f,s) = \text{erode}(\text{dilate}(f,s),s)$	• The distance transform ➢ Quickly pre-computing the distance to a curve or set of points ✓ Manhattan distance $d_1(k,l) = \lvert k \rvert + \lvert l \rvert$ ✓ Euclidean distance $d_2(k,l) = \sqrt{k^2 + l^2}$ • How to calculate distances to the nearest background pixel $D(i,j) = \min_{k,l \mid I(k,l)=0} d(i-k, j-l)$
α-trimmed mean (averages together all of the pixels except for the *α* fraction that are the smallest and the largest)	non-linear, edge-preserving		Connected Components • First step ➢ Split the image into horizontal runs of adjacent pixels ➢ Then color the runs with unique labels ➢ Re-use the labels of vertically adjacent runs whenever possible • Second step ➢ Adjacent runs of different colors are then merged 
weighted median (each pixel is used a number of times depending on its distance from the center. This turns out to be equivalent to minimizing the weighted objective function)			

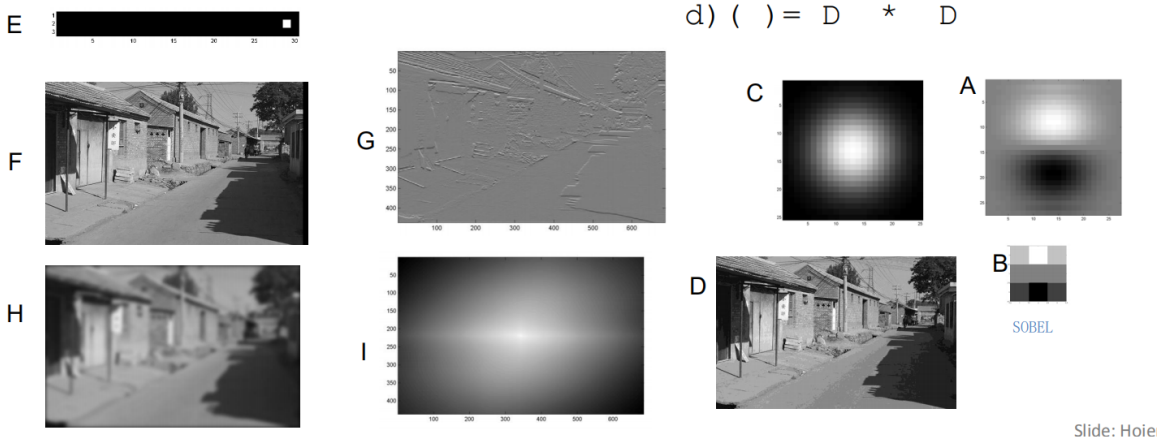
e.g.

Review

- Fill in the blanks

Filtering Operator

a) ( ) = D * B
b) A = ( ) * ( )
c) F = D * ( )
d) ( ) = D * D

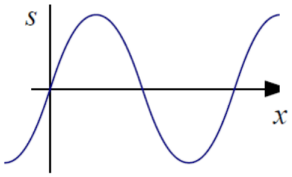


Slide: Hoie

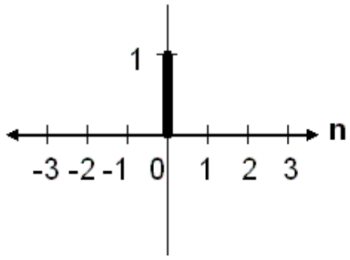
3.4 Thinking in Frequency

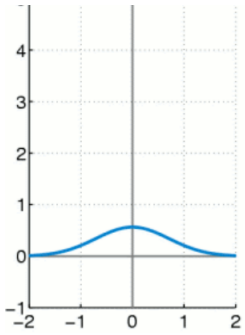
$$s(x) = \sin(2\pi f x + \phi_i) = \sin(\omega x + \phi_i)$$

frequency — angular frequency — phase



- Impulse response: h(x)
  - Continuous function
  - Discrete function





  - The impulse function contains all frequencies
- If we convolve the signal with a filter whose impulse response is h(x), we get another sinusoid of the same frequency but different magnitude and phase

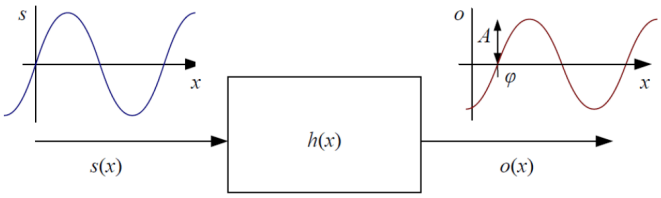
$$o(x) = h(x) * s(x) = A\sin(\omega x + \phi_o)$$

- Convolution

$$g(\boldsymbol{x}) = \int f(\boldsymbol{x} - \boldsymbol{u})h(\boldsymbol{u})d\boldsymbol{u}.$$

  - A weighted summation of shifted input signals (sinusoids)
  - The summation of a bunch of shifted sinusoids of the same frequency is just a single sinusoid at that frequency (same)

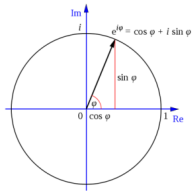
$$o(x) = h(x) * s(x) = A\sin(\omega x + \phi_o)$$



$e^{ix} = cosx + isinx$

- The complex-valued sinusoid is

$$s(x) = e^{j\omega x} = \cos\omega x + j\sin\omega x$$



- The filtered sinusoid is

$$o(x) = h(x) * s(x) = Ae^{j\omega x + \phi}$$

- The Fourier transform in continuous domain

One frequency

$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-j\omega x}dx$$

在该方向投影的强度

- The transform in discrete domain

length of the signal

$$H(k) = \frac{1}{N}\sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}}$$

3.5 Pyramids

interpolation

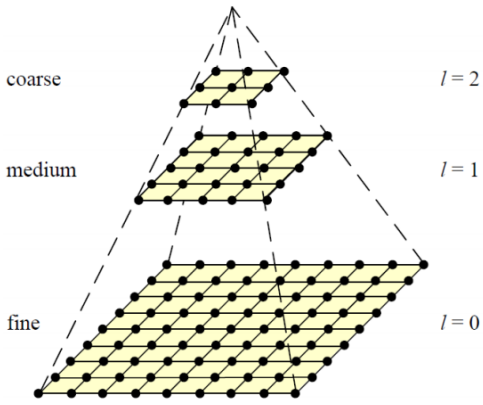
Select some interpolation kernels to convolve the image

decimation

- Octave pyramids
  - Can be used to accelerate coarse-to-fine search algorithms

$$\frac{1}{16}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 6 & 4 & 1 \\ \hline\end{array}$$

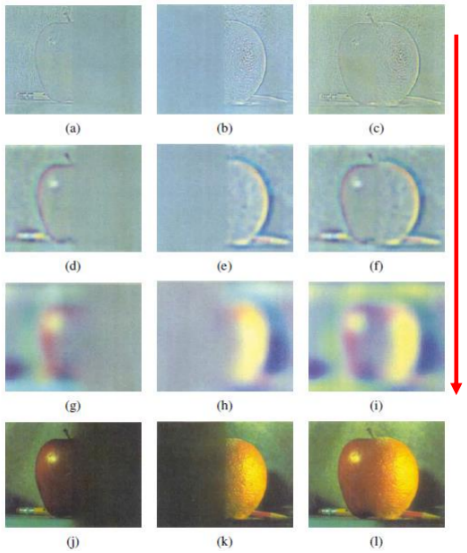
A traditional image pyramid



multi-resolution

image blending

- Steps:
  - Each source image is first decomposed into its own Laplacian pyramid
  - Each band is then multiplied by a smooth weighting function
    - ✓ Create these weights is to take a binary mask image
  - Each Laplacian pyramid image is then multiplied by its corresponding Gaussian mask
  - The sum of these two weighted pyramids is then used to construct the final image



3.6 Geometric transformations

Parametric transforms

forward Warp

- Limitations:
  - Target location has a non-integer value
    - ✓ round the value
    - ✓ "distribute" the value
  - The second major problem with forward warping is the appearance of cracks and holes

procedure *forwardWarp*$(f, \boldsymbol{h}, \textbf{out}\ g)$:

For every pixel $\boldsymbol{x}$ in $f(\boldsymbol{x})$

1. Compute the destination location $\boldsymbol{x}' = \boldsymbol{h}(\boldsymbol{x})$.
2. Copy the pixel $f(\boldsymbol{x})$ to $g(\boldsymbol{x}')$.

inverseWarp

- Resampling an image at non-integer locations is a well-studied problem
- High-quality filters that control aliasing can be used

procedure *inverseWarp*$(f, \boldsymbol{h}, \textbf{out}\ g)$:

For every pixel $\boldsymbol{x}'$ in $g(\boldsymbol{x}')$

1. Compute the source location $\boldsymbol{x} = \hat{\boldsymbol{h}}(\boldsymbol{x}')$
2. Resample $f(\boldsymbol{x})$ at location $\boldsymbol{x}$ and copy to $g(\boldsymbol{x}')$

Mesh-based warping

4. Feature detection and matching

keyPoints: Repeatability, Saliency, Compactness and efficiency, Locality

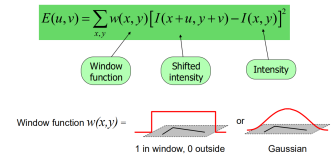
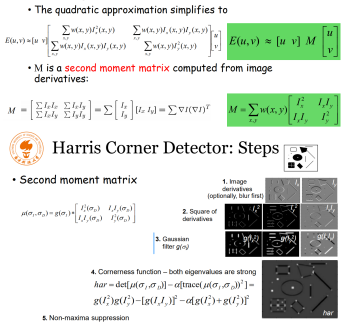
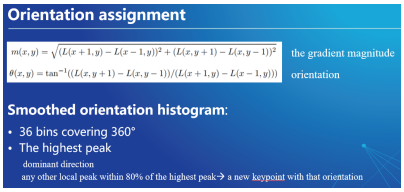
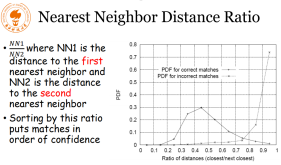
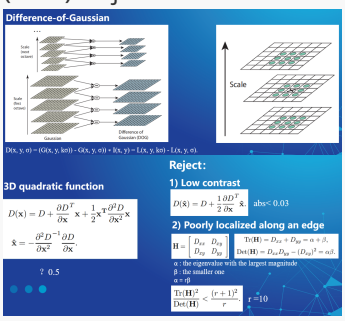
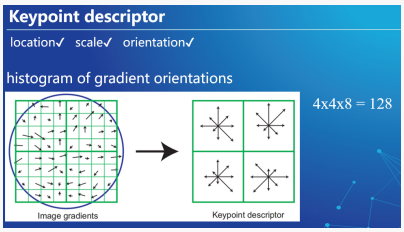
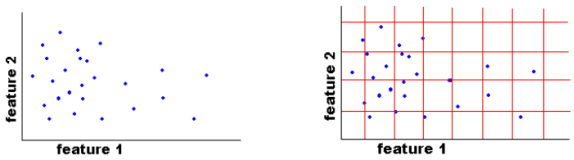
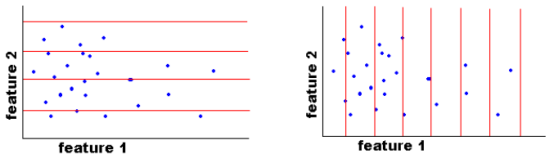
Point detection	Point decriptor	Point Matching
Harris Corner detection [Precise localization in x-y]  Denote the change around E(u,v) [E(0,0)=0 denoting identity]  	**SIFT** Histogram of Oriented Gradients  	nearest neighbor 
Difference-of-Gaussian [Good localization in scale] (DoG)+rejection  	• 8 orientations x 4x4 array = 128 dimensions  	
Affifine invariance		

***image representation:**

- Histogram: probability or count of data in each bin
- Joint histogram
  - Requires lots of data
  - Loss of resolution to avoid empty bins



- Marginal histogram
  - Require independent features
  - More data/bin than joint histogram



5. Fitting

Building a Model for a Set of Features

□ Noise

□ Extraneous data: clutter (outliers), multiple lines

□ Missing data: occlusions

• If we know which points belong to the line, how do we find the “optimal” line parameters?

  □ Least squares

• What if there are outliers?

  □ Robust fitting, RANSAC

• What if there are many lines?

  □ Voting methods: RANSAC, Hough transform

• What if we’re not even sure it’s a line?

  □ Model selection

5.1 Least Squares

$$\bar{A}^{\mathrm{T}}A\hat{x} = A^{\mathrm{T}}\hat{b}.$$

Look at a line

$$C + Dt = b.$$

If there is no experimental error, then two measurements of b will determine the line.

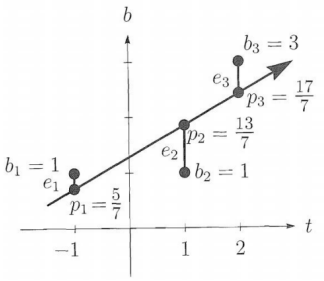
But if there is error, by a series of experiments, we obtain a system with two unknowns C, D:

$$\begin{cases} C + Dt_1 = b_1, \\ C + Dt_2 = b_2, \\ \cdots \\ C + Dt_m = b_m. \end{cases}$$

In matrix form $\boldsymbol{Ax} = \boldsymbol{b}$, where

$$\boldsymbol{A} = \begin{bmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \vdots \\ 1 & t_m \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} C \\ D \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Example 3 Three measurements b_1, b_2, b_3 are marked on the figure:



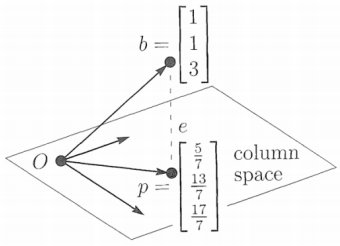
$b = 1$ at $t = -1$; $b = 1$ at $t = 1$; $b = 3$ at $t = 2$.

$$\begin{bmatrix} 1 & -1 \\ 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 3 \end{bmatrix}.$$

$\boldsymbol{Ax} = \boldsymbol{b}$ can't be solved because the points are not on a line.

Therefore they are solved by least squares:

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\hat{\boldsymbol{x}} = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{b}.$$



is $\begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix} \begin{bmatrix} C \\ D \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$ ➔ $\hat{C} = \frac{9}{7}$, $\hat{D} = \frac{4}{7}$.

The best line is $\frac{9}{7} + \frac{4}{7}t$.

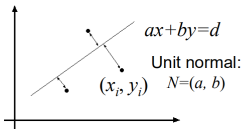
Remark. The mathematics of least squares is not limited to fitting the data by straight lines– *nonlinear least squares*.

Total Least Squares

- Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$:

$$|ax_i + by_i - d|$$

$$E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$$



$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)$$

U N

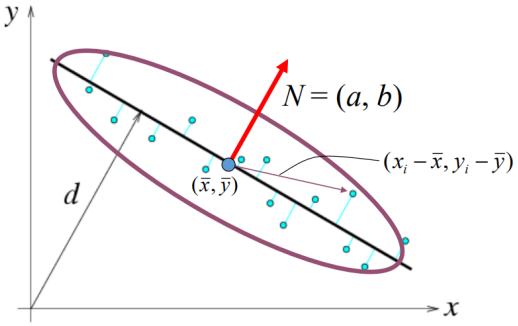
$$\frac{dE}{dN} = 2(U^T U)N = 0$$

- Solution to $(U^TU)N = 0$, subject to $||N||^2 = 1$: eigenvector of U^TU associated with the smallest eigenvalue (least squares solution to *homogeneous linear system* $UN = 0$)

- Second moment matrix

$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix}$$

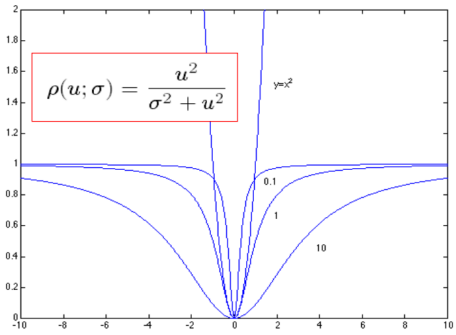
$$U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{bmatrix}$$



Robust Estimators

- General approach---minimize: $r_i(x_i, \theta)$ – residual of ith point w.r.t. model parameters θ
  ρ – robust function with scale parameter σ
- The robust function ρ behaves like squared distance for small values of the residual u but saturates for larger values of u

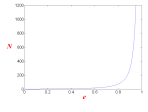
$$\sum_i \rho(r_i(x_i, \theta); \sigma)$$



!

5.2 RANSAC

- Repeat N times:
  - Draw s points uniformly at random
  - Fit line to these s points (TLS)
  - Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than t)
  - If there are d or more inliers, accept the line and refit using all inliers
- End
- Four parameters: s, t, d and N

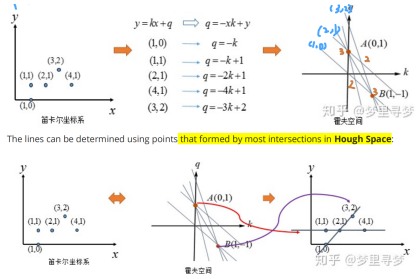
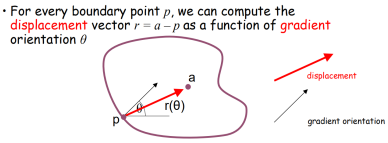
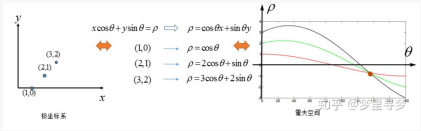
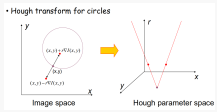
s	t	d	N
minimum: 2	➢ (1) Choose t so probability for inlier is p (e.g. 0.95) ➢ (2) Zero-mean Gaussian noise with standard deviation σ: $t = 3.84\sigma$	➢ Should match expected inlier ratio $(1-(1-e)^s) = 1-p$ $N = \log(1-p)/\log(1-(1-e)^s)$ 	➢ Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) Desired success rate after N times: p Outlier ratio (Unknown): e $(1-(1-e)^s) = 1-p$ $N = \log(1-p)/\log(1-(1-e)^s)$ proportion of outliers e s 5% 10% 20% 25% 30% 40% 50% 2 2 3 5 6 7 11 17 3 3 4 7 9 11 19 35 4 3 5 9 13 17 34 72 5 4 6 12 17 26 57 146 6 4 7 16 24 37 97 293 7 4 8 20 33 54 163 588 8 5 9 26 44 78 272 1177

- Inlier ratio e is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield $e=0.2$
- Adaptive procedure:
  - $N=\infty$, $sample\_count=0$
  - While $N > sample\_count$
    - ✓ Choose a sample (fitting) and count the number of inliers
    - ✓ Set e = 1 – (number of inliers)/(total number of points)
    - ✓ Recompute N from e:

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

    - ✓ Increment the $sample\_count$ by 1

5.3 Hough Voting

Line	Circle	Generalized
Cartesion 	Known radius: 2D Hough Space	 voting for each edge point p by r(θ)
Polar 	Unknown radius: 3D Hough Space 	

5.4 Image Alignment

fitting a model to a transformation between pairs of features (matches) in two images[previosly, one image]

Direct (pixel-based) alignment

Feature-based alignment

- Extract features
- Compute *putative* matches
- Loop:
  - *Hypothesize* transformation T
  - *Verify* transformation (search for other matches consistent with T)

6. Machine Learning

- Taxonomy

	Supervised Learning	Unsupervised Learning
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

6.1 Dimensionality Reduction

Linear

PCA

Find projections that capture the largest amounts of variation in data

Find the eigenvectors of the covariance matrix, and these eigenvectors define the new space

- Given a set of data $X \in R^{d \times N}$, find the principal axes are those orthonormal (正交) axes onto which the variance retained under projection is maximal

one dim	two dim	three dim
• Variance = Standard deviation^2 $s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}$	PCA: Now Consider Two Dimensions • Covariance: measures the correlation between X and Y • $cov(X,Y)=0$: independent • $cov(X,Y)>0$: move same direction • $cov(X,Y)<0$: move opposition direction $cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})$	• Contains covariance values between all possible dimensions (=attributes): $C^{m \times n} = (c_{ij} \mid c_{ij} = \mathrm{cov}(Dim_i, Dim_j))$ • Example for three attributes (x,y,z): $S = \begin{pmatrix} \mathrm{cov}(x,x) & \mathrm{cov}(x,y) & \mathrm{cov}(x,z) \\ \mathrm{cov}(y,x) & \mathrm{cov}(y,y) & \mathrm{cov}(y,z) \\ \mathrm{cov}(z,x) & \mathrm{cov}(z,y) & \mathrm{cov}(z,z) \end{pmatrix}$

X=Temperature	Y=Humidity
40	90
40	90
40	90
30	90
15	70
15	70
15	70
30	90
15	70
30	70
30	70
30	90
40	70
30	90

令 P 是这组基按行组成的矩阵，Y是 X 对 P 做基变换后的数据，记 Y 的协方差矩阵为 D，则：

$$\begin{aligned} D &= \frac{1}{m}YY^{\top} \\ &= \frac{1}{m}(PX)(PX)^{\top} \\ &= \frac{1}{m}PXX^{\top}P^{\top} \\ &= P(\frac{1}{m}XX^{\top})P^{\top} \\ &= PCP^{\top} \end{aligned}$$

算法步骤：

设有 m 条 n 维数据。

1）将原始数据按列组成 n 行 m 列矩阵 X

2）将 X 的每一行（代表一个属性字段）进行零均值化(即减去这一行的均值)

3）求出协方差矩阵 C=(1/m)*(XXT)

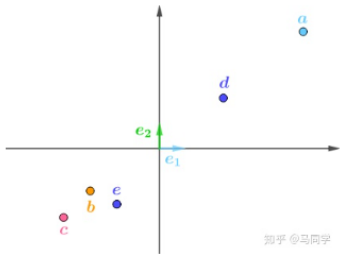
4）求出协方差矩阵的特征值及对应的特征向量

5）将特征向量按对应特征值大小从上到下按行排列成矩阵，取前 k 行组成矩阵 P

6）Y=PX 即为降维到 k 维后的数据

	房价(百万元)	面积(百平米)
a	5.4	4.4
b	−2.6	−1.6
c	−3.6	−2.6
d	2.4	1.9
e	−1.6	−2.1

这些数据按行，在自然基下画出来就是：



按列解读得到两个向量：

$$\boldsymbol{X}=\begin{pmatrix}5.4\\-2.6\\-3.6\\2.4\\-1.6\end{pmatrix}\quad \boldsymbol{Y}=\begin{pmatrix}4.4\\-1.6\\-2.6\\1.9\\-2.1\end{pmatrix}$$

组成协方差矩阵：

$$Q=\begin{pmatrix}Var(X) & Cov(X,Y)\\ Cov(X,Y) & Var(Y)\end{pmatrix}=\frac{1}{5}\begin{pmatrix}X\cdot X & X\cdot Y\\ X\cdot Y & Y\cdot Y\end{pmatrix}=\frac{1}{5}\begin{pmatrix}57.2 & 45.2\\ 45.2 & 36.7\end{pmatrix}$$

进行奇异值分解：

$$Q\approx\begin{pmatrix}-0.78 & -0.62\\ -0.62 & 0.78\end{pmatrix}\begin{pmatrix}18.66 & 0\\ 0 & 0.12\end{pmatrix}\begin{pmatrix}-0.78 & -0.62\\ -0.62 & 0.78\end{pmatrix}$$

根据之前的分析，主元1应该匹配最大奇异值对应的奇异向量，主元2匹配最小奇异值对应的奇异向量，即：

$$\boldsymbol{e}_1=\begin{pmatrix}-0.78\\-0.62\end{pmatrix}\quad \boldsymbol{e}_2=\begin{pmatrix}-0.62\\0.78\end{pmatrix}$$

Gram矩阵简介

gram矩阵是计算每个通道 i 的feature map与每个通道 j 的feature map 的内积
gram matrix的每个值可以说是代表 i 通道的feature map和 j 通道的 feature map的互相关程度。
参考博客

$$\mathrm{G}=\mathrm{A}^{\mathrm{T}}\mathrm{A}=\begin{bmatrix}\mathbf{a}_1^{\mathrm{T}}\\ \mathbf{a}_2^{\mathrm{T}}\\ \vdots\\ \mathbf{a}_{\mathrm{n}}^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_{\mathrm{n}}\end{bmatrix}=\begin{bmatrix}\mathbf{a}_1^{\mathrm{T}}\mathbf{a}_1 & \mathbf{a}_1^{\mathrm{T}}\mathbf{a}_2 & \cdots & \mathbf{a}_1^{\mathrm{T}}\mathbf{a}_{\mathrm{n}}\\ \mathbf{a}_2^{\mathrm{T}}\mathbf{a}_1 & \mathbf{a}_2^{\mathrm{T}}\mathbf{a}_2 & \cdots & \mathbf{a}_2^{\mathrm{T}}\mathbf{a}_{\mathrm{n}}\\ & & & \\ \mathbf{a}_{\mathrm{n}}^{\mathrm{T}}\mathbf{a}_1 & \mathbf{a}_{\mathrm{n}}^{\mathrm{T}}\mathbf{a}_2 & \cdots & \mathbf{a}_{\mathrm{n}}^{\mathrm{T}}\mathbf{a}_{\mathrm{n}}\end{bmatrix}$$

上面的$\mathbf{a}_{\mathrm{i}}$均为列向量，$\mathrm{i}=1,2...\mathrm{n}$

对于上面的矩阵，就是一个矩阵自己的转置乘以自己。

协方差矩阵

建议先看懂这一篇：一文读懂 协方差矩阵
而**协方差矩阵的求解步骤**是：
1.对于$\mathrm{X}_{\mathrm{n}\times\mathrm{d}}$,先求每一列的均值
2.然后对应列减去此列的均值，得矩阵$\mathrm{C}_{\mathrm{n}\times\mathrm{d}}$
3.$\mathrm{COV}_{\mathrm{d}\times\mathrm{d}}=\frac{1}{\mathrm{n}-1}\mathrm{C}_{\mathrm{n}\times\mathrm{d}}^{\mathrm{T}}\mathrm{C}_{\mathrm{n}\times\mathrm{d}}=\frac{1}{\mathrm{n}-1}(\mathrm{C}^{\mathrm{T}})_{\mathrm{d}\times\mathrm{n}}\mathrm{C}_{\mathrm{n}\times\mathrm{d}}$
也就是说协方差矩阵是先求均值，然后减去均值(作了一个中心化处理，**白化**)，再求协方差矩阵(除以$\frac{1}{\mathrm{n}-1}$,即进行了**标准化**)

Definition 1 Let $\boldsymbol{A}$ be a *square matrix* of degree n.

If there exist a non-zero vector $\boldsymbol{x}$ and a scalar λ such that

$$\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x},$$

then λ is called an **eigenvalue (特征值)** of $\boldsymbol{A}$, and $\boldsymbol{x}$ is called an **eigenvector (特征向量)**, *corresponding to* the eigenvalue λ.

Example 1 Let $\boldsymbol{A} = \begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix}$. Then

$$|\boldsymbol{A} - \lambda\boldsymbol{I}| = \begin{vmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{vmatrix} = \lambda^2 - \lambda - 2.$$

The roots are $\lambda_1 = -1$ and $\lambda_2 = 2$. Eigenvectors of $\boldsymbol{A}$ can be obtained as follows.

For $\lambda_1 = -1$, we solve the system of linear equations $(\boldsymbol{A} - \lambda_1\boldsymbol{I})\boldsymbol{x} = (\boldsymbol{A} + \boldsymbol{I})\boldsymbol{x} = \boldsymbol{0}$, i.e.,

$$\begin{bmatrix} 5 & -5 \\ 2 & -2 \end{bmatrix}\boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \quad \boldsymbol{x} \neq \boldsymbol{0}$$

It follows that $\boldsymbol{x}_1 = k_1(1,1)^{\mathrm{T}}$ ($k_1 \in \mathbf{R}$, $k_1 \neq 0$) are eigenvectors corresponding to the eigenvalue $\lambda_1 = -1$.

For $\lambda_2 = 2$, we solve the system of linear equations $(\boldsymbol{A} - \lambda_2\boldsymbol{I})\boldsymbol{x} = (\boldsymbol{A} - 2\boldsymbol{I})\boldsymbol{x} = \boldsymbol{0}$, i.e.,

$$\begin{bmatrix} 2 & -5 \\ 2 & -5 \end{bmatrix}\boldsymbol{x} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

It follows that $\boldsymbol{x}_2 = k_2(5,2)^{\mathrm{T}}$ ($k_2 \in \mathbf{R}$, $k_2 \neq 0$) are eigenvectors corresponding to the eigenvalue $\lambda_2 = 2$.

Kernel

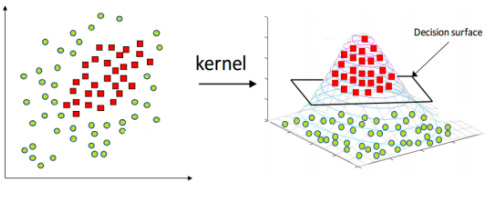
$k(x,y) = <\Phi(x), \Phi(y)>$ $\quad \Phi : x \to \mathcal{H}$ $\quad x \mapsto \Phi(x)$

- Must be continuous, symmetric, and most preferably should have a positive (semi-) definite Gram matrix
- Kernel Functions
  - Linear Kernel
  - Polynomial Kernel
  - Gaussian Kernel

$k(x,y) = x^T y + c$

$k(x,y) = (\alpha x^T y + c)^d$

$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$ 高斯核函数



Example 3 (SVD) Let $\boldsymbol{A} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}$.

Then $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$, $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$.

The eigenvalues of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ are 3 and 1, which are also the non-zero eigenvalues of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. So $\sigma_1 = \sqrt{3},\ \sigma_2 = 1$.

Finding three eigenvectors of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ gives rise to

$$\boldsymbol{V} = \begin{bmatrix} \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ -\frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \end{bmatrix} = [\boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \boldsymbol{v}_3].$$

Two unit eigenvectors of $\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}$ are

$$\boldsymbol{u}_1 = \frac{\boldsymbol{A}\boldsymbol{v}_1}{\sigma_1} = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \boldsymbol{u}_2 = \frac{\boldsymbol{A}\boldsymbol{v}_2}{\sigma_2} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Thus $\boldsymbol{U} = \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}$.

Now $\boldsymbol{\Sigma}$ is a (2×3)-matrix $\boldsymbol{\Sigma} = \begin{bmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$, it is easy to get that

$$\boldsymbol{A} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathrm{T}} = \frac{1}{\sqrt{2}}\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} \sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix}.$$

MDS

- Steps of a Classical MDS algorithm:
  - Set up the squared proximity matrix
  - Apply double centering $-\frac{1}{2} H D^{(X)} H$
  - Determine the largest k eigenvalues and corresponding eigenvectors
  - The original coordinate is $X = \Lambda^{1/2} V'$, if we have had
  - The NEW coordinate is $X_k = \Lambda_k^{1/2} V_k'$

Nonlinear(Manifold Learning)

LLE

- Procedure
  - Identify the neighbors of each data point
  - Compute weights that best linearly reconstruct the point from its neighbors

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \|\mathbf{x}_i - \sum_{j=1}^{k} w_{ij}\mathbf{x}_{N_i(j)}\|^2$$

Locally

  - Find the low-dimensional embedding vector which is best reconstructed by the weights determined in Step 2

$$\min_{Y} \sum_{i=1}^{N} \|\mathbf{y}_i - \sum_{j=1}^{k} w_{ij}\mathbf{y}_{N_i(j)}\|^2 \iff \min_{Y} \operatorname{tr}(Y^\top Y L)$$

Centering Y with unit variance

where $L = R - W$, R is diagonal and $R_{ii} = \sum_{j=1}^{N} W_{ij}$.

LE

- Similar to locally linear embedding
- Different in weights setting and objective function
  - Weights

$$W_{ij} = \begin{cases} 1 & i,j \text{ are connected} \\ \exp\left(\frac{-\|x_i - x_j\|^2}{s}\right) & \text{otherwise} \end{cases}$$

Locally

Has a different meaning to the weights in LLE

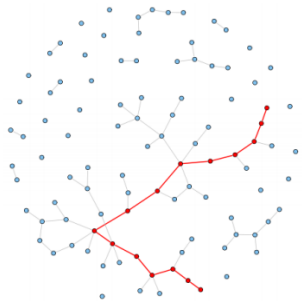
  - Objective

$$\min_{Y} \sum_{i=1}^{N}\sum_{j=1}^{N} (\mathbf{y}_i - \mathbf{y}_j)^2 W_{ij} \iff \min_{Y} \operatorname{tr}(YLY^\top)$$

where $L = R - W$, R is diagonal and $R_{ii} = \sum_{j=1}^{N} W_{ij}$.

Isomap

- Procedure-similar to LLE
  - Find neighbors of each data point - graph
  - Compute geodesic pairwise distances (e.g., shortest path distance) between all points
  - Embed the data via MDS



T-SNE

t-SNE (t-distributed stochastic neighbor embedding)

1）**SNE降维算法**：利用条件概率来衡量数据点之间的相似性，通过最小化条件概率 pj|i 与 pi|j 之间的 KL-divergence，将数据从高维空间映射到低维空间。

2）**symmetric SNE**：使用联合概率分布来替换条件概率分布

3）**t-SNE**：在低维空间下使用更重长尾分布的t分布来避免crowding问题和优化困难的问题。

$$p_{ij} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma^2)}{\sum_{k\neq l}\exp(-\|x_k - x_l\|^2/2\sigma^2)}$$

$$q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|y_k - y_l\|^2)^{-1}}$$

$$\frac{\partial C}{\partial y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1+\|y_i - y_j\|^2)^{-1}$$

KL散度：

$$C = \sum_i KL(P_i\|Q_i) = \sum_i \sum_j p_{j|i}\log\frac{p_{j|i}}{q_{j|i}}$$

困惑度（求方差）：

$$H(P_i) = -\sum_j p_{j|i}\log_2 p_{j|i} \qquad Perp(P_i) = 2^{H(P_i)}$$

- The similarity of data point x_j to data point x_i is the conditional probability: $p_{j|i}$

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$$

The relationships are only related to point i

- For the low-dimensional counterparts, a similar conditional probability is defined as: $q_{j|i}$

$$q_{j|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k\neq i}\exp(-\|y_i - y_k\|^2)}$$

Symmetric SNE

What is preserved? Similarity distribution

- In symmetric SNE, the pairwise similarities in the low-dimensional map is:

$$q_{ij} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k\neq l}\exp(-\|y_k - y_l\|^2)}$$

All points

- The pairwise similarities in the high-dimensional space is:

$$p_{ij} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma^2)}{\sum_{k\neq l}\exp(-\|x_k - x_l\|^2/2\sigma^2)}$$

- The gradient of symmetric SNE is fairly similar to that of asymmetric SNE

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

- SNE minimizes the sum of Kullback-Leibler divergences over all data points using a gradient descent method. The cost function C is given by

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i}\log\frac{p_{j|i}}{q_{j|i}}$$

  - P_i: conditional probability distribution over all others given x_i
  - Q_i: conditional probability distribution over all other map points given map point y_i

- The gradient has a surprisingly simple form

$$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

T-distributed Stochastic Neighbor Embedding (T-SNE)

•Data: X=x1,...,xn

•计算cost function的参数：困惑度Perp

•优化参数: 设置迭代次数T， 学习速率η, 动量α(t)

•目标结果是低维数据表示 YT=y1,...,yn

•开始优化

计算在给定Perp下的条件概率pj | i(参见上面公式)

令 pij=pj | i+pi | j2n

用 N(0,10−4I) 随机初始化 Y

迭代，从 t = 1 到 T， 做如下操作:

-计算低维度下的 qij(参见上面的公式)

-计算梯度（参见上面的公式）

-更新 Yt=Yt−1+ηdCdY+α(t)(Yt−1−Yt−2)

结束

•结束

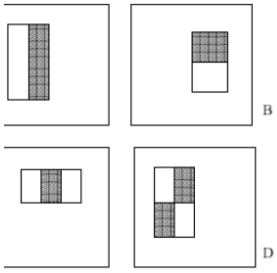
7. Object Detection

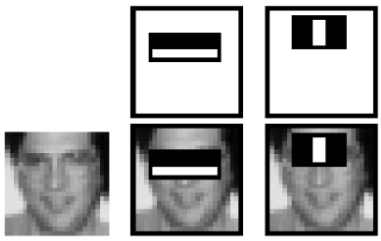
The process of finding and classifying objects in an image.

7.1 Viola/Jones detector

Rectangle features

- "Rectangle filters": $Value = \sum (pixels\ in\ white\ area) - \sum (pixels\ in\ black\ area)$
  - They are easy to calculate
  - The white areas are subtracted from the black ones
  - A special representation of the sample called the **integral image** makes feature extraction faster





- *two-rectangle feature* type
- *three-rectangle feature* type
- *four-rectangle feature* type

Integral images for fast computation

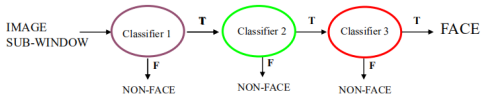
3.2

Boosting for feature selection

- Initially, weight each training example equally
- In each boosting round:
  - Find the weak learner that achieves the lowest *weighted* training error
  - Raise the weights of training examples misclassified by current weak learner
- Compute final classifier as linear combination of all weak learners (weight of each learner is directly proportional to its accuracy)
- Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost)

□ Attentional cascade(注意级联) for fast rejection of negative windows

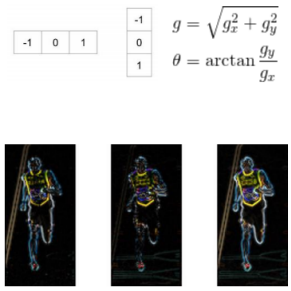
- Positive response from the first classifier triggers the evaluation of a second (more complex) classifier
- A negative outcome at any point leads to the immediate rejection of the sub-window



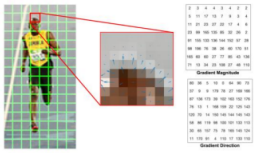
7.2 Deformable Part Models (DPM)

Histogram of gradients

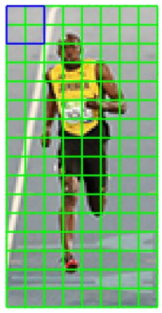
Step 1: Calculate the Gradient Images



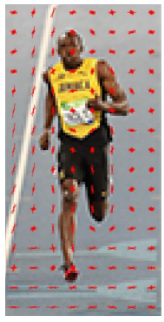
Step 2: Calculate Histogram of Gradients in 8×8 cells



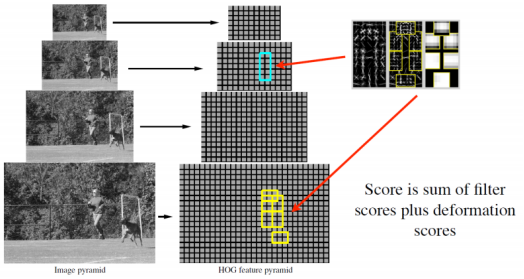
Step 3: 16×16 Block Normalization



Step 4: Calculate the HOG feature vector



- Coarser level for the root filter (whole object) and higher level for part filters



Partial models - Pictorial Structure

Objects are decomposed into parts and spatial relations among parts

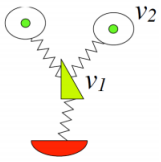
- Matching = Local part evidence + Global constraint

$$L^* = \arg\min_L \left(\sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i, l_j) \right)$$

It is not the Euclidean distance

- $m_i(l_i)$: matching cost for part i
- $d_{ij}(l_i,l_j)$: **deformable cost** for connected pairs of parts
- (v_i,v_j): connection between part i and j

$$E(L) = \sum_{i=1}^{n} m_i(l_i) + \sum_{(v_i,v_j)\in E} d_{ij}(l_i,l_j)$$



- For each l_1, find best l_2:

$$\text{Best}_2(l_1) = \min_{l_2} \left[m_2(l_2) + d_{12}(l_1,l_2) \right]$$

- Remove v_2, and repeat with smaller tree, until only a single part
- Complexity: $O(nk^2)$: n parts, k locations per part

Latent (Structured) SVM

w is a model
x is a detection window
z are filter placements

$$f_w(x) = \max_z w \cdot \Phi(x,z)$$

concatenation of filters and deformation parameters

concatenation of features and part displacements

- Linear SVM (convex) when z is fixed

$$f_w(x) = \max_z w \cdot \Phi(x, z)$$

- Solving by coordinate descent

1. Fixed w, find the latent variable z for the positive examples

$$z_i = \text{argmax}_{z \in Z(x_i)} w \cdot \Phi(x, z).$$

2. Fixed z, solve the Linear SVM to find w

- Select root filter window size
- Initialize root filter by training model **without** latent variables on un-occluded examples
- Root filter update: get new positives (best score and significant overlap with ground truth), add to positives and retrain
- Part initialization: sequentially choosing area a having high positive score and $6a = 80\%$ root area

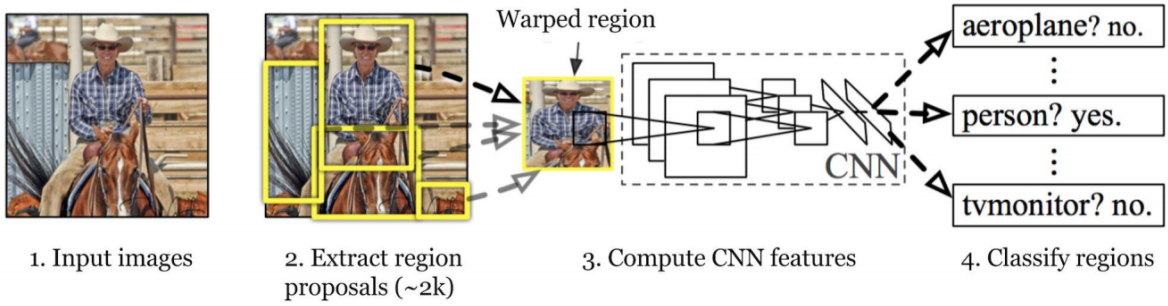
7.3 Two-stage Object Detection

□ The first stage identifies a subset of regions in an image that might contain an object

□ The second stage classifies the object in each region

R-CNN

(Regions with CNN features)



Propose class-independent regions of interest by **selective search**(**Warp** to have a fixed size as required to pretrained CNN)

------->Finetune CNN on warped proposal regions for K+1classes

------->Create features from the image proposals(□ One SVM for each object class□ The positive sample: IoU overlap threshold >= 0.3)

--------->correction offset using CNN features(Non-max suppression)

Fast R-CNN

Whole input image to the CNN to generate a convolutional feature map

Alter the pre-trained CNN:

 Replace the last max pooling layer of the pre-trained CNN with a **RoI** pooling layer

1. 根据输入image，将 ROI 映射到 feature map 对应位置；
2. 将映射后的区域划分为相同大小的 sections（sections数量与输出的维度相同）；
3. 对每个 sections 进行 max pooling 操作

 Replace the last fully connected layer and the last softmax layer (K classes) with a fully connected layer and softmax over K + 1 classes

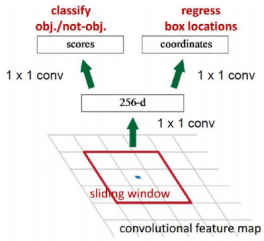
• Two output layers:

 A softmax estimator of K + 1 classes outputs a discrete probability distribution per RoI

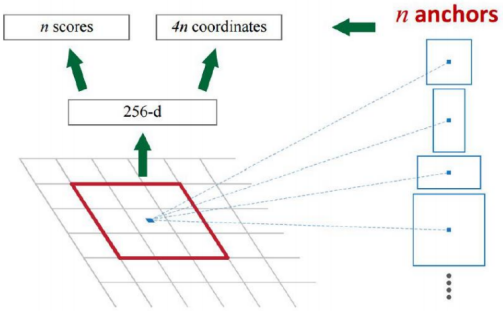
 A bounding-box regression model predicts offsets relative to the original RoI for each of K classes

Faster R-CNN

- Region Proposal Network (RPN)
  - RPN trained to produce region proposals directly
  - RoI Pooling, upstream classifier and bbox regressor (Fast R-CNN)
- Build a small network for:
  - Classifying object or not-object
  - Regressing bbox locations
  - Position of the sliding window: provide localization information with reference to the image
  - Box regression: provide finer localization information with reference to this sliding window



- Use **N anchor boxes** at each location
  - Anchors are **translation invariant**: use the same ones at every location
  - Regression gives offsets from anchor boxes
  - Classification gives the probability that each
  - (regressed) anchor shows an object

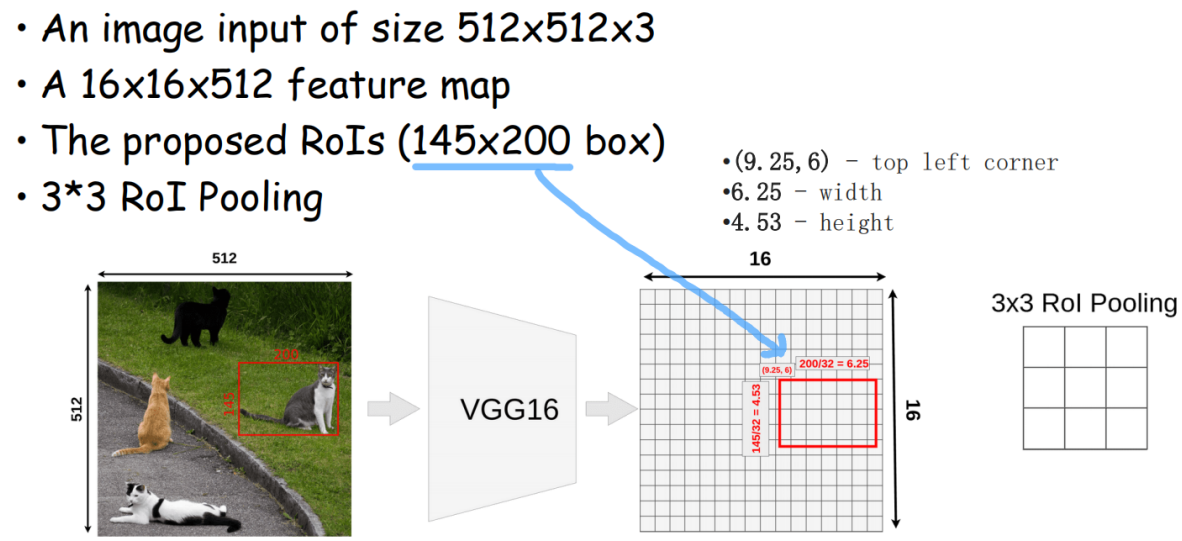


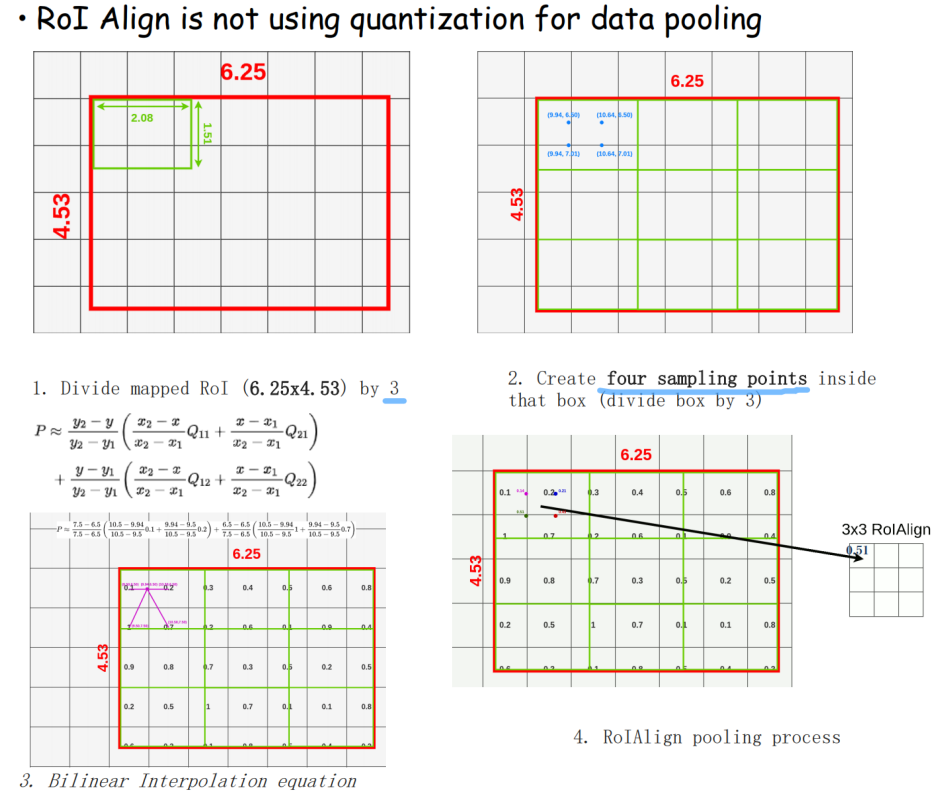
- Pre-train a CNN network
- Fine-tune the RPN
  - Initialization: Positive samples have IoU (intersection-over-union) > 0.7, while negative samples have IoU < 0.3
  - Slide a small n x n spatial window
  - Predict multiple regions (3 scales + 3 ratios => k=9 anchors at each sliding position)
- Train a Fast R-CNN object detection model using the proposals generated by the current RPN
- Use the Fast R-CNN network to initialize RPN training
- Fine-tune the RPN-specific layers
- Fine-tune the unique layers of Fast R-CNN
- Above three steps can be repeated if needed

Masked R-CNN

 Replacing the ROI Pooling module with a more accurate **ROI Align module**

- Inserting an additional branch out of the ROI Align module





7.4 One-stage Object Detection

YOLO

Our system divides the input image into a 7*7 grid

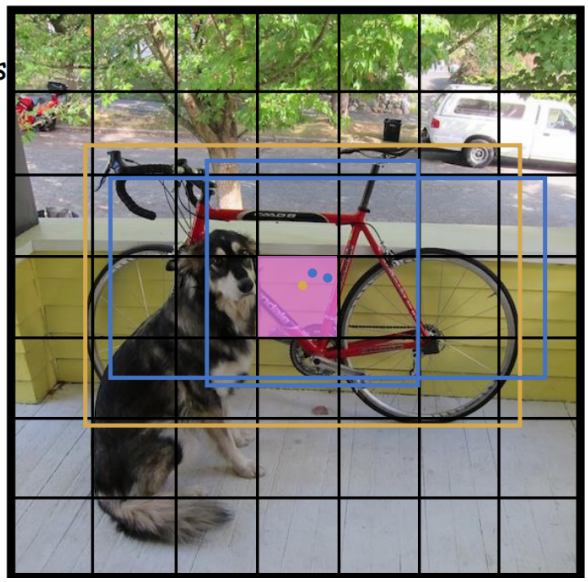
Each grid cell predicts 2 bounding boxes and confidence scores for those boxes

$$\begin{cases} p_{conf}, x, y, w, h \\ p_{conf}, x, y, w, h \\ p_{c_1}, p_{c_2}, \cdots, p_{c_{20}} \end{cases}$$

Ground Truth

1. Object belongs to the cell which the center located in

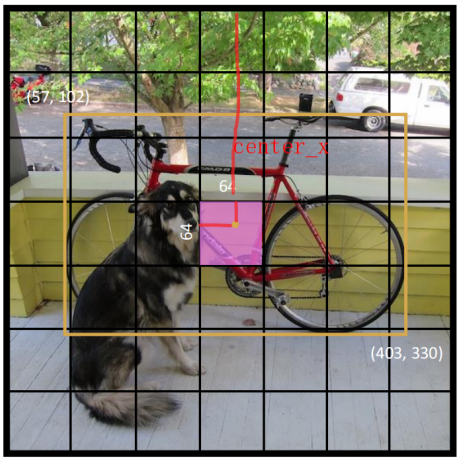
2. Object belongs to the predictor which has the IoU of highest score



$$IOU = \frac{A \cap B}{A \cup B}$$

Loss

The index of grid is (3, 3).
The gt representation is calculated by



minus 3grids

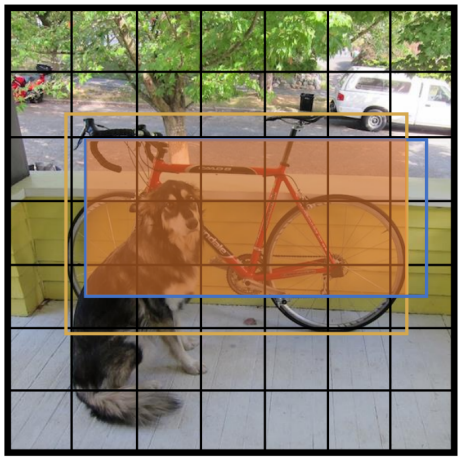
$\hat{x}_i$ = (center_x-64*3)/64

$\hat{y}_i$ = (center_y-64*3)/64

$\hat{w}_i$ = (403-57)/448

$\hat{h}_i$ = (430-102)/448

Loss



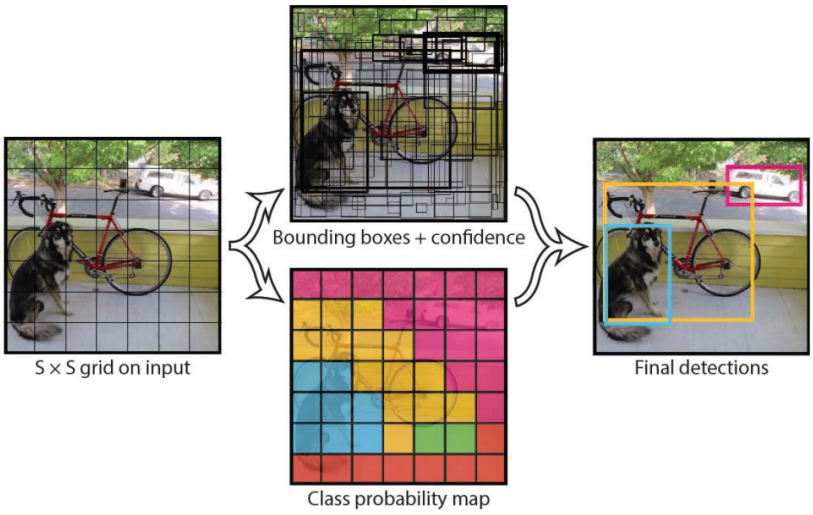
If has object:

$$\hat{C}_i = \text{IOU}^{\text{truth}}_{\text{pred}}$$

If has no object:

$$\hat{C}_i = 0$$

If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.



At test time we multiply the conditional class probabilities and the individual box confidence predictions

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

13. Segmentation

13.1 Semantic Segmentation

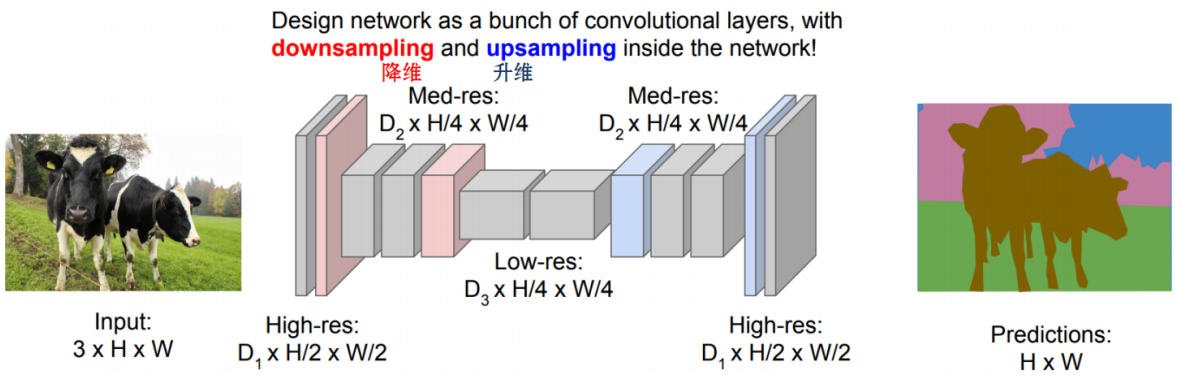
(半静态分割)

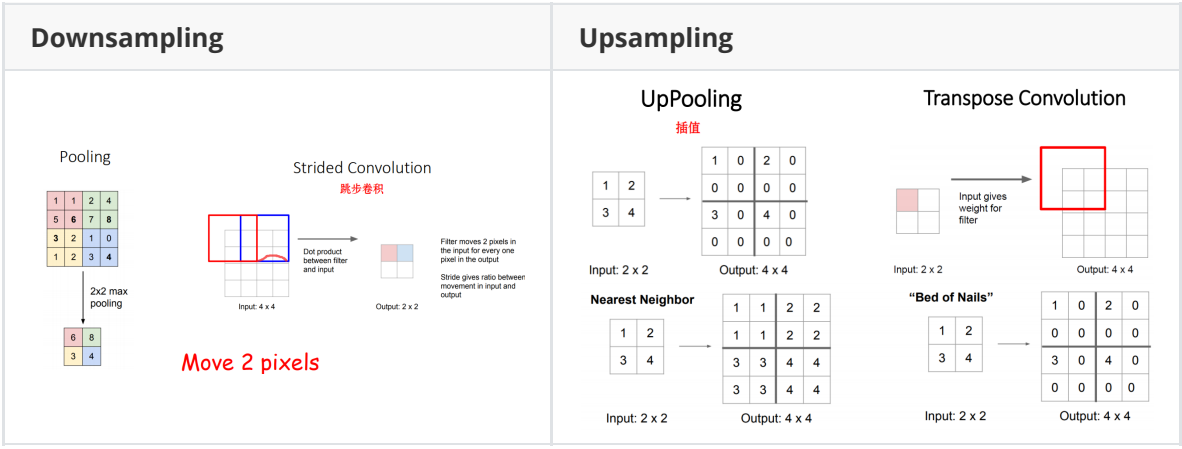
Things/Stuff

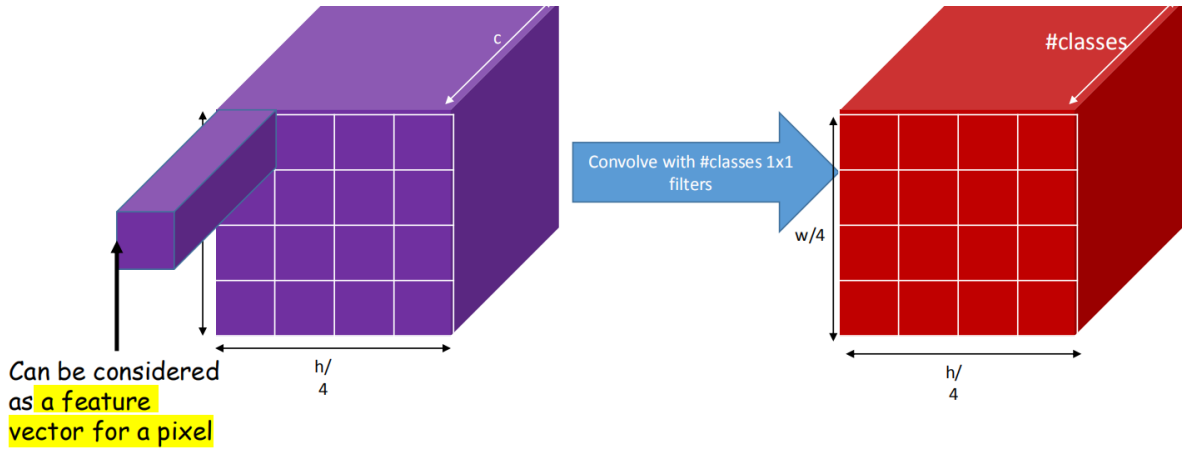
Things	Stuff
□ Person, cat, horse, etc □ Constrained shape □ Individual instances with separate identities □ May need to look at objects	□ Road, grass, sky etc □ Amorphous, no shape □ No notion of instances □ Can be done at pixel level □ “texture
Do object detection, then segment out detected objects	”Texture classification” Compute histograms of filter responses Classify local image patches

Down/Upsampling

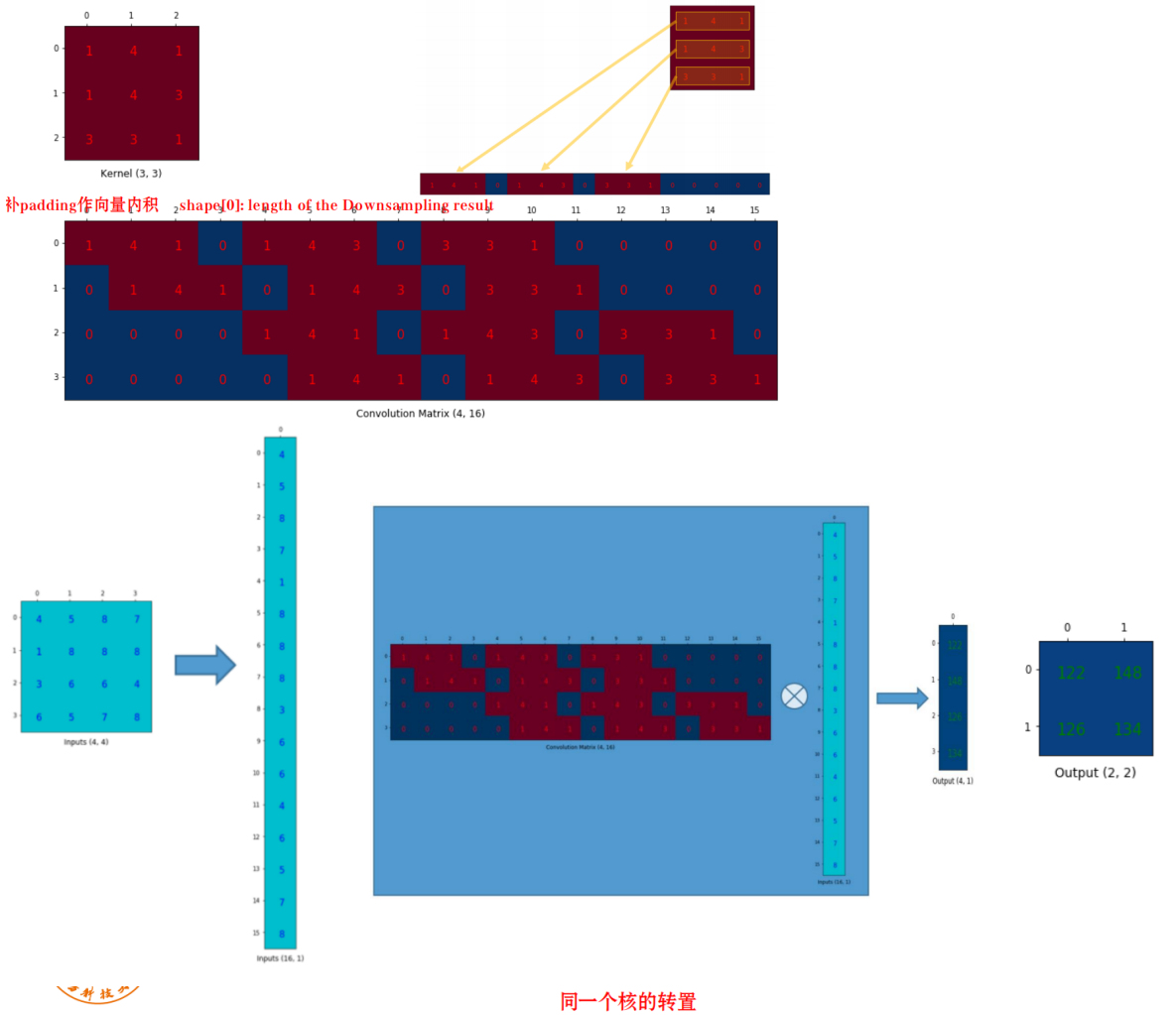
Fully convolutional



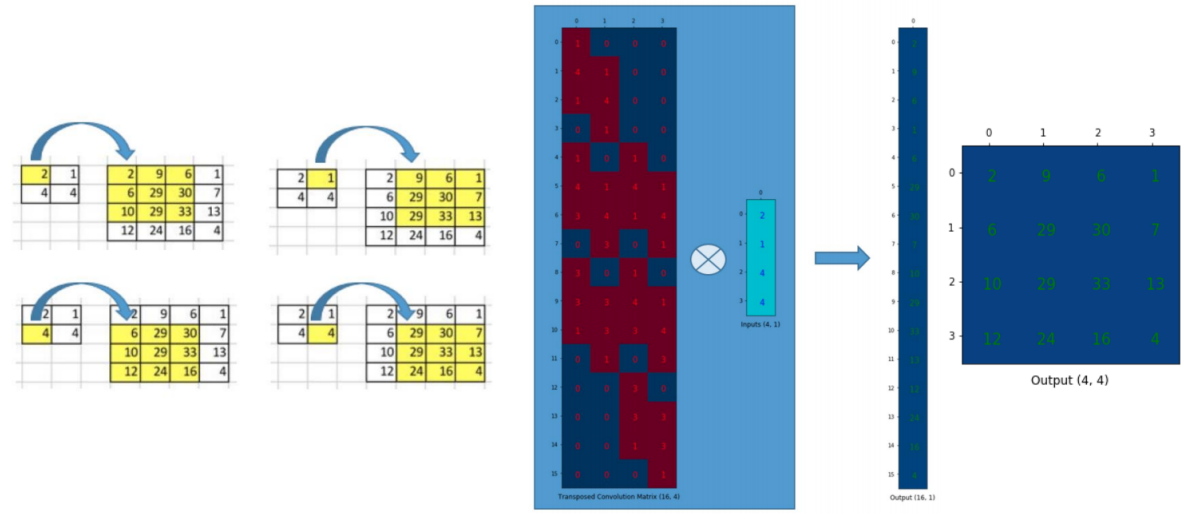




Transpose Convolution



同一个核的转置



Problems

• Problem: Need fine details!

  • Shallower network / earlier layers?

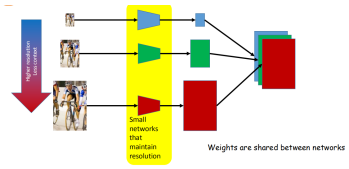
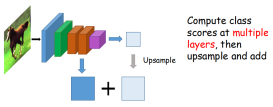
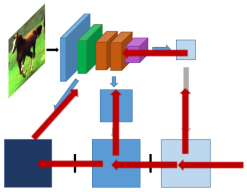
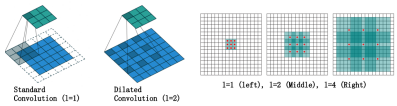
  □ Deeper networks work better: more abstract concepts

  □ Shallower network => Not very semantic!

• Remove subsampling?

  □ Subsampling allows later layers to capture larger and larger patterns

  □ Without subsampling => Looks at only a small window

Image pyramid	Skip connections	Dilation
	 Problem: early layers not semantic  Red arrows indicate backpropagation	• The figures illustrate an example of **dilated convolution when l=2**. We can see that **the receptive field is larger** compared with the standard one. 

13.2 Video Object Segmentation

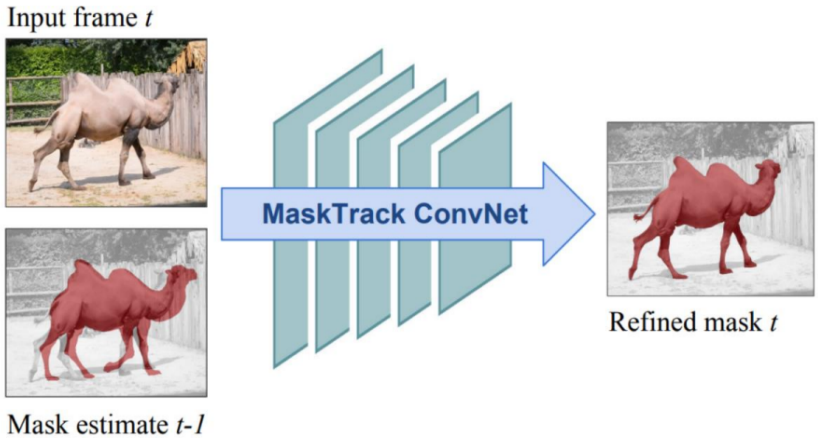
Appearance changes， Occlusions， Distraction from similar backgrounds

Semi-supervised (one-shot)	Unsupervised (zero-shot)	Interactive	Language-guided
 • We get the first frame ground truth mask, we know what object to segment	 • We just get a raw video, we have to find the objects as well as their masks (generally based on saliency or motion information).	 • We get coarse annotations (e.g., scribbles) to indicate what object to segment, simpler to provide than ground truth masks. • we can also refine the unsatisfactory results by providing additional scribbles	 • We get a sentence describing what object to segment.

One classical SVOS model: MaskTrack

前一帧的mask+当前帧--》当前帧的mask

• Overview: MaskTrack takes a tensor with four channels (RGB + previous frame mask) as input, predicts object masks for the current frame.



• Network

  □ DeepLabv2-VGG network

  □ pre-trained on ImageNet

  □ Extra mask channel of filters: gaussian initialization

• Offline training

  □ Does not require pixel-label annotations on videos

 Images and masks: ECSSD, MSRA10K, SOD, and PASCAL-S

 Deforming the binary segmentation masks

• Online training

 200 iterations

 1000 augmented training samples from the first frame

 Same learning parameters as for offline training

14. Object Tracking

• Input: target

• Objective: Estimate target state over time

• State: Position, Appearance, Shape, Velocity, affine transformation w.r.t. previous patch

• Choice: (O. S. S.)

 Object representation

 Similarity measure

 Searching process

Four stages:

① Target initialization

② Appearance modeling

 Visual representation: construct robust features and representation that can describe the object (template)

 Statistical modeling: use statistical learning techniques to build mathematical models for object identification effectively.

③ Motion estimation

④ Target positioning

Challenges

• Illumination Variation

• Scale Variation

• Occlusion

• Deformation – non-rigid object deformation

• Motion Blur

• Fast Motion – the motion of the ground truth is larger than 20 pixels

• In-Plane Rotation

• Out-of-Plane Rotation

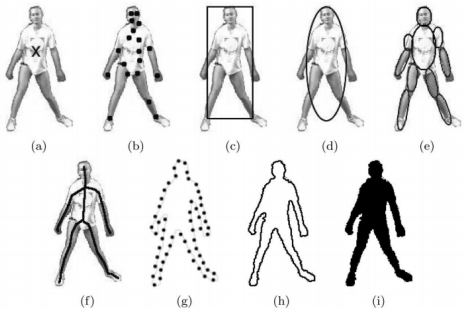
• Out-of-View

• Background Clutters

• Low Resolution – the number of pixels inside the groundtruth bounding box is less than 400

Object representation

- Object approximation:
  - Segmentation / Polygonal approximation
  - Bounding ellipse/box
  - Position only
- Goal: Measure affinity

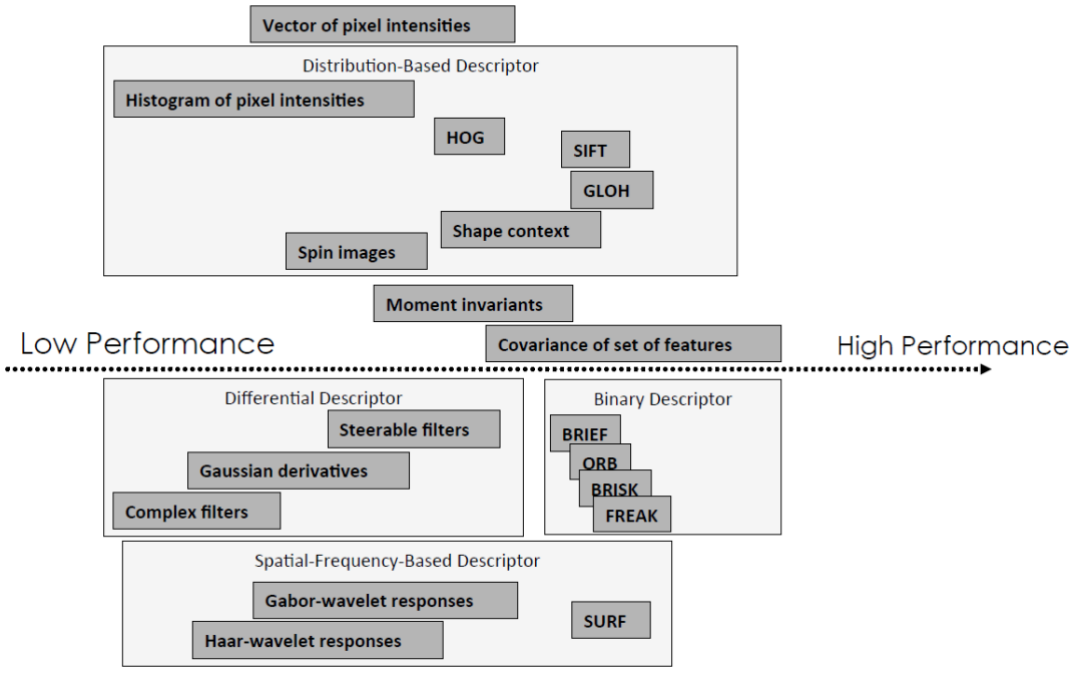


- In general: $aff(x,y) = \exp\left(-\frac{1}{2\sigma_d^2}\|f(x)-f(y)\|^2\right)$
- Examples:
  - Distance: $f(x) = location(x)$
  - Intensity: $f(x) = intensity(x)$
  - Color: $f(x) = color(x)$
  - Texture: $f(x) = filterbank(x)$

Pixels => Regions

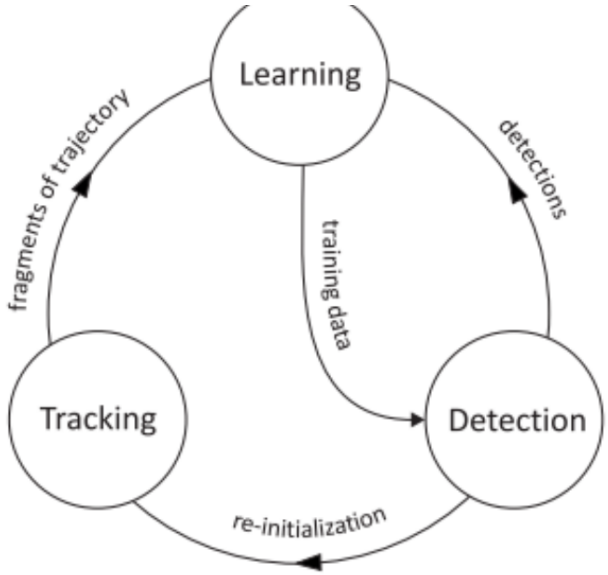
- Note: Can also modify distance metric

A bulk of Low-level features



Recent trend: CNN features

Online object tracking



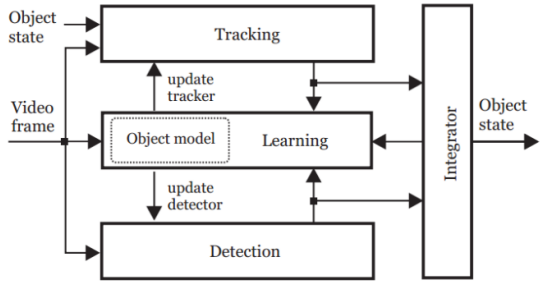
Built in the first or last frame

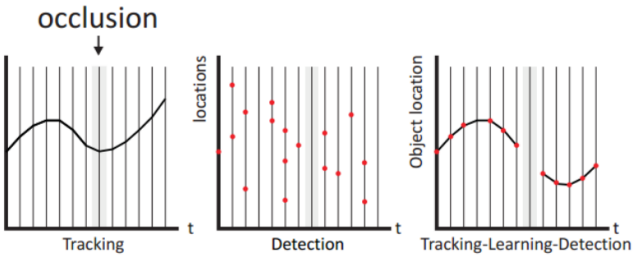
Focus more on: Similarity

Built offline or in past frames

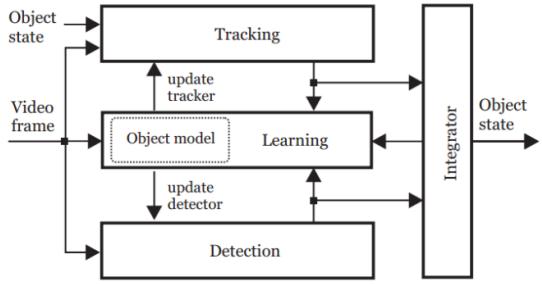
Focus more on: categories

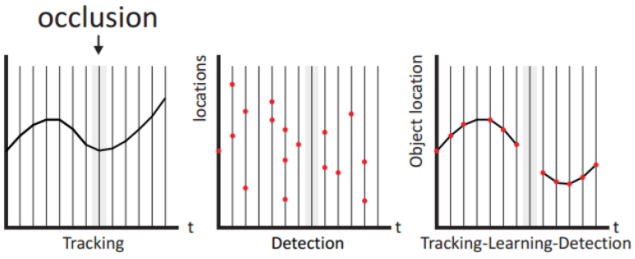
- Multiple models:
  - Tracking model (motion models: Kalman filter or Optimal flow)
  - Object model (features or templates)
  - Detection model (build in last frames)
- Cooperate and supervise each other





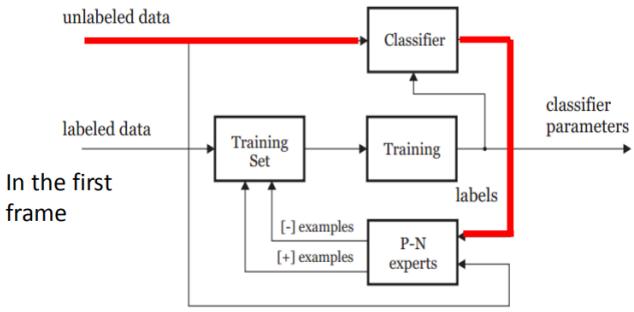
- Multiple models:
  - Tracking model (motion models: Kalman filter or Optimal flow)
  - Object model (features or templates)
  - Detection model (build in last frames)
- Cooperate and supervise each other





PN Learning

In the following frames



(i) A classifier for detection to be learned

(ii) Training set – a collection of labeled training examples

(iii) Supervised training– a method that trains a classifier from training set

(iv) P-N experts (object model)

Deep SORT Tracker

MOT(multiple object tracking):

Locate multiple objects of interest in a given video simultaneously, maintain their IDs and record their trajectories.

detections-->association

Motion information

- The state of each target at some point is modelled as:

$$(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$$

where (u, v) is bounding box center position, γ is aspect ratio, h is height, overdot means their respective velocities in image coordinates.

- A standard Kalman filter with constant velocity motion and linear observation model, is used to update the above target state.
- Link detections to existing tracks:

$$d^{(1)}(i,j) = (\boldsymbol{d}_j - \boldsymbol{y}_i)^{\mathrm{T}} \boldsymbol{S}_i^{-1} (\boldsymbol{d}_j - \boldsymbol{y}_i) \quad \text{(cost)}$$

马氏距离

where $\boldsymbol{d}_j$ is the j-th bounding box detection, $\boldsymbol{S}_i$ is the covariance matrix of the Kalman filter prediction, $\boldsymbol{y}_i$ is the Kalman filter prediction bounding box.

The equation calculates the Mahalanobis distance of groudtruth detection and the Kalman filter prediction.

motion model i 个target，j个追踪

- Then apply a threshold $t^{(1)}$ to $d^{(1)}(i,j)$, to check whether it's feasible to accept this link:

$$b_{i,j}^{(1)} = \mathbb{1}[d^{(1)}(i,j) \leq t^{(1)}] \quad \text{(gate)}$$

- For each bounding box detection $\boldsymbol{d}_j$, we compute an appearance descriptor $\boldsymbol{r}_j$ with $\|\boldsymbol{r}_j\| = 1$ by L2 normalization, where $\boldsymbol{r}_j$ comes from a convolutional neural network(wide residual).
- Keep a gallery $\mathcal{R}_k = \{\boldsymbol{r}_k^{(i)}\}_{k=1}^{L_k}$ of the $L_k = 100$ associated appearance descriptors for each track k, i.e, only keep the last 100 descriptors.
- Distance between the *i*-th track and *j*-th detection in appearance space is the smallest cosine distance the *i*-th track and *j*-th detection that:

$$d^{(2)}(i,j) = \min\{1 - \boldsymbol{r}_j^{\mathrm{T}} \boldsymbol{r}_k^{(i)} \mid \boldsymbol{r}_k^{(i)} \in \mathcal{R}_i\}$$

- Then, apply a threshold $t^{(2)}$ to $d^{(2)}(i,j)$, check whether it's feasible to accept this link:

$$b_{i,j}^{(2)} = \mathbb{1}[d^{(2)}(i,j) \leq t^{(2)}]$$

appearance model

- Combine both metrics using a weighted sum:

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j) \qquad (5)$$

This term is interpreted as the cost of associating the *i*-th track and *j*-th detection.

- And check whether motion and appearance are both less than the threshold:

$$b_{i,j} = \prod_{m=1}^{2} b_{i,j}^{(m)}. \qquad (6)$$

This term is interpreted as the gate of associating the *i*-th track and *j*-th detection.

Listing 1 Matching Cascade

Input: Track indices $\mathcal{T} = \{1, \ldots, N\}$, Detection indices $\mathcal{D} = \{1, \ldots, M\}$, Maximum age $A_{\max}$

Get detections from another alogrithm(YOLO, faster RCNN, ...)

Compute cost matrix $\boldsymbol{C} = [c_{i,j}]$ using Eq. 5
Compute gate matrix $\boldsymbol{B} = [b_{i,j}]$ using Eq. 6

Compute the association cost matrix and the matrix of admissible associations.

Initialize set of matches $\mathcal{M} \leftarrow \emptyset$
Initialize set of unmatched detections $\mathcal{U} \leftarrow \mathcal{D}$
for $n \in \{1, \ldots, A_{\max}\}$ **do**

n is in how many frames a tracker has not been updated.

Select tracks by age $\mathcal{T}_n \leftarrow \{i \in \mathcal{T} \mid a_i = n\}$

select the subset of tracks $\mathcal{T}_n$ that have not been associated with a detection in the last n frames

$[x_{i,j}] \leftarrow \text{min\_cost\_matching}(\boldsymbol{C}, \mathcal{T}_n, \mathcal{U})$

Solve the linear assignment between tracks in Tn and unmatched detections U

$\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j) \mid b_{i,j} \cdot x_{i,j} > 0\}$
$\mathcal{U} \leftarrow \mathcal{U} \setminus \{j \mid \sum_i b_{i,j} \cdot x_{i,j} > 0\}$

Update the set of matches and unmatched detections

end for
return $\mathcal{M}, \mathcal{U}$

- $\mathcal{T} = \{1, \ldots, N\}$: every track contains all the past detections in that track.
- $\mathcal{D} = \{1, \ldots, M\}$: detections in all frames.
- $A_{\max}$: tracks has no new added frame in the past $A_{\max}$ frames are thought dead.
- $\boldsymbol{C} = [c_{i,j}]$: $c_{i,j}$ is the cost of associating the *i*-th track and *j*-th detection.
- $\boldsymbol{B} = [b_{i,j}]$: $b_{i,j}$ is the gate of associating the *i*-th track and *j*-th detection.

Offline object tracking

GOTURN

Generic Object Tracking Using Regression Network

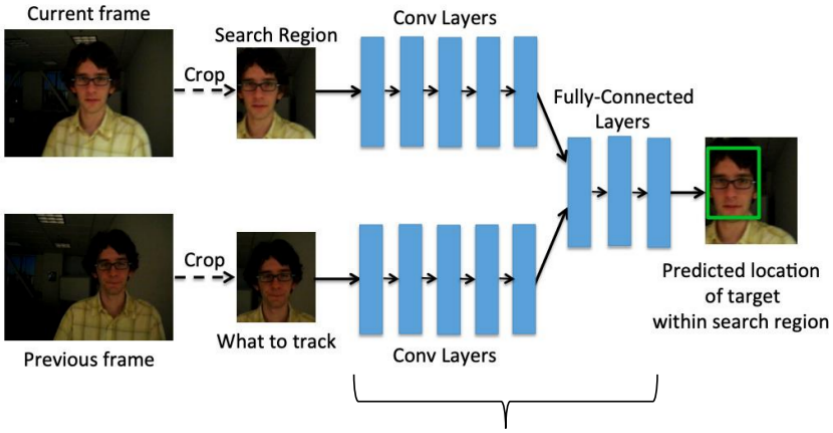


image-comparison tracking architecture

What to track?

• Crop and scale the previous frame to be centered on the target object

• Padding-receive extra contextual information about the surroundings

Where to look?

• Hypothesis: objects tend to move <u>smoothly</u> through space

• Choose a search region in current frame based on the object's previous location—> crop

$$c'_x = c_x + w \cdot \Delta x \qquad w' = w \cdot \gamma_w$$
$$c'_y = c_y + h \cdot \Delta y \qquad h' = h \cdot \gamma_h$$

– (c'_x, c'_y): the center of the bounding box in the current frame
– (c_x, c_y): the center of the bounding box in the previous frame
– w, h: the width and height
– $\Delta x, \Delta y$: position change of the bounding box relative to its size (a Laplace distribution with a mean of 0 ,in practice)
– γ_w, γ_h: the size change of the bounding box (modeled by a Laplace distribution with a mean of 1)

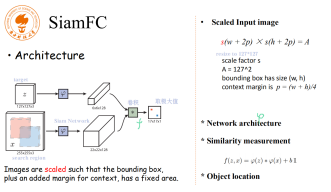
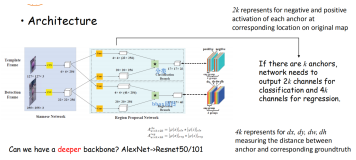
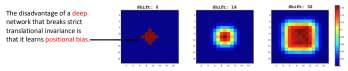
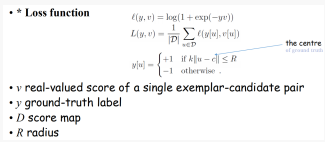
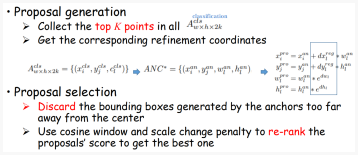
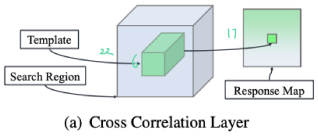
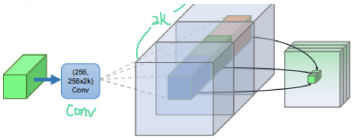
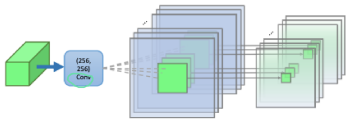
在周围固定范围提取

• Conv Layers: extract image features

• Fully connected layers: compare the features from the target object to the features in the current frame to find where the target object has moved and output bounding box.

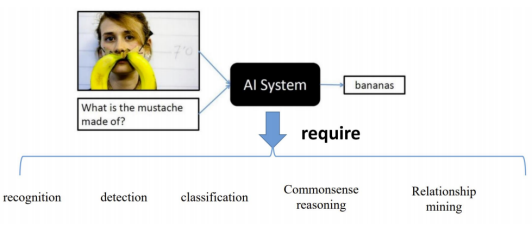
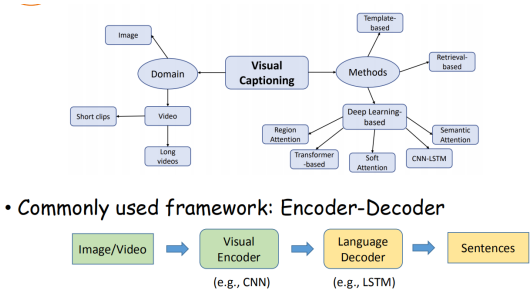
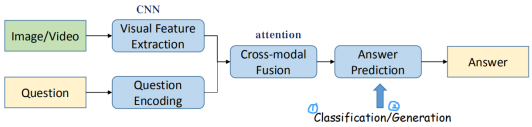
• Loss: L1 loss between predicted bounding box and ground truth.

Siamese-based Tracker

SiamFC	SiamRPN	SiamRPN++
		**Spatial aware sampling strategy** [目标不一定在bbox中心] • Positive samples are evenly distributed within a certain range, not always at the center • The range is the distance from the center point, which means shift  Target may appear at any position in the search region and earned feature representation should stay spatial invariant
		**Layer-wise Aggregation** 
Cross Correlation (XCorr) layer predicts a single channel similarity map between target template and search patches  (a) Cross Correlation Layer	Up-Channel Cross Correlation (UPXCorr) layer outputs a multi-channel correlation features by cascading a heavy convolutional layer with several independent XCorr layers 	Depth-wise Cross Correlation (DW-XCorr) layer predicts multi-channel correlation features between a template and search patches  (c) Depth-wise Cross Correlation Layer

15. Vision-Language Learning

aims to build models that can process and relate information from vision and language

Visual Question Answering:	Cross-modal retrieval
[vision+language--->language] Generating natural language description of an image/video	[language--->vision] Given an image and a question (text) about the image/video, the model aims to provide an accurate natural language answer
• Applications: ➢ Alt-text generation (from PowerPoint). ➢ Generating summaries for Videos (YouTube) ➢ Content-based image retrieval.	
	

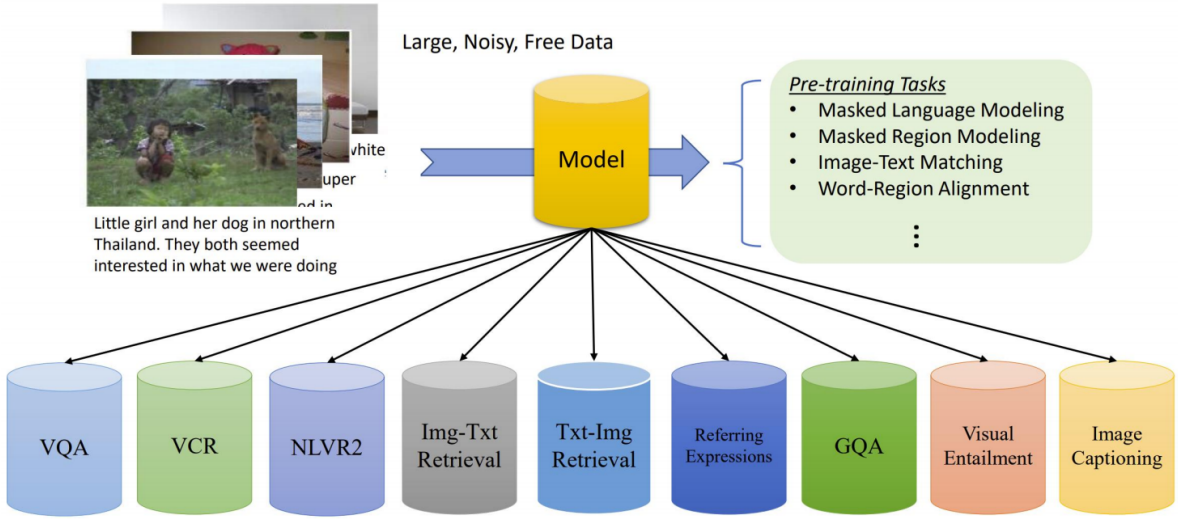
Visual captioning: vision--->language

Text-to-image Synthesis: language--->vision

Vision-Language Navigation

Vision dialog

NLVR2: Predict if the caption is True or False



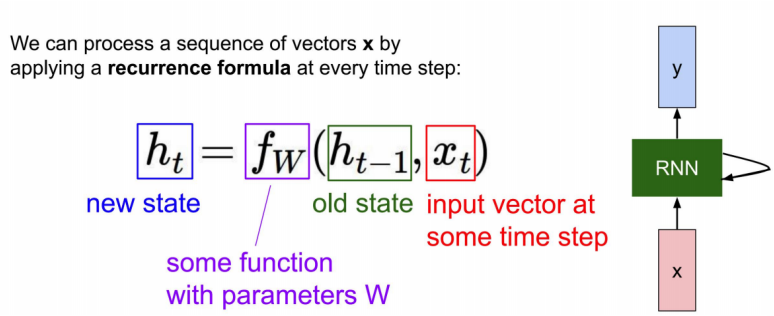
Sequence-to-Sequence Modeling (Seq2Seq)

Text, image and video are all sequence data

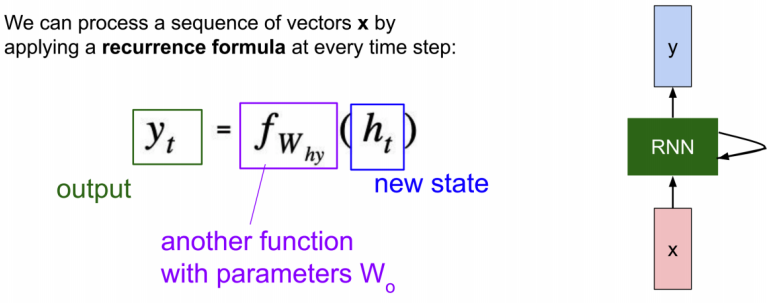
RNN

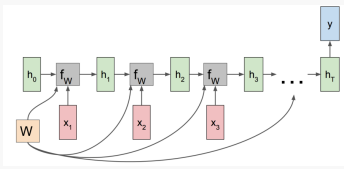
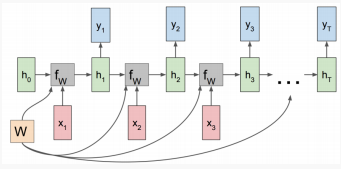
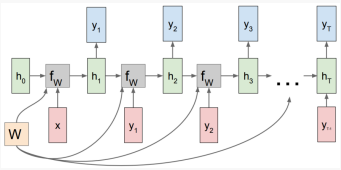
Recurrent Neural Networks (RNN): connections between nodes form a directed or undirected graph along a temporal sequence

RNN Hidden State Update

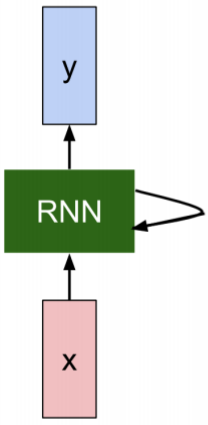


RNN Output Generation



Many input to one output	Many input to many output	One input to many output
• Text classification • Stock forecasting	• Language translation	• Image caption
		

The state consists of a single *"hidden"* vector **h**:



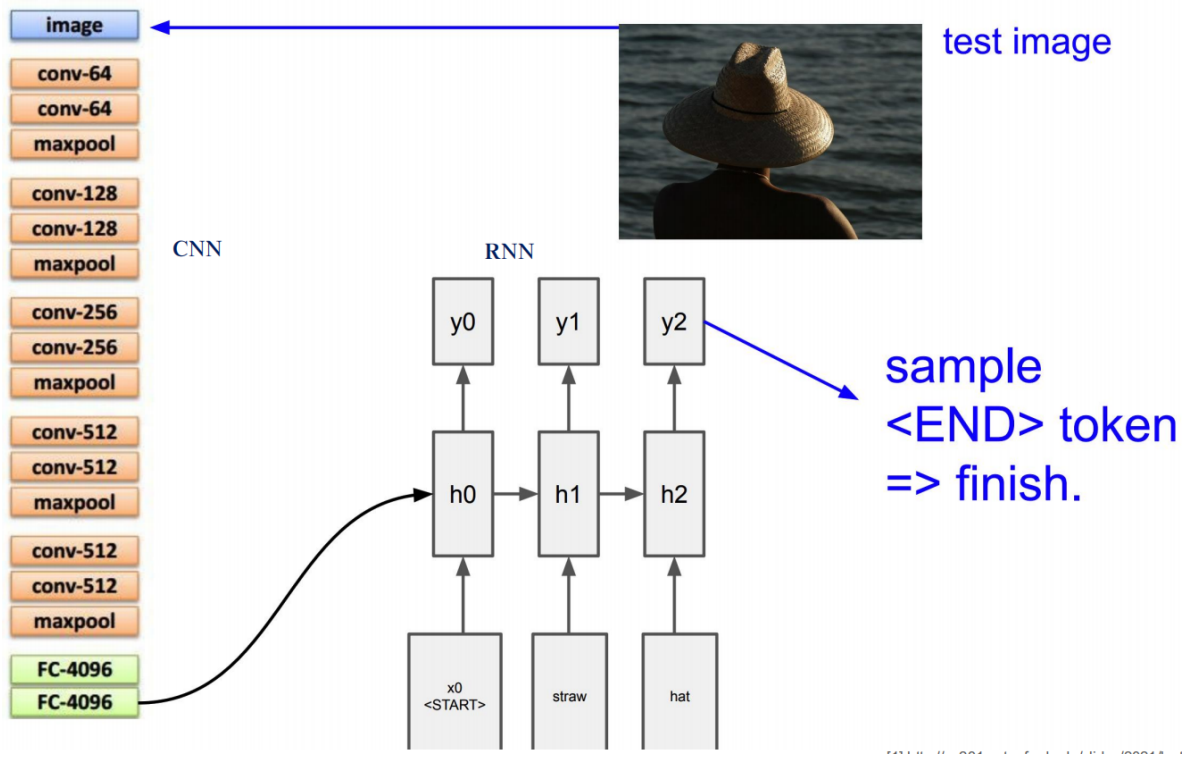
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Sometimes called a "Vanilla RNN" or an "Elman RNN" after Prof. Jeffrey Elman

Long Short-Term Memory (LSTM)

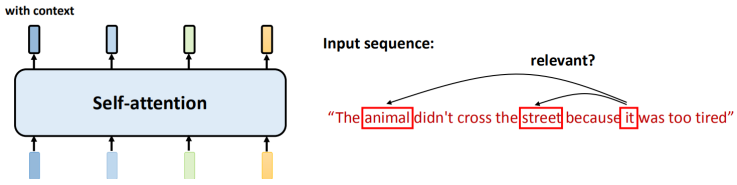


Transformers

• Sequence-to-sequence (Seq2seq) • Encoder and Decoder Stacks

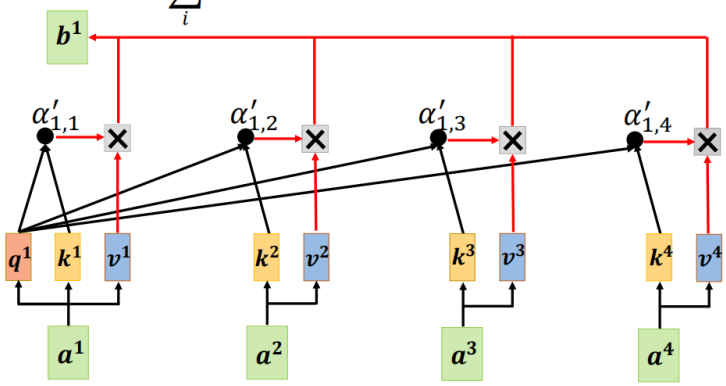
Self-Attention

- How to consider the context of the whole sequence?



- **Calculate self-attention using vectors**

$$b^1 = \sum_i \alpha'_{1,i} v^i$$ Extract information based on attention scores



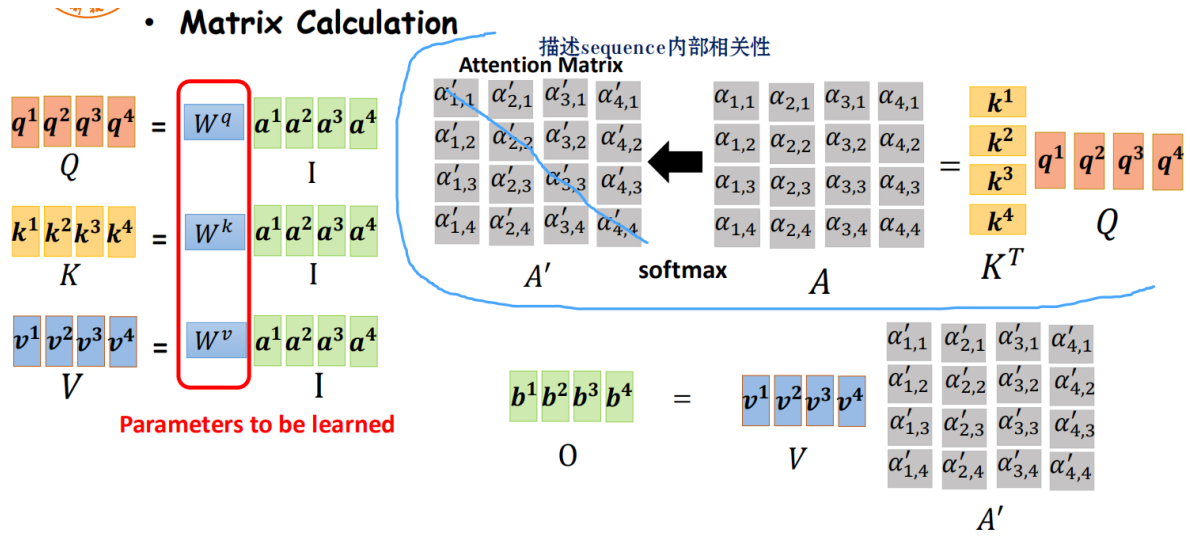
Query: $q^i = W^q a^i$

Key: $k^i = W^k a^i$

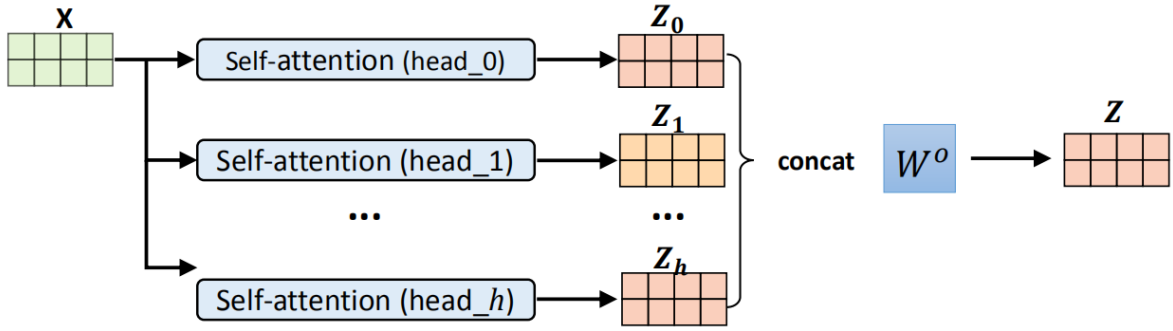
Value: $v^i = W^v a^i$

$$\alpha_{1,i} = \frac{q^1 \cdot k^i}{\sqrt{d_k}}$$ 越像的值贡献越大 ***Dot-product***

$$\alpha'_{1,i} = \text{Softmax}(\alpha_{1,i}) = \frac{\exp(\alpha_{1,i})}{\sum_j \exp(\alpha_{1,j})}$$



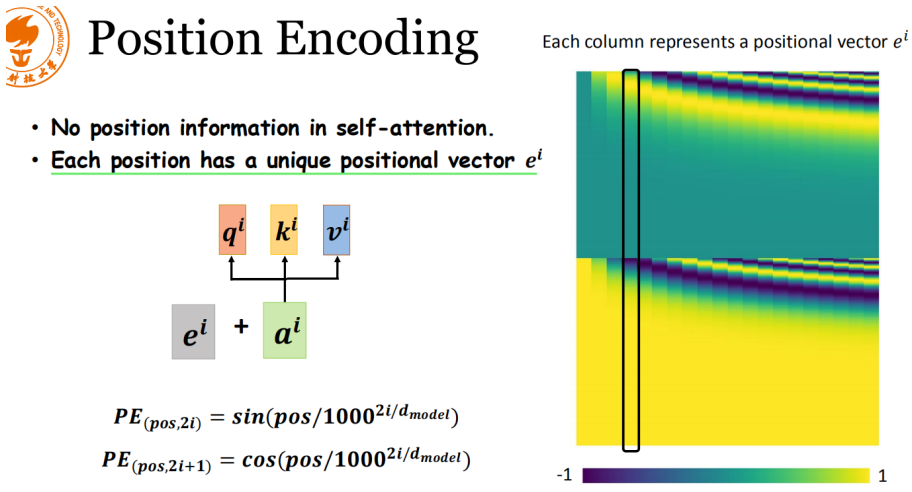
Multi-head Self-Attention



$$\mathbf{MultiHead}(Q,K,V) = \mathbf{Concat}(\mathbf{head}_1,\ldots,\mathbf{head}_h)W^O$$

$$\mathbf{where\ head_i} = \mathbf{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Position Encoding



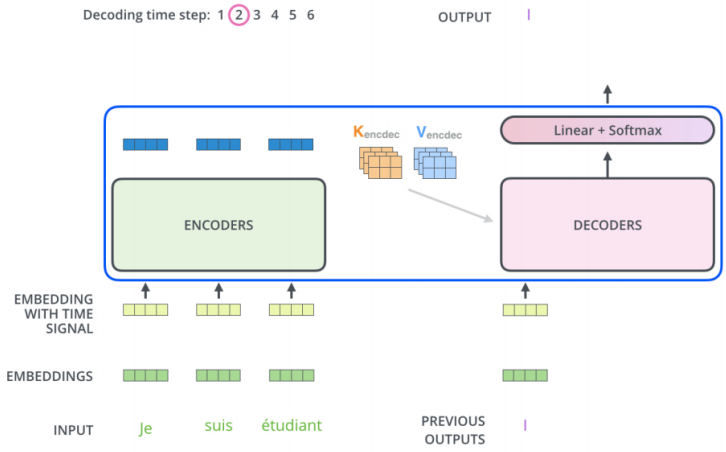
- No position information in self-attention.
- Each position has a unique positional vector e^i

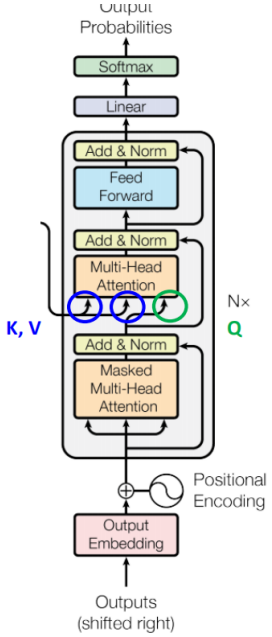
$$PE_{(pos,2i)} = sin(pos/1000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = cos(pos/1000^{2i/d_{model}})$$

Transformer Decoder

Autoregressive





• CNN: self-attention that can only attends in a receptive field.

• RNN： hard to learn long-range dependencies; nonparallel, more computation time

in Neural Language Processing (NLP)

• BERT: same network architecture as **transformer encoder.**

Masked language modeling (MLM) pre-training

in Computer Vision (CV)

• Image Recognition

• Object detection

• Vison-Language (VL)

Appendix

a.1 Transfer Learning

目标

将某个领域或任务上学习到的知识或模式应用到不同但相关的领域或问题中。

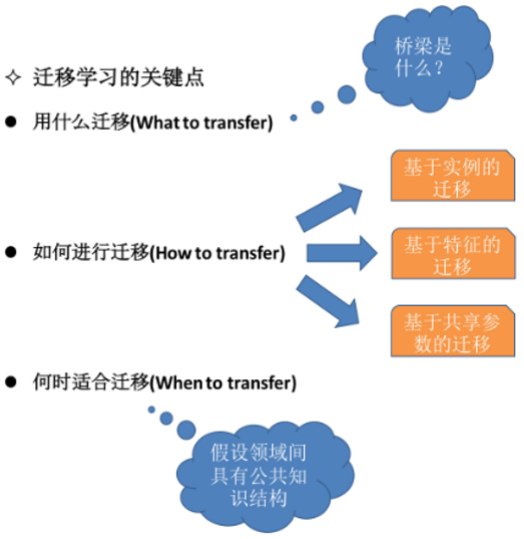
主要思想

从相关领域中迁移标注数据或者知识结构、完成或改进目标领域或任务的学习效果。

迁移学习里有两个非常重要的概念

域（Domain）：可以理解为某个时刻的某个特定领域，比如书本评论和电视剧评论可以看作是两个不同的domain

任务（Task）：就是要做的事情，比如情感分析和实体识别就是两个不同的task



通常情况下第一层与具体的图像数据集关系不是特别大，而网络的最后一层则是与选定的数据集及其任务目标紧密相关的；文章中将第一层feature称之为一般(general)特征，最后一层称之为特定(specific)特征。

随着参数被固定的层数n的增长，两个相似度小的任务之间的transferability gap的增长速度比两个相似度大的两个任务之间的transferability gap增长更快

两个数据集越不相似特征迁移的效果就越差

右图：到了4~5层的时候，精度开始下降，我们直接说：一定是feature不general了！

影响迁移能力的两个主要问题：

a) 网络断开带来的最优化困难问题；

b) 由于高层网络层中特定化的特征表示使得在迁移过程中所带来的性能损失。

• 两个因素哪一个占支配位置，取决于迁移的特征是在网络的bottom、middle还是top