1. Introduction to image processing.

Definition: set of computational techniques for analyzing, enhancing, compressing, reconstructing images.
(import, analysis, manipulation, output)

Structure:

1. Image representation
   - resolution
   - color

2. Point Processing
   - Grey value transforms.
   - Co-ordinate transform

3. Region Processing
   - Convolution
   - Smoothing (Gaussian smoothing)
   - Edge detection (Basic, Prewitt, Sobel, Canny, Laplavian/M-H algo, Difference of gaussian)
   - Template Matching
   - Rank Order Filters

4. Image Processing
   - Region Description
   - Finding Blobs
   - Measurements

1. Image Representation

1. 图像表示：基于像素坐标和三原色：R, G, B 分量.

RGB Format - Red, Green, Blue colour channels

三原色，三通道.

R	G	B	R	G	B	R	G	B	⋯
R	G	B	R	G	B	R	G	B	⋯
R	G	B	R	G	B	R	G	B	⋯
⋮			⋮			⋮			⋱

Important!

One byte per colour → three bytes per pixel (0~255, 0~255, 0~255)

Qu. How many bits in a byte? 8.

Qu. How many bits per colour pixel? 24.

Qu. 12Mp colour image takes up how many Mb? 12M·24 / 1024²

Greyscale Format - Intensity

I	I	I	⋯
I	I	I	⋯
I	I	I	⋯
⋮	⋮	⋮	⋱

One byte per pixel

0 255

Qu. What are the minimum and maximum values that can be stored in an **unsigned** byte?

彩色：存储三个 (R, G, B) color matrix. 每个均为对应色彩分量的 intensity.

灰度：只存储合灰度强度信息的 intensity matrix.

Images are often represented by a matrix:

Matrix size = **R**ows x **C**olumns [*Mnemonic: **R**hubarb and **C**ustard*]

For example, 2848 x 4272

Pixel coordinates are (r, c)

[*Take note if using Python/OpenCV/numpy*]

像素矩阵中的信息表示：

行 × 列.

以左上角为原点.

2. 颜色表示：结合人眼的生理特性，用标准化的方式表示色彩空间.

1. 三原色理论：

Thomas Young. 使用 red, green, blue 即可 approximate many colors. (真实世界颜色无限多)

2. 人眼生理特性：

视网膜上的光感细胞分别对：短波 (roughly red), 中波 (roughly green), 长波 (roughly red) 敏感.

⇒ 三原色理论和实践贴合人眼生理特性.

3. 人眼观察 (有颜色的) 物体：物体吸收其他颜色 (波长) 的光，反射 x 颜色的光 ⇒ 看见 x 颜色的物体.

Light-sensitive cells in the eye:

- **Rods** for low-light vision, no colour 杆状细胞
- **Cones** for colour 柱状细胞

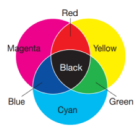
Three type of Cone:

- S sensitive to short-wavelength visible light (roughly blue)
- M sensitive to medium-wavelength visible light (roughly green)
- L sensitive to long-wavelength visible light (roughly red)

- 65% of cones are sensitive to red light
- 33% of cones are sensitive to green light
- 2% of cones are sensitive to blue light (but are the most sensitive)

Additive and subtractive primary colours

Additive primaries
Mixing of light



Subtractive primaries
Mixing of pigments (e.g. paint)

How do we see a green leaf?

White light from the Sun illuminates the leaf

Pigments in the leaf reflect green light and absorb all other colours

The reflected green light is detected by the green cones in our eyes

4. Color Matching 问题.

目标：exactly describe color to others.

Solution: use a standard to describe color, so all parties know the exact color being specified.

* CIE 1931 - x, y, z color space：古老，规定的 R, G, B 波长和人眼敏感的不一致.

CIE 1931 XYZ colour space

Commission Internationale de l'Éclairage (International Commission on Illumination), in 1931

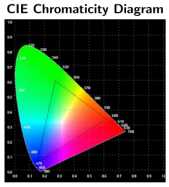
Standard colour frequencies

Colour	Wavelength (nm)
Blue	435.8
Green	546.1
Red	700.0

Note. These values do not correspond with the response of cone cells mentioned above

This is because the experimental data wasn't available in 1931

CIE Chromaticity Diagram



Pure saturated colours are around the edge of the shape

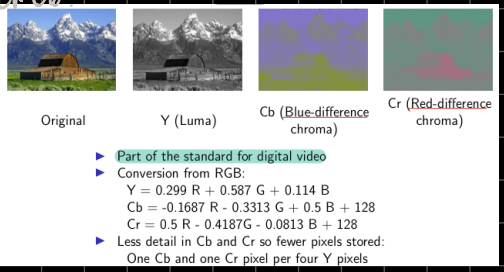
White is added as you move towards the centre - the point of equal energy - to make less saturated colours

(1931 年没有实验测量的条件...

规定：红 = X, 绿 = Y, 蓝 = Z.

归一化 (normalise)：$x = \frac{X}{X+Y+Z}$, $y = \frac{Y}{X+Y+Z}$, $z = \frac{Z}{X+Y+Z}$. 故有 $x+y+z=1$.

* YCbCr:



1. part of standard for digital video.
2. can be converted from RGB.
3. less detail in Cb, Cr ⇒ fewer pixels stored.

* Perceptual Spaces:

相比 CIE，更关注的感受效果：

equal distances on CIE diagram does not correspond to equal changes in perceived color.

but human care about color changes. ⇒ describe colors in a way suitable for humans.

常见的 Peneptual Spaces: IHS / HSI, HSV, HSB, Lab, ...

IHS color model
- intensity (brightness)
- hue (basic color)
- saturation (color depth)

Lab Color Space
- L: lightness
- a: green to red
- b: blue to yellow.

3. 图像处理的信息来源：

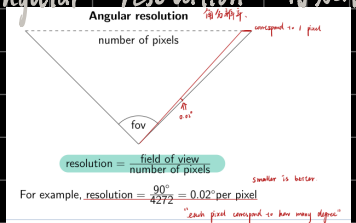
任何可以生成一个 spacial coherent measurement of some property 的物体/物质(如电磁波、光波、波物)均可被视为"图形信息"加以处理：

- 电磁波：
  - γ 射线，X 射线，微波，红外波，可见光。
    机械波物
- 其他：
  - 声波 (sonar?)、波物。

注：
1. 使用 x 光可进行 CT，胸片显影 (medical) 和机械生产检测，结构探伤。
2. γ 射线显影：辐射被人体吸收。
3. 红外线：不文观测，指纹显像。
4. 红外线：热能检测，侦测 (可以和海洋线 (水吸收 infrared，显黑色)
5. 微波，雷达：地形扫描。
6. 超声波：显像 (胎儿)，利用声纳测距。

4. 空间分辨率 Spatial Resolution

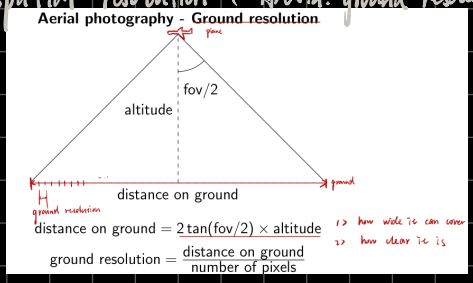
1. angular resolution 角分辨率：



1. 固定 fov 不变，则 fov 越小，像素密度越大，可识别的物体越精细。
2. 固定 fov 不变，增加像素 density，则 resolution 越小：

⇒ 在 spatial resolution 中，resolution 表示每个 pixel 对应的角度大小，该值反而越小越好！

2. Spatial resolution (或称: ground resolution)



$$\text{distance on ground} = 2\tan(\text{fov}/2) \times \text{altitude}$$

1> how wide is can cover
2> how clear it is

$$\text{ground resolution} = \frac{\text{distance on ground}}{\text{number of pixels}}$$

distance on ground: 视野的宽度.

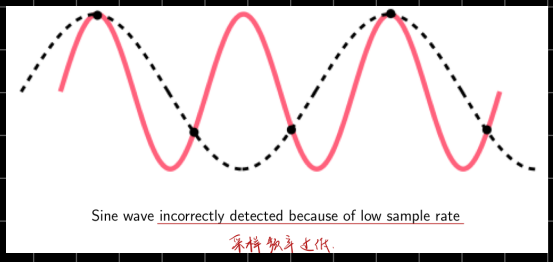
ground resolution: 清晰程度.

注: camera 的 fov 不影响视野的宽度!

5. Nyquist's Theorem 采样定理.

若要通过采样重建某个周期性信号, 则
- 采样频率不得低于被采样信号频率的2倍.
- 采样周期不得大于被采样信号周期的 1/2.

- A periodic signal can be reconstructed if the sampling interval is at least half the period
- An object can be detected if two samples span its smallest dimension
- More samples are needed if we want to recognise the object



Sine wave incorrectly detected because of low sample rate

采样频率过低

6. Intensity resolution 强度分辨率.

- Humans can see:
  - About 40 shades of brightness
  - About 7.5 million shades of colour
- Images are commonly stored with:
  - 256 shades of grey (brightness)
  - 16.8 million shades of colour

better than we can see...

人眼可感知的光强等级和色彩数是有限.

7. noise 噪音.

噪音来源: { 外界: fog, rain, ...
内部: 传感器信号干扰. 数字电路内部存在噪声.

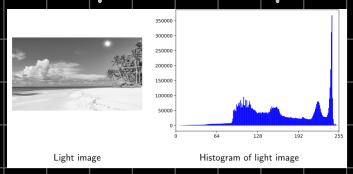
人造噪声: gaussian noise.

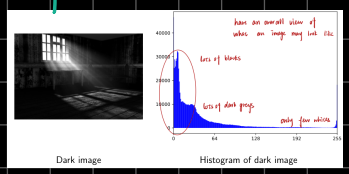
衡量指标: Signal to Noise ratio (SNR, 信噪比).
SNR 以 dB 为单位. 信噪比↑, 噪音↓.

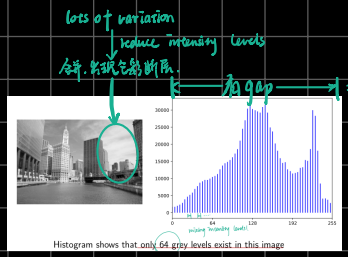
2. Point-processing techniques.

1. Image histogram 图像(灰度值)直方图.

作用: 统计(灰度)图像中不同 intensity 值的数量.





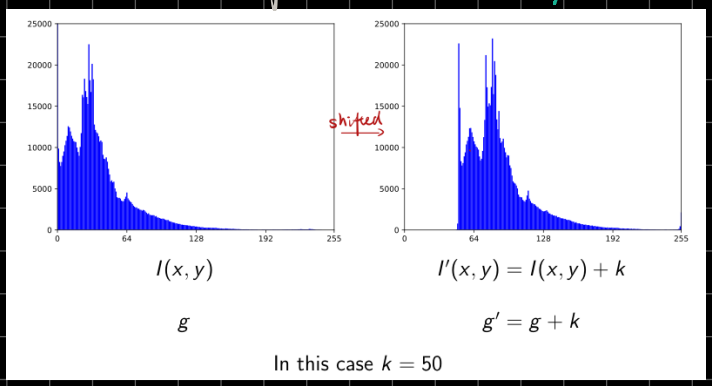


⇒ 说明图中不含 intensity 为某些值的像素点.

2. 对图像的灰度值执行的映射(调节):

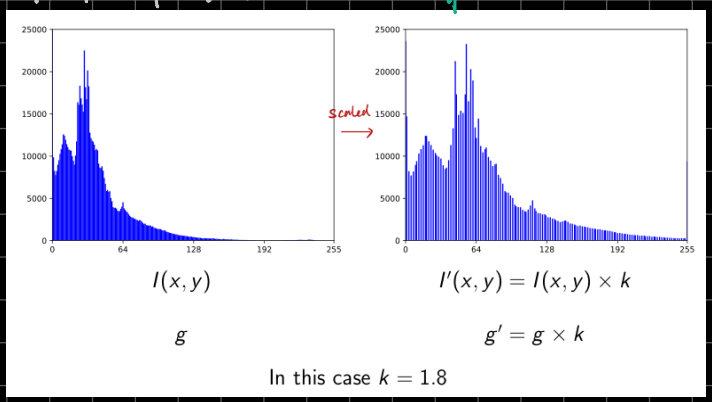
i> Brightness adjustment 亮度调节

亮度调节: 为像素的灰度值(intensity)加上 offset.
在 histogram 上表现为整体平移.



ii> Contrast Adjustment 对比度调节

对比度调节: 为像素灰度值乘上 offset.



注: 调节对比度同时拉大了像素 intensity 间的差距, 有可能更便于 thresholding.

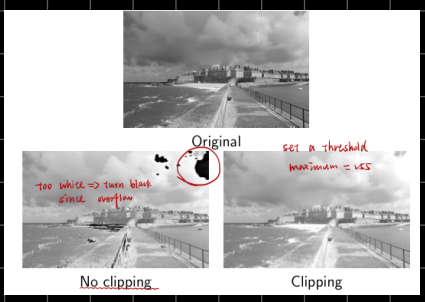
iii> Clipping

调节亮度和对比度时, 若经变换后 intensity 超出表示范围(0~255), 则要进行 clipping 防止 overflow.

When you process pixel values, you need to check for the result going over 255 and below 0

Example pseudocode

```
uint8 px = 250;
...
int32 p = px;          // can store bigger numbers
p += 50; // Increase brightness
if(p > 255) p = 255;   // avoid overflow.
if(p < 0) p = 0;
px = p;
```

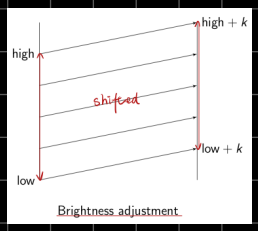


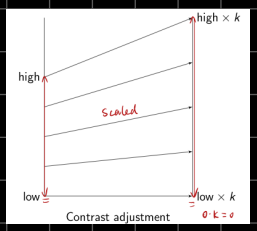
iv> 于之上的 intensity 映射

我们可将上述的 pointwise operations 视为 input-output mapping:

i> Brightness adjustment: $x \longmapsto x + k$

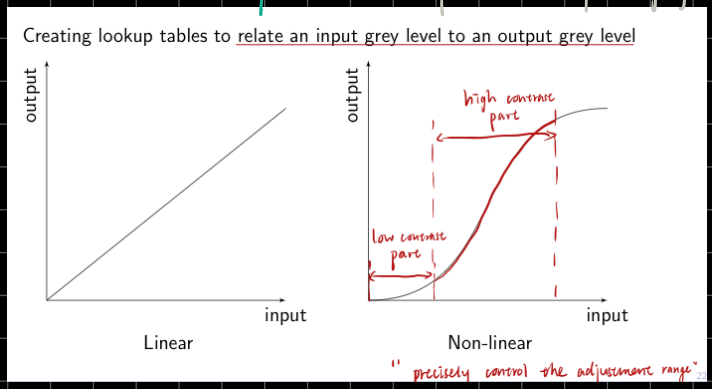
ii> Contrast adjustment: $x \longmapsto x \cdot k$





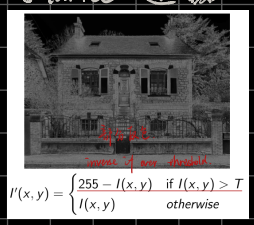
iii) (non-linear) Input-output mapping

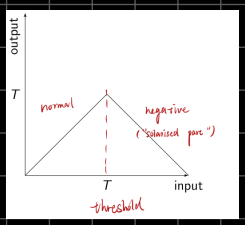
we can create lookup tables to map (relate) input gray level to output gray level.



一些常见的复杂 input-output mapping:

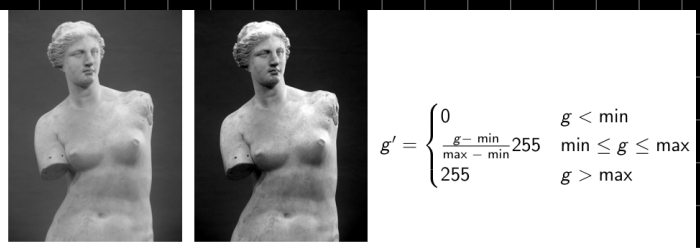
i) solarize 过曝:

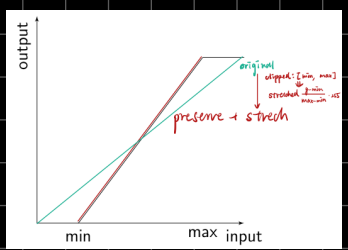




从 threshold 开始灰度值关于 T 取反.

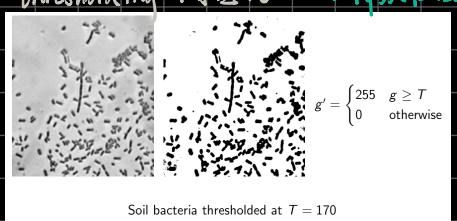
ii) min-max linear stretch 最小-最大拉伸

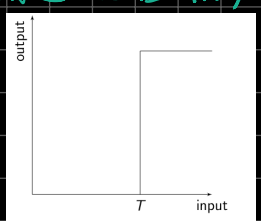




将原来的映射关系在区间 [min, max] 间压缩. 让黑者更黑, 亮者更亮.

iii) thresholding 阈值化 (将灰度图像变化为二值图像)





得 image 转为只有 0 和 255 两种 intensity 的 binary-intensity image.

区分前景. 本质: classification.

3. 对 threshold 方法和阈值选择/计算的讨论:

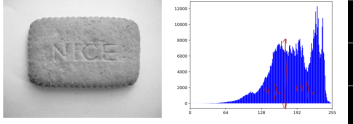
1. 人工选择: 肉眼调参
2. P-tile Automated Thresholding

known the percentage of pixels an object should occupy,

then automatically select threshold value T based on this information.

Algorithm

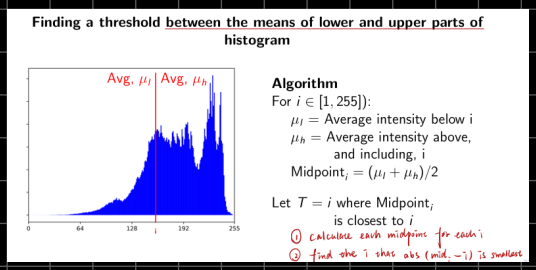
- Calculate the number of pixels that should be object (% of image size)
- Create image histogram
- Accumulate frequencies until total exceeds number of expected object pixels
- Return current grey level as T



Threshold calculated as $T = 176$ for object occupying 41% of the image
(since histo is darker than by so search from it)

注：由于等该点统计不同intensity对应pixel number的累积数，故不受 reduced lighting 影响.
背景被设为 255 (白)，要分离的前景物设为 0 (黑).
P-tile 的抗噪 (anti-noise) 强.

3. Simple Method Automatic Thresholding
原理：find a threshold between the mean of lower & upper parts of histogram.



遍历前景/背景的intensity分界 i.

取这些i中，前景，背景的 average intensity 的平均最接近 i 的i做对应的 i.

即：让分界线尽可能向所划分的前景，背景平均的平均接近.

4. Otsu's method (大津法) Automatic Thresholding
原理：当前阈值 i，寻找使类内方差最小 (类间方差最大) 的取值 i.

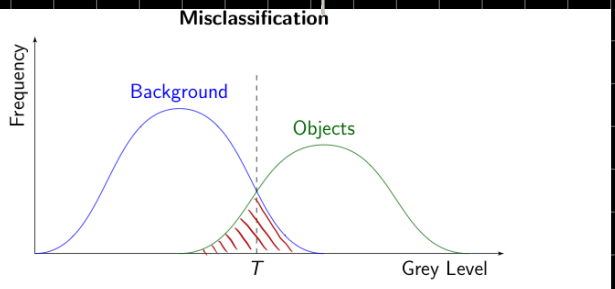
```
// Perform thresholding with a supplied threshold value
threshold(inputImage, outputImage, 128, 255, THRESH_BINARY);
// Save new image
imwrite("supplied_threshold.png", outputImage);

// Perform thresholding with Otsu threshold value
threshold(inputImage, outputImage, 0, 255, THRESH_BINARY | THRESH_OTSU);
// Save new image
imwrite("otsu_threshold.png", outputImage);
```

don't care when performing Otsu thresholding

C code for thresholding with Otsu's method in OpenCV

5. Thresholding 优劣用称的条件
条件：将 objects 和 background 相分离 (不是 foreground 和 background).
亦即：minimise the misclassification rate.



The histograms of background and objects overlap so some background pixels will be classified as objects and vice versa

4. 基于 Pointwise Operation 和，对图形的 Geometrical Transformations
1. 仿射变换 Affine Transformation
定义：保持直线和平行关系不变的几何变换.
- 缩放 Scaling
- 平移 Translation
- 对称 Reflecting / flipping
- 错切 Shear
- 旋转 Rotation

缩放：直接乘上 scaling vector.

Scaling 缩放：乘以缩放向量.

scaling vectors

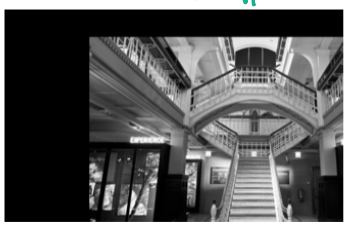
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$

e.g. $s_x = 2, s_y = 2$
$(1,0) \rightarrow (2,0)$
$(2,0) \rightarrow (4,0)$
in this case: size of the image is doubled.

(We will discuss interpolation later)
插值

注：缩放（由其是对采样）一般伴随 interpolation 从填充空 pixels.

平移：加/减平移 offset.



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \end{bmatrix}$$

e.g. $t_x = 160, t_y = 60$
$(1,0) \rightarrow (161,60)$
$(2,0) \rightarrow (162,60)$

注：和 CG 章节中一样，为了统一表示 scaling, translation 和其他类型的仿射变换，从此开始引入齐次坐标：
齐次坐标 augment the matrix + vector

Or, using **homogeneous** coordinates throughout which will allow us to multiply transformation matrices together:

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

与 CG 中 model transformation matrix, view transformation matrix 的合成一样，对图像进行的仿射变换也可依次合成为一个复合变换：

Multiplication of Transformation Matrices

For example, we have a matrix for a translation and one for a scaling

We want to scale and then translate

We can multiply the matrices together (in reverse order) to obtain a single matrix to apply to the image
because right matrix is timed first i.e. the effect is applied first

$$T = \underbrace{\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{translate}} \underbrace{\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{scale}} = \begin{bmatrix} s_x & 0 & t_x \\ 0 & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Notice that the result is the same as the matrix on the previous slide

Scale, Shear, Rotate, Translate

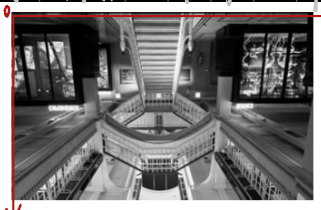
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

As we have seen on previous slides, the different matrix entries are responsible for the following transformations:

- a, e → scaling
- b, d → shearing
- a, b, d, e → rotation
- c, f → translations

所有仿射变换的变换 matrix 表示的可归一化.

镜像翻转：一般对 x 或 y 或 x 和 y 坐标取 negation.



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ -y \\ 1 \end{bmatrix}$$
仍用齐次坐标

Note It is usual to translate the image centre to the origin, perform the transformation and then translate back. In this case for an image with height 480:
翻转一支林一翻转回来

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 480 \\ 0 & 0 & 1 \end{bmatrix}$$

为何保翻转后仍在图片范围内.

错切变换：

将位一点 (x, y) 映射到 $(x + my, y)$ 或 $(x, y + mx)$ 的变换.

注：不scale x和y本身

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \lambda_x & 0 \\ \lambda_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + 0.8y \\ y \\ 1 \end{bmatrix}$$

基于x轴的错切

即：平行于x轴 or 平行于y轴的错切。
将角一点沿 x (y) 轴移动关于该点纵(横)坐标比例的距离。

x轴错切 $\begin{cases} a(x,y) = x + d_x y \\ b(x,y) = y \end{cases}$ x,y ⇒ a(x,y), b(x,y)

In this case $\lambda_x = 0.8$, $\lambda_y = 0$

Remember It is usual to create a matrix to translate the image centre to the origin, perform a transformation and then translate back

错切 → 变换 → 错切回来

旋转变换：将图像中每一点沿图像坐标系原点旋转。

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

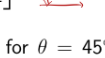
For $\theta = 45°$:

$$T = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 0 \\ \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

To rotate about image centre, create a matrix that translates centre to origin, rotates and translates back

由极坐标角度进行三角变换

$$R = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



Single matrix example for $\theta = 45°$, image $= 640 \times 480$:

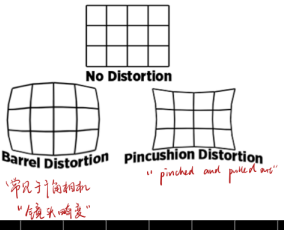
$$T = \begin{bmatrix} \sqrt{2}/2 & -\sqrt{2}/2 & 263.5 \\ \sqrt{2}/2 & \sqrt{2}/2 & -156.0 \\ 0 & 0 & 1 \end{bmatrix}$$

注：1. 旋转后(通常)会插值。

2. 回忆：图像坐标系原点在左上角！
故若要沿图像中心旋转，应：
先将图像平移(translate)使图像中心和原点重合

执行关于原点的 rotation 变换后再平移回来。

2. 非仿射变换：camera lens distortion (畸变)



"常见于广角相机"
"镜头畸变"

Radial and Tangential distortions

Radial distortion

(r is the radius from the image centre, assuming here that the origin is the image centre)

$$\begin{cases} x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots) \\ y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots) \end{cases}$$

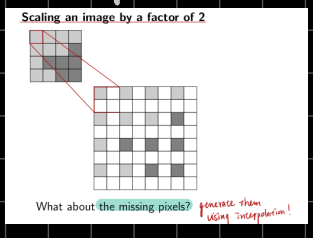
Tangential distortion

$$\begin{cases} x_{corrected} = x + (2p_1 xy) + p_2(r^2 + 2x^2) \\ y_{corrected} = y + p_1(r^2 + 2y^2) + (2p_2 xy) \end{cases}$$

注：opencv 用"distortion matrix"为镜头畸变参数。

5. 插值：Interpolation

Usage：为 scaling, rotation 等可能产生 missing pixels 的变换善后。



使用 interpolation 方法填充这些 missing pixel.

本质：利用近邻现有 pixel 的信息"近似"出 missing pixel 应有的信息。

种类：

nearest neighbor ~ 取近邻的信息。

linear ~ 基于近邻的信息，作线性平均近似。(take 2 known pixels, set unknown one in between)

cubic ~ 基于现有信息，拟合多项式。(look for more known pixels to make better choice)

像素映射问题

problem：对 rotation，原图中的 pixel 在 rotation 后坐标和 image matrix 可能无法一一对应。

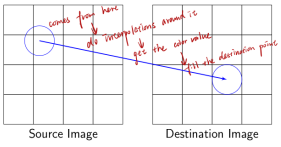
solution：对新图中每个位置上的像素，基于旋转后(theoretically)的原图计算最可能的 color 值。

"从哪来?"

Solution

Rather than cycling through source pixels and calculating a destination pixel,

cycle through destination pixels and calculate their values from source pixels...



For each destination pixel, calculate where it came from and then use interpolation to calculate the pixel value

3. Region-Processing Techniques.

引入：除了单纯的，面向单个像素的 Pointwise Processing，还可将图像局部 / 图像整体视为处理对象.

处理的主要问题：edge detection, denoise, pattern matching.

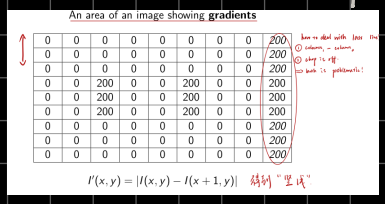
定义：将涉及不止一个 pixel 的 image processing 为 Region Processing.

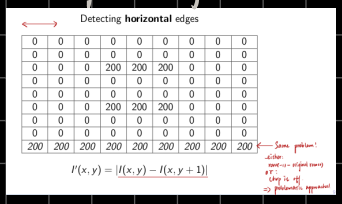
1. Simple Edge detection

原理：计算水平 / 垂直方向上的 pixel intensity 梯度：

vertical line detection: $I'(x,y) = |I(x,y) - I(x+1,y)|$.

horizontal line detection: $I'(x,y) = |I(x,y) - I(x,y+1)|$.





注：在两个方向上的梯度计算均会碰到"最后一行/列无后续行/列可减"的问题.

两种 workaround：
1. 用原来的第n行/列作为第n+1的减数.
2. 直接舍弃：chop it off.

Both workarounds are problemorific.

2. Way to combine horizontal & vertical gradient to find all edges:

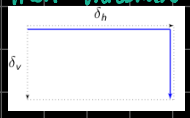
1. OR the gradients. (注：适用于 binary image，梯度只为 1 或 0)

$$I'(x,y) = I_H(x,y) \vee I_V(x,y)$$

2. Add the (absolute value of) horizontal & vertical gradients then threshold.

$$I'(x,y) = |I(x,y) - I(x,y+1)| + |I(x,y) - I(x+1,y)|$$

(City-block distance)

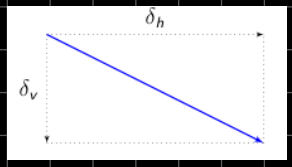


3. Find the magnitude of the gradient vector then threshold.

Find the magnitude of the gradient vector and threshold:

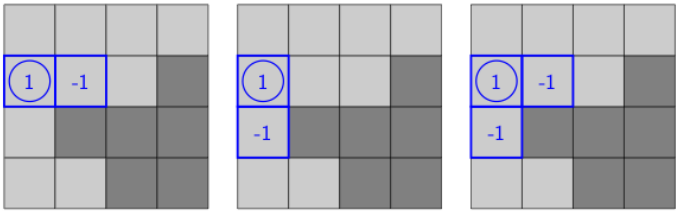
$$I'(x,y) = \sqrt{(I(x,y) - I(x,y+1))^2 + (I(x,y) - I(x+1,y))^2}$$

(Euclidean distance)



thresholding: 可用于去除 weak edges.

Combine horizontal and vertical edge detectors into one operation



$$I'(x,y) = \text{threshold}\left(|I(x,y) - I(x,y+1)| + |I(x,y) - I(x+1,y)|, T\right)$$

此处的 Simple Edge Detection 用了第(2)种：Manhattan Distance.

2. Convolution 卷积

卷积搭配不同的 convolution kernel 可以完成如 blurring, edge detection, pattern matching, denoise 等复杂的 region processing.

符号化表示：

$$I'(x,y) = W \otimes I(x,y) = \sum_{s=-a}^{a}\sum_{t=-b}^{b} W(s,t)I(x-s,y-t)$$

applied on all pixels; mask each pixel with kernel

w_0	w_1	w_2
w_3	w_4	w_5
w_6	w_7	w_8

以给定 pixel: $I(x,y)$ 为中心，对含它在内的周边像素和卷积核作线性组合。

记卷积符号为 $\otimes$ or $*$.

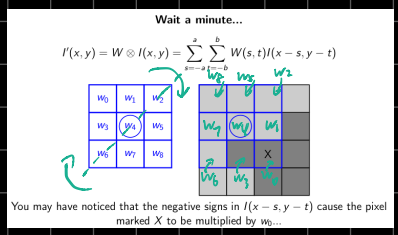
注：1. kernel is applied to the image at every pixel. (和ML不同，学术5设为1)

2. 卷积进行的运算实质上是 kernel 覆盖范围内每个 pixel 的 intensity 和 kernel 上对应权重的线性组合。

3. result: placed into the image pixel under the kernel center.

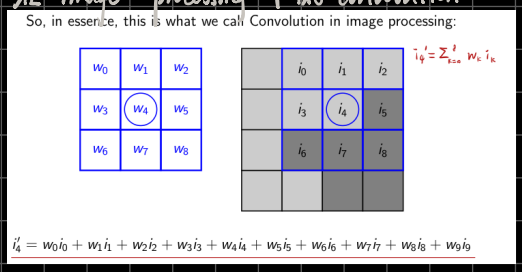
4. 真正的 convolution 会 flip the kernel diagonally. image processing 中的 convolution 实际上是不会 flip kernel 的 correlation.

> **Correlation and convolution in image processing**
>
> Convolution and correlation are similar mathematical operations. **Correlation is also a convolution operation between the two signals but one of the signals is the functional inverse.** So, in correlation process one of the signals is rotated by 180 degree. This is the basic difference between convolution and correlation. 16 Aug 2017

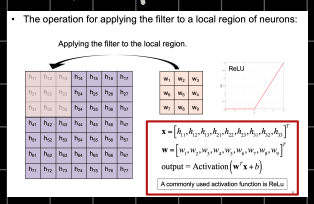


real convolution: 沿反对角线 翻转卷积核.

⇒ 在 image processing 中的 convolution:

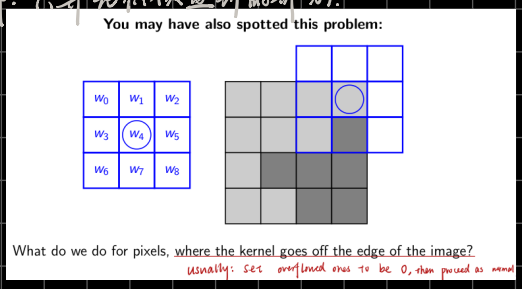


定义和ML中是一致的.

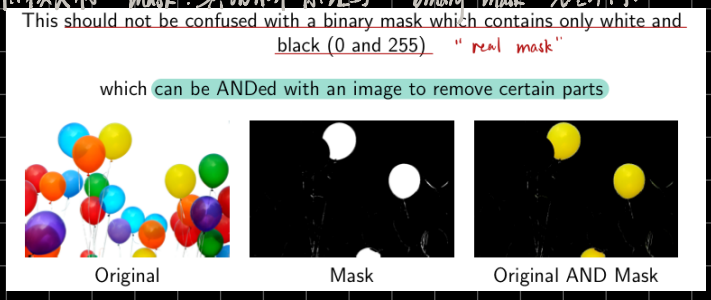


5. 卷积核常为奇数边长的方阵，这样便于确定卷积中心.

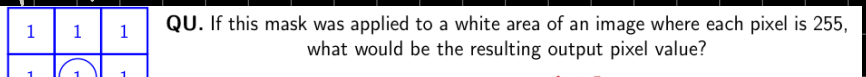
6. 在 kernel goes off the edge of image: 超出部分 intensity 认为是 0，照常计算.
(即：只算卷积核盖到的部分).



7. 卷积核又称 "mask". 其功能和原理与 " binary mask" 完全不同：



8. 为避免统计算后 kernel center 位置的 pixel intensity 超过 255, (overflow), 应确保卷积核是 normalized 的.



1	1	1
1	1	1

$px = 255 \times 9$ OVERFLOW **Too big for an unsigned byte**

1. 卷积满足结合律.

Convolution is **associative**:

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C$$

(A and B are kernels and C is an image)

Convolving an $M \times N$ image with a 3×3 kernel takes

$9MN$ multiplications and additions

Convolving an $M \times N$ image with a 1×3 and then a 3×1 kernel takes

$2 \times 3 \times MN = 6MN$ multiplications and additions

= **Efficiency gain** $\frac{4}{9}MN$ or: $\frac{1}{4}MN$ (be more efficient!)

Qu. What would these results be for a 7×7 kernel?

卷积核大小: $r \cdot c$
图像大小: $x \cdot y$
时间复杂度: $O(r \cdot c \cdot x \cdot y)$.

3. 卷积运算的应用.

1. Smoothing: reduce noise

目标: 减少 Image 中的 noise.

定义 (noise):

the derivation of a value from its expected value.

噪声的类别:

- Random changes
  - $x \to x + n$
  - n can be positive or negative
  - Random distribution and mean is zero (gaussian distribution)
  - Usually much smaller than max(x) "doesn't change much"
- Salt and pepper
  - $x \to \{\max, \min\}$ randomly
  - Much less common

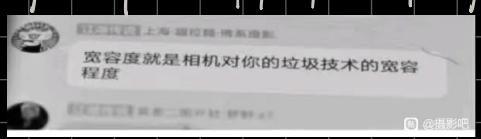
"Image transmitted through some communication channels and get some noise generating spikes affect the signal".

- Imaging artefacts
  - Streaks and blooms

Random changes $\sim N(0, \sigma^2)$

Salt & Pepper $\sim x \to \{\max, \min\}$.

Image artefacts ~ Improper operation of Imager.



噪声的来源:

(preferred definition)

- Anything within the imaging system that causes a change
  - Electrical interference 电路干扰
  - Optical aberration 光学偏差.
- Anything that causes a change
  - Atmospheric disturbance

⇒ { system noise (file transfer corrupted, electric, ...)
external noise (heat, cloud, fog, ...)

降噪 Noise Reduction

1. Smoothing:

$\Sigma(x+n) = \Sigma(x) + \Sigma(n) \approx \Sigma(x)$ since noise is random + zero mean, $\Sigma(n)$ cancels itself.

2. Local Smoothing:

利用卷积核, remove detail + introduces ringing (?)

3. Smoothing temporally (in time)

通过"取多张 image 平均"的方式 cancel 掉 noise.

不同的 smoothing 方法:

1. Box filter: averaging kernel.

A **box blur** (also known as a box linear filter) is a spatial domain linear filter in which each pixel in the resulting image has a value equal to the average value of its neighboring pixels in the input image. It is a form of low-pass ("blurring") filter. A 3 by 3 box blur ("radius 1") can be written as matrix

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Box filter's side effect:
blur the image, lose sharp boundaries.

2. Adaptive Smoothing：只有和平滑后的value和原来差别不大时才应用该值。实质：box filter + threshold。

For each pixel in image, g:

- Compute smoothed value, s
- $g' = \begin{cases} s & |s-g| < T \\ g & \text{otherwise} \end{cases}$

if the smoothed value is not far away from current one: since noise is usually small

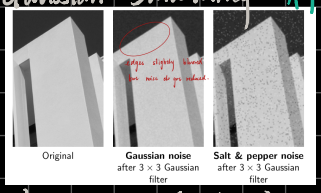
this pixel doesn't need to be smoothed!

Idea: noise is usually small.

表现：可解决 gaussian，edge is good and sharp.

但由于算理限制无法处理由极端值组成的 salt & pepper noise.

3. Gaussian Smoothing：利用 gaussian (distribution) function 生成 filter kernel.



Usage: 1> reduce ringing (gaussian kernel 有抹平极端值的作用).

2> weighted smoothing (视 gaussian filter kernel 中的值为 weight)

(weight derived from Normal Distribution).

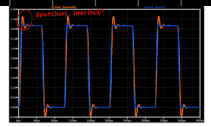
(gaussian filter doesn't introduce ringing at edges).

注：ringing (振铃效应)：

在信號處理中，特別是數位影像處理，**振鈴效應**是一種出現在信號快速轉換時，附加在轉換邊緣上導致失真的信號。而在圖像或影像上，振鈴效應會導致出現在邊緣附近的環帶或像是"鬼影"的環狀偽影；在音訊中，振鈴效應會導致出現在短暫音附近的回聲，特別是由打擊樂器發出的聲音；最容易注意到的是預回聲。使用"振鈴"這一個詞則是因為輸出信號在輸入信號快速轉換的邊緣附近出現一有一定衰減速度的震盪，這個現象相似於鐘被敲擊之後發出聲音的過程。振鈴效應就如同其他的失真一樣，他們的最小化在濾波器設計中是很重要的一項指標。

目录 [隐藏]

圖片顯示為振鈴效應產生的環狀偽影。在顏色變換的兩邊各有3個等級：過衝，第一環，和(較微弱的)第二環。



Ringing in 1D

wi gradually reducing from center.

$$\begin{bmatrix} 0.1021 & 0.1153 & 0.1021 \\ 0.1153 & 0.1304 & 0.1153 \\ 0.1021 & 0.1153 & 0.1021 \end{bmatrix}$$

$\delta = 4$

A Gaussian Kernel (sd = 4)

振铃效应：边缘失真.

4. Rank Filter 中值滤波.

Median Filter

Like convolution, we move a rectangle around over every pixel in the image

5	9	6	
191	11	4	
6	2	12	

the median of:
1> all values in mask
2> sorted
(the sorted array containing all values in mask)

but the centre pixel is set to the **median** value...

去除极端值的影响.

还用于排除 salt & pepper.

4. Edge detection

1. "Edge" 的基于 intensity gradient 的定义

edge: extended, significant, local change in image intensity.

2. Gradient in Images:

A region of an image:

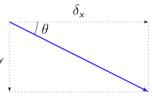
I_0	I_1	I_2	I_3
I_4	I_5		

We find the (signed) horizontal and vertical gradients at I_0 as follows:

$$\delta_x = I_1 - I_0$$
$$\delta_y = I_4 - I_0$$

- Gradient vector at pixel (x, y) is

$$\nabla = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}$$



- Magnitude of the gradient vector is 大小.

$$M(x,y) = \text{mag}(\nabla) = \sqrt{\delta_x^2 + \delta_y^2}$$

- Direction of the gradient vector is 方向.

$$\theta(x,y) = \arg(\nabla) = \tan^{-1}\left(\frac{\delta_y}{\delta_x}\right)$$

梯度向量
- gradient magnitude
- direction

利用卷积模板的 edge detection 利用了 gradient：

1. Roberts Cross Edge Detector：仅用于检测斜线.

Proposed by **Lawrence Roberts** in 1963

-1	0	0	-1
0	1	1	0

是对角线.

- Simplest edge detector kernel
- Awkward edge localisation
  On the joint between the four pixels then not good...
- Noise sensitive
  If one pixel is corrupted, the edge strength is equally corrupted

1. simplest, 2D detector
2. not very good, especially at joint between 4 pixels
3. noise sensitive

2. Prewitt Operator：

horizontal			vertical		
-1	-1	-1	-1	0	1
0	0	0	-1	0	1
1	1	1	-1	0	1

- Edge location estimate is at the centre pixel
- More robust against noise (see later slide)

记录上下位置相差. 显著则说明是水平方向线.

记录左右位置相差. 显著则说明是竖直方向线.

注：prewitt 同时有 reduce noise 的功效：

Why does the Prewitt filter reduce noise?

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \otimes \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

The Prewitt kernel separates into an **averaging filter** that smooths along the rows

and an **edge detector** which finds horizontal edges

(For information, convolution is commutative: $f \otimes g = g \otimes f$)

Prewitt 可拆成一个 averaging filter 和一个 edge detector.

3. Sobel Operator

more significant weights?

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

记录上下色异面差异显著则说明是水平方向线.

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

记录左右方向面差异显著则说明是竖直方向线.

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

The Sobel kernel separates into a **weighted average filter** that smooths along the rows

and an **edge detector** which finds horizontal edges

The results are very similar to the Prewitt filter

同样，sobel operator 也可拆成为一个 weighted average filter 和 edge detector.

其效果和 Prewitt 非常相似，但可得到更清晰的边缘.

4. Canny Edge Detector

目标：1. accurately find as many edges as possible and not falsely detect edges.
2. edge point detected should accurately localise on the center of the edge. (见 canny 的第3, 4, 5步)
3. a given edge in the image should only be marked once.

流程：

Step 1. Gaussian Blurring:

Gaussian function - 2D

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Note We can also have a different σ so that we can blur different amounts in the x and y directions

$$G(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}\right)}$$

使用 Gaussian kernel 降噪 (做 gaussian smoothing) $I(x,y)$

目标：reduce noise before the edge detector.

Step 2. Apply edge detector

3. Apply **edge detector** to smoothed image (can use Roberts, Prewitt, ...) and calculate **gradient magnitude** and **direction** for each pixel
(These are the edge normals)

目标：利用 edge detector 检出边，并计算每条边的 edge normal.

注：edge normal ~ gradient magnitude + direction.

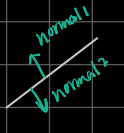
$M(x,y)$ $\theta(x,y)$

Step 3. Classify edge normals.

4. Classify the edge **normals** (gradient vectors) in $\theta(x,y)$ into one of the four categories shown, corresponding to the edge directions:
horizontal edge (0/180°)
vertical edge (90/-90°)
45° edge (135/-45°)
-45° edge (45/-135°)
Each pixel in $\theta(x,y)$ now only contains one of four values for its direction

将 edges 基于 direction 分为 4 类：horizon, vertical, 45°, -45°.

注：an edge has two normals:



Step 4. Non-maxima Suppression.

让"粗线变细"

Algorithm - Part 3

Non-maxima suppression (edge thinning)
5. Create an image $M_{thin}(x,y)$ to contain thinned edges
6. **For** each pixel in $M(x,y)$
Find its direction from $\theta(x,y)$
Find the direction of its 8 neighbouring pixels
If two or more neighbouring pixels in the same direction have a higher magnitude
then $M_{thin}(x,y) = 0$
otherwise $M_{thin}(x,y) = M(x,y)$

direction!

$M_{thin}(x,y)$

目标：识别出像的"主轴"，让"粗线变细".

若斜后更共代表边则将自身设为0.

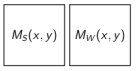
Step 5, 6: "断线回填".

Double Thresholding

Algorithm - Part 4

7. Have **two thresholds**, T_L and T_H
   Magnitudes below T_L are not edges
   Magnitudes between T_L and T_H are **Weak** edges
   Magnitudes above T_H are **Strong** edges

8. Create two binary images using the thresholds:
   $M_S(x,y) = [M_{thin}(x,y) >= T_H]$
   $M_W(x,y) = [M_{thin}(x,y) >= T_L]$



(Binary images)

9. Because T_H is higher than T_L, $M_W(x,y)$ will contain pixels that are in $M_S(x,y)$, so delete them:
   $M_W(x,y) = M_W(x,y) - M_S(x,y)$

Connectivity Analysis

Algorithm - Part 5

10. Assume all pixels in $M_S(x,y)$ are valid edge pixels (There will likely be gaps in the edges in $M_S(x,y)$ so we try to fill them with pixels from $M_W(x,y)$)

11. **For** each 'set' pixel in $M_S(x,y)$
    If any of its 8 neighbours in $M_W(x,y)$ are set,
    Then set those same pixels in $M_S(x,y)$

Final binary image ■

總結:

1. Smooth the Input Image with Gaussian filter
2. apply edge detector, compute gradient magnitude and direction images
3. apply non-maxima suppression to the magnitude image (thinning the edge to its center)
4. use double thresholding & connectivity analysis to detect & link edges (重連)

註:

1. T_H 比較 T_L 嚴 2~3倍. (T_H, T_L: strong / weak line magnitude's classification threshold)
2. Canny are less noisy, lines mostly 1 pixel wide, edges are more connected than other edge detectors. (e.g. Sobel)

Threshold levels chosen to achieve similar edges around neck and ear


Jackie Chan


Sobel, threshold = 50

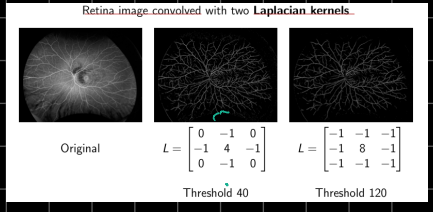

Canny, low = 40, high = 80

Canny appears less noisy, edges are mostly one pixel wide and edges appear more 'connected'

5. 利用 Laplacian 差分核檢測也:

原理: Laplacian 核計算 edge normal 中 direction 的二階導數.
⇒ zero-crossing point between a positive and negative gradient is the location of the edge.

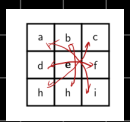
(由 Marr-Hildreth Algorithm)



特徵: 和 Roberts, Prewitt, Sobel 不同, single Laplacian kernel 可檢測全方向的也.
但 Laplacian kernel 的副作用是 highlight noise.

優化: Marr-Hildreth Algorithm (結合 Gaussian 和 Laplacian)

1. 先應用 Gaussian Blurring, 對冲 Laplacian 突出 noise 的副作用.
2. 將模糊後的 image 套用 laplacian kernel.
3. 檢查圖上每个像素点的 gradient 符号:

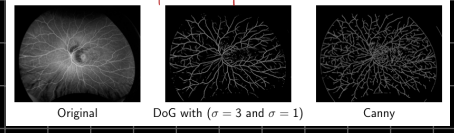


Zero-crossing pixels are marked as edge pixels:

- Pixel **e** is an edge pixel if there is a differing sign on **any** of the pairs b/h, d/f, a/i, c/h

b. 利用 difference of gaussian:

The Difference of Gaussians is performed by blurring an image with two different sigmas and then subtracting one from the other
take the difference (result of subtraction) as the result.



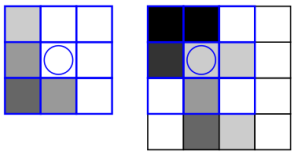
5. Template Matching

任务: given a template, find it in an image.

原理: use cross correlation to find the similarity between template and image.
(cross correlation find similarity between signals)

$$I'(x,y) = W \otimes I(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} W(s,t) I(x+s, y+t)$$

do cross correlation use template as the kernal



- As we having been doing with filtering, move the template around the image
- Multiply the template pixels with the corresponding image pixels — just like convolution do
- Put the sum in the centre pixel

和此前应用 Edge detection 和 blurring 的卷积核一样，对 image 中每个 pixel 均视为卷积中心，应用，卷积核超出 image 的部分视为0.

- The pixel(s) with the highest value will be where the match is closest

issue: ↓ normalisation

But we will get the highest result when the template is placed over a white area of the image (with all values at 255). So,

- **Normalise** the image pixels under the template (by dividing each pixel by the region sum):

$$I'(x,y) = W \otimes I(x,y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} W(s,t) I(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} I(x+s, y+t)}$$

← sum

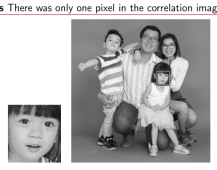
← normalize: divide by region sum.

Result of correlation of image and template

(Black border caused by preventing the template going off the edge of the image)

Correlation result thresholded at 245 to see locations of best match

Success There was only one pixel in the correlation image ≥ 254



Rectangle the size of the template is drawn around the maximum ✓

问题: if there's any variability in the object (image), then matching work badly.
(if the template but not image transforms, the problem also occurs).

This template was taken from the image so it is identical to a region of the image — what if from different images?

If the template was rotated slightly, the match might not occur — prone to (linear) transformations...

Much better results can be achieved with **feature detection**

Template matching is a bad choice for object recognition if

Answers:
1. ✓ There is any variability in the object to be found
2. The object to be found varies in colour
3. The object to be found is quite big
4. The object to be found varies in orientation

4. Image Processing Techniques

1. Blob Finding (connected component analysis / region labeling)

定义 (Blob): a set of pixels that share some property and connected.

关注的 blob property:

"properties (intensity, texture, color, ...) of two adjacent pixels sufficiently similar to infer they are from the same object?"

⇒ Thresholding the grey value.

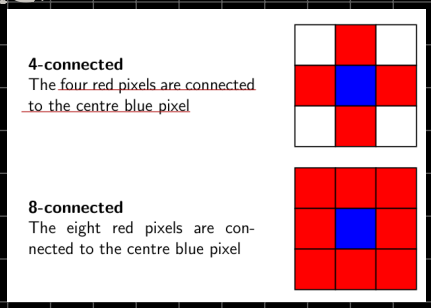
Examples:

- **Thresholding the grey value**
  Pixels' values $\in \{0, 1\}$
  Grouping is simple: pixels are from the same object if they have the same value

only case about is in this course.

- **Other properties** e.g. grayscale histogram
  Can define statistical tests based on blob's properties
  Probability density function (PDF) of grey values or colours
  Can build a classifier to answer the question

连通性:



Connected Component Analysis (Region Labeling):

目标:

1. identify groups of contiguous pixels.
2. label separate blobs.

假设:

背景的 intensity 设为0, object region 编号从1开始.

分析算法:

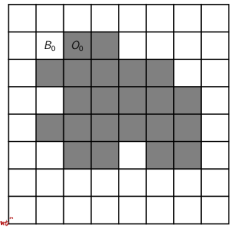
Algorithm

- **First pass** (working from left to right and top to bottom) traverse every pixels
  - If zero neighbours have a label, Pixel receives the next free label
  - If one or more neighbours have the same label, Pixel receives same label (take the lowest one if clash)
  - If two or more neighbours have different labels, Pixel receives one label, equivalence is recorded
- **Second pass** (working from left to right and top to bottom)
  - Re-label all equivalent labels

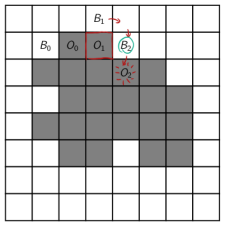
2. Blob 的边缘检测

- Scan the image from left to right and top to bottom
- First object pixel found is the start point O_0 (save location of O_0 so we know when to stop)
- Pixel to the left (must be background) is B_0

"determine the start point"



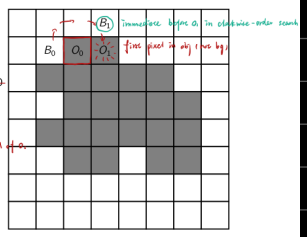
- Continue in this manner...



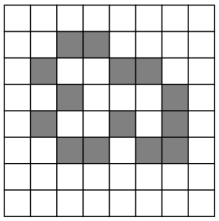
Boundary Detection

Detecting the boundary of a blob

- Go in a clockwise direction around the neighbours of O_0 starting at B_0
- First object pixel found is O_1
- The background pixel found immediately before O_1 is B_1



- ...until we get back to O_0 **AND** the next object pixel found is O_1
- This extra condition is in case the boundary continues off in another direction



"顺时针. B_0 为 O_0 左侧的 pixel.

($i \geq 1$) B_i 为 O_i 的 immediately before pixel. "顺时针", 循环至再次遇到 O_0.

3. 对 Blob 的描述.

Descriptive information:

- moments of area
- chain codes
- color distribution

i) Moments of Area

Formal definition

$$M_{\alpha\beta} = \sum_{\text{image}} x^\alpha y^\beta I(x,y)$$

axis; intensity $I \in [0,1]$.

- For binary image ($I(x,y) \in \{0,1\}$):
  - Letting $\alpha = \beta = 0$ gives sum of pixels
  - Letting $(\alpha, \beta) = (1,0)$ gives sum of x values of the region's pixels (similarly for y)
- $\left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right)$ gives region's centre of gravity (as on previous slide)

"加权和"

"像以 intensity 做 sum. 像"平均"."

- Moments can be defined for all values of α and β
- There's a limit to the number of useful ones
- Can use lower order moments to make higher order ones invariant to:
  - Position
  - Orientation
  - Size of region
- Can compute for non-binary images
- Can modify computation to use labelled blobs
- Values of the moments can be used to discriminate blobs based on size/shape

ii) Central moments of area

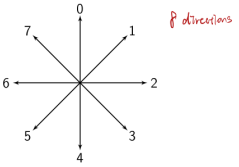
$$M_{\alpha\beta} = \sum_{\text{image}} (x-\bar{x})^\alpha (y-\bar{y})^\beta I(x,y)$$

- $(\bar{x}, \bar{y})$ is the centre of gravity 重心
- Can move the region and the central moments don't change
- Orientation of region is given by $\theta = \frac{1}{2}\tan^{-1}\left(\frac{2M_{11}}{M_{20}-M_{02}}\right)$ "find the angle of tilt"

iii) Chain code

describe the boundary of the blob

8-connected directions for travelling around a boundary (border)



8 directions

Chain code:

1. cyclic
2. not rotationally invariant (用 differential chain code 解决)
3. compare representation

differential chain code:

Look at the **ordinary chain code** we found:

$c = 2\,3\,2\,3\,4\,4\,6\,7\,5\,6\,7\,1\,7\,1$

Now, let us calculate a new sequence as:

$d_n = c_n - c_{n-1} \pmod 8$, $n = 0\ldots$

and we conveniently defined our start direction as 0, so $c_{-1} = 0$

0	1	2	3	4	5	6	7

so, $2-0 = \mathbf{2}, 3-2 = \mathbf{1}, 2-3 = \mathbf{7}, 3-2 = \mathbf{1}, 4-3 = \mathbf{1}, \ldots$

Which just happens to be our **differential chain code**:

$d = 2\,1\,7\,1\,1\,0\,2\,1\,6\,1\,1\,2\,6\,2$

⇒ rotationally invariant, also cyclic (can start anywhere)

⇒ then can compare shapes by comparing the chain code (need to cyclically rotate the codes to fully compare)

iv) Perimeter from chain code 由 chain code 算周长.

奇数 chain code ~ $\sqrt{2}$ D:= 奇数位的数量

偶数 chain code ~ 1 E:= 偶数位的数量

Perimeter: $E + \sqrt{2}D$.

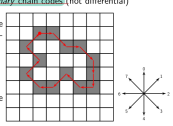
Perimeter from ordinary chain codes (not differential)

Notice that the **odd** chain codes are the hypotenuse of a right-angle triangle with sides equal to 1

Odd chain codes have length $\sqrt{2}$

Even have length 1.

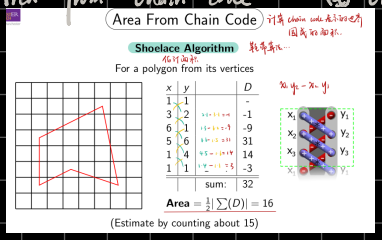
(Basic length measurement is the pixel side)

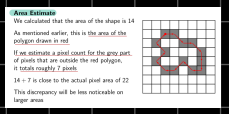


Let E = number of even codes, and D = number of odd codes

Perimeter = $E + \sqrt{2}D$

v) Area from chain code 由 chain code 算面积.





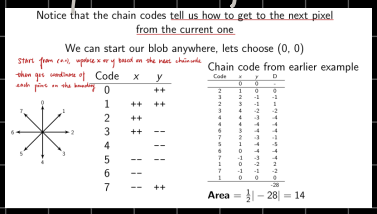
注: area of the polygon

actural area, not pixel count. 略小于 pixel count.

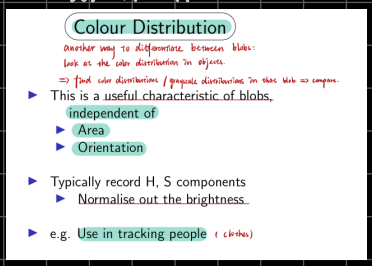
(由坐标算出)

由 chain code 得到 coordinate:

Start from blob anywhere, 记初始点 (0,0). 利用 chain code 行进方向生成每一步后的新坐标:



vi) color distribution



(normalise out intensity, only care chromaticity values)

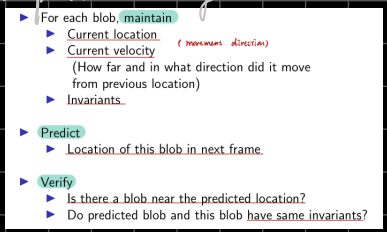
remove perturbations from the outside

6. Blob Tracking

目标:

match a blob in time series: look for invariant properties

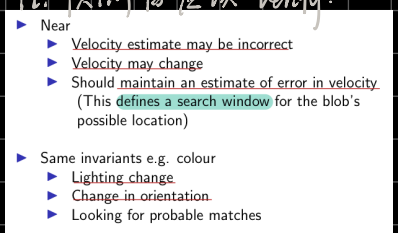
方法: predictive tracking



利用记录的, 每个 blob 的位置, 速度和 invariants

预测"它在下一时刻 (下一帧) 的位置.

注: 预测后应该 verify:



5. Hough Transform

问题: 边上边缘检测后得到的 (本应相连的) 断线.

想象: 给定一系列点, 找到在同一条直线上的那些. => 直线, 实现"断线合并".

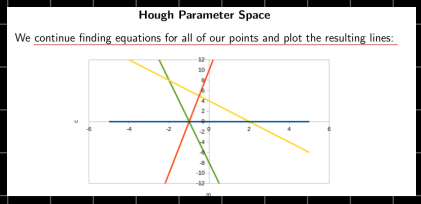
1. Hough lines:

直角坐标系中应用于 Hough lines 的直线表示: $c = -x \cdot m + y$. (c, m 视为变量, x, y 视为给定常量)

极坐标系中: $r = x\cos\theta + y\sin\theta$.

给定点 (x_0, y_0) 可在 Hough Parameter Space 中得直线: $c = -x_0 m + y_0$.

在 Hough Space 中绘制出所有和图像中的前景 pixel 对应的直线. 相交处确定了 c, m 值
⇒ 即: 找到了一条在图像中可能对应的直线.



Hough Param Space 中的每一条直线表示了在图像空间中过点 (x_0, y_0) 的所有直线.
每一个点代表图像空间中的一条直线.

程序化:
确定"最小分割手", 为确可扫描到的 m 和 c.
对前景图像中的点, 在表中填上"在对应的 Hough line 上的, Hough Space 上的点".

We can create a 2D array and use the columns for the values of m and the rows for the values of c

c \ m	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
10									
8									
6									
4									
2									
0									
-2									
-4									
-6									
-8									
-10									

(This range of values has been chosen to allow ease of demonstration)

Point $(-10, 10)$, $c = 10m + 10$:

c \ m	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
10					1				
8									
6				1					
4									
2									
0	1	1	2	1	1	1	1	1	1
-2									
-4									
-6		1							
-8									
-10	1								

Hough Lines

Entries are incremented by using the Hough parameter equations for each point (by rounding where necessary)

Point $(0, 0)$, $c = 0$:

c \ m	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
10									
8									
6									
4									
2									
0	1	1	1	1	1	1	1	1	1
-2									
-4									
-6									
-8									
-10									

Hough Lines

Point $(2, 4)$, $c = -2m + 4$:

c \ m	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
10					1				
8	1	1							
6			1	2					
4					1	1			
2							1	1	
0	1	1	2	1	1	1	1	1	2
-2									
-4									
-6		1							
-8									
-10	1								

Then choose the entries with highest values: 前景 entry 对应图像中在有的一条线.
(一般用 threshold 控制找到的线的数量)

Once this has been done with all of the points, we can choose the entries with the highest values

(We can decide on a threshold – what is the minimum number of points that we want to find on a line to consider it a valid line?)

Hough Lines

If we decide that three points is sufficient then we find one valid line

c \ m	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
10					1				
8	2	1							
6			1	2					
4		1			1	1			
2							1	1	
0	1	1	3	1	1	1	1	1	2
-2									
-4				1					
-6		1							
-8					1				
-10	1								

Reading off its coordinates: $m = -1$, $c = 0$ gives us our line

$y = -x$

同样, 为解决"直角坐标系下的斜率表示 vertical line (斜率为 $+\infty$) 的问题,
可改用极坐标系表示: $r = x\cos\theta + y\sin\theta$. Hough Space 由 r, θ 表示.

Point $(-10, 10)$, $r = -10\cos(\theta) + 10\sin(\theta)$:

r \ theta	-90	-67.5	-45	-22.5	0	22.5	45	67.5	90
10									1
8									
6								1	
4									
2									
0	1	1	1	1	1	1	2	1	1
-2									
-4									
-6						1			
-8									
-10					1				

Point $(8, -8)$, $r = 8\cos(\theta) - 8\sin(\theta)$:

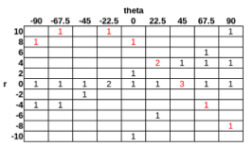
r \ theta	-90	-67.5	-45	-22.5	0	22.5	45	67.5	90
10		1		1					1
8	1				1				
6								1	
4						2	1	1	1
2					1				
0	1	1	1	2	1	1	3	1	1
-2			1						
-4	1	1						1	
-6						1			
-8									1
-10					1				

Hough Lines – Polar

Point $(2, 4)$, $r = 2\cos(\theta) + 4\sin(\theta)$:

r \ theta	-90	-67.5	-45	-22.5	0	22.5	45	67.5	90
10									1
8									
6								1	
4						1	1	1	1
2					1				
0	1	1	1	2	1	1	2	1	1
-2			1						
-4	1	1							
-6						1			
-8									
-10					1				

Hough Lines – Polar



From this, we can see that the highest count occurs at

$\theta = 45°$ and $r = 0$

So this corresponds to a line at $-45°$ and 0 distance from the origin

(Remember that θ is the angle of the normal)

That is, the line $y = -x$

6. Binary Morphology.

Erosion:

- If the centre of structuring element (SE) is over an object pixel and any of the other elements of the SE are over a background pixel,
  ⇒ the pixel under the centre becomes background
- The output must be drawn into a new image

Dilation:

- If the centre of the structuring element (SE) is over a background pixel and any of the other elements of the SE are over an object pixel,
  ⇒ the pixel under the centre becomes object
- The output must be drawn into a new image

- Erosion makes an object smaller than it was
- Dilation makes an object larger than it was

What is often required is that:

- 'sticky out bits' are removed without changing the object size
- 'holes' are filled in without changing the object size

Opening will get rid of 'sticky out bits' and also enlarge holes

It is achieved by first eroding and then dilating:

$$\text{open}(I) = \text{dilate}(\text{erode}(I))$$

Closing will fill in holes and fill out 'sticky out bits'

It is achieved by first dilating and then eroding:

$$\text{close}(I) = \text{erode}(\text{dilate}(I))$$