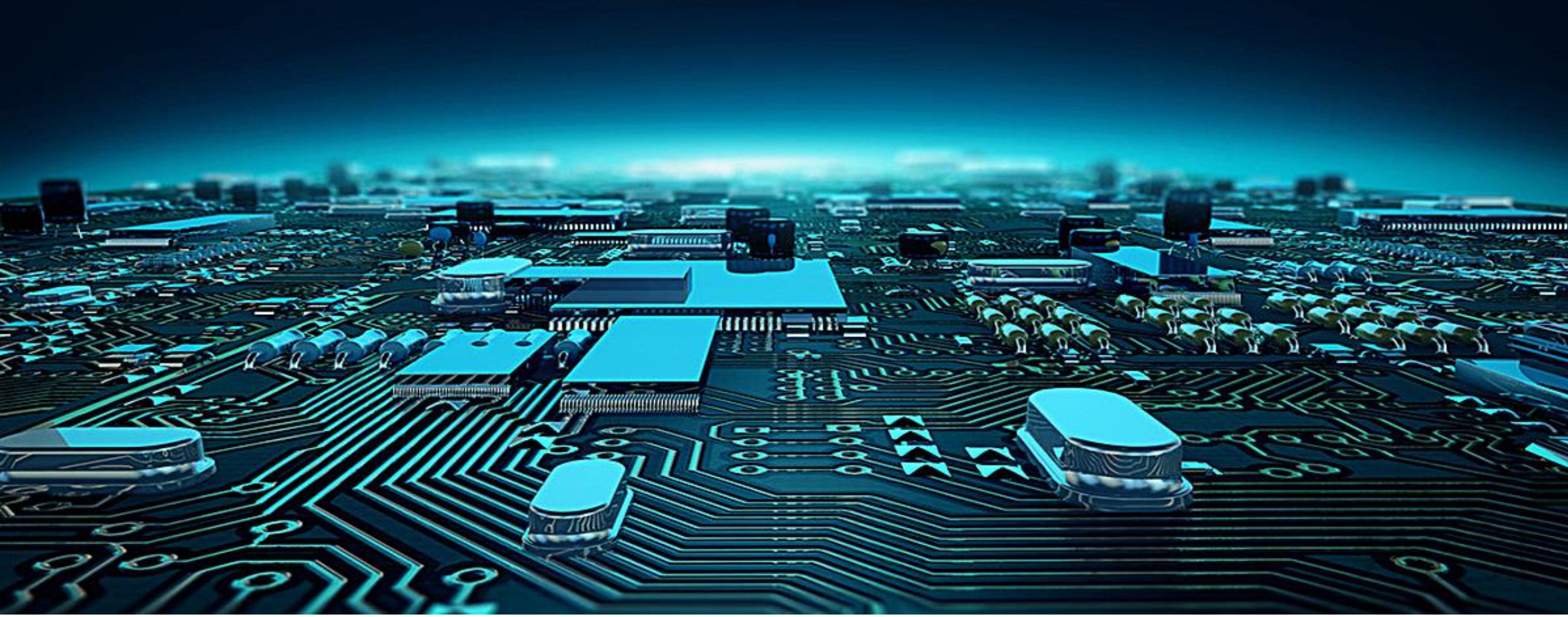
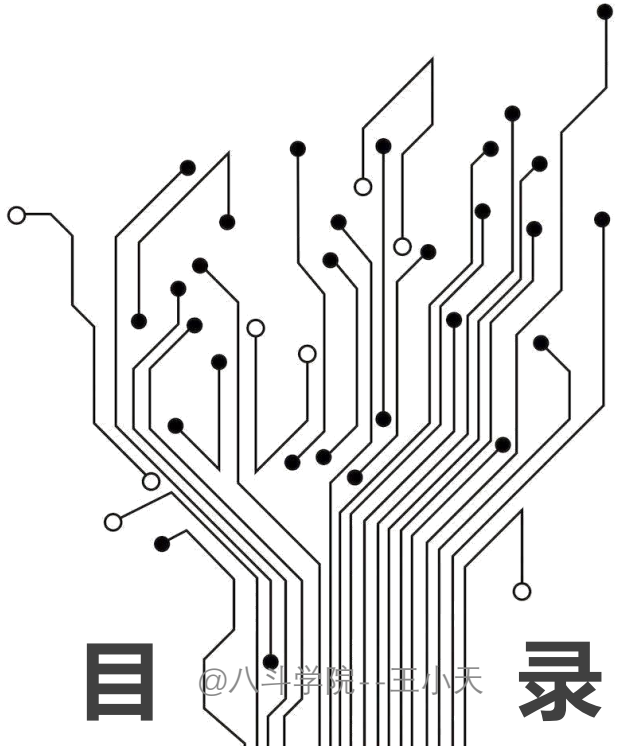


OpenCV算法解析

@八斗学院--王小天(Michael)

2021/12/26



目　录

OpenCV

- OpenCV是一个开源的计算机视觉库，可以从 http://opencv.org 获取。
- OpenCV 库用C语言和 C++ 语言编写，可以在 Windows、Linux、Mac OS X 等系统运行。同时也在积极开发 Python、Java、Matlab 以及其他一些语言的接口，将库导入安卓和 iOS 中为移动设备开发应用。
- OpenCV 设计用于进行高效的计算，十分强调实时应用的开发。它由 C++ 语言编写并进行了深度优化，从而可以享受多线程处理的优势。
- OpenCV 的一个目标是提供易于使用的计算机视觉接口，从而帮助人们快速建立精巧的视觉应用。
- OpenCV 库包含从计算机视觉各个领域衍生出来的 500 多个函数，包括工业产品质量检验、医学图像处理、安保领域、交互操作、相机校正、双目视觉以及机器人学。

opencv大坑之BGR

opencv对于读进来的图片的通道排列是BGR，而不是主流的RGB！谨记！

```
#opencv读入的矩阵是BGR，如果想转为RGB，可以这么转
img4 = cv2.imread('1.jpg')
img4 = cv2.cvtColor(img4,cv2.COLOR_BGR2RGB)
```

1. 除了opencv读入的彩色图片以BGR顺序存储外，其他所有图像库读入彩色图片都以RGB存储。
2. 除了PIL读入的图片是img类之外，其他库读进来的图片都是以numpy 矩阵。
3. 各大图像库的性能，最好的OpenCv，无论是速度还是图片操作的全面性，都属于碾压的存在，毕竟他是一个巨大的cv专用库。

a test with an image (image_size = (3120,4160)), run 100 times

load image function	load image time(s)	resize image function	resize image(s)
cv2.imread	14.90	cv2.resize	0.087
skimage.io.imread	18.18	skimage.transform.resize	29.609
caffe.io.load_image	38.01	caffe.io.resize	26.35
caffe.io.load_image(cv2)	22.01	caffe.io.resize(cv2)	7.085

OpenCV常见算法

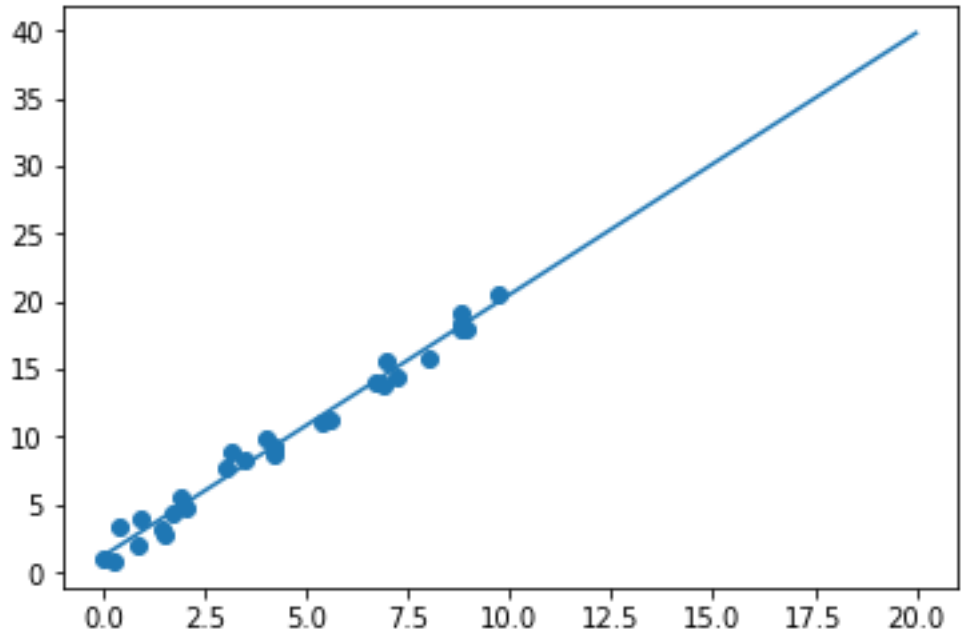
1. 图像的基本操作读取、显示、存储：通过调用OpenCV中的cv2.imread()，cv2.imshow()，cv2.write()分别实现。
2. 在OpenCV中实现将彩色像素转化为灰度像素。
3. 图像的几何变换：平移、缩放、旋转、插值（最近邻、双线性）。
4. 对比增强：线性变换、伽马变换、直方图均衡化。
5. 边缘检测：Sobel、Canny
6. 图像的二维滤波：cvFilter2D

线性回归

什么是线性回归?

举个例子，某商品的利润在售价为2元、5元、10元时分别为4元、10元、20元，
我们很容易得出商品的利润与售价的关系符合直线：y=2x.
在上面这个简单的一元线性回归方程中，我们称“2”为回归系数，即斜率为其回归系数。
回归系数表示商品的售价(x)每变动一个单位，其利润(y)与之对应的变动关系。

线性回归



线性回归表示这些离散的点**总体上**“**最逼近**”哪条直线。

线性回归--最小二乘法（Least Square Method）

- 它通过**最小化误差的平方和**，寻找数据的最佳函数匹配。
- 利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间**误差的平方和**为最小。
- 假设我们现在有一系列的数据点(xi,yi) (i=1,…,m)，那么由我们给出的拟合函数h(x)得到的估计量就是h(xi)
- 残差：ri = h(xi) – yi

线性回归--最小二乘法（Least Square Method）

- 残差：ri = h(xi) – yi
- 三种范数：
  1. ∞-范数：残差绝对值的最大值，即所有数据点中残差距离的最大值：$\max_{1\le i\le m}|r_i|$
  2. 1-范数：绝对残差和，即所有数据点残差距离之和：$\sum_{i=1}^{m}|r_i|$
  3. 2-范数：残差平方和：$\sum_{i=1}^{m} r_i^2$
- 拟合程度，用通俗的话来讲，就是我们的拟合函数h(x)与待求解的函数y之间的相似性。那么2-范数越小，自然相似性就比较高了。

线性回归--最小二乘法（Least Square Method）

由此，我们可以写出最小二乘法的定义了：

$$\min_{a,b}\sum_{n=1}^{N}(y_n-(k\times x+b))^2$$

$$\sum_{i=1}^{m} r_i^2$$

这是一个无约束的最优化问题，分别对k和b求偏导，然后令偏导数为0，即可获得极值点。

$$k=\frac{N\sum_{n=1}^{N}x_n\times y_n-(\sum_{n=1}^{N}x_n)(\sum_{n=1}^{N}y_n)}{(N\sum_{n=1}^{N}(x_n)^2-(\sum_{n=1}^{N}x_n)^2)}$$

$$b=\frac{(\sum_{n=1}^{N}y_n)}{N}-k\times\frac{(\sum_{n=1}^{N}x_n)}{N}$$

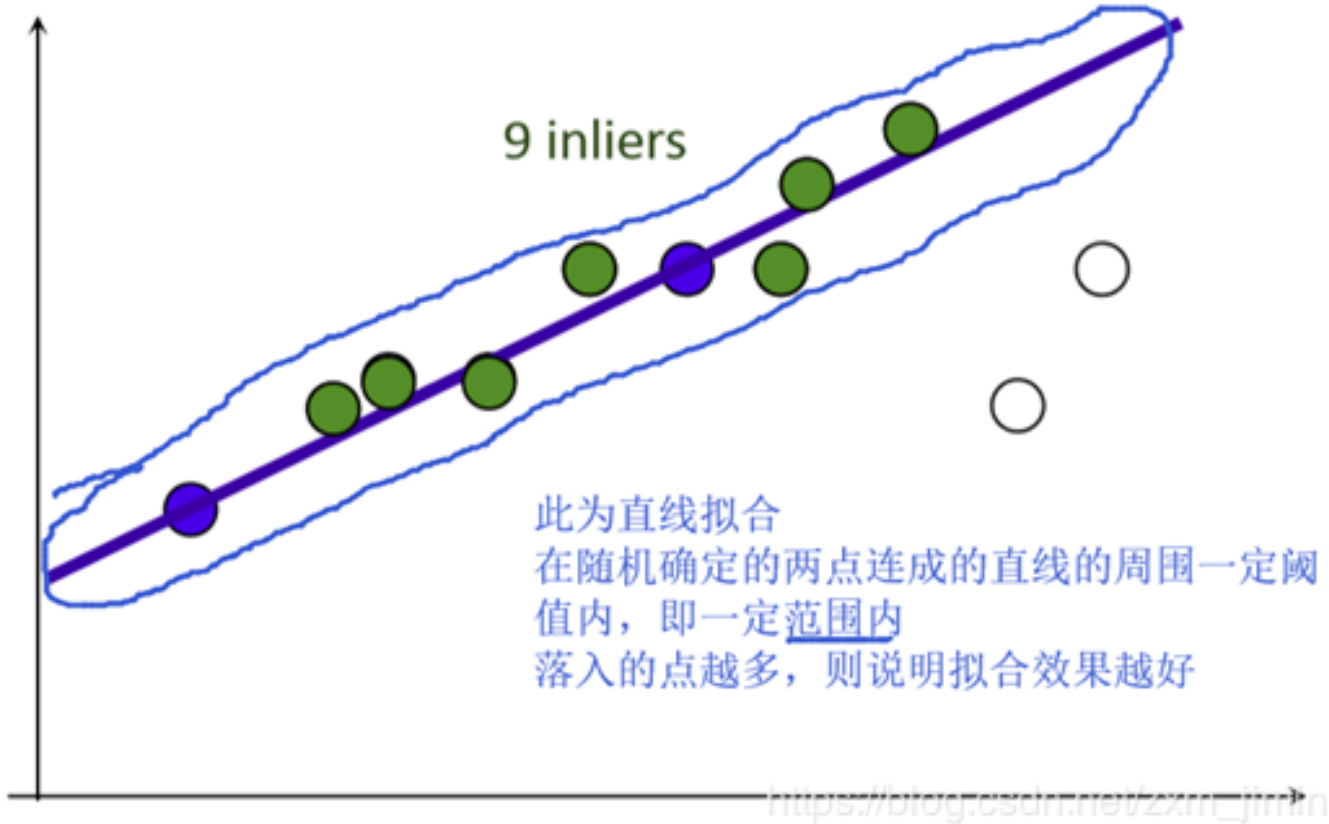
RANSAC

- 随机采样一致性（random sample consensus）。

- **RANSAC是一种思想**，一个求解已知模型的参数的框架。**它不限定某一特定的问题，可以是计算机视觉的问题，同样也可以是统计数学，甚至可以是经济学领域的模型参数估计问题。**

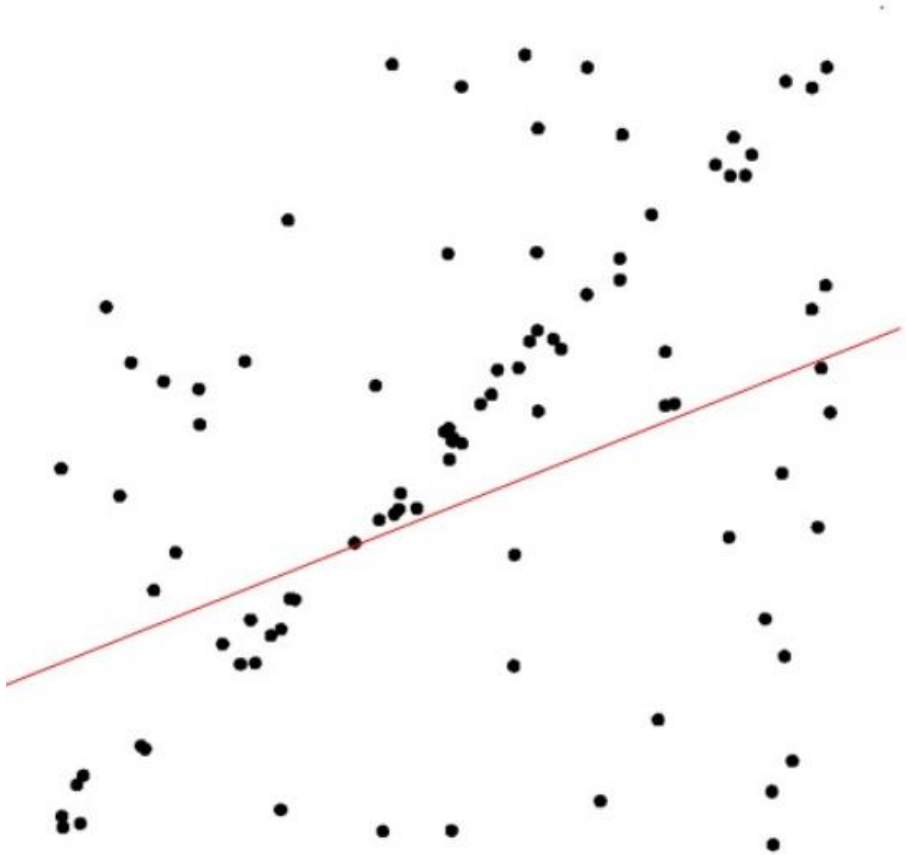
- 它是一种迭代的方法，用来在一组包含离群的被观测数据中估算出数学模型的参数。 RANSAC是一个非确定性算法，在某种意义上说，它会产生一个在一定概率下合理的结果，其允许使用更多次的迭代来使其概率增加。

- RANSAC的基本假设是 “内群”数据可以通过几组模型参数来叙述其数据分布，而“离群”数据则是不适合模型化的数据。 数据会受噪声影响,噪声指的是离群，例如从极端的噪声或错误解释有关数据的测量或不正确的假设。 **RANSAC假定，给定一组（通常很小的）内群，存在一个程序，这个程序可以估算最佳解释或最适用于这一数据模型的参数。**


9 inliers
此为直线拟合
在随机确定的两点连成的直线的周围一定阈值内，即一定范围内
落入的点越多，则说明拟合效果越好

RANSAC与最小二乘法

- 生产实践中的数据往往会有一定的偏差。
- 例如我们知道两个变量X与Y之间呈线性关系，Y=aX+b，我们想确定参数a与b的具体值。通过实验，可以得到一组X与Y的测试值。虽然理论上两个未知数的方程只需要两组值即可确认，但由于系统误差的原因，任意取两点算出的a与b的值都不尽相同。我们希望的是，最后计算得出的理论模型与测试值的误差最小。
- 最小二乘法：通过计算最小均方差关于参数a、b的偏导数为零时的值。事实上，很多情况下，最小二乘法都是线性回归的代名词。
- 遗憾的是，最小二乘法只适合于误差较小的情况。
- 在模型确定以及最大迭代次数允许的情况下，RANSAC总是能找到最优解。（对于包含80%误差的数据集，RANSAC的效果远优于直接的最小二乘法。）
- 由于一张图片中像素点数量大，采用最小二乘法运算量大，计算速度慢。



RANSAC的步骤

RANSAC算法的输入:

1. 一组观测数据（往往含有较大的噪声或无效点），
2. 一个用于解释观测数据的参数化模型，**比如** y=ax+b
3. 一些可信的参数。

RANSAC的步骤

1. 在数据中随机选择几个点设定为内群
2. 计算适合内群的模型 e.g. y=ax+b ->y=2x+3　y=4x+5
3. 把其它刚才没选到的点带入刚才建立的模型中，计算是否为内群 e.g. hi=2xi+3->ri
4. 记下内群数量
5. 重复以上步骤
6. 比较哪次计算中内群数量最多,内群最多的那次所建的模型就是我们所要求的解

注意：**不同问题对应的数学模型不同，因此在计算模型参数时方法必定不同，RANSAC的作用不在于计算模型参数。（这导致ransac的缺点在于要求数学模型已知）**

这里有几个问题：

1. 一开始的时候我们要随机选择多少点(n)
2. 以及要重复做多少次(k)

RANSAC的参数确定

- 假设每个点是真正内群的概率为 w:

$$w = \text{内群的数目}/(\text{内群数目}+\text{外群数目})$$

- 通常我们不知道 w 是多少, w^n是所选择的n个点都是内群的机率, 1-w^n 是所选择的n个点至少有一个不是内群的机率, (1 − w^n)^k 是表示重复 k 次都没有全部的n个点都是内群的机率, 假设算法跑 k 次以后成功的机率是p， 那么,

$$1 - p = (1 - w^n)^k$$

$$p = 1 - (1 - w^n)^k$$

- 我们可以通过P反算得到抽取次数K， K=log(1-P)/log(1-w^n)。

- 所以如果希望成功机率高:
- 当n不变时，k越大，则p越大； 当w不变时，n越大，所需的k就越大。
- 通常w未知，所以n 选小一点比较好。

RANSAC的优缺点

优点：

1. 它能鲁棒的估计模型参数。例如，它能从包含大量局外点的数据集中估计出高精度的参数。

缺点：

1. 它计算参数的迭代次数没有上限；如果设置迭代次数的上限，得到的结果可能不是最优的结果，甚至可能得到错误的结果。
2. RANSAC只有一定的概率得到可信的模型，概率与迭代次数成正比。
3. 它要求设置跟问题相关的阈值。
4. RANSAC只能从特定的数据集中估计出一个模型，如果存在两个（或多个）模型，RANSAC不能找到别的模型。
5. 要求数学模型已知

图像相似度比较哈希算法

相似图像搜索的哈希算法有三种:

1. 均值哈希算法
2. 差值哈希算法
3. 感知哈希算法

图像相似度比较哈希算法

什么是哈希（Hash）？

- 散列函数（或散列算法，又称哈希函数，英语：Hash Function）是一种从任何一种数据中创建小的数字“指纹”的方法。散列函数把消息或数据压缩成摘要，使得数据量变小，将数据的格式固定下来。该函数将数据打乱混合，重新创建一个叫做散列值（hash values，hash codes，hash sums，或hashes）的指纹。散列值通常用一个短的随机字母和数字组成的字符串来代表。

- 通过哈希算法得到的任意长度的二进制值映射为较短的固定长度的二进制值，即哈希值。此外，哈希值是一段数据唯一且极其紧凑的数值表示形式，如果通过哈希一段明文得到哈希值，哪怕只更改该段明文中的任意一个字母，随后得到的哈希值都将不同。

- 哈希算法是一个函数，能够把几乎所有的数字文件都转换成一串由数字和字母构成的看似乱码的字符串。

图像相似度比较哈希算法

哈希函数作为一种加密函数，其拥有两个最重要特点：

1. 不可逆性。输入信息得出输出的那个看似乱码的字符串（哈希值）非常容易，但是从输出的字符串反推出输入的结果却是却非常非常难。

2. 输出值唯一性和不可预测性。只要输入的信息有一点点区别，那么根据哈希算法得出来的输出值也相差甚远。

图像相似度比较哈希算法

哈希算法是一类算法的总称，共有三种：

1. 均值哈希算法aHash
2. 差值哈希算法dHash
3. 感知哈希算法pHash

汉明距离

两个整数之间的汉明距离指的是这两个数字对应二进制位不同的位置的数目。

输入: x = 1, y = 4

输出: 2

解释:

```
1   (0 0 0 1)
4   (0 1 0 0)
        ↑   ↑
```

均值哈希算法

步骤

1. 缩放：图片缩放为8*8，保留结构，除去细节。
2. 灰度化：转换为灰度图。
3. 求平均值：计算灰度图所有像素的平均值。
4. 比较：像素值大于平均值记作1，相反记作0，总共64位。
5. 生成hash：将上述步骤生成的1和0按顺序组合起来既是图片的指纹（hash）。
6. 对比指纹：将两幅图的指纹对比，计算汉明距离，即两个64位的hash值有多少位是不一样的，不相同位数越少，图片越相似。

差值哈希算法

差值哈希算法相较于均值哈希算法，前期和后期基本相同，只有中间比较hash有变化。

步骤

1. 缩放：图片缩放为**8*9**，保留结构，除去细节。
2. 灰度化：转换为灰度图。
3. 求平均值：计算灰度图所有像素的平均值。 **---这步没有，只是为了与均值哈希做对比**
4. 比较：像素值大于后一个像素值记作1，相反记作0。本行不与下一行对比，每行9个像素，八个差值，有8行，总共64位
5. 生成hash：将上述步骤生成的1和0按顺序组合起来既是图片的指纹（hash）。
6. 对比指纹：将两幅图的指纹对比，计算汉明距离，即两个64位的hash值有多少位是不一样的，不相同位数越少，图片越相似。

图像相似度比较哈希算法

三种算法的比较：

- aHash：均值哈希。速度比较快，但是有时不太精确。
- pHash：感知哈希。精确度较高，但是速度方面较差一些。
- dHash：差值哈希。精确度较高，且速度也非常快。

拓展--感知哈希算法

均值哈希算法过于严格，不够精确，更适合搜索缩略图，为了获得更精确的结果可以选择感知哈希算法，它采用的是DCT（离散余弦变换）来降低频率的方法。

步骤：

1. 缩小图片：32 * 32是一个较好的大小，这样方便DCT计算
2. 转化为灰度图：把缩放后的图片转化为灰度图。
3. 计算DCT:DCT把图片分离成分率的集合
4. 缩小DCT：DCT计算后的矩阵是32 * 32，保留左上角的8 * 8，这些代表图片的最低频率。
5. 计算平均值：计算缩小DCT后的所有像素点的平均值。
6. 进一步减小DCT：大于平均值记录为1，反之记录为0.
7. 得到信息指纹：组合64个信息位，顺序随意保持一致性。
8. 最后比对两张图片的指纹，获得汉明距离即可。

拓展--离散余弦变换DCT

- **离散余弦变换(Discrete Cosine Transform)**，主要用于将数据或图像的压缩，能够将空域的信号转换到频域上，具有良好的去相关性的性能。

- DCT变换本身是无损的，同时，由于DCT变换是对称的，所以，我们可以在量化编码后利用DCT反变换，在接收端恢复原始的图像信息。

- DCT变换在当前的图像分析以及压缩领域有着极为广大的用途，我们常见的JPEG静态图像编码以及MJPEG、MPEG动态编码等标准中都使用了DCT变换。

拓展--离散余弦变换DCT

$$F(u,v) = c(u)c(v)\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)\cos\left[\frac{(i+0.5)\pi}{N}u\right]\cos\left[\frac{(j+0.5)\pi}{N}v\right]$$

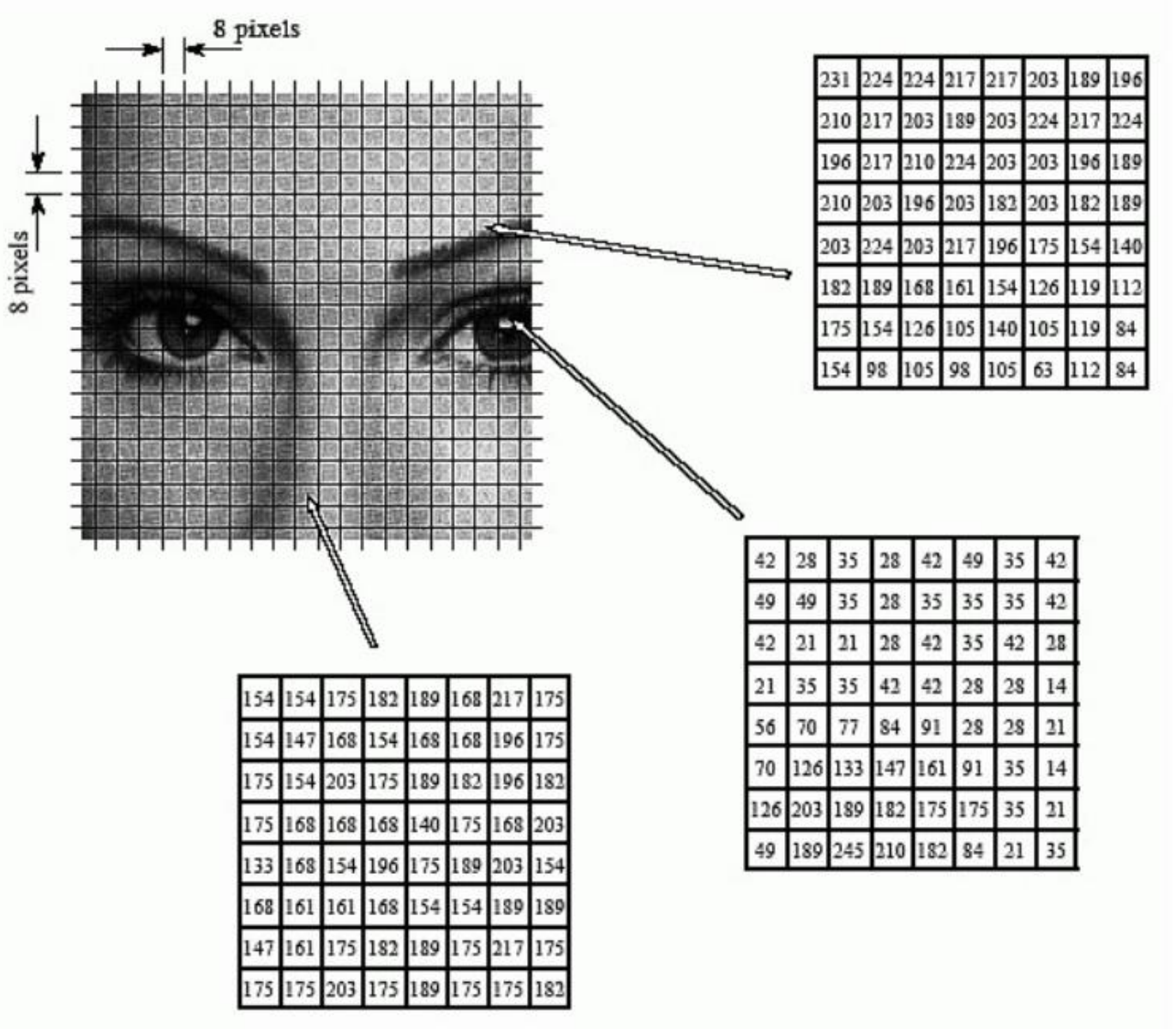
$$c(u) = \begin{cases} \sqrt{\frac{1}{N}}, & u = 0 \\ \sqrt{\frac{2}{N}}, & u \neq 0 \end{cases}$$

其中，

- F(u,v)是输出的变换结果;
- N为原始信号的点数。
- f(i,j)是原图像中像素点(i,j)的像素值;
- c(u), c(v)是DCT系数。

拓展--离散余弦变换DCT

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos\left[\frac{(2x+1)u\pi}{16}\right]\cos\left[\frac{(2y+1)v\pi}{16}\right]$$



拓展--离散余弦变换DCT

应用举例：

某个图象的一个8*8方块，的亮度值。　　为了减少绝对值波动，先把数值移位一下，变成-128~127

$$\begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix}$$

$$g = \begin{bmatrix} -76 & -73 & -67 & -62 & -58 & -67 & -64 & -55 \\ -65 & -69 & -73 & -38 & -19 & -43 & -59 & -56 \\ -66 & -69 & -60 & -15 & 16 & -24 & -62 & -55 \\ -65 & -70 & -57 & -6 & 26 & -22 & -58 & -59 \\ -61 & -67 & -60 & -24 & -2 & -40 & -60 & -58 \\ -49 & -63 & -68 & -58 & -51 & -60 & -70 & -53 \\ -43 & -57 & -64 & -69 & -73 & -67 & -63 & -45 \\ -41 & -49 & -59 & -60 & -63 & -52 & -50 & -34 \end{bmatrix}$$

$x \longrightarrow$　　$y \downarrow$

拓展--离散余弦变换DCT

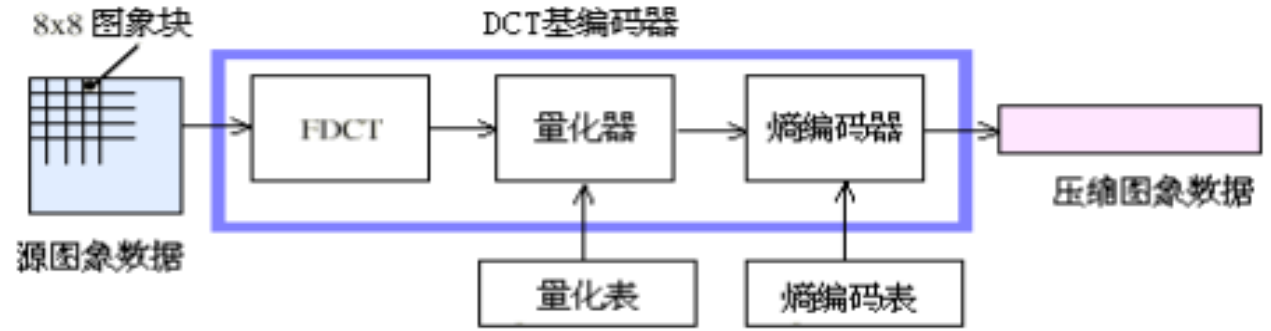
根据DCT变换公式，各种计算，获得临时结果

$$
G = \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}
\begin{matrix} \overset{u}{\longrightarrow} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}
\begin{bmatrix}
-415.38 & -30.19 & -61.20 & 27.24 & 56.12 & -20.10 & -2.39 & 0.46 \\
4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\
-46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\
-48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\
12.12 & -6.55 & -13.20 & -3.95 & -1.87 & 1.75 & -2.79 & 3.14 \\
-7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\
-1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\
-0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68
\end{bmatrix}
\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}
\begin{matrix} \\ \\ \\ \\ \\ \\ \\ \\ \end{matrix}
\Big\downarrow v.
$$

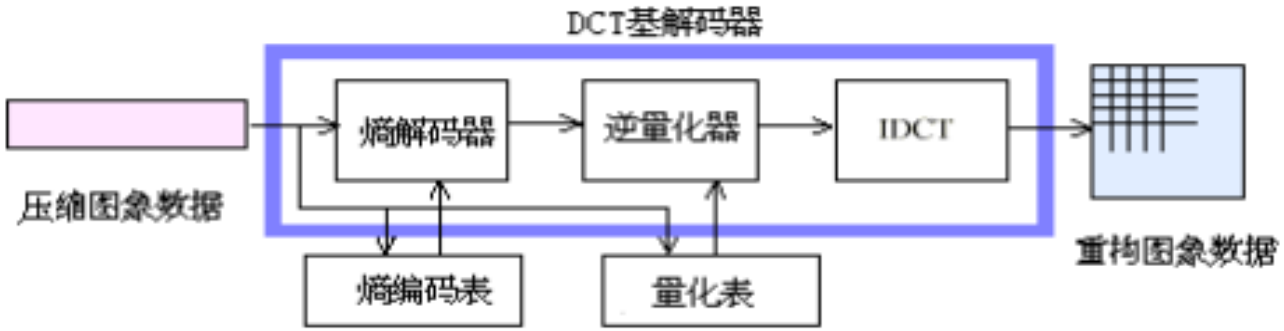
拓展--离散余弦变换DCT

简介DCT在JPEG压缩编码中的应用：

JPEG(Joint Photographic Experts Group) 专家组开发了两种基本的压缩算法，一种是采用以离散余弦变换(DCT)为基础的有损压缩算法，另一种是采用以预测技术为基础的无损压缩算法。使用有损压缩算法时，在压缩比为25:1的情况下，压缩后还原得到的图像与原始图像相比较，非图像专家难于找出它们之间的区别，因此得到了广泛的应用。



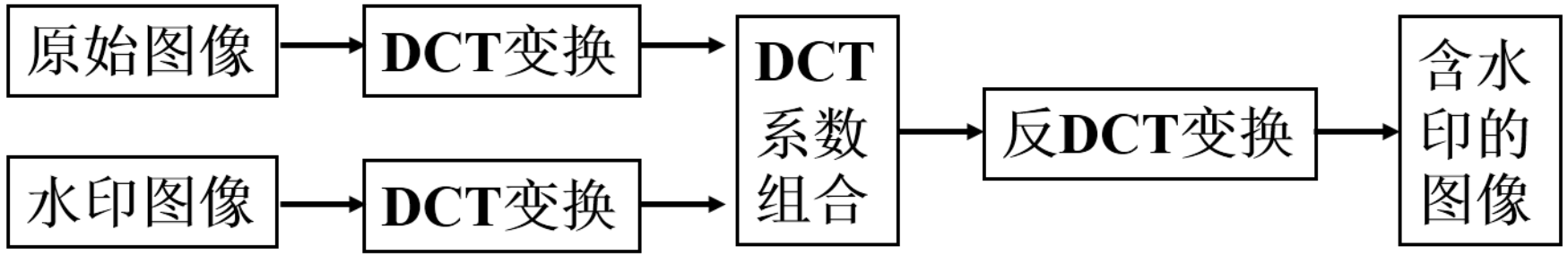
(a) DCT基压缩编码步骤



(b) DCT基解压缩步骤

拓展--离散余弦变换DCT

DCT在数字水印（digital watermarking）技术中的应用:
数字水印技术是将特定的信息嵌入到数字信息的内容中,要求嵌入的信息不能被轻易的去除,且在一定的条件下可以被提取出来,以确认作者的版权。

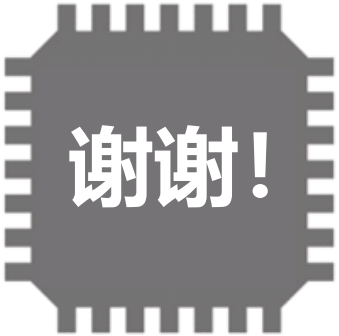


拓展--感知哈希算法

均值哈希算法过于严格，不够精确，更适合搜索缩略图，为了获得更精确的结果可以选择感知哈希算法，它采用的是DCT（离散余弦变换）来降低频率的方法。

步骤：

1. 缩小图片：32 * 32是一个较好的大小，这样方便DCT计算
2. 转化为灰度图：把缩放后的图片转化为灰度图。
3. 计算DCT:DCT把图片分离成分率的集合
4. 缩小DCT：DCT计算后的矩阵是32 * 32，保留左上角的8 * 8，这些代表图片的最低频率。
5. 计算平均值：计算缩小DCT后的所有像素点的平均值。
6. 进一步减小DCT：大于平均值记录为1，反之记录为0.
7. 得到信息指纹：组合64个信息位，顺序随意保持一致性。
8. 最后比对两张图片的指纹，获得汉明距离即可。


谢谢！