第十章：图像识别

中国科学技术大学
电子工程与信息科学系

图象识别

- 简单形状检测
  - 霍夫变换 (Hough Transform)
  - 倒角距离变换 (Chamfer Distance Transform)
- 图像分类
- 图像检索

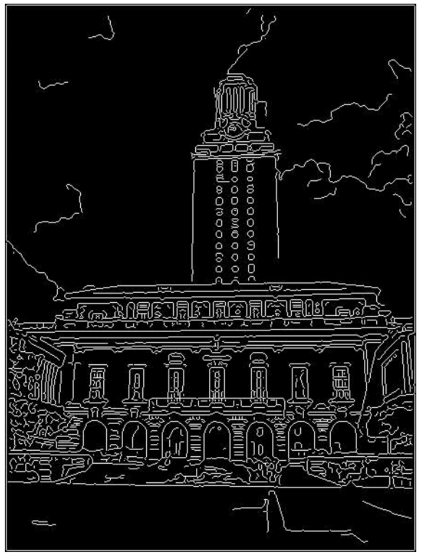
霍夫变换 (Hough Transform)

□ 应用场景：直线拟合

如何从**边缘检测**的结果得到直线拟合的结果？

□ 直线拟合的难点

- 边缘检测点杂乱且多余
- 不同的检测点属于不同的直线
- 部分线段可能漏检
- 边缘检测点上存在噪声



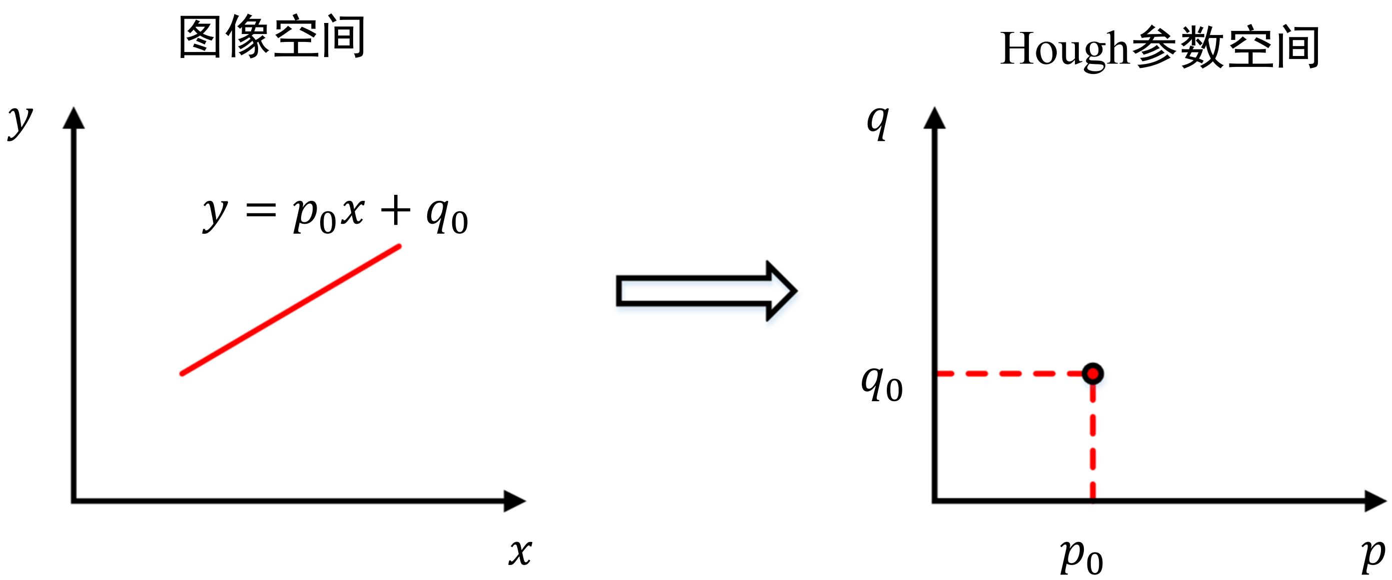
霍夫变换 (Hough Transform)

- 基本思想：基于投票的机制
  - 视待检测形状为一个模型(model)
  - 让每个样本（像素）对所有与其兼容的模型进行投票
  - 假设：噪声样本不会偏好任何单个模型
  - 即使有部分样别缺失，当有足够的样本保留时，仍可将目标模型（形状）检测出来

霍夫变换原理：以直线检测为例

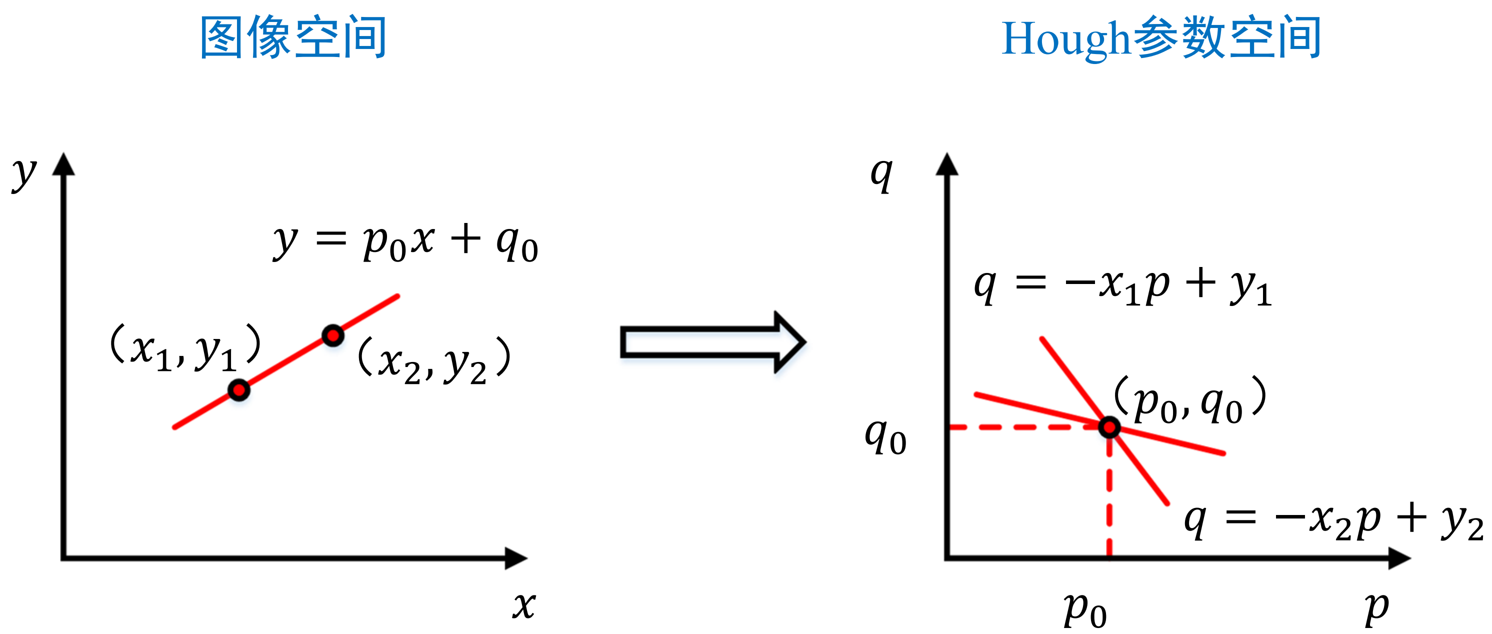
- Hough Transform
  - 图像空间与参数空间之间的一种变换

- Hough参数空间
  - 在图像空间中的一条直线，对应于Hough参数空间中的一个点



霍夫变换：直线检测

- 参数空间的一条直线对应图像空间的一个点
- 图像空间中同一条直线上的任意两个点，在参数空间中对应于两条相交的直线
  - 即在图像空间共线的n个点，对应于参数空间n条共点(p_0, q_0)的直线



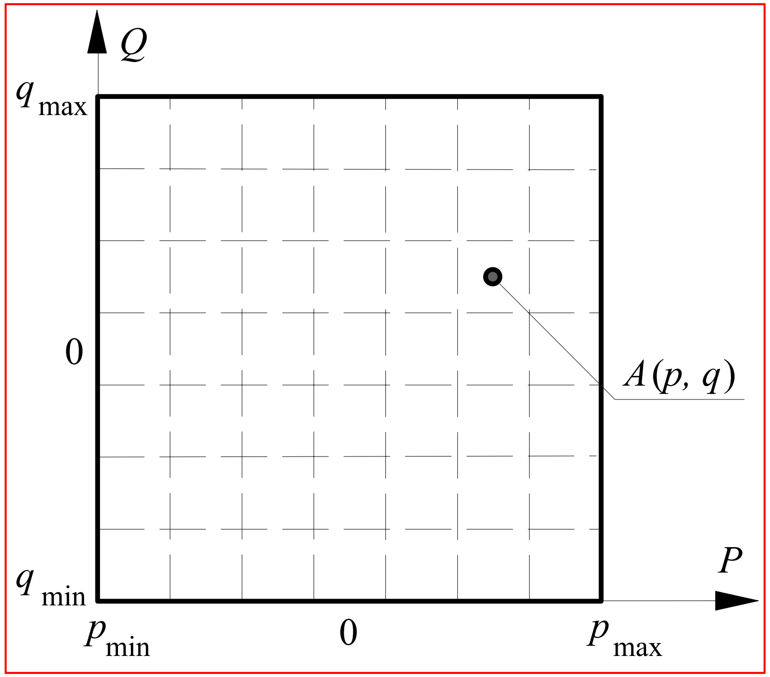
霍夫变换：直线检测

- 具体方法
  - 将参数空间离散化为一个2-D的累加数组$A(p, q)$

$$p \in [p_{\min}, p_{\max}]$$

$$q \in [q_{\min}, q_{\max}]$$

  - $A(p, q) = A(p, q) + 1$
    $A(p, q)$：共线点数
    (p, q)：直线方程参数



- 潜在的问题
  - $p_{\min}$和$q_{\min}$可能为无穷小，$p_{\max}$ 和$q_{\max}$可能为无穷大，难以对其进行离散化

霍夫变换：直线检测的改进形式

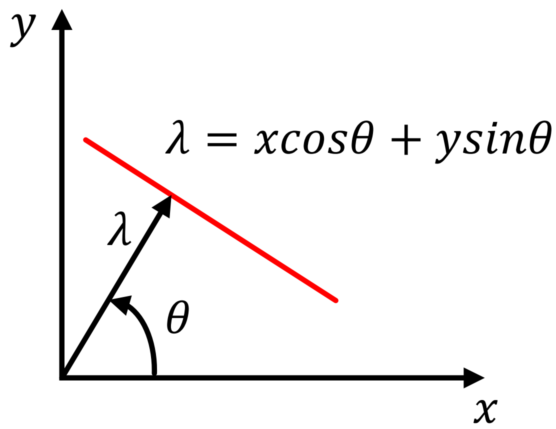
- 直线的极坐标方程

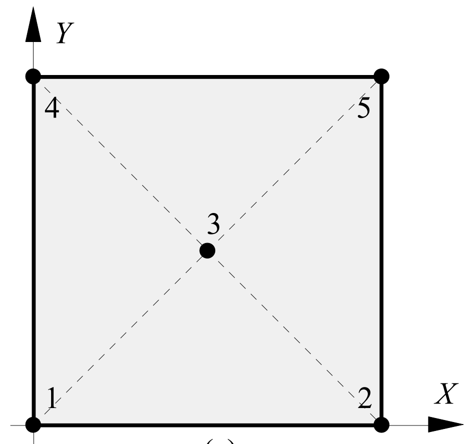
$$\lambda = x\cos\theta + y\sin\theta$$

  - 参数λ和θ唯一确定一条直线

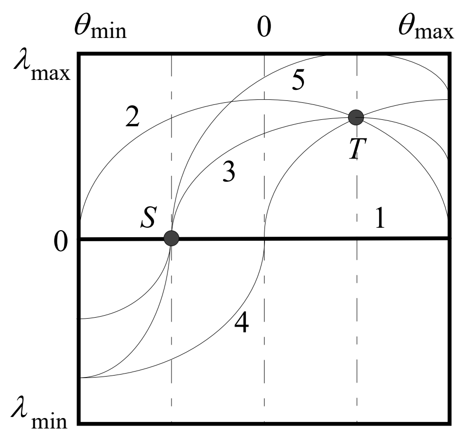
- X-Y平面的一个点对应参数空间的一条正弦曲线
  - $\lambda = x_0\cos\theta + y_0\sin\theta \boxed{\leftrightarrow} \lambda = A\sin(\theta + \alpha)$

$$\text{其中}\alpha = \tan^{-1}\left(\frac{x_0}{y_0}\right), A = \sqrt{x_0^2 + y_0^2}$$





(a)



(b)

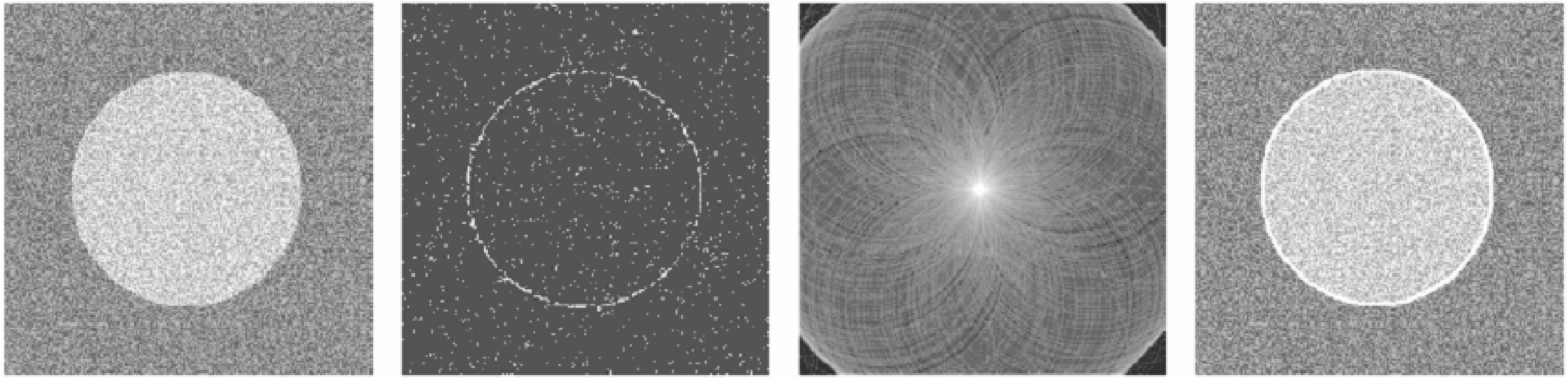
霍夫变换：其他形状的检测

- 对于满足显式解析式 $f(x, c) = 0$形式的各类曲线， 霍夫变换可以将其检测出来，并把曲线上的点完整地连接起来

- 示例：以圆周检测为例
  - 圆周方程：$(\mathrm{x} - a)^2 + (y - b)^2 = r^2$
  - 三个参数 a、b、r ，所以需要在参数空间中建立3-D累加数组，其中的元素可以记为$A(a, b, r)$

霍夫变换：圆周检测

- 示例：检测半径为确定值的圆



(a) (b) (c) (d)

图(a)为256x256，灰度256级，叠加随机噪声；

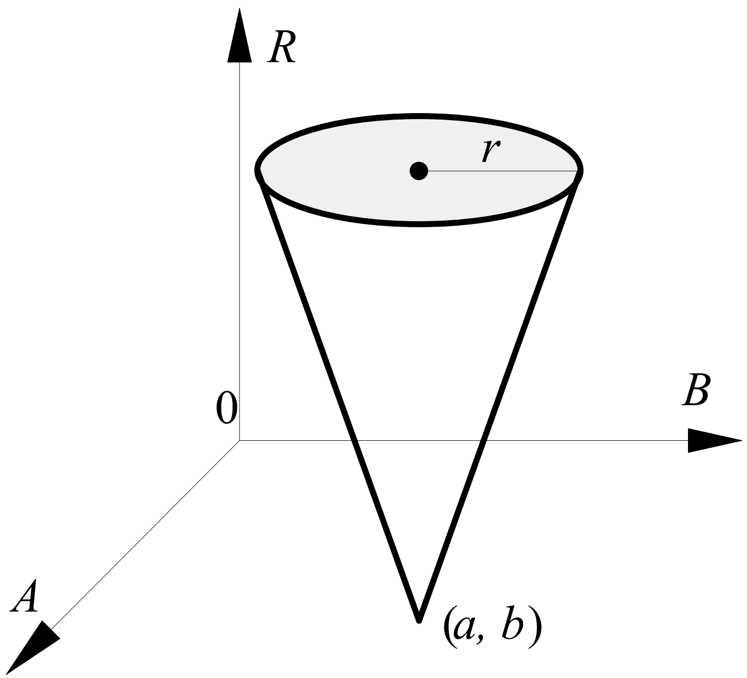
图(b)为求梯度(Sobel算子)取阈值后的结果（边缘检测）；

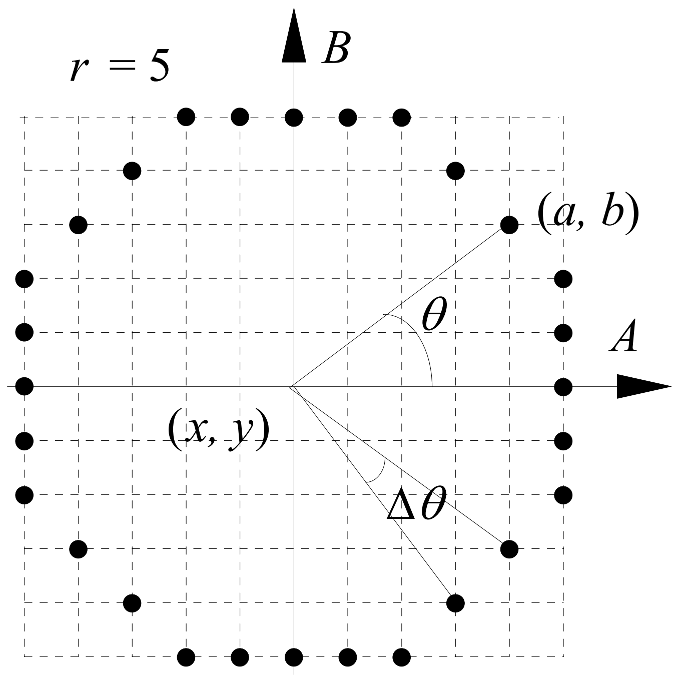
图(c)哈夫变换累计器图；

图(d)为检测出的圆周附加在原图上的效果

霍夫变换：圆周检测

- 利用梯度降维
  - 使累加数组的维度减少一维
  - 圆周——圆周对偶性





$$(a-x)^2+(b-y)^2=r^2$$

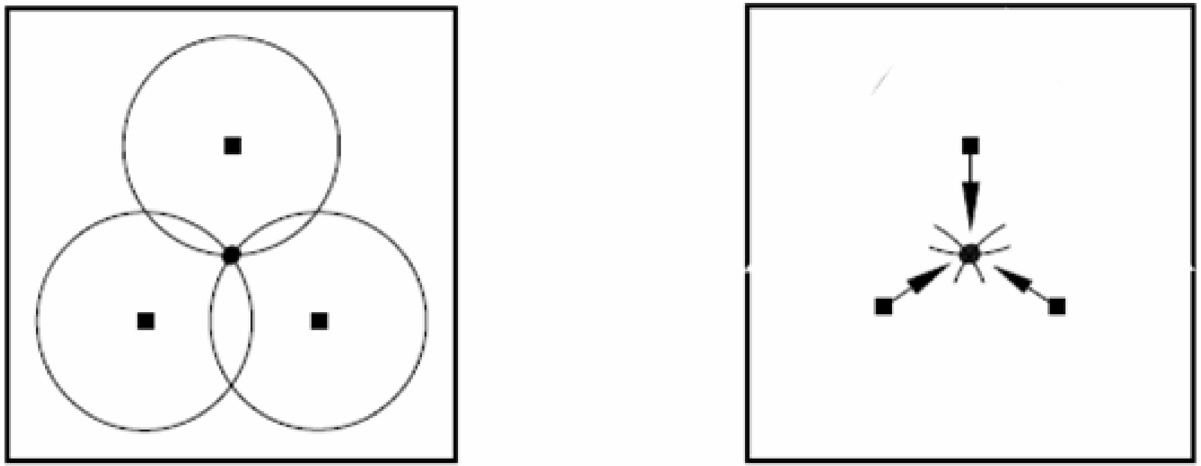
霍夫变换：圆周检测

- 利用梯度降维
  - 圆周圆心在圆周边缘点的梯度方向上

$$a = x - r\sin\theta$$

$$b = y + r\cos\theta$$

1个3-D累加器数组 → 2个2-D累加器数组



(a) (b)

利用梯度与否 2 种情况下的累加数组示意

广义霍夫变换

- 广义Hough变换将一般的**模板匹配**与**Hough变换**相结合
- 先对模板与图象上的物点作**坐标变换**，然后**求相关**
- 并用类似Hough变换检测物体的表决方法来确定匹配点

广义霍夫变换原理

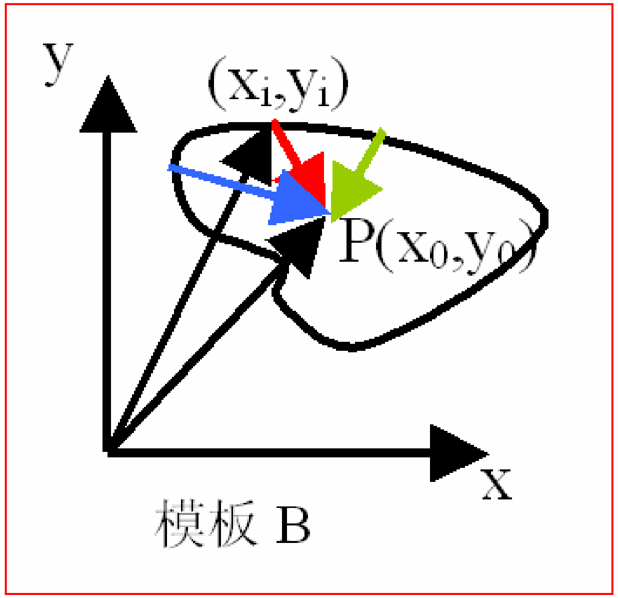
- 给定模板的一组点集
  - $\mathbf{B} = \{(x_i, y_i), i = 1, 2, \cdots, m\}$
- $P(x_0, y_0)$ 为一参考点，常把P取为**B**的中心点。
  - 从而将 **B** 表达为$\mathbf{B} = \{(dx_i, dy_i), i = 1, 2, \cdots, m\}$

$$\boxed{H(B,P)}$$

$$\boxed{\{(dx_i, dy_i), i = 1, 2, \cdots, n\}}$$

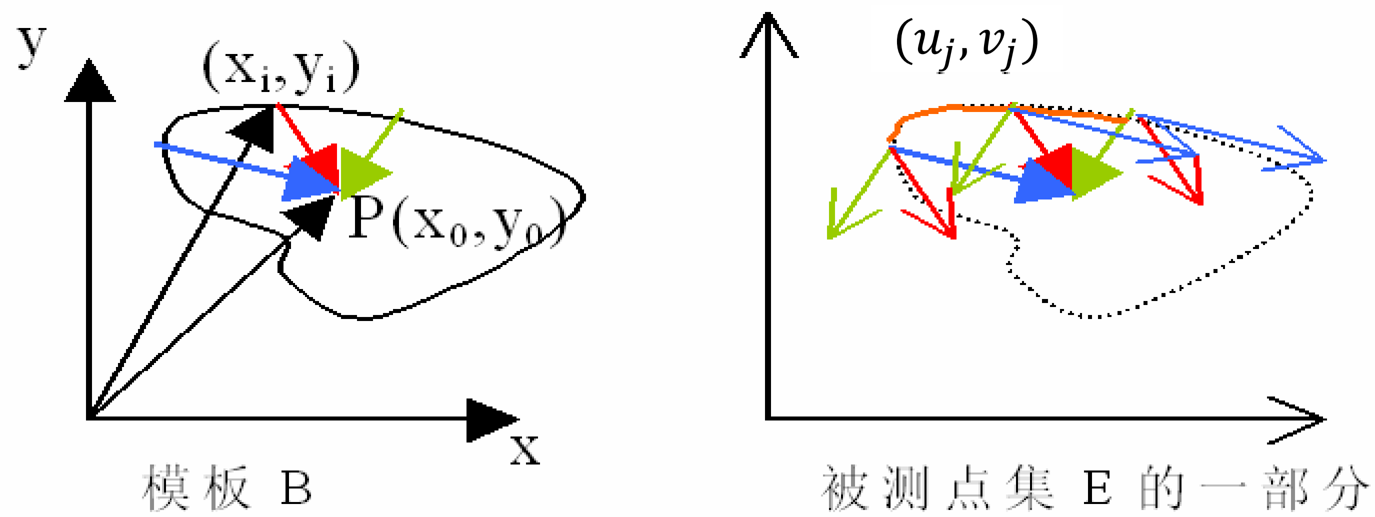
$$\boxed{dx_i = x_0 - x_i = -(x_i - x_0)}$$

$$\boxed{dy_i = y_0 - y_i = -(y_i - y_0)}$$



广义霍夫变换原理

- 检测时，将待测图象，记为点集E：
  - $E=\{(u_j,v_j),j=1,2,\cdots,n\}$
- 将待检测点集与变换后的模板B做相关运算
  - $\forall i,j$, 计算$\boldsymbol{m}(i,j)=(dx_i+u_j,dy_i+v_j)$,
  - 如果有很多点$\boldsymbol{m}(i,j)$都对应同一个坐标点，则该点为与B相匹配的形状的中心位置



广义霍夫变换原理

- 在所需检测的曲线或目标轮廓没有或不易用解析式表达时，可以利用表格来建立**曲线或轮廓点**与**参考点**间的关系，从而可继续利用哈夫变换进行检测

建立参考点与轮廓点的联系：

$$p = x + r(\theta) \cdot \cos(\phi(\theta))$$
$$q = y + r(\theta) \cdot \sin(\phi(\theta))$$

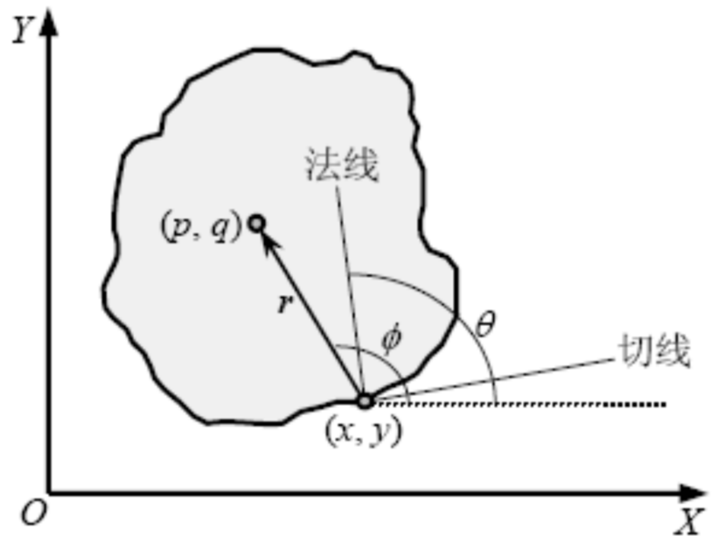


图 6.1.8 建立参考点和轮廓点的对应关系

广义霍夫变换原理

- 已知轮廓形状、朝向和尺度，而只需检测位置信息
- 根据θ, r和ϕ 的函数关系作出参考表

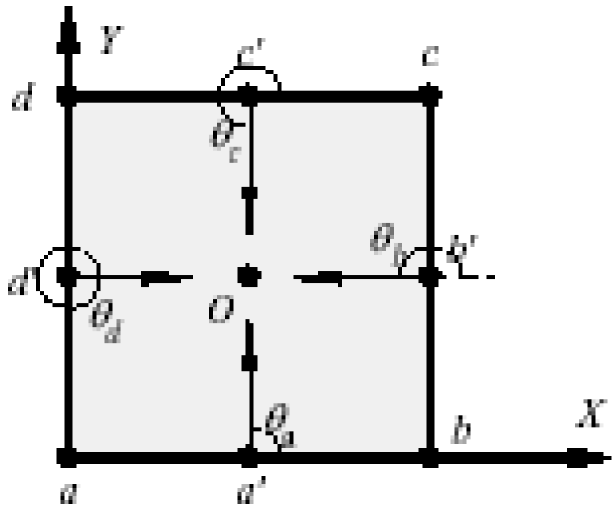
梯度角θ	矢径$r(\theta)$	矢角$\phi(\theta)$
θ_1	$r_1^1,\ r_1^2,\ \cdots,\ r_1^{N_1}$	$\phi_1^1,\ \phi_1^2,\ \cdots,\ \phi_1^{N_1}$
θ_2	$r_2^1,\ r_2^2,\ \cdots,\ r_2^{N_2}$	$\phi_2^1,\ \phi_2^2,\ \cdots,\ \phi_2^{N_2}$
$\cdots$	$\cdots$	$\cdots$
θ_M	$r_M^1,\ r_M^2,\ \cdots,\ r_M^{N_M}$	$\phi_M^1,\ \phi_M^2,\ \cdots,\ \phi_M^{N_M}$

- 给定一个测试点(x', y') 及其梯度角θ'，即可确定一组可能的参考点位置
  - 根据梯度角θ'，找到表中对应梯度角所在行的矢径r和矢角ϕ序列
  - 基于坐标(x', y')和序列中每一组(r, ϕ)值，反推出形状参考点坐标

$$p' = x' + r(\theta) \cdot \cos\big(\phi(\theta)\big)$$
$$q' = y' + r(\theta) \cdot \sin(\phi(\theta))$$

广义霍夫变换原理

轮廓点	a	a'	b	b'	C	c'	d	d'
矢径 $r(\theta)$	$\sqrt{2}/2$	$1/2$	$\sqrt{2}/2$	$1/2$	$\sqrt{2}/2$	$1/2$	$\sqrt{2}/2$	$1/2$
矢角 $\phi(\theta)$	$1\pi/4$	$2\pi/4$	$3\pi/4$	$4\pi/4$	$5\pi/4$	$6\pi/4$	$7\pi/4$	$8\pi/4$



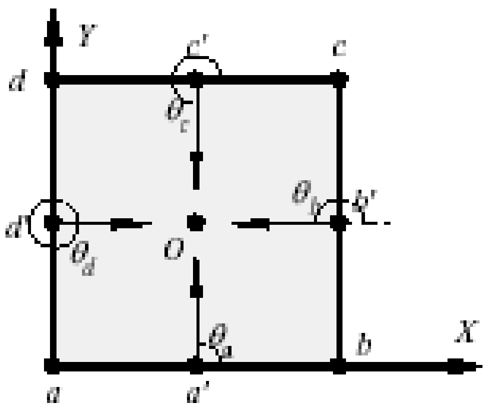
梯度角θ	矢径 $r(\theta)$		矢角 $\phi(\theta)$	
$\theta_a = \pi/2$	$\sqrt{2}/2$	$1/2$	$\pi/4$	$2\pi/4$
$\theta_b = 2\pi/2$	$\sqrt{2}/2$	$1/2$	$3\pi/4$	$4\pi/4$
$\theta_c = 3\pi/2$	$\sqrt{2}/2$	$1/2$	$5\pi/4$	$6\pi/4$
$\theta_d = 4\pi/2$	$\sqrt{2}/2$	$1/2$	$7\pi/4$	$8\pi/4$

广义霍夫变换原理

- 利用正方形上的8个轮廓点判断可能参考点位置(p', q')
- 对每个θ有 2个 r 及2个 ϕ 与之对应

梯度角θ	矢径 $r(\theta)$		矢角 $\phi(\theta)$	
$\theta_a = \pi/2$	$\sqrt{2}/2$	$1/2$	$\pi/4$	$2\pi/4$
$\theta_b = 2\pi/2$	$\sqrt{2}/2$	$1/2$	$3\pi/4$	$4\pi/4$
$\theta_c = 3\pi/2$	$\sqrt{2}/2$	$1/2$	$5\pi/4$	$6\pi/4$
$\theta_d = 4\pi/2$	$\sqrt{2}/2$	$1/2$	$7\pi/4$	$8\pi/4$

$$p' = x' + r(\theta) \cdot \cos(\phi(\theta))$$
$$q' = y' + r(\theta) \cdot \sin(\phi(\theta))$$



梯度角	轮廓点	可能参考点		轮廓点	可能参考点	
θ_a	a	O	d'	a'	b'	O
θ_b	b	O	a'	b'	c'	O
θ_c	c	O	b'	c'	d'	O
θ_d	d	O	c'	d'	a'	O

点O出现频率最高

广义霍夫变换的性能

- 运算量较小
- 抗干扰性也较强
- 可以求出曲线的某些参数
- 可适用于不规则曲线
- 仍不具有旋转不变性和缩放不变性

完整广义霍夫变换

- 轮廓的平移 ＋ 轮廓放缩、旋转
- 累加数组：
- $A(p_{\min}: p_{\max}, q_{\min}: q_{\max}, \beta_{\min}: \beta_{\max}, S_{\min}: S_{\max})$

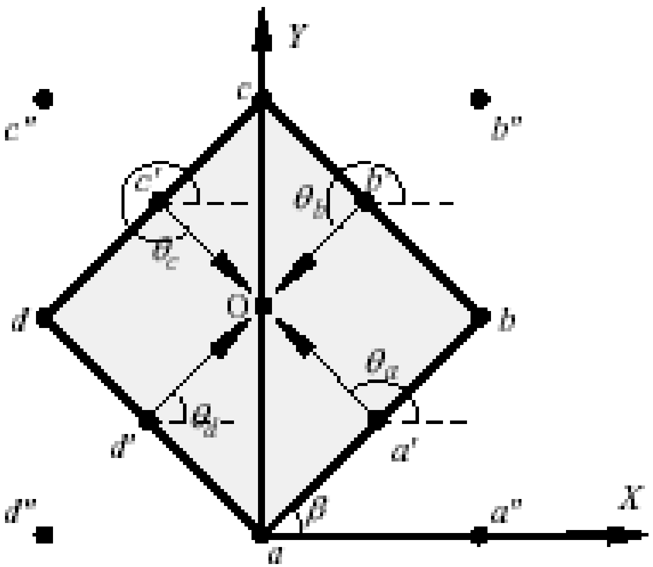
$$\boxed{p = x + S \times r(\theta) \times \cos[\phi(\theta) + \beta]}$$

$$\boxed{q = y + S \times r(\theta) \times \sin[\phi(\theta) + \beta]}$$

- 累加数组的累加：$A(p, q, \beta, S) = A(p, q, \beta, S) + 1$

完整广义霍夫变换

计算示例



假设已知$S = 1$，$\beta = \pi/4$

原梯度角θ	新梯度角θ'	矢径 $r(\theta)$		新矢角$\phi(\theta)$	
$\theta_a = \pi/2$	$\theta'_a = 3\pi/4$	$\sqrt{2}/2$	$1/2$	$2\pi/4$	$3\pi/4$
$\theta_b = 2\pi/2$	$\theta'_b = 5\pi/4$	$\sqrt{2}/2$	$1/2$	$4\pi/4$	$5\pi/4$
$\theta_c = 3\pi/2$	$\theta'_c = 7\pi/4$	$\sqrt{2}/2$	$1/2$	$6\pi/4$	$7\pi/4$
$\theta_d = 4\pi/2$	$\theta'_d = \pi/4$	$\sqrt{2}/2$	$1/2$	$8\pi/4$	$1\pi/4$

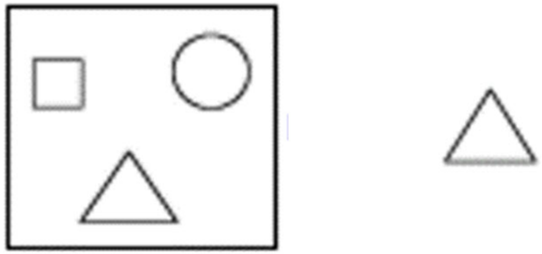
梯度角	轮廓点	可能参考点		轮廓点	可能参考点	
θ'_a	a	O	d'	a'	b'	O
θ'_b	b	O	a'	b'	c'	O
θ'_c	c	O	b'	c'	d'	O
θ'_d	d	O	c'	d'	a'	O

目标检测

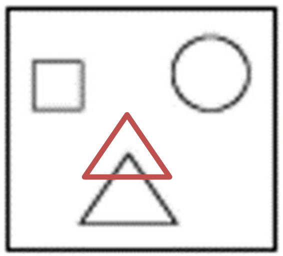
- 简单形状检测
  - 霍夫变换 (Hough Transform)
  - 倒角距离变换 (Chamfer Distance Transform)
- 图像分类
- 图像检索

倒角距离变换 (Chamfer Distance Transform)

- 从一个问题出发
  - 如何从下面左图二值边缘图像中检测匹配右图所示的三角形形状？



- 基本思路：
  - 把右图作为模板，将其中心置于左图所有可能的像素位置
  - 对于每个放置位置$\boldsymbol{x}$，计算模板形状与测试图像边缘的匹配距离



$$D(\boldsymbol{x}) = \frac{1}{|T|}\sum_{\boldsymbol{t}\in T} d_I(\boldsymbol{t}+\boldsymbol{x})$$

•T是模板形状（像素点的集合），$\boldsymbol{t}$为其中一个点的坐标
•I 是待匹配的边缘图像（像素点的集合）
•$\mathrm{d}_I(\boldsymbol{y})$是坐标点$\boldsymbol{y}$到 I 中所有边缘点的最近点的距离

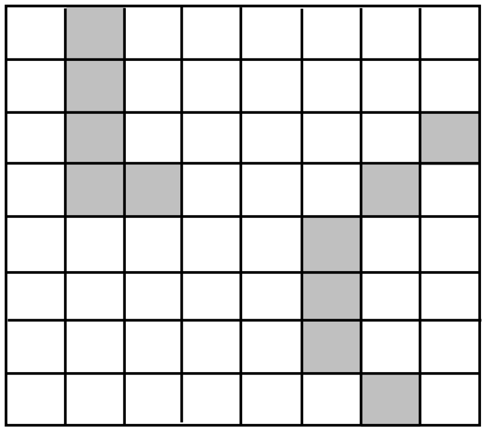
  - 将最小匹配距离 所对应的位置$\boldsymbol{x}$，视为匹配位置

倒角距离变换 (Chamfer Distance Transform)

- 计算复杂度分析
  - 假设测试图像中边缘点数量为M，所有像素点数量为P，模板中边缘点数量为N，上述距离计算的复杂度为$O(MNP)$
  - 上面距离测度存在大量计算冗余
- 如何降低冗余？
  - 事先计算测试图像中边缘点的距离变换
  - 图 I 中的边缘点集合$\boldsymbol{E}$，像素点$\boldsymbol{p}$的距离变换结果为：

$DF_I(\boldsymbol{p}) = \min_{\boldsymbol{x}\in \boldsymbol{E}} dist(\boldsymbol{p}, \boldsymbol{x})$，其中$dist(\cdot,\cdot)$表示距离函数，如棋盘距离

边缘图



距离变换结果

1	0	1	2	3	4	3	2
1	0	1	2	3	3	2	1
1	0	1	2	3	2	1	0
1	0	0	1	2	1	0	1
2	1	1	2	1	0	1	2
3	2	2	2	1	0	1	2
4	3	3	2	1	0	1	2
5	4	4	3	2	1	0	1

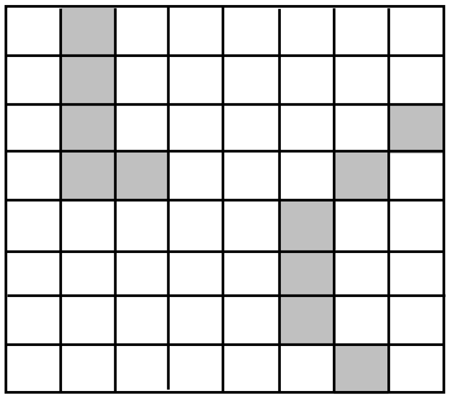
倒角距离变换 (Chamfer Distance Transform)

- 倒角距离变换：
  - 基于DF_I，对模板形状进行倒角匹配(Chamfer matching)

$$D_{chamfer}(\boldsymbol{x}) = \frac{1}{|T|}\sum_{\boldsymbol{t}\in T} d_I(\boldsymbol{t}+\boldsymbol{x}) = \frac{1}{|T|}\sum_{\boldsymbol{t}\in T} DF_I(\boldsymbol{t}+\boldsymbol{x})$$

    - T是模板形状（像素点的集合）
    - I 是待匹配的边缘图像（像素点的集合）
    - $\mathrm{d}_I(t)$是模板中的点t到 I 中边缘点的最小的距离
  - 计算复杂度： $O(MNP) \rightarrow O(NP)$

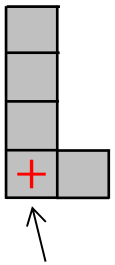
边缘图



距离变换结果

1	0	1	2	3	4	3	2
1	0	1	2	3	3	2	1
1	0	1	2	3	2	1	0
1	0	0	1	2	1	0	1
2	1	1	2	1	0	1	2
3	2	2	2	1	0	1	2
4	3	3	2	1	0	1	2
5	4	4	3	2	1	0	1

模板



倒角变换距离

1	1	3					
2	1	4					
3	1	5					
4	0	4					
6	2	5					
9	5	6					
13	9	8					
18	14	13					

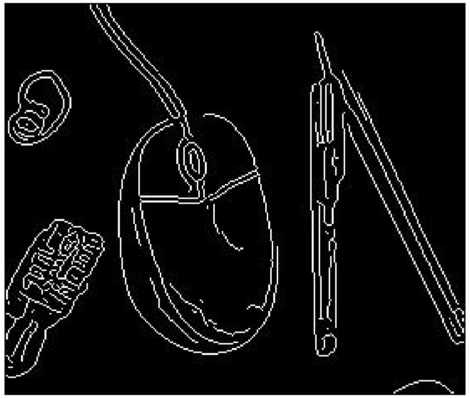
（自行计算后面五列结果）

倒角距离变换 (Chamfer Distance Transform)

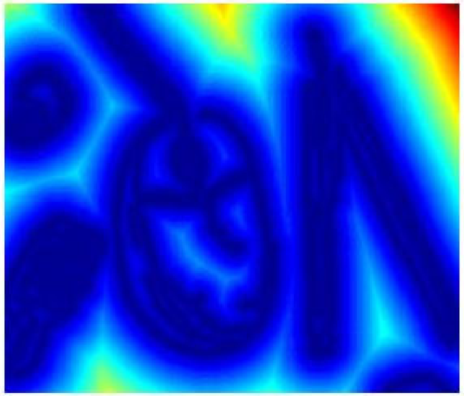
- 距离变换实例
  - 在距离变换的结果中，每个位置的值表示这个位置到最近的边缘点（或者其他二值化的图片结构）的距离

$$DF_I(\boldsymbol{p}) = \min_{\boldsymbol{x}\in \boldsymbol{E}} dist(\boldsymbol{p},\boldsymbol{x})$$

原图



边缘检测结果



距离变换结果

倒角距离变换：如何做距离变换？

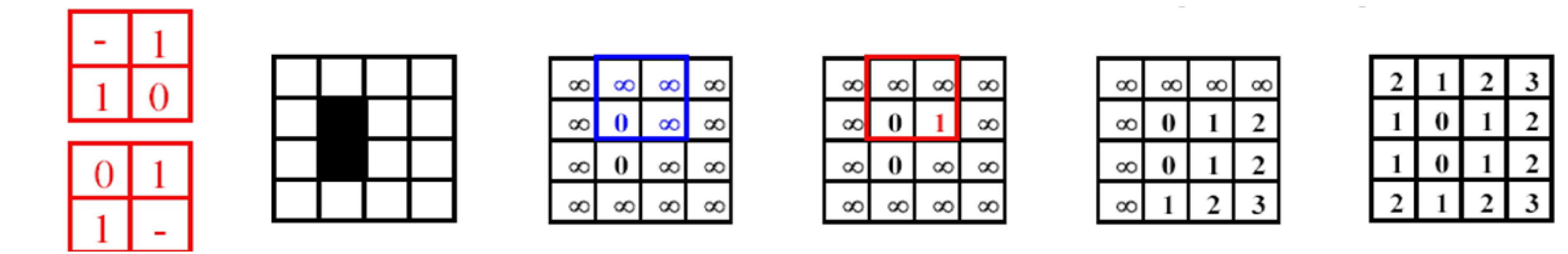
- 1-D距离变换
  - 1-D L_1范数的距离变换是一个计算复杂度为O(n)的算法
  - 算法步骤
    1. 对图中任意位置的值进行初始化，当j在特征P中时初始化为0，否则初始化为inf。将第 j 个位置的值记为$D[j]$
    2. 前向过程: for j from 1 up to n-1，更新$D[j]$
    $$D[j] = \min(D[j], D[j-1]+1)$$
    3. 后向过程: for j from n-2 down to 0，更新$D[j]$
    $$D[j] = \min(D[j], D[j+1]+1)$$

倒角距离变换：如何做距离变换？

- 2-D距离变换：算法步骤与1-D情况类似
  - 初始化距离矩阵
  - 前向过程：从上方和左方找离特征点最近的距离

    $D[i,j] = \min(D[i,j], D[i,j-1]+1, D[i-1,j]+1)$
  - 后向过程：从下方和右方找离特征点最近的距离

    $D[i,j] = \min(D[i,j], D[i,j+1]+1, D[i+1,j]+1)$



倒角距离变换 (Chamfer Distance Transform)

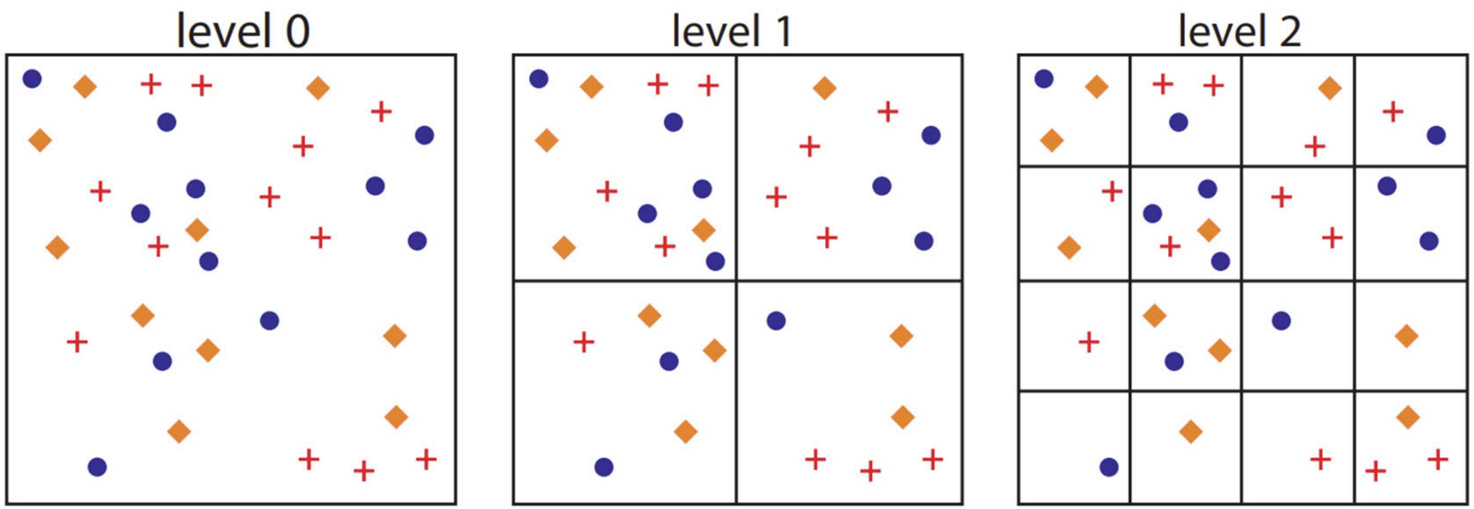
- 优点
  - 对混乱背景干扰较为鲁棒
  - 计算效率高
- 缺点
  - 对缩放和旋转变换敏感
  - 对形状的小的形变敏感
  - 需要大量的模板形状，应应对目标形状的形变
- 改进方法
  - 多尺度匹配
  - 层级模型组织（hierarchical model organization）

图像识别

- 简单形状检测
- 图像分类
  - 空间金字塔匹配
- 图像检索

面向图像分类的空间金字塔匹配

- 空间金字塔匹配 (Spatial Pyramid Matching)
  - 基于局部视觉特征（如SIFT）和词袋模板，一副图像中的各个局部视觉特征可表达为相应的视觉单词
  - 不同类别的图像，视觉单词在图像平面服从某种空间分布
  - 如何表达视觉单词间的空间上下文关系？
    - ✓ 空间金字塔：在第 l 层，将图像平面均匀划分为$2^l \times 2^l$ 份， $0 \le l \le L$

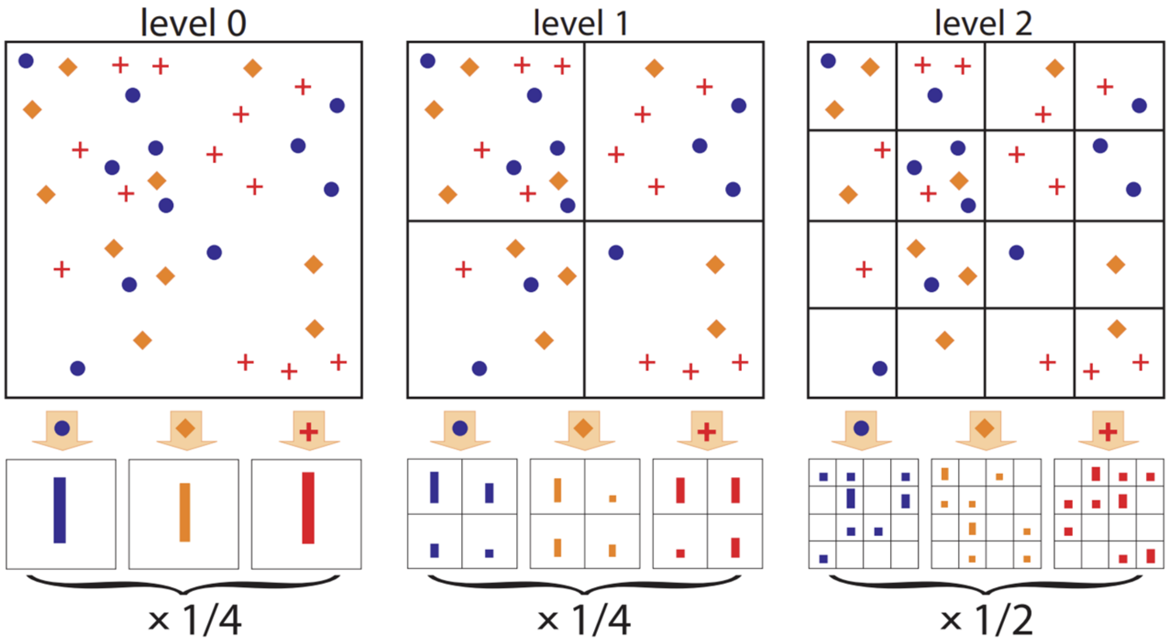


- Lazebnik S, Schmid C, Ponce J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. IEEE CVPR, 2006, 2: 2169-2178.

面向图像分类的空间金字塔匹配

- 视觉单词的空间金字塔表达
  - 空间金字塔：在第 l 层，将图像平面均匀划分为$2^l \times 2^l$ 个格子区域
  - 对每个划分的格子区域，将其表达为视觉单词直方图
  - 在第 l 层，两幅图像X 和 Y匹配的视觉单词数量（直方图交）：

$$\mathcal{I}^\ell = \mathcal{I}(H_X^\ell, H_Y^\ell) = \sum_{i=1}^{D} \min\left(H_X^\ell(i), H_Y^\ell(i)\right). \qquad (D\text{为格子数量})$$



面向图像分类的空间金字塔匹配

视觉单词的空间金字塔表达

- 对于两幅图像，第 l 层匹配的视觉单词 包含第 $l+1$ 层匹配的所有视觉单词
- 所以，相对于第 $l+1$ 层，第 l 层的新增匹配定义为$I^l - I^{l+1}$，对应的权值为$\frac{1}{2^{L-l}}$，反比于该层格子宽度
  - ✓ 格子宽度越大，匹配噪声越严重，越不可靠，故权值越小
- **空间金字塔匹配（SPM）**
  - ✓ 对于两幅图像 X 和 Y，其L 层的空间金字塔匹配核：

$$
\begin{aligned}
\kappa^L(X,Y) &= \mathcal{I}^L + \sum_{\ell=0}^{L-1} \frac{1}{2^{L-\ell}} \left(\mathcal{I}^\ell - \mathcal{I}^{\ell+1}\right) \\
&= \frac{1}{2^L}\mathcal{I}^0 + \sum_{\ell=1}^{L} \frac{1}{2^{L-\ell+1}} \mathcal{I}^\ell .
\end{aligned}
$$

图像识别

- □ 简单形状检测
- □ 图像分类
- □ 图像检索
  - 倒排索引
  - 空间验证
  - 二值哈希

图像检索

基于内容的图像检索

数据库规模：$>10^6$~10^9
查询响应：实时

图像检索的潜在应用场景

图像溯源检索	商品图像检索	监控目标检索	纹身图像检索	……
TinEye Reverse Image Search	拍立淘 pailitao.com			
版权保护	网络购物	安防监控	刑侦分析	

图像数据库索引：正向索引

- 正向索引
  - 基于词袋模型，每幅图像表达为一个维度为K的矢量
  - 遍历数据库图像，一一计算与查询图像的相似性得分

query

1
0.7
0
$\vdots$
0

	I_1	I_2		I_j		I_M
D_1	0	0.5	$\cdots$	0	$\cdots$	0.5
D_2	0.8	0.2	$\cdots$	0	$\cdots$	0
D_3	0	0	$\cdots$	1	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
D_K	0	0	$\cdots$	0.9	$\cdots$	0.1

$[s_1, s_2, \cdots, s_M]$

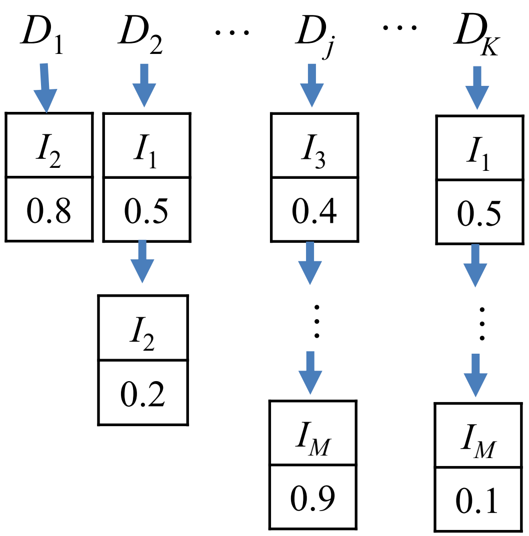
- 在大规模图像检索中: 遍历M张图像逐一计算相似性，耗时太久。
- 如何改进？
  - ✓ 去除向量距离计算时的计算冗余

图像数据库索引：倒排索引

- 先验条件
  - 图像视觉表征向量的**稀疏性**：非零元素比例低，比如< 1%
  - 只需存储向量中的非零元素 → 视觉单词在词典中的索引
- 倒排索引的优势：高效的存储和计算
  - 根据向量距离公式，**仅存储和比较向量中的非0元素**

$$D\left(I_q, I_m\right) = \sum_{i=1}^{K} |q_i - m_i|^p = 2 + \sum_{i|q_i \neq 0, m_i \neq 0} \left(|q_i - m_i|^p - {q_i}^p - {m_i}^p\right)$$

	D_1	D_2	$\cdots$	D_j	$\cdots$	D_K
I_1	0	0.5	$\cdots$	0	$\cdots$	0.5
I_2	0.8	0.2	$\cdots$	0	$\cdots$	0
I_3	0	0	$\cdots$	0.4	$\cdots$	0
$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\cdots$	$\vdots$
I_M	0	0	$\cdots$	0.9	$\cdots$	0.1



图像数据库索引：倒排索引

- 伪代码对比
  - 顺排表实现：

```
D[1 : M] = 0;
For i=1 : M
    For j = 1 : K
        D[i] += (q[j]-I[i][j]) (q[j]-I[i][j]);
    End
End
```

  - 倒排表实现：

```
D[1 : M] = 2;
For j = 1 : K
   len = pLen[j]; // len为第j个链表长度
   if q[j] > 0
         For m = 1 : len
           i = list[m].imgID;
           val = list[m].val;
          D[i] -=2 · q[j] · val;
         End
     End
End
```

图像数据库索引：倒排索引-示例

- 给定查询图像的K维L_1归一化的特征向量为$I_q = [0.7, 0.3, 0, \cdots, 0]$
  - I_q仅在第1、2维的值为非零
- 计算I_q和I_1、I_2的距离：

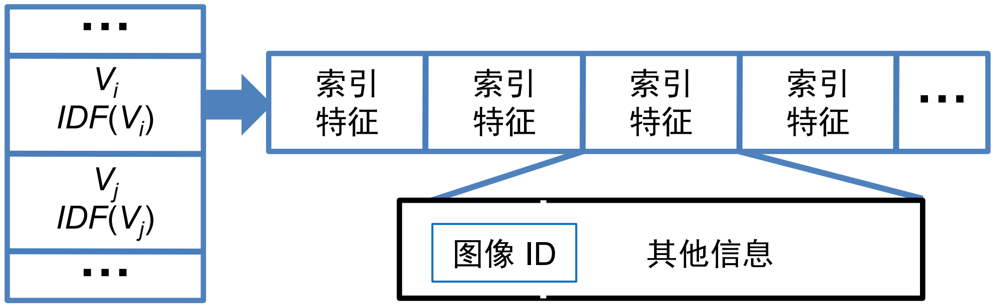
$$D(I_q, I_1) = 2 + \sum_{i|q_i \neq 0, m_i \neq 0} (|q_i - m_i| - q_i - m_i)$$
$$= 2 + (|q_2 - m_2| - q_2 - m_2) = 2 + 0.2 - 0.3 - 0.5 = 1.4$$

$$D(I_q, I_2) = 2 + \sum_{i|q_i \neq 0, m_i \neq 0} (|q_i - m_i| - q_i - m_i)$$
$$= 2 + (|q_1 - m_1| - q_1 - m_1) + (|q_2 - m_2| - q_2 - m_2) = 2 - 1.4 - 0.4 = 0.2$$

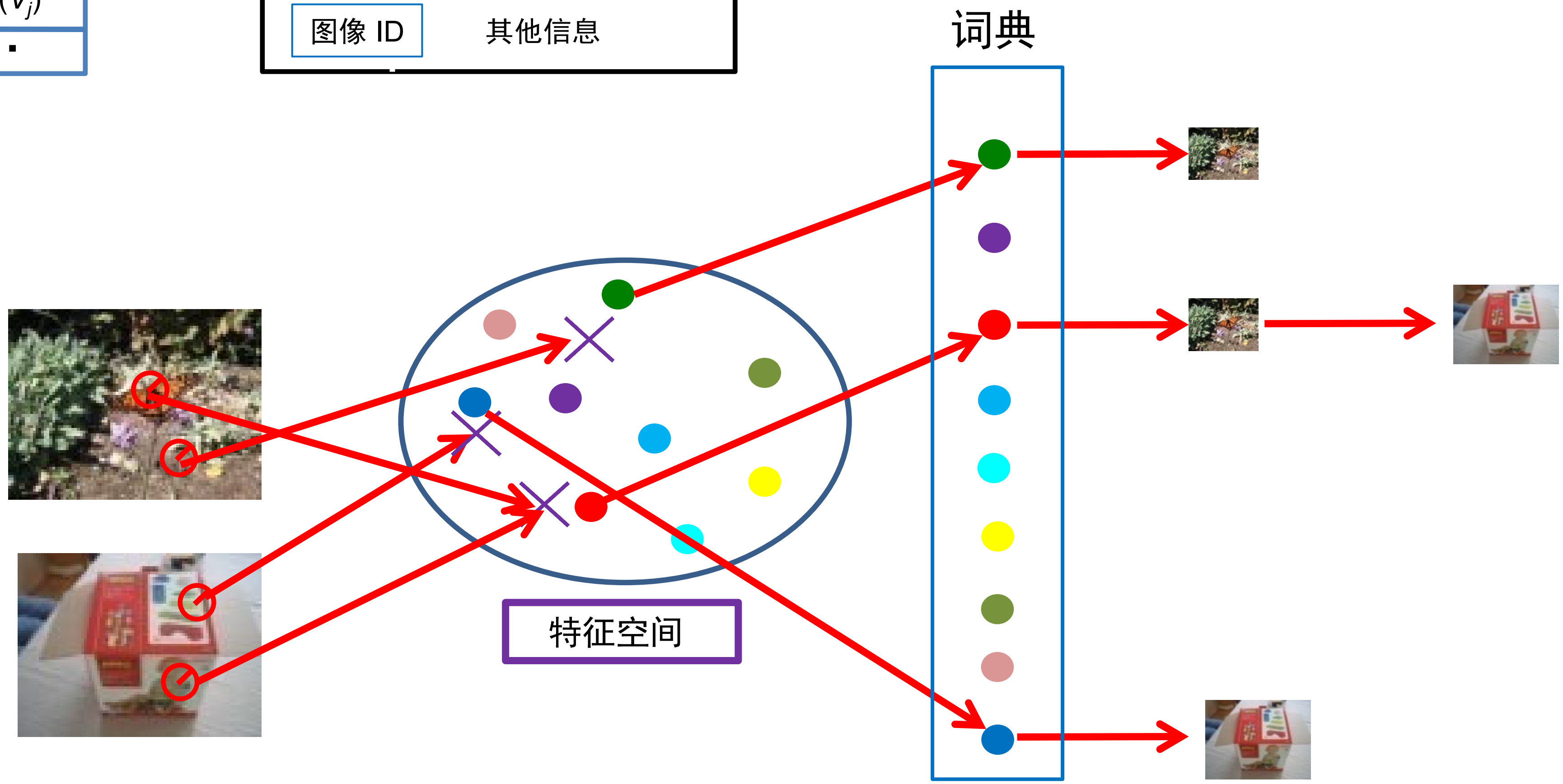
- 计算I_q和I_m $(m > 2)$ 的距离：

$$D(I_q, I_m) = 2 + \sum_{i|q_i \neq 0, m_i \neq 0} (|q_i - m_i| - q_i - m_i) = 2$$

图像数据库索引：倒排索引



$$
\begin{aligned}
D(I_q, I_m) &= \sum_{i=1}^{N} |q_i - m_i|^p \\
&= 2 + \sum_{i|q_i \neq 0, m_i \neq 0} (|q_i - m_i|^p - q_i^p - m_i^p)
\end{aligned}
$$

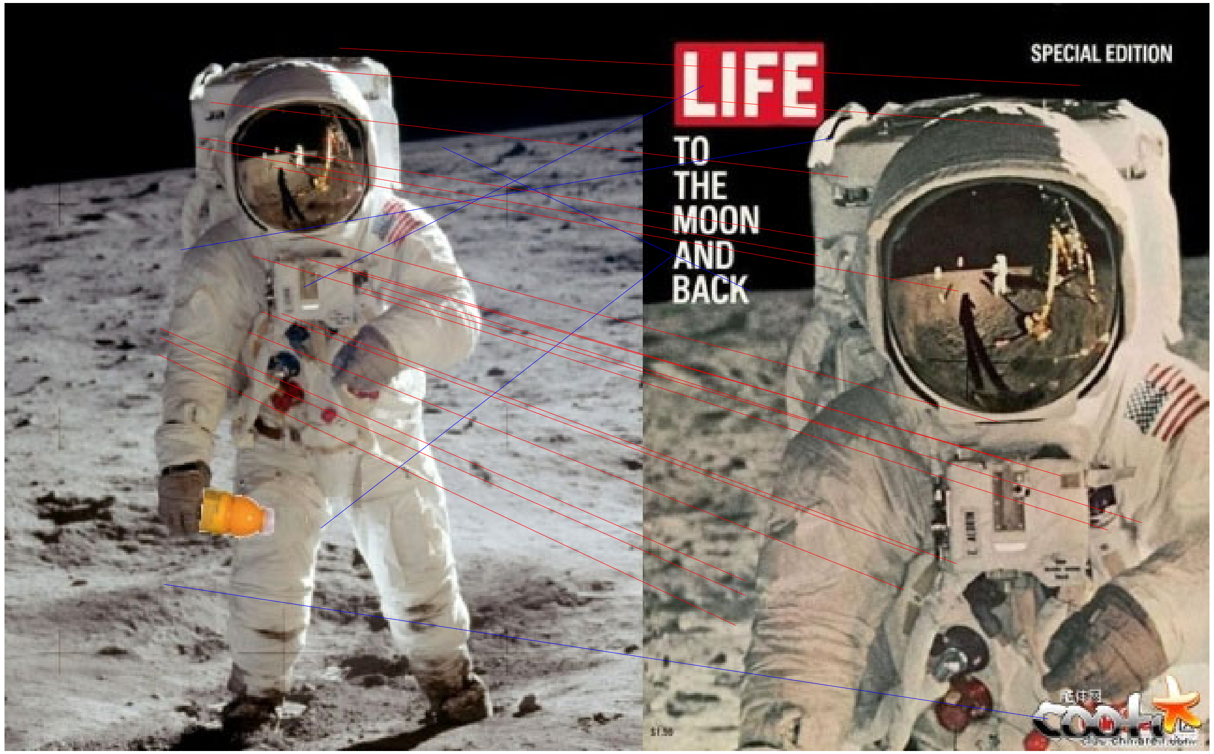


图像分类与检索

- 图像分类
- 图像检索
  - 倒排索引
  - 空间验证
  - 二值哈希

空间验证

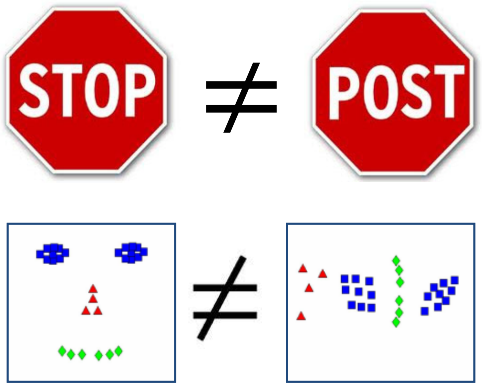
- 动机
  - 局部特征匹配时缺少对位置信息的校验
  - 通过检验几何一致性去掉错误的匹配点对



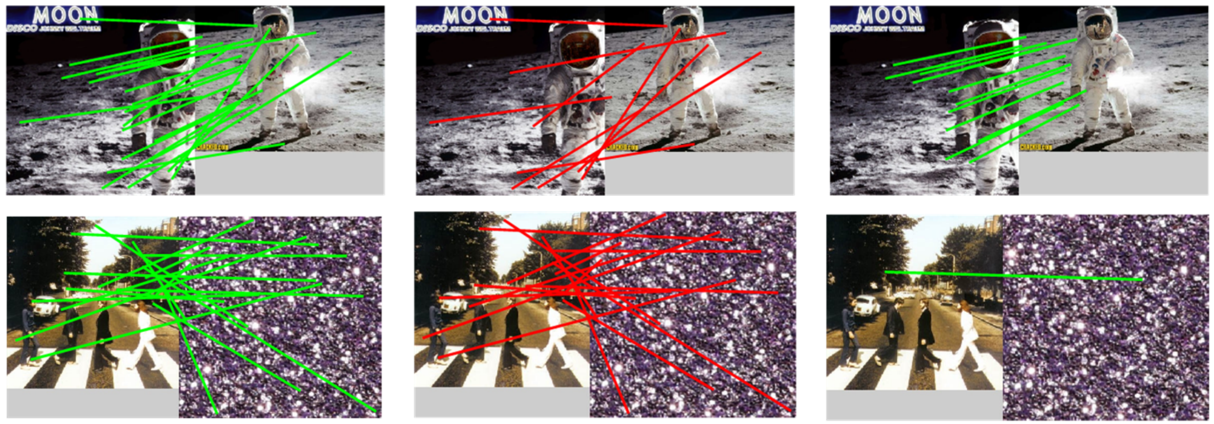
红线: 几何一致的匹配点对
蓝线: 几何不一致的匹配点对

视觉几何上下文表达

- 视觉单词以特定空间布局表达视觉语义
  - 利用几何上下文提升图像匹配质量
  - 有助于准确度量图像内容相关性



空间布局 vs. 视觉语义

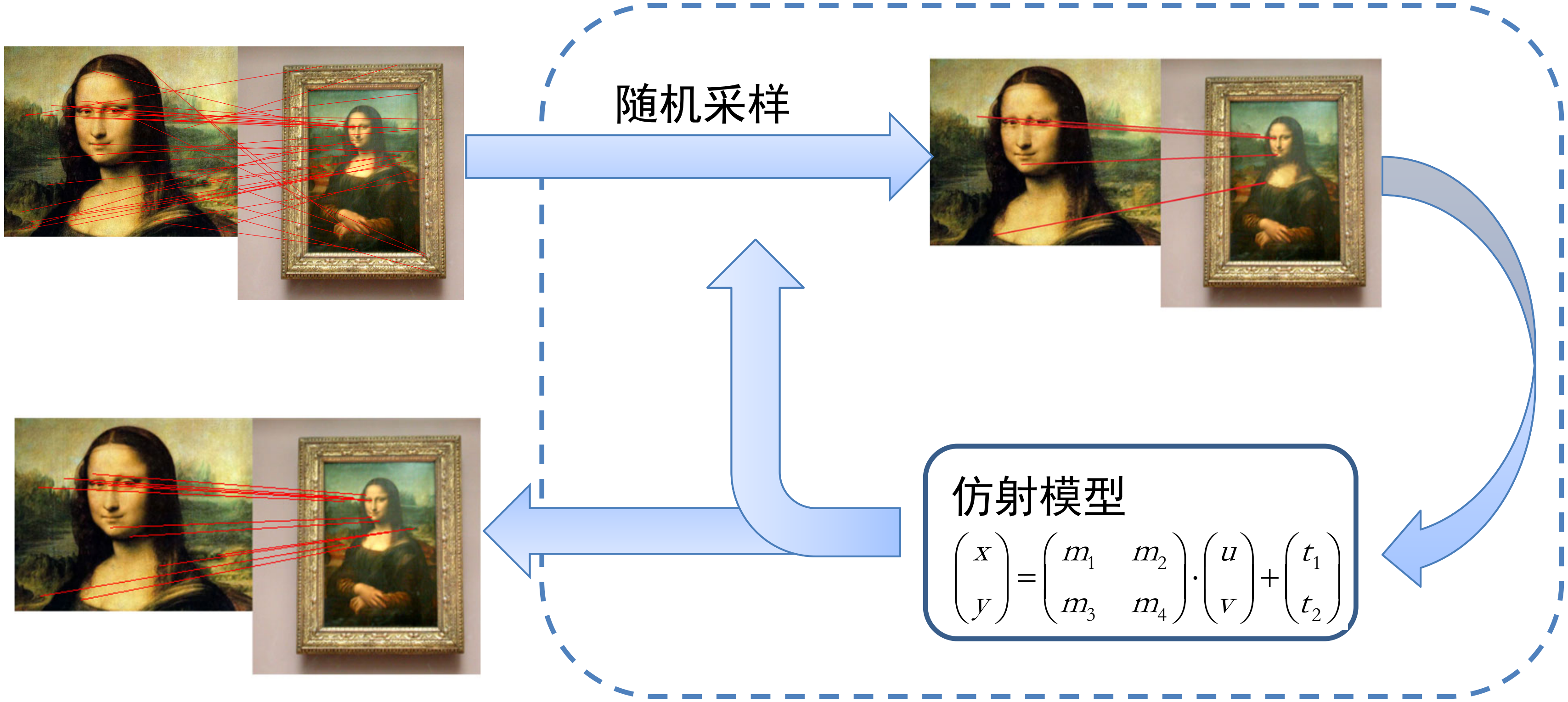


原始匹配　　噪声单词匹配　　几何一致的匹配

- 困难和挑战
  - 几何上下文结构化表达：便于图像匹配
  - 几何上下文快速匹配：保证实时检索

几何校验(1): RANSAC

- RANSAC算法示例
  - 通过匹配的特征点对估计图像的仿射变换



- Fischler, *et al.*, **RAN**dom **SA**mple **C**onsensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Comm. of the ACM*, 24:381-395, 1981.

几何校验(1): RANSAC

- **RANSAC:**
  - 通过正确匹配点对估计仿射模型来排除错误匹配点对
  - inliers：正确的匹配点对
  - outliers：错误的匹配点对
- **RAN**dom **SA**mple **C**onsensus (RANSAC)的先验条件
  - 原始数据由inliers和outliers组成
  - inliers的子集可以正确的估计图像间的仿射变换
- 通过RANSAC估计仿射变换

  1.迭代的随机选取匹配点对当作假设的*inliers*
  2.根据假设的inliers计算一个仿射模型
  3.其他数据点根据上述的仿射模型判断是否是*inliers*
  4.通过所有的inliers重新估计仿射模型
  5.通过所有的匹配点对与模型的拟合程度计算误差

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} m_1 & m_2 \\ m_3 & m_4 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$$

- 缺点：由于随机采样点对估计模型要重复多次导致计算量大，计算复杂度为$O(N^3)$

Fischler, *et al.*, **RAN**dom **SA**mple **C**onsensus: a paradigm for model fitting with applications to image analysis and automated cartography, *Comm. of the ACM*, 24:381-395, 1981

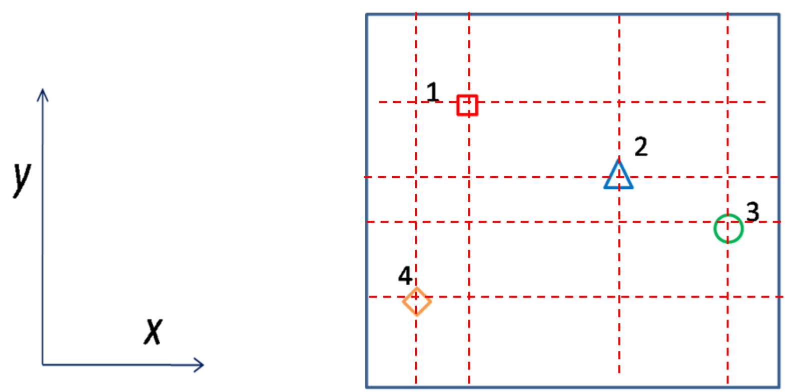
几何校验(2): 空间编码(Spatial Coding)

□ 核心思想: 建立空间编码矩阵

- ■ 匹配特征的相对空间位置关系

$$Xmap(i,j)=\begin{cases}0 & \text{if } x_j > x_i \quad \text{right to } x_i\\ 1 & \text{if } x_j \le x_i \quad \text{left to } x_i\end{cases}\qquad Ymap(i,j)=\begin{cases}0 & \text{if } y_j > y_i \quad \text{above } y_i\\ 1 & \text{if } y_j \le y_i \quad \text{below } y_i\end{cases}$$

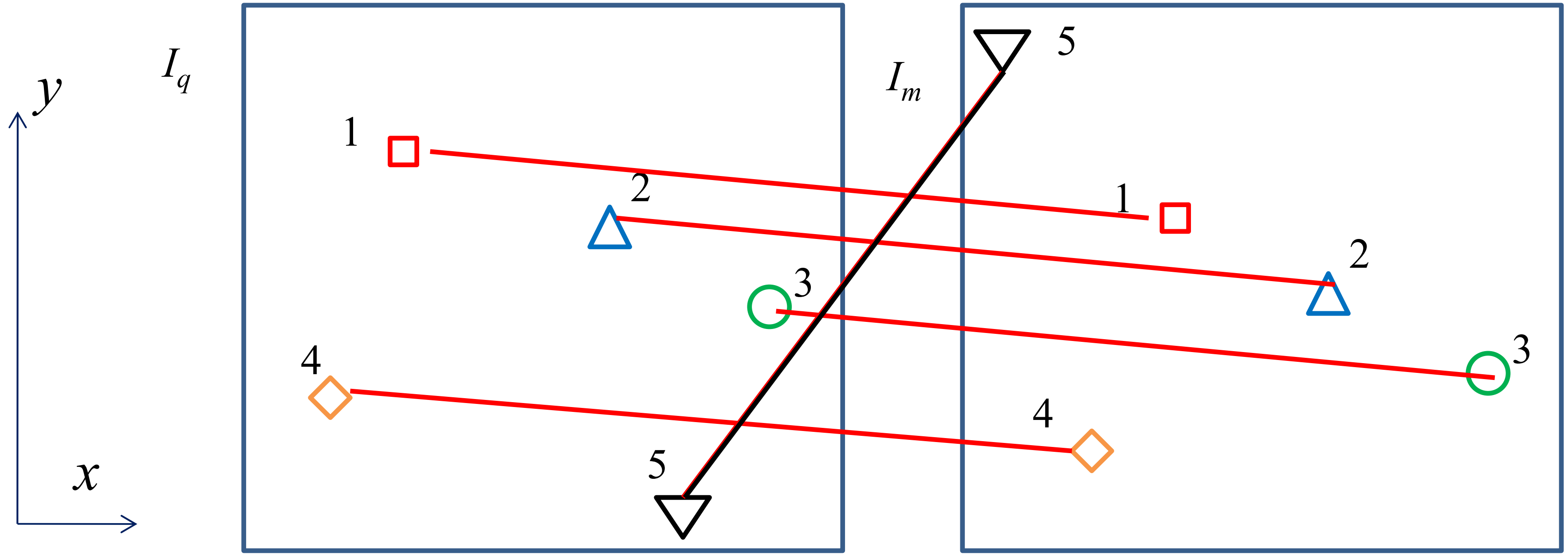
参考点: *i*



$$Xmap=\begin{pmatrix}1 & 0 & 0 & 1\\ 1 & 1 & 0 & 1\\ 1 & 1 & 1 & 1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$Ymap=\begin{pmatrix}1 & 1 & 1 & 1\\ 0 & 1 & 1 & 1\\ 0 & 0 & 1 & 1\\ 0 & 0 & 0 & 1\end{pmatrix}$$

- Wengang Zhou, Yijuan Lu, Houqiang Li, Y. Song, and Qi Tian, "Spatial coding for large scale partial-duplicate web image search," *ACM International Conference on Multimedia* (*MM*), pp.131-140, 2010.



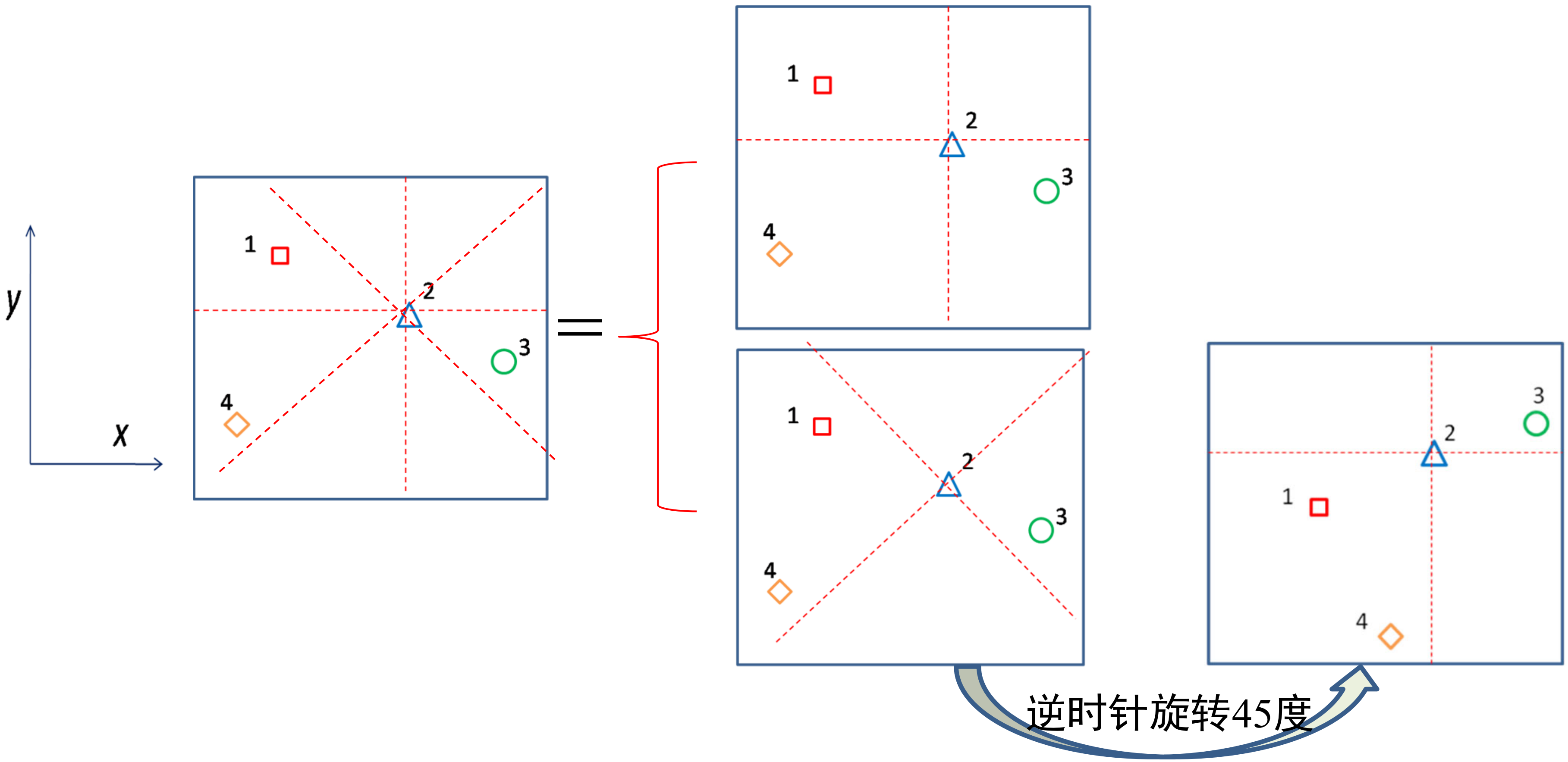
$$Vx = Xq \oplus Xm = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{pmatrix} \oplus \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{pmatrix}, \qquad Sx = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ \boxed{3} \end{pmatrix}$$

逐行求和

$$Vy = Yq \oplus Ym = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \oplus \begin{pmatrix} 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}, \qquad Sy = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \boxed{4} \end{pmatrix}$$

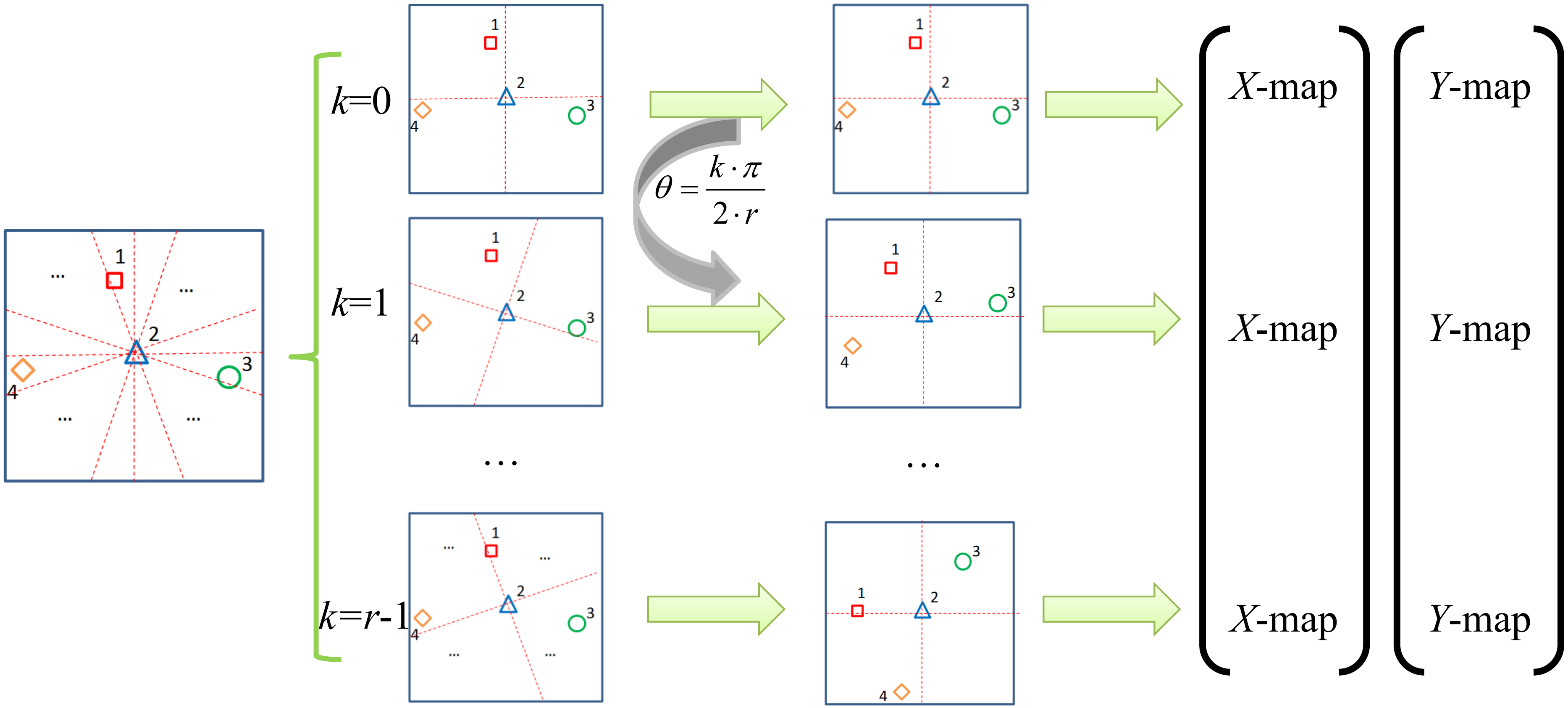
空间编码矩阵生成

- 在前面的例子中，每个象限只有一个部分
  - 现在将每个象限均匀的分成两个部分



空间编码矩阵生成

- 生成空间矩阵GX和GY
  - 每个象限均匀的分成r个部分



- Wengang Zhou, Yijuan Lu, Houqiang Li, Y. Song, and Qi Tian, "Spatial coding for large scale partial-duplicate web image search," *ACM International Conference on Multimedia (MM)*, pp.131-140, 2010.

局部特征匹配的空间校验

- 基于空间矩阵GX和GY的验证
  - 将匹配特征对的空间矩阵进行对比

$$V_x(i,j,k) = GX_q(i,j,k) \oplus GX_m(i,j,k)$$ V_x: 空间矩阵X中不一致的程度

$$V_y(i,j,k) = GY_q(i,j,k) \oplus GY_m(i,j,k)$$ V_y: 空间矩阵Y中不一致的程度

k=0, …, *r*-1; *i*, *j*=1, …, *N*; *N*: 匹配特征对的数量

  - 迭代地**查找**和**删除**最不一致的匹配对

$$S_x(i) = \sum_{k=0}^{r-1}\sum_{j=1}^{N} V_x(i,j,k)$$

$$S_y(i) = \sum_{k=0}^{r-1}\sum_{j=1}^{N} V_y(i,j,k)$$

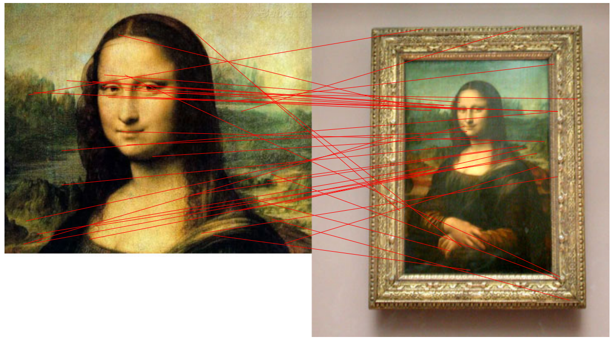
$$i^* = \arg\max_i S_x(i)$$

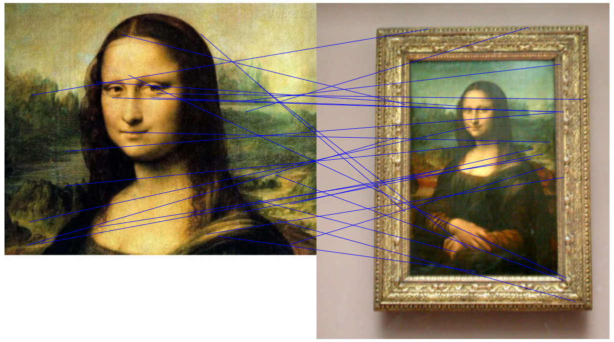
$$j^* = \arg\max_j S_y(j)$$

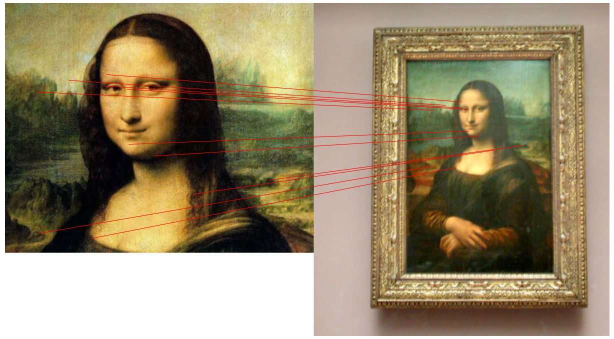
找到 i^*/j^*，并从V_x 和 V_y 中删除对应的行和列

局部匹配的空间校验实例

相关图像







不相关图像

空间验证前

识别的错误匹配对

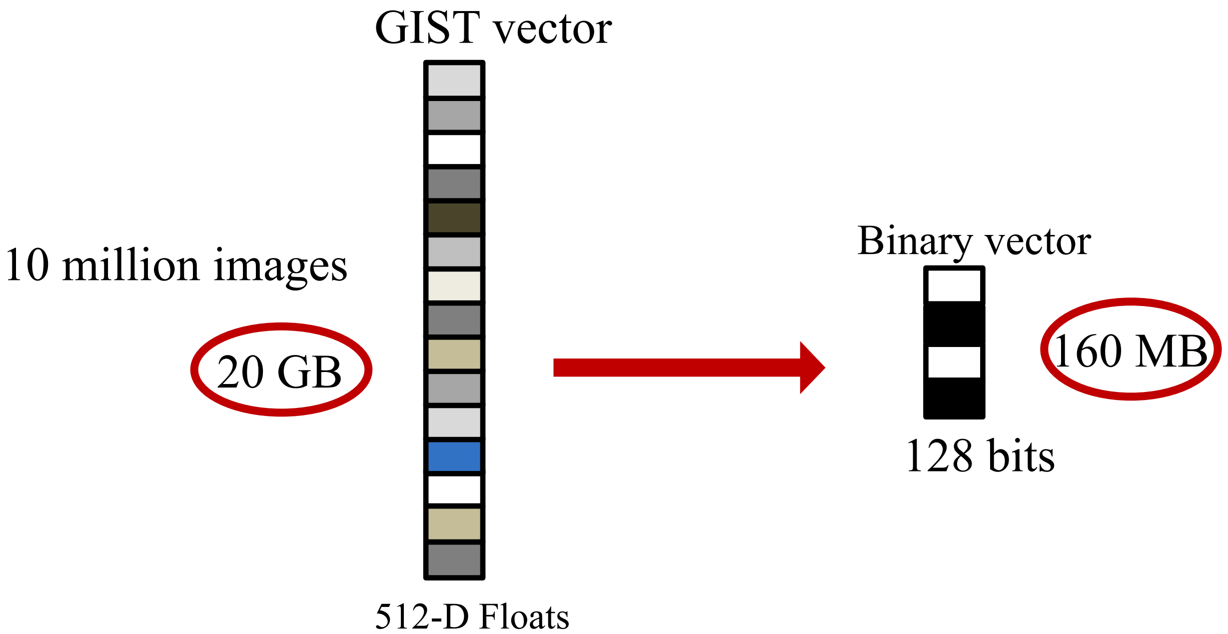
空间验证后

图像识别

- 简单形状检测
- 图像分类
- 图像检索
  - 倒排索引
  - 空间验证
  - 二值哈希

哈希算法

- 大规模数据集图像检索任务的要求
  - 存储开销
  - 检索速度
- 二值哈希算法
  - 减小了存储开销加快了检索速度



L_2 distance

基于异或的汉明距离

汉明距离计算-示例

- 给定两个8-bit的变量x和y（数据类型为uchar），首先计算其异或
$$z = x \oplus y$$
z也是一个8-bit变量， z中比特位为1的个数，即为x和y的汉明距离

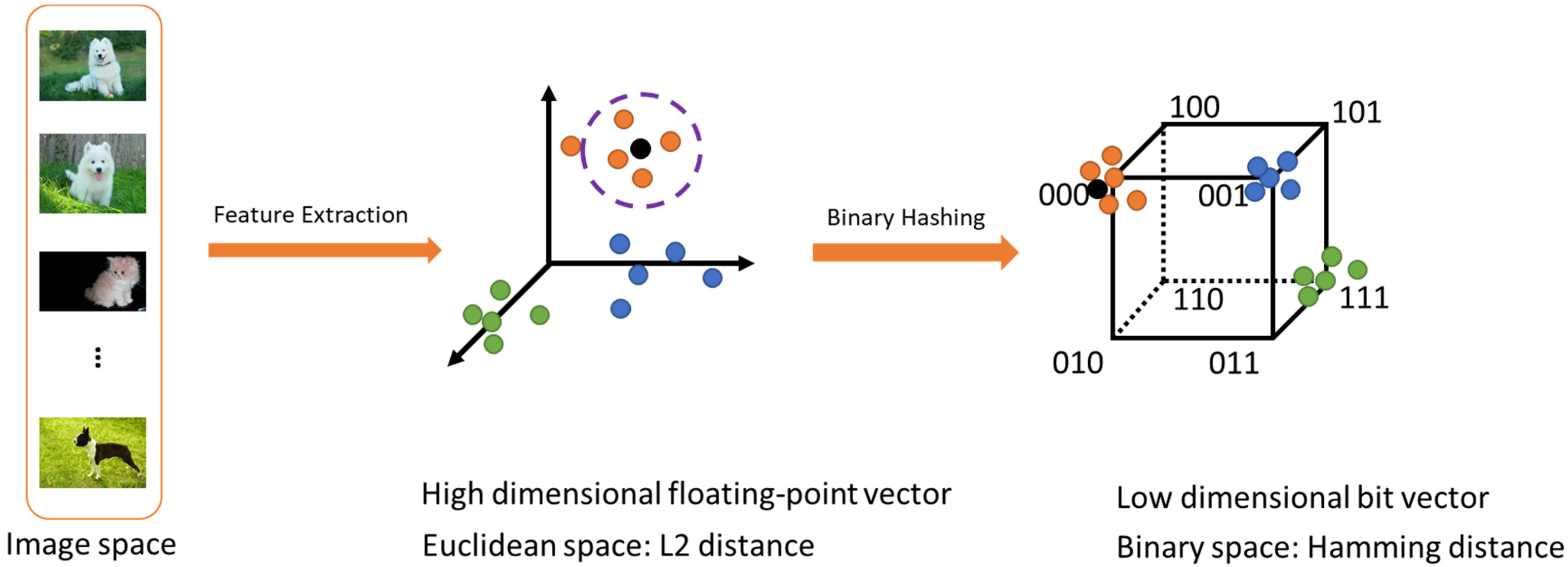
- 为了快速得到z中比特位为1的个数，可事先**离线**计算一个长度为2^8的数组 $\boldsymbol{t}$，其中第 i个数组元素值表示“十进制的正整数i在二进制表达下的比特位为1的个数”
  - 例如$\boldsymbol{t}$[0]=0,
    $\boldsymbol{t}$[1]=1,
    $\boldsymbol{t}$[2]=1,
    $\boldsymbol{t}$[3]=2,
    $\boldsymbol{t}$[4]=1,
    … ,
    $\boldsymbol{t}$[255]=8

- 因此，8-bit变量（数据类型为$uchar$）x和y的汉明距离可通过数组**d**查表得到：$d = \boldsymbol{t}[x \oplus y]$
  - 例如 $x = 4 = (00000100)_2$，
    $y = 8 = (00001000)_2$，
    汉明距离$d = \boldsymbol{t}[x \oplus y] = \boldsymbol{t}[(00001100)_2] = \boldsymbol{t}[12] = 2$
  - 例如 $x = 7 = (00000111)_2$，
    $y = 3 = (00000011)_2$，
    汉明距离$d = \boldsymbol{t}[x \oplus y] = \boldsymbol{t}[(00000100)_2] = \boldsymbol{t}[4] = 1$
- 当计算高比特向量间的汉明距离时，可以将比特向量分段，每段8 bit，分段按上面方法计算汉明距离，然后累加各段的汉明距离
  - 实际实现时，每段长度可以增加到16bit，此时数组 $\boldsymbol{t}$的长度增大为2^{16}

哈希算法

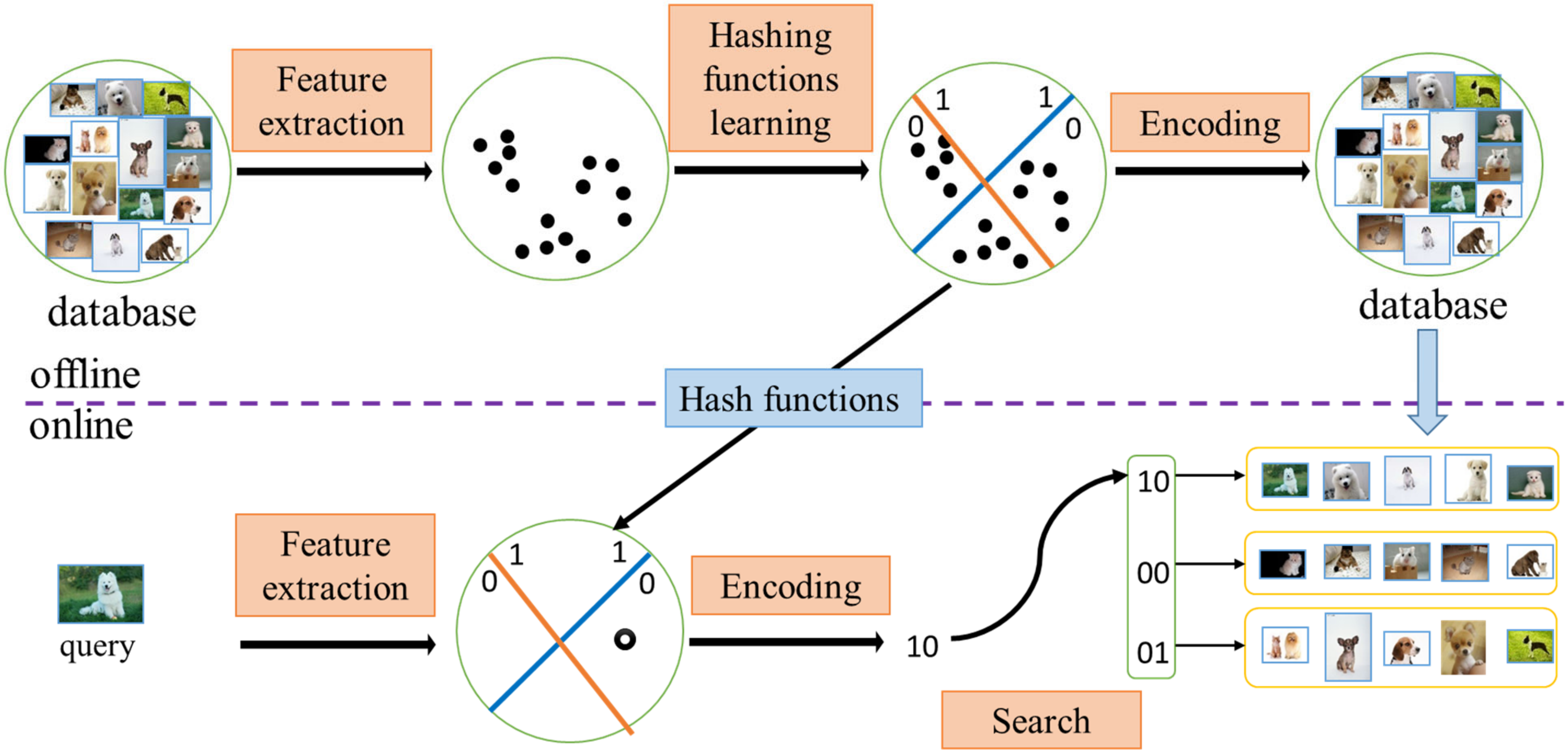
□ 哈希算法原理示意

- 特征提取：首先将一副图像表征为高维特征空间的一个向量
- 二值哈希：然后把图像特征向量映射为高维立方体的一个顶点
  - ✓ 是否存在最优二值哈希函数？ 未知



哈希算法

□ 哈希算法过程图例



哈希算法(1): 局部敏感哈希

- 局部敏感哈希定义
  - 高维空间的两点若距离很近，则这两点映射后的哈希值相同概率较大。
  - 若两点之间的距离较远，则他们哈希值相同概率较小。
- 正整数向量等价变换到汉明空间

2维向量的数据集： A=(1,1)　B=(2,1)　C=(1,2)　D=(2,2)　E=(4,2)　F=(4,3)　⇒　**坐标最大值：** C（4）

每个向量转换为n*C维哈希码：

值为k的坐标转换为长度为C的哈希码，前k位为1，后续位为0

A=(1,1) ⇒ (1000，1000) ⇒ 10001000

B=(2,1) ⇒ (1100，1000) ⇒ 11001000

....

F=(4,3) ⇒ (1111，1110) ⇒ 11111110

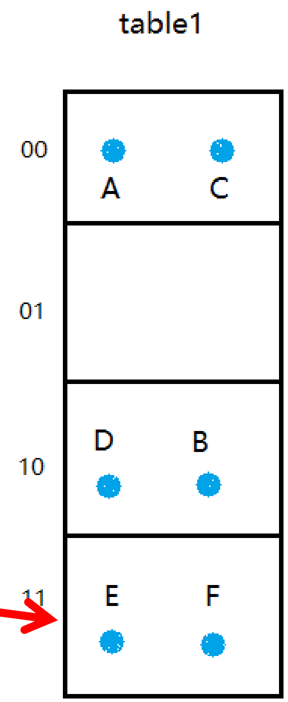
哈希算法(1): 局部敏感哈希

- 一族哈希函数定义

$$h_r(p)=\begin{cases}0, 若p的第r位为0 \\ 1, 若p的第r位为1\end{cases}$$

- 选择k个哈希函数组成构成哈希表g

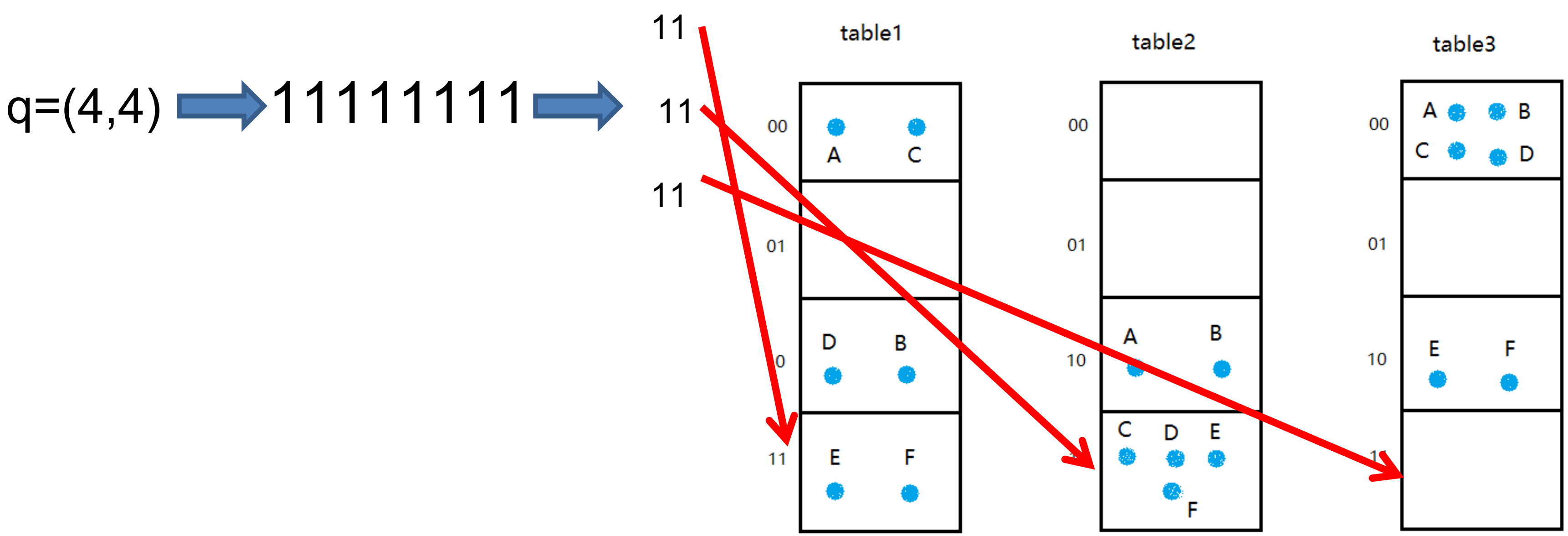
A=10001000 $\xrightarrow{g= h_2(p),h_4(p)}$ A=00
B=11001000 B=10
C=10001100 C=00
D=11001100 D=10
E=11111100 E=11
F=11111110 F=11



- 查询时查找被哈希表映射在同一个桶内的点
  - query q=(4,4) $\Rightarrow$ 11111111 $\xrightarrow{h_2(p),h_4(p)}$ 11
  - 再将q与E,F比较，得到F是q的最近邻

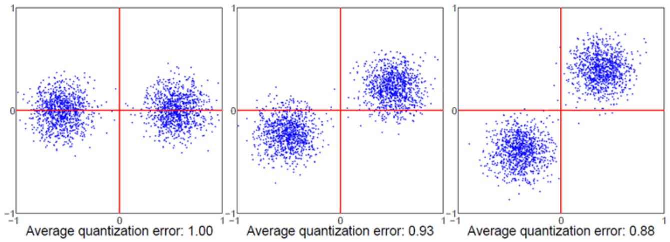
哈希算法(1): 局部敏感哈希

- 选择多组k个哈希函数组成构成多个哈希表g
  - 假设有如下结果。
    - ✓ g1分别抽取第2，4位。
    - ✓ g2分别抽取第1，6位。
    - ✓ g3分别抽取第3，8位

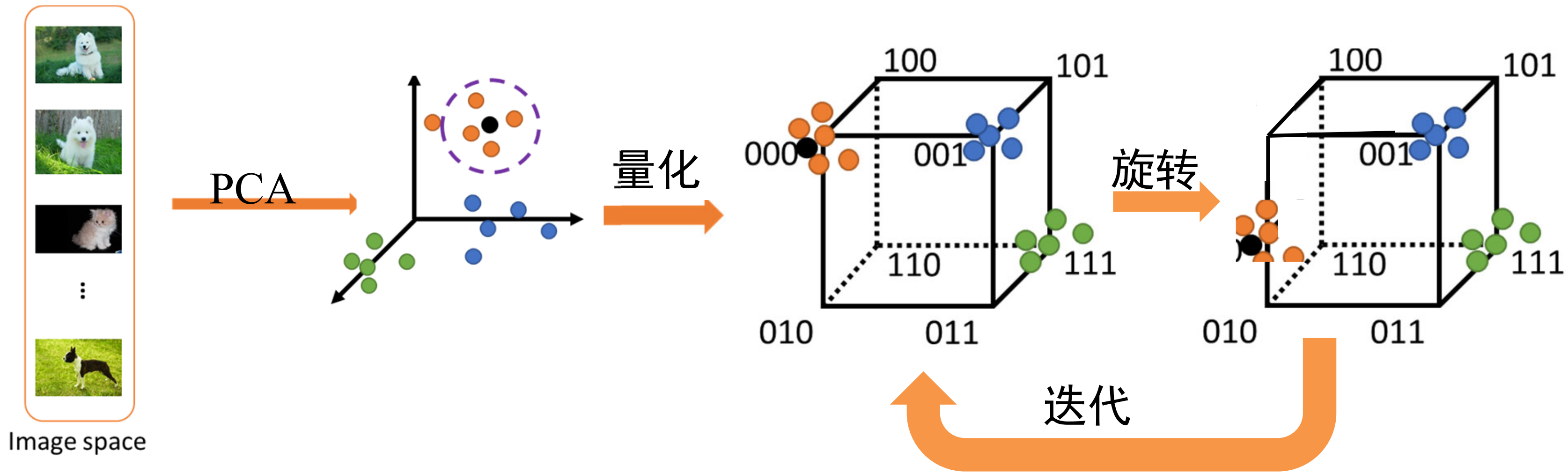


哈希算法(2): 迭代量化（ITQ）

- ITQ算法动机
  - 将原始数据映射到超立方体的顶点，求解量化误差最小的映射
  - 将超立方体在空间中旋转，求解旋转矩阵即能得到最好的映射
  - 迭代这两个步骤



(a) PCA aligned. (b) Random Rotation. (c) Optimized Rotation.



Yunchao Gong and Svetlana Lazebnik, "Iterative Quantization: A Procrustean Approach to Learning Binary Codes," *in CVPR* 2011.

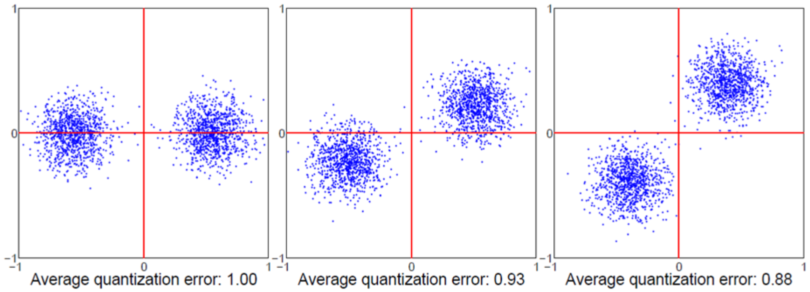
哈希算法(2): 迭代量化（ITQ）

- ITQ(Iterative Quantization)算法步骤
  - 对原始数据进行PCA降维

    $$V = XW$$

  - 最小化量化误差函数

    $$Q(B, R) = \|B - VR\|_F^2$$



(a) PCA aligned. (b) Random Rotation. (c) Optimized Rotation.

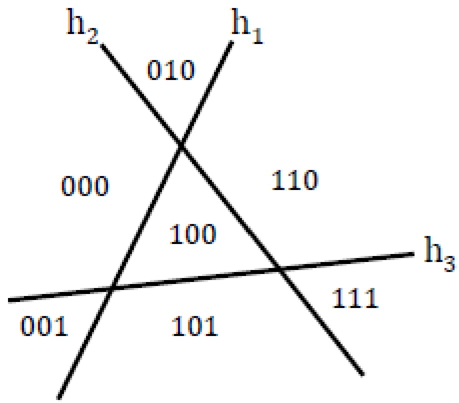
  - ✓ 固定R更新量化结果B: $B = \mathrm{sgn}(VR)$
  - ✓ 固定B更新旋转矩阵R
    - ➢ 计算CxC矩阵 $B^T V$ 的SVD分解 $S\Omega\hat{S}^T$ 然后令 $R = \hat{S}S^T$
  - ✓ 迭代上述步骤，文中为50次

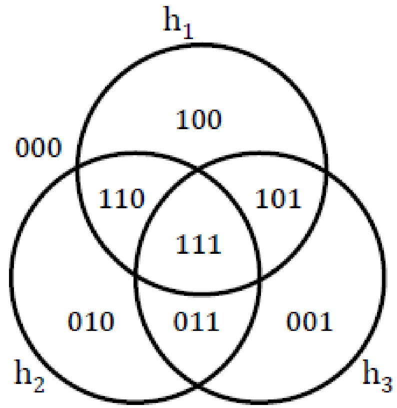
- 优点
  - 没有显式的对量化过程作正交限制
  - 通过学习旋转矩阵代替了对汉明空间的操作

• Yunchao Gong and Svetlana Lazebnik, "Iterative Quantization: A Procrustean Approach to Learning Binary Codes," *in CVPR 2011*.

哈希算法(3): 球面哈希

- □ 动机：用超球面，而非超平面，来分割空间
  - ■ 特征空间更紧凑
  - ■ 在D维特征空间定义一个封闭子空间，只需一个超球面，但需要D+1个超平面
  - ■ 在局部敏感性方面，超球面比超平面更佳



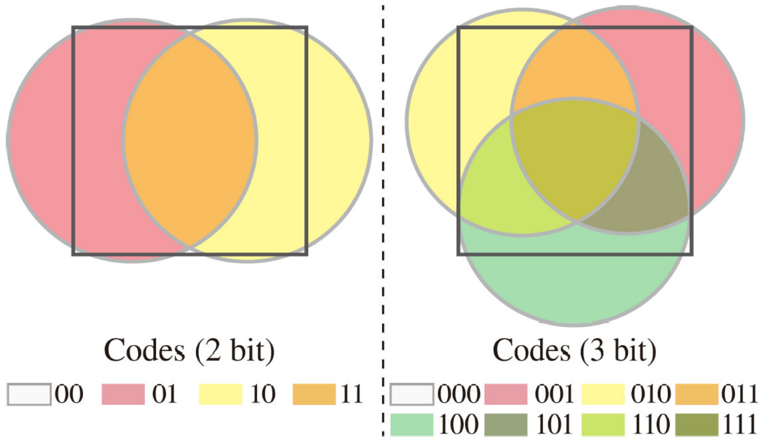


- □ 选择哈希函数即构建超球面：
  - ■ 确定球心和半径

- Heo J P, Lee Y, He J, et al. Spherical Hashing: Binary Code Embedding with Hyperspheres, IEEE TPAMI, 2015.

哈希算法(3): 球面哈希

□ 球哈希示意



$$h_k(x) = \begin{cases} -1 & \text{when } d(p_k, x) > t_k \\ +1 & \text{when } d(p_k, x) \le t_k \end{cases}$$

□ 理想的超球面性质

- **平衡性**：每个球把样本空间均分，即球内球外各占一半
- **独立性**：每个球的交叉部分尽量少，即每个哈希函数相对独立
  - ✓ 任意两个球交叉区域内的样本占总样本的四分之一

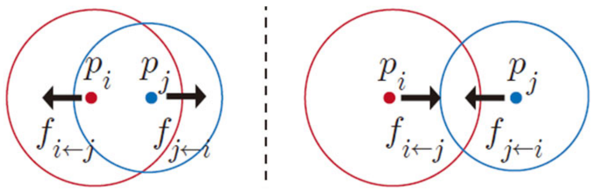
$$\begin{aligned} o_i &= \left| \{ s_k | h_i(s_k) = +1, 1 \le k \le m \} \right|, \\ o_{i,j} &= \left| \{ s_k | h_i(s_k) = +1, h_j(s_k) = +1, 1 \le k \le m \} \right|, \end{aligned}$$

- Heo J P, Lee Y, He J, et al. Spherical Hashing: Binary Code Embedding with Hyperspheres, IEEE TPAMI, 2015.

哈希算法(3): 球面哈希

- 对于训练样本点集$S=\{s_1,s_2,\cdots,s_n\}$，迭代确定超球面
  - 1. 初始化：从训练样本中随机选l个点作为初始球心$p_1,p_2,\cdots,p_l$;
  - 平衡性 2. 对各个球心，确定半径$t_1,t_2,\cdots,t_l$，使得$o_l=\frac{n}{2}$;
  - 3. 对每一对哈希函数，计算$o_{i,j}$;
  - 独立性 4. $\forall i,j$, 计算$f_{i\leftarrow j}=\frac{1}{2}\frac{o_{i,j}-n/4}{\frac{n}{4}}(p_i-p_j)$
  - 独立性 5. $\forall i$,计算$f_i=\frac{1}{l}\sum_{j=1}^{l}f_{i\leftarrow j}$， $p_i=p_i+f_i$
  - 6. 重复步骤2~5，直至收敛，即满足下述条件

$$avg(|o_{i,j}-n/4|)<\varepsilon_m\frac{m}{4} \text{ 且 } std-dev(o_{i,j})<\varepsilon_s\frac{m}{4}$$



- 基于球哈希定义的汉明距离计算：

$$d_{shd}(b_i,b_j)=\frac{|b_i\oplus b_j|}{|b_i\wedge b_j|}$$

其中 $\oplus$:异或；$\wedge$：逻辑与

- Heo J P, Lee Y, He J, et al. Spherical Hashing: Binary Code Embedding with Hyperspheres, IEEE TPAMI, 2015.