



# 第十章：图像识别—图像分类与检索

---

中国科学技术大学  
电子工程与信息科学系



# 图像分类与检索

---

## □ 图像分类

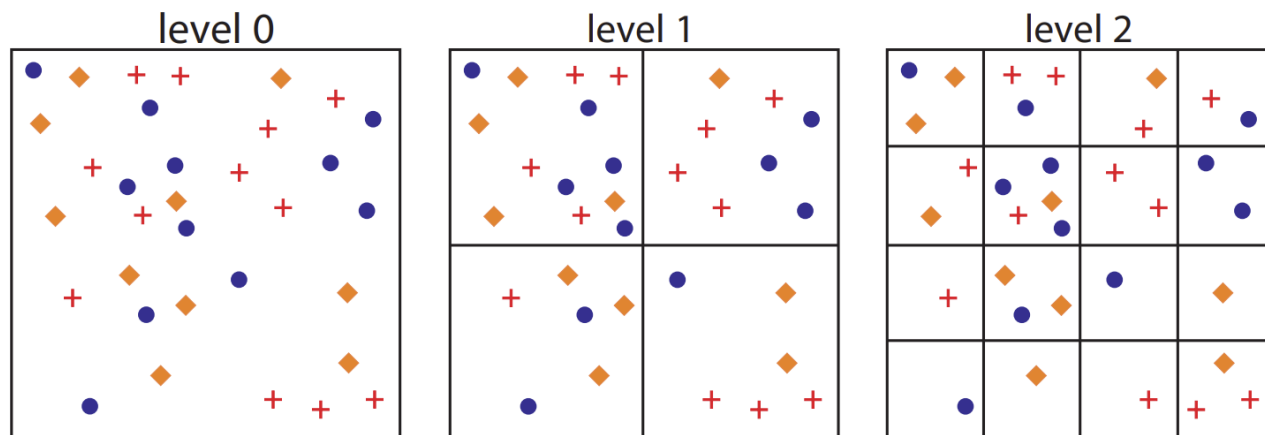
- 空间金字塔匹配
- KNN
- SVM

## □ 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，  $0 \leq l \leq L$



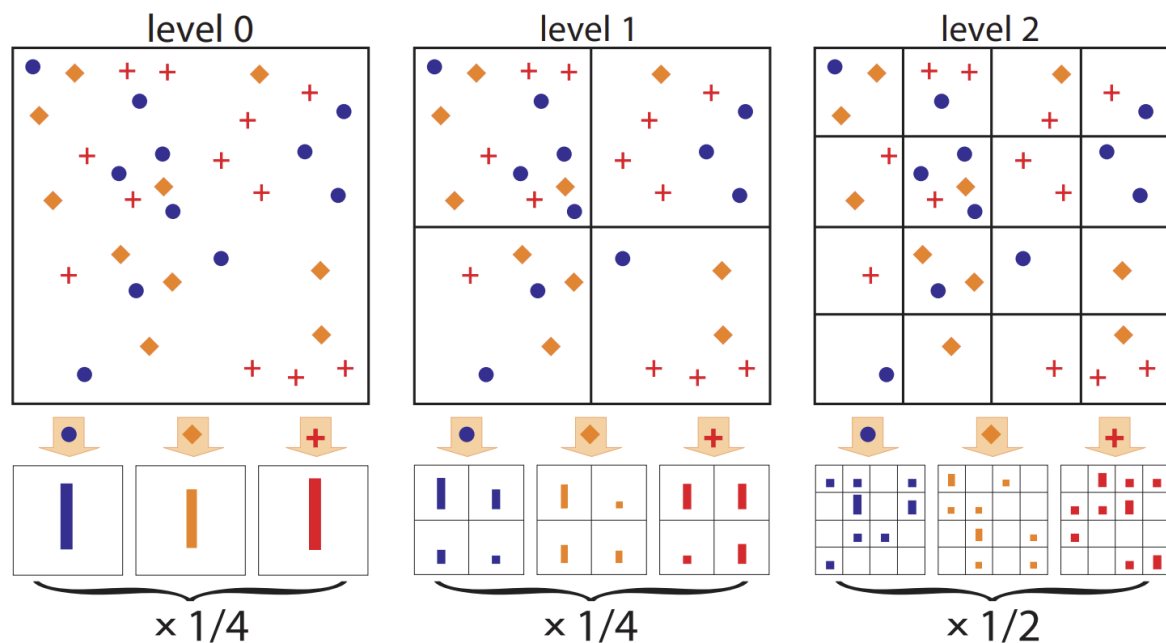
- Lazebnik S, Schmid C, Ponce J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. IEEE CVPR, 2006, 2: 2169-2178.

# 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）：

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



# 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

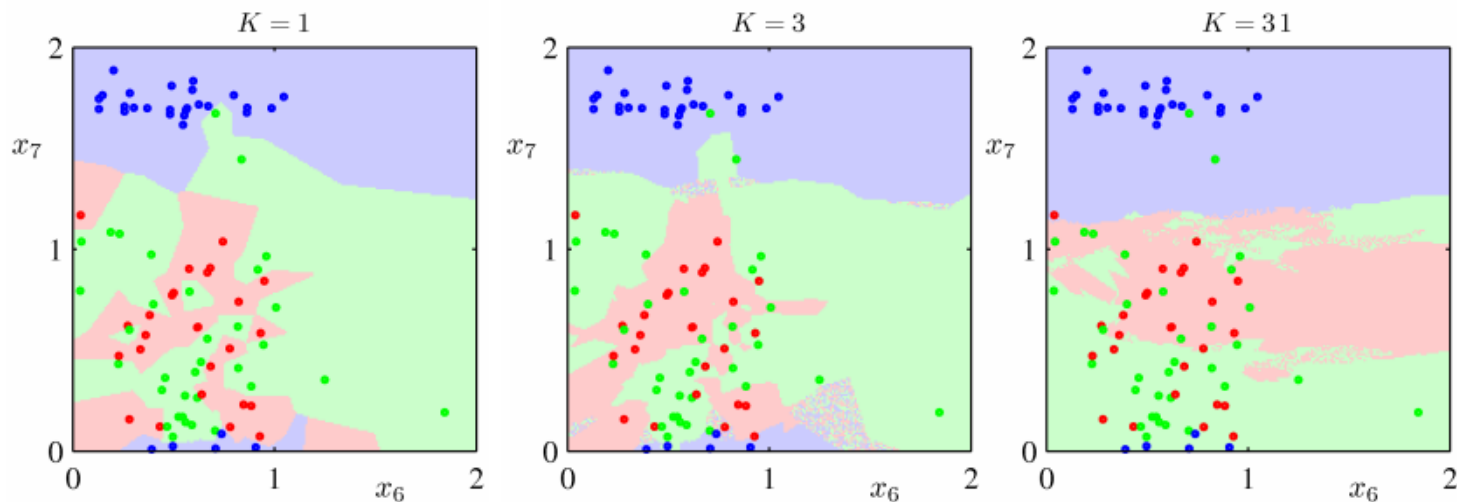
- 对于两幅图像，第  $l$  层匹配的视觉单词 包含第  $l + 1$  层匹配的所有视觉单词
- 所以，相对于第  $l + 1$  层，第  $l$  层的新增匹配定义为  $I^l - I^{l+1}$ ，对应的权值为  $\frac{1}{2^{L-l}}$ ，反比于该层格子宽度
  - ✓ 格子越大，匹配噪声越严重，越不可靠，故权值越小
- **空间金字塔匹配 (SPM)**
  - ✓ 对于两幅图像  $X$  和  $Y$ ，其  $L$  层的空间金字塔匹配核：

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

# 图像分类模型

## □ K-NN分类器

- 将类别预测转换为求解最近邻问题
- 预测类别 → 数据库中近邻图像的分类（多数类别）
- 参数K的影响



- 优点：不需要训练分类模型，简洁
- 缺点：依赖训练数据，预测速度慢（受ANN算法性能影响）

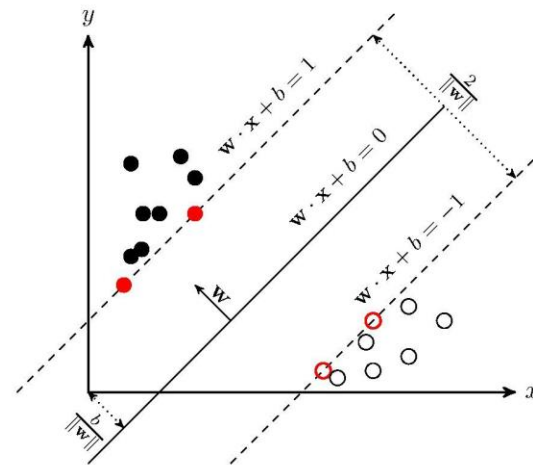
# 图像分类模型：SVM分类器

## □ 基于SVM的二类分类

- 学习几何间隔最大的分离超平面

$$\min_{w,b} \frac{1}{2} \|w\|^2$$

$$s.t. \quad y_i (w \cdot x_i + b) \geq 1, i = 1, 2, \dots, N$$



- 将有约束的原始目标函数转换为无约束的拉格朗日目标函数

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i (y_i (w \cdot x_i + b) - 1)$$

## □ 基于SVM的K类分类

- One-versus-all rule: 训练SVM分类器将某一类与其他(K-1)类分开
- 对于每一类图像，均训练一个one-versus-all SVM分类器
- 测试时，图像被分到响应最强的分类器所对应的类别



# 图像分类与检索

---

- 图像分类

- 图像检索

  - 倒排索引

  - 空间验证

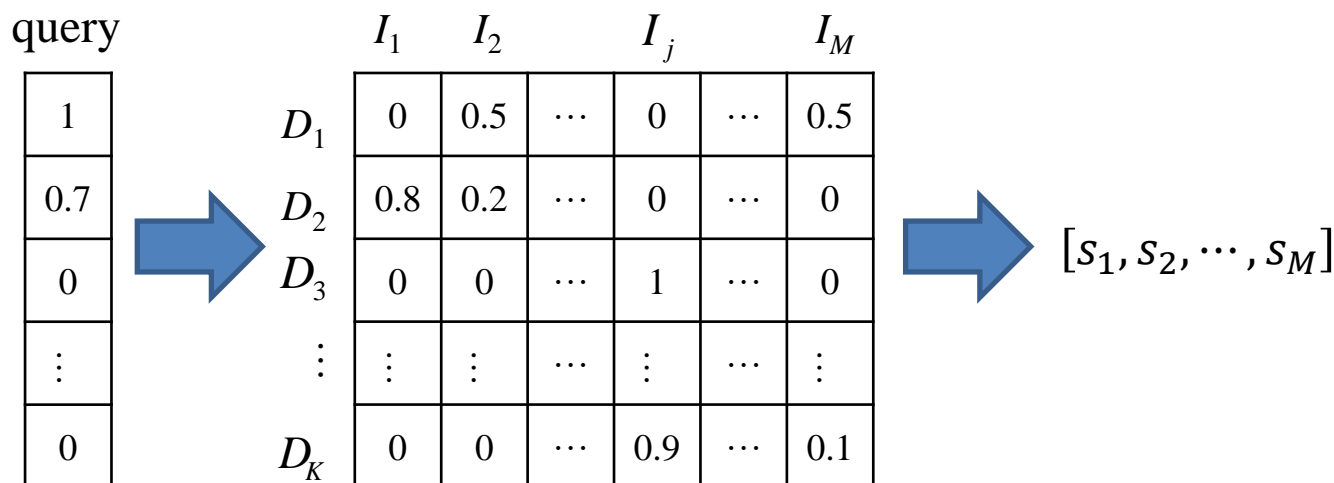
  - 二值哈希



# 图像数据库索引：正向索引

## □ 正向索引

- 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



- 在大规模图像检索中:  $M \gg K$ 。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

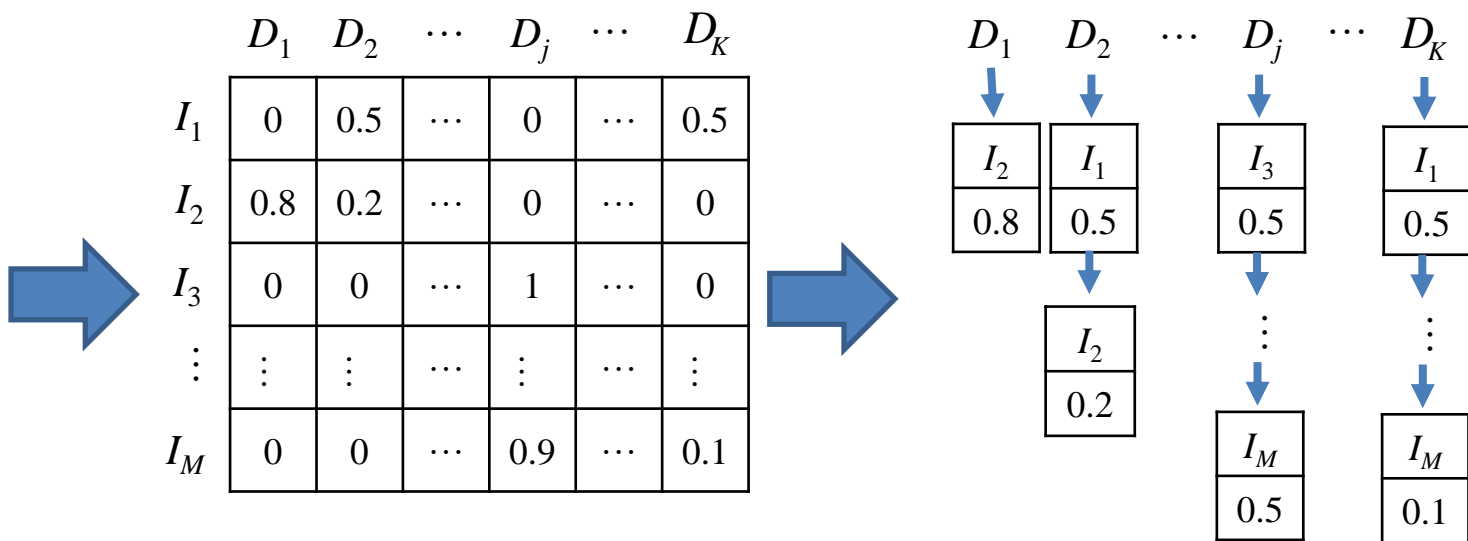
- 图像视觉表征向量的**稀疏性**：非零元素比例低，比如  $< 1\%$
- 只需存储向量中的非零元素  $\rightarrow$  视觉单词在词典中的索引

## □ 倒排索引的优势

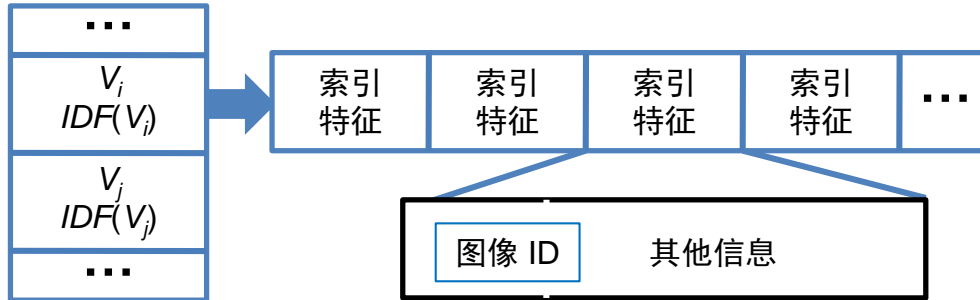
- 高效的存储
- 高效的计算
- ✓ 仅比较向量中的非0元素

$$D(I_q, I_m) = \sum_{i=1}^N |q_i - m_i|^p$$

$$= 2 + \sum_{i|q_i \neq 0, m_i \neq 0} (|q_i - m_i|^p - q_i^p - m_i^p)$$

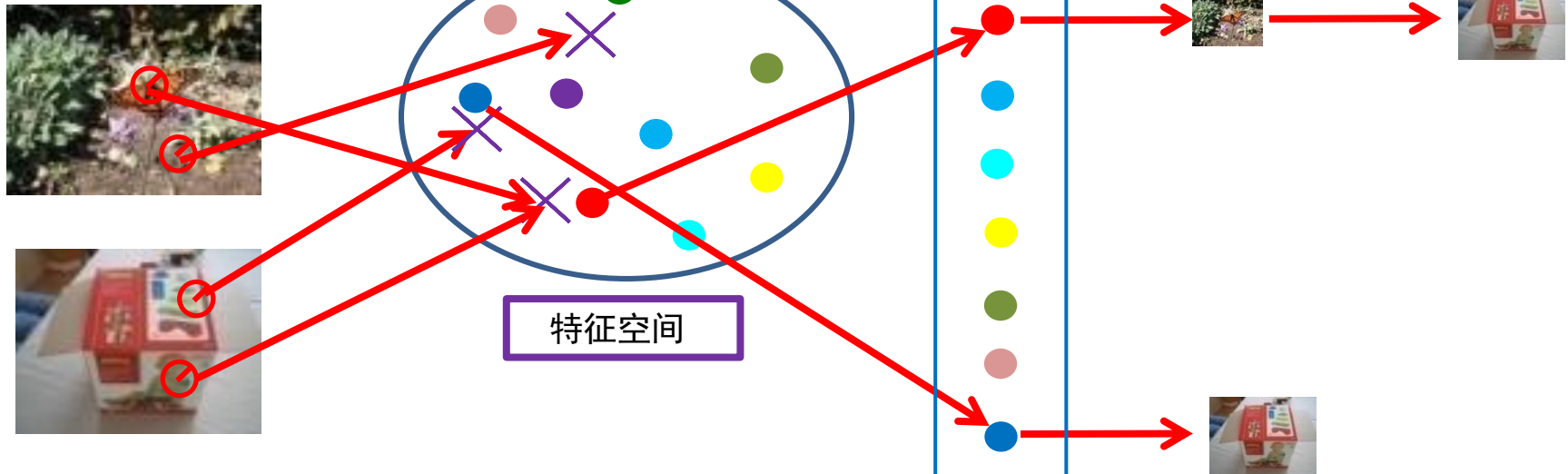


# 图像数据库索引：倒排索引



$$D(I_q, I_m) = \sum_{i=1}^N |q_i - m_i|^p$$

$$= 2 + \sum_{i|q_i \neq 0, m_i \neq 0} (|q_i - m_i|^p - q_i^p - m_i^p)$$





# 图像分类与检索

---

## □ 图像分类

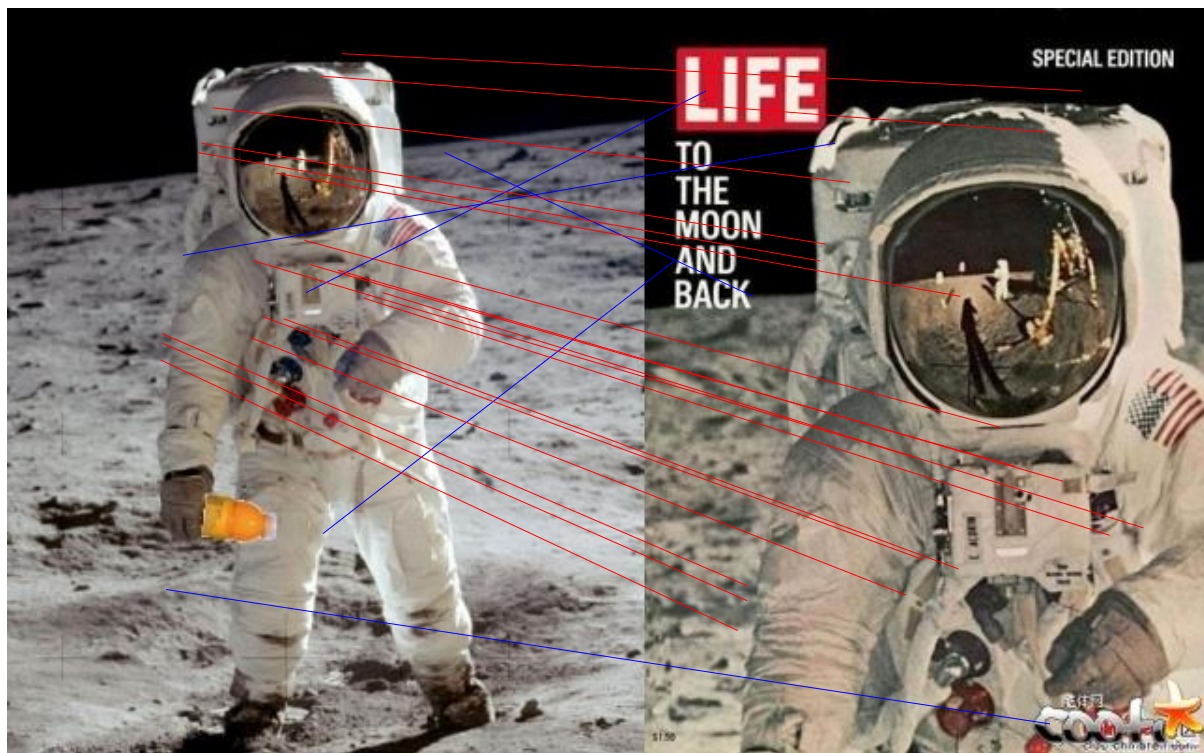
## □ 图像检索

- 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

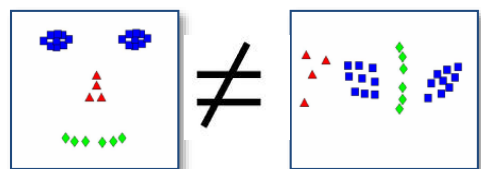
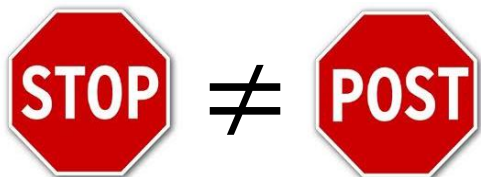


红线: 几何一致的匹配点对  
蓝线: 几何不一致的匹配点对

# 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



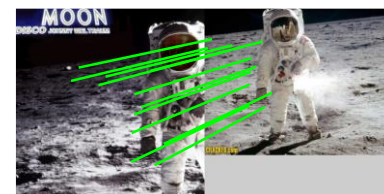
空间布局 vs. 视觉语义



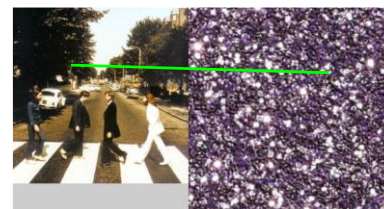
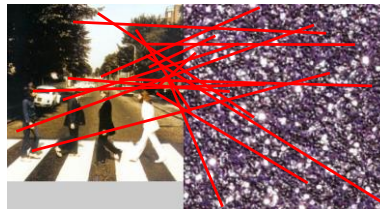
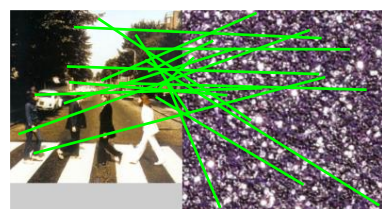
原始匹配



噪声单词匹配



几何一致的匹配



## □ 困难和挑战

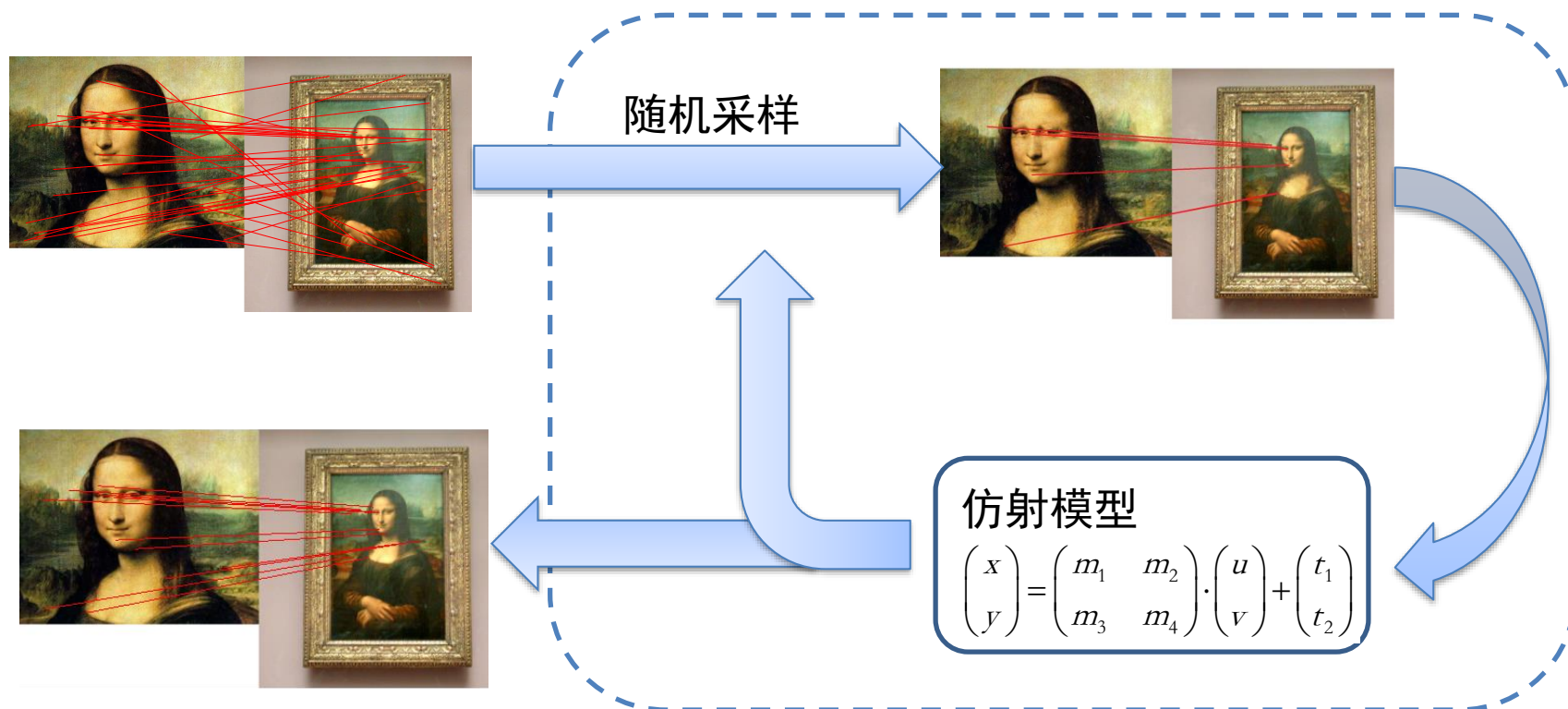
- 几何上下文**结构化**表达：便于图像匹配
- 几何上下文**快速**匹配：保证实时检索



# 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.*, **R**ANdom **S**Ample **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

## □ RANdom SAmple Consensus (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 $O(N^3)$



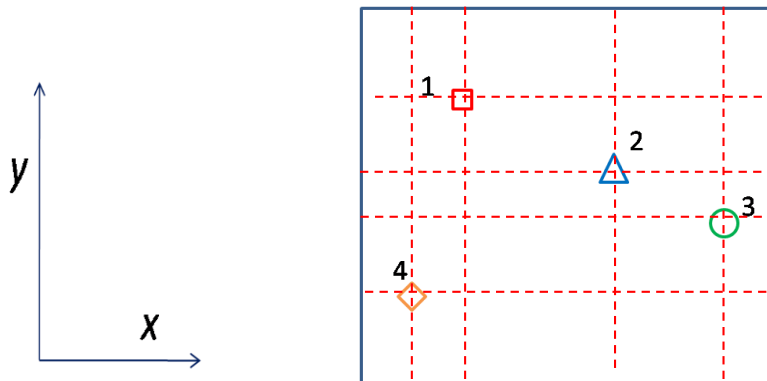
# 几何校验(2): 空间编码(Spatial Coding)

## □ 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

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

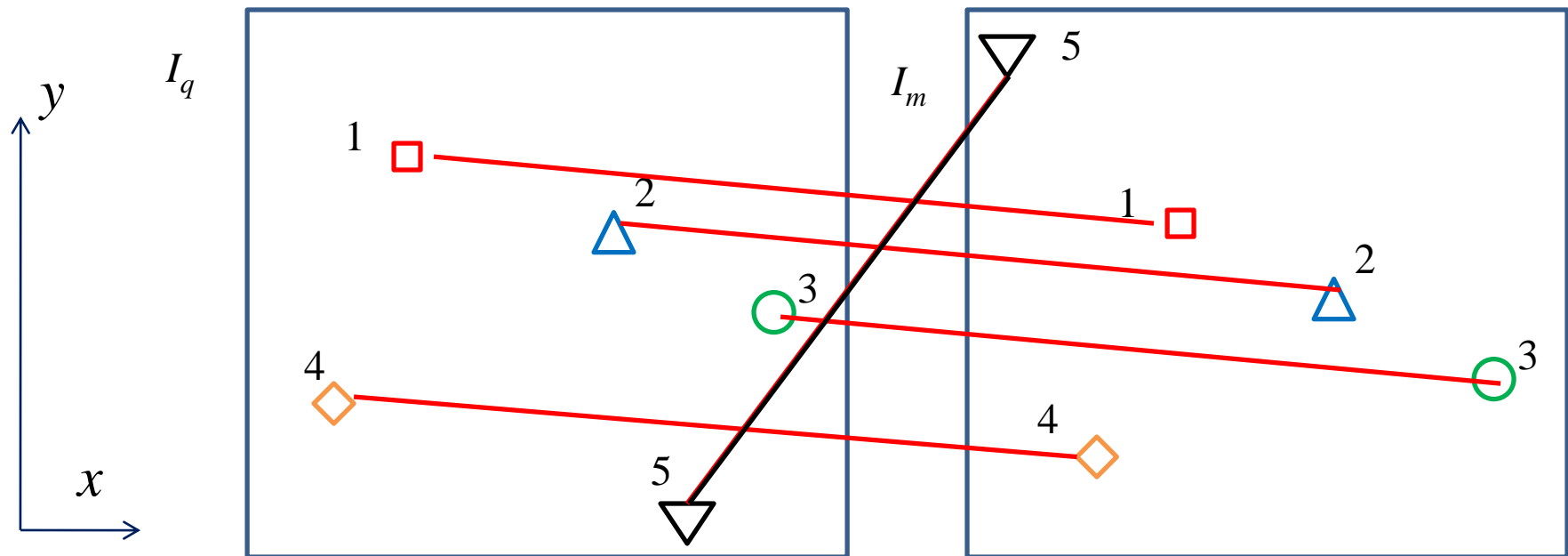
参考点:  $i$



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

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

- 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.



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

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

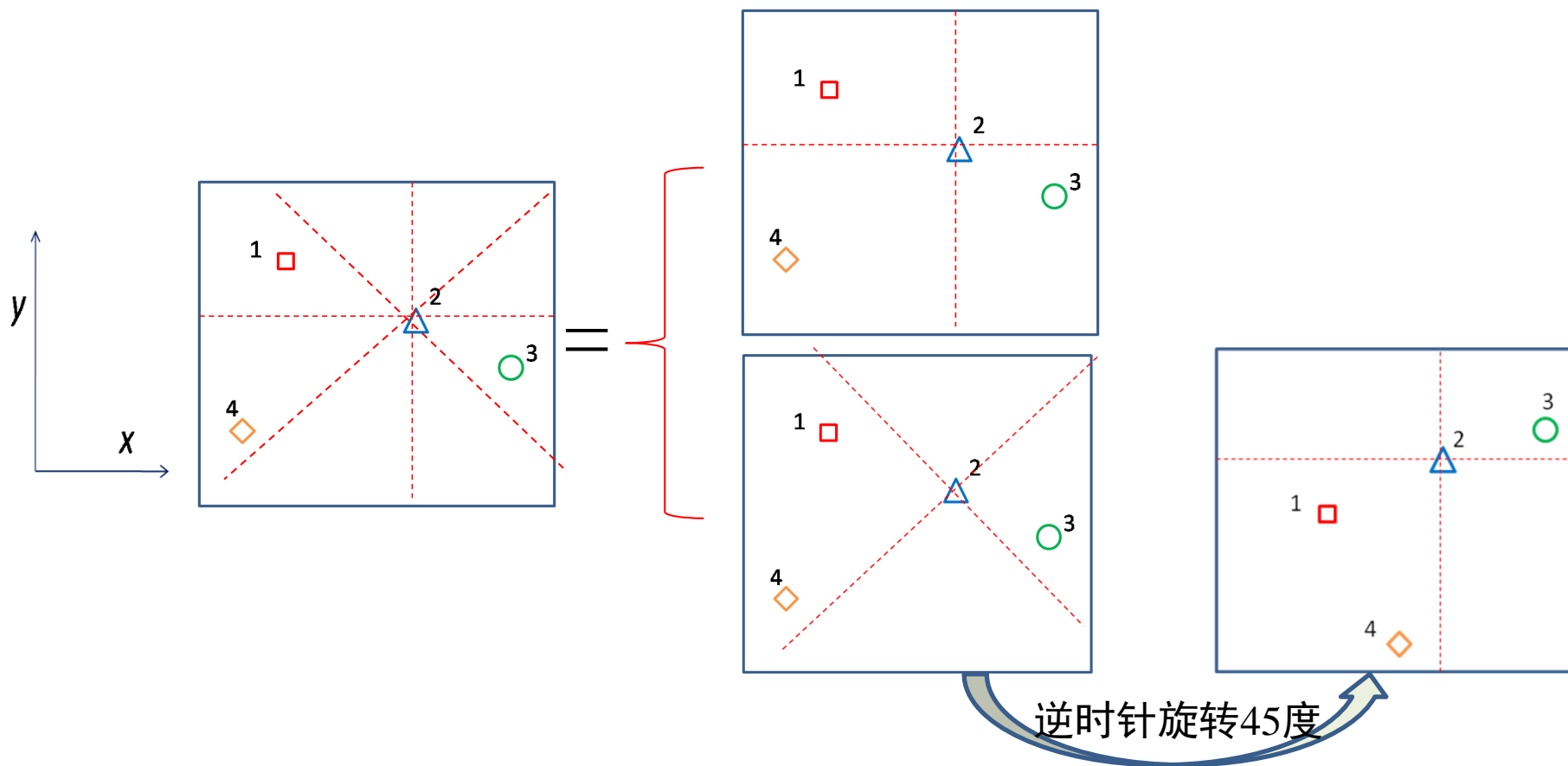
逐行求和

$$Sx = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ \boxed{3} \end{pmatrix}, \quad Sy = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ \boxed{4} \end{pmatrix}$$

18

# 空间编码矩阵生成

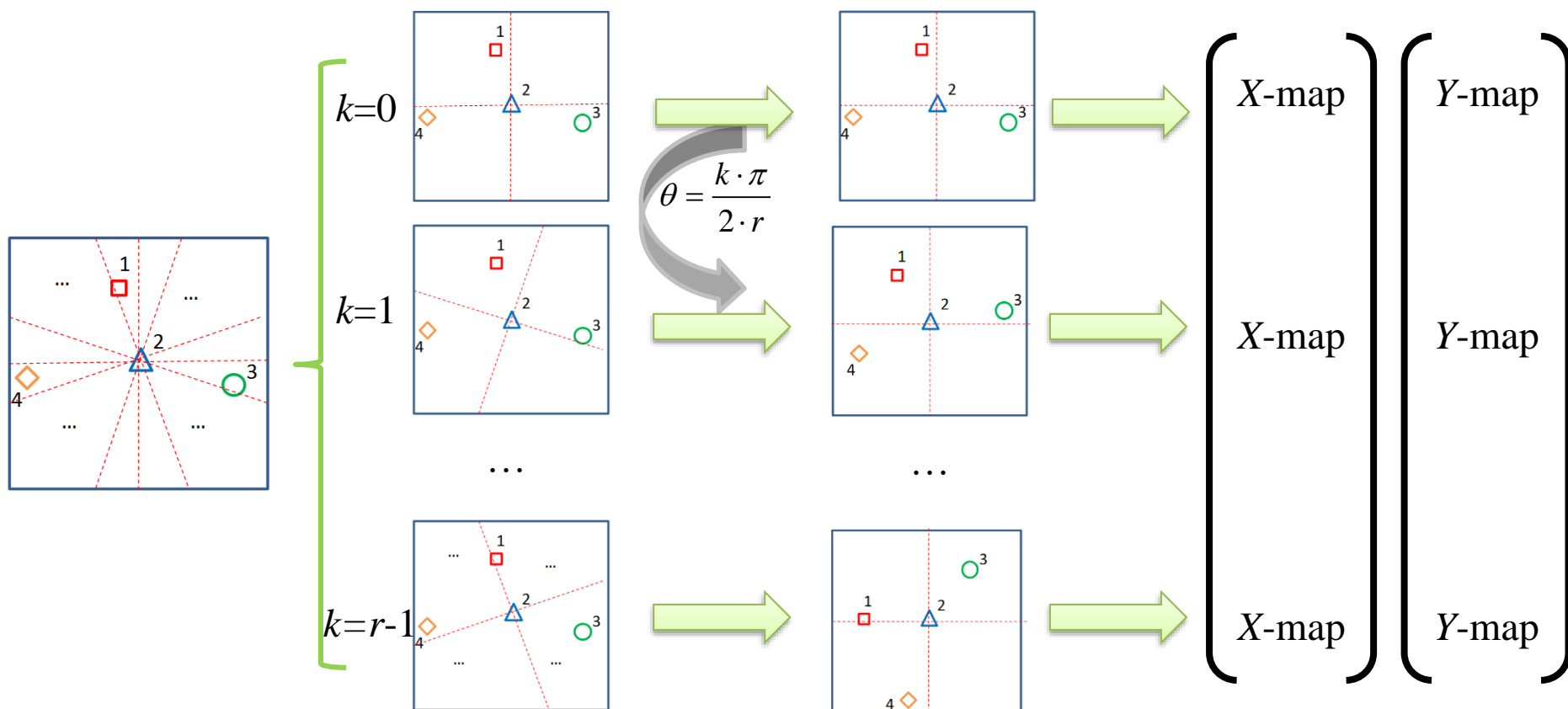
- 在前面的例子中，每个象限只有一个部分
  - 现在将每个象限均匀的分成两个部分



# 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

$k=0, \dots, r-1; i, j=1, \dots, N; N$ : 匹配特征对的数量

### ■ 迭代地查找和删除最不一致的匹配对

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

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

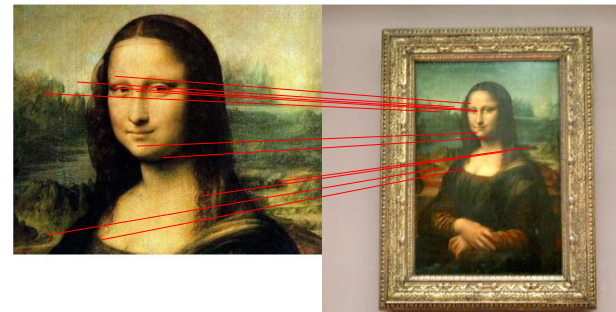
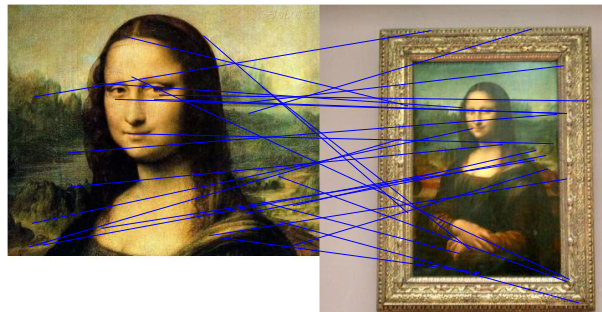
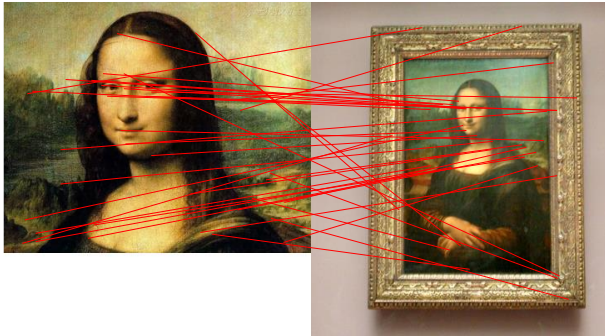
$$i^* = \arg \max_i S_x(i)$$

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

Identify  $i^*/j^*$   
and remove the  
corresponding row  
and column from  
both  $V_y$  and  $V_x$

# 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



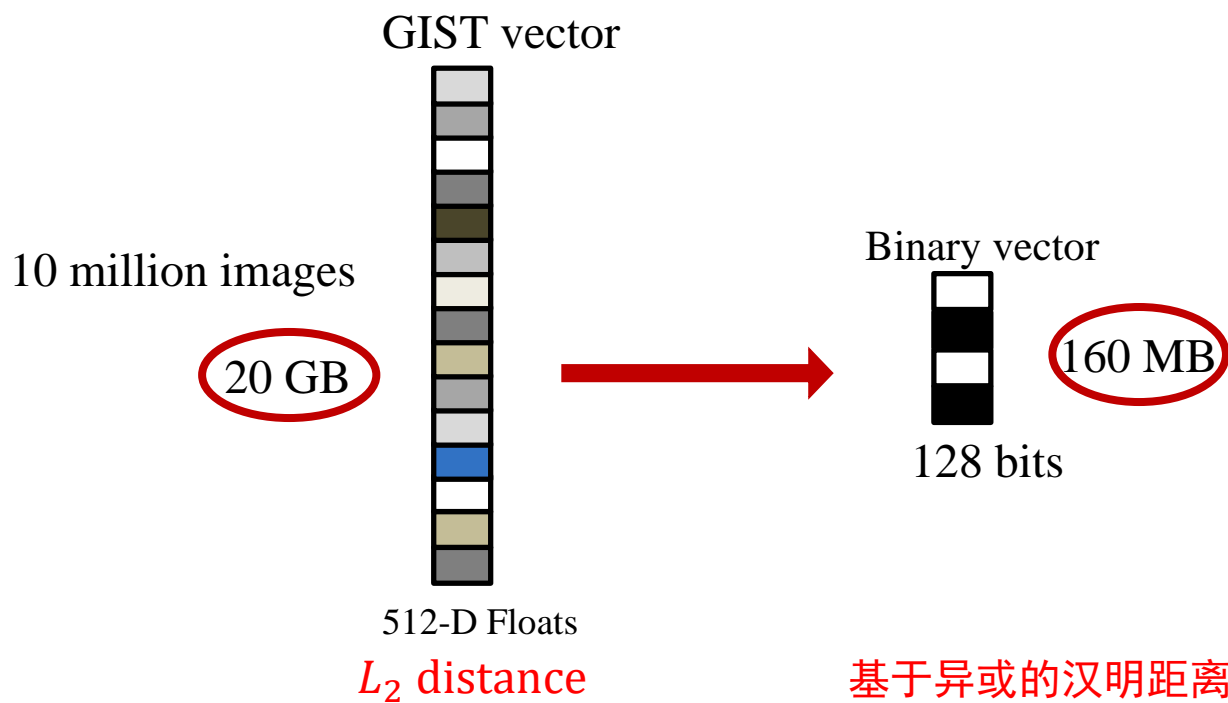
# 哈希算法

## □ 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

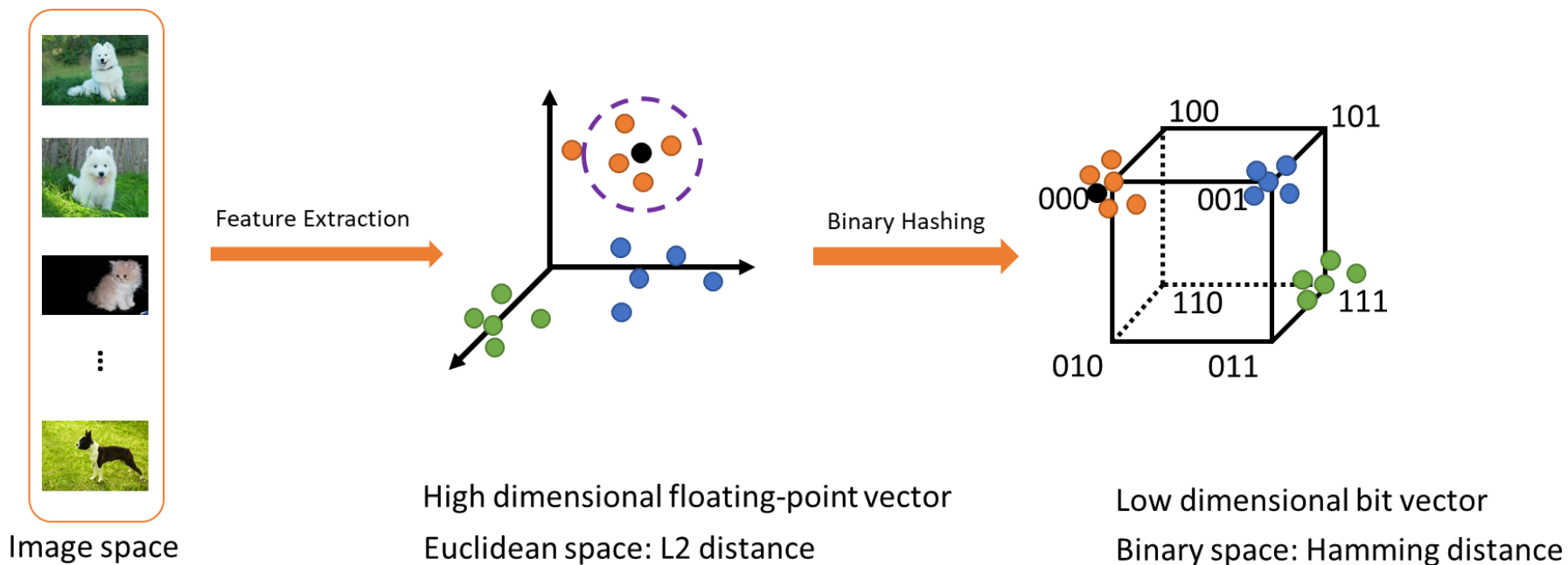




# 哈希算法

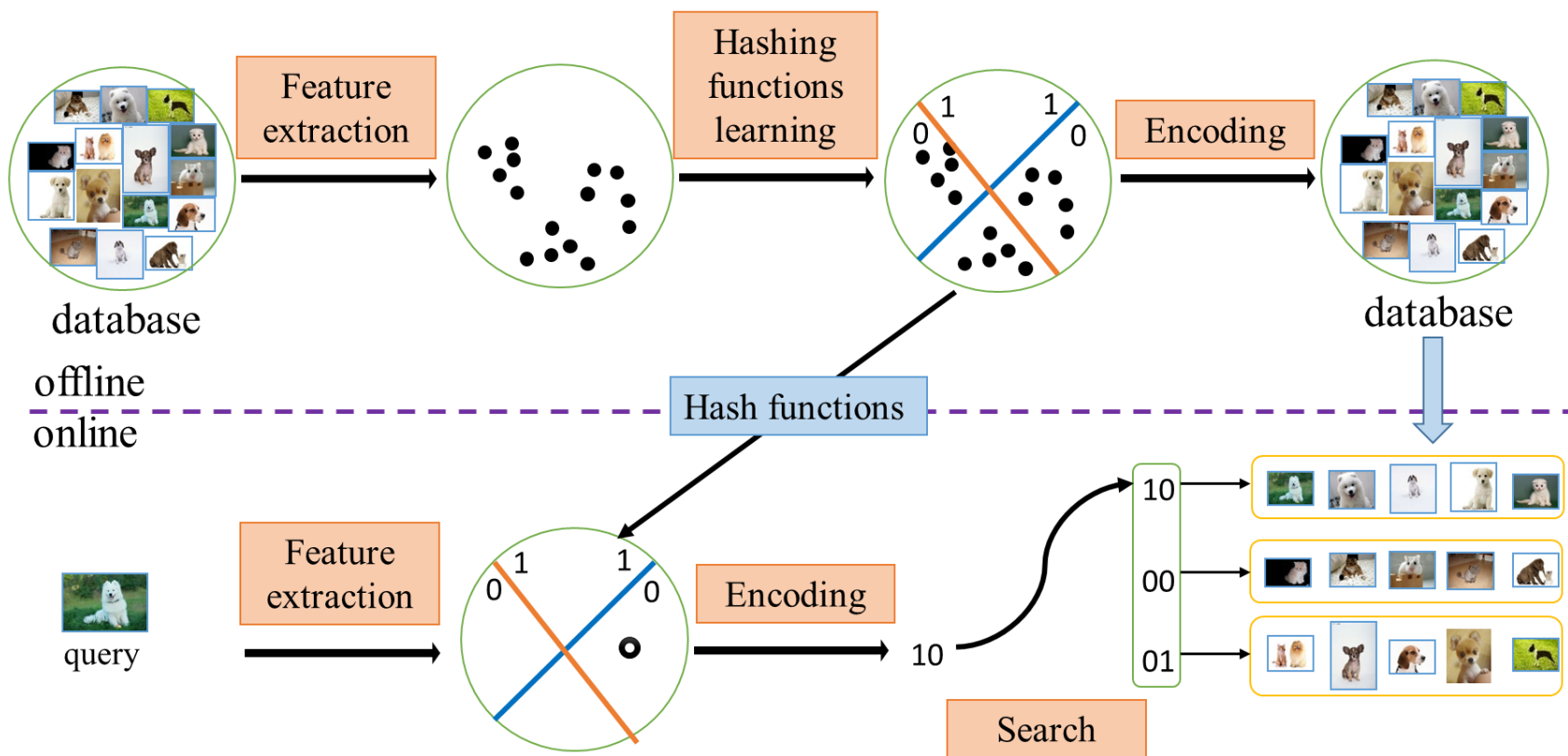
## □ 哈希算法原理示意

- 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。

## □ 正整数向量投影到汉明空间

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

每个向量转换为 $n \times C$ 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

$A=(1,1) \rightarrow (1000, 1000) \rightarrow 10001000$

$B=(2,1) \rightarrow (1100, 1000) \rightarrow 11001000$

.....

$F=(4,3) \rightarrow (1111, 1110) \rightarrow 11111110$

# 哈希算法(1): 局部敏感哈希

## □ 一族哈希函数定义

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

## □ 选择k个哈希函数组成构成哈希表g

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

table1

00	A	C
01		
10	D	B
11	E	F

## □ 查询时查找被哈希表映射在同一个桶内的点

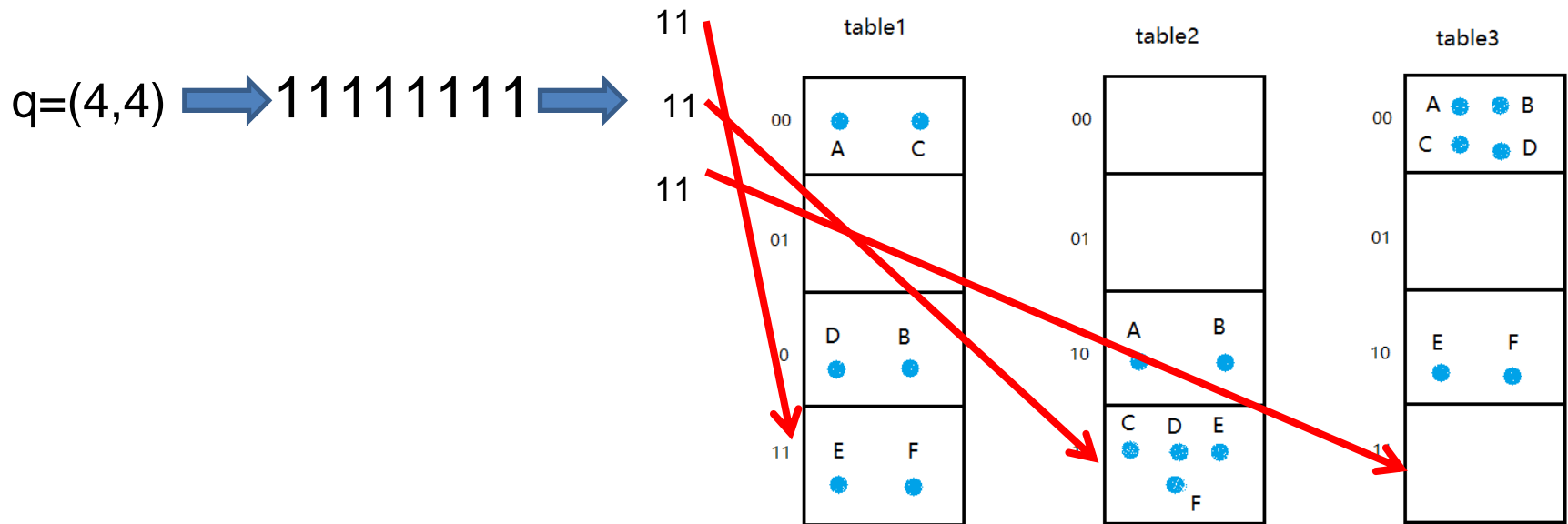
- query  $q=(4,4) \xrightarrow{h_2(p), h_4(p)} 11111111 \xrightarrow{h_2(p), h_4(p)} 11$
- 再将q与E,F比较, 得到F是q的最近邻

# 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。

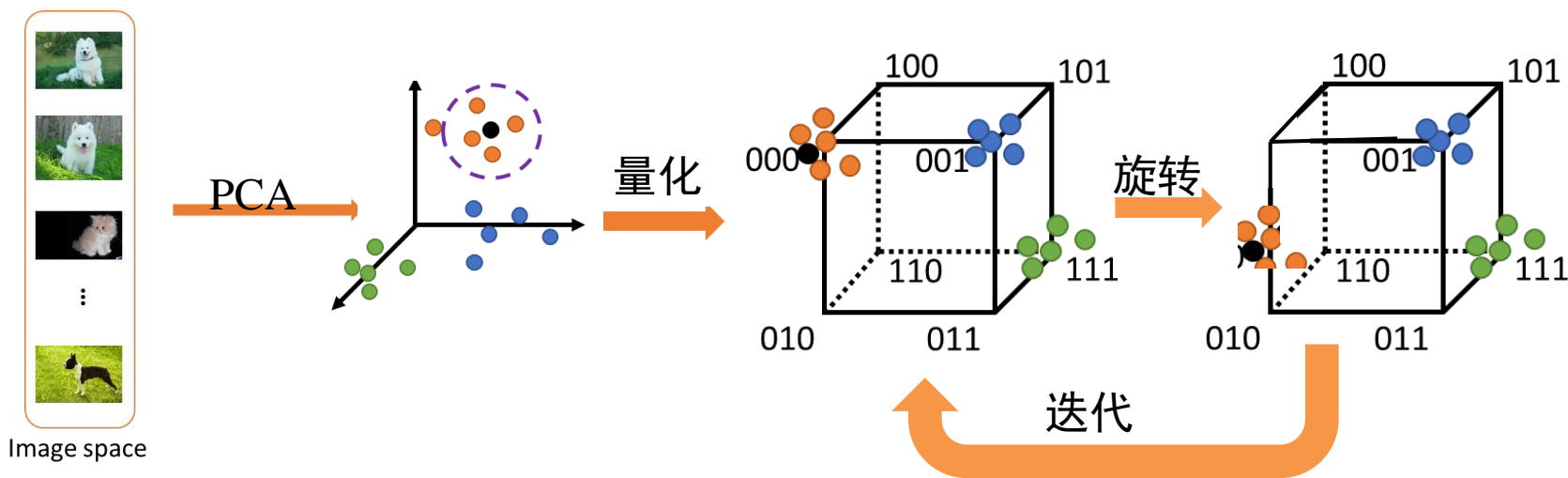
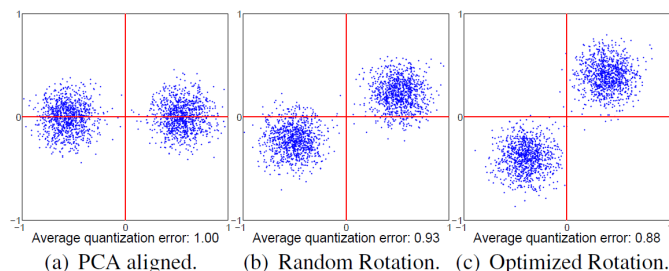
- ✓ g1分别抽取第2, 4位。
- ✓ g2分别抽取第1, 6位。
- ✓ g3分别抽取第3, 8位



# 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



# 哈希算法(2): 迭代量化 (ITQ)

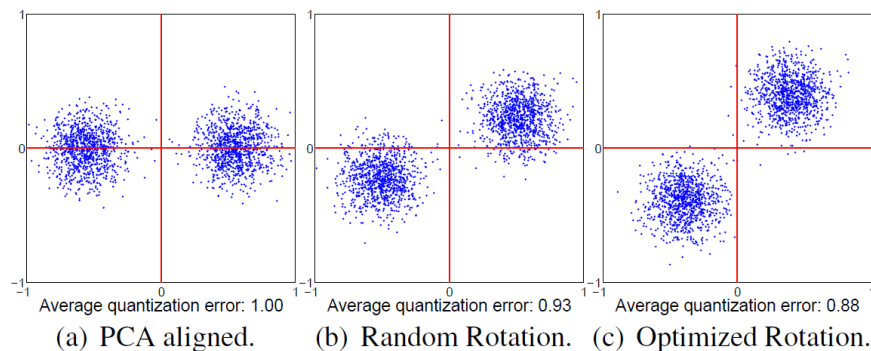
## □ ITQ(Iterative Quantization)算法步骤

- 对原始数据进行PCA降维

$$V = XW$$

- 最小化量化误差函数

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



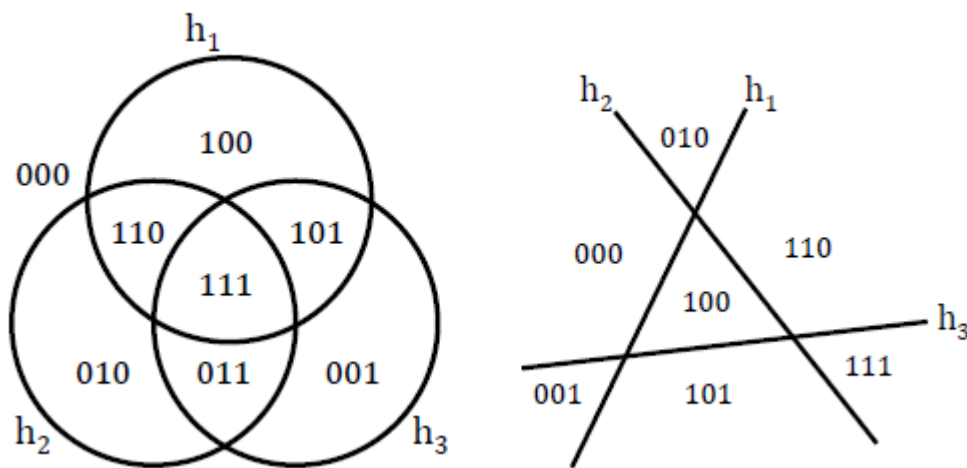
- ✓ 固定R更新B:  $B = \text{sgn}(VR)$
- ✓ 固定B更新R
  - 计算CxC矩阵  $B^T V$  的SVD分解  $S\Omega\hat{S}^T$  然后令  $R = \hat{S}S^T$
- ✓ 迭代上述步骤，文中为五十次

## □ 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

# 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
  - 分割D维特征空间需要一个超球面，D+1个超平面
  - 局部敏感性较超平面更佳

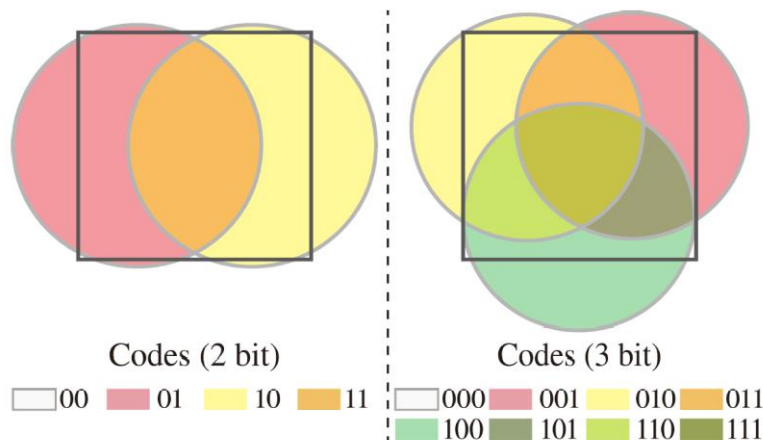


- 选择哈希函数即构建超球面：
  - 确定球心和半径



# 哈希算法(3): 球面哈希

## □ 球哈希示意



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

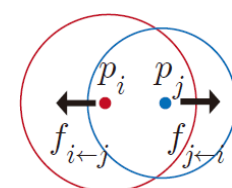
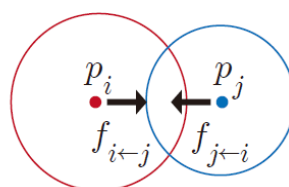
## □ 理想的超球面性质

- 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

$$o_i = |\{s_k | h_i(s_k) = +1, 1 \leq k \leq m\}|,$$

$$o_{i,j} = |\{s_k | h_i(s_k) = +1, h_j(s_k) = +1, 1 \leq k \leq m\}|,$$

# 哈希算法(3): 球面哈希

- 对于训练样本点集  $S = \{s_1, s_2, \dots, s_n\}$ , 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  $p_1, p_2, \dots, p_l$ ;
  - 平衡性 ■ 2. 对各个球心, 确定半径  $t_1, t_2, \dots, t_l$ , 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
 
$$\text{avg}(|o_{i,j} - n/4|) < \varepsilon_m \frac{m}{4} \text{ 且 } \text{std} - \text{dev}(o_{i,j}) < \varepsilon_s \frac{m}{4}$$

□ 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