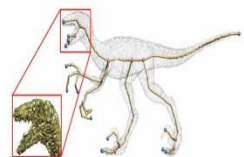


# 计算机视觉 ——目标表达和描述

2022年春季

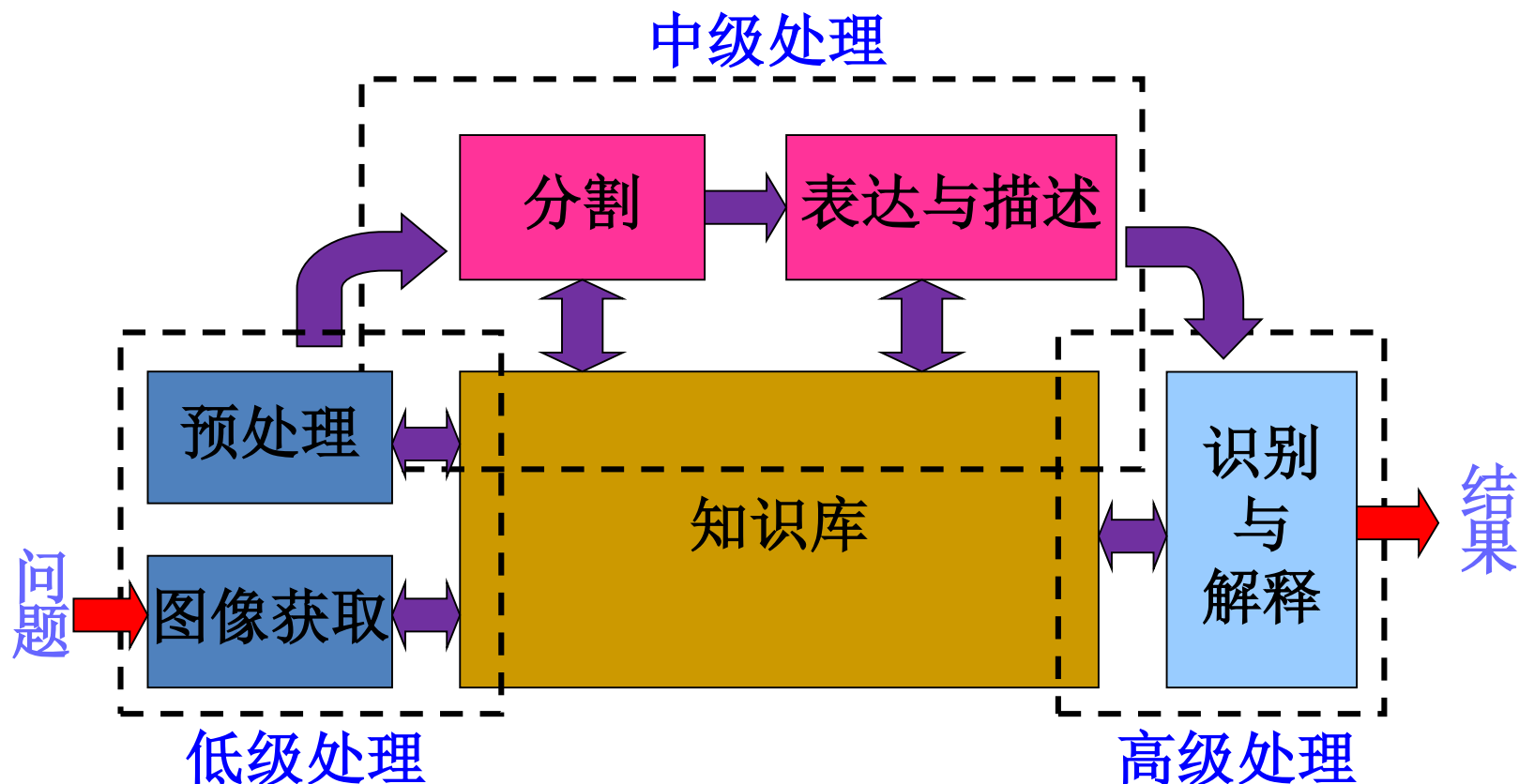
桑农 王岳环

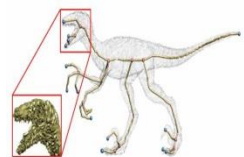
yuehwang@hust.edu.cn



# 目标表达和描述

- 目标表达和描述在图像分析系统的位置





# 目标表达和描述

---

数据来源:

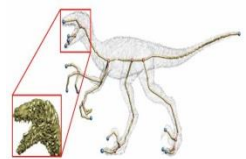
- (图像分割等得到的) 感兴趣区域

处理:

- 对这些区域进行形式化表达和描述

目的:

- 从图像中进一步获取有用信息, 为目标识别解译等提供数据支撑

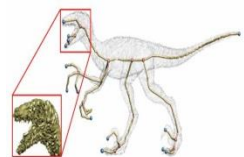


# 目标表达和描述

---

- 颜色特征





# 目标表达和描述

- 形状特征



A



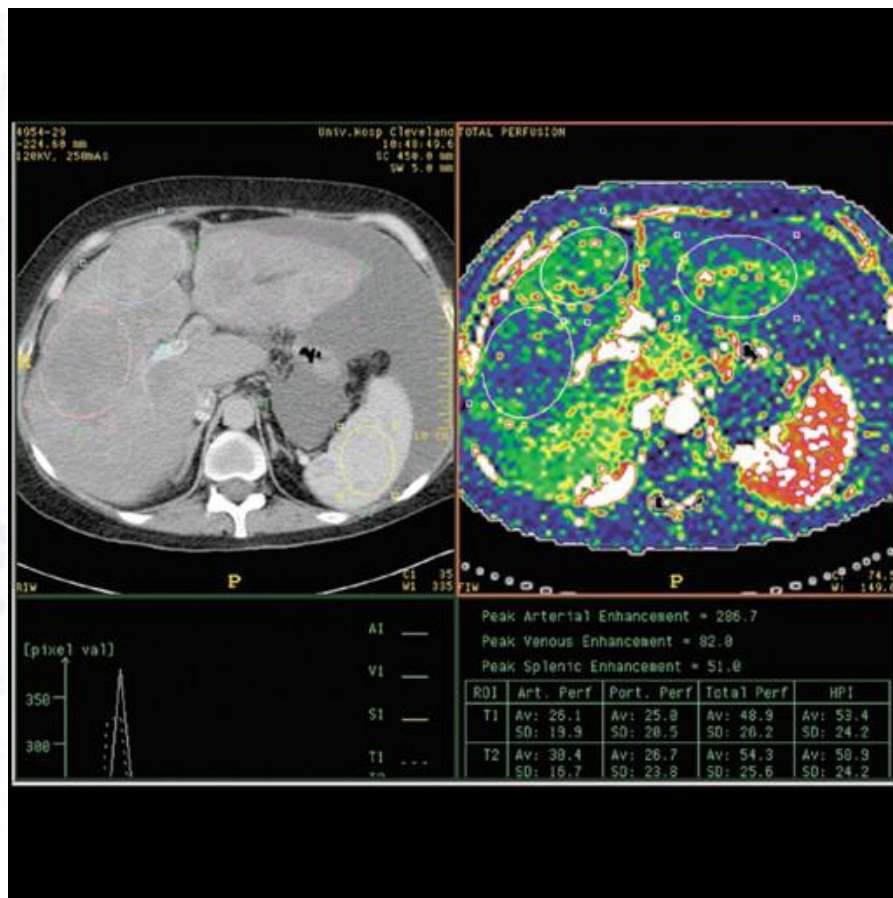
B



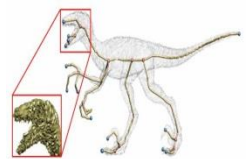
C



D



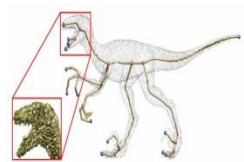




# 目标表达和描述

- 纹理特征

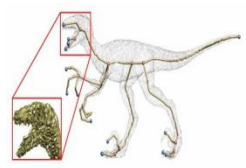




# 目标表达和描述

---

- 区域包含两方面的信息
  - 区域内部（反射特性，如灰度、颜色、纹理等）
  - 区域外部（形状）
- 区域的表达可分为内部表达法和外部表达法
  - 对这些区域表达和描述
- 什么样的描述符是一个好的描述符
  - 对尺度、平移、旋转等不敏感

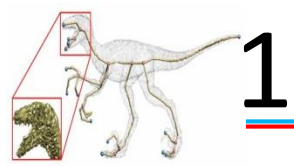


# 目标表达和描述

---

- 1 基于区域的表达
- 2 基于区域的描述

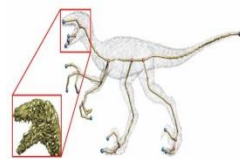




# 1 基于区域的表达

---

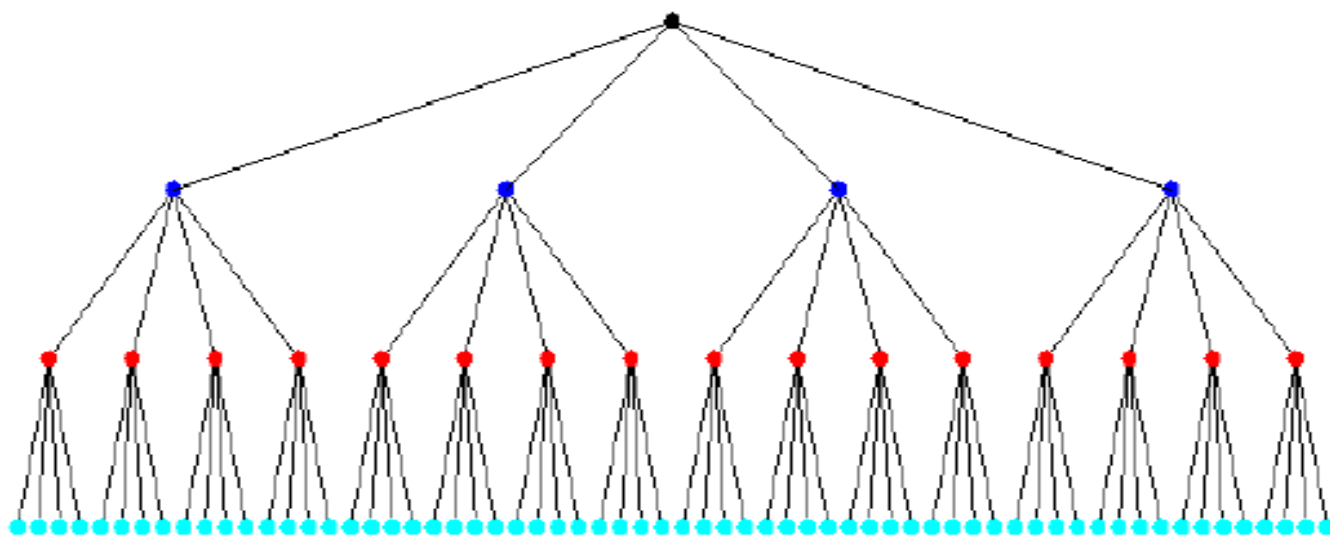
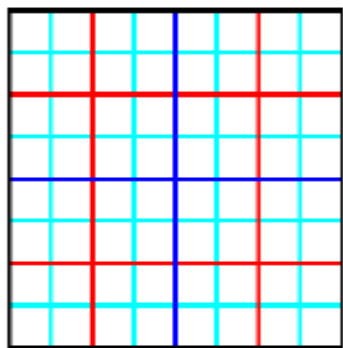
- 7.2.1 四叉树 (Quadtrees)
- 7.2.3 骨架 (Skeleton, thinning)

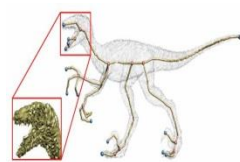


## 7.2.1 四叉树

- 四叉树结构

四层完全四叉树结构示意图

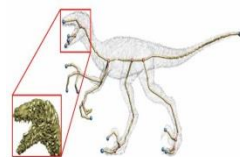




## 7.2.1 四叉树

---

- 四叉树的构建
  - 隔行隔列抽样，把图像分为四部分
  - 相邻的四像素平均实现降采样（四像素与均值的差可表征降采样后节点内部的一致性）
  - 对降采样后的图像重复上述过程直到只有一个点，作为四叉树的顶点



## 7.2.1 四叉树





- 四叉树表达法利用金字塔式的数据结构对图像进行表达。结点可分成3类：

- ①目标结点（用白色表示）
- ②背景结点（用深色表示）
- ③混合结点（用浅色表示）

白   
灰   
黑 

 E

0级

 A   
 C 

1级

		1	2
A		3	4
		5	6
C		7	8

2级

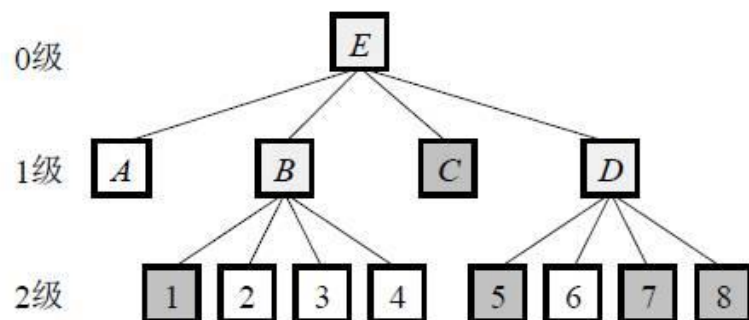
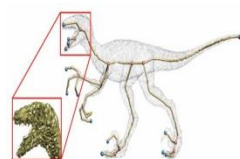


图 7.2.1 四叉树表达图示



## 7.2.1 四叉树

- 四叉树表达法

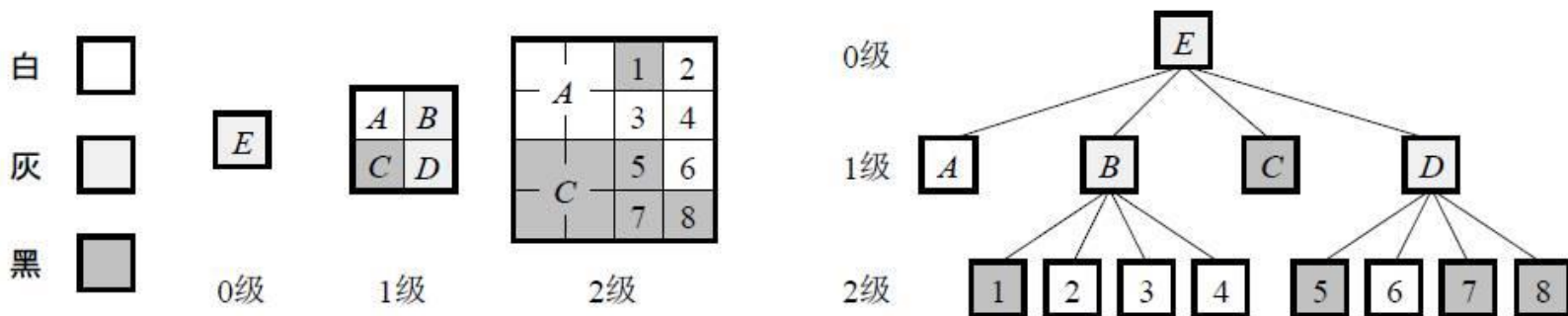
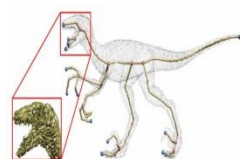


图 7.2.1 四叉树表达图示

- 当图像是方形，且像素点个数是2的整数次幂时，四叉树最适用
- n级的四叉树，结点最多：
$$N = \sum_{k=0}^n 4^k = \frac{4^{n+1} - 1}{3} \approx \frac{4}{3} 4^n$$
 含义?



## 7.2.1 四叉树

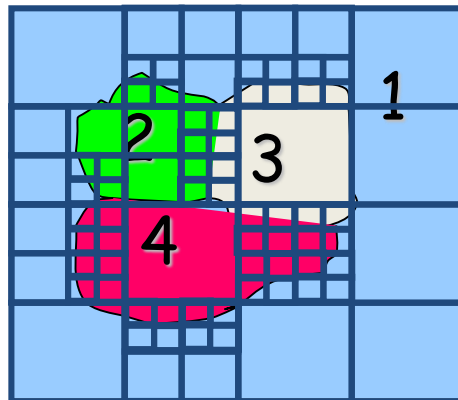
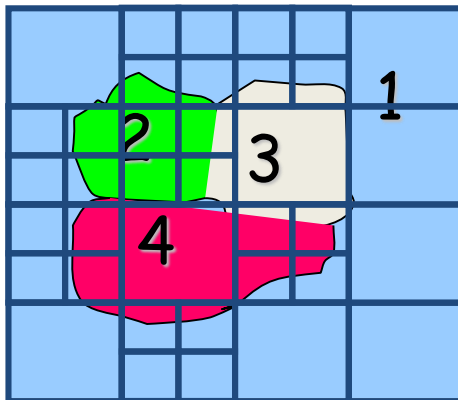
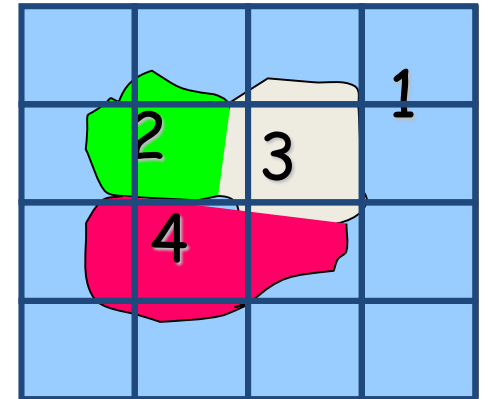
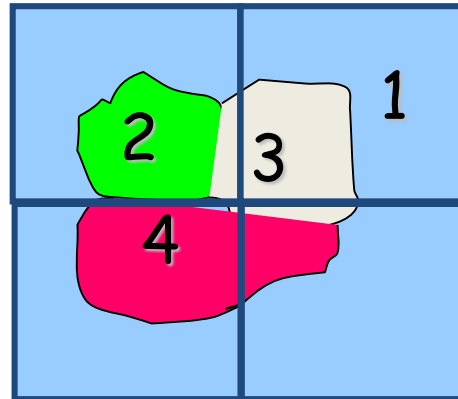
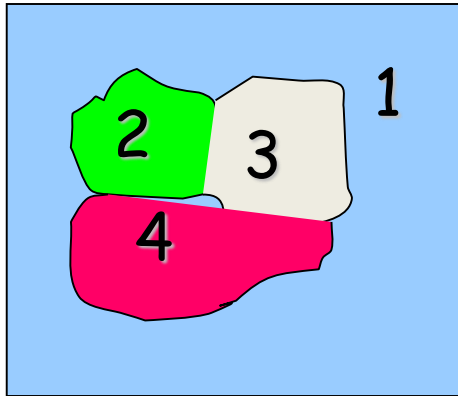
- 四叉树表达法

含义?

- n级的四叉树，结点最多：
$$N = \sum_{k=0}^n 4^k = \frac{4^{n+1} - 1}{3} \approx \frac{4}{3} 4^n$$
- 结点最多  $\rightarrow$  完整的四叉树
- $2^n * 2^n$  图像，完全四叉树有？级
- n级结点个数  $\frac{4}{3} 4^n$ ，而像素个数为  $4^n$
- 完整的四叉树结点个数比图像像素个数多，但是实际情况中有连续的目标（背景）区域



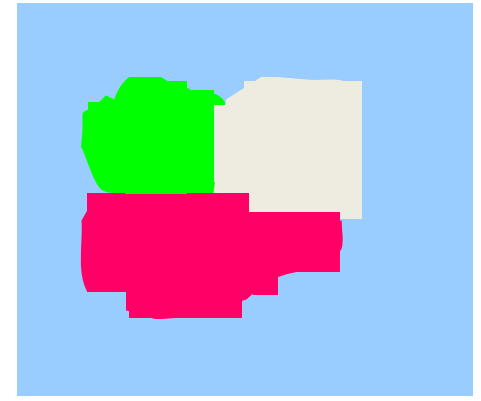
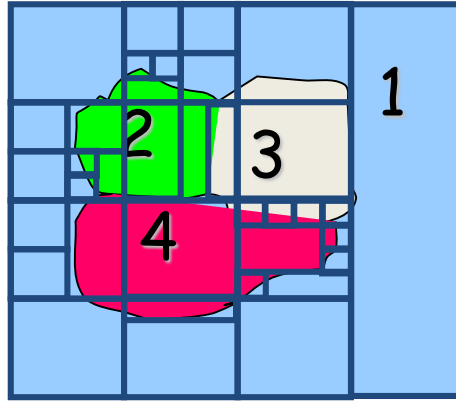
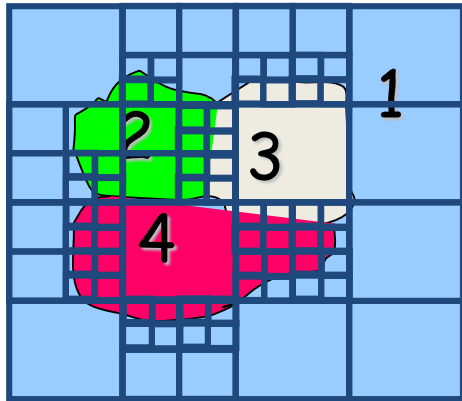
# 四叉树



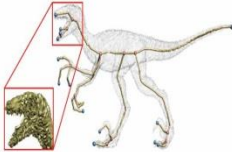
...

Splitting...

# 四叉树



Merging...



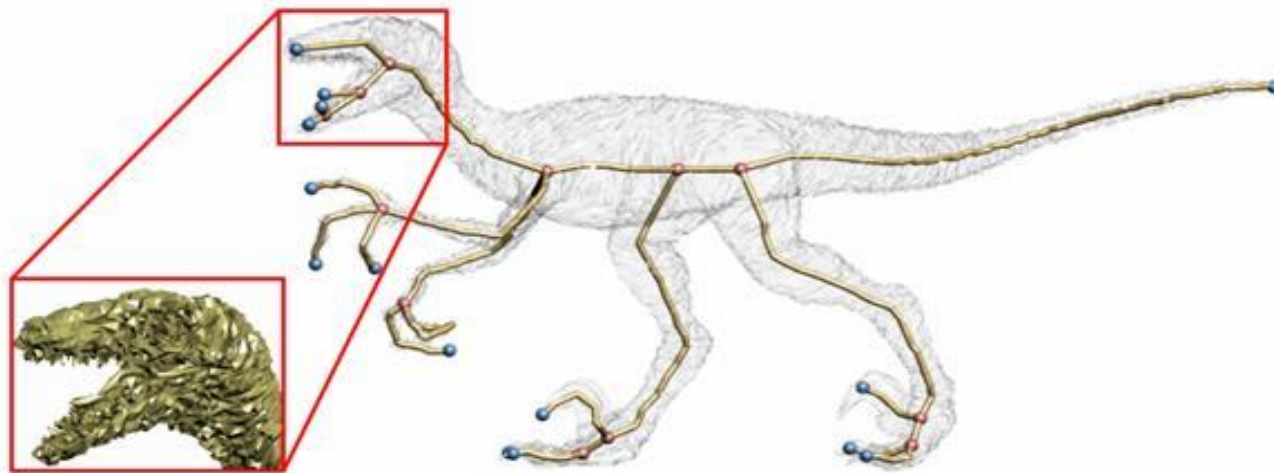
## 7.2.1 四叉树

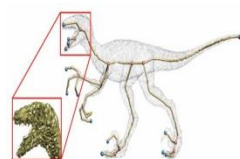
---

- 四叉树特点
  - 四叉树的优点：容易生成，方便计算区域特征信息“由粗到精”表达
  - 四叉树的缺点：一旦结点在树中的级确定后，分辨率就不能进一步提高

## 7.2.3 骨架

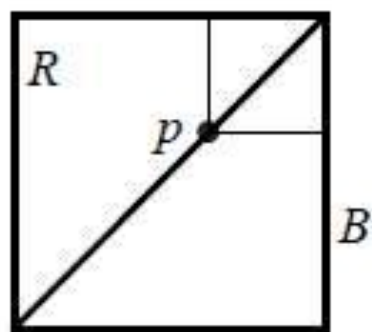
- 骨架是对目标区域形状结果的一种表达
- 对区域目标的一种抽象
- 区域 → 图形





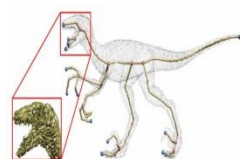
## 7.2.3 骨架

- 骨架化（细化）：对目标区域骨架抽取
  - 对每个 $R$ 中的点 $P$ ，可在 $B$ 中搜寻与它距离最小的点。如果对 $P$ 能找到多于一个这样的点（即有两个或以上的 $B$ 中的点与 $P$ 同时距离最小），就可认为 $P$ 属于 $R$ 的骨架，或者说 $P$ 是一个骨架点



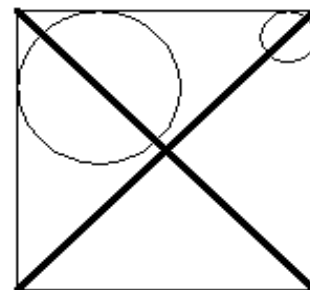
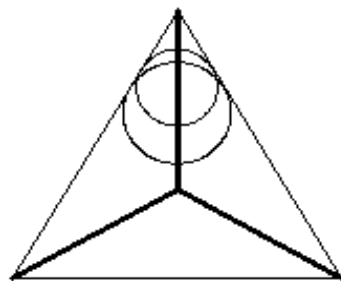
$$d_s(p, B) = \inf\{d(p, z) \mid z \in B\}$$

图 7.2.3 区域  $R$ ，边界  $B$  和骨架点  $P$

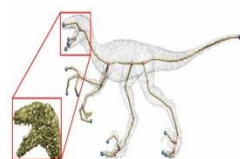


## 7.2.3 骨架

- 骨架直观理解
  - 所谓骨架，可以理解为图像的中轴，一个长方形的骨架，是它的长方向上的中轴线，圆的骨架是它的圆心，直线的骨架是它自身，孤立点的骨架也是自身。
- 两种形象化的骨架获取方法：
  - 基于烈火模拟
  - 基于最大圆盘







## 7.2.3 骨架

- 每个骨架点都保持了其与边界点距离最小的性质，所以如果用以每个骨架点为中心的圆的集合（利用合适的量度），就可恢复出原始的区域

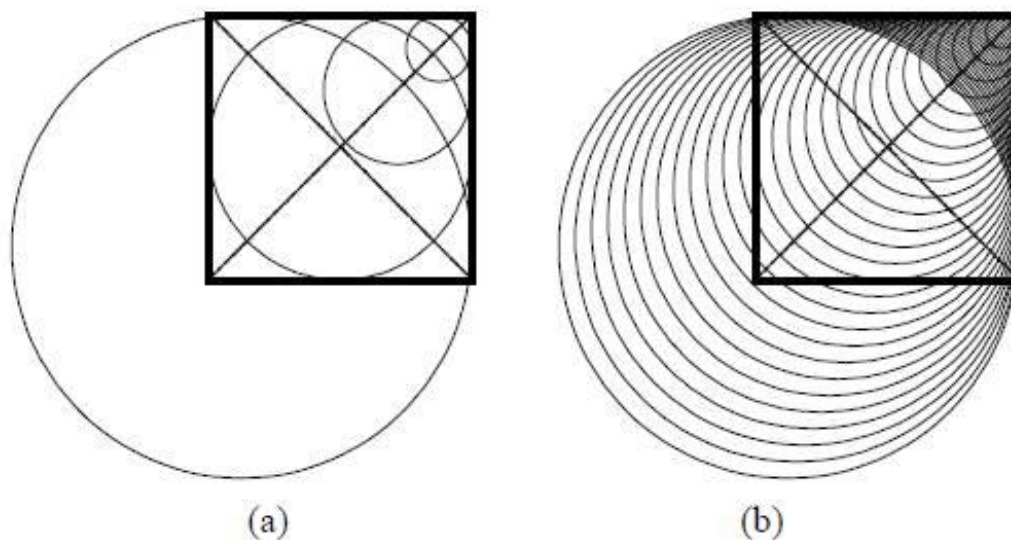
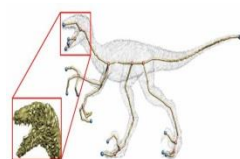
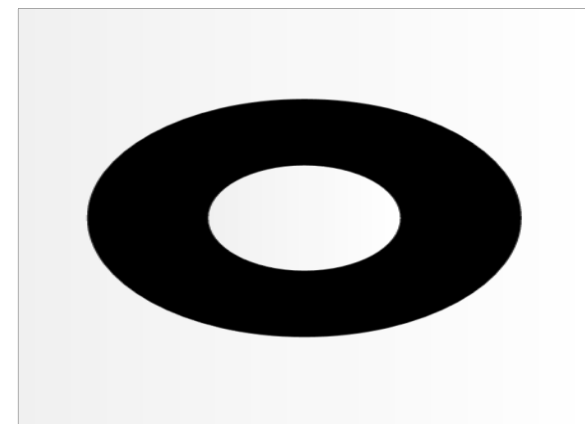


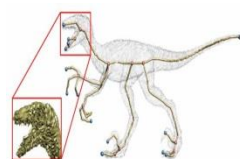
图 7.2.5 用圆的并集重建区域



## 7.2.3 骨架

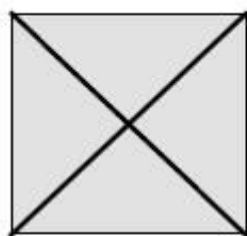
- 骨架的特点：
- 如果 $S$ 是区域 $R$ 的骨架，则满足（实际中不一定）
  - $S$ 完全包含在 $R$ 中， $S$ 处于 $R$ 的中心位置
  - $S$ 为单个像素宽
  - $S$ 与 $R$ 有相同的联通组元
  - $S$ 的补与 $R$ 的补有相同的联通组元
  - 可以根据 $S$ 重建 $R$



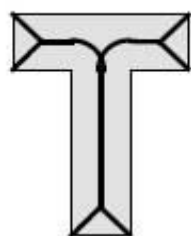


## 7.2.3 骨架

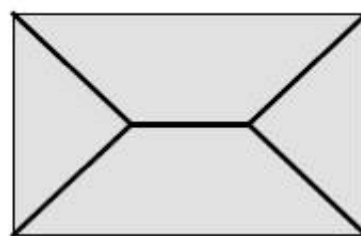
- 骨架化（细化）



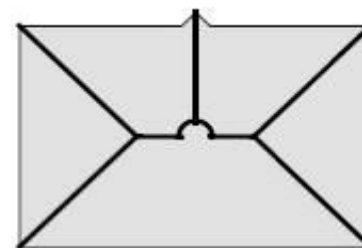
(a)



(b)



(c)

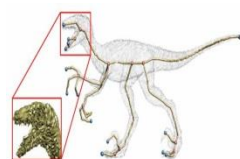


(d)

图 7.2.4 用欧氏距离算出的一些骨架的示例

- 存在问题：计算量大  
因为需要逐点扫描

$$d_s(p, B) = \inf\{d(p, z) \mid z \in B\}$$



## 7.2.3 骨架

---

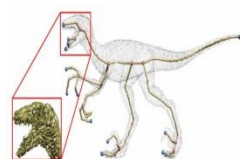
- 改进的骨架提取方法
- 思想：在保证产生正确的骨架的同时,改进算法的效率。比较典型的是一类细化算法，它们不断删去边缘，但保证删除满足：
  - 不移去线段端点
  - 不破坏连通性
  - 不引起区域的过度腐蚀



## 7.2.3 骨架

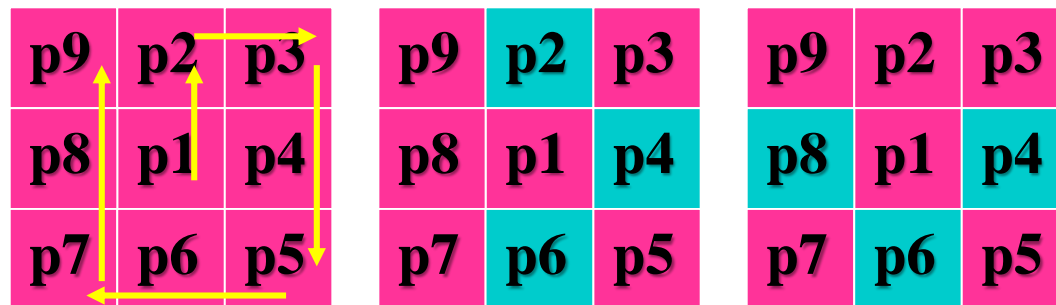
---

- 实用二值图像骨架计算方法
  - 假设区域内的点值为1，背景值为0
  - 这个方法由对给定区域的边界点连续进行两个基本操作构成
  - 这里边界点是指任何值为1且至少有一个8邻域上的点为0的像素



## 7.2.3 骨架

### 基本操作1



同时满足以下四个条件的边界点标记，准备删除

(a)  $2 \leq N(p_1) \leq 6$  其中  $N(p_1)$  是点  $p_1$  的邻域中 1 的个数，

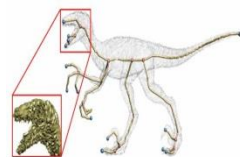
即：  $N(p_1) = p_2 + p_3 + \dots + p_9$

(b)  $S(p_1) = 1$  其中  $S(p_1)$  是按  $p_2, p_3, \dots, p_9, p_2$  顺序，  
0-1 转换 的个数

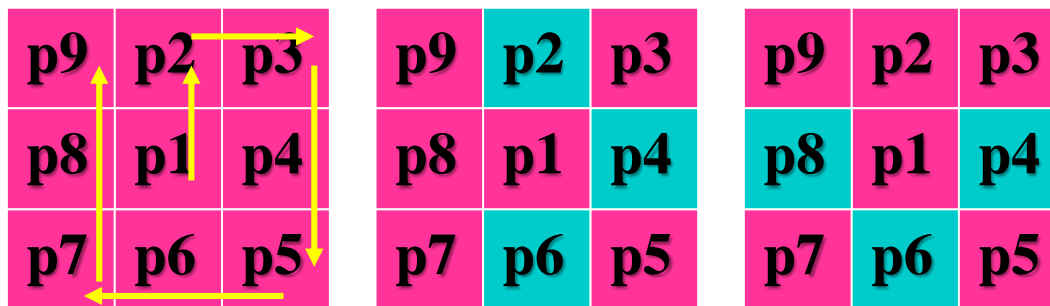
(c)  $p_2 * p_4 * p_6 = 0$  ( $p_2$ 、 $p_4$ 、 $p_6$  至少有一个 0)

(d)  $p_4 * p_6 * p_8 = 0$  ( $p_4$ 、 $p_6$ 、 $p_8$  至少有一个 0)





## 7.2.3 骨架



基本操作1

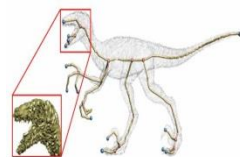
(a)  $2 \leq N(p_1) \leq 6$  其中  $N(p_1)$  是点  $p_1$  的邻域中 1 的个数,

即:  $N(p_1) = p_2 + p_3 + \dots + p_9$

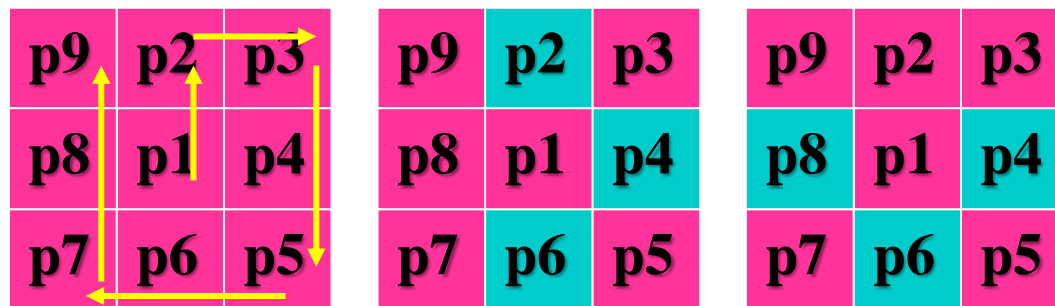
意义:

◆ 线段端点 ( $N(p_1) = 1$ )、深入内部点 ( $N(p_1) = 7$ )

不能删



## 7.2.3 骨架



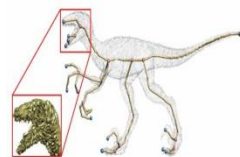
基本操作1

(b)  $S(p_1) = 1$  其中  $S(p_1)$  是按  $p_2, p_3, \dots, p_9, p_2$  顺序,  
0-1转换的个数

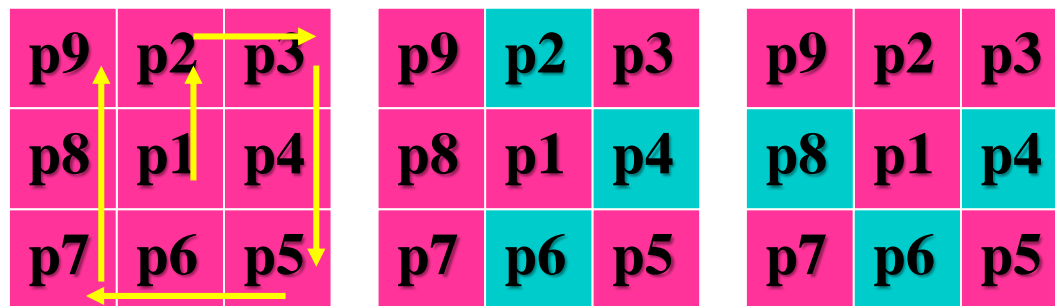
0	0	0
1	$p$	1
0	0	0

意义:

- ◆ 单像素边缘不能删 (影响连通性)
- 分析所有情况 (T, Y, L, 等)



## 7.2.3 骨架



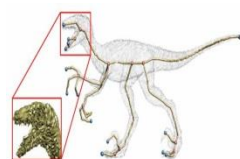
基本操作1

(c)  $p_2 * p_4 * p_6 = 0$  ( $p_2$ 、 $p_4$ 、 $p_6$  至少有一个0)

(d)  $p_4 * p_6 * p_8 = 0$  ( $p_4$ 、 $p_6$ 、 $p_8$  至少有一个0)

意义:

◆为迭代考虑: 左、上 边缘不删



## 7.2.3 骨架

p9	p2	p3
p8	p1	p4
p7	p6	p5

### 基本操作2

满足以下四个条件的边界点标记，准备删除

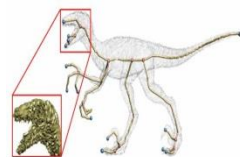
(a)  $2 \leq N(p_1) \leq 6$  其中 $N(p_1)$ 是点 $p_1$ 的邻域中1的个数，

即：  $N(p_1) = p_2 + p_3 + \dots + p_9$

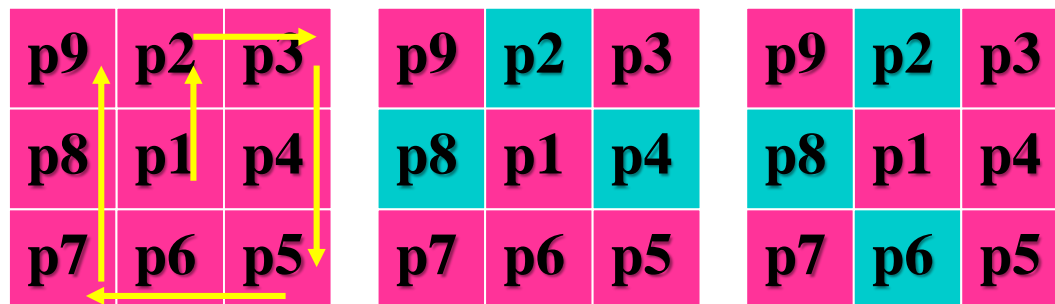
(b)  $S(p_1) = 1$  其中 $S(p_1)$ 是按 $p_2, p_3, \dots, p_9$ 顺序，0-1转换的个数

(c)  $p_2 * p_4 * p_8 = 0$  ( $p_2$ 、 $p_4$ 、 $p_8$ 至少有一个0)

(d)  $p_2 * p_6 * p_8 = 0$  ( $p_2$ 、 $p_6$ 、 $p_8$ 至少有一个0)



## 7.2.3 骨架



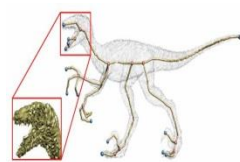
基本操作2

(c)  $p_2 * p_4 * p_8 = 0$  ( $p_2$ 、 $p_4$ 、 $p_8$  至少有一个0)

(d)  $p_2 * p_6 * p_8 = 0$  ( $p_2$ 、 $p_6$ 、 $p_8$  至少有一个0)

意义:

◆为迭代考虑: 下、右 边缘不删

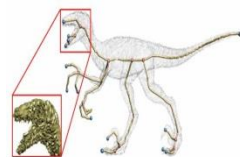


## 7.2.3 骨架

---

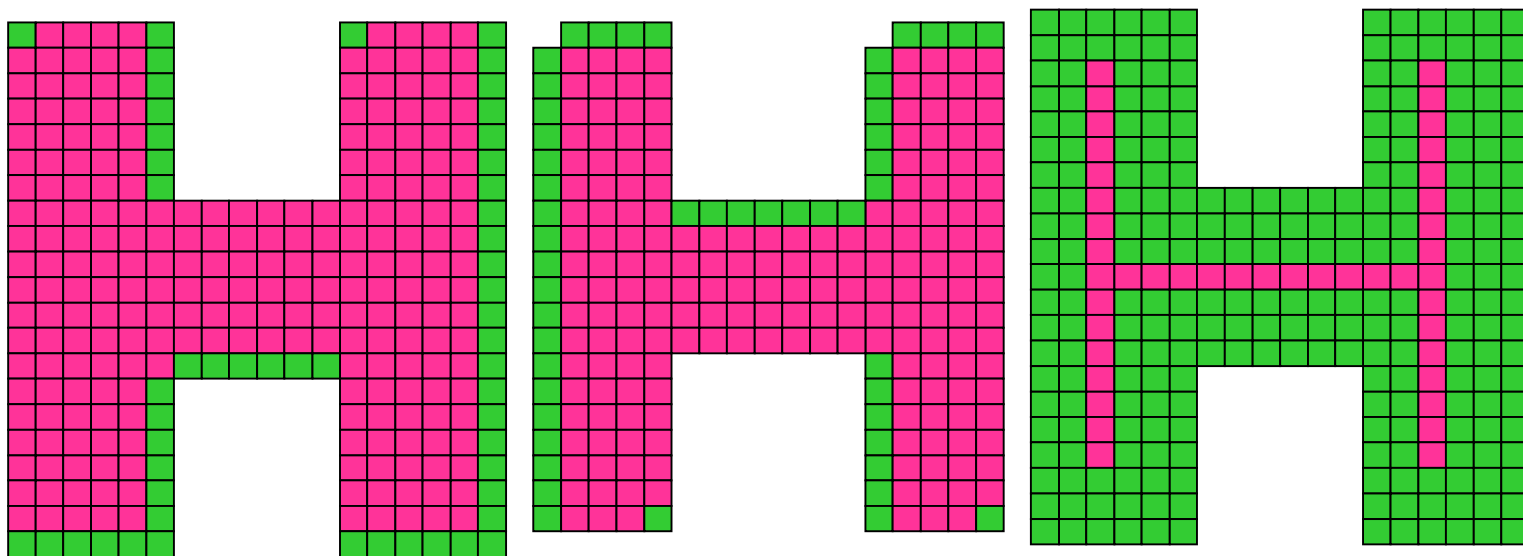
- 实用二值图像骨架计算方法
- 细化算法的一轮操作包括：
  - 按操作1，给边界点标记——删除点
  - 按操作2，给边界点标记——删除点
  - 这个基本过程反复进行，直至没有点可以删除为止。此时算法终止。

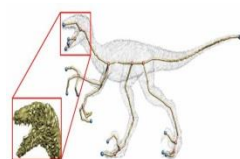




## 7.2.3 骨架

- 实用二值图像骨架计算方法





## 7.2.3 骨架

- 计算骨架的一种实用方法
  - 图7.2.7(a):  $p1$ 只有一个标记为1的8-邻域点
  - 图7.2.7(b):  $p1$ 有7个标记为1的邻点
  - 图7.2.7(c)和(d): 宽度为单个像素的线段
  - 图7.2.7(e):  $p1$ 为边界的右或下端点
  - 图7.2.7(f):  $p1$ 为边界的左或上端点

0	1	0
0	$p$	0
0	0	0

(a)

1	1	1
1	$p$	1
1	0	1

(b)

0	0	0
1	$p$	1
0	0	0

(c)

0	1	1
0	$p$	0
1	1	0

(d)

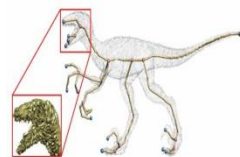
1	1	0
1	$p$	0
0	0	0

(e)

0	0	0
0	$p$	1
0	1	1

(f)

图 7.2.7 对各标记条件的解释示例

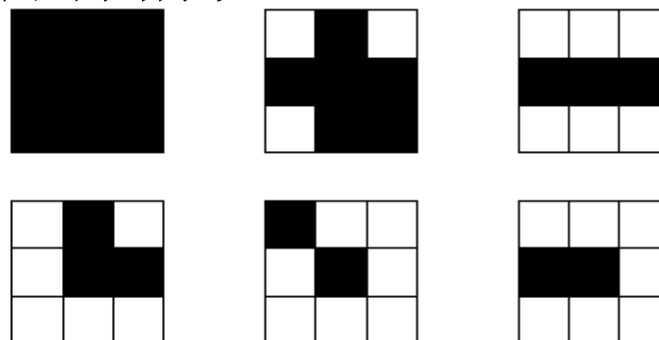


## 7.2.3 骨架

- 实用二值图像骨架计算方法

- 实现：查表法，穷举所有情况

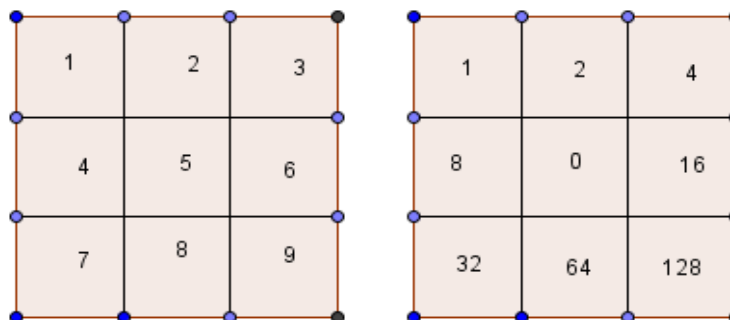
- $F(x)$   $x=0,1,..255$

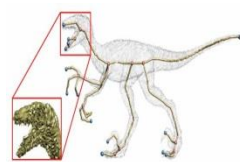


比如第三幅索引值：

（黑色为目标）

$$8+16=24$$

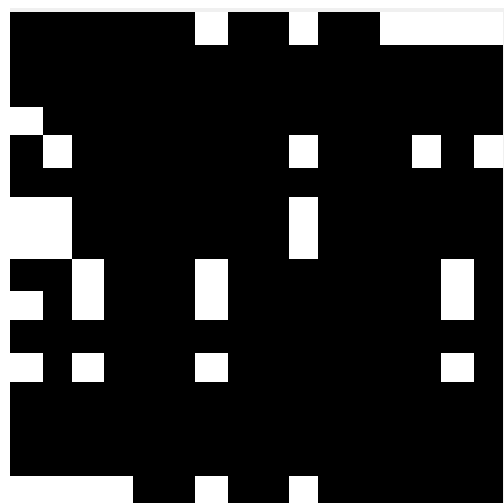




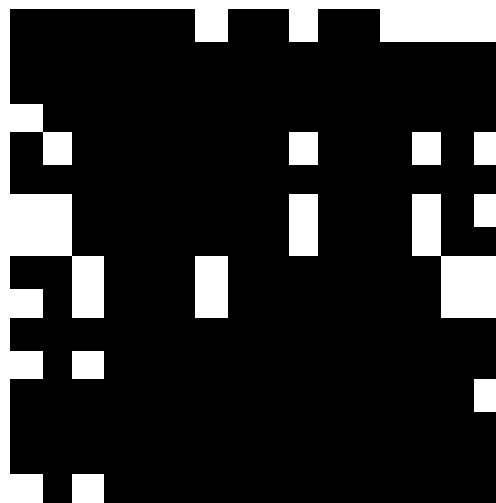
## 7.2.3 骨架

- 实用二值图像骨架计算方法
- 实现
- 查表法

p9	p2	p3
p8	p1	p4
p7	p6	p5



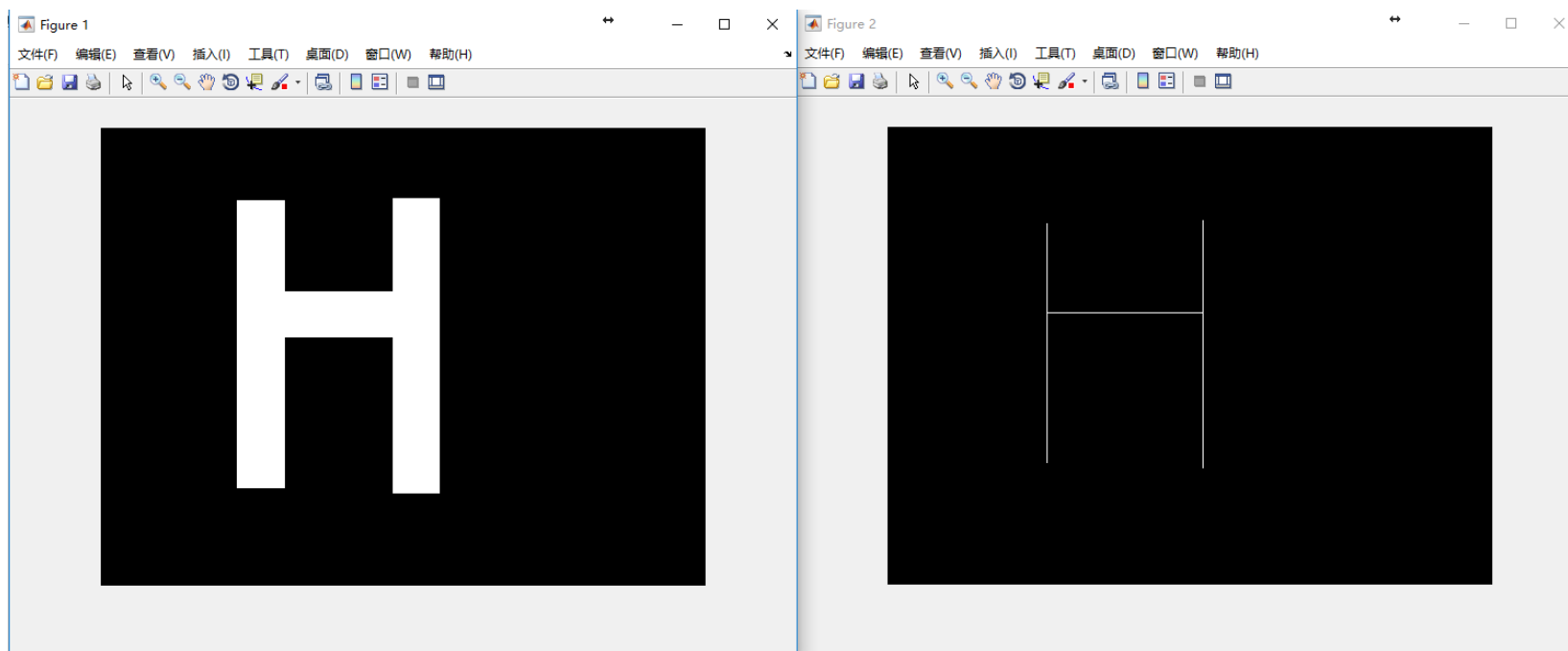
操作1索引表

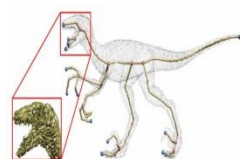


操作2索引表

## 7.2.3 骨架

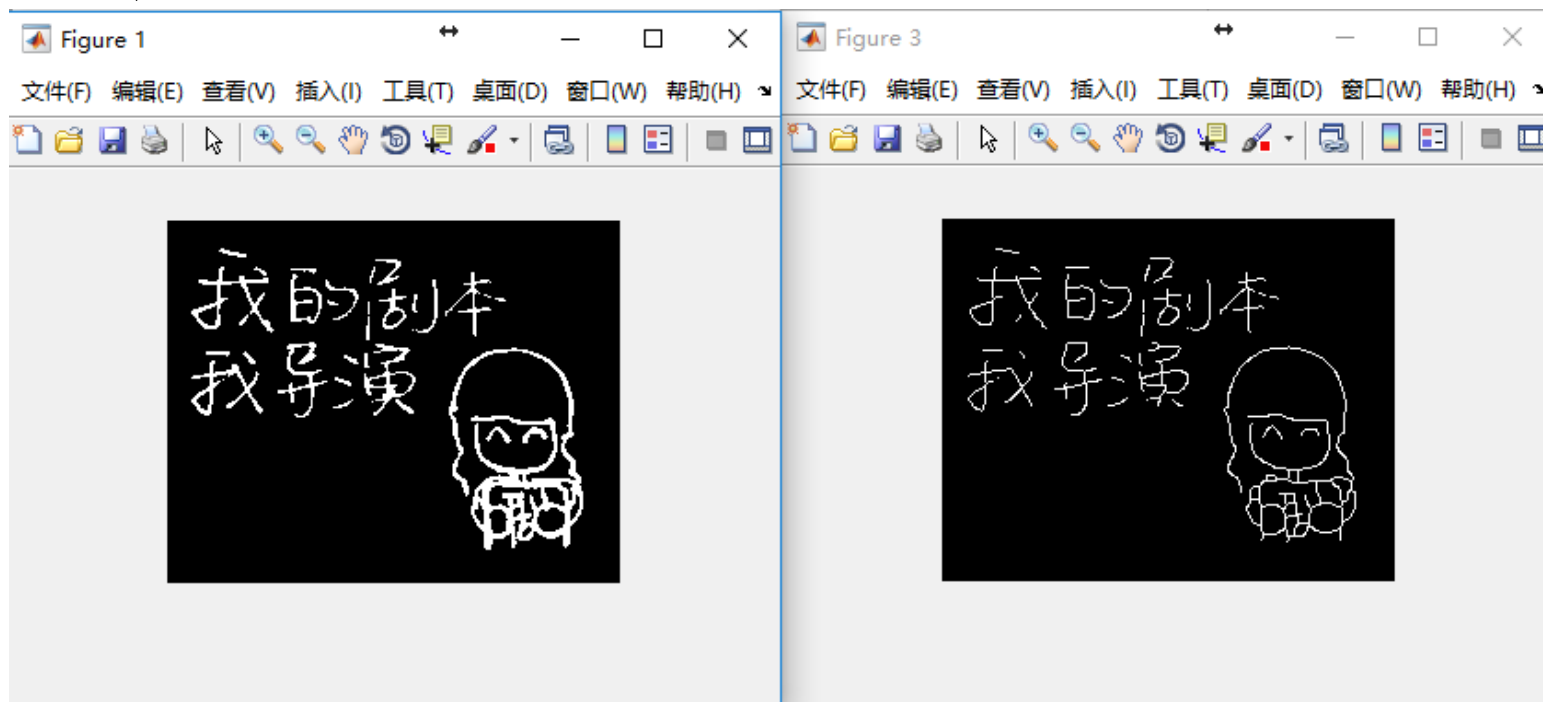
- 实用二值图像骨架计算方法
- 实现
- 查表法

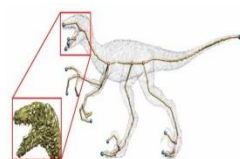




## 7.2.3 骨架

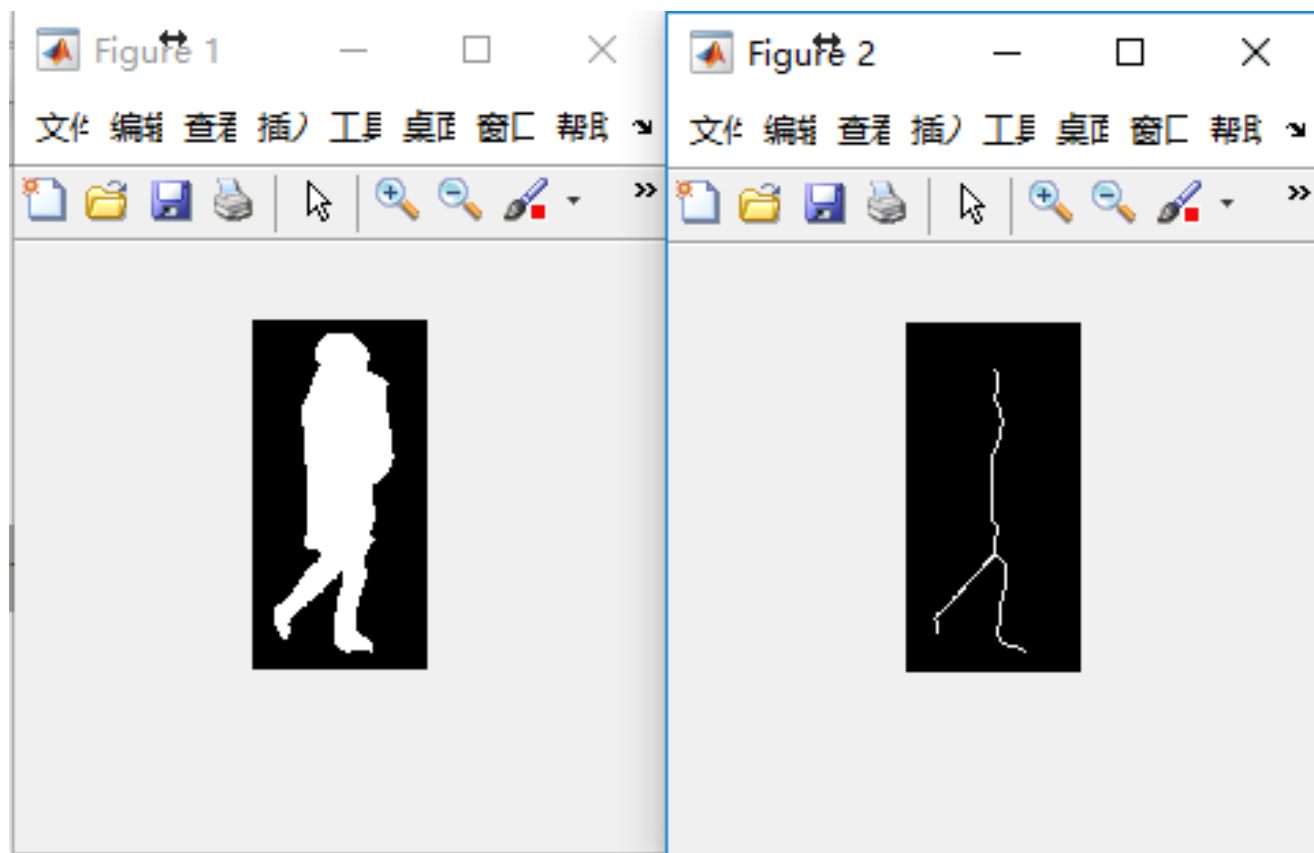
- 实用二值图像骨架计算方法
- 实现
- 查表法





## 7.2.3 骨架

- 实用二值图像骨架计算方法
- 实现
- 查表法





# 作业

---

- 7.5 (1) 讨论下图p点处骨架提取算法在基本操作第一步的情况 (2) 在第二步中的情况
  - 有兴趣可以把所有可能分析一下

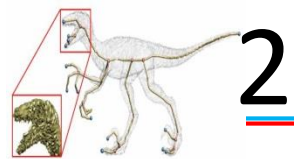
0	1	1
0	p	1
0	1	1

0	0	0
1	p	0
0	0	0

0	1	0
1	p	1
0	1	0

1	1	0
0	p	1
0	0	0

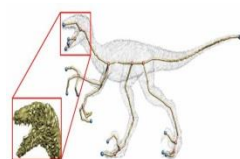




## 2 基于区域的描述

---

- 7.4.3 不变矩

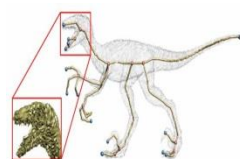


## 7.4.3 不变矩

- 当一个区域 $R$ 只是以其内部点的形式给出时，期望找到另一种区域描绘子，它对大小、旋转和平移的变化都是不变的。
- “矩”就是其中的一种
- 图像 $f(x,y)$ 的 $(p+q)$ 阶矩定义为

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y)$$

- $m_{pq}$ 唯一地被 $f(x,y)$ 所确定， $m_{pq}$ 也唯一地确定了 $f(x,y)$



## 7.4.3 不变矩

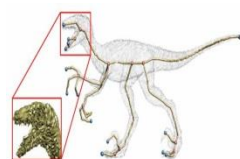
- $f(x, y)$  的  $p + q$  阶中心矩定义为

$$M_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

$\bar{x} = m_{10} / m_{00}$     $\bar{y} = m_{01} / m_{00}$  ,  $m_{00}$  为  $f(x, y)$  的质量

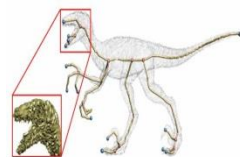
- $f(x, y)$  的归一化的中心矩可表示为

$$N_{pq} = \frac{M_{pq}}{M_{00}^\gamma} \quad \text{其中 } \gamma = \frac{p+q}{2} + 1, \quad p+q = 2, 3, \dots$$



## 7.4.3 不变矩

- 零阶矩为 $m_{00} = \sum_x \sum_y f(x, y)$ 。当 $f(x, y)$ 相当于物体的密度时则零阶矩 $m_{00}$ 是密度的总和，即物体的质量。
- 低阶矩中的一阶矩 $m_{10} = \sum_x \sum_y xf(x, y)$ 和 $m_{01} = \sum_x \sum_y yf(x, y)$ 分别除以零阶矩 $m_{00}$ 后所得的便是物体质量中心的坐标，或者直接表示的是区域灰度重心的坐标。
- 中心矩反映区域 $R$ 中的灰度相对于灰度重心是如何分布的度量。例如 $M_{20}$ 和 $M_{02}$ 分别表示 $R$ 围绕通过灰度重心的垂直和水平轴线的惯性矩。若 $M_{20} > M_{02}$ ，那么这可能是一个水平方向拉长的物体。



## 7.4.3 不变矩

示例

序号	中心矩	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
1	$M_{02}$	3	35	22	22	43	43	43	43
2	$M_{11}$	0	0	-18	18	21	-21	-21	21
3	$M_{20}$	35	3	22	22	43	43	43	43
4	$M_{12}$	0	0	0	0	-19	19	-19	19
5	$M_{21}$	0	0	0	0	19	19	-19	-19

$M_{02}$  : (1) 重心  $(3.5, 3.5)$

$$(2) \quad M_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

$$M_{02} = \sum (y - 3.5)^2$$

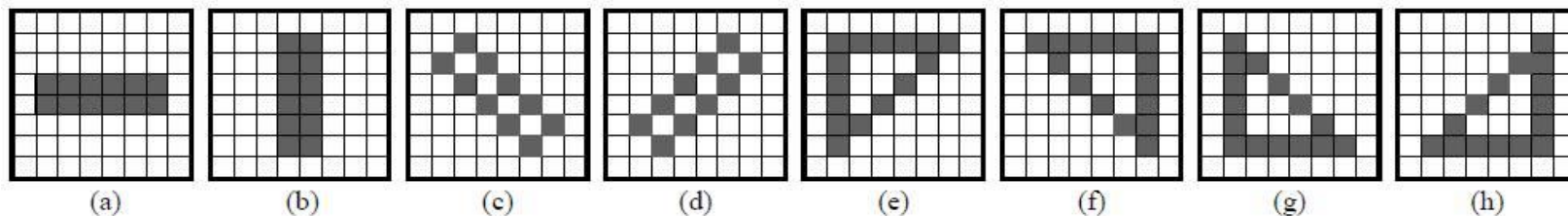
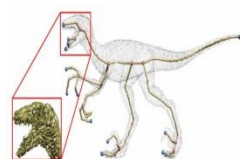


图 7.4.3 一些用于计算区域矩的示例图像



## 7.4.3 不变矩

示例

序号	中心矩	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
1	$M_{02}$	3	35	22	22	43	43	43	43
2	$M_{11}$	0	0	-18	18	21	-21	-21	21
3	$M_{20}$	35	3	22	22	43	43	43	43
4	$M_{12}$	0	0	0	0	-19	19	-19	19
5	$M_{21}$	0	0	0	0	19	19	-19	19

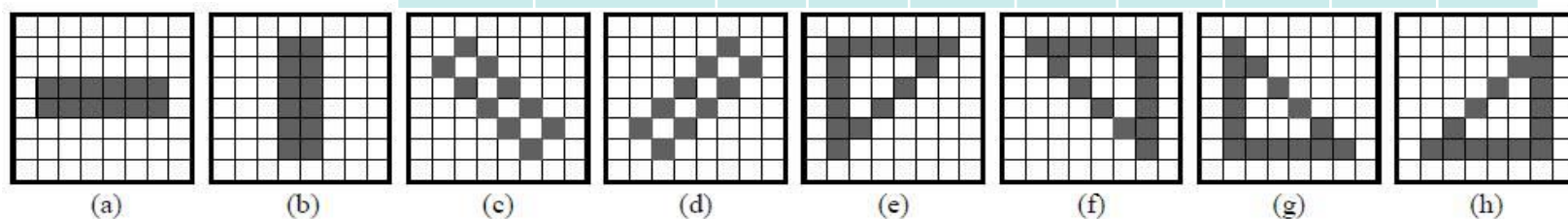
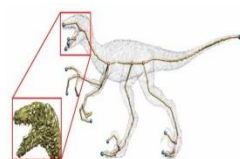


图 7.4.3 一些用于计算区域矩的示例图像

$M_{20}$ 和 $M_{02}$ 分别表示R围绕通过灰度重心的垂直和水平轴线的惯性矩。若 $M_{20} > M_{02}$ ，那么这可能是一个水平方向拉长的物体。



## 7.4.3 不变矩

- 7个对平移、旋转和尺度变换保持不变的不变矩

$$T_1 = N_{20} + N_{02}$$

$$T_2 = (N_{20} - N_{02})^2 + 4N_{11}^2$$

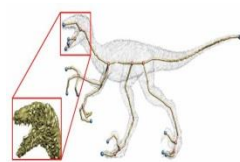
$$T_3 = (N_{30} - 3N_{12})^2 + (3N_{21} - N_{03})^2$$

$$T_4 = (N_{30} + N_{12})^2 + (N_{21} + N_{03})^2$$

$$T_5 = (N_{30} - 3N_{12})(N_{30} + N_{12})[(N_{30} + N_{12})^2 - 3(N_{21} + N_{03})^2] \\ + (3N_{21} - N_{03})(N_{21} + N_{03})[3(N_{30} + N_{12})^2 - (N_{21} + N_{03})^2]$$

$$T_6 = (N_{20} - N_{02})[(N_{30} + N_{12})^2 - (N_{21} + N_{03})^2] + 4N_{11}(N_{30} + N_{12})(N_{21} + N_{03})$$

$$T_7 = (3N_{21} - N_{03})(N_{30} + N_{12})[(N_{30} + N_{12})^2 - 3(N_{21} + N_{03})^2] \\ + (3N_{12} - N_{30})(N_{21} + N_{03})[3(N_{30} + N_{12})^2 - (N_{21} + N_{03})^2]$$

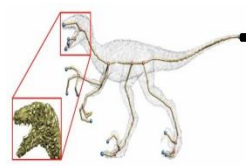


## 7.4.3 不变矩

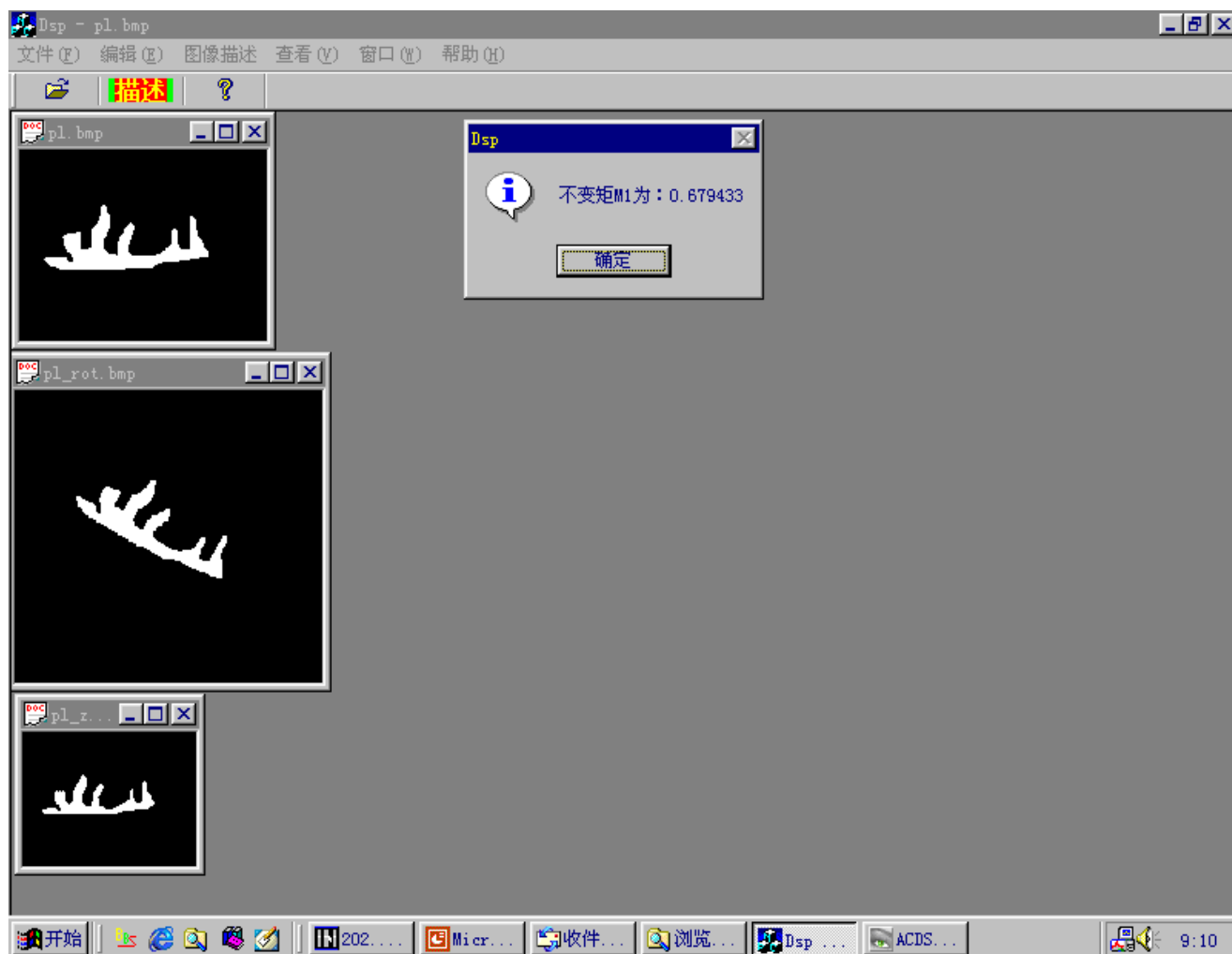
---

- 7个对平移、旋转和尺度变换保持不变的不变矩
- 物体识别过程中，只有T1和T2的不变性比较好，有学者认为只有基于二阶矩的不变矩对二维物体的描述才是真正的具有旋转、缩放和平移不变性
- T1和T2都是由二阶矩组成的



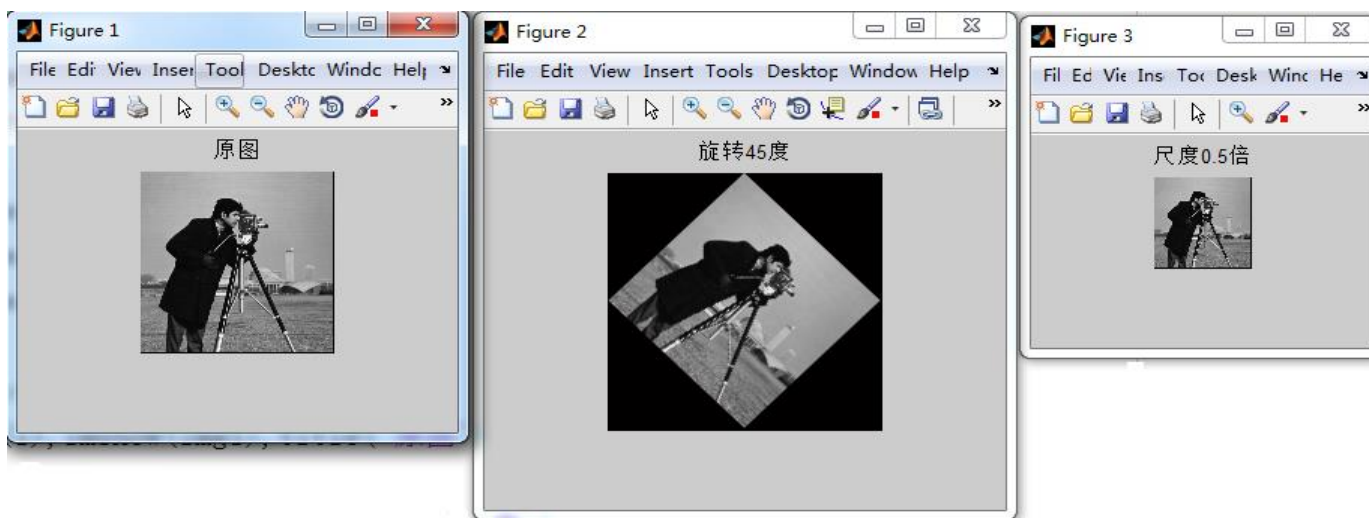


## 7.4.3 不变矩

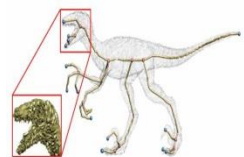


## 7.4.3 不变矩

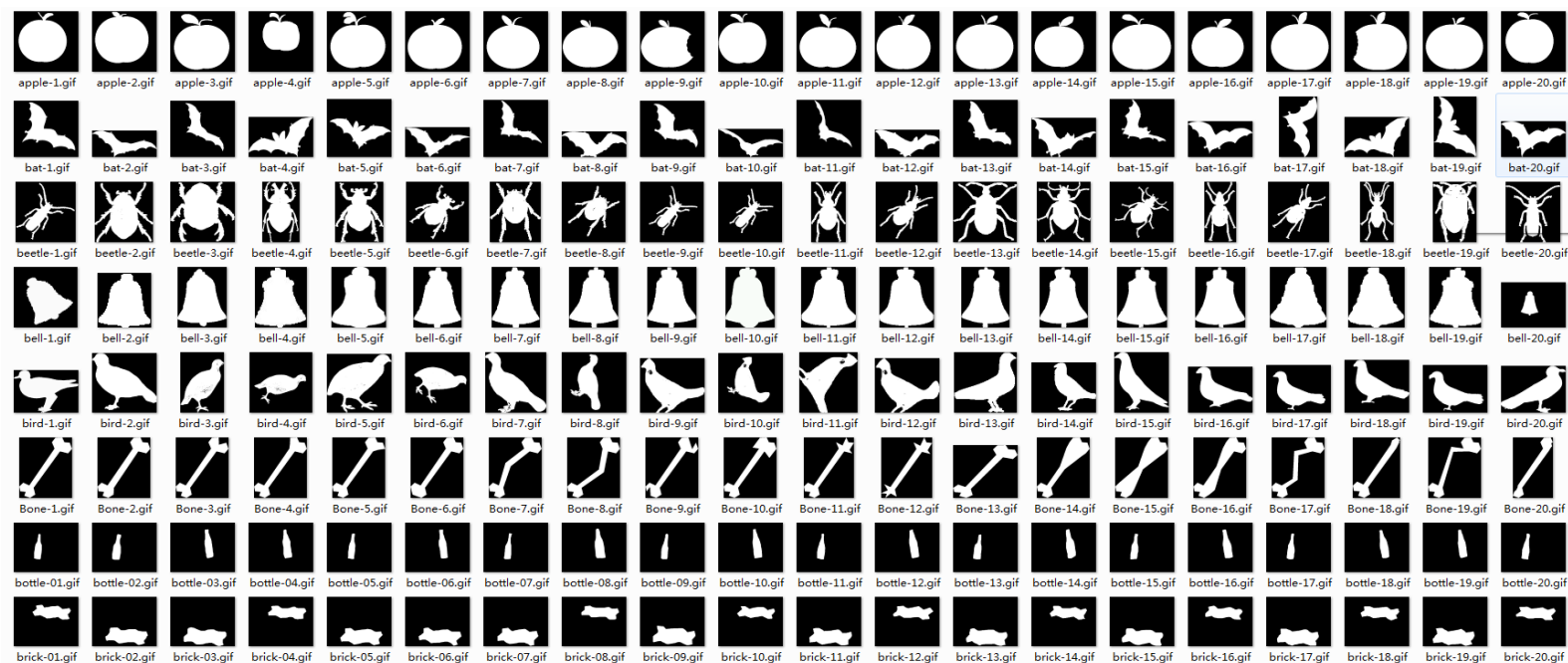
- 7个不变矩示例

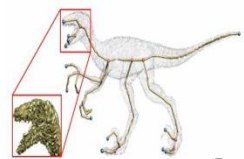


	T1	T2	T3	T4	T5	T6	T7
image1	0.001513	9.82E-09	4.49E-11	7.14E-11	-4.03E-21	-6.83E-15	-3.69E-22
image2	0.001513	9.82E-09	4.49E-11	7.14E-11	-4.03E-21	-6.83E-15	-3.69E-22
image3	0.001502	9.65E-09	4.16E-11	6.96E-11	-3.74E-21	-6.68E-15	-4.26E-23



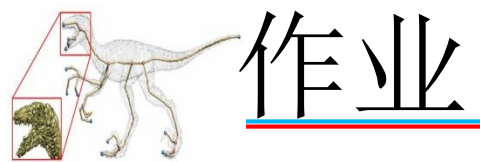
# 目前形状分析任务与应用





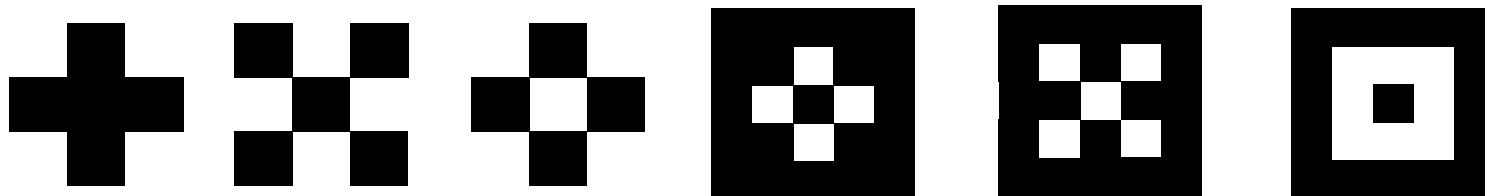
## • Matlab 函数 regionprops （度量区域属性）

- 'Area' 图像各个区域中像素总个数
- 'BoundingBox' 包含相应区域的最小矩形
- 'Centroid' 每个区域的质心（重心）
- 'MajorAxisLength' 与区域具有相同标准二阶中心矩的椭圆的长轴长度（像素意义下）
- 'MinorAxisLength' 与区域具有相同标准二阶中心矩的椭圆的短轴长度（像素意义下）
- 'Eccentricity' 与区域具有相同标准二阶中心矩的椭圆的离心率（可作为特征）
- 'Orientation' 与区域具有相同标准二阶中心矩的椭圆的长轴与x轴的交角（度）
- 'Image' 与某区域具有相同大小的逻辑矩阵
- 'FilledImage' 与某区域具有相同大小的填充逻辑矩阵
- 'FilledArea' 填充区域图像中的on像素个数
- 'ConvexHull' 包含某区域的最小凸多边形
- 'ConvexImage' 画出上述区域最小凸多边形
- 'ConvexArea' 填充区域凸多边形图像中的on像素个数
- 'EulerNumber' 几何拓扑中的一个拓扑不变量——欧拉数
- 'Extrema' 八方向区域极值点
- 'EquivDiameter' 与区域具有相同面积的圆的直径
- 'Solidity' 同时在区域和其最小凸多边形中的像素比例
- 'Extent' 同时在区域和其最小边界矩形中的像素比例
- 'PixelIdxList' 存储区域像素的索引下标
- 'PixelList' 存储上述索引对应的像素坐标
- 'Perimeter' 图像各个区域边界地区的周长



# 作业

- 6.10 计算下图目标区域的3个二阶中心矩和2个三阶中心矩



*The end!*