第二节 矢量数据的结构特征

知识点

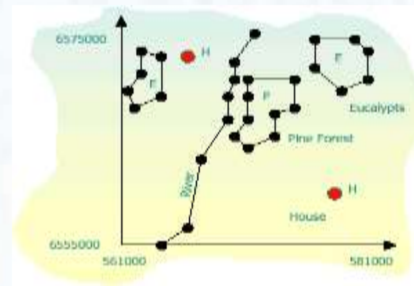
- 矢量数据的概念
- 矢量数据的编码
- 矢量数据的压缩
- 矢量数据的变换

矢量数据的结构特征

矢量数据的概念

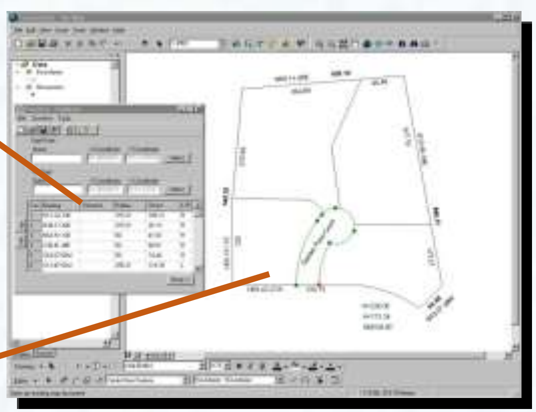
基本概念

矢量数据是指通过记录地理实体坐标的方式精确地表示点、线、面等实体的空间位置和形状。



矢量数据库

属性数据



几何数据

数据引擎

矢量数据库

OID	Shape	Attribute	…
1	XY,…	…	…

矢量数据的结构特征

矢量数据的编码

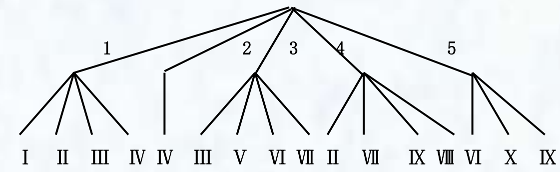
编码方法的分类

数据编码

把需要加工处理的信息，根据一定的数据结构和目标的定性特征，用特定的代码或编码字符来表示，以便于计算机识别和管理。

坐标序列法

任何点、线、面实体都可以用某一坐标体系中的坐标点坐标（x, y）来表示。



树状索引编码法

采用树状索引以减少数据冗余并间接表示邻域信息。

拓扑结构编码方法

记录数据的同时记录拓扑连接关系。

线段号	始结点	终结点	左多边形	右多边形
			A	NULL
			B	NULL
			A	B
			…	…

结点号	X坐标	Y坐标
1	X1	Y1
2	X2	Y2
…	…	…

矢量数据的结构特征

矢量数据的编码

坐标序列法

优点

- 文件**结构简单**。
- 易于实现以多边形为单位的**运算**和**显示**。

缺点

- 公共边**重复存储**，造成冗余和匹配误差。
- 缺少**邻域关系**信息。
- 不能解决**多边形嵌套**问题。

重复存储

多边形嵌套

邻域关系

特征码	位置坐标
1	x1,y1;x2,y2;x3,y3;x4,y4;x5,y5;x6,y6;x7,y7;x8,y8;x9,y9;x10,y10;x1,y1;
2	x28,y28;x29,y29;x30,y30;x31,y31;x32,y32;x33,y33;x28,y28;
3	x1,y1;x11,y11;x12,y12;x13,y13;x14,y14;x15,y15;x16,y16;x17,y17;x18,y18;x19,y19;x9,y9;x10,y10;x1,y1;
4	x18,y18;x19,y19;x9,y9;x8,y8;x7,y7;x20,y20;x21,y21;x22,y22;x23,y23;x24,y24;x18,y18;
5	x16,y16;x17,y17;x18,y18;x24,y24;x23,y23;x27,y27;x26,y26;x25,y25;x16,y16;

矢量数据的结构特征

矢量数据的编码

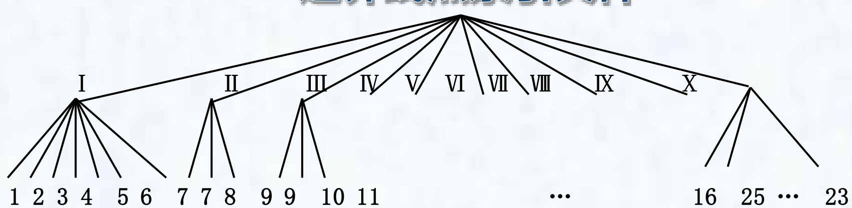
树状索引编码法

采用树状索引以减少数据冗余并间接表示邻域信息。

点号	坐标
1	x1, y1
2	x2, y2
:	:
33	x33, y33

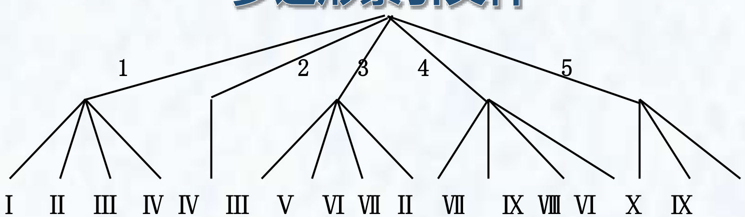
- 以坐标对顺序方式存储所用边界点

边界线点索引文件



- 以点索引与边界线号相联系

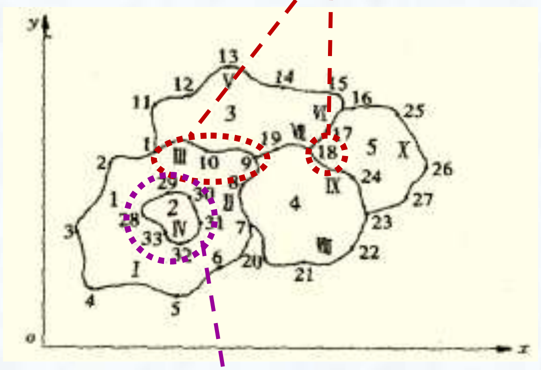
多边形索引文件



- 以线索引与各多边形相联系

- 消除了相邻多边形边界的数据冗余和不一致。
- 获取邻域信息和岛状信息较麻烦。

公共边界、公共节点的坐标只存储了一次。



通过检索索引文件获取邻域和岛状信息。

矢量数据的结构特征

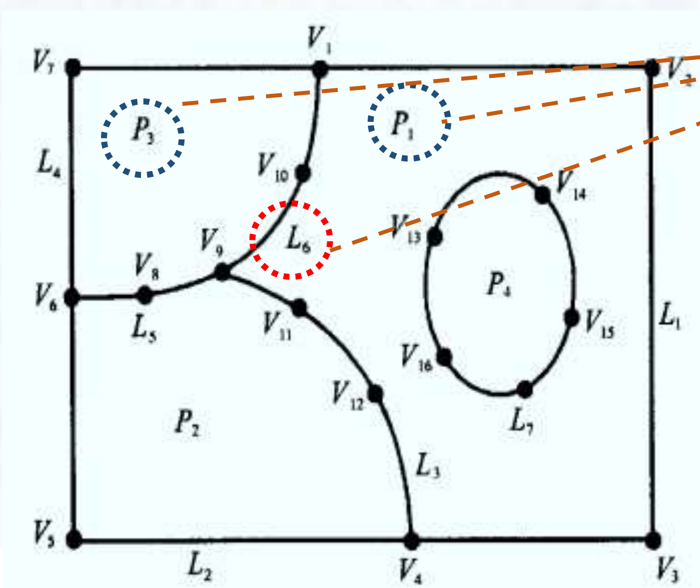
拓扑结构编码法

矢量数据的编码

拓扑关系应用于数据编码

解决**多边形嵌套**和**邻域关系**。

- 输入数据的同时**输入拓扑连接关系**。
- 从一系列**相互关联的链**中建立拓扑结构。



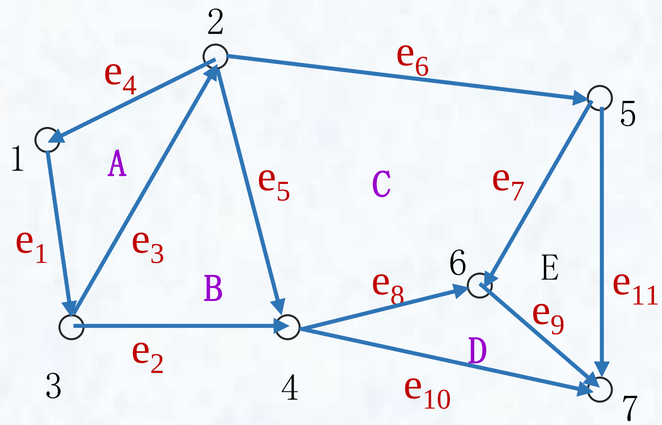
关联关系的存储

矢量数据的结构特征

双重独立地图编码系统

矢量数据的编码方法

DIME(Dual Independent) 数据文件的基本元素是由始末点定义的弧段，复杂的曲线可由多条弧段组成。每条弧段有两个指向结点的指针，和两边多边形的编码。



线段号	始结点	终结点	左多边形	右多边形
e1	1	3	A	NULL
e2	3	4	B	NULL
e3	3	2	A	B
…	…	…	…	…

结点号	X坐标	Y坐标
1	X1	Y1
2	X2	Y2
…	…	…

查找组织多边形的各条边界线的效率较低。

矢量数据的结构特征

基本概念

从所取得的矢量数据集合中抽出一个子集作为一个新的信息源，在规定的精度范围内最好地逼近原集合，而又取得尽可能大的压缩比。

矢量数据的压缩

数据量大

- 存储空间大
- 运算速度慢
- 显示效率低

数据压缩

以牺牲时间换取空间

压缩前

减少数据量，反映地物特征

压缩后

- 减少了存储空间，提高数据了数据传输效率和应用处理速度。
- 形成了不同详细程度的数据，以提供不同层次的管理、规划与决策服务。

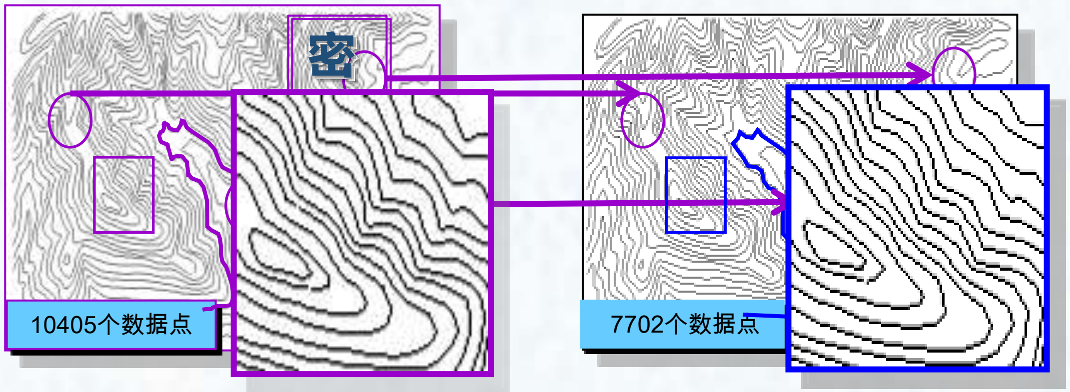
矢量数据的结构特征

矢量数据的压缩

数据压缩的基本原则

最好地逼近原集合

- 保持形状特征
- 保持密度对比
- 保持特征转折点的精度
- 保持空间关系的正确



取得尽可能大的压缩比

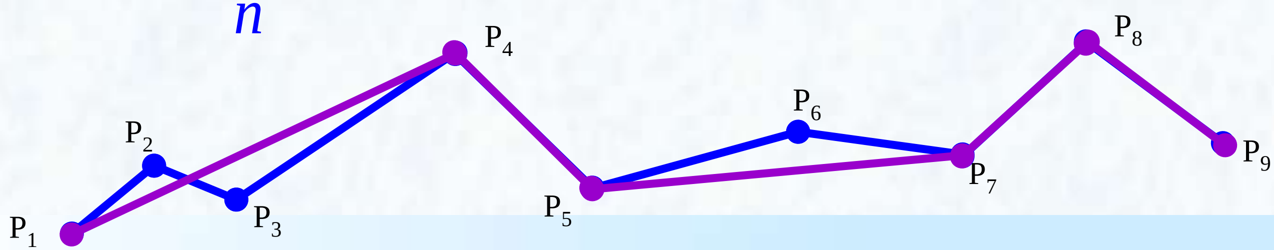
压缩比：表示信息载负量减少的程度。

- 设曲线的原点序A: $|A_1, A_2, A_3, \ldots\ldots A_m|$
- 经过压缩处理后的点序为As: $|A_{s1}, A_{s2}, A_{s3}, \ldots\ldots A_{sn}|$
- 则压缩比α为: $\alpha = \dfrac{m}{n}, \ \alpha \geq 1$

原始数据

压缩后数据

$$a = \frac{9}{6} = 1.5$$



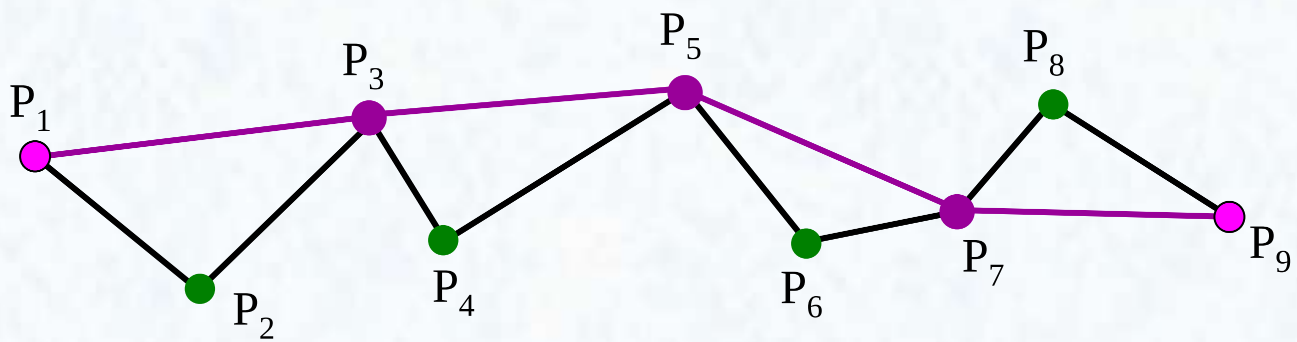
矢量数据的结构特征

矢量数据的压缩

间隔取点法

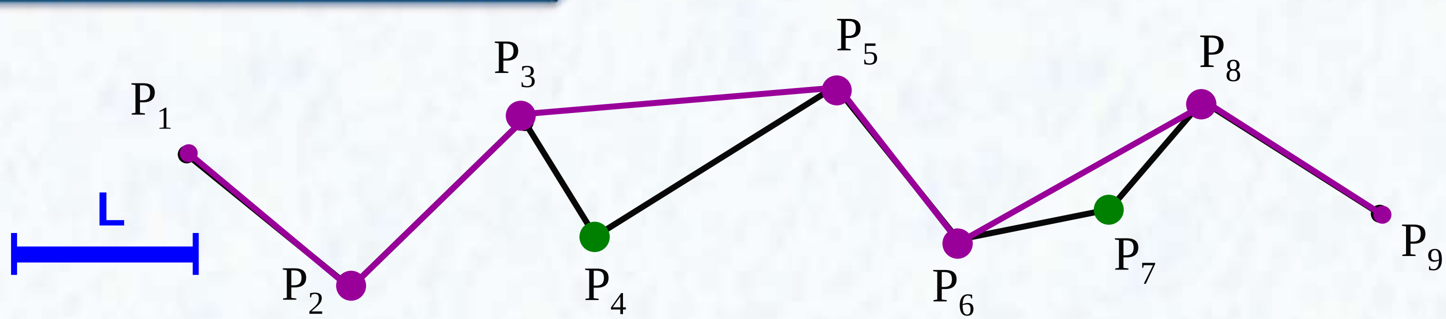
每隔n个点取一点，或每隔一规定的距离取一点，但首末点一定要保留。

间隔取点



间隔取点法-按距离取点

按距离取点



$P_1P_2>L$ $P_2P_3>L$ $P_3P_4<L$ $P_3P_5>L$ ……

没有考虑相邻点对之间的精确关系，可能会遗漏拐点和极值点。

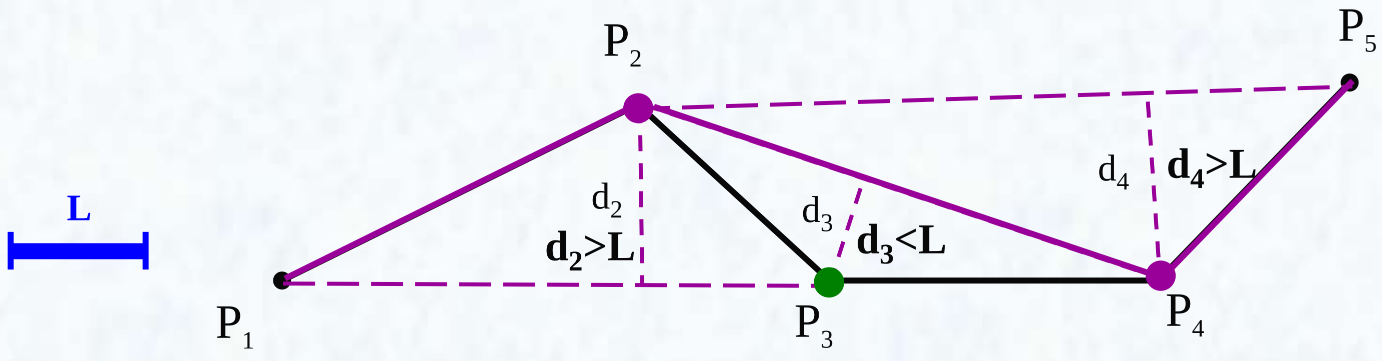
矢量数据的结构特征

矢量数据的压缩

垂距法

在给定的曲线上每次**顺序取三个点**，计算中间点与其它两点连线的**垂距d**，并与**阈值L**比较。若大于阈值，则保留该节点，否则删除。

考虑点与点之间的距离变化



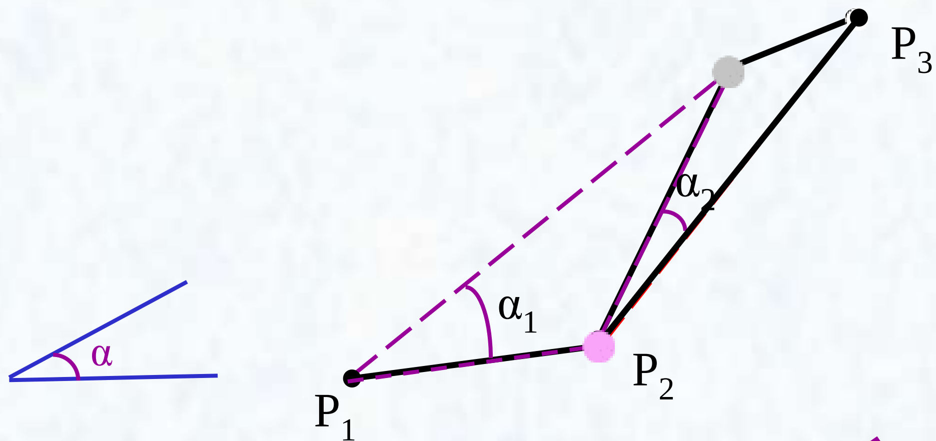
矢量数据的结构特征

矢量数据的压缩

偏角法

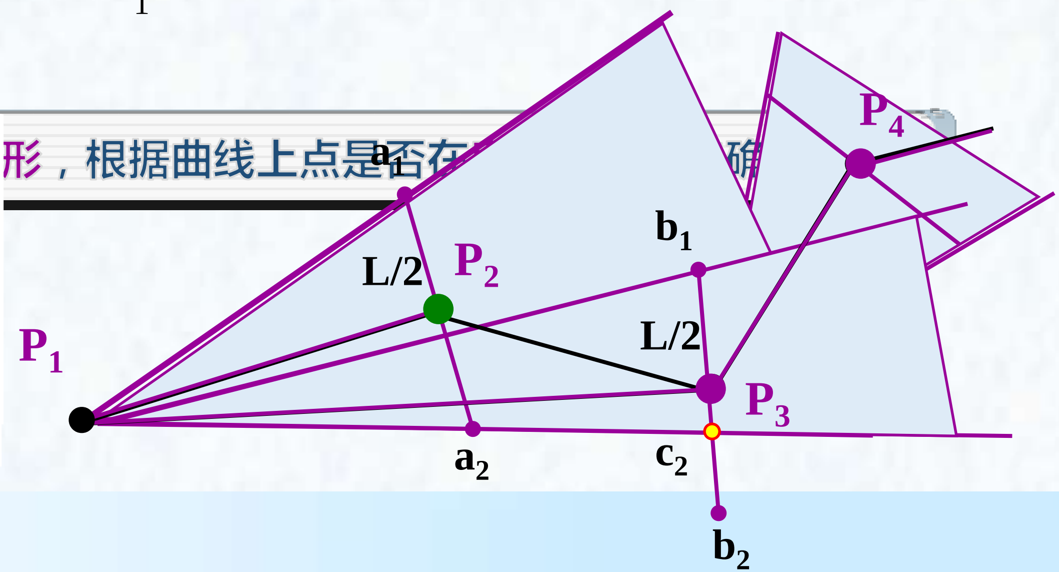
计算连接第一点和第二点，与连接第一点和第三点的矢量间的夹角。如果这个角度超过了预定的角度限差，就保留中间点，否则删除。

考虑点与点之间的角度变化



光栏法

形，根据曲线上点是否在……确



④ 当发现在扇形外的节点，如图 P_4，此时保留 P_3，并以 P_3 为新的起点，重复1—3的步骤。直到整个点列处理完，保留的节点顺序构成新的点列。

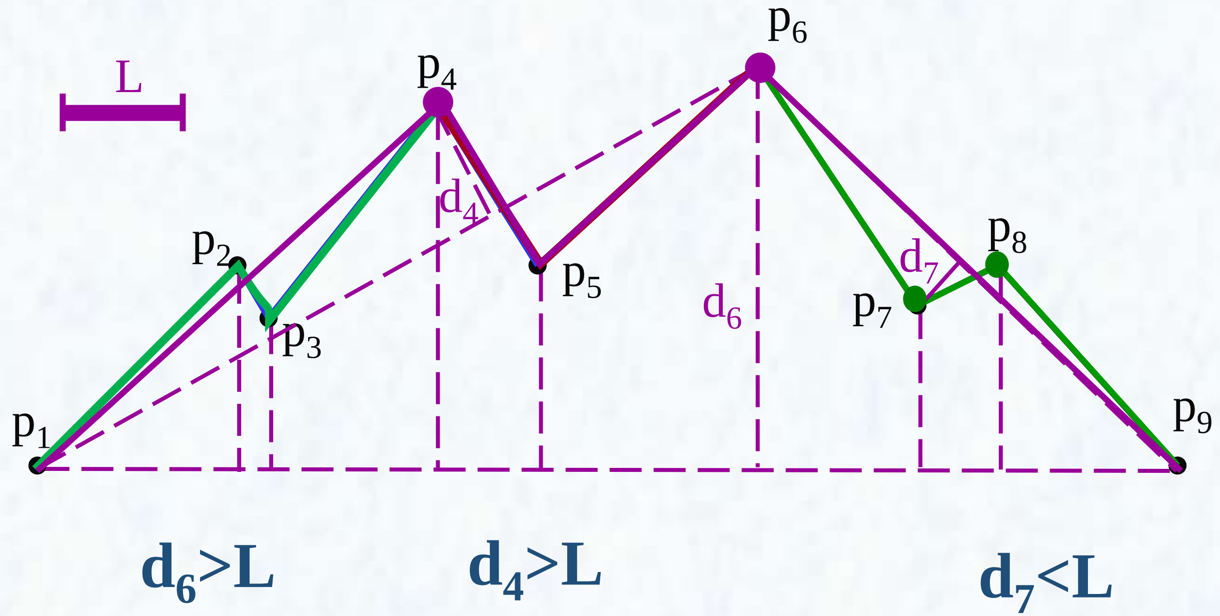
矢量数据的结构特征

矢量数据的压缩

道格拉斯-普克法

对给定曲线的首末点虚连一条直线，求中间所有点与直线间的距离，并找出最大距离dmax，用dmax与限差 L比较：

- 大则保留对应点，以该点为界将曲线分为两段，对每一段重复使用该方法。
- 小则舍去所有中间点。



矢量数据的结构特征

道格拉斯-普克法

矢量数据的压缩

从曲线整体出发的一种全局的处理方法，通过递归的算法来选择曲线上的特征点，即弯曲强(转角大)的地方。

优点

- 选择的点是原始曲线上的点，保证了精度；
- 与原曲线相比，整体位移最小；
- 与手工综合中选择的关键点基本一致；

矢量数据的结构特征

二维几何图形的变换

矢量数据的变换

点的变换可以由矩阵算子来实现，含有点的坐标矩阵(x,y)和一个一般的2*2的变换矩阵（称之为矩阵算子）相乘的结果是：

$$(x,y)\begin{pmatrix} a & b \\ c & d \end{pmatrix} = (ax+cy, bx+dy) = (x', y')$$

这个算子的意思是初始坐标x和y变换为x′ 和y′ ，这里

$$x' = ax+cy, \qquad y' = bx+dy$$

由此可见：

当a=d=1, b=c=0时，变换矩阵简化为单位矩阵：

$$(x,y)\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = (x,y) = (x', y')$$

P点的坐标没有什么变化。

矢量数据的结构特征

二维几何图形的变换

矢量数据的变换

当a>0, d=1, b=c=0时，即： $(x,y)\begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}=(ax,y)=(x',y')$

由于x ‘=ax， 这就产生了比例变换。因此，这个矩阵乘法可使初始坐标在x方向拉伸或收缩。

当a, d均大于0, b=c=0时，即： $(x,y)\begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}=(ax,dy)=(x',y')$

这个变换在x和y两个方向均有拉伸或收缩，如果a≠d，则两个拉伸或收缩不等；如果a=d>1，则p点坐标均放大；如果0<a=d<1，则p点坐标将被缩小。放大和缩小均为比例变换。

当a=d=cosθ, b=sin θ, c=-b时，即：

$$(x,y)\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}=(x\cos\theta-y\sin\theta, x\sin\theta+y\cos\theta)=(x',y')$$

即： $x'=x\cos\theta-y\sin\theta$　　$y'=x\sin\theta+y\cos\theta$

说明这个变换使点p绕坐标原点旋转了一个角度θ，θ是从x轴正向起算按反时针方向旋转取正值。

矢量数据的结构特征

二维几何图形的变换

矢量数据的变换

矢量符号的四种基本变换可用3×3矩阵的形式给出：

平移变换

$x'_{(i)} = x_{(i)} Trl(x_p, y_p)$

式中，$Trl(x_p, y_p)$是平移变换算子，其元素排列形式为：

$$Trl(x_p, y_p) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_p & y_p & 1 \end{bmatrix}$$

比例变换

$x'_{(i)} = x_{(i)} Scl(S_1, S_2)$

式中，$Scl(S_1\ S_2)$为比例变换算子，其元素排列形式为：

$$Scl(S_1, S_2) = \begin{bmatrix} S_1 & 0 & 0 \\ 0 & S_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

显然，$S_1 = S_2 \neq 1$时，为相似变换

当$S_1 = 1$，$S_2 \neq 1$时，为y方向的变换

当$S_1 \neq 1$，$S_2 = 1$时，为x方向的变换

当$S_1 \neq S_2 \neq 1$时，为x和y方向都变比例，且不等。

矢量数据的结构特征

二维几何图形的变换

矢量数据的变换

旋转变换

$x'_{(i)}=x_{(i)}Rot(\theta)$

式中, θ为旋角，它从x轴正向起算，逆时针为正，顺时针为负，Rot(θ)为旋转变换因子，其矩阵元素排列为：

$$Rot(\theta)=\begin{bmatrix}\cos\theta & \sin\theta & 0\\ -\sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}$$

错移变换

$x'_{(i)}=x_{(i)}T(m,n)$

式中， m和n方向错移，T（m, n）为错移变换算子，其矩阵元素排列形式为：

$$T(m,n)=\begin{bmatrix}1 & n & 0\\ m & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

当n=0时，向x方向错移，当m=0时，向y方向错移。

以上四种3×3矩阵变换因子可彼此相乘，进而产生多种变换的3×3矩阵变换算子，与矢量符号组成的K×3（k为离散点坐标矩阵的行数，每行的第1、2个元素为坐标值，第3个元素为1）的矩阵相乘，，可以得到变换后的新坐标。

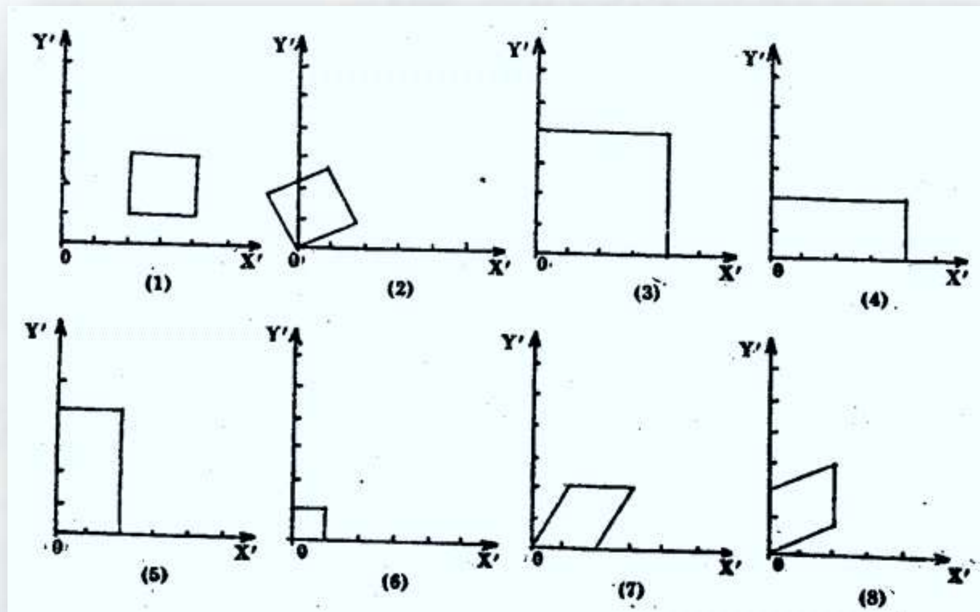
矢量数据的结构特征

二维几何图形的变换举例

矢量数据的变换

一个正方形矢量符号的顶点坐标为（0，0），（2，0），（2，2），（0,2），（0, 0）。可用5×3的坐标矢量x表示：

$$x = \begin{pmatrix} 0 & 0 & 1 \\ 2 & 0 & 1 \\ 2 & 2 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$



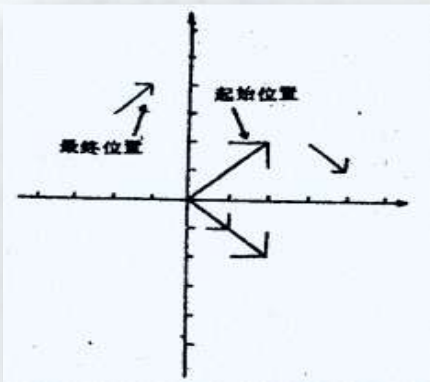
上图分别表示：Trl(2,1), Rot(30°），Scl（2,2），Scl（2，1），Scl（1，2），Scl（0.5, 0.5）T(0.5, 0), T(0, 0.5)的变换。

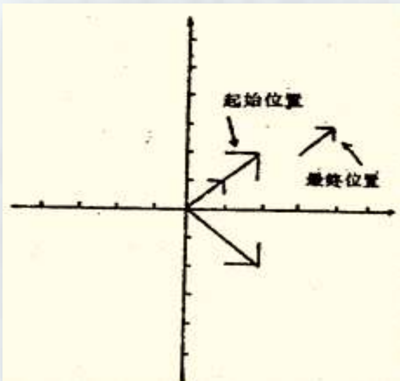
矢量数据的结构特征

二维几何图形的变换举例

矢量数据的变换

矩阵算子可以实施连续的变换，后一种变换的算子可依次加在前一次算子的右边。但要特别注意，算子排列的顺序是至关重要的。例如一个箭头符号的坐标矢量是：





方程 $x' = x.\ Rot(-90^\circ)Scl(0.5, 0.5)Trl(3,2)Rot(90^\circ)$和方程$x' = x.\ Rot(-90^\circ)Rot(90^\circ)Scl(0.5, 0.5)Trl(3,2)$所产生的结果是不同的。左图表示前一个变换的结果，右图表示后一个变换的结果。

由上例还可以发现Rot（- θ ）是Rot（ θ ）的逆运算，因为它们在一起并不产生联合的效果，即Rot（- θ ）.Rot(θ)=1; 由矩阵法则可知： $Rot^{-1}(\theta)=Rot(-\theta)$。同理，$Trl^{-1}(x_p, y_p)=Trl(-x_p, -y_p)$。但$Scl^{-1}(S_1, S_2)=Scl(\frac{1}{S_1}, \frac{1}{S_2})$, 这是因为乘和除互为逆运算。

谢 谢 大 家！