1984

国际图形会议译文集

上 册

机械工业部武汉计算机外部设备研究所

目　录

译　序

在设计任何可见目标时，只要利用了“前设计”，工作就容易多了，计算机图形的发展日趋重要，其原因是计算机图形促进了设计的利用。“前设计”表示设计结果和设计过程两个部分，一般来看，被设计目标各种各样，对工程技术人员来说，这些目标可以是机器也可以是电子线路，如在第三章CAD/CAM”中所讨论的那样。在外科手术之前，医生常根据计算的层析X射线摄形图象设计病人器官之模型，以帮助诊断，如第八章医学图象所研究的内容。第七章“计算机艺术”涉及到艺术家用计算机图形生成美妙可视图象的方法。第一章“计算几何学”提供了被设计物体形状的理论基础，这是一种常用的技术领域，在该领域中，计算机图形引起世界范围的技术进展。此册25篇文章综述反映了计算机图形在各个领域的国际性进展，给每一个潜在的或实际的图形用户提供了最新信息技术。

一般说，收集这种信息的方法有两种，一种方法是邀请国际上的作者写出他们专业领域被充分建立，能准确判断人们熟悉的东西，那么这种方法是很好的方法。但是计算机图形仍处在开发阶段，所以不便使用这种方法。我们必须依靠另一种方法，这种方法是由二个过程组成的，第一个过程，宣布对论文提出要求，按要求得到论文，其中论述新发现、新理论、新工艺以及总结经验。第二个过程是组织一个规划会，逐篇评论提出的论文，每篇论文要反复比较，抓住要点，评出结果，对于论文中不足之处，告知作者，予以修改补充。

国际计算机图形讨论会已召开了四次，1975年在美国洛杉矶召开了第一次，之后三次1983——1985年在日本东京举行。国际计算机图形研究讨论会是由日本管理协会组织，并受世界计算机图形协会日本计算机图形协会赞助，同美国计算机图形协会合作，举办的国际性研讨会，所发表的学术论文在国际上颇有影响。1985年文集涉及到许多新的领域，如第六章“计算机动画片”、第五章“图象通讯与接口”，第二章“图形标准化及软件”、第四章讨论了“图形网络”，这也是国际上新出现的领域。

我们知道，，计算机图形仍处于萌芽状态，还存在一些困难和未开发的重要领域，如图形数据库管理、图形网络、图形通讯、工程动画片和制造过程设计等，还需要在将来的研究中加以开发应用，希望本册对上述方面的研究有所借鉴。

柴贵亭

一九八六年十月

立体造型的可逆性运算

理光公司软件研究中心

Hiroshi Toriya, Toshiaki Saton, Kenji Ueda, Hiroaki chiyokura

本文介绍了一种用树形数据结构来表示实体设计过程的方法。（通过规定数据树的节点，能很快地更新前几步设计。本文还详细介绍了可逆集运算。

1、绪言

有效的交互式立体造型系统对开发实用的CAD/CAM是至关重要的。由于建立实体模型的过程是一种"试探法"的过程，用户必须不断地反复更新前阶段的设计。但常规建模系统难以做到这一点。这是由于它们采用的多数运算没有相应的逆运算。最糟的是，在试图重新产生上一步设计时，稍有疏忽，会把整个设计破坏掉。因此，我们提出了一种技术，把设计的全部过程以树形数据结构形式在系统里表示出来。采用此方法，实体设计过程中的任何阶段的设计均能很快得到更新。本文还描述了设计过程前后关系中的集运算。本文所讨论的工具和运算已DESIGN—BASE中得到实体。DESIGN—BASE是一个继承MODIF〔3〕所有特征的建模系统。该系统由C语言写成，并在Unix下运行。

2、实体设计过程的表示法

2．1 原语运算支持UNDO和REDO指令

在我们系统中，边界用来表示实体形状，用局部和集运算生成及修改实体边界表示法。生成或修改实体运算中，边界表达式往往借助原始运算进行处理的。原始运算包括欧拉运算〔1，2〕，几何运算，整体运算。几何运算修改与实体有关的几何信息，例如直边改为曲边等，实体的平移与旋转属于整体运算。在DESIGN—BASE中，有20种原始运算，每一种都有一个相应的可逆运算。

Mantyla和Snlonen〔5·6〕提出了一个实体建模系统，在这个系统中，采用欧拉运算来分步实现集合运算，而在DESIGN—BASE中，所有的高级运算都通过原始运算来实现。设计中使用的所有原始运算存储在表示设计全过程的树中，系统通过调用若干序列的存贮的原始运算，能很快地重新生成设计工作中任何阶段的设计。前一步设计在本文被称为"祖先实体"（ancester Solid）。一个多级树结构表示设计过程的方法完全可以支持UNDO和REDO操作。用户可简便地规定一条沿着树向上移动的通道，取消某一序列的运算。也可沿着树向下的一条通道，重做早期规定的某一序列运算。下面将详细描述UNDO和REDO运算。

2．2 构造单个实体的过程

修改实体的过程分为两类：一类是修改单个实体的运算，例如围边，面扫描运算；另一

类是复合实体运算，即从两个现有实体中生成一个新实体，集运算和拼合运算属于此类。下面介绍用单个实体运算构成一个实体的过程。为了弄清楚这一过程，我们先介绍一下单个实体运算的设计过程是如何产生的。复合实体运算将在下节进行介绍。

用户可通过规定调用原始运算指令来构制一个实体，第一条指令产生一个原始实体，以后指令用以修改实体。同时，构成以表示设计过程本身的树。树的每一节点表示设计开发的某个特定阶段中实体的状态。弧线表示单个实体指令和用来执行这些指令的原始运算的定义。系统给每个节点赋于一个内部标识符，这种标识符以后可用来调用由其代表的原始指令规定的原始运算。这样的安排支持了UNDO和REDO运算，也就是说，使用这些标识符，用户一经发现有错识，就可恢复到设计过程的任何阶段。

图1表示构制一个实体的过程以及表示设计过程的树结构。

(1) 首先生产出树根节点AO，这一节点以表示无效实体。然后产生第二个节点以表示原始实体。这个节点的标识符A_1，1表示在设计过程第一阶段中产生的第一个实体。

(2) 在实体A的F_1面上生成一条边，把F_1分成了F_1和F_2两个面。新产生的实体节点标识符为$A_{1.2}$。然后扫描F_1面，产生实体$A_{1.3}$，最后将E_1边变成弧形。

(3) 完成两次UNDO操作，再次生成$A_{1.2}$实体，然后对F_2面扫描，完成一次UNDO运算后，就对实体进行了一次修改，这时，表示设计阶段的数加1。因此，新实体的标识符为$A_{2.1}$，然后将边E_2变成弧形而形成$A_{2.2}$实体。

(4) 完成两次UNDO运算，一次REDO运算，重新生成实体$A_{1.3}$，然后对F_3面扫描，新产生的实体标成$A_{3.1}$。

由于该系统采用这样的方法表示实体生成的过程，所以它能够很快地再生成祖先实体的边界表达式。为了重新产生祖先实体，用户得完成UNDO操作，REDO操作或两者的组合。当完成UNDO时，采用原始运算的逆运算由现行实重新产生树可逆结构中较上一级的实体。当完成REDO时，生成树下层的实体。通过确定实体状态（即标识符）可以再次生成任何祖先实体。UNDO和REDO的计算时间远比其常规运算所要求的时间少得多。这是因为后者包括许多与原始运算不同的过程，例如相交检查和输入数据检查。用设计过程表示法再生成祖先实体的速度很快，使得有可能在交互式设计环境中采用。

2．3复合实体运算的表示法

一个实体采用单个实体运算构成后，我们可以对它采用复合动算，例如集合和拼合运算，图2表示复合实体运算的设计过程，由实体A.B生成新实体C。在图中，⊕节点对应一个复合实体运算。这一节点的上端与$C_{1.1}$相连；下端与实体$A_{2.2}$和$B_{1.3}$相连节⊕。
点表示：视两个实体为一单个实体。从复合实体运算调用的原始运算是以连到⊕节点的弧线所表示的。

终端用户并不需要了解怎样用原始运算的树结构表示实体生成的过程，他们只需要知道用于生成所需实体形状的命令。当某个用户请求再生成一个前阶段设计时，系统首先显示出所有祖先实体的标识符（ID）。用户这时可选择和输入所需实体的标识符，然后由系统重新产生对应于该标识符的实体。

3、概述集运算

表 1 示出了用于集运算的原始运算，在欧拉运算中，面中的开口称为“环”（ring），由于绝大部分的原始运算同时用于面和环的边界，所以两者都称为“围”（Loop）面的边界称为文围（Parant Loop）；面中开口的边界签为“子围”（Child Loop）。

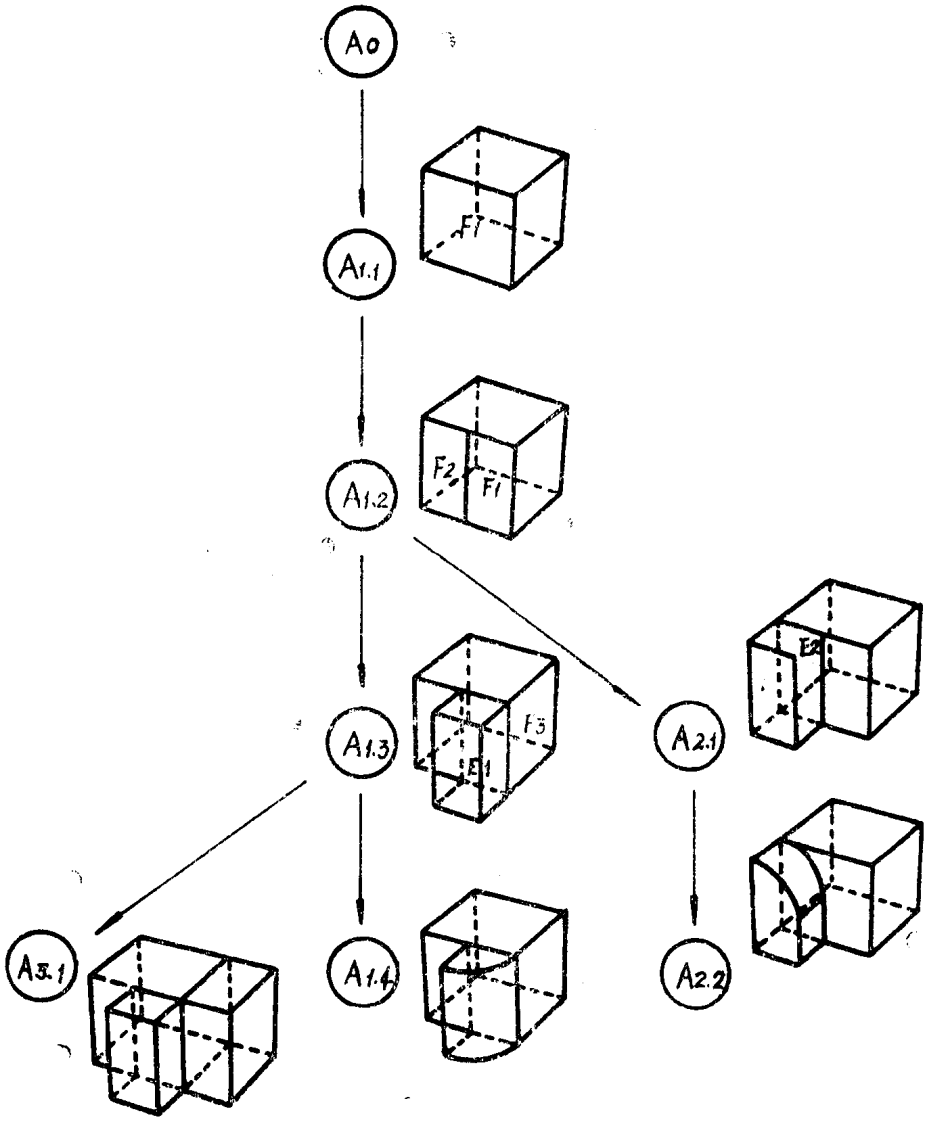


图 1　单个实体运算

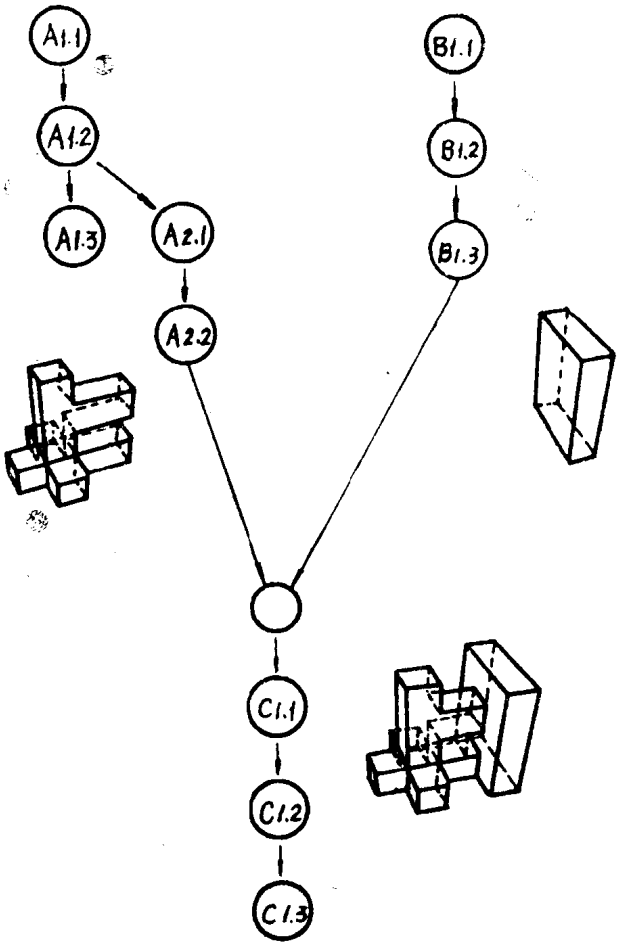


图 2　复合实体运算

集运算有三种形式：合并，求差，相交运算。A、B两实体的求差与相交运算是通过"实体求反"（NS）的原始运算和合并运算来实现的。

求差（A，B）=NS（合并（NS（A），（B））

相交（A，B）=NS（合并（NS（A），NS（B））

由于合并运算用于执行其它两种运算，因此它是本文主要讨论的对象。

请对照图3，每个实体有一对应的围（Loop），两个相对的围表示两个实体要拼合的部位。这些线条称为连线（J线）；线条的端点称为连点（J点）。当执行必要的运算后，系统根据J点生成连接顶点（J顶），根据J线生成连边（J边）。如图3所示图：在实体S_2上生成顶V_{21}，V_{22}，V_{23}，V_{24}。然后系统用原始运算产生连接这些点的各边，在实体S_2上生成边E_{21}，E_{22}，E_{23}，E_{24}，接着在实体S_1上生成J边E_{11}，E_{12}，E_{13}，E_{14}。J顶V_{11}，V_{12}，V_{13}，V_{14}。J边确定两实体之间的界面。实体S_2插入到实体S_1中的那部分的边界是通过一组原始运算消除掉。（图3（c））。

表 1　用于集运算的原始运算

Name	Definition
名称	定义
MEL	产生一条边和一个围
MEL	删除一条边和一个围
MVE	产生一个顶点和一条边
KVE	删除一个顶点和一条边
KPLMCL	删除一个父围，产生一个子围
KCLMPL	删除一个子围，产生一个父围
MEV	产生一条边和一个顶点
KEV	删除一条边和一个顶点
MEKL	产生一条边，删除一个围
KEML	删除一条边，产生一个围
MEVVL	产生一条边和一个顶，以及一个顶和一个围
KEVVL	删除一条边和一个顶，以及一个顶和一个围
NS	对一个实体求反
AS	加两个实体
DS	还原一个实体

为了拼合这两个实体，如图 3（d）所示，在两个对应的J顶间生成零长边。当所有不必要的边和顶消除后，集运算就到此结束。如图 3（e）所示。

4、合并运算的算法

本节对合并运算作出了更详细的介绍。所涉及的四个主要过程介绍如下：

J一点和J线的计算	4．1 节
J一边和J顶的生成	4．2 节
相关部分的删除	4．3 节
拼合两实体	4．4 节

4．1　J点和J线的计算

这一节介绍确定J点和J线的方法，这一过程包括四个步骤。

〈第一步〉：作初步的相交检查

为得到两实体所有的面生成出最小的三维框，以找出各对相交面。两实体面之间的相交可以用这些三维框作初步检查。

〈第二步〉面与面边界各交点的计算

在图 4 中，F_a代表实体A上的一个面，F_b代表实体B上的一个面。系统首先要计算出F_a平面与F_b面各条边所有交点的位置，尔后再计算出F_b平面与Fa面的各条边所有交点的位置。每与文实体有以下三种关系。

1）　在一个顶上

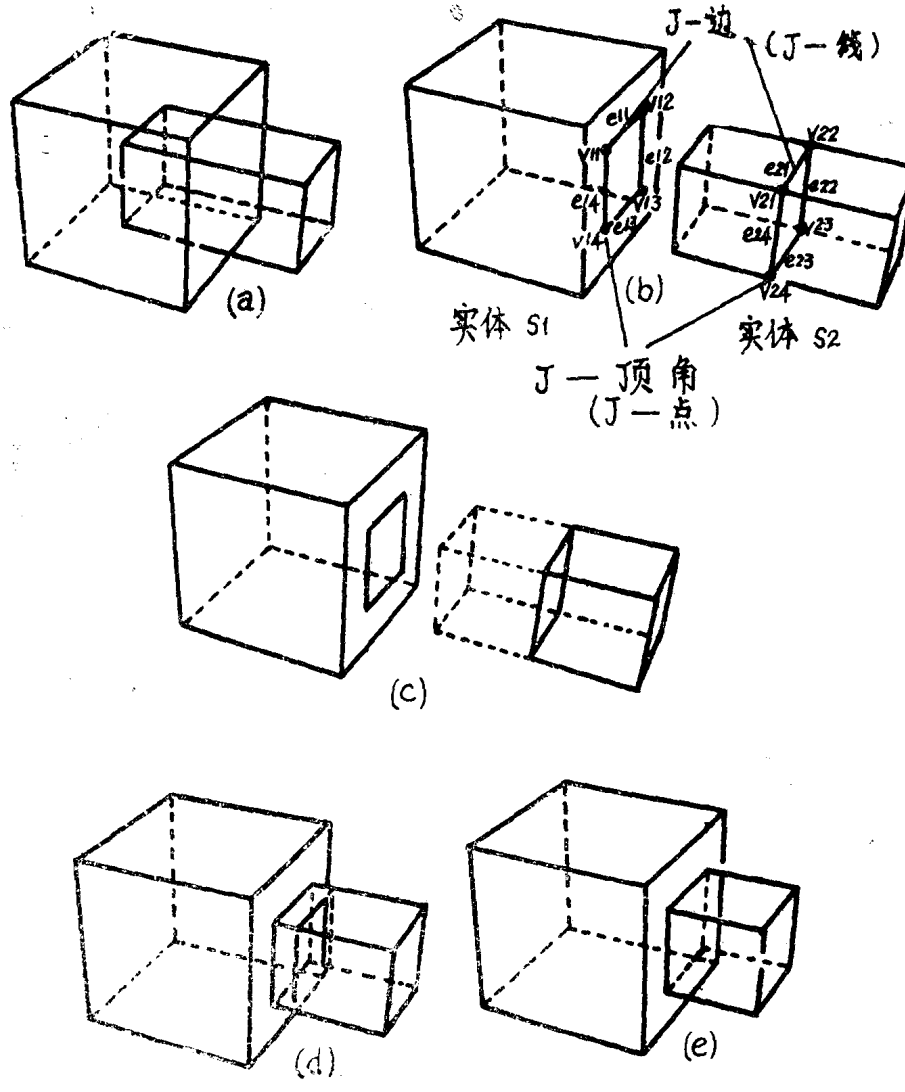


图 3　合并运算过程

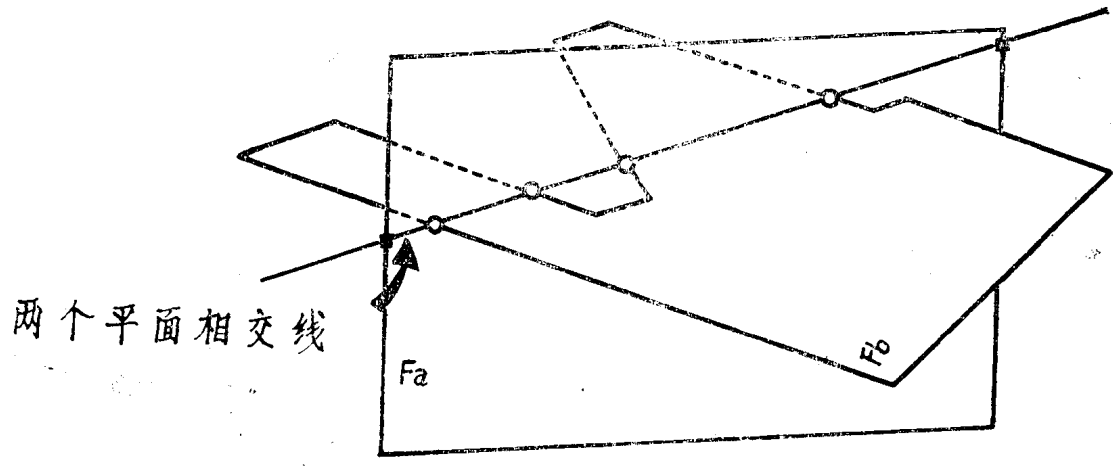
2)　在一条边中（不包括顶）

3)　包含在一个围内（不包括边）

这些关系得在这一步中计算出来。

〈第三步〉确定J点

在这一步中，J点在前一步获得的两点中间确定。这些点位于表示F_a面和F_b面相交的一条直线上(图 4)。把J点按其座标值排序后，两个任意的相邻点确定一条线段。只有当其端点包含在两个面中，这一线段就被称为J线。同理，同在两个面上的一条线段的两端点称为J点；点是由点围算法确定的。



围，F_b 平面与 F_a 面的交点　　O：J—点

图 4　J—点的运算

点围的算法求出“围”和给定目标点开始的半线之间交点数。如果结果是奇数，点位于围的内部；如果是偶数，点位于围外部。

注意：在面的平面与面的边之间相交的所有各点由第二步求出，位于一条直线上。由于这条直线与上述的半条线类似，我们仅数一下在第二步中计算的相交点数就可以了，用不着再进行几何算法。

〈第四步〉数据采集

系统把相交实体的几何和拓补数据存贮在两个表中（一表用于J点，另一表用于J线）。

4．2 J边和J顶的生成

这时系统用J点和J线表格中的数据生成J顶和J边。

4．2．1 J顶的生成的生成

J点与其文实体的拓朴关系由J点表格确定。正如4．1节中谈到的，有三种可能出现的系：J点可能处于一个顶上，也可能在一条中，或者包含在一个围中。如果一个点对应某个要拼合的实体的一个顶时，系统把该顶看作为J顶。如果J顶在位于某实体边上的J点上时，则可用原始运算MVE求算。若其它所有J点在一个围中，则应先从此点开始产生边，再求出各相应的J顶来。

产生出J顶后，产生J边需用什么样的原始运算，就比较容易确定了。

4．2．2　J边的生成

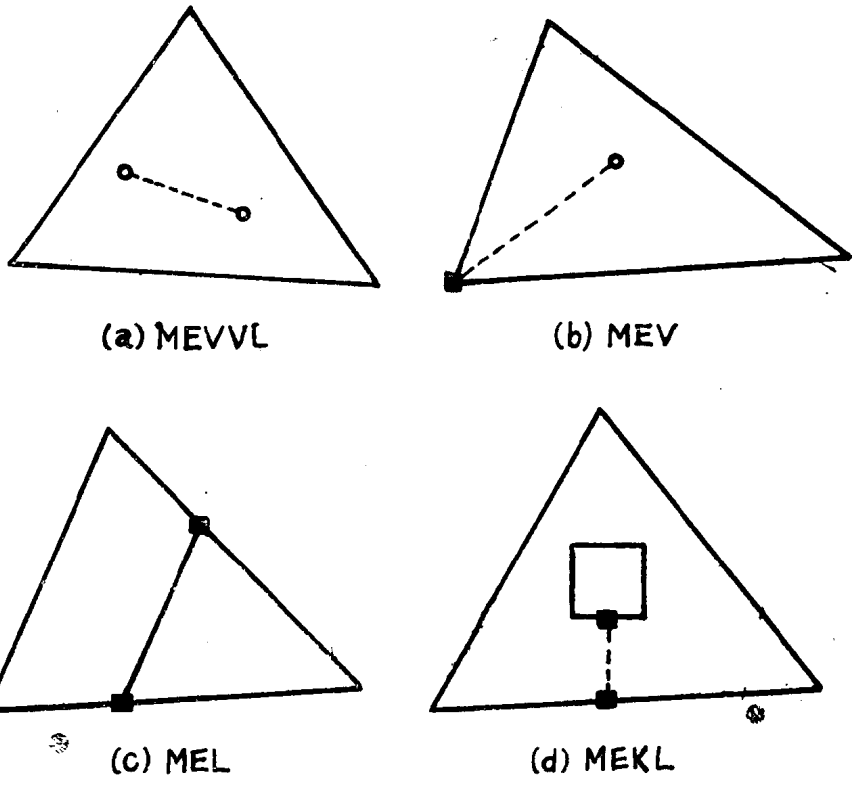
这时系统用J点和J线表格中的数据生成J顶和J边

4．2．1　J顶的生成

系统用J线表格的信息，通过原始运算，在J线上生成J边（图5）。

J线按其设计阶段可分为以下三类：

a）其J顶即两个端点还未生成，在这种情况下，用MEVVL运算生成J边，两个J顶和一个“围”（图5（a）所示）



○··J点　　产生J顶的J点

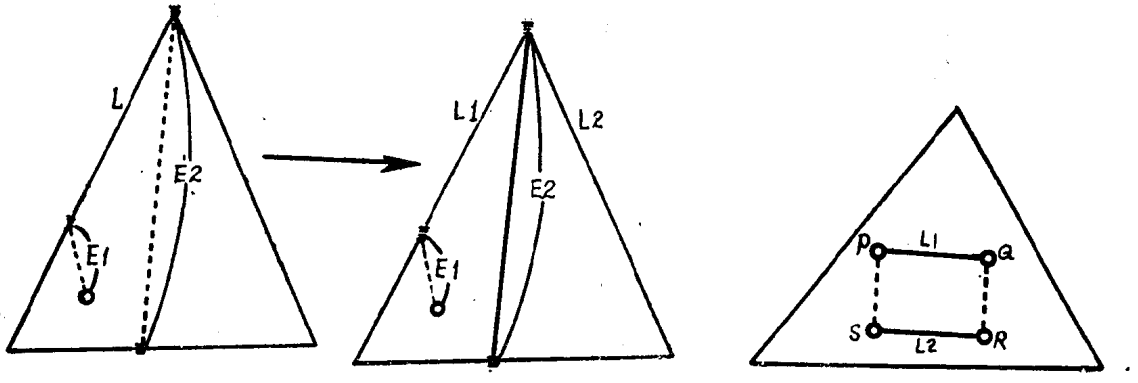
图5　边的生成

b) 只生成了其中一个顶（端点），在这种情况下，用MEV运算生成J边和另一个顶（图5（b）所示）。

c) 两个顶都已生成，在这种情况，如果两个顶都位于同一围和同一平面上，（如图5（c)），用MEL运算生成J边和一个围；如J顶位于同一面上，而不在同一围上(如图5(b))，则采用MEKL运算。

以逻辑顺序运用原始运算，可以避免不必要的计算，例图6示出：在“围”L中，E_1边和E_2边的生成。如果E_2边在E_1边之前生成，E_1将属于新围L_1，而不属L中的边，因此，在生成E_1边之前，系统必须确定该边属哪个围。如果E_1在E_2之前用MEV运算产生出来，就不必要进行这样的算法。

图6（b）示出了按逻辑顺序运用原始运算的重要性的另一例子，图中，四边形PQRS形成了一个围，通过MEVVL运算，产生边PQ和边PS，则必须消去L_1围或L_2围。如果依次产生出边PQ，QR，RS，SP，则可避免产生这个不必要的围。



(a)MEV运算和MEL运算　　(b)重复MEVVL运算

图 6　边生成的顺序

用以生成实体A，实体B的各边的数据存入J边表内。此表用于下一步的合并运算，以删除插入部分的边界。

4.3　插入部份的消除

在合并运算里，我们把一实体插入另一实体边界的任一部分称为“插入部分”，下面简要地介绍这部分的消除。

4.3.1　插入部分的检测

如果一个实体一条J边所在的面未插入到另一实体中，系统自行分配给该面一个“+”号，它表示要保留。如某实体的面插入到另一实体中，系统给其一个“-”号，表示该面要被“消除”与之相邻的面标上“0”这些面以后可以标上“+”号，或“-”号。

图8示出两实体A和B在J边会合的横截面。阴影部分表示两实体的内部。A_1，A_2，和A_3是与两相邻连接的四个面相交的角度。采用反三某函数计算出这些角度来确定标号“+”、“-”或“0”。然而这些函数要花大量的时间计算，我们大多采用了另一种计算，即通过每一面的上交的内积求出对角度的量，这种计算比求出每个角度要快得多。

为了简便起见，设角A_1，A_2，A_3具有下列值：

$0 < A_1 < 360$

$0 < A_2$，$A_3 < 360$

$A_2 \neq A_3$

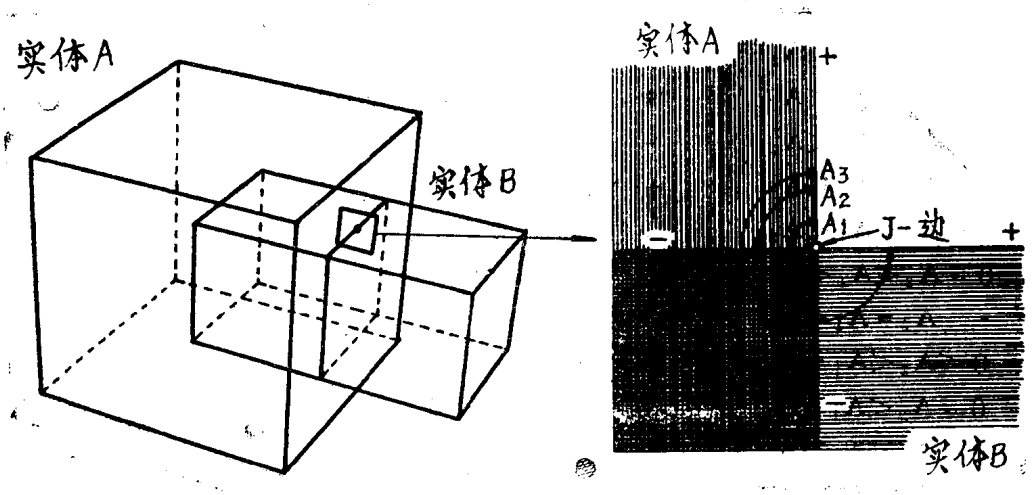


图 7　给出了生成J边的算法

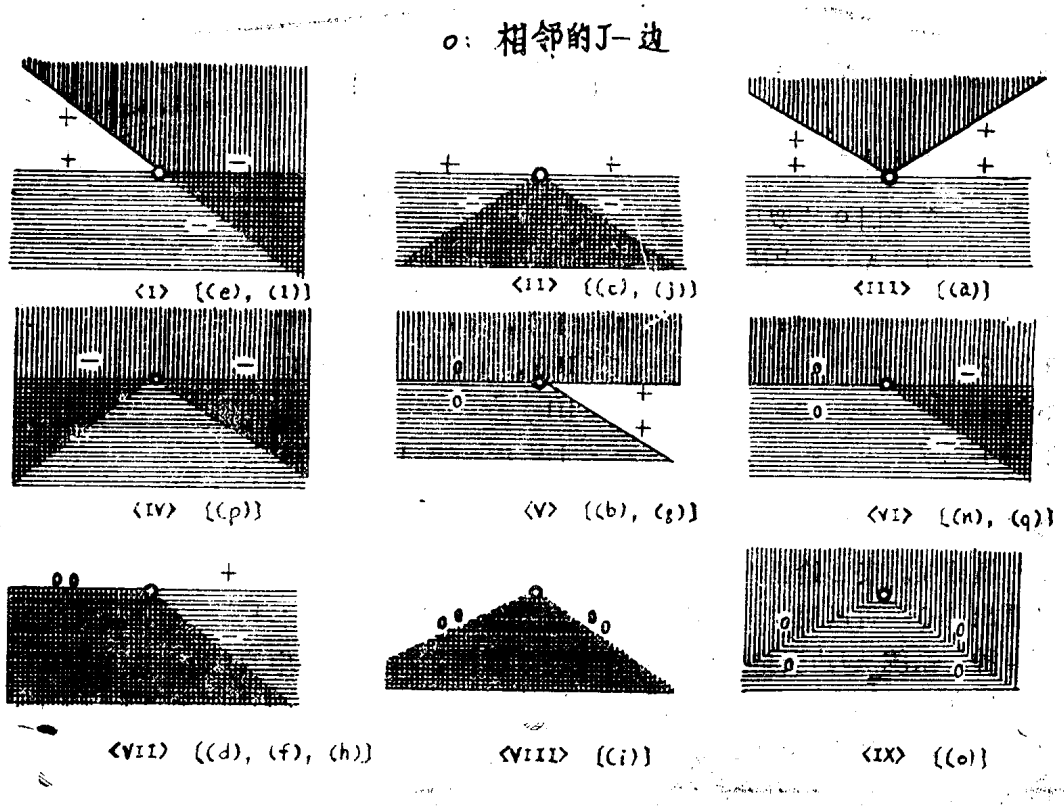


图 3　在两个相交实体截面

表 2　用A_1，A_2，A_3确定的符号

		solid B		solid A		solid B			
(a)	$A_1<A_2<A_3$	+	+	+	+				
(b)	$A_1=A_2<A_3$	+	0	+	0→+	−	+	−	
(c)	$A_1<A_3<A_2$	−	−	+	+				
(d)	$A_1=A_3<A_2$	−	0	+	0→−	−	+	+	
(e)	$0<A_2<A_1<A_3$	+	−	+	−				
(f)	$0=A_2<A_1<A_3$	−	0	+	0→−	−	+	+	
(g)	$A_2=A_1<A_3$	+	0	+	0→+	−	+	−	
(h)	$0<A_2<A_1=A_3$	+	0	−	0→+	+	−	−	
(i)	$0=A_2<A_1=A_3$	0	0	0	0	+	+	−	−
(j)	$0<A_2<A_3>A_1$	+	+	−	−				
(k)	$0=A_2<A_3<A_1$	+	0	−	0→+	+	−	−	
(l)	$0=A_3<A_1<A_2$	+	−	+	−				
(m)	$0=A_3<A_1<A_2$	+	0	+	0→+	−	+	−	
(n)	$0<A_3<A_1=A_2$	−	0	−	0→−	−	−	−	
(o)	$0=A_3<A_1=A_2$	0	0	0	0→−	−	−	−	
(p)	$0<A_3<A_2<A_1$	−	−	−	−				
(q)	$0=A_3<A_2<A_1$	−	0	−	0→−	−	−	−	

表 3　赋于实体A和B的符号

（参考图 9 的罗马数字）

	实体A		实体B		
事例 1	+	−	+	−	〔〈 Ⅰ 〉，〈 II 〉〕
事例 2	+	+	−	−	〔〈 II 〉，〈 VII 〉，〈 VIII 〉〕
事例 3	+	+	+	+	〔〈 III 〉〕
事例 4	−	−	−	−	〔〈 ⅠV 〉，〈 VⅠ 〉，〈 ⅠX 〉〕

每条J边有两个附着的面，图 9 示出了合并算中 4 个这样的面会合于两相邻J边上的九种情况下的横截面。表 2 给出的“+”，“−”，“0”符号以及图9中对应各实例角度数字。从表 2 看出“0”可以用“+”或“−”号代换。如表3所示标号“+”与“−”的组合数，减少到 4 种。系统从J边表格中删除所有在表 3 事例 2，3，4 中的J 边，这时“R”标志（表示删除）赋予标有“−”号的面；“K”标志（表示保留）赋予标有“+”的面。

4．3．2　插入部分边的消除

插入部分的边界由赋有“R”标记的面组成。系统这个符号确定插入部分的边。我们称这些为“插入边”。在这一程序中，所有的插入边用原始运算消除。

我们感兴趣的是，用逆运算对实体作有效的再生成。该算法的目的在于避免用几何运

算，这是因为几何运算降低了对实体进行再生的能力。然而，逆程序可能会要求这种运算。例，当一个包含有“子”围的围被一条新的边分割时（如图10所示），子围LC被插入在围L_1中。当边E生成后，必须确定子围是插在围L_1中，还是在围L_2中。把插入部分的子围变成一个父围就可以避免这种情况。然后，用KEV，KEVVL，KEL，或KEML原始运算消去“插入边”。

```
Wake_J—Edge
    do
        make aii J-edges that can be
                created by MEV;
        make an edge that can be
                created by WEVVL;
    while (edge that can be
                created by WEV or WEVVL exists`
    make all remainng ebges using WEL or WEKL;
```

Figure 7 . Algorithm for creatinh J-ddes

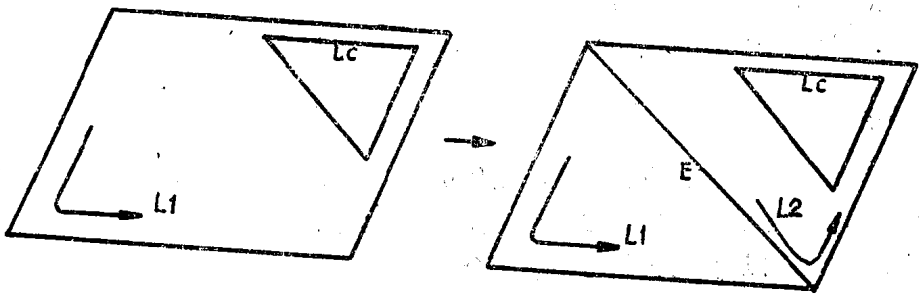


图 9　相交实体间的典型关系

4．4　合并两实体

删除插入部份以后，两实体这时在J边上会合。这些J边形成了两围，一个实体上一个围（如图 3 所示）。这些围称为连接围（J围）。系统在合并两个实体时，运用MEKL或WE L运算产生出两个J围上相应J顶间的零长边。

4．4．1　收缩围

我们合并实体的方法要求一个实体J围中相连的所有J顶必须与另一实体J围中的J顶一一相对。然而，在插入边被消除时，可能会产生一个“收缩围”如图11所示，收缩围是在两个或两个以上的顶之间被“收缩”的围。这种围中的J顶不直接与相交实体的顶相对应。

收缩围定义如下：

在一个围中的顶依次用V_0，V_1…，V_n表示。Vi位于Vi－1和Vi＋1之间（i＝1，2…，n－1）。Vo与Vn及V1相邻，Vn与Vn－1及Vo相邻。收缩围必须具有一个满足下列条件的Vi顶。

(1)　Vi＝Vj，对于某些j（i≠j）

(2)　Vi－1和Vi＋1（必须不满足条件(1)）。我们称这些顶为子顶（C－Vertices）。

如果出现了一个收缩围，则不能采用两实体合并的方法。图11(a)说明了这样一个情况。实体X中的收缩围（图11(b)）与两实体Y中的两个非收缩围相对应。因此，必须在子顶上分开收缩围。分离收缩围是分两步完成的。首先，使子顶个数加倍，然后生成两个分离的实体，使每一实体含有一个子顶，这样就把“收缩围”分成两个未收缩围，其J顶直接与另一物体的J顶相对应。

4．4．2　删除多余的围

两个实体在完全相同的两J围上拼合，下面过程用以删除其中的一个围，为方便起见，我们把两实体标为A和B。

(1)　采用KEL，KEV和KVE联合运算，删除实体A中所有的J顶和J边。

(2)　这时，只保留实体B的中的J边。如果接于J边的两个面的向量方向一致，那么不必用这条边表示实体。因此，用KEL或KVE运算删除它。

这时，合并运算结束。

5、例子

图12示出合并运算中使用的原始运算，原始实体A，B可以用这些运算的逆运算重新产生出来。图13示出了集运算的另一实例。图13(a)和(b)为两实体的草图；图13(c)为两实体相交的结果。这两个实例都是用DESIGN—BASE产生出来的。

6、结束语

本文已介绍了表示实体设计过程，以便快速生成设计过程中前步设计的方法。并介绍了实现可逆集运算的方法。用这种方法，实体可以用原始运算得到生成和修改。所有这些运算存入在表示实体设计过程的树结构中。某个已完成的集运算可以通以相反顺序重新做此运算而被取消掉。DESIGN—BASE实体制模系统具有这种能力。

在集运算中，求出两实体之间的相交点的计算量很大，而逆运算因仅采用原始运算而速度非常快。

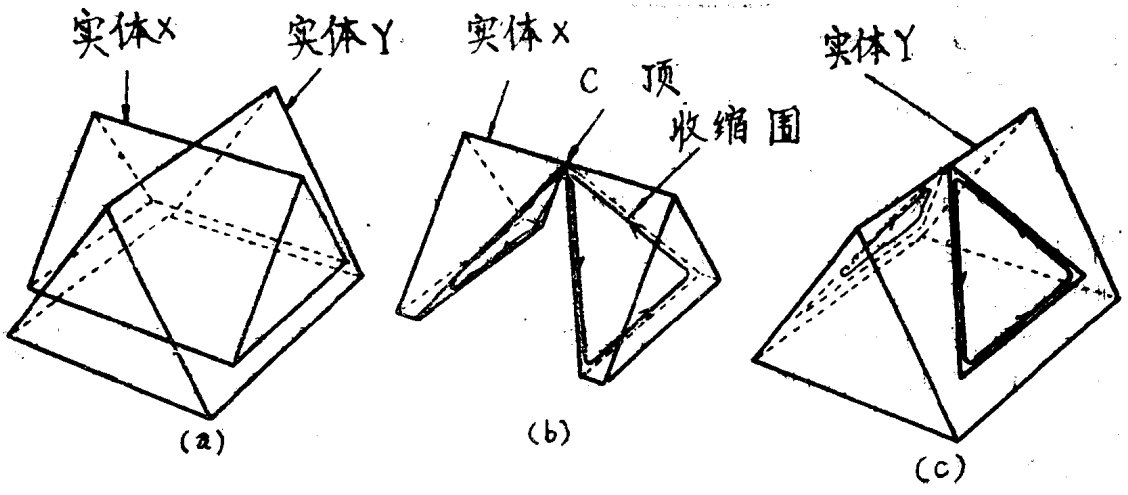


图11　收缩的围