IPC 平面裁剪算法的设计与实现

刘 晖,田 泽,黎小玉,陈 佳
(中航工业西安航空计算技术研究所,陕西 西安 710119)

摘 要:Sutherland-Hodgman 算法是平面裁剪中常用的一种算法,其以顶点序列为基准,对多边形进行逐边裁剪,算法实现简单、效率高,但不适用于对凹多边形进行裁剪。文中以 Sutherland-Hodgman 算法(文中称为预处理算法)思想为基础,提出了一种亦可对凹多边形进行裁剪的综合多边形裁剪(Integrated Polygon Clip,IPC)算法,实现了对多边形裁剪的统一化操作。介绍了平面裁剪的使用对象,预处理裁剪算法的原理及实现关键步骤,分析了该算法用于对凹多边形裁剪的局限性,在此基础上提出一种适用于任意多边形裁剪的算法。以例证的方式演示了算法的过程,验证了算法的正确性。验证结果表明 IPC 算法实现了对多边形的统一裁剪。

关键词:平面裁剪;Sutherland-Hodgman 算法;凹多边形;综合多边形裁剪算法
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2014)02-0224-05
doi:10.3969/j.issn.1673-629X.2014.02.056

Design and Implementation of IPC Arithmetic Used in Plane Clipping

LIU Hui,TIAN Ze,LI Xiao-yu,CHEN Jia
(Aeronautical Computing Technique Research Institute,Xi'an 710119,China)

Abstract:Sutherland-Hodgman arithmetic is commonly used in clipping plane,that arithmetic implements polygon clipping one side by another in the order of vertex sequence. The arithmetic is easy to realize and high efficiency,but not support for the concave polygon clipping. Based upon Sutherland-Hodgman arithmetic (also called preprocessing arithmetic),rise an IPC (Integrated Polygon Clip) arithmetic,that also can be used in concave polygon clipping,realized the unification of polygon cutting operation. Introduce the objection of plan clipping,the theory of preprocessing arithmetic and its key point in implementation,analyze the arithmetic's limitations,then rise a new arithmetic,supported in all polygon clipping. In the way of illustration,it proves the correctness of the new arithmetic. The result indicates that IPC arithmetic realizes the polygon clipping with consolidated actions.

Key words:plane clip;Sutherland-Hodgman arithmetic;concave polygon;IPC (Integrated Polygon Clip) arithmetic

0 引 言

所谓平面裁剪,是为了分区描述或重点描述图形对象的某一部分,指定一个窗口或裁减区域作为图形边界,对于将要描述的图形,若其置于窗口内则予以保留,若其置于窗口外则将被舍弃[1]。

Sutherland-Hodgman 算法是早期的一种线段裁剪算法,其算法简单且易于实现,应用较为广泛。但由于现在的图形处理器要处理的多边形越来越复杂,文中对其算法进行了相应的改进,提出了一种适用于对任意多边形进行裁剪的 IPC 算法。该算法以接收到的顶点序列为输入,经过预处理算法的处理,将输出顶点按照一定顺序重新排列并进行相应的处理后,输出最终的顶点序列,从而实现了对任意多边形的正确裁剪。

1 平面裁剪

平面裁剪是为了判断图形元素是在平面的内侧还是外侧,并对不同的情况进行分类处理,图形元素包括点、直线、多边形(包括三角形),因此平面裁剪模块的功能主要完成对以上三种图元的裁剪[2-3]。

1.1 点的平面裁剪

用指定的裁剪平面来裁剪空间中的点,只需要判断点与裁剪平面的位置即可:若点位于平面的外侧,则

收稿日期:2013-06-08 **修回日期**:2013-09-06 **网络出版时间**:2013-11-29
基金项目:总装 2012 预研基金(9140A08010712HK61095);中国航空工业集团公司创新基金(2010BD63111);总装"十二五"预研项目(51308010601)
作者简介:刘 晖(1986-),男,陕西宝鸡人,硕士研究生,研究方向为 SoC 设计与验证;田 泽,博士,研究员,中国航空工业集团首席技术专家,研究方向为 SoC 设计、嵌入式系统设计、VLSI 设计等。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20131129.0826.010.html

丢弃该点的信息;若点在平面上或平面的内侧,则保存该点的信息。

1.2　直线的平面裁剪

用指定的裁剪平面来裁剪空间中的直线,根据直线两个端点的坐标信息判断线段与裁剪平面的位置关系,从而进行线的平面剪裁,确定输出结果;裁剪过程中,如果没有新点产生,把相关输入顶点的信息全部输出[4];如果有新点产生,需要用求交点的公式计算新点的位置坐标;直到最后一个剪裁平面完成剪裁后,再将保存的顶点信息全部输出。

1.3　多边形的平面裁剪

若对一个封闭多边形进行裁剪后生成若干个点或线段,这显然是不合理的。因此,将要实现的算法是将一个封闭的多边形裁剪为若干个封闭的多边形。

用指定的裁剪平面来裁剪空间中的多边形,根据 Sutherland-Hodgeman 多边形裁剪思想对多边形平面进行剪裁[5];同样在裁剪过程中,如果没有新点产生,把相关输入顶点的信息全部输出;如果有新点产生,用求交点的公式计算新点的位置坐标,并插值计算出新点的属性信息;但是在平面裁剪结束后,需要根据输出点的个数将所有点按照规则装配成若干三角形后再输出。

2　IPC 算法分析与设计

2.1　预处理算法

2.1.1　预处理算法原理

预处理算法以 Sutherland-Hodgeman 裁剪算法为基础,用裁剪平面对三角形进行逐边裁剪,主要完成有向线段与裁剪平面不同位置关系的处理结果,其过程采用了分而治之的策略。

具体流程描述如下:该算法的输入参数是多边形的一串有序顶点 $P_0,P_1,\cdots,P_n$,头节点记为 P,尾节点记为 S。当用一个裁剪平面对多边形进行裁剪后,若产生新的点,则将新点的序列作为下一条裁剪边处理过程的输入。首先沿着多边形的边从顶点 P_n 移动到 P_0,再顺序移动到 P_n,在每一次移动时,都检测连续的两个顶点与裁剪边的相互关系。判断要不要计算交点,若有交点则计算交点并输出,紧接着再判断 P 点是否在平面的内侧,若在内侧则输出,若不在内侧则丢弃;若没有交点则直接判断 P 点在平面的内侧还是外侧。在二维空间中,多边形的边是根据其顶点依次连接生成的,即从 P_i 到 P_{i+1} 是一条边,而最后一条边是从 P_n 到 P_1,依次裁剪完每一条边后裁剪过程结束[6-7]。

2.1.2　预处理算法实现多边形裁剪的关键步骤

对多边形而言,得到的是连续的顶点信息,因此对多边形的裁剪就简化为对连续直线的裁剪,裁剪后将需要保留的顶点信息保存起来,依次将所有的顶点有序连接,即得到裁剪后的多边形。

裁剪时需要判断直线的两个顶点与指定裁剪平面的位置关系,如图 1 所示。

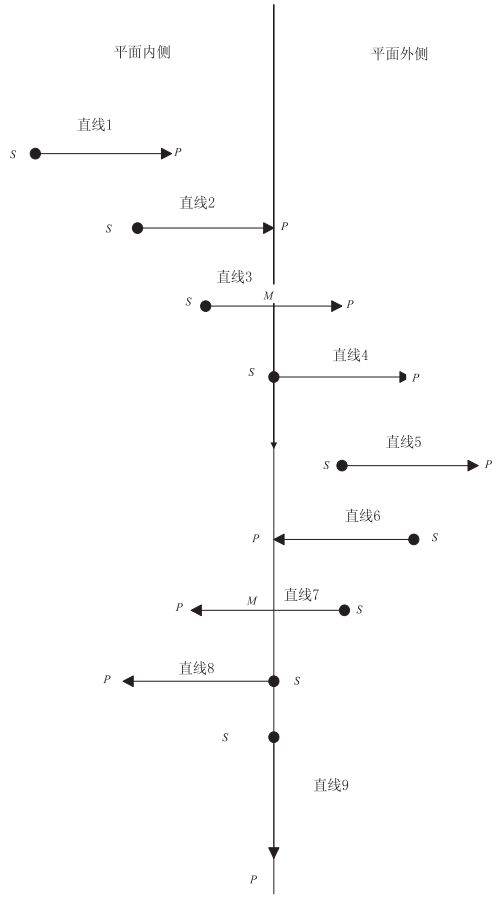


图 1　直线与平面相交分类图

● 直线 1:直线的两个顶点都在平面的内侧且都不在平面上,只保存尾部顶点 P 的信息;

● 直线 2:直线的头顶点在平面的内侧,尾顶点在平面上,只保存尾部顶点 P 的信息;

● 直线 3:直线的头顶点在平面的内侧,尾顶点在平面的外侧,直线与平面的交点为 M,只保存交点 M 的信息;

● 直线 4:直线的头顶点在平面上,尾顶点在平面的外侧,不保存相关的顶点信息;

● 直线 5:直线的两个顶点都在平面的外侧且都不在平面上,不保存顶点的信息;

● 直线 6:直线的头顶点在平面的外侧,尾顶点在平面上,不保存相关的顶点信息;

● 直线 7:直线的头顶点在平面的外侧,尾顶点在平面的内侧,直线与平面的交点为 M,保存交点 M 和尾顶点 P 的信息;

● 直线 8:直线的头顶点在平面上,尾顶点在平面的内侧,保存头顶点 S 和尾顶点 P 的信息;

● 直线 9:直线在平面上,保存头顶点 S 和尾顶点

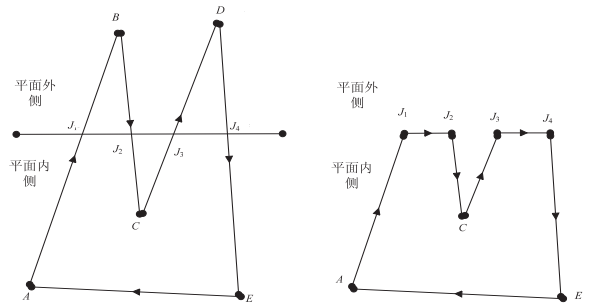
P 的信息。

图 2 包括了有向线段与裁剪平面所有可能的位置关系,是进行多边形裁剪的基础步骤[8-9]。

2.1.3 预处理算法的缺陷

预处理算法的特点在于多边形的每条边依次地沿着某一个裁剪平面进行裁剪,每条边之间不具有数据相关性,裁剪相对独立;若指定了多个裁剪平面,裁剪也是按照顺序进行的,裁剪的对象为上次裁剪后所保存的多边形图形。此算法相对直接简便,容易实现。

预处理算法在对凸多边形进行裁剪时可以正确的执行,但是在对凹多边形进行裁剪时,若裁剪出来的是一个多边形,算法正确;若裁剪出来的是多个多边形则会存在一个缺陷:其中会包括一些不属于这个多边形的边,如图 2 所示。



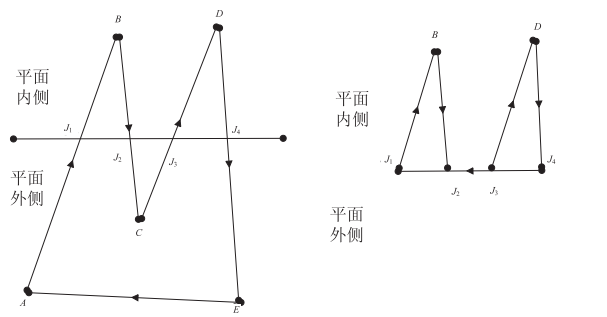
(a) 凹多边形 (b)裁剪结果

图 2 凹多边形裁剪(1)

对图 2(a)所示的凹多边形进行裁剪时,先得到原多边形的顶点顺序 (A、B、C、D、E) 和一个裁剪平面(方程表示为 $hX + kY + mZ + nW = 0$,h、k、m、n 为平面的系数),由预处理算法得到的裁剪后顶点顺序为(J_1、J_2、C、J_3、J_4、E、A),按照顶点顺序和坐标连接后,得到的仍为一个凹多边形,如图 2(b) 所示。

由图可见,裁剪后得到的图形与期望得到的图形一致,预处理算法对剪裁后仍为一个凹多边形的图形,能够实施正确的裁剪。

对图 3(a)所示的多边形进行裁剪时,裁剪过程同上。



(a)凹多边形 (b)裁剪结果

图 3 凹多边形裁剪(2)

最后得到的顶点顺序为 (J_1、B、J_2、J_3、D、J_4),将顶点按照顺序连接起来后得到的多边形如图 3(b) 所示。

由图可见,原凹多边形应被裁剪为 2 个三角形,但是又多了一条线段(J_3J_2),所以在对凹多边形裁剪后得到的图形可能会增加一些不属于多边形的边。

因此,IPC 算法的改进主要针对这一类的凹多边形裁剪。

2.2 IPC 算法的设计

为了改进预处理算法,一般有两种思路:一是将凹多边形在剪裁前划分为多个凸多边形后再进行裁剪,裁剪后将得到的图形再进行拼接,消去重复的边即可;另一种是在凹多边形裁剪后,将产生的多余线段消去,这样也可得到正确的裁剪图形。文中基于第二种思路进行算法的改进,找到一种可以对凹多边形进行正确裁剪的改进算法,同时此算法又适用于凸多边形,这样就完成了对任意多边形的裁剪。

2.2.1 算法设计思路

IPC 算法的设计是在预处理算法的处理基础上实现的,主要是对裁剪后的有序顶点集合进行相关操作,从而得出正确的裁剪后的顶点集合,具体思路设计如下[10-11]:

● 将多边形的顶点按顺序依次存在集合 A 中;

● 按照预处理算法进行裁剪,将裁剪后的顶点有序集记为 B;

● 求出多边形与平面的交点,按照 X 轴坐标或 Y 轴坐标进行排序,记有序的交点集合为 C;

● 按照集合 C 中的顺序,将集合 B 中的第一个点设置为第一个交点,按顺序重新排列集合 B 中的顶点,新的顶点序列记为 D,若第一个交点不存在,依次取下一个交点排序;

● 在 D 中依次判断 C 中的点是否连续成对出现;

● 若上一步判断结果为“是”,则按照预处理算法进行裁剪,若其中有连续的顶点,将其合并,输出裁剪后的多边形,裁剪结束;若判断结果为“否”,继续执行;

● 将 D 中的非 C 顶点按照 D 中的顺序取出,组成点集 E;

● 在 E 中取出第一个点(M_1),在 D 中按顺序查找 M_1 及其以后的顶点,直到其后的点为 C 中的点,记为 M_2。输出 D 中 M_1 的前一个顶点到点 M_2 的顶点序列,记为 K_1。在 D 中删除 K_1 的顶点序列,在 E 中删除点 M_1;

● 执行上一步骤,直到 E 中的顶点集为空,每次产生的新的顶点序列记为 $K_x(K_2, K_3, K_4, \cdots)$;

● 若 D 中的顶点集为空,则裁剪结束;若 D 中的顶

点集不为空,则剩余的点全为集合 C 中的点,裁剪结束;

● 裁剪后的多边形即为顶点序列 K_x。

2.2.2　算法流程图

具体的算法流程清晰地描述了整个算法的实现过程,其中顶点集合的表述与设计思路中一致[12-13],流程图如图 4 所示。

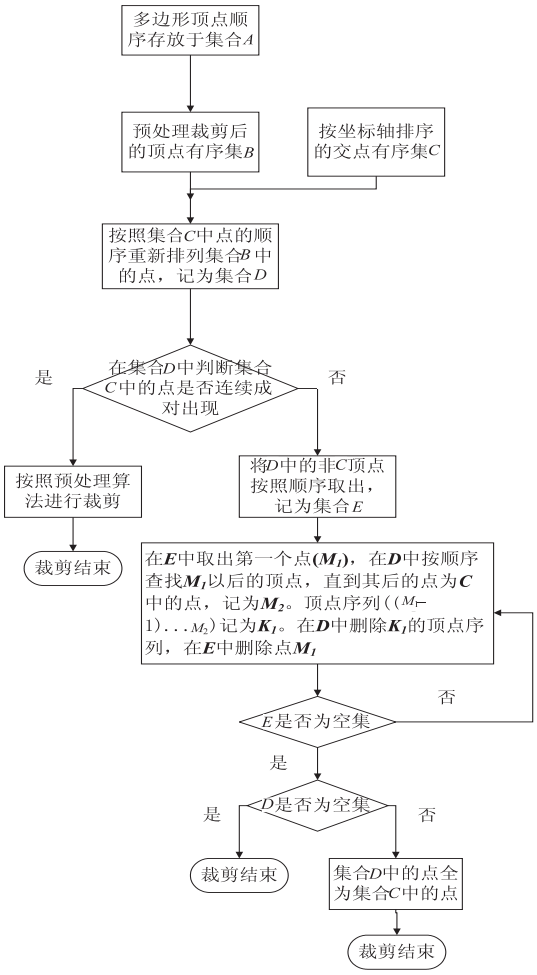


图 4　算法流程框图

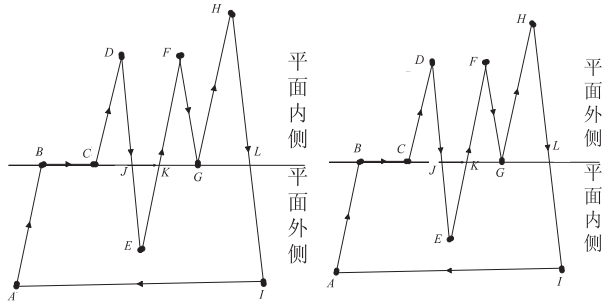
2.2.3　算法伪代码实现

算法的伪代码提供了算法实现的具体过程,如下所示:

```
Void IPC_arithmetic(type *A, type *C)//参数集合 A 与集合 C
{
B=pre_process(A);//对 A 中的顶点进行预处理操作;
D= re_arrang(B, C);//按照 C 中点的顺序将 B 中的点重新排列;
if((B[i]∈C)&&(B[i+1]∈C))
{ Exit();}
Else
{
E=D-C;//取出 D 中不属于 C 的点;
WHILE(E! =NULL)
do{
If((E[1]==D[i])&&(D[j]∈C))
{  K[1]={D[i-1]…D[j]};
   D=D-K[1];
   E=E-E[1];}
}
If(D==NULL)
{ Exit(); }
Else
{  D[i]∈C;
Exit();}
}
```

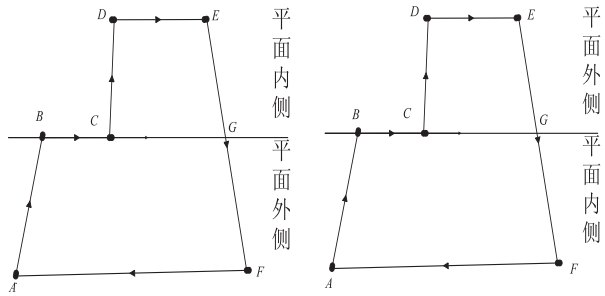
3　IPC 算法的验证

如图 5 和图 6 所示的四组实验分别包括了凹多边形和凸多边形不同方向的剪裁,每种多边形中都涵盖了线段拐点与裁剪平面的所有可能性[14],覆盖了可能会出现的多边形的所有情况,具有典型的代表意义。



(a) 凹多边形 1　　(b) 凹多边形 2

图 5　凹多边形验证图



(a) 凸多边形 1　　(b) 凸多边形 2

图 6　凸多边形验证图

对每种多边形的验证都严格按照算法设计的步骤进行,将裁剪后的结果与预期结果相比较。

情况一:

如图 5(a)所示的凹多边形,按改进后的算法来剪裁:

(1) 集合 $A=\{A,B,C,D,E,F,G,H,I\}$;

(2) 从边 FG 开始按照改进算法进行裁剪,生成集合 $B=\{G,G,H,L,C,D,J,K,F\}$;

(3) 集合 $C=\{B,C,J,K,G,L\}$;

(4) 集合 $D=\{C,D,J,K,F,G,G,H,L\}$;

(5) 在 D 中依次判断 C 中的点是否连续成对出现,$(C)(J,K)(G,G)$ 和(L),所以不是成对出现的,继续执行;

(6) 集合 $E=\{D,F,H\}$;

(7) 顶点 $M_1=D,M_2=J$,顶点序列 $K_1=\{C,D,J\}$;

(8) 依次可得 $K_2=\{K,F,G\}$,$K_3=\{G,H,L\}$;

(9) 顶点集合 D 为空集,剪裁结束。

最终的裁剪结果为三个独立的三角形$\{C,D,J\}$,$\{K,F,G\}$,$\{G,H,L\}$,与预期结果相同。

情况二:

如图5(b) 所示的凹多边形,按改进后的算法来剪裁:

(1) 集合 $A=\{A,B,C,D,E,F,G,H,I\}$;

(2) 从边 HI 开始按照改进算法进行裁剪,生成集合 $B=\{L,I,A,B,J,E,K\}$;

(3) 集合 $C=\{B,C,J,K,G,L\}$;

(4) 集合 $D=\{B,J,E,K,L,I,A\}$。

在 D 中依次判断 C 中的点是否连续成对出现,$(B,J)(K,L)$,是成对出现的,裁剪结束。

所得的裁剪多边形为$\{B,J,E,K,L,I,A\}$,与预期结果相同。

结论:改进后的裁剪算法可对凹多边形进行正确的裁剪。

情况三:

如图 6(a)所示的凸多边形,按改进后的算法来剪裁:

(1) 集合 $A=\{A,B,C,D,E,F\}$;

(2) 从边 BC 开始按照改进算法进行裁剪,生成集合 $B=\{C,D,E,G\}$;

(3) 集合 $C=\{B,C,G\}$;

(4) 集合 $D=\{C,D,E,G\}$;

(5) 在 D 中依次判断 C 中的点是否连续成对出现,(C) 和(G),所以不是成对出现的,继续执行;

(6) 集合 $E=\{D,E\}$;

(7) 顶点 $M_1=D,M_2=G$,顶点序列 $K_1=\{C,D,E,G\}$;

(8) 顶点集合 D 为空集,剪裁结束。

所得的裁剪多边形为$\{C,D,E,G\}$,与预期结果相同。

情况四:

如图6(b) 所示的凸多边形,按改进后的算法来剪裁:

(1) 集合 $A=\{A,B,C,D,E,F\}$;

(2) 从边 BC 开始按照改进算法进行裁剪,生成集合 $B=\{G,F,A,B\}$;

(3) 集合 $C=\{B,C,G\}$;

(4) 集合 $D=\{B,G,F,A\}$;

(5) 在 D 中依次判断 C 中的点是否连续成对出现,(B,G) 是成对出现的,裁剪结束。

所得的裁剪多边形为$\{B,G,F,A\}$,与预期结果相同。

结论:改进后的裁剪算法可对凸多边形进行正确的裁剪。

由四组实验可见,改进后的算法可实现对多边形裁剪的统一操作。

4 结束语

文中在介绍以 Sutherland-Hodgeman 算法为基础的预处理算法及其实现的基础上,对预处理算法进行了改进,提出的 IPC 算法实现了对凹多边形和凸多边形的正确裁剪。文中详细描述了改进算法及其实现过程,改进后的算法适用于对所有的多边形进行裁剪,算法增加了对顶点数组的排序、删除、取数等操作,以算法的空间复杂度为代价,实现了多边形裁剪的统一化处理。

IPC 算法的提出改进了图形处理器的裁剪能力,可广泛应用于医学成像、地理环境绘制和模式识别等图形处理领域。

参考文献:

[1] Boykov Y, Jolly M P. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d image[C]// Proc of ICCV. [s. l.]: [s. n.], 2001: 105-112.

[2] Rother C, Kolmogorov V, Blake A. Grab cut-interactive foreground extraction using iterated graph cuts[J]. ACM transactions on graphics, 2004, 23(3): 309-314.

[3] 杨国东. 嵌入式图形处理器的研究与实现[D]. 济南: 山东大学, 2010.

[4] 韩明峰. 介绍一种多边形裁剪算法[J]. 计算机应用研究, 1998(5): 87-88.

[5] Wang Jin, Lu Guodong, Peng Qunsheng, et al. Line clipping against polygonal window algorithm based on the multiple virtual boxes rejecting[J]. Journal of Zhejiang university science, 2005(S1): 100-107.

[6] Deng Weiyan, Lu Guodong, Chen Long. New 3D line clipping algorithm against spherical surface window[C]//Proceedings of international technology and innovation conference. [s. l.]: [s. n.], 2006.

[7] 王浩朋. 二维图形的裁剪算法研究与改进[D]. 西安: 西安电子科技大学, 2011.

[8] 王艳娟, 肖刚强, 任洪海. 改进的 Cohen-Sutherland 线段裁剪算法[J]. 现代计算机, 2007(2): 15-16.

(下转第 233 页)

度均值及方差没有按 2.4 节的算法设定阈值,即每个窗口均作了灰度分布标准化处理。

灰度分布标准化的计算实验对比如表 3 所示。算法 GDN1 为直接公式法,即先使用式(7)和式(8)计算窗口的灰度均值及方差,再使用式(9)进行灰度分布标准化;算法 GDN2 和 GDN3 为灰度分布标准化处理时,均使用式(10)进行灰度变换,不同之处是快速计算窗口的灰度均值及方差时,算法 GDN2 使用了算法 GAV2 而算法 GDN3 使用了算法 GAV3。

表 3　掩膜图像窗口灰度分布标准化计算时间对比

窗口尺寸	灰度分布标准化的总计算时间(ms/10 000 个窗口)		
	GDN1	GDN2	GDN3
25 * 25	929.3	656.3	575.7
33 * 33	1 301.8	915.9	803.6
49 * 49	2 113.2	1 480.5	1 298.5

从表 3 可知,算法 GDN2 比 GDN1 的计算时间大大减少,这是因为采用广义积分图像计算掩膜图像窗口的灰度均值及方差,减少了计算开销,同时灰度分布标准化变换时,式(10)相比式(9)计算效率要高些;算法 GDN3 比算法 GDN2 的计算速度又进一步提高,是因为利用了圆形掩膜的对称性减少了窗口扫描范围。因此,掩膜不具对称性时,建议使用算法 GDN2,掩膜具有对称性时,建议参照算法 GDN3 的原理减少计算时间。

表 3 同时也说明 3 种算法的计算速度与窗口的大小均有关。因为在灰度分布标准化变换时,要对窗口的每一个有效像素进行变换处理,所以它们窗口越大,其有效像素越多,变换处理所需的时间越长。

4　结束语

文中应用广义积分图像,实现了一种掩膜图像窗口的快速灰度分布标准化算法。理论分析表明该算法不受掩膜的形状约束,可快速计算掩膜图像窗口的灰度均值和方差,最终实现窗口的快速灰度分布标准化处理。由实现结果可以看出掩膜图像窗口的灰度均值和方差计算速度提高了 1 倍以上,说明了算法的有效性。文中还以圆形掩膜为例,说明了掩膜具有对称性时,算法的计算速度能进一步提高。特别适于多尺度目标检测时需要对大量的掩膜图像窗口计算灰度均值、方差和进行灰度分布标准化处理的应用场合。

参考文献:

[1] 郑永凯,张　凌,董守斌. 图像灰度分布标准化算法研究[J]. 小型微型计算机系统,2002,23(10):1218-1221.

[2] 江月松,王龙奇. 基于积分图像的红外图像降噪去条带方法[J]. 红外,2012,33(7):25-28.

[3] 尤红建,詹芊芊. 组合 Edgeworth 逼近和交叉熵的 SAR 变化检测[J]. 电子与信息学报,2011,33(1):38-42.

[4] 邵　平,杨路明,黄海滨. 基于积分图像的灰度分布标准化快速算法[J]. 计算机应用研究,2007,24(3):277-279.

[5] Ioannou D,Huada W,Laine A F. Circle recognition through a 2D hough transform and radius histogramming[J]. Image and vision computing,1999,17(1):15-26.

[6] Chung Kuoliang, Huang Yonghuai, Shen Shiming, et al. Efficient sampling strategy and refinement strategy for randomized circle detection[J]. Pattern recognition,2012,45(1):252-263.

[7] 钟　凡,莫铭臻,秦学英,等. 基于 WSSD 的不规则图像块快速匹配[J]. 中国图象图形学报,2010,15(3):495-501.

[8] Viola P,Jones M. Rapidobject detection using a boosted cascade of simple features[C]//Proc of IEEE conference on CVPR' 2001. USA:IEEE Computer Society Press,2001:511-518.

[9] 张敏贵. 基于小波和支持向量机的人脸识别方法研究[D]. 西安:西北工业大学,2003.

[10] 邢藏菊. 人脸自动检测方法研究[D]. 北京:中国科学院,2001.

[11] 梁路宏,艾海舟,何克忠,等. 基于多关联模板匹配的人脸检测[J]. 软件学报,2001,12(1):94-102.

[12] 梁路宏,艾海舟,何克忠,等. 基于仿射模板匹配的多角度单人脸定位[J]. 计算机学报,2000,23(6):640-645.

(上接第 228 页)

[9] 王结臣,沈定涛,陈焱明,等. 一种有效的复杂多边形裁剪算法[J]. 武汉大学学报(信息科学版),2010,35(3):369-372.

[10] 苏　诚,韩俊刚. Sutherland-Hodgman 裁剪算法的改进[J]. 西安邮电大学学报,2013(3):80-82.

[11] 邓惠子,韩俊刚,马　超,等. 改进的三维剪裁算法及其硬件设计[J]. 电子科技,2013,26(7):1-4.

[12] 马培华. 直线反走样生成和裁剪的算法改进研究[D]. 南宁:广西大学,2012.

[13] 汪　文,夏礼吉,彭毓峰,等. 基于 VC 的任意不自相交多边型新裁剪算法[J]. 控制工程,2012,19(1):110-113.

[14] 周清平,陈学工. 大规模等值线图任意多边形裁剪算法[J]. 计算机与现代化,2012(4):196-200.

IPC平面裁剪算法的设计与实现

作者： 刘晖， 田泽， 黎小玉， 陈佳， LIU Hui， TIAN Ze， LI Xiao-yu， CHEN Jia
作者单位： 中航工业西安航空计算技术研究所,陕西 西安,710119

刊名： 计算机技术与发展

ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2014(2)