整合的な公差に基づく多角形の頑健な集合演算

乾 正知†

与えられた2つの多角形の集合和や積，差を計算する集合演算は，図形処理における基本的な演算のひとつである．図形処理では，幾何的な諸量の算出や判定を，浮動小数点演算を用いて行う．浮動小数点演算では，数値誤差の発生が不可避なため，得られた結果に矛盾が生じやすく，最悪の場合処理が破綻してしまう．このような矛盾の多くは，近接した幾何要素の順序を取り違えることで生じる．そこである微小な公差値よりも接近している2要素を一致していると見なし，順序の逆転を防止することが経験的に行われている．しかし従来の手法では，公差の導入が新たな矛盾の原因となる場合が多く，問題の解決になっていなかった．多角形の集合和が計算できれば，積や差の計算は容易である．そこで集合和を計算するための位相的な条件を導出し，処理に必要な幾何的判定の手続きを，この条件を満たすように定義した．そしてこの判定を，辺と頂点に定義された公差域に基づいて，整合的に行うように拡張することで，多角形の集合和を安定に計算する手法を開発したので報告する．実際にプログラムを作成し，数値実験により手法の有効性を検証した．

Robust Boolean Set Operations of Polygons with Consistent Tolerances

MASATOMO INUI†

Boolean set operations of polygons are fundamental operations for making an objective shape in geometric modeling. The numerical data describing geometric objects are given using floating point numbers. Since floating point numbers are basically approximations, there may be imprecision that leads to inconsistencies about the represented objects. Most programmers in practice are aware of this imprecision and give a small tolerance value for relaxing various geometric tests. This approach does not always succeed, because it is difficult to control all the implications of approximate tests. In this paper, the authors propose a new tolerance based algorithm for stably realizing Boolean set operations of two polygons. A topological condition for computing Boolean union is derived based on a classification of intersection points of polygon boundaries. Tolerance based incidence tests of vertices and edges are consistently designed so that the test results satisfy the condition. Proposed algorithm is implemented and some computational experiments are demonstrated.

1. はじめに

与えられた2つの多角形の集合和や積，差を計算する集合演算は，CADなどの図形処理における基本的な演算のひとつである．図形処理では，幾何的な諸量の算出や判定を，浮動小数点演算を用いて行う．浮動小数点演算では，数値誤差の発生が不可避なため，得られた結果に矛盾が生じやすく，最悪の場合処理が破綻してしまう．矛盾の多くは，近接した幾何要素の順序を取り違えることが原因で生じる．そこである微小な公差値よりも接近している2要素を一致していると見なし，順序の逆転を防止することが経験的に行われている．機械製品のCADでは，2つの図形を，それらの辺や頂点が一致するように配置し集合演算するなど，ある種の退化した交差を含む図形を処理する場合が多い．公差を用いた幾何的な判定では，意図的に指示された退化した交差を，数値誤差の影響で退化していないと間違って判定することがないので都合が良い．しかし従来の手法では，公差の導入が新たな矛盾の原因となる場合が多く，問題の解決になっていなかった[3),4)]．

公差の定義や一致判定の方法を改良し，矛盾を生じにくくした研究は，これまでにもいくつか知られているが，実用的な処理での利用には問題が多い．Segalは，一致と判定されるたびに公差を大きくすることで，安定な図形処理が実現できることを示している[7)]．この手法では，処理が進むにつれて誤差の許容量が

† 茨城大学工学部
Faculty of Engineering, Ibaraki University

拡大するため，得られる図形の精度が保証できない．Milenkovicは，2頂点が公差に基づいて一致と判定されると，それらを移動させ座標値を完全に一致させることで，矛盾を解消する手法を提案している[6]．この手法では，移動した頂点と他の頂点がさらに一致する可能性があり，処理が連鎖し図形の誤差が拡大してしまう．計算中に発生する誤差の上限を見積もり，完全に信頼できる判定のみを利用して処理を進める方法も報告されているが（たとえば2)），このようなアルゴリズムは複雑になりやすく，設計が困難である．アルゴリズムを見直し，幾何的判定の重複を取り除くことで矛盾の発生を防止する試みもあるが，同様の理由から実際の問題解決での利用は難しい[5),11)]．

杉原は，幾何的な判定の結果よりも，図形処理を進める前提となる位相的な条件の充足を優先することで，安定な処理を実現する手法を提案している[9),10)]．この手法は，退化した交差が扱えないので，CADのためのプログラム作成では利用が難しい．処理を多角形の集合演算に限定すると，必要な幾何的判定は，いくつかの直線の方程式の解法に帰着できるので，有理数演算を用いることで厳密に実現できる[1)]．有理数演算では，処理が進むにつれて分子と分母の桁数が増すため，演算時間と記憶容量が膨大なものになってしまう．杉原は，直線の方程式の係数を，数平面をある精度で分割した格子上の点に制限することで，集合演算を厳密かつ効率的に行う手法を開発した[8)]．これらの手法は，図形を配置する際の入力誤差も厳密に評価する．したがって，退化した交差が生じるように図形を配置しても，演算により得られた図形に，誤差に起因する意図しない微小構造ができてしまう[8)]．

多角形の集合和が計算できれば，積や差の計算は容易である．そこで集合和を計算するための位相的な条件を導出し，処理に必要な幾何的判定の手続きを，この条件を満たすように定義した．そしてこの判定を，辺と頂点に定義された公差域に基づいて，整合的に行うように拡張することで，多角形の集合和を安定に計算する手法を開発したので報告する．提案する手法は，辺や頂点の一致判定に公差を用いる．したがって図形の配置時に微小な誤差が生じても，意図的に指示された退化した交差を確実に判定できる．従来の手法とは異なり，処理中に公差の大きさは一定に保たれるので，要求精度の厳しい図形処理でも問題を生じない．この手法は，有理数演算に基づく厳密な方法などとは違い，数値誤差に起因する処理の破綻を完全に防止することはできない．しかしこの手法に基づいてプログラムを作成し，様々な数値実験を行ったが，処理が破綻する例を見つけることができなかった．したがって本手法は，実用的には十分に頑健と考えられる．

2章では，退化した交差が生じない場合の，多角形の集合和の計算アルゴリズムを概説する．3章では，提案したアルゴリズムに基づいて，集合和を安定に計算するための位相的な条件を導出する．4章では，退化した交差も扱えるように，アルゴリズムを拡張する．5章では，多角形の辺と頂点に整合的な公差域を定義し，必要な幾何的判定の手続きを，公差域に基づいて定義しなおす．6章では，手法の有効性を検証するために行った数値実験の方法と，その結果について述べる．

2. 多角形の集合和の計算

2つの多角形 $\mathcal{P}$ と $\mathcal{P}'$ の集合和 $\mathcal{P} \cup \mathcal{P}'$ を計算するアルゴリズムを，以下に示す．

2.1 多角形の定義

多角形の境界は，ループとよばれる複数の閉曲線からなる．$\mathcal{P}$ の境界を構成する m 個のループを，$l_0, l_1, ..., l_{m-1}$ とする．各ループ l_i を構成する n 個の頂点を反時計周りにたどり，遭遇した頂点を順に $v_{i_0}, v_{i_1}, ..., v_{i_{n-1}}$ と名付ける．ただし添字は n の剰余系とする．次に連続した2頂点 v_{i_k} と $v_{i_{k+1}}$ を，それぞれ始点と終点とする n 本の有向線分 $e_{i_k} = (v_{i_k}, v_{i_{k+1}})$ を考え，これらを多角形の辺とよぶ．定義から，辺の左側が多角形の内部に対応する．同様の手順で，$\mathcal{P}'$ のループ l'_j と，その頂点 v'_{j_l} と辺 e'_{j_l} を定義する．

2.2 処理の概略

アルゴリズムは，以下の4ステップからなる．

Step 1 交点の計算 $\mathcal{P}$ のループと $\mathcal{P}'$ のループの間の全交点を算出する．

Step 2 ループの取り出し 交点を持たない $\mathcal{P}$ と $\mathcal{P}'$ のループのうち，他方の図形の外部に存在するものを取り出す．

Step 3 交点のソート $\mathcal{P}$ と $\mathcal{P}'$ の残りのループ上の交点を，ループの向きに従ってソートする．

Step 4 ループの構成 ソートの結果を利用して $\mathcal{P}$ と $\mathcal{P}'$ のループをたどり，新しい図形のループを構成する．

Step 2で取り出されたループと，Step 4で構成されたループが，集合和に相当する図形のループとなる．

これらのステップは，いずれも数値誤差の影響を受けるため，結果が不確実である．ただしループを取り出すステップでは，数値誤差の影響で，ループが図形外部か否かの判定を間違えても，処理の破綻に至るような問題は生じない．そこで $\mathcal{P}$ と $\mathcal{P}'$ のループはすべて交点を持つものとし，Step 2の説明を省略する．

以下にStep1, 3, 4の詳細を示す．議論を簡単にするために，$\mathcal{P}$ と $\mathcal{P}'$ の間に退化した交差が生じていないこと，言い換えば，ループ間の交点は，すべて2辺の交差により生じていることを仮定する．

2.3 Step 1 交点の計算

$\mathcal{P}$ のループ l_i の辺 e_{i_k} と $\mathcal{P}'$ のループ l'_j の辺 e'_{j_l} の，すべての組み合わせについて交差を調べ，交点を算出する．交点は以下の2種類に分類できる．

in の交点 e_{i_k} の始点が e'_{j_l} の右側にあり，e_{i_k} の終点が e'_{j_l} の左側にある場合の交点．e_{i_k} はこの交点から $\mathcal{P}'$ の内部に入り込むので，これを in の交点とよぶ（**図1** (a) 参照）．

out の交点 e_{i_k} の始点が e'_{j_l} の左側にあり，e_{i_k} の終点が e'_{j_l} の右側にある場合の交点．e_{i_k} はこの交点から $\mathcal{P}'$ の外部へ出るので，これを out の交点とよぶ（図1 (b) 参照）．

図2 の例では，ip_0, ip_2, ip_4 が in の交点，ip_1, ip_3, ip_5 が out の交点である．

2.4 Step 3 交点のソート

$\mathcal{P}$ の各ループ l_i について，l_i 上の全交点を l_i の向きに従って並べ，循環リスト L_i に格納する．具体的には，l_i 上に基準点をとり，その点から交点 ip_i までの距離 $D(ip_i)$ を l_i に沿って測り，昇順にソートする．同様に，$\mathcal{P}'$ の各ループ l'_j について，l'_j 上の全交点を l'_j の向きに従って並べ，循環リスト L'_j に格納する．図2の例では，L_0 と L'_0 は，ともに $ip_0 \to ip_1 \to ip_2 \to ip_3 \to ip_4 \to ip_5$ となる．循環リストなので，ip_5 の次の要素は先頭の ip_0 に戻る．

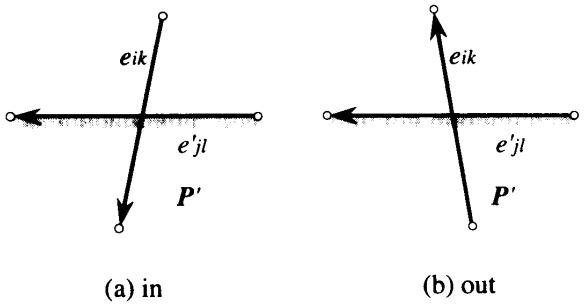


図1 ループ l_i 上の辺 e_{i_k} とループ l'_j 上の辺 e'_{j_l} の交差
Fig. 1 Intersection between an edge e_{i_k} on l_i and another edge e'_{j_l} on l'_j.

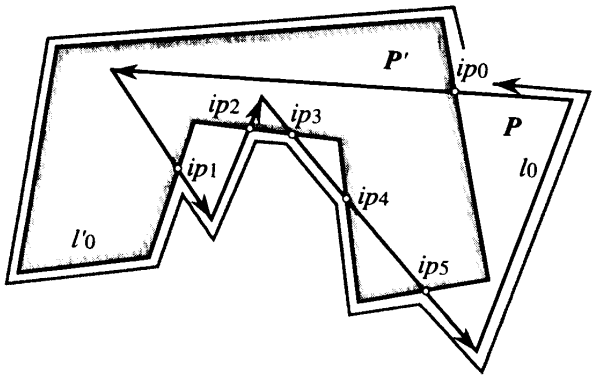


図2 2つの多角形 $\mathcal{P}$ と $\mathcal{P}'$ の集合和の算出
Fig. 2 Computation of Boolean union of two polygons $\mathcal{P}$ and $\mathcal{P}'$.

2.5 Step 4 ループの構成

以下の手順で，循環リストを参照しつつ $\mathcal{P}$ と $\mathcal{P}'$ のループをたどり，新しいループを構成する．

Step 4.1 in の交点を1つ選び，これを出発点 p_s とする．この出発点を変数 ip にセットする．

Step 4.2 ip を含む $\mathcal{P}'$ の辺の向きにしたがい，この辺を含むループ l'_j をたどる．l'_j に対応する循環リスト L'_j において，ip の次にあたる out の交点に遭遇したときには，この点を新たに ip にセットし，Step 4.3 へ処理を移す．

Step 4.3 ip を含む $\mathcal{P}$ の辺の向きにしたがい，この辺を含むループ l_i をたどる．l_i に対応する循環リスト L_i において，ip の次にあたる in の交点に遭遇したときには，この点を新たに ip にセットする．$ip = p_s$ のときには処理を終える．そうでなければ Step 4.2 へ処理を戻す．

ip が出発点 p_s に戻ってきたとき，集合和にあたる図形のループが1つ検出されたことになる．そこで，まだたどられていない in の交点を出発点に選び直し，すべてのループを得るまで，以上の処理を繰り返す．図2には，ip_0 を出発点に選んだときを例に，$\mathcal{P} \cup \mathcal{P}'$ のループを探索する様子を示した．

3. 安定な計算のための位相的な条件

前述のアルゴリズムを安定に実行するには，各ループ上に in の交点と out の交点が同数あり，しかもループに対応する循環リストにおいて，それらが交互に並んでいる必要がある．in の交点と out の交点が同数得られれば，Step 3 でループ上の交点をソートする際に，in の交点と out の交点を別々にソートし，次にそれらを，in の交点と out の交点が交互に現れるように併合することで，この条件を必ず満足させられる．

例として，ループ l_i 上に，in の交点と out の交点が，それぞれ2個得られた場合を考える．in の交点を $ip_{\mathrm{in}_0}, ip_{\mathrm{in}_1}$，out の交点を $ip_{\mathrm{out}_0}, ip_{\mathrm{out}_1}$ とする．l_i 上のある基準点から，これらの交点までの距離をループに沿って測ったところ，$D(ip_{\mathrm{in}_0}) < D(ip_{\mathrm{in}_1})$，$D(ip_{\mathrm{out}_0}) < D(ip_{\mathrm{out}_1})$ であった．そこで，まず in の交点と out の交点をそれぞれ別々にソートし，2つの

リスト $ip_{\mathrm{in}_0} \rightarrow ip_{\mathrm{in}_1}$ と $ip_{\mathrm{out}_0} \rightarrow ip_{\mathrm{out}_1}$ を得る．次にそれらを，in の交点と out の交点が交互に現れるように併合し，たとえば $ip_{\mathrm{in}_0} \rightarrow ip_{\mathrm{out}_0} \rightarrow ip_{\mathrm{in}_1} \rightarrow ip_{\mathrm{out}_1}$ を得る．

併合する際には，in の交点と out の交点のそれぞれのグループについて，基準点から全交点までの距離の和を計算し，値のより小さいほうの交点が先頭になるようにする．すなわち，in の交点 ip_{in_i} の距離 $D(ip_{\mathrm{in}_i})$ と，out の交点 ip_{out_i} の距離 $D(ip_{\mathrm{out}_i})$ に関して以下の計算を行い，d の値が正の場合には in の交点を先頭に，そうでなければ out の交点を先頭に併合する．ただし式中の s は，各交点群の個数である．

$$d = \sum_{i=0}^{s-1} D(ip_{\mathrm{out}_i}) - \sum_{i=0}^{s-1} D(ip_{\mathrm{in}_i})$$

このように交点をソートすることで，幾何的に捩れた自己干渉を生じている多角形の集合演算も安定に実現できる．以上の議論から，各ループ上に in の交点と out の交点が同数存在することが，多角形の集合和を安定に計算するための条件となる．

4. 退化した交差の扱い

退化した交差も扱えるように，アルゴリズムを拡張する．

4.1 退化した交差の判別

前述した多角形 $\mathcal{P}$ と $\mathcal{P}'$ の辺どうしの交差を，以後 X 交差とよぶことにする．すなわち，

X 交差 $\mathcal{P}$ の辺の両端点と $\mathcal{P}'$ の辺の両端点が，互いに他方の辺の左右異なる側に存在する場合．

X 交差に加えて，以下に示す 3 種類の退化した交差を扱うことにする．

V 交差 $\mathcal{P}$ と $\mathcal{P}'$ の頂点が一致する場合．

T1 交差 $\mathcal{P}$ の頂点が，$\mathcal{P}'$ の頂点以外の辺上の場合．

T2 交差 $\mathcal{P}'$ の頂点が，$\mathcal{P}$ の頂点以外の辺上の場合．

以上の 4 種類の交差は，多角形のループ間に生じ得るすべての交差を網羅している．

これらの交差を判別するために，2 頂点 v と v' の一致を判定する述語 *CoincidentVrs* と，頂点 v と辺 e の位置関係を判別する関数 *VrSide* を，以下のように定義する．

predicate *CoincidentVrs*(v, v': 頂点)
  if (v と v' の座標が一致) true を返す;
  else false を返す;
function *VrSide*(v: 頂点; e: 辺)
  if (v は e 上) on を返す;
  else if (v は e を含む直線の左側) left を返す;
  else if (v は e を含む直線の右側) right を返す;
  else on_line を返す;

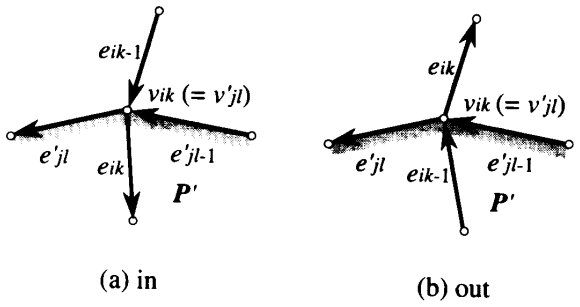


図 3 多角形 $\mathcal{P}$ の頂点 v_{i_k} と多角形 $\mathcal{P}'$ の頂点 v'_{j_l} の一致
Fig. 3 Vertices v_{i_k} of $\mathcal{P}$ and v'_{j_l} of $\mathcal{P}'$ are coincidently positioned.

4.2 V 交差の分類規則

T1 交差は，$\mathcal{P}'$ の辺上の $\mathcal{P}$ の頂点の位置で，その辺を仮想的に分割することで，幾何的に等価な V 交差へ変換できる．同様の手法で，T2 交差も等価な V 交差へ変換できる．そこで以後は，V 交差の扱いのみを考える．

$\mathcal{P}$ の頂点 v_{i_k} と，$\mathcal{P}'$ の頂点 v'_{j_l} が一致しているものとする．v'_{j_l} に接続する 2 辺 $e'_{j_{l-1}}$ と e'_{j_l} の構成するくさび形の領域と，v_{i_k} に接続する 2 辺 $e_{i_{k-1}}$ と e_{i_k} の位置関係を調べる．そして，2.3 節に示した分類規則と同様の考え方にしたがい，以下の条件を満たす頂点 v_{i_k} を選び出し，これらを交点と考えることにする．

in の交点 v_{i_k} を終点とする辺 $e_{i_{k-1}}$ がくさび形の外部にあり，v_{i_k} を始点とする辺 e_{i_k} がくさび形の内部，もしくはくさび形の境界上にあるとき，辺 $e_{i_{k-1}}$ は，頂点 v_{i_k} から多角形 $\mathcal{P}'$ の内部に入り込むと見なせる．そこで v_{i_k} を in の交点と考える（図 **3** (a) 参照）．

out の交点 辺 $e_{i_{k-1}}$ がくさび形の内部，もしくはくさび形の境界上にあり，辺 e_{i_k} がくさび形の外部にあるとき，辺 e_{i_k} は，頂点 v_{i_k} から $\mathcal{P}'$ の外部へ出ていくと見なせる．そこで v_{i_k} を out の交点と考える（図 3 (b) 参照）．

数値誤差の影響が無視できるなら，多角形の各ループにおいて，上述の分類規則により得られた in の交点数と，X 交差により得られた in の交点数の和は，同様にして得られた out の交点数の和と必ず一致する．

4.3 分類手続きの実現

上述の V 交差の分類を実現するために，まず関数 *EdSide* を用意する．関数 *EdSide*(v_{i_k}, v'_{j_l}, e, $e'_{j_{l-1}}$,

e'_{jl}) は，頂点 v_{i_k} と v'_{j_l} が一致しているとき，v_{i_k} を始点または終点とする辺 e が，v'_{j_l} に接続する2辺 $e'_{j_{l-1}}$ と e'_{j_l} の構成するくさび形の，境界上 (on)/外部 (outside)/内部 (inside) のいずれに存在しているのかを判別する．定義中で利用している関数 $OtherVr(v, e)$ は，辺 e とその一端点 v を与えると，e のもう1つの端点を値として返す．

```
function EdSide(v_{i_k}, v'_{j_l}: 頂点; e, e'_{j_{l-1}}, e'_{j_l}: 辺)
  ESideP:= VrSide(OtherVr(v_{i_k}, e), e'_{j_{l-1}});
  ESideN:= VrSide(OtherVr(v_{i_k}, e), e'_{j_l});
  PSideE:= VrSide(OtherVr(v'_{j_l}, e'_{j_{l-1}}), e);
  NSideE:= VrSide(OtherVr(v'_{j_l}, e'_{j_l}), e);
  /* e がくさび形の境界上かどうかの判別 */
  if ((ESideP = on) ∨ (ESideN = on)
      ∨ (PSideE = on) ∨ (NSideE = on))
        on を返す;
  else if (e'_{j_l} の終点が e'_{j_{l-1}} を含む直線の左側)
    /* くさび形は凸形状 */
    if ((ESideP = left) ∧ (ESideN = left))
        inside を返す;
    else
        outside を返す;
  else
    /* くさび形は非凸形状 */
    if ((ESideP = right) ∧ (ESideN = right))
        outside を返す;
    else
        inside を返す;
```

EdSide を用いると，2頂点 v_{i_k} と v'_{j_l} が一致している場合の V 交差の分類規則は，以下のように記述できる．

in の交点 $EdSide(v_{i_k}, v'_{j_l}, e_{i_{k-1}}, e'_{j_{l-1}}, e'_{j_l}) =$ outside $\wedge$ $EdSide(v_{i_k}, v'_{j_l}, e_{i_k}, e'_{j_{l-1}}, e'_{j_l}) \neq$ outside ならば，頂点 v_{i_k} を in の交点と見なす．

out の交点 $EdSide(v_{i_k}, v'_{j_l}, e_{i_{k-1}}, e'_{j_{l-1}}, e'_{j_l}) \neq$ outside $\wedge$ $EdSide(v_{i_k}, v'_{j_l}, e_{i_k}, e'_{j_{l-1}}, e'_{j_l}) =$ outside ならば，頂点 v_{i_k} を out の交点と見なす．

4.4 前処理

関数 *EdSide* は，以下に示す特殊な状態では，判別を正しく行えない．そこで適切な前処理を行い，そのような状態を事前に取り除く．

- くさび形を構成する $\mathcal{P}'$ の辺 $e'_{j_{l-1}}$ もしくは e'_{j_l} の長さがゼロで，その両端点が，V 交差を構成する $\mathcal{P}$ の頂点 v_{i_k} と一致していると，*EdSide* は，交差の状態にかかわらず，つねに on を出力してしまう．そこでこのような辺をあらかじめ除去する．

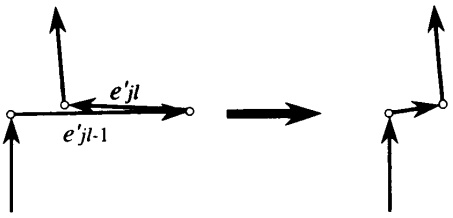


図4 面積ゼロの尖角や谷の除去
Fig. 4 Elimination of acute "spikes" and "valleys".

- 2辺 $e'_{j_{l-1}}$ と e'_{j_l} が重なりあった状態のとき，すなわち $VrSide(OtherVr(v'_{j_l}, e'_{j_{l-1}}), e'_{j_l}) =$ on $\vee$ $VrSide(OtherVr(v'_{j_l}, e'_{j_l}), e'_{j_{l-1}}) =$ on のときには，*EdSide* の判別に誤りが生じる．そこでこのような2辺を，$e'_{j_{l-1}}$ の始点と e'_{j_l} の終点を結ぶ辺に付け換える（**図4** 参照）．これは幅ゼロの尖角や谷の除去に相当するので，多くの場合問題を生じない．

5. 整合的な公差に基づく幾何的判定

これまでの議論では，数値誤差の影響を考慮していなかった．辺の両端点を通過する直線の方程式を，浮動小数点演算を用いて決定すると，数値誤差の影響で，端点と直線の間に微小な隙間が生じてしまう．このような隙間ができると，辺がくさび形領域の境界付近に位置しているとき，辺と領域の位置関係の判定が不安定になりやすい．そこで公差 ε を導入し，幾何要素の微小な位置の差異を無視することで，安定な判定を実現する．

5.1 公差域の定義

多角形を構成する辺と頂点に，互いに整合的な公差域を定義する．

5.1.1 辺の公差域

辺 e_{i_k} を含む直線を中心とする幅 2ε の領域を考え，これを直線の公差域 $\mathcal{L}(e_{i_k}, \varepsilon)$ と定義する．そして，点 p が $\mathcal{L}(e_{i_k}, \varepsilon)$ の左（右）側のとき，「p は e_{i_k} の左（右）側」と判定する．$\mathcal{L}(e_{i_k}, \varepsilon)$ の両端を，e_{i_k} の始点 v_{i_k} を通過し e_{i_k} と直交する直線と，終点 $v_{i_{k+1}}$ を通過し e_{i_k} と直交する直線で切り落とし，長方形領域 $\mathcal{R}(e_{i_k}, \varepsilon)$ を得る．次に v_{i_k} と $v_{i_{k+1}}$ をそれぞれ中心とする半径 ε の円形領域 $\mathcal{O}(v_{i_k}, \varepsilon)$ と $\mathcal{O}(v_{i_{k+1}}, \varepsilon)$ を得る．これら3つの領域の和は，辺 e_{i_k} から距離 ε 以下の平面領域を表している．そこでこれを e_{i_k} の公差域とよび，$\mathcal{Z}(e_{i_k}, \varepsilon)$ と記述する（**図5** 参照）．すなわち，

$$\mathcal{Z}(e_{i_k}, \varepsilon) = \mathcal{R}(e_{i_k}, \varepsilon) \cup \mathcal{O}(v_{i_k}, \varepsilon) \cup \mathcal{O}(v_{i_{k+1}}, \varepsilon)$$

そして，点 p が $p \in \mathcal{Z}(e_{i_k}, \varepsilon)$ のとき，「p は辺 e_{i_k} 上」と判定する．

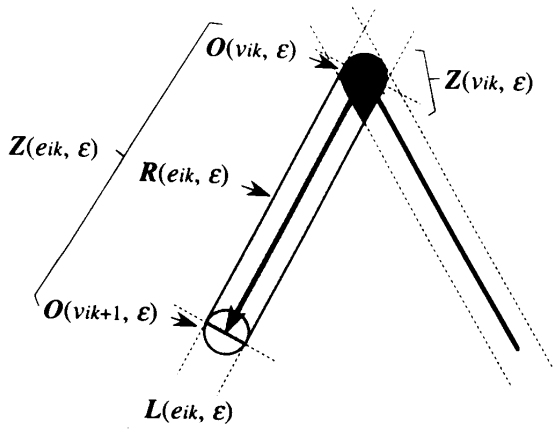


図5 辺 e_{i_k} と頂点 v_{i_k} の公差域の定義
Fig. 5 Tolerance zones of an edge e_{i_k} and a vertex v_{i_k}.

5.1.2 頂点の公差域

これまでの手法では，辺の公差域と同様の考え方に基づいて，頂点 v_{i_k} から距離 ε 以下の領域，すなわち $\mathcal{O}(v_{i_k},\varepsilon)$ を，v_{i_k} の公差域として用いることが一般的であった．この定義では，頂点 v_{i_k} に接続する2辺の公差域の両方に含まれているにもかかわらず，v_{i_k} の公差域には含まれていない点が生じるため，幾何的判定が矛盾しやすい．

多角形の頂点は，接続している2辺の交点と見なすことができる．そこで本研究では，頂点 v_{i_k} の公差域 $\mathcal{Z}(v_{i_k},\varepsilon)$ を，v_{i_k} に接続する2本の辺 $e_{i_{k-1}}$ と e_{i_k} の公差域の積として定義する（図5参照）．すなわち，

$$\mathcal{Z}(v_{i_k},\varepsilon)=\mathcal{Z}(e_{i_{k-1}},\varepsilon)\cap\mathcal{Z}(e_{i_k},\varepsilon)$$

辺の場合と同様に，$p\in\mathcal{Z}(v_{i_k},\varepsilon)$ のとき，「点 p は頂点 v_{i_k} 上」と判定する．

5.2 公差域に基づく幾何的判定

集合演算を安定に実現するためには，X，V，T1，T2の各交差の分類と，in交点とout交点の分類が正しく行えればよい．これらの作業は，2頂点の一致を判定する述語 *CoincidentVrs* と，頂点と辺の位置関係を判別する関数 *VrSide* の結果に基づいて行われる．そこで *CoincidentVrs* と *VrSide* を，公差域に基づく以下の定義に変更する．ただし関数 *EdSide* において，2辺 $e'_{j_{l-1}}$ と e'_{j_l} が構成するくさび形領域が凸形状か否かを判定する処理には，公差を用いない．公差を用いて判定すると，緩やかな凸形状を非凸形状と判定してしまうため，意図とは逆に判定に矛盾が生じやすい．

5.2.1 2頂点の一致判定

2頂点 v と v' の，公差域に基づく一致判定を考える．頂点の公差域は，図5に示したように涙滴形状なので，v は v' の公差域の内部だが v' は v の公差域の外部，すなわち，$v\in\mathcal{Z}(v',\varepsilon)$ だが $v'\notin\mathcal{Z}(v,\varepsilon)$ となる可能性がある．そこで，述語 *CoincidentVrs*(v,v') を以下のように定義し，引数の可換性を実現する．

predicate *CoincidentVrs*$(v,v'$: 頂点$)$
  if $((v\in\mathcal{Z}(v',\varepsilon))\vee(v'\in\mathcal{Z}(v,\varepsilon)))$ true を返す;
  else false を返す;

5.2.2 頂点と辺の位置関係の判別

次に頂点 v と辺 e_{i_k} の位置関係の，公差域に基づく判別を考える．e_{i_k} の始点と終点を，それぞれ v_{i_k} と v_{i_k+1} とする．上述のように *CoincidentVrs* を定義した結果，v は e_{i_k} の公差域の外部だが e_{i_k} の始点もしくは終点とは一致する，すなわち，$v\notin\mathcal{Z}(e_{i_k},\varepsilon)$ だが *CoincidentVrs*$(v,v_{i_k})=$ true $\vee$ *CoincidentVrs*$(v,v_{i_k+1})=$ true の可能性がある．そこで関数 *VrSide* として，以下の定義を採用する．

function *VrSide*$(v$: 頂点; e_{i_k}: 辺$)$
  if $((\mathit{CoincidentVrs}(v,v_{i_k})=\text{true})$
    $\vee\ (\mathit{CoincidentVrs}(v,v_{i_k+1})=\text{true})$
    $\vee\ (v\in\mathcal{R}(e_{i_k},\varepsilon)))$ on を返す;
  else if (v は $\mathcal{L}(e_{i_k},\varepsilon)$ の左側) left を返す;
  else if (v は $\mathcal{L}(e_{i_k},\varepsilon)$ の右側) right を返す;
  else on_line を返す;

5.3 補足的な処理

公差を用いた幾何的判定では，2つの幾何要素が一致と判定されても，それらの形状は微小に異なっている．前述の集合演算アルゴリズムは，このような微小な幾何情報の差異を考慮していないので，そのままでは処理中に問題を生じる場合がある．そこで以下に示す補足的な処理を施す．

5.3.1 T1交差やT2交差の変換

本研究では，T1交差やT2交差は，辺上の頂点の位置で辺を仮想的に分割し，幾何的に等価なV交差に変換し評価する．公差を用いて頂点 v と辺 e の位置関係を判定すると，*VrSide*$(v,e)=$ on であっても，厳密には v は e にのっていない．そこで**図6**に示したように，v から e へ垂線をおろし，その足に仮想的な頂点 p を挿入して e を分割する．p の公差域 $\mathcal{Z}(p,\varepsilon)$ は，p を中心とする半径 ε の円形領域であり，v を必ず含む．したがって *CoincidentVrs*$(p,v)=$ true となるので，v と p はV交差の条件を満たす．

5.3.2 交点の再計算

公差を用いると，V交差と判定された2頂点 v_{i_k} と v'_{j_l} の座標が，厳密には一致しない．そのため，アルゴリズムのStep 4に従って新しいループを構成すると，辺の方程式が変化してしまう．集合演算では，演算前の図形の辺の形状が，演算後も保存されているべきなので，このような変形は望ましくない．そこでループ

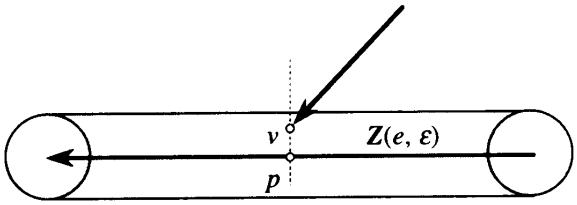


図6 仮想的な頂点 p の挿入による，T1 交差や T2 交差の V 交差への変換

Fig. 6 Insertion of a virtual vertex p to convert a T1 or T2 intersection to a V intersection.

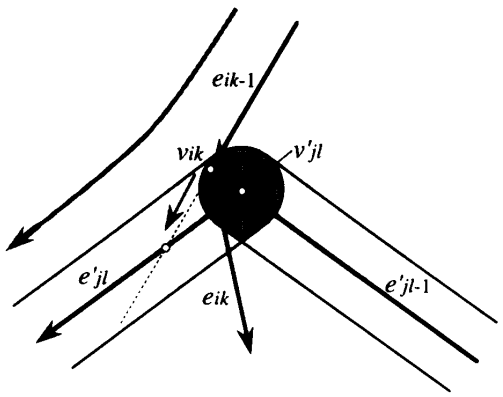


図7 V 交差に対応する交点の座標の再計算

Fig. 7 Re-computation of an intersection point caused by a V intersection.

をたどる際に，交点 v_{i_k} を介してのり移られる 2 辺（図 7 では辺 $e_{i_{k-1}}$ と e'_{j_l}）の交点として v_{i_k} の座標を再計算し，図形に補正を加える．このような 2 辺が平行な場合には，交点が計算できない．また本研究の定義では，頂点の公差域が涙滴形状なので，一致と判定された 2 頂点 v_{i_k} と v'_{j_l} の座標の違いが，無視できないほど大きいこともある．このような場合には，ループをたどる際に，v_{i_k} と v'_{j_l} を微小な辺で連結して処理を進める．

6. 数値実験

提案した手法は，公差域そのものの作成や，公差域に基づく判定に浮動小数点演算を用いるので，理論的には，数値誤差に起因する処理の破綻を防止できない．そこで本手法に基づいて，C 言語で多角形の集合和を計算するプログラムを作成し，数値実験により手法の頑健性を評価した．プログラムの大きさは，説明を省略したループの取り出し処理を含めて，3,000 行ほどである．計算には，倍精度の浮動小数点演算を用いた．また公差の大きさは，$\varepsilon = 1 \times 10^{-7}$ とした．

文献 8) の実験を参考に，半径 1000 と半径 0.001 の同心円に囲まれた領域内に，三角形を，2 つの円周上に頂点が 1 つずつのるようにランダムに 1,000 個生成し，それらの集合和を計算した（図 8 参照）．図 9 には，得られた集合和の様子を示した．また図 10 には，計算前 (a) と計算後 (b) の中心部を，1,000,000 倍に拡大した様子を示した．この例以外にも，公差値や図形を変えて様々な数値実験を行ったが，実験した範囲では，処理が破綻する例を見つけることができなかった．したがって本手法は，実用的には十分に頑健と考えられる．

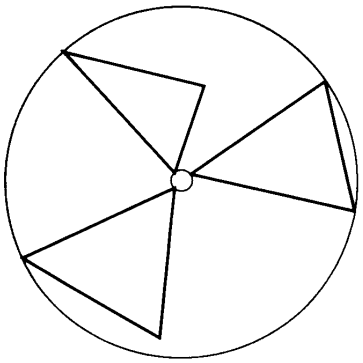


図8 数値実験のための三角形の配置

Fig. 8 Placement of triangles for computational experiments.

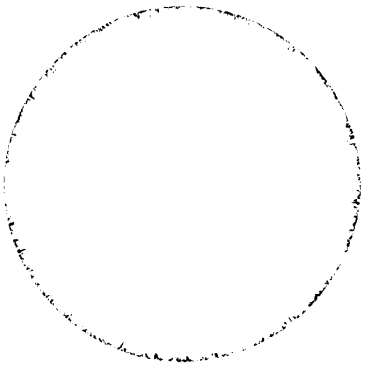
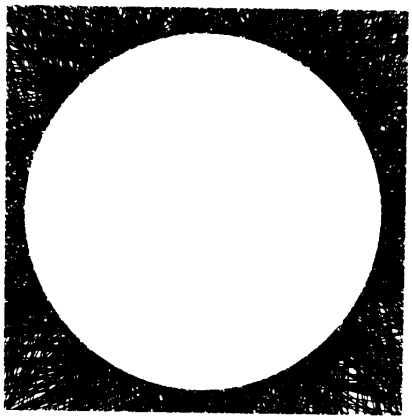


図9 1,000 個の三角形の集合和

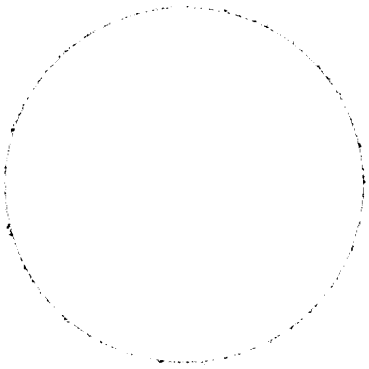
Fig. 9 Boolean union of 1,000 random triangles.

7. まとめと展望

本論文では，集合和を計算するための位相的な条件を導出し，公差に基づく幾何的判定の手続きを，この条件を満たすように定義することで，多角形の集合和を安定に計算する手法について議論した．まず退化した交差が発生しないという仮定のもとで，多角形の集合和を計算するための位相的な条件を導出した．次に退化した交差の判定法を示し，判定された交差を位相的に同等な退化していない交差と見なすことで，導出された条件が，一般の多角形の集合演算にも適用できることを示した．最後に，多角形の辺と頂点に公差域を定義し，処理に必要な幾何的な判定を，公差域に基



(a)



(b)

図 10 中心部の 1,000,000 倍の拡大図
Fig. 10 1,000,000 times magnified views of triangles.

づいて整合的に行うように拡張した．この手法に基づいてプログラムを作成し，数値実験によりその有効性を検証した．

提案した手法は，より複雑な図形処理にも応用できる．現在われわれは，同様の手法を用いた曲線を含む図形の集合演算プログラムを開発しており，満足できる結果を得ている．また本手法の立体処理への応用も検討している．

参 考 文 献

1) Benouamer, M., Michelucci, D. and Peroche, B.: Error-free Boundary Evaluation Using Lazy Rational Arithmetic - A Detailed Implementation, *Proc. 2nd ACM Symp. Solid Modeling and Applications*, pp.115–126 (1993).
2) Guibas, L., Salesin, D. and Stolfi, J.: Epsilon Geometry: Building Robust Algorithms from Imprecise Calculations, *Proc. 5th ACM Annual Symp. Computational Geometry*, pp.208–217 (1989).
3) Hoffmann, C.M.: *Geometric and Solid Modeling: An Introduction*, Morgan Kaufmann (1989).
4) Hoffmann, C.M., Hopcroft, J.E. and Karasick, M.S.: Towards Implementing Robust Geometric Computations, *Proc. 4th ACM Annual Symp. Computational Geometry*, pp.106–117 (1989).
5) Hoffmann, C.M., Hopcroft, J.E. and Karasick, M.S.: Robust Set Operations on Polyhedral Solids, *IEEE Computer Graphics and Applications*, Vol.9, No.6, pp.50–59 (1989).
6) Milenkovic, V.: Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic, *Artif. Intell.*, Vol.37, pp.377–401 (1988).
7) Segal, M.: Using Tolerances to Guarantee Valid Polyhedral Modeling Results, *Computer Graphics*, Vol.24, No.4, pp.105–114 (1990).
8) 杉原厚吉，伊理正夫：計算誤差による暴走の心配のないソリッドモデラの提案，情報処理学会論文誌，Vol.28, pp.962–974 (1987).
9) Sugihara, K.: A Robust and Consistent Algorithm for Intersecting Convex Polyhedra, *Computer Graphics Forum*, Vol.13, No.3, pp.C45–C54 (1994).
10) 杉原厚吉：計算幾何工学，培風館 (1994).
11) Zhu, X., Fang, S. and Brüderlin, B.D.: Obtaining Robust Boolean Set Operations for Manifold Solids by Avoiding and Eliminating Redundancy, *Proc. 2nd ACM Symp. Solid Modeling and Applications*, pp.147–154 (1993).

(平成 7 年 12 月 25 日受付)
(平成 8 年 3 月 12 日採録)

乾　正知（正会員）

昭和 36 年生．昭和 61 年東京大学大学院工学研究科情報工学専攻修士課程修了．昭和 63 年東京大学先端科学技術研究センター助手．平成 3 年東京大学工学部講師．平成 5 年茨城大学工学部助教授．工学博士．形状処理技術とその機械生産自動化への応用に関する研究に従事．精密工学会，IEEE，ACM 各会員．