社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－AL－119 (4)
2008／9／12

凸多面体間の定方向貫通距離計算と交差判定のための平均手間が $\theta(\log^2 n)$のアルゴリズム

仁尾 都
明星大学経済学部

凸多面体 P,Q 間の定方向貫通距離計算問題と交差判定問題に対して P,Q の階層表現を前処理で行うことを前提とした $O(\log^2 n)$アルゴリズムが存在する．しかしその階層表現アルゴリズムのロバスト性には問題があると指摘されている．

本報告では階層表現を用いずに両問題に対処可能な，平均手間が $\theta(\log^2 n)$のアルゴリズムを提案する．

Algorithms for computing the directional penetration distance and intersection detection between two polytopes

Misato Nio
School of Economics, Meisei University

There are already two $O(\log^2 n)$ algorithms for computing the directional penetration distance and intersection detection between two polytopes P,Q. These algorithms depend on the hierarchical representation of P and Q. But it is pointed out that the robustness of the representation is missing. We propose algorithms which do not depend on the hierarchical representations and whose average processing time are $\theta(\log^2 n)$ for both of problems.

1．はじめに

多面体 P,Q($P \cap Q = \phi$）と 3 次元ベクトル d が与えられたとき，P,Q の重なりの有無を判定する問題を交差判定(intersection detection)問題と呼び，重なりが存在する場合，それを無くすために必要な Q の d 方向最小平行移動距離 $\delta_d(P,Q)$を計算する問題を定方向貫通距離(directional penetration distance)計算問題と呼ぶ．そのアルゴリズムの応用は 3 次元シミュレーション，ロボットの経路探索問題などの広域にわたる．本報告は P,Q が凸多面体の場合の $\delta_d(P,Q)$計算と交差判定のアルゴリズムに関わる．

共に頂点数を n とする凸多角形 P,Q に対し，Dobkin らは手間が $O(\log^2 n)$で交差判定と $\delta_d(P,Q)$の計算が共に可能であることを示した[1,2]. そのための前処理として手間が O(nlogn),メモリが O(n)の Hierarchical Presentation（以降,HR と呼ぶ）を P,Q に個別に施すことが必要である．しかしながら HR はロバスト性に問題があり，手間や必要メモリの最悪オーダーの係数が大きいことも指摘されている．このため，両分野の研究が今も続けられている．

以前に筆者らは Minkowski 差 P-Q に手間が $O(n^2 \log n)$ の前処理で必要メモリ量が $O(n^2)$の前処理を施せば，P-Q から得られる単位球面グラフ上の点位置決定問題が手間 O(logn)で解決できる[3]ことを利用して，$\delta_d(P,Q)$の平均的計算手間を殆ど $\theta(\log n)$にできる事を示した[4]．しかしながら P-Q は頂点数が $O(n^2)$

となるため，前処理手間と必要メモリが大きくなるという欠点だけでなく，m 個の凸多面体間の $\delta_d(P,Q)$ 計算を行なう場合，$O(m^2)$ 回の前処理が必要となるという欠点がある．

今回，手間が O(nlogn) で必要メモリ量が O(n)の 2 次元データ構造を作成する前処理を凸多面体 P,Q に対して個別に施せば，平均的手間が $\theta(\log^2 n)$の $\delta_d(P,Q)$計算および交差判定のアルゴリズムを提案する．

2．用語の定義

定義 1. P,Q ：頂点数を n とする凸多面体.

定義 2. P,Q($P\cap Q\neq\phi$)の間の d 方向貫通距離 $\delta_d(P,Q)$：$\inf\{t \mid P\cap Q^{td}=\phi, t\in R^1, t\geqq 0\}$，d は 3 次元単位ベクトル，$Q^{td}=\{p+td \mid q\in Q, t\in R^1\}$.

定義 3. P の法線ベクトル地図 NVM(P)：$\{h(N(p)) \mid p\in\partial P\}$，$\partial P$ は P の境界．N(p)とは $p(\in\partial P)$が所属する P の構成要素(頂点,辺,面)に定義された単位法線ベクトル$(1,\phi,\theta)$をいう．頂点 v_i，v_j を共有し，かつ法線ベクトルをそれぞれ nf_i,nf_j とする面 f_1,f_2 に挟まれた辺 $e_{i,j}$ の法線ベクトル $ne_{i,j}$ はベクトル群 $nf_i(1-t)+nf_j t$，$(0\leqq t\leqq 1)$が与えられる．$h(1,\phi,\theta)=(\phi,\theta)$． $R=\{(\phi,\theta) \mid 0\leqq\phi\leqq 2\pi, 0\leqq\theta\leqq\pi\}$上の矩形領域を分割する曲線グラフ$\{h(ne_{i,j})\}$で囲まれた領域が頂点 v_i の頂点ベクトル nv_i に 1 対 1 に対応する(図 1)．NVM は Normal Vector Map の略[3].

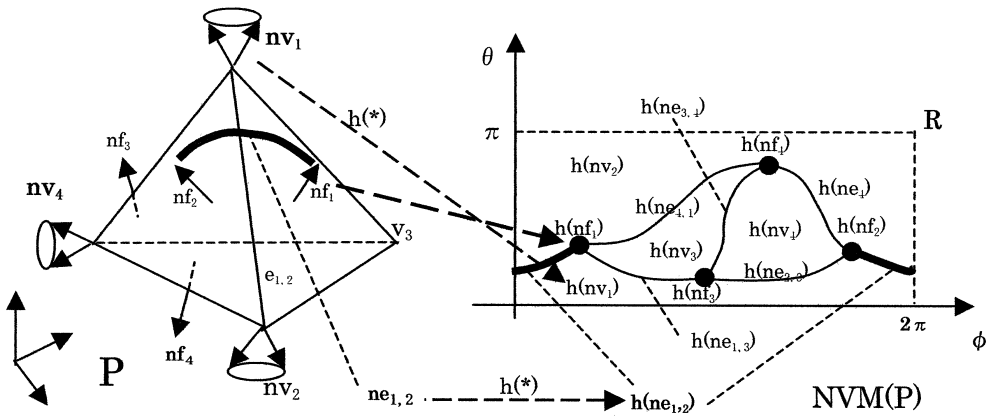

図 1. 4 面体 P の NVM(P)の例

定義 4. M(NVM(P))：分割グラフ NVM(P) に対する点位置決定問題を Sarnak-Tarjan のアルゴリズム[5]で解くためのパーシステントなデータ構造.

定義 5. a・b および a×b：ベクトル a,b 間の内積と外積.

定義 6. v(p,q)：点 p,q を始点終点とするベクトル.

定義 7. S(P,d)：ベクトル d に対する P の支持点.

定義 8. P-Q：$\{p-q \mid p\in P, q\in Q\}$．P,Q 間の Minkowski 差．なお，P-Q は凸で，$S_d(P-Q)=S_d(P)-S_d(P)$.

定義 9. dist(p,q)：$|v(p,q)|$． 点 p,q 間のユークリッド距離.

定義 10.　prj(*,d)：ベクトル d の+∞方向から見た図形*の輪郭図形.

3. δ_d(P,Q)の計算

3.1 アルゴリズム

入力：3 次元直行座標系（原点は g）で定義された凸多面体 P,Q($P \cap Q \neq \phi$).

g を始点とし,方向を 3 次元ベクトル d とする光線 L.

前処理：M(NVM(P))と M(NVM(Q))の作成.

出力：δ_d(P,Q).

1)　$|\forall a \times d| \neq 0$, p_1=S(P-Q,a), p_3=S(P-Q,-a).　　//a はベクトル,凸多角形 prj(P-Q,d)の上の 2 頂点を求める.

2)　if $(v(p_1,p_3)\times d)\cdot v(p_1,g)<0$　then $w=p_1,p_1=p_3,p_3=w$.　　//prj(g,d)が prj($v(p_1,p_3)$,d)の右側にあるようにする.

3)　do　　//L の延長線と交差し, $(v(p_1,p_2)\times v(p_2,p_3))\cdot d>0$ となるような prj(P-Q,d)に内接する$\triangle(p_1,p_2,p_3)$を求める.

3.1)　p_2=S(P-Q,$v(p_1,p_3)\times d$).　　//ベクトル $v(p_1,p_3)\times d$ に対する P-Q の支持点を求める.

3.2)　if $(v(p_1,p_2)\times d)\cdot v(p_1,g)>0$　　//prj(g,d)が prj($v(p_1,p_2)$,d)の右側にあるか.

3.3)　then $p_3=p_2$　　//凸鎖(p_1,p_3)から凸鎖(p_2,p_3)を削除し,残りの凸鎖(p_1,p_2)を凸鎖(p_1,p_3)として再定義.

3.4)　elseif $(v(p_2,p_3)\times d)\cdot v(p_2,g)>0$　　//prj(g,d)が prj($v(p_2,p_3)$,d)の右側にあるか.

then $p_1=p_2$　　//凸鎖(p_1,p_2)を削除し,残りの凸鎖(p_2,p_3)を凸鎖(p_1,p_3)として再定義.

else exit Do.　　//prj($\triangle(p_1,p_2,p_3)$,d)$\supset$prj(g,d)とすることに成功.

4) do　while (p_4=S(P-Q,$v(p_1,p_2)\times v(p_2,p_3)$　&　$(v(p_4,p_1)\times v(p_4,p_2))\cdot v(p_4,p_3)\neq 0$)

//p_4=$\triangle(p_1,p_2,p_3)$の法線ベクトルに対する P-Q の支持点, p_1,p_2,p_3,p_4 の独立性が無くなったら終了.

4.1)　min=dist(h,L$\cap$(△(p_4,p_1,p_2)を含む平面)), j=1.　　//L と交差する$\triangle(p_4,p_j,p_{j+1})$を求める準備.

4.2)　for　i=2 to 3.　　//ただし i=3 のときは i+1=1.

if min>dist(h, (△(p_4,p_i,p_{i+1})を含む平面)$\cap$L) then min=dist(h,H$\cap$L), j=i.

//min を最小とする$\triangle(p_4,p_j,p_{j+1})$を求める.

4.3)　$\triangle(p_4,p_j,p_{j+1})$を$\triangle(p_1,p_2,p_3)$と再定義.

5) print dist(g,$\triangle p_1,p_2,p_3 \cap$L).

3.2 アルゴリズムの正しさと性能評価

〔定理〕頂点数 n の凸多面体 P と 3 次元ベクトル d が与えられたとき，S(P,d)を探索する問題は，前処理手間が O(nlogn)，必要メモリが O(n)で，かつ質問処理が O(logn)で最適に解決可能である[3].

$P \cap Q \neq \phi$ であるので P-Q の座標系の原点 g は P-Q の内点となる. g を光源とし方向 d の光線 L が ∂(P-Q)

と交差する点 r とすれば，dist(g,r)= δ_d(P,Q)となる．r を収束計算で求める．3)では prj(△(p_1,p_2,p_3),d)⊇prj(g,d)であり，P-Q に内接し，d・(v(p_1,p_2)×v(p_2,p_3))>0 となるような△(p_1,p_2,p_3)を求める(図 2)．3.1)で得られる p_2 は prj(P-Q,d)の反時計回りの部分凸鎖(prj(p_1,d),prj(p_3,d))が直線分でない限り，これを 2 分する頂点となり，p_2 は p_1,p_3 のいずれとも異なるので，再定義された凸鎖(prj(p_1,d),prj(p_3,d))が含む頂点数は単調減少する．したがって 3)は収束する．

3.2)から 3.4)までの処理で一度削除された頂点は，再定義された凸鎖(prj(p_1,d),prj(p_3,d))に 2 度と含まれる事はない．また，p_2 は平均的に凸鎖(prj(p_1,d),prj(p_3,d))の中位の頂点が選ばれる．したがって，3)の繰り返しの度に,平均的に半数の頂点が削除される．P-Q の頂点数は O(n^2) であるから，3)のループ回数は O(n^2)であり，3)の平均的なループ回数は θ (log(n^2)),すなわち θ (logn)となる．一方，3.1)の支持点計算手間は定理より O(logn)であるから,結局 3)の手間は θ ($\log^2 n$)となる．

3)では△(p_1,p_2,p_3)の法線ベクトルに対する P-Q の支持点 p_4 を求めると，prj(p_4,d)∈prj(△(p_1,p_2,p_3),d)であるので，L の延長線は 3 つの△(p_4,p_1,p_2),△(p_4,p_2,p_3),△(p_4,p_3,p_1)のいずれかとは必ず交差することが保障される(図 3)．どの 3 角形が L と交差するかを 4.1)から 4.2)で決定する．L が交差する単体の三角形との交点を t とすれば，dist(h,t)は他の三角形を含む 2 つの平面と L との交点と h の距離のいずれよりも小さくなる．4.1)から 4.2)の処理に示すように，単体の三角形を含む 3 平面のうち，L との交点が最も短い平面を見つけるほうが，L と交差する単体の三角形を見つけるより簡便である．p_1,p_2,p_3,p_4 の独立性が無くなるまで t は d の＋方向に前進するので，L の延長線と交差する三角形を△(p_1,p_2,p_3)と再定義して,同じことを繰り返せば，4)のループは必ず収束する．収束したときの dist(g,t)が δ_d(P,Q)に等しい．これらの処理は，L と交差する単体の三角形を見つけるより簡便である．

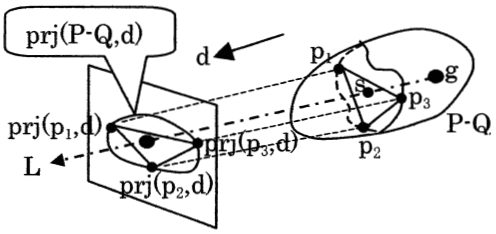

図 2.　prj(g,d)∈prj(△(p_1,p_2,p_3),d)となるような P-Q に内接する△(p_1,p_2,p_3)を求める．

（注：v(p_1,p_3)×d に対する P-Q の支持点 p_4 が求められ，prj(g,d)∈pr(△(p_1,p_2,p_3),d)とすることに成功したときの状態を示している．）

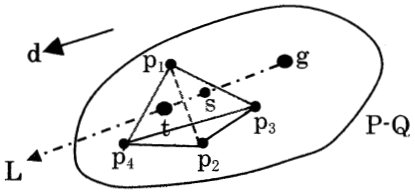

図 3.　prj(g,d)∈prj(△(p_1,p_2,p_3),d)となるような P-Q 内接単体(p_4,p_1,p_2,p_3)を求める．

（注：図では，△(p_1,p_2,p_3)の法線ベクトルに対する P-Q の支持点 p_4 が求められ，L の延長線は内接単体(p_4,p_1,p_2,p_3)と 2 点 s,t で交差している状況が示される．）

P-Q の頂点数は O(n^2)であるので，P-Q の相異なる 3 つの頂点よりなる内接△p_1,p_2,p_3 の組合せ数は O(n^6)となり，4)のループ回数は O(n^6)であり，手間は O(n^6logn)である．一方，P-Q の 3 頂点を通る平面が L の延長線と交差する O(n^6)個の交点を g から ∂ (P-Q)∩L に至る交点に限定し，その方向に並べた交点列を想

定しよう．このとき，交点列の中位の点 u が選ばれると期待できる．g と u に挟まれた交点は以降 2 度と選ばれないので，ループの度に交点列の交数は半減すると期待できる．よって，4)の平均ループ回数は $\theta(\log(n^6))$,すなわち $\theta(\log n)$である．p_4 の計算手間は定理より O(logn)であるから，結局 4)の手間は $\theta(\log^2 n)$ となる．また,前処理は定理より,手間は O(nlogn)で,必要メモリは O(n)である．よって，本アルゴリズムは，手間を $O(n^6 \log n)$, $\theta(\log^2 n)$とし，前処理の手間を O(nlogn),必要メモリを O(n)とする．

4. 交差判定

4.1 アルゴリズム

入力：3 次元直行座標系（原点は g）で定義された凸多面体 P,Q($P \cap Q \neq \phi$).
前処理：M(NVM(P))と M(NVM(Q))の作成.
出力：$P \cap Q = \phi$，または $P \cap Q \neq \phi$.

1) $|\forall a \times d| \neq 0, p_1=S(P\text{-}Q,a), p_3=S(P\text{-}Q,\text{-}a), p_3=S(P\text{-}Q,v(p_1,p_3)\times d$　//a はベクトル，P-Q の内接$\triangle(p_1p_2p_3)$を求める.
2) if h=g　//h は$\triangle(p_1p_2p_3)$の重心.
　then print "$P \cap Q \neq \phi$"
　else d=v(h,g)/|v(h,g).　//以降，h を光源とし，g を通る光線を L とする.
3) if $(v(p_1,p_2)\times v(p_2,p_3))\cdot v(p_1,g)<0$　then $w=p_1,p_1=p_2,p_2=w$.　//g が$\triangle(p_1p_2p_3)$の法線方向にあるようにする.
4) do　while ($p_4=S(P\text{-}Q,v(p_1,p_2)\times v(p_2,p_3)$　&　$(v(p_4,p_1)\times v(p_4,p_2))\cdot v(p_4,p_3)\neq 0$)
　//$p_4=\triangle(p_1,p_2,p_3)$の法線ベクトルに対する P-Q の支持点，$p_1,p_2,p_3,p_4$ の独立性が無くなったら終了.
4.1)　min=dist(h,L∩(△(p_4,p_1,p_2)を含む平面)), j=1, max=min.　//L と交差する$\triangle(p_4,p_j,p_{j+1})$を求める準備.
4.2)　for　i=2 to 3.　//ただし i=3 のときは i+1=1．h から最も遠い交点距離を求める.
　If min>dist(h, H∩L)　then min=dist(h,H∩L), j=i.
　//H とは$\triangle(p_4,p_i,p_{i+1})$を含む平面をいう．min を最小とする$\triangle(p_4,p_j,p_{j+1})$を求める.
　if max<dist(h,H∩L) then max=dist(h,H∩L).　//h から最も遠い交点距離を求める.
4.3)　if min>dist(h,g)
　then print "$P \cap Q \neq \phi$", stop　//収束の途中で，交差と断定できた.
　elseif max<dist(h,g)　then print "$P \cap Q = \phi$", stop.　//収束の途中で，非交差と断定できた.
4.4)　$\triangle(p_4,p_j,p_{j+1})$を$\triangle(p_1,p_2,p_3)$と再定義.
5) if dist(h,L∩(△(p_1,p_2,p_3)を含む平面))≧dist(h,g)
　then print "$P \cap Q \neq \phi$".
　else print "$P \cap Q = \phi$".

4.2 アルゴリズムの正しさと性能評価

凸多面体 P,Q 間の交差判定アルゴリズムは δ_d(P,Q)計算アルゴリズムを修正したものである．δ_d(P,Q)計算では，3 次元座標系の原点 g を光源とし方向 d の光線 L が ∂(P-Q)と交差する点 r を収束計算により探索したが，交差判定では，P-Q の任意の内点 h を光源とし，かつ g を通る光線を L とし，L が初めて ∂(P-Q)と交差する点 r を探索する．dist(h,g)>dist(h,r)であるならば g は P-Q の外点,すなわち P,Q は交差しない，そうでないなら交差すると判定できる．

L と交差する△(p_1,p_2,p_3)を決定するための 1)から 3)までの手間は定理により O(logn) である．△(p_1,p_2,p_3)の法線ベクトルに対応する P-Q の支持点 p_4 を求めると，L と△(p_1,p_2,p_3)は交差するので，L は 3 つの△(p_4,p_1,p_2),△(p_4,p_2,p_3),△(p_4,p_3,p_1)のいずれかとは必ず交差することが保障される．P-Q の原点 g から各 3 角形と L との交点までの最小値 min,最大値 max を求めると，また 4.3)において，min=dist(h,t)>dist(h,g)であれば，g は単体(p_1,p_2,p_3,p_4)の内点であるので，P∩Q≠φと断定できる．逆に max<dist(h,g)であれば，単体の△(p_1,p_2,p_3)を除く 3 つの三角形の法線ベクトル方向の半空間の共通集合内に g が存在する,すなわち P-Q の外側に g が存在するので，P∩Q=φと断定できる．min を実現する三角形が L と交差するから，これを△(p_1,p_2,p_3)と再定義して,4)のループを繰り返す．このときの交点を t とすると，dist(g,t)は単調増加し，p_1,p_2,p_3,p_4 の独立性が無くなる時点で収束する．

本アルゴリズムは，手間を $O(n^6 \log n)$, $\theta(\log^2 n)$とし，前処理の手間を O(nlogn),必要メモリを O(n)とするが,この説明は 3.2 と同様である．

5．結果の検討

5.1 本提案アルゴリズムと GJK 法の最悪手間の比較

本提案アルゴリズムの最悪手間 $O(n^6 \log n)$があまりに大きすぎることに対し懸念が当然起きる．凸多面体間交差判定法として多用されている平均的手間が θ(n)の GJK 法[6]の最悪手間との比較をすることにより，この懸念が大きな問題とならない可能性があることわかる．

GJK 法の各繰り返しでは原点 g から P-Q の内接 4 面体までの最短点ペアとして 4 面体の頂点でも辺でもない，一つの面 H 上の点が常に選ばれるケースを考える．この場合，次の繰り返し時点での内接単体の形状は H の法線方向の新しく求められた支持点が頂点とする 4 面体を常に保ち続けることになり，ループは $O(n^6)$ 回続く．なぜなら頂点数が $O(n^2)$の P-Q の内接 2 次元単体は P-Q の 3 つの頂点を選ぶ組合せ数 $O(n^6)$ 個に等しいからである．一方，各繰り返しの支持点計算は手間を O(n)だけ必要とするから GJK 法の手間は $O(n^7)$ となる．したがって，本提案アルゴリズムの最悪手間は GJK 法の最悪手間よりも小さい．したがって，手間が $O(n^7)$の GJK 法が実用に供されているのと同じ理由で，本提案アルゴリズムが実用に耐えうる可能性は存在する．

5.2 平均手間

本提案交差判定アルゴリズムの平均手間は，実際は θ(logn)であろうと予想する．

5.1 で述べたように，GJK 法のループは $O(n^2)$から $O(n^6)$ 回続く．結局，いずれの場合も GJK 法のループの平均回数 O(logn)となり，全体の平均手間は O(nlogn)となる．しかしながら一般に GJK 法の平均手間が θ (n)であるといわれているが,これは性能測定の実験結果によるものと思われる．その理由として，GJK 法を用いた交差判定では，各収束段階で,常に P-Q の原点 g が支持平面の法線方向にあれば非交差，単体内にあれば交差という早期診断を行う事により，平均収束回数 θ (logn)が θ (1)にまで減少すると思われる．なお，GJK 法により P,Q 間の距離を計算しようとする場合，原点から最も近い P-Q の面を最後まで追い求める必要があるから，その平均手間は θ (nlogn)である．

これと同様に，本提案交差判定アルゴリズムの 4.3)の処理において，収束途中段階で交差判定の結論が得られる確率が高く，ループ回数の平均は θ (1)にまで減少すると期待できる．すなわち，本提案交差判定アルゴリズムの平均手間は，実際は θ (logn)であろうと予想する．手間が θ (n)の GJK 法より本交差判定アルゴリズムの方が高速に交差判定できることは十分期待できる．

3.2 で提案した δ_d(P,Q)計算アルゴリズムでは,L と交差する P-Q の面を最後まで追い求める必要があるから，平均手間は θ ($\log^2 n$)であるのはやむをえない．

5.3 必要メモリ

前処理で生成されるデータ構造のメモリサイズは O(n)であり,しかも 2 次元データであるから，Dobkin らのサイズが O(n)の HR が 3 次元データである事と比較して,非常に有利である．また，HR のメモリサイズのオーダーの係数が非常に大きい事も指摘されている．

5.3 ロバスト性

多面体の頂点数 n が大きくなるにつれて,その HR が扁平な内接 4 面体を含む確率が高まる．このため，HR の生成過程での除去すべき独立頂点の選択時と残された頂点に対する凸包面の作成時における座標計算誤差発生が生じやすくなる．いまだ HR を利用したアルゴリズムの実装に成功した例が無いといわれている．

元来，HR の生成過程で必要とするような図形データ構造の変更操作の繰り返し処理を行う場合，構造のトポロジーに矛盾をきたすことがあるという解決困難な図形処理基本問題が存在する．その主な原因は座標値計算の誤差発生にある．とくに大量の座標計算を必要とする 3 次元図形処理においてはこの問題は大きな障害になっている．本報告アルゴリズムは 3 次元データ構造の変更操作の繰り返し処理を必要としない点で非常に有利である．

しかしながら，本アルゴリズムは凸多面体 P,Q の法線ベクトルマップ NVM(*)に適用する 2 次元点位置決定問題のアルゴリズムのロバスト性に依存する．たとえばスラブ法を採用する場合,具体的には下記の精度上の問題点が残る．

1) スラブ幅が非常に小さい場合の問題．

2) 曲線グラフの始点，終点を(ϕ_s, ϕ_e)，(θ_s, θ_e)とする枝の関数（下記）の計算精度の問題．

$\theta = \tan^{-1}(\sin(\phi_e - \phi_s)/(\sin(\phi_e - \phi)/\tan\theta_s + \sin(\phi - \phi_s)/\tan\theta_e))$

1)の解決のためにはスラブ幅の最小値が最大になるように極座標系の回転方法を決定する問題を解く必要があるが，これは検討課題である．一方，スラブ幅の最小値はグラフ NVM(*)の頂点の x 座標を手間が O(nlogn)のソート処理で行えば極容易に計算できる．そのため許容できる処理時間内で，スラブ幅ができるだけ大きくなるような座標回転方法を試行錯誤により探索する方法も現実的な解決方法も考えられる．P-Q の頂点数が非常に多く，殆どの面が殆ど同じ法線ベクトル成分を持っているような最悪のケースではいかなる回転操作によっても必要なだけのスラブ幅が確保できないことが考えられる．

2)の計算精度の確保に対しても，幅$|\phi_e - \phi_s|$の確保が必要となるので，やはり極座標の回転対策が必要である．

倍精度小数点演算を利用し，極座標系の回転による対策を講じても$|\phi_e - \phi_s| \geqq 10^{-6}$程度の確保を困難にするような形状を持つ凸多面体に対しては本アルゴリズムを適用できないようである．また，P,Q の頂点が北極,南極に近づかないように極座標系を設定する必要があるかもしれない．その他にもどのようなロバスト上の問題が実際に起きるかを実装実験で確認する事が必要である．

6．終わりに

凸多面体 P,Q 間の定方向貫通距離計算問題と交差判定問題に対して，球面分割グラフにおける O(logn)点位置決定アルゴリズムを利用する事により，平均手間が$\theta(\log^2 n)$のアルゴリズムを提案した．

[1] D. Dobkin, D. Kirkpatrick, Determining the separation of preprocessed polyhedra – a unified approach, Proc. of ICLALP '90,400-413,1990, Lecture Notes in Computer Science 443.

[2] D. Dobkin, J. Hershberger, D. Kirkpatrick, S. Suri, Computing the Intersection-Depth of Polyhedra, Algorithmica, Vol. 6, no. 6, 381-392, 1985.

[3] 仁尾都,球面分割グラフにおける点位置決定問題と凸多面体の支持点計算への応用，情報処理学会，2007-AL-114(4),25-31,2007/9/21

[4] 仁尾都，星状多面体の内点判定のための O(logn) アルゴリズム，情報処理学会，2008-AL-117(4),25-32,2008/3/7

[5] N. Sarnak and R. E. Tarjan, Planar point location using persistent search trees, Comm. ACM. 29(1986),669-679

[6] E. G.Gilbert, D. W. Johnson and S. S. Keerthi, Fast Procedure for Computing the Distance Between Complex Objects in Three-Dimensional Space, IEEE Journal of Robotics and Automation, Vol.4, No.2, 193-203,April (1988)