ソフトウェア基礎論　25−15
(1988. 5. 27)

点集合の正方格子度を求めるアルゴリズム

An Algorithm for Fitting a Perfect Grid to a Set of Points

炭野重雄*　　今井　浩*　　内藤史門+
Shigeo SUMINO　　Hiroshi IMAI　　Simon NAITO

* 九州大学工学部情報工学科
Department of Computer Science
and Communication Engineering,
Kyushu University, Fukuoka 812, Japan

+富士電機株式会社
Fuji Electric Co. Ltd.
Hino, Tokyo 191, Japan

あらまし　ピングリッドアレイ型ＬＳＩの位置決め問題に関連して，ほぼ正方格子点上に並んだＮ個の点の集合を，与えられた一定間隔の正確な正方格子でできるだけ良く当てはめる問題がある．本論文では，正方格子を回転・平行移動することによって，点集合の各点と対応する各正方格子点との距離の最大値を最小にする問題を考える．その際の距離の最大値と正方格子の間隔から，その点集合の配列が正方格子にどの程度近いかを示す尺度である正方格子度を定義し，それをＯ(N log N)の手間で求めるアルゴリズムを与える．

Abstract: In connection with the problem of setting a pin-grid-array type LSI to a prited board automatically, there arises a problem of fitting a perfect grid to a set of N distorted grid points. This paper considers such a fitting problem that locates the perfect grid of a given interval to the set of disorted points by rotation and translation with minimizing the maximum distance between each point in the set and its corresponding grid point. Here, the distance between a point and its corresponding grid point is defined not to better Euclidean distance but to be the maximum of the distances from the point to two grid lines intersecting at the grid point. This problem then corresponds to the LSI attachment problem now widely used such that patterns on the printed board are squares arranged according to the grid. We propose an O(N log N)-time algorithm for this problem. This algorithm partly makes use of Davenport-Schinzl sequence of degree two.

1．まえがき

現在，自動車産業などの工業界で盛んに産業用ロボットが使用されている．しかし，ロボットと言っても一般の人が漠然とイメージしている様な自分自身で思考し行動するようなものではなく，ただ単に自動制御技術の延長である場合が多い．

言い替えれば，産業ロボットは，ある決められたところに部品を決められた動作で機械的に配置し（この操作を位置決めと呼ぶ），それに対してある定められた操作をする，といったような完全に決められたことのみを処理する装置である．そのために，何らかの原因で部品が正確に位置決めされないというような故障が生じると，定められた操作ができない，といったようなことも生じかねない．

特に，ＬＳＩなどの超微細な部品を基板上に実装するようなものに対しては，より精密さを要求されるので，定型的な機械的位置決めによるだけでなく，視覚センサを補正機能として用いたことが，より正確な位置決めのために有効である（内藤[3]）．このとき，視覚センサが付加されたことによって，センサから得た視覚的な情報を精密かつ効率よく処理することが問題になる．この問題で，センサよりの視覚情報は幾何的な情報であり，その処理のためには幾何情報の高速処理アルゴリズムを研究する分野である計算幾何学が有用であると思われる．本論文では，このような立場からＬＳＩの位置決め問題に対するアルゴリズムを与える．

この視覚センサを有する位置決め装置の考え方に基づき，内藤らはピンが正方格子状に並んだピングリッドアレイ型のＬＳＩ（直観的には，ピンが正方形のセ

ラミックパッケージの一面に剣山のように植え付けられたＬＳＩ）を，対応する基板上に実装する位置決め装置（ＬＳＩマウンタ）を開発している[3,4,5]．この場合のＬＳＩの位置決めの基準としては，まず，視覚センサから得たピンの一番外側の縦横の列に対して，直交する２本の直線を最小２乗法で近似し，その２直線をもとにピンの外形の正方形を構成する．次に，その構成した正方形と基板上のパタン列の外形の正方形を一致させ，位置決めの基準にするという方法を用いている．また，そのピンの外形の正方形から正方格子を計算し，その構成した正方格子点とピンとの距離からピンがどの程度正方格子点の近傍に存在するか具合いの善し悪しを求める方法を示している．しかし，ピンの外形の正方形を構成する際に内部の点は全く考慮されていないという欠点を持っている．

本論文では，この位置決め問題に関連して，ピングリッドアレイ型ＬＳＩのピンをほぼ正方格子状に並んだＮ個の点集合とみなし，そのＮ点集合が与えられた間隔から構成される正方格子の格子点のどの程度の近傍に存在するかを示す尺度である正方格子度を定義し，それを$O(N \log N)$の手間で求めるアルゴリズムを示す．

この点集合の正方格子度を求める問題は，パタン認識などでの典型的な問題である対応の与えられた２つの点集合間の最適マッチングを求める問題の特殊な場合とみなせ，この特殊な場合の最適解が効率よく求められることを示した点からも，本稿のアルゴリズムは意義があると思われる．

2．点集合の正方格子度の定義

ほぼ正方格子点の近傍に与えられたＮ個の点集合Ｓに対して，与えられた間隔Ｌの正方格子を構成する直線群を考える．ただし，Ｓの各点と各正方格子点には対応付けがなされているとする．ＬＳＩの位置決め問題では，ピンを基板上の指定されたパタンに装着しなければならないので，この仮定は満たされている．

また，Ｓの各点とそれに対応する格子点との距離は，Euclid距離ではなく，点と対応する格子点を構成している直交する２直線を考え，その２直線への垂線の長さのうち大きい方と定める．実際のＬＳＩの表面実装では，パタンは正方格子上に置かれた正方形の配列であり，従ってその場合にはこのように距離を定義することがまさしく現実問題に対応している（図１）．

点集合Ｓの正方格子度Ｇは，Ｓの各点と対応する正方格子点との距離の最大値を，正方格子を平行・回転移動することによって最小化し，その最小化した際の距離の最大値εの２倍を間隔Ｌで割ったもの，すなわちＧ＝２ε／Ｌで定義する．（論文[2]では正方格子度をＧ＝ε／Ｌと定義していたが，ここでの定義に従った方がＧ＝１の場合の意味が下記のように明瞭になるので以降この様に定義する．）

Ｓの全点が与えられた正方格子点に完全に一致する場合，Ｇ＝０になる．また，Ｓに対応するパタンの正方形が重ならず，かつ隙間なく並んだ場合，Ｇ＝１になる．

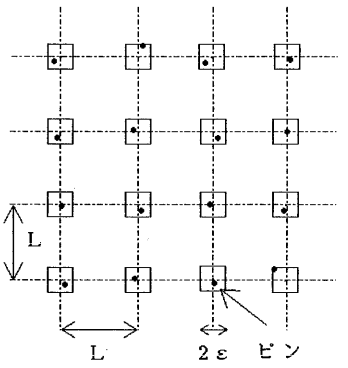

図　１．　正方格子とパタンの配置

3．点集合の正方格子度を求めるアルゴリズムの概略

与えられた点集合Ｓに対して定義を行う．

【定義】　与えられたほぼ正方格子点の近傍に存在する点集合Ｓを，

$S = \{p_{ij}\}_{i=1,\ldots,m,\ j=1,\ldots,n}$

$= \{(x_{ij}, y_{ij})\}_{i=1,\ldots,m,\ j=1,\ldots,n}$,

（ただし，p_{ij}は，間隔Ｌの正方格子直線群のｉ行ｊ列の２直線の交点に対応している．$N = mn$）

また，Ｓを構成するｊ列，ｉ行の複数の点列を，

$A_j = \{(x_{ij}, y_{ij})\}_{i=1,\ldots,m}$　$(j=1,\ldots,n)$,

$B_i = \{(x_{ij}, y_{ij})\}_{j=1,\ldots,n}$　$(i=1,\ldots,m)$,

とし，直交座標系ｘ軸，ｙ軸を原点に関してθ（$0 \leq \theta < 2\pi$）だけ回転した座標系を各々，ｘ(θ)軸，ｙ(θ)軸，p_{ij}の回転座標値を

$$p_{ij}(\theta) = (x_{ij}(\theta), y_{ij}(\theta))$$

（すなわち，$x_{ij}(\theta) = x_{ij}\cos\theta + y_{ij}\sin\theta$,

$y_{ij}(\theta) = -x_{ij}\sin\theta + y_{ij}\cos\theta$）

とする．（図２参照）□

まず，３．１において，B_iの点列だけに対して，その複数点列がどの程度平行であるかを表す平行度を定義し，それを求める方法を与える．次に，３．２において，３．１の結果を利用して，ほぼ直交するA_jとB_iの複数の点列，すなわちＳの正方格子度を求めるアルゴリズムを示す．

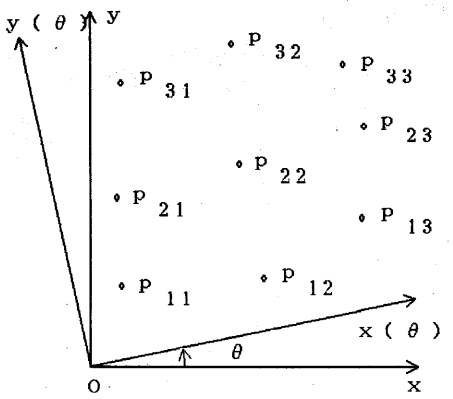

図　2.　点集合Sに対する定義

3.1　複数点列の平行度を求めるアルゴリズム

ほぼ平行直線群の近傍に存在するm個の点列B_iと，それに対応する与えられた一定間隔Lで配置されたm本の平行直線が与えられたとする．このとき，ある点列の点とその点列に対応する直線の距離を，点からその直線に下ろした垂線の長さと定める．そのm個の点列に対して，ある1つの点列の各点とその点列に対応する直線との距離の最大値を求める．その最大値は各点列に対して存在するのでm個存在するがその内の最大値を，平行直線群を平行・回転移動することによって最小化し，その最小化した際の距離の最大値δの2倍をLで割ったものを，その複数点列の平行度Pと定義する．

この場合，B_iに対応する直線群は一定間隔Lで配置されているので，m本の平行直線は適当なy(θ)軸上の座標（0，Y）と角度θを用いて，$y(\theta)=Y+(i-1)L$で表すことができる．

また，iとθを固定したとき，B_iの各点とB_iに対応する直線との距離の最大値を考えると，その最大値を与える可能性があるものは，少なくともy(θ)軸上でのB_iの最大値・最小値を与える点と，B_iに対応する直線すなわち直線$y(\theta)=Y+(i-1)L$との距離のうちで大きい方である．

θを固定したときのy(θ)軸上でのB_iの最大値・最小値を与える点は，各々次のような式で表せる．

$$(B_i\text{の最大値を与える点})=\max_{1\leqq j\leqq n}\{y_{ij}(\theta)\}$$
$$(B_i\text{の最小値を与える点})=\min_{1\leqq j\leqq n}\{y_{ij}(\theta)\}$$

よって，ある固定されたθに対して，iが1からmまで変化したとき，B_iとB_iに対応する直線との距離の最大値は，

$$Y_{max,i}(\theta,Y)\equiv \max_{1\leqq j\leqq n}\{y_{ij}(\theta)\}-\{Y+(i-1)L\}$$
$$Y_{min,i}(\theta,Y)\equiv \{Y+(i-1)L\}-\min_{1\leqq j\leqq n}\{y_{ij}(\theta)\}$$

とする（図3）と，次のような式$f_y(\theta,Y)$で表せる．

$$f_y(\theta,Y)=\max_{1\leqq i\leqq m}[Y_{max,i}(\theta,Y),\ Y_{min,i}(\theta,Y)]$$

上記の$Y_{max,i}(\theta,Y)$と$Y_{min,i}(\theta,Y)$で，本来の距離の定義からは付けるべきの絶対値を付けていないのは，全てのiに対して，

$$\max\{y_{ij}(\theta)\}\geqq\min\{y_{ij}(\theta)\}$$

であるから，2つの式の和をとると，

$$Y_{max,i}(\theta,Y)+Y_{min,i}(\theta,Y)\geqq 0$$

と変形でき，この式より

$$Y_{max,i}(\theta,Y)<0\quad\text{かつ}\quad Y_{min,i}(\theta,Y)<0$$

が成立することはない．また，$f_y(\theta,Y)$はiを固定して$Y_{max,i}(\theta,Y)$と$Y_{min,i}(\theta,Y)$との大きい方を判定し，その判定された結果のm個のものの内の最大値をとる関数と解釈できる．以上より，その大小の判定の段階で負のものは振り落とされてしまい，絶対値を付ける必要がない．

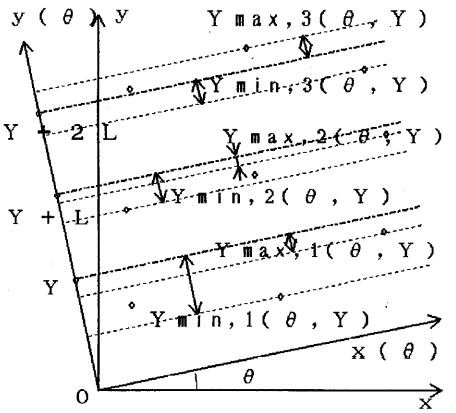

図　3.　$Y_{max,i}(\theta,Y)$，$Y_{min,i}(\theta,Y)$

$f_y(\theta,Y)$を最小化することを考えるとき，θとYを同時に動かして最小化すると議論が複雑になるが，まずθを固定して$f_y(\theta,Y)$をYに関して最小化し，その後の最小化した値をθの関数と見なしてθに関して最小化することを考えると見通しがよい．そこで，

θを固定してYに関する最小化を試みる．

$f_y(\theta, Y)$からYと独立な部分を抜き出し，

$$Y_{ij}(\theta) \equiv y_{ij}(\theta) - (i-1)L,$$

$$Y_{max}(\theta) \equiv \max_{1 \leq i \leq m, 1 \leq j \leq n} \{Y_{ij}(\theta)\},$$

$$Y_{min}(\theta) \equiv \min_{1 \leq i \leq m, 1 \leq j \leq n} \{Y_{ij}(\theta)\},$$

とおくと，

$$f_y(\theta, Y) = \max [\, Y_{max}(\theta) - Y,\ Y - Y_{min}(\theta) \,]$$

となる．この$f_y(\theta, Y)$を固定したθに対して，Yに関して最小化することは容易で，

$$Y = (Y_{max}(\theta) + Y_{min}(\theta)) / 2$$

のとき最小値（これを以降$g_y(\theta)$で表す）をとる：

$$g_y(\theta) = (Y_{max}(\theta) - Y_{min}(\theta)) / 2$$

この$g_y(\theta)$の導出過程は，点列B_iを直線$y(\theta) = Y + (i-1)L$で近似するという観点からではなく，B_iの各点を$y(\theta)$軸の方向に$-(i-1)L$だけ平行移動させて，それに対応する平行直線群を全て一致させる観点から見直すこともできる．（図4）

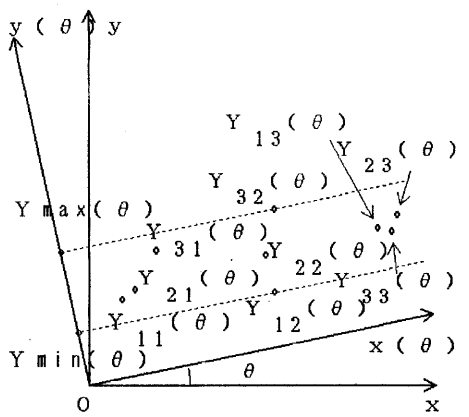

図　4．　$Y_{max}(\theta)$，$Y_{min}(\theta)$

この複数点列B_iの平行度Pは，$g_y(\theta)$をθに関して最小化した値をδをすると，$P = 2\delta / L$になる．

3．2　点集合に対する正方格子度を求めるアルゴリズム

3．1の考察をy（θ）軸に対するB_iについてだけでなく，x（θ）軸に対するA_jについても含めて拡張する．この場合，点と正方格子点の距離は，点から対応する格子点を構成している直交する2直線への垂線の長さの内で大きい方であるから，θを固定した下での正方格子のx（θ），y（θ）軸各々の方向の平行移動は独立に扱え，その固定したθに対するSの各点と対応する格子点との距離の最大値$g(\theta)$は，x（θ）軸に対するn個の点列A_jからなる関数$g_x(\theta)$とy（θ）軸に対するm個の点列B_iからなる関数$g_y(\theta)$のどちらか大きい方になる．

ここで$g_x(\theta)$は$g_y(\theta)$と同様に表せ，

$$X_{ij}(\theta) \equiv x_{ij}(\theta) - (j-1)L,$$

$$X_{max}(\theta) \equiv \max_{1 \leq i \leq m, 1 \leq j \leq n} \{X_{ij}(\theta)\},$$

$$X_{min}(\theta) \equiv \min_{1 \leq i \leq m, 1 \leq j \leq n} \{X_{ij}(\theta)\},$$

とおくと（図5），

$$g_x(\theta) = (X_{max}(\theta) - X_{min}(\theta)) / 2$$

となる．このとき$g(\theta)$は次の式で表せる．

$$g(\theta) = \max [\, g_x(\theta),\ g_y(\theta) \,]$$

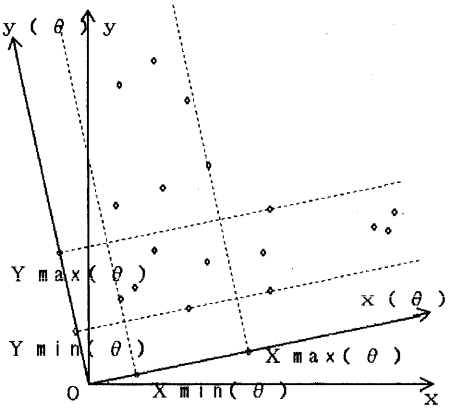

図　5．　$X_{max}(\theta)$，$X_{min}(\theta)$，
$Y_{max}(\theta)$，$Y_{min}(\theta)$

この点集合Sに対する正方格子度Gは，$g(\theta)$をθに関して最小化した値をεとすると，$G = 2\varepsilon / L$となる．以下で$g(\theta)$のθに関する最小化問題を考える．

4．点集合の正方格子度を求めるアルゴリズムの詳細とその解析

点集合の正方格子度を求めるアルゴリズムの計算複雑度，すなわち関数$g(\theta)$の最小化の計算複雑度を示すために，4．1において，θの関数$X_{max}(\theta)$，$X_{min}(\theta)$，$Y_{max}(\theta)$，$Y_{min}(\theta)$を求めるアルゴリズムとその計算複雑度を示し，4．2において，その結果を用いて関数$g(\theta)$の最小化の計算複雑度を示す．

4．1　関数$X_{max}(\theta)$，$X_{min}(\theta)$，$Y_{max}(\theta)$，$Y_{min}(\theta)$を求めるアルゴリズムとその計算複雑度

X，Yおよびmax，minは対称であるので，$X_{max}(\theta)$が求められれば，そのアルゴリズムをXをYに，maxをminに置き換えることによって，他の3つにそのまま利用できる．

$X_{max}(\theta)$は，N個の関数$X_{ij}(\theta)$の最大値を与える関数である（図6参照）ので，$X_{max}(\theta)$を求めるには，定義域［0，2π）の各部分区間内において，i，jが変化したときのN個の$X_{ij}(\theta)$のうちで最大値を与えるある1つの関数を求めて，その部分区間とその区間に対応する関数をリストに格納することによって実現できる．このとき，隣接する部分区間において対応する関数が同じなら，それらを併合することによって，更に簡潔な表現が得られるので，隣接する部分区間では対応する関数は異なるものとする．また，長さ0の部分区間はないものとする．

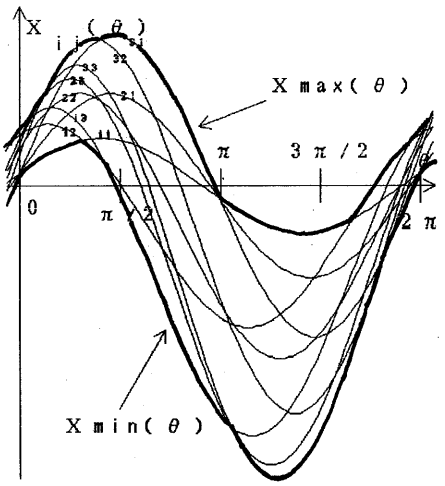

図　6．$X_{max}(\theta)$，$X_{min}(\theta)$のグラフ
（ただし，グラフ内の2桁の数字は$X_{ij}(\theta)$のijを表している）

このようなN個の関数の集合に対してそれらの最大値（最小値）をとる関数を求める問題は，最近，計算幾何学の分野において，Davenport-Schinzel列との関連で活発に研究されている（例えば，Hart,Sharir[1]参照）．このような場合，N個の関数の最大値をとる関数を，上述のように定義域を部分区間に分解し，各部分区間で最大値を与えている元の関数が1つ定まるようにして表現したときの必要となる部分区間の数が問題となる．

$X_{max}(\theta)=max\{X_{ij}(\theta)\}$については，異なる$X_{ij}(\theta)$，$X_{kh}(\theta)$（ただし，$(i,j)\neq(k,h)$）の関数は，次式より定義域［0，$2\pi$）において高々2回しか交わらない．

$$X_{ij}(\theta)-X_{kh}(\theta)=R\sin(\theta+\alpha)-(j-h)L\cdots\cdots\cdots①$$

（ただし，$R=[(x_{ij}-x_{kh})^2+(y_{ij}-y_{kh})^2]^{1/2}$，

$\alpha=\tan^{-1}[(x_{ij}-x_{kh})/(y_{ij}-y_{kh})]$）

よって，このような条件での問題は，2次のDavenport-Schinzel列の問題として扱えり，この場合は容易に必要となる部分区間の数が高々2N－1個であることがわかる（例えば，Hart，Sharir[1]参照）．

このようなような条件を満たす関数は分割統治法により，O(N log N)の手間で求められることが知られている．本稿では，$g(\theta)$，$g_x(\theta)$，$g_y(\theta)$を求めるときに同様な手間が必要となるので，この分割統治法のアルゴリズムを詳しく述べておく．

分割統治法のアルゴリズムは次のように表せる．

アルゴリズム1

1°　N個のθの関数$X_{ij}(\theta)$の集合Fを，ほぼN／2個ずつの2つの集合F_1，F_2に分割する；

2°　集合F_1，F_2各々に対して再帰的にアルゴリズムを適用し，各々の各部分区間とその区間で最大値を与える関数から構成される関数f_1，f_2を求める；

3°　2つの関数f_1とf_2の最大値をとる関数fを求める；

以下では，ＳＴＥＰ３において，2つの関数f_1とf_2をマージして関数fを求める手法とその手間について述べる．f_1のソートされた部分区間のリストを$[\theta_{1,1},\theta_{1,2}),\ldots,[\theta_{1,r},\theta_{1,r+1}),\ldots,$その各区間において対応する最大値を与える関数を$f_{1,1},\ldots,f_{1,r},\ldots$，また同様に，$f_2$のソートされた部分区間のリストを$[\theta_{2,1},\theta_{2,2}),\ldots,[\theta_{2,s},\theta_{2,s+1}),\ldots$，その各区間において対応する最大値を与える関数を$f_{2,1},\ldots,f_{2,s},\ldots$，として与えられたとする．ここで，$[\theta_{1,r},\theta_{1,r+1})$と$[\theta_{2,s},\theta_{2,s+1})$に着目して，この2つの部分区間がマージされてfの部分区間になるとすると，2つの部分区間に対して次の2つの場合が考えられる．

1）2つの部分区間の境界値が交互に値をとる場合，すなわち，

(a) $\theta_{1,r} < \theta_{2,s} < \theta_{1,r+1} < \theta_{2,s+1}$の場合,

(b) $\theta_{2,s} < \theta_{1,r} < \theta_{2,s+1} < \theta_{1,r+1}$の場合,

2) どちらかの部分区間が一方の区間に含まれてしまう場合, すなわち,

(a) $\theta_{1,r} \leqq \theta_{2,s} < \theta_{2,s+1} \leqq \theta_{1,r+1}$の場合,

(b) $\theta_{2,s} \leqq \theta_{1,r} < \theta_{1,r+1} \leqq \theta_{2,s+1}$の場合,

が考えられる.

1) の場合, 一般にその2つの区間に対応する関数$f_{1,r}$と$f_{2,s}$は高々2回しか交わらない. よって, 交点の計算をし, その交点のθ座標をθ_{rs1}, θ_{rs2}とする(ただし, $\theta_{rs1} \leqq \theta_{rs2}$)と, マージされて更に細分化したfの部分区間は,

(a)のとき $[\theta_{2,s},\ \theta_{rs1})$, $[\theta_{rs1},\ \theta_{rs2})$, $[\theta_{rs2},\ \theta_{1,r+1})$,

(b)のとき $[\theta_{1,r},\ \theta_{rs1})$, $[\theta_{rs1},\ \theta_{rs2})$, $[\theta_{rs2},\ \theta_{2,s+1})$ になる.

また, 交点がない場合のfの部分区間は,

(a)のとき $[\theta_{2,s},\ \theta_{1,r+1})$,

(b)のとき $[\theta_{1,r},\ \theta_{2,s+1})$ になる.

その各部分区間において最大値を与える関数を求め, その部分区間と共にfのリストに格納する. この操作は, その新しく細分化された部分区間の数(定数でおさえられる)に比例する手間でできる.

2) の場合, 同様に$f_{1,r}$と$f_{2,s}$は高々2回しか交わらない. よって, 同様に交点の計算をし, その交点のθ座標をθ_{rs1}, θ_{rs2}とすると, マージされて更に細分化した部分区間は,

(a)のとき $[\theta_{2,s},\ \theta_{rs1})$, $[\theta_{rs1},\ \theta_{rs2})$, $[\theta_{rs},\ \theta_{2,s+1})$,

(b)のとき $[\theta_{1,r},\ \theta_{rs1})$, $[\theta_{rs1},\ \theta_{rs2})$, $[\theta_{rs2},\ \theta_{1,r+1})$ になる.

また, 交点がない場合のfの部分区間は,

(a)のとき $[\theta_{2,s},\ \theta_{2,s+1})$,

(b)のとき $[\theta_{1,r},\ \theta_{1,r+1})$ になる.

その各部分区間において最大値を与える関数を求め, その部分区間と共にfのリストに格納する. この操作もその新しく細分化された部分区間の数に比例する手間でできる.

2つの関数$f_{1,r}$と$f_{2,s}$の交点のθ座標θ_{rs}と, その前後の部分区間における最大値を与える関数を求めるには, $f_{1,r} = X_{ij}(\theta)$と$f_{2,s} = X_{kh}(\theta)$, $\theta_{rs} = \theta_{ij,kh}$とすると, $\theta_{ij,kh}$は①式を変形して次の式から求められる(ただし, $(i,j) \neq (k,h)$).

$$\theta_{ij,kh} = \sin^{-1}[(j-h)L/R] - \alpha$$

また, その交点の値$\theta_{ij,kh}$に対して, そのθの値を微小変化させたときの2つの関数の関数値の大小を比較することにより, $\theta_{ij,kh}$を境界値とする前後の部分区間における最大値を与える関数を求めることができる.

交点がない場合にも, 新しく細分化された部分区間内のあるθの値をその区間に対応する2つの関数に代入し, その関数値の大小を比較することによって容易にその部分区間に対応する最大値を与える関数を求めることができる.

よって, ＳＴＥＰ３は部分区間の数の合計に比例するO(N)の線形の手間で求められる.

分割統治法の考え方に基づき, ＳＴＥＰ１で関数の集合Ｆをほぼ2分割していることから, アルゴリズム1の手間は全体でO(N log N)になる.

以上の議論より, $X_{min}(\theta)$, $Y_{max}(\theta)$, $Y_{min}(\theta)$も同様の操作によって, O(N log N)の手間で求められる(図7).

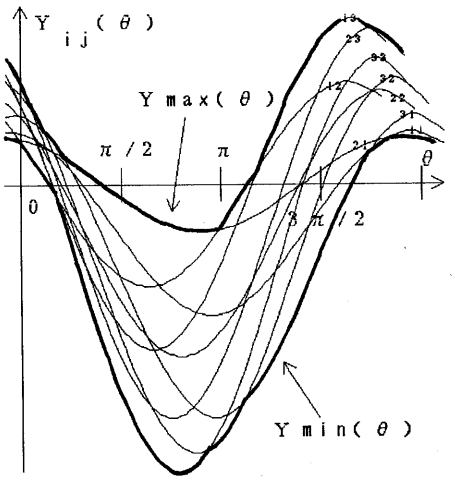

図 7. $Y_{max}(\theta)$, $Y_{min}(\theta)$のグラフ
(ただし, グラフ内の2桁の数字は$Y_{ij}(\theta)$のijを表している)

4.2 関数g(θ)の最小化の計算複雑度

前項で示した$X_{max}(\theta)$, $X_{min}(\theta)$, $Y_{max}(\theta)$, $Y_{min}(\theta)$をO(N log N)の手間で求めるアルゴリズムを用いて, 以下では$g_x(\theta)$, $g_y(\theta)$を求める手法を示す. ただし, xとyは対称であるので, 以下は$g_x(\theta)$を求める手法のみ示す.

すでに$X_{max}(\theta)$と$X_{min}(\theta)$を表すソートされた部分区間とその部分区間に対応する関数の2つリストは与えられているとする.

$$g_x(\theta) = (X_{max}(\theta) - X_{min}(\theta)) / 2$$

であるから，２つのソートされた部分区間をマージし，更に細分化された$g_x(\theta)$のある部分区間において，$X_{max}(\theta)$を表すある１つの関数$X_{ij}(\theta)$と$X_{min}(\theta)$を表すある１つの関数$X_{kh}(\theta)$との差を２等分することによって実現できる．この操作は，４．１で議論したと同様に部分区間の総数の手間，すなわちO(N)の線形の手間で求められる．

よって，同様にして$g_y(\theta)$もO(N)の線形の手間で求められる．

$g(\theta)$は上記のようにして求めた$g_x(\theta)$と$g_y(\theta)$の内で最大値を与える関数であるので，アルゴリズム１のＳＴＥＰ３での手法をそのまま用いることができる．すなわち，$g_x(\theta)$を表すソートされた部分区間のリストと$g_y(\theta)$を表すソートされた部分区間のリストをマージし，その細分化された$g(\theta)$の部分区間で最大値を与える関数を求めることによって表現できる．ここで，マージされる$g_x(\theta)$のある１つの部分区間と$g_y(\theta)$のある１つの部分区間において，各々の区間に対応する２つの関数の交わる回数を考えると，$g_x(\theta)$のある１つの部分区間に対応する関数は，上記で示したように$X_{max}(\theta)$を表すある１つの関数$X_{ij}(\theta)$と$X_{min}(\theta)$を表すある１つの関数$X_{kh}(\theta)$との差を２等分したものであるので，①式を用いて変形でき，正弦関数に定数を足したものになる．同様に，$g_y(\theta)$のある１つの部分区間に対応する関数も正弦関数に定数を足したものになる．よって，マージされる２つの区間に対応する関数の交点の数は，高々２回であり，これより定義域［０，2π）において$g_x(\theta)$と$g_y(\theta)$が交わる回数の総数はO(N)回になる．また，それによって細分化された$g(\theta)$の部分区間の総数はO(N)個である．よって，前項と同様にして$g(\theta)$も部分区間の総数であるO(N)の線形の手間で求められる．

$g(\theta)$をθに関して最小化する際，そのθに関して，次の２つの場合が考えられる．

１）そのθがある部分区間の境界値である場合，
２）そのθがある部分区間に対応する関数の最小値である場合，

が考えられる．

１）の場合，部分区間の総数はO(N)個存在するので，その境界値の総数もO(N)個存在する．よって，$g(\theta)$をθに関して最小化する手間はO(N)である．

２）の場合も，１）の場合と同様に部分区間の総数はO(N)個存在し，その部分区間に対応する関数の総数もO(N)個存在する．ある部分区間に対応する関数は正弦関数に定数を足したものであるので容易にこの条件をチェックすることができる．よって，$g(\theta)$をθに関して最小化する手間はO(N)である．

以上より，次の定理を得る．

【定理】　Ｎ点からなる点集合の正方格子度は，O(N log N)の手間で求められる．□

５．実際の計算時間の短縮についての考察

本章では，実際に$g(\theta)$を最小化する際の計算時間の短縮について考察してみる．

実際の計算時間を短縮する方法として，次の２つが考えられる．

１）点集合Ｓからある条件のもとで点を間引きして，その間引きされた点集合Ｓ’に対して$g(\theta)$を構成し，それを最小化する方法，
２）ある方法を用いて，$g(\theta)$を最小化するθの探索範囲を制限する方法，

が考えられる．

１）の場合，３．１の$f_y(\theta, Y)$から考察すると，$g(\theta)$を構成するある部分区間で最大値を与える点は，少なくともA_jとB_iの凸包の対蹠点対（凸包上の２点に対して，その各点において平行な支持直線を持つとき，その２点を対蹠点対と呼ぶ）であることがいえる．よって，点集合ＳからA_jとB_iの総数である（ｍ＋ｎ）個の点列において凸包を構成し，その凸包の内部に含まれる点を間引きし，その間引きされた点集合Ｓ’に対して４章の議論を適用して$g(\theta)$を最小化する．

しかし，この方法は全ての点列に対して凸包を構成しなければならず，一般にｋ点の凸包を求めるにはO(k log k)の手間がかかることから，結局O(n(m log m) + m(n log n))＝O(N log N)の手間がかかる．また，間引きされた点集合Ｓ’に対して，その点の数だけ４章の手間がかかってしまい，余り効率が良くなったとは言いがたい．

２）の場合，１）の場合でも述べたが$g(\theta)$を構成するある部分区間で最大値を与える点は，少なくともA_jとB_iの凸包の対蹠点対である．この点を考慮して図６，７をみると，$X_{max}(\theta)$と$X_{min}(\theta)$との差すなわち２$g_x(\theta)$と，$Y_{max}(\theta)$と$Y_{min}(\theta)$との差すなわち２$g_y(\theta)$が明かに増大するのは，図２から考えてθを変化したときのx(θ)軸上でのA_{j+1}の最小値とA_jの最大値との大小関係(j=1,...,n-1)とy(θ)軸上でのB_{i+1}の最小値とB_iの最大値との大小関係(i=1,...,m-1)が逆転したときである．このような観察が必ず成り立つのは，点集合の正方格子度が小さい場合であ

る．以下では正方格子度が１より小さいと仮定する．（もとの実際問題ではピングリッドアレイ型ＬＳＩのピンを点集合と見なしているので，ピンの配置が余りずれておらず，この仮定が満たされる）．このとき，容易にわかるように，A_jとA_{j+1}の凸包，B_iとB_{i+1}の凸包は交わらない($i=1,\ldots,m; j=1,\ldots,n$)．

$g(\theta)$を最小化するθの存在範囲は，$2g_x(\theta)$と$2g_y(\theta)$が明かに増大している様な範囲に存在するとは考えられず，この隣接する点列の凸包の大小関係が逆転するθの範囲は除いて良いことになる．

よって，$x(\theta)$軸上でのA_{j+1}の最小値とA_jの最大値との大小関係（図８），かつB_{i+1}の最小値とB_iの最大値との大小関係が保たれているθの範囲を検出して，その部分の範囲だけ$g(\theta)$を構成し，最小化するθを探索すれば良い($i=1,\ldots,m-1, j=1,\ldots,n-1$)．

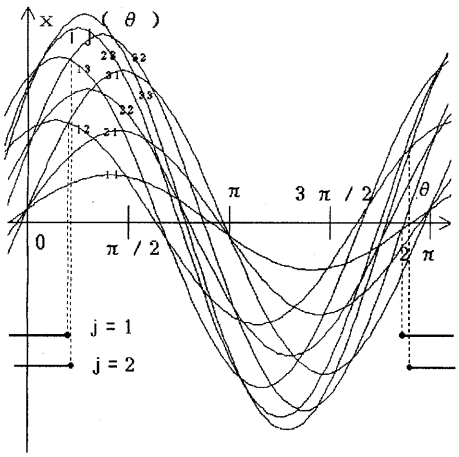

図　８．$x(\theta)$軸上でのA_{j+1}の最小値とA_jの最大値との大小関係が保持されるθの範囲($j=1,2$)
（ただし，グラフ内の２桁の数字は$x_{ij}(\theta)$のijを表している）

そのθの探索範囲を制限するためには，A_jに対して２ｎ個の最大・最小値を与える関数を，B_iに対して２ｍ個の最大・最小値を与える関数を求めると，前章の議論より，１つのA_jに対して$O(m \log m)$の手間，１つのB_iに対して$O(n \log n)$の手間がかかり，全体として，$O(n(m \log m)+m(n \log n)) = O(N \log N)$の手間がかかってしまい，結局，１）の場合と同じ手間がかかってしまう．しかし，θの探索範囲を制限するのに全ての隣接する点列の最大・最小値を与える関数から大小関係を検出しなくても，１組の隣接する点列の最大・最小値を与える関数の大小関係のみ検出するだけで十分に定義域［０，2π）を制限することは可能である．その最大・最小値を与える関数を求めることは前章の議論により$O(\sqrt{N} \log N)$の手間，関数の大小関係が保たれるθの範囲の検出に部分区間の総数の$O(\sqrt{N})$の手間がかかる（ただし，ピングリッドアレイ型ＬＳＩの縦横のピンの数は等しいので，$m=n=\sqrt{N}$としている）．

よって，計算複雑度としては１）の場合と２）の場合とも変わらないが，２）の場合の方が前処理としてθの探索範囲を制限した分，実際の計算時間としては短縮できる．

６．むすび

ほぼ正方格子点の近傍に存在する点集合に対して正方格子度を定義し，それを効率的に求めるアルゴリズムを与えた．このアルゴリズムは，ピングリッドアレイ型ＬＳＩを正方格子上に正方形のパタンが配列されている基板に表面実装する際の位置決めに利用できる．

今後の課題としては，ＬＳＩの実装問題で基板上のパタンが正方形でなく円形の場合やスルーホールの場合，すなわち正方格子度の定義でピンと正方格子点との距離をEuclid距離とした場合に対して効率的なアルゴリズムを構成することである．

謝辞

日頃から御助言，御討論頂いている九州大学工学部情報工学科上林彌彦教授ならびに研究室の皆様に深謝致します．本研究の一部は，（財）井上科学振興財団研究奨励金の援助を受けた．

参考文献

[1]S.Hart and M.Sharir: Nonlinearity of Davenport-Schinzel Sequences and of Generalized Path Compression Schemes. Combinatorica, Vol.6, No.2 (1986), pp.151-177.

[2]今井浩，炭野重雄，内藤史門：点集合の正方格子度判定アルゴリズムについて．情報処理学会第36回（昭和63年前期）全国大会講演論文集，pp.63-64.

[3]内藤史門：産業用ロボットへの視覚センサ応用．センサ技術，Vol.7, No.5 (1987), pp.90-93.

[4]内藤史門：小室明夫：電子部品の位置計測アルゴリズム．昭和60年電子学会全国大会講演論文集，1428 (1985), p.1821.

[5]内藤史門，森俊二，吉田誠：ＬＳＩマウンタの開発．富士時報，Vol.59, No.12 (1986), pp.763-767.

盛光印刷所