グラフィクスとCAD 49－6
（1991. 2. 22）

仮想形状のなぞりアルゴリズム

福井幸男、下条　誠　（製品科学研究所）

あらまし　設計段階において、曲面形状の形状評価を行いながら形状変更を行うためには、容易に面の形状が理解でき、変更も直ちに行える手段が望ましい。そこで、曲面上を手指でなぞって局所的な評価を行うことを最終目的とした基礎実験を行った。曲面の断面を想定すれば、まず平面内の曲線上をなぞることが基本であるから、平面内でのなぞりアルゴリズムを２通り開発した。反力を返すことができる装置をXYレコーダを利用して試作し、なぞり動作を行った。うねりのある形状を計算機上に定義し、その境界上をなぞる実験の結果、視覚で確認するときと同様な傾向の、うねりの周期に依ったうねり振幅の知覚限界が得られた。

Tracing algorithm of vitual shape using force-feedback

Yukio FUKUI and Makoto SHIMOJO
(Industrial Products Research Institute)

Abstract　It is important for the designer to understand easily the shape of free-form surface at the design stage. The most easy way to understand the local characteristics of the shape is to trace the surface by his finger. As this tracing movement almost lie on a plane at a time, it is useful to develop 2-D tracing environment. We have developed two 2-D tracing algorithms, and have reformed a xy-recorder into an input device with force-feedback function. We have made some experiments of tracing virtual curves and obtained the characteristics of sensation by finger movement with force-feedback.

1．はじめに

エンジニアリングワークステーション(EWS)の高性能化にともなって、3次元曲面形状の設計・評価を行う際に、あたかもモックアップを製作して見ているかのように視覚的に現実感の高い画像をEWSで得ることができるようになった[1)]。さらに、正確な曲面評価をEWS上で視覚的に容易に行うために、曲面の特徴を捉える手法が開発されてきた[2)]。EWS上で形状評価をすべて行うことにすれば、形状変更は直ちに行えるので全体の設計がやりやすくなる。しかし3次元的な曲面のうねりなどは、画面上で評価するよりは、手で面をなぞっていく方が直観的に把握しやすい。しかも、表面をなぞる動作は特別な訓練を必要としない。そこで、EWS上で設計した曲面モデルを3次元プロッタや光硬化性樹脂加工によって実際のモデルとして製作することがよく行われる[3)]。実際のモデルを製作するときは、EWS上での画像生成と異なり、ある程度の時間が必要であるため、直ちに何度も形状設計変更することが困難となる。これでは設計者の、形状設計に関する試行錯誤のペースが乱れやすくなる。そこで設計者が、EWS上で生成した形状をその表面に沿って、任意の方向に、手に反力が返って来るような方法で、手指でなぞることができるならば、設計変更しても容易にその結果を確認することができる。この場合、なぞる形状は3次元曲面と、それと交差する平面との交線（平面内の曲線）に限定してもよい。なぜならば、交差する平面自体の方向を任意に変えてなぞる動作を繰り返すことにより、3次元形状の把握は可能となるからである。したがって、なぞって経常を認識するために必要な基本的な機能として、2次元形状の曲線境界にそってなぞっていくための機構とアルゴリズムがある。そこで本報告では、2次元曲線に沿ってなぞるアルゴリズムを示し、それを使った実験結果を述べる。

2．なぞり動作アルゴリズム

曲線境界に沿ってカーソルを動かすアルゴリズムにおいて、曲面が定義されていないときには、カーソルは通常のマウスに対するカーソルと同じように、移動させようと作用が働く方向にそのまま動いてよい。しかし作用する方向が定義された曲面を横切る位置かつ方向の場合は、作用する方向のうち接線成分のみを取り出し、カーソルは形状の境界に接しながら接線方向に沿って動かなければならない。形状表面に沿ってカーソルが動くとき、本来曲線形状のどちら側にカーソルがあるのかを明確にしておく必要がある。そのための方法として、(1)形状境界からわずかに離れた距離を保ってカーソルが境界に沿って動くようにすることで、空間的に本来カーソルが存在している領域を保持する、(2)正確に境界上に沿って動くことを許し、別のフラグで本来の存在領域を保持する、の2通りのアルゴリズムを作成した。なお、曲線形状は短い折れ線の連なりで近似されているものとし、各交点は順に$P_i(x_i, y_i)$と名付けられているものとする。

2.1 アルゴリズム1

図1において、形状境界をハッチングを施した折れ線形状で示し、現在のカーソル位置を$C(x_c, y_c)$で表わし、カーソルに作用する方向を作用ベクトルfとし、その座標を$F(x_f, y_f)$とする。作用ベクトルfとは、もし、定義された形状がなければ次のカーソル位置はfの終点Fとなるようなベクトルで、カーソルの次の位置への移動ベクトルを意味する。図1では形状境界を構成する一つの線分（形状素）$\overline{P_iP_{i+1}}$と$\overline{CF}$が交わる。そこで、fを$\overline{P_iP_{i+1}}$に平行な成分f_tと垂直な成分f_nに分け、さらにf_nの絶対値を縮小してベクトルの終点が線分$\overline{P_iP_{i+1}}$からε（一定な正の微小量）の

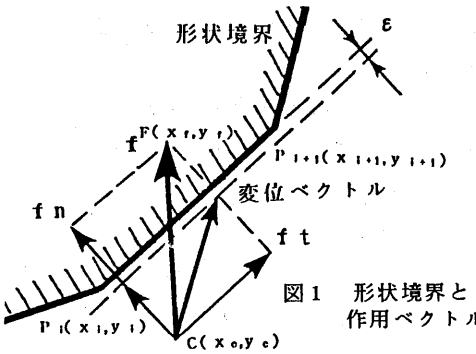

図1　形状境界と
作用ベクトルの交差

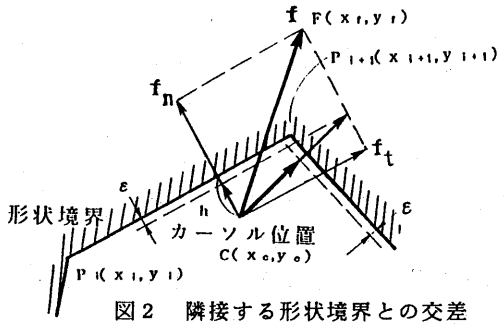

図2　隣接する形状境界との交差

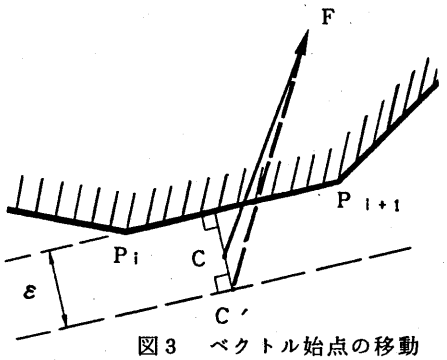

図3　ベクトル始点の移動

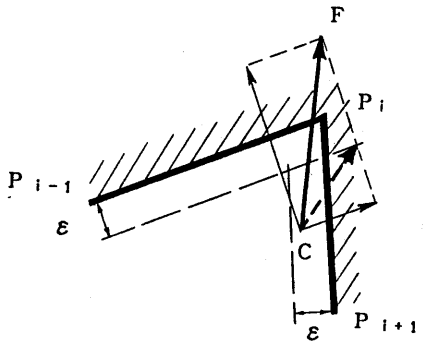

図4　アルゴリズム1

距離離れているようなベクトルf_{n1}とする。そして、f_tとf_{n1}とを合成したベクトルf_1を実際の移動ベクトルとし、その終点を次のカーソル位置とする。

基本的には図1に示したように本来の移動ベクトルfを形状を横切らないようなベクトルf_1に変更してカーソルを移動させるだけであるが、形状境界の位置によって、いくつかの派生的な処理が必要になる。**図2**には、$\overline{P_iP_{i+1}}$の接線に沿うように変更したベクトルf_1がさらに隣接する線分$\overline{P_{i+1}P_{i+2}}$と交わる場合である。このとき、この$\overline{P_{i+1}P_{i+2}}$より距離$\varepsilon$だけカーソルの存在する側の平行な線と$f_1$との交点までに$f_1$の終点を縮めたものが移動ベクトル$f_2$となる。また、現在のカーソル位置Cが既に線分$\overline{P_iP_{i+1}}$から$\varepsilon$未満の距離の位置にあるとき（**図3**参照）、C点を移動させてC'点とし、その後図1または、さらに図2に示す処理をしなければならない。あるいは、カーソル位置Cが交差する相手の線分$\overline{P_{i-1}P_i}$とはε以上はなれていても、隣接する線分$\overline{P_{i+1}P_{i+2}}$からの距離が$\varepsilon$未満の場合があるが（**図4**参照）、このときは移動ベクトルは**0**とする（付録参照）。

2.2　アルゴリズム2

アルゴリズム1では、カーソルの存在領域の情報を空間的に保持していた。そのために、交差計算部で現在のカーソル位置をシフトさせるなどの複雑な処理が必要であった。そこで、カーソルを形状境界に沿って動かすとき、境界上を動かすことにし、もともとカーソルが存在していた空間情報は別途「内外フラグ」で保持することにすれば、アルゴリズムが簡略化する。以下にこれをアルゴリズム2として説明する[4]。

図5に、移動ベクトルの生成状態を示す。現在のカーソル位置と作用ベクトルとの相対的な位置関係から、次の3つの場合に分けて考察する。

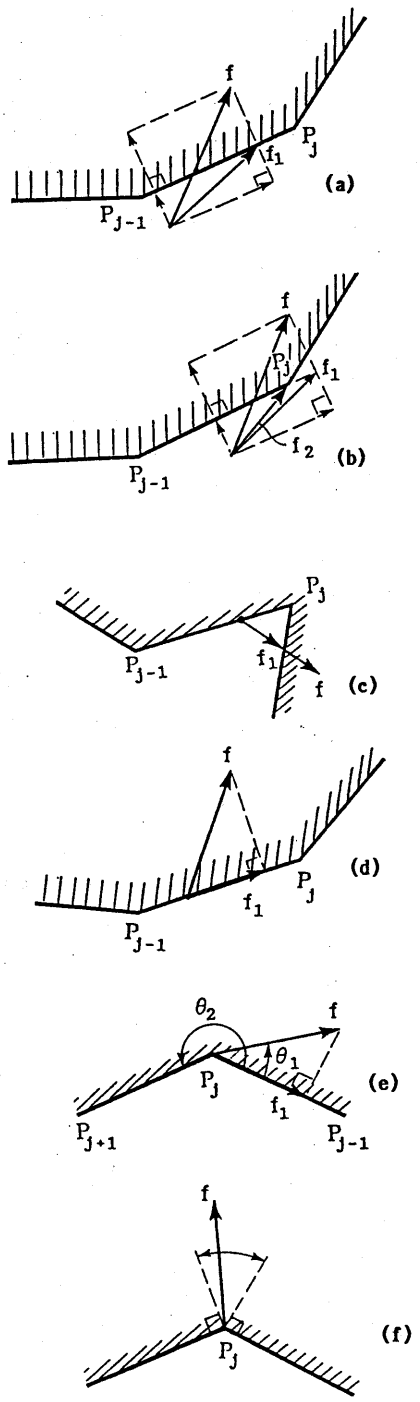

図5　アルゴリズム2

(a)作用ベクトルfと形状素$\overline{P_{j-1}P_j}$が一般の位置で交差する場合

図5(a)は一般の交差状態で作用ベクトルfが、移動ベクトルf_1に変換される過程を示す。ただし**同図(b)**に示すように、生成された移動ベクトルf_1が該当する形状素$\overline{P_{j-1}P_j}$の線分の外部に出た場合は、境界点P_jまで短縮されて移動ベクトルはf_2となる。このとき、カーソルが形状境界のどちら側にあるかを内外フラグとして保存しておく。

(b)作用ベクトルfの始点が境界素$\overline{P_{j-1}P_j}$上に存在する場合

まずfの方向が形状の内部か外部のどちら側を向いているかを内外フラグを参照して判定する。そして、最初に作用ベクトルが形状と交差したときのカーソルの位置と同じ側にあれば、隣接する境界素と交わらないことを確認してカーソルはfの方向に変位させる。このとき、隣接する境界素と交われればその交点までに移動ベクトルを短縮させてf_1とする（**同図(c)**参照）。最初に接した側と反対側にfがあれば、境界素に沿って移動できる成分があるか確認し、同図(a)と同様に移動ベクトルf_1を生成しその方向に変位させる（**同図(d)**参照）。

(c)作用ベクトルfの始点が境界素の交点P_jに一致する場合

形状素と作用ベクトルとの相対角度から形状内外の判定を行って、移動ベクトルを決定する（**同図(e)**参照）。滑りベクトルが他の形状素と交差する場合は、図(d)の場合と同様に終点を交点までに短縮させる。また、いずれの方向にも移動できなくて$f_1=0$の場合もある（**同図(f)**参照）。

3．動作実験

上記アルゴリズムに従ったカーソルの動きに対応した動きを生じる2次元アクチュエータをXYレコーダを利用して試作した。作用ベクトルとして、

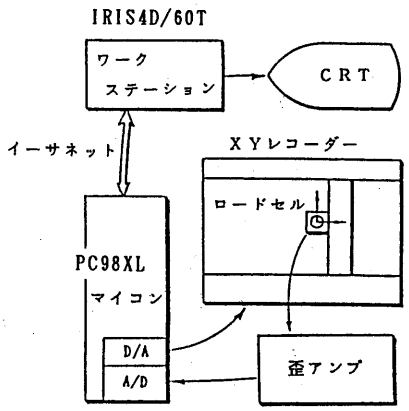

図6　システム構成

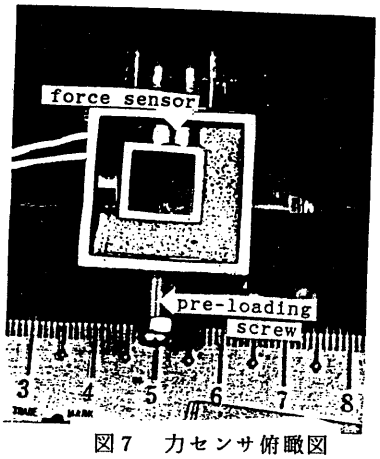

図7　力センサ俯瞰図

図8　うねり振幅を変化させた仮想形状

操作者の手指が与える力を検出して用いた。アルゴリズム1は、アルゴリズム2より約2倍ほど演算時間がかかることがわかった。そこで、以下の実験はすべて、アルゴリズム2を用いた。

3.1　システム構成

図6にシステムのブロック図を示す。XYレコータのヘッド部の動きはカーソルの動きと一致するように制御する。ヘッド部に2次元力センサ（**図7**参照）を装着している。操作者はセンサ内部のくぼみに手指を軽く乗せ、XYレコーダの動き得る平面内の方向に力を加えると、その力が検出されて作用ベクトルに変換される。カーソルが形状境界に交差する場合、その方向にはカーソルは動かないため、操作者が手指を乗せているセンサもうごかなくなり、手指には反力が返って来る。力を形状境界の法線方向よりあたかも透明物体に触れたときのように力を加えてもその方向には動かなくなる感覚を味わうことができる。

3.2　仮想形状

形状境界を予め定義しておけば、その境界に沿ってなぞる動作が

2次元形状の境界曲線を次のようにパラメータ曲線（x(t),y(t)）で与え、周期p_0、振幅a_0の正弦状のうねりがベースとなる曲率半径r_0の円周上に重畳された形状とする。このように定義された形状でうねりの振幅を変えた状態の例を**図8**に示す。

$$x(t)=a(t)\cos t$$

$$y(t)=a(t)\sin t$$

ここで

$$a(t)=a_0\cos\left(\frac{2\pi r_0}{p_0}t\right)+r_0$$

$$0\leqq t<\pi$$

3.3 なぞり実験

手指で軽く押しながら、XYレコーダのヘッド部に装着した力センサをヘッド部が動く平面内の力を与えると、その方向に手指と一緒にヘッド部は動く。そこで、図6に示すようなうねりのある境界をもつ形状を定義し、外形をこするように手指をセンサに押し付けながら動かせば、その形状境界に沿ってうねりながら手指が動く。うねりのピッチやベースとなる曲線の曲率を一定にして、振幅を0から次第に大きくしていったとき、どの程度の振幅になった段階でうねりが反力として認識できるかその検出限界となる振幅を調べた。2人の被験者で繰り返し実験を行い、その検出限界となる振幅の平均と標準偏差を求めた。自由に動ける動作範囲はx，y方向とも約±10cmである。手指に伝わる位置の変化と力の感覚だけの振幅の検出限界を求めたものと、約1m離れた位置にある19インチディスプレイ上で同じ大きさのうねりのある形状を表示した状態を視覚で見ながら、同じくうねりの振幅を変化させていったときの検出限界を求めたデータの例を図9に示す。いずれも同様な傾向を示し、うねりのピッチが大きい場合は、視覚の方が手指による力・運動感覚による認識より優れていることがわかった。しかし、最終的な目的である、微妙な形状変化の実験をするためには、さらに装置の精度をあげる必要がある。

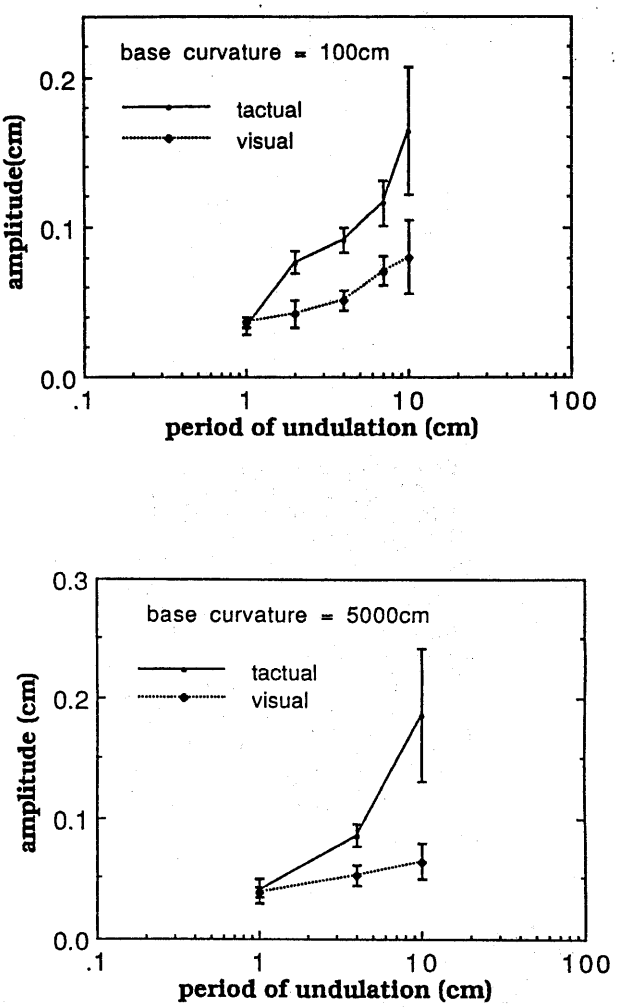

図9　うねり知覚限界測定結果例

4．おわりに

なぞって形状を認識するという最終目的にはまだ至っていないが、実際の形状ではなく、計算機上に定義された仮想形状をさわる感触が得られ、さらに精度と自由度をあげることによって、設計段階で形状評価に使えるインタフェースになるものと期待している。

参考文献

1)畠中兼司：家電プロダクツにおけるビジュアライゼーション技術とCAD/CAM事例、PIXEL、101、pp.89-93 (1991)

2)補坂衛、東正毅、久志本琢也：自由曲面の特徴および評価に関係する諸量および表示、グラフィックスとCADシンポジウム論文集、pp.45-53 (1989)

3)斉藤勝政：光造形法による3次元プロッタ、PIXEL、97、pp.106-109 (1990)

4)福井、下条：なぞり動作による仮想形状のうねり検出特性、第6回ヒューマンインタフェースシンポジウム論文集、pp.31-34 (1990)

付　録　（アルゴリズム1の説明）

カーソルに働く移動させようとする作用ベクトルを$\mathbf{f}$、実際に移動するための移動ベクトルを$\mathbf{f}_1$または、$\mathbf{f}_2$とする。このとき、$\mathbf{f}$が線分$\overline{P_{i-1}P_i}$と交わる場合に　移動ベクトルを次のようにして求める。

もし$\mathbf{f}$が線分$\overline{P_iP_{i+1}}$とも交わるとき、Cから各線分との交点までの距離の短い方の線分を改めて$\overline{P_{i-1}P_i}$とする。

次に$\mathbf{f}$の線分$\overline{P_{i-1}P_i}$に対する接線成分$\mathbf{f}_t$、法線成分$\mathbf{f}_n$、法線線分を縮小してεだけ線分から離れた終点を持つようにしたベクトル$\mathbf{f}_{n1}$を求める。

$$\mathbf{f}_t=(\mathbf{f}\cdot\mathbf{e})\,\mathbf{e}$$

ただし、$\mathbf{e}$は線分$\overline{P_{i-1}P_i}$に沿った単位ベクトル

$$\mathbf{f}_n=\mathbf{f}-\mathbf{f}_t$$

$$h=|h_1|/\sqrt{a_0^2+b_0^2},$$

ただし、hはCから$\overline{P_{i-1}P_i}$までの距離、ここで

$$h_1=a_0x_c+b_0y_c+c_0$$

$$a_0=y_i-y_{i-1}$$

$$b_0=x_{i-1}-x_i$$

$$c_0=x_iy_{i-1}-x_{i-1}y_i$$

if　$h>2\varepsilon$　then

$$\mathbf{f}_{n1}=(h-\varepsilon)\,\mathbf{f}_n/|\mathbf{f}_n|$$

else　$\mathbf{f}_{n1}=0$

次に、カーソル点Cが$P_{i-1}P_i$からε離れるようにシフトさせる。ただし、移動に伴って隣接する$\overline{P_iP_{i+1}}$、$\overline{P_{i-2}P_{i-1}}$ともとの作用ベクトルが交わらないことを確認する。もし交われば、C点のシフトを行わない。

C（x_cy_c）から$\overline{P_{i-1}P_i}$におろした垂線の足をH（x_hy_h）を求めると、

$$x_h=\frac{b_0(b_0x_c-a_0y_c)-a_0c_0}{a_0^2+b_0^2}$$

$$y_h=\frac{-a_0(b_0x_c-a_0y_c)-b_0c_0}{a_0^2+b_0^2}$$

となる。$\overline{P_{i-1}P_i}$を含む直線ℓ_0および、それよりカーソル側にε離れた平行な直線ℓ_1の方程式はそれぞれ、

$$\ell_0:a_0+b_0+c_0=0$$

$$\ell_1:a_0+b_0+c_1=0$$

ここで、$c_1=c_0-\varepsilon c\sqrt{a_0^2+b_0^2}$

である。ただし、向きフラグcは次で求める。

if　$h_1>0$　then　$c=1$

else if　$h_1<0$　then　$c=-1$

else if　$a_0x_f+b_0y_f+c_0<0$

then　$c=1$

else　$c=-1$

このとき、H（または、C）から直線ℓ_1におろした垂線の足をT（x_ty_t）とすると、

$$x_t=\frac{b_0(b_0x_h-a_0y_h)-a_0c_1}{a_0^2+b_0^2}$$

$$y_t = \frac{-a_0(b_0x_h - a_0y_h) - b_0c_1}{a_0^2 + b_0^2}$$

CをTにシフトさせるときに形状境界を横切らないことを次で確認する。

if 線分CTが線分$\overline{P_iP_{i+1}}$と交わらない
かつ
線分CTが線分$\overline{P_{i-2}P_{i-1}}$と交わらない
ならば、
(x_cy_c)に(x_ty_t)を代入する
endif

移動ベクトルf_1は

$$f_1 = f_t + f_{n1}$$

となる。したがって、

$$(x_fy_f) = (x_cy_c) + f_1$$

移動ベクトルf_1が隣接する形状境界と交差していないか以下で調べる。

隣接する形状境界の線分$\overline{P_iP_{i+1}}$の方程式を改めて

$$a_0x + b_0y + c_0 = 0$$

とする。$\overline{P_iP_{i+1}}$と平行で、εの距離だけC側の直線の式l_2は

$$l_2 : a_0x + b_0y + c_2 = 0$$

ただし、 $a_0 = y_{i+1} - y_i$

$b_0 = x_i - x_{i+1}$

$c_0 = x_{i+1}y_i - x_iy_{i+1}$

$c_2 = c_0 - c\varepsilon\sqrt{a_0^2 + b_0^2}$

f_1が隣接する形状境界$\overline{P_iP_{i+1}}$と交差する場合、次の式でf_1ベクトルとl_2との交点(x_sy_s)にf_1の終点を移動させる。

$p_1 = (a_0x_c + b_0y_c + c_2)\ c$

$p_2 = (a_0x_f + b_0y_f + c_2)\ c$

if $p_1 > 0$ then
if $p_2 < 0$ then

$$x_s = \frac{x_c|p_2| + x_f|p_1|}{|p_1| + |p_2|}$$

$$y_s = \frac{x_f|p_2| + y_f|p_1|}{|p_1| + |p_2|}$$

else
$x_s = x_c$
$y_s = y_c$
endif
endif