Génération de taches bicolores : application aux caractères d'imprimerie; problèmes de nature géométrique

Christian Sico

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT-ETIENNE

N° d'ordre : 28 II

THESE

présentée par

Christian SICO

pour obtenir le grade de

DOCTEUR-INGENIEUR

INFORMATIQUE

GENERATION DE TACHES BICOLORES

-APPLICATION AUX CARACTERES D'IMPRIMERIE-

PROBLEMES DE NATURE GEOMETRIQUE

Soutenue à Saint-Etienne le 8 Mars 1982 devant la commission d'examen :

Président	M. LUCAS
Examinateurs	MM. COUEIGNOUX
	LEMAIRE
	MAHL

ECOLE NATIONALE SUPERIEURE

DES MINES DE SAINT-ETIENNE

N° d'ordre: 28 II

THESE

présentée par

Christian SICO

pour obtenir le grade de

DOCTEUR-INGENIEUR

INFORMATIQUE

GENERATION DE TACHES BICOLORES

-APPLICATION AUX CARACTERES D'IMPRIMERIE-

PROBLEMES DE NATURE GEOMETRIQUE

Soutenue à Saint-Etienne le 8 Mars 1982 devant la commission d'examen:

Président M. LUCAS

Examinateurs MM. COUEIGNOUX

LEMAIRE

MAHL

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur	: M. M.	MERMET
Directeur des Etudes et de la Formation	: M. J.	LEVASSEUR
Directeur des Recherches	: M. J.	LEVY
Directeur Administratif et Financier	: M. A.	COINDE

PROFESSEURS DE 1ère CATEGORIE

MM.	COINDE	Alexandre	Gestion
	GOUX	Claude	Métallurgie
	LEVY	Jacques	Métallurgie
	LOWYS	Jean-Pierre	Physique
	RIEU	Jean	Mécanique - Résistance des Matériaux
	SOUSTELLE	Michel	Chimie
	FORMERY	Philippe	Mathématiques Appliquées

PROFESSEURS DE 2ème CATEGORIE

M.	TOUCHARD	Bernard	Physique Industrielle

DIRECTEUR DE RECHERCHE

M.	LESBATS	Pierre	Métallurgie

MAITRES DE RECHERCHE

MM.	BISCONDI	Michel	Métallurgie
	DAVOINE	Philippe	Géologie
Mle	FOURDEUX	Angeline	Métallurgie
MM.	KOBYLANSKI	André	Métallurgie
	LALAUZE	René	Chimie
	LANCELOT	Francis	Chimie
	LE COZE	Jean	Métallurgie
	MATHON	Albert	Gestion
	PERRIN	Michel	Géologie
	THEVENOT	François	Chimie
	TRAN MINH	Canh	Chimie

PERSONNALITES HABILITEES A DIRIGER DES TRAVAUX DE RECHERCHE

MM.	DRIVER	Julian	Métallurgie
	GUILHOT	Bernard	Chimie
	THOMAS	Gérard	Chimie

Je tiens à remercier:

Monsieur Michel LUCAS, Professeur à l'Université de Nantes, qui a accepté de juger cette thèse et d'en présider le jury.

Monsieur Robert MAHL, Ancien Professeur au Servive Informatique de l'ENSMSE, qui, tout au long de l'étude, s'est intéressé à nos problèmes.

Monsieur Alain LEMAIRE, Responsable de l'Informatique Graphique à la société DANECOM, pour l'intérêt qu'il a manifesté à l'égard de ce travail et pour avoir accepté de participer au jury.

Monsieur Philippe COUEIGNOUX, Maitre de Recherche à l'ENSMSE durant le projet, initiateur de cette étude, qui nous a permis de progresser par ses critiques et ses conseils. Actuellement, Monsieur Philippe COUEIGNOUX est Conseiller Technique à l'IDI.

J'exprime aussi toute ma sympathie à Marc BLOCH pour son amicale complicité.

Enfin je remercie tous ceux qui ont contribué à la réalisation de cet ouvrage: Mesdames AVONDO et BONNEFOY, Messieur BROSSARD, DARLES, LOUBET et VELAY.

AVANT-PROPOS

Le travail présenté dans le premier chapitre repose sur une étude menée conjointement avec Marc Bloch. Il s'agit de l'étude et de la réalisation d'un générateur de caractères digital pour des applications qui demandent des formes de très grande qualité dans les domaines de la bureautique et de la photocomposition.

Ce chapitre présente les principes généraux qui ont permis la réalisation du projet et décrit brièvement l'architecture matérielle et les logiciels du prototype construit. Cette description très succinte est nécessaire à une meilleure compréhension des problèmes abordés dans la thèse de M. BLOCH et dans celle-ci. Cette partie, qui a été rédigée en commun, sert d'introduction aux deux mémoires.

M. BLOCH s'est intéressé à l'étude de relations sur l'ensemble des parois d'une tache bicolore régulière, ce qui correspond à des problèmes de nature ordinale. Il a étudié le classement des parois d'après leurs ordonnées et l'affectation optimale du calcul des parois sur un nombre variable de processeurs en parallèles. Nous allons nous intéresser dans cette étude aux problèmes de nature géométrique.

Dans cette seconde partie où est exposé le travail personnel sur lequel repose ce mémoire, nous présenterons:

-la courbe évoluée qui est utilisée avec la droite comme interpolant des contours.

-les études concernant la réalisation de l'étape de décomposition des courbes évoluées en cercles.

-un logiciel de codage automatique de contours qui permet de simplifier et d'abaisser le coût de cette étape de préparation des caractères.

-un logiciel réalisant des transformations géométriques sur des caractères codés par contours.

-deux variantes d'algorithme; l'une pour accélérer le tracé des cercles, l'autre pour remplacer les cercles par des ellipses.

Plan

CHAPITRE 1 : UN GENERATEUR DIGITAL DE CARACTERES

CHAPITRE 2 : INTRODUCTION AUX PROBLEMES DE NATURE GEOMETRIQUE

CHAPITRE 3 : METHODOLOGIE DE DECOMPOSITION DES COURBES EVOLUEES

CHAPITRE 4 : DECOUPAGE AUTOMATIQUE DE CONTOUR

CHAPITRE 5 : TRANSFORMATIONS GEOMETRIQUES

CHAPITRE 6 : VARIANTES D'ALGORITHMES

Chapitre 1

CHAPITRE 1

UN GENERATEUR DIGITAL DE CARACTERES

1.- Représentation des caractères d'imprimerie.

A.- L'esprit et la lettre.
B.- Le caractère d'imprimerie.
C.- Codage digital des caractères d'imprimerie.

2.- Définition d'un nouveau générateur digital de caractères.

A.- Constat initial.
B.- Objectifs.
C.- Marchés.

3.- Principes de réalisation.

A.- Interpolation du contour.
    1) Codage du contour.
    2) Compression numérique.
    3) Compression linguistique.
B.- Structures de parallélisme.
    1) Mise en parallèle des parois.
    2) Pipe-line de décodage et modularité.
    3) Variantes.

4.- Résultats obtenus.

A.- Structure effective.

1) Expansion linguistique.

2) Expansion numérique.

3) Suivi de contour.

4) Mise à l'échelle.

5) Synchronisation.

6) Réalisation.

B.- Performances.

1 - PRESENTATION DES CARACTERES D'IMPRIMERIE

A - L'ESPRIT ET LA LETTRE

Depuis les grandes découvertes de la Renaissance, qui ont assuré la généralisation de sa diffusion par l'impression, l'écriture est, dans notre civilisation, l'instrument privilégié de la transmission de l'information et de la connaissance. Aujourd'hui encore, ses avantages : faible coût, disponibilité immédiate, faible dégradabilité, richesse graphique, confidentialité, etc... expliquent qu'elle résiste bien aux autres moyens de communication de notre époque (audiovisuel, télématique) ; ces avantages n'auraient jamais été valorisés sans les progrès continus des techniques d'impression et de manipulation du texte. A l'heure actuelle, la mémorisation des textes, leur mise en page sont effectuées sur des ordinateurs, et leur restitution sur des machines complexes (presses offset, héliogravure, imprimante).

L'un des problèmes essentiels pour la reproduction des textes est le passage d'une représentation interne, (c'est-à-dire en machine), qui condense provisoirement le message en sa plus simple expression (codes ASCII, binaire...), à la représentation externe et définitive, dont la forme sur le support restitue le sens du message : c'est la phase de composition, d'abord manuelle, puis automatisée (HOU 78) qui engendre le texte en tant que forme.

Il est ici important de bien distinguer le sens d'un texte, le signifié, de la forme qu'il prend, le signifiant : un ensemble de symboles archétypiques, les caractères qui ne condensent aucune signification dans notre écriture se regroupent en mots, puis en phrases pour former un texte qui peut ou non avoir un sens. Mais, comme c'est le cas pour d'autres canaux de communication (la parole par exemple), la forme de l'écriture est elle même chargée d'une signification secondaire (caractères anciens ou modernes, gras ou légers,

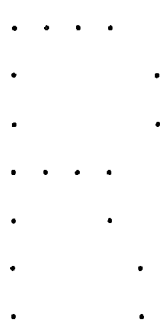
ronds ou droits). Au message principal, le sens des mots et du texte se superpose un message secondaire , l'ambiance du texte. Même automatisé, le processus d'écriture doit conserver ces variations, ce surplus d'information, cette redondance qui permettent une meilleure lisibilité et restituent une certaine sensibilité.

Pour des raisons technologiques et économiques, l'informatique a beaucoup sous-estimé l'importance de la forme de l'écrit ; de toutes façons, l'informaticien, qui connaît le prix de l'information, n'aime guère la redondance. Tout cela rend les listages d'ordinateur à la fois si uniformes et si peu lisibles... Par contre, dans ses applications au traitement de texte et à l'impression classique, l'informatique doit permettre d'égaler la richesse graphique des outils qu'elle tend à supplanter (machines à écrire, caractères au plomb...).

B - LE CARACTERE D'IMPRIMERIE

Les méthodes rapides d'impression pour le livre, la presse ou l'informatique exigent des méthodes rapides de restitution du texte ; il en existe deux grandes classes. Dans la première, la forme du caractère préexiste et est simplement transférée sur le support, par des moyens mécaniques (imprimantes à marguerites ou à chaîne) ou optiques (photocomposeuses à flash). De tels procédés ne nous concernent pas directement ; signalons cependant que les technologies utilisées ont limité l'évolution de deux caractéristiques importantes : la vitesse (à qualité donnée), et l'aptitude à manipuler la forme même du caractère (mise à l'échelle, inclinaison, épaississement). La seconde classe comprend toutes les méthodes où la forme du caractère doit être engendrée à chaque utilisation, (imprimantes à aiguille, afficheurs électroluminescents à segments ou à points, écrans de visualisation à balayage vidéo, photocomposeuses digitales) ; il faut alors mémoriser indépendamment de la technique de visualisation, une représentation du caractère. La plus commode, la mieux adaptée aux outils modernes de traitement de l'information, est la représentation digitale : la surface du caractère est discrétisée par échantillonnage sur une grille régulière, et on obtient une matrice de points

blancs ou noirs, à laquelle on fait correspondre une matrice binaire, la matrice du caractère. Une réalisation très répandue de ces principes se retrouve dans les écrans à points ou les imprimantes à aiguilles. La taille de la matrice est alors très petite : 5x7, 7x9 (figure 1.1).



. . . .	1 1 1 1 0
. .	1 0 0 0 1
. .	1 0 0 0 1
. . . .	1 1 1 1 0
. .	1 0 0 1 0
. .	1 0 0 0 1
. .	1 0 0 0 1

Discrétisation d'un caractère — Matrice associée (taille 7 x 5)

Figure 1-1

L'utilisation de petites matrices convient parfaitement aux imprimantes d'ordinateur, dont la qualité n'est pas la caractéristique essentielle. Il en va autrement pour le traitement de texte ou l'impression classique ; il faut alors représenter les caractères par de grandes matrices, et cela pour trois raisons principales :

* le nombre de formes que l'on peut engendrer à partir d'une petite matrice est très réduit ; une grille 5x7, par exemple, permet de représenter de manière schématique les majuscules de l'alphabet latin, les chiffres et quelques symboles spéciaux, mais non les caractères d'autres alphabets, souvent plus complexes (arabe, chinois).

* on ne peut représenter l'esthétique de la forme ; le message est pratiquement brut, et la signification secondaire, dont nous avons dégagé la nécessité, en est absente.

* l'appréhension de la forme doit être immédiate et agréable ; le bruit de quantification inhérent à la numérisation, qui se manifeste par des sauts quantifiés sur le support, doit être insignifiant après transfert sur le support, c'est-à-dire au maximum de l'ordre du pouvoir séparateur de l'oeil ; on peut alors restituer l'apparence d'une surface homogène dont le bord est parfaitement régulier.

  C'est ici qu'intervient la technique de visualisation par l'intermédiaire de la résolution du support, dont la gamme s'étend de quelques points au centimètre (écrans à balayage vidéo) à quelques centaines de points au centimètre (écrans graphiques de haute qualité, imprimantes xérographiques ou électrostatiques).

La résolution du support, et la taille de la matrice définissant le caractère déterminent ensemble la taille du caractère sur le support : un caractère défini dans une grille de 2000 points par 2000 points aura une hauteur de 4 cm sur un support de résolution 500 points/cm. Réciproquement, la taille maximale désirée des caractères sur le support, et la définition de ce support déterminent la taille des matrices : pour une hauteur maximale de 4 cm (pensons aux gros titres des journaux), et une résolution de 400 points/cm (légèrement supérieure au pouvoir séparateur de l'oeil), il faut représenter le caractère dans une grille de 1600 points sur 1600 points.

Il est malheureusement impossible de mémoriser les caractères par leur matrice, dès que la taille de celle-ci devient grande : le stockage d'un seul caractère de taille 2000x2000 exige 4 millions de bits... De toutes façons, la quantité d'information utilisée est alors grossièrement surévaluée ; par exemple, lorsque les caractères sont assez réguliers, les points du contour sont bien moins nombreux que les points de la surface, et en permettent pourtant la reconstitution. Un codage de la matrice est nécessaire ; la restitution se fait alors par l'intermédiaire d'un générateur de caractères, plus complexe que celui qui assure une simple recopie de la matrice.

C - CODAGE DIGITAL DES CARACTERES D'IMPRIMERIE

Avant d'expliciter les différentes catégories de code, nous allons décrire les variables qui vont nous permettre de les caractériser (COU -). Les premières ne dépendent que du code utilisé ; ces facteurs endogènes sont :

. la quantité moyenne de mémoire nécessaire à la description d'un caractère ; nous dirons la taille du code ;

. la dimension de la plus grande matrice de points engendrable sans déformation, ou qualité du code ;

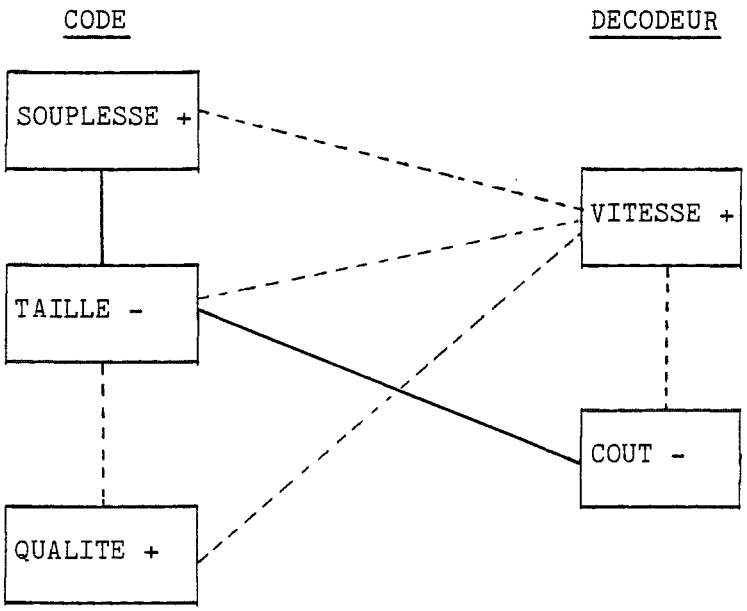
. l'ensemble des diverses modifications (mise à l'échelle, rotations, inclinaisons, épaississement du trait) que la nature du code permet d'automatiser simplement, ou souplesse du code.

D'autres facteurs dépendent aussi du décodeur :

. la vitesse de génération ;

. le coût du générateur.

Bien sûr, ces variables ne sont pas indépendantes, et la figure 1-2 montre certaines de leurs interrelations :



1) Pour chaque facteur, le sens de variation souhaité est indiqué par le signe.

2) Une liaison pleine (resp. pointillée) indique que les sens de variation souhaités sont compatibles (resp. incompatibles).

Figure 1-2

Des critères externes peuvent aussi intervenir (COG 80) :

. la résolution du support, que nous avons déjà mentionnée.

. la méthode de remplissage : les codes qui utilisent ou calculent une représentation du contour ont besoin d'un tampon de mémoire pour le remplissage de la surface. Si le contour est déterminé de manière purement séquentielle, le remplissage se fait après calcul dans un tampon dont la taille doit être celle de la matrice de représentation. Nous verrons que certains types de codes permettent le remplissage au vol du contour, en une passe, horizontale ou verticale, la taille du tampon est alors celle d'une ligne ou d'une colonne de la matrice.

. la méthode de balayage : les balayages cavaliers sont peu adaptés à l'encrage des surfaces ; pour minimiser le déplacement, on en arrive à simuler un balayage vidéo ligne par ligne. Nous sommes donc concernés essentiellement par les balayages vidéo ; il faut en distinguer deux classes :

  - balayage vidéo unitaire : les caractères sont engendrables individuellement sur le support.

  - balayage vidéo composite : il faut calculer toute une ligne de texte avant de pouvoir la restituer sur le support, c'est-à-dire engendrer, individuellement ou non, l'ensemble des caractères qui la constituent dans un tampon de mémoire spécifique.

Les compromis dépendent ici de la technique de visualisation : les supports qui sont gérés par une mémoire d'image permettent aussi bien un remplissage a postériori qu'au vol et sont mieux adaptés à un balayage composite, puisque les tampons nécessaires existent déjà ; par contre, les supports qui n'ont pas besoin de mémoire intermédiaire propre - par exemple tubes de photocomposeuses où la

mémorisation se fait immédiatement sur un film photosensible -, demandent plutôt un remplissage au vol et un balayage unitaire, pour minimiser la quantité de tampon externe.

Il existe trois grandes classes de code : représentations de la surface, du contour ou de la structure des caractères ; leurs diverses variantes permettent d'obtenir un codage optimal pour un usage déterminé. Le lecteur pourra trouver dans (COU -) une bibliographie et des développements sur le sujet.

1°) Codages de la surface

N est la dimension de la grille de discrétisation.

* matrice binaire

Le caractère est codé par sa matrice ; la vitesse de génération est optimale, mais la taille du code est proportionnelle à N^2 (fig 1-3-1).

* code par plages

Pour chaque ordonnée Y_i (resp. abscisse X_i), on mémorise les abscisses (resp. les ordonnées) des points d'intersection du contour avec la ligne $Y=Y_i$ (resp. la colonne $X=X_i$) ; les segments ainsi déterminés sont des plages. C'est un code en $kN \log_2 N$, où k est une constante représentative de la complexité moyenne du caractère : 4 pour les caractères de polices romaines de corps de texte, plus de 10 pour les chinoises.
Ce code est particulièrement adapté au balayage vidéo (figure1-3-2).

2°) Codages du contour

* code par plages différentiel

On se contente de noter les variations des extrémités des plages. Les deux extrémités d'une plage particulière peuvent alors être traitées séparément sous forme de deux listes de

listes de variations. Cette dichotomie a l'avantage de faire apparaître des morceaux de contour, qu'on nommera parois, indépendants les uns des autres, donc adaptés à un traitement parallèle. Il faut malheureusement rajouter au code une superstructure qui permet de repérer les naissances et les morts des parois ; on perd aussi une notion d'ordre qui était implicite dans le balayage par plage (fig. 1-3-3).

La taille d'un tel code, bien adapté au balayage vidéo, peut être approchée par $6\ k\ N + b \log_2 N + c$.

k : nombre moyen d'intersections, donc de parois $\simeq 4$

b : nombre moyen de points de naissance (Romaine $\simeq 4$)

c : caractéristique de la gestion des naissances (16 bits).

* code interpolé du contour

Plutôt que de coder toutes les variations des points du contour, on considère ce dernier comme la concaténation d'interpolants canoniques, tels que segments de droites, arcs de coniques ou plus généralement splines (COU 73) (COU 75), mémorisés sous forme analytique. Une telle approche est compatible avec la structure dégagée au paragraphe précédent, à condition qu'une paroi soit codée par un nombre entier de courbes. Les points d'une paroi sont calculés à partir de l'équation aux différences finies de l'interpolant.

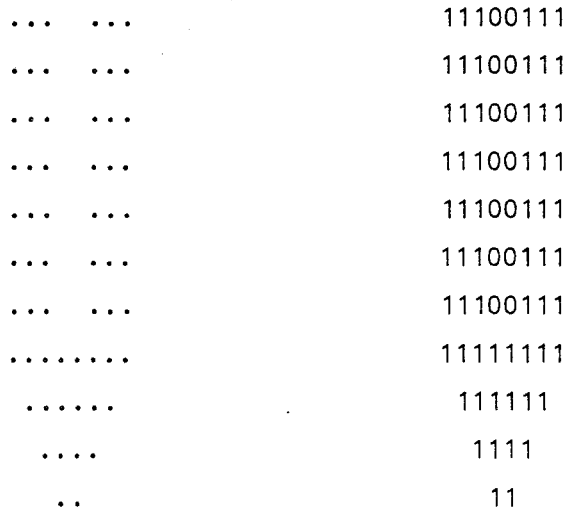
3°) codage de la structure

* Primitives (COU 75)

Le caractère est décrit comme la juxtaposition de primitives globales : barres, courbes... sur lesquelles on peut opérer certaines opérations (inclinaisons, symétries). Les paramètres géométriques mémorisés sont hauteur, longeur, carrure, pente, etc... ; chaque primitive est décrite par une interpolation de son contour.

* Squelettes (KNU 78)

Le caractère est codé par son squelette, et par une fonction qui en définit l'épaisseur en tout point ; le modèle imite le déplacement sur le papier d'un pinceau d'épaisseur et d'orientation variables.



1-3-0 : Discrétisation

1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 3 6 8
1 8
2 7
3 6
4 5

1-3-1 : Matrice binaire

1(*)	3(*)	6(*)	8(*)
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0(†)	0(†)	0
1			-1
1			-1
1(†)			-1(†)

* : naissance
† : mort

1-3-2 : Code par plages (plages horizontales)

1-3-3 : Codage par plages différentiel

Figure 1-3

Le tableau 1-4 synthétise les caractéristiques générales de ces différents codes.

		Taille du code	Qualité	Souplesse	Vitesse	Tampon	Balayage vidéo
SURFACE	Matrice binaire	N^2	$N \leq 50$	négligeable	très grande	0	+ + +
	Code par plages	$kN\log_2 N$	$N \leq 100$	"	grande	N	+ +
CONTOUR	Code par plages différentiel	$6kN + b\lceil \log_2 N + c \rceil$	$N \leq 100$	"	moyenne	N	+ +
	Code interpolé	$K \log_2 N$	$N \leq 2000$	dépend de l'interpolant	faible	N ou N^2	+
STRUCTURE	Primitives	$K'\log_2 N$	$N \leq 250$	très grande	très faible	N^2	-
	Squelettes	$K'\log_2 N$	?	grande	faible	N^2	-

Figure 1-4 : Tableau récapitulatif codage

2 - DEFINITION D'UN NOUVEAU GENERATEUR DIGITAL DE CARACTERES

A - CONSTAT INITIAL

. Local et global

Les codes qui autorisent des manipulations sur la forme des caractères utilisent une représentation très éloignée de la matrice d'origine. En effet, pour les inclinaisons, rotations... une information globale sur la structure est nécessaire ; elle ne peut être extraite des codes à dominante locale (matrice, contour), sans faire appel à des techniques complexes et coûteuses voisines de celles de la reconnaissance des formes.

L'utilisation de codes à dominante globale présente cependant deux inconvénients majeurs : d'une part les nombreux calculs limitent la vitesse de génération, et d'autre part une restitution en une passe de balayage est pratiquement impossible.

Si l'on ne veut pas tenir compte de la structure globale du caractère, les codages du contour, locaux (code par plages différentiel) ou interpolés (splines...) sont optimaux au sens de la théorie de l'information.

Il est bien sûr toujours envisageable de rajouter des informations globales à un code local pour l'adapter.

. Interpolation

On peut toujours améliorer un codage du contour par interpolation :

- en choisissant comme noeuds des points significatifs du contour qui permettent de relier le local au global (points à tangente horizontale ou verticale, points d'inflexion ou de rebroussement).

- en déterminant un interpolant canonique qui soit à la fois suffisamment simple pour faciliter le codage, minimiser les calculs requis par les opérations spéciales, augmenter la souplesse, tout en permettant une grande rapidité de décodage des contours (HOC 79).

. Vitesse et parallélisme

La vitesse effective du décodage peut être élevée, pour peu qu'on puisse découper le processus de génération en tâches indépendantes ; il suffit de prévoir leur réalisation en parallèle.

. L'état de l'art

Les machines actuelles les plus modernes utilisent en fait un codage par contour, mais elles sont limitées par le choix des interpolants.

En particulier les interpolants polygonaux produisent des angles peu esthétiques (Linotron 202) ; les interpolants circulaires sont difficiles à placer sur le contour et les interpolants d'ordre supérieur engendrent trop de calculs.

Signalons pour terminer que, faute de savoir décoder un contour suffisamment vite, les photocomposeuses de haut de gamme (Lasercomp, Digiset...) utilisent un codage par plages, extrêmement coûteux en mémoire de masse.

B - OBJECTIFS

A partir de ce constat initial, et compte tenu de son expérience dans le codage digital et le décodage des caractères d'imprimerie (COU 73) (COU 75) (HOU 78) (BLO 79) (SIC 80), l'équipe Communications Visuelles a développé un nouveau générateur de caractères.

Ce générateur utilise un code original d'interpolation du contour et une structure hautement parallèle, pour atteindre les objectifs suivants :

. Très bonne qualité :

Chaque caractère est codé sur une grille de 2000 points sur 2000 points, et peut être décodé à cette définition sans perte d'information.

. Forte compacité :

Pour une police romaine de corps de texte, 800 bits en moyenne par caractère.

. Souplesse

Tout facteur d'échelle peut être appliqué pour amener le caractère dans une grille de taille désirée, inférieure à 2000x2000 ; cette transformation se fait par réduction à partir de la représentation de base, donc sans augmentation du bruit de quantification. Les rapports peuvent être différents en X et en Y.

. Vitesse

1000 caractères par seconde, pour des caractères réduits dans une grille 100 x 100.

. Balayage

Génération du caractère en une passe de balayage vidéo.

Ces caractéristiques définissent un générateur de caractères de haut de gamme, destiné au marché de la photocomposistion, et qui s'avère nettement supérieur à la concurrence.

C - MARCHES

Nous nous sommes rendu compte que le code et l'organisation du décodeur étaient en fait compatibles avec les opérations spéciales que nous ne pensions pas prendre en charge (rotations, inclinaisons, variations de l'épaisseur du trait). Leur réalisation pourrait se faire simplement au prix d'une étape de décodage supplémentaire, et d'une réduction de la vitesse globale de la machine. Il s'agit là d'extensions de ses possibilités sans bouleversement de l'architecture.

L'indépendance des différentes phases du décodage a induit naturellement une structure fortement modulaire, qui sera analysée plus loin. L'existence de cette modularité nous a permis de décrire et d'étudier des variantes déduites de la version de base. Il s'agit alors de s'adapter à des segments de marché différents - écrans, traitement de texte, imprimantes rapides, - avec une compatibilité totale aussi bien au niveau du logiciel que du matériel. Cela est très important lorsque l'on sait que le traitement de texte est de plus en plus utilisé, à un coût de plus en plus bas ; il faut alors retrouver en numérique la souplesse du plomb et la qualité du travail d'un petit atelier de typographie, à un coût bien moindre.

Quelques exemples de variantes étudiées :

. La première expansion peut être supprimée, à condition d'interdire l'utilisation de primitives ; la taille moyenne du code passe alors de 800 à 1600 bits par caractère.

. Il est possible de faire varier le nombre de processeurs chargés du calcul effectif des points des parois, (processeurs de suivi de contour), pour adapter la vitesse de décodage, - et le prix -, à une application précise :

2 processeurs - 200 car/s - 40 kF
10 processeurs - 1000 car/s - 80 kF

. Pour des versions bas de gamme, une diminution importante du coût, et de la vitesse, sera obtenue par changement de la technologie. Ainsi, plusieurs cartes microprogrammées effectuent la deuxième phase du décodage (transformation des interpolants canoniques), de manière à garantir une vitesse de fonctionnement de 1000 car/s. Une autre possibilité utilise un microprocesseur standard et son coprocesseur de calcul, mais limite la vitesse de la machine à environ 50 car/s. Le même type de modification peut être envisagé pour la mise à l'échelle, en remplaçant un composant rapide (multiplieur TRW) par un microprogramme. Il s'agit bien sûr d'aspects divers de l'éternel dilemme matériel-logiciel ; tout algorithme peut être entièrement cablé, ou entièrement programmé ; c'est au concepteur, à l'architecte des systèmes de dégager de l'étude un compromis correct.

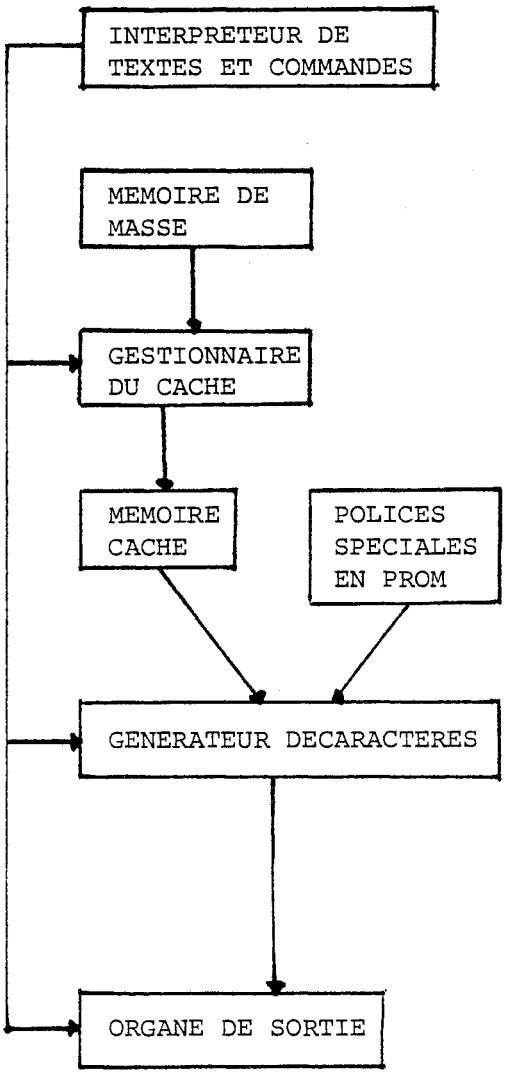
Différentes versions peuvent donc être dérivées du prototype réalisé de manière à optimiser la compacité du code, la quantité de tampon, la souplesse, la vitesse et le coût pour une application déterminée.

Normalement, la configuration complète correspond à un environnement de photocomposeuse (fig. 1-5) ; on porte l'attention sur la qualité (2000 x 2000), et la vitesse (1000 car/s). Au contraire, une version plus lente et meilleur marché peut être utilisée pour engendrer des caractères en code par plages ou matrice de points, pour une imprimante d'ordinateur ou de machine de traitement de texte (figure 1-6) ; on porte alors l'attention sur la souplesse (opérations spéciales) et la taille du code (800/1600 bits/car).

Le tableau 1-7 résume quelques applications élémentaires de notre générateur de caractères.

Type d'application	Vitesse (car/s)	Qualité maximale (points)	Souplesse	Prix (kF)
Photocomposeuse (haut de gamme)	1000	2000	mise à l'échelle X, Y	70 - 100
Photocomposeuse (bas de gamme)	200	2000	"	40 - 70
Imprimante (haut de gamme)	200	200	toutes opérations par materiel	20 - 40
Imprimante (bas de gamme)	50	200	toutes opérations par logiciel	10

Tableau 1-7



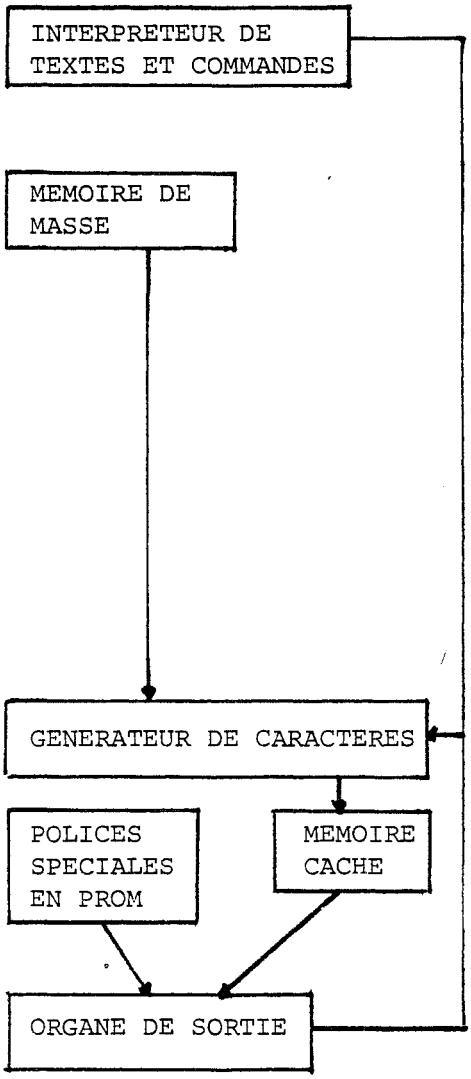
ENVIRONNEMENT DE PHOTOCOMPOSEUSE

Résolution:
De 200 à 640 lignes/cm

Taille des caractères:
De 2 à 180 points par 1/2 point

Vitesse:
De 500 à 1000 car/s

Figure 1-5



ENVIRONNEMENT D'IMPRIMANTE D'ORDINATEUR

Résolution:
De 80 à 200 lignes/cm

Taille des matrices:
De 15 à 300 points par 1

Vitesse:
De 50 à 250 car/s pour remplir la mémoire cache

Figure 1-6

On peut aller plus loin : les caractères sur lesquels nous avons travaillé jusqu'à présent ne sont autre chose que des taches bicolores régulières, et notre décodeur un générateur de taches ; on peut ainsi concevoir une application générale pour tout type de terminal à balayage vidéo : générations de caractères, symboles et logotypes ; modification d'images en noir et blanc simples et régulières par déformations géométriques pour l'animation.

Les algorithmes dégagés peuvent eux-mêmes participer, de manière autonome, aux développements de l'informatique graphique :

. le codage du contour utilise une méthode d'interpolation très simple, bien adaptée à une saisie semi-automatique ou interactive.

. le code retenu contient une information supplémentaire, destinée à rendre compatible la définition séquentielle sur contour avec l'ordre nécessaire au balayage vidéo ; elle est normalement dégagée au moment du codage, hors-ligne, à priori, dans une phase de compilation du caractère. Nous avons longuement étudié cette structure, et essayé de la déterminer de manière optimale. Ce travail n'est pas gratuit : dans le cas de certaines opérations spéciales, elle doit être recalculée en temps réel à partir de la description du contour après transformation.

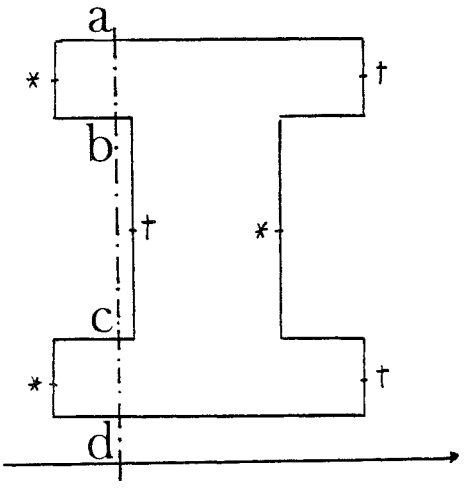
3 - PRINCIPES DE REALISATION

A - INTERPOLATION DU CONTOUR

1°) Codage du contour :

Nous avons vu que l'un des objectifs du projet est d'augmenter la finesse du caractère tout en diminuant la taille des fichiers du code en entrée. Pour arriver à une densité permettant de respecter ces deux contraintes seul un codage du contour est envisageable. Si l'on regarde la manière dont le caractère est créé

sur un tube cathodique, les seuls points qu'il est nécessaire de connaître à un instant donné sont les points d'intersections du contour et de la droite de balayage en cours. Ainsi, il est nécessaire de suivre en temps réel abscisse après abscisse les parois, portions du contour représentables comme une fonction univoque de l'abscisse. Le débit maximum est, pour une force de corps donnée, fonction de la résolution désirée, qui détermine le nombre d'abscisses à calculer, et du nombre de parois à suivre. Les opérations de suivi des parois étant rendues totalement indépendantes l'une de l'autre, il est possible de rendre le temps de calcul indépendant du nombre de parois à calculer en plaçant en parallèle autant de processeurs qu'il y a de parois. Le débit de la machine doit être de 1000 car/s pour une matrice de 100 x 100 ce qui donne un temps de calcul de 10 s par point. Les seules courbes utilisables sont les droites et les cercles pour lesquelles il existe des algorithmes rapides et précis permettant de trouver l'accroissement sur un axe en connaissant la variation sur l'autre (HOR 77) (BRE 77). Les contours d'un caractère peuvent être codés sous la forme d'une structure d'anneaux, chaque anneau est la description d'une composante connexe découpée en une suite de droites et de cercles. Sous cette forme le codage d'un signe nécessite en moyenne 3200 bits soit 64 K octets pour une police de 160 signes. Nous allons maintenant examiner deux méthodes pour augmenter la densité du code.



* : points de naissance de paroi
† : points de mort de paroi
a, b, c, d : points à calculer pour le balayage en cours

2°) Compression numérique :

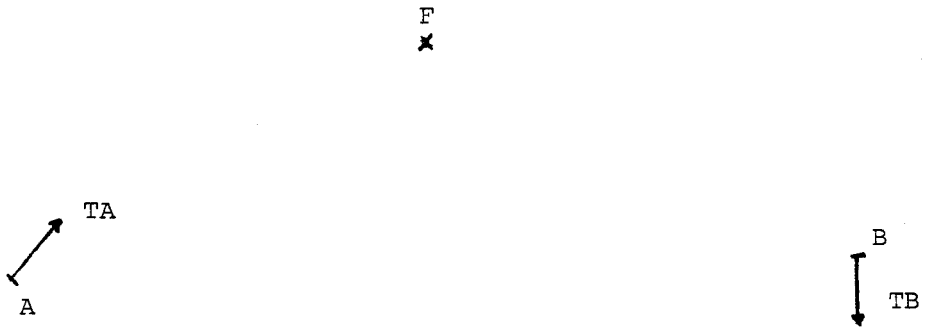
La difficulté de trouver des arcs de cercle réalisant une bonne approximation d'une courbe quelconque lors de la saisie du contour des caractères, a conduit à étudier les méthodes de découpage automatique de courbe en arcs de cercle ; c'est la notion de courbe évoluée introduite par Marc Hourdequin (HOU 78).

Le codage d'une courbe évoluée demande six mots alors que le codage de quatre cercles en nécessite seize, il apparaît donc possible de comprimer le code dans un rapport deux au prix de nombreux calculs numériques à effectuer lors du décodage. La taille des fichiers d'entrée est ainsi ramenée à 32 K octets par police. De plus, l'utilisation de mots de 16 bits alors que les coordonnées de grille sont sur 11 bits permet d'améliorer la densité du code en utilisant les quatre bits de poids forts pour coder le cas très fréquent où les tangentes des extrémités de la courbe sont verticales ou horizontales.

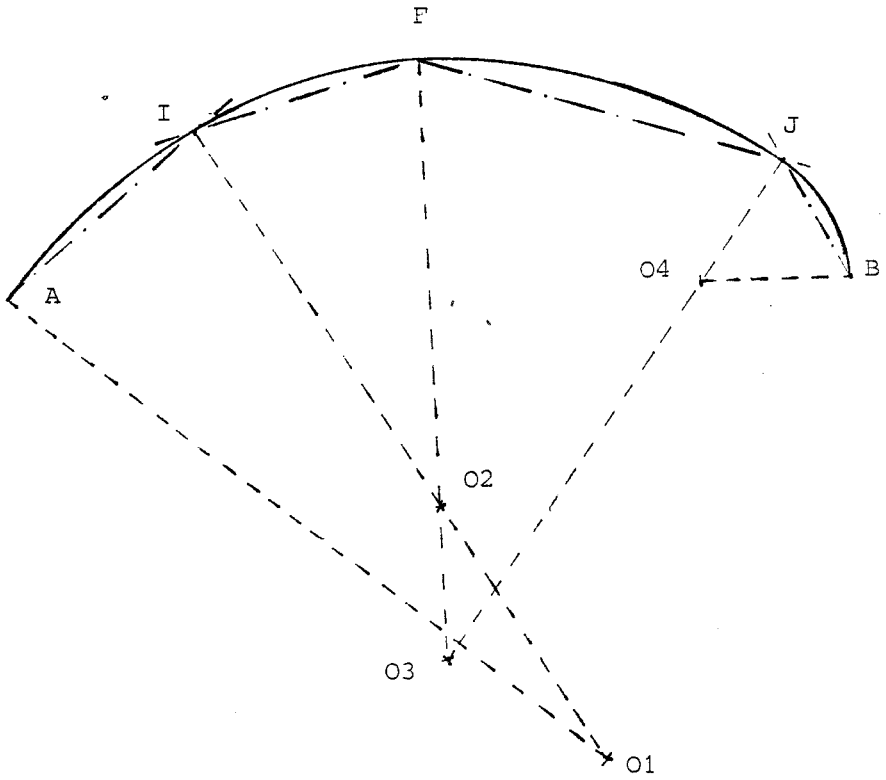
3°) Compression Linguistique

Cette compression est liée aux polices utilisées qui présentent de nombreuses portions de contour identiques. Ces éléments graphiques répétitifs, appelés primitives, peuvent être codés séparément. Lors de la rencontre d'un de ces éléments dans un contour, le codage du contour est remplacé par une simple référence à la primitive. Il est également possible de préciser des opérations simples telles que la translation, les symétries par rapport à X et Y et l'inversion du code afin d'utiliser au mieux cette notion. L'efficacité de cette compression est très variable d'une police à l'autre et suivant celle-ci la taille du fichier variera de 12 à 16 K octets (fig. 1-8).

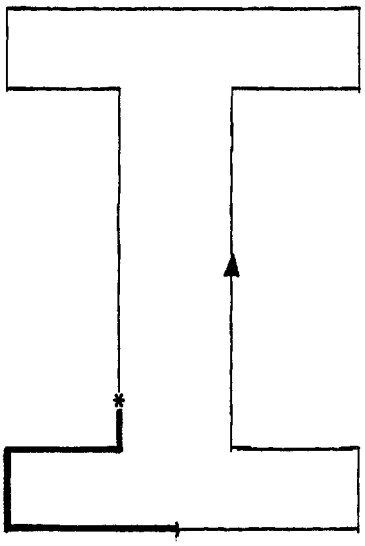
PARAMETRES D'ENTREE



RESULTATS



COURBE EVOLUEE



→ sens de parcours

* début primitive

PRIMITIVE EMPATTEMENT

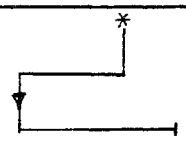
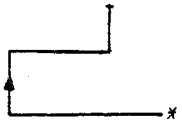
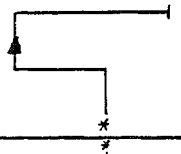
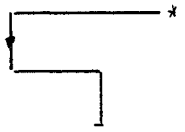
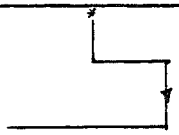
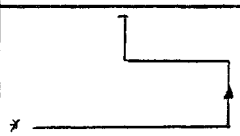
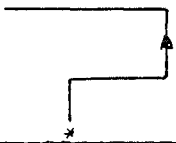
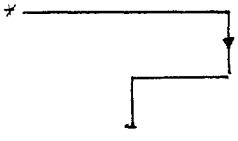
	CODE NON INVERSE	CODE INVERSE
SANS SYMETRIE		
SYMETRIE /OX		
SYMETRIE /OY		
SYMETRIE /O		

Figure 1-8

STRUCTURE D'ARBRE

1 POINTEUR
N° ORDRE N° PAROI N° ORDRE N° PAROI
0 ABSCISSE DE NAISSANCE
1 POINTEUR
N° ORDRE N° PAROI N° ORDRE N° PAROI
1 POINTEUR
N°ORDRE N° PAROI N° ORDRE N° PAROI
"
0 ABSCISSE FIN DE LETTRE

STRUCTURE D'ANNEAU

000 X DROITE
Y DROITE
100 XA
YA
TA
XF' } COURBE
YF } EVOLUEE
TB
XB
YB
111 REFERENCE
TRANSLATION X } PRIMITIVE
TRANSLATION Y
010 FIN D'ANNEAU
"
110 FIN DE LETTRE

ANNEAU

CODE D'ENTREE

B - STRUCTURES DE PARALLELISME

1°) Mise en parallèle des parois

L'opération de suivi des parois du caractère à partir de la seule description des contours contenue dans la structure d'anneau demanderait une exploration préliminaire du fichier afin d'acquérir des informations sur la structure globale de la lettre. La vitesse demandée à la machine ne permet pas ces calculs préliminaires. Une structure d'arbre placée en tête du fichier, contient toutes les informations nécessaires pour initialiser les processeurs de suivi de contour, et les rendre autonomes. Ces informations sont l'abscisse de naissance, un pointeur dans la structure d'arbre, le numéro de paroi et le numéro d'ordre.

- L'abscisse de naissance indique aux processeurs à quel moment ils doivent commencer le calcul de la paroi.

- Le pointeur dans la structure d'anneau indique où se trouve le paramètre concernant la paroi.

- Le numéro de paroi permet au processeur qui doit prendre en charge la paroi de se reconnaître. Ce numéro évite d'avoir une gestion dynamique de la charge des processeurs de suivi de contour. On remplace une allocation dynamique par une auto-allocation précalculée. Chaque processeur reçoit à la mise en route le nombre de processeurs présents et un numéro de référence puis il calcule les numéros des parois qu'il devra reconnaître.

- Le numéro d'ordre permet de classer les ordonnées avant de les transmettre à l'organe de visualisation.

Toutes ces informations sont calculées lors de la saisie du caractère et, à part les abscisses de naissances qui sont mises à l'échelle, elles ne sont pas modifiées lors du décodage.

Les pointeurs dans la structure d'anneau sont déterminés de façon à pointer les coordonnées du point de naissance après les expansions du code correspondant aux deux compressions.

2°) Pipe-line de décodage et modularité

Il est impossible de concevoir une machine à un seul étage capable de décoder les caractères en respectant le débit de 1000 car/s. Le décodage peut facilement être découpé en trois étapes totalement indépendantes et séquentielles, ce qui permet d'utiliser une structure pipe-line dans laquelle chaque étage correspond à une étape d'expansion du code.

La machine a donc été construite sous une forme très modulaire. La seule étape nécessaire pour garder un codage par contour est l'étape 3 qui réalise le suivi des contours. Les autres étapes réalisent une expansion du code et ne sont intéressantes que pour obtenir une densité très élevée en entrée. L'étape intermédiaire qui réalise la mise à l'échelle est bien entendu obligatoire dans toute application industrielle ; il faut remarquer également que la mise à l'échelle indépendante en abscisse et en ordonnée rend nécessaire l'étape 2 qui transforme les courbes évoluées en cercles.

En plus de cette modularité en nombre d'étage, il existe une modularité au niveau de l'étape 3 qui autorise de deux à dix cartes travaillant en parallèle avec, bien entendu, une répercussion sur le débit de cette étape. Une autre possibilité, que nous réalisons, est de remplacer l'étape 2 relativement complexe (4 cartes) mais rapide par une carte utilisant le 8087, coprocesseur du microprocesseur 8086 d'Intel, ce qui entraîne un débit vingt fois plus faible. Les versions qu'il est possible de réaliser à partir de la machine vont de deux cartes si on accepte une densité de code non maximale à 18 cartes si on désire un code très compact et une machine très rapide. Les diverses versions sont indiquées dans le tableau qui suit. A l'intérieur d'une colonne, le code des caractères en entrée est inchangé.

Pour arriver à la première ligne, la fonction de contrôle général est incluse dans la carte de mise à l'échelle et la carte de tri en sortie est supprimée en considérant que l'unique processeur de suivi de contour effectue les calculs des ordonnées dans l'ordre de leur transmission.

Densité du code

	800 bits/car	1600 bits/car	3200 bits/car (1)
CAR/s	nombre de cartes		
20	4	3	2
50	7	6	-
200	10	9	5
300	12	11	7
1000	18	17	13

(1) impose échelle X = échelle Y

3°) Variantes

La structure pipe-line et le code utilisé offrent une grande souplesse et permettent de rajouter des opérations spéciales sans bouleverser la partie déjà réalisée.

Toutes ces opérations : épaississement, inclinaison et rotation imposent de modifier la structure d'anneau et la structure d'arbre. Dans la structure d'anneau elles modifient les coordonnées et les angles et la rotation impose en plus une rotation de chaque anneau afin qu'il débute par un point de mort ce qui simplifie leur exploitation. Dans la structure d'arbre les deux

premières opérations n'introduisent qu'une modification des abscisses de naissance alors que la rotation demande de recalculer les points de naissance, dont le nombre peut varier, ainsi que les pointeurs et numéros d'ordre et de paroi qui leur sont associés. Si l'on étudie l'effet produit par toutes ces opérations appliquées à un caractère on constate que celles-ci ne sont pas commutatives, il est donc nécessaire de fixer une priorité entre elles de façon à obtenir un résultat qui semble naturel à l'utilisateur. La modification de la structure d'arbre pour toutes ces opérations peut être réalisée sur une seule carte placée, dans la structure de la machine réalisée, après l'étape de calcul des courbes évoluées.

Les modifications dans la structure d'anneau peuvent être réalisées grâce à une carte par opération placée dans l'ordre des priorités entre opérations. Pour des versions lentes ou n'utilisant que rarement ces opérations spéciales, il est possible de les regrouper sur une ou deux cartes en fonction des caractéristiques souhaitées. L'épaississement et l'inclinaison peuvent être rajoutés à la maquette sans modifier le travail réalisé ; pour introduire la rotation il serait nécessaire de modifier le logiciel de l'étape de transformation de courbes évoluées en cercles afin de calculer les points à tangentes verticales (naissances ou morts) ou à tangentes horizontales (changement de cercle) à l'intérieur des courbes évoluées.

4 - RESULTATS OBTENUS

A - STRUCTURE EFFECTIVE

Pour certifier le générateur de caractère décrit plus haut, il a été décidé de se limiter à un débit de 500 car/s. C'est la vitesse des photocomposeuses modernes sur le marché (LINOTRON 202, COMPUGRAPHIC 8600). La description de la maquette est donnée étape par étape et le schéma fonctionnel est détaillé au niveau de la carte.

1°) Expansion linguistique (Etape 1)

Au cours de cette étape les références aux primitives indiquées dans le fichier d'entrée sont remplacées par le code du contour correspondant, translaté, modifié par les symétries et inversé si nécessaire. Cet étape réalise le transfert d'environ 200 mots avec addition sur 100 ; si nous considérons un temps d'addition égal au temps de transfert, cela nous donne 300 opérations élémentaires à réaliser en 2 ms. Cette carte est réalisée à partir d'un microprocesseur 16 bits (le 8086 d'Intel) qui nous permet de manipuler l'information par mot et rend l'accès à la base de données très simple.

Pour une application utilisant plus d'une dizaine de polices la taille du fichier d'entrée impose de le placer sur disque, sur disquette ou sur mémoire à bulles. Un gestionnaire de la mémoire de masse recopie à la demande une police dans une mémoire tampon qui sera exploitée par l'étape 1. Pour la maquette nous n'avons qu'un petit nombre de caractères d'essai, ce qui nous a permis de placer le fichier d'entrée sur mémoires EPROM implantées sur la carte de l'étape 1.

2°) Expansion numérique (Etape 2)

Au cours de cette étape chaque courbe évoluée est remplacée par quatre arcs de cercles. Une courbe évoluée est une courbe de classe C^1 sans point d'inflexion et dont la variation de l'angle du vecteur tangent est inférieure à π. Elle est définie dans le fichier d'entrée par ses deux extrémités A et B, les angles des tangentes en ces points TA et TB, et le point de carrure F. Le remplacement de la courbe par quatre arcs de cercles demande le calcul de deux points intermédiaires I et J respectivement compris entre AF et FB, et des centres des quatre cercles. Les calculs préliminaires donnent les angles des tangentes en F, I et J ; ces deux derniers points étant considérés comme les points de carrure des courbes AF et FB. Le calcul des coordonnées des 6 points est effectué par la résolution de 6 systèmes linéaires 2x2 dont les coefficients sont fonction des angles des tangentes à la courbe.

Afin de respecter le débit de la machine on a réalisé un sous pipe-line de calcul comprenant 2 cartes microprogrammées à base de microprocesseurs en tranche (2901 de AMD). La première réalise les calculs préliminaires c'est-à-dire le calcul des tangentes en F, I, J par division et l'accès à des tables trigonométriques pour obtenir les coefficients des systèmes linéaires. La deuxième résoud les systèmes linéaires par une suite de cinq multiplications qu'elle sous-traite à une carte réalisant la multiplication en virgule flottante. Ce sous pipe-line est alimenté par une carte de lecture et de tri qui transmet à la partie calcul les paramètres d'entrée des courbes évoluées et poursuit, en parallèle avec la partie calcul, l'exploitation du fichier d'entrée.

3°) Suivi de contour (Etape 3)

Au cours de cette étape la représentation séquentielle du contour est transformée en une représentation par matrice creuse créée colonne après colonne.

Cet étage est composé de 2 à 10 processeurs de suivi de contour ; afin de pouvoir suivre toutes les parois, chaque processeur en calcule un nombre variable en fonction de la configuration présente. A la mise en route de la machine chaque processeur reçoit un numéro et le nombre de processeurs présents et détermine les numéros des parois qu'il prendra en charge. Cette procédure de repliement présente 3 avantages :

- La possibilité d'effectuer la mise au point avec seulement 2 cartes ;

- Le repliement automatique qui lors de la mise en route de la machine permet par un programme d'autotest de configurer cet étage en fonction des cartes saines ;

- La possibilité de produire des machines ayant des débits variant dans un rapport 10 sans changer le matériel ni le logiciel.

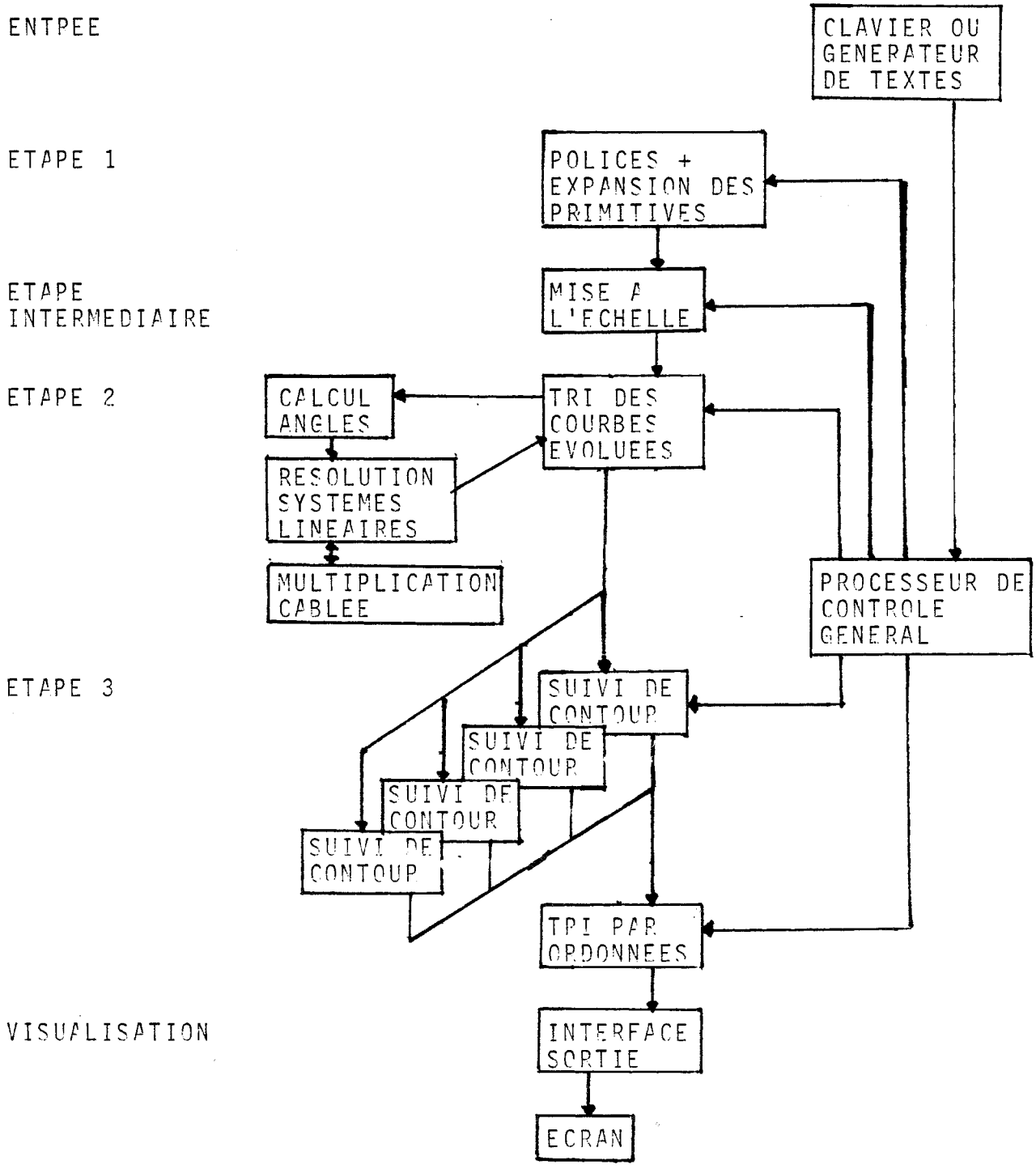
En contrepartie, il est nécessaire d'associer à chaque paroi un numéro (numéro de paroi) qui permet au processeur concerné de se reconnaître. De même, le problème de classement des ordonnées a été résolu en associant à chaque paroi un numéro d'ordre. Une carte réalisée en logique câblée effectue rapidement le tri par adressage à l'aide du numéro d'ordre lors de la saisie des résultats et l'adressage en séquence lors de leur transmission.

4°) Mise à l'échelle (Etape intermédiaire)

Au cours de cette étape le caractère est réduit afin d'être placé dans une grille de taille désirée ; le format de la grille peut aller de 2048 x 2048 à 30 x 30. Le nombre élevé de coordonnées à mettre à l'échelle nous a conduit à utiliser un multiplieur rapide, la division par un nombre constant étant remplacée par une multiplication suivie d'un décalage. La possibilité d'avoir une échelle différente en X et en Y empêche d'effectuer la mise à l'échelle après l'étape 2 à cause des cercles, qui seraient transformés en ellipses ; celle-ci est donc placée après l'étape 1. Il est alors nécessaire de modifier les angles en fonction du rapport des échelles ; cette modification est réalisée grâce à une table et limite à 32 le nombre de rapports d'échelle possibles, pris entre 0,5 et 2.

5°) Synchronisation

Les temps de calcul des étapes 1, 2 et intermédiaires ne dépendent que du caractère à traiter ; par contre, le temps de calcul de l'étape 3 dépend également de l'échelle adoptée. De plus, l'origine du temps de calcul critique est fonction de la suite de caractères engagés dans le pipe-line ; il est ainsi impossible de connaître à priori l'étape critique. En outre, pour des raisons de simplicité la synchronisation a été réservée à un processeur indépendant PCG. Ce dernier supervise les tests de



SCHEMA FONCTIONNEL DE LA MAQUETTE

fonctionnement correct à la mise en route de la machine, il transmet aux processeurs de suivi de contour les informations nécessaires au recouvrement, il indique à l'étape 1 le numéro du caractère désiré, il calcule le rapport d'échelle, de plus il pourra gérer la mémoire de masse des polices située sur disque et le positionnement des caractères sur la page (justification, passage à la ligne).

6°) Réalisation

La technique du wrapping a été utilisée pour le câblage afin d'autoriser facilement les modifications de câblage. Les cartes de l'étape 3 et PCG ont été réalisées à partir de cartes prépercées au format européen 6U (160 mm x 233 mm). Les autres cartes ont été réalisées à partir de cartes (230 mm x 233 mm) ayant un plan de masse côté composants et les alimentations imprimées côté câblage. Ces cartes ont été placées dans deux racks, l'un contenant l'étape 3 formée de 2 à 8 cartes de processeurs de suivi de contour, l'autre contenant le reste de la machine.

La connexion entre cartes d'un même rack a été réalisée par un fond de panier wrappé. La connexion entre les deux racks ou avec la boîte de commande et le kit d'interface en sortie a été réalisée par des câbles plats à 16 fils.

B - PERFORMANCES

Le code utilisé est pratiquement identique à celui certifié par Marc HOURDEQUIN (HOU 78) grâce à un programme de simulation réalisé sur un ordinateur PDP 11/40. D'après les estimations réalisées sur les polices Baskerville et Univers, ce code atteint une compacité de 800 bits/car en moyenne. La principale différence entre le code utilisé et le code de Marc HOURDEQUIN est l'introduction du numéro de paroi et du numéro d'ordre. La qualité apportée par l'utilisation d'une grille de définition de 2000x2000 peut être appréciée sur les planches de caractères présentées et en particulier sur le U Baskerville.

La souplesse du codage et l'effet produit par la mise à l'échelle différente en X et en Y peuvent être appréciés sur la planche présentant le sigle de l'Ecole des Mines de Saint-Etienne.

La mesure de la vitesse de la machine est rendue complexe par le fait que plusieurs paramètres interfèrent à chaque étape pour donner le temps de calcul : cette remarque nous a conduits à concevoir des caractères spéciaux pour effectuer les mesures. Les principaux paramètres sont :

- Pour l'étape 1
  - La longueur totale du fichier du caractère
  - Le nombre de primitives utilisées pour coder le caractère

- Pour l'étape intermédiaire
  - La longueur totale du fichier du caractère
  - Le nombre de droites sur le contour
  - Le nombre de courbes évoluées sur le contour

- Pour l'étape 2
  - La longueur du fichier
  - Le nombre de droites
  - Le nombre de courbes évoluées

- Pour l'étape 3
  - Le nombre de points de naissances
  - L'échelle en X (nombre d'abscisses à engendrer)
  - Le nombre de droites
  - Le nombre de cercles
  - Le taux moyen de recouvrement (nombre de parois par processeur)

Ces mesures ont été effectuées avec les temps de cycles suivants :

- Etape 1		250 ns
- Etape intermédiaire		300 ns
- Etape 2	Partie supérieure	210 ns
	Partie 1	260 ns
	Partie 2	230 ns
- Etape 3		250 ns

L'interprétation des mesures a conduit aux résultats suivants :

- Etape 1

L : longueur en mots de 16 bits de la structure d'arbre
l : nombre de mots hors primitive
p : longueur totale en mots du code des primitives
n : nombre de primitives

$T1\ (\mu S) = 4L + 51 + 30n + 15p$

- Etape intermédiaire

a : longueur en mots de la structure d'arbre + nombre de contours distincts
b : nombre de droites
c : nombre de courbes évoluées

$TI\ (\mu s) = 1,5a + 3,75b + 9c$

- Etape 2

a : longueur en mot de la structure d'arbre + nombre de contours distincts
b': nombre de droites avant la 1ère courbe évoluée + pour chaque courbe évoluée le nombre de droites au dessus de 15 avant la courbe suivante (en général b' est réduit au nombre de droites avant la 1ère courbe évoluée).
c : nombre de courbes évoluées

$T2\ (\mu s) = 1,2a + 3,8b' + 42 \times (C \neq 0) + 58c$

- Etape 3

e : nombre de points de naissance
X : nombre d'abscisses
M : nombre moyen de parois par abscisses
N : nombre de processeurs disponibles
b": nombre de droites après mise à l'échelle (b"=b pour l'échelle 1x1)
c": nombre de cercles après mise à l'échelle (c"=4c pour l'échelle 1x1)

dans le cas N=10 $T3\ (\mu s)=105e + 1/M(32b"+82c")+18X$ (1)

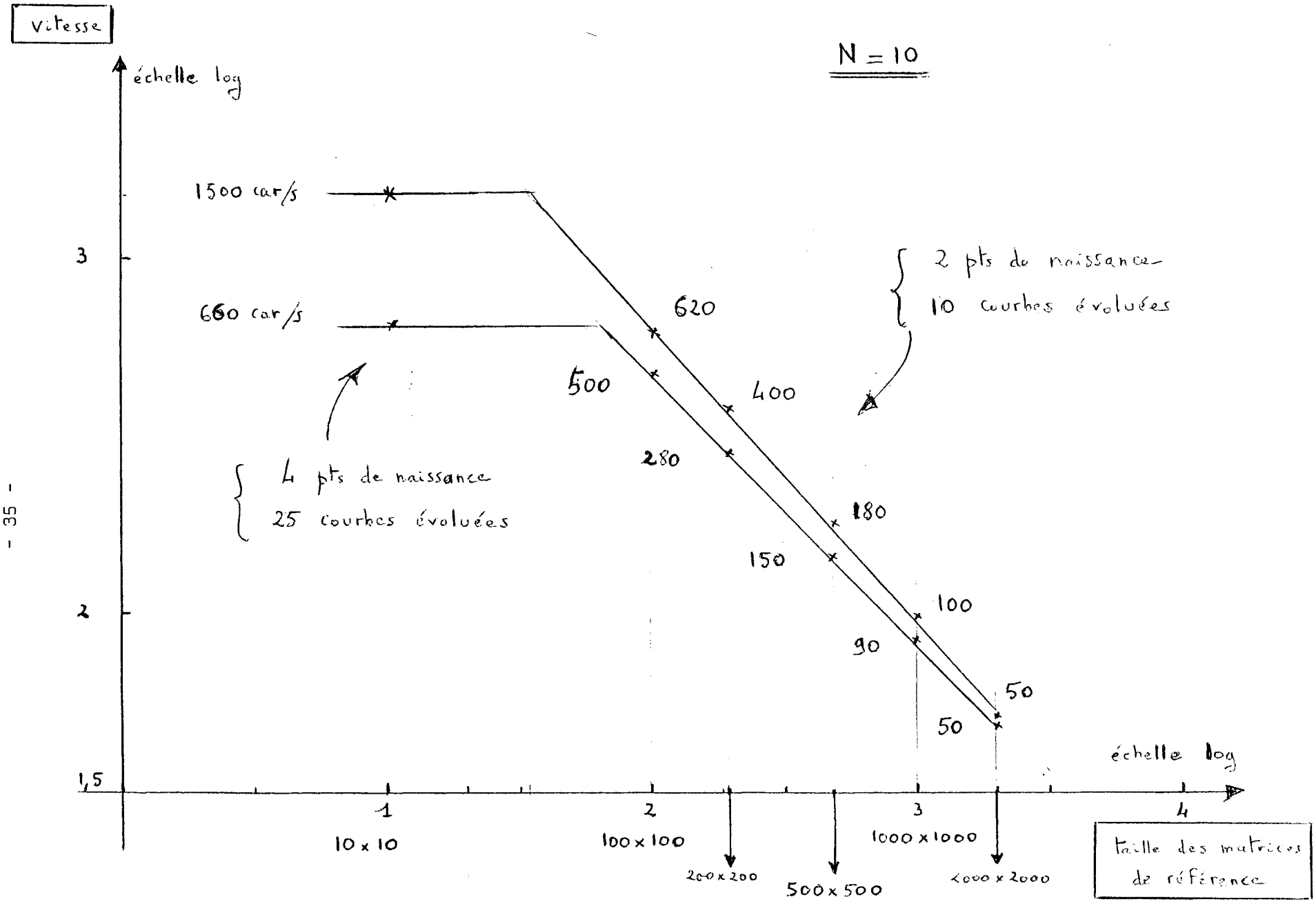
dans le cas N < M $T3\ (\mu s)=250e + 1/N(32b"+82c")+M/N\ 38X$ (2)

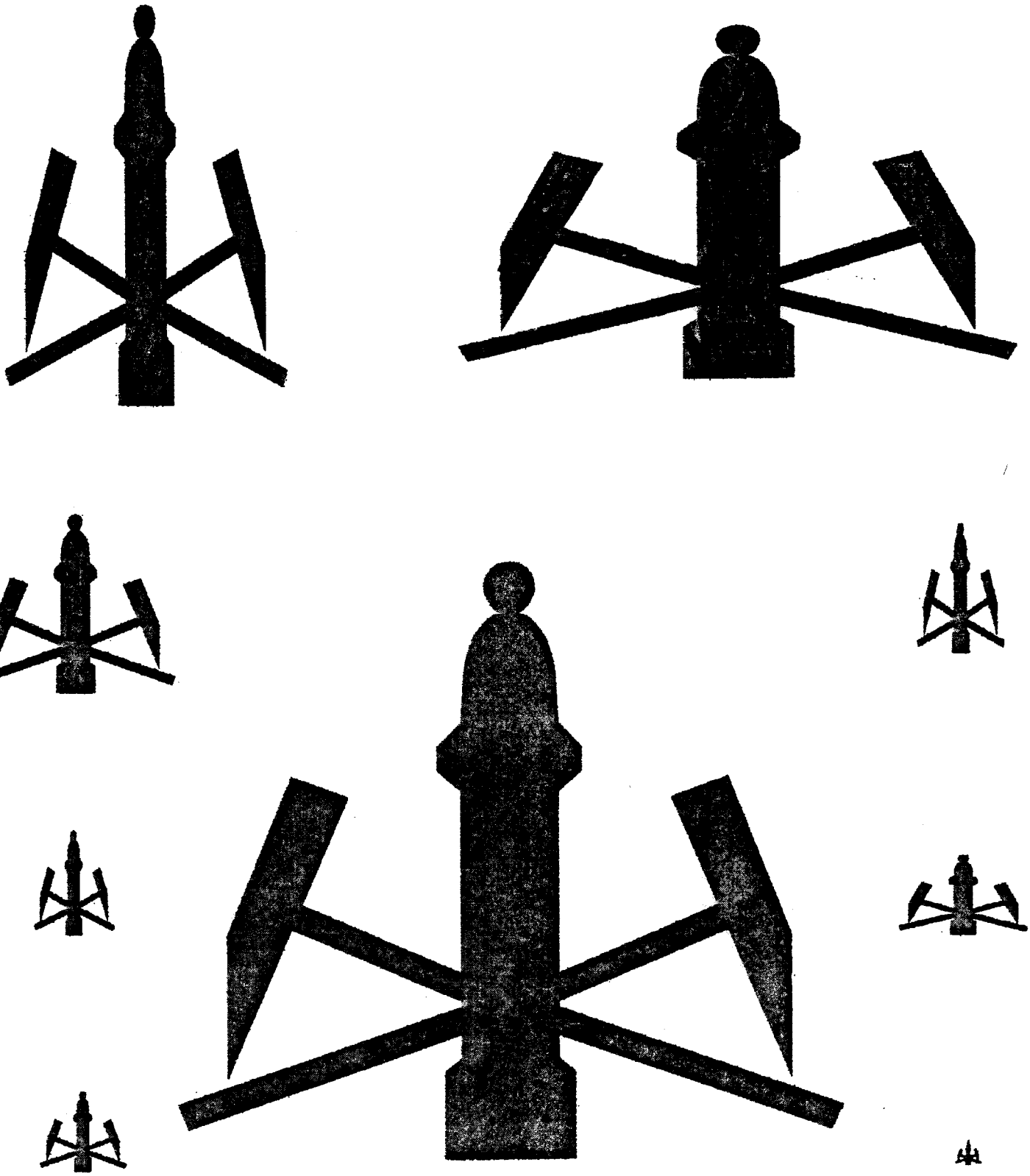
La comparaison de (1) et (2) montre que la gestion de plusieurs parois par processeur dans l'étape 3 est pénalisante en temps. De fait, le contexte des calculs ne peut plus être constamment gardé dans les registres internes du microprocesseur et la vitesse de calcul est réduite de moitié. Ce facteur s'ajoute évidemment au surcroît de travail de chaque processeur mesuré en moyenne par M/N. Si l'on estime M voisin de 5, la vitesse est divisée approximativement par un facteur 3 pour N=4 et par un facteur 4 pour N=2.

Bien qu'elles soient approchées, les formules des temps de calculs sont représentatives des performances du générateur de caractères.

Afin de présenter les résultats sous une forme plus lisible, on a tracé les courbes de vitesse, pour le cas N=10 en fonction d'une échelle homogène en X et en Y, pour deux caractères de complexités différentes couvrant bien la gamme des polices de corps de texte.

On voit que pour une matrice de référence de 100x100, la vitesse du générateur de caractère est de 500 caractères par seconde, ce qui atteint l'objectif visé. D'autre part, la maquette fait appel à la version lente des microprocesseurs utilisés ; le gain théorique est de 2 si on utilise les versions rapides des microprocesseurs et des mémoires associées ; la vitesse du générateur de caractère pourrait donc atteindre les 1000 caractères à la seconde, vitesse de référence des photocomposeuses de haut de gamme.


Vitesse
échelle log
N = 10
1500 car/s
660 car/s
3
2
1,5
620
500
400
280
180
150
100
90
50
50
2 pts de naissance
10 courbes évoluées
4 pts de naissance
25 courbes évoluées
échelle log
1
10 x 10
2
100 x 100
200 x 200
500 x 500
3
1000 x 1000
2000 x 2000
4
Taille des matrices de référence



ABCDEFGHIJKLMN
OPQRSTUVWXYZ

abcdefghijklmnop
qrstuvwxyz

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/9

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/18

ABCDEFGHIJKLMNOPQRSTUVWXYZ

abcdefghijklmnopqrstuvwxyz

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/9

ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/18

UNIVERS MEDIUM 689

&ÆŒœ 0123456789

nopqrstuv ".,:!?

GHIJKLMNO

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/9

&ÆŒœ ".,:!?? 0123456789

nopqrstuv GHIJKLMNO

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/18

&ÆŒœ ".,:!? 0123456789

nopqrstuv GHIJKLMNO

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/9

&ÆŒœ ".,:!?? 0123456789 *nopqrstuv* *GHIJKLMNO*

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/18

ABCDEFGHI JKL
MNOPQ RSTUV
WXYZ

abcdef ghi jklmn op
qrstuvwxyz

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/9

ABCDEFGHI JKL MN OPQ RST

UVWXYZ

abcdef ghi jklmnopqrstuvwxyz

Epreuve imprimante, rapport de réduction 1/1
Réduction photographique 1/18

BASKERVILLE 169

ABCDEFGHIJKLMNOPQRSTU

VWXYZ

abcdef ghi jklmn opqr stuvwxyz

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/9

ABCDEFGHIJKLMN OPQRSTUVWXYZ

abcdef ghi jk lmn opqrst uvwxyz

Epreuve imprimante, rapport de réduction 1/2
Réduction photographique 1/18

BASKERVILLE 169

Chapitre 2

CHAPITRE 2

INTRODUCTION AUX PROBLEMES DE NATURE GEOMETRIQUE

I.-POSITION DU PROBLEME

II.-COURBE EVOLUEE

1.-Définition

2.-Décompositon en cercles

3.-Respect des contraintes

III.-PRESENTATION DES AUTRES CHAPITRES

INTRODUCTION AUX PROBLEMES DE NATURE GEOMETRIQUE

I.-POSITION DU PROBLEME

La fonction du générateur de caractères présenté au chapitre précédent est de produire des formes à la demande. Pour cela il part d'une description condensée des caractères pour en déduire une suite de plages compatible avec les techniques actuelles de visualisations par trames.

La définition des caractères dans des grilles de 2000X2000 nous a conduit à les décrire à l'aide d'un code par contour. La génération des plages d'un caractère en une seule passe impose d'une part de rajouter à ce code par contour une structure d'arbre, qui apporte des informations globales sur la forme, et d'autre part d'effectuer les calculs sur plusieurs processeurs en parallèles.

Les problèmes liés au calcul de ces informations globales et au parallélisme ont été étudiés par M. BLOCH. (BLO 81) Dans cette étude nous nous intéressons aux méthodes et moyens utilisés pour coder le contour d'un caractère, ainsi qu'aux calculs nécessaires pour décoder celui-ci. Le premier problème à résoudre est le choix des interpolants qui permettent de respecter les contraintes suivantes:

-Continuité des tangentes
-Débit de la machine
-Facilité de codage
-Souplesse du code
-Densité du code

Continuité des tangentes

Le codage par contour le plus simple est celui qui utilise des segments de droites comme interpolants. Le principal désavantage de ce code est qu'il rend nécessaire d'établir un compromis entre la densité et la continuité des tangentes.

En effet, utiliser un petit nombre de droites donne un code dense mais le contour présente alors des variations brutales du vecteur tangent; respecter la continuité de la tangente produit un résultat de meilleure qualité mais au prix d'un grand nombre de droites et par conséquent d'un code peu dense.

De plus, le nombre de droites nécessaires pour coder un contour augmente avec la taille de la grille de définition. Pour illustrer ce phénomène, considérons un cercle de rayon R que nous voulons approcher par un polygone. Avec un écart maximum toléré e entre la courbe d'origine et les côtés du polygone, nous arrivons à un nombre de droites proportionnel à la racine carrée du rayon:

$N > \pi (R/e)^{1/2}$

Ces deux remarques montrent que l'utilisation de droites est insuffisante, nous devons donc utiliser un interpolant d'un ordre supérieur.

Débit de la machine

Selon le principe de la struture pipe-line adoptée, ce débit doit être respecté au niveau de l'étape finale de suivi des contours, mais aussi au cours des opérations préliminaires de transformation du code initial en code intermédiaire.

Le temps de calcul alloué pour passer d'un point à un autre au cours de l'étape de suivi des contours est de 10 µs. Dans ces conditions les seuls interpolants utilisables sont les droites et les cercles pour lesquels il existe des algorithmes rapides permettant de calculer la variation suivant un axe à chaque incrémentation sur l'autre axe. (BRE 77, HOR 77) A l'entrée de cette étape nous voulons donc un code intermédiaire de description des contours ne contenant que des droites et des cercles.

Le moyen le plus simple pour respecter le débit de la machine est d'adopter le code intermédiaire comme code initial. Les contraintes suivantes ne permettent pas cette solution.

Facilité de codage

Indiquer les coordonnées des extrémités d'une droite est une opération simple pour un opérateur; par contre déterminer sur un contour les points de changement d'arcs de cercles et calculer les centres correspondants est une opération longue et complexe. De plus, la qualité du résultat est liée à la précision avec laquelle ces points ont été calculés. Cette remarque a conduit M. HOURDEQUIN (HOU 78, HOC 79) à définir une méthode automatique de découpage des courbes en arcs de cercles.

Nous pourrions toutefois imaginer une solution très simple qui consiste à appliquer ce découpage une seule fois lors du codage du caractère. Cette opération, étant réalisée hors ligne, n'aurait plus d'influence sur le débit du générateur.

Souplesse du code

Par souplesse du code nous entendons la possibilité d'effectuer des transformations géométriques, comme la mise à l'échelle avec facteurs en X et en Y. L'utilisation de cercles pour coder les contours interdit cette opération. Les cercles sont, en cas de facteurs différents en X et en Y, transformés en ellipses et ne sont plus acceptables pour l'étape de suivi des contours.

Nous sommes donc contraints d'utiliser un codage plus évolué en entrée et d'effectuer en ligne un transcodage après mise à l'échelle.

Densité du code

Cette contrainte est rappelée ici pour mémoire car il s'agit d'un paramètre important du générateur de caractères. Le codage par courbes doit donc apporter une compression par rapport au codage par cercles.

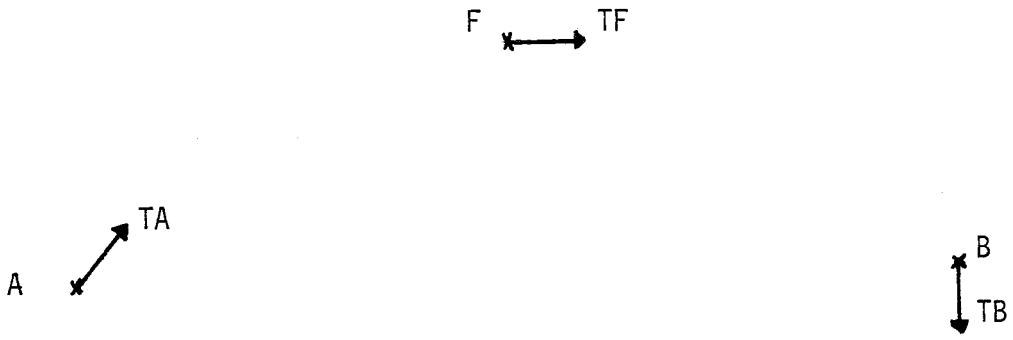


Fig 1

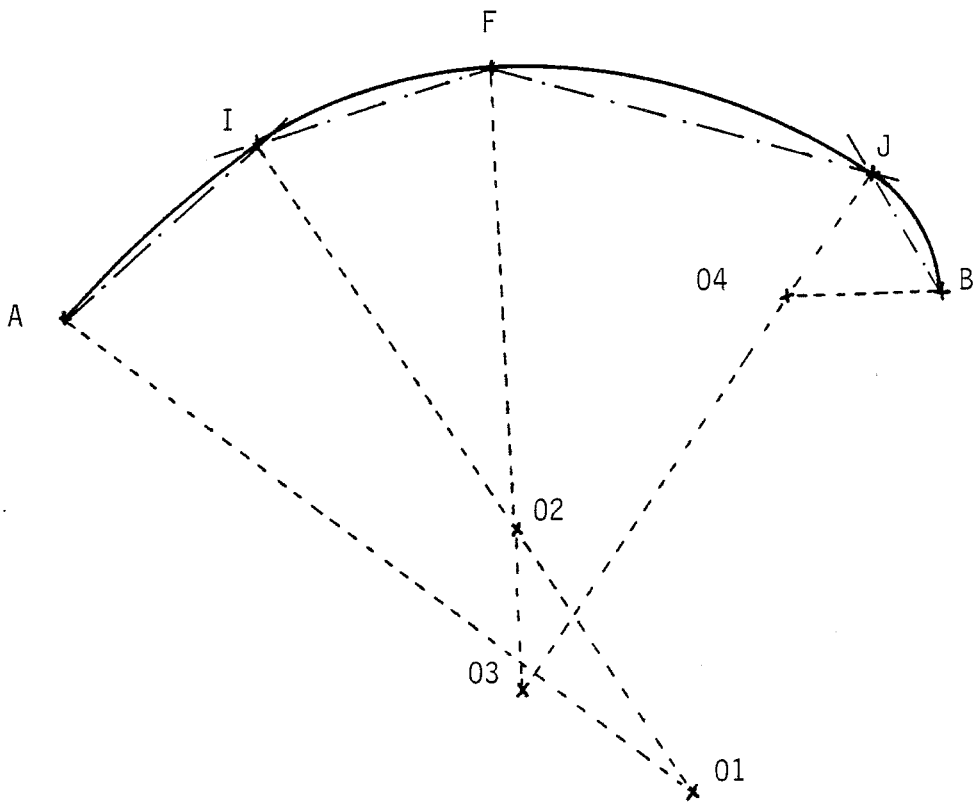


Fig 2

II.-COURBE EVOLUEE

1.-Définition

C'est une courbe de classe C1 sans point d'inflexion et dont la variation du vecteur tangent est inférieure à π.

Paramètres d'entrée

La courbe est définie par ses deux extrémités A, B, un point intermédiaire F appartenant à la courbe et les angles des vecteurs tangents en ces points. (Fig 1) Lorsque la tangente en F est parrallèle à la droite AB, nous dirons que F est le point de carrure. (HOU 78)

Résultats

Nous remplaçons cette courbe par quatre arcs de cercles visuellement très proches de celle-ci. (Fig 2)

Pour réaliser cette substitution nous devons déterminer deux points intermédiaires I, J, de changement d'arc de cercle, l'un étant compris entre les points A et F, l'autre entre F et B; ainsi que les quatre centres de cercles.

2.-Décomposition en cercles

Points de changement d'arc

Nous fixons tout d'abord la tangente TI commune, au point I, aux arcs de cercles issus des points A et F; le choix de celle-ci sera précisé plus loin.

Nous recherchons le lieu des points où un cercle, partant du point A avec une tangente TA, arrive avec une tangente TI. (Fig 3)

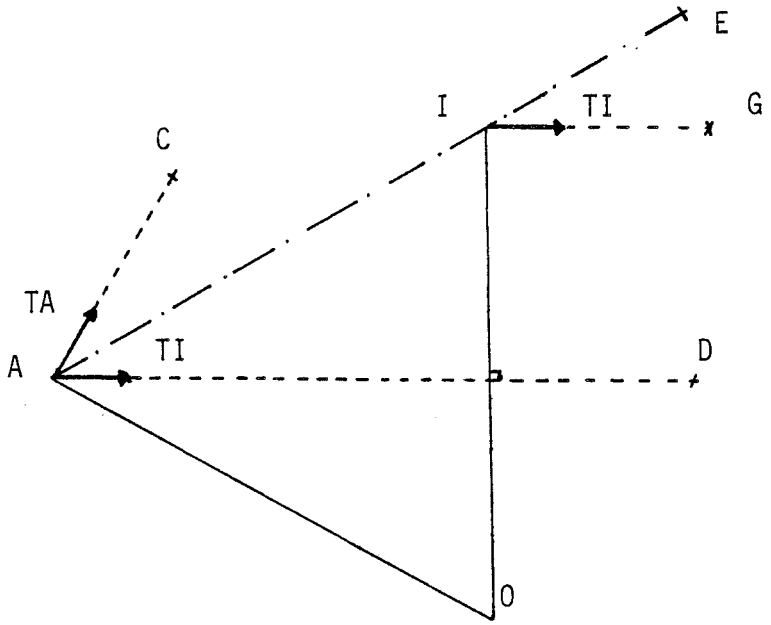


Fig 3

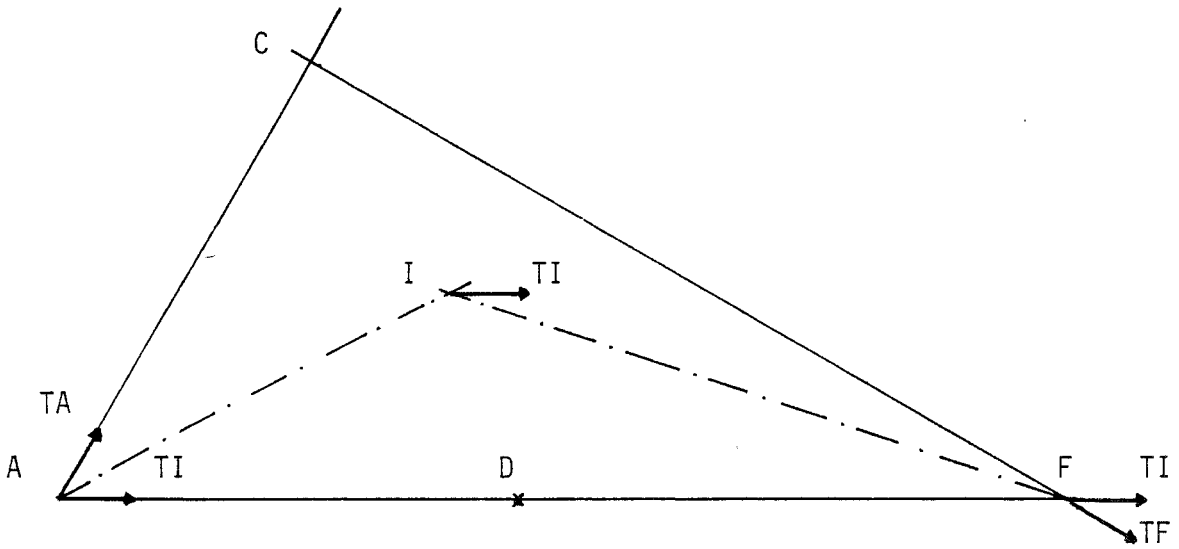


Fig 4

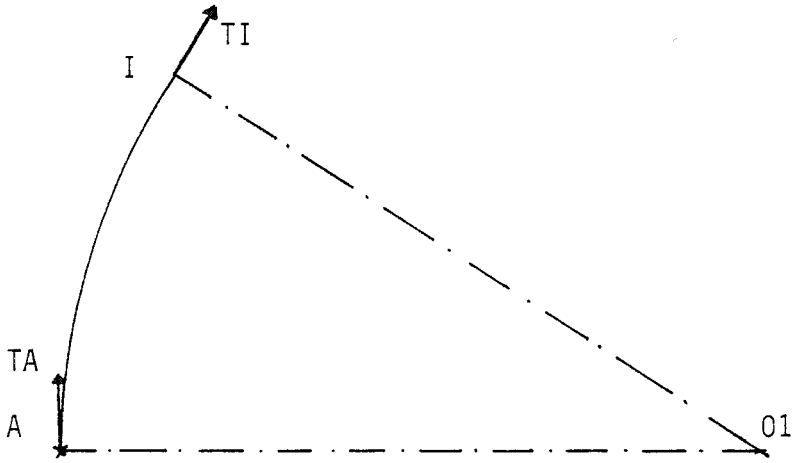


Fig 5

Soit I appartenant à ce lieu, TA est représentée sur la figure par le segment de droite AC, TI par le segment de droite AD, O est le centre du cercle passant par A et par I.

Par définition le segment IO est perpendiculaire à AD et AO à AC:

$$\widehat{CAI} = \pi/2 - \widehat{IAO}$$

$$\widehat{IAD} = \pi/2 - \widehat{OIA}$$

D'autre part les points I et A étant sur un cercle, les angles $\widehat{IAO}$ et $\widehat{OIA}$ sont égaux et nous avons donc $\widehat{CAI} = \widehat{IAD}$.

Réciproquement nous considérons un point I appartenant à la bissectrice de l'angle $\widehat{CAD}$.

Soit O l'intersection des droites perpendiculaires aux tangentes en A et I. Par construction:

$$\widehat{CAI} = \widehat{IAD} = \widehat{EIG}$$

$$\widehat{IAO} = \pi/2 - \widehat{CAI}$$

$$\widehat{AIO} = \pi/2 - \widehat{EIG}$$

Nous avons donc $\widehat{IAO} = \widehat{AIO}$. Les segments de droites AO et IO sont égaux. Les points I et A sont sur un cercle de centre O.

Le lieu des points recherché est donc la bissectrice de l'angle $\widehat{CAD}$.

Par un raisonnement similaire, le point de changement d'arc de cercle I doit se trouver sur la bissectrice de l'angle $\widehat{CFD}$; il est donc situé à l'intersection de ces deux bissectrices. (Fig 4)

A chaque tangente TI correspond une solution, c'est à dire un point I particulier; cette remarque sera exploitée dans le chapitre sur la méthodologie de décomposition des courbes évoluées. Sur la figure nous avons représenté la solution qui considère I comme un point de carrure de la courbe A-F pour lequel la tangente TI est parallèle à la corde A-F, D appartient au segment AF.

Centre des cercles

Il s'agit simplement de trouver un centre de cercle en connaissant deux points appartenant au cercle et les tangentes en ces points. Ce qui revient à calculer l'intersection de deux droites qui sont les rayons en ces points. (Fig 5)

En conclusion, le passage d'une courbe évoluée à des cercles demande le calcul de quatre bissectrices, la résolution de deux systèmes linéaires pour déterminer les points intermédiaires et la résolution de quatre systèmes linéaires pour déterminer les centres de cercles.

3.-Respect des contraintes

Continuité des tangentes

La décomposition des courbes évoluées en cercles donne, par construction, des tangentes continues aux points de changement d'arc de cercles I, F et J. La continuité aux extrémités A et B, entre deux courbes évoluées consécutives, est assurée par les paramètres d'entrée.

Débit de la machine

Le débit de la machine au niveau de l'étape de suivi des contours est respecté car nous sommes capables de calculer au rythme voulu un fichier ne contenant que des cercles et des droites. Les droites contenues dans ce fichier correspondent à des portions de contour rectilignes et sont présentes dans le fichier initial.

En ce qui concerne le débit de l'étape d'expansion des courbes évoluées, nous consacrons un chapitre complet à son étude relativement complexe. (Chap. 3)

Facilité de codage

Elle est liée à la facilité de définir les paramètres d'entrée des courbes évoluées. Les coordonnées à déterminer appartiennent au contour, la recherche des tangentes peut être facilement réalisée à l'aide d'une règle.

Nous étudierons dans un autre chapitre la possibilité de réaliser automatiquement cette opération de codage des contours. (Chap 4)

Souplesse du code

La mise à l'échelle avec des coefficients différents en X et en Y modifie les paramètres d'entrée d'une courbe évoluée, les coordonnées et les angles, mais le résultat est toujours une courbe évoluée décomposable en arcs de cercles.

Nous verrons dans un prochain chapitre (Chap 5) que le codage par courbes évoluées est très souple puisqu'il autorise de nombreuses autres transformations géométriques.

Densité du code

Le codage de quatre cercles demande seize mots alors que le codage de la courbe évoluée telle qu'elle a été présentée dans ce chapitre en demande sept. Nous verrons qu'en choisissant le point F de telle façon que l'angle de la tangente soit calculable à partir des autres paramètres nous pouvons gagner un mot. De plus la représentation des coordonnées n'utilise que douze bits sur les seize de la machine, nous pouvons donc utiliser les quatre bits restant pour coder les tangentes des extrémités dans le cas très fréquent où celles-ci sont des valeurs simples. Ces deux remarques nous permettent de comprimer par un facteur trois le code initial.

III.-PRESENTATION DES AUTRES CHAPITRES

Nous parlerons d'abord dans le chapitre 3 des études préliminaires, de la réalisation et des performances de l'étape chargée de l'expansion des courbes évoluées dans le générateur de caractères tel qu'il a été effectivement construit.

Dans le chapitre 4 nous étudierons la réalisation d'un logiciel de codage automatique des contours. L'étude de ce logiciel qui concerne un travail en amont du générateur de caractères permet de simplifier et d'abaisser le coût de ce travail.

Dans le chapitre 5 nous présenterons les diverses transformations géométriques qui peuvent être placées dans le générateur de caractères de façon à élargir son champ d'application.

Enfin nous verrons dans le dernier chapitre deux variantes pour le décodage des caractères; l'une a pour objectif d'accélérer le tracé des cercles, l'autre propose de remplacer la décomposition des courbes évoluées en cercles par une décomposition en ellipses.

Chapitre 3

CHAPITRE 3

METHODOLOGIE DE DECOMPOSITION DES COURBES EVOLUEES

I.-PROBLEMES DE REALISATION

II.-ETUDES PRELIMINAIRES

1.-Utilisation de circuits rapides

2.-Recherche d'algorithmes efficaces

2.1-Calcul par cosinus directeurs

2.2-Calcul par les angles

2.3-Choix du point F

2.4-Précision des calculs sur les angles

2.5-Choix des points intermédiaires

2.6-Problèmes liés au suivi des contours

2.7-Précision des calculs

3.-Mise en parallèle des calculs

III.-REALISATION

IV.-ANALYSE DES PERFORMANCES

METHODOLOGIE DE DECOMPOSITION DES COURBES EVOLUEES

I.-PROBLEMES DE REALISATION

Le point de départ de ce projet était une simulation du décodeur sur un ordinateur, qui avait démontré la faisabilité du produit. La réalisation d'une machine à base de microprocesseurs avait pour objectif d'augmenter le débit du décodeur pour atteindre les 500 caractères/s et potentiellement les 1000 caractères/s, ce qui correspond à une photocomposeuse de classe moyenne. Cette contrainte de vitesse très forte nous a conduits à utiliser une structure pipe-line en découpant le décodage en une succession d'étapes. Pour illustrer ce problème de vitesse au niveau de l'expansion des courbes évoluées, il suffit de comparer les temps de calcul entre les deux applications. Les mesures effectuées sur la simulation donnent 5 ms/courbe évoluée, et le débit de 500 car/s exige 66 µs/courbe évoluée au niveau du prototype.

L'obtention de ce gain de 75 nous a amenés à étudier les trois solutions classiques pour accélérer les calculs:

-Utilisation de circuits rapides

-Recherche d'algorithmes plus rapides

-Mise en parallèle des calculs

De plus, au niveau de cette étape apparait un problème de précision des calculs si nous excluons, pour des raisons de vitesse, le calcul en réel classique utilisé lors de la simulation.

Dans ce chapitre nous présentons les études préliminaires sur la rapidité et la précision des calculs, la réalisation effective et les performances obtenues.

II.-ETUDES PRELIMINAIRES

1.-Utilisation de circuits rapides

L'expansion des courbes évoluées demandant la résolution de systèmes linéaires, il n'est pas possible d'utiliser les microprocesseurs 16 bits actuels qui nécessitent entre 10 et 20 µs pour exécuter une multiplication de deux entiers. Nous nous sommes alors intéressés aux microprocesseurs en tranches qui sont intrinsèquement plus rapides et qui permettent d'optimiser les chemins de données et les opérations en fonction de l'application. Pour des raisons de disponibilité et de simplicité de mise en oeuvre nous avons choisi les circuits de la famille 2900 en logique TTL.

Le système de simulation de mémoire de microprogrammes que nous utilisons est limité à un temps de cycle de 200 ns. Il est à noter que cette limite est avec les circuits actuellement disponibles, tout à fait raisonnable. La recherche d'un temps de cycle plus court, pour porter le débit de la machine à 1000 caractères/s, n'a pas été entreprise dans ce projet. Elle sera facilitée par l'utilisation de circuits à technologie ECL interne qui arrivent sur le marché et qui permettent un gain de 25% sur le temps de traversée des circuits entièrement TTL. (2901-C AMD)

Sachant par ailleurs que la résolution d'un système linéaire demande cinq mutiplications, nous avons été conduits à utiliser un multiplieur rapide qui réalise une multiplication 16X16 bits en 120 ns. (TDC 10 10 J de TRW)

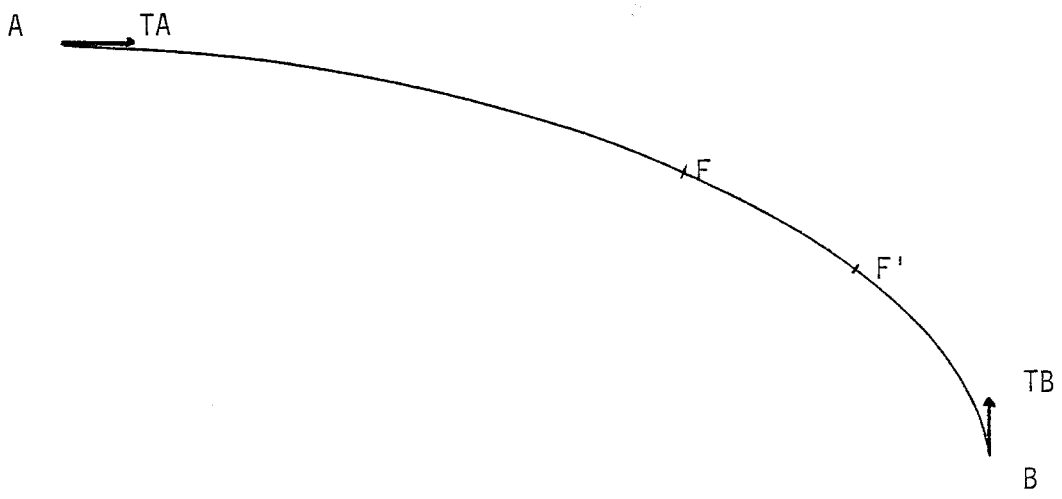
Une fois choisie la technologie de réalisation nous avons estimé le temps de calcul des opération élémentaires. Ces estimations sont présentées dans le tableau de la figure 1.

-Les décalages d'un bit sont effectués, en dehors de l'unité arithmétique et logique, à l'aide de multiplexeurs, ce qui permet de réaliser une addition suivie d'un décalage au cours du même cycle.

TABLEAU RECAPITULATIF DES TEMPS DE CALCUL

Opérations	Temps en µs
Addition et/ou décalage	0,2
Carré	1
Multiplication	1,2
Division	3,4
Racine carrée	4
Recherche en table	0,6

Fig 1



F : Point de carrure

F': Point à tangente moitié

Fig 2

-Le temps donné pour la multiplication comprend le chargement des opérandes, la normalisation et la récupération du résultat, ce qui explique la différence avec le temps de traversée du circuit.

-Le temps donné pour la division correspond à un algorithme de division sans restauration de deux entiers positifs de 16 bits.

Ces estimations vont nous permettre de comparer les diverses solutions envisagées et la plupart des améliorations proposées n'ont de sens qu'au vu de ce tableau. Il est d'autre part prévisible que les temps obtenus seront doublés, lors de l'implémentation effective des algorithmes, par les opérations d'entrée sortie et les tests.

2.-Recherche d'algorithmes plus efficaces

Dans ce paragraphe nous apportons des réponses aux problèmes de vitesse et précision posés par la réalisation.

En ce qui concerne la vitesse, nous présentons l'algorithme utilisé lors de la simulation et une variante plus rapide. Nous analysons ensuite le choix de la position du point F et de celles des points intermédiaires. Ces diverses propositions nous permettent de passer successivement d'un temps d'expansion d'une courbe évoluée de 113,8 µs à 74 µs, 70 µs et enfin 62 µs.

En ce qui concerne la précision, nous nous intéressons aux angles et à la résolution des systèmes linéaires. La précision sur les angles est importante car elle fixe la taille des tables trigonométriques. Elle nous indique également des limites sur les calculs des points intermédiaires et nous permet d'étudier des variantes sur ces derniers. L'étape finale de suivi des parois nous impose une précision très forte sur les calculs. Pour répondre à cette contrainte nous utilisons des calculs en réels et nous définissons les formats des opérandes. L'étude de la propagation des erreurs lors de la résolution des systèmes linéaires nous permet de vérifier le respect de cette contrainte.

2.1-Calcul par cosinus directeurs

C'est l'algorithme utilisé lors de la simulation logicielle du générateur de caractères. Nous considérons comme données les extrémités A, B de la courbe, le point de carrure F et les cosinus directeurs CA, SA et CB, SB des tangentes aux extrémités.

Calculs annexes

Ils consistent à trouver les cosinus directeurs en F et aux points intermédiaires I et J. Par définition F est le point de carrure de la courbe, la tangente est parallèle à la corde AB. Ainsi:

$$CF = (XB - XA) / ((XB - XA)^2 + (YB - YA)^2)^{1/2}$$

$$SF = (YB - YA) / ((XB - XA)^2 + (YB - YA)^2)^{1/2}$$

Les points I et J sont considérés comme les points de carrures des demi-courbes AF et FB. Les cosinus directeurs sont donnés par des formules analogues.

Calculs des points

La recherche des points intermédiaires I et J ainsi que des quatre centres de cercles demandent la résolution de systèmes linéaires 2X2. Nous allons nous intéresser à la résolution d'un système particulier, celui qui permet de calculer le point I. Les autres sont similaires et peuvent très facilement se déduire de l'exemple.

Soit X,Y les coordonnées du point recherché, le système le plus général est:

$$Y\ CAI - X\ SAI = YA\ CAI - XA\ SAI$$

$$Y\ CIF - X\ SIF = YF\ CIF - XF\ SIF$$

Le calcul en coordonnées relatives par rapport au point A nous permet de remplacer deux multiplications par quatre additions. En considérant XF, YF, X et Y en relatif, nous obtenons:

$$Y\ CAI - X\ SAI = 0$$

$$Y\ CIF - X\ SIF = YF\ CIF - XF\ SIF$$

Par substitution:

X = CAI (YF CIF - XF SIF) / (SAI CIF - CAI SIF)

Y = SAI " " "

Si nous récapitulons toutes les opérations, nous obtenons 59 additions, 3 racines carrées, 6 carrés, 12 divisions et 36 multiplications. D'après le tableau de la figure 1 nous arrivons à un temps de calcul de 113,8 µs. En doublant l'estimation pour tenir compte des entrées sorties et des tests nous devons utiliser trois processeurs en parallèle pour respecter le temps de calcul de 66 µs.

2.2-Calcul par les angles

Nous supposons identiques les données A, B, F et les cosinus directeurs sont remplacés par les angles DA, DB des tangentes A et B.

Calculs annexes

Le point F est le point de carrure de la courbe AB.

DF = Artg ((YB - YA) / (XB - XA))

Les points I et J sont les points de carrure des demi courbes.

DI = Artg ((YF - YA) / (XF - XA))

DJ = Artg ((YB - YF) / (XB - XF))

Les angles des bissectrices sont égaux à la moyenne des angles de base.

Calculs des points

Pour illustrer les calculs nous prenons le même exemple, le calcul des coordonnées du point I en relatif par rapport au point A.

X sin DAI - Y cos DAI = 0

X sin DIF - Y cos DIF = XF sin DIF - YF cos DIF

Par substitution:

X=cos DAI (YF cos DIF-XF sin DIF)/(sin DAI cos DIF-cos DAI sin DIF)

Y=sin DAI " " "

Il est impensable de calculer les fonctions trigonométriques utilisées; nous utilisons des tables enregistrées dans des mémoires mortes.

Le dénominateur forme une identité remarquable, nous avons:

cos DIF sin DAI - cos DAI sin DIF = sin (DAI - DIF)

Nous remplaçons deux multiplications par une recherche en table. Si de plus nous disposons de la fonction précalculée 1/sin X, nous remplaçons une division par une multiplication.

En tenant compte de ces remarques nous obtenons pour cette méthode de calcul 40 additions, 3 divisions, 30 multiplications et 33 recherches en tables; soit un temps de calcul de 74 µs.

Cette méthode présente trois avantages par rapport à la précédente: -Un gain de 35% sur le temps de calcul

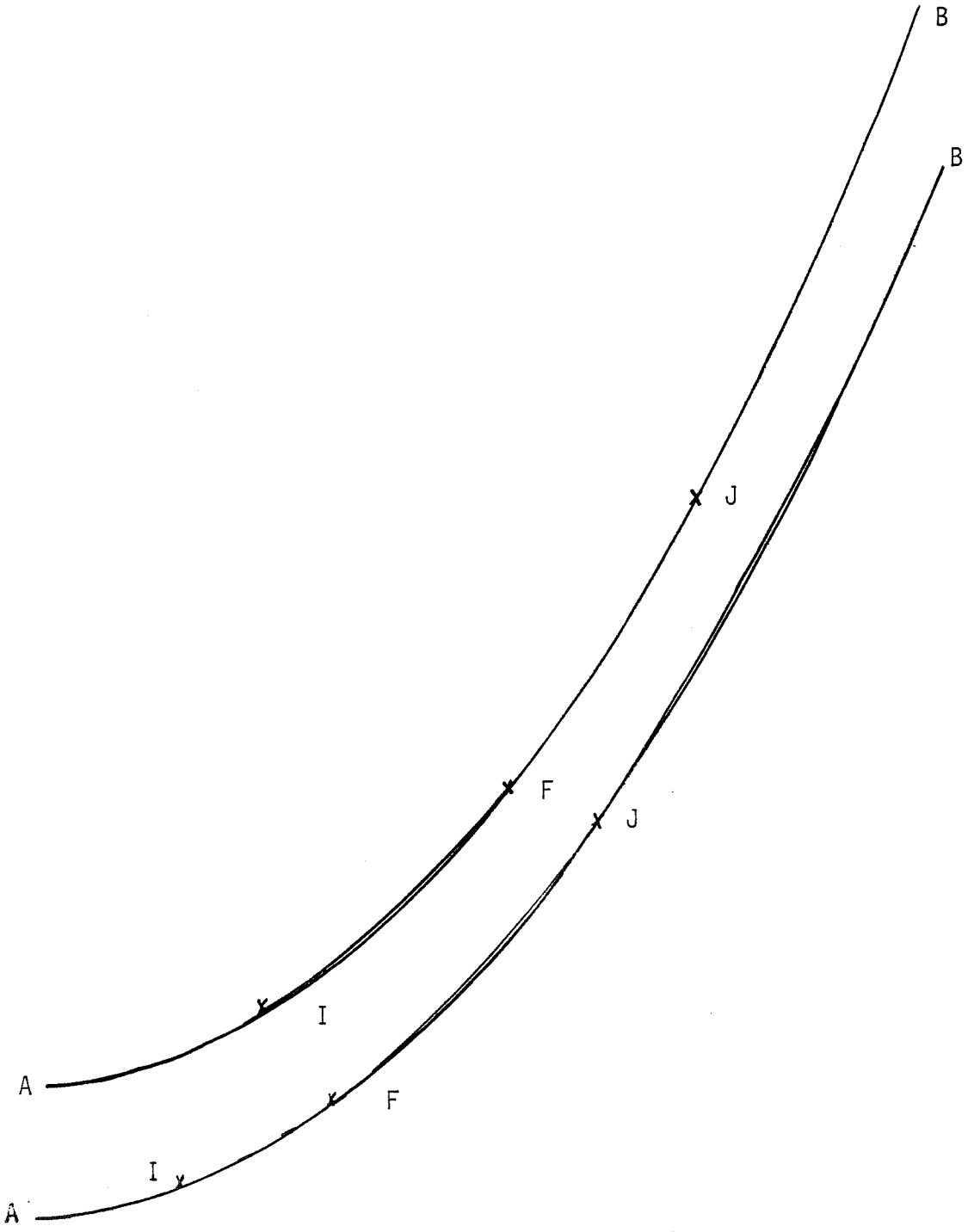
-Un gain de deux mots par courbe évoluée dans le fichier d'entrée puisque nous remplaçons les cosinus directeurs par les angles.

-Les formats des termes de tous les systèmes linéaires sont identiques.

L'unique inconvénient de cette méthode est qu'elle utilise des tables trigonométriques, ce qui augmente le nombre de circuits. Cet inconvénient est compensé par le gain sur le temps de calcul qui permet de réduire de trois à deux le nombre de processeurs à utiliser. Il est à noter également que la réalisation de tables trigonométriques est une opération très simple.

2.3-Choix du point F

Dans la démonstration du passage des courbes évoluées aux cercles, la position du point F n'est pas imposée. En fait, il suffit que ce point appartienne à la courbe et que sa tangente soit connue. Pour gagner un mot par courbe évoluée dans le fichier d'entrée nous nous intéressons aux points dont nous pouvons calculer la tangente à partir des autres paramètres. Le point F doit rester "représentatif", c'est à dire caractériser la concavité de la courbe, mais ce dernier critère étant totalement subjectif nous ne le considèrerons pas explicitement. (Fig 2)

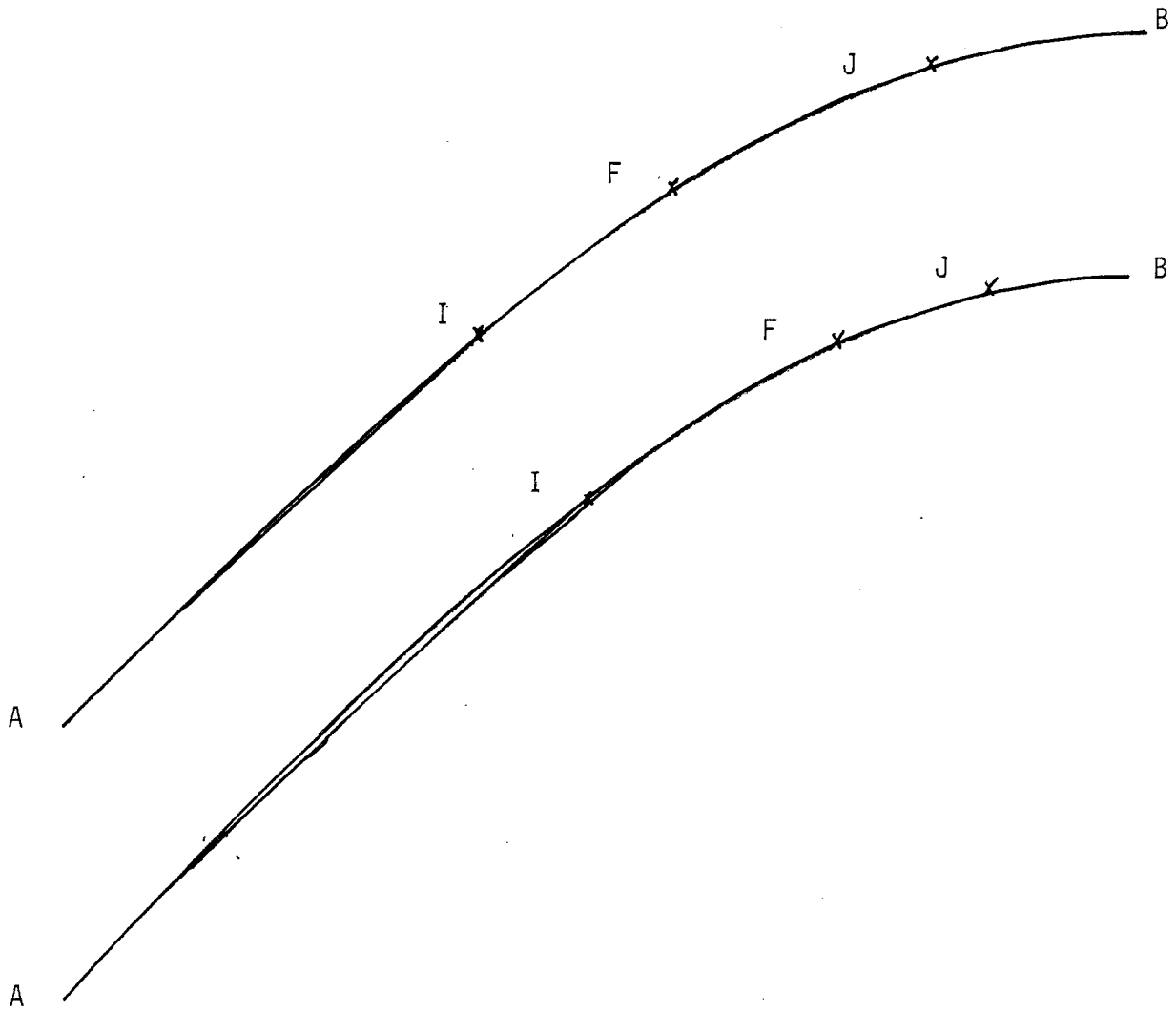


Tracé de la fonction X^2 entre 0 et 1 et de l'approximation par les cercles.

F point de carrure, courbes du haut.

F point de tangence 1/2, courbes du bas.

Fig 3



Tracé de la fonction sin X entre 0 et $\pi/2$ et de l'approximation par les cercles.

F point de carrure, courbes du haut.

F point de tangence 1/2, courbes du bas.

Fig 4

Nous avons jusqu'à présent pris comme point F le point de carrure qui est à la fois représentatif et facile à déterminer ou à calculer lors du codage du contour. La tangente étant parallèle à la corde AB nous avons:

DF = Artg ((YB - YA) / (XB - XA))

Le calcul de la tangente au point de carrure repose sur les coordonnées des extrémités; nous pouvons également obtenir un angle en faisant la moyenne des angles des tangentes aux extrémités. Le calcul est ramené à une addition au lieu d'une division, deux additions et une recherche en table; soit une diminution du temps de calcul de 4 µs et un gain de 6%. Par contre la détermination de ce point de tangente moitié rend le codage du caractère plus délicat.

Pour comparer ces deux points nous avons simulé le remplacement de courbes analytiques par des cercles en prenant le point F comme point de carrure ou comme point de tangence moitié. (Fig 3-4)

Les résultats de cette simulation ne permettent pas de préférer réellement l'un de ces points pour sa "représentativité".

2.4-Précision des calculs sur les angles

Dans la démonstration et les calculs qui précédent, les angles des tangentes aux points de changement d'arc I et J ont été donnés par les droites AF et FB, ce qui correspond à une solution très sûre. (Fig 5) En effet la définition des courbes évoluées impose que la variation de l'angle du vecteur tangent entre les points A et B soit inférieure à π . Les droites tangentes en A et F ont donc un point d'intersection C et forment avec la corde AF un triangle. le point I, intersection de deux bissectrices d'un triangle, est intérieur au triangle; nous sommes ainsi assurés que les cercles seront intérieurs au triangle ACF, ce qui correspond à une interpolation visuellement satisfaisante. Un raisonnement similaire peut être fait pour le point J.

Nous allons maintenant examiner avec quelle précision les angles doivent être calculés pour respecter la contrainte imposée au point de changement d'arc d'être intérieur au triangle.

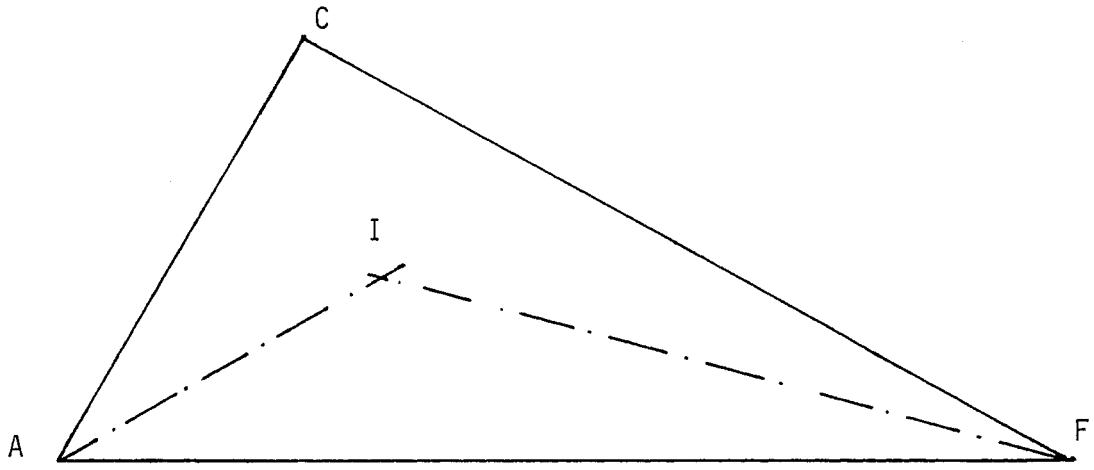


Fig 5

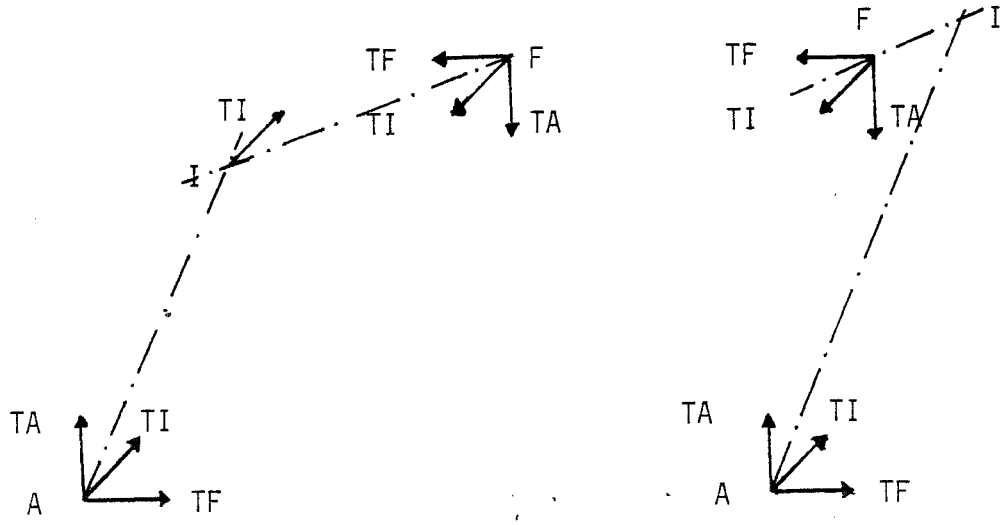


Fig 6

ANGLE	CALCUL	ERREUR
0	0	0
4.5	3.54158	-.958423
9	7.1273	-1.8727
13.5	10.8035	-2.69646
18	14.6214	-3.37861
22.5	18.6396	-3.86039
27	22.9286	-4.07136
31.5	27.576	-3.92396
36	32.6944	-3.30558
40.5	38.4336	-2.06638
45	45	0
45	45	0
49.5	51.5664	2.06638
54	57.3056	3.30559
58.5	62.424	3.92396
63	67.0713	4.07135
67.5	71.3604	3.8604
72	75.3786	3.37862
76.5	79.1965	2.69646
81	82.8727	1.8727
85.5	86.4584	.958426
90	90	0

Fig 7

Nous considérons DAF = Arctg ((YF - YA) / (XF - XA))

DI = une approximation de DAF

DA et DF des angles justes

Pour que le point I soit intérieur au triangle il faut et il suffit que DAI soit entre DA et DAF et que DIF soit entre DF et DAF

L'approximation de DAF par DI se traduit par:

DAI = (DA + DI) / 2 DIF = (DF + DI) / 2

Dans le cas où DA > DAF > DF, le respect de la contrainte sur I entraine:

$$DA > (DA + DI) / 2 > DAF > (DF + DI) / 2 > DF$$

Ce qui impose $|DI - DAF| < |DA - DAF|$

Une étude complète de tous les cas de figure conduit à la conclusion:

$$|DAF - DI| < |DA - DAF| \text{ et } |DAF - DI| < |DF - DAF|$$

La précision des calculs se trouve ainsi reliée à des contraintes au niveau de l'éditeur de caractères. Pour une précision donnée, il faut que les écarts entre les angles soient supérieurs à celle-ci. Si nous considérons les angles donnés avec la même tolérance que les résultats des calculs, la contrainte s'énonce: les angles DA, DAF, DF, DFB et DB doivent tous différer entre eux d'une valeur supérieure à cette tolérance.

La taille des tables trigonométriques est donnée par le nombre de bits utilisés pour coder les angles. Nous avons donc intérêt, pour minimiser la taille des tables, à prendre la précision la plus faible qui satisfasse les utilisateurs. Nous avons considéré qu'un écart de 1/2 degré n'était pas visuellement décelable; cette décision peut paraitre très brutale mais il faut considérer que ces dessins sont en pratique reproduits à une échelle très réduite où une meilleure précision devient superflue. Comme la connaissance des valeurs trigonométriques entre 0 et $\pi/4$ permet une rapide extension de $-\pi/2$ à $\pi/2$, la taille des tables a donc été fixée à 128 mots répartis entre 0 et $\pi/4$.

La contrainte sur les angles s'énonce maintenant:

$$|DA - DAF| > \pi / 256 \qquad |DAF - DF| > \pi / 256$$

$$|DF - DFB| > \pi / 256 \qquad |DFB - DB| > \pi / 256$$

2.5-Choix des points intermédiaires

L'étude de la précision sur les calculs des angles nous a amenés à considérer des calculs approximatifs sur ceux-ci.

Calcul par TG 1/2:

Le calcul le plus simple est de prendre la tangente en I comme la moyenne des tangentes en A et F soit DI =(DA+DF)/2. Nous remplaçons 2 divisions, 2 recherches en table et 4 additions par 2 additions ce qui permet de gagner 8 µs en temps de calcul, soit environ 11%. Un autre avantage est de rendre inutile la table des arctangentes, en contrepartie, nous ne sommes plus certains de trouver le point I à l'intérieur du triangle ACF. Il est possible de corriger cette erreur ou de l'éviter au niveau de l'éditeur de caractères. (Fig 6)

Correction:

Les courbes tracées sont, par définition, croissantes en X, la condition: "le point I intérieur au triangle" peut être remplacée par: "XI compris entre XA et XF". La correction consiste alors à placer si nécessaire le point I en A ou F et à remplacer les deux cercles par une droite.

Prévention:

Les conditions nécessaires pour que le point I appartienne au triangle ACF sont identiques à celles définies au paragraphe précédent sur la précision.

Dans le cas DA > DAF > DF, nous devons avoir:

$$DA > DAI > DAF > DIF > DF$$

Comme DI = (DA + DF)/2, nous obtenons:

DA > (3 DA + DF)/4 > DAF > (DA + 3 DF)/4 > DF

En considérant DF =(DA + DB)/2 et tous les cas de figures possibles nous sommes arrivés aux inégalités suivantes:

DA > DB (5 DA + 3 DB)/8 < DAF < (7 DA + DB)/8

(7 DB + DA)/8 < DFB < (3 DA + 5 DB)/8

DA < DB (5 DA + 3 DB)/8 > DAF > (7 DA + DB)/8

(7 DB + DA)/8 > DFB > (3 DA + 5 DB)/8

Les critiques formulées sont dans le cas d'une correction, une déformation de la courbe qui peut être importante et dans le cas d'une prévention, un accroissement du nombre de courbes évoluées, pour respecter les contraintes sur les angles, et par conséquent une augmentation de la taille des fichiers d'entrée.

Calcul par approximation des TG:

Un calcul simple qui permet de supprimer la table des arctangentes consiste à approximer la fonction arctangente par une droite entre 0 et 1. Soit:

$DI = \pi/4\ (YF-YA)/(XF-XA)$ si $|YF-YA| < |XF-XA|$

$DI = \pi/2 - \pi/4\ (XF-XA)/(YF-YA)$ si $|YF-YA| > |XF-XA|$

La multiplication par $\pi/4$ peut être supprimée, car les angles calculés ne sont utilisés que pour effectuer des moyennes ou des recherches en table. De cette façon la division donne directement le résultat et il suffit pour rester cohérent d'appliquer le même facteur d'homothétie lors de l'édition des caractères et lors du calcul des tables trigonométriques.

Les contraintes sur les angles que doit respecter l'éditeur de caractères sont toujours issues des mêmes inégalités. Nous obtenons:

$|DAF - DI| < |DA - DAF|$ et $|DAF - DI| < |DF - DAF|$

L'erreur commise est:

$ER = |DAF - DI|$

La vérification de ces contraintes variables impose de calculer la valeur de l'angle en cours d'édition et de prévenir l'opérateur afin qu'il modifie le découpage si les contraintes ne sont pas respectées. Le tableau de la figure 7 donne l'erreur en fonction de l'angle.

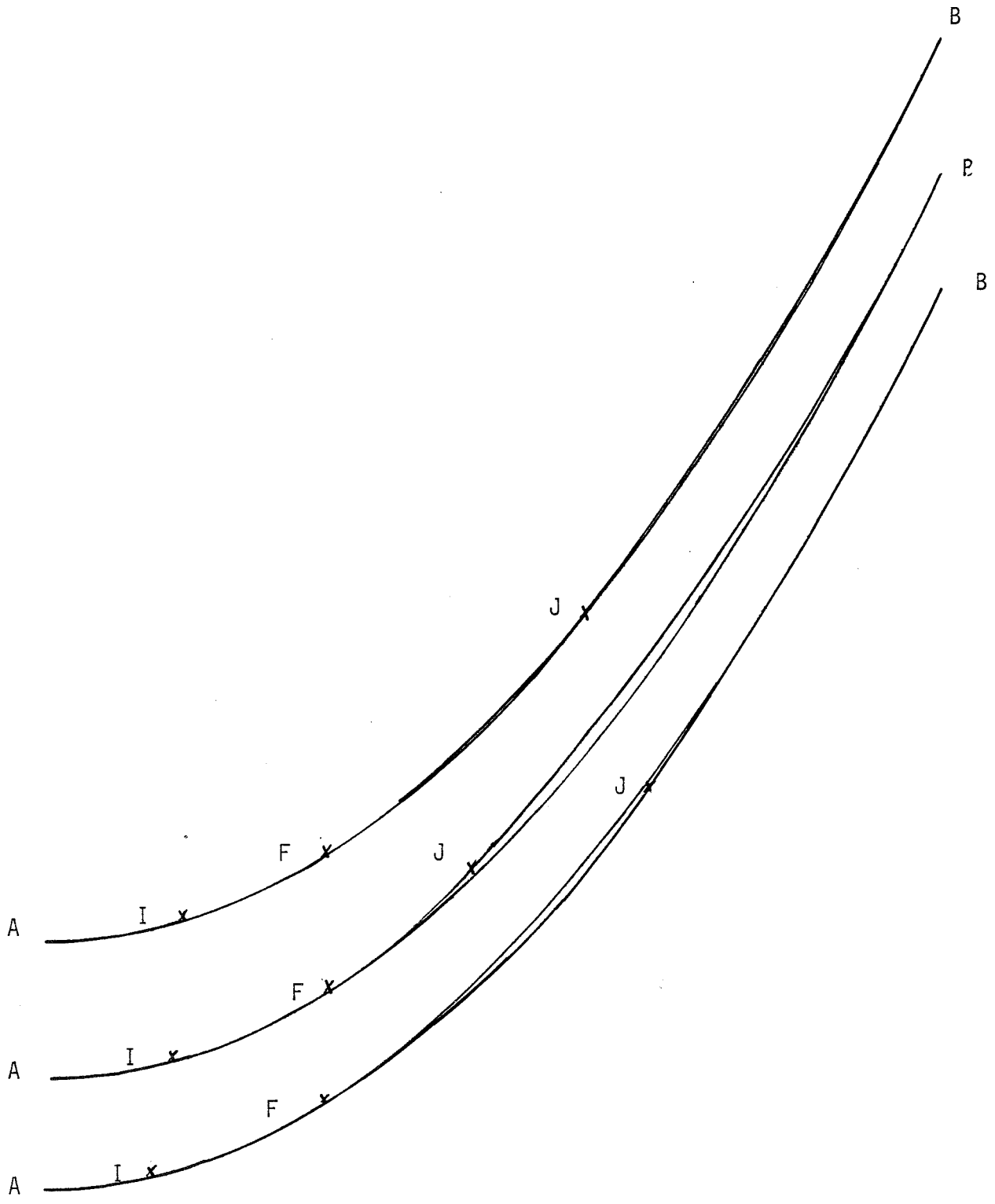
Les tracés d'une simulation de remplacement de courbes analytiques permettent de comparer les résultats obtenus par les trois méthodes de calculs. (FIG 8-9-10-11-12)

2.6-Problème lié au suivi des contours

L'algorithme de suivi des cercles, dérivé de l'algorithme de BRESENHAM modifié (HOR 77, COG 80), remplace les cercles par des droites dont les cosinus directeurs sont donnés par les dérivées partielles de la fonction implicite, et il calcule avec une erreur inférieure au point de grille en essayant d'annuler à chaque abscisse la fonction implicite. Les dérivées partielles secondes sont égales à 2, ce qui permet d'obtenir les nouveaux cosinus directeurs par de simples incréments de 2.

Cet algorithme très rapide impose, pour éviter des tests à chaque pas, qu'un arc de cercle appartienne à un seul quadrant et que les centres soient connus avec une grande précision. Le fait qu'un arc de cercle appartienne à un seul quadrant est résolu au niveau de l'éditeur de caractères; la fonction de suivi des contours impose que les courbes soient croissantes en X, ce qui empêche le cas des tangentes verticales en milieu de courbes évoluées; les tangentes horizontales peuvent également être interdites par définition.

Le problème de précision est critique lorsque le point d'arrivée d'un arc de cercle a une tangente verticale; dans ces conditions, si la distance centre-point d'arrivée est inférieure d'un point de grille à la distance centre-point de départ, la dérivée partielle de la fonction implicite par rapport à Y change de signe lors du calcul du dernier point et l'algorithme diverge. Une telle précision inférieure au 1/2 point de grille ne peut pas être obtenue.



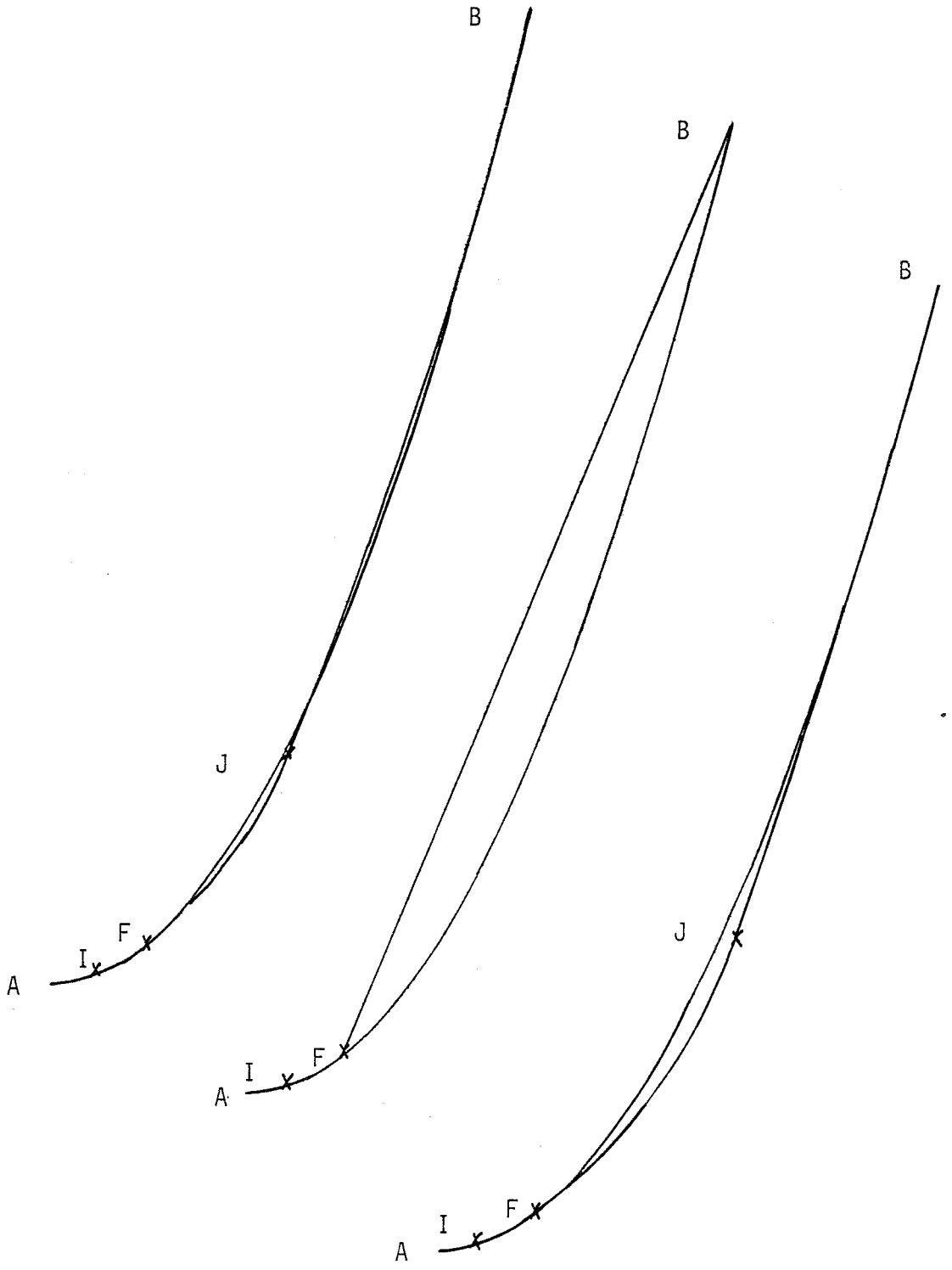
Tracé de la fonction X^2 entre 0 et 1 et de l'approximation par les cercles, F est le point de TG 1/2.

Calcul précis, courbes du haut.

Calcul par tangence 1/2, courbes du milieu.

Calcul par approximation des tangentes, courbes du bas.

Fig 8



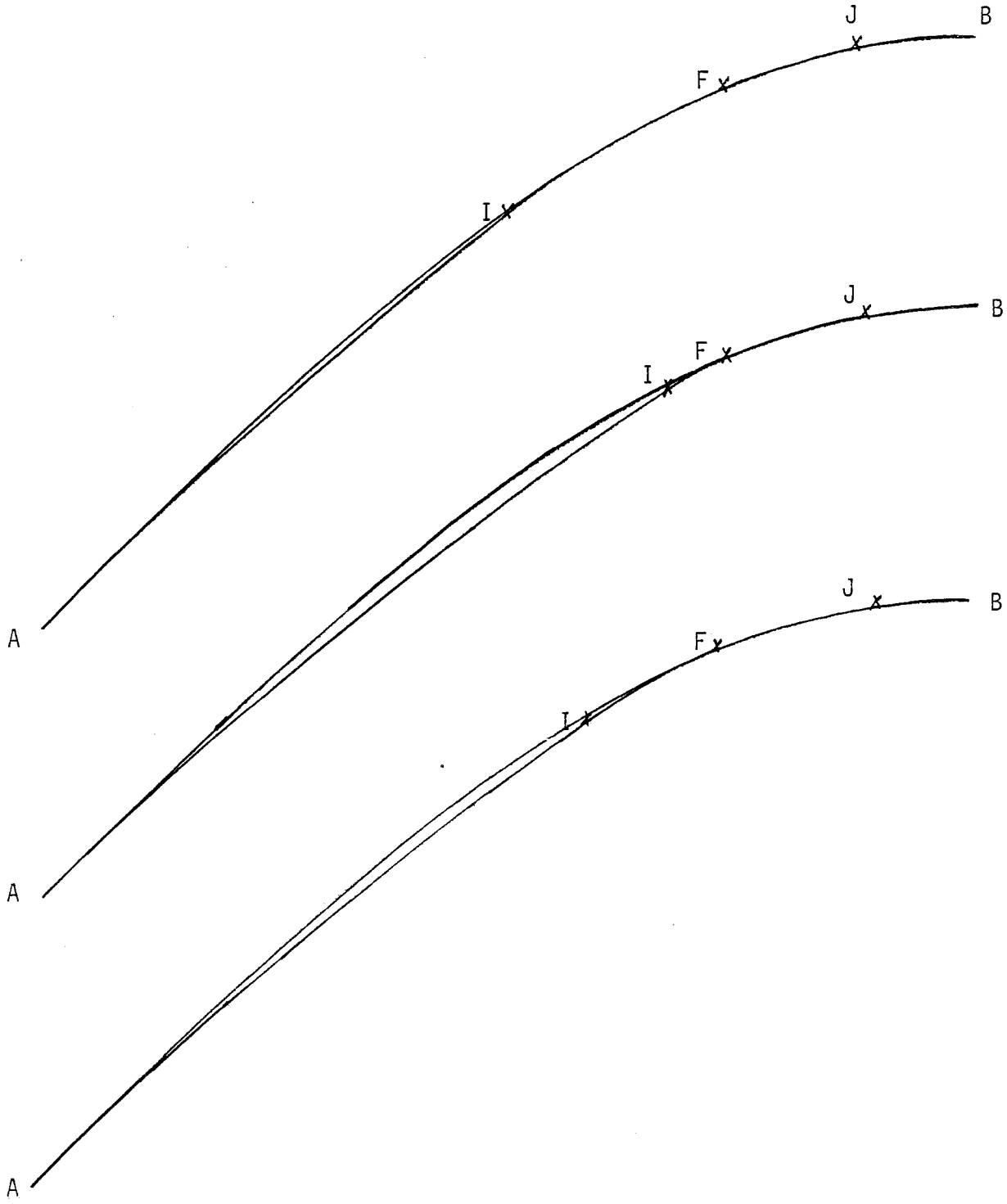
Tracé de la fonction X^2 entre 0 et 2 et de l'approximation par les cercles, F est le point de TG 1/2.

Calcul précis, courbes du haut.

Calcul par tangence 1/2, correction débordement, courbes du milieu.

Calcul par approximation des tangentes, courbes du bas.

Fig 9



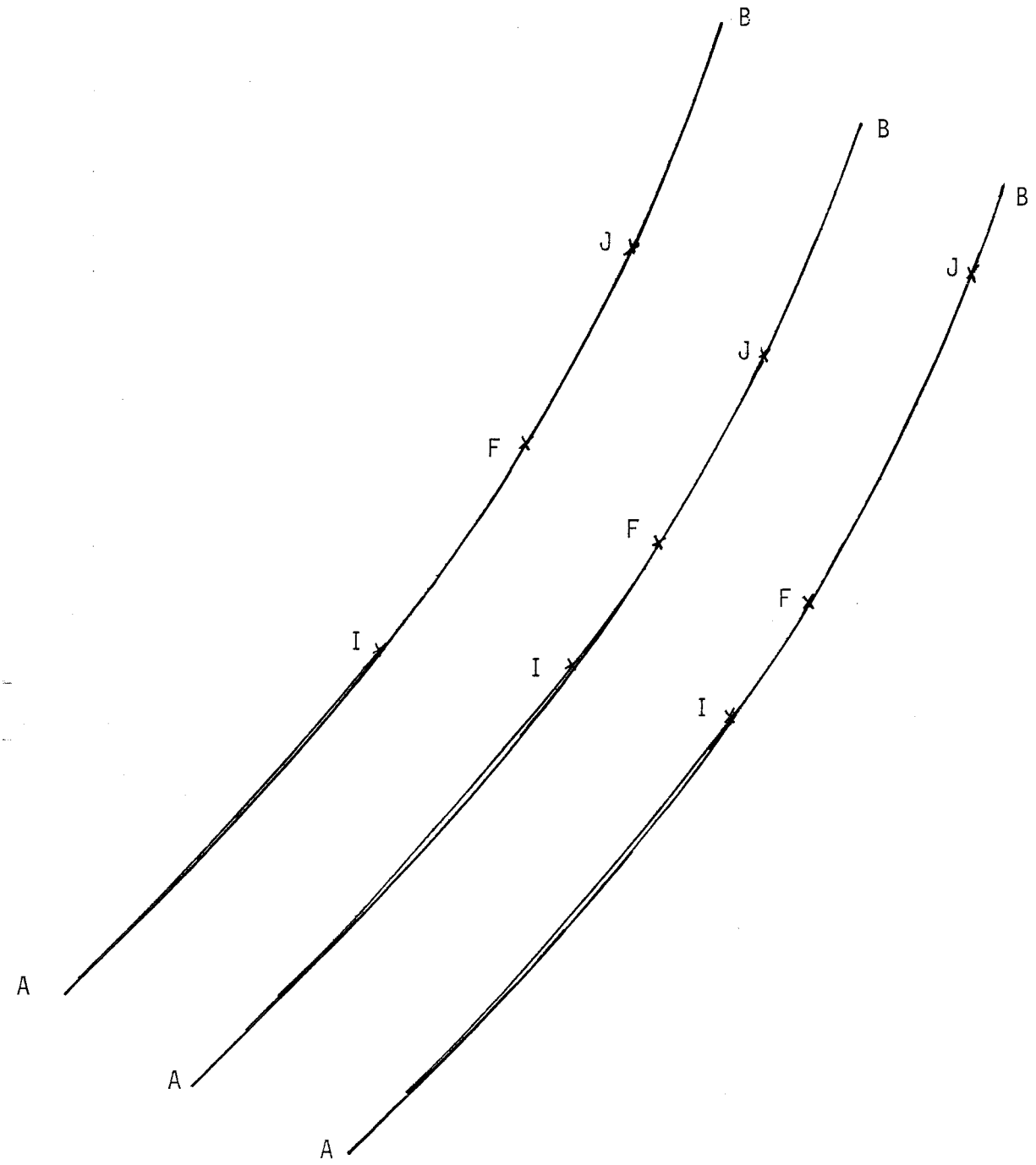
Tracé de la fonction sin x entre 0 et $\pi/2$ et de l'approximation par les cercles, F est le point de TG 1/2.

Calcul précis, courbes du haut.

Calcul par tangence 1/2, courbes du milieu.

Calcul par approximation des tangentes, courbes du bas.

Fig 10



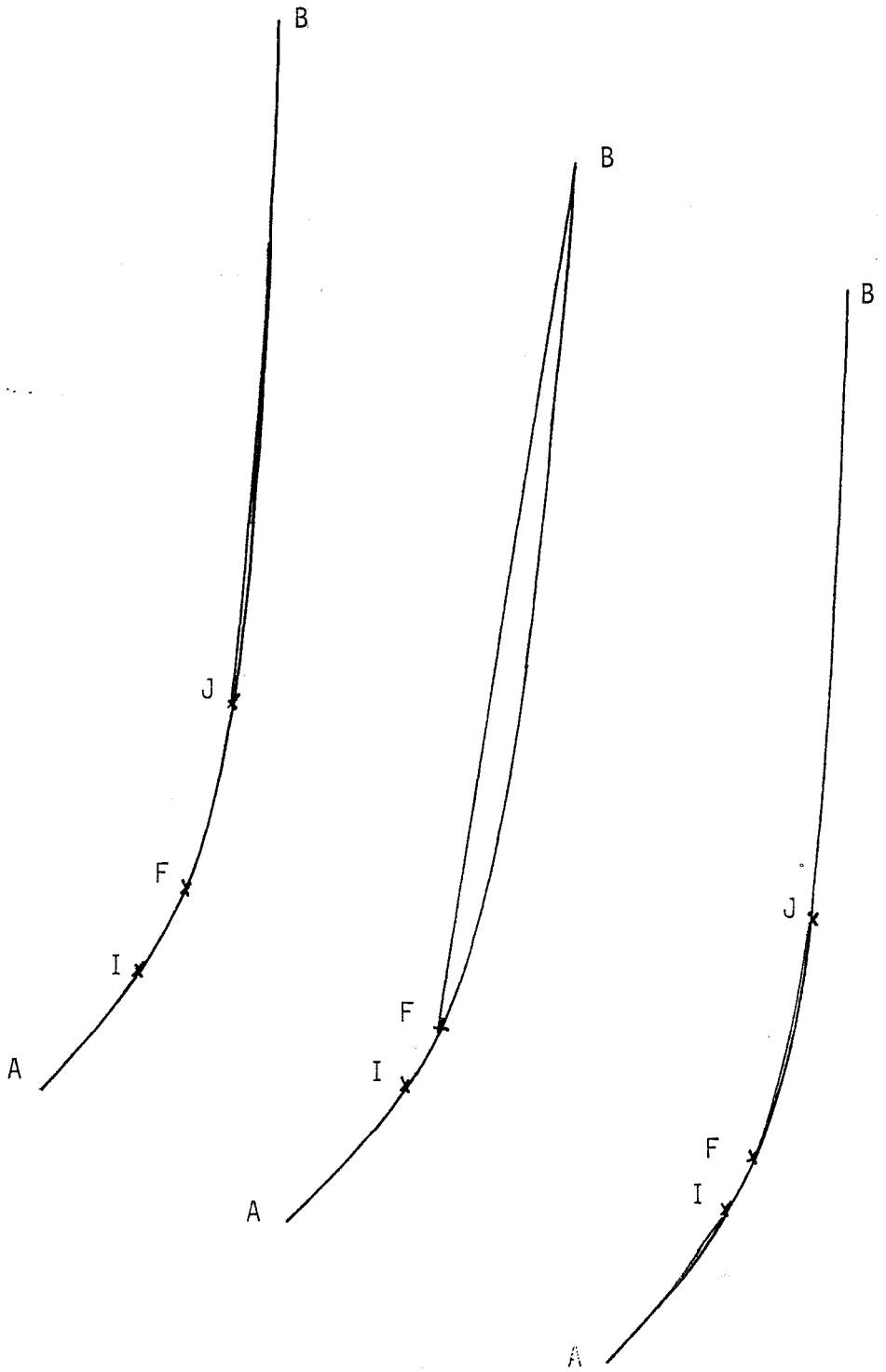
Tracé de la fonction tg x entre 0 et $\frac{\pi}{3}$ x.3 et de l'approximation par les cercles, F est le point de TG 1/2.

Calcul précis, courbes du haut.

Calcul par tangence 1/2, courbes du milieu.

Calcul par approximation des tangentes, courbes du bas.

Fig 11



Tracé de la fonction tg X entre 0 et $\frac{\pi}{4}$ x.45 et de l'approximation par les cercles, F est le point de TG 1/2.

Calcul précis, courbes du haut.

Calcul par tangence 1/2, correction débordement, courbes du milieu.

Calcul par approximation des tangentes, courbes du bas.

Fig 12

Dans le cas général le point d'arrivée est également point de départ d'un autre arc et nous avons le choix entre marquer le premier ou le dernier point: marquer le premier et non pas le dernier nous permet d'arrêter les calculs à l'avant-dernière abscisse et autorise une précision inférieure à 1.5 point de grille.

2.7-Précision des calculs

La précision de 1.5 point de grille, nécessaire pour l'algorithme de tracé des cercles, doit être respectée pour des nombres variant de 1 à 2048. Nous ne pouvons donc pas utiliser pour les calculs de l'étape 2 un format en virgule fixe sur 16 bits et nous avons donc effectué les résolutions de systèmes linéaires en flottant.

Les systèmes linéaires sont résolus par une succession de 4 multiplications 16x16 avec résultat sur 32 bits. Pour ramener les résultats intermédiaires de 32 à 16 bits tout en gardant une bonne précision nous utilisons un décaleur variable qui nous donne les 16 bits les plus significatifs et la valeur du décalage réalisé. Le résultat final doit être donné en entier, ce qui nécessite un cadrage lors du dernier calcul.

Les centres de cercles sont transmis sur 16 bits à l'étape 3 et en relatifs par rapport au premier point. Le tracé d'un cercle ayant un rayon supérieur à 2^{15} est remplacé par celui d'une droite, sans dégrader l'aspect du caractère; de plus cette substitution permet de réduire le temps de calcul de cette dernière étape.

Le format des coordonnées des points donnés ou calculés est fixé par la grille de définition 2048x2048 à des entiers sur 11 bits. Le multiplieur nous impose d'autre part une limite de 15 bits significatifs pour les résultats intermédiaires. Enfin les sinus et cosinus compris entre 0 et 1 seront codés en 15 2^{-14}.

Pour respecter la précision autour de $\pi/2$ et l'amplitude autour de 0 nous utilisons pour 1/sin X un format .15 2^a avec $1 \leqslant a < 10$.
Autour de 0: $1/\sin X \# 1/X \leqslant 512/\pi < 160$

Autour de $\pi/2$: si nous posons $d = \pi/2 - X$, $\sin X = 1 - (d^2/2)$

Le plus petit écart possible pour $1/\sin X = 2^{-15}$

Nous avons $d^2 = 2^{-14}$ et $d = 2^{-7}$

Nous considérons donc la résolution du format .15 2^a suffisante, l'erreur relative due à l'arrondi étant inférieure à 2^{-16}.

Nous allons maintenant nous intéresser à la propagation des erreurs lors de la résolution d'un système linéaire.

Calcul I et J

Le premier produit est de la forme:

$$P1 = X \sin D - Y \cos D$$

X et Y sont des points justes par définition et ils sont inférieurs à 2^i. "Normalement i=11"

L'erreur sur les sinus et cosinus est une erreur d'arrondi:

$ER = 2^{-16}$.

$P1 = X (\sin D + ER) - Y (\cos D + ER)$

$P1 = X \sin D - Y \cos D + ER (X-Y)$

Au maximum $ERP1 = 2^{-16+i}$

Le second produit est de la forme:

$$P2 = P1 \sin^{-1} S$$

$P2 = (P1 + ERP1) (\sin^{-1} S + ERS)$

$P2 = P1 \sin^{-1} S + ERP1 \sin^{-1} S + P1\ ERS + ERS\ ERP1$

Au maximum $P1 = 2^{i+1}$, $ERP1 = 2^{-16+i}$, $ERS = 2^{-16}$ et $\sin^{-1} S = 2^7$.

$ERP2 \# ERP1 \sin^{-1} S \lesssim 2^{-9+i}$

Le troisième produit est de la forme:

$$P3 = P2 \cos B$$

$P3 = (\cos B + ER) (P2 + ERP2)$

$P3 = P3 \cos B + ERP2 \cos B + P2\ ER + ERP2\ ER$

$ERP3 \# ERP2 \cos B < 2^{-9+i}$

Une précision au demi-point de grille impose $ERP3 < 2^{-1}$, soit i = 8, les écarts entre les coordonnées de A, F et B doivent être inférieurs à 256.

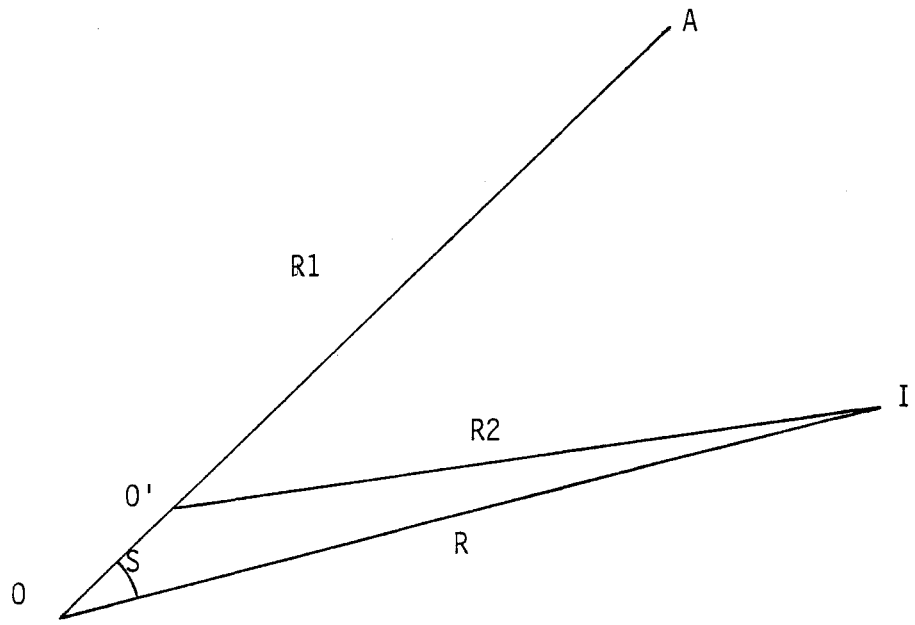


Fig 13

ETAPE D' EXPANSION DES COURBES EVOLUEES

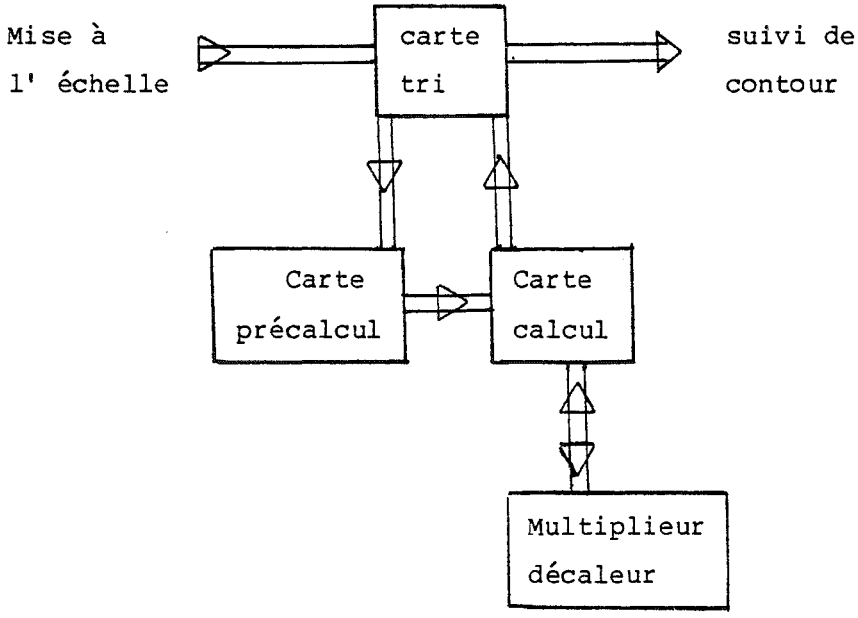


Fig 14

Pour une grille de 2048x2048, cela correspond à diviser la hauteur du "O" en 4 courbes évoluées au lieu de 2.

Calcul des centres

Premier produit:

Les points I et J sont connus au demi-point près.

$P1 = (X + ERX)\ (\sin D + ER) - (Y + ERY)\ (\cos D + ER)$

$ERP1max = ERX \sin D + ERY \cos D$

$= \sqrt{2}\ /2$

Second produit:

$P2 = (P1 + ERP1)\ (\sin^{-1} S + ERS)$

$ERP2 = ERP1 \sin^{-1} S$

$ERP1max = 1\ ,\ (\sin^{-1} S)max = 2^7,$

$ERP2max = 2^7$

Une interprétation géométrique des calculs montre que P2 est égal au rayon du cercle. Or nous savons que le tracé des cercles est plus sensible à une différence entre rayon de départ et rayon d'arrivée qu'à l'erreur absolue. Nous allons donc étudier cet écart en fonction de l'angle S. (Fig 13)

$R1 = R - ERP2$

$R2 = (R^2 + ERP2^2 - 2\ ERP2\ R \cos S)^{1/2}$

L'erreur ERP2 est maximale lorsque s tend vers 0 soit cos S = 1

$R2 = R - ERP2 = R1$

La différence des rayons tend vers 0 quelque soit l'erreur commise sur le calcul du rayon.

Lorsque S tend vers $\pi/2$ cos S=0, $\sin^{-1} S = 1$ et $ERP2 = 1/\sqrt{2}$

$R - ERP2 < R2 < R + ERP2$

$R2 - R1 < 2\ ERP2$

$< \sqrt{2}$

L'erreur commise est toujours inférieure à 1.5 point de grille

Troisième produit:

P3 = (P2 + ERP2) (cos D + ER)

ERP3 # ERP2 cos D $\leqslant$ ERP2

Ainsi, malgré une erreur qui peut atteindre 88 points de grille sur le calcul du rayon, nous obtenons une différence de rayon inférieure à 1.5 point de grille.

3.-Mise en parallèle des calculs

Même avec les algorithmes les plus rapides nous obtenons un temps de calcul de 62 µs, soit 120 µs si nous comptons les tests et les opérations d'entrée-sortie. Pour respecter le débit de la machine nous devons adopter une structure parallèle à deux processeurs.

Pour l'expansion des courbes évoluées nous pouvons considérer deux étapes totalement séquentielles; la première au cours de laquelle nous déterminons les coefficients des systèmes linéaires par des divisions et des recherches en tables, et la seconde au cours de laquelle nous résolvons ceux-ci par des multiplications en réel. Une structure pipe-line est intéressante car elle nous évite de dupliquer les circuits qui ne sont utiles que pour une étape: cablage de la division, tables trigonométriques et multiplieur en réel.

III.-REALISATION

Comme nous venons de le voir l'expansion des courbes évoluées est une opération coûteuse en temps de calcul. De plus cette étape doit s'insérer dans le pipe-line principal et assurer le transfert des informations. La spécialisation de ces deux travaux, tri et calcul, nous a conduits à prévoir une carte de tri qui s'intercale dans le pipe-line et qui alimente des circuits de calculs.

Les circuits de calcul sont eux mêmes répartis sur trois cartes qui forment un sous pipe-line. Une première carte de précalculs détermine les coefficients des systèmes linéaires, une seconde les résout en sous-traitant les multiplications à une carte purement matérielle qui réalise ces multiplications en réel. Nous arrivons ainsi à la structure de l'étape d'expansion des courbes évoluées présentée au niveau de la carte sur la figure 14. L'utilisation de microprocesseurs en tranches nous a permis de réaliser des chemins de données adaptés à chaque utilisation. Les liaisons à l'intérieur du sous pipe-line sont assurées par des mémoires premier entré, premier sorti (FIFO); ce qui rend chaque étape totalement asynchrone.

Carte de tri

Cette carte est chargée de lire le fichier de description des contours, d'extraire les paramètres des courbes évoluées, de recopier la structure d'arbre, les droites et enfin les résultats fournis par les cartes de calcul. Afin d'utiliser au mieux la structure pipe-line, l'exploration du fichier d'entrée doit se poursuivre en parallèle avec les calculs. Cette contrainte implique de mémoriser les adresses où seront rangés les résultats. Celles-ci sont placées sur les registres internes de l'unité arithmétique et logique. Un compteur extérieur contient le nombre de courbes évoluées en cours de calcul, ce qui facilite les tests et permet de pointer le registre où ranger l'adresse d'écriture du résultat. Les trois bits de poids forts des données nous indiquent la nature de la donnée et par conséquent le traitement qu'elle doit subir. A une suite de tests et sauts conditionnels nous avons préféré un branchement indexé réalisé en un seul cycle; pour cela les trois bits de poids faibles de l'adresse de la micro-instruction suivante sont remplacés par les trois bits de poids forts des données. (Fig 15)

Carte précalcul

Cette carte est chargée de calculer les paramètres des systèmes linéaires.

CARTE TRI

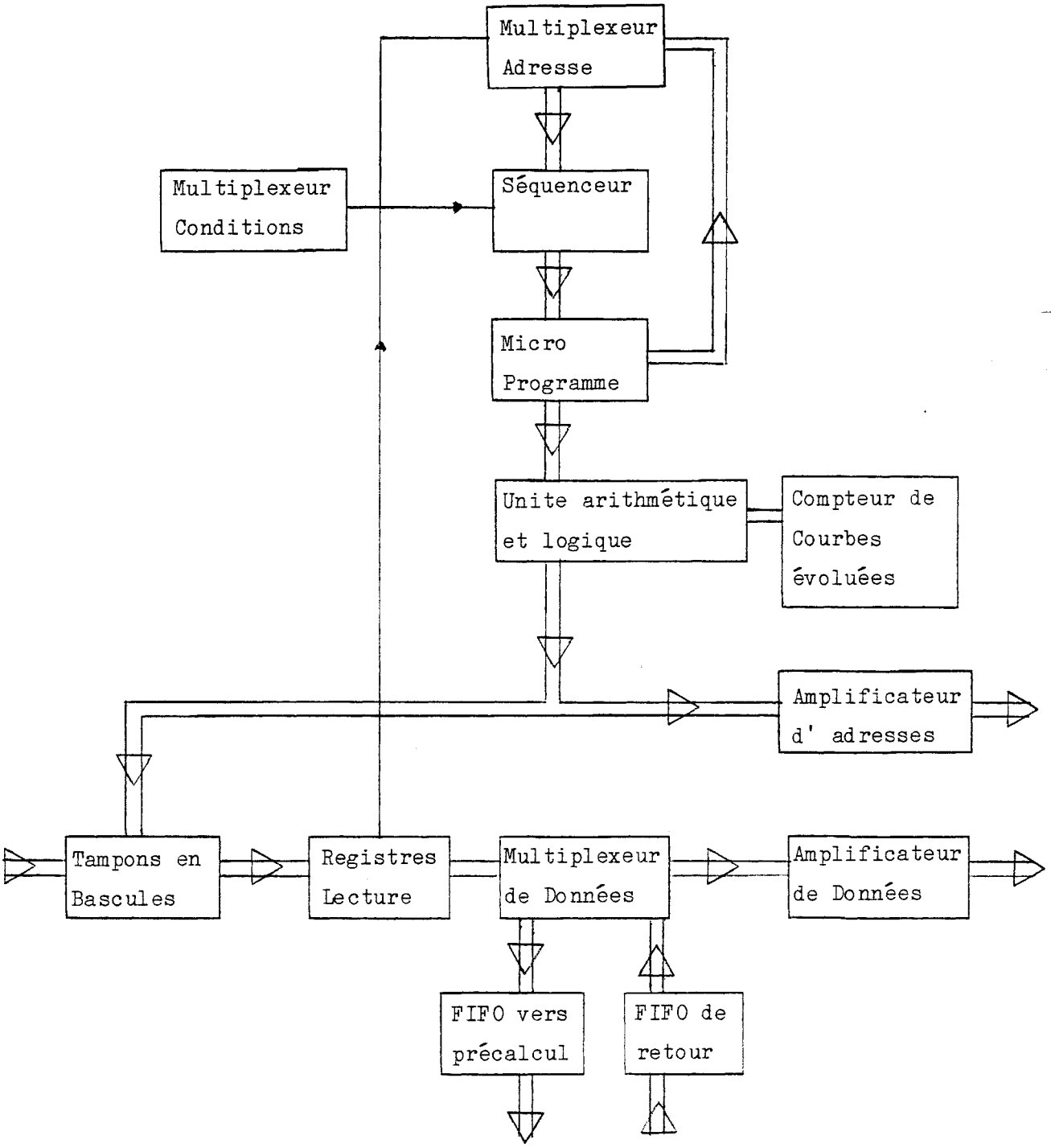


Fig 15

CARTE PRECALCUL

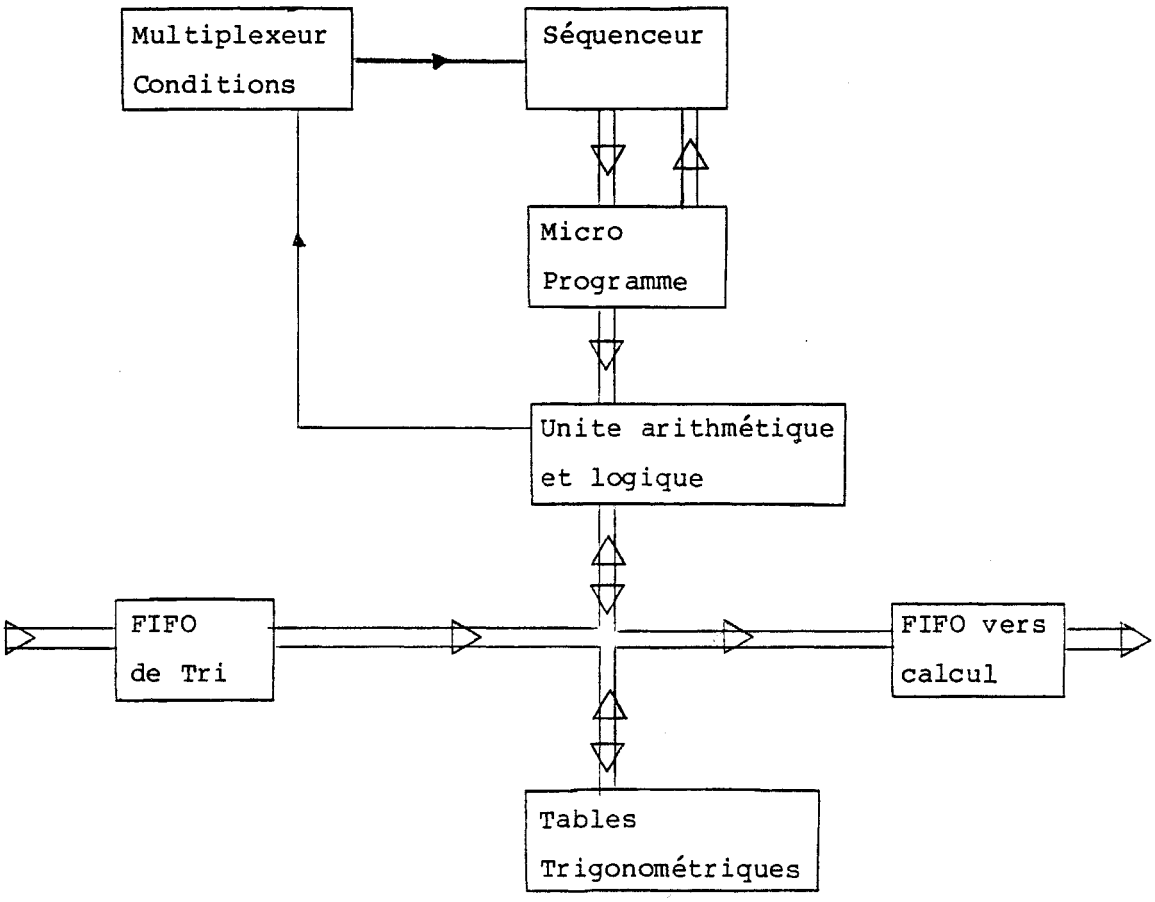


Fig 16

L'unité arithmétique et logique est cablée pour réaliser un algorithme de division sans restauration. Les opérandes doivent être positifs et le nombre de cycles est égal au nombre de bits dans le résultat. Le calcul des valeurs trigonométriques est réalisé par des accès à des tables. (Fig 16)

Carte calcul

Elle résout les systèmes linéaires par une suite de quatre multiplications sous-traitées à la carte multiplieur décaleur; son rôle est donc limité aux contrôles de l'exécution et des résultats. (Fig 17)

Les coordonnées des centres de cercles sont transmises sous forme d'entier de seize bits. En cas de débordement nous remplaçons le cercle par une droite.

Le point intermédiaire I doit se trouver à l'intérieur du triangle formé par la corde AF et les tangentes en A et F. Dans le cas contraire le point I est un point de rebroussement particulièrement inesthétique. Nous avons vu dans le paragraphe sur la précision des calculs des angles que cette condition d'appartenance du point I au triangle était équivalente à ce que l'abscisse du point I soit comprise entre celle de A et celle de F. En cas d'erreur la correction consiste à forcer le point I en A ou en F et à remplacer les cercles par des droites.

D'autre part les coordonnées des centres de cercles doivent être données en relatif par rapport au premier point lors du tracé. L'étape de suivi des contours effectue des accès à la mémoire dans l'ordre croissant ou décroissant suivant que la paroi est directe ou inverse. La carte de tri lit les valeurs uniquement dans l'ordre croissant. Lors du suivi des parois, celles-ci sont toujours en abscisses croissantes; nous devons donc corriger les centres de cercles dans le cas où nous trouvons des points en abscisses décroissantes.

CARTE CALCUL

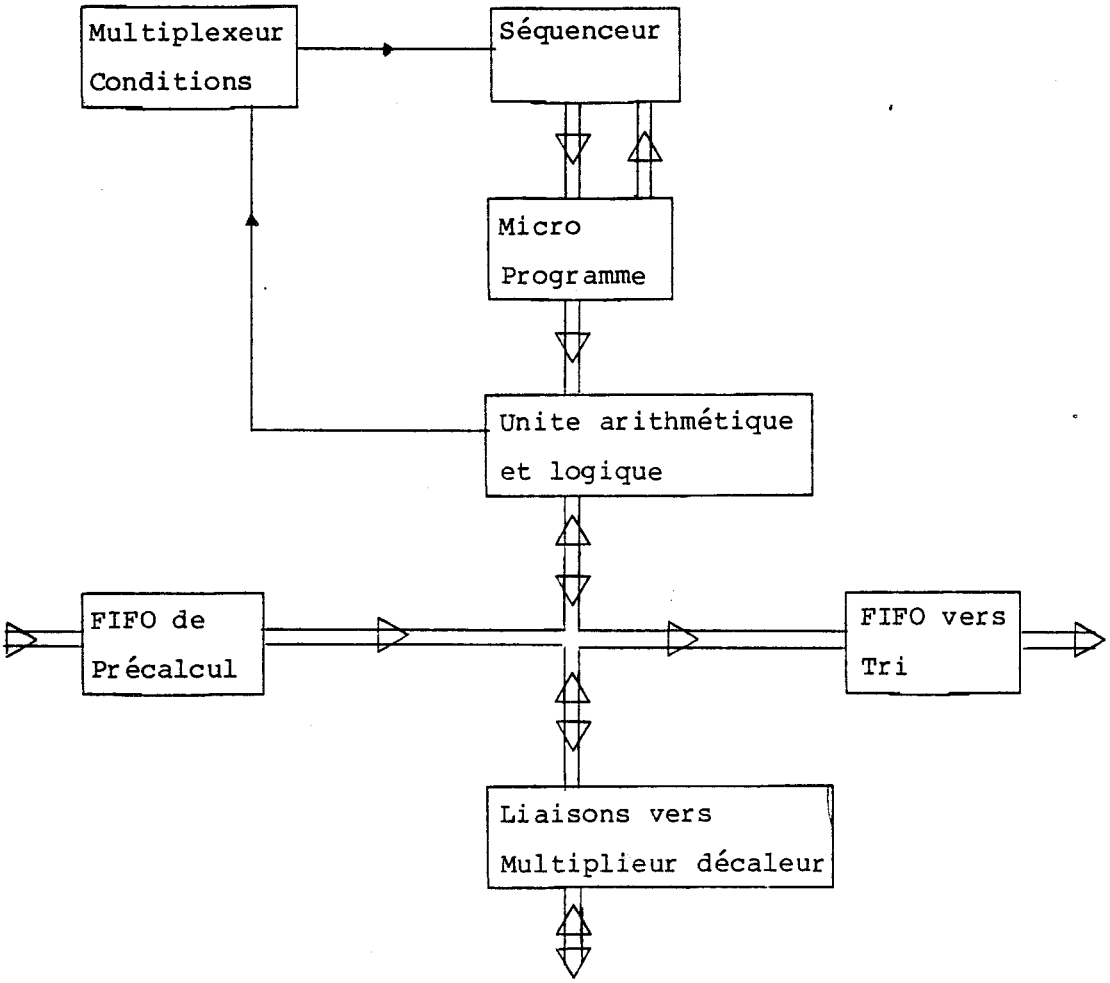


Fig 17

CARTE MULTIPLIEUR DECALEUR

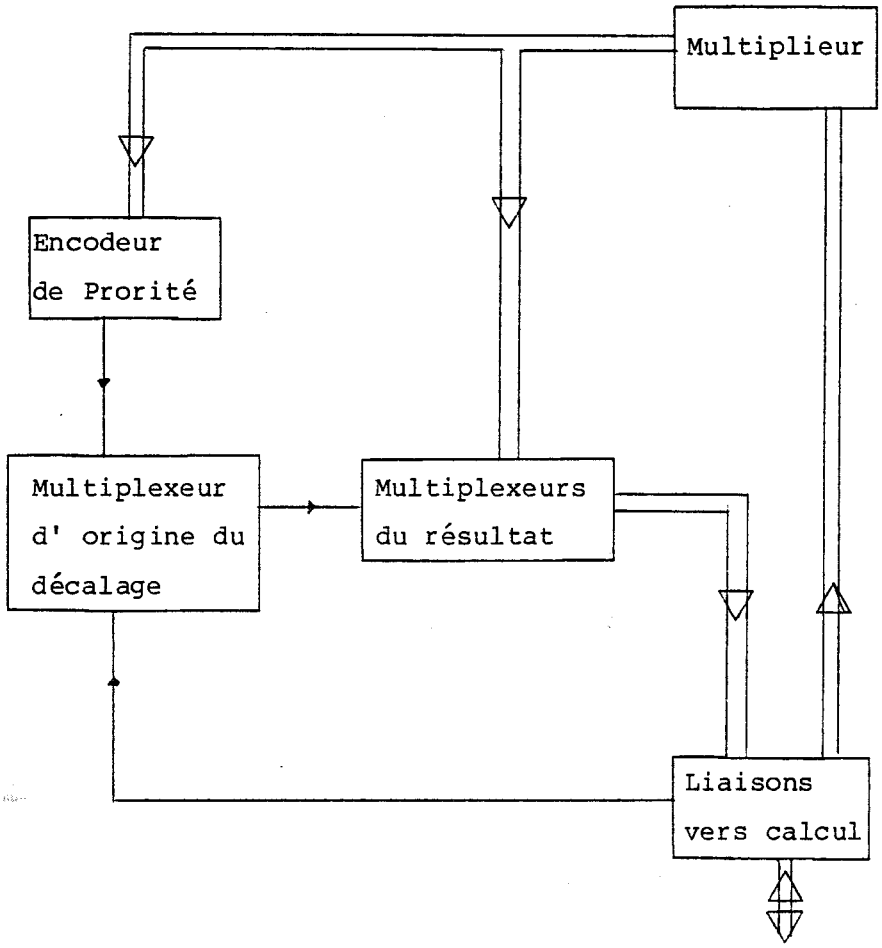


Fig 18

Carte multiplieur décaleur

La résolution des systèmes linéaires demande une suite de quatre multiplications, et ces opérations doivent être rapides et précises. La rapidité est donnée par l'utilisation d'un multiplieur ayant un temps de traversée de 120 ns. La précision est obtenue en sélectionnant après chaque produit les seize bits les plus significatifs parmi les trente deux du résultat. Cette opération de normalisation est réalisée en transmettant les seize bits de poids forts à deux encodeurs de priorité qui indiquent où se trouve le premier bit significatif. Cette position est transmise à des multiplexeurs qui décalent le résultat et à la carte de calcul qui utilise cette valeur pour cadrer le dernier résultat en entier et pour tester le débordement des rayons. (Fig 18)

IV.-ANALYSE DES PERFORMANCES

Les deux critères importants au niveau de cette étape sont la précision des calculs et le temps d'expansion d'une courbe évoluée.

Pour contrôler la précision des calculs nous avons codé le contour d'un cercle et vérifié que tous les centres calculés avaient les mêmes coordonnées. Nous avons également codé d'autres caractères pour mesurer les débits des diverses étapes et nous avons toujours obtenu des formes correctes.

Pour déterminer les coefficients de la formule qui donne le temps de calcul de cette étape nous utilisons des caractères tests qui permettent d'isoler l'influence d'un paramètre. Ces paramètres sont:

- a: Nombre de mots de seize bits dans la structure d'arbre
- b: nombre de droites avant la première courbe évoluée (b') + nombre de droites au dessus de quinze après une courbe évoluée
- c: nombre de courbes évoluées

La complexité du paramètre b vient du parallélisme entre, d'une part, les calculs et, de l'autre, l'analyse du fichier qui rend transparent en temps le transfert des quinze droites qui suivent. Il est à remarquer qu'en général:

$$b = b'.$$

D'après le tableau de mesure nous avons la formule:

$$T(\mu s) = 1,2\ a + 3,8\ b + 42\ (c \neq 0) + 58\ c$$

Le terme $(c \neq 0)$ correspond à l'amorçage du pipe-line, pour la première courbe évoluée, au cours duquel seule la carte précalcul travaille. Nous obtenons un temps de traversée de 42 µs pour la carte précalcul et 58 µs pour la carte calcul. Nous avons une carte précalcul plus rapide que la carte calcul en prenant les points intermédiaires et le point F comme points de carrure; dans ces conditions nous n'avons pas programmé les autres algorithmes qui n'accélèrent que la carte précalcul.

Les mesures ont été réalisées avec des temps de cycle de 210 ns pour la carte de tri, 260 ns pour la carte précalcul et 230 ns pour la carte calcul. Nous sommes capables de calculer quinze courbes évoluées en 1 ms; si nous considérons une moyenne de vingt courbes évoluées par caractère, le passage à 1000 car/s demande de réduire les temps de cycle d' 1/4. Cette réduction donne pour la partie calcul un temps de cycle de 170 ns qui est incompatible avec notre système de développement, limité à 200 ns, mais qui est tout à fait réalisable avec les produits actuellement sur le marché.

TABLEAU DE MESURE

a	b	c	Temps (µs)
12	12	0	58.1
10	12	0	55.8
12	16	0	70
6	1	2	157.4
6	1	4	273
6	1	6	389
6	1	8	506
6	1	20	1206
6	20	20	1210

Chapitre 4

CHAPITRE 4

DECOUPAGE AUTOMATIQUE DE CONTOUR

I.-PRESENTATION

II.-SAISIE DE L'IMAGE

III.-DECOUPAGE EN SOUS PAROIS

1.-Points particuliers
2.-Recherche du contour
3.-Suivi d'une sous paroi
4.-Problèmes rencontrés

IV.-CODAGE DES SOUS PAROIS

1.-Choix des interpolants et de l'erreur
2.-Algorithme d'approximation
3.-Calcul de l'interpolant
4.-Filtrage des droites

V.-STRUCTURE D'ARBRE

VI.-DECODAGE

VII.-PERFORMANCES

1.-Réalisation

2.-Codage par droites et courbes évoluées

3.-Codage par droites uniquement

4.-Temps de calcul

DECOUPAGE AUTOMATIQUE DE CONTOUR

I.-PRESENTATION

Les fabriquants de photocomposeuses possèdent de nombreuses polices sous forme d'images ou codées par plage. L'utilisation de codage par contour demande la saisie ou le transcodage de ces polices. De façon générale pour réaliser ce travail nous pouvons utiliser une méthode interactive ou une méthode automatique.

La première a été réalisée par M. Hourdequin (HOU 78) et correspond à une édition de caractères. L'opérateur définit à partir d'un dessin les paramètres des interpolants; ceux ci sont saisis par un programme interactif qui réalise un contrôle de cohérence et trace le contour. L'opérateur vérifie le résultat obtenu et peut s'il le désire modifier la forme en changeant les paramètres.

La deuxième méthode correspond à une découpe automatique avec arrêt sur respect d'une contrainte. Le caractère est dans ce cas saisi par une caméra ou décodé et il est ensuite approximé par une suite d'interpolants.

L'intérêt de la première méthode est qu'elle utilise l'opérateur pour lisser les contours du dessin et pour séparer les droites et les courbes évoluées. Elle n'exige donc que des originaux de qualité moyenne. En contrepartie le temps de saisie, la qualité du résultat et la taille du code dépendent pour une grande part des compétences et motivations de l'opérateur.

La deuxième méthode est bien entendu indépendante de l'opérateur et elle est plus rapide, ce qui est un avantage important puisqu'il existe des milliers de polices. En contrepartie la qualité du résultat est fortement liée à la qualité des contours de l'image originale à coder. Le codage automatique réalisé peut être découpé en trois étapes:

-La saisie de la forme à coder

-La recherche des contours

-Le codage des contours.

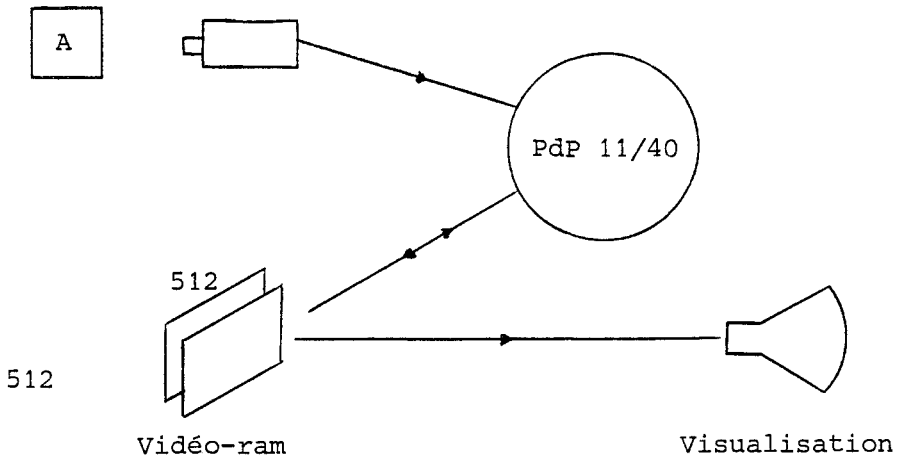


Fig 1

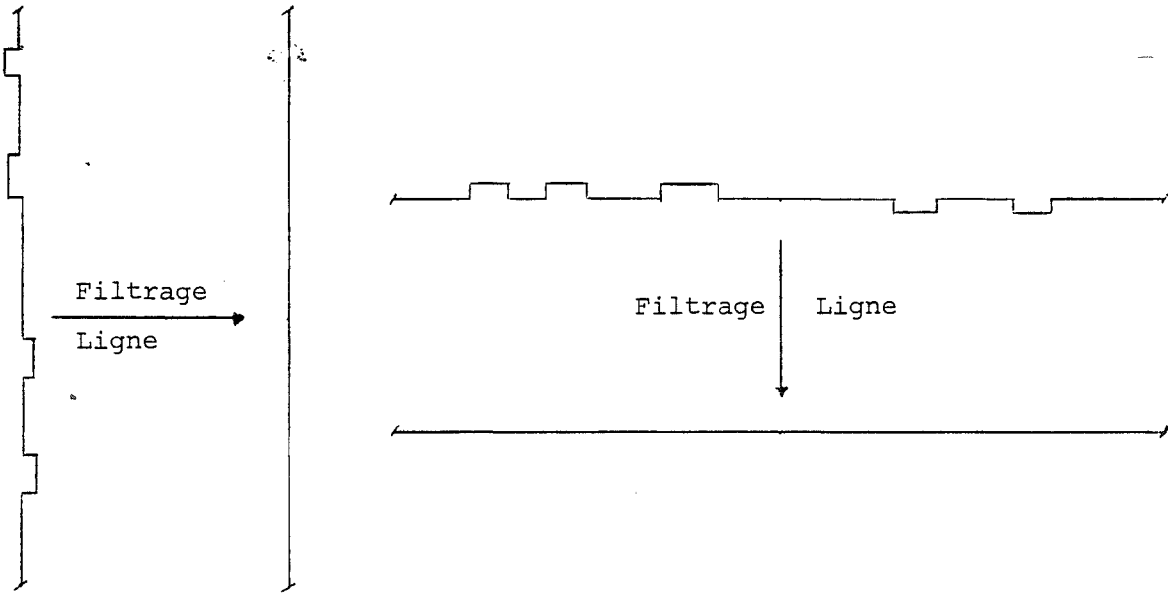


Fig 2

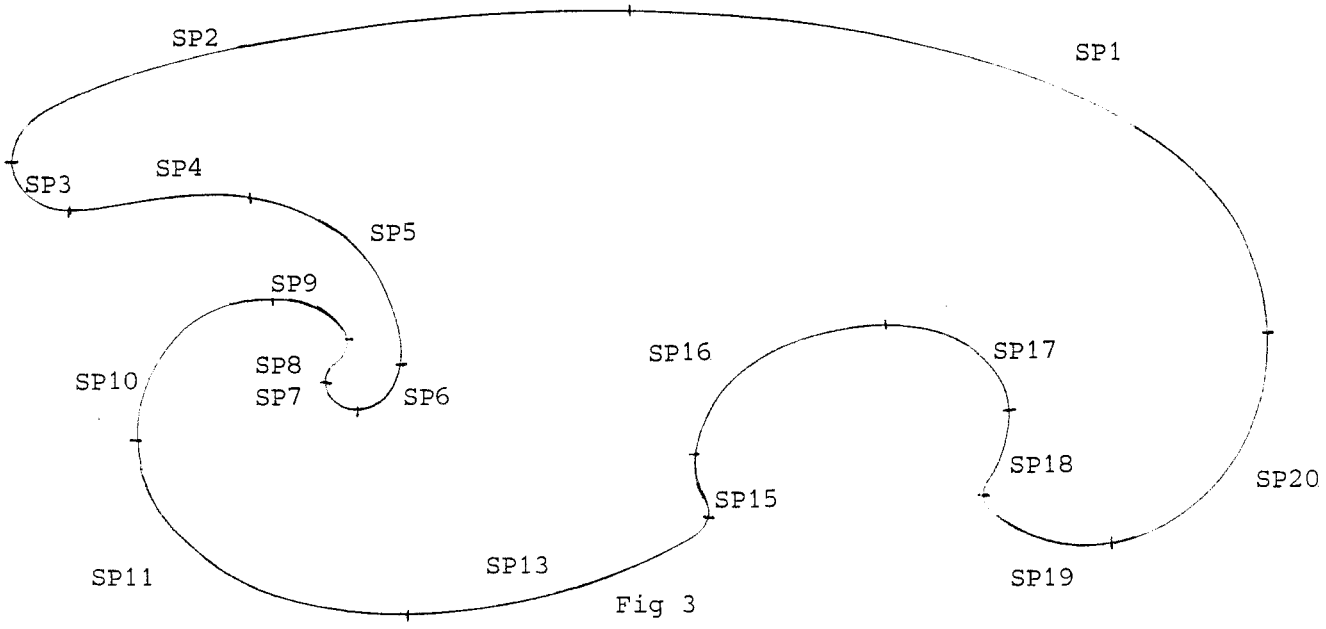


Fig 3

Dans les deux méthodes le calcul de la structure d'arbre qui est une opération longue et complexe doit être pris en charge par le programme.

II.-SAISIE DE L'IMAGE

Le système de saisie utilise une caméra vidéo et transfère une image bicolore de 512X512 points dans une mémoire vidéo-ram qui nous sert de mémoire de données et qui possède des circuits de génération d'un signal vidéo transmis en permanence vers un moniteur de visualisation. (Fig 1) Les deux problèmes rencontrés au niveau de cette étape sont la qualité et le positionnement des documents.

Afin d'utiliser des reproductions de caractères de qualité courante, nous avons prévu un programme de correction interactif de l'image saisie; l'utilisation de ce programme nous ramène aux problèmes évoqués pour le programme d'édition des caractères: le résultat obtenu est fonction de l'opérateur, et le temps de codage se trouve augmenté dans une forte proportion. Les dessins originaux d'artiste utilisés par les industriels pour coder les caractères sont d'une qualité suffisante mais sont malheureusement d'un accès difficile.

Le positionnement angulaire du document est important puisqu'il permet d'obtenir les portions de contour horizontales ou verticales. Pour cela les caractères de grand format simplifient le travail en rendant le positionnement moins critique. Dans le cas d'une utilisation intensive, le cadrage du caractère par rapport aux bords du document permettrait de n'effectuer ce réglage qu'une fois. La caméra de saisie nous fournit une matrice de points; il est donc impossible d'obtenir des bords horizontaux ou verticaux parfaits (Fig 2). Dans le cas où ces erreurs sont inférieures à un point de grille nous avons un programme de lissage qui permet de corriger ce bruit.

Il est possible de remplacer cette étape de saisie par une étape de décodage pour des caractères préalablement codés par plage par exemple.

Les deux problèmes évoqués dans ce paragraphe disparaissent. Bien entendu il faut disposer de caractères définis dans une grille au moins égale à celle que nous utilisons. Dans notre configuration il faudrait des caractères codés à partir de matrice 512X512.

II.-DECOUPAGE DU CONTOUR EN SOUS-PAROIS

Pour simplifier l'opération de codage nous décomposons le contour en sous-parois qui sont des sous ensembles limités par des points particuliers. L'opération de suivi des sous-parois est simplifiée par une recherche préalable des contours. Nous allons ainsi examiner:

-Les points particuliers
-La recherche des contours
-Le suivi d'une sous-paroi
-Les problèmes rencontrés au cours de cette étape.

1.-Points particuliers

La génération du caractère doit se faire en une seule passe avec un balayage suivant les abscisses croissantes. Pour cela nous découpons le caractère en parois; chaque paroi est une portion de contour monotone en X et comprise entre un minimum local en X, ou point de naissance, et un maximum local en X, ou point de mort.

D'autre part les interpolants utilisés ne peuvent franchir les extréma locaux en X ou en Y, la droite de par sa définition géométrique, la courbe évoluée afin de respecter une contrainte de l'algorithme de suivi des cercles qui n'accepte que des arcs de cercles appartenant à un seul quadrant. Il faut donc décomposer chaque paroi en sous-parois. Une sous-paroi est une portion de contour monotone en X et en Y et comprise entre deux extréma locaux. (Fig 3)

Ces points particuliers où le contour présente une tangente verticale ou horizontale sont visuellement très importants et il est donc naturel d'avoir à les repérer très précisément.

2.-Recherche du contour

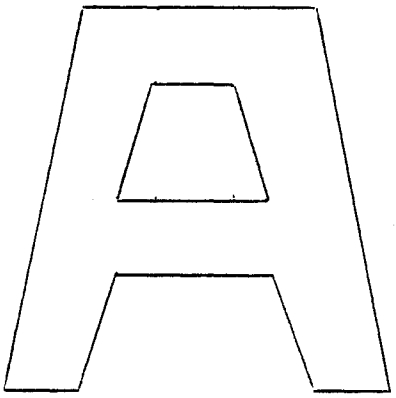
Lors du tracé du caractère, le décodeur ne calcule qu'une seule ordonnée par abscisse et par paroi; il est donc possible au niveau du codage de ne retenir qu'une seule ordonnée par abscisse et par paroi et ceci sans perte d'information si nous repérons l'ordonnée à calculer pour obtenir la forme d'origine.

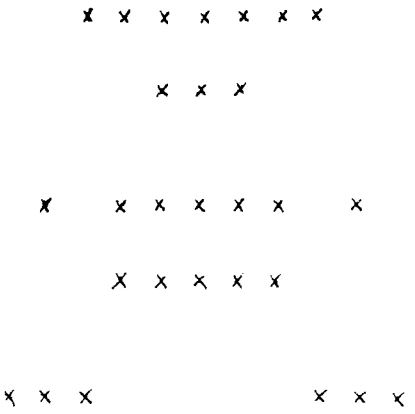
Ces deux dernières remarques nous ont permis d'utiliser une structure de données très simple. Pour ranger la description d'une sous paroi nous utilisons un tableau d'abscisses et un tableau d'ordonnées; la taille maximale de ces tableaux est égale au côté de la grille de définition soit 512 dans notre cas. Le tableau d'abscisses contient une suite croissante ou décroissante de nombres et pourrait être remplacé par deux variables; cependant afin de simplifier les algorithmes et vue la taille réduite de ce tableau nous avons préféré le conserver tel quel.

Nous désirons donc obtenir à un instant donné une ordonnée par abscisse pour une sous paroi. Le caractère étant représenté sur une grille, il peut exister pour un contour plusieurs ordonnées à une même abscisse. Afin de réaliser un codage correct nous devons choisir celle qui devrait être calculée par le décodeur pour restituer le caractère. Cette ordonnée correspond au codage par plage.

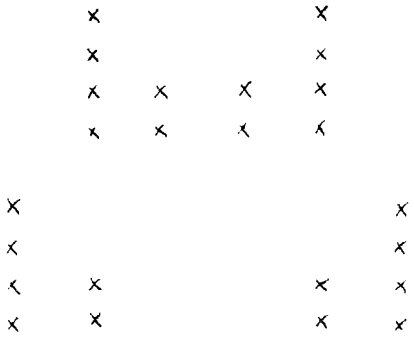
Si nous réduisons le caractère à son codage par plage nous pouvons déterminer si un point est à l'intérieur ou à l'extérieur du caractère grâce à la parité du nombre des points conservés par le codage par plage, qui le suivent ou le précédent à cette abscisse. Nous utilisons cette information pour fixer le sens de parcours d'un contour suivant qu'il est intérieur ou extérieur; ceci nous permet de simplifier l'épaississement. (§ Transformations géométriques)

A partir d'un caractère réduit à son codage par plage nous ne pouvons suivre correctement les contours dans un sens déterminé (Fig 4). Nous devons donc rajouter un codage de liaison qui relie les points du codage par plage. Le caractère est ainsi ramené à deux ensembles de points décrivant les contours: les points de liaison et les points de codage par plage.





Codage par plage



Codage de liaison

Fig 4

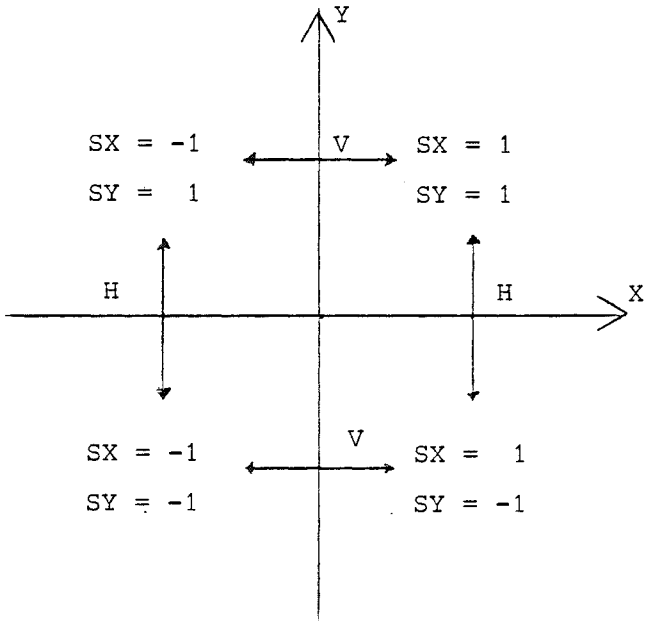


Fig 5

Afin de ne lire qu'une seule fois chaque point du contour, ceux-ci sont marqués comme tels au fur et à mesure de leur lecture. Il n'est pas possible de les effacer tous purement et simplement, car nous devons conserver trace des points du codage par plage afin d'appliquer la règle donnée ci-dessus pour différencier l'intérieur de l'extérieur.

Nous utilisons deux plans de bits ce qui nous permet de coder quatre informations par pixel:

-Point du fond ou de liaison lu

-Point de liaison

-Point du codage par plage

-Point du codage par plage lu

Dans les méthodes plus classiques de suivi de contours, il est également nécessaire d'utiliser deux plans de bits pour marquer les points déjà lus.

L'algorithme de recherche du contour analyse pour chaque point appartenant au caractère les deux voisins verticaux et horizontaux.

Soit A l'image d'origine et B l'image résultat

i l'indice de ligne et j l'indice de colonne

$A_{i,j}$ = noir si le point appartient au caractère

$A_{i,j}$ = blanc " n'appartient pas "

```
si A(i,j) = noir
   alors si A(i,j+1) = blanc ou A(i,j-1) = blanc
            alors B(i,j) = plage
            sinon
                  si A(i+1,j) = blanc ou A(i-1,j) = blanc
                     alors B(i,j) = liaison
                     sinon B(i,j) = fond
   sinon B(i,j) = fond
```

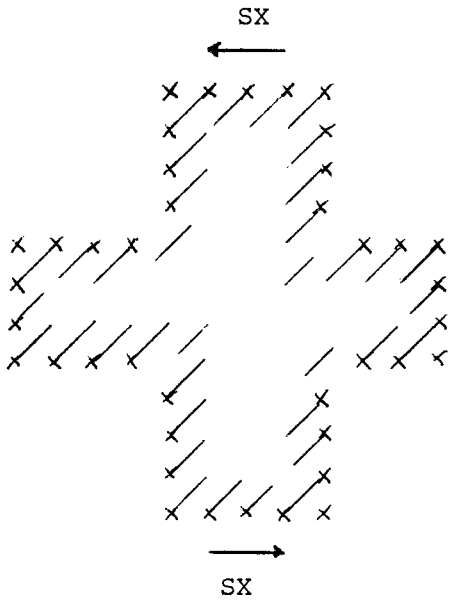
Nous devons théoriquement utiliser deux images; en pratique il n'est nécessaire de mémoriser que trois lignes ou colonnes.

3.-Suivi d'une sous paroi

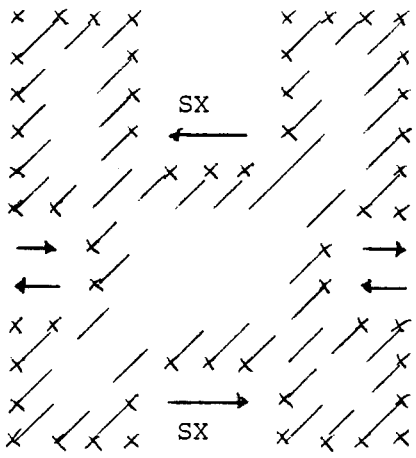
Excepté pour le premier point du contour, suivre une sous-paroi revient à rechercher le voisin du dernier point lu. Nous avons représenté le résultat de la recherche de contours pour deux formes particulières avec contour intérieur ou extérieur. (Fig 6) La première remarque est qu'il existe des points qui n'ont que deux voisins; ces points ne posent pas de problème puisqu'il ne reste qu'un seul voisin lorsqu'ils sont atteints. La deuxième remarque est que, si nous adoptons un sens de parcours différent pour les contours intérieurs ou extérieurs, l'algorithme est indépendant de ce sens et ne dépend que de la variation en X. (noté SX sur la figure) L'algorithme que nous avons utilisé teste en premier le point d'abscisse antérieure par rapport à SX et ensuite les autres voisins en tournant dans le sens trigonométrique. (Fig 7)

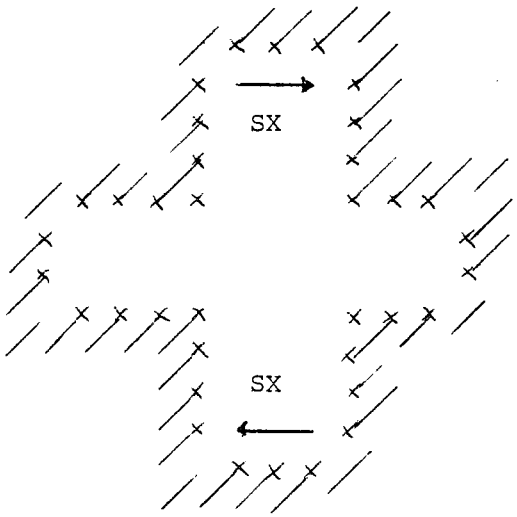
Une fin de sous-paroi peut être due à une fin de contour, simplement détectée par une absence de voisin. Un autre cas de fin de sous-paroi est un extremum local en X ou en Y. Si le contour est propre il suffit d'une variation inverse en X ou en Y. Si le contour n'est pas lisse, il est possible de le lisser en ne terminant une sous paroi que si l'écart inverse dépasse un seuil.

Afin de n'être lu qu'une fois, chaque point saisi est effacé. Les coordonnées sont enregistrées dans deux tableaux; lorsque nous lisons une nouvelle sous-paroi il est nécessaire de récupérer les points lus pour l'ancienne et non utilisés ainsi que les coordonnées du dernier point. Cette opération est réalisée grâce à deux pointeurs sur les tableaux de coordonnées; un premier pour indiquer, au programme qui calcule les interpolants, où se trouve la fin de la sous paroi et un autre pointeur qui indique où se trouve le dernier point lu.



Contours
Extérieurs





Contours
Intérieurs

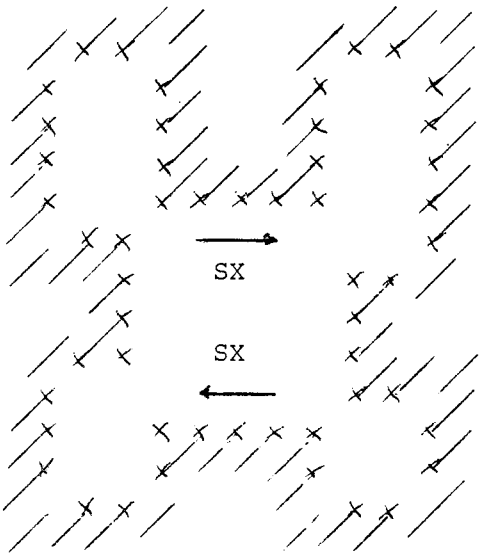


Fig 6

SX

4	3	2
5	.	1
6	7	8

SX

8	7	6
1	.	5
2	3	4

Fig 7

4.-Problèmes rencontrés

Suivi de contour

Une erreur de parcours se produit si l'épaisseur d'un trait est égale à un point de la grille. Dans le cas de la figure 8 l'algorithme lit et efface les points D, C, B, A et ne peut plus revenir au point de départ du contour O. Dans le cas de la figure 9 le programme lit correctement le contour extérieur mais pas le contour intérieur. Pour éviter ces erreurs il suffit d'imposer des traits ayant au minimum deux points d'épaisseur.

Cette contrainte n'est pas très forte puisqu'elle limite à 0.5% la taille minimale d'un trait par rapport à l'image. De plus la saisie de l'image conduit à des erreurs égales à un point de grille sur les bords.

Dans les conditions de la figure 10 l'algorithme de suivi de contour indiqué dans le paragraphe précédent commet une erreur; il indique les points dans l'ordre A, C, B, au lieu de A, B, C. Pour corriger ce défaut il suffit d'inverser le sens de variation des abscisses (SX) dans le cas d'une verticale.

En nous appuyant sur le fait que la saisie d'une image n'est précise qu'à un point de grille près, nous avons un programme de filtrage qui élimine les portions de caractère d'épaisseur inférieure à un et permet d'éviter les erreurs que nous venons d'évoquer.

Horizontales et verticales

En fin de sous-paroi la détection d'une portion de contour horizontale ou verticale pose le problème de différencier une droite et une courbe évoluée. (Fig 11) En effet, dans le cas d'une droite, nous devons transmettre une extrémité de l'horizontale ou de la verticale et, dans le cas d'une courbe évoluée, le point milieu. Comme le programme de suivi n'est pas capable de trancher, nous avons adopté la position la plus neutre. Si nous transmettons une extrémité dans le cas d'une courbe évoluée, nous introduisons une déformation importante.

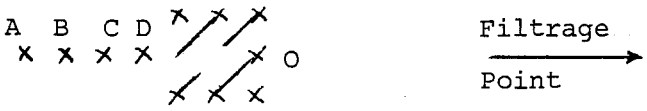


Fig 8

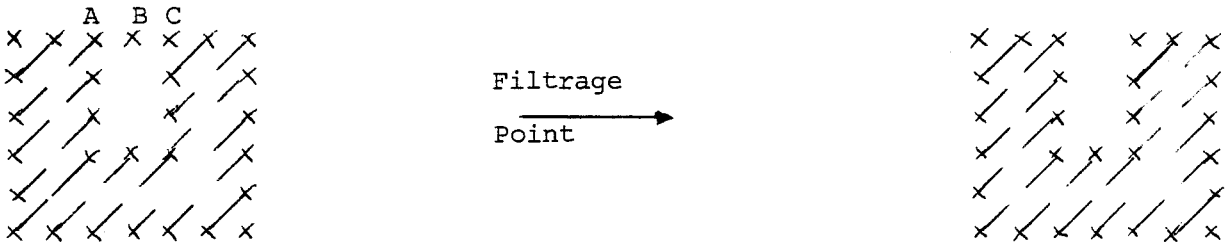


Fig 9

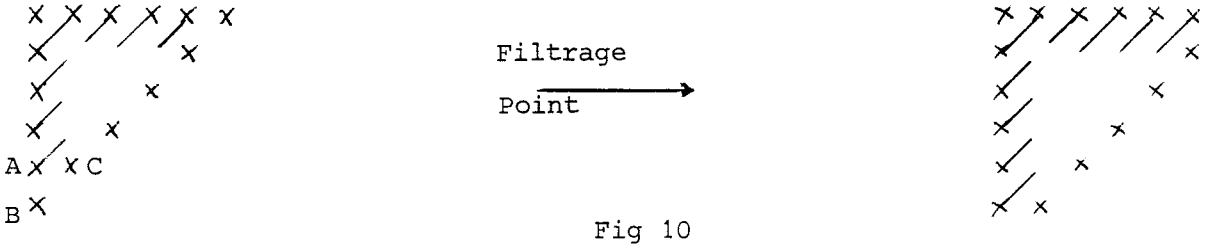


Fig 10

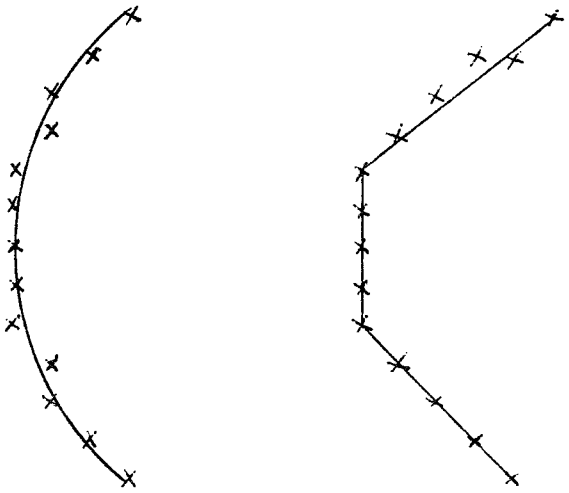


Fig 11

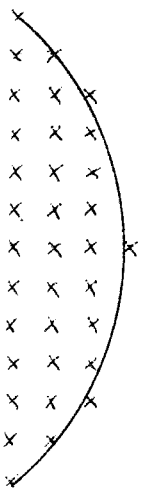


Fig 12

Par contre si nous transmettons le point milieu dans le cas d'une droite nous ne faisons que remplacer une droite par deux sans aucune déformation et comme nous le verrons plus loin cette duplication peut être supprimée.

Pour une verticale nous effectuons la moyenne des ordonnées des extrémités; pour une horizontale nous modifions le pointeur de fin de sous paroi, les points étant rangés dans deux tableaux avec une bijection abscisse-indice, et nous effectuons la moyenne de l'indice de début d'horizontale avec celui de fin.

Cercles à tangentes verticales

Lors du tracé d'un cercle à tangente verticale sur une grille par un algorithme incrémental le dernier point tracé est un point double (Fig 12). Ce point nécessaire pour coder correctement la courbe évoluée correspondante est toujours supprimé sur les documents d'origine. Chaque fois que nous rencontrons une verticale nous introduisons un point supplémentaire décalé d'une abscisse par rapport au sens de parcours en X et ayant comme ordonnée la moyenne des ordonnées des extrémités de la verticale. Cette modification évite de décaler d'un point les courbes évoluées calculées et n'apporte qu'une variation mineure et récupérable pour les droites.

Début de contour

La description du contour débute arbitrairement par le maximum global en X. La recherche de ce premier point est réalisée par une exploration colonne après colonne de la vidéo-ram jusqu'à l'obtention d'un point du codage par plage. Une fois un contour lu, la recherche du contour suivant est effectuée à partir de la dernière colonne lue.

Pour déterminer si le contour est intérieur ou extérieur nous comptons le nombre de points du codage par plage lus, situés à la même abscisse et en dessous le point de départ. Si le nombre est pair le contour est extérieur sinon il est intérieur.

Nous voulons également déterminer si le contour débute par une verticale et imposer un sens de parcours alors que nous avons un algorithme indépendant de celui-ci. Pour cela nous recherchons le point du codage par plage suivant. L'abscisse de ce dernier nous indique la présence ou non d'une verticale. Pour imposer un sens de parcours nous donnons comme dernier point lu le premier point pour un parcours dans le sens inverse et le second pour un parcours dans le sens direct.

Afin d'obtenir un contour fermé il est nécessaire de mémoriser un ou deux points en début de contour; ceux-ci seront rajoutés à la dernière sous-paroi lors de la détection d'une fin de contour.

II.-CODAGE DES SOUS-PAROIS

Les étapes précédantes ont effectué la saisie d'une sous-paroi; nous allons maintenant nous intéresser au codage de celle-ci. Nous allons examiner:

-Le choix des interpolants et de l'erreur

-L'algorithme d'approximation

-Le filtrage des droites rendu nécessaire par la correction des extrémités des courbes évoluées.

1.-Choix des interpolants et de l'erreur

Nous avons vu que les contours d'un caractère doivent être approché par des droites et des courbes évoluées. La recherche d'interpolants à un contour nécessite de fixer une condition d'arrêt à l'algorithme de décomposition. Le test le plus simple à mettre en oeuvre est la mesure de l'écart entre la courbe d'origine et l'approximation calculée. La mesure la plus souvent utilisée est la variance des écarts qui permet de lisser la courbe par rapport aux bruits de fortes amplitudes; en contrepartie les calculs sont longs et proportionnels au nombre de points controlés.

Dans notre cas nous avons un grand nombre de points, une centaine par sous-paroi en moyenne, et un bruit de faible amplitude. C'est pourquoi nous préférons utiliser l'erreur maximale comme critère d'arrêt. Au lieu de calculer la puissance d'un point par rapport à la courbe, pour mesurer l'erreur commise, nous comparons l'ordonnée calculée et l'ordonnée du contour. Bien entendu cette mesure doit être corrigée en la multipliant par le cosinus de l'angle de la tangente à la courbe. Le cas d'une droite verticale qui conduirait à une division par zéro ne peut se produire puisque nous ne notons qu'une ordonnée par abscisse.

2.-Algorithme d'approximation

Pour chaque portion de contours nous avons deux interpolants possibles; nous devons donc fixer une priorité entre ces derniers. La première solution consiste à approximer le contour par des droites; ensuite si nous avons plus de quatre droites pour une sous-paroi nous essayons de remplacer les droites par une courbe évoluée. La seconde solution consiste à placer en premier une courbe évoluée en définissant des critères qui conduisent à placer une droite.

La première méthode fait perdre la continuité de la tangente au contour et demande des calculs inutiles dans le cas où l'interpolant est finalement une courbe évoluée. Ces deux remarques nous ont conduits à adopter la deuxième méthode.

Il nous reste à préciser les critères de remplacement d'une partie de courbe évoluée par une droite. Dans le cas où le contour est une droite entre les deux extrémités A et B de la sous-paroi, il n'existe pas de point de carrure F (Fig 13); de même la recherche des tangentes nous conduit à déterminer le point de carrure pour chaque demi courbe évoluée, et son absence nous amène à utiliser une droite (Fig 14).

Une fois les paramètres de la courbe évoluée calculés nous devons mesurer l'écart maximum entre le contour et la courbe; si l'erreur est trop importante nous itérons l'essai d'une courbe évoluée sur chaque demi-courbe évoluée.

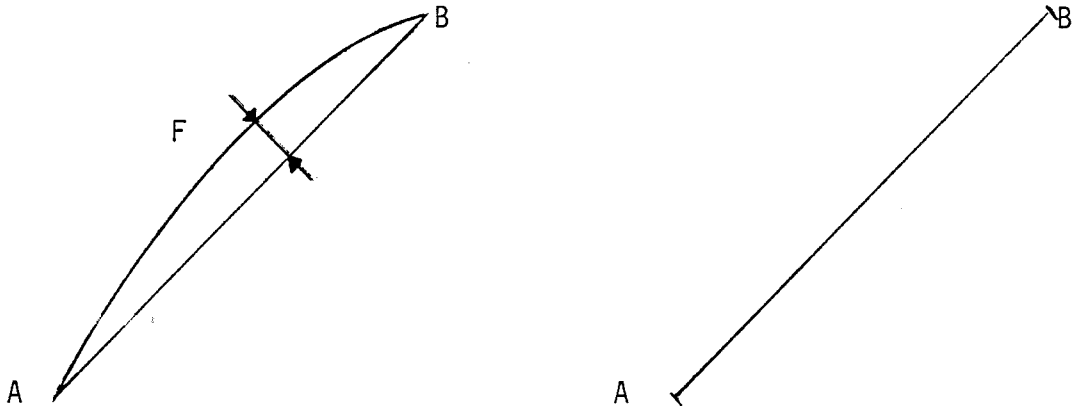


Fig 13

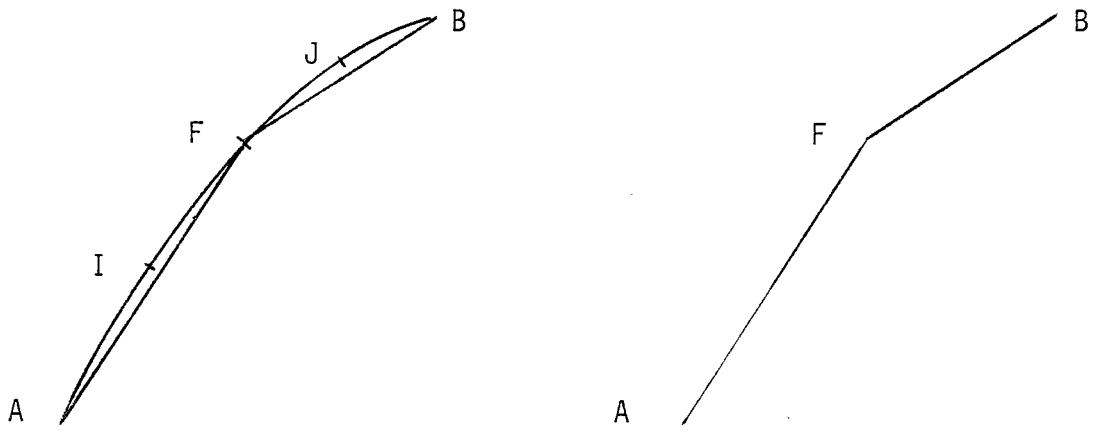


Fig 14

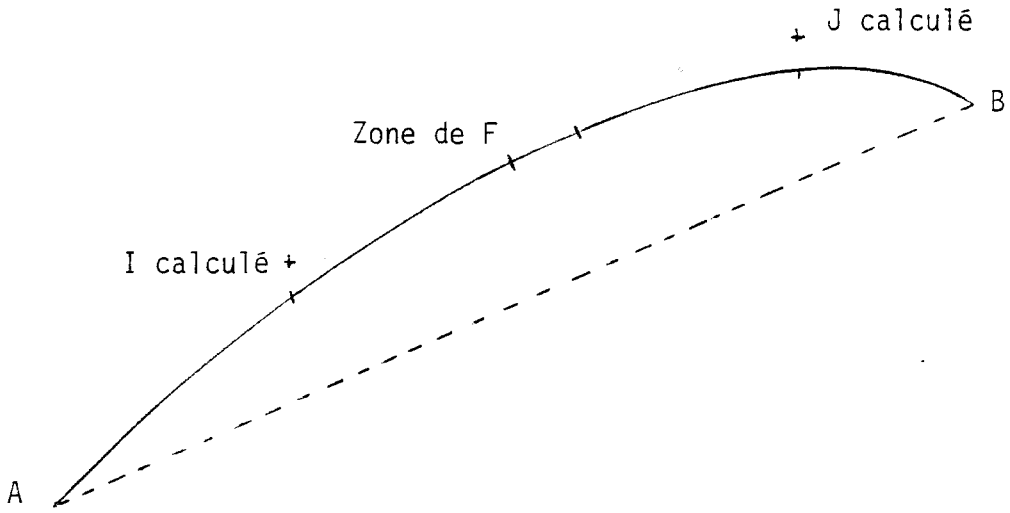


Fig 15

A chaque itération nous appliquons les critères de détection de droite. En pratique la mesure de l'écart est effectuée indépendamment sur chaque demi-courbe évoluée. Pour limiter les calculs nous avons introduit la demi-courbe évoluée comme interpolant; ainsi lorsqu'un essai donne une moitié correcte et l'autre pas, nous itérons uniquement sur la portion incorrecte.

3.-Calcul de l'interpolant

La connaissance de ses extrémités suffit à déterminer une droite; par contre, pour une courbe évoluée, il nous faut calculer en plus les tangentes en ces points et le point de carrure.

Trouver le point de carrure est une opération facile si la sous-paroi ne présente pas de point d'inflexion; dans ce cas celui-ci est le point du contour le plus éloigné de la corde passant par les extrémités.

Ajustement du point F

Lors de la recherche d'un point de carrure l'algorithme mesure une distance entre une droite et un contour qui ont tous deux des coordonnées quantifiées; ainsi dans la plupart des cas il n'existe pas un point optimal mais plusieurs; ceux-ci appartiennent à un segment parallèle à la droite. (Fig 15)

Nous avons dans un premier temps considéré le point milieu de ce segment comme point de carrure. Ensuite pour déterminer l'influence de la position de ce point sur le résultat nous avons prévu un ajustement automatique de ce point. Pour cela nous additionnons les écarts entre les deux points intermédiaires de la courbe évoluée et le contour et nous déplaçons le point F de façon à minimiser cette somme. Dans le cas où une moitié de courbe est incorrecte, c'est à dire où l'écart est supérieur au seuil, nous indiquons une distance nulle afin de minimiser l'erreur sur la demi-courbe correcte. Dans le programme nous avons trois possibilités:

-Point F au milieu du segment

-Minimisation des écarts en valeur absolue

-Minimisation des écarts signés

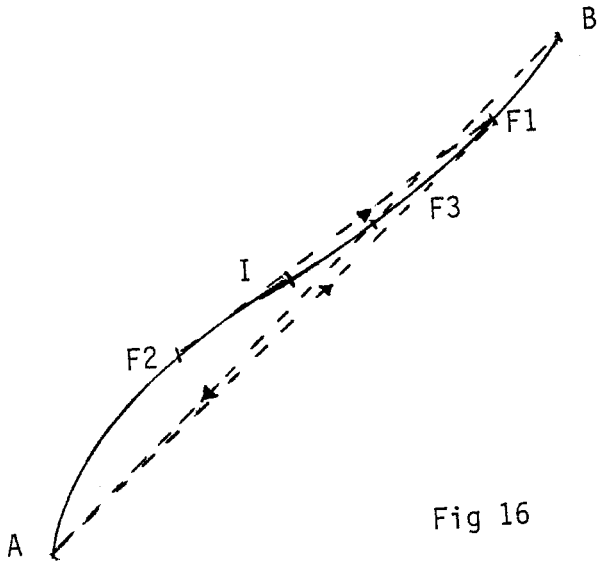


Fig 16

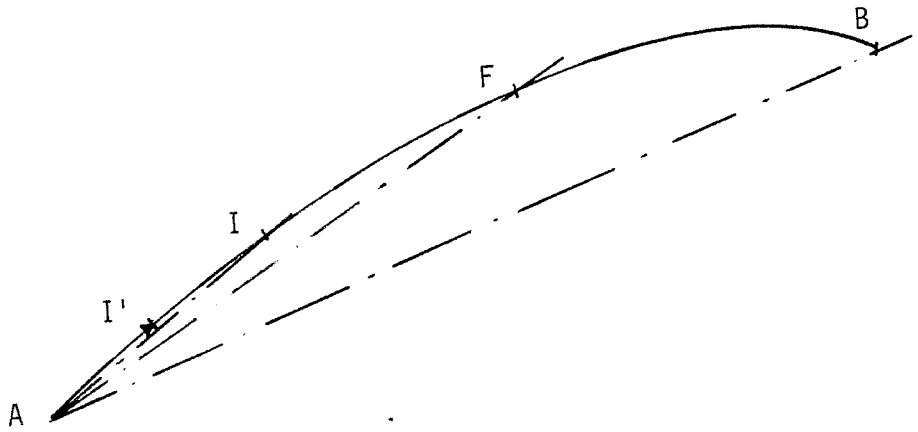


Fig 17

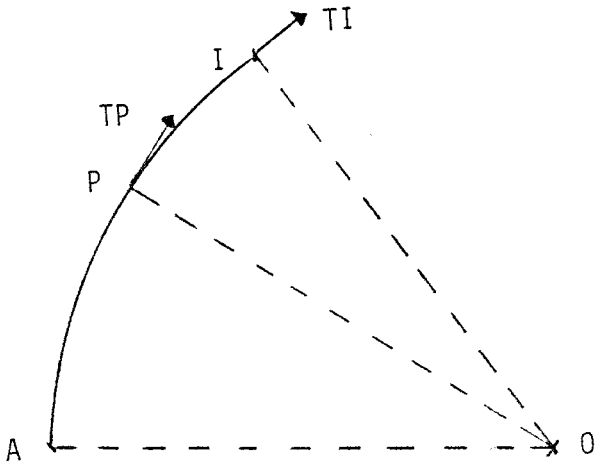


Fig 18

Point d'inflexion

C'est un point particulier où le changement de concavité impose un changement de courbe évoluée; n'étant pas un extrémum local en X ou en Y, il n'est pas détecté comme une fin de sous-paroi par le programme de suivi de contour. La détection de ces points est réalisée au cours de la recherche des points de carrure; le changement de signe de l'écart entre la corde et le contour indique la présence d'un point d'inflexion. (Fig 16)

Dans le cas où l'écart entre la corde et le contour présente plusieurs maxima nous donnons comme "point de carrure" le point qui correspond au premier maximum rencontré dans le sens de parcours de la corde.

La localisation d'un point d'inflexion est réalisée en itérant le même algorithme de recherche de point de carrure. A chaque itération nous remplaçons l'extrémité d'origine de la corde par le point de carrure et, dans le cas où l'écart change de signe, nous inversons le sens de parcours de la corde. L'algorithme s'arrête lorsqu'il ne trouve plus de point de carrure; le point d'inflexion se situe au milieu de la dernière corde.

Mesure des tangentes aux extrémités

Il est possible de mesurer les tangentes aux extrémités en utilisant l'algorithme de recherche du point de carrure. Nous itérons celui-ci en remplaçant à chaque pas une extrémité par le point de carrure; l'arrêt est produit par l'absence de point de carrure. La direction de la tangente est donnée par la dernière corde testée (Fig 17).

La mesure des tangentes par cette méthode fournit des résultats très peu précis. La cause de ces erreurs est l'opposition entre la définition théorique de la tangente, qui indique que la précision augmente lorsque la corde diminue, et la mesure d'écart entre valeurs quantifiées dont la précision relative diminue lorsque la différence diminue. (LAR 77)

En pratique cette imprécision provoquait le rejet de toute courbe évoluée et conduisait à un codage par droites; nous avons donc abandonné cette méthode de mesure de tangentes.

Calcul des tangentes par le tracé de cercles

Nous venons de voir que nous ne pouvons pas déterminer les tangentes directement à partir des points décrivant le contour. Nous allons calculer celles-ci en utilisant l'algorithme de tracé de cercle (HOR 77,BRE 77).

Les coefficients utilisés par cet algorithme sont les dérivés premières de la fonction implicite du cercle. Nous pouvons donc calculer ces valeurs pour chaque point du cercle où nous connaissons l'angle de la tangente.

Soit T l'angle de la tangente

[1] $C = -k \sin T$

$S = k \cos T$

D'après l'algorithme incrémental nous avons une relation simple entre les coefficients pour deux points A et I appartenant au cercle.

[2] $CA - CI = XI - XA$

$SA - SI = YI - YA$

Nous supposons trois points A, P, I appartenant à un cercle; nous connaissons les angles TP et TI; nous recherchons l'angle TA en A. (Fig 18)

D'après les relations [1] et [2] nous avons:

[3] $-kI \sin TI + kP \sin TP = - XI + XP$

$kI \cos TI - kP \cos TP = - YI + YP$

[4] $-kP \sin TP + kA \sin TA = - XP + XA$

$kP \cos TP - kA \cos TA = - YP + YA$

D'après [3] nous pouvons calculer kP:

$$kP = \frac{(XP-XI) \cos TI + (YP-YI) \sin TI}{\cos TI \sin TP - \cos TP \sin TI}$$

D'après [4] nous avons:

$$tg\ TA = \frac{XA - XP + kP \sin TP}{-YA + YP + kP \cos TP}$$

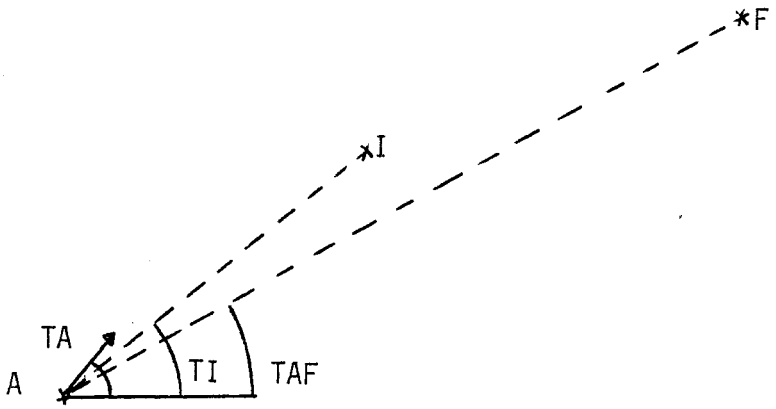

F
I
TA
A
TI
TAF

Fig 19

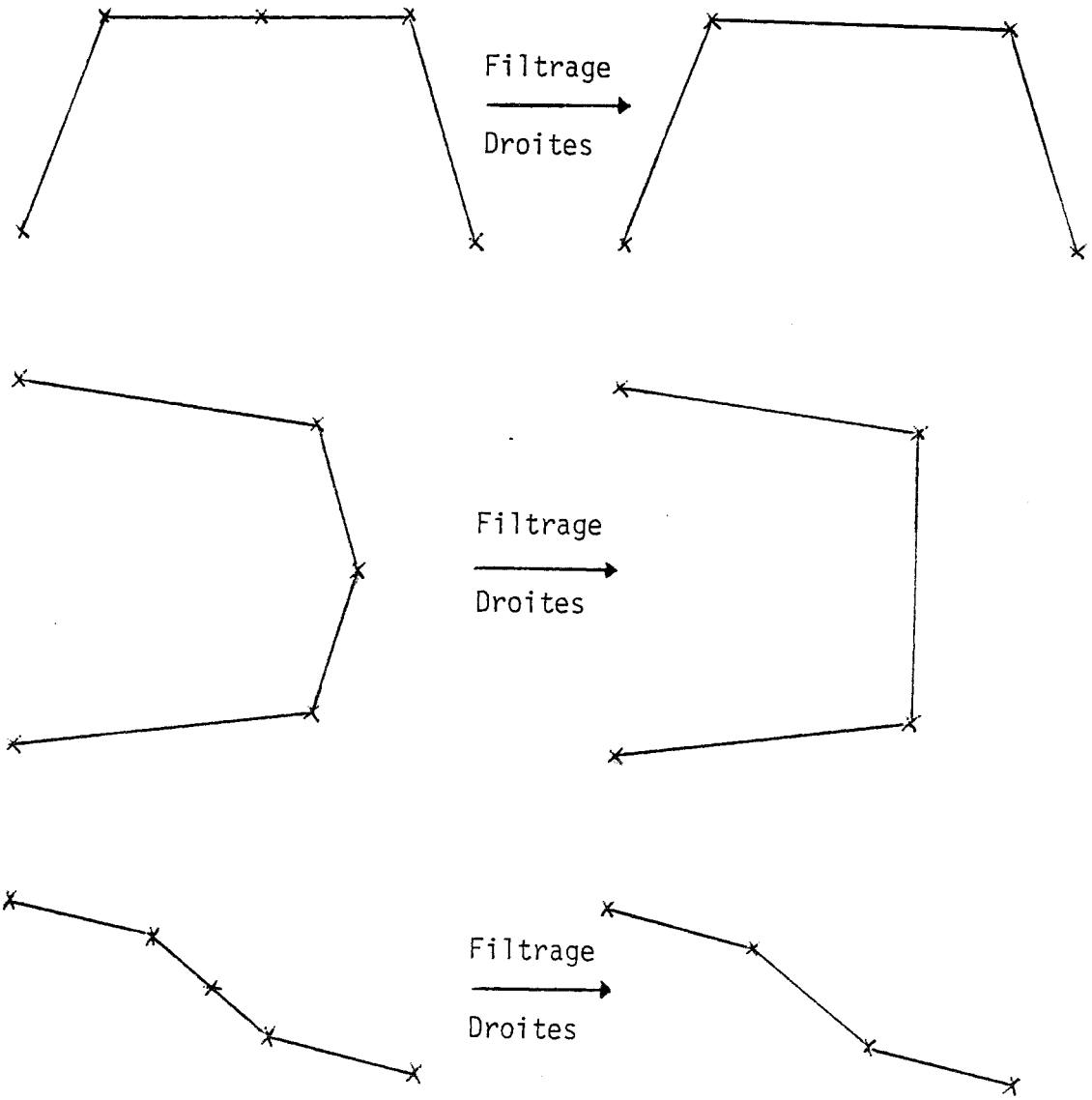

Filtrage
Droites
Filtrage
Droites
Filtrage
Droites

Fig 20

Dans la décomposition des courbes évoluées en cercles le point intermédiaire est un point de carrure. Le point I est le point de carrure de la courbe AF, TI = Artg (YF-YA)/(XF-XA). Le point P est le point de carrure de la courbe AI, TP=Artg(YI - YA)/(Xi - XA). Ces deux points sont localisés sur le contour grâce à l'algorithme de recherche de point de carrure.

Une fois les tangentes aux extrémités connues, nous décomposons la courbe évoluée en cercles, nous traçons et comparons ceux-ci au contour de façon à accepter ou rejeter cette courbe.

Cette méthode donnait des résultats corrects; la méthode suivante apporte une simplification du calcul et évite de tracer et tester les cercles obtenus.

Calcul des tangentes par les points intermédiaires

Nous utilisons ici la méthode de recherche des points intermédiaires présentée au chapitre II.

Le point intermédiaire I est situé sur la bissectrice de la tangente en A et de la droite AF. (Fig 19) Nous sommes capables de trouver le point I sur le contour et nous cherchons la tangente en A.

D'après la démonstration:

TI = (TA + TAF)/2

Nous avons donc:

TA = 2 TI - TAF

Pour contrôler l'approximation, nous calculons la position du point intermédiaire d'après les paramètres définis et nous mesurons l'écart entre ce point et le contour.

Cette méthode donne d'excellents résultats; c'est celle que nous utilisons dans le programme d'approximation de contours.

4.-Filtrage des droites

Lorsqu'une sous-paroi se termine par une horizontale ou une verticale, le programme de suivi du contour indique le milieu de celle ci comme fin de sous-paroi; de plus dans le cas d'une verticale il décale d'une abscisse le point milieu.

Ces modifications nous permettent de coder correctement les courbes évoluées à tangentes verticales ou horizontales; par contre nous obtenons deux droites horizontales au lieu d'une et aucune droite verticale. (Fig 20)

Un point d'inflexion présente une analogie avec une fin de sous-paroi horizontale: nous obtenons deux droites contiguës de même pente de part et d'autre du point d'inflexion au lieu d'une droite le contenant.

Si nous traçons le contour décrit par ce fichier, nous obtenons des erreurs d'un point de grille sur les points milieux des verticales uniquement. La duplication des horizontales et pseudo verticales est plus gênante puisqu'elle nuit à la densité du code.

Pour comprimer et corriger les verticales nous testons si deux droites contiguës peuvent être remplacées par une seule et dans l'affirmative nous supprimons le point milieu.

V.-STRUCTURE D'ARBRE

Les informations contenues dans la structure d'arbre permettent d'accélérer les calculs du décodeur. Nous trouvons les abscisses des points de naissance associées à des pointeurs vers la structure d'anneau, les numéros d'ordre et de paroi.

Les abscisses de naissance sont les minima locaux en X et indiquent les abscisses où le générateur de caractères introduit de nouvelles parois à calculer. Ces abscisses de naissance sont déterminées par une exploration de la description du contour. Les pointeurs qui leur sont associés doivent désigner la position où sera la première coordonnée lors de l'étape de suivi des contours; c'est à dire après l'expansion du fichier, nous devons donc simuler les déplacements dûs au décodage. Nous rajoutons deux valeurs nulles pour les droites et six mots pour les courbes évoluées.

Le numéro d'ordre permet de classer rapidement les ordonnées calculées par un tri indexé. Les parois ne peuvent se croiser et les numéros ne sont pas obligatoirement contigus, ce qui nous permet d'allouer un numéro unique par paroi. Nous supposons à un instant donné une liste des parois actives classées; l'emplacement dans la liste constitue un numéro d'ordre provisoire. Ce numéro devient définitif et l'emplacement correspondant libre à la mort de la paroi. A chaque abscisse de naissance nous devons interclasser les nouvelles parois dans la liste. Ce classement est facilité si nous avons noté, pour chaque abscisse de naissance au cours du codage dans l'étape de suivi des contours, le nombre de parois au dessus et au dessous de l'ordonnée de naissance.

Le numéro de paroi permet une allocation rapide sur une machine ayant des processeurs en parallèle. La principale difficulté est d'équilibrer la charge sur chaque processeur sans connaitre le nombre de processeurs qui seront utilisés. Ce problème a été présenté dans la thèse de M. BLOCH. (BLO 81)

VI.-DECODAGE

Le programme de codage tel qu'il est écrit ne calcule pas la structure d'arbre; l'étape de décodage ne peut donc pas générer en une seule passe le caractère. En pratique nous utilisons la vidéo-ram pour classer et mémoriser les points du codage par plage au fur et à mesure de leurs calculs; ensuite nous relisons la vidéo-ram et nous graissons le caractère en reliant les points du codage par plage par des vecteurs. Au début nous tracions le contour de manière purement séquentielle dans le sens de la description, ce qui provoquait une assymétrie inesthétique notamment dans le cas du O. Pour remédier à ce défaut nous traçons désormais chaque arc de cercle ou droite dans le sens des abscisses croissantes, ce qui simule le tracé du générateur de caractères fabriqué.

Pour obtenir un graissage correct nous devons respecter la parité du nombre d'ordonnées pour un intérieur, dans ces conditions nous devons choisir entre marquer le premier ou le dernier point d'un interpolant; par analogie avec le générateur de caractères nous marquons le premier point.

Dans le générateur de caractères construit, le problème des points doubles ne se pose pas; leurs ordonnées sont transmises deux fois. Dans le cas du décodeur logiciel les points doubles ne peuvent être marqués spécifiquement sur un seul plan de bits. Les deux solutions possibles sont, soit d'utiliser un deuxième plan de bits pour marquer les points doubles, soit de les effacer. La première méthode correspond au comportement du générateur de caractère et évite les coupures dans les parties déliées lors de fortes réductions.

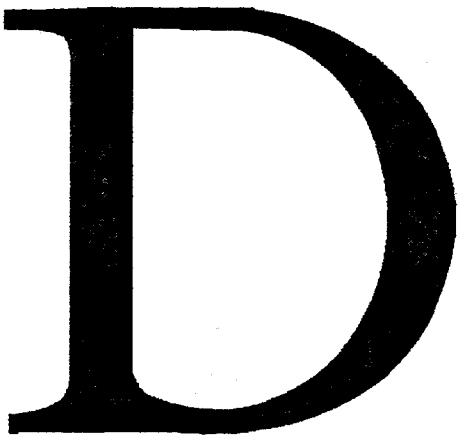
VII.-PERFORMANCES

1.-Réalisation

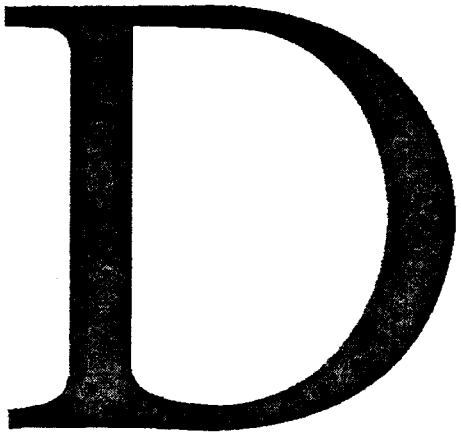
Ce programme a été écrit pour prouver la faisabilité de ce produit. Nous avons écrit la majorité du programme en FORTRAN IV en réservant l'assembleur pour les routines élémentaires de manipulation concernant la caméra de saisie et les vidéo-ram.

Pour simplifier l'étude nous n'avons pas calculé la structure d'arbre, puisque nous sommes capables de tracer les formes sans celle-ci.

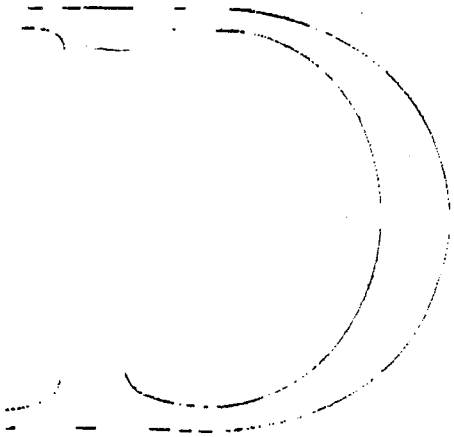
Les primitives, définies par M. HOURDEQUIN (HOU 78) comme des portions de contour identiques utilisées dans plusieurs caractères d'une même police, sont très difficiles à rechercher de manière automatique. Nous n'avons fait aucune étude concernant la possibilité et la complexité d'une telle réalisation.



Original

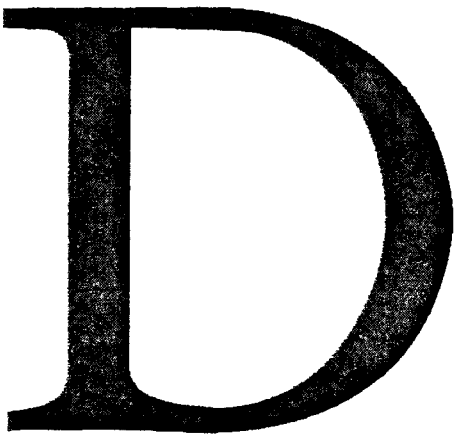


Codé sans ajustement du point F

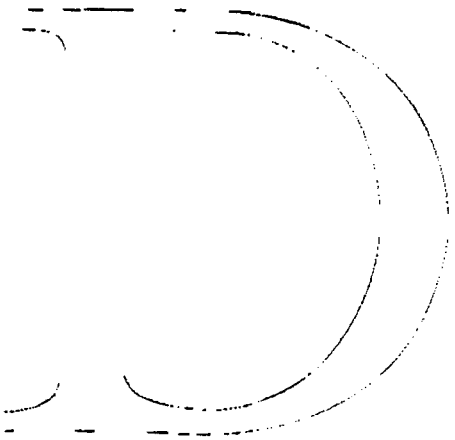


Différence codé-original

Fig 21

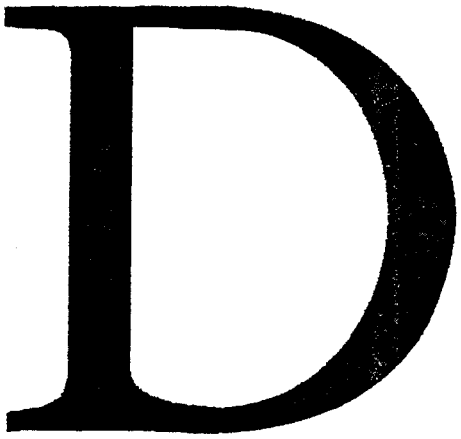


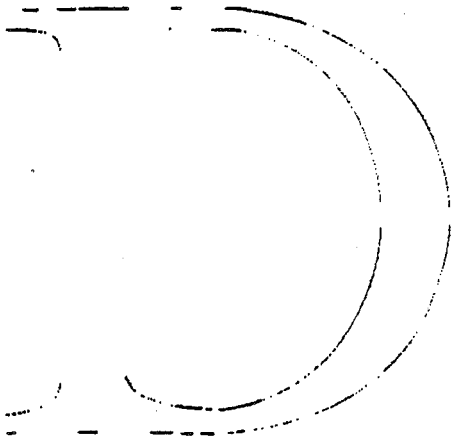
Codé ajustement point F signé



Différence codé-original

Fig 22

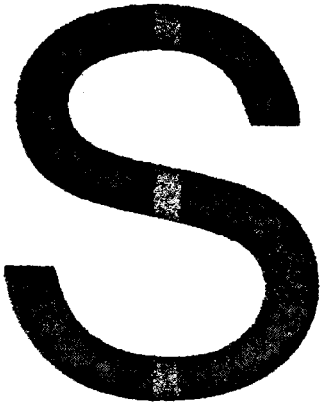




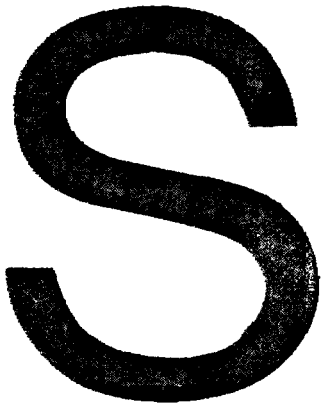
Codé ajustement point F non signé

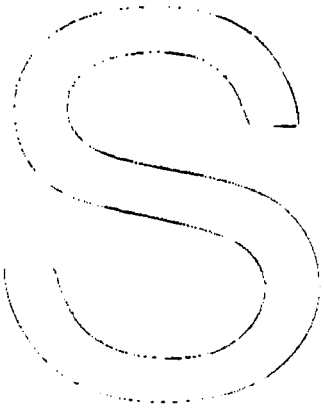
Différence codé-original

Fig 23



Original





Codé sans ajustement du point F

Différence codé-original

Fig 24

2.-Codage par droites et courbes évoluées

Pour juger la qualité obtenue par ce codage automatique nous utilisons un programme de comparaison entre l'original mémorisé grâce à un codage par plage sur disque et le résultat du décodage placé en vidéo-ram. (Fig 21-22-23)

Le codage du D de GARAMOND donne une erreur maximale de un point dans les parties concaves et aucune erreur sur les parties rectilignes. En ce qui concerne la densité du code nous obtenons pour le D de GARAMOND 10 droites, 7 courbes évoluées et 1 demi-courbe évoluée, ce qui nous donne en tenant compte de la structure d'arbre 1376 bits. Un codage manuel aurait donné 8 droites et 7 courbes évoluées soit 1304 bits avec une qualité moindre.

Les diverses planches présentent le codage sans ajustement du point F, avec l'ajustement en valeur absolue et avec l'ajustement en valeur signée; nous présentons également pour illustrer le cas d'un caractère ayant des points d'inflexion le codage d'un S. (Fig 24)

3.-Codage par droites uniquement

Afin de juger des résultats d'un codage par droites uniquement nous avons inclu cette possibilité dans le programme de codage automatique. (Fig 25)

La qualité des contours est moindre puisque les changements de droites dans les parties concaves forment des angles très inesthétiques. En ce qui concerne la densité le programme utilise 59 droites pour coder le D de GARAMOND soit en comptant la structure d'arbre 3272 bits, ce qui représente un code près de trois fois moins dense. La conclusion que nous pouvons tirer de cet essai est que le critère d'erreur maximale n'est pas suffisant dans le cas d'un codage par droites uniquement, il faudrait rajouter un critère de continuité des tangentes dans les parties concaves. L'application d'un tel critère conduirait à une augmentation du nombre de droites, ce qui dégraderait encore la densité de ce type de codage.

4.-Temps de calul

Afin de donner un ordre de grandeur du temps d'exécution des diverses opérations, nous avons effectué des mesures résumées dans le tableau de la figure 26. Les manipulations en vidéo-ram: filtrage, extraction de contours, recherche du premier point du contour sont relativement longues par rapport au codage proprement dit.

Une première amélioration consisterait à écrire ces routines en assembleur: nous pouvons ainsi espérer un gain d'un facteur 5. L'utilisation d'une vidéo-ram microprogrammée permettrait des gains de l'ordre de 20 et équilibrerait les temps des diverses opérations.

Lors du tracé du caractère nous retrouvons les mêmes disproportions entre le calcul du contour et le graissage.

Avec les programmes en FORTRAN le codage d'un caractère avec la saisie et les deux filtrages demande 390 s; la mise en assembleur des utilitaires donnerait 98 s; avec un gain de 5.

Codé par droites uniquement Différence codé-original

Fig 25

Temps d' exécution des diverses opérations

		Temps en s	
Saisie image		10	
Rappel image codée par plage		25	
Filtrage points		80	
Filtrage lignes		180	
Calcul contour (ajustement)		120	(130)
dont	Extraction contours	80	(80)
	Recherche premier point	25	(25)
	Calcul	10	(20)
	Filtrage droites	5	(5)
Tracé d' un caractère		35	
dont	Calcul contour	2	
	Graissage	33	

Fig 26

Chapitre 5

CHAPITRE 5

TRANSFORMATIONS GEOMETRIQUES

I.-PRESENTATION

II.-FORMULAIRE

III.-PROBLEMES LIES AUX TRANSFORMATIONS LOCALES

1.-La mise à l'échelle
2.-L'épaississement
3.-Répercussion sur l'étape d'expansion de courbes évoluées
4.-Conclusion

IV.-TRANSFORMATIONS GLOBALES

1.-Modifications
2.-Transformations
3.-Conclusion

V.-INDICATION SUR LA REALISATION

1.-Simulation
2.-Combinaison des transformations

3.-Insertion dans le pipe-line

4.-Mise à l'échelle

5.-Italique

6.-Rotation

7.-Epaississement

8.-Transformations globales

9.-Regroupement

TRANSFORMATIONS GEOMETRIQUES

I.-PRESENTATION

Une transformation géométrique est pour nous une modification en temps réel de la forme générale d'un caractère sans destruction de l'information qui permet de le reconnaitre.

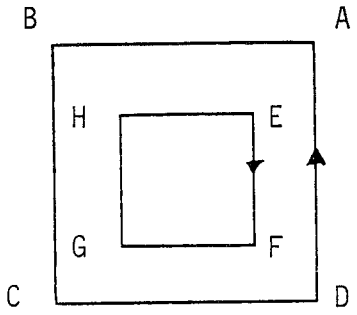
Cette notion de temps réel nous interdit les transformations trop complexes telles que l'embellissement ou la génération de caractères à partir d'un code structurel tel que celui étudié par COUEIGNOUX (COU 80). En contrepartie la possibilité de calculer les transformations en ligne nous permet d'engendrer un grand nombre de formes à partir d'un seul modèle. Nous obtenons ainsi une densité apparente du code plus grande au prix d'un accroissement raisonnable de la complexité de la machine.

Les transformations que nous allons étudier dans ce chapitre sont:

-La mise à l'échelle
-L'inclinaison à 13° (Italique)
-L'épaississement
-La rotation

La première est d'un usage courant dans la composition et elle a été prévue dès le début du projet. Les autres opérations, qui n'ont pas été réalisées sur la machine, nous sont apparues intéressantes pour élargir le champ d'application du générateur de caractères vers le traitement de texte et l'insertion de textes sur les plans et les cartes.

L'utilisation des transformations géométriques n'est pas nouvelle, mais les calculs sont liés au codage retenu. Dans le cas d'un codage par matrice il est nécessaire d'effectuer n^2 calculs, plus un cadrage de la nouvelle matrice sur la grille de tracé. Dans le cas d'un codage par plage le nombre de calculs est réduit à kn avec toujours la nécessité de recadrer la nouvelle matrice.


B
A
H
E
G
F
C
D

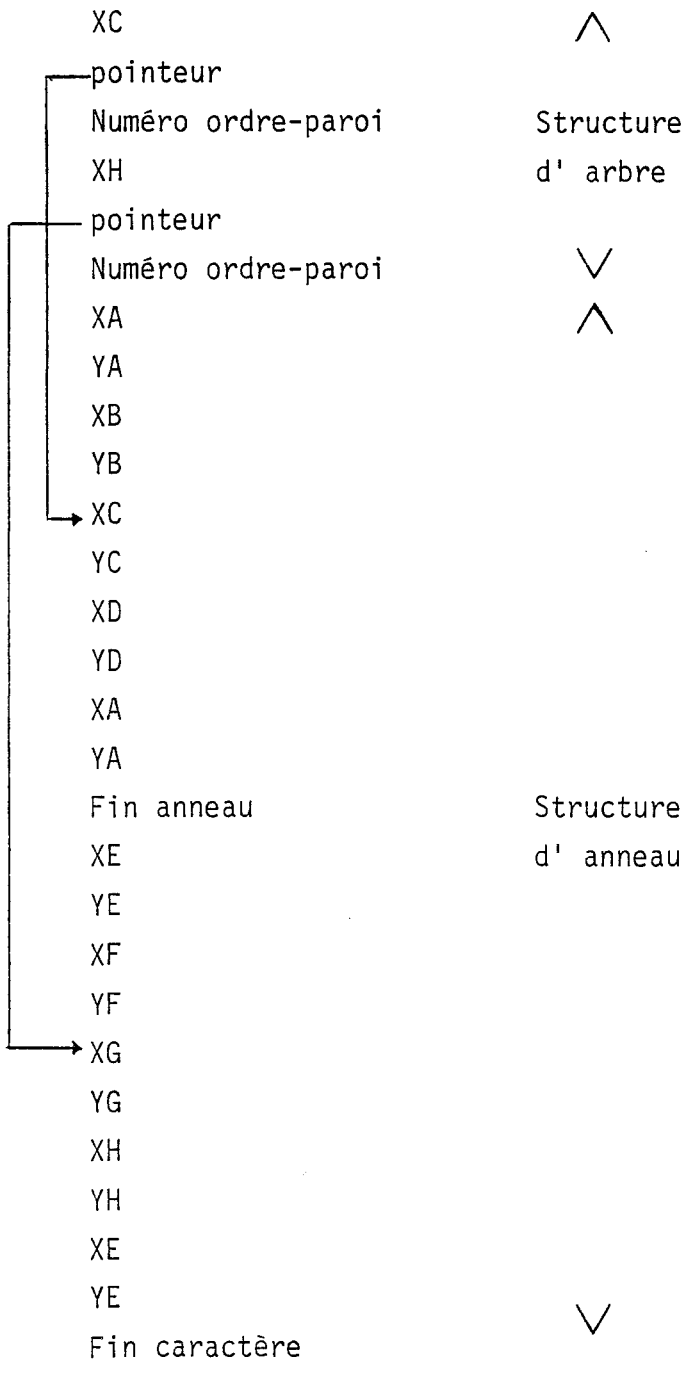

XC
pointeur
Numéro ordre-paroi
XH
pointeur
Numéro ordre-paroi
XA
YA
XB
YB
XC
YC
XD
YD
XA
YA
Fin anneau
XE
YE
XF
YF
XG
YG
XH
YH
XE
YE
Fin caractère
Structure
d' arbre
Structure
d' anneau

Fig 1

Du point de vue nombre de calculs le codage par contour est très intéressant puisque nous ne modifions que les coordonnées des points décrivant le contour soit un nombre d'opérations pratiquement indépendant de n.

Le fichier contenant la description du caractère est composé de deux parties: la structure d'arbre et la structure d'anneau.(Fig 1)

La structure d'anneau contient les paramètres des interpolants, droites et courbes évoluées, qui permettent de recalculer la forme du caractère.

La structure d'arbre contient des informations globales sur la structure du caractère. Les abscisses de naissance, associées à des pointeurs, permettent d'engendrer le caractère en une seule passe. Les numéros de parois autorisent des allocations rapides et équilibrées de façon à répartir la charge sur l'ensemble des processeurs placés en parallèle. Les numéros d'ordre accélèrent le tri des ordonnées calculées qui doivent être transmises classées vers l'organe de visualisation.

Toutes les transformations étudiées modifient les paramètres contenus dans la structure d'anneau et, dans des proportions variables ceux de la structure d'arbre. Nous appelons transformation locale l'effet sur la première et transformation globale l'effet sur la seconde. Nous étudierons tour à tour ces deux types d'effet.

II.-FORMULAIRE

Transformations locales

Nous allons distinguer les transformations locales strictes et les tranformations locales au sens large.

Les transformations locales strictes déduisent les nouveaux paramètres d'un point du contour à partir des valeurs de transformation et des paramètres de ce point uniquement.

	Mise à l' échelle	Italique	Rotation
X'	$k_x X$	X + k Y	X cosB - Y sinB
Y'	$k_y Y$	Y	Y cosB + X sinB
A'	Artg k_x/k_y tg A	Artg tg A/1 + k tg A	A + B
	k_x échelle en X k_y échelle en y	k = tg 13°	B angle de rotation

Fig 2

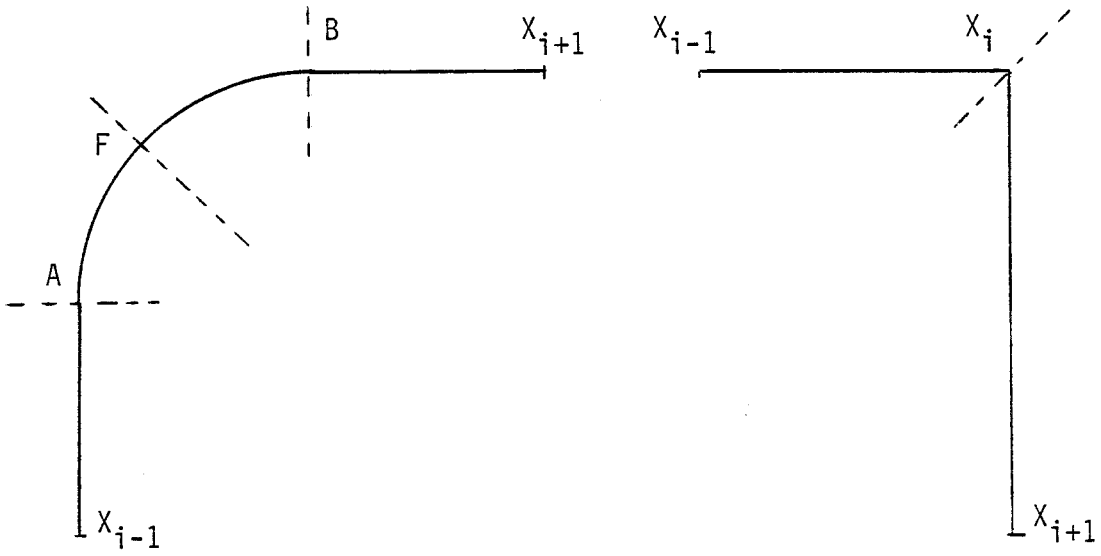


Fig 3

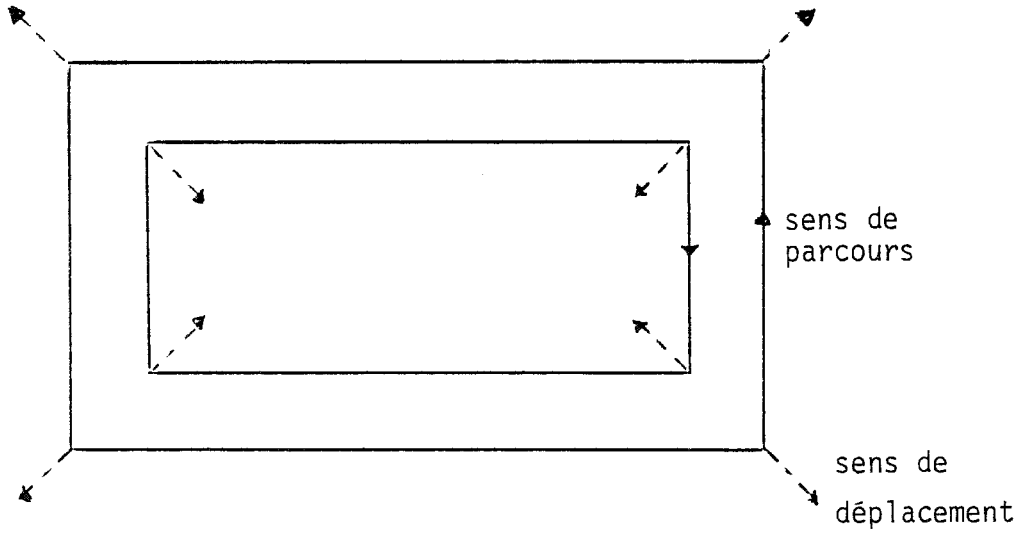


Fig 4

Nous obtenons la formule générale:

$(x' , y' , A') = f(x , y , A , k)$

x' et y' les nouvelles coordonnées

A' nouvel angle de la tangente dans le cas d'une extrémité de courbe évoluée

k valeurs de transformation

Les transformations géométriques qui demandent des transformations locales strictes sont, parmi celles étudiées, la mise à l'échelle la rotation et l'italique.

Nous avons réuni dans un tableau (Fig 2) les diverses formules pour les transformations précitées. Les formules concernant les angles ne sont utilisées que pour les courbes évoluées, les paramètres des droites ne comportent pas d'angle.

Dans le cas de transformations locales au sens large il n'est plus possible d'appliquer la formule générale donnée au dessus. L'exemple étudié est celui de l'épaississement. Pour calculer les nouveaux paramètres nous devons respecter la continuité des tangentes et donc tenir compte des deux demi-tangentes en chaque point. (Fig 3)

Soit EP la valeur de l'épaississement

AA la demi tangente antérieure au point (x, y)

AP la demi tangente postérieure au point (x, y)

$x' = x - EP \sin (AA + AP)/2$

$y' = y + EP \cos (AA + AP)/2$

A' = A (courbes évoluées)

Dans le cas d'une droite avant le point (x, y)

$AA = \text{Artg } (y_{i-1} - y_i \,/\, x_{i-1} - x_i)$

Dans le cas d'une droite après le point (x, y)

$AP = \text{Artg } (y_{i+1} - y_i \,/\, x_{i+1} - x_i)$

En début de courbe évoluée

AP = A

En fin de courbe évoluée

AA = A

Pour le calcul du point F

$AA = AP = \text{Artg } (y_A - y_B \,/\, x_A - x_B)$

Lors de l'épaississement d'un caractère les points situés sur le contour extérieur et ceux situés sur le contour intérieur doivent être déplacés dans une direction opposée (Fig 4). Ce problème disparait si nous adoptons la convention de parcourir les contours extérieurs dans le sens trigonométrique et les contours intérieurs dans le sens inverse. Cette contrainte est très facile à respecter au niveau de l'édition des caractères. Par contre déterminer si le contour est interne ou externe à partir de sa description est une opération plus complexe qui devrait de plus être réalisée en ligne.

INVARIANCE DU POINT F

D'après la démonstration sur l'expansion des courbes évoluées donnée au chapitre 2, la position du point F est arbitraire si nous connaissons la tangente en ce point. Nous avons également vu que pour comprimer le fichier d'entrée il est intéressant que la tangente en F soit calculée à partir des autres paramètres. Il existe deux points particuliers qui répondent à ce critère: le point de carrure et le point de tangence moitié. Nous obtenons deux formules de calcul:

point de carrure $Y_F' = YB - YA / XB - XA$

point tangente moitié $TF = 1/2\ (TA + TB)$

Nous allons pour ces deux points comparer le résultat du calcul à partir des paramètres modifiés par la transformation géométrique et le résultat de l'application de la transformation sur l'angle ou la tangente en F.

Point de carrure

Mise à l'échelle

Application $\underline{Y_F}' = (k_y / k_x)\ Y_F'$

Calcul $\underline{Y_F}'' = (k_y\ YB - k_y\ XB) / (k_x\ XB - k_x\ XA) = \underline{Y_F}'$

Italique

Application $\underline{Y_F}' = Y_F' / 1 + k\ Y_F' \qquad (k = tg\ 13°)$

Calcul $\underline{Y_F}'' = (YB - YA) / (XB - XA + k(YB - YA) = \underline{Y_F}'$

Rotation B angle de rotation

Application $\underline{Y_F}' = tg\ (Arctg\ Y_F' + B)$

$= (Y_F' + tg\ B) / (1 - Y_F'\ tg\ B)$

Calcul $\underline{Y_F}'' = \dfrac{(YB - YA)\cos B + (XB - XA)\sin B}{(XB - XA)\cos B - (YB - YA)\sin B} = \underline{Y_F}'$

Epaississement

Application $\underline{Y_F}' = Y_F'$

Calcul $\underline{Y_F}'' = \dfrac{YB - YA + EP(\cos A1 - \cos A2)}{XB - XA + EP(\sin A2 - \sin A1)} \neq \underline{Y_F}'$

Point tangente moitié

Mise à l'échelle

Application $T_F' = Artg((k_y / k_x)\ tg\ T_F)$

Calcul $T_F'' = 1/2\ (Artg\ ((k_y / k_x)\ tg\ T_A) + Artg\ ((k_y / k_x)\ tg\ t_B))\quad T_F'$

Italique

Application $T_F' = Artg(tg\ T_F / 1 + k\ tg\ T_F)$

Calcul $T_F'' = 1/2\ (Artg(tg\ T_A / 1 + k\ tg\ T_A) + Artg(tg\ T_B / 1 + k\ tg\ T_B)) \neq T_F'$

Rotation

Application $T_F' = T_F + B$

Calcul $T_F'' = 1/2\ (T_A + T_B) = T_F'$

Epaississement

Application $T_F' = T_F$

Calcul $T_F'' = 1/2\ (T_A + T_B) = T_F'$

Cette étude montre que la tangente du point intermédiaire n'est pas invariante, ni dans un cas ni dans l'autre, lors des transformations géométriques. Toutefois le point de carrure est le meilleur choix puisqu'il se produit une erreur uniquement lors d'un épaississement.

Cependant il faut remarquer qu'une erreur sur le calcul de la tangente en F n'altère pas la continuité de la courbe. D'après les résultats obtenus lors de la simulation, la distinction entre les déformations dues à la transformation et à l'erreur n'est pas facile à opérer.

II.-PROBLEMES LIES AUX TRANSFORMATIONS LOCALES

Nous allons analyser dans ce paragraphe les problèmes rencontrés lors de la simulation des transformations locales; nous parlerons successivement:

-de la mise à l'échelle

-de l'épaississement

-de la répercussion de ces transformations sur l'expansion des courbes évoluées.

1.-La mise à l'échelle

Les deux problèmes rencontrés lors de la réduction de la taille des caractères sont les déliés et l'arrondi des valeurs.

Le problème des déliés est dû à la quantification de l'épaisseur des traits. Il est possible de marquer ou de ne pas marquer les portions de contour qui ont une épaisseur inférieure au point de grille. Dans le premier cas l'épaisseur des déliés est artificiellement augmentée par rapport à celle des pleins. Dans le deuxième cas il se produit des coupures du caractère qui sont particulièrement inesthétiques. Pour illustrer ce problème nous avons paramétré ce choix dans le programme de simulation. (Fig 5) Il apparait préférable de marquer les traits d'épaisseur inférieure au point, ce qui correspond en outre au comportement de la maquette.

Le problème d'arrondi est lié à la conversion réel entier qui suit une mise à l'échelle. (Fig 6) Pour illustrer ce problème nous considérons un exemple où R(x) signifie arrondi de x.

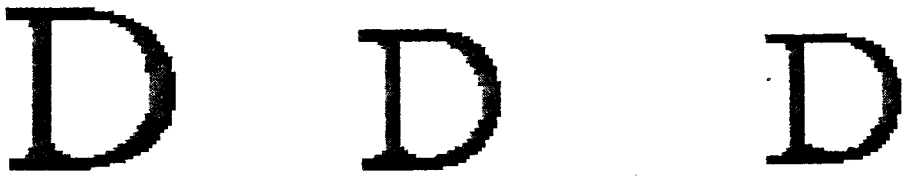
Echelle 2/35 Sortie échelle 4 & 1

Non marqués Traits inférieurs à 1 point Marqués

Fig 5

Echelle 1/8 10/95 1/10

Arrondi non corrigé



Echelle 1/8 10/95 1/10

Arrondi corrigé

Fig 6

Echelle 1/1 1/5 1/9 1/16

Fig 7

Soit x1 = 156 et x2= 159

avec une échelle 1/10 x1' = R(15.6) = 16

x2' = R(15.9) = 16

avec une échelle 1/15 x1' = R(10.4) = 10

x2' = R(10.6) = 11

Ainsi deux valeurs égales pour une certaine échelle peuvent devenir différentes pour une réduction plus importante. Nous allons maintenant déterminer le seuil au dessous duquel apparait ce phénomène. Soit $k, x, y \in \mathbb{N}$

$a \in \mathbb{R}$ et $0 < a < 1$

$f = R(ax) - R(ay)$

La fonction f n'est pas strictement décroissante. Pour corriger le défaut le plus gênant, une oscillation entre 0 et 1, il est nécessaire de forcer f à 0 dès que cette valeur a été atteinte une fois.

Pour que f = 0 il faut:

$k - 0.5 < ax < k + 0.5$

$k - 0.5 < ay < k + 0.5$

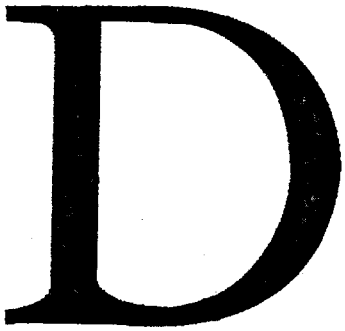
$-k - 0.5 \leqslant -ay \leqslant -k + 0.5$

$-1 < ax - ay < 1$ Soit $|a(x - y)| < 1$

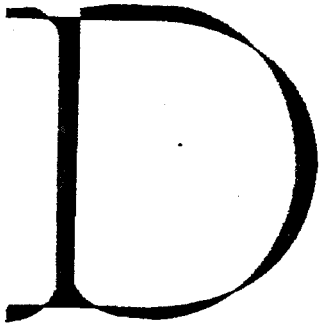
Le problème d'arrondi peut ainsi se produire dès que l'écart entre deux valeurs devient par mise à l'échelle inférieur à 1.

Théoriquement nous devons tester chaque coordonnée avec toutes les autres et forcer l'égalité lorsque l'écart devient inférieur à 1. En pratique pour réduire le temps de calcul nous limitons les tests à des valeurs consécutives. Ce contrôle nous permet d'éviter le cas le plus choquant où la pente d'une droite alterne entre 0 et 1.

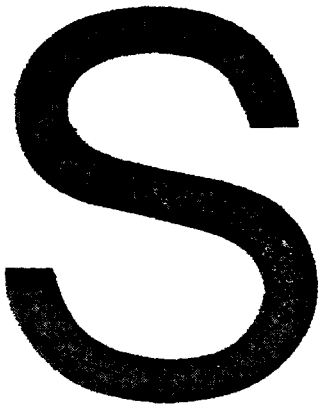
Il est à noter que dans les polices utilisées par les typographes il existe, du fait que les caractères de gros corps et de petits corps ne sont pas homothétiques, au moins trois formes différentes, chacune étant utilisée sur une plage d'échelle. Une solution plus simple serait de changer de forme chaque fois que le problème d'arrondi devient possible; l'obtention du code pour les petits corps pourrait être obtenue automatiquement à partir du code des gros corps en appliquant hors ligne la correction évoquée au-dessus.



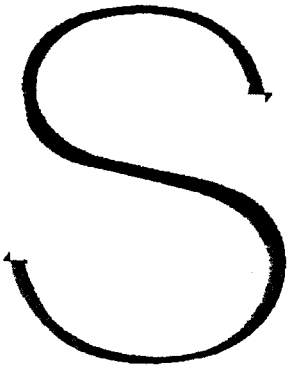
Original



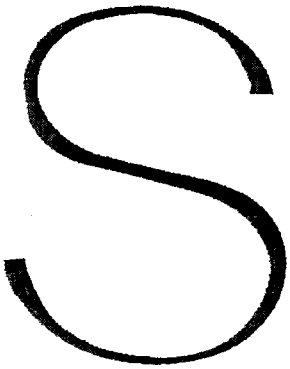
Epaississement -15



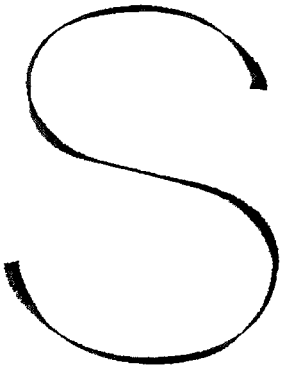
Original



Epaississement -20
non corrigé



Epaississement -20
corrigé



Epaississement -25
corrigé

Fig 8

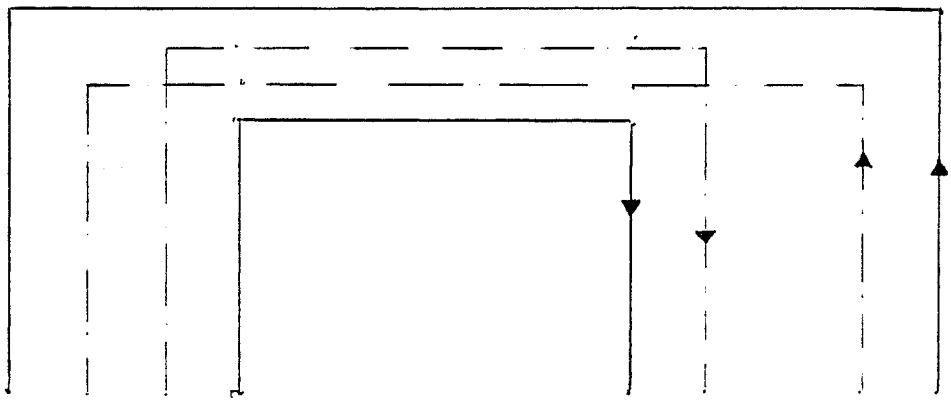


Fig 9

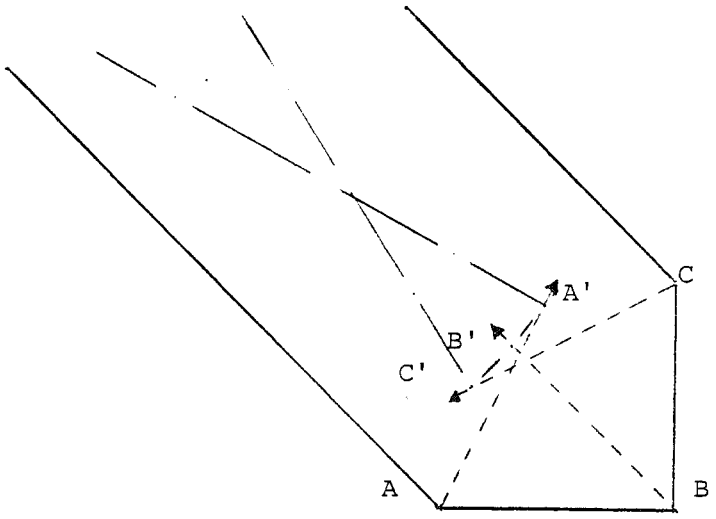


Fig 10

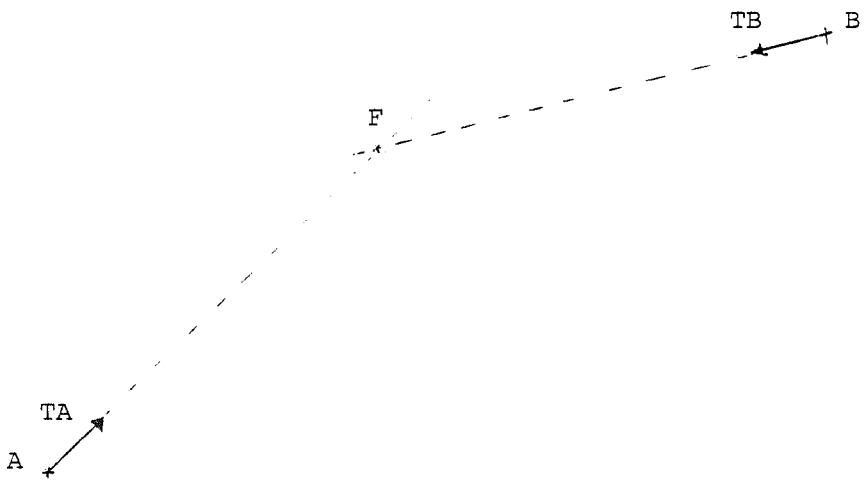


Fig 11

Pour illustrer cette idée de caractères spéciaux pour les petits corps, nous avons codé un caractère spécialement conçu pour les petites tailles (Fig 7).

Il est à noter également qu'une erreur d'un point est négligeable sur une photocomposeuse ayant une définition de 1200 points au pouce. Par contre il est nécessaire de corriger cette erreur sur des imprimantes ayant une définition de 200 points au pouce.

2.-L'épaississement

En fait ce problème apparait lors de l'amincissement du caractère; il s'agit du croisement des parois. (Fig 8)
Ces croisements de paroi sont interdits afin de faciliter le classement des ordonnées calculées: à chaque paroi est associé un numéro d'ordre qui permet le tri en temps réel et en cas de croisement de parois ces numéros deviennent erronés. De plus ce phénomène est inesthétique.

L'apparition de ce problème peut être due à la finesse d'un délié; en effet sur un délié d'épaisseur N un amincissement supérieur à N/2 conduit au croisement des deux bords (Fig 9). La détection de ce croisement impose une connaissance globale du contour puisqu'il faut contrôler l'épaisseur de chaque partie de la lettre. Cette opération est longue et complexe; de plus la démarche à adopter n'est pas simple. C'est pourquoi il nous semble préférable de fixer une limite d'amincissement au niveau de chaque police. L'éditeur de textes peut alors contrôler que la valeur d'amincissement est compatible avec la police utilisée et prévenir l'utilisateur dans le cas contraire.

L'autre cause d'apparition de cette erreur est un changement de direction par une suite d'interpolant de faible longueur. (Fig 10)
Dans ces conditions nous obtenons un changement de signe de la différence entre les abscisses et les ordonnées avant et après transformation. La détection est donc très simple. La correction consiste à rendre égaux les deux points en cause. De plus si le point supprimé appartient à une courbe évoluée, celle-ci est remplacée par des droites en forçant les tangentes des extrémités à passer par le point de carrure (Fig 11).

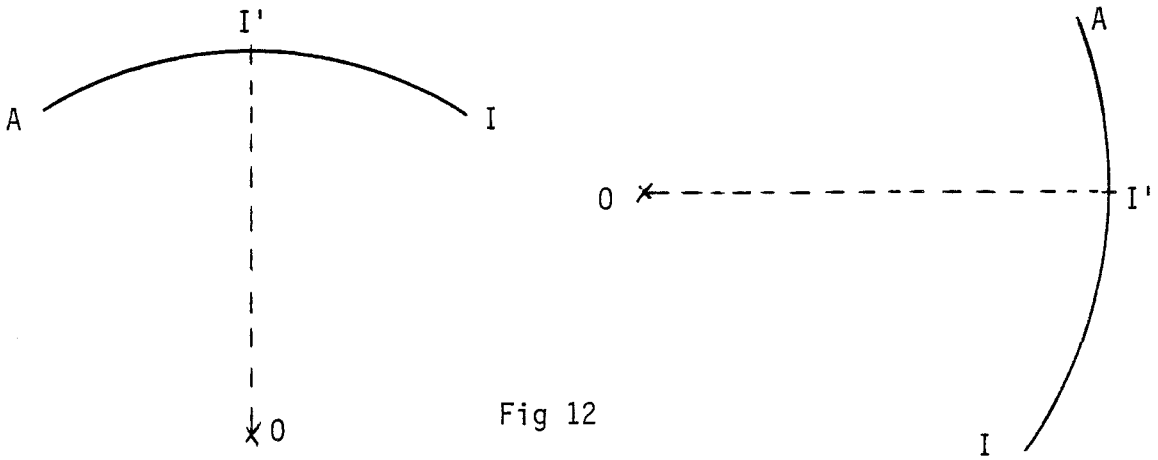


Fig 12

Fichier-courbes évoluées

Avant expansion	Aprés expansion	
	Sans	Avec
	Point supplémentaire	
XA	XA	XA
YA	YA	YA
TA	XO1	XO1
XF	YO1	YO1
YF	XI	XI'
TB	YI	YI'
XB	XO2	XO1
YB	YO2	YO1
	XF	XI
	YF	YI
	XO3	XO2
	YO3	YO2
	XJ	XF
	YJ	YF
	XO4	XO3
	YO4	YO3
	XB	XJ
	YB	YJ
		XO4
		YO4
		XB
		YB

Fig 13

3.-Répercussion sur l'étape d'expansion de courbes évoluées

L'algorithme de suivi des cercles impose pour être rapide de ne lui transmettre que des cercles appartenant à un seul quadrant; de cette façon les tests de changement de quadrant sont supprimés. De plus les points à tangentes verticales sont des points de naissance ou de mort qui doivent apparaitre dans la description du contour. Or au cours d'une rotation ou d'une mise en italique, les angles des tangentes sont modifiés. Un point à tangente verticale ou horizontale peut donc se trouver à l'intérieur d'une courbe évoluée (Fig 12).

Ces points supplémentaires doivent être calculés et insérés dans le fichier transmis à l'étape de suivi des contours.

Détection et calcul

La présence d'un point supplémentaire peut être simplement détectée en comparant les coordonnées du centre du cercle avec les coordonnées des extrémités de l'arc de cercle (Fig 12). Ainsi nous avons un point à tangente verticale si les ordonnées des extrémités sont de part et d'autre de l'ordonnée du centre et un point à tangente horizontale si les abscisses des extrémités sont de part et d'autre de l'abscisse du centre.

Le calcul des coordonnées de ce point est simple puisque lorsque nous calculons le centre du cercle nous avons un résultat intermédiaire égal au rayon.

Pour un point à tangente horizontale

$XIS = XO$

$YIS = YO \pm R$

Pour un point à tangente verticale

$XIS = XO \pm R$

$YIS = YO$

Insertion

Dans la maquette réalisée l'expansion des courbes évoluées est menée en parallèle avec l'exploration du fichier de façon à diminuer le temps de calcul. Ce parallélisme implique de déterminer avant le début du calcul la présence d'un point supplémentaire afin de prévoir un emplacement où le ranger (Fig 13).

Nous allons appliquer le raisonnement à la 1/2 courbe comprise entre A et F. Par raison de symétrie des calculs par rapport au point F ce même raisonnement pourra être étendu à la courbe entière.

La présence d'un point à tangente horizontale est due à un changement de signe de la tangente; celle d'un point à tangente verticale à une tangente de part et d'autre de $\pm \pi/2$.

Les angles des tangentes aux extrémités sont connus mais le calcul de la tangente en F nécessite une division; nous pouvons éviter cette division en remarquant que la présence d'un point supplémentaire est liée à un changement de quadrant des tangentes en A et F. Déterminer à quel quadrant appartient la tangente en A est une opération facile puisque nous connaissons l'angle. Pour l'angle en F il suffit de tester les signes des valeurs YB - YA et XB - XA.

Il est intéressant de ne pas prévoir de place supplémentaire dans le cas où le point supplémentaire est connu ou déjà calculé.

DA = 0	tangente	horizontale en A
DA = $\pm \pi/2$	"	verticale "
XA = XB	"	horizontale en F
YA = YB	"	verticale "
XA = XF	"	horizontale en I
YA = YF	"	verticale "

4.-Conclusion

Les transformations locales sont comparables en complexité à la mise à l'échelle; sauf en ce qui concerne l'épaississement qui demande le calcul d'une demi tangente à chaque point et nécessite des contrôles sur les résultats.

Le travail supplémentaire apporté à l'étape d'expansion des courbes évoluées se limite à quelques tests et additions. L'introduction des transformations géométriques ne ralentira pas la machine au niveau des transformations locales.

IV.-TRANSFORMATIONS GLOBALES

Les transformations locales qui modifient les coordonnées des points clés du contour peuvent rendre erronées les informations contenues dans la structure d'arbre. Nous appelons transformations globales les modifications de celles-ci car elles dépendent des positions des points clés du contour.

Nous avons deux types de transformations géométriques, celles qui modifient la structure d'arbre et celles qui la transforment complètement. La mise à l'échelle et l'épaississement appartiennent au premier type, la rotation et l'italique au second.

1.-Modifications

Pour la mise à l'échelle la seule modification consiste à mettre à l'échelle les abscisses de naissance. Cette opération très simple peut être considérée comme une transformation locale dans la structure d'arbre. C'est la solution que nous avons retenue pour la machine. Le programme analyse la structure d'arbre où il multiplie chaque abscisse de naissance par l'échelle en X et il passe ensuite à la modification de la structure d'anneau.

Pour l'épaississement nous avons le même problème de mettre à jour les abscisses de naissance. Cette opération est plus complexe que dans le cas de la mise à l'échelle car elle dépend de l'ordonnée et des deux demi-tangentes. Une solution possible est d'explorer la structure d'arbre après la structure d'anneau et de remplacer chaque abscisse de naissance par la valeur indiquée par le pointeur associé.

Lorsque plusieurs points de naissance ont la même abscisse de naissance nous ne dupliquons pas cette information et nous avons plusieurs pointeurs associés à celle-ci. Après épaississement les nouvelles abscisses peuvent ne plus être égales; ce qui impose de marquer les abscisses distinctes et entraine un décalage de la structure d'arbre avec une mise à jour de tous les pointeurs. Pour éviter ce bouleversement il est préférable de dupliquer préventivement les informations concernant les points de naissance multiples lors du codage du caractère.

2.-Transformations

Lors d'une rotation ou d'une mise en italique les points à tangentes verticales sont déplacés, les parois ne correspondent plus aux mêmes portions de contour, les pointeurs et les points de naissance sont modifiés. L'ordre des parois et le nombre de parois sont également variables. Toutes ces modifications impliquent de calculer entièrement la structure d'arbre.

Points particuliers

Ce sont les points de naissance et de mort qui forment l'information de base pour reconstruire la structure d'arbre. La solution la plus élémentaire consiste à explorer la structure d'anneau pour rechercher les points minima et maxima locaux en X et dresser une liste de ces points.

Au cours de l'étape d'expansion des courbes évoluées nous sommes obligés de détecter et d'inclure dans le fichier les points à tangentes verticales qui appartiennent aux coubes évoluées; si nous effectuons ces contrôles également pour les droites et si nous mémorisons ces informations, l'exploration du fichier est ramenée à une analyse autour des points mémorisés.

La liste des points de naissance nous permet d'inscrire dans la structure d'arbre les abscisses de naissance et les pointeurs associés; il nous reste à calculer les numéros de paroi et d'ordre.

Numéro de paroi

Ce numéro est utilisé pour réaliser une allocation rapide et équilibrée des parois sur un nombre variable de processeurs en parallèle. (BLO 81)

Lors du codage du caractère il est possible de rechercher une solution optimale puisque ce calcul est réalisé hors ligne et une seule fois par forme. Lors du calcul en ligne l'optimisation conduirait à une perte de temps; en effet c'est une opération qui demande de nombreux calculs et dont le temps de réalisation serait supérieur au temps de suivi des parois pour une solution non optimale. C'est pourquoi nous retenons la solution la plus simple qui consiste à affecter les numéros de paroi dans l'ordre croissant au fur et à mesure des naissances et à réutiliser un numéro dès qu'il est libéré par une mort.

Numéro d'ordre

Il est utilisé pour classer les ordonnées calculées avant de les transmettre à l'organe de visualisation. Nous considérons que deux parois ne peuvent se croiser ce qui simplifie le classement puisque le résultat d'un test à une abscisse est vrai sur toute la longueur commune aux deux parois; cette remarque et le fait que les numéros d'ordre ne sont pas obligatoirement contigus expliquent pourquoi il est possible d'attribuer un numéro d'ordre unique par paroi.

Au moment du calcul de la structure d'arbre il nous faut classer les parois en fonction des ordonnées, en disposant de la liste des points de naissance et de pointeurs vers la structure d'anneau qui sont déjà dans la structure d'arbre.

Nous considérons à un instant donné que nous avons la liste des parois actives classée; les emplacements dans cette liste constituent des numéros d'ordre provisoires (Fig 14). Ils deviennent définitifs, c'est à dire inscrits dans la structure d'arbre, et libres, c'est à dire réutilisables, à la mort des parois.

Liste des parois actives

Ai pointeur vers structure d' arbre, Ai = 0 place libre

Ti pointeur vers 1' arc courant

6	A1	T1			0				0		
5	A2	T2	Mort de		A2	T2	Naissance	si Y7 > Y5	A2	T2
4	A4	T4	2-6		A4	T4	de 7	Y7 < Y3	A4	T4
3	A6	T6			A3	T3			A3	T3
2	A3	T3			0				A7	T7
1	A5	T5			A5	T5			A5	T5
6									A2	T2
5								si Y7 > Y4	A7	T7
4								Y7 < Y2	A4	T4
3									A3	T3
2									0	
1									A5	T5

Fig 14

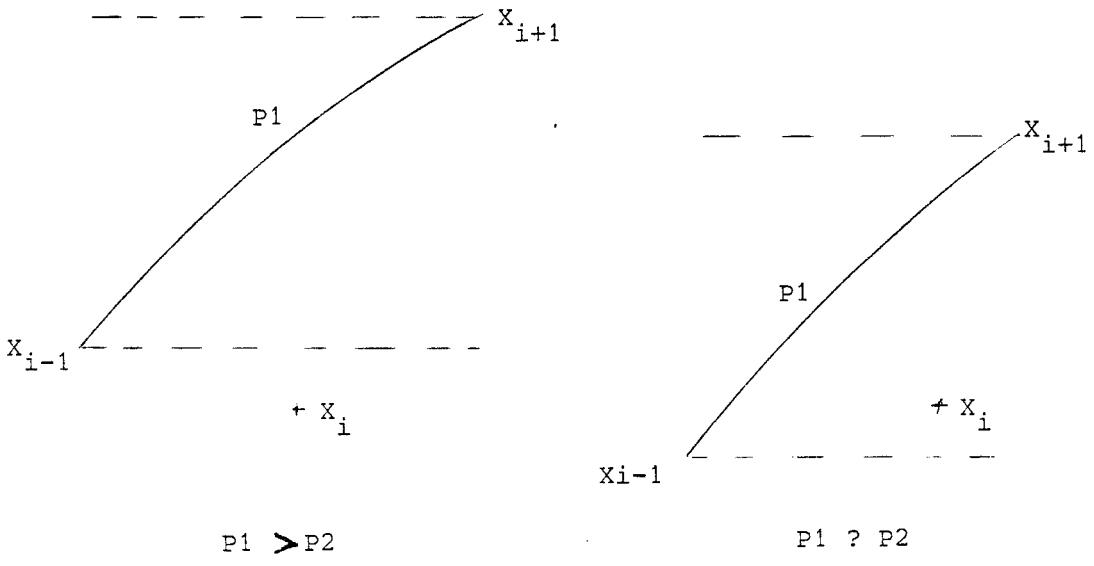


Fig 15

Lors d'un point de naissance il suffit d'interclasser les nouvelles parois (au moins deux) dans la liste des parois actives. Pour initialiser la table il suffit de classer entre elles les parois activées à la première abscisse de naissance.

L'interclassement de nouvelles parois demande à première vue de connaitre toutes les ordonnées à l'abscisse de naissance courante. Suivre toutes les parois abscisse par abscisse est une opération longue et inutile lorsqu'un interpolant ne chevauche pas une abscisse de naissance. Nous pouvons remplacer les calculs de suivi de paroi par un parcours de la structure d'anneau jusqu'à la détection d'un arc chevauchant le nouveau point de naissance. Pour cela nous devons associer à chaque paroi active un pointeur vers l'arc courant. Il est également possible de simplifier l'interclassement en effectuant un test point contre rectangle (Fig 15); c'est à dire tester si l'arc courant est entièrement au-dessus ou au-dessous de l'ordonnée de naissance. Ce cas très fréquent permet de réduire le temps de calcul: le suivi d'un arc point à point n'est utilisé qu'après l'échec de ce test.

Les algorithmes incrémentaux de suivi des droites et des cercles peuvent également être améliorés en tenant compte du fait que nous ne désirons connaitre qu'une seule ordonnée et non toutes les ordonnées.

Permutation des fichiers

Pour simplifier le parcours de la structure d'anneau au cours de l'étape de suivi des parois nous débutons la description d'un contour par un point de mort; dans ces conditions le dernier point d'un contour est également un point de mort et les deux points extrêmes du contour sont des points d'arrêt de parcours de l'anneau. Les pointeurs, qui ferment l'anneau de façon à le rendre utilisable dans une mémoire linéaire, ne sont plus utiles ni les tests de détection de ceux-ci.

Cette contrainte impose donc de permuter les "anneaux" de description des contours et bien entendu de modifier les pointeurs associés aux points de naissance.

3.-Conclusion

Les transformations globales sont plus longues que les transformations locales et réduiront le débit de la machine. Par rapport à l'étude de M. BLOCH sur le calcul de la structure d'arbre, la recherche des numéros de parois est ramenée à la solution de base, et la recherche des numéros d'ordre est ramenée à une étude locale autour des points de naissance au lieu d'une étude globale au niveau des parois.

V.-INDICATION SUR LA REALISATION

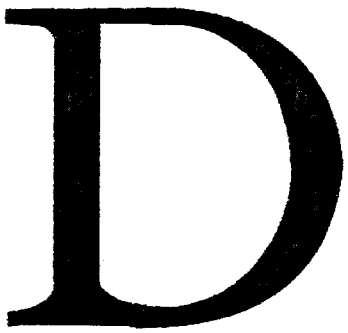
1.-Simulation

cette simulation écrite en FORTRAN IV sur un PDP 11/40 nous a permis d'illustrer ce chapitre. Les fichiers de description des caractères sont très proches de ceux utilisés par le prototype. La différence principale vient du fait que le caractère n'est pas engendré en une passe, ce qui rend inutile la struture d'arbre.

Nous n'avons simulé que les transformations locales, ce qui nous a permis d'illustrer les problèmes liés à celles-ci. La principale difficulté liée aux transformations globales est l'optimisation du temps de calcul et la simulation de celles-ci ne présentait pas d'intérêt.

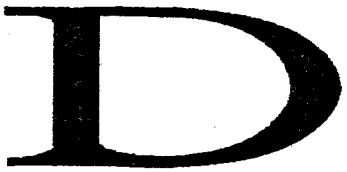
2.-Combinaison des transformations

La mise à l'échelle est une opération qui doit être réalisée avec toute autre transformation géométrique; les autres combinaisons sont également utiles pour la réalisation de plans ou cartes par exemple. La figure 16 montre l'effet produit par quelques unes de celles-ci et permet de vérifier que l'ordre des opérations n'est pas commutatif.

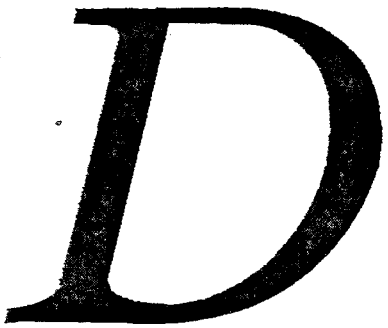


Original

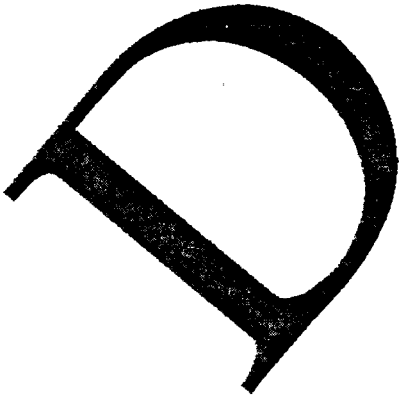
Echelle 1/2 1/1



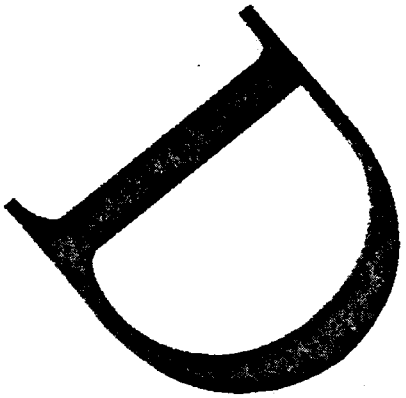
Echelle 1/1 1/2



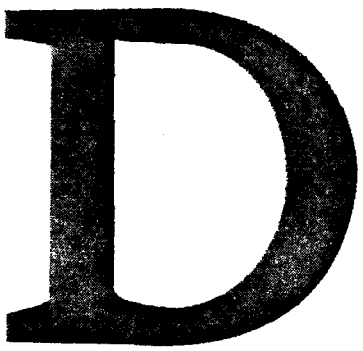
Italique



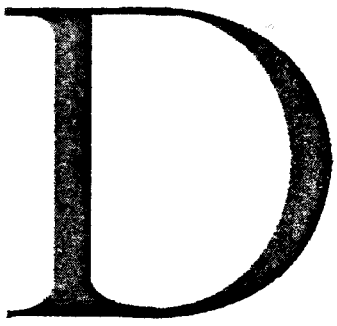
Rotation 50°



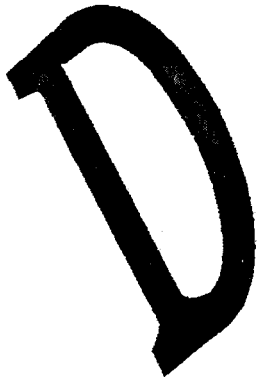
Rotation -50°



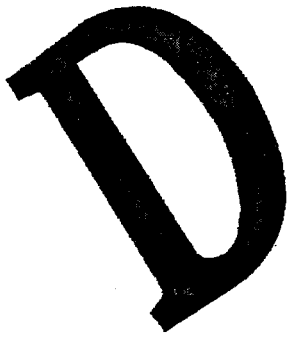
Epaississement 10



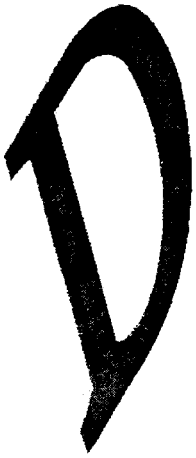
Epaississement -5



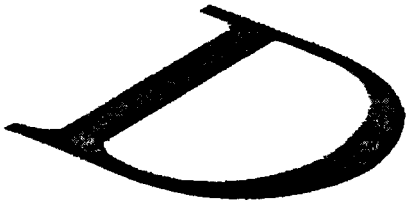
Epaississement 10 Echelle 1/2 1/1
Italique Rotation 40°



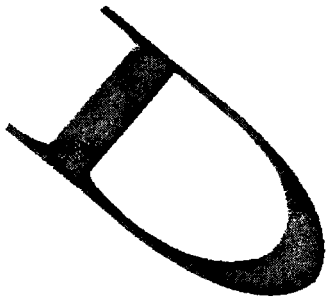
Echelle 1/2 1/1 Rotation 40°
Epaississement 10 Italique



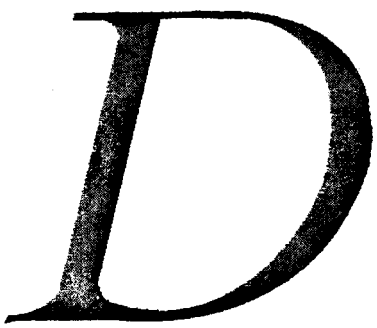
Italique Epaississement 10
Rotation 40° Echelle 1/2 1/1



Rotation -40°
Echelle 1/1 1/2



Echelle 1/1 1/2
Rotation -40°



Epaississement -5
Italique

Fig 16

Pour les transformations locales il suffit d'appliquer successivement les opérations propres à chaque opération.

Pour les transformations globales il nous faut appliquer la plus importante, soit recalculer la structure d'arbre après une rotation ou une italique; la correction des abscisses de naissance pour une mise à l'échelle ou un épaississement peut être réalisée durant les transformations locales.

En ce qui concerne l'ordre d'application des opérations il nous semble préférable de prendre celui dont les résultats paraissent logiques à l'utilisateur soit:

-L'épaississement

-La mise à l'échelle

-L'italique

-La rotation

Il est à noter également que la mise à l'échelle doit permettre d'obtenir un caractère d'une certaine taille et elle doit donc être corrigée en fonction de l'épaississement demandé. Cette correction peut facilement être calculée par l'éditeur de texte lors de la saisie des paramètres des caractères (type de police, échelle, épaississement, italique, rotation,...). Soit N la taille de la grille dans laquelle le caractère a été défini, EX l'échelle en X, EY l'échelle en Y et EP l'épaississement; nous obtenons:

$$EX' = EX\ N\ /\ (N + 2\ EP)$$

$$EY' = EY\ N\ /\ (N + 2\ EP)$$

3.-Insertion dans le pipe-line

La structure de la machine nous permet d'insérer facilement des cartes supplémentaires pour les tranformations géométriques (Fig 17). Les transformations locales sont réalisées sur les paramètres des droites et des courbes évoluées et les cartes correspondantes doivent être placées avant l'étape d'expansion des courbes évoluées; l'ordre entre les cartes est fixé par l'ordre d'application désiré. Les transformations globales nécessitent de connaitre tous les points à tangente verticale, elles doivent se situer après l'étape d'expansion des courbes évoluées.

SYNOPTIQUE DE LA MACHINE

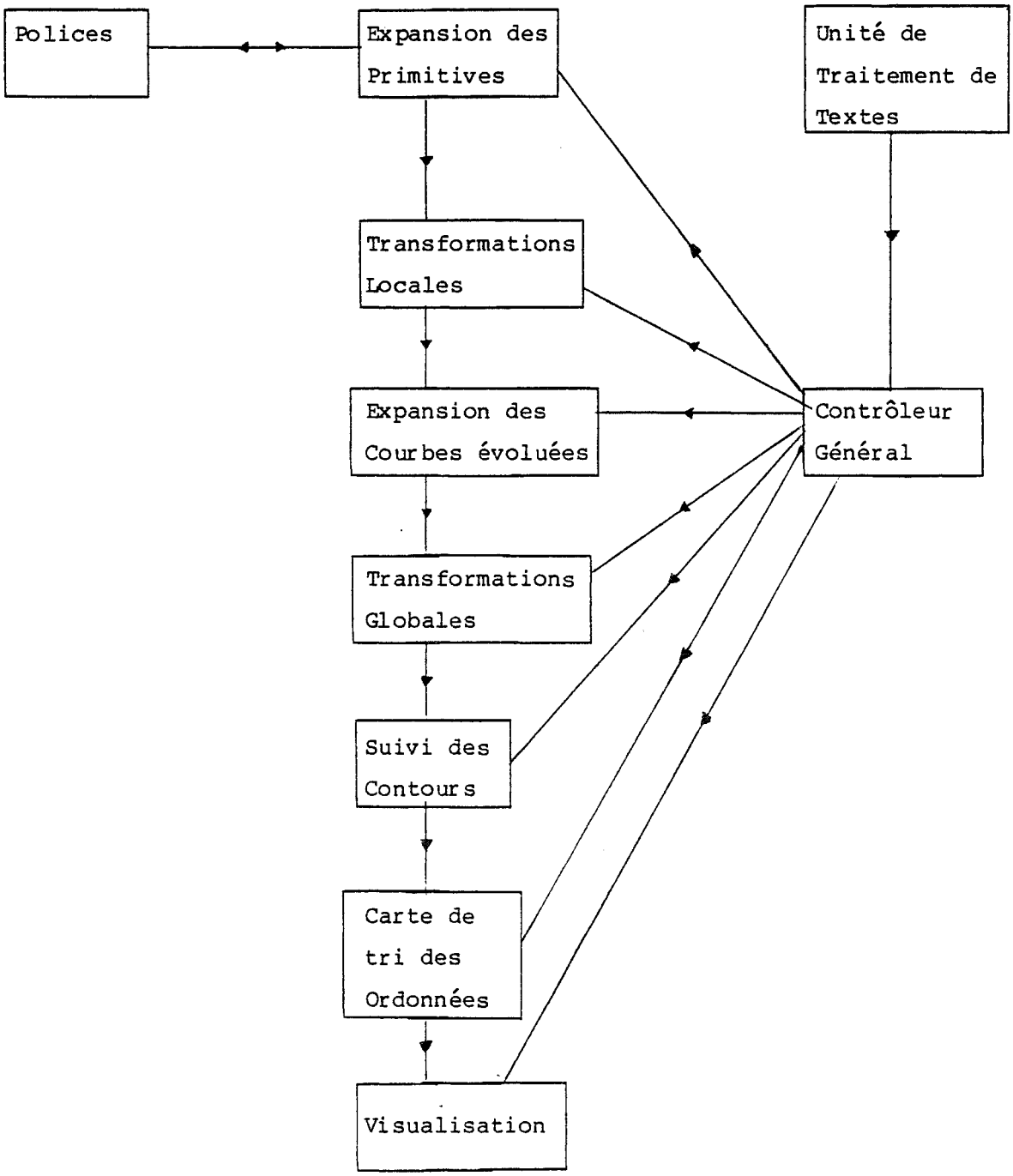


Fig 17

De plus les programmes d'expansion des courbes évoluées doivent être modifiés de façon à calculer et insérer les points supplémentaires.

4.-Mise à l'échelle

C'est la seule transformation géométrique prévue dans le prototype que nous avons construit. Nous avons utilisé les microprocesseurs en tranches de la famille 2900 et la technique de microprogrammation car le programme est très simple, (il contient 64 microinstructions), et nous avons construit un chemin de données adapté aux tranferts des informations.

Afin de conserver une bonne définition du caractère la mise à l'échelle se fait par réduction d'une forme définie dans la grille maximale. Pour respecter le débit de la machine nous avons remplacé les divisons par des multiplications suivies de décalages. La mise à l'échelle des coordonnées est effectuée sur un multiplieur rapide 16X16 bits ayant un temps de traversée de 120 ns. (TDC 10-10 J de TRW)

La possibilité d'avoir des échelles différentes en abscisse et en ordonnée nous impose de modifier les angles. Cette opération complexe est réalisée par un accès à une table; ce qui limite le nombre de rapports entre les échelles à trente deux valeurs comprises entre 1/2 et 2.

5.-Italique

Les opérations sur les coordonnées ont la même complexité que pour la mise à l'échelle; le multiplieur utilisé possède un accumulateur et peut additionner des résultats intermédiaires. La modification des angles est une opération complexe qui doit également être réalisée par un accès à une table. Une carte identique, aux tables et au programme près, à celle de la mise à l'échelle peut être utilisée pour cette transformation.

6.-Rotation

Pour cette transformation nous devons calculer des cosinus et sinus et pour cela nous utilisons des tables trigonométriques.
La taille de ces tables dépend du pas entre les valeurs de rotation; dans la simulation nous avons pris un pas de 1°; ce qui donne une taille de 360 mots par table.

Les opérations sur les coordonnées sont plus complexes que pour les transformations précédentes mais elles sont encore réalisables sur le multiplieur utilisé.

Les calculs sur les angles sont réduits à de simples additions exécutables sur le multiplieur.

Une carte très proche de celle utilisée pour la mise à l'échelle peut également être utilisée pour cette transformation.

7.-Epaississement

C'est la transformation la plus complexe puisque pour chaque point nous devons connaître les demi-tangentes et donc effectuer une division et un calcul d'arctangente. En plus de la table d'arctangente nous utilisons celles des cosinus et sinus.

Pour pouvoir calculer le premier point d'un contour nous devons rechercher le dernier point afin de lire dans le cas d'une courbe évoluée ou calculer dans le cas d'une droite l'une des demi tangentes.

Pour cette transformation il est nécessaire de concevoir une nouvelle carte capable d'effectuer des divisions.

8.-Transformations globales

Nous avons vu que cette étape n'est nécessaire qu'après une rotation ou une mise en italique. Afin de ne pas ralentir la machine dans les autres cas cette carte doit posséder un mode transparent. La façon la plus simple d'assurer cette transparence est d'utiliser un programme qui recopie le fichier d'entrée en sortie.

Nous avons indiqué dans le paragraphe sur les transformations globales le principe de calcul de la structure d'arbre. Les algorithmes effectuent de nombreux tests et manipulations de données; un microprocesseur du type 8086 d'INTEL nous semble bien adapté à ce genre de travail.

Une estimation du temps de calcul de cette étape nous a donné un temps d'environ 5 ms; ce qui correspond à un ralentissement par cinq de la machine.

9.-Regroupement

Nous avons vu que les transformations locales de la mise à l'échelle, l'italique et la rotation peuvent être réalisées avec de très faibles modifications sur la même carte. Il est également possible de concevoir une carte capable de supporter toutes les transformations locales. L'intérêt de ce regroupement est bien entendu de diminuer le coût de la machine. Le ralentissement de cette étape à cause du regroupement n'est pas gênant tant que nous n'excédons pas celui de la carte des transformations globales; nous pouvons estimer grossièrement à quatre le ralentissement dû au regroupement et donc considérer celui-ci tout à fait acceptable.

Chapitre 6

CHAPITRE 6

VARIANTES D'ALGORITHMES

I.-ACCELERATION DU TRACE DE CERCLES

II.-DECOMPOSITION DE COURBES EVOLUEES EN ELLIPSES

VARIANTES D'ALGORITHMES

I.-ACCELERATION DU TRACE DE CERCLES

Dans l'étape de suivi de contours, nous utilisons pour tracer les cercles un algorithme incrémental (HOR 77, BRE 77); à chaque point nous remplaçons le cercle par une droite dont les paramètres directeurs sont donnés par les dérivées partielles de la fonction implicite. Les dérivées secondes partielles étant égales à deux, les paramètres directeurs varient de deux à chaque variation d'un point sur la grille de tracé.

L'amélioration proposée consiste à remplacer l'addition de deux par une incrémentation de un, qui est une opération plus rapide sur le 8086 d'INTEL.

A chaque point l'algorithme essaye d'annuler la variation de la fonction implicite. Pour cela nous pouvons considérer le cumul des paramètres, respectivement ERX et ERY et en déduire l'ordonnée du dernier point calculé.

Soit ΔX l'écart en X, ΔY l'écart en Y; A_n, B_n les paramètres directeurs de la tangente; s signe de YB - YA.
Les conditions initiales s'écrivent:

$$A_0 = 2(XA - XC) + 1$$
$$B_0 = 2s(YA - YC) + 1$$

et la formule de récurence:

$$A_n = A_{n-1} + 2$$
$$B_n = B_{n-1} + 2$$

Nous avons donc:

$$ERX = \sum_{n=0}^{\Delta X-1} A_n = (A_0 - 1)\,\Delta X + \Delta X^2$$

$$ERY = \sum_{n=0}^{\Delta Y-1} B_n = (B_0 - 1)\,\Delta Y + \Delta Y^2$$

Sachant que ERX + ERY = 0; pour tout point $(X_A+\Delta X\ ,\ Y_A+\Delta Y)$ situé sur le cercle.

Nous en déduisons pour un tel point:

$$Y = -1/2\ (B_0-1) \pm (B_0-1)^2-4\ X(A_0-1)-4\ X^2)^{1/2}$$

Nous pouvons calculer la dernière ordonnée dans le cas le plus critique où l'arc de cercle se termine par une tangente verticale.

Nous avons pour un quart de cercle :

$$A_0 = 1$$
$$B_0 = -\ 2R + 1$$
$$X = R$$

La formule ci-dessus nous permet de trouver :

$$Y = R$$

Dans un premier temps nous pouvons simplement diviser par deux les calculs d'initialisation et l'incrément; nous obtenons :

$$A_0' = XA - XC$$
$$B_0' = s(YA - YC)$$
$$A_n' = A_{n-1}' + 1$$
$$B_n' = B_{n-1}' + 1$$
$$ERX' = (A_0' - 1/2) \sum_{n=0}^{\Delta X-1} A_n' = \Delta X + X^2/2$$
$$ERY' = (B_0' - 1/2) \sum_{n=0}^{\Delta Y-1} B_n' = \Delta Y + Y^2/2$$

Un point situé sur le cercle est encore supposé satisfaire à ERX' + ERY' = 0 et nous en déduisons:

$$Y = -1/2\ (2B_0'-1) \pm (2B_0'-1)^2-4\ X(2A_0'-1)-4\ X^2\ 1/2$$

Si nous considérons le même quart de cercle.

$$A_0' = 0$$
$$B_0' = -R$$
$$X = R$$

Nous obtenons: $Y = R + 1/2 \pm 1/2(8R + 1)^{1/2}$

Ce calcul conduit donc à une erreur cumulée importante. Si nous voulons un calcul correct, nous devons obtenir Y = R, ce qui suppose X = R + 1. Il suffit donc d'itérer l'algorithme un coup supplémentaire. Nous avons utilisé cette variante pour le décodeur simulé sur le PDP 11/40. Bien entendu nous n'avons obtenu aucun gain de temps mais nous avons pu ainsi contrôler la justesse de ce raisonnement. La simulation utilise un biais de façon à ramener l'erreur à ± 1/2 au lieu de ± 1; pour cela la fonction implicite n'est pas initialisée à $A_0 + B_0$ mais à $1/2\ (A_0 + B_0)$.

GAIN DE TEMPS SUR 8086

Boucle de calcul

	Classique			Variante
	ADD A_n,2	4	2	INC A_n
	ADD PHI,A_n	3	3	ADD PHI,A_n
	JLE FC	8/4	8/4	JLE FC
LC:	DEC X	2	2	DEC X
	ADD B_n,2	4	2	INC B_n
	ADD PHI,B_n	4	4	ADD PHI,B_n
	JG LC	8/4	8/4	JG LC
FC:	RTN	8	8	RTN
		---	---	
Boucle interne		17	15	cycles
Un parcours simple		32	28	cycles

Initialisation

	Classique			Variante
	SUB XA,XC	3	3	SUB XA,XC
	ADD XA,XA	3	3	SUB YA,YC
	INC(DEC) XA	2	2	MOV PHI,XA
	SUB YA,YC	3	3	ADD PHI,YA
	ADD YA,YA	3		
	INC(DEC) YA	2		
	SUB PHI,PHI	3		
		-	-	
		19	11	cycles

Le gain de temps est illustré dans le tableau de la page 6.3 où nous avons repris les algorithmes programmés dans l'étape de suivi de contours. Nous appelons PHI la valeur de la fonction implicite égale à ERX + ERY. Nous utilisons 15 cycles d'horloge au lieu de 17 dans la boucle interne de calcul de l'ordonnée et 28 cycles au lieu de 32 pour un parcours simple, ce qui donne un gain d'environ 11%. Nous pouvons exprimer d'une autre façon le gain en remarquant que nous faisons varier les valeurs des dérivées partielles $\Delta X + \Delta Y$ fois. Nous gagnons 2 cycles par incrémentation:

$\Delta T(\text{cycles}) = 2\ (\Delta X + \Delta Y)$

L'initialisation est elle aussi plus rapide mais ceci est moins important car moins fréquent.

II.-DECOMPOSITION DE COURBES EVOLUEES EN ELLIPSES

La décomposition des courbes évoluées en arcs de cercles conduit à de nombreux calculs. Nous pouvons rapprocher le tracé de deux cercles par demi-courbe évoluée d'un tracé d'ellipse en dessin industriel. De façon à ne pas ralentir le débit de l'étape de suivi de contours nous devons utiliser un algorithme incémental simple; pour cela nous considérons des ellipses avec des axes parallèles aux axes X/Y, ce qui conduit à des dérivées secondes simples. Soit la fonction implicite :

$$f(X,Y) = aX^2 + bY^2 + cX + dY + e$$

Nous obtenons les dérivées partielles :

$$df/dX = 2aX + c$$

$$df/dY = 2bY + d$$

$$d^2f/dX^2 = 2a$$

$$d^2f/dY^2 = 2b$$

Nous voulons tracer une ellipse entre les points A et F en respectant les tangentes en A et F. pour cela nous posons :

[1] $df/dXA = -k1 \sin TA = 2aXA + c$

$df/dYA = k1 \cos TA = 2bYA + d$

$df/dXF = -k2 \sin TF = 2aXF + c$

$df/dYF = k2 \cos TF = 2bYF + d$

$\Delta X = XF - XA$

$\Delta Y = YF - YA$

Nous en déduisons :

[2] $d^2f/dX^2 = (k1 \sin TA - k2 \sin TF)/\Delta X = 2a$

$d^2f/dY^2 = (k2 \cos TF - k1 \cos TA)/\Delta Y = 2b$

Nous voulons également que l'algorithme incrémental calcule correctement la dernière ordonnée. Soit :

$A_0 = -k1 \sin TA$

$B_0 = k1 \cos TA$

$A_n = A_{n-1} + 2a$

$B_n = B_{n-1} + 2b$

$ERX = \Delta X (A_0 - 2a) + 2a \Delta X^2$

$ERY = \Delta Y (B_0 - 2b) + 2b \Delta Y^2$

Nous posons :

$ERX + ERY = 0$; d'où;

$k2 (\sin TF (1 - \Delta X) + \cos TF (1 - \Delta Y) = k1 (\sin TA - \cos TA)$

Il existe une infinité de solutions; nous considérons la plus simple:

$k2 = \sin TA - \cos TA$

$k1 = \sin TF (1 - \Delta X) + \cos TF (1 - \Delta Y)$

Une fois k1 et k2 déterminés nous calculons les dérivées premières A_0, B_0, ainsi que les dérivées secondes 2a et 2b. Nous arrivons à un total de 12X, 5/, 14+ et 7 recherches en table; soit un temps de calcul de 37,4 µs au lieu de 74 µs. Nous obtenons un gain de près de 50% au niveau de l'expansion des courbes évoluées. Nous effectuons également deux fois moins d'initialisation d'arc au niveau du tracé de contours. En contrepartie les dérivées secondes ne sont plus des constantes et elles font partie du contexte de tracé d'une paroi.

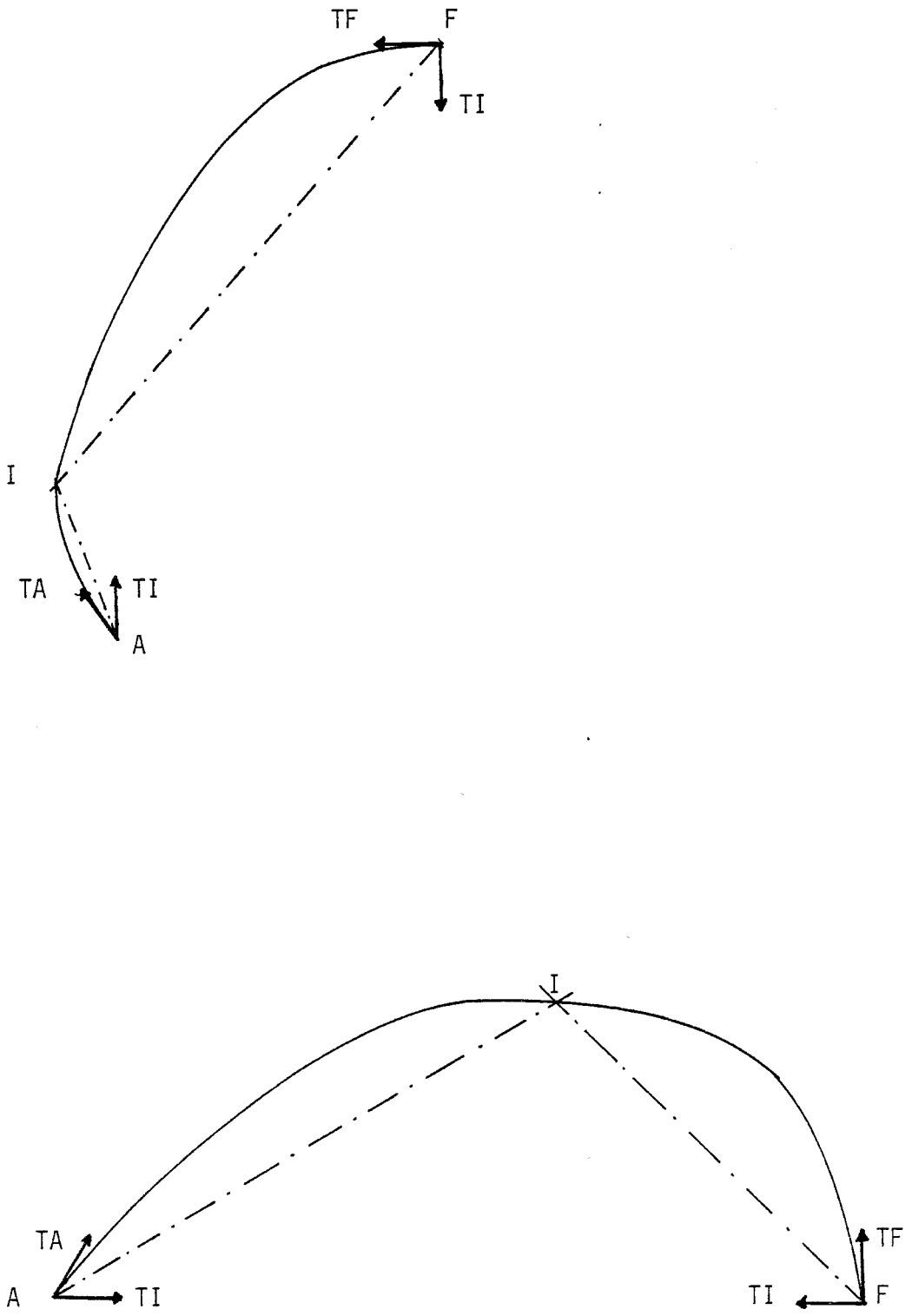


Fig 1

Lors d'une rotation ou d'une italique la détermination des points supplémentaires à tangente verticale ou horizontale est plus complexe et ne peut comme dans le cas des cercles s'appuyer sur le calcul des centres. Une solution possible consiste à appliquer l'algorithme de recherche des points intermédiaires utilisé au chapitre 2. Le point supplémentaire a une tangente verticale ou horizontale TI; le calcul des coefficients des ellipses nous impose de calculer les tangentes en A et F. Le point recherché est à l'intersection de la bissectrice en A par rapport à TA et TI et de la bissectrice en F par rapport à TF et TI. (Fig 1)

annexes

SOMMAIRE

UTILSATION DU PROGRAMME

Généralités

Le programme comporte quatre modules:

-Saisie, modification, mémorisation et rappel d'images bicolores;

-Codage par contour de la forme;

-Décodage de contour;

-Modifications géométriques (mise à l'échelle, Italique, Rotation et Epaississement).

Le programme principal permet de sélectionner l'opération désirée et de saisir les paramètres correspondants. Le programme s'appelle "CODAGE"; lors du lancement il demande:

FICHIER OU CONSOLE

L'utilisateur doit préciser la console où il se trouve ou le nom d'un fichier de commande.

Menu

Le programme affiche:

0 : HELP**

0 : Affichage d'un menu réduit.

1 : Arrêt du programme.

2 : Saisie d'une image bicolore par caméra.
Résultat en vidéo-ram.

3 : Modification d'une image bicolore en vidéo-ram.

Le programme trace un réticule en X = 255 et Y = 255 et affiche:

0 : HELP X/Y:

VAL : -128 -128 = blanc 0 = noir.

X1 : 255 Y1 : 255 Position début droite ou rectangle.

X : 255 Y : 255 " fin " "

0 : Affichage d'un menu de modification réduit.

1 : Retour au programme principal.

2 : Déplacement du réticule suivant X.

0 : Retour au programme modification.

A : Entier = valeur de déplacement.

Si A + X sup 512 alors X = 512

Si A + X inf 0 alors X = 0

3 : Déplacement du réticule suivant Y.

0 : Retour au programme modification.

A : Entier = valeur de déplacement.

Si A + Y sup 512 alors Y = 512

Si A + Y inf 0 alors Y = 0

4 : Met à noir le point indiqué par le réticule.

Début d'une droite noire ou point de la diagonale principale d'un rectangle noir.

5 : Met à blanc le point indiqué par le réticule.

Début d'une droite blanche ou point de la diagonale principale d'un rectangle blanc.

6 : Fin d'une droite, la couleur a été précisée par la commande 4 ou 5.

7 : Second point de la diagonale principale d'un rectangle, la couleur a été précisée par la commande 4 ou 5.

8 : Efface le bord bas et gauche. Met à blanc à gauche et au dessous du réticule.

9 : Efface le bord haut et droit. Met à blanc à droite et au dessus du réticule.

10 : Loupe logicielle centrée sur le réticule.

LOUPE

1 : Suppresion de la loupe.

2 : Grossissement de 2 sur une grille 64X64.

4 : Grossissement de 4 sur une grille 64X64.

4 : Rappel d'une image codée par plage mémorisée sur disque par la commande 5.

NOM DU FICHIER :

5 : Codage par plage de la vidéo-ram et mémorisation sur disque.

NOM DU FICHIER :

6 : Sortie de la vidéo-ram sur l'imprimante électrostatique (GOULD) à une échelle 4; visualisation des trois bits de poids forts.

7 : Codage par contour de la forme et mémorisation sur un fichier.

NOM DU FICHIER CONTOUR :

8 : Sortie de la vidéo-ram sur l'imprimante électrostatique (GOULD) à une échelle 1; visualisation du bit de poids fort uniquement.

9 : Visualisation en vidéo-ram du dernier caractère manipulé, codé ou modifié par une transformation géométrique.

10 : Mise à l'échelle.

NOM DU FICHIER :

ECHELLE NX,DX,NY,DY

NX Entier = numérateur échelle sur les abscisses.

DX " = dénominateur " "

NY " = numérateur échelle sur les ordonnées.

DY " = dénominateur " "

11 : Mise en italique (Inclinaison 13°).

NOM DU FICHIER :

12 : Rotation.

NOM DU FICHIER :

ANGLE DE ROTATION

A Entier = angle de rotation en degré.

13 : Epaississement.

NOM DU FICHIER :

EPAISSISSEMENT

E Entier = épaississement en points de grille.

14 : Déplacement d'un caractère codé par contour.

NOM DU FICHIER :

DECALAGE X,Y

I Entier = déplacement en X.

J Entier = déplacement en Y.

15 : Modification des paramètres (par défaut).

CERCLE INUM IDEN 2 ENTIERS

I (1) Entier = numérateur du biais de tracé des cercles.

J (2) Entier = dénominateur " "

CORRECTION F 0:SANS 1:SIGNE -1:NON SIGNE

I (0) Entier.

LIMITE DE TRACE DES CERCLES

I (0) Entier = la taille minimale des cercles à tracer.

*** TRACE (0 : SANS) ***

I (0) Entier utilisé pour indiquer le mode de trace lors de la mise au point du programme.

0 : Pas de sortie.

2 : Sortie des sous parois et essai d'interpolants.

3 : Sortie des suivis des contours point à point + 2.

PRECISION ANGLE

I (1) Entier = l'écart minimal entre un point F et la corde AB.

PRECISION DROITE

I (1) Entier = la tolérance lors du filtrage des droites.

PRECISION C.E.

I (1) Entier = la tolérance sur les courbes évoluées.

I = 0 codage par droites uniquement.

VALEUR SEUIL K*DX REAL

X (1.) Réel = le seuil de correction aprés arrondi lors de la mise à l'échelle.

TRAITS(1 MARQUES:-64 NON:0

I (-64) Entier indique si on marque ou pas les traits fins.

COR CROIS EPAIS:2 NON:4

I (2) Entier indique si on corrige les croisements lors de l'épaississement.

16 : Permet de préciser un fichier de commande ou la console.

FICHIER OU CONSOLE

17 : Visualisation en vidéo-ram d'un caractère codé par contour.

NOM DU FICHIER CONTOUR

18 : Filtrage point, permet d'éliminer les points isolés ou les traits d'épaisseur inférieur au N points de grille.

PRECISION

N Entier = la taille minimale des traits.

19 : Comparaison d'une image en vidéo-ram et d'un fichier codé par plage mémorisé sur disque (OU exclusif).

NOM DU FICHIER :

20 : Déplacement d'une image codée par plage.

NOM DU FICHIER :

POSITION X,Y

I Entier = déplacement en X.

J Entier = déplacement en Y.

21 : Filtrage ligne, permet de rectifier les portions horizontales et verticales dans le cas d'un bruit inférieur à un point.

22 : Comparaison d'une image en vidéo-ram et d'un fichier codé par plage mémorisé sur disque (Original - codé).

NOM DU FICHIER :

23 : Comparaison d'une image en vidéo-ram et d'une image codée par plage mémorisée sur disque (Codé - original).

NOM DU FICHIER :

24 : Superpose à l'image sur la vidéo-ram une grille qui facilite la mesure des coordonnées; les traits sont obtenus par complémentation de la couleur, ce qui permet de retrouver l'original en répétant la commande.

TABLE D'APPEL DES SOUS PROGRAMMES

CODAGE	
0	/
1	/
2	CVRAM (CMEN MVRAM) MODIF (COLIN COLOUT RETIC LOUP DELOUP)
3	MODIF (COLIN COLOUT RETIC LOUP DELOUP)
4	FICH AFFICH (COLOUT)
5	FICH CALCUL (COLIN)
6	VGD
7	FICH2 CONTOR (COLIN COLOUT) APPROX (SOUPAR PS ANGLI (DROITE) EVOL2) TRACE (DROITR (COLIN COLOUT EVOLR (RAY CERCB (CERCR))) FILDRO
8	VGD3
9	EFFAS TRACE (DROITR EVOLR (COLIN COLOUT RAY CERCB (CERCR))) REMPLI (COLION COLOUT)
10	FICH SCALE (FACTER) CENTRE DECAL
11	FICH ITALIC (ITA) CENTRE DECAL
12	FICH ROTATE (ROT) CENTRE DECAL
13	FICH EPAIS (EPA) CENTRE DECAL
14	FICH DECAL
15	/
16	/
17	FICH2 EFFAS TRACE (DROITR EVOLR (COLIN COLOUT RAY CERCB (CERCR))) REMPLI (COLIN COLOUT)
18	FILVER (COLIN COLOUT) FILHOR (LIGIN LIGOUT)
19	FICH COMPAR (COLIN COLOUT)
20	FICH POSI
21	FILIG (LIGIN LIGOUT) FICOL (COLIN COLOUT)
22	FICH ORGCOD (COLIN COLOUT)
23	FICH CODORG (COLIN COLOUT)
24	ECHEL (LIGIN LIGOUT COLIN COLOUT)

CODAGE

Généralités

Le programme CODACE sélectionne l'opération et receuille les paramètres d'appel précisés au chapitre UTILISATION DU PROGRAMME. Pour limiter le nombre de noms de fichiers demandés à l'utisateur, il utilise des fichiers temporaires.

Liste des fichiers temporaires

DK0:ITA.TMP
Contient le caractère mis en italique.

DK0:ROT.TMP
" rotation.

DK0:ECH.TMP
" à l'échelle.

DK0:EPA.TMP
" épaissi.

DK0:DEC.TMP
" recentré aprés une transformation géométrique.

DK0:COD.TMP
" codé par contour avant le filtrage des droites.

DK0:POS.TMP
" codé par plage, décalé et permet de le replacer dans le fichier d'origine.

Liste des paramètres

NN (0) contrôle la trace du programme.

0 : pas de trace.

2 : trace les coordonnées des sous-parois et les résultats des essais de droites et de courbes évoluées.

3 : trace 2 plus les points lus au cours des suivis de contours.

NA (1) écart minimal entre un point F et la corde AB.

ND (1) tolérance sur les droites lors du filtrage des droites.

NEV (1) tolérance sur les courbes évoluées.
NEV = 0 codage par droites uniquement.

N1 (0) limite des cercles tracés.

N4 (0) correction du point F. 0 : sans.
1 : minimisation des écarts signés.
-1 : " non signés.

INUM (1) controle du biais de tracé des cercles.
IDEN (2)

X (1.) correction d'arrondi lors d'une mise à l'échelle; X est le seuil en dessous lequel les coordonnées contiguës sont forcées à l'égalité.

IJ (-64) indique si nous marquons ou pas les traits d'épaisseur inférieure à un point.
-64 : marqués.
0 : non marqués.

IIF (2) indique si nous corrigeons les croisements lors d'amincissement.
2 : correction.
4 : pas de correction.

Tableaux de stockage

Le programme utilise trois tableaux T, BUF, BUF2; T contient le nom du fichier de commande ou de la console d'entrée des commandes, il est rempli au lancement du programme ou aprés la commande 16, le numéro d'unité logique utilisé pour effectuer les lectures est le 1.
BUF et BUF2 respectivement remplis par FICH et FICH2 sont utilisés pour lire et écrire des fichiers séquentiels; les numéros d'unités logiques sont le 3 et le 4.

FICH & FICH2

Ils remplissent respectivement les tableux BUF et BUF2 qui contiennent les noms des fichiers de lecture ou d'écriture.
FICH écrit NOM DU FICHIER :, remet à 0 le tableau BUF et effectue une lecture de 20 caractères.
FICH2 écrit NOM DU FICHIER CONTOUR :, remet à 0 le tableau BUF2 et effectue une lecture de 20 caractères.

CVRAM

Il effectue 32 fois une boucle lecture caméra écriture vidéo-ram de façon à saisir une image bicolore.
Nous utilisons le bit de poids fort (7).
Le programme CMEN, écrit en assembleur, remplit le tableau TAB avec 16 colonnes.
Le programme MVRAM, également écrit en assembleur, transfére le contenu de TAB en vidéo-ram.

COLIN COLOUT LIGIN LIGOUT

Programmes en assembleur d'accés à la vidéo-ram suivant les lignes ou les colonnes.
Paramètres d'appel (I,J,TAB(1),N).

I	numéro colonne pour COLIN ou COLOUT.
	numéro ligne pour LIGIN ou LIGOUT.
J	numéro ligne début pour COLIN ou COLOUT.
	numéro colonne début pour LIGIN ou LIGOUT.
TAB(1)	contient la liste en logique des valeurs à afficher (OUT).
	contiendra les valeurs lues (IN).
N	nombre de valeurs lues ou écrites.

VGD3

Programme en assembleur qui recopie le contenu de la vidéo-ram sur l'imprimante électrostatique. (GOULD)
La sortie est effectuée à l'échelle 1, nous n'examinons que le bit de poids fort (7).

VGD

Programme en assembleur qui recopie le contenu de la vidéo-ram sur l'imprimante électrostatique. (GOULD)
La sortie est effectuée à l'échelle 4, chaque point de la vidéo-ram est représenté par une matrice 4X4, ce qui permet de visualiser les trois bits de poids forts.

EFFAS

Programme en assembleur qui remet la vidéo-ram à blanc.

ECHEL

Il trace une grille de mesure en vidéo-ram. Il complémente une valeur toutes les dix pour les lignes et colonnes 50, 150, 300 et 450; et une valeur sur deux pour les lignes et colonnes 100, 200, 300 et 400.
Le rappel de ce programme une seconde fois permet de recouvrer l'image d'origine.

POSI

Il décale une image codée par plage et mémorisée sur un fichier. Le fichier d'entrée a le numéro d'unité logique 4 et celui de sortie le numéro 3. Les valeurs de décalage sont deux entiers qui constituent les paramètres d'appel et qui sont simplement rajoutés aux coordonnées du codage par plage.

PS

Programme en assembleur permettant de lire la prochaine valeur sur un contour. Le contour a été isolé par le programme CONTOR et est décrit par des points de codage par plage et des points de liaisons.

Paramétres d'appel (X,Y,SX,SY,IX,IY)

X	abscisse du dernier point lu.
Y	ordonnée "
SX	sens de déplacement en X.
SY	SY = -SX
IX	nouvel écart en X avec le dernier point.
IY	" Y "

Le programme lit dans l'ordre les points:

abscisse	ordonnée
X - SX	Y
X - SX	Y + SY
X	Y + SY
X + SX	Y + SY
X + SX	Y
X + SX	Y - SY
X	Y - SY
X - SX	Y - SY

Dés que nous trouvons un point du codage de liaison, nous affectons à X et Y les coordonnées de ce point, nous effaçons le point et nous recommençons l'algorithme.

Dés que nous trouvons un point du codage par plage, nous calculons les paramètres IX et IY, nous marquons le point comme point de codage par plage lu et nous retournons au programme appelant.

FILVER & FILHOR

Ils permettent d'éliminer les traits d'épaisseur inférieure à ER, qui est le paramètre d'appel de ces sous programmes.
FILVER utilise LIGIN & LIGOUT et traite les traits verticaux.
FILHOR utilise COLIN & COLOUT et traite les traits horizontaux.
TAB tableau de 512 logiques. (8 bits)

Algorithme

```
Faire pour I = 0 à 511
        TAB = Ligne ou colonne I
        K = blanc
        N = 0
        M = 0
        Faire pour J = 1 à 512
                Si TAB(J) = K
                   Alors K = TAB (J)
                   Sinon Si K = blanc
                              Alors Si J - N inf ER
                                       Alors TAB(N à J) = blanc
                                       Sinon M = J
                              Sinon Si J - M inf ER
                                       Alors TAB(M à J) = noir
                                       Sinon N = J
        Fin Faire
        Ligne ou colonne I = TAB
Fin Faire
```

FILIG & FICOL

Ils permettent de rectifier les bords horizontaux et verticaux pour un bruit de quantification égal à un point de grille.
FILVER utilise LIGIN & LIGOUT et traite les bords horizontaux.
FILHOR utilise COLIN & COLOUT et traite les bords verticaux.
T1, T2, T3 et T4 sont des tableaux de 512 logiques
Le tableau T2 contient la ligne (colonne) courante, T1 la précédente et T3 la suivante, T4 sert de tampon de sortie.
Nous effectuons d'abord le ou-exclusif entre T1 et T3, ce qui nous permet de détecter les zones de transition où se trouvent les bords. Ensuite dans chaque zone de transition, nous comptons le nombre de points blancs et noirs et nous forçons cette zone à la couleur majoritaire uniquement si nous trouvons plus de deux changements de teinte. Ce qui nous permet de corriger les erreurs de quantification sur les bords, sans supprimer les intérieurs.

Algorithme

```
T1 = Ligne ou colonne 0
T2 = Ligne ou colonne 1
Faire pour I = 3 à 511
          T2 = Ligne ou colonne I
          T1 = T1 ou-exclusif T3
          T4 = T2
          Faire pour chaque zone de transition "T1(K à L) = noir"
                        N = nombre de points blancs
                        M = nombre de points noirs
                        T = nombre de changement de teinte
                        Si T sup 2
                           Alors Si N sup M
                                    Alors T4(K à L) = blanc
                                    Sinon T4(K à L) = noir
                        T1 = T2
                        Ecrire ligne ou colonne I = T4
Fin Faire
```

MODIF

Il permet de modifier une image bicolore contenue dans la vidéo-ram. Il est possible de mettre à blanc ou à noir un point, une droite ou un rectangle, et de mettre à blanc un cadre de façon à isoler une forme. Lors de l'appel, le programme trace un réticule en X = 255 et Y = 255; celui-ci peut être déplacé en X et Y et permet de repérer le point courant. Nous avons prévu une loupe logicielle qui double ou quadruple les points centraux d'une matrice 64X64 centrée sur le réticule.
Le tracé d'une droite utilise un algorithme incrémental type BRESENHAM. Nous utilisons les sous-programmes RETIC, LOUP et DELOUP pour marquer le réticule et effectuer la loupe logicielle.

RETIC

Les paramètres d'appel sont deux entiers qui indiquent la position du réticule. Le programme complémente la ligne et la colonne correspondantes. L'utilisation du complément permet d'effacer le réticule par un second appel du sous-programme.

LOUP

Ce programme réalise une loupe logicielle. Les paramètres d'appel sont:

IX abscisse du point courant.
JX ordonnée "
L grossissement 2 ou 4.

Le programme efface le réticule, mémorise une matrice 64X64 de façon à pouvoir recouvrer l'image d'origine et remplace les points centraux par des matrices 2X2 ou 4X4.

DELOUP

Il retrouve l'image d'origine aprés une loupe logicielle. Les paramètres d'appel sont identiques à ceux de LOUP.

IX abscisse du point courant

JX ordonnée "

L grossissement 2 ou 4

Le programme se contente d'écrire la matrice 64X64 mémorisée par LOUP et replace le réticule.

CALCUL

Il calcule le codage par plage de l'image en vidéo-ram et mémorise les coordonnées dans un fichier. Le numéro d'unité logique associé au fichier par CODAGE est le numéro 4. Pour chaque point d'intersection du contour et de l'abscisse de balayage courante nous mémorisons l'abscisse et l'ordonnée en format I4, dans l'ordre des abscisses et des ordonnées croissantes. La fin du fichier est indiquée par une abscisse supérieure à 512.

AFFICH

Il relit un fichier écrit par CALCUL et place l'image correspondante en vidéo-ram. Nous utilisons le numéro logique 4 comme unité de lecture.

CONTOR

Il permet d'isoler les contours d'un caractère de façon à obtenir une image compatible avec le programme de suivi de contour PS.
L'algorithme de recherche du contour analyse pour chaque point appartenant au caractère les deux voisins verticaux et horizontaux.
Soit A l'image d'origine et B l'image résultat;
i l'indice de ligne et j l'indice de colonne;
$A_{i,j}$ = noir si le point appartient au caractère;
$A_{i,j}$ = blanc " n'appartient pas "

si $A_{i,j}$ = noir
  alors si $A_{i,j+1}$ = blanc ou $A_{i,j-1}$ = blanc
    alors $B_{i,j}$ = plage
    sinon
      si $A_{i+1,j}$ = blanc ou $A_{i-1,j}$ = blanc
        alors $B_{i,j}$ = liaison
        sinon $B_{i,j}$ = fond
  sinon $B_{i,j}$ = fond

Nous devons théoriquement utiliser deux images, en pratique il n'est nécessaire de mémoriser que trois lignes ou colonnes.

APPROX

Les paramètres d'appel sont:

N écart minimal entre le point de carrure et la corde.

N1 distance maximale entre une courbe évoluée et le contour.

NN mode de trace pour corriger le programme.

N2 type d'ajustement du point F.

Il controle le remplacement d'un contour par les interpolants. Pour cela il lit une sous paroi en appelant SOUPAR, recherche le point de carrure grace à ANGLI et calcule les paramètres des courbes évoluées ainsi que la validité de celles ci en appelant EVOL2.

Algorithme de recherche d'interpolants

```
Lire une sous-paroi
Faire jusqu'à l'obtention d'une bonne approximation
      Recherche du point de carrure
      Si droite
        Alors Marquer droite
        Sinon Essai courbe évoluée
              Si 2 demi-courbes correctes
                Alors Marquer une courbe évoluée
                Sinon Si Première correcte
                              Alors Marquer une demi courbe évoluée
                              Sinon Si Seconde correcte
                                            Alors Mémoriser une demi courbe évoluée
      Réduire le domaine
Fin faire
```

SOUPAR

Programme de lecture d'une sous-paroi pour APPROX. Les paramètres d'appel sont:

FD	en entrée Indicateur	Premier appel	-1.
		Fin de contour	0.
		Suite de contour	1.
	en sortie Indicateur	Fin de caractère	-1.
		Normal	0.
M	pointeur de fin de sous-paroi.		
NN	mode de trace.		

Dans le cas d'un premier appel ou d'une fin de contour le programme recherche un nouveau contour respectivement à partir de la colonne 511 ou de la dernière colonne analysée pour la recherche précédente.
Il détermine si le contour est intérieur ou extérieur et corrige le premier point s'il appartient à une portion de contour verticale.
Pour suivre le contour le programme appelle PS qui ne lui transmet que les points corespondants au codage par plage. Une fin de sous-paroi peut être dû à l'abscence de point suivant ou à une variation inverse en X ou Y.
Dans le cas d'une fin de contour nous rajoutons les premiers points lus de façon à fermer le contour. Dans le cas d'une fin de sous paroi nous modifions le sens de parcours en X ou en Y.
Pour indiquer le milieu d'une portion de contour horizontale, nous utilisons un pointeur de début et un de fin d'horizontale et nous indiquons la moyenne de ceux-ci comme pointeur de fin de sous-paroi.
Pour corriger une fin de sous paroi verticale, nous devons rajouter un point décalé d'une abscisse par rapport au sens de variation en X et ayant une ordonnée égale à la moyenne des ordonées des extrémités de la verticale; cette opération est effectuée chaque fois que la variation en X est nulle et celle en Y supérieure à 3.

ANGLI

Les paramètres d'appel sont:

I pointeur sur le premier point du contour.

J " dernier "

L pointeur sur le point de carrure F.

K " intermèdiaire I.

IJ " " J.

N écart minimal entre le point de carrure et le contour.

NN mode de trace.

KI nombre de points à décaler pour l'ajustement de F.

C'est le programme de recherche des points spéciaux d'une courbe évoluée, point de carrure F et points intermédiare I et J. Ces points sont obtenus par appel à DROITE qui indique où se trouve le point de carrure sur la portion de contour précisée.

IL permet de détecter et d'isoler les points d'inflexion; le sous programme DROITE précise si l'écart entre la corde et le contour a toujours eu le même signe; dans le cas contraire il existe un point d'inflexion. La localisation de ce point est réalisée grace à DROITE; nous prenons comme borne le dernier point de "carrure" et nous inversons le sens de parcours de la corde dans le cas de plusieurs maxima locaux.

DROITE

Programme de recherche du point de carrure. Il mesure l'écart entre la courbe et la corde, indique le premier maximun local, supérieur à l'erreur détecté, et précise si l'écart à changer de signe ou non.

Les paramètres d'appel sont:

I pointeur sur le premier point du contour.

J " dernier "

K pointeur sur le point de carrure.

M2 indicateur de changement de signe de l'écart.

0 changement de signe.

1 pas de "

N écart minimal entre le point de carrure et le contour.

NN mode de trace.

KI Nombre de point à décaler pour l'ajustement de F.

EVOL2

Les paramètres d'appel sont:

I pointeur sur le premier point du contour.

J pointeur sur le point intermédiaire I ou J.

IB " dernier point du contour.

IX abscisse indiquant la direction de la tangente.

IY ordonnée "

IAX écart entre l'abscisse début et celle du point F.

IAY " ordonnée "

IE Indicateur si approximation correcte.
0 correcte.
1 incorrecte.

N écart maximal entre la courbe évoluée et le contour.

NN mode de trace.

CER valeur en réel de l'écart entre la courbe évoluée et le contour; si IE = 1 alors CER = 0.

IFF pointeur sur le point de carrure.

Le programme travaille par demi-courbe évoluée et calcule la tangente en utilisant l'algorithme de recherche du point intermédiaire. Une fois la tangente calculée, il détermine le point intermédiaire et mesure la distance entre ce point et le contour. Dans le cas où l'écart est inférieur à N la courbe est acceptée (IE = 0), sinon elle est refusée (IE = 1 & CER = 0.).

FILDRO

Le paramètre d'appel N indique l'écart maximal entre le contour et les droites. Le programme lit le fichier contenant la description des contours sur l'unité logique 3 et écrit le résultat du filtrage sur la 4. Il est chargé de supprimer les droites redondantes; le programme de codage duplique les horizontales et les droites de part et d'autre des points d'inflexion. Il corrige également les verticales qui ont le point milieu décalé d'une abscisse pour améliorer les courbes évoluées.

Dans un premier temps le programme lit un contour, le place dans un tableau et corrige les verticales. Nous avons une verticale lorsque deux droites contiguës ont les abscisses extrèmes égales et l'abscisse commune décalée d'un point par rapport aux extrèmes. La correction consiste à supprimer le point milieu.

Ensuite le programme relit le tableau et corrige les duplications de droites. Lorsque deux droites sont contiguës, nous les remplaçons par une droite et nous mesurons la distance entre cette dernière et le point commun. Dans le cas où cette distance est inférieure à l'écart N nous supprimons le point milieu.

TRACE

Les paramètres d'appel sont:

J indique si nous marquons ou pas les traits inférieurs à un point. -64 : marqués.
0 : non marqués.

INUM précisent le biais utilisé lors du tracé des cercles.
IDEN

N2 indique la limite inférieure de tracé des cercles.

Le programme lit le fichier de numéro logique 3, supprime les droites et les courbes évoluées de longueur nulle (par mise à l'échelle), appelle DROITR pour tracer les droites et EVOLR pour tracer les courbes évoluées.

DROITR

Les paramètres d'appel sont:

IX1 coordonnées du début de la droite.

IY1

IX2 " de fin "

IY2

IJ indique si nous marquons ou pas les traits inférieurs à un point. -64 : marqués.

0 : non marqués.

Le programme effectue le tracé des droites en utilisant un algorithme incrémental type BRESENHAM. Pour symétriser la forme du caractère nous calculons le contour dans le sens des abscisses croissantes.

Lorsque deux points du contour ont les mêmes coordonnées; nous devons, pour respecter la relation entre l'intérieur-extérieur et la parité du nombre d'ordonnée, effacer ce point ou le marquer sur un autre plan de bits; ce choix est paramètré par l'indicateur IJ.

EVOLR

Les paramètres d'appel sont:

IXA coordonnées du point A.

IYA

IXAP " donnant la direction de la tangente en A.

IYAP

IXF " du point F.

IYF

IXBP " donnant la direction de la tangente en B.

IYBP si IXBP = 0 et IYBP = 0 alors nous n'avons qu'une demi courbe évoluée.

IXB coordonnées du point B.

IYB

J indique si nous marquons ou pas les traits inférieurs à un point. -64 : marqués.

0 : non marqués.

INUM précisent le biais utilisé lors du tracé des cercles.

IDEN

N2 indique la limite inférieure de tracé des cercles.

Le programme calcule les points intermèdiaires I et J ainsi que les quatre centres de cercles O1, O2, O3 et O4. Pour chaque cercle il appelle CERCB qui teste la présence de point à tangente verticale ou horizontale. Le sous programme RAY remet à 0 les deux paramètres d'appel.

CERCB

Les paramètres d'appel sont:

IXI coordonnée du premier point.

IYI

IXB " du dernier point.

IYB

IX1 " du centre du cercle.

IY1

J indique si nous marquons ou pas les traits inférieurs à un point. -64 : marqués.

0 : non marqués.

INUM précisent le biais utilisé lors du tracé des cercles.

IDEN

T rayon du cercle.

N2 indique la limite inférieure de tracé des cercles.

Le progamme teste si l'arc de cercle appartient à un seul quadrant; dans la négative il coupe l'arc de cercle en deux arcs de part et d'autre du point à tangente verticale ou horizontale.

CERCR

Les paramètres d'appel sont:

IXI coordonnée du premier point.

IYI

IXB " du dernier point.

IYB

IX1 " du centre du cercle.

IY1

IJ indique si nous marquons ou pas les traits inférieurs à un point. -64 : marqués.

0 : non marqués.

INUM précisent le biais utilisé lors du tracé des cercles.

IDEN

N2 indique la limite inférieure de tracé des cercles.

Le programme trace les cecles en utilisant une variante de l'algorithme incrémental de BRESENHAM. Comme pour les droites, les cercles sont tracés suivant les abscisses croissantes.

REMPLI

L'indicateur IJ est égal au paramètre d'appel IJ de DROITR et CERCR et précise la nuance de gris affectée aux points du contour.
Le programme lit l'image colonne aprés colonne et trace un vecteur entre chaque couple de points appartenant au contour.

COMPAR & ORGCOD & CODORG

Ils comparent l'image en vidéo-ram et celle d'origine mémorisée sur fichier disque. Pour chaque pixel:
COMPAR réalise un ou-exclusif.
ORGCOD soustrait l'image en vidéo-ram à celle mémorisée.
CODORG " mémorisée à celle en vidéo-ram.

DECAL

Les paramètres d'appel sont:
I1 décalage en X.
J1 " Y.
Il rajoute à chaque coordonnée le décalage; il est utilisé pour centrer le caractère aprés une transformation géométrique. Les valeurs des décalages sont calculées par CENTRE. Il travaille sur fichiers, 3 est le fichier d'entrée et 4 celui de sortie.

CENTRE

Les paramètres d'appel sont:
I1 décalage en X.
J1 " Y.
Il lit le fichier 3 et recherche les maximun et minimun en X et Y. Il calcule ensuite les décalages pour centrer le caractère, ceux-ci seront transmis à DECAL.

EPAIS

Les paramètres d'appel sont:

IP valeur d'épaississement.

IIF indique si nous corrigeons les croisements de parois.

2 : corrigés.

4 : non corrigés.

Il lit un contour sur le numéro logique 3 et le range dans un tableau. Pour chaque point il calcule ou lit les deux demi-tangentes et effectue la transformation par un appel à EPA. Le controle de croisement est effectué en comparant les signes des écarts entre deux points consécutifs avant et aprés transformation. Dans le cas où les signes sont différents la correction consiste à supprimer l'un des points.

EPA

Les paramètres d'appel sont:

I coordonnées du point courant.

J

I1 " donnant la direction de la demi tangente avant.

J1

I2 " " aprés.

J2

P valeur d'épaississement.

Soit D2 = la moyenne des demi tangentes.

I = I + P * sin D2

J = J + P * cos D2

SCALE

Les paramètres d'appel sont:

NX coefficients d'échelle en X.

DX

NY " Y.

DY

X2 seuil de correction d'arrondi.

Le programme met à l'échelle les coordonnées des points appartenant aux contours par appel à FACTER. Afin de maintenir la précision sur le calcul des angles, les points indiquant les directions des tangentes sont mis à l'échelle en relatifs par rapprot aux points de bases et en gardant les plus grands écarts possibles. La correction des arrondis est réalisée en mettant à l'échelle l'écart entre deux points contigus et en forçant l'égalité dès que cette différence devient inférieure au seuil X2.

FACTER

Les paramètres d'appel sont:

I abscisse ou ordonnée.

X échelle pour la valeur précédente.

I = I * X

ITALIC

Le programme lit le fichier de numéro logique 3 et écrit le résultat en 4. Les coordonnées de chaque point sont envoyées au sous programme ITA qui réalise la transformation.

ITA

Les paramètres d'appel sont:

I coordonnées d'un point.

J

I = I + J * tg 13°

ROTATE

Le paramètre d'appel ANG indique l'angle de rotation en degré. Le programme lit le fichier de numéro logique 3 et écrit le résultat en 4. Les coordonnées de chaque point sont envoyées au sous programme ROT qui réalise la transformation.

ROT

Les paramètres d'appel sont:

I	coordonnées d'un point.
J	
C	cosinus de l'angle de rotation.
S	sinus "

```
I = I * C - J * S
J = J * C + I * S
```

TAILLE DES SEGMENTS

PROGRAMMES	Taille en Octets
CODAGE CVRAM CMEN MVRAM VGD3 VGD COLIN COLOUT LIGIN LIGOUT PS	29976
FILDRO FILVER FILHOR FILIG FICOL	4592
EPAIS EPA	5216
SCALE FACTER	3524
ITALIC ITA	1292
TRACE DROITR EVOLR CERCB CERCR RAY	6684
ROTATE ROT	1888
DECAL CENTRE	1508
EFFAS POSI	212
REMPLI	380
APPROX SOUPAR ANGLI EVOL2 DROITE	8596
CONTOR ECHEL	900
MODIF RETIC LOUP DELOUP	3352
AFFICH COMPAR ORGCOD CODORG	2300
CALCUL	544
FICH FICH2	308
TOTAL	71280

Bibliographie

COU 73 P. COUEIGNOUX
"Compression of type faces by contour coding", MS thesis, Dep. Elec. Eng., Massachussetts Institute of Technology, USA, JAN 1973, 60 pages.

COU 75 P. COUEIGNOUX
"Generation of roman printed fonts", Ph.D, Dep. Elec. Eng., Massachussets Institute of technology, JUN 1975,139 pages.

HOR 77 B.K. HORN
"Circle Generator for display devices", CGIP, vol 6, 1977,pages 196-198.

BRE 77 J.E. BRESENHAN
"A linear algorithm for incremental digital display of circular arcs", comm. of the ACM, vol 20 N° 2, pages 100-106, FEB. 1977.

LAR 77 LARRY S. DAVIS
"Understanding shape : Angles and Sides", IEEE transaction on computer, vol C-26 N° 3, MAR 1977, pages 236-242

HOU 78 M. HOURDEQUIN
"Génération de polices d'imprimerie pour photocomposeuse digitale", doct. ing. ENS des Mines de Saint-Etienne, France, NOV. 1978, 150 pages.

KNU 78 D.E. KNUTH
"Mathematical typography", Stanford University Comp. Science Dpt, Stan-CS-78-648, FEB. 1978, 68 pages.

HOC 79 P. COUEIGNOUX and M. HOURDEQUIN
"Specifying arbitrary planar smooth curves for fast drawing", Eurographis, 1979.

BLO 79 M. BLOCH
"Architecture parallèle de génération du contour d'une tache", DEA, ENS des Mines de Saint-Etienne, OCT. 1979, 54 pages.

COG 80 R. GUEDJ and P. COUEIGNOUX
"Computer generation of colored planar patterns on TV-like rasters", Proceeding of the IEE, vol 68, N° 7, JUI. 1980, pages 909 à 922.

SIC 80 C. SICO
"Approximation par arcs de cercles du contour d'une tache", DEA, ENS des Mines de Saint-Etienne, NOV. 80, 50 pages.

COU 81 P. COUEIGNOUX
"Character generation by computer", CGIP, vol 16, 1981, pages 240-269.

BLO 81 M. BLOCH
"Génération de taches bicolores, application aux caractères d'imprimerie, problèmes de nature ordinale", doct. ing., ENS des Mines de Saint-Etienne, France, JUI. 81, 134 pages.

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU les rapports de présentation de M. COUEIGNOUX et M. LUCAS

M. SICO Christian

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité Informatique

Fait à Saint-Etienne, le 3 mars 1982

Le Directeur de l'EMSE,