SOMMAIRE

LISTE DES ABREVIATIONS

BMP :Bitmap.

CCD : (Charge Couple Devices).

DPI : Dots per inch .

DXF: Drawing eXchange Format

GIF : Graphic Interchange Format.

MOS : Metal-Oxide-Semiconductor.

PNG : Portable Network Graphics.

PPP : Point par pouce.

RGB : Red Green Blue.

RVB : Rouge Vert Bleu.

SVG : Scalable Vector Graphics.

TIFF : Tagged Image Format File.

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION

Il existe deux types de formats de donnée comme support d'une image numérique, le format matriciel et le format vectoriel. Le travail présenté dans ce mémoire concerne l'étude de ces deux types de formats et diverses méthodes de traitement et création d'image numérique.

L'utilisation des images vectorielles n'est pas encore bien reconnue actuellement, on l'emploi souvent dans le domaine industriel comme le dessin avec Autocad. Les images vectorielles sont utilisées pour avoir de précision, comme pour déterminer le découpage d'une feuille de métal sur une fraiseuse numérique. Toutes les machines d'usinage numérique ou automatique utilisent toujours des images vectorielles comme référence ou gabarit.

L'une des caractéristiques avantageuse de l'image vectorielle par rapports à l'image matricielle est qu'elles ne présentent pas de pixels quand on l'agrandie. On peut faire des zooms sans modification de la qualité et du contenu des images.

Les images vectorielles sont obtenues généralement par dessin manuel. La conversion d'une image matricielle en image vectorielle est encore en phase de recherche, raison pour laquelle ce mémoire a pour titre : ***Transformation d'image matricielle en image vectorielle par combinaison d'arc***.

Pour ce faire notre manuscrit sera divisé en 4 chapitres. Le premier chapitre consiste à décrire les principes de l'image numérique et les caractéristiques des images matricielles. Le deuxième chapitre s'articule autour des principes et caractéristiques de l'image vectorielle et la comparaison des deux types d'images numériques. Notre but est d'avoir une image vectorielle à partir une image matricielle, et vue que notre image source est constituée par des courbes, nous allons utiliser l'approximation d'une courbe par un arc paramétré à l'aide deux vecteurs, ce que nous développerons dans le troisième chapitre. Dans le quatrième chapitre, nous allons présenter notre méthode pour la conversion d'une image matricielle en image vectorielle.

CHAPITRE I

IMAGE OU PHOTO AU FORMAT MATRICIEL.

Il existe deux types de format de codage numérique d'une image : le format matriciel et le format vectoriel. Ce chapitre présentera les propriétés générales des images numériques et les caractéristiques des images matricielles.

I . 1 . Définition

L'image est une représentation d'une personne ou d'un objet par la peinture, la sculpture, le dessin, la photographie, le film, etc. C'est aussi un ensemble structuré d'informations qui, après affichage, a une signification pour l'œil humain.

Une Photo, doit être impérativement prise par un appareil photo, et le plus souvent représentative de la vie réelle. La photo est donc une image, alors que l'image ne peut pas forcément être une photo.

On désigne sous le terme d'**image numérique** toute image (dessin, icône, photographie, etc.) *acquise, créée, traitée* ou *stockée* sous forme binaire (suite de 0 et de 1). Lorsque l'on dit images numériques il faut tout de suite avoir en tête que l'on parle donc de signal échantillonné. Les images proprement dites sont vues comme des fonctions mathématiques de $\mathbb{Z}$ x $\mathbb{Z}$ dans $\mathbb{Z}^n$ où $\mathbb{Z}$ représente l'ensemble des entiers relatifs et n le nombre de couleurs de base composant l'image (Exemple $n=1$ pour du niveau de gris et $n=3$ pour les images RGB). [1]

Pour des raisons de commodité de représentation pour l'affichage et l'adressage, les données images sont généralement rangées sous formes de tableau (ou matrice) I de p lignes et q colonnes. Chaque élément $I(x,y)$ du tableau représente un pixel de l'image et à sa valeur est associé un niveau de gris codé sur m bits (2^m niveaux de gris ; 0 = noir ; $2^m - 1$ = blanc). La valeur en chaque point exprime la mesure d'intensité lumineuse perçue par le capteur. La Figure 1.1 représente l'affichage et l'adressage des données images.

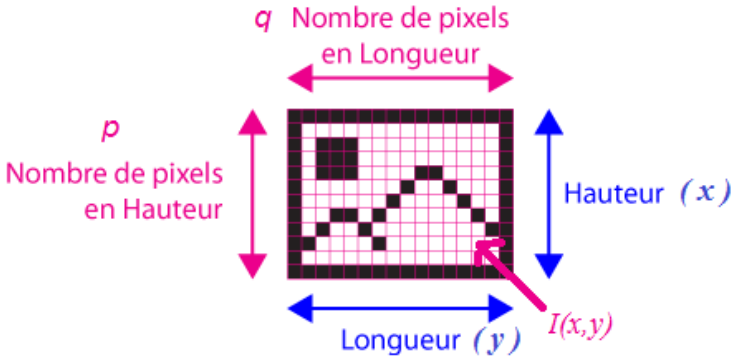


*Figure 1.1 - Division d'une image en forme de tableau p * q.*

I . 2 . Acquisition d'images

Tout système de traitement d'images peut être vu comme la combinaison de deux étapes : l'acquisition et le traitement proprement dit. La qualité des résultats du système dépend de l'algorithme mis en place et de son adéquation au problème posé, mais aussi de la qualité initiale des images.

I . 2 . 1 . Caméras à tubes

Ces capteurs sont relativement anciens et font appel à l'archéologie électronique puisqu'ils ont étés inventés en 1931 par Vladimir Zworykin.

Leur principe est le suivant : la scène est projetée sur une cible photosensible qui convertit une quantité de lumière en une quantité de charge. Il se forme ainsi un relief de potentiel correspondant à l'image analysée sur la cible. Celle-ci est l'anode du tube dont la source électronique est la cathode. Les électrons issus de la cathode sont attirés par l'anode polarisée positivement. Ce flux d'électrons est dévié par l'ensemble des bobines de déviation afin de scruter l'ensemble de la cible rectangulaire à analyser, Illustré dans la Fig.1.2. Le courant issu de la cible correspond à l'image projetée. L'amplitude du courant est en fonction de l'éclairement.[1]

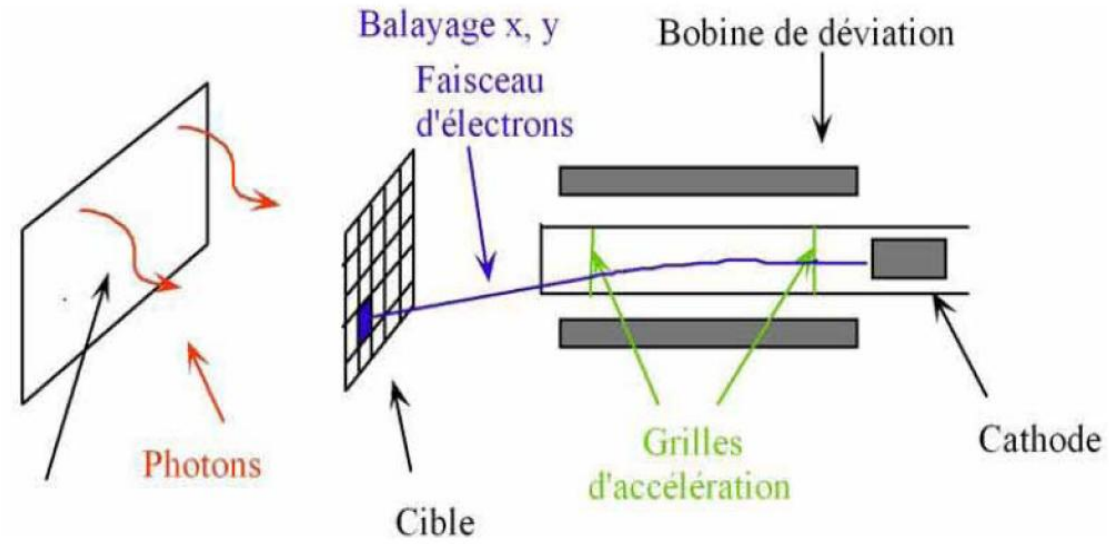


Figure 1.2 - Principe de fonctionnement d'un camera a tubes.

A ce jour, ces tubes ont été remplacés par des capteurs de type CCD (Charge Couple Devices). En effet, même si les tubes vidéo présentent certains avantages (bonne sensibilité spectrale de certains tubes dans l'infrarouge), ils ont aussi une durée de vie plus courte que les caméras CCD, ils produisent des images avec de fortes distorsions géométriques et ont un encombrement beaucoup plus important que celui des capteurs CCD.

I . 2 . 2 . Caméras CCD

Ces capteurs sont relativement récents. Leur principe est le suivant : la scène est projetée au moyen d'un objectif sur un réseau de capteurs discret, ce qui réalise un échantillonnage spatial de l'image. Chaque photo-élément convertit l'énergie lumineuse en énergie électrique. Il existe deux types de capteurs : les photodiodes ou les condensateurs MOS de type P. Dans les deux cas, l'arrivée de photons conduit à la formation d'une charge électrique sous la photodiode ou le photomos.

I . 2 . 3 . Principe du transfert de charge

Pour réaliser le transfert de charge, les condensateurs MOS sont associés les uns à la suite des autres. Le substrat des semi-conducteurs ainsi que l'isolant est commun. Les grilles métalliques sont indépendantes pour chaque capteur. Le but consiste à transférer successivement les charges électriques présentes sous les différentes grilles vers la sortie. La figure 1.3 illustre ce principe.

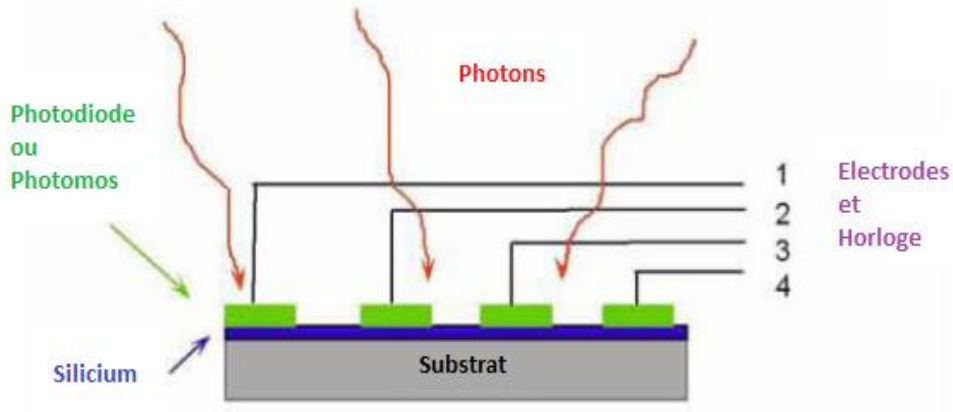


Figure 1.3 - Illustration du transfert de charge

Plusieurs types de transfert sont possibles, nous allons étudier le plus simple: le transfert biphasé. Supposons que des charges électriques soient présentes sous la grille 1 (électrode 1 polarisée). Pour les déplacer, on va successivement polariser les électrodes 2, 3, 4, etc. comme le montre la Fig. 1.4. Pour optimiser le déplacement des charges, une phase intermédiaire où deux électrodes sont simultanément polarisées est mise en place.

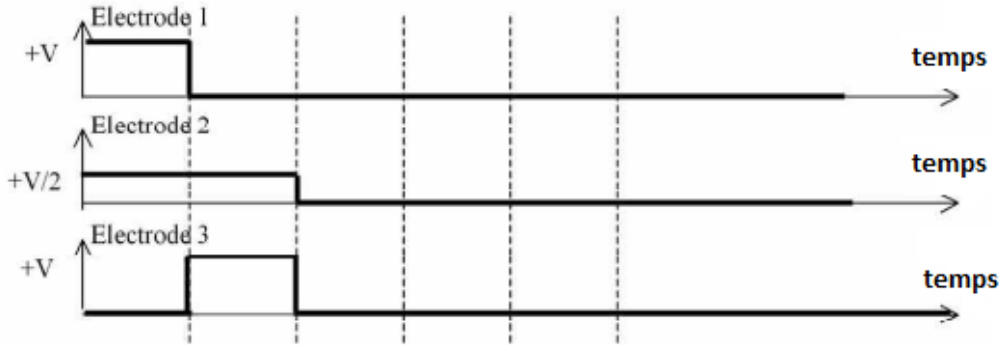


Figure 1.4 - Transfert biphasé

I . 2 . 4 . Les CCD linéaires

La zone photosensible est composée d'un alignement de photo-éléments. La Figure 1.5 illustre le transfert entre la zone sensible et la zone de transfert.

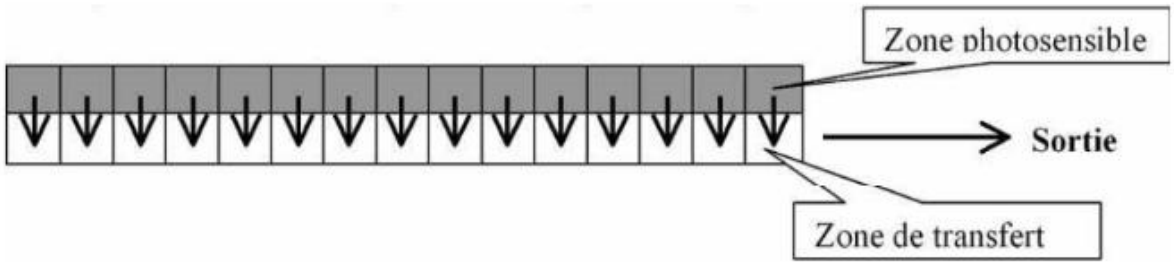


Figure 1.5 - Illustration du principe des CCD linéaires

Les charges de la zone photosensible sont transmises toutes en même temps en zone de transfert. Elles sont ensuite évacuées séquentiellement vers la sortie.

I . 2 . 5 . Les CCD matricielles

a. CCD matricielles avec zone de transfert.

Le capteur possède deux zones : une composée des capteurs photosensibles, l'autre est une zone de transfert comme dans la Fig. 1.6. Après la saisie de l'image, elle est transférée par un déplacement vertical en zone de transfert puis évacuée ligne par ligne en zone mémoire. Les charges élémentaires sont alors récupérées séquentiellement en sortie. Le problème de cette technologie est que lors du transfert en zone mémoire, les cellules photosensibles restent actives. La première ligne image passe donc devant toutes les autres cellules durant son transfert, ce qui amène des variations.

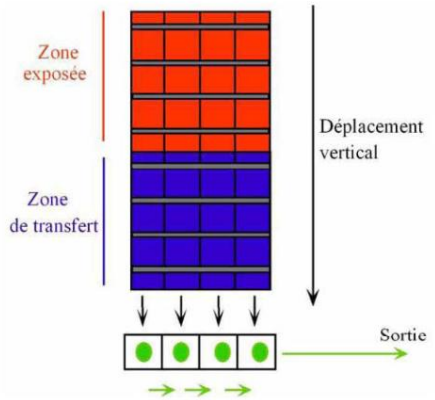


Figure 1.6 - Illustration du principe des CCD matricielles avec zone de transfert.

b. CCD matricielles avec transfert direct

Avec cette nouvelle technologie, la zone photosensible est transférée directement en zone masquée par un déplacement vertical. Ensuite, les charges sont transférées ligne par ligne en zone mémoire avant d'être évacuées séquentiellement, comme ce qui est représenté sur la Fig. 1.7. L'inconvénient ici est que l'on aura une mauvaise résolution spatiale dans l'image finale.

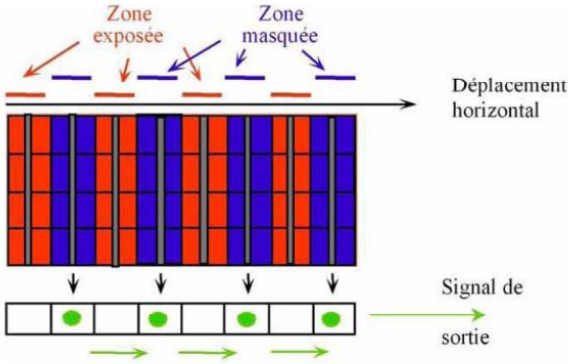


Figure 1.7 - Illustration du principe des CCD matricielles avec transfert direct.

I . 3 . Image matricielle

Une image matricielle, ou « carte de points » (de l'anglais bitmap), est une image constituée d'une matrice de points colorés. C'est-à-dire, constituée d'un tableau, d'une grille, où chaque case possède une couleur qui lui est propre et est considérée comme un point. Il s'agit donc d'une juxtaposition de points de couleurs formant, dans leur ensemble, une image.

Cette expression (Image matricielle) est principalement utilisée dans les domaines de l'imagerie numérique (infographie, informatique, photographie numérique, etc.) afin de marquer l'opposition de ce concept avec celui des images vectorielles.

L'image de la Fig.1.8 représente une image matricielle binaire, les 1 représentent les pixels de couleur blanc et les 0 sont les noirs.

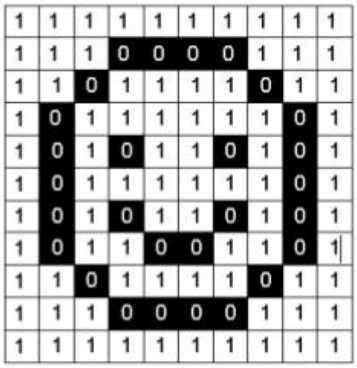


Figure 1.8 - Illustration d'une image matricielle noir & blanc sans niveau de gris.

I . 4 . Caractéristiques

L'image est un ensemble structuré d'informations caractérisé par plusieurs paramètres.

I . 4 . 1 . Pixel

Le terme Pixel est la contraction de l'expression anglaise « Picture Elements » : éléments d'image. Le pixel est le plus petit point de l'image, c'est une entité calculable qui peut recevoir une structure et une quantification. Si le bit est la plus petite unité d'information que peut traiter un ordinateur, le pixel est le plus petit élément que peuvent manipuler les matériels et logiciels d'affichage ou d'impression. La Figure 1.9 montre les pixels quand on agrandie une image matricielle,

a) ***b)***

Figure 1.9 - Illustration d'un pixel sur une image matricielle,
***a** : image original avec un cadre repère de zoom, **b** : image de ce qui est dans le cadre de zoom, on voit les pixels.*

I . 4 . 2 . Dimension ou définition

La définition d'une image est définie par le nombre de points la composant. En image numérique, cela correspond au nombre de pixel qui compose l'image en hauteur (axe vertical) et en largeur (axe horizontal) soit p x q. *206 pixels par 345 pixels* par exemple, abrégé en «206x345» dont le nombre de points est de 71070 points.

I . 4 . 3 . Résolution

La résolution d'une image est définie par un nombre de pixels par unité de longueur de la structure à numériser (classiquement en PPP ou DPI). Ce paramètre est défini lors de la numérisation (passage de l'image sous forme binaire), et dépend principalement des caractéristiques du matériel utilisé lors de la numérisation. Plus le nombre de pixels par unité de longueur de la structure à numériser est élevé, plus la quantité d'information qui décrit cette structure est importante et plus la résolution est élevée. La résolution d'une image numérique définit le degré de détail de l'image. Ainsi, plus la résolution est élevée, meilleure est la restitution. La Figure 1.10 illustre deux différentes résolutions.

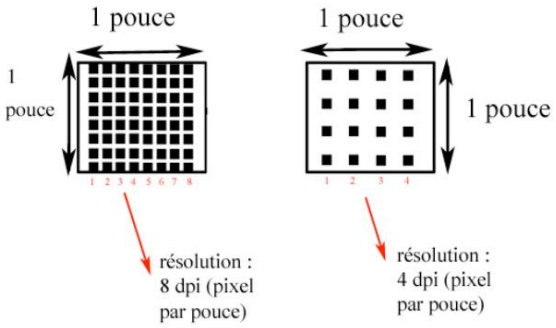


Figure 1.10 - Résolution en dpi (1 pouce = 2,54cm) ,
la Fig. 1.10.a présente une résolution plus élevée que la Fig. 1.10.b.

La Figure 1.11 représente la même image mais avec deux résolutions différentes. On remarque la différence sur la qualité entre les deux images du fait que la Fig. 1.11.a possède une résolution plus élevée avec une résolution de 250dpi que la Fig.1.10.b qui n'a qu'une résolution de 40dpi.

Figure 1.11 - Comparaison de deux résolutions différentes d'une même image.
a) résolution 250 dpi ; b) résolution 40 dpi

I . 4 . 4 . Bruit

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur.

I . 4 . 5 . Contours et textures

Les contours représentent la frontière entre les objets de l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure des objets de l'image. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes.

I . 5 . Codage des couleurs

I . 5 . 1 . Les couleurs primaires

Les **couleurs primaires** en peinture, sont les 3 couleurs de base dans le monde de la matière et des pigments, permettant d'obtenir les autres teintes par mélanges: on parle de la théorie des couleurs par **synthèse soustractive** : ce sont le bleu Cyan, le Jaune et le rouge Magenta (rose violacé), on utilise ces couleurs généralement pour les imprimantes.

Mais , Dans le domaine des **longueurs d'ondes** (lumière visible) , c'est le principe que l'on utilise sur tous les écrans, on a la théorie des couleurs par **synthèse additive**, élaborée à partir de la décomposition de la lumière en différentes longueurs d'ondes, les 3 couleurs de base qui permettent d'obtenir les autres teintes par mélanges sont **le bleu** (bleu violet), **le Vert** et le **Rouge vermillon** (légèrement orangé). On les appelle **les primaires RVB**, (ou primaires additives).Dans la synthèse additive (les ondes), le vert, le bleu violet et le rouge vermillon ne peuvent être obtenus par mélange. Et on a le tableau récapitulatif Tableau I suivant:

Tableau I : Composition des couleurs de base

Les couleurs Primaires			Résultats
Rouge vermillon	Vert		Magenta
Rouge vermillon		Bleu violet	Cyan
	Vert	Bleu violet	Jaune
Rouge vermillon	Vert	Bleu violet	Blanc

Et les autres couleurs sont obtenues par proportion en additionnant ces 3 couleurs de base. La Figure 1.12 représente les 3 couleurs primaires additives quand il se mélange.

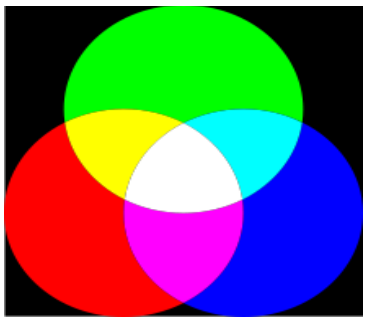


Figure 1.12 - Les 3 couleurs primaires dans la synthèse additive

I . 5 . 2 . Echantillonnage des couleurs

L'échantillonnage est le processus de conversion (encodage) d'une image en format numérique. Il détermine la qualité du signal couleur de l'image et se mesure en nombre de bits.

La régularité du pas d'échantillonnage dépend de la finesse du codage en numérique.

• Une image codée sur 1 bit est l'équivalent du film au trait. L'image n'a aucun niveau de gris. Le pixel prend la valeur 0 ou 1, noir ou blanc.

• Une image codée sur 8 bits correspond à une image monochrome, c'est-à-dire une image ayant au moins 256 niveaux de gris possibles, le pixel prenant les valeurs comprises entre 0 et 256.

• Une image codée sur 8 bits par couleur primaire RVB correspond à une image en millions de nuances de couleur différentes. Chaque pixel est représenté par ses composantes : rouge, verte et bleue. Chacune est stockée sur 1 octet, et peut donc prendre des valeurs de 0 à 255 niveaux (2^8). Chaque pixel nécessite donc 3 octets ; c'est pourquoi ce mode est appelé aussi 24 bits par pixel. Il y a dans ce mode 16,777216 millions de couleurs possibles (256^3)

Le Tableau II récapitule les différents types d'échantillonnages avec les caractéristiques de l'image obtenue.

Tableau II : Récapitulatif de l'échantillonnage des couleurs

Type d'image	**Echantillonnage**	**Nombre de bits**	**Niveau**	**Nombre de couleurs**
Dessin au trait	1bit / pixel	1	Noir et blanc	2
Image en niveaux de gris	8 bits / pixel	8	256 niveaux de gris	256
Image en RVB	8 bits / couleurs primaire	24	256 x 256 x 256 couleurs	16, 7 millions
Image RVB+transparence (utilisée en vidéo)	8 bits / couleurs primaire+8 bits pour la transparence	32	256 x 256 x 256 couleurs	16, 7 millions
Image CMJN	8 bits / couleurs primaire	32	256 x 256 x 256 x 256 couleurs	4,3 milliards

Dans la Fig.1.13, le smiley dans le coin supérieur gauche est une image bitmap. Lorsqu'elle est élargie, les pixels individuels s'affichent sous forme de carrés. Zoomez plus fort, pour pouvoir les analyser, leurs couleurs sont construites en ajoutant les valeurs de rouge, vert et bleu.

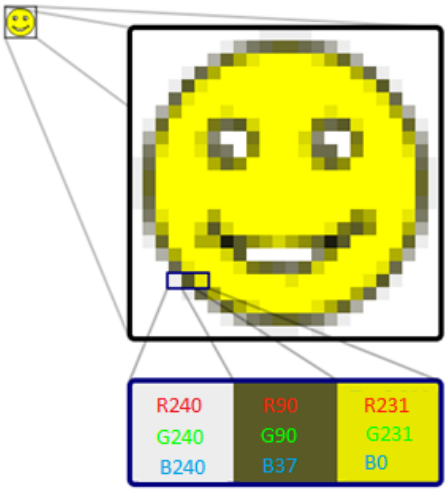


Figure 1.13 - Exemple de pixel sur une image matricielle

I . 5 . 3 . Image en niveau de gris

Une image est représentée par une matrice de dimension «nombre de lignes » x « nombre de colonnes ». Chaque pixel, représente l'intensité lumineuse comprise entre 0 et 255, soit 256 niveaux de gris. Le niveau de gris 0 correspond au noir tandis que 255 est représenté en blanc. Ci-dessous dans Fig. 1.14 sont représentés une image en niveau de gris de chromosome et des zooms d'une partie de cette Image.

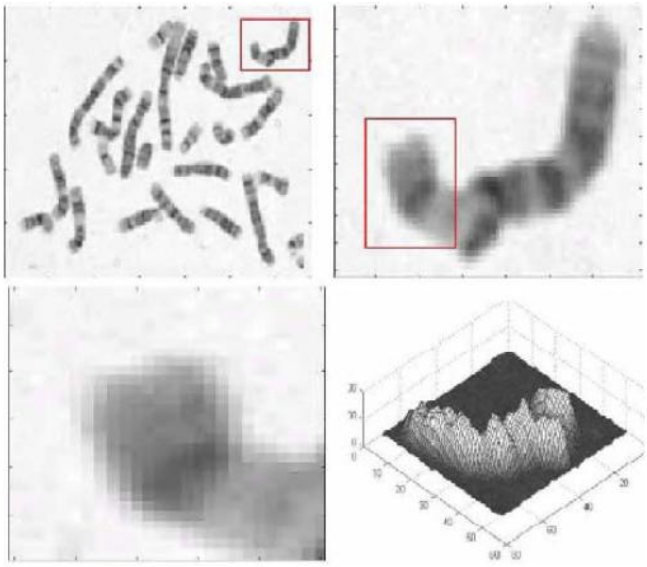


Figure 1.14 - Illustration d'une image en niveau de gris

On peut constater, en zoomant sur ces images l'effet de la discrétisation : des pixels carrés apparaissent. On peut également visualiser les images comme une surface 3D, les axes x et y représentant la position spatiale des images tandis que l'axe des z représente la luminance .

I . 6 . Les supports

I . 6 . 1 . Les fichiers images à format matriciel

Il existe un certain nombre de format de fichier image, comme les formats BMP, PNG, JPEG, TIFF, etc. Et ils sont, soient comprimé ou non comprimé.

Pour les formats non comprimés, le fichier image a une taille plus importante que tous les octets de l'image, car il comporte l'entête en plus. C'est le cas par exemple des formats BMP, PNG et TIFF (TIFF acceptent également le stockage avec compression).

Un même format peut accepter plusieurs possibilités de non-compression, ou de compression.

Deux types de compression sont possibles : compression sans perte et avec perte.

-sans perte : à l'issue de la décompression, l'image sera restituée à l'identique. C'est le cas, par exemple des formats BMP, PNG, GIF, TIFF.

-avec perte : à l'issue de la décompression, l'image sera légèrement dégradée. C'est le cas, par exemple des formats JPEG, JPEG-2000 TIFF.

La compression obtenue est plus importante avec un algorithme avec perte, mais la dégradation n'est généralement pas visible à l'œil, sauf pour des taux de compression importants. Ces algorithmes tirent profit des imperfections de l'œil.

Un fichier image comporte l'entête de l'image, c'est-à-dire les informations sur l'image : sa résolution, son type (Noir & Blanc ou Couleur), ainsi que l'information contenue par les pixels et peuvent aussi contenir d'autre information.

I . 6 . 2 . Les logiciels

Un certain nombre de logiciels permettent de visualiser, changer de format, traiter (traitement de bas niveau) les images : par exemple : IrfanView , the GIMP , Paint , Photoshop

D'autres permettent des traitements automatiques :

-la bibliothèque traitement d'image de MatLab, de LabView (National Instrument),

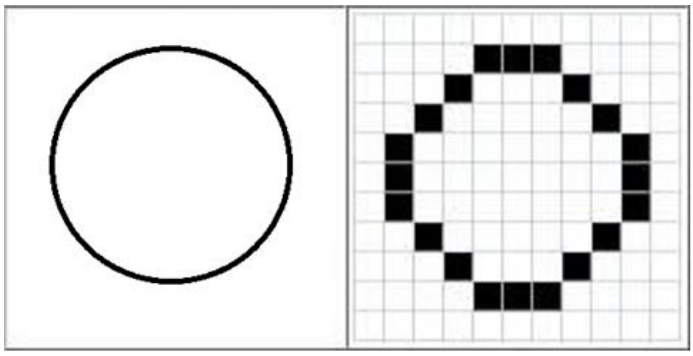
-les bibliothèques de vision industrielle : OMRON, ADEPT, COGNEX, etc

-les bibliothèques libres de vision : OpenCV , CLEOPATRE(22), EDUMEC. etc

CHAPITRE II
IMAGE VECTORIELLE

II . 1 . Définition

Une image vectorielle (ou image en mode trait), en informatique, est une image numérique composée d'objets géométriques individuels, des primitives géométriques (segments de droite, arcs de cercle, courbes de Bézier, polygones, etc.), définis chacun par différents attributs (forme, position, couleur, remplissage, visibilité, etc.) et auxquels on peut appliquer différentes transformations (homothéties, rotations, écrasement, mise à l'échelle, extrusion, inclinaison, effet miroir, dégradé déformés, morphage, symétrie, translation, interpolation, coniques ou bien les formes de révolution). Elle se différencie en cela des images matricielles (ou images bitmap), qui sont constituées de pixels comme montré dans les Fig.2.1. et Fig.2.2.



a) ***b)***

Figure 2. 1- Illustration d'une image vectorielle et d'une image matricielle,

a : Image vectorielle b : Image matricielle

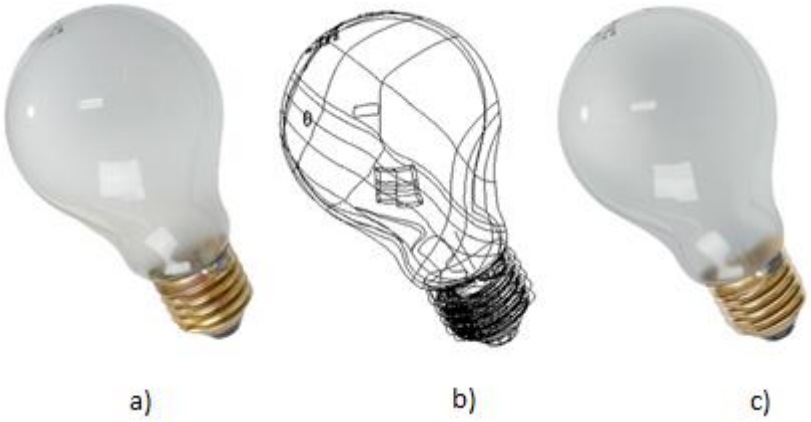


Figure 2. 2- Illustration d'une image vectorielle.

a)Image photo matricielle b)Image vectorielle c) Image rendue vectorielle.

II . 2 . Principe

L'image vectorielle est dépourvue de matrice de pixels. Elle est créée à partir d'équations mathématiques, un ensemble d'entités géométriques simples (carré, trapèze, ligne , ovale.etc.), et chaque forme dépend des paramètres (hauteur, largeur, couleur, épaisseur, rayon, etc.) et des attributs (forme, position, couleur, remplissage, visibilité, etc.), donnés à des vecteurs. Comme montré dans la Fig. 2.3.

Figure 2. 3 - Quelques formes d'Images vectorielles avec différents attributs.

II . 3 . Les différentes formes

Les différentes formes sont les possibilités offertes au graphiste pour réaliser une image vectorielle. Exemple : les segments (définie par deux points), les cercles (définie par son centre et son rayon), les polygones (définie par les coordonnées des sommets du polygone).

Mais ces formes géométriques ne suffisent pas à couvrir l'ensemble des formes naturelles. Ainsi les ellipses remplacent avantageusement un cercle et les courbes de Bézier permettent de reproduire des différentes formes de courbes rapidement.

II . 3 . 1 .Les ellipses

a . ***Les équations paramétriques***

La forme d'équation la plus "classique" est l'équation cartésienne Eq. 2.1.

$$y = f(x) \tag{2.1}$$

Mais les logiciels de dessins vectoriels utilisent les équations paramétriques qui se révèlent énormément plus pratique pour l'ellipse [10]. Elle se présente sous la forme Eq. 2.2 .

$$\begin{cases} x = f(t) \\ y = g(t) \quad \text{ou t est une variable.} \end{cases} \tag{2.2}$$

L'équation paramétrique du cercle Eq. 2.3, par exemple, est de la forme Eq. 2.2.

$$\begin{cases} x = x_0 + R\cos t \\ y = y_0 + R\sin t \end{cases} \quad (2.3)$$

Avec $O(x_0 ; y_0)$ centre du cercle et R le rayon du cercle.

L'Equation 2.3 se vérifie avec le cercle trigonométrique illustré dans la Fig. 2.4 suivante:

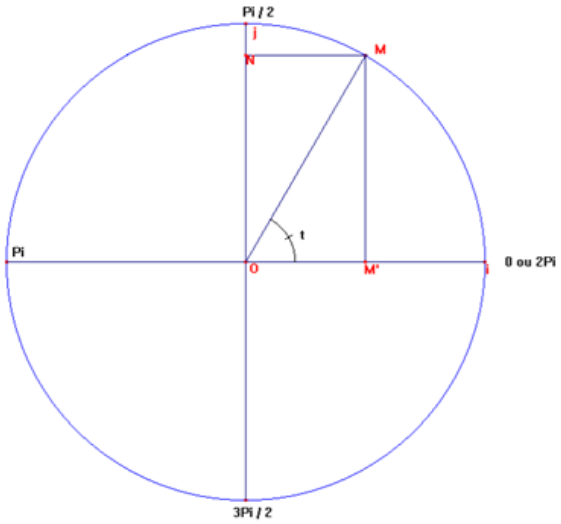


Figure 2. 4 - Représentation d'un cercle par l'équation paramétrique.

Avec $t = \pi/3$ on a :

$$Mx = OM' = Ox + R * \cos t = 0 + 1 * \cos\left(\frac{\pi}{3}\right) = \frac{1}{2} \quad (2.4)$$

$$My = ON' = Ox + R * \sin t = 0 + 1 * \sin\left(\frac{\pi}{3}\right) = \frac{\sqrt{3}}{2} \quad (2.5)$$

Alors Coordonnées de $M = (\frac{1}{2} ; \frac{\sqrt{3}}{2})$ ce qui nous montre que l'Eq.2.3 convient le plus au calcul des images vectorielles.

b . ***Équation paramétrique de l'ellipse***

La différence entre l'ellipse et le cercle est que l'ellipse a deux "rayons" : Le demi grand-axe en abscisse et le demi petit-axe en ordonnée, ce qui lui permet d'obtenir plus de formes autre que le cercle. L'équation paramétrique d'une ellipse de centre O et de demi grande-axe « a » et demi petit-axe « b » est présenté par l'Eq.2.6:

$$\begin{cases} x = x_O + a \cos t \\ y = y_O + b \sin t \end{cases} \quad (2.6)$$

La figure 2.5 représente une ellipse avec ses deux rayons a et b et son centre (x_o, y_o)

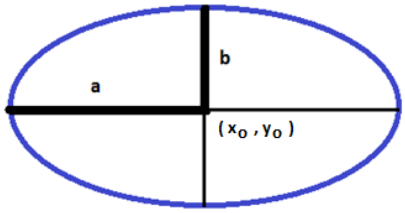


Figure 2. 5 - Les paramètres caractéristiques d'une ellipse.

II . 3 . 2 . Les courbes de Bézier

a . ***Principe***

Le but est de tracer une courbe plane par déplacement du barycentre d'une famille de points. Les points de base de cette famille sont appelés points de contrôle. Deux points définissent les limites de la courbe et n point la forme de la courbe:

b . ***Exemple d'une courbe de Bézier à trois points de contrôle.***

Comme illustré dans la Fig.2.6 , [10] Considérons 3 points de contrôle A; B; C ; t une variable entre 0 et 1.

B' barycentre de {(B; 1-t); (A; t)}. (2.7)

C' barycentre de {(B; 1-t); (C; t)}. (2.8)

M barycentre de {(B'; 1-t); (C'; t)}. (2.9)

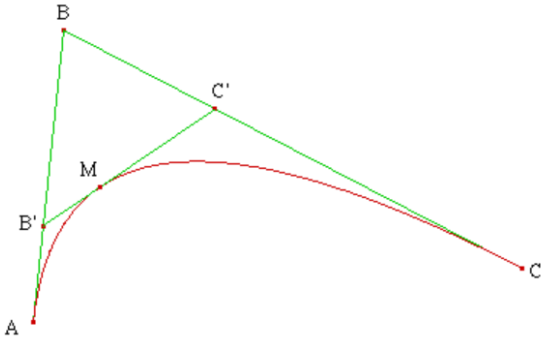


Figure 2. 6 - Illustration du principe de la courbe de Bézier.

On vérifie que la somme des coefficients est non nulle (égale à 1),

donc M est barycentre du système $\{(A; (1-t)^2); (B; 2t(1-t)); (C; t^2)\}$. (2.10)

En choisissant O comme origine on arrive à cette équation Eq. 2.11.

donc : $OM = (1 - t)^2.OA + 2t(1 - t).OB + t^2.OC$ (2.11)

Si O est choisi en A :

On a $AM = 2t(1 - t).AB + t^2.AC$ (2.12)

On peut supposer (A,AC,AB) orthonormé. Dans ce repère, on a :

$M(x,y)$ avec $x = t^2$, $y = 2t(1 - t)$ (2.13)

Ce qui donne la courbe dans la Fig. 2.7 suivante :

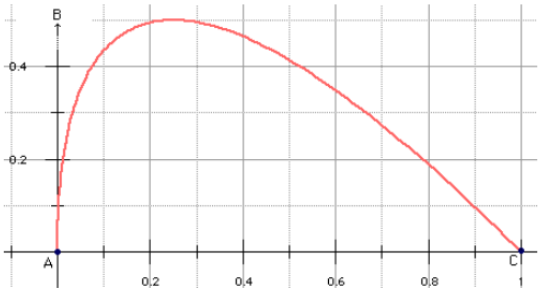


Figure 2. 7 - Tracée de la Courbe de Bézier.

On arrive donc à la courbe recherchée, courbe sur laquelle on peut appliquer diverses modifications (étirement, agrandissement...) en déplaçant l'un de ces points de contrôle.

On peut utiliser plusieurs points de contrôle pour obtenir les courbes de Bézier comme montre la Fig.2.8 suivante :

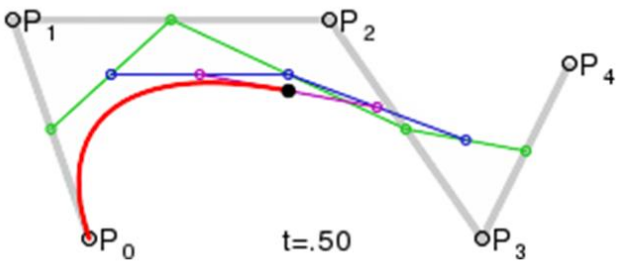


Figure 2. 8 - Exemple de traçage de Courbe de Bézier à 5 points de contrôle

c . *Un exemple d'enregistrement de dessin vectoriel par GEOPLAN*

On a le Code suivant extraite du logiciel GEOPLAN [10] permet de donner l'image vue à la Fig. 2.9.

Figure Géoplan
Numéro de version: 2
Position de Roxy: Xmin: -4.6274217586, Xmax: 5.3725782414, Ymax: 6.2220566319
Objet dessinable Roxy, particularités: rouge, non dessiné
O point libre
Objet libre O, paramètres: -0.58122205663, 1.8852459016
C cercle de centre O et de rayon 4 (unité Uoxy)
Objet dessinable C, particularités: bleu foncé
A point libre sur le cercle C
Objet libre A, paramètre: -0.2885873619
B point sur le cercle C, angle /ox: 3pi/2 (radian)
D point libre sur le cercle C
Objet libre D, paramètre: 2.6093132365
G centre de gravité du triangle ABD
C1 cercle inscrit dans le triangle ABD
Objet dessinable C1, particularités: jaune
O' centre du cercle C1
Segment [AB]
Segment [BD]
Segment [AD]
Objet dessinable [AD], particularités: rouge

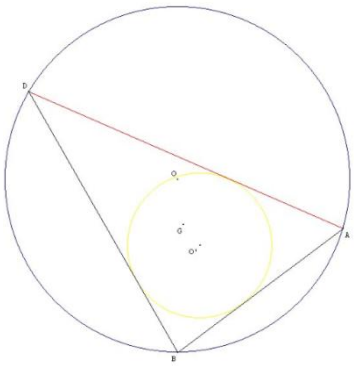


Figure 2. 9 - Un extrait d'image dessiné sur GEOPLAN

Remarque Dans le langage utilisé par GEOPLAN:

-Les deux premières lignes décrivent le logiciel.

-Les deux suivantes définissent le repère (Oxy), son domaine visible et s'il est dessiné ou non.

-Ensuite les objets (point, cercle, droite,etc.) sont décrits.

d . ***Exemple2 : code SVG (Scalable Vector Graphics) sur XML .***

La figure 2.10 montre un code permettant d'afficher un rectangle, un segment, un cercle et un texte sur XML .

Code XML

```
<?xml version="1.0" encoding="utf-8"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="300" height="200">
  <title>Exemple simple de figure SVG</title>
  <desc>
    Cette figure est constituée d'un rectangle,
    d'un segment de droite et d'un cercle.
  </desc>

  <rect width="100" height="80" x="0" y="70" fill="green" />
  <line x1="5" y1="5" x2="250" y2="95" stroke="red" />
  <circle cx="90" cy="80" r="50" fill="blue" />
  <text x="180" y="60">Un texte</text>
</svg>
```

Image correspondante

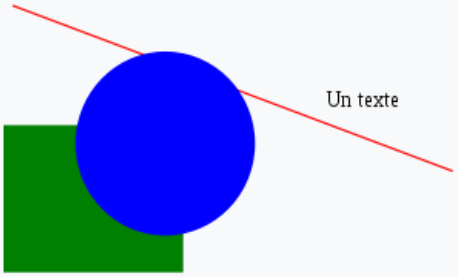


Figure 2. 10 - Exemple d'un code SVG pour représenter une image vectorielle.

Symboliquement pour l'image d'un cercle : seule la position du centre, la taille du rayon et ses informations de couleurs seront mémorisées.

II . 4 . Formats des fichiers

Les logiciels de dessin industriel utilisent souvent des images vectorielles, ainsi que les logiciels de traitement de texte ou de PAO (Publication Assistée par Ordinateur) (exemple : Illustrator, InDesign, Autocad, etc.).

Il existe de nombreux formats de fichiers vectoriels tels que : Postscript, PDF, Illustrator, Flash, CGM ou SVG. Le logiciel de DAO, Autocad a imposé ses formats de fichier DXF et DWG qui ont leurs propres caractéristiques.

II . 5 . Logiciels de dessin vectoriel

Il existe de nombreux logiciels de dessin vectoriel : qui peuvent être des logiciels libres, propriétaires ou open source. Mais tous ces logiciels ont tous ses propres caractéristiques qui ne sont pas toujours compatible entre eux, le Tableau III montre quelques types de logiciel.

Tableau III : Quelques Logiciels usuelles pour créer des images vectorielles

type	***Logiciels***	***Descriptif***
Open source	SVG	un format de données ASCII conçu pour décrire des ensembles de graphiques vectoriels et basé sur XML
propriétaire	ADOBE ILLUSTRATOR	Les images vectorielles sont constituées de courbes générées par des formules mathématiques
propriétaire	AFFINITY DESIGNER	un éditeur graphique vectoriel pour Apple macOS et Microsoft Windows
propriétaire	ADOBE FLASH	permettant la manipulation de graphiques vectoriels, d'images matricielles et de scripts
propriétaire	CORELDRAW	logiciel de création graphique vectorielle
Libre	INKSCAPE	logiciel professionnel de dessin vectoriel pour Windows, Mac OS X et GNU/Linux
Libre	DRAW LIBREOFFICE	module de dessin vectoriel pour schémas et illustrations simples
Libre	SKENCIL	logiciel libre de dessin vectoriel sous licence GNU GPL
propriétaire	AUTOCAD	Logiciel de dessin industriel
propriétaire	FREEHAND	logiciel de création graphique vectorielle 2D de Adobe
propriétaire	COREL POWERTRACE	Logiciel de transformation image matricielle en image vectorielle

II . 6 . Application à la cartographie

La représentation vectorielle est très utile pour l'établissement de cartes. En effet, le principe de couches, où différents plans se superposent, permet de superposer par exemple plusieurs informations comme les fleuves, les routes, le relief, etc. Les écritures peuvent être réduites et toujours lisibles avec précision. La propriété de changement d'échelle rapide et intacte de la représentation vectorielle correspond aux demandes de la cartographie. Ainsi l'utilisateur peut très vite observer avec précision et clarté la zone qui l'intéresse. Pendant très longtemps, les formats vectoriels cartographiques ne comprenaient que des droites, plus récemment, différentes approches des courbes ont commencé à être intégrées dans les logiciels SIG (Système d'Information Géographique). La figure 2.11 montre une image topographique c'est une image vectorielle avec des courbes.

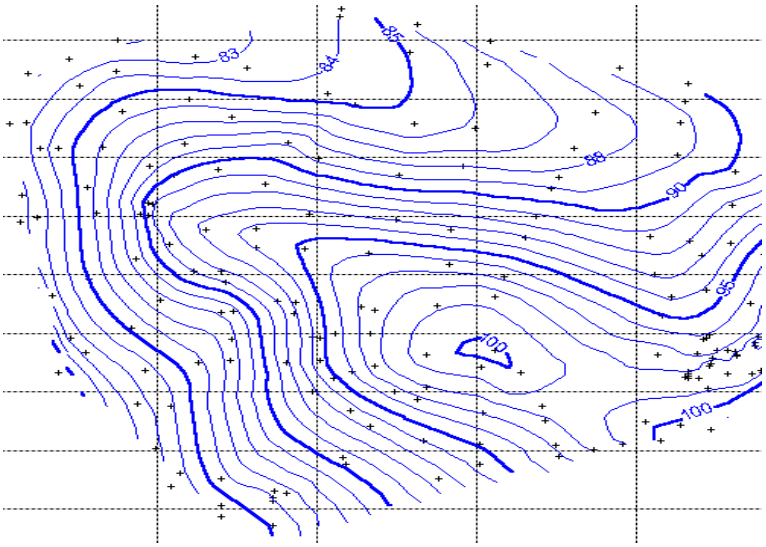


Figure 2. 11 - Utilisation en topographie, l'image vectorielle utilise des courbes,

II . 7 . Les polices de caractères True Type

Ces polices de caractères sont utilisées par windows mais sont compatibles avec d'autres environnements notamment le Mac. Elles sont décrites par des courbes quadratiques et enregistrées au format *.TTF (True Type Font).

Une fonction quadratique f est une fonction de la forme comme l'Eq.2.14

$$f(x) = ax^2 + bx + c \qquad (2.14)$$

où a, b et c sont des nombres réels et une égalité n'est pas à zéro. Le graphe de la fonction quadratique est appelé une parabole. Il s'agit d'une courbe en forme "U" qui peut être concave ou convexe selon le signe du coefficient a.

L'image vectorielle permet à la qualité de la police de ne pas être altéré par un changement de taille ce qui est différent des polices de caractère bitmap.

Comme montre la Fig2.12, comparaison entre une lettre A avec deux polices de caractère différents, l'une bitmap « Courrier » et l'autre une police vectorielle « Times new roman » sous le Bloc notes de windows.

Ci-dessous dans la Fig.2.12 les deux lettres A en taille 12 avec deux polices différentes

A :police vectorielle (Times new roman)

A : police bitmap (courrier)

Figure 2. 12- Comparaison entre deux images du lettre A même taille 12 mais avec deux polices de caractères différentes.

Remarque :

Les qualités de la lettre A sont similaires avec les deux polices de caractères différents, la différence n'est pas remarquable à l'œil nue.

Et ci-dessous dans la Fig. 2.13 les deux lettres A en taille 100 .


: police vectorielle (Times new roman)

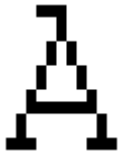

: police bitmap (courrier)

Figure 2. 13- Comparaison entre deux images du lettre A taille 100 mais avec deux polices de caractères différentes.

Remarque :

Du crénelage apparaît sur la police bitmap alors que la police vectorielle reste de bonne qualité. C'est pour cela que de nombreux logiciels de traitement de texte (à partir de word 2000) utilisent uniquement les polices vectorielles.

II . 8 . Comparaison des images matricielles avec les images vectorielles

On peut résumer dans les tableaux IV et V les différences entre ces deux formats.

Tableau IV : Différence entre image matricielle et image vectorielle en caractéristique.

Image matricielle	Image vectorielle
Le codage est par pixel qui divise l'image en ligne et colonne et constitue une matrice.	Le codage est vectoriel, qui code les images par des éléments géométriques, et des formules mathématiques avec des attributs.
L'image est déjà pré-dessiner	la courbe est codée par du texte (une équation) puis est retranscrite en image par une calculatrice,
Le volume de l'image dépend de sa taille	Le volume de l'image dépend seulement de sa complexité (nombre de points, complexité des courbes.etc.).
Quand on agrandi l'échelle on voit les pixels qui induit une perte de qualité visible , Comme montré sur la Fig. 2.14.a	L'image peut être redimensionnée sans perdre en qualité. Comme montré sur la Fig. 2.14.b

Figure 2. 14- Illustration d'une image bitmap a) agrandie avec de crénelage et une image vectorielle b) agrandie à l'infini (pas d'effet de pixellisation – crénelage)

Tableau V : différence entre image matricielle et image vectorielle en termes d'exploitations.

La superposition des images en couche n'est pas très pratique.	Le format vectoriel gère la superposition de couches (ou calques), c'est à dire la superposition d'entités géométriques.
les images matricielles sont plus appropriées que les images vectorielles aux travaux sur photographies ou sur photos réalistes	il est aujourd'hui impossible en pratique d'obtenir une image vectorielle à partir d'une photo mais elles sont généralement employées afin de réaliser des tracés précis, des cartes, des logos ou des illustrations.
Beaucoup de compatibilités sur tous les logiciels	Chaque format de fichier et logiciels vectoriels possédant ses propres attributs, la compatibilité entre les formats est difficile. Le format vectoriel n'est pas reconnu par les navigateurs internet et par certains logiciels multimédias. Il faut avoir un logiciel ou un plug-in comme Flash.
Une image bitmap très complexe comme une photographie ne peut être converti en image vectorielle, ses formes n'étant pas du tout géométriques. Les logiciels proposant une telle conversion nécessite souvent l'utilisation de l'utilisateur.	. Si les images vectorielles ne sortent pas de leur programme ou d'un type de programme pouvant gérer l'affichage de ses images, leurs caractéristiques restent ainsi, sinon elles sont transformées en images matricielles pour les afficher .
Seul type qui soit exploitable par la carte graphique pour le moment.	Une image vectorielle ne peut pas être affichée directement sur un écran. Elle doit auparavant être transformée en image de type bitmap , Tous les logiciels qui produisent des images vectorielles (notamment les logiciels de dessin industriel) réalisent cette transformation.

CHAPITRE III

APPROXIMATION D'UN GRAPHE PAR UNE COURBE A PARAMETRE VECTORIELLE

III . 1 .Introduction

Il y a plusieurs méthodes pour approcher un graphe par une courbe. Il y a, par exemple l'approximation polynômiale qui est souvent utilisé du fait que les polynômes font partie des fonctions les plus simples que l'on rencontre en analyse , mais dans cette recherche, nous allons utiliser l'approximation d'un graphe par composition de plusieurs arcs (caractérisé par deux vecteurs) .

III . 2 .Courbe caractérisée par deux vecteurs

III . 2 . 1 . Définition

Soit une portion de courbe (C) représentée par deux vecteurs Va et Vb . La figure .3.1 montre les deux vecteurs paramètres de la forme de l'arc (C) .

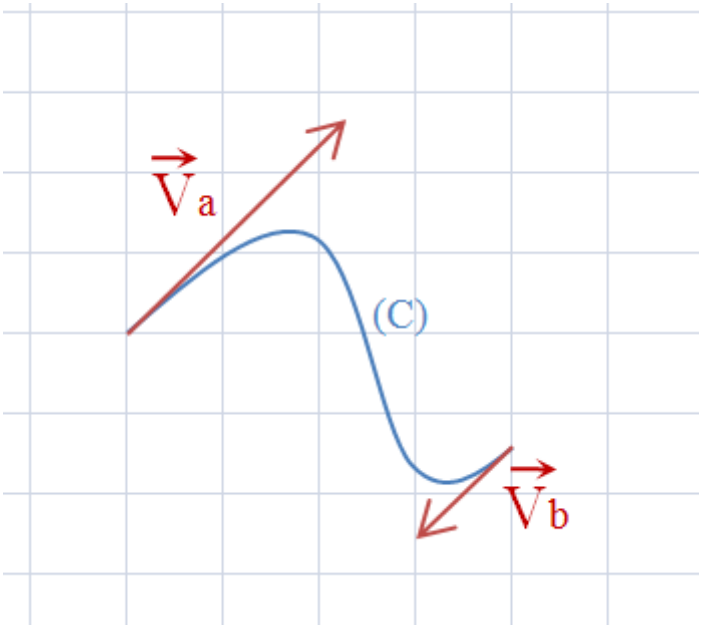


Figure 3.1- Une courbe avec les deux vecteurs de transformation

Ces deux vecteurs sont caractérisés par ses points d'application respectifs (xa,ya) et (xb,yb) ses modules Va et Vb et ses directions par l'angle Θa et Θb comme montre la Fig.3.2.

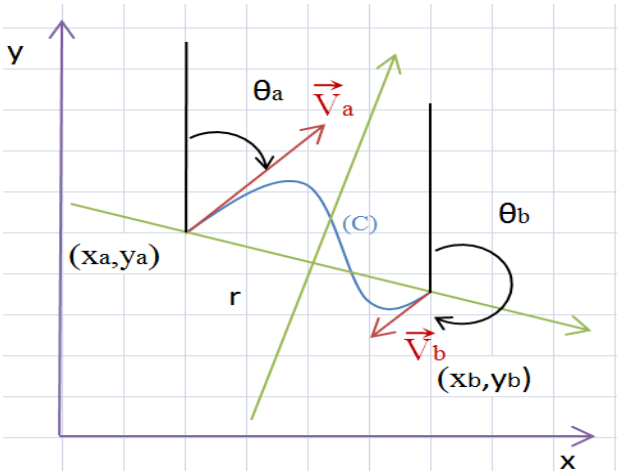


Figure 3.2 - Montre les paramètres des 2 vecteurs qui représentent la forme de l'arc.

Soit (d) la distance entre les deux points d'application des deux vecteurs,

La forme de la courbe (C) dépend ainsi de ces deux vecteurs Va et Vb, et peut être approché à l'aide d'un arc.

III . 2 . 2 . Transformation d'un graphe

Nous avons pris comme base de graphe la courbe Demi-cercle présentée par la fonction

$$y = \sqrt{r^2 - x^2} \qquad (3.1)$$

Et on applique à ces deux extrémités les deux vecteurs de transformation distant de d dans l'Eq.3.2.

$$d = 2r \qquad (3.2)$$

III . 2 . 3 . Transformation verticale suivant l'axe Y

La transformation verticale suivant l'axe Y est obtenue par modification des modules de Va et Vb. Un point M dans la courbe dans la Fig.3.3 ayant les coordonnées (x , y), après la transformation devient le point M' avec ses coordonnées (x' , y') .

On remarque que x' = x, après la transformation quelque soit la valeur des vecteurs Va et Vb. Ainsi on a l'Eq.3.3.

$$x = x' \qquad (3.3)$$

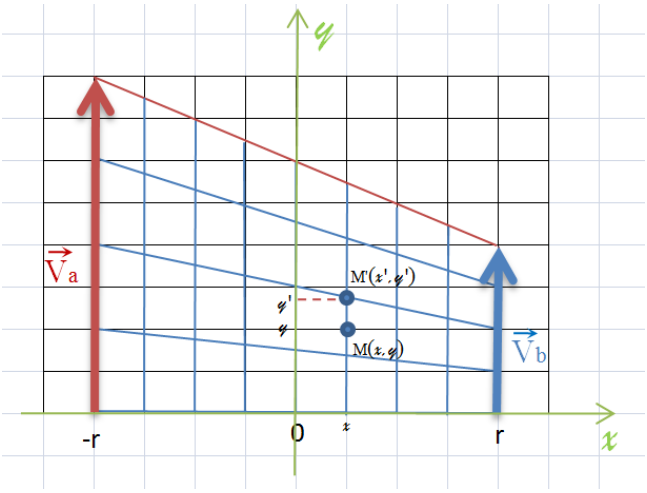


Figure 3.3 - Déplacement du point M après transformation

la droite D(x), est une droite qui présente le taux de translation du point M vers M' en fonction de Va , Vb et de x ,qui est représentée dans la Fig.3.4 suivante .

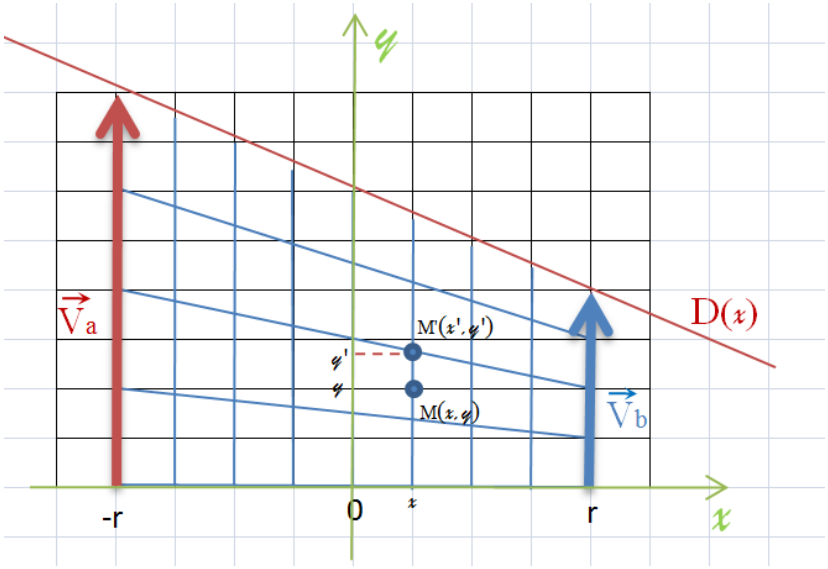


Figure 3.4 - Droite correspond au taux de translation D(x)

a. Determination de la droite $D(x)$

La droite $D(x)$ est une droite linéaire de la forme $D(x) = \alpha x + \beta$ (3.4)

Dont on a l' Eq.3.5 après calcul des constantes $\alpha \; et \; \beta$

$$D(x) = (\frac{Vb-Va}{2r})x + \frac{Va+Vb}{2} \quad (3.5)$$

b. Translation du point M en M’

Soit le point $M'(x', y')$ l’image du point $M(x, y)$ après l’application des deux vecteurs de transformation Va et Vb .

Apres application à l’équation (3.1) $y = \sqrt{r^2 - x^2}$ on a l’Eq.3.6, qui détermine l’équation de l’arc fonction de x , Va et Vb , r est obtenue par la distance entres les deux points A et B points d’applications respectifs des deux vecteurs .

L’équation finale :

$$y' = \left(\left(\frac{Vb-Va}{2r^2}\right)x + \frac{Va+Vb}{2r}\right)\left(\sqrt{r^2 - x^2}\right) \qquad (3.6)$$

III . 2 . 4 . Transformation du courbe par rotation

L’angle Θa est l’ange de rotation du vecteur Va par rapport a l’axe des y et l’angle Θb du vecteur Vb par rapport a l’axe des y .

Le point $M''(x'', y'')$ représenté dans la Fig.3.6 a pour image $M'(x', y')$ après la transformation (soit rotation d’angle $\theta x'$)

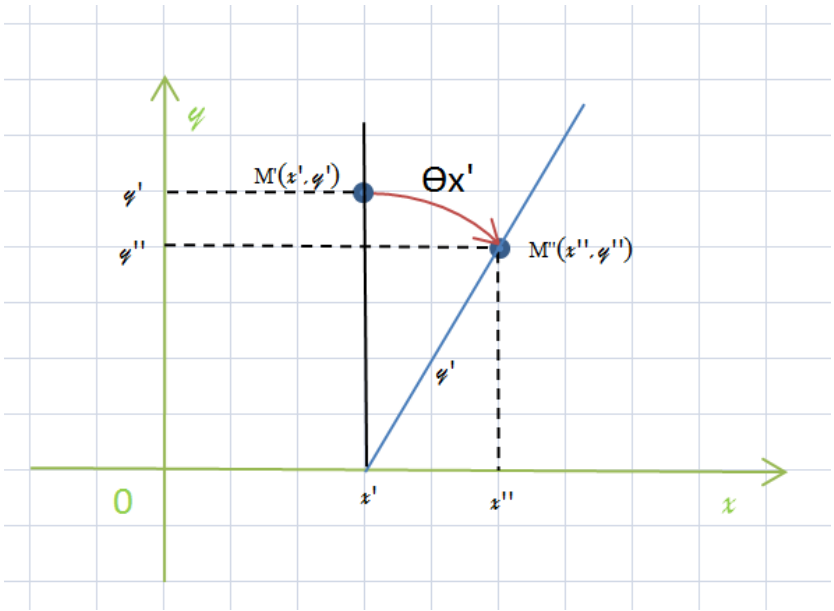


Figure 3.5 - Montre le déplacement de M' vers M'' par rotation $\boldsymbol{\theta x'}$

On a l'Eq.3.7 qui détermine cette transformation.

$$\begin{cases} y'' = y'cos(\theta x') \\ x'' = (x' + y'\sin(\theta x') \end{cases} \quad (3.7)$$

a. Détermination de l'angle $\theta x'$

L'angle θx′ varie linéairement suivant x proportionnelle à la distance et en fonction des deux angles θa et θb , comme montre la Fig.3.7

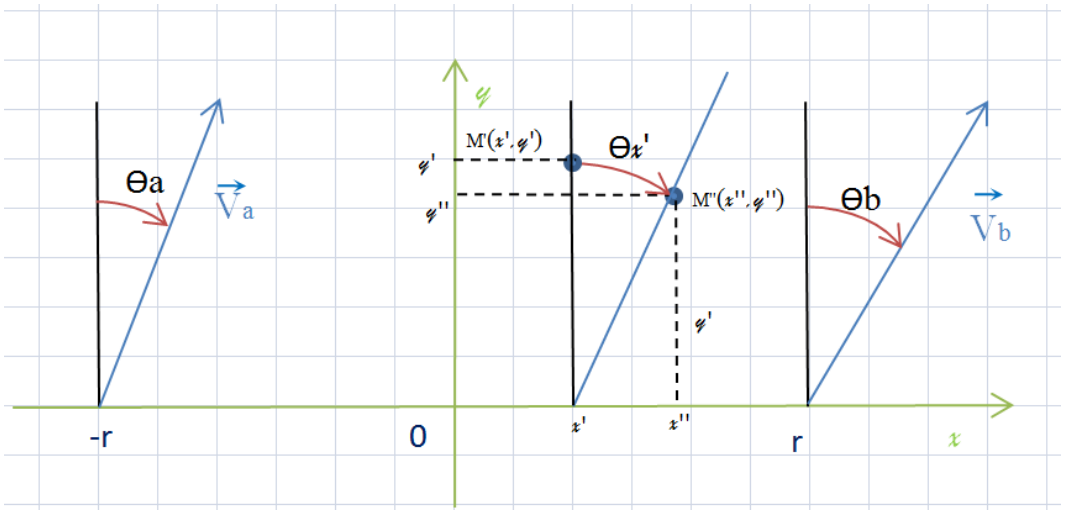


Figure 3.6 - Les rotations $\boldsymbol{\theta a}$ *et* $\boldsymbol{\theta b}$

Et la figure.3.8 représente la variation de $\theta x'$ suivant x

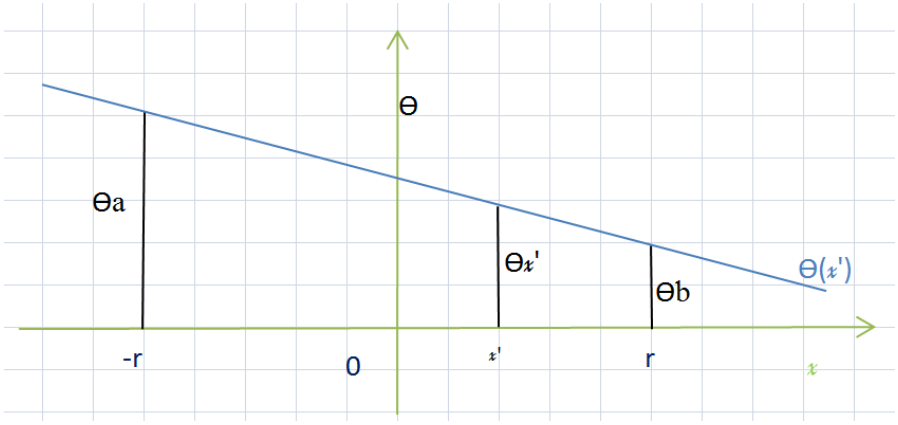


Figure 3.7- L'angle rotation en fonction de x

D'où on a l'Eq.3.7 suivante représente la variation d'angle d'un point suivant x

$$\theta(x) = \left(\frac{\theta b - \theta a}{2r}\right) x + \frac{\theta b + \theta a}{2} \tag{3.7}$$

III . 2 . 5 . L'équation finale

Apres combinaison de ces deux transformation, on a l'Eq.3.8 qui est une fonction représente un arc en fonction des paramètres des deux vecteurs . (Va,Vb, θa, θb, r).

$$\begin{cases} y'' = \left(\left(\frac{Vb - Va}{2r^2}\right) x + \frac{Va + Vb}{2r}\right)\left(\sqrt{r^2 - x^2}\right) cos\left(\left(\frac{\theta b - \theta a}{2r}\right) x + \frac{\theta b + \theta a}{2}\right) \\ x'' = \left(x + \left(\left(\frac{Vb - Va}{2r^2}\right) x + \frac{Va + Vb}{2r}\right)\left(\sqrt{r^2 - x^2}\right) \sin\left(\left(\frac{\theta b - \theta a}{2r}\right) x + \frac{\theta b + \theta a}{2}\right)\right. \end{cases} \tag{3.8}$$

III . 3 .Simulation sur MATLAB

Nous avons créé Sur Matlab un programme pour tracer l'arc qui est déterminé par les paramètres des deux vecteurs caractéristiques.

L'arc est tracée en utilisant la fonction plot (x'',y'') avec x'' et y'' en fonction de x en variant x de (–r) à (r).

III . 3 . 1 . Les résultats

En variant Va et Vb, et l'angle Thetaa et Thetab avec des valeurs caractéristiques on a de différentes formes de courbe caractéristiques dans l' ANNEXE II. Les portions de code du programme peuvent être vues dans l'annexe III,

CHAPITRE IV

CREATION D'UNE IMAGE VECTORIELLE A PARTIR D'UNE IMAGE MATRICIELLE

Dans ce chapitre, nous allons utiliser Matlab afin de créer des scripts pour transformer une image matricielle en image vectorielle.

IV . 1 . Le logiciel MATLAB

C'est un logiciel de simulation Développé par la société The MathWorks . Il permet de réaliser des simulations numériques basées sur des algorithmes d'analyse numérique, avec des fonctions mathématiques. Ce qui inclus la manipulation des matrices, l'affichage des courbes et des données, la mise en œuvre des algorithmes, la création des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran.

Le logiciel MATLAB est construit autour du langage MATLAB. Une interface en ligne de commande, qui est un des éléments du bureau MATLAB, permet d'exécuter des commandes simples. Des séquences de commandes peuvent être sauvegardées dans un fichier texte, typiquement avec l'éditeur MATLAB, sous la forme d'un « script » ou encapsulées dans une fonction.

IV . 2 . Algorithme de création d'une image vectorielle à partir d'une image matricielle.

On a créé un algorithme qui a pour but de créer une image vectorielle à partir des images matricielles de format JPEG ou JPG ou BMP.

Le type d'image source est un image de plan ou carte, contenant essentiellement des courbes , ainsi le principe est de tracer un ensemble d'arcs qui correspond à l'image source, et ces arcs sont caractérisé par des vecteurs dont ces derniers sont calculer à partir de la position de chaque pixels qui constitue l'image source.

La création de ces vecteurs est un calcul approché expliqué dans le chapitre précédent. La figure 4.1 représente l'algorithme correspondant.

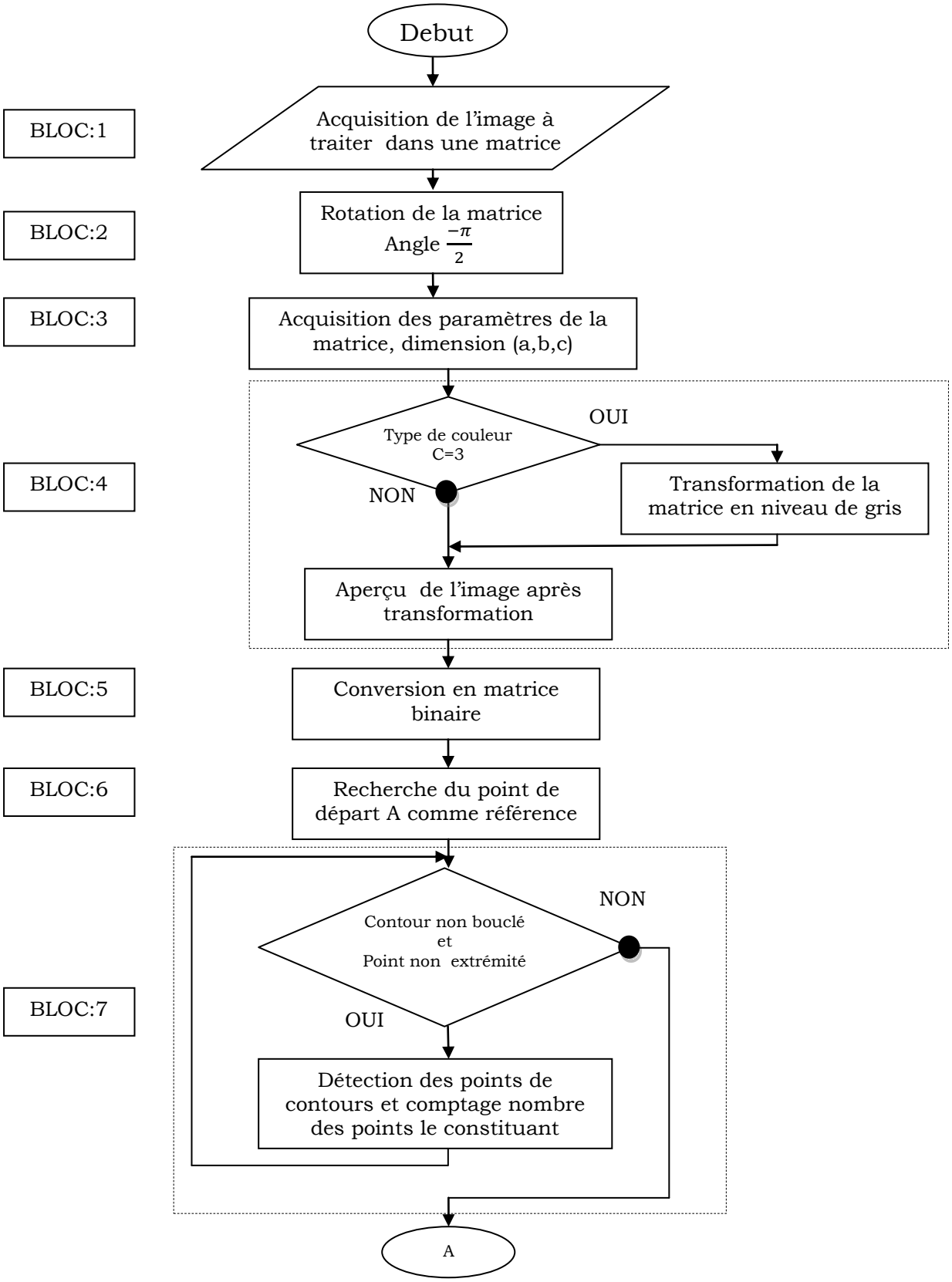


Figure 4.1 - 1er Partie de l'algorithme de création d'image vectorielle

Le Bloc 10, 11, 12 dans la Fig. 4.2 représente la création de l'image vectorielle.

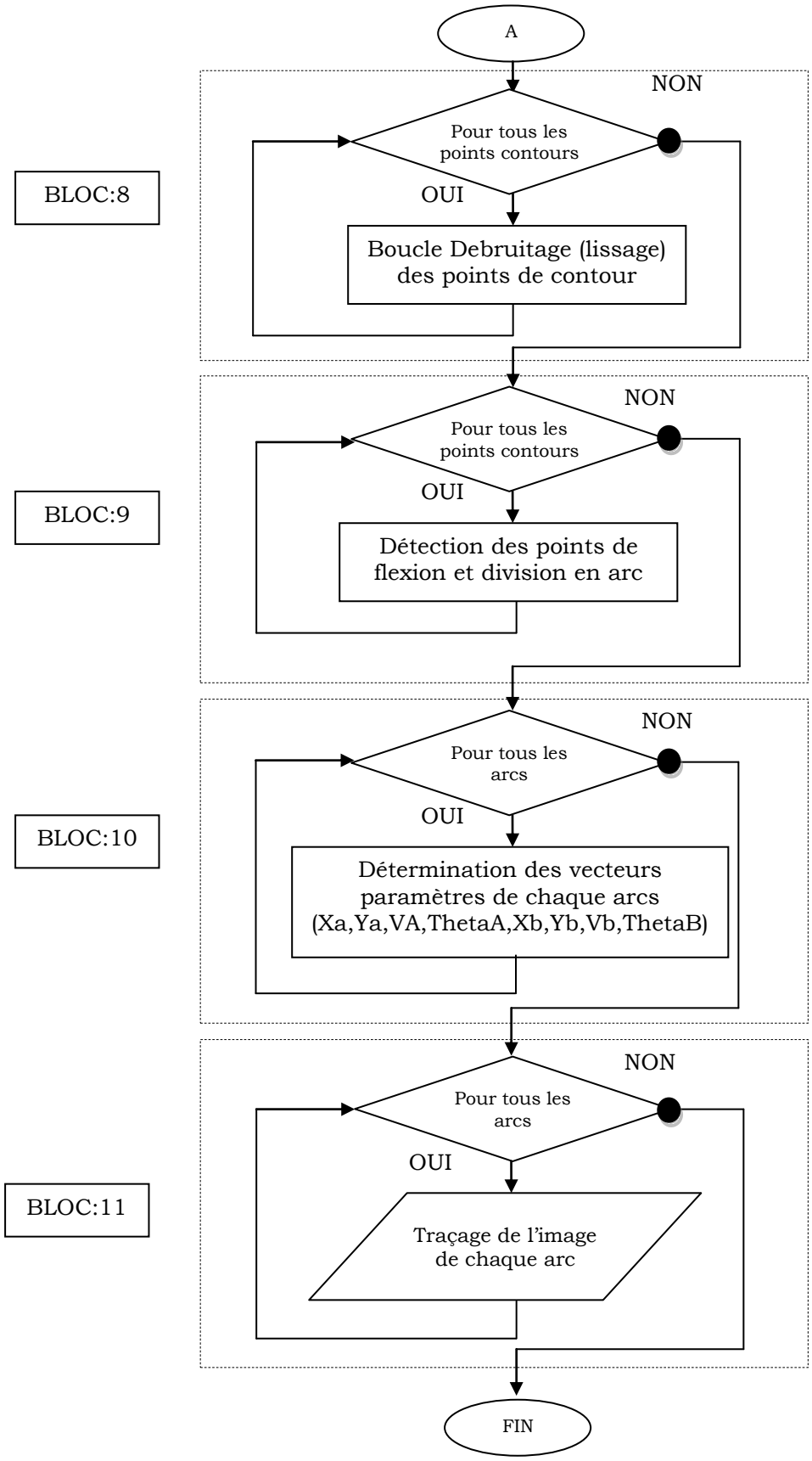


Figure 4.2 - 2eme Partie de l'algorithme de création d'image vectorielle

Chaque bloc de l'algorithme est représenté par une fonction afin de séquencer chaque étage de calcul.

IV . 2 . 1 .BLOC 1 : Acquisition de l'image à traiter dans une Matrice

Dans ce bloc on enregistre le fichier image dans une matrice, pour être traité dans le programme, en utilisant la fonction MATLAB « imread » , on peut lire les fichier image de format type jpg, jpeg ou bmp et mètre dans une matrice les valeurs et les paramètres correspond à l'image .

Pour Simuler nous avons utilisé une image en format JPG de carte de Madagascar représenté dans Fig.4.3.

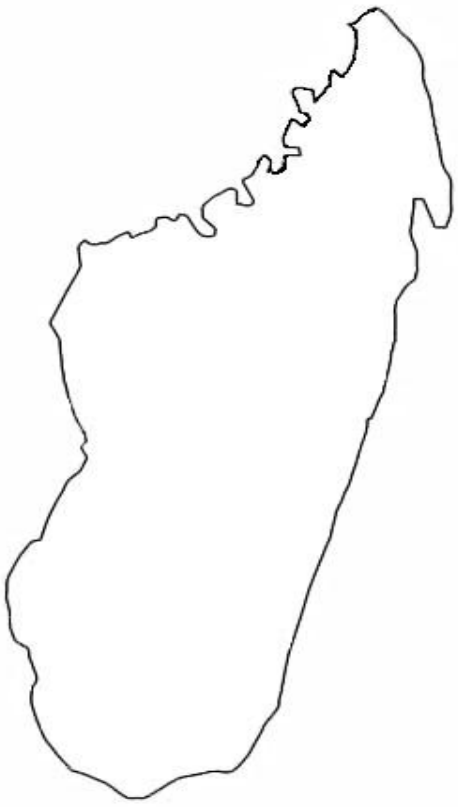
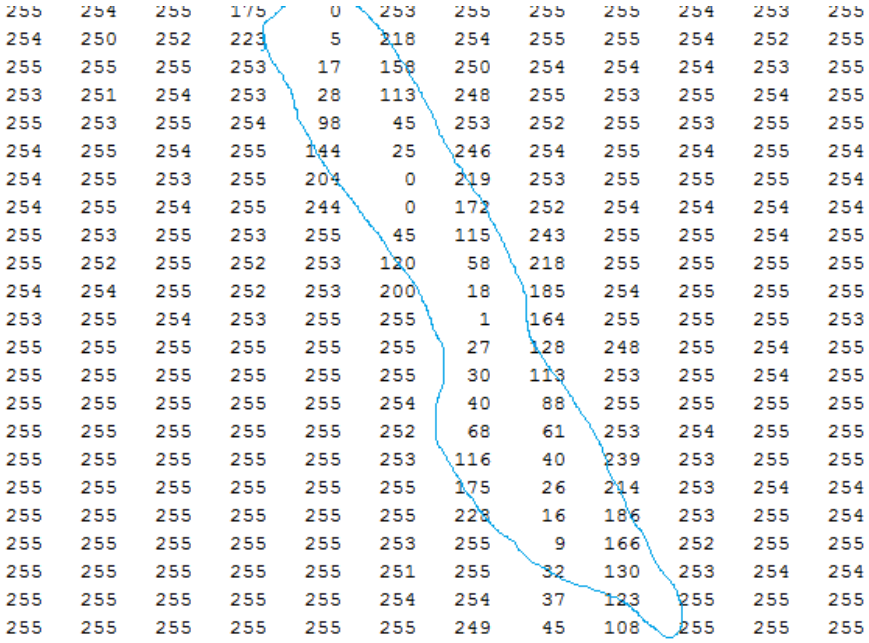


Figure 4.3 - Image en format JPG de carte de Madagascar

Apres utilisation de la fonction « imread » nous avons une matrice avec des valeurs de 0 à 255, comme montre la Fig 4.4 qui représente une portion des valeurs de la matrice. Les valeurs dans la zone bleu représente les tracés de la courbe (couleur noir), les valeurs approximatifs de 255 sont des couleurs blancs .



255	254	255	175	0	253	255	255	255	254	253	255
254	250	252	223	5	218	254	255	255	254	252	255
255	255	255	253	17	158	250	254	254	254	253	255
253	251	254	253	28	113	248	255	253	255	254	255
255	253	255	254	98	45	253	252	255	253	255	255
254	255	254	255	144	25	246	254	255	254	255	254
254	255	253	255	204	0	219	253	255	255	255	254
254	255	254	255	244	0	172	252	254	254	254	254
255	253	255	253	255	45	115	243	255	255	254	255
255	252	255	252	253	120	58	218	255	255	255	255
254	254	255	252	253	200	18	185	254	255	255	255
253	255	254	253	255	255	1	164	255	255	255	253
255	255	255	255	255	255	27	128	248	255	254	255
255	255	255	255	255	255	30	113	253	255	254	255
255	255	255	255	255	254	40	88	255	255	255	255
255	255	255	255	255	252	68	61	253	254	255	255
255	255	255	255	255	253	116	40	239	253	255	255
255	255	255	255	255	255	175	26	214	253	254	254
255	255	255	255	255	255	228	16	186	253	255	254
255	255	255	255	255	253	255	9	166	252	255	255
255	255	255	255	255	251	255	32	130	253	254	254
255	255	255	255	255	254	254	37	123	255	255	255
255	255	255	255	255	255	249	45	108	255	255	255

Figure 4.4 - Echantillon de valeurs de la matrice image

IV . 2 . 2 .BLOC 2 : Rotation de la matrice

Alors que l'image matricielle est codée dans une matrice avec ligne i et colonne j , et que l'élément de la Matrice indice (i=1,j=1) correspond au pixel le plus haut à gauche de l'image .

L'image vectorielle tracée sur un repère orthonormé (0,x,y) de Coordonnées (x,y) correspond à l'image matricielle tournée de angle $-\frac{\pi}{2}$ si on garde les indices comme coordonnées . Pour avoir l'image vectorielle à la bonne position il faudrait tourner l'image matricielle source de $\frac{\pi}{2}$.

La fonction MATLAB « rot90 » tourne les valeurs dans une matrice à un angle de 90^{o} vers le sens antihoraire. L'utilisation de cette fonction nécessite son application 3 fois pour avoir la rotation de l'image source de $\frac{\pi}{2}$, afin de garder les indices et de permettre un traçage de l'image vectorielle à la bonne position .

La Figure .4.5 illustre la représentation d'une même image, la Fig.4.5 a) représente l'image matricielle avec les indices i et j commencent du haut à gauche de l'image . et la Fig.4.5 b) représente l'image vectorielle tracée sur un repère oxy, les coordonnées x et y commencent en bas à gauche de l'image.

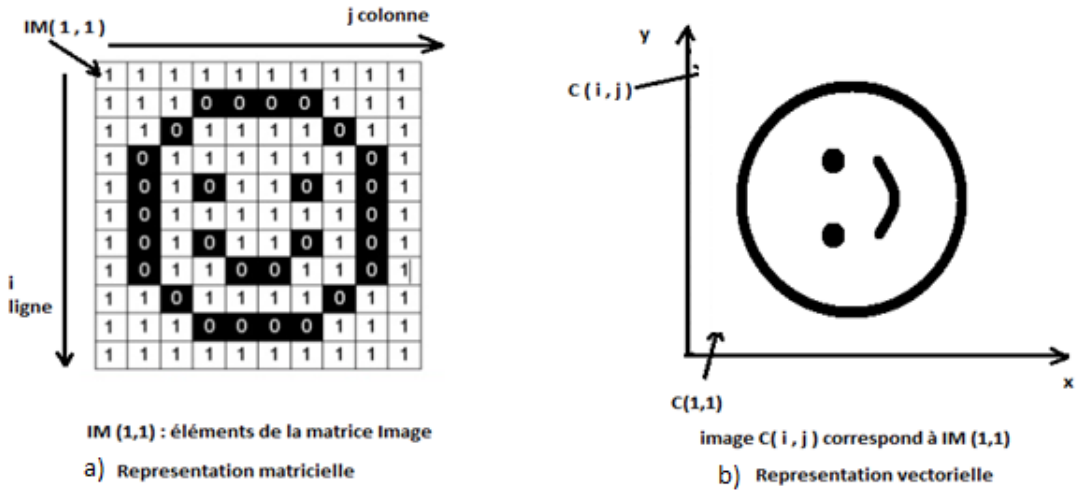


Figure 4.5 - Illustration de la différence entre les indices dans différentes repères.

Apres la rotation 90^{o} on a la Fig.4.6 l'image de carte de Madagascar suivante.

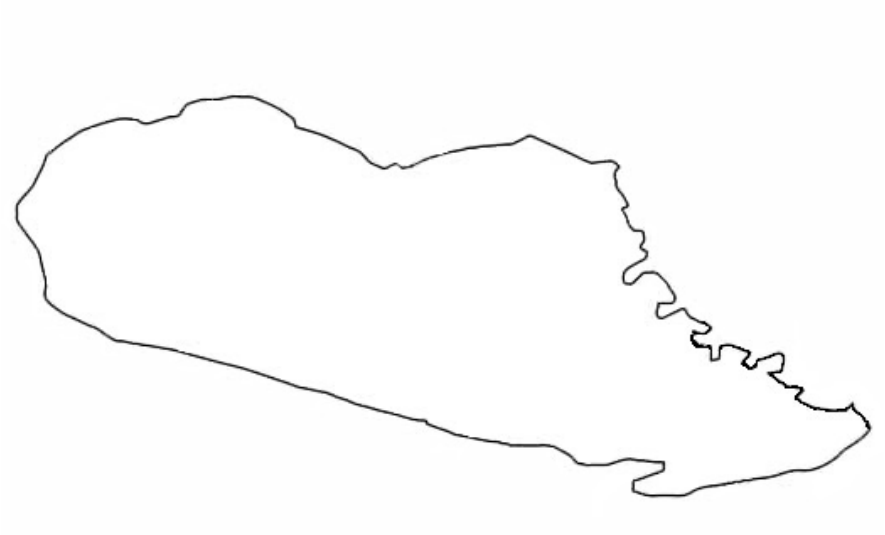


Figure 4.6 - Représentation de L'image source après utilisation 3 fois de la fonction rot90

IV . 2 . 3 . BLOC 3 : Acquisition des paramètres de la matrice

Dans ce bloc on prend les paramètres de la matrice plus précisément les dimensions de l'image que l'on mets dans des variables (a,b,c)

a : la hauteur de la matrice (nombre de ligne),

b : la largeur de la matrice (nombre de colonne),

c : Type de l'image (1 : a niveau de gris et 3 : couleur RGB)

Le code :

```
[a b c]=size(Imatrice);
```

Avec notre image carte de Madagascar nous avons les résultats suivants :

```
a =

   356

b =

   528

c =

     3
```

Notre image est une image couleur (c=3) de taille 356 x 528 soit 187968 pixels

IV . 2 . 4 .BLOC 4 : Transformation de la matrice en niveau de gris

Pour la pré-transformation, si l'image est une image couleur (c =3) il faudrait transformer la matrice image en niveau de gris (Hmatrice). En utilisant la fonction Matlab « rgb2gray» , après transformation de notre image de carte de Madagascar nous avons les valeurs de notre matrice en niveau de gris, on remarque que les pixels sont représenté par seulement un valeur qui est de 0 a 255 .

Un pixel de i ligne et j colonne de la matrice image couleur est présenté par 3 valeurs suivants :

```
Imatrice(i,j,1)  ; Imatrice(i,j,2)  ; Imatrice(i,j,3)
```

Mais après la transformation, la matrice qui représente l'image en niveau de gris n'a qu'une seule valeur.

```
Hmatrice (i,j)
```

IV . 2 . 5 .BLOC 5 : Conversion en matrice binaire

La conversion en matrice binaire est importante pour éliminer les pixels parasites, son efficacité dépend du paramètre du seuil que l'on utilise pour la conversion.

Le seuil détermine si il faut mètre 1 ou 0 .

La fonction correspondent et l'Eq.4.1 suivant :

$$\mathbf{g(x,y)} = \begin{cases} \mathbf{1 \text{ si } f(x,y) \geq T} \\ \mathbf{0 \text{ sinon}} \end{cases} \qquad (4.1)$$

Avec g(x, y) l'élément de la matrice binaire et f(x, y) la matrice image en niveau de gris et T le seuil.

Apres Transformation nous avons une matrice binaire, la Fig.4.7 représente une portion de ce matrice binaire , on remarque qu'il n'y a que des valeurs 0 et 1 dans toutes la matrice . Les valeurs 0 représentent la couleur noire et les valeurs 1 représentent la couleur blanche des pixels.

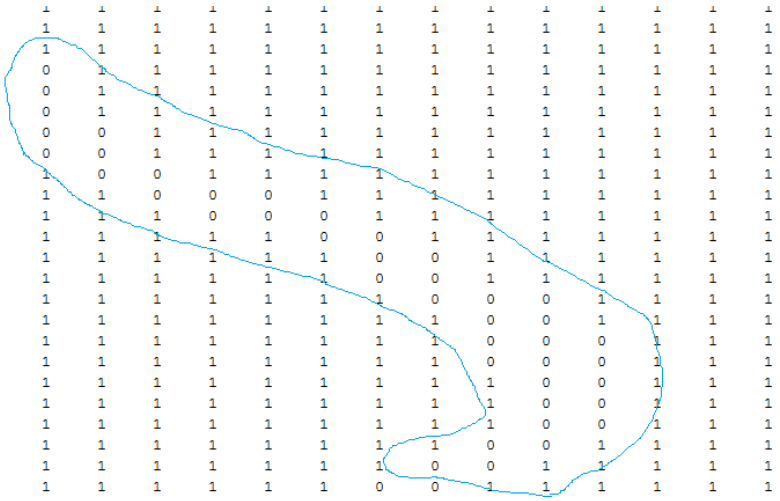


Figure 4.7 -. Représentation d'une portion de la matrice binaire

IV . 2 . 6 .BLOC 6 : Recherche du point de départ A comme référence

Dans ce bloc on recherche un point initial de départ que l'on va utiliser comme référence de traçage du graphe.

C'est le premier point extrémité du premier arc. A partir de ce point on calcule les directions de courbes, pour déterminer les points de flexions.

La figure 4.8 représente le point de départ sur notre image carte de Madagascar. C'est le point le plus haut (déterminé par la boucle de recherche).

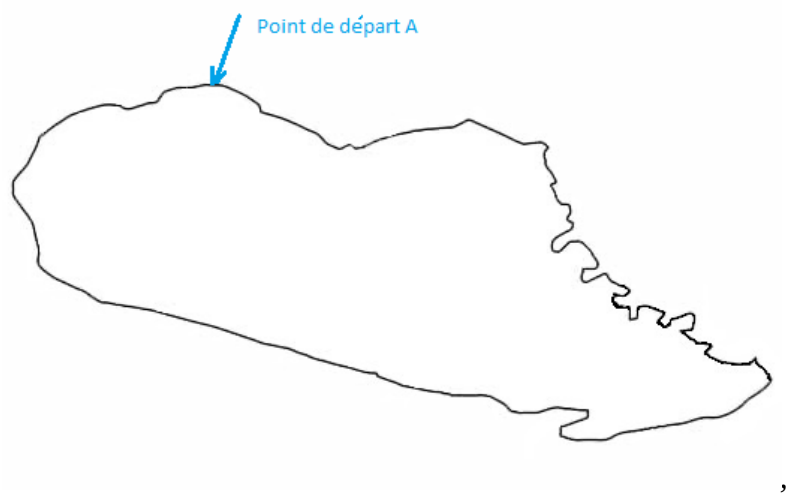


Figure 4.8 - Point de départ ou point de référence A

IV . 2 . 7 .<u>BLOC 7 : Détection des points de contours et comptage nombre des points le constituant</u>

Ce bloc consiste à détecter tous les points de contour qui illustre le graphe. Les points seront enregistrés dans une matrice avec un indice qui détermine leur rang respectif ainsi que leur nombre. On enregistre aussi l'angle de déviation par rapport au point précédent.

Le principe est de faire une boucle de recherche de pixel voisin selon le sens de l'aiguille d'une montre, comme illustrer à la Fig.4.9. La boucle de recherche du point suivant ne s'arrête pas jusqu'à ce qu'elle ne trouve plus aucun point voisin ou s'il revient sur un point déjà passé.

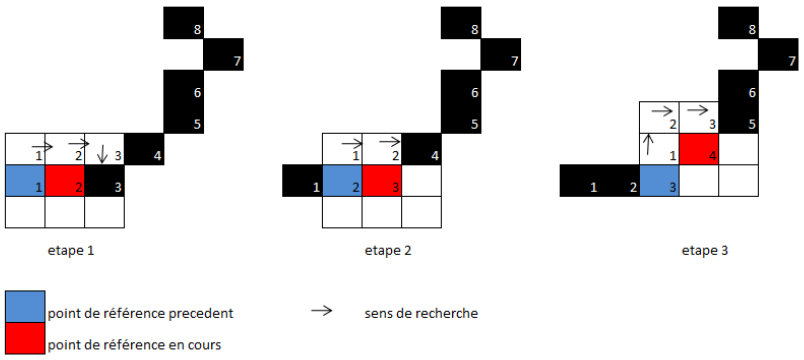


Figure 4.9 - Illustration du sens de recherche du point voisin

IV . 2 . 8 .BLOC 8 : Boucle Débruitage (lissage) des points de contour

Alors qu'une image matricielle est constituée de pixel, il existe souvent des parasites qui dérangent la qualité du traitement. On va éliminer dans cette boucle les pixels ayant une déviation de forme -1. 1. -1 ou 1.-1.1 .

Les déviations sont le sens de rotations de position des points voisins par rapports aux points précédents, la direction est négatives si le sens est sens horaire et positive s'il est de sens antihoraire.

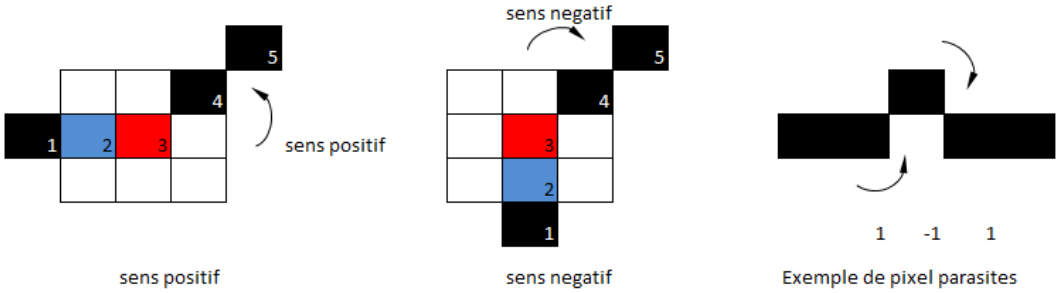


Figure 4.10 - Illustration de la direction sur chaque point

IV . 2 . 9 .BLOC 9 : Détection des points de flexion et division en arc

On peut déterminer les points de flexion en analysant les variations de la direction du point de contour à chaque point. Un point de flexion est le point là où sa direction change. On obtient un arc à partir de deux points de flexions consécutives.

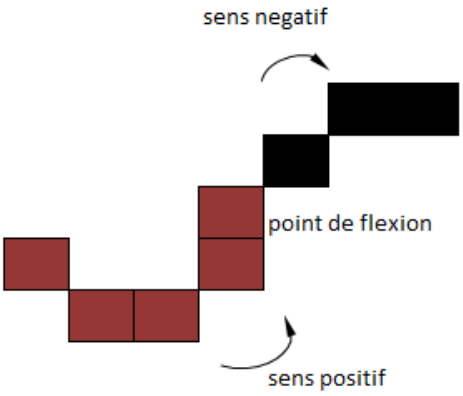
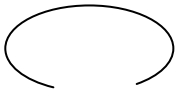
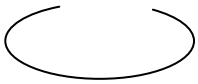


Figure 4.11- Illustration du point de flexion ou extrémité des arcs

On enregistre tous ces points de flexions dans une matrice pour déterminer ensuite les points extrémités de chaque arc.

Les formes d'arcs probables sont présenté dans le tableau VI et toutes ces formes sont caractérisées par la différence entre les deux angles tangente à chaque extrémité qui sont illustré dans la Fig.4.12.

Tableau VI : Les formes probables d'un arc

Forme de l'arc	Valeur de l'angle $\Delta\theta$
	$[\,0 \quad \pi\,]$
	$[\,\pi \quad 3\pi/4\,]$
	$[\,3\pi/4 \quad 2\pi\,]$
	$[\,0 \quad -\pi\,]$
	$[\,-\pi \quad -3\pi/4\,]$
	$[\,-\frac{3\pi}{4} \quad -2\pi\,]$

La figure 4.12 représente l'angle $\Delta\theta$ qui caractérise la forme de l'arc.

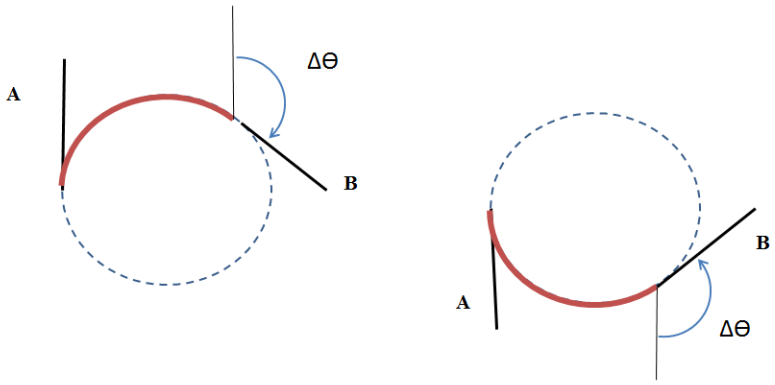


Figure 4.12- L'angle qui caractérise la forme des arcs

IV . 2 . 10 . BLOC 10 : Détermination des vecteurs paramètres de chaque arc

Toutes les arcs sont caractérisé par les vecteurs de position (Xa,Xb,Ya,Yb) et les vecteurs de transformation (Va, θa ,Vb, θb).

Les vecteurs de position sont obtenus par les coordonnées des points extrémités de chaque arc. Les angles sont obtenus par les directions du point voisin aux extrémités.Et les modules des vecteurs de transformation sont obtenues après calcul suivant la forme de l'arc.

IV . 2 . 11 . BLOC 11 : Traçage de l'image de chaque arc

Apres toutes les paramètres qui déterminent les arcs sont reçues on trace un à un toutes les arcs sur le même plan.

La figure 4.13 représente le résultat après réalisation du programme en transformant un dessin de carte de Madagasikara en format JPG .

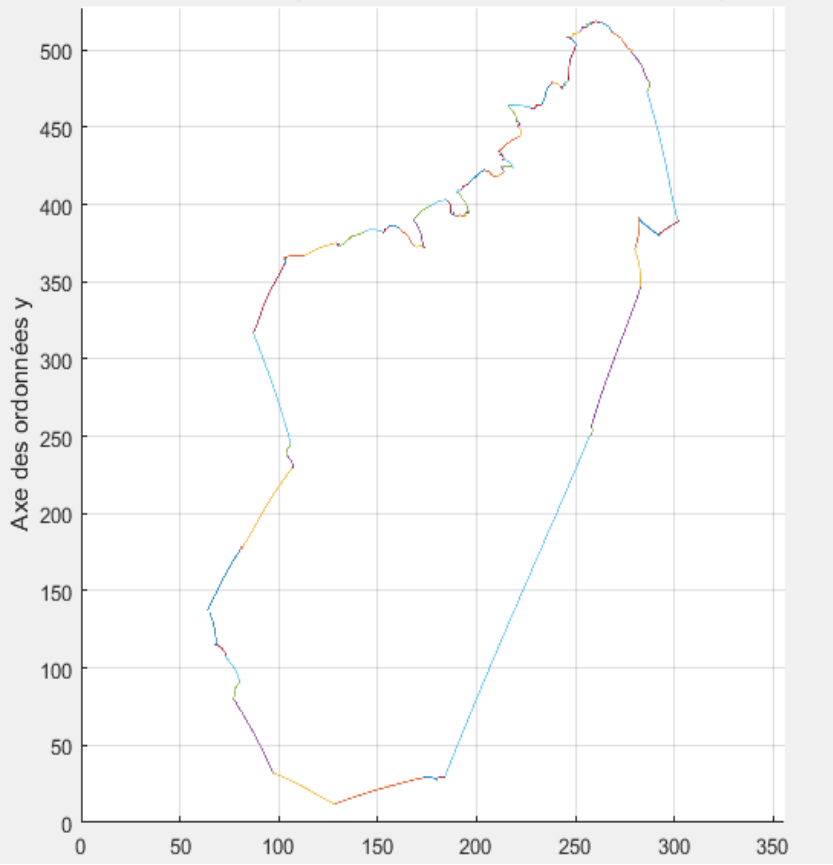


Figure 4.13-Figure de l'Image vectorielle résultat tracé sur MATLAB

Remarque : Et en utilisant le zoom sur le graphe on ne voit plus les pixels, La figure.4.14 montre un zoom sur une partie de l'image , l'image est bien précis.

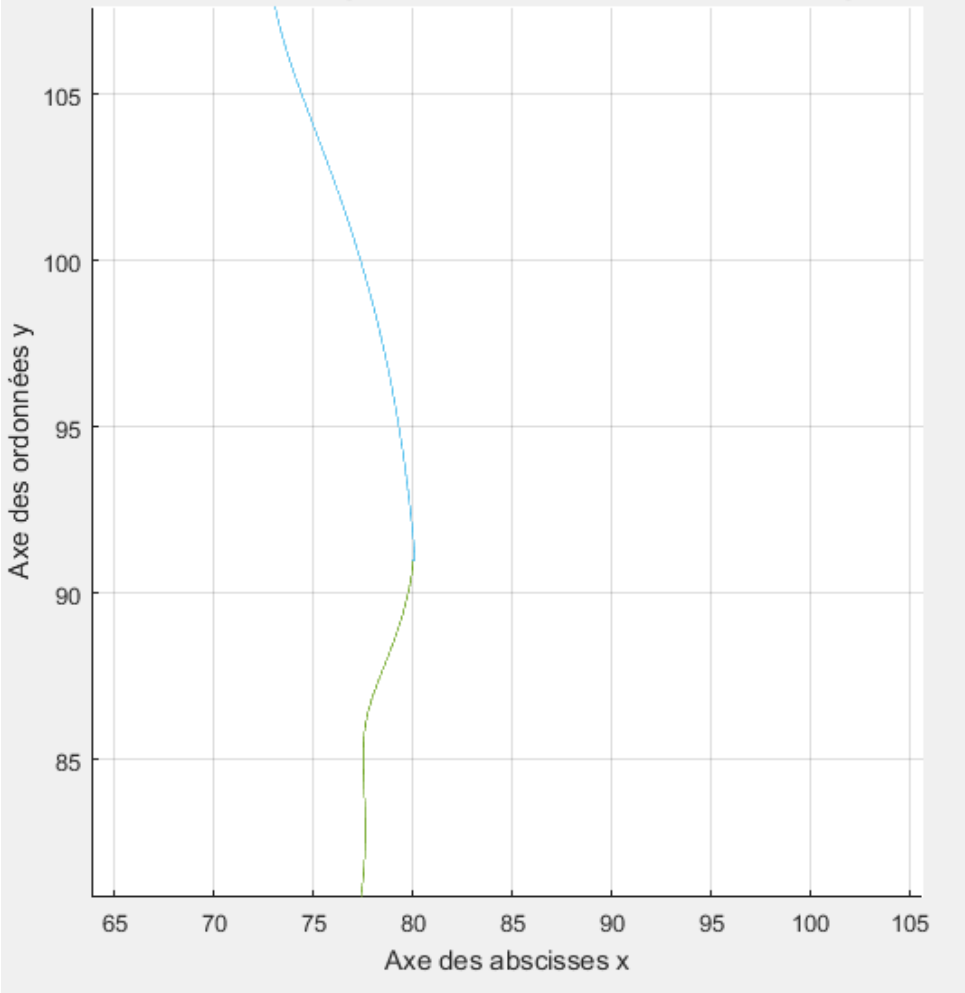


Figure 4.14- Figure qui montre qu'il n'y a pas de crénelage sur l' image vectorielle obtenue quand on l'agrandie.

IV . 3 . Format vectorielle de la carte de Madagascar

Nous avons enregistré les paramètres des vecteurs qui représentent chaque arcs dans une matrice nommée (arcx) de la forme suivante : arcx [Xa ,Ya,Va, θa, Xb , Yb ,Vb, θb]

Avec Xa, Ya, Xb, Yb coordonnées des extrémités et Va, θa, Vb, θb sont les paramètres qui détermine la forme de l'arc. L'image vectorielle de la carte de Madagascar que nous avons créé contient 78 arcs, ainsi nous avons une matrice de 78 x 8 qui contient les données de l'image vectorielle tandis que l'image matricielle de départ est de 528 x 356 pixels.

CONCLUSION

L'utilisation des images et des photos est très évoluée actuellement, grâce à l'avancement des technologies de l'électronique et de l'informatique. Les images vectorielles sont encore un champ pas encore bien exploré.

Les images matricielles sont encore les plus utilisées grâce à sa qualité de présenter directement l'image, car toutes les cellules de captures et d'affichages sont encore en format matriciel. Le format matriciel étant une forme de représentation d'image relativement facile à manipuler par les outils informatiques.

La caractéristique avantageuse des images vectorielles est que l'on peut avoir des images très précises, mais il faut toujours les dessiner manuellement. Bien que l'exploitation de ces images nécessitent des algorithmes complexes, la précision des informations contenues dans ces dernières est un plus indéniable pour l'analyse d'image.

La conversion d'image matricielle en image vectorielle apporterait plus de précision dans les processus d'analyses d'image. Notre algorithme de conversion est fonctionnel pour des images matricielles contenant des figures géométriques. Le programme réaliser sous Matlab est relativement rapide vue que nous représentons l'image vectorielle obtenue à l'aide de suite de segments d'arcs.

Bien que notre algorithme soit fonctionnel, on peut l'affiner en lui permettant de traiter des images matricielles complexes, c'est-à-dire des images contenant des formes en superposition ou même des textures. La recherche d'autres formes de représentation d'image vectorielle autre que des arcs pourrait aussi être envisagée pour faire suites aux études réalisées dans ces travaux.

ANNEXE I
TRANSFORMATION GEOMETRIQUE D'UNE GRAPHE

I . 1 . Introduction

Comme une image vectorielle est une image numérique composée d'objets géométriques individuels, des primitives géométriques (segments de droite, arcs de cercle, courbes de Bézier, polygones, etc.) .

On va chercher à utiliser un ensemble d'arcs pour représenter notre image vectorielle.

Chaque arc est caractérisé par une fonction initiale, sa position et les transformations géométriques appliquées.

Alors, c'est la transformation géométrique caractérisé par des vecteurs qui détermine la forme d'une portion de l'image.

Il existe plusieurs transformations pour un graphe . la majorité de ces dernières seront détaillée dans cette partie. Pour ce faire, la fonction suivante

Prenons $f(x) = x^2$.

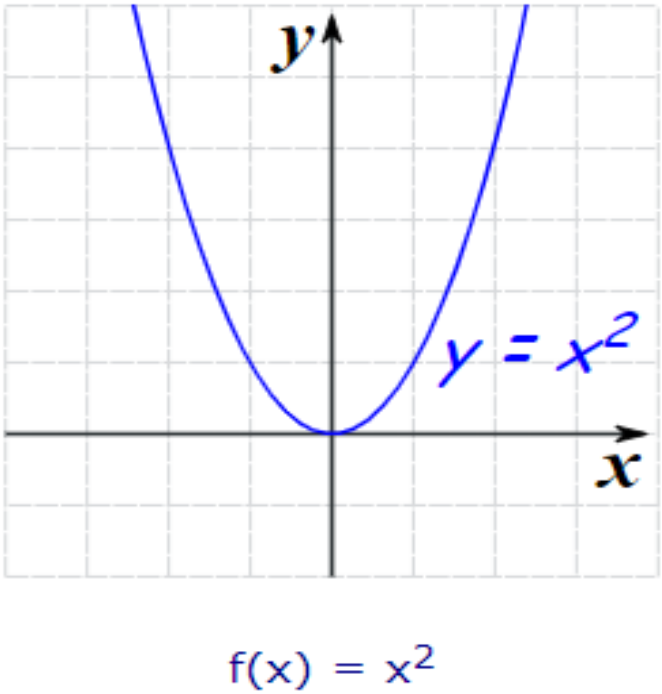
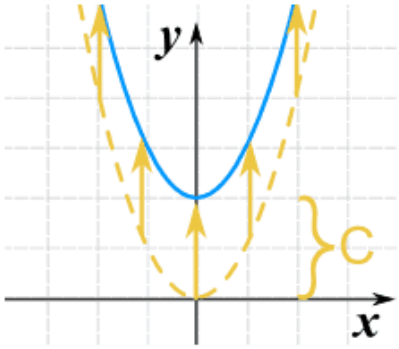


Figure A . 1 - forme de la fonction f (x) = x2.

I . 2 . Déplacement suivant Y

En ajoutant une constante à la valeur de y , la courbe se déplacent suivant l'axe y en conservant sa forme



$g(x) = x^2 + C$

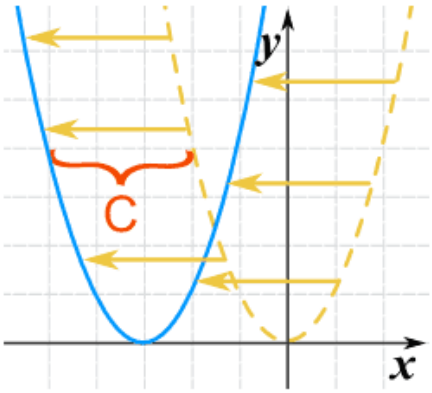
Figure A . 2- Illustration du Déplacement suivant l'axe Y

Sens de déplacement suivant la valeur de C :

- C > 0 déplacement suivant le sens de l'axe y .
- C < 0 déplacement contraire au sens de l'axe y.

2-Déplacement suivant X

En ajoutant une constante à la valeur de x , la courbe se déplacent suivant l'axe x en conservant sa forme



$g(x) = (x+C)^2$

Figure A . 3- Illustration du Déplacement suivant l'axe X

Sens de déplacement suivant la valeur de C :

- C> 0 déplacement contraire au sens de l'axe x.
- C <0 déplacement suivant le sens de l'axe y.

I . 3 . Compression et étirement suivant Y

La compression ou l'étirement dans la direction y est obtenue en multipliant la fonction entière par une constante.

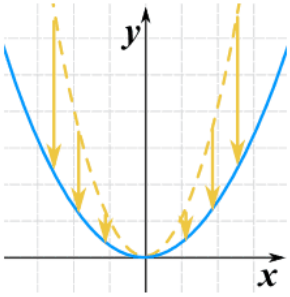


Figure A . 4- Etirement suivant l'axe Y

Transformation suivant la valeur de C :

- C> 1 Etirement
- 0 <C <1 Compression

I . 4 . Compression et étirement suivant X

La compression ou l'étirement dans la direction y est obtenue en multipliant x par une constante.

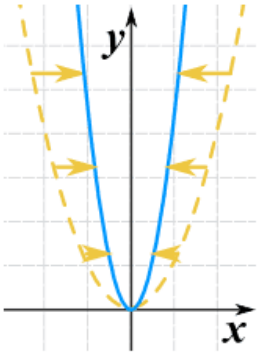


Figure A . 5- Compression ou étirement suivant l'axe X

Transformation suivant la valeur de C :

- C> 1 Compression
- 0 <C <1 Etirement

Contrairement à la direction y, des valeurs plus grandes entraînent plus de compression.

I . 5 . Réflexion suivant X

La réflexion suivant l'axe x est obtenue en multipliant la fonction entière par −1:

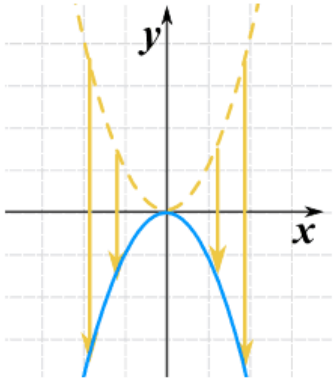


Figure A . 6- Réflexion suivant X

Ceci est aussi appelé réflexion sur l'axe des x (axe où y = 0)

Nous pouvons combiner une valeur négative avec une mise à l'échelle:

Exemple: multiplier par -2 le retournera et l'étirera dans la direction y.

I . 6 . Réflexion suivant Y

La reflexion suivant l'axe y est obtenue en multipliant la valeur x par −1:

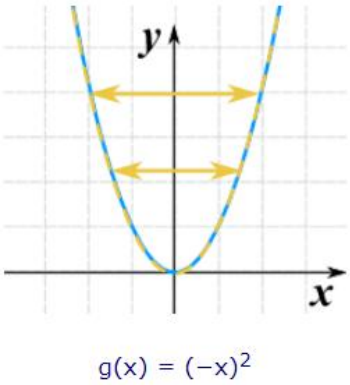


Figure A . 7- Réflexion suivant Y

Si la fonction est déjà symétrique par rapport à l'axe des ordonnées, le graphe reste le même.

Voici un autre exemple utilisant √ (x):

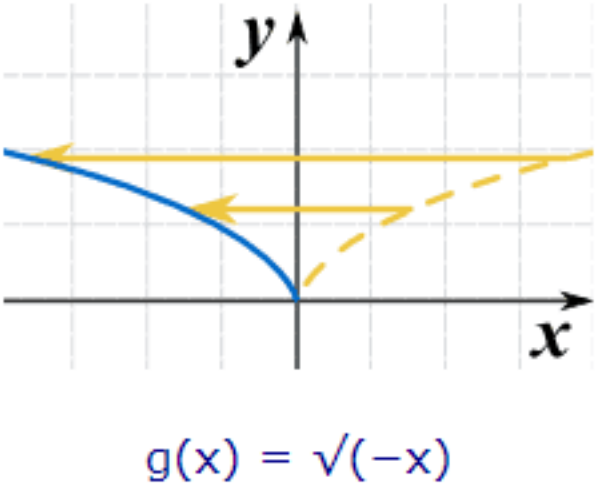


Figure A . 8- Exemple Réflexion suivant Y d'une fonction

Ceci est aussi appelé réflexion sur l'axe des y (l'axe où x = 0)

Tableau A - I – Résumé de la formule de transformation d'un graphe

y = f (x) + C	C> 0 déplacement suivant le sens de l'axe y C <0 déplacement contraire au sens de l'axe y.
y = f (x + C)	C> 0 déplacement contraire au sens de l'axe x. C <0 déplacement suivant le sens de l'axe x
y = Cf (x)	C> 1 Etirement dans la direction y 0 <C <1 Compression dans la direction y
y = f (Cx)	C> 1 Etirement dans la direction x 0 <C <1 Compression dans la direction x
y = -f (x)	Réflexion sur l'axe des x
y = f (-x)	Réflexion sur l'axe des y

Nous pouvons combiner toutes les transformations en utilisant ceci:

$$g(x) = af\big(b(x + c)\big) + d$$

.a constante pour l'étirement / compression dans la direction y

- | a | > 1 , étirement
- | a | <1 , compression
- a <0 Réflexion sur l'axe des x , (retourne le graphique à l'envers)

.b constante pour l'étirement / compression dans la direction y

| b | > 1 compression

| b | <1 étirement

b <0 Réflexion sur l'axe des y, (fait basculer le graphe de gauche à droite)

.c constante pour le déplacement suivant l'axe x

c <0 déplacement suivant le sens de l'axe x

c> 0 déplacement contraire le sens de l'axe x

.d est le décalage vertical

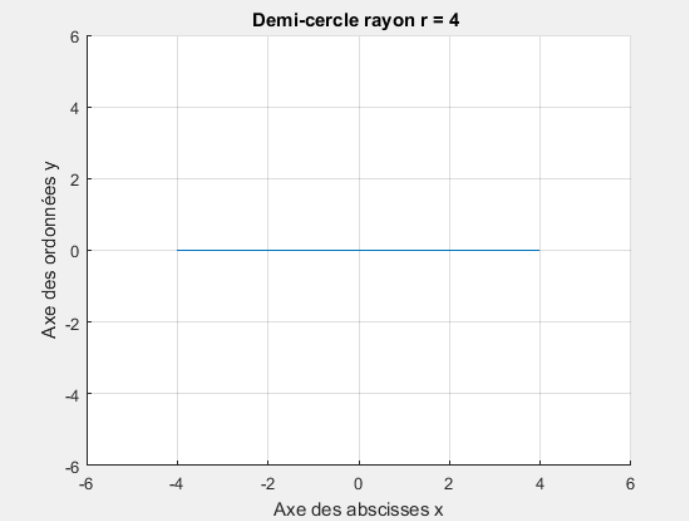
d> 0 déplacement suivant le sens de l'axe y

d <0 déplacement contraire le sens de l'axe y

ANNEXE II
DIFFERENTS TYPES DE FORMES DE L'ARC

Nous avons changé les paramètres pour avoir les différentes formes suivantes

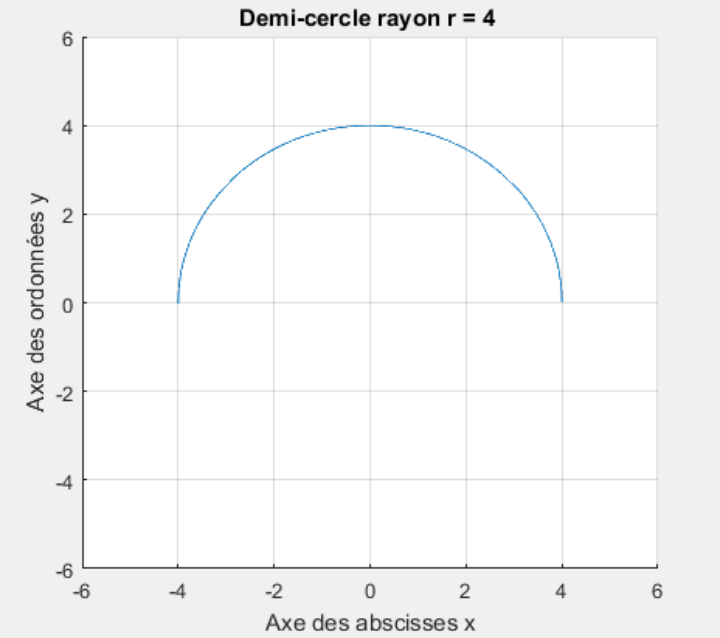
Avec `Va = 0; ThetaA = 0; Vb = 0; ThetaB = 0;`



Graphe 1

Remarque : Nous pouvons tracer une ligne droite en mettant Va=0 et Vb=0

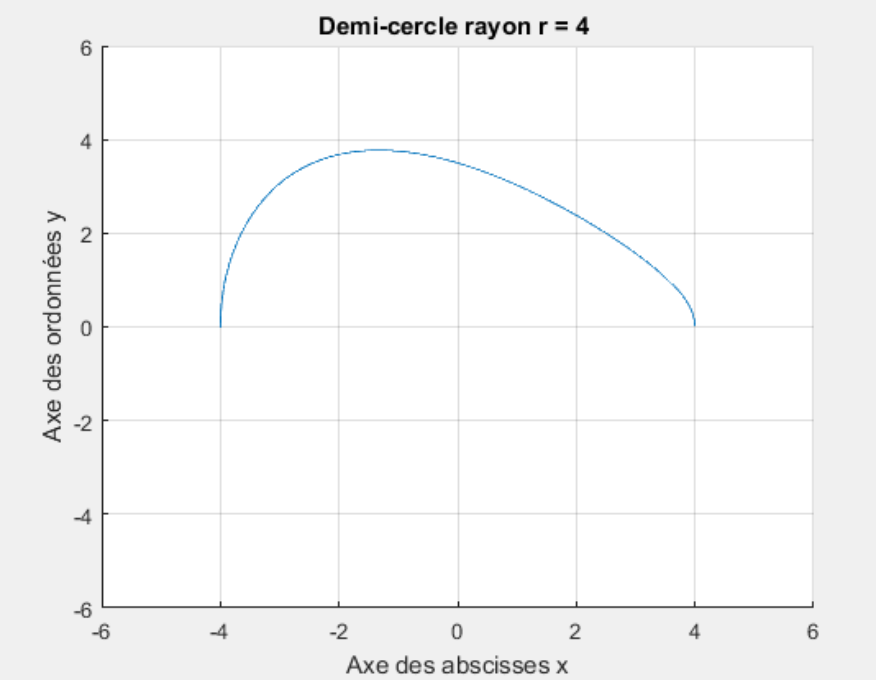
Avec `Va = 4; ThetaA = 0; Vb = 4; ThetaB = 0;`



Graphe 2

Remarque : nous avons un arc de cercle si les angles sont 0 et avec Va =Vb = r

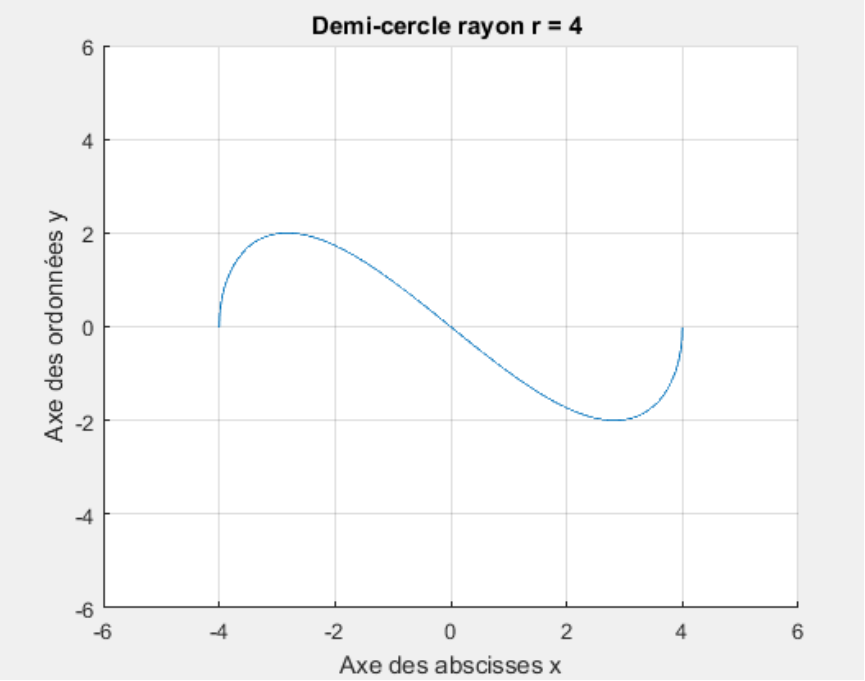
vec `Va = 5; ThetaA = 0; Vb = 2; ThetaB = 0;`



Graphe 3

Remarque : une inclinaison vers Va si Va > Vb

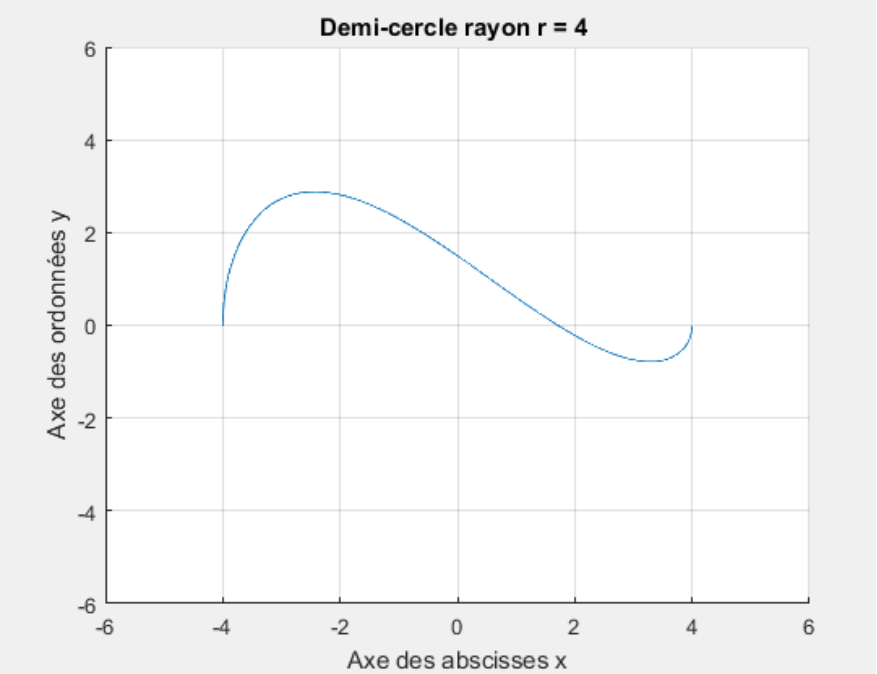
Avec `Va = 4; ThetaA = 0; Vb = -4; ThetaB = 0;`



Graphe 4

Remarque : obtention d'une courbe symétrique par rapport au point 0

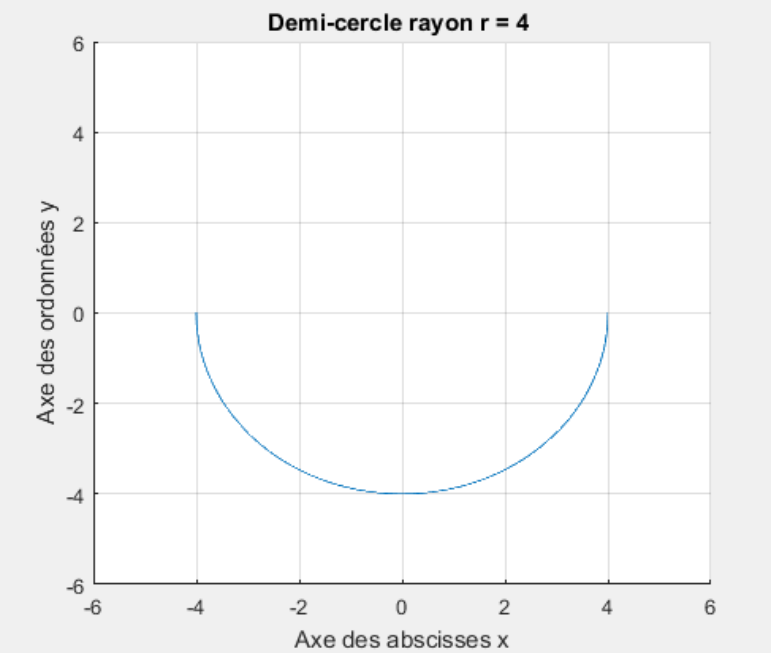
Avec `Va = 5; ThetaA = 0; Vb = -2; ThetaB = 0;`



Graphe 5

Remarque : en faisant Va > Vb avec leurs sens opposés on a une courbure plus grande côté Va .

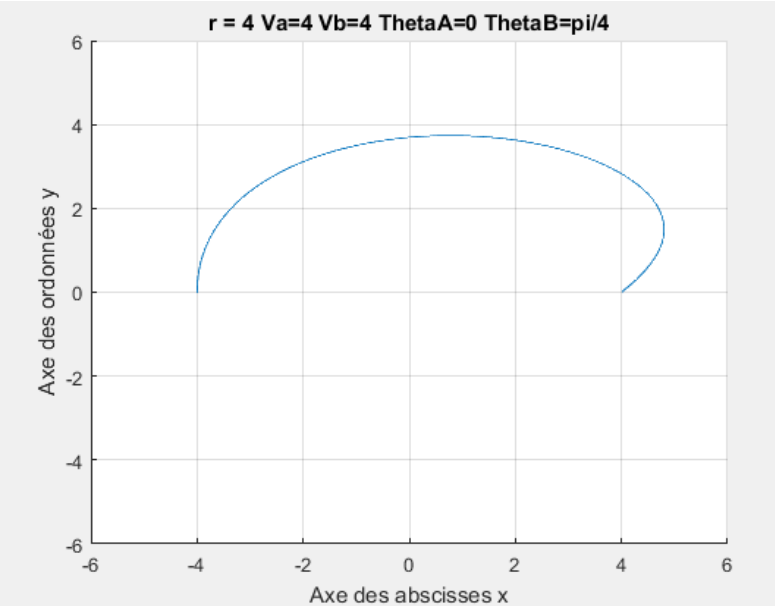
Avec `Va = -4; ThetaA = 0; Vb = -4; ThetaB = 0;`



Graphe 6

Remarque : une forme convexe si les deux vecteurs ont leurs sens contraire à l'axe y.

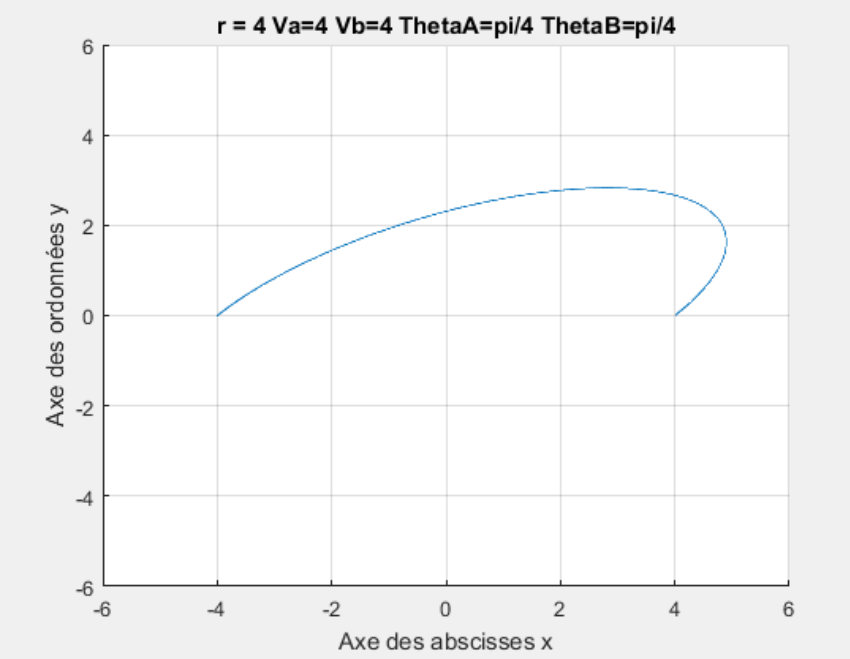
Avec `Va = 4; ThetaA =0; Vb = 4; ThetaB = pi/4;`



Graphe 7

Remarque : on obtient une forme étirée suivants l'axe des abscises si on met l'angle θb entre 0 et π

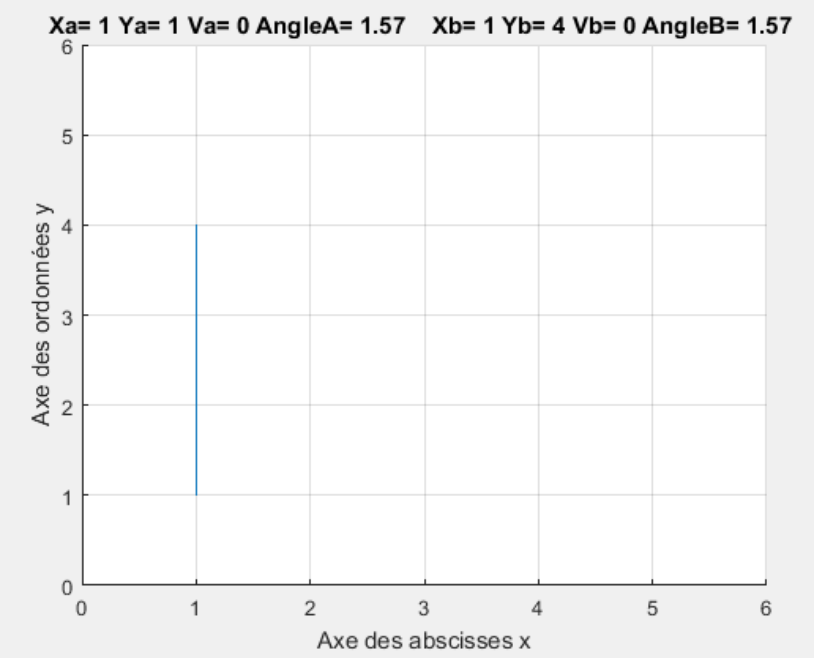
Avec `Va = 4; ThetaA = pi/4; Vb = 4; ThetaB = pi/4;`



Graphe 8

Remarque : l'arc s'incline dans le sens des abscisses

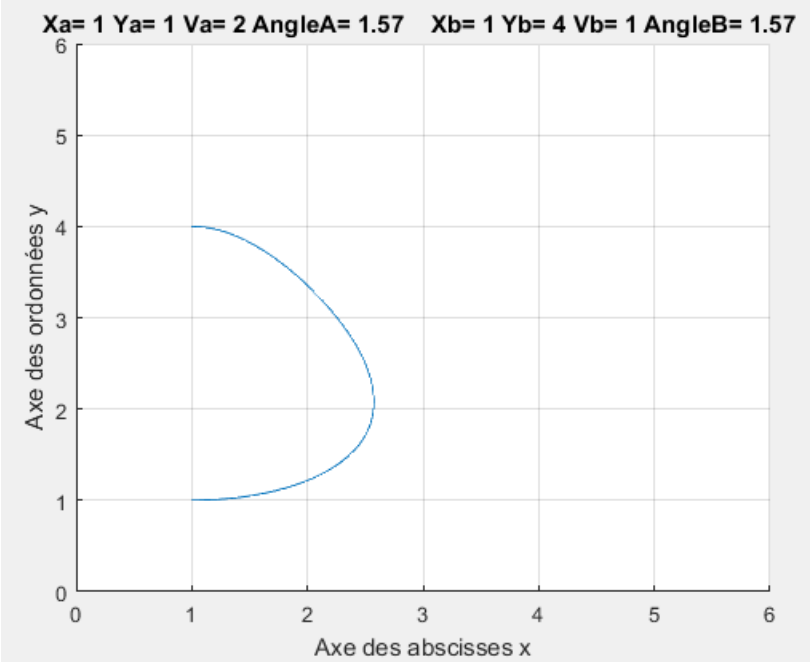
TraceSegmentVecteur(1,1,0,pi/2,1,4,0,pi/2,6,6)



Graphe 9

Remarque : obtention d'un segment vertical

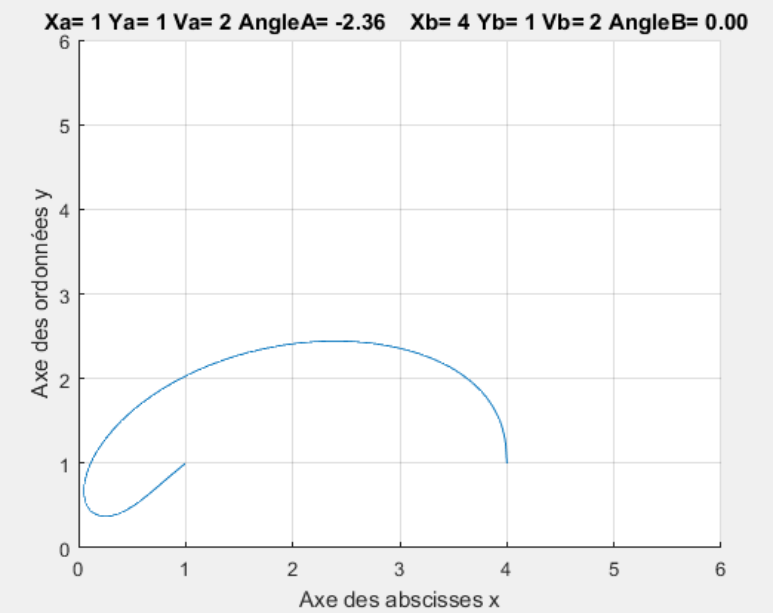
TraceSegmentVecteur(1,1,2,pi/2,1,4,1,pi/2,6,6)



Graphe 10

Remarque : obtention d'une courbe avec extrémité même coordonnée x

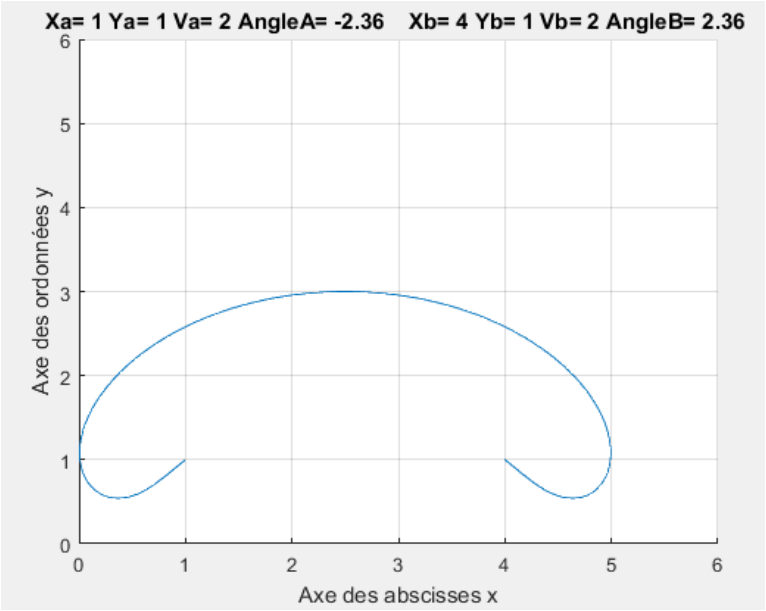
TraceSegmentVecteur(1,1,2,-3*pi/4,4,1,2,0,6,6)



Graphe 11

Remarque : forme résultant de la variation de l'angle des vecteurs

TraceSegmentVecteur(1,1,2,-3*pi/4,4,1,2,3*pi/4,6,6)



Graphe 12

Remarque : la forme dépend de l'angle des vecteurs

ANNEXE III

CODE COMPLET SUR MATLAB

A– Code pour tracer un arc à partir des paramètres des 2 vecteurs caractéristiques.

```
% Mémoire MVRSDE par Herinirina 02/12/2018
% traçage d'une courbe caractérisée par deux vecteurs Va et Vb

% les Vecteurs Va et Vb

Va = 4;
ThetaA = 0.5;
Vb = 2;
ThetaB = 1;
r = 4; % demi distance entre Va et Vb .

p=0.01; % pas du traçage
x= -r : p : r ; % domaine du traçage

% Initialisation

close all;
hold on;


% les Axes
xmin = -6;
xmax = 6 ;
ymin = -6 ;
ymax = 6 ;
xlabel('Axe des abscisses x');
ylabel('Axe des ordonnées y');
title('Demi-cercle rayon r = 4 ');
axis([xmin xmax ymin ymax]);
box off ;
grid on;




 % Tracage de la fonction Plot (x",y") en variant x

plot(x +  ((sqrt ( r*r - ((x).*(x)))).*((((Vb-
Va)/(2*r*r)).*x)+((Vb+Va)/(2*r)))).*sin(((ThetaB-ThetaA)/(2.*r)).*x +
((ThetaA+ThetaB)/2)) , ((sqrt ( r*r - ((x).*(x)))).*((((Vb-
Va)/(2*r*r)).*x)+((Vb+Va)/(2*r)))).* cos(((ThetaB-ThetaA)/(2.*r)).*x +
((ThetaA+ThetaB)/2)));
```

B– Code pour la Transformation d'image matricielle en image vectorielle par combinaison d'arc.

```
close all

Extremum=0;
flag = 1;
MemePoint = 0;
k=0;
deviat=0;

IM= imread('Madagasikara.jpg');
%IM= imread('essais3.jpg')

%Rotation matrice
Imatrice=rot90(IM);
IM=rot90(Imatrice);
Imatrice=rot90(IM);
imshow(Imatrice)

%dimension
[a b c]=size(Imatrice)

%Transformation en niveau de gris
if c==3
    Hmatrice=rgb2gray(Imatrice);
else
    Hmatrice=Imatrice;
end

%figure
imshow(Hmatrice)

%Transformation en binaire
Jmatrice = convert2binary(Hmatrice,150);

% Recherche de 1er Point A ( Ai,Aj)
 Terminer =0;
for i=1:1:a
    for j=1:1:b
        if Jmatrice(i,j)==1
            Terminer =1;
            A0i=i;% Point A(n,m)
            A0j=j;
            X0i=i;% point imaginaire avant
            X0j=j-1;
            break
        end
    end
    if Terminer==1
        break
    end
end
A0i
A0j
X0i
X0j
```

```
 % detection des point et contour
 while flag == 1 && MemePoint == 0
      k=k+1; % compteur point et indice point
      Point(k,1)=A0i;
      Point(k,2)=A0j;
      Point(k,3)=deviat;

      % Recherche de point suivant ( Bi,Bj)
      [B0i,B0j,flag,deviat,AngleXX] =
PointSuivant(Jmatrice,A0i,A0j,X0i,X0j);
      Point(k,4)=AngleXX;

      for CompteurN=1:k
            if (B0i==Point(CompteurN,1)) && (B0j==Point(CompteurN,2))
            MemePoint = 1;
            end
      end
X0i=A0i;
X0j=A0j;
A0i=B0i;
A0j=B0j;
end

% Elimination des parasites ou bruit
 for p=2:k-1

    if Point(k-1,3)==1 && Point(k,3)==-1 && Point(k+1,3)==1
         Point(k,3)=1;
    end
    if Point(k-1,3)==-1 && Point(k,3)==1 && Point(k+1,3)==-1
         Point(k,3)=-1;
    end
 end

% Detection extremum
Direction =-1
NombreExtremum=1;
 Extremum(1,1)=1;
 Extremum(2,1)=-1; % initialisation forme

for m=2:k
   if Point(m,3)<0 && Direction>0
       NombreExtremum=NombreExtremum+1;
       Extremum(1,NombreExtremum)=m-1;
       Extremum(2,NombreExtremum)=-1;
       Direction=-1;
       AngleIni=Point(m,4);
   end
   if Point(m,3)>0 && Direction<0
       NombreExtremum=NombreExtremum+1;
       Extremum(2,NombreExtremum)=1;
       Extremum(1,NombreExtremum)=m-1;
       Direction=1;
   end

end

NombreExtremum =NombreExtremum+1;
Extremum(1,NombreExtremum)=k;
NombreExtremum
```

```
%calcul des vecteurs

for q=1:NombreExtremum-1;

    %arc(XA,YA,VecteurA,AngleA,XB,YB,VecteurB,AngleB)

    arcx(q,1)=Point(Extremum(1,q),1);% XA
    arcx(q,2)=Point(Extremum(1,q),2); %YA

    arcx(q,5)=Point(Extremum(1,q+1),1); %XB
    arcx(q,6)=Point(Extremum(1,q+1),2);%YB

    AngleRotation = atan(arcx(q,5)-(arcx(q,1))/(arcx(q,6)-arcx(q,2)));

    arcx(q,4)=Point(Extremum(1,q),4); % Angle A
    arcx(q,8)=Point(Extremum(1,q+1)-1,4)-pi; % Angle B

    arcx(q,3)=1;
    arcx(q,7)=1;

end

for Indicearcx =1:NombreExtremum-1


TracearcxVecteur(arcx(Indicearcx,1),arcx(Indicearcx,2),arcx(Indicearcx,3),ar
cx(Indicearcx,4),arcx(Indicearcx,5),arcx(Indicearcx,6),arcx(Indicearcx,7),ar
cx(Indicearcx,8),a,b)

end

figure
xmin=0;
ymin=0;
xmax=a;
ymax=b;
xlabel('Axe des abscisses x');
ylabel('Axe des ordonnées y');


x = 1:1:k
plot( Point(x,1),Point(x,2));
axis([0 a 0 b]);
```

REFERENCES

[1] : **Filtrages et segmentations d'images** ;Cours 5ieme année, Université d'Antananarivo , Ecole Supérieur Polytechnique. 2016-2017

[2] : **Géométrie algorithmique** ; Cours MVR-SDE ; Université d'Antananarivo, Ecole Supérieur Polytechnique. 2016-2017

[3] : **Infographie** ; Cours MVR-SDE ; Université d'Antananarivo , Ecole Supérieur Polytechnique. 2016-2017

[4] : **Simulation et modélisation sur Matlab** ; Cours MVR-SDE ; Université d'Antananarivo , Ecole Supérieur Polytechnique. 2016-2017

[5] : **Outils de simulation de système** ; Cours MVR-SDE ; Université d'Antananarivo , Ecole Supérieur Polytechnique. 2016-2017

[6] : **Théorie de l'approximation** ; Cours MVR-SDE ; Université d'Antananarivo , Ecole Supérieur Polytechnique. 2016-2017

[7] : **théorie des graphes** ; Cours MVR-SDE ; Université d'Antananarivo , Ecole Supérieur Polytechnique. 2016-2017

[8] : **Les images vectorielles.**
https://www.imedias.pro/cours-en-ligne/graphisme-design/definition-resolution-taille-image/les-images-vectorielles-matricielles/
Consultation : novembre 2018.

[9] : **Images vectorielles.**
https://fr.wikipedia.org/wiki/Image_vectorielle
Consultation : novembre 2018.

[10] : **Les images numeriques .**
http://imagenumerique.pagesperso-orange.fr/L'imagevectorielle.htm
Consultation : novembre 2018.