Table des matières

Table des matières

Table des matières

Chapitre 2 Reconnaissance de panneaux de signalisation routière

2.1 Introduction :

La reconnaissance d'objets est une étape primordiale pour la mise en œuvre de plusieurs applications actuelles qui nécessitent une interprétation de haut niveau d'images, c'est une tâche difficilement réalisable, compte-tenu notamment des problèmes qu'on va citer par la suite.

2.2 Historique des panneaux de signalisation routière :

Les panneaux routiers sont aussi vieux que les routes. Les premiers signes furent ceux donnant les directions. L'Empire romain a érigé des bornes, en colonnes de pierres, le long des routes pour indiquer les distances jusqu'à Rome. Au Moyen Âge, les signaux directionnels devinrent courants, indiquant la direction des villes aux carrefours. Inscrites sur des colonnes ou des bornes, les indications sont progressivement, à partir du XVIII^{e} et au XIX^{e} siècle, placées sur des plaques (en fonte, à partir de 1835), dites plaques de cocher, fixées sur les murs, les croix ou obélisques des carrefours, ou sur des poteaux métalliques.

La signalisation devient beaucoup plus importante avec le développement de l'automobile. Les principes de base de la plupart des signalisations furent définis le 11 octobre 1908 lors du premier congrès international de la route à Paris [14].

2.3 Reconnaissance des panneaux de signalisation routière :

La reconnaissance automatique des panneaux de signalisation routière est réalisée pour le but d'aider les conducteurs dans la route et augmenter leurs sécurité en cas de fatigue lors de la conduite.

Lors de la réalisation de la reconnaissance nous faisons face à plusieurs difficultés parmi ces difficultés on cite :

- Les conditions d’éclairage
- Le bruit de capteur d’image
- La ressemblance de couleur
- Le mauvais positionnement et la qualité des plaques en Algérie

Il faut surmonter ces obstacles afin de réaliser une bonne reconnaissance d'objet.

Notre travail est basé sur des différentes étapes représentées par la **Figure2.1 :**

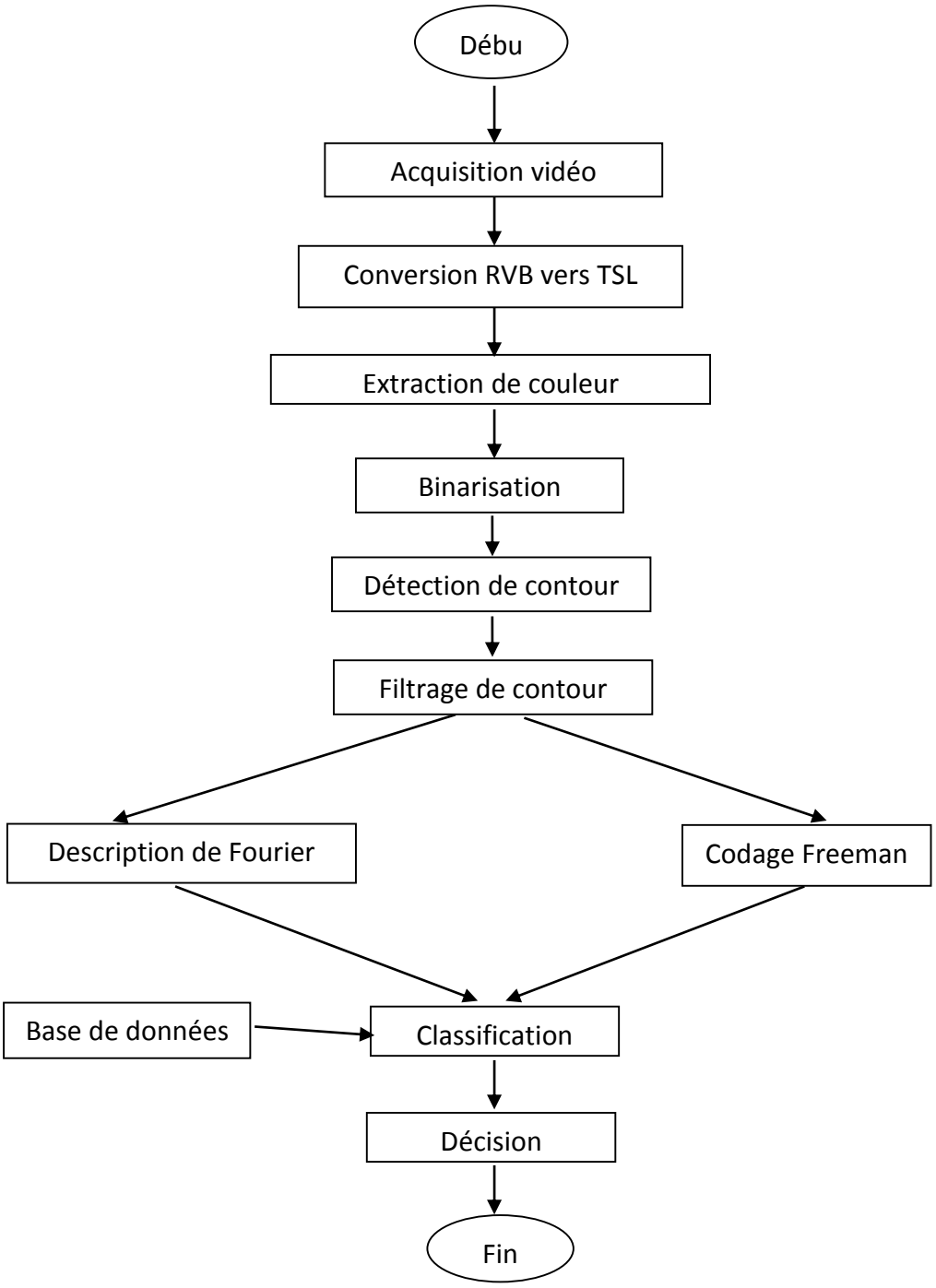


Figure2.1 : Schéma synoptique de la méthode proposée.

2.3.1 Lecture de la vidéo :

Pour les vidéos nous avons utilisé une caméra de résolution VGA (480x640) pixels.

Lors de la lecture nous avons pris chaque frames (images) de notre vidéo pour ensuite les traiter.

2.3.2 Conversion RGB vers TSL :

Nous devons convertir nos images de l'espace RGB (Red, Green, Blue /Rouge, Vert, Bleu) vers l'espace TSL (teinte, saturation, luminance) car ce dernier nous seras utile pour extraire une plage de couleur très facilement.

2.3.3 Extraction de couleur :

Dans notre projet on a besoin de deux couleurs : le rouge et le bleu (couleurs des panneaux), donc on doit extraire la couleur rouge pour les panneaux qui indiquent un danger ou une interdiction et la couleur bleu pour les panneaux qui indiquent une obligation, voir les figures (**Figure2.2, Figure2.3, Figure2.4)**.

Figure2.2 : Image original d'un panneau danger et sont extraction de couleur (rouge).

Figure2.3 : Image original d'un panneau d'interdiction et son extraction de couleur (rouge).

Figure2.4 : Image original d'un panneau d'obligation et son exctration de couleur(bleu).

L’intervalle choisi pour la couleur rouge (Teinte, Saturation, Luminance) :

La teinte : de [0 20] à [286 360]

La saturation : de [0.39   1] à [0.27 1]

La luminance : [0.19 1]

L’intervalle choisi pour la couleur bleu (Teinte, Saturation, Luminance) :

La teinte : [196 260]

La saturation : [0.66   1]

La luminance : [0 0.59]

2.3.4 Binarisation :

Apres l’extraction de la couleur on applique une binarisation sur notre image, touce qui est de couleur rouge (ou bleu) prend la valeur 1 (blanc) et toutes les autres couleurs prennent la valeur 0 (noir) voir les figures (**Figure2.5, Figure2.6)**.

Figure2.5 : image d’un panneau rouge et sa binarisation.

Figure2.6 : image d'un panneau bleu et sa binarisation.

2.3.5 Algorithme du suivi de contour:

Cet algorithme est très rapide car il ne nécessite que peu de calculs à chaque étape. Il présente l'avantage de détecter les pixels de frontières suivant toutes les directions. Tous les pixels seront traités une seule fois [15].

L'algorithme de suivi de contour permet de détecter les contours d'objet d'images, ces contours peuvent être représentés de la façon suivante :

- Les objets sont constitués de pixels « blancs », leur valeur « 1 » sera attribués, tandis que l'extérieur (fond) est formé de pixels « noir» de valeur « 0 ».

Ces contours sont classés selon deux catégories :

- Ceux qui englobent un objet. Ils sont dits extérieurs, c'est la silhouette de l'objet.
- Ceux qui sont noyés dans un objet, ce sont les contours intérieurs.

Apres la détection du premier pixel contour, la nature de la transition détermine sont type :

- Si cette transition est de nature fond forme (1 – 0), c'est un contour de type externe.
- Sinon (0 – 1), le contour est de type interne voir (Figure 2.7)

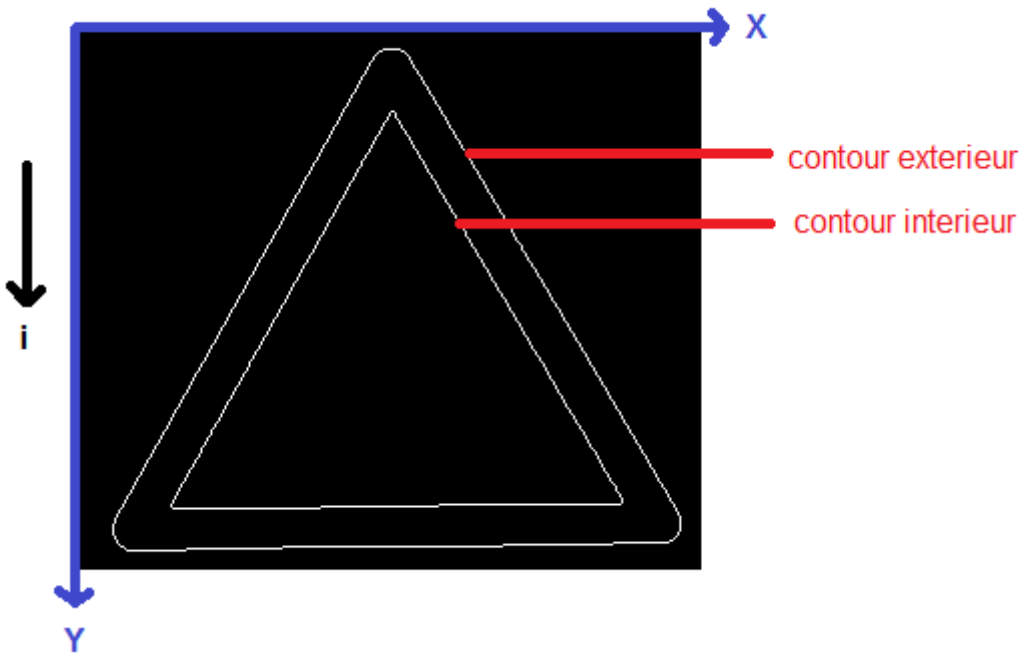


Figure 2.7 : extraction de contours

Cette méthode consiste à balayer l'image jusqu'à trouver une transition. C'est le premier pixel de contour.

Etiqueter les huit voisinages de ce pixel. Tester si ces pixels appartiennent au contour. À chaque fois qu'un pixel objet est rencontré, on lui étiquette ses huit voisinages et continuer jusqu'à la rencontre du premier pixel trouvé.

- **Implémentation de la méthode**

Les principales étapes de l'algorithme sont les suivantes :

1er étape :

Balayage de l'image jusqu'à trouver le premier pixel objet.

Premier pixel contour à (i_1,j_1) Le pixel qui le précède est noté par (i_D,j_D).

2eme étape :

A partir du pixel (i_D,j_D) en tournant autour du pixel (i_1,j_1) de façon à balayer ses voisins dans le sens horaire, on numérote les sept autres voisins du pixel (i_1,j_1) comme 2 ,……,8.

3eme étape :

Evaluer les coordonnées **Lx(k)**, **Ly(k)** du **k^{ieme}** voisin de (i_1,j_1) en utilisant la table de la (Figure 2.8)

	Lx(k)	Ly(k)
1	i_d	j_d
2	Lx(1)+k1	Ly(1)-k2
3	Lx(2)-k2	Ly(2)-k1
4	Lx(3)-k2	Ly(3)-k1
5	Lx(4)-k1	Ly(4)+k2
6	Lx(5)-k1	Ly(5)+k2
7	Lx(6)+k2	Ly(6)+k1

Figure 2.8 : coordonnés des 8 voisins

Coordonnées du pixel contour (i_1, j_1)

Coordonnées du 1er pixel voisin (i_d, j_d)

K1 = $j_d - j_1$

K2 = $i_d - i_1$

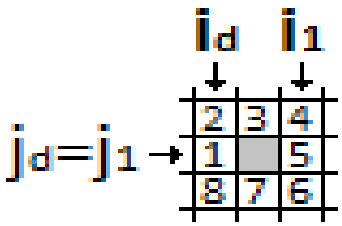


Figure 2.9 : Définition du point contour

4eme étape :

Si le **k**$^{\textbf{ieme}}$ voisin est un point noir, il sera le point suivant du contour et on définit $\mathbf{(i_1,j_1)}$ en ce point. $\mathbf{(i_d,j_d)}$ sera le **(k – 1)** $^{\textbf{ieme}}$ point de la table aller à l'étape 2.

5eme étape :

Si le **k**$^{\textbf{ieme}}$ voisin est un pixel blanc, prendre k=k+1 et aller à l'étape 3.

6eme étape :

Continuer le processus jusqu'à la rencontre du premier point contour détecté.

7eme étape :

Affecter à tous les points contours détecté une valeur différente de 0 et de 1 comme intensité du contour et numéro du contour. Cela permet d'éviter la détection du même contour une notre fois.

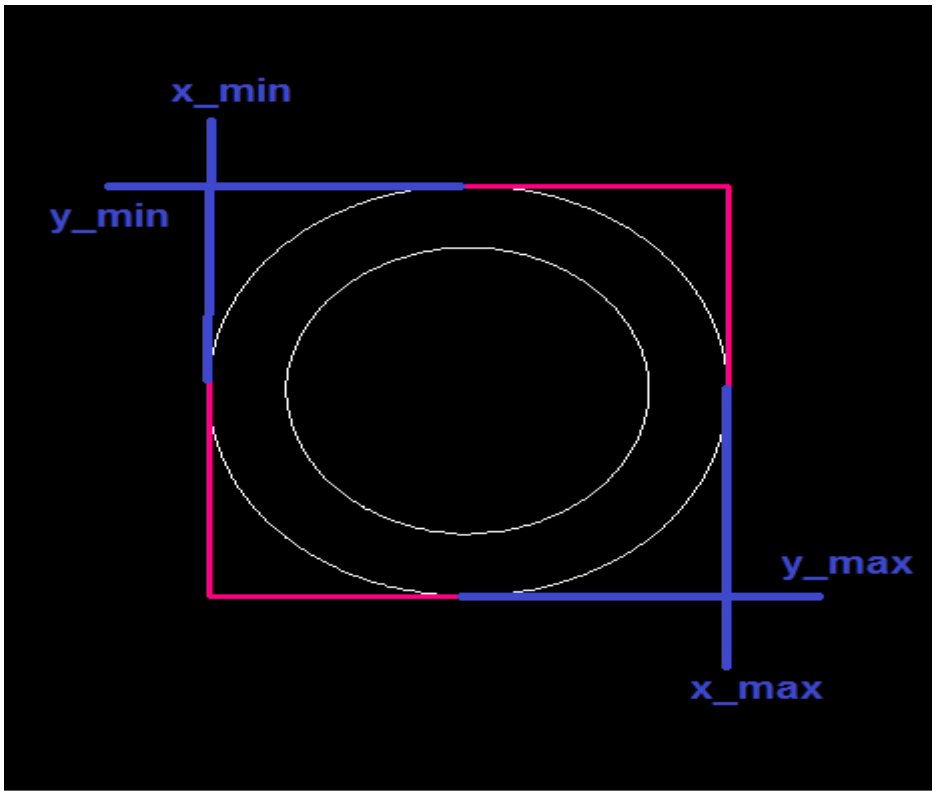
8eme étape :

Poursuivre le balayage de l'image pour détecter de nouveaux contours jusqu'à la fin de l'image.

2.3.6 Filtrage de contour :

Le Filtrage de contour consiste à réduire de manière significative la quantité de données et élimine les informations qu'on peut juger moins pertinentes, tout en préservant les propriétés structurelles importantes de l'image. Les étapes de cette technique sont les suivantes :

- Trouver les contours de l'image.
- Boucler sur les contours et chercher les coordonnés max et min du chaque contour
- Calculer le rapport et enlever les contours. (**Figure** 2.10).



le rapport : $$r = \frac{x\_max - x\_min}{y\_max - y\_min}$$

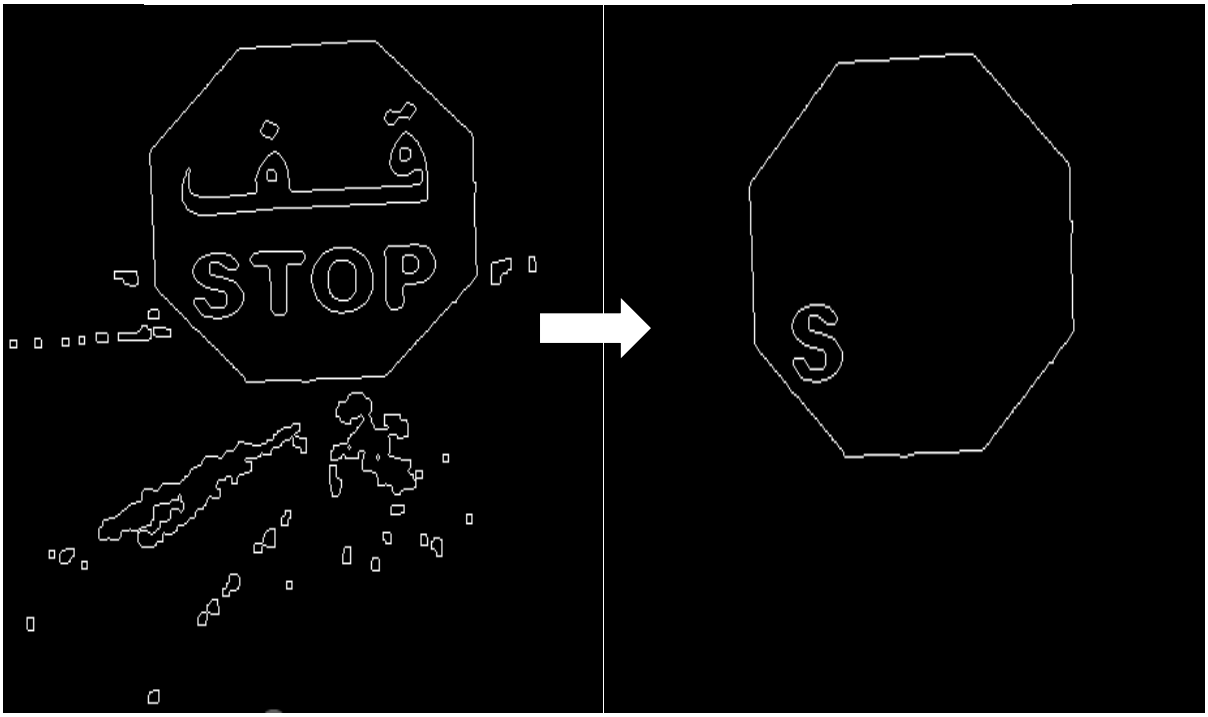


Figure 2.10 : Filtrage de contour.

2.3.7 Description de contour :

a)Transformée de Fourier :

L'idée d'exprimer une fonction de forme compliquée comme une combinaison linéaire de fonction élémentaire de forme simple est très utile.

Par exemple, dans un intervalle [u1, u2], une fonction θ(u) peut être représentée par :

$$\theta(u) = \sum \alpha i.\, \varphi i(u) \quad \text{..... (2.1)}$$

Où les fonctions $\varphi i(u)$ forment un ensemble de fonctions élémentaires de forme simple. Si ces fonctions sont orthogonales, alors les coefficients αi sont indépendants les uns des autres on parle dans ce cas d'un développement en série de fonctions orthogonales.

Parmi les développements en séries orthogonales, la série de Fourier est sans doute celle dont on fait le plus grand usage en traitement des signaux.

On peut généraliser la notion de série de Fourier pour représenter une fonction dans un intervalle infini. Cette généralisation aboutit à la transformation intégrale de Fourier

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{(-j2\pi ft)}\, dt \quad \text{..... (2.2)}$$

La transformée inverse est donnée par :

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{(j2\pi ft)}\, df \quad \text{..... (2.3)}$$

La transformée de Fourier des signaux numériques est définie par :

$$X(f) = \sum_{k=-\infty}^{k=+\infty} x(k) e^{(-j2\pi fk)} \quad \text{..... (2.4)}$$

Soit f(x) une fonction continue de la variable réelle x. La transformée de Fourier notée

F {f(x)} est définit par l'équation :

$$F\{f(x)\} = F(u) = \int_{-\infty}^{+\infty} f(x) e^{(-j2\pi ux)} \quad \text{..... (2.5)}$$

De même, à partir de F(u) on peut revenir à f(x)

$$F^{-1}\{f(x)\} = f(x) = \int_{-\infty}^{+\infty} F(u) e^{(j2\pi ux)} \quad \text{..... (2.6)}$$

Généralement la transformée de Fourier d'une fonction réelle est complexe et s'écrit :

$$F(u) = R(u) + jI(u) \quad \text{..... (2.7)}$$

Où R(u) et I(u) sont respectivement les parties réelles et imaginaires de F(u)

$$F(u) = |F(u)|e^{-j\emptyset(u)} \text{ .....(2.8)}$$

Ou

$$|F(u)| = [R^2 + I^2]^{\frac{1}{2}} \text{ ..... (2.9)}$$

Et

$$\emptyset(u) = arctg\left[\frac{I(u)}{R(u)}\right] \text{..... (2.10)}$$

La variable apparaissant dans la transformée de Fourier est souvent dite variable fréquentielle (ou fréquence). Ceci provient de l'utilisation de la formule d'Euler des nombres complexes $e^{2\pi ux}$ peut être exprimé sous la forme :

$$e^{-j2\pi ux} = \cos(2\pi ux) - j\sin(2\pi ux) \text{.....} (2.11)$$

Si nous interprétons l'intégrale de l'équation (2.11) comme une sommation limitée de termes discrets, il est évident que F(u) est composé d'une somme infinie de termes en sinus et cosinus et chaque valeur de u détermine la fréquence correspondant à une paire sinus-cosinus.[15]

b) Transformée de Fourier discrète :

Supposons qu'une fonction continue est discrétisée en séquences

$$\{f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), \ldots, f(x_0 + (N-1)\Delta x)\} \text{.....} (2.12)$$

En prenant N unités$\Delta \boldsymbol{x}$.

On peut prendre $\boldsymbol{x}$ comme variable en écrivant :

$$f(x) = f(x_0 + x\Delta x) \text{.....} (2.13)$$

Ou $\mathbf{x}$ prend des valeurs discrètes 0, 1,2,…, N-1

Ceci nous donne la relation de la DFT suivante :

$$F(u) = \sum_{x=0}^{N-1} f(x) e^{(\frac{-j2\pi ux}{N})} \qquad u = 0,1,2,\dots,N-1 \text{ ……} (2.14)$$

Et

$$F(x) = \frac{1}{N} \sum_{x=0}^{N-1} f(u) e^{(\frac{-j2\pi ux}{N})} \qquad x = 0,1,2,\dots,N-1 \dots\dots (2.15)$$

Les valeurs $u = 0,1,2,\dots,N-1$ pour la transformée discrète correspondant à :

$0, \Delta u, 2\Delta u, (N-1)\Delta u$, pour la transformée continue.

En d'autres termes $F(u)$ représente $F(u\Delta u)$.

Pour une fonction à deux variables la DFT est donnée par :

$$\mathrm{u} = 0,1,2,\dots,\mathrm{M}-1 \quad , \quad \mathrm{v} = 0,1,2,\dots,\mathrm{N}-1$$

Et

$$(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} f(u,v) e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \text{ ……} (2.16)$$

Dans le cas d'une matrice carrée M = N

$$F(u,v) = \frac{1}{N}.M \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})} \text{ ……} (2.17)$$

$$u = v = 0,1,2,\dots,N-1$$

c) Descripteur de Fourier :

L'une des techniques les plus promettant de description de forme est basée sur les descripteurs de Fourier.

Supposons que nous avons N points de la frontière d'une région. Nous pouvons considérer la région comme placée dans le plan complexe, avec l'axe des ordonnées (y) comme axe imaginaire et l'axe des abscisses (x) comme l'axe réel. Alors les coordonnées x, y de chaque point du contour à analyser peuvent être représentées comme des nombres complexes (x+jy).

Le contour peut être écrit sous la forme de séquence de nombres complexes.

$$Z_i = x_i + jy_i \qquad i = 0,1,2,\dots,N-1$$

La séquence est périodique pour chaque contour. Les descripteurs de Fourier sont définis comme suit :

$$A(u) = 1/N \sum_{i=0}^{N-1} Z_i e^{(\frac{j2\pi ui}{N})} \qquad u = 0,1,2,\dots,N-1 \dots\dots\dots\dots (2.18)$$

La transformation étant réversible

$$Z_i = \sum_{u=0}^{N-1} A(u) e^{(\frac{j2\pi ui}{N})} \qquad i = 0,1,2,\dots,N-1 \dots\dots\dots\dots (2.19)$$

Avant d'utiliser les descripteurs de Fourier il faut éliminer leur dépendance de la position, la dimension, l'orientation et le point de départ, comme suit :

1 - Le changement de la position du contour altère A(0) seulement.

2 - En multipliant la dimension du contour par une constante, les descripteurs de Fourier sont multipliés par une constante également].

3 - La rotation du contour dans le domaine spatial requiert la multiplication de chaque coordonnées par $\exp(j\theta)$ où θ est l'angle de rotation. La linéarité de la transformée de Fourier fait que la multiplication des coefficients du domaine fréquentiel produit le même effet.

4 - Le changement du point de départ du contour dans le domaine spatial (temporel) correspond à multiplier le k^{ieme} coefficient du domaine fréquentiel par $\exp(jkT)$ où T est une fraction de période par laquelle le point de départ a été décalé (T varie de 0 à 2π , le point de départ traverse le contour une fois).

On normalise les DF en prenant A(0) égal à 0 (pour rendre les DF indépendantes de la position) et on divise chaque coefficient par l'amplitude de la premier coefficient A(1) (pour normaliser leur taille) [16].

- **Algorithme utilisant les descripteurs de Fourier**

Soit (x(m), y(m)) les coordonnées d'un pixel appartenant au contour ; avec *0 ≤ m ≤ L-1*

En prenant la transformée de Fourier discrète (DFT) de la donnée $\big(x(m), y(m)\big)\ 0 \leq m \leq L-1$, on obtient les coefficients de Fourier pour $0 \leq k \leq L-1$

$$A(k) = \sum_{m=0}^{L-1} Z(m)\exp(-j2\pi km/L) \ldots\ldots\ldots\ldots (2.20)$$

$$e^{(j\theta_0)} = cos\,\theta_0 + j\,sin\,\theta_0 \ldots\ldots\ldots\ldots (2.21)$$

$$Z(m) = x(m) + jy(m) \ldots\ldots\ldots\ldots (2.22)$$

D'après les équations (2.21) et (2.22) :

$$A(k) = \frac{1}{L}\sum_{m=0}^{L-1}[x(m) + jy(m)].\left[\cos\left(\frac{2\pi km}{L}\right) - jsin(\frac{2\pi km}{L})\right] \ldots\ldots\ldots\ldots (2.23)$$

$$a(k) = Re\big(A(k)\big) = \frac{1}{L}\sum_{m=0}^{L-1}\left[x(m).\cos\left(\frac{2\pi km}{L}\right) + y(m).\sin(\frac{2\pi km}{L})\right] \ldots\ldots\ldots (2.24)$$

$$b(k) = Im\big(A(k)\big) = \frac{1}{L}\sum_{m=0}^{L-1}\left[-x(m).\sin\left(\frac{2\pi km}{L}\right) + y(m).\cos(\frac{2\pi km}{L})\right] \ldots\ldots\ldots (2.25)$$

On distingue les composants DC $a(0)$ et *b(0)* puisqu'il donne l'information seulement sur la position du centre de l'image, On laisse

$$|A(k)| = \sqrt{|a(k)|^2 + |b(k)|^2} \ldots\ldots\ldots\ldots\ldots (2.26)$$

$$S(k) = A(k)/A(1) \qquad k = 1, \ldots, L-1 \ldots\ldots\ldots\ldots (2.27)$$

Ou $|\mathrm{a(k)}|, |\mathrm{b(k)}|$ denote la valeur absolue des nombres complexes $a(k)$et $b(k)$. Alors, il devient facile de voir que $\mathrm{A(k)}$ est invariant pour la rotation, la translation et plus loin $S(k)$ est invariant pour le changement d'échelle.

Les descripteurs invariants s(k) sont appelés descripteurs de Fourier [16].

2.3.8 chaine de Freeman :

C'est la méthode la plus ancienne de description des contours dans les images et aussi la plus utilisée encore aujourd'hui [Freeman, 1961, Freeman, 1977].

- **Définition :**

C’est une technique de représentation des directions du contour (on code la direction le long du contour dans un repère absolu lors du parcours du contour à partir d’une origine donnée). Les directions peuvent se représenter en 4-connexité (codage sur 2 bits) ou en 8-connexité (codage sur 3 bits)[2].

Le codage d’un contour se fait donc de la façon suivante :

- transmission des coordonnées absolues du point de départ,
- transmission de la liste des codes de d´emplacement d’un point du contour au suivant sur le maillage.

Les codes des contours sont donnés par la **Figure2.11**.

Dans d’autres techniques, on code de façon différentielle le changement de direction d’un point au suivant.

Cela peut se justifier en codage sur 3 bits si une direction est très dominante par rapport aux autres [17].

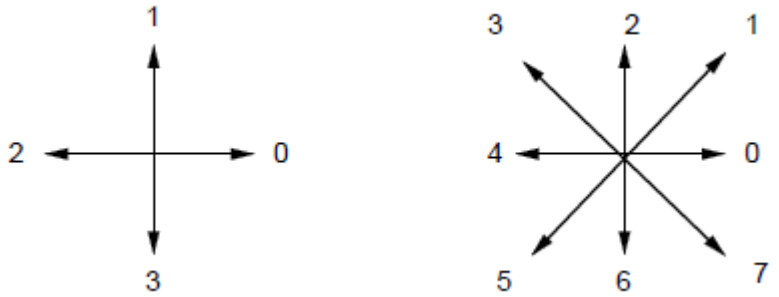


Figure2.11 : Les codes de Freeman en 4-connexité (à gauche) et en 8-connexité (à droite).

Une chaîne 8-directionnelle d’une image codé est illustrée par la (**Figure 2.11**).

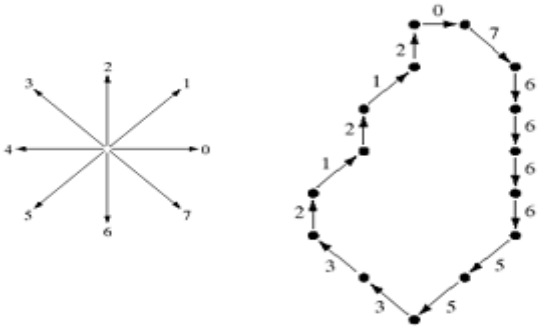


Figure 2.12 : chaîne code de 8-connectivité.

On constate que la chaîne code pour différents chiffres donne différentes longueurs du code et chaque longueur de chaîne code dépend de la taille de la référence de l'objet.

Dans certain cas la taille de la référence de l'objet est très élève, et pour résoudre ce problème il faut normaliser la chaîne code comme suit :

Supposer que cette chaîne code est généré pour un contour traversant en sens anti horaire (sens horaire).

V1= [1 1 2 3 0 1 0 0 0 1 3 2 5 5 5 5 0 7 2 2 2 2 2 3 3 5 7 6 4 4 4 4 4 1 2 4 4 4 4 6 0 0 3 1 6 6 0 0 0 2 2 1 2 4 4 4 6 6 6 7 1 0 0 0 0 0 0 7 7 7 7 1 1]

Calculer la fréquence des codes 0, 1, 2 ..... 7, du vecteur V1 pour obtenir le vecteur fréquence V2 comme ci-dessous.

V2= [14 12 11 5 12 5 7 7]

La fréquence normalisée représenté par le vecteur V3, calculé par la formule suivant :

$$V3 = \frac{V2}{|V1|} \qquad |V1| = \sum V2 \ldots\ldots\ldots\ldots \text{(2.28)}$$

Pour l'exemple considéré ci-dessus, on a :

V3= [0.1917 0.1643 0.1506 0. 0684 0.1643 0.0684 0.0958 0.0958]

Finalement, la concaténation de la vecteur V2 et V3 on obtient un vecteur caractéristique de taille 16 comme suit :

V4= [14 12 11 5 12 5 7 7 0.1917 0.1643 0.1506 0. 0684 0.1643 0.0684 0.0958 0.0958]

2.3.9 Classification :

La classification consiste à classer des individus en fonction de certaines de leurs caractéristiques. Il existe différents types de classification, mais un des plus intuitifs et des plus utilisés est la classification supervisée. L'idée de la classification supervisée est d'apprendre une règle de classement à partir d'un ensemble de données dont le classement est déjà connu. Une fois la règle apprise, il est possible de l'appliquer pour catégoriser de nouvelles données, dont le classement est inconnu. Les machines à vecteurs de support, ou SVM (Support Vector Machines), sont une technique relativement récente de classification supervisée qui suscite beaucoup d'intérêt pour ses bonnes performances dans un large éventail d'applications pratiques.

a) Méthode des machines à vecteurs support (SVM) :

Le SVM (Support Vector Machines) est une méthode de classification binaire par apprentissage supervisé, cette méthode repose sur :

- l'existence d'un classifieur linéaire dans un espace approprié.
- l'utilisation de fonction noyau qui permet une séparation optimale des données.

Cette méthode a été développée dans les années 1990 à partir des considérations théoriques de Vladimir Vapnik sur le développement d'une théorie statistique de l'apprentissage : la Théorie de Vapnik-Chervonenkis [18].

b) Principe de fonctionnement général du SVM :

-Hyperplan, marge et support vecteur :

Le but du SVM est de Trouver un classifieur linéaire (hyperplan) qui va séparer les données et maximiser la distance entre ces 2 classes. Voir la **Figure 2.13**

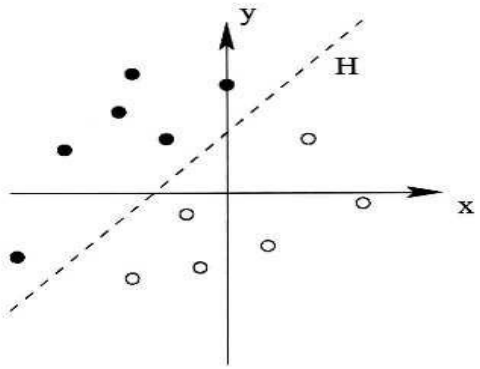


Figure 2.13 : hyperplan.

Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support. Voir la **Figure 2.14**

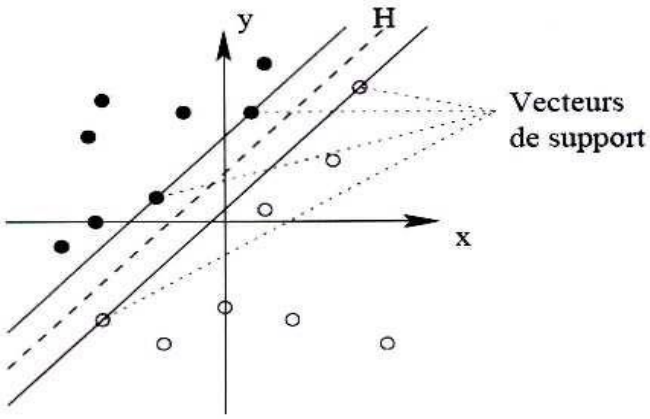


Figure 2.14: support vecteurs.

Il est évident qu'il existe une multitude d'hyperplan valide mais la propriété remarquable des SVM est que cet hyperplan doit être optimal. Nous allons donc en plus chercher parmi les hyperplans valides, celui qui passe "**au milieu**" des points des deux classes d'exemples. Intuitivement, cela revient à chercher l'hyperplan le "plus sur". Voir **Figure 2.15**

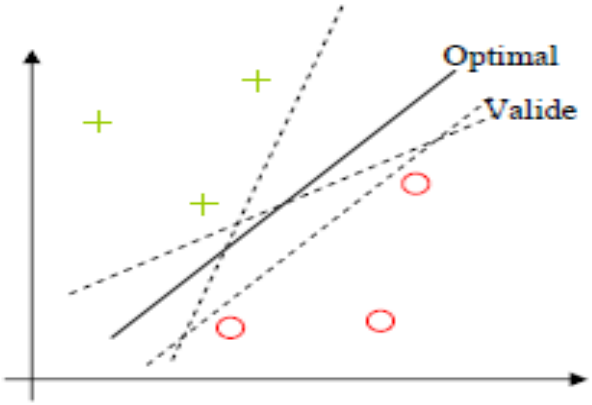


Figure 2.15: Hyperplan optimal.

En supposant qu'un exemple n'ait pas été parfaitement décrit, une petite variation ne modifiera pas sa classification si sa distance à l'hyperplan est grande.

Parmi les hyperplans valides, chercher l'hyperplan dont la distance minimale aux exemples d'apprentissage est maximale. Cette distance est appelée distance 'marge' entre l'hyperplan et les exemples. Voir la **Figure 2.16**

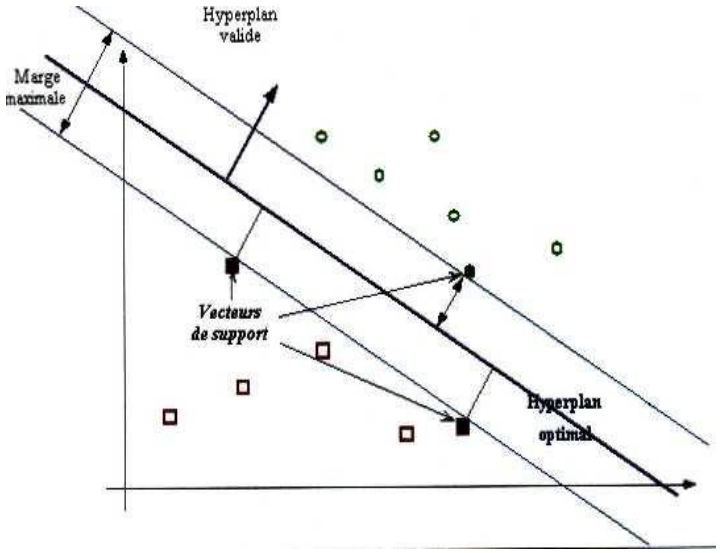


Figure 2.16 : hyperplan, marge et support vecteurs.

-linéarité et non linéarité :

Parmi les modèles des SVM, on distingue les cas linéairement séparables et les cas non-linéairement séparables. Dans les premiers cas, il est facile de trouver le classifieur linéaire.

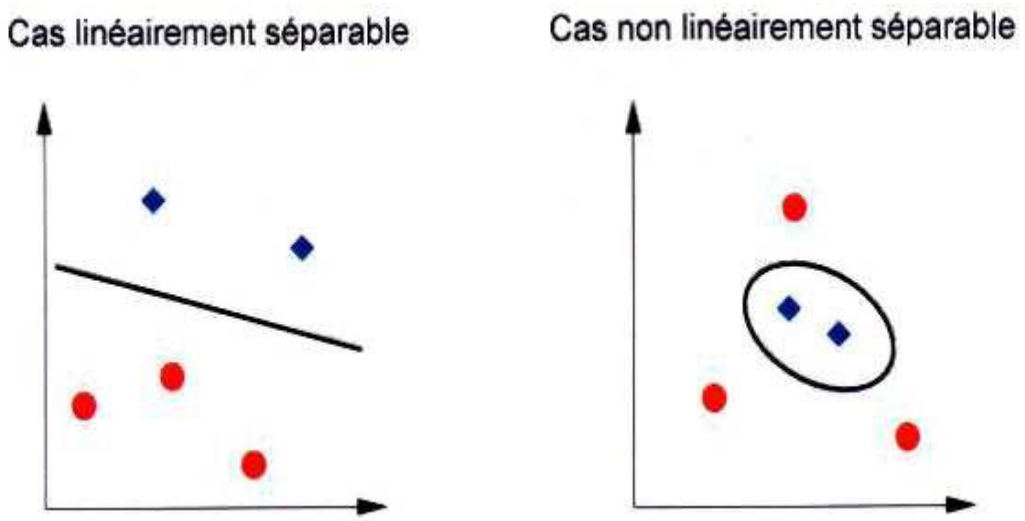


Figure 2.17 : Cas linéaire et non linéaire.

Dans la plupart des problèmes réels il n'y a pas de séparation linéaire possible.

-Cas non-linéaire :

Il faut changer l'espace des données afin de surmonter l'inconvénient des cas non-linéairement séparables. La transformation non-linéaire des données peut permettre une séparation linéaire des exemples dans un nouvel espace, 'espace de représentation'. Intuitivement, plus la dimension de l'espace de représentation est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée. Voir **Figure 2.18**

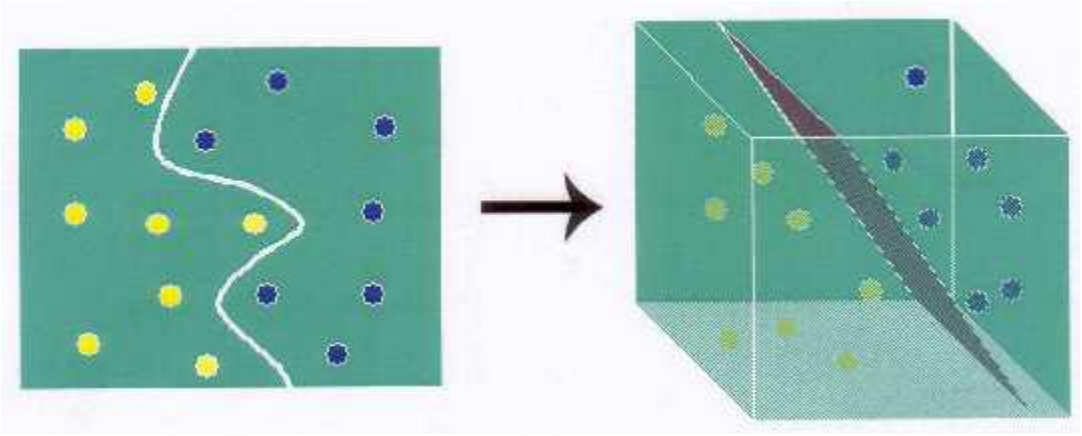


Figure 2.18 : Changement de l'espace.

En a une transformation d'un problème de séparation non-linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace de représentation de plus grande dimension.

Cette transformation non-linéaire est réalisée via une fonction noyau. En pratique, quelques familles de fonctions noyau paramétrables sont connues : polynômiale, gaussien, sigmoïde et Laplacien. Il revient à l'utilisateur de SVM d'effectuer des tests pour déterminer celle qui convient le mieux pour son application.

c) Formules mathématiques :

-Cas linéairement séparable :

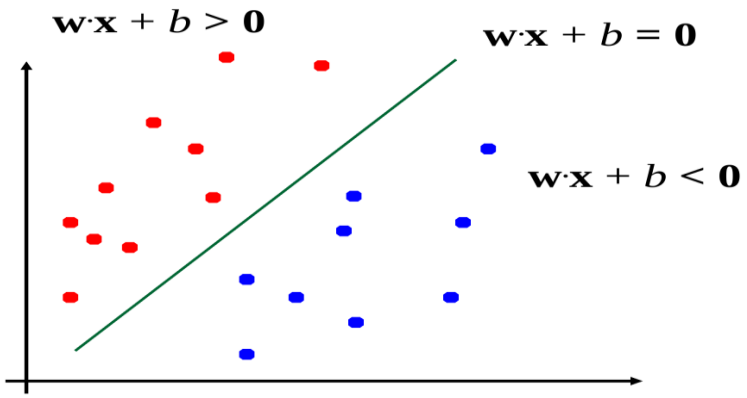


Figure 2.19 : formules mathématiques

Classifieur linéaire **:**

$$y(x) = signe\ (w \cdot x + b) \ldots\ldots\ldots\ldots \quad (2.28)$$

Avec : w: Le vecteur norme par rapport au hyperplan.

x: L'observation ou le point appartenant à la classe.

b: Le biais.

Tout ce qui dépasse la limite de décision doit avoir l'étiquette 1. De même, tout ce qui est en dessous de la limite de décision doit avoir l'étiquette -1. Voir la **Figure 2. 20**

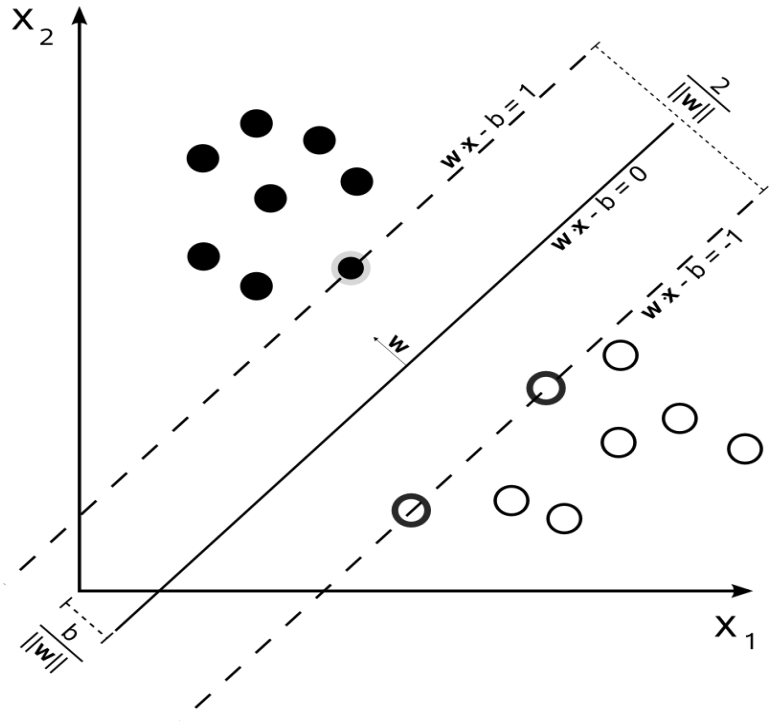


Figure 2.20 : Séparation des classes

Nous pouvons déterminer si une instance a été correctement classés en vérifiant que:

$$y(w^T x + b) \geq 1 \ldots\ldots\ldots\ldots \quad (2.19)$$

Dans ce cas (linéairement séparable), on va considérer les points les plus près de l'hyperplan séparateur: vecteurs supports (support vectors). Voir **Figure (2.21)**

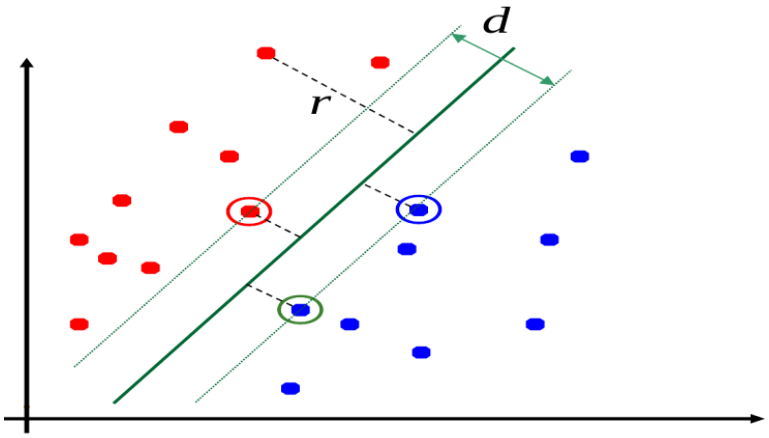


Figure 2.21 : support vecteurs.

Pour tout point de l'espace des exemples, la distance à l'hyperplan séparateur est donnée par :

$$x = \frac{|\mathrm{w} \cdot \mathrm{x} + \mathrm{b}|}{||\mathrm{w}||} \text{..... (2.20)}$$

Notons d'abord que les deux lignes sont parallèles, et ainsi partagent leurs paramètres w, b. En choisissant un point arbitraire x_1 qui appartient à la ligne $w \cdot x + b = -1$ alors le point le plus proche de x_1 sur la ligne $w \cdot x + b = 1$ est le point $x_2 = x_1 + \lambda w$ (puisque le point le plus proche sera toujours perpendiculaire, et le vecteur w est perpendiculaire au deux lignes). En utilisant cette formulation, λw sera le segment de ligne reliant x_1 et x_2, et ainsi, $\lambda \|w\|$ la distance entre x_1 et x_2, est la plus courte distance entre les deux lignes.

Pour trouvé λ:

$$w^T x_2 + b = 1 \text{ Avec } x_2 = x_1 + \lambda w$$

$$w^T (x_1 + \lambda w) + b = 1$$

$$w^T x_1 + b + \lambda w^T w = 1 \text{ Avec } w^T x_1 + b = -1$$

$$-1 + \lambda w^T w = 1$$

$$\lambda w^T w = 2$$

$$\lambda = \frac{2}{w^T w} = \frac{2}{\|w\|^2}$$

Alors, la distance $\lambda \|w\|$ est: $\frac{2}{\|w\|^2} \|w\| = \frac{2}{\|w\|} = \frac{2}{\sqrt{w^T w}}$. Voir **Figure (2.20)**

Maximaliser la distance entre les frontières, revient à maximiser $\frac{2}{\sqrt{w^T w}}$ se qui est équivalent a minimiser $\frac{\sqrt{w^T w}}{2}$ qui est à son tour équivalent à minimiser $\frac{w^T w}{2}$. Ce problème de programmation quadratique est exprimée en tant que:

$$\begin{cases} min_{w,b} \frac{w^T w}{2} \\ sous\ contraintes \\ y_i(w^T x_i + b) \geq 1\ (\forall\ les\ éléments\ des\ donnés\ x_i) \end{cases} \quad (2.21)$$

-SVM a marge souple :

Dans le cas où les données ne sont pas linéairement séparables, il n'est pas garanti que tous les points des données sont correctement étiquetés, et on veut permettre à certains points de données d'une classe à apparaître sur l'autre côté du frontières, L'idée est d'ajouter des variables d'ajustement ξ_i pour chaque x_i dans la formulation pour prendre en compte les erreurs de classification ou le bruit. Voir **Figure (2.22)**

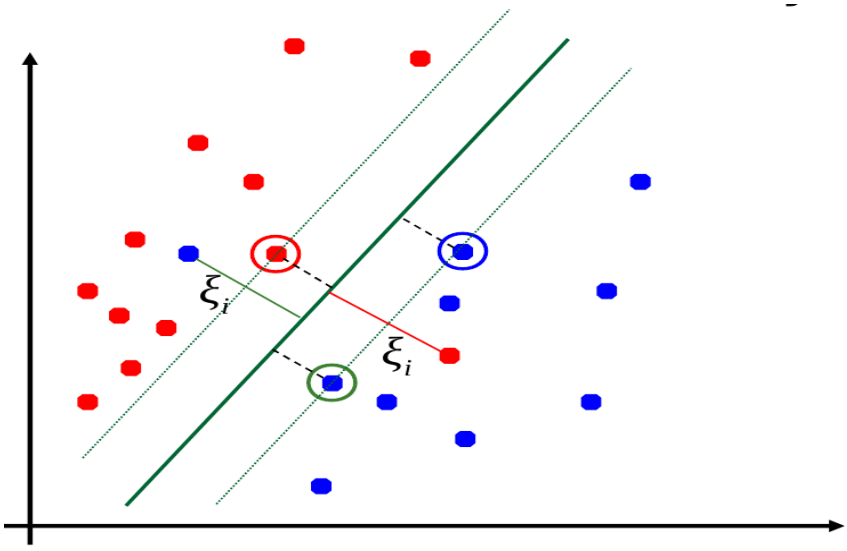


Figure 2.22 : la variable d'ajustement.

Notre problème de programmation quadratique devient:

$$\begin{cases} min_{w,b,\xi} \ \frac{w^T w}{2} + C \sum_i \xi_i \\ sous\ contraintes \\ y_i(w^T x_i + b) \geq 1 - \xi_i \ et \ \xi_i > 0 \ (\forall x_i) \end{cases} \quad (2.22)$$

-utilisation des noyaux :

L'espace des données peut toujours être plongé dans un espace de plus grande dimension dans lequel les données peuvent être séparées linéairement.
La résolution des SVM ne s'appuient que sur le produit scalaire < xi, xj > entre les vecteurs d'entrée Si les données d'apprentissage sont plongées dans un espace de plus grande dimension via la transformation $\emptyset : x \rightarrow \emptyset(x)$ le produit scalaire devient :

$$K(x_i, x_j) = < \emptyset(x_i), \emptyset(x_j) > \ldots\ldots\ldots\ldots (2.23)$$

Avec : *K (xi, xj)* est appelée fonction noyau.
Il existe plusieurs noyaux et les plus utilisé sont :
Noyau linéaire :

$$K(xi, xj) = xi \cdot xj$$

Noyau polynomial de dégrée P :

$$K(xi, xj) = (1 + xi \cdot xj)^P$$

Noyau Gaussien :

$$K(x_i, x_j) = e^{(-\frac{\|x_i - x_j\|^2}{2\sigma^2})}$$

Où σ est un réel positif qui représente la largeur de bande du noyau.

-SVM multi-classes :

Les séparateurs à vaste marge ont été développés pour traiter des problèmes binaires mais ils peuvent être adaptes pour traiter les problèmes multi-classes.
L'adaptation des SVM bi-classes au cas multi-classes peut se faire de façons différentes. Le choix va dépendre de la taille du problème :

1- La méthode un contre reste (1 vs reste) :

Elle consiste à déterminer pour chaque classe k, un hyperplan la séparant de toutes les autres classes. Cette classe k est considéré comme étant la classe positive (+1) et les autres classes comme étant la classe négative, ce qui résulte, pour un problème à K classes, en K SVM binaires.

2- La méthode un contre un (1 vs 1) :

Elle consiste à utiliser un classifieur pour chaque paire de classes. Au lieu d'apprendre K fonctions de décisions, la méthode 1vs1 discrimine chaque classe de chaque autre classe, ainsi K (K - 1)/2 fonctions de décisions sont apprises.

-Application des SVM :

Les avantages théoriques et pratiques des SVM sont très efficaces dans de nombreux problèmes pratiques de classification.

Dans bien des cas, il s'agit de construire un noyau (donc une mesure de similarité) adapté aux données à traiter.

Le SVM est utiliser dans :

- Classification de données biologiques/physiques
- Classification de documents numériques
- Classification d'expressions faciales
- Classification de textures
- Détection d'intrusion
- Reconnaissance de la parole
- CBIR : Content Based Image Retrieval

2.4 Conclusion :

Dans ce chapitre nous avons présenté les différentes étapes utilisé dans la reconnaissance des panneaux de signalisation routière, qui consiste premièrement à détecter un panneau, puis utiliser le descripteur de fourrier et le codage de Freeman jusqu'à arriver à la classification en utilisant le SVM.

On a pris en considération pour choisir cette méthode plusieurs facteurs importants :

- ✓ Possibilité d'implémentation sous Qt Creator C++
- ✓ Simple à utiliser.

Chapitre 1 Généralité sur le traitement d'image

1.1 Introduction

Avec la parole, l'image constitue l'un des moyens les plus importants qu'utilise l'homme pour communiquer avec autrui. C'est un moyen de communication universelle dont la richesse du contenu permet aux êtres humains de tout âge et de toute culture de se comprendre.

C'est aussi le moyen le plus efficace pour communiquer, chaque personne peut analyser l'image à sa manière, pour en dégager une impression et d'en extraire des informations précises.

De ce fait, le traitement d'images est l'ensemble des méthodes et techniques opérant sur celles-ci, dans le but de rendre cette opération possible, plus simple, plus efficace et plus agréable, d'améliorer l'aspect visuel de l'image et d'en extraire des informations jugées pertinentes.

1.2 Définition d'une image

Intrinsèquement, on peut définir l'image comme étant une représentation planaire d'une scène ou d'un objet situé en général dans un espace tridimensionnel [1].

De tout temps, l'image a eu une place de choix chez l'homme car elle constitue une représentation imprimée d'un sujet quelconque, c'est également l'un des premiers moyens d'expression et de communication. Son utilisation en tant que moyen de communication s'est amplifiée au fil des temps en raison de la richesse de son contenu et la facilité de sa compréhension.

1.3 Vision assisté par ordinateur :

La vision nous permet de percevoir et d'interpréter le monde qui nous entoure. La VAO est un ensemble d'outils qui permet à l'ordinateur d'imiter la perception humaine afin d'extraire des informations d'une image brute pour pouvoir prendre des décisions. C'est un problème difficile en raison du fait que l'information disponible : des images 2D fournies par des capteurs (CCD, CMOS etc.), correspondent à une projection du monde 3D. La projection 3D-2D entraîne une perte d'informations importante, de plus, l'information disponible n'est pas parfaite (numérisation des capteurs, déformation des objectifs, bruitages). Nous pouvons alors dire que la VAO constitue une chaîne de traitements allant de l'acquisition de l'image brute jusqu'à son interprétation par la machine [2].

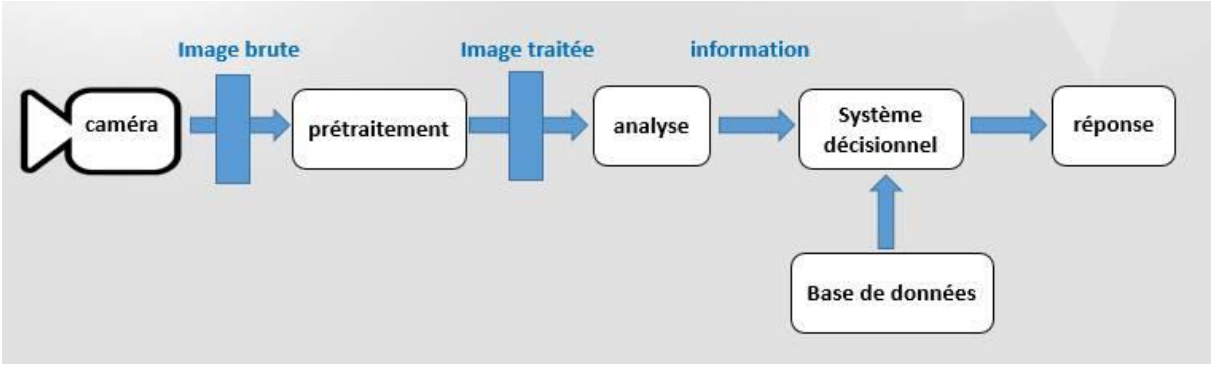


Figure 1.1 : la chaine de la VAO

La VAO est utilisée dans plusieurs domaines :

- Aide à la conduite de voiture.
- Construction et correction de cartes géographiques d'après des images satellites ou des images aériennes.
- Surveillance et évaluation de la production agricole. Il est possible de déterminer le degré de maturation des cultures, la quantité d'eau nécessaire pour l'irrigation, le rendement moyen etc.
- Reconnaissance de l'écriture.
- Reconnaissance des matricules.
- Recherche d'image par le contenu.
- Vérification et identification d'empreinte digitale.

- Analyse de la vidéo.
- Segmentation et suivi de cellules vivantes en microscopie.
- En robotique : segmentation de l'environnement, détection d'obstacles, détection, reconnaissance et suivi de cibles etc.

1.4 L'image numérique :

Contrairement aux images obtenues à l'aide d'un appareil photo, ou dessinées sur du papier, les images manipulées par ordinateur sont numériques (représentées par une série de bits), donc l'image numérique fait appel à l'informatique.

On distingue deux types d'images numériques à la composition et au comportement différent : images matricielles et les images vectorielles.

1.4.1 Images vectorielles :

Elle est composée de différents objets repérés par leurs coordonnées et comportant différentes attributs (bordure, fond, forme, coordonnées). Leur avantage c'est qu'elles peuvent être facilement redimensionnées. Leur codage dépend directement du logiciel qui a permis de les créer.

1.4.2 Images matricielles (ou images bitmap) :

Elle est composée comme son nom l'indique d'une matrice (tableau) de points à plusieurs dimensions, chaque dimension représentant une dimension spatiale (hauteur, largeur), ou autre (niveau de résolution). Dans le cas des images à deux dimensions, les points sont appelés pixels.

a) le modèle RVB :

Le modèle Rouge, Vert, Bleu, abrégé en RVB (ou RGB, de l'anglais « Red, Green, Blue ») est le modèle idéal pour expliquer l'addition de la luminosité dans les mélanges de couleur, encore appelée la synthèse additive. Le modèle RVB repose sur trois couleurs primaires qui par leurs mélanges vont restituer toutes les autres. Le modèle RVB a largement été exploité dans l'industrie dès lors qu'on s'est rendu compte que l'œil utilise ce modèle colorimétrique.

Le principe est très simple : si on éclaire le projecteur rouge, l'écran est rouge ; si on éclaire à la fois le projecteur rouge et le projecteur vert, l'écran est jaune (et deux fois plus lumineux, bien sûr). Et si les trois projecteurs sont éclairés, la neutralité de la couleur s'impose avec du blanc (ou avec du gris si les projecteurs se_sont pas très puissants) [3].

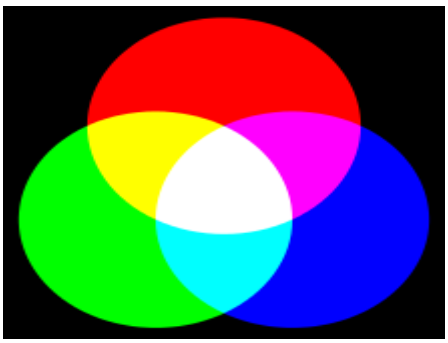


Figure1.2 : la synthèse additive

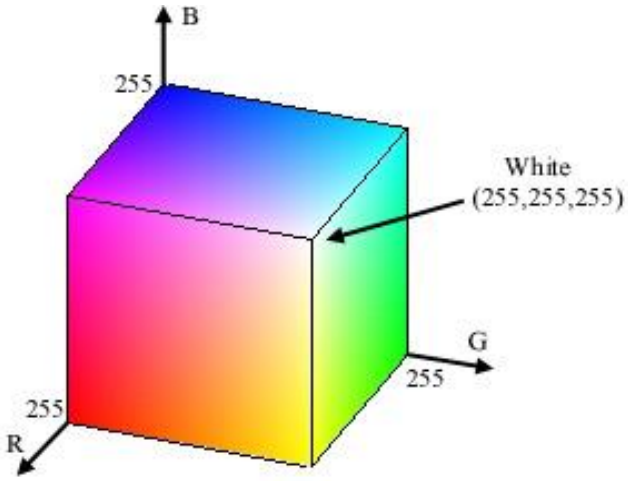


Figure1.3 : le modèle RVB sous forme de cube.

La luminosité de chaque primaire s'exprime entre 0 et 1 dans les normes de la colorimétrie. Mais dans les débuts de l'informatique, seul le mode 8 bits avec seulement 256 niveaux était disponible, donc cette notation sur 256 niveaux pour les couleurs RVB s'est naturellement imposée [3].

b) le modèle TSL :

Le modèle TSL (acronyme de Teinte, Saturation, Luminosité) est issu des travaux du peintre Albert Munsell (1858-1918), c'est le plus intuitif de tous les modèles colorimétriques que nous étudierons. Il est basé sur le ressenti de la perception humaine d'où son nom de modèle perceptuel. Chaque critère de couleur est clairement séparé, ce qui en fait le modèle le plus pratique pour la retouche d'image ou l'ajustement des couleurs.

- **Teinte :**

La teinte est mesurée par un angle de 0 à 360° autour de la roue chromatique, elle permet de sélectionner la couleur souhaitée à partir de la roue chromatique

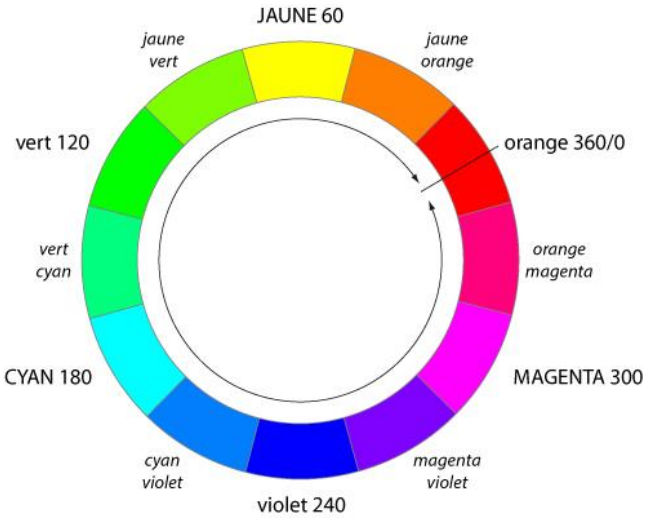


Figure1.4 : la roue chromatique

- **Saturation :**

La saturation mesure la pureté d'une couleur, elle s'évalue sur une échelle allant de 0 (pas de sensation colorée) à 1 (l'impression colorée est au maximum permis par le système de reproduction des couleurs), cette grandeur s'exprime en pourcentage [4].

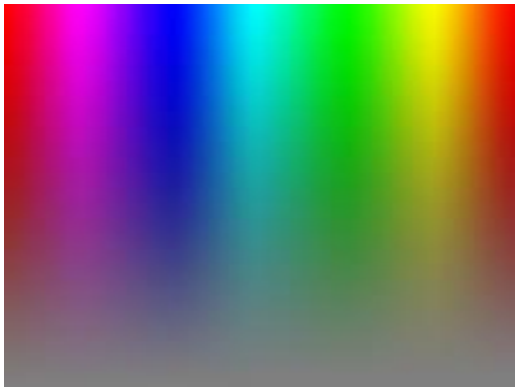


Figure1.5 : les couleurs avec des différentes saturations (saturation max=1 en haut, saturation min=0 en bas).

- **luminance :**

La luminance est mesurée entre le noir (pas de lumière ou valeur 0) et le blanc (lumière maximum ou valeur 1) [5].

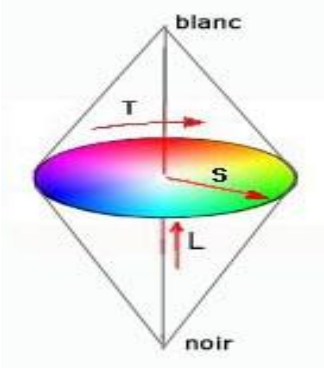


Figure1.6 : axe de luminance

c) le modèle CMJN :

Le système CMJN ((Cyan, Magenta, Jaune, Noir) ou de l'anglais CMYK «Cyan, Magenta, Yellow, Black»)) a été conçu pour l'imprimerie. Il s'appuie sur le principe de la synthèse soustractive. C'est-à-dire un système où le mélange de couleurs amène une résultante plus sombre (moins de lumière, donc "soustraction" de lumière).

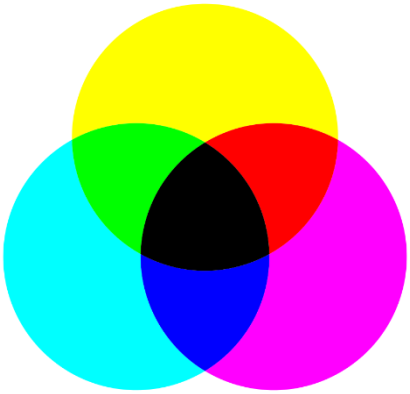


Figure1.7 : la synthèse soustractive

d) le modèle niveau de gris :

On peut considérer le modèle niveau de gris comme une simplification du modèle CMJN dont on ne tiendrait compte que d'une seule couche : celle du noir.

Le niveau de gris est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allons du noir au blanc en passant par un nombre fini de niveaux intermédiaires, donc pour représenter les images a niveaux de gris, on peut attribuer a chaque pixel de l'image une valeur correspondant à la quantité de lumière renvoyée. Cette valeur peut être compromise par exemple entre 0 et 255.

Chaque pixel n'est donc plus présenté par un bit, mais par un octet. Pour cela il faut que le matériel utilisé pour afficher l'image soit capable de produire les diffèrent niveaux de gris correspondant [6].

Figure1.8 : Les Niveaux de Gris

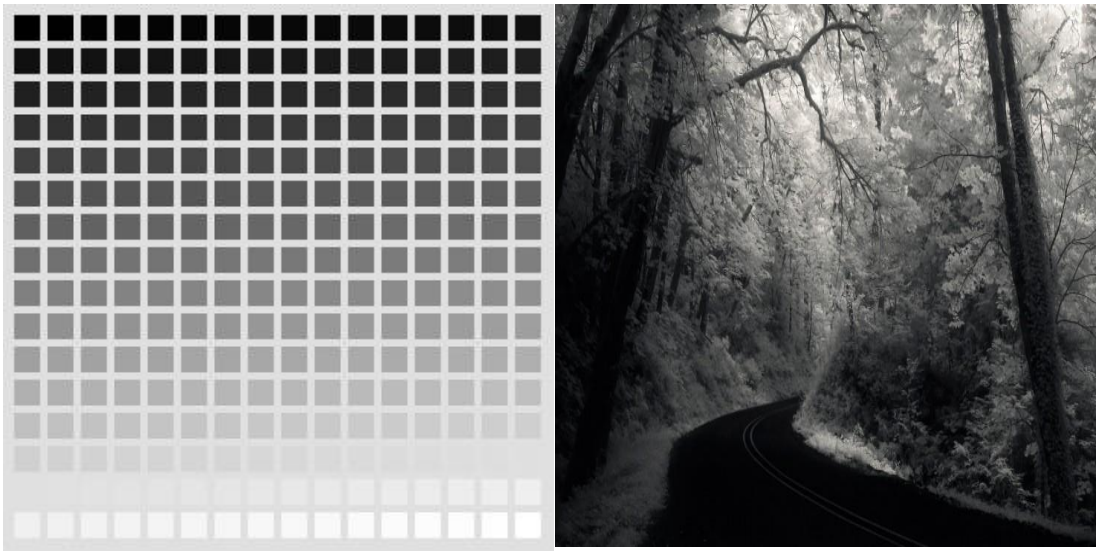


Figure1.9 : exemple image en niveau de gris**.**

Le nombre des niveaux de gris dépend du nombre de bits utilisés pour décrire la couleur de chaque pixel de l'image. Plus ce nombre est important, plus les niveaux possibles sont nombreux.

1.5 les caractéristiques d'une image numérique :

L'image est un ensemble structuré d'informations caractérisé par les paramètres suivants :

1.5.1 Pixel :

Contraction de l'expression anglaise (Picture éléments «élément 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 [7].

La lettre A, par exemple peut etre affichée comme un groupe de pixels (**Figure1.10**)

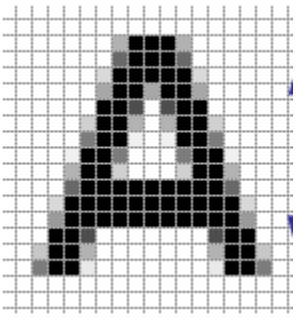


Figure1.10 : groupe de pixels

La quantité d'information que véhicule chaque pixel donne des nuances entre images monochrome et images couleurs. Dans le cas d'une image monochrome, chaque pixel est codé sur un octet, et la taille mémoire nécessaire pour afficher une telle image est directement liée à la taille de l'image

Dans une image couleur (RVB), un pixel peut être représenté sur trois octets : un octet pour chacune des couleurs : rouge (R), vert (V) et bleu (B).

1.5.2 Dimension :

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels).

Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image [8].

1.5.3 Résolution :

La résolution d'une image est définie par un nombre de pixels par unité de longueur. Plus ce nombre 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. Augmenter la résolution peut entraîner des temps de visualisation et d'impression plus longs, et conduire à une taille trop importante du fichier contenant l'image et à de la place excessive occupée en mémoire [9].

1.5.4 Bruit :

Le bruit est un terme issu du domaine de l'acoustique et désigne un signal parasite. Que ce soit pour le son ou pour l'image, le principe est identique : sur tout signal de base vient s'adjoindre un ensemble d'informations parasites aléatoires [10].

Un bruit dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, Il existe différentes causes de bruit. On peut notamment citer le contexte d'acquisition (lumière, perturbation de capteurs), les étapes d'échantillonnage et de quantification qui apporte leur propre perturbation, mais aussi l'étape de transmission (perte ou corruption de données). On peut bien remarquer ce bruit notamment sur la transmission analogique de la télévision.

1.5.5 Contours :

Les contours représentent la frontière entre les objets et l'image, ou la limite entre deux pixels dont les niveaux de gris représentent une différence significative. Les textures décrivent la structure de ceux-ci. L'extraction de contour consiste à identifier dans l'image les points qui séparent deux textures différentes [11].

1.5.6 Luminance :

C'est le degré de luminosité des points de l'image. Elle est définie aussi comme étant le quotient de l'intensité lumineuse d'une surface par l'aire apparente de cette surface,

pour un observateur lointain, le mot luminance est substitué au mot brillance, qui correspond à l'éclat d'un objet. Une bonne luminance se caractérise par :

- Des images lumineuses (brillantes).
- Un bon contraste : il faut éviter les images où la gamme de contraste tend vers le blanc ou le noir, ces images entraînent des pertes de détails dans les zones sombres ou lumineuses.
- L'absence de parasites.

1.5.7 Contraste :

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. Le contraste est défini en fonction des luminances de deux zones d'images.

Si L1 et L2 sont les degrés de luminosité respectivement de deux zones voisines A1 et A2 d'une image, le contraste Ce défini par le rapport :

$$C = \frac{\mathrm{L1} - \mathrm{L2}}{\mathrm{L1} + \mathrm{L2}}$$

1.5.8 Histogrammes :

Un histogramme est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. L'histogramme est très utile pour contrôler l'exposition d'une image.

• A l'acquisition, il permet de contrôler et affiner les réglages de prise de vue.

• Pour le traitement, il permet de corriger ou modifier l'exposition de l'image, ainsi que l'échelle des couleurs.

Par exemple : améliorer le contraste, corriger une image sous-exposée, renforcer la composante rouge, corriger la non-linéarité du capteur....

• En utilisant judicieusement l'histogramme, on peut faire apparaître les détails et les nuances acquises par le capteur et présentes dans le fichier, mais non visibles à l'œil.

a) Histogramme des images niveaux de gris :

L'histogramme des images en niveaux de gris montre toutes les valeurs des pixels entre le blanc (255) et le noir (0).

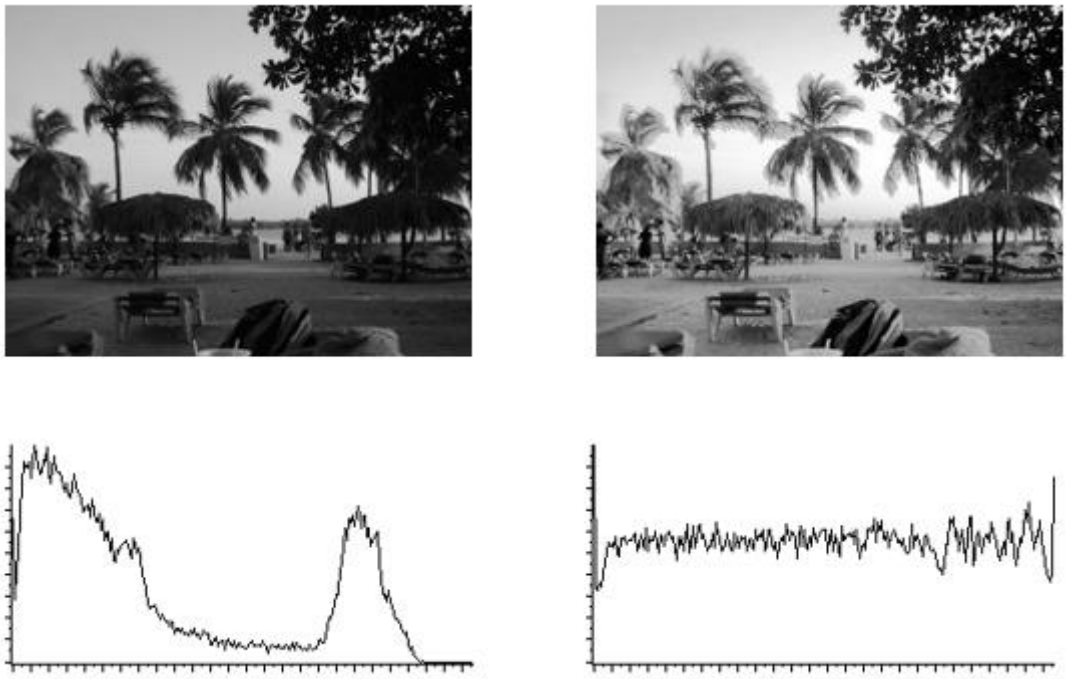


Figure 1.11 : exemple d'histogramme avec différents niveaux de gris.

b) Histogramme des images couleurs :

Pour une image en couleur, chaque plant (rouge, vert, bleu) a son histogramme comme montre la **Figure 1.12.**

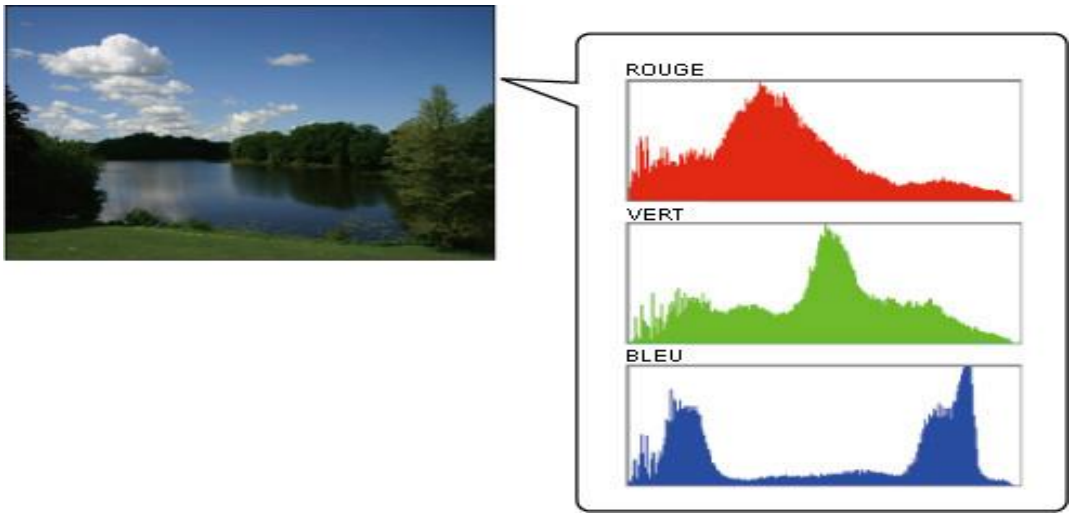


Figure 1.12 : Histogrammes des trois plants de l'image en couleur

1.6 Filtrage :

Le filtrage d'image a pour but d'améliorer la qualité d'une image numérique.

Chaque filtre cherche à atténuer un type de défauts plus précis. Il n'y a pas de filtre universel capable de corrigé tous les défauts, il faut choisir les bons filtres suivants les défauts que nous désirons corriger.

Il existe deux types de filtrage :

- Filtrage linière ou la transformation d'un pixel est le fruit d'une combinaison linière des pixels voisins.
- Filtrage non linière ou les pixels voisins interviennent suivant une loi non linière.

1.6.1 Filtres linéaires :

a) Filtre moyen :

Ce filtre consiste à considérer chaque point de l'image et d'en faire la moyenne avec les huit pixels voisins. Ceci va adoucir l'image en réduisant les fluctuations des niveaux de gris.

Ce type de filtre utilise la moyenne non pondérée des voisins, il est la forme d'un masque tel que :

$$H=1/9\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Ce masque ne changeras pas de taille, ce seras toujours une matrice 3×3 (ou 5×5, 7×7…..) qui va se déplacer sur toute l'image.

Pour chaque pixel P de l'image, on effectuera la convolution entre le masque et la sous-image de même taille que le masque est centrée sur P, et on remplacera la valeur de P par le résultat obtenu [13].

b) Filtre gaussien :

Il représente une bonne qualité de résultats ainsi qu'une facilité de mise en œuvre.

Il utilise la moyenne pondérée des voisins et peut être mis sous la forme du masque suivant :

$$Hg=1/16\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

1.6.2 Filtres non linéaires :

a) Filtre médian :

L'idée principale du filtre médian est de remplacer chaque entrée par la valeur médiane de son voisinage.

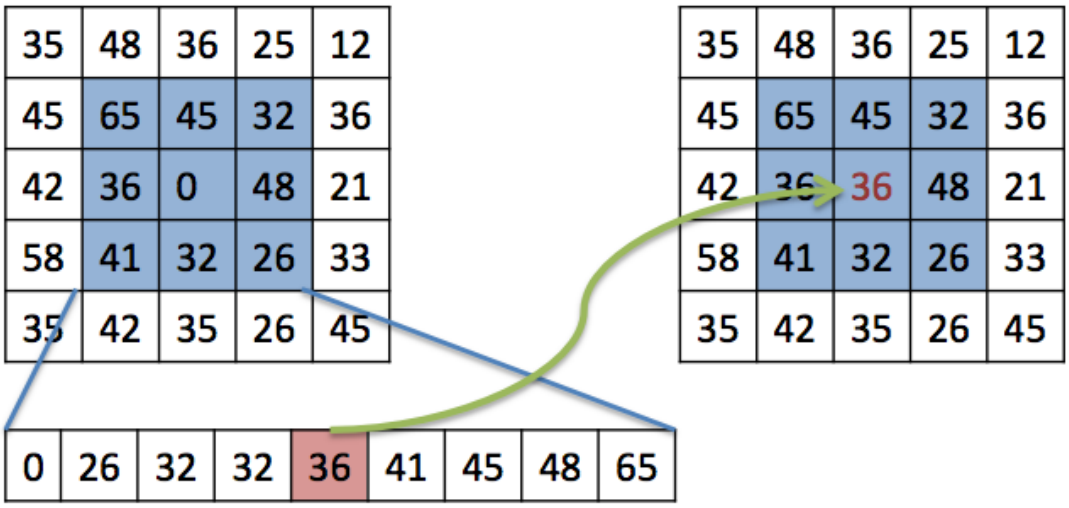


Figure 1.13 : fonctionnement du filtre médian.

b) Filtre maximum :

Le filtre maximum est un filtre non linéaire utilisé pour supprimer le bruit de type "poivre" dans les images, il fonction sous le même principe que le filtre médian, sauf qu'il utilise la valeur maximale au lieu que la valeur médiane. Pour chaque pixel (i, j) :

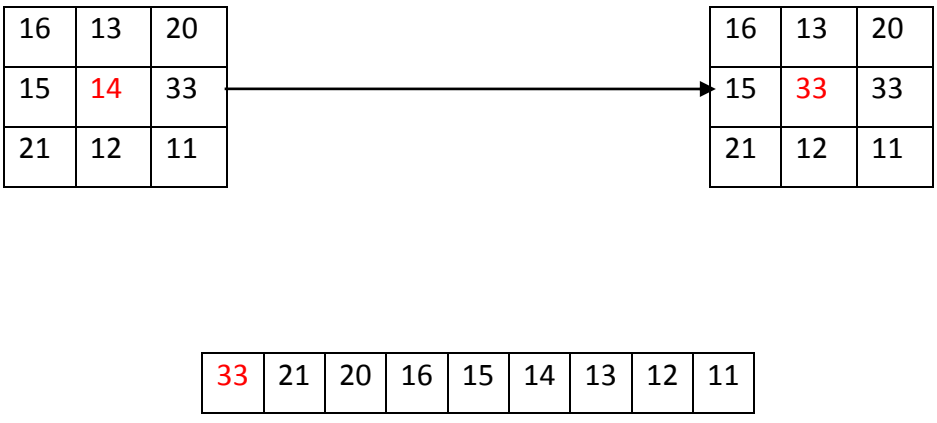


Figure 1.14 : fonctionnement du filtre maximum.

c) Filtre minimum :

Ce filtre est utilisé pour éliminer le bruit de type 'Sel'. Il a le même principe que le filtre médian et le filtre maximum, mais contrairement au filtre maximum il utilise la valeur minimale au lieu que la valeur maximale.

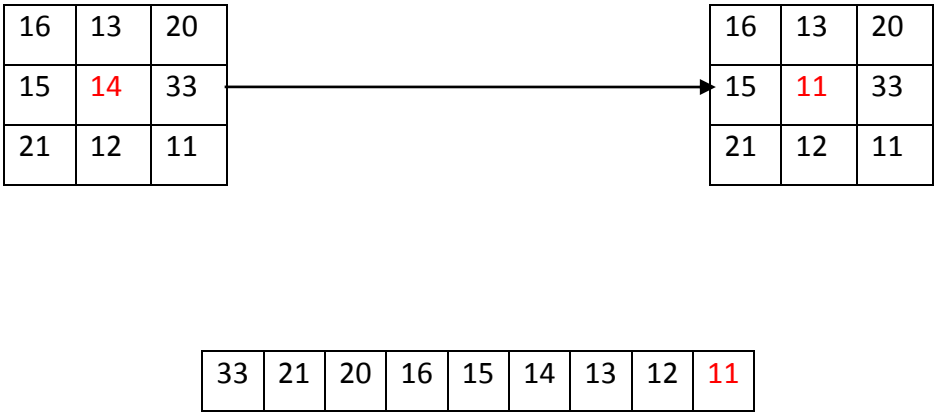


Figure 1.15 : fonctionnement du filtre minimum.

1.6.3 Les filtres morphologiques :

Les filtres morphologiques sont souvent utilisés pour éliminer des pixels isolés considérés comme un bruit dans une image binarisme. Ces méthodes utilisent un élément structurant. Parmi ces opérations morphologiques il y a la dilatation, l'érosion, l'ouverture et la fermeture mathématiques.

a) La dilatation :

Elle permet d'éliminer les pixels blancs isolés mais ajoute des pixels noirs au contour des objets présents dans l'image. Le résultat de cette opération est l'augmentation de la taille de ces objets.

b) L'érosion :

Elle permet d'éliminer les pixels noirs isolés au milieu des parties blanches de l'image. Le résultat de cette opération est la diminution de la taille des objets présents dans l'image.

c) L'ouverture :

L'ouverture est constituée par une opération d'érosion suivie d'une dilatation. Elle permet de retrouver les taches noires dans l'image.

d) La fermeture :

La fermeture est l'opération inverse de l'ouverture, qui consiste à faire subir à l'image une dilatation suivie d'une érosion. Elle permet d'éliminer les blancs qui se trouvent dans l'objet [12].

1.7 la binarisation :

La binarisation (seuillage) est la technique de classification la plus simple ou les pixels de l'image sont partagés par un seul seuil «S» en deux classes, ceux qui appartiennent au fond et ceux qui appartiennent à la scène (l'objet**),** l'image est alors séparée en deux classes de façon à ce que seul l'information comprise entre 0 et «S» est retenue, ou vice-versa [13].

Il existe deux grandes techniques de sélection du seuil «S» : binari7sation globale, binarisation local.

1.7.1 la binarisation globale :

Dans cette technique on applique le même seuil sur tous les pixels de l'image. Le choix du seuil dépend de la qualité de l'image. Ce dernier est calculé de la manière suivante [13] :

$$g(x,y) = \begin{cases} 0 & f(x,y) \leq S \\ 255 & f(x,y) > S \end{cases}$$

Avec : $\boldsymbol{S}$ représente le seuil, $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y})$ l'image d'entrée et $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y})$ l'image binarisée (sortie).

1.7.2 binarisation locale :

Elle consiste à modifier la valeur de chaque pixel selon son voisinage, mais l'inconvénient de cette technique c'est qu'elle est lente, car la binarisation de chaque pixel nécessite l'analyse de son visionnage [13].

1.8 Quelques applications concrètes de traitement d'images :

- Contrôle de présence/absence : Sur des chaînes de production, on vérifie en bout de chaîne avec une caméra vidéo la présence d'une pièce dans un ensemble plus complexe. Pour cela bien souvent il suffit de faire un simple seuillage dans une région spécifique
- Contrôle du niveau de maturation des fruits sur une chaîne de conditionnement. Il s'agit de reconnaître à la couleur et à la texture du fruit son degré de maturité et donc la catégorie sous laquelle il sera emballé puis vendu.
- Construction et correction de cartes géographiques d'après des images satellites ou des images aériennes : On recale d'après des informations topographiques les images reçues, puis on les met sur la carte en correspondance avec les informations trouvées dans l'image : voies de communication, voies et plans d'eau, parcelles agricoles...
- Surveillance et évaluation de la production agricole : Il est possible de déterminer le degré de maturation des cultures, la quantité d'eau nécessaire pour l'irrigation, le rendement moyen... On peut ainsi établir des prévisions à large échelle de la récolte à venir.
- Reconnaissance d'écriture : La reconnaissance de l'écriture manuscrite progresse de jour en jour. Elle est suffisamment opérationnelle pour que la majorité des adresses, même manuscrites, soient reconnues automatiquement sur le courrier postal.
- Recherche d'images par contenu : L'objectif de cette technique est de rechercher, parmi une base de données d'images, les images similaires à une image exemple, ou ayant certaines caractéristiques, par exemple rechercher toutes les images comportant un vélo.
- Segmentation et suivi de cellules vivantes en microscopie : Cela permet d'analyser le comportement d'une population de cellules et ainsi de détecter certaines anomalies.

1.9 Conclusion :

Nous avons introduit dans ce chapitre les notions de base de la vision par ordinateur qui servent de fondement à la compréhension de différentes techniques de traitement d'images, nous avons présenté quelques-unes qui nous semble les plus courantes dans le processus du traitement et analyse d'image. Les prétraitements d'images permettent d'améliorer la qualité de l'image en vue les filtrages.

Dans le chapitre suivant nous allons détailler l'un des objectifs de la vision assisté par ordinateur qui est la reconnaissance des panneaux de signalisation routière.

Chapitre 3: Implementation, test et résultats

Nous présentons un algorithme qui consiste à la reconnaissance des panneaux de signalisation en temps réel basée sur la vision artificielle. Ce chapitre présente tout d'abord la partie logicielle et matérielle utilisée pour le développement de ce projet, ensuite la partie test et résultats expérimentaux.

3.1 Environnement de développement

3.1.1 Environnement matériel :

Un ordinateur ACER avec les caractéristiques suivantes :

- Processeur : Intel(R) Pentium(R) CPU -2117U @ 1.80GHz
- RAM : 4,00 Go
-Disque Dur 500 Go
-Système: Windows 8.

La Camera :

Nous avons utilisé la camera de smartphone LG-G3.

-Résolution de l'image (480 X 640) VGA

3.1.2 Environnement logiciel

- **Logiciel de développement :** Qt Creator C++
- **Bibliothèque graphique :** Open CV (Open Computer Vision)

3.2 Qt Creator C++

Qt Creator est un environnement de développement intégré libre et gratuit permettant la programmation en langage C++, QML et JavaScript. Cet IDE, nativement intégré dans le framework Qt, permet de concevoir des applications et interfaces multiplateformes notamment compatibles avec iOS et Android.

Parmi ses nombreux outils intéressants, vous disposerez notamment d'un éditeur avec la coloration syntaxique et l'auto-complétion du code. Un débogueur intégré ainsi qu'un outil de contrôle de version sont aussi disponibles. Sur ce dernier point, le logiciel est compatible avec les systèmes Git, Perforce, Mercurial ou encore Subversion.

Qt et aussi une bibliothèque graphique qui comporte beaucoup de module Gui (Graphical user interface) bien qu'il soit possible de développer en C++ avec Qt en utilisant d'autre IDE (comme Code::Blocks) il est recommandé d'utiliser l'IDE Qt Creator. Il est particulièrement optimisé pour développer avec Qt. En effet, c'est un programme tout-en-un qui comprend entre autres :

- un IDE pour développer en C++, optimisé pour compiler des projets utilisant Qt (pas de configuration fastidieuse) .
- un éditeur de fenêtres, qui permet de dessiner facilement le contenu des interfaces à la souris.
- une documentation indispensable pour tout savoir sur Qt.

Notons enfin que Qt Creator propose une interface soignée simple à prendre en main malgré les connaissances nécessaires pour bien utiliser des outils disponibles.

Voici à quoi ressemble Qt Creator quand on le lance pour la première fois (figure 3.1).

Figure3.1 Qt creator à l'ouverture

3.3 La librairie OpenCv :

OpenCV (Open Computer Vision) est une bibliothèque graphique libre, initialement développée par Intel, spécialisée dans le traitement d'images en temps réel, La bibliothèque OpenCV met à disposition de nombreuses fonctionnalités très diversifiées permettant de créer des programmes partant des données brutes pour aller jusqu'à la création d'interfaces graphiques basiques.

En résumé elle est :

- Une librairie open source de traitement et analyse d'images et vidéos avec des Interfaces pour les principaux langages de programmation C, C++, Java, Python ...
- Optimisée pour les applications temps réelles.
- Fournit une API bas et haut niveau.
- Utilisé aussi bien dans les laboratoires de recherche que dans l'industrie.

Fonctions :

- Manipulation d'images (chargement, sauvegarde, copie, conversion...).
- Manipulation et acquisition de vidéos.
- Manipulations de matrices et algèbre linéaire.
- Structure de données utilitaires variées (listes, files, ensembles, graphes...).
- Traitement d'images (filtrage, détections de discontinuités, morphologie mathématique...)
- Analyse d'images (composantes connexes, ajustement de primitives, transformée de distance...).
- Vision (calibration de caméra, stéréovision, recherche d'association...)
- Reconnaissance de forme.
- Interface graphique (affichage d'images, de vidéos, gestion des évènements...) .

3.4 Interfaces de l'application

3.4.1 Interface principale :

Nous avons créé deux types d'applications (Figure 3.2 et 3.3), l'une pour détailler les fonctions utilisées dans notre projet, et l'autre pour les vidéos (temps réel).

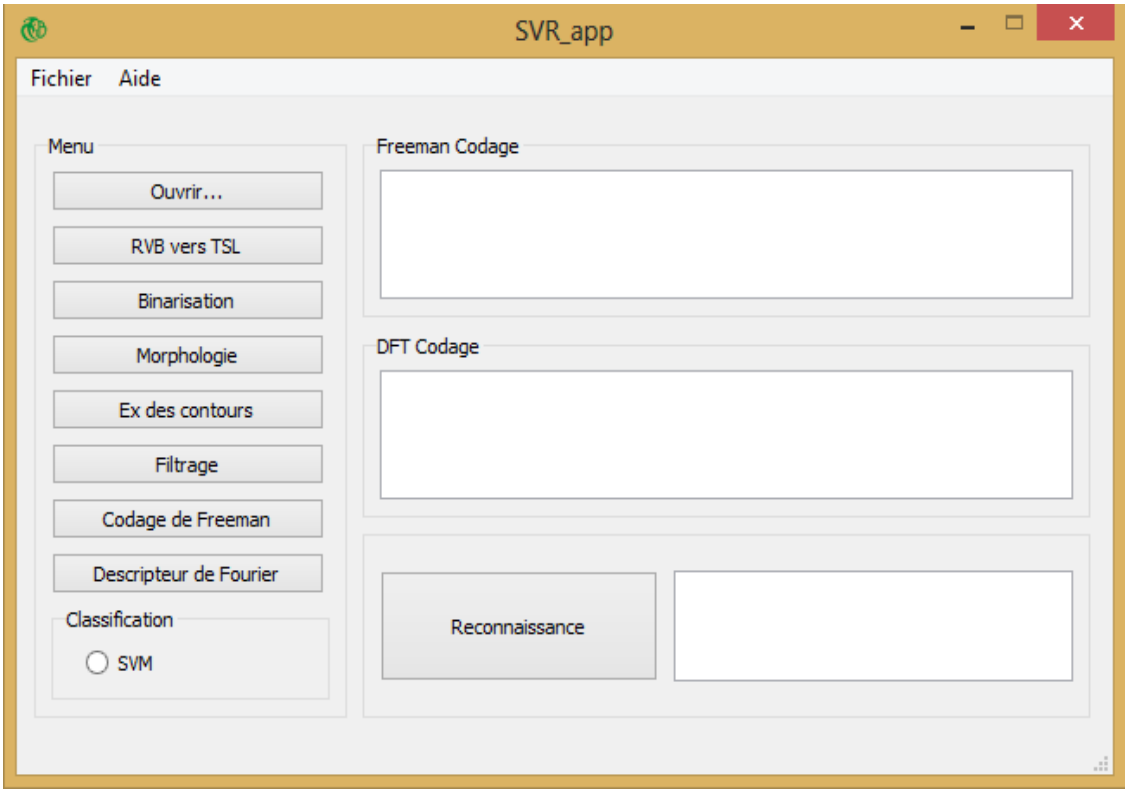


Figure 3.2 : Interface de l'application n°1

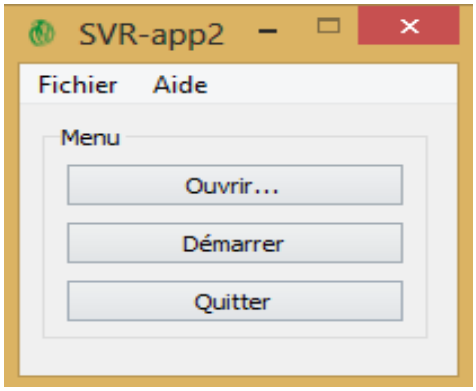


Figure 3.3: Interface de l'application n°2

3.4.2 Description de l'interface :

Nous allons découvrir l'espace de travail de notre application :

- **1er application :**

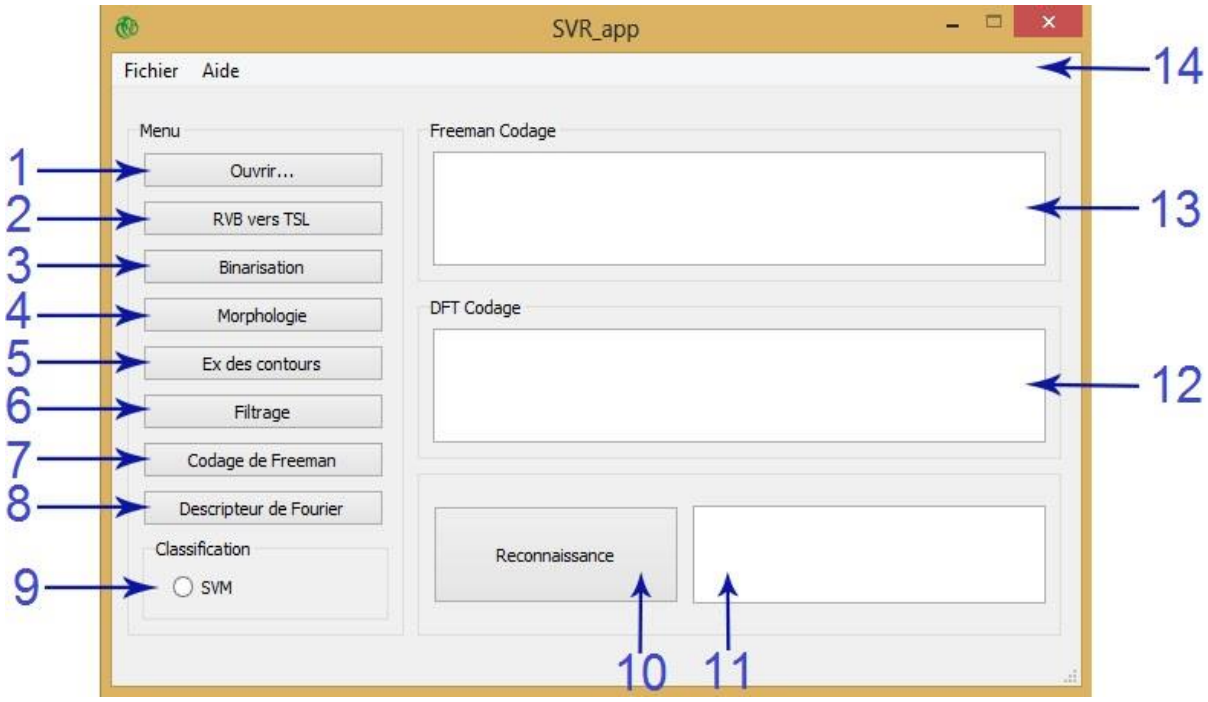


Figure 3.4: interface graphique n°1

1- Bouton d'ouverture d'image.
2- Bouton de conversion RVB en TSL.
3- Bouton de binarisation.
4- Bouton de morphologie.
5- Bouton pour extraire les contours.

6- Bouton de filtrage.
7- Bouton de calcul du descripteur Freeman.
8- Bouton de calcul du descripteur de Fourier (DFT).
9- Case d'option, choix de classificateur (svm).
10- Bouton de Reconnaissance.
11- Affichage du type des Panneaux.
12- Affichage du matrice descripteur Fourier (DFT).
13- Affichage du matrice descripteur Freeman.
14- barre de menu.

- **2 éme application:**

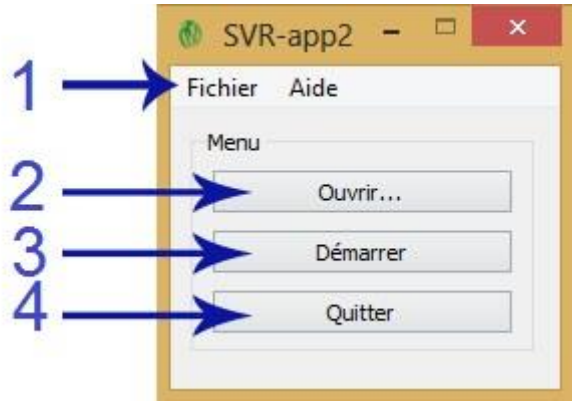


Figure 3.5: interface graphique n°2

1- barre de menu.
2- Bouton d'ouverture de vidéo.
3- Bouton pour démarrer la reconnaissance.
4- Bouton pour quitter l'application.

3.5 Résultats et Discussions :

Le travail que nous avons développé dans ce projet pour la reconnaissance des panneaux routiers est basé sur les différentes étapes déjà représentées par la (figure 2.1) dans le chapitre 2. Les résultats de ces étapes sont représentons comme suit :

3.5.1 L'acquisition d'image:

Appuyer sur le bouton ouvrir pour choisir une image. (**Figure** 3.6).

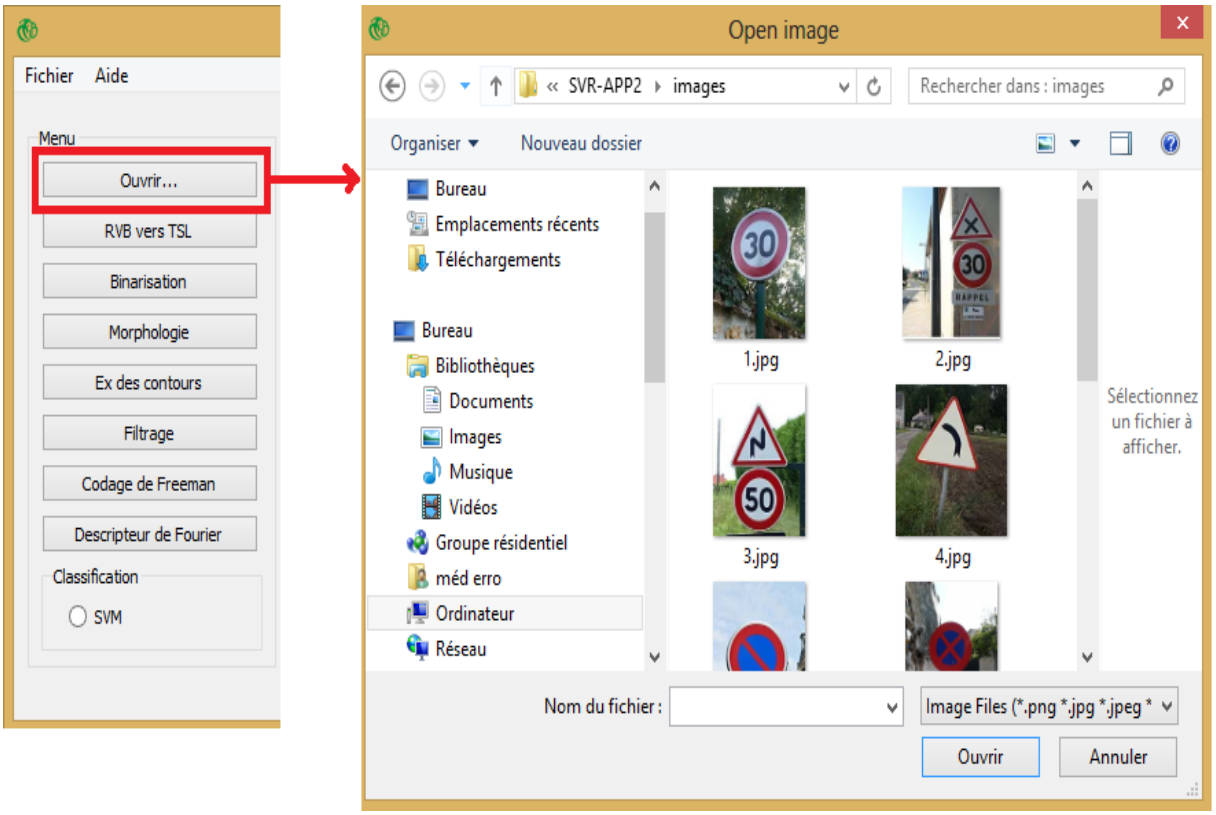


Figure 3.6: ouverture de l'image.

3.5.2 Conversion RVB vers TSL :

Nous allons tout d'abord convertir notre image qui est en RVB en TSL. Nous passons de l'image RVB en HSV car ainsi nous pourrons travailler sur la teinte et la saturation de la couleur. Ce qui nous permet, en partie seulement, d'éloigner les problèmes liés à l'éclairage. Le résultat de la conversion est représenté par la figure 3.7.

Figure 3.7: image en espace TSL

3.5.3 : Binarisation :

Les couleurs ciblés sont (rouge,bleu) .nous verrons sur notre image binarisée tous les éléments rouges et bleus apparaître en blanc sur un fond noir. Le résultat de binarisation est représenté dans la figure 3.8 suivante.

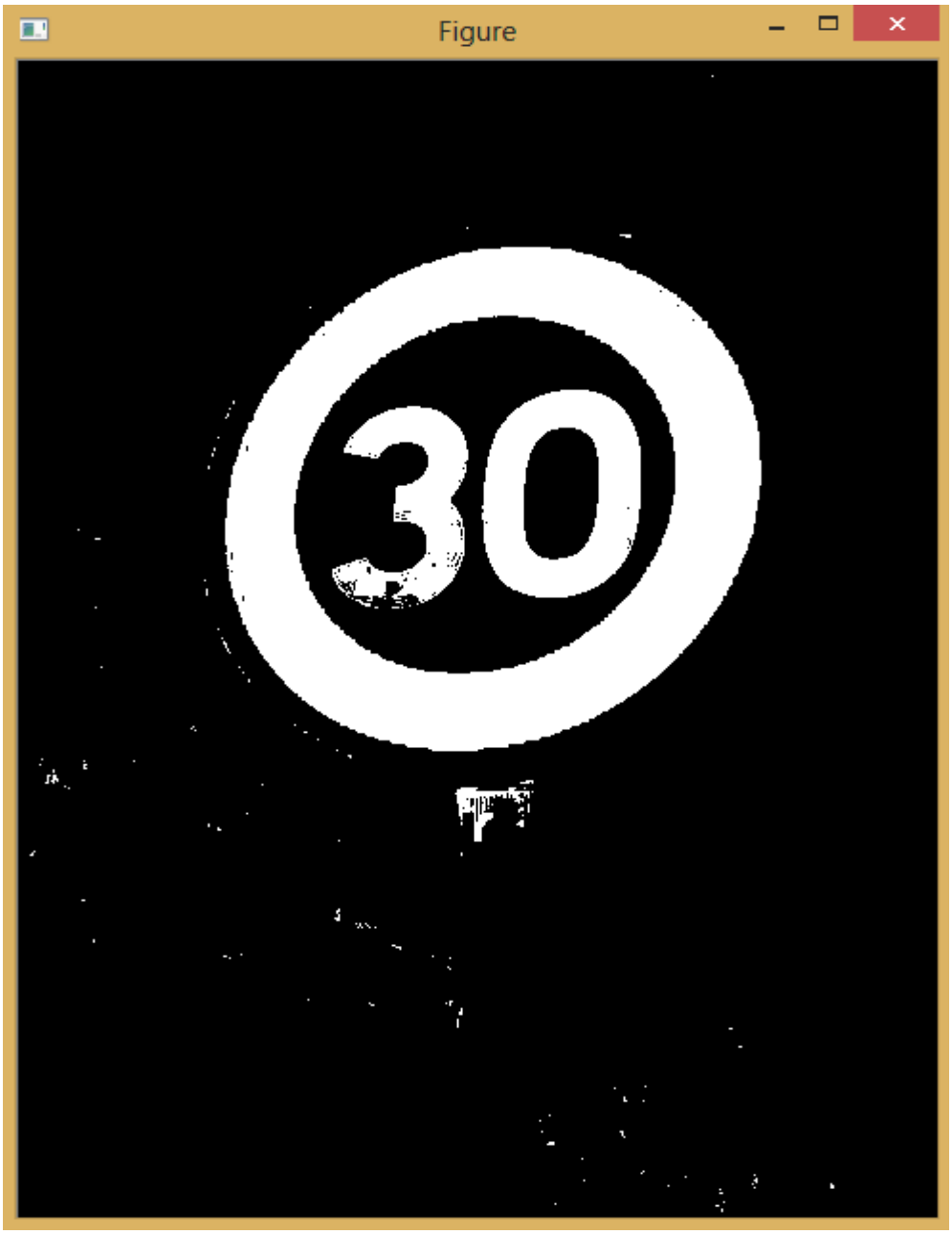


Figure 3.8: image binarisée.

3.5.4 Morphologie :

Nous avons maintenant une image en noir et blanc qui nous indique où se trouve notre couleur dans l'image par des « tâches blanches ». Nous voulons suivre un objet de couleur, il va donc nous falloir l'isoler. Nous allons effectuer une « erosion ». Cela nous permet de supprimer les pixels « isolés » qui ne correspondent pas à notre objet de couleur. Ensuite nous effectuons une « dilatation » qui nous permet de renforcer les groupes denses de pixels (donc notre objet). (Figure 3.9).

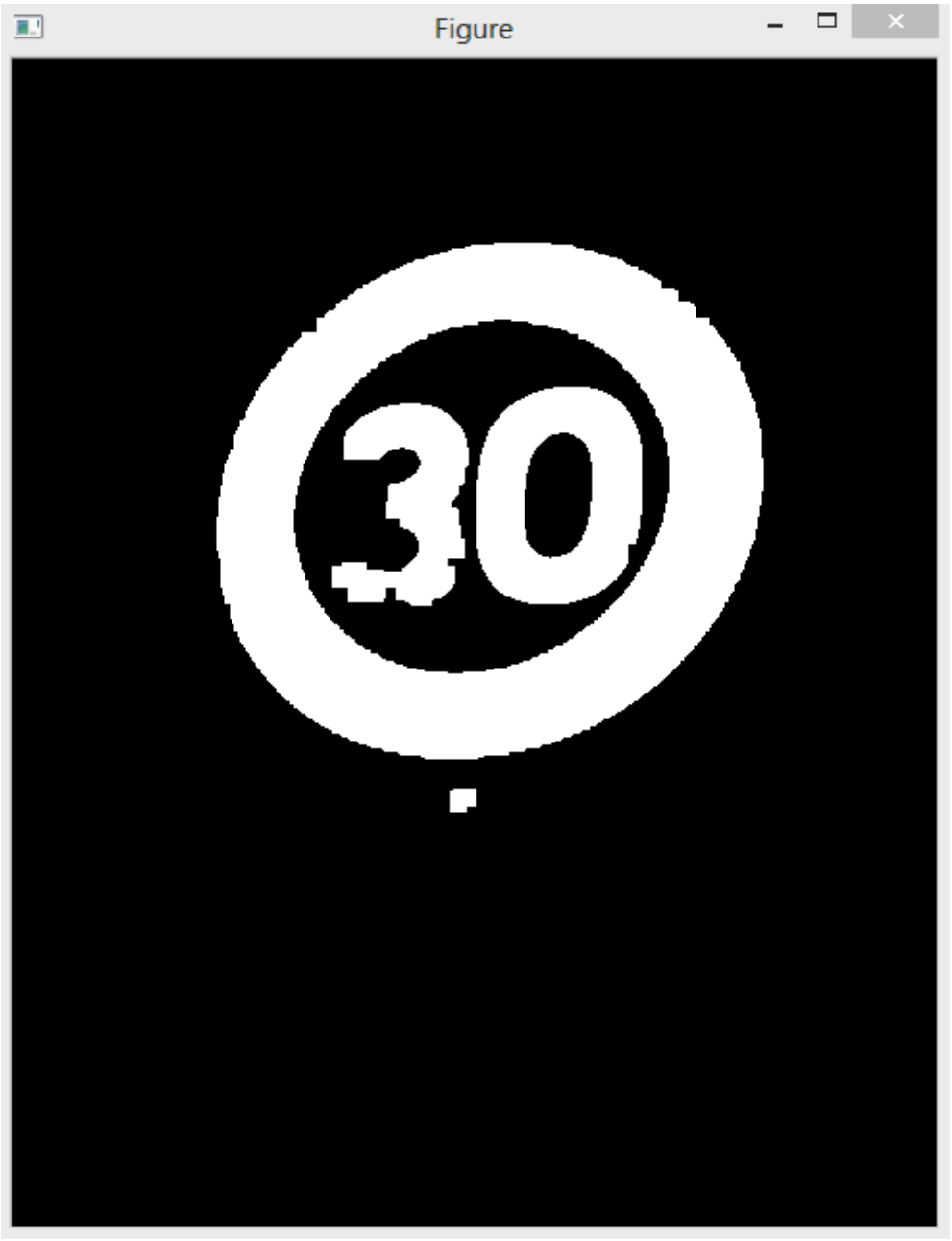


Figure 3.9*:* filtrage morphologique.

3.5.5 Extraction de contours:

Dans cette étape nous utilisons l'algorithme de suivi de contour pour détecter les contours existent dans l'image. (Figure 3.10).

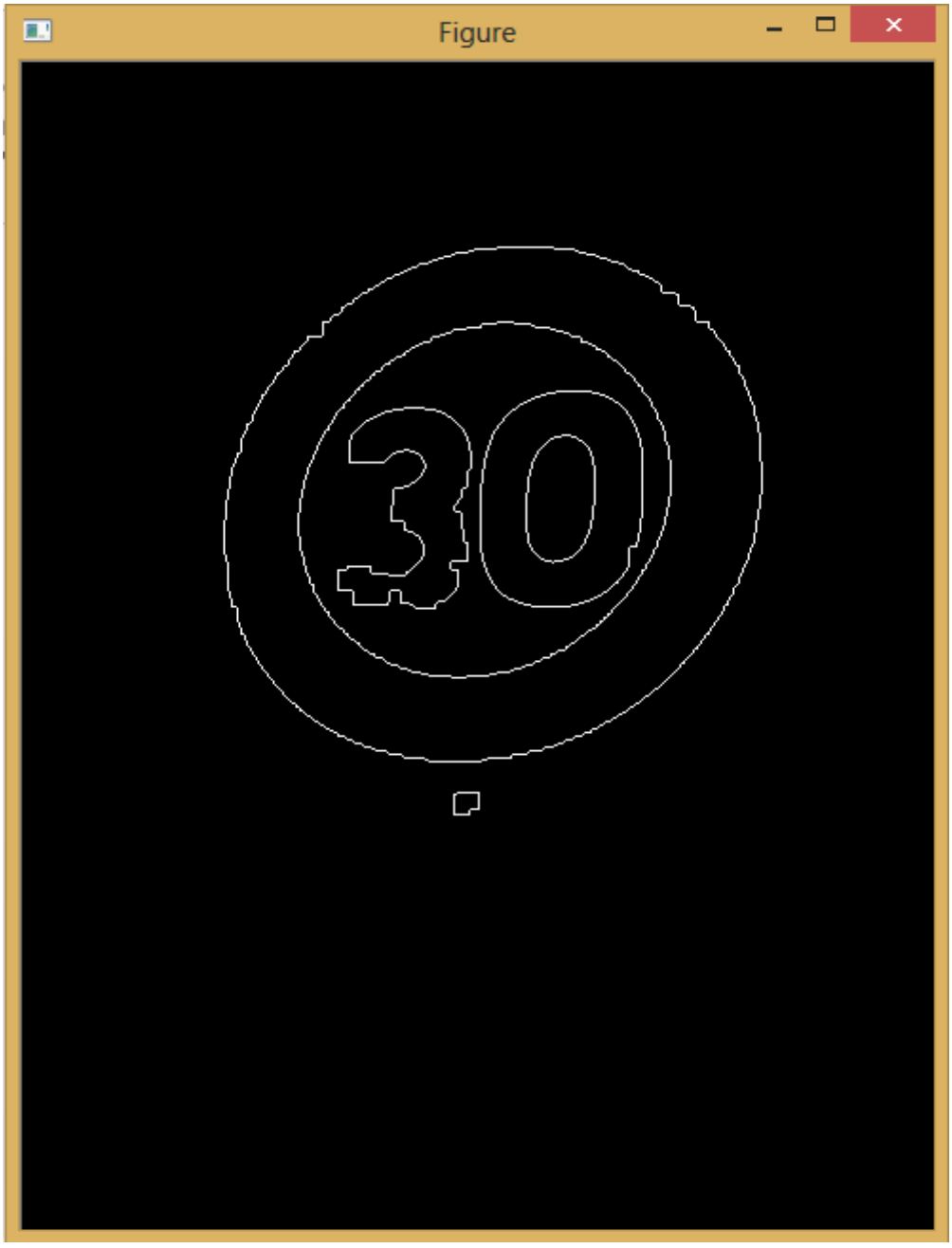


Figure 3.10: extraction de contours.

3.5.6 Filtrage du contours:

Dans cette étape Nous avons utilisé un filtrage qui permet de réduire de manière significative la quantité de données et élimine les informations qu'on peut juger moins pertinentes.(Figure 3.11).

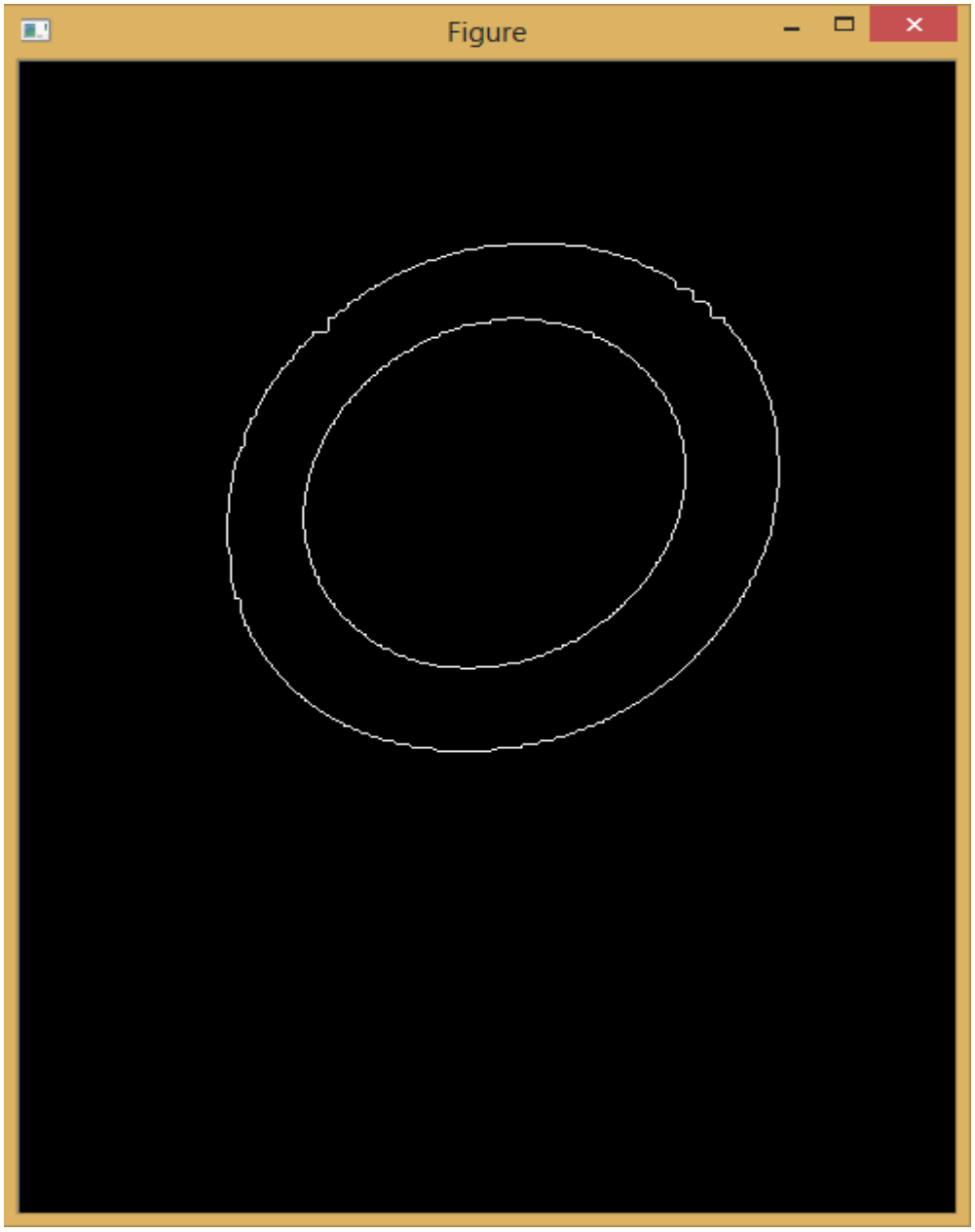


Figure 3.11: filtrage de contour

3.5.7 résultat du descripteur:

Dans cette partie on a appliqué La (DFT) et (Freeman) dans le but d'obtenir un codage de contour. Le résultat de cette étape est représenté par la figure 3.12

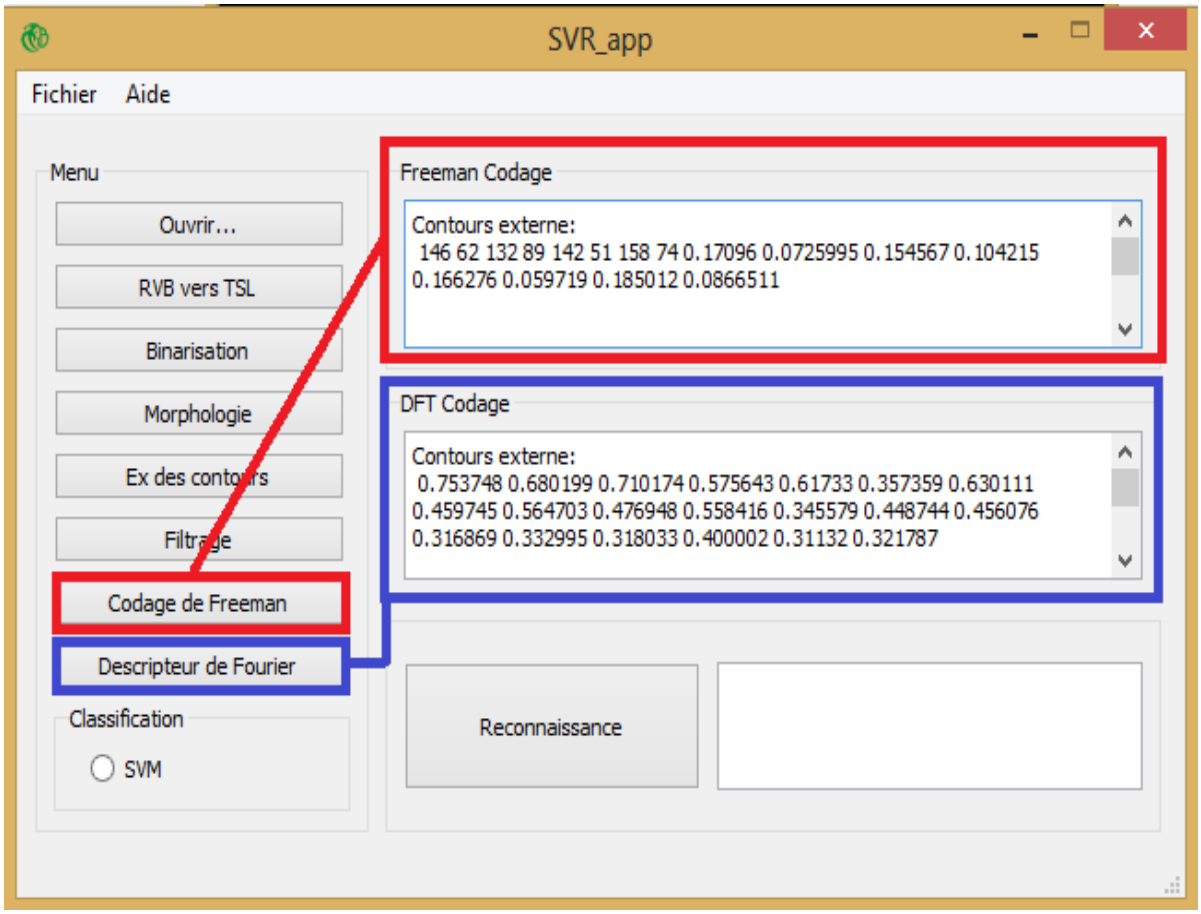


Figure 3.12: résultat du descripteur.

3.5.8 La reconnaissance de Panneau de signalisation :

Dans cette étape qui représente la reconnaissance des panneaux, toutes les étapes de la chaine de reconnaissance décrites précédemment ont été appliquées. Une base de données a été introduite et utilisée pour faire la classification. Cette base et chargée automatiquement avec le démarrage du programme .Les résultats de la classification est représenté par les (Figures 3.13,3.14,3.15)

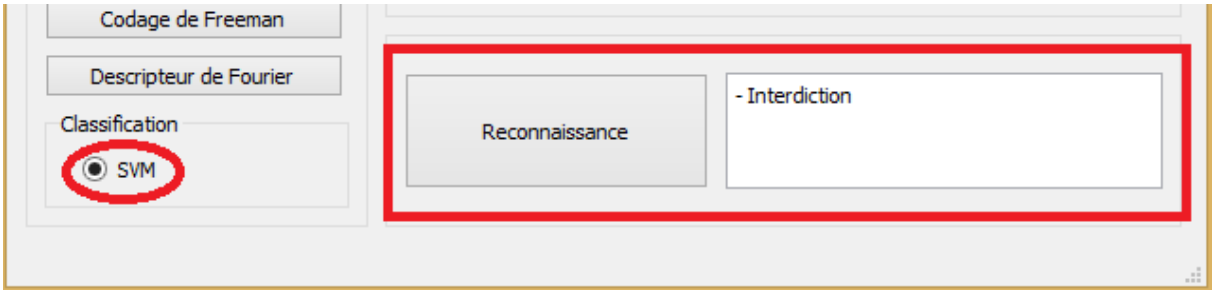


Figure 3.13 Résultat de la reconnaissance.

Figure 3.14 Résultat de la reconnaissance.

Figure 3.15 Résultat de la reconnaissance.

3.5.9 La reconnaissance en temps réel :

Dans cette partie nous utilisons la 2eme application pour la reconnaissance des panneaux en temps réel (voir Figure 3.5). Le résultat de cette étape est représenté par la (figure 3.17)

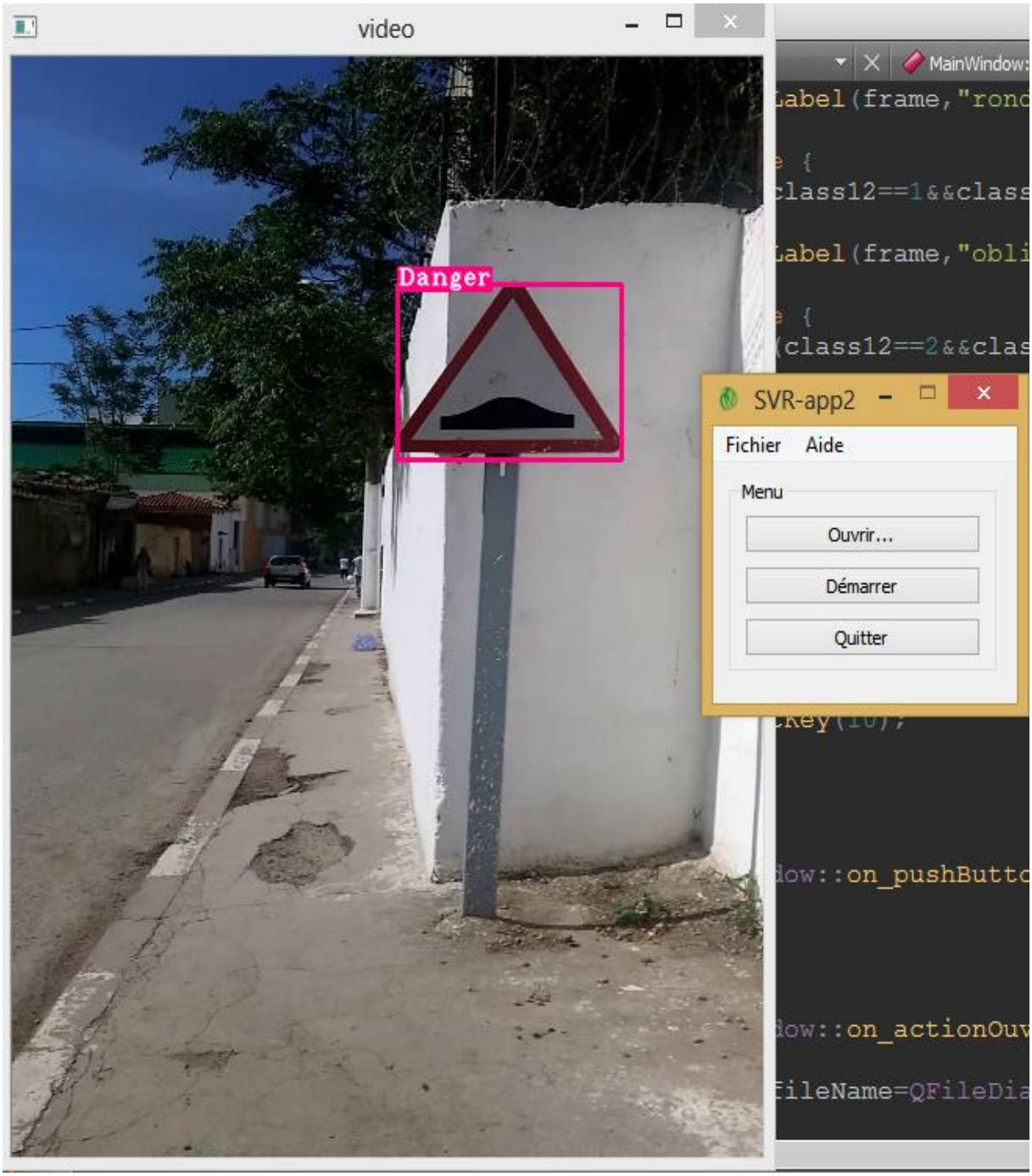


Figure 3.16 Résultat de la reconnaissance en temps réel.

3.6 Résultats des tests:

Apres le test des panneaux de signalisation de la base de données nous sommes arrivé a une reconnaissance de 100% pour chaque catégorie(figure 3.17). ce qui représente le meilleur résultat obtenu. Nous avons aussi appliquer notre travail sur une base d'images test constituée de 50 images (figure3.18).

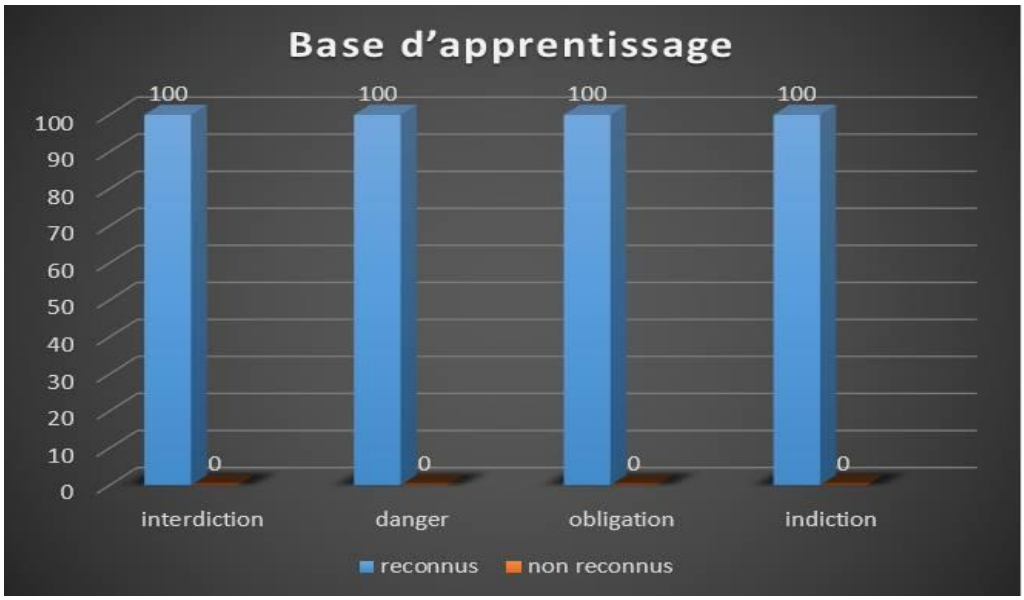


Figure 3.17 Base d'apprentissage.

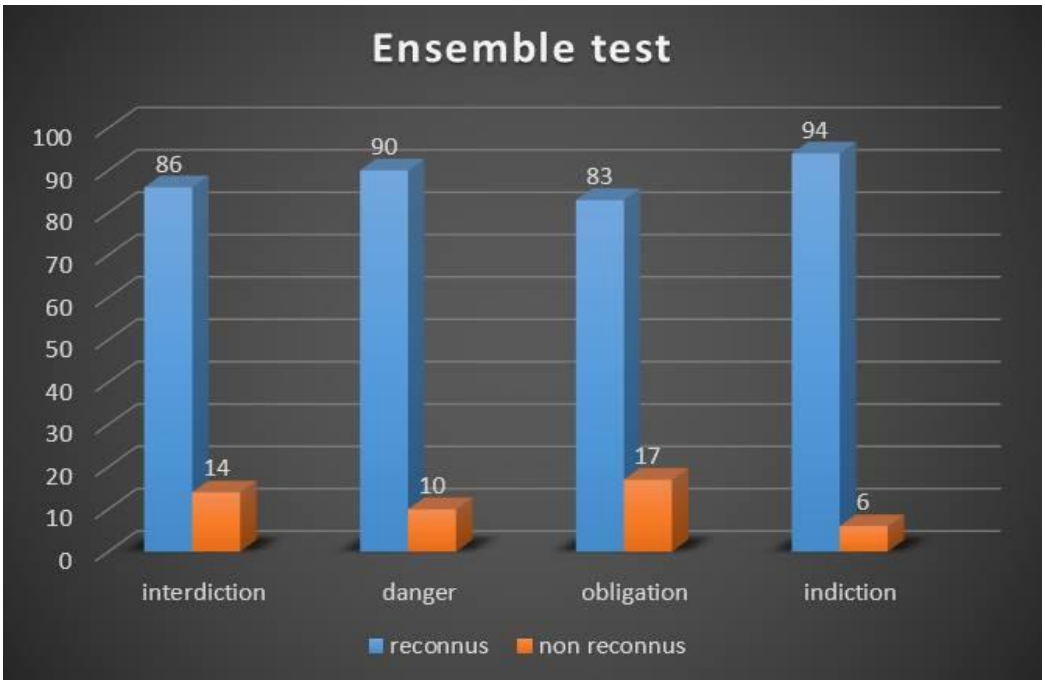


Figure 3.18 Ensemble test.

3.6.1 Panneaux non reconnu:

Les résultats obtenus sont satisfaisant et acceptable et depend de la qualiter des panneaux de signalisation de notre pays.

3.7 Conclusion

Le passage de la théorie à l'application afin d'obtenir des résultats satisfaisants n'est pas toujours évident. Ce passage est très délicat et plus important. Après plusieurs tests nous avons réussi à trouver l'enterval des couleurs rouge et bleu, qui convient mieux à notre système de reconnaissance proposé. Les résultats expérimentaux montrent bien que le descripteur proposé (DFT+Freeman) et la méthode de classification choisie (SVM) donnent d'excellents résultats. Cela montre bien que cette chaine de reconnaissance et prometteuse et peut faire l'objet d'une application industrielle.

Introduction générale

La vision par ordinateur (aussi appelée vision artificielle ou vision numérique) est une branche de l'intelligence artificielle dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs images prises par un système d'acquisition (par exemple: caméras, etc.).

La vision industrielle fait plutôt référence à une combinaison de techniques d'analyse d'image automatique, de mise en correspondance, de traitement de l'image acquise par caméra et de technologies d'acquisition d'image

Le traitement d'image est un ensemble des traitements automatisés qui permet à partir d'images numérisées, de produire d'autres images numériques.

La reconnaissance des objets est une branche de la vision artificielle et un des piliers de la vision industrielle. Elle consiste à identifier des formes pré-décrite dans une image numérique et s'attache à reconnaitre des formes géométriques dans une image.

La reconnaissance des formes cherche à identifier des motifs dans des données statistiques, on l'utilise souvent comme technique appliquée à la reconnaissance d'objet.

Dans ce travail, nous présentons une méthode de reconnaissance automatique des panneaux de signalisation routière. Cette méthode consiste à l'application de suivi de contour pour représenter les frontières du panneau. Ensuite, on fait une description de ces frontières par l'application de descripteur de Fourier avec codage de Freeman. Un vecteur de forme calculé sera utilisé par le module de classification qui consiste à la dernière étape de reconnaissance des panneaux.

L'objectif principal de ce projet de fin d'étude est d'arriver à réaliser un système de reconnaissance des panneaux de signalisation routière.

Ce système de reconnaissance passe par plusieurs étapes importantes :

1- Etape de prétraitement d'image nous a permis d'arriver à une image prête à faire la reconnaissance.
2- Etape utilisant l'algorithme des descripteurs de Fourier et le codage de Freeman et les SVM pour la classification.
3- Etape d'application du système.

ملخص:

يهدف مشروع نهاية الدراسة هذا إلى إنشاء نظام تحديد إشارات المرور في الوقت الحقيقي ، وهذا بالتركيز على تقنيات معالجة الصور, ويستند الاعتراف بشكل رئيسي على واصفات فورييه، آلة ترميز فريمان و ناقلات الدعم. لتحسين المعالجة الأولية بغية تقليل الشوائب في الصورة، نظام التحديديستند على تقنية التعرف على الألوان.

كلمات المفاتيح: الألوان، التعرف على إشارات المرور, ترميز فريمان.

Résumé :

Ce projet de fin d'étude a pour but la réalisation d'un système de reconnaissance des panneaux de signalisation routière en temps réel, en se basant sur les techniques du traitement d'images. La reconnaissance est basée principalement sur les descripteurs de Fourier, le codage de Freeman et le support vecteurs machine (SVM).

Mots clés : Reconnaissance des panneaux de signalisation routière, espace RVB, espace TSL, descripteur de Fourier, codage de Freeman, SVM

Abstract :

This final project study aims the realization of a road traffic signs recognition system in real time, based on the techniques of image processing. The recognition is based mainly on Fourier descriptors, the coding machine and Freeman vectors support (SVM).

Keywords: Handwritten numerals, Automatic recognition of Handwritten numerals, Freeman coding.

Conclusion générale

Pour réaliser un tel travail, nous avons été confrontés à des problèmes liés à l'aspect experimental que est purement du monde réel. L'environnement étant très complexe, les premiers problèmes concernaient les occultations, le changement de luminosité, la grande variation de texture et surtout la grande différence entre les panneaux routiers (couleur, position par rapport à la caméra, etc.).

Nous avons présenté en premier lieu des généralités sur le traitement d'images et détailler quelques outils de traitement que nous avons utilisé dans nos programmes , En deuxième lieu, nous avons présenté la méthode de reconnaissance avec toutes les étapes proposées en détaillant les algorithmes utilisés. En troisième lieu nous avons présenté l'application créée avec l'implémentation et les résultats des tests.

Au cours de ce projet et à travers le travail fait, nous avons pu acquérir beaucoup de connaissances. Nous avons enrichi nos connaissances dans le domaine du traitement d'image en apprenant de nouveau outils (OpenCV). Nous avons aussi appris à utiliser le langage de programmation C/C++.

Perspective……

[1] : R. AMMAM, S. BOUABBACHE « Classification des images texturées par la logique floue à partir d'estimation des paramètres texturaux » PFE 2000, université SAAD DAHLEB Blida, institut d'électronique.

[2] : M. BADECHE « Suivi de modèle et incrustation d'objets virtuels pour la réalité augmentée » Thèse de magistère 2006, université de Batna.

[3] : http://www.profil-couleur.com/ec/106-modele-rvb.php

[4] : https://fr.wikipedia.org/wiki/Teinte_saturation_lumi%C3%A8re

[5] : http://www.profil-couleur.com/ec/102-espace-couleur-tsl.php

[6] : A. D'HARDANCOURT « Fou du multimédia » Sybex 1995.

[7] : Encyclopédie Encarta Microsoft 1997.

[8] : M. HADALLAH « Codage des images fixes par méthode des hybride basée sur la QV et les approximation fractales » PFE USTHB

[9] : http://www.cndp.fr/crdp-dijon/-Les-images-numeriques-notions-de-.html

[10] : R. C. GONZALES, P. WINTZ « Digital image processing » Addition Wessley 1997.

[11] : M. KUNT « traitement numérique d'images ».

[12] : BOUMANDJI Samah, BOULFANI Yasmine « implémentation sur DSP TMS 320C5000 des filtres optimaux appliqué aux images et introduction de réseaux de neurones » PFE, Electronique, ENP, Alger 2004

[13] : BERBARA Ahmed, SAYAH DJEBBOUR Hassan « détection et localisation de caractères de codes à barres de type EAN 13 » PFE 2009, SAAD DAHLEB Blida.

[14] : https://fr.wikipedia.org/wiki/Panneau_de_signalisation_routi%C3%A8re#Histoire

[15] : A. Chottera and M. Shridhar, "Feature extraction of manufactured parts in the presence of spurious surface reflections, «Canadian Electron. J., vol. 7, no. 4, pp. 29-33,1982

[16] : BOUAZZA Bilal, MELBOUS Youcef « Reconnaissance de chiffres imprimés/manuscrits en temps réel via WebCam. Application à la reconnaissance de numéro de mobile » PFE 2011-2012, SAAD DAHLEB Blida.

[17] : HENRY MAITRE « description de contour et de formes » PDF.

[18] : https://fr.wikipedia.org/wiki/Machine_%C3%A0_vecteurs_de_support

Annexes

- ## Ouvrir le flux d'une vidéo / caméra

Avant de pouvoir lire un quelconque **flux vidéo** il faut déjà l'ouvrir. Définissez premièrement la variable qui va stocker ce flux

```
CvCapture *capture;
```

a) Ouvrir une vidéo

Ouvrir une vidéo est chose relativement aisée. Par contre il se peut que ve.us ne puissiez pas ouvrir tout type de vidéo. Ceci peut entre autre varier selon la version d'OpenCV mais aussi selon les différentes librairies installées sur votre système d'exploitation.

```
capture = cvCreateFileCapture("C:/patte/to/your/video/test.avi");
```

- ## Ouvrir le flux d'une Webcam / Caméra

Avec un flux vidéo récupéré en temps réel les capacités d'OpenCV prennent tout leur sens. Que ce soit une webcam ou une caméra, il est possible de récupérer ce qu'elles voient. De manière générale il est facile de lire le flux d'une webcam. Si vous souhaitez utiliser une caméra analogique (non numérique) il vous faudra probablement utiliser un convertisseur pour pouvoir la brancher en USB sur votre ordinateur. Notez par ailleurs que selon ce convertisseur (Dazzle, Easy Cap etc) vous aurez très probablement besoin du driver approprié selon votre système d'exploitation.

```
capture = cvCreateCameraCapture( CV_CAP_ANY );
```

*Le paramètre dans cvCreateCameraCapture() permet de définir quel flux vidéo récupérer. Généralement le CV_CAP_ANY permet de prendre le premier disponible (donc par exemple une webcam intégrée à votre ordinateur). Il se peut que cela ne marche pas tout le temps, donc si vous rencontrez un problème mettez 0 à la place ! Les valeurs diffèrent ensuite selon le port utilisé (usb, firewire). Pour plus d'information sur les valeurs disponibles, référez-vous au fichier highgui.h

*Toujours dans la même optique, vous pouvez y indiquer un autre chiffre pour ne pas récupérer le premier flux vidéo (par exemple /dev/video0 sur Linux) mais un autre. Ainsi si vous avez une Webcam intégrée et une caméra branchée en USB, et que vous souhaitez récupérer le flux de cette dernière, indiquez 1 à la méthode.

a) Vérifier que le flux est bien ouvert

Il est possible qu'ouvrir le flux vidéo ne se passe pas comme prévu. Pour pallier à ce problème, il est fortement conseillé de réaliser une petite vérification sur la variable "capture".

```
if (!capture) {
    printf("Ouverture du flux vidéo
    impossible !\n");
    return 1;
    }
```

Pour améliorer la vérification vous pourriez même retenter l'ouverture du flux X fois jusqu'à décider d'arrêter (pour éviter une boucle infinie). En effet dans certains cas, la webcam/caméra n'est pas détectée immédiatement. Si l'on ne laisse pas un petit intervalle de temps à l'ouverture, alors il faudrait lancer le programme plusieurs fois de suite jusqu'à ce qu'il ne fasse plus l'erreur.

- **Récupérer une image et l'afficher**

Le principe du traitement vidéo est de procéder à une analyse du flux image par image**.** A chaque traitement d'image terminé vous affichez l'image résultante dans une fenêtre.

a) Créer une fenêtre

Ce qui est bien avec OpenCV c'est que la création d'une fenêtre ne change pas en fonction du système d'exploitation. Par ailleurs vous pouvez **créer plusieurs fenêtres** pour afficher dedans plusieurs résultats (images ou vidéos). Une fenêtre est définie par un nom :

```
cvNamedWindow("GeckoGeek Window", CV WINDOW AUTOSIZE);
```

Le flag "CV_WINDOW AUTOSIZE" (qui est en fait un nombre : 0 ou 1) permet d'indiquer si la fenêtre doit s'adapter automatiquement ou non.

b Récupérer une image

Une image est stockée dans une classe spécifique à OpenCV : IplImage**.** Donc quelque part dans votre code avant de commencer vous devez la définir :

```
IplImage *image;
```

Il existe deux façons de récupérer une image du flux vidéo. La première (et la plus utilisée) est la suivante :

```
image = cvQueryFrame(capture);
```

Il faut indiquer en paramètre le flux vidéo à utiliser pour récupérer l'image. Cela devient vital dans le cas où vous utilisez plusieurs sources de vidéo !

L'autre possibilité pour lire l'image est d'utiliser deux méthodes séparées. La première permet de demander à OpenCV de récupérer une image du flux vidéo et de la stocker en mémoire. La méthode retourne 0 ou 1 en fonction du résultat.

```
cvGrabFrame(capture);
```

Ensuite il suffit de récupérer l'image dans la variable appropriée :

```
image = cvRetrieveFrame(capture);
```

Notez toutefois que c'est cette dernière méthode qui prend le plus de temps (5 à 10 fois plus que cvGrabFrame).

c Afficher une image

Une fois l'image stockée dans une variable il suffit de l'afficher dans une fenêtre préalablement créée.

```
cvShowImage("GeckoGeek Window", image);
```

Notez qu'en général, avant d'afficher l'image, on effectue des traitements sur cette dernière : zoom, contraste, reconnaissances variées (faciale par exemple), etc. Prenez en compte que la communauté d'OpenCV offre de nombreux codes en exemple qu'il est conseillé de parcourir pour vous faire une idée des possibilités.

• Boucler et temporiser

Pour récupérer les images petit à petit il vous faudra **créer une boucle** et faire une petite pause entre chaque image.

a) Faire une pause

Faire une pause dans OpenCV permet en même temps de récupérer une interaction clavier. Cela permet ainsi d'analyser la touche qui a subie une interaction et de procéder à certaines actions en conséquence. Notez que ce système peut malheureusement entrer en conflit selon votre programme. Par exemple si vous utilisez en plus une librairie graphique (comme QT) avec des champs texte, il vous faudra trouver une solution pour partager ces événements clavier.

```
key = cvWaitKey(10);
```

La commande cvWaitKey() attente X ms et renvoie l'interaction clavier (s'il y en a eu une).

Le temps d'attente varie en fonction de framerate que vous souhaitez imposer. Par contre vous ne pourrez pas aller au-delà de ce que propose votre caméra. Notez par ailleurs que mettre un temps d'attente trop court peut causer un freeze sur certaines versions de Linux. Mettez donc en général au moins 5 ms.

b) Faire une boucle

Et enfin il vous faut assembler tous les aspects dont on a parlé ci-dessus pour lire et afficher la vidéo ! Ici nous nous basons sur l'interaction clavier pour arrêter la boucle while. Il existe d'autres moyens d'arrêter la boucle (comme se baser sur le retour de cvGrabFràme).

```
// Boucle tant que l'utilisateur n'appuie pas sur la touche q (ou Q)
while(key !_ 'q' && key'Q')
{
     // On récupère une image
     image = cvQueryFrame(capture);
    // On affiche l'image dans une fenêtre
    cvShowImage("GeckoGeek Window", image);
    //On attend 10ms
    key = cvWaitKey(10);
}
```

- ## Libérer la mémoire utilisée

Etape très (très) importante, libérer toute la mémoire que vous avez utilisé en créant ces divers objets.

a) Libérer la fenêtre

Une seule fonction suffit à fermer toutes les fenêtres :

cvDestroyAllWindows0 ;

Toutefois dans le cas où vous ne souhaiteriez que fermer une seule et unique fenêtre, vous pouvez utilisez cette dernière :

```
cvDestroyWindow("GeckoGeek Window");
```

b) Libérer la variable d'image

Lorsque vous récupérez une image d'un flux vidéo vous n'avez pas besoin de la "libérer" dans la boucle while, ni même après. OpenCV gère ceci automatiquement et vous n'avez qu'à votre charge de dés allouer une image créée par vos soins. Si vous veniez à le faire, pensez donc à faire ceci :

```
cvReleaseImage(&uneAutreImage);
```

c) Libérer la capture vidéo

Et enfin, il vous faut libérer la capture vidéo (entre autre pour que la webcam/caméra ne soit plus utilisée par le programme).

```
cvReleaseCapture(&capture);
```