MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE
DEPARTEMENT AUTOMATIQUE

MEMOIRE DE MAGISTER

en ***Automatique***

Option: **Traitement d'Images et Reconnaissance de Formes**

Présenté par

GHALEB Souhila
Ingénieur U.M.M.T.O

Etude et application de la reconnaissance automatique de formes évolutives

Mémoire soutenu le 06/12/2015 devant le jury d'examen composé de :

BENFDILA Arezki, Professeur à l'U.M.M.T.O..... **Président**
DIAF Moussa, Professeur à l'U.M.M.T.O..... **Rapporteur**
RASSOUL Idir, M.C.A, à l'U.M.M.T.O..... **Examinateur**
LAZRI Mourad, M.C.A, à l'U.M.M.T.O..... **Examinateur**

Avant-propos

Ce mémoire a été effectué au Laboratoire Vision Artificielle et Automatique des Systèmes (**LVAAS**) du département Automatique, FGEI, UMMTO.

Mes vifs remerciements vont, tout d'abord, à Monsieur **DIAF Moussa**, professeur à l'UMMTO pour m'avoir proposé le thème de ce mémoire et m'avoir dirigée, aidée et conseillée tout au long de notre travail.

Nous ne manquerons pas de remercier Monsieur **BENFDILA Arezki**, Professeur à l'UMMTO pour m'avoir fait l'honneur d'accepter de présider le jury de ce mémoire.

Que Monsieur **RASSOUL Idir**, Maître de Conférences classe A à l'UMMTO trouve ici, nos sincères remerciements pour avoir accepté de faire partie du jury d'examen de notre mémoire.

Nous exprimons notre reconnaissance à Monsieur, **LAZRI Mourad**, Maître de Conférences classe A, à l'UMMTO pour avoir accepté de faire partie du jury d'examen de notre mémoire.

Enfin, je remercie vivement ma famille et mes amis pour leurs soutiens, mes collègues du département d'automatique pour leurs conseils, et mes collègues de travail pour leurs compréhension et leurs tolérance.

A mes très chers parents que je ne saurais remercier, et exprimer mon amour et ma reconnaissance.

A mes adorables enfants, Meriouma, Feriel, Yacine, Adel et Younes, vous êtes mon espoir. Que Dieu vous guide dans le droit chemin.

A mon Rachid pour son soutien et sa présence à mes cotés.

A mes frères et sœurs, avec tout mon amour et mon respect.
A tous ceux qui me sont chers.

à tous je dédie ce travail

Résumé

En reconnaissance de formes, particulièrement, dont le principe est d'associer une étiquette à une donnée, plusieurs méthodes ont été proposées permettant ainsi d'extraire automatiquement des informations de ces données pour former les classes, phase qui constitue l'apprentissage et d'affecter automatiquement la donnée à la classe appropriée, ce qui constitue la phase de reconnaissance. Cependant, ces méthodes ne sont généralement efficaces que si les données sont exhaustives. Si les objets sont évolutifs avec des caractéristiques de forme, de couleur ou de texture qui changent graduellement, la reconnaissance devient délicate. Pour être efficace dans ce cas, le système de reconnaissance de formes doit s'adapter à plusieurs situations en nécessitant une classification dynamique de ces données évolutives.

Dans ce travail, deux approches sont entretenues, la première consiste à concevoir une structure adaptative permettant la prise en compte du caractère évolutif des données en utilisant la méthode des k-Plus proches Voisins Flous Dynamique, qui sera appliquée sur différentes classes générées artificiellement. La deuxième approche concerne les évolutions périodiques, elle consiste à diviser la période de l'évolution sur des intervalles ce qui nous conduit à considérer statiques les classes correspondant à ces intervalles. Dans cette approche deux méthodes sont utilisées pour l'estimation du niveau de maturité de la tomate, il s'agit de la méthode de Syarir et *al* et la méthode des SVMs multiclasses.

Mots-clés : reconnaissance de formes, formes évolutives, classification dynamique, maturité de la tomate, SVM multiclasses.

Sommaire

Introduction générale

Le problème que cherche à résoudre la reconnaissance des formes (RdF) est d'associer une étiquette à une donnée qui peut se présenter sous forme d'une image ou d'un signal. Son objectif est donc de recueillir les données d'un capteur, c'est-à-dire une représentation, et en obtenir une ou des interprétations par l'exécution d'algorithmes[1]. Des méthodes générales ont été développées pour extraire automatiquement des informations des données sensibles afin de caractériser les classes de formes (apprentissage) et d'assigner automatiquement des données à ces classes (reconnaissance). Ces méthodes ont le même principe qui est réalisé en deux phases : l'apprentissage à partir des données connues et la classification des nouvelles données. En amont de ces deux phases, une étape de prétraitement est utilisée pour trouver l'ensemble minimal de paramètres informatifs (extraction des caractéristiques).

Pour les objets naturels, la RdF pose des problèmes souvent plus complexes que pour les objets manufacturés. En effet, bien souvent, les caractéristiques de ces objets (couleur, texture, forme) subissent une évolution naturelle : évolution saisonnière et vieillissement ; on dit que les classes sont dynamiques ou encore évolutives. Ce caractère dynamique rend la classification plus complexe et voir inefficace dans beaucoup de cas. Par ailleurs, il nécessite d'avoir un système de reconnaissance dynamique qui s'adapte à l'évolution des classes.

Les objets (au sens physique) sont dynamiques lorsqu'ils sont en mouvement. C'est le cas d'un objet réel (un solide non déformable) en mouvement. Cette dynamique n'implique pas nécessairement une évolution

(variation) de leurs caractéristiques (visuelles) dans le temps. Les classes constituées par de tels objets sont statiques (stationnaires). Lorsque les caractéristiques d'objets appartenant à une classe subissent des évolutions, cette classe est appelée classe évolutive ou classe dynamique. On a alors à traiter des classes dynamiques, d'où découle l'utilisation de l'expression *reconnaissance de classes dynamiques* [2]. Il est nécessaire de distinguer les notions d'Objets et de formes (ou individus). En effet, deux aspects dynamiques différents entrent en jeu pour caractériser : la dynamique de l'Objet dans l'espace physique et la dynamique de la forme associée à l'Objet, dans l'espace de représentation. Nous avons quatre cas de figures, le premier est quand les objets sont statiques et les classes associées à ces objets sont aussi statiques. Ce problème est largement traité dans la littérature, il fait l'objet d'une classification statique. Le deuxième cas, les objets sont dynamiques et leurs classes sont statiques, ce qui veut dire que les objets sont en mouvement, sans modifications de leurs caractéristiques [3]. Dans le troisième cas les objets sont statiques et leurs classes sont dynamiques ou évolutives. En effet, on distingue les deux types de classes évolutives, celles qui subissant des évolutions apériodiques et celles dont les évolutions sont périodiques. Et en dernier, on a le cas où les objets sont dynamiques et leurs classes aussi, autrement dit, les objets sont en mouvement et leurs caractéristiques évoluent avec le temps.

Dans ce mémoire, nous nous intéressons au troisième cas où nous considérons les objets statiques et leurs classes évolutives.
Cependant pour effectuer une reconnaissance de formes sur ce type de données deux approches ont été proposées dans la littérature [3]. L'une consiste en l'utilisation d'algorithmes de classification dynamiques qui mettent en œuvre les notions de déplacement, d'élimination, de fusion et de scission des classes [4]. L'autre approche concerne les évolutions périodiques et qui consiste à diviser la période de l'évolution sur des intervalles, ce qui nous conduit à considérer statiques, les classes correspondant à ces intervalles.
Ainsi, le travail présenté dans ce mémoire est scindé en trois chapitres.

Dans le premier chapitre nous présentons d'abord, des généralités sur la reconnaissance automatique de formes évolutives, puis le principe général d'une méthode de reconnaissance dynamique de formes. Nous passons en revue certaines de ces méthodes, ainsi qu'un état de l'art sur l'application de la classification dynamique dans différents domaines et disciplines.

Le second chapitre est consacré à la méthode des K plus proches voisins flous dans sa version dynamique et son application. Nous présentons d'abord le principe de cette méthode et ses principales étapes. Par la suite nous décrivons la mise en œuvre de la méthode sur un ensemble de classes générées artificiellement pour le suivi des différentes évolutions à savoir le déplacement, la fusion et la scission. Nous terminons ce chapitre par une discussion des résultats obtenus.

Le troisième chapitre est dédié à l'application de la deuxième approche de reconnaissance automatique de formes évolutives pour l'estimation du niveau de maturité de la tomate en utilisant le paramètre évoluant qu'est la couleur. Nous présentons les deux méthodes appliquées dans ce chapitre, à savoir, la méthode de Syarit et *al*, et la méthode basée sur les machines à vecteurs de supports (SVM), ainsi que, les tests et les résultats obtenus. Nous finalisons ce chapitre par une conclusion.

Ce mémoire est terminé par une conclusion générale et des perspectives ainsi que des références bibliographiques.

Chapitre 1

Généralités sur la reconnaissance de formes évolutives

1.1 Introduction

En classification automatique, lorsque les formes des objets sont évolutives, les classes constituées de ces objets changent également de manière dynamique. Les centres de gravité aussi bien que la matrice de dispersion de ces classes évoluent aussi. C'est ainsi que l'on parle de classification dynamique lorsque les formes des objets à classer évoluent en fonction du temps. Cependant, pour effectuer ce type de classification, deux approches ont été proposées dans la littérature [1][2]. L'une consiste en l'utilisation d'algorithmes de classification dynamiques qui mettent en œuvre les notions de déplacement, d'élimination, de fusion et de scission des classes [4]. L'autre approche consiste à diviser la période de l'évolution sur des intervalles ce qui nous conduit à considérer statiques les classes correspondant à ces intervalles. Parmi les applications réelles nécessitant ce type de classification, on peut citer le diagnostic industriel pour l'évolution des modes de fonctionnement [3][4], le diagnostic médical dans le cas de l'expansion de maladies tel le cancer [2], la télésurveillance de cibles en

mouvement [2]. Dans le cadre de ce mémoire, la classification dynamique sera appliquée à la reconnaissance automatique d'objets dont la forme évolue en fonction du temps.

1.2 Principe de la reconnaissance dynamique de formes

On rappelle que dans un processus classique de reconnaissance de formes, on distingue principalement quatre phases, à savoir, l'acquisition des données, le prétraitement de ces données, l'apprentissage et la reconnaissance proprement dite. Lors de l'opération d'acquisition, différents moyens peuvent être utilisés selon la nature des données. En fonction de la qualité de cette acquisition, des opérations de prétraitement sont souvent nécessaires. Ensuite, il s'agit d'extraire un ensemble de descripteurs qui seront normalisés et sélectionnées. A ces données, on applique la classification automatique pour l'obtention des classes d'apprentissage. La décision ou la reconnaissance proprement dite constitue la dernière phase de ce processus.

Dans le cas où les caractéristiques des objets sont évolutives, le processus de reconnaissance doit s'effectuer en détectant et en suivant ces changements. Le processus de classification doit suivre efficacement ces évolutions. La figure 1.1 montre un exemple d'évolution d'une classe.

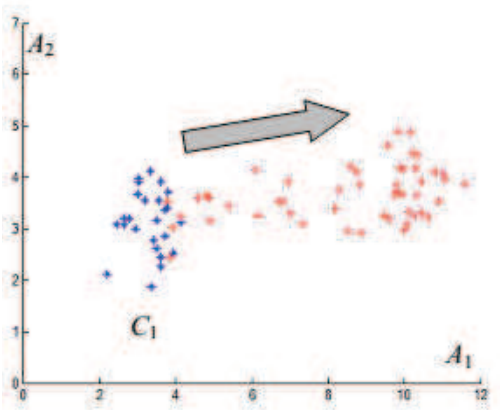


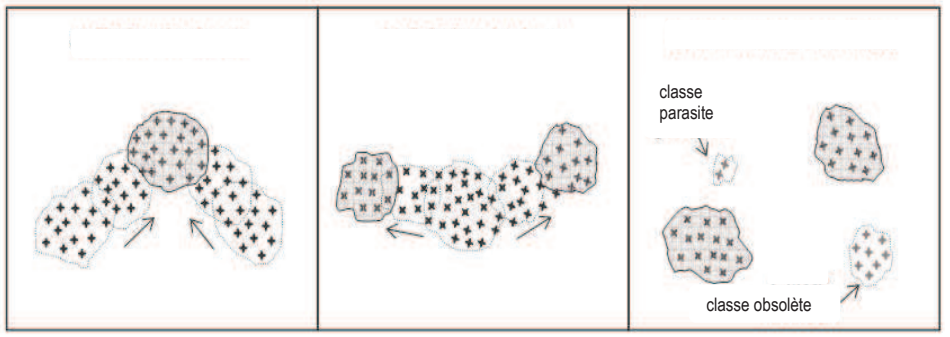
Fig.1.1 Evolution d'une classe dynamique.
Les formes qui évoluent sont représentées en rouge.

L'évolution d'une classe dynamique peut se traduire par un déplacement de certaines ou de toutes les formes des objets la constituant. Ainsi, elle peut avoir comme conséquence, sa fusion avec une autre classe ou sa scission en plusieurs nouvelles classes.

Dans le cas de la fusion, une ou plusieurs classes initialement disjointes peuvent se rejoindre et partager des données. Ces données dites ambiguës sont porteuses d'informations mutuelles. Afin de lever cette ambiguïté, la méthode de classification dynamique doit fusionner les classes concernées pour ne former qu'une seule classe (fig.1.2a).

Lorsqu'il s'agit de la scission de classes, le phénomène inverse de la fusion peut se produire. Ainsi, une classe se scinde en deux ou plusieurs classes (fig.1.2b).

Notons que l'évolution des classes peut entrainer leurs suppressions dans le cas où elles sont obsolètes autrement dit, de taille réduite ou sont des classes parasites constituées du bruit en présence (fig.1.2c). Pour la prise en considération de ces évoluions, différentes méthodes de reconnaissances dynamiques ont été préposées.



(a) Fusion de classes (b) Scission de class (c) Suppression de classes

Fig.1.2 Evolution des classes

1.3 Méthodes de reconnaissance dynamique de formes

Compte tenu de la non-stationnarité de données et de l'évolution de classes, le développement d'algorithmes de classification dynamique devient difficile [2]. Parmi les méthodes parues dans la littérature récente, on trouve différentes méthodes de reconnaissance dynamique telles que les méthodes

connexionnistes ou réseaux de neurones [2] [3] [4], les méthodes à noyau [2] et bien d'autres encore.

1.3.1 Les méthodes connexionnistes

Les réseaux de neurones sont des méthodes connexionnistes dont la structure est organisée en différentes couches reliées les unes aux autres. Ils sont basés sur une représentation mathématique (statistique) simplifiée des neurones biologiques ; on parle alors de l'utilisation de neurones formels. Les réseaux de neurones possèdent une couche d'entrées, une ou plusieurs couches cachées et une couche de sortie. La couche d'entrées correspond aux vecteurs formes de l'espace de représentation. La (ou les) couche cachée comprend les prototypes des classes. La couche de sortie définit le nombre de classes connues et permet de connaître le degré d'appartenance d'une observation à une classe. Pour chaque nouvelle observation présentée à l'entrée d'un réseau de neurones, les degrés d'appartenance de cette observation sont évalués par rapport aux différents prototypes des classes. A la suite de la phase d'apprentissage, le réseau dispose d'un ensemble d'observations classées, et d'observations écartées par ambiguïté. Plusieurs versions incrémentales des réseaux de neurones ont été développées, qui ont une architecture évolutive, autrement dit leur nombre de neurones et de connexions change en utilisant les nouvelles informations. Avant de détailler plusieurs réseaux de neurones dynamiques, nous allons commencer par détailler la structure. En effet, dans cette catégorie de RN, nous avons ceux qui sont appelés à prototypes. Ceux-ci permettent de représenter des prototypes comme des unités regroupant les données de caractéristiques typiques et leur architecture peut être de type constructif ou destructif. Dans le cas constructif ou incrémental, on commence par un petit nombre de neurones pour, ensuite, ajouter des neurones et des connexions. Dans le cas destructif, on commence par un grand réseau et on élague ensuite les éléments inutiles [5]. Parmi les méthodes proposées dans ce cadre, on peut citer le réseau de neurones Cluster Detection and Labeling (CDL) [3] [4] [5] et le réseau de neurones Auto-Adaptive & Dynamical Clustering (AUDyC) développé par C.Lurette [3] [4].

Le réseau de neurone Cluster Detection and Labeling (CDL)

Développé en 1998 par T.Eltoft [5], ce réseau dispose d'une architecture évolutive et de règles d'apprentissage non supervisé qui lui confèrent des capacités d'auto-adaptation lorsque l'environnement des données est non-stationnaire. Comme dans le cas classique, son architecture est constituée d'une couche d'entrée, d'une couche cachée et d'une couche de sortie. Les connexions entre la couche d'entrée et la couche cachée sont stockées dans une matrice de pondération W^1 pour mémoriser les paramètres des prototypes alors que celles entre la couche cachée et la couche de sortie le sont dans une autre matrice binaire W^2. Dans le cas d'une similarité entre l'observation X_i et le prototype P_j, la sortie Y_j de la couche cachée vaut 1. Dans le cas contraire, elle vaut 0. Il est de même dans le cas où la sortie Y_k de la couche de sortie vaut 1 si l'observation X_i est affectée à la classe C_k. Elle est nulle dans le cas inverse. Un réseau annexe intégrant une mesure de similarité effectue cette comparaison, cette similarité évalue le niveau de ressemblance des données par rapport aux prototypes. Le déroulement de l'algorithme du processus d'apprentissage du CDL tel que montré dans la figure 3.1, s'effectue en trois phases, à savoir, la classification, la fusion et l'évaluation. Toutefois, avant de débuter ce processus, le réseau est initialisé avec la première observation en créant le premier prototype(premier neurone de la couche cachée) ainsi que la première classe (premier neurone de la couche de sortie). Par ailleurs il faut également fixer les valeurs de seuils de similarité ξ_{min} et $\xi_{max.}$

Dans la phase de classification, on présente les observations X_i successivement et évalue leur similarité S_{ij} par rapport aux prototypes P_j existants. Différentes situations peuvent se présenter par comparaison de cette similarité aux seuils de similarité ξ_{min} et $\xi_{max.}$.

- Si l'observation est trop éloignée des prototypes existants ($S_{ij} < \xi_{min}$),

alors un nouveau prototype et une nouvelle classe sont créés et un neurone est crée sur chacune des couches cachée et de sortie.

- Si l'observation est proche d'un prototype mais pas suffisamment pour l'associer à celui-ci($\xi_{min} < S_{ij} < \xi_{max}$), alors un nouveau prototype est créé, donc un nouveau neurone sur la couche cachée.
- Si l'observation est très proche d'un prototype d'une classe donnée ($S_{ij} > \xi_{max} > \xi_{min}$), alors elle est simplement affectée à la classe considérée.
- Si l'observation est proche de plusieurs prototypes appartenant à différentes classes, alors elle est considérée comme ambiguë et sera écartée dans un ensemble.

Dans la phase de fusion, les classes partageant des données ambiguës sont fusionnées.
Dans la phase d'évaluation, les classes présentant un nombre très faible d'observations vis-à-vis d'un seuil de cardinalité sont éliminées.
Ces trois phases constituent le premier cycle de l'algorithme. Une mise à jour des seuils de similarités est effectuée et un nouveau cycle est relancé. Ce processus est répété jusqu'à ce qu'un pourcentage d'observations soit atteint.

```
Fixer les seuils de similarité minimale ξmin et ξmax
Initialiser le réseau avec la première observation

   Pour    chaque nouvelle observation    Faire
     Calculer  sa similarité S  avec l'ensemble des prototypes
         Si  Sij < ξmin   pour tous les prototypes  Alors
          Créer un nouveau prototype et une nouvelle classe
            Sinon
              Si ξmin< Sij < ξmax  pour des prototypes de la même classe Alors
                Créer un nouveau prototype pour cette classe
                  Sinon
                    Si ξmin< Sij pour des prototypes de classes différentes    Alors
                     Mettre cette observation dans une classe ambiguïté
     Fusionner les classes pour lesquelles au moins une ambiguïté a été soulevée
Adapter les seuils de similarité
Jusqu'à  atteindre un nombre d'itérations fixé
```

Fig.1.3 Algorithme CDL [3]

Le réseau de neurone Auto-Adaptive & Dynamical Clustering (AUDyC)

Cette méthode, développée par C.Lurette[3], et Amadou Boubacar [4], est basée sur un réseau de neurones structuré en trois couches qui s'est inspiré du réseau CDL. La structure d'AUDyC correspond à une organisation multiprototype permettant de caractériser les classes de formes complexes à l'aide d'un ensemble de sous-classes gaussiennes. La couche d'entrée d'AUDyC représente les observations de l'espace de représentation.

La couche cachée représente les prototypes des classes et la couche de sortie correspond aux classes connues. Ces deux couches sont évolutives au niveau de leurs connexions et de leur nombre de neurones. La couche cachée est entièrement connectée à la couche d'entrée à l'aide d'arcs pondérés. L'utilisation des poids en fonction de chacun des attributs de l'espace de représentation permet d'obtenir un prototype pour chaque neurone *j* de la couche cachée.

Dans la couche cachée, on appelle fonction d'activation d'un neurone *j* la fonction d'appartenance (degré d'appartenance) μ_j correspondant au prototype *j*. Le choix de cette fonction d'appartenance est libre. AUDyC intègre une fonction d'appartenance gaussienne utilisant la distance de Mahalanobis. La couche de sortie est reliée à la couche cachée afin d'associer les prototypes existants aux classes connues. Ces liaisons sont réalisées à l'aide de valeurs binaires déterminant ou non l'appartenance d'un prototype à une classe.

Initialement la méthode commence par apprendre une première observation pour créer le premier prototype et la première classe. Il est alors nécessaire de définir la valeur initiale de l'écart type de la classe par rapport à chaque attribut. Puis, les autres formes sont apprises et les valeurs des poids sont affinées. Dans la phase de classification, le degré d'appartenance d'une forme *X* aux classes et aux prototypes est comparé à deux seuils μ_{min} et μ_{max} . μ_{min} représente le seuil minimum d'appartenance d'une forme à une classe et μ_{max} est le seuil minimal d'appartenance à un prototype. Chaque nouvelle forme est classée selon ses valeurs d'appartenance. Soit :

- la forme est proche d'un prototype existant*($\mu(X) \geq \mu_{max}$)*, une mise à jour du prototype est réalisée.

- la forme est proche d'une classe connue mais sa valeur d'appartenance est faible ($\mu_{min} \leq \mu(X) < \mu_{max}$). Dans ce cas, un nouveau prototype est créé .
- la forme est loin des classes connues ($\mu(X) < \mu_{min}$), ce qui engendre la création d'une classe et de son premier prototype.

Si des prototypes d'une même classe ont des fonctions d'appartenance qui se chevauchent et si un certain nombre de points apparaissent dans cette zone d'ambiguïté, les prototypes des classes sont fusionnés. Si une nouvelle forme ne peut être classée en fonction d'une valeur d'appartenance ambiguë entre plusieurs classes, elle est placée dans un ensemble d'ambiguïté qui sera analysé par la suite pour vérifier l'éventuelle fusion entre deux classes. Cette vérification se fait en fonction du nombre de formes ambiguës pouvant apparaître entre deux classes. AUDyC permet également l'élimination des classes et des prototypes peu représentatifs à l'aide d'un critère de cardinalité. Une étape de scission des classes a été ajoutée à la méthode dans [4] afin de diviser une classe en deux lorsque c'est nécessaire. Cette étape utilise la mesure de Fisher pour mesurer la proximité des prototypes. La scission a lieu si deux prototypes ont une mesure les séparant supérieure à un seuil donné.

Cette méthode permet l'adaptation des classes au cours du temps. Elle intègre plusieurs mécanismes afin de tenir compte de l'évolution des caractéristiques des classes évolutives. Cependant, de nombreux paramètres doivent être définis pour utiliser la méthode et plusieurs d'entre eux ont une importance majeure dans ses résultats. Ces paramètres sont entre autres les seuils d'appartenance minimum et maximum aux classes et aux prototypes, la valeur initiale de la matrice de covariance des prototypes, les seuils de fusion et de scission et les paramètres de robustesse (seuil d'ambiguïté, seuil de cardinalité).

1.3.2 Méthodes à noyaux

Parmi les méthodes à noyaux, les plus utilisées: les Séparateurs à Vaste Marge ou support ou Vector Machine (SVM), qui est développée en 1990

par Vapnick [6]. Cette méthode ayant une capacité à travailler avec des données de grandes dimensions, elle a été adoptée rapidement et de nombreuses autres versions ont été proposées dans la littérature [7][8]. Parmi ces versions, on peut citer les SVM linéaires dont les vecteurs de supports permettent de discriminer les classes en définissant une frontière de décision linéaire [9] [10]. On peut citer aussi les SVM non linéaires adaptés aux classes dont la séparation ne peut pas être linéaire [11]

Les SVM sont donc des méthodes supervisées permettant, initialement, de prendre une décision binaire, ce qui correspond à une séparation en deux classes au maximum. Cependant, des versions multi-classes ont été développées afin de considérer les cas plus complexes comprenant plus de deux classes. Dans ce cas, plusieurs SVM sont nécessaires. Il existe la stratégie «un contre tous» [12] [13,] où l'on construit autant de SVM qu'il y a de classes. Il s'agit alors de chercher à distinguer les données d'une classe de celles de toutes les autres classes. Une autre stratégie appelée « un contre un » [14], consiste à construire autant de SVM qu'il y a de paires de classes. Autrement dit pour N classes on aura N*(N-1)/2 SVM.

Pour classifier les formes, les méthodes SVM cherchent l'hyperplan, ou la frontière de décision, qui maximise la distance entre les points de chaque classe. Dans la phase d'apprentissage, la méthode SVM calcule une fonction f qui lie chaque point de l'ensemble d'apprentissage à une classe. Cette fonction f définit l'hyperplan séparateur. L'hyperplan est basé sur un certain nombre de points caractéristiques de la séparation des classes, les vecteurs de supports. Le nombre et le choix des vecteurs de supports sont estimés en minimisant le critère d'optimisation des paramètres. L'optimisation des paramètres passe par une formulation duale du problème, généralement plus simple à traiter. A partir de cette formulation, les coefficients d'optimisation (multiplicateurs de Lagrange) sont obtenus par minimisation d'une fonction objective quadratique. Cette phase d'optimisation est réalisée initialement dans la phase d'apprentissage puis elle peut être poursuivie en ligne. Dans le cas non linéaire, les SVM réalisent une transformation de l'espace de représentation initial en un espace de Hilbert comprenant plus de dimensions. Cette transformation est réalisée à l'aide de différentes

fonctions noyau à savoir le noyau linéaire, le noyau polynomial, le noyau gaussien, etc. Cette mise en œuvre permet aux SVM d'être plus résistants que certaines méthodes de classification aux problèmes de dimensionnalité et de bruit. Il peut être reproché à la méthode SVM de reposer uniquement sur des vecteurs de support (données existantes) pour chercher à séparer les classes. De même, la complexité des méthodes SVM est importante et le choix de leurs paramètres initiaux (dont le noyau) est difficile et change complètement les résultats.

Toutefois des méthodes à noyau permettant de suivre les évolutions des classes ont été développées. Il s'agit des SVM en version dynamique et la méthode Self-Adaptive Kernel Machine (SAKM) qui seront détaillées ci-dessous.

Version dynamique des séparateurs à Vaste Marge

La méthode SVM dans sa version dynamique, proposée par K.Boukharouba [15], Met à jour les caractéristiques des classes, ce qui Permet d'améliorer la définition de la séparation des classes en intégrant les nouvelles informations et en supprimant les plus anciennes. Les vecteurs de supports initiaux de cette méthode sont obtenus de la même manière que pour la méthode SVM classique. Cependant nous avons trois types de données, à savoir, les vecteurs de supports, les formes situées dans la classe à l'intérieur de la zone délimitée par les vecteurs de supports et les formes situées en dehors de cette zone. Quand une nouvelle donnée est disponible, elle correspond à l'un de ces trois types. Si une nouvelle forme est classée dans la zone délimitée par les vecteurs de supports, alors la forme est bien classée. Cependant, aucun changement dans les vecteurs de supports ou dans la frontière de décision n'est nécessaire. A l'inverse, si la nouvelle observation est un vecteur de support ou une forme située en dehors de la classe, hors de la zone délimitée par les vecteurs de support, elle est utilisée pour la mise à jour durant laquelle les anciennes données voient leurs poids diminués. Si le poids d'un vecteur de support atteint la valeur zéro, celui-ci n'est plus considéré comme un vecteur de support. Lors de cette mise à jour, il est nécessaire de supprimer des anciennes données. Le nombre de

données à supprimer est déterminé expérimentalement lors de l'initialisation de la méthode en fonction de la dynamique des données. Plus les données évoluent rapidement, plus le nombre de données à supprimer est important. Pour changer les caractéristiques d'une classe [15] l'auteur a proposé des mécanismes de fusion et d'élimination afin d'améliorer l'estimation des classes. La fusion est détectée quand un nombre de données ambigües est présent entre plusieurs classes. Dans ce cas :

- la classe possédant le plus de données parmi les classes à fusionner est entièrement conservée,
- les formes des autres classes sont supprimées à l'exception de leurs vecteurs de supports qui sont conservés,
- les vecteurs de supports conservés sont classés un par un comme des nouvelles formes de la classe possédant le plus de données. Ainsi, la classe la plus représentative de celles à fusionner est conservée, mais elle est adaptée en tenant compte des formes les plus caractéristiques des autres classes à fusionner, càd, leurs vecteurs de supports.

Parmi les contraintes propres à cette méthode, on peut citer la définition du nombre de données anciennes à supprimer, la définition des seuils pour les données ambigües et les données obsolètes et l'absence de mécanisme concernant la scission d'une classe.

Self-Adaptive Kernel Machine (SAKM)

La méthode SAKM [4] est basée sur la méthode SVM. Elle intègre une structure reposant sur une architecture neuronale évolutive où chaque neurone caché correspond à un vecteur de support. La méthode se décompose en trois phases :

- pour chaque nouvelle donnée, une mesure de similarité permet de choisir la phase d'apprentissage adaptée qui suit,
- la phase d'apprentissage peut consister en la création d'une nouvelle classe, en la mise à jour d'une classe connue (adaptation), ou en la fusion de plusieurs classes,

- puis la phase d'évaluation permet de vérifier la représentation des classes créées, de séparer ces dernières si nécessaire (scission) et d'éliminer les classes obsolètes.

Elle modélise les classes de la partition dynamique en s'appuyant sur les fonctions d'apprentissage à noyau représentées par des hyperplans dans l'espace de Hilbert. Le critère de similarité de SAKM exploite la propriété du noyau choisi. Il consiste à mesurer la distance entre chaque nouvelle donnée et le vecteur support le plus proche pour chaque classe dans l'espace d'Hilbert. En fonction de sa position par rapport à chaque classe, la forme peut être rejetée ou classée comme nouvelle forme de la classe ou comme nouveau vecteur de support. Une marge de tolérance est utilisée lors de la classification, elle permet de tenir compte des erreurs de classification réalisées par la méthode lors de la phase d'apprentissage. C'est durant cette phase que les formes et les classes sont progressivement apprises. Le premier vecteur de support de chaque classe est initialement représenté par le premier point de cette dernière, puis la procédure d'adaptation permet de modifier ces vecteurs de supports. En effet, un poids est défini pour chaque forme et durant la procédure d'adaptation de SAKM, ces poids sont mis à jour en utilisant la technique du gradient stochastique. Seul un nombre maximal de vecteurs de supports est contenu dans chaque classe. Par conséquence, lors de la procédure d'adaptation, si un nouveau vecteur de support est défini, le plus ancien (poids le plus faible) sera supprimé. La procédure de fusion de SAKM est basée sur les données ambigües se trouvant entre les classes. Cette procédure consiste à sélectionner les classes qui doivent fusionner puis à réinitialiser ces classes en utilisant l'ensemble des nouvelles données sélectionnées pour la fusion. La procédure de scission de SAKM consiste à étudier la proximité des groupes de formes sur l'ensemble de la partition des données. Cependant, cette procédure est jugée trop couteuse en temps par les auteurs et elle n'est donc pas utilisable pour les applications en ligne. La procédure d'élimination des classes obsolètes repose sur un critère de cardinalité. La méthode réalise ainsi le suivi des classes et la mise à jour des vecteurs de supports tout en réalisant certains mécanismes de mise à

jour de l'ensemble des données (élimination, fusion). Les limites de la méthode viennent de la difficulté à définir certains de ses nombreux paramètres (noyau, seuil de similarité, ratio d'apprentissage, etc.).

1.3.3 Autres méthodes

Parmi les autres méthodes de classification réalisant la mise à jour des classes comprenant notamment la fusion, la scission, etc., on peut citer la méthode des K Plus Proches Voisins Flous Dynamiques (KPPVFD) proposée par Laurent HARTERT [1]. C'est cette méthode qui fera l'objet de notre travail comme première approche dynamique. Parmi ces méthodes, on peut aussi citer l'algorithme FCM Adaptative [16], et la méthode AddC[1]. Leur principes sont expliqués dans les parties qui suivent.

Fuzzy C-Means (FCM) Adaptative

Cette méthode a été développée pour intégrer en ligne de nouvelles données et d'adapter leurs classes, son principe est le même que celui de la méthode statique(fig1.4) proposée par Bezdek [17], déjà bien expliquée dans la littérature. En effet, suite à la phase d'apprentissage (des classes et de leurs prototypes), chaque nouvelle forme peut être classée en calculant sa valeur d'appartenance par rapport à chaque classe. Le mécanisme proposé par Marsili-Libelli [16] permet de mettre à jour de façon incrémentale les classes uniquement lorsqu'il s'agit des informations représentatives qui sont classifiées. comme exemple, des données bruitées ne devraient pas modifier les caractéristiques d'une classe. Pour cela, dans [17], la notion d'entropie d'une classe est intégrée pour indiquer le degré d'homogénéité de celle-ci, elle est calculée, et en fonction de son évolution suite à la classification d'une donnée, il est choisi ou non de mettre à jour les caractéristiques de la classe. Dans le cas où l'évolution de l'entropie d'une classe ne dépasse pas une augmentation autorisée, la classe est adaptée. De la même façon, l'évolution négative d'une entropie signifie que le la formation des classes devient plus discriminative et, dans ce cas, la mise à jour de la classe est également réalisée. En conservant seulement les faibles évolutions d'entropie d'une classe, l'auteur considère seulement les déplacements progressifs des

classes, sans aucune procédure réalisant la scission ou la fusion de classes. Ainsi, étant une méthode non supervisée, la méthode FCM Adaptative permet de prendre en compte la création de nouvelles classes.

1. Fixer les paramètres:
    a. le nombre de K classes
    b. Le seuil Ɛ représentant l'erreur de convergence
    c. Le degré flou m, généralement pris à 2
2. Initialiser les centres des K classes de manière aléatoire.
3. Mettre à jour la matrice U des degrés d'appartenance
4. Mettre à jour le vecteur V des centres de classes
5. Répéter les étapes 3 et 4 jusqu'à satisfaction du critère d'arrêt:
    $\|V^{(t)} - V^{(t+1)}\| < Ɛ$, t étant la t^{ème} itérration.

Fig.1.4 Algorithme FCM

La méthode An-online dynamique data Clustering (AddC)

C'est une méthode non supervisée pour la classification des données non statiques. L'approche proposée [27] consiste à considérer le premier point à classer comme le premier prototype, et à classer les nouvelles données. Si une nouvelle donnée est trop éloignée des prototypes connus un nouveau prototype de classe est créé, ainsi le prototype le plus proche d'une nouvelle donnée est considéré comme son représentant. Chaque prototype est désigné comme le représentant d'un certain nombre de points. Ces points sont donc considérés comme appartenant à une même classe, et après la classification d'un point, le prototype qui le représente est mis à jour incrémentalement. Seul un nombre maximum de prototypes est autorisé. Quand deux prototypes sont trop près on les fusionne (position dans l'espace et nombre de formes qu'ils représentent). Cette approche ressemble fortement à la méthode des C-Moyennes consistant à minimiser l'inertie intraclasse et à maximiser l'inertie interclasse, seulement cette méthode permet une approche dynamique puisque chaque nouvelle donnée qui peut potentiellement représenter une nouvelle classe est directement traitée comme telle. Ainsi l'approche proposée permet de détecter les évolutions des

classes (déplacements progressifs et brutaux) et de faire leur mise à jour. Il est cependant important de bien définir les paramètres de la méthode (seuil de fusion, nombre de prototypes autorisés, etc.) pour obtenir les meilleurs résultats.

1.4 Etat de l'art

Les méthodes de classification dynamique sont utilisées pour répondre à de nombreux problèmes posés dans certains domaines comme l'exploration des données, la bioinformatique, la classification de documents, l'analyse d'images, la reconnaissance biométrique, la reconnaissance vocale, le diagnostic industriel, etc. [18] [19] [20] [21] [22] [23] [24]. Dans [25], un ensemble de règles générées à partir de l'ensemble d'apprentissage est utilisé pour réaliser le diagnostic des fautes des transmissions automatiques. Par la suite, ces règles ne sont plus valides suite à des changements survenant dans les paramètres de l'appareil. L'application proposée dans [26] traite de l'évaluation des risques clients pour une banque en fonction des changements dans leur mode de consommation. Dans [27], est étudié le problème du tri des fruits en fonction de leur qualité liée au climat ou à d'autres effets extérieurs. Dans [18], les auteurs cherchent à détecter et suivre l'évolution progressive des modes de fonctionnement d'un régulateur thermique en fonction de l'âge de ses composants et d'autres facteurs temporels de son environnement. Dans [28], on s'intéresse à l'estimation, dans le même lot, du ratio entre le nombre de cartes microélectroniques défectueuses et le nombre de celles qui sont de bonne qualité. L'estimation est effectuée sur les valeurs journalières qui évoluent selon différents facteurs humains et des équipements. Dans [29], Cohen traite les flux dynamiques de données au niveau des intersections des routes dans le but de réduire le temps d'attente des conducteurs. Dans [30], Rengaswamy propose une méthode appliquée au suivi et au diagnostic d'un système pour rendre le flux d'hydrocarbone plus fluide en cherchant à trouver les caractéristiques principales de la tendance du système et les conséquences potentielles tout en fournissant suffisamment d'informations aux opérateurs. Pour sa part, Stephanopoulos, dans [31], utilise la méthode définie par

Bakshi [32] pour réaliser le diagnostic d'un procédé industriel concernant la fermentation.

Dans cet état de l'art non exhaustif, nous remarquons bien que les domaines d'application sont diversifiées, ce qui nous montre que la reconnaissance dynamique de formes peut être appliquée pour des données de différents types.

1.5 Conclusion

Dans ce chapitre, nous avons présenté l'aspect dynamique de la reconnaissance de formes autrement dit, la reconnaissance de formes lorsque leurs caractéristiques évoluent dans le temps, ce qui se produit souvent dans les phénomènes naturels qui, souvent, subissent des évolutions périodiques ou apériodiques rendant ainsi la reconnaissance plus complexe particulièrement pour les évolutions apériodiques. Pour résoudre de tels problèmes, deux approches ont été proposées jusqu'à présent.

La première approche utilise une méthode de reconnaissance dynamique de formes après avoir rendu dynamique, le système de reconnaissance. Parmi les méthodes proposées, nous nous sommes intéressés à la méthode des k-plus proches voisins flous dynamiques en version supervisée. Nous utiliserons cette approche sur différentes données et particulièrement des classes gaussiennes générées artificiellement, ce qui fera l'objet du second chapitre. La deuxième approche concernant les évolutions périodiques consiste à rendre statique un problème dynamique. Dans ce cas, une classe évoluant périodiquement est observée sur au moins une période. Cette approche sera appliquée pour le suivi et l'estimation du niveau de maturité d'un fruit comme nous le verrons au troisième chapitre. En effet, la caractéristique couleur évolue durant la période de maturité, en allant d'une couleur verte à une couleur rouge signifiant la maturité totale, en passant par plusieurs couleurs intermédiaires correspondants aux différents stades de maturité. La classification se fera à travers des méthodes de classifications classiques connues.

Chapitre 2

Méthode des k-plus proches voisins flous dynamique (KPPVFD)

2.1 Introduction

La classification, en générale, est définie comme étant l'action de construire une collection d'objets similaires au sein d'un même groupe, dissimilaires quand ils appartiennent à des groupes différents. Il s'agit généralement d'identifier le type de données qui peuvent être qualitatives ou quantitatives, de choisir la méthode de classification automatique selon le nombre de données, le type de variables et les connaissances *a priori* sur ces données, puis d'analyser les résultats obtenus par leur interprétation, leur évaluation etc. Les méthodes de classification automatique visent à répartir des objets décrits par un certain nombre de descripteurs en classes homogènes. Une multitude de méthodes ont été présentées dans la littérature, on peut distinguer les méthodes supervisées (ou avec professeur) et les méthodes non supervisées (ou sans professeur). Dans le cas où les données sont classées *a priori*, il s'agit de classification supervisée.

L'utilisateur forme alors lui-même des classes pour chaque observation. Dans le cas contraire où l'on ne dispose pas d'information *a priori* sur les données, il s'agira d'une méthode non supervisée. Ces méthodes sont encore subdivisées en méthodes paramétrique et non paramétrique compte tenu des informations dont on dispose *a priori*. Cependant, on s'oriente de plus en plus vers des méthodes non paramétrique et non supervisées ou des méthodes de recherche de groupements ou "clustering" [16].
Dans la seconde partie de ce chapitre, nous présenterons, dans les détails, la méthode des k-plus proches voisins et particulièrement la version floue qui sera appliquée pour une classification dynamique d'objets évolutifs.

2.2 L'algorithme des k-plus proches voisins flous

Parmi les algorithmes des k-plus proches voisins (KPPV), on peut citer d'abord celui de J.M.Keller [33]. Trois étapes distinctes composent cette méthode non supervisée et non paramétrique. La première étape consiste à recenser pour chaque observation les k observations qui lui sont les plus proches. A la seconde étape, pour chaque observation X_i, on calcule l'estimateur. A la troisième étape de la procédure, on applique le filtre médian pour raffiner davantage les estimateurs pour que les points situés dans un proche voisinage aient des estimateurs semblables pour aboutir à la bonne classification. Cependant, l'algorithme des KPPV le plus utilisé consiste à calculer, pour chaque objet, les distances qui le séparent de tous les autres objets de chaque classe initiale pour trouver ses k plus proches voisins. La classe de l'objet en question est déterminée en fonction de la classe majoritaire parmi les k plus proches voisins. Cet algorithme présente l'avantage d'être simple et relativement efficace. Cependant il présente quelques inconvénients, notamment dans le choix de la valeur de k qui ne doit être ni trop grande, ni trop petite. Par ailleurs, cet algorithme ne spécifie pas le degré d'appartenance de l'élément à la classe à laquelle il a été attribué. Pour palier à ces inconvénients, d'autres versions de la méthode ont été développées dans la littérature [34] [35]. Celle qui a suscité notre intérêt est la version floue des k-plus proches voisins (KPPVF) développée par

J.M.Keller [33]. Le principe de cet algorithme consiste à allouer le degré d'appartenance d'un l'élément à une classe en fonction de la distance du vecteur de ces k plus proches voisins et de l'appartenance de ces voisins à la classe. Les détails de cet algorithme sont donnés à la figure 2.1. Dans cet algorithme, K est le nombre des k plus proches voisins, $\{X_1,X_2\ldots,X_n\}$, l'ensemble des formes étiquetées, μ_{ij}, le degré d'appartenance de la $j^{ème}$ forme à la $i^{ème}$ classe qui est entre 0 et 1, m, un paramètre qui détermine le degré de pondération de la distance dans le calcul de l'appartenance et $\|x-x_j\|$, la distance entre un vecteur $X_i(xi_1,xi_2\ldots,xi_n)$ et son $j^{ème}$ plus proche voisin $X_j(x_{j1},x_{j2}\ldots,x_{jn})$ La classe à laquelle est affecté le vecteur x est donnée par l'expression:.$argmax\left(\mu_i(X)\right)$.

Pour détecter l'aspect évolutif des classes, plusieurs mécanismes ont été ajoutés à la méthode des KPPVF étant donné que celle-ci tient compte des valeurs d'appartenance floues des points aux classes. Aussi, des mesures de similarité et de validité intégrées à cette méthode permettent de fusionner, de diviser ou de supprimer certaines classes. Ainsi, cette méthode des KPPVF prend le caractère dynamique.

2.3 Méthode des K-Plus Proches Voisins Flous Dynamique

Rappelons que la méthode des K-Plus Proches Voisins Flous en version Dynamique (KPPVFD) a comme objectif de former les classes en suivant leurs évolutions. Elle fonctionne en plusieurs phases. A partir de données initiales, la méthode KPPVF permet, tout d'abord de réaliser la classification de chacune des nouvelles formes. Par la suite, comme proposé dans [1], la méthode met à jour les paramètres qui permettent de suivre l'évolution des classes. Lorsqu'une évolution est détectée et confirmée par plusieurs indicateurs, les classes sont adaptées. En dernière phase, ces classes sont confirmées en ne conservant que celles qui sont représentatives. Ces différentes phases sont détaillées ci-après.

Soit L l'ensemble d'apprentissage
Soit X l'observation dont on souhaite déterminer la classe
Fixer la valeur de K
Initialiser i=1
Pour chaque élément Xi de L calculer la distance $\|X\text{-}X_j\|$,
Si (i<= k) Alors
Ajouter Xi à l'ensemble des k-ppv
Sinon Si(Xi est plus proche à X qu'un de ses k-ppv) Alors
Remplacer ce voisin par Xi
Pour chaque classe i des k-ppv calculer le degré d'appartenance de X par l'expression suivante:

$$\mu_i = \frac{\sum_{j=1}^{k} \mu_{ij} \left(\frac{1}{\|X-X_j\|^{2/(m-1)}}\right)}{\sum_{j=1}^{k} \left(\frac{1}{\|X-X_j\|^{2/(m-1)}}\right)}$$

Affecter X à la classe majoritaire donnée par l'expression suivante:

$$argmax\left(\mu_i(X)\right)$$

Fig.2.1 Algorithme des KPPVF

2.3.1 Phase d'apprentissage

Dans cette phase, l'utilisateur donne les classes initiales *Ci, i=1,2,…,K.* Les centres de gravité (CG_i^{jActu}) et l'écart type initial (σ_i^{jInit}) de chacune de ces classes *Ci* sont calculés.

2.3.2 Détection des évolutions des classes

Durant cette étape, la classe *Ci* qui reçoit un nouvel objet est la seule qui doit être mise à jour, autrement dit, les valeurs de l'écart type (σ_i^{jActu}) et du centre de gravité (CG_i^{jActu}) de *Ci* sont recalculés de manière incrémentale pour chaque attribut *j* en utilisant les expressions (2.1) et 2.2).

$$\sigma_i^{j_{Actu}} = \sqrt{\frac{N_i - 1}{N_i} \times \left(\sigma_i^{j_{Actu-1}}\right)^2 + \frac{\left(x^j - CG_i^{j_{Actu-1}}\right)^2}{N_i + 1}} \qquad (2.1)$$

$$CG_i^{j_{Actu}} = \frac{CG_i^{j_{Actu-1}} \times N_i}{N_i+1} + \frac{x^j}{N_i+1} \quad (2.2)$$

où *Ni* est le nombre de formes présentes dans *Ci*, $(\sigma i^{jActu-1})^2$ et $(CG_i^{jActu-1})$, respectivement la variance et le centre de gravité de la classe, par rapport à l'attribut *j* et tout cela avant l'ajout d'un nouvel élément.

Pour suivre les changements temporels du système en se basant sur les valeurs calculées de (σ_i^{jInit}), (CG_i^{jActu}) et (σ_i^{jActu}), deux indicateurs ind_{1j} et ind_{2j} sont utilisés.

L'indicateur ind_{1j} qui représente l'évolution de la compacité de la classe est calculé à partir de l'écart entre (σ_i^{jActu})et (σ_i^{jInit})pour chaque attribut *j* en utilisant l'expression (2.3). Cet indicateur est donné en pourcentage.

$$ind_{1j} = \frac{\sigma_i^{j_{Actu}} \times 100}{\sigma_i^{j_{Init}}} - 100 \quad (2.3)$$

Si, au moins, l'un des attributs *j* obtient une valeur de ind_{1j} supérieure à un seuil *th*1 donné, cela indique que les caractéristiques la classe C_i ont changé. Ce seuil *th*1 est usuellement fixé à une petite valeur de l'ordre de 5.

Contrairement, quand la classe évolue de façon abrupte, une valeur plus importante de ce seuil est nécessaire.

Quant à l'indicateur ind_{2j}, il représente l'écart d'un point par rapport à la dispersion moyenne de la classe. C'est la distance entre le $j^{ème}$ attribut et le centre de gravité actuel (CG_i^{jActu}) en fonction de l'écart-type actuel (σ_i^{jActu}). Cet indicateur est calculé par l'expression (2.4) suivante. Comme le précédent, cet indicateur est donné en pourcentage.

$$ind_{2j} = \frac{\left|x^j - CG_i^{j_{Actu}}\right| \times 100}{\sigma_i^{j_{Actu}}} - 100 \quad (2.4)$$

Si, au moins, l'un des attributs *j* obtient une valeur de ind_{2j} supérieure au seuil th_1 $(\max(ind_{2j}) \geq th_1)$, ceci indique que la valeur d'appartenance de ce point à la classe *Ci* est faible. Cependant, un seul point loin de la classe ne signifie pas un changement de caractéristiques. En effet, ce point peut être du bruit. De ce fait, il est nécessaire de définir un seuil *NbMin* représentant

le nombre de fois successives que ind_{2j} doit dépasser th_1 pour confirmer une évolution. Si *NbMin* est fixé à une valeur importante, l'évolution de la classe peut être détectée après un retard important. Ce nombre *NbMin* doit être défini en fonction d'un compromis entre le bruit présent dans les formes et le retard maximal de détection d'évolution souhaité. L'évolution de la classe est confirmée quand *NbMin* valeurs successives des deux indicateurs ind_{1j} et ind_{2j} sont supérieures à th_1.

2.3.3 Adaptation des classes

Dans sa phase d'adaptation, la méthode des KPPVFD permet d'ajuster les paramètres d'une classe ayant évoluée lorsque suffisamment de changements de caractéristiques ont été détectés et confirmés lors de la phase de détection. Ainsi, le nouveau centre de gravité (CG_i^{jActu}) et l'écart type initial de (σ_i^{jInit})de la classe sont calculés selon les équations (2.1) et (2.2) précédentes.

2.3.4 Validation des classes

Dans cette phase de validation des classes, la méthode des KPPVFD réalise les opérations suivantes :

- elle supprime des classes aberrantes et non représentatives. Ce qui permet également de tenir compte de la taille grandissante de l'ensemble des données,
- elle fusionne des classes qui ont une similarité forte,
- Elle fait la scission des classes qui ne peuvent plus être considérées comme une seule classe.

La suppression des classes non représentatives est effectuée si un nombre $n1$ insuffisant de formes est contenu dans la classe ($n_1 > k$) ou si aucune forme n'a été classée dans la classe depuis qu'un nombre n_2 suffisant de formes ont été classées dans d'autres classes.

La fusion de classes est effectuée lorsque ces classes évoluant vers une même direction se chevauchent suffisamment à un moment donné. Pour décider de cette fusion, une mesure de similarité δ_{iz} entre les deux clases *Ci* et *Cz* proposée dans [36] est calculée pour toutes les paires de classes après

la classification de chaque nouvelle forme. Cette mesure est donnée par l'expression (2.5). Si cette mesure dépasse un seuil th_{Fusion} entre deux classes alors ces dernières sont fusionnées.

$$\delta_{iz} = 1 - \frac{\sum_{x \in C_i et\, x \in C_z} |\pi_i(x) - \pi_z(x)|}{\sum_{x \in C_i} \pi_i(x) + \sum_{x \in C_z} \pi_z(x)} \qquad (2.5)$$

où $\pi_i(X)$ et $\pi_z(X)$ sont respectivement les valeurs d'appartenance de X à Ci et Cz. Plus δ_{iz} est proche de 1, plus les deux classes sont similaires. La valeur maximale représente deux classes complètement chevauchées donc il n'est pas nécessaire d'attendre que la valeur de similarité obtenue soit égale à 1 pour fusionner ces deux classes.

La figure (2.2) montre les valeurs de similarité obtenues en fonction de la proximité de deux classes gaussiennes.

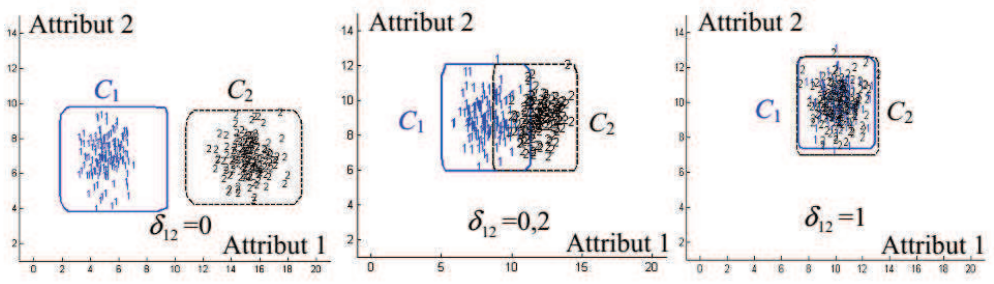


Fig.2.2 Evolution de la mesure de similarité entre deux classes fusionnant[4]

En ce qui concerne la détection de la scission, l'opération est plus complexe. Pour détecter la division d'une classe, on peut appliquer l'algorithme bien connu des Fuzzy-C-Means (FCM) donné dans le premier chapitre à la figure et la mesure de validité de Xie et Beni [37] définie par l'expression (2.6) suivante :

$$XB_{(c)} = \frac{J_m(U,V)/n}{\min_{i,j=1,c;j\neq i} \|v_i - v_j\|^2} \quad o\grave{u}\ J_m(U,V) = \sum_{k=1}^{n} \sum_{i=1}^{c} \pi_i\,(x_k)^m \|x_k - v_i\|^2 \qquad (2.6)$$

Dans cette expression, J_m représente la fonction objectif à minimiser, $\|.\|$, la norme euclidienne et $\pi_i(x_k)$ est la valeur d'appartenance du $k^{ème}$ point de la *ième* classe qui a pour centre v_i. Toutes les valeurs d'appartenance sont contenues dans *U* et les centres des classes sont contenus dans *V*. Le coefficient *m* est généralement fixé à 2. Il permet d'obtenir un partitionnement plus ou moins flou. Le nombre de classes trouvé par FCM doit être supérieur à 1 pour appliquer cette mesure de validité. Plus la mesure est faible, plus le nombre de classes est optimal. Une fois le nombre optimal de classes est estimé par FCM, la scission est réalisée si chacune de ces classes contient au moins *k* objets. Si le nombre de formes n'est pas suffisant au moment où le nombre de classes est estimé par FCM, alors il faut attendre que ce nombre d'objets soit classé pour de nouveau vérifier la scission de la classe.

L'algorithme général de la méthode KPPVFD peut être résumé par la figure (2.3) ci-dessous:

2.3.5 Paramètres de l'algorithme des KPPVFD

Comme pour plusieurs algorithmes faisant intervenir différents paramètres, un problème se pose souvent quant à l'affectation de valeurs à ces paramètres. En effet, celles-ci ont une influence sur les performances de la méthode, ce qui est de même en classification automatique. Des solutions ont été proposées dans plusieurs papiers comme dans [1] où l'auteur propose des valeurs par défaut pour l'algorithme des KPPVFD qui fait intervenir six paramètres *k*, th_1, *NbMin*, th_{Fusion}, n_1 *et* n_2. Les significations de ces paramètres sont les suivantes :

•*k* correspond au nombre de voisins de chaque point pris en compte pour réaliser la classification. Ce paramètre est commun à toutes les méthodes KPPV. Il est le plus important et il doit être défini en fonction de la taille de la base de données,

• th_1 est pris en compte par les deux indicateurs de détection d'évolution Ind_1 et Ind_2 présentés dans la section 2.3.2 précédente. Il est aussi l'un des

plus importants paramètres pour les KPPVFD. Les liens entre th_1 et Ind_1 et Ind_2 a été expliqué dans la même section 2.3.2 précédente. On rappelle cependant que lorsque la valeur 5 est attribuée à th_1,cela signifie un bon compromis permettant d'adapter les classes et leurs caractéristiques sans avoir besoin d'attendre la détection d'une évolution importante,

• *NbMin* est un paramètre qui permet de valider une évolution. Rappelons, en effet, que ce paramètre doit être défini en fonction d'un compromis entre le bruit présent dans les formes et le retard maximal de détection d'évolution souhaité. Il influence donc, les résultats au niveau du temps d'adaptation de la classe. Il est au moins égal au nombre des plus proches voisins k ($k \geq 1$) dans le but d'attendre suffisamment de formes représentatives pour bien estimer les caractéristiques d'une nouvelle classe. Il ne doit pas non plus prendre une trop grande valeur pour ne pas retarder la détection de l'évolution. Ainsi, il a été recommandé de fixer cette valeur entre k et k+5 [4],

• th_{Fusion}*:* est un paramètre d'optimisation utilisé pour fusionner les classes. En effet, même si aucune fusion n'apparaît, la simple apparition d'une classe signifie qu'une évolution du système a eu lieu. Il a été recommandé que lorsque th_{Fusion} prend une valeur proche de 0,2 cela permet de fusionner les classes qui ont commencé à avoir les mêmes caractéristiques [1],

•n_1 est un des paramètres utilisé dans la phase de validation des classes. Il devrait être défini supérieur à k ($n1>k$) puisqu'une classe, à sa création, contiendra au moins k formes. Une valeur par défaut de n_1 peut être k*2,

• n_2 est un des paramètres utilisé aussi dans la phase de validation des classes. Sa valeur ne doit pas être définie trop petite puisqu'après la création d'une classe, il peut être nécessaire d'attendre pour classifier plus de formes dans la classe créée. Au contraire, si aucune forme n'est classifiée dans la nouvelle classe après un nombre important de points, la classe n'est pas ou plus représentative. Il s'agit probablement d'une classe de bruit. La valeur de n_2 peut être définie par défaut à 20 [1]. Cela signifie qu'une forme sur 20 devrait être classifiée dans la nouvelle classe en vue de confirmer progressivement son utilité.

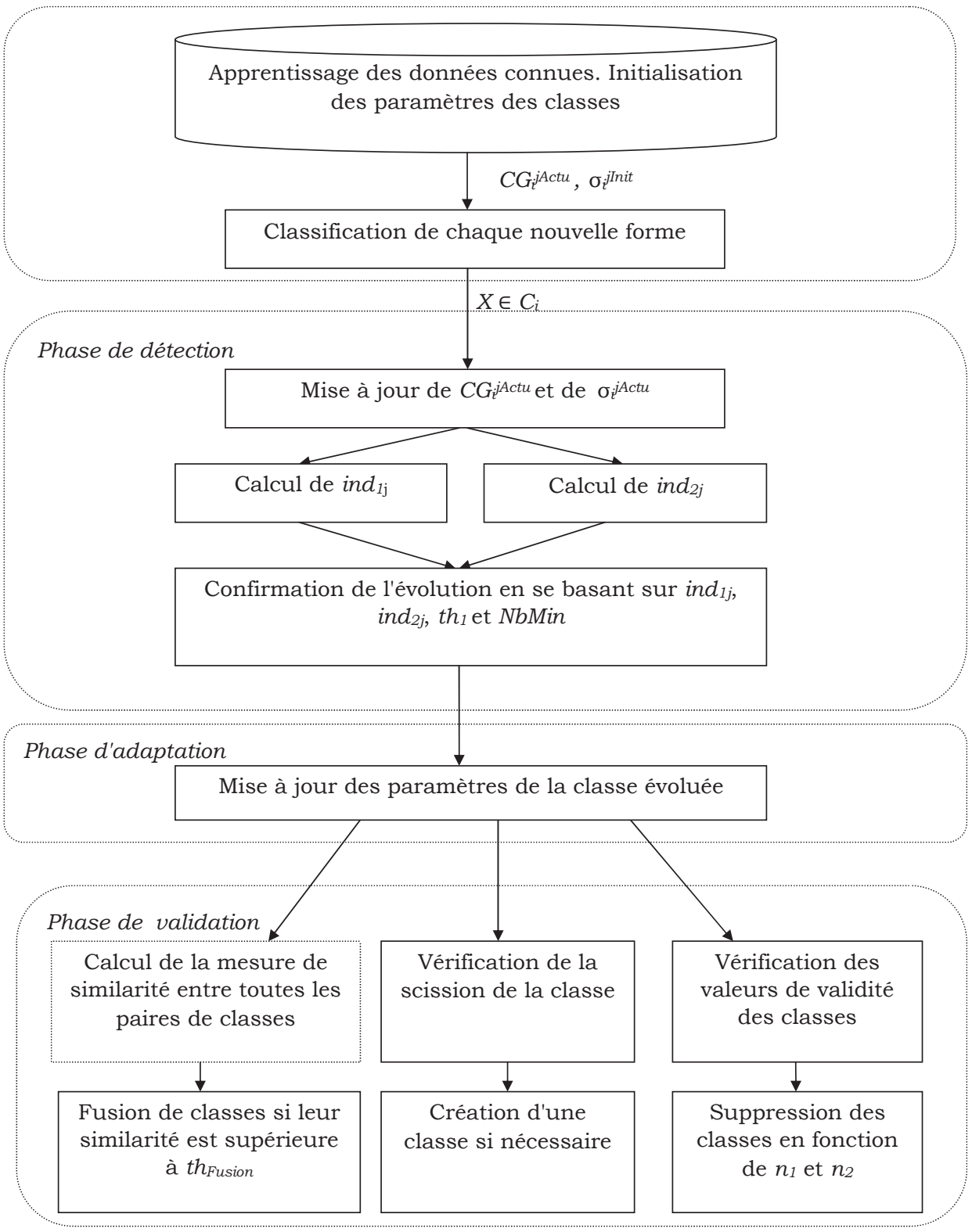


Fig.2.3 Description générale de la méthode KPPVFD

Pour les autres classes, elles ne seront pas supprimées puisqu'elles ont déjà confirmé leur utilité en ayant un nombre suffisant de formes même si elles ne reçoivent plus de formes après un certain temps.

2.4 Mise en œuvre de la méthode des KPPVFD

Dans cette partie, nous allons appliquer sur des données générées artificiellement, la méthode des KKPVFD dans les trois cas d'évolution à savoir le déplacement, la fusion et la scission. Notons que ces classes de données sont générées artificiellement car il est difficile d'obtenir une base de données réelles contenant des classes avec des caractéristiques évoluant avec le temps. Cependant ces classes artificielles bidimensionnelles gaussiennes traduisent parfaitement les différents aspects d'évolutions, ce qui nous a permis d'appliquer avantageusement la méthode en question sur les différents cas. Rappelons qu'une classe gaussienne est représentée par une distribution normale de deux variables indépendantes, la moyenne μ et l'écart type σ. Tous les résultats obtenus peuvent être étendus à des dimensions supérieures. Nous supposons qu'une unité de temps correspond à la classification d'un point [1] [4].

2.4.1 Déplacement d'une classe

La méthode KPPVFD a été évaluée sur le déplacement d'une classe. Ce déplacement a été généré comme suit :

- à t=0, 150 observations formant la classe initiale sont utilisées comme ensemble d'apprentissage. Les valeurs de moyenne et d'écart type de la classe sont pour l'attribut 1, μ_1=3 et σ=1, et pour l'attribut 2, μ_2=3 et σ_2=1.

- à t=1 jusque 50, 50 nouvelles formes apparaissent avec les mêmes écart type et moyenne que celles de la classe initiale. Dans ce cas, il n'y a pas d'évolution ou de déplacement.

- à t=51 jusque 200**,** un changement apparaît dans les valeurs de la moyenne de la classe par rapport à chaque attribut *j*, j=1,2. Ce changement est un déplacement progressif de la moyenne de la classe par rapport à chaque attribut selon les équations ci-dessous.

$$\left.\begin{aligned}\mu^{1'}(t) &= \mu^{1} + 2 + \frac{4\times(t-50)}{150}\\ \mu^{2'}(t) &= \mu^{2} + 2 + \frac{2\times(t-50)}{150}\end{aligned}\right] \quad 51 \le t \le 200$$

(2.7)

Après ce déplacement, nous obtenons une nouvelle classe ou classe finale.

- à *t*=201 jusque 300, 100 nouvelles observations apparaissent. Elles ont les mêmes écart type et moyenne que celles de la classe finale. Ces observations sont classées dans la classe finale sans qu'une nouvelle évolution suffisamment importante soit détectée. Ceci est fait avec les valeurs des paramètres fixées à savoir *k=5 ; th_1=5 ;NbMin=10 ; th_{Fusion}=0,2 ; n_1=10 et n_2=20.*La figure (2.4) montre un exemple de ce déplacement.

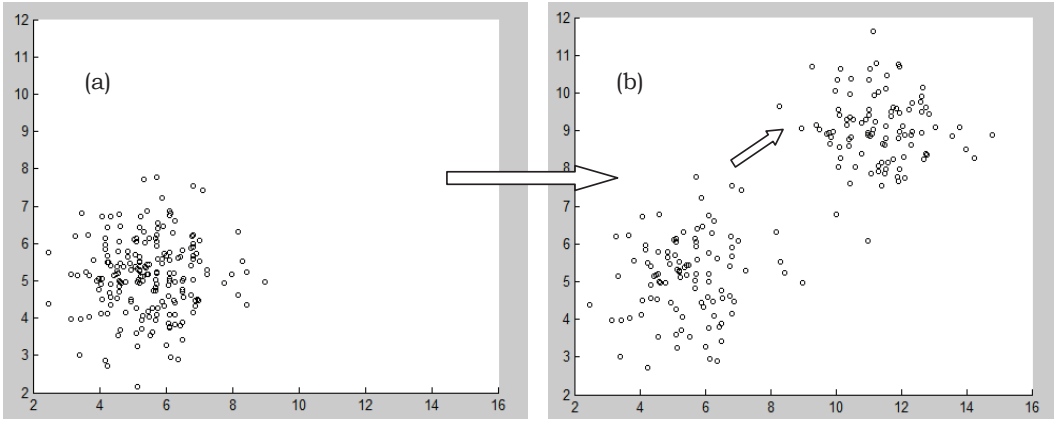


Fig.2.4 Déplacement d'une classe. (a), classe initiale. (b), déplacement de certains objets de la classe.

2.4.2 Fusion de classes

Dans cet exemple, nous prenons deux classes se déplaçant vers une zone commune pour fusionner. La fusion de deux classes peut représenter le cas d'une évolution temporelle d'un système vers un nouveau système. La base de données des deux classes qui fusionnent a été créée de la manière suivante :

- à *t*=0, 125 observations sont utilisées pour apprendre les fonctions d'appartenance de la classe C_1 (μ_1=5, μ_2=7) et 125 autres sont utilisées pour la classe C_2 (μ_1=15, μ_2=7). L'écart type de ces classes est égal à 1 pour chaque attribut.

- à *t*=1 jusque 150, les nouvelles formes appartenant à chaque classe se dirigent vers une zone commune aux deux classes (fig.2.5).

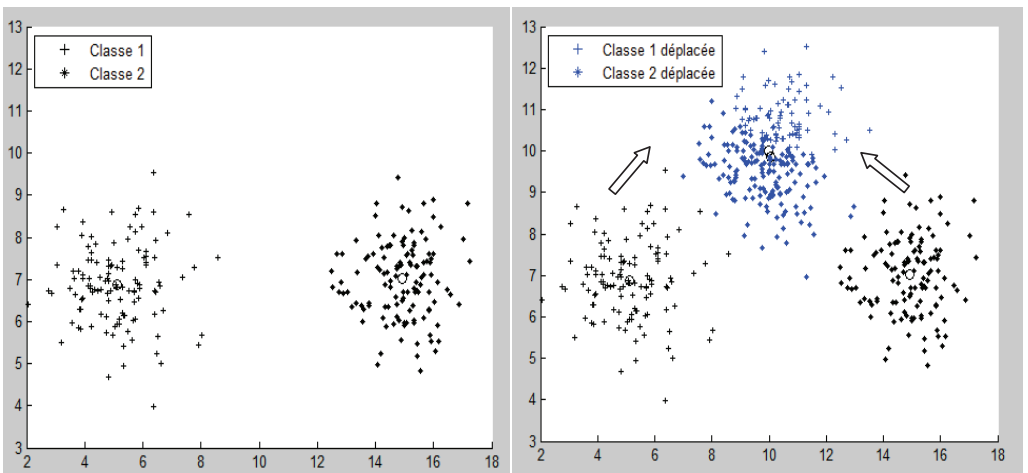


Fig.2.5 Déplacement de deux classes vers une même zone.

Ces déplacements sont générés pour les deux classes en fonction de chaque attribut *j, j=1,2*, en utilisant les expressions suivantes:

$$C1 \quad : \quad \left.\begin{matrix} \mu_1^{1'}(t) = \mu_1^1 + 2 + \frac{3\times t}{150} \\ \mu_1^{2'}(t) = \mu_1^2 + 2 + \frac{t}{150} \end{matrix}\right] \; 1 \le t \le 150, \tag{2. 8}$$

$$C2 \quad : \quad \left.\begin{matrix} \mu_2^{1'}(t) = \mu_2^1 - 2 - \frac{3\times t}{150} \\ \mu_2^{2'}(t) = \mu_2^2 + 2 + \frac{t}{150} \end{matrix}\right] \; 1 \le t \le 150,$$

le calcul de la valeur de similarité par l'expression (2.5) donne un résultat supérieur au seuil fixé, th_{Fusion}=0.2. Ceci veut dire que les deux classes se chevauchent suffisamment pour devoir fusionner.

La figure (2.6) montre le résultat de la fusion des classes et de la classification de l'ensemble des observations.

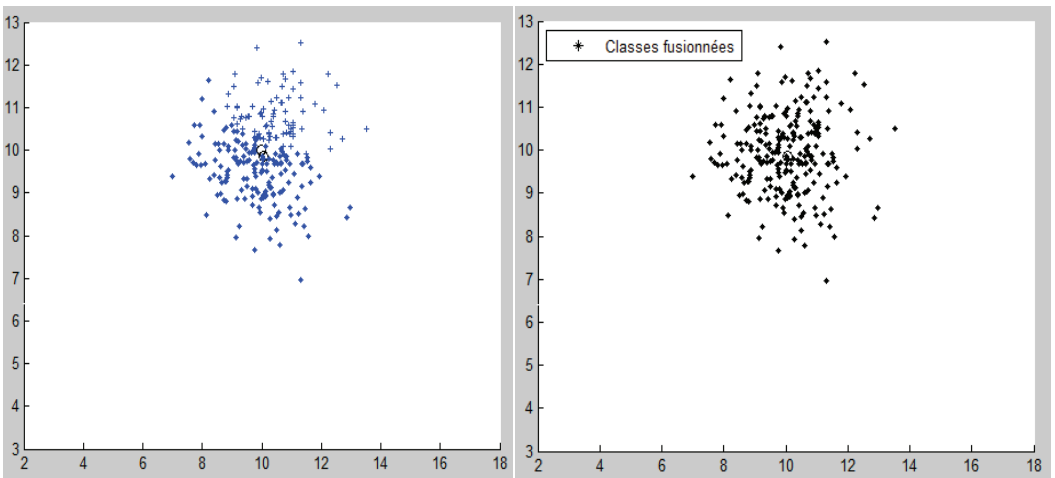


Fig.2.6 Résultat de classification obtenu par KPPVFD après fusion et classification de l'ensemble des formes . Ici *k=5 ; th_1=5 ;NbMin=10; th_{Fusion}=0,2 ; n_1=10 et n_2=20.*

2.4.3 Scission de classes

La méthode KPPVFD a été également testée dans le cas de la scission d'une classe dynamique sur une base de données crée artificiellement et évoluant en fonction du temps comme suit:

- à *t*=0**,** 100 observations sont crées comme classe initiales et qui serviront pour l'apprentissage.

- à *t*=1 jusque 150, les nouveaux points évoluent dans deux directions différentes comme le montre la figure (2.7).

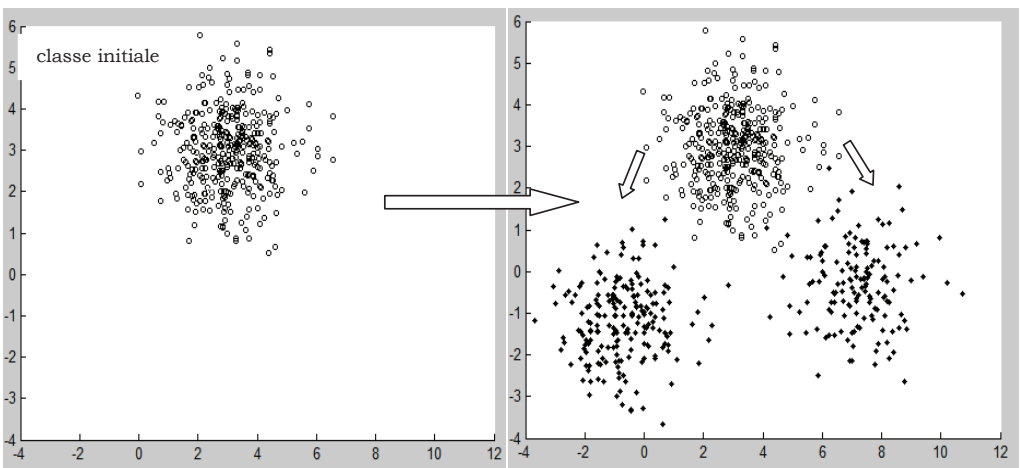


Fig.2.7 Evolution des points dans deux directions différentes

Cette évolution est générée pour C_1 en fonction de chaque attribut *j*, *j*=1,2, par les expressions (2.9).

$$\left.\begin{aligned}\mu_1^{1'} &= \mu_1^1 - 2 - \frac{4\times t}{150}\\ \mu_1^{2'} &= \mu_1^2 - 2 - \frac{5\times t}{150}\end{aligned}\right] \quad 1 \le t \le 150, \tag{2.9}$$

$$\left.\begin{aligned}\mu_1^{1'} &= \mu_1^1 + 2 + \frac{5\times t}{150}\\ \mu_1^{2'} &= \mu_1^2 - 2 - \frac{3\times t}{150}\end{aligned}\right] \quad 1 \le t \le 150,$$

A *t*=18, l'évolution de la classe initiale est évaluée, en appliquant la méthode FCM afin de déterminer le nombre de classes engendré par l'éloignement des points appartenant à la classe initiale. Ainsi, des mesures de validité calculées par cette méthode, et qui correspondent à plusieurs nombres de classes sont données dans le tableau (2.1).

La mesure de validité minimale obtenue est égale à 0.03, pour *c*=2, ce qui indique que le meilleur nombre de classes trouvé est deux. Ainsi, la classe initiale sera scindée en deux nouvelles classes.

Tableau 2.1 Mesures de validité pour la classe obtenue à t=18.

Nombre de classes	**Mesure de validité correspondante**
2	0.03
3	0.18
4	0.20
5	0.40

Une scission en deux classes a été mise en évidence par l'application de la méthode FCM qui a décelé les nouvelles classes. *La* figure *(2.8)* montre le résultat obtenu par FCM avec 2 classes.

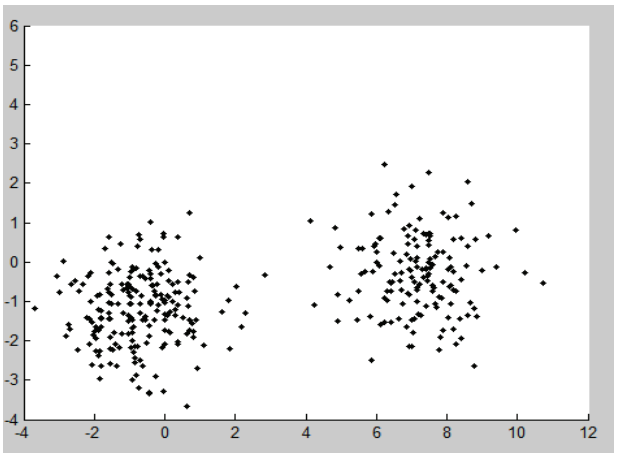


Fig.2.8 Résultat des classes obtenus par FCM à *t*=18.

La scission de la classe initiale en deux classes est ainsi détectée. Alors, le reste de la classification peut avoir lieu.

Les deux classes présentées sur la figure (2.9) montrent que l'algorithme KPPVFD a détecté les évolutions des classes et qu'avec son mécanisme de scission, deux classes ont été obtenues.

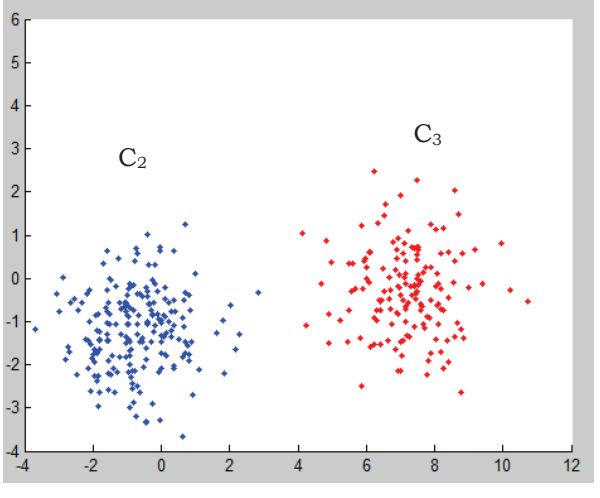


Fig.2.9 Résultat de classification obtenu pour le cas de scission avec *k=9, th_1=5, NbMin=15, th_{Fusion}=0,2, n_1=10 et n_2=20*

- *à t*=151 jusque 250, de nouvelles observations sont classées avec les mêmes écart type et moyenne que les dernières observations de l'évolution. Aucune nouvelle évolution n'a été détectée.

2.5 Discussion des résultats

Les résultats de classification obtenus par la méthode des KPPVFD notamment dans les différents cas d'évolution de classes gaussiennes générées, montrent que cette méthode est efficace pour la détection du déplacement de classes ainsi que leurs adaptations et validations. Néanmoins, dans le cas de la scission de classes et sur différents ensembles de tests, nous constatons que les résultats obtenus sont moins bons, mais acceptables. Bien que minimes, les erreurs d'estimation du nombre de classes obtenues après scission en faisant intervenir l'algorithme FCM, montrent bien la difficulté de la détection d'une scission et de sa validation. Cependant, d'une manière générale, pour les trois cas d'évolution, à savoir, le déplacement, la fusion et la scission nous avons atteint un taux de classification de 87%, ce qui peut être satisfaisant.

2.6 Conclusion

Dans ce chapitre, nous avons présenté la classification automatique de formes évolutives en appliquant une méthode dynamique de classification à savoir la méthode des k-plus proches voisins flous dynamique ou KPPVFD. Cette méthode tient compte de l'aspect évolutif des classes et permet de traiter les différentes évolutions que des classes dynamiques peuvent rencontrer comme le déplacement, la scission et la fusion. Par la suite plusieurs cas d'applications de la méthode ont été réalisés sur des classes générées artificiellement suivant la loi gaussienne. Les résultats obtenus montrent l'intérêt et l'apport d'une méthode de classification dynamique, en termes de résultats et de temps de classification. Néanmoins, le choix des valeurs des paramètres influence directement ces résultats. Dans le chapitre suivant nous allons étudier la deuxième approche de reconnaissance de formes évolutives à évolution périodique en divisant la période sur des intervalles concernant l'évolution. La réalisation pratique sera appliquée au suivi de la maturité de la tomate.

Chapitre 3

Application de la RdF pour l'estimation du niveau de maturité de la tomate

3.1 Introduction

La reconnaissance de formes évolutives touche plusieurs domaines d'application. En ce qui nous concerne, l'application à laquelle nous nous intéressons relève du domaine de l'industrie agro-alimentaire et particulièrement du contrôle de la qualité des tomates et de leur tri de façon automatique selon leur niveau de maturité. En effet, la tomate est un fruit climactérique qui continue sa maturité même après la cueillette. C'est aussi un fruit disponible tout au long de l'année largement consommé à l'échelle mondiale ainsi que dans notre pays. Ainsi, une automatisation du procédé du tri des tomates dans l'industrie est nécessaire. Cependant, jusqu'à présent, le tri, la sélection ainsi que la prédiction de la durée de conservation de ces produits s'effectuent manuellement dans beaucoup de pays, y compris l'Algérie. Ce type de décision se prend selon la coloration de la tomate qui évolue tout au long de la période de maturité en passant

progressivement par cinq couleurs correspondant à cinq stades de maturité codifiés par l'USDA (United States Departement of Agriculture), comme suit: vert blanchâtre qui tourne au jaune, tournant, orange, rouge claire, et rouge foncé [38]. Cependant, l'exploration visuelle peut être sujette à des erreurs liées à des troubles de la vue et à la fatigue du personnel manipulateur. Pour éviter toutes ces erreurs, l'automatisation de la procédure de reconnaissance est nécessaire. Pour ce faire, nous proposons d'utiliser la couleur qui évolue et détermine donc le stade de maturité de la tomate. C'est ce seul descripteur qui sera utilisé dans notre procédure de classification.

Dans la suite de ce chapitre, nous présentons, dans la seconde section, un aperçu concernant les travaux effectués sur l'estimation automatique de la maturité de différents fruits et légumes. Dans la troisième section, il s'agit de la description de l'objet de notre travail qui est la tomate. Les méthodes de reconnaissance développées que nous avons appliquées ainsi que les résultats obtenus sont proposés dans la troisième et dernière section. Ce chapitre est terminé par une conclusion.

3.2 Travaux effectués sur la maturité de fruits

Beaucoup de travaux de recherches ont été effectués pour l'évaluation et l'estimation du niveau de maturité de différents fruits et légumes. Une approche basée sur l'analyse d'images spectrales de tomates a été proposé par Polder Van Der Heijeden et Young [39] en comparant les images saisies et exprimées dans l'espace couleur RGB, avec les images hyper-spectrales. Une méthode d'analyse discriminante linéaire a été appliquée pour la classification des tomates selon leurs niveaux de maturité. et les résultats obtenus par les auteurs montrent que les images spectrales sont meilleures que les images RGB pour la classification

Dans une autre application concernant la maturité d'avocats, Guerrero et Benavides [40] ont proposé un système de classification automatique avec l'analyse discriminante linéaire de Fisher sur les paramètres RGB et en appliquant l'algorithme des K-means pour la classification. Les avocats sont alors classés en trois catégories: verts, mûrs et très mûrs avec un succès de 87,85%. Concernant la maturité de la banane, une autre application

utilisant un réseau de neurones artificiel a été développé par P. Hema et *al.* [41]. La méthode proposée se base sur les composantes RGB des images acquises quotidiennement sur quatre catégories de bananes de tailles différentes et à des stades de maturité variés, jusqu'à pourrissement du fruit, et ceci dans quatre différentes positions. Le réseau de neurone utilisé est achevé avec un taux de classification de 96%.

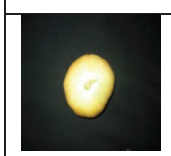
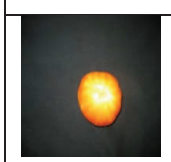
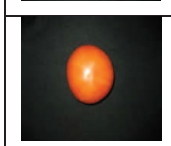
De leur côté, Dadwal et Banga [42] se sont intéressés à la classification des pommes en trois catégories, à savoir, immature, mature et trop-mature en utilisant la logique floue qui permet de calculer le degré d'appartenance du fruit à ces trois catégories. Par ailleurs, la classification du citron vert a été traitée par Damiri et Slamet [43] en utilisant aussi un réseau de neurones artificiels et en se basant sur des descripteurs de forme, de taille, de couleur et de texture. Un taux de classification de 100% a été obtenu.

Notons que plusieurs autres applications ont été proposées pour l'estimation du niveau de maturité de différents autres fruits et légumes [44] [45].

3.3 Estimation de la maturité de la tomate

En ce qui concerne la tomate, les paramètres couleur s'avèrent être très discriminants du taux de maturité. Ces paramètres sont calculés en appliquant deux méthodes à savoir la méthode de Syarir et *al* [45] et la méthode des SVMs et ce, en considérant les cinq stades repères de la maturité de la tomate codifiés par l'USDA (United States Departement of Agriculture) conformément au tableau 3.1. Cette couleur sera exprimée dans l'espace couleur L*a*b* en appliquant la méthode de Syarir et dans l'espace HSV en appliquant les SVMs. Ces deux espaces feront l'objet d'un bref rappel dans le paragraphe qui suit.

Tableau.3.1 les stades de maturité de la tomate

Image	Stade de maturité	Description
	Vert	La surface de la tomate et complètement verte
	Tournant	10% à 30% de la surface n'est pas verte
	Orange	30% à 60% de la surface n'est pas verte
	Rouge claire	60% à 90% de la surface n'est pas verte
	Rouge foncé	Plus de 90% de la surface n'est pas verte

En effet, rappelons, que pour caractériser la couleur, la commission internationale de l'éclairage (CIE), a développé des bases de représentation de la colorimétrie. La représentation standard est basée sur les trois composantes RGB. Sur la base de ces couleurs, plusieurs autres systèmes de représentation ont vu le jour tels que le système XYZ, le système L*u*v, le système L*a*b*, le système HSV et bien d'autres encore. Le passage d'un système à un autre s'effectue facilement à travers des expressions mathématiques comme le montre les expressions (3.1), (3.2) et (3.4). Dans le modèle L*a*b*, les trois composantes sont les suivantes [47]:

- la composante **L*** est la clarté allant de 0 (noir absolu) à 100 (blanc parfait).
- La composante **a*** représente une gamme de 600 niveaux sur un axe allant du rouge (+299 valeur positive) au vert (-300 valeur négative) en passant par le gris.

- La composante **b*** représente une gamme de 600 niveaux sur un axe allant du jaune (+299 valeur positive) au bleu (-300 valeur négative) en passant par le gris.

➢ **Conversions de CIE XYZ vers CIE L*a*b* (CIELAB):**

$$\begin{aligned} L^* &= 116\, f(Y/Y_n)-16, \\ a^* &= 500[f(X/X_n)-f(Y/Y_n)], \\ b^* &= 200[f(Y/Y_n)-f(Z/Z_n)], \end{aligned} \qquad (3.1)$$

avec:

$$f(t)=\begin{cases} t^{1/3} & Si\ t > (\frac{6}{29})^3 \\ \frac{1}{3}\left(\frac{29}{6}\right)^2 t + \frac{4}{29} & Sinon \end{cases} \qquad (3.2)$$

Ici X_n, Y_n et Z_n sont les composantes du blanc de référence (blanc décrit dans l'espace XYZ, n pour neutre). Les valeurs des composantes X, Y et Z sont obtenues par la conversion des valeurs R,G et B du système RGB suivant la relation matricielle ci-dessous:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = A \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad (3.3)$$

où A est une matrice (3*3) dite matrice de passage qui a différentes valeurs selon le système colorimétrique utilisé(NTSC, SECAM, PAL...).

➢ **Conversions RGB vers HSV:**

$$H = \cos^{-1} \frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}$$

$$S = 1 - \frac{3(\min(R,G,B))}{R+G+B} \qquad (3.4)$$

$$V = (R+G+B)/3$$

De son coté, l'espace de représentation HSV (Hue, Saturation, Value), couramment utilisé, est basé sur trois grandeurs, la teinte H qui caractérise la couleur elle-même; la saturation S représentant le niveau de pureté de la couleur (0 pour le noir ou le blanc et maximum pour une couleur pure), et la valeur V qui correspond à la contenance relative de noir et de blanc. L'avantage d'un tel système est que la couleur est représentée par des parties visuellement distinctes, ce qui montre une relation étroite entre la chrominance et la manière dont les humains perçoivent la couleur. Ce qui fait de ce modèle l'un des modèles idéals pour développer des applications de vision.

3.4 Application

Les deux méthodes que nous avons appliquées pour l'évaluation du niveau de maturité de la tomate sont La méthode de Syarir et al et la méthode basée sur les SVMs multi classes.

3.4.1 Méthode de Syarir et *al.*

Dans cette méthode, les histogrammes des deux valeurs a* et b* de tomates ont été tracés tout au long de la période de maturation, et les résultats obtenus ont montré que la valeur de b* ne manifeste pas de grands changements, alors que la valeur de a* accroit avec le taux de maturité. Dans notre application, c'est aussi ce paramètre a* que nous prendrons en considération. Un tableau (3.2) fait la correspondance entre la valeur de a* et le taux de maturité. Cette table nous informe aussi sur la durée de conservation des tomates. Ainsi, on peut sélectionner celles qui seront destinées à être transportées vers de longues ou petites distances.

Tableau.3.2 Correspondance de la valeur de a* et le niveau de maturité de la tomate [45] [46]

Taux de maturité	Valeur de a*	Couleur de la tomate	Durée de conservation
10% à 20% de maturité	a* < -5.8	Vert à jaune	21 à 28 jours
30% à 40% de maturité	-5.8<= a* < 2.1	Tournant	15 à 20 jours
50% à 60% de maturité	2.1<= a* < 9.2	Orange	7 à 14 jours
70% à 80% de maturité	2.1<= a* < 21.5	Rouge claire	5 à 6 jours
Maturité totale	21.5<=a*	Rouge foncé	2 à 4 jours

3.4.2 Méthode basée sur les SVMs multi classes

Dans cette approche nous allons utiliser les SVMs multi classe à noyau linéaire, selon la stratégie un contre tous que nous avons définie dans le premier chapitre, ce qui nous emmène à construire cinq machines pour la classification des tomates, étant donnée que nous avons cinq classes correspondant aux cinq stades de maturité. L'architecture générale de l'approche est décrite à la figure 3.1. Après la constitution de notre base d'apprentissage et de tests, trois étapes se succèdent, à savoir le prétraitement, l'extraction des attributs et enfin l'apprentissage et la classification. Les détails relatifs à ces étapes sont décrits plus bas. Cependant dans cette méthode des SVMs, à partir de la couleur des tomates exprimée dans le système HSV, et pour chacune des trois composantes H, S et V, nous allons établir les trois histogrammes et calculer les trois premiers moments à savoir: la moyenne, l'écart-type et le skewness. Au total, nous aurons trois histogrammes et neuf moments qui seront les composantes du vecteur caractéristique utilisé pour la classification des tomates.

3.4.3 Acquisition d'images

Deux cent cinquante images de tomates ont été acquises à l'aide d'une caméra CCD ayant une résolution de 10 méga pixels et placée approximativement à 100 mm par dessus la tomate avec des conditions d'éclairages similaires. On notera que les tomates sont placées de telle sorte que l'image saisie soit celle de la base tournée vers le haut et posée sur un fond noir pour atténuer l'ombre noir. La figure 3.2 nous montre quelques exemples d'images de tomates à différents stades de maturité.

Ces images sont réalisées sur 50 tomates à 3 ou 5 jours d'intervalle. Ainsi, une base de données contenant 250 images a été établie pour être utilisée pour l'apprentissage et pour les tests avec 50 images pour chacun des cinq niveaux de maturité.

Après l'acquisition de ces images, suit l'opération de prétraitement.

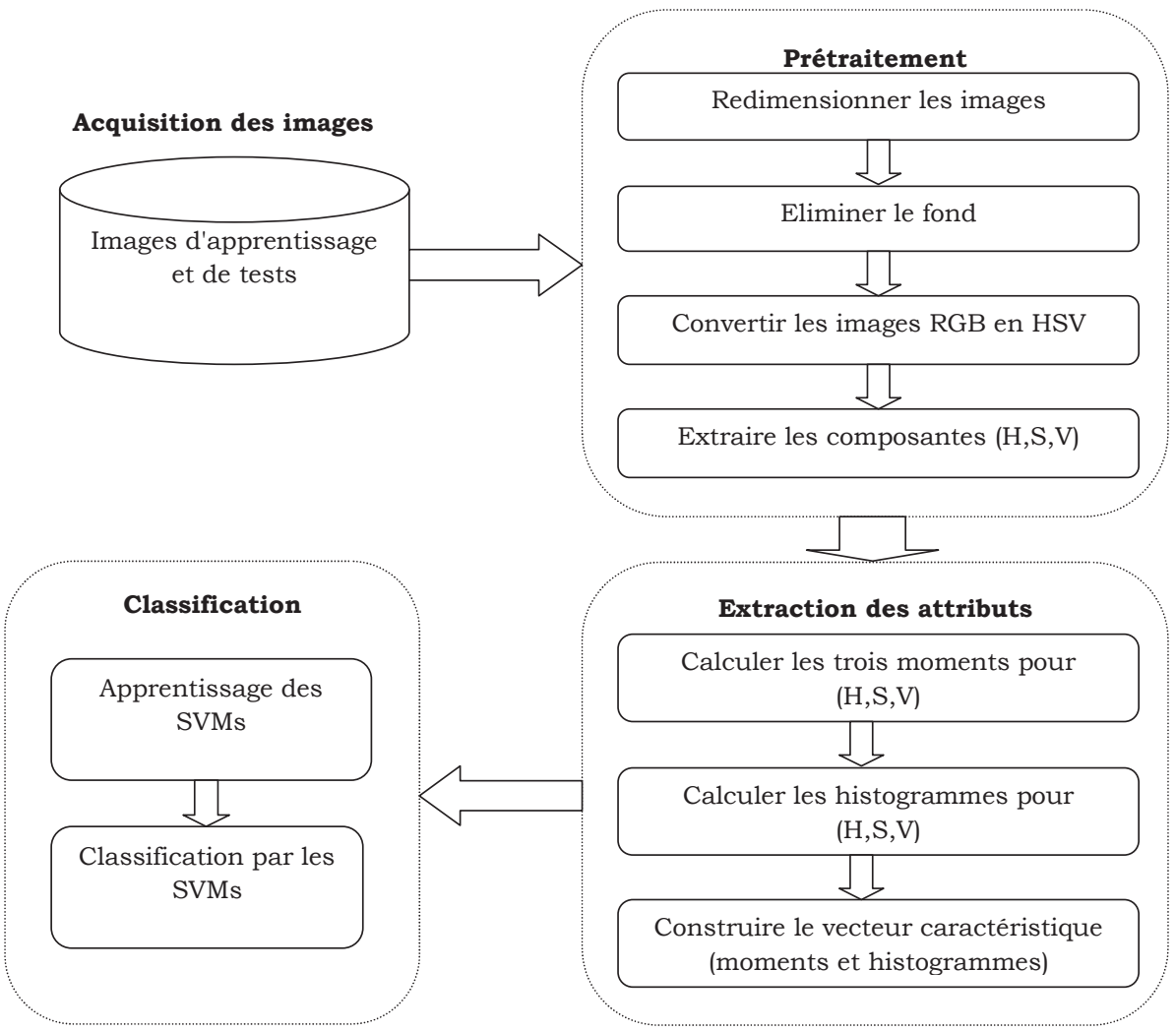
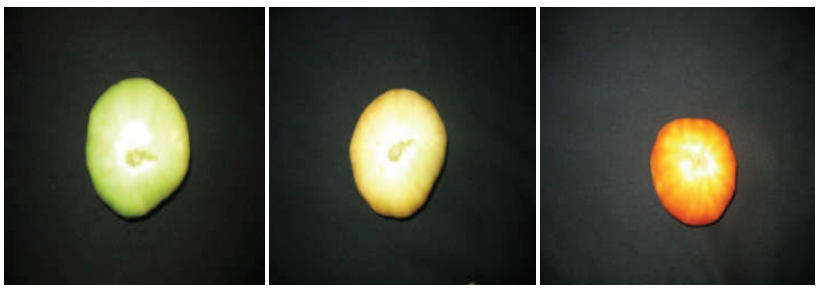


Fig.3.1 Architecture de la méthode basée sur les SVMs

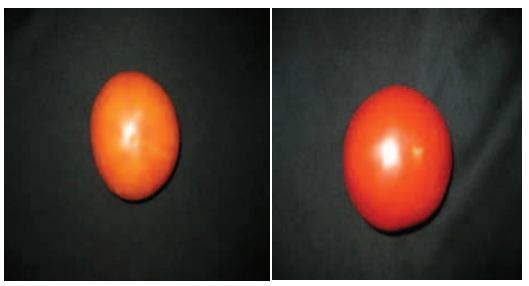
3.4.4 Prétraitement des images

Dans cette étape, les images vont subir plusieurs transformations pour obtenir des images dénuées de toute information inutile et pour avoir juste

l'objet qui nous intéresse. Ces transformations sont le redimensionnement des images en 250*250 pixels, la conversion des images RVB dans les deux systèmes L*a*b* (fig.3.3b) et HSV. Les opérations de filtrage, d'élimination du fond et enfin d'extraction du contour sont aussi effectuées.



(a) couleur verte (b) couleur tournante (c) couleur orange



(d) couleur rouge claire (e) couleur rouge foncé

Fig.3.2 Evolution de la couleur de la tomate

Le filtrage

L'opération de filtrage a pour but de supprimer le bruit contenu dans les images. Parmi les filtres existant, notre choix s'est porté sur le filtre médian car il élimine bien le bruit sans dégrader le contour. En effet chaque pixel de l'image est remplacé par la valeur médiane de son voisinage qui est une fenêtre 3x3.

La binarisation

La binarisation consiste à partitionner l'image en deux régions homogènes correspondant à l'objet et le fond avec un seuil déterminé à l'aide de la méthode d'Otsu [48]. Rappelons que le principe de cette méthode est

basé sur la séparation ou partitionnement des pixels de l'image en deux classes C_1 et C_2 à partir d'un seuil t, tel que C_1={0,1, … , t} *et* C_2={$t+1, \dots, n_g - 1$} *où* n_g est le nombre de niveaux de gris. La détermination du seuil optimum est obtenu en maximisant l'un des critères suivants:

$$\gamma(t) = \frac{\sigma_b^2}{\sigma_w^2}; \qquad \eta(t) = \frac{\sigma_b^2}{\sigma^2}; \qquad k(t) = \frac{\sigma^2}{\sigma_w^2}; \tag{3.5}$$

où σ_w^2 est la variance d'une classe, σ_b^2 la variance interclasse et σ^2 la variance totale. On obtient ainsi une image binaire (fig 3.3d) à partir de l'image à niveaux de gris.

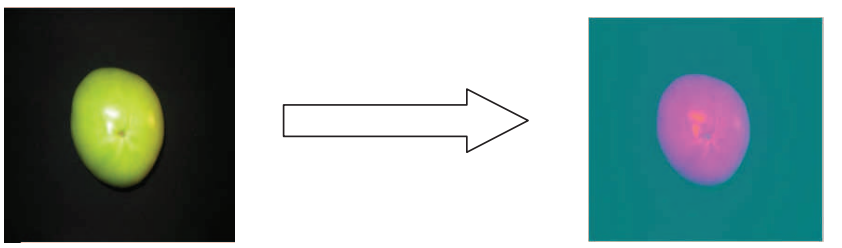
Suivi de contour

A partir de l'image binaire où l'on considère que les pixels à 1 font partie de l'objet et les pixels à zéro représentent le fond, on peut procéder à l'opération de suivi de contour par connexité [49] après l'extraction du fond à l'aide des opérations morphologiques (fig 3.3e).

Une fois toutes ces opérations achevées on passe à l'extraction des caractéristiques.

3.4.5 Extraction des caractéristiques

Plusieurs travaux [44][45][46], ont montré que le paramètre couleur est le seul paramètre qui évolue avec la maturité de la tomate. Ainsi, c'est ce paramètre que nous utiliserons dans notre application. En ce qui concerne la mise en œuvre de la première méthode de Syarir et *al,* que nous avons appliquée [45], pour chaque image exprimée dans l'espace L*a*b*, nous calculons les valeurs de a* de tous les pixels de l'image de l'objet. La moyenne de toutes ces valeurs de a* est évaluée. Cette moyenne constitue la caractéristique couleur de la tomate correspondante et elle est comparée aux valeurs déterminées dans le tableau (3.2) puis affectée à la classe correspondante parmi les cinq classes déjà prédéfinies.



(a) image d'origine (b) image L*a*b*

(c) image à niveau de gris (d) image binaire

(e) tracé du contour

Fig.3.3 Les opérations de prétraitement

En ce qui concerne la méthode basée sur les SVMs, les images sont converties dans l'espace HSV. Pour chaque composante H, S et V, nous calculons l'histogramme (fig.3.4) ainsi que les trois premiers moments statistiques qui sont:

- la moyenne : $\mu_i = \frac{\sum_{j=1}^{M.N} X_{ij}}{M.N}$ (3.6)
- l'écart-type : $\partial_i = \sqrt{\frac{1}{M.N}\sum_{j=1}^{M.N}(X_{ij} - \mu_i)^2}$ (3.7)
- le skewness : $S_i = \sqrt[3]{\frac{1}{M.N}\sum_{j=1}^{M.N}(X_{ij} - \mu_i)^3}$ (3.8)

où X_{ij} est la valeur du pixel j et i la composante H,S ou V. *NxM est* le nombre total de pixels de l'image. Ainsi chaque objet ou chaque tomate est représenté par un vecteur caractéristique composé des trois histogrammes et des neuf moments dont trois moments pour chaque composante. L'ensemble des vecteurs ainsi obtenus sera utilisé pour l'apprentissage et la classification des tomates.

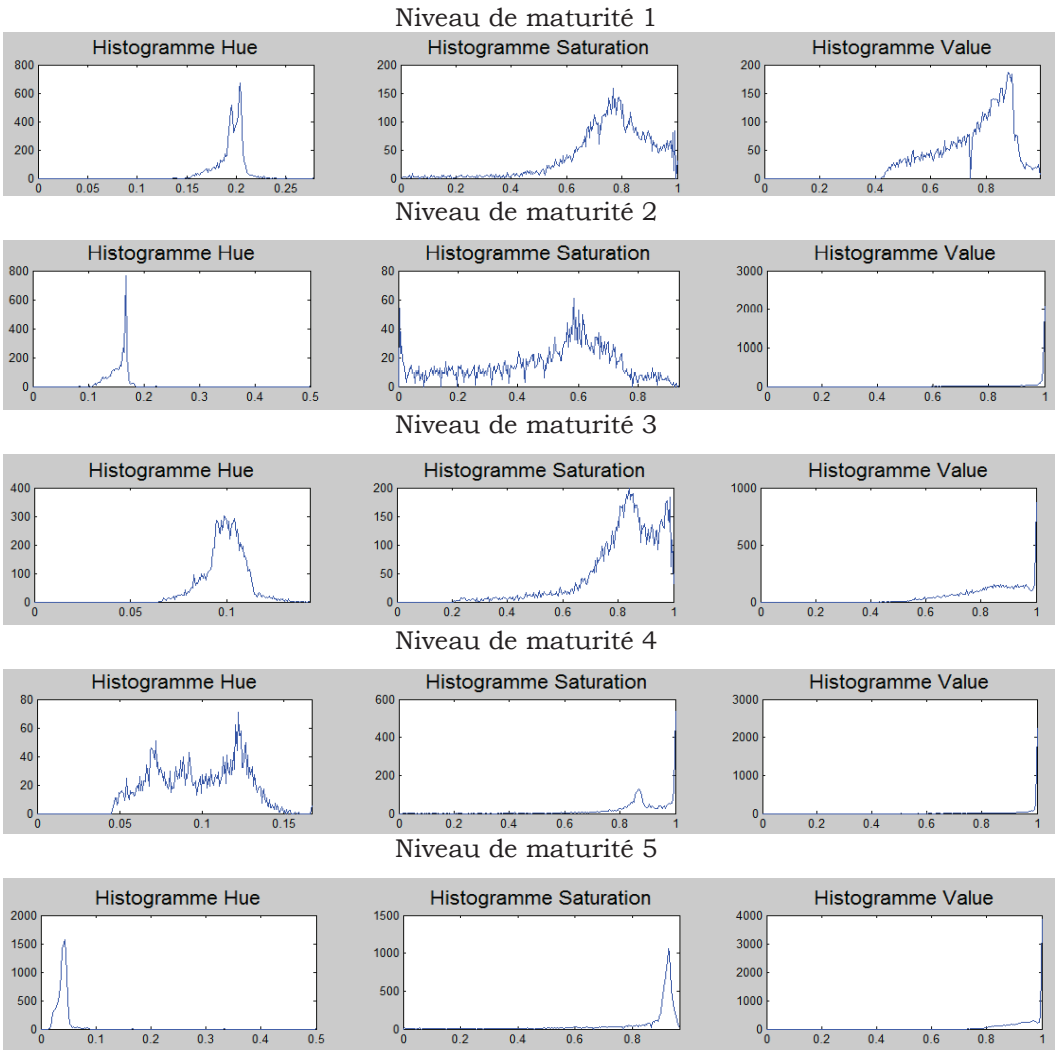


Fig.3.4 Histogrammes de chaque composante couleur de cinq échantillons

3.5 Tests et résultats

L'efficacité de la méthode de Syarir et *al* et la méthode basée sur les SVMs à été évaluée sur la base des cinquante images de tests. Ces images

sont réparties en cinq ensembles, chaque ensemble regroupant dix images correspondant à un niveau de maturité. Ainsi les résultats obtenus par les deux méthodes que nous avons appliquées sont présentés dans le tableau (3.3). Sur ce tableau, les cellules en couleur correspondent aux images mal classées. En effet pour le premier niveau de maturité (tomate verte), nous n'avons aucune erreur de classification pour les deux méthodes. Pour le deuxième niveau de maturité (couleur tournante), nous comptons une erreur de classification pour chacune des deux méthodes. Pour le quatrième stade de maturité (rouge clair), nous avons quatre erreurs pour la méthode de Syarir et *al*. et deux seulement pour les SVMs. Cependant, avec la méthode utilisant les SVMs, aucune erreur de classification pour le troisième et le dernier stade de maturité (orange et rouge foncé), tandis que la méthode de Syarir et *al* compte une erreur pour chacun de ces deux niveaux.

Les résultats ainsi obtenus sont récapitulés dans le tableau (3.4), où nous donnons le taux de classification TC (%) obtenu par chacune des deux méthodes pour les cinq classes concernant la maturité des tomates.

Tableau.3.4 Résultats de la classification

	Classe1	**Classe2**	**Classe3**	**Classe4**	**Classe5**	**Classification globale**
T.C(%) par Syarir et al.	100	90	90	60	90	**86**
T.C(%) par SVMs	100	90	100	80	100	**94**

Comme le montre le tableau (3.4) ci-dessus, nous avons des taux de classification plus ou moins proches, à l'exception de la quatrième classe qui correspond à la couleur rouge claire pour laquelle le taux est le plus faible pour les deux méthodes. En effet, nous avons un taux de classification de 60% pour la méthode de Syarir et *al*, et de 80% pour la méthode basée sur les SVMs, concernant la quatrième classe.

Tableau.3.3 Résultats obtenus sur la base des tests

Estimation du niveau de maturité par l'homme	exemple	Valeur de a*	Résultat de la méthode de Syarir et *al*	Résultat des SVMs
10% à 20%	1	-13.99	10% à 20%	10% à 20%
	2	-19.21	10% à 20%	10% à 20%
	3	-13.63	10% à 20%	10% à 20%
	4	-12.18	10% à 20%	10% à 20%
	5	-5.99	10% à 20%	10% à 20%
	6	-6.16	10% à 20%	10% à 20%
	7	-15.12	10% à 20%	10% à 20%
	8	-10.33	10% à 20%	10% à 20%
	9	-7.65	10% à 20%	10% à 20%
	10	-13.45	10% à 20%	10% à 20%
30% à 40%	1	-2.55	30% à 40%	30% à 40%
	2	-4.65	30% à 40%	30% à 40%
	3	-5.25	30% à 40%	30% à 40%
	4	-3.12	30% à 40%	10% à 20%
	5	-5.22	30% à 40%	30% à 40%
	6	3.52	50% à 60%	30% à 40%
	7	-0.35	30% à 40%	30% à 40%
	8	-3.67	30% à 40%	30% à 40%
	9	-0.14	30% à 40%	30% à 40%
	10	-4.13	30% à 40%	30% à 40%
50% à 60%	1	3.13	50% à 60%	50% à 60%
	2	2.89	70% à 80%	50% à 60%
	3	3.43	50% à 60%	50% à 60%
	4	4.01	50% à 60%	50% à 60%
	5	6.11	50% à 60%	50% à 60%
	6	6.05	50% à 60%	50% à 60%
	7	4.36	50% à 60%	50% à 60%
	8	3.17	50% à 60%	50% à 60%
	9	4.44	50% à 60%	50% à 60%
	10	5.89	50% à 60%	50% à 60%
70% à 80%	1	50.87	Maturité totale	70% à 80%
	2	9.95	70% à 80%	70% à 80%
	3	10.56	70% à 80%	Maturité totale
	4	13.12	70% à 80%	70% à 80%
	5	3.15	50% à 60%	70% à 80%
	6	18.23	70% à 80%	70% à 80%
	7	5.63	50% à 60%	50% à 60%
	8	14.11	70% à 80%	70% à 80%
	9	16.31	70% à 80%	70% à 80%
	10	53.37	Maturité totale	70% à 80%
Maturité totale	1	32.98	Maturité totale	Maturité totale
	2	28.48	Maturité totale	Maturité totale
	3	30.11	Maturité totale	Maturité totale
	4	42.36	Maturité totale	Maturité totale
	5	4.55	70% à 80%	Maturité totale
	6	52.04	Maturité totale	Maturité totale
	7	48.17	Maturité totale	Maturité totale
	8	44.00	Maturité totale	Maturité totale
	9	50.69	Maturité totale	Maturité totale
	10	54.53	Maturité totale	Maturité totale

Ces taux de classification de 86% pour la méthode de Syarir et *al*, et de 94% pour la méthode basée sur les SVMs peuvent être considérés satisfaisants avec un léger avantage pour les SVMs comparativement aux résultats déjà publiés [44][45][46]. Néanmoins, nous avons énuméré quelques avantages et inconvénients pour chacune des deux méthodes, que nous présenterons ci-après.

3.6 Discussions

Compte tenu des résultats obtenus, nous pouvons considérer que les deux méthodes, à savoir la méthode de Syarir et *al* et la méthode basée sur les SVMs, comme étant une bonne alternative comparée au travail effectué visuellement par l'être humain pour le jugement du niveau de maturité de la tomate. Il est de même pour l'estimation des délais de conservation de cette dernière, notamment quand il s'agit de les transporter à des distances différentes, autrement dit, quand il s'agit de distinguer les tomates qui seront transportées plus loin de celles qui seront destinées vers des distances plus réduites. Cependant, la performance de ces méthodes dépend fortement de la qualité des images acquises liées au matériel, surtout aux conditions d'éclairage, ce qui présente une limite pour ces deux méthodes et en général comme pour toute méthode de reconnaissance de formes. Toutefois, en ce qui concerne la méthode de Syarir et *al* , elle présente deux inconvénients. Le premier est qu'elle ne peut faire le traitement que sur une seule tomate à la fois. Le deuxième est qu'elle est spécialement conçue pour la catégorie des tomates, ce qui fait qu'on ne peut la généraliser à d'autres fruits ou légumes. Par contre, la méthode basée sur les SVMs est une méthode réutilisable. Elle peut facilement être adaptée à l'estimation du taux de maturité d'autres fruits et légumes. Pour ce faire, il suffit juste de revoir l'apprentissage avec de nouvelle bases d'apprentissage concernant l'objet à classifier. Néanmoins, un léger inconvénient des SVMs qu'il faut citer, concernant la base d'apprentissage est que, plus cette base est riche, plus la classification est meilleure. Donc il faut veiller à ce que la base d'apprentissage soit suffisamment consistante pour pouvoir réaliser une

classification correcte avec des taux satisfaisants. Avec cet important aspect de réutilisabilité de la méthode des SVMs, nous pouvons conclure que cette dernière l'emporte largement sur la méthode de Syarir et *al.*

3.7 Conclusion

L'approche mise en œuvre dans ce troisième chapitre qui consiste à diviser la période de l'évolution de la couleur de la tomate tout au long de sa maturation en cinq intervalles, qui correspondent aux cinq stades de maturité, afin de déterminer le niveau de maturité de ce fruit, a donné des résultats de classification satisfaisants pour les deux méthodes que nous avons utilisées, à savoir la méthode de Syarir et *al* et la méthode basée sur les SVMs. Néanmoins, les deux méthodes présentent quelques limites, principalement celle liée à la qualité des images des tomates et aux différentes opérations de prétraitement qui doivent être efficaces afin d'éliminer le maximum de bruit pour garantir une meilleur précision dans le calcul de la valeur de a* de la couleur de la tomate concernant la méthode de Syarir et *al*, et pour le calcul du vecteur caractéristique concernant les SVMs. Par ailleurs, la généralisation aisée de la méthode des SVMs à d'autres fruits et légumes, fait de celle-ci, une bonne alternative pour la classification automatique des fruits et légumes selon leurs niveaux de maturité, contrairement à la méthode de Syarir et *al* qui ne peut être utilisée que dans le cas des tomates.

Conclusion générale

Le travail que nous avons présenté dans ce mémoire a été dédié à la reconnaissance automatique de formes évolutives. En effet lorsque les caractéristiques des objets évoluent avec le temps, il est nécessaire d'utiliser des méthodes de classification dynamiques afin de détecter les différentes évolutions que subissent les classes et de mettre à jour les nouveaux paramètres de ces dernières. Différentes méthodes de classification dynamiques ont été présentées dans le premier chapitre et la méthode des K plus proches voisins flous en version dynamique (KPPVFD) a été appliquée sur différentes bases contenants des classes générées artificiellement selon la loi Gaussienne. En effet l'utilisation de ces classes nous a permis d'étudier les différentes évolutions telles que le déplacement, la fusion et la scission. La méthode des KPPVFD a montré son efficacité dans la détection de ces évolutions ainsi que leurs adaptations et leurs validations. Toutefois, lorsque les évolutions des classes sont périodiques, la classification devient statique en divisant la période de l'évolution sur des intervalles bien définis. Cette approche a été abordée et appliquée dans le troisième chapitre pour l'estimation du niveau de maturité de la tomate réalisée, par deux méthodes, à savoir la méthode de Syarir et *al* et la méthode des SVMs. Ces deux méthodes utilisent respectivement les images convertie dans le système de représentation L*a*b*, et les images converties dans le système de représentation HSV. Les résultats ainsi obtenus sont satisfaisants.

Bien que la méthode des KPPVFD ait été appliquée sur des classes générées artificiellement, n'ayant pas pu obtenir des classes réelles avec des caractéristiques évolutives, cette méthode peut être aisément appliquée sur

des cas réels tel que des systèmes de diagnostic industriel afin de tirer profit des avantages de cette méthode.

Si l'application entreprise dans ce mémoire parait naturelle, d'autres applications réelles et plus complexes peuvent être envisagées comme dans le cas de vieillissement de la peau.

On peut aussi penser à adapter d'autres méthodes de classification automatique connues efficaces au cas de données dynamiques.

Références bibliographiques

[1] L. Hartert, "*Reconnaissance des formes dans un environnement dynamique: appliquée au diagnostic et au suivi des systèmes évolutifs*". Thèse de doctorat, université Reims Champagne-Ardenne 2010.

[2] V. Gunes, "*Reconnaissance de formes évolutives par combinaison, coopération et sélection de classifieurs*". Thèse de doctorat, Université de la Rochelle 2001.

[3] C. Lurette, "*Développement d'une technique neuronale auto-adaptative pour la classification dynamique de données évolutives: application à la supervision d'une presse hydraulique*". Thèse de doctorat, université de Lille 2003.

[4] H. Amadou Boubacar, "*Classification Dynamique de Données non-stationnaires : Apprentissage séquentiel des Classes évolutives".* Thèse de Doctorat de l'Université des Sciences & Technologies de Lille, France, 2006.

[5] T. Eltoft, " *A new Neural Network for Cluster-Detection-and-Labeling*". *IEEE Trans. on Neural Networks*, vol. 9(5), pp: 1021-1035, 1998.

[6] V.N. Vapnik "*The Nature of Statistical Learning Theory*". Springer-Verlag New York, 1995.

[7] V. Guigue, "*Méthodes à noyaux pour la représentation et la discrimination de signaux non-stationnaires"*. Thèse de Doctorat de l'INSA de Rouen, France, 2005.

[8] J.A.K. Suykens, J. Vandewalle., "*Least squares support vector machine classifiers*", Neural Processing Letters 9 (3), pp. 293-300, 1999.

[9] S. Rüping., "*Incremental Learning with Support Vector Machines*". IEEE International Conference on Data Mining (ICDM 01), San Jose, CA, 2001.

[10] F. Dufrenois, J. Colliez , D. Hamad, " *Bounded Influence Support Vector Regression for Robust Single-Model Estimation*". IEEE Transactions on Neural Networks 20 (11), pp. 1689-1706, 2009.

[11] B. E. Boser, I.M. Guyon, V.N. Vapnik, *"A training algorithm for optimal margin classifers"*. 5th Annual Workshop on Computational Learning Theory, pp. 144-152, Pittsburgh, USA, 1992.

[12] Y. Guermeur*, " Combining Discriminant Models with New Multi-Class SVMs"*. Pattern Analysis and Applications, vol. 5(2), pp: 168-179, 2002.

[13] S. Borer, " *New Support Vector Algorithms for Multi-categorical Data: Applied to Real-Time Object Recognition"*. Thèse de doctorat, EPFL, Lausanne Suisse, 2003.

[14] J. Salomon, " *Support Vector Machines for Phoneme Classification"*. Master of science, School of Artificial Intelligence, University of Edinburgh, 2001.

[15] K. Boukharouba, S. Lecoeuche, "*Online Clustering of Non-stationary Data Using Incremental and Decremental SVM*". International Conference on Machine Learning and Applications, pp. 336-345, *2009.*

[16] S. Marsili-Libell, "*Adaptive Fuzzy Monitoring and Fault Detection*", International Journal of COMADEM, 1998.

[17] J.C. Bezdek, "*Pattern Recognition with fuzzy objective function algorithm".* Plenum Press, 1981.

[18] C. Byington, M. Roemer, G. Kacprzynski, T. Galie," *Prognostic Enhancements to diagnostic Systems for Improved Condition-based maintenance*". IEEE Aerospace Conference, Big Sky, Montana, 2002.

[19] F.E. Ciarapica, G. Giacchetta, *"Managing the condition-based maintenance of a combined-cycle power plant: an approach using soft computing techniques*". Journal of Loss Prevention in the Process Industries, *19*, pp. 316-325, 2006.

[20] O. Dragomir, R. Gouriveau, N. Zerhouni, "*Adaptive neuro-fuzzy inference system for mid term prognostic error stabilization*". International Journal of Computers Communications & Control, 3, pp. 271-276, 2008.

[21] A. Muller, M.C. Suhner, B. Iung, "*Formalisation of a new prognosis model for supporting proactive maintenance implementation on industrial System*". Reliability Engineering and System Safety, 2008.

[22] L. Rafiq, M. Chrys santhopoulos, T. Onoufriou, "*Performance updating of concrete bridges using proactive health monitoring methods*". Reliability Engineering and System Safety, 86(3), pp. 247-256, 2004.

[23] H. Wang, W. Fan, P.S. Yu, J. Han, "*Mining concept-drifting data streams using ensemble classifiers*". IEEE Trans Image Process. 12 (9), pp. 1120-1131, 2003.

[24] R. Zemouri, "*Contribution à la surveillance des systèmes de production à l'aide des réseaux de neurones dynamiques: application à la e-maintenance*", Thèse de doctorat de l'université de Franche-Comté, 2003.

[25] D. M. Michie, D. J. Spiegelhalter, "*Machine learning, Neural and Statistical Classification*". C.C. Taylor (Eds), Chichester: Ellis Horwood, 1994.

[26] L. Angstenberger, R. Weber, et W. Meier, " *Data Engine: A software tool for intelligence data analysis"*. VDI-Verlag, Düsseldorf, pp 348-350, Germany 2000.

[27] I. D. Guedalia, M. London, M. Werman," *An On-line Agglomerative Clustering Method for Non-Stationary Data*". Neural Computation, 11 (2), pp 521-540, 1999.

[28] M. Last," *Online Classification of Non-stationary Data Streams*". Intelligent Data Analysis, Vol. 6, No. 2, pp. 129-147, 2002.

[29] L. Cohen, G. Avrahami, M. Last," *Incremental Info-Fuzzy Algorithm for Real Time Data Mining of Non-Stationary Data Streams*". TDM Workshop, Brighton UK, 2004.

[30] R. Rengaswamy," *A syntactic pattern-recognition approach for process monitoring and fault diagnosis*". Engineering Applications of Artificial Intelligence, pp. 35-51, 1995.

[31] G. Stephanopoulos, G. Locher, M. Duff, R .Kamimura" *Fermentation data base mining by pattern recognition*". Biotechnology and Bioengineering, vol. 53-5, pp. 443-452, 1997.

[32] B.R. Bakshi, G. Stephanopoulos, "*Representation of process trends. IV: Induction of real-time patterns from operating data for diagnosis and supervisory control*". Computers and Chemical Engineering, 18(4), pp. 267-302, 1994.

[33] J.M. Keller, M.R. Gray and J.A. Givens,"*A fuzzy k-nearest neighbor algorithm*". IEEE Transactions on Systems, Man, and Cybernetics, SMC-15(4), pp. 580–585, 1985.

[34] T.M. Cover, P.E. Hart, "*Nearest neighbor pattern classification*". IEEE Transactions on Information Theory, vol. 13, pp. 21-27, 1967.

[35] E. Fix, J.L Hodges," *Discriminatory analysis: Non-parametric discrimination: Consistency properties*". USAF School of Aviation Medicine, Randolph Field, TX, pp. 261–279, 1951.

[36] H. Frigui, R. Krishnapuram, "*Clustering by competitive agglomeration*". Pattern Recognition, vol. 307, 1997.

[37] X.L. Xie, G. Beni, "*A validity measure for fuzzy clustering*". IEEE transactions on Pattern Analysis and Machine Intelligence, 1991

[38] USDA/AMS, "*United states standards for grades of fresh tomatoes"*, Washington,1991.

[39] G. Polder, G. Van der Heijden et I. Young*, " Spectral image analysis for measuring ripeness of tomatoes*". Transactions-American Society of Agricultural Engineering, pp 1155-1162, 2002.

[40] E.R. Guerrero, G.M. Benavides," *Automated system for classifying hass avocados based on image processing techniques*". IEEE, Colombian conference on communications and computing pp. 1-6, 2014.

[41] M. Paulraj, C.R. Hema, K. Pranesh et M.R. Siti Sofiah, " *Color recognition algorithm using a neural network model in determining the ripeness of a banana*". International Conference on Man-Machine Systems (ICoMMS), 2009. Univ. Malysia.

[42] M. Dadwal et V. Banga, "*Estimate ripeness level of fruits using rgb color space and fuzzy logic technique*". International Journal of Ingenieering and Advanced Technology, pp 225-229, 2012.

[43] D.J. Damiri et C. Slamet, *" Application of image processing and artificial neural networks to identify ripeness and maturity of the lime(citrus medica)".* International Journal for basic. and applied Science pp 171-179, 2012.

[44] N. E. Bendary, E.E. Hariri, A.E. Hassanien, A. Badr, "*Using machine learning techniques for evaluating tomato ripeness*". Elsevier, Expert Systems with Application, pp 1892-1905, 2014.

[45] W.M. Syarir, A. Suryanti et C. Connsynn, " *Color grading in tomato maturity estimator using image processing technique*". Faculty of Computer Systems and Software Ingineering, Universiti Malaysia Pahang. IEEE 2009 pp 276-280.

[46] G. yoshinori, H. Zhang, M. Nagata,*" Judgment on level of maturity for tomato quality using L*a*b* color image processing*". IEEE International Conference on Advanced Intelligence Mechatronics, pp 1355-1359, 2003.

[47] R. Sève, "*Science de la couleur: Aspects physiques et perceptifs*". Chalagam Edition, Juin 2009.

[48] P.K Saho, S. Soltani, A.K.C. Wong and Y.C. Chen". *A Survey of thresholding techniques*", Computer Vision Graphics and Image Processing, pp 233-260. 1988.

[49] J. Canny," *A computational approach to edge detection*". IEEE Transactions on Pattern Analysis and Machine Intelligence, pp 679-698, 1986.