Université Paris 8
Master 2 Technologie et Handicap
Sous la direction de : Jaime Lopez Krahe et Pascale Pousset

Reconnaissance de la marque d'une canette à partir d'une photo numérisée pour déficients visuels

Catherine SAUVAGET et Bounkong KHAMPHOUSONE
<catherine.sauvaget@gmail.com>

Saint Denis Université, vendredi 23 fevrier

Remerciements

Je tiens à remercier :

- Jaime LOPEZ KRAHE sans lequel ce projet n'aurait pas vu le jour cette année ;
- Philippe FOUCHER et Anis ROJBI dont les précieux conseils ont été d'une grande aide ;
- Pascale POUSSET qui m'a aidé à organiser le projet tout au long de sa réalisation ;
- Bounkong KHAMPHOUSONE qui a gentiment accepté de participer à ce projet ;
- Toutes les personnes qui nous ont aidés de près ou de loin à l'avancement de ce projet de recherche.

Table des matières

Table des figures

Résumé

Le projet

Nous avons développé un système permettant d'identifier, à partir d'une photo prise par un PDA, la marque et la nature d'une canette. Cette application est destinée à être portée sur PDA afin que la population ciblée puisse s'en servir à l'endroit où elle se trouve.

Le principe est le suivant : l'utilisateur prend une photo à partir de son PDA et le système lui indique la marque de la canette (par exemple Oasis) ainsi que la nature (par exemple Tropical).

Le but de ce projet est de valider les méthodes par lesquelles nous ferons la reconnaissance de la canette car le temps imparti est trop court afin de pouvoir l'embarquer sur PDA. En effet, la problématique de ce sujet réside dans le fait que le futur utilisateur, lorsqu'il prendra la photo de la canette, ne pourra pas cadrer aisément. D'autres problèmes que celui du cadrage se posent. Par exemple, l'utilisateur étant déficient visuel, il ne peut pas concevoir l'influence qu'aura la lumière environnante sur la canette. Notre sujet d'étude porte sur des objets qui peuvent parfois avoir de forts reflets sur leur surface. Notre travail consistera donc à trouver et implémenter les méthode qui semblent les meilleures afin d'obtenir de bons taux de reconnaissance.

Population visée

Cette application a été créée pour rendre service aux déficients visuels qui ne peuvent pas discerner les diverses marques et natures de canettes puisque celles-ci ont une forme standard. Ainsi, ces personnes peuvent choisir de façon autonome une canette dans un réfrigirateur à domicile, ou dans tout autre commerce : épicerie, grande surface, etc.

Méthodes employées

Pour réaliser cette application, nous avons utilisé le logiciel Matlab, qui nous a permis un gain de temps précieux car beaucoup de fonctions dont nous nous servons sont déjà implémentées. Après une courte phase d'initiation à Matlab, nous avons pu implémenter nos algorithmes.

Avant toute implémentation, il a été nécessaire de créer une base de données avec des canettes collectées qui *in fine* sont au nombre de 25. Pour chacune de ces canettes, l'image gravée sur le tour du cylindre est représentée dans la base de données par dix vues différentes.

Par la suite, nous avons codé les algorithmes permettant de parcourir la base de données. Un algorithme permet de mettre toutes ces photos, ainsi que celle prise par l'utilisateur, sous une taille donnée qui est de 32x32. Ceci permet de parcourir moins de pixels lors de la phase de corrélation entre la photo prise par l'utilisateur et les photos de la base de données. Cette taille est celle qui permet les meilleurs résultats, elle a été déterminée suite à une batterie de tests. En effet, il ne faut pas que l'image soit trop petite sinon la corrélation est mauvaise et il ne faut pas que l'image soit trop grande non plus, sinon la corrélation met trop de temps.

Une fois cette étape réalisée, nous nous sommes attelés à extraire la canette du reste de l'image. Pour ce faire, nous avons utilisé la méthode de la transformée de Hough qui permet de détecter des lignes droites dans l'image. Sur une photo, une canette étant presque rectangulaire, nous avons essayé de détecter les lignes composant le rectangle qui entoure la canette. Cette étape est difficile à réaliser car nous détectons beaucoup plus de lignes que nécessaire et parfois pas les bonnes. Nous avons alors opté pour une autre méthode plus simple et plus rapide. Faute de temps, nous avons posé comme restrictions que nos photos soient cadrées et que la luminosité soit correcte pour ne pas avoir trop de reflets. Nous nous sommes servi de ceci afin de sélectionner une fenêtre centrée dans l'image, ce qui nous permet d'extraire une partie de l'image de la canette.

La fenêtre extraite vaut 1/3 en longueur et 1/3 en largeur d'une image 32x32. Cette valeur a été obtenue suite à des tests. Elle est celle avec laquelle nous obtenons les meilleurs résultats. Nous la corrélons avec les photos en 32x32 de la base de données. Cette fenêtre est donc décalée sur toute la longueur et sur toute la largeur de chaque image 32x32 de la base de données. Nous fixons un seuil manuellement et gardons la photo courante dans notre liste de résultats si à un temps t donné, la corrélation a été supérieure au seuil.

Nous testons la corrélation sur deux types de matrices. La première méthode est appliquée sur une matrice dont les couleurs RGB ont été mises au système HSL. Nous utilisons simplement la valeur de luminance pour faire la corrélation. La deuxième méthode est appliquée sur chaque filtre R, G et B de l'image couleur. On dénombre donc, trois corrélations pour cette méthode au lieu d'une pour la première. Par conséquent, cette dernière met trois fois plus de temps à être réalisée.

Cette liste de résultats est classée par ordre décroissant. Les 5 meilleurs résultats sont gardés afin d'être réexaminés à une échelle plus grande (64x64) pour les affiner.

Résultats

Les résultats montrent que la méthode par couleurs RGB donne de meilleurs résultats que la méthode par luminance. Pour presque toutes les canettes ayant un résultat avec le système par luminance, un résultat supérieur est obtenu avec la méthode par crominance (couleurs). La réciproque est fausse. Il arrive que l'on obtienne un résultat avec la méthode couleurs et alors qu'il n'y a pas de canette reconnue avec la méthode par luminance.

Malgré les meilleurs résultats de la méthodes RGB, celle-ci met trois fois plus de temps que la méthode par luminance qui ne nécessite qu'une seule corrélation. La meilleure méthode prend trois minutes pour une base de données de 250 images. Avec la méthode par luminance, le temps de calcul est seulement d'une minute.

Le taux d'échec avec la méthode couleur est très faible : seulement 9 canettes ne sont pas reconnues contre 21 canettes avec la luminance sur un total de 75 canettes. Les canettes non reconnues sont souvent celles qui permettent de forts reflets comme schweppes light, Kas light, Heineken, ou encore Coca light Lemon.

Nous avons calculé le taux de reconnaissance pour lequel la canette recherchée se trouve dans la liste des résultats. Le taux de cette reconnaissance est de 72% pour la luminance et de 88% pour la méthode couleurs.

Nous avons également cherché à savoir combien de fois la marque de la canette recherchée arrive en première position dans notre liste de résultats. En luminance, le type de canette recherché arrive dans 62,66% des cas en première position contre 76% des cas avec la méthode RGB. Le fait de retraiter les images à l'échelle supérieure permet d'améliorer le taux de reconnaissance d'environ 3%.

1 Introduction

Les applications destinées au handicap sont de plus en plus nombreuses et se font dans des domaines très diversifiés. De nombreux travaux portants sur la déficience visuelle ont été réalisés à ce jour. Cependant il existe très peu de dispositifs permettant d'aider les personnes non-voyantes dans un rituel auquel chacun se livre tous les jours et qui est vital : boire et manger.

Les aliments sont de nos jours conservés dans divers types d'emballages. Pour cette raison, nous nous sommes attachés à un type d'aliments dont la forme de l'emballage est commune à toutes les marques. Ainsi, nous nous attendons à obtenir des résultats satisfaisants avec un objet de taille standard ce qui permettra peut-être, par la suite, de développer d'autres dispositifs ayant le même but avec d'autres aliments.

Cette application a été créée pour rendre service aux déficients visuels qui ne peuvent pas discerner les diverses marques et natures de canettes puisque celles-ci ont une forme standard. Ainsi, ces personnes peuvent choisir de façon autonome une canette dans un réfrigérateur à domicile, ou dans tout autre commerce : épicerie, grande surface, etc.

Dans cet état d'esprit, nous avons développé un système permettant d'identifier, à partir d'une photo prise par un PDA, la marque et la nature d'une canette. Le but de ce projet est de valider les méthodes par lesquelles nous ferons la reconnaissance de la canette, le temps imparti étant trop court afin de pouvoir l'embarquer sur PDA. Cette application est donc destinée à être portée sur PDA afin que la population ciblée puisse s'en servir où qu'elle se trouve.

Le principe est le suivant : l'utilisateur prend une photo à partir de son PDA et le système lui indique la marque de la canette (par exemple Oasis) ainsi que sa nature (par exemple Tropical).

Le but de ce projet est de valider les méthodes par lesquelles nous ferons la reconnaissance de la canette car le temps imparti est trop court afin de pouvoir l'embarquer sur PDA. En effet, la problématique de ce sujet réside dans le fait que le futur utilisateur, lorsqu'il prendra la photo de la canette, ne pourra pas cadrer aisément. D'autres problèmes que celui du cadrage se posent. Par exemple, l'utilisateur étant déficient visuel, il ne peut pas concevoir l'influence qu'aura la lumière environnante sur la canette. Notre sujet d'étude porte sur des objets qui peuvent parfois avoir de forts reflets sur leur surface. Notre travail consistera donc à trouver et à implémenter les méthodes qui semblent les meilleures afin d'obtenir de bons taux de reconnaissance.

Le plan de notre document se divise en six grandes parties. Il nous sera nécessaire, avant toute chose, d'introduire certaines notions. Nous détaillerons ensuite les différentes phases de l'analyse préalable de notre sujet. Nous expliquerons la manière dont nous avons

modélisé et conçu notre projet puis nous introduirons quelques connaissances générales sur le logiciel dont nous nous sommes servi afin de réaliser notre programme. Les détails du développement de chaque module seront ensuite expliqués. Enfin, nous ferons un bilan de ce projet et des résultats obtenus et nous donnerons les perspectives envisagées pour ce projet.

2 Glossaire des termes utilisés dans le projet

- Canette : petite boite métallique contenant un liquide. Les canettes existent en différentes tailles. Nous utilisons, pour notre projet, des tailles de canettes identiques (standard) afin de minimiser les problèmes pour la reconnaissance de formes.

- Reconnaissance de formes : La reconnaissance de formes ou reconnaissance de motifs est un sous-domaine de l'apprentissage automatique et peut être défini comme l'action de prendre en entrée des données brutes afin de prendre une décision basée sur la catégorie de ces données. Le but de la reconnaissance de formes est de classifier des données en se basant sur des connaissances à priori ou sur de l'information statistique puisée dans les motifs. Les motifs à classer sont habituellement des ensembles de mesures ou d'observations qui définissent des points dans un espace multidimensionnel approprié [4].

- PDA : Un assistant personnel est un appareil numérique portable, souvent appelé par son acronyme anglais PDA pour Personal Digital Assistant[4].

- Base de données : Une base de données est un ensemble structuré et organisé permettant le stockage de grandes quantités d'informations afin d'en faciliter l'exploitation (ajout, mise à jour, recherche de données). Une base de données se traduit physiquement par un ensemble de fichiers sur disque [4].

- Corrélation : En probabilités et en statistiques, étudier la corrélation entre deux ou plusieurs variables aléatoires ou statistiques, c'est étudier l'intensité de la liaison qui peut exister entre ces variables. La liaison recherchée est une relation affine [5].

- Représentation pyramidale : La représentation pyramidale de l'image est une structure de données dans laquelle l'image est représentée sur des niveaux successifs par des copies de l'image originale en utilisant des résolutions différentes [1].

- Logiciel : Un logiciel est l'ensemble des éléments informatiques qui permettent d'assurer une tâche ou une fonction [4].

- Embarqué : On désigne sous le terme informatique embarquée les aspects logiciels se trouvant à l'intérieur des équipements n'ayant pas une vocation purement informatique. L'ensemble logiciel-matériel intégré dans un équipement constitue un système embarqué [4].

3 Analyse préalable

3.1 Présentation du contexte et des problématiques du sujet

Le contexte de ce sujet se situe dans l'idée de la vie quotidienne de l'utilisateur. La personne déficiente visuelle dans le cadre de son domicile, de son travail, de ses déplacements, de ses courses etc... peut avoir l'envie d'acheter une ou plusieurs canettes.

Un des problèmes pouvants se poser pour les concepteurs d'un tel dispositif est celui relatif aux différents environnements dans lesquels l'utilisateur se déplace. En effet, les divers lieux cités précédemment ont des intensités lumineuses différentes et les canettes seront également photographiées sur des fonds très différents.

Un autre problème vient du fait que les utilisateurs n'ont pas la notion de l'angle d'ouverture de l'appareil photo de leur PDA. De ce fait, en tenant une canette dans leur main devant l'appareil, ils ne sont pas conscient de l'ampleur de ce qui est cadré.

Ces problématiques nous obligent donc à prendre certaines restrictions en considérant le temps qui nous est imparti.

3.2 Périmètre, domaine couvert par le projet

Notre objectif consiste à réduire le handicap des déficients visuels dans leur vie au quotidien, à l'aide d'un logiciel permettant de reconnaître la marque et la nature d'une canette. Celui-ci sera dans un futur proche embarqué sur PDA. Dans cette optique, notre système devra être en collaboration avec des périphériques du PDA (Fig 1) :

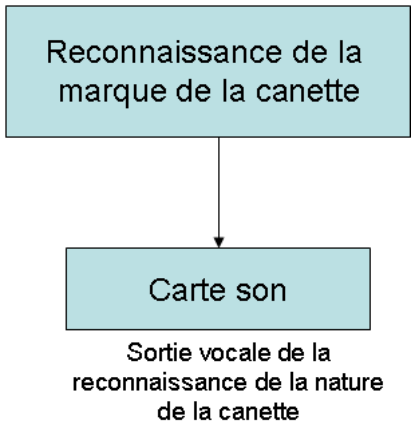


FIG. 1 – Diagramme de collaboration

La collaboration entre la carte son et notre programme permet à l'utilisateur de

connaître de façon auditive le résultat trouvé. Il pourra éventuellement demander à entendre les marques et natures suivantes trouvées, contenus dans la liste des résultats.

Par ce dispositif, il lui sera également possible d'écouter les diverses marques existantes dans la base de données. Ceci peut servir quand l'utilisateur a plusieurs canettes à sa disposition dont il connaît les différentes marques. Dans un cas comme celui-là, l'utilisateur ne sait pas laquelle des marques il va prendre en main. Avec un tel dispositif, il peut s'assurer que les marques qu'il possède sont dans la base de données.

3.3 Etude de l'existant et domaines connexes

A notre connaissance, le problème de reconnaissance de canettes n'a jamais été vraiment posé. Néanmoins, dans le domaine de la reconnaissance de formes ou d'objets, il existe un bon nombre de recherches ayant été effectuées (voire [9]).

Il existe par exemple des programmes reconnaissant des pièces de monnaies de pays différents et d'années différentes [10].

D'autres traitent de la reconnaissance de formes détectées statistiquement dans laquelle existe par exemple la reconnaissance de visage par analyse en composantes principales (PCA) [6].

La reconnaissance de plaques d'immatriculation fait également partie des domaines connexes. Cette reconnaissance est de nos jours très utilisée notamment par les radars sur les routes. Par exemple dans l'article de reconnaissance de plaques d'immatriculation, les auteurs utilisent la détection de contours, la localisation de caractères et un algorithme génétique pour trouver la plaque dans une image.

L'utilisation de masques binaires dans une rétine de reconnaissance de formes est un autre cas d'utilisation des outils possibles de reconnaissance de formes. En effet tous les types de filtres tels que les filtres de stretching linéaire sont beaucoup utilisés dans le traitement d'images. On retrouve aussi fréquemment les outils de morphologie mathématiques dans ce type de reconnaissance.

Il est possible de voir sur le site de la RFIA (Reconnaissance des Formes et Intelligence Artificielle) [8] de nombreuse recherches de reconnaissance de formes en imagerie.

3.4 Limites

Le temps imparti étant relativement court pour une telle réalisation, certaines exigences ont été posées dès le départ.

La partie la plus importante des photos est bien sûr celle où se trouve la canette. Nous

nous désintéressons du fond de l'image. Nous supposons la photo cadrée sur la canette de façon à ce que les lignes constituées par la forme de la canette (formant plus ou moins un rectangle) soient le plus possible parallèles aux lignes constituant la longueur et la largeur de la photo. Nous prenons également des photos sur un fond uni de couleur blanche afin que la détection de la canette soit plus simple. Néanmoins, nous avons conservé le fait que la canette est tenue dans les mains de l'utilisateur lors de la prise de la photo, ce qui constitue une difficulté lors de la détection de cette canette.

Une fois extraite la partie de l'image constituée par la canette, l'étape suivante consiste à réaliser la phase de corrélation. Il est alors facile d'imaginer que certains problèmes peuvent se poser lors de cette phase. En effet, les canettes sont métalliques et reflètent donc la lumière de façon plus ou moins forte selon la couleur et la brillance d'origine de la canette. Nous faisons donc en sorte d'être dans un milieu le plus idéal possible du point de vue de la luminosité : ni trop lumineux afin de ne pas avoir de trop gros reflets, ni trop sombre afin que les couleurs soient visibles. Sans ces restrictions, nos résultats seraient faussés.

3.5 Liste des exigences

Les exigences pour ce projets sont les suivantes :

- créer un simulateur permettant de tester l'efficacité des algorithmes choisis ;
- Obtenir un résultat en une durée inférieure à 1 minute (ce temps est aussi grand car Matlab n'est pas très rapide) ;
- Obtenir un taux de réussite supérieur à 60% ;
- L'ensemble des photos d'une marque ou nature de canette dans la base de données fournit la totalité de l'image fixée autour de la canette.
- Avoir un objectif propre afin que les résultats soient corrects ;

3.6 Choix des techniques développées : rapport qualité/coût

Les techniques développées sont issues de nombreuses concertations avec nos responsables de suivis de projet. Celles qui sont ressorties de nos réunions sont celles qui semblent donner les résultats les plus fiables quant au taux de reconnaissance. Nous choisissons des techniques qui nous semblent performantes sur un tel sujet tout en essayant de garder un temps de calcul convenable.

3.6.1 La transformée de Hough

Cette technique nous permet d'extraire la forme de la canette du reste de la photo. Le principe général d'une reconnaissance de forme est le suivant :

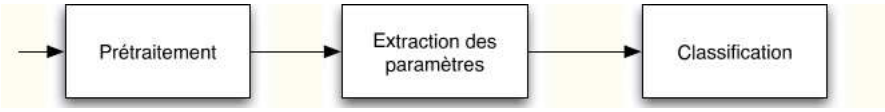


FIG. 2 – Schéma classique de la reconnaissance de formes

Cette méthode a été retenue car elle est rapide et il semble que les droites qu'elle permet de détecter soient fiables. Nous savions d'avance qu'il y aurait beaucoup plus de lignes détectées que les quatre qui nous seraient utiles. Pour cette raison, nous avons développé un algorithme de relation entre les longueurs entre elles et les largeurs entre elles. Nous expliquons en détail cette étape dans le chapitre lié à la conception. Une fois les quatre lignes détectées, nous extrayons le rectangle ainsi constitué. Ce dernier constitue notre nouvelle image dans laquelle ne se trouve que l'image de la canette, l'image est donc plus petite et il n'y a plus de fond. C'est de cette image que nous sélectionnons une fenêtre que l'on corrèle avec les images de la base de données.

3.6.2 Le plan de rechange pour Hough

Au cas ou nous aurions des problèmes pour extraire la canette avec la transformée de Hough, nous avons conçu un plan de secours. Puisque dans nos restrictions il a été décidé de cadrer la canette, nous pouvons extraire une fenêtre, d'une taille donnée, du centre de l'image d'origine. Ceci est plus rapide que la transformée de Hough, puisque nous supprimons une étape pour le même résultat.

Cette méthode est possible du fait que les canettes sont cadrées le plus près possible de l'objectif tout en conservant la totalité de la canette dans la photo prise. Cette solution est valable dans notre cas, mais elle ne permet pas de régler les problèmes d'homothétie.

3.6.3 Le système pyramidal

Le but de ce système est de concevoir des images réduites à partir des photos de la base de données et de la photo prise par l'utilisateur. Ceci permet que la phase de corrélation soit plus rapide, car il y a moins de pixels à analyser. Les images de taille supérieure sont utilisées pour les premières canettes obtenues afin d'affiner le résultat.

3.6.4 Les systèmes RGB et HSL

Le système le plus connu du grand public est probablement le système RGB ou RVB (pour Rouge Vert Bleu). La couleur y est en effet composée à partir des trois couleurs primaires, selon le principe de la synthèse additive.

Ce système part donc d'un état noir au repos, l'addition totale des trois couleurs primaires donnant le blanc. Il s'agit typiquement du principe de fonctionnement des moniteurs d'ordinateur et des télévisions, le signal passant entre l'ordinateur et son moniteur étant de cette nature.

Il s'agit aussi du principe de fonctionnement de l'oeil humain, qui possède trois types de récepteurs, chacun sensible à une couleur [3].

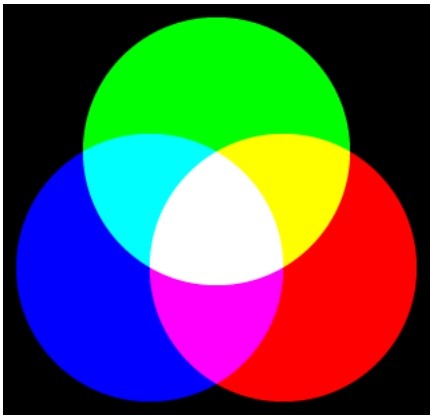


FIG. 3 – Principe de la synthèse additive : modèle RGB

Le format HSL (Hue Saturation Light) ou TSL (Teinte Saturation Lumière) est organisé très différemment. Il est plus proche de la perception humaine et de la physique et est apparenté au système YUV utilisé dans la réception des téléviseurs (avant d'être traduit en RGB).

Il décompose une couleur selon trois composantes :

- La teinte est la couleur pure qui est à la base de la couleur effective. En physique, elle correspond à la longueur d'onde de la couleur.
- La lumière détermine la luminosité. Une lumière à 0 entraîne une couleur noire, quelque soit la teinte.
- La saturation est le taux de la composante teinte dans la couleur par rapport au bruit visuel (qui est par définition aléatoire donc blanc -un mélange de toutes les longueurs d'onde-). A 0, la teinte ne se distingue pas du reste et la couleur est un gris fonction de la lumière. Au maximum, le bruit n'est pas présent et la couleur est la teinte, plus ou moins foncée en fonction de la lumière.

Le gros avantage de ce système, c'est bien sûr de permettre des dégradés de luminosité très aisés, puisqu'il suffit d'agir sur la composante du même nom.

De même, l'on peut rendre une couleur plus ou moins vive en jouant simplement avec la saturation [2].

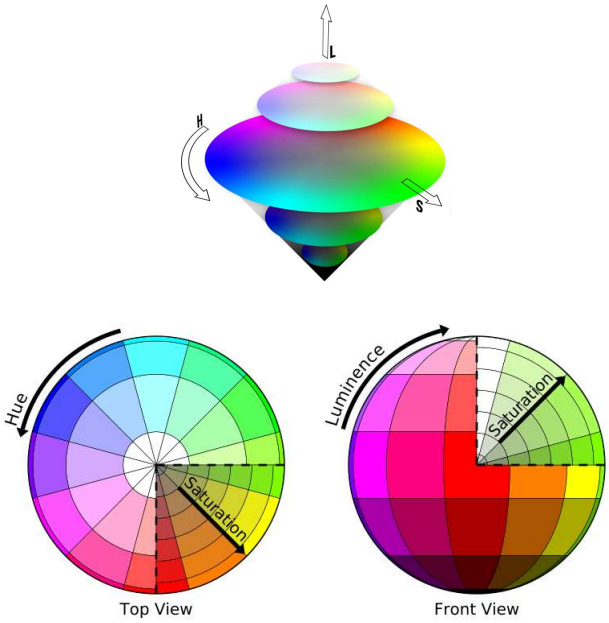


FIG. 4 – Principe du système HSL

3.6.5 La corrélation

La corrélation est effectuée entre une fenêtre contenant une partie de l'image de la canette et la totalité des images de la base de données. Notre fenêtre parcourt chaque image de la base de données dans toute sa longueur et dans toute sa largeur. Un seuil de corrélation est fixé : c'est le taux de ressemblance à un endroit donné du parcours de la fenêtre (issue de la photo prise par l'utilisateur) avec une autre fenêtre de même taille contenant une partie de l'image courante de la base de données.

Deux tests de corrélations sont faits. Le premier est sur la luminance : les valeurs de rouge, vert et bleu d'un pixel sont transformées en une valeur de luminance avec la formule suivante :

$$Y = 0{,}299\ R + 0{,}587\ G + 0{,}114\ B$$

Pour le premier test, la corrélation est donc rapide, il n'y a qu'une seule valeur à comparer pour chaque pixel. L'image courante est sélectionnée si la corrélation donne un

résultat supérieur au seuil fixé lors de son parcours.

Le deuxième est sur la couleur. Ceci nécessite d'effectuer des corrélations pour chaque filtre rouge, vert et bleu. Avec ce type de corrélation, nous sélectionnons l'image courante si pour un filtre donné (rouge, vert, ou bleu) la corrélation à un moment du parcours a été supérieure au seuil fixé.

3.7 Choix des ressources utilisées

3.7.1 Les canettes

Nous avons besoin d'un grand nombre de canettes différentes afin de vérifier la fiabilité de nos méthodes. Il est souhaitable d'avoir des marques de canettes étrangères afin d'avoir par exemple des canettes de marques dérivées ressemblant de très près aux marques originales. Nous avons ainsi pu collecter des canettes françaises et espagnoles. A ce jour, notre base de données contient vingt-cinq natures de canette différentes.

Comme expliqué précédemment, il est nécessaire d'avoir la totalité de l'image d'une nature de canette afin que lors de la prise de photo par l'utilisateur, la partie de l'image de la canette prise ait une chance d'être trouvée dans la base de données. Pour cette raison, nous avons 10 photos par nature de canettes, ce qui revient à dire que nous avons 250 images dans notre base de données.

4 Modélisation et conception

4.1 Explication globale du système

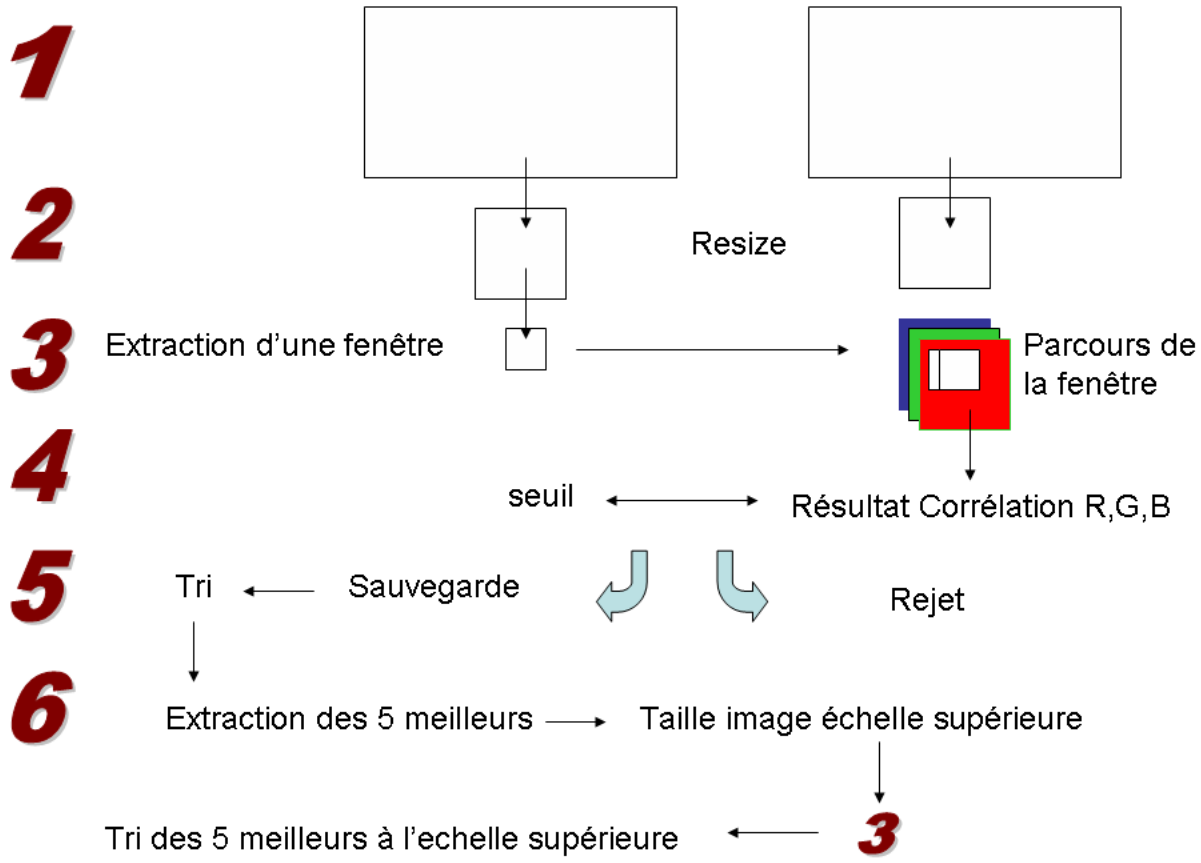


FIG. 5 – Modélisation globale du système

4.2 Informations sur les données

Nos données sont des images prises par un PDA. Elles ont à l'origine la résolution suivante : 640x480. Cette taille est la plus grande résolution possible sur nos PDA. Elles sont au format JPEG afin de tenir moins de place sur notre espace mémoire car nous avons beaucoup de données. Les photos prises sont en couleur.

4.3 Acteurs et cas d'utilisation

Dans le futur programme qui sera embarqué sur PDA, la seule action de l'utilisateur consiste à lancer le programme et appuyer sur une touche afin de prendre une photo de la canette. De ceci découle automatiquement les différents processus du programme permettant de déterminer la marque de la canette. L'administrateur gère la base de

données, il ajoute uniquement les nouvelles marques ou natures de canettes. S'il y a poursuite de ce projet, dans un cas qui ne sera pas une simulation, nous pourrons ajouter l'interaction avec la carte son qui permettra à l'utilisateur d'obtenir une réponse sonore.

A l'heure actuelle, notre simulateur n'est pas relié à la carte son. Nous cherchons en priorité à valider les méthodes qui permettront d'avoir un bon taux de reconnaissance.

4.4 Diagramme d'utilisation

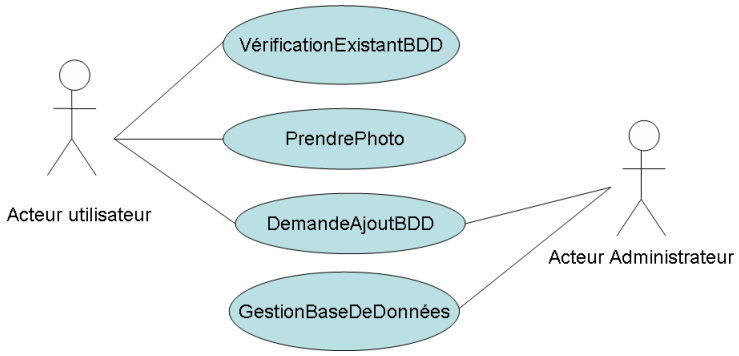


FIG. 6 – Diagramme d'utilisation

4.5 Fonctionnement détaillé de chaque module

Pour le moment, il n'y a pas d'interaction entre l'utilisateur et le simulateur. Ce sont les administrateurs qui gèrent la base de données ainsi que les différents paramètres du programme. Dans la future application portée sur PDA, le fonctionnement ressemblera à ceci :

- **Vérification de l'existant dans la base de données** (Utilisateur) : L'utilisateur a la possibilité de vérifier la liste des canettes disponibles dans la base de données. Ceci est utile afin que l'utilisateur ne demande pas une reconnaissance de canette qui n'existerait pas dans la base de données.

- **Demande d'ajouts dans la base de données** (Utilisateur) : L'utilisateur possède des canettes qui n'existent pas dans la base de données. Il souhaite alors que la base de données soit complétée. Il est nécessaire pour cette étape que l'utilisateur ait préalablement testé le programme afin de savoir ce qui existe ou non dans la base de données. Une lecture des différentes natures de canettes sera possible de façon audio lors du lancement du programme (voir UC précédente).

– **Prise de photo** (Utilisateur) : L'utilisateur lance le logiciel. Ce dernier lui indique d'appuyer sur une touche lorsqu'il pense pouvoir prendre une photo correcte et cadrée de la canette. C'est le premier événement ou processus déclenché par l'utilisateur. Le système répond en donnant la nature de la canette.

– **Gestion de la base de données : Reçoit demande d'ajout** (Administrateur) : L'administrateur ajoute les photos de canettes nécessaires à la demande de l'utilisateur, ainsi que les images redimensionnées de façon pyramidale des natures de canettes ajoutées.

4.6 Diagramme de séquence

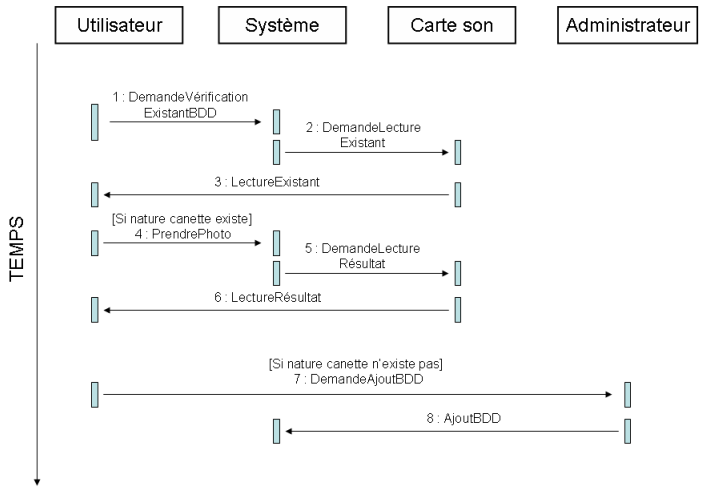


FIG. 7 – Diagramme de séquence

4.7 Implémentation

Pour la réalisation de ce programme, nous avons choisi d'utiliser Matlab. Ce logiciel inclut des outils et des fonctions qui nous sont nécessaires afin de réaliser nos algorithmes. Il nous a fallu du temps afin de nous adaptater au langage et au fonctionnement de ce logiciel. Cependant ce temps est relativement court en comparaison du gain de temps que ce logiciel nous apporte avec ses foncions déjà implémentées. Le seul défaut que nous pouvons faire à cet outil, c'est d'être un peu lent. Les temps de calculs constatés sont assez longs. Nous ne savons pas si les temps de calculs obtenus sont très significatifs par rapport aux méthodes employées.

MATLAB (raccourci de matrix laboratory, laboratoire matriciel) est un logiciel de calcul numérique édité par la société américaine The MathWorks qui est disponible sur plusieurs plateformes, voir le site web :

http ://www.mathworks.com/.

Il fait partie d'une suite de logiciels plus vaste appelée the MathWorks comprenant notamment Simulink, un environnement graphique permettant de concevoir et de simuler des systèmes qui varient dans le temps.

Matlab est un langage simple et très efficace. Il est optimisé pour le traitement des matrices, d'où son nom. Pour le calcul numérique, Matlab est beaucoup plus concis que des langages comme le C, le Pascal, le Fortran ou le Basic qui ne sont pas aussi récents. Par exemple, il n'est plus nécessaire de programmer des boucles pour modifier un à un les éléments d'une matrice. Il est possible de traiter une matrice comme une simple variable. Matlab contient également une interface graphique puissante ainsi qu'une grande variété d'algorithmes scientifiques.

Il est possible d'enrichir Matlab en ajoutant des boites à outils (toolbox) qui sont des ensembles de fonctions supplémentaires, profilées pour des applications particulières (traitements de signaux, analyses statistiques, optimisation, etc.).

5 Développement de chaque module

5.1 Création de la base de données pyramidale

A l'origine, nous possédons un nombre N de photos au format JPEG de chaque nature de canette. Nous appelons B le nombre total d'images de la base de données, le reste étant les T images de la base de tests.

Afin d'avoir un gain de temps lors du parcours des B images pour la corrélation, une nouvelle base de données est créée à partir des B images d'origine. Le but est de mettre les B images à une taille relativement petite afin que le parcours de chaque image soit le plus rapide possible. Néanmoins, il ne faut pas que cette taille soit trop petite, car si trop de détails sont supprimés, la corrélation peut être mauvaise. C'est lors de la phase de tests que nous changeons la taille des images si les résultats obtenus ne sont pas assez satisfaisants.

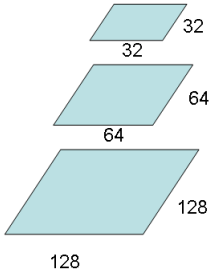


FIG. 8 – Système pyramidale

Cette réduction de taille est faite avec la fonction Matlab IMRESIZE qui redimensionne une image à la taille désirée. Cette méthode fait de la décimation. En traitement du signal, décimation est synonyme de sous-échantillonnage : on ne garde qu'un certain nombre d'échantillons par rapport au signal original (1 sur 2, 1 sur 10...).

Avec cette fonction, il suffit de fournir en argument le nom de l'image, la longueur et la largeur de la nouvelle matrice désirée puis le type d'algorithme souhaité pour réduire la matrice.

Nous utilisons une méthode dite "bicubic" pour redimensionner les images. Le filtre bicubic est une moyenne pondérée sur les seize pixels voisins, contrairement à la méthode bilinéaire qui ne fait cette moyenne que sur quatre voisins. Pour l'interpolation bicubique, les seize pixels les plus proches sont donc pris en compte, et les poids sont calculés selon la fonction suivante [7] :

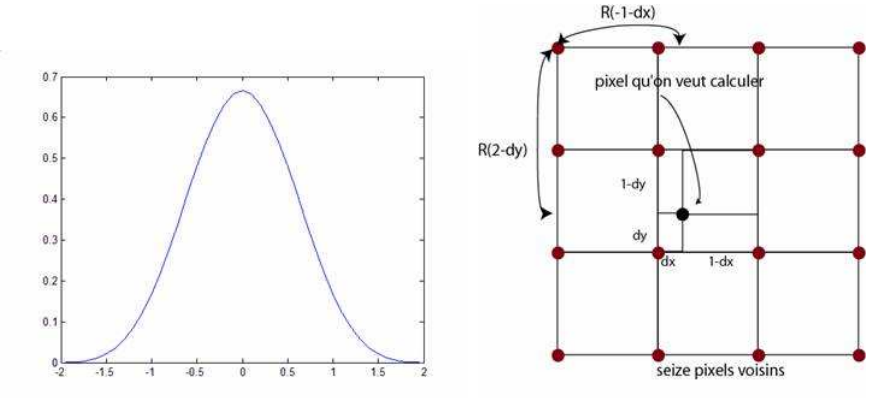


FIG. 9 – Représentation graphique du système bicubic

$$R(x) = \frac{1}{6}\left[P(x+2)^3 - 4\,P(x+1)^3 + 6\,P(x)^3 - 4\,P(x-1)^3 \right]$$

$$P(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$

5.2 Détection de la canette dans l'image : transformée de Hough

Comme expliqué précédemment, dans les B photos prises pour la base de données, ainsi que pour la photo prise par l'utilisateur, la seule chose qui nous intéresse est la partie de cette image où se trouve la canette.

Afin de créer une nouvelle image dans laquelle il n'y a que la canette, nous utilisons la transformée de Hough. Son principe est le suivant : il existe un nombre infini de lignes qui passent par un point, dont la seule différence est l'orientation (l'angle). Le but de la transformée est de déterminer lesquelles de ces lignes passent au plus près du schéma attendu. Afin de déterminer que deux points se trouvent sur une même ligne potentielle, on doit créer une représentation de la ligne qui permet une comparaison dans ce contexte.

Dans la transformée de Hough, chaque ligne est un vecteur de coordonnées paramétriques où ρ est la norme du vecteur et θ est l'angle. En transformant toutes les lignes possibles qui relient un point à un autre, c'est-à-dire en calculant la valeur de ρ pour chaque θ, on obtient une sinusoïde unique appelée espace de Hough. Si les courbes associées à deux points se coupent, l'endroit où elles se coupent dans l'espace de Hough correspond aux paramètres d'une droite qui relie ces deux points.

Notre but consiste en la détection des quatre lignes qui forment le rectangle encadrant

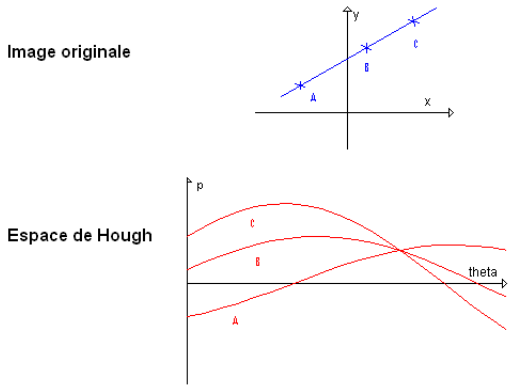


FIG. 10 – Transformée de Hough

la canette. Malheureusement, le nombre de lignes détectées est supérieur à quatre. En effet, il suffit qu'une petite partie dans l'image forme un segment relativement droit pour que la détection ajoute la ligne correspondante. Nous avons, bien sûr, la possibilité de régler le seuil de détection, mais le fait de baisser ce seuil peut dans certains cas éliminer les lignes essentielles.

Afin de pallier à ce problème, nous avons choisi de sélectionner les lignes en fonction de leur rapport entre elles. Nous cherchons les lignes ayant un écart d entre leur longueur et un écart D entre leur largeur. Nous faisons un premier constat : la taille de ces écarts n'est pas figée, elle varie selon plus ou moins ϵ correspondant à une légère homothétie. Nous cherchons donc plus précisément les lignes qui correspondent aux distances longueur et largeur d'une canette, à une homothétie près. Les canettes sont des cylindres droits. Cependant, nous constatons également sur les photos que les lignes formant leurs bords ne sont pas parallèles. Nous avons donc besoin d'une variable θ afin que l'angle des droites que nous cherchons puisse varier de quelques degrés. Ainsi, même les droites qui ne sont pas strictement horizontales (ou verticales) sont détectées.

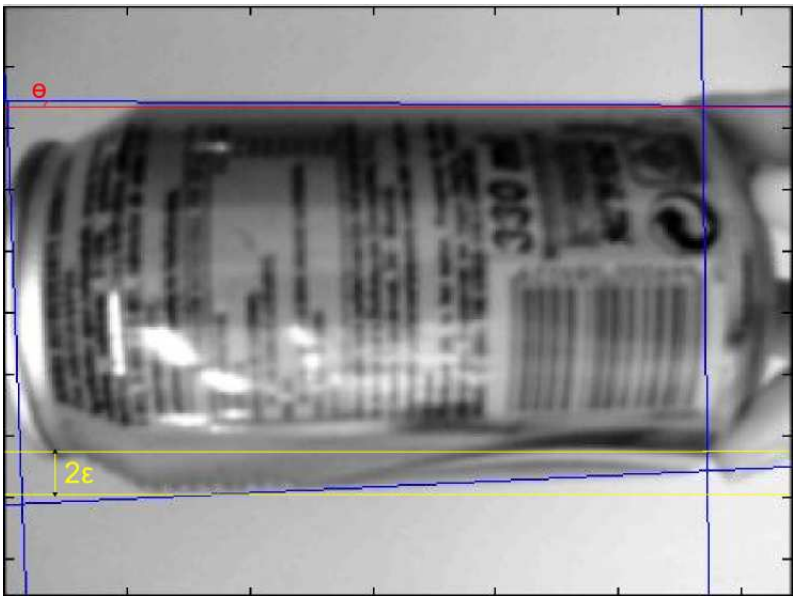


FIG. 11 – Problèmes rencontrés : Epsilon et Theta

Cette méthode ne nous a pas permis d'éliminer toutes les lignes inutiles. En effet trop de lignes sont détectées, souvent à cause des lettres en gros caractères ou encore à cause du code barre.

FIG. 12 – Une bonne et une mauvaise détection par la transformée de Hough

Après cette étape vient celle de la corrélation. Nous ne corrélons pas l'image entière de la canette avec les images de la base de données, car il est peu probable d'avoir exactement la même face. Pour cette raison, il est nécessaire de sélectionner une partie de la canette. C'est cette fenêtre que nous corrélons avec les photos de la base de données.

5.3 Le plan de secours : Création d'une bande "fenêtre" pour la corrélation

Dans le cas où notre détection par la transformée de Hough ne serait pas au point, nous avons développé un système rapide et simple nous faisant gagner une étape grâce aux restrictions que nous nous sommes fixées.

Il s'agit de sélectionner directement une fenêtre centrée dans l'image qui est, rappelons le, cadrée. Cette méthode permet d'obtenir directement une partie de l'image de la canette. Cette partie est extraite de la façon suivante :

```
longueurBande = cast(taille/3,'uint8');
largeurBande = cast(taille/3,'uint8');

centreL = cast(taille/2,'uint8');
centreC = cast(taille/2,'uint8');

moitieBandeL = cast(longueurBande/2,'uint8');
moitieBandeC = cast(largeurBande/2,'uint8');

%nouvelle matrice qui va contenir la partie de la canette selectionnée
%cette matrice va correspondre en taille a notre fenêtre comparative

photo = imresize(photo,[taille taille],'bicubic');

resizeRouge = photo(centreL-moitieBandeL+1 :
centreL+moitieBandeL, centreC-moitieBandeC+1 : centreC+moitieBandeC);
resizeVert = photo(centreL-moitieBandeL+1 :
centreL+moitieBandeL, taille+centreC-moitieBandeC+1 :
taille+centreC+moitieBandeC);
resizeBleu = photo(centreL-moitieBandeL+1 :
centreL+moitieBandeL, 2*taille+centreC-moitieBandeC+1 :
2*taille+centreC+moitieBandeC);
```

Les variables longueurBande et largeurBande représentent les tailles de longueur et largeur de la fenêtre qui va parcourir les images de la base de données. Elles valent 1/3 de la longueur et de la largeur des images mises en 32x32.

CentreL et CentreC nous servent à repérer le centre de l'image pour que l'extraction de la fenêtre se fasse au centre de l'image prise par l'utilisateur.

Avant de faire l'extraction, la photo de l'utilisateur est mise en 32x32. Cette valeur est représentée par la variable taille. L'image originale étant en couleurs, il nous faut faire cette extraction de fenêtre sur les filtres R, G et B de la photo.

Nous rappelons que cette méthode ne permet pas les homothéties. Par exemple, si la photo de la canette est prise de trop loin, en prenant une fenêtre avec les dimensions précédentes, nous aurons des objets dans la fenêtre (ici le mur) qui ne nous intéressent pas pour la corrélation.

5.4 La corrélation

En sciences, on parle de corrélation quand, pour quantifier deux mesures, on doit faire appel à une loi où l'un des facteurs dépend de l'autre. Cela signifie que les deux facteurs varient simultanément, si un seul varie au départ. Corrélation n'est pas synonyme de causalité au sens strict, car la relation peut également trouver son origine dans une cause commune, ou dans une corrélation fortuite. La relation de proportionnalité est une relation de corrélation. Il ne faut pas non plus confondre corrélation et Dépendance (mathématiques) : deux variables aléatoires indépendantes ont une corrélation nulle, mais deux variables ayant une corrélation nulle ne sont pas forcément indépendantes [5].

La corrélation linéaire (ou corrélation de Pearson) est définie pour des couples de variables aléatoires de variances finies par :

$$r(X,Y) = \frac{cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}}$$

soit encore :

$$r(X,Y) = \frac{\mathbb{E}((X-\mathbb{E}(X))(Y-\mathbb{E}(Y)))}{\sqrt{\mathbb{E}((X-\mathbb{E}(X))^2)}\sqrt{\mathbb{E}((Y-\mathbb{E}(Y))^2)}}$$

De cette dernière expression, on déduit la version empirique pour un échantillon :

$$(X_1,Y_1),...,(X_n,Y_n)$$

indépendant et identiquement distribué, à savoir :

$$\hat{r}_n(X,Y) = \frac{\sum(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum(X_i - \overline{X})^2}\sqrt{\sum(Y_i - \overline{Y})^2}}$$

où $\bar{X}$ et $\bar{Y}$ désignent les moyennes empiriques, c'est à dire :

$$\overline{X} = \frac{1}{n}\sum X_i$$

et respectivement pour $\bar{Y}$.

La corrélation est testée entre la fenêtre choisie de la partie canette de l'image d'origine réduite et chaque image de canette de la base de données. Autrement dit, notre fenêtre de longueur L et de largeur l parcourt toute la longueur et toute la largeur de chaque image de la base de données, afin d'observer si à un moment donné, la corrélation est supérieure à notre seuil. Nous garderons le nom des X images dont la corrélation est supérieure à 70%.

Nous avons utilisé la fonction CORR2 de Matlab utilisant la formule décrite précédemment afin d'effectuer nos corrélations.

Nous implémentons également deux méthodes pour lesquelles la corrélation porte sur deux différents types de données. La première corrélation est effectuée entre des fenêtres dont le contenu est constitué de valeurs de luminance. La deuxième corrélation porte sur les couleurs RGB de l'image. Chaque filtre R, G et B est comparé à son filtre respectif sur l'autre fenêtre.

Pour la première méthode, nous sélectionnons l'image courante de la base de données si à un moment donné du parcours de l'image le seuil est supérieur au seuil fixé. Pour la deuxième méthode, nous gardons également l'image courante par le même procédé, à la différence que ceci est fait pour chaque filtre R, G et B. Il y a donc trois corrélations au lieu d'une pour la première méthode. Le temps de calcul est donc trois fois plus long. Pour ces deux méthodes, nous gardons le taux de corrélations maximum obtenu afin de pouvoir classer nos résultats par la suite.

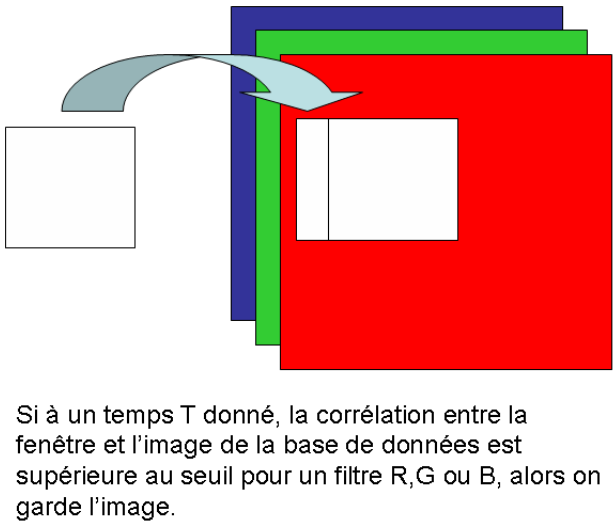


FIG. 13 – Schéma explicatif de la corrélation couleur

5.5 Tests unitaires

Le simulateur n'étant pas destiné à être utilisé par des personnes non expérimentées en informatique, les tests unitaires ont été effectués sur la fonctionnalité de chaque processus.

Chaque fonction a été testée séparément du programme principal, puis intégrée par la suite. Des tests unitaires sur la bonne intégration des diverses fonctions dans le programme principal ont également été effectués.

Les fonctions ont été développées de façon à être les plus génériques possible. Ainsi, s'il s'avère que de nouvelles photos rajoutées dans la base de données ne sont pas de la même taille que les photos actuelles, cela ne posera aucun problème à notre programme.

6 Bilan

6.1 Bilan des choix des techniques

Les techniques utilisées ont été choisies lors de la phase d'analyse. Nous voyons dans cette sous-partie pourquoi nous avons choisi ces techniques.

La première étape de notre démarche consiste à extraire la canette du reste de l'image. Nous avons utilisé la transformée de Hough. Cette méthode nous permet de détecter les lignes dans une image. Une fois la canette prise en photo et cadrée, sa représentation 2D ressemble fortement à un rectangle. Un rectangle étant composé de quatre segments parallèles deux à deux, ils sont relativement simples à détecter. Cette technique nous semble théoriquement la meilleure pour détecter les contours de nos canettes. Les problèmes que nous rencontrons parfois lors de la détection sont dus en partie aux reflets qui sont très clairs et au fond qui est un mur blanc.

La démarche suivante consiste à redimensionner les images. Le redimensionnement bilinéaire est une moyenne pondérée des quatre pixels sources voisins. Elle permet de générer (agrandissement) ou de supprimer (réduction) des pixels. La méthode bilinéaire est rapide mais donne un résultat de moins bonne qualité que le filtre bicubique. Les pondérations correspondent à la distance entre la position des pixels de la nouvelle image par rapport à la position des pixels de l'ancienne image. Le redimensionnement par la méthode bicubique reprend le même principe mais elle prend en considération ses 16 voisins. Cette approche est plus performante mais plus longue. Ces deux méthodes sont meilleures que la troisième méthode disponible sous Matlab : "nearest". Le redimensionnement par plus proche voisin prend la valeur du pixel qui est le plus proche (spatialement). Nous pouvons donc conclure de ce qui précède que le filtre choisi est bien le meilleur.

6.2 Bilan de la planification

La planification de la production de notre simulateur n'a pas été évidente. Certains points planifiés ont dû être revus à plusieurs reprises. C'est le cas notamment pour les méthodes de la transformée de Hough ou encore celle de la luminance.

Dans certains cas nous avons été obligés de faire du rétro planning. Nous avons parfois changé de méthode car nous avons rencontré des problèmes avec celle choisie mais aussi parfois parce que la méthode choisie de donnait pas les résultats escomptés.

Il nous semblait plus important d'obtenir des résultats à la fin du temps imparti, plutôt que de développer des méthodes complexes que nous n'aurions peut être pas pu tester.

Malgré quelques contre-temps et grâce à un gain de temps sur certaines méthodes dont nous avions estimé l'implémentation trop longue, nous somme parvenus à respecter les délais imposés tout en obtenant des résultats qui nous semblent satisfaisants.

6.3 Réalisation des objectifs

L'objectif était de réaliser un simulateur permettant de reconnaître la marque d'une canette à partir d'une photo. Cet objectif a été atteint. Nous avons même obtenu un taux de réussite de reconnaissance supérieur à celui que nous nous étions fixé.

Nous nous étions fixé des sous-objectifs à l'intérieur de celui du simulateur. Ceux-ci consistaient majoritairement à respecter les méthodes que nous nous étions imposées. Nous ne sommes pas parvenus à garder dans notre projet la méthode de la transformée de Hough car il subsistait trop de lignes détectées. Cependant, nous avons implémenté plus de méthodes que prévu pour la corrélation, ce qui nous a permis de faire des études comparatives entre les méthodes implémentées.

7 Perspectives

Nous pensons que les perspectives pouvant être réalisées pour ce projet dans le futur sont nombreuses.

Tout d'abord, il serait intéressant de réussir à terminer la fonction d'extraction de la canette par la transformée de Hough. Celle-ci permettrait de sélectionner la partie de l'image où se trouve la canette. Contrairement à la méthode que nous avons implémenté qui tenait compte de nos restrictions, la nouvelle méthode ne nécessiterait pas que la canette soit cadrée au centre, ni même qu'elle soit maintenue de façon relativement droite devant l'objectif.

Nous pensons qu'il serait intéressant de programmer un système d'apprentissage pour les seuils, de manière à ce que ces derniers soient déterminés automatiquement par le système. Cet apprentissage serait fait, non pas à chaque lancement du programme, mais à chaque fois que la base de données serait changée.

Il est également nécessaire d'agrandir au maximum la base de données afin d'avoir un panel de canettes très diversifié. Plus la base de données est grande, plus le temps de calcul augmente. Pour cette raison, la diminution de nombre de vues par canettes serait sans doute la bienvenue, à condition que ceci ne fasse pas chuter le taux de reconnaissance.

La meilleure perspective que nous puissions espérer, est que tous les types de canettes soient reconnus par une méthode ou par une autre. Dans cette optique, si l'une des méthodes ne fonctionne pas avec un type de canettes, l'autre méthode est lancée pour que la canette soit détectée. Les problèmes de détections étant majoritairement dus aux problèmes de reflets, nous pourrions essayer de régler le problème en prenant les photos avec un éclairage diffus. Cependant, cette approche ne reflèterait pas des conditions réelles.

7.1 Modifications selon les résultats obtenus

Nous avons à plusieurs reprises dû modifier nos méthodes et les divers paramètres de nos fonctions. Au vu des résultats de certaines méthodes, nous avons préféré en implémenter d'autres. C'est par exemple le cas pour la méthode RGB que nous avons rajoutée. Elle est certes plus lente, mais donne de meilleurs résultats. Nous avons également modifié une fois le choix de l'algorithme car nous ne parvenions pas à atteindre notre but avec une certaine méthode. Pour cette raison nous avons créé un plan de rechange qui nous fait gagner une étape. Ce plan fonctionne uniquement parce que nos photos sont cadrées.

La phase de tests consistant à enlever les bugs et à affiner nos résultats, nous avons dû

ajuster les valeurs de nos paramètres afin que notre algorithme donne les meilleurs résultats possibles avec les méthodes utilisées. Ces paramètres sont réglés manuellement. Faute de temps nous n'avons pas, pour le moment, implémenté de système d'apprentissage.

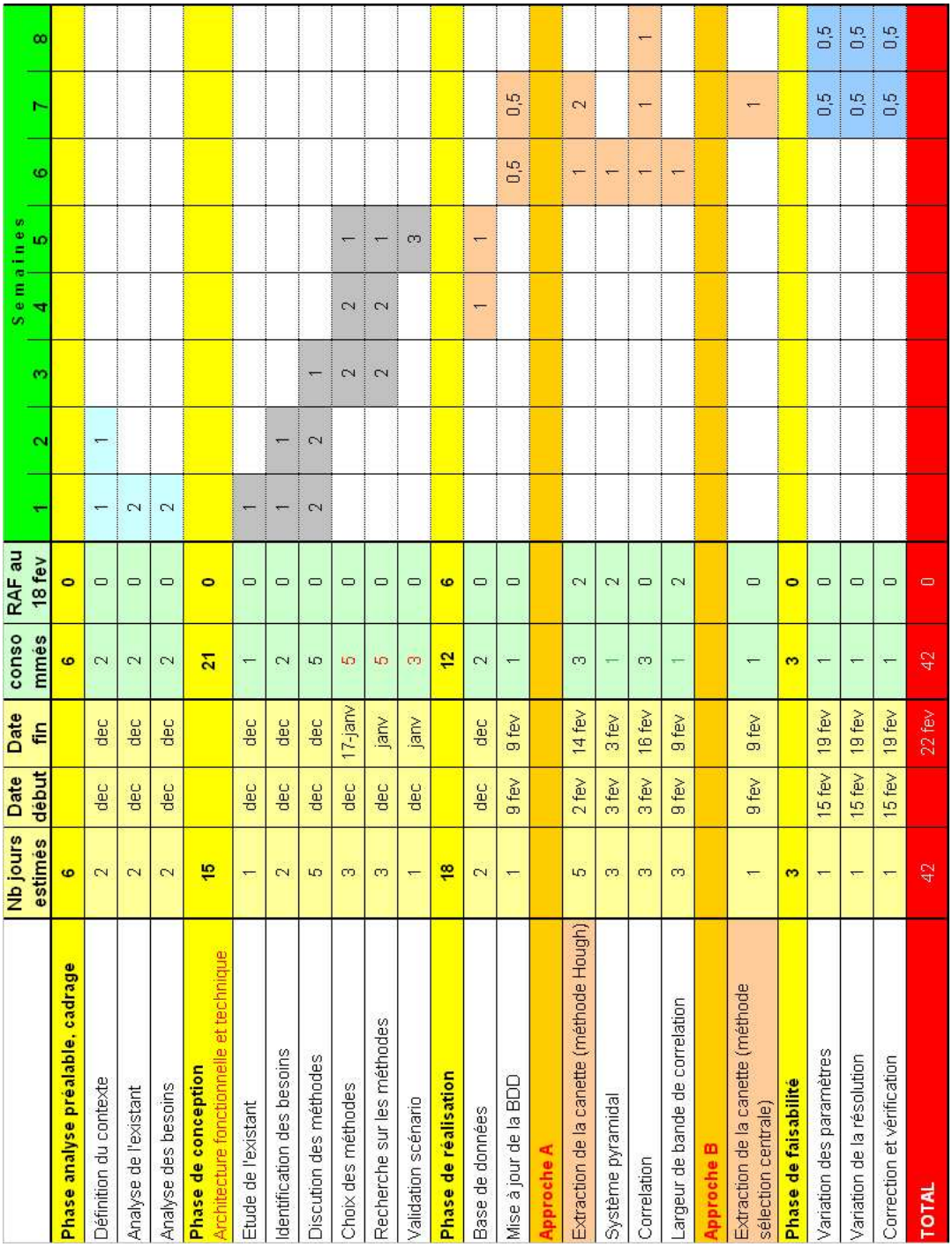
7.2 Evolution et poursuite du projet

D'après toutes les évolutions possibles que nous avons vues dans les perspectives, la problématique de ce sujet est loin d'être totalement résolue. En effet, certaines méthodes ont été implémentées mais pas incluses dans le programme. Il nous semble important de les terminer afin de pouvoir observer les résultats qui en découlent. D'autres méthodes n'ont pu être développées faute de temps, il serait néanmoins très intéressant de les implémenter.

Un des buts importants et qui nous semble urgent est d'embarquer notre simulateur sur un PDA. Afin d'y parvenir, il faut recoder tous les algorithmes dans un autre langage. En codant ce logiciel en C#, il serait aisé de faire une interface simple d'utilisation. Le reste du code peut être codé en C++, ce qui serait plus simple pour créer les outils de traitement des images.

8 Annexe

8.1 Organisation de l'étude



	Nb jours estimés	Date début	Date fin	conso mmés	RAF au 18 fev	Semaines 1	2	3	4	5	6	7	8
Phase analyse préalable, cadrage	6			6	0								
Définition du contexte	2	dec	dec	2	0	1	1						
Analyse de l'existant	2	dec	dec	2	0	2							
Analyse des besoins	2	dec	dec	2	0	2							
Phase de conception Architecture fonctionnelle et technique	15			21	0								
Etude de l'existant	1	dec	dec	1	0	1							
Identification des besoins	2	dec	dec	2	0	1	1						
Discution des méthodes	5	dec	dec	5	0	2	2	1					
Choix des méthodes	3	dec	17-janv	5	0			2	2	1			
Recherche sur les méthodes	3	dec	janv	5	0			2	2	1			
Validation scénario	1	dec	janv	3	0					3			
Phase de réalisation	18			12	6								
Base de données	2	dec	dec	2	0				1	1			
Mise à jour de la BDD	1	9 fev	9 fev	1	0						0,5	0,5	
Approche A													
Extraction de la canette (méthode Hough)	5	2 fev	14 fev	3	2						1	2	
Système pyramidal	3	3 fev	3 fev	1	2						1		
Correlation	3	3 fev	16 fev	3	0						1	1	1
Largeur de bande de correlation	3	9 fev	9 fev	1	2						1		
Approche B													
Extraction de la canette (méthode sélection centrale)	1	9 fev	9 fev	1	0							1	
Phase de faisabilité	3			3	0								
Variation des paramètres	1	15 fev	19 fev	1	0							0,5	0,5
Variation de la résolution	1	15 fev	19 fev	1	0							0,5	0,5
Correction et vérification	1	15 fev	19 fev	1	0							0,5	0,5
TOTAL	42		22 fev	42	0								

FIG. 14 – Planning

8.2 La base de données en images

FIG. 15 – Exemples de photos des marques de canettes de la base de données

8.3 Code Matlab du programme principal

```
% PROGRAMME PRINCIPAL
%Création d'une fenetre carré coulissante d'1/3 de taille.
%Corrélation de cette fenetre avec les images taille x taille de la BDD
%Ei corrélation > taux => stocker
%Fin des corrélations de taille taille
%Elections des 5 meilleurs et corrélation avec la taille double pour
%paufiner les résultats.
%Fin de programme

% creation de la base de données (a retirer si deja fait)
%creationBDD(taille); % la base de donnée de toutes les images a la meme
%taille est cree

clc;
tic;

%Configuration du fichier Canettes3.m
taille = 32; %taille du redimensionnement

taux = 0.50; %taux de correlation a avoir au minimum

%methode permet de choisir la methode de correlation
%1 = luminance, 11 = max(RGB), 111 = 1 - max(RGB) - min(RGB),
%1111 = min(RGB)
methode = 11;

%photo d'entrée

image_to_compare= 'sprite_05.jpg';
%image_to_compare= 'mahou_01.jpg';


%FIN CONFIGURATION

methode
image_to_compare
```

```
%lecture de la photo venant d etre prise par l utilisateur
Orig = imread(strcat('D:\maj_190207\',image_to_compare));

figure(10); imagesc(Orig);

%creation de la fenetre coulissante en rouge vert bleu
[OrigResizeRouge,OrigResizeVert,OrigResizeBleu]=creationFenetre(Orig,taille);

%transformation de chiffre en chaine de caractère
val = num2str(taille);

%creation tableau luminance de la fenetre coulissante
if methode == 1
    tab_h = 0.299 * OrigResizeRouge + 0.587 * OrigResizeVert + 0.114 * OrigResizeBleu
end

%listing des fichiers du répertoire des miniatures de taille taille
DirectoryName = 'D:\maj_190207\nbdd\';
NewDirectoryName = strcat(DirectoryName,val,'\');
listeFichiers = dir(NewDirectoryName);
nbFichiers = size(listeFichiers);

%correlation du listing avec la fenetre
indice = 1; % indice de la structure
for i = 1 : nbFichiers
    if listeFichiers(i).isdir == 0 % si c est un fichier et non un repertoire
        fichierCourant=listeFichiers(i).name;
        ImageName = strcat(NewDirectoryName,fichierCourant);
        ImageName2 = imread(ImageName);
ImageNameRouge=ImageName2(:,1:taille);
ImageNameVert=ImageName2(:,taille+1:2*taille);
ImageNameBleu=ImageName2(:,2*taille+1:3*taille);

        %tableau luminance image BDD
        if methode == 1
            tab_h_imageBDD = 0.299 * ImageNameRouge + 0.587 * ImageNameVert + 0.114 *
ImageNameBleu;
            valDeCorr = correlation2(tab_h,tab_h_imageBDD,taux);
        elseif methode == 11
```

```
            valDeCorr = correlation(OrigResizeRouge, OrigResizeVert, OrigResizeBleu,
ImageNameRouge, ImageNameVert, ImageNameBleu, taux);
        elseif methode == 111
            valDeCorr = correlation3(OrigResizeRouge, OrigResizeVert, OrigResizeBleu,
ImageNameRouge, ImageNameVert, ImageNameBleu, taux);
        elseif methode == 1111
            valDeCorr = correlation4(OrigResizeRouge, OrigResizeVert, OrigResizeBleu,
ImageNameRouge, ImageNameVert, ImageNameBleu, taux);
        end

        if valDeCorr >= taux
           ressemble(indice).nom = fichierCourant;
           ressemble(indice).correl = valDeCorr;
           indice = indice + 1;
        end
    end
end

%récupération des 5 meilleures corrélations classés
if indice ~= 1
    i = 1;
    taille
    while length(ressemble) ~= 0
        max = -1;
        maxk = -1;

        for k=1:length(ressemble)
            if ressemble(k).correl > max
               max = ressemble(k).correl;
               maxk = k;
            end
        end

        if( i <= 5)
            ressemblePlus(i).nom = ressemble(maxk).nom;
            ressemblePlus(i).correl = ressemble(maxk).correl;
            strcat(num2str(i),'_',
ressemblePlus(i).nom,'_',num2str(ressemblePlus(i).correl))
            %ressemblePlus(i).nom
```

```
            %ressemblePlus(i).correl
        end
        if i==5
            break;
        end
        i = i+1;
        ressemble(maxk)=[]; % on detruit la maxk-ieme structure
    end
end

% Dans ressemblePlus, on a maintenant dans l'ordre du plus ressemblant au
% moins ressemblant le nom des plus forte ressemblances de canettes
% on refait un test sur les 5 premiers avec la resolution superieure

% on double taille
taille=2*taille;

%on crée la nouvelle fenetre
[OrigResizeRouge,OrigResizeVert,OrigResizeBleu]=creationFenetre(Orig,taille);

%creation tableau luminance
if methode == 1
    tab_h = 0.299 * OrigResizeRouge + 0.587 * OrigResizeVert + 0.114 * OrigResizeBleu
end

%on change en string
val=num2str(taille);

% on va parcourir le nouveau dossier de la bonne taille
NewDirectoryName = strcat(DirectoryName,val,'\');
%récupération du nombre de canettes élu <= 5
nbFichiers = length(ressemblePlus);

indice = 1; % indice de la structure
for i = 1 : nbFichiers
        fichierCourant=ressemblePlus(i).nom;
        ImageName = strcat(NewDirectoryName,fichierCourant);
        ImageName2 = imread(ImageName);
ImageNameRouge=ImageName2(:,1:taille);
```

```
ImageNameVert=ImageName2(:,taille+1:2*taille);
ImageNameBleu=ImageName2(:,2*taille+1:3*taille);

        %tableau luminance image BDD
        if methode==1
            tab_h_imageBDD = 0.299 * ImageNameRouge + 0.587 * ImageNameVert + 0.114 *
ImageNameBleu;
            valDeCorr = correlation2(tab_h,tab_h_imageBDD,taux);
        elseif methode == 11
            valDeCorr = correlation(OrigResizeRouge, OrigResizeVert, OrigResizeBleu,
ImageNameRouge, ImageNameVert, ImageNameBleu, taux);
        elseif methode == 111
            valDeCorr = correlation3(OrigResizeRouge, OrigResizeVert, OrigResizeBleu,
ImageNameRouge, ImageNameVert, ImageNameBleu, taux);
        elseif methode == 1111
            valDeCorr = correlation4(OrigResizeRouge, OrigResizeVert, OrigResizeBleu,
ImageNameRouge, ImageNameVert, ImageNameBleu, taux);
        end

        if valDeCorr >= taux
           ressemble2(indice).nom = fichierCourant;
           ressemble2(indice).correl = valDeCorr;
           indice = indice + 1;
        end
end

if indice ~= 1
    taille
    while length(ressemble2) ~=0
        max = -1;
        maxk =-1;
        for k=1:length(ressemble2)
            if ressemble2(k).correl > max
               max = ressemble2(k).correl;
               maxk = k;
            end
        end
        strcat(ressemble2(maxk).nom,'_',num2str(ressemble2(maxk).correl))
        figure(k); imagesc(imread(strcat(DirectoryName,ressemble2(maxk).nom)));
```

```
        %ressemble2(maxk).nom
        %ressemble2(maxk).correl
        ressemble2(maxk)=[];
    end
end
toc;
```

Références

[1] http ://cat.inist.fr/ ?aModele=afficheN&cpsidt=156586.

[2] http ://en.wikipedia.org/wiki/HSL_color_space.

[3] http ://fr.wikibooks.org/wiki/Infographie/Notions_énérales/Couleur_en_informatique.

[4] http ://fr.wikipedia.org/wiki/.

[5] http ://fr.wikipedia.org/wiki/Corrélation.

[6] http ://www prima.imag.fr/jlc/Courses/2002/ENSI2.RNRF/ENSI2.RF.S6.pdf.

[7] http ://www710.univ-lyon1.fr/ hbriceno/teaching/istg/si05/TD1.htm.

[8] http ://www.antsearch.univ tours.fr/rfia2006/.

[9] http ://www.iapr.org/.

[10] http ://www.iapr.org/members/newsletter/Newsletter07 01/index.htm.