



télécom
saint-étienne

école d'ingénieurs
nouvelles technologies

Rapport Projet Vision

OCR sur pneu

Étudiants :

Hugo CONSTANT
Hélène EHRHARDT
Thomas PEROTTO
Justin THOMAS

Responsable Projet VISION :

Yves BRINGER

Dates du projet :

Novembre 2019 – Février 2020

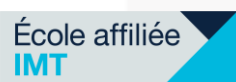


Table des matières

Table des matières	2
Introduction	4
1. Cahier des charges	5
1.1. Spécifications techniques.....	5
1.2. Calendrier	5
2. Développement de l'application.....	6
2.1. Architecture.....	6
2.2. Développement de l'IHM	7
a) Présentation générale.....	7
b) Protocole d'utilisation	8
c) Analyse de l'application.....	10
2.3. Traitement des images	10
a) Images originales.....	Erreur ! Signet non défini.
b) Images prétraitées par l'enseignant	Erreur ! Signet non défini.
2.4. OCR.....	11
a) Théorie.....	14
b) Mise en place.....	16
c) Résultats	18
d) Intégration à l'IHM	19
3. Conclusion	20
3.1. Conclusion technique	20
3.2. Conclusion personnelle	21
Justin Thomas	21
Hugo Constant.....	21
Lexique	22

Introduction

Avant-propos

Ce rapport est issu du travail réalisé en projet Vision, au cours de la formation Image, Photonique et Smart-Industries à Télécom Saint-Etienne.

Le module se mène en groupe de 4 à 5 personnes sur une période d'environ 3 mois. Les objectifs du projet Vision, en dehors de mener le projet à son terme, est de délivrer les différents rendus dans les temps, développer nos compétences en gestion de projets, argumenter en prise de décision de choix techniques, et communiquer à l'oral comme écrite sur l'évolution du projet. L'enseignant en charge du module attend de chaque projet une présentation finale sur l'avancée du projet, une démonstration de l'application développée, un rapport et optionnellement, toute documentation supplémentaire qu'un groupe jugerait intéressant d'ajouter.

Sujet

Nous avons pour objectif de développer une application Vision de reconnaissance de caractères pour pneus de voitures. L'application doit être en mesure de valider la possibilité de mettre en vente un pneu analysé, en fonction de critères d'admissibilité fournis par l'utilisateur (exemple : DOT, permettant entré-autre de lire la date de fabrication du pneu).

Organisation

Notre équipe est composé d'une cheffe de projet Hélène ERHADRT, d'un développeur en traitement d'images Justin THOMAS, d'un développeur en architecture informatique Thomas PEROTTO et pour finir d'un développeur en méthodes d'OCR Hugo CONSTANT.

A la demande de l'enseignant responsable du module, les membres de l'équipe possèdent des niveaux en informatique hétérogènes. Thomas PEROTTO étant le collaborateur le plus à l'aise dans le domaine du traitement d'images, il suivi Justin THOMAS tout au long du projet afin de l'aider à développer ses connaissances et compétences dans ce domaine. De son côté, Hugo CONSTANT suivi Hélène ERHADRT dans le but de l'aider dans le design de l'IHM.

Plan détaillé

Ce rapport est décomposé afin de faire ressortir chaque partie de la réalisation de notre projet. Un rappel des spécifications techniques ainsi que du calendrier de travail est tout d'abord présenté. Celui-ci est suivi par la présentation de l'application finale. La partie suivante détaille la réalisation des différentes parties techniques. Pour finir, une conclusion globale est donnée afin d'apporter un recul sur le projet et sa réalisation.

1. Cahier des charges

1.1. Spécifications techniques

Développement d'une interface Homme-machine
<p>Sélection du répertoire de travail</p> <p>Ouverture de chaque image l'une après l'autre</p> <p>Boutons permettant de faire défiler les images manuellement</p> <p>Affichage dans différentes cellules des caractéristiques :</p> <ul style="list-style-type: none"> Affichage du N° de pneu (idéal : DOT) Affichage de la validité du pneu (PASS/FAILED)
Développement d'une DLL pour traiter les images
<p>Mise en valeur des caractères. Plusieurs possibilités :</p> <ul style="list-style-type: none"> Expansion dynamique sur pixels blanc Détection gradients (ex : Prewitt) <p>Détection de chaque caractère séquentiellement</p> <ul style="list-style-type: none"> Chaque caractère considéré comme un objet Histogramme local <p>Comparaison de l'histogramme de l'objet avec la base de donnée</p> <ul style="list-style-type: none"> Comprend chaque lettre et chaque caractère Associe chaque objet avec un caractère. Plusieurs possibilités : <ul style="list-style-type: none"> Similarité histogramme Classement par arbre de reconnaissance Retourne la lettre associée
Développement d'un Wrapper pour passer de la DLL à l'IHM
<p>Passer le MINIMUM de fonctions par le wrapper</p> <ul style="list-style-type: none"> Initialisation Analyse image Verdict Dé-initialisation (optionnel si absence d'allocation dynamique)

1.2. Calendrier

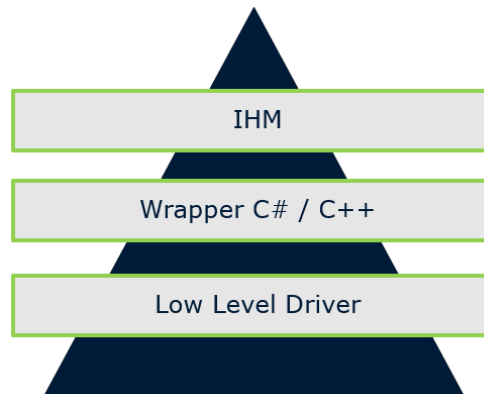
Jalons	Objectifs
28/11/2019	Spécifications projet
03/12/2019	Initialisation de l'IHM Initialisation de la DLL permettant de traiter les images envoyées par l'IHM
14/01/2020	Extraction du numéro de série Ajout du seuil Seuillage et étiquetage Arbre de reconnaissance
28/01/2020	Intégration du Wrapper Optimisation du masquage Conclusion et analyse des résultats
06/02/2020	Présentation du projet

2. Développement de l'application

2.1. Architecture

L'architecture logicielle a été réalisée dans le but de faciliter les différents axes de développement. En amont du projet, trois niveaux ont été identifiés (IHM C#, Wrapper C#/C++, Traitement + OCR). Chaque niveau est encapsulé, de telle sorte à réaliser uniquement les opérations désirées dans leur couche respective.

L'IHM C# contient les fonctions de chargement et défilement d'images, mais également le verdict du pneu. Le verdict est basé sur une chaîne de caractère (contenant le DOT par exemple), provenant du niveau « Traitement + OCR ». Un Wrapper C#/C++ permet d'accéder à celui-ci. Dans le programme, ce plus bas niveau est appelé LLD (*Low Level Driver*). La figure ci-dessous permet d'illustrer l'architecture mise en place.



Afin de limiter les dérives de développement depuis l'IHM, seul deux fonctions sont accessibles via le Wrapper :

- Une fonction d'initialisation, permettant de réinitialiser l'ensemble des variables du LLD,
- Et une fonction d'extraction du DOT, retournant une chaîne de caractère contenant le DOT.

Il n'est pas nécessaire d'ajouter une fonction de dés-allocation, étant donné qu'aucune allocation de mémoire dynamique est réalisée dans le LLD.

L'appel de la fonction permettant l'extraction du texte (DOT dans notre exemple) entraîne automatiquement un traitement d'image, et l'algorithme d'OCR. Chaque fabricant de pneu étant différent, les paramètres du traitement d'images doivent être modifiés d'une image à l'autre (exemple : taille des filtres, seuils, ...). Afin de faire varier ces paramètres internes au LLD, un fichier de *Presets* nécessite d'être chargé depuis l'IHM. Différents *Presets* sont disponibles, et permettent d'adapter le traitement en fonction de l'image. Le tableau ci-dessous en décrit quelques-uns. Une description complète est fournie dans la documentation.

Preset	Indication
Demo	1=Activation mode démo
DOT_size_element_1	Taille 1 ^{er} élément du DOT
DOT_size_element_2	Taille 2 nd élément du DOT
DOT_size_element_3	Taille 3 ^{eme} élément du DOT

Preset	Indication
ROI_i	Position i du centre de la ROI
ROI_J	Position j du centre de la ROI
ROI_w	Largeur de la ROI
ROI_h	Hauteur de la ROI

Note : Le logiciel est livré avec deux fichiers « *prese demo.txt* » et « *preset_pretraitées.txt* », fonctionnant avec des images de démonstration.

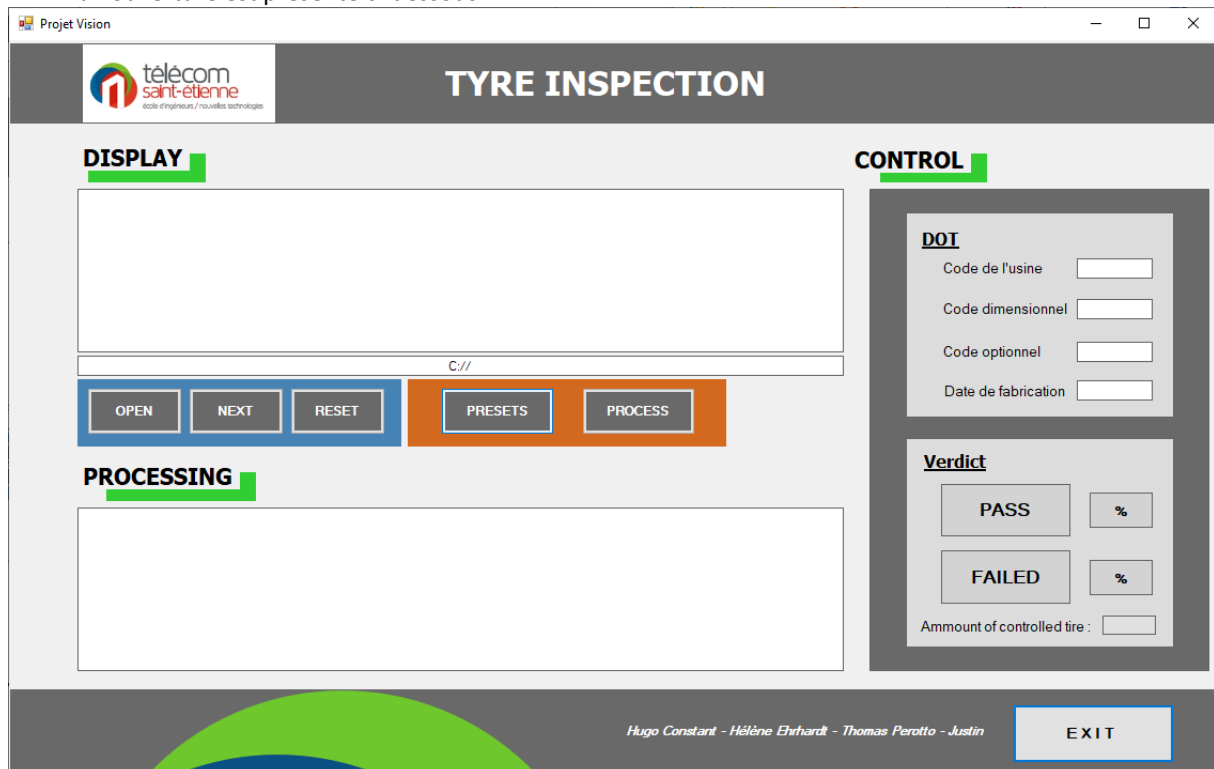
2.2. Développement de l'IHM

a) Présentation générale

Nous pouvons envisager que l'utilisateur final de l'application soit un opérateur dans une usine, un garagiste, ou encore un vendeur. De ce fait, l'application se doit de proposer une IHM ergonomique et épurée afin de répondre aux besoins de chacun de ces trois types d'utilisateurs.

N'ayant pas de demande spécifique du client, nous avons considéré que le critère d'admissibilité d'un pneu est que sa date de fabrication doit dater de moins de 5 ans. En effet, au-delà de cette période le pneu requiert une analyse visuelle de l'état du caoutchouc pour pouvoir être mis en vente. L'information de la date de fabrication se trouve dans le DOT du pneu, en général en 4^{ème} position.

L'IHM à l'ouverture est présenté ci-dessous :



Voici la description des fonctionnalités de l'IHM :

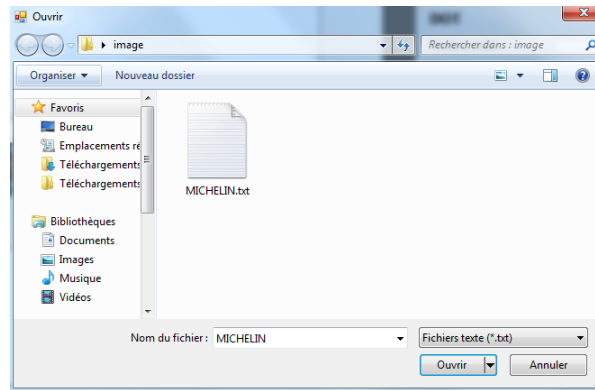
1. Bouton **OPEN** : Ouverture du dossier dans lequel les images de pneus sont à analyser
2. Bouton **NEXT** : Défilement des images stockées dans le dossier
3. Bouton **RESET** : Réinitialisation du contrôle
4. Bouton **PRESETS** : Sélection des paramètres définis pour la marque de pneu correspondant
5. Bouton **PROCESS** : Traitement de l'image
 - a. Affichage de l'image traitée dans le cadre « *Processing* »
 - b. Affichage des caractères du DOT extraits dans le cadre « DOT »
 - c. Dans le cadre « Verdict » :
 - i. Si le pneu est valide, alors le cadre **PASS** s'affiche vert,
 - ii. Sinon le cadre **FAILED** s'affiche rouge
 - iii. Deux textes supplémentaires affichent les pourcentages de pneus acceptés et rejetés.
6. Bouton **EXIT** : Ferme l'application

b) Protocole d'utilisation

Cette section présente le protocole d'utilisation client de l'application « Tyre Inspection ».

1- Sélection des Presets

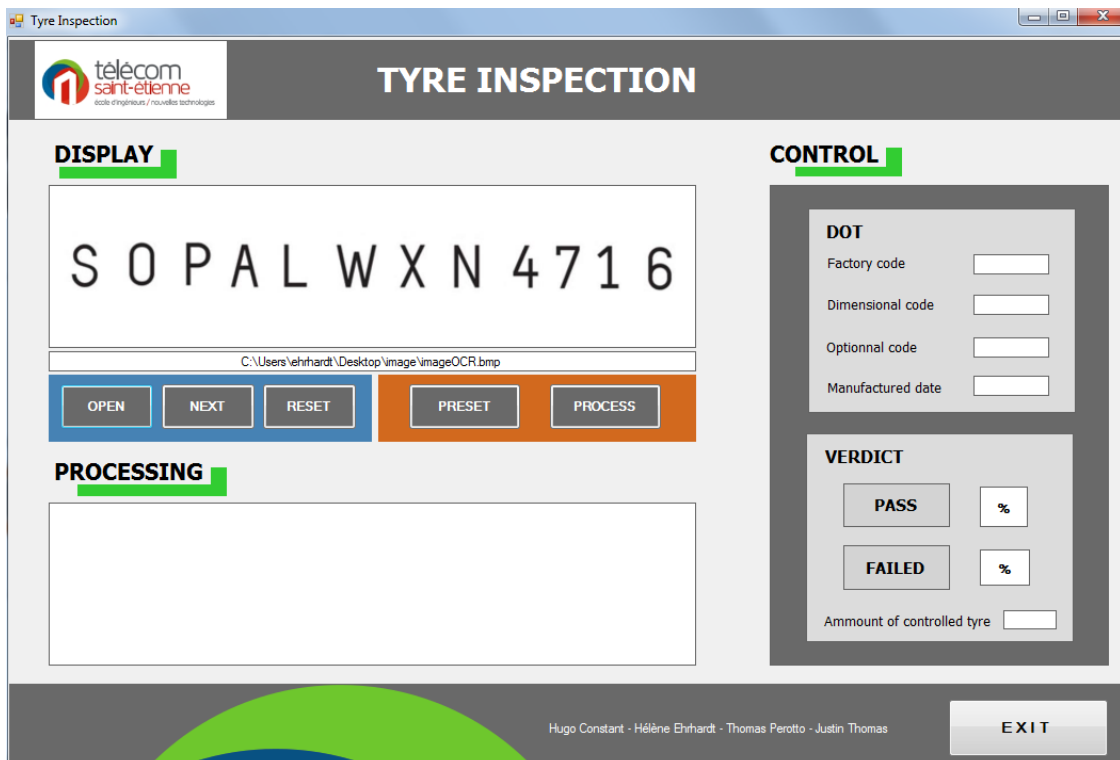
Les fichiers « presets » sont des fichiers texte (.txt), contenant les paramètres fixés pour les différents types de pneus. Le client doit en premier lieu sélectionner le fichier « presets » portant le nom de la marque des pneus à analyser.



Sélection du fichier preset pour pneus Michelin

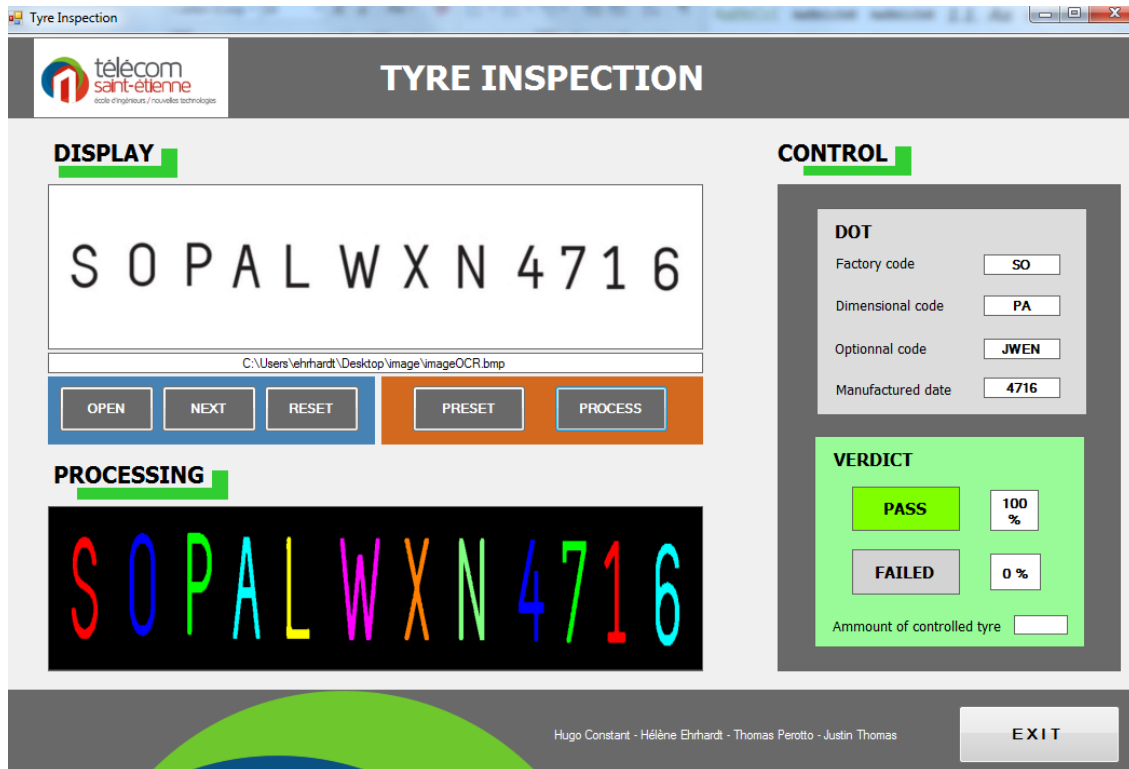
2- Ouverture du dossier de travail « OPEN »

Les images à analyser sont stockées dans un répertoire de travail. Cette action affiche la première image du dossier dans la fenêtre « display ». L'ensemble des images à traiter doit être au format BMP.



3- Analyse du pneu

L'action « process » effectue le traitement de l'image en cours. Une image étiquetée représentant chacun des caractères est affichée dans la section « *processing* ».



Dans notre démonstrateur, nous avons simulé un verdict basé sur le DOT des pneus :

Les caractéristiques DOT du pneu sont extraites et affichées dans le panel « control » - « DOT ». Les différents codes sont retrouvables dans les quatre sections associées :

- « **Factory code** » : Code de l'usine où le pneu a été fabriqué.
- « **Dimensional code** » : Code dimensionnel propre au manufacturier.
- « **Optionnal code** » : Code optionnel propre au fabriquant.
- « **Manufactured date** » : Date de fabrication du pneu. Ici, le pneu a été fabriqué lors de la 47ème semaine de l'année 2016.

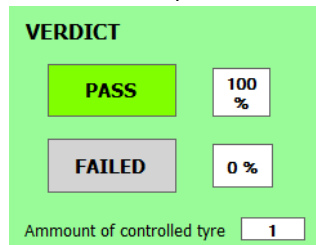
Le panneau ci-dessous illustre l'affichage d'un DOT dans l'IHM.

DOT	
Factory code	SO
Dimensional code	PA
Optionnal code	JWEN
Manufactured date	4716

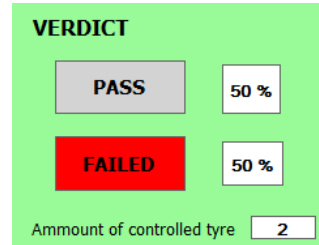
Exemple d'un DOT extrait

Le panel « *verdict* » indique la validation ou non du pneu selon le critère imposé dans ce projet. Dans le cas d'un succès, l'indicateur « PASS » s'affichera vert, dans le cas contraire l'indicateur « FAILED » s'affichera rouge. Une couleur de contour rouge ou verte indique également si la quantité de pneus valide est supérieure à 50%

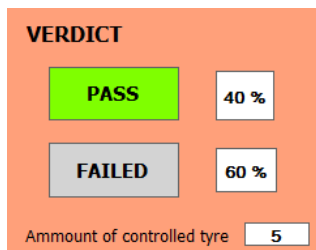
Un pourcentage relatif au nombre de succès ou d'échec s'affiche et s'actualise en fonction des images traitées. Les différents cas possibles sont illustrés ci-dessous.



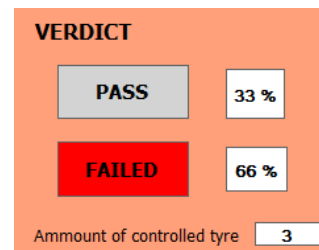
Pneu actuel valide, et plus de 50% des pneus testés également valides



Pneu actuel non valide, mais plus de 50% des pneus testés valides



Pneu actuel valide, mais plus de 50% des pneus testés non valides



Pneu actuel non valide, et plus de 50% des pneus testés non valides

4- Traitement de l'image suivante « NEXT »

Cette action sélectionne l'image suivante dans le dossier de travail. Un clic sur l'action « PROCESS » permet de réitérer le traitement de l'image.

5- Réinitialisation du panel « RESET »

L'action « reset » permet de réinitialiser le dossier de travail et le panel de contrôle. Les *Presets* sont réinitialisés, et les images effacées. Le logiciel est prêt à être utilisé, comme si celui-ci venait de s'ouvrir.

6- Fermeture de l'application « EXIT »

La fermeture de l'application « Tyre Inspection » se fait via le bouton « Exit ».

c) Analyse de l'application

L'application développée est simple d'utilisation et ergonomique, afin d'être accessible à une grande diversité d'utilisateurs. De plus, chaque client peut adapter l'application à ses besoins via la modification des fichiers de *Presets*, qui peuvent s'adapter à différents types de fournisseurs de pneus.

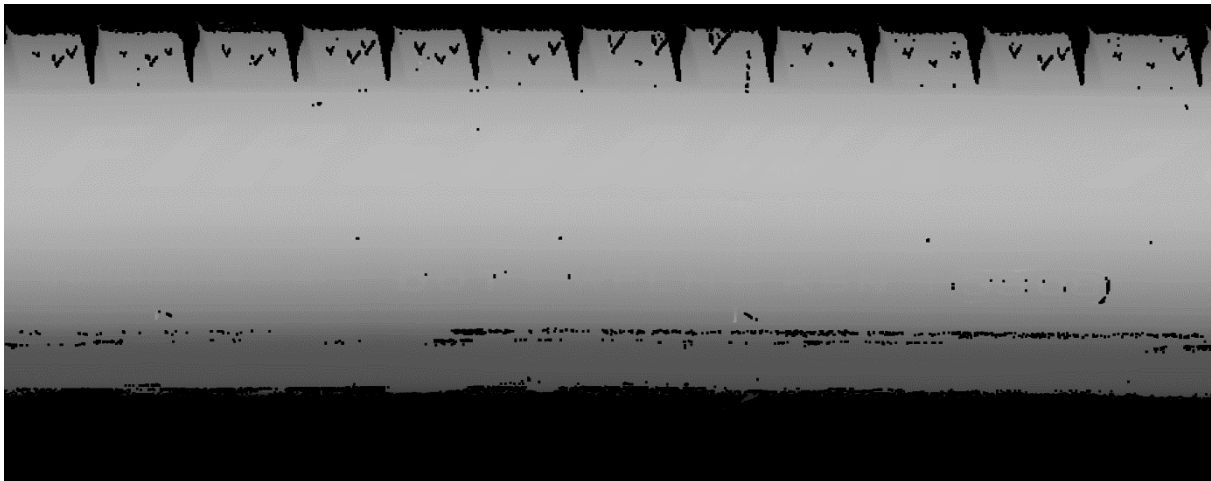
Un des points d'amélioration consiste à réduire temps de traitement, afin de ne pas figer l'IHM lors du traitement des images. De même, un autre point à améliorer serait un défilement automatique des images, afin de traiter directement l'ensemble des images d'un dossier.

Ces options n'ont pas été retenues, n'étant pas formellement rédigées dans la spécification technique.

2.3. *Traitement des images*

a) *Images originales*

Une des difficultés principales du projet se situe sur la base d'images : Les images sont issues d'une acquisition 3D retransmises en 2D sur laquelle la variation des nuances de gris représente l'état de la surface des pneus. Les images présentent donc de fortes variations horizontales de niveaux de gris (car le pneu n'est pas plat). Aussi, les caractères sont peu discernables du fond à cause de la faible différence d'intensité avec leur fond. De ce fait, il a été compliqué de traiter l'image afin d'obtenir un résultat satisfaisant pour l'algorithme d'OCR. L'image ci-dessous représente l'image initiale d'un pneu.



La première étape du projet a été d'identifier les traitements permettant une bonne segmentation de l'image. Une étude des méthodes de traitement sur Matlab a permis d'identifier les outils à utiliser. Celle-ci a permis de tester des méthodes de filtrages et d'explorer l'ensemble des fonctions disponibles. Une approche par gradient a permis d'obtenir des premières images exploitables (au sens de la vision humaine) :

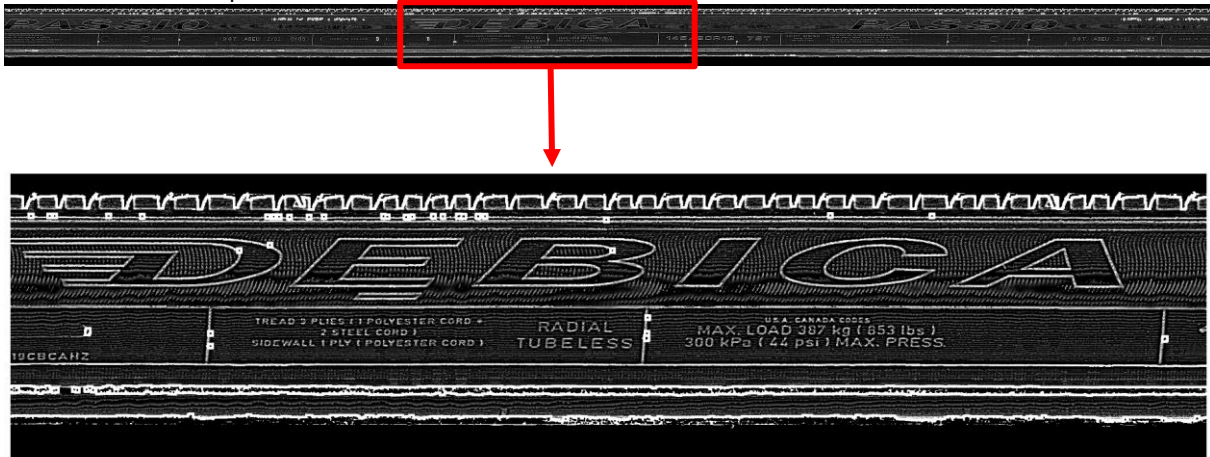


Après avoir testé les images sur Matlab, les mêmes méthodes ont été appliquées sous Visual studio. A l'aide de la bibliothèque de traitement d'images disponible, un traitement similaire à celui effectué sur Matlab a été réalisé, le but de ne garder que les informations qui nous intéressent (extraction du DOT avec le moins de bruit possible).

Malgré les différents traitements testés, il ne fut pas possible d'extraire le DOT en minimisant le bruit. Les images utilisées dans la suite du projet sont ainsi des images prétraitées, fournies par Yves Bringer.

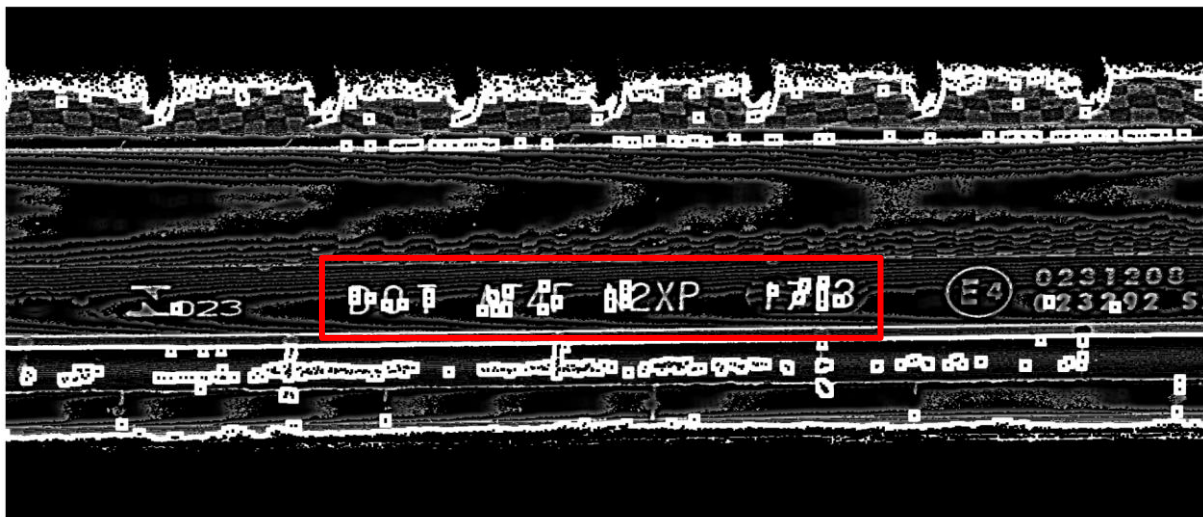
b) *Images prétraitées par l'enseignant*

Suite à l'échec du traitement sur les images initiales, une base d'image prétraitée a été utilisée. L'image ci-dessous est un exemple d'une des images utilisée :



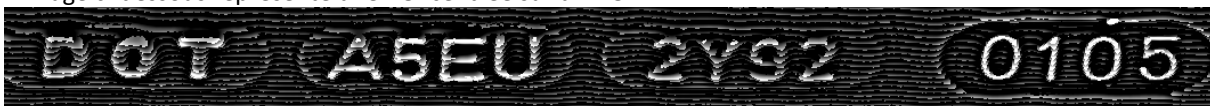
Lorsqu'on zoom sur certaines images, on distingue à certains endroits que l'image est bruitée, rendant impossible l'exploitation de tous les caractères.

Sur l'image ci-dessous, on peut observer que le DOT est parasité par des carrés blancs (en rouge). Ces carrés blancs sont issus du filtre gradient. Ils indiquent des détections de contours.



L'autre difficulté des images est leurs tailles importante (14707 * 749 pixels). Du fait de cette taille, le temps de traitement de chaque opération est relativement long, de l'ordre de plusieurs dizaines de secondes. Afin d'éviter les traitements trop longs, des ROIs manuelles ont été effectuées.

L'image ci-dessous représente une ROI centrée sur un DOT :

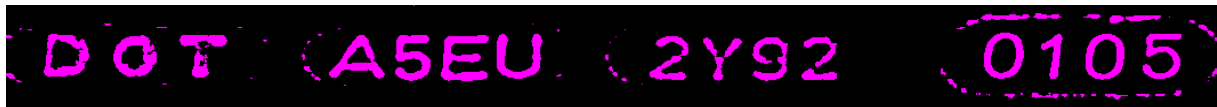


Par la suite, plusieurs étapes de traitement d'images ont été effectuées afin d'extraire les caractères.

c) Etapes de traitement d'image

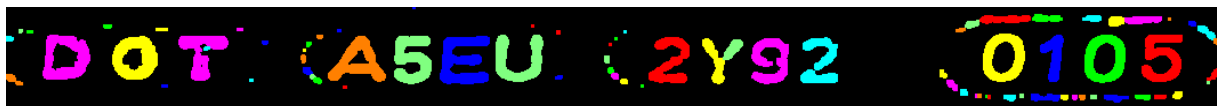
Etape 1 : filtre moyen et seuillage automatique

- Le filtrage moyen permet de lier les éléments entres eux.
- Le seuillage automatique permet de segmenter les éléments qui nous intéressent.



Etape 2 : Morphologie : Dilatation

- Etiquetage avant la dilatation. Durant la phase de tests, celui-ci permettait de suivre l'évolution des morphologies et évaluer l'impact du traitement.
- La dilatation permet d'étendre les éléments afin de pouvoir lier les numéros (par exemple le « Y »).



Etape 3 : Filtrage des éléments

- Le filtrage par taille d'objet (min et max) permet de supprimer les éléments inutiles pour la lecture OCR par la suite.



Cette méthode traitement a été appliquée sur l'ensemble des images de pneus disponibles. Les images suivantes sont des exemples de résultats obtenus, illustrant la capacité du code à segmenter les numéros :



Concernant la différence entre les images, celles-ci proviennent de la segmentation, qui a fait ressortir les contours des numéros et pas forcément l'intérieur. Dans le cas des caractères non remplis, un remplissage peut toutefois être réalisé, en utilisant les fonctions disponibles dans la bibliothèque fournie par Telecom Saint Etienne.

2.4. OCR

Il existe de multiples méthodes de mise en place de reconnaissance de caractères. Aujourd'hui, la meilleure méthode est celle employant des algorithmes de machine learning ou de deep learning. Néanmoins, de tels algorithmes nécessitent une très grande phase d'apprentissage, couplée à une très grande base d'images. Pour de l'OCR, il faut compter en général environ 6000 images pour commencer à obtenir un résultat satisfaisant avec des images « parfaites ».

Les images proposées dans notre sujet n'étant ni « parfaites », ni très nombreuses, il nous était donc impossible d'utiliser cette solution.

Nous avons donc décidé de générer un arbre de vérité (ou décision) afin d'être capable de classer les différents caractères.

a) Théorie

Cette méthode permet de classer les caractères d'une image à l'aide de la comparaison de leurs signatures avec les caractéristiques d'un arbre de vérité.

Exemple de fonctionnement :

1. Extraction des signatures des images de la base de référence (en général sous format .csv) :

CARACTÈRE	SIG1	SIG2	SIG3	SIG4	SIG5	SIG6	SIG7
I	10	0,5
l	11	0,51
A	20	9,6
Z	54	0,9
l	11	0,48
1	10	0,932
...							

2. Réalisation de l'arbre de vérité (ou arbre de décision) :

A l'aide de logiciels (par exemple : WEKA) ou manuellement, déterminer les « limites » de chaque signature permettant de différencier les caractères entre eux.

Si on reprend l'exemple ci-dessus :

- La signature « SIG1 » permet de différencier le « A », le « Z » et le duo « l/1 »
- La signature « SIG2 » permet de différencier le « l », le « A » et le duo « Z/1 »

En théorie, avec juste ces deux signatures, il est possible d'isoler chacun de ces 4 caractères dans une image.

3. Décision :

Il y a plusieurs moyens de coder cet arbre de décision, le premier est de construire une arborescence de conditions IF (prend beaucoup de place mais permet de visuellement ne pas se perdre dans l'arbre) :

```

if(SIG1<15)
{
    if(SIG2<0,75)
    {
        result = I;
    }
    else
    {
        result = 1;
    }
}
else if(...)

```

Un second serait de composer une condition globale pour chaque résultat possible (peu pratique) :

```
if(SIG1<15 && SIG2<0,75)
{
    result = I;
}
if(SIG1<15 && SIG2>0,75)
{
    result = 1;
}
if(SIG1<15 && SIG2>0,75 && SIG3<xx && SIG4<xx && ...)
{
    result = ...;
}
```

Les deux méthodes ci-dessus sont des algorithmes stricts, si un caractère possède seulement un paramètre ne correspondant pas aux conditions, il ne pourra pas être classé.

Un troisième moyen, peut employer un système de probabilité afin de déterminer le caractère, il permettrait aussi de voir quels autres caractères étaient proches (demande plus de temps, mais permet d'obtenir un résultat plus détaillé et plus souple) :

```
if(SIG1<15)
{
    probaI ++;
    proba1 ++;
}
else if(SIG1>15 && SIG1<40)
{
    probaA ++;
}
[...]

if(SIG2<0,75)
{
    probaI ++;
}
else if(...)
[...]

// result = highest proba
```

Ce troisième algorithme est plus souple mais présente un gros bémol, comment est-ce que la machine doit réagir dans le cas de deux probabilités égales ?

- Une première réponse serait d'utiliser des coefficients sur les signatures :
Par exemple : si SIG1>40, il y a de très fortes chances que le caractère soit celle d'un « Z » donc la probabilité probaZ est augmentée de 2 à la place de 1.
- Une seconde réponse, similaire dans la façon de penser, mais amenée différemment serait de rendre certaines signatures décisives :
Par exemple : en cas d'égalité de probabilité, si SIG>40 alors le caractère est un « Z »

Il existe très certainement de nombreux autres moyens d'interpréter l'utilisation d'un arbre de vérité à notre besoin, mais les algorithmes présentés ci-dessus sont des algorithmes « faciles » à implémenter. Notre temps de projet étant très limité, il va de soi qu'une méthode rapide était la plus adaptée.

Dans le projet, c'est l'algorithme utilisant le système de probabilité qui a été implémenté.

b) Mise en place

La génération des signatures a été réalisée à partir des bibliothèques développées à TSE, ainsi que de la documentation trouvée sur Internet. Huit signatures robustes ont été retenues. Nous sommes partis sur ce nombre en ne sachant pas s'il serait suffisant pour effectuer l'OCR. Deux signatures robustes supplémentaires ont également été ajoutées afin de limiter les mauvaises détections.

La bibliothèque d'analyse et traitement d'images, les fonctions liées à l'Image Classe ont été utilisées, adaptées et complétées afin de répondre au besoin formulé ci-dessus. Une représentation graphique de ces signatures et de leurs attributs est présentée ci-dessous :



Signatures complémentaires : Le nombre d'Euler et l'angle de l'axe majeur.

Détail des signatures :

$$* \text{Ratio d'aspect} = \frac{\text{hauteur}}{\text{largeur}}$$

* Non imagée

$$\text{Élongation} = \frac{\text{largeur}}{\text{longueur}}$$

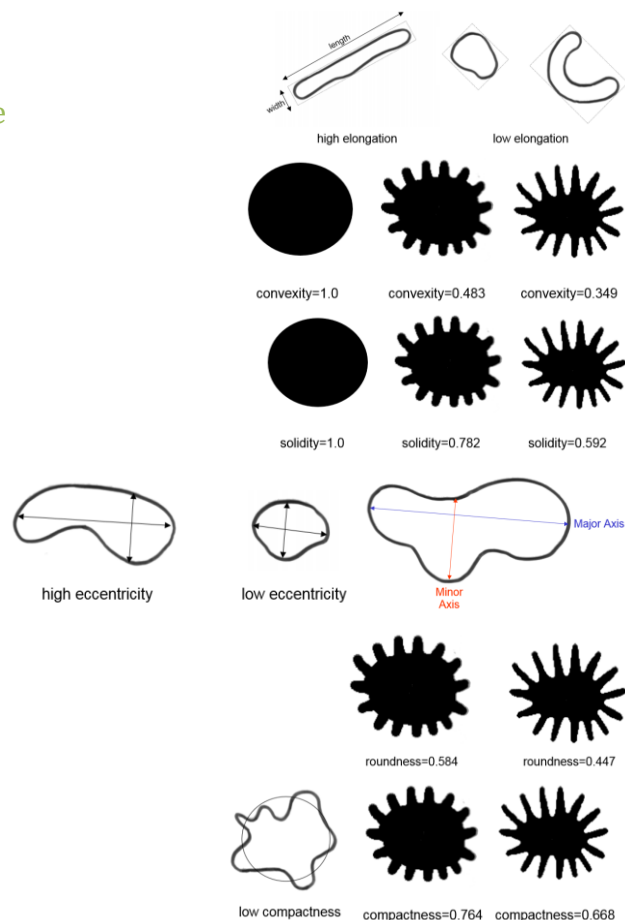
$$\text{Convexité} = \frac{\text{périmètre}_{\text{convexe}}}{\text{périmètre}}$$

$$\text{Solidité} = \frac{\text{aire}}{\text{aire}_{\text{convexe}}}$$

$$\text{Éllipticité} = \frac{\text{longueur}_{\text{Axe mineur}}}{\text{longueur}_{\text{Axe majeur}}}$$

$$\text{Circularité} = \frac{4\pi * \text{aire}}{\text{périmètre}_{\text{convexe}}^2}$$

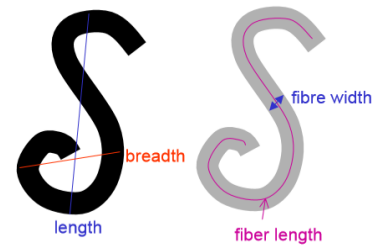
$$\text{Compacité} = \frac{\text{périmètre}^2}{4\pi * \text{aire}}$$



$$Boucle = \frac{longueur}{longueur_{fibre}}$$

$$longueur_{fibre} = \frac{perimetre - \sqrt{perimetre^2 - 16 * aire}}{4}$$

$$largeur_{fibre} = \frac{aire}{longueur_{fibre}}$$



Réalisation de la table de vérité :

Etant donné qu'aucune image de pneu n'était utilisable au moment du développement de la table de vérité, et ne pouvant pas nous permettre de les attendre, nous avons décidé d'utiliser une image d'un alphabet lettre et chiffre en provenance d'internet. Celui-ci présente uniquement des caractères « parfait », en effet, une première étape de l'OCR a été la validation de l'arbre de vérité et du nombre de signatures utilisées :

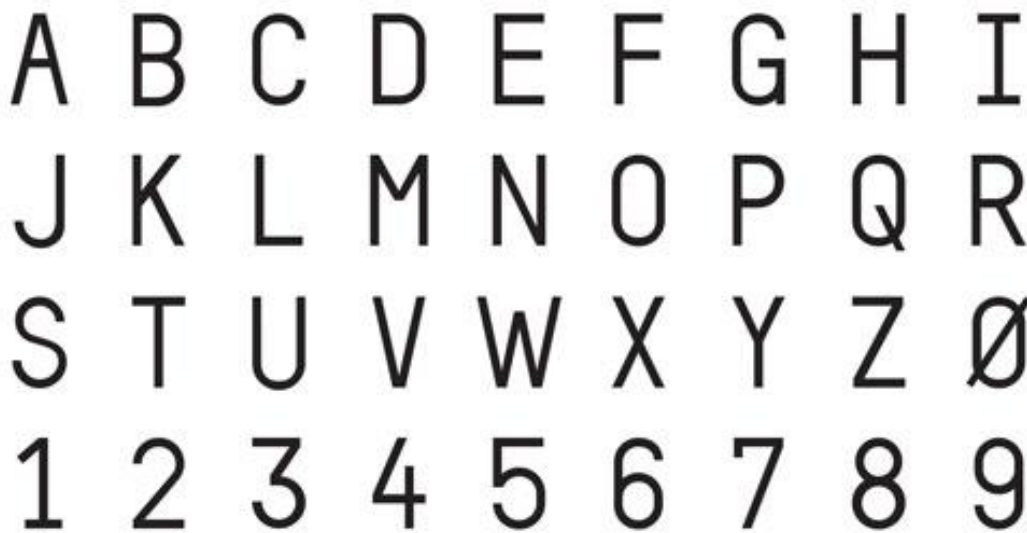


Tableau des signatures obtenu :

Objet	elongation	compacité	ellipticité	circularité	Convexité	Ratio d'aspect	boucle	solidité
A	0.609756	4.61348	0.371429	32.1723	0.340589	1.64	25.3033	140.878
B	0.547619	2.44016	0.299489	50.6928	0.377362	1.82609	15.4012	257.765
C	0.571429	7.28722	0.326149	24.2494	0.27432	1.75	36.3245	115.358
D	0.585366	2.56081	0.342262	47.5628	0.399619	1.70833	15.8042	215.977
...								

Fonctionnalités supplémentaires de l'OCR :

Comme les écritures sur les pneus pouvaient être sur plusieurs lignes, présenter de long espaces entre 2 « mots » et avoir les lettres d'un même mot décalées en hauteur, des fonctionnalités additionnelles ont été programmées :

- La gestion du saut de ligne,
- La gestion des espaces entre mots,
- La réorganisation des caractères de sorte à ce qu'ils soient rangés de droite à gauche.

c) Résultats

Une première phase de test avec l'image d'apprentissage a retourné un résultat avec un taux de détections correct à 100%.

Néanmoins, une seconde phase de test avec une image modifiée à l'échelle comprenant plusieurs combinaisons de caractères ne nous offrait qu'un niveau de détections correctes à 55%. Cela a démontré que contrairement à ce que nous pensions, nos signatures ne sont pas toutes entièrement robustes à l'échelle. En effet, nous obtenions des valeurs sur nos signatures similaires mais tout de même un peu différentes.

Ci-dessous, image test utilisée :

A	B	C	D	E	F	G	H	1	2	1	9
A	B	C	D	E	F	G	H	3	4	1	4
A	B	C	J	K	L	G	H	4	5	1	8
U	V	C	J	K	J	G	H	5	1	1	3
U	O	P	J	K	J	G	H	2	8	1	3
T	O	P	Z	K	J	G	N	Ø	3	2	Ø
T	O	P	A	K	W	X	N	Ø	7	1	2
S	O	P	A	L	W	X	N	4	7	1	6
Q	C	V	A	L	W	X	N	1	7	1	7
Q	I	Y	A	L	D	X	N	1	4	1	1

Suite à cette observation, nous avons mis à jour notre arbre de vérité afin d'assouplir les conditions :

- Nous avons commencé par assouplir 3 signatures, le taux de détection correctes est monté à 70%.
- Et en assouplissant 6, le taux de détections correctes est monté à 95%

Cette observation nous a permis de voir que notre choix d'avoir 8 signatures était, par chance, correcte. Néanmoins, l'ajout de 2 signatures supplémentaire ne peut qu'améliorer la qualité de la détection.

Ci-dessous, le résultat obtenu pour l'image test précédente après assouplissement de 6 conditions :

```

C:\Users\hugoc\Desktop\HC_Image_TestPneu\Devs\Debug\prise_en_main.exe
***** START *****
img charged
img Seuil
Seuillage des pixels entre 101 et 255
img Etiquettage
120 objets dans l'image...
img Save img_Etiq
Signatures Ndg
Signatures forme
ABCEDEFGH1219
ABCEDEFGH3414
ABCJKLGH4518
UVCJKJGH5113
UOPJKJGH2813
TOPZKJGN0320
TOPAKWXN0712
SOPALWXN4716
QCVALWXN1717
QIYALDXN1411

***** END *****
Appuyez sur une touche pour continuer...
  
```

d) Intégration à l'IHM

Comme indiqué dans la partie 3.1 – *Architecture*, l'OCR est située dans la couche bas-niveau (Low Level Driver). La fonction de traitement d'OCR prend en entrée une image niveau de gris, et retourne une chaîne de caractère contenant le DOT. La fonction n'est pas directement accessible via le Wrapper, celle-ci étant privée au sein du LLD (encapsulation). L'unique moyen de lancer le traitement est d'appeler la fonction d'extraction du DOT, faisant elle-même appel à l'OCR. Le passage par cette étape permet de réaliser des opérations de traitement d'image en amont de la reconnaissance de caractères.

Depuis le Wrapper, l'IHM récupère une chaîne de caractères contenant le DOT. Celle-ci est découpée, afin d'en extraire le code de l'usine, le code dimensionnel, le code optionnel et la date de fabrication du pneu. Ces informations permettent d'évaluer la validité du pneu, et ainsi d'établir un verdict.

3. Conclusion

3.1. Conclusion technique

Au cours de ce projet, nous nous sommes consacré au développement d'un algorithme, capable d'évaluer si un pneu pouvait encore être vendu en fonction de sa date de fabrication. L'objectif était donc de développer une interface utilisateur simple et efficace (utilisable par des opérateurs), un traitement d'image, ainsi qu'une reconnaissance de caractères. L'ensemble des points clefs est spécifié en *partie I-1 Spécification technique*.

Concernant l'interface, celle-ci a été réalisée comme convenue dans la spécification : Les éléments importants sont colorés, afin d'obtenir un visuel rapide sur les informations. Deux zones d'affichages permettent de voir l'image initiale, et l'image traitée, dans le but de toujours avoir la vérité terrain à disposition. Enfin, l'extraction des champs du DOT est dans une partie séparée, chacune des parties étant visible par l'utilisateur. Le verdict est indiqué clairement, et un taux de rejet/d'acceptance est également intégré.

Le traitement d'image a été initialement réalisé sur les images fournies par le client. Une première approche réalisée sous Matlab, a permis d'étudier les outils à disposition et de sélectionner les plus efficaces. L'approche par gradient permet de faire ressortir le DOT, mais induit également un taux de bruit important. Compte tenu de celui-ci, les images client initiales ont par la suite été mises de côté. Des images prétraitées en provenance de Yves Bringer ont finalement été utilisées. Le traitement d'image réalisé n'est donc pas optimal.

Enfin, l'OCR a été développé sur une base de huit signatures. Ces signatures, ont permis de réaliser le programme de reconnaissance de caractères, sur l'ensemble de l'alphabet et des chiffres disponibles. Lors de la phase de tests, un taux de 100% de réussite sur l'image de test a été obtenu : chaque caractère a été reconnu correctement. En revanche, un point à améliorer est la robustesse de l'OCR. En utilisant une police de caractère différentes, les valeurs de l'arbre de vérité sont à mettre à jour.

En conclusion, ce projet peut être considéré comme terminé à 80%. Les points critiques à améliorer sont axés autour du traitement d'images, et de la robustesse des signatures. Par rapport au traitement d'images, d'autres méthodes de traitement pourraient être envisagées, comme par exemple l'implémentation d'un HOG local (Histogrammes des Gradient Orientés). Concernant l'OCR, il pourrait être possible de réaliser une base de signatures par type de police d'écriture. En fonction du fabricant de pneu, la base de signatures pourrait être sélectionnées via un *Preset*.

Enfin, une approche différente consisterait à utiliser un algorithme de machine learning pour l'apprentissage des caractères. Cette méthode permettrait de classifier les caractères parmi des bases d'images apprises, afin d'éliminer les problématiques de robustesse liées à la police.

3.2. Conclusion personnelle

Ce projet a tout d'abord permis de conforter les bases en développement d'IHM et en traitement d'image. Le fait de réaliser une application intégrale pour un besoin client peut être considéré comme une des finalités du module Vision.

Au cours du projet, des difficultés techniques ont ralenti la progression, et nécessités une augmentation de l'effort. Malgré l'énergie déployée, il n'a pas été possible de fournir une solution optimale pour l'ensemble des images proposées par le client. A la fin du projet, des badges de compétences ont été accordées à chaque membre de l'équipe, illustrant les compétences clefs de chacun :

Justin Thomas



Motivation



Envie
d'apprendre



Esprit
d'initiative

Hugo Constant



Goût
du challenge



Sens de
l'efficacité



Multi-tâches

Hélène Ehrhardt



Créativité



Optimisme



Gestion du stress

Thomas Perotto



Management



Leadership



Intelligence
relationnelle

Un document personnel comprenant le rapport d'étonnement de chacun des membres du groupe est également disponible, afin de décrire le ressenti de chacun au cours du projet.

Lexique

DLL : Dynamic Link Library : Bibliothèque de fonctions et méthodes. Dans le cadre de ce projet, le Low Level Driver est encapsulé dans la DLL.

DOT : Department Of Transportation : Code en quatre parties, permettant d'établir la traçabilité d'un pneu (date de fabrication, code de l'usine de fabrication, ...)

LLD : Low Level Driver : Bas-niveau de l'architecture logicielle, contenant les opérations traitement d'image et d'OCR.

OCR : Optical Character Recognition : Procédé informatique permettant de reconnaître des caractères sur une image.

Presets : Tableau de valeurs permettant de configurer les opérations de traitement d'image situées dans le LLD.

ROI : Region Of Interest : Zone particulière d'une image