N°Ordre……../Faculté/UMBB/2016

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE M'HAMED BOUGARA BOUMERDES

Faculté des Hydrocarbures et de la Chimie

Mémoire de fin d'études
En vue de l'obtention du diplôme

MASTER

Présenté par

KISSOUM Ghani

Spécialité : Automatisation des procédés industriels
Option : Commande automatique

THEME

Détection et reconnaissance des panneaux de signalisation routière sous LabVIEW
Application à un robot mobile de type véhicule « ROBUCAR »

Devant les membres de jury :

ACHELI	Dalila	Professeur	UMBB	Président
MEZIANE	Nacera	MC/B	UMBB	Examinateur
KAHOUL	Fadhila	MA/A	UMBB	Encadreur
HAMMADOU	Dalila	ING RE	CDTA	Co-promoteur

Année universitaire : 2015/2016

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE M'HAMED BOUGARA BOUMERDES

Faculté des Hydrocarbures et de la Chimie

Mémoire de fin d'études
En vue de l'obtention du diplôme

MASTER

Présenté par

KISSOUM Ghani

Spécialité : Automatisation des procédés industriels
Option : Commande automatique

THEME

Détection et reconnaissance des panneaux de signalisation routière sous LabVIEW
Application à un robot mobile de type véhicule « ROBUCAR »

Avis favorable de l'encadreur :

Mme. KAHOUL Fadhila

Avis favorable du Président du jury :

Pr. ACHELI Dalila

Résumé:

La reconnaissance des panneaux de signalisation routière est une thématique qui suscite un intérêt scientifique et industriel élevé. Relevant du domaine de traitement d'images, la détection des panneaux se base sur des techniques de reconnaissance de formes, de couleurs, de présence de pictogrammes ou de texte, etc. Une classification s'impose donc pour faire face aux problèmes liés aux changements de conditions d'éclairage, d'occultation partielle, rotations, etc. L'objectif de ce travail est de développer et implémenter un programme pour la détection de panneaux routiers sous LabVIEW à partir des images issues d'une caméra embarquée sur le robot mobile « ROBUCAR ». Dans un premier temps, une familiarisation avec l'outil LabVIEW ainsi qu'une recherche bibliographique sur les techniques employées sont nécessaires. Le reste du travail consiste à classifier un ensemble de panneaux grâce à un apprentissage multi positions, et un traitement des détections et identification des consignes (actions sur le véhicule).

Mots-clés : Reconnaissance de panneaux de signalisation, vision artificielle, traitement d'images, LabVIEW.

Abstract:

The recognition of road signs is a subject that arouses a high scientific and industrial interest. Reporting to the image processing field, detection of road signs is based on pattern recognition techniques, colors, presence of pictograms or text, etc. A classification is therefore needed to deal with problems related to changes in lighting conditions, partial occultation, rotations, etc. The goal of this work is to develop and implement a program for the detection of road signs in LabVIEW from the images provided by a camera mounted on the mobile robot "ROBUCAR". First, familiarization with the LabVIEW tool and a bibliographic research on the techniques used are necessary. The remaining work is to classify a set of panels through learning multiple positions, and treatment for detection and identification of instructions (actions on the vehicle).

Keywords: Recognition of road signs, computer vision, image processing, LabVIEW.

Agzul:

Asikez n yiglisen n yiberdan d asentel i yesɛan azal meqren deg wannar usnan d wanar amguran. Asikez n tugniwin yettilid s useqdec n tfuksiwin n usssikez n talɣa d yiniten atg. Ilaq usismel iwaken anbed mgal uguren yeqnen ɣer wesfaw, tufra d tuzya atg.Amesɣaru n unadi agi d anegmu d usekcem n yiwen wahil i tifin d wefraz n yiglisen n wedrid s useqdec n ufcku n LabVIEW, s tugniwin id yettwaṭfen s twessaft yerssen ɣef yiwen urubu iyelleḥun. Di tazwara s useqdec n ufecku n LabVIEW d unadi amkardan ɣef tifuksiwin ara neseqdec deg usenfar agi .Ayen id yeqqimen deg ssenfar agi newid awal degs ɣef usismel n yiwet n tgumma n yeglisen n yiberdan s useqdec n yiwen walal n usselmed ig cuden ɣer watas n yidisan, d usqardec d umeyez n yiwelihen I nefka i urubu.

Awalen iğenṭaḍen: assikez n yiglisen n yiberdan, asqerdec n tugniwin, LabVIEW.

موجز:

ان التعرف على إشارات المرور هو موضوع يثير اهتمام علمي وصناعي عالي. منتميا الى مجال معالجة الصور، يعتمد هذا الكشف على تقنيات التعرف على الأشكال والألوان، ووجود الصور التوضيحية أو النص،... الخ . هناك حاجة إلى التصنيف للتعامل مع المشاكل المتصلة بالتغيرات في ظروف الإضاءة، وعرقلة جزئية، ... الخ . الهدف من هذا العمل هو تطوير وتنفيذ وحدة للكشف عن اشارات المرور في لاب فيو إنطلاقا من الصور الملتقطة بواسطة كاميرا محمولة على الروبوت المتنقل

أولا،يجب الألفة مع أداة لاب فيو ودراسة ببليوغرافية للتقنيات المستعملة. سيكون العمل المتبقي يتمحور حول كشف وتصنيف مجموعة من لوحات من خلال تعلم متعدد الوضعيات ، معالجة الكشف وتحديد التعليمات (الإجراءات على الروبوت)

الكلمات المفتاحية: التعرف على إشارات المرور ، معالجة الصور ، لاب فيو ، الرؤية الإصطناعية

Sommaire

Chapitre II : Présentation de l'environnement LabVIEW et du contrôleur CompactRIO

Chapitre III : Développement du programme de détection et reconnaissance des panneaux

Chapitre IV : Résultats et discussions.

Chapitre V : Actions sur le Robucar.

Liste des figures

Liste des tableaux

Introduction générale.

Les robots sont développés pour assurer plusieurs taches dans différents domaines : en industrie pour assurer les taches d'assemblage, en astronomie pour explorer d'autres planètes. L'un des champs de recherche en robotique les plus actifs est le développement des véhicules autonomes.

Un véhicule autonome peut naviguer dans un environnement familier sans intervention ou control humain. Pour naviguer correctement, ils ont besoin d'informations sur leur environnement et doivent réagir en conséquence. Depuis les années 60 **[23]**, les recherches sont orientés au développent de robots utilisant la vision pour la navigation, ce qui a donné naissance à une nouvelle discipline qui est la vision par ordinateur, appelée aussi vision artificielle ou vision numérique. Etant donné le nombre d'applications industrielles militaires aérospatiales et médicales qui peut être envisagé, la vision par ordinateur a vite fait de dépasser le cadre relativement restreint des laboratoires de recherche. Aujourd'hui rares sont les universités et les écoles d'ingénieurs ne proposant pas des cours de vision par ordinateur.

Cette discipline est une branche de l'intelligence artificielle dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs images prises par un système d'acquisition (caméra CCD, infra-rouge …etc.). Une approche consiste à tenter d'imiter la vision humaine ou animale par le truchement de composants électroniques. Cette manière de procéder peut être perçue comme un traitement des données visuelles par le biais de modèles fondés sur la géométrie, la physique, la biologie, les statistiques et la théorie d'apprentissage. La vision par ordinateur a aussi été décrite comme une initiative dans l'automatisation et l'intégration d'une vaste gamme de processus et de modèles sur la perception visuelle.

Pour des véhicules complètement autonomes, il est extrêmement important qu'ils soient dotés d'un système de reconnaissance de panneaux signalétiques qui fournissent un grand nombre d'informations utiles à la navigation. Ce système ne se limite pas aux véhicules autonomes, mais il peut aussi constituer un outil d'aide à la conduite, un conducteur peut être distrait de sa tâche principale, la conduite, ce qui occasionne un manque de vigilance vis à vis de la signalisation courante. Cette situation augmente le risque d'accident. En effet, manquer un panneau de limitation de vitesse ou d'interdiction génère une situation à risque en plus du fait d'exposer le conducteur à des sanctions. Afin de réduire ce risque, le système de reconnaissance de panneaux routiers (TSR-Traffic Signs Recognition) assiste le conducteur en le tenant informé de la signalisation liée à la route parcourue.

Les recherches dans le domaine du TSR (Traffic Sign Recognition) ont commencé à partir des années 1980. La première approche naïve et directe est d'appliquer un filtre de corrélation croisée normalisée aux images brutes afin à la fois de détecter et de reconnaître les panneaux. Cependant cette méthode par brute force demande un temps de calcul beaucoup trop important pour être réellement envisageable (les panneaux pouvant avoir n'importe quelle taille et se trouver n'importe où dans l'image). **[23]**

Introduction générale :

Le travail décrit dans ce mémoire s'intègre dans le projet de recherche au niveau du CDTA (Centre de Développement des Technologies Avancées) d'Alger, intitulé : Navigation et Contrôle d'un Robot Mobile Autonome dans un Environnement Dynamique (Véhicule Intelligent pour le Transport Urbain).

La tâche qu'on doit réaliser est devisée en deux parties : la première partie est le développement d'un programme sous LabVIEW capable de détecter et reconnaitre les panneaux de signalisation routière à partir des images issues d'une caméra embarquée sur le robot véhicule « ROBUCAR ». La deuxième partie sert à traiter les détections en précisant les actions sur le Robucar.

La difficulté à laquelle on doit faire face dans le programme de reconnaissance des panneaux est la capacité à détecter et à reconnaitre les panneaux dans de différentes conditions, notamment les conditions d'éclairage (obscurité, forte luminosité ...), et le positionnement des panneaux.

Afin d'atteindre l'objectif de ce mémoire, on décompose le travail en une succession d'étapes. Premièrement, une recherche bibliographique sur les techniques de traitement d'images ainsi qu'une familiarisation avec l'outil LabVIEW sont nécessaires. Ensuite, nous allons rechercher une stratégie adéquate pour élaborer le programme de détection des panneaux et des feux de signalisation routières en employant les techniques de traitement d'images et en exploitant les fonction mises à notre disposition par le logiciel de programmation LabVIEW dédiées à la vision artificielle. Et enfin, nous allons élaborer un programme de traitement des détections en spécifiant au rebot les taches à exécuter pour chaque détection.

Le travail est devisé en cinq chapitres. Le premier chapitre porte sur les notions et les techniques de traitement d'images. Le deuxième chapitre est consacré à la description de l'outil software et hardware employé dans ce projet. Le troisième chapitre sert à détailler la démarche d'élaboration du programme de détection. Le quatrième chapitre a pour but d'évaluer les différentes étapes de la détection. Et enfin, le cinquième chapitre est consacré à la définition des actions sur le Robucar.

La conclusion sera un résumé du travail, des résultats obtenus et des problèmes rencontrés avec quelques perspectives qui sont tracées en guise de travaux futurs sur ce sujet.

Chapitre I

Notions et techniques de traitement d'images

I.1 Introduction :

Théoriquement, le traitement d'images numériques se définit par l'ensemble d'approches, de méthodes, de techniques et d'outils dont l'ambition est de résoudre la majorité des problèmes qui peuvent se présenter lorsqu'il est nécessaire d'extraire et d'analyser de façon automatique les informations présentes dans une image.

Cependant, dans ce chapitre nous nous intéressons qu'aux notions et aux théories qui sont en relation avec notre travail et qui peuvent nous éclaircir la voie pour atteindre l'objectif de ce projet.

I.2 L'image :

L'image est une Représentation d'un objet par les arts graphiques ou plastiques : la sculpture, la photographie, le dessin, le film, …etc. [1]

Elle peut être décrite sous la forme d'une fonction *F(x, y)* de brillance analogique continue, définie dans un domaine borné, tel que *x* et *y* sont les coordonnées spatiales d'un point de l'image et *F* est une fonction d'intensité lumineuse et de couleur. Sous cet aspect, l'image est inexploitable par la machine, ce qui nécessite sa numérisation. [2]

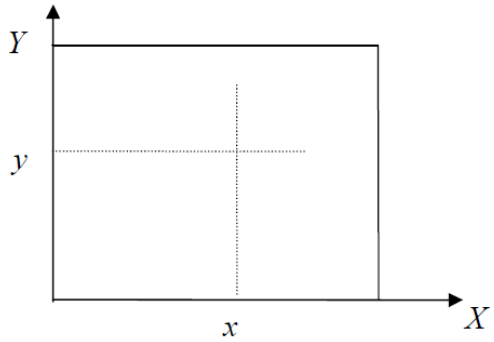


Figure I.1: un point de l'image de coordonnées *(x,y).* [2]

I.3 L'image numérique :

Contrairement aux images obtenues à l'aide d'un appareil photo ou dessinées sur du papier, les images manipulées par un ordinateur sont numériques (représentées par une série de bits). L'image numérique est l'image dont la surface est divisée en éléments de tailles fixes appelés cellules ou pixels, ou calculé à partir d'une description interne de la scène à représenter [2].

La numérisation d'une image est la conversion de celle-ci de son état analogique (distribution continue d'intensités lumineuses dans un plan *xOy*) en une image numérique représentée par une matrice bidimensionnelle de valeurs numériques *I(x,y)* où : *x, y* sont les coordonnées cartésiennes d'un point de l'image et *I(x,y)* le niveau de gris ou de couleur en ce point.

Cette conversion nécessite à la fois une discrétisation de l'espace : c'est l'échantillonnage, et une discrétisation des couleurs : c'est la quantification.

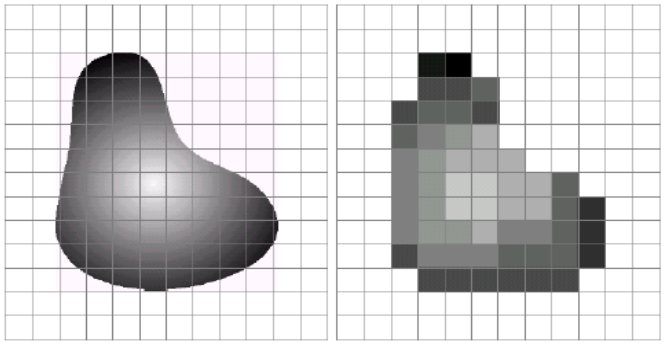


Figure I.2 : illustration de l'opération d'échantillonnage et de quantification d'une image

I.4 Types d'images :

Selon le nombre de bit sur lequel est codée la valeur de chaque pixel, on peut citer les catégories suivantes :

I.4.1 Image binaire :

C'est la représentation la plus simple d'une image, dans ce type d'image, chaque pixel est codé sur un bit : 0 pour le noir et 1 pour le blanc.

I.4.2 Image en niveau de gris :

Chaque pixel est codé sur 8bit, ce qui donne 2^8=256 valeurs possibles, ces valeurs représentent le niveau de gris : la valeur 0 représente le noir extrême et la valeur 255 représente le blanc extrême, les valeurs intermédiaires forment un dégradé du noir jusqu'au blanc.

I.4.3 Image couleur:

Chaque pixel possède une couleur décrite par la quantité de rouge (R), vert (G) et bleu (B). Chacune de ces trois composantes est codée sur l'intervalle [0, 255], ce qui donne 255^3 = 16 777 216 couleurs possibles. Il faut 24 bits pour coder un pixel.

I.5 Caractéristiques d'une image numérique:

L'image numérique est un ensemble structuré d'informations caractérisées par les paramètres suivants

I.5.1 Le Pixel :

Le Pixel est la contraction de l'expression anglaise "*picture element*". Le pixel est le point élémentaire d'une image, il peut être représenté par 1 bit (noir ou blanc) ou plus souvent sur 8 bit pour une image en niveaux de gris ou sur 16, 24 ou 32 bit pour une image couleur.

I.5.2 La dimension :

C'est la taille de l'image. Cette dernière se présente sous forme de matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image. [2]

I.5.3 La résolution :

C'est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre). [2]

I.5.4 Le bruit :

Un bruit (parasite) dans une image est considéré comme un phénomène de brusque variation de l'intensité d'un pixel par rapport à ses voisins, il provient de l'éclairage des dispositifs optiques et électroniques du capteur. [2]

Figure I.3: Exemple d'image bruitée.

I.5.5 L'histogramme :

L'histogramme des niveaux de gris ou des couleurs d'une image est une fonction qui donne la fréquence d'apparition de chaque niveau de gris (ou couleur) dans l'image. [2]

En d'autres termes, l'histogramme d'une image est une fonction qui assimile à chaque valeur de niveau de gris ou de couleur, le nombre de pixel qui ont cette valeur.

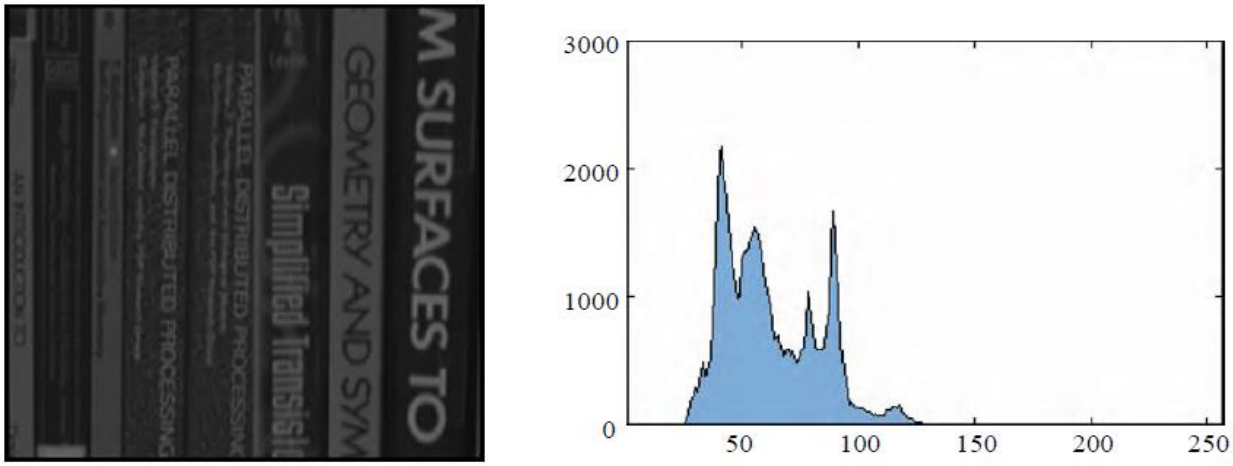


Figure I.4: Exemple d'une image (à droite) et son histogramme (à gauche) [3]

I.5.6 La luminance :

C'est le degré de luminosité des points de l'image. [2].

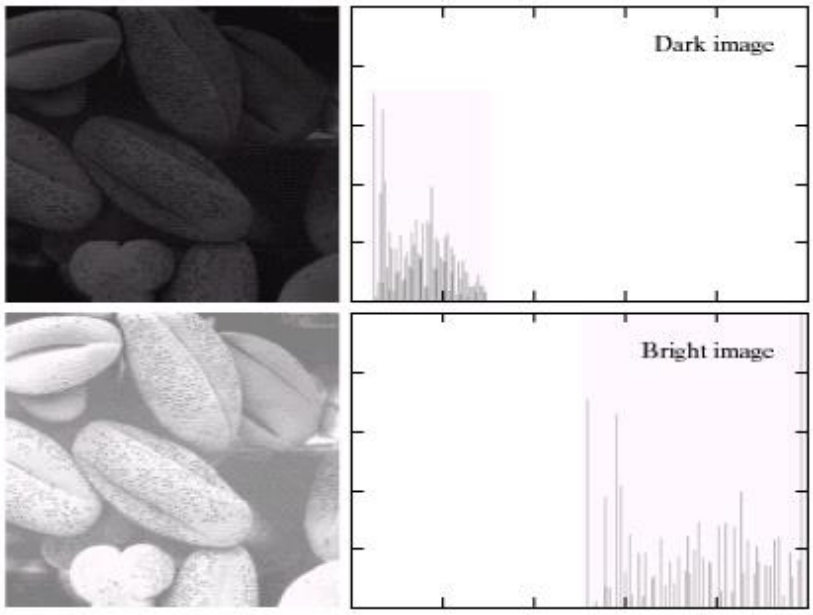


Figure I.5: Illustration de l'effet de la luminance sur l'image et sur son histogramme. [4]

I.5.7 Le contraste :

C'est l'opposition marquée entre deux régions d'une image, plus précisément entre les régions sombres et les régions claires de cette image. [2].

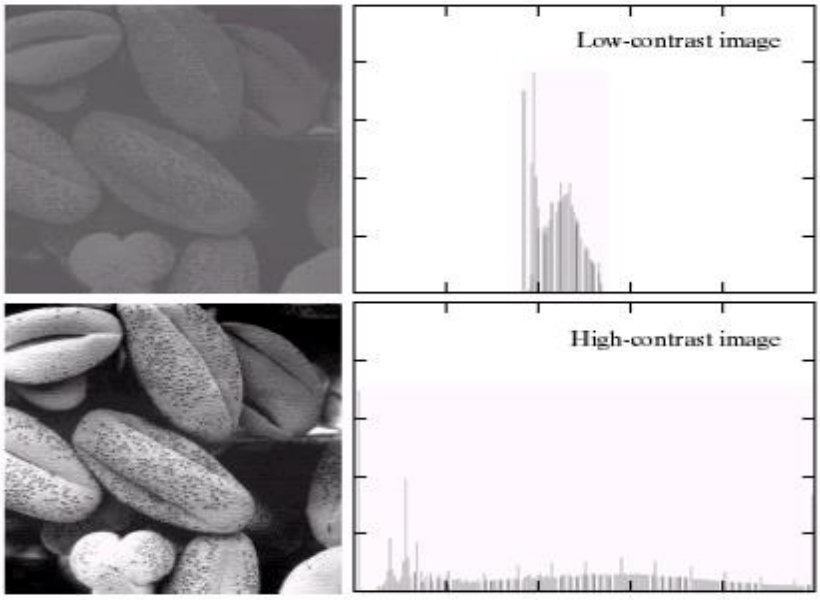


Figure I.6: Illustration de l'effet du contraste sur l'image et son histogramme. [4]

I.6 Le codage des images couleurs :

Les images numériques sont constituées de valeurs scalaires organisées en autant de plans que le système d'acquisition dont elles sont issues comporte de filtres ou de matrices de capteurs. Une image couleur possède ainsi trois plans qui correspondent à l'enregistrement de trois ondes réparties selon le modèle (rouge, vert, bleu). Cependant, l'image n'est pas nécessairement conservée dans ce format et de nombreux systèmes de codages ont été établis pour répondre à diverses nécessités technologiques ou algorithmiques. Tous ces systèmes ont en commun le codage de l'image en trois composantes obtenues par transformation des valeurs RGB initiales. Les espaces de représentation ainsi constitués sont des espaces vectoriels permettant la mesure des couleurs ; à ce titre, ils peuvent être qualifiés d'espaces colorimétriques, même si leur origine fondatrice n'est pas la colorimétrie. [4]

On appelle ainsi espace de couleurs la représentation mathématique d'un ensemble de couleurs. Il en existe plusieurs, nous présentons ici ceux qui sont utilisés dans ce projet.

I.6.1 Le modèle Rouge, Vert, Bleu (RVB) :

Le codage RVB (En anglais RGB : Red, Green, Blue), mis au point en 1931 par la *Commission Internationale de l'Eclairage* (CIE) consiste à représenter l'espace des couleurs à partir de trois rayonnements monochromatiques de couleurs :

- rouge (de longueur d'onde égale à 700,0 nm).
- vert (de longueur d'onde égale à 546,1 nm).
- bleu (de longueur d'onde égale à 435,8 nm).

Cet espace est basé sur le fait que toutes les couleurs peuvent être définies par une combinaison de rouge, vert et bleu. Il correspond à la façon dont les couleurs sont généralement codées informatiquement, ou plus exactement à la manière dont les tubes cathodiques des écrans d'ordinateurs représentent les couleurs.

Ainsi, le modèle RGB propose de coder sur un octet chaque composante de couleur, ce qui correspond à 256 intensités de rouge (2^8), 256 intensités de vert et 256 intensités de bleu, soient 16777216 possibilités théoriques de couleurs différentes, c'est-à-dire plus que ne peut en discerner l'œil humain (environ 2 millions). Toutefois, cette valeur n'est que théorique car elle dépend fortement du matériel d'affichage utilisé.

Par exemple dans un espace contenant 16,77 millions de couleurs, le rouge "pur" est défini par les paramètres suivants : rouge = 255, vert = 0, bleu = 0. Pour le noir "pur", toutes ces valeurs sont à 0, alors que pour le blanc, elles sont toutes à 255. Le modèle RVB est dit de "synthèse additive des couleurs".

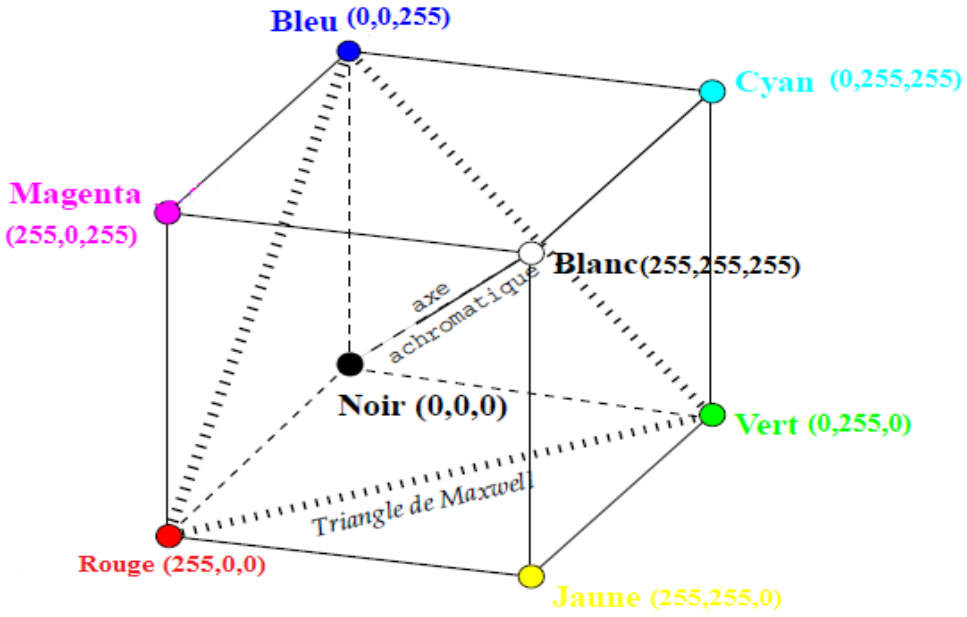


Figure I.7 : Représentation de la couleur dans l'espace RGB.

I.6.2 Le modèle HSV : [7]

Le principe de l'espace HSV (En anglais hue, saturation, value) est de caractériser les couleurs de façon plus intuitive, conformément à la perception naturelle des couleurs. Il est représenté par les trois composantes suivantes:

- **teinte:** intuitivement, c'est le nom qu'on utilisera pour désigner la couleur, "vert", "mauve", "orange", etc. Idéalement associé à une longueur d'onde, donc à une position sur le cercle de Newton.
- **saturation:** c'est le taux de pureté de la couleur, qui doit varier entre la pureté maximale (couleur éclatante) et l'achromatisme (niveau de gris).
- **Valeur:** c'est la mesure de l'intensité lumineuse de la couleur, qui doit varier entre le noir absolu et le blanc.

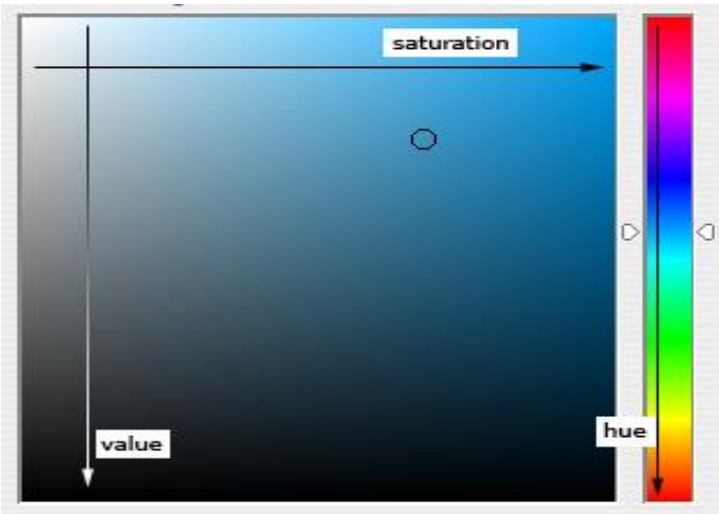


Figure I.8 : Représentation de la couleur dans l'espace HSV.

Le passage de RGB à HSV se fait par une transformation non linéaire. Plusieurs opérateurs ont été proposés pour la conversion. Ceci est un exemple :

$$v = \frac{r+g+b}{3}$$

$$s = 1 - \frac{3\min(r,g,b)}{r+g+b} \qquad \text{(I.1)}$$

$$h = \begin{cases} \theta & si \quad b \le g \\ 2\pi - \theta & si \quad b > g \end{cases} \qquad \theta = \arccos\left(\frac{(r-g)+(r-b)}{2\sqrt{(r-g)^2+(r-b)(g-b)}}\right)$$

Une autre transformation définie par les équations suivantes :

$$v = \frac{r+g+b}{3}$$

$$s = \begin{cases} \frac{3}{2}(M-v) & si\ v \ge med \\ \frac{3}{2}(v-m) & si\ v \le med \end{cases} \quad \left| \begin{array}{c} M = \max(r,g,b) \\ m = \min(r,g,b) \\ med = mediane(r,g,b) \end{array} \right. \qquad \text{(I.2)}$$

$$h = \frac{\pi}{3}\left(\lambda + \frac{1}{2} - (-1)^{\lambda}\frac{M+m-2med}{v}\right) \qquad \lambda = \begin{cases} 0\ si\ r \ge g \ge b\ ;\ 1\ si\ g \ge r \ge b \\ 2\ si\ g \ge b \ge r\ ;\ 3\ si\ b \ge g \ge r \\ 4\ si\ b \ge r \ge g\ ;\ 5\ si\ r \ge b \ge g \end{cases}$$

Il existe d'autres modèles naturels de représentation proches du modèle HSV :

- **HSB** : *Hue, Saturation, Brightness* soit *Teinte, Saturation, Brillance* en français. La brillance décrit la perception de la lumière émise par une surface.
- **HSI** : *Hue, Saturation, Intensity* soit *Teinte, Saturation, Intensité* en français.
- **HCI** : *Hue, Chrominance, Intensity* soit *Teinte, Chrominance, Intensité.*

I.7 La segmentation d'images :

La segmentation représente l'étape la plus importante en traitement d'images, elle consiste à partitionner l'image en zones homogènes ayant des caractéristiques (niveau de gris, couleur, texture) identiques, où une zone peut correspondre à un objet ou une partie d'un objet.

La définition mathématique de la segmentation est donnée par Zucker de la manière suivante : [9]

Segmenter une image I en n régions revient à la partitionner en n sous-ensembles R_1, R_2, …, R_n tels que :

1. $I = \bigcup_i R_i$
2. R_i est constitué de pixels connexes pour tout i.
3. $P(R_i)=vrai$ pour tout i.
4. $P(R_i \cup R_j)=faux$ pour tous i,j R_i et R_j étant adjacentes dans I.

Il n'existe pas de théories générales s'appliquant à différents types d'images, mais plutôt des méthodes variées que l'on choisit et que l'on développe pour résoudre des problèmes d'analyse sur un type d'image bien défini.

I.7.1 Approches de segmentation d'images :

I.7.1.1 Segmentation par région :

Ces approches cherchent à regrouper directement des pixels ayant des propriétés communes. Ce qui conduit à une région unique. L'ensemble des regroupements de pixels définit à la fin une segmentation de l'image. Les techniques courantes de cette catégorie sont la croissance de régions, la fusion et la division et la division-fusion. [8]

I.7.1.2 Segmentation par contour :

Ces approches se basent sur la recherche des contours qui délimitent les régions de l'image. Un contour est représenté par l'ensemble de pixels formant une frontière entre deux ou plusieurs régions voisines il est défini par une variation rapide du niveau de gris, de couleur ou de texture. Ces approches emploient plusieurs méthodes qu'on peut regrouper en trois catégories : les méthodes dérivatives, par filtrage optimal et les contours actifs.

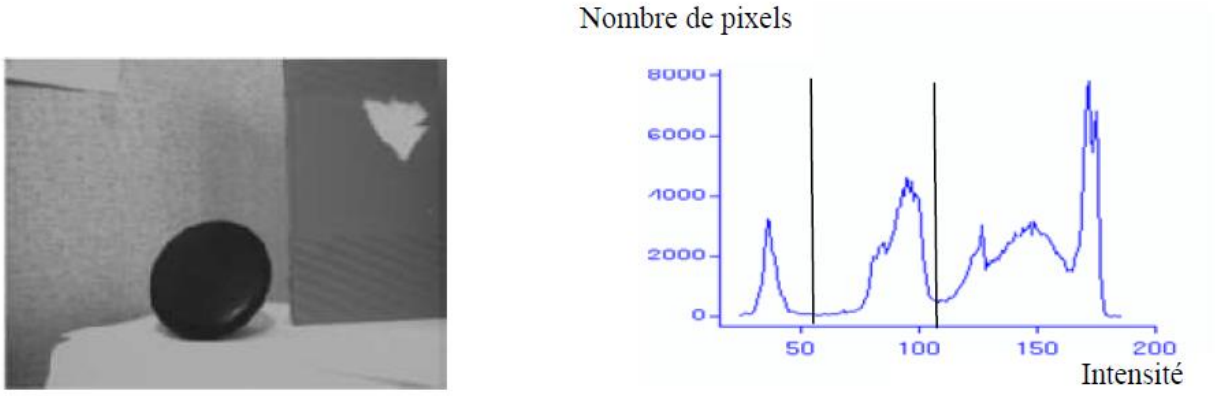
I.7.1.3 Segmentation hybride (ou par coopération) :

La segmentation par coopération région contour peut être exprimée comme une entraide entre ces deux concepts afin d'améliorer le résultat final, cette coopération contribue à une meilleure prise en compte des caractéristiques des entités de l'image en exploitant une coopération au niveau des algorithmes. [8]

I.7.2 La segmentation par seuillage d'histogramme :

Elle constitue un cas particulier de la segmentation par classification qui appartient aux approches de segmentation par régions. Elle permet de classer les pixels selon leur niveau de gris.

La segmentation par seuillage consiste à repartir les pixels en K classes ($C_1, C_2, \ldots, C_K$) à partir d'un ensemble de seuils $T=\{t_1, t_2, \ldots, t_{K-1}\}$. Un pixel de coordonnées (x,y) et de niveau de gris $l(x,y)$ est affecté à la classe C_K si $t_k \leq l(x,y) \leq t_{k+1}$ avec $k=0, 1, \ldots, K-1$. (voir la figure I.9)



a) Image original (b) Histogramme

Figure I.9 : segmentation par seuillage d'histogramme. [8]

I.8 Introduction à la reconnaissance de formes en traitement d'images:

La reconnaissance de formes (ou de motifs) occupe une place très importante en traitement d'images, Elle est définie comme l'ensemble des techniques et méthodes visant à identifier un motif parmi d'autres à partir de données informatiques brutes (une image), elle ne se limite pas à la reconnaissance des formes géométriques mais plutôt de motifs qui peuvent être de nature très variée. Il peut s'agir de contenu visuel (code barre, visage, empreinte digitale...), d'images médicales (rayon X, EEG, IRM...) ou multi spectrales (images satellitaires) et bien d'autres.[10] La Reconnaissance de Forme (RdF) est un thème particulièrement vaste et pluridisciplinaire qui fait l'objet de nombreux travaux de recherches. [11]

I.8.1 Les approches de la reconnaissance de formes :

Une recherche bibliographique dans le domaine de la RdF montre qu'il n'y a pas une théorie unifiée pour cette discipline. Cependant, il existe deux approches principales: l'approche basée sur la théorie statistique de la décision et l'approche structurelle.

I.8.1.1 Approche statistique :

Une approche classique en RdF est fondée sur l'étude statistique des mesures que l'on a effectuées sur les objets à reconnaitre. L'étude de leur répartition dans un espace métrique et la caractérisation statistique des classes permet de prendre une décision de reconnaissance du type « plus forte probabilité d'appartenance à une classe ». Ces méthodes s'appuient en général sur des hypothèses concernant la description statistique des familles d'objets analogues dans l'espace de représentation.

Dans cette approche, on supposera donc que les mesures faites sur une forme peuvent s'exprimer sous la forme d'un vecteur $X=(x_1,\ldots\ldots\ldots\ldots, x_n)$ de l'espace R^n. On dispose d'un ensemble d'apprentissage, c'est-à-dire d'un jeu de tels vecteurs dont on connaît en plus la classe d'appartenance. Le problème peut ainsi se résumer sommairement : Etant donné un vecteur inconnu, obtenu par mesures sur une forme, trouver la classe à laquelle on doit l'affecter. Autrement dit, reconnaître la forme inconnue en fonction de l'apprentissage effectué. [11]

I.8.1.2 Approche structurelle :

Si l'approche statistique permet de se placer dans un cadre mathématique solide et général, elle présente néanmoins le défaut d'oublier la nature des mesures qui sont faites sur les formes et

de les traiter de façon abstraite. On conçoit cependant qu'il est plus simple et plus riche d'utiliser des paramètres descriptifs liés à la nature même des formes étudiées. Si l'on possède une technique de suivi de contour (ce qu'il est facile d'imaginer dans les images à deux niveaux de gris) et un détecteur de segment, on voit que tous les triangles par exemple peuvent se décrire par une caractéristique du type : ''la frontière est composée d'un segment horizontal de longueur *l*, suivi d'un segment oblique de longueur à peu près *l*, suivi d'un segment oblique de longueur à peu près *l*, qui se termine au point de départ de premier''

L'intérêt d'une telle description est d'être universelle pour tous les triangles équilatéraux, indépendamment de leur taille et de leur position dans l'image. Elle décrit ces figures, d'une part en définissant des formes élémentaires (segments de droites dans la frontière) et, d'autre part, par l'assemblage de ces formes élémentaires pour constituer la figure totale. On dit qu'une telle description est de type structurel, puisqu'elle s'attache plutôt à définir les caractéristiques intrinsèques de la forme qu'à donner sa description métrique (qui ne se révèle pas riche dans un tel exemple). [11]

I.8.2 Le processus de reconnaissance de formes :

Le schéma général de la reconnaissance de forme est illustré dans la figure suivante :

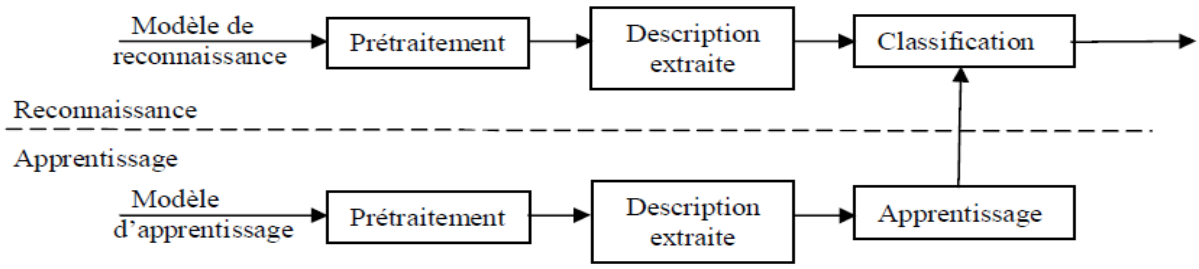


Figure I.10 : processus de reconnaissance de formes. [11]

I.8.2.1 Prétraitement :

Le contenu d'une image est représenté sur ordinateur par une matrice de grandes dimensions, une ou plusieurs opérations de prétraitement sont nécessaires pour faciliter la représentation de de la forme à reconnaitre soit en nettoyant la forme pour éliminer le bruit ou en réduisant la quantité d'information à traiter pour ne garder que les informations les plus significatives.

Le prétraitement est basé sur les différentes techniques de traitement d'images.

I.8.2.2 Extraction des caractéristiques et primitives :

L'extraction des caractéristiques ou primitives, en anglais, ''feature extraction'' transforme la forme d'origine, en une description numérique ou symbolique dans un espace abstrait, selon un formalisme prédéfini. Les attributs à extraire doivent être discriminantes le plus possible et répondent aux exigences suivants [11]:

Une faible variance intra classe, une grande variance inter classe, un faible nombre d'attributs et l'indépendance de la translation, de la rotation et le facteur d'échelle.

I.8.2.3 L'apprentissage :

L'apprentissage consiste à donner au classifieur plusieurs exemples de formes et à lui indiquer à chaque fois de quelle forme il s'agit. Tous ces exemples sont placés dans ce que l'on appelle ''une base d'apprentissage''

Le rôle du module d'apprentissage consiste à caractériser chaque classe par les paramètres définissant la forme (c'est la détermination des espaces d'appartenance des individus).

Supposons que des formes soient définies par des vecteurs de R2 (donc deux paramètres réels) et supposons que nous disposons d'échantillons de deux classes d'objets. On peut alors représenter ces échantillons dans l'espace R2 des paramètres (voir figure I.12). L'apprentissage peut alors par exemple consister à trouver automatiquement une courbe séparant les échantillons de chacune des deux classes. C'est la recherche de cette courbe qui va être l'apprentissage des deux classes.

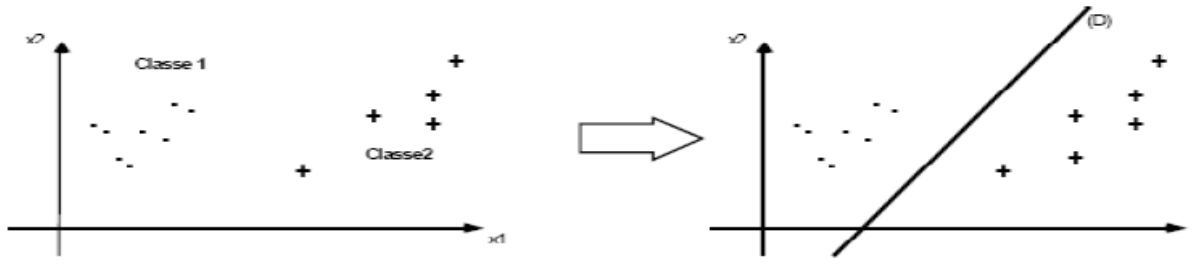


Figure I.11 : Exemple d'apprentissage. [11]

I.8.2.4 Classification :

La classification sert à attribuer à un élément (objet, forme … etc.) une classe précise parmi plusieurs classes définis à l'avance. Un algorithme qui réalise automatiquement une classification est appelé classifieur.

Lorsque les données sont étiquetées, la reconnaissance consiste à décider de l'étiquette à attribuer à tout nouvel individu en fonction de sa position dans l'espace des représentations.

I.8.3 Les paramètres caractéristiques: Nous présentons dans cette section quelques paramètres utilisés pour la reconnaissance de forme, pour illustrer l'utilisation de ces paramètres, nous allons présenter des exemples appliqués sur des chiffres. Les attributs à extraire doivent être discriminant le mieux possible.

I.8.3.1 Détection des cavités : [12]

On définit 5 types de cavités: Ouest, Nord, Sud, Est et Centrale. Cela suppose la définition de 4 directions cardinales dans l'image : Ouest vers la gauche, Nord vers le haut, etc.

Par exemple, un point de l'image appartient à une cavité Ouest si est seulement si les conditions suivantes sont simultanément vérifiées :

- Ce point n'appartient pas au tracé.
- A partir, de ce point, en se déplaçant en ligne droite vers l'Ouest, on ne croise pas le tracé.
- A partir de ce point, en se déplaçant en ligne droite vers le Nord, le Sud ou l'Est, on croise le tracé.

Par analogie, on peut facilement en déduire les définitions des cavités Nord, Sud, Est. Par contre un point de l'image appartient à une cavité centrale si et seulement si les conditions suivantes sont simultanément vérifiées :

- Le point n'appartient pas au tracé.
- A partir de ce point, en se déplaçant en ligne droite vers l'Ouest, le Nord, le Sud, ou l'Est, on croise le tracé.

On montre dans la figure (I.12) les cavités détectées sur les chiffres. Chaque type de cavité est représenté par une couleur (Est : rouge, Ouest : vert, Sud : bleu, Centrale : violet).

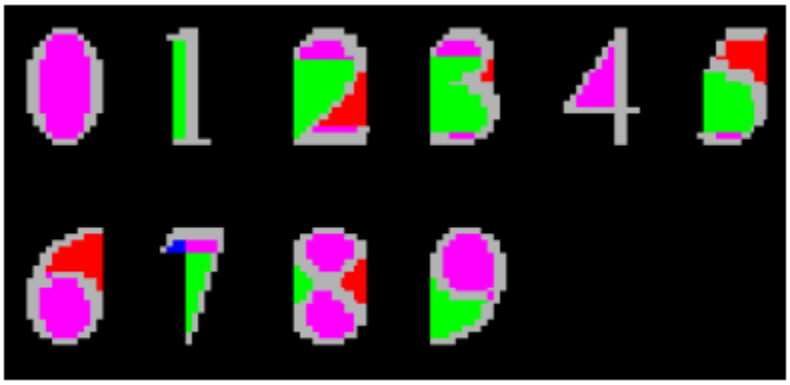


Figure I.12: les différentes cavités. [11]

A partir de l'image d'un chiffre, le programme calculait les images CE, CS, CO, CN, et CC, représentant respectivement les cavités Est, Sud, Ouest, Nord, et Centrales.
On utilise comme paramètres caractéristiques, les surfaces relatives des différents types de cavités. Par exemple, notons S_{ce} la surface relative des cavités Est et S_{CE} leur surface absolue (en nombre de pixels). On a alors :

$$S_{ce} = \frac{S_{CE}}{S_{CE}+S_{CW}+S_{CS}+S_{CN}+S_{CC}+1} \quad \text{(I.3)}$$

Une définition similaire est utilisée pour les surfaces relatives des autres cavités. L'ajout de 1 au dénominateur a en général une influence négligeable sur le résultat.
Son objectif est uniquement d'éviter une division par zéro dans le cas d'une forme qui ne comprendrait aucune cavité.

I.8.3.2 Les histogrammes de projection: [11]

Les histogrammes de projection sont depuis très longtemps utilisés dans le domaine de la reconnaissance de formes. Leur principe est de sommer le nombre de pixels noirs de chacune des lignes (respectivement des colonnes) de l'image binaire de la forme.

Les projections des histogrammes ont été introduites en 1956 par Glauberman, cette technique est principalement utilisée pour la segmentation des caractères, des mots et de lignes de texte, ou pour détecter si une image d'un texte numérisé est tournée.
La projection horizontale (respectivement verticale) est le nombre de pixel de chaque ligne (respectivement colonne). Elles peuvent être indépendantes de l'échelle en divisant par le nombre total de pixels imprimés en caractère d'image. Toutefois, les projections d'histogrammes sont très sensibles à la rotation et, dans une certaine mesure, la variabilité dans le style d'écriture. En outre, des informations importantes sur le caractère forme semble être perdu.

Figure I.13: Histogramme horizontale et verticale d'un panneau. [22]

I.8.3.3 Zoning (densités) :

C'est la densité de pixels appartenant à la forme (motif ou chiffre) calculée dans différentes zones (zoning) de l'image. Pour obtenir ces mesures, on découpe horizontalement et verticalement le rectangle englobant la forme en zones de taille égale; Le nombre de pixels appartenant à la forme dans chaque zone forme alors les composantes du vecteur de caractéristiques. En découpant par exemple l'image en *n* zones verticales (égales en largeur) et *m* zones horizontales (égales en hauteur), on obtient un vecteur à *n*x*m* composantes. [11]
Les densités dans chaque zone devront être normalisées (en les divisant par exemple par la surface de la zone-moyenne-) puisque les chiffres ne sont pas tous de même taille.

I.8.3.4 Profils :

Les caractéristiques à extraire de la forme sont leurs profils gauche et droite. Pour obtenir ces mesures, on détermine sur un certain nombre de lignes, en général, réparties uniformément sur la hauteur de la forme, la distance entre le bord gauche (respectivement. droite) du motif et le

premier pixel appartenant au motif rencontré sur cette ligne; L'ensemble de ces distances définit un profil gauche (respectivement. droit) de la forme. [11]

Les profils gauche et droit doivent être normalisés (en les divisant par exemple par la largeur de l'image du chiffre traité) puisque les chiffres ne sont pas tous de même taille.

I.8.3.5 Les moments géométriques : [13]

Les moments sont souvent utilisés en physique pour décrire la répartition des masses dans un corps. En associant le niveau de gris d'un point de l'image à la masse d'un corps en chaque point, on peut reprendre le même formalisme pour décrire la répartition des niveaux de gris dans un objet. Dans le cas d'un objet binaire où *g(k,l)* prend la valeur '1' à l'intérieur (si le point appartient à l'objet) et '0' à l'extérieur, les différents moments fournissent des informations importantes concernant l'arrangement spatial des points de l'objet.

L'équation générale des moments est donnée par :

$$M_{i,j} = \sum_{k=1}^{K} \sum_{l=1}^{L} (k)^i (l)^j g(k,l) \qquad \text{(I.4)}$$

où *g(k,l)* est le niveau de gris de l'image défini pour $k=1,\dots,K$ et $l=1,\dots,L$.

Le principe d'utilisation des moments est de sélectionner un sous-ensemble de $M_{i,j}$ qui contienne suffisamment d'information pour caractériser l'image de façon unique, pour une application donnée.

Le centre de gravité d'une image (x_g ,y_g) est défini par les relations suivantes :

$$x_g = \frac{M_{1,0}}{M_{0,0}} \qquad \text{(I.5)}$$

$$y_g = \frac{M_{0,1}}{M_{0,0}} \qquad \text{(I.6)}$$

$M_{0,0}$ représente la surface de la forme dans le cas d'une image binaire.

Afin de rendre les moments $M_{i,j}$ invariants par la translation, on les définit en choisissant (x_g ,y_g) comme origine, on parle alors de moments centrés $\mu_{i,j}$ dont la définition formelle est la suivante :

$$\mu_{i,j} = \sum_{k=1}^{K} \sum_{l=1}^{L} (k - x_g)^i (l - y_g)^j g(k,l) \qquad \text{(I.7)}$$

Les moments centrés peuvent se calculer au moyen de la relation précédente ou en fonction des moments non centrés par la relation:

$$\mu_{ij} = \sum_{r=0}^{i} \sum_{s=0}^{j} C_r^i C_s^j \left(-x_g\right)^r \left(-y_g\right)^s M_{i-r,j-s} \quad \text{(I.8)}$$

Avec :

$$C_b^a = \frac{a!}{b!(a-b)!} \quad \text{(I.9)}$$

Il est donc assez facile d'exprimer les moments centrés en termes de moments géométriques.

En divisant chaque moment μ_{ij} par $(\mu_{00})^{(i+j+2)/2}$ on construit les moments normalisés η_{ij} qui sont invariants par homothétie (changement d'échelle, grossissement).

$$\eta_{i,j} = \frac{\mu_{i,j}}{\mu_{00}^{(i+j+2)/2}} \quad \text{(I.10)}$$

Les 7 moments de Hu sont définis à partir des moments normalisés η_{ij} d'ordre 2 et 3, qui sont invariants en translation, échelle et rotation [13]:

$$\phi_1 = \eta_{20} + \eta_{02} \quad \text{(I.11)}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \quad \text{(I.12)}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad \text{(I.13)}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \quad \text{(I.14)}$$

$$\phi_5 = (\eta_{30} - 3\eta_{12}(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] + (3\eta_{21} - \eta_{03}) + (\eta_{21} + \eta_{03})[3(\eta_{30} - \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] \quad \text{(I.15)}$$

$$\phi_6 = (\eta_{20} + \eta\eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{03} + \eta_{12})(\eta\eta_{21} + \eta_{03}) \quad \text{(I.16)}$$

$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12}[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] - (\eta_{30} + 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \quad \text{(I.17)}$$

Les six premiers moments sont invariants aux translations et aux changements, quand il s'agit de reconnaitre des images « miroirs » on exploite le moment sept qui n'est pas invariant aux réflexions. Ce dernier change de signe lorsqu'une telle transformation est appliquée à l'image et permet donc de détecter celle-ci.

I.8.4 La classification :

La classification est l'étape où se réalise la reconnaissance. Il existe plusieurs méthodes pour la classification, nous présentons ci-après celles utilisée dans notre application

I.8.4.1 La méthode de K plus proches voisins :

C'est une méthode qui a l'avantage d'être extrêmement simple et offre généralement de bons résultats.

I.8.4.1.1 Principe de la méthode :

Etant donnée une base d'apprentissage d'images, pour prédire la classe d'un nouveau cas, le classifieur K-PPV cherche les K plus proches voisins de ce nouveau cas et prédit la réponse la plus fréquente de ces K plus proches voisins. La méthode utilise donc deux paramètres : le nombre K et la fonction de similarité pour comparer le nouveau cas aux cas déjà classés. Le principe est résumé par ces étapes:

1. Choisir un nombre entier K (souvent k = nombre d'attributs + 1).
2. Calculer les distance entre le nouveau vecteur et les vecteurs de la base d'apprentissage, on utilise souvent la distance euclidienne.
   La distance euclidienne entre deux point de coordonnées ($x_1, x_2, \ldots, x_n$) et ($y_1, y_2, \ldots, y_n$) se calcule à partir de la formule suivante :

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad \text{(I.18)}$$

3. Retenir les K vecteurs de la base d'apprentissage les plus proches du nouveau vecteur en appliquant la notion de la distance.
4. Déterminer les classes correspondantes des K vecteurs.
5. Choisir la classe la plus représentée.

I.8.4.1.2 Le choix du nombre K :

Le choix du paramètre K dans la règle des KPPV a une influence directe sur le style du classifieur ainsi déterminé. Une faible valeur de K va donner un classifieur de bonne résolution (définir des frontières compliquées entre classes) mais très sensible au bruit sur les échantillons et sur le vecteur à classer. Une valeur grande de K aura un comportement inverse, lissant les frontière mais peu sensible au bruit [11].

I.8.4.1.3 Critique de la méthode de K-PPV :

Cette méthode à l'avantage d'être extrêmement simple, et de donner en général de bons résultats. Mais son inconvénient majeur reste le fait que le temps de calcul devient extrêmement important lorsque la base d'apprentissage contient beaucoup d'exemples car l'algorithme doit parcourir toute la base d'apprentissage pour chaque nouveau cas à classifier [11].

I.8.4.2 La méthode de distance moyenne minimale (minimum mean distance)

C'est l'une des plus simples techniques pour la classification, sont principe est défini comme suit :

Chaque classe ω_i apprise est représentée par un vecteur calculé par la formule suivante :

$$m_i = \frac{1}{N_i} \sum_{x \in \omega_i} x \qquad \text{(I.19)}$$

N_i : nombre d'échantillons de la classe ω_i

En se basant sur cette formule, pour chaque nouvelle forme représentée par son vecteur caractéristique *x*, on calcule les distances entre le vecteur *x* et les vecteurs m_i, ainsi le nouveau échantillon sera affecter à la classe dont la distance séparant les vecteur *x* et m_i présente la valeur minimale.

En pratique, cette méthode est efficace quand les distances séparant les moyennes m_i sont larges comparées à la distribution des éléments de chaque classe par rapport à la moyenne.

I.9 Conclusion:

Nous avons présenté au début de ce chapitre les notions de base et les caractéristiques liées aux images numériques, ensuit nous avons présenté les différentes techniques employées pour le traitement, l'analyse et l'extraction d'informations contenues dans une image telles que le seuillage, la segmentation, la reconnaissance de formes et la classification, ainsi que les approches et les méthodes employées dans ces techniques.

On constate que le domaine de traitement d'images est tellement riche qu'il est impossible de rapporter toutes les méthodes concernant les techniques citées.

Chapitre II

Présentation de l'environnement LabVIEW et du contrôleur CompactRIO

II.1 Introduction :

Ce chapitre est consacré à la présentation de l'outil software et hardware servant à la mise en œuvre de notre projet. Nous commençons par décrire l'outil de programmation LabVIEW et ces programmes périphériques notamment ceux qui servent au traitement d'images (ou vision artificielle). Ensuite, nous présentons le dispositif de CompactRIO. Et enfin nous allons donner une description du robot ''ROBUCAR'' sur lequel le travail de ce projet est implémenté.

II.2 Le langage LabVIEW :

LabVIEW (contraction de **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) est le cœur d'une plate-forme de conception de systèmes de mesure et de contrôle, basée sur un environnement de développement graphique.

Le langage graphique utilisé dans cette plate-forme est appelé « G ». Créé à l'origine sur Apple Macintosh en 1986, LabVIEW est utilisé principalement pour la mesure par acquisition de données, pour le contrôle d'instruments et pour l'automatisme industriel. La plate-forme de développement s'exécute sous différents systèmes d'exploitation comme Microsoft Windows, Linux et Mac OS X. LabVIEW peut générer du code sur ces systèmes d'exploitation mais également sur des plates-formes temps réel, des systèmes embarqués ou des composants reprogrammables FPGA. [14]

II.3 Le langage G :

Un programme en langage G se présente comme un schéma, le diagramme, réunissant différentes icônes reliées par des fils de couleur. Chaque fil symbolise le passage d'une donnée

depuis une source dont elle sort (comme résultat), vers une cible où elle entre (comme paramètre).Les diagrammes du langage G ont donc une signification bien différente de celle des schémas électroniques qu'ils évoquent parfois. Dans un diagramme LabVIEW, la donnée ne transite dans le fil qu'au moment où elle est générée par son icône source. L'icône cible ne commencera son exécution que lorsque toutes ses données d'entrée seront disponibles. Ce modèle d'ordonnancement par flots de données détermine l'ordre d'exécution des traitements du programme. Une conséquence importante de cette règle est que les traitements qui n'échangent pas de données sont libres de s'exécuter en parallèle. Cette propriété du langage G facilite le développement d'applications multiprocessus, particulièrement intéressantes dans le cadre du contrôle de systèmes réactifs (embarqués ou non). [14]

II.4 Structure du langage graphique LabVIEW :

Le diagramme de LabVIEW est lié à une interface utilisateur graphique nommée face-avant. Les programmes et sous-programmes sous forme d'icônes sont appelés des instruments virtuels (En anglais Virtual Instrument VI) et les fichiers source enregistrés sur disque ont l'extension de nom de fichier «.vi»

Chaque VI possède trois composants : un diagramme qui intègre le code graphique, une face-avant personnalisable par l'utilisateur et un panneau de connexions pour les icônes qui sert d'entrées/sorties pour les variables sous forme de fils.

Une fois un VI soit édité, il devient une icône pour un programme de plus haut niveau et pourrait ainsi intégré dans le nouveau diagramme. Il devient alors un sous-VI appelé donc un sous-programme. Chaque icône de sous-VI créé peut être personnalisée par un dessin représentant au plus près sa fonction.

La face-avant est construite en utilisant des objets dénommés commandes et indicateurs. Les commandes sont des entrées qui servent à saisir des valeurs à l'écran et les indicateurs sont des sorties qui servent à afficher des variables ou des résultats de calculs. Tous les objets de la face-avant (commandes et indicateurs) apparaissent sur le diagramme afin de les relier aux fonctions.

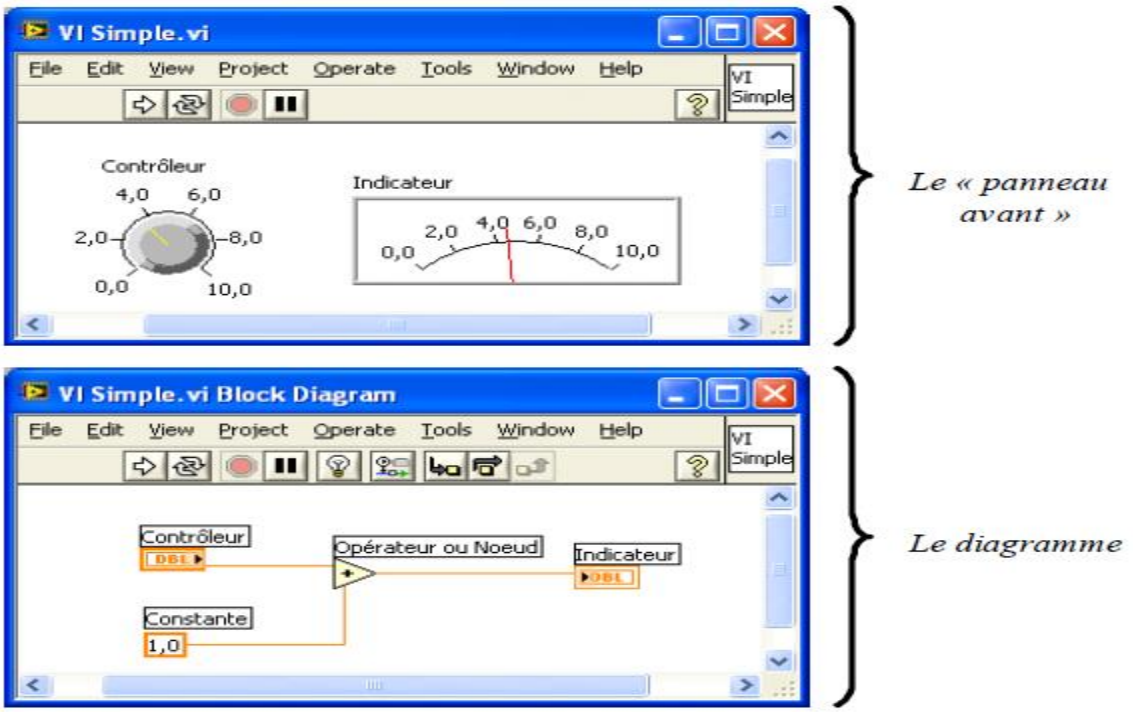


Figure II.1 : présentation de l'interface avant et de l'interface diagramme d'un VI sous LabVIEW.

II.5 Quelques structures et fonctions de base :

La programmation graphique permet d'avoir une vision claire sur le programme, le tableau (II.1) présente quelques éléments basiques de la programmation sous LabVIEW.

Objet	**nom**	**description**
	Boucle While	Permet d'exécuter en boucle les fonctions mises à l'intérieur du rectangle tant qu'une condition est vérifiée. Cette condition est connectée par un fil au cercle rouge en bas du rectangle.

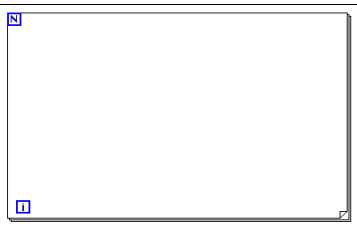
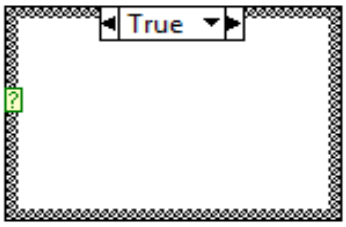
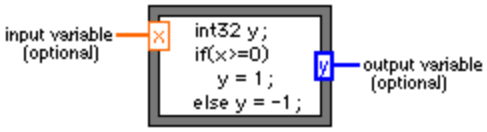
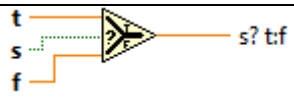
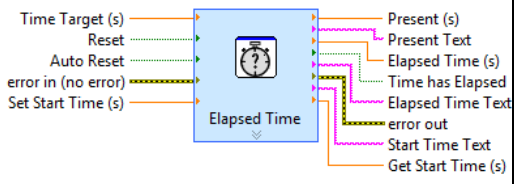
	Boucle For	Permet d'exécuter le programme mis à l'intérieur pour un nombre d'itérations déterminé. Ce nombre N doit être spécifié
	Structure Case	Elle a un ou plusieurs sous diagrammes contenant des programmes, le programme à exécuter est déterminée par la valeur connecté au point de sélection.
	Formula node	Permet d'évaluer une expression mathématique ou un algorithme en langage C.
	Fonction de sélection	Renvoie la valeur connectée à **t** ou à **f** selon la valeur de la variable booléen connecté à **s**.
	Temps écoulé	Indique le temps écoulé depuis l'exécution d'une tache spécifiée.

Tableau II.1 : Description de quelques éléments de programmation sous LabVIEW.

II.6 Acquisition et traitement d'images sous LabVIEW :

II.6.1 Acquisition d'images :

Il existe deux moyens pour l'acquisition d'images sous LabVIEW : soit on charge les images depuis le disque dure, soit on les obtient depuis une caméra. Pour le deuxième cas, la tâche est simplifiée par l'outil NI MAX de LabVIEW qui permet de configurer les paramètres d'acquisition tels que le choix de la source d'acquisition, le type d'image et sa résolution, le nombre de frames par seconde (fps) pour le cas d'une vidéo…etc.

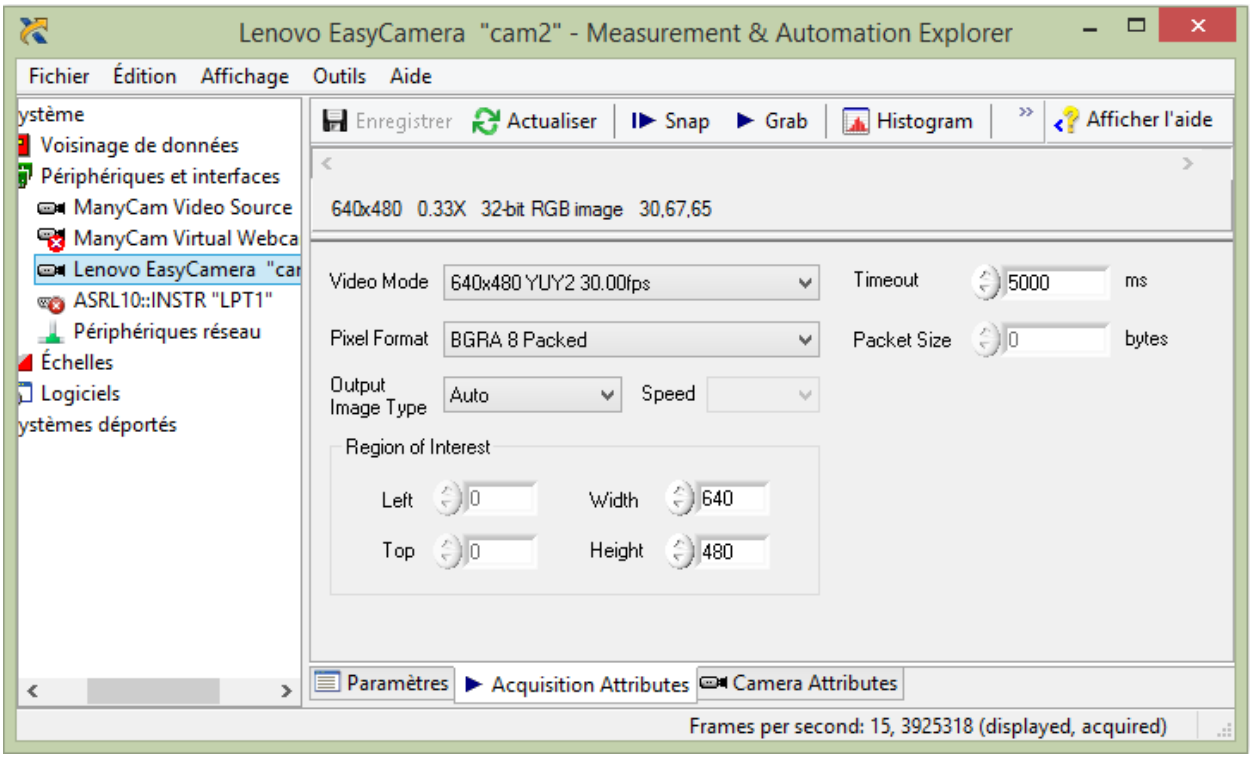


Figure II.2 : configuration de l'acquisition d'image sous le programme NI MAX.

LabVIEW offre quatre possibilités d'acquisition et de traitement d'images :

a) Une seule acquisition avec traitement :

Cette acquisition est utilisée pour acquérir une seule image. On n'a pas besoin d'une boucle dans le programme.

b) Acquisition continue avec traitement en ligne :

Ce type est utilisé pour une acquisition continue d'images. Le temps moyen du traitement doit être inférieur au temps d'acquisition pour ne pas perdre quelques images, dans ce cas on a deux choix : soit on acquiert juste les images récentes (d'autres images sont perdues), ou bien on choisit d'acquérir toutes les images et on fixe le nombre d'images qu'on met en mémoire temporaire.

c) Acquisition complète avec traitement en ligne :

Ce type est utilisé pour acquérir un nombre fixe d'images, dès qu'une image est acquise, elle sera valable pour le traitement, ce type est utilisé si on veut afficher ou traiter l'image avant la prochaine acquisition.

d) Acquisition complète avec post-traitement:

Ce type est utilisé pour acquérir un nombre fixe d'images premièrement, les images ne seront valable pour le traitement qu'à la fin d'acquisition du nombre fixé d'images. Ce type est utilisé si le temps nécessaire pour le traitement est supérieur au temps nécessaire pour l'acquisition.

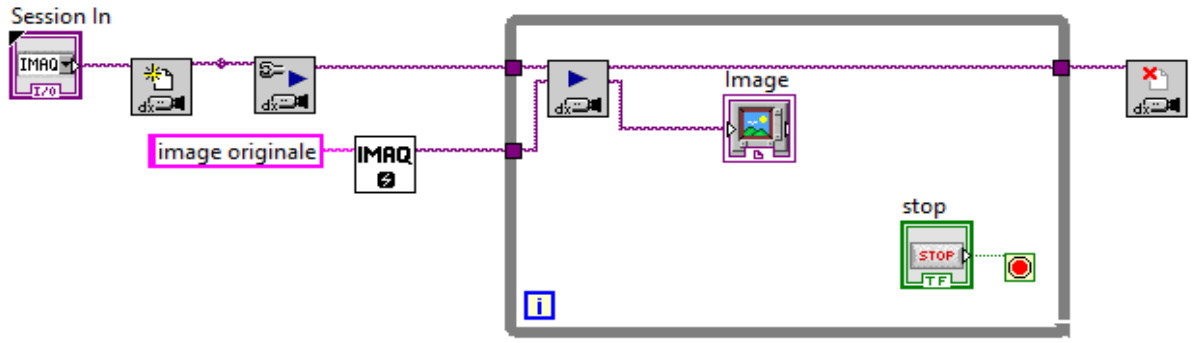


Figure II.3 : Diagramme d'une acquisition d'image continue sous LabVIEW.

II.6.2 traitement d'images sous LabVIEW :

La National Instrument fournit un module très riche pour le développement des applications basées sur le traitement d'images, il s'agit de deux softwares ''NI Vision assistant'' et ''NI IMAQ vision'' ces deux outils travaillent ensemble afin de simplifier le développement de programmes pour la vision artificielle.

II.6.2.1 NI Vision Assistant :

La conception d'une application pour la vision artificielle nécessite souvent un temps considérable d'expérimentation. L'environnement de l'outil Vision Assistant permet d'apprendre et de rechercher des stratégies pour le traitement d'images à travers des prototypes configurables qui ne nécessitent pas de la programmation de haut niveau. On peut citer d'autres avantages de cet outil :

- avoir des idées pour résoudre des problématiques de traitement d'images.

- Tester des différentes stratégies pour le traitement.
- Tester une stratégie particulière sur une variété d'images.
- Visualiser l'effet de certaines fonctions ainsi que le changement de leurs paramètres sur l'image.
- Générer des scripts en plusieurs langages.

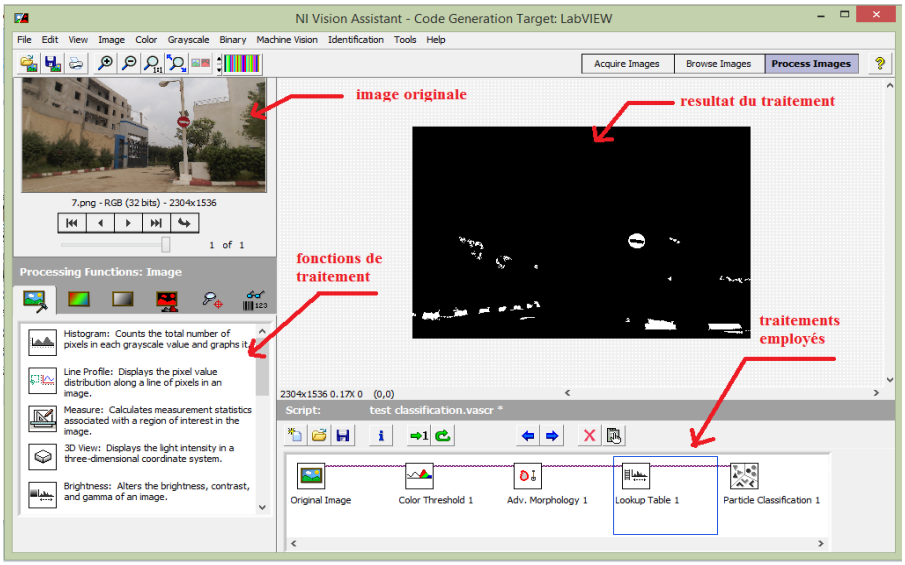


Figure II.4 : Environnement du logiciel Vision Assistant.

II.6.2.2 IMAQ Vision :

IMAQ Vision fournit à LabVIEW une centaine de fonctions avancées pour le traitement d'images ; des fonctions pour la manipulation des images couleur, en niveau de gris ou des images binaires, des fonctions pour le traitement (statistiques, filtrage, manipulations géométriques…), des fonctions pour la recherche d'appariement (pattern matching, shape matching, et color matching).

IMAQ Vision est utilisé en industrie comme dans des laboratoires d'opérations automatiques qui nécessite une fiabilité extrême et des systèmes de vision de hautes performances [15].

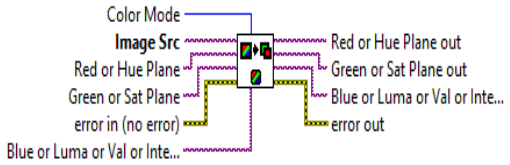
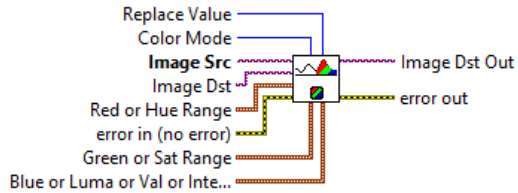
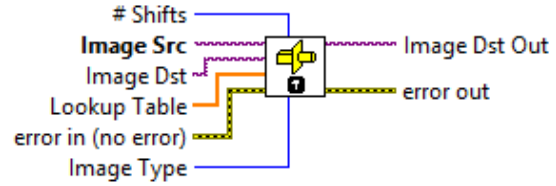
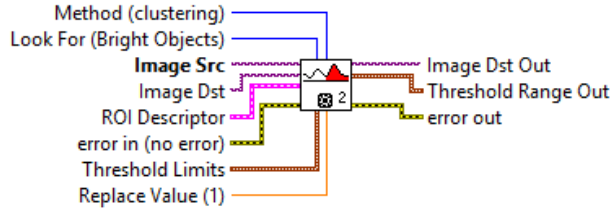
Représentation de la fonction	nom	description
	IMAQ ExtractColorPlanes	Permet d'extraire d'une mage les plans rouge, vert ou bleu selon le mode RGB, ou les plans teinte, saturation et valeur selon le mode HSV.
	IMAQ ColorThreshold	Permet d'appliquer des seuillages sur les histogrammes des trois plans de l'image en couleurs
	IMAQ Cast	Converti une image couleur à une image en niveau de gris.
	IMAQ AutoThreshild	Calcul la valeur optimale de seuil sur une image ou une région de l'image.

Tableau II.2 : Quelques fonctions de base pour le traitement d'images sous LabVIEW.

II.7 Les fonctions de traitement d'images avancées :

II.7.1 La fonction Shape Matching :

Cette fonction recherche les formes dans l'image qui ressemblent à une forme prise comme modèle (Template).

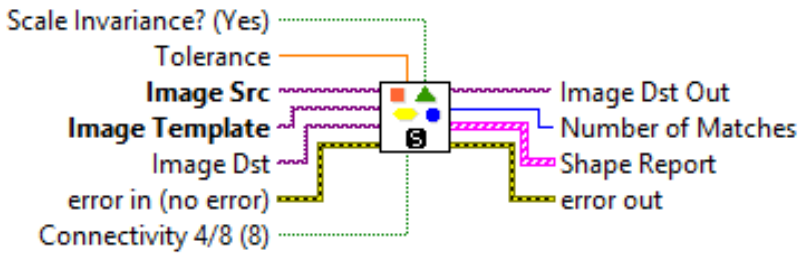


Figure II.5 : La fonction « Shape Matching » sous LabVIEW.

LabVIEW emploie plusieurs techniques pour la recherche d'appariement (matching) dans une image, nous allons présenter quelques-unes : [16]

II.7.1.1 Inter-corrélation :

Le processus mathématique de l'inter-corrélation des images est simple. Conceptuellement, le gabarit est posé sur l'image source, et les valeurs d'intensité pour chacun des pixels correspondant sont multipliées individuellement, puis tous sont additionnés pour produire une valeur de corrélation unique. La forme est ensuite déplacée d'un pixel, et le processus est répété jusqu'à ce que l'image entière de la source ait été couverte, et une matrice des valeurs de corrélation est créée. La matrice de valeur de corrélation est ensuite scannée pour sa valeur la plus élevée ; cette position se réfère généralement à la position dans l'image source qui correspond le mieux au modèle. Bien entendu, la matrice de corrélation peut avoir plusieurs valeurs élevées en elle, correspondant à plusieurs instances de la forme dans la l'image source.

Pour expliquer le concept de l'inter-corrélation, on considère une matrice de l'image source de dimension *a*x*b*, et une matrice de l'image contenant la forme qui est de dimension *c*x*d*, (avec $c \leq a \; et \; d \leq b$).

Si les deux l'image source et la forme sont normalisées, la matrice d'inter-corrélation est formée en utilisant l'équation suivante:

$$M_{i,j} = \sum_{x=0}^{b-1} \sum_{y=0}^{a-1} (F_{x,y})(S_{(i+x),(j+y)}) \qquad \text{(II.1)}$$

M : matrice d'inter-corrélation.

F : matrice de la forme.

S : matrice de l'image source.

II.7.1.2 Invariance d'échelle et invariance à la rotation :

L'un des plus gros défauts de l'inter-corrélation est son incapacité de faire correspondre les objets dans une image source qui sont soit de taille différente pour le modèle, ou ont été mis en rotation. Si cette question est à surmonter, le modèle doit être ré analysé sur l'image source en utilisant différentes rotations et tailles, ceci peut être extrêmement long.

Plusieurs techniques existent qui peuvent aider à accélérer le processus de la recherche dans le cas de la variation de la taille et de la rotation. La méthode la plus importante est de refaire la recherche seulement pour les variations qui peuvent exister. Si l'élément dans l'image originale est de même taille, on n'a pas besoin de faire recherche pour la variation de la taille, de même pour la rotation.

II.7.1.3 Correspondance pyramidale:

Bien que la précision de l'inter-corrélation est assez bonne, mais c'est une technique très lente. La correspondance pyramidale utilise une technique de recherche similaire, mais elle sous-échantillonne l'image source et la forme à une plus petite résolution spatiale, réduisant efficacement le nombre de données à rechercher de plus de 75 %. Comme les résolutions sont plus faibles, plus il faut prendre soin de définir ce qui est une correspondance potentielle, comme la recherche est effectuée sur beaucoup moins de données, donc seulement dans des zones avec des résultats très élevés correspondants seront considérées comme corrélant.

II.7.1.4 Techniques de correspondance complexes :

Au lieu d'essayer de faire correspondre simplement la matrice d'intensité du modèle avec la matrice d'intensité d'une image source, d'autres techniques sont utilisées telles que la modélisation géométrique, qui est le processus de décomposition à la fois de la source et du modèle en objets géométriques simplement définis. (voir la figure I.10).

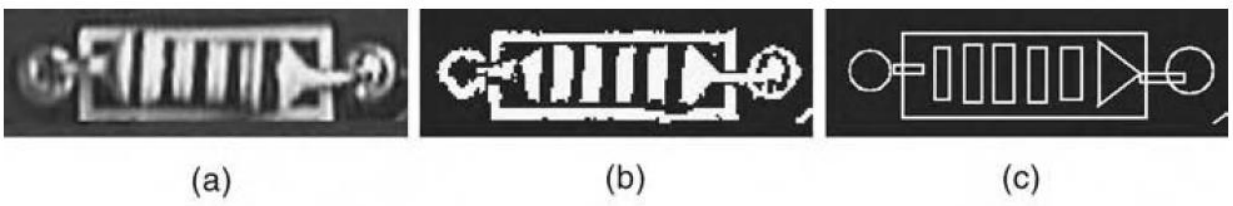


Figure II.6 : (a) Image originale (b) image traitée (c) modélisation géométrique. [16]

La forme est alors recherchée d'une manière plus simple et plus rapide. Une autre technique utilisée est l'échantillonnage spatial non uniforme. Considérons l'image binaire suivant échantillonné uniformément :

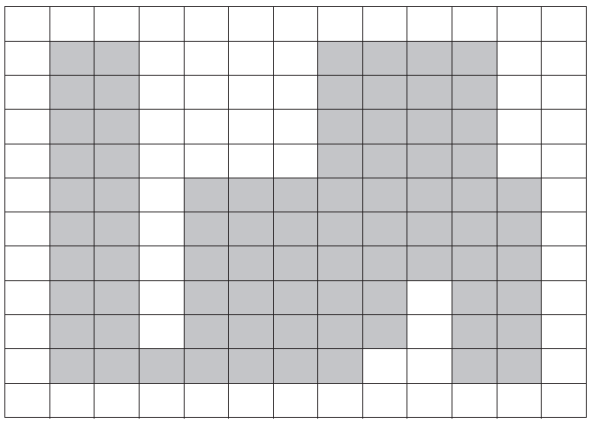


Figure II.7 : image binaire. [16]

Comme il y a des sections de pixels avec les mêmes intensités que leurs voisins, elle peut être ré échantillonnée de manière non uniforme :

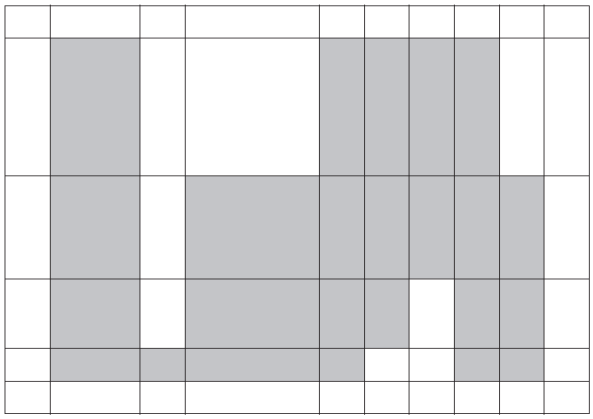


Figure II.8 : image ré échantillonnée. [16]

L'image est la même, mais elle est échantillonnée à travers moins de pixels de forme irrégulière.

II.7.2 Classification de formes :

Le processus de classification de formes est simplifié sur LabVIEW grâce à la fonction « Particle Classification ». Cette fonction utilise l'approche statistique pour la reconnaissance de motifs.

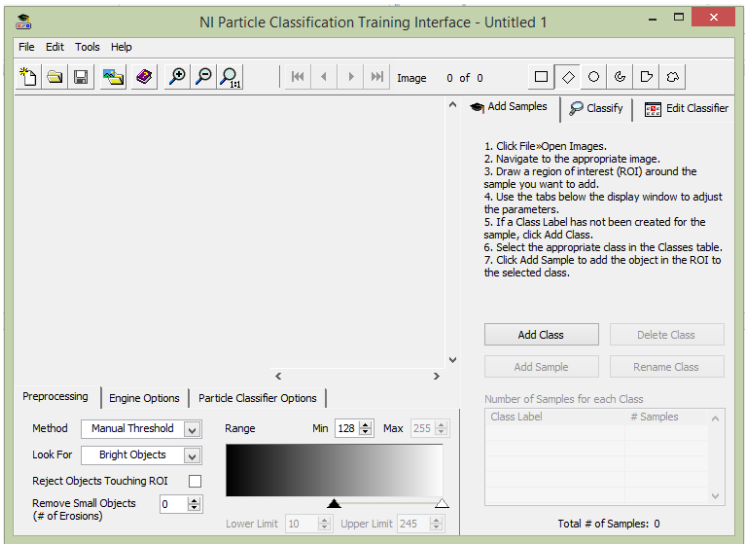


Figure II.9 : Environnement de création d'un classifieur de formes.

Le diagramme ainsi que les différentes étapes de la procédure de classification sont présentées dans la figure (II.7)

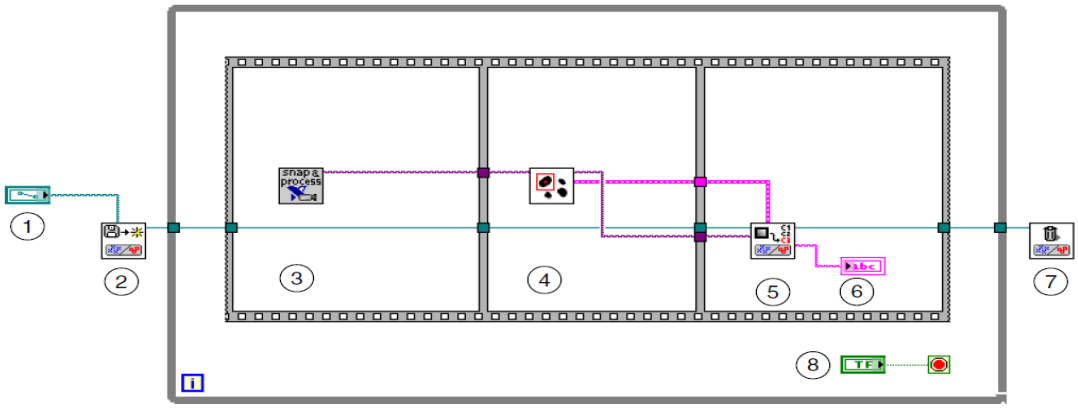


Figure II.10 : Classification de formes sous LabVIEW.

Ces étapes sont :

1) Spécifier le chemin au fichier de classification.

2) Lire le fichier le fichier de classification.

3 Acquisition d'images.

4) localiser les éléments de l'image.

5) Classer les formes.

6) Indiquer la classe.

7) Déposer le classifieur.

8) stop.

II.7.2.1 Le processus de classification :

a) Calcul des paramètres caractéristiques: [17]

NI Particle Classifier utilise les caractéristiques suivantes pour la description d'une forme, ces caractéristiques sont invariants aux dimensions, à la rotation et à la réflexion des échantillons.

- Caractéristique 1 : décrit la fonction de circularité de l'échantillon.
- Caractéristique 2 : décrit le degré d'allongement de l'échantillon.
- Caractéristique 3 : décrit la convexité de la forme de l'échantillon.
- Caractéristique 4 : une description plus détaillée de la convexité d'une forme d'échantillon.
- Caractéristique 5 : est utilisé pour la discrimination des échantillons avec des vides (trous).
- Caractéristique 6 : est utilisé pour la discrimination plus détaillée des échantillons avec des vides.
- Caractéristique 7 : représente la propagation de l'échantillon.
- Caractéristique 8 : représente l'élancement de l'échantillon.

b) Classification :

Les méthodes de classification qu'on peut utiliser dans cette fonction sont : la distance moyenne minimale, la méthode de K plus proches voisins, et la méthode du plus proche voisin. (Ces méthodes sont présentées dans le premier chapitre)

Chacune de ces méthode peut employer différents espaces métriques pour le calcul de la distance :

Un espace métrique est un ensemble au sein duquel une notion de distance entre les éléments de l'ensemble est définie.[20]

Etant donnés deux vecteurs $x=(x_1, x_2, \ldots, x_n)$ et $y=(y_1, y_2, \ldots, y_n)$

1. distance euclidienne (définie dans le chapitre 1)

2. distance max :[19]

$$d(x,y) = \max(\|x_1 - y_1\|, \|x_2 - y_2\|, \dots, \|x_n - y_n\|) \quad \text{(II.2)}$$

3. somme :[19]

$$d(x,y) = \|x_1 - y_1\| + \|x_2 - y_2\| + \cdots + \|x_n - y_n\| \quad \text{(II.3)}$$

II.7.2.2 Evaluation de la classification :

A la fin de l'opération de classification, ''NI Particle Classification'' renvoie deux valeurs (scores) pour évaluer l'opération : score de classification et score d'identification. [18]

a) Score de classification :

Il indique le degré pour lequel la classe assigné au motif représente mieux la forme que les autres classes. Il est défini par la formule suivante :

$$Score\ de\ classification = 1000\left(1 - \frac{d_1}{d_2}\right) \quad \text{(II.4)}$$

d_1 : représente la distance à la classe la plus proche.

d_2 : représente la distance à la deuxième classe plus proche.

La distance dépend de l'algorithme utilisé, d_1 et d_2 sont normalisées et sont comprises entre 0 et 1 donc le score de classification est une valeur entre 0 et 1000.

b) Score d'identification :

Il indique la similarité entre le nouveau motif et la classe pour laquelle il est affecté, il est défini par la formule suivante :

$$Score\ d'identification = 1000(1 - d) \quad \text{(II.5)}$$

Avec :

$$d = \frac{distance\ entre\ le\ motif\ et\ sa\ classe}{facteur\ de\ normalisation} \quad \text{(II.6)}$$

Le facteur de normalisation est défini comme la distance maximale interclasses.

II.7.3 Reconnaissance des caractères OCR :

OCR (Optical Caracter Recognition) est le processus d'extraction d'informations textuelles dans une image, il est souvent très utilisé dans le monde industriel, comme dans l'extraction des numéros d'immatriculations des véhicules, le scan des documents et la sauvegarde des textes dans des fichiers. L'outil emploie des techniques très puissantes qui sont capable d'extraire des textes dans des différentes conditions de luminosité et de différents types de police.

Bien que cette boîte à outils soit particulièrement adaptée à la lecture de caractères alphanumériques qui ont été produites en utilisant la composition et l'impression traditionnelle, il est également capable de lire les polices couramment utilisés dans les domaines de l'automobile, l'électronique et la pharmaceutique.

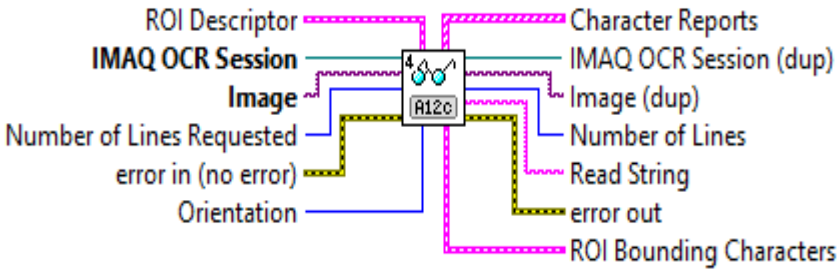


Figure II.11 : la fonction « OCR Classification » Sous LabVIEW.

II.8 Le contrôleur CompactRIO :

II.8.1 Description :

Le compact RIO (Compact Reconfigurable Input Output) est un système d'acquisition et de contrôle industriel robuste alimenté par une technologie FPGA de haute performance. NI compact RIO contient un processeur temps réel et un châssis FPGA reconfigurable pour des applications embarquées ou distribuées, et des modules reconfigurables (entrée/sortie) avec conditionnement de signal pour une connexion directe à des capteurs et actionneurs. Les systèmes embarqués avec compact RIO sont développés en utilisant la programmation graphique LabVIEW.

II.8.2 Architecture du CompactRIO: [21]

Le compact RIO combine un processeur embarqué temps réel, un FPGA de haute performance et des modules Entrée/Sortie. Chaque module Entrée/Sortie est directement connecté au FPGA. Le FPGA est connectée au processeur embarqué temps réel via un bus PCI, ce qui représente une architecture minime avec peu de ressources matérielles.

LabVIEW contient des mécanismes de transfert d'informations des modules E/S à l'FPGA au processeur pour une analyse temps réel, post-traitement, exploitation ou pour une communication sur réseau.

Chaque module E/S contient des circuits pour connexion, conditionnement de signal, et la convention (Analogique/Numérique ou Numérique/Analogique).

II.8.2.1. Contrôleur temps réel :

Le contrôleur temps réel contient un processeur qui est capable d'exécuter des applications LabVIEW d'une manière fiable et déterministe et permet d'établir un control multiniveaux et de communiquer avec d'autres périphériques.

Le contrôleur contient aussi un duel port Ethernet de 10 à 100 Mo/s pour une communication à travers un réseau et sur des serveurs web (HTTP) et FTP. Il inclut aussi une alimentation de 9 à 30 volt, une horloge temps réel, wtchdog timers, plus de 2GO pour le stockage des données, des supports USB et RS232.

Figure II.12: Contrôleur temps-réel du cRIO.

II.8.2.2 Châssis FPGA reconfigurable:

Le circuit reconfigurable FPGA est le centre de système embarqué, il est directement connecté aux modules E/S. Puisque chaque module est connecté directement au circuit FPGA au lieu d'une connexion à travers un bus, l'expérience a montré qu'il y a plus de retard dans la réponse du système en comparant avec d'autres architectures. Par défaut, le circuit FPGA communique automatiquement avec les modules E/S et fournit des données au processeur.

Figure II.13: Châssis FPGA reconfigurable.

II.8.2.3 Modules E/S:

Le module E/S contient des circuits de conversion, de conditionnement de signal et des connexions directes à des capteurs et actionneurs.

En offrant une variété d'options pour le câblage et en intégrant des jonctions de connexion, les ingénieurs ont le choix entre plus de 70 séries de modules pour connecter n'importe quel capteur ou actionneur. Les modules incluent des entrées pour thermocouples $\pm$10V. Modules E/S numérique 24 bit, modules E/S analogique 24V et un courant de plus de 1A, support carte (SD).

Figure II.14: Modules entrée/sortie du compactRIO.

II.8.3 Architecture software des CompactRIO:

Généralement, tous les systèmes compact RIO ont au minimum trois niveaux d'exécutions asynchrone à travers trois cibles : le circuit FPGA, le système d'exploitation temps réel et le PC. Le VI sur le PC permet à l'operateur d'interagir avec le système, le RTOS exécute les commandes de haut niveau et le FPGA exécute les commandes de bas niveau.

Il y a deux chemins de communication, Le premier est pour envoyer les commandes de l'utilisateur au compactRIO. Le deuxième est pour envoyer les infos du compactRIO à l'interface utilisateur pour affichage.

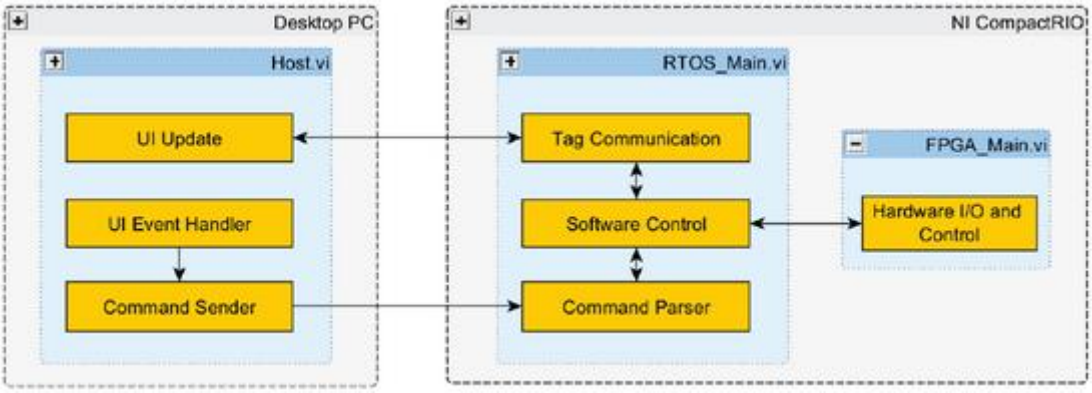


Figure II.15: Architecture software du compactRIO.

UI Update: Cette fonction reçoit les dernières données mises à jour comme la valeur de la température, la vitesse ou l'accélération et les afficher sur l'interface.

Command Sender: Il envoi chaque commande reçue de UI Event Handler au contrôleur cRIO.

UI Event Handler: reçoit les commandes de l'operateur.

Command Parser: C'est une boucle qui utilise le flux réseau pour recevoir les commandes de l'interface utilisateur à travers le câble Ethernet. Les commandes sont placées dans une chaine puis envoyées à la boucle Message Handler pour la distribution.

Message Handler: Cette boucle reçoit chaque commande de l'interface utilisateur et l'envoyer au processeur FPGA.

II.8.4 Modes de programmation du CompactRIO:

L'architecture du compact RIO présentée précédemment permet de faire le choix entre l'implémentation du code en faisant une adaptation ou une personnalisation du Hardware de l'FPGA à travers LabVIEW FPGA ou en utilisant le mode scan du compactRIO fourni par le module temps réel du LabVIEW.

II.8.4.1 Le mode interface LabVIEW FPGA:

Dans ce mode, on peut bénéficier de la puissance réel du compact RIO en personnalisant l'FPGA. Ce qui permet d'atteindre une grande performance. En communicant des données entre un VI FPGA et un VI temps réel on a la possibilité d'atteindre une haute vitesse dans le flot des données.

II.8.4.2 Le mode Scan:

En utilisant ce mode on peut programmer le processeur temps-réel du compactRIO mais pas le FPGA qui possède dans ce cas une configuration prédéfinie qui scanne les données des entrées/sorties et les placent dans la mémoire pour les rendre accessibles par LabVIEW.

Le mode Scan est suffisant pour les applications possédant qu'un seul point d'accès pour les E/S à une fréquence qui ne dépasse pas quelques centaines Hertz.

II.8.4.3 Les utilisations du mode LabVIEW FPGA :

De même que les systèmes basés sur les processeurs, les FPGA sont aussi utilisés pour le control des procédés industriels par commande analogique, discret ou logique. Par contre, les systèmes basés sur les FPGA diffèrent des systèmes basés sur des processeurs.

On utilise le mode FPGA si l'application visée nécessite l'une de ces besoins :

- Performances et fiabilité maximale.
- Acquisition ou génération d'ondes de haute fréquence (supérieur à 500Hz).
- Synchronisation.

II.8.4.4 Les utilisations du mode Scan :

Ce mode est utilisé pour le control des systèmes possédant un seul point d'accès pour les E/S. les informations transmises dans ces systèmes sont la valeur courante d'une grandeur physique, et n'ont pas besoin d'une fréquence supérieur à 500Hz.

Même si on ne peut pas l'utiliser pour l'acquisition de données à haut fréquence, le mode Scan offre plusieurs avantages :

- Facilité de programmation et diagnostic.
- Interaction dynamique des E/S.

II.9 Présentation du robot-véhicule « ROBUCAR » :

Le Robucar est un prototype de petit véhicule électrique construit sur la base d'un châssis tubulaire des petites voitures utilisées dans les terrains de Golf. Il comporte une architecture parallèle permettant le développement d'applications temps réel. Ses caractéristiques générales et techniques sont les suivantes :

• longueur totale : 1836 mm.
• largeur totale : 1306 mm.
• hauteur : 616 mm.
• poids total avec batteries : 310 Kg.
• motorisation : 4 moteurs électriques de 1200 Watts.
• 4 roues motrices et directrices.
• vitesse maximale : 18 Km/h (5m/s).
• autonomie : 2 heures d'utilisation continue.
• capacité d'accueil : 2 personnes avec bagages.
• conduite automatique ou manuelle.

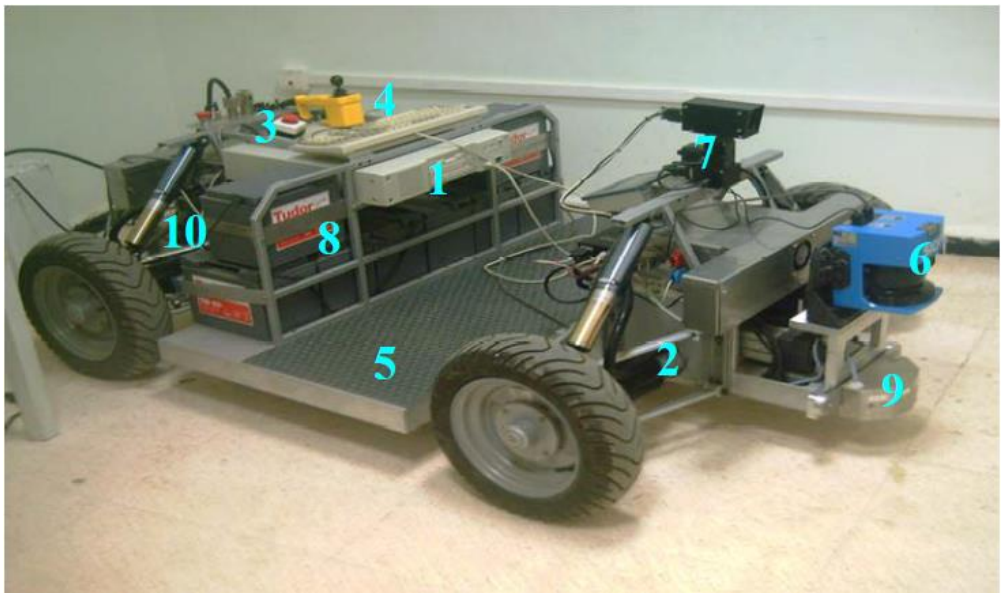


Figure II.16 : Présentation du Robucar.

Le tableau suivant résume les différentes parties du Robucar telles qu'elles sont numérotés dans la figure (II.13)

1	Pc embarqué	6	LMS Sick200.
2	Axe avant (2roues motrices, vérin direction, et un µCrtl Mpc555)	7	Caméra CCD.
3	Arrêt d'urgence.	8	Batteries.
4	Joystick.	9	Ultrasons
5	Châssis nid d'abeille.	10	Axes arrière (2roues motrices, vérin direction)

Tableau II.3: Les principaux composants du Robucar.

II.10 Conclusion :

Dans ce chapitre nous avons présenté l'environnement de programmation LabVIEW qui emploie le langage graphique, et nous avons illustré la structure des programmes sous LabVIEW. Ensuite nous avons présenté les techniques et l'application de quelques fonctions essentielles servant pour le traitement d'images dans le cadre de notre projet LabVIEW inclut plein d'autres fonctions de hautes performances pour le traitement d'images que nous ne pouvons pas aborder d'une manière exhaustive. Et enfin nous avons donné une description du robot-véhicule « Robucar ».

CHAPITRE III

Développement du programme de détection et de reconnaissance des panneaux de signalisation routière.

III.1 Introduction :

La reconnaissance d'un panneau de signalisation routière est une tâche facile pour l'être humain, mais ce n'est pas le cas pour un ordinateur. Afin d'élaborer un programme capable d'accomplir cette tâche, nous allons exploiter les techniques de traitement d'images et les différentes fonctions offertes par LabVIEW pour ce domaine.

III.2 Définition de la tâche :

Notre projet consiste à développer programme fonctionnant en temps réel sous LabVIEW pour la détection et la reconnaissance des panneaux de signalisation routière à partir des images issues d'une caméra embarquée sur le robot mobile « ROBUCAR ». Il existe plusieurs contraintes pour lesquelles le programme doit faire face telles que la variation des conditions d'éclairage, l'occultation partielle et le positionnement des panneaux, ajoutant à tous ces problèmes le fait que le traitement et la détection se font en temps réel ce qui nous exige d'élaborer des algorithmes qui ne sont pas couteux en termes de temps d'exécution.
Le travail est limité à la détection et la reconnaissance de quatre panneaux (figure III.1) ainsi que les feux tricolores.

Figure III.1 : Panneaux utilisés dans le projet.

III.3 La stratégie employée pour la détection des panneaux :

Afin d'aboutir à l'objectif de ce travail nous devons exploiter toutes les caractéristiques de chaque panneau, à savoir la couleur, la forme ou le pictogramme du panneau ainsi que les caractères contenus dans le panneau. Notons que cette troisième caractéristique n'est exploitée que pour le panneau de limitation de vitesse (troisième panneau dans la figure III-1) puisqu'il est important de savoir la valeur de la vitesse limitée, pour les autre panneaux, le programme se limite à la détection de la couleur du panneau et la reconnaissance de son pictogramme.

Donc la stratégie générale employée est la suivante : premièrement, nous devons extraire la couleur du panneau. Puis, faire un traitement pour mieux illustrer les informations utiles et éliminer les données indésirables. Et enfin, rechercher les formes qui correspondent aux panneaux enregistrés dans le fichier de classification.

Nous présentons ci-dessous les différentes étapes du programme, puis nous détaillerons chaque tache.

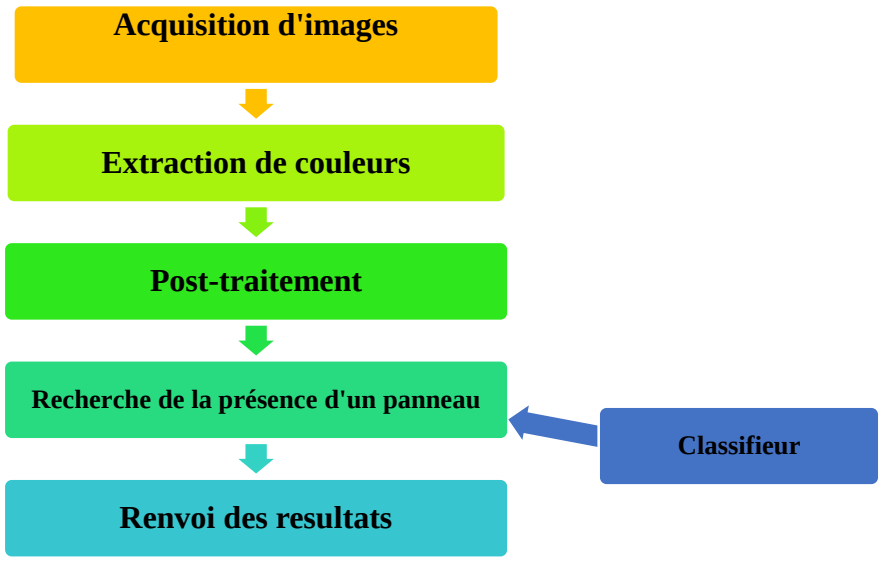


Figure III.2 : les différentes étapes de fonctionnement global du programme de détection.

III.3.1 Acquisition d'images :

L'acquisition d'image s'effectue en temps réel (flux vidéo) à travers une caméra USB fixée sur le robot.

Figure III.3 : La camera utilisée dans le projet.

III.3.2 Extraction de couleurs :

Il est clair que les couleurs caractéristiques des panneaux sont le rouge et le bleu, pour extraire les panneaux du reste des éléments de l'image, on procède à l'extraction de ces deux couleurs.

L'apparence de ces couleurs change selon les conditions d'éclairage du milieu telles que les conditions météorologiques (temps ensoleillé, nuageux) et le changement de la luminosité durant les différents moments de la journée. Donc nous devons tenir compte de toutes ces conditions.

Il n'existe pas des théories ou de méthodes fixes pour l'extraction de couleurs, mais on cherche des approches plus ou moins complexes pour atteindre l'objectif désiré. Il faut d'abord choisir le système de représentation de couleur le plus adéquat parmi ceux qui existent (RGB, HSV, Lab, … etc.), ensuite, on peut exploiter l'histogramme de l'image et faire des manipulations sur ce dernier afin d'extraire les informations désirées.

A l'issu de l'opération d'extraction de couleur, on obtient une image binaire où les pixels ayant les caractéristiques de la couleur désirée sont mis à la valeur binaire '1' et sont représentés en blanc, les autres pixels sont mis à la valeur binaire '0' et sont représentés en noir.

III.3.2.1 Extraction de la couleur rouge :

Nous avons mis au point deux algorithmes différents pour cet objectif en utilisant deux espaces de représentation de couleurs différents.

a) Utilisation de l'espace HSV :

D'après les définitions des différents espaces de représentation de couleur, il parait clair que l'espace HSV est le plus représentatif des informations concernant la couleur et la luminosité.

Notre approche consiste à appliquer un seuillage en fixant des intervalles sur les trois composantes de l'espace HSV afin de garder les valeurs représentant la couleur désirée. Un grand nombre d'images prises dans de conditions d'éclairage différentes sont exploitées nous ont permis de fixer les valeurs de seuils d'une manière empirique et ainsi d'élaborer l'algorithme suivant :

$$for\ x = 1\ to\ \ x = l$$

$$for\ y = 1\ to\ \ y = h$$

$$if\ \begin{cases} 0 \le I_H(x,y) \le 12\ \ or\ \ 238 \le I_H(x,y) \le 255 \\ \qquad\qquad and \\ 95 \le I_S(x,y) \le 255 \\ \qquad\qquad and \\ 40 \le I_V(x,y) \le 255 \end{cases} then\ I'(x,y) = 1$$

$$else\ I'(x,y) = 0$$

$$end\ for;$$

$$end\ for;$$

h et *l* sont respectivement la hauteur et la largeur de l'image.

I_H, I_S, et I_V sont des images en niveau de gris représentant respectivement les plans teinte (Hue en anglais), Saturation et Valeur (Value en anglais).

I' est l'image binaire issue de l'exécution de l'algorithme.

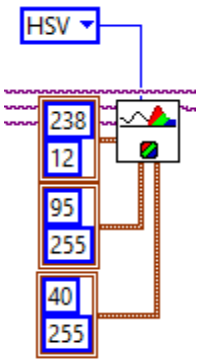


Figure III.4 : Diagramme du seuillage sous LabVIEW.

b) Utilisation de l'espace RGB :

Dans cet espace, les informations sur les couleurs changent considérablement selon la luminosité ce qui ne nous permet pas de faire un seuillage qui soit pratique pour les différentes

conditions de luminosité, mais une autre caractéristique peut être exploitable, en effet les trois composantes R, G et B varient d'une manière similaire sous l'effet des conditions d'éclairage, donc on peut exploiter le rapport entre ces trois grandeurs. L'algorithme mis au point est le suivant :

$$for\ x = 1\ to\ \ x = l$$

$$for\ y = 1\ to\ \ y = h$$

$$if\ \frac{I_R(x,y)}{I_G(x,y)+1} \geq s\ and\ \frac{I_R(x,y)}{I_B(x,y)+1} \geq s\ \ then\ I'(x,y) = 1$$

$$else\ I'(x,y) = 0$$

$$end\ for;$$

$$end\ for;$$

I_R, I_G, et I_B sont des images en niveau de gris représentant respectivement les plans rouge, vert (green) et bleu.

s : est une valeur seuil fixée de manière empirique en faisant des tests sur plusieurs images.

Nous avons rajouté la valeur '1' aux dénominateurs pour éviter la division sur zéro dans le cas où l'une des valeurs I_G ou I_B est nulle sachant que cette valeur n'a pas d'effet sur le programme dans le cas contraire.

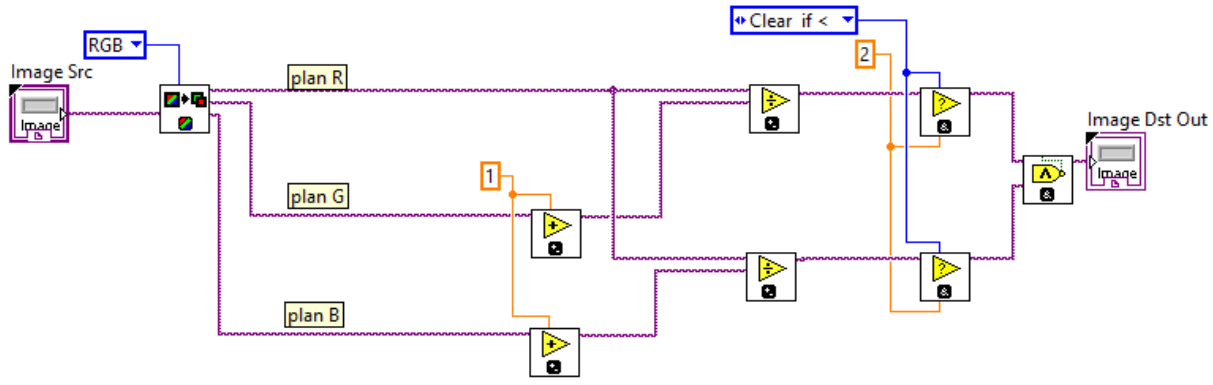


Figure III.5 : Diagramme du programme sous LabVIEW.

Figure III.6 : extraction de la couleur rouge.

III.3.2.2 Extraction de la couleur bleu :

De la même manière que pour l'extraction de la couleur rouge, nous avons établi un seuillage sur les histogrammes des composantes H, S et V.

L'utilisation d'un grand nombre d'images prises dans de différentes conditions d'éclairage nous ont permis de fixer les intervalles de seuils suivantes :

H: [145,180] **S:** [64,255] **V:** [45,255].

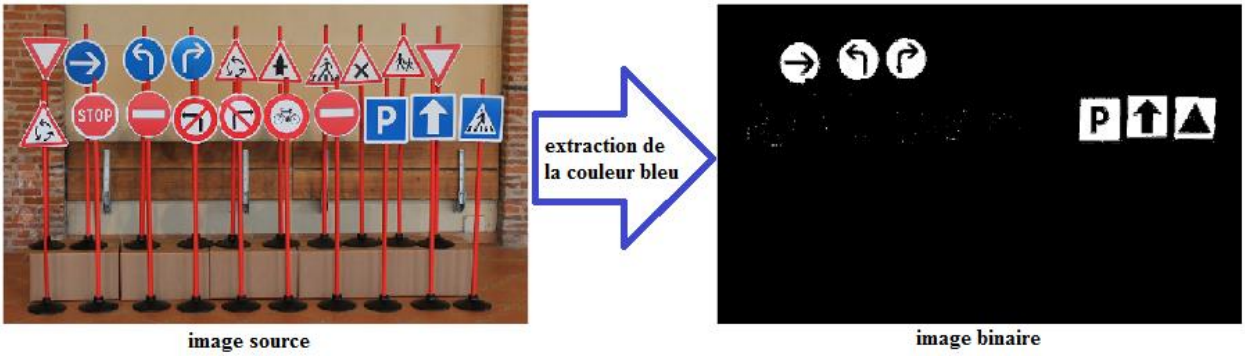


Figure III.7: extraction de la couleur bleu.

III.3.3 Post-traitement :

L'objectif de cette opération est d'améliorer la représentation des données de l'image en éliminant les éléments indésirables et insignifiants afin d'accélérer le processus de recherche des panneaux.

La première opération consiste à éliminer les particules de petite taille en utilisant un algorithme qui calcule la surface de chaque élément de l'image puis élimine les éléments dont la surface ne dépasse pas le seuil minimal fixé.

La deuxième opération consiste à éliminer les éléments de l'image qui touche la bordure de l'image, ces éléments représentent des données incomplètes et insignifiantes.

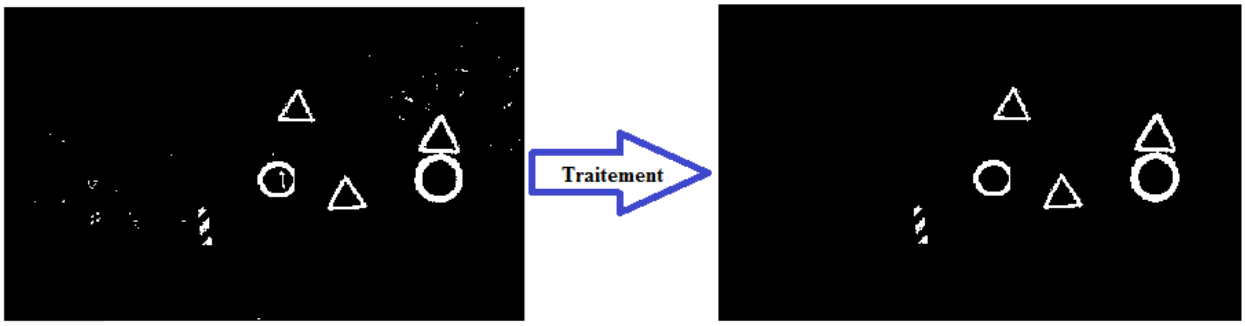


Figure III.8 : Post-traitement de l'image.

III.3.4 La localisation :

Nous cherchons dans cette étape à localiser tous les éléments de l'image en relevant leurs coordonnées ROI (Region Of Interest) pour les soumettre ensuite au classifieur. Cette tâche est accompli par la fonction « IMAQ ParticleAnalysisReport » qui donne à sa sortie le nombre d'éléments dans l'image ainsi que leurs coordonnées ROI.

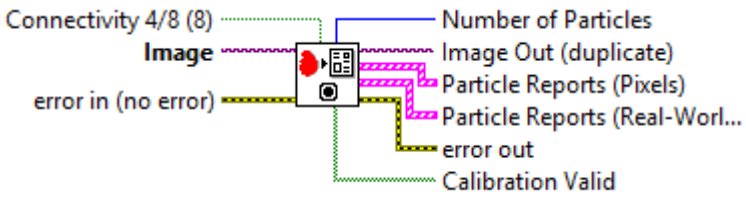


Figure III.9 : Représentation de la fonction « IMAQ ParticleAnalysisReport » sous LabVIEW.

III.3.5 La classification :

Il s'agit dans cette partie de créer les bases d'apprentissage pour le classifieur et de fixer une méthode pour la classification.

a) Création de la base d'apprentissage :

Nous allons créer les différentes classes, chaque classe représente un panneau. Puis, pour chaque classe (type de panneau), nous utiliserons des échantillons du panneau, ces échantillons sont des images binaires du même panneau prises dans des différents cas de positionnement, il faut utiliser un nombre minimal d'échantillons qui puisse couvrir un nombre maximal de cas afin de ne pas rendre la procédure de classification lente.

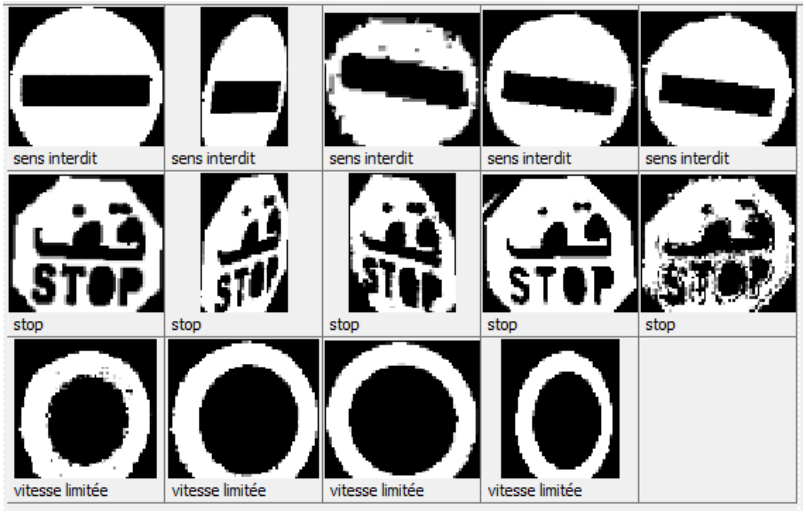


Figure III.10 : Base d'apprentissage des panneaux rouges.

Figure III.11: Base d'apprentissage des panneaux bleus.

b) Choix de la méthode de classification :

LabVIEW propose les trois méthodes suivantes pour la classification :

- La méthode de K plus proches voisins.

- La méthode du plus proche voisin.

- La méthode de la distance moyenne minimale.

On dispose aussi de trois formules pour le calcul de la distance :

- Distance euclidienne.

- Somme.

- Max.

(Ces distances sont expliquées dans les chapitres I et II)

Les résultats seront présentés dans le chapitre IV.

III.4 Cas particulier pour le type de panneaux de limitation de vitesse :

Ce type de panneaux nécessite d'extraire une information de plus, il s'agit de la valeur de la vitesse limitée qui doit être interprétée correctement par le programme, pour cela, on intègre dans le programme la fonction OCR qui permet de lire des caractères alphanumériques appris auparavant.

Le processus de reconnaissance de ce type de panneaux se résume comme suit : si le cercle rouge du panneau est reconnu dans la procédure présenté dans l'organigramme de la figure (III.2), on prélève les coordonnées de cette zone de l'image (coordonnées ROI :region of interest). Ces coordonnées étant prélevées seront utilisées dans une autre image de la même scène mais issue d'un autre traitement afin d'extraire les caractères numériques contenue dans cette zone. Le traitement opéré sur l'image se définit par deux opérations en série:

La première opération est une transformation de l'image originale en image en niveau de gris, cette transformation est obtenue par l'équation suivante :

$$I_{gris} = 0.2989\, I_R + 0.5870 I_G + 0.1140 I_B$$

La deuxième opération consiste à appliquer un seuillage optimale sur la zone contenant le panneau dans l'image transformée. Cette opération est réalisée à l'aide de la fonction « AutoThreshld ».

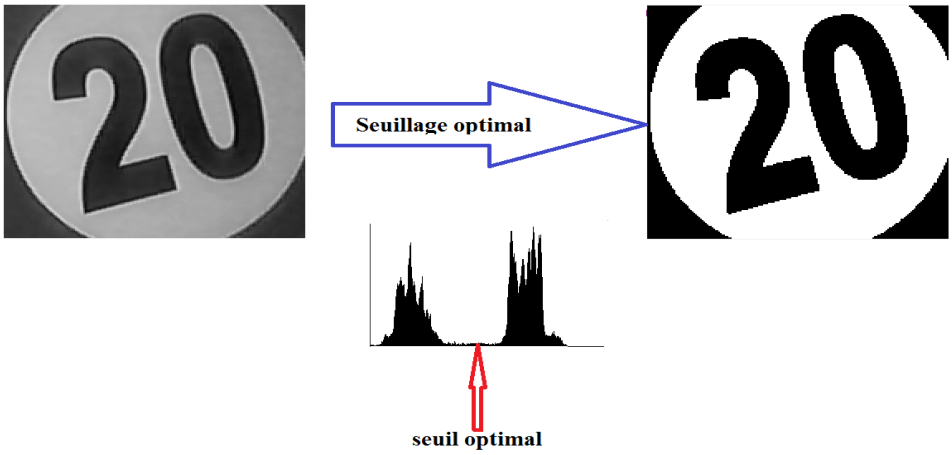


Figure III.12 : Seuillage optimal.

L'étape suivante est la création de la base d'apprentissage des caractères numériques contenus dans le panneau.

De la même manière utilisée pour la création de la base d'apprentissage des panneaux, nous intégrons plusieurs exemples de chiffres au classifieur en lui indiquant à chaque de quel chiffre il s'agit.

Figure III.13 : Base d'apprentissage pour le numéro 20.

Nous avons utilisé plusieurs échantillons pour le numéro 20, des échantillons où nous avons pris chaque chiffre séparément, et d'autres échantillons où nous avons pris les deux chiffres qui composent le numéro pour faciliter la reconnaissance.

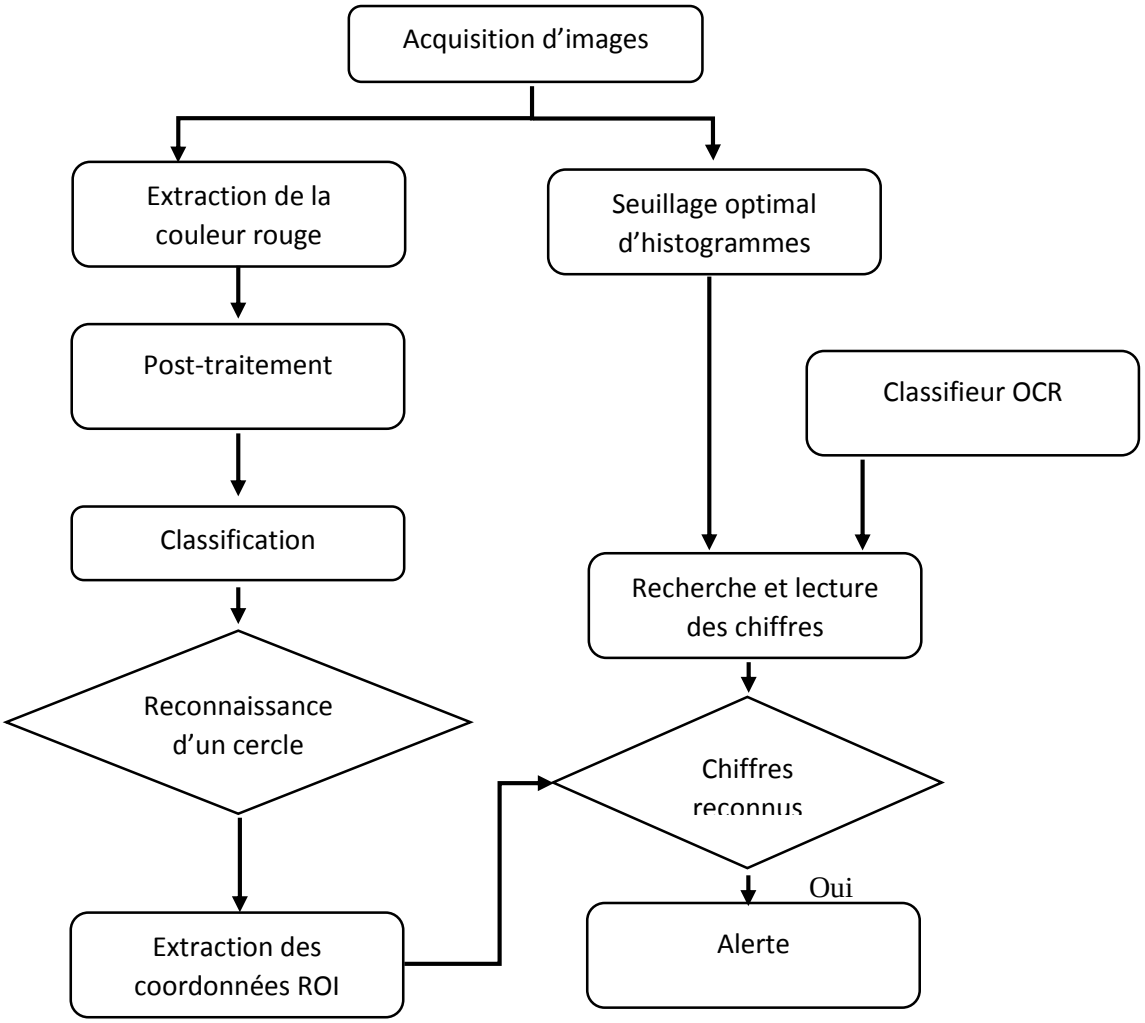


Figure III.14 : Organigramme de fonctionnement de l'algorithme de reconnaissance des panneaux de limitation de vitesse.

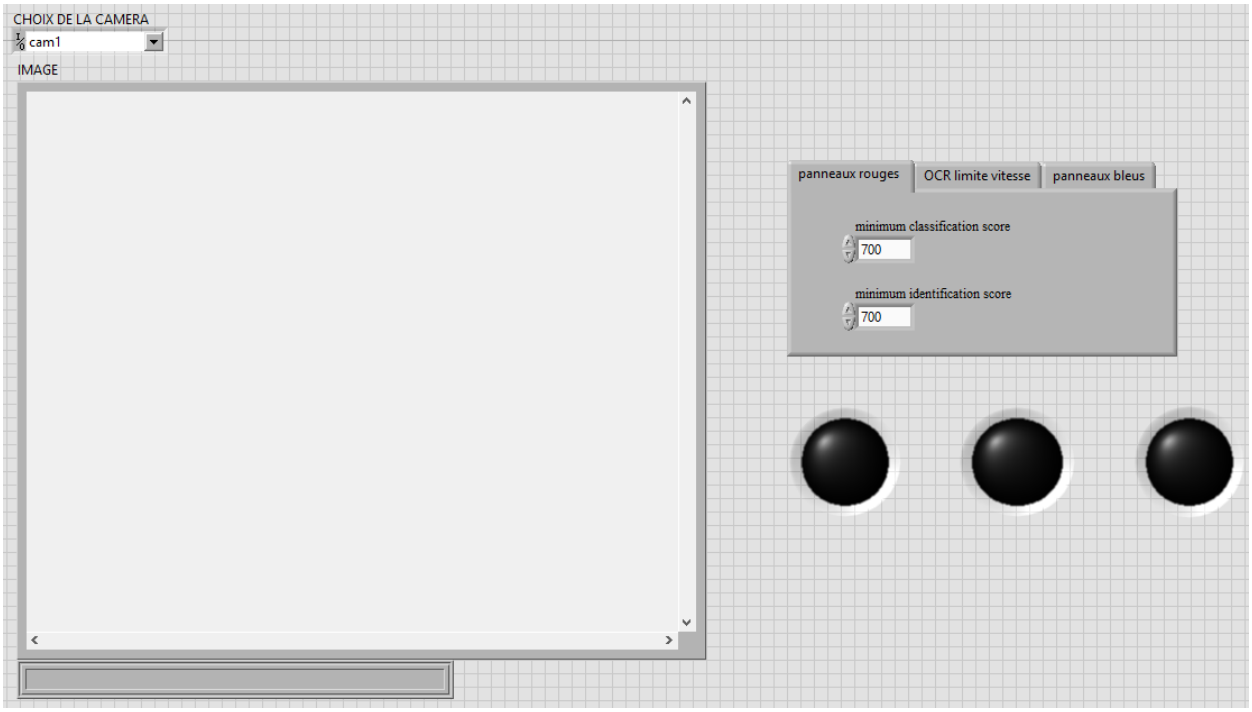


Figure III.15 : capture d'écran de l'interface graphique du programme de reconnaissance des panneaux

III.5 Détection des feux tricolores :

Nous avons développé une approche pour la détection des feux tricolores en exploitant les fonctions appropriées offertes par LabVIEW. Notons que les feux disponibles au laboratoire du CDTA et qui sont utilisés pour le projet ont moins de performances que les vrais feux tricolores employées dans la circulation.

Figure III.16 : Panneau des feux tricolores utilisé dans ce projet.

Notre stratégie repose premièrement sur l'extraction des zones de l'image qui possèdent une forte luminosité afin de détecter les sources de lumière dans l'image, cette opération donne issu à une image binaire. Ensuite, à l'aide de la fonction « shape matching » nous recherchons les éléments de forme circulaire qui sont susceptibles d'être des feux tricolores. La dernière étape consiste à reconnaitre la couleur du feu, cette tâche est réalisée à l'aide d'une fonction d'analyse du spectre de lumière qui permet de donner le niveau des composantes spectrales d'une zone dans l'image. Les graphes des composantes spectrales relevées par cette fonction pour chaque feu sont présentés dans la figure (II.18):

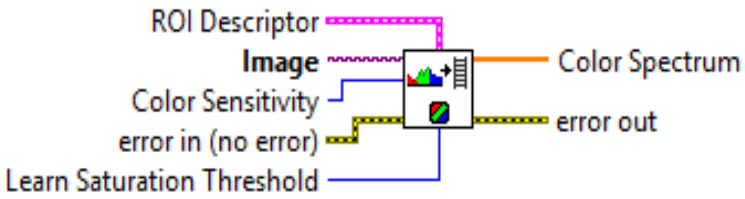


Figure III.17 : Représentation de la fonction ColorLearn sous LabVIEW.

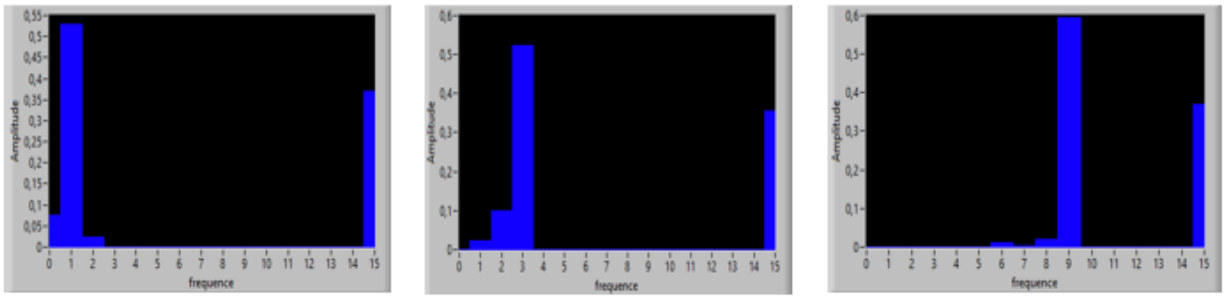


Figure III.18 : spectres des trois feux.

Notons que l'axe des abscisses est un répartis en 16 échantillons qui possèdent des indices de 0 à 15. Ces caractéristiques spectrales sont exploitées pour reconnaitre chaque feu puisque chacun de ces feux possède des composante fréquentielle propre à lui et différentes des caractéristiques fréquentielles des autres feux. L'organigramme général de la stratégie employée pour la détection des feux tricolores est représenté comme suit :

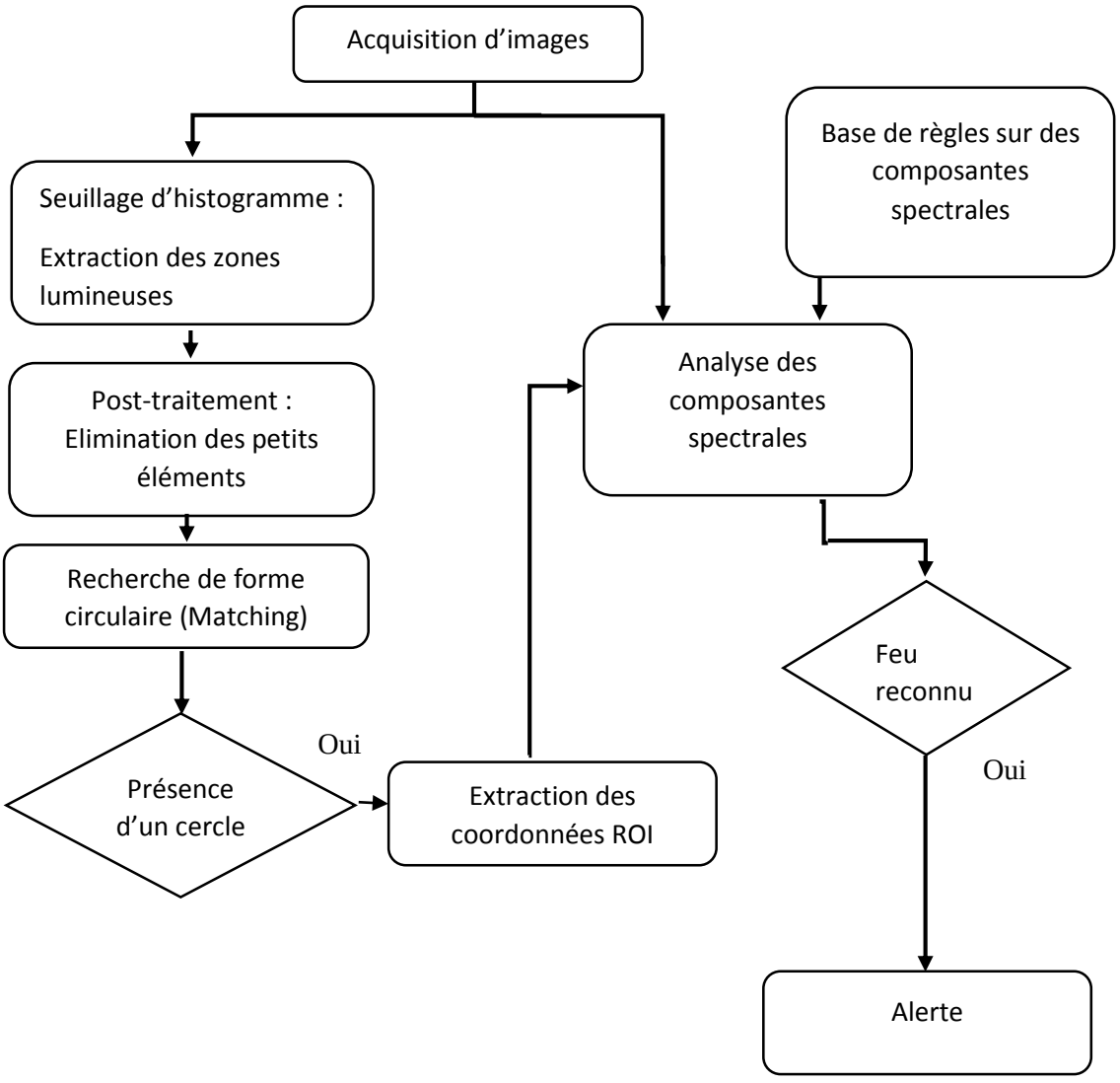


Figure III.19 : Organigramme de fonctionnement de l'algorithme de détection des feux tricolores.

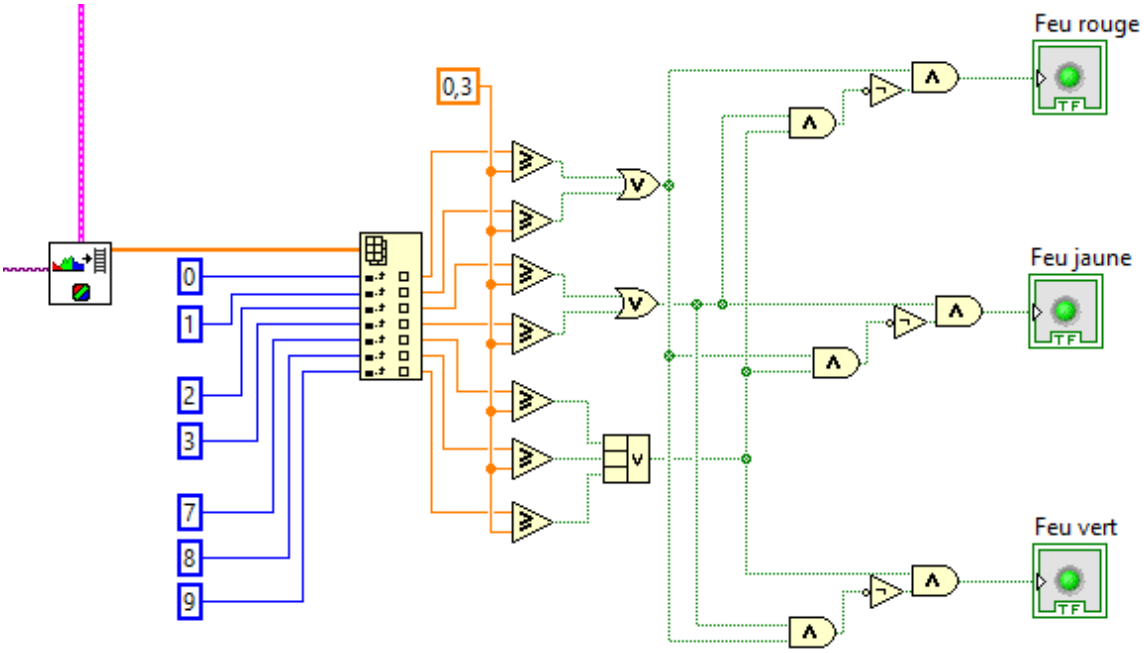


Figure III.20 : Reconnaissance de la couleur du feu détecté.

III.6 Conclusion :

Nous avons présenté dans ce chapitre les stratégies employées à l'implémentation d'un programme pour la détection des panneaux de signalisation et des feux tricolores par les techniques de traitement d'images et en utilisant les fonctions de la bibliothèque NI VISION de LabVIEW.

Les résultats des différentes étapes seront présentés dans le chapitre suivant.

Chapitre IV

Résultats et discussion.

IV.1 Introduction :

Apres avoir détaillé les étapes des stratégies employées, nous allons tester les différents algorithmes mis au point sur plusieurs images prises dans des conditions d'éclairage différentes et représentant des panneaux sous différentes positions afin d'évaluer les performances des différentes techniques.

IV.2 Seuillage des histogrammes pour l'extraction de couleurs :

IV.2.1 Extraction de la couleur rouge :

On sélectionne quelques images qui comportent des panneaux dans différentes conditions d'éclairage afin d'évaluer les deux algorithmes élaborées pour l'extraction de la couleur rouge.

Figure IV.1 : Images utilisées pour les tests.

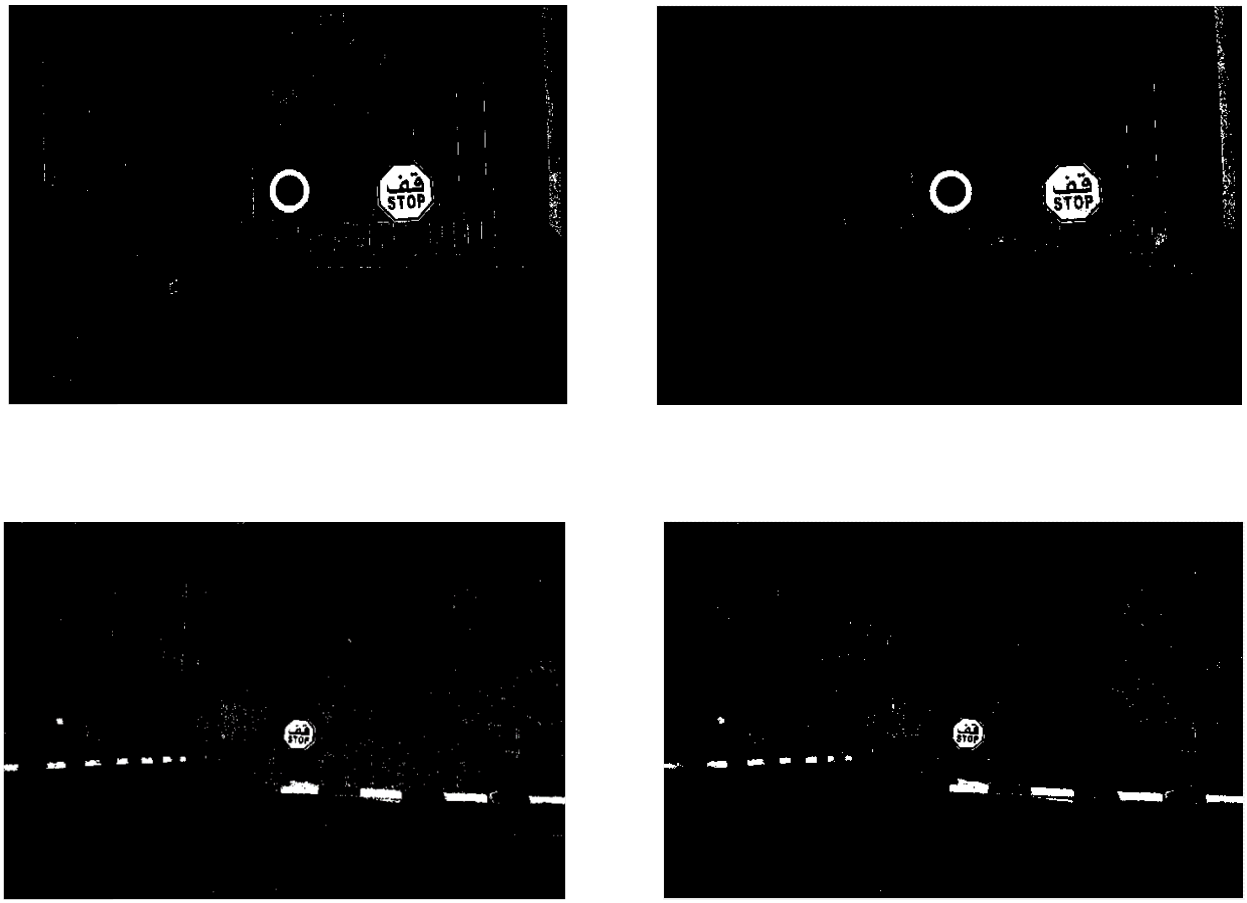


Figure IV.2: Resultats obtenus avec les deux algorithmes. (à gauche : alogorithme utilisant les composantes HSV. à droite : alogorithme utilisant les composantes RGB.)

Une estimation du temps d'exécution de chacun des deux algorithmes avec des images de dimensions 2592*1944 en utilisant un PC avec un microprocesseur Intel CELERON® 2.16Ghz a donné les résultats suivants :

- Algorithme 1 utilisant les composantes HSV : 210 ms
- Algorithme 2 utilisant les composantes RGB : 560 ms

Les résultats obtenus par les deux algorithmes sont presque identiques et sont très satisfaisants, l'algorithme 1 est favorisé et il est le plus adéquat pour une application en temps réel puisqu'il met moins de temps par rapport au deuxième algorithme.

IV.2.2 Extraction de la couleur bleu :

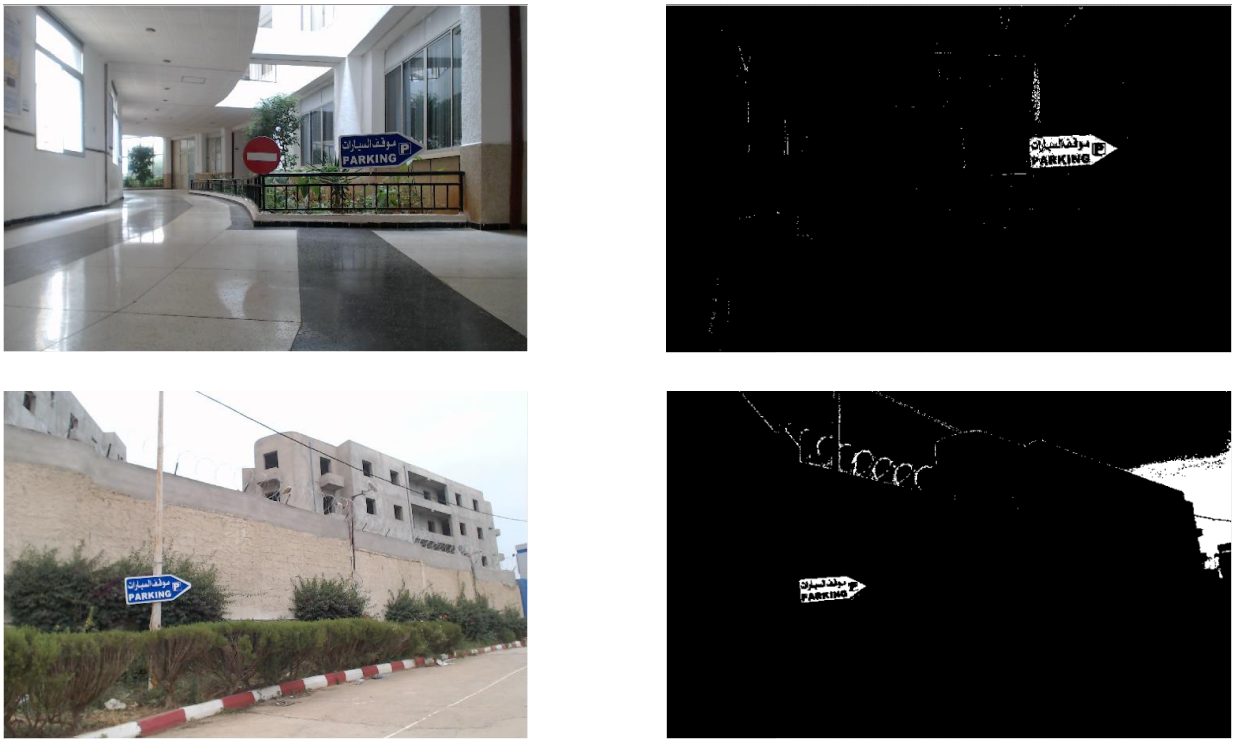


Figure IV.3: Extraction de la couleur bleu dans l'image. (à gauche : image originale, à droite : image binarisée)

Malgré leur principe très simple, les seuillages appliqués pour l'extraction des couleurs rouge et bleu ont montré leur efficacité à travers le test d'un grand nombre d'images contenant des panneaux et qui sont prises des conditions d'éclairage variées.

Remarque :

Les images binaires utilisées pour la reconnaissance des formes des panneaux ne tiennent compte que de la couleur principale du panneau (rouge ou bleu) sans prendre en considération la couleur blanche à l'intérieur du panneau, car nous considérons que l'exploitation de la couleur principale du panneau est suffisante pour le reconnaitre. Si on veut tenir compte des deux couleurs du panneau pour plus de précision, on applique la technique présentée par un organigramme dans la figure (VI.4).

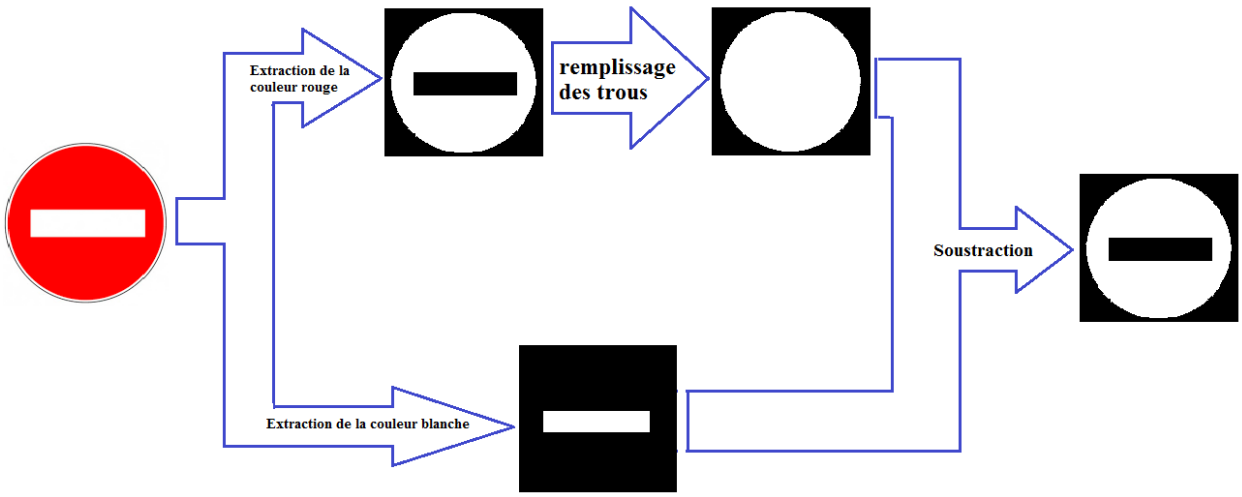


Figure IV.4: Organigramme de la deuxieme technique pour l'extraction du panneau.

IV. 3 Evaluation de la procédure de classification des panneaux :

Afin d'évaluer le processus de classification, on effectue un test de classification sur quelques échantillons (figure IV.5) et on récolte les résultats obtenus pour chaque échantillon selon trois méthodes de classification et en utilisant trois types de distance.

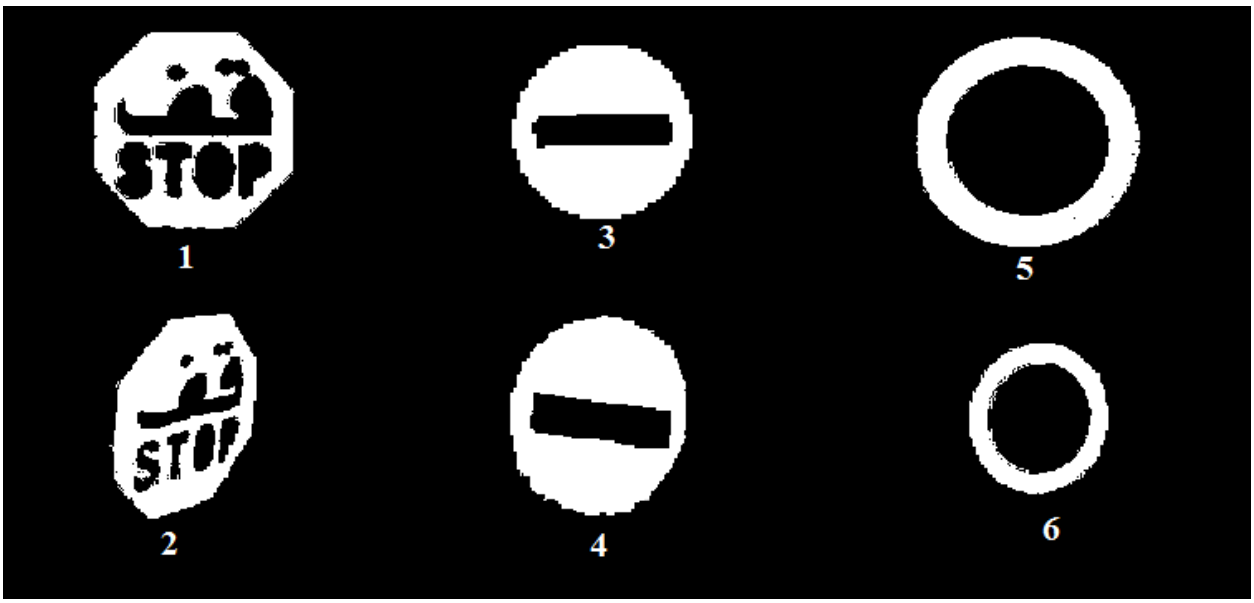


Figure IV.5 : échantillons utilisés pour l'évaluation du classifieur.

On prélève les résultats de classification représentés par le score de classification et le score d'identification pour chaque échantillon, et on compare les résultats donnés par les trois méthodes de classification. (Ces scores varient entre 0 et 1000 et sont défini dans le chapitre II)

Pour la méthode K-PPV, on prend K=3. (K doit être inférieur ou égale au nombre d'échantillons dans chaque classe)

Méthode de classification	La distance utilisée	élément 1	élément 2	élément 3	élément 4	élément 5	élément 6
K-PPV	Max	500 762	500 680	1000 858	1000 853	1000 878	1000 787
	Somme	500 734	1000 747	1000 839	1000 903	1000 945	1000 792
	Euclidienne	500 743	1000 723	1000 852	1000 887	1000 918	1000 804
PPV	Max	564 762	617 680	828 858	800 853	860 878	766 787
	Somme	3 734	548 747	718 839	770 903	945 945	805 792
	Euclidienne	318 743	635 723	791 852	831 887	917 918	819 804
Minimum mean distance	Max	316 452	413 447	572 547	838 837	801 831	680 720
	Somme	154 441	364 517	648 577	792 796	784 738	780 725
	Euclidienne	242 478	427 513	628 602	802 816	802 778	750 711

Tableau IV.1 : Résultats de la classification de quelques échantillons.

Une estimation du temps nécessaire au programme pour faire la classification a donné les indices suivant :

Temps minimale : 26 millisecondes. Temps maximal : 32 millisecondes.

Discussion :

Les résultats du tableau (IV.1) montrent premièrement que tous les échantillons testés sont correctement affectés à leurs classes. Ensuite, la méthode de classification K-PPV donne des scores de classification élevés par rapport aux autres méthodes pour la majorité des échantillons. Les résultats obtenus par la méthode 'minimum mean distance' sont inférieurs par rapport aux autres méthodes. Enfin, une comparaison des résultats selon la distance utilisé ne permet pas de distinguer lequel qui donne de meilleurs résultats.

Une autre expérience montre que les transformations géométriques (rotation, symétrie verticale et horizontale) appliquées aux échantillons testés n'affectent pas les résultats obtenus (Vérification des informations donnée par la documentation LabVIEW et qui sont rapportées dans le chapitre II). Cela peut être considéré comme avantage dans certains cas, mais aussi un inconvénient dans d'autres cas (Comme dans le cas de la classification des panneaux comportant des flèches d'indication de sens).

Nous présentons dans le tableau (IV.2) les résultats de l'évaluation du temps moyen nécessaire pour chaque étape du programme pour l'analyse d'une seule image.

La fonction	**Le temps moyen nécessaire (en millisecondes)**
Seuillage.	210
Suppression des éléments indésirables.	8
Localisation des éléments de l'image	21
Classification des pictogrammes	27
Classification OCR	97

Tableau IV.2 : Estimation de temps d'exécution des différentes fonctions du programme.

Les images suivantes montrent le résultat final de la reconnaissance des panneaux :

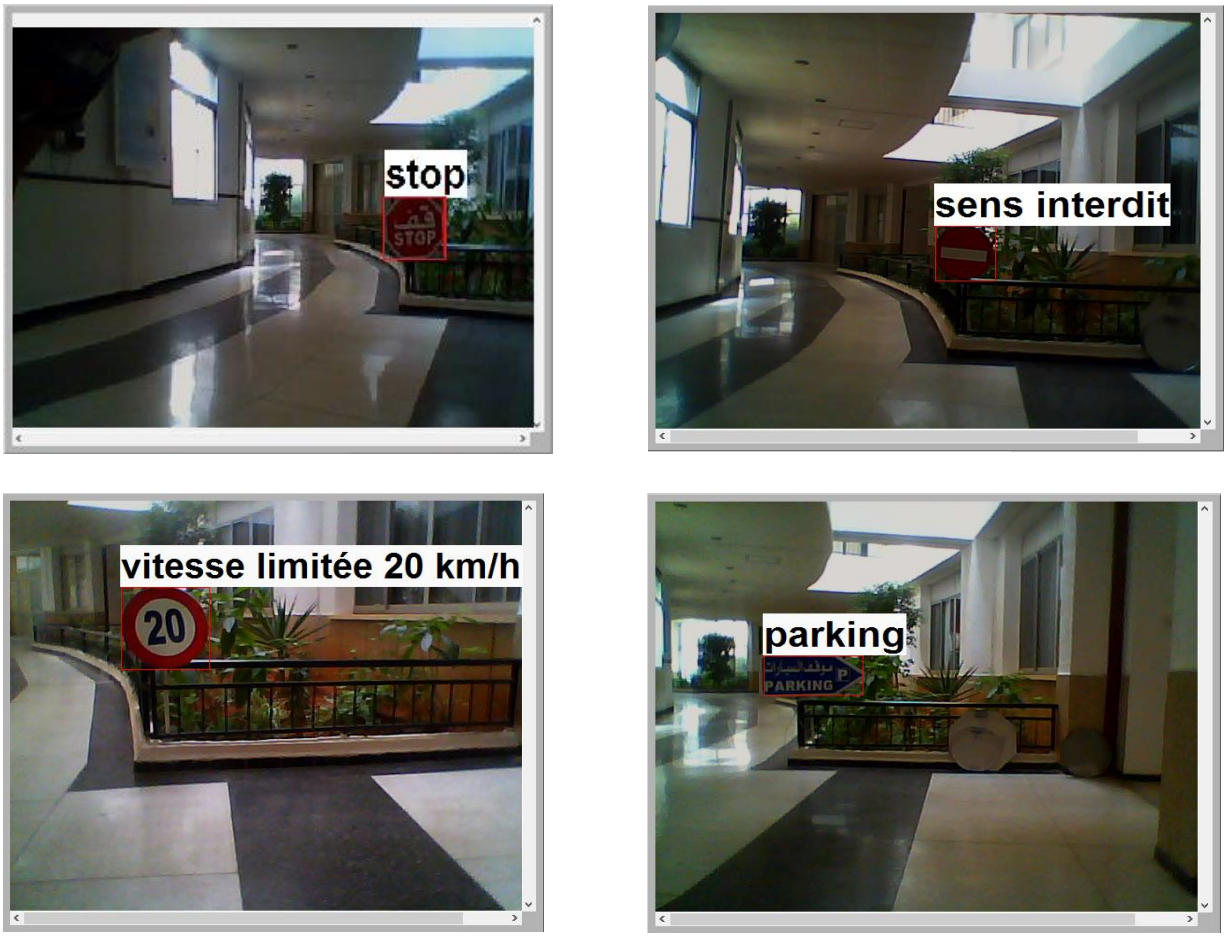


Figure IV.6 : Résultats de reconnaissance des panneaux.

IV.4 Détection des feux tricolores:

Pour évaluer la stratégie élaborée pour la détection des feux tricolores, nous appliquons les différents algorithmes sur des images prises pour des feux tricolores réels (images prises dans les villes de Boumerdès et Alger)

IV.4.1 Test de l'algorithme de seuillage :

L'algorithme de seuillage élaboré permet d'extraire toutes les zones qui ont une forte intensité lumineuse, c'est ainsi que les feux tricolores sont extraits pourvu qu'ils aient une intensité lumineuse relativement élevée.

Figure IV.7 : Résultats d'application de seuillage.

IV.4.2 Test de la fonction (Shape matching) et de la détection de la couleur du feu:

La fonction Shape matching permet de rechercher l'existence d'un élément qui ressemble à un modèle (pattern) désigné auparavant. Dans notre cas, nous recherchons les éléments de forme circulaire dans l'image binaire issue de l'application de seuillage sur l'image originale acquise. Si

l'élément recherché a été trouvé, on prélève les coordonnées de la zone de l'image contenant cet élément (qui peut correspondre à un feu) pour identifier la couleur du feu sur l'image originale.

Plusieurs éléments correspondant à un feu peuvent être détectés dans une seule image, l'algorithme extrait le spectre lumineux de chaque élément, si les valeurs du spectre de l'un de ces éléments correspondent à l'un des critères fixés pour les trois feux, le programme indique de quel feu il s'agit et il désigne son emplacement dans l'image par un rectangle. Le programme est affiné pour ne détecter qu'un seul feu à la fois.

Les tests ont donné les résultats représentés dans les images de la figure (IV.6)

Figure IV.8 : Résultats de détection des feux tricolores.

IV.4.3 Discussion des résultats :

Pour que l'un des feux soit détecté, il doit passer par les trois du programme du programme, si l'une de ces étapes échoue, le feu ne sera pas détecté. On présente ci-dessous une analyse des résultats obtenus dans la figure (IV.6)

Dans l'image (1), il y a deux feux jaunes, mais un seul est détecté, le deuxième n'est pas détecté parce qu'il n'a pas la forme circulaire (partiellement caché deriére la feuille du palmier).

Dans l'image (2), il y a deux détection de feu vert l'un d'eux est vrai feu vert, l'autre est en réalité un afficheur à LEDs vertes qui affiche le temps (en secondes) qui reste pour allumer le feu jaune, il a été puisqu'il possède une forme circulaire dans l'image, c'est un défaut de la résolution de la camera.

Les images (3) et (4) incluent respectivement deux feux rouges et deux feux verts qui sont correctement détectés.

Remarques :

1) Les feux utilisés pendant les expériences au CDTA n'ont pas les performances de feux réels utilisés dans régulation du trafic routier. D'après les expériences, ils ne sont détectables qu'à une distance maximale de sept mètres. Au-delà de cette distance, leurs intensité lumineuse dans l'image prise par la camera.

2) Pour éviter que les feux d'arrières des voitures soit détectés (feux de stop, feux de direction…etc.), on néglige la partie bas de l'image dans le traitement en introduisant un masque dans l'image source.

Le tableau (IV.3) illustre les résultats d'évaluation du temps d'exécution des différentes fonctions du programme de détection des feux tricolores.

La fonction	**Le temps moyen nécessaire (en millisecondes)**
Seuillage.	210
Matching.	238
Identification de la couleur	65

Tableau IV.3 : Estimation de temps d'exécution des différentes fonctions du programme de détection des feux tricolores.

IV.5 Conclusion :

Nous avons présenté dans ce chapitre les résultats d'évaluation des différentes techniques employées pour la détection des panneaux et des feux de signalisation routière.

Nous avons premièrement évalué les techniques de seuillage employées pour l'extraction des couleurs des panneaux, ces techniques sont de principe très simple mais elles ont montré leur efficacité dans notre projet. Ensuite nous avons effectué un test de classification des panneaux qui a donné des résultats très satisfaisants.

Concernant la détection des feux tricolores, elle reste une tache très difficile à résoudre parfaitement malgré de nombreuses approches proposées par des chercheurs à travers le monde. Nous avons testé notre approche sur des images contenant des feux tricolores, le programme réussit à détecter correctement les feux dans la majorité des cas.

Chapitre V

Actions sur le Robucar

V.1 Introduction :

Ce chapitre est consacré au traitement des détections en indiquant au Robucar les taches à exécuter pour chaque détection. On commence par définir les actions requises pour les différentes signalisations, ensuite on va développer un programme sous LabVIEW qui permet de générer les consignes pour le Robucar à travers le contrôleur CompactRIO.

V.2 Définition des actions :

Le tableau (V.1) désigne les taches ordonnées au Robucar en fonction de chaque signalisation détectée.

Panneau détecté	Action sur le Robucar
STOP	Décélérer et arrêter à une distance d_a du panneau pendant une période T_a.
Limitation de vitesse	Si la vitesse du Robucar est supérieure à v_l, décélérer puis maintenir la vitesse à v_l
Sens interdit	Décélérer à une vitesse v_b , faire un braquage (à droite).
Feux rouge	Décélérer et arrêter à une distance d_a du panneau.
Feux jaune	Si la vitesse est inférieure à v_l, et la distance au feu est moins de d_l, Décélérer et arrêter à une distance d_a du panneau.
Feux vert	Si le Robucar est à l'arrêt, accélérer puis maintenir la vitesse à v.

Tableau V.1 : Définition des actions sur le Robucar selon chaque signalisation.

V.3 Elaboration du programme de de génération des consignes :

V.3.1 Rappel théorique :

On suppose que le Robucar est en mouvement rectiligne. Pour programmer les différentes actions, on doit faire appel aux lois de la cinématique :

$$x(t) = x_i + v_i t + \frac{1}{2} a t^2 \qquad \text{(V.1)}$$

$$x(t) = x_i + \frac{v_i + v_f}{2} t \qquad \text{(V.2)}$$

$$v_f^2 = v_i^2 + 2a(x_f - x_i) \qquad \text{(V.3)}$$

Où :

t : est le temps.

$x(t)$: la position à l'instant t.

x_i : la position initiale.

x_f : la position finale.

v_i : la vitesse initiale.

v_f : la vitesse finale.

a : accélération.

V.3.2 Elaboration du programme :

Dans le cadre de notre objectif, on aura besoin des données suivantes

d_p : Distance du panneau ou du feu au moment de la détection.

$d_{a'}$: La distance que doit laisser le Robucar au panneau au moment de l'arrêt.

d_d : La distance de début de décélération.

d_l : La distance limité de détection du feu jaune où il faut s'arrêter.

t_d : le temps à l'instant d'application de la décélération.

d_a : la distance d'arrêt.

t_a: le temps à l'instant d'application de l'accélération.

v : consigne de vitesse.

$v(t)$: consigne de vitesse à l'instant t.

v_l : Vitesse limitée.

$v_{l'}$: Seuil de vitesse maximale au moment de détection du feu jaune où le Robucar doit s'arrêter.

v_b : Vitesse de braquage.

a : L'accélération.

a_d : la décélération. ($a_d < 0$)

$d(t)$: distance parcouru à l'instant t.

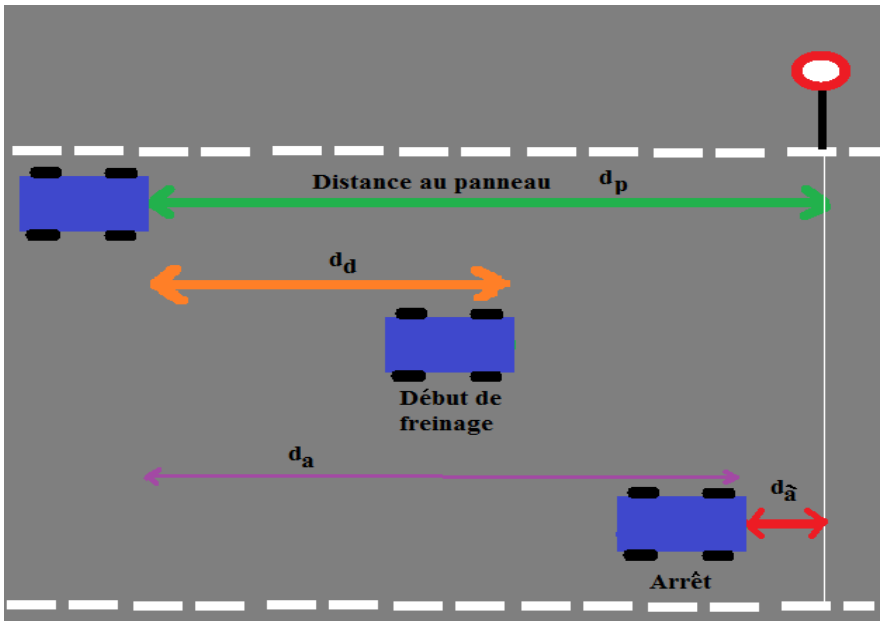


Figure V.1: Règlementation de la vitesse du Robucar face à une signalisation.

D'après la figure (V.1) on a :

$$d_a = d_p - d_{a'} \tag{V.4}$$

Dans l'intervalle de freinage et depuis la formule (V.3) on a :

$$v^2 + 2a_d(d_a - d_d) = 0 \tag{V.6}$$

Ce qui donne :

$$d_d = \frac{v^2}{2a_d} + d_a \tag{V.5}$$

Au moment de début de freinage et depuis l'équation (V.2) on a :

$$d(t_d) = v.t_d = d_d \tag{V.6}$$

On en déduit :

$$t_d = \frac{d_d}{v} \tag{V.7}$$

A l'intervalle de freinage on a

$$a_d(t_a - t_d) + v = 0 \tag{V.8}$$

Donc :

$$t_a = \frac{-v}{a_d} + t_d \tag{V.9}$$

L'organigramme de général du programme est donné comme suit :

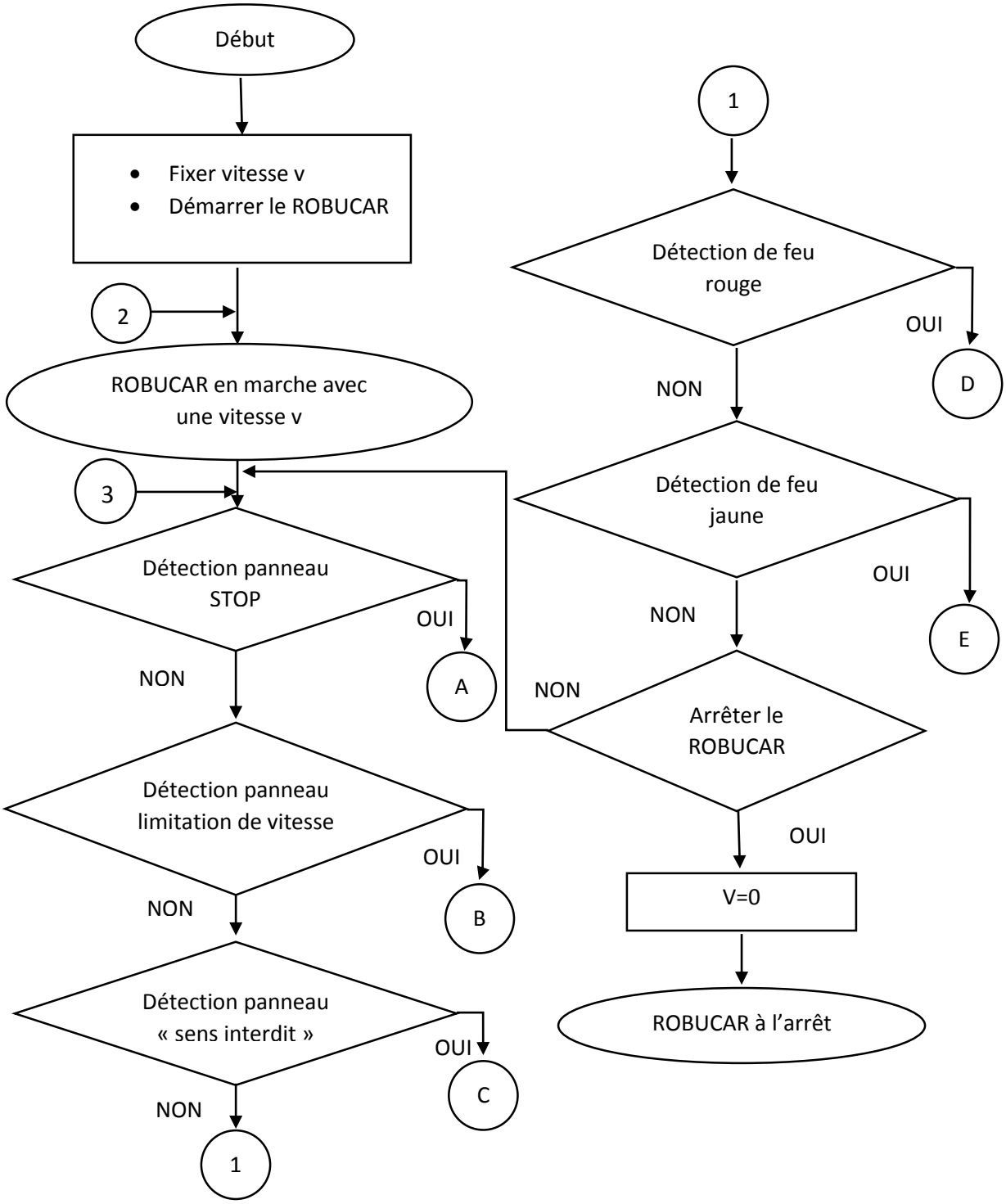


Figure V.2 : Organigramme principal du programme de d'actions.

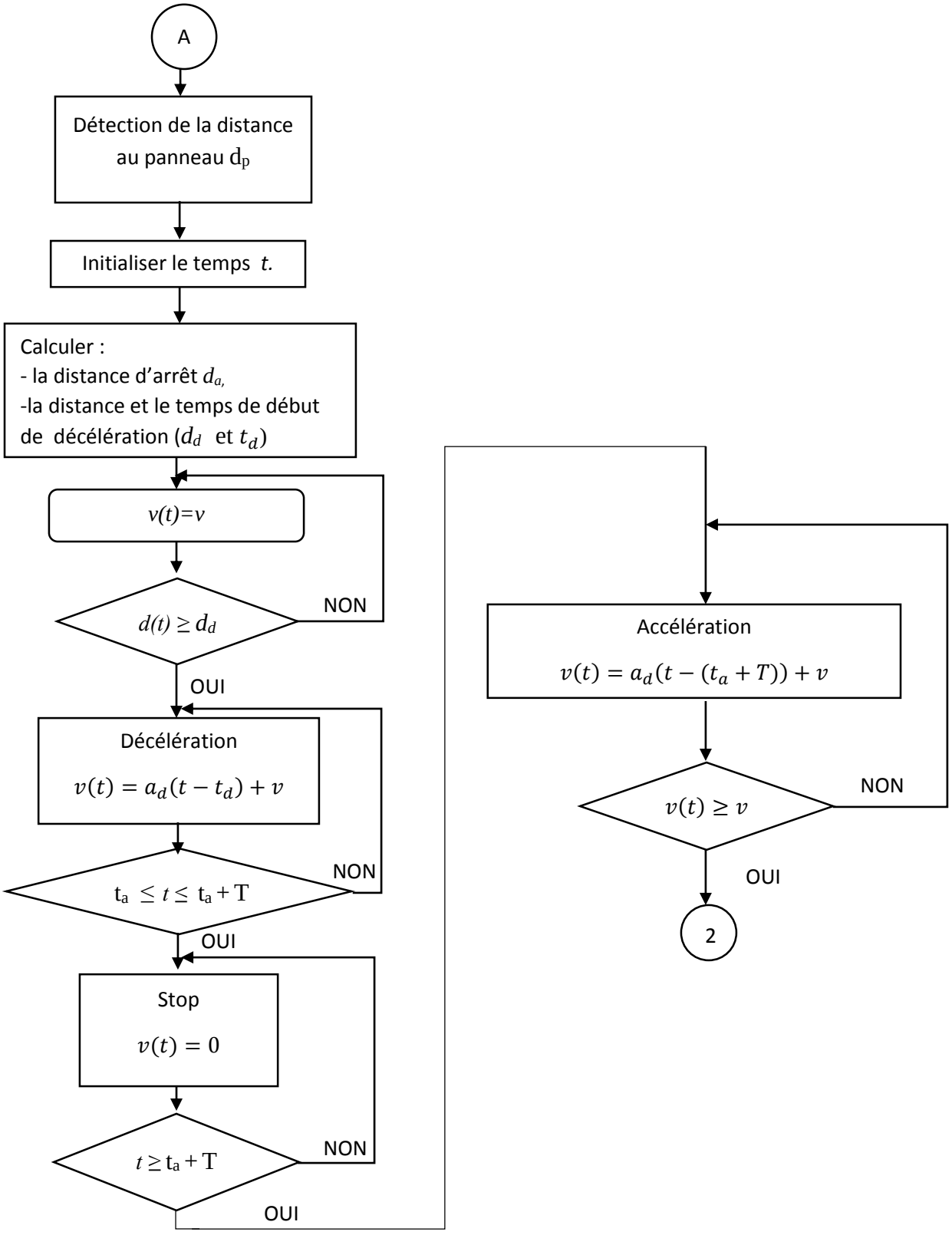


Figure V.3 : Organigramme du sous-programme d'actions sur le Robucar face à une signalisation « stop »

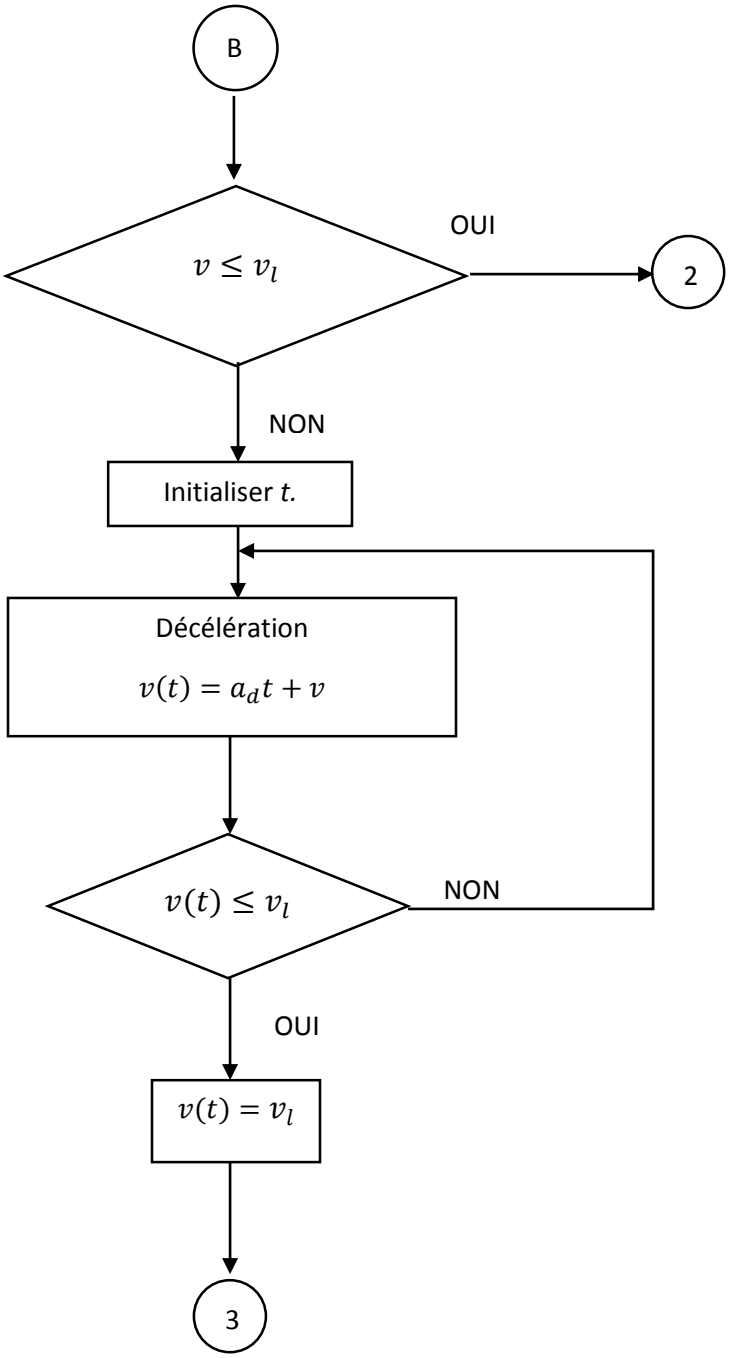


Figure V.4 : Organigramme du sous-programme d'actions sur le Robucar face à une signalisation « Limitation de vitesse »

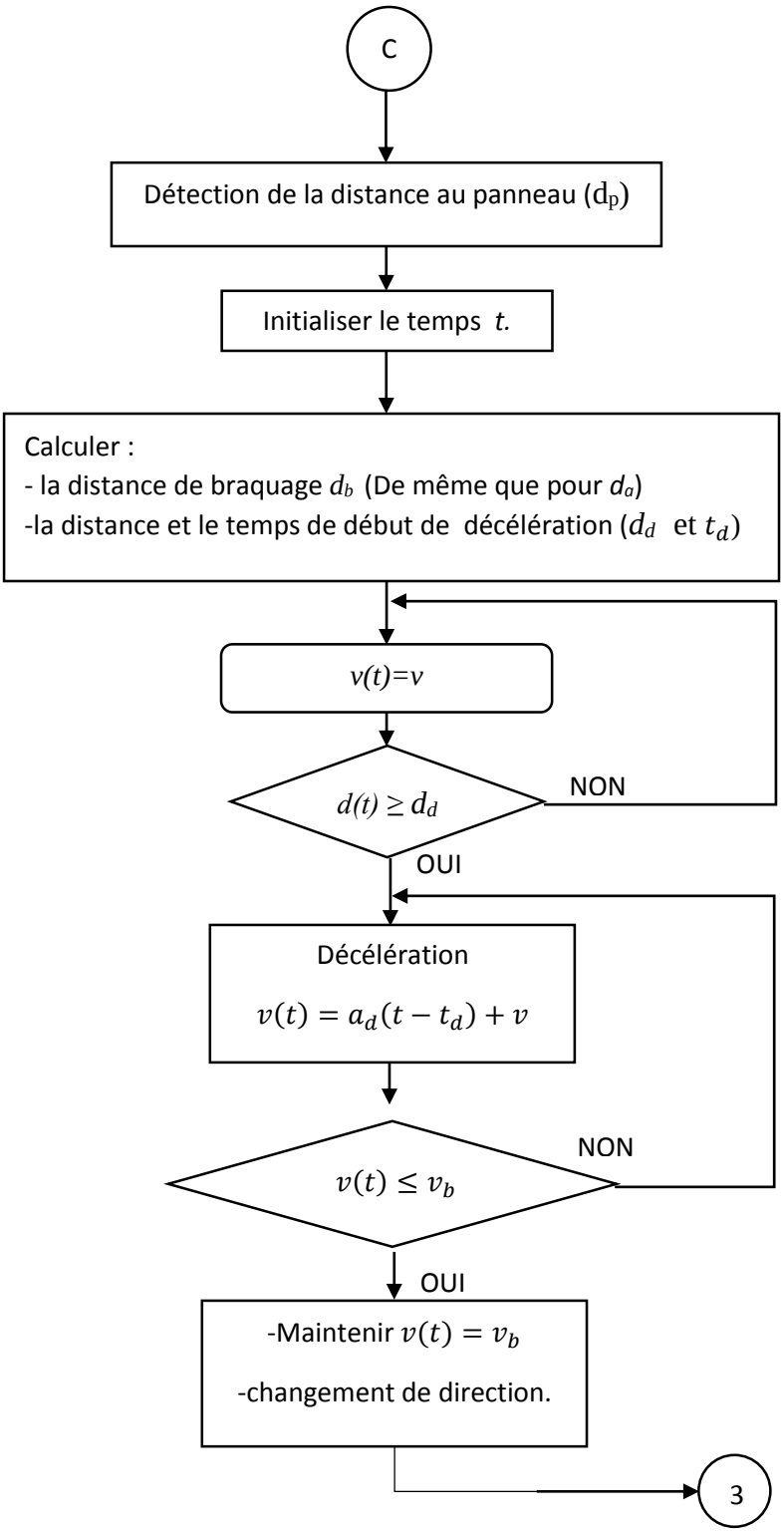


Figure V.5 : Organigramme du sous-programme d'actions sur le Robucar face à une signalisation « Sens interdit ».

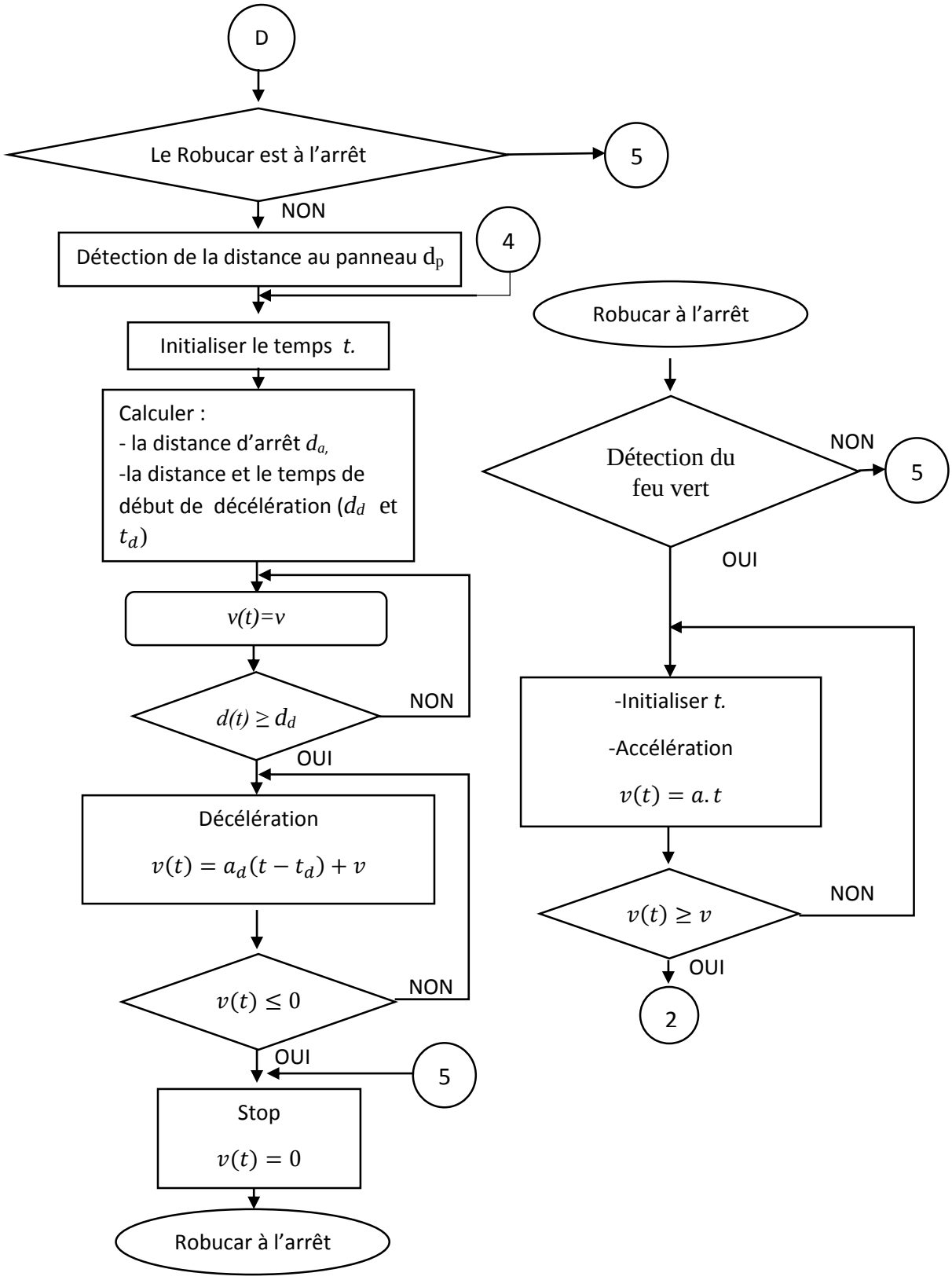


Figure V.6 : Organigramme du sous-programme d'actions sur le Robucar face à une signalisation « feu rouge »

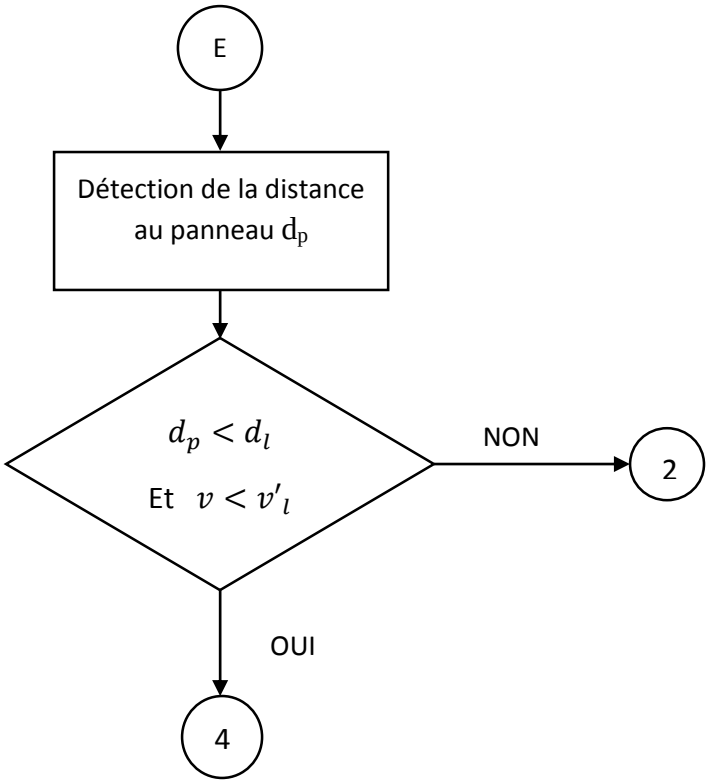


Figure V.7 : Organigramme du sous-programme d'actions sur le Robucar face à une signalisation « feu jaune »

V.4 Simulation :

Avant de tester le programme sur le Robucar, nous avons construit une interface graphique sous LabVIEW pour simuler les différentes actions, cette interface permet de visualiser le comportement du Robucar face à chacune des signalisations à savoir son déplacement en temps-réel ainsi que sa vitesse instantanée. Si dessous on présente des captures d'écran de cette interface :

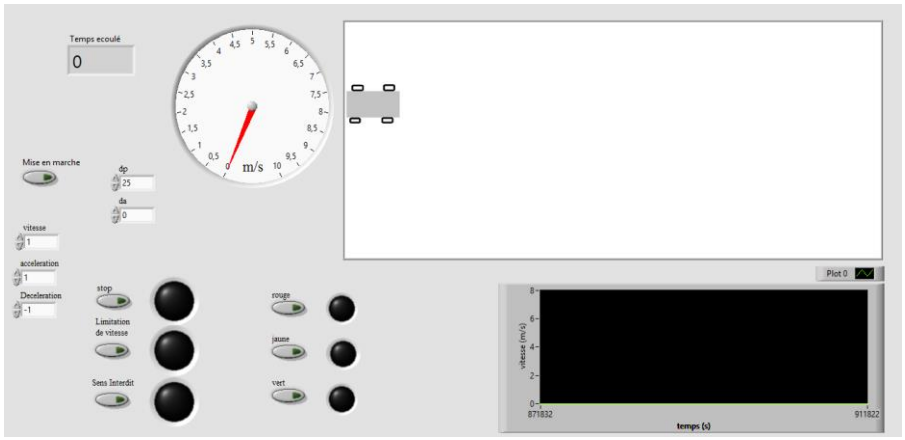


Figure V.8 : Interface de simulation d'actions.

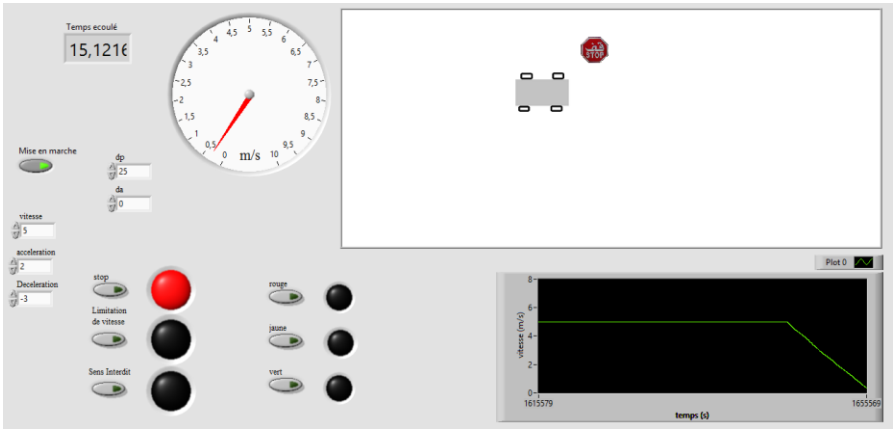


Figure V.9 : Simulation d'une signalisation « stop ».

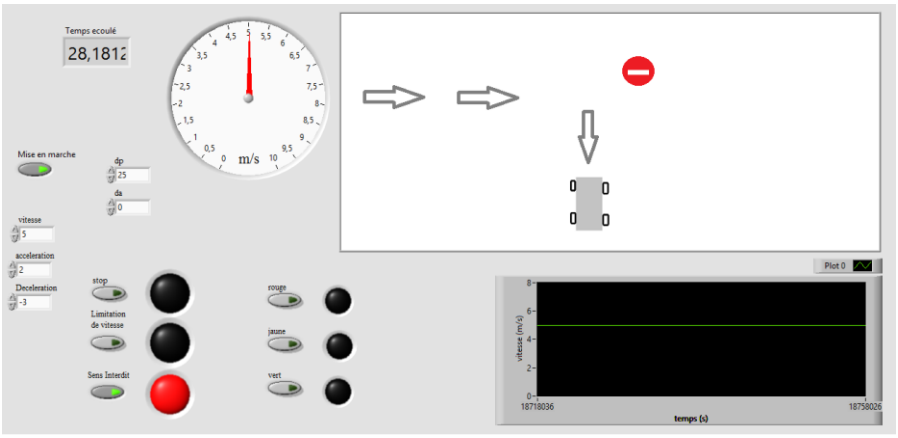


Figure V.10 : Simulation d'une signalisation « Sens interdit ».

V.5 Implémentation sur le Robucar :

Après avoir simulé le programme sur l'interface graphique, nous allons le tester sur le Robucar en utilisant le contrôleur compactRIO. On présente d'abord les étapes de configuration du compactRIO, une fois le programme est implémenté, on présente les résultats des tests.

V.5.1 Configuration du compactRIO :

Le compactRIO emploie le Protocol TCP/IP et en utilisant un câble Ethernet pour communiquer avec le PC. Dans notre cas, on connecte le compactRIO à un routeur sans fil avec un câble Ethernet ainsi, le PC emploie le Protocol DHCP à travers une connexion WI-FI pour communiquer avec le compactRIO. Dans ce cas les opérations de control et de communication des données sont effectuées sans fil ce qui offre plus de flexibilité à notre travail.

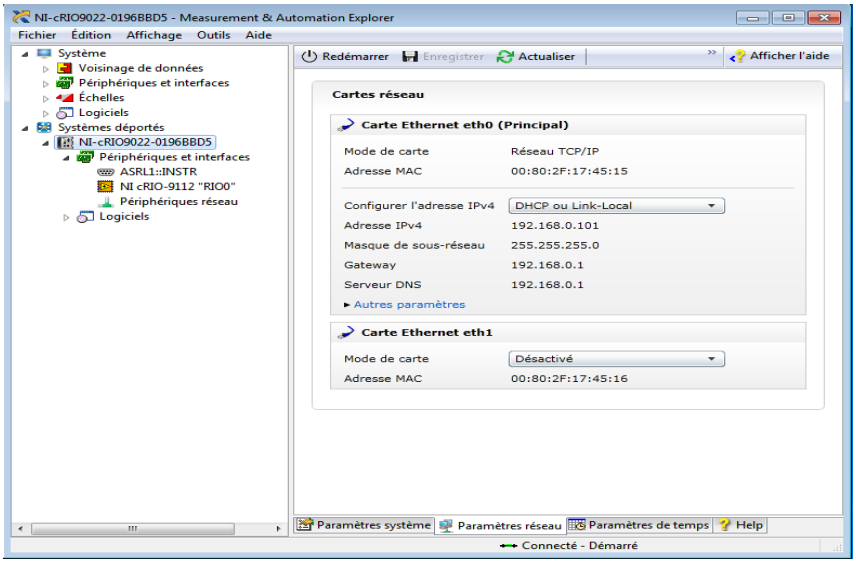


Figure V.11 : Configuration de la communication avec le compactRIO.

Apres avoir établi la connexion, et afin de pouvoir utiliser le compactRIO dans envoyer les consignes au Robucar, on suit les étapes suivantes :

1- On crée un nouveau projet sur LabVIEW dont on va intégrer tous les programmes élaborés, le programme de génération de consignes sera associé à un programme de commande des actionneurs du Robucar à travers un contrôleur PID.

2- On rajoute le compactRIO à la liste des constituants du projet créé, cela s'effectue en sélectionnant l'action ''ajouter une cible'' dans la liste d'options concernant notre projet.

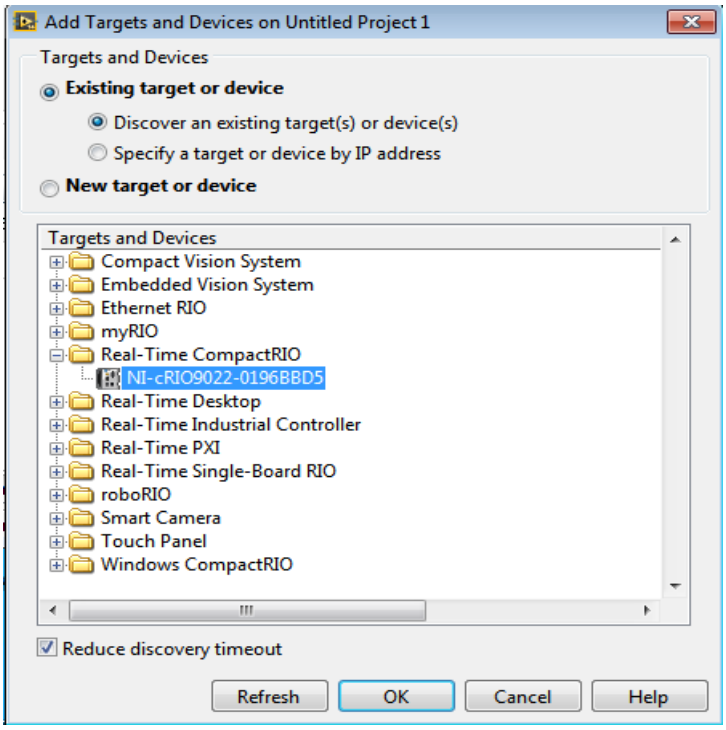


Figure V.12 : Sélection du compactRIO pour dans nouveau projet.

3- Une fenêtre s'ouvre pour choisir le mode de programmation du compactRIO, sélectionne le mode Scan (ces modes de programmations sont expliqués dans le chapitre II).

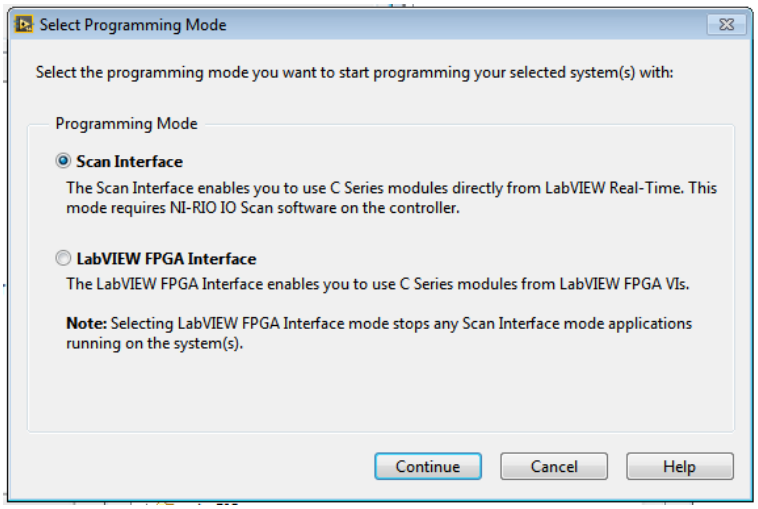


Figure V.13 : Sélection du mode de programmation.

Il reste à sélectionner les modules et nommer les entrées/sorties à utiliser pour l'application.

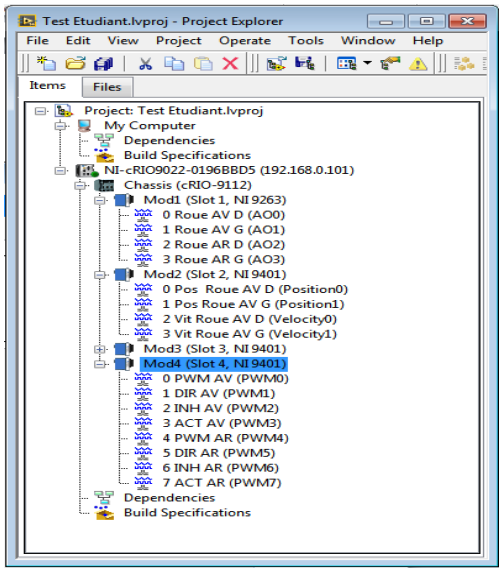


Figure V.14 : Liste des constituants du projet.

V.5.2 Expérimentation avec le Robucar :

Le programme de génération des consignes que nous avons élaboré a été associé à un programme de commande des actionneurs du Robucar en employant un correcteur PID. Des essais ont été effectués pour tester le suivi de consignes par les actionneurs du Robucar. Dans toutes les actions programmées pour la gestion des détections, il s'agit toujours d'une accélération, une décélération ou un arrêt, ces actions ont toutes un rapport avec la vitesse du robot. L'une des signalisations qui nécessites toutes ces actions est la signalisation ''STOP''. La figure (V.15) illustre la consigne de vitesse générée par le programme d'actions (en rouge) et la vitesse réelle du robot calculée par un encodeur (en blanc).

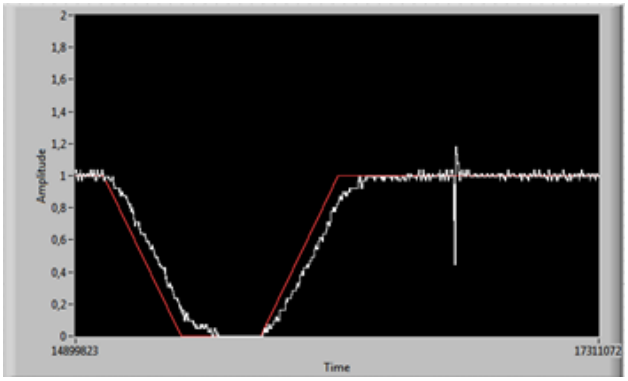


Figure V.15 : Graphe de la vitesse du robot dans une signalisation « stop ».

Le graphe de la figure (V.15) montre la poursuite de l'un des moteurs du robot pour la consigne de vitesse en fixant la consigne de vitesse du robot avant la détection de la signalisation à 1 m/s et la décélération à -0.1 m/s^2 et l'accélération à 0.1 m/s^2 et en fixant la durée d'arrêt à 10 secondes à une distance de 1 mètre du panneaux, le résultat expérimental montre que le Robucar s'arrête après 5.88 secondes de temps prévu pour l'arrêt, parcourant ainsi une distance estimée à 20 centimètres de plus, c'est-à-dire qu'il s'arrête à une distance de 0.8 mètres du panneaux. Ce résultat nécessite une amélioration dans les cas d'une application dans un environnement publique en intervenant sur la boucle d'asservissement des actionneurs du robot.

Un autre test pour une signalisation ''Sens interdit'' a été effectué, la vitesse consigne est fixée au début à 2 m/s (équivalent de 10 radian/s pour les roues du Robucar), et l'angle de braquage est fixé à 18 degrés. Nous avons enregistré les courbes de vitesse et de braquage du robot :

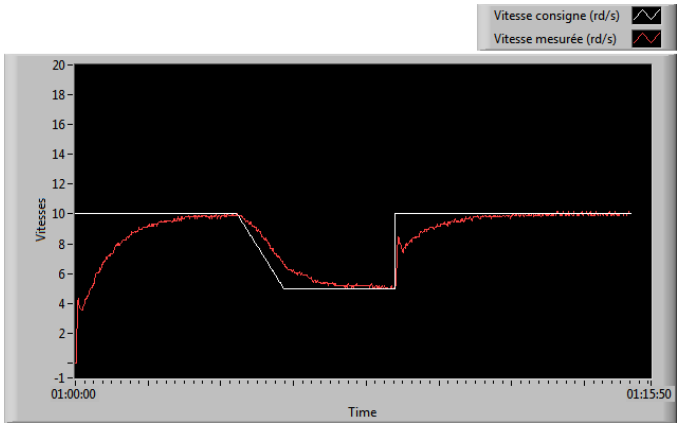


Figure V.16 : Evolution de la vitesse du robot.

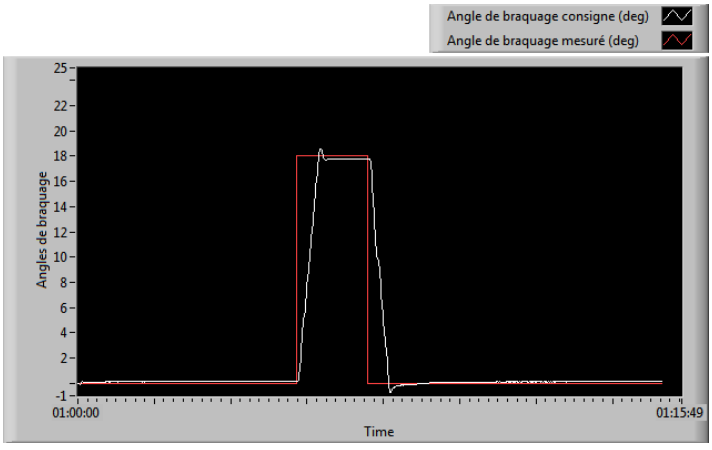


Figure V.17 : évolution de l'angle de braquage du robot.

V.6 Conclusion :

Ce chapitre avait pour but d'implémenter sur le robot un programme pour la gestion des détections, nous avons défini d'abord les actions requises pour chaque type de signalisation, ensuite nous avons donné les organigrammes du programme mis au point sous LabVIEW, ce programme était associé à une interface graphique pour simuler et visualiser le comportement du Robucar face à une détection, et enfin nous avons implémenté le programme sur le CompactRIO pour pouvoir l'appliquer directement au robot. Nous avons effectué des tests sur le Robucar pour voir son comportement réel, les courbes de vitesse et de braquage prélevées montrent que les actionneurs du robot ont un temps de réponse légèrement lent ce qui nécessite l'amélioration de la boucle d'asservissement de ses actionneurs.

Conclusion générale

Ce projet avait pour but d'équiper un robot-véhicule d'un programme fonctionnant en temps réel lui permettant premièrement de détecter et de reconnaitre les panneaux de signalisation routière, ensuite, de traiter les détections et d'agir en conséquence selon la signification de chaque signalisation.

Au début du travail, une familiarisation avec l'outil de programmation LabVIEW ainsi qu'une recherche bibliographique sur les techniques de traitement d'images ont été nécessaires. Ensuite, en croisant les outils offerts par LabVIEW avec les techniques de vision artificielle connues et en tenant compte de la spécificité des problèmes à affronter, nous avons élaboré deux stratégies générales dont la première concerne la détection et la reconnaissance des panneaux routières, et la deuxième pour la détection des feux tricolores.

Le programme de détection et de reconnaissance de panneaux est constitué d'une succession d'opérations à commencer par des opérations d'extraction des couleurs principales des panneaux (rouge et bleu), ces opérations sont obtenues par seuillage d'histogrammes de l'image acquise en choisissant d'abord l'espace de représentation de couleur le plus adéquat. L'opération suivante concerne l'élimination des éléments indésirables de l'image afin d'accélérer les prochaines opérations. Ensuite, nous avons introduit une fonction de localisation des éléments de l'image issue des traitements précédents pour pouvoir enfin les introduire au classifieur qui, par un apprentissage réalisé auparavant avec des échantillons représentant des pictogrammes de panneaux, indique si un de ces éléments correspond à un panneau. Cette opération de classification emploie une approche statistique ainsi que des méthodes de classification connues et largement utilisées dans le domaine de reconnaissance de forme.

Nous avons également employé une technique de reconnaissance des caractères OCR pour la reconnaissance des caractères numériques dans les panneaux de limitation de vitesse. Cette technique est simplifiée sous LabVIEW grâce à une fonction dédiée à cette tâche qui ne nécessite que de faire un apprentissage sur les caractères numériques en introduisant leurs modèle et en indiquant de quel caractère il s'agit. Le classifieur OCR donne des résultats de reconnaissance très satisfaisants.

Pour la détection de feux tricolores, la stratégie repose premièrement sur l'extraction des zones représentant une forte luminosité en employant un seuillage d'histogramme, ensuite sur la recherche de formes circulaires susceptibles d'être des feux tricolores, et enfin, après plusieurs tentatives échouées utilisant les différents espaces colorimétriques pour identifier la couleur du feu, nous avons opté pour une fonction décrivant le spectre lumineux d'une zone de l'image pour identifier la couleur du feu et cela après plusieurs expérimentations sur les feux tricolores pour extraire leurs caractéristiques spectrales et construire des règles d'identification.

Les tests expérimentaux des deux programmes de détection ont montré leur efficacité à détecter correctement les signalisations routières en un temps très court. Sauf quelque cas minimes d'échec ont été enregistrés.

De nombreux problèmes de détection ont été rencontrés avec les feux tricolores utilisés pendant les expérimentations à cause de leur qualité très inférieure à celle des feux utilisés pour la régulation de la circulation. En effet le programme de détection fonctionne mieux avec les feux réels utilisés dans les villes qu'avec ceux qu'on a utilisés dans le laboratoire.

Dans la deuxième partie du projet, le traitement des détections était abordé, nous avons d'abord défini les actions sur le robot selon chaque détection, puis nous avons développé un programme désignant au robot les actions à exécuter selon la signalisation détectée. Un programme de simulation avec une interface graphique a été élaboré pour visualiser le comportement du robot devant une signalisation détectée. Puis, nous avons implémenté le programme sur le contrôleur du Robucar et nous avons effectué quelques tests sur la poursuite des actionneurs du Robucar pour les consignes du programme de traitement des détections de signalisations, les résultats montre une bonne poursuite des consignes, mais nécessite une amélioration au niveau des boucles d'asservissement des actionneurs du Robucar.

Enfin, nous proposons quelques idées pour améliorer ce projet :

- Améliorer les techniques d'extraction de couleur en recherchant d'autres types de seuillage ou de segmentation et en les adaptant aux conditions d'éclairage du milieu, tout en exigeant des algorithmes rapides.
- Elaborer un algorithme pour spécifier les zones de recherche sur l'image en sélectionnant les zones susceptibles de contenir des signalisations (bords de route par exemple)

et éliminer les autres zones (milieu de la route, ciel …), cela dans le but d'accélérer le traitement et éviter les fausses détections.

- Introduire une technique pour la poursuite (tracking) des panneaux détectés dans une séquence d'images et cela pour réduire le nombre d'opérations de traitement sur les images.
- Introduire une commande par logique floue au Robucar pour gérer les réactions aux différentes signalisations détectées puisque c'est une commande proche du raisonnement humain.

Bibliographie

[1] Encyclopédie Microsoft® Encarta® 2010.

[2] Zeroual Dj. *Implémentation d'un environnement parallèle pour la compression d'images à l'aide des fractales.* Mémoire de Magister. Université de Batna, 2006.

[3] Algorithms for Image Analysis. Université of Western Ontario.

[4] Elements of image (pre)-processing and features détection. Université of Western Ontario.

[5] Clément A. *Algorithmes et outils informatiques pour l'analyse d'images couleur.* Thése de Doctorat. Université d'Angers, France. 2002.

[6] https://fr.wikipedia.org/wiki/Rouge_vert_bleu

[7] Antoine MANZANERA Cours TERI – Master 2 UPMC Paris 6.

[8] Airouche M. *Développement de méthodes de segmentation et de suivi d'objets mobiles dans des sequences d'mages d'un système multi-cameras*. Thése de Doctorat. UMBB 2012.

[9] Abdelli Ou. *Segmentation d'images par seuillage d'histogrammes bidimensionnels.* Mémoire de Magister. UMMTO 2011.

[10] Encyclopedia Wikipedia 2012.

[11] Derdour Kh. *Reconnaissance de forms de chiffre arabe imprimé.* Mémoire de Magister. Université de Batna, 2009.

[12] G. Burel, *Introduction au traitement d'images*, Edition Lavoisier, 2001.

[13] M. Kunt, *Reconnaissance des formes et analyse de scènes*, collection électricité, traitement de l'information : volume 3, presses polytechniques et universitaires romandes,2000.

[14] https://fr.wikipedia.org/wiki/LabVIEW

[15] National instruments *Visoin developpement module for LabVIEW.*

[16] Christopher G. Relf. *Image Acquisition and Processing with LabVIEW.* Edition CRC Press, 2004.

[17] http://zone.ni.com/reference/en-XX/help/372916T-01/nivisionconcepts / binary_ particle_ classification/

[18] http://zone.ni.com/reference/en-XX/help/372916T-04/nivisionconcepts / binary_ particle_ classification/

[19] https://www.math.ucdavis.edu/~hunter/m125a/intro_analysis_ch7.pdf.

[20] https://fr.wikipedia.org/wiki/Espace_m%C3%A9trique.

[21] NI LabVIEW for compactRIO developer's guide.

[22] Alexandre Bargeton. Fusion multi-sources pour l'interprétation d'un environnement routier. Ecole Nationale Supérieure des Mines de Paris, 2009.

[23] https://fr.wikipedia.org/wiki/Vision_par_ordinateur.