RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITÉ IBN-KHALDOUN DE TIARET

FACULTÉ DES SCIENCES APPLIQUEES

DÉPARTEMENT DE GENIE ELECTRIQUE

MEMOIRE DE FIN D'ETUDES

Pour l'obtention du diplôme de Master

Domaine : Sciences et Technologie

Filière : Génie Electrique

Spécialité : Informatique industrielle

THÈME

Programmation d'un robot mobile avec évitement d'obstacles

Préparé par : Mr MOUMENE Islam Ibn Eddine
Mr BENTHAMEUR Abdelouahab

Devant le Jury :

Nom et prénoms	Grade	Qualité
BENABID Houari	**MAA**	Président
BENATIA Adda	**MAA**	Examinateur
GOISMI Mohamed	**MAA**	Examinateur 1
MAASKRI Mustapha	**MAA**	Encadreur

PROMOTION 2016 /2017

Remerciements

Nous tenons à exprimer notre gratitude et nos remerciements pour toutes les personnes qui ont contribué à la réalisation de ce projet de fin d'étude.

Nous tenons tout d'abord à remercier notre encadreur Mr **Maaskri** pour son aide, ses conseils, son encouragement et sa disponibilité dans ce projet.

Nous remercions aussi Dr **Chadli Abdelhafid** pour ces aides et ses orientations.

Nous remercions également Mr Alem professeur à l'université de Tiaret pour son assistance.

Nous présentons nos sincères remerciements à tous nos enseignants.

Et enfin, nos profonds remerciements pour les membres de jury qui ont accepté d'évaluer ce travail.

Dédicace

Avant tout je remercie **Allah** *qui nous a aidés à élaborer ce modeste travail que je dédie :*

A la mémoire de mes grands-parents paternels
Qu'ils trouvent ici l'expression de ma reconnaissance et qu'Allah les garde.

A mes grands-parents maternels

Qui m'ont tant aimé et élevé dans le respect des autres, la discipline et la rigueur.

A mes parents
Qui m'ont comblé de leur soutien et m'ont voué un amour inconditionnel.
Vous êtes pour moi un exemple de courage et de sacrifice continu, que cet humble travail témoigne mon affection et mon éternel attachement.

Et spécialement a ma chère et tendre mère

A celle qui a tant souffert sans me faire souffrir qu'elle trouve dans ce mémoire le témoignage de ma reconnaissance de son affection pour les sacrifices, l'extrême amour et de bonté qu'elle m'a offerts.

Et particulièrement à mon père

A qui je dois ma réussite et tout le respect, qu'il trouve ici l'expression de mon affection et une récompense des sacrifices consentis pour moi.

A mes frères Saadeddine et Mourad et Abderrahmane et mes sœurs Souad et Amina et Bouchra et Najet et à toute ma famille

A mes amis de l'université

A tous ceux que J'aime et qui m'aiment et qui me son très cher

Sommaire

Sommaire

Sommaire

Introduction Générale

Depuis la révolution industrielle, Une discipline a marqué l'évolution du monde technologique : La robotique. L'avènement des robots dans l'industrie a permis de soulager l'homme des travaux réplétifs et difficiles tels que : le déplacement d'objet lourds, les taches d'assemblages, les microsoudures…etc. Ceci avec d'efficacité et de précision.

L'humain rêve de créer des machines intelligentes capables d'effectuer des tâches à sa place. Ainsi, les humains auraient plus de temps à consacrer pour leurs loisirs, ou prendraient moins de risques pour effectuer des tâches dangereuses. Or créer une machine pouvant réaliser des tâches que seuls les humains sont normalement capables de faire n'est pas aussi simple qu'on pourrait le penser. En effet, sans toujours y penser, les taches les plus élémentaires de la vie quotidienne d'un humain peuvent de venir extrêmement complexes lorsqu'on les analyses de plus prés

Afin d'être autonome, un robot mobile doit posséder de nombreuses capacités. Premièrement, il doit être capable de percevoir son environnement et de se localiser dans celui-ci pour ce faire, un robot possède des capteurs, comme des sonars et un dispositif à balayage laser servant à mesurer des distances entre lui-même et les obstacles à proximité.

La robotique comporte deux grands pôles d'intérêt la robotique de manipulation (robotique industrielle) et la robotique mobile. Un des problèmes majeurs de la robotique mobile est la planification de mouvement. Autour de ce problème de planification de mouvement de nombreuses études ont été réalisées dans le but de développer des méthodes générales pour guider les robots.

Une fois localiser son environnement le robot doit être capable de se déplacer d'un point à l'autre en trouvant des chemins efficaces et sécuritaires afin d'éviter des collisions avec les obstacles.En plus de pouvoir percevoir globalement son environnement, un robot doit souvent être capable d'identifier des objets

Dans le cadre ce travail, nous allons aborder le problème de la navigation réactive dans un environnement inconnu et on va étudier l'apport de la logique flou dans ce type de navigation et comment prendre les meilleures décisions pour attendre la cible.

Ce mémoire est organisé comme suit

Le premier chapitre nous abordons quelques notions de base nécessaires à la compréhension du domaine de la robotique mobile. Ensuite nous présentons les différents types de robots mobiles, et décrit les domaines d'applications robotiques. Le deuxième chapitre présente une introduction sur la logique floue .Le troisième chapitre nous allons présenter la plateforme de développement tell que Java et éclipse .Le quatrième chapitre illustre notre application par une simulation de Robot

Enfin, nous terminons notre mémoire par une conclusion générale et des annexes.

Chapitre 1
La robotique

1. Introduction

La robotique est un ensemble de disciplines (mécanique, électronique, automatique, informatique), elle se subdivise en deux types : les robots industriels et les robots mobiles. Les robots industriels sont généralement fixes, ils sont utilisés dans des nombreuses applications industrielles: l'assemblage mécanique, la soudure, la peinture... Les robots mobiles ne sont pas fixes, ils sont classifies selon la locomotion en robots marcheurs, à roues, à chenilles... comme ils peuvent être classifié selon le domaine d'application en robots militaires, de laboratoire, industriels et de services.[19]

Les robots mobiles présentent un cas particulier en robotique. Leur intérêt réside dans leur mobilité", destinés à remplir des taches pénibles (exemple : transport de charges lourdes) et ils travaillent même en ambiance hostile (nucléaire, marine, spatiale, lutte centre l'incendie, surveillance...). [19]

L'aspect particulier de la mobilité impose une complexité technologique (capteurs, motricité, énergie) ct méthodologique tel que le traitement des informations par utilisation des techniques de l'intelligence artificielle ou de processeurs particuliers (vectoriels, cellulaires).

L'autonomie du robot mobile est une faculté qui lui permet de s'adapter ou de prendre une décision dans le but de réaliser une tache même dans un environnement peu connu ou totalement inconnu.[19]

2. Généralités sur les robots

2.1 Les origines :

Étymologie : origine tchèque « robota » (travail).

Définition : *un robot est un système mécanique poly articulé mû par des actionneurs et commandé par un calculateur qui est destiné à effectuer une grande variété de tâches.*

2.2 Historique :

La robotique est passée par plusieurs générations comme suit:

- 1947: Premier manipulateur électrique télé-opéré.
- 1954: Premier robot programmable.
- 1961: Utilisation d'un robot industriel, commercialisé par la société UNIMATION (USA), sur une chaîne de montage de General Motors.
- 1961: Premier robot avec contrôle en effort.
- 1963: Utilisation de la vision pour commander un robot.
- 1978 : Le robot ARGOS. Développé à l'Université Paul Sabatier de Toulouse (France). Le robot ARGOS simule la navigation d'un robot mobile équipé d'un système de vision au fur et

à mesure de ses déplacements.

- 1979: Le robot HILARE. Les chercheurs du L.A.A.S. [1]de Toulouse (France) étudièrent la planification des trajectoires d'un robot mobile ponctuel, dans un environnement totalement connu.
- 1981: Le robot VESA. Ce robot, construit à l'I.N.S.A [2](France). de Rennes, est équipée d'un arceau de sécurité pour réaliser la détection d'obstacles dans un environnement totalement inconnu.
- 1984: Le robot FLAKEY. Ce robot, conçu et construit au Stanford Research Institute et le reflet des améliorations apportées par 14 années de développement. Le robot FLAKEY est équipé de deux roues motrices avec encodeurs, mais sa vitesse maximale est de 66 cm/s au lieu de quelques centimètres par seconde. Ce robot est capable de naviguer dans des environnements réels.
- 1993: Les robots ERRATIC et PIONNER. Le robot ERRATIC a été conçu par Kurt Konolige, au Stanford Research Institue, comme un robot mobile de faible coût pour ses cours de robotique.
- Les robots mobiles actuels : A présent la plupart des travaux de recherche portent sur les problèmes de perception. La planification de trajectoires, l'analyse et la modélisation de l'environnement de robot, appliqué sur des robots mobiles commerciaux. Également la recherche actuelle sur la conception mécanique des robots mobiles pour des applications hautement spécialisées, comme l'exploration sous-marine, les robots volants et le micro robots.[19]

3. Robot industriel

3.1 Présentation

La **robotique industrielle** est officiellement définie par l'ISO comme un contrôle automatique, reprogrammable, polyvalent manipulateur programmable dans trois ou plusieurs axes. Les applications typiques incluent des robots de soudage, de peinture et d'assemblage. La robotique industrielle inspecte les produits, rapidement et précisément. Les robots industriels sont beaucoup utilisés en automobile. Leur conception nécessite une bonne connaissance et un très haut niveau dans le domaine de l'ingénierie.[3]

3.1 Robot manipulateur industriel

(ISO 8373) : une machine, un mécanisme constitué normalement d'une série de segments qui sont reliés par un joint assurant une rotation ou une translation relative entre segments, dont le but est de prendre et déplacer des objets (pièces ou outils) avec plusieurs degrés de liberté. Il peut être commandé par un opérateur, une unité de commande électronique ou un système logique (dispositif à cames relais câbles etc).[3]

[1] Laboratoire d'analyse et d'architecture des systèmes
[2] L'institut national des sciences appliquées

3.2 Définition d'un robot industriel

Un robot industriel est un système poly articulé à l'image d'un bras humain souvent composé de 6 degrés de liberté, 3 axes destinés au positionnement et 3 axes à l'orientation permettant de déplacer et d'orienter un outil (organe effecteur) dans un espace de travail donné.

On peut distinguer :

- les robots de peinture ou soudure largement utilisés dans l'industrie automobile,
- les robots de montage de dimension souvent plus réduite,
- les robots mobiles destinés à l'inspection souvent associé à de l'intelligence artificielle et capables, dans certains cas, de prendre en compte l'environnement.

Un robot se compose d'une partie mécanique, le bras lui même, d'une armoire de commande composée d'une unité centrale qui pilote les électroniques de commande d'un ou plusieurs axes qui en assure l'asservissement, de variateurs de vitesse et d'un langage de programmation spécialisé qui permet de commander le robot (LM développé par l'Ensimage Grenoble, langage Adepte type basic) qui intègre un transformateur de coordonnées pour transformer une valeur cartésienne en données codeur du moteur.

Certains robots disposent d'un mode d'apprentissage qui permet de répéter les mouvements réalisés librement à la main, l'élément essentiel étant la fidélité la capacité du robot à atteindre successivement la même position dans une tolérance définie, une procédure de calibration permet de reprendre le zéro de chacun des axes. Ils peuvent être associés à un système de vision artificielle qui leur permet de corriger les déplacements.

Pour des raisons de sécurité, ces robots sont protégés par des cages ou des carters pour interdire à l'homme de les approcher de trop près. [3]

3.3 Domaines d'utilisation

- Domaine de l'exploration
  - Accès difficile
  - Nettoyage
  - Espace
  - Démantèlement nucléaire
  - Déminage
  - Chantier sous- marin…

- ❖ Domaine du Médical
  - ▪ Assistance aux opérations chirurgicales
  - ▪ Robotique médicale
- ❖ Domaine du Service
  - ▪ Robots Humanoïdes
- ❖ Domaine industriel
  - ▪ Robots industriels

3.4 Types de robots industriels

• Certains robots sont programmés pour exécuter fidèlement des actions spécifiques répétitives. Ils sont programmés avec un haut degré de précision.

• D'autres robots sont beaucoup plus flexibles. Ils sont par exemple utilisés en peinture. L'intelligence artificielle est en passe de devenir un facteur important dans la robotique industrielle. [3]

4. Robot Mobile

Contrairement au robot industriel qui est généralement fixé, le robot mobile est doté de moyens qui lui permettent de se déplacer dans son espace de travail. Suivant son degré d'autonomie ou degré d'intelligence, il peut être dote de moyens de perception et de raisonnement. Certains sont capables, sous contrôle humain réduit, de modéliser leur espace de travail et de planifier un chemin dans un environnement qu'ils ne connaissent pas forcément d'avance.

Actuellement, les robots mobiles les plus sophistiqués sont essentiellement orientés vers des applications dans des environnements variables ou incertains, souvent peuplés d'obstacles, nécessitant une adaptabilité à la tâche. La figure (1.1) illustre la structure d'un tel robot [3].

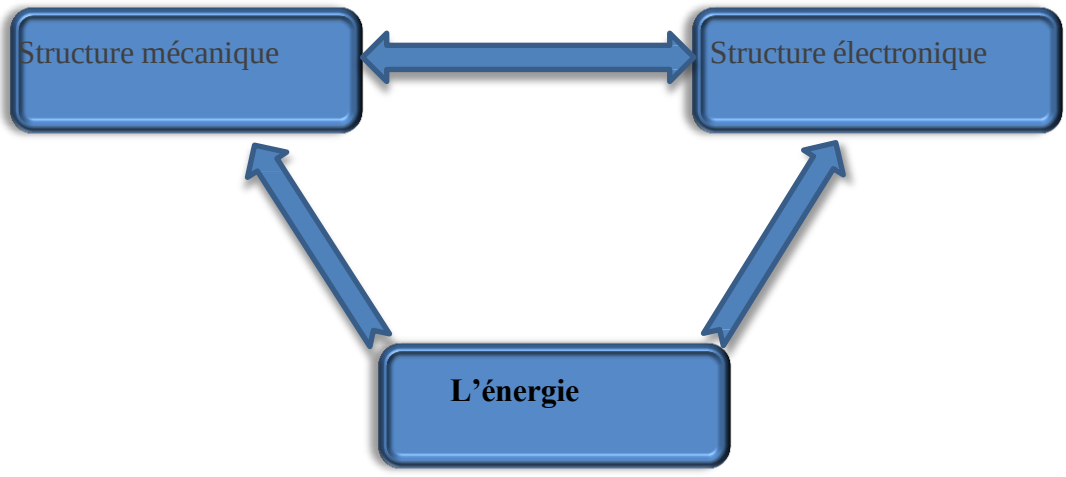


Figure 1.1 : Structure d'un robot mobile

Un A.G.V (Automated Guided Vehicule) ou véhicule à guidage automatique est un cas particulier d'un robot mobile voué à des applications purement industrielles. II est parfois appelé chariot automatique et il est doté d'un équipement de guidage automatique qu'il soit inductif (filoguidé), optique, électromagnétique ou autre. Ce type de véhicule est capable de suivre des chemins prédéfinis et programmables ou de planifier ses propres trajectoires selon le type de guidage et de navigation utilisée. Les A.G.V se présentent sous des aspects assez variés, que ce soit par leur forme, par leur taille ou par leur poids. [3]

4.1 Architecture des robots mobiles

En général un robot mobile est constitué dc trois structures :

1- **Structure mécanique**: elle assure le mouvement du robot par des roues motrices placées selon le type de mouvement et la précision de la tache voulue.

2- **Structure instrumentale** : un robot est équipé d'un certain nombre de capteurs de sécurité afin de leur donner une certaine connaissance de l'environnement. Selon l'application, les capteurs peuvent être :

- **Capteurs infrarouges**

Les capteurs infrarouges sont constitués d'un ensemble émetteur/récepteur fonctionnant avec des radiations non visibles, dont la longueur d'onde est juste inférieure à celle du rouge visible. La mesure des radiations infrarouges étant limitée et, en tout état de cause, la qualité très dégradé d'un mètre, ces dispositifs ne servent que rarement de télémètres.

On les rencontrera le plus souvent comme détecteurs de proximité, Ou dans un mode encore plus dégradé de présence. [3]

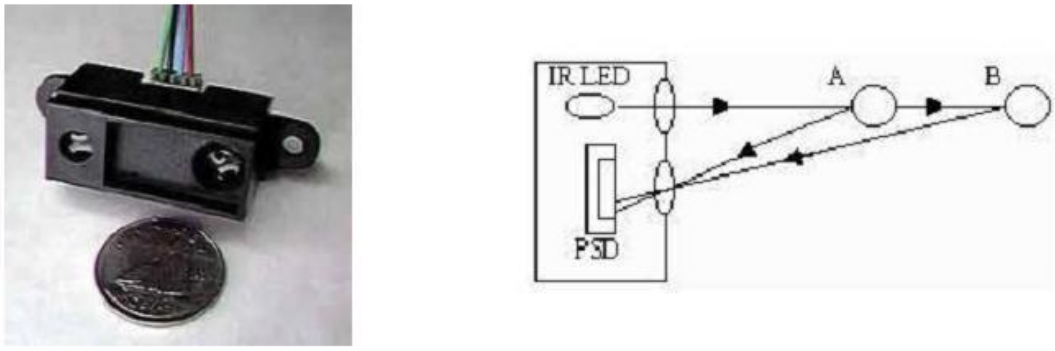


Figure 1.2 : Télémètres infrarouges.

- **Capteurs ultrasonores**

Les capteurs ultrasonores utilisent des vibrations sonores dont les fréquences ne sont pas perceptibles par l'oreille humaine. Les fréquences couramment utilisées dans ce type. De

technologie vont de 20 kHz à 200 kHz. Les ultrasons émis se propagent dans l'air et sont réfléchis partiellement lorsqu'ils heurtent un corps solide, en fonction de son impédance acoustique. L'écho en retour prend la forme d'une onde de pression à l'image des vaguelettes circulaires déformant la surface de l'eau lorsqu'on y jette une pierre. [3]

La distance entre la source et la cible peut être déterminée en mesurant le temps de volume séparant l'émission des ultrasons du retour de l'écho.

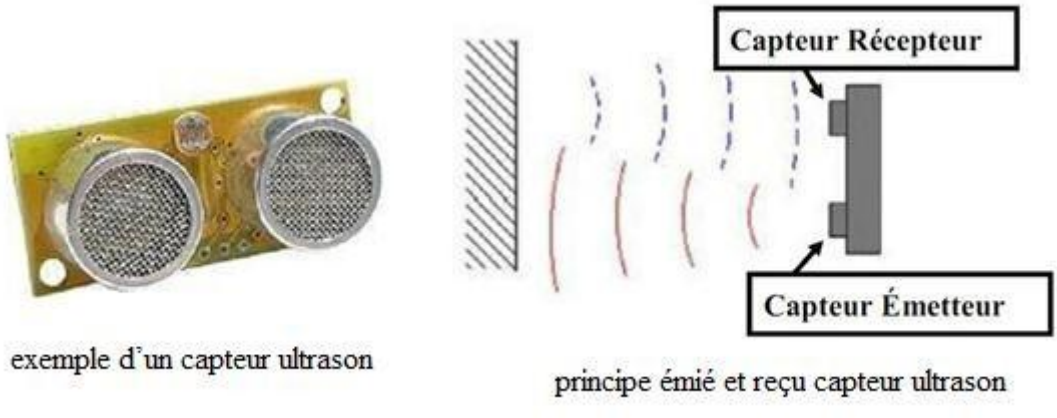


Figure 1.3 : Télémètres ultrasonores.

- **Télémètre laser**

Les télémètres laser (Figure 1.4) sont à ce jour le moyen le plus répandu en robotique mobile pour obtenir des mesures précises de distance. Leur principe de fonctionnement est le suivant:[5]

A un instant donné, une impulsion lumineuse très courte est envoyée par l'intermédiaire d'une diode laser de faible puissance. La réflexion de cette onde donne un écho qui est détecté au bout d'un temps proportionnel à la distance capteur obstacle. La direction des impulsions est modifiée par rotation d'un miroir. Par l'angle de balayage couvrant généralement entre 100 et 180 degrés sur des produits commerciaux.

Figure 1.4 : Exemple d'un télémètre laser.

- **Les caméras**

L'utilisation d'une caméra pour percevoir l'environnement est une méthode attractive car elle semble proche des méthodes utilisées par les humains. Le traitement des données volumineuses et complexes fournies par ces capteurs reste cependant difficile à l'heure actuelle, même si cela reste une voie de recherche très explorée. [19]

- **Les capteurs tactiles**

Les robots peuvent être équipés de capteurs tactiles, qui sont le plus souvent utilisés pour des arrêts d'urgence lorsqu'il rencontre un obstacle qui n'avait pas été détecté par le reste du système de perception. Ces capteurs peuvent être de simples contacteurs répartis sur le pourtour du robot. Il ne détecte alors le contact qu'au dernier moment.

Il est également possible d'utiliser des petites tiges arquées autour du robot pour servir d'intermédiaire à ces contacteurs, ce qui permet une détection un peu plus précoce et donne ainsi plus de marge pour arrêter le robot.[19]

- **Les boussoles**

Les boussoles permettent, par la mesure du champ magnétique terrestre, de déduire la direction du nord. Ces capteurs peuvent utiliser différentes technologies et ont l'avantage de fournir une direction de référence stable au cours du temps (au contraire des gyroscopes qui dérivent). Ces capteurs sont toutefois très délicats à utiliser en intérieur car ils sont très sensibles aux masses métalliques présentes dans la structure des bâtiments. En pratique, on les utilise donc principalement en extérieur en apportant le plus grand soin à leur positionnement sur le robot pour éviter les influences des composants du robot, notamment les moteurs électriques.[19]

- **Les balises**

Dans certaines applications, il est également possible d'utiliser des balises dont on connaît la position, et qui pourront être facilement détectées par le robot, afin de faciliter sa localisation. Des techniques très diverses peuvent être utilisées pour ces balises. On peut par exemple utiliser un signal radio, émis de manière omnidirectionnel par la balise.[19]

Le robot sera alors équipé d'une antenne directionnelle qui lui permettra de détecter la direction des différentes balises, afin de déduire sa position par triangulation. On peut également utiliser des codes couleurs où des codes barrent qui pourront être détectés par une caméra. [19]

- **Le GPS** (Universel Placement Système en Anglais : Global Positionning System)

Un système de balises dont on a placé les balises sur des satellites en orbite terrestre et qui est par conséquent accessible de quasiment partout à la surface du globe. Ce système permet donc d'avoir une mesure de sa position dans un repère global couvrant la terre avec une précision variant de quelques dizaines de mètres à quelques centimètres suivant les équipements. [5]

3- Structure informatique : une commande numérique est impérative, afin de bien analyser les différentes informations, soit du système de perception ou de localisation. Cette commande peut être à base d'un microprocesseur ou microcontrôleur.[5]

4.2 Classification des Robots Mobiles

On peut classer les robots mobiles selon leur degré d'autonomie, système de locomotion, leur domaine d'application, leur système de localisation, l'énergie utilisée...

Nous allons présenter ici quatre classifications qui semblent être les plus intéressantes

4.2.1 Classification selon le degré d'autonomie

Un robot mobile autonome est un système automoteur doté de capacités décisionnelles et de moyens d'acquisition et de traitements de reformation qui lui permettent d'accomplir sous contrôle humain réduit un certain nombre de tâches, dans un environnement non complètement connu. On peut citer quelques types : [5]

• Véhicule télécommande par un opérateur

Ces robots sont commandés par un opérateur qui leurs impose chaque tache élémentaire à réaliser.

• Véhicule télécommandé au sens de la tâche à réaliser Le véhicule contrôle automatiquement ses actions.

• Véhicule semi-autonome

Ce type de véhicule réalise des tâches prédéfinies sans 1'aidc de 1'opdrateur.

• Véhicule autonome

Ces derniers réalisent des tâches semi-définies.

4.2.2 Classification selon le type de locomotion

Selon le système de locomotion, on peut distinguer quatre types des robots:

A. Les robots mobiles à roues

La mobilité par roues est la structure mécanique la plus utilisée. Ce type de robot assure

un déplacement avec une accélération et une vitesse rapide mais nécessite un sol relativement plat. On distingue plusieurs classes de robots à roues déterminées, principalement, par la position et le nombre de roues utilisées. Nous citerons ici les quatre classes principales de robots à roues. [5], [8]

- **Robot uni-cycle**

Un robot de type uni cycle est actionné par deux roues indépendantes, il possédant éventuellement des roues folles pour assurer sa stabilité. Son centre de rotation est situé sur l'axe reliant les deux roues motrices. C'est un robot non-holonome, en effet il est impossible de le déplacer dans une direction perpendiculaire aux roues de locomotion. Sa commande peut être très simple, il est en effet assez facile de le déplacer d'un point à un autre par une suite de rotations simples et de lignes droites.(Figure 1.5) [19]

Figure 1.5: Robot de type uni-cycle.

- **Robot tricycle**

Un robot de type tricycle est constitué de deux roues fixes placées sur un même axe et d'une roue centrée orientable placée sur l'axe longitudinal. Le mouvement du robot est donné par la vitesse des deux roues fixes et par l'orientation de la roue orientable. Son centre de rotation est situé à l'intersection de l'axe contenant les roues fixes et de l'axe de la roue orientable. C'est un robot non-holonome. En effet, il est impossible de le déplacer dans une direction perpendiculaire aux roues fixes. Sa commande est plus compliquée. Il est en général impossible d'effectuer des rotations simples à cause d'un rayon de braquage limité de la roue orientable. (Figure 1.6) [19]

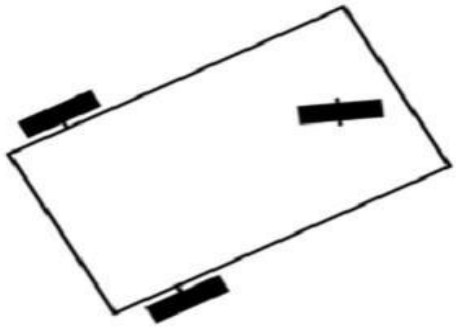


Figure 1.6 : Robot de type tricycle.

- **Robot voiture**

Un robot de type voiture est semblable au tricycle, il est constitué de deux roues fixes placées sur un même axe et de deux roues centrées orientables placées elles aussi sur un même axe (Figure1.7).

Le robot de type voiture est cependant plus stable puisqu'il possède un point d'appui supplémentaire. Toutes les autres propriétés du robot voiture sont identiques au robot tricycle, le deuxième pouvant être ramené au premier en remplaçant les deux roues avant par une seule placée au centre de l'axe, et ceci de manière à laisser le centre de rotation inchangé. [19]

Figure 1.7 : Robot de type voiture.

- **Robot omnidirectionnel**

Un robot omnidirectionnel est un robot qui peut se d´placer librement dans toutes les directions. Il est en général constitué de trois roues décentrées orientables placées en triangle équilatéral. [19]

L'énorme avantage du robot omnidirectionnel est qu'il est holonome puis qu'il peut se déplacer dans toutes les directions. Mais ceci se fait au dépend d'une complexité mécanique bien plus grande (Figure 1.8).

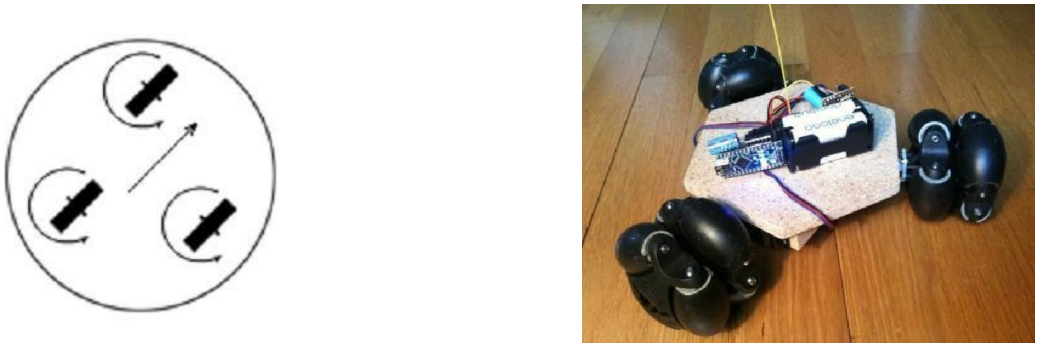


Figure 1.8 : Robot mobile omnidirectionnel.

B. Les robots mobiles à chenilles

L'utilisation des chenilles présentées 1'avantage d'une bonne adhérence au sol .et d'une faculté de franchissement d'obstacles. L'utilisation est orienté vers l'emploi sur sol accidenté ou dc mauvaise qualité au niveau dc l'adhérence (présence de boue, herbe,..). [19]

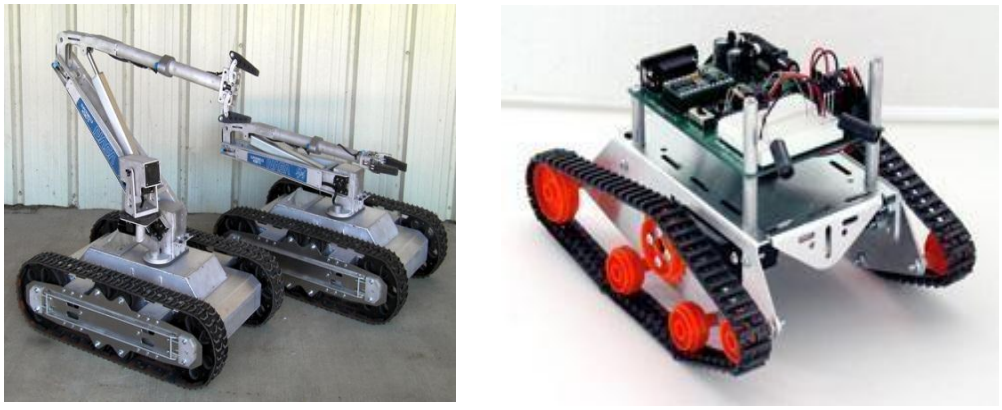


Figure 1.9 : Exemples de robots mobiles à chenilles

C. Les robots mobiles marcheurs

Les robots mobiles marcheurs sont destinés à réaliser des taches variées dont l'accès nu site est difficile ct dangereux à l'homme. Leur structure dans plusieurs degrés de liberté permet un rapprochement avec les robots manipulateurs. On distingue les robots marcheurs à deux jambes (humanoïdes), à quatre pattes (type cheval), et a six pattes (type araignée).

Figure 1.10 : Exemples des robots marcheurs

D. Les robots mobiles rampants

La reptation est une solution de locomotion pour un environnement de type «tunnel» qui conduit à réaliser des structures filiformes. Le système est compost d'un ensemble de module ayant chacun plusieurs mobilités. Ici aussi les techniques utilisées découlent des méthodes de locomotion des animaux et des insectes.

E. Autres moyens de locomotion

Les applications de ce type de robots sont très spécialisées et les architectures des robots sont en général spécifiques à l'application visée. (Figure 1.11) [6]

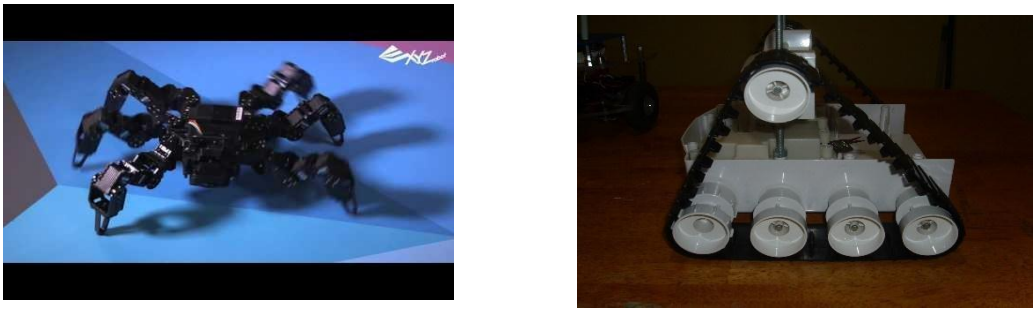


Figure 1.11 : Exemple d'un robot rampant

4.2.3 Classification selon le domaine d'application

Bien que le champ d'application des robots mobiles reste illimité, nous présentons ici quelques domaines d'application.[19]

- **Les robots industriels et de service**

Il existe des robots mobiles destinés à des applications industrielles. Celles-ci concernent principalement le transport et la distribution (dans les usines, les mines, les hôpitaux et les ateliers), le nettoyage, 1'entretien et la maintenance, la surveillance et la manutention. Quant

aux robots de service, ils sont destinés à aider des handicapés moteurs, à guider les aveugles el à piloter des voitures automatiques.[19]

- **Les robots militaires**

Les applications militaires de la robotique mobile sont nombreuses. Ce champ d'application présente l'intérêt de fournir des spécifications serrées telles que la vitesse des véhicules, leurs capacités de franchissement des obstacles et leur rapidité de réaction.[19]

- **Les robots de laboratoires**

De nombreux laboratoires travaillant dans le domaine de la robotique, disposent de plates-formes expérimentales pour valider des travaux théoriques en perception ou en planification de mouvement.[19]

4.2.4 Classification selon la motricité et l'énergie

Le déplacement des robots est réalisé par des moteurs de types :

- Électrique
- Thermique
- Hydraulique

L'énergie électrique la plus fréquemment employée offre 1'avantage d'une commande aisée. Par contre le transport et la génération présentent des difficultés.

Plusieurs méthodes sont employées :

- Par batteries qui soul :
  - Soit recharges périodiquement de manière automatique ou manuelle.
  - Soit par un échange avec d'autre lorsqu'elles sont déchargées
- Par groupe électrogène embarqué dont l'inconvénient constitue la masse élevée, l'énergie de base est alors thermique.
- Par cordon ombilical qui réduit l'autonomie du robot.

L'énergie thermique est essentiellement employée par des véhicules de forte puissance comme énergie de base pour la traction ou pour activer un compresseur hydraulique.[2]

5. Caractéristiques d'un robot

Un robot doit être choisi en fonction de l'application qu'on lui réserve. Voici quelques paramètres à prendre, éventuellement, en compte :

5.1 La charge maximale transportable

De quelques kilos à quelques tonnes, à déterminer dans les conditions les plus défavorables (en élongation maximum). [2]

5.2 Le volume de travail

Défini comme l'ensemble des points qu'on peut atteindre par l'organe terminal.

Tous les mouvements ne sont pas possibles en tout point du volume de travail. L'espace (volume) de travail (reachable workspace), également appelé espace de travail maximal, est le volume de l'espace que le robot peut atteindre via au moins une orientation. L'espace de travail dextre (dextrous- workspace) est le volume de l'espace que le robot peut atteindre avec toutes les orientations possibles organe terminal. Cet espace de travail est un sous-ensemble de l'espace de travail maximal, [2].

5.3 Le positionnement absolu

Correspondant à l'erreur entre un point souhaité (réel), défini par une position et une orientation dans l'espace cartésien et le point atteint. Il est calculé via le modèle géométrique inverse du robot. Cette erreur est due au modèle utilisé, à la quantification de la mesure de position et à la flexibilité du système mécanique .En général, l'erreur de positionnement absolu, également appelée précision, est de l'ordre de 1 mm. [2]

5.3.1 La répétabilité

Ce paramètre caractérise la capacité que le robot à retourner vers un point (position, orientation) donné. La répétabilité correspond à l'erreur maximum de positionnement sur un point prédéfini dans le cas de trajectoires répétitives.[2]

5.3.2 La vitesse de déplacement

Vitesse maximum en élongation maximum ou accélération. Il existe d'autres caractéristique comme: [6]

- La masse du robot.
- Le coût du robot.
- La maintenance, …

6. Conclusion

Nous avons présenté, dans ce chapitre la diversité des robots et leurs applications et la diversité des disciplines concernée par la robotique et en particulier le robot mobile ,qui remplacent l'homme dans les taches pénibles et dangereuses actuellement plus encore dans l'avenir ,les robots sont utiliser à jouer un rôle de plus en plus important dans notre vie mais ceci n'annule pas l'existence de certains problèmes pour assurer une bonne application de ces robots, Comme exemples de ces problèmes nous citons l'analyse de l'environnement, planification, navigation ….

Dans le chapitre suivant nous allons présenter un aperçu sur les notions de base de la logique floue.

Chapitre 2
La logique floue

1. Introduction

La logique floue est introduite de la constatation que certains problèmes rencontrés ne sont pas modélisables mathématiquement ou à l'aide des variables booléennes qui ne peuvent prendre que deux valeurs (0 ou 1). Les problèmes du monde réel doivent tenir compte d'informations imprécises, incertaines…. Prenons l'exemple suivant : peut on considérer une eau à 18°C comme étant chaude ou froide ? N'est-elle pas ni vraiment chaude, ni vraiment froide mais tout simplement tiède ? On voit apparaître dans cet exemple la difficulté d'interprétation des variables linguistiques comme chaud, froid, …

Pour répondre à ce type de question, la logique floue fait introduire la notion d'appartenance d'un objet à un ensemble non plus comme une fonction booléenne mais comme une fonction qui peut prendre toutes les valeurs entre 0 et 1.

2. Généralité sur la logique floue

2.1 Bref historique

Le concept de la logique floue fut réellement introduit en 1965 par Lotfi Zadeh, un professeur d'électronique à l'université de Berkeley (USA). Sa "Fuzzy Set Theory" n'eut pas un succès immédiat. Elle fut développée surtout en Europe et au Japon.

Voici quelques points historiques permettent de situer dans le temps le développement de la logique floue :

- 1965, Le professeur Lotfi Zadeh de l'université de Berkeley (Californie) pose les bases théoriques de la logique floue « Théorie des sous ensembles flous ».
- 1973, Lotfi Zadeh introduit la notion de variables linguistiques.
- 1974, Mamdani (Londre) réalise un controleur flou pour moteur à vapeur.
- 1985, Premiers produits industriels utilisant le principe de la logique floue (Japon).

Aujourd'hui, une vaste gamme de nouveaux produits a une étiquette «produit flou».[10]

2.2 Domaine d'application

La logique floue a déjà fait ses preuves dans les domaines d'application suivants :

- Gestion de projet.
- Analyse démographique des marchés.

- Pilotage d'un système d'autofocus d'appareil photo.
- Lecture automatique, reconnaissance des caractères.
- Traitements d'image.
- Bases de données, recherche d'information.
- …

2.3 Définition

Logique qui substitue à la logique binaire une logique fondée sur des variables pouvant prendre, outre les valeurs « vrai » ou « faux » , les valeurs intermédiaires « vrai » ou « faux » avec une certaine probabilité. (Citation : cf.supra).[11]

2.4 Les éléments de la logique floue

La logique floue est une branche des mathématiques et toute une série de notions fondamentales sont développées. Ces fonctions permettent de justifier et de démontrer certains principes de base. Dans ce qui suit on ne retiendra que les éléments indispensable à la compréhension du principe de la logique floue. Ces éléments sont [11] :

- Les variables floues.
- Les règles d'inférences.

2.4.1 Variables floues

Contrairement aux variables binaires qui sont définies par les deux états « vrai » ou « faux », les variables floues présentent toute une gradation entre la valeur « vrai » et la valeur « faux ». L'exemple qui suit permet de mieux saisir la distinction qui existe entre les variables binaires et les variables floues :

Si l'on désire classer un groupe d'individu par leur taille en définissant la catégorie des petits par une taille en dessous de 160 cm, la catégorie des moyens par une taille comprise entre 160 cm et 180 cm et la catégorie des grand par une taille supérieure à 180 cm, la logique binaire donne la représentation de la figure2.1 pour les trois variables « petit », « moyen » et « grand ».

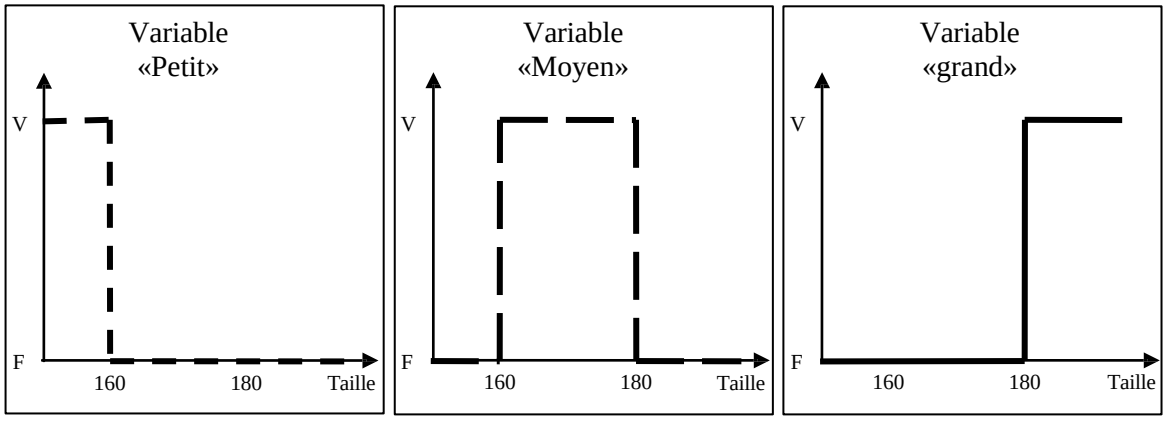


Figure 2.1 : Représentation binaire des variables

Deux remarques s'imposent au sujet de cette représentation :

- D'une part, on préfère représenter l'état de la variable à l'aide de son degré de vérité en associant la valeur 1 (degré de vérité de 100%) à la valeur « vrai » et le degré de vérité nul à la valeur « faux ».
- D'autre part, on constate que cette façon de faire est très éloignée de ce que fait l'être humain lorsqu'il résout ce genre de problème. En effet, l'homme ne fait pas naturellement une distinction franche entre « petit » et « moyen ». Par exemple, il utilise des expressions du genre « plutôt petit » pour qualifier une taille légèrement inférieure à 160 cm et « plutôt moyen » pour une taille légèrement supérieure à cette valeur.

En conclusion, la logique binaire présente l'avantage de la simplicité mais elle est assez éloignée de la logique utilisée naturellement par l'être humain. Si l'on représente le même problème à l'aide de la logique floue, les variables ne sont plus binaires mais présentent une infinité de valeurs possible entre le « vrai » et le « faux » figure2.2.

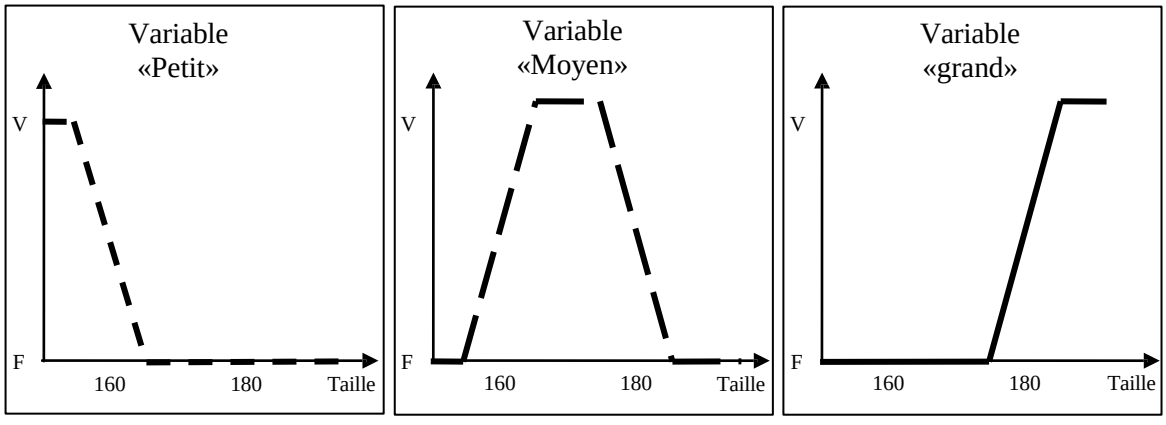


Figure 2.2 : Représentation logique des variables

On constate que cette représentation est beaucoup plus proche de la façon dont l'être humain raisonne puisqu'elle permet de faire intervenir des notions telles que « plutôt petit », « assez grand »... Cet avantage se fait, évidemment, au détriment de la simplicité de la représentation.

Les grandeurs utilisées dans n'importe quelle système doivent être converties en variables floues. Pour ce faire on définit les deux notions suivantes :

- Les *fonctions d'appartenances* qui permettent de définir le degré de vérité de la variable floue en fonction de la grandeur d'entrée.
- Les *intervalles flous* qui déterminent le nombre de variables floues

Dans l'exemple de la figure 2.2, on fait intervenir trois intervalles flous : « petit », « moyen » et « grand ». En outre chaque intervalle fait référence à une fonction d'appartenance qui permet de définir le degré de vérité de la variable floue correspondante en fonction de la taille. [11]

2.4.1.1 Fonctions d'appartenances

Il s'agit d'établir une relation entre le degré de vérité de la variable floue et la grandeur d'entrée correspondante (figure 2.2). On parle de *fuzzification.*

On peut choisir n'importe quelle forme pour les fonctions d'appartenance, en triangles, en cloche ou encore en trapèze. Cependant, en pratique, on utilise les formes trapézoïdales (figure 2.2).[11]

2.4.1.2 Intervalles flous

Ces intervalles définissent le nombre de variables floues associées à une grandeur d'entrée. Par exemple dans le cas du réglage, trois à cinq intervalles s'avèrent suffisants.

De façon générale ils sont caractérisés à l'aide de symboles tels que ceux présentés dans le tableau suivant :

Symbole	Signification
NG	Négatif Grand
NM	Négatif Moyen
EZ	Environ Zéro
PM	Positif Moyen
PG	Positif Grand

Tableau 1 : Exemple d'intervalles flous

Un cas particulier qui est la grandeur de sortie qui peut être définie à l'aide d'un certain nombre d'intervalles flous et diverses fonctions d'appartenance. Toutefois, en pratique, cette définition peut sembler assez lourde et le concepteur (l'*expert*) peut choisir d'associer une seule valeur à chaque intervalle flou. Par exemple, pour une grandeur à cinq intervalles flous, on peut définir les valeurs suivantes (tableau 2) :

Intervalle	Valeur en % du maximum
Très petit	0
Petit	25
Moyen	50
Grand	75
Très grand	100

Tableau 2 : Intervalles flous pour variables de sortie

Ce qui définit des fonctions d'appartenances en forme de raies comme illustré à la figure2.3.[12]

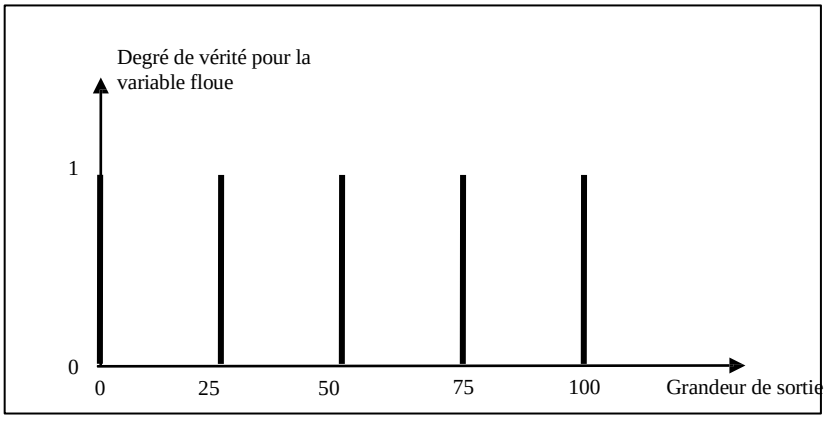


Figure 2.3 : Exemple de fonctions d'appartenance.

2.4.2 Règle d'inférence

On appelle « règles d'inférence » l'ensemble des différentes règles reliant les variables floues d'entrée d'un système aux variables floues de sortie de ce système. Ces règles se présentent sous la forme :

Si condition 1 **et/ou** condition 2 (**et/ou...**) **alors** action sur les sorties

L'exemple suivant, tiré de la vie quotidienne, permet d'illustrer ceci :

Lorsque l'on prend une douche, un des problèmes qui se présente est de régler la température de l'eau. La variable d'entrée du système homme-douche est la température de l'eau mesurée à l'aide de nos capteurs de température. Les variables de sorties sont les deux robinets eau chaude et eau froide. Dans la pratique, le réglage de la température se fait en utilisant notre expérience, expérience qui recouvre à la fois nos préférences et notre connaissance de l'installation sanitaire utilisée. Ce réglage se fait en utilisant des règles du genre :

- **Si** la température est très froide **alors** ouvrir à fond l'eau chaude.
- **Si** la température est un peu trop froide **alors** fermer un peu l'eau froide.
- **Si** la température est bonne **alors** laisser les deux robinets dans leur état.
- **Si** la température est trop chaude **alors** fermer un peu l'eau chaude et ouvrir un peu l'eau froide.
- ...

En termes d'intelligence artificielle, ces règles résument en fait « l'expérience » de « l'expert » et elles ne sont en général pas définissables de façon unique puisque chaque individu crée ses propres règles.

Les règles d'inférences font appel aux opérateurs **et**, **ou** et **non**, qui s'appliquent aux variables floues. Dans le cas de la logique binaire ces opérateurs sont définis de façon simple et univoque. Dans le cas de la logique floue, la définition de ces opérateurs n'est plus univoque et on utilise le plus souvent les relations présentées dans le tableau suivant :

Opérateur	Opération sur le degré de vérité des variables
et	minimum
	produit
ou	maximum
	valeur moyenne
non	complément à 1

Tableau 3 : Relations entre règles d'inférences

Les opérations **minimum** et **maximum** présentent l'avantage de la simplicité lors du calcul, par contre, elles privilégient l'une des deux variables. Les opérations de **produit** et **valeur moyenne** sont plus complexes à calculer mais elles produisent un résultat qui tient compte des valeurs des deux variables.[12]

3. Système flou

3.1 Principe de fonctionnement

Le principe de fonctionnement d'un système flou est simple. Celui-ci se décompose en trois étapes distinctes.[12]

3.1.1 Fuzzification

Cette première étape consiste à déterminer le degré d'appartenance de chaque variable d'entrée à chaque état. Celui-ci est déterminé à l'aide des fonctions d'appartenance définies dans le système. [12]

3.1.2 Inférence

Les degrés d'appartenance de chaque variable à chaque état permettent d'appliquer les règles floues qui on été préalablement définies. Le degré d'appartenance des variables de sortie à chaque état est ainsi obtenu.[12]

3.1.3 Défuzzification

En sortie, le système flou ne peut pas communiquer des valeurs floues qu'il peut seul exploiter. Par exemple pour un processus de contrôle l'organe de commande nécessite un signal d'entrée précis. Il lui est donc nécessaire de fournir des valeurs précises, c'est le rôle de la défuzzification, en quelque sort elle est l'étape de l'interprétation de la solution floue afin de convertir l'information floue à une grandeur physique.

Cette étape s'effectue à l'aide des fonctions d'appartenance, cela pour déterminer la valeur précise à utiliser, on peut soit conserver le maximum, soit calculer la moyenne pondérée, soit déterminer le centre de gravité des valeurs obtenues, donc il y a plusieurs façons de faire mais, en pratique, on utilise surtout les deux méthodes suivantes :

- Défuzzification par calcul du centre de gravité
- Défuzzification par calcul du maximum. [12]

3.1.4 Défuzzification par calcul du centre de gravité

Il s'agit de calculer le centre de gravité de la fonction d'appartenance de la variable de sortie.

Le calcul du centre de gravité permet bien d'obtenir une seule valeur pour la grandeur de sortie. Son calcul est cependant relativement complexe puisqu'il nécessite le calcul d'une intégrale (Figure2.4).[10]

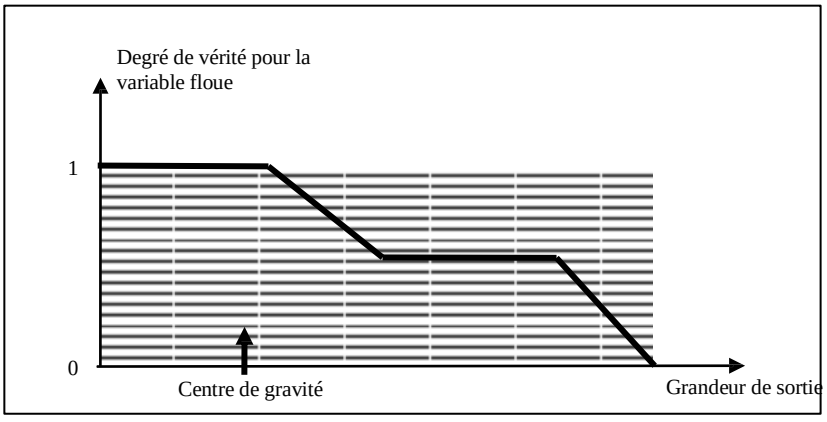


Figure 2.4 : Calcul du centre de gravité

3.1.5 Défuzzification par calcul du maximum

Il s'agit de la façon la plus simple, au point de vue du volume de calcul, pour effectuer la défuzzification. La façon de procéder diffère cependant fondamentalement du cas général exposé ci-dessus. Tout d'abord, la grandeur de sortie doit être normalisée (en pour-cent par exemple) et la définition des intervalles flous doit se résumer à une valeur : par exemple « petit » correspond à 0 et « moyen » à 0,5 (fonctions d'appartenance en forme de raies). L'opération de défuzzification consiste à prendre d'abord le minimum entre la

valeur produite par la règle concernée et la valeur de la variable floue de sortie. La valeur de sortie est définie par la valeur maximale des variables floues de sortie.

L'exemple suivant permet d'en illustrer le principe :

Soit un système avec trois règles :

La règle 1 : donne une sortie de type « petit » d'un degré de vérité de 0,8

La règle 2 : sortie de type « moyen » d'un degré de vérité de 0,3

La règle 3 : sortie de type « grand » d'un degré de vérité de 0,1

La valeur normalisée de l'intervalle « petit » vaut 0

La valeur normalisée de l'intervalle « moyen » vaut 0,5

La valeur normalisée de l'intervalle « grand » vaut 1

La règle 1 donne une valeur de sortie de 0 (minimum entre 0,8 et 0)

La règle 2 donne une valeur de sortie de 0,3 (minimum entre 0,3 et 0,5)

La règle 3 donne une valeur de sortie de 0,1 (minimum entre 0,1 et 1)

La grandeur de sortie est déterminée par le maximum des valeurs obtenues et vaut 0,3 ce qui correspond à une valeur « plutôt petite ».

On constate que cette méthode est simple à appliquer mais, étant basée sur l'opérateur **maximum**, elle privilégie une seule règle parmi celles présentes.[10]

3.2 Logique floue et système expert

La logique floue et les systèmes experts sont fortement liés. En effet la construction d'un modèle en logique floue passe toujours par la transcription d'une expertise humaine sous la forme de règles floues. Celles-ci sont ensuite utilisées par le moteur d'inférence d'un système expert.

Comme ces règles sont exprimées en langage naturel à l'aide de termes vagues, elles sont en général facilement formulable par l'expert lui-même, ce qui n'est pas du tout le cas pour les systèmes experts classique qui nécessitent des valeurs précises pour les variables manipulées.

4. Exemple d'application

4.1 Description du problème

On souhaite commander l'installation de chauffage d'un immeuble à l'aide d'un contrôleur flou. On dispose de deux sondes de température : l'une à l'extérieur de

l'immeuble et l'autre à l'intérieur. Sur la base de ces deux mesures et en faisant appel aux règles d'inférence, le contrôleur flou doit régler la puissance de l'installation de chauffage.[10]

4.2 Fuzzification de la température externe

On choisit deux intervalles flous et des fonctions d'appartenance de type trapézoïdales en définissant le « froid » comme correspondant à une température inférieure à 5 °C et le « chaud » comme étant une température supérieure à 20 °C (Figure2.5).[10]

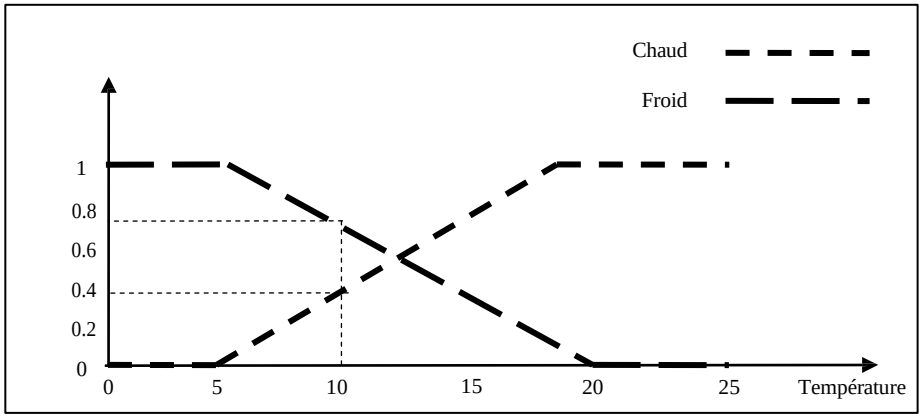


Figure 2.5 : Fuzzification de la température externe

4.3 Fuzzification de la température interne

On choisit trois intervalles flous et des fonctions d'appartenance de type trapézoïdales en définissant le « froid » comme correspondant à une température inférieure à 15 °C, le « bon » comme étant une température comprise entre 19 °C et 21 °C et le « chaud » comme étant une température supérieure à 25 °C (Figure2.6).[10]

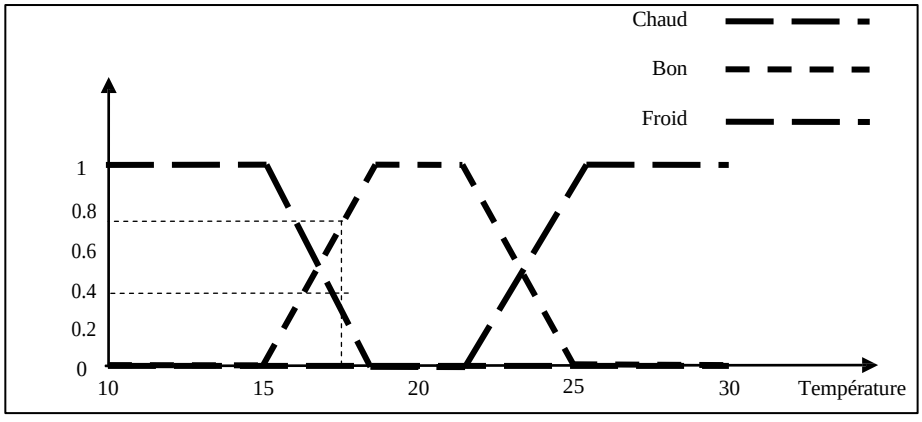


Figure 2.6 : Fuzzification de la température interne

4.4 Fuzzification de la puissance

On choisi quatre intervalles flous pour définir la puissance de l'installation avec des fonctions d'appartenance en forme de raies. On définit les valeurs suivantes :

Puissance	Valeurs en %
Nulle	0
Faible	33
Moyenne	67
maximale	100

Ce qui définit les fonctions d'appartenance illustrées à la figure 2.7 :

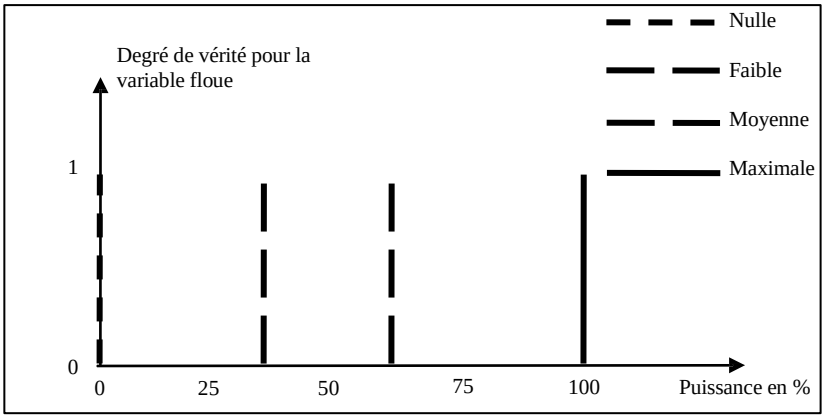


Figure 2.7 : Fuzzification de la puissance

4.5 Règles d'inférences

L'expérience acquise sur l'installation de chauffage a permis de définir les six règles suivantes :

- Si la température extérieure est « froide » et la température intérieure est « froide » alors mettre la puissance au « maximum ».
- Si la température extérieure est « froide » et la température intérieure est « bonne » alors mettre une puissance « moyenne ».
- Si la température extérieure est « froide » et la température intérieure est « chaude » alors mettre une puissance « faible ».

- Si la température extérieure est « chaude » et la température intérieure est « froide » alors mettre une puissance « moyenne ».
- Si la température extérieure est « chaude » et la température intérieure est « bonne » alors mettre une puissance « faible ».
- Si la température extérieure est « chaude » et la température intérieure est « chaude » alors mettre une puissance « nulle ».[10]

4.6 Choix des opérateurs

- L'opérateur **et** est réalisé par le calcul du **minimum.**
- L'opérateur **ou** est réalisé par le calcul du **maximum.**

4.7 Choix du type de défuzzification

La défuzzification se fait par le calcul du centre de gravité.

4.8 Exemple de calcul

Hypothèse : la température extérieure est de 10 °C et la température intérieure est de 22 °C.

- Les variables floues de la température extérieure sont donc :

« Froide » avec un degré de vérité de 0,67

« Chaude » avec un degré de vérité de 0,33

- Les variables floues de la température intérieure sont donc :

« Froide » avec un degré de vérité de 0

« Bonne » avec un degré de vérité de 0,75

« Chaude » avec un degré de vérité de 0,25

Les règles d'inférences donnent les valeurs suivantes pour les variables floues de sortie :

- « maximum » avec un degré de vérité de 0 (minimum de 0,67 et 0)
- « moyenne » avec un degré de vérité de 0,67 (mini. de 0,67 et 0,75)
- « faible » avec un degré de vérité de 0,25 (mini. de 0,67 et 0,25)
- « moyenne » avec un degré de vérité de 0 (mini. de 0,33 et 0)
- « faible » avec un degré de vérité de 0,33 (mini. de 0,33 et 0,75)
- « nulle » avec un degré de vérité de 0,25 (mini. de 0,33 et 0,25)

L'opérateur **ou** appliqué sur les règles qui donnent les mêmes variables floues donne :

- « maximum » avec un degré de vérité de 0
- « moyenne » avec un degré de vérité de 0,67 (maxi. de 0,67 et 0)
- « faible » avec un degré de vérité de 0,33 (maxi. de 0,25 et 0,33)
- « nulle » avec un degré de vérité de 0,67

Le calcul du centre de gravité se calcule à l'aide de :

$$P = \frac{\sum_{i=1}^{4} \mu i * Pi}{\sum_{i=1}^{4} \mu i} = \frac{0*100 + 0{,}67*67 + 0{,}33*33 + 0{,}25*0}{0 + 0{,}67 + 0{,}33 + 0{,}25}$$

Pi : Valeur de la variable floue

μi : Degré de vérité de la variable floue.

Le contrôleur flou impose donc une puissance de 44,6% sur l'installation de chauffage.[10]

5. Conclusion

La logique floue à une très grande flexibilité de représenter les connaissances de l'expert ce qui implique une compréhension plus rapide par l'utilisateur qui manipule des technologies utilisant la logique floue ; malgré que ses applications ne sont pas fondamentalement plus performantes mais elle sont plutôt facile à réaliser et à utiliser.

Cependant la manipulation des règles non précisent peut engendrer des erreurs non négligables ce qui implique la fourniture d'un effort considérable pour la mise en place d'un tel système.

Chapitre 3
Le langage Java

1. Introduction :

Le langage Java a été conçu pour permettre l'exécution du même code sur diverses plates-formes. En particulier, mais pas uniquement, sur le web. Il y a plusieurs types de programmes Java, dont en particulier les applets Java, qui sont intégrés à des pages web et doivent respecter des règles très strictes pour ne pas risquer de causer des dégâts sur les machines d'innocents surfers, et les applications Java, qui fonctionnent comme d'autres programmes, en local est compilé ,mais les fichiers résultants de la compilation nécessitent encore une interprétation différente suivant chaque plate-forme : cette opération est réalisée par la JVM (Java Virtual Machine)

2. Objectifs :

Le langage Java a été développé afin de pouvoir générer des applications qui soient indépendantes des machines et de leur système d'exploitation. Une des autres caractéristiques du langage est de pouvoir écrire des applications structurellement distribuées sur des réseaux. Il appartient, par ailleurs, à la famille des langages objets purs ou rien ne peut exister en dehors des classes

3. Historique :

La naissance de Java à 1991. À cette époque, des ingénieurs de chez SUN ont cherché à concevoir un langage applicable à de petits appareils électriques (on parle de code embarqué). Pour ce faire, ils se sont fondés sur une syntaxe très proche de celle de C++, en reprenant le concept de machine virtuelle déjà exploité auparavant par le Pascal UCSD. L'idée consistait à traduire d'abord un programme source, non pas directement en langage machine, mais dans un pseudo langage universel, disposant des fonctionnalités communes à toutes les machines. Ce code intermédiaire, dont on dit qu'il est formé de byte codes, se trouve ainsi compact et portable sur n'importe quelle machine ; il suffit simplement que cette dernière dispose d'un programme approprié (on parle alors de machine virtuelle) permettant de l'interpréter dans le langage de la machine concernée. [13]

Après 18 mois d'isolement total, les 13 personnes faisant partie du projet avaient fabriqué une télécommande digitale, munie d'un écran tactile graphique et animé. Cette seule télécommande était capable de contrôler tout un équipement audio et vidéo de

salon. On peut d'ailleurs regretter qu'elle n'ait jamais été commercialisée, et que les constructeurs préfèrent aujourd'hui saturer leurs appareils de multiples boutons aux fonctionnements obscurs... . [14]

La raison pour laquelle cette télécommande était unique est qu'elle était programmée dans un nouveau langage, indépendant du processeur sur lequel il s'exécutait, conçu par James Gosling, un des membres du Green Project. James Gosling aimait bien le chêne qu'il voyait de la fenêtre de son bureau, et a appelé ce langage Oak. Le projet gagna alors en importance, des contacts furent pris avec des câblo-opérateurs américains, et il changea de nom pour s'appeler First Person. Il ne rencontra pas de succès commercial, probablement trop en avance dans un milieu industriel qui était encore à la recherche de moyens de gagner de l'argent. [14]

L'arrivée du protocole http et du navigateur Mosaic fut un événement déterminant pour ce projet, en 1993. L'équipe comprit que l'Internet allait être le réseau idéal pour positionner leur produit. Après quelques travaux supplémentaires

En 1995, OAK devient Java et se trouve popularise rapidement pour ses possibilités de développement lies au Web. Les années qui suivent font connaıtre à Java une popularité en tant que langage généraliste qui dépasse les prévisions de SUN. Les grands industriels du développement l'adopte rapidement en l'espace de quelques années. Pour les années à venir, les industriels projettent la création de puces électroniques dédiées à Java ... un juste retour vers les préoccupations initiales des concepteurs originaux de OAK/Java. . [15]

4. **Une machine virtuelle :**

Le langage Java est un langage compile et interprète. Cette définition contradictoire s'explique par le fait que le code source est transformé dans un byte-code universel exécutable par une machine virtuelle. Cette machine virtuelle peut être installée à partir d'une distribution du langage comme le Java Développent Kit (JDK) de SUN. Elle peut être également intégrée sous une forme compactée dans un navigateur afin qu'il puisse exécuter des applets Java. Ces caractéristiques confèrent au langage des propriétés de portabilité sans précédents, mais ont aussi un cout correspondant au fonctionnement de la machine virtuelle qui réduit les performances d'exécution du programme. Afin de résoudre ce problème, un effort important est

consenti pour développer des compilateurs à la volée en code machine qui fonctionnent lors de l'exécution du programme

5. Caractéristiques :

Le langage Java possède une syntaxe inspirée du C++. Il se veut plus propre en terme de développement objet, ne permettant pas de construction en dehors des classes. Il se veut aussi plus simple en affranchissant le programmeur de toute la gestion dynamique des objets construits, grâce au fonctionnement d'un ramasse-miettes (Garbage Collector) dont la fonction est d'identifier et d'éliminer tous les objets qui ne sont plus références.

Java propose des exécutions parallèles de programmes grâce à une utilisation qui se veut plus facile des *threads* (ou processus legers) par rapport au langage C. Il possède également des aspects lies a la distribution grâce à ses possibilités d'intégration dans des documents Web distribues par les applets, mais également avec la bibliothèque RMI (Remote Methods Invocation) qui propose de la programmation d'objets repartis. L'évolution de cette bibliothèque la fait converger progressivement vers CORBA, le standard dans le domaine des objets repartis. Les bibliothèques CORBA sont aujourd'hui intégrées dans les distributions du JDK 1.2 ou supérieures de SUN. [15]

5.1 Types de données :

5.1.1 Types primitifs :

Comme tout langage de programmation évolué le Java possède un certain nombre de types de données de base :

- Le type `booléen` qui prend une des 2 valeurs `true` ou `false` sur 1 octet.
- Le type `char` qui correspond à un caractère sur 2 octets.
- Les types `byte, short, int` et `long` qui sont 4 types d'entiers stockes respectivement sur 1, 2, 4 et 8 octets.
- Les types `float` et `double` qui sont 2 types de flottants stockes respectivement sur 4 et 8 octets.

Les déclarations de telles variables peuvent se faire n'importe où dans le code mais avant leur utilisation, comme en C++.

On utilise les opérateurs usuels sur les variables de ces types avec la syntaxe du C.

5.1.2 Tableaux

Un tableau va permettre de stocker un certain nombre d'éléments de même type dans une structure de données qui dispose d'un index pour y accéder .Un tableau en Java est un objet à part entière.

Par exemple, un tableau monodimensionnel de flottants de type double sera déclaré de la manière suivante :

```
Double monTableau [ ] ;
```

Ou encore :

```
Double [ ] monTableau ;
```

Comme nous l'avons expliqué précédemment, cette déclaration a permis d'affecter une référence au nom du tableau une référence pour le construire effectivement le tableau c'est à dire disposer de plusieurs emplacement mémoire.

Un tableau a toujours une taille fixe qui doit être précisé avant l'affectation de valeurs à ses indices, de la manière suivante :

```
Int [ ] mon_tableau=new int [ ] ;  [18]
```

5.1.3 Chaines de caractères :

Les chaines de caractères ne sont pas considérées en Java comme un type primitif ou comme un tableau. On utilise une classe particulière, String, fournie dans la package Java. Lang.

Les variables de type String ont les caractéristiques suivantes :

Leur valeur ne peut pas être modifiée

On peut utiliser l'opérateur + pour concaténer deux chaines de caractères :

String S1= ''hello'' ;

String S2=''world'' ;

String S3=s1+'' ''+s2 ;

5.2 Structures de contrôle :

Ce sont essentiellement les mêmes constructions qu'en C, à savoir L'affectation qui se fait par l'operateur = et qui consiste à recopier la valeur de la variable primaire à droite du symbole dans la variable située à gauche. [15]

5.2.1 Les structures conditionnelles :

If (cond) instruction1 ; [else instruction2 ;]

-switch (selecteur) {

Case c1 : instructions1 ;

...

case cn : instructions N ;

default : instructions ;

}

5.2.2 Les structures répétitives ou boucles :

- for (initialisation ; condition ; instruction De Suite)

- while (condition) instruction ;

- do instruction ; while (condition).

6. Classes et objet en Java :

La programmation orientée objet est basée sur ce principe .Dans la réalité, des objets en constituent d'autres plus grands. Ce procédées de combinaisons n'est toutefois qu'un aspect très général de la POO la programmation orientée objets présente plusieurs autres concepts et caractéristiques, qui facilitent et assouplissent la création et l'utilisation des objets, Les classes en sont les éléments les plus importants.

L'écriture d'un programme dans un langage POO ne définit pas des objets réels, mais des classes d'objets, une classe étant un modèle applicable à de multiples objets ayant des caractéristiques similaires. [16]

6.1 Définition d'une classe :

Une classe permet de définir un type d'objets associant des données et des opérations sur celles-ci appelées *méthodes* Les données et les *méthodes* sont appelés *composants* de la classe. L'exemple de base décrit dans un paragraphe suivant permettra de définir une classe Vecteur représentant des vecteurs au sens mathématique. A chaque vecteur, seront rattaches :

– une structure de données - un tableau - pour le stockage de ses coefficients ;
– des traitements comme les suivants,
– son addition avec un autre vecteur,
– son produit scalaire avec un autre vecteur,
– et aussi son procédé de création. [15]

Par exemple, une classe Arbre peut décrire les caractéristiques de tous les arbres (possèdent des feuilles et des racines, poussent, fabriquent de la chlorophylle) Elle sert alors de modèle abstrait pour le concept d'arbre .Pour la sélectionner ou l'abattre, il faut avoir une instance avec l'arbre, il est possible de créer plusieurs instances d'une classe d'arbres déterminée. Chacune d'elles éventuellement des caractéristiques différentes (court, haut, perd ces feuilles en automne) même si elles se ressemblent et sont identifiable en tant qu'arbre [16]

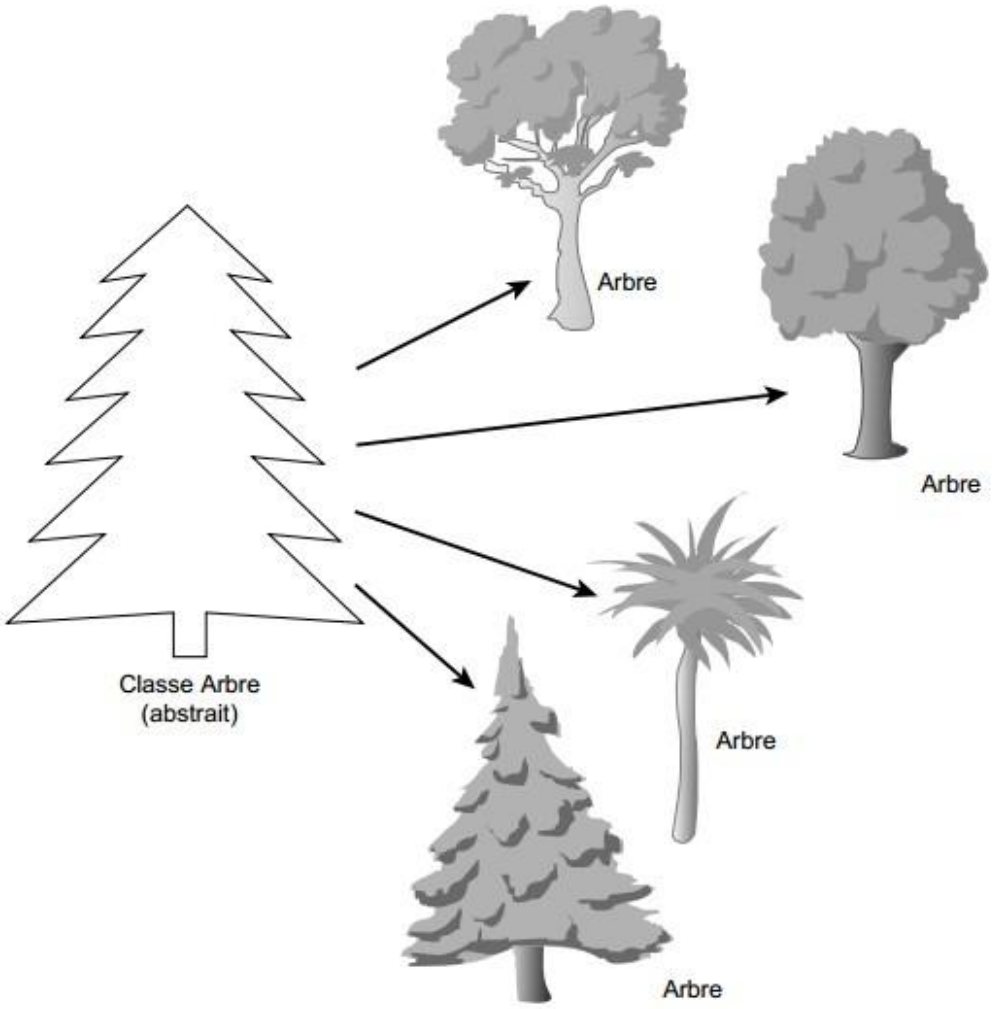


Figure 3.1 : classe d'Arbre et plusieurs instances d'Arbres

Une instance est l'autre nom de l'objet réel, si les classes sont une représentation abstraite d'un objet, une instance en est une représentation concrète

Il n y a donc aucune différence entre une instance de classe et un objet .Le terme objet est plus générale mais ils sont tous deux une représentation concrète d'une classe, les termes d'instance et d objet s utilisent indifféremment en langage de la POO .Une instance Arbre ou objet d'Arbre représente la même chose. [16]

6.2 Attributs et comportement des classes :

Chaque classe écrite en Java se compose généralement de deux parties : les attributs et le comportement. Dans cette section, une classe théorique simple appelée Motorcycle permet de les étudier. Ensuite le code Java met en œuvre la représentation d'une moto.

6.2.1 Attributs :

Les attributs sont les caractéristiques individuelles qui différencient un objet d'un autre. Ils en définissent l'apparence, l'état et les autres qualités. Soit une classe théorique appelée Motorcycle avec les attributs suivants :

-couleur : rouge, vert gris, brun

-style : patrouille, sport, standard

-marque : Honda, BMW,

Les attributs d'un objet peuvent aussi inclure des informations sur son état, telles que l'état du moteur(en marche ou à l'arrêt), ou la vitesse sélectionnée.

Les attributs sont définis dans les classes par des variables. Les types et les noms de ces variables sont définis dans la classe .chaque objet d'une classe pouvant avoir des valeurs différentes pour ces variables, on utilise le terme variable d'instance pour désigner la variable d'un objet [16]

Les variables d'instance peuvent être initialisées à la création d'un objet et demeurer constantes pendant toute la vie de l'objet, ou être changées au cour du déroulement du programme. La modification des valeurs des variables entraine celle des attributs de l'objet

Aux variables d'instance viennent s'ajouter des variables de classe. Ces dernières s'appliquent à la classe elle-même et à tous ses objets. Les valeurs des variables d'instance sont stockées dans l'instance. Les valeurs des variables de classe sont stockées dans la classe elle-même.

6.2.2 Comportement :

Le comportement d'une classe, c'est la manière dont opère une instance de cette classe, par exemple, la façon dont elle réagit à une demande d'une autre classe ou d'un autre objet, ou ce qu'elle fait lorsque son état interne se modifie. Le comportement d'un objet, c'est sa façon de faire quelque chose ou l'obtenir. A titre d'exemple, voici quelque comportement de la classe Motorcycle :

-Démarrer le moteur

-Arrêter le moteur

-Accélérer

-Changer de vitesse

La définition du comportement d'un objet passe par la création de méthodes, des ensembles d'instructions Java exécutant certaines tâches. Elles ressemblent aux fonctions des autres langages, mais elles sont définies dans une classe et accessibles seulement dans cette classe A l'inverse de C++, Java ne possède pas de fonction définies in dehors des classes. [16]

6.3 Association des classes :

Deux classes sont en association lorsque certain de leurs objets sont besoin s'envoyer des messages pour réaliser un traitement en Java, une association est représenté par l'ajout une référence d'une classe comme attribut dans l'autre.

```
public class Vehicule {
  public float age ;
  protected float taille ;
  private int poids ;
  private boolean moteur ;

  private Personne proprietaire ;

  public Vehicule() {
    age = 1.0F ;
    taille = 1.0F ;
    poids = 1 ;
    moteur = false ;
  }

  public int getWeight() {
    return(poids) ;
  }
}
```

Figure 3.2 : Exemple d'association des classes

7. Héritage, interfaces et packages:

L'héritage, les interfaces et les packages sont des mécanismes d'organisation des classes et de leurs comportements. Les bibliothèques de classe Java utilisent tous ces concepts.

7.1 Héritage :

L'héritage est essentiel pour la programmation orientée objet et influence directement la conception et l'écriture des classes Java. L'héritage est un mécanisme puissant qui facilite l'écriture d'une classe. En effet, il suffit de spécifier les différences de cette classe par rapport à une autre, l'héritage donne un accès automatique à toute l'information contenue dans cette autre classe.

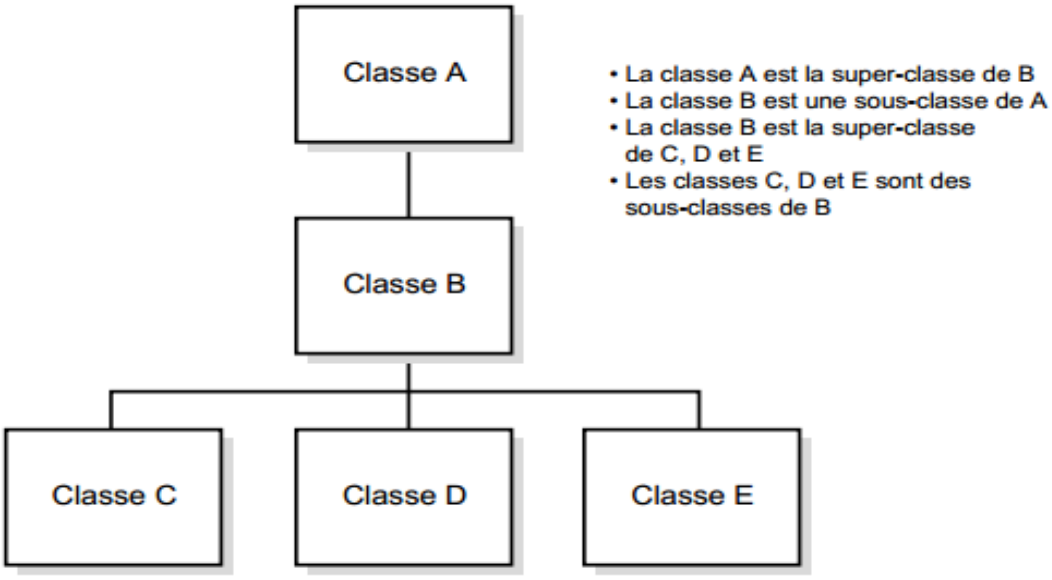


Figure 3.3 : Une hiérarchie de classe

Chaque classe possède une super classe (la classe située au-dessus dans la hiérarchie) et éventuellement une ou plusieurs sous-classes(les classes situées au-dessus dans la hiérarchie).Les classes héritent de celles qui sont situées au-dessus d'elles dans la hiérarchie.

Les sous-classes héritent de toutes les méthodes et variables de leurs super-classes. Ainsi, si une super-classe définit un comportement utile à une classe, il n'est pas nécessaire de redéfinir cette dernière ou de copier ce code à partir d'une autre classe. En effet, la classe récupère automatiquement le comportement à partir de sa super-classe, et ainsi de suite jusqu'au sommet de la hiérarchie. Une classe devient alors une combinaison de toutes les caractéristiques des classes dont elle hérite. [16]

L'héritage est un concept de la programmation orientée, selon lequel toutes les classes font partie d'une hiérarchie stricte. Chaque classe possède des super-classes, et le nombre quelconque de sous-classe. Les sous classes héritent des attributs de leurs super-classes [16]

La classe objet se trouve au sommet de la hiérarchie des classes Java. C'est la classe la plus générale de la hiérarchie. Elle définit le comportement spécifique à tous les objets

dans Java. Chaque classe située à un niveau plus bas dans la hiérarchie ajoute des informations supplémentaire .Aussi plus elles sont basses dans la hiérarchie, plus leur usage est spécifique Ainsi, la hiérarchie de classe est une ensemble de concepts Ils sont très abstraits au sommet mais deviennent concrets quand on descend dans la chaine des super-classes. [16]

7.1.1 Les difficultés inhérentes à l'utilisation de l'héritage :

Il est souvent beaucoup plus difficile de déterminer la bonne hiérarchie. La règle est simple comme nous l'exprimions ci-dessus, une relation d'héritage doit pouvoir se traduire par "la classe dérivée est une forme spécialisée de sa classe de base"

7.1.2 Hiérarchie trop lourdes :

Il faut faire attention à ne pas trop alourdir la hiérarchie en dérivant à tour de bras. Considérons par exemple une classe Animal de laquelle on dérive deux classes Chien et Chat. Jusqu'ici rien à dire, on peut en effet comprendre que les différences significatives de comportement des deux races d'animaux justifient l'existence de deux classes différentes. En revanche, il serait maladroit de dériver de Chien des classes telles que Chien Jaune ou Chien Noir sous prétexte que la couleur du pelage est différente. En effet, je ne pense pas que la couleur d'un chien soit discriminante quand à son comportement : il s'agit de toute évidence d'une simple différence de valeur d'un attribut correspondant à la couleur [18]

En effet, si la classe Base dispose de deux classes filles, la classe Intermédiaire1 ne fait qu'alourdir inutilement la hiérarchie. En effet, elle est abstraite (comme en témoigne son nom en italique) donc elle ne peut pas avoir d'instances et elle n'est dérivée qu'une seule fois. On pourrait donc la supprimer pour intégrer ces caractéristiques dans la classe Intermédiaire2. Cet exemple permet en outre d'ajouter une touche de vocabulaire : on appelle Feuille une classe qui n'a pas de dérivées [6]

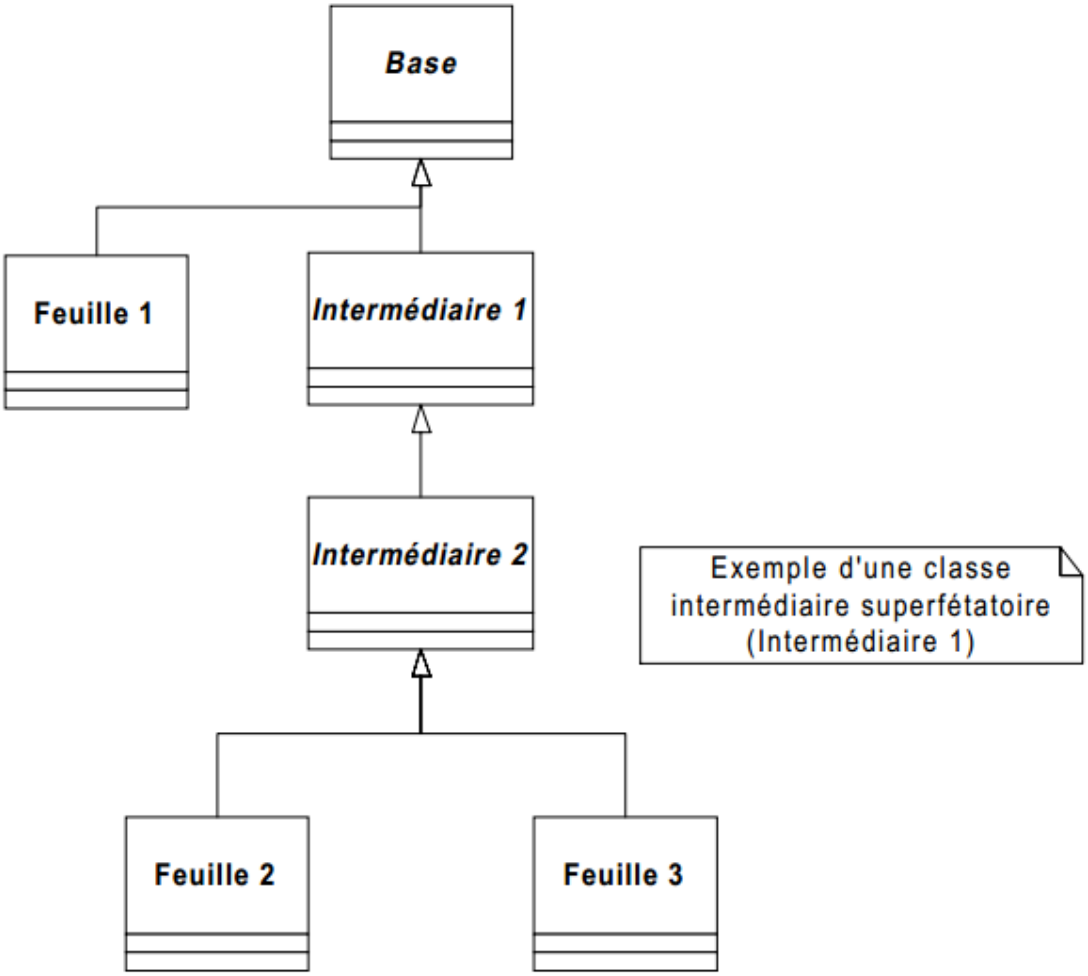


Figure 3.4 : Hiérarchie contenant une classe inutile

7.1.3 L'héritage de construction :

L'héritage de construction est un autre exemple de mauvaise utilisation de l'héritage à bannir absolument. En effet il consiste à dériver une classe en lui ajoutant de nouveaux attributs de manière à modifier totalement le concept utilisé. Par exemple, cela revient à considérer qu'un rectangle est une ligne auquel on a rajouté une deuxième dimension. En outre, dans notre dernier cas, il est facile de voir que l'héritage est impropre, car la phrase "Un rectangle est une version spécialisée d'une ligne" n'a pas de sens.

7.1.4 Les incohérences conceptuelles :

Parfois l'héritage peut conduire à des aberrations conceptuelles. Considérons par exemple une classe oiseau pourvue de la méthode Voler. On peut, par exemple dériver la

classe Pingouin qui sera elle-même pourvue de la méthode Voler, soit par héritage, soit par redéfinition alors que chacun sait très bien que les pingouins ne volent pas même si, dans ce cas, on peut dire sans crainte "Un pingouin est un oiseau spécialisé". Aussi, certains langages proposent de l'héritage sélectif : le programmeur est invité à spécifier quels attributs et quelles méthodes seront héritées. Hors supprimer la méthode Voler dans la hiérarchie d'Oiseau n'est pas sans poser de problème. En effet, cela nuit totalement au principe de polymorphisme qui veut qu'une méthode présente dans la classe de base puisse toujours se retrouver dans les classes dérivées

7.1.5 L'héritage multiple :

L'héritage multiple est une extension au modèle d'héritage simple où l'on autorise une classe à posséder plusieurs classes mères afin de modéliser une généralisation multiple.

Par exemple, supposons que l'on souhaite ajouter la classe Hovercraft à notre modèle de parc de véhicules. Un Hovercraft est à la fois un bateau et un véhicule terrestre. Aussi, il est possible de le modéliser de la manière suivante

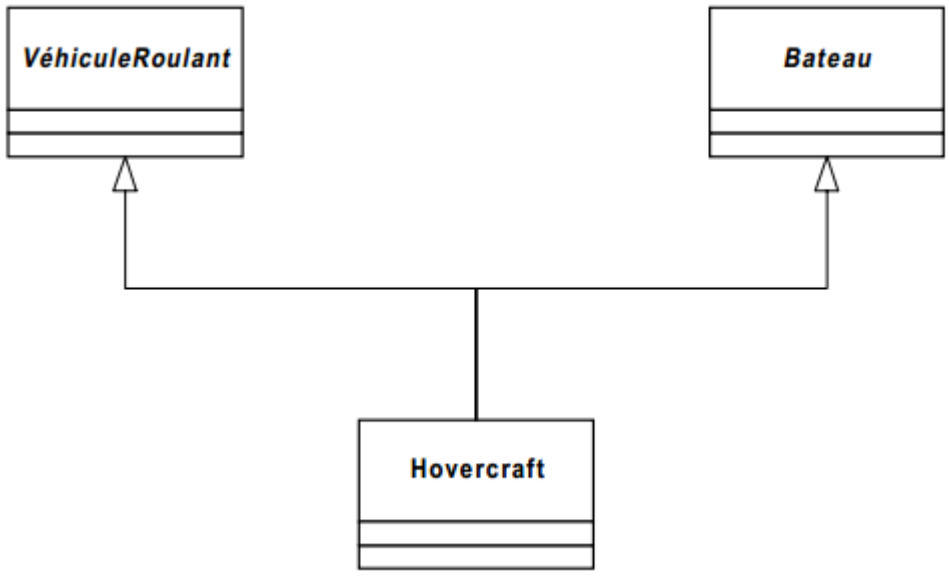


Figure 3.5 : d'utilisation de l'héritage multiple

L'utilisation de l'héritage multiple n'est pas sans poser certains problèmes. Par exemple, si les deux classes de base ont des attributs ou des méthodes de même nom, on

se trouve confrontés à des collisions de nommage qu'il convient de résoudre. Certains langages préfixent le nom de l'attribut par sa classe d'origine

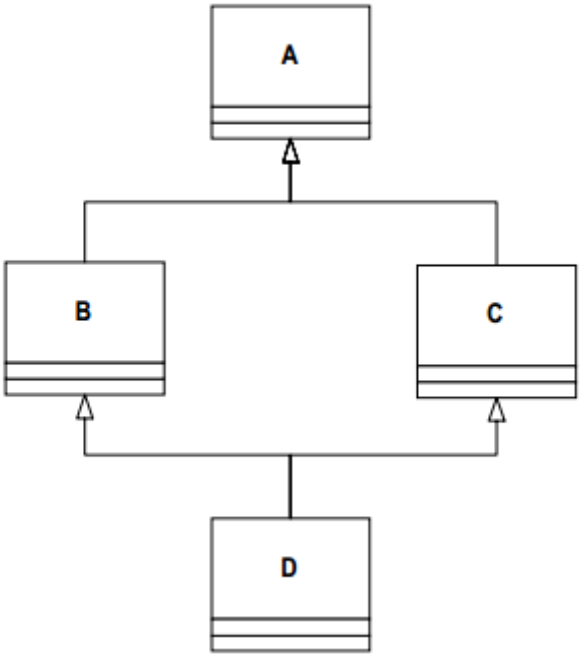


Figure 3.6 : héritage à répétition

Autre problème (illustré par la Figure) : l'héritage multiple se transforme très souvent en héritage à répétition. Ici la classe D hérite de deux copies de la classe A, une à travers la classe B, l'autre à travers la classe C

7.2 Les interfaces :

Les problèmes liés à l'héritage multiple ont conduit certains concepteurs de langage à le bannir des fonctionnalités proposées. On a même vu l'apparition d'autres mécanismes, tels que les *interfaces*. Une interface est semblable à une classe sans attribut (mais pouvant contenir des constantes) dont toutes les méthodes sont abstraites.

En plus de ses caractéristiques d'héritage, une classe implémente une interface si elle propose une implémentation pour chacune des méthodes décrites en interface. Ce mécanisme est particulièrement puissant car il crée des relations fortes entre différentes classes implémentant les mêmes interfaces sans que ces dernières aient une relation de parenté. En particulier, les méthodes décrites dans les interfaces sont, par définition,

polymorphes puisqu'elles sont implémentées de façon indépendante dans chaque classe implémentant une même interface. [18]

En outre, chaque classe peut implémenter autant d'interfaces qu'elle le désire. Ce mécanisme, a été repris par les langages Objective C et Java en particulier

7.3 Déclaration, création et destruction d'objets :

Pour manipuler un objet (par exemple, un vecteur particulier de la classe Vecteur), on doit tout d'abord déclarer une *référence* sur la classe correspondant au type d'objet. Cette opération permet de réserver en mémoire une adresse qui référencera l'objet. Par exemple, pour déclarer un objet x de type Vecteur on écrira :

```
Vecteur x;
```

Pour que cet objet soit réellement construit, c'est à dire que la référence désigne un emplacement a partir duquel on pourra accéder aux caractéristiques de l'objet, on devra appeler l'opération `new` qui alloue cet emplacement en mémoire et le renvoie à la référence de l'objet. On dispose alors d'une nouvelle *instance* de l'objet. Par exemple, pour la construction effective de notre vecteur x, on écrira :

```
X = new Vecteur ();
```

Cette instruction fait appel à un procède de construction d'objets, appelé *constructeur*, qui est défini par défaut, mais qui peut aussi être redéfini comme opération dans la classe Vecteur. Si cette instruction de construction n'est pas effectuée, la référence associée à x est initialisée par défaut a NULL, qui est une adresse fictive ne pointant vers aucun emplacement mémoire réel [15]

Par ailleurs, si l'on a besoin de référencer l'objet courant dans la définition de la classe, cette référence se fait par le mot réserve This qui vaut donc l'adresse de l'objet courant dans une *instance* de la classe. [15]

La destruction des objets est pris en charge par le *Garbage Collector*, appelé encore *ramasse-miettes* dans sa dénomination francisee. Cet outil logiciel fonctionne en même temps que le programme, il recherche, identifie et supprime de la mémoire les objets qui ne sont plus référençables. [15]

8. **Présentation de la notion objet :**

Un *objet* est une entité cohérente rassemblant des données et du code travaillant sur ses données. Une *classe* peut être considérée comme un moule à partir duquel on peut créer des objets. La notion de *classe* peut alors être considérée comme une expression de la notion de classe d'équivalence chère aux mathématiciens. En fait, on considère plus souvent que les classes sont les *descriptions* des objets (on dit que les classes sont le méta donné des objets), lesquels sont des *instances* de leur classe. [18]

Considérons par exemple la modélisation d'un véhicule telle que présentée par la figure suivante :

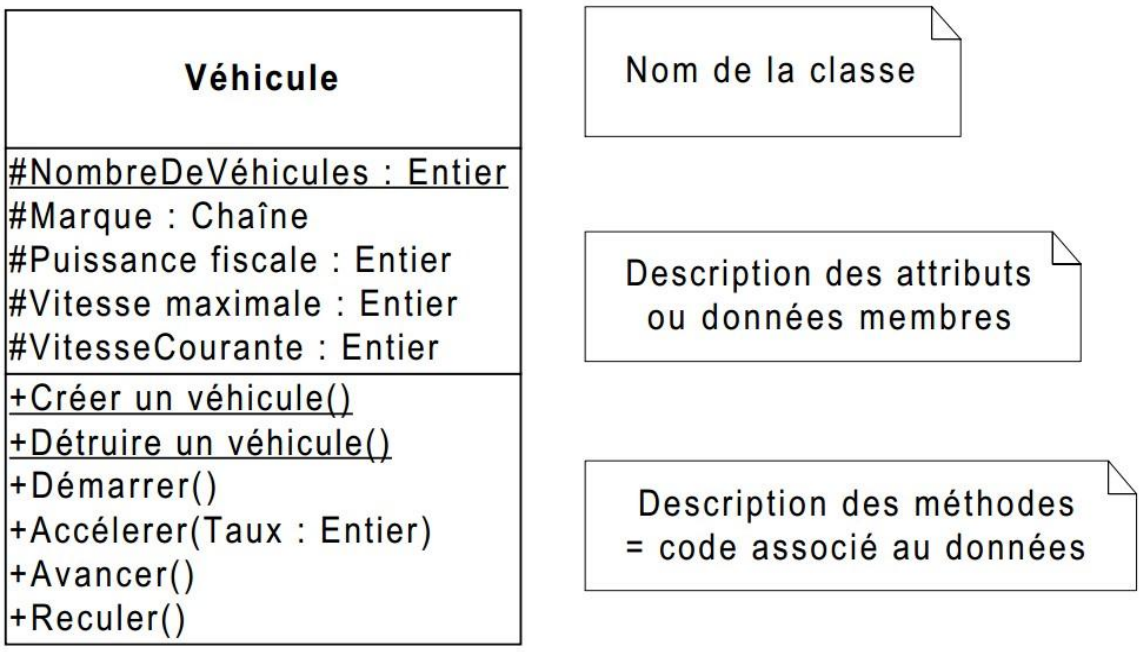


Figure 3.7 : Représentation de la classe véhicule

Dans ce modèle, un véhicule est représenté par une chaîne de caractères (sa *marque*) et trois entiers : la *puissance fiscale*, la *vitesse maximale* et la *vitesse courante*. Toutes ces données sont représentatives d'un véhicule particulier, autrement dit, chaque objet véhicule aura sa propre copie de ses données : on parle alors d'attribut *d'instance*. L'opération *d'instanciation* qui permet de créer un objet à partir d'une classe consiste précisément à fournir des valeurs particulières pour chacun des attributs d'instance [18]

En revanche, considérons l'attribut Nombre de véhicules chargé de compter le nombre de véhicules présents à un moment donné dans la classe. Il est incrémenté par l'opération Créer un véhicule et décrémenté par l'opération Détruire un véhicule. C'est un exemple typique d'attribut partagé par l'ensemble des objets d'une même classe. Il est donc inutile et même dangereux (penser aux opérations de mise à jour) que chaque objet possède sa copie propre de cet attribut, il vaut mieux qu'ils partagent une copie unique située au niveau de la classe. On parle donc d'attribut de classe [18]

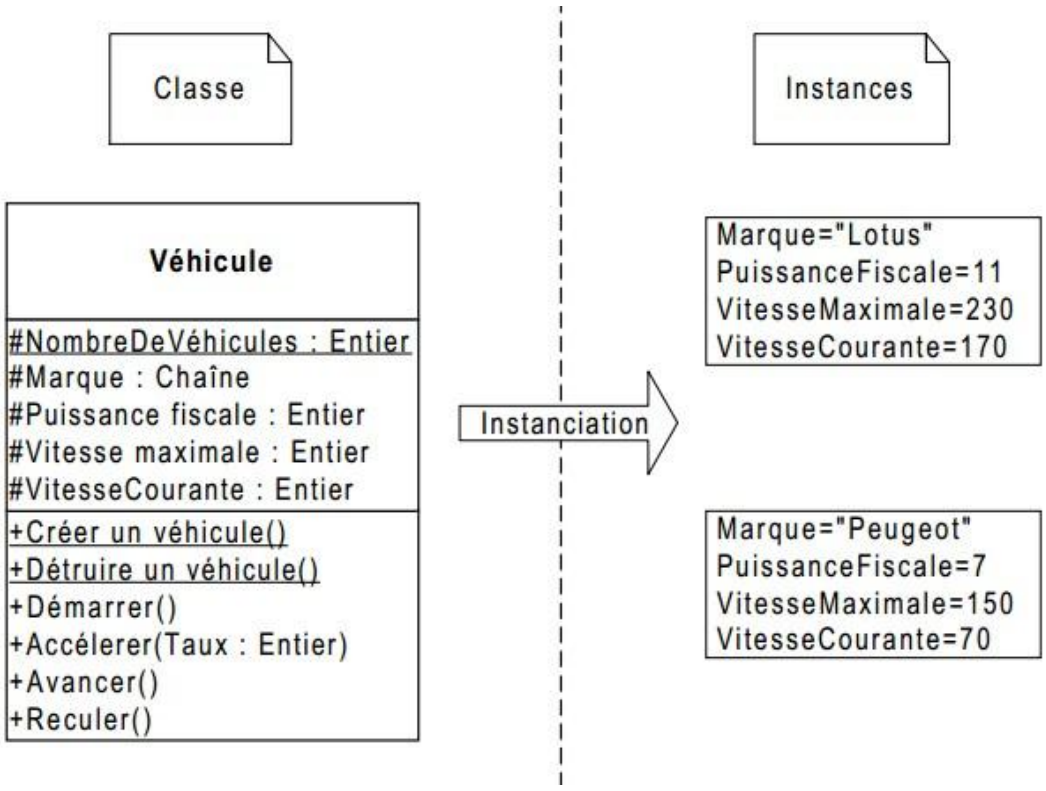


Figure 3.8 : Instanciation d'une classe en deux objets

Le même raisonnement s'applique directement aux méthodes. En effet, de la même manière que nous avions établi une distinction entre attributs d'instance et attributs de classe, nous allons différencier méthodes d'instances et méthodes de classe

Prenons par exemple la méthode Démarrer. Il est clair qu'elle peut s'appliquer individuellement à chaque véhicule pris comme entité séparée. En outre, cette méthode va clairement utiliser les attributs d'instance de l'objet auquel elle va s'appliquer c'est donc une méthode d'instance.

Considérons désormais le cas de la méthode Créer un véhicule. Son but est de créer un nouveau véhicule, lequel aura, dans un second temps, le loisir de positionne des valeurs initiales dans chacun de ces attributs d'instance. Si nous considérons en détail le processus permettant de créer un objet, nous nous apercevons que la première étape consiste à allouer de la mémoire pour le nouvel objet. Hors cette étape n'est clairement pas du ressort d'un objet : seule la classe possède suffisamment d'informations pour la mener à bien : la création d'un objet est donc une méthode de classe. Notons également qu'au cours de cette étape, l'objet recevra des indications additionnelles, telles que, par exemple, une information lui indiquant à quelle classe il appartient. En revanche, considérons la phase d'initialisation des attributs. Celle-ci s'applique à un objet bien précis : celui en cours de création. L'initialisation des attributs est donc une méthode d'instance [18]

Nous aboutissons finalement au constat suivant : la création d'un nouvel objet est constituée de deux phases:

-Une phase du ressort de la classe : allouer de la mémoire pour le nouvel objet et lui fournir un contexte d'exécution minimaliste

- Une phase du ressort de l'objet : initialiser ses attributs d'instance.

9. **Environnement Eclipse :**

Prévue pour fournir une plateforme ouverte de développement :

Fonctionne sur un grand nombre de system d'exploitation.

Interface graphique très performante et facilitant le développement d'applications. [19]

Indépendance du langage de programmation :

Permet de restriction l'utilisation plusieurs types de contenus.

HTML, Java, C, JSP, EJB, XML,…….

Facilite l'intégration de nouveaux outils :

Au niveau de l'interface et en profondeur.

Ajout de nouveaux outils pour les produits installés.

Attire une grande communauté de développeurs :

Y compris des éditeurs de logiciels indépendants.

Capitalise la popularité de Java pour l'écriture de nouveaux outils. [19]

10. **plateforme de développement java (Eclipse):**

Il fournit un ensemble d'outils permettant de créer facilement des classes

Possibilité de créer des classes JAVA comportant :

- une méthode principale « main ».
- des méthodes héritées.
- des accesseurs get et set.

Existence de plusieurs plugins permettant de développer dans d'autres langages que JAVA.

11. **Conclusion :**

Nous avons présenté, dans ce chapitre l'environnement de développement et les outils utilisés telle que java et éclipse, et ne se propose pas comme une description exhaustive de Java, le langage Java a l'avantage d'être modulaire, rigoureux la plupart des erreurs se produisent a la compilation et non à l'exécution , et portable le programme peut s'exécuter sur différente environnements

Chapitre 4
Simulation de robot

1. Introduction

Le développement de technique de navigation de robot mobile a trouvé une attention considérable dans ces dernières années. Il constitue des principes préoccupés de la recherche en robotique mobile. La navigation d'un robot mobile nécessite une structure de contrôle visant à doter ce dernier d'un mécanisme de raisonnement lui permettant de se déplacer de manière autonome dans un environnement inconnu.

Les stratégies réactives sont basées sur des informations relatives aux interactions entre le robot mobile et l'environnement inconnu, Il doit être réactif et stable et capable de réagir en temps réel.[20]

L'idée principale dans ce travail est de développer un système flou pour dresser un chemin sur la base des informations qui peut être ou ne pas être complète /ou précis et de l'adapter à un environnement donnée.

Nous nous sommes intéressés à l'utilisation de la commande floue pour le déplacement du robot mobile d'une position initiale vers une quelconque destination désirée en évitant les obstacles, tout en respectant les contraintes cinématiques du robot et de façon autonome.

Dans ce chapitre on va présenter l'architecture de notre système et par la suite on va illustrer la partie de navigation par exemple.

2. La logique floue en robotique

La commande floue a montre depuis plusieurs années sa fiabilité et sa capacité d'adaptation aux problèmes industriels concrets. Les exemples d'utilisation concluant de la logique floue sont très nombreux. La navigation par la logique floue pour un robot mobile est présentée par un système de navigation intelligent. Ce système doit être réactif est particulièrement adapte pour les applications en temps réel .[21][22]

La navigation autonome basée sur la logique floue pour un robot mobile est proposée dans ce travail ,un ensemble de règles floues ont été conçu pour la navigation dans un environnement inconnu.la première étape consiste à détecter la cible et rapprocher vers le but et la deuxième étape éviter la collision avec les obstacles.

3. Architecture du robot

Le robot simulé dans notre étude est un robot qui a une plate forme circulaire, par ce qu'elle est la plus standard et la plus utilisée dans la robotique mobile. En plus elle est facile à commander, il suffit de spécifier les vitesses des roues motrices pour faire tourner le robot.

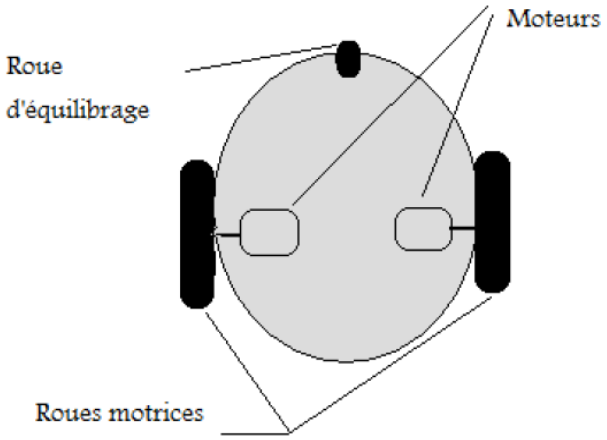


Figure. 4.1: Architecture du robot

Les capteurs jouent un rôle vital dans le fonctionnement du robot, ils fournissent une quantité d'information très importante pour cela notre robot porte trois capteurs :

1. Capteur droit.
2. Capteur frontal.
3. Capteur gauche

Le robot doit se déplacer le long d'une trajectoire, d'un point de départ à un point objectif quelconque. Par ses trois capteurs, le robot calcule la distance par rapport à un éventuel obstacle, son orientation et l'angle par rapport à l'objectif

- l'angle de vision des capteurs latéraux par rapport au frontal est 90^{o}

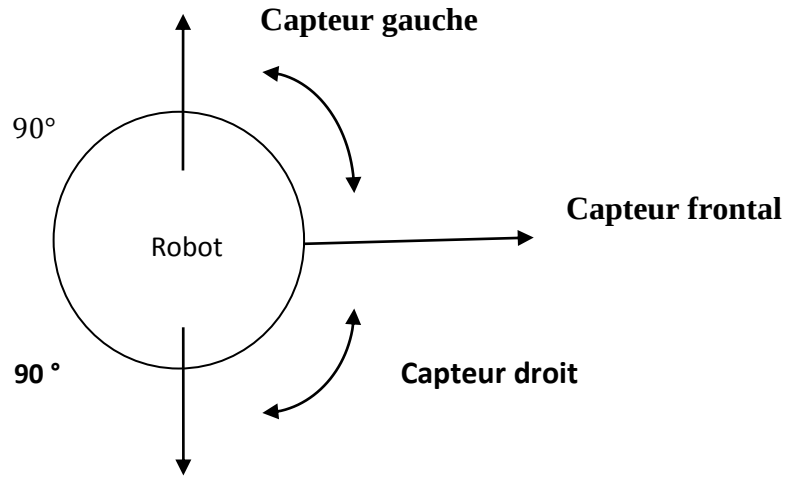


Figure. 4.2: Architecture du robot

4. Architecture structurelle du système

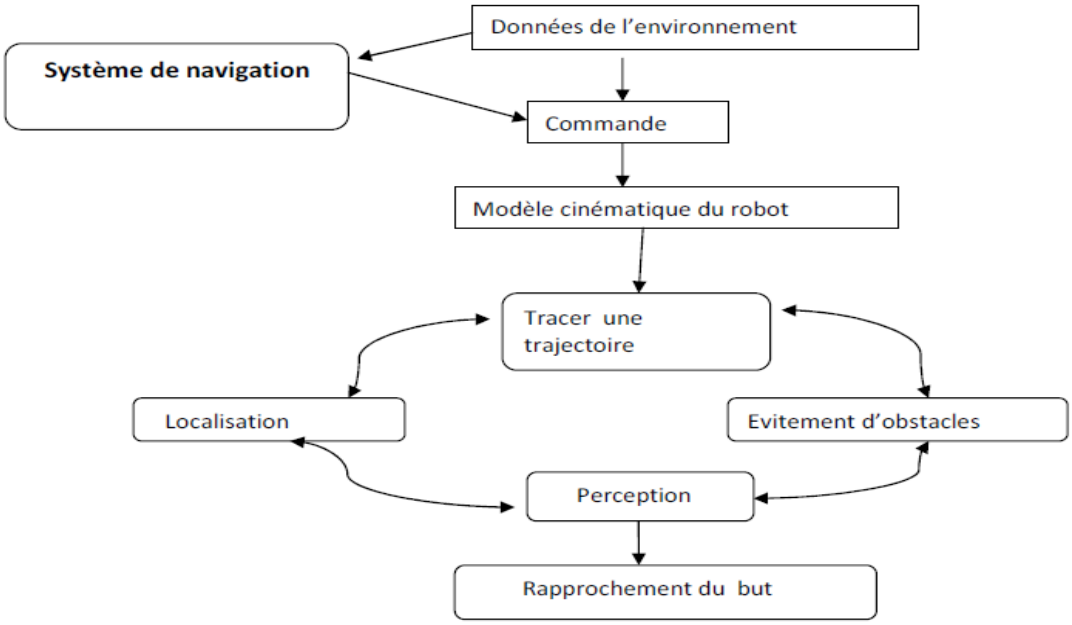


Figure. 4.3: Architecture structurelle du système de navigation proposé

Le problème à résoudre dans ce travail nécessite une architecture réactive pour querobot réagisse en temps réel afin d'éviter la collision avec les obstacles détectées. Le système de navigation autonome reçoit les données de l'environnement et génère la commande à une situation donnée. Le modèle cinématique exploite la commande pour tracer la trajectoire nécessaire pour le rapprochement du but. La figure 4.3 illustre l'architecture structurelle de

notre système ainsi que les fonctionnalités pour que le robot exécute sa tache de navigation autonome.[22]

5. Navigation du robot mobile

Dans cette section on va représenter l'approche étudiée et les structures de navigation du robot. En utilisant la commande floue .cette méthode de conception est motivée par l'efficacité et la simplicité pour rendre le contrôleur plus appropriés en implémentation temps réel. Le système de commande global du robot mobile est basé sur le modèle cinématique du centre du robot. Permettent de générer les deux actions de mouvement (angle de braquage α et la vitesse de déplacement du robot V). Ces commandes sont liées par des comportements élémentaires de base pour le robot mobile : recherche de la cible et l'évitement d'obstacles. Les actions générées sont basées sur les mesures des capteurs du robot mobile pour la détection de la position de but ou pour la mesure des distances aux obstacles .chaque comportement sont représenté par une règle avec deux entrées et une sorties sous la forme suivante :

Six1 est A1 **ET**x2 est A2 **ALORS**V est B1

Six3 est A3**ET**x4 est A4**ALORS**α est B2

- entrée : La distance séparant le robot de la cible (x1) et de l'obstacle (x2)

, elle possède trois variables floues (petite, moyenne, grande).

- Sortie : La vitesse de déplacement

Possède trois variables floues (vitesse petite, vitesse moyenne, vitesse grande).

Les règles de décision utilisées dans notre programme sont des règles universelles dans la navigation des robots mobiles. Comme il est montré dans le tableau suivant :

dis obst \ dis cible	Grand	Moyen	Petit
Grande	G	M	P
Moyenne	M	M	P
Petite	P	P	P

- entrée : La direction de la cible

Elle possède trois variables floues (Gauche, Tout droit, Droite).

- Sortie : angle de braquage.

Possède trois variables floues (angle négative, angle nulle, angle positive).

Les règles de décision utilisées dans notre programme pour la navigation des robots mobiles est comme suite :

- LE cas de présence l'obstacle :

Si la cible à gauche **ET** obstacle présent **ALORS** angle nulle.

Si la cible à droite **ET** obstacle présent **ALORS** angle nulle.

Si la cible tout droit **ET** obstacle présent **ALORS** angle Positive ou Négative.

- LE cas de l'absence l'obstacle :

Si la cible à gauche **ET** pas obstacle **ALORS** angle Négative.

Si la cible à droite **ET** pas obstacle **ALORS** angle Positive.

Si la cible tout droit **ET** pas obstacle **ALORS** angle Nulle.

6. Réalisation et résultats du test

6.1. Outils utilisés

Le logiciel utilisé c'est le java swing est un langage de programmation le plus populaire et le plus célébrée au monde et très utilisé en robotique qui apprécie sa performances et sa stabilité. Il permet de crée rapidement des applications ainsi une interface graphique d'une façon simple.

6.2. Description de la maquette

Nous présentons ici notre maquette logicielle de la commande floue appliquée à la navigation d'un robot mobile Pour atteindre le but désiré.

Notre Maquette logicielle est composé de : (Figure.4.4)

- Option Fichier
- Barre d'outils qui contient :
  - Icone de création.
  - Icone de déplacement

- Icone de redimensionne
- Icone de changer le plan
- Icone de supprimer
- Icone de supprimer tout

❖ Zone de permet de choisi la forme d'obstacle :

- carré
- cercle
- triangle
- multi rectangle

❖ trois boutons :

- bouton source
- bouton cible
- bouton naviguer

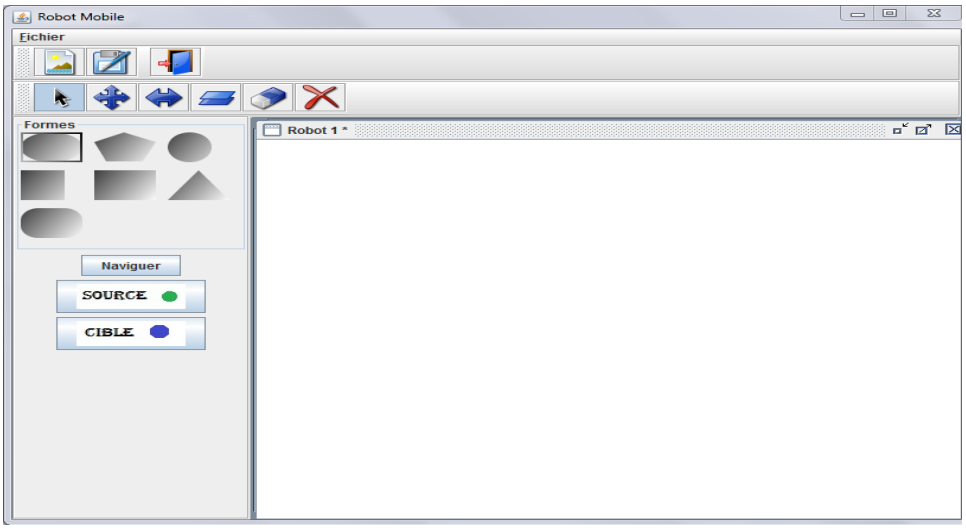
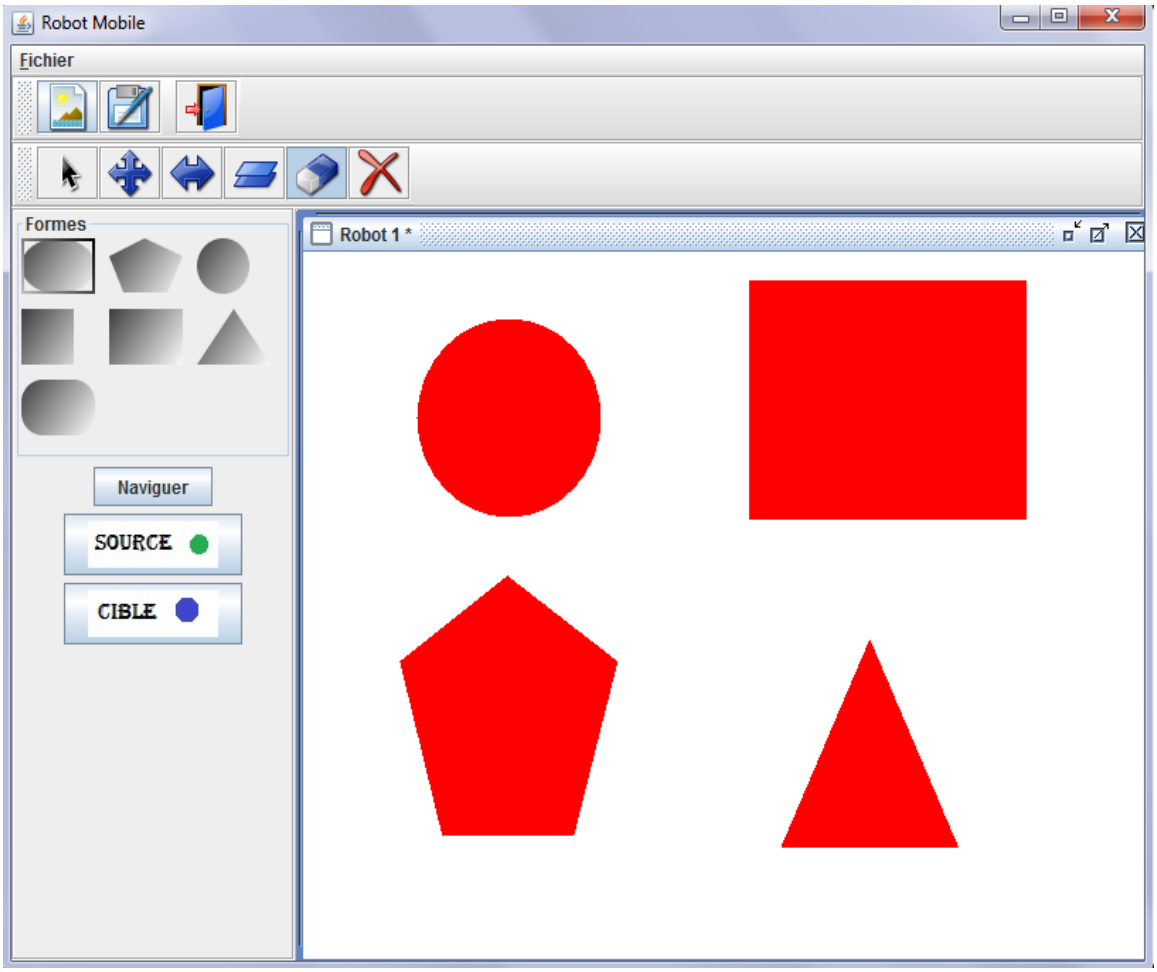


Figure. 4.4 :l'architecture de la maquette

6.3. Type des obstacles

On a utilisé des obstacles simples. Pour ce faire on a choisir 4 types d'obstacles qui sont :

1- Multi Rectangle.
2- Cercle.
3- Triangle.
4- Carré.



On dessine les obstacles avec la souris.

Figure. 4.5:les types des obstacles utilisés

6.4. Organigramme de décision

La décision doit permettre au robot de prendre en considération d'activer ou de désactiver les outils qui nous avons présentés.

On utilise un organigramme qui construit une stratégie de navigation tout le long de trajectoire d'un robot mobile.

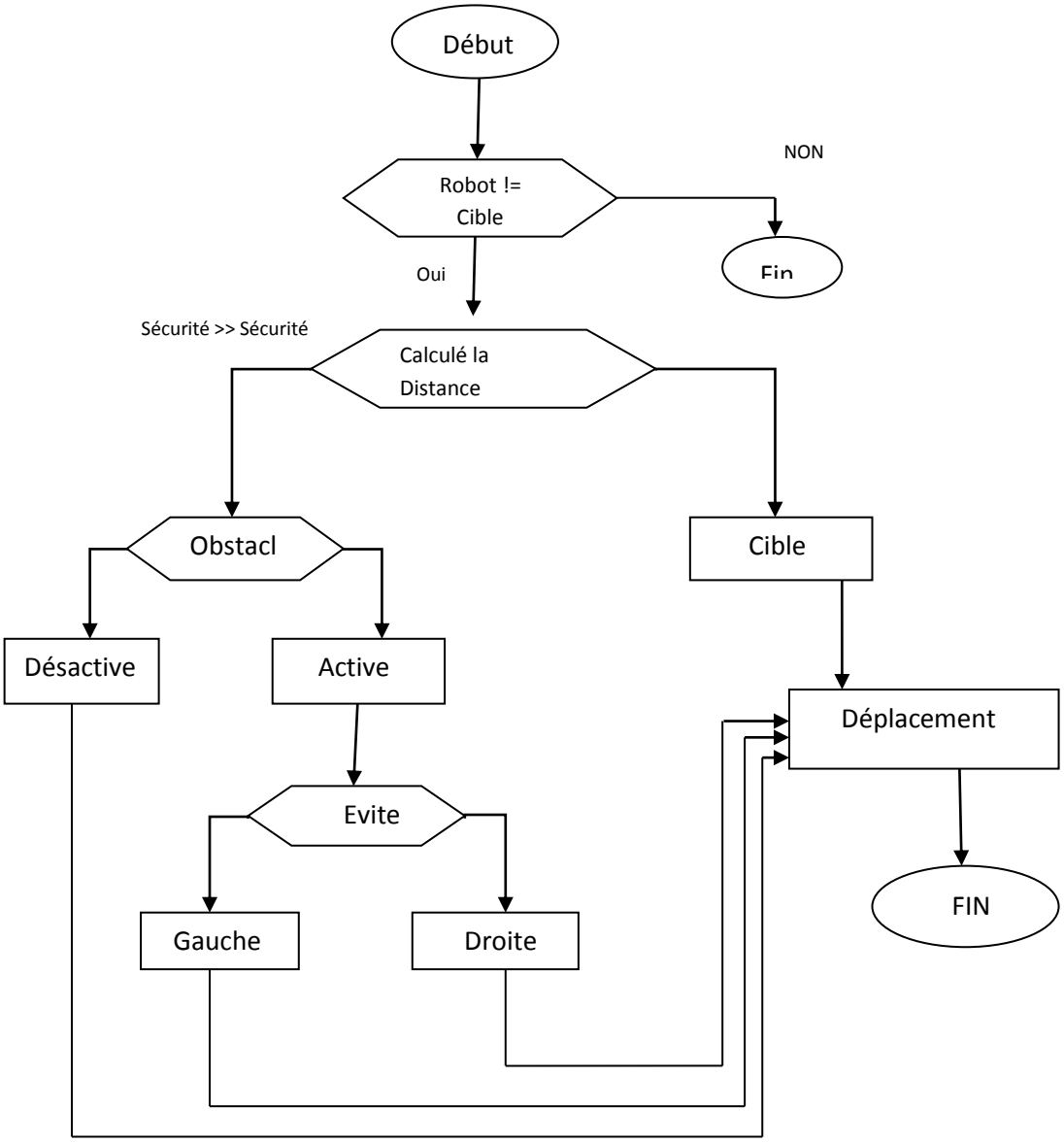
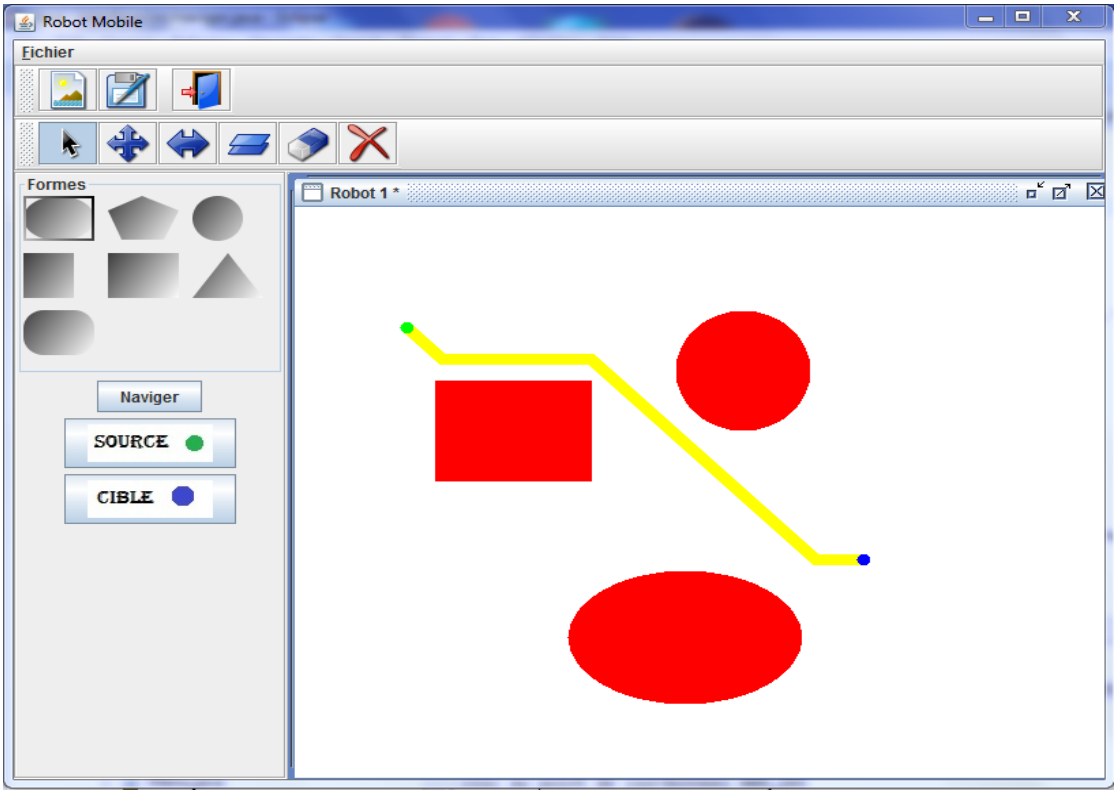
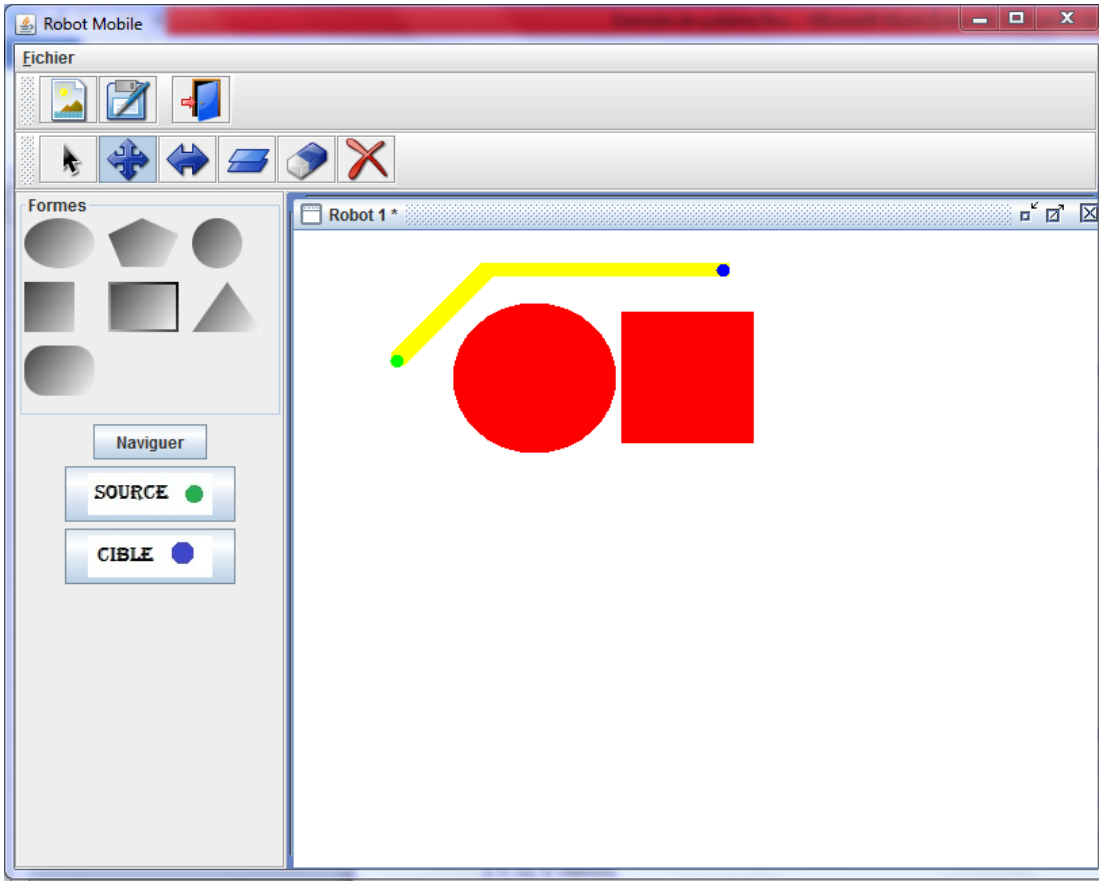
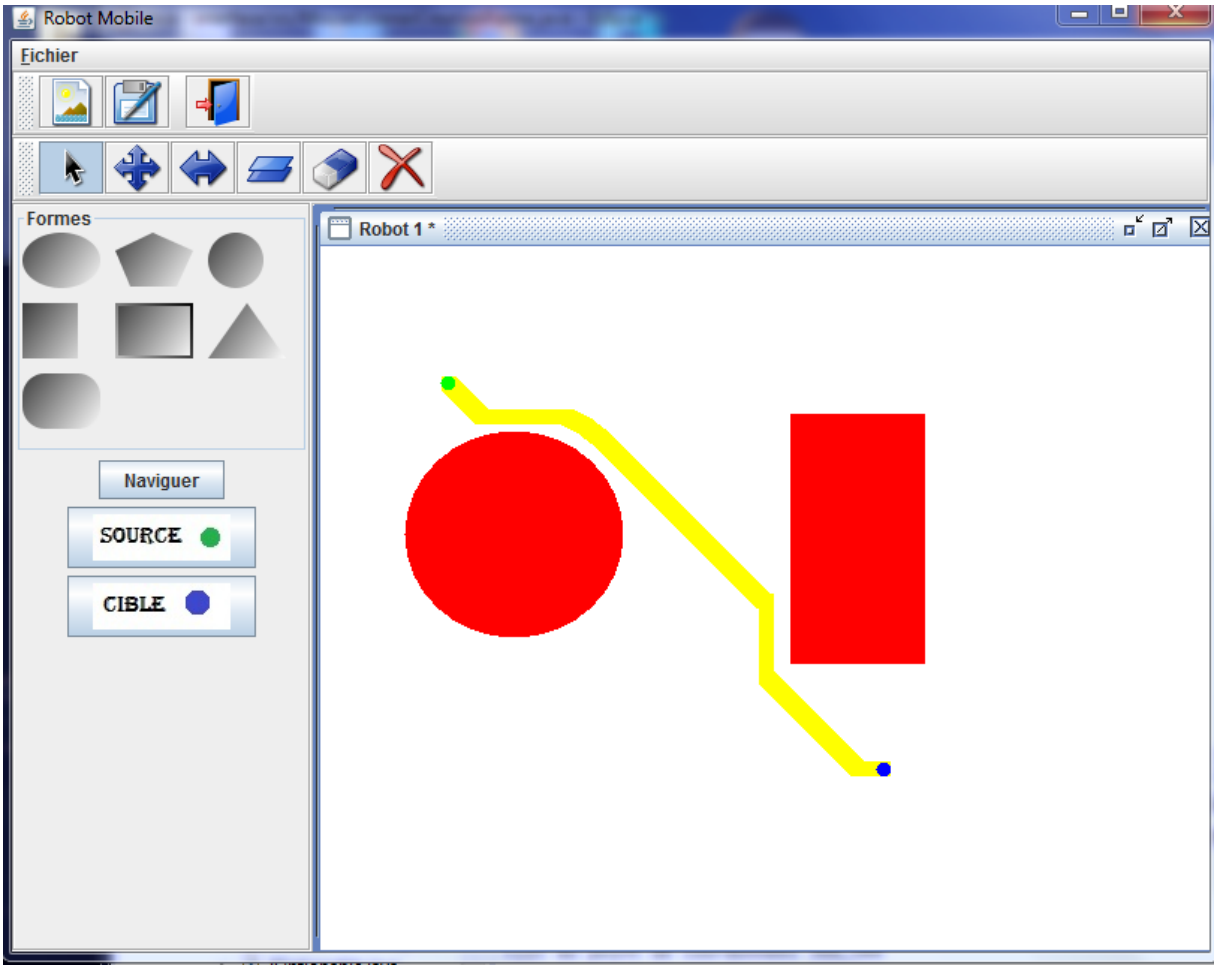


Figure. 4.6: Organigramme de décision

6.5. Résultat du test







7. Conclusion

Dans ce chapitre on a constaté que la logique floue a montré l'efficacité et la robustesse dans une navigation réactive notamment dans un environnement inconnu et plein d'obstacle avec un minimum des capteurs.

Ça se focalise sur la conception détaillée de notre système de navigation floue. Nous avons décrit la structure globale et l'architecture de notre système qui participe par un système d'inférence floue. Ensuite nous avons présenté les résultats de l'application en utilisant logiciel Eclipse de langage Java.

Conclusion Générale :

Le but de notre travail est l'étude et la programmation d'un robot mobile à trajectoire programmé avec évitement d'obstacle en utilisant la logique floue .Le robot est basé sur un capteur de distance. Pour atteindre notre objectif principal, on a passé par différente étapes :

Nous avons fait un état de l'art sur la robotique et les deux grands pôles (robotique industrielle robotique mobile). Après une présentation des différents types de robot mobile nous avons étudié les outils de perception qui permet aux robots de percevoir leur environnement et de repérer sa position initiale les positions des obstacles

Après on a présenté un aperçu sur les notions de base sur la logique floue. Comme outil adéquat est utilisé pour traiter le problème de la navigation autonome d'un robot mobile. L'emploi de cette technique, en plus de sa simplicité, à un intérêt supplémentaire qui réside dans sa capacité d'impliquer l'intelligence humaine dans sa stratégie de contrôle.Nous avons décrit l'environnement de développement et les outils utilisés telle que java et éclipse là où on a testé notre robot

Les résultats de déplacement démontrent bien l'efficacité des approches utilisées pour la conception des contrôleurs accomplissant une tache de navigation, le robot à la capacité de se déplacer dans son environnement sans aucune collision avec les obstacles avec l'utilisation des méthodes d'inférence, plus le chemin de déplacement obtenu est proche de l'optimal et plus réaliste

Grace au travail continu, on a peu atteindre notre but, mais cela ne veut pas dire qu'il est complet, nous proposons que le robot réalisé soit la base de toute une série d'améliorations que nous n'avons pas eu la chance de les faire. Le nombre d'améliorations que peuvent être ajoutes sont : utilisons de capteurs plus performants comme les laser ou les caméras et d'utilisé plus de capteurs

Bibliographie

[1] : F.ABABSA, « Commande d'un robot mobile par IBM-PC» Mémoire de magister université de Batna 1998

[2] : M. Dou, A. DJOKHRAB. « Commande D'un Robot Mobile Type Voiture Par Réseaux de Neurones». Mémoire d'ingénieur d'état en automatique, Université Biskra, Juin 2005.

[3] : A. ALLOUI, A. HAJ Brahim. «Proposition d'une solution multi-agent pour la commande et la coopération multi -robot mobile». Mémoire d'ingénieur d'état en automatique, Université Biskra, Juin 2007.

[4] : M. GHAOUI. « Planification d'un mouvement pour un robot mobile» Mémoire de magister université de Batna année 1997

[5] : B. BAYLE. «Robotique Mobile», Ecole Nationale supérieur de Strasbourg, France,

2008-2009.

[6] : A. PRUSKI. «Robotique générale» Edition Ellipase 1988.

[7] : A. PRUSKI. «Techniques De L'ingénieur, Robots Mobiles Autonomes». Université de

Metz, 1998.

[10]: PROCESSEURS FLOUS. Auteur: Gerald Huguenin, he-arc, Baptiste savoye 26, ch-2610 st-imier'

[11]: 'Article' : Implémentation d'un contrôleur flou pour la navigation d'un robot mobile de type de voiture. Auteur : N. Ouadah, O. Azouaoui, M. Hamerlain Division Productique et Robotique, Centre de Développement des Technologies Avancées, Algerie

[12] : BOUISFI Achraf 'PFE' : « Etude en simulation d'une régulation thermique par logique floue », Université sidi Mohamed ben Abdellah – Fès - Maroc

[15] M.A. Aziz Alaoui et C. Bertelle « LIVRE MN JAVA » Faculte des Sciences et Techniques Le Havre France 13 septembre 2002

[18] Introduction Aux objets ISIMA - école doctorale sciences fondamentales 1999

[19] OUAFI Abdelkrim « Réalisation d'un Robot mobile avec d'évitement d'obstacle et trajectoire programmé » mémoire de master Université Khider Biskra, 2011-2012

[20] E . Tunstel Behaviour hierarchy for autonomous mobile robot : *Fuzzy* – behaviour modulation and evolution " , Intl J . of Inteligent automationand soft computing 1997

[21]X. Yang, M. Moallem, and Rajni V. Patel, «*A Layered Goal-Oriented Fuzzy*

[22]Motion Planning Strategy for Mobile Robot Navigation », IEEE Transactions on Systems, Man, and Cybernetics, Part B, Volume 35. Volume 35, N

Webographie

[8] : http://fr.wikipedia.org/wiki/Robotique_industrielle

[9] : FANUC ROBOTICS www.fanuc.eu

[13] https://blog.erlem.fr/programmation/langage/java/11-petite-historique-du-langage-java

[14] http://blog.paumard.org/cours/java/chap01

[16] http://www.thebanque-pdf.com/fr_apprenez-java.html

[17] 1999-2016 Jean-Michel Doudoux
http://jmdoudoux.developpez.com/cours/developpons/java/index.php

Index des figures

Chapitre 01 : La Robotique

Chapitre 02 : LOGIQUE FLOUE

Chapitre 03 : Le Langage Java

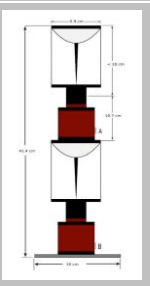
Chapitre04 : SIMULATION DE ROBOT

Liste des tableaux

Chapitre 02 : LOGIQUE FLOUE

ANNEXE A

Capteur	*Utilisation*	*Caractéristiques*	*Retour d'expérience 2010*	*2011*	*Position sur le robot*	*Photo*
Lidar 30 m (1) Hokuyo 30	SLAM	Distance max. : 30 m Angle de balayage : 270° Résolution : 0.25°.	Très bons résultats. Pas de détection des vitres, les bâches noires ne sont pas visible de loin (>15m) mais détectées proches.	Capteur reconduit. Pas de solution alternative meilleure.	Au centre à plat	
Lidar 30 m (2) Hokuyo 30	Reconstruction 3D	Non utilisé		Exploitation d'un deuxième capteur (sur un robot uniquement) pour effectuer une reconstruction 3D de l'environnement.	A 90° de l'autre lidar 30m	
Lidar 4m Hokuyo URG04-LX-UG01	Obstacles	Distance max de détection : 4m, Angle : 240°, Résolution : 0.35°.	Bonne détections des obstacles. Mais les plaques métalliques posées au sol, perturbent beaucoup les mesures et génères des obstacles négatifs.	Capteur reconduit. L'utilisation d'un Lidar30 en lieu et place, n'apporte pas d'améliorations significatives.	Devant, incliné vers le bas	
Télémètre US Maxbotix	Obstacles	Mini-module (19,9 x 22,1 x 16,4 mm) télémètre ultrason "low-cost". Doté d'une seule cellule US (émetteur/récepteur 42 KHz), il est capable de détecter la présence d'un objet de 0 à 6,45 m.	Ce module très bas coût n'est pas suffisamment fiable et nécessite un calibrage régulier. L'information de distance retournée est fortement dépendante du type d'obstacles (taille, matière, …)	Module remplacé par le module SICK	De chaque coté, incliné vers le haut	

Télémètre US SICK UM-30	Obstacles Module	industriel, intègre une électronique de filtrage et de traitement. Différents filtres sont disponibles, et il est possible de définir une fenêtre de détection.	Non utilisé	Modules intégrés aux robots	De chaque coté, incliné vers le haut	
Mesure d'attitude MEMSIC CXTLA02	Détection des pentes	Mesure +/- 20° sur 2 axes Précision 0.03 ° RMS	Non utilisé	Intégré pour évaluation	Au centre du robot (en x,y)	
Caméra détection AVT Marlin F-131B	Détection d'objets	Caméra industrielle NB est équipée d'un objectif Schneider à focale courte (4.8mm). Résolution max : 1280x1024 @ 25Hz, Capteur : type CMOS, 2/3''	Très bonne qualité d'image. La focale courte permet de couvrir un champ de vue relativement large devant le robot. L'espace au dessus du robot est également bien perçu et permet la détection d'objets placés en hauteur.	Capteur reconduit, bien que le règlement requiert l'utilisation de la couleur.	Sur le coté à plat.	
Stéréo Pano	SLAM 6DoF	Distance max de détection: 30 m ; Taille capteur : CCD - ½ '' Angle : 360° dans l'horizontale, 75° dans la verticale Résolution : 640 x 480 @ 25 Hz	Non utilisé		Derrière le lidar 30 (1)	

ANNEXE B

Solution alternatives explorées : Capteur LMS111

Pour améliorer la fiabilité, nous nous sommes intéressés à d'autres capteurs lidar La société SICK propose un capteur Lidar très compact aux performances proches du lidar HOKUYO 30m. Son gabarit lui permet d'être embarqué sur le robot, mais reste plus imposant que la version Hokuyo. Ce capteur dispose de différentes fonctionnalités de filtrage, détection d'intrusion dans une zone, robustesse à l'environnement, … Il apporte une vraie amélioration en milieu extérieur mais en intérieur les performances restent comparables. A cela s'ajoute le fait qu'il est lourd et qu'il consomme plus d'énergie. Pour le moment, nous ne comptons pas le retenir pour le défi.

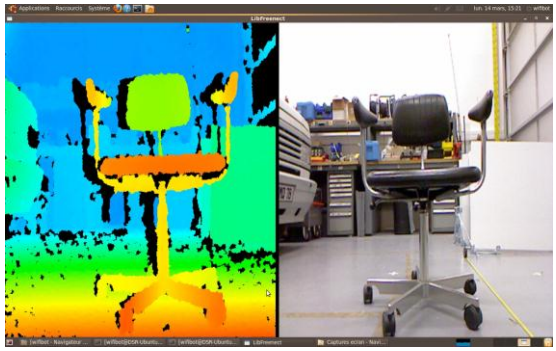
Capteur LMS 111

Kinect

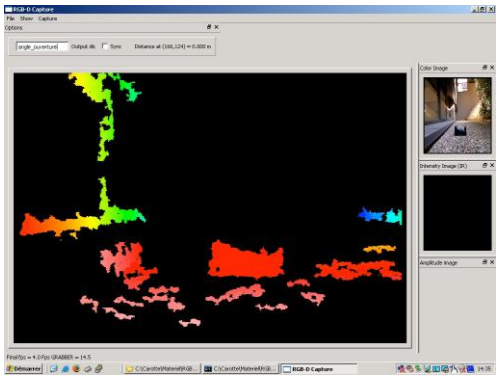
Dans le cadre de nos recherches de solutions innovantes pour à la fois la détection d'obstacles, mais également numériser l'environnement du robot, nous avons effectuer l'an dernier une évaluation d'une caméra TOF (Time Of Flight) de la société suisse MESA IMAGING, la caméra 3D Swiss Ranger SR4000. Elle nous avait été prêtée par cette société afin de l'évaluer dans notre contexte robotique. Cette année nous nous sommes intéressé à un nouveau capteur qui fait grand bruit dans le domaine de la robotique : le Kinect de la console de jeux de Microsoft. Des essais ont permis d'évaluer la précision du capteur et ses limitations.

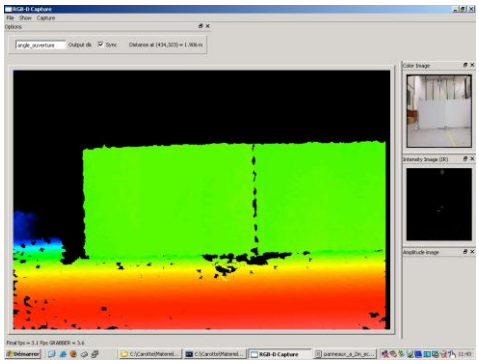
La conclusion que l'on a, après ces essais préliminaires, est que ce capteur a été conçu pour un domaine de fonctionnement particulier pour lequel il est parfaitement dimensionné. Il est tout à fait comparable en terme de performances aux caméras TOF testées l'an dernier, voir même de certains systèmes de stéréovision. Cependant

nous allons, en parallèle à nos développements, continuer à travailler avec ce capteur, pour effectuer par exemple une reconstruction 3D de l'environnement en exploitant la localisation fournie par le SLAM Lidar et une reprojection en 3D des points colorisés fournis par ce capteur.



Kinect de Microsoft





Exemples d'acquisitions : à gauche en extérieur, à droite deux plans à 2 m du capteurs séparés de 3 cm