ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

INTERFACE UTILISATEUR AVANCEE

projet de semestre - hiver 1998/99

auteur : Sébastien Grange

professeur : Reymond Clavel

assistants : Charles Baur
Terry Fong

Table des Matières

Table des Illustrations

Table des Tableaux

1. Introduction

1.1. contexte

Le problème de la téléopération a toujours été un élément crucial dans l'histoire de la robotique. De nombreuses applications reposent en effet largement sur l'efficacité de systèmes commandés à distance par des opérateurs humains, et ce pour de nombreuses raisons:

- sécurité
  de nombreux environnements sont hostiles à l'homme et requièrent un agent robotique pour leur maintenance (environnements radioactifs ou chimiquement contaminés)
- accessibilité
  de nombreuses tâches d'explorations font appel à des robots pour l'étude de lieux inaccessibles, sur terre ou ailleurs: de Dante, le robot explorateur de cratères volcaniques, à Sojourner, le célèbre héros de la mission Voyager d'exploration martienne, les exemples de robots découvreurs ne manquent pas.
- militaires
  applications certes moins nobles, mais très nombreuses. L'armée cherche à réduire les risques encourus par ses hommes tout en augmentant l'efficacité de ses opérations. Des programmes de recherche, notamment DARPA, visent à mettre au point des systèmes téléopérés militaires fiables, indécelables et simples d'utilisations, à des fins de surveillance et d'exploration.

1.2. but du projet

Le but de ce projet est la création d'une interface de commande d'un robot mobile téléopéré dans un environnement inconnu (supposé plat, typiquement l'intérieur d'un bâtiment). L'interface doit être suffisamment riche pour permettre à l'utilisateur de se former une représentation mentale correcte de l'environnement et de ce qui s'y passe, et suffisamment simple pour permettre une navigation aisée et sans formation spécifique.

La démarche suivie cherche à mettre en place une architecture d'une grande souplesse répondant à plusieurs critères récents dans le monde de la téléopération.

- utilisation du Web
  la croissance extrêmement rapide d'Internet (le cyber-trafic double tous les 18 mois) font de la Toile un outil privilégié de communication et de téléopération. Toutefois, l'utilisation d'Internet pose des contraintes diverses (cf. 2.3)
- utilisateurs multiples
  un des atouts de l'Internet est la diversité de ses utilisateurs, avec toutes les conséquences que cela implique (cf. 2.4)
- écran tactile
  les moyens à disposition au sein du groupe VRAI permettent l'utilisation d'un écran tactile, ce qui engendre quelques contraintes supplémentaires dont le design de l'interface devra tenir compte (cf. 2.5)
- grande flexibilité
  en voulant être la plus générique possible, l'interface doit être très facilement modifiable et modularisable (cf. 2.6)

Ce travail présente tout d'abord un aperçu en profondeur des contraintes susmentionnées, puis propose un échantillon des interfaces Internet "traditionnelles". Enfin, une solution de base est proposée, qui laisse une grande liberté pour des développement futurs.

1.3. conventions typographiques

Ce rapport est écrit en police Arial pour la plus grande partie, avec les exceptions suivantes :

- les `extraits de code source` sont imprimés en police `Courier`
- les mots et expressions *en italiques* sont en général référencés dans le *glossaire*

2. Contraintes

2.1. Les systèmes téléopérés

Dans le monde de la robotique, beaucoup d'interfaces existent qui veulent rendre la téléopération plus efficace et plus sure. De nombreux problèmes se posent cependant lorsqu'un système distant et équipé de capteurs est opéré par un agent humain.

Plusieurs facteurs technologiques affectent l'efficacité et l'aisance d'une téléopération :

- bande passante
  la qualité et la quantité des informations fournies par le robot sur son environnement sont des éléments clés de la facilité pour l'opérateur de se représenter la situation, et donc de prendre les bonnes décisions
- le délai de transmission
  autre problème technologique lié à la qualité de l'information, le délai de transmission prend en compte le traitement des données et le temps inhérent à leur transfert. Il a été montré [1] qu'au-delà d'un certain délai, il devenait difficile pour un opérateur de contrôler un robot par *commande directe* (i.e. en utilisant un joystick ou tout autre mécanisme de commande de direction instantanée)
- les capteurs embarqués
  même dans le cas d'une transmission idéale sans délai ni limitation de bande passante, la perception de l'environnement du robot par l'opérateur reste liée au nombre, à l'efficacité et à la diversité des capteurs embarqués sur l'engin. Les capteurs de distance les plus courants (*ultrasoniques*, infrarouges) offrent des informations très limitées et peu fiables.
  Une alternative courante à ces capteurs de base est l'utilisation d'une caméra, l'image étant l'un des outils les plus naturels à l'homme. D'une image simple transmise à l'opérateur jusqu'aux modèles beaucoup plus complexes de reconstruction tridimensionnelle de l'environnement, basés sur la vision stéréoscopique, les applications de visions varient passablement. Le choix est généralement un compromis entre qualité de l'information et temps de traitement/transmission de l'image.

Un autre problème commun à tous les systèmes téléopérés est le facteur humain. Plusieurs facteurs limitent les interactions entre l'homme et la machine distante :

- orientation dans l'espace
  un des problèmes majeurs de la téléopération est la difficulté qu'a l'opérateur à se représenter le *monde* tridimensionnel dans lequel évolue le robot. Ce phénomène se traduit surtout par une incapacité à garder la notion de l'orientation du robot dans l'espace. Même en présence d'une image vidéo fiable et continue, l'opérateur est souvent incapable d'évaluer sa position par rapport à des éléments extérieurs. Les distances sont également difficiles à évaluer.
- construction mentale (situational awareness)
  La capacité d'un opérateur humain à construire une représentation mentale de l'environnement distant est un facteur déterminant dans l'efficacité de la téléopération d'un robot mobile. Les mêmes raisons qui font perdre le sens de l'orientation du robot à l'opérateur s'appliquent à la position d'éléments extérieurs (murs, portes, objets), et de ce fait l'image seule n'est pas suffisante à une navigation efficace.

- commande directe
  dans le cas où l'opérateur contrôle directement toutes les réactions du robot (vitesse, direction) en s'aidant d'un retour vidéo, il est extrêmement sensible aux délais de transmission, plus qu'à la qualité de l'image. L'exemple le plus courant étant sans doute la tendance de l'opérateur à survirer, puis à osciller autour d'une trajectoire rectiligne lorsqu'il ne voit pas une réaction immédiate du robot sur l'image.

Un aperçu plus complet des problèmes courant de téléopération de robots mobiles peut être trouvé dans [2].

2.2. matériel utilisé

Pour le développement de ce projet, nous avons utilisé un robot mobile produit par ActivMedia, Inc. (USA), de type Pioneer AT. Il est équipé d'une caméra autofocus directionnelle Sony EVI-D30, permettant des mouvements latéraux et verticaux, en plus d'une capacité de zoom.

Les capacités de calculs on-board du Pioneer étant très limitées, une station de base est nécessaire. Elle consiste en une Silicon Graphics O2, configurée en serveur Web, et sur laquelle tourne le *contrôleur* du robot.

Les liaisons entre le robot et la station de base se font au moyen de deux radio-transmetteurs, l'un dédié au microcontrôleur embarqué qui communique avec le contrôleur, l'autre utilisé par la caméra embarquée et relié au serveur Web. La faible bande passante des radio-transmetteurs (de l'ordre de 9600 bds) et leur relative fiabilité (erreurs de transmission) sont un élément perturbateur non négligeable des performances de la téléopération.

Le contrôleur utilisé est *Saphira*, mis au point au SRI (Menlo Park, CA, USA). Une fois connecté au robot, Saphira permet de définir des *behaviours* (comportements), tels que le déplacement vers un point définit dans le monde géométrique du robot et l'évitement d'obstacles. Ce contrôleur très évolué permet ainsi au robot de prendre le pas sur une commande donnée par l'utilisateur et qui entraînerait une collision. La semi-autonomie ainsi donnée au robot augmente grandement l'efficacité de la téléopération. Un aperçu plus complexe des commandes utilisables et de leurs interactions est donné au point 5.3.

Pour les données techniques complètes des différents éléments, se référer à l'Annexe C.

2.3. utilisation d'Internet

L'utilisation très répandue d'Internet en fait un outil de choix pour le développement d'interfaces de systèmes téléopérés, ne serait-ce que par la facilité de mise en place et d'accès que le Réseau des réseaux offre.

Ainsi, l'interface est potentiellement accessible par des millions d'utilisateurs déjà familiers aux techniques de navigation Internet. Cet atout certain permet entre autre la mise en place de pages d'aides sous forme de lien hypertexte très facilement et intuitivement accessible.

Cependant, travailler sur Internet ne va pas sans limitations.

bande passante

Un des inconvénients majeurs du Web est sa faible bande passante. Le transfert des informations et leur traitement à travers les couches du réseau limitent la richesse des échanges entre l'interface et le système téléopéré. La qualité des données est en permanence altérée par deux phénomènes liés à l'architecture même d'Internet:

- la quantité d'information pouvant être transmise est lourdement limitée, empêchant tout transfert de données complexes entre le contrôleur et l'interface.
- afin de prévenir toute corruption des informations, les protocoles de communication utilisés par Internet ne peuvent pas garantir une transmission continue. Il peut donc y avoir un délai sensible et inconstant dans les échanges de données.

Ces limitations incontournables ont deux conséquences majeures:

- l'information doit être traitée sur le site même, puis envoyée à l'interface de manière la plus compacte possible, afin de limiter au maximum le trafic par Internet
- le recours à des éléments traditionnels de téléopération très gourmands en bande passante, tels que images vidéo continues ou retour de son continu sont à bannir si l'on veut éviter des retards dans la communication et l'affichage des données

Par conséquent, il n'est pas possible d'avoir un contrôle direct en temps réel, tel que joystick ou joystick virtuel, de l'interface sur le robot par le biais d'Internet.

software

Travailler sur Internet oblige à se borner aux outils tolérant les restrictions de sécurité et de compatibilité qui lui sont propres. Il existe deux mécanismes permettant la circulation d'informations dynamiques sur le Web :

- CGI-BIN [3]
  les *CGI-BIN* sont largement utilisés sur Internet, notamment pour leur facilité de mise en place et la flexibilité de leur création. Il s'agit ni plus ni moins d'un programme exécuté à distance, dont toutes les sorties sont renvoyées au navigateur Internet. Cependant, et bien que la plupart des interfaces de commande de robot mobile disponibles sur le Web soient réalisées en CGI-BIN, cette technique ne permet qu'une interaction limitée (aucune fusion de donnée possible, retour d'information limité, mémorisation d'informations difficile, …)
- JAVA [4]
  langage de programmation sur Internet par excellence, JAVA permet la réalisation de programmes complexes offrant une interface graphique, la mémorisation possible des données, et un accès à certaines ressources physiques des machines sur lesquelles les *applets* JAVA sont exécutées.
  L'atout principal du JAVA est la grande interaction qu'il permet avec l'utilisateur, ainsi que la disponibilité permanente des données. La seule limitation de JAVA est un léger délai introduit dans le traitement des données, du fait que chaque programme est exécuté par un autre software appelé VM (*Virtual Machine)* afin d'assurer la compatibilité entre les différents *OS*.

De ce fait, JAVA apparaît comme le candidat idéal à la création d'une interface interactive et performante sur Internet.

2.4. utilisateurs multiples

L'intérêt d'Internet est qu'il présente un grand éventail de profils d'utilisateurs. Il devient donc intéressant de concevoir l'interface de façon à ce qu'un utilisateur novice en matière de robotique s'en sorte facilement, tout en permettant à l'expert d'effectuer des opérations complexes.

L'interface devra donc offrir un mode de commande intuitif et simple d'approche, en tolérant cependant des opérations plus évoluées, le cas échéant en ayant recours à des modifications mineures.

2.5. écran tactile

L'utilisation d'un écran tactile engendre des contraintes sur le design de la *GUI* (Graphical User Interface):

- le doigt est plus large qu'un pixel
  s'il est possible avec une souris d'avoir une précision de l'ordre de 5 pixels, la largeur du doigt, ajoutée à l'imprécision de la détection du touché par l'écran tactile, oblige à utiliser des objets interactifs de grandes tailles. Ainsi, les boutons et autres zones à cliquer doivent-elles tenir compte de cette imprécision.
- le doigt est monomodal
  alors que la souris, sur la plupart des systèmes, possède au moins deux boutons et peut être déplacée à l'envi sans ou avec pression, l'utilisation du doigt sur l'écran tactile ne permet guère que deux fonctions (le clic et le glissement), limitant ainsi les interactions avec l'utilisateur.

Ces considérations imposent l'utilisation d'un nombre limité d'éléments graphiques. Ces composantes devront en outre être de grande taille sur l'écran, contraignant une fois de plus à développer une interface simple et limitée.

2.6. flexibilité

Pour toutes les raisons mentionnées ci-dessus, l'interface de base, exécutée à travers Internet sur un écran tactile, n'offrira qu'un éventail de possibilités limité, qui doit être facilement modifiable ou extensible.

En renonçant, par exemple, à l'écran tactile, il devrait être possible d'ajouter en un rien de temps des fenêtres de contrôle supplémentaires permettant d'agir sur des composantes que l'interface de base ne prend pas en compte pour des raisons de simplicité.

Cette flexibilité devra se traduire dans le design de l'interface par une architecture permettant l'ajout d'*objets* JAVA à l'interface de base sans devoir modifier celle-ci. Ainsi, n'importe quel utilisateur désirant adapter l'interface à son robot ou ajouter une certaine fonctionnalité pourra le faire sans devoir modifier la programmation existante.

3. State of the art

Avant de proposer une solution tenant compte des multiples contraintes développées ci-dessus, il convient d'analyser quelques exemples d'interface de robot mobile disponibles sur Internet, et leurs limitations.

3.1. Kephera on the web

Mis au point à l'EPFL par le LAMI, Khépéra on the Web [6] consiste en un robot de la classe Khépéra développé par K-team et équipé d'une caméra CCD de tourelle. Ce robot est pilotable par le biais d'une interface CGI-BIN disponible sur Internet. L'interface graphique de Khépéra on the Web est montrée à la figure 3.1.

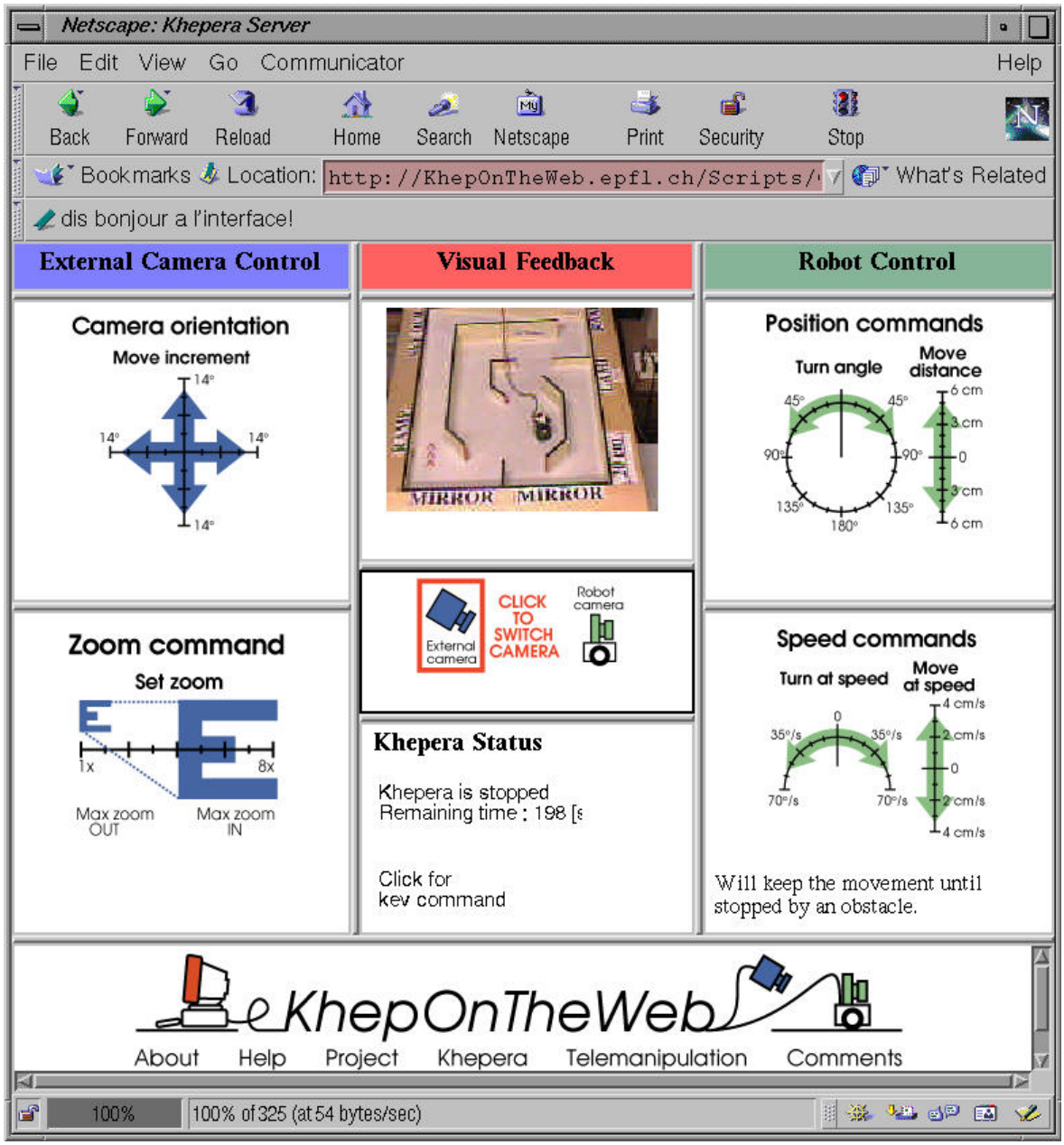

figure 3.1 – Khépéra on the web GUI

Les caractéristiques principales de cette interface sont les suivantes:

- la page Web est divisée en 8 *frames* de taille ajustée pour un écran 800x600
- chaque frame contient soit une information statique soit un outil de commande
- le seul élément dynamique est l'image brute retournée par la caméra CCD embarquée par le Khépéra
- chaque fenêtre de commande consiste en une image reliée à un CGI-BIN ; chaque CGI-BIN transmet soit une commande au contrôleur du robot, soit un retour d'information à l'utilisateur
- tous les modes de commande directs sont disponibles, des commandes par références absolues aux commandes par vélocités, en plus des commandes d'orientation et de zoom de la caméra embarquée.
- le seul élément permettant à l'utilisateur de situer le robot dans son environnement est une caméra « bird-eye » placée au-dessus du monde du robot. Un bouton permet de passer de la vue de la caméra embarquée à la caméra extérieure.

Cette interface présente plusieurs limitations, dues notamment au fait qu'elle a recours à 6 différents programmes de type CGI-BIN cloisonnés et parfois concurrents:

- le fait que la navigation se fasse à l'aide d'une caméra « bird-eye » limite le monde du robot au champs de vision de cette caméra, soit une aire d'env. 2 m^2. La navigation faite uniquement à l'aide de la caméra embarquée est par trop inefficace, et l'utilisateur se perd presque inévitablement
- toutes les commandes sont disponibles sur la même page, ce qui nuit passablement à la clarté de l'interface. Si un utilisateur initié au monde de la robotique peut sans trop de problèmes s'y retrouver, il n'en reste pas moins que l'aspect graphique de l'interface est complexe et peu intuitif
- les commandes disponibles, qu'elles soient absolues ou en vélocité, ne sont pas les plus appropriées dans la mesure où l'orientation du robot dans son monde n'est pas connue de manière claire
- le robot ne peut naviguer que dans un monde construit à son échelle. L'interface est entièrement adaptée à ce monde limité, et ne permet pas d'appréhender des situations inconnues ou imprévues.

En conclusion, l'interface du Khépéra on the Web, si elle offre toutes les commandes directes souhaitables, est bien trop complexe à utiliser pour un utilisateur novice et ne présente pas de retour d'information suffisant pour assurer une navigation efficace. De plus, le monde du robot est entièrement artificiel et limité dans l'espace, ne laissant aucune place à des événements imprévus.

3.2. VRAI group interface (CGI-BIN version)

A l'inverse de Khépéra on the Web, l'interface du VRAI group pour le Pioneer est d'une grande simplicité, comme le montre la figure 3.2.

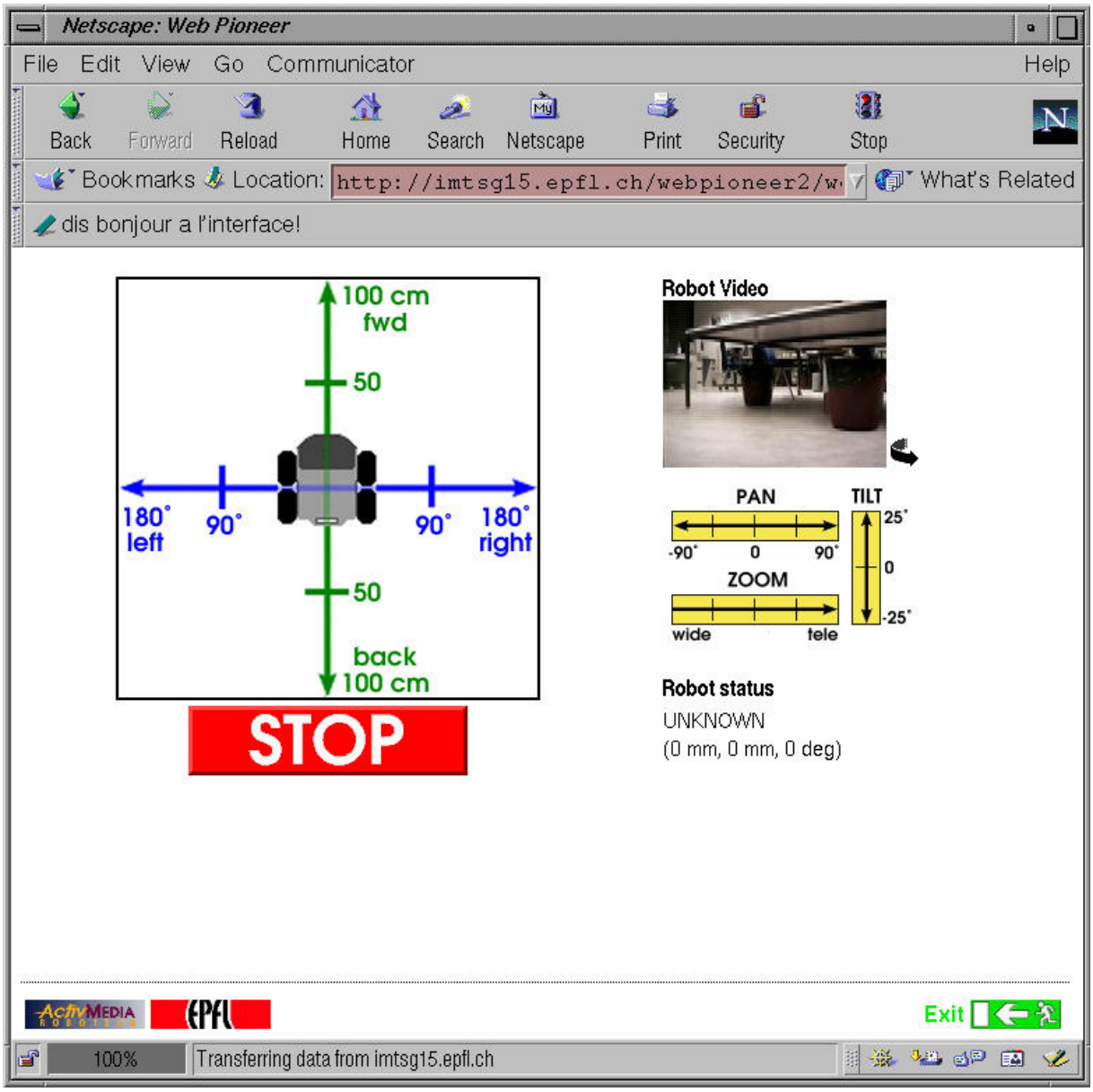

figure 3.2 – VRAI group GUI

Les éléments de base de cette interface sont:

- un mode de commande direct unique et simple, basé sur un CGI-BIN, permettant à l'utilisateur de déplacer le robot d'une certaine distance ou de changer son cap d'un certain angle
- un mode de contrôle direct de la position et du zoom de la caméra embarquée
- un retour d'image vidéo brute de la caméra embarquée
- un écran de status, indiquant l'état du robot

Cette grande simplicité, bien qu'elle rende le système facile à utiliser même pour des novices, ne va pas sans conséquences quant aux limitations de cette interface :

- aucun élément ne propose de représentation de l'espace dans lequel évolue le robot, toute la navigation étant basée sur l'image retournée par la caméra embarquée du robot
- le mode de commande demande une attention permanente de l'utilisateur, le robot s'arrêtant entre chaque commande
- l'information retournée par le robot est relativement pauvre et se borne à du texte

En résumé, l'interface du VRAI group est très intuitive à utiliser, mais n'offre que peu de possibilités d'interactions, ainsi qu'une appréhension limitée du monde du robot.

Le problème commun à ces deux interfaces, qui sont parmi les plus complètes que l'on peut trouver à ce jour sur Internet, est le manque d'aide à la navigation fournie à l'utilisateur, rendant très difficile la formation d'une représentation mentale de l'environnement exploré. De plus, l'utilisation d'un flux d'image vidéo brut ne fourni que peu d'informations utiles à la navigation et engendre des délais sensibles entre l'état de l'interface et celui du robot pour des raisons de bande passante.

4. Concept

L‘interface développée dans ce projet se veut un compromis idéal entre simplicité d‘utilisation et richesse d‘informations présentées à l‘utilisateur. Un autre souci permanent dans la mise en place de l‘interface est l‘exploitation optimale de la bande passante limitée, sans que cela n’entraîne de délai sensible dans la transmission de l‘information.

4.1. Organisation générale

Le système triple composé de l‘interface, de la station de base et du robot se répartissent les tâches de collecte et traitement des informations. Leur agencement est tel que le transfert des données entre la station de base et l‘interface est minimal.

L‘organisation générale des différentes composantes est illustrée à la figure 4.1:

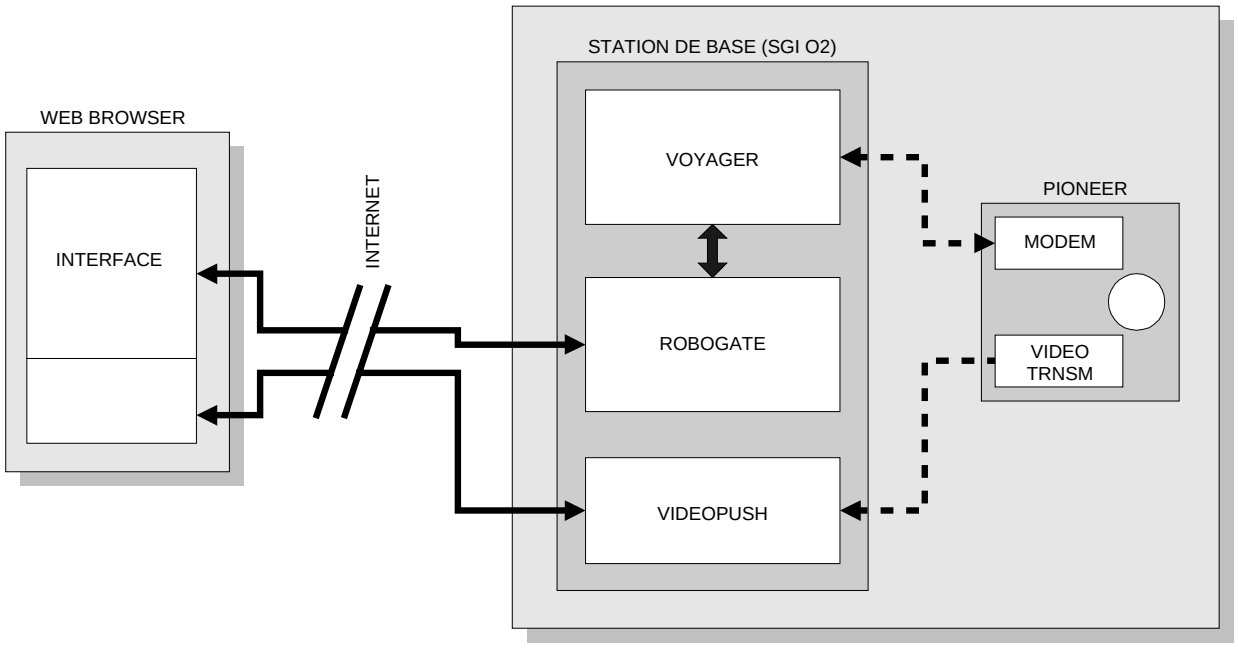

figure 4.1 – organisation générale du système téléopéré

Les composantes principales sont:

- PIONEER le robot mobile
  il joue le rôle de serveur pour le contrôleur à travers MODEM et envoie un flux d‘images vidéo à la station de base à travers VIDEO TRANSMITTER.
- VOYAGER le contrôleur du robot
  il s‘agit d‘une version de Saphira recompilée avec les caractéristiques du Pioneer. Pour plus d‘informations sur Saphira, consulter [5]
- ROBOGATE l‘interpréteur de commande
  ce programme attend sur un port dédié les requêtes de connexion faites par le biais d‘Internet. Il reçoit ensuite les commandes envoyées par l‘applet et les transmet à Voyager. Dans le même temps, il relaie en permanence le *vecteur d‘état* du robot vers l‘interface à travers le Web.

- VIDEOPUSH le décodeur d'image
  il s'agit d'un CGI-BIN appelé par l'interface chaque fois que celle-ci a besoin d'une nouvelle image prise par la caméra embarquée du robot (cf. 5.3)
- IUA interface utilisateur avancé
  pour une description détaillée de l'architecture de l'interface, consulter la section 4.2.

Cette architecture présente l'avantage que le traitement des données n'est effectué ni par le robot, qui ne dispose que d'un microcontrôleur de faible puissance de calcul, ni par l'interface, ce qui nécessiterait un transfert d'une grande quantité de données brutes à travers le Web. La puissance de calcul est concentrée sur la station de base qui gère le contrôleur du robot et la compression de l'image.

4.2. Organisation de l'interface

L'organisation de l'interface dépend lourdement de sa nature, autrement dit de l'outil utilisé pour sa programmation. Parmi les outils de programmation Internet disponibles, JAVA est sans doute le candidat le plus prometteur. Les atouts de JAVA sont les suivants :

- platform independant
  pour autant qu'une *applet* JAVA soit chargée par un *browser* supportant JAVA, elle peut être appelée depuis n'importe quel *système d'exploitation* et s'exécutera de la même manière (en théorie du moins…)
- orienté objet
  JAVA est un langage *orienté objet*, ce qui permet de définir des modules et des éléments distincts de manière simple et efficace particulièrement adaptée à ce projet
- langage haut niveau
  autre avantage du JAVA, de nombreuses routines de bas niveau telles que gestion de réseau et gestion du graphisme sont définies dans le standard JAVA, soulageant le programmeur de nombreuses heures de programmation difficiles et répétitives.

L'interface est une applet JAVA composée de 4 objets de base se répartissant les tâches de gestion et de traitement des données comme illustré à la figure 4.2.

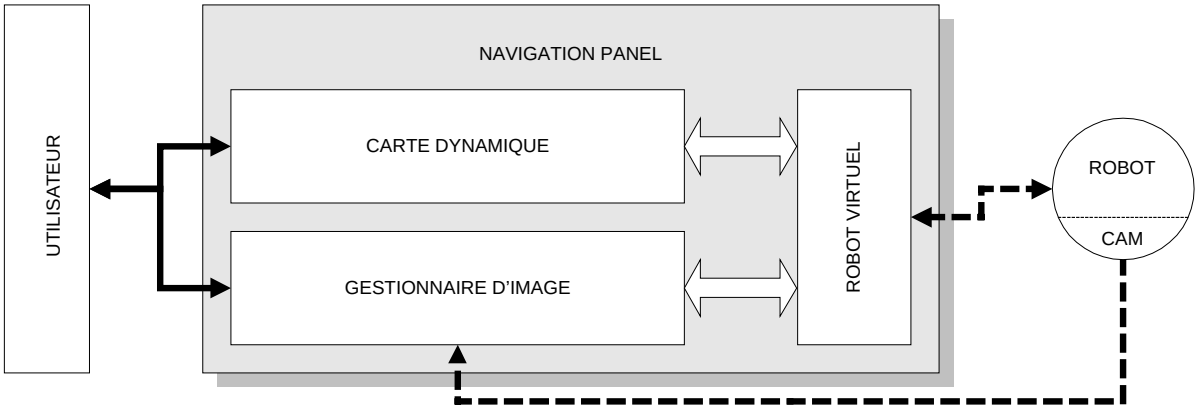

figure 4.2 – organisation de l'applet

Ces 4 composantes sont:

- NAVIGATION PANEL
  l'élément principal de l'applet, celui qui contient toutes les *méthodes* de base standardisées pour l'usage de JAVA sur Internet. Il gère l'initialisation des paramètres, l'affichage des éléments et contrôle leurs communications.
- ROBOT VIRTUEL
  cet objet JAVA contient un lien de type Internet vers la station de base et gère les communications bidirectionnelles avec le robot. Il s'occupe de décoder les informations reçues et de les transmettre aux autres objets. Il est également responsable de la transmission des ordres au robot réel (cf. 4.3).
- CARTE DYNAMIQUE
  élément clé de la navigation, la carte dynamique est construite à l'aide des données retournées par les capteurs embarqués du robot. Cet objet filtre les valeurs des ultrasoniques de Pioneer et construit un modèle approximatif du monde dans lequel le robot évolue (cf. 4.4).
- GESTIONNAIRE D'IMAGE
  le gestionnaire d'image est chargé de toutes les opérations sur les images. Les tâches qui lui incombent incluent téléportation des images, addition d'informations sur l'image en cours et affichage de l'image, ainsi que l'affichage d'images stockées à la demande de l'utilisateur. Il permet également d'ajuster le cap du robot et de donner à celui-ci des commandes directes (cf. 4.5).

4.3. Le robot virtuel

L'élément primordial de l'architecture de l'interface est le robot virtuel et toute l'infrastructure qui le supporte. C'est lui qui sert de passerelle entre le robot et les éléments de navigation de l'interface. Il assure les deux fonctions de base de l'interface, à savoir la réception et la transmission des données. Pour plus de détail sur l'implémentation complète de cette architecture, consulter la section 5.

Les fonctions remplies par le robot virtuel sont multiples et vont bien au-delà des possibilités intégrées à l'interface graphique pour écran tactile présentée au point 4.4. Elles en font un outil de base puissant et facilement utilisable dans de multiples applications de téléopérations par Internet.

4.3.1. Réception des informations

Une fois l'applet ouverte dans le navigateur Internet, le robot virtuel tente d'établir une connexion de type Internet avec la station de base. L'implémentation de cette architecture est détaillée au point 5.3.

Sur la station de base, ROBOGATE reçoit la demande de connexion et commence immédiatement à transmettre le vecteur d'état du robot au robot virtuel. Le vecteur d'état transmis consiste en un ensemble d'entiers représentants:

- la position (x, y, angle)
- la vitesse (translation, rotation)
- les coordonnées absolues des points repérés par les sonars

Ces données sont décodées et placées dans des variables internes du robot virtuel pour être utilisables par les autres éléments de l'interface.

4.3.2. Transmission des commandes

Le robot virtuel contient tout une liste de *méthodes* correspondant à des commandes reconnues par le contrôleur de Pioneer. Lorsque l'utilisateur indique une manœuvre, l'ordre est transmis au robot virtuel qui le relaie à ROBOGATE sur la station de base.

Les commandes sont nombreuses et de natures diverses :

- commandes directes relatives
  soit une rotation d'un certain angle ou une translation d'une certaine distance
- commandes directes absolues
  soit une rotation vers un cap donné
- commandes directes de vélocité
  soit avancer/tourner à vitesse constante
- commandes complexes
  soit déplacement vers un point donné ou retour à l'emplacement précédent
- commandes de caméra
  soit l'orientation verticale et horizontale de la caméra, en plus du zoom réglable

4.4. Interface graphique

Un aperçu de l'interface telle qu'elle se présente à l'utilisateur dans Netscape 4 sous Unix est présentée à la figure 4.3.

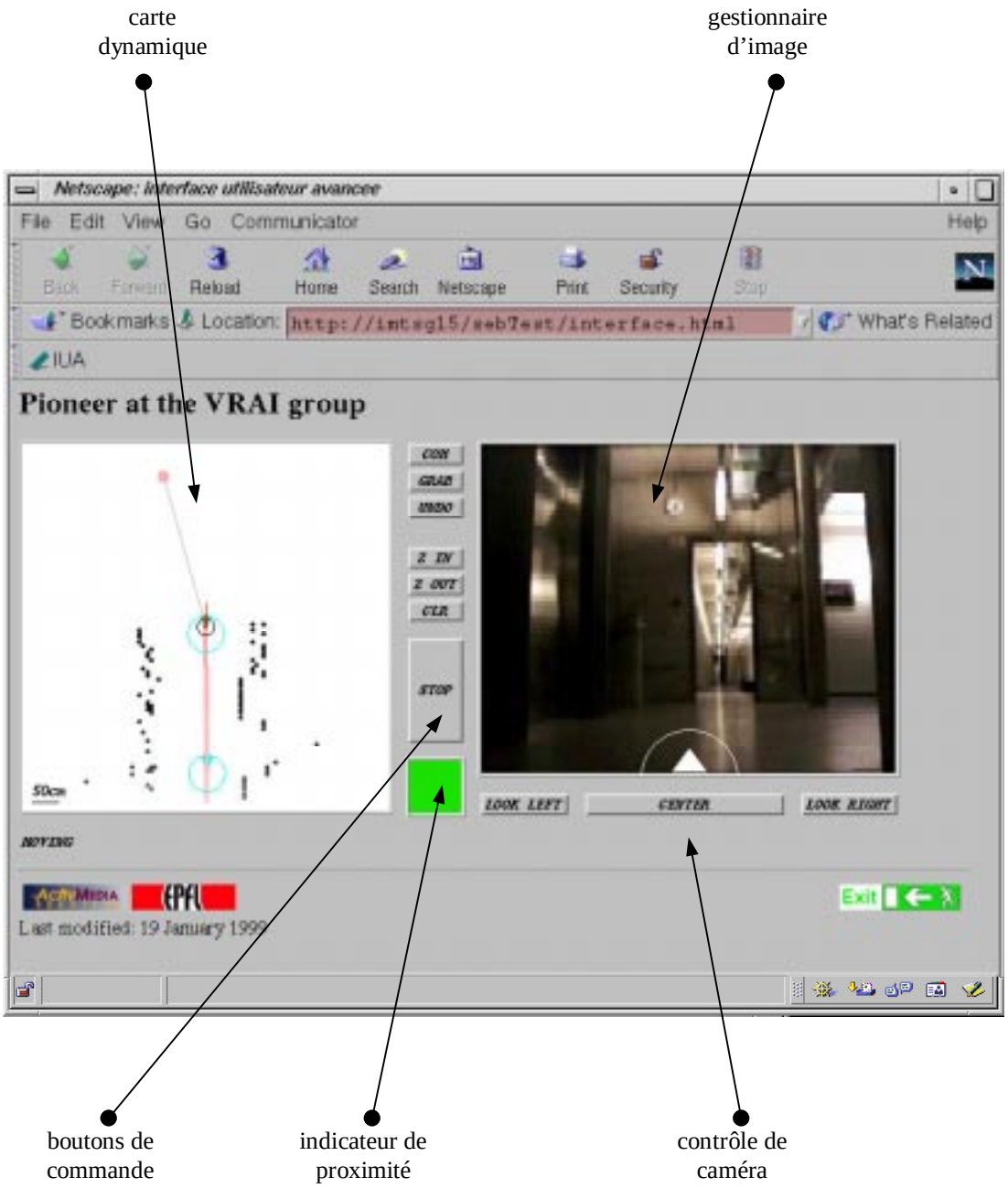

figure 4.3 – GUI de l'interface

Bien que l'aspect graphique d'une interface qui se veut intuitive et conviviale soit d'une importance certaine, il n'était pas envisageable dans le cadre de ce projet de mettre au point un design parfaitement adapté au fonctionnement de l'interface. Ce projet portant essentiellement sur les concepts d'interaction, une importance moindre à été donnée à la mise en place des éléments graphiques, opération longue et fastidieuse. L'implémentation se borne donc à utiliser des objets JAVA déjà existant, qu'il serait tout à fait possible de modifier afin d'augmenter la clarté de l'utilisation de l'interface.

Les rôles et principes des éléments constituants l'interface graphique sont détaillés ci-dessous.

4.4.1. La carte dynamique

La carte dynamique est une construction en temps réel d'une représentation de l'environnement du robot. Elle est robocentrique et est remise à jour en permanence par les valeurs retournées par les capteurs embarqués (des ultrasoniques dans le cas de Pioneer). Une représentation de la carte est donnée à la figure 4.4.

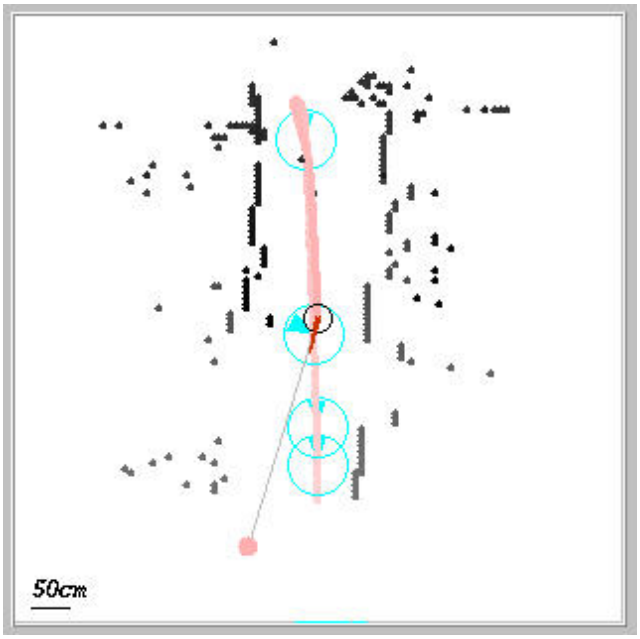

figure 4.4 – carte dynamique

La carte dynamique comme élément de navigation

Les valeurs retournées par les capteurs sont transformées en *points*, représentant des obstacles détectés dans le monde extérieur. Ces points sont ensuite affichés à l'écran relativement à la position du robot. Les valeurs brutes étant par trop entachées de bruit, un filtrage est nécessaire. Le filtre utilisé est décrit au paragraphe 5.4.

L'odométrie des robots mobiles n'étant pas parfaite, la carte dynamique contient également un mécanisme permettant d'indiquer à l'utilisateur la probabilité qu'un point soit réellement à l'endroit indiqué. Il se produit en effet un décalage dans le temps proportionnel au mouvement entre la position réelle du robot est sa position estimée. La carte dynamique évalue ce décalage et la netteté des points affiché devient proportionnelle à leur probabilité

de correspondre à un élément du monde réel. Une explication détaillée de cette technique est donnée au paragraphe 5.4.

La carte dynamique peut aussi stocker des images à la demande de l'utilisateur. Lorsqu'une image lui est transmise, la carte y adjoint la position et l'orientation de la caméra, et l'ajoute à la liste des points. A l'écran, les points comportant une image sont représentés par un cône encerclé orienté dans la direction correspondante. Lorsque l'utilisateur clic dans le cercle, l'image est affichée par le gestionnaire d'image.

la carte dynamique comme élément de commande

La carte dynamique occupe en outre le rôle d'élément de commande central. L'utilisateur peut, à n'importe quel moment, désigner un point de la carte comme objectif à atteindre par le robot. Le robot essaie d'atteindre le point choisi en ligne droite en évitant les obstacles détectés par les ultrasoniques.

Les avantages de ce mode de commande sont reliés à l'orientation Internet de l'interface :

- la trajectoire du robot est beaucoup moins affectée par les délai de transmission propres à Internet, puisqu'une fois la commande transmise, aucun traitement de donnée supplémentaire n'est nécessaire pour mener à bien la manœuvre. Le robot ne risque ainsi pas d'aller trop loin ou de survirer lorsque l'utilisateur ne peut pas réagir à temps en raison d'une congestion éventuelle du réseau.
- ce mode ne demande pas d'attention permanente de la part de l'utilisateur ; à l'opposé des modes de commande direct, il suffit d'indiquer au robot sa destination finale.

4.4.2. L'image

L'image étant l'élément le plus naturel à l'homme pour s'orienter, il est indispensable d'intégrer au système de téléopération une caméra. Cependant, pour les raisons évoquées au paragraphe 2.3, il n'est pas possible d'avoir un retour d'image en temps réel à travers Internet de manière stable et sans délai.

Une alternative est d'utiliser des images de manière discrète, et qui sont délivrées à l'utilisateur uniquement dans certains cas :

- l'utilisateur demande une image (bouton GRAB)
  dans ce cas, l'applet télécharge une image et la stocke dans la carte dynamique
- le robot s'arrête après l'exécution d'une commande
  lorsque le contrôleur détecte l'arrêt du robot, il ordonne à l'interface de télécharger une image pour faciliter la tâche de l'utilisateur
- lorsqu'un obstacle est détecté
  i.e. un sensor ultrasonique retourne une valeur inférieure à une valeur seuil prédéfinie; dans ce cas, l'interface télécharge une image.
- affichage automatique
  afin de ne pas perdre la notion de l'orientation et de garder une estimation de la vitesse de progression du robot, le contrôleur du robot ordonne à l'interface de télécharger une image toutes les 5 secondes, sous réserve qu'aucune autre image n'ait déjà été demandée dans ce laps de temps

l‘image comme élément de navigation

La caméra embarquée sur Pioneer est calibrée à chaque initialisation de l‘interface avec les paramètres suivants :

- zoom minimal
  afin de garantir le champs de vision le plus large possible
- inclinaison verticale de 10° vers le haut
  la caméra étant placée à 20cm du niveau du sol, un angle de vue vertical permet de donner une vision meilleure de l‘environnement tout en gardant un contact avec le sol
- orientation horizontale 0°
  la caméra pointe vers l‘avant du robot, direction de mouvement la plus naturelle

Cette réinitialisation est particulièrement nécessaire dans le cas de cette application basée sur Internet. Dans le cas où un utilisateur quitte l‘interface en laissant la caméra dans une orientation différente, le prochain téléopérateur ne doit pas être induit en erreur quant à l‘orientation initiale du robot.

L‘utilisateur peut à tout moment changer l‘orientation de la caméra à gauche ou à droite, par incréments de 30°. Un indicateur sur l‘image renseigne l‘utilisateur sur l‘orientation actuelle de la caméra relativement à l‘axe du robot. Si la caméra est orientée vers l’avant du robot, un triangle blanc est affiché (figure 4.6). Dans le cas où la caméra est orientée latéralement, une flèche colorée (rouge à droite, verte à gauche) ainsi que l’angle de vue sont affichés (figure 4.7).

figure 4.5 – image avec caméra centrée

figure 4.6 – image avec caméra orientée à gauche

La zone image de l‘interface permet en outre une fusion de capteurs simple. Dans le cas où un obstacle est détecté par les ultrasoniques, il engendre l‘importation d‘une image.Un triangle d‘avertissement est superposé à l‘image dans la direction de l‘obstacle (ou au bord de l‘image si l‘élément est hors du champs de vision de la caméra). Un exemple est montré à la figure 4.8. Un simple clic sur le triangle d‘avertissement suffit à le faire disparaître pour avoir une vue complète de la zone considérée.

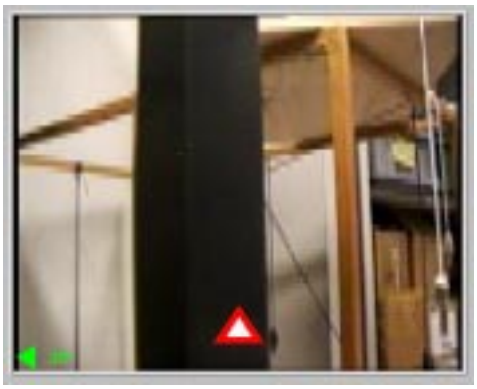

figure 4.7 – avertisseur d’obstacle

La zone image permet également d‘afficher les images stockées par l‘utilisateur dans la base de données de la carte dynamique. Lorsqu‘une image est rappelée, un ‘R‘ majuscule est superposé à l‘image (figure 4.9) pour éliminer tout risque de confusion.

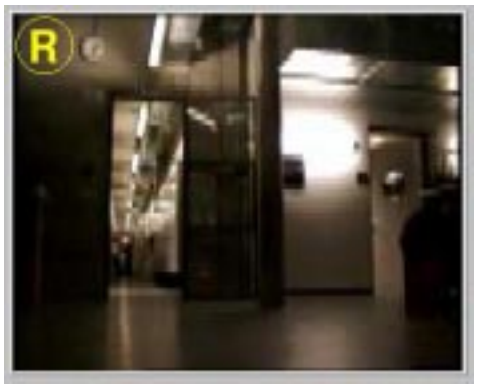

figure 4.8 – affichage d’une image stockée

l‘image comme élément de commande

La zone image peut également servir à transmettre des ordres au robot, et ce de deux manières différentes:

- changement de cap
  lorsque l‘utilisateur clic sur un point de l‘image, la position horizontale de l‘action est repérée et l‘ordre est donné au robot de s‘orienter dans la direction choisie, et ce quelle que soit l’orientation de la caméra
- translation
  en cliquant sur l’indicateur de direction lorsque la caméra est centrée, l’utilisateur fait avancer le robot de 50 cm

- commande directe
  dans certaines situations, il peut s'avérer nécessaire de recourir aux commandes directes de navigation. Par simple pression sur un des boutons de l'interface, l'utilisateur fait apparaître l'outil de commande directe illustré à la figure 4.10, qu'il peut alors utiliser pour diriger le robot.

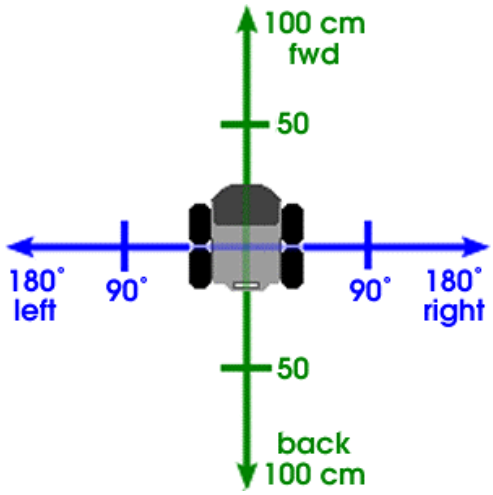

figure 4.9 – outil de commande directe

4.4.3. L'indicateur de proximité

L'indicateur de proximité est un avertisseur graphique. La couleur de la zone d'affichage change en fonction de la proximité d'obstacles éventuels, donnant ainsi à l'utilisateur un sens intuitif des dimensions du robot. Quelques extraits de l'éventail de couleur (du vert au rouge) sont montrés à la figure 4.10 :

figure 4.10 – indicateur de proximité

S'il n'est pas d'une importance capitale à la navigation, l'indicateur de proximité se révèle néanmoins utile lorsqu'il s'agit d'évaluer la distance à un élément du monde distant. Alors que l'échelle de la carte peut être choisie par l'utilisateur, l'espace réel occupé par le robot dans son environnement n'est pas facile à représenter sur la carte dynamique de manière intuitive, notamment pour des raisons de vitesse d'affichage. L'indicateur de proximité pallie donc à cette lacune.

4.4.4. Autres fonctionnalités

D'autres fonctionnalités sont également intégrées non seulement dans l'interface, mais dans l'architecture de commande complète:

- boîtes de dialogues
  le contrôleur (Saphira), le relayeur (ROBOGATE) et l'applet elle-même ont la possibilité de faire ouvrir des fenêtres d'avertissement à l'importe quel moment, pour par exemple avertir l'utilisateur d'un danger ou signaler la perte de la connexion avec le robot. Ces fenêtres d'avertissement ont un effet bloquant et ne peuvent pas être ignorées par l'utilisateur. Un exemple est donné à la figure 4.11

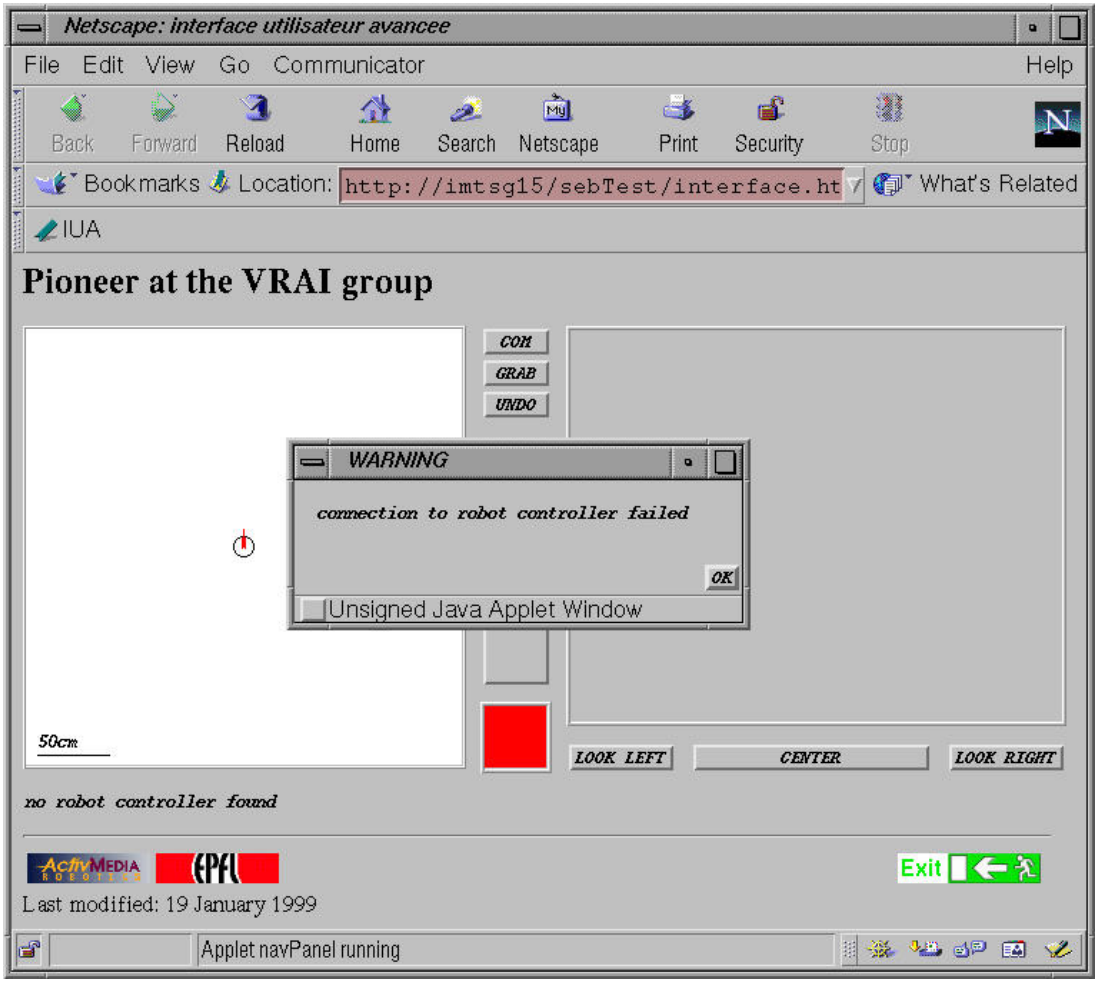

figure 4.11 – boîte de dialogue

- fonction UNDO
  plusieurs utilisateurs testés ont exprimé le désir d'annuler la dernière action. Bien que cela ne soit pas possible dans tous les cas de figure, il est possible d'annuler toutes les commandes directes en exécutant l'action contraire. Le bouton UNDO ordonne au robot virtuel de revenir à la position précédant la dernière action.

4.5. Flexibilité

L'architecture globale choisie pour cette interface se prête très facilement à des adaptations et améliorations, et ce pour deux raisons majeures :

- le robot virtuel est déjà implémenté avec de nombreuses méthodes de contrôle et d'accès aux données du *vecteur d'état*, permettant la création de nouvelles fonctionnalités sans devoir modifier la base de l'interface, à savoir la transmission de l'information par Internet.
- l'interface étant écrite en JAVA, il est facile de créer un objet JAVA indépendant et de l'interfacer ensuite au robot virtuel pour lui donner accès aux données du robot.

Il devient ainsi aisé d'ajouter de nouvelles capacités à l'interface, par exemple sous forme d'une barre d'outil contenant des boutons JAVA, chacun exécutant une nouvelle action ou ouvrant une fenêtre de contrôle.

Exemples de fonctionnalités facilement interfaçable avec la version courante :
- contrôle complet des mouvements de la caméra
- joystick virtuel
- données numériques sur l'état et la progression du robot
- données graphiques sur l'état et la progression du robot
- gestion de la banque d'image
- tout autre menu spécifique à l'application considérée (gestion de capteurs, fusion, …)

Un autre aspect de la flexibilité de l'interface a trait à son utilisation avec d'autres robots mobiles. Il est relativement aisé d'adapter la conception courante de l'interface à des systèmes téléopérés différents :

- le robot virtuel est le seul objet spécifique au modèle de robot téléopéré
- tous les autres objets, une fois initialisés avec les paramètres du robot virtuel, sont indépendants du type de robot et ne nécessitent aucune reprogrammation

5. Implémentation

La figure 5.1 décrit les interactions entre les objets JAVA composant l'interface :

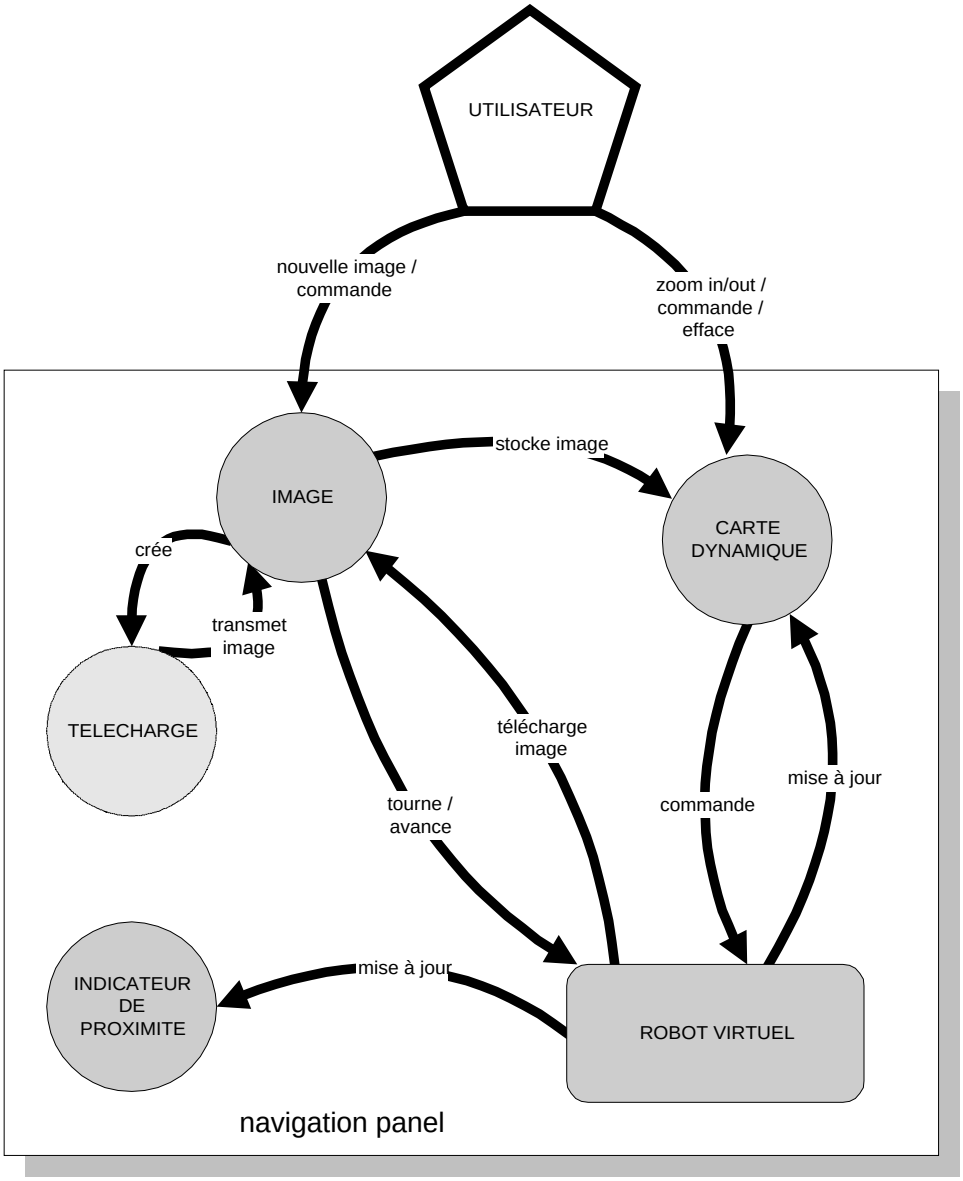

figure 5.1 – interactions des objets JAVA

5.1. le contrôleur

La première autorité de contrôle sur le robot est Saphira, le contrôleur. Saphira est un programme complexe, souple et performant, offrant de nombreux niveaux d'interaction différents avec l'utilisateur.

La version de Saphira utilisée dans le cadre de ce projet est configurée avec les paramètres suivants pour faciliter la commande à distance :

- les commandes sont prioritaires par ordre d'arrivée
  pour éviter l'accumulation ou la compétition de plusieurs commandes suite à l'encombrement des voies de communication, chaque commande transmise par l'utilisateur devient la commande en cours et les ordres précédents sont ignorés.
  Certaines commandes complexes (p .ex. une rotation et une translation simultanée) ne sont donc pas utilisables par l'utilisateur, le délai de transmission inconstant à travers Internet ne permettant pas un contrôle suffisamment efficace du mouvement.
- l'évitement de collision est systématique
  puisque les informations ne peuvent pas être transmises en temps réel garanti, et que la pauvreté des informations transmises ne permet pas toujours au pilote d'appréhender la situation, il peut arriver que le robot doive prendre lui-même la décision de s'arrêter ou d'éviter un obstacle. Lors de chaque mouvement, une routine d'évitement d'obstacle est donc superposée à la commande de déplacement.
  L'efficacité de l'évitement d'obstacle est toutefois soumise à la fiabilité et au nombre de capteurs.

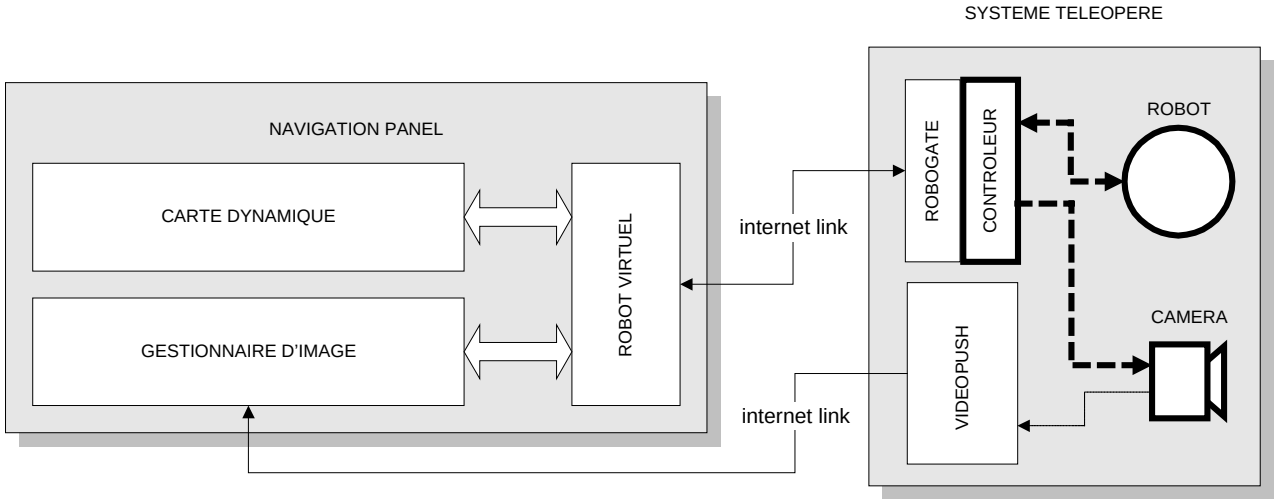

figure 5.2 – contrôleur

Le contrôleur est aussi responsable des commandes de caméra, à savoir orientation horizontale, verticale et réglage du zoom. Le module de commande de la caméra est cependant en boucle ouverte et ne permet pas de contrôle des réglages de la caméra. Dans le cas où la commande de caméra est ignorée par le robot, il n'y a donc aucun moyen de le savoir. Pour remédier à ce problème, il faudrait reprogrammer le microcontrôleur à bord de Pioneer, tâche sortant largement du cadre de ce projet.

L'interface offre quelques commandes de contrôle de caméra pour illustrer les capacités du robot virtuel, et garde trace elle-même de l'orientation de la caméra. Ces commandes ne sont cependant pas garanties.

5.2. le serveur vidéo

Le serveur vidéo est le dispositif en charge du transfert des images du robot à l'interface. Le schéma de principe de cette architecture est donné à la figure 5.3.

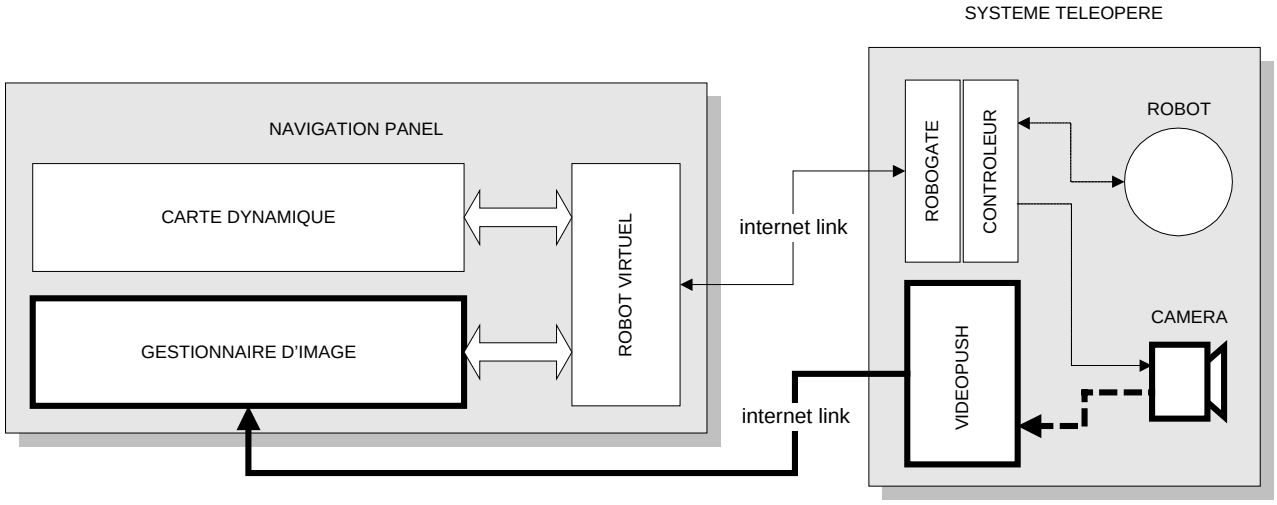

figure 5.3 – organisation du serveur vidéo

Le transfert d'une image se fait en trois étapes :

- importation de l'image de la caméra embarquée du robot à la station de base
- compression de l'image
- transfert de l'image à l'interface

La compression de l'image est rendue nécessaire par la bande passante limitée d'Internet afin de limiter le temps de transmission. L'image est convertie en format JPEG. Le format JPEG utilise des algorithmes de prédiction et un filtrage des hautes fréquences pour réduire drastiquement et de manière réglable la taille des images, avec des pertes d'informations compatibles avec la sensibilité de l'œil humain et donc minimales.

Afin de ne pas bloquer le fonctionnement de l'interface durant la transmission de l'image, le gestionnaire d'image crée un nouveau process parallèle chargé de recevoir l'image et de l'afficher une fois que la transmission est complète. Ainsi, la connexion normale entre le robot virtuel et la station de base n'est pas affectée par l'arrivée d'une image.

5.3. Le robot virtuel

Comme décrit au paragraphe 4.1, le robot virtuel sert de référence aux éléments de l'interface JAVA et permet de faire abstraction de toute l'architecture de communication mise en place entre la station de base et l'applet. Cette section s'intéresse à cette organisation sous-jacente qui synchronise le robot téléopéré avec l'interface de commande.

L‘organisation générale de la couche de communication est décrite à la figure 5.4 :

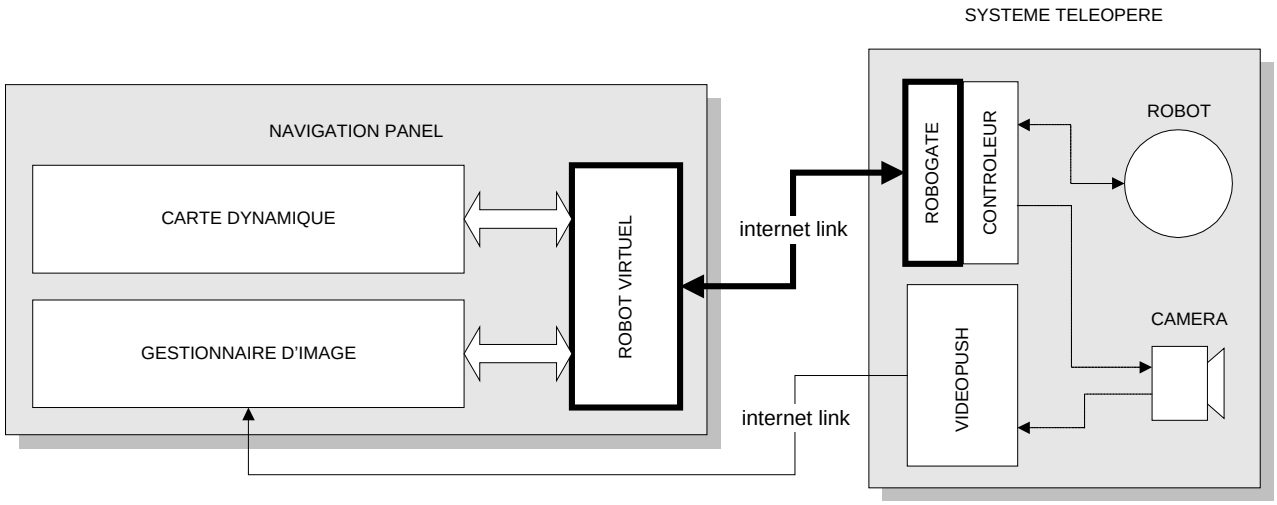

figure 5.4 – mise en place du robot virtuel

Deux éléments sont impliqués dans l‘architecture de communication.

le serveur : ROBOGATE

ROBOGATE est un programme serveur écrit en C qui attend en permanence des requêtes de connexion *TCP/IP* sur un port dédié de la station de base. Lorsqu‘une interface JAVA est chargée par le biais d‘internet, l‘objet ROBOT VIRTUEL de l‘applet envoie une requête de connexion à la station de base.

La figure 5.5 décrit brièvement le fonctionnement de ROBOGATE.

ROBOGATE utilise un pont de type *mémoire partagée* (shared memory) avec le contrôleur afin de permettre un transfert quasi instantané des informations. Le pont est initialisé lorsque le contrôleur est exécuté. Les éléments du vecteur d‘état utiles à l‘applet (positions, vitesses, capteurs, status) sont chargés dans une structure de donnée propre à la mémoire partagée par le contrôleur et lues par ROBOGATE. Une autre partie de la zone mémoire est dédiée aux commandes de l‘utilisateur, décodées par ROBOGATE et lues par le contrôleur. La figure 5.6 fourni une vision complète de la mémoire partagée.
La connexion entre ROBOGATE et l‘applet est maintenue par des *sockets TCP/IP* non bloquantes. Le recours à des socket non bloquantes [7] permet d‘utiliser la même connexion pour l‘émission et la réception des informations, évitant ainsi d‘occuper inutilement des ressources systèmes. ROBOGATE envoie à l‘applet :

- une chaîne de valeurs entières représentant le vecteur d’état du robot
- des commandes sous forme de chaînes de caractère
  consulter l’annexe B pour une liste exhaustive des commandes.

Il est également à relever que, contrairement à l‘interface, ROBOGATE est *spécifique au contrôleur utilisé.* De ce fait, adapter l‘interface à un nouveau modèle de robot soumis à un autre programme de contrôle nécessite une modification de ROBOGATE afin qu‘il soit adapté au nouveau contrôleur. Ces changement se limitent toutefois à la configuration de la structure de donnée de la mémoire partagée.

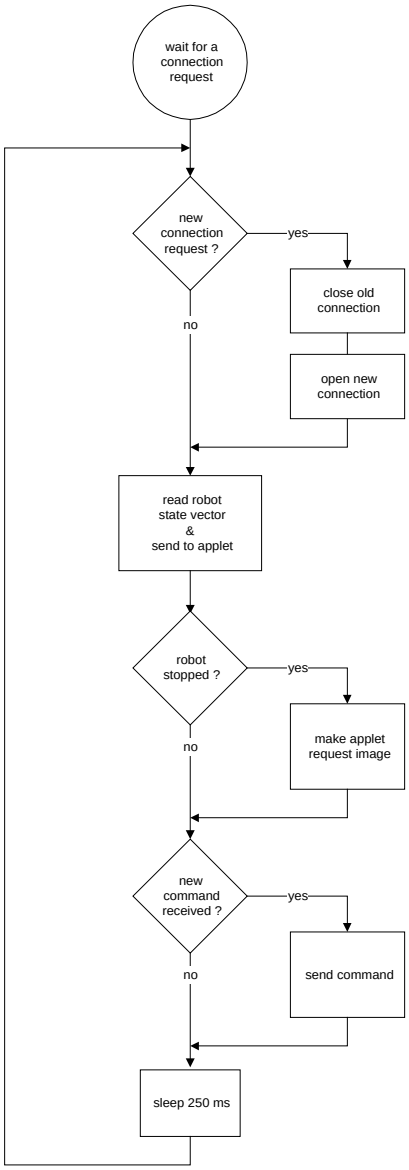

figure 5.5 – organigramme de ROBOGATE

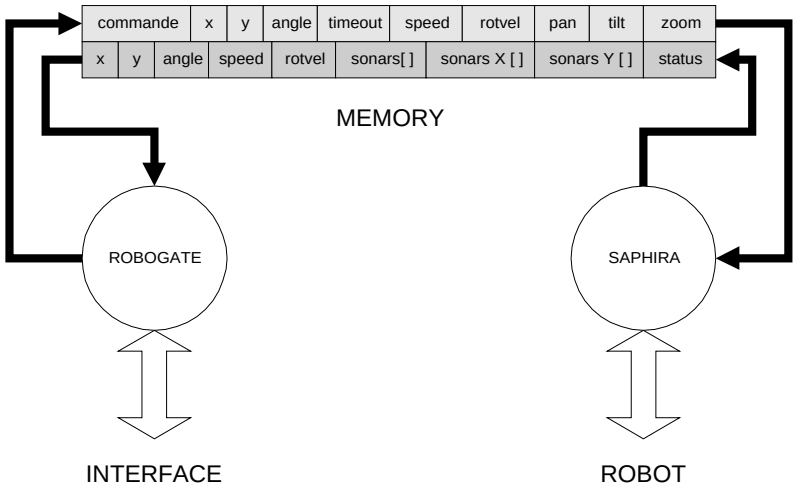

figure 5.6 – structure de la mémoire partagée

le client : ROBOT VIRTUEL

Le robot virtuel est un objet JAVA initialisé par l'interface lorsque l'applet est téléchargée et exécutée. Son rôle est parfaitement complémentaire à celui de ROBOGATE. Après avoir ouvert une connexion TCP/IP avec la station de base, le robot virtuel remplit les tâches suivantes:

- il maintient une écoute permanente sur le port TCP/IP dédié et stocke le vecteur d'état dans des variables privées ; dans le cas où des commandes sont reçues, elles sont interprétées et transmises à l'objet concerné.
- il envoie les commandes de l'utilisateur sous forme de chaînes de caractères reconnues par ROBOGATE. Pour une liste exhaustive des commandes, voir annexe B.

Le vecteur d'état est en permanence accessible par les autres objets JAVA composant l'interface. Une liste complète de la structure de données et des méthodes implémentées par le robot virtuel est donnée en annexe A.

5.4. La carte dynamique

Le premier problème à résoudre avant de construire la carte dynamique est le filtrage des données. Les capteurs de distance en général, les ultrasoniques en particulier, ne sont pas sans erreurs et retournent fréquemment des valeurs incorrectes. Les facteurs d'erreur courant pour un capteur ultrasonique sont :

- l'orientation de l'obstacle
  l'onde sonore est facilement déviée par une surface non perpendiculaire. Une réflexion peut entraîner une fausse mesure dans le cas où, après plusieurs réflexions, l'onde revient au capteur. La distance mesurée n'est donc pas la distance à l'obstacle le plus proche, mais le chemin total parcouru par l'onde.
- l'ordre de mise à feu
  plusieurs ultrasoniques fonctionnant sur le même robot sont mis à feu en alternance afin de limiter les risques de recouvrement. Toutefois, il arrive que les ondes émises par un capteur soient réceptionnées par un autre capteur, entraînant une mesure erronée.
- le matériau de l'obstacle
  l'onde sonore est absorbée et réfléchie différemment suivant le matériau composant l'obstacle.

Les mesures brutes sont donc filtrées afin de prévenir les nombreux artefacts produits par les ultrasoniques. Le filtrage est basé sur trois techniques:

- division du monde en blocs
  le monde du robot est divisé en carrés de 7 cm de côté afin de réduire le nombre de points à afficher. Il n'est en effet pas nécessaire d'utiliser une résolution de l'ordre du millimètre dans une carte à l'échelle du mètre. Chaque bloc est représenté sur la carte par un point.
- compteur de seuil d'affichage
  chaque bloc créé initialise un compteur interne qui est incrémenté à chaque fois qu'un capteur pointe à nouveau sur le même élément. Le point n'est pris en compte (i.e. reconnu comme existant et donc affiché à l'écran) qu'une fois le compteur dépassant une certaine valeur seuil.

- intervalles de fiabilité
  la fiabilité de chaque sonar dépendant de son orientation et de sa position par rapport aux autres, tous les sonars ne sont pas fiables sur les mêmes intervalles de distances, d'où le recours à un filtrage spécifique à chaque sonar. Si la valeur retournée par un sonar n'appartient pas à l'intervalle correspondant, elle n'est simplement pas prise en compte.

La figure 5.7 représente la même situation que la figure 4.4 après quelques manœuvres, sans filtrage des données. Un filtrage est visiblement indispensable.

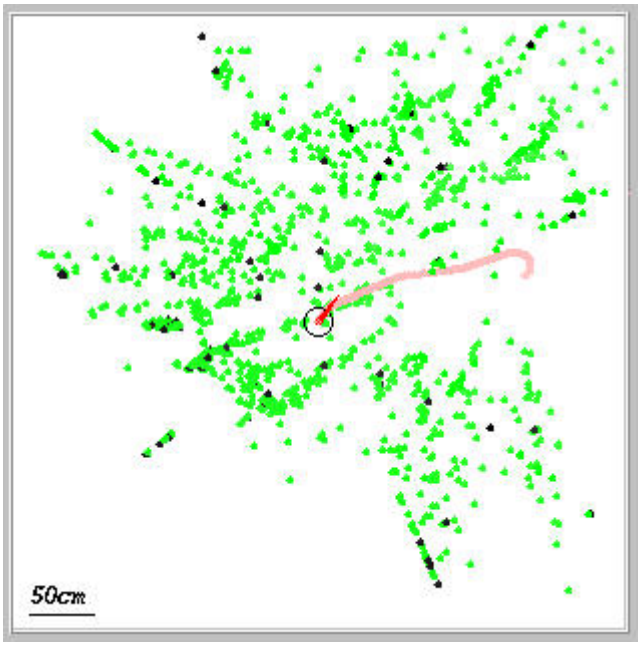

figure 5.7 – affichage des données des capteurs non filtrés

Une autre fonctionnalité principale de la carte dynamique est de tenir compte de l'imprécision de l'odométrie du robot, et ce en décrémentant une variable de certitude contenue dans la structure de donnée de chaque point. Le décrément est proportionnel aux deux mouvements combinés du robot:

- translation
  lorsque le robot avance en ligne droite, l'erreur sur le calcul de la distance parcourue s'accumule ; la différence entre la position réelle du robot et la position calculée par le contrôleur croît donc en permanence.
  De plus, un léger mouvement de rotation indésiré (provoqué notamment par le glissement asymétrique des roues sur le sol) peut également induire le robot en erreur quant à sa position réelle dans le monde.
- rotation
  le calcul de l'angle parcouru lors d'une rotation, particulièrement dans le cas de Pioneer qui ne dispose pas de roues orientables, peut être entaché d'erreurs considérables, avec des conséquences dramatiques sur l'orientation du robot dans la carte dynamique.

Ainsi, à chaque remise à jour des valeurs retournées par les sonars, la carte dynamique calcule la distance parcourue par le robot et son changement de cap, les multiplie par des facteurs définis empiriquement, et décrémente la variable de certitude de chaque point contenu dans le monde virtuel. A l'écran, l'intensité des points affichés est proportionnelle à leur certitude d'être à cet endroit.

D‘autre points critiques ont déterminé l‘implémentation de la carte dynamique. Les capacités du JAVA et les limitations du Web ont mené aux choix suivants :

- pour éviter des flashs à l’écran lors du rafraîchissement de la carte, une technique de *double buffering* est utilisée lors de l’affichage
- pour ne pas ralentir excessivement le système avec des routines d’affichage, la carte n’est pas rafraîchie à chaque fois que l’état du robot virtuel est mis à jour, mais après 3 de ces itérations ; la fréquence d’affichage tombe ainsi à 2 Hz, laissant plus de temps libre pour les calculs de position et le traitement des entrées de l’utilisateur
- afin d’éviter une saturation de la mémoire, les points de la carte sont contenus dans une liste dynamique (un *Vector* [4]) dont la taille est ajustée au fur et à mesure que les éléments sont ajoutés ou retirés de la carte

5.5. L’image

L’objet image est implémenté comme une extension de la classe JAVA standards *Canvas*. Quand une image doit être chargée, une nouvelle thread JAVA est créée puis exécutée. Afin de limiter au maximum le temps de transfert, la méthode `imageUpdate`(), responsable de l’affichage synchronisé des images, a été réécrite avec un contrôle d’erreur minimal.

La nouvelle version de `imageUpdate()` est donnée ci-après :

```
// override the imageUpdate method
public boolean imageUpdate(Image image, int i, int j, int k, int l, int m)
{
  // if we received all the bits from image download thread
  if ((l & 32) != 0){
    this.image = image;
    repaint();
    if(imageSave == true) controller.map.addImage(image);
    imageSave = false;
  }
  else if ((l & 64) != 0)
    controller.dialogBox("error downloading image");
  else if ((l & 128) != 0)
    controller.dialogBox("image downloading aborted");
  return true;
}
```

L’implantation d’éléments graphiques sur l’image elle-même est faite au moment de l’affichage, laissant l’image libre de toute modification. Il est ainsi possible de retravailler l’affichage d’une même image à l’infini ou de la stocker sans altérations.

Les éléments superposés à l’image brute sont de trois nature différente :
- information d’angle de caméra
- marqueur d’obstacles
- marqueurs d’images stockées

Ces éléments graphiques sont ajoutés en utilisant les paramètres du robot virtuel.

6. perspectives futures

6.1. améliorations du hardware

Plusieurs améliorations du hardware auraient de grandes conséquences sur la fiabilité et l'efficacité de l'interface. Que ces modifications soient réalisables ou non, elles sont toutes représentatives des limitations qu'un système téléopéré par Internet doit gérer.

traitement des données

L'élément déterminant des performances de l'interface est la rapidité du transfert et du traitement de l'information. Si le transfert dépend uniquement des performances du Web, le traitement de cette information dépend des capacité de l'ordinateur considéré, JAVA ne générant pas de programmes rapides pour des raisons de compatibilité. Une machine puissante est donc nécessaire.

Une autre solution serait d'augmenter la fluidité du transfert d'image en ayant recours à des algorithmes de compression plus poussés, avec bien sûr des pertes de qualités proportionnelles (utilisation du noir/blanc, filtrage préalable des hautes fréquences, …)

amélioration des capteurs embarqués

Il serait possible de renforcer la sécurité du système téléopéré en ajoutant quelques capteurs simples à bord du robot distant. Leur rôle serait d'intercepter des commandes lorsque la situation l'impose. Il s'agirait par exemple d'ajouter :

- un capteur d'assiette
  un tel capteur donnant l'inclinaison du robot dans le plan (tangage et roulis) permettrait au robot de ne pas s'engager dans des pentes trop raides, ou de s'arrêter lorsqu'il grimpe sur un obstacle trop haut pour lui que l'utilisateur aurait mal apprécié.
- un capteur de force sur les axes de roue
  en montant un capteur de force ou de moment sur les axes des roues, le robot serait à même de détecter lorsqu'une roue est bloquée ou tourne dans le vide. Il pourrait alors corriger son odométrie en conséquence, d'où une bien meilleure précision de la navigation.

De même, alors que les sonars sont d'une efficacité toute relative, il existe des capteurs de distance (certes bien plus chers!) d'une grande précision. En montant deux lasers RangeFinders sur le robot téléopéré, il deviendrait possible de créer une carte beaucoup plus détaillée de l'environnement du robot sans même avoir à le faire se déplacer (le laser RangeFinder est continu sur 180°). L'opérateur pourrait ainsi choisir une destination sur la carte de manière plus sure et plus efficace.

6.2. amélioration de l'interface courante

Une première amélioration de l'interface consisterait à améliorer l'aspect graphique de la GUI de l'applet JAVA. Il s'agirait essentiellement de :

- remplacer les boutons JAVA standards utilisés dans la version actuelle par des images plus expressives et d'une taille plus appropriées à un écran tactile
- redisposer les composantes sur l'écran de manière plus intuitive

Une autre modification concernerait la carte dynamique. Il pourrait être utile de pratiquer une interpolation linéaire intelligente des points de la carte afin d'extraire les droites (typiquement les parois des pièces et des couloirs). Cette fonctionnalité devrait être relativement simple à implémenter sur la structure de donnée choisie pour l'interface.

Dans le même ordre d'idée, et afin de réduire le temps de calcul et la taille mémoire occupée par le programme, il pourrait être intéressant de ne pas considérer que des points comme composantes du monde du robot, mais d'introduire quelques objets simples (lignes, rectangles, cercles, …) Chaque nouveau point retourné par un capteur de distance serait soit intégré à un objet existant en en modifiant les propriétés, soit créerait une nouvelle entité.

L'utilisation d'un langage orienté objet comme JAVA pour implémenter ces formes élémentaires simplifie grandement le concept. En intégrant les méthodes de mise à jour dans l'implémentation des objets, il devient possible d'optimiser l'algorithme de rafraîchissement de la carte. De plus, au lieu d'une multitude de points, la liste dynamique regroupant les éléments à afficher ne contiendrait plus que des objets génériques, rapidement dessinable par des routines JAVA existantes, et simples à stocker.

6.3. développement futurs de l'interface

Un concept intéressant lié à la carte dynamique consisterait à permettre à l'utilisateur de dessiner à l'écran, complétant ainsi la carte générée automatiquement. En liant entre eux des segments de taille ajustables, il deviendrait ainsi possible de construire un plan statique exact des lieux superposé aux éléments dynamiques détectés par les capteurs.

La carte dynamique agirait ainsi comme aide active au dessin d'un plan de situation. A tout moment, l'utilisateur pourrait redimensionner un segment ou réorienter la carte statique par rapport à la position réelle du robot, en s'aidant des données dynamiques et des images de la caméra embarquées.

Le repositionnement de la carte statique pourrait également se faire de manière semi-automatique dans le cas où le robot est suffisamment certain de sa position et/ou reconnaît un élément caractéristique de son environnement. Seulement dans la cas où l'odométrie rencontrerait un problème inattendu qui plongerait le moteur de réalignement dans l'embarras, l'utilisateur serait appeler à réaligner les deux cartes.

7. Conclusion

Ce projet avait pour but la mise au point d'une interface de commande d'un robot mobile à travers l'Internet, en tenant compte des différences sensibles opposant une commande directe à une commande par le Web. Les différences majeures se situent notamment au niveau de la bande passante et du délai inconstant de transfert de l'information.

Ce projet a permis de vérifier qu'il était possible de construire un mode de commande propre à Internet, en limitant au maximum le recours aux sources d'informations lourdes (image vidéo continues, contrôle de type joystick, …) et en combinant au mieux les sources d'information limitées utilisables par le biais du Web. Plusieurs outils de construction de carte et de fusion de capteurs ont été développés dans ce sens.

Bien que cette première version de l'interface utilisateur avancée n'offre pas toutes les fonctionnalités d'une interface classique, elle forme néanmoins une plate-forme solide pour toute implémentation future d'une interface basée sur Internet, et est à même d'intégrer facilement des sources d'informations nouvelles (capteurs embarqués, aide extérieure à la navigation, …) afin de rendre la téléopération sur Internet plus fiable et plus accessible.

Remerciements

Je tiens à remercier Reymond Clavel, Charles Baur et Terry Fong, qui m'ont permis de continuer ma découverte du monde de la robotique mobile.

Un merci sous forme de clin d'œil au Dr. Illah Nourbakhsh de Carnegie Mellon University pour m'avoir introduit à la robotique mobile autonome.

Finalement, mes plus chaleureux remerciements à Sébastien Follonier, Sébastien Dessimoz, Laetitia Michaud, Catherine Pernet et surtout Jane Fallwell pour leurs conseils avisés durant l'élaboration et les tests de l'interface graphique, ainsi que leur patience à mon égard.

Lausanne, le 27 janvier 1999 Sébastien Grange

Bibliographie

[1] McGovern, Douglas, E. 1990 : ‘‘Experiences and Results in Teleoperation of Land Vehicles‘‘, Sandia Report SAND87-0646, Sandia National Laboratories, Albuquerque, NM

[2] Jenny S. Kay, 1997 : ‘’STRIPE Remote Driving Using Limited Image Data’’, CS Thesis [CMU-CS-97-100].

[3] Shishir Gundavaram, 1996 : ‘‘CGI Programming on the World Wide Web‘‘, O‘Reilly & Associates, Inc.

[4] Peter van der Linden, 1998 : ‘‘just JAVA 1.1 and beyond‘‘, 3rd edition, the Sunsoft Press Java Series, Sun Microsystems Press, Palo Alto.

[5] Konolige, K. and Myers, K. 1997 : ‘‘The Saphira Architecture for Autonomous Mobile Robots‘‘, in Artificial Intelligence and Mobile Robots (Bonasso, R. and Murphy, R. eds.) MIT Press, Cambridge.

[6] O. Michel, P. Saucy and F. Mondada, 1997 : "KhepOnTheWeb : An Experimental Demonstrator in Telerobotics and Virtual Reality“, Proceedings of the International Conference on Virtual Systems and Multimedia (VSMM'97), IEEE Computer Society Press.

[7] W.Richard Stevens, 1995 : ‘‘Advanced Programming in the UNIX Environnment‘‘, Addison-Wesley Professional Computing Series.

Glossaire

applet	application écrite en JAVA [4] et destinée à être utilisée sur Internet ; les applets sont différentes des applications JAVA standard, notamment quant aux restrictions de sécurités qu'imposent l'utilisation on-line.
behaviour	en français *comportement* ; dans le cas du contrôleur de Saphira [5], module de programmation définissant un but et un moyen de l'atteindre pour le robot. Plusieurs behaviours peuvent être appliqués simultanément et donc de manière complémentaires ou compétitives.
browser	en français *navigateur*, application de téléchargement et d'affichage des documents HTML. Dans le cas d'un browser compatible JAVA 1.1 (Netscape 4.x, Internet Explorer 4), permet également l'exécution d'*applets* JAVA.
CGI-BIN	abréviation de Common Gateway Interface Binary (exécutables interfacés par une passerelle commune). Programmes exécutables par le biais d'Internet. Les entrées de l'exécutables sont fournies par le *browser* en respectant un protocole défini, et les sorties sont redirigées vers le *browser*.
commande directe	se dit d'un contrôle direct exercé sur une grandeur physique (p. ex. position, vitesse, …) d'un système. Tourner un volant est une commande directe. Définir un point de destination n'est pas une commande directe.
contrôleur	en robotique, logiciel en charge de l'intégration des données de navigation du robot (voir *vecteur d'état*), ainsi que de l'interprétation et de l'exécution des commandes par le robot.
double buffering	technique d'affichage consistant à dessiner une image d'abord dans une zone mémoire avant de l'afficher à l'écran, et ce pour éviter le scintillement.
frame	en français *volet* : panneau d'affichage composant un document HTML et contenant une page HTML.
GUI	Graphic User Interface : interface graphique entre un utilisateur humain et un logiciel
instance	élément composant une classe d'un langage *orienté objet*
java	langage de programmation orienté objet et indépendant du système d'exploitation utilisé (cf. [4]).
mémoire partagée	zone mémoire partagée par deux programmes distincts dans le but d'échanger des informations de manière rapide
méthode	dans les langages *orientés objets*, se dit des fonctions implémentées par un objet (par opposition aux variables)
monde	en robotique, l'environnement dans lequel évolue un robot et qui lui est perceptible.
orienté objet	concept de programmation définissant des objets ; chaque objet contient des *instances*, pouvant être des variables ou des *méthodes*

socket	élément virtuel de communication entre programme à travers des couches de réseau – typiquement socket UDP (datagrammes) ou *TCP/IP* (Internet)
système d'exploitation	couche d'interface entre un logiciel et le hardware d'un ordinateur ; parmi les plus célèbres : Unix, Microsoft Windows, MacOS, …
TCP/IP	Transfer Control Protocol / Internet Protocol : protocole de communication utilisé par les réseaux de type Internet
ultrasonique	capteur de distance à ultrason : une onde sonore est émise par une membrane vibrante, puis réceptionnée par la même membrane après réflexion sur un objet. Le temps de vol de l'onde détermine la distance à l'obstacle.
vecteur d'état	la structure de donnée qui contient les paramètres d'état du robot (typiquement : position, vitesses, valeur des capteurs, …)
virtual machine	logiciel qui interface un programme JAVA (indépendant du *système d'exploitation*) avec le système d'exploitation de la machine concernée

ANNEXE A – interfaces des objets JAVA

le robot virtuel

```
// constructeur
protected Robot(String hostname,int port, navPanel controller)

// gestion de la connection TCP/IP
public boolean connected()
public int openSocket()
public int closeSocket()
public int sendMessage(String str)
public String getMessage()
public int refresh()

// parseur de données
public void update()
public boolean ready()

// accès au vecteur d'état
public int getX()
public int getY()
public int getSpeed()
public int getRotVel()
public int getHeading()
public int getCamHeading()
public int getCamAngle()
public int[] getSonar()
public int getSonar(int i)
public int getObstacleAngle()

// configuration du robot
public void setTimeout(int timeout)
public void setSpeed(int speed)
public void setRotVel(int rotVel)

// commandes de mouvement
public void stopMotion()
public void turnAngle(int angle)
public void turnTo(int angle)
public void velTurn(int vel)
public void moveDist(int dist)
public void moveTo(int x, int y)
public void moveTo(int point[])
public void undo()
public void velMove(int vel)

// commandes de caméra
public void camPanTilt(int pan, int tilt)
public void camPan(int pan)
public void camTilt(int tilt)
public void setCamZoom(int zoom)
```

la carte dynamique

```
// constructor
public Map( int w, int h,
            int sensorAngle[], int sensorMinRange[],
            int sensorMaxRange[],
            navPanel controller)

// geometric stuff
public void initDoubleBuffer()
public Dimension getMinimumSize()
public Dimension getPreferredSize()

// map display
public void paint(Graphics g)
public void update(int x, int y, int speed, int hdg,
                   int sonar[], int sonarX[], int sonarY)
public void updatePointList(point newPoint)
public void clearMap()
public void addImage(Image image)
private void createTarget(int px, int py)
public void removeTarget()
public void zoomIn()
public void zoomOut()
private int xCoord(point p)
private int yCoord(point p)

// unused yet
public void rotateMap(Vector pointList, int angle)
public void rotate(point current, int angle)
public void setHeading(int hdg)
```

le gestionnaire d‘image

```
// constructor
public ImageCanvas(int w, int h, navPanel controller)

// geometric stuff
public Dimension getMinimumSize()
public Dimension getPreferredSize()

// display method
public void paint(Graphics g)
public boolean imageUpdate(Image image,
                           int i, int j, int k, int l, int m)

public void update(Graphics g)
public void issueCommand()

// image grabbers
public void grabImage()
public void grabImage(boolean save)
public void grabImage(int angle)

// image processing
public void setMarkerAngle(int angle)
public void noMarker()
public void addReplay()
public void displayImage(Image image)
public void removeImage()
```

l'indicateur de proximité

```
// constructor
public WarningLight(int w, int h)

// geometric stuff
public Dimension getMinimumSize()
public Dimension getPreferredSize()

// display
public void paint(Graphics g)
public void update(Graphics g)
public void setIntensity(int value)
```

ANNEXE B - protocoles

Cette section décrit les protocoles de communication utilisés entre l'applet et ROBOGATE.

applet vers ROBOGATE

commande	*arguments*	*description*
"TIMEOUT "	timeout	définit le timeout du robot
"SPEED "	speed	définit la vitesse max du robot
"ROTVEL "	rotVel	définit la vitesse de rotation max du robot
"STOP"		arrête toute action du robot
"TURNANGLE "	angle	rotation de [angle] degrés
"TURNTO "	cap	rotation au cap [cap]
"VELTURN "	vel	rotation à vitesse [vel] constante
"MOVEDIST "	dist	translation de [dist] mm
"MOVETO "	x,y	translation au point [x,y]
"VELMOVE "	vel	translation à vitesse [vel] constante
"CAMERA "	pan+tilt	oriente la caméra horizontalement de [pan] degrés verticalement de [tilt] degrés
"CAMPAN "	pan	oriente la caméra horizontalement de [pan] degrés
"CAMTILT "	tilt	oriente la caméra verticalement de [tilt] degrés
"ZOOM "	zoom	ajuste le zoom de la caméra à [zoom]

tableau B.1 – protocole I

ROBOGATE vers applet

commande	*arguments*	*description*
"GETIMAGE "		demande à l'applet de télécharger une nouvelle image
"STATUS "	statusString	met à jour l'indicateur de status avec la valeur [statusString]
"WARNING "	message	demande à l'applet d'ouvrir une boîte de dialogue avec le texte [message]
default	stateVector[]	par défaut, envoie un tableau de valeurs entières représentant le vecteur d'état du robot

tableau B.2 – protocole II

ANNEXE C - configuration matérielle

caméra embarquée

Characteristics	**Sony EVI-D30**
Image Sensor	1/3"IT Color CCD
Video Signal	NTSC
Effective Pixels	768(H)X492(V)
H. Resolution	460TV Fines
V. Resolution	350TV Fines
Lens	X12 Power Zoom f=5.4 - 64.8mm F1.8 - F2.7
H. Angle of View	4.4 - 48.8degrees
Shortest Subject - Dist	l0mm(WIDE end), 800mm(TELE end)
Min. Illumination	71x(F1.8)
Illumination Range	7-100,0001X
Auto Exposure	Auto Iris, AGC
Shutter Speed	1/60- 1 / 10,000(VIS CA control)
Gain	Auto/Manual(VISCA control)
White Balance	TTL Auto Tracing/One Push Hold, Indoor Preset, Outdoor Preset(VISCA control)
S/N Ratio	More than 48dB
Pan/Tilt	Horizontal ±100 degree (Max speed 80degree/sec), Vertical ±25 degree (Max speed 50degree/sec)
Video Output	RCA pin jack, 1 Vp-p, 75ohm unbalanced
S Video Output	4 pin mini DIN
Audio Output	RCA pin jack (monaural), Rated output 327mV, Output impedanceless than 2.2 kilohms
Control Terminal	RS232C, 8 pin mini DIN, 9600bps, Data 8 bit, Stop 1 bit
Power Terminal	DC IN 13.5V(EIAJ unified polarity type»
irements	DC 12-14V
Power Consumption	10.5W
Dimensions(W/H/D)	142xl09xl64mm
Mass	1200g

tableau C.1 – caractéristiques de la caméra embarquée

robot mobile

Characteristics	**Pioneer AT "The Outlaw"**
Physical	
Length	45 cm
Width	50 cm
Height (body/antenna)	24 / 44 cm
	11.3 kg
Power	
Batt	12V sealed, lead-acid
Charge	84 watt-hr
Run time	2-3 hrs, hot-swappable batteries
Recharge time	8 hrs
Mobility	
Wheel diameter	16 cm
Wheel width	11 cm
Steering	Differential
Pushing force	8 kg
Swing radius	30 cm
Turn radius	0 cm
Translate speed max	1.5 m/sec
Rotational speed max	360 deg/sec
Traversable step max	8.9 cm
Traversable gap max	12.7 cm
Traversable slope max	80% grade
Traversable terrains	All surfaces
Sensors	
Ultrasonic sonars	1 each side 5 forward at 15 deg intervals (7 total)
Position encoders	100 ticks per rev
Serial Port	1 serial port 1 Mercury RF1 Radio Modem

tableau C.2 – caractéristiques du robot mobile

radio modem

Characteristics	**Mercury - RF1 Radio Modem**
Physical	
Imperial Dimensions	5.5 x 2.7 x .88
Weight	10 Ounces
Metric Dimensions	13.5 x 7.5 x 1.7
Temperature	-20 to 60 deg. C
Humidity	20% to 90% (Non-condensing)
Radio	
	Compatible with Proxim RangeLAN2 Frequency hopping, spread spectrum 15 independent channels
Frequency	2.4-2.4835Ghz
Output power	100 mW
Data rate	1.6Mbits/sec
Modulation	4fsk (bsk in back off mode)
Serial interface	
Data rates	300 to 155K Baud
Data format	7,8 Data; 1,1.5,2 Stop;ENO Parity
Control lines	TXD, RXD, RTS, CTS, DSR, DTR @RS232 Levels
Connector	Female DB9
Configure and reset control lines	
Pinout:	1-Configure, 2-TX, 3-RX, 4-DTR, 5-GND, 6-DSR, 7-RTS, 8-CTS, 9-Reset
Power	
Input voltage	6-15VDC
Input current	675 mA RX typical 850 mA TX typical
Standby mode	< 40mA
Sleep mode	<10mA
Power connector	2.5 mm DC power jack

tableau C.3 – caractéristiques du radio modem embarqué

vidéo transmetteur

Characteristics	**Dell Star DSX-15249 TXA**
Electrical	
Frequency range	902MHz - 928MHz (Single Channel)
Frequency stability	+/-0.001%, Crystal Accuracy-Tuned PLL
RF power output	0.5mW (-3.0dBm, +/-1.0dBm)
Field strength	*50Mv / meter @ 3 Meters
Modulation	Frequency Modulation (True FM)
Video freq. response	30Hz to 4.2MHz @ -3dB points
Deviation	+/- 6.0MHz
Bandwidth	FCC Emission Code 17M5F3W
Video input voltage	1 volt P-P (neg. sync.)
Video input impedance	75 Ohms Nominal
Audio freq. response	200Hz TO 35 KHz @ -3dB points
RF output impedance	50 Ohms
Audio input impedance	10K Ohms (1v p-p Line Level)
Pre-emphasis	Per CCIR-405
Input voltage	11 vdc to 30 vdc
Reverse polarity	Protected
Input current	<85mA
Physical	
Case	Machined Aluminum, Anodized (HARDCOAT), Black
Dimensions	3..5" x 2.0" x 0.72"
Weight	<0.28 lbs. (4.5 oz.)
Connectors	Power input LEMO (9MM) Video input BNC Female Audio input BNC Female *RF input SMA Female

tableau C.4 – caractéristiques du transmetteur vidéo embarqué

vidéo récepteur

Characteristics	**Dell Star DSX-15249 TXA**
Electrical	
Frequency range	902MHz - 928MHz
Frequency select	User Specified - Factory Set
Frequency stability	+/-0.001%, Crystalized Accuracy-Tuned PLL
Noise figure	<3dB
Sensitivity	-80dBm @ 10dB SNR
IF bandwidth	12 MHz
Video frequency response	30Hz to 4.2MHz @ -3dB points
RF input impedance	50 Ohms (VSWR < 2.00)
Video output impedance	75 Ohms Nominal
Audio output impedance	600 Ohms (Line Level)
Audio frequency response	200Hz TO 35KHz @ -3dB points
De-emphasis	Per CCIR-405
Input voltage	11 vdc to 30 vdc
Input current	<200mA
Physical	
Case	Machined Aluminum, Anodized (HARDCOAT), Black
Dimensions	3.00" x 4.25" x 0.75"
Weight	<0.5 lbs.
Connectors	Power input LEMO (9MM) Video input BNC Female Audio input BNC Female RF ouptut SMA Female

tableau C.5 – caractéristiques du récepteur vidéo