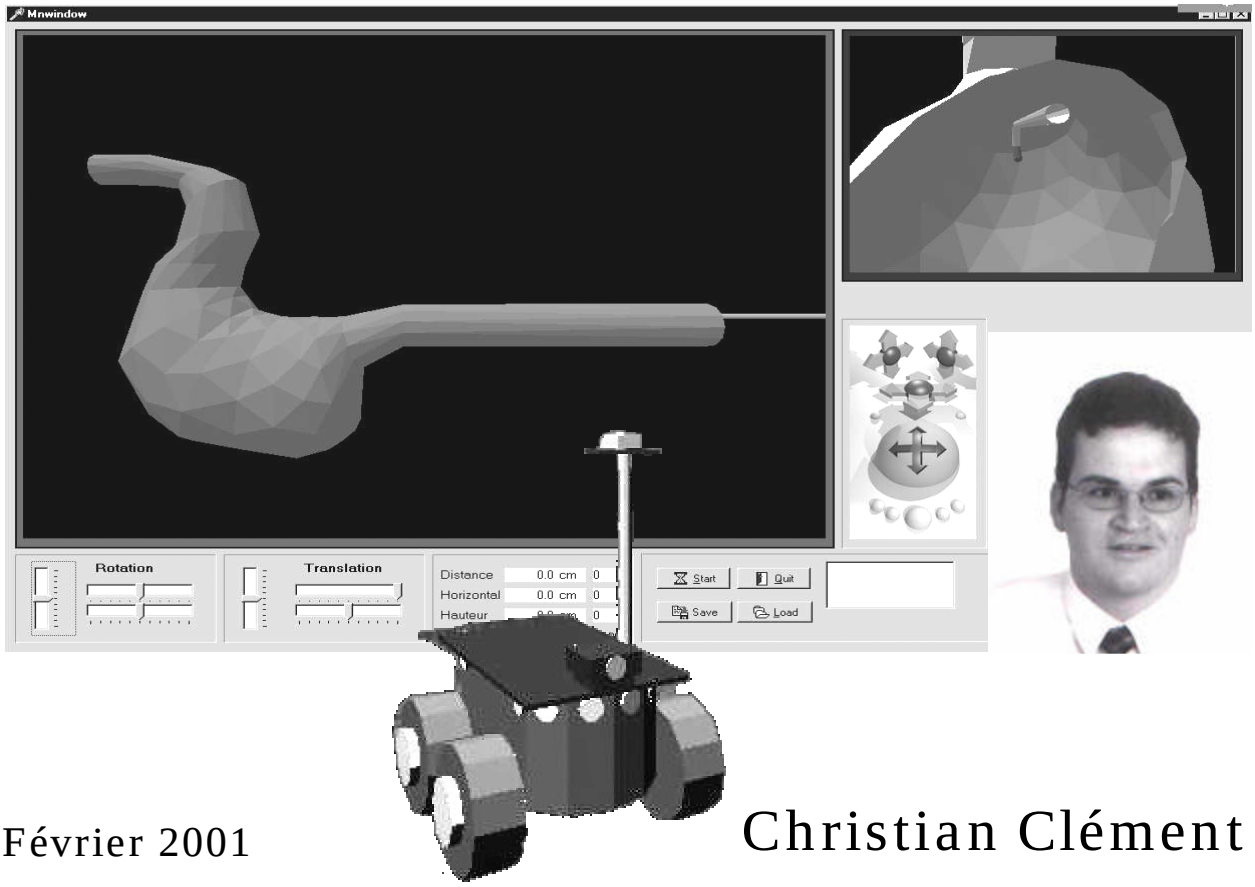
Vrai Group
Institut des Systèmes Robotiques
Département de microtechnique

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Projet de diplôme

Création de mondes virtuels par système de vision



Février 2001

Christian Clément

Assistants
C. Baur, S. Grange, F. Conti

Professeur
Reymond Clavel

Création de mondes virtuels par système de vision

Christian Clément, microtechnique

Assistant(s): *Charles Baur, Sébastien Grange, François Conti*

Professeur: *Reymond Clavel*

Introduction

Avec l'augmentation de la puissance de calcul des machines actuelles, la création de mondes virtuels s'ouvre sur de nouveaux horizons. Des mondes virtuels, on passe aux mondes virtualisés très proches de la réalité. De l'autre côté, cette puissance permet également de progresser dans le domaine de la vision tri-dimentionnelle.

Des travaux précédents ont permis le développement d'un moteur 3D simplifiant la création de mondes virtuels. Un autre projet a proposé des librairies utilisant un système stéréoscopique couleur afin de localiser des objets dans l'espace.

Objectif

Le but de ce projet est la création de mondes virtuels à l'aide de systèmes de vision.

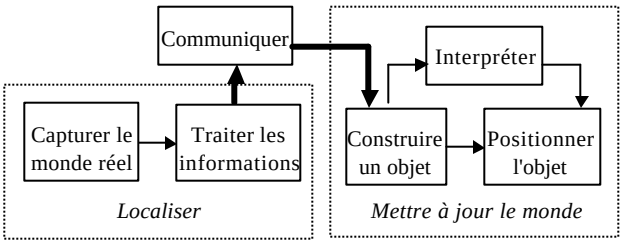


Fig. 1 : Fonctions de l'application

Ce travail (Fig.1) propose de suivre dans l'espace des objets connus à priori, d'en déterminer la localisation et l'orientation, puis d'envoyer ces données au réseau. Côté virtuel, on profite des connaissances à priori de l'objet afin d'interpréter les données pour rendre le mouvement de l'objet virtuel plus réaliste.

L'objet tracké est un marqueur à partir duquel on interprète la localisation et l'orientation.

Résultats

Plusieurs fonctions ont été mises au point afin de permettre la détection d'orientation sur la base d'un multi-tracking de notre marqueur.

Une autre partie du développement propose une librairie interfaçant les deux mondes.

Enfin, plusieurs outils ont été ajoutés dans le monde virtuel afin de rendre le comportement des objets mobiles plus réels : la détection de collision et le blocage de degrés de liberté.

A partir de là, deux exemples démontrant les capacités du système sont proposés :

La localisation d'un robot

Fig. 2: Robot autonome

Le marqueur est placé sur un robot autonome (Fig.2) afin de permettre sa localisation.

Le monde virtuel, après avoir reçu puis interprété les informations, recrée un robot virtuel dans une scène avec l'image réelle comme fond. Le robot peut être remplacé par tout autre objet.

Le simulateur de gastroscopie

Fig. 3: Organe virtuel

Afin de pouvoir exercer les mouvements à effectuer avec un gastroscope, le marqueur peut être manipulé comme le serait un véritable instrument. Le système est simplifié, mais démontre les possibilités futures d'un véritable simulateur (Fig. 3) ou d'une aide lors de l'opération réelle.

1. Introduction

1.1. Situation actuelle

Aujourd'hui, la création de mondes virtuels devient une composante incontournable dans de nombreuses applications. Avec l'augmentation de la puissance de calcul disponible, il est permis de créer des systèmes de plus en plus complexes. Cette complexité est inévitable si l'on désire restituer au mieux la réalité et baigner l'utilisateur dans cet environnement virtuel. L'intérêt peut être ludique, l'utilisateur se trouvant au cœur de l'aventure. Mais il est aussi scientifique, par exemple dans l'objectif de simulation, de téléopération ou de représentation 3D mieux adaptée aux capacités ou aux sens de l'utilisateur. On parle alors souvent de virtualisation de la réalité.

Pourquoi vouloir virtualiser une scène réelle ou autrement dit, en obtenir une vision adaptée ? Les scènes virtuelles offrent de nombreux avantages, à commencer par la position de la caméra : dans le monde virtuel, on peut la placer n'importe où et ainsi proposer le meilleur angle (et même plusieurs angles à la fois) à celui qui regarde. Ensuite, il y a la meilleure connaissance du contenu : les objets virtuels, leurs dimensions et positions sont parfaitement connus et ainsi, leur comportement peut être contrôlé. On peut également représenter des événements que nos sens ne pourraient sinon appréhender.

Parallèlement, parmi les méthodes fournissant des informations sur le monde réel, les systèmes de visions ont un potentiel croissant. Autrefois, la puissance de calcul nécessaire en limitait l'usage. Certains travaux ont d'ailleurs proposé [FLU94] des solutions, mais des solutions se limitant à un espace plan. Aujourd'hui, l'analyse d'images peut se faire en temps réel et dans l'espace tridimensionnel. C'est ce type de système qui est choisi et nous verrons dans ce travail ses avantages comme ses limitations.

Le but de ce projet est de réaliser une solution de ce type dans deux contextes différents. Le premier est le simple suivi du mouvement d'un objet et la mise à jour de son double dans le monde virtuel. Le second est le suivi d'un mouvement, mais avec une interprétation de celui-ci afin de mettre à jour un objet virtuel différent.

Nous souhaitons tirer les informations du monde réel pour mettre à jour son équivalent virtuel ou virtualisé. Toutefois, les procédés diffèrent totalement suivant le but recherché. Dans le cas qui nous occupe, nous ne cherchons pas à reconstruire un objet inconnu. Nous posons comme base que l'objet est connu, comme dans beaucoup d'applications, et souhaitons par contre le voir évoluer dans l'environnement virtuel en temps réel.

1.2. L'objectif

Notre but est de proposer un système complet de mise à jour de monde virtuel utilisant la vision comme source d'information. Le monde réel étant connu à priori, on utilise cette information pour créer l'interprétation. Ainsi, il s'agit de transcrire en temps réel le déplacement d'un objet connu à priori dans

un environnement virtuel. La commande de déplacement se fait par l'observation d'une scène réelle et l'extraction des informations, suivies par l'interprétation de ces données acquises.

Du moment que l'objet virtuel créé est connu à priori, on connaît aussi ses caractéristiques usuelles de déplacement ou son comportement. L'utilisation de ces informations permet plus de réalisme, élimine une partie des erreurs provenant de la capture du monde réelle et permet d'utiliser des procédés d'acquisitions simplifiés.

Au niveau de la capture du monde réel, l'acquisition doit permettre un suivi en temps réel. Que ce soit pour localiser une cible dont on reconstruira son image virtuelle ou pour suivre le mouvement d'un marqueur qu'on interprétera comme un objet totalement différent, le dispositif doit pouvoir s'adapter au milieu d'acquisition. De la sorte, le champ d'applications est très ouvert.

Une fois les mouvements détectés, ils devront être interprétés et éventuellement corrigés avant d'être traduits en leurs homologues virtuels. L'interprétation permet plusieurs voies. D'un côté, on peut s'en servir pour recréer en virtuel le même objet que celui qui est suivi dans la scène réelle (exemple : on suit le déplacement d'un robot mobile et on crée son jumeau dans le monde virtuel). D'un autre côté, il y a aussi la possibilité de filtrer le déplacement et de l'utiliser pour faire évoluer un objet virtuel différent de celui qui a été traqué (exemple : la poursuite d'une cible conditionne les mouvements d'un bras de robot).

1.3. Les réponses apportées

Dans la scène réelle, afin d'obtenir la localisation selon les 6 degrés de liberté, le choix s'est posé sur un système de vision stéréoscopique couleur. Les travaux précédents [GRA00] ont démontré que ce système avait une très grande flexibilité, autant au niveau de l'objet traqué que du lieu d'utilisation. Avec ce genre de dispositif, les données concernant le plan de la caméra sont très précises. Mais pour la profondeur, elles le sont nettement moins. Ce n'est pas un problème, car les connaissances à priori de l'objet permettent de corriger une partie des erreurs.

Un marqueur composé de quatre sphères a été choisi pour simplifier la détection d'orientation. La localisation de ces sphères détermine les six degrés de liberté du marqueur. Celui-ci peut être posé sur un objet ou directement manipulé. Ce système a l'avantage de limiter la puissance de calcul nécessaire et de ne pas demander la mise au point de librairies complexes pour assurer la détermination de l'orientation. Par contre, on restreint le domaine d'applications d'utilisation. En cas de nécessité, le tracking peut toujours être adapté à un objet spécifique sans affecter le reste du système.

Côté environnement virtuel, des travaux précédents [CON99] ont mis au point un moteur 3D permettant la création aisée de mondes virtuels. Mais des compléments ont dû être apportés comme, par exemple, la détection de collision entre deux objets. Cet aspect fait également partie de ce travail.

Deux connaissances à priori nous permettent d'améliorer le système. La connaissance à priori du marqueur permet d'améliorer le tracking. Certaines localisations fantaisistes sont éliminées si le marqueur détecté a des dimensions non conformes. Ensuite, la connaissance de l'objet virtuel et de l'environnement dans lequel il est placé permet l'interprétation du déplacement. On peut ainsi exclure les collisions ou limiter des mouvements dans certains plans ou directions. Par exemple, on force un robot mobile à évoluer dans le plan défini par le sol.

2. Cahier des charges

2.1. Le cadre

La mise à jour de mondes virtuels peut s'effectuer de nombreuses manières. La piste suivie est de localiser une cible et de mettre à jour l'environnement tout en tenant compte des caractéristiques comportementales de l'objet virtuel lié à la cible.

Une fois le mouvement de la cible identifié, on peut soit utiliser ces informations pour animer son équivalent virtuel, soit les interpréter pour animer un autre objet totalement différent. Le monde virtuel peut représenter un miroir de la réalité traquée ou une interprétation se servant uniquement des indications de mouvement.

L'unité d'acquisition est séparée de celle servant à l'affichage du monde virtuel. Cette dissociation a deux objectifs : différencier géographiquement les deux peut être souhaitable, par exemple, dans le cas de monitoring à distance ou de téléopération. De plus, cela permet également d'optimiser les performances en travaillant simultanément sur deux machines séparées. Pour des raisons de flexibilité, la liaison est une connexion par réseau internet.

2.2. Tâches à mettre en oeuvre

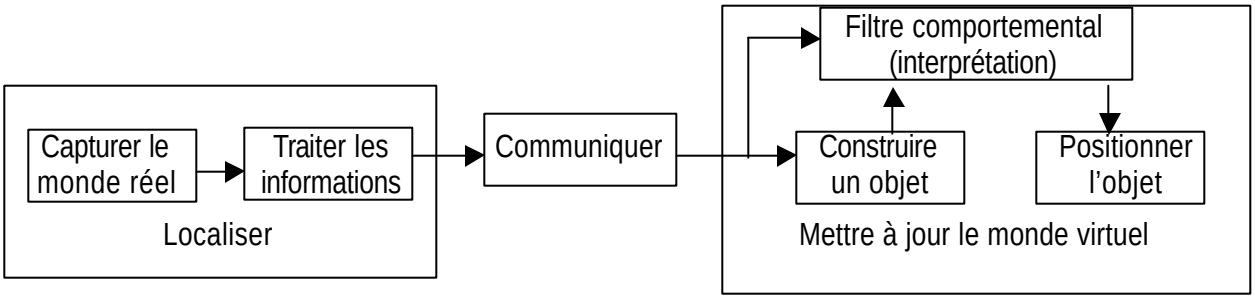


Fig. 1: Distinction des tâches

Comme on peut le voir sur la Fig. 1, les tâches du système à mettre en place sont : localiser, communiquer et mettre à jour le monde virtuel

2.2.1. Localisation

Le moyen proposé pour localiser les objets est un système de vision stéréoscopique couleur. Deux caméras pointant dans la même direction et séparées par une distance connue arrivent à donner une

information sur l'éloignement d'un objet (on peut comparer ce système avec la vision humaine). Il existe ensuite plusieurs méthodes pour extraire la profondeur d'une paire d'images.

Des librairies de tracking ont déjà été mises au point lors d'un travail précédent [GRA00]. Elles se composent d'un ensemble de fonctions écrites en C/C++. Elles permettent de localiser un objet dans l'espace.

Il s'agit d'utiliser au mieux ces librairies, répondant uniquement au problème de localisation, afin de définir les six degrés de liberté de l'objet traqué. La détection d'orientation devant être implémentée, l'utilisation d'un marqueur est suggérée. Sa détection est adaptée aux modalités de tracking, n'est pas trop gourmande en ressources de calcul et ne nécessite pas de développement complexe.

Les connaissances à priori de ce marqueur doivent également permettre de filtrer les informations retournées par le système de tracking. Certaines localisations fantaisistes sont éliminées si le marqueur détecté possède des dimensions non conformes.

2.2.2. Communication

Il existe plusieurs types de protocoles et de formats de communication possibles au travers du réseau Internet. Les informations envoyées concerneront principalement la localisation des objets selon 6 degrés de liberté. On souhaite également pouvoir envoyer des images de la scène filmée. Ainsi, on peut situer précisément le contexte de la scène virtuelle si son image de fond est une photo du monde virtuel.

Concernant le code, il est nécessaire de veiller à la facilité de sa transposition pour des projets futurs.

2.2.3. Réalité virtuelle

Les bases pour la création d'environnements à trois dimensions sont mises à disposition dans une série de fonctions [CON99] simplifiant les accès à OpenGL (plate-forme de bas niveau commune à tous les systèmes d'exploitation pour la programmation en 3D). Tout l'environnement fonctionnant sous Delphi, cette partie devra être programmée dans ce même langage, à savoir le Pascal Objet.

Dans le but de démontrer les possibilités de recréation virtuelle d'une cible traquée ou d'interprétation des mouvements d'un repère, une librairie d'objets avec quelques exemples choisis doit être mise en place.

2.2.4. Interprétation et comportement

Etant donné que les objets virtuels sont connus, certains comportements doivent être implémentés. La non-collision entre objets ou la restriction de certaines libertés de mouvement en font partie. De cette manière, l'interprétation du déplacement d'un objet virtuel semblera plus réelle et la construction du monde virtuel sera sensiblement plus plausible.

2.3. Résultats recherchés

Le but est de proposer un système complet de mise à jour de monde virtuel utilisant la vision comme source d'information. Le monde réel étant à priori connu, on utilise cette connaissance pour corriger et interpréter les données reçues.

Le système proposé doit pouvoir s'adapter avec un minimum d'effort. Que ce soit pour visionner le déplacement d'un robot, pour simuler et tester le passage d'un capteur dans des vaisseaux sanguins ou encore pour faciliter l'enregistrement de mouvements pour une animation de synthèse réaliste, le noyau du système reste identique.

3. Etat de l'art

3.1. Les mondes virtuels

OpenGL est une librairie commune à tous les systèmes d'exploitation qui permet de dessiner à l'écran des objets en trois dimensions. Les fonctions fournies sont d'assez bas niveau et il est laborieux de vouloir les utiliser telles quelles. C'est pourquoi de nombreux développeurs ont mis au point des moteurs 3D simplifiant la création de mondes virtuels.

Ces librairies sont rarement libres de tout usage commercial. Raison pour laquelle un travail précédent [CON99] a permis la mise au point d'un moteur 3D pour le laboratoire. Celui-ci est destiné à être programmé sous Delphi (Pascal orienté objet).

On trouve d'autres auteurs qui ont aussi travaillé dans les mêmes environnements et proposent leurs outils [ALL00], [LIS00] ou [JIA00]. Toutes ces librairies sont certainement plus complètes, mais en raison de la propriété intellectuelle et dans le but de standardiser les travaux du groupe, ne peuvent pas être introduites dans notre application.

3.2. Vision

Lors de ce travail [GRA00], il a été démontré que la vision était adaptée dans de nombreux cas pour obtenir des informations de positionnement dans l'espace. En effet, avec la puissance de calcul qui s'est accrue, des algorithmes de tracking peuvent être utilisés en temps réel. L'avantage est d'être totalement passif et de pouvoir s'adapter facilement à n'importe quel environnement. De plus, la saisie de l'environnement est globale par opposition à un tracker dédié. Les techniques de vision stéréoscopiques utilisées engendrent des limitations quant à la résolution en profondeur du système. Ces limitations et leur impact sont présentées dans le paragraphe 3.2.2.

La localisation est donnée par l'addition de deux techniques : la segmentation en profondeur et la détection de couleurs. Séparément, ces deux techniques n'auraient pas de résultats suffisants, mais leur

addition donne des réponses robustes et consistantes. Leur grand avantage est surtout de ne pas être trop gourmande en ressources et de pouvoir ainsi fonctionner en temps réel.

3.2.1. Analyse des couleurs

La détection de couleur analyse l'image et la filtre suivant une zone de couleur normalisée. Les marges inférieures et supérieures sont données pour le rouge normalisé ainsi que pour le vert. Tous les pixels n'entrant pas dans cette zone de couleur sont éliminés. En situation idéale, il ne reste que notre objet traqué.

$$R_{normalisé}=\frac{R}{R\cdot G\cdot B} \quad G_{normalisé}=\frac{G}{R\cdot G\cdot B}$$ où R, G, B sont les intensités du rouge, vert et bleu

Ainsi, on se libère des variations d'intensité lumineuse en ne conservant que la chrominance. Il reste malgré tout les limitations dues au bruit que pourraient fournir des objets de même couleur situés dans la scène filmée.

Toutefois, analyser à chaque image tous les pixels serait superflu et trop gourmand (fréquence de balayage inférieure à 2 Hz pour notre installation). C'est pourquoi, une fois l'objet localisé, on découpe autour une zone de travail dans laquelle on limitera notre recherche de déplacement (Fig. 2). Ainsi, les performances obtenues sont nettement meilleures (15Hz, pour un multi-tracking simultané de 4 objets).

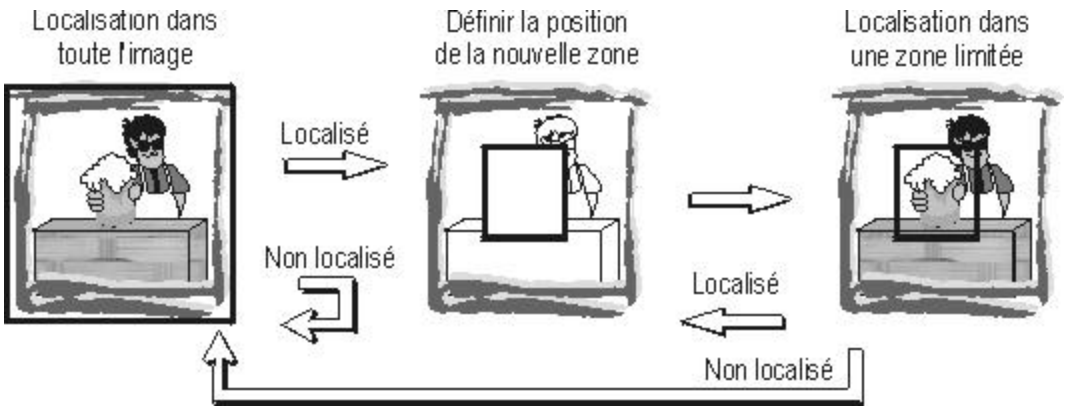


Fig. 2: Etapes dans la localisation d'une image

3.2.2. Stéréo

La stéréo permet tout d'abord la segmentation en profondeur. La Fig. 3 en donne le schéma depuis les deux caméras jusqu'à l'obtention d'une image de disparités filtrée.

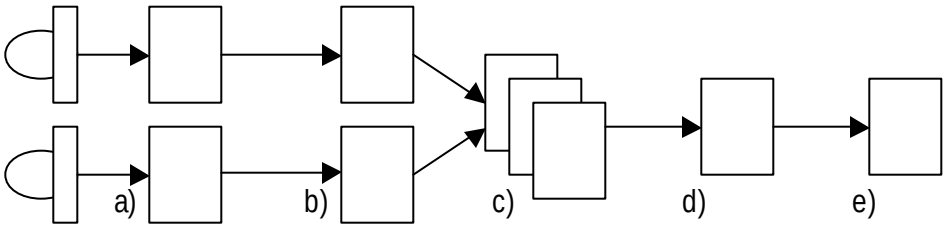


Fig. 3: Système stéréoscopique :
a) Image en niveaux de gris b) Image transformée c) Pattern matching d) Image de disparités e) Im. disparités filtrée

La mesure en profondeur est également effectuée par le système stéréo. Connaissant la distance entre les deux caméras, on retrouve la profondeur grâce à l'analyse des décalages entre deux images prises simultanément.

Tout comme dans la vision humaine, la précision en profondeur varie avec la distance à la caméra (Fig. 4).

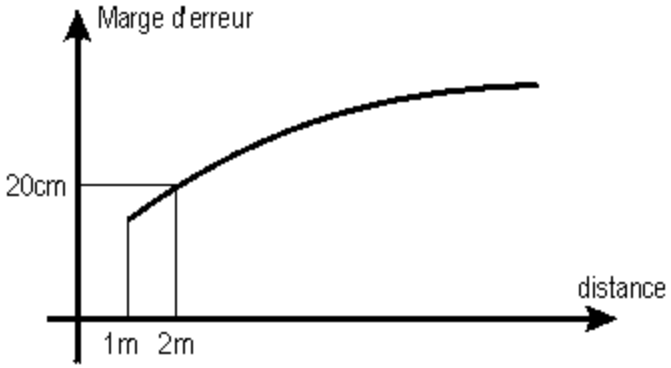


Fig. 4: Marge d'erreur augmentant avec la distance d'éloignement

3.2.3. Dans la même direction

D'autres auteurs ont également recherché les mêmes objectifs, on citera par exemple : [MUL98], [BRA98]. Toutefois, ces études n'ont pas utilisé cette fusion d'informations (couleur + stéréo). Ils ont pris d'autres directions telle que la détection de mouvement entre deux images séparées.

3.3. Virtualisation de la réalité

Recréer en virtuel la copie exacte de ce qui se passe dans une scène réelle pourrait être une des applications de mon projet. Cette fin est déjà atteinte dans les projets suivants : [RAN98], [VED99] et [KAN99].

Fig. 5 : Bulle d'acquisition

Fig. 6: Exemple de résultat obtenu

Pour réaliser cet objectif, la scène est filmée par un ensemble de 27 caméras situées tout autour d'un espace restreint (bulle d'acquisition Fig. 5). Le monde virtuel recréé sans à priori correspond d'une manière assez exacte à la réalité. Les objets ainsi que les déplacements à l'intérieur de la zone sont du vrai modelage 3D. On peut aussi bien recréer un objet simple qu'un basketteur tirant un panier (Fig. 6)

Toutefois, la grande limitation est la puissance de calcul gigantesque nécessaire. Ainsi, il est pour l'instant impossible de parler de temps réel.

La direction choisie pour ce projet diffère cependant de celle présentée ici, puisque nous traitons de représentation en temps réel d'objet connus à priori.

3.4. Robotique, Réalité Virtuelle et Vision

3.4.1. Résultats

Dans ce projet [FLU94], le but est de créer un monde virtuel propre à un robot et à l'environnement dans lequel il évolue (Fig. 7). Et afin de simplifier cette création, on utilise un système de vision reconnaissant certains objets simples et donnant des informations sur leur position. L'environnement virtuel peut également servir pour la téléopération à distance.

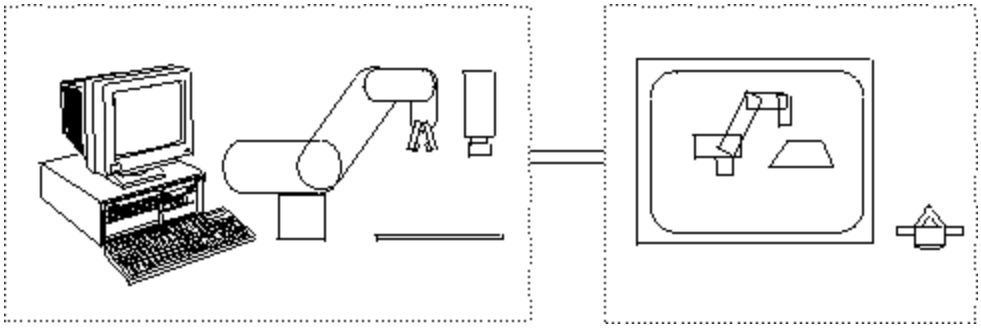


Fig. 7 : Plate-forme du projet Robot, réalité virtuelle et vision

Le système de vision, fixé au-dessus de l'environnement de travail, observe la scène et envoie les images à une station. Celle-ci analyse les images, reconnaît des formes simples (Fig. 8) sur un plan et envoie les commandes de déplacement au contrôleur du robot. De plus, elle envoie également des informations sur le réseau concernant les objets détectés et leur position.

A l'autre bout, une station recevant les informations reconstruit le même monde en virtuel et une souris 3D permet de manipuler le robot virtuel.

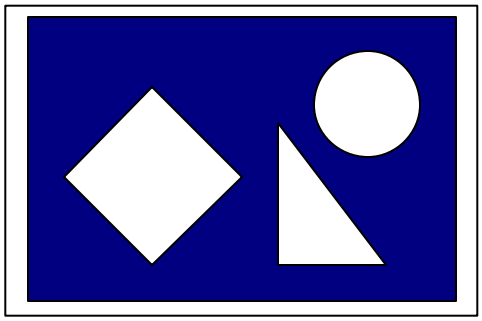


Fig. 8 : Formes typiques détectées

3.4.2. Différences

Dans ce projet, la vision travaille uniquement dans le plan de l'image et donne des informations selon deux dimensions. Ainsi, les objets doivent se présenter selon une face connue et une profondeur définie.

Dans notre projet, il n'y a pas de contrainte tridimensionnelle, nous travaillons selon 3 dimensions. Les objets se présentent en n'importe quelle position et à une distance variable du système de vision. Cette exigence complique nettement la tâche mais offre un champ d'applications beaucoup plus grand. De la même manière, le marqueur peut évoluer dans une scène complexe et inconnue (voire en mouvement), ce que ne permet pas l'application décrite ici.

De plus, on ne trouve pas d'interprétation du mouvement. L'objet qui se déplace dans la scène réelle ne peut que déplacer son double dans le monde virtuel.

4. Description de l'installation

4.1. Vue globale

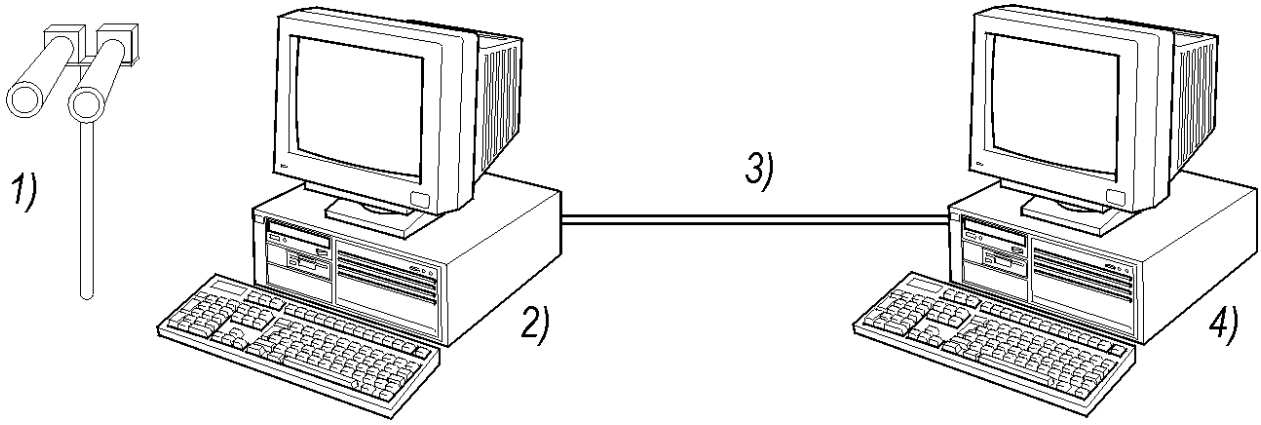


Fig. 9 : Schéma de l'installation avec 1) Caméra stéréo 2) PC pour le tracking 3) Réseau 4) PC créant le monde virtuel

L'acquisition d'informations se fait à l'aide d'un système de vision stéréoscopique couleur (Fig. 9 /1) qui envoie des images vers une station de traitement (Fig. 9 /2). Ce PC analyse les images et cherche les degrés de liberté d'un repère filmé. Ce repère peut être utilisé en mode direct (objet réel et virtuel identiques) ou en mode totalement interprété.

En mode direct, on peut le déposer sur un objet et son analyse donne la position et la rotation de celui-ci (ex : repère posé sur un robot). Ensuite, les caractéristiques connues à priori de l'objet limitent ses mouvements possibles (déplacement d'un robot limité au sol). On corrige ainsi une imprécision du tracking.

En mode totalement représentatif, le repère n'a aucun lien avec l'objet représenté dans le monde virtuel, il en substitue uniquement les mouvements. Mais là aussi, les mouvements réels peuvent différer en raison des caractéristiques de l'objet et de l'environnement (ex. deux boîtes s'insérant l'une dans l'autre).

Les informations sont envoyées sur le réseau Internet (Fig. 9 /3) dont le débit maximal est de 10 Mbits.

Un monde virtuel est créé à l'aide du moteur 3D sur une station dédiée (Fig. 9 /4) et on y insère des objets de base définissant notre environnement : des murs, une porte, une table, les parois d'un intestin, etc… A tous les objets, on ajoute certaines propriétés indiquant leur comportement dans l'environnement ou en interaction avec les autres. Exemple : un mur ne peut être traversé, une porte ne peut subir qu'une rotation, un train suivre son rail, etc…

4.2. Données techniques

- Tracking :

  PC : Pentium III 700 MHz, 256 Mb RAM

  carte d'acquisition : PXC200 (caractéristiques en annexe)

  systême stéréoscopique : Stereographics Corp.

- Monde virtuel

  PC : Pentium III 450 MHz dual processors, 128 Mb RAM

  carte graphique : AccelGalaxy (caractéristiques en annexe)

5. Solutions

5.1. Le tracking d'objets

5.1.1. Les fonctions existantes

La librairie de fonctions disponibles permet de tracker un ou plusieurs objets différents dans une scène. Les informations en sortie sont la localisation dans le plan de la caméra ainsi que l'éloignement de l'objet.

En utilisant la localisation sur plusieurs objets simultanément, on peut définir la position dans l'espace de plusieurs points. Cela permet de retourner l'orientation relative de ceux-ci et de l'objet qu'ils constituent ou auquel ils sont associés.

5.1.2. L'orientation

La première méthode à laquelle on pense pour détecter l'orientation d'un objet quelconque est d'utiliser la stéréo pour recomposer totalement la cible. Les algorithmes existent déjà, mais c'est une méthode gourmande en ressources et il n'est pas envisageable de faire du temps réel [RAN99].

Parmi les autres solutions, on trouve l'analyse des arêtes, sommets ou points caractéristiques. Il existe des algorithmes qui parviennent à cette solution [RAM96]. C'est une direction qui peut être envisagée à l'avenir, mais le travail nécessaire est sort du cadre de ce projet et n'entre pas dans la philosophie suivie au sein du groupe.

La méthode choisie est nettement simplifiée. Comme la localisation d'un objet connu ne pose pas de problème, on cherche en fait à tracker un marqueur composé de quatre sphères différentes. Partant du principe que trois points suffisent à définir les six degrés de liberté d'un objet, les positions de 3

sphères donnent également l'orientation. Que ce soit posé sur un objet mobile ou tenu dans les mains comme commande, tout est caractérisé.

Afin de limiter les erreurs dans la localisation des sphères, un filtre a été implémenté afin de supprimer les solutions qui donneraient des distances fantaisistes entre les sphères. Si une erreur est détectée, on évince la solution et on continue la recherche dans l'image.

5.2. L'envoi d'informations

Pour communiquer au travers du réseau Internet, des méthodes standard à tous les systèmes d'exploitation ont été mises en place afin de pouvoir communiquer indifféremment de la plate-forme utilisée. L'entrée du tunnel bi-directionnel pour l'entrée et la sortie des données entre deux stations en réseau s'appelle "socket". Elle permet de lire et d'écrire sur des connexions à d'autres machines, sans se soucier des détails concernant l'interface avec le réseau.

Plusieurs protocoles différents permettent l'échange d'informations. Le plus simple est l'UDP (User Datagram Protocol). Il permet l'envoi à une machine de données divisées en paquets. Il n'y a pas d'établissement de communication à proprement parlé. On envoie les paquets à une certaine machine sur un certain port, mais il n'y a aucun signal de retour indiquant si les informations sont arrivées ou ont été lues. Ainsi, il peut y avoir des pertes en cours de route ou écrasement si la réception n'a pas pu se faire avant l'arrivée d'un nouveau paquet. Ce protocole est moins sûr, mais plus rapide que le TCP/IP.

Le deuxième protocole est le TCP/IP (Transmission Control Protocol/Internet Protocol). Il permet également l'envoi à une machine de données divisées en paquets. Mais cette fois, il doit y avoir établissement de communication entre deux machines. Une fois les ports ouverts, on peut envoyer et recevoir les données. De plus, il n'y a aucune perte d'information. Le protocole vérifie que les données ont réellement été reçues et les paquets qui arrivent ne remplacent jamais les précédents. Ils sont mémorisés dans des positions mémoires différentes. Si un problème survient (fin de communication à une extrémité, plus de possibilité de mémoriser les données, etc…) un message d'erreur est envoyé afin d'en avertir l'interlocuteur.

Les données qu'on souhaite envoyer sont les informations concernant notre objet tracké avec sa localisation selon 6 degrés de liberté. Il n'est pas important qu'un paquet ne soit pas reçu, puisqu'à chaque cycle, les informations sont des valeurs absolues. La fréquence d'envoi (15Hz en pleine fréquence) fait en sorte qu'une position manquante a peu d'incidence sur le rendu de la scène. Ainsi, le protocole UDP peut être utilisé.

En plus, il arrive parfois qu'on souhaite envoyer des images de la scène réelle afin de s'en servir pour texturer le mur du fond de notre monde virtuel. Comme ces envois ne se font que de temps en temps et qu'on préfère qu'il n'y ait aucun trou dans notre image, on préfère utiliser dans ce cas le TCP/IP.

Pour des besoins futurs, les deux protocoles seront peut-être nécessaires. Aussi ont-ils été les deux implémentés. Dans notre application, l'utilisation de l'un ou de l'autre n'aurait pas été critique, car les fréquences de travail sont relativement faibles en vue des performances du réseau. Toutefois, l'UDP est choisi pour envoyer les données concernant l'objet et le TCP/IP s'occupe des images. La justification de ce choix correspond aux motivations explicitées ci-dessus.

5.3. Le monde virtuel

A partir du moteur 3D fourni sous Delphi, la manière de créer un monde virtuel est déjà définie [CON98]. On peut créer un monde, poser les lumières, y insérer des squelettes d'objets, remplir les surfaces d'une certaine couleur ou d'une texture et poser une caméra dans une direction choisie.

Une fois ces objets virtuels créés, on peut les déplacer ou les tourner sans problème. Mais telle quelle, la librairie ne considère ces objets que comme un ensemble de surfaces qui peuvent se croiser tout naturellement. Elle ne garde aucune trace de l'identité de l'objet et donc, de ses caractéristiques propres.

5.4. L'interprétation

5.4.1. Détection de collision

Dans le monde réel, les objets ont toujours certaines propriétés réflectives ou interactives avec l'environnement. La principale est la solidité des objets qui fait en sorte que deux parties d'objets différents ne peuvent pas se trouver à la même place au même moment.

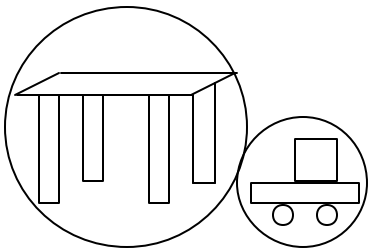


Fig. 10 : Collision entre sphères de

Afin de tenir compte de cette solidité, il faut détecter les collisions et si c'est le cas, revenir à une position antérieure qui donnait une réponse négative à ce test.

Il existe plusieurs méthodes plus ou moins gourmandes en ressources afin d'effectuer ce test. La première est la détection de collision entre sphères entourant les objets (Fig. 10).

Cette méthode est très simple, mais elle est relativement peu précise. Tous les cas de collision sont détectés, mais il y a beaucoup de détections qui le sont à tort.

Une deuxième méthode pourrait être la décomposition des objets en plusieurs sphères intérieures plus ou moins grandes (Fig. 11). Le problème de cette méthode est la complexité à trouver les bonnes sphères pour des objets complexes et les erreurs qui peuvent survenir dans certains cas (coin d'un objet sur une surface plane)

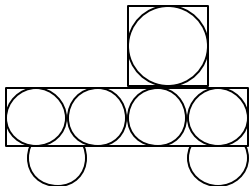


Fig. 11 : Décomposition en sphères intérieures

La troisième méthode est celle détectant la collision entre les surfaces de deux objets. Avec notre moteur 3D, toute surface d'objet est composée de triangles élémentaires. Si on arrive à déceler le croisement dans l'espace entre un triangle d'un premier objet avec un autre du deuxième, on détecte une collision.

Le défaut de cette méthode est la gourmandise en calculs lorsque les objets sont œmposés de beaucoup de triangles. Par exemple, la détection de croisement entre deux sphères (composées chacune de 72 triangles) demandera une analyse de 5184 croisements. Mais cette détection est absolue et aucun cas n'est incorrect.

Par contre, il est inutile de détecter le croisement entre tout l'environnement et notre objet. Seul l'environnement proche a besoin d'être étudié. Pour cela, un filtre grossier est mis en place et combine la première méthode décrite dans ce chapitre. Chaque élément possède une sphère de contour. Avant de tester une détection précise entre triangles des deux objets, on regarde si les deux sphères se croisent. Si c'est le cas, on procède à la détection précise. Dans le cas contraire, on passe à l'analyse de croisement avec un autre objet.

5.4.2. Contraintes sur certains degrés de liberté

Dans certains cas, plusieurs degrés de liberté peuvent être bloqués. Ces contraintes proviennent des caractéristiques connues à priori des objets virtuels. Par exemple, un robot mobile ne se déplace que sur le sol et ne possède qu'une seule possibilité de rotation. Pour ce modèle, les déplacements verticaux ainsi que deux degrés de rotation sont bloqués.

6. Implémentation

6.1. L'orientation

6.1.1. Principe

Le procédé adopté est le suivant : on part du principe que trois points non rectilignes suffisent à caractériser tous les degrés de liberté d'un solide. Ainsi, avec un repère composé d'un minimum de 3 sphères, en localisant leur emplacement, on peut définir sa position et ses angles de rotations. Les sphères sont choisies de couleur différente afin de ne pas interférer entre elles durant le tracking.

La méthode retenue pour obtenir les angles de rotation à partir des positions des sphères est celle-ci :

1) A chaque vecteur de position des sphères, on soustrait les coordonnées du centre de rotation afin d'éliminer les composantes de translation. Le centre de rotation peut être défini à choix comme la position d'une certaine sphère ou le centre entre toutes.

$$(y_x^1\ y_y^1\ y_z^1)^T = (yloc_x^1\ yloc_y^1\ yloc_z^1)^T - (ycrot_x\ ycrot_y\ ycrot_z)^T$$

2) La matrice de corrélation K est le fruit de la somme des multiplications de y^*x^T (où x^i sont les positions des sphères pour des rotations initiales égales à 0)

$$K = \sum_{i=1}^{n} y^i \cdot x^{iT}$$

3) La matrice K peut toujours être décomposée sous la forme : $K = U * S * V^T$. U et V sont orthogonales tandis que S est une matrice diagonale. Cette décomposition s'appelle la SVD (singular value decomposition). Un algorithme a été implémenté afin d'obtenir les matrices souhaitées.

4) La matrice de rotation donnant les angles d'Euler se trouve ainsi :

$$R = U \cdot \begin{pmatrix} 1 & & \\ & 1 & \\ & & \det(U \cdot V^T) \end{pmatrix} \cdot V^T$$

5) De cette matrice, on déduit les 3 angles de rotation. Il faut simplement prendre garde qu'il y a 8 résultats solutions des équations (dus à la périodicité du sinus et de la tangente). Afin de retrouver les angles de rotation corrects, on teste les 8 couples d'angles solutions en les insérant dans une matrice de rotation. Une seule de ces matrices correspond à la matrice originale.

Toutefois, les angles de rotation calculés lors de chaque itération peuvent varier même si l'objet n'a pas bougé. Ce défaut, dû aux erreurs d'imprécision, peut être limité en ajoutant un offset minimal avant d'envoyer des angles différents des précédents. Si un angle n'a pas varié de plus de ±5° par rapport à la dernière position, c'est l'ancien angle qui est envoyé. La valeur définissant cette zone morte est un choix résultant de l'expérimentation. Mais elle peut être modifiée sans difficulté.

6.1.2. Fonctions implémentées

La librairie "Tlorient.c" contient les diverses fonctions utilisées pour trouver les angles de rotation (Fig.12).

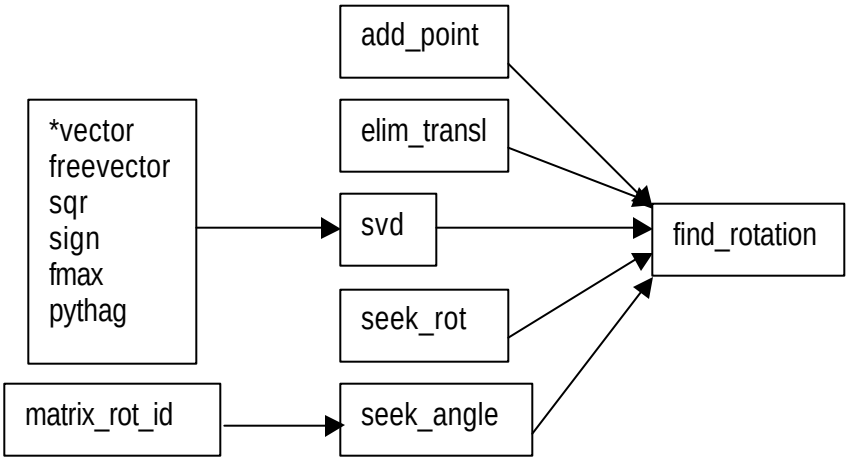


Fig. 12 : Tableau des fonctions pour la recherche d'orientation

Le premier bloc (*vector…) est un ensemble de petites fonctions plusieurs fois utilisées dans le calcul de la décomposition en valeurs singulières.

int matrix_rot_id(double psi, double tetha, double phi, double mat[3][3]) vérifie que les angles trouvés correspondent à la matrice de rotation. En effet, on trouve 8 solutions d'angles pour une certaine matrice de rotation, mais seule une est correcte.

void add_point(double p_old[3], double p_new[3], double k[4][4]) crée une matrice de corrélation à partir de la nouvelle position d'un point et de son ancienne, ainsi que de la matrice déjà obtenue pour les points précédents.

void elim_transl(double gived_pt[3], double center_pt[3]) enlève à un point la quantité correspondant à la translation dans le mouvement.

void svd(double a[4] [4], int m, int n, double w[4], double v[4][4]) est la fonction qui calcule la décomposition en valeurs singulières d'une matrice.

void seek_rot(double U[4][4], double V[4][4], double R[3][3] renvoie la matrice de rotation à partir des matrices obtenues par la svd.

void seek_angle(double R[3][3], double angle[3]) renvoie les 3 angles d'Euler d'une matrice de rotation.

void find_rotation(double pt_find1[3], double pt_find2[3], double pt_find3[3],double angle[3]) est la fonction principale de cette unité. Les 3 points donnés en entrée sont les positions des 3 sphères dans l'espace. En sortie, elle renvoie les 3 angles de rotation.

Connaissant notre repère, nous savons que le premier point se situe à l'origine et que les autres se trouvent à angle droit dans une certaine direction. A partir de là, on retrouve les angles de rotation par rapport à cette position d'origine.

6.2. Filtrage du tracking du marqueur

Connaissant à priori le marqueur tracké, on peut utiliser ces informations pour évincer les solutions erronées. Ces solutions fantaisistes sont obtenues dès que l'une des sphères qu'on croit avoir localisées est une méprise avec un autre objet.

Le marqueur utilisé (Fig. 13) est constitué de 4 sphères situées à angle droit par rapport à l'une d'elles.

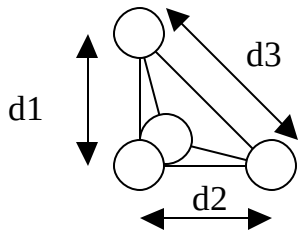


Fig. 13 : Marqueur utilisé

A chaque tentative de localisation, on calcule les distances entre les 3 sphères détectées. On prend évidemment en compte les distances dans l'espace à 3 dimensions.

Si (d1± 30% ? d2± 30% ? (d3* 0.707) ± 30%), alors une ou deux des sphères n'ont pas été localisées correctement. Cette détection d'erreur relance la procédure de localisation.

Une autre solution aurait été de considérer les dimensions absolues du marqueur pour les comparer avec les distances obtenues. De cette manière, quelques cas supplémentaires d'erreur pourraient certainement être évités. Toutefois, il a fallu trancher entre un perfectionnement du tracking du marqueur et les inconvénients que cette solution aurait apporté. En effet, la stéréo donne des informations de distances multipliées par un certain facteur. Il faudrait garantir que le

système soit calibré, ce qui n'est pas une contrainte nécessaire. De plus, pour autant que la forme du marqueur soit conservée, une mise à l'échelle du système n'implique aucune modification.

L'implémentation de cette fonction est faite directement lors de l'utilisation des librairies de tracking.

6.3. La liaison réseau

6.3.1. Explication

Deux ports différents sont ouverts. Le premier (UDP) sert à l'envoi des informations concernant l'objet traqué. Il s'agit d'une chaîne de bytes dans laquelle on place les données. Les informations envoyées sont le type d'objet (1 byte), ses couleurs rouges, vertes et bleues (3 bytes), les dimensions, les positions selon chaque axe (3*2 bytes) et les angles de rotation en degré (3*2 bytes).

Un deuxième port (TCP/IP) est également ouvert pour l'envoi d'images prises à l'aide de la caméra. Les images sont au format RGB, 320*240 pixels, ce qui veut dire que chaque pixel est composé d'un byte indiquant la quantité de rouge, un pour le vert et un autre pour le bleu. Cela fait une chaîne de 230'400 bytes pour une image.

6.3.2. Fonctions implémentées côté tracking (C++)

Du côté de l'envoi des informations, une unité "nettools.c" contient les fonctions nécessaires à la préparation d'une communication sur le réseau ainsi qu'à leur envoi (Fig. 14).

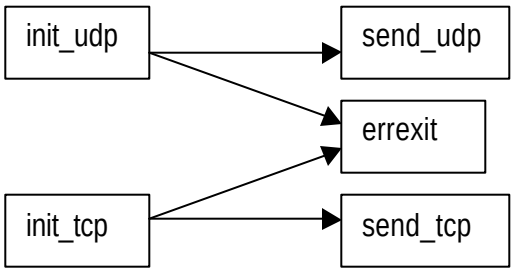


Fig. 14 : Tableau des fonctions de communication

*void init_udp(const char *hostname, unsigned short UDPPort)* prépare le connexion UDP.

*void init_tcp(const char *hostname, unsigned short TCPPort)* prépare le connexion TCP/IP.

void send_tcp (char Buffer[], int n_bytes) envoie un message d'une longueur de n_bytes sur le port TCP/IP.

void send_udp (char Buffer[], int n_bytes) envoie un message d'une longueur de n_bytes sur le port UDP.

*void errexit (const char *ErrorMessage)* est la fonction qui sort du programme et envoie un message en cas d'erreur (par exemple lorsque une connexion ne peut se faire).

6.3.3. Fonctions implémentées côté monde virtuel (Delphi)

Pour la connexion UDP, il suffit d'utiliser le composant standard TNMUDP et de lui spécifier dans ses propriétés le nom de la machine qui se connecte, un numéro de port et de l'activer. Toute la communication est gérée par l'environnement et un événement "OnDataReceive" est produit lorsqu'un message arrive. Dans la procédure lancée par cet événement, on crée un flux stockant les données dans la mémoire vive (TmemoryStream.Create) et on lit les données arrivées dans ce flux.

Pour la connexion TCP/IP, il y a également un composant standard : TServerSocket. On lui spécifie également le nom de la machine qui se connecte, un numéro de port et un état actif. Lorsqu'un message arrive, un événement OnGetThread est produit et on lance la création d'un nouveau Thread.

Une procédure TMyServerThread.ClientExecute a été implémentée afin de lire ces données qui arrivent. La procédure balaie le port et lit les données jusqu'à ce que l'entier des données soit reçu.

6.4. Le monde virtuel

Le code implémenté crée un monde virtuel contenant les lumières, la caméra, des objets bornant l'environnement et l'objet en déplacement. De plus, la caméra peut être déplacée afin de donner la possibilité de regarder la scène sous un autre angle.

Delphi étant un environnement visuel, les composants sont placés dans une fenêtre créée à cet effet. Seules les principales fonctions sont décrites. Tant que le contenu des événements provoqués est simple, une nouvelle procédure n'a pas été implémentée séparément (comme par exemple le déplacement de la caméra). Pour les événements impliquant des tâches plus importantes, des procédures séparées ont été déclarées (Fig. 15).

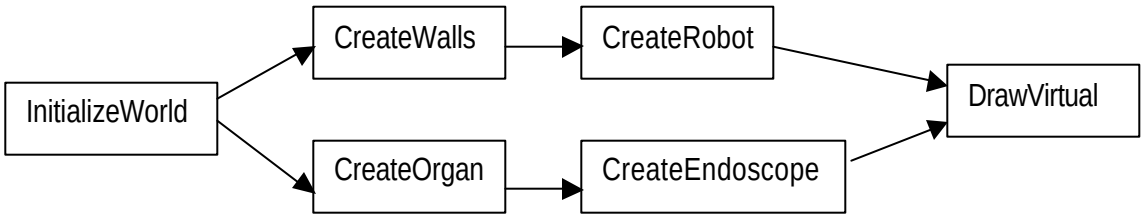


Fig. 15 : Tableau des procédures créant les objets virtuels

Procedure InitializeWorld initialise notre monde. Elle crée une scène virtuelle, initialise les propriétés de la lumière ainsi que de la caméra. Elle lance l'une ou l'autre des deux applications.

Procedure CreateWalls crée dans la fenêtre virtuelle les murs qui serviront de base pour l'exemple du Robot Pionneer.

Procedure CreateRobot sert toujours le même exemple. Il dessine, élément après élément, un robot Pionneer.

Procedure CreateOrgan crée dans la fenêtre virtuelle un œsophage suivi par un estomac pour l'exemple du simulateur de gastroscopie.

Procedure CreateEndoscope est utilisée pour le même exemple dans la création virtuelle d'un gastroscope.

Procedure DrawVirtual(s :string) a pour but de traduire la chaîne reçue par la communication réseau et de traduire cette chaîne pour positionner l'objet mobile selon les informartions reçues (après leur interprétation)

6.5. L'interprétation

6.5.1. Détection de collision

Pour le filtre de collision grossier, une fonction implémentée permet de calculer le centre d'un objet et de trouver la distance avec son point le plus éloigné (par simple itération). Ces données sont introduites dans des propriétés ajoutées à celles existantes pour les objets. Dès qu'un objet est créé, on procède à cette opération.

Pour notre objet mobile se déplaçant dans son environnement composé d'objets fixes, on regarde si la distance entre chacun des objets fixes et mobiles est inférieure à la somme des deux rayons des sphères. Si c'est le cas, il y a peut-être collision, aussi procède-t-on à la détection plus fine.

La détection de croisement entre deux triangles se fait de la manière suivante (Fig. 16):

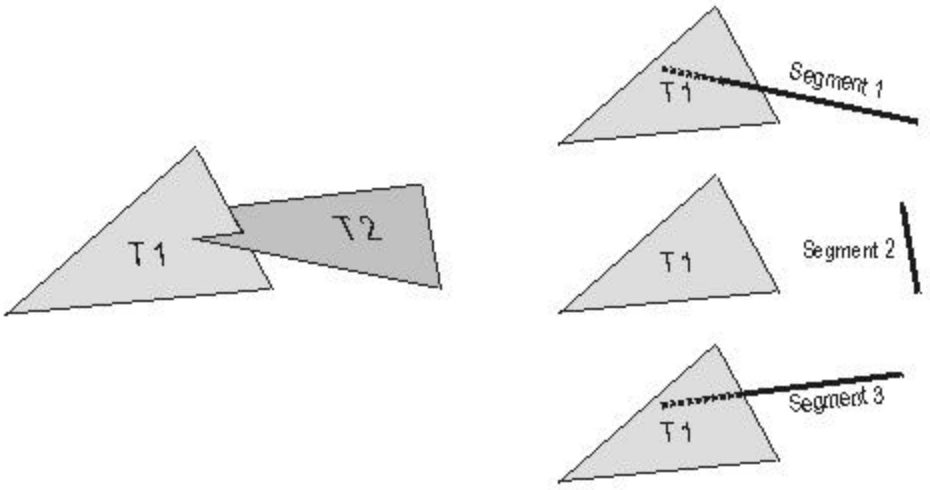


Fig. 16 : Croisement entre 2 triangles : effectif si un des segments formant le 2ème triangle intercepte le 1er

La difficulté vient que les triangles se trouvent dans l'espace et non dans le plan. On commence par décomposer le triangle 2 en trois segments de droite et on va, en fait, détecter le croisement entre ces trois segments et l'autre triangle.

On commence par chercher le croisement entre la droite porteuse du segment et le plan du triangle. Si ce point est en dehors du segment, il n'y a pas croisement. Il en est de même si le point est en dehors des limites formées par le triangle lui-même. Dans ce cas, on peut passer à l'analyse avec un autre segment ou entre deux autres triangles. Dans le cas contraire, une collision est décelée et on force l'objet mobile à revenir dans sa position avant la collision.

Lorsqu'une collision est détectée, on revient à la position précédente afin de retrouver la position où les solides virtuels ne font que de se toucher.

6.5.2. Exemple de collision

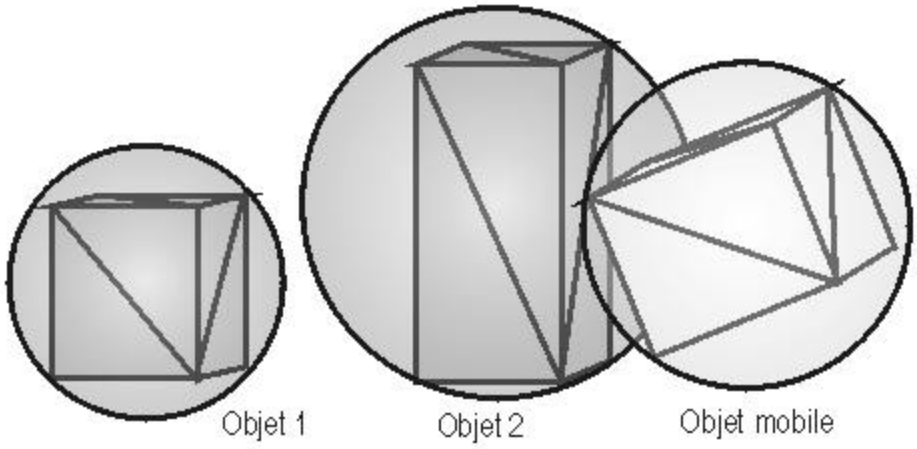


Fig. 17: Exemple d'objets virtuels avec sphères les délimitant

Prenons l'exemple d'un environnement composé de deux prismes droits et d'un objet mobile étant également un prisme (Fig. 17). Pour la détection de collision, le filtre grossier va s'appliquer entre l'objet mobile et l'objet 1. Comme les sphères les entourant ne se croisent pas, la détection entre tous les triangles des 2 objets n'est pas effectuée, car les objets sont trop éloignés. En effet, la distance entre les centres des 2 objets est supérieure à la somme des deux rayons.

Ensuite, le filtre va s'appliquer entre l'objet 2 et lobjet mobile. Cette fois, les objets sont déclarés proches car les sphères se croisent. La détection fine est opérée et chaque triangle de l'objet 2 est analysé avec chacun de l'objet mobile.

En enlevant le filtre grossier : détection de collision entre 2 * 12 triangles et 12 triangles * 3 segments –›

détection de croisement entre 24 triangles et 36 segments –› 864 itérations

Avec le filtre grossier: détection de collision entre 1*12 triangles et 12 triangles* 3 segments + 2 filtres –›

détection de croisement entre 12 triangles et 36 segments –› 432 itérations + 2 filtres grossiers

Si l'environnement est très complexe, de même que l'objet mobile (décomposé en plusieurs sous-élements), l'efficacité et la nécessité du filtre grossier ne fait pas de doute.

6.5.3. Contraintes sur certains degrés de liberté

Suivant le modèle virtuel créé, l'interprétation nécessite l'application de contraintes sur certains degrés de liberté. La manière de procéder est la suivante : depuis le module réseau, les informations concernant les six degrés de liberté sont interceptées et filtrées. Ce n'est qu'ensuite qu'elles passent vers le module de positionnement de l'objet mobile.

6.5.4. Fonctions implémentées

L'unité "xinterpret.pas" contient les fonctions utilisées pour l'interprétation (Fig. 18).

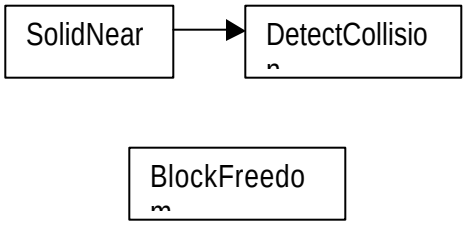


Fig. 18 : Fonctions d'interprétation de mouvement

function SolidNear(Solid1, Solid2 : TDSSolid) : Boolean exécute le filtrage avant la détection de collision. Cette fonction regarde si deux objets ont leur sphère de contour qui se touchent.

function DetectCollision(Solid1, Solid2 : TDSSolid) : Boolean détecte les collisions entre deux objets virtuels par les croisements entre triangles formant les surfaces, une fois le filtre de proximité appliqué.

procedure BlockFreedom(var posx,posy,posz,angley,angley,anglez:Real) applique certaines contraintes sur la position ou l'orientation. Par exemple, c'est ici qu'on introduit le blocage d'un degré de liberté.

7. Evaluation du système

7.1. La localisation

7.1.1. Multi-tracking

Fig. 19 : Segmentation en profondeur : une des images source et image filtrée

La Fig. 19 nous montre une image "avant et après" la segmentation en profondeur. Comme on peut le voir sur l'image filtrée, notre marqueur est effectivement détouré, mais il n'est par le seul. Une certaine quantité de bruit apparaît également. Ce filtre est visiblement insuffisant pour être utilisé seul, mais dans le cas qui nous intéresse, il y a déjà un apport d'information qui pourra être utilisé par la suite.

La deuxième étape est le filtre couleur à partir de la même image. Etant donné que le filtre couleur est différent pour chaque sphère, on peut ainsi appliquer le filtre pour chacune et les repérer.

La troisième étape est le filtre morphologique sur l'image avec filtre couleur. On peut choisir quel type de forme doit être recherché et sa dimension (ex : rectangle de 5 cm sur 5 cm). Ainsi, on évite que des pixels distincts sur l'image, ne correspondant qu'à un point, ne viennent perturber notre tracking. De cette manière, on évite les bruits de petite dimension.

Il n'y a aucun problème pour effectuer simultanément un multi-tracking. Lorsque toutes les sphères ont été repérées et que la recherche des nouvelles positions se fait uniquement dans la zone proche et plus sur toute l'image, on atteint des fréquences de tracking de 15Hz avec notre installation (voir chap. 4). En phase de recherche sur toute l'image, on atteint une fréquence de 2Hz.

La figure 20 montre l'image qui est conservé avec l'addition des trois filtres.

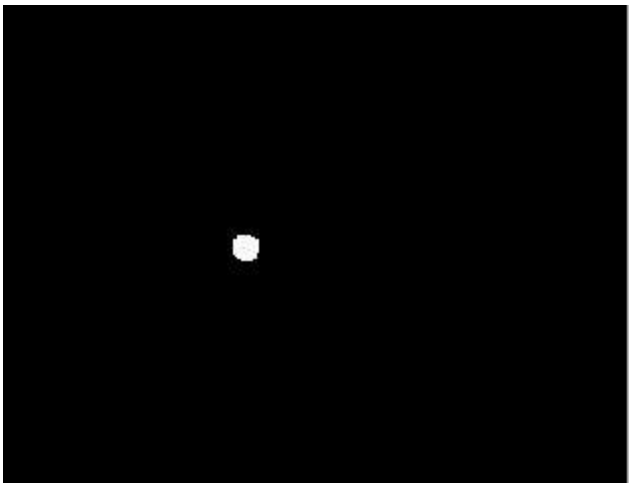


Fig. 20 : Image avec la combinaison des 3 filtres

Evidemment, pour parvenir à un résultat optimal, il faut choisir pour les sphères des couleurs qui ne sont pas trop proches entre elles ou avec l'environnement. Ainsi, lorsqu'on applique le filtre, on évite les perturbations et limite les erreurs.

7.1.2. Le marqueur

Fig. 21 : Marqueur utilisé

Plusieurs modifications ont été effectuées sur le marqueur (Fig. 21). La première concerne la grandeur des sphères. Les premières sphères utilisées mesuraient 27mm de diamètre. Mais avec de telles dimensions, il était difficile d'être suivi avec un éloignement de quelques mètres. Elles ont été remplacées par des sphères de 50 mm. Ainsi, on peut s'éloigner un peu plus.

La deuxième modification a été la distance entre les sphères. Pour commencer, l'arête la plus courte mesurait 50 cm. Mais après quelques essais, nous avons remarqué que cette distance n'était pas nécessaire, aussi a-t-elle été diminuée par deux.

La troisième modification a été les sphères en elles-même. Le système stéréoscopique détectant mieux les surfaces structurées, un filet a été déposé sur chaque sphère afin d'améliorer la segmentation en profondeur.

Il faut remarquer que la dimension du marqueur optimal dépend de la distance sur laquelle on souhaite travailler. Notre but est de travailler entre 1 et quelques mètres. Sur cette profondeur, le marqueur devrait avoir les dimensions utilisées. Mais si on traquait des objets plus petits à une distance inférieure, le marqueur idéal aurait également des dimensions à l'échelle. Il suffirait alors d'ajuster e système stéréoscopique.

Tel quel, le système permet de tracker notre marqueur entre 1 et 5 mètres. Mais un facteur limitant les performances est la variation d'illumination (lumière extérieure, artificielles,…) qui rend les filtres de

couleur moins idéaux. L'utilisation de couleurs normalisées limite déjà en partie ce phénomène, mais des problèmes dus à la saturation subsistent.

7.2. La liaison réseau

L'envoi des informations concernant la localisation de l'objet de pose aucun problème. Etant donné que les envois ne se font qu'à 15 Hz, fréquence maximale du multi-tracking, et que la chaîne de bytes est courte, le réseau n'a aucun problème à suivre ce débit. Le réseau peut suivre des débits nettement plus élevés.

Pour l'envoi d'images trackées en continue, le réseau peut suivre des fréquences de 3Hz. Ce qui correspond à un débit d'environ 6Mbits/sec. Toutefois, cette fréquence dépend du type de réseau utilisé, de sa surcharge à un moment donné. Si l'expérience devait se réaliser au travers du réseau Internet jusqu'au USA, jamais ce débit ne serait atteint, même durant des heures moins surchargées.

7.3. Le monde virtuel

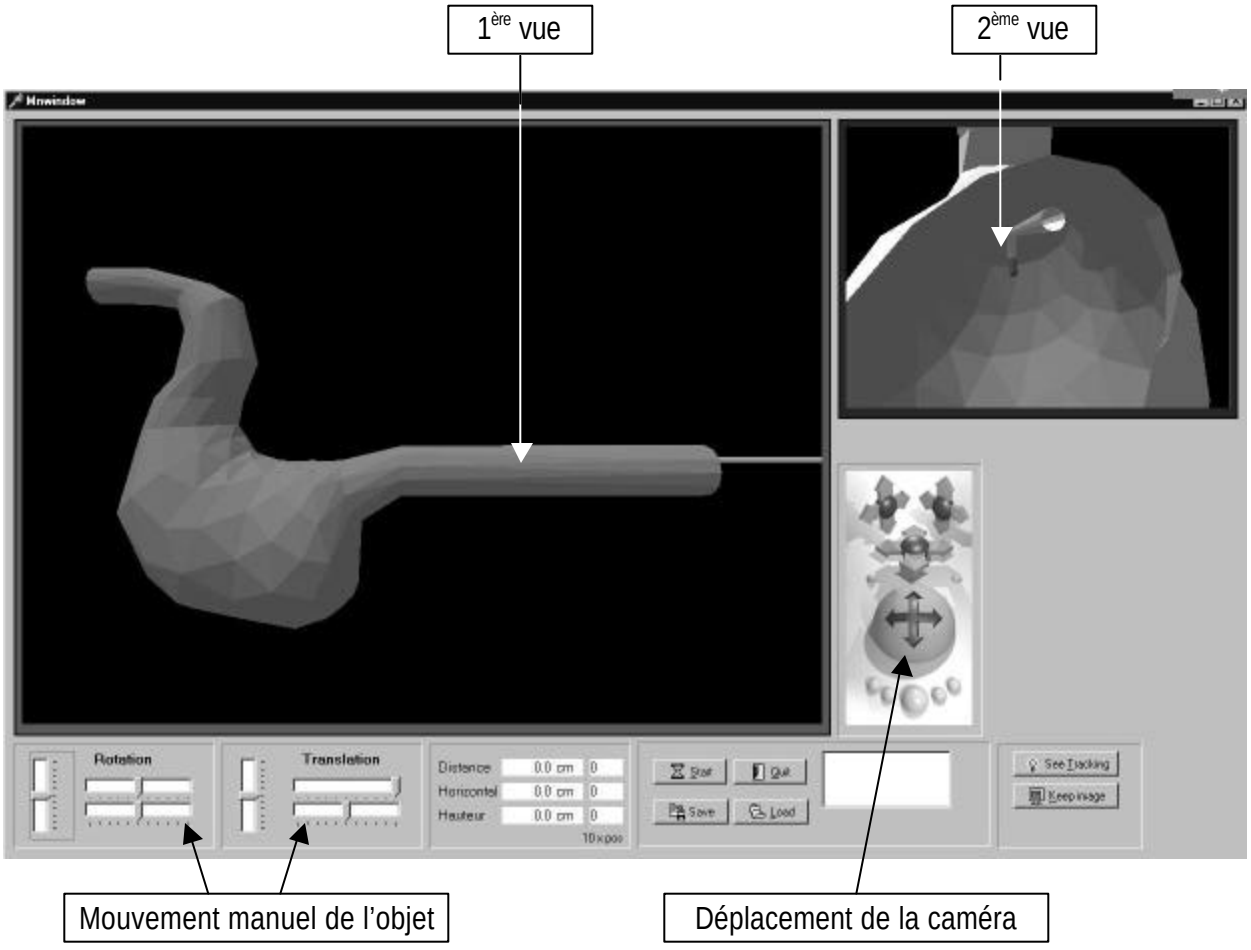


Fig. 22 : Capture d'écran du monde virtuel

La fenêtre du monde virtuel crée l'environnement dans lequel sera placé l'objet mobile (Fig. 22), à savoir pour nos exemples, une salle avec des murs ou des organes humains. La position de la caméra ainsi que la direction dans laquelle elle regarde peut être choisir à l'aide d'un sélecteur approprié.

Deux vues sont proposées simultanément : par exemple une vue de face et la vue qu'aurait notre objet mobile. Ces vues peuvent être modifiées au gré de l'utilisateur, car un bouton permet de déplacer la caméra suivant la position et l'angle de son choix.

7.4. L'interprétation

L'interprétation générale bloquant certains degrés de liberté fonctionne correctement. De plus, lorsque l'utilisateur manipule lui-même le marqueur, le mouvement est assez intuitif.

Concernant la détection de collision, celle-ci s'effectue également correctement. Un problème qui aurait pu survenir est la lenteur provenant de la multiplication des détections de collision. Mais heureusement, grâce au filtre grossier testant la proximité entre solides, un bon nombre de calculs est évité.

Pour des scènes complexes, composées de beaucoup de triangles, des tests ont été effectués avec ou sans le filtre grossier. La différence est visible : saccadés sans le filtre, les mouvements deviennent fluides une fois le filtre intégré.

7.5. 1er exemple : le robot Pionneer

7.5.1. Description

Fig. 23 : Robot Pionneer avec le marqueur

Dans cet exemple (Fig. 23), le but est de recréer virtuellement à distance un objet qui se déplace afin d'en obtenir une représentation mieux adaptée. Il s'agit de montrer les capacités de télé-observation par l'acquisition d'information, de transfert sur le réseau et d'interprétation afin de rendre la virtualisation de notre objet plus précise.

Le Pionneer est un robot mobile autonome avec plusieurs capteurs de distance et une caméra. Notre marqueur est placé sur ce robot et la caméra stéréo observe la scène.

La première étape est de détecter la position du marqueur et de ses sphères comme décrite au point 7.1. Ces informations sont ensuite envoyées sur le réseau.

Du côté monde virtuel (Fig. 24), l'environnement peut être totalement imaginaire (robot se déplaçant dans un tunnel) ou très proche de la réalité. Dans ce cas, une salle avec ses murs est déjà prête et il est possible de prendre une image de la scène réelle pour l'utiliser comme texture des murs.

Nos connaissances à priori de l'objet suivi permettent d'implémenter plusieurs comportements. Le premier est le blocage de certains degrés de liberté. L'apesanteur fait que notre robot réel est collé au sol, il en est de même pour son jumeau virtuel. Ainsi, bien que ce soit la position du marqueur qui soit donnée et attribuée dans un premier temps au robot virtuel, l'interprétation rend la position finale du robot correcte.

La deuxième correction est la détection de collision. Cette détection permet, par exemple, de bloquer l'avancement virtuel du robot si une erreur s'est glissée dans la localisation et que rien n'indique qu'il se trouve face au mur. De plus, cet anti-collision permet également de bloquer les degrés de rotation qui ne peuvent pas s'effectuer lorsque le robot est au sol. Il aurait été plus facile de bloquer à la base les rotations qui ne sont généralement pas utilisées. Mais cela n'est plus applicable lorsque le sol n'est pas plat ou que le passage sur certains obstacles fait pencher le robot.

Ainsi notre robot virtuel se déplace identiquement à son équivalent réel.

Fig. 24 : Scène virtuelle contenant le robot Pionneer

7.5.2. Avantages d'utilisation du système

Les avantages de se représenter notre robot dans un environnement virtuel sont tout d'abord le peu d'informations qui transitent par le réseau. En quelques bytes, on peut se représenter à distance notre scène réelle. Alors que si on envoie des images filmées, la quantité d'informations est nettement plus élevée et les données reçues ne correspondent pas forcément aux questions essentielles que sont la position, l'orientation, devant ou derrière tel objet…etc. L'autre avantage est de pouvoir changer la position de la caméra et ainsi choisir la vue qui nous sied le mieux.

De plus, en utilisant la vision comme source d'information (par opposition aux capteurs et encodeurs du robot), on obtient une localisation absolue et fiable. A l'inverse des encodeurs, il ne souffre pas de l'accumulation d'erreur. Ainsi le système pourrait être utilisé pour recalibrer la position estimée par les encodeurs.

Dans cette situation, le choix d'un système de vision pour l'acquisition du positionnement est justifié si l'on souhaite un système totalement passif et qui peut s'adapter rapidement à un nouvel environnement ou un nouvel objet qui se déplace (le robot, ici en l'occurrence). Le marqueur peut être posé sur n'importe quel objet et une fois son équivalent virtuel créé et les conditions d'interprétations posées, notre monde virtuel peut être créé et mis à jour. On pourrait voir évoluer, par exemple, un stylo avec un marqueur accroché, un robot Puma, une machine de chantier, etc…

7.5.3. Limitations

Les principales limitations de l'utilisation de ce système au travers de cette application proviennent d'abord des limites du tracking et des algorithmes actuellement utilisés. La gamme des distances de travail et la précision des informations concernant la profondeur peuvent être gênantes.

Ensuite, la pose d'un marqueur sur l'objet en mouvement n'est pas réalisable dans toutes les applications.

7.5.4. Autres solutions

Pour améliorer la localisation de notre robot, mis à part les perfectionnements possibles avec le système de vision, on pourrait également utiliser des systèmes sans vision. On pense notamment à une pastille magnétique qui interférerait avec des signaux envoyés depuis les bords de l'espace de travail.

Mais dans une telle situation, on perd les avantages de la vision et ensuite, le système n'est pas non plus applicable dans toutes les situations. Par contre, en se libérant d'un marqueur pour le tracking avec la vision, le système peut être appliqué dans quasiment tous les domaines et environnements.

7.6. 2ème exemple : le simulateur de gastroscopie

7.6.1. Introduction

Le domaine médical est une terre très fertile pour les recherches dans le domaine de la réalité virtuelle. En effet, que ce soit pour de la simulation ou pour obtenir des vues mieux adaptées d'une partie de corps humain, la puissance de calcul permet aujourd'hui de recréer des scènes très réalistes et même interactives.

Les simulations sont tout aussi utiles pour le débutant qui désire s'initier que pour le pratiquant averti qui veut s'exercer à effectuer une tâche délicate. Toute pratique n'est jamais bénigne et les interventions qui peuvent être évitées ou écourtées procurent moins de désagrément au patient. C'est également souvent un allégement du facteur de risque.

Dans ce contexte, une application possible pour notre système est l'oesophago-gastro-duodeno-scopie (Fig. 25). Evidemment, pour obtenir un produit final utilisable pour un médecin et commercialisable, le développement n'est de loin pas terminé. Mais là n'est pas notre prétention. Il s'agit principalement de montrer par un exemple approximatif les potentialités du système et laisser les portes ouvertes pour un travail futur basé sur ce travail.

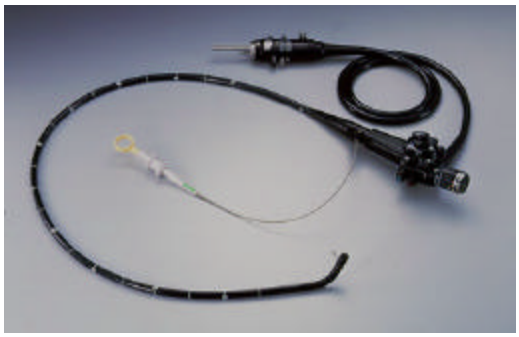


Fig. 25 : Exemple de gastroscope : Olympus JF 1T40

Cette opération consiste à introduire un gastroscope par la bouche à travers l'œsophage jusque dans l'estomac. Au bout de cet instrument (Fig. 26) se trouve l'extrémité d'une fibre optique qui permet de visionner l'intérieur du conduit (Fig. 27) et d'un outil vibrant destiné à effectuer des incisions.

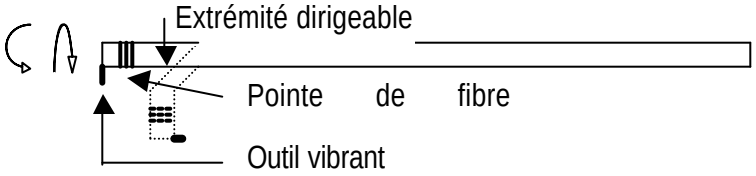


Fig. 26 : Gastroscope modélisé

Pour celui qui manœuvre cet instrument, deux manipulations peuvent être effectuées : l'avancée dans l'œsophage et la rotation. Mais en plus de cela, le dispositif possède une extrémité dirigeable commandable à distance. Cette extrémité peut généralement se pencher jusqu'à 90° en avant et 90° en arrière. Mais elle peut également tourner sur elle-même de 90° à gauche et à droite.

Des modèles moins récents avaient encore un corps principal rigide. Mais aujourd'hui, celui-ci est flexible. Toutefois, pour simplifier la démonstration, cette flexibilité n'a pas été insérée.

Fig. 27 : Oesophage avec tumeur

7.6.2. Description

Pour cette démonstration, le marqueur tracké représente l'avancée virtuelle du gastroscope dans l'œsophage. Si on omet le mouvement de l'extrémité dirigeable, le gastroscope suit un mouvement parallèle à celui-ci (Fig. 28).

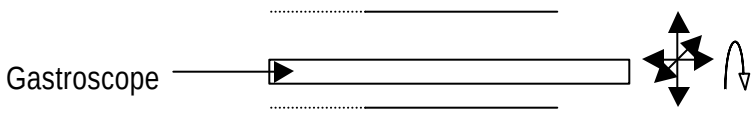


Fig. 28 : Degrés de liberté du gastroscope dans l'oesophage

Afin de manipuler l'extrémité dirigeable, on utilise les 2 rotations du marqueur restantes.

La position du marqueur tracké est envoyée sur le réseau et ces informations sont interprétées afin de manipuler un gastroscope virtuel. L'environnement dans lequel il s'insère est évidemment un œsophage et un estomac. L'application offre deux vues simultanées à l'utilisateur : une vue virtuelle représentant ce qu'il verrait effectivement à la pointe de son instrument et une vue de face éloignée lui permettant de mieux voir ses manipulations.

7.6.3. Avantages d'utilisation du système

Dans l'état actuel (Fig. 29), ce système permet déjà de voir les manipulations à effectuer pour arriver dans une certaine zone du conduit et de montrer les types d'images qu'on reçoit du gastroscope une fois à l'intérieur. En effet, la vue de face accouplée avec celle de la pointe offre des informations supplémentaires.

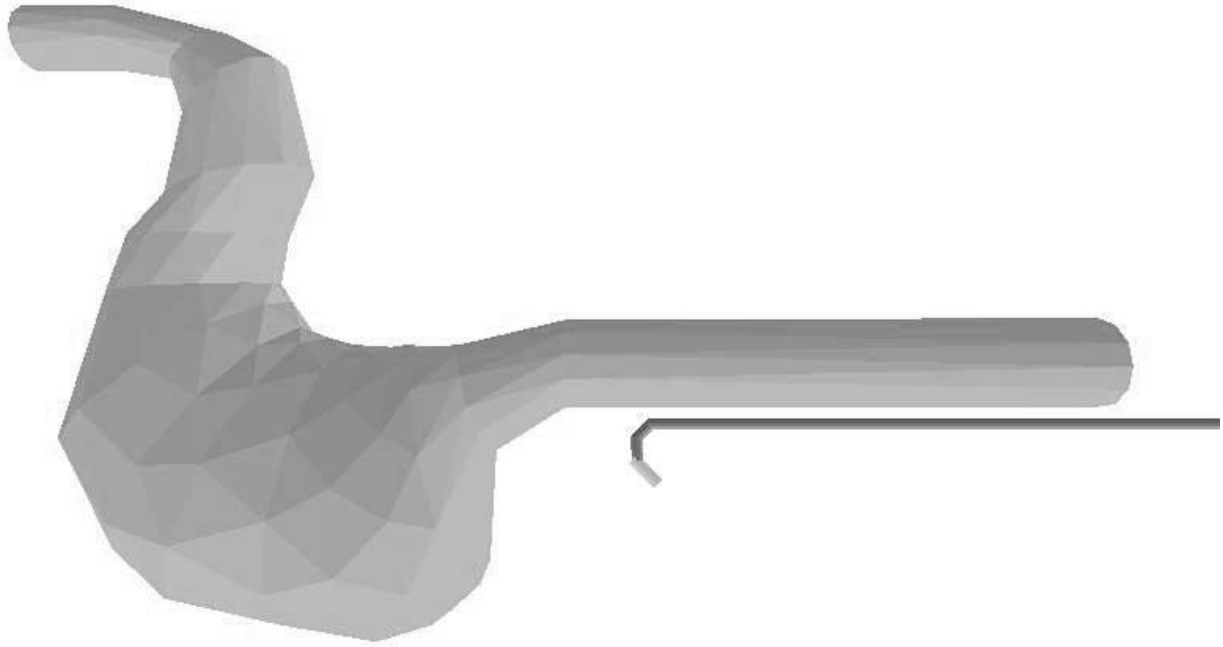


Fig. 29 : Oesophage et estomac virtuel avec le gastroscope

7.6.4. Limitations

La démonstration actuelle est trop simplifiée pour être déjà couramment utilisée. Toutefois, elle laisse entrevoir un produit qui, après quelques développements, sera d'un intérêt avancé.

Comme première limitation, il y a tout d'abord les simplifications apportées dans notre modèle. Un instrument virtuel flexible, comme généralement dans la réalité, serait préférable. Ensuite, il y a toujours le marqueur. Si on parvient à observer un instrument réel et à en définir la position uniquement par l'analyse d'image, on enlève l'encombrement que celui-ci peut causer.

Finalement, il manque toujours dans un tel modèle les sensations que ressent l'utilisateur dans la réalité. Pour un véritable simulateur, cet aspect doit absolument être intégré. Une technologie de retour de force telle que disponible au groupe VRAI pourrait être greffée sur le système.

7.6.5. Perfectionnement

A partir de ce travail deux directions peuvent être entrevues. L'assistant de gastroscopie en temps réel et le simulateur. Dans les deux cas, un modèle flexible de gastroscope doit être utilisé. Pour introduire la flexibilité dans l'instrument, une base de départ pourrait être le modèle dynamique des objets qui ont été insérés dans la librairie du moteur 3D [CON99].

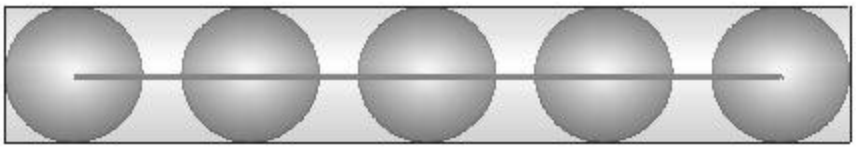


Fig. 30: Modèle de solide dynamique pour gastroscope flexible

Dans ce modèle de solide dynamique (Fig. 30), le solide est composé de sphères intérieures pour lesquelles on applique les lois physiques. La surface du gastroscope n'a qu'à suivre les mouvements des sphères. Pour appliquer totalement ce modèle, quelques améliorations doivent encore être apportées.

A côté de cela, notre modèle d'organe approximatif n'est, suivant quel cas, pas adapté. L'intégration des fonctions de reconstruction tridimensionnelle à partir de plusieurs images est réalisable.

L'assistant en temps réel de gastroscopie

Lorsque le médecin opère, il n'a à disposition que la vue apportée par son gastroscope. Il doit estimer la position où il se trouve à partir de ces images. Il est possible de recréer des organes virtuels et d'y positionner notre instrument. Ainsi, en plus de la vue réelle de l'intérieur, il possède une vue virtuelle de face (où d'un autre lieu de son choix) qui situe exactement à quel endroit il se trouve. Les manipulations gagnent alors en précision.

De plus, on peut également imaginer une reconstruction virtuelle des organes basée sur des images réelles d'un scanner et non sur des approximations. Si cette étape est trop lourde à supporter par le patient, on peut également imaginer de commencer l'observation avec notre modèle approximatif et corriger notre modèle virtuel grâce aux images fournies par la fibre optique.

Pour un tel objectif, la suppression du marqueur est certainement une nécessité, à moins de parvenir à l'intégrer à l'extrémité sans trop d'encombrement. Pour y arriver, un tracking avec reconnaissance des bords est assurément une solution.

Le simulateur de gastroscopie

Un simulateur, uniquement pour l'observation, permettrait déjà à des novices d'apprendre comment manipuler un tel instrument. Mais dans le simulateur, on peut également imaginer insérer les fonctions d'opération chirurgicale.

Le médecin manipulerait l'instrument et au moment venu, pratiquerait une incision. Il serait possible de créer des maladies virtuelles (tumeurs, etc...) et de mettre le chirurgien dans des conditions réelles critiques.

Une autre application du simulateur serait d'exercer le chirurgien pour une opération d'un patient réel. On commencerait par prendre des images par scanner ou passer le gastroscope à l'intérieur afin d'améliorer notre modèle d'organe virtuel et de situer plus précisément la maladie. Ensuite, une fois ces informations intégrées dans l'application, l'utilisateur aurait la possibilité d'exercer les manipulations à effectuer.

Assurément, pour être réaliste, le simulateur doit absolument être interactif et répondre aux mouvements effectués par l'utilisateur (retour de force). Une solution envisagée serait de supprimer le système de vision et de le remplacer par des moteurs incrémentaux indiquant la position du gastroscope. Une solution envisagée serait celle proposée par la Fig. 31.

L'avantage serait de pouvoir agir sur ces moteurs afin d'imposer des forces sur l'instrument manipulé. Ainsi on virtualiserait également les forces en retour lorsque le gastroscope est en contact avec l'organe.

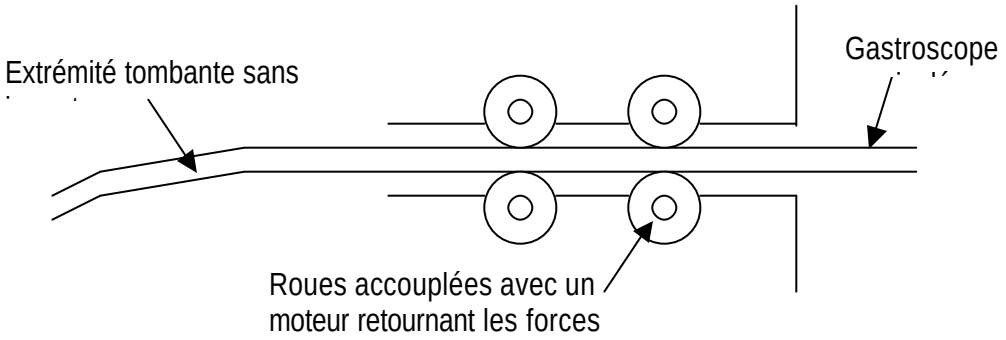


Fig. 31 : Partie mécanique d'un simulateur envisagé

7.7. Directions futures

7.7.1. La vision

Au travers des deux exemples mis au point et de l'expérimentation du système, on remarque tout d'abord que la partie vision n'a pas encore atteint toutes ses limites. L'utilisation d'un marqueur peut être suffisante et même utile suivant quel cas. Mais dans un certain nombre d'applications, son usage peut être encombrant, voire même inutilisable.

Une solution est d'utiliser les algorithmes [SHU00] délinéant les objets et en ressortant les arêtes. Et afin d'améliorer ce système, on peut toujours profiter de la segmentation en profondeur pour en confondre les informations avec celles provenant de la délinéation. Ainsi, il est possible de retrouver l'orientation des objets sans l'adjonction d'un marqueur. Evidemment, la forme de l'objet a ici toute son importance, ainsi que l'environnement dans lequel il évolue.

Une fois séparé de cette contrainte, de nouveaux horizons s'ouvrent pour l'utilisation d'un tel système. On peut citer, par exemple, la reconstruction en 3D de micro tubes de carbones observés à l'aide d'un microscope AFM et leur manipulation simultanément réelle et virtuelle.

De plus, on pourrait également implémenter une reconstruction 3D de l'objet sans connaissance à priori. Cette fonction serait utile quand on désire reconstruire plus précisément un environnement (intérieur de l'estomac) ou créer le déplacement d'un objet non connu.

7.7.2. Le gastroscope

Le simulateur de gastroscopie et l'assistant en opération est une porte ouverte sur un avenir prometteur. Le premier pas serait de mener une enquête parmi le monde compétant sur l'utilité d'un tel système, afin de déterminer l'attente exacte des principaux intéressés.

Après un tour d'horizon sur des produits déjà existant, il n'existe pour l'heure pas de véritable simulateur avec retour de force pour ce genre d'opération. La faisabilité étant quasi-certaine, il serait dommage de ne pas s'engager dans cette voix-là.

8. Conclusions

8.1. Aujourd'hui

Cette plate-forme mise au point est un premier pas dans la combinaison de systèmes de vision travaillant dans un espace tri-dimensionnel avec la mise à jour d'un monde virtuel. La premier pas qui avait été franchi lors d'un ancien projet, reliant la vision à la réalité virtuelle [FLU94], est maintenant dépassé.

Le cahier des charges demandant de mettre au point un tel système et d'en montrer l'usage par quelques exemples a été réalisé. Il s'agissait de relier deux mondes existants, celui observé par un système de vision stéréoscopique couleur, l'autre destiné à la création d'objets 3D.

Mais une simple connexion n'est pas un vrai défi en soi. La partie vision a dû être complétée par un certain nombre de fonctions permettant la détection d'orientation et une première interprétation. Du côté monde virtuel également, plusieurs fonctions ont été rajoutées, comme la détection de collision et l'interprétation des ordres de mouvement reçus. En effet, un ensemble de triangles n'est pas suffisant pour rendre réaliste une scène virtuelle.

Deux exemples démonstratifs des possibilités futures ont été implémentés. Le premier est le tracking d'un objet simple qui se déplace, le robot Pionneer. Le second est la simulation d'un gastroscope qui descend explorer les profondeurs d'un estomac.

Les résultats obtenus par ces différents développements établissant la faisabilité du système montrent que la voie est libre pour progresser dans la combinaison de la vision et de la réalité virtuelle.

8.2. Demain

Mais ce ne sont pour l'instant que des portes entre-ouvertes vers le futur. Avec l'accroissement des performances de l'informatique, la vision permettra de définir la réalité au détail prêt. La réalité virtuelle sera si proche de la réalité que ne pourra plus la distinguer.

Monde réel et réalité virtuelle vont certainement s'interpénétrer, voire même se confondre dans un avenir très proche. La vision en sera logiquement la principale interface.

Un seul danger guette à l'horizon : la manipulation. Le jour où la frontière entre le réel et le virtuel sera insaisissable, on pourra faire dire ce qu'on veut à la réalité. Dans notre société, l'image et encore plus le film, sont considéré comme preuve suffisante de la réalisation d'un événement.

Je n'y vois q'un seul remède : le public doit être continuellement informé sur les progrès de la technologie et rester critique face à ce qu'il reçoit. Notre travail d'ingénieur responsable et encore plus, l'objectif d'une haute école publique, doit autant faire progresser la technologie et la science que de contribuer à l'information du public sur les progrès en cours.

9. Remerciements

Je tiens tout d'abord à remercier le professeur Reymond Clavel pour m'avoir permis d'effectuer le travail de diplôme au sein de son institut et toute l'attention qu'il porte au suivi de ses diplômants. Je remercie également le Dr. Charles Baur, responsable du groupe VRAI, pour les libertés offertes quant aux directions à prendre, ses critiques constructives durant le travail et lors de la lecture de ce rapport.

Un très grand merci va au principal assistant qui m'a suivi et dirigé, Sébastien Grange. Ses compétences, sa disponibilité et son engagement ont été un facteur important dans la réussite de ce projet. Je tiens également en estime tous les autres membres du groupe VRAI, qui par leur personnalité et leur soutien, ont créé un climat de travail très agréable.

Au-delà de l'EPFL et de tous les membres qui la composent, il y a également des personnes chères qui m'ont apporté leur soutien tout au long de mes études. Je citerai tout d'abord mes parents qui ont cru en mes possibilités, m'ont toujours soutenu et encouragé. Merci également à mon amie Carole pour sa patience, son aide et ses encouragements.

Lausanne, le 23 février 2001 Christian Clément

10. Références

[GRA00] GRANGE Sébastien, Vision-based Sensor Fusion for Active Interfaces, EPFL 2000

[CON99] CONTI François, Deformation of Virtual Objects, EPFL 1999

[FLU94] FLÜCKIGER Lorenzo, Robotique, réalité virtuelle et vision, EPFL 1994

[BLO97] BLOW Jonathan, Practical Collision Detection, Computer Game Developer's Conference1997

[BRA98] BRADSKI Gary R., Computer Vision Face Tracking For Use in a Perceptual User Interface, Intel Technology Journal 1998

[KAN99] KANADE T., Virtualized Reality: Digitizing a 3D Time-Varying Event As Is and in Real Time, Springer-Verlag 1999

[MAS97] MASTEN Michael K., Acquisition, Tracking and Pointing, Bellingham 1997

[MER99] MERRITT John O., Stereoscopic Displays and Virtual Reality Systems, Bellingham 1999

[MUL98] MULUKUTLA Prasanth and ROUF Tahsin, Stereoscopic vision and tracking, University of New Brunswick 1998

[RAN98] RANDER Peter, A Multi-Camera Method for 3D Digitization of Dynamic, Real-World Events,
Robotics Institute Technical Report 1998

[RAP98] RAPPAZ Jacques, PICASSO Marco, Introduction à l'analyse numérique, PPUR 1998

[SHU00] SHUFELT Jefferey, Geometric contraints for object detection and delineation, Kluwer Academic Publishers 2000

[STA98] STARK Katrin, Visual tracking of three-dimensional solid objects, Dresden 1998

[VED99] VEDULA S., Appearance-Based Virtual View Generation of Temporally-Varying Events from Multi-Camera Images in the 3D Room , Computer Science Technical Report 1999

[VET94] VETTERLING Wuilliam T. , Numerical recipes : The art of scientific computing, Cambridge University Press 1994

[ALL00] ALLEN Jason, DELPHIGL, http://delphigl.cfxweb.net/

[LIS00] LISCHKE Mike, GLSCENE, http://glscene.cjb.net/

[JIA00] JIAYI'S Chong, BOLT3D, http://www.geocities.com/SiliconValley/Way/7233/

11. Annexes

11.1. Carte d'acquisition vidéo

11.2. Carte graphique pour la reconstruction 3D