UNIVERSITÉ BLAISE PASCAL

RAPPORT

Pour obtenir le diplôme de
Master 2 recherche en Robotique

DIPLOME VISÉ

de : **L' UNIVERSITÉ BLAISE PASCAL**

présentée et soutenue publiquement

par

Mohamedou Amina

le 26 Septembre 2016

préparée dans l'équipe MACCS de l'axe ISPR de l'Institut Pascal :

Reconstruction de mouvements à partir de séquences vidéos

Le jury est composé de :

Mr Mezouar Youssef	Professeur des Universités Institut Français de Mécanique Avancée
Mr Bénoit Thuilot	Maître de conférences de l' Université Blaise Pascal
Mr Sébastien Lengagne	Maître de conférences de l' Université Blaise Pascal

Remerciements

En préambule de ce mémoire, je tiens à adresser mes sincères remerciements à tous les chercheurs de l'Institut Pascale, et surtout le responsable de la formation du master, Mr Mezouar Youcef, qui m'a offert l'opportunité de faire un master recherche. Je tiens à remercier aussi mon encadrant et tuteur de stage Mr Lengagne Sébastien, pour son soutien durant toute la période du stage, il était présent tous le temps pour me fournir de l'aide et répondre à mes questions.
Je tiens aussi à remércier tous les professeurs qui m'ont enseigné durant la période de la formation et surtout mon professeur de vision Mr. Ait Aider, qui n'a pas hésité à me fournir les informations sur la vision par ordinateur.

Un grand merci également aux membres du jury, pour le temps consacré à la correction de ce rapport et dont les remarques seront les bienvenues et contribueront sans doute à l'amélioration de la qualité de ce rapport.

Résumé

Dans ce rapport, nous abordons une méthode de reconstruction de mouvements 3D d'un système robotique ou anthropomorphe générique, à partir d'une ou d'ensembles de caméras synchronisées, filmant la même action sous différents points de vues. On entend par générique, que la méthode proposée n'est pas liée à un modèle de robot, ou de caméra spécifique.

Ce travail s'inscrit dans le cadre d'une architecture logicielle appellée Mogs2 (Motion Generation Software) qui a pour but d'offrir un ensemble d'outils qui facilite l'éxécution des problèmes en lien avec l'optimisation, la visualisation et la génération des modèles, pour faire de la plannification de mouvements. C'est une librairie générique basée sur des plugins.

En calculant le modèle géométrique direct du modèle choisi, et en utilisant un algorithme d'optimisation, on procède au calcul de notre critère représenté par l'erreur de projection des coordonnées tridimensionnelles du modèle dans le repère image.

Nous avons commencé par optimiser l'erreur entre la position de l'effecteur du robot et une position désirée, après validation de cette étape, nous nous sommes amenés à calculer l'erreur de reprojection.

Le programme est fait de telle sorte qu'on puisse modifier le critère, en modifiant une méthode du programme, aussi changer de type de caméra ou charger un modèle à partir d'un fichier XML ou URDF, d'oú la généricité du programme.

Abstract

In this report, a 3D motions construction of robotic system, or anthropomorphic generic is broached. This reconstruction is reached from one or sets of synchronized cameras, covering the same action from different positions. The term generic used to show that our method is not related to a specific robot neither to a specific camera.

This work is part of a software architecture called Mogs2 (Motion Generation Software) which aims to provide a set of tools that make easier the execution of the optimization, visualization and models generation problems in order to attend the motion plannning. This library, aims at being very generic and is based on plugins.

Once the direct geometrical model of the selected robot is calculated, and by using the optimization algorithm, the criterion is calculated, it is represented by the projection error of three-dimensional coordinates of the model in the picture frame.

First of all we began by optimizing the error between the end effector position of the robot and the desired position. After the validation of this step, the projection error is calculated.

The program is organized in way allowing the modification of the criterion, by changing the method of the program, and modifying the type of the camera and the robot's model by modifying an XML or URDF file, hence the generic of this program.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Contexte

L'institut Pascal est une unité mixte de recherche (UMR 6602) créée en 2012. Il est placé sous la tutelle de l'Université Blaise Pascal, du CNRS (Centre National de la Recherche Scientifique) et de SIGMA. Il rassemble au sein d'une même structure plus de 300 personnes (135 enseignants-chercheurs, 3 chercheurs, 33 BIATSS/ITA, 138 doctorants) rattachées à des domaines disciplinaires relevant des Sciences pour l'Ingénieur (automatique, mécanique, électronique, génie des procédés, robotique) et des Sciences Fondamentales (physique, biochimie), réunis dans quatre axes de recherches et un programme transversal :

1. GePEB (Génie des Procédés, Energétique et Biosystèmes) : traite de l'ingénierie des bioprocédés à plusieurs échelles avec une approche biologique, chimique et physique multidisciplinaire des processus. Elle travaille dans le domaine de la biochimie, de l'environnement et l'agro-alimentaire.
2. MMS (Mécanique, Matériaux et Structures) : a pour ambition de relever les défis scientifiques que posent les exigences industrielles dans les domaines de la mécanique et du génie civil, à travers des approches innovantes et pluridisciplinaires. Les activités de recherche couvrent un spectre très large allant de la mécanique des matériaux et structures à la conception des mécanismes et des robots, en passant par les méthodes probabilistes, les techniques de mesure sans contact et les modélisations numériques des systèmes complexes.
3. PHOTON (Photonique, Ondes, Nanomatériaux) : rassemble physiciens et ingénieurs en électronique. Ils couvrent les thèmes de recherche concernant la Nanophotonique et Nanostructures, les Microsystèmes et nanomatériaux, et la Compatibilité électromagnétique.
4. ISPR (Image, Systèmes de Perception, Robotique) : se concentre principalement sur les domaines de la vision par ordinateur et de la robotique. L'équipe est reconnue pour ses applications dans le domaine des véhicules autonomes. Elle dispose d'ailleurs du site PAVIN (Plateforme Auvergnate pour les Véhicules INtelligents)
5. Programme Transversal : fédère des recherches interdisciplinaires de l'Institut Pascal. ll se décompose en 5 actions : Machines et robots intelligents, innovation pour les bio-procédés, matériériaux et modélisations multi-echelles, approche probabilistes, imagerie quantitative.

Ce stage se déroule dans l'axe ISPR qui comprend 4 équipes :

1. ComSee : ComSee s'investit les domaines de la vision par ordinateur et des solutions photogrammétriques pour la localisation 3D, la reconstruction 3D, l'étalonnage des

capteurs et le suivi en temps réel depuis environ 20 ans. Il a été impliqué dans la création de plusieurs sociétés : Poseidon, DxO, ... ComSee se concentre sur deux principaux paradigmes de recherche, Image Matching et Reconstruction 3D.

2. PerSyst : Les principaux travaux de l'équipe sont dédiés à de nouvelles approches de localisation prenant en compte à la fois des capteurs de modélisation, du SLAM multi-modalités (lidar, vision, radar, ...), du filtrage optimal, à la localisation non centralisée et multi-robots / multi-capteurs SLAM pour la robotique mobile et en pelotons et à la cartographie des scènes 3D, modélisation et compréhension afin de conduire à une meilleure compréhension des scènes dynamiques.
3. DREAM : Cette équipe se concentre sur la recherche de pointe dans la modélisation et l'analyse architecturale, des caméras intelligentes, middleware et la technologie multi-core. L'expertise du groupe couvre de la conception de concepts à diverses technologies de mise en œuvre pour la construction de prototypes de travail réalistes dans des domaines d'application tels que le traitement d'image et la vision par ordinateur.
4. MACCS : Nos recherchent portent principalement sur la modélisation et le contrôle de robots mobiles et manipulateurs, la vision des robots, la vision active, la maitrise visuelle et les comportements d'anticipation. Les applications de nos recherches sont liées à la fabrication robotisée et aux systèmes de transport intelligents. L'équipe travaille sur deux thématiques : les véhicules autonomes (AV) et la modélisation, apprentissage et perception pour la commande (MAPC).

Ce stage se déroule dans l'équipe MACCS.

Chapitre 2

Introduction

L'objectif du travail présenté dans ce rapport, est de créer une méthode de reconstuction de mouvements générique, précise et rapide. En se basant sur l'optimisation, l'objectif est de minimiser l'erreur de reprojection des points tridimensionnelles.

Ce rapport est divisé en deux parties essentielles, une étude bibliographique qui regroupe une introduction sur la vision par ordinateur et ses applications, puis une analyse des méthodes existantes de la reconstruction de mouvements, les avantages et inconvénients de chaque approche. Une partie developpement de l'approche utilisée, dans laquelle nous citons dans une première partie les outils nécessaires à la mise en place du programme. Ensuite nous traitons la méthode utilisée pour le calcul du modèle géométrique direct du robot, permettant le calcul de la position de son organe terminal. Afin de pouvoir optimiser l'erreur entre cette position et une position désirée, nous utilisons un algorithme d'optimisation ipopt que nous détaillerons par la suite. Une partie qui traite la modélisation de la caméra, le calcul des différentes transformations du repère monde au repère caméra, et la méthode de calcul de la matrice de projection.

Le programme est conçu de telle sorte qu'il soit adapter à n'importe quel type de caméra. En considérant que la caméra utilisée est étalonnée c'est à dire, on connaît sa matrice de calibration et sa pose par rapport au repère monde, en insérant un fichier contenant ses paramètres intrinsèques et extrinsèques, nous devons pouvoir calculer la projection du point 3D dans le repère image. De même pour le critère, il suffit de modifier une classe du programme pour que le critère soit changer, chose que nous avons fait, afin de minimiser l'erreur de reprojection de la position 3D du robot.

L'objectif est de pouvoir généraliser le principe de la reconstruction 3D, et générer un programme qui peut être utilisé avec n'importe quel type de caméras ou de modèles juste en changant des fichiers qui contiennent les données des modèles et les paramètres de ou des caméras d'où la généricité du programme. la figure 2.1 illustre l'approche utilisée.

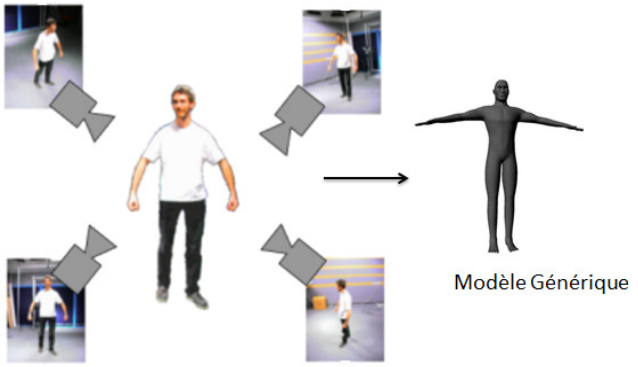

FIGURE 2.1 – Représentation de l'approche, généricité

Enfin, un apperçu des résultats obtenus et une conclusion qui contiendra les avantages et les incovients de notre solution.

Chapitre 3

État de l'art

Introduction Générale

La reconstruction de mouvements représente un problème majeur dans la vision par ordinateur. Elle consiste à estimer la géométrie tridimensionnelle du modèle ou de la scène filmée ainsi que la position et l'orientation de la caméra utilisée. Dans ce cadre de nouvelles interactions homme-machine, basées sur des images, ont été implémentées pour répondre à ce besoin, allant de la simple détection de mouvements dans une scène à la capture tridimensionnelle du mouvement d'un personnage, à la reconstruction de ses mouvements. Il s'agit d'un problème de localisation du capteur et de cartographie de l'environnement. Pour répondre à ce problème plusieurs méthodes ont été développées, des méthodes en temps réel [20], qui exigent l'utilisation ou l'implémentation des procédés optiques qui nécessitent l'utilisation de caméras spécifiques, soit associées à des marqueurs fixés aux articulations des personnes à suivre, soit sans marqueurs et cela exige l'implémentation des approches de reconnaissance d'objets ou de formes dont on donnera plus de détails dans ce qui suit. Des méthodes qui ne sont pas en temps réel [29], et qui sont génériques, dont fait partie notre approche.

Les applications de la reconstruction de mouvements sont multiples, on peut citer la vidéo-surveillance, la création de modèles 3D numériques, la recherche d'informations dans des bases d'images ou de films vidéos, la création de personnage virtuels à l'apparence et aux mouvements réalistes, comme dans les jeux vidéos etc...

Les procédés d'acquisition de mouvements, tentent d'apporter une solution pour que les mouvements reproduis soient proches de la réalité, pour ensuite pouvoir transférer ces mouvements à des personnages virtuels ou des avatars.

La réussite commerciale des consoles de salon Wii, ou encore du périphérique EyeToy sont des exemples concrets du capture de mouvements pour l'interaction homme-machine.

Dans cette partie, nous abordons les principales méthodes utilisées pour la reconstruction de mouvements 3D en temps réel.

3.1 Capture de mouvements : Principales méthodes

Le domaine de recherche de la capture et suivi de mouvements humains à partir de caméra, reste un sujet de recherche actif et alimente des domaines connexes tels que la création de personnages virtuels à l'apparence et aux mouvements réalistes, qu'on trouve surtout dans la production de jeux vidéos et des films.
Pour répondre à ce besoin, des solutions commerciales existent aujourd'hui dont on peut

identifier les entreprises suivantes :
Microsoft[4], elle utilise une caméra Kinect RGB-D montrée dans la figure 3.1 pour reconstruire la squelette de la personne qui doit être toujours en face de la caméra. [22]

FIGURE 3.1 – Kinect [4]

Capture Lab, qui représente le plus grand studio et le mieux equipé des laboratoires de capture de mouvements dans le monde.[8], la figure 3.2 est une image du studio.

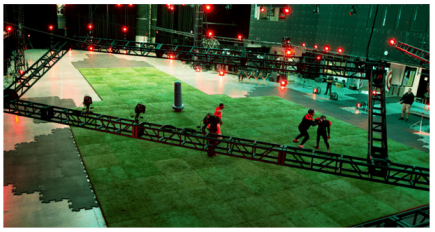

FIGURE 3.2 – Studio capture Lab pour la capture motion

Vicon, équipe les studio qui utilise cette technologie, comme capture Lab, elle est équipé de plus de 132 caméras vicon mais aussi des derniers logiciels de capture de mouvements, le Vicon Blade.[5]

Captury Studio, qui représente une solution polyvalente sans marqueurs, elle capture les mouvements des humains et aussi des animaux, c'est un logiciel qui peut capturer en détails la surface d'un modèle en trois dimensions

Dans la suite, nous présentons un état de l'art non exhaustif, qui relate les principales méthodes liées aux captures et reconstruction de mouvements.

3.1.1 Systèmes optiques à base de marqueurs

Les systèmes optiques à base de marqueurs utilisent le positionnement des marqueurs sur les articulations du sujet à suivre. On distingue deux types de marqueurs, lumino-réfléchissants qu'on appelle marqueurs passifs, et des marqueurs actifs composés d'un matériau hautement réflecteur ou des diodes Electro-luminescentes, comme montré dans la figure 3.3 Parmis les entreprises qui utilisent ces marqueurs on peut citer, Vicon, Optitrack, Motion Analysis et Phasespace.

Des caméras infrarouges sont utilisées pour filmer et suivre les personnes en mouvement, elles captent le signal émis ou réfléchit par les marqueurs, c'est pour cela qu'ils faut un nombre élevé de caméras pour garantir un suivi non interrompu, le système suit la position 3D de chaque marqueur, par ce qu'on appelle la triangulation, qui nécessite la connaissance de cette position par au moins deux caméras à un instant donné.

La reconstruction des personnages virtuels sont parmis les applications de cette méthode comme montré dans la figure 3.4.[9]

FIGURE 3.3 – Capture de mouvements avec marqueurs passifs Vicon(à gauche), actifs Visualeyez(à droite) [22]

FIGURE 3.4 – Capture de mouvements avec marqueurs pour reconstruire un personnage virtuel

Le problème principal de cette méthode, se manifeste dans la sensibilité pour les obstacles qui se présentent entre les marqueurs et les caméras, les rayons infrarouges ne pourrons pas traverser les obstacles, donc ne seront pas réfléchis par les marqueurs.

3.1.2 Systèmes optiques à base de capteurs

Le principe de cette approche est à peu près le même qu'avec les marqueurs, sauf que ces derniers vont être remplacés par des capteurs gyroscopiques et inertiels qui permettent de capter en temps réel, l'angle et la position de la partie du corps où ils sont placés. Cette méthode contrairement à celle avec les marqueurs, ne réprésente pas de problème d'obstacle, mais reste malgré cela moins précise. Une solution commerciale utilisant cette approche existent actuellement notamment l'entreprise Tea [7] et XSens [6].

3.1.3 Systèmes optiques sans marqueurs

Cette méthode repose sur l'estimation du mouvement de l'humain à partir de plusieurs images provenant d'une ou plusieurs caméras. Les approches générales qu'utilisent ces méthodes se basent sur la segmentation de l'image pour en extraire des primitives, telles que les contours, ou nuage de points 3D par exemple, puis le suivi est réalisé pour trouver l'information de la silhouette qui correspond à la personne de la scène.

Il existe certains facteurs qui font que ce problème est difficile. Premièrement le corps humain est une structure complexe et très articulée, même les modèles cinématiques simples du corps humain contiennent plus de trente degrés de liberté. Ce fait rend la recherche de solution difficile, surtout lorsque l'on souhaite acquérir les mouvements fins du corps humain. Deuxièment, la mise en correspondance des point 3D du modèle à partir des points 2D, ne définit pas une relation bijective, c'est à dire qu'on peut trouver les mêmes résultats avec de différentes configuration du modèle surtout quand le nombre de caméras utilisé est faible.

3.2 Principales méthodes de reconstruction de mouvements 3D optiques

La reconstruction de mouvements 3D optiques, peut être approvisionnée par des méthodes dites passives ou actives. La première méthode extrait de l'informations 3D à partir de la scène et des objets dans les images, elle ne nécessite aucune source de lumière, contrairement à la méthode active, qui quand à elle, nécessite une projection de lumière sur la scène.

La figure 3.5 présente une classification non-exhaustive des méthodes actives et passives, dans ce qui suit nous citerons le fonctionnement de chaque technique.

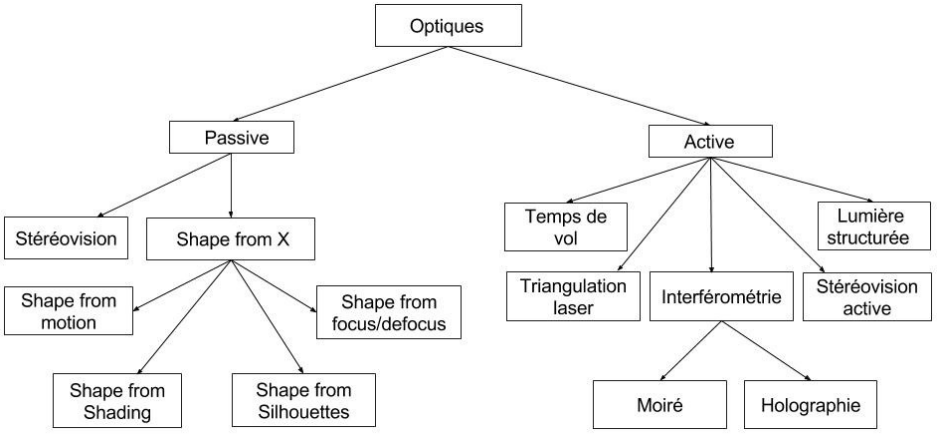

FIGURE 3.5 – Classification non-exhaustive des techniques optiques

3.2.1 Vision passive

Les méthodes de la vision passive peuvent être soit à base d'une seule caméra, ou plusieurs caméra. Les approches de reconstruction de mouvements qui utilisent une seule caméra avec des images successives de la scène, se basent sur certaines caractéristiques de la scène pour déterminer l'information de profondeur, cette information peut être le mouvement de l'objet (Shape form motion), elle peut être aussi par une analyse de l'ombrage ou l'éclairement (Shape from shading), la déformation de la texture (Shape from texture), par focalisation ou défocalisation ou la mise au point de l'objectif utilisé (Shape from focus/defocus). La profondeur peut être aussi retrouvée en utilisant deux ou plusieurs caméras, c'est le cas de la stéréovision, par photogrammétrie ou par une analyse des silhouettes visuelles des objets présents dans la scène créées par différentes acquisitions 2D prise de différents points de vue (Shape from silhouette).

3.2.1.1 Stéréovision

Le processus de la Stéréoscopie est souvent décrit en trois étapes :
- Prétraitement (Calibration, Acquisition, Rectification)
- Appariement (Mise en correspondance)
- Reconstruction (Triangulation).[23]

La difficulté principale de la technique stéréovision provient du problème de la mise en correspondance, elle consiste à estimer la position 3D par appariement, c'est à dire trouver

les points dans chaque image qui correspondent aux mêmes points dans la scène 3D, puis faire la triangulation pour pouvoir déterminer la position 3D de la scène correspondante.

Il existe des méthodes pour estimer la mise en correspondance, comme la méthode qui recherche les correspondance pour chaque pixels par des approches d'auto-corrélation normalisées, c'est une méthode sensible au bruit et aux conditions d'éclairage dans chaque vues.

3.2.1.2 Shape From Shading

On peut traduire ce terme "Shape From Shading" par forme à partir d'ombrage. toutefois, le nom anglais nous paraît plus intuitif et est très utilisé, même dans la littérature française. L'approche Shape from Shading estime la forme d'objets éclairés à partir des variations graduelles de l'éclairement dans l'image, la profondeur est trouvée en analysant pour chaque point 3D de la scène, sa projection pixelliques dans le repère image correspondant à un niveau de gris, l'éclairement d'un point 3D de la surface de la scène dépend de la nature de la scène, qui doit être parfaitement lambertienne (i.e. émet ou réfléchit un rayonnement de la même manière quelle que soit la direction) et de la forme de la surface, elle dépend aussi de la projection de l'image sur le capteur.

La méthode de Shape From Shading peut par exemple être utilisée pour obtenir une carte d'élévation de la surface lunaire à partir d'une photo. On trouve aussi des cas d'utilisation sur des photos de visage voir figure 3.6.

Du fait des conditions restrictives d'utilisation de cette approche, elle ne paraît pas du tout adaptée pour traiter le cas d'une surface très fortement réfléchissante et avec des parties transparentes.[36].

FIGURE 3.6 – Shape from shading

3.2.1.3 Shape From Texture

En général les méthodes d'analyse de l'éclairement ne prennent pas en compte les variations de réflectance de la surface à estimer, c'est-à-dire les changements de couleur dus aux matériaux. Dans le monde réel, les changements de réflectance participent à l'apparence perçue des matériaux : le bois, la fourrure, l'herbe, les éraflures et les bosselures conduisent à des variations de l'éclairement perçu, mais ne relèvent pas du champ d'application des approches Shape From Shading, ce groupe de phénomènes est généralement appelé texture. Les approches Shape from texture estiment la forme des objets en fonction des variations observées dans la texture, ainsi les objets à construire doivent être texturés que ce soit naturellement ou artificiellement.[24]

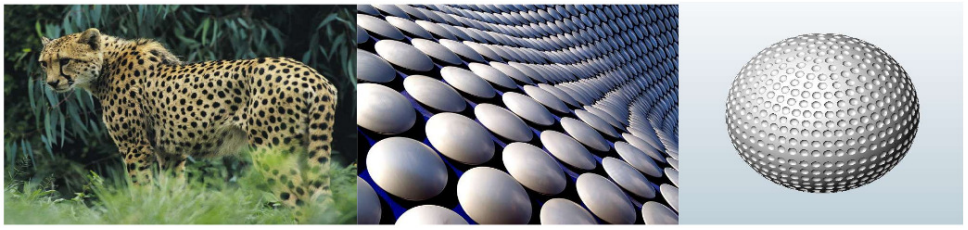

FIGURE 3.7 – Exemples de textures

3.2.1.4 Shape from motion

L'approche shape from motion ou la forme à partir du mouvement en français, se base sur l'analyse du mouvement d'un objet sur deux ou plusieurs trames successives capturées par une seule caméra, afin d'estimer la profondeur de l'objet.[26] Cette technique exploite le mouvement relatif entre la caméra et la scène, similaire à la technique stéréo, la reconstruction 3D d'un objet dans la scène à partir d'une séquence vidéo acquise par une seule caméra peut être considérée comme un problème stéréo, il suffit d'estimer la disparité entre chaque trame et sa suivante. En d'autres termes, le mouvement d'un objet en face d'une seule caméra fixe peut être considéré comme le mouvement d'une caméra en face d'un objet fixe. Cette approche ne nécessite pas une étape d'étalonnage préalable de la caméra car les paramètres intrinsèques de la caméra sont estimés en ligne. Cette méthode a l'inconvénient d'être très sensible aux bruits, car le fait que le déplacement entre deux trames successives est très faible engendre une instabilité dans l'estimation de la disparité.[14][32]

3.2.1.5 Shape from Focus/Defocus

L'approche shape from Focus/ defocus représente un problème de l'estimation de la surface 3D d'une scène à partir d'un ensemble de deux ou plusieurs images de cette scène. Elle utilise les propriés intrinsèques de l'objectif d'une caméra pour extraire des paramètres de profondeur d'une scène, à partir de deux méthodes : une dite la focalisation, Shape from Focus et l'autre défocalisation, Shape from defocus, pour la premère méthode la mise au point s'éffectue en chacun des points de l'image avec une focale propre à chaque point de l'image, pour la deuxième technique, un nombre fini d'acquisition d'images à différentes focales est réalisé,

Les images sont obtenues en modifiant les paramètres de la caméra, généralement le réglage de focale ou la position axiale du plan d'image, et sont prises du même point de vue, un nombre fini d'acquisitions d'images à différentes focales est réalisé.

Ainsi, une fonction de flou est caractérisée pour chacun des points, permettant de déterminer leur profondeur. Les propriétés de l'ouverture de l'objectif et la profondeur de champ du système optique d'acquisition d'image doivent être connues, si la profondeur de champ est finie, le flou dépend de la distance entre l'objet et la caméra.

Les systèmes de reconstruction 3D basés sur la focalisation sont limités en rapidité puisqu'ils nécessitent plusieurs prises de vues avec différentes mises au point. Ils souffrent également d'un manque de précision, en fait, la profondeur de champ est liée à l'agrandissement et au système optique d'observation et l'agrandissement varie avec la mise au point.

En plus, il est très difficile d'extraire une information de profondeur suffisamment précise de la défocalisation. Même si certains travaux ont proposé une approche temps

réel, ce type d'approche reste cependant peu robuste aux bruits ambiants et sensible au manque de texture des objets à reconstruire. [13].

3.2.1.6 Shape From Silhouette

Shape-From-Silhouette est une méthode qui estime, à partir de plusieurs vues, l'enveloppe visuelle des objets d'intérêts, qui représente une estimation englobante de leur forme 3D comme montré dans la figure 3.8.

Baumgart a été le premier en 1974 à utiliser cette méthode [10]. A partir d'un ensemble de n vues, l'information de silhouette, est extraite des images capturées par une approche dite d'extraction de silhouette. La forme produite fournit un volume englobant des objets d'intérêts.

Les approches Shape-From-Silhouette ont été utilisées pour diverses applications, telles que la surveillance de foule, la modélisation 3D ainsi que l'acquisition de mouvements sans marqueurs. [24].

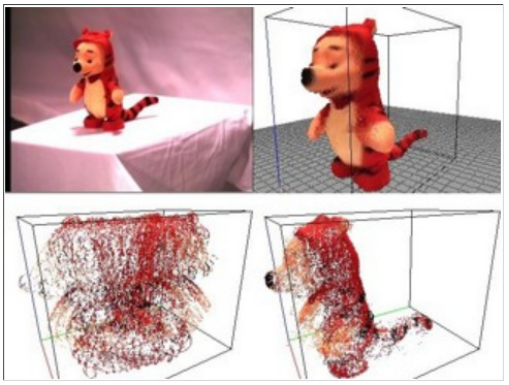

FIGURE 3.8 – Shape From Silhouette

3.2.1.7 Différentes Caractéristiques des approches passives

Approches Passives	Nbre de caméras	Nbre images/caméras	Image de profondeur	information de profondeur	Orientation de la surface
stéréovision	2	1	X	X	
Shape from motion	1	2	X	X	
Shape from focus/defocus	1	2			
Shape from Shading	1	1	X		X
Shape from Silhouette	N>2	1	X	X	
Shape from Texture	1	1			X

TABLE 3.1 – Différentes Caractéristiques des approches passives [27]

Le tableau 3.1 propose les différentes caractéristiques des approches passives, relatives au nombre de caméras utilisées pour chaque approche, image de profondeur c'est à dire si l'approche peut fournir une information de profondeur ou non, et aussi si elle est fournie directement ou non. L'orientation de la surface, permet de montrer laquelle des approches fournie des cartes d'orientation de la surface comme information de profondeur.

3.2.2 Vision actives

Pour répondre aux difficultés des approches de stéréovision liées à la mise en correspondance, certaines méthodes dites "actives" projettent de l'énergie lumineuse dans la scène. En remplaçant l'une des caméras d'un système de stéréovision par un périphérique qui projette un rayon sur la scène, afin de détécter sa réflexion et de pouvoir prospecter un objet, la mise en correspondance revient donc à calculer les correspondances entre les pixels et les points du rayon projeté.

Parmis les différentes sources de rayonnements utilisés on peut citer : les rayons X, lumières blanches, faisceaux laser, ultrasons, infrarouges. L'information 3D est alors obtenue par estimation du temps de vol des rayons lumineux refléter par des objets, soit par le principe de la triangulation optique, soit par le principe de l'interférométrie ou la moiré.

La principale difficulté de ce type d'approche provient du choix du motif à projeter, ainsi que de la stratégie de codage de ce motif. [19]

3.2.2.1 Lumière structurée

Cette approche utilise une caméra et un projecteur, comme montré sur la figure 3.9 et a le même principe de la triangulation optique que pour la stéréovision et la triangulation laser, un motif connu est projecté sur la scène, une grille souvent ou barres horizontales, la déformation de ce motif lorsqu'il heurte la surface permet le calcul de la profondeur et d'extraire des informations sur les objets présents dans la scène. [12]

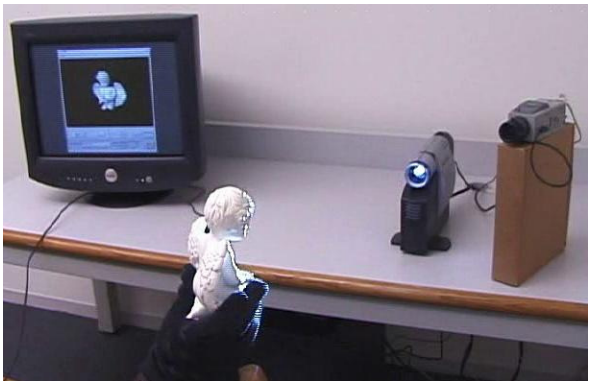

FIGURE 3.9 – Lumière structurée

3.2.2.2 Interférométrie et Moiré

Les méthodes interférométriques s'appuient sur la variation de phase entre deux ou plusieurs signaux périodiques de même fréquence, associés par exemple à l'état de référence et déformé d'un objet. Les signaux sont obtenus par l'interaction entre la surface de l'objet et des faisceaux incidents. La corrélation entre ces différents signaux permet de déterminer les déplacements surfaciques relatifs entre les différents états. Les différentes techniques

se distinguent par rapport à l'interaction entre la surface de l'objet et le faisceau incident, mais aussi par rapport au choix de la nature des faisceaux. [16]

3.2.2.3 Caméra 2,5D à temps de vol

Les caméras temps de vol, représenté sur la figure 3.10 sont des capteurs actifs qui fournissent des images de profondeur en temps réel, ils envoient un signal optique proche infrarouge, le signal réfléchi par les objects de la scène est ensuite détecté par le capteur qui calcul la profondeur. Il existe un premier type de temps de vol qui se base sur des impulsions, il consiste à mesurer le temps que met le signal pour effectuer le trajet entre l'objet et la caméra. Un autre type qui se base sur des ondes de modulation continues, il consiste à mesurer le déphasage entre le signal émis et celui réfléchi par démodulation synchrone du signal réfléchi. le principe de ce dernier type est basé sur la détection et la sauvegarde des charges photo-électriques par synchronisation des électrodes des pixels.[17] Les caméras à temps de vol sont beaucoup utilisées, elles ont l'avantage d'acquérir les images à haute fréquence, de consommer peu d'énergie et d'avoir un plus faible poids. [15]

FIGURE 3.10 – Exemple de caméra temps de vol

3.2.2.4 Différentes Caractéristiques des approches actives

Approches Actives	Nbre de caméras	Nbre images/caméras	Image de profondeur	Directe	Orientation de la surface
Time-Of-Flight	1	1	X	X	
Lumière structurée	1	N>=3	X	X	
Stéréovision active	2	N>=1	X	X	
Active shape from defocus	1	N>=2			X

TABLE 3.2 – Différentes Caractéristiques des approches actives

Les approches actives fournissent généralement une meilleure précision que les approches passives. Le tableau 3.2 propose les différentes caractéristiques de quelques approches actives. Il indique le nombre de caméras utilisées, le nombre d'images utilisées par chaque caméra. L'information de profondeur qui permet de distinguer les approches qui fournissent une image de profondeur comme résultat ou non. La directe qui signale les approches qui fournissent directement une information de profondeur ou non. Enfin, l'orientation de la surface quiu permet de discerner les approches qui fournissent des cartes d'orientation de la surface comme information de profondeur. [27]

3.3 Tracking

Le tracking ou le suivi du corps humain, a représenté un grand sujet de recherche pendant ces dernières années. Cette technique, exploite de différentes primitives pour l'estimation de la pose, telleque la texture, le mouvement, la silhouette, qui sont citées au préalable. Cette technique englobe les différentes catégories suivantes : avec ou sans modèle, avec ou sans marqueurs [11], et prédiction et apprentissage.

Les catégories avec marqueurs est largement utilisée pour le suivi des mouvements de la personne en 3D, elle utilise le même principe cité dans la partie capture de mouvements avec marqueurs. Quand à la deuxième catégorie, le suivi de mouvements sans marqueurs représente un défi majeur, en abscence de marqueurs il faut se baser sur des caractéristiques extraites à partir des images. Parmis ces caractéristiques nous citons, la forme, la couleur, la silhouette etc.

Les catégories avec ou sans modèle, la première catégorie avec modèle consiste à faire correspondre le modèle à l'image [28]. Et la catégorie sans modèle se base sur l'apprentissage d'une correspondance entre une image et une pose 3D, d'autres modèles se basent sur des bases de données pour stocker une collection de modèle 3D et interpolent entre les poses disponibles pour trouver celle qui correspond le plus à l'image réelle.[25].

Et enfin, prédiction et apprentissage c'est une autres méthode qui se base sur des algorithmes qui permettent de prédire la pose et de réaliser des corrélations sur ses prédictions, on peut citer, la méthode d'estimation dynamique filtre bayésien comme le filtre de Kalman et le filtrage particulaire.[30][33].

Conclusion

Les approches de captures de mouvements monoculaires consistent à recupérer les paramètres du modèle observé à partir d'informations 2D de la scène, cependant ces rapproches présentent quelques limitations par rapport au temps d'exécution et présentent aussi l'inconvénient de ne pas fournir des résultats correctes surtout en présence d'occultations. L'utilisation des approches multicaméras, permettent de compenser le problème des occultations, par les nombres de d'informations apportées par les autres caméras. La reconstruction de mouvements 3D, quand à elle représente quelques contraintes liées aux placement des personnes, ces dernières doivent être entièrement visibles par tous les caméras. Aisin toute erreur d'extraction de primitives influe directement sur la qualité de la reconstruction. L'approche que nous présentons dans ce rapport, a comme objectif principales de répondre à quelques critères liés à la vitesse d'éxécution, la précision aussi le matériel utilisé de telle sorte qui soit accessible c'est à dire qui ne nécessite pas de matériel spécifique en dehors de quelques webcams.

Chapitre 4

Développement

Dans ce chapitre, nous abordons la méthode utilisée pour la reconstruction de mouvements à partir de séquences vidéos. Nous avons implémenté une méthode qui n'est pas en temps réel mais qui peut être utilisée pour différents modèles, c'est à dire une méthode générique.

4.1 Objectif de l'approche

L'objectif principal visé par notre approche est la généricité, c'est à dire, qu'elle ne dépend pas d'un modèle ou type de caméra spécifique. On cherche à créer dans un premier temps un programme qui répond à ce critère, c'est à dire qui soit implémenté de telle façon qu'on puisse faire sur un même programme le test avec différents modèles robotiques, humanoîdes, etc ... et aussi avec n'importe qu'elle type de caméra.

Il s'agit d'un programme codé en langage C++ (programmation orientée objet), nous nous sommes basé sur le concept de l'héritage, et des classes virtuelles afin de donner au code un aspect compréhensif et facile à manipuler. Ensuite il sera possible de charger les informations d'une nouvelle caméra sur un fichier pour tester le programme avec ses paramètres, il est de même pour ce qui est du modèle d'où la généricité du programme.

4.2 La démarche suivie

Nous détaillons dans cette partie, les outils mis en place pour la réalisation du critère demandé qui est la généricité du programme. Ayant calculé la position de l'effecteur du robot, l'objectif était de pouvoir minimiser l'erreur entre cette position et une position désirée, et cela à l'aide d'un algorithme d'optimisation qui dans notre cas s'agit de IpOpt.

Ipopt (Interior Point OPTimizer) est un logiciel d'optimisation non linéaire à grande échelle. Il a l'avantage d'être facile à implémenter et à une grande vitesse d'éxecution. Il a été conçu pour trouver des solutions locales des problèmes d'optimisation, pour plus de détails, se référer au document [34].

IpOpt, nécessite le calcul du gradient du critère, pour se faire, nous avons procédé à l'approximation du gradient par ce qu'on appelle la méthode des différences finie, mais pour avoir un temps de calcul satisfaisant, nous avons testé les différents algorithmes de différentiation automatique qui garantissent un résultat efficace et un temps de calcul réduit, Adol-C, Adept, CppAD sont parmis ces algorithmes qui garantissent le calcul des dérivées premieres et supérieures des fonctions vectorielles.

Nous avons choisi l'algorithme ADOL-C (Automatic Differentiation by OverLoading in C++), qui convenait le plus à notre programme. Pour finir nous avons créé une classe qui calcule la projection du point 3D du robot, à partir d'un fichier XML qui englobe les paramètres de la caméra choisie, et qu'on considère étalonnée et une estimation de la projection du point 3D sur l'image.

4.3 Notions Théoriques

Avant de présenter notre programme, il sera utile de donner un aperçu sur les notions théoriques dont nous avons besoin pour la réalisation de ce code.

Il s'agit d'un programme C++ (orienté objet), pour pouvoir l'utiliser et le modifier, cela nécessite la connaissance de certains concepts fondamenteaux de la programmation orientée objet :

- Le concept de la modélisation à partir des notions de classes et l'instanciation de ces classes.
- Le concept des classes virtuelles.
- Le concept des classes templatées.
- Le concept d'action, c'est à dire, l'envoi de messages et de méthodes à l'intérieur des objects
- Le concept de construction par réutilisation en utilisant la notion d'héritage.

4.3.1 Qu'est ce qu'une classe templatée

Les templates sont la base de la généricité en C++. Les templates représentent des modèles génériques qui à partir d'un ou de plusieurs paramètres permettent de créer automatiquement des classes. Le fait de fournir un paramètre à un modèle générique s'appelle la spécialisation. Elle aboutit en effet à la création d'un code spécialisé pour un type donné à partir d'un modèle générique. Ces modèles manipulent généralement un type abstrait qui est remplacé par un vrai type C++ au moment de la spécialisation. Ce type abstrait est fourni sous forme de paramètre template qui peut être un type C++, une valeur (entier, pointeur...) ou même un autre template. La spécialisation d'un template est transparente et invisible. Elle est effectuée lors de la compilation, de manière interne au compilateur, en fonction des arguments donnés au template. [1]

4.3.2 Qu'est ce qu'une classe virtuelle, classe abstraite

Une méthode virtuelle pure est une méthode qui est déclarée mais non définie dans une classe, elle est définie dans une des classes dérivées de cette classe. Une classe abstraite est une classe comportant au moins une méthode virtuelle pure. Étant donné que les classes abstraites ont des méthodes non définies, il est impossible d'instancier des objets pour ces classes, en revanche, on pourra les référencer avec des pointeurs, le mécanisme des méthodes virtuelles pures et des classes abstraites permet de créer des classes de base contenant toutes les caractéristiques d'un ensemble de classes dérivées, pour pouvoir les manipuler avec un unique type de pointeur. En effet, les pointeurs des classes dérivées sont compatibles avec les pointeurs des classes de base, on pourra donc référencer les classes dérivées avec des pointeurs sur les classes de base, donc avec un unique type sous-jacent, celui de la classe de base. Cependant, les méthodes des classes dérivées doivent exister dans la classe de base pour pouvoir être accessibles à travers le pointeur sur la classe

de base, c'est ici que les méthodes virtuelles pures apparaissent. Elles forment un moule pour les méthodes des classes dérivées, qui les définissent, bien entendu, il faut que ces méthodes soient déclarées virtuelles, puisque l'accès se fait avec un pointeur de classe de base et qu'il faut que ce soit la méthode de la classe réelle de l'objet, c'est-à-dire la classe dérivée, qui soit appelée. Pour déclarer une méthode virtuelle pure dans une classe, il suffit de faire suivre sa déclaration de (=0), la fonction doit également être déclarée virtuelle de la fome : virtual type nom(paramètres) =0 ;

=0 signifie qu'il n'y a pas d'implémentation de cette méthode dans cette classe.[2]

4.4 Structure du code

Le travail réalisé s'inscrit dans le cadre d'une architecture logicielle appellée Mogs2. La librairie Mogs2 (Motion Generation Software) utilisée, a pour but d'offrir un ensemble d'outils qui facilite l'éxécution des problèmes en lien avec l'optimisation, la visualisation et la génération des modèles, pour faire de la plannification de mouvements. C'est une librairie générique basée sur des plugins. La structure du code utilsé est comme montré sur la figure 4.1

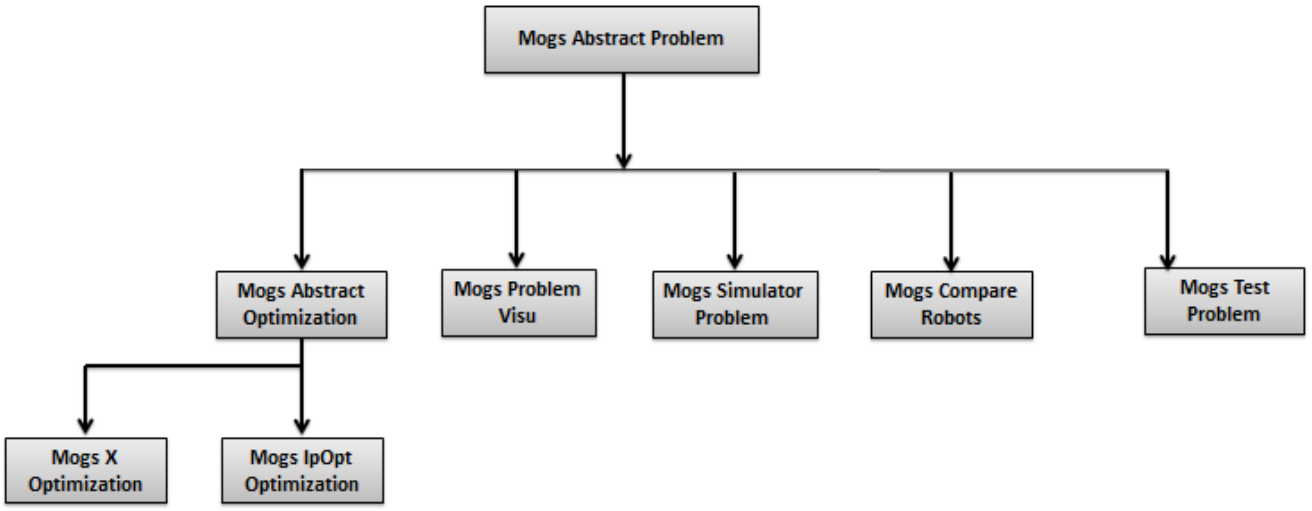

FIGURE 4.1 – Structure du code utilisé avec la librairie mogs2

La Figure 4.1, montre les différentes classes qui héritent de la classe Mogs Abstract Problem. Cette dernière classe, permet la lecture des différents problèmes dans le programme, et la sauvegarde des résultats. La classe Mogs_Abstract_Optimisation permet la lecture et la résolution du problème d'optimisation, on peut avoir d'autres classes qui héritent de la classe Mogs_Abstract_Optimisation, mais en utilisant d'autres algorithmes d'optimisation pour la résolution du critère.

4.4.1 La librairie RBDL

RBDL : the Rigid Body Dynamics Library est une librairie contenant un code très efficace permettant de trouver les modèles géométrique, dynamiques directs et inverses

pour de différentes chaînes cinématiques.[3] Elle comprend : - Un algorithme récursive de Newton Euler (RNEA)
- Algorithme corps rigide composite(CRBA)
- Algorithme corps articulé (ABA).
Le modèle du robot peut être chargé à partir des fichiers URDF. ([3]). Pour un fonctionnement performant de cette librairie, nous avons utilisé aussi la dernière version de la librairie mathématique Eigen.

4.5 Description du logiciel utilisé : Ipopt

IPOPT : Interior Point OPTimizer, est un algorithme conçu pour l'optimisation des problèmes non linéaires, il est designé pour trouver une solution locale à des problèmes d'optimisation mathématique de la forme suivante :

$$min f(x)$$

$$x\, in\, \mathbb{R}^n$$

$$s.t\ g\_l <= g(x) <= g\_u$$

$$x\_l <= x <= x\_u$$

n : le nombre de variables dans le problème.
x : le vecteur de variables du problème.
f(x) : $\mathbb{R}^n$ -> $\mathbb{R}$ est la fonction objective.
g(x) : $\mathbb{R}^n$ -> $\mathbb{R}^m$ représente la fonction des contraintes.
Les vecteurs g_L et g_U représentent les limites des contraintes.
Les vecteurs x_L et x_U sont les bornes des variables x.
Les fonctions f(x) et g(x) peuvent être non linéaires et non convexes mais doivent être continûment différentiable.

Étant donné une fonction $\mathbf{f} : \mathbf{A} \to \mathbb{R}$ définie sur un ensemble $\mathbf{A}$ à valeurs dans l'ensemble $\mathbb{R}$ des nombres réels, trouver un élément $\overline{x}$ de $\mathbf{A}$ tel que $\mathbf{f}(\overline{x}) <= \mathbf{f(x)}$ pour tous les $\mathbf{x}$ dans $\mathbf{A}$.

On dit que l'on cherche à minimiser la fonction $\mathbf{f}$ sur l'ensemble $\mathbf{A}$. La fonction $\mathbf{f}$ porte le nom de : fonction-coût, fonction-objectif ou critère. Le point $\overline{x}$ est appelé solution du problème d'optimisation (ou minimum).

On note : $argmin(f(x) : x \in \mathbf{A})$ l'ensemble des solutions du problème.

L'ensemble $f(\mathbf{A}) := (f(x) : x \in \mathbf{A})$ est une partie de $\mathbb{R}$ et sa borne inférieure $inf\mathbf{f(A)}$ est appelée la valeur optimale du problème, cette valeur optimale est atteinte si, et seulement si, le problème d'optimisation a une solution.

4.6 Le Calcul du gradient

L'une des méthodes d'algorithme d'optimisation utilisé requiert l'évaluation du gradient de la fonction coût. Le calcul du gradient peut être fait par les méthodes suivantes

1. Une dérivée analytique

2. Une méthode des différences finie
3. Une méthode de différentiation automatique.

Nous avons approximé le gradient du critère par une méthode dite méthode des différences finies, ensuite nous avons testé quelques librairies des différentiations automatiques, afin de choisir la plus rapide au niveau de la vitesse d'exécution et la plus fiable.

Nous citerons plus de détails dans ce qui suit.

4.6.1 Dérivée analytique

Bien que l'on puisse, en principe, réaliser une dérivée analytique des équations de mouvements des robots, la complexité, à la fois numérique et symbolique, des équations rend cette tache encore plus difficile. Malgré cela il existe des articles qui ont traité la dérivée analytique de quelques systèmes robotiques telle que l'article de [21].

4.6.2 Méthode des différences finies

Pour fournir à Ipopt les informations nécessaires à sa compilation, il a fallu remplir chaque classe de l'algorithme d'optimisation par les données nécessaires, allant du calcul du critère, jusqu'au calcul du gradient de ce critère, pour cette dernière l'approximation du gradient a été réalisé par la méthode des différences finies.

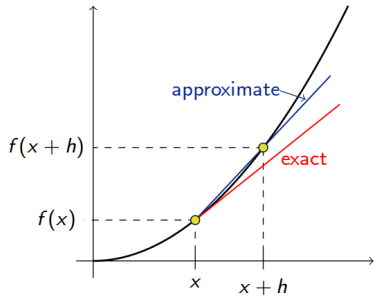

FIGURE 4.2 – différences finies

La différentiation numérique décrit les algorithmes d'estimation de la dérivée d'une fonction mathématique en utilisant les valeurs de la fonction ou d'autres informations sur la fonction. La méthode la plus facile, consiste à utiliser des approximations aux différences finies, une simple estimation de deux points est de calculer la pente d'une droite à proximité par les points $(x,f(x))$ et $(x+h\ f(x+h))$. la dérivée de la fonction f au point x est représentée par la limite présentée sur la figure **??**

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

La méthode des différences finies, permet de faire une approximation du gradient, mais elle ne donne pas des résultats précis. C'est pour cela que nous avons implémenté une méthode plus efficace qui est la différentiation automatique.

4.6.3 Différentiation Automatique

La différentiation automatique, ou la différentiation du calcul algorithmique, est un ensemble de techniques pour évaluer numériquement la dérivée d'une fonction mathématique, par un programme informatique. Elle diffère des autres méthodes classiques, par le temps de calcul et les résultats fiables et ne présente aucun problème pour le calcul des dérivées supérieures. Plusieurs logiciels ont été développés ces dernières années afin de générer automatiquement les dérivées, on peut citer : AD Model Builder, AD4CL, ADC, ADEL, Adept, ADNumber, ADOL-C, AUTODIF, CasADi ....etc

4.6.3.1 Adept

Adept (A fast automatic differentiation library for C++) est une bibliothèque de différentiation automatique, très rapide grace à l'utilisation des expressions templates, elle permet à des algorithmes écrits en C et C ++ d'être automatiquement différenciés. Il utilise une approche de surcharge de l'opérateur, donc très peu de modifications de code sont nécessaire.

De plus, la façon dont les modèles d'expression ont été utilisés et plusieurs autres optimisations importantes, signifient que la différenciation en mode inverse est nettement plus rapide que d'autres bibliothèques qui offrent des fonctionnalités équivalentes (Adol-C, CppAD et Sacado) et moins de mémoire est utilisée.[18] voir Annexe.

Nous avons implémenter cette librairie, mais malheusement nous avions des erreurs liées à cette librairie que nous n'avions pas pu résoudre. Nous avons donc passé à l'utilisation de d'autres librairies vu la contrainte de temps.

4.6.3.2 Adolc

Adol-C (Automatic Differentiation by OverLoading in C++) est algorithme d'automatique différentiation, utilisé surtout pour le calcul des dérivées premières et supérieures dérivés des fonctions vectorielles qui sont définis par des programmes informatiques écrits en C ou C ++ Adol-C peut gérer les codes basés sur des classes, des modèles et d'autres fonctionnalités avancées de C ++.

Pour plus de détails se référer à la documentation.[35] voir Annexe.

4.6.3.3 Structure du code avec Ipopt et Adolc

La structure du code utilisant Ipopt et Adolc peut être représentée comme montré sur la figure 4.3 ci-dessous

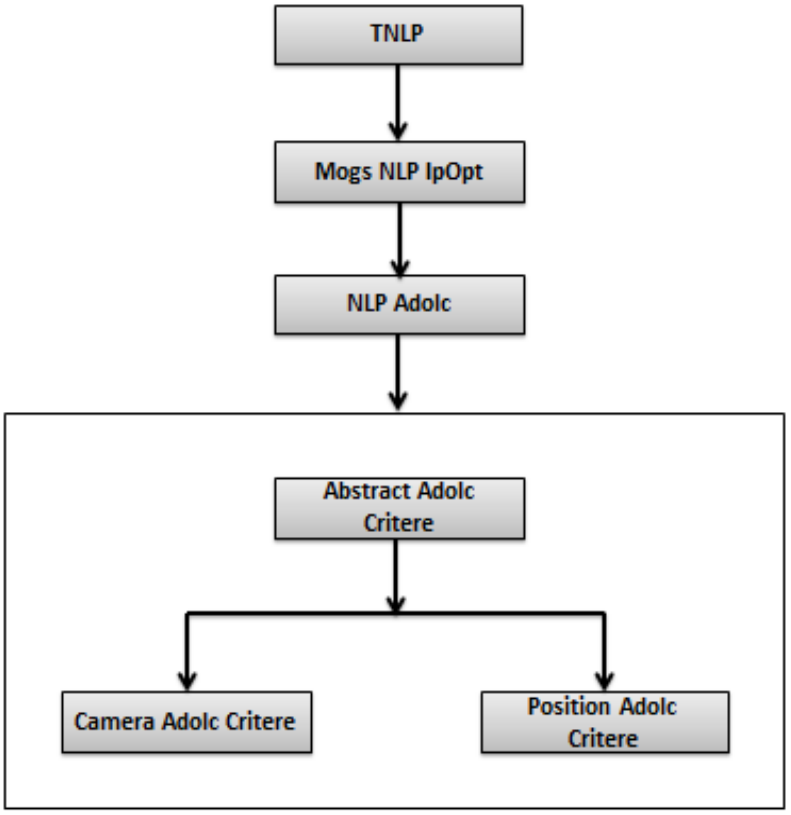

FIGURE 4.3 – Structure du code

La classe NLP Adolc contient les méthodes de calcul du critère et les informations nécessaires à l'éxécution de l'algorithme, elle hérite de la classe Mogs NLP Ipopt qui hérite à son tour de la classe TNLP. Un vecteur de critère a été créé représenté par Abstarct adolc critère, qui permet le calcul du critère en fonction des données présent dans chaque classe, Camera Adolc critere ou Position adolc critere etc.

4.7 Modélisation du Robot

4.7.1 Robot KUKA

Le robot utilisé pour les premiers tests est le robot kuka LWR, c'est un robot sériel avec sept degrés de liberté sur lequel l'effecteur est connecté à l'élement de référence par une seule chaîne. figure ci dessous.

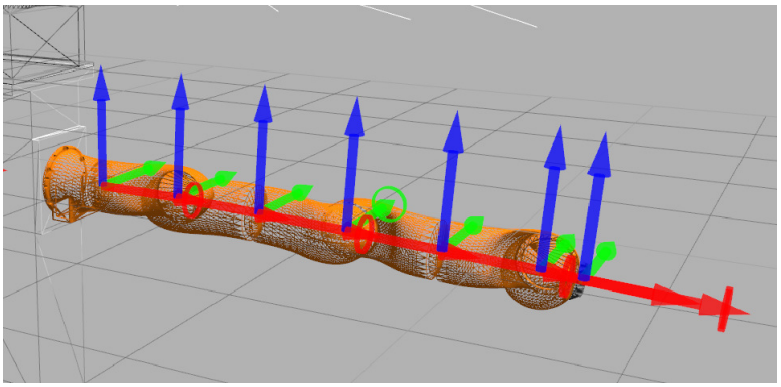

FIGURE 4.4 – Robot KUKA

Comme cité au préalable, la librairie RBDL (the Rigid Body Dynamics Library) utilisée dans le code est conçue de telle sorte, qu'on puisse charger n'importe quel types de robot, à partir d'un fichier correspondant (XML, URDF), contenant les informations sur le modèle.

Dans le but de reproduire les mouvements humains capturés dans une scène, quelques modèles ont été mises en places, dont on peut citer le Master Motor Map, comme représenté dans la figure 4.5. [31].

A partir des différentes captures de mouvements humains, le modèle Master Motor Map sert à unifier tous ces mouvements capturés du corps humain, et les convertir pour tout types de robots ou Avatar virtuel. La figure 4.5 représente la procédure de conversion des mouvements capturés.

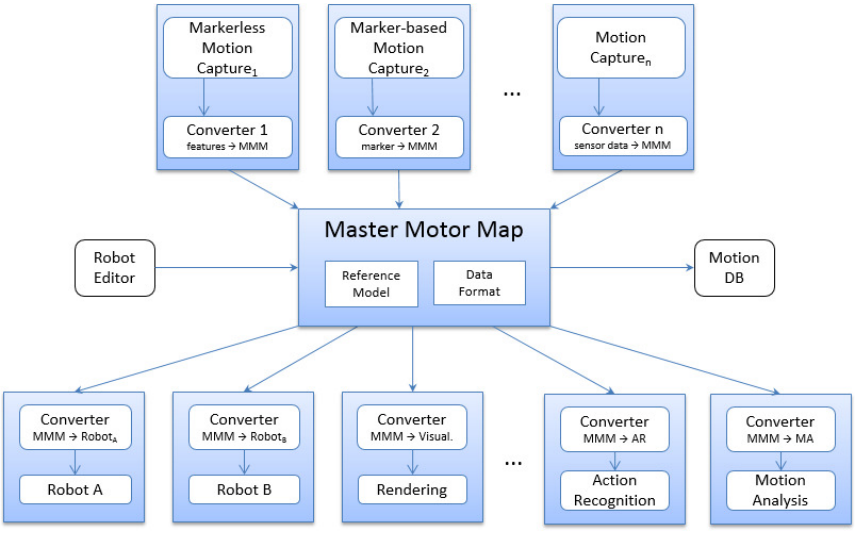

FIGURE 4.5 – Master Motor Map[31]

.

4.8 Modélisation de la caméra

Une caméra transforme les points 3D de l'espace en points 2D sur l'image, cette transformation suit un modèle géométrique. L'une des tâches les plus importantes est de lier les coordonnées image des élements de la scène avec leur coordonées spatiales.

Le calibrage de la caméra permet de mesurer les positions spatiales des objets de la scène et estimer la position de la caméra par rapport à la scène. Il est donc nécessaire de commencer par modéliser la caméra. Afin de pouvoir effectuer des calculs numériques ou des raisonnements géométriques à partir d'images, nous avons besoin d'un modèle qui décrit comment les positions 3D dans le monde se projettent sur une image 2D. Il existe plusieurs modèles qui décrivent les caractéristiques de la caméra, le modèle le plus communément utilisé en vision par ordinateur est le modèle de caméra dit sténopé ou encore pinhole.

4.8.1 Le modèle de sténopé :

Le modèle de caméra dit sténopé peut être décrit comme un ensemble d'un plan image, et un centre Optique.
Il se décompose en trois transformations élémentaires successives, une transformation repère monde vers repère caméra, une transformation repère caméra vers repère capteur et une transformation repère capteur vers repère image, comme montré dans la figure **??**.

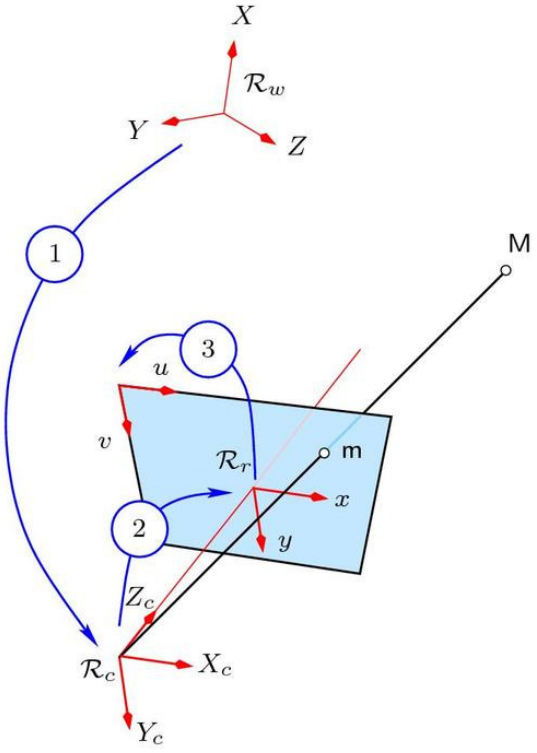

FIGURE 4.6 – le modèle sténopé

Afin de calculer les paramètres de la camèra, nous aurons besoin des coordonnées des points et donc des repères de ses coordonnées : Repère «Caméra» Rc : Lié à la caméra, nous choisissons comme origine le centre Optique de la caméra, et comme axe des Zc l'axe optique orthogonal au plan image. Les axes Xc et Yc sont choisis en parallèle au plan de l'image et perpendiculaire entre eux. Repère «Capteur» : Lié au plan image, origine sur l'axe optique.

Repère «Image» : Nous choisissons comme origine à ce repère le point centrale de l'image. Nous définissons la distance focale f comme étant la distance entre le centre optique et le centre de l'image.

C'est une projection perspective entre le repère du monde et le repère caméra qui transforme un point 3D de la scène $M(X, Y, Z)^t$ en un point dans l'image $m(x, y)^t$.

4.8.2 Une translation T et une rotation R :

Nous allons commencer par modéliser la position de la caméra, c'est à dire la position du centre de projection, et l'orientation de la caméra. Soit T la translation de la caméra, et R la rotation : R et T représentent la position et l'orientation de la caméra par rapport au monde, elles sont définies comme suit :

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}; R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

T et R, représentent les paramètres extrinsèques

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} + T$$

En coordonnées homoogènes la matrice de transformation homogène est donnée par :

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Nous avons calculé la position et l'orientation de la caméra, les déplacements peuvent alors être modélisés par une suite de matrices de rotation et de vecteurs de translation (vois Annexe)

4.8.3 Transformation repère caméra vers repère capteur :

x et y : la projection du point 3D. Les coordonnées x et y du point **m** , peuvent être exprimés à l'aide de la relation entre triangles similaires comme ce qui suit :

$$x = f\frac{X_c}{Z_c}$$

$$y = f\frac{Y_c}{Z_c}$$

C'est une projection perspective qui transforme un point 3D en un point image en unité métrique.

En coordonnées homogènes nous obtenons :

$$\begin{bmatrix} x \\ y \\ f \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

4.8.4 Transformation repère capteur vers repère image :

Nous définissons, le repère des pixels comme montré dans la figure 4.6.
u ,v étant les coordonnées pixelliques. Il faut tout d'abord faire un changement d'unité, le repère image étant métrique, (on mesure par exemple en mm), dans le repère pixels, l'unité est le nombre de pixels.
Il faut après, faire le changement de repère.
Soit :
f : la focale.
u ,v :coordonnées pixelliques.
u_o ,v_o : coordonnées du centre optique de la caméra dans le plan image (en pixels).
k_u et k_v représentent la taille du pixel.
x et y : la projection du point 3D.

Les coordonnées du point 3D par rapport au repère pixels, sont obtenues en effectuant une translation puis un changement d'unité.

$$u = \frac{x}{K_u} + u_o$$
$$v = \frac{y}{K_v} + v_o$$

En coordonnées homogènes on a :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{f} \begin{bmatrix} f.k_u & 0 & u_o \\ 0 & f.k_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix} = \frac{1}{f} \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix}$$

$$K = \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix}$$

K est la matrice des paramètres intrinsèques ou matrice de calibrage, elle représente les caractéristiques internes de la caméra, qui ne varient par en fonction de la position.
α_u : exprime, la distance focale, en nombre de pixels dans la direction horizontale.
α_v :exprime, la distance focale, en nombre de pixels dans la direction verticale.

Exemple : Si nous considérons, une rétine $1/2''$ correspondant à une résolution de 640 x 480 pixels, la surface photosensible correspondant à 6.4 x 4.8 mm^2, nous avons alors les valeurs de $k_u = k_v$ =100/mm.
u_0 =3,2 mm =320 en pixels
v_0 =2,4 mm =240 en pixels

Remarque : Un paramètre intrinsèque supplémentaire peut être introduit dans la matrice de calibrage, ce paramètre permet de modéliser les axes u et v non perpendiculaires.

$$K = \begin{bmatrix} \alpha_u & s & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix}$$

dans notre cas, on néglige ce paramètre on le considère egal à 0.

La transformation homogène repère monde vers repère image :

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{f} \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \frac{f}{Z_c} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$s. \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\tilde{q} = K \begin{bmatrix} R & T \end{bmatrix} . \tilde{P}$$

4.9 Identification de la matrice de projection de la caméra

En abscence des paramètres intrinsèques et extrinsèques de la caméra, il est donc nécessaire de passer à la calibration de la caméra en commençant par l'identification de la matrice de projection. Comme cité au préalable il faut au moins six correspondances pour déterminer la matrice de projection, mais en pratique, les données sont généralement bruitées, pour plus de précision il faut donc, une ou plusieurs centaines de points. Reprenons le modèle sténopé calculé au préalable :

Reprenant l'équation précédente :

$$\tilde{q} = K \begin{bmatrix} R & T \end{bmatrix} . \tilde{P}$$

Qu'on peut écrire sous la forme :

$$s. \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_o \\ 0 & \alpha_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

On appelle **M** la matrice de projection 3x4 qui regroupe les paramètres extrinsèques et intrinsèques de la caméra :

$$\begin{bmatrix} s.u \\ s.v \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34}z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Pour le calcul de la matrice de projection de la caméra, il nous faut au moins six correspondances, entre les points **P** dans le repère monde et leurs projections dans l'image **q**.

Chaque correspondance $(\mathbf{P}, \mathbf{q})$ de points fournit deux équations,

$$u = \frac{m_{11}X + m_{12}Y + m_{13}Z}{m_{31}X + m_{32}Y + m_{33}Z}$$

$$v = \frac{m_{21}X + m_{22}Y + m_{23}Z}{m_{31}X + m_{32}Y + m_{33}Z}$$

Pour chaque correspondance $(\mathbf{P}, \mathbf{q})$ on peut écrire le système d'équations :

$$Xm_{11} + Ym_{12} + Zm_{13} + m_{14} - uXm_{31} - uYm_{32} - uZm_{33} = u\ m_{34}$$
$$Xm_{21} + Ym_{22} + Zm_{23} + m_{24} - vXm_{31} - vYm_{32} - vZm_{33} = v\ m_{34}$$

En théorie on considère le $m_{34} = 1$, nous avons donc une équation avec 11 inconnus, comme nous avons mentionné au préalable, nous nécessitons au moins 6 correspondances pour pouvoir calculer la matrice de projection **M**.
Cependant il est conseillé d'utiliser un grand nombre de correspondances. En pratique une ou plusieurs centaines de points sont utilisés.

Pour **N** correspondances (P_i, q_i) on obtient le système linéaire suivant :

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 \\ X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -u_NX_N & -u_NY_N & -u_NZ_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -v_NX_N & -v_NY_N & -v_NZ_N \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \end{bmatrix} = \begin{bmatrix} . \\ . \\ . \\ u_im_{34} \\ v_im_{34} \\ . \\ . \end{bmatrix}$$

Système linéaire qui peut être représenté par la forme suivante : $\mathbf{A}.\mathbf{x} = \mathbf{b}$. On peut donc résoudre le système $\mathbf{x} = \mathbf{A}^{-1}.\mathbf{b}$. en utilisant la pseudo inverse de la matrice A, au sens des moindres carrés.

4.10 Extraction des paramètres intrinsèques et extrinsèques

De la matrice de projection **M**, nous pouvons extraire les paramètres intrinsèques ou la matrice de calibrage K, et la matrice de rotation R et la position t de la caméra.

4.10.1 Calcul de la matrice de calibrage

On sait que :

$$M = K \begin{bmatrix} R & T \end{bmatrix}$$

$$M = \begin{bmatrix} K.R & K.T \end{bmatrix}$$

On pose

$$\bar{M} = K.R$$

On multiplie de part et d'autres

$$\bar{M}.\bar{M}^T = (K.R).(K.R)^T = (K.R.R^T.K)$$

On sait que

$$R.R^T = I_{3x3}$$

Nous obtenons une matrice symétrique et définie positive, sa décomposition de Cholesky donne une matrice triangulaire supérieure proportionnelle à K.(voir Annexe)

4.10.2 Calcul des paramètres extrinsèques de la caméra

On distingue deux méthodes pour le calcul de la position de la caméra par rapport à un object :

1. **Une méthode linéaire :**
   Connaisant la matrice de projection

   $$M = K \begin{bmatrix} R & T \end{bmatrix}$$

   Connaisant la Matrice de calibrage K, on multiplie de part et d'autre par K^{-1} :

   $$\begin{bmatrix} R & T \end{bmatrix} = K^{-1}M$$

   On obtient la matrice de rotation et le vecteur de translation entre les repères monde et caméra.
   Cette méthode présente certains problèmes, dont on peut citer : - Nombre de paramètres non minimal
   - Contrainte d'orthogonalité non prise en compte
   - Sensible au bruit : nécessite des points 3D et 2D extêmement précise
   - Nécessite un nombre relativement élevé de correspondances 3D-2D (6 correspondances au minimum)
   - Peu adaptée aux contraintes pratiques et du temps réel.
   Cependant, pour plus de précision, il sera mieux d'utiliser la seconde méthode qui optimise la pose dans le sens des moindres carrés non linéaires.
2. **Une méthode non linéaire :**

Si on connaît K et une estimation R et t on peut estimer les projection de points 3D dans l'image

$$u_i = \alpha_u \frac{r_{11}.X_i + r_{12}.Y_i + r_{13}.Z_i + t_x}{r_{31}.X_i + r_{32}.Y_i + r_{33}.Z_i + t_z} + u_0$$

$$v_i = \alpha_v \frac{r_{21}.X_i + r_{22}.Y_i + r_{23}.Z_i + t_x}{r_{31}.X_i + r_{32}.Y_i + r_{33}.Z_i + t_z} + v_0$$

On note $\breve{u}$ et $\breve{v}$ les points détectés dans l'image correspondant à $P_i = (X_i, Y_i, Z_i)$

L'erreur de reprojection est donc égale à $(u_i - \breve{u}, v_i - \breve{v})$ elle s'annule lorsque $\mathbf{R}$ et $\mathbf{t}$ sont exactes, ce qui est impossible en pratique en raison du bruit.

Pour cela on va cherche à optimiser la pose dans le sens des moindres carrés de la fonction erreur de reprojection, ou fonction de coût.

f est la somme des distance entre les points estimés et les points mesusés dans l'image.

$$f(R,t) = \sum_i ((u_i - \breve{u})^2 + (v_i - \breve{v})^2)$$

On cherche R et t tels que la fonction f soit minimale.
Généralement la méthode utilisée est l'algorithme de Levenberg-Marquardt.

4.11 Résultats obtenus

Les tests ont été réalisés sur un modèle humain Yogi, en considérant les informations représentées sur le tableau 4.1 comme données d'entrées :

Considérant u_0 et v_0 : les coordonnées du centre de l'image en pixels.

x et y la projection d'un point 3D dans l'image.

K_u et K_v : la taille du pixel.

f : la focale

En considérant un pixel ponctuel dans une image, la projection des points 3D correspondant à ce pixel peut être représentée par une droite qui passe par l'origine de la caméra, (0,0,0) et un point de coordonnés $(x - u_0, y - v_0, f)$ dans le repère de la caméra.

En coordonnées métrique, on multiplie par la taille du pixel : $((x - u_0) * K_u, (y - v_0) * K_v, f)$

Nom du corps	Position dans l'image (u,v)
extremite majeur main gauche	(185 5)
extremite majeur main droite	(186 5)
poignet droit	(182 37)
poignet gauche	(189 32)
coude droite	(147 71)
coude gauche	(216 75)
epaule droite	(162 119)
epaule gauche	(208 121)
haut du cou	(185 101)
bas du cou	(183 119)
bas du cou	(185 5)
hanche droite	(165 220)
hanche gauche	(198 214)
genou droit	(174 285)
genou gauche	(252 253)
cheville droite	(189 365)
cheville gauche	(184 252)
extremite pied droit	(171 378)
extremite pied gauche	(181 284)

TABLE 4.1 – Les coordonnées en pixels des corps du robot Yogi

Le tableau 4.1, donne les coordonnées en pixels du corps illustré sur la figure 4.7

FIGURE 4.7 – Représentation de la posture à reconstruire

Les résultats qui ont été obtenus sont représentés sur les figures ci-dessous : La posture a été reconstruite avec le robot yogi qui a 60 degrés de liberté, une première vue nous a donné le résultat représentée sur la figure 4.8, nous constatons qu'à partir de cet angle de vue, le posture est bien reconstruite, sauf que l'angle de vue est différent de celui présent dans la figure 4.7.

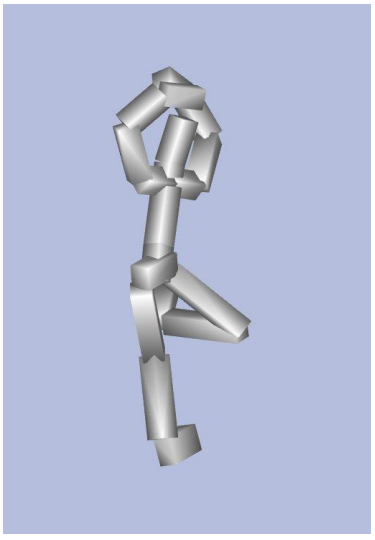

FIGURE 4.8 – Reconstruction de la posture avec le robot Yogi vue1

Dans les figure 4.9, 4.10, 4.11 ci-dessous, à partir d'autres angles de vues nous constatons que les mains ne sont plus superposées, ce qui peut être justifié par le nombre de points utilisé, qui n'est pas suffisant pour avoir plus de précision, et le fait qu'on ait utilisé qu'une seule caméra.

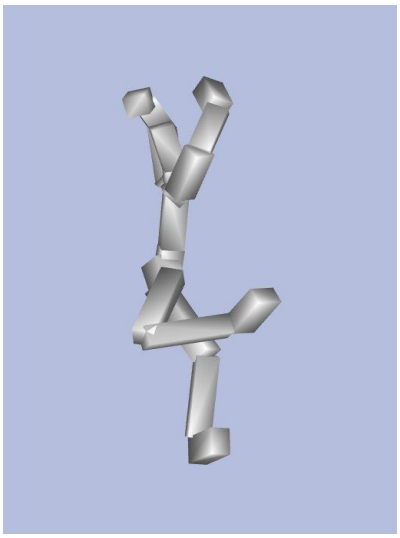

FIGURE 4.9 – Reconstruction du mouvement avec le robot Yogi vue2

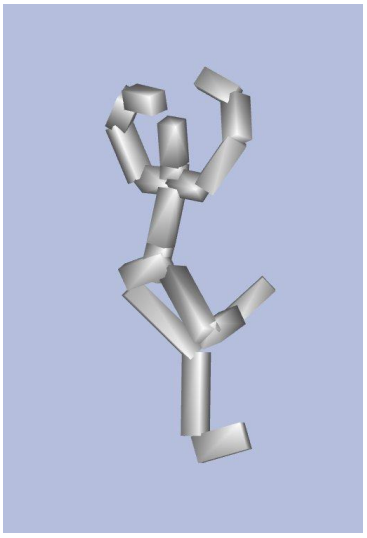

FIGURE 4.10 – Reconstruction du mouvement avec le robot Yogi vue3

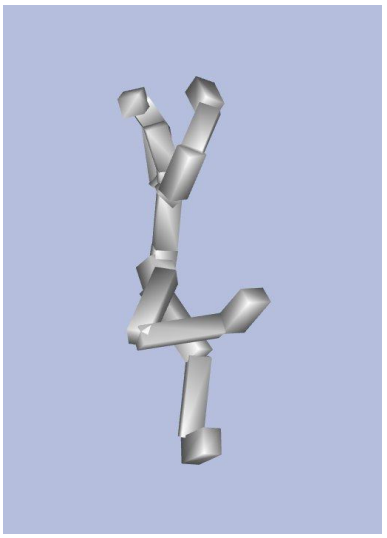

FIGURE 4.11 – Reconstruction du mouvement avec le robot Yogi vue4

Les résultats ont été obtenus en utilisant les points cités au tableau précédent, en utilisant une seule caméra, cependant l'utilisation d'une seule caméra avec un nombre de points limités engendre la perte de profondeur du corps à reconstruire. Chose qui peut justifier les résultats obtenus sur les figures 4.9, 4.10, 4.11. Par contrainte de temps, nous n'avons pas pu faire plus de tests avec d'autres caméras et d'autres modèles.

4.12 Conclusion et travaux futurs

Le sujet traité dans le cadre de ce stage est en lien avec l'équipe CAMIN de l'INRIA basées au LIRMM à Montpellier. Il consistait à créer un programme générique permettant de reconstruire le mouvement d'un modèle quelconque. Après implémentation du programme les tests ont été réalisés au début avec quelques modèles de base pour vérifier son fonctionnement et les résultats étaient cohérents. La méthode utilisée répond aux critères demandés telle que la vitesse d'éxécution, c'est une approche fiable et qui ne nécessite pas un matériel spécifique. Un mouvement est une succession d'images ou de postures reconstruites, parvenir à bien reconstruire une posture nous permettera par la suite de reconstuire le mouvement du corps. Cependant la contraite de temps ne nous a pas permis de faire plus de tests, avec d'autres caméras et d'autres modèles. D'autres tests sur des nouveaux modèles vont être implémentés, aussi avec d'autres types d'algorithmes d'optimisation et de différentiation automatique.

Bibliographie

[1] cpp.developpez.com.

[2] http ://casteyde.christian.free.fr/cpp/cours/online/x3782.html.

[3] http ://rbdl.bitbucket.org/.

[4] https ://developer.microsoft.com/fr-fr/windows/kinect.

[5] https ://www.vicon.com/vicon/about.

[6] https ://www.xsens.com/.

[7] http ://teaergo.com/site/fr/produits/fabricants/tea/captiv-motion.

[8] http ://www.capturelab.com/motion-capture/.

[9] http ://www.realite-virtuelle.com/tout-savoir-motion-capture.

[10] Bruce Guenther Baumgart. Geometric modeling for computer vision. Technical report, DTIC Document, 1974.

[11] Peter N Belhumeur and Gregory D Hager. Tracking in 3d : Image variability decomposition for recovering object pose and illumination. *Pattern Analysis & Applications*, 2(1) :82–91, 1999.

[12] Vincent Daval, Alban Bajard, Frederic Truchetet, and Olivier Aubreton. Estimation de surface de bézier à partir d'images de lumière structurée dans une optique de compression. In *CORESA 2013-16ème édition du colloque COmpression et REprésentation des Signaux Audiovisuels*, page 43, 2013.

[13] Paolo Favaro. Depth from focus/defocus. *document dated Jun*, 25, 2002.

[14] P Fua, L Herda, R Plankers, and R Boulic. Human shape and motion recovery using animation models. In *International Society for Photogrammetry and Remote Sensing*, number CVLAB-CONF-2000-001, 2000.

[15] S Burak Gokturk, Hakan Yalcin, and Cyrus Bamji. A time-of-flight depth sensor-system description, issues and solutions. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, pages 35–35. IEEE, 2004.

[16] Luis Gomes Perini. *La PGD au service de la corrélation volumique pour l'analyse du comportement des composites à l'échelle micro*. PhD thesis, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2014.

[17] Txomin Hermosilla. Automatic building detection and land-use classification in urban areas using multispectral high-spatial resolution imagery and lidar data. *ELCVIA : electronic letters on computer vision and image analysis*, 13(2) :0004–6, 2014.

[18] Robin J Hogan. Adept automatic differentiation library for c++ : User guide.

[19] Ilias Kalisperakis, Lazaros Grammatikopoulos, Elli Petsa, and George Karras. A structured-light approach for the reconstruction of complex objects. *Geoinformatics FCE CTU*, 6 :259–266, 2011.

[20] Juho Kannala and Sami S Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE transactions on pattern analysis and machine intelligence*, 28(8) :1335–1340, 2006.

[21] Sung-Hee Lee, Junggon Kim, F.C. Park, Mumsang Kim, and James E. Bobrow. Newton-type algorithms for dynamics-based robot movement optimization. In *IEEE Transactions on robotics*, volume 21, pages 657– 667, 2005.

[22] MAHERHANAFI. Analyse du mouvement humain par vision artificielle pour consoles de jeux vidéos. 2012.

[23] David Marr and Tomaso Poggio. Cooperative computation of stereo disparity. In *From the Retina to the Neocortex*, pages 239–243. Springer, 1976.

[24] Brice Michoud. *Reconstruction 3D à partir de séquences vidéo pour l ?acquisition du mouvement de personnages en temps réel et sans marqueur*. PhD thesis, Université Claude Bernard-Lyon I, 2009.

[25] Greg Mori and Jitendra Malik. Recovering 3d human body configurations using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(7) :1052–1062, 2006.

[26] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Generic and real-time structure from motion using local bundle adjustment. *Image and Vision Computing*, 27(8) :1178–1193, 2009.

[27] OujiKarima. *Numerisation 3D de visages par une approche de super-resolution spatio-temporelle non-rigide*. PhD thesis, Ecole Centrale de Lyon, 2012.

[28] Ronald Poppe. Vision-based human motion analysis : An overview. *Computer vision and image understanding*, 108(1) :4–18, 2007.

[29] Srikumar Ramalingam, Suresh K Lodha, and Peter Sturm. A generic structure-from-motion framework. *Computer Vision and Image Understanding*, 103(3) :218–228, 2006.

[30] Hedvig Sidenbladh, Michael J Black, and David J Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *European conference on computer vision*, pages 702–718. Springer, 2000.

[31] Orner Terlemez, Stefan Ulbrich, Christian Mandery, Martin Do, Nikolaus Vahrenkamp, and Tamim Asfour. Master motor map (mmm) ?framework and toolkit for capturing, representing, and reproducing human motion on humanoid robots. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 894–901. IEEE, 2014.

[32] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography : a factorization method. *International Journal of Computer Vision*, 9(2) :137–154, 1992.

[33] Raquel Urtasun, David J Fleet, and Pascal Fua. 3d people tracking with gaussian process dynamical models. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 238–245. IEEE, 2006.

[34] Andreas Wächter. Short tutorial : getting started with ipopt in 90 minutes. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[35] Andrea Walther. Getting started with adol-c. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2009.

[36] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading : a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8) :690–706, 1999.

Annexe A

Matrice de transformation 3D

A.1 changement de système de coordonnées

Soit un point P avec les coordonnées$(X', Y', Z')^t$ dans le repère d'origine O' lié à l'object. Pour calculer ses coordonnées (X, Y, Z) dans le repère d'origine O, il faut avoir : la translation **t** et la rotation **R** **t** et **R** représentent la pose de l'object dans le repère d'origine O et décrivent le mouvement amenant le repère d'origine O sur le repère d'origine O'.

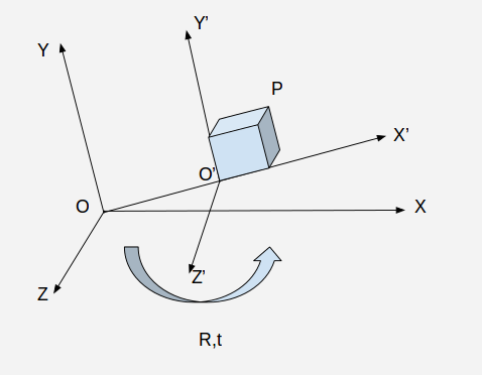

FIGURE A.1 – changement de système de coordonnées

.

A.1.1 Translation

La translation **t** est un vecteur de 3 dimensions reliant O et O' :

$$\mathbf{t} = O' - O$$

elle contient tout simplement les coordonnées de **O'** dans le repère d'origine **O** :

$$\mathbf{t} = \begin{pmatrix} X_{o'} \\ Y_{o'} \\ Z_{o'} \end{pmatrix}$$

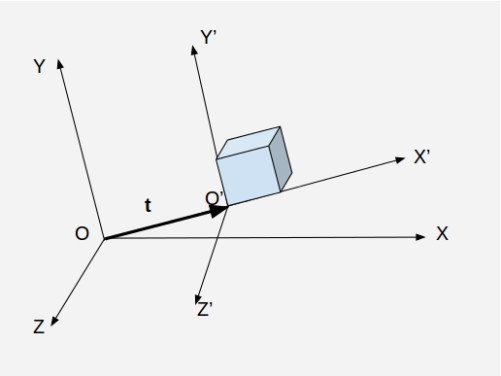

FIGURE A.2 – changement de système de coordonnées : translation

.

A.1.2 Rotation

La rotation **R** permet d'aligner chacun des axes du repère d'origine **O** avec l'axe correspondant sur le repère **O'**.

Cas 2D :

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} X'.cos(\alpha) - Y'.sin(\alpha) \\ X'.sin(\alpha) + Y'.cos(\alpha) \end{pmatrix}.$$

soit sous fome matricielle :

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} cos(\alpha) - sin(\alpha) \\ sin(\alpha) + cos(\alpha) \end{pmatrix} . \begin{pmatrix} X' \\ Y' \end{pmatrix}$$

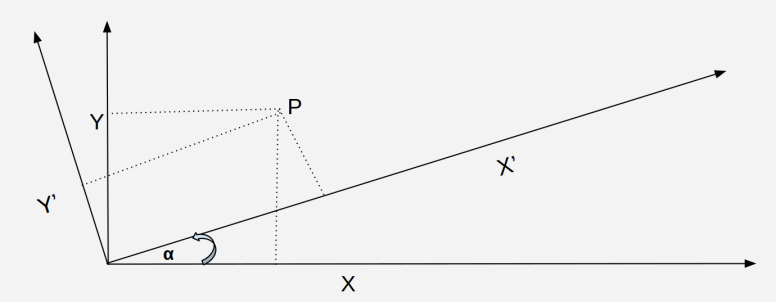

FIGURE A.3 – changement de système de coordonnées : rotation 2D

.

Généralisation au cas 3D : Une rotation autour d'un axe **Z** ne change pas les coordonnées sur cet axe.
Les coordonnées sur l'axe de rotation n'interviennent pas dans le calcul des nouvelles coordonnées sur les autres axes.

La rotation 3D autour de l'axe **Z** s'exprime donc :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix}$$

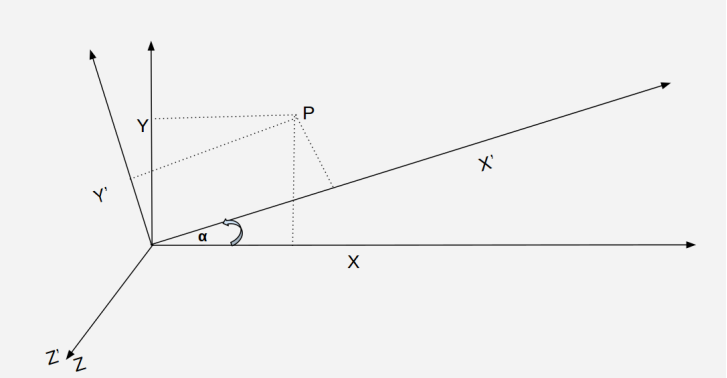

FIGURE A.4 – changement de système de coordonnées : rotation 3D

.

Rotation autour de l'axe **Z**

$$\mathbf{R} = rot(Z, \alpha) = \begin{pmatrix} cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation autour de l'axe **X**

$$\mathbf{R} = rot(X, \alpha) = \begin{pmatrix} 1 & 0 & 0 \\ cos(\alpha) & -sin(\alpha) & 0 \\ sin(\alpha) & cos(\alpha) & 0 \end{pmatrix}$$

Rotation autour de l'axe **Y**

$$\mathbf{R} = rot(Y, \alpha) = \begin{pmatrix} cos(\alpha) & sin(\alpha) & 0 \\ 0 & 1 & 0 \\ -sin(\alpha) & cos(\alpha) & 0 \end{pmatrix}$$

Les matrices qui, pour nous, servent à représenter les rotations, sont les matrices orthonormales, avec un déterminant qui vaut 1 ou -1. Pour une matrice de rotation, le déterminant doit valoir 1 . cela implique que l'inverse d'une matrice de rotation est sa transposée : $RR^T =$ I.
Une rotation autour d'un axe quelconque **U** est obtenue par le produit de trois rotatios élémentaires autours de chacun des trois axes :

$$\mathbf{R} = rot(U, \alpha) = rot(X, \alpha_x).rot(Y, \alpha_y).rot(Z, \alpha_z)$$

Calcul du point **P** dans le repère **O** :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} + \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} . \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix}$$

Annexe B

Décomposition Cholesky

Soit **A** une matrice symétrique définie positive de taille mxm
A peut être décomposée en

$$A = B.B^T$$

avec **B** une matrice triangulaire supérieure de taille mxm.

Annexe C

Principe de la reconstruction 3D

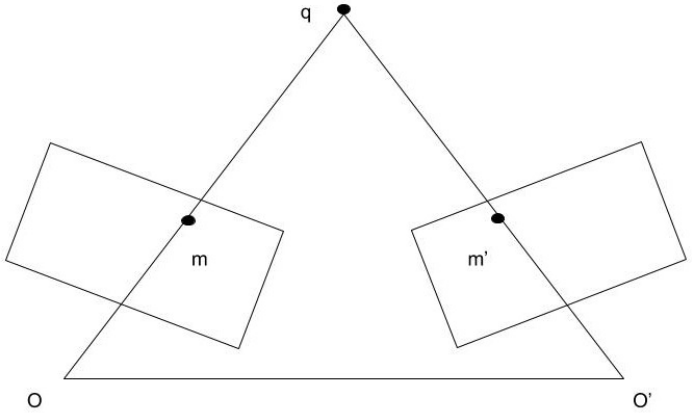

FIGURE C.1 – Principe de la triangulation

Connaissant le mouvement **R** et **t** entre deux caméras, on peut calculer le point 3D $q = (X, Y, Z)^T$ correspondant au couple (m, m') q est égale à l'intersection de Om et O'm', on parle de **triangulation**, il est exprimé dans le repère de la caméra gauche.

Projection de q dans la caméra gauche :

$$u = \alpha_u \frac{X}{Z} + u_0$$

$$v = \alpha_v \frac{Y}{Z} + v_0$$

Projection de q dans la caméra droite :

$$u' = \alpha'_u \frac{r_{11}.X + r_{12}.Y + r_{13}.Z + t_x}{r_{31}.X + r_{32}.Y + r_{33}.Z + t_z} + u'_0$$

$$v' = \alpha_v \frac{r_{21}.X + r_{22}.Y + r_{23}.Z + t_x}{r_{31}.X + r_{32}.Y + r_{33}.Z + t_z} + v'_0$$

Nous avons qautres équations linéaires avec comme inconnues **X**,**Y**,**Z**. En prenant les trois premieres équations nous pouvons calculer les coordonnées de q dans le repère de la caméra gauche.

Annexe D

Adept

Adept est algorithme d'automatique différentiation, très rapide et performant. Nous présentons ci-dessous un exemple de l'implémentation de cet algorithme. Considèrant l'example du code suivant qui a deux entrées et qui retourne une seule sortie.

```
double algorithme (const double x[2] ){
double y = 4.0;
double s = 2.0 * x[0] + 3.0 * x[1] * x[1] ;
y = *s sin(s)
return y ;

    }
```

La modification de ce code sera de la manière suivante :

```
#include<adept.h>
using adept : :adouble ;
adouble algorithme( const adouble x[2] ){
adouble y=4.0 ;
adouble s = 2.0 * x[0] + 3.0 * x[1] * x[1] ;
y = *s sin(s)
return y ;

    }
```

Maintenant pour pouvoir calculer et retourner le gradient du résultat, il faut faire :

```
#include<adept.h>
double algorithme_and_gradient( const double x_val[2],    // la variables d'entrées
                                double dy_dx[2] ){     // gradients de sorties
adept : :stack stack ;                  // Stockage des informations des dérivées
using adept : :adouble ;                // Importer adouble à partir d'adept
adouble x[2]= {x_val[0] , x_val[1]} ;   // Initialisation des variables d'entrées actives
stack.new_recording() ;                 // Commencement de l'enregistrement
adouble y= algorithme(x) ;              // appelle de la version sauvegardée pour l'argument
adouble
y.set_gradient(1.0) ;                   // Définir y comme Objective fonction
stack.compute_adjoint() ;               // Exécuter l'algorithme adjoint
dy_dx[0]=x[0].get_gradient() ;          // sauvegarder le premier gradient
dy_dx[1]=x[1].get_gradient() ;          // sauvegarder le second gradient
return y.value() ;                      // Retourner le résultant }
```

Annexe E

Adol-c

Adol-c est algorithme d'automatique différentiation, permettant la dérivation premières et supérieures. Ci-dessous un exemple de l'implémentation de cet algorithme. Exemple d'un programme qui doit être activé :

```
void eval(int n, int m, double *x,
double *y,
int double *k,
double double *z
) { double t=0;
for (i=0; i<n; i++)
t+=z[i]*x[i];
y[m-1]=t/m;
}
```

Après activation :

```
void eval(int n, int m,
double *px,
double *py,
int double *k,
double double *z)
{ short int tag=0;
trace_on(tag);
adouble *x, *y
x= new adouble[n];
y= new adouble[m];
for (i=0; i<n; i++)
x[i]«= px[i];
adouble t=0;
for (i=0; i<n; i++)
t+=z[i]*x[i];
y[m-1]=t/m;
for (j=0; j<m; j++)
y[j]»= py[j];
delete[] y;
delete[] x;
trace_off(); }
```