Titre: Title:	Protocole pour la création d'une base de données multimodale d'estimation de pose 3D d'enfants en salle de soins intensifs
Auteur: Author:	Olivier Desclaux
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Desclaux, O. (2022). Protocole pour la création d'une base de données multimodale d'estimation de pose 3D d'enfants en salle de soins intensifs [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10538/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10538/
Directeurs de recherche: Advisors:	Lama Séoud, Mickaël Begon, & Philippe Jouvet
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Protocole pour la création d'une base de données multimodale d'estimation de pose 3D d'enfants en salle de soins intensifs

OLIVIER DESCLAUX
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Juillet 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Protocole pour la création d'une base de données multimodale d'estimation de pose 3D d'enfants en salle de soins intensifs

présenté par **Olivier DESCLAUX**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

François GUIBAULT, président
Lama SÉOUD, membre et directrice de recherche
Mickaël BEGON, membre et codirecteur de recherche
Philippe JOUVET, membre et codirecteur de recherche
Luc DUONG, membre externe

DÉDICACE

À David, pour m'avoir donné le goût des sciences.

REMERCIEMENTS

Je voudrais tout d'abord remercier ma directrice de recherche, Lama Séoud, pour son accompagnement durant toute la durée de ma maîtrise. Même à l'autre bout de l'Atlantique, à tout moment de la semaine, et de jour comme de nuit, elle a toujours su m'aider techniquement et mentalement afin de mener à bien ce projet. Je remercie aussi mes co-directeurs, Mickaël Bégon, qui a toujours su m'aiguiller correctement face aux difficultés techniques, et Philippe Jouvet, dont l'aide précieuse à permis au projet PICUPE de faire ses premiers pas au CHU Sainte Justine.

Je remercie aussi tous les étudiants avec qui j'ai eu la chance de collaborer afin de faire avancer mon projet : d'une part Guillaume Jones et Renato Castillo, stagiaires durant l'été 2021, dont les contributions au développement logiciel sont toujours utilisées aujourd'hui, et d'autre part Amedeo Ceglia, dont l'expertise en modélisation squelettique m'a permis d'économiser de nombreuses heures.

De plus, je voudrais remercier les personnes incroyables qui m'ont accompagné durant ce séjour à Polytechnique Montréal : Gabriel Lepetit-Aimon, Emmanuelle Richer, et Antonin Tranchon, éminents cofondateurs du groupe culturel "Poly Cool Kids" dont le rayonnement et la bonne humeur perdurera dans les couloirs du 3ème étage du pavillon Lassonde pour de nombreuses années à venir.

Enfin, je remercie mes parents, mes frères et soeurs ainsi que mes amis qui m'ont soutenu durant mes 7 années d'études, et qui s'achèvent par ce mémoire. Leur soutien permanent m'a toujours poussé à donner le meilleur de moi-même, au travail comme dans ma vie.

RÉSUMÉ

Nous présentons les fondations du projet PICUPE, qui vise à intégrer un algorithme d'estimation de pose humaine dans les salles de soins intensifs du CHU Sainte Justine (CHUSJ). Pour cela, nous souhaitons filmer les patients avec deux caméras - une caméra de profondeur et une caméra thermique - placées au dessus de leur lit, et utiliser un réseau de neurones sur les images obtenues. Devant le manque de bases de données spécifiques pour entraîner notre réseau, nous proposons un protocole d'acquisition afin de créer notre propre base mutimodale au CHUSJ. Nous développons pour cela un système stéréo de deux caméras synchronisées temporellement, qui permet de faire l'acquisition de séquences vidéo à 29 images par seconde. Afin d'automatiquement générer la pose du sujet dans chaque image, nous combinons l'information de capteurs inertiels placés sur les membres du sujet avec un modèle musculosquelettique de la librairie OpenSim. Approuvé par le comité d'éthique du CHUSJ, notre protocole a pu être testé sur un patient 0. Notre base de données n'attend plus qu'à être populée.

ABSTRACT

We introduce the foundations of the PICUPE project, which works towards integrating Human Pose Estimation algorithms in CHU Sainte Justine's (CHUSJ) Intensive Care Unit. To do so, we aim to film patients with two cameras - a depth camera and an infrared camera - placed above their bed, and use a neural network on the retrieved images. Given the lack of specific databases to train our network, we introduce an acquisition protocol to create our own dataset at CHUSJ. We develop a stereo system of two time-synchronized cameras, which allows the recording of video sequences at 29 Frames Per Second. In order to automatically generate the subject's pose in each frame, we combine information from Inertial Measurement Units placed on the subject's limbs with a musculoskeletal model from the OpenSim library. Approved by the CHUSJ ethics committee, our protocol has been tested on a patient 0. Our database is now waiting to be populated.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES ALGORITHMES

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AGM	Analyse Générale du Mouvement
AP	Apprentissage Profond
CHUSJ	Centre Hospitalier Universitaire Sainte-Justine
EEG	Électroencéphalogramme
EPH	Estimation de Pose Humaine
GIL	Global Interpreter Lock
IA	Intelligence Artificielle
IMU	Inertial Measurement Unit
IPS	Images Par Seconde
IR	Infrarouge
IU	Interface Utilisateur
MoCap	Motion Capture
PCK	Probability of Correct Keypoint
PICUPE	Pediatric Intensive Care Unit Pose Estimation
SDK	Software Development Kit
SLP	Simultaneously-Collected Multimodal Lying Pose Dataset

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

Dans ce mémoire, nous présentons les fondations du projet Pediatric Intensive Care Unit Pose Estimation (PICUPE), qui vise à introduire l'Estimation de Pose Humaine (EPH) aux soins intensifs pédiatriques du Centre Hospitalier Universitaire Sainte-Justine (CHUSJ). Ce projet s'inscrit dans la continuité d'un travail de longue durée au CHUSJ, qui vise à digitaliser le suivi de ses patients aux soins intensifs, notamment par le biais de l'Intelligence Artificielle (IA). En effet, les récents progrès de l'IA offrent des perspectives prometteuses pour soutenir le corps médical dans sa surveillance en temps réel des patients. Par exemple, le projet MEDEVAC, existant au CHUSJ depuis 3 ans maintenant, cherche entre autres à anticiper des détresses cardiologiques chez le nourrisson en analysant l'évolution de sa température corporelle au cours du temps moyennant une caméra infrarouge.

Dans ce cadre, le projet PICUPE vise à évaluer la *pose* des patients alités, c'est-à-dire la configuration de leurs membres dans le lit, pour, ultérieurement, en déduire une analyse de leur mouvement au cours du temps. Si le sujet de l'EPH est vastement abordé dans la communauté de vision par ordinateur, notre cas d'application présente de nombreux défis propres. Tout d'abord, les patients concernés présentent une grande variété de morphologie (60% des patients aux soins intensifs ont moins de 2 ans, les autres peuvent atteindre jusque 17 ans) et sont allongés, donc peuvent prendre des poses très particulières (contrairement à des personnes debout). De plus, le modèle d'EPH que nous souhaitons développer devra être robuste aux nombreuses occlusions visuelles provenant des couvertures ou du matériel médical, ainsi qu'aux variations d'illumination, notamment si l'on souhaite pouvoir surveiller les patients durant la nuit. Enfin, notre solution devra pouvoir fonctionner en temps réel et être déployée au sein des soins intensifs. Elle doit ainsi combiner du matériel financièrement accessible et peu volumineux avec un traitement logiciel de l'information rapide et précis.

Dans la revue des connaissances (**Chapitre 2**), nous verrons qu'un nombre grandissant de travaux s'intéressent à l'étude de la pose dans un contexte médical, et plus particulièrement dans le cas de la surveillance de patients alités. Nous verrons que l'approche la plus adaptée dans notre contexte consiste à entraîner des réseaux de neurones à reconstruire la pose à partir d'images du patient. Cependant, la plupart des travaux d'EPH s'intéressent à l'étude d'adultes debout. Or, la force des réseaux de neurones - efficacement exécuter la tâche à laquelle ils ont été entraînés - fait aussi leur faiblesse dans le sens qu'ils ne parviennent pas à généraliser à de nouvelles situations. Ainsi, la plupart des bases de données développées pour

l'apprentissage de réseaux de neurones ne sont pas adaptées à notre cas spécifique d'enfants alités.

La première étape du projet PICUPE est donc de développer une base de données spécifique à ce cas d'application précis. Dans un contexte d'apprentissage supervisé de réseaux de neurones, ceci nécessite d'avoir d'une part des séquences vidéos d'enfants alités, et d'autre part leur pose associée pour chaque image de la base. Dans ce mémoire, nous proposons et évaluons un protocole afin de générer ces deux données conjointement. Les différents objectifs spécifiques associés sont donnés en **Chapitre 3**.

Dans le **Chapitre 4** nous entraînons un réseau de neurones sur une base de données d'adultes alités, afin de dresser un cahier des charges précis de la base que l'on souhaite créer. Nous nous intéresserons notamment à la modalité dans laquelle les patients devront être filmés. Nous verrons que la combinaison multimodale d'images thermiques et de profondeur donnera les résultats les plus pertinents.

Le **Chapitre 5** s'articulera autour de la génération de séquences vidéos multimodales : en travaillant avec une caméra thermique et une caméra de profondeur, nous nous intéresserons à la calibration des caméras individuelles, à la calibration du système stéréo global, et enfin à la génération d'images synchronisées temporellement.

Dans le **Chapitre 6**, nous nous intéresserons à une méthode de génération automatique de pose via le couplage d'information de capteurs inertiels avec des modélisations du squelette humain. Nous verrons que cette approche, quoique offrant des premiers résultats intéressants, n'est pas parfaitement adaptée dans notre contexte pour plusieurs raisons techniques et pratiques.

Dans le **Chapitre 7** nous montrons le protocole final obtenu. Ayant été approuvé par le comité éthique du CHUSJ, il a été testé sur un patient 0 dans une salle de simulation des soins intensifs. Avec une interface graphique intuitive, notre système stéréo est facilement et rapidement calibré, et permet de générer des séquences vidéo de plusieurs milliers d'images en 3 modalités (RGB, profondeur, IR) synchronisées temporellement. Ce protocole servira de fondation solide pour l'ensemble du projet PICUPE et, nous l'espérons, d'autres projets au CHUSJ.

CHAPITRE 2 REVUE DES CONNAISSANCES

Dans cette section, nous souhaitons justifier l'intérêt du travail réalisé durant cette maîtrise et présenter les assises du projet. Tout d'abord, nous justifierons l'utilisation de l'EPH dans un contexte clinique. Le sujet de l'EPH étant vastement abordé dans la communauté scientifique, nous nous limiterons à citer le plus possible des travaux appliqués à un contexte médical. Nous analyserons les forces et faiblesses des différentes modalités et méthodes utilisées pour l'EPH ainsi que leur pertinence dans notre cas d'application, afin de sélectionner la plus adaptée. Enfin, nous évaluerons en détail les bases de données existantes les plus adaptées dans notre cas, afin de justifier la nécessité de construire une nouvelle base.

2.1 Sur l'estimation de pose et son application dans le monde médical

Dans cette section, nous introduisons brièvement le concept d'EPH, ses objectifs et donnons deux exemples d'application générale. Nous verrons ensuite ses nombreuses applications spécifiques au contexte médical.

2.1.1 Présentation succincte de l'estimation de pose

L'Estimation de Pose Humaine (EPH) est une discipline de la vision par ordinateur visant à trouver la configuration d'une représentation corporelle d'une personne dans un jeu de données capturées par des senseurs, le plus souvent des images ou séquences vidéo. Cette représentation peut avoir différents niveaux de précision [1]. L'approche la plus simpliste, dite cinématique, consiste à détecter les articulations du corps humain et les relier en une structure de graphe, appelé squelette. Légèrement plus complexe, l'approche planaire vise à représenter schématiquement les différents membres du corps par des rectangles. Enfin, l'approche volumétrique représente le corps humain par un modèle paramétrique 3D [15].

Dans les deux premiers cas (cinématique et planaire), il est important de noter que cette reconstruction peut se faire soit en 2D - en considérant que tous les points ou rectangles sont dans un même plan - ou en 3D. L'approche volumétrique est intrinsèquement en 3 dimensions.

Plusieurs raisons expliquent la variété de ces approches. Ces dernières années, l'amélioration du matériel de capture de données (caméras, senseurs, etc.) ainsi que du matériel de traitement de données (processeurs, cartes graphiques, etc.) ont progressivement permis de créer des modèles plus complexes. De plus, la précision (cinématique, planaire ou volumétrique) du modèle que l'on souhaite développer dépend bien évidemment de son application.

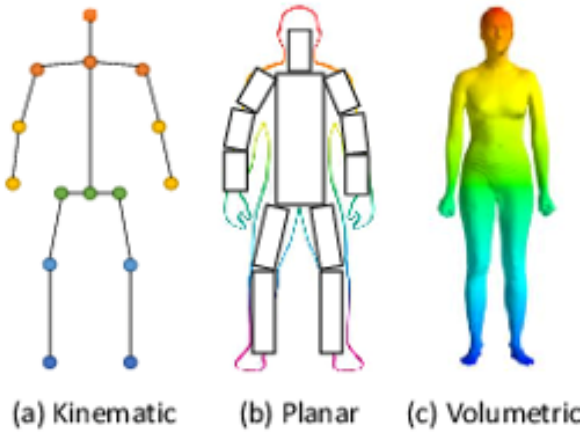


FIGURE 2.1 Les trois grands types d'estimation de pose. Source : [1]

Considérons deux exemples.

Les systèmes d'aide à la conduite utilisent des caméras pour filmer l'environnement d'un véhicule. La tâche principale est de prévenir les collisions avec des membres externes vulnérables, tels que des piétons ou des cyclistes. Ici, l'EPH est utilisée pour comprendre leur mouvement, anticiper leurs positions futures, et ainsi diriger le système dans sa prise de décision. Dans ce cas, ce n'est pas une reconstruction précise du corps humain qui nous intéresse, mais une simple description de son mouvement : l'approche cinématique est la plus appropriée [16].

Dans les domaines de la réalité augmentée ou virtuelle, les personnes environnant un sujet doivent être digitalisées en avatars : on doit les reconstruire dans l'environnement virtuel. Afin d'offrir un réalisme optimal, une simple reconstruction cinématique de ces personnes n'est pas suffisante, et l'approche volumétrique est bien plus adaptée [17].

2.1.2 L'estimation de pose pour l'aide au diagnostic

Chez l'enfant Dans leur étude pionnière [18], Einspieler et al. développent la méthodologie d'Analyse Générale du Mouvement (AGM), définissant un ensemble de mouvements "normaux" et "anormaux" chez le nourrisson. Cette analyse qualitative permet alors de détecter une grande variété de détresses neurologiques (hémorragie cérébrale, leucomalacie, etc.) uniquement en regardant ses mouvements [19]. Dans la continuité de ces travaux, Adde et al. [20] montrent que l'AGM est particulièrement efficace dans l'anticipation du développement de la Paralysie Cérébrale (PC) chez l'enfant en bas-âge, tandis que Teitelbaum et al. l'utilisent pour poser un diagnostique anticipé de l'autisme [21].

Cependant, l'AGM possède plusieurs défauts. C'est en effet un processus long qui doit être

performé par un médecin entraîné, et le résultat est par conséquent subjectif à l'expérience de l'observateur. Ainsi, de nombreux travaux ([10], [22], [23]) visent à utiliser l'EPH dans l'AGM afin d'obtenir une analyse objective. Développer des algorithmes d'EPH robustes permettrait aussi d'augmenter le nombre de patients auscultés, tout en libérant du temps pour les médecins. Par ailleurs, une quantification précise du mouvement des enfants pourra donner lieu à une catégorisation standardisée des mouvements associés à la PC [24].

Chez l'adulte D'une manière analogue au cas des enfants, plusieurs travaux chez des adultes utilisent l'EPH pour apporter une analyse quantitative à des diagnostics souvent encore subjectifs. Ainsi, dans [25], l'estimation de pose est utilisée pour la détection de symptômes Parkinsoniens, afin d'optimiser la mise-à-jour du traitement de patients. En post-opératoire, des algorithmes d'EPH sont utilisés pour l'analyse du cycle de marche chez des patients en rémission d'un infarctus afin de suivre leur progrès [26].

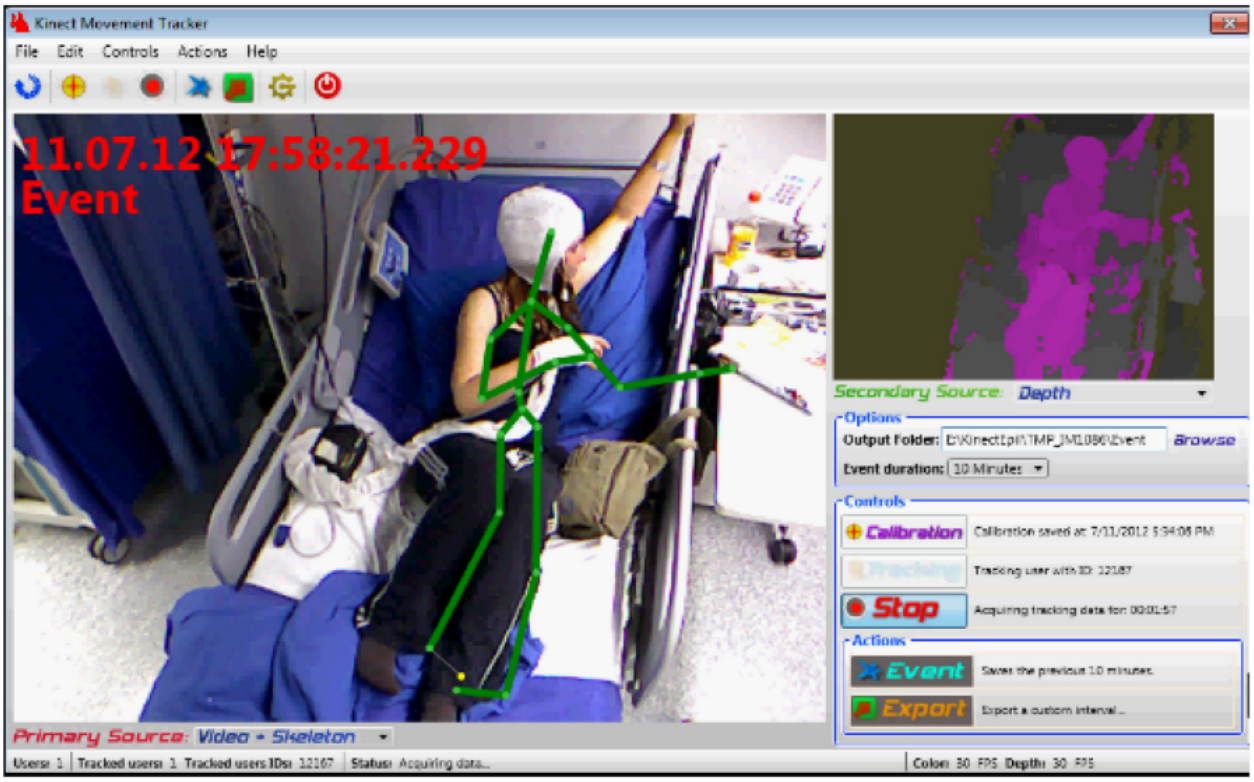


FIGURE 2.2 Un exemple de surveillance de patients avec de l'EPH. Les auteurs utilisent un réseau de neurones sur des images obtenues avec une caméra placée au-dessus du lit. La pose est couplée avec des informations EEG pour analyser les crises de patients épileptiques. On observe une mauvaise reconstruction du bras gauche. Source : [2]

.

2.1.3 L'estimation de pose pour la surveillance clinique

Dans un contexte clinique, l'estimation de pose peut s'avérer utile pour assurer une surveillance continue de patients. En effet, certains travaux montrent comment la pose de patients peut affecter l'évolution de leur état de santé lorsqu'ils sont alités à l'hôpital. Le développement d'algorithmes d'EPH apparaît comme un vrai soutien au personnel hospitalier pour le suivi de patients en temps réel. Par exemple, Lee et al. ont montré l'impact de la pose du patient sur l'apnée du sommeil [27], et dans cette continuité, une architecture de réseau de neurones a été développée pour le suivi de la posture au cours du sommeil avec l'hôpital de Toronto [28].

Dans [2], les auteurs couplent des algorithmes d'estimation de pose en 3D à l'information d'électroencéphalogramme (EEG) pour l'analyse du mouvement de patients épileptiques, et assurent par ce système un suivi permanent (24h/24, 7j/7) de leurs patients. Dans le même but de suivi d'épilepsie, Chen et al. développent des algorithmes d'estimation de pose spécifiques à chaque individu, dans le cadre de patients semi-allongés [29].

2.1.4 Conclusion sur l'EPH dans le monde médical

Dans cette section, nous avons vu un panel d'applications où l'EPH s'inscrit comme un vrai support pour le corps médical, que ce soit un médecin lors de la pose d'un diagnostic, ou un personnel hospitalier lors de la surveillance de patients en continu. Sur ce dernier point, si les travaux que nous avons cités se concentrent sur de la *détection* d'évènements de détresse, à notre connaissance, aucuns travaux n'ont été mené dans un but d'*anticipation*.

2.2 Les différentes méthodes d'estimation de pose

Dans cette section, nous introduisons les principales méthodes utilisées dans la littérature pour reconstruire la pose d'une personne. Les différentes méthodes sont catégorisées selon la modalité des données d'entrée utilisées.

2.2.1 Utilisation d'une unique caméra

Caméras RGB

L'approche la plus classique pour estimer la pose d'une personne dans l'espace est de la filmer avec une caméra. Jusqu'au milieu des années 2010, les caméras couleur classiques (que nous appellerons caméra RGB, "Red-Green-Blue") étaient le standard utilisé dans la majorité des publications académiques.

À cette époque, les méthodes d'EPH étaient globalement divisées en deux étapes [30]. La première étape consistait en une extraction de caractéristiques de différents niveaux avec des méthodes déterministes. Par exemple, des calculs de contours ou de silhouette, une détection des visages par algorithme de Viola-Jones et filtres de Haar, ou encore la description de points clés par SIFT. La deuxième étape consiste à associer ces caractéristiques à un modèle du corps, qu'il soit cinématique, planaire ou volumétrique. Ceci peut être fait selon deux grandes directions : soit en apprenant une relation allant du modèle vers l'image (approche générative), soit en allant de l'image vers le modèle (approche discriminative).

L'EPH a toutefois pris un tournant aux alentours de 2015, avec l'explosion de l'Apprentissage Profond (AP), offrant des gains de performance spectaculaires sur les bases de données existantes. Une des premières architectures, DeepPose [31] propose de voir l'EPH comme un problème de régression d'une image vers un ensemble de coordonnées des articulations du corps en utilisant une architecture de réseau de neurones convolutifs. Depuis, la quasi-totalité des méthodes d'EPH sur des images monoculaires sont basées sur de l'AP : dans une revue de littérature de 2020 [1], les auteurs recensent plus de 250 articles différents utilisant de l'AP pour l'EPH.

Si la modalité RGB est très pratique pour entraîner des réseaux de neurones du fait de la disponibilité de nombreuses bases de données, elle présente plusieurs lacunes pour une application dans le contexte du suivi d'un patient alité. Premièrement, elle est particulièrement sensible aux occlusions de sources externes (par exemple une couverture). Deuxièmement, l'information recueillie dépend de l'illumination de la pièce, par exemple, aucune information ne peut être recueillie dans le noir (problématique dans le cadre d'une surveillance de l'état de santé du patient pendant la nuit).

Caméras de profondeur

Le développement de caméras dites *3D* ou *de profondeur* ont donné une nouvelle dimension à l'EPH [32]. En effet, la commercialisation du premier senseur Microsoft Kinect en 2010 a démocratisé l'utilisation des caméras de profondeur dans des applications commerciales, les derniers modèles sur le marché (Kinect Azure pour Microsoft, RealSense pour Intel) ne coûtant que quelques centaines de dollars.

En utilisant une combinaison projecteur-capteur dans le spectre de l'infrarouge, une caméra de profondeur est capable de déterminer à quelle distance de celle-ci se trouvent les objets de la scène imagée. Elle construit une image 2D où l'intensité de chaque pixel représente la distance associée. Cette image est souvent couplée à une image RGB prise par une caméra intégrée sur le même dispositif, pour former une seule image dite RGBD (de l'anglais Red-

Green-Blue-Depth). Un exemple de ces deux images est donné en **Figure 2.3**.

L'information de profondeur s'est peu à peu imposée pour son accessibilité mais surtout parce qu'elle facilite implicitement l'extension de l'EPH du 2D vers la 3D.

Une des raisons de la popularité des images de profondeur est la possibilité d'utiliser des architectures de réseaux de neurones développés pour des images RGB sur des images de profondeur avec un minimum de modifications, ces dernières n'étant finalement des images RGB à un seul canal. Par exemple, les auteurs de [33] utilisent une architecture de réseau de neurones en sabliers emboîtés (dite Stacked-Hourglass), initialement développée sur des images RGB [34], pour estimer la pose 3D de personnes filmées avec une caméra de profondeur Kinect V2.

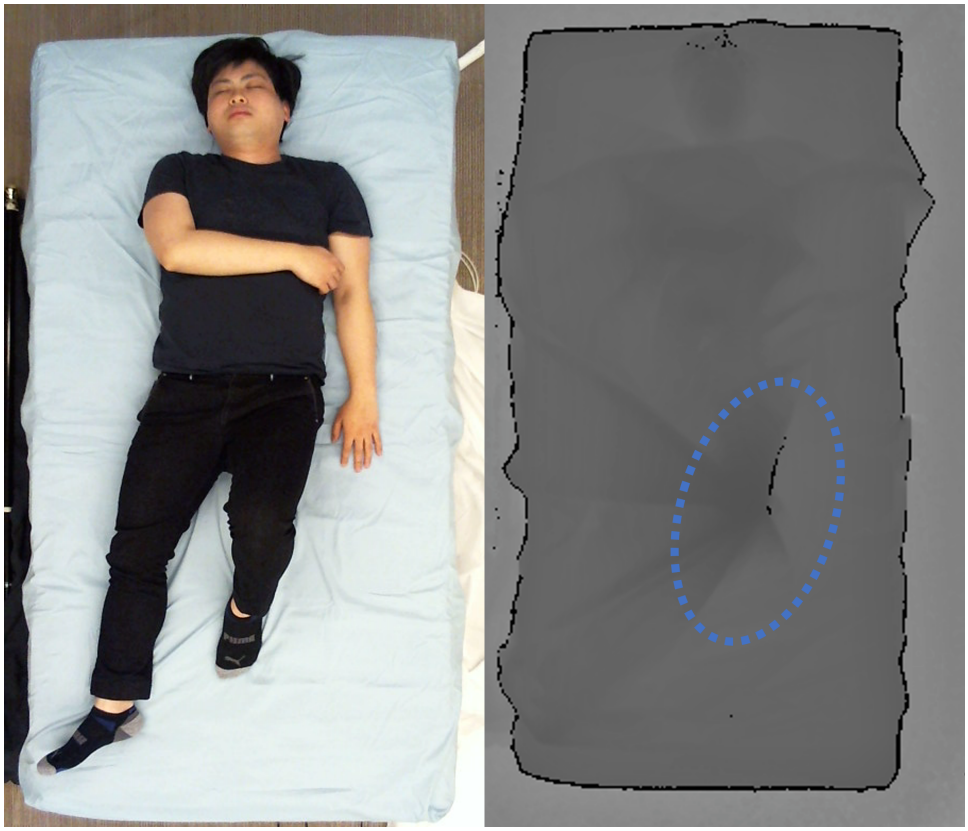


FIGURE 2.3 Exemple de limitation de la modalité de profondeur. A gauche, la pose du patient en RGB. À droite, la même pose mais sous une couverture, captée avec une caméra de profondeur. Le genou gauche (cercle bleu) du patient tire la couverture, faisant disparaître les autres membres. Inspiré de [3]. Images tirées de [4].

Si l'imagerie de profondeur est moins sensible aux changements d'illumination qu'une caméra RGB (puisqu'elle est basée sur une projection de lumière infra-rouge), elle présente elle aussi une limitation principale dans le cadre hospitalier. En effet, les patients étant la majeure

partie du temps sous une couverture, une partie de l'information de profondeur est forcément perdue [3]. Yin et al. considèrent l'exemple d'un patient allongé sur le dos, drapé d'une couverture, représenté dans la **Figure 2.3**. En relevant son genou gauche, le patient tend la couverture, et fait disparaître la jambe droite de l'image de profondeur.

Caméras Infrarouge

Une dernière modalité utilisée de plus en plus pour l'EPH, notamment pour la vision nocturne, est l'imagerie Infrarouge (IR). Dans une image IR, l'intensité de chaque pixel est une mesure du rayonnement de l'objet observé dans une gamme de longueur d'onde spécifique. L'information thermique est encodée sur un seul canal, pour chaque pixel de la scène. Il existe de plus en plus des capteurs infrarouge de grade commercial, donc abordable, comme la FLIR Lepton 3.5, toutefois, la résolution spatiale offerte est limitée à 160x120 pixels.

Pour le suivi de patients dans un lit d'hôpital, un système d'acquisition infrarouge sélectif de la tranche (700-1000 nm) a été proposé [35] et des réseaux de neurones convolutifs ont été entrainés sur les images obtenues pour faire de l'EPH. De même, en utilisant un capteur infrarouge lointain (FLIR Lepton), Liu et Ostadabbas modélisent les membres de patients par des cylindres, couplés à des modèles thermodynamiques dans le spectre (800 - 1500 nm) [36].

Si ces deux approches offrent des résultats prometteurs, les auteurs de [3] montrent encore une fois la limitation de l'imagerie thermique due à la couverture placée au-dessus du patient : les phénomènes de diffusion thermique viennent altérer la précision des algorithmes d'EPH.

2.2.2 Utilisation de plusieurs caméras

Motion Capture - MoCap

Lorsqu'il s'agit de faire de l'EPH dans un environnement contrôlé, un système de capture de mouvement à multiple caméras (en anglais *Motion Capture*, plus connu sous le nom de MoCap) vient immédiatement à l'esprit. Considéré comme une des références en termes de précision [37], cette méthode utilise un grand nombre de marqueurs réfléchissants placés sur les articulations du sujet, et filmés par plusieurs caméras à rayonnement infrarouge, comme on peut le voir dans la **Figure 2.4a**.

Comme dans le cas monoculaire, l'intégration de ces données pour reconstruire le squelette en 3D peut être réalisé d'une multitude de manières. Une des approches les plus traditionnelles est la résolution du problème de cinématique inverse [38], où l'on utilise des modèles du corps humain sur lequel on a aussi placé des marqueurs virtuels [39]. L'intégration des données

(a) **(b)**

FIGURE 2.4 (a) Exemple de système MoCap, dans le cadre de la création de la base de données Human3.6M [5]. (b) Visualisation schématique du problème de cinématique inverse : on cherche à retranscrire le mouvement humain dans un modèle musculosquelettique. [6]

.

des caméras devient alors un problème d'optimisation : à chaque pas de temps, la nouvelle configuration de notre modèle est obtenue en minimisant la distance entre les marqueurs expérimentaux et les marqueurs virtuels (cf. **Figure 2.4b**).

Malheureusement, un tel système n'est pas envisageable dans un contexte de surveillance clinique. Il est tout d'abord intrusif et long à mettre en place puisqu'il nécessite de placer plusieurs capteurs sur le patient. De plus, il comporte plusieurs caméras donc est très dispendieux et requiert souvent un post-traitement considérable des données.

Enfin, un système MoCap présente la même problématique que les autres modalités face aux occlusions visuelles. Pour contrer ce problème, les auteurs de [40] effectuent une capture de mouvement d'un patient alité (5 caméras, 14 marqueurs) sans couverture, puis simulent une couverture qui tombe itérativement sur le patient pour générer artificiellement des images de profondeur. Si les auteurs obtiennent de bons résultats sur leur base de données synthétiques, aucun travail ultérieur sur des données réelles n'a été conduit.

Une redondance d'informations

Utiliser plusieurs caméras ayant des angles de vue différents reste néanmoins pertinent dans de nombreux cas, car la redondance d'informations peut permettre une meilleure reconstruction de la pose. Dans un contexte médical toujours, mais hors du suivi de santé de patients, [41] utilisent un jeu de 3 caméras de profondeur placées dans une salle opératoire pour reconstruire la pose 3D de chirurgiens. Leurs différents modèles basés sur l'information d'une, deux ou des trois vues montrent clairement que plus le nombre de vues augmente, plus la précision du modèle augmente.

2.2.3 Utilisation de senseurs physiques

L'utilisation de caméras n'est pas la seule manière d'estimer la pose d'un sujet : certains senseurs mesurant des grandeurs physiques comme des changements de pression, des points de contact, ou encore des accélérations peuvent s'avérer pertinents dans certains contextes.

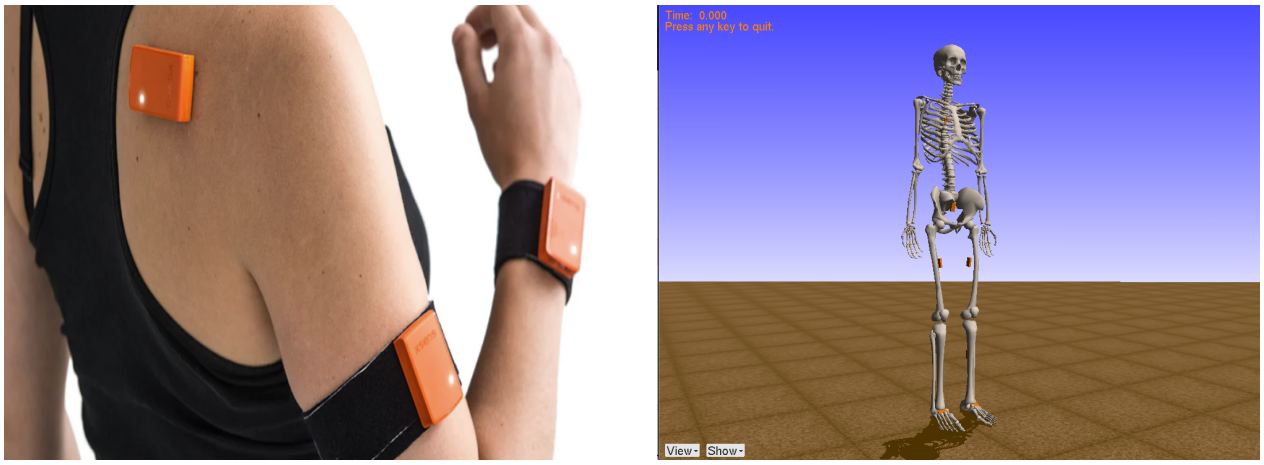
Capteurs de pression

Dans [42], les auteurs utilisent de fins matelas parsemés de capteurs de pression pour reconstruire des cartes de pression du lit. Ces dernières sont utilisées à des fins de classification des différents membres du corps, et l'estimation de leur pression sur le matelas, afin d'anticiper des risques d'escarres chez des patients en récupération post-opératoire. Cependant, les auteurs de [43] expliquent que les cartes de pression souffrent d'une faible précision au niveau des articulations à faible poids (main ou pied par exemple), et ne permettent pas la reconstruction précise d'une pose. De plus, ils argumentent que ces cartes de pression ne peuvent être considérées comme des images classiques, et donc utilisées comme telles dans des architectures de réseaux de neurones traditionnels, et doivent subir de lourdes transformations en prétraitement.

Capteurs inertiels IMU

Un autre type de senseurs de plus en plus utilisés sont les Inertial Measurement Unit (IMU) : composés d'un accéléromètre, d'un magnétomètre et d'un gyroscope, ces boîtiers placés sur le corps sont capables d'enregistrer leur accélération et orientation au cours du temps. Comme dans le cas de caméras, les progrès technologiques des dernières décennies ont permis de créer des IMU de plus en plus petits (de la taille d'une petite boîte d'allumettes), précis et au prix abordable. Pouvant être sans-fil, ils ne souffrent d'aucun problème d'occlusion contrairement

aux systèmes optiques, et peuvent même être utilisés dans des environnements non-contrôlés (typiquement en extérieur).



(a) **(b)**

FIGURE 2.5 (a) Un exemple d'IMU non-filaire de la société XSens [7] (b) Exemple de quelques IMU virtuels placés sur un modèle squelettique du corps. [8]

De manière analogue aux systèmes MoCap, la pose d'une personne peut être reconstruite en utilisant des modèles 3D du corps auxquels on attache des IMU virtuels. Le problème revient encore une fois à une résolution de cinématique inverse. Un exemple est donné en **Figure 2.5b**, où l'on peut voir un modèle squelettique de la librairie OpenSim [8], sur lequel sont placés des IMU représentés par des boîtiers oranges.

L'orientation des IMU au cours du temps est reconstruite en intégrant sa vitesse angulaire mesurée par le gyroscope. Mais les mesures de ce dernier souffrent d'une dérive au cours du temps (notamment liée aux champs magnétiques externes), et introduisent un biais [44]. De fait, les mesures d'orientation globale d'un IMU peuvent dériver de 15°en seulement quelques minutes [45].

Si la principale utilisation des IMU était initialement dans le divertissement (industrie cinématographique, jeux vidéo, etc.), de plus en plus de travaux les utilisent pour des applications médicales. Ainsi, les auteurs de [46] utilisent 7 IMU placés sur la partie inférieure du corps de 26 patients pour une analyse de la marche, et comparent leurs mesures avec un système Vicon calibré. Ils démontrent alors que les IMU peuvent être utilisés comme outil de mesure d'angles articulaires précis, et comme une aide utile à un physiothérapeute dans ce contexte. Dans [47], les auteurs utilisent l'information de 5 IMU combinée avec des séquences vidéo pour classifier les mouvements d'enfants dans le cadre d'AGM. Cependant, ils s'intéressent au mouvement des IMU au cours du temps, et non pas à reconstruire la pose des enfants.

2.2.4 Conclusion sur les méthodes d'EPH

Dans cette section, nous avons donc vu plusieurs méthodes d'estimation de pose, dont nous résumons les forces et faiblesses dans le **Tableau 2.1**.

Dans notre contexte de suivi de patients aux soins intensifs, nous souhaitons avoir une méthode la moins intrusive possible pour le patient. De fait, placer des marqueurs d'un système MoCap ou des IMU ne semble pas adapté. À l'inverse, les matelas de pression semblent parfaitement adaptés, et de nombreuses solutions commerciales existent. Elles restent cependant très onéreuses, et nous avons vu que le traitement de cartes de pression nécessitait un traitement logiciel poussé.

De fait, l'utilisation d'un système de caméra monoculaire paraît être un bon compromis. Peu intrusive, une caméra placée au-dessus du lit d'un patient pourrait facilement être intégrée dans une salle de soins intensifs. De plus, le coût associé aux caméras (dépendamment de leurs caractéristiques) fait en sorte qu'il est parfaitement envisageable d'équiper toutes les chambres d'un service de soins intensifs. Quelle que soit la modalité choisie, nous pourrons nous appuyer sur un large éventail d'architectures de réseaux de neurones pour estimer la pose de patients. Mais attention : entraîner un réseau de neurones dans notre cas d'application est plus difficile qu'il n'y paraît.

2.3 Comment allons-nous entraîner notre réseau de neurones ?

2.3.1 Réseaux de neurones : avantages et limitations

Des modèles ultra-spécifiques Comme nous l'avons dit dans la **Section 2.2**, les réseaux de neurones ont pris progressivement l'exclusivité sur les méthodes d'EPH car ils sont excellents à performer la tâche à laquelle ils ont été entraînés. Cependant, Taori et al. montrent que les réseaux de neurones, entraînés à une tâche spécifique sur une base de données précise, sont incapables de performer la même tâche aussi bien sur une base de données différente ; c'est ce qu'on appelle un décalage de distribution (de l'anglais distribution shift) [48]. Nous allons

voir que ce problème de généralisation des réseaux de neurones s'applique particulièrement à l'EPH.

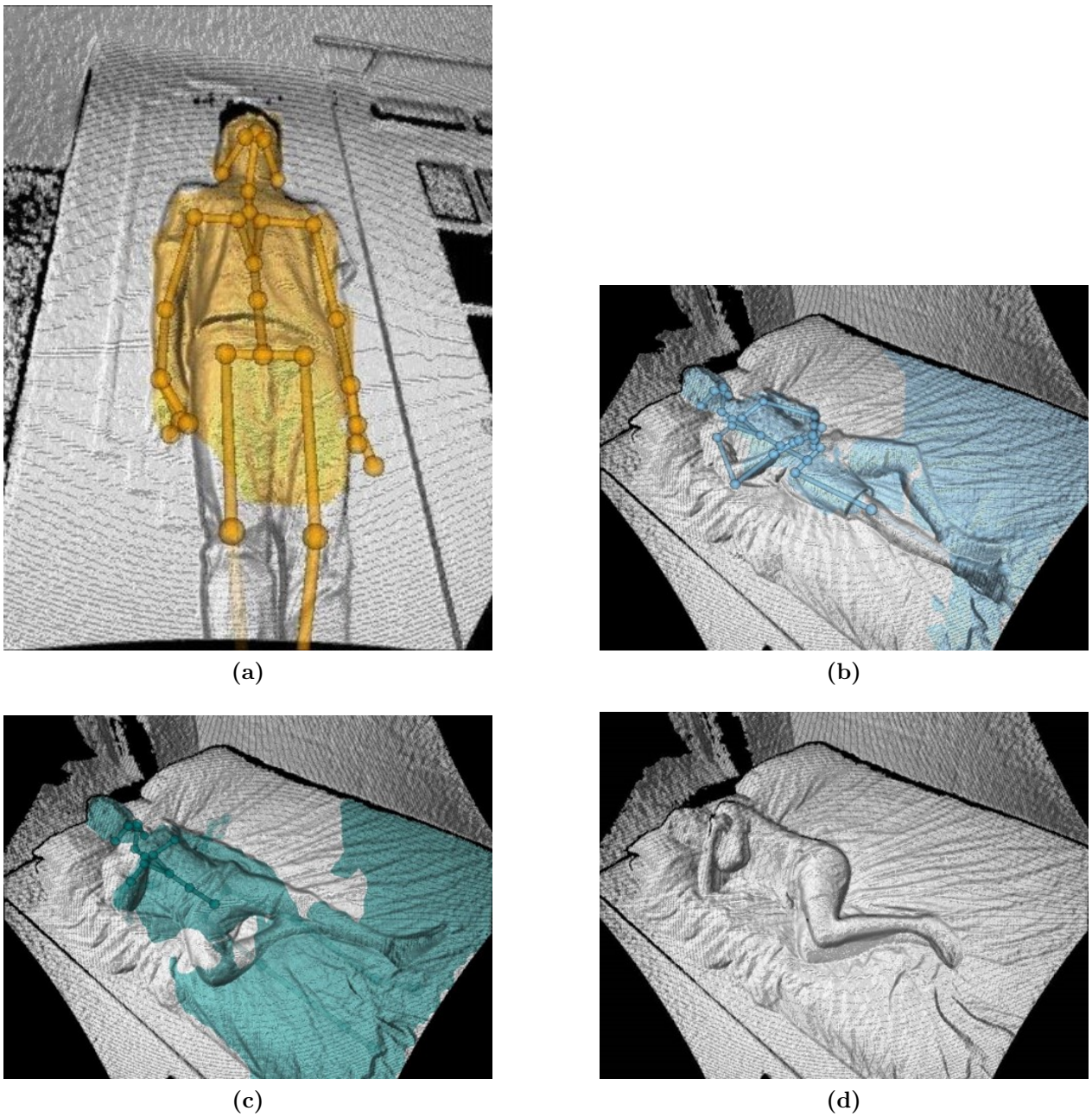
Les principales bases de données pour l'EPH Dans l'état de l'art, la plupart des bases de données existantes (MPII [49], COCO [50], etc.) sont composées de dizaines de milliers d'images d'adultes effectuant des activités de tous les jours : marcher, téléphoner, jouer au soccer, etc. Les images sont le plus souvent dans la modalité RGB, et annotées manuellement en 2D. Une autre base de données de grande envergure, Human3.6M [5], est composée de plusieurs millions d'images d'acteurs filmés avec une caméra de profondeur ainsi que d'un système de capture de mouvement à 4 caméras pour générer des annotations 3D.

Généralisation debout/couché Un des points communs de toutes ces bases des données ? Les personnes filmées sont le plus souvent debout. Ainsi, un réseau de neurones entraîné sur de telles bases a de la difficulté à estimer la pose d'une personne allongée. Ce problème est constaté notamment par les auteurs de [2] et illustré en **Figure 2.2** : ils utilisent le réseau de neurones intégré dans la suite logiciel de la Kinect et qui a été entraîné sur des personnes debout. Nos expérimentations avec ce même réseau donnent des résultats similaires, rapportés en **Figure 2.6**.

En analysant la **Figure 2.6**, on observe que le modèle retrouve parfaitement la pose d'une personne debout, adossée à un mur, les bras le long du corps (**2.6a**). Dans les cas où le sujet est allongé sur un lit (**2.6b**) et (**2.6c**), la partie supérieure du corps est bien reconstituée mais pas la partie inférieure : on constate que l'ouverture du bassin de la personne filmée est possible et naturelle en étant allongé, mais pas du tout en étant debout. Enfin, dans le cas d'une pose parfaitement spécifique à une personne qui dort allongée sur le coté (**2.6d**), l'algorithme est incapable de détecter quoi que ce soit.

Généralisation adulte/enfant Similairement au problème de généralisation debout/couché, on retrouve le problème de réseaux entraînés sur des adultes à généraliser aux enfants. Selon les auteurs de [51], ceci est notamment dû aux grandes différences de morphologie entre l'adulte et l'enfant.

Nous allons désormais analyser en profondeur les différentes bases de données développées en réponse à ces deux problèmes particuliers de généralisation.



(a) **(b)** **(c)** **(d)**

FIGURE 2.6 Exemple des résultats obtenus avec l'algorithme d'EPH intégré avec la caméra de profondeur Microsoft Kinect Azure. (a) Pour une personne debout, (b) et (c) Poses similaires à une posture debout (d) Posture propre à une personne qui dort [9].

2.3.2 Une base de données pour la généralisation debout/couché

Devant la nécessité d'avoir une base de données spécifique au contexte hospitalier, les auteurs de [4] ont créé le Simultaneously-Collected Multimodal Lying Pose Dataset, mieux connu sous le nom de SLP. Elle est composée d'images de 102 participants (28 femmes) allongés dans

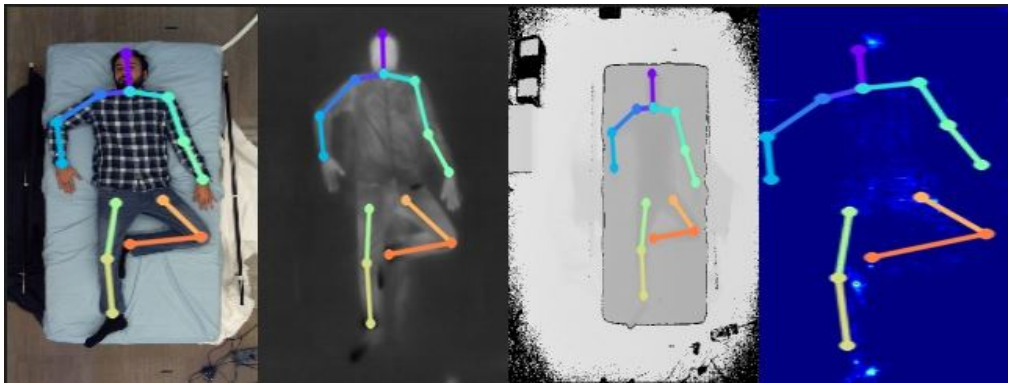
un environnement de chambre à coucher (appelé environnement *danaLab*), et 7 participants (3 femmes) dans une chambre d'hôpital simulée (environnement appelé *simLab*). L'environnement de l'hôpital diffère de celui de la chambre à coucher sur plusieurs points : hauteur de plafond différente, lit différent (utilisation d'un lit d'hôpital), draps et couvertures de différentes marques et couleurs et enfin de nouveaux participants (c'est-à-dire qui n'apparaissent pas dans les données de configuration de la chambre). Pour chaque participant, 45 poses différentes ont été enregistrées : 15 sur le dos, 15 en position latérale gauche et 15 en position latérale droite. Pour chaque pose, les données ont été collectées dans 3 conditions de couverture - sans couverture, couverture fine, couverture épaisse - et en utilisant 4 modalités d'imagerie - RGB, infrarouge lointain (LWIR), profondeur, carte de pression - simultanément. Cela nous amène à $102 * 45 * 3 = 13770$ images par modalité pour danaLab, et $7 * 45 * 3 = 945$ images par modalité pour simLab. Les caméras RGB, IR et de profondeur sont installées au plafond, côte à côte. Les cartes de pressions sont générées par des capteurs placés dans le matelas.

Malgré le fait que cette base de données contienne des images multimodales de sujets en posture couchée et partiellement dans un environnement hospitalier, elle ne se prête pas à notre cadre d'application pour deux raisons : les sujets sont des adultes et les annotations sont en 2D. De plus, il est important de noter quelques limitations dans la méthode des auteurs.

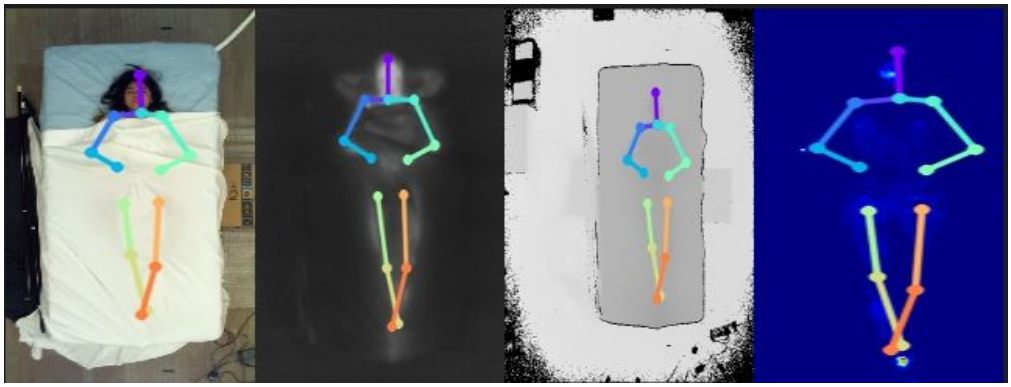
Un processus d'annotation laborieux Les auteurs annotent les images dans chacune des modalités manuellement. Ceci à l'inconvénient d'être extrêmement fastidieux et long. Il est aussi moins évident d'agrandir la base de données avec de nouveaux patients.

Les annotateurs sont incapables d'annoter les images RGB dès qu'une couverture est posée. Pour contrer cela, les auteurs demandent au patient de prendre une certaine pose, et de rester immobile pendant qu'ils lui mettent une couverture et prennent un nouveau cliché. Ce processus, répété 2 fois pour chaque type de couverture, est aussi pénible pour le patient qui est contraint de rester immobile.

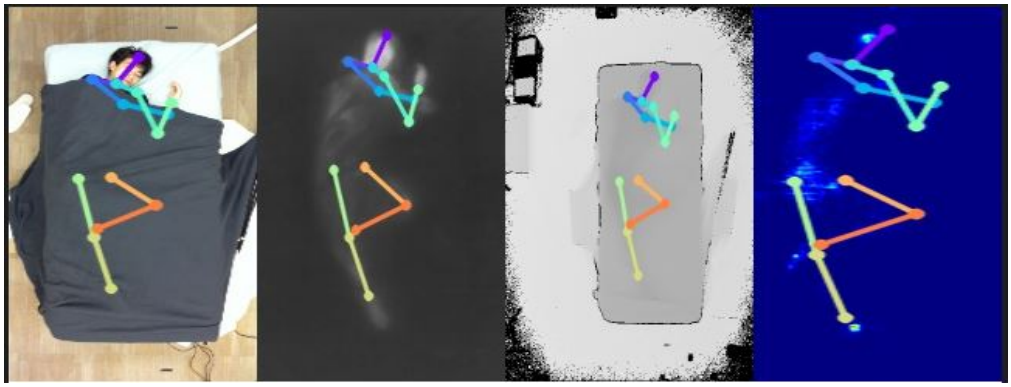
Un manque de précision des annotations Comme nous l'avons dit, chaque modalité est annotée manuellement et indépendamment. Or, dans le cas des images thermiques, ceci amène deux limitations. Premièrement, lorsque deux membres du corps se superposent (par exemple un bras posé sur le ventre), les annotateurs sont incapables de distinguer précisément les deux membres, et l'annotation devient impossible. Pour obtenir les annotations IR, ils



(a)



(b)



(c)

FIGURE 2.7 Exemple d'images de SLP [4]. Annotations des poses dans chaque modalité pour différents types de couvertures : (a) Aucune, (b) Couverture fine, (c) Couverture épaisse. De gauche à droite, les modalités RGB, IR, profondeur et cartes de pressions.

appliquent donc une transformation homographique sur les annotations RGB. Cependant, ceci nécessite de faire l'hypothèse que le corps du sujet est une surface plane, ce aui n'est pas le cas. De fait, cette approximation introduit un biais dans l'annotation thermique dépendant de la morphologie du sujet, de la distance des caméras au sujet et de la distance entre elles.

Même si le biais est borné selon les auteurs, une approche plus précise aurait été de réaliser une stéréocalibration entre les deux caméras. Les auteurs de [4] mentionnent que ce n'est pas faisable pour des caméras de différentes modalités, pourtant certains travaux de calibration multimodale existent [52].

Deuxièmement, les résidus thermiques (dus à la précédente pose du patient qui a réchauffé le matelas) amènent des incertitudes dans le travail des annotateurs, et donc des annotations imprécises, voire erronées. Ce problème est relevé dans [3].

Un manque de réalisme Enfin, nous pouvons argumenter deux points de cette base de données qui traduisent un manque de réalisme : tout d'abord, les patients sont des personnes en pleine santé, et ayant toute une morphologie semblable. Les poses qu'ils doivent effectuer sont toutes identiques d'un patient à l'autre. Même s'il y a une grande variété de poses enregistrées, elles ne sont pas forcément représentatives des poses d'un patient alité dans un lit de soins intensifs par exemple.

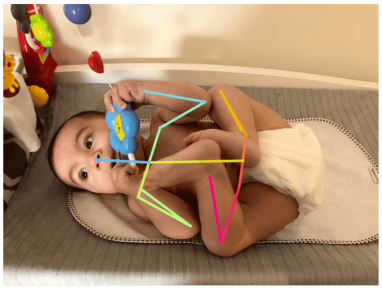
De plus, cette base de données est constituée d'images individuelles. Or, dans le contexte d'un suivi de patients en temps réel, ce n'est pas tant la pose absolue mais surtout son évolution au cours du temps qui nous intéresse. Il serait donc plus intéressant d'avoir des annotations de séquences vidéo.

Enfin, lors de la création de la base de données, les couvertures sont alternativement posées, puis enlevées du patient. De fait, les phénomènes de diffusion thermique n'ont pas le temps de se mettre en place. Il est donc probable que des réseaux de neurones entraînés sur ces images (où les segments du corps humain apparaissent bien définis) voient leur performance décroître lorsque les phénomènes de diffusion uniformiseront la température sous la couverture.

2.3.3 Deux bases de données sur les enfants

Comme nous l'avons vu en **Section 2.1.1**, l'EPH peut être utilisée pour l'analyse générale du mouvement de nourrissons. Dans ce but, les auteurs de [10] créent la base de données Mini-RGBD. Elle est composée de 12 séquences vidéo de nourrissons allongés sur le dos et vus du dessus. Cette base a cependant une particularité : elle est synthétique. En effet, les auteurs adaptent les mouvements d'enfants filmés avec une caméra RGBD à un modèle volumétrique de nourrisson [53]. En rajoutant une texture et un arrière plan aux modèles, ils arrivent à générer des images réalistes d'enfants filmés avec une caméra RGBD vus du dessus. L'avantage de cette méthode est qu'elle permet de créer une base de données annotée automatiquement, les coordonnées 3D des points clés du squelette étant extraites du modèle paramétrique. Un exemple du rendu couleur de Mini-RGBD est montré en **Figure 2.8a**.

(a)



(b)

(c)

FIGURE 2.8 Exemple d'images de deux bases de données d'enfants. (a) Mini-RGBD [10] (b) Partie réelle de SyRIP (c) Partie synthétique de SyRIP [11]

.

Cependant, les auteurs de [11] avancent deux limitations à la base de données Mini-RGBD. Premièrement, ils argumentent que les poses générées sont trop simplistes, et ne sont pas représentatives de la variété de poses que peuvent prendre des nourrissons. Deuxièmement, ils soulignent que les images étant synthétiques, un réseau de neurones entraîné sur cette base de données ne saurait généraliser à un contexte réel. Ceci est directement relié au problème de décalage de distribution cité en **Section 2.3.1**. Ils proposent donc une nouvelle base de données, SyRIP (Synthetic and Real Infant Pose), qui est composée à la fois d'images réelles d'enfants extraites de Google ou Youtube (500 images annotées manuellement en 2D), et d'images synthétiques similaires à celles de Mini-RGBD (1000 images synthétiques avec annotations 2D et 3D). À l'inverse de Mini-RGBD, ils ne s'intéressent pas à l'information de profondeur dans leurs rendus synthétiques, et ne contiennent donc que des images RGB. De fait, SyRIP n'est pas non plus adaptée à notre cas d'application, puisqu'elle contient elle aussi des images synthétiques, et uniquement en modalité RGB.

2.3.4 Conclusion sur les bases de données existantes

Nous avons vu que les bases de données classiques d'EPH ne sont pas adaptées à notre cas, notamment dû au problème de généralisation des réseaux de neurones. Si certaines bases de données plus spécifiques à la posture couchée et aux nourrissons ont vu le jour dans de récents travaux, elles restent peu adaptées à notre cas d'utilisation.

2.4 Conclusion de la revue des connaissances

De cette revue de littérature, trois informations essentielles sont à retenir. Premièrement, que l'EPH a un clair potentiel dans le monde médical, notamment dans le suivi en temps réel

de l'état de santé de patients. Deuxièmement, que l'EPH résulte d'une combinaison entre systèmes d'acquisition matériel (caméras, marqueurs, capteurs inertiels, etc.) et intégration logicielle (réseaux de neurones, cinématique inverse, etc.) ; toutes ces approches présentent leurs forces et faiblesses respectives. Troisièmement, nous avons conclu qu'utiliser des réseaux de neurones sur des images obtenues par des caméras placées en tête de lit du patient était la solution la moins intrusive, la plus abordable financièrement et facilement déployable pour une intégration aux soins intensifs du CHUSJ.

TABLEAU 2.1 Résumé des différentes modalités envisageables pour l'EPH dans le cadre de patients alités.

Méthode	Type de données	Avantages	Inconvénients
Monoculaire	Images RGB	— Beaucoup de travaux utilisent cette modalité — Est peu dispendieux — Est portatif	— Est sensible aux occlusions et à l'illumination
	Images de profondeur	— Résiste aux changements d'illumination — Est insensible à la couleur de la peau — Donne une information 3D — Est peu dispendieux — Est portatif	— Est sensible aux occlusions
	Images IR	— Résiste aux changements d'illumination — Est peu sensible aux occlusions — Est portatif	— Sensible à la diffusion thermique au cours du temps
MoCap	Images IR & positions 3D des marqueurs	— Offre une haute précision en 3D (référence)	— Est sensible aux occlusions — Est très intrusif — Nécessite une calibration régulière — Est très dispendieux — Nécessite du post-traitement — Le sytème est difficilement portatif
IMU	Orientations et Accélérations	— Offre une reconstruction 3D — Est insensible aux occlusions/illumination — Le système est portatif	— Nécessite une pose de calibration du sujet — Souffre d'une dérive des mesures au cours du temps — Est intrusif (moins que MoCap)
Matelas Connectés	Cartes de pression	— Directement intégré dans le lit du patient — Est insensible aux occlusions et à l' illumination	— Offre une faible précision — Prétraitement lourd pour analyse de données — Onéreux

CHAPITRE 3 PROBLÉMATIQUE ET OBJECTIFS SPÉCIFIQUES

De notre revue de connaissances, nous avons conclu que nous souhaitions privilégier une approche basée sur des réseaux de neurones. Cependant, nous avons vu que cette méthode requièrent un entraînement sur de grandes bases de données contenant des images annotées et spécifiques à notre cas d'application. À ce sujet, nous pouvons dresser 4 constats.

C1 : Les algorithmes entraînés sur des personnes en posture debout ne parviennent pas à généraliser à des postures allongées.

C2 : La grande différence de morphologie chez l'enfant rend les algorithmes entraînés sur des adultes inefficaces pour l'estimation de pose d'enfants, surtout les plus jeunes.

C3 : L'utilisation d'images RGB pour estimer la pose est insuffisante dans les cas d'occlusions et de variations d'illumination. Nous supposons que des images de profondeur et en infrarouge peuvent être une alternative.

C4 : Un suivi pertinent des patients nécessite une estimation de la pose en 3D, notamment dans les cas d'auto-occlusion où une partie du corps en cache une autre. Or la majorité des algorithmes développés à ce jour se concentrent sur l'EPH en 2D.

Ces constats soulèvent la problématique principale à laquelle nous sommes confrontés : il n'existe à ce jour aucune base de données réelle (i.e. non synthétique), multimodale(C3), d'enfants(C2) allongés (C1) et avec des annotations de pose en 3D (C4). D'où l'objectif principal de ce projet de recherche :

Objectif principal du projet de maîtrise :
Élaborer un protocole de création d'une base de données multimodale avec annotations de pose en 3D

Cet objectif principal se décompose en 4 objectifs spécifiques (OS) :

OS 1 : Déterminer quelle modalité ou combinaison de modalités est la plus adaptée pour l'EPH dans un contexte hospitalier.

OS 2 : Mettre en place un système d'acquisition vidéo multicaméras (une pour chaque modalité).

OS 3 : Mettre en place un système de génération d'annotations 3D automatique ou semi-automatique.

OS 4 : Faire approuver le protocole par le comité d'éthique du CHUSJ afin de pouvoir populer la base de données.

L'OS1 sera abordé dans le **Chapitre 4**. Nous verrons que la combinaison d'images de profondeur et thermique est la plus pertinente. L'OS2 sera abordé dans le **Chapitre 5**. Nous parcourrons toute la chaîne d'acquisition de séquences vidéo, de la calibration de caméras à l'écriture de données au disque en temps réel. L'OS3 sera abordé dans le **Chapitre 6**. Nous investiguerons l'utilisation de capteurs inertiels IMU pour générer les poses 3D d'un modèle cinématique du corps. Dans ce mémoire, nous ne détaillerons pas l'ensemble des procédures menées pour l'OS4, mais affirmons que le comité éthique a donné son accord pour ce projet. Dans l'**Annexe C**, nous donnons la lettre d'approbation éthique signée par le comité d'éthique du CHUSJ.

CHAPITRE 4 TRAVAIL PRÉLIMINAIRE : CHOIX DE MODALITÉS D'IMAGERIE

Comme nous l'avons vu dans la revue des connaissances, chaque modalité d'imagerie - RGB, profondeur et IR - présente différentes forces et faiblesses. Pour créer notre base de données, nous devons avant tout déterminer quelle modalité ou combinaison de modalités serait la plus adaptée pour faire de l'EPH.

4.1 Matériel

Pour ce travail préliminaire, nous allons considérer la base de données SLP [4], que nous avons déjà présentée en **Section 2.3.1**. Si nous y avons énuméré certaines de ses limitations, elle reste néanmoins très proche de notre d'application. Premièrement, elle contient des images acquises simultanément dans les trois modalités d'imagerie à l'étude et deuxièmement, ces images (notamment la partie simLab) sont acquises sur des sujets couchés dans un lit d'hôpital.

Dans leur article de présentation [4], les auteurs s'intéressent aux performances de 6 différentes architectures de réseaux de neurones sur les différentes modalités. Ils y analysent les performances de ces réseaux sur la base de données dans son entièreté, en mélangeant tous les types de couvertures dans la phase d'entraînement et de test (cf. **Figure 2.7**). Pour faciliter la compréhension dans la suite de cette section, nous définissons trois niveaux d'occlusion :

- **Niveau 0** Absence totale d'occlusions qui correspond au cas sans couverture.
- **Niveau 1** Occlusion légère, qui correspond à la couverture fine (1mm d'épaisseur)
- **Niveau 2** Occlusion importante, qui correspond à la couverture épaisse (3mm d'épaisseur).

Or, nous avons déjà vu que les occlusions représentaient un des défis majeurs dans notre cas d'application. Il devient donc nécessaire d'analyser comment les performances d'un réseau de neurones évoluent avec le niveau d'occlusion.

4.2 Méthode

4.2.1 Architecture utilisée

Cette section décrit brièvement le modèle que nous avons utilisé. Modèle de pointe en EPH, le High Resolution Net (HRNet) [12] est basé sur la combinaison de sous-réseaux de haute

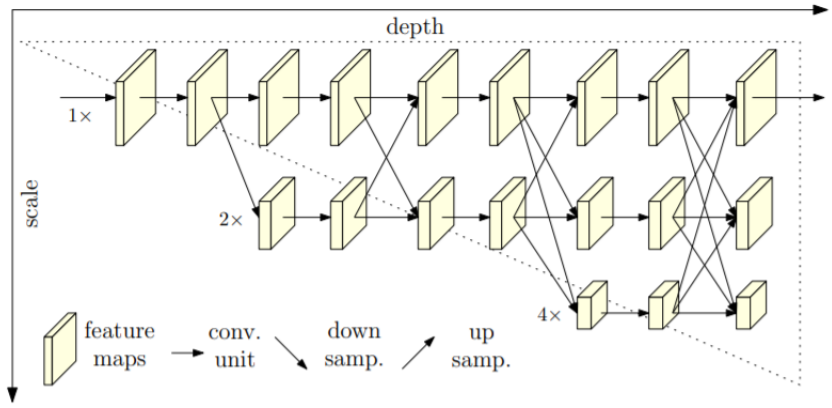


FIGURE 4.1 Illustration de l'architecture HRNet. Il se compose de sous-réseaux parallèles de haute à basse résolution avec échange répété d'informations entre les sous-réseaux. La direction horizontale (resp. verticale) correspond à la profondeur du réseau (resp. la résolution des cartes de caractéristiques). Source : [12]

à basse résolution qui sont connectés en série, comme le montre la **Figure 4.1**. On peut observer que le flux d'informations est continuellement partagé entre les différents niveaux de profondeur du réseau, ce qui, selon les auteurs, enrichit sa résolution. Nous allons entraîner ce réseau à détecter les 14 points clés 2D annotés dans la base de données SLP.

L'entrée de notre modèle est un tenseur PyTorch de taille (256, 256). Les images brutes sont donc redimensionnées lors d'une étape de prétraitement. La sortie du réseau est un jeu de 14 cartes de chaleurs (que l'on nommera *heatmaps*) de taille (64, 64). Chaque heatmap contient une gaussienne 2D dont le centre correspond à la position du point clé dans l'image d'entrée. Dans la **Figure 4.2**, nous montrons l'entrée et la sortie superposées. Pour une meilleure lisibilité, les heatmaps ont été redimensionnées pour correspondre à la taille de l'image d'entrée, et nous ne montrons que deux points clés, à savoir la cheville et l'épaule gauches.

4.2.2 Entraînement

Pour mener à bien notre analyse, nous allons entraîner 4 fois la même architecture d'HRNet, mais sur des jeux de données différents. Nous garderons aussi les mêmes hyperparamètres. Ces derniers sont reportés dans le **Tableau 4.1**. Nous effectuons ce choix dans un but d'optimisation de temps. En effet, approfondir le choix d'hyperparamètres par des stratégies de grid-search est fastidieux, et résulte souvent en des gains de performances de l'ordre de quelques pourcent. Comme nous le verrons plus loin, dans certain cas, les résultats sont tellement faibles que l'on comprend que le choix des hyperparamètres n'est pas la principale

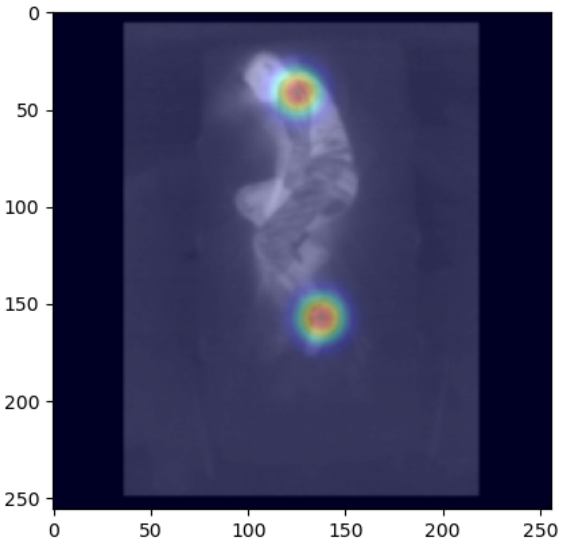


FIGURE 4.2 Illustration des entrées et sorties superposées de notre modèle. Ici, on considère une image d'entrée IR. Les heatmaps de sortie ont été redimensionnées pour être lisibles et s'adapter à la taille de l'entrée $(256, 256)$. L'articulation inférieure représente la cheville gauche, l'articulation supérieure représente l'épaule gauche.

cause d'erreurs.

Hyperparamètres du modèle	
Epochs	100
Batch size	60
Learning Rate	0.001
Learning Rate Decay	0.1 (epochs 70 and 90)
Optimizer	Adam

TABLEAU 4.1 Hyperparamètres du modèle HRNet utilisé

.

Dans le tableau **Tableau 4.2**, nous notons les différentes expériences que nous allons effectuer. La colonne *Origine* indique si nous avons utilisé un modèle préentraîné directement accessible depuis le repo Github (lien) de la base de données, ou si nous avons effectué nous-mêmes l'entraînement (via les serveurs Compute Canada). Les modèles seront entraînés sur $102 * 45 * 3 = 13770$ images de poses différentes dans chaque modalité. Ceci est du même ordre de grandeur qu'une base de données reconnue comme MPII par exemple, qui contient 25000 images. Notre socle d'entraînement paraît donc pertinent.

Expérience #	Entraînement	Test	Origine
1	danaLab/IR	simLab/IR	Préentraîné
2	danaLab/Profondeur	simLab/Profondeur	Préentraîné
3	danaLab/IR-Profondeur	simLab/IR-Profondeur	Préentraîné
4	danaLab/RGB	simLab/RGB	De zéro

TABLEAU 4.2 Liste de toutes les expériences menées.

Une fois que nous avons notre jeu de modèles différents, il s'agit de comparer leur performances. Nous procédons systématiquement de la même manière : lorsque l'on confronte deux modèles, nous allons regarder leurs performances à chaque niveau d'occlusion. L'exemple de la comparaison entre modèle IR et de profondeur est donné dans la **Figure 4.3**

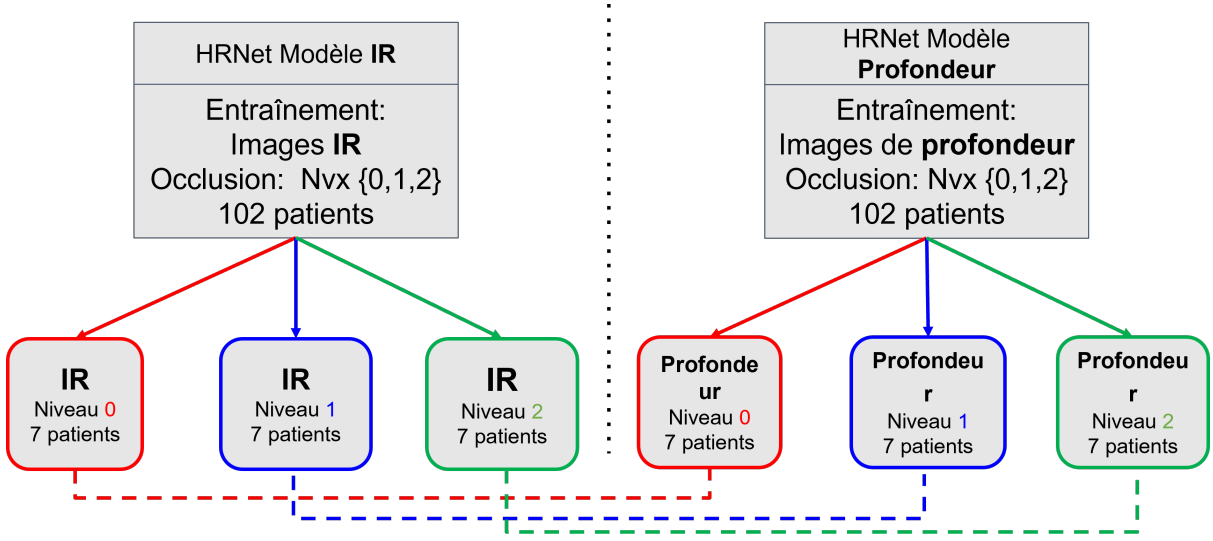


FIGURE 4.3 Exemple de notre méthodologie de comparaison entre deux modèles, dans le cas IR vs profondeur.

4.2.3 Métrique Utilisée : PCKh

Dans cette section, nous donnons une description précise de la métrique utilisée pour évaluer la performance d'un algorithme d'EPH : la *Probability of Correct Keypoint* (PCK, ou Probabilité de Point Clé Correct). La métrique PCK mesure la précision de la localisation des points clés du corps. Elle correspond à une simple distance euclidienne en pixels entre la position du point clé prédite et la vérité terrain. Pour une image quelconque x, on a donc :

$$PCK = \sum_{i \in PC} ||\phi_i(x) - y_i||^2 \tag{4.1}$$

Où ϕ_i est la prédiction de la position du point clé i dans l'image x, y_i sa position réelle, et PC l'ensemble des points clés considérés.

Le problème avec la métrique PCK est qu'elle n'est pas normalisée : la distance entre le point clé prédit et réel dépendant forcément de la taille du sujet, de sa position dans l'image, etc. Il est donc commun de comparer la distance entre une prédiction et la réalité à la longueur du segment nuque-tête de la personne considérée (notée h pour *head*). On introduit une fonction seuil $\psi_{\tau,h} = (\psi^1_{\tau,h}, \psi^2_{\tau,h}, ..., \psi^N_{\tau,h})$, où N représente le nombre de points clés que l'on essaie de reconstruire. Si la position du point clé prédite est suffisamment *proche* de la vérité terrain (en d'autres termes, leur distance est inférieure au seuil choisi), on considère que c'est une bonne prédiction (cf. **Équation 4.2**).

Ainsi, le PCKh devient la simple proportion entre le nombre de points clés correctement prédits et le nombre de points clés dans l'image (cf. **Équation 4.3**).

$$\forall i \in PC, \psi^i_{\tau,h}(x) = \begin{cases} 1 & \text{si } ||\phi_i(x) - y_i||^2 < \tau * h \\ 0 & \text{sinon.} \end{cases} \tag{4.2}$$

$$PCKh_\tau(x) = \frac{1}{N} * \sum_{i \in PC} \psi^i_{\tau,h}(x) \tag{4.3}$$

Dans la définition du PCKh, le seuil τ peut varier, et il est courant de voir la valeur de 50% dans les travaux d'EPH. Cependant, dans cette analyse nous examinerons les performances du modèle pour des valeurs de τ variant uniformément entre 0 et 0,5 et nous utiliserons toujours la notation $PCKh_\tau$.

4.3 Résultats et Discussion

4.3.1 Comparaison Profondeur vs Infrarouge : Expérience 1 vs Expérience 2

Résultats

La **Figure 4.4** présente une comparaison des performances obtenues par les modèles entrainés soit avec les images de profondeur, soit avec les images en infrarouge, en fonction du seuil τ sur la longueur du segment nuque-tête (h). À chaque couleur correspond un niveau d'occlusion.

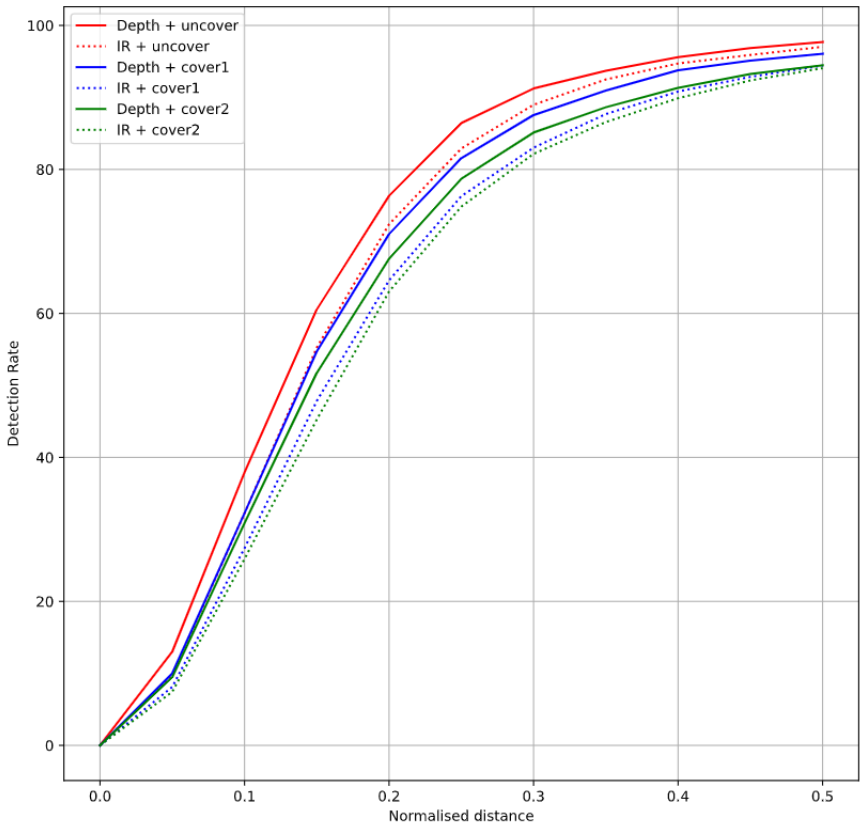


FIGURE 4.4 Évolution du PCKh des modèles de profondeur (traits pleins) et IR (traits pointillés) et pour différents niveaux d'occlusion : niveau 0 en rouge, niveau 1 en bleu, niveau 2 en vert.

Les résultats obtenus montrent essentiellement que quelle que soit la couverture, le modèle de profondeur surpasse le modèle IR. En effet, sur la **Figure 4.4**, pour tout type de couverture, la ligne pleine est supérieure à la ligne pointillée et ce quelle que soit la valeur du seuil PCKh.

Discussion

Tout d'abord, nous sommes satisfaits de voir que ces résultats sont fortement similaires à ceux obtenus dans [4]. En effet, les courbes que nous avons obtenues pour les modalités IR et de profondeur ainsi que celles de [4] montrent les mêmes tendances. Nous retrouvons aussi des valeurs de $PCKh_{0.5}$ autour de 95% ce qui est cohérent avec l'article.

Deuxièmement, nous observons que pour les deux modalités, le taux de détection diminue lorsque l'épaisseur de la couverture augmente (pour rappel, Niveau 0 = 0mm, Niveau 1 = 1mm, Niveau 2 = 3mm). Cette observation valide l'intuition que l'EPH est plus difficile en présence d'occlusions visuelles, comme c'est le cas pour l'œil humain dans le spectre visible.

Troisièmement, on observe que le modèle thermique est moins performant que le modèle de profondeur pour tous les niveaux d'occlusion. Cette moins bonne performance est probablement liée aux résidus thermiques de précédentes poses du patient, phénomène déjà mentionné en **Section 2.2**. Nous supposons en effet que ces résidus font moins bien ressortir les différentes parties du corps, rendant la tâche plus compliquée pour le réseau.
De cette expérience, nous concluons que **l'information de profondeur est plus robuste aux occlusions que l'information IR**.

4.3.2 Combinaison profondeur et infrarouge : Expérience 3

Nous avons montré que les images en profondeur donnent de meilleurs résultats que les images thermiques, mais nous n'avons pas pour autant prouvé que les données IR sont inutiles. D'où l'intérêt de l'expérience 3 qui combine les modalités de profondeur et IR dans le réseau. En pratique, l'entrée dans HRNet est devenue une concaténation des canaux IR et Profondeur, afin de créer un tenseur pyTorch $(2, 256, 256)$. Par conséquent, les entrées pour les tests sont également des images concaténées IR et Profondeur.

Résultats

Les résultats sont présentés dans la **Figure 4.5** ci-dessous. Comme nous avons vu que la profondeur surpasse IR, nous comparons les résultats IR+Profondeur uniquement avec les résultats de la profondeur pour une meilleure lisibilité.

Si l'ajout d'images IR aux images de profondeur ne montre pas d'amélioration significative en l'absence d'occlusions (courbe rouge sur la Figure 4.5), la différence en présence d'occlusions est plus marquée. De plus, on observe que les trois lignes pointillées sont plus proches les unes des autres, ce qui traduit une meilleure robustesse du modèle face à l'occlusion.

Discussion

Les résultats de l'expérience 3 permettent de conclure que **la combinaison des informations de profondeur et en infrarouge permet d'obtenir des modèles plus performants et robustes aux occlusions que des modèles entraînés avec une seule modalité.**

Il existe un point limitant avec notre modèle entraîné sur les images IR-profondeur : c'est le choix de la vérité terrain. En effet, nous avons vu que dans la base SLP, chaque modalité possède sa propre annotation manuelle. De fait, nous avons choisi arbitrairement au réseau

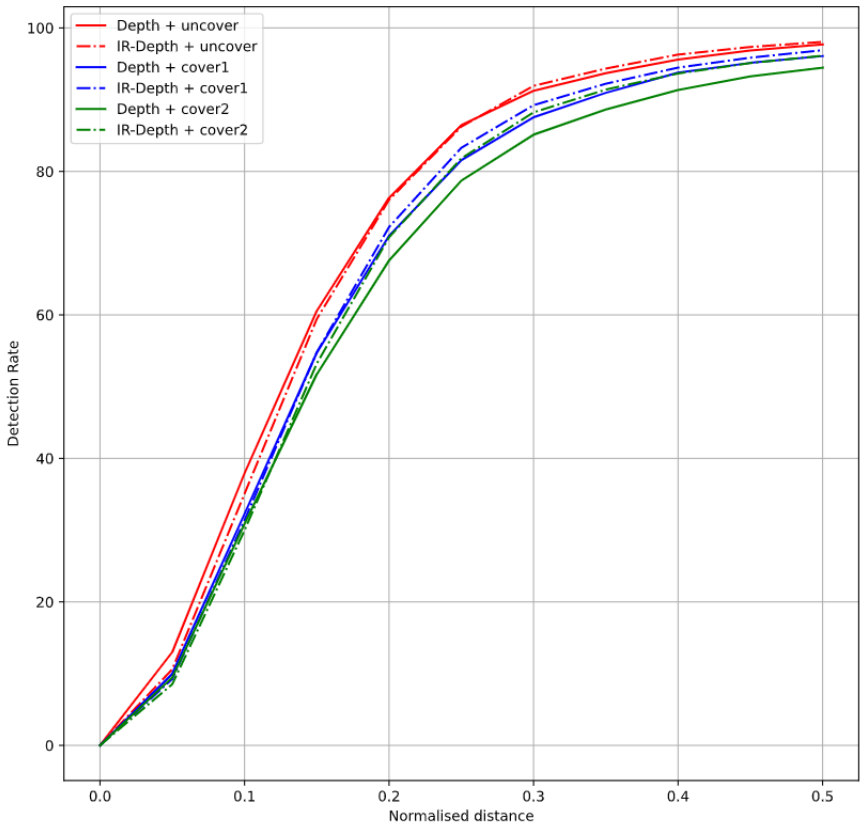


FIGURE 4.5 Évolution du PCKh de HRNet entraîné avec la modalité profondeur (traits pleins) ou en combinant profondeur et IR (traits pointillés), pour différents niveaux d'occlusion : niveau 0 en rouge, niveau 1 en bleu, niveau 2 en vert.

d'apprendre les positions des points clés dans l'image thermique, mais nous aurions pu tout autant choisir les annotations de profondeur.

En réalité, avoir deux vérités terrain pour une seule observation est problématique selon nous. Il faudrait plutôt réaliser un recalage multimodal entre les images. Une approche plus rigoureuse serait d'avoir une unique reconstruction cinématique de la pose en 3D, dans un référentiel commun aux deux caméras. Cependant, ceci nécessiterait de réaliser une stéréo-calibration entre les deux caméras, ce qui n'a pas été réalisé dans la base SLP.

4.3.3 Modalité RGB : Expérience 4

Nous avons déjà donné plusieurs arguments qui expliquent que la modalité RGB serait inefficace dans notre cas d'application, mais il paraît important de démontrer cela numériquement. Pour ce faire, nous entraînons un modèle sur les données RGB de danaLab, et le testons sur les données RGB de simLab.

Nous profiterons de cette expérience pour dresser une comparaison globale de nos 4 modèles (RGB, Profondeur, IR, IR+Profondeur). Pour ce faire, nous prenons la valeur de $PCKH_{0.5}$ pour chaque niveau d'occlusion, et reportons les résultats dans la **Figure 4.6**.

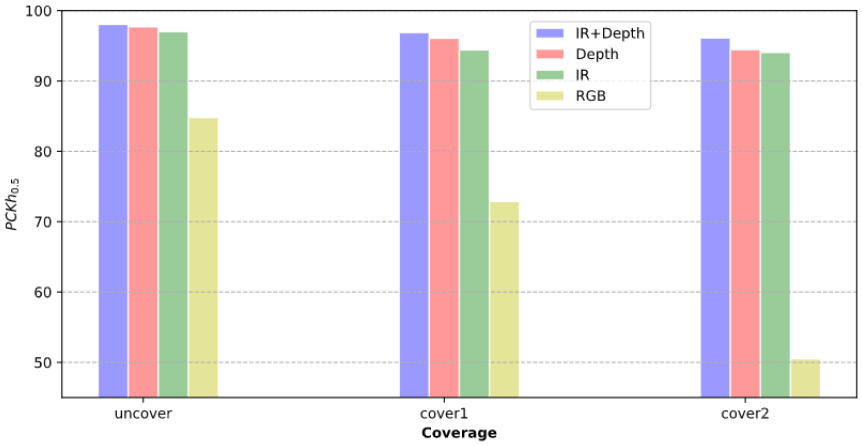


FIGURE 4.6 Comparaison des valeurs de $PCKH_{0.5}$ des différents modèles et pour les différents niveaux d'occlusion.

Les résultats présentés en **Figure 4.6** valident clairement la nécessité d'utiliser d'autres modalités que le modèle RGB pour l'EPH en contexte hospitalier. En effet, bien que le modèle RGB donne un $PCKH_{0,5}$ acceptable en l'absence d'occlusions (83%), nous observons une baisse drastique de la performance en leur présence : un $PCKH_{0,5}$ de 73% avec la couverture fine, et à peine supérieur à 50% avec la couverture épaisse.

Pour constater visuellement ce que représentent ces chiffres, nous proposons quelques résultats visuels en **Figure 4.7**. De gauche à droite, on observe la vérité terrain en RGB, puis les prédictions des modèles de profondeur, thermique et RGB dans cet ordre. En regardant la dernière colonne, on constate clairement que les prédictions du modèle RGB sont absurdes surtout en présence de couverture, en comparaison aux autres modèles.

4.3.4 Discussion

Lorsqu'on regarde plus en détail les performances de réseau RGB dans la dernière colonne de la **Figure 4.7**, il est intéressant de constater qu'il parvient toujours à retrouver la tête du sujet (sauf dans le cas de la dernière rangée) même dans les cas d'occlusions, car c'est le seul point clé visible. Ceci nous conforte dans l'idée qu'un réseau, comme un humain, n'est pas capable d'interpoler ce qui se passe sous la couverture. De plus, si les reconstructions paraissent absurdes, on peut constater qu'elles restent centrées autour de la tête du sujet,

comme si le réseau savait que les membres doivent se trouver proche de la tête, mais ne sait pas précisément où ils se situent.

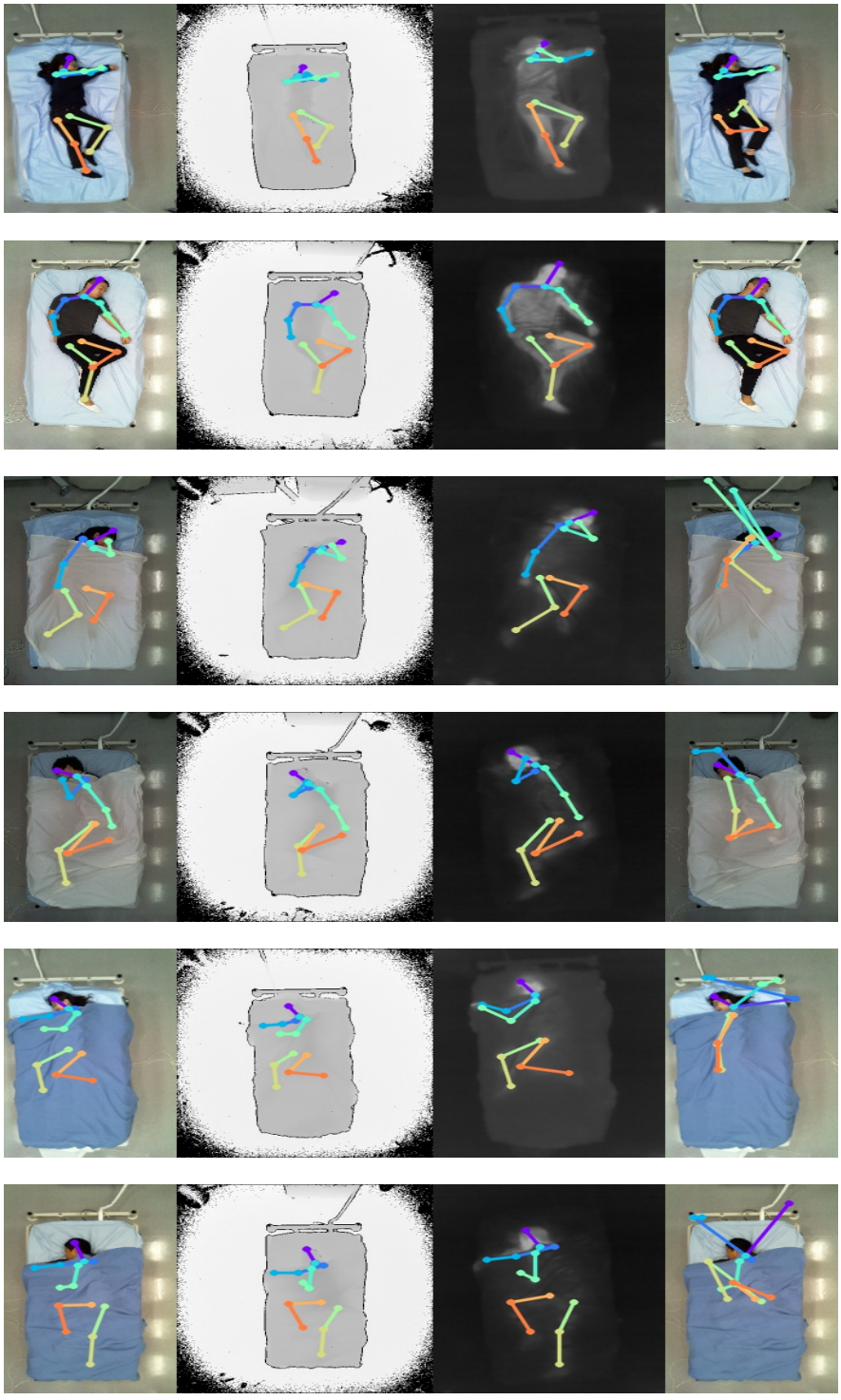


FIGURE 4.7 Divers exemples de nos résultats. De gauche à droite, les images représentent la vérité terrain, puis les résultats pour le modèle de profondeur, le modèle IR et le modèle RGB. Les 2 premières lignes correspondent au cas sans couverture, les 2 lignes suivantes à la couverture fine et les 2 dernières lignes la couverture épaisse.

4.4 Conclusion sur le choix de modalité

De cette étude, nous tirons des conclusions essentielles pour la suite de ce travail. Nous avons prouvé, sur la base de données SLP, que la modalité RGB est la moins adaptée à l'EPH dans un contexte hospitalier, surtout en présence d'occlusions visuelles. Nous avons aussi montré que l'imagerie thermique donnait des résultats légèrement moins bons que l'imagerie de profondeur, mais que la combinaison de ces deux modalités était la plus performante en terme d'EPH 2D et la plus robuste aux occlusions. Même si SLP se concentre sur des adultes, et que nous avons vu certaines de ses limitations, il paraît cohérent de considérer que ces résultats se transposeront au cas de l'estimation de pose 3D d'enfants.

Cependant, nous argumentons que pour avoir une combinaison pertinente des données de profondeur et thermique, nous devons travailler avec des images recalées. Lors de la création de notre base de données, nous devons donc travailler avec un système stéréo multimodal comportant une caméra de profondeur et une caméra thermique.

CHAPITRE 5 MISE EN PLACE D'UN SYSTÈME D'ACQUISITION VIDÉO MULTIMODAL

Dans cette section, nous détaillons l'ensemble des étapes nécessaires à l'acquisition multimodale (RGB, profondeur et IR) des images de notre base de données. Nous commencerons par décrire le système de caméras. Puis, nous expliciterons les étapes de calibration de chacune des caméras afin de déterminer leurs paramètres intrinsèques, ainsi que les défis rencontrés. Ensuite, nous détaillerons le processus de stéréocalibration entre les caméras afin de les repérer l'une par rapport à l'autre dans l'espace. Enfin, nous présenterons le protocole de communication entre les caméras afin d'optimiser leur synchronisme d'acquisition.

5.1 Configuration du système de caméras

Suite à l'étude préliminaire présentée au chapitre précédent, notre système d'acquisition d'images combine une caméra de profondeur et une caméra thermique. Les sous-sections suivantes présentent chacune d'entre elles, ainsi que le montage physique des deux caméras.

5.1.1 Caméra de profondeur Kinect Azure

Pour acquérir l'information de profondeur, nous utilisons la caméra RGB-D Microsoft Kinect Azure. Cette caméra de profondeur à temps de vol (Time-Of-Flight TOF) présente de nombreux avantages. Elle offre des séquences d'images de profondeur à hautes résolutions spatiale (640x576 pixels) et temporelle (30 Images Par Seconde (IPS)). De plus, pour chaque image de profondeur acquise, la caméra fait l'acquisition en simultané d'une image RGB avec une résolution de 1280x720 pixels. Les images RGB et de profondeur étant recalées, il est possible de connaître la distance de chaque pixel RGB à la caméra. Dans la suite, nous distinguerons les deux sous-caméras Kinect RGB et Kinect Profondeur, qui sont toutes deux montées sur un unique système nommé Kinect.

La communication avec la Kinect est assurée par le Software Development Kit (SDK) fourni par Microsoft.

5.1.2 Caméra Thermique FLIR T1020

Nous utilisons comme caméra infrarouge le modèle portatif T1020 de la marque FLIR [54].

Contrairement à la caméra Lepton (160x120 pixels) utilisée pour la base de données SLP,

la T1020 présente l'intérêt d'avoir une très haute résolution d'image (1024x768 pixels) et une très grande sensibilité thermique (de l'ordre du milliKelvin). Cependant, elle présente un système optique non-fixe qui nécessite une mise au point à chaque fois que la caméra est allumée. La mise au point de l'image peut être réalisée soit manuellement, soit à l'aide d'un autofocus. Enfin, le SDK de la FLIR étant développé en C#, nous utilisons plus simplement OpenCV pour communiquer avec la caméra.

5.1.3 Montage physique des caméras

Les caméras sont placées côte à côte sur une plateforme métallique horizontale, elle même montée sur un bras de support articulé Manfrotto. Afin de s'assurer que les caméras aient leur centres optiques sensiblement à la même hauteur, nous avons créé un support adapté aux formes des caméras et à la plateforme métallique. Ce support a été modélisé en conception assistée par ordinateur, et imprimé à l'aide d'une imprimante 3D du PolyFab.

(a) **(b)**

FIGURE 5.1 (a) Modélisation du support des 2 caméras (b) Disposition des caméras : la FLIR en noir, la Kinect en gris sur le support 3D en orange, disposé sur la plateforme métallique. Le tout repose sur un bras Manfrotto portatif, ici fixé sur l'extrémité d'une table.

Les caméras sont fixées à l'aide de vis passant sous la plateforme, et serrées à l'aide de papillons. Les deux caméras sont liées de manière filaire aux ports USB d'un ordinateur portatif qui sera utilisé pour les acquisitions.

5.2 Calibration des caméras

5.2.1 Objectif de la calibration d'une caméra

La calibration d'une caméra consiste à déterminer les différents paramètres qui composent son modèle (voir **Annexe A**), à savoir sa matrice **K** de paramètres intrinsèques, la matrice **R** de paramètres extrinsèques d'une vue considérée, ainsi que ses coefficients de distorsion formant le vecteur **D**. Nous cherchons ainsi à obtenir :

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{R} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{T} \\ \mathbf{0}_{3\times1} & 1 \end{bmatrix} \quad \mathbf{D} = \begin{pmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{pmatrix}$$

Avec f_x et f_y les distances focales selon les deux axes principaux, et c_x, c_y les coordonnées du centre optique de la caméra.

5.2.2 Algorithme de calibration

Lors de l'étape de calibration, nous cherchons donc à retrouver 15 paramètres : 4 pour la matrice **K**, 6 pour la matrice **R** (3 angles de rotation, 3 coordonnées de translation), et 5 pour la matrice **D**. Nous allons utiliser pour cela l'algorithme présenté dans [55], et intégralement implémenté dans OpenCV. Nous présentons cependant les grandes lignes de cet algorithme. Le problème de calibration est résolu en 2 temps : calcul analytique de **K** et **R** sous l'hypothèse d'un modèle de sténopé, puis calcul de **D**.

Résolution analytique de K et R

Nous reprenons l'**Équation A.1**, mais dans le cas particulier où tous les points sont coplanaires (valeur de Z constante). En pratique, ceci est atteint en photographiant un objet plan, que l'on nommera *objet de calibration*. Sans perte de généralité, nous pouvons choisir de fixer Z=0, ce qui permet de simplifier cette équation à :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} \mathbf{r1} & \mathbf{r2} & \mathbf{T} \end{pmatrix} \begin{pmatrix} X_m \\ Y_m \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} X_m \\ Y_m \\ 1 \end{pmatrix} \tag{5.1}$$

Dans l'**Équation 5.1**, la matrice **H** est une matrice dite d'*homographie*. Elle présente 8 degrés de liberté, et peut donc être résolue analytiquement avec seulement la connaissance des coordonnées (X, Y) de 4 points sur l'objet de calibration ainsi que des coordonnées (u, v) des

points associés dans l'image. Cependant, comme nous avons 6 paramètres extrinsèques, seuls 2 paramètres intrinsèques peuvent être retrouvés à partir de cette homographie. Il faut donc considérer au minimum $n \geq 2$ homographies pour retrouver tous les paramètres intrinsèques, c'est-à-dire regarder l'objet de calibration dans au moins 2 positions 3D différentes (nous parlerons alors des différentes *vues* de l'objet).

Affinage de K et R et obtention de D

En pratique, pour des raisons de stabilité numérique, on utilise un nombre de points sur l'objet de calibration bien supérieur à 4. Pour la même raison, il convient aussi de regarder l'objet dans une grande variété de vues. Pour chaque vue i, les paramètres intrinsèques de la caméra $\mathbf{K}$ restent identiques, mais les paramètres extrinsèques $\{\mathbf{R}_i, \mathbf{t}_i\}$ changent.

L'étape de calibration peut alors être vue comme un problème d'optimisation d'une fonction appelée *erreur de rétroprojection*. En considérant n vues différentes d'un objet de calibration composé de m points distincts aux coordonnées connues, nous cherchons à minimiser :

$$E = \sum_{i=1}^{n} E_i = \sum_{i=1}^{n} \sum_{j=1}^{m} E_{im} = \sum_{i=1}^{n} \sum_{j=1}^{m} \|\mathbf{m}_{ij} - \hat{\mathbf{m}}(\mathbf{K}, \mathbf{D}, \mathbf{R}_i, \mathbf{T}_i, \mathbf{M}_j)\| \tag{5.2}$$

L'erreur de rétroprojection globale de notre calibration E est la somme des erreurs de rétroprojection dans chaque vue E_i. Le terme $\hat{\mathbf{m}}(\mathbf{K}, \mathbf{D}, \mathbf{R}_i, \mathbf{T}_i, \mathbf{M}_j)$ est la projection du point M_j de notre objet de calibration, suivant l'**Équation A.1**, tandis que $\mathbf{m}_{ij}$ est la position mesurée expérimentalement dans l'image.

Ce problème d'optimisation est résolu par un algorithme de Levenberg-Marquardt, qui prend comme solution initiale pour $\mathbf{K}$ et $\{\mathbf{R}_i, \mathbf{t}_i | i = 1...n\}$ les valeurs trouvées analytiquement en inversant l'**Équation 5.1**, tandis que les valeurs de $\mathbf{D}$ sont toutes initialisées à 0.

5.2.3 Création d'un objet de calibration

L'approche la plus classique pour calibrer une caméra RGB est de photographier un damier noir et blanc dont les dimensions des carrés sont connues. La différence de couleurs entre les cases permet une reconstruction facile de ses contours, et donc d'extraire précisément la position des points d'intersection des cases.

Cependant, nous voulons utiliser un objet de calibration qui soit utilisable à la fois pour notre caméra Kinect et notre caméra FLIR. Or un damier noir et blanc simple ne présente aucune propriété thermique (on pourrait argumenter que les cases noires absorbent plus de chaleur que les cases blanches, mais ceci ne transparaît pas en pratique) ni de propriété 3D

étant donné que c'est une surface plane. Nous utilisons donc une autre approche, qui est de détecter des cercles dans une planche trouée (cf. **Figure 5.2a**).

Visibilité par la FLIR

Au lieu de chercher des points d'intersection de cases comme dans un damier classique, nous cherchons désormais à retrouver les centres des cercles dans notre objet de calibration. Fabriqué en contre-plaqué, cet objet peut être chauffé en seulement quelques dizaines de secondes à l'aide d'un sèche-cheveux commercial. La différence de température au niveau des cercles ressort alors suffisamment bien à l'image thermique (cf. **Figure 5.2c**).

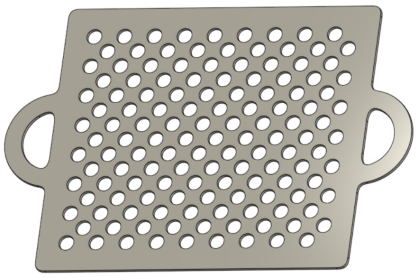
Visibilité par la Kinect RGB

Lors de la calibration d'une caméra RGB, OpenCV prend systématiquement en entrée une image ayant un seul canal. Comme les méthodes de calibration sont initialement développées pour travailler sur des images de damiers noir et blanc, les algorithmes s'attendent généralement à des images RGB converties en nuance de gris. Malheureusement, les cercles de l'objet de calibration ne ressortent pas clairement après cette conversion. L'astuce consiste à peindre notre objet de calibration dans une certaine couleur, et de garder uniquement le canal R, G ou B associé à cette couleur. En pratique, nous avons peint notre objet en orange, et effectué la soustraction du canal bleu au canal rouge. Le rendu est montré en **Figure 5.2b**. Les trous ressortent donc très bien, même en présence d'objets derrière la grille, et peuvent être détectés par la fonction d'OpenCV (cf. **Figure 5.2d**).

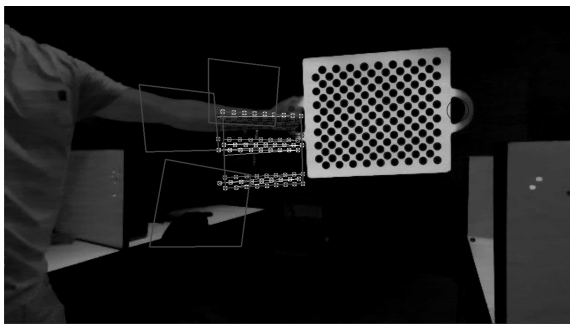
Visibilité par la Kinect profondeur

La grande différence de profondeur entre l'arrière-plan visible à travers les trous et la surface en bois au premier-plan permet à la Kinect profondeur de distinguer les différents cercles de notre objet de calibration. Cependant, sa faible résolution spatiale ne permet pas toujours de distinguer tous les cercles de l'objet, ce qui rend les résultats de la calibration imprécis.

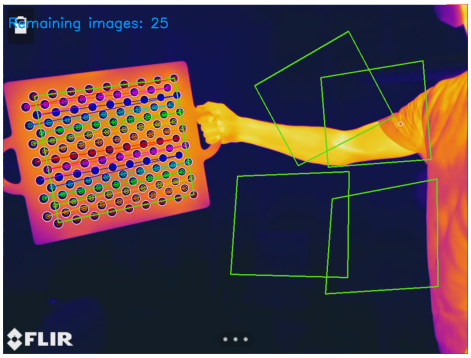
Comme les matrices de calibration de la Kinect sont données par le manufacturier, nous ne nous intéresserons donc pas à la modalité de profondeur dans la suite de ce travail. Ceci est d'autant plus pertinent que la stéréocalibration entre la Kinect profondeur et la Kinect RGB est performée par le manufacturier. De fait, nous pouvons nous contenter de travailler uniquement sur la modalité RGB



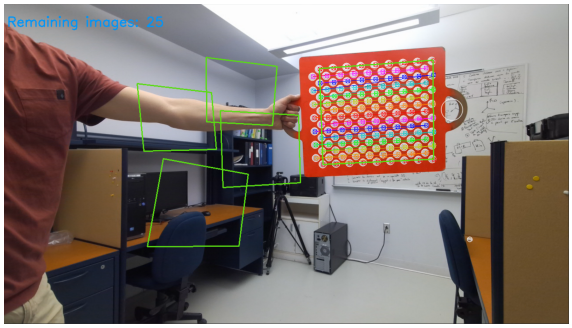
(a)



(b)



(c)



(d)

FIGURE 5.2 Notre objet de calibration. (a) Modélisé en CAO avant découpe (b) Vue de l'image RGB en soustrayant le canal bleu au canal rouge (c) Détection de cercles dans l'image thermique (l'objet ayant été chauffé au préalable) (d) Détection de cercles dans l'image RGB.

Spécifications

Notre objet de calibration final est composé de 9 lignes et 15 colonnes, pour un ensemble de 135 trous, chacun de 21,2mm de diamètre. Il a été perforé dans une planche de contre-plaqué de 0,5cm d'épaisseur, à l'aide de la graveuse laser du PolyFab. Nous avons aussi intégré deux poignées afin de permettre une manipulation plus aisée pour l'opérateur. Nous avons enfin appliqué deux couches de peinture orange à l'aide d'une bombe aérosol.

5.2.4 Implémentation de l'algorithme de calibration

Le processus global de calibration est bien développé et documenté par la librairie OpenCV. Nous proposons une interface graphique simple qui permet à l'utilisateur de réaliser la calibration avec un rendu visuel en temps réel. Le pseudo-code de notre méthode est donné dans l'**Algorithme 1**.

Les **Figure 5.2c et 5.2d** illustrent l'Interface Utilisateur (IU) lors du processus de calibra-

Algorithm 1: Protocole de calibration d'une caméra

Entrée: N images max, n images min
Sorties: Paramètres intrinsèques et de distorsion de la caméra

1 **Initialisation**
k le nombres de vues différentes trouvées
$k \leftarrow 0$;

2 **Acquisition**
tant que $k < N$ **faire**
Chercher les cercles dans l'image ;
si Tous les cercles de l'objet sont trouvés **alors**
Afficher les cercles sur l'UI ;
Enregistrer les positions des centres des cercles ;
$k \leftarrow k + 1$;
Attendre 2 secondes `// On laisse le temps au manipulateur de déplacer l'objet`
sinon
Aller à la prochaine image ;
fin
fin

3 **Calibration**
Calcul des matrices de calibration (**K** et **D**) et des erreurs de rétroprojection E.

4 **Validation**
Pour tout e *dans* E **faire**
`/* On regarde l'erreur moyenne des 135 centres et l'erreur maximale */`
si $\bar{e} \geq 1px$ *ou* $max(e) \geq 3px$ **alors**
Retirer l'image associée des images de calibration ;
$k \leftarrow k - 1$;
sinon
Continuer
fin
fin

5 **Conclusion**
si $k < n$ **alors**
Aller à l'étape 2 `// Il faut reprendre des images.`
sinon
si Au moins une image a été retirée **alors**
Aller à l'étape 3 `// On recalcule la matrice de calibration`
sinon
Terminer
fin
fin

tion. Les positions de l'objet de calibration qui ont été retenues par l'algorithme sont affichées sous forme de rectangles verts. Lorsqu'une nouvelle configuration est retenue, l'ensemble des cercles est colorié d'une manière différente afin de prévenir l'utilisateur. En haut à gauche, on affiche le nombre de vues restantes avant de passer au calcul des matrices de calibration.

Robustesse numérique

Lors de l'étape de calibration effectuée par OpenCV, chaque vue possède le même poids dans l'optimisation de Levenberg-Marquardt vue en **Équation 5.2**. Nous constatons cependant expérimentalement que dans certaines cas, certaines vues aberrantes peuvent faire tomber le problème d'optimisation dans un minimum local, et donner des résultats de calibration absurdes.

Pour contrer cela, une fois l'étape de calibration terminée, nous calculons l'erreur de rétroprojection E_{im} de chaque point m de l'objet et pour chaque vue i. Nous introduisons alors deux critères de seuillage sur l'erreur de rétroprojection de la vue. Si l'erreur maximale est supérieure à 3 pixels, ou si l'erreur moyenne est supérieure à 1 pixel, la vue est rejetée. Une fois que toutes les vues ont été considérées, le calcul des matrices de calibration est répété.

Cependant, nous devons garder un nombre suffisant de vues pour notre problème d'optimisation. Nous introduisons ainsi deux paramètres : N le nombre de vues différentes que doit prendre l'utilisateur avant de passer au problème d'optimisation, et n le nombre minimal d'images à garder pour calculer les matrices de calibration. Si au moins $N - n$ images sont rejetées, l'IU est relancée et l'utilisateur doit reprendre le nombre manquant de vues. Expérimentalement, nous avons constaté que les valeurs de $N = 30$ et $n = 25$ donnent lieu à des résultats satisfaisants.

5.2.5 Résultats et validation de la méthode

Via le SDK de la Kinect, nous pouvons accéder aux matrices intrinsèques et de distorsion dites "usines", c'est-à-dire obtenues par le manufacturier. Nous allons donc pouvoir utiliser ces valeurs comme vérité terrain pour valider notre méthode de calibration. Toutefois, les manufacturiers de la Kinect utilisent un modèle de distorsion différent de celui d'OpenCV, donc nous comparerons uniquement les valeurs des paramètres intrinsèques.

Métrique utilisée

Pour comparer nos résultats, nous nous intéressons aux erreurs moyennes absolues (RMSE : Root-Mean Squared Error) et relatives (RMSRE : Root-Mean Squared Relative Error), dont

les définitions sont données en **Équation 5.3**. Dans ces équations, y est une variable qui peut représenter soit les longueurs focales (f_x et f_y), soit les positions des centres optiques (c_x et c_y).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_{usine} - \hat{y}_i)^2} \qquad RMSRE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_{usine} - \hat{y}_i}{y_{usine}}\right)^2} \tag{5.3}$$

Calibration de la Kinect RGB

Comme précisé en **Section 5.2.3**, nous ne nous intéressons qu'à la modalité RGB de la Kinect. Nous en réalisons 6 calibrations indépendantes ($N = 30, n = 25$) et reportons les résultats obtenus dans le **Tableau 5.1**. Les résultats obtenus à l'issue de notre calibration sont extrêmement similaires à ceux de l'usine : sur les 4 paramètres intrinsèques, nous obtenons une erreur moyenne relative inférieure à 1.5%. De plus, cette erreur est concentrée autour des valeurs moyennes, avec un écart type relatif moyen pour sa part inférieur à 1.2%. Ceci indique que les résultats obtenus au travers de différentes calibrations sont globalement uniformes. Notre approche semble donc numériquement cohérente et stable.

Paramètre	Valeur Usine	Valeur Moyenne	RMSE	RMSRE (%)	$\sigma_{relatif}$(%)
f_x	600.5	599.2	3.76	0.75	0.79
f_y	600.2	599.6	3.76	0.73	0.79
c_x	638.4	633.4	4.93	0.82	0.29
c_y	365.9	369.7	5.21	1.49	1.17

Tableau 5.1 Résultats de la méthode de calibration sur la modalité RGB de la Kinect.

Nous choisissons donc comme valeurs des paramètres intrinsèques et de distorsion les valeurs moyennes obtenues sur ces 6 expériences. Ceci nous amène aux résultats suivants :

$$\mathbf{K}_{Kinect} = \begin{pmatrix} 599.2 & 0 & 633.4 \\ 0 & 599.6 & 369.7 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{D}_{Kinect} = \begin{pmatrix} 0.066 & -0.030 & 0.002 & -0.002 & 0 \end{pmatrix}$$

Calibration de la caméra thermique

En ce qui concerne la caméra thermique, nous ne possédons pas de matrice usine à laquelle nous comparer. En réalisant dix répétitions de la même expérience de calibration ($N = 30, n = 25$) indépendantes, nous pouvons cependant observer une certaine robustesse numérique de la méthode, comme montré dans le **Tableau 5.2**. Nous retrouvons en effet un

écart type relatif inférieur à 5% pour les paramètres intrinsèques, ce qui est sensiblement du même ordre de grandeur que pour la Kinect.

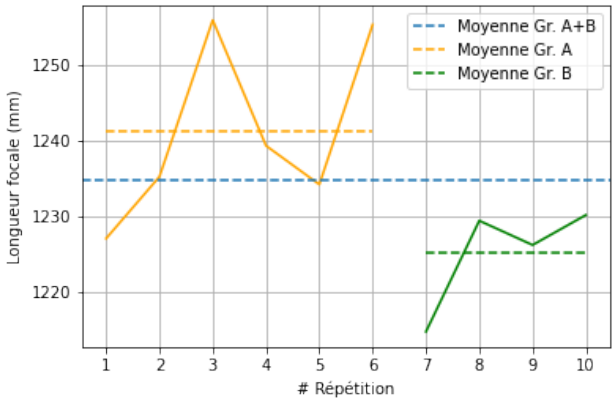


FIGURE 5.3 Différentes valeurs focales pour chaque répétition. En pointillés bleu, la valeur moyenne sur les 10 répétitions.

Paramètre	Valeur Moyenne	$\sigma_{relatif}(\%)$
f_x	1234.8	1.08
f_y	1234.8	1.04
c_x	482.2	2.98
c_y	396.9	4.39

TABLEAU 5.2 Résultats moyens de la calibration de la FLIR sur 10 expériences ($N = 30, n = 25$).

Impact de l'autofocus de la FLIR sur la calibration

Nous expliquons les plus grandes variations de la calibration de la caméra thermique par la nécessité d'effectuer une mise au point de la caméra après chaque redémarrage. En effet, les auteurs de [56] montrent que cette étape a un impact significatif dans le processus de calibration d'une caméra, et nous proposons une analyse quantitative pour le visualiser.

Pour les 6 premières expériences (Groupe A), nous n'avons rien modifié à la caméra thermique entre les expériences. Puis, nous l'avons éteinte, rallumée, et effectué une mise au point à 1 mètre grâce à l'autofocus. Nous avons ensuite procédé aux 4 dernières expériences de calibration (Groupe B). La **Figure 5.3** montre en détail l'évolution de la longueur focale f_x pour les deux différents groupes. En pointillés bleu, on montre la valeur moyenne sur les 10 expériences, aussi visible dans le **Tableau 5.2**. On peut clairement constater un avant et un après la mise au point via l'autofocus.

De fait, nous ne pouvons nous contenter d'utiliser une unique matrice de calibration de la FLIR pour toutes nos expériences. En réalité, la calibration de cette dernière sera obligatoire avant toute nouvelle acquisition. Nous pouvons cependant fournir à l'**Algorithme 1** une solution initiale pour son problème d'optimisation. Nous choisissons donc la matrice moyenne

obtenue sur les dix expériences citées, ce qui nous fait :

$$\mathbf{K}_{FLIR} = \begin{pmatrix} 1234.8 & 0 & 482.2 \\ 0 & 1234.8 & 396.9 \\ 0 & 0 & 1 \end{pmatrix} \qquad \mathbf{D}_{FLIR} = \begin{pmatrix} -0.037 & -0.071 & 0.004 & -0.006 & 0 \end{pmatrix}$$

5.2.6 Conclusion sur la calibration

Le protocole de calibration proposé est satisfaisant à la fois pour la précision des résultats qu'il propose, mais aussi par sa simplicité d'utilisation grâce à son interface graphique. Avoir un bon protocole de calibration était d'autant plus nécessaire que l'étape de stéréocalibration est entièrement basée dessus. C'est l'objet de la prochaine section.

5.3 Stéréocalibration des caméras FLIR et Kinect

Nous allons désormais considérer le cas de deux caméras qui filment ensemble une scène. Dans notre cas, ce sera donc les caméra Kinect et FLIR. Afin de rester dans un cas général, nous les nommerons caméras 1 et 2. La caméra 1 correspond à la FLIR et est placée à gauche, la caméra 2 correspond à la Kinect et est placée à droite (comme nous le présentons dans la **Figure 5.4**).

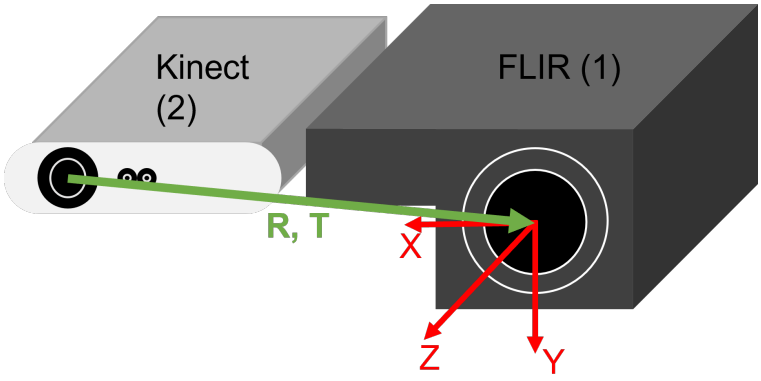


FIGURE 5.4 L'essentiel de la stéréocalibration est de retrouver les matrices **R** et **T**, qui expriment la transformation pour aller du repère de la caméra 2 à la caméra 1 (dans le repère de la caméra 1).

L'objectif de la stéréocalibration d'un ensemble de caméras est de déterminer l'ensemble des matrices qui coordonnent notre système. En pratique, ceci se traduit donc par le calcul des matrices **R** et **T**, et d'en déduire les matrices essentielle **E** et fondamentale **F** (pour cette dernière, si les caméras n'ont pas été préalablement calibrées, il faudra aussi retrouver les matrices de paramètres intrinsèques $\mathbf{K}_1$ et $\mathbf{K}_2$ des deux caméras). L'ensemble de la théorie de stéréocalibration est rappelé en A.

Nous avons déjà vu que le système stéréo RGB-Profondeur de la Kinect était calibré par les manufacturiers, et que nous obtenions des images recalées entre les deux modalités. En d'autres termes, nous connaissons déjà les matrices **R** et **T** de la paire RGB-profondeur. De fait, nous ne cherchons plus qu'à trouver les matrices **R** et **T** de la paire RGB-Thermique. Par simple opération matricielle, nous pourrons retrouver les matrices de la paire Profondeur-Thermique, et ainsi obtenir un système stéréo à 3 caméras calibré.

5.3.1 Implémentation d'OpenCV

Théorie analytique

Nous allons utiliser l'implémentation d'OpenCV pour la stéréocalibration de notre système. L'approche est assez similaire à l'étape de calibration et s'appuie sur le même principe de photographies d'un objet de calibration. Nous reprendrons le même objet que décrit en **Section 5.2.3**.

Pour chaque caméra, nous prenons une vue de l'objet de calibration. Par le même principe que vu en **Section 5.2.2**, nous pouvons reconstruire les matrices de paramètres extrinsèques de chaque caméra $[\mathbf{R}_i, \mathbf{T}_i], i \in \{1, 2\}$. Pour n'importe quel point M de l'objet de calibration (noté M_1 lorsque vu dans le repère de la caméra 1, M_2 lorsque vu dans le repère de la caméra 2), nous avons 3 équations :

$$\begin{cases} M_1 = \mathbf{R}_1 M + \mathbf{T}_1 \\ M_2 = \mathbf{R}_2 M + \mathbf{T}_2 \\ M_1 = \mathbf{R}^\top (M_2 - \mathbf{T}) \end{cases} \tag{5.4}$$

En combinant les différents termes du système d'équations ci-dessus, on peut directement retrouver :

$$\begin{cases} \mathbf{R} = \mathbf{R}_2 \mathbf{R}_1^\top \\ \mathbf{T} = \mathbf{T}_2 - \mathbf{R}\mathbf{T}_1 \end{cases} \tag{5.5}$$

Résolution Numérique

En pratique, nous prenons un grand nombre de vues de l'objet de calibration avec les deux caméras. Pour chaque paire de vues, OpenCV recalcule les paramètres extrinsèques et les utilise pour obtenir une matrice de rotation et translation grâce à l'**Équation 5.5**. La médiane des résultats dans chaque vue est utilisée comme la solution initiale pour le résultat final, qui est obtenu en optimisant l'erreur de rétroprojection dans chaque vue par un algorithme de Levenberg-Marquardt, similairement à ce que nous avons vu en **Section 5.2.2**.

5.3.2 Robustesse et contrainte épipolaire

Dans le cas de la calibration d'une caméra, nous avons vu que l'algorithme d'OpenCV nous renvoyait les paramètres extrinsèques pour chaque vue. Ceci nous permettait de calculer les erreurs de rétroprojection de chaque point de l'objet de calibration. De fait, nous pouvions retirer les vues qui amenaient des valeurs aberrantes dans ce calcul.

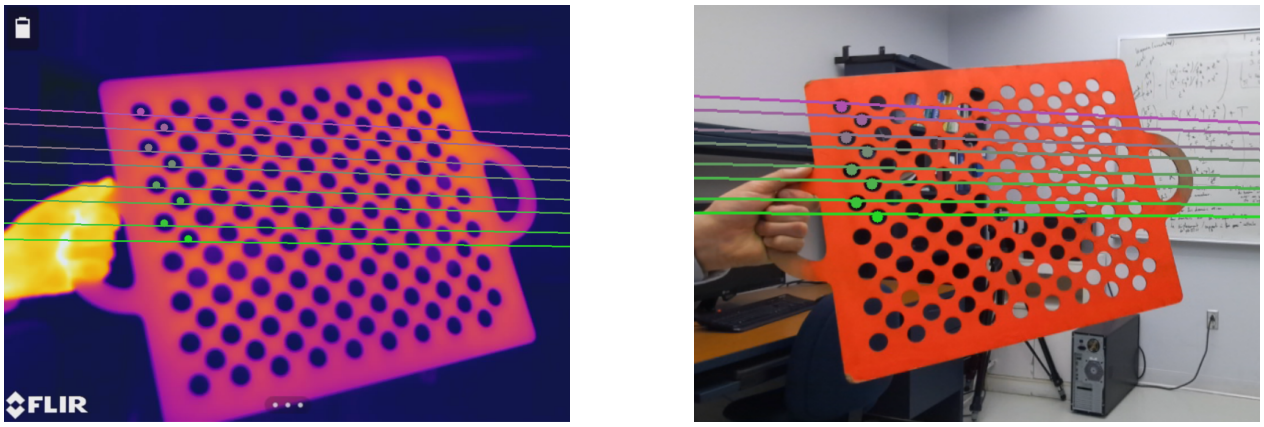
Cependant, dans le cas de la stéréocalibration, OpenCV ne fournit pas les paramètres ex-

trinsèques de chaque vue. Seule l'erreur de rétroprojection finale est renvoyée, en plus des matrices recherchées : **E**, **F**, **R** et **T**. Nous introduisons donc une nouvelle métrique afin de quantifier la précision des sorties de l'algorithme de stéréocalibration : l'*erreur de contrainte épipolaire*. Elle est définie pour tous les points par :

$$E = \frac{1}{n}\sum_{i=1}^{n} E_i = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m} {w_2^{ij}}^{\top} \mathbf{F} w_1^{ij} \tag{5.6}$$

En effet, d'après l'**Équation A.11**, dans un système parfait, cette erreur devrait valoir 0. Ici, nous calculons l'erreur pour chaque point j de l'objet de calibration, observé dans la vue i, soit par la caméra 1 (w_1^{ij}), soit par la caméra 2 (w_2^{ij}).

Pour visualiser physiquement ce que représente l'**Équation 5.6**, nous montrons en **Figure 5.5** quelques lignes épipolaires pour chaque vue. Pour rappel, si nous considérons un point w_1 dans la vue 1, la ligne épipolaire associée dans la vue 2 l_2 est obtenue selon $l_2 = \mathbf{F}w_1$. L'erreur de contrainte épipolaire représente donc la distance moyenne de chaque point à sa ligne épipolaire associée.



(a) **(b)**

FIGURE 5.5 Visualisation de lignes épipolaires dans chaque vue. Pour chaque paire de points, une couleur associée. Plus la stéréocalibration est correcte, plus la ligne épipolaire passe par le point associé.

De manière totalement analogue au cas de la calibration, nous introduisons les paramètres N et n pour l'algorithme de stéréocalibration. Expérimentalement, nous avons constaté que les valeurs de $N = 30$ et $n = 25$ donnent lieu à des résultats satisfaisants.

5.3.3 Implémentation globale de l'algorithme de stéréocalibration.

Sur le principe, l'algorithme de stéréocalibration est sensiblement pareil à l'**Algorithme 1**, à l'exception que l'objet de calibration doit être vu par deux caméras simultanément. Pour ne pas surcharger cette section, nous supposerons que l'algorithme reçoit en simultané les images des deux caméras. Nous verrons plus en détail dans la **Section 5.4** comment ceci est possible.

5.3.4 Résultats de la stéréocalibration.

Interprétation physique des résultats.

En sortie de notre algorithme, nous rappelons que nous obtenons 4 matrices : les matrices essentielle et fondamentale **E** et **F** ainsi que la matrice de rotation **R** et de translation **t** entre les deux caméras, qui amènent le repère de la caméra de droite sur la caméra de gauche. Dans notre cas, nos deux caméras sont quasiment parallèles, et situées côte à côte. Théoriquement, nous devrions trouver des résultats proches de :

$$\mathbf{R}_{théorique} = \mathbf{I}_{33} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \mathbf{T}_{théorique} = \begin{pmatrix} T_x & 0 & 0 \end{pmatrix} \tag{5.7}$$

Nous réalisons un jeu de 10 expériences et trouvons en moyenne les matrices ci-dessous. Dans le cas de la matrice de translation, les grandeurs sont en millimètres.

$$\mathbf{R}_{expérimental} = \begin{pmatrix} 0.999 & 0.008 & -0.008 \\ -0.008 & 0.999 & 0.042 \\ 0.008 & 0.042 & 0.999 \end{pmatrix} \quad \mathbf{T}_{expérimental} = \begin{pmatrix} -222.6 & 4.1 & -4.8 \end{pmatrix} \tag{5.8}$$

En mesurant physiquement la distance entre les centres des objectifs des deux caméras avec un pied à coulisse, nous trouvons une valeur de 225mm, ce qui est tout à fait cohérent.

Robustesse numérique

Par analogie au cas de calibration, nous analysons aussi la robustesse numérique de notre méthode en réalisant dix expériences de stéréocalibration identiques. À chaque expérience, nous réalisons une calibration de la caméra thermique ($N = 30, n = 25$), suivie immédiatement d'une stéréocalibration du système ($N = 30, n = 25$). Comme les valeurs de translation

Algorithm 2: Protocole de stéréocalibration de deux caméras

Entrée: N images max, n images min
Sorties: **E**, **F**, **R**, **T**

1 **Initialisation**
k le nombres de vues différentes obtenues
$k \leftarrow 0$;
V les positions en pixels des centres des cercles
$V \leftarrow []$;

2 **Acquisition**
tant que $k < N$ **faire**
Récupérer une image de chaque caméra. Chercher les cercles dans les 2 images ;
si Tous les cercles de l'objet sont trouvés dans chaque image **alors**
Afficher les cadres de l'objet sur l'UI ;
Enregistrer les positions des centres des cercles dans V ;
$k \leftarrow k + 1$;
Attendre 2 secondes `// On laisse le temps au manipulateur de déplacer l'objet`
sinon
Aller à la prochaine paire d'images ;
fin
fin

3 **Stéréocalibration**
Calcul de les matrices de stéréocalibration et de l'erreur de rétroprojection globale E.

4 **Validation**
Pour tout v *dans* V **faire**
Calculer l'erreur de contrainte épipolaire ECE_v pour chaque point i.
si $max(ECE_v) \geq 3px$ **alors**
Retirer la vue des images utilisées pour la stéréocalibration ;
$k \leftarrow k - 1$;
sinon
Continuer
fin
fin

5 **Conclusion**
si $k < n$ **alors**
Aller à l'étape 2 `// Il faut reprendre des images.`
sinon
si Au moins une image a été retirée **alors**
Aller à l'étape 3 `// On recalcule les matrices de stéréocalibration`
sinon
Terminer
fin
fin

sont celles qui ont le plus de sens physique à nos yeux, nous rapportons à la **Figure 5.6**, les valeurs de translation obtenues selon chaque dimension . Ces valeurs sont synthétisées dans le **Tableau 5.3**.

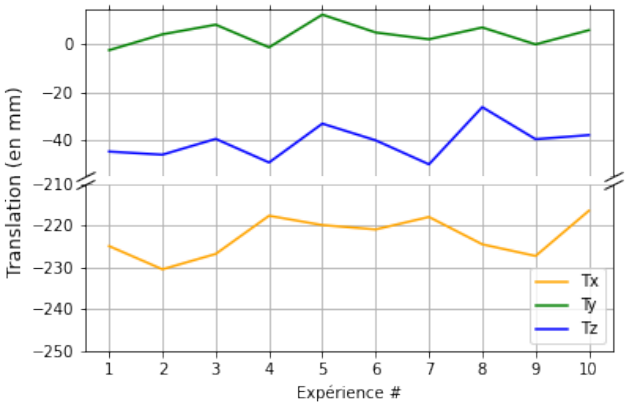


FIGURE 5.6 Valeurs de translation obtenues sur 10 expériences (en mm)

Translation	Valeur Moyenne	$\sigma\,(mm)$
T_x	-222.6	4.76
T_y	4.1	4.61
T_z	-40.8	7.35

TABLEAU 5.3 Moyenne et écart type des valeurs de translation obtenues sur une répétition de 10 expériences. $N = 30, n = 25$

Nous constatons bien une stabilité au travers de nos expériences, avec un écart type de l'ordre de quelques millimètres pour chacune des directions du vecteur de translation. Nous observons cependant une plus grande variabilité dans la direction Z, qui correspond à l'axe principal de la distance focale. Nous interprétons cette variabilité comme étant principalement liée aux erreurs lors de l'étape de calibration, et l'incertitude de la mesure de distance focale.

Visualisation

Dans cette section, nous proposons un outil visuel pour observer l'efficacité de notre stéréocalibration. Nous avons créé une interface graphique qui permet, en annotant un point dans l'image RGB de la Kinect, d'afficher le point associé sur l'image de la FLIR. Considérons un point $M(X, Y, Z)$ dans le repère monde, ce point est projeté dans l'image RGB au point $m(u_2, v_2)$. Alors, le point M, vu dans le repère associé à la Kinect, a pour coordonnées (X_2, Y_2, Z_2) qui satisfont :

$$\begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix} = Z(u_2, v_2)\mathbf{D}_2 \star^{-1} \mathbf{K}_2^{-1} \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} \tag{5.9}$$

Avec $Z(u_2, v_2)$ la profondeur du point annoté, obtenu grâce à l'image de profondeur de la Kinect. Les matrices $\mathbf{D}_2$ et $\mathbf{K}_2$ sont celles obtenues à la **Section 5.2.5**. Enfin, l'opérateur $\star$

correspond à la correction associée à la distorsion, comme montré dans les **Équations A.4** et **A.5**. Pour l'**Équation 5.9**, nous considérons l'opérateur $\star^{-1}$ qui n'est autre que l'opération inverse.

Une fois les coordonnées du point M exprimée dans le référentiel de la caméra 2, nous pouvons l'exprimer dans le référentiel de la caméra 1, puis reprojeter dans l'image associée.

$$\begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} = \mathbf{R}^{-1} \left(\begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \end{pmatrix} - \mathbf{T} \right) \tag{5.10}$$

$$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = \frac{1}{Z_1} \mathbf{K}_1 \mathbf{D}_1 \star \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \end{pmatrix} \tag{5.11}$$

Visuellement le résultat obtenu est illustré dans la **Figure 5.7**. Tout d'abord, on peut constater un succès relativement général de la pose. Il est intéressant de constater un léger décalage vers la droite, pour lequel nous n'avons pas d'interprétation.

(a) **(b)**

FIGURE 5.7 Visualisation des résultats de stéréocalibration. (a) Pose d'un sujet annotée manuellement sur l'image RGB de la Kinect. (b) Pose reconstruite en utilisant les différentes matrices de calibration et stéréocalibration.

5.4 Synchronisation inter-caméras

Afin de construire une base de données d'images annotées dans différentes modalités, il convient d'avoir des images synchronisées temporellement. Dans les approches de type Mo-Cap, ceci est le plus souvent effectué par des impulsions électriques depuis un système maître vers les différentes caméras du montage. Dans notre cas, nos caméras ont été développées de manière complètement indépendante, et la synchronisation devra donc être assurée d'une manière purement logicielle.

Des fréquences d'acquisition différentes

Nos caméras ne font pas l'acquisition d'images à la même fréquence. D'une part, les fournisseurs de la Kinect indiquent une fréquence de 30 IPS, ce que nous avons pu constater expérimentalement. D'autre part, si cette information n'est pas fournie par la FLIR, nous avons expérimentalement trouvé une vitesse plus proche de 29 IPS.

Il est surtout important de préciser qu'une fréquence d'acquisition en IPS est une moyenne réalisée sur plusieurs secondes, voire minutes d'acquisition. Par exemple, si l'on considère une caméra fonctionnant à 20 IPS, il est absurde de penser que nous recevrons une image toutes les $1000/20 = 50$ millisecondes.

5.4.1 Limitations d'une approche séquentielle

Que ce soit pour le processus de stéréocalibration ou pour l'acquisition générale de notre base de données, il est nécessaire d'avoir des images le plus synchrones possibles. En effet, comme nous souhaitons combiner les informations thermiques et de profondeur pour faire notre EPH, il est nécessaire que ces informations soient capturées au même moment. Dans le cas de la stéréocalibration, nous ne pouvons pas nous permettre de comparer des vues de l'objet qui ont été prises à deux moments différents.

Pour assurer ce synchronisme, l'approche la plus traditionnelle consiste à envoyer des impulsions électriques depuis une source extérieure vers les deux caméras afin de leur signaler quand réaliser une capture [57]. Cependant, ceci nécessite d'avoir des ports adaptés sur les caméras (typiquement reliés via des câbles coaxiaux) que nos deux caméras ne possèdent pas.

Une approche standard pour contourner cette limitation est d'instaurer du parallélisme dans la communication entre l'ordinateur et les caméras. En effet, un ordinateur procède inhéremment de manière séquentielle. Dans l'approche la plus simpliste, montrée en **Figure 5.8**, l'algorithme récupère d'abord l'image depuis la caméra 1, puis l'image de la caméra 2, avant

de traiter ces données (l'écriture des images au disque dans le cas d'une acquisition, et la détection des cercles de l'objet dans le cas de la stéréocalibration).

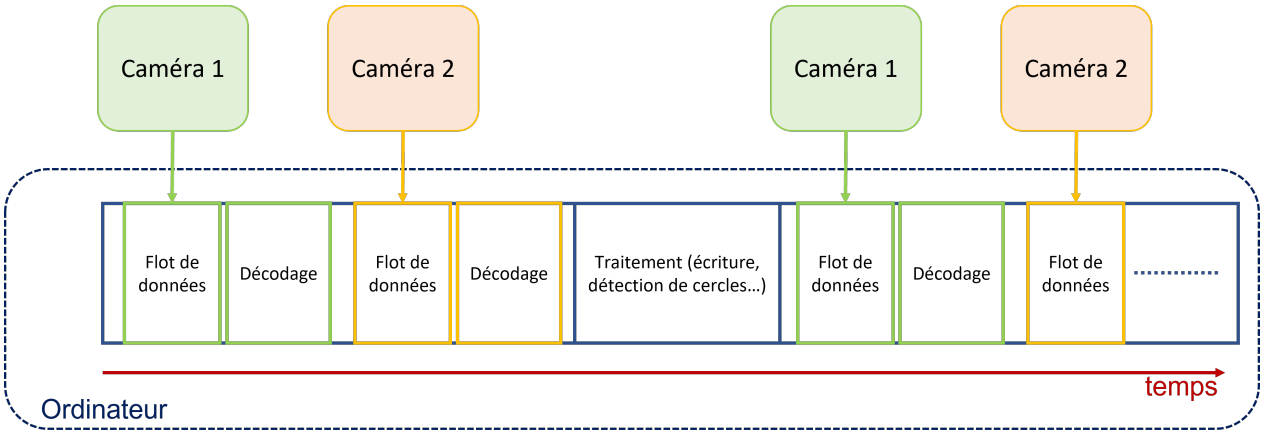


FIGURE 5.8 Communication ordinateur-caméras vue d'une manière séquentielle

Clairement, cette approche ne répond pas à nos contraintes de synchronisme. Premièrement, les images récupérées par les 2 caméras ne sont pas acquises simultanément. Deuxièmement, le traitement des données entre les acquisitions vient diminuer la fréquence d'acquisition de notre système global. Si nos deux caméras peuvent fonctionner à environ 30 IPS indépendemment, un tel système ne fonctionnerait qu'à quelques IPS en pratique.

5.4.2 Parallélisation par multithreading

La **Figure 5.8** nous montre que notre cas appartient à la catégorie des problèmes dits *subordonnés en temps de calcul* (*CPU-bound* en anglais). Brièvement, ceci indique que la performance de notre programme est principalement déterminée par la performance du processeur de notre ordinateur (le CPU). L'approche la plus classique pour contourner les limitations physiques de notre CPU est de paralléliser les tâches de notre algorithme via du *multithreading*. Ceci fait particulièrement sens dans le cas de la stéréocalibration : en plus de capturer les images et de les analyser, on veut aussi proposer une interface graphique à l'utilisateur pour faciliter le processus. Un exemple d'une architecture multithreading est donnée en **Figure 5.9**. Pour alléger la lecture, nous regroupons les étapes de réception du flot de données et de décodage en une seule étape nommée *capture*.

Dans la **Figure 5.9**, on considère le cas où le Thread 1 communique avec la caméra 1, le thread 2 avec la caméra 2. Le thread 3 est un thread général pour une utilisation plus globale du CPU (par exemple la détection de cercles). Un des atouts de l'approche multithread est que les processus d'écriture au disque peuvent se faire en arrière plan. De plus, l'espace

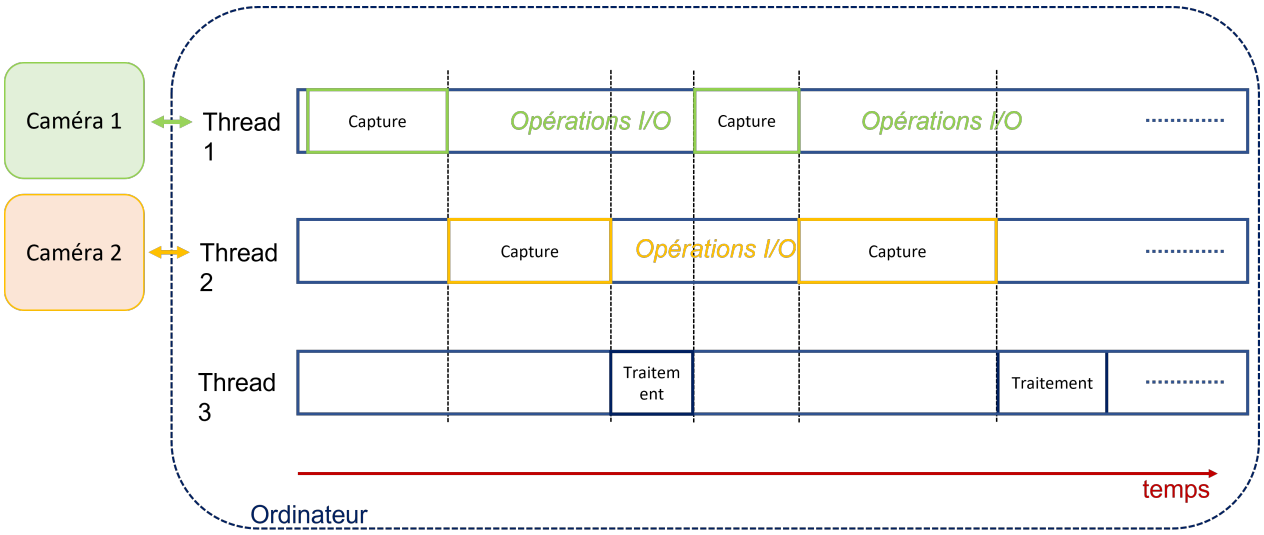


FIGURE 5.9 Communication ordinateur-caméras en utilisant du multithreading

mémoire étant distribué entre les threads, le thread 3 peut facilement récupérer et traiter les images des deux caméras. Malheureusement, si cette approche paraît être parallélisée, elle est exécutée de manière séquentielle.

En effet, une des grandes limitations de l'approche multithread vient de l'implémentation même de la librairie dans python. Afin d'éviter que plusieurs threads puissent modifier de manière simultanée une variable, Python possède un GIL (*Global Interpreter Lock*). Ce *lock* (un verrou en français) empêche plusieurs thread d'utiliser le CPU de manière concurrente, et se pose donc en véritable frein au parallélisme. Dans la **Figure 5.9** chaque trait vertical en pointillé noir correspond à un changement de propriétaire du GIL. Quand un thread commence à monopoliser les ressources du CPU, il fait l'acquisition du GIL et le relâche quand il a terminé sa tâche.

5.4.3 Contourner le GIL : le multiprocessing

Pour contourner les limitations du GIL, nous utilisons la libraire de *multiprocessing* de python. Avec une syntaxe similaire à celle du multithreading, cette librairie permet de paralléliser des calculs en générant des processus indépendants. Chaque processus possède son propre espace mémoire, son propre interpréteur et, à fortiori son propre GIL. Si nous gagnons en parallélisme, nous perdons en partage de mémoire.

Architecture d'acquisition

Pour faire l'acquisition de nos images, nous avons une contrainte supplémentaire. Il faut, en parallèle de l'acquisition d'images en simultané, les écrire au disque. En effet, garder toutes les images en mémoire vive et les écrire au disque une fois l'acquisition terminée serait impossible en raison de la taille limitée de la RAM. Notre écriture doit donc être rapide, réalisable en parallèle des captures, et sans perte d'informations (nous ne pouvons pas nous permettre d'enregistrer les images au format JPG par exemple). De plus, l'information de profondeur en sortie de la Kinect est encodée sur 16 bits : un enregistrement au format PNG par exemple ne fonctionnerait pas non plus. L'analyse comparative réalisée en [58] montre que le moyen le plus efficace est d'enregistrer les images comme des matrices numpy, avec un temps moyen d'écriture au disque de 25 millisecondes.

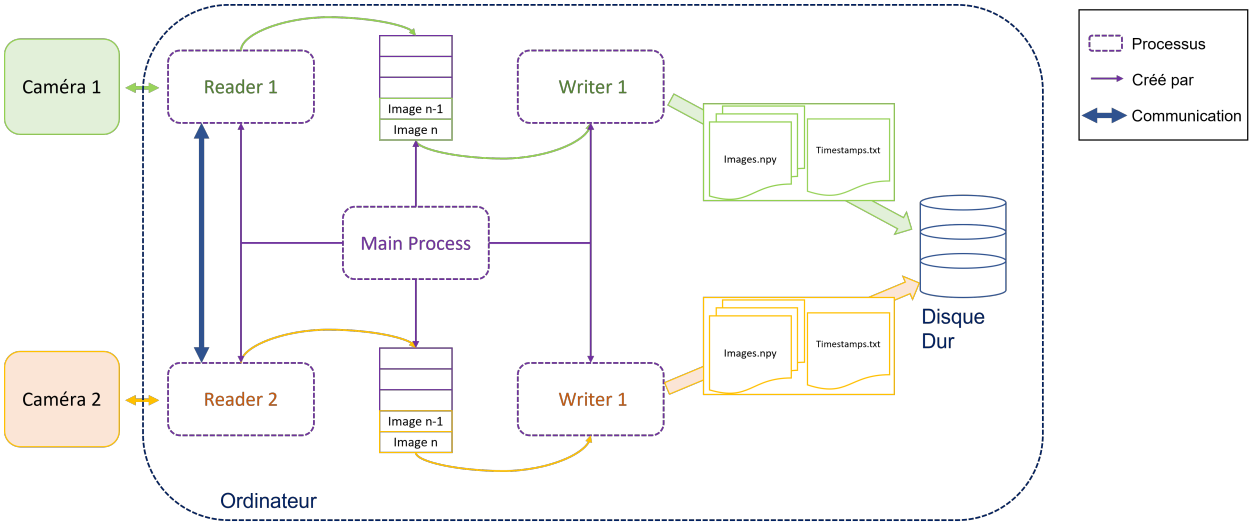


FIGURE 5.10 Architecture en multiprocessing pour la phase d'acquisition

Afin de ne pas générer de concurrence entre la lecture et l'écriture d'une image, nous créons deux processus indépendants pour chaque caméra : un processus de lecture (le *Reader*), qui communique avec la caméra, et un processus d'écriture (le *Writer*), qui communique avec le disque. Le *Reader* capture les images et les stocke dans une *Queue*, qui est une structure de données de type FIFO (*First In First Out*). Afin de s'assurer de ne pas "perdre" des images, nous introduisons aussi un compteur, qui suit le remplissage de la Queue : chaque fois que le Reader ajoute une image à la Queue, il incrémente le compteur. Chaque fois que le Writer écrit une image au disque, il le décrémente. Ce compteur permet aussi de s'assurer que les queues sont vides au moment de tuer les processus, afin d'éviter des problèmes de *deadlock*.

L'avantage principal de la Queue est qu'elle permet de garder les images en mémoire vive le

temps que le Writer les écrive au disque, sans pour autant saturer cet espace. En pratique, on observe expérimentalement que la Queue ne dépasse jamais une taille de 2.

Dans le cas de la Kinect, nous souhaitons sauvegarder à la fois l'image de profondeur et l'image RGB. Nous créons donc une queue et un Writer pour chaque modalité.

Architecture de stéréocalibration

La stéréocalibration demande une architecture complètement différente de l'acquisition, car nous devons traiter les images en temps réel, tout en présentant une UI dynamique et intuitive. Pour cela, nous nous inspirons de la **Figure 5.10**, en conservant les processus de Reading, et créons le protocole présenté en **Figure 5.11**. Le processus principal est pour sa part décomposé en 3 threads. Le thread 1 demande en continu des images aux queues. Lorsqu'il récupère une paire d'images, ces dernières sont dupliquées et transférées d'une part au thread principal pour être affichées sur l'UI, d'autre part au thread 2 qui y cherche l'objet de calibration. Si le thread 2 trouve l'objet dans les 2 images, il envoie les positions au thread principal qui les affiche sur l'UI.

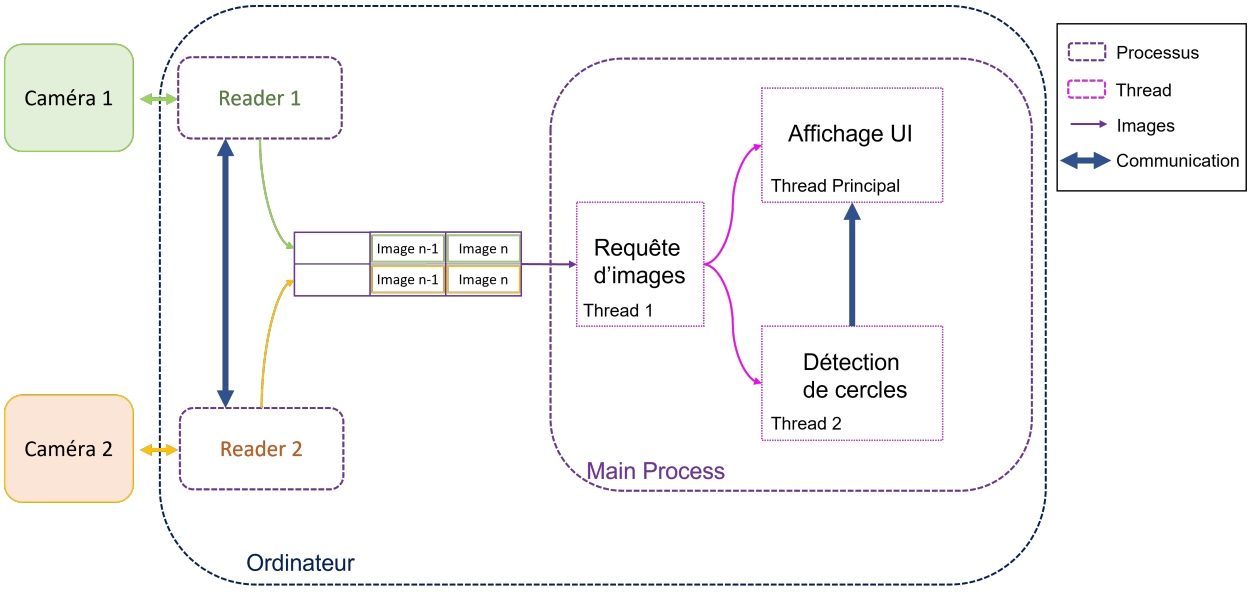


FIGURE 5.11 Architecture en multiprocessing pour la phase de stéréocalibration

Une fois que l'algorithme de stéréocalibration a trouvé suffisamment d'images (conformément à l'**Algorithme 2**), les différents processus et threads sont stoppés afin de laisser la puissance de calcul pour la détermination des matrices du système.

5.4.4 Protocole de communication inter-caméra

Nos processus de lecture aux caméras pouvant dorénavant fonctionner indépendamment, nous devons désormais nous assurer que les images qui sont envoyées aux queues respectives soient synchronisées. Si nous prenions simplement les images de chaque caméra les unes à la suite des autres, la différence de vitesse d'acquisition impliquerait un décalage croissant entre ces images.

En pratique nous nous servons du fait que la Kinect a une fréquence IPS supérieure à la caméra thermique. L'idée est la suivante : chaque fois que nous capturons une nouvelle image avec la FLIR, nous considérons l'image de la Kinect la plus récemment acquise, qui est constamment stockée grâce à un sous-thread, comme montré en **Figure 5.12**.

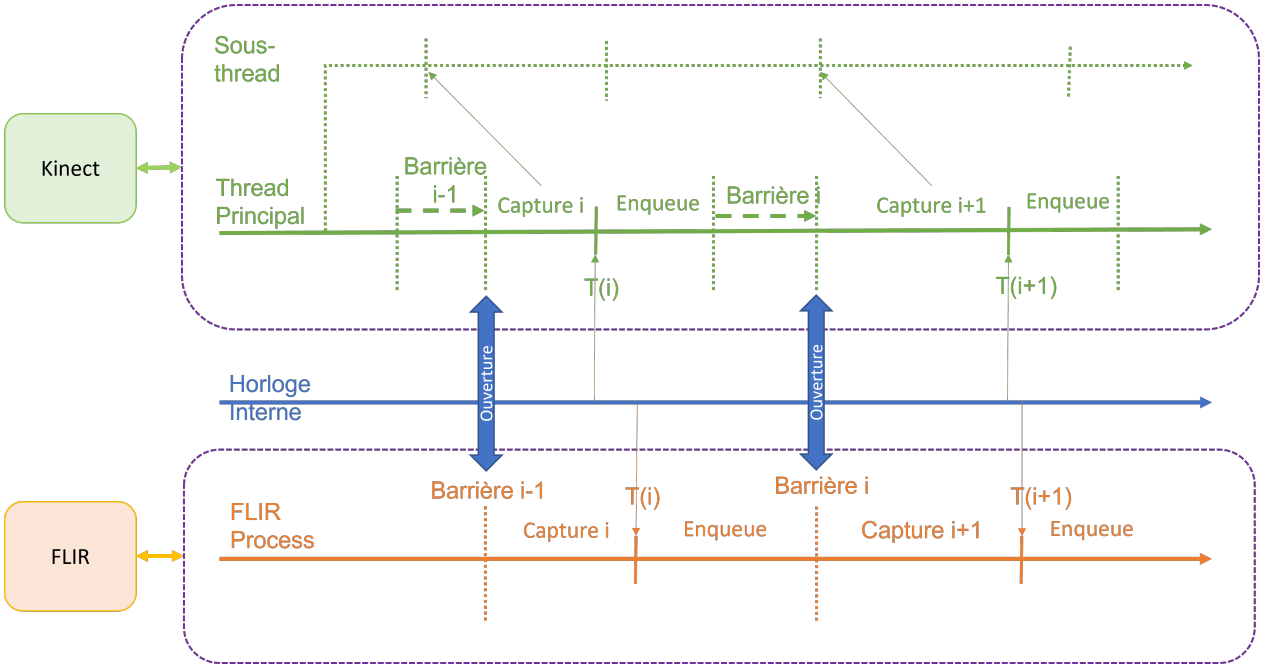


FIGURE 5.12 Méthode de synchronisation entre les deux caméras grâce à l'implémentation de barrières.

Pour rappel, un bloc de 3 opérations doit être réalisé lors de la capture d'une image : récupérer le flot de données, le décoder pour le structurer en une matrice (2D pour une image de profondeur ou thermique, 3D pour une image couleur), puis l'envoyer à la queue afin qu'elle soit traitée. Dans la **Figure 5.12**, les deux premières étapes sont regroupées dans la partie *Capture* et la dernière étape est nommée *Enqueue*. Ce bloc d'acquisition n'est pas de durée constante, bien au contraire (que ce soit pour un même bloc entre deux caméras, ou entre deux blocs d'une même caméra). Pour maintenir notre synchronisation, nous intégrons un moyen de communication entre les processus via une *barrière*. Lorsqu'un des deux processus

termine de son bloc, il atteint une barrière, et doit attendre que l'autre processus atteigne la sienne avant de pouvoir procéder au bloc suivant. Les deux barrières sont alors ouvertes. En pratique, le processus associé à la Kinect arrive toujours en premier à la barrière. En effet, la capture étant performée par le sous-thread en arrière-plan en continu : à chaque nouveau bloc, l'étape de capture de la Kinect ne consiste qu'à un appel au sous-thread pour la dernière image en mémoire.

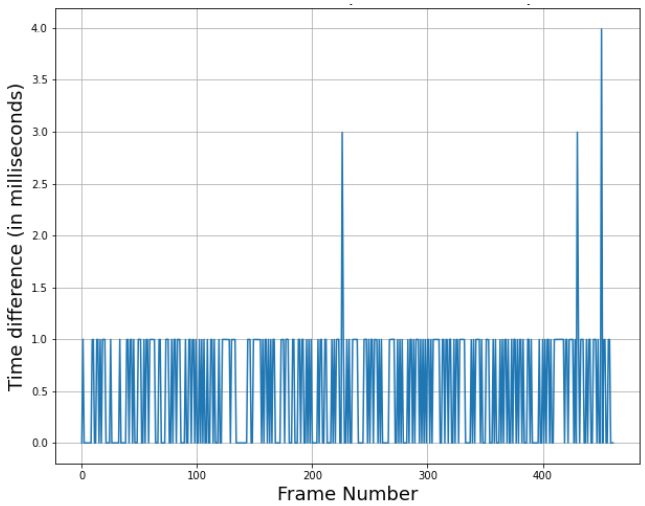


FIGURE 5.13 Écart entre deux timestamps d'une paire d'images obtenues par le protocole décrit en Figure 5.12

Après chaque capture, un appel est fait à l'horloge interne de l'OS de l'ordinateur pour obtenir le *timestamp* de l'image. Cet appel possède une résolution de l'ordre de la microseconde, ce qui est suffisamment précis dans notre cas : 30 IPS correspondent à un temps moyen de 30 millisecondes entre deux captures. En pratique, nous observons que l'écart entre deux timestamps d'une même paire ne dépassent rarement la milliseconde, comme montré en **Figure 5.13** : nous y comparons les paires de timestamps pour une acquisition de 450 images (environ 15 secondes). On observe un écart moyen de 0.786 ms. Cette performance est maintenue pour des acquisitions plus longues (de l'ordre de la minute).

5.5 Prise en compte des interruptions hardware

Comme dans tout travail faisant intervenir du matériel hardware, ce dernier peut parfois cesser de fonctionner de manière ponctuelle ou permanente : problème interne, faux contact au niveau du branchement USB, panne de batterie, etc. Actuellement, notre protocole traite ces cas d'une unique manière : si un des composants hardware (une des deux caméras ou la station IMU, cf. la section suivante) est défaillant, l'ensemble du protocole d'acquisition est stoppé. Ceci est assuré par un processus indépendant nommé *listener*. Ce dernier est divisé en plusieurs threads, un pour chaque processus créé. Si un des processus fait remonter un message d'erreur au listener, il retransmet ce message à tous les autres processus, leur ordonnant d'interrompre leur travail, mettant ainsi un terme à l'acquisition.

Si cette approche est rigoureuse, elle n'est pas pour autant optimale. En effet, dans un contexte de travail en temps réel pour le futur de PICUPE, nous voulons que notre système d'acquisition puisse fonctionner en permanence. De fait, si une des caméras venait à cesser de fonctionner pour une raison quelconque, l'autre caméra devrait continuer à tourner. Les processus devraient même pouvoir être capable de relancer la caméra et se resynchroniser au système. Ceci n'est pas implémenté aujourd'hui et constitue un axe d'amélioration de notre système d'acquisition.

5.6 Conclusion sur notre système d'acquisition vidéo

Nous avons élaboré un système d'acquisition vidéo qui possède deux principales caractéristiques. Premièrement, notre système est un système stéréocalibré. En pratique, ceci implique que les matrices de transformation pour passer du repère d'une caméra à celui d'une autre sont connues. De fait, pour tout point dans une modalité, nous pouvons trouver son point associé dans une autre modalité. Deuxièmement, notre système d'acquisition vidéo permet l'acquisition d'images synchronisées temporellement dans 3 modalités - RGB, profondeur, thermique - avec une fréquence de 29 IPS.

Nous avons ainsi réussi la première étape de la création de notre base de données. Afin d'entraîner notre réseau de neurones dessus, il s'agit désormais d'annoter les images.

CHAPITRE 6 MISE EN PLACE D'UN SYSTÈME AUTOMATIQUE DE GÉNÉRATION D'ANNOTATIONS DE POSE

6.1 Méthodologie

Nous avons mis en place un système d'acquisition vidéo performant, il s'agit désormais d'annoter les poses dans toutes les images et toutes les modalités obtenues. Cependant, nous souhaitons créer une base de données de grande envergure, et tout annoter à la main, comme c'est le cas dans SLP, n'est pas pertinent pour plusieurs raisons déjà citées (cf. **Section 2.3.1**).

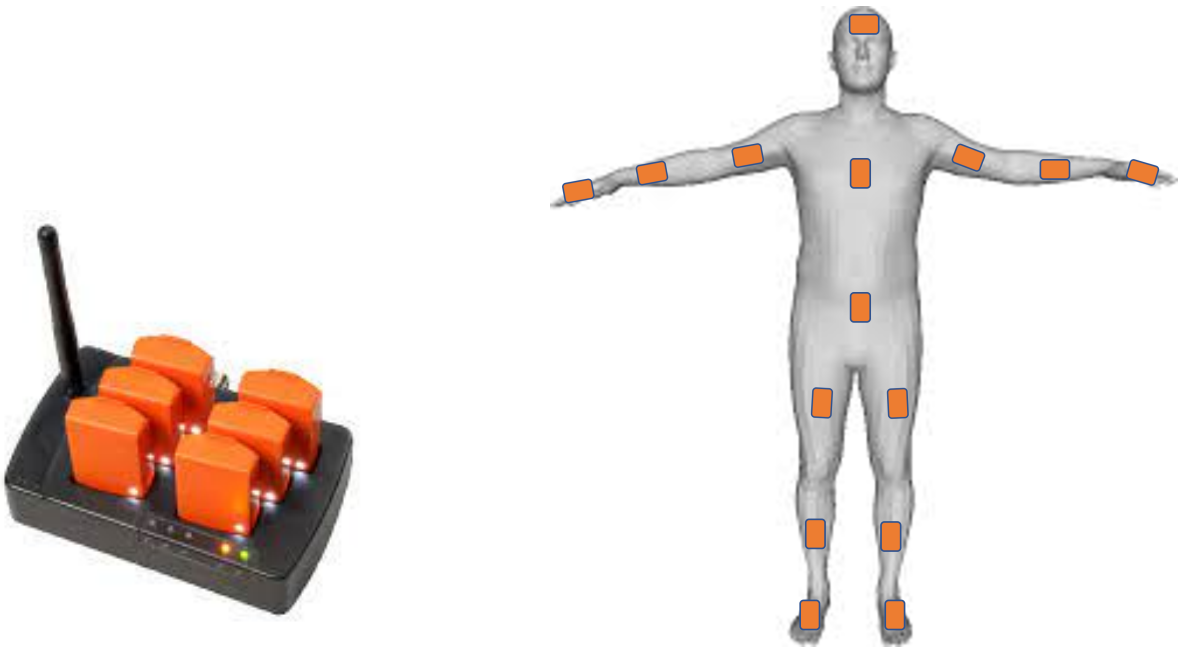
Comme nous l'avons présenté en **Section 2.2.2**, les IMU sont des capteurs inertiels qui peuvent être utilisés dans un cadre d'EPH. Pour ce projet, nous pensons que les IMU peuvent être une bonne méthode pour la génération de notre vérité terrain qui sera utilisée pour entraîner nos réseaux de neurones, d'autant plus que l'avantage majeur de ces capteurs est l'insensibilité aux occlusions visuelles et aux auto-occlusions. Nous sommes conscients que les IMU sont intrusifs, long à mettre en place et nécessitent une phase de calibrage. C'est pour cela que nous souhaitons uniquement les utiliser afin d'annoter notre base de données. Une fois intégré au CHUSJ, le réseau de neurones estimera la pose de patients uniquement en se basant sur les images multimodales. Dans cette section, nous présentons des résultats obtenus à travers des essais dans le laboratoire de Polytechnique Montréal sur des adultes.

6.1.1 Matériel utilisé

Nous utilisons un jeu de 15 IMU de la société XSens, et montrés en **Figure 6.1**. Ces boîtiers d'environ 5x3 cm pour un peu plus d'1cm d'épaisseur ne pèsent que 16 grammes et peuvent transmettre des données à une fréquence de 60Hz. Non-filaires, ils communiquent indépendamment par micro-ondes avec une station maîtresse, elle-même reliée à l'ordinateur par voie filaire.

6.1.2 Principe d'intégration de données

Pour estimer la pose humaine, l'information issue des IMU doit nécessairement être couplée à un modèle du corps humain. Nous utiliserons la libraire open-source OpenSim, qui a créé le package OpenSense, entièrement dédié à l'intégration de données IMU. Développé en C++, il possède un wrapper Python avec lequel nous interfaçons.



(a) **(b)**

FIGURE 6.1 (a) Lot de 6 IMU branchés sur la station maîtresse. (b) Emplacement des IMU suggéré par le fabricant dans le cas où les 15 sont utilisés.

Dans toute la suite de cette section, nous ferons la distinction entre un IMU *physique* et un IMU *virtuel*. Le premier terme définit le senseur réel placé sur un membre du sujet, tandis que le deuxième est sa modélisation sur le squelette OpenSim.

L'intégration se fait typiquement en 3 étapes.

Première étape : Collecte de données

1. Les senseurs IMU physiques sont placés aux divers endroits du corps du sujet. Chaque IMU a une position attitrée : fémur gauche, main droite, sternum, etc. indiquée sur le boîtier. Ils sont fixés sur les membres par un strap velcro, et le plus possible parallèle à l'axe de la colonne vertébrale. Pour fixer l'IMU sur le sternum, le constructeur nous fournit une veste. Au total, nous pouvons travailler avec jusque 15 IMU, disposés comme en **Figure 6.1b**, mais il est possible d'en utiliser moins.
2. Le patient prend une pose de référence durant quelques secondes, souvent nommée N-Pose (debout, les pieds joints et les bras le long du corps) ou T-Pose (idem, mais les bras levés perpendiculairement), comme montré en **Figure 6.1b**. Dans nos expériences, nous choisirons la N-Pose, mais n'importe quelle pose connue fonctionnerait. Cette étape est cruciale car elle permettra de calibrer notre modèle musculosquelettique.

3. L'acquisition peut alors commencer et le sujet peut bouger librement.
4. Une fois l'acquisition terminée, les données doivent être traitées, nettoyées et transformées au bon format pour être intégrées par OpenSense. Nous verrons cette étape plus en détail en **Section 6.2**.

Deuxième étape : calibrage et cinématique inverse

Dans cette section, nous utilisons exclusivement les fonctions et modèles développées par les auteurs d'OpenSense.

Modèle choisi Pour nos premières expériences, nous choisissons un modèle squelettique simple nommé Rajagopal [39]. Ce modèle décrit la morphologie d'un adulte, sans modélisation des muscles. Présentant un total de 37 degrés de liberté, couvrant l'ensemble des articulations du corps, nous choisissons de réduire certains degrés de liberté dans notre travail, notamment au niveau des pieds et des mains. Ceci permet de réduire l'espace de recherche lors du problème de cinématique inverse présenté plus bas. En gardant un seul degré de liberté pour les mains et pieds, nous obtenons 29 degrés de liberté.

IMU Placer L'étape de calibrage consiste à placer les IMU virtuels sur notre modèle musculosquelettique, et de leur donner la même orientation que les IMU physiques lors de la N-Pose. Ceci est réalisé grâce à une fonction OpenSense dédiée (nommée *IMUPlacer*). Pour cela, nous prenons comme IMU de référence celui du bassin et définissons deux systèmes de coordonnées 3D. Un premier repère est associé à l'IMU physique, un deuxième à l'IMU virtuel associé. La fonction IMUPlacer commence par calculer la rotation qui permet d'aligner l'axe des abscisses de ces deux repères. En appliquant cette transformation aux valeurs d'orientation de l'IMU de base, ceci permet d'*aligner* l'IMU physique et l'IMU virtuel. La même rotation est alors appliquée aux orientations de tous les IMU physiques. On considère alors que le modèle est calibré.

Cinématique Inverse La procédure de cinématique inverse prend en entrée un modèle calibré ainsi que les orientations des IMU au cours du temps. À chaque pas de temps, l'algorithme calcule une nouvelle position des membres du modèle musculosquelettique, et la nouvelle orientation des IMU virtuels. La différence d'angle entre l'orientation virtuelle obtenue et l'orientation physiquement mesurée est notée θ_i. La nouvelle position du modèle est

alors optimisée par Levenberg-Marquardt en minimisant la fonction d'erreur :

$$\min_{q} \sum_{i \in IMU} w_i \theta_i^2 \tag{6.1}$$

Ici, q correspond au vecteur des valeurs associées aux 29 degrés de liberté de notre modèle. On comprend donc ici le choix de réduire ce nombre le plus possible, afin de diminuer la complexité du problème d'optimisation. Pour chaque IMU i, nous associons un poids w_i, dont les valeurs sont données dans le tableau ci-dessous.

IMU	Poids relatif
Pelvis	1
Sternum	1
Tête	1
Humérus	1
Cubitus	0.5
Main	0.1
Fémur	1
Tibia	0.5
Pied	0.1

Tableau 6.1 Liste des poids associés à chaque IMU lors de l'optimisation de Levenberg-Marquardt dans le problème de cinématique inverse.

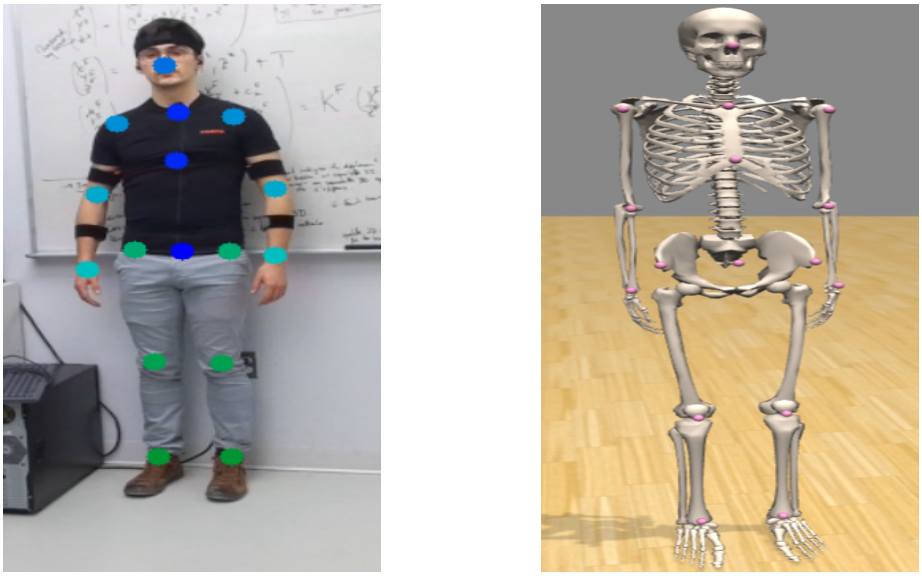
Ces valeurs ont été choisies en s'inspirant des travaux initiaux de OpenSense [59], en suivant l'idée que les IMU les plus proches du centre de gravité du corps sont plus importants dans la reconstruction de la pose globale du sujet. Expérimentalement, nous avons constaté que ces coefficients doivent rester relativement proches. En effet, si le rapport entre le plus gros et le plus léger des poids est supérieur à un facteur 10, nous observons un phénomène de "vibration" des membres à poids faible.

Troisième étape : Extraction de pose

En sortie de l'étape de cinématique inverse, nous obtenons donc une liste de vecteurs q (comme défini en **Équation 6.1**) pour chaque pas de temps. Cependant, cette information n'est pas une pose à proprement parler. Nous allons donc essayer d'extraire un modèle cinématique de la liste de vecteurs q.

Nous plaçons pour cela des marqueurs virtuels sur le modèle Rajagopal, qui correspondent au points clés du modèle cinématique que l'on vise à reconstruire. Sur la **Figure 6.2**, on peut voir d'une part les points clés sur le sujet, et d'autre part les marqueurs virtuels associés

en rose sur le modèle musculosquelettique. Nous utilisons les résultats de la cinématique inverse pour déplacer le modèle à chaque pas de temps, et en déduire la nouvelle position des marqueurs. Nous avons choisi de travailler avec 16 marqueurs, mais ce nombre peut sans difficulté être modifié.



(a) **(b)**

FIGURE 6.2 Sélection des points clés dont la position sera générée pour créer un modèle cinématique. (a) Sur le sujet (b) Marqueurs virtuels associés en rose sur le modèle OpenSim.

6.2 Acquisition des données IMU

6.2.1 Intégration dans le pipeline global

L'intégralité du protocole d'acquisition IMU a été réalisé grâce au SDK Python développé par XSens. Une fois la station Awinda branchée à l'ordinateur, nous établissons un tunnel de communication avec cette dernière : elle devient notre intermédiaire avec les IMU. Comme nous avons vu en **Section 5.4**, nous avons recours au multiprocessing pour faire l'acquisition en simultané de nos deux caméras. De fait, nous pouvons facilement rajouter un processus indépendant uniquement dédié à l'acquisition des données IMU.

6.2.2 Quelles données récupérer ?

La communication entre IMU et la station est assurée par un transfert de *paquets*. Un paquet contient toute l'information que nos demandons à l'IMU d'enregistrer. Spécifiquement, nous

recueillons 4 type de données :

— L'**accélération** de l'IMU en X, Y, Z.

— L'**orientation** de l'IMU sous forme d'une matrice de rotation.

— L'instant auquel le paquet a été mesuré. Par analogie à l'acquisition vidéo, on appellera ces pas de temps les **timestamps**. Lorsque la station Awinda est connectée à l'ordinateur, elle se synchronise à son horloge interne, synchronisation qui se propage aux IMU. Les timestamps IMU sont donc mesurés avec la même précision et la même horloge que les timestamps vidéo.

— L'**identifiant** du paquet qui est un entier entre 1 et 65535. Le premier paquet d'une acquisition a un identifiant quelconque. Puis, l'identifiant de chaque nouveau paquet est incrémenté de 1 par rapport au paquet précédent.

6.2.3 Écriture au disque

Une première écriture binaire Dans le même souci d'éviter de surcharger la mémoire vive que lors de l'acquisition vidéo, nous écrivons les paquets reçus par les IMU dès l'instant que ceux-ci sont reçus par la station. Afin d'optimiser ce temps d'écriture, nous les enregistrons au format binaire, avec un fichier pour chaque IMU.

Une fois l'acquisition terminée, nous convertissons les fichiers binaires en fichiers texte. Cette conversion est nécessaire et respecte un gabarit strict requis par les fonctions OpenSim afin que les données soient intégrées correctement.

Interpolation de paquets manquants Lors de cette conversion, nous analysons les identifiants des paquets écrits. Il arrive que certains paquets soient *manquants* car ils n'ont pas été captés par la station Awinda. Au cours de nos acquisitions, nous avons constaté que ce problème concernait une faible partie des paquets (inférieur à 1% pour une acquisition d'une minute, comme montré en Annexe A).

Nous réalisons une interpolation simple pour générer artificiellement les paquets manquants. En pratique, les valeurs d'un paquet manquant i sont interpolées comme la moyenne entre les informations des paquets $i - 1$ et $i + 1$. En considérant le faible nombre de paquets manquants, et la haute fréquence d'acquisition des IMU (60 Hz), nous considérons cette méthode numériquement satisfaisante. Dans le cas où le paquet initial ou final est manquant, nous dupliquons simplement le paquet le plus proche.

Ce traitement permet d'avoir en fin d'acquisition autant de fichiers texte que d'IMU, tous ayant exactement le même nombre de paquets.

6.3 Résultats et discussion

6.3.1 Synchronisation OpenSim-Vidéo

Lors de l'acquisition de nos données, nous avons constamment récupéré en simultané le timestamp associé. Il est important de noter que ce timestamp est toujours associé à l'horloge interne de l'OS, donc la comparaison entre eux est cohérente. Pour chaque paire d'images FLIR/Kinect ayant un timestamp i nous associons la configuration OpenSim au timestamp j le plus proche.

6.3.2 Visualisation d'une expérience

Afin de visualiser nos résultats, nous considérons une expérience réalisée dans un des laboratoires de Polytechnique. Les différentes étapes de cette expérience sont comme suit :

1. De 0 à 7 secondes : N-Pose.
2. De 7 à 18 secondes : Marche aller-retour sur une ligne droite.
3. De 18 à 25 secondes : T-Pose
4. De 25 à 37 secondes : Marche aller-retour sur une ligne droite.
5. De 37 à 46 secondes : Réalisation de deux squats.

Nous précisons que la calibrage du modèle OpenSim est réalisé en utilisant uniquement la N-Pose du début. Nous réalisons une T-Pose en étape 3 uniquement dans le but de voir si le modèle peut reprendre une pose simple après une étape de marche.

La visualisation des résultats est assurée par un script simple qui affiche les 3 modalités ainsi que le modèle cinématique obtenu, le tout de manière synchrone. Un exemple est montré en **Figure 6.3**, où le sujet performe un squat. On y retrouve bien la flexion des genoux, ainsi que les bras tendus vers l'avant.

Si notre architecture parvient à suivre des mouvements relativement complexes comme le squat, nous observons certains problèmes liés à l'orientation des IMU, comme montré en **Figure 6.4**. Alors que le sujet est en T-Pose, les bras sont reconstruits comme pointant vers l'avant. Ceci est particulièrement étonnant puisque la reconstruction est cohérente sur le reste de la séquence. Nous donnons une piste de réflexion dans la prochaine section.

6.3.3 Discussion

Notre protocole de génération de pose, quoique prometteur, ne semble pour autant pas pertinent pour une application au CHUSJ pour 3 raisons principales.

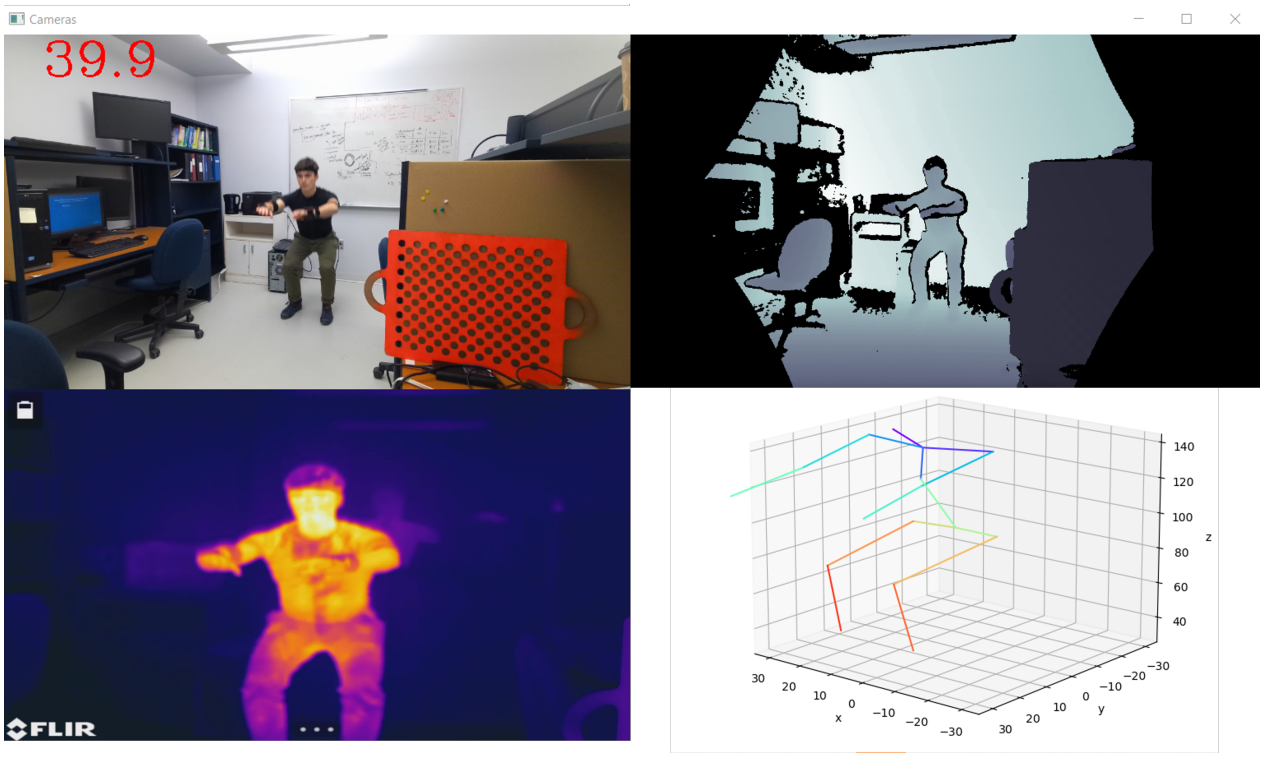


FIGURE 6.3 Reconstruction de pose lors d'un squat.

Une étape de calibrage insuffisante

Comme nous l'avons vu, le sujet est obligé de prendre une pose de calibrage afin que ce dernier et le modèle OpenSim soient dans la même configuration au début de l'acquisition. En pratique, nous avons choisi la N-Pose, car la plus intuitive et la plus simple, mais n'importe quelle pose aurait suffi, à condition de modifier le modèle OpenSim en conséquence.

Cependant, la très récente publication de (Di Raimondo et al.) montre que la N-Pose n'est pas du tout assez efficace pour calibrer un modèle OpenSim [**?**]. Ils comparent cette méthode de calibrage *statique* à une calibrage *dynamique* où le sujet effectue des mouvements spécifiques (ex : un mouvement de flexion de la jambe droite). Ils argumentent que la calibrage statique permet uniquement une reconstruction précise dans le plan sagittal, mais pas dans les autres plans (i.e. frontal et transverse). Nous constaterons d'ailleurs que ceci est cohérent avec l'observation des **Figure 6.4 et 6.3** : le squat reste dans le plan sagittal et est correctement suivi. La T-Pose, dans le plan frontal, est pour sa part totalement erronée.

Dans ce but, ils développent une méthode de calibrage combinant les approches statique et dynamique, et comparent leurs résultats à une vérité terrain obtenue par système MoCap. Leurs résultats montrent que cette nouvelle méthode de calibrage donne la reconstruction la

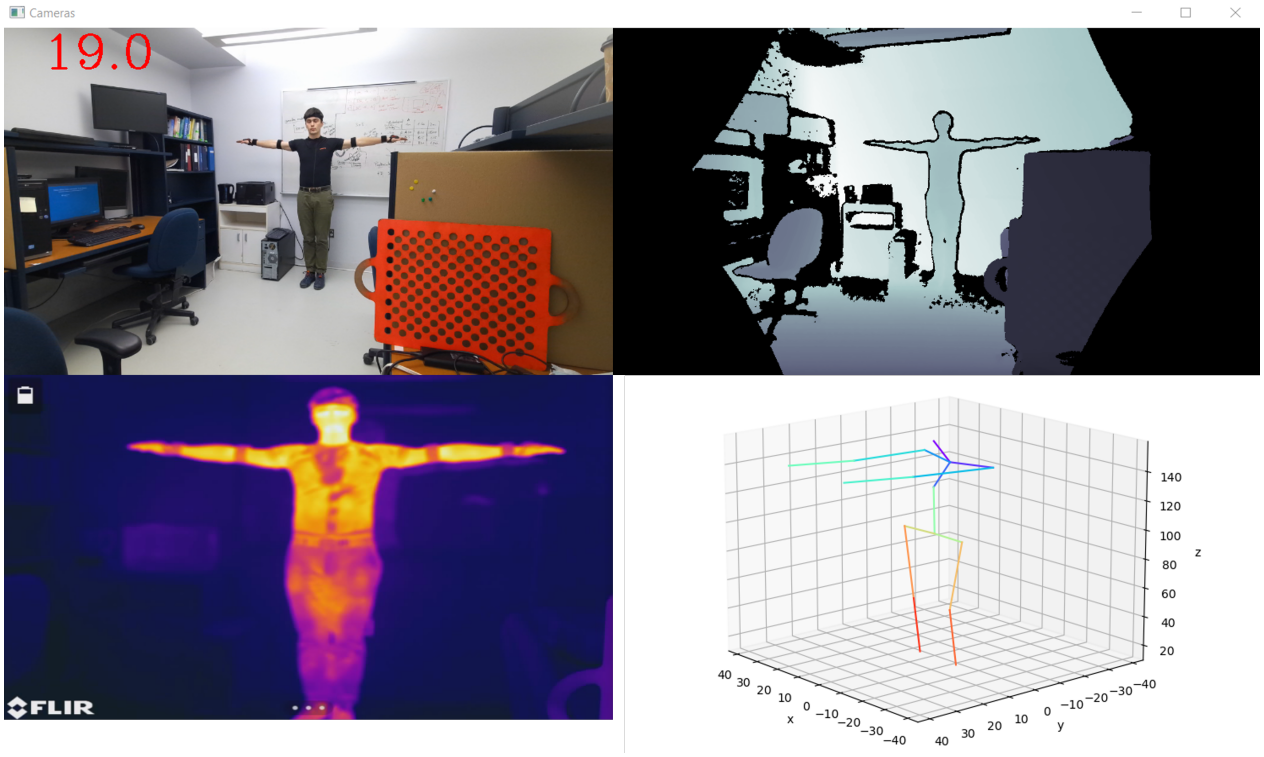


FIGURE 6.4 Erreur de reconstruction des bras : le sujet les lève sur le côté, tandis que le modèle les reconstruit en face.

plus précis dans le plan sagittal, tandis que la reconstruction basée sur une N-Pose donne les moins bons résultats.

De cette étude, nous voyons la première limitation des IMU pour la génération de pose : s'il était envisageable, (même si nous nous attendions à des difficultés), de demander aux patients du CHUSJ d'adopter une pose statique quelques instants pour calibrer notre modèle, leur demander de faire des mouvements spécifiques est difficilement concevable. De plus, nos observations sur le patient 0 (cf. le **Chapitre 7**) ont montré à quel point il était difficile de demander à un enfant de rester immobile, ne serait-ce que quelques secondes.

Une reconstruction centrée sur le bassin

La deuxième limitation de notre reconstruction réside dans la reconstruction de la position absolue du corps. En effet, lors de l'étape de cinématique inverse, le modèle est toujours considéré comme étant centré sur le bassin, le centre de ce dernier étant fixé dans l'espace 3D. Il n'y a donc aucun suivi du mouvement absolu du sujet dans l'espace.

Deux approches intuitives apparaissent pour suivre la position du bassin dans le temps :

— Intégrer l'information d'accélération de l'IMU du bassin 2 fois pour remonter à la position. La position 3D du bassin dans la première image de l'acquisition (obtenue grâce à l'outil présenté en **Section 5.3.4**) servirait alors de solution initiale. Cependant, cette approche est globalement reconnue comme étant numériquement instable [60].

— Utiliser une pastille de couleur placée sur le bassin et la détecter dans chaque image RGB, puis reconstruire sa position 3D. Ceci serait facilement réalisable, car techniquement proche de la détection de cercles lors de l'étape de calibrage. Cependant, cette méthode n'est absolument pas robuste dans le cas d'auto-occlusions (par exemple si la main passe devant la pastille), ou d'occlusions (par exemple avec un drap).

Une reconstruction sensible à la position des IMU

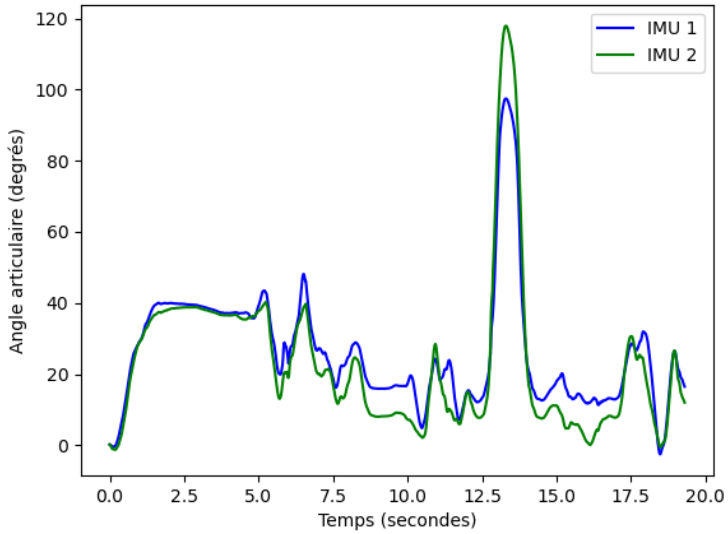
Les aléas d'une expérience rendent les résultats d'une intégration de données IMU extrêmement sensibles aux plus petits changements dans leur placement sur le corps du patient. Nous montrons cela par une expérience inspirée des travaux de (Hafer et al.) [61]. Nous considérons une analyse de la marche d'un sujet sur lequel sont placés 8 IMU : sternum (1), pelvis (1), cuisses (2), tibias (2), pieds (2). Pour les IMU des cuisses et des tibias, nous plaçons des senseurs supplémentaires, comme montré en **Figure 6.6**. Nous reconstruisons alors le mouvement du sujet avec deux jeux de données. Dans chaque jeu, les données du sternum, pelvis et des pieds sont identiques, mais les données des cuisses et des tibias proviennent soit de la paire 1, soit de la paire 2.

FIGURE 6.5 Emplacement des senseurs du jeu 1 et du jeu 2. Seuls les IMU de la cuisse et du tibia changent. Les IMU du sternum, du pelvis et du sternum sont identiques pour les deux jeux.

Lors de l'acquisition, le sujet fait un aller-retour en marchant. Sur le trajet du retour, il lève

particulièrement haut la jambe droite, comme montré en **Figure 6.6a**.

(a)



(b)

FIGURE 6.6 Intégration des données d'une expérience où le sujet marche en aller-retour (a) Sur la phase de retour, le sujet s'arrête et lève très haut son genou droit. (b) Comparaison de l'angle articulaire de la hanche droite obtenu soit en intégrant les données du jeu 1 (en bleu), soit les données du jeu 2 (en vert).

Suite à l'intégration de nos données, nous nous penchons sur les valeurs de flexion de la hanche droite. Nous constatons qu'avec un décalage de simplement quelques centimètres, l'intégration des données IMU donne lieu à des angles articulaires très différents. En effet, la **Figure 6.6b** présente l'évolution au cours du temps de l'angle articulaire au niveau de la hanche droite, en fonction de quels IMU ont été utilisés, et nous pouvons constater des

écarts importants entre les deux courbes. Le pic à 14 secondes, qui correspond au moment de flexion maximale montré en **Figure 6.6a**, montre un décalage de presque 20 degrés entre les deux reconstructions. Partout ailleurs, on peut observer des écarts dépassant parfois les 10 degrés (à la 10ème et à la 16ème seconde)

Cette sensibilité à l'emplacement des IMU physiques est d'autant plus bloquante que l'emplacement des IMU virtuels sur le modèle OpenSim lors de la cinématique inverse ne peut pas être modifié sans une modification du code source C++.

Enfin, afin d'assurer une reconstruction optimale, les IMU doivent être particulièrement bien fixés aux membres respectifs du patient, et rester au même endroit durant toute l'acquisition. Cette contrainte est facilement respectable dans le cas d'une acquisition sur un adulte debout. Cependant, il est probable que les frottements avec les draps et le matelas amènent de faibles variations dans le cas de patients alités, encore plus avec des enfants.

6.4 Conclusion sur l'annotation automatique de pose

Dans cette section, nous avons exploré l'utilisation de capteurs inertiels IMU afin d'annoter automatiquement la pose 3D de nos patients. Notre méthode a proposé des résultats intéressants, et permet de reconstruire assez globalement les mouvements d'un sujet, mais présente 3 limitations. Premièrement, nos résultats perdent en précision dès que les mouvements sortent du plan sagittal. Deuxièmement, notre reconstruction est centrée sur le bassin et ne suis pas le déplacement du sujet. Enfin, notre intégration souffre d'une grande sensibilité à l'emplacement des IMU sur le membre du patient.

Notre protocole d'acquisition de données IMU étant basé sur une architecture parallélisée, nous pouvons le traiter comme un bloc indépendant de l'acquisition vidéo. Il peut ainsi facilement être ajouté ou enlevé du protocole, tel une solution "brancher et utiliser" (solution "plug-and-play"). La combinaison de ces deux systèmes d'acquisition est justement le sujet du prochain et dernier chapitre.

CHAPITRE 7 MISE BOUT À BOUT DU PROTOCOLE D'ACQUISITION

Jusqu'à présent, nous avons traité les sections concernant l'aspect d'acquisition vidéo et l'aspect génération de pose indépendamment. Nous allons désormais présenter le rendu final de notre protocole d'acquisition.

7.1 Interface Utilisateur

Afin de faciliter le plus possible l'acquisition de données sur un nouveau patient, nous créons une interface utilisateur la plus intuitive et simple possible. Nous la présentons en **Figure 7.1**.

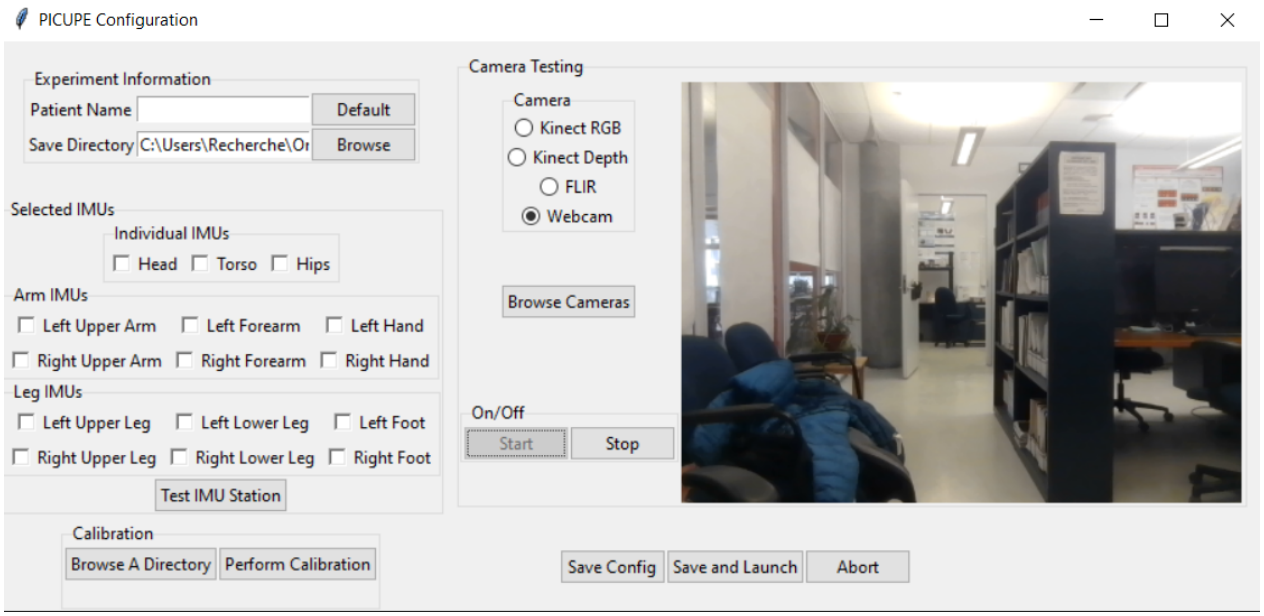


FIGURE 7.1 Interface utilisateur pour le début d'une acquisition.

Cette interface est divisée en 5 grandes sections :

— En haut à gauche, l'utilisateur choisit le nom de son expérience ainsi que l'emplacement de sauvegarde des données. Un dossier au nom de l'expérience sera alors créé.

— Au milieu à gauche, la sélection de tous les IMU que l'on veut utiliser lors de l'acquisition. Un bouton permet aussi de tester que la station Awinda est bien connectée à l'ordinateur. L'utilisateur a le choix de ne sélectionner aucun IMU. Dans ce cas, seules les images seront enregistrées.

— En bas à gauche, la possibilité de réaliser calibration et stéréocalibration des caméras. Si elle a déjà été réalisée, on peut indiquer les fichiers de calibration à réutiliser.

— Sur la partie droite, la possibilité de tester la connexion de chaque caméra, et d'avoir un aperçu de leur champ de vision.

— En bas à droite, le bouton pour lancer l'acquisition.

Lorsque l'utilisateur choisit de calibrer le système vidéo, une nouvelle interface montrée en **Figure 7.2** lui permet de choisir les paramètres N et n définis dans les **Sections 5 et 5.3.2**. Comme nous l'avons précisé dans la **Section 5.2.5**, nous utilisons comme données de la calibration de la Kinect les valeurs de sorties d'usine fournies par le manufacturier.

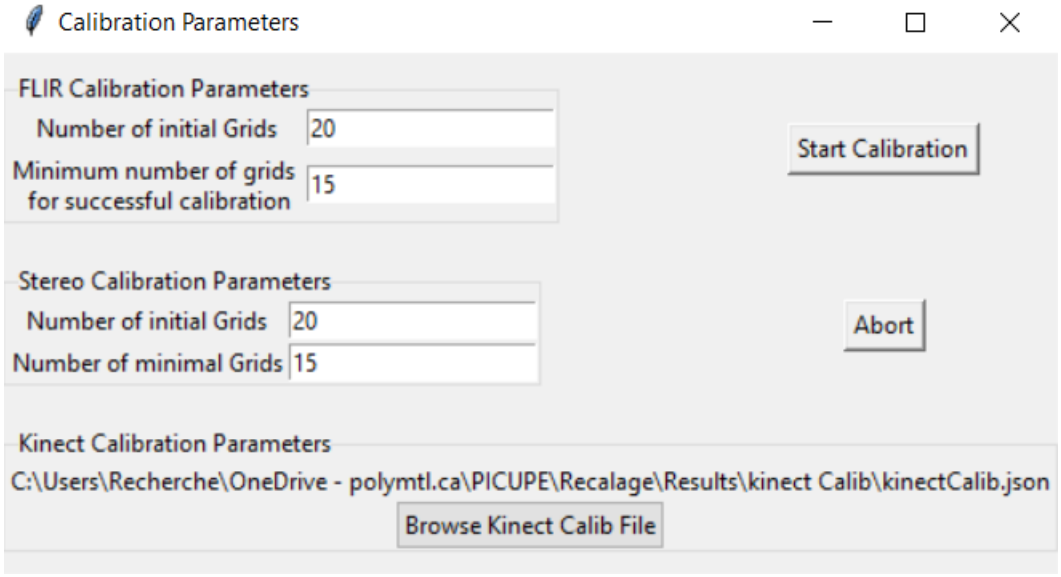


FIGURE 7.2 Interface Utilisateur initiale pour définir les paramètres de la calibration du système stéréo de caméras.

7.2 Log d'une acquisition

Lors d'une acquisition, tous les processus sont "surveillés" par un thread exécuté dans le processus principal. Ce dernier fait remonter les informations des différents processus, et les écrit en temps réel dans un fichier de log, dont un exemple est donné en **Annexe B**. Ceci est particulièrement intéressant dans le cas d'une erreur quelconque survenant lors de l'acquisition, afin de faciliter l'étape de débogage (surtout durant l'étape de développement du protocole !).

7.3 Organisation Finale

Une fois l'acquisition terminée, les images sont écrites au disque ainsi que les données IMU, qui sont immédiatement converties en fichiers texte lisibles par OpenSim. Une fonction simple prenant en argument uniquement le nom de l'expérience peut alors être exécutée afin d'intégrer les données IMU dans OpenSim et générer les poses associées. Nous obtenons alors un dossier final dont l'arborescence est montrée en **Figure 7.3**. Afin de ne pas surcharger la figure, nous montrons uniquement les informations clé qui sont sauvegardées, même si d'autres métadonnées sont stockées en pratique.

Nous générons un dossier pour chaque modalité : profondeur, RGB et IR. Toutes les images sont stockées au format binaire de la libraire python *numpy*. Chaque image est nommée par ordre chronologique, et possède toujours 5 chiffres (par exemple *00234.npy*). Ceci permet d'assurer que les images sont toujours lues dans l'ordre chronologique par un fichier de script qui parcourt le dossier.

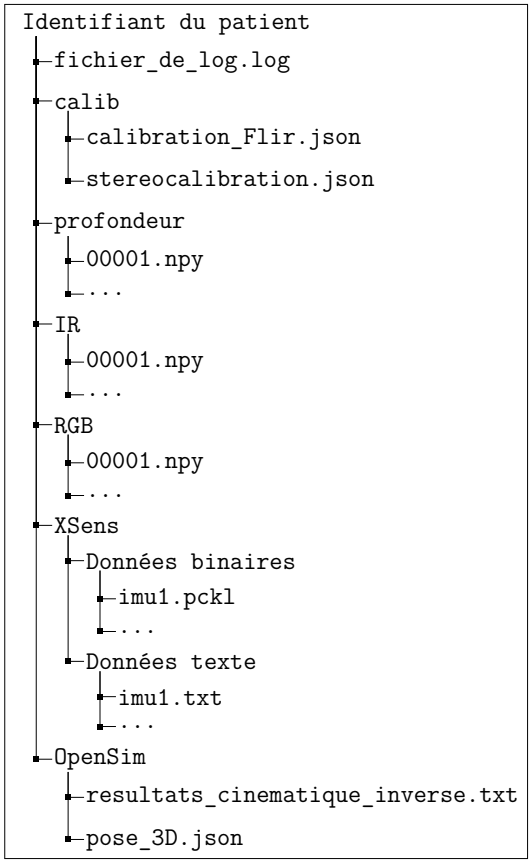


FIGURE 7.3 Arborescence finale d'une acquisition. On obtient 6 sous dossiers ainsi qu'un fichier de log.

7.4 Mise en pratique au CHUSJ - Patient 0

En conclusion, nous voulons montrer un exemple obtenu sur un patient 0 dans une salle de simulations de soins intensifs du CHUSJ. Le patient 0 était un enfant sain qui n'était pas hospitalisé. Plusieurs caméras indépendantes de celles de notre protocole ont filmé l'expérience d'un point de vue extérieur, et nous en montrons quelques images en **Figure 7.4**.

(a)

(b)

(c)

FIGURE 7.4 Expérience sur le patient 0 vue depuis différents angles extérieurs. (a) Le pédiatre pose un IMU sur l'avant-bras gauche, vue du dessus. (b) Le pédiatre pose un IMU sur le bras gauche, vue du côté. (c) Acquisition en cours du patient 0. On distingue bien le support des deux caméras, et on devine la station IMU avec ses deux voyants rouges.

Afin de maintenir les IMU en place, le pédiatre utilise du strap Coban, qui a l'avantage d'être auto-adhésif, donc qui n'irritera pas le patient (ce type de strap est couramment utilisé aux soins intensifs du CHUSJ). Cependant, il doit être très serré afin de maintenir l'IMU sur le membre, et nous avons constaté que le patient se plaignait d'une gêne. En **Figure 7.4c**, on

le voit essayer de d'enlever l'IMU.

Dû à une erreur de transmission des données IMU au moment de l'expérience, nous n'avons malheureusement pas d'annotation automatique à présenter pour le patient 0. Cependant, nous montrons en **Figure 7.5** le rendu visuel obtenu.

7.5 Conclusion sur le protocole

Nous avons donc créé un protocole d'acquisition simple et rapide à mettre en place, notamment grâce à son interface graphique. L'écriture des données est organisée dans des dossiers avec une arborescence rigoureuse et identique pour chaque expérience. L'acquisition des données OpenSim est entièrement modulable, tout comme la nécessité de calibrer le système stéréo. Enfin, l'ensemble des hyperparamètres et des évènements de l'expérience sont écrits dans un fichier log clair et concis. Nous considérons que notre solution est prête pour créer la base de données PICUPE.

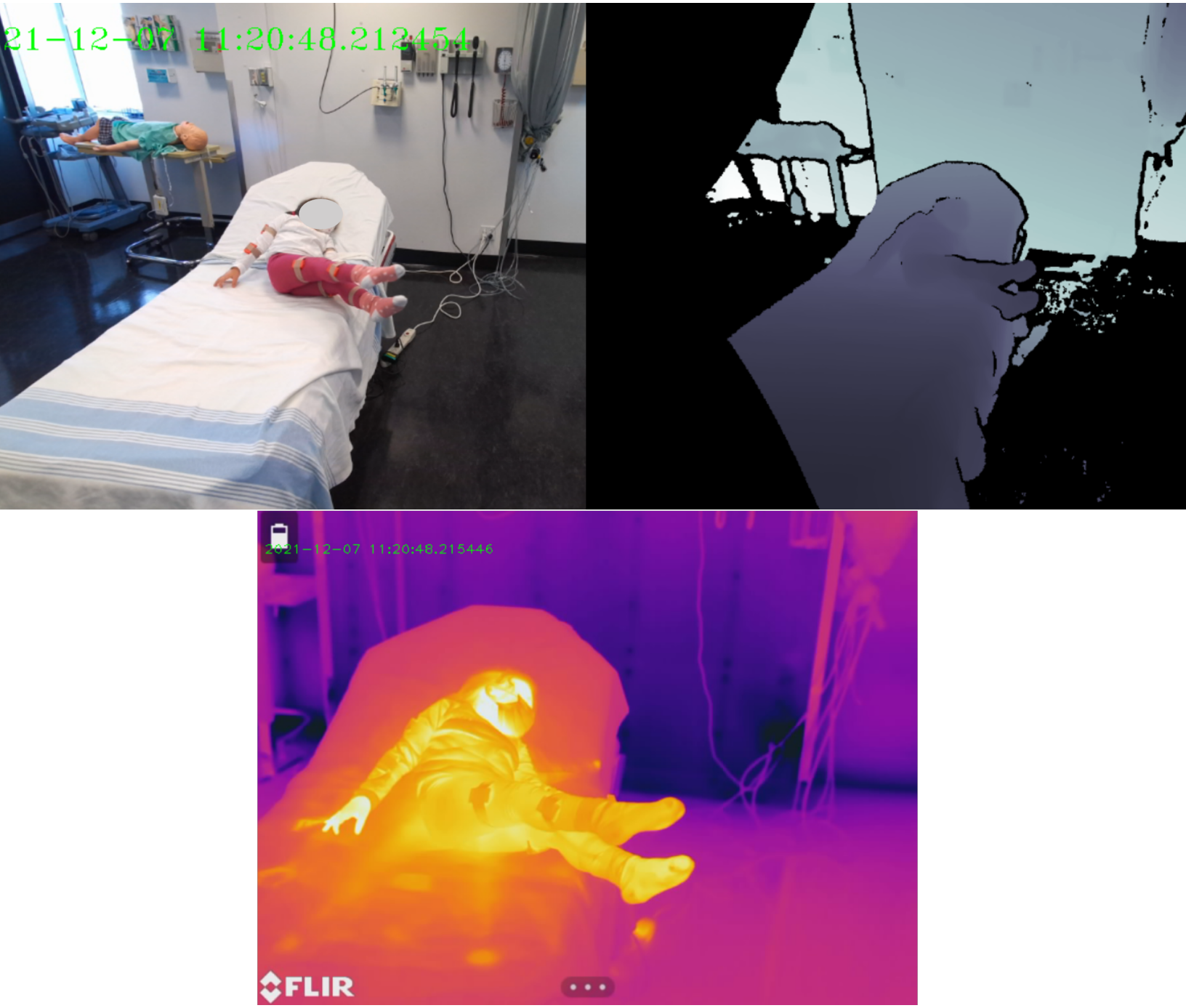


FIGURE 7.5 Visualisation de la partie vidéo de notre acquisition sur le patient 0.

CHAPITRE 8 CONCLUSION

8.1 Synthèse des travaux

Nous avons proposé un protocole pour la création d'une base de données vidéo d'enfants alités aux soins intensifs du CHUSJ. Approuvé par le comité éthique du CHUSJ, notre système permet l'acquisition d'images multimodales (RGB, Profondeur, IR) synchrones à une fréquence de 29 FPS. En seulement quelques minutes, nous permettons à un utilisateur de calibrer ce système avec un minimum de matériel. Posé sur un bras articulé, notre montage est facilement transportable, et ne nécessite qu'une unique prise pour brancher la caméra Kinect. Il pourrait donc être utilisé dans des contextes autres que le projet PICUPE. Ce système, facilement exploitable par son interface graphique, permettra la création d'une large base de données d'enfants alités aux Soins Intensifs du CHUSJ.

Nous avons aussi développé une méthode de génération de pose en 3D synchronisée avec le système vidéo, en intégrant des données inertielles mesurées par des capteurs IMU. Nous avons pu obtenir des premiers résultats satisfaisants sur des adultes dans un environnement contrôlé, même si nous n'avons pas pu valider quantitativement la précision de notre méthode.

8.2 Limitations de la solution proposée

D'une part, nous pouvons être satisfait de la partie vision de ce projet, qui offre de très bons résultats. Dans le futur, nous pourrions considérer l'utilisation d'une caméra thermique à longueur focale fixe. Ce faisant, l'étape de stéréocalibration n'aura pas besoin d'être répétée à chaque nouvelle expérience.

D'autre part, nous devons concéder que plusieurs limitations existent dans notre approche de génération de pose. Entre une étape de calibration trop peu précise, une grande sensibilité des résultats à l'emplacement des IMU sur les membres, et enfin une absence de suivi de la position du bassin dans l'espace, nous concluons que cette méthode d'annotation n'est pas suffisamment rigoureuse pour être mise en place dans le protocole final implémenté au CHUSJ.

8.3 Travaux futurs

Nous avons vu que l'approche la plus pertinente pour l'EPH dans le contexte hospitalier était d'utiliser un réseau de neurones sur des images de profondeur et thermiques. Cepen-

dant, notre approche visait à entraîner notre réseau de manière supervisée, c'est-à-dire en apprenant des poses annotées. Cependant, de récents travaux [62] utilisent une approche non-supervisée pour faire de l'EPH, notamment sur des enfants dans un contexte hospitalier. Une des nouvelles directions de PICUPE pourrait être de développer une méthode non-supervisée, justifiée avec des annotations manuelles.

RÉFÉRENCES

[1] C. Zheng, W. Wu, T. Yang, S. Zhu, C. Chen, R. Liu, J. Shen, N. Kehtarnavaz et M. Shah, "Deep learning-based human pose estimation : A survey," *CoRR*, vol. abs/2012.13392, 2020. [En ligne]. Disponible : https://arxiv.org/abs/2012.13392

[2] J. P. S. Cunha, H. M. P. Choupina, A. P. Rocha, J. M. Fernandes, F. Achilles, A. M. Loesch, C. Vollmar, E. Hartl et S. Noachtar, "Neurokinect : A novel low-cost 3dvideo-eeg system for epileptic seizure motion quantification," *PLOS ONE*, vol. 11, p. 1–17, 01 2016. [En ligne]. Disponible : https://doi.org/10.1371/journal.pone.0145669

[3] Y. Yin, J. P. Robinson et Y. Fu, "Multimodal in-bed pose and shape estimation under the blankets," *CoRR*, vol. abs/2012.06735, 2020. [En ligne]. Disponible : https://arxiv.org/abs/2012.06735

[4] S. Liu, X. Huang, N. Fu, C. Li, Z. Su et S. Ostadabbas, "Simultaneously-collected multimodal lying pose dataset : Towards in-bed human pose monitoring under adverse vision conditions," *arXiv preprint arXiv :2008.08735*, 2020.

[5] C. Ionescu, D. Papava, V. Olaru et C. Sminchisescu, "Human3.6m : Large scale datasets and predictive methods for 3d human sensing in natural environments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, n°. 7, p. 1325–1339, jul 2014.

[6] G. Di Raimondo, B. Vanwanseele, A. van der Have, J. Emmerzaal, M. Willems, B. A. Killen et I. Jonkers, "Inertial sensor-to-segment calibration for accurate 3d joint angle calculation for use in opensim," *Sensors*, vol. 22, n°. 9, 2022. [En ligne]. Disponible : https://www.mdpi.com/1424-8220/22/9/3259

[7] Xsens, "Mtw awinda." [En ligne]. Disponible : https://www.xsens.com/products/mtw-awinda

[8] "Opensense - kinematics with imu data." [En ligne]. Disponible : https://simtk-confluence.stanford.edu:8443/display/OpenSim/OpenSense+-+Kinematics+with+IMU+Data

[9] G. Jones, "Rapport de stage - protocole d'acquisition vidéo avec une caméra kinect."

[10] N. Hesse, C. Bodensteiner, M. Arens, U. G. Hofmann, R. Weinberger et A. S. Schroeder, "Computer vision for medical infant motion analysis : State of the art and RGB-D data set," dans *Computer Vision - ECCV 2018 Workshops*. Springer International Publishing, 2018.

[11] X. Huang, N. Fu et S. Ostadabbas, “Infant pose learning with small data,” *CoRR*, vol. abs/2010.06100, 2020. [En ligne]. Disponible : https://arxiv.org/abs/2010.06100

[12] K. Sun, B. Xiao, D. Liu et J. Wang, “Deep high-resolution representation learning for human pose estimation,” *CoRR*, vol. abs/1902.09212, 2019. [En ligne]. Disponible : http://arxiv.org/abs/1902.09212

[13] R. Boutteau, “Reconstruction tridimensionnelle de l’environnement d’un robot mobile, à partir d’informations de vision omnidirectionnelle, pour la préparation d’interventions,” Thèse de doctorat, 04 2010.

[14] G. R. Bradski et A. Kaehler, *Learning OpenCV*. Sebastopol, CA : O’Reilly Media, oct. 2008.

[15] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll et M. J. Black, “SMPL : A skinned multi-person linear model,” *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, vol. 34, n°. 6, p. 248 :1–248 :16, oct. 2015.

[16] V. Kress, J. Jung, S. Zernetsch, K. Doll et B. Sick, “Human pose estimation in real traffic scenes,” dans *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, p. 518–523.

[17] Z. Li, L. Chen, C. Liu, Y. Gao, Y. Ha, C. Xu, S. Quan et Y. Xu, “3d human avatar digitization from a single image,” ser. VRCAI ’19. New York, NY, USA : Association for Computing Machinery, 2019. [En ligne]. Disponible : https://doi.org/10.1145/3359997.3365707

[18] C. Einspieler, H. F. Prechtl, F. Ferrari, G. Cioni et A. F. Bos, “The qualitative assessment of general movements in preterm, term and young infants — review of the methodology,” *Early Human Development*, vol. 50, n°. 1, p. 47–60, 1997, spontaneous Motor Activity as a Diagnostic Tool Functional Assessment of the Young Nervous System. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0378378297000923

[19] H. Prechtl, “Qualitative changes of spontaneous movements in fetus and preterm infants are a marker of neurological dysfunction.” *Early Human Development*, vol. 23, n°. 3, p. 151–158, sept. 1990.

[20] L. Adde, M. Rygg, K. Lossius, G. K. Øberg et R. Støen, “General movement assessment : Predicting cerebral palsy in clinical practise,” *Early Human Development*, vol. 83, n°. 1, p. 13–18, 2007. [En ligne]. Disponible : https://www.sciencedirect.com/science/article/pii/S0378378206000892

[21] P. Teitelbaum, O. Teitelbaum, J. Nye, J. Fryman et R. G. Maurer, “Movement analysis in infancy may be useful for early diagnosis of autism,” *Proceedings of the National*

Academy of Sciences, vol. 95, nº. 23, p. 13 982–13 987, 1998. [En ligne]. Disponible : https://www.pnas.org/doi/abs/10.1073/pnas.95.23.13982

[22] V. Marchi, A. Hakala, A. Knight, F. D'Acunto, M. L. Scattoni, A. Guzzetta et S. Vanhatalo, "Automated pose estimation captures key aspects of general movements at eight to 17 weeks from conventional videos," *Acta Paediatrica*, vol. 108, nº. 10, p. 1817–1824, 2019. [En ligne]. Disponible : https://onlinelibrary.wiley.com/doi/abs/10.1111/apa.14781

[23] C. Chambers, N. Seethapathi, R. Saluja, H. Loeb, S. Pierce, D. Bogen, L. Prosser, M. Johnson et K. Kording, "Computer vision to automatically assess infant neuromotor risk," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 28, p. 2431–2442, 11 2020.

[24] A. Stahl, C. Schellewald, O. Stavdahl, O. M. Aamo, L. Adde et H. Kirkerod, "An optical flow-based method to predict infantile cerebral palsy," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, nº. 4, p. 605–614, 2012.

[25] M. H. Li, T. A. Mestre, S. H. Fox et B. Taati, "Vision-based assessment of parkinsonism and levodopa-induced dyskinesia with deep learning pose estimation," *CoRR*, vol. abs/1707.09416, 2017. [En ligne]. Disponible : http://arxiv.org/abs/1707.09416

[26] L. Lonini, Y. Moon, K. Embry, R. J. Cotton, K. McKenzie, S. Jenz et A. Jayaraman, "Video-based pose estimation for gait analysis in stroke survivors during clinical assessments : A proof-of-concept study," *Digital Biomarkers*, vol. 6, nº. 1, p. 9–18, 2022. [En ligne]. Disponible : https://www.karger.com/DOI/10.1159/000520732

[27] C. H. Lee, D. K. Kim, S. Y. Kim, C.-S. Rhee et T.-B. Won, "Changes in site of obstruction in obstructive sleep apnea patients according to sleep position : a DISE study."

[28] S. Akbarian, G. Delfi, K. Zhu, A. Yadollahi et B. Taati, "Automated non-contact detection of head and body positions during sleep," *IEEE Access*, vol. 7, p. 72 826–72 834, 2019.

[29] K. Chen, P. Gabriel, A. Alasfour, C. Gong, W. K. Doyle, O. Devinsky, D. Friedman, P. Dugan, L. Melloni, T. Thesen, D. Gonda, S. Sattar, S. Wang et V. Gilja, "Patient-specific pose estimation in clinical environments," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 6, p. 1–11, 2018.

[30] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu et E.-h. Zahzah, "Human pose estimation from monocular images : A comprehensive survey," *Sensors*, vol. 16, nº. 12, 2016. [En ligne]. Disponible : https://www.mdpi.com/1424-8220/16/12/1966

[31] A. Toshev et C. Szegedy, “Deeppose : Human pose estimation via deep neural networks,” *CoRR*, vol. abs/1312.4659, 2013. [En ligne]. Disponible : http://arxiv.org/abs/1312.4659

[32] S. Giancola, M. Valenti et R. Sala, “A survey on 3d cameras : Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies,” dans *SpringerBriefs in Computer Science*, 2018.

[33] T. Schnürer, S. Fuchs, M. Eisenbach et H.-M. Gross, “Real-time 3d pose estimation from single depth images,” 01 2019, p. 716–724.

[34] A. Newell, K. Yang et J. Deng, “Stacked hourglass networks for human pose estimation,” *CoRR*, vol. abs/1603.06937, 2016. [En ligne]. Disponible : http://arxiv.org/abs/1603.06937

[35] S. Liu, Y. Yin et S. Ostadabbas, “In-bed pose estimation : Deep learning with shallow dataset,” *CoRR*, vol. abs/1711.01005, 2017. [En ligne]. Disponible : http://arxiv.org/abs/1711.01005

[36] S. Liu et S. Ostadabbas, “Seeing under the cover : A physics guided learning approach for in-bed pose estimation,” *CoRR*, vol. abs/1907.02161, 2019. [En ligne]. Disponible : http://arxiv.org/abs/1907.02161

[37] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur et X. Savatier, “A study of vicon system positioning performance,” *Sensors*, vol. 17, n°. 7, 2017. [En ligne]. Disponible : https://www.mdpi.com/1424-8220/17/7/1591

[38] C. Pizzolato, M. Reggiani, L. Modenese et D. G. Lloyd, “Real-time inverse kinematics and inverse dynamics for lower limb applications using opensim,” *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 20, n°. 4, p. 436–445, 2017.

[39] A. Rajagopal, C. L. Dembia, M. S. DeMers, D. D. Delp, J. L. Hicks et S. L. Delp, “Full-body musculoskeletal model for muscle-driven simulation of human gait,” *IEEE Transactions on Biomedical Engineering*, vol. 63, n°. 10, p. 2068–2079, 2016.

[40] F. Achilles, A.-E. Ichim, H. Coskun, F. Tombari, S. Noachtar et N. Navab, “Patient mocap : Human pose estimation under blanket occlusion for hospital monitoring applications,” dans *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal et W. Wells, édit. Cham : Springer International Publishing, 2016, p. 491–499.

[41] L. Hansen, M. Siebert, J. Diesel et M. Heinrich, “Fusing information from multiple 2d depth cameras for 3d human pose estimation in the operating room,” *International Journal of Computer Assisted Radiology and Surgery*, vol. 14, 08 2019.

[42] S. Ostadabbas, M. B. Pouyan, M. Nourani et N. Kehtarnavaz, “In-bed posture classification and limb identification,” *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings*, p. 133–136, 2014.

[43] V. Davoodnia, S. Ghorbani et A. Etemad, “In-bed pressure-based pose estimation using image space representation learning,” 08 2019.

[44] F. Wittmann, O. Lambercy et R. Gassert, “Magnetometer-based drift correction during rest in imu arm motion tracking,” *Sensors (Basel, Switzerland)*, vol. 19, 2019.

[45] B. Fan, Q. Li et T. Liu, “How magnetic disturbance influences the attitude and heading in magnetic and inertial sensor-based orientation estimation,” *Sensors*, vol. 18, n°. 1, 2018. [En ligne]. Disponible : https://www.mdpi.com/1424-8220/18/1/76

[46] M. Al-Amri, K. Nicholas, K. Button, V. Sparkes, L. Sheeran et J. L. Davies, “Inertial measurement units for clinical movement analysis : Reliability and concurrent validity,” *Sensors (Basel, Switzerland)*, vol. 18, 2018.

[47] A. Machireddy, J. van Santen, J. L. Wilson, J. Myers, M. Hadders-Algra et X. Song, “A video/imu hybrid system for movement estimation in infants,” dans *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2017, p. 730–733.

[48] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht et L. Schmidt, “Measuring robustness to natural distribution shifts in image classification,” *CoRR*, vol. abs/2007.00644, 2020. [En ligne]. Disponible : https://arxiv.org/abs/2007.00644

[49] M. Andriluka, L. Pishchulin, P. Gehler et B. Schiele, “2d human pose estimation : New benchmark and state of the art analysis,” dans *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[50] T. Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár et C. L. Zitnick, “Microsoft COCO : common objects in context,” *CoRR*, vol. abs/1405.0312, 2014. [En ligne]. Disponible : http://arxiv.org/abs/1405.0312

[51] G. Sciortino, G. Farinella, S. Battiato, M. Leo et C. Distante, “On the estimation of children’s poses,” 10 2017, p. 410–421.

[52] S. Gonzalez Perez, D. Perea Strom, N. Arteaga-Marrero, C. Luque, I. Sidrach-Cardona, E. Villa et J. Ruiz-Alzola, “Assessment of registration methods for thermal infrared and visible images for diabetic foot monitoring,” *Sensors (Basel)*, vol. 21, n°. 7, mars 2021.

[53] N. Hesse, S. Pujades, J. Romero, M. J. Black, C. Bodensteiner, M. Arens, U. G. Hofmann, U. Tacke, M. Hadders-Algra, R. Weinberger, W. Müller-Felber et A. S. Schroeder,

"Learning an infant body model from RGB-D data for accurate full body motion analysis," dans *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer, 2018.

[54] "Flir t1020." [En ligne]. Disponible : https://www.flir.com/products/t1020/

[55] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, n°. 11, p. 1330–1334, 2000.

[56] C. Ricolfe-Viala et A. Esparza, "The influence of autofocus lenses in the camera calibration process," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, p. 1–15, 2021.

[57] X. Chen, X. Wu, S. Gao, X. Xie et Y. Huang, "Synchronization and calibration of a stereo vision system," dans *Global Oceans 2020 : Singapore – U.S. Gulf Coast*, 2020, p. 1–6.

[58] Mverleg, "Mverleg - array-storage-benchmark : Compare some methods of array storage in Python (numpy)." [En ligne]. Disponible : https://github.com/mverleg/array_storage_benchmark

[59] M. Al Borno, J. Day, V. Ibarra, J. Dunne, A. Seth, A. Habib, C. Ong, J. Hicks, S. Uhlrich et S. Delp, "Opensense : An open-source toolbox for inertial-measurement-unit-based measurement of lower extremity kinematics over long durations," *bioRxiv*, 2021. [En ligne]. Disponible : https://www.biorxiv.org/content/early/2021/07/02/2021.07.01.450788

[60] H. Yan, Q. Shan et Y. Furukawa, "Ridi : Robust imu double integration," 2017. [En ligne]. Disponible : https://arxiv.org/abs/1712.09004

[61] J. Hafer, J. Mihy, A. Hunt, R. Zernicke et R. Johnson, "Imu-derived kinematics detect gait differences with age or knee osteoarthritis but differ from marker-derived inverse kinematics," 2022. [En ligne]. Disponible : https://doi.org/10.1101/2022.01.10.22269024

[62] L. Schmidtke, A. Vlontzos, S. Ellershaw, A. Lukens, T. Arichi et B. Kainz, "Unsupervised human pose estimation through transforming shape templates," *CoRR*, vol. abs/2105.04154, 2021. [En ligne]. Disponible : https://arxiv.org/abs/2105.04154

ANNEXE A THÉORIE DE CALIBRATION ET STÉRÉOCALIBRATION

A.1. Modélisation d'une caméra

Dans cette section, nous donnons une explication succincte du principe de calibration, et introduisons les grandeurs qui nous serons utiles dans la suite de ce rapport.

A.1.1. Caméra parfaite : le modèle de sténopé

La modélisation la plus courante de formation d'une image dans une caméra est le modèle de sténopé (*pinhole* en anglais). Dans cette modélisation, la projection du monde réel 3D en une image 2D peut facilement être représenté par des opérations matricielles.

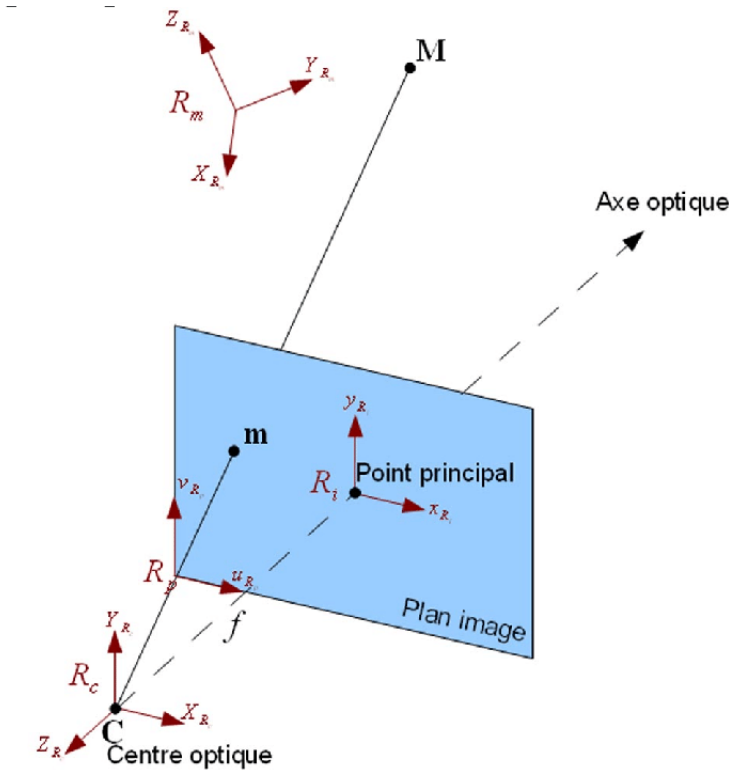


FIGURE A.1 Le modèle de sténopé, représentation la plus courante du fonctionnement d'une caméra. Source : [13]

Dans le modèle de sténopé, tous les rayons lumineux d'une scène convergent vers un point virtuel situé derrière le senseur capturant la lumière, nommé *Centre Optique*. Le senseur (représenté par le *Plan Image*) n'étant pas parfaitement centré autour de l'*axe optique*, nous définissons le *Point principal*, qui correspond au point d'intersection entre l'axe optique et le

plan image. Un point sur l'écran a pour coordonnées (x, y, f) où f est la *distance focale*. Le plus souvent, les coordonnées de l'objet considéré (représenté par le point M) sont représentées dans un référentiel absolu noté *Référentiel monde*.

La transformation des coordonnées 3D de notre point M(X, Y, Z) vu dans le référentiel monde, en des coordonnées de pixels m(u,v) dans le référentiel du plan image peut-être décomposée en deux étapes. La première consiste en une transformation rigide $\mathbf{R}$ qui permet d'exprimer les coordonnées de M dans le référentiel de la caméra. Cette transformation rigide est elle-même décomposée en une matrice de rotation $\mathbf{R}_{3\times3}$ et une matrice de translation $\mathbf{T}$. La deuxième transformation consiste en une projection perspective P qui permet de passer du point M au point m, représenté dans le plan image en unité métriques (x, y), suivie d'une transformation affine A, qui représente le point m en pixels (u, v).

La projection 3D en 2D d'une caméra peut donc être représentée par la transformation :

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} * \mathbf{R} * \begin{pmatrix} X_m \\ Y_m \\ Z_m \\ 1 \end{pmatrix} \tag{A.1}$$

$$Avec\ \mathbf{K} = \mathbf{A} * \mathbf{P} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{A.2}$$

$$Et\ \mathbf{R} = \begin{bmatrix} \mathbf{R}_{3\times3} & \mathbf{T} \\ \mathbf{0}_{3\times1} & 1 \end{bmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.3}$$

La matrice $\mathbf{K}$ est nommée matrice de **paramètres intrinsèques**, car elle contient uniquement des données inhérentes à la caméra : sa distance focale et les coordonnées de son point principal. Si nous introduisons dans l'**Équation A.2** deux distances focales f_x et f_y, nous verrons que nous pouvons faire l'hypothèse raisonnable d'une unique distance focale : $f_x = f_y = f$.

La matrice $\mathbf{R}$ est nommée matrice des **paramètres extrinsèques** de la caméra. Elle représente la transformation rigide entre le référentiel monde et le référentiel de la caméra. Lorsque la caméra est déplacée dans l'espace, cette matrice est évidemment modifiée. Comme montré dans l'**Équation A.3**, elle se décompose en une matrice de rotation et une matrice de

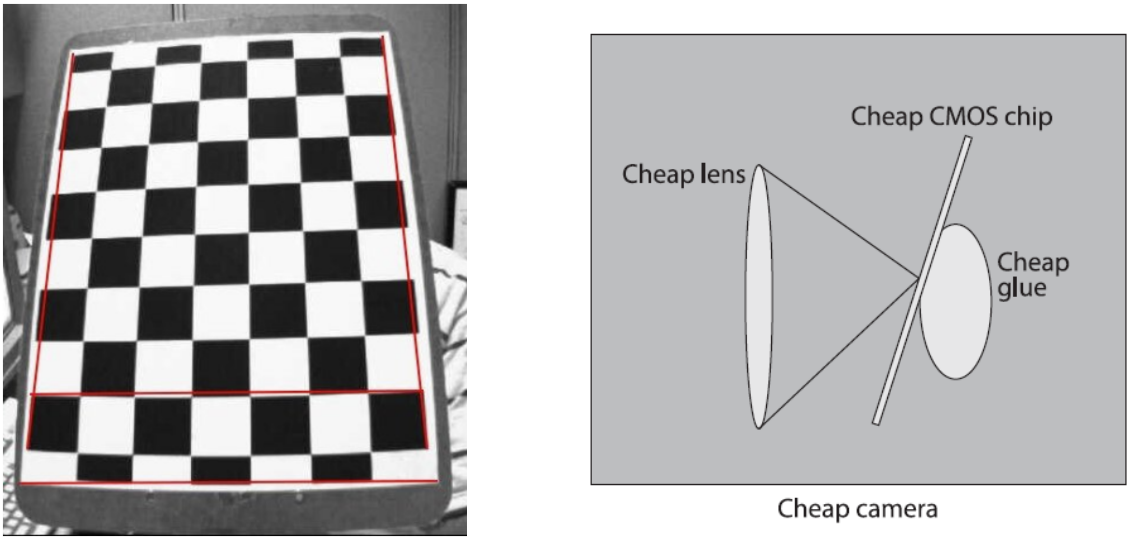
translation.

A.1.2. Caméra réelle : phénomènes de distorsion

L'approche du modèle de sténopé est rigoureuse, à l'exception près qu'elle ne prend pas en compte les deux phénomènes de distorsion liés à une lentille.

Une lentille étant sphérique, elle introduit un phénomène de **distorsion radiale**. En effet, les rayons éloignés du centre de la lentille ont tendance à être plus "tordus" que les rayons centrés. Ceci traduit dans une image par une courbure des lignes droites proche des bords de notre image, comme montré en **Figure A.2a**. Les phénomènes de distorsion peuvent cependant être corrigés numériquement via un développement de Taylor autour de r, la distance radiale du point au centre de la lentille. En considérant un point $m(x, y)$ sur le plan image, le point corrigé (i.e. sans distorsion) $m_c(x_c, y_c)$ a pour coordonnées :

$$\begin{cases} x_c = x * (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_c = y * (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \quad Avec\ r = x^2 + y^2 \tag{A.4}$$



(a) **(b)**

FIGURE A.2 Les phénomènes de distorsion. (a) visualisation du phénomène de distorsion radiale. Les lignes droites du damier apparaissent courbées loin du centre. (b) Explication de l'origine de la distorsion tangentielle : la lentille et le senseur ne sont pas parfaitement parallèles. Source : [14]

Le phénomène de **distorsion tangentielle** pour sa part ne provient pas de la lentille elle-même, mais plutôt de sa position par rapport au senseur optique. En effet, dans le modèle de sténopé, les matrices de transformation sont obtenues en supposant que la lentille et le sen-

seur étaient parfaitement parallèles. Ceci est rarement le cas en pratique, comme représenté en **Figure A.2b**. Ce non-parallélisme peut faire apparaître certains points de l'image plus proches qu'ils ne le sont vraiment. De manière analogue à la distorsion radiale, le phénomène de distorsion tangentielle peut être corrigé par :

$$\begin{cases} x_c = x + [2p_1y + p_2(r^2 + 2x^2)] \\ y_c = y + [2p_2x + p_1(r^2 + 2y^2)] \end{cases} \quad Avec\ r = x^2 + y^2 \tag{A.5}$$

A.2. Stéréocalibration de deux caméras

Nous allons désormais considérer le cas de deux caméras qui filment ensemble une scène. Dans notre cas, ce sera donc les caméra Kinect et FLIR. Afin de rester dans un cas général, nous les nommerons caméras 1 et 2. La caméra 1 correspond à la FLIR et est placée à gauche, la caméra 2 correspond à la Kinect et est placée à droite (comme nous l'avons vu dans la **Figure 5.1a**).

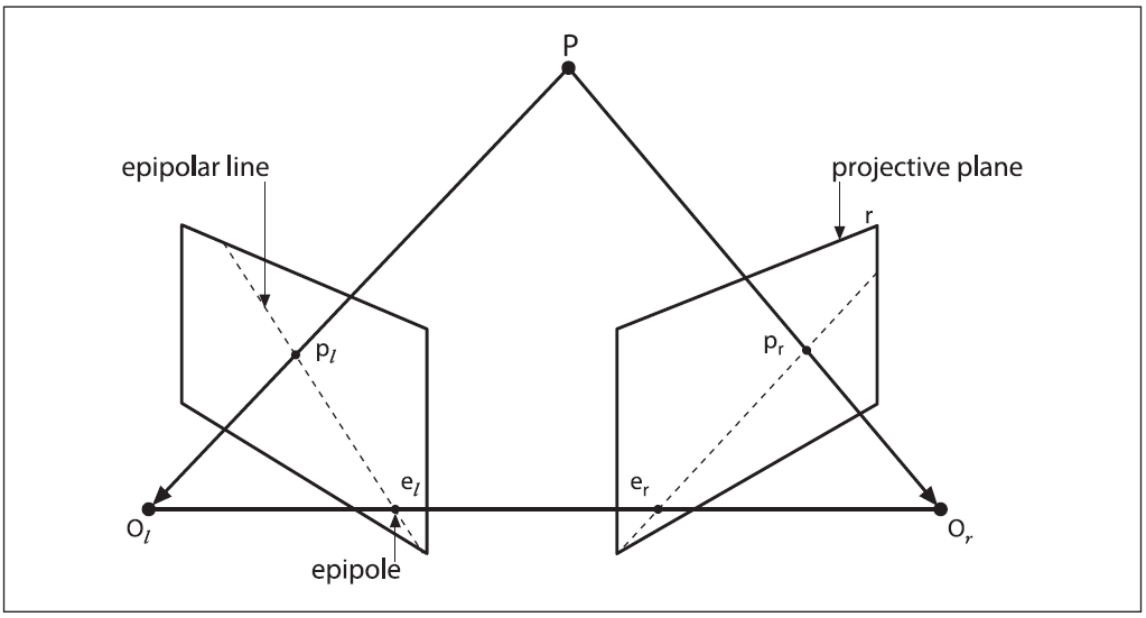


FIGURE A.3 Visualisation schématique de la théorie épipolaire. Source : [14]

Sur la **Figure A.3**, nous observons nos deux caméras 1 et 2 qui observent un point $M(X, Y, Z)$. Nous modélisons les caméras par un modèle de sténopé, et introduisons la notion d'*épipole* :

l'épipôle de la caméra 1 correspond à la projection du centre optique O_2 de la caméra 2 sur le plan image Π_1. La position des épipôles dépend des paramètres intrinsèques des 2 caméras, mais aussi de leurs positions relatives : si l'on déplace les caméras, la position des épipôles sera modifiée. Si ici nous schématisons les épipôles comme appartenant au plan image, ce n'est pas forcément le cas.

Considérons désormais la projection du point M sur le plan image 2, que l'on nomme $m_2(x, y)$. En réalité, m_2 n'est pas uniquement la projection du point M, mais de tous les points situés sur la droite Mm_2. La projection de cette droite sur le plan image 1 est appelée *ligne épipolaire* : elle passe par m_1 la projection de M sur le plan image 1, ainsi que par son épipôle e_1. L'épipôle du plan image est donc le point par lequel passe toutes les lignes épipolaires de ce plan. Nous introduisons aussi le *plan épipolaire*, défini par les deux épipoles et le point M.

Nous introduisons les matrices $\mathbf{R}$ et $\mathbf{T}$ (qui n'est autre que le vecteur O_1O_2) qui représentent le changement de repère entre les repères associés à O_1 et O_2 (et donc qui modélisent la positon relative de la caméra 2 par rapport à la caméra 1). Ainsi, les deux valeurs m_1 et m_2 sont reliées par :

$$m_2 = \mathbf{R}(m_1 - \mathbf{T}) \iff (m_1 - \mathbf{T}) = \mathbf{R}^\top m_2 \tag{A.6}$$

Nous avons vu que les points m_1, m_2 O_1 et O_2 sont coplanaires et définissent le plan épipolaire. En considérant O_1 comme origine, et par définition du produit vectoriel, nous avons :

$$(m_1 - \mathbf{T})^\top (\mathbf{T} \times m_1) = 0 \tag{A.7}$$

En effet, $m_1 - T$ appartient au plan épipolaire, et le produit vectoriel de deux vecteurs d'un plan y est orthogonal. En combinant les **Équation A.6** et **Équation A.7**, nous obtenons :

$$m_2^\top \mathbf{R}(\mathbf{T} \times m_1) = 0 \tag{A.8}$$

Nous pouvons transformer le produit vectoriel $\mathbf{T} \times m_1$ en un produit matriciel en introduisant la matrice $[\mathbf{T}_\times]$ telle que :

$$\mathbf{T} \times m_1 = [\mathbf{T}_\times] m_1 \qquad avec\ [\mathbf{T}_\times] = \begin{bmatrix} 0 & -T_3 & T_2 \\ T_3 & 0 & -T_1 \\ -T_2 & T_1 & 0 \end{bmatrix} \tag{A.9}$$

Ceci nous permet finalement la notion de *matrice essentielle*, notée $\mathbf{E}$, qui encode la position

relative des deux caméras entre elles. Pour tout point M, dont les projections sur Π_1 et Π_2 sont m_1 et m_2, nous avons :

$$\boxed{m_2^\top \mathbf{E} m_1 = 0 \qquad avec\ \mathbf{E} = \mathbf{R}[\mathbf{T}_\times]} \tag{A.10}$$

Pour l'instant, nous n'avons considéré que des transformations de points dans les différents repères, mais n'avons pas impliqué les paramètres intrinsèques des caméras, que l'on nommera $\mathbf{K}_1$ et $\mathbf{K}_2$. En reprenant, l'**Équation A.1**, nous savons que $w_i = K_i m_i | i \in \{1, 2\}$, avec w_i les coordonnées en pixels des points observés. L'**Équation A.10** devient alors :

$$\boxed{w_2^\top \mathbf{F} w_1 = 0 \qquad avec\ \mathbf{F} = \mathbf{K}_2^{-1} \mathbf{E} \mathbf{K}_1^{-1}} \tag{A.11}$$

Nous avons ainsi introduit la *matrice fondamentale* de notre système : elle contient les relations sur les caméras entre elles, ainsi que les informations intrinsèques à chaque caméra.

L'objectif de la stéréocalibration d'un ensemble de caméras est donc de déterminer l'ensemble des matrices qui coordonnent notre système. En pratique, ceci se traduit donc par le calcul des matrices $\mathbf{R}$ et $\mathbf{t}$, et d'en déduire les matrices essentielle $\mathbf{E}$ et fondamentale $\mathbf{F}$ (pour cette dernière, si les caméras n'ont pas été préalablement calibrées, il faudra aussi retrouver les matrices de paramètres intrinsèques $\mathbf{K}_1$ et $\mathbf{K}_2$ des deux caméras.)

ANNEXE B FICHIER DE LOG D'UNE ACQUISITION

```
2022-03-10 15:50:43,040 - INFO - MainProcess: Initialising processes...


2022-03-10 15:50:47,042 - INFO - Xsens: Creating saving directory
2022-03-10 15:50:47,089 - INFO - Xsens: Creating an XsControl object...
2022-03-10 15:50:47,151 - INFO - Xsens: Detecting ports...

2022-03-10 15:50:48,715 - INFO - Xsens: Opening port COM3
2022-03-10 15:50:50,024 - INFO - Xsens: Configuring device...

2022-03-10 15:50:50,725 - INFO - Xsens: Wait 10 seconds before starting data
acquisition. XSens filters are warming up...

2022-03-10 15:51:00,737 - INFO - Xsens: IMUs are ready.

2022-03-10 15:51:00,879 - INFO - MainProcess: Set up all the IMUs. When ready,
press s to start...

2022-03-10 15:51:08,921 - INFO - Xsens: XSens is Recording
2022-03-10 15:51:08,952 - INFO - FLIR Reader: flir is Recording...
2022-03-10 15:51:09,015 - INFO - Kinect Reader: Kinect is Recording...
2022-03-10 15:51:09,095 - INFO - MainProcess: Recording, press enter to stop

2022-03-10 15:52:05,537 - INFO - MainProcess:
Stopping recording...

2022-03-10 15:52:05,538 - INFO - MainProcess: Destroying writer processes...
2022-03-10 15:52:05,541 - INFO - Xsens: Stopping XSens recording...

2022-03-10 15:52:05,541 - INFO - FLIR Writer: Finished FLIR writing. Grabbed 1686
frames in the queue
2022-03-10 15:52:05,541 - INFO - Depth Writer: Finished Depth writing. Grabbed 1686
frames in the queue
2022-03-10 15:52:05,541 - INFO - RGB Writer: Finished RGB writing. Grabbed 1686
frames in the queue
2022-03-10 15:52:05,545 - INFO - Xsens: Writing XSens PICKLE data to .txt ...

2022-03-10 15:52:05,697 - INFO - FLIR Reader: Finished flir acquisition. Put 1687
frames in the queue
2022-03-10 15:52:05,698 - INFO - FLIR Reader: FLIR acquisition lasted 56.702s
2022-03-10 15:52:05,698 - INFO - FLIR Reader: Overall flir FPS: 29.752
2022-03-10 15:52:05,759 - INFO - MainProcess: Closing Kinect...
2022-03-10 15:52:05,760 - INFO - MainProcess: Closing Kinect...
2022-03-10 15:52:05,760 - INFO - MainProcess: Successfully destroyed writer
processes...

2022-03-10 15:52:05,770 - INFO - MainProcess: Finished FLIR writing. Grabbed 1687
frames in the queue
2022-03-10 15:52:05,771 - INFO - MainProcess: Destroying reader processes...
2022-03-10 15:52:06,490 - INFO - MainProcess: Closing FLIR...
2022-03-10 15:52:06,527 - INFO - Xsens: 00B4875B Number of data packets: 3401 with
0 packets interpolated, (0.0 %)
2022-03-10 15:52:06,840 - INFO - Xsens: 00B4875C Number of data packets: 3401 with
9 packets interpolated, (0.2646 %)
2022-03-10 15:52:07,148 - INFO - Xsens: 00B4875D Number of data packets: 3401 with
4 packets interpolated, (0.1176 %)
2022-03-10 15:52:07,459 - INFO - Xsens: 00B48761 Number of data packets: 3401 with
2 packets interpolated, (0.0588 %)
2022-03-10 15:52:07,770 - INFO - Xsens: 00B48765 Number of data packets: 3401 with
```

```
2 packets interpolated, (0.0588 %)
2022-03-10 15:52:08,086 - INFO - Xsens: 00B48768 Number of data packets: 3401 with
1 packets interpolated, (0.0294 %)
2022-03-10 15:52:08,401 - INFO - Xsens: 00B48769 Number of data packets: 3401 with
2 packets interpolated, (0.0588 %)
2022-03-10 15:52:08,716 - INFO - Xsens: 00B4876C Number of data packets: 3401 with
13 packets interpolated, (0.3822 %)
2022-03-10 15:52:09,026 - INFO - Xsens: 00B4876D Number of data packets: 3401 with
7 packets interpolated, (0.2058 %)
2022-03-10 15:52:09,343 - INFO - Xsens: 00B4876F Number of data packets: 3401 with
13 packets interpolated, (0.3822 %)
2022-03-10 15:52:09,669 - INFO - Xsens: 00B48770 Number of data packets: 3401 with
6 packets interpolated, (0.1764 %)
2022-03-10 15:52:09,981 - INFO - Xsens: 00B48772 Number of data packets: 3401 with
2 packets interpolated, (0.0588 %)
2022-03-10 15:52:10,307 - INFO - Xsens: 00B48773 Number of data packets: 3401 with
11 packets interpolated, (0.3234 %)
2022-03-10 15:52:10,636 - INFO - Xsens: 00B48784 Number of data packets: 3401 with
15 packets interpolated, (0.441 %)
2022-03-10 15:52:10,952 - INFO - Xsens: 00B487B6 Number of data packets: 3401 with
5 packets interpolated, (0.147 %)
2022-03-10 15:52:10,954 - INFO - Xsens:
 Closing XSens log file...
2022-03-10 15:52:11,302 - INFO - Xsens: Closing ports...
2022-03-10 15:52:11,462 - INFO - Xsens: Destroying XsControl object.
2022-03-10 15:52:11,582 - INFO - MainProcess: Closing
2022-03-10 15:52:11,583 - INFO - MainProcess: Successfully destroyed reader
processes

2022-03-10 15:52:11,583 - INFO - MainProcess:
 Acquisition was a success !
```

ANNEXE C APPROBATION ÉTHIQUE DU CHUSJ

CHU Sainte-Justine
Le centre hospitalier universitaire mère-enfant
Université de Montréal

Le 19 janvier 2022

Docteur Philippe Jouvet
CHU Sainte-Justine

Objet	Autorisation de réaliser la recherche
	2022-3505 Création d'une base de données vidéos d'enfants hospitalisés pour l'estimation de pose et le suivi du mouvement 3D Cochercheurs: Lama Seoud

Bonjour,

Il nous fait plaisir de vous autoriser à réaliser la recherche identifiée en titre dans notre établissement et/ou sous ses auspices.

Cette autorisation vous est accordée sur la foi des documents que vous avez déposés auprès de notre établissement afin de compléter l'examen de convenance ainsi que la lettre du CER évaluateur. Si ce CER vous informe pendant le déroulement de cette recherche d'une décision négative portant sur l'acceptabilité éthique de cette recherche, vous devrez considérer que la présente autorisation de réaliser la recherche dans notre établissement est, de ce fait, révoquée à la date que porte l'avis du CER évaluateur.

Notre établissement a reçu une copie de la version finale des documents se rapportant à la recherche, approuvée par le CER évaluateur.

Cette autorisation de réaliser la recherche suppose également que vous vous engagez :

1) à vous conformer aux demandes du CER évaluateur, notamment pour le suivi éthique continu de la recherche;
2) à rendre compte au CER évaluateur et à la signataire de la présente autorisation du déroulement du projet, des actes de votre équipe de recherche, s'il en est une, ainsi que du respect des règles de l'éthique de la recherche;
3) à respecter les moyens relatifs au suivi continu qui ont été fixés par le CER évaluateur;
4) à conserver les dossiers de recherche pendant la période fixée par le CER évaluateur, après la fin du projet, afin de permettre leur éventuelle vérification;
5) à respecter les modalités arrêtées au regard du mécanisme d'identification des sujets de recherche dans notre établissement, à savoir la tenue à jour et la conservation de la liste à jour des participants de recherche recrutés dans notre établissement. Cette liste devra nous être fournie sur demande.

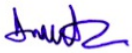
La présente autorisation peut être suspendue ou révoquée par notre établissement en cas de non-respect des conditions établies. Le CER évaluateur en sera alors informé.

Vous consentez également à ce que notre établissement communique aux autorités compétentes des renseignements personnels qui sont nominatifs au sens de la loi en présence d'un cas avéré de manquement à la conduite responsable en recherche de votre part lors de la réalisation de cette recherche.

Je vous invite à entrer en communication avec moi pendant le déroulement de cette recherche dans notre établissement, si besoin est. Vous pouvez aussi contacter notre CER en vous adressant au Bureau de l'éthique de la recherche (ethique.hsj@ssss.gouv.qc.ca).

En terminant, je vous demanderais de toujours mentionner dans votre correspondance au sujet de cette recherche le numéro attribué à votre demande par notre établissement ainsi que le numéro attribué au projet de recherche par le CER évaluateur.

Veuillez accepter mes sincères salutations.



Mariana DUMITRASCU
Coordonnatrice du Bureau de l'éthique de la recherche
CHU Sainte-Justine

Pour Marc Girard, M.D.
Directeur des services professionnels (DAMU)
Personne formellement mandatée au CHU Sainte-Justine pour autoriser la réalisation des projets de rercherche