

**Rapport de stage d'Ingénieur
de 2^{ème} année de Master**

Filière : SIAD

Étiquetage automatique de trajectoires 3D

Présenté par :

Martin BENITO-RODRIGUEZ

Tuteur académique : Lhouari NOURINE Date de la soutenance : 7 septembre 2022

Tuteur Entreprise : E. Bergeret, JM Favreau Durée du stage : 5 mois

Remerciements

Je remercie mes tuteurs entreprises Emmanuel Bergeret et Jean-Marie Favreau pour m'avoir accompagné durant ce stage. Je remercie aussi les doctorants du LPC, notamment Cédric Muhoza qui m'a suivi durant ces mois. Je remercie également les autres stagiaires qui m'ont accompagné : Imane, Samuel et Patricia. Je remercie aussi tout le personnel du LPC, notamment Alexis pour m'avoir guidé dans l'utilisation de Qualisys. Je remercie mon tuteur ISIMA : Lhouari Nourine. Enfin, je remercie Victor Blanchi et Florient Kolly, mes camarades du camp ML4G pour m'avoir aidé à corriger mon article.

Table des matières

Remerciements	2
Table des matières	3
Table des figures	4
Résumé	5
Abstract	5
Lexique	6
Introduction	7
Contexte du stage	8
Présentation de l'entreprise	8
Contexte du projet	9
L'outil Qualisys et la problématique	12
Matériel et méthode	20
Organisation du stage et outils	20
La détection des artefacts	22
La reconstruction des marqueurs	24
L'obligation des 15 marqueurs	26
Méthode d'évaluation du programme	28
Résultats et Discussion	30
Exécution du programme	30
Évaluation de la détection d'artefacts	32
Évaluation de la reconstruction de marqueurs	33
Pistes d'amélioration	37
Conclusion	38
Webographie	

Table des figures

1	Organigramme du LPC	9
2	Les partenaires du projet	10
3	Un bracelet et son capteur	11
4	Le principe du projet	12
5	Caméras Qualisys modèle Miquis M3	13
6	Les marqueurs	13
7	Le laboratoire avec les caméras en évidence	14
8	À gauche : une personne avec des capteurs, à droite : les capteurs vus par le logiciel	15
9	La séquence d'action à effectuer	16
10	Les caméras dans l'interface Qualisys à une frame donnée	17
11	Reconstitution des trajectoires en 3D après une acquisition	18
12	Le diagramme de Gantt prévisionnel estimé	21
13	Le diagramme de Gantt réel	22
14	Un exemple d'artefact	23
15	La matrice spatio-temporelle	25
16	L'algorithme de construction	25
17	Représentation de 4 trajectoires en fonction du temps	27
18	Trajectoires assemblées par l'humain	29
19	Dossier pour lancer le programme build	30
20	Exécution du fichier builder en ligne de commande	31
21	Le dossier après l'exécution	31
22	Les trajectoires labelisées par mon algorithme	32
23	Score de détection d'artefacts	33
24	L'évaluation de mon programme	34
25	Évaluation de plusieurs métriques spatio-temporelles	35
26	Comparaison de la métrique "évolution" avec les autres métriques	36
27	Évaluation de la méthode d'avancée parallèle	37

Résumé

L'objectif du stage est de rendre opérationnel un réseau de neurones profonds pour étiqueter des trajectoires 3D avec des étiquettes issues d'une ontologie de mouvement dédiée. Grace au logiciel de **Motion Capture Qualisys**, les mouvements de marqueurs placés sur une personne en mouvement sont récupérés sous forme d'un fichier c3d. Ce fichier est traité et les mouvements sont étiquetés.

L'analyse des mouvements permettra à des professionnels de santé d'étudier les gestes de leurs patients. D'autres personnes dans le projet travaillent sur l'analyse de mouvements avec d'autres technologies comme l'utilisation de capteurs accéléromètres. Les informations recueillis par les différentes technologies permettront une analyse de mouvements complète et pertinente et permettra aux professionnels de santé de porter des diagnostics conformément aux données.

Le programme a été écrit à l'aide du langage **Python**. Le développement a été réalisé sous Windows 10 avec l'environnement de programmation Jupyter Notebook.

A ce jour, le programme n'est pas encore fonctionnel, le stage s'achève fin septembre, il reste encore un mois pour que le programme soit performant. Il faut améliorer les performances du programme de construction.

Mots-clés : Qualisys, Python, Motion Capture, Etiquettage, Deep Learning, Esanté

Abstract

The objective of the internship is to make operational a deep neural network to label 3D trajectories with labels from a dedicated motion ontology.

Thanks to the **Motion Capture Qualisys** software, the movements of markers placed on a moving person are retrieved as a c3d file. This file is processed and the movements are tagged.

The analysis of the movements will allow health professionals to study the gestures of their patients. Other people in the project are working on the analysis of movements with other technologies such as the use of accelerometers. The information collected by the different technologies will allow a complete and relevant analysis of movements and will allow health professionals to make diagnoses according to the data.

The program was written using the **Python** language. The development has been done under Windows 10 with the programming environment Jupyter Notebook.

To date, the program is not yet functional, the internship ends at the end of September, there is still a month to make the program perform. We need to improve the performance of the the performance of the construction program.

Keywords : Qualisys, Python, Motion Capture, Labelling, Deep Learning, Ehealth

Lexique

Artefact : Dans la Motion Capture, les artefacts ou artifacts sont des erreurs de détection. Des points sont détectés par le système alors qu'ils ne sont pas désirés.

C3d : Format de fichier utilisé en biomécanique, en motion capture et dans l'industrie de l'animation pour lire et analyser des données de mouvements C3d.

Deep Learning : Technique d'apprentissage automatique qui utilise des réseaux de neurones.

E-santé : Ensemble de technologies de communication des informations médicales qui permet aux personnes recevant des soins médicaux de participer à la gestion de leur santé en accédant à des informations médicales fiables.

Marqueur : Les marqueurs de Qualisys sont de petits objets sphériques qui réfléchissent la lumière pour être détectées par les caméras LED Qualisys. Ils permettent d'analyser les mouvements du corps sur lequel on place les marqueurs.

Motion Capture : Technique permettant d'enregistrer les positions et rotations d'objets ou de membres d'êtres vivants.

PyTorch : bibliothèque logicielle Python open source d'apprentissage machine. Elle permet d'effectuer les calculs tensoriels nécessaires notamment pour l'apprentissage profond.

Qualisys : Entreprise spécialisé dans la Motion Capture

Trajectoire : une trajectoire selon Qualisys est le chemin que parcours un marqueur pendant un intervalle de temps.

Introduction

Un des pôles du LPC concerne la physique dans des thèmes liés à la santé ou à l'environnement. Mon stage s'inscrit dans le projet EMOB (E-Mobilité) et concerne la e-santé. Le projet utilise les nouvelles technologies pour avoir des informations sur le suivi des patients dans leurs habitudes de vie et ainsi donner des diagnostics appropriés.

Le projet e-santé, mobilité et big data s'articule autour de 3 acteurs : l'Université Clermont Auvergne, la Simon Fraser University (au Canada), et le CHU de Clermont-Ferrand. Il est aussi soutenu par l'institut Analgesia pour étudier la douleur chronique et l'entreprise Withings sur la création de montres connectées.

Parmi les technologies utilisées, on retrouve des capteurs de mouvements, ceux-ci donnent des informations sur la gestuelle des patients qui peut nous renseigner sur des potentielles douleurs musculo-squeletiques. Ces informations permettront aux spécialistes de donner des diagnostics pertinents. Plusieurs types de capteurs sont utilisés dans le projet, mon stage concerne des capteurs réfléchissant la lumière qui est captée par des caméras.

Un réseau de neurones a déjà été mis en place pour labeliser les données de mouvements. Mon objectif est de mettre à jour ce réseau de neurones et qu'il puisse fonctionner à partir des données directement fournis par les caméras sans interventions intermédiaires. La majorité du travail que j'ai effectué a donc été faite en amont du deep learning pour traiter les données et qu'elles soient exploitable par l'IA. On inclut dans ce travail la détection et suppression des artefacts ainsi que la reconstitution des marqueurs.

Pour présenter ce stage, nous présenterons d'abord l'entreprise et le rôle de mon stage au sein du projet global. Nous verrons par la suite quels outils et quelles méthodes j'ai employé pour résoudre les différents problèmes qui se sont présentés. Enfin, nous verrons comment s'exécute le programme et les résultats des différents algorithmes.

Contexte du stage

Présentation de l'entreprise

L’Université Clermont-Auvergne (UCA) est une université française fondée en 2017 par la réunion de l’Université d’Auvergne et l’Université Blaise-Pascal. Elle est située à Clermont-Ferrand mais possède plusieurs campus en Auvergne.

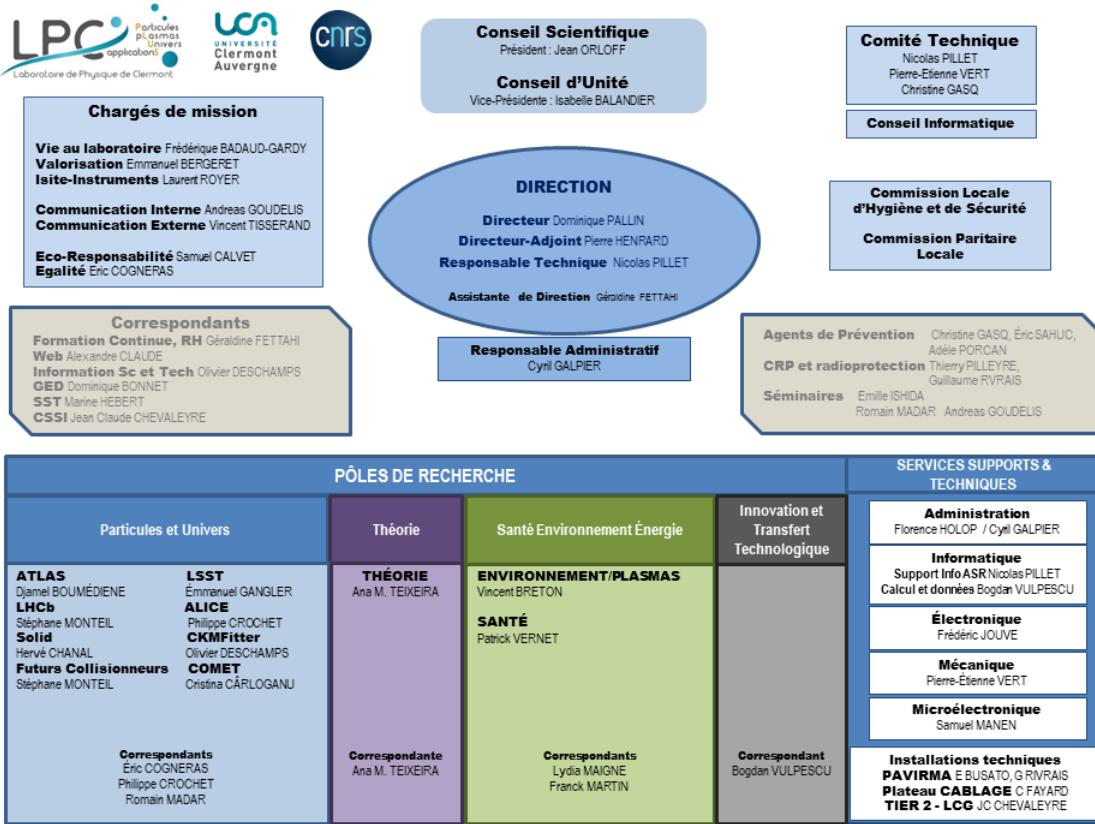
Elle est devenue en 2021 un établissement public expérimental (EPE) et depuis se divise en 6 instituts, eux-mêmes divisés en UFR, instituts et écoles :

- Institut de Droit, Économie et Management (IDEM)
- Lettres, Langues, Sciences Humaines et Sociales (LLSHS)
- Sciences de la Vie, Santé, Agronomie et Environnement (SVSAE)
- Institut des Sciences (IdS)
- Institut Universitaire de Technologie (IUT)
- Institut Clermont Auvergne INP

Au total, en 2022, elle réunit 39 000 étudiants et 1281 enseignants-chercheurs et possède un budget de 300 millions d’euros.

Le service dans lequel le stage a été effectué est le Laboratoire de Physique de Clermont (LPC).

Le LPC est une Unité Mixte de Recherche (UMR) de l’UCA et du CNRS, elle appartient à l’institut de Physique Nucléaire et de Physique des Particules (IN2P3). Depuis 2017, le LPC a été rejoint par 3 équipes : le Laboratoire Arc Electrique et Plasmas Thermiques (LAEPT), Physico-Chimie des Surfaces Nanostructurées (PCSN/C-biosenss) et Réparation du Génome Mitochondrial (RGM).



Avril 2022

<http://clrwvw.in2p3.fr/>

FIGURE 1 – Organigramme du LPC

Le laboratoire exerce des activités de recherche fondamental, allant de l'infiniment petit (les quarks, les noyaux atomiques, leurs interactions...) jusqu'à l'infiniment grand (l'astroparticule et la cosmologie).

Ses activités s'étendent également sur les domaines des plasmas, des sciences du vivant et de l'environnement, réunissant physiciens, biologistes et physico-chimistes.

Le laboratoire exerce aussi des recherches expérimentales à l'aide d'accélérateurs et de télescopes.

Depuis 2018, le LPC a organisé 33 manifestations comme des conférences et des colloques. L'équipe est composée de 157 personnes, dont 47 enseignants-chercheurs, 21 chercheurs, 18 ingénieurs de recherche, 23 ingénieurs d'étude, 7 post-doctorants et chercheurs contractuels et 24 doctorants.

Le LPC est aussi un acteur de l'enseignement supérieur d'Auvergne dans les domaines de la physique, des sciences pour l'ingénieur et de la biologie. Le LPC est impliqué dans des formations du schéma Licence Master Doctorat (LMD), des écoles d'ingénieurs, dans les IUT.

Contexte du projet

À cause du vieillissement de la population, de l'obésité et des douleurs chroniques la mobilité devient un sujet de plus en plus préoccupant pour les pays développés.

Les nouvelles technologies peuvent contribuer à résoudre ces problèmes, c'est le pari pris par le projet e-santé, mobilité et big data créé en 2019 avec de nombreux partenaires.

Partenaires impliqués

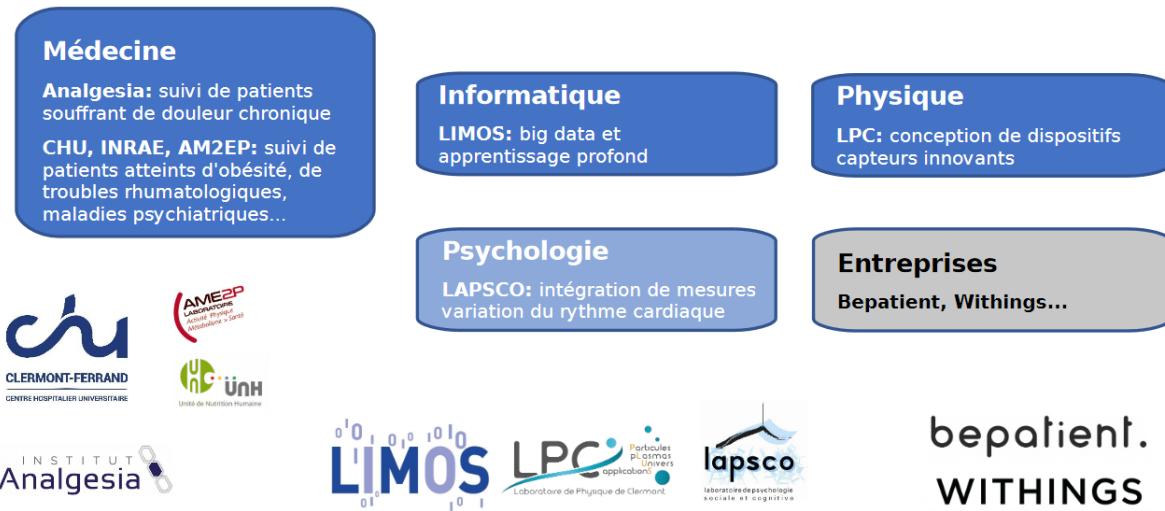


FIGURE 2 – Les partenaires du projet

Parmi ces technologies, on trouve des outils d'évaluation de l'activité des patients, et de leur suivi, comme le projet eDOL développé avec l'institut Analgesia. C'est une application smartphone qui recueille des informations sur les patients. Les données sont analysées avec de l'apprentissage machine. Ainsi, les spécialistes de santé peuvent ainsi donner des diagnostics et réagir en temps réel.

Pour recueillir des données sur la mobilité des patients, on peut les équiper de capteurs de mouvements.

Jean-Marie Favreau développe l'application e-santé Sensor Recorder qui utilise les capteurs embarqués dans les smartphones pour collecter des données de mouvements.

Cédric Muhoza est doctorant au LPC, il développe des capteurs spécifiques à placer à différents endroits du corps (poche, poignet, hanche, genou...). Ces capteurs calculent l'accélération et la vitesse angulaire et grâce au deep learning, infèrent quels mouvements sont effectués. Ils peuvent prendre la forme de montres connectées ou bien de capteurs spécifiques.



FIGURE 3 – Un bracelet et son capteur.

Amane qui est en stage de master à l’UCA se spécialise dans la maîtrise du hardware. Elle va développer des composants qui peuvent optimiser les calculs de deep learning pour perfectionner les capteurs de Cedric.

Le projet s’intéresse aussi à comment l’étiquetage sémantique les activités de la personne suivie. C’est sous cette problématique que s’opère mon stage. Ala Mhalla est un post-doctorant du Limos. Depuis 2019, il utilise le deep learning pour étiqueter les mouvements réalisés par le patient.

Toutes ces données collectées pour résoudre ces problématiques doivent être stockées. C'est notamment le travail de Bastien Doreau qui développe un serveur logiciel pour collecter les données de santé et en rendre l'exploitation aisée.

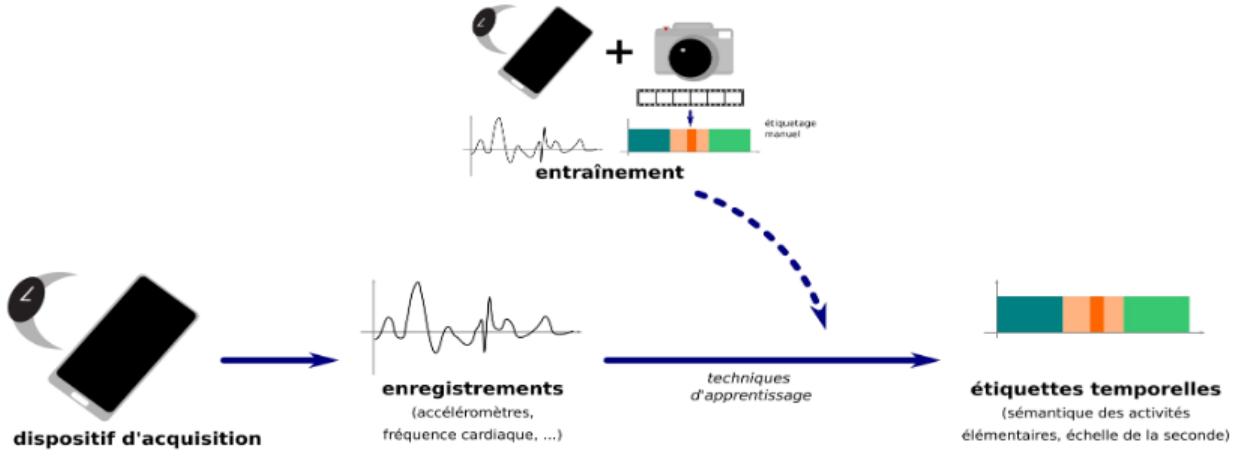


FIGURE 4 – Le principe du projet.

Pour en revenir à mon stage, Ala Mhalla a commencé à écrire un code en PyTorch pour labeliser les mouvements à l'aide de leurs coordonnées dans l'espace et le temps. Pour capturer ces coordonnées, on utilise des caméras qui enregistrent le mouvement de marqueurs disposés sur un humain en mouvement.

Le code n'est pas totalement fonctionnel. Ma mission est de rendre ce code opérationnel et d'automatiser au maximum le passage entre la capture des caméras et l'output du réseau de neurones. Les données récupérées par les caméras ne peuvent pas être directement utilisées comme input pour le deep learning. Je dois traiter ces données afin qu'elles puissent servir d'entrée au réseau de neurones et rendre ce dernier plus performant.

A la fin du stage, je devrais avoir un programme qui prend en entrée un fichier c3d (qui correspond aux données capturées par les caméras) et qui ressort la liste des mouvements effectués.

L'outil Qualisys et la problématique

Qualisys est une entreprise faisant partie des leaders dans le domaine du motion capture optique. L'entreprise crée des caméras et des logiciels pour capturer des mouvements.

La motion capture a plusieurs domaines d'applications, on peut citer la médecine (ce qui nous intéresse ici), le sport (pour que les sportifs puissent analyser et perfectionner leurs mouvements) ou bien l'animation cinématographique ou vidéoludique.

La motion capture se divise en 2 branches : la motion capture marker-based (basée sur les marqueurs) ou markerless (sans marqueurs). La dernière a l'avantage d'être rapide à mettre en place car il n'y a pas besoin d'équipements, mais cela se fait au détriment de la précision. Le projet a choisi l'option marker-based.

Qualisys fournit plusieurs modèles de caméras. On peut trouver les différents modèles sur leur site web [\[1\]](#).



FIGURE 5 – Caméras Qualisys modèle Miquus M3

Les marqueurs sont de petites billes blanches qui réfléchissent la lumière de manière à être détecté par les caméras. Pour capturer les mouvements d'un humain, on lui place des marqueurs à plusieurs endroits du corps.

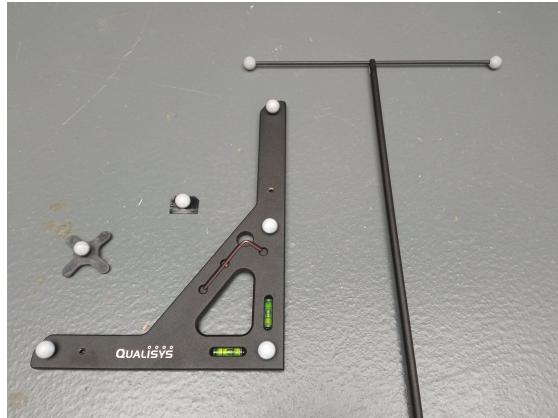


FIGURE 6 – Les marqueurs

On place ensuite les caméras de manière à détecter efficacement les marqueurs. Il faudrait privilégier un endroit lumineux pour que la lumière se réfléchisse, mais nous verrons plus tard avec les artefacts que la luminosité peut aussi être un problème. Plus on a de caméras, plus la détection de mouvements est précise.

Parmi les caméras, on peut aussi mettre une "Miquus video" qui ne cherche pas à détecter les marqueurs, mais qui se contente de filmer de manière "classique". Cela peut être utile pour voir les mouvements du personnage de manière interprétable par l'humain avec les mouvements des marqueurs à côté.

Pour nos séances de captures, nous sommes munis de 6 caméras Miquus M3 et d'une caméra Miquus Video. Sur la personne en mouvement, on place 15 marqueurs à des endroits clés : la tête, le torse, l'épaule gauche, l'épaule droite, le coude gauche, le coude droit, la main gauche, la main droite, le bassin, la hanche gauche, la hanche droite, le genou gauche, le genou droit, le pied gauche et le pied droit.

Les caméras enregistrent à 100 images par seconde.



FIGURE 7 – Le laboratoire avec les caméras en évidence.

En jaune sont entourées les caméras, les 2 caméras au fond à droite sont absentes. La Miquis Video est la caméra en haut à gauche.

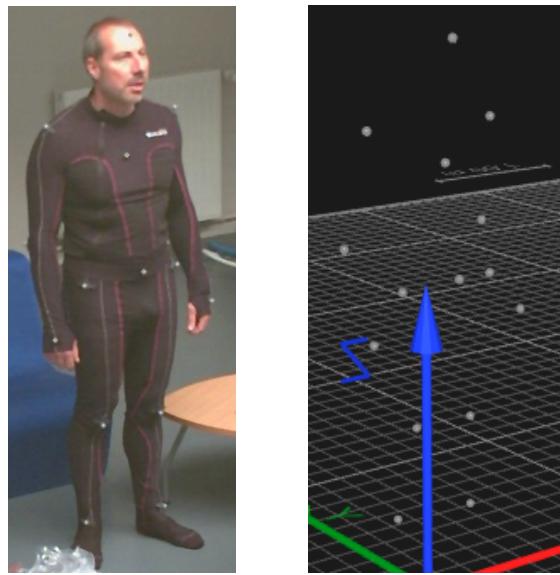


FIGURE 8 – À gauche : une personne avec des capteurs, à droite : les capteurs vus par le logiciel

Notre but étant d'analyser les mouvements quotidiens de personnes, une liste d'actions a été faite. La personne doit effectuer ces différentes actions lors d'une session d'enregistrement.

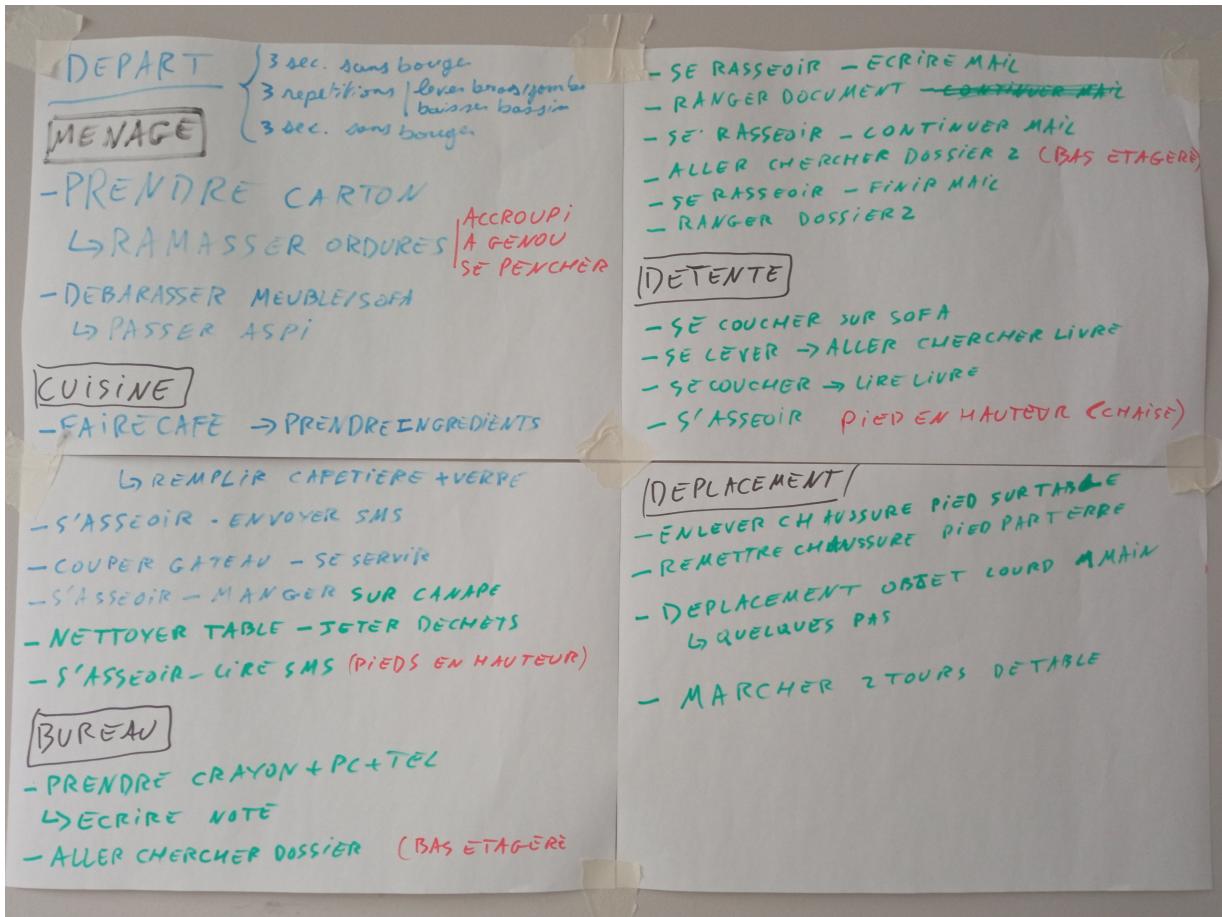


FIGURE 9 – La séquence d'action à effectuer

Pour effectuer tous ces mouvements, il faut environ 10 minutes.

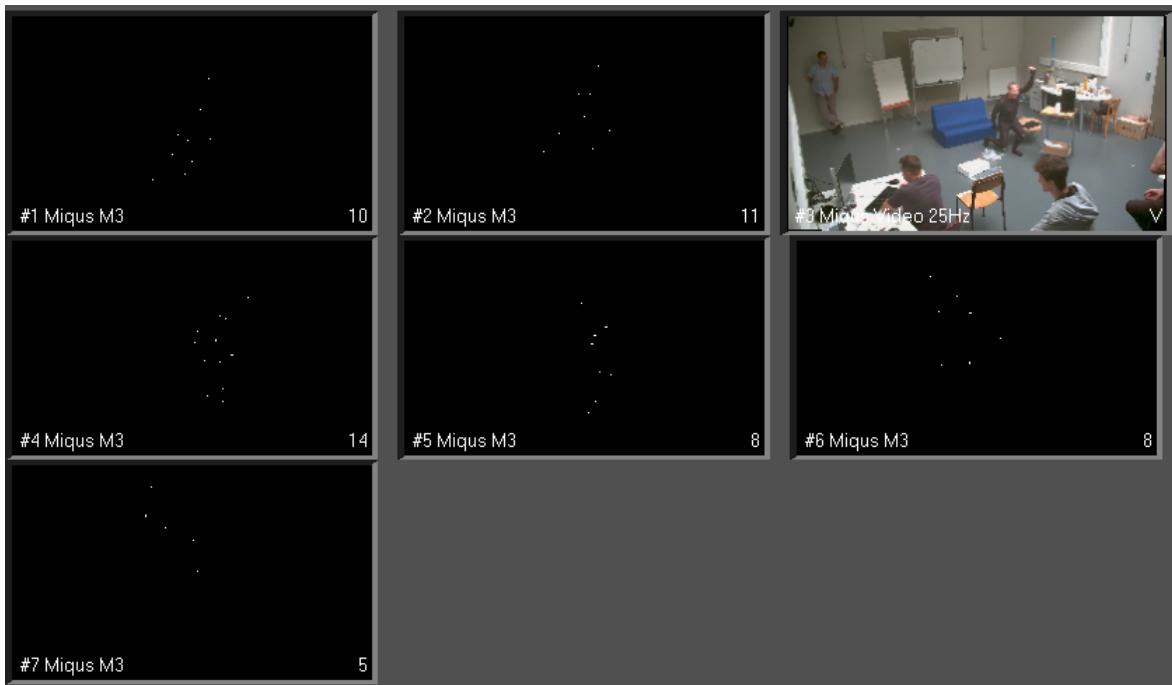


FIGURE 10 – Les caméras dans l’interface Qualisys à une frame donnée

On voit sur la figure 10 l’interface de Qualisys pendant un enregistrement. Les marqueurs sont les billes blanches détectées par chacune des 6 caméras. On voit aussi ce que voit la septième caméra Miquus Video. Pour chaque caméra Miquus M3, il y a un nombre en bas à droite, il correspond au nombre de marqueurs détectés par la caméra à la frame actuelle. On voit qu’aucune des caméras ne détecte les 15 marqueurs, ce qui est normal car les marqueurs sont placés tout autour du corps. Des marqueurs sont alors cachés par l’humain ou un objet. Lorsque les marqueurs seront reconstitués en 3D par les caméras, on pourra voir les 15 marqueurs simultanément. Le problème est que malgré la reconstitution avec les 6 caméras, certains marqueurs ne seront pas détectés par suffisamment de caméras pour être positionnés avec précision. Donc même après la reconstruction, certains marqueurs ne seront pas visibles pendant un intervalle de frames. Il faudra attendre que le marqueur soit à nouveau visible par suffisamment de caméras pour le positionner en 3D.

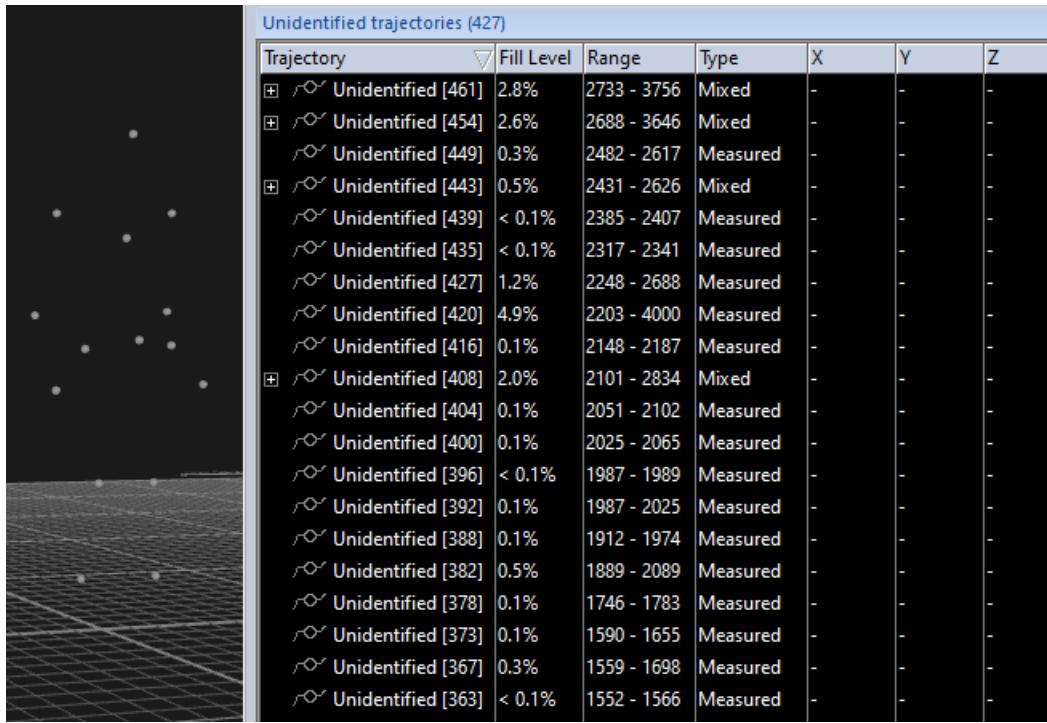


FIGURE 11 – Reconstitution des trajectoires en 3D après une acquisition

On peut voir sur la figure 11 qu'il y a 427 trajectoires. Une trajectoire est le chemin d'un marqueur. Dans l'idéal, on ne devrait trouver que 15 trajectoires, une trajectoire par marqueur. Quand un marqueur est détecté par les caméras, il peut devenir invisible comme expliqué précédemment, mais lorsqu'il est à nouveau détecté, les caméras ne peuvent pas savoir que c'est le même marqueur que celui qui a disparu. Dans ce cas là, le marqueur sera représenté par une deuxième trajectoire sans savoir que ces 2 trajectoires correspondent au même marqueur. Au lieu d'avoir 15 trajectoires, nous en avons plusieurs centaines. Un des objectifs du stage est de recomposer le mouvement des marqueurs, c'est à dire ré-assembler les trajectoires pour suivre toute la séquence du marqueur.

Une autre cause de l'excès de trajectoires et la présence d'artefacts. Ce sont des objets qui sont détectés par les caméras comme des marqueurs alors qu'ils ne sont pas des marqueurs. La principale source d'artefacts est la réflexion de la lumière sur des objets réfléchissants, comme du papier blanc ou des objets métalliques. On peut limiter l'apparition d'artefacts pendant l'enregistrement en faisant attention aux objets présents ou en mettant du scotch sur les zones métalliques. Mais malgré ces précautions, des artefacts persistent et doivent donc être supprimés.

Traiter un fichier avec 427 trajectoires est beaucoup trop. La plupart des logiciels ou des bibliothèques traitant les fichiers C3d ne peuvent que rarement excéder les 255 trajectoires. Pour alléger cette quantité, nous garderons uniquement les trajectoires qui vivent pendant au moins 1% du temps total. Nous aurons alors des "gaps" pendant nos trajectoires, mais les mouvements plus longs et qui contiennent donc plus d'informations seront conservés.

Pour mettre en place l'étiquetage automatique, il faut créer et entraîner un réseau de neurones. Les données d'entraînements sont les coordonnées de trajectoires sous forme de fichier csv. Les features à inférer doivent être ajoutées à la main au fichier csv. Pour cela il faut relever les actions qui nous intéressent et sur quelles frames elles sont présentes à l'aide des vidéos. Grâce à ce fichier csv, le deep learning s'entraînera à inférer un type de mouvements à partir des coordonnées des trajectoires sur un intervalle de frames.

Il y a 20 actions à inférer : Assis_Debout, Assis_Couche, Couche_Assis, Debout_Assis, Debout_Agenou, Agenou_Debout, Debout_Penche, Penche_Debout, Autre_Transition, Marcher, Monter_Escaliers, Descendre_Escaliers, Lever_Bras, Lever_2_Bras, Baisser_Bras, Baisser_2_Bras, Autre_Mouvement_Bras, Lever_Jambe, Baisser_Jambe, et Autre_Mouvement_Jambe.

Le programme de deep learning a commencé à être écrit par Ala en Python et PyTorch. Mais les fichiers csv qui doivent servir à l'entraînement n'ont pas un format conforme aux conventions qui ont été posées. De plus il ne détecte que 6 types de mouvements. Il faudrait que les nouveaux fichiers csv d'entraînements puissent entraîner le réseau de neurones et que ce dernier infère les 20 types de mouvements.

En faisant quelques recherches sur un moteur de recherche avec les mots clés "motion capture movements labelling", je me rend compte que l'étiquetage de mouvements à partir de motion capture est un domaine très étudié, avec de nombreux articles scientifiques qui proposent des résultats prometteurs. Mais ce qui pousse à explorer une approche spécifique, c'est le fait que la sémantique que nous cherchons à reconnaître n'est pas celle de la littérature. Plus fine, orientée sur les propriétés musculo-squelettiques. De plus, on a choisi de traiter des données marqueurs passifs, donc plutôt à partir des coordonnées xyz des parties du squelette qu'à partir de vidéos rgb.

Matériel et méthode

Organisation du stage et outils

J'ai été encadré notamment par Emmanuel Bergeret qui est mon tuteur de stage et son doctorant Cédric Muhoza. Ils ont organisé environ une fois par semaine une réunion avec moi. Durant ces réunions, je devais leur présenter mes avancés et ils me guidaient sur les nouvelles pistes à explorer. Le projet e-santé a aussi un serveur Mattermost pour échanger, j'ai quelques fois utilisé ce moyen pour poser des questions ou demander des ressources dont j'avais besoin, comme le code de Ala. Le projet utilise aussi un drive pour partager les ressources.

J'ai travaillé en présentiel et en distanciel. Le stage commençait mi-avril en distanciel. De début mai à fin juillet, je travaillais en présentiel à l'IUT de Montluçon, puis en août et en septembre je revenais travailler en distanciel.

Mes outils en présentiel et distanciel étaient quasiment identiques, la différence notable est qu'à l'IUT j'avais un ordinateur fixe de l'IUT à disposition. Tandis qu'en distanciel, je travaillais sur mon ordinateur portable personnel.

Les 2 machines sont sous Windows 10.

Pour ce stage, j'ai du manipuler les fichiers c3d, j'ai cherché des bibliothèques pour exploiter ces fichiers. Elles étaient presque toutes manipulables à partir de Python. J'ai téléchargé les bibliothèques que je trouvais les plus intéressantes : PyC3d et EzC3d. J'ai passé un certains temps à les tester. PyC3d contient plein de fonctions pour manipuler les données, il suffit d'appeler les fonctions et le fichier se modifie en conséquence. EzC3d a des fonctions de lectures et d'écritures mais pas de fonction pour faire de "modifications générales". Ce que j'entends par là, c'est qu'avec PyC3d, il y a une fonction pour ajouter un marqueur, il suffit de l'appeler et les modifications s'effectuent. Avec EzC3d, il faut modifier à la main tous les emplacements du fichier pour ajouter un marqueur. Évidemment, si il y a une incohérence dans le fichier, le fichier devient inutilisable. PyC3d permet d'éviter ce problèmes grâce à toutes ses fonctions qui modifient tous le fichier. J'ai donc utilisé cette bibliothèque, même si elle n'est pas parfaite comme nous le verrons plus loin.

Pour utiliser Python, j'ai opté pour Jupyter Notebook avec l'environnement Conda, car j'avais déjà utilisé cette technologie par le passé. Quand je suis arrivé en présentiel, j'ai installé d'autres outils pour pouvoir manipuler Python sous différents angles et voir avec lequel

j'étais le plus à l'aise. J'ai donc téléchargé PyCharm et Visual Studio Code. Je n'ai pas du tout apprécié ces technologies. L'énorme avantage de Jupyter est que grâce à son système de cellules, on peut exécuter son code partie par partie. Si on veut juste tester quelques lignes, on peut créer une cellule avec ces lignes et exécuter la cellule pour étudier son comportement. Dans les autres API, nous sommes obligés d'exécuter le fichier entier, hors quand l'exécution de certaines fonctions sont longues, il est plus économique de les lancer une seule fois. Et c'est le cas avec mon problème : la lecture des fichiers c3d prend environ 30 secondes. J'ai donc continué à travailler sur Jupyter Notebook.

Enfin, pour le Deep Learning, je maîtrisais déjà appris un peu TensorFlow et je ne connaissait que vaguement PyTorch. Je pensais que le réseau de neurones était écrit avec TensorFlow, mais il était écrit en PyTorch. En parallèle de la lecture du code de Ala, je lisais la documentation PyTorch pour comprendre comment fonctionnait le programme.

Le stage n'avait pas de diagramme de Gantt prévisionnel, j'en ai donc créé un moi-même en estimant le temps de chaque tâche à priori.

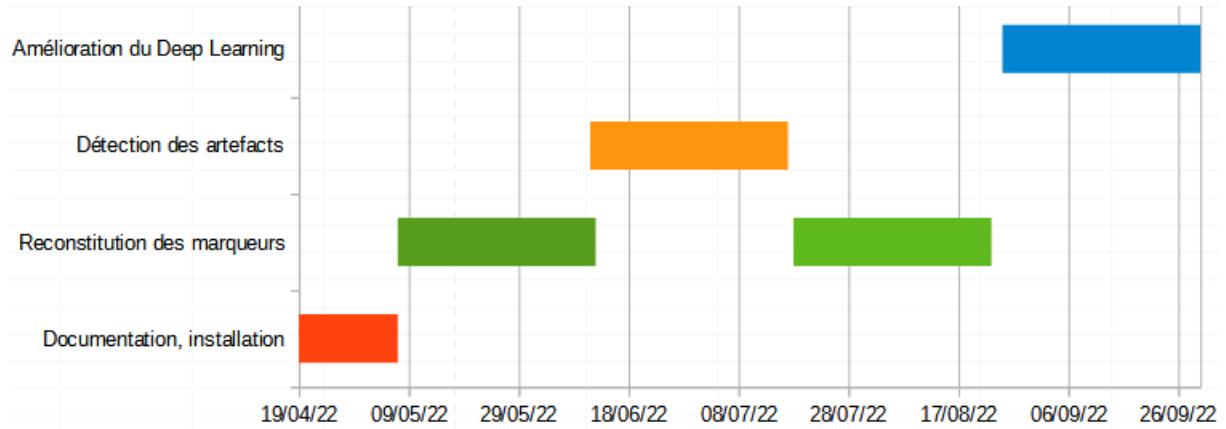


FIGURE 12 – Le diagramme de Gantt prévisionnel estimé

Le stage fini en fin septembre, le diagramme de Gantt prévisionnel va donc plus loin que le diagramme de Gantt réel.

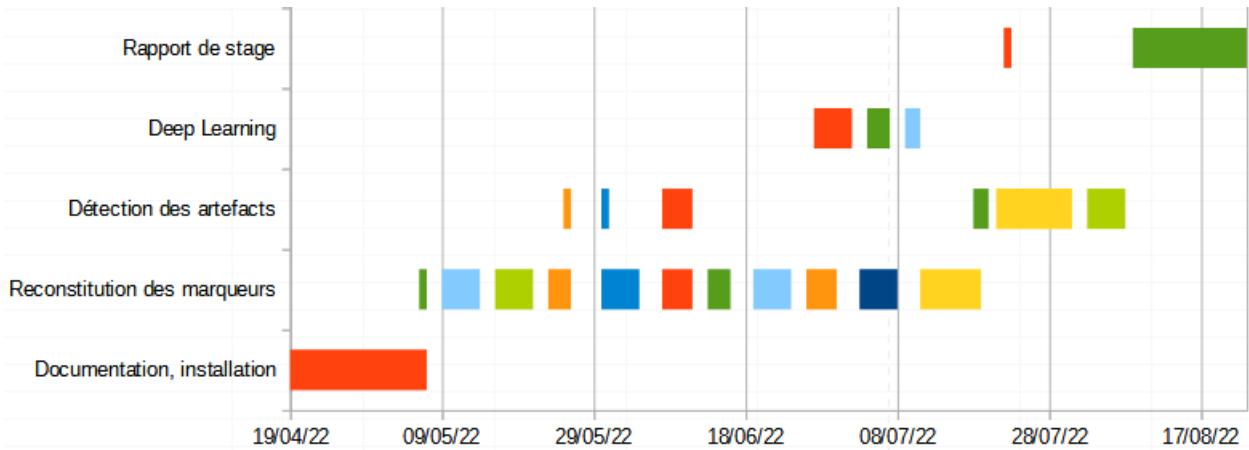


FIGURE 13 – Le diagramme de Gantt réel

Nous voyons que les proportions sont à peu près identiques (même si la comparaison avec le diagramme estimé n'a pas vraiment de sens).

Sur le deep learning, j'ai surtout passé du temps à comprendre le code et à faire des modifications mineurs, je n'en parlerai donc presque pas dans le rapport.

La détection des artefacts

Pour rappel, les artefacts sont des éléments détectés par les caméras comme étant des marqueurs alors que ce n'est pas le cas. Il faut dans un premier temps les détecter et les supprimer.

Quand on observe les artefacts, on se rend compte qu'ils sont pour la grande majorité immobiles. Une conséquence de cela est que lorsque la personne se déplace dans la pièce, l'artefact peut se situer à plusieurs mètres des vrais marqueurs. J'ai donc ici 2 critères pour discriminer les artefacts : la vitesse du point et sa distance par rapport aux autres points présents. On peut encore en ajouter un troisième : si à une frame donnée, on a plus de 15 marqueurs, c'est que nous avons au moins un artefact. On pourrait aussi avoir des artefacts avec moins de marqueurs présents. Si à une frame donnée, nous avons 3 "vrais" marqueurs qui ne sont pas détectés et que nous avons 1 artefact, nous aurons $15-3+1=13$ points présents.

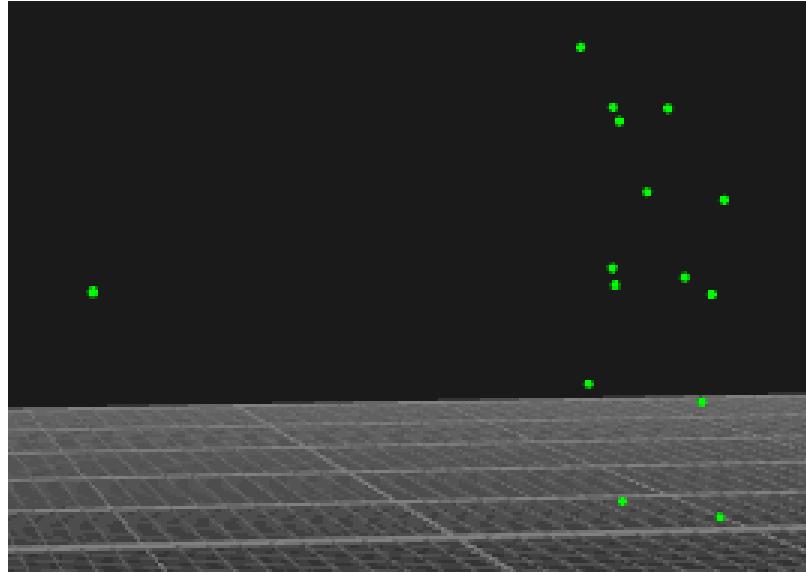


FIGURE 14 – Un exemple d’artefact

On peut raisonnablement penser que le point à gauche est un artefact étant donné sa distance avec les autres points.

Au total, j’ai 5 fonctions pour détecter les artefacts. Il faudra déterminer plusieurs paramètres pour que la détection soit optimisée : à partir de quel distance on considère qu’un point est trop loin des autres pour être un artefact ? A partir de quelle vitesse un point est-il considéré comme artefact ? Ces paramètres seront trouvés en faisant des expériences. On nommera ici ces paramètres $p_1, p_2, p_3\dots$

La première fonction cherche si un marqueur est loin des autres. Si on calcule les distances entre marqueurs à chaque frame, le temps de calcul devient vraiment trop grand. Il faut donc calculer les distances toute les p_1 frames, p_1 étant un paramètre à déterminer en faisant des expériences. Si à une frame étudié, un marqueur est à une distance supérieur à p_2 de tous les autres, on le considère comme artefact.

Le problème avec cette méthode c’est que certains artefacts ont une vie très courte et ne peuvent pas être vus si on regarde toute les p_1 frames. La seconde méthode calcule la distance entre les marqueurs à l’apparition et à la disparition de chaque trajectoires. Ainsi, si des artefacts ont une vie très courte, ils pourront quand même être supprimés si ils sont suffisamment loin des autres.

Les 2 fonctions suivantes se basent sur la vitesse du marqueur. La première calcule la vitesse moyenne des trajectoires. Comme précédemment, si on calcule nos vitesse à chaque frame, le temps de calcul devient déraisonnable. Il faut donc regarder toute les p_3 frames la position tridimensionnel du point et calculer la distance entre les positions à chaque pas p_3 . Si la vitesse moyenne dépasse un certain seuil p_4 , la trajectoire sera considéré comme un artefact.

La quatrième méthode calcule la vitesse toute les p5 frames, si à une seule reprise la vitesse excède un seuil p6, la trajectoire est considéré comme artefacts.

Enfin, la dernière fonction crée une liste vide. A chaque apparition de trajectoire, la fonction compte le nombre de points présents, si il y en a plus de 15, toutes les trajectoires présentes sont des artefacts potentiels et sont rajoutées à la liste. On calcule ensuite la vitesse moyenne de chaque trajectoire, la trajectoire avec la vitesse la plus faible sera considéré comme un artefact. On continue ainsi jusqu'à ce qu'il n'y ai plus de frames avec plus de 15 marqueurs.

La reconstruction des marqueurs

Une fois que les artefacts ont été supprimés, il faut assembler les trajectoires.

La question est la suivante : j'ai une trajectoire A qui se termine à un instant et j'ai un ensemble de trajectoires $B=\{B_1, B_2, \dots, B_n\}$ qui apparaissent après la fin de A. Comment savoir quelle trajectoire de B est la suite de A ? Pour assembler les trajectoires, je me suis basé sur 2 indices : la distance temporelle et la distance spatiale.

La distance temporelle est le nombre de frames entre la fin de A et le début des trajectoires de B. Plus cette distance est faible, plus il y a de chance qu'une trajectoire B_i soit successeur de A.

La distance spatiale est la distance entre la position de A à sa dernière frame et la position des trajectoires de B à sa première frame. Plus la distance entre A et une trajectoire B_i est faible, plus il y a de chance que B_i soit successeur de A.

Pour estimer la probabilité que B_i soit successeur de A, on calcul le produit de la distance spatiale et la distance temporelle pour obtenir une distance spatio-temporelle. Ainsi on a une distance spatio-temporelle pour chaque élément de B. On obtient les distances $\{A \rightarrow B_1, A \rightarrow B_2, \dots, A \rightarrow B_n\}$.

Nous plaçons ensuite ces distances dans une matrice que j'appelle la matrice spatio-temporelle.

	New 0023	New 0024	New 0025	New 0026	New 0027
New 0000	NaN	NaN	NaN	NaN	NaN
New 0001	5.129504e+06	6.915226e+06	6.291790e+06	6.131266e+06	7.877085e+06
New 0002	NaN	NaN	NaN	NaN	NaN
New 0003	NaN	NaN	NaN	NaN	NaN
New 0004	NaN	1.960742e+04	2.191753e+05	2.224343e+05	2.857608e+05
New 0005	5.819792e+06	7.421492e+06	6.989954e+06	6.615754e+06	8.588806e+06
New 0006	NaN	NaN	NaN	NaN	NaN
New 0007	4.717878e+06	6.257699e+06	5.818230e+06	5.420407e+06	7.187558e+06
New 0008	1.794813e+06	2.561278e+06	2.820518e+06	2.325567e+06	3.446977e+06
New 0009	1.691100e+06	2.251356e+06	2.773906e+06	2.166863e+06	3.173274e+06

FIGURE 15 – La matrice spatio-temporelle

Les lignes et les colonnes de cette matrice correspondent aux noms des trajectoires. La cellule à l'intersection de la ligne i et de la colonne j correspond à la distance spatio-temporelle entre la trajectoire i et j. La majorité des cellules de la matrice ont une valeur nulle, il y a 2 causes possibles. Soit i et j sont présents simultanément sur au moins une frame et donc ne peuvent pas se suivre ou alors j existe avant i et donc calculer la distance i->j n'a pas de sens, on calculera plutôt la distance j->i.

Une fois cette matrice remplie, on va pouvoir lancer l'algorithme de connexion. Ce que j'appelle une connexion dans ce contexte est une liaison entre une trajectoire i et j. Cela signifie que j est la suite de i. L'algorithme de construction est le suivant :

```

dist.max(1).max()+1
while (dist.min(1).min()<d_max):
    min=dist.stack().idxmin()
    connexion(min[0], min[1])
    dist=dist.drop(min[0])
    dist=dist.drop(min[1], axis=1)

```

FIGURE 16 – L'algorithme de construction

Tout d'abord, on cherche la distance maximum dans la matrice, tant que la distance minimum est inférieure, on relance la boucle. Elle s'arrêtera alors quand la matrice sera vide.

On cherche ensuite la cellule avec la plus petite valeur et on connecte la trajectoire de la ligne avec la trajectoire de la colonne.

On supprime la ligne et la colonne de la cellule. Étant donné qu'on a trouvé un successeur à la trajectoire de la ligne, nous n'avons plus d'intérêt à la conserver. On cherche maintenant la nouvelle plus petite cellule pour faire une connexion, on supprime la ligne et la colonne de la cellule et on continue ainsi. La matrice se vide progressivement et quand toutes les valeurs seront des NaN ou qu'il n'y aura plus de lignes et colonnes, la boucle s'arrêtera.

Dans l'idéal, on devrait avoir à la fin 15 chaînes de connexions, ce que j'appelle chaîne de connexion ici est une liste de trajectoires que mon algorithme détecte comme des suites, par exemple [A->B->C->D]. Chaque chaîne de connexions devra représenter un marqueur.

Il pourra y avoir des erreurs, si on reprend l'exemple ci-dessus : A et B peuvent être des trajectoires de la main gauche, donc faire une connexion entre elles est une bonne chose. Mais peut-être que C correspond au bassin. Il se peut que les marqueurs du bassin et de la main gauche aient disparus au même moment, et quand la personne a placé sa main gauche sur son bassin le capteur du bassin soit réapparu mais pas celui de la main. A partir de cette instant il y aura une confusion entre la main gauche et le bassin. Et ces confusions peuvent s'accumuler au fil des trajectoires.

Quand les données seront envoyées au réseau de neurones, ce dernier aura besoin de connaître quels membres correspondent aux marqueurs. J'ai mis en place un algorithme qui donne un label aux premières trajectoires. Pour faire ça, j'ai regardé les positions des marqueurs à la première frame sur les 10 fichiers pour trouver des régularités. Par exemple : la tête est toujours le marqueur le plus haut, les pieds les plus bas, etc... Grâce à ce système le réseau de neurone saura à quel membre correspondent les différentes trajectoires.

L'obligation des 15 marqueurs

Pour donner des données à notre réseau de neurones, il a besoin de 15 marqueurs. Or, l'exécution de l'algorithme décrit précédemment peut faire en sorte qu'on a plus que 15 chaînes de connexions. Si c'est le cas, nos résultats ne sont pas optimaux, mais surtout : que va-t-on donner à notre réseau de neurones s'il y a 18 marqueurs eu lieu de 15 ? Il faut absolument avoir 15 trajectoires à la fin.

Voici une explication de la source du surplus de chaînes de connexions.

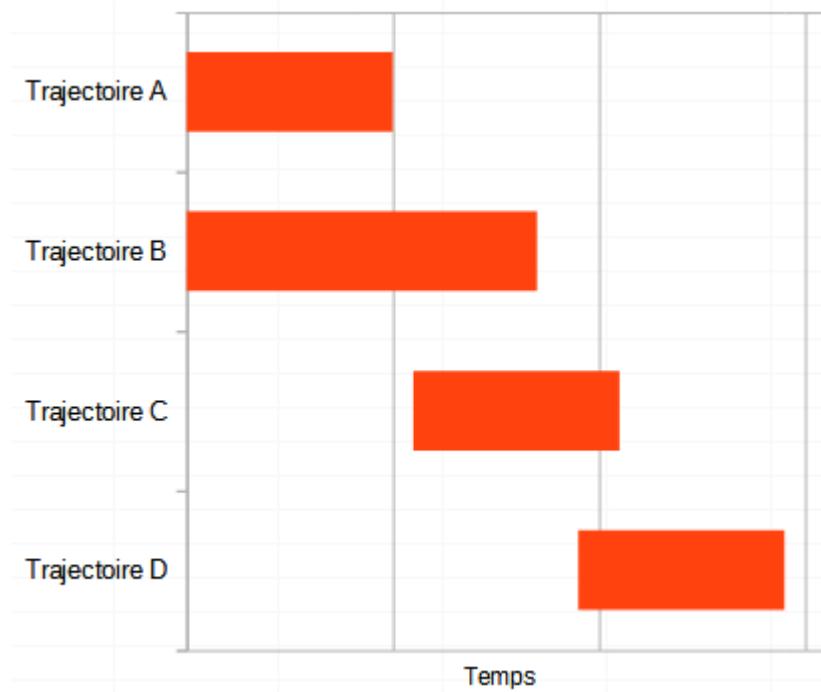


FIGURE 17 – Représentation de 4 trajectoires en fonction du temps

Pour comprendre la figure 17, chaque barre correspond à l'existence d'une trajectoire. Les trajectoires A et B apparaissent en même temps, mais A disparaît avant B.

Sur l'exemple ci-dessus, admettons que les vraies connexions sont A->C et B->D, il y a donc 2 marqueurs. Mais imaginons que mon programme fasse la connexion A->D. Une fois cette connexion faite, les trajectoires B et C ne pourront plus être casées car elles se superposent entre elles et avec D. Mon programme trouvera alors 3 trajectoires à la fin : A->D, B, C, alors qu'il n'y en a que 2 en réalité.

J'ai alors essayé de forcé le programme à avoir 15 marqueurs. Dans ce nouvel algorithme, je commence avec les 15 premières trajectoires et l'objectif va être de reconstruire la vie des marqueurs en parallèles. Chaque vie de marqueur peut-être représentée par une liste de trajectoire, on a donc 15 listes avec au début une trajectoire par liste (ce sont les trajectoires présents à la première frame).

Pour chaque dernier élément de liste, on cherche la trajectoire qui n'est pas déjà dans une liste qui a la distance spatio-temporelle la plus faible avec le dernier éléments. Pour chaque liste, on retient une trajectoire proche et la distance avec cette trajectoire. Ensuite, on cherche la plus petite distance et on connecte la trajectoire associée avec le dernier élément de la liste.

$$\begin{aligned}
 L_a &| d(a, T), T_a \\
 L_b &| d(b, T), T_b \\
 L_c &| d(c, T), T_c \\
 L_d &| d(d, T), T_d
 \end{aligned} \tag{1}$$

Ici L_a, L_b, L_c, L_d sont des listes qui finissent respectivement par les trajectoires a, b, c, d.

$d(a, T)$ est la distance spatio-temporelle la plus petite entre a et toutes les trajectoires qui ne sont pas déjà dans une liste. Et T_a est la trajectoire la plus proche spatio-temporellement de a qui n'appartient à aucune liste, donc pour tout cas $i : L_i \mid d(i, T), T_i$

On cherche ensuite parmi les distances présentes, la plus petite distance $d(i, T)$ et on ajoute T_i à la liste L_i . Ajouter cet élément à la liste revient à connecter i et T_i . Il faut maintenant mettre à jour le système. Si dans l'exemple précédent c'est la distance $d(a, T)$ qui est la plus faible, après la connexion, le système devient :

$$\begin{aligned} &L_{T_a} \mid d(T_a, T), T_{T_a} \\ &L_b \mid d(b, T), T_b \\ &L_c \mid d(c, T), T_c \\ &L_d \mid d(d, T), T_d \end{aligned} \tag{2}$$

Et on avance ainsi en remplissant progressivement les listes et en faisant les connexions. A la fin, on aura 15 listes de trajectoires, et donc 15 trajectoires à la fin de notre programme.

Méthode d'évaluation du programme

Pour évaluer mes connexions, je dispose des 10 fichiers c3d. Mais je dispose aussi d'une autre version de ces fichiers dont les trajectoires ont été assemblées par un humain. En comparant ce fichier et mes résultats, je peux évaluer les performances de mon programme.

Labeled trajectories (15)						
Trajectory	Fill Level	Range	Type	X	Y	Z
+ G_pied	97.3%	1 - 62576	Measured	769.6	126.9	77.7
+ tete	99.6%	1 - 62576	Measured	538.0	-58.7	1717.8
+ torse	91.0%	1 - 62576	Measured	587.7	31.8	1465.6
- G_epaule	99.1%	1 - 62576	Measured	844.3	111.9	1437.8
Part 1	83.7%	1 - 52385	Measured	844.3	111.9	1437.8
Part 2	15.4%	52933 - 62...	Measured	-	-	-
- D_epaule	98.9%	1 - 62576	Measured	359.5	146.0	1472.4
Part 1	19.4%	1 - 12167	Measured	359.5	146.0	1472.4
Part 2	64.0%	12368 - 52...	Measured	-	-	-
Part 3	15.4%	52903 - 62...	Measured	-	-	-
+ G_coude	99.5%	1 - 62576	Measured	853.7	111.0	1146.7
+ G_main	97.7%	1 - 62576	Measured	891.3	-21.1	840.1
+ G_hanche	88.0%	1 - 62576	Measured	745.0	-2.7	921.1
+ G_genou	97.4%	1 - 62576	Measured	742.7	77.7	542.5
+ D_genou	99.2%	1 - 62576	Measured	406.9	131.4	558.4
+ D_pied	77.8%	1 - 62371	Measured	409.7	196.6	95.1
+ D_hanche	98.7%	1 - 62576	Measured	391.2	45.3	935.0
+ D_coude	99.7%	1 - 62576	Measured	321.2	146.3	1156.5
+ D_main	97.7%	1 - 62576	Measured	286.4	16.0	846.5
+ bassin	73.4%	1 - 62576	Measured	579.7	-74.1	1095.4

FIGURE 18 – Trajectoires assemblées par l’humain

On peut voir qu’il y a bien 15 trajectoires comme attendu. Chacune des 15 trajectoires n’est qu’une suite de trajectoires, on le voit sur l’image avec G_epaule et D_epaule.

Pour chaque connexion que mon programme fait, je regarde dans le fichier corrigé si la même connexion existe. Si elle existe, c’est une réussite, sinon, c’est une erreur. L’objectif est donc de minimiser le nombre d’erreurs et de maximiser les réussites.

Résultats et Discussion

Exécution du programme

Pour construire le fichier reconstruit avec une ligne de commande, j'ai besoin d'un dossier comme ceci :

Nom	Type	Taille
builder	Fichier PY	12 Ko
Documentation_builder	Document texte	2 Ko
Measurement05	Fichier C3D	127 177 Ko
Measurement10	Fichier C3D	133 693 Ko
template	Fichier C3D	981 Ko

FIGURE 19 – Dossier pour lancer le programme build

Il faut tout d'abord le fichier builder.py qui est le programme lui-même. La documentation builder n'est pas nécessaire, mais je l'ai placé ici pour que l'utilisateur y trouve toutes les informations nécessaires pour lancer le programme. Il faut aussi les fichiers c3d qu'on veut builder et un fichier c3d spécial : le "template".

Ce fichier est un fichier c3d que j'ai créé, il ne contient qu'une seule trajectoire. C'est dans ce fichier qu'on va placer nos trajectoires connectées. L'existence de ce fichier tient à une chose : l'impossibilité de supprimer des trajectoires avec PyC3d. Cette bibliothèque permet de lire les données des trajectoires, de les modifier, de renommer les trajectoires, mais la suppression est impossible. Mon plan à la base était de modifier les trajectoires en fonction des connexions et de supprimer les autres. Par exemple, si j'ai la connexion A->B->C, je modifie A pour étendre son existence avec les données de B et C, puis je supprime B et C.

Étant donné que cette manipulation est impossible, builder.py fait une copie du fichier template, le renomme et écrit nos trajectoires à l'intérieur. Il faut au moins une trajectoire pour pouvoir écrire dans le fichier : template contient alors une seule trajectoire nommée "à supprimer". Après que nos connexions soient écrites, il faut supprimer la trajectoire "à supprimer" à la main depuis l'interface Qualisys.

```
python builder.py Measurement05.c3d Measurement10.c3d
```

FIGURE 20 – Exécution du fichier builder en ligne de commande

Le programme prend comme argument les fichiers c3d à traiter, ici : Measurement05.c3d et Measurement10.c3d.

Nom	Type	Taille
Abstract	Document texte	4 Ko
builder	Fichier PY	12 Ko
Documentation_builder	Document texte	2 Ko
Measurement05	Fichier C3D	127 177 Ko
Measurement05_build	Fichier C3D	49 409 Ko
Measurement10	Fichier C3D	133 693 Ko
Measurement10_build	Fichier C3D	59 737 Ko
template	Fichier C3D	981 Ko

FIGURE 21 – Le dossier après l'exécution

Après l'exécution, un fichier texte "Abstract" a aussi été créé, ce fichier contient des informations utiles sur ce qui a été fait. On y trouve les artefacts qui ont été supprimés et les connexions qui ont été faites.

Sur le nommage des marqueurs, celui-ci fonctionne bien. Voilà ce que j'obtiens après l'exécution de mon programme :

Labeled trajectories (18)							
Trajectory	Fill Level	Range	Type	X	Y	Z	
✓ torse	100.0%	1 - 62576	Measured	589.1	31.4	1466.5	
✓ tête	100.0%	1 - 62576	Measured	542.2	-57.2	1719.6	
✓ bassin	100.0%	1 - 62576	Measured	580.1	-75.4	1096.0	
✓ New 0110	100.0%	1 - 62576	Measured	0.0	0.0	0.0	
✓ New 0103	100.0%	1 - 62576	Measured	0.0	0.0	0.0	
✓ New 0084	100.0%	1 - 62576	Measured	0.0	0.0	0.0	
✓ G_pied	100.0%	1 - 62576	Measured	769.8	127.0	77.6	
✓ G_main	100.0%	1 - 62576	Measured	901.5	-21.9	842.2	
✓ G_hanche	100.0%	1 - 62576	Measured	745.0	-2.1	921.3	
✓ G_genou	100.0%	1 - 62576	Measured	742.7	78.2	542.5	
✓ G_epaule	100.0%	1 - 62576	Measured	846.4	111.4	1438.7	
✓ G_coudé	100.0%	1 - 62576	Measured	859.2	110.1	1148.1	
✓ D_pied	100.0%	1 - 62576	Measured	409.6	196.5	95.2	
✓ D_main	100.0%	1 - 62576	Measured	278.3	12.6	850.9	
✓ D_hanche	100.0%	1 - 62576	Measured	391.2	45.5	935.2	
✓ D_genou	100.0%	1 - 62576	Measured	406.9	131.5	558.5	
✓ D_epaule	100.0%	1 - 62576	Measured	360.2	144.5	1475.3	
✓ D_coudé	100.0%	1 - 62576	Measured	316.2	143.4	1160.2	

FIGURE 22 – Les trajectoires labelisées par mon algorithme

On voit que les 15 trajectoires ont été nommées, ce qui est bien. Mais on remarque autre chose de très préoccupant : il y a 18 trajectoires au lieu des 15 attendues et il y a donc 3 trajectoires en trop : New 0110, New 0103, New 0084.

Évaluation de la détection d’artefacts

Chacune des fonctions artefacts a été testée pour trouver quels paramètres sont optimaux. Les fonctions s’enchainent dans un ordre précis. D’abord, on supprime les artefacts qui bougent trop peu.

Ensuite on cherche les artefacts qui sont trop loin les uns des autres, on les supprime et on boucle ainsi jusqu’à ce qu’il n’y ai plus de points trop loin des autres.

Enfin, on cherche les intervalles avec plus de 15 artefacts pour supprimer les artefacts probables. Voilà les résultats :

Measurement03:	3/3	1.0
Measurement04:	23/24	0.96
Measurement05:	10/11	0.91
Measurement06:	1/1	1.0
Measurement07:	17/17	1.0
Measurement08:	10/10	1.0
Measurement09:	3/3	1.0
Measurement10:	8/8	1.0
Measurement11:	13/13	1.0
Measurement12:	18/19	0.95
Total:	106/109	0.97

FIGURE 23 – Score de détection d’artefacts

Dans l’image ci-dessus : la première colonne correspond au nom du fichier, la seconde à la fraction d’artefacts détectés et la troisième à cette fraction en pourcentage.

Le score final est assez satisfaisant, une critique que l’on peut cependant faire est que j’ai testé mes fonctions sur "seulement" 10 fichiers, les personnes faisait à peu près les mêmes mouvements et la salle ou les personnes ont été filmées est toujours la même. Cette dernière critique est importante car la présence d’artefacts dépend notamment de la luminosité. Une pièce avec des sources de lumières venant de différents angles ou avec une luminosité plus intense pourrait peut-être rendre mes fonctions très inefficaces.

Évaluation de la reconstruction de marqueurs

Pour évaluer mes résultats, j’ai tracé un histogramme et y ai rangé les connexions en fonction de la distance spatio-temporelle. On peut s’attendre à ce que les premières connexions faites (celles avec une petite distance spatio-temporelle) soient plutôt correctes, alors que les dernières soient plutôt erronées. J’ai rangé les données de cette manière pour voir à partir de quelle distance les connexions commencent à être sérieusement fausses.

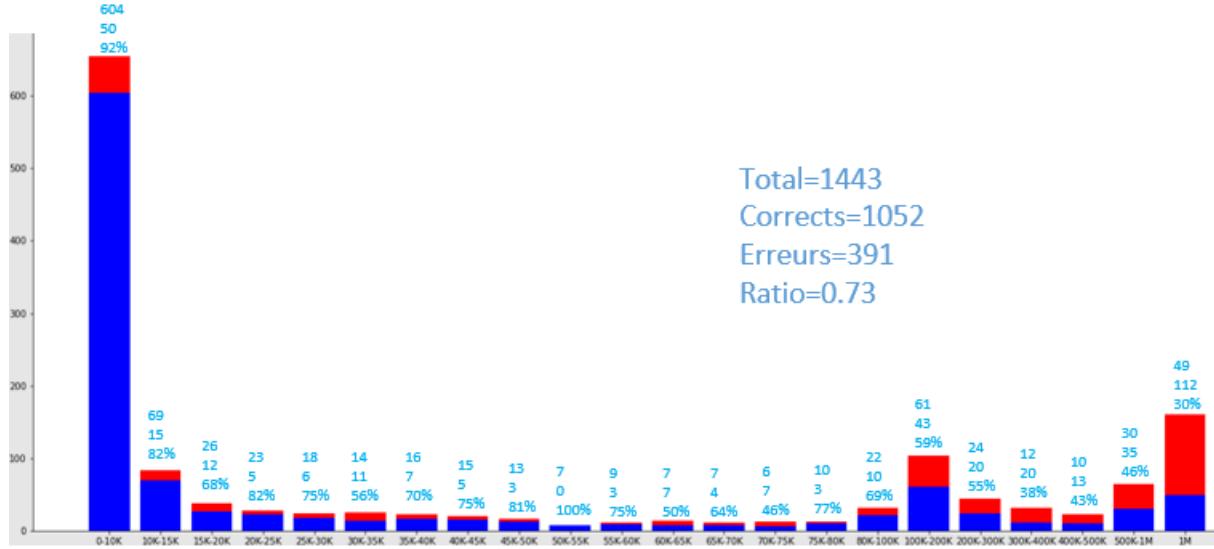


FIGURE 24 – L'évaluation de mon programme

Chaque barre correspond à un intervalle de distance spatio-temporelle. La hauteur de la barre correspond à la quantité de connexions faite. La proportion bleue correspond à la proportion de connexions correctes, la proportion rouge aux erreurs. Les intervalles des barres ne sont pas tous de la même taille. On voit qu'à partir de 60K, les résultats deviennent assez faibles. L'évaluation a été faite sur les 10 fichiers. Au dessus de chaque barre : le premier nombre est le nombre de réussites, le second correspond au nombre des erreurs, et le dernier au pourcentage de réussite.

Mon idée de base était d'arrêter mon programme quand les résultats deviennent faibles, j'aurai ainsi des connexions assez sûres. Mais en fin de compte, avoir la distance à partir de laquelle les résultats se détériorent n'est pas utile. En effet, je dois donner au réseau de neurones les mouvements des 15 marqueurs de bases, or en arrêtant le programme trop tôt, il ne peut pas recomposer les 15 marqueurs. Il peut en donner une trentaine environ. Il y a deux conclusions à en tirer : il n'est plus nécessaire de chercher un seuil, et surtout il faut vraiment que le programme ai de bons résultats pour le donner au réseau de neurones.

Sur mes 10 fichiers après traitement, j'ai 2 fichiers qui ont 15 trajectoires, 5 fichiers avec 16 trajectoires, 2 fichiers avec 17 trajectoires et un fichier avec 18 trajectoires. Il faut donc améliorer le programme pour que mes résultats soient exploitables par le réseau de neurones en ayant les 15 chaînes de connexions.

Le graphe de la figure 24 a été fait sur la méthode que j'ai décrit plus tôt : distance spatiale*distance temporelle.

J'ai aussi testé d'autres méthodes en donnant plus de poids à la distance spatiale ou la distance temporelle. J'ai cherché quelles connexions étaient données en modifiant cette distance et l'évaluation de ces connexions.

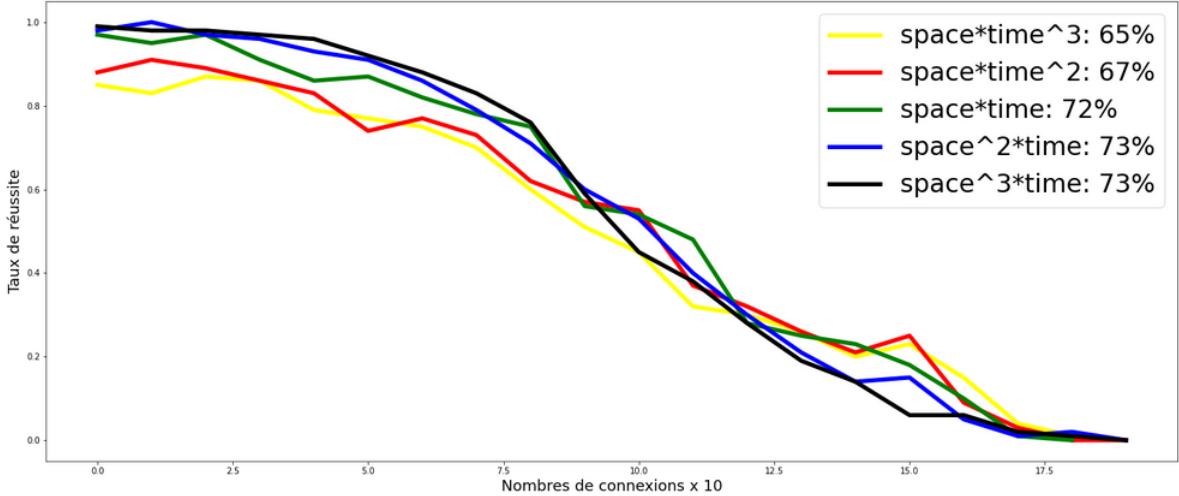


FIGURE 25 – Évaluation de plusieurs métriques spatio-temporelles

Chaque courbe correspond à une distance spatio-temporelle. Plus la courbe est haute, plus elle est performante. Ce taux de réussite est étudié en fonction du nombre de connexions sur l'axe des abscisses. Les résultats correspondent aux moyennes sur les 10 fichiers. Dans la légende, le pourcentage à côté de chaque courbe est le score moyen.

On voit sur la figure 25 les résultats des différentes méthodes. Une première information intéressante est que les méthodes où le poids de l'espace est plus important que celui du temps ont un meilleur score global. Un second élément à retirer est que dans les premières connexions, plus le poids de l'espace est important et plus le score est élevé. Et cette tendance s'inverse progressivement en fonction du nombre de connexions. A la fin ce sont les méthodes avec le plus de poids au temps qui ont un meilleur score.

Mon hypothèse pour expliquer ce curieux phénomène est que quand les premières connexions se font, chaque trajectoire peut être connectées à beaucoup de trajectoires qui sont proches dans l'espace et le temps. Mais au fur et à mesure des connexions, les trajectoires ont de moins en moins de connexions possibles proches. Ce qui fait qu'à la fin, la distance temporelle entre les connexions restante peut être de l'ordre de la minute, or en une minute, la trajectoire peut faire un grand déplacement dans l'espace. Et donc la distance spatiale n'est plus vraiment pertinente.

Pour essayer de palier ce problème, j'ai utilisé une nouvelle technique : faire varier l'importance des distances spatiales et temporelles en fonction du nombre de connexions effectué. On voit sur les premières connexions que les meilleures méthodes mettent un exposant 2 ou 3 à l'espace, et sur les dernières connexions, la meilleure méthode met un exposant 3 au temps. On va donc essayer de passer d'une méthode à l'autre en fonction du nombre de connexions.

$$dist(A, B) = dist_{spat}(A, B)^{3 - \frac{n_connexions}{100}} \times (dist_{temp}(A, B)^{1 + \frac{n_connexions}{100}}) \quad (3)$$

Où $n_connexions$ correspond au nombre de connexions, commençant à 0 et finissant environ à 200. Si on écrit les formules des cas particuliers où $n_connexions=0$ et $n_connexions=200$,

on obtient :

$$dist(A, B, n_connexions = 0) = dist_{spat}(A, B)^3 \times dist_{temp}(A, B) \quad (4)$$

$$dist(A, B, n_connexions = 200) = dist_{spat}(A, B) \times dist_{temp}(A, B)^3 \quad (5)$$

On revient donc bien au deux cas extrêmes en passant progressivement de l'un à l'autre. Le n_connexions est mis à jour toute les 10 connexions.

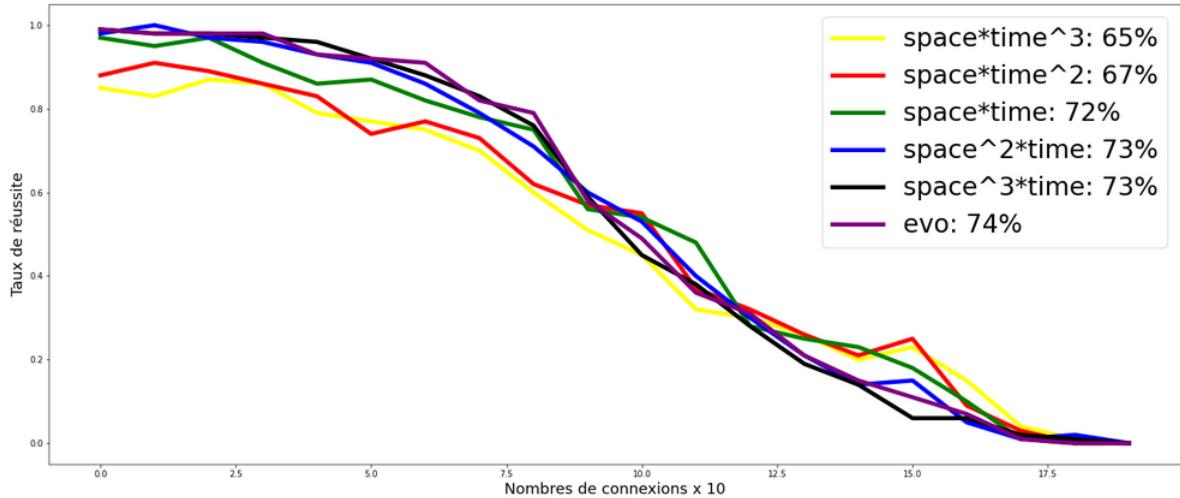


FIGURE 26 – Comparaison de la métrique "évolution" avec les autres métriques

La courbe "evo" a été ajoutée en violet. Elle correspond à la métrique qui évolue avec les connexions comme expliqué plus tôt.

Contrairement à ce qui était attendu, la courbe de la méthode évolution n'a pas un si bon score. Certes, c'est la meilleure courbe actuelle, mais elle ne fait qu'un pourcent de plus que d'autres méthodes. Quand on regarde ses résultats en fonction du nombre de connexions, on se rend compte qu'elle est quasiment toujours comprise entre $space^3 * time$ et $space^2 * time$, même quand il y a un nombre important de connexions.

Enfin, voilà les résultats du programme d'avancée parallèle où l'on force à avoir 15 trajectoires à la fin.

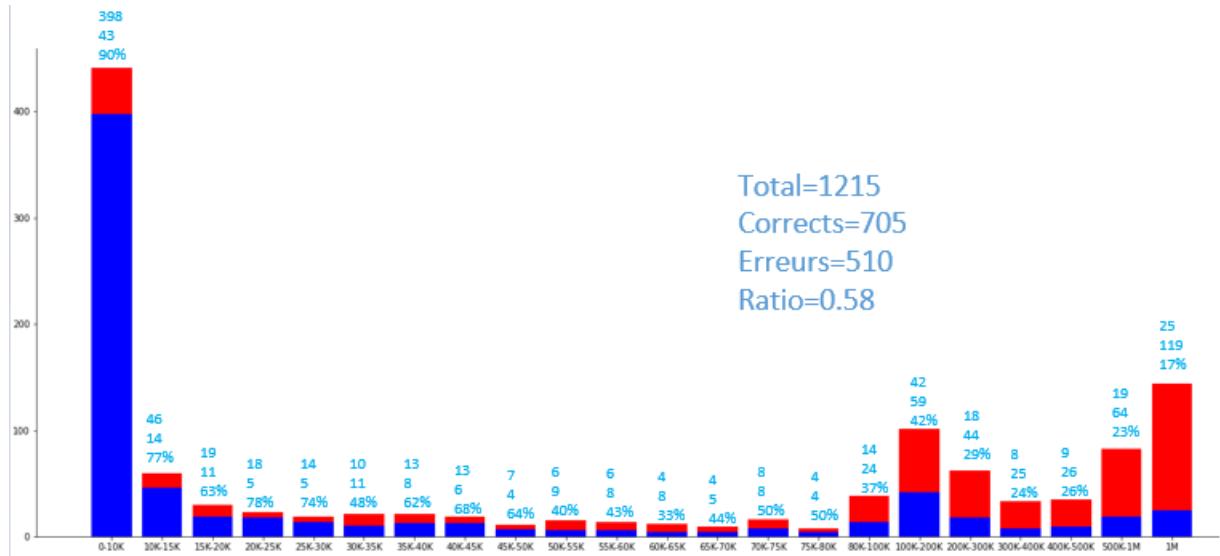


FIGURE 27 – Évaluation de la méthode d'avancée parallèle

Cette méthode semblait prometteuse, mais le score obtenu est assez faible par rapport aux autres méthodes.

Pistes d'amélioration

Une première chose à faire est de ne pas se limiter aux trajectoires qui durent plus de 1% du temps total. Pour traiter plus de 255 trajectoires, on peut découper un fichier avec -admettons- 600 trajectoires en 3 fichiers, chacun avec moins de 255 trajectoires. Les données pourront ainsi être traités avec beaucoup plus de détails sur les mouvements. En vérité, j'ai déjà implémenté cette approche, mais la méthode d'évaluation que j'utilise n'est pas adaptée. Il faudra trouver une nouvelle méthode d'évaluation plus pertinente pour voir si les résultats sont concluants.

On pourra aussi prendre en compte la vitesse et l'accélération des trajectoires. Si une trajectoire a une vitesse très élevée, à cause de son inertie, elle ne peut pas changer de vitesse trop rapidement. Ce critère permettra de discriminer plus efficacement.

Enfin, à cause des gaps générés par les caméras il faut reconstruire les mouvements comme expliqué. Mais une autre chose à faire après la reconstruction est le remplissage des gaps. Qualisys possède déjà 5 algorithmes pour remplir ces trous, et la bibliothèque Python que j'utilise, PyC3d, peut remplir ces gaps. Ces différents algorithmes donnent des résultats corrects quand dans un contexte précis. Il faut alors utiliser ces différents algorithmes en fonction du contexte. Une fois ces algorithmes bien utilisés, on pourra voir le déplacement des marqueurs sans interruptions.

Conclusion

Notre objectif était d'implémenter un programme pour faire fonctionner un réseau de neurones à partir de données c3d. Le travail n'est pas terminé, il faut encore améliorer les performances du programme de construction et améliorer le réseau de neurones.

La principale difficulté est de trouver un algorithme avec de bonnes performances pour la reconstruction. Des idées qui peuvent sembler prometteuses ne sont pas forcément concluantes.

Ce stage m'a permis d'approfondir mes connaissances en Python et d'être à l'aise avec ce langage. J'ai aussi pu en apprendre plus sur PyTorch en comprenant et en modifiant le programme. J'ai également trouvé intéressant de voir le stage sous l'angle plus humain avec la mise en place de l'enregistrement des capture Qualisy plutôt que de tout voir sous l'angle du code. J'ai enfin découvert l'univers de la Motion Capture que je ne connaissais presque pas et quelles sont les problématiques qui se posent dans ce milieu.

Le programme n'est pas encore fonctionnel. La détection des artefacts est efficace, mais les performances du programme de reconstruction sont vraiment trop faibles. Il reste encore un mois pour finir le stage et plusieurs pistes sont à étudier pour résoudre les difficultés restantes.

Webographie

[1] [qualisys - caméras], <https://www.qualisys.com/cameras/miqus/> !tech-specs, (date de consultation 15/08/2022).