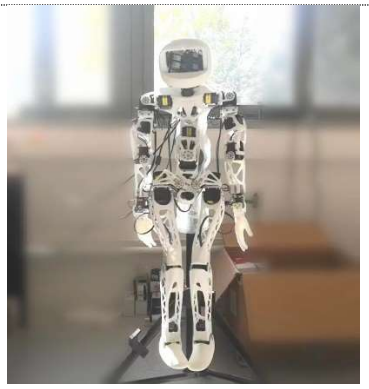
Projet de Recherche (PRe)

Spécialité : STIC

Année scolaire : 2A

Détection et Imitation de Mouvements Humains par un Robot Humanoïde

Projet Keraal



non confidentialité

Auteur : MA Ziqi

Promotion : 2023

Tuteur ENSTA Paris : David FILLIAT
Tuteur organisme d'accueil : Sao Mai NGUYEN

Stage effectué du 20 /05 /2022 au 19/08 /2022

NOM de l'organisme d'accueil : ENSTA Paris
Adresse : 828 boulevard des Maréchaux,
91162 Palaiseau Cedex - France

Attestation de confidentialité / non confidentialité

A remplir par l'entreprise d'accueil et à remettre à la bibliothèque de l'ENSTA Paris[1]

Rapport de Projet de Recherche (PRe)

Par la présente, je soussigné (e) Madame / Monsieur [2] NOM, Prénom

……Nguyen Sao Mai….

Employé(e) en tant que (qualité) ……Enseignante-chercheuse .....

dans la société (NOM et ADRESSE de l'établissement) : ……Ensta Paris .....

Atteste sur l'honneur que LES DONNÉES contenues dans le rapport de

Madame / Monsieur 错误!未定义书签。 NOM Prénom……Ziqi Ma…….. **SONT:**

☐ **CONFIDENTIELLES**

Par conséquent, **l'entreprise** (ou organisme) d'accueil **S'ENGAGE A CONSERVER** le rapport de stage **pendant la Durée d'Utilité Administrative (DUA) soit 5 ans.**

A l'issue de la DUA, l'entreprise s'engage (cocher le choix correspondant) :

☐ A conserver le document
☐ A détruire le document

☒ **NON CONFIDENTIELLES** (**cocher** le choix correspondant)

☐ L'entreprise autorise une **mise en ligne du rapport de stage** (connexion via l'annuaire LDAP de l'école[3]). L'étudiant doit déposer son rapport sur BibNum **- https://bibnum.ensta.fr - (archive institutionnelle de l'Ecole).**

☒ L'entreprise autorise un accès restreint au document (uniquement sur place et au format électronique). L'étudiant doit déposer son rapport sur BibNum **- https://bibnum.ensta.fr - (archive institutionnelle de l'Ecole).**

Un rapport consultable uniquement sur place est stocké sur un PC non connecté au réseau et à internet. Les copies électroniques et imprimées ne sont pas autorisées.

À …Palaiseau……

Le 08 /…08 /…2022….

Signature et cach

[1] Possibilité de transmettre cette attestation par mail - à documentation@ensta-paris.fr – si le document est confidentiel. Si le rapport est non confidentiel, elle devra être déposée en même temps que le rapport sur https://bibnum.ensta.fr

[2] Rayer la mention inutile.

[3] La communauté ENSTA Paris, seule, aura accès au texte intégral. Les métadonnées (titre, auteur etc.) du rapport, en revanche, seront accessibles en ligne par tous.

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont aidé lors de la rédaction de ce rapport.

Tout d'abord, j'adresse mes remerciements au professeur, **M. Sébastien REYMOND de l'ENSTA.** Il m'a beaucoup aidé dans ma recherche de stage et m'a proposé de postuler dans U2IS. Son écoute et ses conseils m'ont permis de cibler mes candidatures, et de trouver ce stage qui était en totale adéquation avec mes attentes.

Je tiens à remercier vivement mon maitre de stage, **Mme Sao Mai NGUYEN, chercheuse de l'Unité d'Informatique et d'Ingénierie des Systèmes(U2IS)**, pour son accueil, le temps passé ensemble et le partage de son expertise au quotidien. Grâce aussi à sa confiance j'ai pu m'accomplir totalement dans mes missions.

Je tiens ensuite à remercier particulièrement **M. Louis ANNABI, Post-doctorant de U2IS**, pour l'attention et l'aide qu'il m'a apportées au quotidien pendant mon stage au sein d'U2IS. Il m'a enseigné beaucoup et m'a encouragé beaucoup.

Je remercie également toute l'équipe U2IS pour leur accueil, leur esprit d'équipe et en particulier **M. Thibault TORALBA** et **M. Philippe BAUMSTIMLER**, qui m'ont beaucoup aidé à travailler avec le robot Poppy.

Enfin, je tiens à remercier toutes les personnes qui m'ont conseillé et relu lors de la rédaction de ce rapport de stage : ma famille, mes amis **Changda TIAN, Mengjun HOU, Yue WANG** et mes camarades de promotion.

Résumé

Le projet Keraal a pour objectif le développement d'un robot kinésithérapeute appelé Poppy capable de « coacher » les patients durant leurs séances de rééducation. Notre travail est une partie du projet Keraal pour améliorer la capacité du robot Poppy.

L'objectif de notre travail est de détecter et imiter des mouvements humains par le robot Poppy. Dans un premier temps, nous essayons le librairie Blazepose pour détecter des squelettes humains dans les vidéos collectées, et nous comparons les résultats avec ceux réalisées par la librairie Kinect, Openpose et Vicon et calculons les écarts. Nous choisissons la librairie Kinect comme la librairie le plus adaptée au travail d'imitation.

Après avoir obtenu les mouvements humains, nous construisons un modèle d'apprentissage pour appliquer un re-ciblage de mouvements entre deux personnages. Nous testons le modèle sur les données d'animation viens de Mixamo. Il est dommage que le modèle ne soit pas performant, ainsi, nous analysons la raison et nous proposons des pistes de travail.

Mots-clés: détection des squelettes, Blazepose, re-ciblage de mouvement, algorithme d'apprentissage

Abstract

The Keraal project aims to develop a physiotherapist robot called Poppy capable of "coaching" patients during their rehabilitation sessions. Our work is a part of the Keraal project to improve the capability of the Poppy robot.

The objective of our work is to detect and imitate human movements by the Poppy robot. Firstly, we try the Blazepose library to detect human skeletons in the collected videos, and we compare the results with those made by the Kinect, Openpose and Vicon library and calculate the differences. We choose the Kinect library as the most suitable library for imitation work.

After obtaining the human motions, we build a learning model to apply motion retargeting between two characters. We test the model on animation data from Mixamo. It is a pity that the model is not efficient, so we analyze the reason and we propose a method to improve.

Keywords: Skeleton detection, Blazepose, motion retargeting, learning algorithm

Table des matières

Table des illustrations

Chapitre 1 - Introduction

1.1 Introduction du projet Keraal

Après un accident certains patients doivent réaliser des séances de rééducation fonctionnelle durant de longs mois afin de recouvrir leurs fonctionnalités musculaires. Une fois rentrés à leur domicile, la plupart d'entre eux ont des difficultés à poursuivre les exercices en toute autonomie, car ils perdent en motivation. Le projet Keraal a pour objectif le développement d'un robot kinésithérapeute appelé Poppy capable de « coacher » les patients durant leurs séances de rééducation. Il peut montrer des exercices de rééducation aux patients, regarder les mouvements du patient, les analyser et faire un retour en encourageant les patients. Notre travail est une partie du projet Keraal pour bien développer le robot Poppy.

1.2 Descriptif du travail

Notre mission lors de ce projet est alors d'améliorer les capacités du robot Poppy. Pour que Poppy soit capable de coacher les patients durant leurs séances de rééducation, il devrait avoir la capacité d'analyser les squelettes des patients. Dans notre travail, nous essayons certains algorithmes de détection des squelettes de l'humain, calculons-les écarts et comparons ses performances. Pour que le robot puisse faire une démonstration de l'exercice au patient, il faut entrainer le robot avec les exercices corrects. Dans notre travail, nous essayons d'appliquer un re-ciblage de mouvement qui peut transférer un mouvement humain au robot par un algorithme d'apprentissage par imitation.

1.3 Description de contribution

L'objectif de notre travail est de détecter et imiter des mouvements humains par le robot Poppy. Donc nous séparons le travail en trois étapes :

1. Détecter les mouvements d'humain.
2. Apprendre les mouvements humains par robot en utilisant la méthode re-ciblage
3. Transférer un mouvement robotique en commande en utilisant le cinématique inverse spécifique.

À cause de la limitation du temps, nous réalisons que deux premières étapes pendant le période du stage.

Pour la première étape, les vidéos de mouvement humain sont données par Mai, il est suffisant de chercher et utiliser des algorithmes de détection. Nous écrivons toutes les détaillées dans le chapitre 2, ce qui contient la détection de pose humain par la librairie BlazePose, la comparaison entre les librairies Blazepose, Kinect, OpenPose et Vicon et le choix de la librairie la plus adaptée.

Après cette étape, nous obtenons les données des mouvements humains et nous pouvons passer à l'étape suivante. Pour la deuxième étape, il faut trouver un algorithme d'apprentissage qui prend les mouvements humains et le squelette de robot comme entrée

et nous rend les mouvements robotiques comme sortie. En premier temps, nous trouvons les articles et les codes qui fait un re-ciblage de mouvement et nous essayons de transférer nos données des mouvements humains sous forme de ses entrées, mais nous avons obtenus de mauvais résultat avec leur modèle pré-entraîné car il ne se transfert pas correctement à nos données. Ainsi, nous abandonnons d'utiliser ses modèles directement. Par contre, en inspirant par ce que nous avons vu, nous essayons de construire notre modèle de re-ciblage de mouvement. Nous écrivons les idées sur notre modèle, la performance de notre modèle et comment nous rajustons le modèle dans le chapitre 3.

Dans le chapitre 5, nous écrivons la conclusion. Un planning du stage est mis dans la section 1.4.

1.4 Planning du Stage

Planning du Stage - 20/05/2022 - 19/08/2022			
Indice	**Terme**	**Date de début**	**Date de Fin**
1	Détection de mouvement avec BlazePose	20/05/2022	29/05/2022
2	Comparaison des écarts	24/05/2022	04/07/2022
3	Travail sur les annotations de donnée Keraal	08/06/2022	16/06/2022
4	Visualisation des mouvements	11/07/2022	13/07/2022
5	Travail sur l'écran du Poppy	25/06/2022	30/06/2022
6	Lire les articles sur l'imitation	13/06/2022	22/06/2022
7	Étude le code dans l'article de [Aberman et al., 2020]	23/06/2022	19/08/2022
8	Transforme le Kinect sous forme .bvh	29/06/2022	06/07/2022
9	Télécharge les données Mixamo	07/07/2022	18/07/2022
10	Traitement des données comme entrée	14/07/2022	22/07/2022
11	Construction des grandes parties de model	19/07/2022	21/07/2022
12	Construction de l'architecture global de réseau	21/07/2022	25/07/2022
13	Entrainement du modèle	26/07/2022	19/07/2022
14	Traitement des données de l'évaluation	03/08/2022	04/08/2022
15	Évaluation	04/08/2022	18/08/2022
16	Modification des paramètres du réseau	04/08/2022	18/08/2022
17	Modification des réseaux	04/08/2022	18/08/2022
18	Rapport final	21/07/2022	19/08/2022

Chapitre 2 – Détection de mouvements

Cette section est dédiée à la détection de mouvement humain. Dans la section 2.1, nous décrivons comment utiliser la librairie BlazePose pour détecter le mouvement. Dans la section 2.2, nous écrivons la comparaison et les écarts entre l'algorithme BlazePose que nous utilisons, et les squelettes générés par Kinect, OpenPose et Vicon. Dans la section 2.3, nous analysons le résultat de la comparaison et choisissons la librairie la plus adaptée au travail suivant. Et dans la section 2.4, nous visualisons les mouvements détectés par quatre librairies.

2.1 Traitement le squelette en utilisant le BlazePose

Commençons tout d'abord par l'analyse d'une vidéo par l'algorithme BlazePose. Nous avons à notre disposition une petite base de données fournie par Mme Nguyen qui contient des vidéos d'exercices de kinésithérapie, dans un environnement idéal. Nous utilisons BlazePose pour obtenir le squelette associé dans chaque vidéo. Une fois exécuté sur chacune des vidéos, les informations de l'articulation sont alors stockées dans la variable POSE_LANDMARK et chaque point de repère se compose des éléments suivants :

- x et y : coordonnées du point de repère normalisées à [0.0, 1.0] par la largeur et la hauteur de l'image respectivement.
- z : Représente la profondeur du point de repère avec la profondeur au milieu des hanches comme origine, et plus la valeur est petite, plus le point de repère est proche de la caméra. La magnitude de z utilise à peu près la même échelle que x.
- visibilité : une valeur dans [0.0, 1.0] indiquant la probabilité que le point de repère soit visible (présent et non occulté) dans l'image.
  L'ordre des articulations est indiqué dans la Fig. 1.

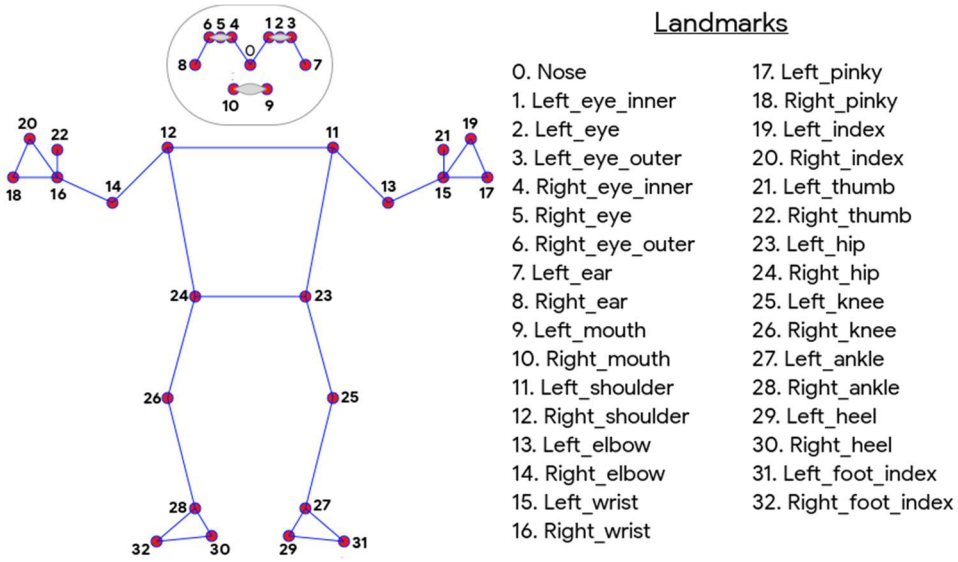


Fig. 1: L'ordre des articulations pour le **Blazepose**.

Pour chaque frame, nous ne traitons que les informations de x, y et z de l'articulation et nous les mettons dans un dictionnaire:

dic[“nom de landmark”] = (x, y, z)

Pour chaque vidéo, nous mettons toutes les frames dans un grand dictionnaire, et nous les stockons sous format ‘.json’. Et voilà, maintenant nous avons toutes les informations associées aux squelettes générés par BlazePose.

2.2 Comparaison des squelettes

2.2.1 Blazepose, Openpose et Kinect

Maintenant, il nous faut comparer le squelette obtenu aux autres squelettes. Tout d’abord, squelette de Openpose nous est donné par l'encadrant et utilise le modèle dans la Fig. 2.

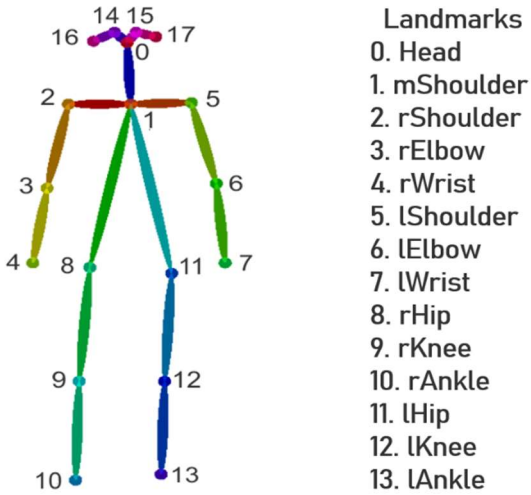


Fig. 2: L’ordre des articulations pour le **Openpose**.

Un point de repère possède les caractéristiques suivantes:

- x et y : coordonnées du point de repère normalisées à [0.0, 1.0] par la largeur et la hauteur de l'image respectivement.

De même, le squelette de Kinect est donné par l’encadrant et possède la forme dans la Fig. 3.

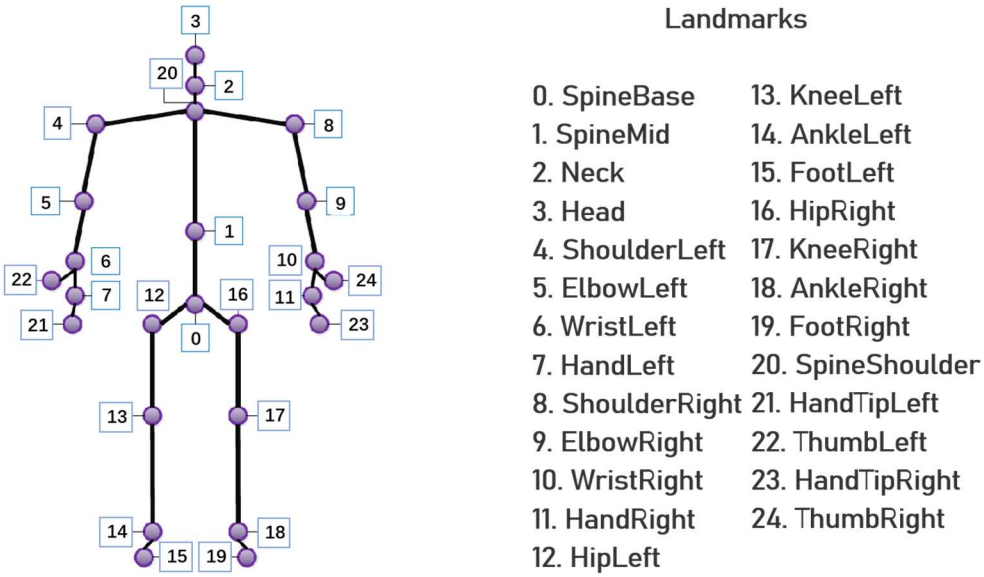


Fig. 3: L'ordre des articulations pour le **Kinect**.

Chaque point de repère se compose des caractéristiques suivantes:

- x_pos, y_pos, z_pos, x_quat, y_quat, z_quat, w_quat dans le repère caméra, ce dont nous avons besoin, c'est le (x_pos, y_pos, z_pos).

Pour faciliter la comparaison, nous transposons le repère de Blazepose et le repère de Kinect par rapport au repère de Openpose， et nous utilisons les relations correspondantes dans la Fig. 4.

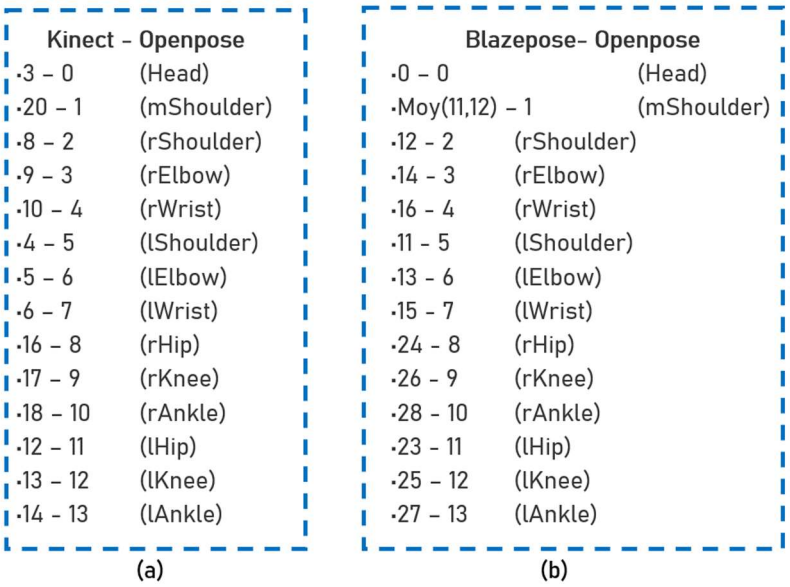


Fig. 4: Les relations correspondantes entre (a) Kinect et Openpose. (b) Blazepose et Openpose.

Ainsi par exemple, le point 3 du squelette Kinect correspond au point 0 du squelette OpenPose. De même, le milieu des points 11 et 12 du squelette Blazepose correspond au point 1 du squelette OpenPose.

Nous souhaitons maintenant réaliser la fonction d'écart entre les squelettes. Comme Blazepose et Openpose utilisent le repère de l'image, une simple relation de transfert sera suffisante pour la comparaison. L'écart se fait de manière immédiate par comparaison des positions sur l'image des points que nous considérons équivalent.

Cependant nous observons un cas pathologique pour le squelette de la Kinect. En effet, les deux algorithmes ne sont pas basés sur le même repère, ce qui fait que nous ne pouvons pas les comparer directement. Nous devrons procéder à une transformation un peu plus élaborée.

Pour changer le repère de Kinect par rapport au repère Openpose, Nous choisissons le point mShoulder comme point de référence et la distance de 'mShoulder-Hip' comme distance de référence dans chaque repère. Ainsi, afin de comparer la position de deux points équivalent des deux squelettes, nous comparons la distance de ces deux points par rapport au point de référence, et nous normalisons par rapport à la distance de référence. Nous obtenons alors une valeur dans [0;1] que nous pouvons comparer.

Il suffit alors, pour chaque articulation, de calculer l'écart moyen et son écart-type

dans toutes les vidéos selon les relations de transfert décrites précédemment, et de mettre les données calculées de toutes les articulations pour toutes les vidéos dans un tableau. Nous traçons les courbes de toutes les articulations pour toutes les vidéos dans la section 2.3.

2.2.2 Blazepose, Openpose, Kinect et Vicon

Nous trouvons aussi dans le projet de Keraal, il y a aussi des squelettes de plusieurs participants traités par le Vicon, donc nous avons besoin d'ajouter les squelettes de Vicon dans la comparaison des écarts. Le squelette de Vicon est donné par l'encadrant et possède la forme dans la Fig. 5.

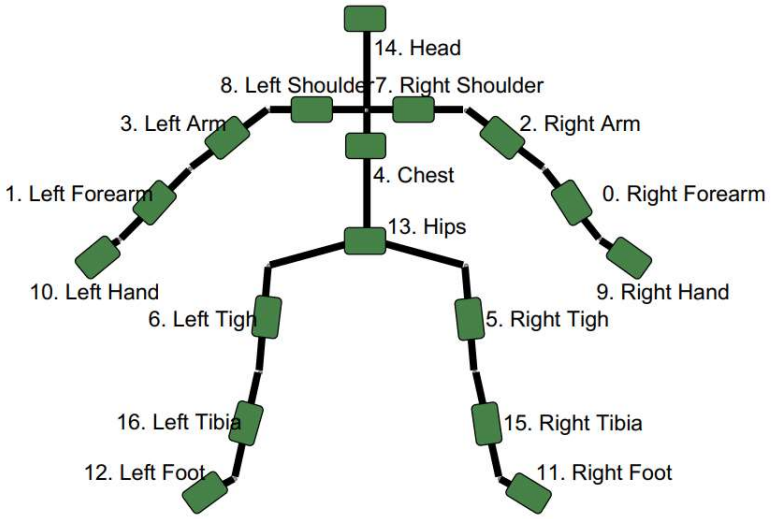


Fig. 5: L'ordre des articulations pour le **Vicon**.

Chaque point de repère se compose des caractéristiques suivantes:

- x_pos, y_pos, z_pos, x_quat, y_quat, z_quat, w_quat dans le repère caméra, ce dont nous avons besoin, c'est le (x_pos, y_pos, z_pos).

Nous constatons que les squelettes notés par Vicon sont différents que les trois premiers, il note les <forearm>, <Tigh> ou <Tibia> qui est une position entre deux articulations, nous pensons que ce sont les positions plus stables et plus précis et nous utilisons les articulations données par Vicon comme la vérité terrain.

Nous transposons le repère de Blazepose, Kinect et Openpose par rapport au repère de Vicon, et nous utilisons les relations correspondantes dans la Fig. 6.

Blazepose - Vicon	
·0 – 14	(Head)
·Moy(11,12,23,24) – 4	(Chest)
·Moy(23,24) – 13	(Hips)
·11 - 8	(Left Shoulder)
·Moy(11,13) – 3	(Left Arm)
·Moy(13,15) – 1	(Left Forearm)
·Moy(17,19) – 10	(Left Hand)
·12 – 7	(Right Shoulder)
·Moy(12,14) - 2	(Right Arm)
·Moy(14,16) - 0	(Right Forearm)
·Moy(18,20) - 9	(Right Hand)
·Moy(23,25) - 6	(Left Tigh)
·Moy(25,27) - 16	(Left Tibia)
·Moy(29,31) - 12	(Left Foot)
·Moy(24,26) - 5	(Right Tigh)
·Moy(26,28) – 15	(Right Tibia)
·Moy(30,32) – 11	(Right Foot)

(a)

Openpose- Vicon	
·0 - 14	(Head)
·Moy(1, Moy(8,11)) – 4	(Chest)
·Moy(8,11) - 13	(Hips)
·5 - 8	(Left Shoulder)
·Moy(5,6) - 3	(Left Arm)
·Moy(6,7) - 1	(Left Forearm)
·nan - 10	(Left Hand)
·2 - 7	(Right Shoulder)
·Moy(2,3) - 2	(Right Arm)
·Moy(3,4) - 0	(Right Forearm)
·nan - 9	(Right Hand)
·Moy(11,12) - 6	(Left Tigh)
·Moy(12,13) - 16	(Left Tibia)
·nan - 12	(Left Foot)
·Moy(8,9) - 5	(Right Tigh)
·Moy(9,10) - 15	(Right Tibia)
·nan - 11	(Right Foot)

(b)

Kinect- Vicon	
·3 - 14	(Head)
·Moy(1, 20) - 4	(Chest)
·0 - 13	(Hips)
·4 - 8	(Left Shoulder)
·Moy(4,5) - 3	(Left Arm)
·Moy(5,6) - 1	(Left Forearm)
·7 - 10	(Left Hand)
·8 - 7	(Right Shoulder)
·Moy(8,9) – 2	(Right Arm)
·Moy(9,10) – 0	(Right Forearm)
·11 – 9	(Right Hand)
·Moy(12,13) – 6	(Left Tigh)
·Moy(13,14) – 16	(Left Tibia)
·15 – 12	(Left Foot)
·Moy(16,17) – 5	(Right Tigh)
·Moy(17,18) – 15	(Right Tibia)
·19 – 11	(Right Foot)

(c)

Fig. 6 : Les relations correspondantes entre (a) Blaze et Vicon. (b) Openpose et Vicon. (c) Kinect et Vicon.

Le changement de repère est similaire à celui dans la section 2.2. Nous mettons aussi les résultats de comparaisons dans la section 2.4.

2.3 Analyse des écarts et interprétation

Nous avons effectué l'analyse des écarts sur trois types d'exercices: "cache tête", "étirement latéral" et "rotation du tronc". Les vidéos nous ont été fournies par Mme Nguyen. Pour chacun des exercices, nous avons plusieurs vidéos de deux catégories différentes : des exercices dit réussis par le patient et des exercices ratés par le patient. Nous ne nous intéressons pas particulièrement à cette distinction pour l'instant car nous souhaitons simplement obtenir l'écart entre les squelettes.

Pour chaque type d'exercices et pour chaque type de squelettes, nous visualisons une partie en utilisant la librairie cv2 dans le Python, nous présentons le résultat dans la Fig. 7.

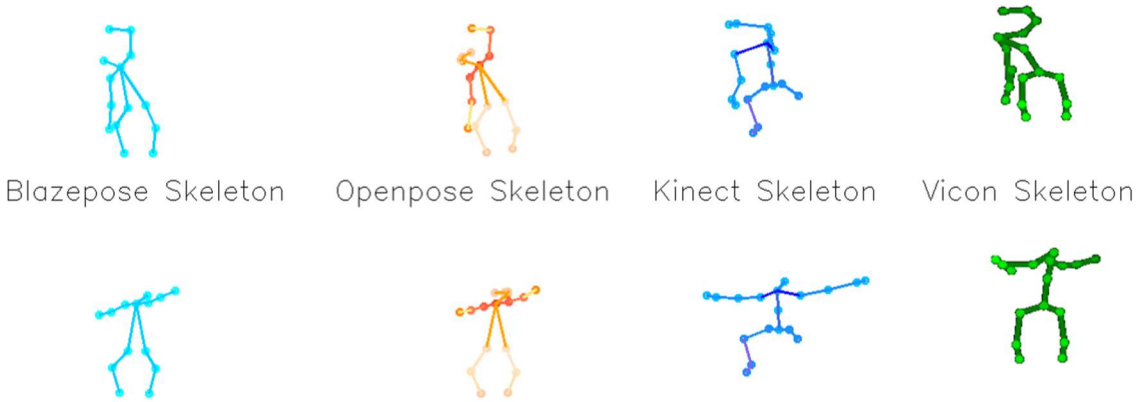


Fig. 7: Visualisation des mouvements. Le mouvement dans la première ligne est “cache tête”, dans la deuxième est “étirement latéral” et dans le troisième est “rotation du tronc”, les squelettes correspondants aux colonnes sont Blazepose, Openpose, Kinect et Vicon.

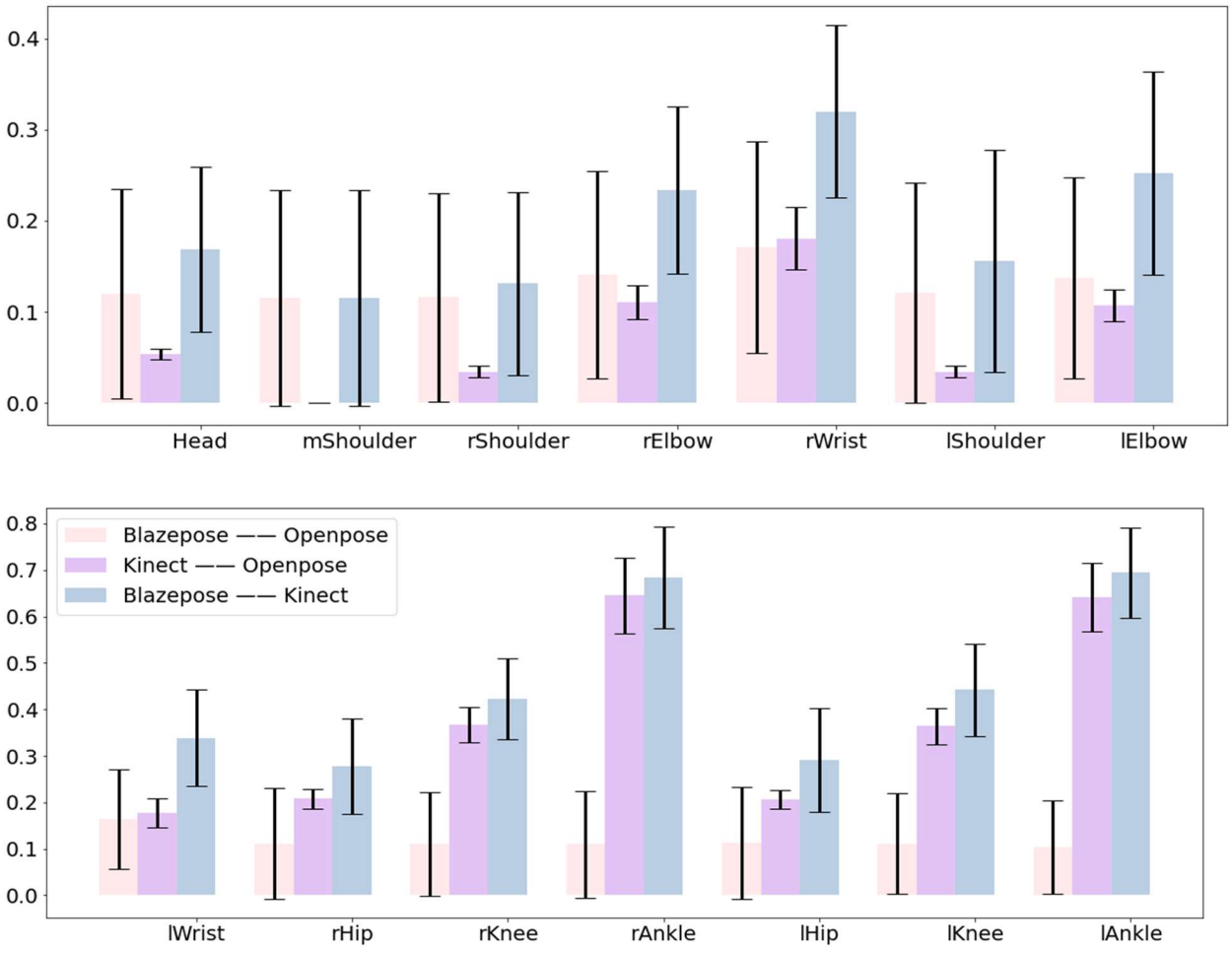


Fig. 8: L’écart pour les articulations entre (b) Blazepose et Openpose. (c) Kinect et Openpose. (a) Blazepose et Kinect.

Nous observons que le squelette obtenu via Blazepose est plus proche du squelette réalisé par Openpose que celui de Kinect. Quand nous comparons l’écart entre Blazepose et

Kinect, les écarts des points lAnkle, rAnkle, lKnee et rKnee sont très élevés comparé aux autres. Nous pensons que cela est dû au fait que tous les mouvements soient assis, il est alors plus difficile de détecter les articulations en bas, car les mouvements en bas sont moins étirés. Les comparaisons de trois méthodes avec la vérité terrain (Vicon) sont dans le Tab. 1. Il y a des 'nan' dans le tableau, c'est parce que Openpose ne note pas les notations sur les mains et les pieds.

Tab. 1: Les comparaisons du Blazepose, Kinect et Openpose avec Vicon.

Articulation	Type de comparaison		
	Blazepose - Vicon	Kinect - Vicon	Openpose - Vicon
Right_Forearm	0.3102+-0.0694	0.3232+-0.0644	0.1423+-0.0387
Left_Forearm	0.3290+-0.0884	0.1899+-0.0573	0.2490+-0.0742
Right_Arm	0.2282+-0.0616	0.2788+-0.0442	0.0880+-0.0228
Left_Arm	0.2931+-0.0571	0.1364+-0.0378	0.2425+-0.0458
Chest	0.1368+-0.0198	0.1097+-0.0188	0.0799+-0.0093
Right_Tigh	0.1611+-0.0332	0.0862+-0.0333	0.1510+-0.0369
Left_Tigh	0.1036+-0.0372	0.1457+-0.0382	0.0666+-0.0514
Right_Shoulder	0.2436+-0.0525	0.2456+-0.0381	0.1159+-0.0204
Left_Shoulder	0.2802+-0.0429	0.1777+-0.0285	0.1937+-0.0179
Right_Hand	0.4125+-0.0811	0.3731+-0.0848	nan+-nan
Left_Hand	0.3914+-0.1423	0.2677+-0.0757	nan+-nan
Right_Foot	0.3607+-0.0996	0.2078+-0.0866	nan+-nan
Left_Foot	0.2800+-0.1132	0.3246+-0.0845	nan+-nan
Head	0.4217+-0.0586	0.2913+-0.0423	0.2037+-0.0247
Right_Tibia	0.2756+-0.0783	0.1308+-0.0800	0.2759+-0.0602
Left_Tibia	0.1754+-0.0952	0.2532+-0.0668	0.1645+-0.0711
Moyen	0.2752+-0.0706	0.2214+-0.0551	0.1644+-0.0395

Nous constatons que la valeur moyenne de l'écart de Openpose est beaucoup plus petite que les deux autres, c'est parce que les end-effecteurs ont souvent plus de l'erreur que d'autre articulation, mais le Openpose ne mesure pas les end-effecteurs. Et puis, le Openpose ne mesure que deux dimensions au lieu de trois, ce qui peut aussi réduire l'écart de l'Openpose. Comme Openpose manque la dimension de profondeur pour les données, dans les étapes suivantes, nous ne considérons pas le Openpose.

D'après le tableau, comme l'écart moyen du Kinect est plus petite que celle de Blazepose, nous pensons que Kinect performe mieux que les autres. **Donc dans les expérimentations suivantes, nous utilisons les squelettes traités par le Kinect.**

Chapitre 3 Re-ciblage de mouvement basé sur les données

Après avoir fini les travaux sur la détection du mouvement, nous commençons les travaux sur l'apprentissage des mouvements humains par le robot Poppy. Nous étudions le re-ciblage de mouvement. L'objectif de cette partie est de construire un modèle qui peut apprendre des trajectoires d'un personnage par un autre personnage. Le squelette des deux personnages peut-être possède les structures différentes, mais ils sont topologiquement équivalents.

Dans la section 3.1, nous écrivons les méthodes que nous avons étudiées sur le re-ciblage de mouvement. Dans la section 3.2, nous construisons les opérateurs pour traiter les informations squelettiques, dans la section 3.3, nous construisons notre modèle, dans la section 3.4, nous faisons des expérimentations avec notre modèle et plus tester sa performance, dans la section 3.5, nous analysons la performance et nous décrivons comment nous rajustons le modèle.

3.1 Travail relative

Traditionnellement, le re-ciblage de mouvement est effectué en définissant manuellement une transformation entre deux morphologies différentes (par exemple, un acteur humain et un personnage d'animation). Cela nécessite de commencer par concevoir la fonction de pose dans le domaine de source, puis de trouver la pose correspondante dans le domaine cible. Le re-ciblage de mouvement basé sur les données a été utilisé pour changer le processus de transformation manuelle par des méthodes d'apprentissage automatique. Ces méthodes basées sur l'apprentissage bénéficient de flexibilité et d'évolutivité car elles réduisent le besoin d'une connaissance extensive du domaine et des processus de réglage fastidieux nécessaires pour définir correctement les caractéristiques de pose.

Parmi tous les articles que nous avons vus, [Devanne et al., 2018] nous propose une méthode qui s'est appuyée sur des modèles de variables latentes à processus gaussien (GPLVM) pour construire des espaces latents partagés entre deux domaines de mouvement.

L'article écrit par [Choi et al., 2021] nous présente un encodeur-décodeur pour faire le re-ciblage de mouvement. Il considère le fait que l'espace de mouvement du robot est souvent plus restreint que celui des humains, donc il construit un encodeur-décodeur entre l'espace de mouvement d'humain que peut faire par le robot (l'espace Q) et l'espace de mouvement de robot, et puis il crée une projection de tous les mouvements d'humain à l'espace Q (On l'appelle Projx dans la partie suivante). Son idée est illustrée dans la Fig. 9.

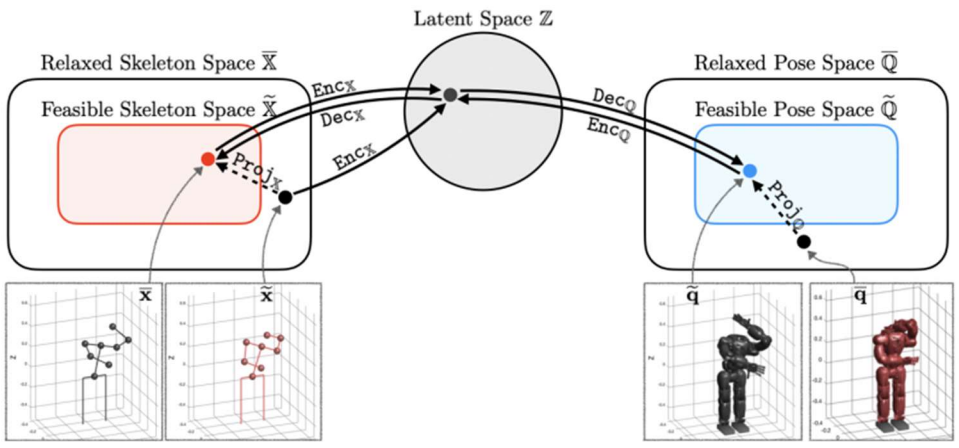


Fig. 9: L'idée de l'article dans [Choi et al., 2021].

Mais pour entrainer le modèle dans cette méthode, il utilise les données appariées sous une quantité de de 200,000 poses. N'ayant pas à notre disposition une telle base de données appariées humain-Poppy, nous cherchons d'autres méthodes.

L'article [Aberman et al., 2020] nous offre une bonne inspiration. Dans son article, il voit un squelette de l'articulation comme une topologie de graphe, et traite les informations de pose avec des couches de convolution sur graphe et des couches de pooling sur graphe. En appliquant une couche de convolution graphe sur les données de pose, les auteurs associent à chaque articulation un ensemble de caractéristiques qui sont calculées à partir des caractéristiques des articulations voisines. En combinant ces couches avec des couches de convolution temporelles, ils construisent un encodeur et un décodeur permettant de traiter des trajectoires de poses. La structure de son modèle est dans la Fig. 10.

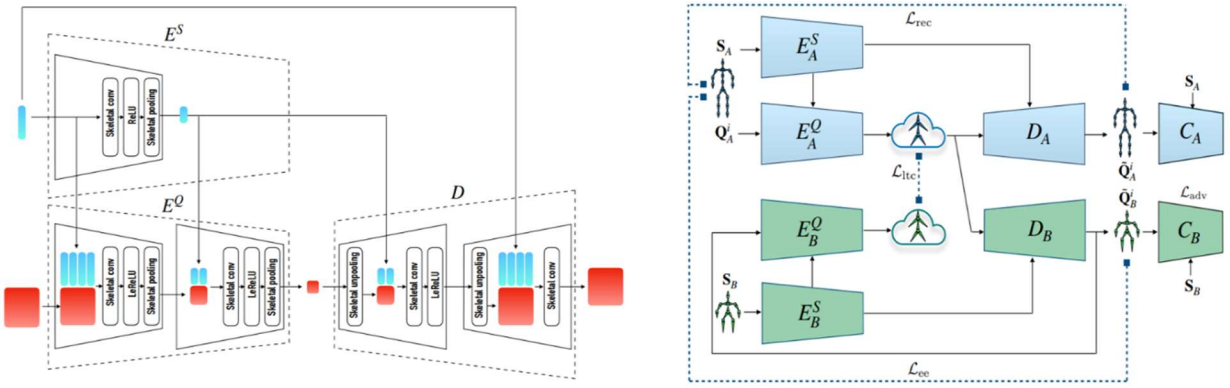


Fig. 10: La structure de l'article [Aberman et al., 2020].

Leur méthode s'inspire de l'architecture CycleGAN, apprenant à transférer les données d'un domaine A à un domaine B sans utiliser de données appariées, en exploitant des discriminateurs coentraînés sur chaque domaine. L'architecture proposée par [Aberman et al. 2020] permet d'apprendre à recibler des séquences de positions d'un domaine à un autre, **en utilisant uniquement des données non-appariées. Cet atout majeur a fait de cet article notre principale source d'inspiration pour ce travail.**

3.2 Opérateurs du squelette

En s'inspirant de l'article écrit par [Aberman et al., 2020], ainsi que de la littérature existante sur les Graph Neural Network, nous proposons un nouvel opérateur pour apprendre les informations profondes du squelette. Par la suite, nous décrivons le format de fichier employé pour décrire les mouvements du Mixamo utilisé par [Aberman et al., 2020], ainsi que les nouveaux opérateurs que nous avons définis pour traiter les séquences de poses.

3.2.1 Représentation de mouvements

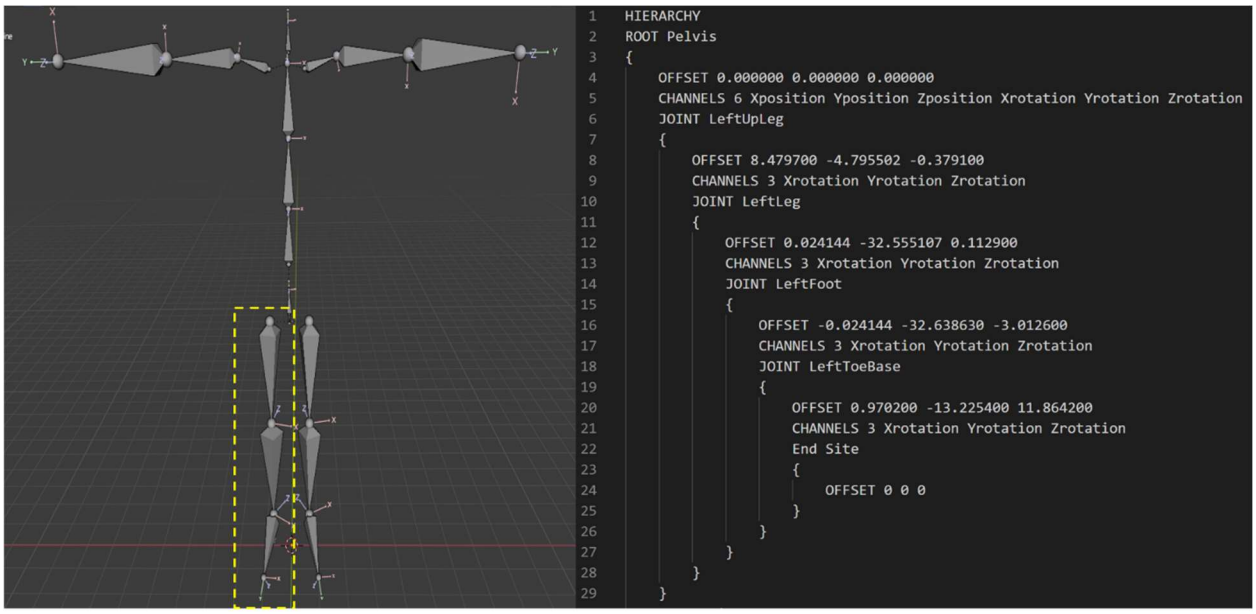
Dans l'article [Aberman et al., 2020] dont nous nous inspirons, une séquence de mouvement de longueur T est décrite par une composante statique $S \in \mathbb{R}^{J \times S}$, et une composante dynamique $Q \in \mathbb{R}^{T \times J \times Q}$, où J est le nombre d'articulation, et S et Q sont les

dimensions de caractéristique statique et de caractéristique dynamiques, respectivement. En général, S = 3 pour les positions dans la direction de x, y et z. Q = 3 si on utilise les 3 angles de rotations par rapport aux axes x, y et z, ou Q = 4 les quaternions dans x, y, z et w.

La composante statique S consiste en un ensemble de longueur d'os (vecteurs 3D), qui décrivent le squelette dans une pose initiale T, tandis que la composante dynamique Q spécifie les séquences temporelles de rotations de chaque articulation (par rapport au cadre de coordonnées de son parent dans la chaîne cinématique), représenté par des quaternions unitaires. L'articulation racine $R \in \mathbb{R}^{T \times (S+Q)}$ est représentée séparément des J armatures (ses enfants), comme une séquence de translations et de rotations globales.

La structure du squelette est représentée par un graphe arborescent dont les nœuds correspondent aux articulations et aux effecteurs terminaux, tandis que les arêtes correspondent aux armatures, comme illustré sur la Fig. 11. Ainsi, pour un squelette à J armatures, le graphe a J+1 nœuds. La connectivité est déterminée par les chaînes cinématiques (les chemins de l'articulation de racine aux effecteurs terminaux) et exprimée par des listes d'adjacence $N^d = \{N_1^d, N_2^d, \ldots, N_j^d\}$, où N_i^d désigne les arêtes dont la distance dans l'arbre est égale ou inférieure à d à partir de la i-ième arête.

Motivés par l'intuition que la manipulation de représentations de rotations sous forme de quaternions est compliquée pour un réseau de neurones [Xiang et al., 2020], nous proposons à la place d'utiliser les données de positions des articulations. **Dans notre cas, nous posons S = 1 pour la longueur de l'armature et Q = 3 pour les positions de l'articulation par rapport au axes x, y et z. Dans la partie suivante, nous utilisons la notation L qui représente la longueur de l'armature et la notation P qui représente la position.**



```
HIERARCHY
ROOT Pelvis
{
    OFFSET 0.000000 0.000000 0.000000
    CHANNELS 6 Xposition Yposition Zposition Xrotation Yrotation Zrotation
    JOINT LeftUpLeg
    {
        OFFSET 8.479700 -4.795502 -0.379100
        CHANNELS 3 Xrotation Yrotation Zrotation
        JOINT LeftLeg
        {
            OFFSET 0.024144 -32.555107 0.112900
            CHANNELS 3 Xrotation Yrotation Zrotation
            JOINT LeftFoot
            {
                OFFSET -0.024144 -32.638630 -3.012600
                CHANNELS 3 Xrotation Yrotation Zrotation
                JOINT LeftToeBase
                {
                    OFFSET 0.970200 -13.225400 11.864200
                    CHANNELS 3 Xrotation Yrotation Zrotation
                    End Site
                    {
                        OFFSET 0 0 0
                    }
                }
            }
        }
    }
```

(a) (b)

Fig. 11: (a) Une représentation de T-pose du mouvement. (b) une structure de squelette pour la jambe en jaune.

3.2.2 Opérateurs pour traiter les squelettes

Nous intégrons l'idée de Graph Neural Network dans notre architecture. Les noyaux de convolution considèrent la structure du squelette pour calculer les caractéristiques locales à travers les armatures. Pour construire un graphe, nous mettons les informations dynamiques dans les nœuds et mettons les informations statiques dans les arrêtes. Chaque frame de la séquence correspond à un graphe comme illustré dans la Fig. 12.

Les données d'entrées sont organisées sous forme de graphe, avec les positions de chaque articulation comme caractéristiques de sommets (nous utilisons la notation **P**), et les longueurs des armatures comme caractéristiques d'arête (nous utilisons la notation **L**). Nous définissons deux opérateurs implémentant des convolutions sur graphe en tenant compte des informations de sommets et d'arêtes. En combinant ces deux opérateurs, nous obtenons une couche qui permet de mettre à jour les caractéristiques de chaque articulation en prenant en compte ses caractéristiques d'entrée, les caractéristiques des articulations voisines, ainsi que les caractéristiques des arêtes voisines. Ces opérateurs permettent donc de combiner et traiter de l'information de manière locale dans le graphe. Le traitement effectué par nos opérateurs est donné dans les équations suivantes:

- Node2Edge : $\hat{L}_i = \frac{1}{|N_i^d|+1}\left\{\sum_{j\in N_i^d}(P_j * W_{ne_j}^{\ i} + b_{ne_j}^{\ i}) + L_i * W_{e_i}^{\ i} + b_{e_i}^{\ i}\right\}$, où $P_j \in \mathbb{R}^{J\times P}$ représente les informations de nœud voisin et $L_i \in \mathbb{R}^{J\times L}$ représente lui-même.
- Edge2Node : $\hat{P}_i = \frac{1}{|N_i^d|+1}\left\{\sum_{j\in N_i^d}(L_j * W_{en_j}^{\ i} + b_{en_j}^{\ i}) + P_i * W_{n_i}^{\ i} + b_{n_i}^{\ i}\right\}$, où $L_j \in \mathbb{R}^{J\times L}$ représente les informations d'arête voisine et $P_i \in \mathbb{R}^{J\times P}$ représente lui-même.

Les sommets et les arêtes partagent la même matrice de connectivité N^d et N_i^d représente la connectivité correspond au i-ième sommet, ce qui nous permet d'appliquer de manière récursive la convolution squelettique aux séquences de mouvement. Une explication dans la Fig. 12 peut-être plus claire.

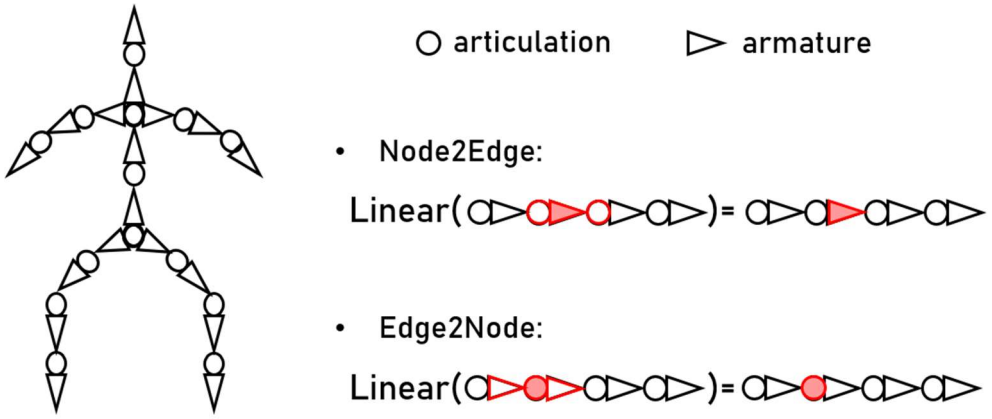


Fig. 12: La représentation de l'humain et les opérateurs Node2Edge et Edge2Node. La couche Node2Edge met à jour les caractéristiques d'une armature en s'aidant des caractéristiques des articulations voisines. La couche Edge2Node met à jour les caractéristiques d'une articulation en s'aidant des caractéristiques des armatures voisines.

En utilisant les deux opérateurs décrits avant, nous construisons un module GNN.

Nous traçons la structure du **module GNN** dans la Fig. 13. Nous donnons les données statiques et dynamiques comme entrée à la couche de Node2Edge, et nous obtenons les caractéristiques statiques, en passant les caractéristiques statiques et les données dynamiques à la couche de Edge2Node, les caractéristiques dynamiques sont obtenues. Les sorties après ce module sont les caractéristiques statiques et dynamiques Nous pensons qu'avec ce module, après la couche Node2Edge, les données dynamiques peuvent influencer sur les caractéristiques statiques, et après la couche Edge2Node, les caractéristiques dynamiques peuvent aussi influencer sur les caractéristiques statiques. **Nous pensons que ce module est capable de faire un échange de caractéristique entre les données dynamiques et les données statiques, ce qui peut traiter les informations profondes et invariantes pour une suite de trajectoire.**

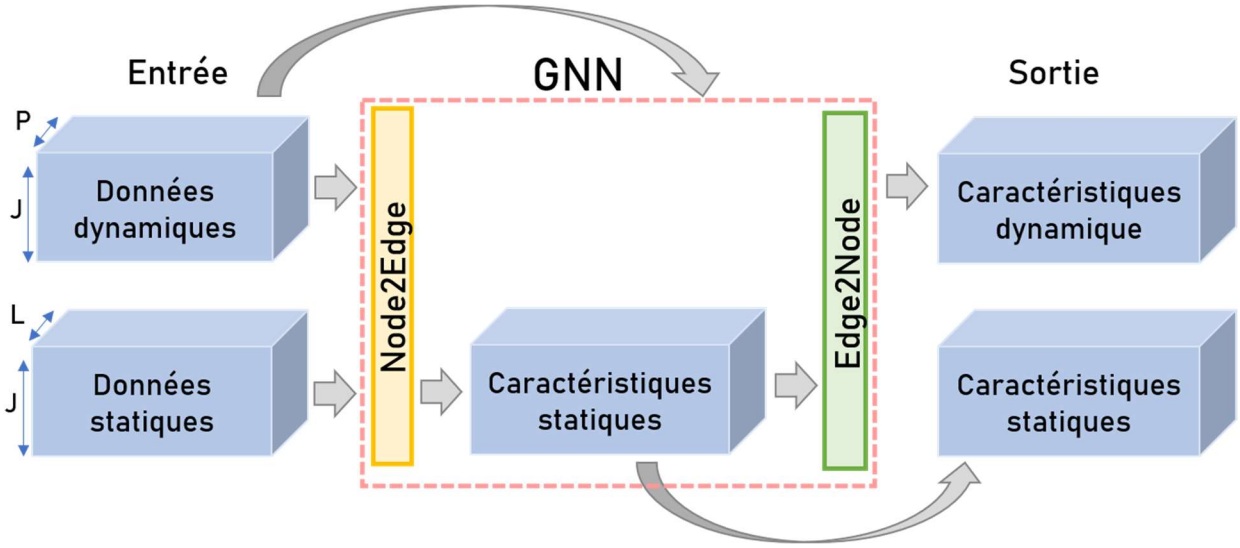


Fig. 13: La structure de couche GNN.

3.3 Modèle

Dans cette partie, nous écrivons les modules que nous concevons pour traiter les trajectoires et l'architecture global du modèle.

3.3.1 Descriptif mathématique du problème

Nous formulons le re-ciblage de mouvement comme une tâche de traduction de domaine non apparié. Plus précisément, nous laissons $\mathcal{M}_A$ et $\mathcal{M}_B$ désigner deux domaines de mouvement, où les mouvements dans chaque domaine sont effectués par des squelettes avec la même structure squelettique, mais peuvent avoir des longueurs et des proportions d'os différentes (L_A et L_B, respectivement). Cette situation s'adapte aux ensembles de données Mixamo publics existants, où chaque ensemble de données contient différents personnages qui partagent la structure squelettique et effectuent divers mouvements. Nous supposons qu'il existe un homéomorphisme entre les structures squelettiques de L_A et L_B. Notons que les domaines ne sont pas appariés, ce qui signifie qu'il n'y a pas de paires explicites de mouvements dans les deux domaines.

Supposons que chaque mouvement $i \in \mathcal{M}_A$ soit représenté par le couple (L_A, P_A^i),

où $L_A \in \mathcal{S}$ est l'ensemble des longueurs des os du squelette et P_A^i sont les positions articulaires, dans notre situation. Étant donné les longueurs des os d'un squelette cible $L_B \in \mathcal{S}$, notre objectif est de mapper (L_A, P_A^i) dans un ensemble reciblé de position P_B^i qui décrivent le mouvement tel qu'il devrait être effectué par L_B. Formellement, on cherche une application pilotée par les données $G^{A\to B}$:

$$G^{A\to B}\left((L_A, P_A^i) \in \mathcal{M}_A, L_B \in \mathcal{S}\right) \to (L_B, P_B^i)$$

3.3.2 Structure des modules

Nous concevons trois modules, un encodeur, un décodeur et un discriminateur. Nous les présentons dans la Fig. 14. L'idée de construction de module est d'effectuer des convolutions squelettiques-temporelles en alternant des couches de GNN et des couches de convolution temporelle (1d).

L'encodeur se compose de trois couches, une couche de Conv1d, une couche de GNN et une couche de Conv1d séparément. La fonction d'activation pour tous les couches est LeakyReLU. La couche de Conv1d est utilisée pour condenser les informations selon l'axe temporel, et après cette action, les données condensées sont passées dans le GNN pour obtenir les informations profondes.

Le décodeur se compose aussi de trois couches, la première couche est une couche de Upsample qui sert à augmenter artificiellement la longueur de la séquence sans paramètres apprenables. Et puis, nous utilisons une couche de GNN pour faire un échange entre données statiques et dynamiques. En fin, nous utilisons une couche de ConvTranspose1d qui ont un fonctionnement opposé que la Conv1d, c'est pour étirer les informations selon l'axe temporel.

Le discriminateur a la même structure que l'encodeur, et nous ajoutons des couches linéaires à la fin, pour que sa sortie soit un scalaire correspondant au score estimé par le discriminateur.

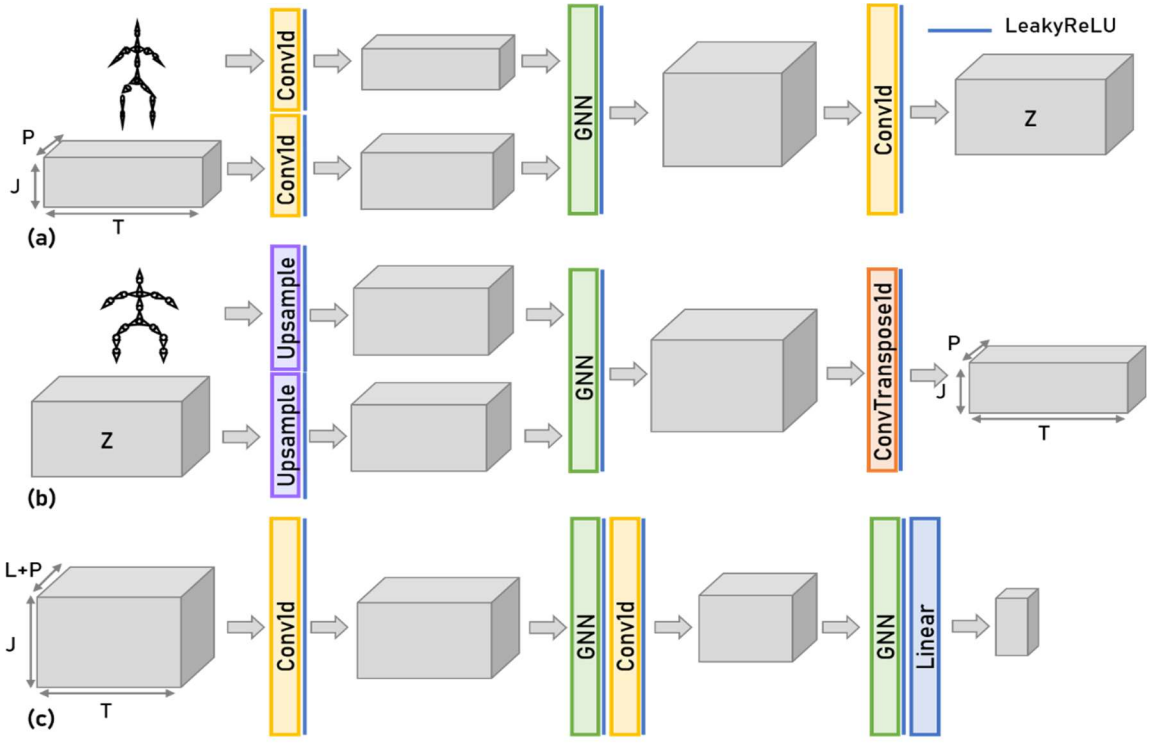


Fig. 14: (a) La structure de l'encodeur (b) La structure du décodeur (c) La structure du discriminateur.

3.3.3 Architecture du réseau

Nous utilisons les trois modules principaux et construisons l'architecture global. Nous réalisons deux architectures globales, une vue de niveau supérieur du flux d'informations dans notre architecture présente dans la Fig. 15 et la Fig. 16. Comme le premier est inspiré par l'article de [Aberman et al., 2020] (Fig. 15) et le deuxième a une architecture comme le Cycle-GAN (Fig. 16), on les appelle **réseau d'encodeur décodeur** et **réseau cycle**. Les deux encodeurs et deux décodeurs dans tous architectures sont les mêmes, nous avons besoin d'entraîner des modules généraux. Après avoir formé les composants susmentionnés, la transformation souhaitée $G^{A\to B}$ est obtenue, au moment du test, en utilisant le décodeur pour combiner la représentation dynamique du mouvement produite par encodeur et A avec la représentation statique du B, comme illustré à la Fig. 15(b)et à la Fig. 16(b).

Et puis, nous décrivons les différentes fonctions de coût utilisées pour entraîner notre réseau, elles sont également illustrées à la Fig. 15(a) et Fig. 16(a). Pour simplifier, nous notons les caractéristiques dynamiques codées par $\tilde{P}_B^i = D(E(P_A^i, L_A), L_B)$.

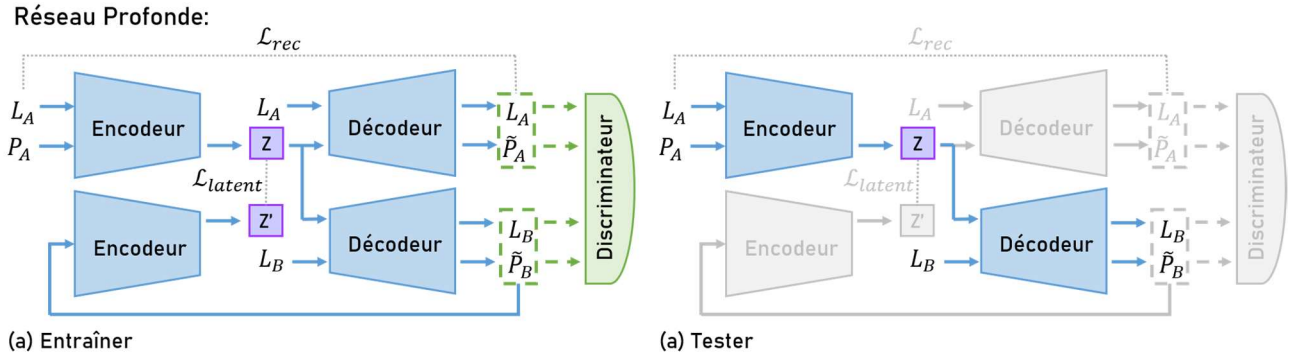


Fig. 15: L'architecture global de **réseau d'encodeur décodeur,** (a) est pour entraîner le modèle, (b) est pour tester le modèle.

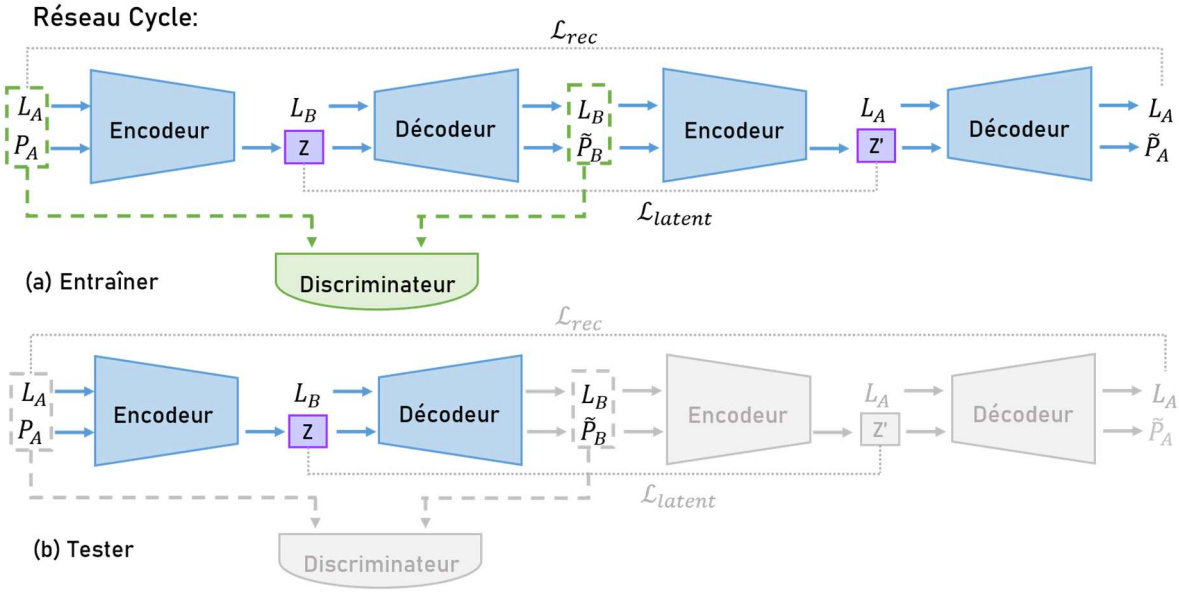


Fig. 16: L'architecture global de **réseau cycle,** (a) est pour entraîner le modèle, (b) est pour tester le modèle.

- **Coût de reconstruction**. Pour former un auto-encodeur (E, D) pour les mouvements, nous utilisons un coût de reconstruction standard sur les positions articulaires :

$$\mathcal{L}_{rec} = \begin{cases} \mathbb{E}_{(L_A, P_A^i) \in \mathcal{M}_A}\left[\left\|D\left(E\left(\tilde{P}_B^i, L_B\right), L_A\right) - P_A^i\right\|\right] & \text{réseau cycle} \\ \mathbb{E}_{(L_A, P_A^i) \in \mathcal{M}_A}\left[\left\|D\left(E\left(P_A^i, L_A\right), L_A\right) - P_A^i\right\|\right] & \text{réseau encodeur décodeur} \end{cases}$$

- **Coût de cohérence latente.** L'intégration d'échantillons de différents domaines dans un espace latent partagé s'est avérée efficace pour les tâches de traduction d'images multimodales [Gonzalez-Garcia et al. 2018 ; Huang et al. 2018]. Des contraintes peuvent être appliquées directement sur cette représentation intermédiaire, facilitant le décodage. Inspirés par cela, nous appliquons un coût de cohérence latente à la représentation partagée pour garantir que le mouvement reciblé P_B^i conserve les mêmes caractéristiques dynamiques que le clip d'origine :

$$\mathcal{L}_{latent} = \mathbb{E}_{(L_A, P_A^i) \in \mathcal{M}_A}\left[\left\|E\left(P_A^i, L_A\right) - E\left(\tilde{P}_B^i, L_B\right)\right\|\right]$$

- **Coût antagoniste.** Étant donné que nos données ne sont pas appariées, le mouvement reciblé n'a aucune vérité de terrain à comparer. Ainsi, nous utilisons un coût contradictoire, où un discriminateur D évalue si l'ensemble temporel décodé des positions P_B^i semble être un mouvement plausible pour le squelette S_B :

$$\mathcal{L}_{ant} = \mathbb{E}_{i \in \mathcal{M}_A}\left[\left\|D\left(\tilde{P}_B^i, L_B\right)\right\|^2\right] + \mathbb{E}_{j \in \mathcal{M}_B}\left[\left\|1 - D\left(P_B^j, L_B\right)\right\|^2\right]$$

  Comme dans d'autres réseaux antagonistes génératifs, le discriminateur D est formé en utilisant les mouvements dans $\mathcal{M}_B$ comme exemples réels, et la sortie de $G^{A \to B}$ comme faux.

Le coût total utilisée pour la formation combine les termes de coût ci-dessus :

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{latent}\mathcal{L}_{latent} + \lambda_{ant}\mathcal{L}_{ant}$$

Où $\lambda_{rec} = 10$, $\lambda_{latent} = 1, \lambda_{ant} = 1$ dans le réseau cycle et $\lambda_{rec} = 10$, $\lambda_{latent} = 4, \lambda_{ant} = 1$ dans le réseau d'encodeur décodeur.

3.4 Expérimentation et Évaluation

3.4.1 Implémentation détaillée

Notre réseau de traitement de mouvement est implémenté dans PyTorch, et les expériences sont réalisées sur un serveur équipé d'un GPU NVIDIA GeForce et un processeur Intel(R) Xeon(R) E5-2603 v3 et les tests sont réalisées sur un PC d'un processeur Intel(R) Core(TM) i7-10710U/1.1GHz (16 Go de RAM). Nous optimisons les paramètres de notre réseau, avec le terme de coût dans la section 3.3.3, en utilisant l'optimiseur Adam.

Afin d'évaluer notre méthode, nous construisons un jeu de données avec 2400 séquences de mouvements, réalisées par 25 personnages distincts, de la collection de personnages Mixamo 3D. Pour chaque mouvement, nous choisissons au hasard un seul personnage pour l'exécuter, pour s'assurer que notre jeu de données ne contient pas de paires de mouvement. De plus, pour permettre la formation du réseau par lots, les mouvements sont découpés en fenêtres temporelles fixes avec T = 64 images chacune. Nous calculons **la longueur de l'armature** et calculons **la position absolue** en utilisant les longueurs d'os et les rotations par axe, et nous les divisons par la hauteur de personnage comme une façons de normalisation. **Le premier est utilisé comme l'information statique et le deuxième est utilisé comme l'information dynamique**. Notre entrainons chaque architecture par 50,000 époques.

3.4.2 Évaluation

Pour bien comparer, nous construirons un ensemble de test par 107 mouvements et 4 personnages. Les mouvements et les personnages n'ont jamais été vus par le réseau pendant l'entraînement. Chaque mouvement est réalisé par 4 personnages. Ainsi, l'ensemble de test contient 428 séquences en total.

Nous présentons une évaluation quantitative de nos méthodes décrites ci-dessus. Les erreurs sont mesurées en effectuant un re-ciblage sur les données appairées dans l'ensemble de test et en les comparant à la vérité terrain, disponible à partir de l'ensemble de données Mixamo d'origine. Nous calculons l'erreur de re-ciblage de chaque squelette k, par rapport à tous les autres dans l'ensemble de test C, comme la distance moyenne entre les positions articulaires, qui est donnée par :

$$E^k = \frac{1}{N \times J \times (|C| - 1)} \sum_{c \in C,\ c \neq k} \sum_{i=1}^{N} \sum_{j=1}^{J} \left\| \tilde{P}^j_{ik,c} - P^j_{ik} \right\|$$

Où $\tilde{P}^j_{ik,c}$ désigne la j-ième position articulaire de la séquence de mouvement reciblée i qui est exécutée par le squelette c et transférée au squelette k, et P^j_{ik} est la vérité terrain. **L'erreur finale est calculée en faisant la moyenne des erreurs E^k pour tous les squelettes $k \in C$.** Nous présentons le résultat dans le Tab. 2.

Tab. 2: L'évaluation de réseau cycle et réseau d'encodeur décodeur

Réseau type	Erreur
Réseau cycle	0.1564
Réseau d'encodeur décodeur	0.085

3.4.3 Analyse

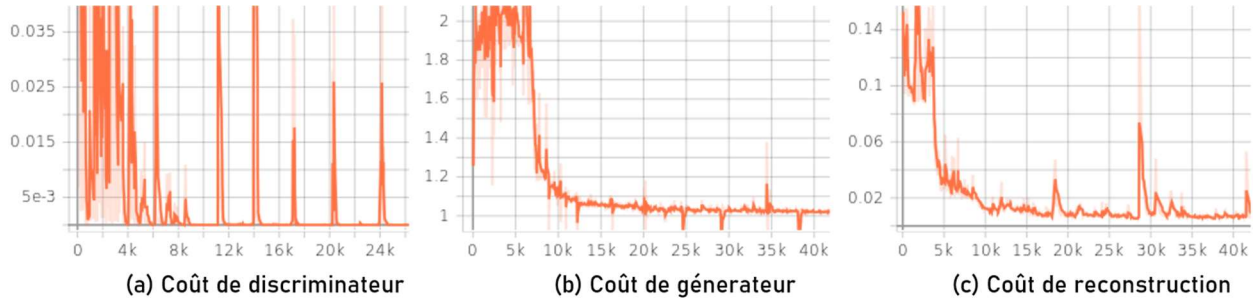


Fig. 17: Le coût pour le **réseau cycle**

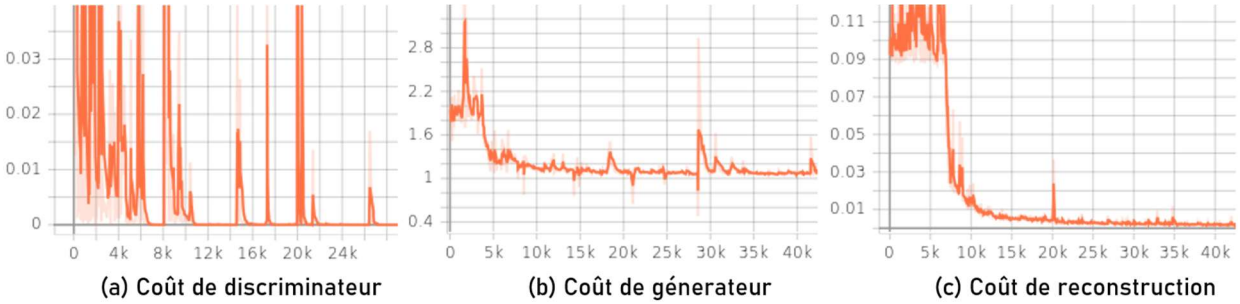


Fig. 18 : Le coût pour le **réseau encodeur décodeur**

Nous présentons le coût évolué pendant l'entraînement dans la Fig. 17 et la Fig. 18. Le générateur de deux réseaux convergent vers une valeur très proche de 1. Mais nous constatons que le coût de reconstruction de réseau cycle converge vers un ordre de grandeur 10^{-2} , celui de réseau d'encodeur décodeur converge vers un ordre de grandeur 10^{-4}. Ce qui peut expliquer une meilleure performance de réseau d'encodeur décodeur dans le Tab. 2, il est évident que le réseau encodeur décodeur apprendre plus d'information viens des données. Et puis, nous choisissons une séquence dans l'ensemble de test et nous traçons un frame pour bien comparer et analyser. (Fig. 19 et Fig. 20)

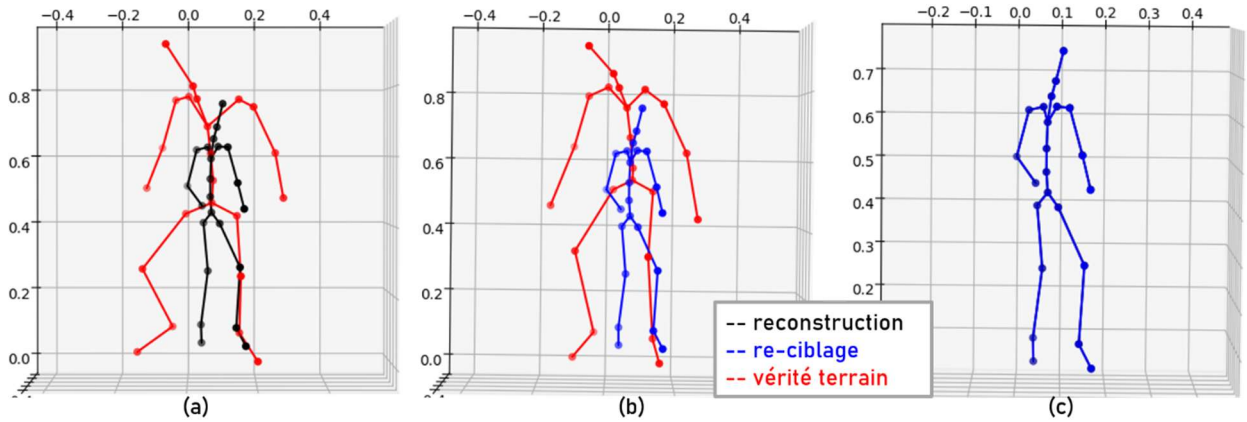


Fig. 19: Le **réseau cycle** (a) comparaison entre reconstruction et sa vérité terrain. (b) comparaison entre re-ciblage et sa vérité terrain. (c) comparaison entre la reconstruction et le re-ciblage.

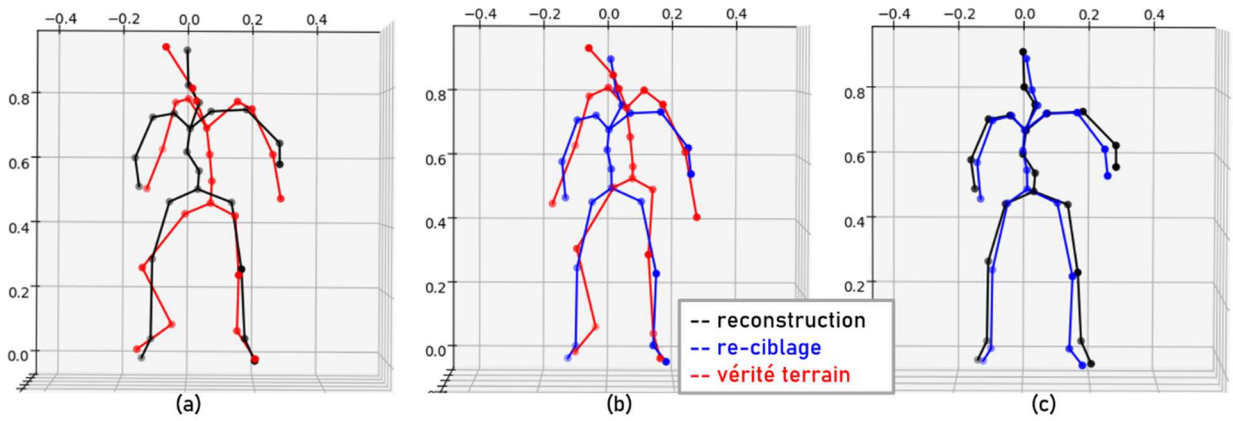


Fig. 20: Le **réseau d'encodeur décodeur** (a) comparaison entre reconstruction et sa vérité terrain. (b) comparaison entre re-ciblage et sa vérité terrain. (c) comparaison entre la reconstruction et le re-ciblage.

Avec les visualisations des squelettes, nous constatons que **le réseau encodeur décodeur performe mieux que le réseau cycle**, il peut construire l'architecture globale pour un personnage, et nous pouvons voir une différence entre la reconstruction et le re-ciblage, c'est-à-dire que le décodeur considère les données statiques pour construire un squelette d'humain. Mais pour le réseau cycle, il y a une superposition entre la reconstruction et le re-ciblage, c'est-à-dire que le décodeur ne pris jamais en compte les informations sur les données statiques. La raison pour la différence est, afin d'obtenir la reconstruction finale, les données doivent passer deux encodeurs et deux décodeurs dans le réseau cycle mais ils ne passent qu'un encodeur et un décodeur dans le réseau encodeur décodeur. Ce qui donne plus de difficultés pour le réseau cycle d'apprendre les paramètres en utilisant la même fonction de coût.

Au contraire, même si le résultat vient du réseau encodeur et décodeur, il n'est pas satisfaisant comme prévu. Nous constatons que le coût de générateur ne converge pas ver 0, et le coût de discriminateur converge toujours vers 1. C'est-à-dire que même si dans le réseau d'encodeur-décodeur, le générateur n'est pas bien entrainé, le mouvement viens de générateur ne peut pas tromper le discriminateur. **Nous pensons que les réseaux ont un gros problème de prendre en compte des données statiques pour prédire. Dans l'étape suivante, Il faut que nous continuions à améliorer le générateur.**

Nous présentons un re-ciblage de mouvement obtenue par la méthode dans l'article [Aberman et al., 2020] (la Fig. 21)

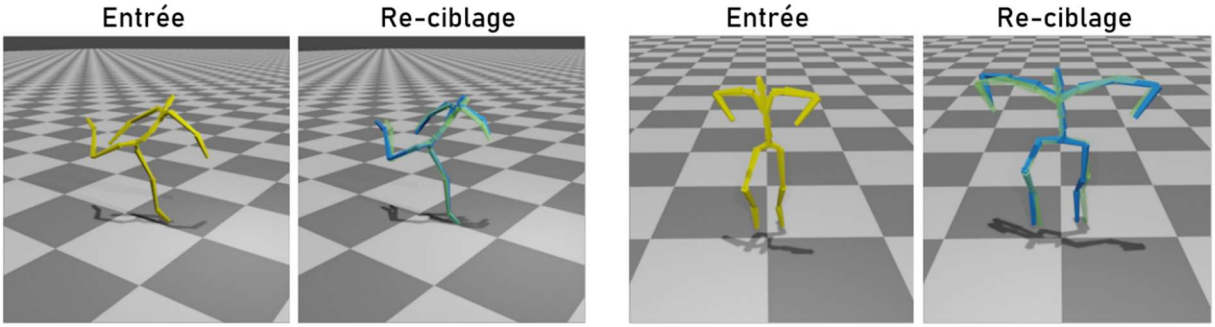


Fig. 21: Le re-ciblage dans l'article [Aberman et al., 2020]. Le jaune est l'entrée, le vert est la vérité terrain, le bleu est le résultat de re-ciblage

Dans leur méthode, ils utilisent les **poses par défaut des personnages** comme les caractéristiques statiques et les **quaternions de chaque articulation (relatif à la pose par défaut)** comme les caractéristiques dynamiques, et ils sont donnés au réseau comme entrée. Et les sorties de son réseau sont des quaternions. Après ils recalculent la position globale par la cinématique directe en utilisant les pose par défaut qui est déjà connus et les quaternions obtenus.

Nous pensons qu'une tellement de différence de performances entre notre méthode et [Aberman et al., 2020], c'est parce que si nous utilisons les positions comme entrée, le réseau doit changer beaucoup entre l'entrée et sortie, mais si nous utilisons les quaternions comme entrée, comme les décalages sont déjà connus, le réseau neurone n'a que besoin de régler finement sur les quaternions. Il est évident que la deuxième opération est beaucoup plus facile que le premier pour le réseau à apprendre.

Mais en réalité, En analysant le réseau pré-entraîné fourni dans leur article, nous observons que le décodeur appris n'utilise pas les données statiques pour prédire les rotations en sortie. En réalité, leur encodeur décodeur réalise une fonction identité, et leurs résultats visuels sont dus au fait qu'ils utilisent la cinématique directe avec les données statiques du squelette B pour leur reconstruction. **Leur modèle a le même problème que nos.**

3.5 Discussion

3.5.1 Modification sur le modèle

Nous modifions le modèle beaucoup pour qu'ils ont une structure comme nous avons présenté. En premier temps, l'encoder et le décodeur ont une couche de 5 et ont presque la même structure (voir la Fig. 25 dans l'annexe).

- Changement 1 : Tout d'abord, la fonction d'activation que nous utilisons est la ReLU même si après la dernière couche de l'encodeur, le décodeur et le discriminateur, cette fonction transfert tous les valeurs négatives aux constant 0. Alors, pour la sortie du décodeur ou la sortie du discriminateur, il faut que les valeurs puissent être négatives. En considérant cette raison, **nous choisissons la LeakyReLU au lieu de ReLU**. Une évolution de coût est présentée dans la Fig. 21, nous constatons qu'avec l'activation ReLU, le coût converge vers un ordre de grandeur 10^{-2} , mais le coût converge vers 10^{-4} , en utilisant l'activation LeakyReLU.

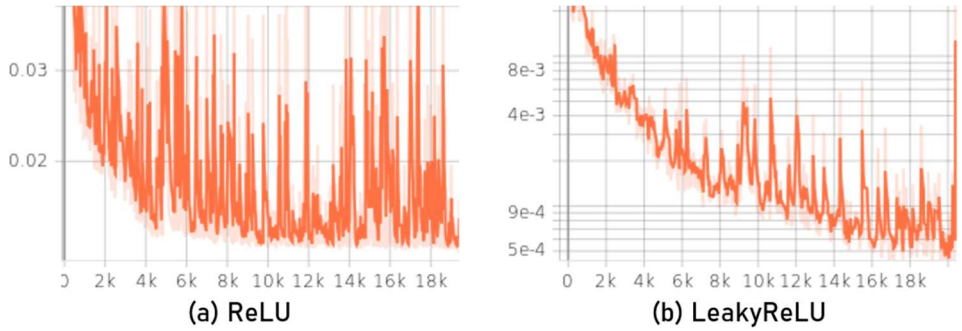


Fig. 22: Évolution de coût (a) la fonction d'activation ReLU. (b) la fonction d'activation LeakyReLU.

- Changement 2 : Après, nous relance le modèle beaucoup de fois et nous trouvons que le coût de discriminateur évolue toujours vers 0, mais le coût de générateur évolue toujours 1. Nous pensons que c'est parce que le discriminateur est beaucoup plus fort que le générateur. Donc **nous modifions la façon de mettre à jour, une mise à jour de discriminateur cinq fois moins fréquente que le générateur**.
- Changement 3 : Et puis, nous constatons aussi les coûts de générateur, nous trouvons qu'il évolue toujours vers 1, nous pensons que peut-être notre réseau est trop profond, le décodeur n'est pas capable de décodeur les informations dans l'espace latent, donc **nous enlevons une couche de GNN et une couche de convolution dans l'encodeur et le décodeur**. Nous pensons que le réseau moins profond pourrait mieux converger.
- Changement 4 : Après ce change, nous trouvons que la reconstruction et le re-ciblage sont encore superposés, ce qui nous confusions beaucoup, après avoir vérifié le réseau, nous pensons que c'est le décodeur qui a plus de problème. Les données que nous donnons aux décodeurs sont l'espace latent et les longueurs des armatures. Mais les sorties de décodeur sont les mêmes, c'est parce que le décodeur ne considère pas les décalages. Pour améliorer cette situation, **nous changeons la première couche de ConvTranspose1D par la couche de Upsample. Et nous initialisons les paramètres des arrêts dans le GNN dix fois plus grande qu'avant.** Toutes les opérations ont un objectif d'augmenter les influences sur les décalages pour le décodeur.

Nous choisissons le réseau d'encodeur décodeur comme exemple. Après chaque modification, nous relance l'entrainement, et nous notons les performances de l'évaluation. Les performances sont dans le Tab. 1Tab. 3.

Tab. 3: L'évaluation pour toutes les modifications sur le réseau d'encodeur décodeur

État	Erreur
Au début	1.8674
Changement 1 : ReLU par LeakyReLU	0.1291
Changement 2 : la façon de mis à jour	0.0894
Changement 3 : Minimise couche	0.0723
Changement 4	0.085

3.5.2 Travail au futur

Nous pensons qu'il y aura deux façons pour améliorer notre réseau, soit nous changeons la forme d'entrée qui est les quaternions au lieu de positions, soit nous utilisons **les données appairés** pour entrainer le réseau. Nous vérifions un peu la deuxième méthode en construisant un encodeur et décodeur présenté dans la Fig. 23.

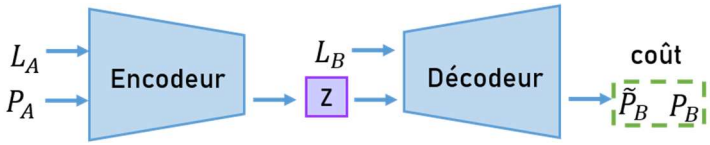


Fig. 23: Un petit réseau

Nous entraînons ce réseau sur un base de données de cent trajectoires présentées par quatre personnages. Après, nous testons sur ① des mouvements inconnus, des personnages connues, ② des mouvements inconnus, des personnages inconnues, ③ des mouvements connus, des personnages inconnues pendant l'entraînement. Les performances est présenté dans le Tab. 4 en utilisant le critère d'évaluation dans la section 3.4.2.

Tab. 4: L'évaluation de petit réseau cycle

Réseau Petit	Erreur
① Squelette vu, mouvement non vu	0.0293
② Squelette non vu, mouvement non vu	0.0398
③ Squelette non vu, mouvement vu	0.0421

Nous traçons quelques frames pour le test ① dans la Fig. 24.

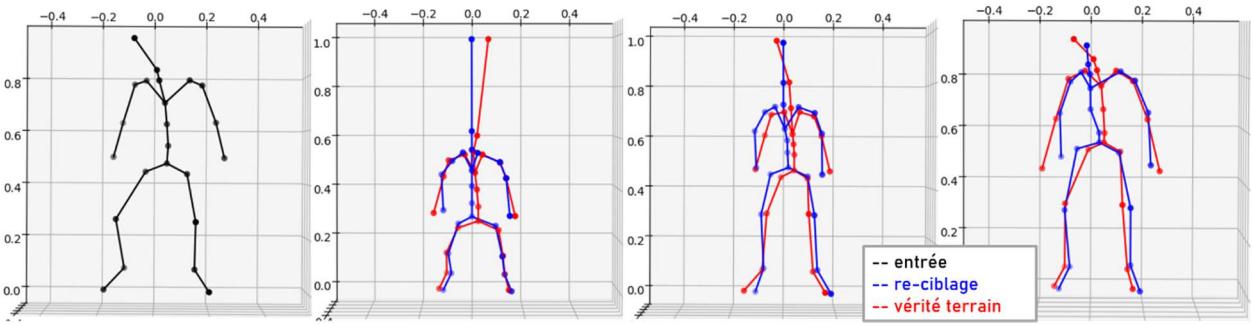


Fig. 24: Le résultat de re-ciblage avec donnée appairée

Nous constatons qu'avec des données appairées, le re-ciblage est beaucoup plus mieux que notre réseau. Et le décodeur prend en compte bien des données statiques. Mais d'après le Tab. 4, nous trouvons que la performance pour les squelettes non vus est moins bien que les squelettes vus, c'est-à-dire qu'il faut ajouter plus de type de squelettes dans l'ensemble d'entraînement pour améliorer la capacité de génération au futur.

Au futur, nous pourrons travailler plus sur cette direction.

Chapitre 4 Conclusion

L'objectif de notre travail est de détecter et imiter des mouvements humains par le robot Poppy. Et nous faisons le travail en deux parties:

Dans la première partie, nous cherchons une librairie de détection de pose – Blazepose, et traiter les vidéos de mouvement humain données par Mai. Et puis, nous comparons la pose détectée par les librairies Blazepose, Kinect, OpenPose et Vicon et nous pensons que la librairie Kinect est la plus adaptée aux taches de re-ciblage de mouvement humain à robot.

Dans la deuxième partie, nous concevons notre modèle de re-ciblage de mouvement avec les données non-appairées, nous proposons les opérateurs GNN pour traiter la structure squelettique, et puis nous l'utilisons pour construisons la partie principale du réseau, un encodeur, un décodeur et un discriminateur. Après nous utilisons trois parties et construisons deux architectures : un architecture cycle et une architecture d'encodeur décodeur. Après avoir évalué les deux réseaux, nous pensons que le réseau encodeur décodeur performe mieux que le réseau cycle mais le générateur dans le réseau encodeur décodeur est encore mauvais. Nous expliquons la raison pour laquelle qu'ils ne fonctionnent pas bien. En fin, nous proposons que les travaux au futur soient nous utilisons les données appairées, soient nous utilisons les quaternions comme entrées.

Bibliographie

Article de périodique

- Devanne, M., & Sao Nguyen, M. (2018). Generating shared latent variables for robots to imitate human movements and understand their physical limitations. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops* (pp. 0-0).
- Choi, S., Song, M. J., Ahn, H., & Kim, J. (2021, May). Self-supervised motion retargeting with safety guarantee. In *2021 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 8097-8103). IEEE.
- Aberman, K., Li, P., Lischinski, D., Sorkine-Hornung, O., Cohen-Or, D., & Chen, B. (2020). Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (TOG)*, *39*(4), 62-1.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2794-2802).
- Nair, A., Chen, D., Agrawal, P., Isola, P., Abbeel, P., Malik, J., & Levine, S. (2017, May). Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE international conference on robotics and automation (ICRA)* (pp. 2146-

2153). IEEE.

- Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., ... & Brain, G. (2018, May). Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 1134-1141). IEEE.
- Pavse, B. S., Torabi, F., Hanna, J., Warnell, G., & Stone, P. (2020). Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration. *IEEE Robotics and Automation Letters*, *5*(4), 6262-6269.

Article électronique

- Torabi, F., Warnell, G., & Stone, P. (2019). Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*.
- Xiang, S., & Li, H. (2020). Revisiting the continuity of rotation representations in neural networks. arXiv preprint arXiv:2006.06234.

Site web ou blog

- Google. Mediapipe[en ligne]. (2018). URL : https://google.github.io/mediapipe/solutions/pose.html
- Mixamo 3D, Adobe Systems Inc. [en ligne]. (2018) URL: https://www.mixamo.com.

Annexes

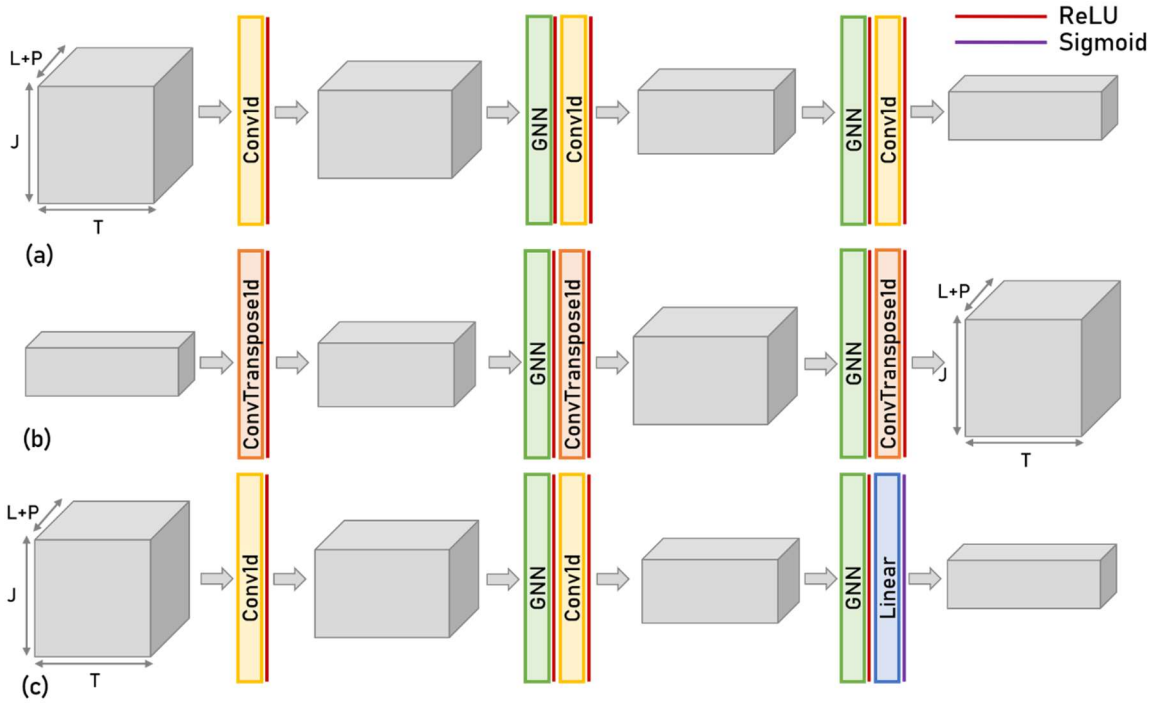


Fig. 25: (a) La structure de l'encodeur (b) La structure du décodeur (c) La structure du discriminateur