Reconnaissance de gestes, expressions à partir de capture de mouvement temps réel

Stanislas BAGNOL, Mathis SPATARO

Mai 2023

Résumé : Etude de la faisabilité de l'ajout de la fonctionnalité de capture de mouvement en temps réel au projet de recherche PyDeepAn. Ce dernier se consacre à la reconnaissance d'expression à partir de la posture humaine en utilisant un réseau de neurones de Deep Learning.

Mots-clés : Deep Learning, Capture de mouvements, expressions, gestes, MediaPipe, HPA, PyDeepAn, temps réel, python.

1. Introduction

Dans le cadre de notre première année de master Informatique et de l'UE d'Ouverture à la Recherche, nous avons travaillé sur le projet de recherche de l'équipe SAARA (Simulation, Analyse et Animation pour la Réalité Augmentée) du laboratoire LIRIS, PyDeePan, auquel notre encadrant Alexandre Meyer participe. Le projet PyDeepAn travaille sur la reconnaissance d'expression à partir de la posture d'un sujet. Notre sujet est l'ajout de la fonctionnalité de capture de mouvement en temps réel pour la reconnaissance d'expression, il fait suite aux thèses d'Arthur Crenn "Reconnaissance d'expressions corporelles dans des mouvements de personnes en vue de la synthèse de style" [ext03] et Mehdi-Antoine Mahfoudi "Génération procédurale d'animations porteuses d'expressions : approche temps réel et interactive" [ext02] ainsi qu'au rapport de projet d'ouverture à la recherche de Thomas Montero [ext01].

2. Contexte

La reconnaissance d'expressions dans des postures est utilisée comme seconde entrée dans la reconnaissance des émotions via l'expression faciale et des postures. L'étude des mouvements du corps s'inscrit dans le domaine de l'analyse du comportement humain qui est réalisée dans des domaines tels que la psychologie, la sociologie ou la médecine. Elle peut aider à détecter des pathologies en comprenant mieux les signaux non-verbaux, les attitudes et les émotions exprimées par le langage corporel.

Pour l'équipe SAARA, la reconnaissance d'expression est utilisée dans le but de mieux concevoir des techniques d'animation. Elle permet de capturer dans les mouvements du corps les expressions des acteurs pour les transférer aux personnages numériques, créant ainsi une expérience réaliste et immersive. Il existe beaucoup d'autres applications possibles. Elle peut par exemple servir à développer des interfaces utilisateur basées sur les gestes et les mouvements du corps ou encore dans le domaine de la santé et de la réadaptation physique où elle permettrait de suivre et d'analyser les mouvements du corps des patients pendant les exercices de rééducation, aidant ainsi à évaluer leur progression et à obtenir des retours en temps réel.

PyDeepAn est donc un projet de reconnaissance d'expression (Expression Recognition) à partir des mouvements du corps d'un sujet. La librairie est écrite en python orienté objet et s'appuie sur le Deep Learning pour classifier les expressions. On entend par expression une émotion, un état physique, défini par un ou plusieurs adjectifs correspondants à des caractéristiques tangibles d'un individu. La position du corps est représentée au format HPA (Holden Position Animation). Le format HPA regroupe un ensemble de 22 points caractéristiques en trois dimensions, principalement situés sur les articulations.

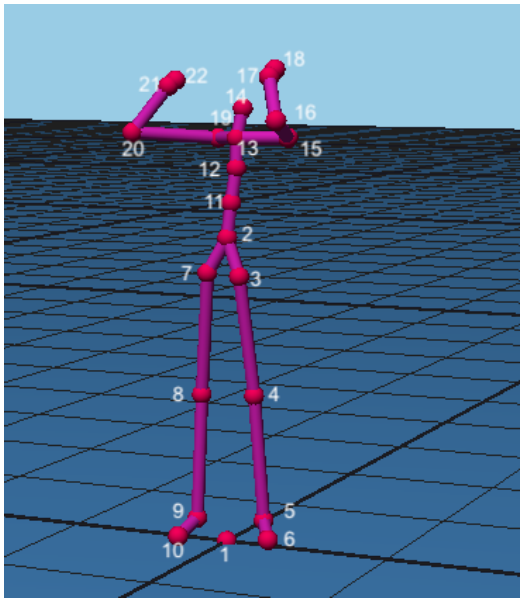

Figure 1: Exemple de squelette au format HPA.

Il inclut également la vitesse de déplacement sur l'axe X et Z, la vitesse angulaire de rotation autour de l'axe Y, ainsi que les positions sur les axes X et Z du contact au sol de chaque pied. Un mouvement du corps, ou animation, est une suite de positions du squelette qui, affichées les unes après les autres, forment une animation.

Le réseau neuronal de PyDeepAn a été entraîné sur un grand nombre d'animations labellisées avec la caractéristique associée. Il existe des modèles pré-entraînés sur différents dataset, reconnaissant différentes expressions, plus ou moins adaptées selon les besoins de l'utilisateur.

dataset	**expressions reconnues**
styletransfert	'angry', 'childlike', 'depressed', 'neutral', 'old', 'proud', 'sexy', 'strutting'
MPI [data02]	'Joy', 'Pride', 'Surprise', 'Sadness', 'Relief', 'Anger', 'Fear', 'Neutral', 'Disgust', 'Amusement', 'Shame'
Emilya [data03]	'Anxiety', 'Pride', 'Joy', 'Sadness', 'Panic Fear', 'Shame', 'Anger', 'Neutral'

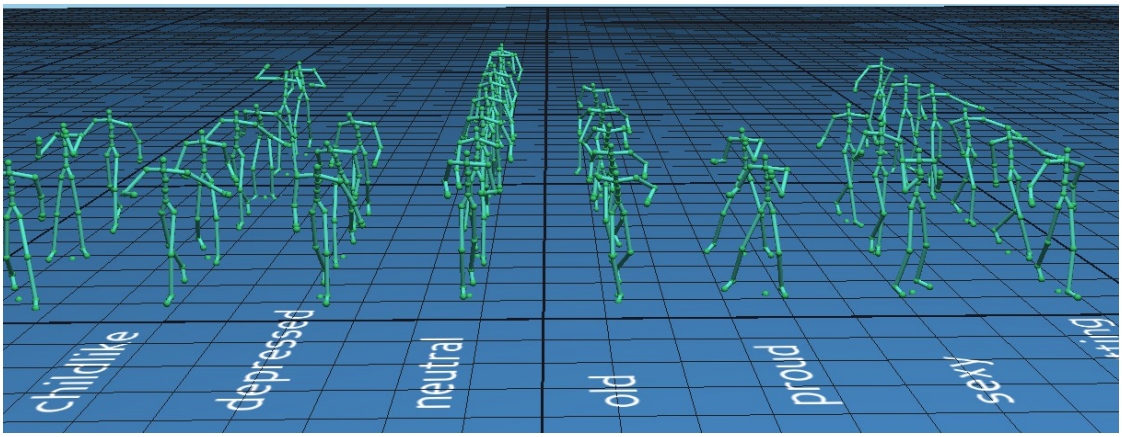

Figure 2: Classification d'un lot d'animations.

La librairie est également dotée d'un visualiseur d'animations s'appuyant sur la librairie Panda3D.

3. Problématiques

L'objectif de notre contribution est d'ajouter la possibilité de faire de la reconnaissance d'expression à partir d'un flux vidéo en temps réel. Pour cela, nous devons utiliser un système de capture de mouvement ("motion capture" ou "MoCap" en anglais) afin de détecter la position d'un squelette dans une image. Nous pouvons ainsi construire une animation de squelettes et la donner en entrée du réseau de neurones pour prédire une expression.

Le travail consiste à intégrer le système de MoCap, collecter les données, et les préparer afin qu'elles respectent le format de squelette utilisé. Notre contribution peut s'apparenter à une étude de faisabilité puisque nous n'avons aucune garantie de fonctionnement. L'ajout d'une telle fonctionnalité permettrait d'élargir le champ d'utilisation de PyDeepAn.

4. Démarche

4.1 Système de capture de mouvement

Initialement, nous devions comparer plusieurs systèmes de capture de mouvements, MediaPipe Pose [MoCap02], OpenPose [MoCap01] et MMPose [MoCap03] afin de déterminer lequel était le plus adapté à nos besoins. Cependant, nous nous sommes rapidement rendu compte, sous les conseils de notre encadrant, que MediaPipe Pose était plus adapté.

Dans un premier temps, notre démarche a été de tester le fonctionnement de MediaPipe Pose et d'étudier les points caractéristiques retenus. Le système de capture de mouvement fonctionne aussi bien avec des vidéos sous forme de fichiers ou via un flux vidéo en direct, provenant d'une webcam par exemple. Nous nous avons d'abord testé des fichiers vidéos, car plus simple à effectuer que de se lever et se mettre à bonne distance de la webcam pour que le corps soit entièrement visible.

4.2 Familiarisation avec les sources

Afin d'intégrer au mieux notre extension à PyDeepAn, nous avons dû passer par une étape de prise en main du code afin de comprendre l'architecture, le fonctionnement et les dépendances entre les différentes classes, d'autant plus que nous ne sommes pas des spécialistes du Python Orienté Objet. Cette partie du travail a été cruciale et nous a permis de gagner du temps par la suite.

4.3 Intégration de MediaPipe et conversion squelette MediaPipe vers HPA

Dans un premier temps, il a fallu intégrer MediaPipe Pose dans PyDeepAn et commencer à faire une conversion simple du format utilisé par MediaPipe vers le format HPA afin de pouvoir visualiser des premiers résultats. En effet, le format HPA utilise 22 points d'ancrage pour modéliser un squelette tandis que MediaPipe Pose en utilise 32. De plus, certains points utilisés par le format HPA ne sont pas recensés par le format MediaPipe.

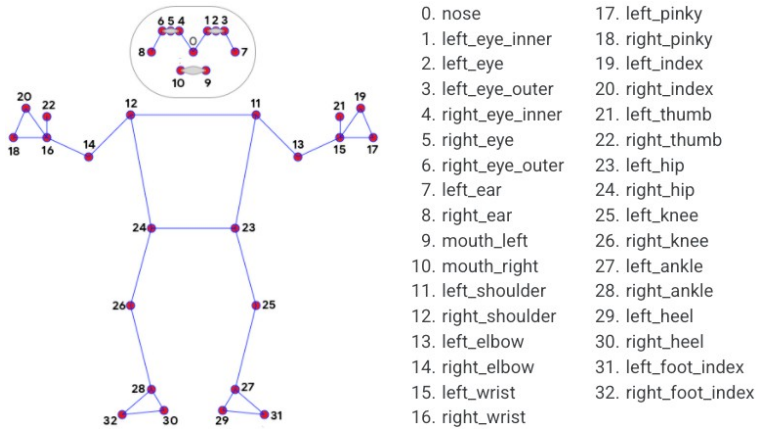

Figure 3: Représentation d'un squelette au format MediaPipe Pose.

[media02]

Ces premiers résultats visuels nous ont alors permis de nous guider vers les points d'améliorations possibles. Le convertisseur initial utilisait essentiellement des interpolations linéaires entre 2 points pour estimer la position des points d'ancrage manquants, supprimer les points supplémentaires et stocker le tout dans l'ordre défini par le format HPA.

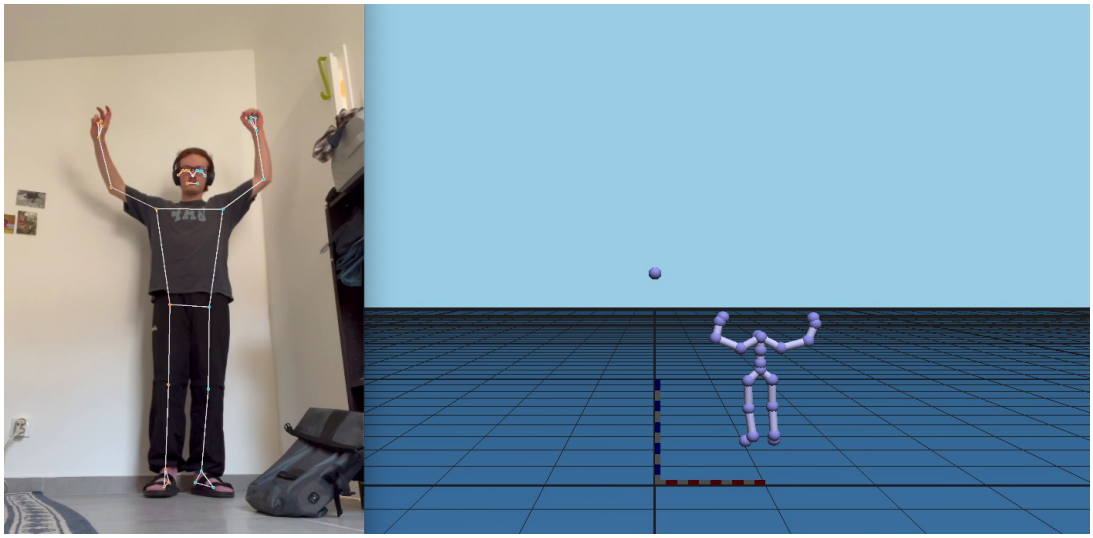

Figure 4: Exemple de capture de mouvement en temps réel.

Nous avons alors travaillé sur les proportions qui étaient manifestement incorrectes et avons procédé à une inversion du squelette sur l'axe Z (profondeur).

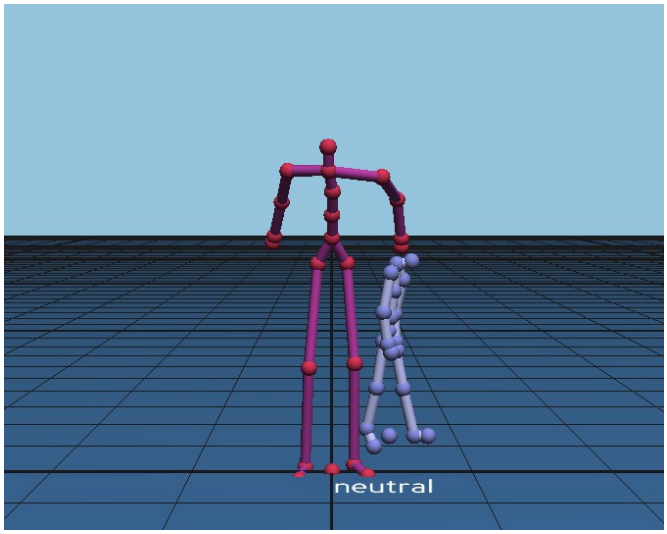

Figure 5: Exemple de squelette non normalisé (en bleu) à côté d'un squelette de référence (en rouge).

Une fois les proportions rectifiées, nous avons affiché un squelette issu du dataset d'entraînement (en rouge ici) pour faire apparaître les différences entre les squelettes. Une mise à l'échelle et un centrage des données furent alors appliqués afin de reformater les squelettes obtenus de la même façon que les données d'entraînement et les rendre ainsi traitables par apprentissage machine.

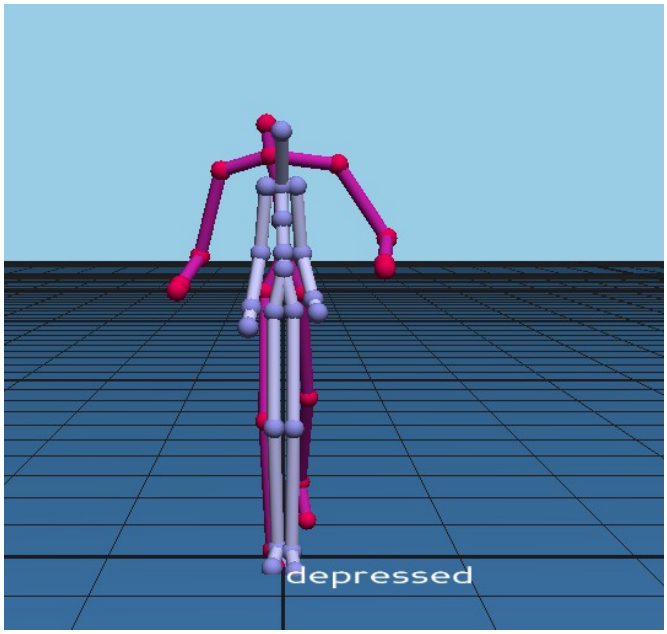

Figure 6: Squelette de la figure 5 normalisé (en bleu), devant un squelette de référence (en rouge).

4.4 Les données supplémentaire dans HPA

Le réseau neuronal utilise également la vitesse de déplacement, la vitesse angulaire, et la position du contact au sol de chaque pied, données très difficiles à détecter dans le cadre d'une capture de mouvement par vidéo, que nous avons finalement choisi de ne pas utiliser.

4.5 Collection et sauvegarde des données en temps réel

Le réseau de neurones de Pydeepan prend en entrée une animation de 240 squelettes. Il faut alors stocker ces données afin de les passer en entrée du réseau de neurones. Notre procédé consiste à évaluer périodiquement une animation formée des 240 dernières frames de l'animation détectée dans le flux vidéo et d'enregistrer le résultat. Nous avons choisi de faire une prédiction par seconde, en ajoutant les squelettes en fin de tableau jusqu'à obtenir 240 frames, faire une prédiction et supprimer les *n* premiers squelette, où *n* est le nombre d'image par seconde capturées par la caméra. Et ainsi de suite.

4.6 Préparation des données

Nous nous sommes inspirés des script de démo pour s'assurer de préparer les données exactement de la même manière que celles sur lesquelles les différents modèles ont été entraînés. Cela consiste d'abord à normaliser les données. Il faut ensuite les convertir en tensor ("tenseur" en français) de flottants capables d'être calculés avec CUDA (Compute Unified Device Architecture), permettant le calcul sur GPU Nvidia. Le calcul sur GPU est particulièrement efficace pour l'apprentissage profond puisqu'il nécessite un nombre massif de calculs parallèles.

Nous passons ensuite nos tensor dans un AutoEncoder qui permet de compresser nos données, en extrayant les caractéristiques les plus importantes de celles-ci. La dernière étape de la préparation des données consiste à calculer la matrice de Gram de notre tensor compressé. Une matrice de Gram, également appelée matrice de covariance gramienne, est une matrice carrée symétrique générée à partir d'un ensemble de vecteurs (nos tensors dans notre cas). Cette matrice est souvent utilisée dans le domaine de l'apprentissage automatique pour extraire des informations sur les relations entre les vecteurs d'un ensemble.

4.7 Prédiction

C'est la matrice de Gram que nous donnons en entrée du réseau de neurones. Ce dernier nous retourne un entier que nous pouvons utiliser pour retrouver le label de l'expression associée stocké dans un tableau. L'entier prédit représente l'indice du label dans ce tableau.

4.8 Test

L'enjeu était de réussir à convertir le format d'un squelette en conservant suffisamment d'informations sur le mouvement pour que le réseau de neurones puisse fonctionner.

N'ayant pas trouvé de jeu d'animations labellisé au format MédiaPipe, nous avons cherché sur internet des vidéos d'humains en mouvement qui pourraient convenir. Cependant les cadrages des vidéos donnaient souvent des squelettes partiels, et les caractéristiques associées aux mouvements étaient issues de notre appréciation personnelle. Pour ces raisons, nous avons choisi de nous rabattre sur des vidéos d'un acteur imitant des manières de marcher selon une caractéristique précise et souvent exagérée ("énervé" ou "vieux" par exemple). [media01]

Figure 7: Acteur imitant une démarche énervée.

Figure 8: Acteur imitant une personne âgée

Il est important de noter que ce jeu de données est trop petit pour mesurer correctement la viabilité de la conversion et qu'un acteur n'est peut-être pas capable d'imiter un mouvement avec assez de précision pour influencer un tel algorithme.

5. Résultats

Pour déterminer la caractéristique à associer à une animation, PyDeepAn compare le mouvement d'un squelette avec un squelette "neutre", représentant un mouvement sans émotions et issu de l'apprentissage machine. L'important est donc de conserver ces déviations par rapport au squelette neutre au cours du changement de format.

Les informations telles que la vitesse de déplacement, la vitesse angulaire, et la position du contact au sol de chaque pied étant difficile voir impossible à obtenir via MediaPipe. Par exemple, comment connaître la vitesse de déplacement sachant que la caméra peut elle aussi être en mouvement ? Comment déterminer la position du contact des pieds sur le sol n'ayant pas l'information de ce qu'est le sol ? Nous avons donc comparé la précision du réseau neuronal entraîné à partir des mêmes données d'entraînement, avec et sans ces informations. La non-utilisation de ces données faisant passer la précision de l'algorithme de 97% à 92% au cours de notre expérience, nous avons donc choisi de ne pas les utiliser et de rester à 92% de précision.

Une autre problématique que nous avons tenté de résoudre est la différence d'alignement entre les données d'entraînement et la MoCap. Dans les jeux de données, les squelettes étaient parfaitement alignés face à la caméra alors qu'avec la capture de mouvement, il se peut que le squelette soit de biais. Nous avons tenté de rectifier l'orientation du squelette mathématiquement. Nous avons calculé l'angle entre la droite reliant les épaules ou le bassin et l'axe X. Une fois cet angle obtenu nous avons pu appliquer une rotation de chaque point du squelette sur l'axe Y à l'aide d'une matrice de rotation. Cependant, cette rotation détériore les proportions du squelette en ne conservant pas la même posture, faussant les résultats. De plus, il arrive que le bassin et les épaules ne soient pas alignés et nous n'avons pas trouvé de critère permettant de choisir entre l'orientation de l'un et de l'autre.

Un squelette de biais implique également qu'une partie du corps peut être masquée par le corps du sujet lui-même, un bras par exemple, engendrant une perte d'information. Nos travaux se concentrent sur les cas où le sujet était entièrement dans le champ de la caméra, négligeant des cas spécifiques où ce ne serait pas le cas. Sachant que pour chaque point d'ancrage, MediaPipe Pose donne également le taux de visibilité du sujet, nous aurions pu faire en sorte d'annuler la prédiction tant que tous les points n'avaient pas un taux de visibilité d'au moins 90% par exemple. Nous avons plutôt privilégié le fonctionnement dans le cas général plutôt que de gérer tous les cas particuliers.

De plus, nous avons eu des difficultés à trouver des vidéos de tests sur des sujets ayant une expression significative et ainsi vérifier la détection correcte d'une expression. Notre système de reconnaissance d'expression via capture de mouvement fonctionne, il prédit bien des valeurs mais leur conformité est difficile à établir sans jeu de données au format MediaPipe puisque même un humain peut avoir du mal à reconnaître une expression de par la subjectivité de l'exercice.

6. Conclusion

Cette première expérience dans le domaine de la recherche a été enrichissante pour nous et nous a permis de découvrir le travail sur un projet sans garantie de résultat. En effet, avec les moyens que nous avons pu mettre en œuvre et notre expérience nous n'avons pas réussi à répondre définitivement à la problématique, nous avons néanmoins pu nous approcher de la réponse tout en ayant des pistes pour tenter d'améliorer les résultats qui résident notamment dans l'orientation du squelette et le manque de données pour tester et valider nos résultats. En effet, si nous avions eu en notre possession une large banque de vidéos labellisées d'acteur (ou non) qui expriment manifestement une expression alignée face caméra, nous aurions pu répondre quant à la conformité de notre implémentation et ainsi répondre partiellement à propos de la faisabilité de l'extension souhaitée.

Ce projet nous a permis d'approfondir nos connaissances dans le domaine du traitement d'image et de l'apprentissage profond dans le cadre de la capture de mouvement et de la prédiction d'expression.

7. Références

[ext01] Thomas Montero, « POM : mocapreco pyDeepan », consulté le 02/06/23, à l'adresse : https://forge.univ-lyon1.fr/p1819506/bagnol-spataro-rendu-ouverture-recherche-m1/-/blob/main/Rapport_POM_MONTERO.pdf

[ext02] Mehdi-Antoine Mahfoudi, «Génération procédurale d'animations porteuses d'expressions : approche temps réel et interactive », consulté le 02/06/23, à l'adresse : https://liris.cnrs.fr/these/these-mehdi-antoine-mahfoudi

[ext03] Arthur Crenn, « Reconnaissance d'expressions corporelles dans des mouvements de personnes en vue de la synthèse de style », consulté le 02/06/23, à l'adresse : https://liris.cnrs.fr/these/these-arthur-crenn

[media01] 3h4k, « 100 Different Ways to Walk by Kevin Parry », consulté le 02/06/23, à l'adresse : https://www.youtube.com/watch?v=cVjIqr8CTtQ

[media02] Illustration du format de squelette MediaPipe, consultée le 02/06/23, à l'adresse : https://developers.google.com/static/mediapipe/images/solutions/pose_landmark_index.png

[data01] Sarah Ribet, Hazem Wannous, Jean-Philippe Vandeborre, « Survey on Style in 3D Human Body Motion: Taxonomy, Data, Recognition and its Applications », IEEE Transactions on Affective Computing, consulté le 02/06/23, à l'adresse : https://hal.archives-ouvertes.fr/hal-02420912/document

[data02] MPI « Human Pose » dataset, consulté le 02/06/23, à l'adresse : http://human-pose.mpi-inf.mpg.de/

[data03] Nesrine Fourati, Catherine Pelachaud, «Emilya: Emotional body expression in daily actions database », consultée le 02/06/23, à l'adresse : http://www.lrec-conf.org/proceedings/lrec2014/pdf/334_Paper.pdf

[MoCap01] Dépôt Github d'Open-pose, consulté le 02/06/23, à l'adresse : https://github.com/CMU-Perceptual-Computing-Lab/openpose

[MoCap02] Dépôt Github de Mediapipe Pose, consulté le 02/06/23, à l'adresse : https://google.github.io/mediapipe/solutions/pose

[MoCap03] Dépôt Github de MMPose, consulté le 02/06/23, à l'adresse : https://github.com/open-mmlab/mmpose