VISUALISATION DE MOUVEMENTS 3D ACQUIS PAR MOTION CAPTURE

DAUJAT BENJAMIN 11411297
FAUROBERT EMERIC 11508845

Résumé :

Ce document présente le travail effectué dans le cadre de notre projet d'orientation en master, réalisé en collaboration avec l'Université Claude Bernard et l'école Centrale de Lyon. Notre sujet consistait à mettre en place une interface de visualisation de mouvements acquis par Motion capture avec la plateforme AmigoCap. La visualisation de ces données était jusqu'alors réalisée via une application très limitée réalisée en python. Nous avons choisi de concevoir une interface plus complète mais restant simple d'utilisation. Ce projet contient également nos efforts de recherches préalables et de documentation.

Mots clés :

Motion capture, mocap, mouvements, web, navigateur, javascript, html.

Introduction :

Ce document présente le travail que nous avons effectué dans le cadre de notre projet d'orientation en master. Il fût réalisé en collaboration avec l'Université Claude Bernard et l'école Centrale de Lyon. Notre sujet consistait à créer une interface de visualisation de mouvements acquis par Motion capture. Tout au long de ce projet nous avons été encadrés par Romain Vuillemot, membre du LIRIS, professeur à l'école Centrale et co-responsable de la plateforme AmigoCap. Pour mener à bien ce projet nous avons choisi de concevoir une interface web la plus simple possible. Pour cela nous avons décidé d'utiliser la bibliothèque JavaScript Three.js. Nous avons pris un soin tout particulier pendant la réalisation de ce projet à bien documenter nos recherches ainsi que notre code source afin qu'il puisse être réutilisé / adapté aussi facilement que possible par la suite.

Travail de recherche :

A. Généralités sur la motion capture

Qu'elles soient utilisées dans les jeux vidéo, l'industrie cinématographique ou encore le milieu médical, les animations de personnages virtuels sont de plus en plus réalistes. Ceci est notamment dû à un procédé s'appelant « motion capture » (qui signifie littéralement « capture de mouvements »). Inventée pendant les années 1980, la mocap connait un fulgurant essor depuis les années 2000 et continue encore à être de plus en plus utilisée. On distingue différents types de motion capture en fonction de la technologie utilisée pour l'acquisition des données :

- Optique
- Mécanique
- Magnétique

Dans le cadre de ce projet, nous avons exclusivement travaillé sur le type de motion capture le plus répandu : l'optique. Il consiste à placer de petites sphères réfléchissantes sur des personnages réels, libres de leurs mouvements. Le personnage est ensuite placé dans une salle spéciale équipée de toute une batterie de caméras (Cf. figure 1). Ce sont ces caméras, disposées de part et d'autre, qui permettront de distinguer, trianguler et d'enregistrer les positions successives que prendront tous ces marqueurs. Plus le nombre de caméras utilisées est grand et plus les données récupérées seront précises. En outre il est également possible d'accrocher une caméra supplémentaire en face du visage d'un acteur pour ne récupérer que ses expressions.

Figure 1 : Studio de motion capture. On constate que les actrices portent des combinaisons noires permettant de mieux faire ressortir les capteurs. Une peut distingue une rangée de caméra dans la partie supérieure de la photo

B. Capture d'un squelette

Le plus souvent, la motion capture sert à récupérer le mouvement d'un corps humain dans son intégralité pour en récupérer les mouvements du squelette. Le Conventionnal Gait Model est la convention la plus couramment utilisée dans les milieux médical et scientifique pour savoir où placer de façon pertinente les capteurs sur le corps humain (Cf. figure 2). Bien que Nexus soit fourni avec un plug-in permettant d'exporter facilement les données respectant ce modèle, nous avons décidé de seulement nous en inspirer sans le respecter strictement. Ce choix a été motivé par le fait que nous voulions expérimenter nous-même différentes choses avec les capteurs comme modifier leur nombre ou leur position.

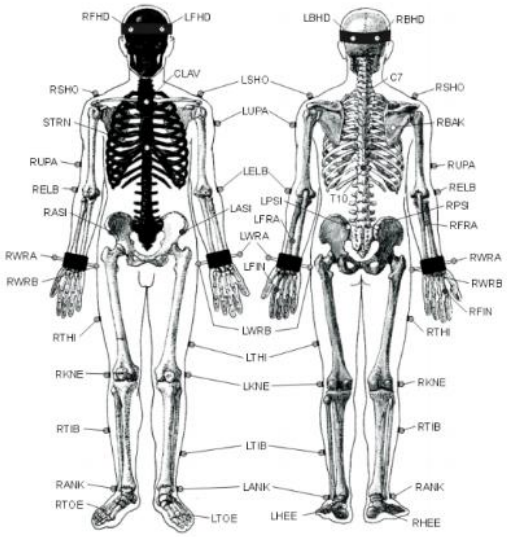


Figure 2 : Position des capteurs respectant le Conventionnal Gait Model

C. Capture d'un Visage

L'acquisition d'un visage par motion capture étant généralement plus difficile que celle d'un squelette,.il existe diverses techniques :

- La première consiste tout comme pour les squelettes à utiliser des marqueurs physiques que l'on place sur le visage d'un acteur. Cependant un visage est plus petit et doit être capturé de façon beaucoup plus précise qu'un squelette pour arriver à en restituer les différentes émotions. En outre il peut être plus difficile, voire carrément impossible de positionner des capteurs à certains endroits du visage (comme les yeux, la bouche…). Il est donc nécessaire d'utiliser un assez grand nombre de plus petits capteurs pour que la méthode reste efficace. Un autre inconvénient de cette technique repose sur la façon dont vont être positionnées les caméras. En effet celles-ci sont généralement positionnées de manière circulaire autour d'une zone afin de pouvoir capturer des squelettes. Or dans le cas d'un visage tous les capteurs vont être situés d'un seul et même côté rendant un certain nombre de caméras inutiles. Il faudrait donc idéalement adapter la position de nos caméras en fonction du type de capture que l'on souhaite réaliser, rendant ainsi impossible la capture en simultanée d'un visage et d'un squelette via un même dispositif. C'est pour cette raison que la capture de visage se fait le plus souvent à l'aide d'une caméra spécifique fixée en face du visage de l'acteur. Tout comme pour les squelettes avec Gait, la figure suivante (Cf. figure 3) illustre comment positionner les capteurs de façon à obtenir un résultat relativement précis.

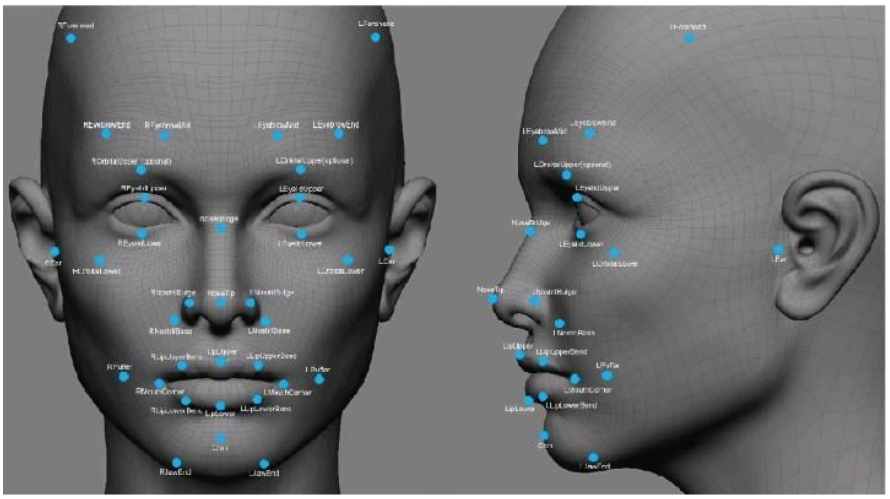


Figure 3 : Positionnement des capteurs sur le visage

- La seconde méthode consiste tout simplement à détecter directement un visage via une simple caméra (Cf. figure 4). Pour ce faire, différents algorithmes peuvent être utilisés. Cependant ce genre de techniques reste assez peu utilisé car il est très difficile de suivre les mouvements d'un visage sans aucun capteur. C'est pourquoi des techniques plus « hybrides » ont vu le jour et tentent de simuler de vrais marqueurs physiques par du maquillage ou de projetant de motifs sur le visage de l'acteur.

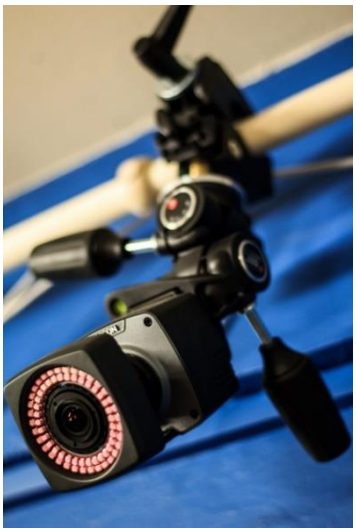
Figure 4 : Exemple de capture via une caméra frontale. Les marqueurs utilisés sont en fait du maquillage

Capture des données :

Pour mener à bien ce projet, nous avons eu le privilège de pouvoir utiliser la plateforme de motion capture AmigoCap, basée à l'école Centrale de Lyon. Celle-ci se compose de 6 caméras infrarouge (Cf. figure 5) à très hautes définitions (à la fois spatiale et temporelle). Nous avions également à notre disposition des combinaisons noires ainsi qu'une multitude de marqueurs réfléchissants (Cf. figure 6). L'acquisition des données a été réalisée par le logiciel Nexus de la société Vicom. C'est lui qui s'est occupé de trianguler la position de nos différents capteurs dans l'espace que nous avions préalablement calibré. Une fois tous les trous comblés et les points étiquetés, nous avions le choix parmi une liste de formats supportés par le logiciel pour exporter nos données.

Figure 6 : Capteur réfléchissant utilisé

Figure 5 : Camera VICON infrarouge



Au total, nous avons pu nous rendre 2 fois sur la site d'AmigoCap pour y réaliser des captures qui nous ont permis d'exporter plusieurs jeux de données. Nous n'avons malheureusement pas eu le temps de capturer de visage comme nous l'aurions souhaité au cours de ces séances. Cependant nous avons essayé de capturer divers squelettes (notamment avec plus au moins de points) ainsi que différents mouvements (marche/course/saut).

La principale difficulté rencontrée lors des séances de captures vient du fait que des points sont régulièrement perdus lors de l'acquisition (à cause d'obstacle comme les cheveux ou les bras pouvant gêner certaines caméras). Or lorsque plusieurs points disparaissent simultanément, le logiciel ne sait plus les différencier lorsque ceux-ci réapparaissent. Nous sommes donc obligées de « corriger » les données de façon manuelle avant de pouvoir les exporter au format souhaité. C'est cette tâche à la fois longue et fastidieuse qui nous a empêché de pouvoir capturer plus de mouvements. Une fois tous les points manquants récupérés de cette façon, une seconde étape consiste à tous les relier entre eux par des jointures (mais nous n'avons malheureusement pas trouvé comment les exporter).

Nous avons ensuite conçu notre outil de façon à pouvoir visualiser ces données. Celui-ci a été pensé de façon suffisamment générique pour théoriquement pouvoir afficher tout type de mouvements acquis via AmigoCap (y compris pour les captures de visage).

La seconde partie de notre projet consistait à créer un module d'export capable de corriger les imprécisions acquises lors de la capture. Après avoir réalisé notre première séance de capture nous avons pu constater que les données sont d'une infime précisions et n'ont aucunement besoin d'être débruité. C'est pourquoi nous avons tout simplement abonné cette partie du projet.

Développement de notre interface de visualisation :

Pour pouvoir visualiser les données que nous avons acquises, nous avons choisi de développer une interface web. Celle-ci fonctionne sur la plupart des navigateurs récents et ne nécessite le recours d'aucun serveur pour fonctionner correctement. Pour développer cet outil nous avons principalement utilisé HTML5 (à cause de la balise <canvas>) ainsi que la bibliothèque JavaScript Three.js. Cette dernière se base sur WebGL et nous donne accès à beaucoup de primitives de haut niveau, en particulier en ce qui concerne la création de la scène, la gestion de la caméra ainsi que le rendu de formes géométriques simples.

Au début de notre projet un premier prototype (Cf. figure 7) a été réalisé à partir de données trouvées sur internet afin de nous familiariser à la motion capture de façon générale. Ce premier prototype en 2D n'utilisait pas Three.js mais D3.js, une autre bibliothèque JavaScript. Nous avons cependant très vite abandonné D3.js au profit de Three.js car nous nous sommes rendu compte que celui-ci n'était finalement pas adapté à notre projet. En effet l'objectif de D3 est d'offrir une multitude d'outils permettant d'afficher de façon dynamique diverses sources de données. Or nous avions surtout besoin d'une bibliothèque de rendu en 3D ce qui correspond parfaitement à Three.

Ce prototype, très simple, permettait seulement d'afficher la décomposition d'un mouvement et d'en jouer l'animation. En outre, celui-ci était assez peu performant car lors de l'animation nous effacions et redessinions complètement la scène à chaque frame.

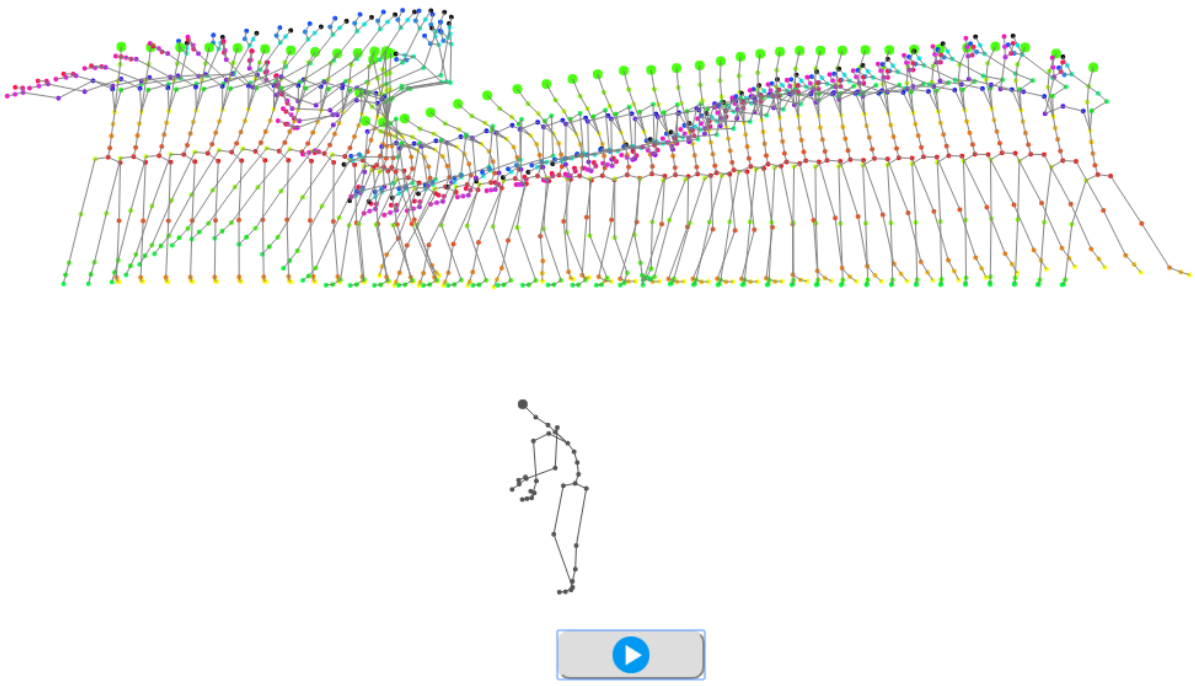


Figure 1 : Premier prototype réalisé sous D3.js

Plusieurs raisons nous ont donc motivé à choisir Three.js :

- Celui-ci est basé sur WebGL, technologie à la fois robuste et largement répandue depuis plusieurs années. La plupart des navigateurs l'implémentent nativement et son utilisation de la carte graphique lui assure de bonnes performances.
- Three.js propose de nombreuses primitives d'assez haut niveau ce qui nous a évité de devoir « réinventer la roue » et nous a permis de gagner beaucoup de temps.
- Le plaisir d'apprendre une nouvelle bibliothèque : Three.js est très bien documenté, la communauté est très active et de nombreux exemples sont disponibles sur le site officiel.

La création de la scène et la gestion de la caméra sont gérées directement par Three.js. Bien que ce dernier soit également capable de gérer par défaut les fichiers BVH, nous avons préféré utiliser le format .csv que nous trouvons beaucoup plus simple. Celui-ci commence par 5 lignes nous donnant diverses informations (comme le nom de nos différents points) que nous n'avons cependant pas exploitées. Les lignes suivantes correspondent toutes à une frame chacune et contiennent les coordonnées de nos différents points pour la frame en question. (Cf. Annexe n°1). Nous avons donc commencé par parser la totalité du fichier ligne par ligne avant de parser individuellement chacune de ces lignes.

Nous stockons ensuite toutes ces coordonnées dans un tableau multidimensionnel nommé positions afin de pouvoir accéder instantanément à n'importe quel frame. Ainsi positions[i][j] nous permet d'accéder aux coordonnées du $j^{ème}$ point de la $i^{ème}$ frame. Ces coordonnées sont sous stockées sous la forme d'objet JSON comme suit {x : 10, y : 20, z : 30}. Cependant nous avons pu constater au cours de nos captures qu'il est tout à fait possible que nos fichiers .csv ne commencent pas par la frame n°1 (s'il manque certains points au lancement de la capture par exemple). Nous avons donc choisi de ne plus faire correspondre le numéro de frame réel à l'index de notre tableau mais de compter nous-même (en commençant par 0) le nombre de frame lors du parsing . Ainsi l'exemple suivant nous permet d'assigner les coordonnées 10/20/30 au $3^{ème}$ point de la $101^{ème}$ frame : positions[100][2] = {x : 10, y : 20, z : 30}

Une seconde variable nommée points stocke nos différents meshs (dans notre cas des sphères) pour pouvoir se contenter de mettre à jour leur position plutôt que de devoir effacer puis redessiner complètement la scène à chaque frame. Comme nos différentes coordonnées n'ont pas de noms, il est indispensable de garder pour nos points exactement le même ordre que celui décrit dans le fichier .csv.

Enfin concernant les os nous avons gardé la même logique à savoir une variable skeleton stockant notre structure ainsi qu'une autre variable bones stockant quant à elle nos meshs (ici des cylindres). Bien que cela soit très certainement possible, nous n'avons pas trouvé le moyen d'exporter de fichier au format .ske lors de nos séances de capture. Nous avons donc été obligés d'écrire ces fichiers "à la main", il s'agit simplement de tableaux contenant les paires de points (c'est-à-dire d'autres tableaux avec exactement 2 éléments) correspondant à nos différents joints. Nous avons choisi cette structure car elle nous avait été présenté lors du projet comme étant un fichier exporté par le logiciel Nexus.

Comme nous n'avons pas trouvé comment directement exporter les jointures via Nexus, nous avons choisi de rendre les fichiers .csv et .ske indépendants. Ainsi il est donc possible de ne charger et visualiser que la position des points sans les jointures (Cf. figure 8), puis d'éventuellement les ajouter a posteriori (Cf. figure 9). En revanche il n'est évidemment pas possible de charger les jointures sans les points au préalable.

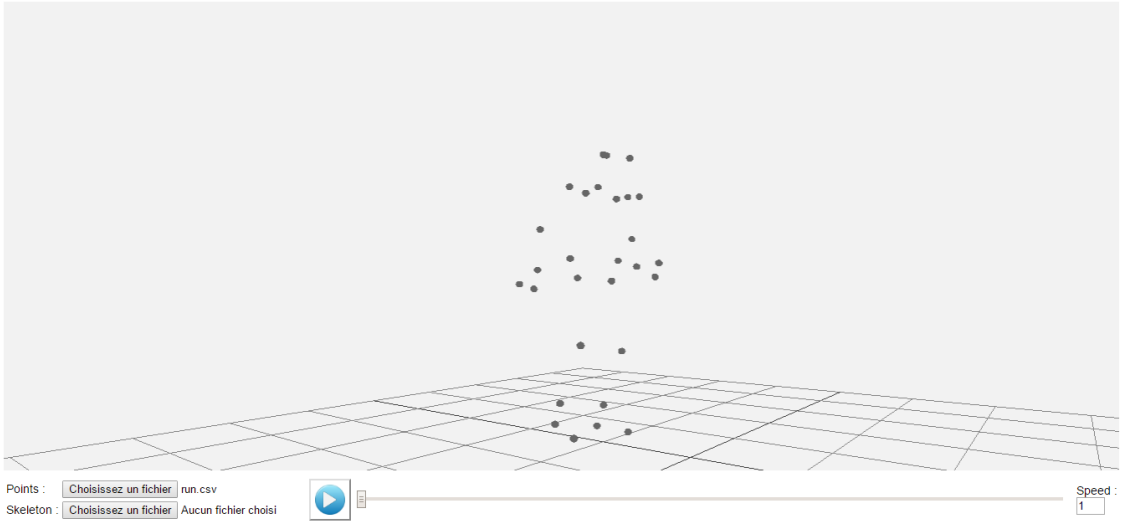


Figure 2 : Silhouette de la capture avec seulement les nœuds

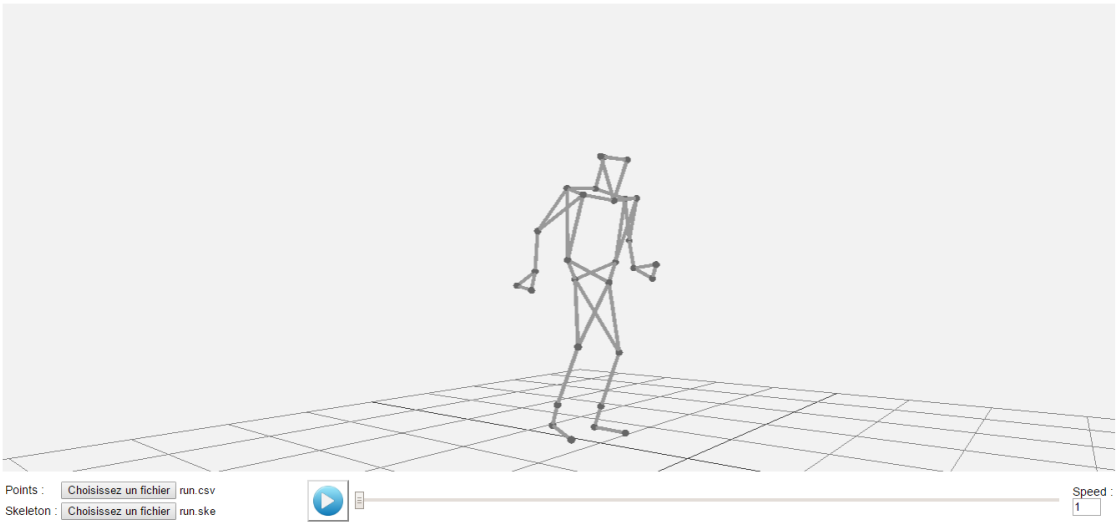


Figure 3 : Silhouette de la capture avec nœuds + jointures

Pour plus de commodité, un bouton start/pause, une barre de navigation ainsi qu'un régulateur de vitesse ont été ajoutés à l'interface.

L'ensemble du code source du projet ainsi que toute la documentation qui y est associée sont disponibles sur GitHub à l'adresse suivante : https://github.com/AmigoCap/Visual_MoCap

Conclusion :

Nous avons trouvé ce projet particulièrement intéressant et pensons qu'il a été très enrichissant pour nous à plusieurs égards :

- Tout d'abord il nous a initié au domaine de la motion capture dont nous avions beaucoup entendu parlé jusqu'alors sans jamais pouvoir réellement l'expérimenter.
- Il nous a permis de réaliser nos propres enregistrements et de manipuler du matériel sophistiqué, à la fois d'une extrême précision et très coûteux (+ 60000 €)
- Il nous a sensibilisé au travail de recherches préalables, et plus particulièrement dans le domaine de la recherche scientifique (lecture d'articles et de thèses).
- Il nous a permis d'intégrer une plateforme récente, ne demandant qu'à être améliorée (AmigoCap a été créé l'année dernière). C'est pourquoi nous avons pris un soin tout particulier à rendre notre code aussi simple, clair et commenté que possible. Nous espérons sincèrement que celui-ci sera utilisé, repris et maintenu par d'autres personnes dans le futur.
- Il nous a également appris à régulièrement rendre-compte de l'avancée de nos travaux (notamment via des entretiens réels ou par visio-conférence).
- Il nous a permis de faire le parallèle et d'approfondir certaines de nos connaissances acquises ce semestre lors de l'UE MIF37 : Animation d'images.
- Enfin, il nous a finalement tous les 2 conforté dans notre décision de poursuivre en master ID3D.

Nous avons donc grandement apprécié réaliser ce projet. Notre seul regret est finalement de ne pas avoir eu le temps de réaliser plus de captures (notamment du visage).

Annexes

Annexe N°1

Exemple de fichier .csv exporté par Nexus :

```
Trajectories
100
,,box:segment11,,,box:segment12,,,box:segment13,,,box:segment14,,,box:segment23,,,box:
Frame,Sub Frame,X,Y,Z,X,Y,Z,X,Y,Z,X,Y,Z,X,Y,Z,X,Y,Z,X,Y,Z
,,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm,mm
1,0,111.187,-68.5404,199.764,-70.0458,-69.5641,198.864,-66.3156,50.341,199.699,105.282
2,0,111.179,-68.548,199.764,-70.0314,-69.5388,198.877,-66.3101,50.3457,199.699,105.283
3,0,111.176,-68.5553,199.767,-70.0427,-69.5536,198.864,-66.3195,50.3504,199.684,105.26
4,0,111.171,-68.5545,199.76,-70.0301,-69.5544,198.857,-66.3151,50.3616,199.705,105.276
5,0,111.166,-68.5921,199.755,-70.047,-69.5479,198.88,-66.3195,50.3568,199.7,105.279,50
6,0,111.188,-68.5673,199.772,-70.0403,-69.5432,198.884,-66.3083,50.352,199.689,105.302
7,0,111.183,-68.5555,199.756,-70.0285,-69.5277,198.886,-66.3076,50.364,199.711,105.299
8,0,111.19,-68.5739,199.756,-70.0303,-69.5214,198.897,-66.3032,50.3817,199.703,105.303
9,0,111.189,-68.5664,199.765,-70.025,-69.5175,198.92,-66.3031,50.3523,199.708,105.309,
10,0,111.197,-68.5612,199.773,-70.0231,-69.5106,198.895,-66.303,50.3506,199.704,105.30
11,0,111.186,-68.5553,199.766,-70.0252,-69.5014,198.905,-66.2961,50.3807,199.721,105.3
12,0,111.185,-68.5879,199.761,-70.0357,-69.5028,198.91,-66.2951,50.3733,199.713,105.30
13,0,111.193,-68.5867,199.757,-70.0267,-69.5072,198.903,-66.2872,50.3685,199.724,105.2
14,0,111.203,-68.5754,199.769,-70.0146,-69.5093,198.939,-66.2936,50.369,199.716,105.30
15,0,111.201,-68.5851,199.761,-70.0359,-69.496,198.911,-66.2904,50.3655,199.714,105.32
16,0,111.194,-68.581,199.757,-70.0259,-69.4926,198.914,-66.2916,50.3605,199.713,105.32
17,0,111.193,-68.6081,199.758,-70.0008,-69.4654,198.896,-66.2886,50.377,199.729,105.33
18,0,111.189,-68.5642,199.771,-70.0154,-69.5061,198.928,-66.2786,50.3784,199.701,105.3
19,0,111.188,-68.59,199.763,-70.0161,-69.4931,198.905,-66.2929,50.3912,199.724,105.315
20,0,111.196,-68.5648,199.775,-70.0123,-69.5042,198.916,-66.2908,50.3753,199.734,105.3
21,0,111.2,-68.5896,199.771,-70.0107,-69.4972,198.917,-66.2878,50.3797,199.729,105.34,
22,0,111.189,-68.5792,199.76,-70.0222,-69.4837,198.931,-66.2915,50.3691,199.72,105.323
23,0,111.192,-68.5867,199.779,-70.0065,-69.4896,198.902,-66.2901,50.3847,199.728,105.3
24,0,111.194,-68.5639,199.764,-70.033,-69.4784,198.909,-66.282,50.3849,199.73,105.33,5
25,0,111.19,-68.5855,199.754,-70.0259,-69.5046,198.918,-66.2778,50.3698,199.722,105.32
26,0,111.203,-68.5814,199.773,-70.0041,-69.5145,198.913,-66.2969,50.3778,199.712,105.3
27,0,111.201,-68.5868,199.765,-70.0098,-69.5001,198.922,-66.2934,50.3879,199.718,105.3
28,0,111.193,-68.5807,199.774,-70.048,-69.5117,198.902,-66.2753,50.3689,199.721,105.31
29,0,111.194,-68.5907,199.783,-70.0262,-69.5051,198.923,-66.2774,50.3771,199.708,105.3
30,0,111.191,-68.5618,199.753,-70.0169,-69.5081,198.909,-66.2952,50.3794,199.728,105.3
31,0,111.194,-68.577,199.75,-70.0219,-69.4984,198.901,-66.2944,50.3767,199.717,105.328
32,0,111.199,-68.5827,199.764,-70.0158,-69.5004,198.922,-66.2983,50.3803,199.713,105.3
33,0,111.197,-68.5899,199.762,-70.0164,-69.5021,198.904,-66.2842,50.384,199.723,105.32
34,0,111.194,-68.6128,199.778,-70.0042,-69.4946,198.923,-66.2992,50.3634,199.737,105.3
35,0,111.2,-68.5907,199.768,-70.0145,-69.495,198.91,-66.2811,50.3808,199.725,105.333,5
36,0,111.198,-68.5816,199.777,-70.0191,-69.4916,198.917,-66.2918,50.3866,199.732,105.3
37,0,111.188,-68.5919,199.768,-70.0271,-69.4904,198.899,-66.3005,50.3862,199.727,105.3
38,0,111.187,-68.5894,199.77,-70.004,-69.4807,198.908,-66.281,50.3694,199.731,105.326,
39,0,111.194,-68.5862,199.749,-70.02,-69.5087,198.908,-66.291,50.3892,199.727,105.315,
40,0,111.18,-68.5878,199.745,-70.0188,-69.5111,198.914,-66.2942,50.3627,199.713,105.31
41,0,111.194,-68.5906,199.753,-70.0201,-69.5052,198.912,-66.3018,50.3719,199.711,105.3
42,0,111.203,-68.5958,199.756,-70.0006,-69.4958,198.911,-66.2854,50.3861,199.73,105.32
43,0,111.213,-68.5815,199.759,-70.0182,-69.5056,198.908,-66.2809,50.374,199.733,105.32
44,0,111.192,-68.6013,199.782,-70.0209,-69.5043,198.918,-66.2859,50.3791,199.729,105.3
45,0,111.198,-68.6086,199.761,-70.0272,-69.4969,198.927,-66.2959,50.3698,199.724,105.3
46,0,111.181,-68.5698,199.764,-70.0183,-69.4936,198.906,-66.2985,50.3787,199.703,105.3
47,0,111.197,-68.6038,199.766,-70.0222,-69.4928,198.909,-66.2899,50.3731,199.72,105.32
48,0,111.206,-68.5949,199.769,-70.0097,-69.5141,198.933,-66.2965,50.3721,199.728,105.3
49,0,111.195,-68.5828,199.755,-70.0137,-69.5114,198.919,-66.2872,50.3859,199.716,105.3
```

Annexe N°2

Exemple de fichier .ske que nous avons utilisé :

```
[[0, 1], [2, 3], [1, 5], [2, 4], [4, 7], [5, 6], [6, 8], [7, 8], [8, 9], [9, 10],
```